SL1 PowerFlow Platform

Version 3.2.0

Table of Contents

Introduction to SL1 PowerFlow and the PowerFlow Builder 20
What is SL1 PowerFlow? 21
What is a Step? 22
Using Steps in a PowerFlow Application 23
Using Input Parameters to Configure a Step 24
Sharing Data Between Steps 25
Types of Steps 25
What is a PowerFlow Application? 26
What is a Configuration Object? 27
What is the SL1 PowerFlow Builder? 27
Elements of the PowerFlow User Interface 29
Logging In and Out of the PowerFlow User Interface 29
PowerFlow Pages 31
Additional Navigation 31
Using the API or Command Line Tool to Create PowerFlow Components 32
Installing and Configuring SL1 PowerFlow 33
PowerFlow Architecture 35
PowerFlow Container Architecture 35
Integration Workflow 36
High-Availability, Off-site Backup, and Proxy Architecture 36
Reviewing Your Deployment Architecture 38
System Requirements 39
Ports 39
Additional Considerations 39
Hardened Operating System 40
Additional Prerequisites for PowerFlow 41
Installing PowerFlow 41
Installing PowerFlow for the First Time 41
Upgrading an Existing PowerFlow System 42
Installing PowerFlow via ISO 42
Locating the ISO Image 42

Installing from the ISO Image 42
Troubleshooting the ISO Installation 45
Installing PowerFlow via RPM to a Cloud-based Environment 46
Considerations for the RPM Installation 46
Locating the RPM file 46
Installing from the RPM File 46
Troubleshooting a Cloud Deployment of PowerFlow 54
Installing PowerFlow on AWS 54
What are the ScienceLogic AMIs? 54
Getting the PowerFlow AMI 55
Launching the New Instance 55
Accessing the Appliance Using SSH 58
Gathering Information Required for Accessing the Appliance Using SSH 58
Configuring SSH 59
Deploying the PowerFlow Application 59
Additional Configuration Steps 60
Converting PowerFlow to Oracle Linux 8 (OL8) 61
Upgrade Options for Converting from PowerFlow 2.x (OL7) to PowerFlow 3.x or Later (OL8) 61
Upgrade Paths Based on PowerFlow Environments 61
Back Up, Re-install, and Restore Your PowerFlow System 62
Upgrading to Couchbase Version 6.6.0 63
PowerFlow Supported Upgrade Paths 63
Logs Buckets 63
Downgrading 64
Upgrading from PowerFlow 3.x to the latest 3.x Version 64
Deploying PowerFlow as a MUD System (Optional) 64
Considerations for Upgrading from PowerFlow 3.x 64
Option 1: Increase the size of the isvg-root(/) filesystem 65
Option 2: Remove the Old PowerFlow Images from the /opt/iservices/images directory 65
Pre-Upgrade Steps 65
Locating the RPM or ISO File for Upgrading 66
Upgrading OS Packages 66

Upgrading OS Packages (for Offline Deployments Only) 67
Upgrading from Version 3.x.x to 3.2.0 67
Single-node Upgrade 68
Cluster Upgrade with Short Downtime 69
Rolling Cluster Upgrade with No Downtime 71
Validating the PowerFlow System Post-Upgrade 72
Troubleshooting Upgrade Issues 72
After upgrading, the syncpacks_steprunner service fails to run 73
SyncPack virtual environments were not recreated 73
SyncPacks cannot be installed after upgrading from PowerFlow version 3.0.0 74
The PowerFlow user interface displays an unauthorized user error 74
Licensing PowerFlow 74
Licensing a PowerFlow System 75
Licensing Solution Types 77
Configuring PowerFlow Services 78
Applying User-Specific Configurations 78
Updating the docker-compose-override File 79
Adding User-Specific Configurations 79
Using Jinja2 in the compose-override File 81
Configuring a Proxy Server 82
Changing the PowerFlow System Password 84
Updating the PowerFlow Administrator (isadmin) user password 84
Updating the PowerFlow Administrator (isadmin) User Password with the ipasswd Script 86
Configuring Security Settings 86
Changing the HTTPS Certificate 86
Using Password and Encryption Key Security 88
Configuring Additional Elements of PowerFlow 89
Setting a Hard Memory Limit in Docker 89
Setting a Soft Memory Limit in the Worker Environment 89
PowerFlow Task Processing and Memory Handling 90
Background 90
CPU and Memory Requirements for PowerFlow 90

Recommended Memory Allocation of PowerFlow Nodes 91
SaaS Deployments 91
Example Code: docker-compose for SaaS 91
16 GB Deployments 92
Example Code: docker-compose for 16 GB Deployments 92
32 GB Deployments 93
Example Code: docker-compose for 32 GB Deployments 93
64 GB Deployments 94
Example Code: docker-compose for 64 GB Deployments 94
128 GB Deployments 97
Example Code: docker-compose for 128 GB Deployments 97
Identifying Oomkills 99
Common Causes of High Memory and Oomkills 99
Questions to Ask when Experiencing Oomkills 99
Avoiding Oomkills 100
Avoiding Node Exhaustion 100
Best Practices for Running PowerFlow with Production Workloads 101
Avoid Debug Logging for Large-scale Runs 101
Additional Queues Might be Needed for Large-scale Runs 101
Avoid Running Large-scale Syncs Simultaneously 102
PowerFlow Management Endpoints 102
Flower API 102
Couchbase API 103
RabbitMQ 104
Docker Statistics 104
Using the SL1 PowerFlow Control Tower Page 106
What is the PowerFlow Control Tower? 107
The System Health Widget 107
Configuring the System Health Widget 107
Configuring the "PowerFlow Control Tower HealthCheck" Application to Gather pfctl Data 109
Using the System Health Widget 112
The Favorite Applications Widget 113

Contents of the Favorite Applications Widget 114
Using the Favorite Applications Widget 115
The Workflow Health and Interconnectivity Widget 115
Configuring the Workflow Health and Interconnectivity Widget 116
Using the Workflow Health and Interconnectivity Widget 117
The All Tasks, Workers, and Applications Widgets 118
Managing SyncPacks 119
What is a SyncPack? 120
Viewing the List of SyncPacks 121
Searching for a SyncPack 121
Viewing a Detail Page for a SyncPack 121
Using the Actions Button to Manage SyncPacks 122
Importing and Installing a SyncPack 123
Locating and Downloading a SyncPack 123
Importing a SyncPack 124
Activating and Installing a SyncPack 125
Locating and Importing Dependencies for a SyncPack 125
Considerations for Custom Syncpacks with PowerFlow 3.1.0 and Later 126
Updating Custom Syncpacks to Work with the New Couchbase SDK 126
Couchbase Locking Method 126
Couchbase N1QL queries 126
Couchbase Queries Metrics (Use only to get metrics) 127
Couchbase Exception 127
Default SyncPacks 128
Base Steps SyncPack 128
Flow Control SyncPack 128
System Utils SyncPack 128
Uploading Custom Dependencies to the PyPI Server with the iscli Tool 129
Managing SL1 PowerFlow Applications 130
Viewing the List of PowerFlow Applications 131
Elements of an Application Page 133
Buttons 134

Status Messages 135
Step Pane 136
Creating a Basic PowerFlow Application 136
Working with Flow Control Operators 139
Creating an Application with a Condition Operator 139
Creating an Application with a Transform Operator 143
Creating an Application that Uses a Trigger Application Operator 149
Parameters Table 152
Editing a PowerFlow Application 155
Editing Mappings in a PowerFlow Application 156
Enabling Run Book Automation Queue Retries 157
Requirements 157
PowerFlow Applications 157
Configuration Object 157
SL1 Action Type 158
Enabling RBA Queue Retries 158
Creating a Step 161
Defining Retry Options for a Step 161
Aligning a Configuration Object with an Application 162
Running a PowerFlow Application 164
Viewing Previous Runs of an Application with the Timeline 165
Scheduling a PowerFlow Application 168
Backing up and Restoring PowerFlow Data 170
Creating a Backup 171
Restoring a Backup 175
Restoring a Backup using the Command-line Interface 177
Managing Configuration Objects 179
What is a Configuration Object? 180
Viewing the List of Configuration Objects 180
Creating a Configuration Object 182
Working with Application Variables for Configuration Objects 184
Edit Configuration Button 184

Available Configuration Values Pop-up 184
Promote Step Variable Option 185
Editing a Configuration Object 187
Downloading and Importing a Configuration Object 187
Generating and Viewing Reports for SL1 PowerFlow Applications 188
Viewing the List of Reports in PowerFlow 189
Bulk Downloading Reports in PowerFlow 190
PowerFlow Platform Reports 191
The PowerFlow System Diagnostics Report 191
The Read SL1 RBA Queue and Retry PowerFlow Applications Report 193
SyncPack Reports 194
ServiceNow CMDB SyncPack Reports 194
Creating and Using API Keys in SL1 PowerFlow 195
Using API Keys 196
Creating an API Key 197
Authenticating with an API Key 198
Removing an API Key 198
Managing Users in SL1 PowerFlow 199
Configuring Authentication with PowerFlow 200
User Interface Login Administrator User (Default) 201
Basic Authentication Using a REST Administrator User (Default) 202
User Interface Login Using a Third-party Authentication Provider 202
Code Example: isconfig.yml file with an Active Directory authentication provider 203
OAuth Client Authentication Using a Third-party Provider 205
Basic Authentication Lockout Removal 205
Common Access Card (CAC) Authentication 206
Applying CAC Authorization 206
Adding CRL to CAC Authentication 206
CAC Authentication with LDAP 207
Environment Expectations 207
Add LDAP to CAC Query 207
CAC Authentication with LDAP and SAN 210

API Key Authentication 210
Role-based Access Control (RBAC) Configuration 210
Assigning a Role to a Specific User 210
Assigning Roles to a Specific User Group 210
Viewing User and Group Information 211
Changing Roles and Permissions 211
Configuring Authentication Settings in PowerFlow 211
User Groups, Roles, and Permissions 212
Creating a User Group in PowerFlow 213
Managing User Sessions 214
Enabling Session Management 215
Authentication and Authorization for Services Used by PowerFlow 216
Couchbase 216
RabbitMQ 217
Viewing Logs in SL1 PowerFlow 218
Logging Data in PowerFlow 219
Local Logging 219
Remote Logging 219
Viewing Logs in Docker 219
Logging Configuration 220
PowerFlow Log Files 221
Logs for the gui Service 221
Logs for the api Service 221
Logs for the rabbitmq Service 221
Working with Log Files 221
Accessing Docker Log Files 222
Accessing Local File System Logs 222
Understanding the Contents of Log Files 222
Managing journald Settings 223
Viewing the Step Logs and Step Data for a PowerFlow Application 224
Removing Logs on a Regular Schedule 225
Using the powerflowcontrol (pfctl) Command-line Utility 226

What is the powerflowcontrol (pfctl) Utility? 227
User Requirements for using the powerflowcontrol (pfctl) utility 228
Installing the powerflowcontrol (pfctl) utility 228
Getting Help with the powerflowcontrol (pfctl) utility 229
healthcheck and autoheal 229
healthcheck 230
Additional Features with the healthcheck Action 230
autoheal 230
Example Output 231
Using powerflowcontrol healthcheck on the docker-compose file 233
autocluster 233
apply_<n>GB_override, verify_<n>GB_override 234
check_dex_connectivity 235
check_docker_service_update_status 235
check_redis_maxmemory, fix_redis_maxmemory 235
logcollect 236
logservicescollect 237
open_firewall_ports 237
Increasing the PowerFlow Docker Swarm Heartbeat in Cluster Environments 237
update_swarm_heartbeat_period 237
check_swarm_heartbeat_period 237
password 238
Encrypting a PowerFlow Password 238
Changing the isadmin User Password 238
Disabling TLS Verification 239
Using SL1 to Monitor SL1 PowerFlow 240
Monitoring PowerFlow 241
Configuring the Docker PowerPack 242
Configuring the ScienceLogic: PowerFlow PowerPack 244
Configuring the PowerPack 245
Events Generated by the PowerPack 246
Stability of the PowerFlow Platform 247

What makes up a healthy SL1 system? 247
What makes up a healthy PowerFlow system? 248
Troubleshooting SL1 PowerFlow 249
Initial Troubleshooting Steps 250
SL1 PowerFlow 250
ServiceNow 250
Resources for Troubleshooting 250
Useful PowerFlow Ports 250
powerflowcontrol healthcheck and autoheal actions 251
Helpful Docker Commands 251
Viewing Container Versions and Status 251
Restarting a Service 251
Stopping all PowerFlow Services 251
Restarting Docker 252
Viewing Logs for a Specific Service 252
Clearing RabbitMQ Volume 252
Viewing the Process Status of All Services 253
Deploying Services from a Defined Docker Compose File 254
Dynamically Scaling for More Workers 254
Completely Removing Services from Running 254
Helpful Couchbase Commands 254
Checking the Couchbase Cache to Ensure an SL1 Device ID is Linked to a ServiceNow Sys ID 254
Clearing the Internal PowerFlow Cache 255
Clearing the Cache using the Command-Line Interface 255
Accessing Couchbase with the Command-line Interface 256
Exposing Couchbase Secondary Nodes User Interfaces 256
Temporarily Exposing Couchbase and RabbitMQ User Interfaces for Troubleshooting 257
Disabling Dex Authentication 257
Temporarily Exposing Couchbase Secondary Nodes User Interface for Troubleshooting 258
Useful API Commands 259
Getting PowerFlow Applications from the PowerFlow API 259
Creating and Retrieving Schedules with the PowerFlow API 259

Diagnosis Tools 260
Identifying Why a Service or Container Failed 261
Step 1: Obtain the ID of the failed container for the service 261
Step 2: Check for any error messages or logs indicating an error 262
Step 3: Check for out of memory events 262
Troubleshooting a Cloud Deployment of PowerFlow 262
Identifying Why a PowerFlow Application Failed 263
Determining Where an Application Failed 263
Retrieving Additional Debug Information (Debug Mode) 263
Troubleshooting Clustering and Node Failover 265
After a failover, Couchbase or the PowerFlow user interface are not available 265
After a cluster or node failover, PowerFlow will not start 266
I get a 502 error when I try to log in using the load balancer IP address 267
After a node goes down, the SyncPacks page does not display the expected content 267
After a node goes down, I cannot access the db port for that instance of Couchbase :8091 directly 268
Couchbase fails to properly initialize or keeps trying to initialize 268
Frequently Asked Questions 268
What is the first thing I should do when I have an issue with PowerFlow? 268
Can the steprunners_syncpack service can be limited to just workers? 269
What is the difference between the steprunner_syncpacks and the steprunner services? 269
What is the minimal image required for workers? 269
If the GUI server is constrained to use only the manager nodes, do the worker nodes need to have their isconfig.yml file updated with the correct HOST value? 269
Can I unload unwanted images from a worker node? 269
If I dedicated workers to one SL1 stack, how are jobs configured to run only on those workers? 269
Approximately how much data is sent between distributed PowerFlow nodes? 269
Why can't I find a SyncPack on the SyncPacks page? 270
Why can't I see or upload a SyncPack? 270
Why do I get a "Connection error" message when I try to install the System Utils SyncPack? 270
How can I optimize workers, queues, and tasks? 271
Why do I get a "Connection refused" error when trying to communicate with Couchbase? 274
Why do I have client-side timeouts when communicating with Couchbase? 274

What should I do if the Couchbase disk is full, the indexer is failing, and the database is unusable? ..275
What causes a Task Soft Timeout? ..276
How do I address an "Error when connecting to DB Host" message when access is denied to user "root"? ..276
How do I identify and fix a deadlocked state? ..277
How can I point the "latest" container to my latest available images for PowerFlow? ..281
Why does the "latest" tag not exist after the initial ISO installation? ..281
How do I address permissions errors with SyncPack virtual environments? ..281
How do I address intermittent user access when using single sign-on? ..282
How do I keep from losing incidents or events if my PowerFlow system is down? ..282
How do I restore an offline backup of my PowerFlow system? ..282
What do I do if I get a Code 500 Error when I try to access the PowerFlow user interface? ..283
What should I do if I get a 500 Error? ..284
What are some common examples of using the iscli tool? ..284
How do I view a specific run of an application in PowerFlow? ..285
Why am I getting an "ordinal not in range" step error? ..285
How do I clear a backlog of Celery tasks in Flower? ..285
Why does traffic from specific subnets not get a response from PowerFlow? ..286
What should I do if the number of tasks listed in the dashboards is not accurate? ..287
Why do I get "context deadline exceeded due to node exhaustion" when checking docker journalctl logs? ..287
Why do I get the following error when updating the PowerFlow administrator user password (isadmin)? ..287
Why is the Monitor tab for Flower no longer visible? ..288
API Endpoints in SL1 PowerFlow ..289
Interacting with the API ..290
Available Endpoints ..290
POST ..290
Querying for the State of a PowerFlow Application ..291
GET ..291
DELETE ..292
Configuring the SL1 PowerFlow System for High Availability ..294
Types of High Availability Deployments for PowerFlow ..295

Standard Single-node Deployment (1 Node) 296
Requirements 296
Risks 296
Configuration 296
Standard Three-node Cluster (3 Nodes) 297
Requirements 297
Risks 297
Mitigating Risks 298
Configuration 298
3+ Node Cluster with Separate Workers (4 or More Nodes) 301
Requirements 302
Worker Node Sizing 302
Risks 303
Mitigating Risks 303
Configuration 303
3+ Node Cluster with Separate Workers and Drained Manager Nodes (6 or More Nodes) 304
Requirements 304
Risks 304
Configuration 305
Additional Deployment Options 305
Cross-Data Center Swarm Configuration 305
Additional Notes 306
Requirements Overview 306
Docker Swarm Requirements for High Availability 307
Couchbase Database Requirements for High Availability 308
RabbitMQ Clustering and Persistence for High Availability 308
RabbitMQ Option 1: Persisting Queue to Disk on a Single Node (Default Configuration) 308
RabbitMQ Option 2: Clustering Nodes with Persistent Queues on Each Node 309
Example Code: docker-compose Definition of Two Clustered Rabbit Services 310
Checking the Status of a RabbitMQ Cluster 310
Preparing the PowerFlow System for High Availability 311
Troubleshooting Ports and Protocols 312

Configuring Clustering and High Availability 312
Automating the Configuration of a Three-Node Cluster 313
Configuring Docker Swarm 314
Configuring the Couchbase Database 315
Code Example: docker-compose-override.yml 318
Scaling iservices_contentapi 322
Single Manager Failure - Automatic Failover 322
Manual Failover 323
Initiating Manual Failover 323
Recovering a Docker Swarm Node 326
Restoring a Couchbase Node 326
Restoring RabbitMQ 327
Additional Configuration Information 328
Load Balancer Recommended Settings 328
Configurations to Improve Load Balancer Compatibility 328
Recommended Load Balancer Modes 328
Recommended HealthCheck Endpoints 329
PowerFlow 2.5.0 or later 329
PowerFlow 2.4.1 329
cURL Commands 329
Optimization Settings to Improve RabbitMQ Reclustering 329
Optimization Settings to Improve Performance of Large-Scale Clusters 330
Exposing Additional Couchbase Cluster Node Management Interfaces overTLS 331
Restricting the Number of Replicas 333
HAProxy Configuration (Optional) 334
Known Issues 335
Docker container on last swarm node cannot communicate with other swarm nodes 335
Couchbase service does not start, remains at nc -z localhost 336
Couchbase-worker fails to connect to master 336
Couchbase database stops unexpectedly and the disk is full 336
Couchbase rebalance fails with "Rebalance exited" error 336
When setting up a three-node High Availability Couchbase cluster, the second node does not 336

appear
The PowerFlow user interface fails to start after a manual failover of the swarm node 337
The PowerFlow user interface returns 504 errors 337
NTP should be used, and all node times should be in sync 337
Example Logs from Flower 337
Configuring the SL1 PowerFlow System for Multi-tenant Environments 338
Quick Start Checklist for Deployment 339
Deployment 339
Core Service Nodes 339
Requirements 340
Configuring Core Service Nodes 340
Critical Elements to Monitor on Core Nodes 340
Worker Service Nodes 340
Requirements 341
Event Sync Throughput Node Sizing 341
Test Environment and Scenario 341
Configuring the Worker Node 341
Initial Worker Node Deployment Settings 342
Worker Failover Considerations and Additional Sizing 342
Knowing When More Resources are Necessary for a Worker 342
Keeping a Worker Node on Standby for Excess Load Distribution 342
Critical Elements to Monitor in a Steprunner 343
Advanced RabbitMQ Administration and Maintenance 343
Using an External RabbitMQ Instance 343
Setting a User other than Guest for Queue Connections 343
Configuring the Broker (Queue) URL 343
Creating Specific Queues for Customers 344
Create the Configuration Object 344
Label the Worker Node Specific to the Customer 344
Creating a Node Label 344
Placing a Service on a Labeled Node 344
Creating a Queue Dedicated to a Specific Application or Customer 345

Add Workers for the New Queues 345
Code Example: docker-compose entries for new steprunners 347
Adding a PowerFlow Application to a Specific Queue 350
Create Application Schedules and Automation Settings to Utilize Separate Queues 350
Scheduling an Application with a Specific Queue and Configuration 351
Configuring Applications to Utilize a Specific Queue and Configuration 351
PowerFlow Queue FAQs 351
What is RabbitMQ, and what messages are placed in it? 351
What does it mean when the queue reports a high message count? 352
When should I be concerned about a high message count? 352
How can I tell what is currently in queue to be processed? 352
How can I tell what caused the queue backlog? 352
What do I do if the high message count was caused by over-scheduling? 353
What do I do if the high message count was caused by an SL1 event flood? 353
How can I clear messages from the queue? 354
Why are PowerFlow applications still showing as "Pending" after I cleared the queue? 354
Why are messages stuck in the broadcast queue in RabbitMQ? 354
Failure Scenarios 355
Worker Containers 355
API 355
Couchbase 356
RabbitMQ 357
PowerFlow User Interface 357
Redis 358
Known Issue for Groups of Containers 358
Examples and Reference 359
Code Example: A Configuration Object 359
Code Example: A Schedule Configuration Object 363
Test Cases 373
Load Throughput Test Cases 373
Failure Test Cases 373
Backup Considerations 374

What to Back Up 374
Fall Back and Restore to a Disaster Recovery (Passive) System 375
Resiliency Considerations 375
The RabbitMQ Split-brain Handling Strategy (SL1 Default Set to Autoheal) 375
ScienceLogic Policy Recommendation 376
Changing the RabbitMQ Default Split-brain Handling Policy 376
Using Drained Managers to Maintain Swarm Health 376
Updating the PowerFlow Cluster with Little to No Downtime 377
Updating Offline (No Connection to a Docker Registry) 377
Updating Online (All Nodes Have a Connection to a Docker Registry) 377
Additional Sizing Considerations 377
Sizing for Couchbase Services 377
Sizing for RabbitMQ Services 377
Sizing for Redis Services 378
Sizing for contentapi Services 378
Sizing for the GUI Service 378
Sizing for Workers: Scheduler, Steprunner, Flower 378
Scaling the PowerFlow Devpi Server 378
When to Add a New Devpi Server Replica to the PowerFlow Stack 379
Adding a New Devpi Server Replica to the Stack 379
Code Example: docker-compose-override file 379
Considerations 380
Configuring Steprunners to Consume Data from Devpi Server Replicas 380
Additional Considerations 381
Node Placement Considerations 382
Preventing a Known Issue: Place contentapi and Redis services in the Same Physical Location 382
Common Problems, Symptoms, and Solutions 383
Common Resolution Explanations 389
Elect a New Swarm Leader 389
Recreate RabbitMQ Queues and Exchanges 389
Resynchronize RabbitMQ Queues 390
Identify the Cause of a Service not Deploying 390

Repair Couchbase Indexes 391
Add a Broken Couchbase Node Back into the Cluster 392
Restore Couchbase Manually 392
PowerFlow Multi-tenant Upgrade Process 393
Performing Environment Checks Before Upgrading 393
Installing the PowerFlow RPM 394
Compare docker-compose file changes and resolve differences 394
Make containers available to systems 395
Perform the Upgrade 395
Upgrade Redis, Scheduler, and Flower 396
Code Example: Image definition of this upgrade group 396
Redis Version 397
Upgrade Core Services (RabbitMQ and Couchbase) 397
Rabbit/Couchbase Versions 398
Update Actions (assuming three core nodes) 398
First node Couchbase update considerations 398
Code Example: docker-compose with images and JOIN_ON for updating the first node 399
Update second and third node services 400
Update the GUI 400
Update Workers and contentapi 400
Code Example: docker-compose definition with one of two worker nodes and contentapi updated: 401

Chapter 1

Introduction to SL1 PowerFlow and the PowerFlow Builder

Overview

SL1 PowerFlow provides a generic platform for integrations between SL1 and third-party applications, such as ServiceNow, Restorepoint, xMatters, Opsgenie, or Cherwell Service Management. The PowerFlow platform sits between SL1 and the third-party application, where it handles the flow of data.

From the PowerFlow user interface, you can use the PowerFlow builder to create complex workflow automations with logical branching, using drag-and-drop components.

This chapter covers the following topics:

What is SL1 PowerFlow? 21

What is a Step? 22

What is a PowerFlow Application? 26

What is a Configuration Object? 27

What is the SL1 PowerFlow Builder? 27

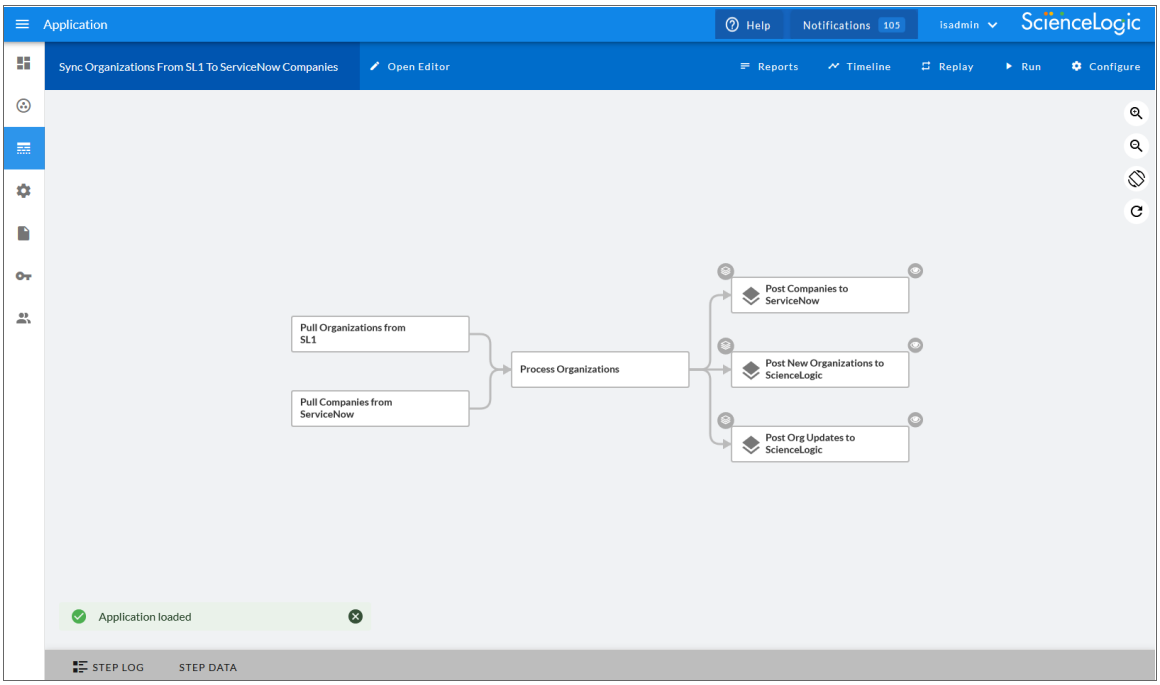
Elements of the PowerFlow User Interface 29

Using the API or Command Line Tool to Create PowerFlow Components 32

What is SL1 PowerFlow?

SL1 PowerFlow enables intelligent, bi-directional integration between SL1 and third-party applications to promote a unified management ecosystem. PowerFlow contains default workflows that let users translate and share data between SL1 and third-party applications, and it also allows the development of standardized, reusable snippets called "steps" that non-developers can use to create integration workflows without writing code. In addition, PowerFlow is designed to provide high availability and scalability.

The following image shows an example of a PowerFlow application workflow and its steps in the PowerFlow user interface:



The key elements of the PowerFlow user interface include the following:

- ***SyncPacks***. A SyncPack contains all the code and logic needed to perform integrations on the PowerFlow platform. You can access the latest steps, applications, and configuration objects for a third-party application (such as ServiceNow, Cherwell, or Restorepoint) by downloading the most recent SyncPack for that application. You can download SyncPacks from the **PowerPacks & SyncPacks** page at the ScienceLogic Support Site at https://support.sciencelogic.com/s/. You can access all SyncPacks that have been uploaded to your PowerFlow system on the **SyncPacks** page in the PowerFlow user interface ( ).

  A SyncPack can include the following items:

  - ***Steps***. A step is the basic building block in PowerFlow. A step is a generic Python class that performs a single action, such as pulling data from SL1 or a third-party application. In the image above, the steps display as part of the flowchart in the main viewing pane as well as the **Steps Registry** pane. You can access all steps by using the PowerFlow builder on the **Applications** page ( ).

    For more information, see *What is a Step?*

  - ***Applications***. A PowerFlow application or workflow is a JSON object that includes all of the information required for executing an integration on the PowerFlow platform. In the image above, the group of connected steps in the large pane make up the "Sync Organizations from SL1 to ServiceNow Companies" application. You can access all applications on the **Applications** page ( ), and you can create new applications using the SL1 PowerFlow builder.

    For more information, see *What is a PowerFlow Application?*

  - ***Configuration Objects***. A configuration object is a standalone JSON file that contains a set of configuration variables used as input for an application. Configuration objects can include variables like hostname, user name, password, or other credential information. You can access all configuration objects on the **Configurations** page ( ).

    For more information, see *What is a Configuration Object?*

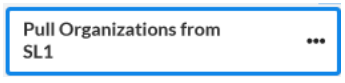
What is a Step?

In PowerFlow, a ***step*** is a generic Python class that performs a single action, such as gathering data about an organization.

Steps can accept zero or many input parameters or data from previous steps, and steps can specify output to be used by other steps. The input parameters are configurable variables and values used during execution.

You can use existing steps to create your own workflows, and you can re-use steps in more than one workflow. When these steps are combined in an application, they provide a workflow that satisfies a business requirement. All Python step code should be Python 3.7 or later.

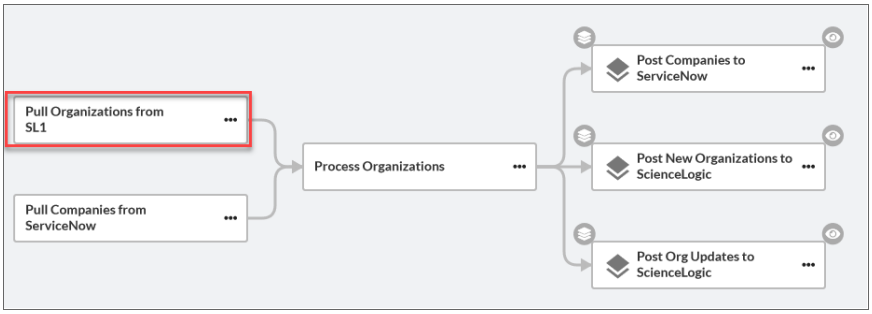
The following image shows a step from the PowerFlow user interface:



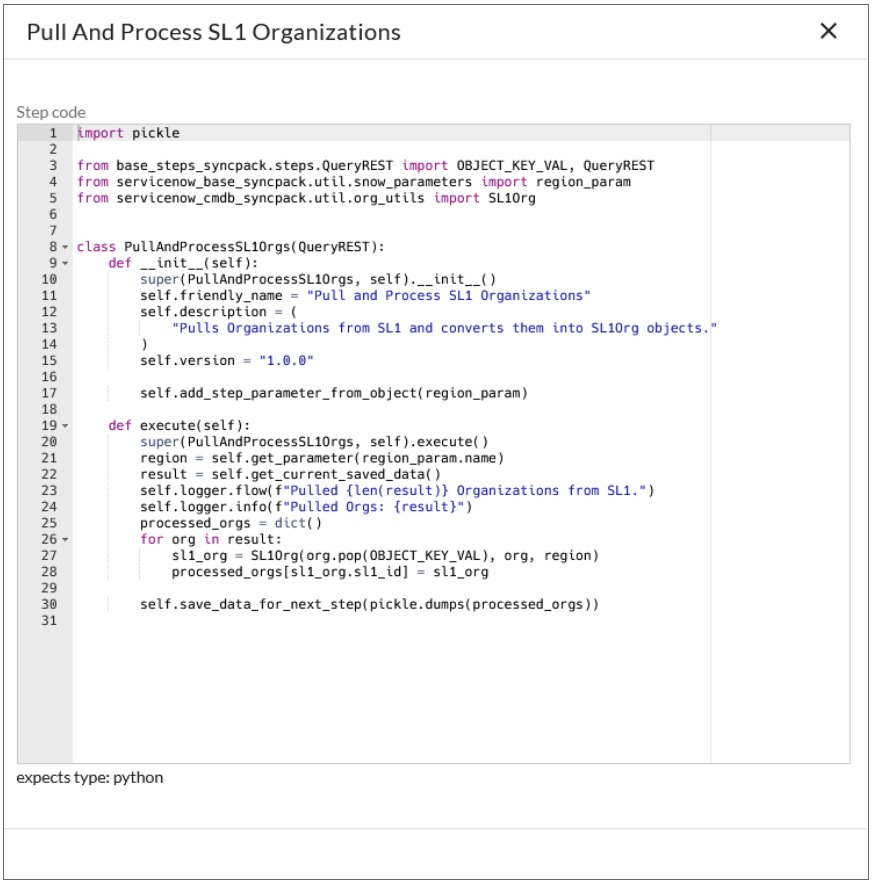
Using Steps in a PowerFlow Application

You can create new steps or use existing steps to create your own workflows, and you can re-use steps in more than one workflow. When these steps are combined as part of a PowerFlow application, they provide a workflow that satisfies a business requirement.

For example, the set of steps below in the "Sync Organizations from SL1 to ServiceNow Companies" application in the PowerFlow user interface gathers data about SL1 organizations and ServiceNow companies, processes that data based on the configuration settings specified for that set of steps, and posts that data to SL1 and ServiceNow to keep the organization and company data in sync in both places:



In the PowerFlow builder user interface, if you click the ellipsis icon (•••) on a step, you can select *View step code* to view the Python code for that step:



Using Input Parameters to Configure a Step

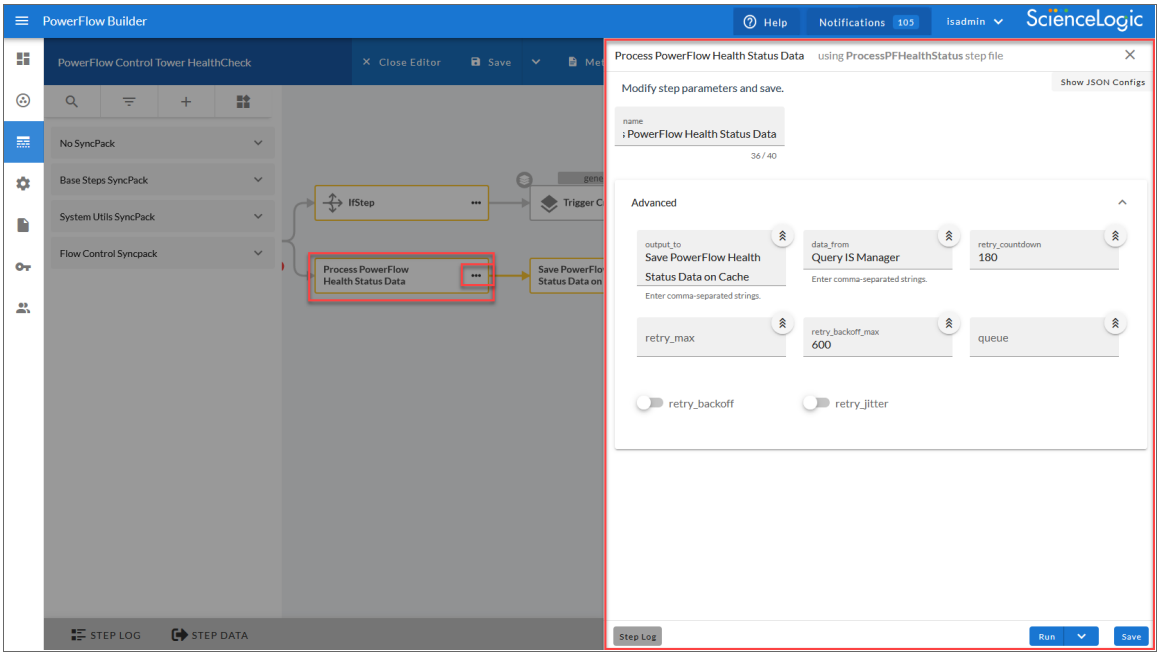
You can configure how a step works by adjusting a set of arguments called ***input parameters***. The parameters specify the values, variables, and configurations to use when executing the step. Parameters allow steps to accept arguments and allow steps to be re-used in multiple integrations.

For example, you can use the same step to query both the local system and another remote system; only the arguments, such as hostname, username, and password change.

To view and edit the input parameters for a step in the PowerFlow builder:

1. Go to the **Applications** page of the PowerFlow user interface and click the name of a PowerFlow application.
2. Click the **[Open Editor]** button.

3. Click the ellipsis icon (•••) on the step and select *Configure*. The **Configuration** pane for that step appears:



Sharing Data Between Steps

A step can pass the data it generates during execution to a subsequent step. A step can use the data generated by another step. Also, you can run test data for that step by hovering over the **[Run]** button and selecting *Custom Run*.

PowerFlow analyzes the required parameters for each step and alerts you if any required parameters are missing before running the step.

Types of Steps

Steps are grouped into the following types:

- ***Standard***. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or a database insert. These steps can be run independently and concurrently.
- ***Aggregated***. Aggregated steps require data that was generated by a previously run step. Aggregated steps are not executed by PowerFlow until all data required for the aggregation is available. These steps can be run independently and concurrently.
- ***Trigger***. Trigger steps are used to trigger other PowerFlow applications. These steps can be configured to be blocking or not (in other words, if the step is set to be blocking and it fails to trigger the application, the application will fail).

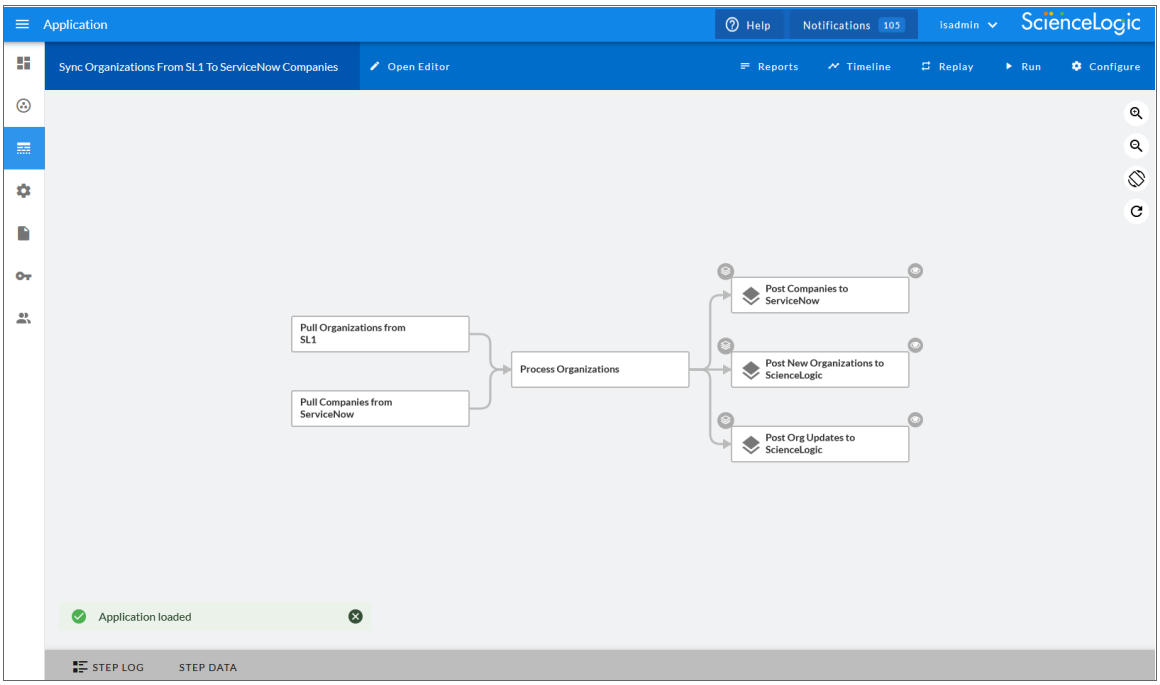
A variety of generic steps are available from ScienceLogic, and you can access a list of steps by sending a GET request using the ***API /steps endpoint***.

What is a PowerFlow Application?

In PowerFlow, an ***application*** is a JSON object that specifies which steps to execute and the order in which to execute those steps. An application also defines variables and provides arguments for each step.

An application combines a set of PowerFlow steps that execute a workflow. The input parameters for each step are also defined in the application and can be provided either directly in the step or in the parent application.

The following is an example of a PowerFlow application:



PowerFlow application JSON objects are defined by configuration settings, steps that make up the application, and application-wide variables used as parameters for each step. The parameters of each step can be configured dynamically, and each step can be named uniquely while still sharing the same underlying class, allowing for maximum re-use of code.

You can run an application in the PowerFlow user interface. You can also execute an application through the REST API, and PowerFlow will process the application as an asynchronous task. Executing an application from the REST API lets you dynamically set parameter values for the variables defined in the application.

During processing, PowerFlow generates a unique task ID for the application and each of its tasks. Using the task IDs, you can poll for the status of the application and the status of each individual running step in the application.

The required parameters of applications are strictly enforced, and PowerFlow will refuse to execute the application if all required variables are not provided.

For more information about PowerFlow applications, see *Managing PowerFlow Applications*.

What is a Configuration Object?

Configuration variables are defined in a standalone JSON file called a ***configuration object*** that lives on the PowerFlow system and can be accessed by all PowerFlow applications and their steps. Configuration objects can include variables like hostname, user name, password, or other credential information.

Each global variable is defined as a JSON object in the configuration object. Typically, the JSON code for a configuration object looks like the following:

```
{
   "encrypted": true,
   "name": "var_name",
   "value": "var_value"
}
```

Configuration objects allow the same application to be deployed in multiple PowerFlow instances, with different configurations. Click the **[Configure]** button from an application in the PowerFlow user interface to access the configuration object for that application.

Configuration objects can map variables from the SL1 platform to a third-party platform. For instance, SL1 has device classes and ServiceNow has CI classes; the configuration object maps these two sets of variables.

Each global variable in the configuration can be encrypted. The values of encrypted variables are encrypted within PowerFlow upon upload through the REST API.

You can access a list of all available configurations on the **Configurations** page ( ) of the PowerFlow user interface. You can also create and edit configuration objects on this page.

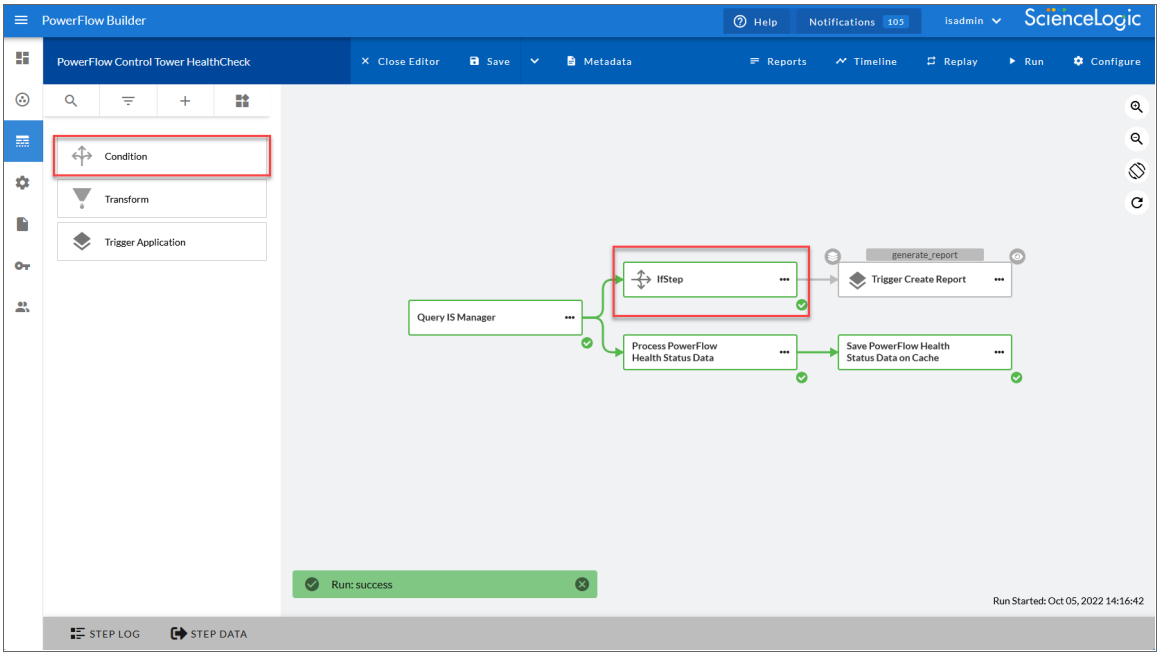
For more information about configuration objects, see *Managing Configuration Objects*.

What is the SL1 PowerFlow Builder?

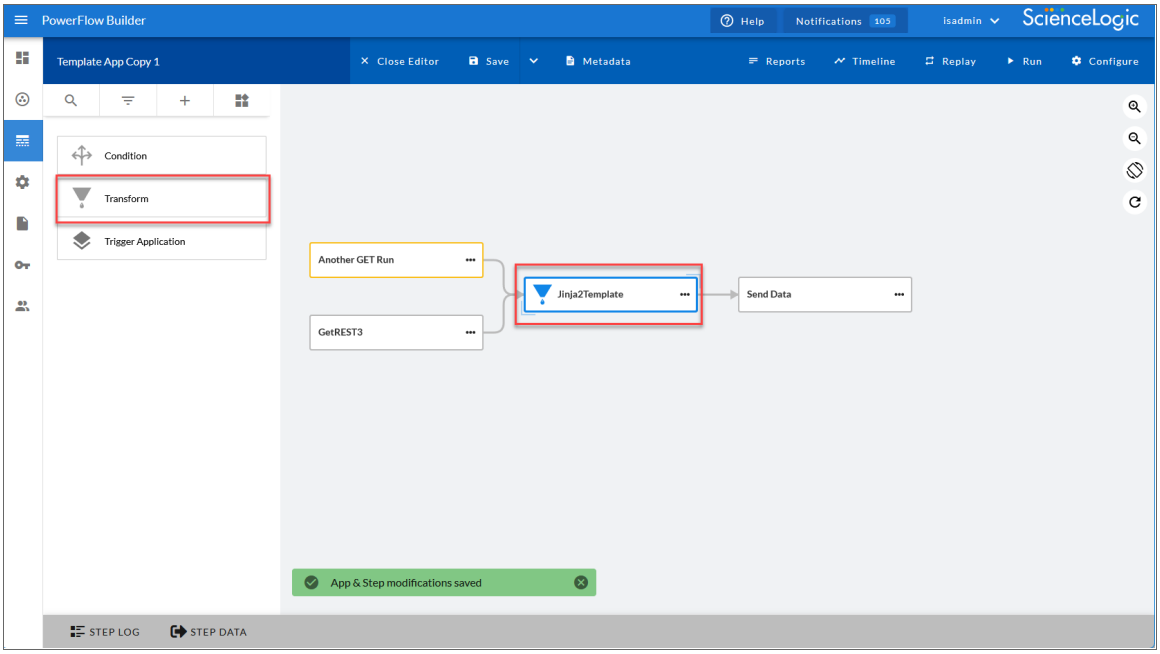
You can use the **SL1 PowerFlow builder** in the PowerFlow user interface to create complicated applications with logical branching and data transformation features using drag-and-drop components. You access the PowerFlow builder on the **Applications** page in the PowerFlow user interface.

NOTE: If your current ScienceLogic SL1 solution subscription does not include the SL1 PowerFlow builder, contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

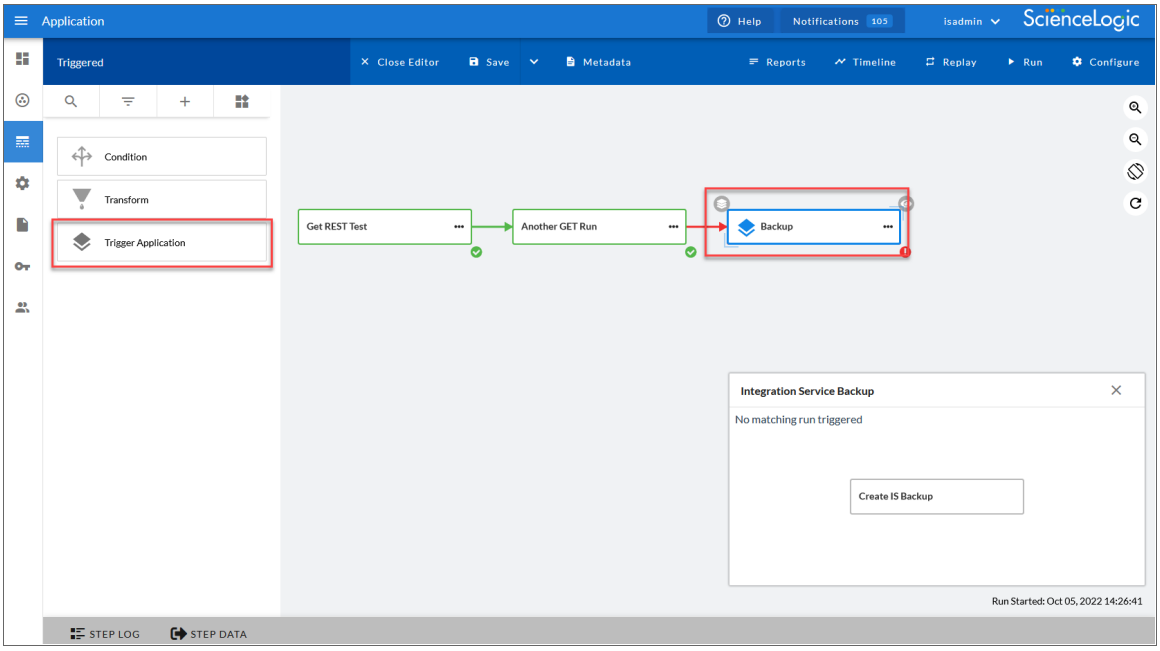
From the **Steps Registry** pane on an **Application** page, you can drag a **Condition** operator ( ) onto an application workflow to create the option for branching flows, such as If-Else or If-Then-Else statements:



You can drag a **Transform** operator ( ) from the **Steps Registry** pane onto an application workflow to pull data gathered by a previous step and modify or transform the data to fit into the next step:



You can also use the **Trigger Application** operator ( ) to launch one or more PowerFlow applications from within a new or existing PowerFlow application. This operator uses the same functionality as the "Trigger Application" step from the Base Steps SyncPack:



TIP: Clicking the eye icon ( ) next to a triggered application generates a smaller window, also called a "picture-within-a-picture", that displays the step or steps for the triggered application.

For more information about the PowerFlow builder and PowerFlow applications, see *Managing PowerFlow Applications*.

Elements of the PowerFlow User Interface

The PowerFlow user interface matches the layout of the SL1 user interface, with the navigation tabs located on the left-hand side of the window. The tabs provide access to the following pages: **PowerFlow Control Tower**, **SyncPacks**, **Applications**, **Configurations**, **Reports**, **API Keys**, and **Admin Panel**.

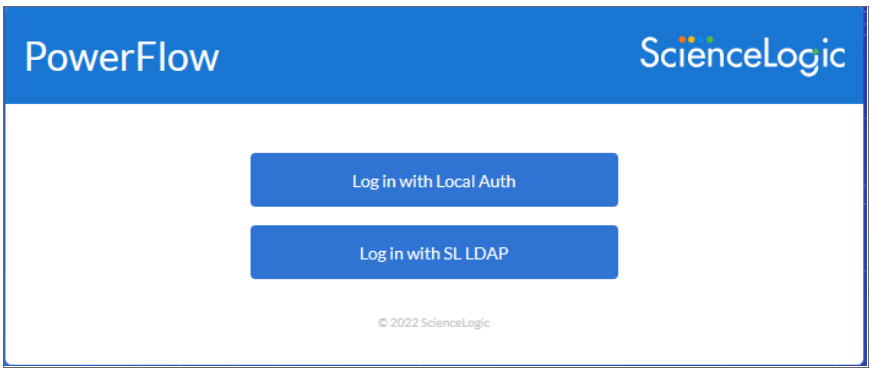
Logging In and Out of the PowerFlow User Interface

You can log in to PowerFlow using one of the following authentication types:

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default. Local authentication only supports the *isadmin* administrator user.
- ***Basic Authentication***. PowerFlow continues to support Basic Authentication as well. Because the PowerFlow SyncPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication.

- ***OAuth***. Lets PowerFlow administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as Active Directory, or using a protocol like LDAP, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.
- ***Common Access Card (CAC) Authentication***. Lets a PowerFlow user provide a CAC card through a browser to the PowerFlow root IP address. After identifying the CAC card, the ingress proxy verifies and authenticates the user. CAC authentication bypasses Dex authentication and does not use OIDC protocols. You can also use CAC authentication with LDAP, or CAC authentication with LDAP and SAN.
- ***API Key Authentication***. Provides access to the PowerFlow API in a controllable manner, with options to restrict which hosts may or may not use certain tokens.

Depending on the authentication used by your PowerFlow system, the PowerFlow login page will display a single option for logging in, or more than one option:



For more information about configuring authorization for users, see *Managing Users in PowerFlow*.

NOTE: On a PowerFlow system configured for Military Unique Deployment (MUD) and Department of Defense (DoDIN), a login banner containing information specific to DoDIN appears after you log in.

TIP: If you get a "SyncPacks service is not reachable" pop-up message in the user interface and the various pages are empty, log out of the PowerFlow user interface and log back in again. You can also click **[Refresh]** in your browser to automatically log out. This situation occurs only if the user interface is idle for a long period of time.

After you log in, pop-up notification appears at the bottom of the PowerFlow user interface that states the last time you logged into the PowerFlow system. This notification disappears after a short period of time, or you can close it. Additional notifications will display in this same area at the bottom left of the window.

To log out of PowerFlow, click your user name in the navigation bar in the top right of any window and select *Log off*.

PowerFlow Pages

The **PowerFlow Control Tower** page ( ) provides a graphical view of the various tasks, workers, and applications that are running on your PowerFlow system. This page was called the **Dashboard** page in previous versions of PowerFlow. For more information, see *Using the PowerFlow Control Tower Page*.

The **SyncPacks** page ( ) lets you import, install, and view SyncPacks, which contain applications, steps, and configuration objects that you can use in PowerFlow. For more information, see *Managing SyncPacks*.

The **Applications** page ( ) provides a list of the applications available on your PowerFlow system. This page was called the **Integrations** page in previous versions of PowerFlow. From this page you can run and schedule applications. If you are a Premium solution user, you can use the PowerFlow builder to create applications that use logical branching and data transformation between steps. For more information, see *Managing PowerFlow Applications*.

The **Configurations** page ( ) lets you create or use a configuration object to define a set of variables that all steps and PowerFlow applications can use. For more information, see *Managing Configuration Objects*.

The **Reports** page ( ) contains a list of reports associated with PowerFlow applications that have the reporting feature enabled, such as the "Report: Identify Unmapped Device Classes" and the "System Diagnostics" application. For more information, see *Generating and Viewing Reports for PowerFlow Applications*.

The **API Keys** page ( ) lets you create API keys, which you can use to send requests to PowerFlow API endpoints, specifying them by a header or a query string. These API keys are based on PowerFlow roles. For more information, see *Creating and Using API Keys in SL1 PowerFlow*.

The **Admin Panel** page ( ) contains a list of user groups, which lets you determine the roles and access for your users. You can also manage user sessions on this page. Only users with the *Administrator* role for this PowerFlow system can edit this page. For more information, see *Managing Users in PowerFlow*.

> TIP: While the **SyncPacks**, **Applications**, **Configurations**, **Reports**, **API Keys**, and **Admin Panel** pages are loading or running a procedure, you will see a dark blue, animated line running across the top of the page until the process completes. On the PowerFlow Control Tower page, the **System Health** widget displays an image that shows the progress of data loading in that widget.

Additional Navigation

The "Auto Refresh" counter lets you see when data on the following pages will update: **SyncPacks**, **Applications**, **Configurations**, and **API Keys**. On these pages, you can also click "Auto Refresh" to refresh the page immediately.

Also, pop-up messages have a countdown timer that displays until it closes, along with a Close icon ( ).

Clicking the **[Help]** button in the navigation bar in the top right of the PowerFlow user interface opens the **Help Menu** pane on the right-hand side of the user interface. The **Help Menu** pane contains an overview of the current page, a list of actions you can take on this page (with accompanying links to the corresponding Help topics), and a link to the corresponding page in the product documentation for more information.

Clicking the **[Notifications]** button in the navigation bar opens the **Notification Center** pane, which contains a log of all previous pop-up notifications that appeared in the PowerFlow system about applications that were run

successfully or with warnings or failures. The button also displays a light-blue "badge" with the number of current notifications. For more information about a notification, click the link for the page where the notification appeared and review the **Step Log** and **Step Data** tabs for the application steps.

> **TIP:** To clear the contents of the **Notification Center** pane, click the **[Delete All]** button. Click the Close icon (✕) to close the **Notification Center** pane.

The user name drop-down, which is found in the navigation bar in the top right of the PowerFlow user interface, contains the following options:

- *About*. Displays package versions, user information for the current user, version information for PowerFlow, and licenses used by PowerFlow. This page also display whether the PowerFlow system is licensed, and when the license expires.
- *Log Off*. Logs you out of the PowerFlow user interface.

The footer at the bottom of each PowerFlowpage displays a timestamp of the last login, as well as your PowerFlow version, whichyou can click to open the **About** page.

Using the API or Command Line Tool to Create PowerFlow Components

Instead of using the PowerFlow user interface, you can create steps, applications, and configurations in your own editor and then upload them using the API or the command line tool (iscli).

For more information, see *SL1 PowerFlow for Developers*.

Chapter 2

Installing and Configuring SL1 PowerFlow

Overview

This chapter describes how to install, upgrade, and configure PowerFlow, and also how to set up security for PowerFlow.

This chapter covers the following topics:

PowerFlow Architecture 35
Reviewing Your Deployment Architecture 38
System Requirements 39
Additional Prerequisites for PowerFlow 41
Installing PowerFlow 41
Installing PowerFlow on AWS 54
Converting PowerFlow to Oracle Linux 8 (OL8) 61
Upgrading from PowerFlow 3.x to the latest 3.x Version 64
Troubleshooting Upgrade Issues 72
Licensing PowerFlow 74
Configuring PowerFlow Services 78
Configuring a Proxy Server 82
Changing the PowerFlow System Password 84
Configuring Security Settings 86
Configuring Additional Elements of PowerFlow 89
PowerFlow Task Processing and Memory Handling 90
Best Practices for Running PowerFlow with Production Workloads 101

PowerFlow Management Endpoints *102*

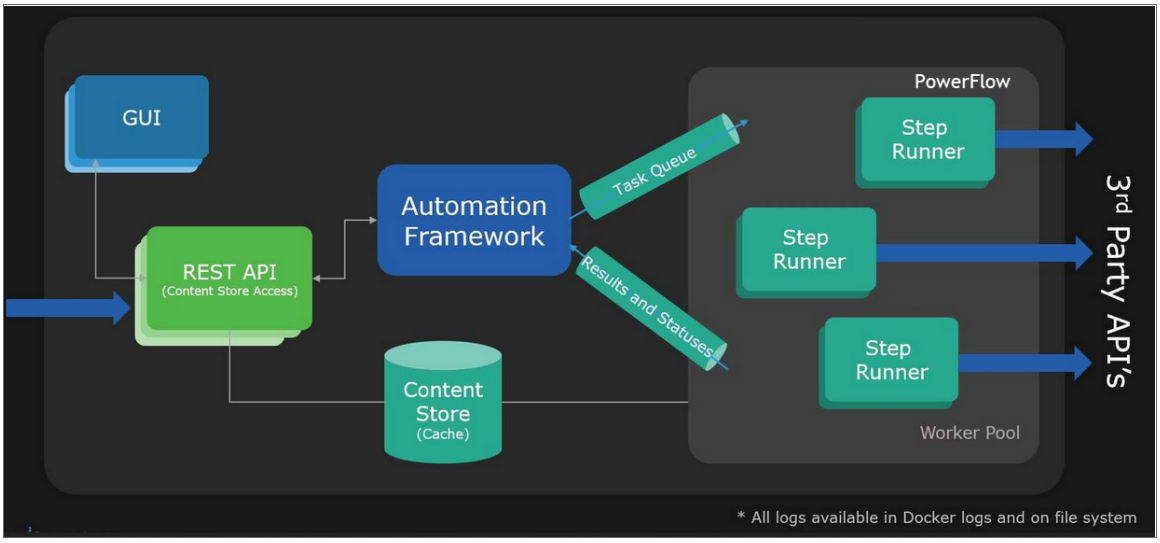
PowerFlow Architecture

This topic describes the different aspects of PowerFlow architecture.

PowerFlow Container Architecture

PowerFlow is a collection of purpose-built containers that are charged to pass information to and from SL1. Building PowerFlow architecture in containers allows you to add more processes to handle the workload as needed.

The following diagram describes the container architecture for PowerFlow:



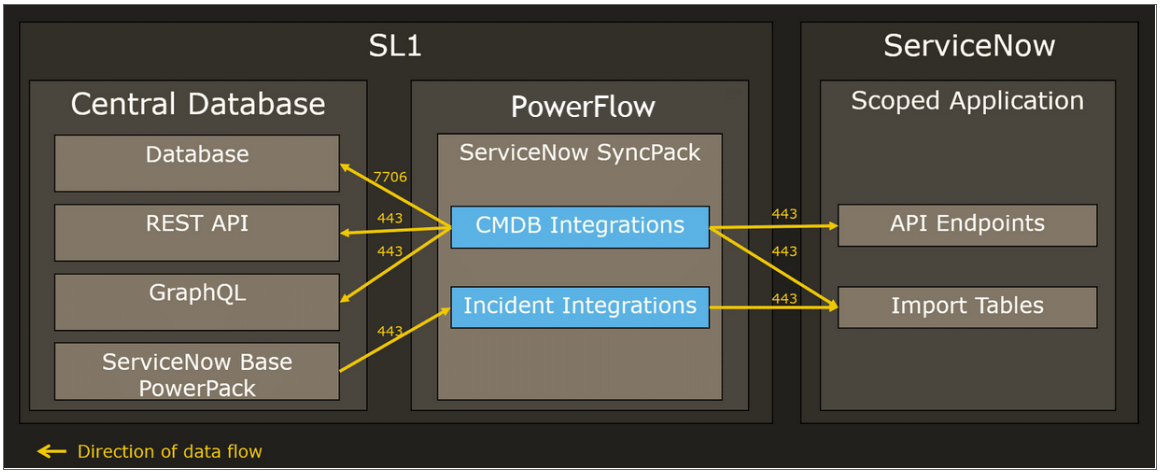
PowerFlow includes the following containers:

- ***GUI***. The GUI container provides the user interface for PowerFlow.
- ***REST API***. The REST API container provides access to the Content Store on the PowerFlow instance.
- ***Content Store***. The Content Store container is basically a database service that contains all the reusable steps, applications, and containers in the PowerFlow instance.
- ***Step Runners***. Step Runner containers execute steps independently of other Step Runners. All Step Runners belong to a Worker Pool and can run steps in order, based on the instructions in the applications. By default there are five Step Runners (worker nodes) include in the PowerFlow platform. PowerFlow users can scale up or scale down the number of worker nodes, based on the workload requirements.

TIP: You can use the **Control Tower** page in the PowerFlow user interface to monitor the health of these containers and workers. For more information, see *Using the SL1 PowerFlow Control Tower Page*.

Integration Workflow

The following high-level diagram for a ServiceNow Integration provides an example of how PowerFlow communicates with both the SL1 Central Database and the third-party (ServiceNow) APIs:



The workflow includes the following components and their communication methods:

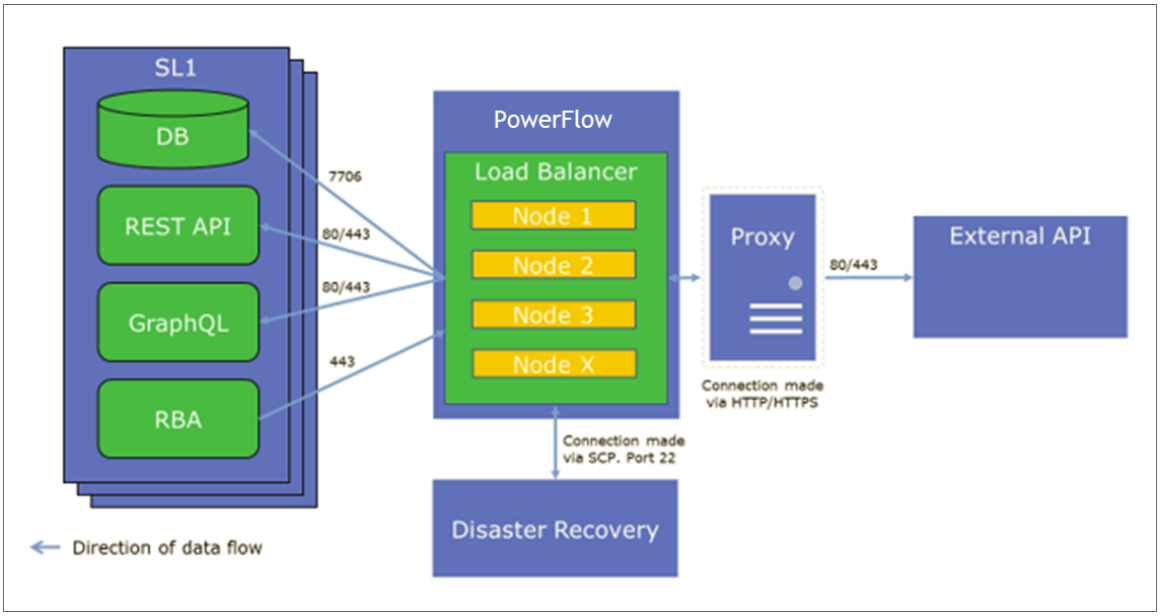
- ***SL1 Central Database***. PowerFlow communicates with the SL1 database over port 7706.
- ***SL1 REST API***. PowerFlow communicates with the SL1 REST API over port 443.
- ***GraphQL***. PowerFlow communicates with GraphQL over port 443.
- ***ServiceNow Base PowerPack***. In this example, the Run Book Automations from the ServiceNow Base PowerPack (and other SL1 PowerPacks) communicate with PowerFlow over port 443.
- ***PowerFlow***. PowerFlow communicates with both the SL1 Central Database and an external endpoint.
- ***ServiceNow API***. In this example, the ServiceNow applications in PowerFlow communicate with the ServiceNow API over port 443.

IMPORTANT: PowerFlow both pulls data from SL1 and has data pushed to it from SL1. PowerFlow both sends and retrieves information to and from ServiceNow, but PowerFlow is originating the requests.

High-Availability, Off-site Backup, and Proxy Architecture

You can deploy PowerFlow as a High Availability cluster, which requires at least *three* nodes to achieve automatic failover. While PowerFlow *can* be deployed as a single node, the single-node option does not provide redundancy through High Availability. PowerFlow also supports off-site backup and connection through a proxy server.

The following diagram describes these different configurations:



- ***High Availability*** for PowerFlow is a cluster of PowerFlow nodes with a Load Balancer managing the workload. In the above scenario, if one PowerFlow node fails, the workload will be redistributed to the remaining PowerFlow nodes. High Availability provides local redundancy. For more information, see *Appendix A: Configuring PowerFlow for High Availability*.
- ***Off-site Backup*** can be configured by using PowerFlow to back up and recover data in the Couchbase database. The backup process creates a backup file and sends that file using Secure Copy Protocol (SCP) to a user-defined, off-site destination system. You can then use the backup file from the remote system and restore its content. For more information, see *Creating a Backup*.
- A ***Proxy Server*** is a dedicated computer or software system running as an intermediary. The proxy server in the above scenario handles the requests between PowerFlow and the third-party application. For more information, see *Configuring a Proxy Server*.

In addition, you can deploy PowerFlow in a multi-tenant environment that supports multiple customers in a highly available fashion. After the initial High Availability (HA) core services are deployed, the multi-tenant environment differs in the deployment and placement of workers and use of custom queues. For more information, see *Appendix B: Configuring PowerFlow for Multi-tenant Environments*.

NOTE: There is no support for active or passive Disaster Recovery. ScienceLogic recommends that your PowerFlow Disaster Recovery plans include regular backups and restoring from backup. For more information, see *Creating a Backup*.

Reviewing Your Deployment Architecture

Review the following aspects of your architecture before deploying PowerFlow:

A. How many SL1 stacks will you use to integrate with the third-party platform (such as ServiceNow, Cherwell, or Restorepoint)?

B. What is a good estimate of the number of devices across all of your SL1 stacks?

C. How many data centers will you use?

D. Specify the location of each data center.

E. What is the latency between each data center? (Latency must be less than 80 ms.)

F. How many SL1 stacks are in each data center?

G. Are there any restrictions on data replication across regions?

H. What is the location of the third-party platform (if applicable)?

I. What is the VIP for Cluster Node Management?

Based on the above list, ScienceLogic recommends the following deployment paths:

- For question A, if you answered **three or fewer** SL1 stacks, consider a standard High-Availability deployment. For more information, see *Appendix A: Configuring PowerFlow for High Availability*.
- For question A, if you answered **more than three** SL1 stacks to question A, consider configuring PowerFlow in a multi-tenant configuration. For more information, see *Appendix B: Configuring PowerFlow for Multi-tenant Environments*.
- For question G, if you answered "**Yes**" to data replication restrictions, consider the following deployment options:
    - **Deploy separate PowerFlow clusters per region**. This deployment requires more management of PowerFlow clusters, but it ensures that the data is completely separated between regions. This deployment also ensures that if a single region goes down, you only lose operations for that region.
    - **Deploy a single PowerFlow cluster in the restrictive region**. This deployment is easier to manage, as you are only dealing with a single PowerFlow cluster. As an example, if Europe has a law that requires that data in Europe cannot be replicated to the United States, but that law does not prevent data from the United States from coming into Europe, you can deploy a single PowerFlow cluster in Europe to satisfy the law requirements.
- If you are deploying a multi-tenant configuration, check to see if your environment meets one the following:
    - **You have three or more data centers and the latency between each data center is less than 80 ms** (question E), consider deploying a multi-tenant PowerFlow where each node is in a separate data center to ensure data center resiliency. This deployment ensures that if a single data center goes down, PowerFlow will remain operational.
    - **You have only two data centers and the latency between data centers is less than 80 ms**, consider deploying a multi-tenant PowerFlow where two nodes are in one data center and the other node is in the other data center. This deployment does not ensure data center resiliency, but it does provide standard High Availability if a single node goes down. If the data center with one node goes

down, PowerFlow will remain operational. However, if the data center with two nodes goes down, PowerFlow will no longer remain operational.

- **You have only two data centers but the latency between data centers is *more* than 80 ms**. In this situation, you can still deploy a multi-tenant PowerFlow, but all nodes must be located in a single data center. This deployment still provides standard High Availability so that, if a single node goes down, the other two nodes ensure PowerFlow operations. If you require more resiliency than a single-node failure, you can deploy five nodes, which will ensure resiliency with two down nodes. However, if the data center goes down, PowerFlow will not be operational.
- **You only have one data center, you can still deploy a multi-tenant PowerFlow, but all nodes are located in a single data center**. This deployment still provides standard High Availability so that, if a single node goes down, the other two nodes ensure PowerFlow operations. If you require more resiliency than a single-node failure, you can deploy five nodes, which will ensure resiliency with two down nodes. However, if the data center goes down, PowerFlow will not be operational.

System Requirements

PowerFlow itself does not have specific minimum required versions for SL1 or AP2. However, certain PowerFlow SyncPacks have minimum version dependencies, which are listed on the *Dependencies for SL1 PowerFlow SyncPacks* page.

Ports

The following table lists the PowerFlow ingress requirements:

Source	Port	Purpose
SL1 host	443	SL1 run book actions and connections to PowerFlow
User client	3141	Devpi access
User client	443	PowerFlow API
User client	5556	Dex Server: enable authentication for PowerFlow
User client	8091	Couchbase Dashboard
User client	15672	RabbitMQ Dashboard
User client	22	SSH access

The following table lists the PowerFlow egress requirements:

Destination	Port	Purpose
SL1 host	7706	Connecting PowerFlow to SL1 Database Server
SL1 host	443	Connecting PowerFlow to SL1 API

Additional Considerations

Review the following list of considerations and settings before installing PowerFlow:

- ScienceLogic highly recommends that you disable all firewall session-limiting policies. Firewalls will drop HTTPS requests, which results in data loss.
- Starting with PowerFlow version 3.0.0, the minimum storage size for the initial partitions is 75 GB. Anything less will cause the automated installation to stop and wait for user input. You can use the tmux application to navigate to the other panes and view the logs. In addition, at 100 GB and above, **PowerFlow** will no longer allocate all of the storage space, so you will need to allocate the rest of the space based on your specific needs.
- PowerFlow clusters do not support vMotion or snapshots while the cluster is running. Performing a vMotion or snapshot on a running PowerFlow cluster will cause network interrupts between nodes, and will render clusters inoperable.
- The site administrator is responsible for configuring the host, hardware, and virtualization configuration for the PowerFlow server or cluster. If you are running a cluster in a VMware environment, be sure to install open-vm-tools and disable vMotion.
- You can configure one or more SL1 systems to use PowerFlow to sync with a *single* instance of a third-party application like ServiceNow or Cherwell. You cannot configure one SL1 system to use PowerFlow to sync with *multiple* instances of a third-party application like ServiceNow or Cherwell. The relationship between SL1 and the third-party application can be either one-to-one or many-to-one, but not one-to-many.
- The default internal network used by PowerFlow services is **172.21.0.1/16**. Please ensure that this range does not conflict with any other IP addresses on your network. If needed, you can change this subnet in the **docker-compose.yml** file.

For more information about system requirements for your PowerFlow environment, see the System Requirements page at the ScienceLogic Support site at https://support.sciencelogic.com/s/system-requirements.

Hardened Operating System

The operating system for PowerFlow is pre-hardened by default, with firewalls configured only for essential port access and all services and processes running inside Docker containers, communicating on a secure, encrypted overlay network between nodes. Please refer to the table, above, for more information on essential ports.

You can apply additional Linux hardening policies or package updates as long as Docker and its network communications are operational.

The PowerFlow operating system is an Oracle Linux distribution, and all patches are provided within the standard Oracle Linux repositories. The patches are not provided by ScienceLogic.

Additional Prerequisites for PowerFlow

To work with PowerFlow, ScienceLogic recommends that you have knowledge of the following:

- Linux and vi (or another text editor).
- Python.
- Postman or another API tool for interacting with the PowerFlow API.
- Couchbase (Community Edition). For more information, see *Helpful Couchbase Commands*.
- Docker. For more information, see *Helpful Docker Commands* and https://docs.docker.com/engine/reference/commandline/cli/.

NOTE: The most direct way of accessing the most recent containers of PowerFlow is by *downloading the latest RPM file* from the ScienceLogic Support Portal. As a separate option, you can also access the PowerFlow containers directly through Docker Hub. To access the containers through Docker Hub, you must have a Docker Hub ID and enable permissions to pull the containers from Docker Hub. To get permissions, contact your ScienceLogic Customer Success Manager.

Installing PowerFlow

NOTE: Starting with version 2.3.0, all [[[Undefined variable ApplianceNames.IS4]]] platform releases are suitable for both MUD and non-MUD systems.

IMPORTANT: Due to the upcoming end of support for Oracle Linux 7, ScienceLogic strongly urges users to upgrade to Oracle Linux 8 (OL8). As such, only the OL8-based package and upgrade path is defined and provided. If you have extenuating circumstances and want to obtain an OL7-based install for PowerFlow 3.0.0, please contact your CSM or ScienceLogic support.

Installing PowerFlow for the First Time

You can install PowerFlow for the first time in the following ways:

- *Via ISO to a server on your network*
- *Via RPM to a cloud-based server*

If you are installing PowerFlow in a clustered environment, see *Configuring the PowerFlow System for High Availability*.

Upgrading an Existing PowerFlow System

- If you are upgrading an existing version of PowerFlow to version 3.0.0 or later, the steps are slightly different, because you will need to convert the operating system to Oracle Linux 8. For more information, see *Converting PowerFlow to Oracle Linux 8 (OL8)*.
- If you are upgrading an existing version of PowerFlow to a version before version 3.0.0, see *Upgrading PowerFlow*.

CAUTION: The site administrator is responsible for configuring the host, hardware, and virtualization configuration for the PowerFlow server or cluster. If you are running a cluster in a VMware environment, be sure to install open-vm-tools and disable vMotion.

Installing PowerFlow via ISO

IMPORTANT: Due to the upcoming end of support for Oracle Linux 7, ScienceLogic strongly urges users to upgrade to Oracle Linux 8 (OL8). As such, only the OL8-based package and upgrade path is defined and provided. If you have extenuating circumstances and want to obtain an OL7-based install for PowerFlow 3.0.0, please contact your CSM or ScienceLogic support.

Locating the ISO Image

To locate the PowerFlow ISO image:

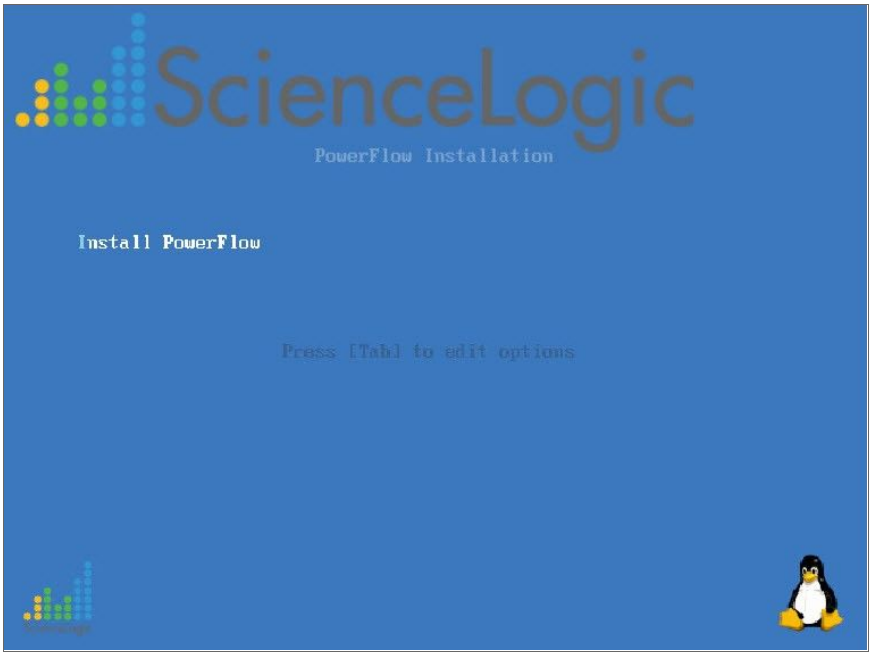
1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *PowerFlow Platform*. The **PowerFlow** page appears.
3. Click the link to the current release. The **Release Version** page appears.
4. In the **Release Files** section, click the ISO link for the PowerFlow image. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page.

Installing from the ISO Image

TIP: When installing PowerFlow from an ISO, you can now install open-vm-tools by selecting **Yes** to "Installing Into a VMware Environment" option during the installation wizard.

To install PowerFlow via ISO image:

1. Download the latest PowerFlow ISO file to your computer or a virtual machine center.
2. Using your hypervisor or bare-metal (single-tenant) server of choice, mount and boot from the PowerFlow ISO. The PowerFlow Installation window appears:



3. Select *Install PowerFlow*. The Military Unique Deployment window appears.
4. Select *Yes* only if you require a Military Unique Deployment (MUD) of the PowerFlow system. In most situations, you would select the default option of *No*. After the installer loads, the **Network Configuration** window appears.
5. Complete the following fields:
   - ***IP Address***. Type the primary IP address of the PowerFlow server.
   - ***Netmask***. Type the netmask for the primary IP address of the PowerFlow server.
   - ***Gateway***. Type the IP address for the network gateway.
   - ***DNS Server***. Type the IP address for the primary nameserver.
   - ***Hostname***. Type the hostname for PowerFlow.

5. Press **[Continue]**. The **Root Password** window appears.
6. Type the password you want to set for the root user on the PowerFlow host (and the service account password) and press **[Enter]**. The password must be at least six characters and no more than 24 characters, and all special characters are supported except the dollar sign ($) character.

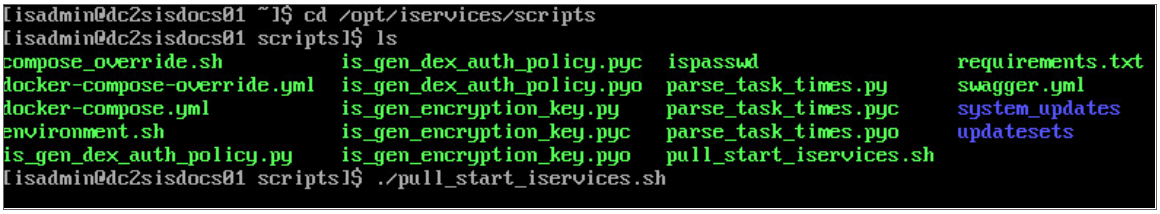
> **NOTE:** You use this password to log into the PowerFlow user interface, to SSH to the PowerFlow server, and to verify API requests and database actions. This password is set as both the "Linux host isadmin" user and in the **/etc/iservices/is_pass** file that is mounted into the PowerFlow stack as a "Docker secret". Because it is mounted as a secret, all necessary containers are aware of this password in a secure manner. For more information, see *Changing the PowerFlow Password*.

> **IMPORTANT:** To avoid authentication issues, do not use the dollar sign ($) character as the first character in any of the passwords related to PowerFlow. You can use the $ character elsewhere in the password if needed.

7. Type the password for the root user again and press **[Enter]**. The PowerFlow installer runs, and the system reboots automatically. This process will take a few minutes.
8. After the installation scripts run and the system reboots, SSH into your system using PuTTY or a similar application. The default username for the system is *isadmin*.
9. To start the Docker services, change directory to run the following commands:

```
cd /opt/iservices/scripts
```

```
./pull_start_iservices.sh
```



> **NOTE:** This process will take a few minutes to complete.

10. To validate that iservices is running, run the following command to view each service and the service versions for services throughout the whole stack:

```
docker service ls
```

11. Navigate to the PowerFlow user interface using your browser. The address of the PowerFlow user interface is:

```
https://<IP address entered during installation>
```

12. Log in with the default username of *isadmin* and the password you specified in step 6.
13. After installation, you must license your PowerFlow system if you want to enable all of the features. For more information, see ***Licensing PowerFlow***.

> **NOTE:** If you are licensing a PowerFlow High Availability cluster, you can run the licensing process on any node in the cluster once the cluster is ready. The node does not have to be the leader, and the licensing process does not have to be run on all nodes in the Swarm. If you are setting up High Availability for the PowerFlow on a multiple-node cluster, see ***Preparing the PowerFlow System for High Availability***.

> **NOTE:** The **HOST_ADDRESS** value in the **/etc/iservices/isconfig.yml** file should be the fully qualified domain name (FQDN) of either the host if there is no load balancer, or the FQDN of the load balancer if one exists. If you change the **HOST_ADDRESS** value, you will need to restart the PowerFlow stack.

Troubleshooting the ISO Installation

To verify that your stack is deployed, view your Couchbase logs by executing the following command:

```
docker service logs --follow iservices_couchbase
```

If no services are found to be running, run the following command to start them:

```
docker stack deploy -c docker-compose.yml iservices
```

To add or remove additional workers, run the following command:

```
docker service scale iservices_steprunner=10
```

> **NOTE:** ICMP is disabled by default after version 3.0.0 of PowerFlow. If you need to enable it, run the following commands:
>
> ```
> firewall-cmd --add-protocol=icmp --permanent
> ```
>
> ```
> firewall-cmd --reload
> ```
>
> ```
> systemctl restart docker
> ```

Installing PowerFlow via RPM to a Cloud-based Environment

> IMPORTANT: Due to the upcoming end of support for Oracle Linux 7, ScienceLogic strongly urges users to upgrade to Oracle Linux 8 (OL8). As such, only the OL8-based package and upgrade path is defined and provided. If you have extenuating circumstances and want to obtain an OL7-based install for PowerFlow PowerFlow 3.0.0 and later, please contact your CSM or ScienceLogic support.

Considerations for the RPM Installation

- The PowerFlow version 3.0.0 and later RPM is OL8-based. As a result, you cannot install the PowerFlow PowerFlow 3.0.0 and later RPM in an OL7 environment.
- If you install the PowerFlow PowerFlow 3.0.0 and later RPM on any operating system *other* than Oracle Linux 8, ScienceLogic will only support the running application and associated containers. ScienceLogic will not assist with issues related to host configuration for operating systems other than Oracle Linux 8 (or Oracle Linux 7 for systems before PowerFlow version PowerFlow 3.0.0 and later).
- If you are deploying PowerFlow without a load balancer, you can only use the deployed IP address as the management user interface. If you use another node to log in to the PowerFlow system, you will get an internal server error. Also, if the deployed node is down, you must redeploy the system using the IP address for another active node to access the management user interface.
- The **HOST_ADDRESS** value in the **/etc/iservices/isconfig.yml** file should be the fully qualified domain name (FQDN) of either the host if there is no load balancer, or the FQDN of the load balancer if one exists. If you change the **HOST_ADDRESS** value, you will need to restart the PowerFlow stack.
- If you are installing the RPM in a cluster configuration, and you want to distribute traffic between the nodes, a load balancer is required.
- If you install the PowerFlow system in a cloud-based environment using a method other than an ISO install, you are responsible for setting up and configuring the requirements of the cloud-based environment.

Locating the RPM file

To locate the PowerFlow RPM file:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *PowerFlow*. The **PowerFlow** page appears.
3. Click the link to the current release. The **Release Version** page appears.
4. In the **Release Files** section, click the RPM link for the PowerFlow image. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page.

Installing from the RPM File

You can also install PowerFlow on other cloud-based environments, such as Microsoft Azure. For other cloud-based deployments, the process is essentially the same as the following steps: PowerFlow provides the containers, and the cloud-based environment provides the operating system and server.

You can install PowerFlow version 3.0.0 or later on any Oracle Linux 8 (OL8) operating system, even in the cloud, as long as you meet all of the operating system requirements. These requirements include CPU, memory, Docker and a **docker-compose** file installed, and open firewall settings. When these requirements are met, you can install the RPM and begin to deploy the stack as usual.

PowerFlow version 3.0.0 and later RPM is OL8-based. As a result, you cannot install the PowerFlow PowerFlow 3.0.0 and later RPM in an OL7 environment. Previous versions of PowerFlow before version PowerFlow 3.0.0 and later can use Oracle Linux 7.x, but ScienceLogic strongly recommends that you convert the operating system to OL8 as soon as possible. The steps below are specific for PowerFlow version 3.0.0 or later. For more information, see *Converting PowerFlow to Oracle Linux 8 (OL8)*.

NOTE: XFS is the default file system for Oracle operating systems, and OverlayFS is the default storage driver for Docker. For them to be compatible, the `d_type=true` option must be enabled. You can validate that setting with the `xfs_info` command, which is documented https://docs.docker.com/storage/storagedriver/overlayfs-driver/.

NOTE: For a clustered PowerFlow environment, you must install the PowerFlow RPM on every server that you plan to cluster into PowerFlow. You can load the Docker images for the services onto each server locally by running `/opt/iservices/scripts/pull_start_iservices.sh`. Installing the RPM onto each server ensures that the PowerFlow containers and necessary data are available on all servers in the cluster. For a High Availability PowerFlow system, run the steps below to install PowerFlow on three different nodes, and then run the steps in *Automating the Configuration of a Three-Node Cluster*.

IMPORTANT: The following procedure describes how to install PowerFlow via RPM to Amazon Web Service (AWS) EC2. The **ec2-user** must belong to the **iservices** group.

To install a single-node PowerFlow version 3.0.0 via RPM to a cloud-based environment (using AWS as an example):

1. In Amazon Web Service (AWS) EC2, click **[Launch instance]**. The **Launch an instance** page appears.

   IMPORTANT: If you are installing PowerFlow to another cloud-based environment, such as Microsoft Azure, set up the operating system and server, and then go to step 7.

2. Deploy a new Oracle Linux 8.0 virtual machine by searching for *131827586825* (the Oracle AWS Owner ID) in the ***Search our full catalog*** field in the **Application and OS Images** section.
3. Press **[Enter]**. The **Choose an Amazon Machine Image (AMI)** page appears.
4. Click the **[Community AMIs]** tab and click **[Select]** for the AMI file. The AMI used should be the latest available OL8 AMI published by Owner ID *131827586825*.
5. From the **Choose an Instance Type** page, select at least a t2.xlarge AMI instance, depending on your configuration:

- *Single-node deployments*. The minimum is *t2.xlarge* (four CPUs with 16 GB memory), and ScienceLogic recommends *t2.2xlarge* (8 CPUs with 32 GB memory).
- *Cluster deployments*. Cluster deployments depend on the type of node you are deploying. Refer to the separate multi-tenant environment guide for more sizing information. ScienceLogic recommends that you allocate at least 50 GB or more for storage.

6. Go to the **Step 6: Configure Security Group** page and define the security group:
   - Inbound port 443 needs to be exposed to any of the systems that you intend to integrate.
   - For access to the PowerFlow user interface, add the following ports to the security group:
     - 15672 TCP for RabbitMQ
     - 5556 for Dex Server authentication
     - 3141 for Devpi access

     For more information about ports, see the *System Requirements*.
   - Port 8091 is exposed through *https*. ScienceLogic recommends that you make port 8091 available externally to help with troubleshooting:

Type	Protocol	Port Range	Source	Description
Custom UDP Rule	UDP	8091	72.165.86.42/32	IS DB Admin Interf...
SSH	TCP	22	0.0.0.0/0	IS SSH access
Custom TCP Rule	TCP	8091	72.165.86.42/32	IS DB Admin interf...
HTTPS	TCP	443	0.0.0.0/0	IS HTTPS access

7. Upload the **sl1-powerflow-3.x.x-1.el8.x86_64.rpm** file to the PowerFlow server using SFTP or SCP.
8. Enable the necessary repositories by running the following commands on the PowerFlow system:

NOTE: Be sure to remove old OL7 repositories configuration from the **/etc/yum.repos.d** directory as they can cause errors while running the dnf update in step 9.

```
sudo dnf install yum-utils
```

```
sudo dnf config-manager --enable ol8_baseos_latest
```

```
sudo dnf config-manager --enable ol8_appstream
```

```
sudo dnf config-manager --enable ol8_addons
```

9. Run the following commands on the server instance to upgrade to Python 3.11 and install the cffi package:

> **NOTE:** If proxies are used, be sure to export the environment variables with the corresponding proxy information (`http_proxy, https_proxy`) so python packages can be installed

> **IMPORTANT:** Do not change the version of pip from 21.3.1 This version is required for PowerFlow.

```
sudo dnf update
```

```
sudo dnf remove -y python3 python3-pip python3-setuptools
```

```
sudo dnf install python3.11-pip
```

```
sudo dnf install python3.11-cffi
```

```
sudo pip3 install --upgrade pip==21.3.1
```

10. Ensure that the latest required packages are installed by running the following commands on the server instance:

```
sudo dnf install wget
```

```
wget --no-check-certificate
https://download.docker.com/linux/centos/8/x86_
64/stable/Packages/containerd.io-1.6.32-3.1.el8.x86_64.rpm
```

```
wget --no-check-certificate
https://download.docker.com/linux/centos/8/x86_
64/stable/Packages/docker-ce-26.1.3-1.el8.x86_64.rpm
```

```
wget --no-check-certificate
https://download.docker.com/linux/centos/8/x86_
64/stable/Packages/docker-ce-cli-26.1.3-1.el8.x86_64.rpm
```

```
wget --no-check-certificate
https://download.docker.com/linux/centos/8/x86_
64/stable/Packages/docker-ce-rootless-extras-26.1.3-1.el8.x86_64.rpm
```

```
wget --no-check-certificate
https://download.docker.com/linux/centos/8/x86_
64/stable/Packages/docker-compose-plugin-2.27.0-1.el8.x86_64.rpm
```

```
sudo dnf install -y containerd.io-1.6.32-3.1.el8.x86_64.rpm docker-
ce-26.1.3-1.el8.x86_64.rpm docker-ce-cli-26.1.3-1.el8.x86_64.rpm
docker-ce-rootless-extras-26.1.3-1.el8.x86_64.rpm docker-compose-
plugin-2.27.0-1.el8.x86_64.rpm
```

IMPORTANT: You might need to remove spaces from the code that you copy and paste from this manual. For example, in instances such as the `wget` command, above, line breaks were added to long lines of code to ensure proper pagination in the document.

NOTE: You will need to update both instances of the Docker version in this command if there is a more recent version of Docker CE on the Docker Download page: https://download.docker.com/linux/centos/7/x86_64/stable/Packages/.

11. Create the Docker group:

```
sudo groupadd docker
```

12. Add your admin user to the Docker group and the wheel group:

```
sudo usermod -aG docker $USER

sudo usermod -aG wheel $USER
```

where `$USER` is the **isadmin** user name or the **ec2-user** in AWS. The **ec2-user** should belong to the **iservices** group, which is created as part of this RPM installation process.

13. Log out and log back in to ensure that your group membership is re-evaluated.

14. Run the following commands for the configuration updates. If selinux is already disabled skip this step.

```
sudo setenforce 0

sudo vi /etc/selinux/config

SELINUX=permissive
```

> **NOTE:** If changing the `SELINUX=permissive` configuration does not work, replace it with `SELINUX=disabled`.

15. Run the following firewall commands as "sudo" Be sure the firewalld service is up and running using `sudo systemctl status firewalld`:

> **TIP:** For Microsoft Azure environments, the firewalld service may be down and masked. Unmask it using `sudo systemctl unmask firewalld`, enable and start it.

```
sudo firewall-cmd --add-port=2376/tcp --permanent

sudo firewall-cmd --add-port=2377/tcp --permanent

sudo firewall-cmd --add-port=7946/tcp --permanent

sudo firewall-cmd --add-port=7946/udp --permanent

sudo firewall-cmd --add-port=4789/udp --permanent

sudo firewall-cmd --add-protocol=esp --permanent

sudo firewall-cmd --reload
```

> **TIP:** To view a list of all ports, run the following command: `firewall-cmd --list-all`.

> **NOTE:** If you copy and paste any of the commands with a `--`, make sure the two hyphens are entered as hyphens and not special characters.

16. Install the remaining Python packages needed for the PowerFlow RPM file:

```
sudo dnf update

sudo rpm -qa|grep python3-pyyaml

sudo dnf remove python3-pyyaml

sudo pip3 install --no-build-isolation wheel

sudo pip3 install requests==2.27.1

sudo pip3 install --no-build-isolation pyyaml==5.4.1

sudo pip3 install --no-build-isolation MarkupSafe

sudo pip3 install --no-build-isolation docker-compose==1.27.4
```

17. Copy the PowerFlow RPM to the instance of installation and install the RPM. Use the complete location of the RPM file if you are located in another directory.

> **NOTE:** If proxies are used be sure to export the environment variables with the corresponding proxy information `http_proxy, https_proxy` so python packages can be installed.

```
sudo dnf install sl1-powerflow-3.X.X-1.el8.x86_64.rpm
```

```
sudo systemctl restart docker
```

> **IMPORTANT:** If an OL8 (hardened) image was used, the **/tmp** mount point might have been mounted using the **noexec** flag. If that is the case, run the following steps to install the RPM:
>
> ```
> mkdir -p $HOME/tmp
> sudo  TMPDIR=$HOME/tmp dnf install sl1-powerflow-3.X.X-1.el8.x86_64.rpm
> sudo systemctl restart docker
> ```

18. Add the user to the iservices group. Then, log out and log back in to ensure that your group membership is re-evaluated and the user was added to the iservices group using `groups`.

```
sudo usermod -aG iservices $USER
```

19. Create a password for PowerFlow. Be sure the group of that file is iservices.

```
printf '<password>' > /etc/iservices/is_pass
```

```
sudo chown root:iservices /etc/iservices/is_pass
```

where `<password>` is a new, secure password.

20. Before starting the PowerFlow application, make sure the `HOST_ADDRESS` in the **isconfig.yml** file is set as expected. The public IP or DNS should be used.

21. Pull and start iservices to start PowerFlow:

> **NOTE:** If an error related to installing syncpacks is displayed at the end, please wait a few moments and rerun 'pfctl init-sps', or manually install the syncpacks through the PowerFlow user interface.

```
sudo /opt/iservices/scripts/pull_start_iservices.sh
```

IMPORTANT: For an AWS deployment, ScienceLogic recommends that you switch to an Amazon EC2 user as soon as possible instead of running all the commands on root.

After installation, you must license your PowerFlow system to enable all of the features. Licensing is required for production systems only, not for test systems. For more information, see *Licensing PowerFlow*.

Troubleshooting a Cloud Deployment of PowerFlow

After completing the AWS setup instructions, if none of the services start and you see the following error during troubleshooting, you will need to restart Docker after installing the RPM installation.

```
sudo docker service ps iservices_couchbase --no-trunc
```

```
"error creating external connectivity network: Failed to Setup IP tables:
Unable to enable SKIP DNAT rule: (iptables failed: iptables --wait -t nat
-I DOCKER -i docker_gwbridge -j RETURN: iptables: No chain/target/match by
that name."
```

Installing PowerFlow on AWS

There are two options to install PowerFlow on AWS:

- Follow the instructions for *installing the RPM in cloud environments*

Or

- Use the PowerFlow AMI (Amazon Machine Image)

What are the ScienceLogic AMIs?

An instance is a virtual server that resides in the AWS cloud. An Amazon Machine Image (AMI) is the collection of files and information that AWS uses to create an instance. A single AMI can launch multiple instances.

For details on AMIs, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

The ScienceLogic AMIs are defined by ScienceLogic. ScienceLogic has created an AMI for each type of ScienceLogic appliance. You can use a ScienceLogic AMI to create Elastic Compute Cloud (EC2) instances for each type of ScienceLogic appliance.

NOTE: Elastic Compute Cloud (EC2) instances are virtual servers that come in a variety of configurations and can be easily changed as your computing needs change. For more information on EC2, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html.

The ScienceLogic AMIs are private and are for ScienceLogic customers only. After you collect specific information about your AWS account, you can send a request (and the collected information) to ScienceLogic, and ScienceLogic will share the ScienceLogic AMIs with you.

Getting the PowerFlow AMI

To get access to the PowerFlow AMIs:

1. Contact ScienceLogic Support to obtain the PowerFlow AMIs.
2. To view the ScienceLogic AMIs in your AWS account, go to the **AWS Management Console** page. Under the heading ***Compute***, click **[EC2]**.
3. In the **EC2 Dashboard** page, go to the left navigation bar. Under the heading ***Images***, click **[AMIs]**.
4. In the main pane, under ***Filters***, click **[Owned by me]** and then select *Private images*.
5. You should see AMIs with names that begin with "ScienceLogic PowerFlow" and end with the current release number for PowerFlow.
6. If you do not see AMIs with names that begin with "ScienceLogic PowerFlow", your EC2 Dashboard might have a default region that does not match the region for the ScienceLogic PowerFlow AMIs. To change the current region in the EC2 dashboard, click the region pull-down in the upper right and choose another region. Do this until you find the ScienceLogic PowerFlow AMIs.

NOTE: A region is a geographic location. AWS has data centers that include multiple regions. You can specify that an instance reside in a specific region. For more details on regions, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

Launching the New Instance

To launch the new EC2 instance from the ScienceLogic AMI:

1. Go to the ***EC2 Dashboard***.
2. Select the PowerFlow AMI. Click the **[Launch]** button.
3. In the **Choose Instance Type** page, select at least a *t2.xlarge* AMI instance, depending on your configuration.
    - For single-node deployments, the minimum is t2.xlarge (four CPUs with 16 GB memory), however ScienceLogic recommends t2.2xlarge (8 CPUs with 32 GB memory).
    - For cluster deployments, the instance type depends on the type of node you are deploying. ScienceLogic recommends allocating at least 100 GB or more for storage.
4. Click the **[Next: Configure Security Group]** button.
5. A security group is a reusable set of firewall rules. In the **Configure Security Group** page, do the following:
    - ***Assign a security group***. Select *Create a new security group*.
    - ***Security group name***. Enter a name or accept the default name.
    - ***Description***. Accept the default value in this field.
6. Use the table below to create security rules for each type of PowerFlow appliance. After completing each row, click the **[Add Rule]** button.

NOTE: Inbound port 443 must be exposed to any systems you want to integrate. Port 8091 is exposed through https. ScienceLogic recommends that you make port 8091 available externally to help with troubleshooting.

For access to the PowerFlow user interface, add the following ports to the security group:

- Rabbit MQ: 15672 TCP
- Dex Server authentication: 5556
- Devpi access: 3141

Type	Protocol	Port Range	Source	Description
Custom UDP Rule	UDP	8091	If you will always log in from a single IP address, select My IP. If you will log in to the instance from multiple IP addresses, enter those IP addresses, separated by commas, in this field.	Couchbase Administrator Dashboard
SSH (edit the default SSH rule)	TCP	22	If you will always log in from a single IP address, select My IP. If you will log in to the instance from multiple IP addresses, enter those IP addresses, separated by commas, in this field.	SSH. For SSH sessions from the user workstation to the appliance.
Custom TCP Rule	TCP	8091	If you will always log in from a single IP address, select My IP. If you will log in to the instance from multiple IP addresses, enter those IP addresses, separated by commas, in this field.	Couchbase Administrator Dashboard
HTTPS	TCP	443	If you will always log in from a single IP address, select My IP. If you will log in to the instance from multiple IP addresses, enter those IP addresses, separated by commas, in this field.	PowerFlow HTTPS access

7. Click the **[Next: Add Storage]** button.
8. In the **Add Storage** page, select the checkbox in the ***Delete on Termination*** column.
9. In the **Add Storage** page, set the disk space as needed. For more information about resource recommendations, see *CPU and Memory Requirements for PowerFlow*.
10. In the **Add Storage** page, select *gp3* for the ***Volume Type***.
11. Click the **[Next: Configure Instance Details]** button.
12. In the **Configure Instance Details** page, define the following:

- ***Number of Instances***. Enter "1".
- ***Request Spot Instances***. Do not select.
- ***Network***. For VPC-enabled accounts, specify the network where the instance will reside. If you are unsure of the network, accept the default.
- ***Subnet***. For VPC-enabled accounts, specify the subnet where the instance will reside. If you are unsure of the subnet, accept the default.
- ***Auto-assign Public IP***. If you select *Enable*, AWS will assign an IP address from the public pool to this instance. If you select *Disable*, you must assign an ***Elastic IP Address (EIP)*** to the instance.

NOTE: If you select *Enable* in the ***Auto-assign Public IP*** field, the IP address will change each time the instance is stopped or terminated. For All-In-One Appliances and for Administration Portals, you might want to use an Elastic IP address (EIP), which is a persistent IP address. See the section on ***Elastic IP Addresses (EIP)*** for details.

NOTE: For more information on Elastic IP Addresses, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html.

- ***IAM role***. If your organization uses IAM roles, select the appropriate role.

13. Configure the remaining settings according to your organization's requirements, or leave them with the default values.
14. Click the **[Review and Launch]** button and review the details of the new instance. Fix any problems to meet the requirements of your organization.
15. Click the **[Launch]** button.

NOTE: Because the root user is disabled for SSH access, you must reset the password for isadmin before using any SSH key.

Accessing the Appliance Using SSH

Before following the steps below, you should have already received the ScienceLogic PowerFlow AMIs and created an EC2 instance based on the ScienceLogic PowerFlow AMI. You also need access to SSH on the command line (for UNIX users) or have installed PuTTY (for Windows users).

Gathering Information Required for Accessing the Appliance Using SSH

To gather the required information:

1. Navigate to the EC2 Dashboard.
2. Select *Instances* from the lefthand navigation menu.
3. Click the row that contains the PowerFlow appliance instance.

4. The lower pane contains information about the instance. Take note of the Public DNS and Public IP, as you will need them later.

> **NOTE:** If you are using AWS instances to create a PowerFlow Cluster, perform the steps above for each AWS instance you want to include in the PowerFlow Cluster.

Configuring SSH

You can connect to your PowerFlow instance using the SSH command line (for UNIX users) or PuTTY (for Windows users).

1. Use the following credentials the first time you attempt to access the PowerFlow appliance instance. The system will immediately prompt you to change the password after accessing the system.
   - `username: isadmin`
   - `password: isadminisadminisadmin`
2. If you need to use an SSH key, you can add it after resetting the password. For information on adding or replacing a public key on your Linux instance, see Amazon's documentation at docs.aws.amazon.com.
   a. Enter the following to get the public key: `ssh-keygen -y -f /path_to_key_pair/my-key-pair.pem`
   b. Set the file `~/.ssh/authorized_keys` with the expected permissions `chmod 700 ~/.ssh/` and `chmod 600 ~/.ssh/authorized_keys`.

Deploying the PowerFlow Application

Once you have access to the EC2 Instance follow the steps below to deploy the PowerFlow application.

1. Update the hostname as needed. The default hostname is `powerflow`. For more information about updating the hostname, see Oracle's documentation at docs.oracle.com.
2. Create the password file. Execute these commands as a superuser (`sudo -i`) for MUD environments.

```
printf "<newpassword>" > /etc/iservices/is_pass
```

```
chmod 660 /etc/iservices/is_pass
```

3. Create isconfig.yml and encryption_key by executing the following command as a superuser.

```
/opt/iservices/scripts/system_updates/recreate_configs.sh
```

For MUD environments only, execute the following:

```
echo 'MUD_ENABLED: true' >> /etc/iservices/isconfig.yml
```

IMPORTANT: Be sure to set the correct value for the HOST_ADDRESS in the `/etc/iservices/isconfig.yml` file. If this doesn't have the public IP set, the PowerFlow application user interface will not be accessible.

4. Restart Docker.

```
systemctl restart docker
```

```
docker info
```

5. Deploy the PowerFlow system. If the stack needs to be configured as a cluster environment, please follow the steps for *Configuring PowerFlow for High Availability*.

```
/opt/iservices/scripts/pull_start_iservices.sh
```

Additional Configuration Steps

The minimum size of the EBS volume used for the PowerFlow deployment is 100 GB. If a bigger size is used, the partition can be resized and the LVM can be extended as needed.

The following is an example to extend the root file system with 10 extra GB, if the EBS volume size was ~110GB.

```
lsblk # To check the real disk size and the partition that needs to be resized. It would be xvda2, as it should not be the boot partition

parted /dev/xvda

(parted) print

(parted) resizepart 2 100%

(parted) quit

pvresize /dev/xvda2

vgs # This will show you how much storage is left for allocation

lvextend -L +10G /dev/isvg/root

xfs_growfs /dev/mapper/isvg-root

df -h # To see the final result
```

Converting PowerFlow to Oracle Linux 8 (OL8)

Starting with version 3.0.0 of PowerFlow, you can convert your PowerFlow system to Oracle Linux 8. ScienceLogic strongly recommends you make the conversion to OL8 as soon as possible, as OL7 is End of Life (EOL) as of the end of 2024.

Complete the upgrade steps in the following order:

1. If needed (see the tables, below), back up the PowerFlow system.
2. Install PowerFlow version 3.0.0 or later using the .iso file.
3. If needed, restore the PowerFlow system.

Upgrade Options for Converting from PowerFlow 2.x (OL7) to PowerFlow 3.x or Later (OL8)

Select one of the following options to upgrade from an older version of PowerFlow running OL7 to PowerFlow version 3.0.0 or later running OL8:

Upgrade Option	Requirements	Implications, Downtime, Other Considerations
Backup, install, and restore, using a separate system	Identical, secondary environment for installing the PowerFlow 3.0.0 .iso file	This approach allows for the existing PowerFlow system to continue running while you deploy and configure a new OL8 based system. Once data is fully restored on the new system, you may switch the load balancer configuration to point to the new system, virtually eliminating downtime. This approach also allows you to use the old PowerFlow system as a fallback or rollback option.
Backup, re-install, and restore, using the same system	A separate file store where backup and configuration files can be stored temporarily	This approach will incur downtime as your existing system will be re-installed to PowerFlow 3.0.0 and restored with the data from the previous version. This approach does not allow you to roll back or switch back over to the older version of PowerFlow.

Upgrade Paths Based on PowerFlow Environments

Your upgrade options depend on your PowerFlow environment, so review the following table before beginning the upgrade and conversion process.

Environment Type	Upgrade Options	Recommended Option	Additional Notes
Internet-connected, on-premises	Run a back up, re-install, and restore.	Run a back up, re-install,	None

Environment Type	Upgrade Options	Recommended Option	Additional Notes
		and restore.	
Offline on-premises	Run a back up, re-install, and restore.	Run a back up, re-install, and restore.	None
MUD installation (FIPS-enforced)	Run a back up, re-install, and restore.	Run a back up, re-install, and restore.	None.
AWS-based cloud installation	Run a back up, re-install, and restore.	Run a back up, re-install, and restore.	None
Azure-based cloud installation	Run a backup, install, and restore using a separate system.	Run a backup, install, and restore using a separate system.	None

Back Up, Re-install, and Restore Your PowerFlow System

If you are backing up, installing, and restoring using the *same* system, you will need a separate file store where backup and configuration files can be stored temporarily. This approach will incur downtime as your existing system will be re-installed to 3.0.0 and restored with the data from previous. This approach does not allow you to roll back or switch back over to the older version.

If you are backing up, installing, and restoring using a *separate* system, you will need an identical, secondary environment for installing the PowerFlow 3.0.0 .iso file. This approach allows for the existing PF system to continue running/operating while you deploy and configure a new OL8 based system. Once data is fully restored on the new system, you may switch load balancer config to point to the new system, virtually eliminating downtime. This approach also allows you to use the old system as a fallback/rollback option

Use the following steps to back up your current PowerFlow configuration after you have converted the operating system to OL8, but before you upgrade to PowerFlow version 3.0.0 or later:

1. Use the "PowerFlow Backup" application in the PowerFlow user interface to create a backup file and send that file using secure copy protocol (SCP) to a destination system. For more information, see ***Backing up and Restoring PowerFlow Data***.

> **NOTE:** The backup and restore applications are application-level backup and restore tools. For full-system backups, you will need to do a filesystem-level backup to ensure that you get the encryption key that was used to encrypt configuration objects as well as other files used to describe the environment, including the **/etc/iservices** directory, the **docker-compose.yml** file and the **docker-compose-override.yml** file.

2. Install a "fresh" version of PowerFlow using the .iso file. For more information, see *Installing PowerFlow via ISO*.

> IMPORTANT: After installing the ISO, but before deploying PowerFlow using the script **/opt/iservices/scripts/pull_start_iservices.sh** or using pfctl **autocluster**, you must copy the old system **encryption_key** and **is_pass** files to the new nodes.

3. Use the "PowerFlow Restore" application in the PowerFlow user interface to retrieve your backup file (from step 1, above) from the remote system and restore its content. For more information, see *Restoring PowerFlow Data*.

Upgrading to Couchbase Version 6.6.0

This section contains a set of upgrade considerations for moving to Couchbase version 6.6.0 (Community Edition). Version 2.6.0 of the PowerFlow Platform includes Couchbase version 6.6.0 (Community Edition).

PowerFlow Supported Upgrade Paths

The following constraints are based on the Couchbase version used by the different versions of PowerFlow. For more information, see Couchbase Supported community upgrade paths.

Starting Versions	Path to Current Versions
PowerFlow 2.0.x (Couchbase 5.1.1)	2.0.x (Couchbase 5.1.1) -> 2.1.x to 2.5.x (Couchbase 6.0.2) -> 2.6.0 (Couchbase 6.6.0)
PowerFlow 2.1.x (Couchbase 6.0.2)	2.1.x (Couchbase 6.0.2) -> 2.6.0 (Couchbase 6.6.0)

Logs Buckets

When upgrading to Couchbase version 6.6.0, the number of documents in the logs bucket could make the upgrade take longer, as a namespace upgrade is needed.

ScienceLogic recommends that you flush the logs bucket if there are more than 300,000 documents that are taking up close to 2 GB of space in every node. Flushing the logs bucket will speed up the upgrade process. Otherwise, migrating a logs bucket of that size would take two to three minutes per node.

> WARNING: Do not interrupt the upgrade process, as that can corrupt documents. Please wait until the upgrade finishes running.

Run the following command to flush the logs bucket after the PowerFlow version 2.6.0 RPM was installed, but before redeploying the PowerFlow Stack:

```
pfctl --host <hostname><host_password>:<password> node-action --action
flush_logs_bucket
```

Alternately, you can flush the logs bucket manually using the Couchbase user interface.

Downgrading

Downgrades from Couchbase 6.6.x are not supported because the namespace is upgraded.

Upgrading from PowerFlow 3.x to the latest 3.x Version

> IMPORTANT: This topic is only relevant for users that are upgrading an existing version of PowerFlow 3.x.x to the latest 3.x.x version.

ScienceLogic releases a major update to PowerFlow every six months. ScienceLogic also releases a monthly maintenance release (MMR) as needed to address major customer-facing bugs. If there are no major bugs to be addressed via MMR, the MMR will not be produced for the month. Security updates are included in an MMR only if an MMR is planned to be released.

You should always upgrade to the most recent release of PowerFlow.

If you need the most recent, stable version of the Oracle Linux 8 operating system (OS) packages, you can upgrade them either using dnf update or by mounting the latest ISO to an existing PowerFlow system. For more information, see *Upgrading OS Packages*.

Deploying PowerFlow as a MUD System (Optional)

Starting with PowerFlow version 2.3.0, you can deploy PowerFlow as a Military Unique Deployment (MUD) system. Please note the following criteria:

- You cannot convert a non-MUD PowerFlow system to a MUD system.
- If you want to upgrade from an older (non-MUD) PowerFlow system to a MUD system, you will need to run a backup and restore to the new deployment.
- If you want to upgrade from an older 2.x MUD system to the latest 3.x line, you will need to run a backup and restore to the new deployment.
- Upgrading from a 3.x MUD system to the latest 3.x system is fully supported.

Considerations for Upgrading from PowerFlow 3.x

- PowerFlow version 3.1.x brings Python3.11 inside its containers. Upgrading SyncPack virtual environments from Python3.8 to Python3.11 is done automatically by the **syncpacks_steprunner** service after re-deploying the PowerFlow stack. This upgrade could take a moment. While upgrading, the steprunners could fail to execute some tasks because the SyncPack virtual environments won't be in place immediately. To avoid this, the steprunners replicas can be set to `0` temporarily in the **docker-compose.yml** file.
- If you made any modifications to the nginx configuration or to other service configuration files outside of the **docker-compose.yml**file, you will need to modify those custom configurations before upgrading, or contact ScienceLogic Support to prevent the loss of those modifications.

- If you are deploying PowerFlow without a load balancer, you can only use the deployed IP address as the management user interface. If you use another node to log in to the PowerFlow system, you will get an internal server error. Also, if the main deployed node is down, you must redeploy the system using the IP address for another active node to access the management user interface.
- For Military Unique Deployment (MUD) systems, ScienceLogic recommends using the `--resolve-image=never` argument when deploying the stack for a faster deployment:

  ```
  docker stack deploy --resolve-image never -c
  /opt/iservices/scripts/docker-compose.yml iservices
  ```

- If PoweFlow 3.0.0 was installed using the official PowerFlow ISO, the free space on the **isvg-root**(/) filesystem may need to be increased for installing the PowerFlow 3.1.x RPM, as this RPM requires approximately 9GB for its installation. When checking the current free space on the isvg-root filesystem use `df -h` . If the free space is less than 9GB use one or both of the following options to make space for the RPM installation.

Option 1: Increase the size of the isvg-root(/) filesystem

1. Run the command `sudo vgs`. This will show you how much storage is left for allocation.
2. If VFree is not more than 10Gg, you can do the following:
   - Increase the size of the physical disk. This depends on the virtualization solution you are using.
   - Run commands for Option 2 instead.
3. Run the command `sudo lvextend -L +10G /dev/isvg/root`.
4. Run the command `sudo xfs_growfs /dev/mapper/isvg-root`.
5. Verify that there is at least 9G of storage for the isvg-root filesystem with the command `df -h`.

Option 2: Remove the Old PowerFlow Images from the /opt/iservices/images directory

1. Remove the old compressed images from the **/opt/iservices/images** directory.
2. Verify that you have at least 9G storage for the new images with the command `df -h`.
3. If there is not enough storage, you will have to find other directories and files to remove to free up space.

Pre-Upgrade Steps

Before upgrading a PowerFlow system, check to ensure the system is up and healthy.

1. Be sure that the gui service is constrained to, and is running the nodes that are expected to receive traffic.
2. Run the **powerflowcontrol**(pfctl) **healthcheck** and **autoheal** actions to make sure the system is healthy.
3. For cluster environments, verify that there are three nodes in the Couchbase and RabbitMQ user interfaces.

4. If you have made custom changes, ensure they have been added to the **docker-compose-override.yml** file. Otherwise, they will be lost during the upgrade process. For more information, see the *Configuring PowerFlow Services* section.

    - If you enabled the `broker_load_from_backend` setting, make sure it is present in the **docker-compose-override.yml** file. After upgrading, make sure the setting is still enabled.
    - When upgrading from PowerFlow version 3.0.0 to the latest version, the steprunners docker healthcheck in the docker-compose-override.yml file must be updated to the following, which is compatible with PowerFlow version 3.0.0 and later:

      ```
      ["CMD-SHELL", "celery -A ipaascommon.celeryapp:app inspect ping -d celery@$${HOSTNAME}"]
      ```

    - Be sure to run the **powerflowcontrol**(pfctl) **autoheal** action to copy the compose files to all nodes in cluster environments.

5. Create the following backups:

    - Use the "Backup" application. For more information, see the *Backing up and Restoring PowerFlow Data* section.
    - Make backups of the **/etc/iservices/**, **/opt/iservices/scripts/docker-compose.yml**, and **/opt/iservices/scripts/docker-compose-override.yml** files.

Locating the RPM or ISO File for Upgrading

CAUTION: As a best practice, you should *always* upgrade to the most recent version of PowerFlow that is currently available at https://support.sciencelogic.com/s/.

To locate and download the PowerFlow RPM file:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *PowerFlow*. The **PowerFlow** page appears.
3. Click the link to the current release. The **Release Version** page appears.
4. In the **Release Files** section, click the RPM or ISO link. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page.

Upgrading OS Packages

If you need the most recent, stable versions of the Oracle Linux 8 operating system (OS) packages, you can upgrade them either using the `dnf update` command or by mounting the latest PowerFlow ISO to an existing PowerFlow system. For more information, see the section on *Upgrading OS Packages (for Offline Deployments Only)*.

When a `dnf update` is performed, there is no risk of PowerFlow operations being affected as long as Docker or networking services are not included in the updates. If these packages are updated, you must restart Docker in all nodes.

Upgrading OS Packages (for Offline Deployments Only)

Upgrading OS packages for an offline deployment requires you to mount the ISO and update the packages that are shipped with the ISO.

1. Mount the PowerFlow ISO onto the system:

```
mkdir /mnt/tmp_install_mount
```

```
mount -o loop /dev/cdrom /mnt/tmp_install_mount
```

2. After you mount the ISO, add a new repository file to access the ISO as if it were a yum repository. Create a **/etc/yum.repos.d/localiso.repo** file with the following contents:

```
[ol8_baseos_latest_offline]
name=Oracle Linux 8 BaseOS Latest PF ISO
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
baseurl=file:///mnt/tmp_install_mount/BaseOS
gpgcheck=0
enabled=1

[ol8_appstream_offline]
name=Oracle Linux 8 Application Stream PF ISO
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
baseurl=file:///mnt/tmp_install_mount/AppStream
gpgcheck=0
enabled=1
```

   After you create and save this file, the Linux system can install packages from the PowerFlow ISO.

3. To disable other repos except the local ISO mount repo, run the following command:

```
dnf --disablerepo="*" --enablerepo="ol8_baseos_latest_offline,ol8_
appstream_offline" update
```

4. Run the following command to update and install the host-level packages:

```
dnf update
```

Upgrading from Version 3.x.x to 3.2.0

Depending on your PowerFlow environment and your company's requirements, select one of the following upgrade options:

- Single-node Upgrade
- Cluster Upgrade with Short Downtime
- Rolling Cluster Upgrade with No Downtime

WARNING: Perform the steps in the following procedure in the order listed below to ensure a proper upgrade.

Single-node Upgrade

To upgrade a single-node PowerFlow system from 3.x.x:

1. Make a copy of the **docker-compose** file that you used to deploy PowerFlow (in case you need to roll back to the previous version). Follow the *pre-upgrade steps*.
2. Either go to the console of the PowerFlow system or use SSH to access the server.
3. Log in as isadmin with the appropriate (root) password. You must be root to upgrade using the RPM file.
4. Download the PowerFlow RPM and copy the RPM file to the PowerFlow system.
5. Type the following in the command line, where `<full_path_of_rpm>` is the name and path of the RPM file, such as `/home/isadmin/sl1-powerflow-3.x.x-1.x86_64`:

```
sudo dnf upgrade <full_path_of_rpm>
```

NOTE: If the disk space to install the new RPM is not enough in the root partition, try removing the old images from the directory **/opt/iservices/images** or increase the size of the **isvg-root**(/) filesystem. For more information, see the section on *Considerations for Upgrading from 3.x*

6. After the RPM is installed, run the following Docker command:

```
docker stack rm iservices
```

After running this command, the stack is no longer running.

NOTE: If you want to upgrade your services in place, without bringing them down, you may skip this step. Please note that skipping this step might cause the services to take longer to update. For MUD environments, you must remove the stack, as there are temporal volumes that need to be recreated.

7. ***If the upgrade process recommends restarting Docker***, run the following command:

```
systemctl restart docker
```

NOTE: If you restart Docker for this step, you should skip step 10, below.

8. Re-deploy the Docker stack to update the containers:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

9. After you re-deploy the Docker stack, the services automatically update themselves. Wait a few minutes to ensure that all services are updated and running before using the system.

10. ***If the upgrade process recommends restarting Docker***, run the following command:

```
systemctl restart docker
```

11. To view updates to each service and the service versions for services throughout the whole stack, type the following at the command line:

```
docker service ls
```

    Each service now uses the new version of PowerFlow. Make sure to run the pfctl **healthcheck** and **autoheal** actions to finish the upgrade process.

Cluster Upgrade with Short Downtime

If you are running PowerFlow in a clustered environment, you should install the RPM on all nodes in the cluster for the upgrade. The order in which you install the RPM on the nodes does not matter. Installing the RPM on each node makes the latest PowerFlow container images and scripts available to the system.

Please note that installing the RPM on the nodes does not change the version of the currently running PowerFlow application stack. The new version of PowerFlow is only deployed when you run the `docker stack deploy` command on the new **docker-compose** file that is generated after you install the RPM.

The following upgrade procedure for a clustered PowerFlow environment results in only five to ten minutes of downtime.

For more information, including extensive examples, see*PowerFlow Multi-tenant Upgrade Process* in *Appendix B: Configuring the PowerFlow System for Multi-tenant Environments*.

To perform a cluster upgrade with short downtime:

1. Make a copy of the **docker-compose** file that you used to deploy PowerFlow (in case you need to roll back to the previous version). Follow the ***pre-upgrade steps***.
2. Use SSH to access the node.
3. Log in as ***isadmin*** with the appropriate (root) password. You must be root to upgrade using the RPM file.
4. Copy the RPM file to each node.

5. Install the RPM on all nodes in the cluster by typing the following command at the command line for each node:

```
sudo yum upgrade <full_path_of_rpm>
```

where *full_path_of_rpm* is the name and path of the RPM file, such as **/home/isadmin/sl1-powerflow-3.x.x-1.x86_64**. For more information, see *Installing the PowerFlow RPM*. If there is not enough space to install the new RPM in the root partition, try removing the tar images from the directory **/opt/iservices/images**.

> **NOTE:** Installing the RPM on all the nodes makes the containers available and updates the **docker-compose** file on each node, but the existing PowerFlow version will continue running.

6. After you have installed the RPM on all of the nodes, open the new **docker-compose.yml** file at **/opt/iservices/scripts/** and confirm that the versions of Couchbase, RabbitMQ, and any custom workers are using the latest, updated version numbers.
7. After the RPM is installed on the nodes, run the following Docker command:

```
docker stack rm iservices
```

After you run this command, the stack is no longer running.

> **NOTE:** If you want to upgrade your services in place, without bringing them down, you may skip this step. Please note that skipping this step might cause the services to take longer to update.

8. ***If the upgrade process recommends restarting Docker***, run the following command:

```
systemctl restart docker
```

> **NOTE:** If you restart Docker for this step, you should skip step 11, below.

9. Re-deploy the Docker stack with the **docker-compose** file you reviewed in step 6:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

The time between removing the old version of the stack and deploying the new version of the stack is the only period of down time.

10. After you re-deploy the Docker stack, the services automatically update themselves. Wait a few minutes to ensure that all services are updated and running before using the system.
11. ***If the upgrade process recommends restarting Docker***, run the following command:

```
systemctl restart docker
```

12. To view updates to each service and the service versions for services throughout the whole stack, type the following at the command line:

```
docker service ls
```

Each service now uses the new version of PowerFlow.

Rolling Cluster Upgrade with No Downtime

For a clustered PowerFlow environment, the following rolling cluster update results in no downtime, but the process requires intermediate compose updates.

For more information, including extensive examples, see *PowerFlow Multi-tenant Upgrade Process* in *Appendix B: Configuring the PowerFlow System for Multi-tenant Environments*.

To perform a rolling cluster upgrade with no downtime:

1. Make a copy of the **docker-compose** file that you used to deploy PowerFlow (in case you need to roll back to the previous version). Follow the *pre-upgrade steps*.
2. Use SSH to access the node.
3. Log in as ***isadmin*** with the appropriate (root) password. You must be root to upgrade using the RPM file.
4. Copy the RPM file to each node.
5. Install the RPM on all nodes in the cluster by typing the following command at the command line for each node:

```
sudo yum upgrade <full_path_of_rpm>
```

where *full_path_of_rpm* is the name and path of the RPM file, such as **/home/isadmin/sl1-powerflow-3.x.x-1.x86_64**. For more information, see *Installing the PowerFlow RPM*.

> **NOTE:** Installing the RPM on all the nodes makes the containers available and updates the **docker-compose** file on each node, but the existing PowerFlow version will continue running.

6. After the RPM is installed on the nodes, remove the "core nodes" one by one to cause a failover, and then re-add a new version of the same node without taking down the stack:
    a. Access the Couchbase user interface at ***https://<IP of PowerFlow>:8091***.
    b. On the **[Servers]**tab, select a single database node and click **Failover**. Select **Graceful Failover**. Manually failing over before updating ensures that the system is still operational when the container comes down.
    c. Modify the **/opt/iservices/scripts/docker-compose.yml** file at that you used to deploy PowerFlow, and change just one of the Couchbase services and RabbitMQ services to use the new version (the same Couchbase server you previously failed over).
    d. Deploy the **docker-compose** file with the new updated Couchbase server.
    e. Wait for the new instance of Couchbase to join the existing cluster as the latest version. When it has joined, that core node is updated.

7. Continue failing over nodes and re-deploying them with the new PowerFlow version until all core nodes are updated.
8. After the core nodes are updated, you can proceed similarly with each individual worker nodes. You can update these nodes in groups if that is faster. You do not need to fail over the worker nodes; you can just change the services and images.
9. At the end of the node-by-node upgrade, your existing **docker-compose** should contain all of the new versions specified by the latest **docker-compose** file shipped with the RPM.

Validating the PowerFlow System Post-Upgrade

Perform the following tasks to ensure the cluster configuration is valid for this version of PowerFlow:

1. Be sure that the **gui** service is accessible.
2. Run the **powerflowcontrol**(pfctl) **healthcheck** and **autoheal** actions to make sure the system is healthy.
3. Restart the **contentapi** service if the **powerflowcontrol**(pfctl)**autoheal** action suggests to do so.
4. Check that you can access the Couchbase(8091) and RabbitMQ(15672) services, and the expected nodes are there. For cluster environments 3 nodes should be present.
5. Check the Couchbase logs. If there are errors with the load powerflow content command, execute the following command:

```
pfctl --host <host_IP_1> user:host_password node-action --action
upload_syncpack_default_content
OR
docker exec -it $(docker ps -q -n 1 --filter name=iservices_
couchbase) couchcontrol load powerflow-content
```

6. Check the RabbitMQ logs. If they are not present in the user interface, restart node one by executing the following command:

```
docker service update --force iservices_rabbitmq
```

7. ScienceLogic recommends installing the latest version of the base SyncPacks that were uploaded during the upgrade:
    - Base Steps SyncPack
    - SL1 PowerFlow System Utils SyncPack
    - SL1 PowerFlow powerflowcontrol (pfctl) Command-Line Utility SyncPack

Troubleshooting Upgrade Issues

The following topics describe issues that might occur after the upgrade to version 2.2.0 or later, and how to address those issues.

After upgrading, the syncpacks_steprunner service fails to run

This error flow tends to happen when the **syncpacks_steprunner** service is deployed, but the database is not yet updated with the indexes necessary for the SyncPack processes to query the database. In most deployments, the index should be automatically created. If the index is not automatically created, which it might do in a clusterd configuration, you can resolve this issue by manually creating the indexes.

In this situation, if you check the logs, you will most likely see the following message:

```
couchbase.exceptions.HTTPError: <RC=0x3B[HTTP Operation failed. Inspect
status code for details], HTTP Request failed. Examine 'objextra' for full
result, Results=1, C Source=(src/http.c,144), OBJ=ViewResult<rc=0x3B[HTTP
Operation failed. Inspect status code for details], value={'requestID':
'57ad959d-bafb-46a1-9ede-f80f692b0dd7', 'errors': [{'code': 4000, 'msg':
'No index available on keyspace content that matches your query. Use
CREATE INDEX or CREATE PRIMARY INDEX to create an index, or check that
your expected index is online.'}], 'status': 'fatal', 'metrics':
{'elapsedTime': '5.423085ms', 'executionTime': '5.344487ms',
'resultCount': 0, 'resultSize': 0, 'errorCount': 1}}, http_status=404,
tracing_context=0, tracing_output=None>, Tracing Output={":nokey:0":
null}>
```

To address this issue, wait a few minutes for the index to be populated. If you are still getting an error after the database has been running for a few minutes, you can manually update the indexes by running the following command inside the couchbase container:

```
couchcontrol index update-secondary --file /tmp/scripts/couchbase_
index.json
```

> **NOTE:** Creating a primary index is only for troubleshooting, and primary indexes should not be left on the system.

SyncPack virtual environments were not recreated

This error can occur if the database was up when the SyncPack steprunners were starting. This is likely to occur after restoring a backup. In this situation, if you check the logs for steprunners, you will likely see the following errors:

```
ipaascommon.ipaas_exceptions.MissingModule: Step requires module system_
utils_syncpack but it's not available in the environment
```

Execute the following commands to restart the **syncpacks_steprunner** so that the SyncPack virtual environments can be recreated:

```
docker service rm iservices_syncpacks_steprunner
```

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml iservices
--resolve-image=never
```

SyncPacks cannot be installed after upgrading from PowerFlow version 3.0.0

In environments where network latency is high, the database may not be able to initialize and apply the new content updates needed for installing SyncPacks. You will see the following error in the steprunners logs when trying to install a SyncPack using the user interface:

```
|   File "InstallSyncpack", line 181, in install_syncpack
|   File "/usr/local/lib/python3.11/site-pack-
ages/couchbase/logic/wrappers.py", line 102, in wrapped_fn
|            ^^^
|   File "InstallSyncpack", line 181, in install_syncpack
|     raise excptn from None
| couchbase.exceptions.InternalSDKException: InternalSDKException(<mes-
sage='SubDocOp' object is not subscriptable>)
|   File "/usr/local/lib/python3.11/site-pack-
ages/couchbase/logic/wrappers.py", line 102, in wrapped_fn
|     raise excptn from None
| couchbase.exceptions.InternalSDKException: InternalSDKException(<mes-
sage='SubDocOp' object is not subscriptable>)
```

Execute the following command in the main node to upload the content, or run the **powerflowcontrol**(pfctl) **autoheal** action if you are using pfctl version 2.8.0:

```
docker exec -it $(docker ps -q -n 1 --filter name=iservices_couchbase)
couchcontrol load powerflow-content
```

Validate that the content was uploaded successfully by executing the following command:

```
pfctl --host <host_IP_1> user:host_password node-action --action verify_
install_syncpack_content
```

The PowerFlow user interface displays an unauthorized user error

The Content API service may have started before the database started. You can run the **powerflowcontrol**(pfctl) **autoheal** actions and restart the API service.

Licensing PowerFlow

Before users can access all of the features of PowerFlow, the *Administrator* user must license the PowerFlow instance through the ScienceLogic Support site. For more information about accessing PowerFlow files at the ScienceLogic Support site, see the following Knowledge Base article: SL1 PowerFlow Download and Licensing.

When you log in to the PowerFlow system, a notification appears at the bottom right of the screen that states how much time is left in your PowerFlow license. The notification displays with a green background if your license is current, yellow if you have ten days or less in your license, and red if your license has expired. You need to click the Close icon (✕) to close this notification.

You can also track your licensing information on the **About** page (username menu > About). You can still log into a system with an expired license, but you cannot create or schedule PowerFlow applications.

NOTE: The administrator and all users cannot access certain production-level capabilities until the administrator licenses the instance. For example, users cannot create schedules or upload PowerFlow applications and steps that are not part of a SyncPack until PowerFlow has been licensed.

TIP: If you are not deploying PowerFlow on a production or pre-production environment, you can skip the licensing process.

NOTE: If you are licensing a PowerFlow High Availability cluster, you can run the following licensing process on *any* node in the cluster. The node does not have to be the leader, and the licensing process does not have to be run on all nodes in the Swarm.

Licensing a PowerFlow System

To license a PowerFlow system:

1. Run the following command on your PowerFlow system to generate the **.iskey** license file:

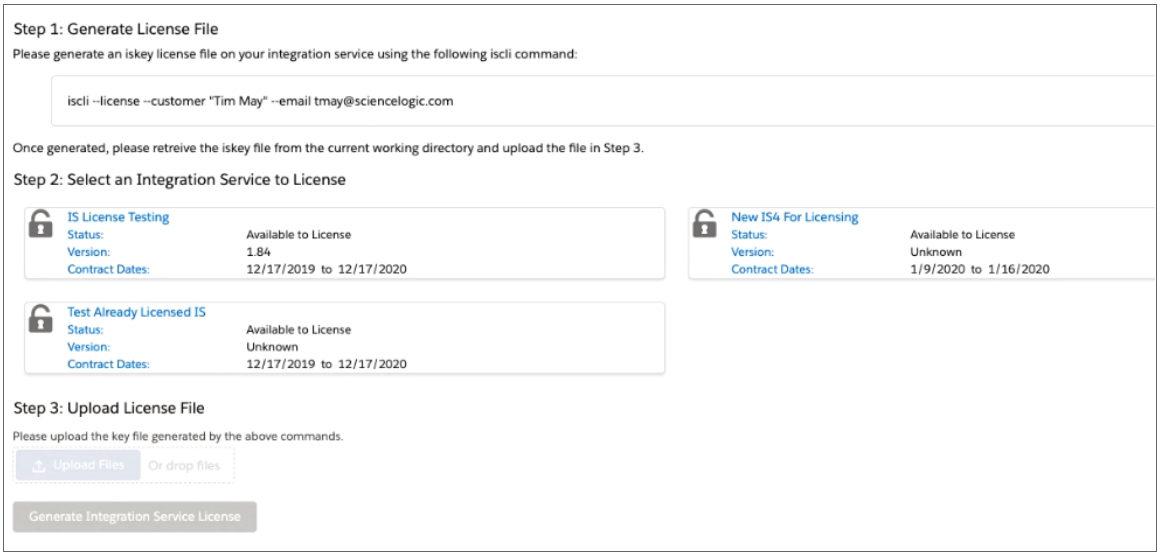
   ```
   iscli --license --customer "<user_name>" --email <user_email>
   ```

   where `<user_name>` is the first and last name of the user, and `<user_email>` is the user's email address. For example:

   ```
   iscli --license --customer "John Doe" --email jdoe@sciencelogic.com
   ```

2. Run an `ls` command to locate the new license file: **customer_key.iskey**.
3. Using WinSCP or another utility, copy the **.iskey** license file to your local machine.

4. Go to the **PowerFlow License Request** page at the ScienceLogic Support site: https://support.sciencelogic.com/s/integration-service-license-request:

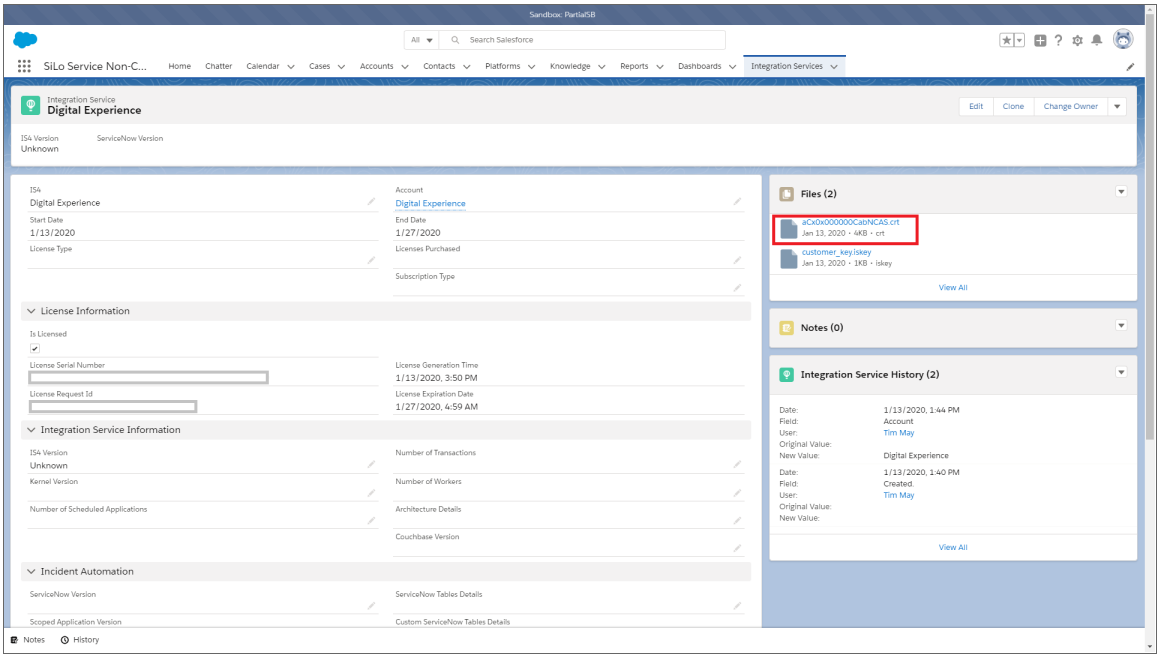


5. For Step 2 of the "Generate License File" process, select the PowerFlow record you want to license.

> **TIP:** You already covered Step 1 of the "Generate License File" process in steps 1-3 of this procedure.

6. Scroll down to Step 3 of the "Generate License File" process and upload the **.iskey** license file you created in steps 1-3 of this procedure.
7. Click **[Upload Files]**.

8. After uploading the license file, click **[Generate PowerFlow License]**. A new **Licensing** page appears:



9. Click the **.crt** file in the **Files** pane to download the new **.crt** license file.
10. Using WinSCP or another file-transfer utility, copy the **.crt** license file to your PowerFlow system.
11. Upload the **.crt** license file to the PowerFlow server by running the following command on that server:

```
iscli -l -u -f ./<license_name>.crt -H <IP_address> -U <user_name> -p
<user_password>
```

where `<license_name>` is the system-generated name for the **.crt** file, `<IP_address>` is the IP address of the PowerFlow system, `<user_name>` is the user name, and `<user_password>` is the user password. For example:

```
iscli -l -u -f ./aCx0x000000CabNCAS.crt -H 10.2.33.1 -U isadmin -p
passw0rd
```

NOTE: ScienceLogic determines the duration of the license key, not the customer.

TIP: If you have any issues licensing your PowerFlow system, please contact your ScienceLogic Customer Success Manager (CSM) or open a new Service Request case under the "Integration Service" category.

Licensing Solution Types

The licensing for the PowerFlow platform was separated into three solution types:

- ***Standard***: This solution lets you import and install SyncPacks published by ScienceLogic and ScienceLogic Professional Services, and to run and schedule PowerFlow applications from those SyncPacks. You cannot customize or create PowerFlow applications or steps with this solution type. Features that are not available display in gray text in the user interface.
- ***Advanced***: This solution contains all of the Standard features, and you can also build your own SyncPacks and upload custom applications and steps using the command-line interface. You can create PowerFlow applications using the PowerFlow command-line interface, but you cannot create and edit applications or steps using the PowerFlow builder in the user interface.
- ***Premium***: This solution contains all of the Advanced features, and you can also use the PowerFlow builder, the low-code/no-code, drag-and-drop interface, to create and edit PowerFlow applications and steps.

A yellow text box appears in the PowerFlow user interface when the license is close to expiring, displaying how many days are left before the license expires. The license status and expiration date also displays on the **About** page in the PowerFlow user interface.

An unlicensed system will not be able to create PowerFlow applications, steps, or schedules. Unlicensed systems will only be able to run applications that are installed manually through SyncPacks.

Features that are locked by licensing solution type are grayed out. If you click on a grayed-out feature, the user interface will display a notification prompting you to upgrade your license.

Configuring PowerFlow Services

PowerFlow systems use docker-compose and docker-compose-override files to deploy the Docker swarm with PowerFlow services. The default compose and override files are available when the PowerFlow ISO or RPM is installed, and are located in the **/opt/iservices/scripts** directory.

PowerFlow uses the docker-compose-override.yml file to persistently store user-specific configurations for containers, such as proxy, replica, and additional node settings, as well as deployment constraints. User-specifc changes are kept in this file so that they can be reapplied when the **/opt/iservices/scripts/docker-compose.yml** file is completely replaced during an RPM upgrade, ensuring that no user-specific configurations are lost. By default, only core services are included in the **docker-compose-override.yml** file. If extra services need to be added, they should be included as needed.

Applying User-Specific Configurations

To apply user-specific configurations:

1. Either go to the console for the PowerFlow system, or use SSH to access the PowerFlow server.
2. Log in as an isadmin using the appropriate password.
3. Using a text editor, edit the **/opt/iservices/scripts/docker-compose-override.yml** file. For information about editing the compose-override file, see the *Updating the docker-compose-override File* section.
4. Save the settings in the file, and then run the **/opt/iservices/scripts/compose_override.sh** script.

NOTE: The **compose_override.sh** script validates that the configured **docker-compose.yml** and **docker-compose-override.yml** files are syntactically correct. If the settings are correct, the script applies the settings to the existing **docker-compose.yml** file that is used to actually deploy.

This script also updates the images to use jinja2 syntax, in order to avoid mismatching versions of the Couchbase, RabbitMQ and steprunners services, all of which are services that can have more than one service defined, especially for High Availability environments.

5. Redeploy the updated services by executing the following commands. For example, if the steprunner service was updated:

```
docker service rm iservices_steprunner
```

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

Updating the docker-compose-override File

When creating a cluster using the **powerflowcontrol** (pfctl) **autocluster** action, configurations for a three node cluster deployment are added to the compose-override file, so that configurations are not lost during an upgrade. The pfctl **autoheal** action also applies some other configurations as needed.

Adding User-Specific Configurations

Check the docker-compose file for the syntax of the services being used. Also check the official Docker documentation to be familiar with the syntax used.

Scenario 1

To add configurations to services that already exist in the compose-override file, edit the file, go to the service, and add the configuration. For example, when adding an extra environment variable to the **contentapi** service:

```
services:
  contentapi:
    environment:
      db_host: couchbase.isnet,couchbase-worker.isnet,couchbase-work-
er2.isnet
      broker_load_from_backend: 'true'
    deploy:
      replicas: 3
```

Scenario 2

If the service to which the new configuration needs to be added is not in the compose-override file, add it as a new service, and add the configuration needed. Be sure the tree level is set as expected. For example, when adding the syncpacks_steprunner with an environment variable:

```
services:
  syncpacks_steprunner:
    environment:
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet
```

Scenario 3

To add a new service recommended by PowerFlow, such as custom steprunners, use the `steprunner` service as a base and edit configurations, such as `user_queues`. The custom service name must start with the prefix `steprunner_`, and the image name must be set as `'{{ services.steprunner.image }}'` so the correct image version is used. For example, when adding the steprunner_custom service using a custom queue called custom_queue that has 15 replicas:

```
  steprunner_custom:
    deploy:
      placement:
        max_replicas_per_node: 5
      replicas: 15
      resources:
        limits:
          memory: 2G
    environment:
      PIP_CONFIG_FILE: /usr/tmp/pip.conf
      additional_worker_args: ' --max-tasks-per-child 1 '
      broker_url: pyamqp://guest@rabbit//
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet
      logdir: /var/log/iservices
      result_backend: redis://redis:6379/0
      user_queues: custom_queue
    image: '{{ services.steprunner.image }}'
    networks:
      isnet: {}
    read_only: true
    secrets:
    - source: encryption_key
    - source: is_pass
    volumes:
    - /var/log/iservices:/var/log/contentapi:rw
    - /var/log/iservices:/var/log/iservices:rw
    - read_only: true
      source: syncpacks_virtualenvs
```

```
      target: /var/syncpacks_virtualenvs
      type: volume
```

The examples above use Jinja2 syntax for the image, which allows the file to be flexible and prevents mismatches during future upgrades. For more complex examples of how to use Jinja2 in the **docker-compose-override.yml** file, see the *Using Jinja2 in the compose-override File* section.

> **NOTE:** To add a new service that is not present in the PowerFlow documentation, you must specify the name of the image and a unique service name for all needed configurations.

Using Jinja2 in the compose-override File

For detailed information on Jinja2 syntax, see the official Jinja2 documentation. Before using Jinja2 syntax in the compose-override file, be aware of the following points:

- The docker-compose.yml file serves as data that the Jinja2 template uses for rendering information.
- Only the **docker-compose-override.yml** file can include Jinja2 syntax.
- Basic Jinja2 syntax, such as replacing strings, arrays, and numbers will work as expected, but more complex configurations should be tested before use. You can test configuration updates by running the **/opt/iservices/scripts/compose_override.sh** script, which will show an error if the configuration did not work.

An example using Jinja syntax

The **docker-compose.yml** file, acting as the data file:

```
  steprunner:
    .... ....
    image: registry.scilo.tools/sciencelogic/pf-worker:rhel3.2.0
    healthcheck:
      interval: 1m
      retries: 5
      start_period: 2m
      test:
      - CMD-SHELL
      - celery -A ipaascommon.celeryapp:app inspect ping -d cel-
ery@$${HOSTNAME}
      timeout: 20s
```

The **docker-compose-override.yml** file, acting as the template file. It will render the image and healthcheck configuration for the new `steprunner_custom` service:

```
  steprunner_custom:
    ... .. ..
```

```
    healthcheck: {{ services.steprunner.healthcheck }}
    image: '{{ services.steprunner.image }}'
```

The resulting **docker-compose.yml** file, after running the **/opt/iservices/scripts/compose_override.sh** script:

```
  steprunner_custom:
    .......
    healthcheck:
      interval: 1m
      retries: 5
      start_period: 2m
      test:
      - CMD-SHELL
      - celery -A ipaascommon.celeryapp:app inspect ping -d cel-
ery@$${HOSTNAME}
      timeout: 20s
    image: registry.scilo.tools/sciencelogic/pf-worker:rhel3.2.0
```

Configuring a Proxy Server

To configure PowerFlow to use a proxy server:

1. Either go to the console of the PowerFlow system or use SSH to access the PowerFlow server.
2. Log in as *isadmin* with the appropriate password.
3. Using a text editor like vi, edit the file **/opt/iservices/scripts/docker-compose-override.yml**.

> **NOTE:** PowerFlow uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/scripts/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost. By default only main core services are included in the **docker-compose-override.yml** file, if extra services need to be added they should be included as needed.

4. In the **environment** section of the **steprunner** service, add the following lines:

```
services:
  steprunner:
    environment:
       https_proxy: "<proxy_host>"
       http_proxy: "<proxy_host>"
       no_proxy: ".isnet"
```

WARNING: If your proxy appears to only use HTTP and not HTTPS, you will need to use `http` in both `https_proxy` environment variables.

TIP: If you do not want to use more than one proxy location, you can use the `no_proxy` setting to specify all of the locations, separated by commas and surrounds by quotation marks. For example: `no_proxy: ".isnet,10.1.1.100,10.1.1.101"`

NOTE: If you want to access external pypi packages while using a proxy, be sure to include `pypi.org` and `files.pythonhosted.org` to this section to ensure the proxy enables those locations.

5. In the **environment** section of the **syncpacks_steprunner** service, add the following lines. Add the **syncpacks_steprunner** service to the **docker-compose-override.yml** file if it's not present there::

```
services:
  syncpacks_steprunner:
    environment:
       https_proxy: "<proxy_host>"
       http_proxy: "<proxy_host>"
       no_proxy: ".isnet"
```

WARNING: If your proxy appears to only use HTTP and not HTTPS, you will need to use `http` in both `https_proxy` environment variables.

NOTE: If you want to access external pypi packages while using a proxy, be sure to include `pypi.org` and `files.pythonhosted.org` to this section to ensure the proxy enables those locations.

6. Save the settings in the file and then run the **/opt/iservices/scripts/compose_override.sh** script.

> **NOTE:** The **compose_override.sh** script validates that the configured **docker-compose.yml** and **docker-compose-override.yml** files are syntactically correct. If the settings are correct, the script applies the settings to your existing **docker-compose.yml** file that is used to actually deploy.

7. Re-deploy the steprunners to use this change by typing the following commands:

```
docker service rm iservices_steprunner
```

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

Changing the PowerFlow System Password

The PowerFlow system uses two primary passwords. For consistency, both passwords are the same after you install PowerFlow, but you can change them to separate passwords as needed.

> **IMPORTANT:** To avoid authentication issues, do not use the dollar sign ($) character in any of the passwords related to PowerFlow.

PowerFlow uses the following passwords:

- ***The PowerFlow Administrator (isadmin) user password***. This is the password that you set during the PowerFlow *ISO installation process*, and it is only used by the default local Administrator user (*isadmin*). You use this password to log in to the PowerFlow user interface and to verify API requests and database actions.
  This password is set as both the "Linux host isadmin" user and in the **/etc/iservices/is_pass** file that is mounted into the PowerFlow stack as a "Docker secret". Because it is mounted as a secret, all necessary containers are aware of this password in a secure manner.
  Alternatively, you can enable third-party authentication, such as LDAP or AD, and authenticate with credentials other than isadmin. However, you will need to set the user policies for those LDAP users first with the default isadmin user. For more information, see *Managing Users in PowerFlow*.
- ***The Linux Host OS SSH password***. This is the password you use to SSH and to log in to isadmin. You can change this password using the standard Linux `passwd` command or another credential management application to manage this user.
  You can also disable this Linux user and add your own user if you want. The PowerFlow containers and applications do not use or know this Linux login password, and this password does not need to be the same between nodes in a cluster. This is a standard Linux Host OS password.

Updating the PowerFlow Administrator (isadmin) user password

There are two ways to update the PowerFlow Administrator (isadmin) user password. ScienceLogic recommends using the powerflowcontrol (pfctl) tool if possible.

Starting in PowerFlow version 3.0.0, you can use the following command to update the PowerFlow Administrator (*isadmin*) user password:

```
pfctl --host pf_node_ip '<isadmin:host_password>' password set -p '<new_
password>'
```

NOTE: The old application (UI) password for the PowerFlow Administrator (isadmin) does not need to be provided in the command.

You can use the following command to update the PowerFlow Administrator (isadmin) user password in cluster environments. Use all of the PowerFlow nodes in the command.

For example, if you have 3 nodes, provide the hosts, users, and passwords of the 3 nodes .

```
pfctl ---host node1_ip '<isadmin:host_password>' ---host node2_ip
'<isadmin:host_password>' ---host node3_ip '<isadmin:host_password>'
password set -p '<new_password>'
```

NOTE: The password must be at least six characters, no more than 24 characters, and all special characters are supported except the dollar sign ($) character.

This command replaces the `ispasswd` script from earlier releases of PowerFlow, which was found in **/opt/iservices/scripts/ispasswd**. The `ispasswd` script will be deprecated in a future release.

The **/etc/iservices/is_pass** file is automatically updated in all the nodes that were provided in the pfctl command.

After the password is correctly updated, make sure the stack is removed and redeployed:

1. Remove the stack using the following command:

   ```
   docker stack rm iservices
   ```

2. Wait until the services are down. Check with the following command:

   ```
   watch docker ps
   ```

3. Redeploy the stack using the following command:

   ```
   docker stack deploy --resolve-image=never -c
   /opt/iservices/scripts/docker-compose.yml iservices
   ```

Next, run the powerflowcontrol (pfctl) **healthcheck** and **autoheal node** or **cluster** actions to make sure the application is healthy.

Updating the PowerFlow Administrator (isadmin) User Password with the ipasswd Script

To change the PowerFlow Administrator (*isadmin*) user password using the `ispasswd` script:

1. You can change the mounted isadmin password secret (which is used to authenticate via API by default) and the Couchbase credentials on the PowerFlow stack by running the `ispasswd` script on any node running PowerFlow in the stack:

   ```
   /opt/iservices/scripts/ispasswd
   ```

2. Follow the prompts to reset the password. The password must be at least six characters and no more than 24 characters, and all special characters are supported except the dollar sign ($) character.

> **NOTE:** Running the `ispasswd` script automatically changes the password for all PowerFlow application actions that require credentials for the isadmin user.

3. If you have multiple nodes, copy **/etc/iservices/is_pass** file, which was just updated by the `ispasswd` script, to all other manager nodes in the cluster. You need to copy this password file across all nodes in case you deploy from a different node than the node where you changed the password. The need to manually copy the password to all nodes will be removed in a future release of PowerFlow.

Configuring Security Settings

This topic explains how to change the HTTPS certificate used by PowerFlow, and it also describes password and encryption key security.

Changing the HTTPS Certificate

The PowerFlow user interface only accepts communications over HTTPS. By default, HTTPS is configured using an internal, self-signed certificate.

You can specify the HTTPS certificate to use in your environment by mounting the following two files in the user interface (**gui**) service:

- **/etc/iservices/is_key.pem**
- **/etc/iservices/is_cert.pem**

The SSL certificate for the PowerFlow system only requires the **HOST_ADDRESS** field to be defined in the certificate. That certificate and key must be identical across all nodes. If needed, you can also add non-HOST_ADDRESS IPs to the **Subject Alternative Name** field to prevent an insecure warning when visiting the non-HOST_ADDRESS IP.

NOTE: If you are using a load balancer, the certificates installed on the load balancer should use and provide the hostname for the load balancer, not the PowerFlow nodes. The SSL certificates should always match the IP or hostname that exists in the **HOST_ADDRESS** setting in **/etc/iservices/isconfig.yml**. If you are using a load balancer, the **HOST_ADDRESS** must also be the IP address for the load balancer.

NOTE: If you are using a clustered configuration for PowerFlow, you will need to copy the key and certificate to the same location on the node.

To specify the HTTPS certificate to use in your environment:

1. Copy the key and certificate files to all PowerFlow hosts that are part of the cluster.
2. Ensure the ownership of the key and certificate files are set to `UID 998` and `GID 996`, as required inside the **gui** container, and modify the permissions to `640` to grant access to the specified user and group. Execute the following command in the PowerFlow host:

   ```
   sudo chown 998:996 key_file_path cert_file_path
   ```

   ```
   sudo chmod 640 key_file_path cert_file_path
   ```

3. Modify the **/opt/iservices/scripts/docker-compose-override.yml** file and mount a volume to the **gui** service. The following code is an example of the volume specification:

   ```
   volumes:
     - "<path to IS key>:/etc/iservices/is_key.pem"
     - "<path to IS certificate>:/etc/iservices/is_cert.pem"
   ```

   where:

   - *<path to IS key>* is the path to the key in the PowerFlow host
   - *<path to IS certificate>* is the path to the certificate in the PowerFlow host

   NOTE: Do not change the text after the colons (`:/etc/iservices/is_key.pem` and `:/etc/iservices/is_cert.pem`), which are the paths to the key and certificate within the container.

   TIP: The location of the key and certificate files in the PowerFlow host does not need to be the same as within the container. It can be in a different location, such as `/home/isadmin`.

4. Run the following script to validate and apply the change to the **/opt/iservices/scripts/docker-compose.yml** file:

```
/opt/iservices/scripts/compose_override.sh
```

> **NOTE:** The **compose_override.sh** script validates that the configured **docker-compose.yml** and **docker-compose-override.yml** files are syntactically correct. If the settings are correct, the script applies the settings to your existing **docker-compose.yml** file that is used to actually deploy.

5. Review the **/opt/iservices/scripts/docker-compose.yml** file and make sure the new volume is set for the **gui** service.
6. Re-deploy the **gui** service by running the following commands:

```
docker service rm iservices_gui
```

```
docker stack deploy --resolve-image=never -c
/opt/iservices/scripts/docker-compose.yml iservices
```

Using Password and Encryption Key Security

When you install the PowerFlow platform, you specified the PowerFlow root password. This root password is also the default *isadmin* password:

- The root/admin password is saved in a root read-only file here: **/etc/iservices/is_pass**
- A backup password file is also saved in a root read-only file here: **/opt/iservices/backup/is_pass**

The user-created root password is also the default PowerFlow password for couchbase (:8091) and all API communications. The PowerFlow platform generates a unique encryption key for every platform installation:

- The encryption key exists in a root read-only file here: **/etc/iservices/encryption_key**
- A backup encryption key file is also saved in a root read-only file here: **/opt/iservices/backup/encryption_key**

> **NOTE:** This encryption key is different from the HTTPS certificate key discussed in the previous topic.

You can use the encryption key to encrypt all internal passwords and user-specified data. You can encrypt any value in a configuration by specifying `"encrypted": true,` when you POST that configuration setting to the API. There is also an option in the PowerFlow user interface to select *encrypted*. Encrypted values use the same randomly-generated encryption key.

User-created passwords and encryption keys are securely exposed in the Docker containers using Docker secrets at https://docs.docker.com/engine/swarm/secrets/ to ensure secure handling of information between containers.

NOTE: The encryption key must be identical between two PowerFlow systems if you plan to migrate from one to another. The encryption key must be identical between High Availability or Disaster Recovery systems as well.

TIP: PowerFlow supports all special characters in passwords.

NOTE: For detailed security information about the configuration of Docker Enterprise, see the ***SL1 PowerFlow: System Security Plan for Docker Enterprise*** document.

Configuring Additional Elements of PowerFlow

If you have multiple workers running on the same PowerFlow system, you might want to limit the amount of memory allocated for each worker. This helps prevent memory leaks, and also prevents one worker using too many resources and starving other workers. You can apply these limits in two ways:

- Set a hard memory limit in Docker (this is the default)
- Set a soft memory limit in the worker environment

Setting a Hard Memory Limit in Docker

Setting a memory limit for the worker containers in your docker-compose.yml file sets a hard limit. If you set a memory limit for the workers in the docker-compose file and a worker exceeds the limit, the container is terminated via SIGKILL.

If the currently running task caused memory usage to go above the limit, that task might not be completed, and the worker container is terminated in favor of a new worker. This setting helps to prevent a worker from endlessly running and consuming all memory on the PowerFlow system.

You can configure the hard memory limit in the steprunner service of the **docker-compose.yml** file:

```
deploy:
  resources:
    limits:
      memory: 2G
```

Setting a Soft Memory Limit in the Worker Environment

You can set the memory limit for a worker application, and not at the Docker level. Setting the memory limit at the application level differs from the hard memory limit in Docker in that if a worker exceeds the specified memory limit, that worker is not immediately terminated via SIGKILL.

Instead, if a worker exceeds the soft memory limit, the worker waits until the currently running task is completed to recycle itself and start a new process. As a result, tasks will complete if a worker crosses the memory limit, but if a task is running infinitely with a memory leak, that task might consume all memory on the host.

NOTE: The soft memory limit is less safe from memory leaks than the hard memory limit.

You can configure the soft memory limit with the worker environment variables. The value is in KiB (1024 bytes). Also, each worker instance contains three processes for running tasks. The memory limit applies to each individual instance, and not the container as a whole. For example, a 2 GB memory limit for the container would translate to 2 GB divided by three, or about 700 MB for each worker:

```
steprunner:

  image: repository.auto.sciencelogic.local:5000/is-worker:2.6.0

  environment:

    additional_worker_args: ' --max-memory-per-child 700000'
```

PowerFlow Task Processing and Memory Handling

Review the settings in this section to prevent an "Out of Memory" error, also called an "Oomkill" error or exit code 137. These errors occur when a container uses more memory than the container has been allotted.

This section will help you to recognize and diagnose these situations, and determine what additional configurations are available when working with a PowerFlow system that is running out of memory.

Background

By default steprunner containers have a 2 GB memory limit with three process threads by default. Limits for containers are set in the **docker-compose** file.

Use the `docker stats` command to see what the current memory usage of containers are in PowerFlow, along with current memory usage for those containers.

CPU and Memory Requirements for PowerFlow

The following table lists the CPU and memory requirements based on the number of synced objects for a PowerFlow system:

Minimum Number of Synced Objects	CPU Cores	Memory RAM (GB)	Hard Disk (GB)
30,000	8	24	100
65,000	8	32	100
100,000	8	64	200

Typical PowerFlow Deployments:

- Standard Single-node Deployment (1 Node): One node, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes.
- Standard Three-node Cluster (3 Nodes): Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes.
- 3+ Node Cluster with Separate Workers (4 or More Nodes): Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes.

Recommended Memory Allocation of PowerFlow Nodes

The following sizings will automatically be applied if you run **powerfcontrol** (pfctl) actions such as **apply 16GB overrides** and **apply 32GB overrides**. These commands should only be run in a Software-as-a-Service (SaaS) environment. For more information about the pfctl actions, see *apply_<n>GB_override, verify_<n>GB_override.*

Template Size	Device Load
16 GB	25,000 to 30,000
32 GB	30,000 to 70,000
64 GB	70,0000 to 350,000
128 GB	350,000 and above

SaaS Deployments

The following settings are specifically for Software-as-a-Service (SaaS) environments, and they ensure full replication of all services in any failover scenario.

Example Code: docker-compose for SaaS

```
services:
  dexserver:
    deploy:
      placement:
        max_replicas_per_node: 1
      replicas: 3
      restart_policy:
        condition: any

  couchbase-worker:
    environment:
      AUTO_REBALANCE: 'true'

  couchbase-worker2:
    environment:
      AUTO_REBALANCE: 'true'
```

```
  scheduler:
    deploy:
      resources:
        limits:
          memory: 200M
      restart_policy:
        condition: any

  steprunner:
    healthcheck:
      interval: 1m
      retries: 5
      start_period: 2m
      test:
        - CMD-SHELL
        - celery -A ipaascommon.celeryapp:app inspect ping -d cel-
ery@$${HOSTNAME}
      timeout: 20s
```

16 GB Deployments

The following settings support up to approximately 25,000-30,000 devices, depending on relationship depth.

Example Code: docker-compose for 16 GB Deployments

```
services:
  redis:
    deploy:
      resources:
        limits:
          memory: 5G
    environment:
      MAXMEMORY: 3000mb

  steprunner:
    deploy:
      replicas: 3
      placement:
        max_replicas_per_node: 1
      resources:
```

```
      limits:
        memory: 3G

  contentapi:
    deploy:
      resources:
        limits:
          memory: 1G
    environment:
      uwsgi-workers: 5
```

Allocations (per node):

- Swarm leader: between 2 to 4 GB left over on node
- Couchbase: reserves 1.5 GB memory, uses 1.5 to 4 GB, depending on operation
- Flower, pypiserver, dexserver, scheduler: no limit by default, never use more than 100 MB
- RabbitMQ: no limit, typically low usage (less than 100 MB), might spike with heavy loads
- Contentapi: 1 GB memory limit
- Redis: 3 GB soft limit, 5 GB hard limit; after 5 GB, Redis will automatically eject older data for new (reduced)
- 1x steprunners: 3 GB memory limit each (steprunner count decreased, memory limit increased)

Total limits/max expected memory usage: 4 GB + 4 GB + 500 MB + 1 GB + 3 GB +3 GB = 15.5GB/16GB

32 GB Deployments

The following settings support up to approximately 70,000 devices, depending on relationship depth.

Example Code: docker-compose for 32 GB Deployments

```
# 32GB specific pfctl action:
  steprunner:
    environment:
      worker_threads: 2
    deploy:
      placement:
        max_replicas_per_node: 2
      replicas: 6
      resources:
        limits:
          memory: 7G

  contentapi:
```

```
  deploy:
    resources:
      limits:
        memory: 2G

redis:
  deploy:
    resources:
      limits:
        memory: 5G
  environment:
    MAXMEMORY: 3000mb
```

Allocations (per node):

- Swarm leader: between 2-4 GB left over on node
- Couchbase: reserves 1.5 GB memory, uses 1.5 to 4GB, depending on operation
- Flower, pypiserver, dexserver, scheduler: no limit by default, never uses more than 100 MB
- RabbitMQ: should anticipate for 1 GB at larger sizes
- Contentapi: 2 GB memory limit (in a healthy running environment, should be less than 100 MB)
- Redis: 3 GB soft limit, 5 GB hard limit; after 3 GB, Redis will automatically eject older data for new (reduced)
- 2x steprunners: 7 GB memory limit each (steprunner count decreased, memory limit increased)

Total limits/max expected memory usage: 4 GB + 4 GB + 200 MB + 1 GB + 2 GB + 4 GB +7(2) GB = 29.2/32GB

64 GB Deployments

The following settings support over 70,000 devices, depending on relationship depth.

Example Code: docker-compose for 64 GB Deployments

```
steprunner:
  deploy:
    placement:
      max_replicas_per_node: 4
    replicas: 12
    resources:
      limits:
        memory: 2G
```

```
  steprunner_xl:
    hostname: xlworker-{{.Task.ID}}
    healthcheck:
      interval: 1m
      retries: 5
      start_period: 2m
      test:
        - CMD-SHELL
        - celery -A ipaascommon.celeryapp:app inspect ping -d cel-
ery@$${HOSTNAME}
      timeout: 20s
    deploy:
      placement:
        max_replicas_per_node: 1
      replicas: 3
      resources:
        limits:
          memory: 15G
      restart_policy:
        condition: any
        delay: 10s
    environment:
      worker_threads: 2
      user_queues: xlsync
      additional_worker_args: ' --max-tasks-per-child 1 '
      broker_url: pyamqp://guest@rabbit//
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet
      logdir: /var/log/iservices
      result_backend: redis://redis:6379/0
    image: sciencelogic/pf-worker:rhel2.4.1
    networks:
      isnet: {}
    secrets:
    - source: encryption_key
    - source: is_pass
    volumes:
```

```
    - /var/log/iservices:/var/log/contentapi:rw
    - /var/log/iservices:/var/log/iservices:rw
    - syncpacks_virtualenvs:/var/syncpacks_virtualenvs:ro

  redis:
    deploy:
      resources:
        limits:
          memory: 5G
    environment:
      MAXMEMORY: 3000mb
```

NOTE: If you use the following format for the names of the custom steprunners, they will display on the PowerFlow **Control Tower** page: `steprunner_<name>`.

Other actions needed:

- Increase Couchbase Allocations: increase data bucket allocation by 5 GB, and increase index allocation by 5 GB
- Update the current Device Sync or Interface Sync applications, and specify them to run on the **xlsync** queue

Allocations (per node):

- Swarm leader: between 2 to 4 GB left over on node
- Couchbase: reserves 1.5 GB memory, uses 1.5 to 4 GB standard; add 10 GB for heavy scale readiness (5 GB to data bucket 5 to index), up to 14 GB
- Flower, pypiserver, dexserver, scheduler: no limit by default; never uses more than 100 MB
- RabbitMQ: anticipate for 4 GB at extremely larger sizes
- Contentapi. 2 GB memory limit (in a healthy running environment, should be less than 100 MB)
- Redis : 3 GB soft limit, 5 GB hard limit
- 4x steprunners: 2 GB memory limit each (steprunner count decreased, memory limit increased), 8 GB total
- 1x steprunner: 15 GB memory limit (xlqueue steprunner), 15 GB total

Total limits/max expected memory usage: 4GB + 14GB +100mb + 4gb + 2GB + 5GB + 8GB + 15GB = 52GB/64

IMPORTANT: There is still approximately 12 GB to be allocated to needed services. This configuration and allotment may change depending on future assessment of customer systems.

128 GB Deployments

This deployment template is only to be used for customers with a very large number of devices, such as over 350,000 devices. For a deployment this large, you will need append additional customizations and queues to the following template. This is just a baseline; discuss with ScienceLogic if you plan to use a 128 GB deployment.

Example Code: docker-compose for 128 GB Deployments

```
    steprunner:
    deploy:
      placement:
        max_replicas_per_node: 24
      replicas: 72
      resources:
        limits:
          memory: 2G


  steprunner_xl:
    hostname: xlworker-{{.Task.ID}}
    healthcheck:
      interval: 1m
      retries: 5
      start_period: 2m
      test:
        - CMD-SHELL
        - celery -A ipaascommon.celeryapp:app inspect ping -d cel-
ery@$${HOSTNAME}
      timeout: 20s
    deploy:
      placement:
        max_replicas_per_node: 1
      replicas: 3
      resources:
        limits:
          memory: 15G
      restart_policy:
        condition: any
        delay: 10s
    environment:
      worker_threads: 2
```

```
      user_queues: xlsync
      additional_worker_args: ' --max-tasks-per-child 1 '
      broker_url: pyamqp://guest@rabbit//
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet
      logdir: /var/log/iservices
      result_backend: redis://redis:6379/0
    image: sciencelogic/pf-worker:rhel2.4.1
    networks:
      isnet: {}
    secrets:
    - source: encryption_key
    - source: is_pass
    volumes:
    - /var/log/iservices:/var/log/contentapi:rw
    - /var/log/iservices:/var/log/iservices:rw
    - syncpacks_virtualenvs:/var/syncpacks_virtualenvs:ro

  redis:
    deploy:
      resources:
        limits:
          memory: 12G
    environment:
      MAXMEMORY: 6000mb
```

Differences from 64 GB Deployments

- The default queue worker count was tripled. These additional workers may be dedicated to any other queue as needed by your customizations.
- Increased redis limits to allow for more processing.

Allocations (per node)

- Swarm leader: Between 2-4 GB left over on node.
- Couchbase: Reserves 1.5 GB memory, uses 1.5 – 4 GB standard, +10 GB for heavy scale readiness (5 GB to data bucket, 5 GB to index): up to 14 GB
- Flower, pypiserver, dexserver, scheduler: No limit by default. Never uses more than 100 MB.
- Rabbitmq: Should anticipate 6 GB at extremely larger sizes.
- Contentapi: 2 GB memory limit. Iin a healthy running environment, should be less than 100 MB.
- Redis: 6 GB soft limit, 12 GB hard limit.
- 24x steprunners: 2 GB memory limit each (steprunner count decreased, memory limit increased): 48 GB
- 1x steprunner: 15 GB memory limit (xlqueue steprunner): 15 GB

Total limits/max expected memory usage:

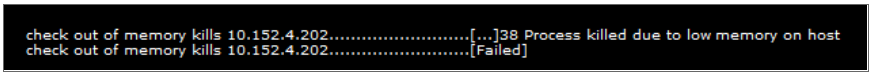
4 GB + 14 GB +100 MB + 6 GB + 2 GB + 12 GB + 48 GB + 15GB = 101 GB/128

Identifying Oomkills

Typically Oomkills occur only on PowerFlow systems with over 8,000 devices with many relationships or 30,000 interfaces that are being synced.

To identify Oomkills:

1. Use the **healthcheck** action with the powerflowcontrol (pfctl) command-line utitlity to identify the occurrence. Sample feedback that shows an Oomkill situation:



2. Log in to the node where the container failed.
3. From the node where the container failed, run the following command:

```
journalctl -k | grep -i -e memory -e oom
```

4. Check the result for any out of memory events that caused the container to stop. Such an event typically looks like the following:

```
is-scale-03 kernel: Out of memory: Kill process 5946 (redis-server)
score 575 or sacrifice child
```

Common Causes of High Memory and Oomkills

- Large-scale Device Sync, Attribute Sync, and Interface Syncs can cause out of memory events. The following situations might need more than the default limit allocation:
  - Device Sync for about 9,000 to 12,000 devices, with large amounts of relationships
  - A large-scale Attribute Sync with large numbers of devices
  - An large-scale Interface Sync with about 10,000 or more interfaces
- Python can be very resource-intensive, especially when it comes to loading or dumping large JSON files. JSON dumps and JSON loads can be inefficient and use more memory than expected. To avoid Oomkill in these situations, instead of using JSON for serialization, you can "pickle" the **dict()** object from Python and then dump or load that.
- A cursor size issue can occur when GraphQL responses contain extremely large cursor sizes, increasing the amount of data returned by SL1 when making API requests. This issue was resolved in SL1 11.1.2 and later.

Questions to Ask when Experiencing Oomkills

- How many devices or interfaces are being synced?
- How often are the devices or interfaces being synced?
- What does the schedule look like? How many scheduled large integrations are running at the same time?

- What is the likelihood that those schedules are hitting double large syncs on one worker?
- If this is a custom SyncPack, should the workload be using this much memory? Can I optimize, or maybe paginate?

Avoiding Oomkills

The following table explains how to configure your PowerFlow system if you are encountering Oomkills:

Configuration	Steps	Requirements	Impact
Update scheduled applications	Review all scheduled application syncs and make sure you do not schedule two very large syncs to run at the same time of day.	None.	Separate timings of large-scale runs.
Increase the memory limit (SaaS only, not on-premises PowerFlow systems)	Increase the memory in the **docker-compose** file.	Host must have enough additional memory to allocate.	More rooms for tasks to run concurrently on one worker, increased memory allocation to host.
Reduce worker thread count	Set **worker_threads=1** for the steprunner environment variable in the **docker-compose** file.	None.	More room for large tasks to run, but fewer concurrent tasks (throughput).
Dedicated worker nodes, dedicated queues	Create dedicated queues for certain workloads to run on only designated workers.	Additional nodes are needed.	Provides dedicated resources for specific workflows. Generally used for very environment-demanding workloads.

IMPORTANT: After making any of the above configuration changes, be sure to run the **healthcheck** and **autoheal** actions with the**powerflowcontrol** (pfctl) command-line utility before you log out of PowerFlow and redeploy the PowerFlow stack. For more information, see *healthcheck and autoheal*.

Avoiding Node Exhaustion

Node exhaustion occurs when more memory is allocated to containers than is available on the host. If memory is exhausted on the Swarm leader node and the cluster operations cannot process, all containers will restart. You will see "context deadline exceeded" in docker logs if you run `journalctl --no-page |grep docker |grep err`.

The following table explains how to configure you PowerFlow system to prevent node exhaustion from occurring again:

Configuration	Steps	Requirements	Impact
Reduce steprunner replica count	Reduce the replica count of the steprunner in the **docker-compose** file.	None.	Fewer concurrent processes, less memory usage on host.

Reduce redis memory limits	Set the **MAXMEMORY** environment variable in the **docker-compose** file for redis (soft limit), reduce memory limit in **docker-compose** (hard limit)	None.	Less possible room for cached data in very large syncs, less ability for heavy concurrent runs at the same time, less ability to view result data in the user interface.
Dedicated worker nodes, dedicated queues	Create dedicated queues for certain workloads to run on only designated workers	Additional nodes are needed.	Provides dedicated resources for specific workflows. Generally used for very environment-demanding workloads.
Drained manager	Similar to dedicated worker nodes. This offloads swarm management work to another node.	Additional (smaller) nodes are needed.	Eliminates possibility of cluster logic failure due to memory exhaustion. Alternatively, just make sure the existing nodes have enough room.

IMPORTANT: After making any of the above configuration changes, be sure to run the **healthcheck** and **autoheal** actions with the**powerflowcontrol** (pfctl) command-line utility before you log out of PowerFlow and redeploy the PowerFlow stack. For more information, see *healthcheck and autoheal*.

Best Practices for Running PowerFlow with Production Workloads

If you are running PowerFlow in a Software-as-a-Service (SaaS) environment in the cloud, consider the following best practices to avoid failed PowerFlow Syncs and memory issues.

Avoid Debug Logging for Large-scale Runs

When you run a large-scale Device Sync or Interface Sync in Debug mode, PowerFlow logs all of the data that is requested, compared, and sen. Using Debug mode in this way moight cause the PowerFlow system to appear to be unresponsive for a period of time, or until the issue is identified and resolved by ScienceLogic Support.

If you need detailed logs for a large number of events, you should use the Info log level instead.

Additional Queues Might be Needed for Large-scale Runs

SaaS environments for PowerFlow are configured by default with a single queue. All Syncs and tasks run in a "first-in, first-out" (FIFO) manner. If an extremely large event spike occurs, or if a backlog of tasks are triggered, PowerFlow will backlog all tasks until the queue is processed. This default is more than sufficient for most PowerFlow environment, and it provides a consistent balance of throughput, scale, and replication for each of your Syncs.

If you want to separate workloads for large-scale environments, such as Device Sync and Incident Sync, you can allocate additional dedicated queues or nodes. To request additional dedicated queues, contact ScienceLogic Support.

Avoid Running Large-scale Syncs Simultaneously

ScienceLogic recommends that you do not simultaneously run multiple Device Syncs or Interface Syncs in large-scale environments (over 15,000 devices). Querying for all devices or interfaces in both ServiceNow and SL1 might have a large performance impact on the PowerFlow system and other systems involved.

If you want to ensure continually optimized performance, run only one large Sync at a time, and schedule the Syncs to run a different times.

For customers of a Managed Service Provider (MSP), ScienceLogic can provide a dedicated node for processing multiple Device Syncs. If you are interested in this deployment, contact ScienceLogic Support.

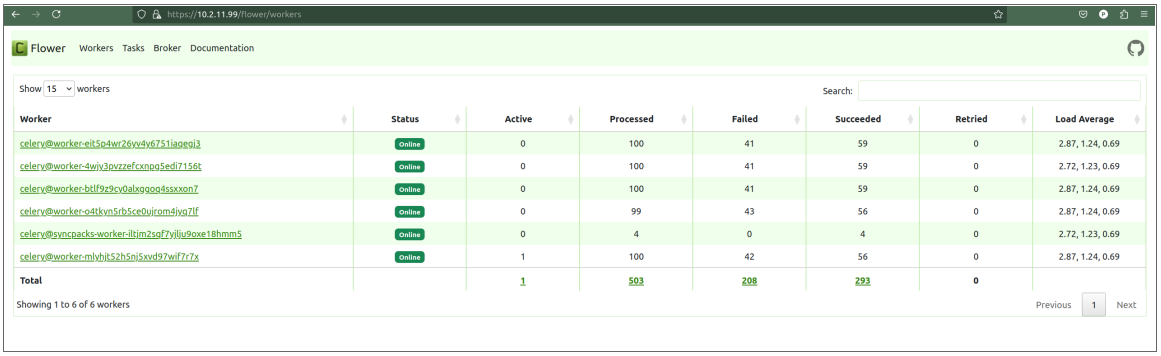
PowerFlow Management Endpoints

This section provides technical details about managing PowerFlow. The following information is also available in the PowerPacks in *Using SL1 to Monitor SL1 PowerFlow*.

Flower API

Celery Flower is a web-based tool for monitoring PowerFlow tasks and workers. You can access Flower at **https://*<IP of PowerFlow>*/flower/workers**.

Flower lets you see task progress, details, and worker status:



The following Flower API endpoints return data about the Flower tasks, queues, and workers. The **tasks** endpoint returns data about task status, runtime, exceptions, and application names. You can filter this endpoint to retrieve a subset of information, and you can combine filters to return a more specific data set.

/flower/api/tasks. Retrieve a list of all tasks.

/flower/api/tasks?app_id={app_id}. Retrieve a list of tasks filtered by app_id.

/flower/api/tasks?app_name={app_name}. Retrieve a list of tasks filtered by app_name.

/flower/api/tasks?started_start=1539808543&started_end=1539808544. Retrieve a list of all tasks received within a time range.

/flower/api/tasks?state=FAILURE|SUCCESS. Retrieve a list of tasks filtered by state.

/flower/api/workers. Retrieve a list of all queues and workers

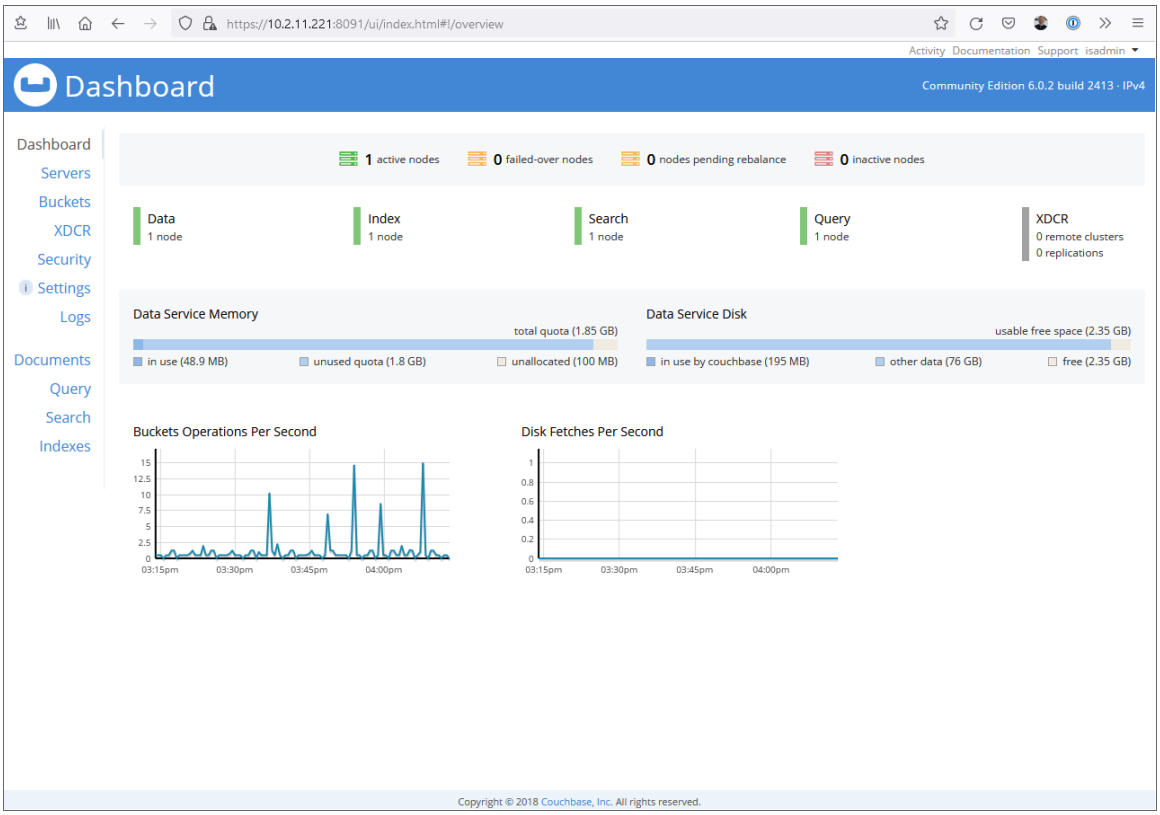
For more information, see the Flower API Reference at https://flower.readthedocs.io/en/latest/api.html.

NOTE: If you use the ScienceLogic: PowerFlow PowerPack to collect this task information, the PowerPack will create events in SL1 if a Flower task fails. For more information, see *Using SL1 to Monitor PowerFlow*.

Couchbase API

The Couchbase Server is an open-source database software that can be used for building scalable, interactive, and high-performance applications. Built using NoSQL technology, Couchbase Server can be used in either a standalone or cluster configuration.

The following image shows the CouchBase user interface, which you can access at port 8091, such as **https://*<IP of PowerFlow>*: 8091**:



The following Couchbase API endpoints return data about the Couchbase service. The **pools** endpoint represents the Couchbase cluster. In the case of PowerFlow, each ***node*** is a Docker service, and ***buckets***

represent the document-based data containers. These endpoints return configuration and statistical data about each of their corresponding Couchbase components.

***<hostname_of_PowerFlow_system>*:8091/pools/default**. Retrieve a list of pools and nodes.

***<hostname_of_PowerFlow_system>*:8091/pools/default/buckets**. Retrieve a list of buckets.

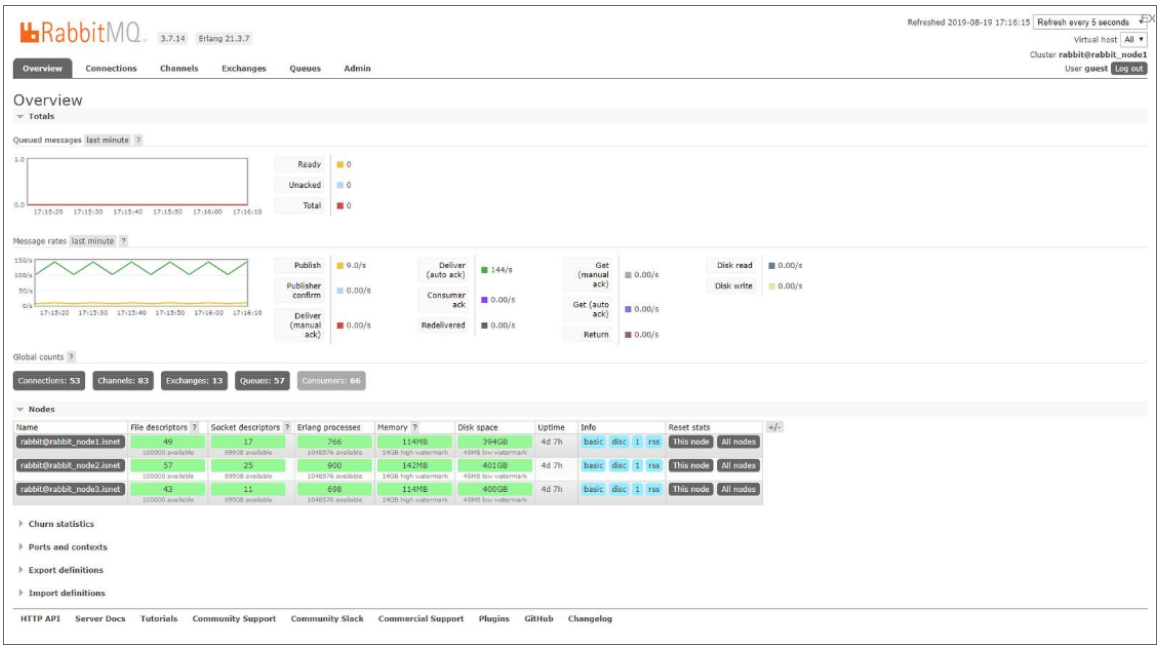
For more information, see the Couchbase API Reference.

NOTE: You can also use the Couchbase PowerPack to collect this information. For more information, see *Using SL1 to Monitor PowerFlow*.

RabbitMQ

RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol, Message Queuing Telemetry Transport, and other protocols.

The following image shows the RabbitMQ user interface, which you can access at port 15672, such as **https://*<IP of PowerFlow*: 15672**:



Docker Statistics

You can collect Docker information by using SSH to connect to the Docker socket. You cannot currently retrieve Docker information by using the API.

To collect Docker statistics:

1. Use SSH to connect to the PowerFlow instance.
2. Run the following command:

   ```
   curl --unix-socket /var/run/docker.sock http://docker<PATH>
   ```

   where `<PATH>` is one of the following values:

   - `/info`
   - `/containers/json`
   - `/images/json`
   - `/swarm`
   - `/nodes`
   - `/tasks`
   - `/services`

NOTE: You can also use the Docker PowerPack to collect this information. For more information, see *Using SL1 to Monitor PowerFlow*.

Chapter 3

Using the SL1 PowerFlow Control Tower Page

Overview

This chapter describes how to use the **PowerFlow Control Tower** page ( ) in the PowerFlow user interface to monitor the health of your PowerFlow system and PowerFlow applications.

This chapter covers the following topics:

What is the PowerFlow Control Tower? *107*

The System Health Widget *107*

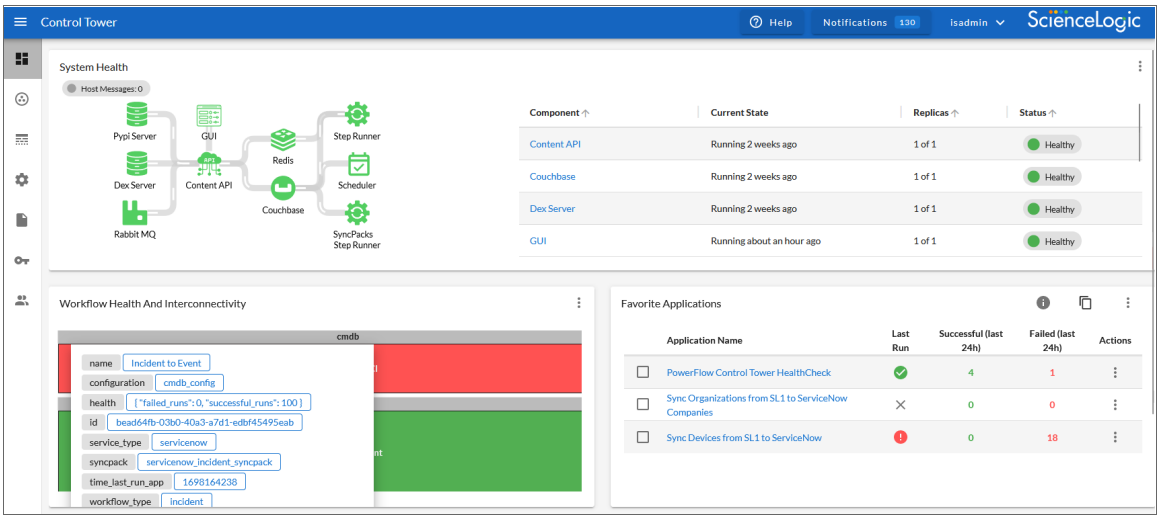
The Favorite Applications Widget *113*

The Workflow Health and Interconnectivity Widget *115*

The All Tasks, Workers, and Applications Widgets *118*

What is the PowerFlow Control Tower?

The **PowerFlow Control Tower** page ( ) in the PowerFlow user interface provides visibility into system health and automation health. This page is made up of a group of widgets that provide key information about your PowerFlow system.



The **PowerFlow Control Tower** page contains the **System Health**, the **Favorite Applications**, and the **Workflow Health and Interconnectivity** widgets alongside high-level statistics about the health of the worker services that are being used by the PowerFlow instance. For more information, see *PowerFlow Architecture*.

You can use the widgets on this page to monitor the health of your PowerFlow system, the various workflows you use regularly, and track the PowerFlow applications that you use the most. You can use this information to quickly determine if your PowerFlow instance is performing as expected.

Many of the widgets on the **Control Tower** page have a Configure icon ( ). If you click the Configure icon and select the *Configure* option from the menu, you can customize each widget, including the title and the size of the widget.

The System Health Widget

The **System Health** widget on the **PowerFlow Control Tower** page lets you see at a glance the health of the various elements of your PowerFlow system. Before you can view Health Status data on the dashboard, you need to configure PowerFlow.

Configuring the System Health Widget

To populate the system health widget, you need to install and activate the latest version of the "System Utils" SyncPack, which includes the "PowerFlow Control Tower HealthCheck" application that gathers system health data and populates it in the cache.

The latest version of the "System Utils" SyncPack requires the latest version of the "Base Steps" and "Flow Control" SyncPacks. You can download these SyncPacks from the ScienceLogic Support site at https://support.sciencelogic.com/s/powerpacks.

To set up PowerFlow Health Status data:

1. Ensure that your PowerFlow system has the following SyncPacks activated and installed:
   - "Base Steps" SyncPack version 1.5.0 or later
   - "System Utils" SyncPack version 1.1.5 or later
   - "Flow Control" SyncPack version 1.0.1 or later. This SyncPack ships with the latest version of PowerFlow.
2. Create a configuration object for the "PowerFlow Control Tower HealthCheck" application. You can make a copy of the "PF Control Tower Configuration Example" to use as a template for this configuration object. For more information, see *Creating a Configuration Object*.
3. Align the new configuration object with the "PowerFlow Control Tower HealthCheck" application by clicking the **[Configure]** button from the detail page for the application and selecting this configuration object from the ***Configuration*** drop-down.

> **NOTE:** The "PowerFlow Control Tower Healthcheck" Application supports using SSH keys for collecting data from a PowerFlow node. You must select the **use_ssh_key** option on the **Configuration** pane for the HealthCheck application to use the **ssh_key** application variable that is defined in the aligned configuration object.

4. You can use this the new "steps" key in the ***Connection_widget*** field to filter the data that displays on the **Workflow Health and Interconnectivity** widget:

    - When this field is present, the application filters for runs that used the "test" configuration object.
    - When this field is blank, the application filters for apps that did not have a configuration object aligned.
    - When the field is left out, the application does not filter, and it fetches and processes all runs regardless of the configuration objects (which is how the application worked by default in previous versions).
    - For example, in the ***Connection_widget*** field, you can add the following JSON code to display the latest application run with the "test" configuration object, and the application will only show successful or failed runs with that configuration object:

```
{
  "name": "Integration Template",
  "steps": [
    {
      "app_name": "integration_template",
      "step_name": "Get REST Test",
      "configuration": "test",
      "syncpack": "base_steps_syncpack"
    }
  ]
}
```

5. Run or schedule the "PowerFlow Control Tower HealthCheck" application to update the PowerFlow Health Status data.

> **NOTE:** The **System Health** widget runs the "PowerFlow Control Tower HealthCheck" application automatically when you are on the **PowerFlow Control Tower** page, but only if the data saved on the most recent run of the application is older than five minutes. You can override this update by creating a schedule. For more information, see *Scheduling Applications*.

Configuring the "PowerFlow Control Tower HealthCheck" Application to Gather pfctl Data

The "PowerFlow Control Tower HealthCheck" application can trigger the "PowerFlow PFCTL HealthCheck" application, which uses **healthcheck** data gathered by the **powerflowcontrol** (pfctl) command-line utility. Both applications are available in the latest version of the "System Utils" SyncPack.

NOTE: ScienceLogic recommends that you make a copy of the "PF PFCTL Healthcheck Configuration Example" configuration object to use with this application. You can find this configuration object in the "System Utils" SyncPack version 1.1.4 or later, which is available from the **PowerPacks & SyncPacks** page of the ScienceLogic Support site at https://support.sciencelogic.com/s/.

To configure the "PowerFlow Control Tower HealthCheck" application to gather pfctl data:

1. Go to the **Configurations** page ( ), click the Actions button ( ), and select *Edit* for the "PF PFCTL Healthcheck Configuration Example" configuration object. The **Configuration** pane appears.
2. Click the **[Copy as]** button and provide values for the following fields in the updated **Configuration** pane:
   - ***Friendly Name***. Name of the configuration object that will display in the user interface.
   - ***Description***. A brief description of the configuration object.
   - ***Author***. User or organization that created the configuration object.
   - ***Version***. Version of the configuration object.
   - ***hosts_config***. Click the **[Toggle JSON Editor]** button to view the JSON code, form which you can edit the node, passphrase, pkey, and user values for the host or hosts. Click the **[Toggle JSON Editor]** button again to return to the original **Configuration** pane fields.
   - ***pf_manager_nodes***. For a clustered environment, specify the three manager nodes of the cluster, separated by commas.
   - ***pf_username***. The SSH username for the node or nodes.
   - ***ssh_key***. The SSH key you want to use in place of a password for the remote location. Use the newline character \n as a separator. If the SSH key needs a paraphrase to be decrypted, set the paraphrase by editing the ***pf_password*** variable, below.

     NOTE: You will need to edit the SSH Key values in the JSON Editor for this release to ensure the key is properly set. For example: `"{config.ssh_key}"`. This is a known issue that will be addressed in a future release.

     TIP: To get a one-line string of the SSH key, run the following command:
     `sed -E ':a;N;$!ba;s/\r{0,1}\n/\\n/g' ~/.ssh/id_rsa`

   - ***pf_password***. Specify the SSH password or paraphrase for the SSH Key.
   - ***node***. Specify the hostname or IP address of the node where the pfctl **healthcheck** action will run.
3. Click **[Save]**.
4. On the **Applications** page ( ), open the "PowerFlow Control Tower HealthCheck" application and click the **[Configure]** button. The **Configuration** pane appears:

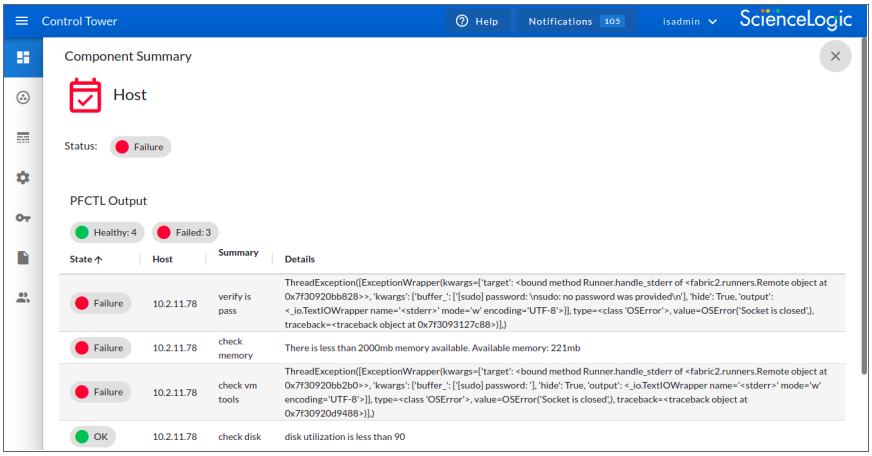
5. In the ***Configuration*** field, select the configuration object from step 2 to align it with this application.
6. To trigger and run the "PowerFlow PFCTL HealthCheck" application as a node-action (running the pfctl **healthcheck** action on just one node), complete the following fields on the **Configuration** pane:
   - ***use_ssh_key***. Select this option if you want to run the PF Control Tower application using an SSH key for authentication instead of using a password. You will need to provide a ***ssh_key*** value in the configuration object you aligned with this application, such as `"${config.ssh_key}"`.
   - ***action_type***: Select *node-action* from the drop-down on the **Configuration** pane.
7. To run the "PowerFlow PFCTL HealthCheck" application as a cluster-action (running the pfctl **healthcheck** action on a cluster), complete the following fields on the **Configuration** pane:
   - ***action_type***: Select *cluster-action*.
   - ***Hosts_config***: Define a list of hosts in this text box, following the example, below:

```
{
        "hosts": [
                {
                        "node": "10.2.11.241",
                        "password": "<password_in_plain_text_or_calling_a_conf
                        "user": "isadmin"
                },
                {
                        "node": "10.2.11.234",
                        "password": "${config.pf_password}",
                        "user": "${config.pf_username}"
                },
                {
                        "node": "10.2.11.242",
                        "password": "${config.pf_password}",
                        "user": "${config.pf_username}"
                }
        ]
}
```

8. Click **[Save]** to close the **Configuration** pane, and then run the "PowerFlow Control Tower HealthCheck" application. The application will trigger the "PowerFlow PFCTL HealthCheck" application to gather the pfctl **healthcheck** data from the node or cluster and display that data in the **System Health** widget.

> **NOTE:** If you have configured the "PowerFlow Control Tower HealthCheck" application to trigger the "PowerFlow PFCTL HealthCheck" application, you do not need to configure the "PowerFlow PFCTL HealthCheck"application.

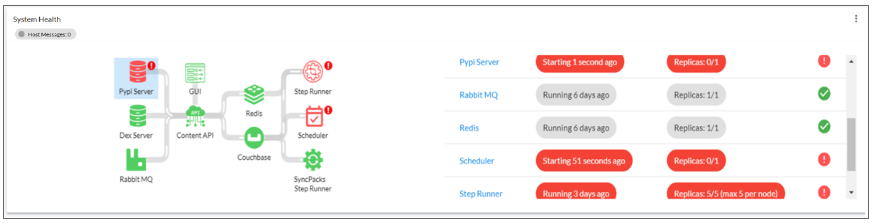
9. Click the **PFCTL Output** link or icon in the **System Health** widget to view the data gathered by the "PowerFlow PFCTL HealthCheck" application:



Using the System Health Widget

The **System Health** widget monitors all of the components that make up your PowerFlow system. These components include the Pypi Server, the Dex Server, RabbitMQ, the GUI service, the Content API, Redis, Couchbase, Step Runner and SyncPacks Step Runner, and the Scheduler. If the newest data is unavailable, the **System Health** widget displays the last available data.

The following image shows an example of a **System Health** widget:



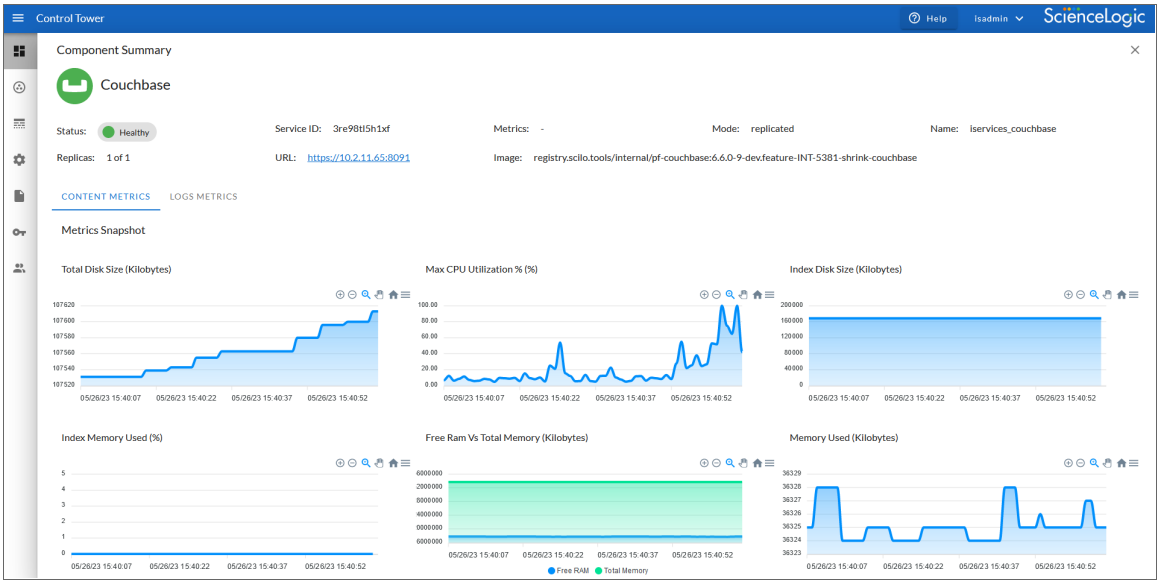
The left pane of the **System Health** widget contains the **Process Flow View** of the components of your PowerFlow system, and the right pane is the **Tabular View** of those components. The following are the possible health statuses for each component and how they are displayed:

- *Successful health*: The component is working as expected when the component's icon is green in the **Process Flow View** and a green icon appears in the corresponding line of the **Tabular View**.
- *Failed health*: The component has errors that need attention (the service is down) when the component's icon is red in the **Process Flow View**, with a red exclamation point ( ) next to the component's icon. The red exclamation point ( ) also appears in the corresponding line of the **Tabular View**, along with red ovals for the second and third columns of the table.

At the top left of the **System Health** widget is the **[Host Messages]** button, which you can click to view a pop-up that lists any issues that are currently occurring with the PowerFlow system.

The **Tabular View** has two behaviors:

- In the default display, the information in the **Process Flow View** on the left is duplicated in a tabular format in the **Tabular View** on the right.
- When you click a component's icon in the **Process Flow View** or a component's name in the **Tabular View**, the high-level information for a particular component appears in a pop-up window:

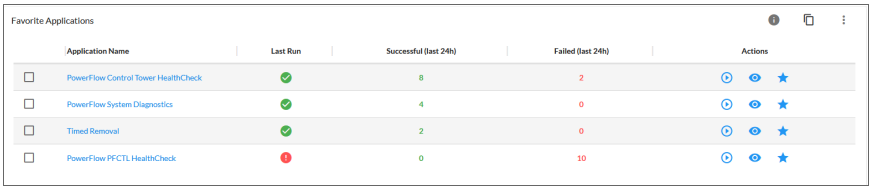


The pop-up displays the current status and related information for all containers in that component. The pop-up also includes a link to the internal user interface for that service within the cluster, such as a link to the Couchbase user interface for the Couchbase component, or the Flower user interface for a step runner.

When the Step Runner service displays a failure in the **System Health** widget, you can now click the Step Runner to display the following error message: "Step runners are not responding to ping, no health data could be collected.

The Favorite Applications Widget

The **Favorite Applications** widget on the **PowerFlow Control Tower** page lets you select the PowerFlow applications that are important to you and track their status:



By displaying the most frequently run applications, you can see how your PowerFlow system is automating your most common use cases.

NOTE: The number of favorite applications is limited to 16 applications per user.

Contents of the Favorite Applications Widget

The toolbar at the top right of the widget includes the following buttons:

- **[Info]** (). Displays a pop-up message with data for the Time Stamp, Number of Runs to Display, and the Queue for the widget.
- **[Duplicate the Widget]** (). Creates a copy of the **Favorite Applications** widget. The copy is added below the original widget. Making a copy lets you display more than one set of favorite applications, and you can create multiple widgets to group applications that serve a specific purpose.
- **[Actions]** (). Displays the following options:
    - *Configure*. Opens the **Configure Widget** pane, where you can update the Widget Title, Widget Size, Time Stamp for the application runs (24 hours or 48 hours), Total Number of Application Runs to Display, Queue information, and an editable list of Favorite Applications to display in the widget.
    - *Reorder Items*. Reorder the applications currently showing in the **Favorite Applications** widget. Use the up and down arrows to arrange the applications, and click **[Save]** when you are done.
    - *Delete*. Deletes that **Favorite Applications** widget.

The following details are included in this widget:

- **Application Name**. Names and links to favorite applications.
- **Last Run**. Status of the most recent run of a favorite applications; hover over the icon to see more information:

Icon	Status
	The application ran successfully.
	The application failed to run successfully.
×	The application has not been run.

- **Successful**. Number of successful runs in the last 24 hours.
- **Failed**. Number of failed runs in the last 24 hours.
- **Actions**. Includes the following icons:
    - *Run* (). Runs that PowerFlow application. If you hover over the button, you can select *Custom Run* to open the **Custom Run** window, where you can specify logging levels, the configuration object, and custom parameters for the run.
    - *View* (). Opens the **Application** detail page, where you can see the steps that make up the application.
    - *Unfavorite* (). Removes the application from the list of favorites.

TIP: If you are using a small screen, or if the browser window where you are running PowerFlow is not maximized, the three Actions icons might not display. To access the icons, click the Actions button (⋮) and select an icon from the pop-up menu.

Using the Favorite Applications Widget

To add an application to the **Favorite Applications** widget:

1. Go to the **Applications** page and click the Favorite icon (☆) for the application you want to add to the list. A **Favorite the App** window appears.
2. Select the group or groups of favorites that will include that application and click **[Save]**. The application is added to the list of favorite applications on the **Favorite Applications** widget.

NOTE: The data that displays in the widget can be adjusted by editing the **Configuration** pane, which you can access by clicking the **[Actions]** button (⋮) and selecting *Configure*.

3. To remove an application from the **Favorite Applications** widget, click the Unfavorite icon (★).

NOTE: If a favorite PowerFlow application is deleted, that application is removed from the **Favorite Applications** widget.

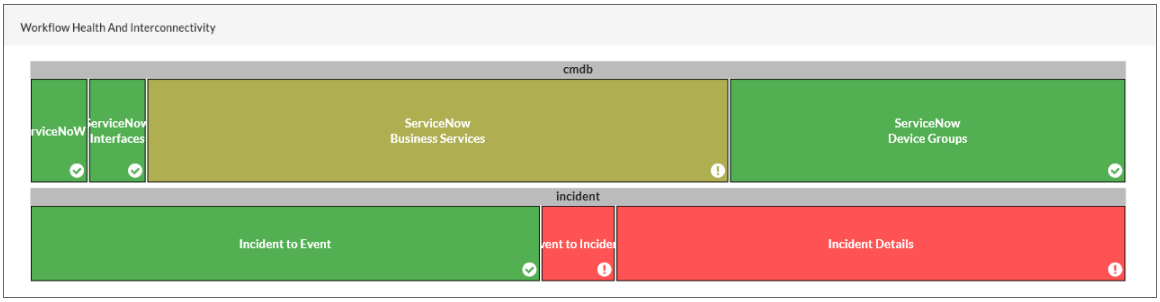
To run a favorite application in the **Favorite Applications** widget:

1. Click the **[Run]** button that corresponds to the application in the ***Actions*** column. If you hover over the button, you can select *Custom Run* to open the **Custom Run** window, where you can specify logging levels, the configuration object, and custom parameters for the run.
2. If the run succeeded, a green check mark will appear; if the run failed, a red exclamation point will appear.

TIP: You can select multiple applications to run them at the same time or remove them from your favorites.

The Workflow Health and Interconnectivity Widget

The **Workflow Health and Interconnectivity** widget on the **PowerFlow Control Tower** page lets you monitor the connectivity of the third-party applications that you are integrating with SL1. Each pane in the widget represents a workflow that you are monitoring with PowerFlow, such as ServiceNow Business Services or Incident Details.



The color of the panes in the widget change based on the number of failed runs compared to the number of successful runs. More failed runs cause a pane to turn red, while successful runs cause a pane to turn green. If there are a combination of failed and successful runs, the pane might be a lighter shade of green or red.

Configuring the Workflow Health and Interconnectivity Widget

The **Workflow Health and Interconnectivity** widget, you will need to configure the "PowerFlow Check Connections" application, which is available in the "System Utils" SyncPack version 1.1.4 or later. You can download this SyncPack from the ScienceLogic Support site at https://support.sciencelogic.com/s/powerpacks.

The "PowerFlow Check Connections" application gathers system connectivity health data from third-party applications, and that data is used by the **Workflow Health and Interconnectivity** widget.

To configure the "PowerFlow Check Connections" application used by the widget:

1. In the PowerFlow user interface, go to the **Applications** page ( ) and select the "PowerFlow Check Connections" application.
2. Click **[Configure]**. The **Configuration** pane for the application appears.
3. From the ***Configuration*** drop-down, select *PF Check Connection Configuration Example*.
4. Click the **[Edit]** button next to the ***Configuration*** drop-down. A new configuration pane appears to the left of the **Configuration** pane.
5. In the new pane, click **[Copy as]**. A **Create Configuration** pane appears.
6. Add the required descriptive information to the fields. For more information, see *Creating a Configuration Object*.
7. For the ***connection_widget*** field, you can click the **[Toggle JSON Editor]** button at the top right of the pane to view and edit the list of existing connections, along with the applications, steps, and SyncPacks aligned with those connections.

> **TIP:** You can update the JSON to look for any applications or steps you are running in PowerFlow. By default the "PowerFlow Check Connections" application only searches for ServiceNow CMDB, Incident, and Change Management applications.

8. Click **[Save]** on the **Create Configuration** pane.
9. On the **Configuration** pane, align the new configuration object you just created by selecting it from the ***Configuration*** drop-down.

10. In the ***number_of_days*** field, you can edit the number of days of data to query. The default is 2.

11. If needed, in the ***Connection_widget*** JSON text box, you can update the list of existing connections, along with the applications, steps, and SyncPacks aligned with those connections. This is the same content from step 7, above.

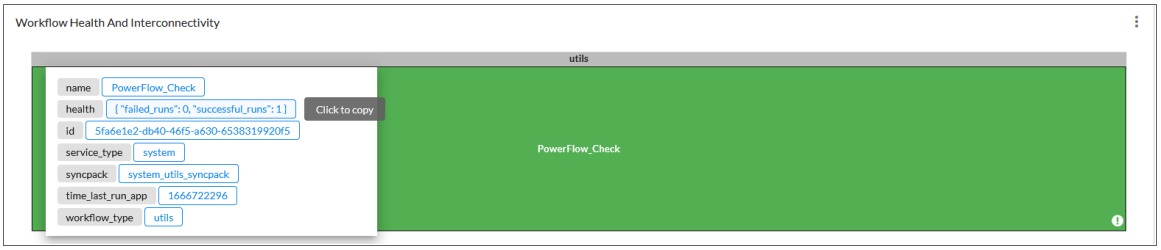
12. Click **[Save]**.

TIP: ScienceLogic recommends that you schedule the "PowerFlow Check Connections" application to run every 300 seconds (5 minutes), because the connections status and all other related documents will automatically delete themselves every seven days. For more information, see *Scheduling Applications*.

NOTE: When a step path is not configured correctly because a step, application, or SyncPack does not exist in the PowerFlow system, the "PowerFlow Check Connections" application will ignore that path and keep checking all the step paths that were configured.

Using the Workflow Health and Interconnectivity Widget

On the **Workflow Health and Interconnectivity** widget, you can hover over an endpoint on the widget to view a pop-up with additional information, including the health, last run and the SyncPacks used by the endpoint.



You can click any field in the pop-up to copy its value for troubleshooting purposes.

If you get an error message stating that the data generated is missing runs or health information, either you have not run or scheduled the "PowerFlow Check Connections" application, or you have not run any of applications that the "PowerFlow Check Connections" application is configured to monitor. By default the application only searches for ServiceNow CMDB, Incident, and Change Management applications.

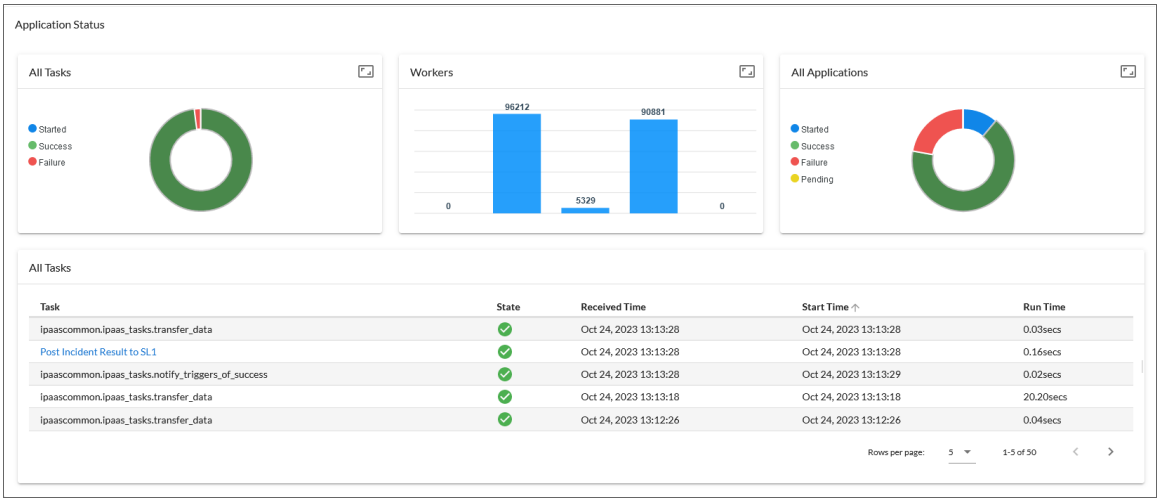
To update the list existing connections, edit the JSON in the ***connection_widget*** field on the **Configuration** pane of the "PowerFlow Check Connections" application. For more information, see *Configuring the Workflow Health and Interconnectivity Widget*.

NOTE: A message will display in the PowerFlow user interface if the **Workflow Health and Interconnectivity** widget or the **System Health** widget detect a missing or misconfigured SyncPack.

The All Tasks, Workers, and Applications Widgets

The **All Tasks**, **Workers**, and **Applications** widgets on the **PowerFlow Control Tower** page let you monitor the status of the various tasks, workers, and applications that are running on your PowerFlow system.

You can use this information to quickly determine if your PowerFlow instance is performing as expected:



To view more information about your system:

1. Hover over a circle graph or a bar chart item to view a pop-up field that contains the count for that item on the graph or chart, such as "Success: 48" for successful tasks on the **All Tasks** graph. Click an item on a circle graph to see more information in the lower pane under the charts.
2. Click the View all icon (⛶) for the **All Tasks**, **Workers**, or **Applications** graphs to view a list of relevant tasks, workers, or applications in the lower pane. Use the left and right arrow icons to move through the list of items. Click the slice of the pie or the bar in one of the graphs to see that specific sub-group.

TIP: If a "Scheduled fetch failed" pop-up message appears on this page or any other page in the PowerFlow user interface, your user interface session might have expired. To address this issue, simply log out of the PowerFlow user interface and log back in again.

Chapter 4

Managing SyncPacks

Overview

This chapter describes how to use the **SyncPacks** page ( ) of the PowerFlow user interface to import, install, and view SyncPacks.

> WARNING: PowerFlow and SyncPack content not created by ScienceLogic is *not* supported by ScienceLogic. This includes custom steps, PowerFlow applications, and SyncPacks.

This chapter covers the following topics:

What is a SyncPack? 120

Viewing the List of SyncPacks 121

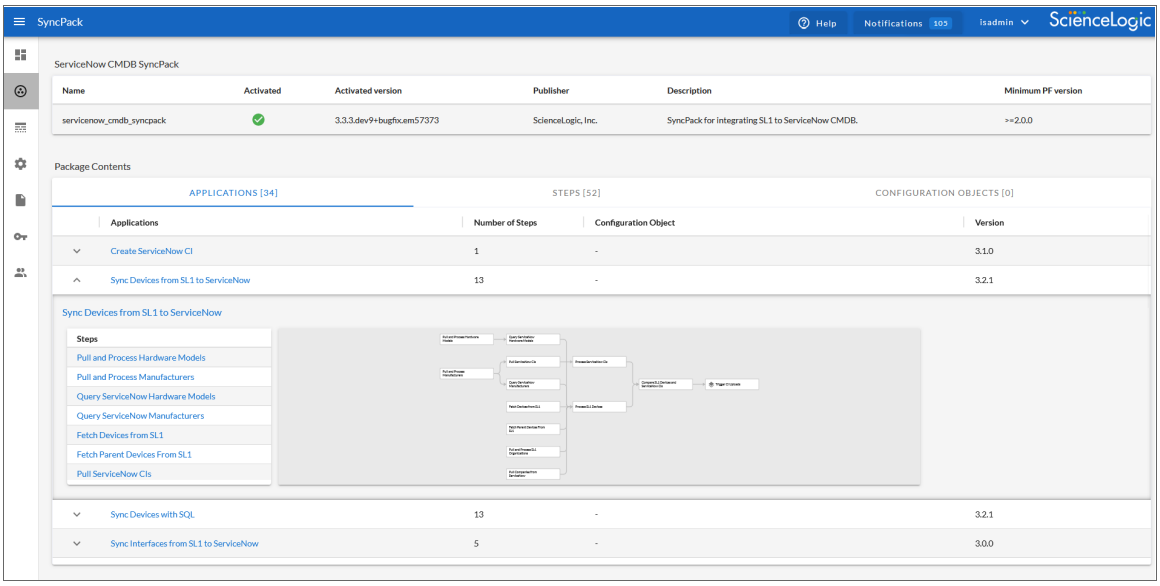
Importing and Installing a SyncPack 123

Default SyncPacks 128

Uploading Custom Dependencies to the PyPI Server with the iscli Tool 129

What is a SyncPack?

A SyncPack contains all of the code and logic needed to perform integrations on the PowerFlow platform. A SyncPack is saved as a Python **.whl** file, and it typically includes Python steps that are listed in PowerFlow applications and shipped with one or configuration objects:



You can view the latest steps, applications, and configurations for PowerFlow or a third-party integration, such as ServiceNow, by downloading the most recent SyncPack for that integration. You can download SyncPacks from the **PowerPacks & SyncPacks** page at the ScienceLogic Support Site at https://support.sciencelogic.com/s/powerpacks (login required).

You can access all SyncPacks that have been uploaded to your PowerFlow system on the **SyncPacks** page of the PowerFlow user interface.

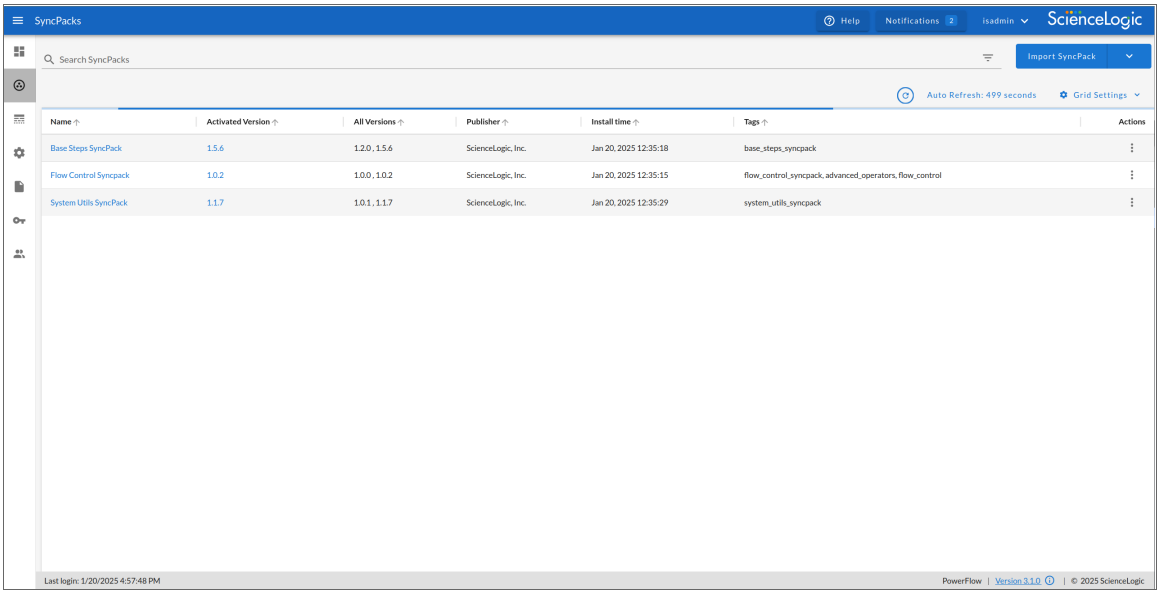
A SyncPack is highly customizable, and you can modify the contents of a SyncPack to meet your business needs, instead of changing your SL1 configuration. You can categorize and segment your business integration solutions using a SyncPack.

SyncPacks let you distribute content to simplify installations, upgrades, and maintenance of integration solutions. Just like PowerPacks, you can also share SyncPacks. In addition, SyncPacks protect a company's intellectual property using licensing and encryption technologies.

Viewing the List of SyncPacks

The **SyncPacks** page ( ) provides a list of the SyncPacks on your PowerFlow system. From this page you can search for and view SyncPacks, and you can also install and uninstall SyncPacks.

You can export a CSV list of the installed SyncPacks by clicking ***Grid Settings*** and then selecting *Export to CSV*.



Searching for a SyncPack

You can search for a specific SyncPack by typing the name of that SyncPack in the ***Search*** field at the top of the **SyncPacks** page. The user interface filters the list as you type. You can also filter the list by typing in the text box above a column header, and sort by clicking most column headers.

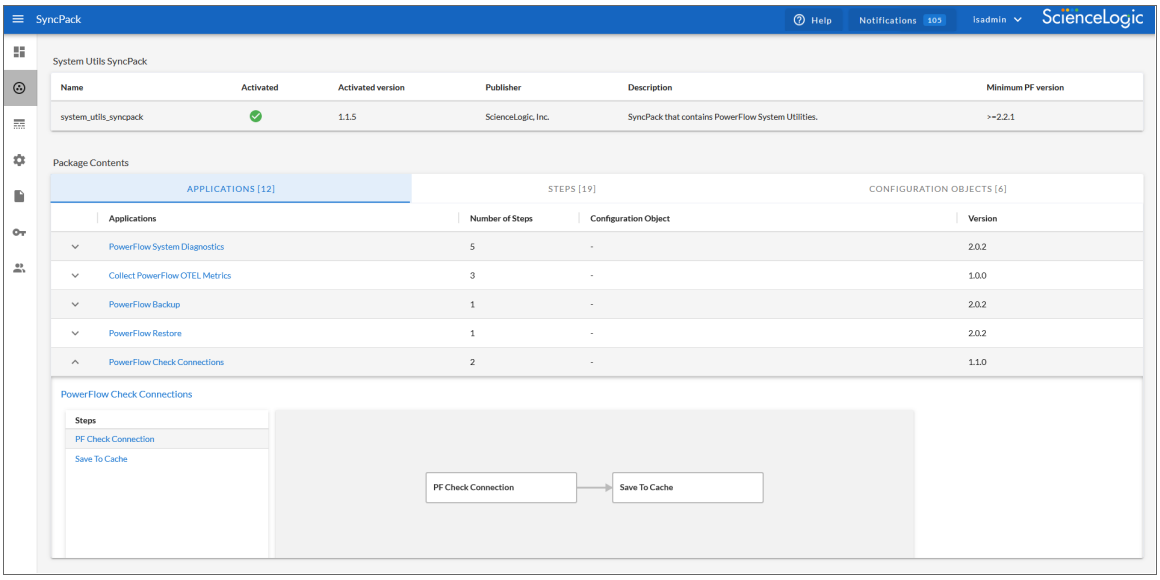
If you have more than one version of a SyncPack the active version of the SyncPack is highlighted in green in the **Versions** column.

TIP: Click the Filter icon ( ) at the top right of the **SyncPacks** page and select *Toggle Inactive SyncPacks* to show all SyncPacks on the PowerFlow system.

Viewing a Detail Page for a SyncPack

Click the name of a SyncPack from the list to view the detail page for the SyncPack.

At the top of this page you can view additional metadata, including the active version, the minimum PowerFlow platform version, and other details.:



In the **Package Contents** section, you can view more details about the steps, applications, and configuration objects include with that SyncPack:

- The **[Applications]** tab includes a link to the detail page for each application, along with a list of steps in the application (with the option of clicking the step name to view the code for that step), and a thumbnail image of the workflow for the application.
- The **[Steps]** tab includes a link to the **Step Code** dialog for each step, along with a list of any applications that are using that step. You can also click the name of the application from the list to go to the detail page for that application.
- The **[Configurations]** tab includes a link to the detail page for each configuration object, along with a list of any PowerFlow applications that are aligned with that configuration object (with the option of clicking the application name to view the detail page for that application).

Using the Actions Button to Manage SyncPacks

On the **SyncPacks** page of the PowerFlow user interface, you can access the following SyncPack options by clicking the **[Actions]** button (⋮) next to a SyncPack:

- *Activate & Install*. Before you can use a SyncPack, you will need to activate and install it. An *activated* SyncPack has been installed and is ready to be used. For more information, see ***Activating and Installing a SyncPack***.

> **NOTE:** If you try to activate and install a SyncPack that is already activated and installed, you can choose to "force" installation across all the nodes in the PowerFlow system.

- *Change active version*. If you have more than one SyncPack installed, select this option to specify a new active version (other than the version that is currently running).
- *Uninstall*. Select this option if you want to completely remove all versions of a SyncPack, instead of deleting one or more versions using the *Delete* option, below.

- *Delete*. Remove one or more versions of a SyncPack from PowerFlow. You cannot delete the active version of a SyncPack (the version that is currently running).

TIP: Click the Filter icon ( ) at the top right of the **SyncPacks** page and select *Toggle Inactive SyncPacks* to show all SyncPacks on the PowerFlow system.

Importing and Installing a SyncPack

The **SyncPacks** page in the PowerFlow user interface lets you import and install the latest version of a SyncPack. After you install a SyncPack, PowerFlow considers that SyncPack to be ***activated*** and ready to be used.

By default, the **SyncPacks** page displays all activated and installed SyncPacks. If you do not see the PowerPack that you want to install, click the Filter icon ( ) on the **SyncPacks** page and select *Toggle Inactive SyncPacks* to see a list of the imported PowerPacks.

After you install a SyncPack, that SyncPack is available on the **SyncPacks** page of the PowerFlow user interface. The PowerFlow applications from that SyncPack are added to the **Applications** page of the user interface. You can click the name of that SyncPack to view the detail page for that SyncPack.

If you want to upload and install multiple ServiceNow SyncPacks at the same time, you should upload *all* of the SyncPacks first, and then install them to address any dependencies between the SyncPacks.

Also, you can click the dropdown arrow next to the **[Import SyncPack]** button to import or view dependencies for PowerFlow SyncPacks. For more information, see *Locating and Importing Dependencies for a SyncPack*.

NOTE: If a SyncPack has a dependency on another specific SyncPack version, you will need to install that SyncPack version on the PowerFlow system before you can install the SyncPack with the dependency. For more information, see the **SL1 PowerFlow Dependency Matrix** page at https://docs.sciencelogic.com/latest/Content/Web_General_Information/Doc_Archive/powerflow_release_matrix.htm.

NOTE: You must have the *Develop* or *Administrator* role to install a SyncPack. For more information, see *Managing Users in PowerFlow*.

NOTE: If your PowerFlow system uses self-signed certificates, you will need to manually accept the certificate before you can upload SyncPacks. Go to **https://*<IP address of PowerFlow>*:3141/isadmin**, accept the certificate, and then exit out of the tab. When you log into PowerFlow again, you will be able to upload SyncPacks.

Locating and Downloading a SyncPack

A SyncPack file has the **.whl** file extension type. You can download the SyncPack file from the ScienceLogic Support site.

WARNING: If you are *upgrading* to this version of the SyncPack from a previous version, make a note of any settings you made on the **Configuration** pane of the various PowerFlow applications in this SyncPack, as these settings are *not* retained when you upgrade.

To locate and download the SyncPack:

1. Go to the ScienceLogic Support Site at https://support.sciencelogic.com/s/.
2. Click the **[Product Downloads]** tab and select *PowerPack*.
3. In the ***Search PowerPacks*** field, search for the SyncPack and select it from the search results. The **Release Version** page appears.
4. On the **[PowerPack Versions]** tab, click the name of the SyncPack version that you want to install. The **Release File Details** page appears.
5. Click the **[Download File]** button or click the name of the **.zip** file containing the **.whl** file for this SyncPack to start downloading the file.

NOTE: After you download a SyncPack, you can import it to your PowerFlow system using the PowerFlow user interface.

NOTE: If you are installing or upgrading to the latest version of this SyncPack in an offline deployment, see "Installing or Upgrading in an Offline Environment" in the SyncPack release notes to ensure you install any external dependencies.

Importing a SyncPack

To import a SyncPack in the PowerFlow user interface:

1. On the **SyncPacks** page (⚙) of the PowerFlow user interface, click **[Import SyncPack]**. The **Import SyncPack** page appears.
2. Click **[Browse]** and select the **.whl** file for the SyncPack you want to install. You can also drag and drop a **.whl** file to the **Import SyncPack** page.
3. Click **[Import]**. PowerFlow registers and uploads the SyncPack. The SyncPack is added to the **SyncPacks** page.
4. You will need to activate and install the SyncPack in PowerFlow. For more information, see the following topic.

NOTE: You cannot edit the content package in a SyncPack published by ScienceLogic. You must make a copy of a ScienceLogic SyncPack and save your changes to the new SyncPack to prevent overwriting any information in the original SyncPack when upgrading.

Activating and Installing a SyncPack

To activate and install a SyncPack in the PowerFlow user interface:

1. On the **SyncPacks** page of the PowerFlow user interface, click the **[Actions]** button (⋮) for the SyncPack you want to install and select *Activate & Install*. The **Activate & Install SyncPack** modal appears.

> **NOTE:** If you try to activate and install a SyncPack that is already activated and installed, you can choose to "force" installation across all the nodes in the PowerFlow system.

> **TIP:** If you do not see the SyncPack that you want to install, click the Filter icon (⧩) on the **SyncPacks** page and select *Toggle Inactive SyncPacks* to see a list of the imported PowerPacks.

2. Click **[Yes]** to confirm the activation and installation. When the SyncPack is activated, the **SyncPacks** page displays a green check mark icon (✔) for that SyncPack. If the activation or installation failed, then a red exclamation mark icon (❗) appears.
3. For more information about the activation and installation process, click the check mark icon (✔) or the exclamation mark icon (❗) in the **Activated** column for that SyncPack. For a successful installation, the "Activate & Install SyncPack" application appears, and you can view the Step Log for the steps. For a failed installation, the **Error Logs** window appears.
4. If you have other versions of the same SyncPack on your PowerFlow system, you can click the **[Actions]** button (⋮) for that SyncPack and select *Change active version* to activate a different version other than the version that is currently running.

Locating and Importing Dependencies for a SyncPack

The most common error that occurs when installing a SyncPack is that the SyncPack dependencies are not installed. If a SyncPack has a dependency on another specific SyncPack version, you will need to import and install that SyncPack version on the PowerFlow system before you can install the SyncPack with the dependency.

Review the "System Requirements" section of the release notes for that SyncPack to ensure that you have installed all of the required applications for that SyncPack.

To view a list of additional PowerFlow and SL1 products that are required by the various SyncPacks, see the *SL1 PowerFlow Dependency Matrix* page.

In addition, you can click the dropdown arrow next to the **[Import SyncPack]** button to import or view dependencies for PowerFlowSyncPacks.

To view a list of available dependency files:

1. Click the down arrow icon (⌄) next to the **[Import SyncPack]** button and select *View Dependencies*. A devpi dependencies page appears.
2. To view more information about a dependency, click the link in the **Info page** column.
3. To download a dependency file, click the link in the **Releases** column.

To import a dependency:

1. Click the down arrow icon ( ) next to the **[Import SyncPack]** button and select *Import Dependency*.
2. Click **[Browse]** and select the relevant dependency file from the previous step.

> **TIP:** You can also drag and drop a file onto the **Import SyncPacks** page.

3. Click **[Import]**. The dependency file is added to the PowerFlow system .

Considerations for Custom Syncpacks with PowerFlow 3.1.0 and Later

As of PowerFlow 3.1.0, PowerFlowservices start using a new version of Couchbase SDK, which might break some Custom Syncpacks that currently use old versions of the Couchbase SDK directly.

Changes were added to keep as much backward compatibility as possible. If you do not follow one of the methods described below, the custom SyncPacks must be updated to use PowerFlow ContentManager, DBConnector, or the new Couchbase SDK methods described in the official Couchbase SDK documentation.

Updating Custom Syncpacks to Work with the New Couchbase SDK

Custom Syncpacks should be updated to stop using Couchbase SDK methods directly, and instead use ContentManager or DBConnector methods, because old Couchbase SDK methods (PowerFlow 3.0.0 or earlier) will be deprecated in a future version of PowerFlow.

Examples are included below, however, contact ScienceLogic Support for more information if needed.

Couchbase Locking Method

Instead of using

```
res = content_manager.db_connector.content_bucket.lock
```

Use

```
value, cas = self.cmanager.get_value_from_content_bucket( self.doc_name,
lock=true, locking_timeout=self.lock_timeout, return_cas=True )
```

Instead of

`TemporaryFailError` from **couchbase.exceptions**

Import

`DocumentLockedException` for locking methods.

Couchbase N1QL queries

Instead of using

```
cmanager.db_connector.logs_bucket.n1ql_query
```

```
cmanager.db_connector.content_bucket.n1ql_query,
```

Use

`cmanager.db_connector.execute_query_logs_bucket`

`cmanager.db_connector.execute_query_content_bucket`

Instead of using a n1ql query to delete logs or cache docs that matches a pattern, use `self.cmanager.delete_from_logs_startswith("prefix-")` or, to delete exact keys use `self.cmanager.db_connector.batch_remove_keys_from_logs([key1, key2, ])`

Couchbase Queries Metrics (Use only to get metrics)

Instead of getting the query metrics using

```
req = self.cmanager.db_connector.logs_bucket.n1ql_query(

N1QLQuery(delete_query)

)

res = req.metrics
```

Use

```
req = cmanager.db_connector.execute_query_logs_bucket(delete_query,
metrics=True)

req.execute()

metrics_response = req.metadata().metrics()._raw
```

Couchbase Exception

Instead of using

```
from ipaascommon.ipaas_exceptions import NotFoundError,
TemporaryFailError, TimeoutError
```

Use

```
from couchbase.exceptions import NotFoundException,
TemporaryFailException, TimeoutException

from couchbase.exceptions import DocumentLockedException for locking
methods
```

Default SyncPacks

When you install PowerFlow, the following SyncPacks are added to the **SyncPacks** page by default:

- *Base Steps*
- *Flow Control*
- *System Utils*

If you need to install these SyncPacks, click the **[Actions]** button (⋮) and select *Activate & Install* for each SyncPack.

Base Steps SyncPack

The "Base Steps" SyncPack contains a default set of steps that are used in most other SyncPacks. You must install and activate this SyncPack before running other SyncPacks.

This SyncPack also includes the **Template App** application, which you can use as a template for creating PowerFlow applications if you are a developer. For more information, see *Creating an SL1 PowerFlow Application*.

This SyncPack is included with the most recent release of the PowerFlow Platform.

Starting with version 1.5.0 of the "Base Steps" SyncPack, the "QueryREST" has been deprecated. ScienceLogic recommends that you use REST steps included in version 1.5.0 instead: "GetREST", "PostREST", "DeleteREST", and "PutREST".

TIP: To view the latest releases of this SyncPack, see *SL1 PowerFlow SyncPack Release Notes*.

NOTE: You can download this SyncPack from the PowerPacks & SyncPacks page (Product Downloads > PowerPacks & SyncPacks) at the ScienceLogic Support Site.

Flow Control SyncPack

The "Flow Control" SyncPack contains just one item: the "IfStep" step that enables the logical branching used by the PowerFlow builder. The most recent version of this SyncPack is version 1.0.1, and it is included in the PowerFlow Platform release.

System Utils SyncPack

The "System Utils" SyncPack is a Standard SyncPack that contains applications, a configuration object, and steps. Version 1.1.5 of this SyncPack is included with PowerFlow version 2.6.0.

You must install the "Base Steps" SyncPack before you can install this SyncPack.

TIP: To view the latest releases of this SyncPack, see *SL1 PowerFlow SyncPack Release Notes*.

NOTE: You can download this SyncPack from the PowerPacks & SyncPacks page (Product Downloads > PowerPacks & SyncPacks) at the ScienceLogic Support Site.

Uploading Custom Dependencies to the PyPI Server with the iscli Tool

You can use the PowerFlow command-line tool (iscli) to upload custom dependencies to the PowerFlow local Python Package Index (PyPI) Server:

1. Copy the Python package to the pypiserver container.
2. Exec into the container and run the following commands:

```
devpi login isadmin
```

```
devpi use http://127.0.0.1:3141/isadmin/dependencies
```

```
devpi upload <location of your package dependencies>
```

Chapter 5

Managing SL1 PowerFlow Applications

Overview

This chapter describes how to use the **Applications** page ( ) of the PowerFlow user interface to view, run, and schedule PowerFlow applications. You can use the PowerFlow builder to create custom applications, and those applications can use "flow control" operators that enable logical branching and data transformation between steps.

WARNING: PowerFlow and SyncPacks content not created by ScienceLogic is *not* supported by ScienceLogic. This includes custom steps, PowerFlow applications, and SyncPacks.

For more information about building low-code integrations with PowerFlow Builder, watch the video at https://sciencelogic.com/product/resources/building-no-codelow-code-integrations-with-powerflow-builder.

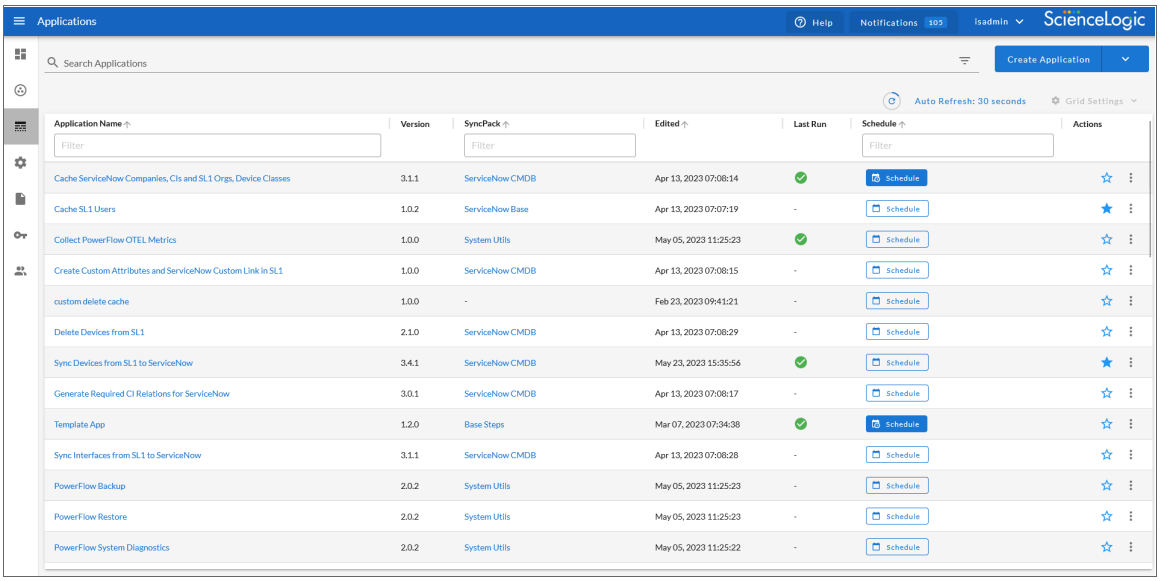
This chapter covers the following topics:

Viewing the List of PowerFlow Applications 131
Elements of an Application Page 133
Creating a Basic PowerFlow Application 136
Working with Flow Control Operators 139
Editing a PowerFlow Application 155
Enabling Run Book Automation Queue Retries 157
Creating a Step 161
Defining Retry Options for a Step 161
Aligning a Configuration Object with an Application 162
Running a PowerFlow Application 164

Viewing Previous Runs of an Application with the Timeline *165*
Scheduling a PowerFlow Application *168*
Backing up and Restoring PowerFlow Data *170*

Viewing the List of PowerFlow Applications

The **Applications** page ( ) provides a list of available PowerFlow applications on your PowerFlow system. From this page you can schedule, edit, view, and create applications and steps.



You can search for a specific application by typing the name of that application in the ***Search*** field at the top of the **Applications** page. The user interface filters the list as you type. You can also filter the list by typing in the text box above a column header, and sort by clicking most column headers.

The **Applications** page displays the following information:

- ***Application Name***. Lists the name of the application.
- ***Version***. Lists the version of the application.
- ***SyncPack***. Lists the name of the SyncPack (or "SyncPack") to which a specific application belongs, where relevant. You can click the name of the SyncPack to which an application belongs to go to the SyncPack page for that pack.
- ***Edited***. List the date and time for when the application was last edited by a user.

- ***Last Run***. Shows the current status of all of the PowerFlow applications:

Icon	Status
	The application ran successfully.
	The application is currently running. An application can have one of the following statuses: • ***Started***. The application is currently running. • ***Pending***. The application has not run or will not run. If you stop the application, the next tasks will not run.
	The application failed to run successfully.
-	The application has not been run.

- ***Actions***. Contains the following icons, or, if the browser window is not fully maximized, displays the **[Actions]** button (⋮) with the following options:

  - The **[Schedule]** button lets you use the Scheduler to define how often or at what time to run an application. A scheduled application displays a check mark in the **[Schedule]** button on this page. For more information, see *Scheduling a PowerFlow Application*.
  - The **[Favorite]** button (☆) lets you select the applications that you want to see in the **Favorite Applications** widget on the **PowerFlow Control Tower** page.
  - The **[Actions]** button (⋮) for an application gives you the option to run, view, or delete that application. You cannot run an application in Debug Mode or run an application with custom attributes from this menu. Click *View* to open the **Application** detail page if you want to use Debug Mode and custom attributes.

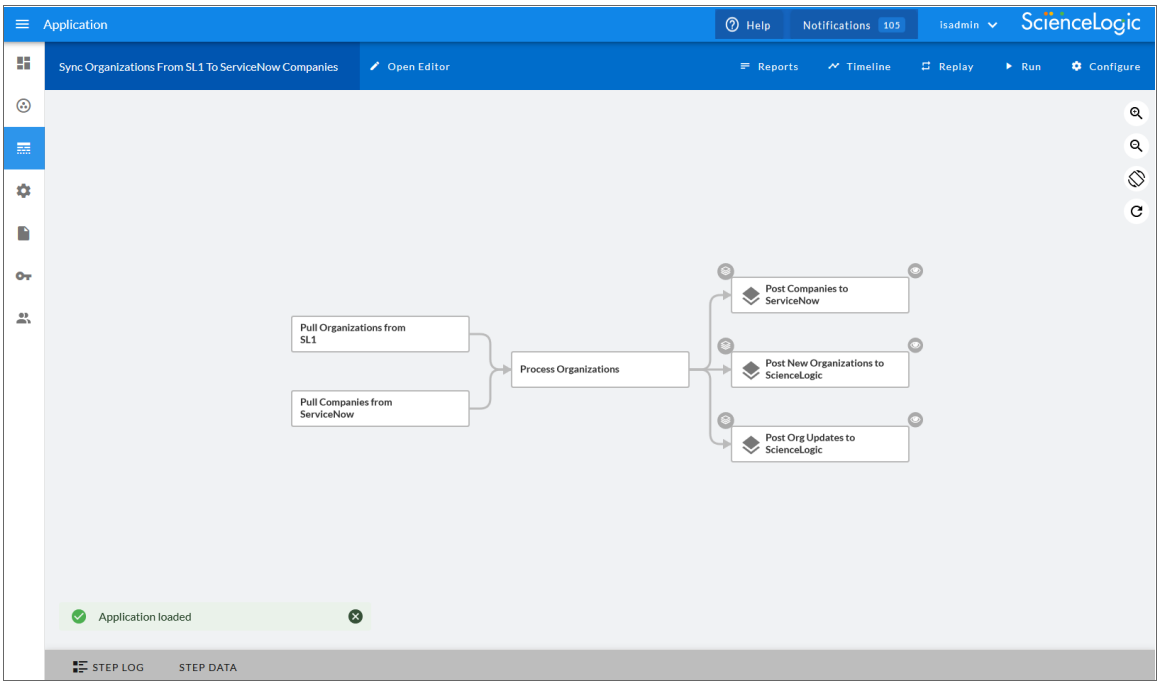
Some of the applications on the **Applications** page are *internal* applications that you should not run directly. Instead, other "parent" applications run these internal applications. To view the internal applications, click the Filter icon (⏷) at the top right of the **Applications** page and select *Show Hidden Applications*. Internal applications are hidden by default.

TIP: To view the applications that belong to only one SyncPack or some of the SyncPacks, click the Filter icon (⏷) and select the SyncPacks that you want to view from the ***Filter by SyncPack*** drop-down. To go back to seeing all applications on the **Applications** page, click **Clear selected items**.

TIP: To open the **Notification Center** pane, which contains a list of all of the pop-up messages about PowerFlow applications that were run successfully or with warnings or failures, click your user name in the navigation bar in the top right and select *Notifications*. The different notifications are color-coded: green for success, yellow for warning, and red for failure. The number of notifications displays as a badge in the menu. For more information about a notification, click the link for the notification and review the **Step Log** and **Step Data** tabs for the application steps.

Elements of an Application Page

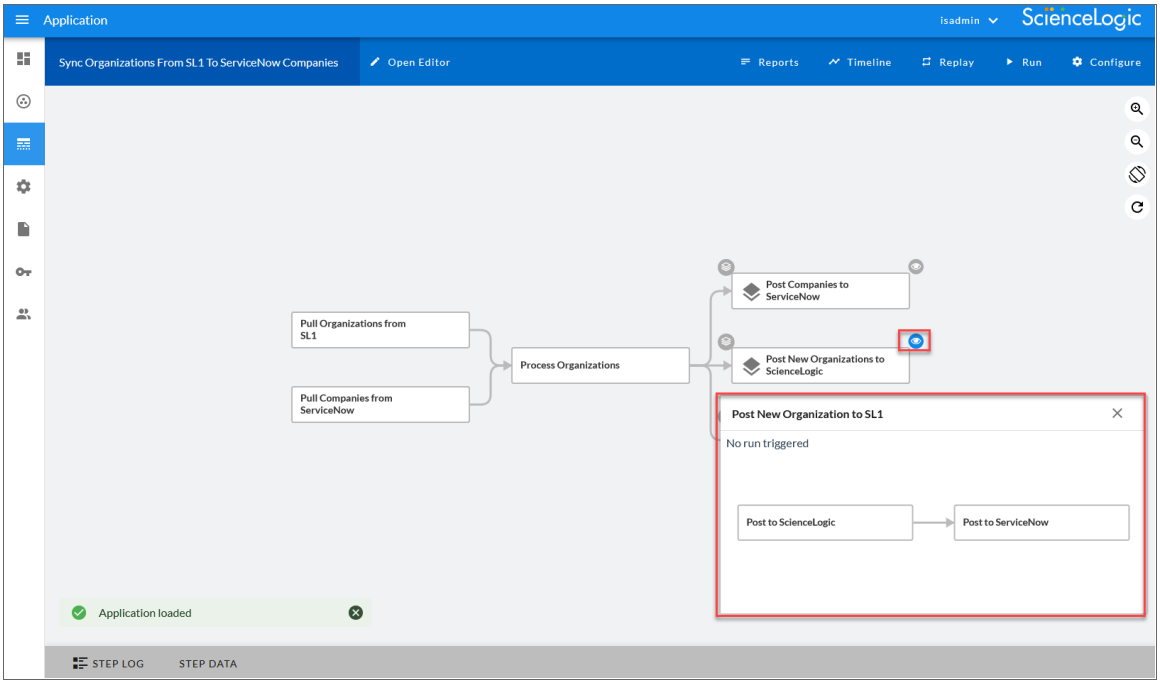
When you click the name of an application on the **Applications** page, an **Application** detail page appears:



The **Application** detail page contains the logic for the application. In the main viewing pane, the steps for the application are organized as a flowchart. Each rectangular block is a step, and the arrows indicate the order in which the steps will execute when you run the application. The color of each step changes to show the progress of the application run as it runs: green for success, red for failure, and blue for running.

If a step triggers a child application, a blue information icon ( ) appears in the upper left corner of the step. Click the icon to display the triggered application's run as a link in a pop-up window, which lists the run IDs by Success, Failure, or In Progress. You can click the link in the pop-up window to view the detail page for the triggered application. If no run ID is present, the pop-up window displays "No runs available".

Also, you can click the eye icon ( ) next to a step to open a smaller window, also called a "picture-within-a-picture", that displays the step or steps for the triggered application:



Buttons

The buttons at the top of an **Application** page include the following:

- **[Open Editor]**. Opens the **Steps Registry** pane and launches the PowerFlow builder interface. The **Steps Registry** pane contains a list of all available steps in the PowerFlow builder.

  After you click **[Open Editor]** button, the following buttons appear in the top navigation bar:

  - **[Close Editor]**. Closes the **Steps Registry** pane.
  - **[Save]**. Lets you save any changes to the steps and application. You can also save the edited application as a new application with new metadata using the *Save as* option.
  - **[Metadata]**. Opens the **Integration Metadata** window for that application so you can update the description, version, or author of the application.

  For more information, see *Editing an Application*.

> **NOTE:** If your current ScienceLogic SL1 solution subscription does not include the SL1 PowerFlow builder, contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

- **[Reports]**. Displays a report of the results of this application, where applicable. For more information, see *Generating and Viewing Reports for PowerFlow Applications*.
- **[Timeline]**. Opens the Timeline view at the top of the window where you can view past runs of this application, and whether the application failed or succeeded. For more information, see *Viewing Previous Runs of an Application*.
- **[Replay]**. Replays the last run of this application. If you hover over this, you can choose from the following options:
    - *Info Replay*. Replay the last run the application in Debug Mode, which provides more log data in the Step Logs to help you with troubleshooting.
    - *Custom Replay*. Replay the last run of the application with custom parameters for testing or troubleshooting.
- **[Run]**. Runs the application if you click the button without hovering over it. If you hover over the **[Run]** button, you can choose from the following options:
    - *Info Run*. Run the application in Debug Mode, which provides more log data in the Step Logs to help you with troubleshooting.
    - *Custom Run*. Run the application with custom parameters for testing or troubleshooting.

> **TIP:** You can click **[Run]** to run an application and then navigate to another page in the PowerFlow, and the application will complete on its own.

For more information, see *Running a PowerFlow Application*.

- **[Configure]**. Opens the **Configuration** pane for the application, where you can change the configuration object aligned with the application and edit other configuration variables as needed. For more information, see *Aligning a Configuration Object with an Application*.

> **TIP:** Click the **Zoom** icons (,) to change the size of the steps on the PowerFlow builder.

> **TIP:** Click the **Rotate** icon ( ) to turn the PowerFlow builder 90 degrees. Starting with PowerFlow Platform version 2.4.0, the flowcharts display horizontally by default instead of vertically.

> **TIP:** Click the **Manual refresh** icon ( ) to refresh the **Application** page.

Status Messages

In the bottom left-hand corner of the page, pop-up messages appear temporarily with the status of the application or an action. In the example above, the status is "Run: success". Click the close button on the

message to close the message.

Step Pane

The **Step** pane at the bottom of an **Application** page displays two tabs:

- **[Step Log]**. Displays the time, the type of log, and the log messages from a step that you selected in the main pane. All times that are displayed in this pane are in seconds.
- **[Step Data]**. Displays the JSON data that was generated by the selected step.

Click the **Step** pane again to hide the pane.

> **TIP:** For longer step log messages, click the down arrow icon ( ) in the **Message** column of the **[Step Log]** tab to open the message. To copy a message, triple-click the text of the message to highlight the entire text block, and then click the Copy Message icon ( ) next to that message. To copy the entire log, click the **[Copy Log]** button.

Creating a Basic PowerFlow Application

On the **Applications** page, you can create new applications using the PowerFlow builder interface.

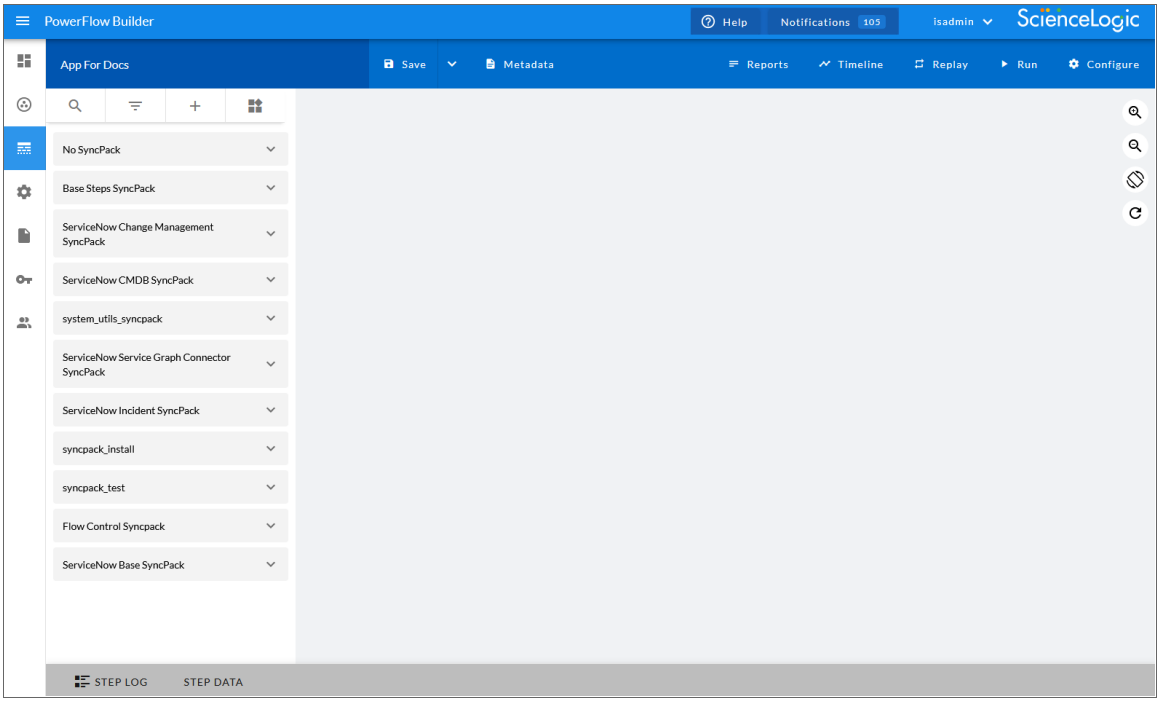
> **TIP:** To add a flow control operator to an application, such as a **Condition**, a **Transform**, or a **Trigger Application** operator, see *Working with Flow Control Operators*.

> **NOTE:** If your current ScienceLogic SL1 solution subscription does not include the SL1 PowerFlow builder, contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

To create a basic PowerFlow application:

1. From the **Applications** page ( ), click **[Create Application]**. A **Create Application** window appears.
2. Complete the following fields:
   - ***Friendly Name***. The name that you want users to see for this application. Required.
   - ***Description***. A short description of what this application does.
   - ***Author***. The name of the person or company that created this application. Use the same name for multiple applications. Required.
   - ***Version***. The version for this application.
   - ***Configuration***. Select a configuration object to align with the new application, or select *Make New Configuration* to create a new configuration object.

3. Click **[Set Values]**. The PowerFlow builder interface appears:



4. On the **Steps Registry** pane, search for a step or filter the list of steps to help you find the step you need:
   - Click the **[Search Steps]** tab ( ) to search the entire list of steps from the *Search Steps Registry* field.
   - Click the **[Group Steps]** tab ( ) to view the steps by SyncPack, by a tag, or to show all of the steps in one list.

> **TIP:** Click the **[Actions]** button ( ) on a step in the **Steps Registry** pane to view more information about that step, including the step ID, the SyncPack for that step, the version, and creator of the step. You can also click **[Edit Step Code]** to edit the code for that step, and if the step does not belong to a published SyncPack, you can also delete that step from the registry.

5. On the **Steps Registry** pane, select the step you want to add and drag it to the main viewing pane ("canvas"). The **Configuration** pane for that step appears.
6. On the **Configuration** pane, type a new name for the step and update the other fields on the pane as needed. If needed, click the down arrow on the **Advanced** section to update the advanced fields. For example, if you are getting data from SL1, you would type the IP address for your SL1 system, along with the API endpoint in the ***prefix_url*** field, such as **10.10.10.1/api/device**.
7. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** at the bottom of the pane to run the step.

8. On the **Configuration** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again. Clicking **[Save]** on the **Parameters** pane only saves the settings for this specific step; it does not save the new application.
9. Click **[Save]** in the top navigation bar to save the new application.
10. To test your application so far, click the **[Run]** button in the top navigation bar. Click the **[Step Log]** to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
11. Repeat steps 4-10 to add more steps to the application.
12. To connect one step to another, click on the bottom of the first step and drag the mouse to the second step. An arrow appears between the two steps, which you can click and drag to reposition.
13. To adjust the position of any step on the main viewing pane, click the step you want to move and drag it to its new location. To *remove* a step from the main viewing pane, click the ellipsis icon (•••) on the step and select *Delete*.
14. Click **[Save]** to save your work.

> **TIP:** To add a flow control operator to an application, such as a **Condition**, a **Transform**, or a **Trigger Application** operator, see *Working with Flow Control Operators*.

15. On the **Application** detail page, add any additional steps and operators to the new application, and then click **[Save]** and the **[Close Editor]** button. The application is added to the **Applications** page.

Working with Flow Control Operators

When you are creating or editing a application in the PowerFlow builder interface, you can use "flow control" operators, including the following operators that you can drag and drop onto the step workflow on the canvas:

- The **Condition** operator ( ) lets you create for branching workflows, such as If-Else or If-Then-Else statements. The **Conditional Wizard** pane lets you modify the conditions that enable branching in the workflow. For more information, see *Creating an Application with a Condition Operator*.
- The **Transform** operator ( ) lets the application pull data gathered by a previous step and modify or transform that data to fit into the next step. The **Transform Wizard** pane lets you specify which data you want to use or transform from the previous steps. For more information, see *Creating an Application with a Transform Operator*.
- The **Trigger Application** operator ( ) lets you launch another PowerFlow application from within a new or existing PowerFlow application. The **Trigger App Wizard** pane lets you select the child application you want to launch from the current (parent) application and specify the data that you want to pull from the child application. For more information, see *Creating an Application with a Trigger Application Operator*.

You can add more than one type of operator to the same PowerFlow application, and you can add more than one operator with the same type as well.

> **NOTE:** The "Flow Control" SyncPack contains the "IfStep" step that enables the logical branching used by the PowerFlow builder. This SyncPack is included in PowerFlow.

Creating an Application with a Condition Operator

You can drag a **Condition** operator ( ) onto an application workflow to create the option for branching flows, such as If-Else or If-Then-Else statements.

To create an automation that contains a **Condition** operator:

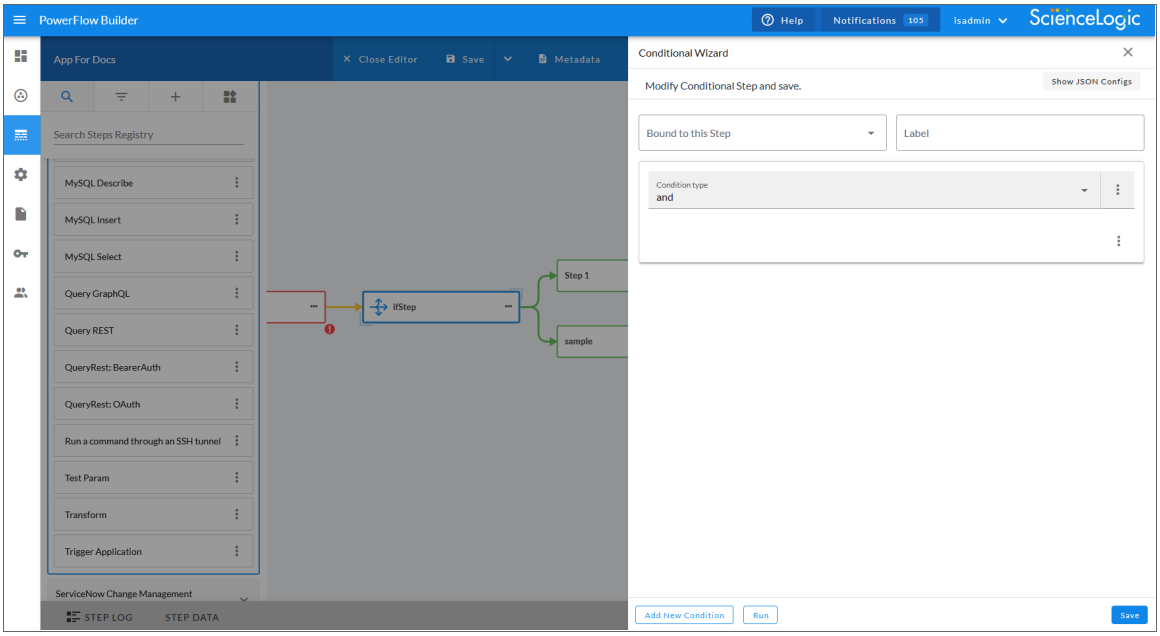
1. From the **Applications** page ( ), click **[Create Application]**. A **Create Application** window appears.
2. Complete the following fields:
    - ***Friendly Name***. The name that you want users to see for this application. Required.
    - ***Description***. A short description of what this application does.
    - ***Author***. The name of the person or company that created this application. Use the same name for multiple applications. Required.
    - ***Version***. The version for this application.
    - ***Configuration***. Select a configuration object to align with the new application, or select *Make New Configuration* to create a new configuration object.
3. Click **[Set Values]**. The PowerFlow builder interface appears.
4. On the **Steps Registry** pane, search for a step or filter the list of steps to help you find the step you need.
5. On the **Steps Registry** pane, click the step you want to add and drag it to the main viewing pane ("canvas"). The **Configuration** pane for that step appears.

6. On the **Configuration** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
7. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** at the bottom of the pane to run the step.
8. On the **Configuration** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
9. Click **[Save]** in the top navigation bar to save the new application.
10. To test your application so far, click the **[Run]** button in the top navigation bar. Click the **[Step Log]** to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
11. Repeat steps 4-10 to add more steps to the application.
12. Click **[Save]** to save your work.
13. Click the **[Run]** button to run the new application and gather data for the steps that will send data to the **Condition** operator.
14. On the **Steps Registry** pane, click the **[Advanced]** tab ( ). The flow control operators appear.
15. To add the option for branching flows, such as If-Else or If-Then-Else statements, drag the **Condition** operator ( ) onto the canvas. The operator displays as a step with an "ifStep" label.
16. Connect the steps for the branching workflow to the **Condition** operator by clicking the outline of each step and dragging the arrow that appears to the **Condition** operator. Repeat this process for all of the steps that should be part of the branching workflow.
17. Click the [Save] button and then click the **[Run]** button to gather data again for all of the steps that will send data to the **Transform** operator.

> **TIP:** The arrows connecting the steps and the **Condition** operator display as green to show the flow of the data, based on the values in the application. In Edit mode, you can click the blue circle above a branched step to open the **Conditional Wizard** pane to see the conditions for that step.

18. Click the ellipsis icon (•••) on the **Condition** operator and select *Configure*. The **Conditional Wizard** pane for that operator appears.

19. Click **[Add New Condition]** to configure the conditions for the operator:



20. Complete the following fields on the **Conditional Wizard** pane for each step you want to include in the branching:

    - ***Bound to this Step***. Specify the step connected to the **Condition** operator.
    - ***Label***. Type a brief description of the branching condition, such as "Change Request Found". This text will display above the selected step when you are out of Edit mode.

- ***Condition type***. Select the condition that this step needs to meet to allow branching. The condition type you select here determines what options display in the field or fields below this field. You can use variables from the previous step, which you can find on the **[Step Data]** tab of the **Step** pane for that step. Your options include:

> **NOTE:** Each of the conditions can accept a dynamic value for the Value field. The available information for this field can be chosen from the available application variables or the data the previous steps connected to the Conditional Step generate. They can be referred to using the syntax `${previous_step_name.property_to_compare}` or `${all_previous_data.name_of_the_var_you_want}` or `${appvar.appvar_existing_name}`. If the step has spaces in its name, replace them with underscores (_).

  - *equal*. In the ***Equal to*** field, type the string to compare. Boolean values should be typed as *True* or *False*. In the ***Value*** field, specify the variable that should be equal to the value in the ***Equal*** field. For example: `${Collect_PF_Configuration_Data.author}` for the Equal field, and *Sciencelogic* for the Value field

  - *numeric*. In the ***Value*** field, specify the variable that should fall within the range specified by the ***Above*** and ***Below*** fields . For example: For the ***Below*** field *1729530125* can be chosen and *1729530121* for the ***Above*** field. For the ***Value*** field the following can be chosen `${Collect_PF_Configuration_Data.last_modified}`.

  - *template*. In the ***Value*** field, type a condition that can include numeric or another kind or comparison. For example: `${Collect_PF_Configuration_Data.last_modified} == 1729530122`

  - *and*. Joins two or more conditions with the AND logic operator, and these conditions can be simple or complex conditions. This is the default option.

  - *not*. Inverts the result of the simple condition type. This condition can contain only one condition inside of it.

  - *or*. Joins two or more conditions with the OR logic operator.

> **TIP:** In a ***Value*** field under the ***Condition type***, you can also click the **[Actions]** button (⋮) and click *Select property* to use data from one of the previous steps, if available. This is available when using the Condition Type template, and the properties are available after running the previous steps.

21. You can add a sub-condition for the same step by clicking the **[Actions]** button (⋮) for the ***Condition type*** field and selecting Add *sub-condition*. You can also click the **[New condition]** button to add a sub-condition.

22. To add another set of conditions in addition to the first set of conditions, click the **[Add New Condition]** button and repeat steps 20-21.
23. Edit the remaining conditions on the **Conditional Wizard** pane, and then click **[Save]**. The **Application** detail page appears.
24. Add additional steps and operators to the new application as needed, and then click the **[Save]** and the **[Close Editor]** buttons. The application is added to the **Applications** page.

Creating an Application with a Transform Operator

In the PowerFlow builder interface, you can drag a **Transform** operator ( ) from the **Steps Registry** pane onto an application workflow.

The **Transform** operator can pull data gathered by a previous step, and then modify or transform the data to fit into the next step. The operator uses a Jinja2 template to merge the data from previous steps.

For example, you might want to use the **Transform** operator if a step in an application is pulling in a large amount of data, but you only want to focus on a specific sub-set of that data. The **Transform** operator lets you filter that data to show only the data you need, and you can use that data in the following steps of the application.

To create an application with a **Trigger Application** operator:

1. From the **Applications** page ( ), click **[Create Application]**. A **Create Application** window appears.
2. Complete the following fields:
    - ***Friendly Name***. The name that you want users to see for this application. Required.
    - ***Description***. A short description of what this application does.
    - ***Author***. The name of the person or company that created this application. Use the same name for multiple applications. Required.
    - ***Version***. The version for this application.
    - ***Configuration***. Select a configuration object to align with the new application, or select *Make New Configuration* to create a new configuration object.
3. Click **[Set Values]**. The PowerFlow builder interface appears.
4. On the **Steps Registry** pane, search for a step or filter the list of steps to help you find the step you need.
5. On the **Steps Registry** pane, click the step you want to add and drag it to the main viewing pane ("canvas"). The **Configuration** pane for that step appears.
6. On the **Configuration** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
7. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** at the bottom of the pane to run the step.
8. On the **Configuration** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
9. Click **[Save]** in the top navigation bar to save the new application.
10. To test your application so far, click the **[Run]** button in the top navigation bar. Click the **[Step Log]** to view the results of the run on the **[Step Log]** and **[Step Data]** tab.

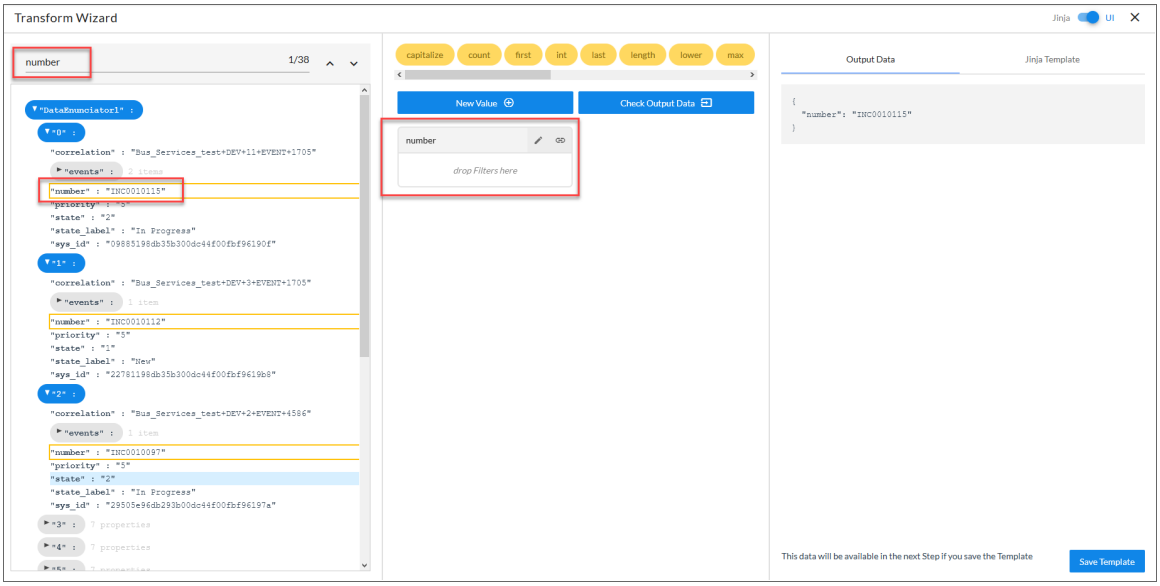
11. Repeat steps 4-10 to add more steps to the application.
12. Click **[Save]** to save your work.
13. Click **[Run]** to run the new application and gather data for the steps that will send data to the **Transform** operator.
14. On the **Steps Registry** pane, click the **[Advanced]** tab ( ). The flow control operators appear.
15. Drag the **Transform** operator ( ) onto the canvas. The operator displays as a step with a "Jinja2Template" label.
16. Locate the step or steps that contain data you want to *send* to the following step or steps. Connect the step or steps to the **Transform** operator by clicking the outline of each step and dragging the arrow that appears to the **Transform** operator.
17. Connect the step or steps that will *receive* the transformed data by clicking the outline of the **Transform** operator and dragging the arrow that appears to those steps.

> **TIP:** The arrows connecting the steps and the **Transform** operator display as green to show the flow of the data, based on the values in the application.

18. Click the **[Save]** button and then click the **[Run]** button to gather data from the step or steps that will send data to the **Transform** operator.
19. Click the ellipsis icon (•••) on the **Transform** operator. The **Transform Wizard** page appears.
20. As needed, click the ***Jinja/UI*** toggle at the top right of the page to switch between the ***Jinja*** code-only view or the ***UI*** drag-and-drop view. The default view is **UI**.
    - If you are using the ***UI*** drag-and-drop view, see steps 21 to 26.
    - If you are using the ***Jinja*** code-only view, see steps 27 to 28.

> **TIP:** Use the ***Search*** field at the top of the left-hand pane to search for the data you want to transform. Use the up and down arrows to move through the search results.

21. In the ***UI*** drag-and-drop view, search for the value that you want to transform in the first pane and drag and drop that value onto the middle pane. A new value box appears in the middle pane:



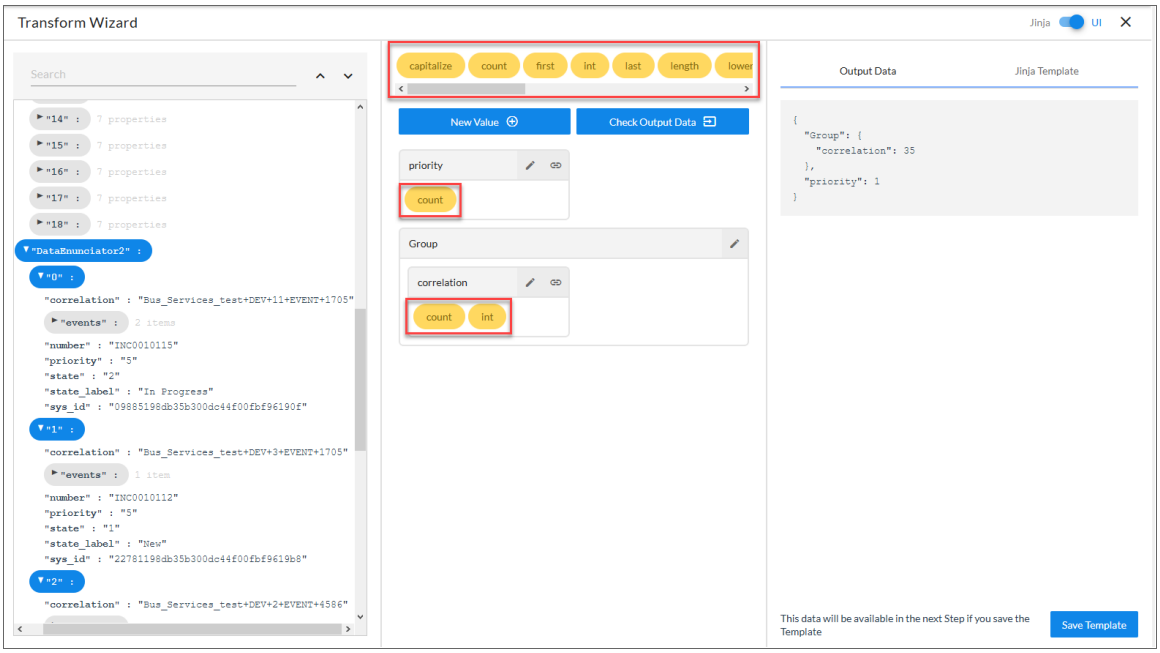
You can drag and drop add multiple values from the first pane onto the middle pane. PowerFlow uses these values to create the Jinja template for this **Transform** step. You can also drag and drop a group of values onto the middle pane by clicking the blue oval at the top of the list of values.

> **TIP:** To edit the name of a value box in the middle pane, click the pencil icon (✎) and type a new name. To see the link between a value from the first pane and the value in the middle pane, click the link icon (🔗). To view the list of values from that value box that will be added to the Jinja2 template, click the Preview icon (⎋) .

22. To delete a value box from the middle pane, drag the box toward the bottom of the middle pane. A "drop here to delete" rectangle appears, and when you drag the box into that rectangle, it turns red. Drop the value box into the red rectangle to delete the box.

23. To create a new value box, click the **[New Value]** button and type a name for it in the middle pane. Then you can drag and drop data from the first pane into that value box.

24. To use a Jinja filter on a value in the middle pane, drag one or more of the yellow filter ovals into the relevant value box in the middle pane. The user interface will show a "Not supported" message for any filters that are not compatible with certain value types, such as a "capitalize" filter with a number value. For more information about Jinja filters, see the List of Built-in Filters in the Jinja documentation.



> **TIP:** If you add a "select" or a "reject" filter to a value box, additional fields will appear at the bottom of the middle pane when you add one of those filters. Depending on the filter you chose, type the value you want to include or reject in the ***Name*** field that appears, and select *String* or *Number* as needed. Click **[Save]** to finish configuring the filter.

25. When you are done adding values from the first pane and adding filters from the middle pane to the various value boxes, click **[Check Output Data]**. The data for the Jinja2 template you just created appears in the **[Output Data]** tab, while the actual code for the template appears in the **[Jinja Template]** tab.

26. If you are satisfied with the results of the Jinja2 template the **[Jinja Template]** tab and the data on the **[Output Data]** tab, go to step 29.

27. In the ***Jinja*** code-only view, on the first pane of the **Transform Wizard** page, you can view all of the data gathered from the steps that you selected to send to the **Transform** operator in step 13. You will use this data to create a Jinja2 template on the **[Jinja Template]** tab.

> **TIP:** The data on the first pane matches the data you would see if you selected each step connected to the **Transform** operator and clicked the **[Step Data]** tab.

28. On the **[Jinja Template]** tab, you can use data from the first pane to create a Jinja2 template to manipulate and change that data to use in the next step of the application. A ***Jinja2 template*** lets you create complex, concatenated (linked) fields. For more information about Jinja2 Templates, see the Template Designer Documentation.

Example 1

For a simple example for a step that gathers specific data about a device, you could create the following Jinja2 template on the **[Jinja Template]** tab:

```
{% set output =

{"sys_name": input["name"],

"sys_ip": input["ip"]} %}{{output|tojson}}
```

> **TIP:** Click **[Check the Output Data]** on the **[Jinja Template]** tab to run the Jinja2 template you created on the **[Jinja Template]** tab. The results of this process appear on the **[Output Data]** tab.

After running the above Jinja2 template, you would see the following data on the **[Output Data]** tab:

```
{

"sys_ip": "10.2.11.154",

"sys_name": "pm-aio-11-154"

}
```

Example 2

For a more complicated example, you could create the following Jinja2 template to gather and merge data from the "GetIncidents" and the "GetCompletedCRs" steps:

```
{% set d=dict() %} {% set crupdate =

d.update(

{'changeRequests': GetCompletedCRs,

'correlation': GetIncidents.0.correlation,

'device_id': GetIncidents.0.events.0.device.id,

'event_id': GetIncidents.0.events.0.event_id,

'device_name': GetIncidents.0.events.0.device.name,
```

```
'device_sysid': GetIncidents.0.events.0.device.sys_id,

'found_cr': GetIncidents.0.sys_id in GetCompletedCRs,

'incident_number': GetIncidents.0.number,

'incident_sys_id': GetIncidents.0.sys_id }) %} {{ d|tojson }} ]
```

After running the above Jinja2 template, you would see the following data on the **[Output Data]** tab:

```
{

"changeRequests": [],

"correlation": "Bus_Services_test+DEV+15+EVENT+3636",

"device_id": "15",

"device_name": "enagley-cmdr-34-242",

"device_sysid": "7b01681edb1df300dc44f00fbf9619e7",

"event_id": "12503",

"found_cr": false,

"incident_number": "INC0010104",

"incident_sys_id": "06685154db35b300dc44f00fbf961933"

}
```

29. As needed, edit any other data on the **Transform Wizard** page, and then click the **[Save Template]** button.
30. Close the **Transform Wizard** page. The **Application** detail page appears again.
31. Click the **[Save]** button to save your work.

> **TIP:** Click the **[Run]** button to test the new application. When the run completes, select the **Transform** operator and click the **[Step Data]** tab to view the filtered set of data.

32. Add any additional steps and operators to the new application, and then click the **[Save]** button and then the **[Close Editor]** button. The application is added to the **Applications** page.

Creating an Application that Uses a Trigger Application Operator

In the PowerFlow builder interface, you can use the **Trigger Application** operator ( ) to launch one or more PowerFlow applications from within a new or existing PowerFlow application. This operator uses the same functionality as the "Trigger Application" step from the Base Steps SyncPack.

The **Trigger Application** operator gathers data from a previous step in the application workflow, in the same way as the **Transform** operator. You can use the **Trigger Application** operator to organize specific data that will be used in the triggered application or applications.

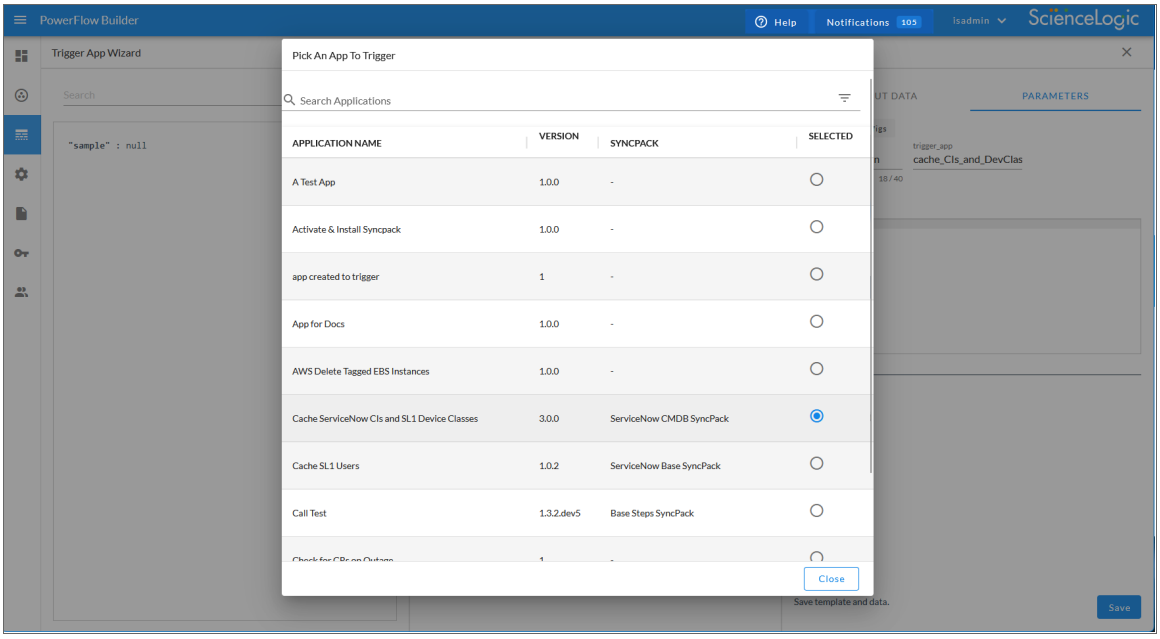
1. From the **Applications** page ( ), click **[Create Application]**. A **Create Application** window appears.
2. Complete the following fields:
    - ***Friendly Name***. The name that you want users to see for this application. Required.
    - ***Description***. A short description of what this application does.
    - ***Author***. The name of the person or company that created this application. Use the same name for multiple applications. Required.
    - ***Version***. The version for this application.
    - ***Configuration***. Select a configuration object to align with the new application, or select *Make New Configuration* to create a new configuration object.
3. Click **[Set Values]**. The PowerFlow builder interface appears.
4. On the **Steps Registry** pane, search for a step or filter the list of steps to help you find the step you need.
5. On the **Steps Registry** pane, click the step you want to add and drag it to the main viewing pane ("canvas"). The **Configuration** pane for that step appears.
6. On the **Configuration** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
7. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** at the bottom of the pane to run the step.
8. On the **Configuration** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
9. Click **[Save]** in the top navigation bar to save the new application.
10. To test your application so far, click the **[Run]** button in the top navigation bar. Click the **[Step Log]** to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
11. Repeat steps 4-10 to add more steps to the application.
12. Click **[Save]** to save your work.
13. Click the **[Run]** button to run the new application and gather data for the steps that will send data to the **Trigger Application** operator.
14. On the **Steps Registry** pane, click the **[Advanced]** tab ( ). The flow control operators appear.
15. Drag the **Trigger Application** operator () onto the canvas. The operator displays as a step with a "TriggerApplication" label.

16. Connect an existing step to the **Trigger Application** operator by clicking the outline of the step and dragging the arrow that appears to the operator.

17. Add additional steps as needed and click the **[Save]** button.

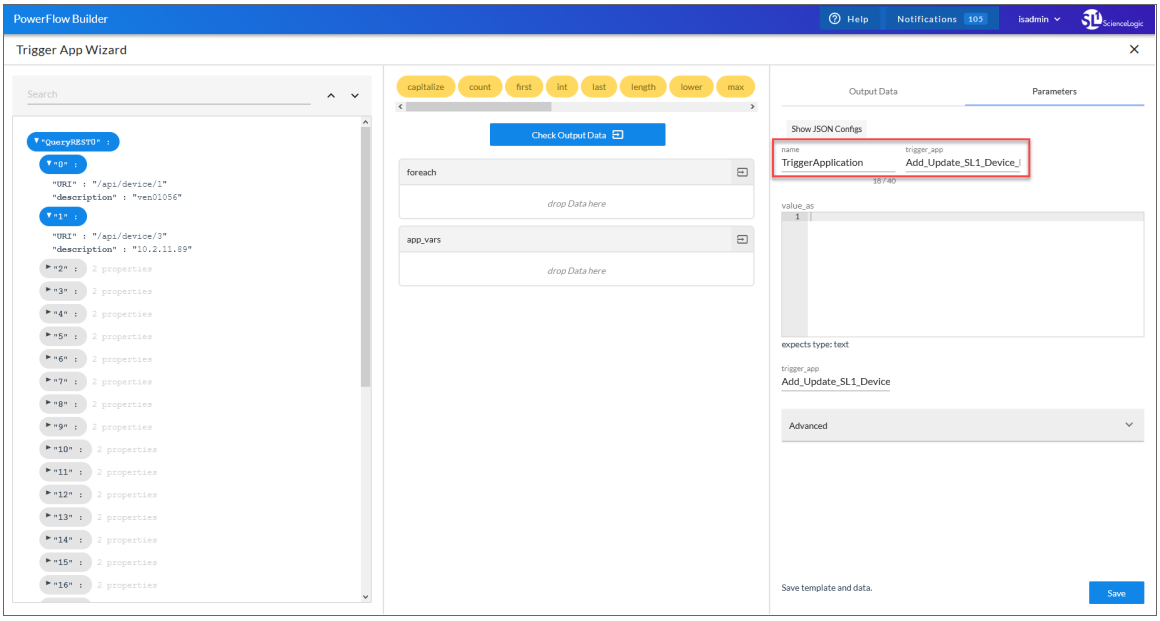
18. Click the **[Run]** button to run the new application and gather data for the **Trigger Application** operator.

19. Click the ellipsis icon (•••) on the **Trigger Application** operator. The **Pick an App to Trigger** window for the **Trigger App Wizard** page appears:



20. Select the PowerFlow application that you want to trigger with this step.

21. Click **[Close]**. The main **Trigger App Wizard** page appears, with the name of the application you selected added to the ***name*** (friendly name) and ***trigger_app*** (system name) fields of the **[Parameters]** tab on the far right of the window:



TIP: To change the application that you want to trigger, click the application name in the ***trigger_app*** field, and the **Pick an App to Trigger** window appears again.

22. In the first pane, search for the values that you want to use in the triggered application and drag and drop those values onto a value box in the middle pane:

- ***foreach***: Triggers one or more instances of the application you specified in step 16 for each value in the list or dictionary. To enable this feature, you will need to specify the ***input_iterator***, ***key_as***, and ***value_as*** parameters on the **Advanced** section of the **[Parameters]** tab. For more information, see the *Parameters table*.
- ***app_vars***: Values added to this box are used as application variables for the triggered application.

NOTE: The middle pane works as a "transform wizard" that adds data to the **[Output Data]** and **[Parameters]** tabs of the third pane.

TIP: To edit the name of a value box in the middle pane, click the pencil icon (✎) and type a new name. To see the link between a value from the first pane and the value in the middle pane, click the link icon (🔗). To view the list of values from that value box that will be included in the output data, click the Preview icon (⇥).

23. To delete a value box from the middle pane, drag the box toward the bottom of the middle pane. A "drop here to delete" rectangle appears, and when you drag the box into that rectangle, it turns red. Drop the value box into the red rectangle to delete the box.

24. To use a Jinja filter on a value in the middle pane, drag one or more of the yellow filter ovals into the relevant value box in the middle pane. The user interface will show a "Not supported" message for any filters that are not compatible with certain value types, such as a "capitalize" filter with a number value. For more information about Jinja filters, see the List of Built-in Filters in the Jinja documentation.

> **TIP:** If you add a "select" or a "reject" filter to a value box, additional fields will appear at the bottom of the middle pane when you add one of those filters. Depending on the filter you chose, type the value you want to include or reject in the ***Name*** field that appears, and select *String* or *Number* as needed. Click **[Save]** to finish configuring the filter.

25. When you are done adding values from the first pane and adding filters from the middle pane to the various value boxes, click **[Check Output Data]**. The data you added in the middle pane is formatted and added to the **[Output Data]** tab.

26. As needed, update the values on the **[Parameters]** tab for the triggered application. For more information, see the ***Parameters table***.

> **NOTE:** If you have a configuration object aligned with this application, that configuration object will also be used by the triggered applications. However, the values from the configuration object ill be overwritten by any of the parameters you set in the **Trigger App Wizard** page.

27. Edit any other data on the **Trigger App Wizard** page, and then click **[Save]**. the **Application** detail page appears.

28. Click the **[Save]** button to save your work.

> **TIP:** Click the **[Run]** button to test the new application. When the run completes, select the **Trigger Application** operator and click the **[Step Data]** to view the filtered set of data that will be sent to the triggered application.

29. Add additional steps and operators to the new application as needed, and then click the **[Save]** and the **[Close Editor]** buttons. The application is added to the **Applications** page.

Parameters Table

The following table describes the different parameters you can set on the **[Parameters]** tab of the **Trigger App Wizard** page:

Parameter Name	Default Value	Description
name	TriggerApplication	Unique name of this step. Edit this field as needed.
trigger_app	N/A	The name of the PowerFlow application you want to run or

Parameter Name	Default Value	Description
		"trigger" from the current application. This value is automatically set to the system name for the application you selected from the **Pick an App to Trigger** window.
value_as	N/A	• If the input in the ***foreach*** value box in the middle pane of the wizard page is a dictionary, specify the name of the application variable for those values. • If the input in the ***foreach*** value box is a list, the application variable will be set with the elements of the list.
Advanced parameters		
retry_countdown	180	The interval between retries, in seconds.
retry_max	0	The maximum number of times the PowerFlow system will retry to execute the step before it stops retrying and logs a step failure.
retry_backoff	unselected	Instead of using a defined interval between retries, the PowerFlow system will incrementally increase the interval between retries.
retry_jitter	unselected	Instead of using a defined interval between retries, the PowerFlow system will retry the step execution at random intervals.
retry_backoff_max	600	The maximum time interval for the ***retry_backoff*** option, in seconds.
template	N/A	Displays the Jinja2 template that is used for rendering the desired data.
foreach	N/A	The contents of the ***foreach*** value box in the middle pane of the **Trigger App Wizard** page triggers multiple applications for each value in that list or dictionary. Any values you specify in this text box will get overwritten if values are added to the ***foreach*** value box in the middle pane of the wizard.
input_iterator	false	• Set to "true" if you want to execute an instance of the triggered application for each item in the set of input data from the previous step. The data is added as an application variable to the triggered application, and that variable is defined by the ***value_as*** parameter . This option is not compatible with the ***foreach*** parameter. • Set to "false" if you only want to trigger one instance of the triggered application.
key_as	N/A	If the input in the ***foreach*** value box is a dictionary, specify the name of application variable for the key.
app_vars	N/A	Specify any application variables you want to add to the triggered applications. Any values you specify in this text box

Parameter Name	Default Value	Description
		will get overwritten if values are added to the ***app_vars*** value box in the middle pane of the wizard page.
exit_on_child_ failure	Selected	If you select this option, if any triggered application fails, the step will be marked as a Failure. If you do not select this option, the step will not fail if a triggered application fails, but the step still wait until all triggered apps are completed. The ***wait_for_child_completion*** option, below, must be selected for this option to be enabled.
wait_for_child_ completion	Selected	Select this option if you want PowerFlow to wait for the triggered applications to complete. If this parameter is not selected, the application ends as soon as the applications are triggered. In either case, if an error occurred with triggering an application, the state of the step is Failed.

TIP: If you receive an error running the step, select the step and click the *Step Log* tab. The error log should list which of the above parameters might be causing the error.

Editing a PowerFlow Application

In the PowerFlow builder interface, you can edit an existing application and its steps. You can also add and remove steps from that application.

> **NOTE:** You cannot overwrite applications where *ScienceLogic, Inc.* is listed as the "Author", but you can edit a ScienceLogic application and save it with a different name.

To edit an application:

1. From the **Applications** page, select the application that you want to edit. The **Application** detail page appears.
2. Click the **[Open Editor]** button. The PowerFlow builder interface appears, including the **Steps Registry** pane, which contains a list of all of the available steps.
3. On the **Steps Registry** pane, you can search for a step or filter the list of steps to help you find the step you need:
   - Click the **[Search Steps]** tab ( ) to search the entire list of steps from the *Search Steps Registry* field.
   - Click the **[Group Steps]** tab ( ) to group the steps by SyncPack, by a tag, or to show all of the steps in one list.

   > **TIP:** Click the **[Actions]** button ( ) on a step in the **Steps Registry** pane to view more information about that step, including the step ID, the SyncPack for that step, the version, and creator of the step. You can also click **[Edit Step Code]** to edit the code for that step, and if the step does not belong to a published SyncPack, you can also delete that step from the registry.

4. To create a step, click **[Create a Step]** (+) on the **Steps Registry** pane, type a file name, and edit the step code for the new step. For more information, see *Creating a Step*.
5. To use existing steps, click the step or steps you want to add from the **Steps Registry** pane and drag them to the main viewing pane ("canvas") to add them to the application. Add any relevant information to the **Configuration** pane for that step, and click **[Save]** to close the **Configuration** pane.
6. To adjust the position of any step in the application, click the step you want to move and drag it to its new location.
7. To redirect the arrow between steps, click the arrow and drag it to reposition it.
8. To edit the configuration for a step, click the ellipsis icon (•••) on the step and update the fields as needed on the **Configuration** pane. You can also click the ellipsis icon (•••) on a step to view he step code for the step or to delete the step.
9. While editing a step, click **[Show JSON Configs]** to view the JSON configuration data for the step. Click **[Hide JSON Editor]** to view the fields instead.
10. To remove a step, click the step to select it and press the **[Delete]** key on your keyboard.

11. To edit the metadata for an application, click the **[Metadata]** button and update the fields in the pop-up as needed.
12. To save the changes you made to the application, click **[Save]**. You can also click the *Save as* option to save the application with a new name.
13. To stop editing and close the **Search Steps Registry** pane, click the **[Close Editor]** button.

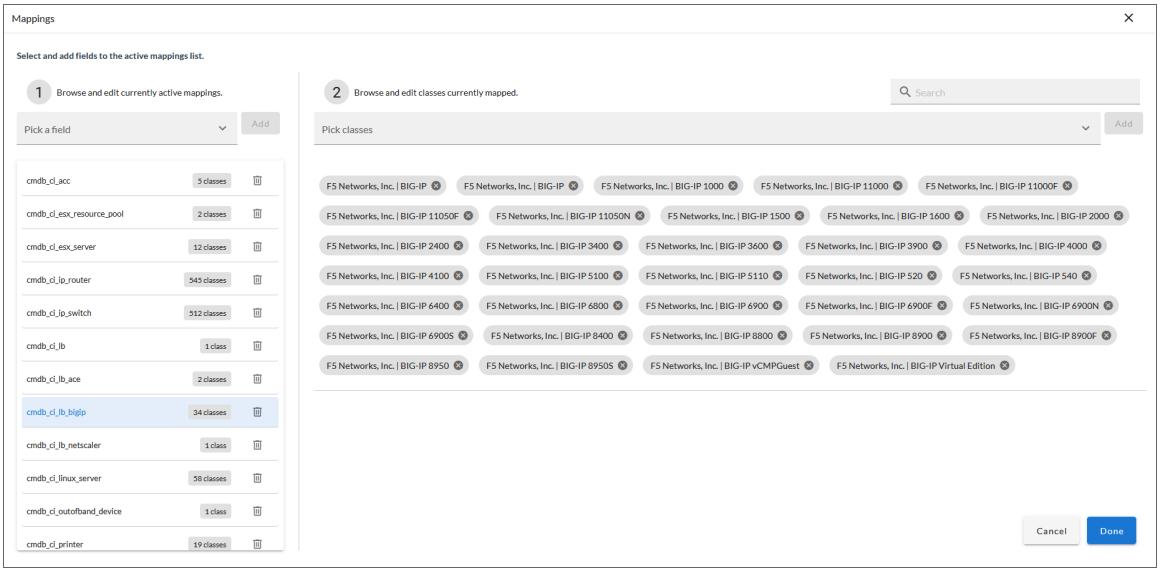
Editing Mappings in a PowerFlow Application

Some PowerFlow applications have one or more ***mappings*** sections, which lets you sync specific data in a third-party application with SL1 data.

There are three types of mappings available on the **Configuration** pane for syncing data, and currently these mappings are only available for specific applications in the "ServiceNow CMDB" SyncPack:

- **Mappings**. On this mapping page for Device Sync and CI Attribute Sync, you can connect an SL1 device class to a ServiceNow CI class, which determines the CI class that ServiceNow uses when creating the CI in ServiceNow. If no mappings are provided for CI Attribute Sync, the sync will only pull CI classes provided in the Device Sync **Mappings** section; if you add mappings for CI Attribute Sync, only CI classes that were included in its **Mappings** section will be returned from ServiceNow. For an Interface Sync, you can map SL1 interfaces to ServiceNow tables.
- **Company Mapping Override**. On this mapping page, you can create a new mapping for the ServiceNow ***company*** field. You can override the ***company*** field with a different ServiceNow field where you can read and write the **company_sys_id** for CIs.
- **Attribute Mappings**. On this mapping page, you can create mappings for any other custom attributes that you want to sync between SL1 and ServiceNow.

All three types of mapping pages have the following layout and options:



In the first column, you can select an existing item from the third-party application (like ServiceNow) to see a list of the items that are currently mapped to that item in SL1. You can also select a field from the list and click the **[Add]** button to start a new mapping. Click the delete icon to remove a mapping.

In the second column, you can review the items mapped with the item you selected in the first column, and you can also add an item to a mapping, or remove an item from a mapping.

When you are done editing the mappings, click **[Done]** to save your changes and return to the **Configuration** pane for that application.

> **TIP:** If you are creating a PowerFlow application, you can use mappings in a similar way for your new application.

Enabling Run Book Automation Queue Retries

If you are using PowerFlow to sync incidents from a third-party application like ServiceNow or Cherwell, you can enable Run Book Action (RBA) queue retries to keep from losing any incidents if PowerFlow is unavailable. Those pending PowerFlow applications are added to an RBA queue that you can access to retry the applications that failed.

Requirements

The RBA queue retries feature has the following requirements:

- PowerFlow version 2.3.0 or later
- SL1 version 11.1.0 or later
- "ServiceNow Base Pack" PowerPack version 106 or later
- "Base Steps" SyncPack version 1.3.2 or later
- "System Utils" SyncPack version 1.1.2 or later (this SyncPack is included when you install PowerFlow Platform

PowerFlow Applications

The "System Utils" SyncPack version 1.1.2 or later includes two PowerFlow applications that handle Run Book Action retries:

- ***Read SL1 RBA Queue and Retry PowerFlow Applications***. Pulls and retries any PowerFlow applications that were not able to be executed by Run Book Actions because PowerFlow was not available.
- ***Run PowerFlow Application and Remove It from SL1 RBA Queue***. Runs any pending PowerFlow applications from the RBA queue, and then removes those applications from the RBA queue. This application gets triggered by the "Read SL1 RBA Queue..." application, so no configuration is needed for this application.

Configuration Object

The "System Utils" SyncPack also includes "PF RBA retry Configuration Example." You can make a copy of this example configuration object to use with the two applications listed above.

SL1 Action Type

The "ServiceNow Base Pack" PowerPack version 106 or later includes a new Run Book Action Type, "ServiceNow: Send to PowerFlow". You can add the following snippet code from that Action Type into the snippet code for the Run Book Action policy you want to use (if the Run Book Action policy does not already contain that code):

```
payload = {"name": "application_name",
           "params": {"configuration": "config_name", "app_var1": "app_
var1"}}
```

For example:

```
payload = { "name": "pf_controltower_healthcheck",
                    "params": {
                        "pf_host": "https://10.2.11.234",
                        "cmdb_integration": "cmdb_integration"
                    },
                }
```

You should also add the following snippet code to the Run Book Action policy if the code is not already present:

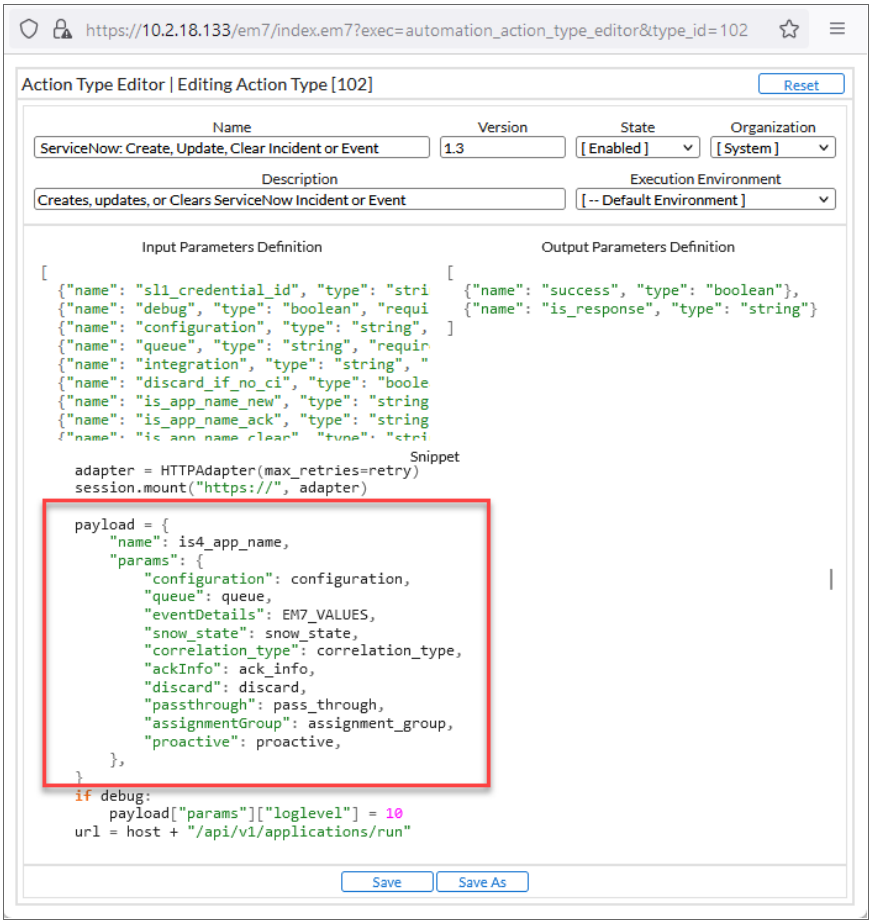
```
EM7_RESULT = {'EM7_RETRY': True, 'EM7_EXTRA_PARAMs': payload}
```

NOTE: The Run Book Action policies in "ServiceNow Base Pack" PowerPack version 106 or later contain this snippet code by default.

Enabling RBA Queue Retries

To enable RBA queue retries:

1. If you need to copy the snippet code, open SL1, navigate to the **Action Types** page (Registry > Run Book > Action Types), and click the edit icon ( ) for the "ServiceNow: Create, Update, Clear Incident or Event" Action Type or the "ServiceNow: Send to PowerFlow" Action Type. The **Action Type Editor** modal appears:



2. Scroll down and copy the `payload` section and close the **Action Type Editor** modal without saving it.
3. Go to the **Actions** page (Registry > Run Book > Action Types) and click the edit icon ( ) for the Run Book Action policy you want to use. The **Action Editor** modal appears.
4. In the **Input Parameters** pane, paste the `payload` section from step 2, along with the following required code:

5. In the **Input Parameters** pane, update the parameters in the `payload` section as needed. The following parameters are examples:
    - ***name***. Specifies the system name of the PowerFlow application you want to use for retries. The system name displays at the end of the URL, after **/integrations/**.
    - ***configuration***. Specifies the system name of the configuration object aligned with the PowerFlow application. The system name displays in the ***Configuration*** field on the **Configuration** pane of any application aligned with that configuration object.
    - ***queue***. Specifies the worker queue on which the application runs.
6. Make sure that the following snippet code is present in the **Input Parameters** pane; copy and paste it into the pane if it is not present:

```
EM7_RESULT = {'EM7_RETRY': True, 'EM7_EXTRA_PARAMs': payload}
```

> **NOTE:** You can also add the snippet code from steps 2 and 3 to the **Input Parameters** pane of a Run Book Action policy.

7. Go to the **Actions** page (Registry > Run Book > Actions) and click the edit button ( ) for the Run Book Action for which you want to enable retries.
8. Make sure that the "ServiceNow: Create, Update, Clear Incident or Event" Action Type is selected as the ***Action Type*** for that Run Book Action policy.
9. In PowerFlow, go to the **Applications** page and select the "Read SL1 RBA Queue and Retry PowerFlow Applications" application.
10. Click the **Configuration** tab and align the corresponding configuration object in the ***Configuration*** field.
11. Update the following fields as needed:
    - ***limit_failover actions***. Specifies the number of rows that will be read from the RBA queue. The default is 1000.
    - ***generate_report***. Select this option to generate a report of all of the triggered PowerFlow applications read from the RBA queue.
12. Click **[Save]**.
13. Click the **[Run]** button. The application pulls and retries any PowerFlow applications that were not able to be executed by Run Book Actions because PowerFlow was not available.

> **NOTE:** This application triggers the "Run PowerFlow Application and Remove It from SL1 RBA Queue" application, so you do not need to configure the "Run PowerFlow..." application.

14. If you selected the ***generate_report*** option, you can view the report for "Read SL1 RBA Queue and Retry PowerFlow Applications" and select an application in the report to see if any child applications failed to get triggered.

Creating a Step

On the **Applications** page, you can create new steps using Python that you can add to new or existing PowerFlow applications.

You cannot edit a step in a published SyncPack. If you want to customize a step in a published SyncPack, you need to create a new step using the code from the existing step.

> **NOTE:** All Python step code should be Python 3.7 or later.

To create a step:

1. From the **Applications** page (), click the down arrow () next to the **[Create Application]** button and select *Create Step*. A **Create Step** window appears.

   > **TIP:** You can also create a step in an existing application by navigating to the **Application** detail page for that application. Click the **[Open Editor]** button and Clicking the Create Step icon (+) on the **Steps Registry** pane.

2. In the ***File Name*** field, type a name for the step. The name *cannot* include spaces, and PascalCase, such as "CacheRead", is recommended. Also, the file name must match the *class* name in the Python code.
3. In the ***Edit Step Code*** text box, add or update the Python code for the step as needed, including the metadata information for the new step in the `def __init__(self):` section. For more information, see the "Creating a Step" chapter in the ***PowerFlow for Developers Manual***.
4. To test the new step and view its output, click the **[Run Step]** button. The **[Available Parameters]** and **[Output Data]** will update with data, including error messages, where relevant.
5. When you are done editing the step, click the **[Save]** button. The step is added to the **Steps Registry** pane.

Defining Retry Options for a Step

The following parameters allows you to define multiple retry options for a step. You can specify that the PowerFlow system try to re-run a step if that step fails. Retries work following the rules of exponential backoff: the first retry will have a delay of 1 second, the second retry will have a delay of 2 seconds, the third retry will delay 4 seconds, the fourth retry will delay 8 seconds, and so on.

> **WARNING:** As a best practice, you should only edit the ***retry_max*** parameter and avoid editing any of the other retry parameters. Only advanced users who understand how the retries work and their side effects when they are not set correctly should change the other retry parameters.

You can include the following retry options in the PowerFlow application file, where you define parameters for each step:

- ***retry_max*** . The maximum number of times the PowerFlow system will retry to execute the step before it stops retrying and logs a step failure. For example, if ***retry_max*** is *3*, PowerFlow will retry after 1 second, then 2 seconds, then 4 seconds, and stop if the last retry fails. The default value is 3.
- ***retry_backoff***. Instead of using a defined interval between retries, the PowerFlow system will incrementally increase the interval between retries. Possible values are True or False. The default value is False.
- ***retry_jitter***. Instead of using a defined interval between retries, the PowerFlow system will retry the step execution at random intervals. Possible values are True or False. The default value is False.
- ***retry_backoff_max***. The maximum time interval for the ***retry_backoff*** option, in seconds. For example, This means, if you have *retry_max* set to 15, the delays will be 1, 2, 4, 8, 16, 32, 64, 120, 240, 480, 600, 600, 600, 600, and 600. The default value is 600 seconds.
- ***retry_countdown***. The interval between retries, in seconds. If you enabled ***retry_backup***, the PowerFlow system will incrementally increase this interval. The default value is 180.

> WARNING: Use caution when editing the ***retry_countdown*** option. If you set it to a value smaller than the default of 180 seconds, PowerFlow might experience collisions between task executions, and PowerFlow might stop unexpectedly. If you set this option to a value larger than the default, you might have to wait longer for a task to execute.

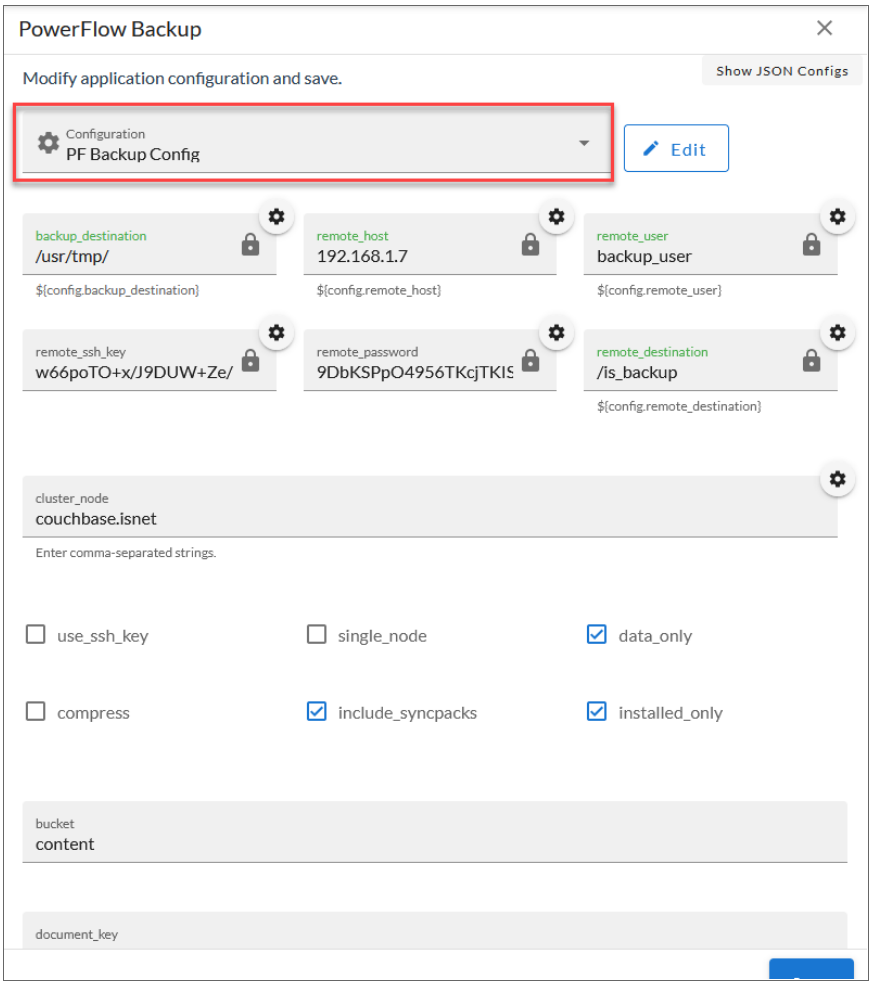
Aligning a Configuration Object with an Application

Before you can run a PowerFlow application, you must align the application with a configuration object from the **Configurations** page. A ***configuration object*** defines global variables, such as endpoints and credentials, that can be used by multiple steps and applications. Each variable in a configuration object is set up as a name and value pair. You can also encrypt a variable to protect sensitive data like a password.

You can "align" the configuration object you want to use with an application from the **Configuration** pane for that application.

To align a configuration object with an application:

1. From the **Applications** page ( ), select the application that you want to align with a configuration object. The **Application** page for that application appears.
2. Click the **[Configure]** button. The **Configuration** pane opens on the right side of the **Application** page. For example:



> **TIP:** To view a pop-up description of a field on the **Configuration** pane for an application, hover over the label name for that field.

3. Select a configuration object from the ***Configuration*** drop-down to "align" to this application. This step is required for all applications.

> **TIP:** You can select *none* from the ***Configuration*** drop-down to clear or "un-align" the selected configuration object from an application. Also, if you did not select a configuration object when editing fields on the **Configuration** pane, the previously set configuration object will remain aligned (if there was a previously set configuration object).

4. If you have a configuration object already selected, you can edit it by clicking the **[Edit Configuration]** button. The **Configuration** pane for that object displays to the left of the application **Configuration** pane. Add or edit the values on that pane and click the **[Save]** button.

5. For any of the application variables with a gear icon (⚙) in the right-hand corner, you can click that icon to open the **Available Configuration Values** pop-up, which contains a list of all available variable values in the configuration object you selected in step 3. Select a value from the list to replace the value in the existing application variable. For more information, see *Working with Application Variables for Configuration Objects*.

> **TIP:** If an application variable is using a value stored in the configuration object, the corresponding code for that variable displays underneath the box for that value, such as `${config.pf_username}`.

6. Click **[Show JSON Configs]** to view the JSON configuration data for the configuration object. Click **[Hide JSON Editor]** again to view the fields instead.
7. As needed, edit the other application variables on the **Configuration** pane.
8. Click **[Save]**. The **Configuration** pane automatically closes.

Running a PowerFlow Application

You can run an application directly from the **Applications** page (the list view) or from an **Application** page (the detail view). If you run the application from the **Application** page, you have the following additional options:

- ***Run***. Executes the application normally, with a log level of 1. This is the default, and it is the same as the *Run Now* option from the **Applications** page.
- ***Debug Run***. Executes the application in Debug Mode with a log level of 10.
- ***Custom Run***. Executes the application using a logging level that you specify (Error, Warning, Info, or Debug). You can also add any customer parameters that you might want to use to test specific features in the application.

> **TIP:** When you run an application, PowerFlow generates a unique task ID for the application and each of its tasks. Using the task IDs, you can poll for the status of the application and the status of each individual running step in the application. For more information, see *Querying for the State of a PowerFlow Application*.

To run an application:

1. From the **Applications** page (), click the **[Actions]** button (⋮) for the application you want to run and select *Run Now*.

> **TIP:** You can also select an application from the **Applications** page and click the **[Run]** button from the **Application** page. If you hover over the **[Run]** button, you can select *Debug Run* or *Custom Run*.

2. As the application runs, the color of the border around each step represents whether it is running, is successful, or has failed:

Step Color	Icon	State
Blue		Running
Green		Successful
Red		Failed
Yellow		Warning

NOTE: Pop-up status messages also appear in the bottom left-hand corner of the **Application** page to update you on the progress of the application run.

TIP: After you start a run, you can click **[Stop]** ( ) next to the **[Run]** button to stop that application and end all running tasks for that application.

3. If a step triggers a child application, a branch icon ( ) appears in the upper left-hand corner of the step. Double-click the branch icon to open the child application. Click the branch icon once to display the triggered application's run ID as a link in a pop-up window. If no run ID is present, the branch icon displays "NONE".

Viewing Previous Runs of an Application with the Timeline

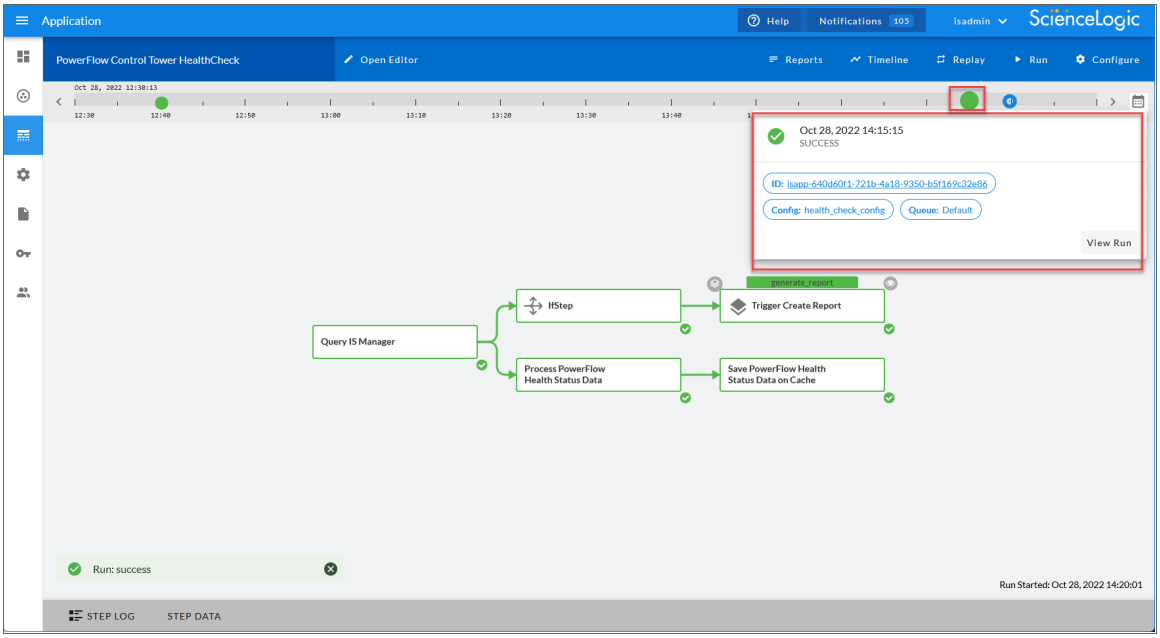
The **Application** detail page for a selected application contains a **[Timeline]** button that displays a history of previous runs of that application.

Also, you can click the **[Replay]** button to replay the last run of an application, such as when an application failed. You can also choose from the following options

- *Info Replay*. Replay the last run the application in Info Mode, which provides more log data in the Step Logs to help you with troubleshooting.
- *Custom Replay*. Replay the last run of the application with custom parameters for testing or troubleshooting.

To view and filter the Timeline:

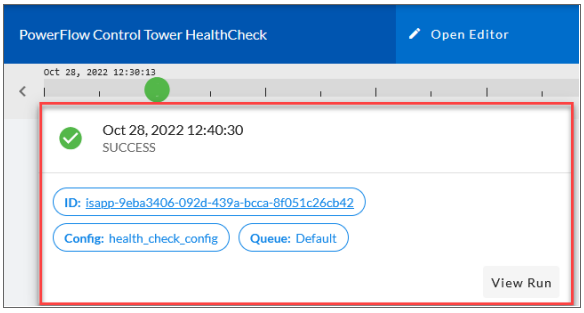
1. From an **Application** detail page, click the **[Timeline]** button. The Timeline displays above the steps for that application:



2. The default view for the Timeline shows the last two hours of runs for that application. The image above shows the last two hours of runs. Use the left arrow icon (❮) and the right arrow icon (❯) to move through the Timeline in 15-minute increments:

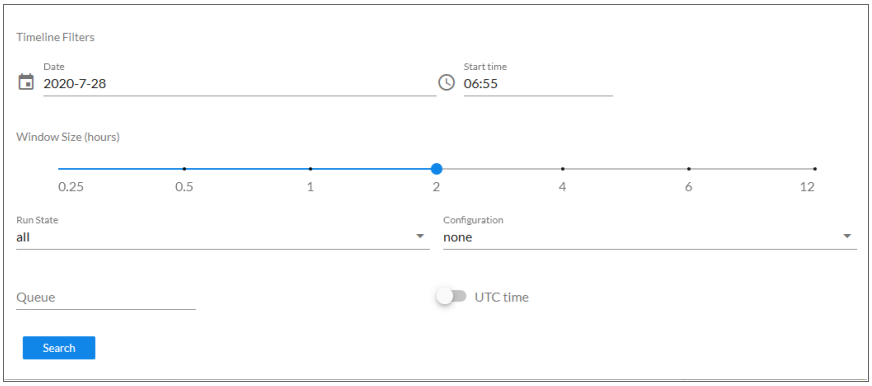
NOTE: The Timeline displays colored dots at a specific time that represent the last time this application was run. A green icon means a run was successful, a blue icon means a run is in progress, and a red icon means a run failed.

3. You can hover over or click an icon for a run on the Timeline to view a pop-up window that displays the run ID, the configuration object and the queue used for that run:



> **TIP:** Click the link for the run ID or click **View Run** to open the **Application** page for that specific run of the application. The run ID also displays in the **Step Log** pane for the "triggering" step for that application, and it also appears at the end of the URL for that application. On that page, you can select a step and open the **Step Log** to view any issues.

4. Click the Filter icon ( ) to filter or search the list of previous runs for this application. A **Filter** window appears:



5. Edit the following fields on the **Filter** window as needed:
   - ***Date***. The date for the history of previous runs you want to view. Click the field to open a pop-up calendar.
   - ***Start Time***. The starting time for the history, using local time instead of UTC time. Click the field to open a pop-up time selector.
   - ***Window Size***. The length of the history, in hours. The default history view for the Timeline is two hours.
   - ***Run State***. Select the type of previous runs you want to view. Your options include *all*, *success*, *failure*, and *pending*. The default is *all*.
   - ***Configuration***. Select a configuration file to filter for application runs using that configuration only.

- ***Queue***. Type a queue name to filter for application runs that use that queue.
- ***UTC Time***. Select UTC if you do not want to use local time. The Schedule feature uses UTC time.

6. Click **[Search]** to run the filter or search.

> **TIP:** If the Timeline is open and you want to close it, click the **[Timeline]** button.

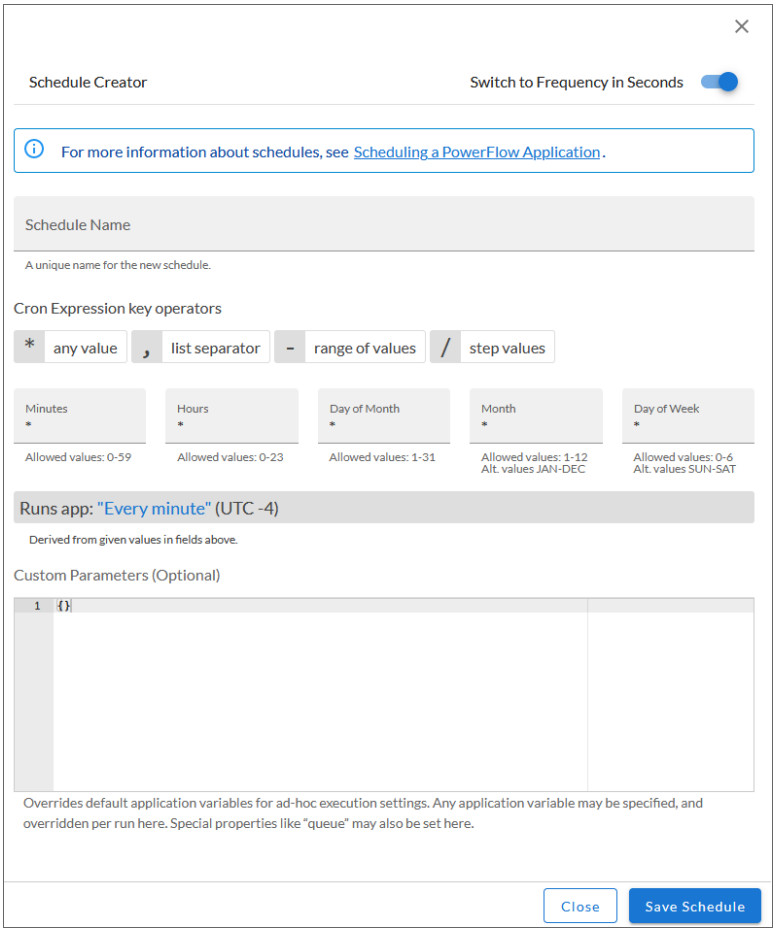
Scheduling a PowerFlow Application

You can create one or more schedules for a single application in the PowerFlow user interface. When creating each schedule, you can specify the queue and the configuration file for that application.

To schedule an application:

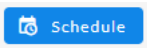
1. On the **Applications** page ( ), click the **[Schedule]** button for the application you want to schedule. The **Scheduler** window appears.
2. In the **Schedule List** pane, click the down arrow icon () next to an existing schedule to view the details for that schedule.
3. In the **Schedule Creator** pane, complete the following fields for the default ***Frequency*** setting:
   - ***Schedule Name***. Type a name for the schedule.
   - ***Frequency in seconds***. Type the number of seconds per interval that you want to run the application.
   - ***Custom Parameters***. Type any JSON parameters you want to use for this schedule, such as information about a configuration file or mappings.

4. To use a cron expression, click the ***Switch to Cron Expression*** toggle to turn it blue. If you select this option, you can create complicated schedules based on minutes, hours, the day of the month, the month, and the day of the week:



As you update the cron expression, the **Schedule** window displays the results of the expression in more readable language, such as *Runs app: "Every 0 and 30th minute past every hour on Sat"*, based on 0,30 in the ***Minutes*** field and 6 in the ***Day of Week*** field.

5. Click **[Save Schedule]**. The schedule is added to the **Schedule List** pane. Also, on the **Applications** page, the Schedule button now displays with a dark blue background:



NOTE: After you create a schedule, it continues to run until you delete it. Also, you cannot edit an existing schedule, but you can delete it and create a similar schedule if needed.

To view or delete an existing schedule:

1. On the **Applications** page, click the **[Schedule]** button for the application that contains a schedule you want to delete. The **Scheduler** window appears.

2. Click the down arrow icon ( ) to view the details of an existing schedule.
3. To delete the selected schedule, click the Actions icon ( ) and select **[Delete]**.

> **TIP:** On the **Scheduler** window for a PowerFlow application, you can click the **[Copy as]** button from the **Schedule List** pane to make a copy of an existing schedule.

Backing up and Restoring PowerFlow Data

You can use PowerFlow to back up and recover data in the Couchbase database.

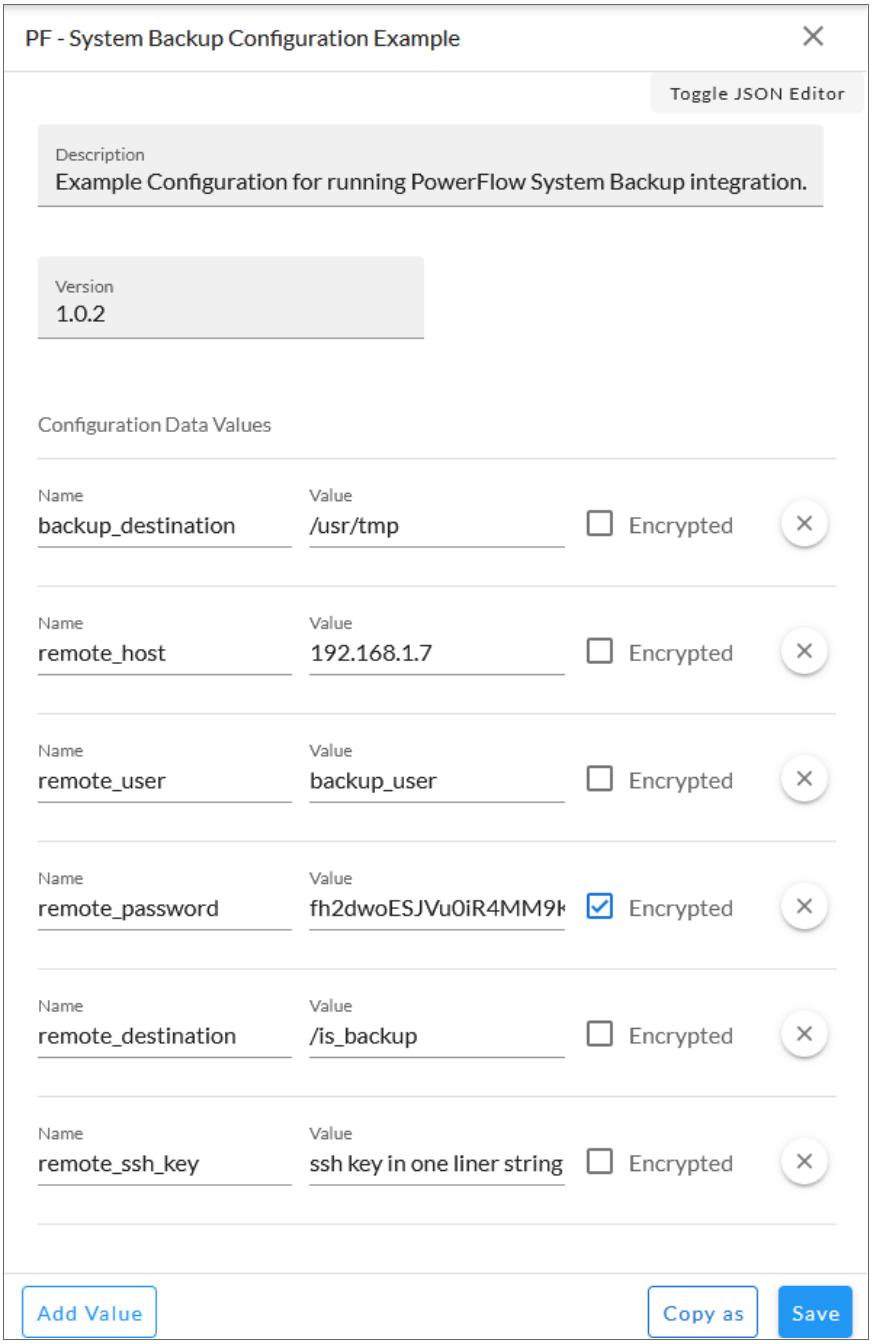
This option uses the "Backup" application in the PowerFlow user interface to create a backup file and send that file using secure copy protocol (SCP) to a destination system. You can then use the "Restore" application to get a backup file from the remote system and restore its content.

> **NOTE:** The backup and restore applications are application-level backup and restore tools. For full-system backups, you will need to do a filesystem-level backup to ensure that you get the encryption key that was used to encrypt configuration objects as well as other files used to describe the environment, including the **/etc/iservices** directory, the **docker-compose.yml** file and the **docker-compose-override.yml** file.

Creating a Backup

To create a backup:

1. To add the relevant configuration information, go to the **Configurations** page (⚙) and click the **[Edit]** button or click the Actions button (⋮) and select *Edit* for the "PF - System Backup Configuration Example" configuration object. The **Configuration** pane appears:



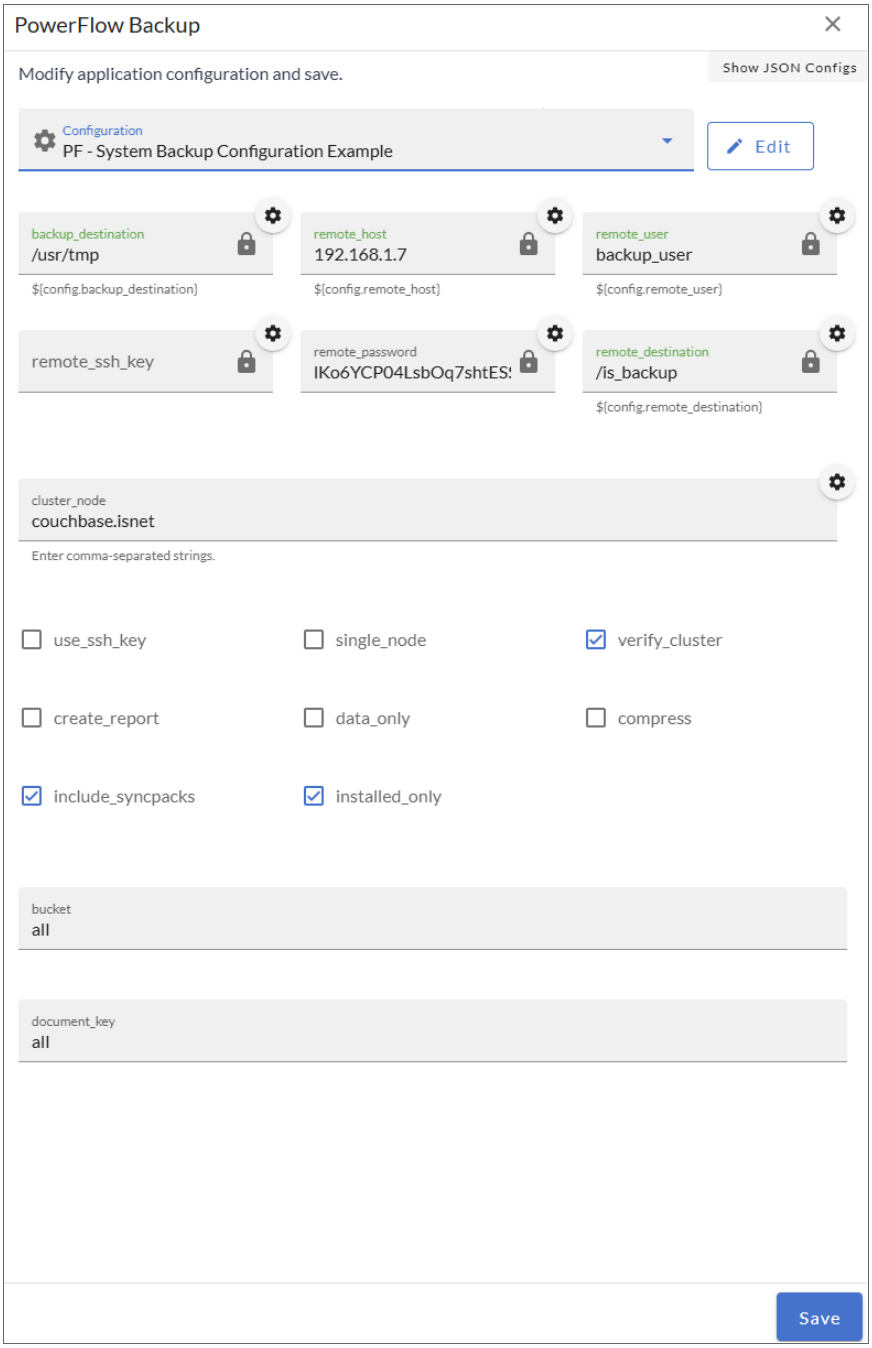
2. Click the **[Copy as]** button and provide values for the following fields in the **Create Configuration** pane:
    - ***Friendly Name***. Name of the configuration object that will display in the user interface.
    - ***Description***. A brief description of the configuration object.
    - ***Author***. User or organization that created the configuration object.
    - ***Version***. Version of the configuration object.
    - ***backup_destination***. The location where the backup file is created inside the container. The default is **/usr/tmp**. This does not need to be changed.
    - ***remote_host***. The hostname for the remote location where you want to send the backup file via SCP from the *backup_destination* location.
    - ***remote_user***. The user login for the remote location.
    - ***remote_password***. The user password for the remote location. Encrypt this value.
    - ***remote_destination***. The remote location where the application will send the backup file. The directory defined here needs to exist in the remote system.
    - ***remote_ssh_key***. The remote SSH key you want to use in place of a password for the remote location. Use the newline character `\n` as a separator. If the SSH key needs a paraphrase to be decrypted, set the paraphrase by creating a ***remote_password*** variable in the configuration object aligned with the applications.

> **NOTE:** You will need to edit the SSH Key values in the JSON Editor for this release to ensure the key is properly set. For example: `"{config.ssh_key}"`. This is a known issue that will be addressed in a future release.

> **TIP:** To get a one-line string of the SSH key, run the following command:
> `sed -E ':a;N;$!ba;s/\r{0,1}\n/\\n/g' ~/.ssh/id_rsa`

3. Click the **[Save]** button.
4. Go to the **Applications** page and select the "PowerFlow Backup" application.

5. Click the **[Configure]** button. The **Configuration** pane appears:



6. On the **Configuration** pane, provide values for the following fields:

- ***Configuration***. Select the configuration object you created in steps 1-3.
- ***cluster_node***. Specify the node from which you want to make the backup (for a single-node backup only). You can specify the order of the nodes that you are backing up by listing each node in order of preference, separated by commas. If one of the nodes fails or is unavailable, the backup application continues down the list of nodes in the order you specified in this field. For example: *couchbase-worker.isnet,couchbase-worker.isnet2,couchbase.isnet*. To use the ordered backup option, you must also check the ***single_node*** field.
- ***use_ssh_key***. Select this option if you want to run the backup application using an SSH key for authentication instead of using a password. You will need to provide a ***remote_ssh_key*** value in the configuration object you aligned with this application, such as `config.remote_ssh_key`.
- ***single_node***. Select this option only if you want to back up from a single node in the cluster. Specify the node in the ***cluster_node*** field. You should also select this option if you want to specify the order of the nodes that you are backing up, and list each node in order of preference in the ***cluster_node*** field.
- ***verify_cluster***. Use to verify Couchbase Cluster Health Status.

IMPORTANT: The ***verify_cluster*** option must not be disabled, otherwise incomplete backups can be created in cluster environments if the cluster is unhealthy.

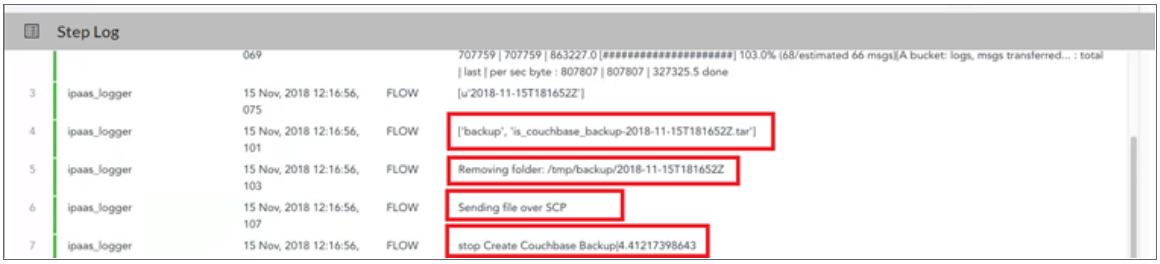
- ***create_report***. Create a report of the content bucket document IDs that were backed up. This only works if the ***verify_cluster*** option is enabled and the cluster is healthy.
- ***data_only***. Select this option if you only want to restore bucket data.
- ***compress***. Select this option to compress backups using Gzip.
- ***include_syncpacks***. Select this option if you want to back up the SyncPacks on the PowerFlow system.
- ***installed_only***. Select this option if you want to back up only SyncPacks that have been installed. If you do not select this option, the application will also back up SyncPacks that have been uploaded to the PowerFlow system.
- ***bucket***. Select which bucket in Couchbase you want to back up. Your options include:
  - *all*. Back up all buckets.
  - *content*. Back up only buckets that have content in them.
  - *logs*. Back up only the logs.

TIP: To keep backups as small as possible, ScienceLogic recommends selecting only ***data_only*** and ***bucket*** > *content*. These settings will keep backups to <5mb each, and contains all essential data (configuration objects, applications, and steps).

- ***document_key***: Select whether you want to back up all record or CI and device cache records.

> **NOTE:** The options you select affect the name of the backup file that this application generates. For example, **is_couchbase_backup-data_only-2019-04-01T185527Z.tar** is a uncompressed data only backup, while **is_couchbase_backup-data_only-cache-logs-couchbase.isnet-2019-04-01T185937Z.tar.gz** is a compressed data-only backup of the CI and device cache from the *couchbase.isnet* node in a cluster.

7. Click the **[Save]** button. The **Configuration** pane automatically closes.
8. On the **Application** detail page, click the **[Run Now]** button. When the application completes, a file named "is_couchbase_backup-<*date*>.tar" is added to the remote server in the specified remote backup destination.
9. To ensure that the backup was created, select the "Create IS Backup" step and open the ***Step Log*** section.
10. Look for entries related to backup and make a note of the of the backup file name, which you will need when you run the "PowerFlow Restore" application:



> **TIP:** You can schedule the "PowerFlow Backup" application to run on a regular basis, or you can run the application as needed. To schedule the application, click the **[Schedule]** button for the "PowerFlow Backup" application on the **Applications** page. For more information, see *Scheduling a PowerFlow Application*.

Restoring a Backup

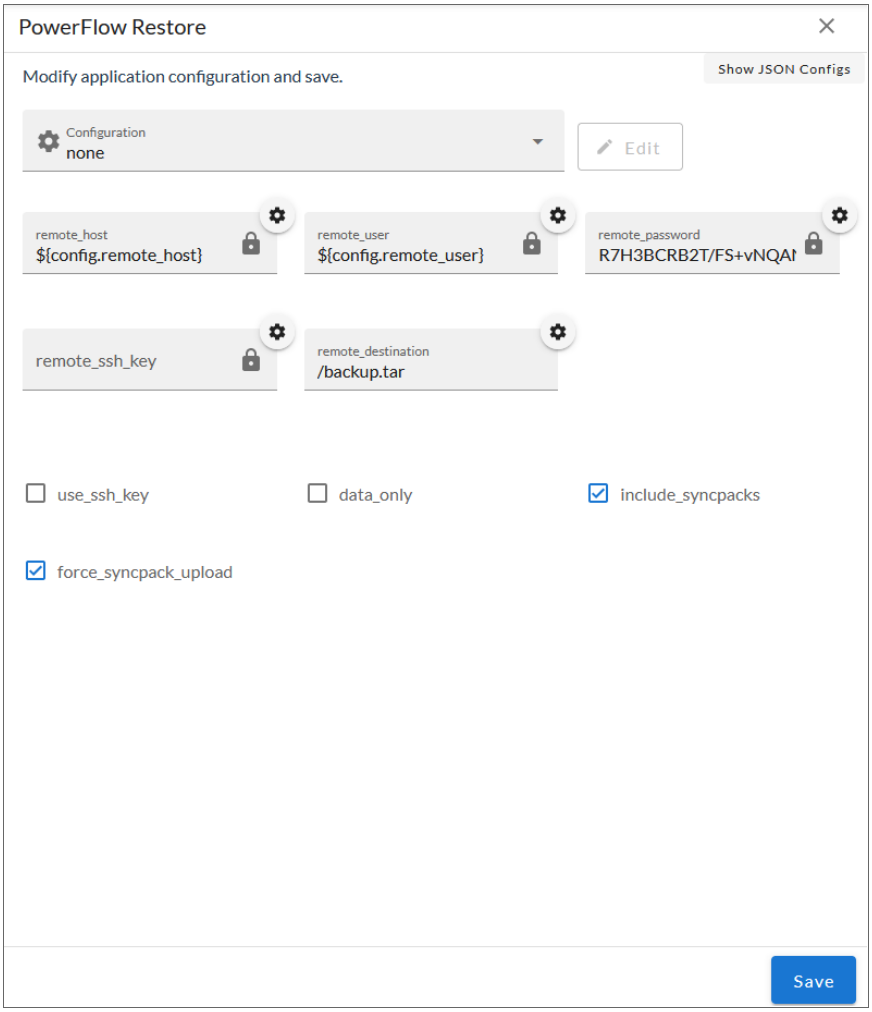
After you have created a backup using the "PowerFlow Backup" application in the PowerFlow user interface, you can use the "PowerFlow Restore" application to restore that file.

> **NOTE:** Do not restore the PowerFlow backup to a system that uses a different encryption key.

To restore a backup:

1. In the PowerFlow user interface, go to the **Applications** page and select the "PowerFlow Restore" application. The **Application** page appears.

2. Click **[Configure]**. The **Configuration** pane appears:



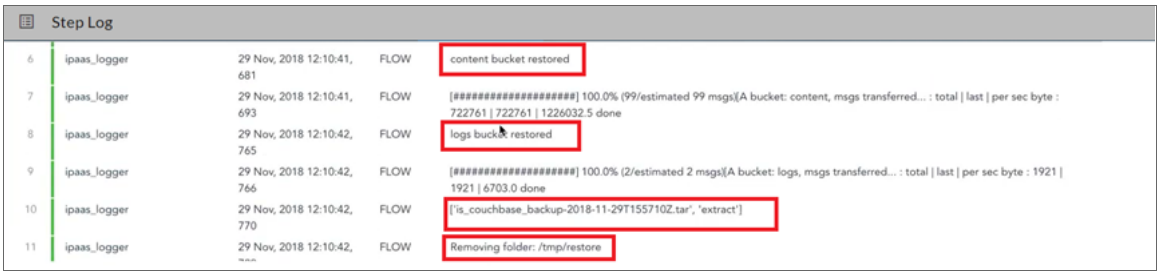
3. In the **Configuration** pane, provide values for the following fields:
    - ***Configuration***. Select the same configuration object you aligned with the "PowerFlow Backup" application. Required.
    - ***use_ssh_key***. Select this option if you ran the backup application using an SSH key for authentication instead of using a password. You will need to provide a ***remote_ssh_key*** value in the configuration object you aligned with this application.

      NOTE: You will need to edit the SSH Key values in the JSON Editor for this release to ensure the key is properly set. This is a known issue that will be addressed in a future release.

    - ***data_only***. Select this option if you only want to restore bucket data.

- ***include_syncpacks***. Select this option if you want to restore the SyncPacks you backed up with the "PowerFlow Backup" application.
- ***force_syncpack_upload***. Select this option if you want to force upload SyncPacks if the files already exist in the PowerFlow system.

4. Click **[Save]**. The **Configuration** pane automatically closes.
5. On the **Application** detail page, click **[Run]**.
6. To ensure that the backup was restored, click to open the **Step Log** section and look for entries related to restoring the backup:



After running the "PowerFlow Restore" application, the "PowerFlow Backup" application might display as "Run status pending". This issue occurs because at the time of the last backup from Couchbase, the logs for the "PowerFlow Backup" application showed a pending state. This message is addressed during the next run, and it does not cause any issues with the backup or restore processes.

In addtion, after restoring a backup that contains installed SyncPacks, the **SyncPacks** page in the PowerFlow user interface will show that the versions of the restored SyncPacks were installed successfully, but their virtual environments might not be in place. This issue will be addressed in a future release of the PowerFlow Platform.

To address this issue, perform *one* of the following actions to make sure that the environments for the SyncPacks are recreated successfully:

- Force the **syncpacks_steprunner** service to restart, using the following command:

```
docker service update --force syncpacks_steprunner
```

- Execute the following **powerflowcontrol** (pfctl) cluster or node action:

```
reinstall_syncpack_venv
```

- Reinstall the SyncPacks in place using the PowerFlow user interface.

Restoring a Backup using the Command-line Interface

To restore from a backup using the command-line interface:

1. Copy the backup for the system to the **/tmp** folder.
2. Extract the backup from the archive with `tar -xvf` or `tar -xzvf`, based on whether the file uses **gz**.
3. Copy the backup directory to **/var/data/couchbase/backup**.

4. Exec into the Couchbase container and run the following command:

```
cbrestore /opt/couchbase/var/backup/couchbase -b content
http://couchbase.isnet:8091 -u isadmin -p
<password>http://couchbase.isnet:8091 -u isadmin -p <password>
```

5. After the backup is restored, remove those files from **/tmp** and **/var/data/couchbase/backup** to free up that space again.

Chapter 6

Managing Configuration Objects

Overview

This chapter describes how to use the **Configurations** page ( ) of the PowerFlow user interface to create or use a ***configuration object*** that contains a set of variables that all steps and applications can use.

This chapter covers the following topics:

What is a Configuration Object? *180*

Viewing the List of Configuration Objects *180*

Creating a Configuration Object *182*

Working with Application Variables for Configuration Objects *184*

Editing a Configuration Object *187*

Downloading and Importing a Configuration Object *187*

What is a Configuration Object?

You can create and edit configuration objects on the **Configurations** page ( ) of the PowerFlow user interface. After you create the configuration object, it appears in the ***Configuration*** drop-down on the **Configuration** pane of an **Application** detail page.

A ***configuration object*** is a stand-alone JSON file that lives on the PowerFlow system. A configuration object supplies the login credentials and other global variables that can be used by all steps and applications in PowerFlow. Configuration objects allow the same application to be deployed in multiple PowerFlow instances, with different credentials and variables.

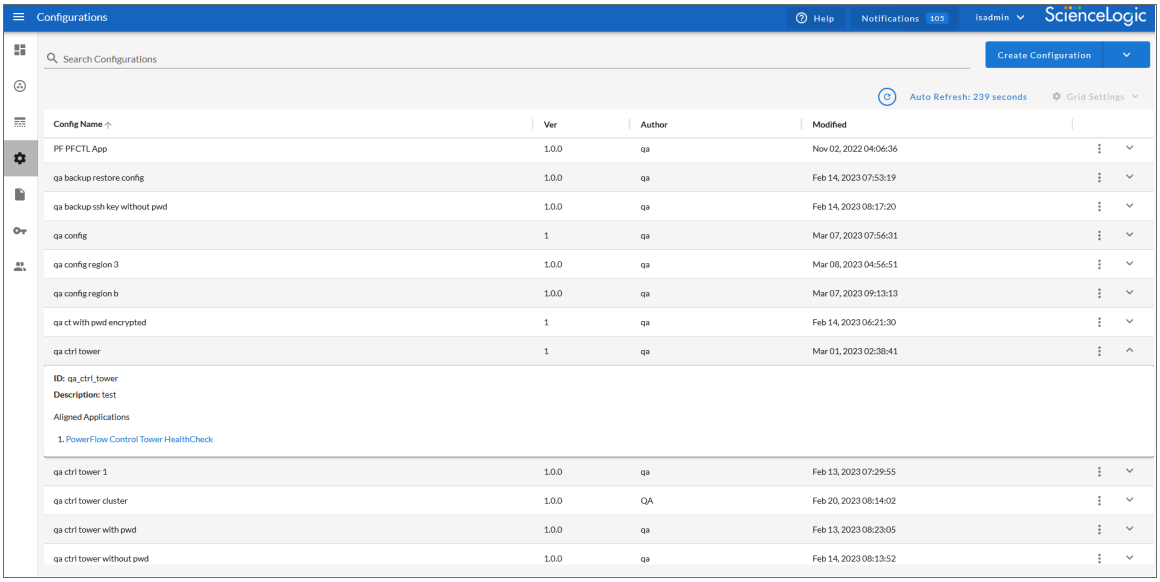
Configuration objects can map variables from SL1 to a third-party platform. For instance, SL1 has device classes, while various third-party platforms like ServiceNow have CI classes; the configuration object would map these two variables.

Before you can run a PowerFlow application, you must select a configuration object and "align" that configuration object with the application.

TIP: You can select *none* from the ***Configuration*** drop-down to clear or "un-align" the selected configuration object from an application. Also, if you did not select a configuration object when editing fields on the **Configuration** pane, the previously set configuration object will remain aligned (if there was a previously set configuration object).

Viewing the List of Configuration Objects

The **Configurations** page ( ) displays a list of available configuration objects. From this page you can create and edit configuration objects:



TIP: You can search for a specific configuration object by typing the name of that configuration in the ***Search*** field at the top of the **Configurations** page. The user interface filters the list as you type. You can also sort by clicking the **Configuration Object Name** column header.

The **Configurations** page displays the following information about each configuration object:

- ***Config Name***. The name of the configuration object.
- ***Ver***. The version number of the configuration object.
- ***Author***. The user or organization that created the configuration object.
- ***Modified***. The date and time that the configuration object was last edited.
- ***Actions***. Contains the following buttons:
    - **[Edit]**. Opens a pane where you can edit the contents of the configuration object.
    - **[Download]**. Lets you download the JSON file for the configuration object. If you have multiple PowerFlow systems, you can download a configuration object from one system and then use the *Import Configuration* option (under the **[Create Configuration]** button) to upload that object to the second PowerFlow system.
    - **[Delete]**. Removed the configuration object from the PowerFlow system. Any PowerFlow applications that were aligned with that configuration object are no longer aligned with that object.

TIP: Click the down arrow icon (⌄) for a configuration object to view a section that lists the applications that are currently aligned with that configuration object. Click the up arrow icon (⌃) to close the section.

Creating a Configuration Object

When creating or editing a configuration object on the **Configurations** page, the name-value pairs in the **Configuration Data** section display in fields by default instead of a block of JSON code.

For more complex configuration objects, you can click **[Toggle JSON Editor]** to switch between text fields and JSON code for the configuration data. In the **Configuration Data** section, you can press **[Ctrl+F]** to search for code. Also, a red warning icon ( ) appears in the first column of the **Configuration Data** section for a row where the JSON is not valid.

If you set 'null' as a value for a JSON parameter in a configuration object, the value reverts to its default value, if a default value exists. You can clear JSON parameters by specifying an empty dictionary {} as the value. Also, upgrading or downgrading a SyncPack will not overwrite a user configuration.

> **TIP:** Many SyncPacks for PowerFlow contain an "example" configuration object that you can use as a template. You should do a ***Save As*** with the example configuration objects so you can make a copy of the example configuration that you can customize for your specific PowerFlow system. Do *not* use the example configuration objects to run PowerFlow applications.

To create a new configuration object:

1. From the **Configurations** page ( ), click **[Create Configuration]**. The **Create Configuration** pane appears.

   > **TIP:** Instead of creating a completely new configuration object, you can also edit an existing configuration object that has some of the configuration data that you want to use and click **[Copy as]** from the **Configuration** pane to create a copy of that configuration object.

2. Complete the following fields:
   - ***Friendly Name***. Name of the configuration object that will display in the user interface.
   - ***Description***. A brief description of the configuration object.
   - ***Author***. User or organization that created the configuration object.
   - ***Version***. Version of the configuration object.
   - ***Configuration Data Values***. To add configuration values in the form of name-value pairs, click **[Add Value]**. Complete the ***Name*** and ***Value*** fields, and select ***Encrypted*** if needed.

     > **TIP:** Click **[Toggle JSON Editor]** to view the JSON configuration data for the configuration object at the bottom of the pane. Click **[Toggle JSON Editor]** again to return to the original view.

3. In the **Configuration Data** section, include the required block of code to ensure that the applications aligned to this configuration object do not fail:

```
{
  "encrypted": false,
  "name": "sl1_db_host",
  "value": "${config.sl1_host}"
},
```

For example:

```
{
  "encrypted": false,
  "name": "sl1_db_host",
  "value": "10.2.11.42"
},
```

TIP: If you are using IPv6 for IP addresses, wrap the IP string in brackets, such as **https://[2001:db8:3333:4444:5555:6666:7777:8888]**

TIP: Click **[Toggle JSON Editor]** to show the JSON code. Click the button again to see the fields. You can also click **[Add Value]** and add a new name-value pair in the **Configuration Data Values** section.

NOTE: If you are using SL1 with an External Database (SL1 Extended architecture or a cloud-based architecture), update the `"value"` of that block of code to be the host of your database. This field accepts IP addresses. For example: `"value": "db.sciencelogic.com"`. If you are *not* using the SL1 Extended architecture or a cloud-based architecture, you do not need to make any changes to the block of code other than pasting the code into the configuration object.

6. To create a configuration variable, define the following keys:
   - ***encrypted***. Specifies whether the value will appear in plain text or encrypted in this JSON file. If you set this to "true", when the value is uploaded, PowerFlow encrypts the value of the variable. The plain text value cannot be retrieved again by an end user. The encryption key is unique to each PowerFlow system. The value is followed by a comma.
   - ***name***. Specifies the name of the configuration file, without the JSON suffix. This value appears in the user interface. The value is surrounded by double-quotes and followed by a comma.
   - ***value***. Specifies the value to assign to the variable. The value is surrounded by double-quotes and followed by a comma.
7. Click **[Save]** to save the new configuration object, or click **[Copy as]** to save the configuration object as a new configuration object with a different name.

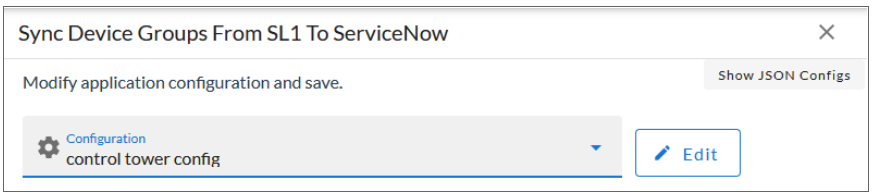
8. Align this configuration object with the PowerFlow applications that you want to run by clicking the **Configure** button from the detail page for each application and selecting this configuration object from the ***Configuration*** drop-down.

Working with Application Variables for Configuration Objects

Starting with version 2.5.0 of the PowerFlow Platform, you can perform a number of new processes with the application variables for configuration objects and the **Configuration** pane for PowerFlow applications. This section summarizes these new features.

Edit Configuration Button

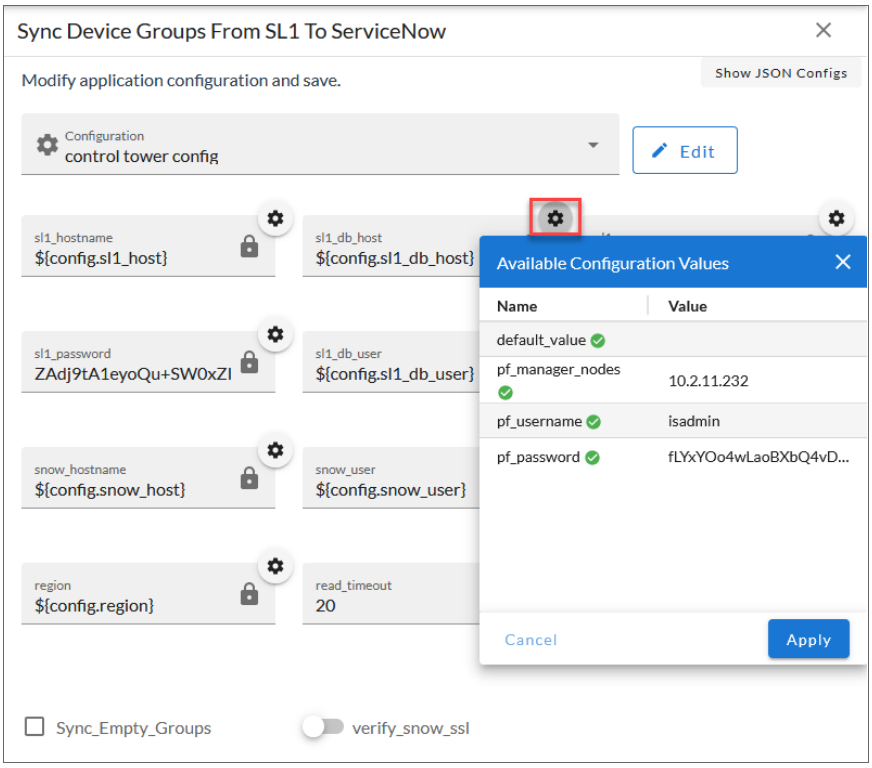
If you have a configuration object already selected on the **Configuration** pane for a PowerFlow application, you can edit it by clicking the **[Edit]** button:



The **Configuration** pane for that configuration object displays to the left of the application **Configuration** pane. Add or edit the values on that new pane and click the **[Save]** button.

Available Configuration Values Pop-up

For any of the application variables with a gear icon (⚙) in the right-hand corner, you can click that icon to open the **Available Configuration Values** pop-up:



This pop-up contains a list of all of the available configuration variable values in the configuration object that you selected in the ***Configurations*** field.

In the **Available Configuration Values** pop-up, you can:

- Select a value from the list to replace the value in the existing application variable.

> **TIP:** If an application variable is using a value stored in the configuration object, the corresponding code for that variable displays underneath the box for that value, such as `${config.pf_username}`.

- Click the **[Set to Default]** button to set a changed application variable value back to its original value.
- Click the **[Clear]** button to clear the selected value.

Promote Step Variable Option

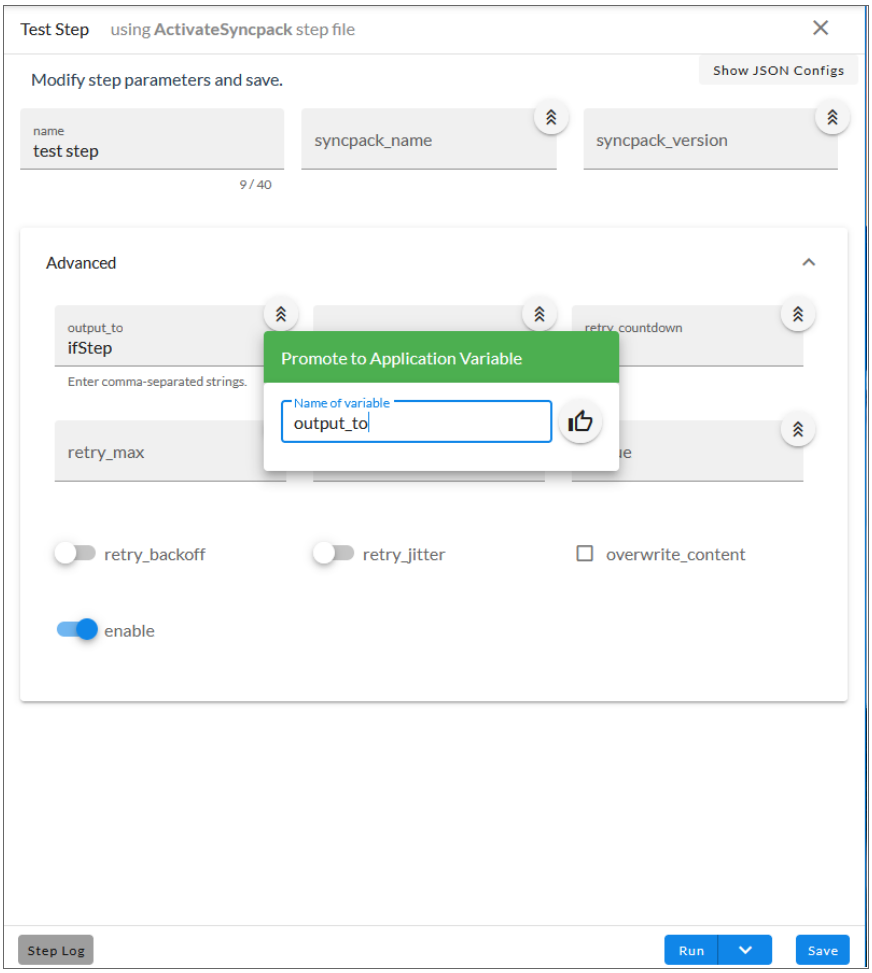
You can use the "Promote to Application Variable" icon ( ) on the **Configuration** pane for a step variable to make that value available in a configuration object. This is also called "promoting" a step variable.

> **IMPORTANT:** The step promotion option is only available for steps that are not part of a ScienceLogicSyncPack.

Previously you could add a step variable to a configuration object, but you would need to type the encoded variable name, such as `${appvar.snow_hostname}`, to promote that step variable to an application variable.

To promote a step variable to a configuration variable:

1. In the PowerFlow user interface, open the application that you want to update and click the **[Open Editor]** button. The **Steps Registry** of the PowerFlow builder appears.
2. Select the step that has the variable you want to promote and drag it to the main pane or canvas. The **Configuration** pane for that step appears automatically:



3. Click the "Promote to Application Variable" icon ( ) for the step variable you want to promote. A **Promote to Application Variable** modal appears.
4. Use the existing name of the variable, or type a new name, and then click the thumbs-up icon ( ).
5. Click the **[Save]** button for the step, and then click the **[Save]** button for the application. This step variable is now available on the **Configuration** pane for applications.

Editing a Configuration Object

To edit an existing configuration object:

1. In the PowerFlow user interface, go to the **Configurations** page, click the Actions icon (⋮), and select *Edit*for the configuration object you want to edit. The **Configuration** pane appears for that object.
2. Edit the values in the following fields as needed:
    - ***Description***. A brief description of the configuration object.
    - ***Version***. Version of the configuration object.
    - ***Configuration Data Values***. To add configuration values in the form of name-value pairs, click **[Add Value]**. Complete the ***Name*** and ***Value*** fields, and select ***Encrypted*** if needed.

    > TIP: Click **[Toggle JSON Editor]** to view the JSON configuration data for the configuration object at the bottom of the pane. Click **[Toggle JSON Editor]** again to view the ***Configuration Data Values*** section with the **[Add Value]** button instead.

4. If you want to make a copy of this configuration object, click **[Copy As]** and update the relevant fields in the **Create Configuration** pane.
5. Click **[Save]** to save your changes.

Downloading and Importing a Configuration Object

If you have multiple PowerFlow systems that use similar environments, you can download the JSON file for a configuration object from one PowerFlow system and then upload that object to the second PowerFlow system.

To download and import a configuration object:

1. In the PowerFlow user interface, go to the **Configurations** page and click the **[Download]** button for the configuration object you want to download.
2. Save the file to a local computer.
3. Log in to the second PowerFlow system and go to the **Configurations** page.
4. Click the down arrow icon (⌄) next to the **[Create Configuration]** button and select *Import Configuration*. The **Import Configuration** page appears.
5. Click the **[Browse]** button and locate the file you saved in step 2, or drag the file onto the **Import Configuration** page.
6. Click the **[Import]** button. The configuration object is added to the **Configurations** page.

Chapter 7

Generating and Viewing Reports for SL1 PowerFlow Applications

Overview

This chapter describes how to generate reports for PowerFlow applications, which you can view on the **Reports** page ( ) of the PowerFlow user interface.

This chapter also lists the reports available in the various SyncPacks that can be used in the PowerFlow user interface.

> **NOTE:** PowerFlow does not have a retention policy for reports. PowerFlow reports will remain until you manually remove them.

This chapter covers the following topics:

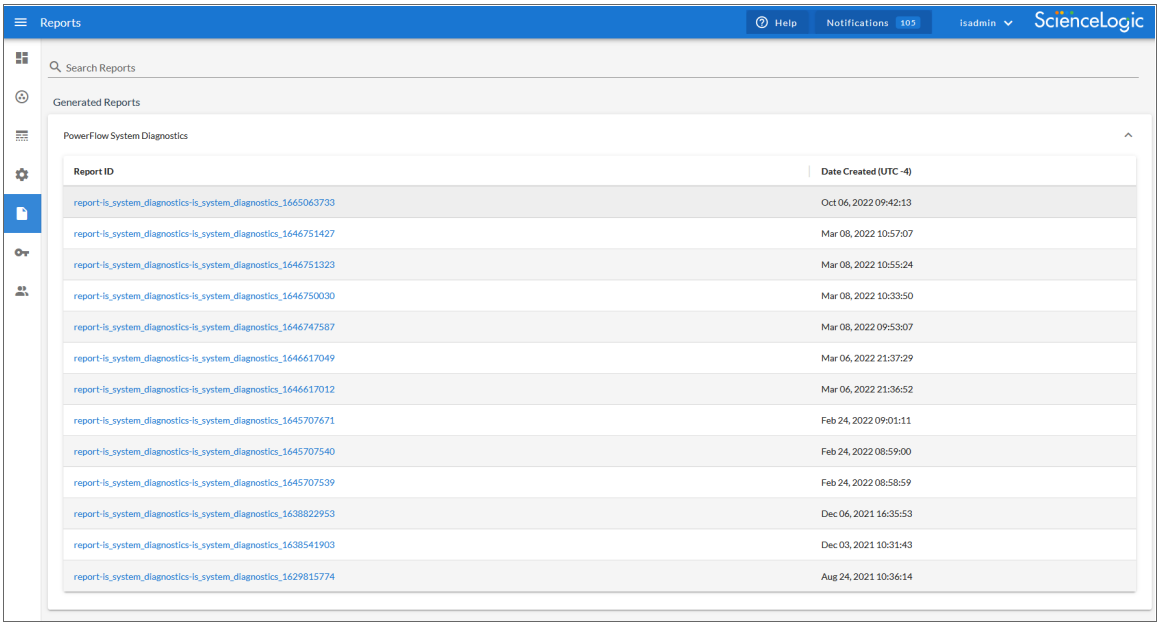
Viewing the List of Reports in PowerFlow 189

PowerFlow Platform Reports 191

SyncPack Reports 194

Viewing the List of Reports in PowerFlow

The **Reports** page ( ) contains a list of reports associated with PowerFlow applications. You can search for a specific report by typing the name of that report in the ***Search*** field at the top of the **Reports** page. The user interface filters the list as you type.



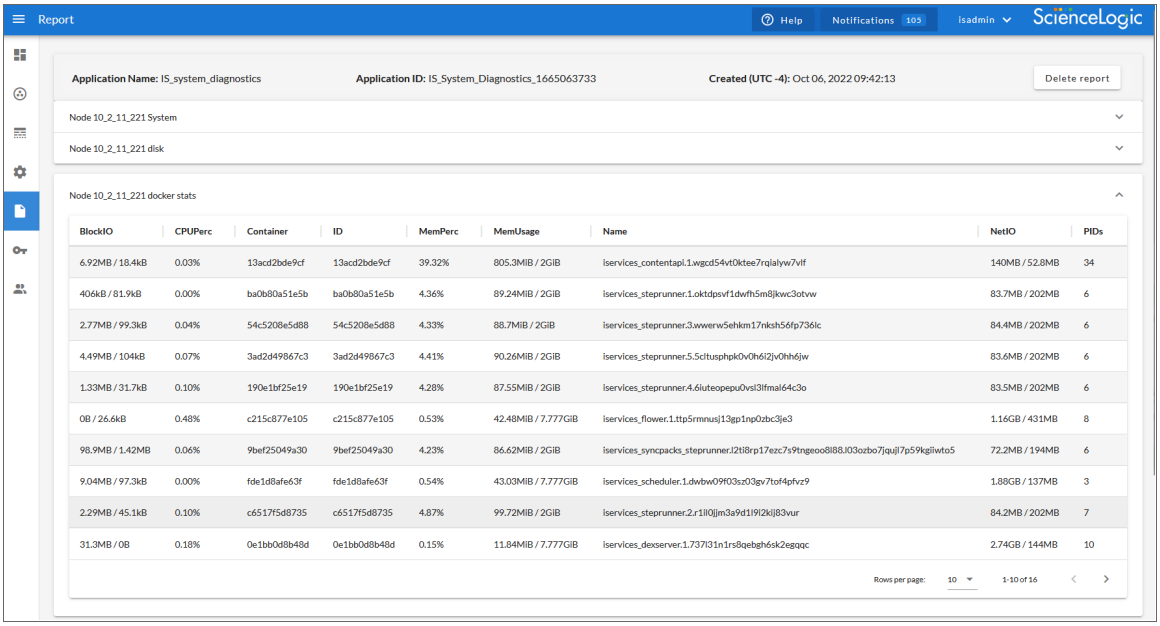
If a PowerFlow application supports reports and the reporting feature is enabled, PowerFlow will generate a report each time you run the application. Each report displays data only from the most recent run of the application; a report is not an aggregation of all previous runs.

TIP: From the detail page for a PowerFlow application, click the **[Reports]** button to go to the **Reports** page.

To view a report:

1. On the **Reports** page ( ), click the name of the application or the down arrow ( ) to expand the list of reports for that application.

2. Click a report name in the **Report ID** column. The **Report Details** page appears.



3. Click the down arrow icon ( ) to open a new section where you can view more information about an item in a report, and click the up arrow icon ( ) to close that section.
4. To download the report, click **[Download]**.
5. To delete the report, click **[Delete]**.

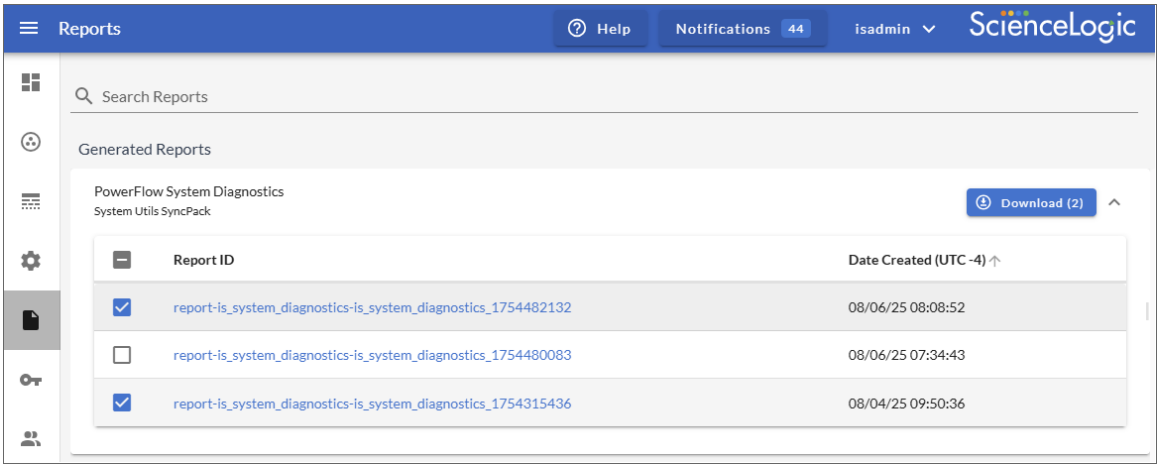
Bulk Downloading Reports in PowerFlow

You can download any number of reports from the **Reports** page ( ). You can select a single report instance or multiple report instances for the same application to download at the same time.

To bulk download multiple report instances:

1. On the **Reports** page ( ), locate the applications from which you want to download a report.
2. Select the checkbox at the top of an application to select all report instances of an application, or select

the checkbox next to each report instance you want to download.



3. Click the **[Download]** button. The **Confirm Bulk Download** modal appears.

> **NOTE:** The number of selected reports displays inside the **[Download]** button and inside the **Confirm Bulk Download** modal.

4. Click **[Yes]** to download the selected report instances.

PowerFlow Platform Reports

The PowerFlow Platform includes the following reports by default:

- *PowerFlow System Diagnostics Report*
- *Read SL1 RBA Queue and Retry PowerFlow Applications Report*

The PowerFlow System Diagnostics Report

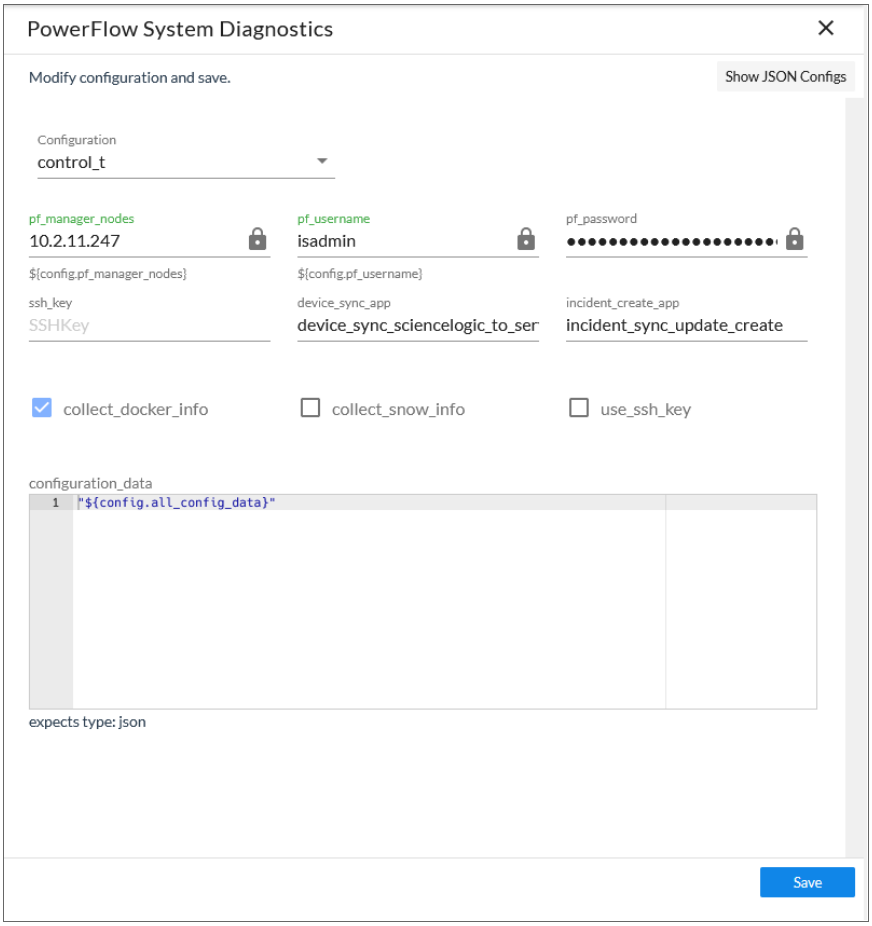
The "PowerFlow System Diagnostics" application lets you view platform diagnostics for the PowerFlow system. You can use the information displayed in these diagnostics to help you troubleshoot issues with the different tools used by PowerFlow.

> **NOTE:** Older versions of this application were named "Integration Service System Diagnostics".

To view the diagnostics report:

1. From the **Applications** page, select the "PowerFlow System Diagnostics" application. The **Application** page appears.

2. Click **[Configure]**. The **Configuration** pane appears:

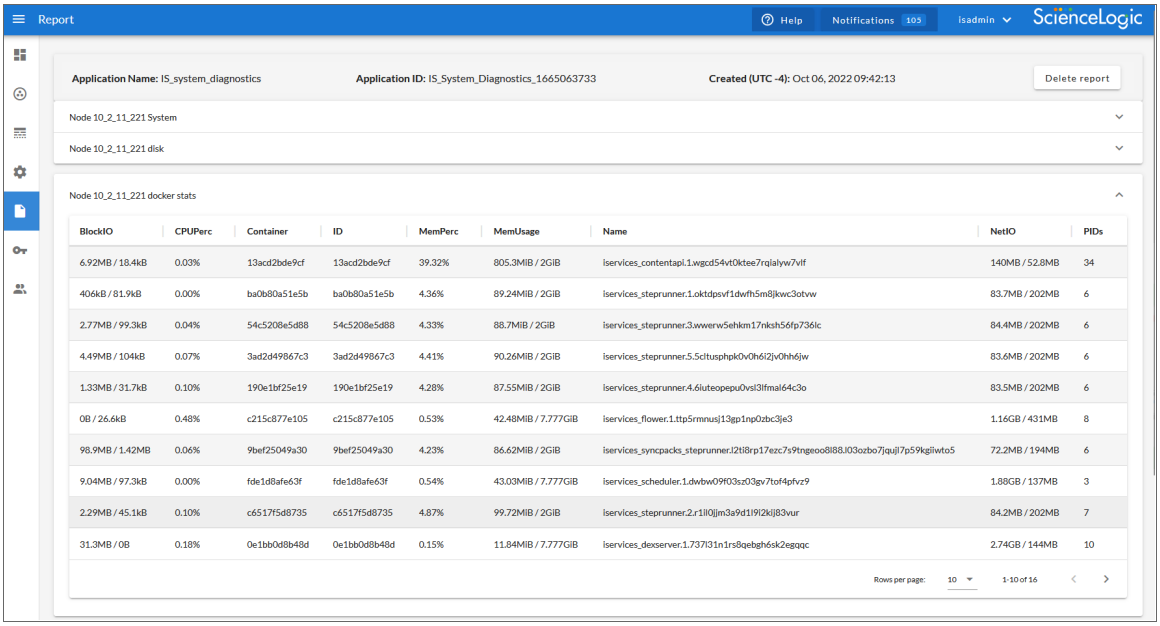


3. Complete the following fields:

   - ***Configuration***. Select a configuration to align with this application. The "IS - System Diagnostic Configuration Example" configuration object contains the structure needed for this application, and you can use that configuration object as a template. Be sure to update the configuration object with values for ***is_swarm_manager***, ***is_username***, and ***is_password***.
   - ***ssh_key***. The remote SSH key you want to use in place of a password for the remote location. You will need to edit the SSH Key values in the JSON Editor for this release to ensure the key is properly set. This is a known issue that will be addressed in a future release.
   - ***device_sync_app***. Specify the name of the Incident Creation application that contains the relevant device mappings.
   - ***incident_create_app***. Specify the name of the Device Sync application that contains information about the incidents that have been created.
   - **api_alias**. Specify the alias to reference the API internally to make calls.
   - ***collect_docker_info***. Select this option to collect Docker information.
   - ***collect_snow_info***. Select this option to collect ServiceNow information.

- ***use_ssh_key***. Select this option if you want to use the value in the ***ssh_key*** field to authenticate to the remote system.

4. Click **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.
5. On the **Application** page, click **[Run]** ( ). The application generates a report that you can access on the **Reports** page ( ):



This diagnostic report displays overall PowerFlow settings, such as the PowerFlow version, Docker version, kernel version, hostname, cluster settings, scheduled applications, CPU and memory statistics, installation date, and cache information.

The Read SL1 RBA Queue and Retry PowerFlow Applications Report

If you are using PowerFlow to sync incidents from a third-party application like ServiceNow or Cherwell, you can enable Run Book Action (RBA) queue retries to keep from losing any incidents if PowerFlow is unavailable. Those pending PowerFlow applications are added to an RBA queue that you can access to retry the applications that failed. For more information, see *Enabling Run Book Automation Queue Retries*.

When you configure the "Read SL1 RBA Queue and Retry PowerFlow Applications" application, you can enable the ***generate_report*** option from the **Configuration** pane to generate a report of all of the triggered PowerFlow applications that were read from the RBA queue.

After PowerFlow runs the application and generates the report, you can select the applications in the report to see if any child applications failed to get triggered.

NOTE: The "Read SL1 RBA Queue and Retry PowerFlow Applications" application is available in the *System Utils*SyncPack version 1.1.2 or later

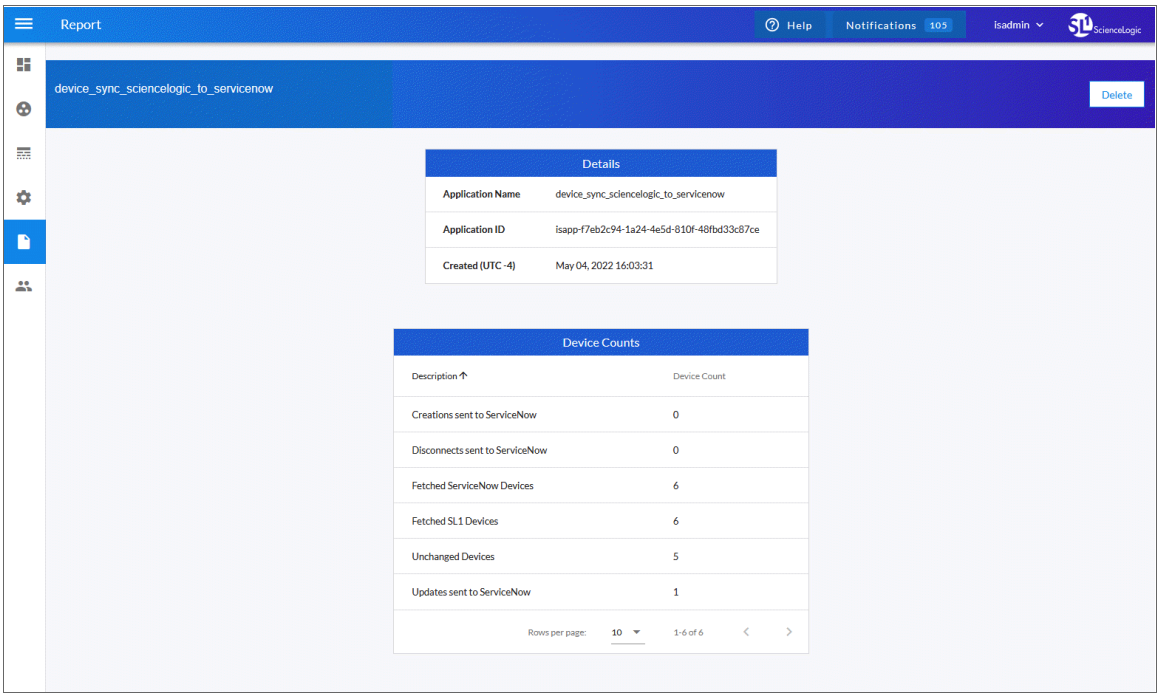
SyncPack Reports

Some reports are included with a specific SyncPack. You can download the most recent version of a SyncPack from the **PowerPacks & SyncPacks** page at the ScienceLogic Support Site at https://support.sciencelogic.com/s/.

ServiceNow CMDB SyncPack Reports

The ServiceNow CMDB SyncPack includes the following reports:

- Report: Identify Unmapped Device Classes
- Sync File Systems from SL1 to ServiceNow
- Sync Interfaces from SL1 to ServiceNow
- Sync Devices from SL1 to ServiceNow



For more information about each report, see the ***ServiceNow CMDB SyncPack*** manual.

Chapter 8

Creating and Using API Keys in SL1 PowerFlow

Overview

On the **API Keys** page ( ) of the PowerFlow user interface, you can create API keys to request PowerFlow API endpoints, specifying them by a header or a query string. These API keys are based on PowerFlow roles, which are described in *Managing Users in SL1 PowerFlow*.

You can use API keys instead of basic authentication when you use PowerFlow to integrate with technologies that do not support sending headers in API requests.

This chapter covers the following topics:

Using API Keys *196*

Creating an API Key *197*

Authenticating with an API Key *198*

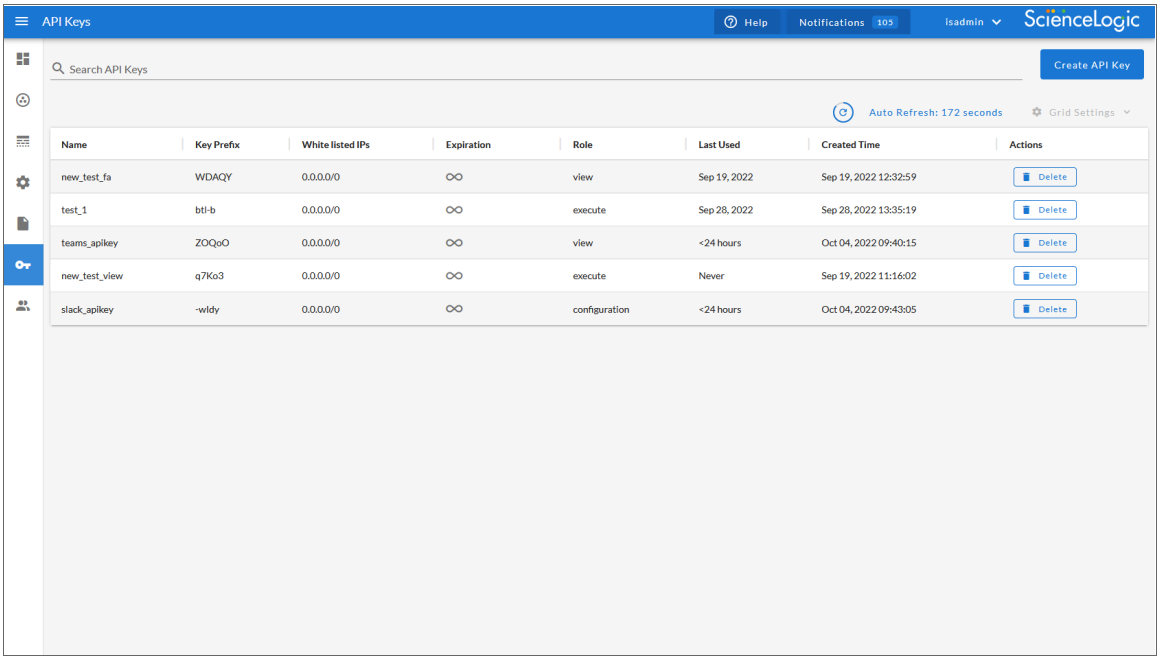
Removing an API Key *198*

Using API Keys

The API key authentication strategy gives users access to the PowerFlow API using API keys, which are also called authentication tokens. This strategy provides access to the PowerFlow API in a controllable manner, with options to restrict which hosts may or may not use certain tokens.

You can create an API key on the **API Keys** page ( ) of the PowerFlow user interface. After you create the API key, you can use it to create requests to the PowerFlow API endpoints based on the role you assigned to the key. The role defines which resources the key has access to in the PowerFlow API.

For more information about the types of roles you can assign to an API key, see *User Groups, Roles, and Permissions*.



The **API Keys** page displays the following information about each API key:

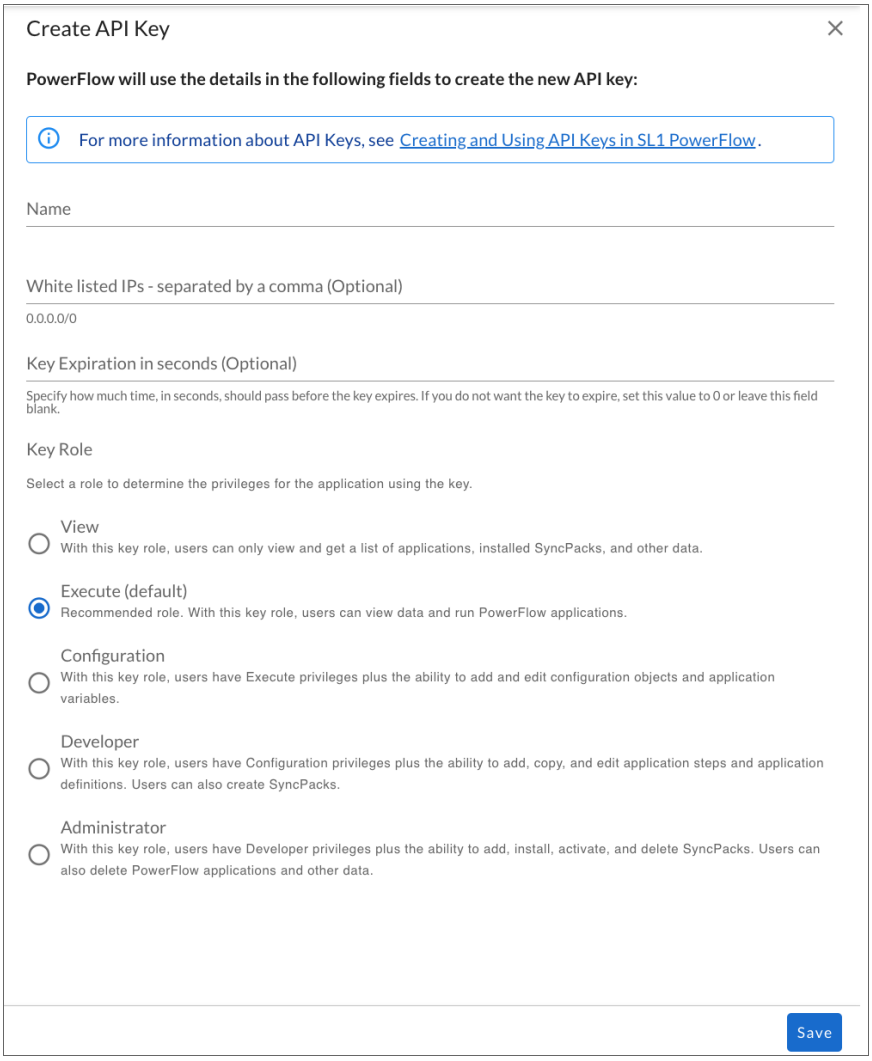
- ***Name***. The name of the API key.
- ***Key Prefix***. The first five characters of the API key. The remaining characters are not shown for security purposes.
- ***White listed IPs***. A list of one or more IP addresses or subnets that have been validated for use.
- ***Expiration***. The amount of time, in seconds, before the key expires. If the key does not have an expiration time, an infinity symbol (∞) displays in this column.
- ***Role***. The user role assigned to the API key. This role controls which resources are available to the API key in the PowerFlow API.
- ***Last Used***. The last day that the API key was used.
- ***Created Time***. The date and time that the API key was created.
- ***Actions***. Includes the **[Delete]** button, for removing an API key.

Creating an API Key

On the **API Keys** page (🔑) page, you can create an API key to use to make requests with PowerFlow API endpoints.

To create an API key:

1. In the PowerFlow user interface, go to the API Keys page and click **[Create API Key]**. The **Create API Key** pane appears:



2. Complete the following fields:
   - ***Name***. Type a name for the API key.
   - ***White listed IPs***. Specify a list of one or more IP addresses that have been approved for use. Optional.

- ***Key Expiration in seconds***. Specify how much time, in seconds, should pass before the key expires. If you do not want the key to expire, set this value to 0 or leave this field blank. Optional.
- ***Role***.Select a user role for the API key. This role controls which resources are available to the API key in the PowerFlow API. The default role is *Execute*. For more information about the different roles, see *User Groups, Roles, and Permissions*.

3. Click Save. The **API Key Successfully Added** modal appears.
4. For security purposes, the full key will only be shown once, in the above modal, so click the **[Copy]** button or the Copy icon ( ) to save the key to the clipboard. After you close this modal, you can only see the first five characters of the key on the **API Keys** page.
5. Click **[Close]**. The new API key is listed on the **API Keys** page.
6. Use the key when authenticating with the PowerFlow API, which is covered in the following topic.

Authenticating with an API Key

You can use an API key to authenticate in the following ways:

- Using a header called `PF-APIKEY` that contains the raw value of the API key, which you copied from the "Create API Key" wizard. The following example is a GET request to the application endpoint using only an API key as an authentication method:

```
curl https://<your_hostname>/api/v1/applications -H "PF-APIKEY:
DiTvNtSJpuJgOwv2OTtMaoghZQYATH3Ono48-psJ-PKrsMaE4CYWMw"
```

- Use a query parameter called `PF-APIKEY` that contains the raw value of the API key, which you copied from the "Create API Key" wizard. The following example is a GET request to the application endpoint using only an API key as an authentication method:

```
curl https://<your_hostname>/api/v1/applications?PF-
APIKEY=DiTvNtSJpuJgOwv2OTtMaoghZQYATH3Ono48-psJ-PKrsMaE4CYWMw
```

Removing an API Key

You can remove an API key by deleting it from the API Keys page. After you delete the API key, the key is removed from the PowerFlow system, and it will not have access to the PowerFlow API any longer.

To remove an API key:

1. In the PowerFlow user interface, go to the **API Keys** page.
2. Click the **[Delete]** button for the key you want to remove. A **Delete API Key** modal appears.
3. Click the **[Delete]** button to permanently delete the key.

Chapter 9

Managing Users in SL1 PowerFlow

Overview

This chapter describes how to configure authentication for PowerFlow to allow access to multiple users with a variety of roles and permissions.

This chapter also describes how to use the **Admin Panel** page ( ) of the PowerFlow user interface to manage user group access to the PowerFlow user interface. Only users with the *Administrator* role for the PowerFlow system can edit this page.

This chapter covers the following topics:

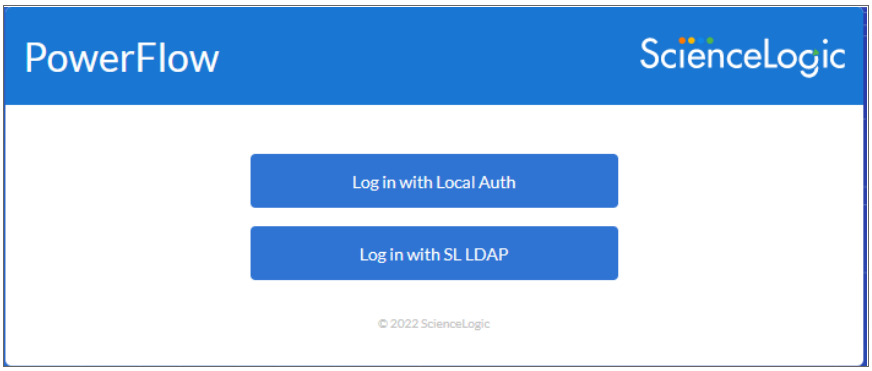
Configuring Authentication with PowerFlow 200
Common Access Card (CAC) Authentication 206
API Key Authentication 210
Role-based Access Control (RBAC) Configuration 210
Configuring Authentication Settings in PowerFlow 211
User Groups, Roles, and Permissions 212
Creating a User Group in PowerFlow 213
Managing User Sessions 214
Authentication and Authorization for Services Used by PowerFlow 216

Configuring Authentication with PowerFlow

SL1 PowerFlow supports the following authentication methods:

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default. Local authentication only supports the *isadmin* administrator user.
- ***Basic Authentication***. PowerFlow continues to support Basic Authentication as well. Because the PowerFlow SyncPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication.
- ***OAuth***. Lets PowerFlow administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as Active Directory, or using a protocol like LDAP, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.
- ***Common Access Card (CAC) Authentication***. Lets a PowerFlow user provide a CAC card through a browser to the PowerFlow root IP address. After identifying the CAC card, the ingress proxy verifies and authenticates the user. CAC authentication bypasses Dex authentication and does not use OIDC protocols. You can also use CAC authentication with LDAP, or CAC authentication with LDAP and SAN.
- ***API Key Authentication***. Provides access to the PowerFlow API in a controllable manner, with options to restrict which hosts may or may not use certain tokens.

Depending on the authentication used by your PowerFlow system, the PowerFlow login page will display a single option for logging in, or more than one option:



NOTE: You should configure your authentication strategy in the **/etc/iservices/isconfig.yml** file on the node from which you are deploying. When you have an acceptable configuration, the autoheal action copies those settings to the other nodes for consistency.

Regardless of authentication strategy, authorization and role access is configured separately, based on user or user group. For more information, see *Creating a User Group in PowerFlow*.

The following topics describe how to configure each authentication strategy for PowerFlow:

- *User Interface Login Administrator User (default)*
- *Basic Authentication Using a REST Administrator User (default)*

- *User Interface Login Using a Third-party Authentication Provider*
- *OAuth Client Authentication Using a Third-party Provider*
- *Common Access Card (CAC) Authentication*
- *CAC Authentication with LDAP*
- *CAC Authentication with LDAP and SAN*
- *API Key Authentication*

User Interface Login Administrator User (Default)

The local Administrator user is the default login user for PowerFlow. The username is "isadmin" by default, and the password gets set during the ISO installation process. The PowerFlow administrator can change the default password or the Administrator user.

This authentication strategy allows authentication of the local Administrator user through the PowerFlow user interface. The "isadmin" user is the local Administrator by default, but you can change the username of the local Administrator to something other than "isadmin" if needed.

WARNING: If you disable this authentication strategy, you must first configure an alternative provider to appropriately authenticate to the PowerFlow system and also configure a user or user group policy that has the *Administrator* role. If you disable this user without a second authentication provider configured, PowerFlow will not be able to authenticate any users.

To disable this user, set the following environment variable in the **/etc/iservices/isconfig.yml** file:

```
BASIC_AUTH: False
```

To change the local Administrator username, set the following environment variable in the **/etc/iservices/isconfig.yml** file:

```
BASIC_AUTH_USER: "username"
```

NOTE: Before changing the local Administrator username, make sure that the user has *Administrator* permissions in the PowerFlow system. For more information, see *User Groups, Roles, and Permissions*.

ScienceLogic recommends that you use the same system-wide password for the local Administrator user, which is located in **/etc/iservices/is_pass**.

Starting in PowerFlow version 3.0.0, you can use the following command to update the PowerFlow Administrator (*isadmin*) user password:

```
pfctl --host IP isadmin:host_password password set -p 'new_password'
```

This command replaces the **ispasswd** script from earlier releases of PowerFlow, which was found in **/opt/iservices/scripts/ispasswd**. The **ispasswd** script will be deprecated in a future release.

Alternatively, you can set a different password for the local Administrator user by setting the following environment variable in **/etc/iservices/isconfig.yml**:

```
BASIC_AUTH_PASSWORD: "BASIC_AUTH_PASSWORD"
```

Basic Authentication Using a REST Administrator User (Default)

Basic Authentication through a REST administrator user enables the local Administrator user to access the PowerFlow API through REST using Basic Authentication. This authentication strategy enables users to make queries through the API using `<isadmin:password>` basic authentication.

This Basic Authentication is enabled by default. The REST administrator uses the same environment variables and configuration as the *user interface login Administrator user*.

Basic Authentication is limited to only the local Administrator user. If your PowerFlow system is configured to use a different authentication provider, you must authenticate using OAuth 2.0 and a bearer token. For more information, see *OAuth Client Authentication Using an Internal Provider*.

User Interface Login Using a Third-party Authentication Provider

This authentication strategy lets you set up user authentication through a third-party authentication provider, such as LDAP, Active Directory (AD), or OAuth 2.0. You configure additional providers and their connectors in the **/etc/iservices/isconfig.yml** file, following the Dex connector configuration described below. This authentication strategy is automatically disabled when no connectors are present in the configuration.

After you configure the providers, users can select one of the defined providers or the local Administrator authentication, and authenticate using that provider.

When using this authentication strategy, the PowerFlow system automatically retrieves the LDAP or AD groups to which the user belongs. You can use these groups can be used to apply specific role permissions. For more information, see *Role-based Access Control (RBAC) Configuration*.

> **NOTE:** By default, no third-party authentication providers are configured in PowerFlow.

Credentials for user authentication exist only with the third-party authentication provider, and the credentials are not imported into PowerFlow. The only information that PowerFlow retains for these users are the roles and permissions attached to the user names.

To configure a third-party authentication provider:

1. Go to https://github.com/dexidp/dex#connectors and locate the connector type, such as LDAP, OIDC, or OAuth 2.0, that you would like to use for authentication.
2. Click the link for the connector type to view example configuration options for the connector.
3. Update the **/etc/iservices/isconfig.yml** file and add a **DEX_CONNECTORS** section with the configuration for the connector you want to use. The **DEX_CONNECTORS** section in the **isconfig.yml** file is identical to the **CONNECTORS** section described in the Dex documentation.
4. Re-deploy the PowerFlow stack and check for errors in docker service logs **iservices_dexserver**. If the dexserver starts successfully and PowerFlow is running, then PowerFlow has accepted the configuration.

5. Next, log in to the PowerFlow system with a user provided by the newly configured authentication.
6. Look for errors with searching users or user groups in the user interface or the Docker service logs.

Code Example: **isconfig.yml** file with an Active Directory authentication provider

For reference, the following is an example **/etc/iservices/isconfig.yml** file with an Active Directory authentication provider configured to look up users and their groups:

```
HOST_ADDRESS: 10.1.1.111
CLIENT_ID: isproxy
CLIENT_SECRET: knivq7uDPVORdrSlWJ0I4YiiwuQgGDsf9rMWquoInYs
SESSION_SECRET: BDLO2xPrBs_s-YkqY-j4lN6VPeBzyrVsYt_P10oWbn0
DB_HOST: couchbase.isnet,localhost
BIND_DN: auser # this can be encrypted for security
BIND_PW: password # this can be encrypted for security

DEX_CONNECTORS:
- type: ldap
  # Required field for connector id.
  name: ldap
  id: ldap
  # Required field for connector name.
  config:
    host: rstcsdc01.sciencelogic.local:636
       # Host and optional port of the LDAP server in the form "host:port".
       # If the port is not supplied, it will be guessed based on "insec-
ureNoSSL",
       # and "startTLS" flags. 389 for insecure or StartTLS connections, 636
       # otherwise.

    insecureNoSSL: false
       # The insecureNoSSL field is required if the LDAP host is not using TLS
(port 389).

    insecureSkipVerify: true
       # If a custom certificate isn't provide, this option can be used to turn
on
       # TLS certificate checks. As noted, it is insecure and shouldn't be used
outside
       # of explorative phases.
```

```
    bindDN: '{{BIND_DN}}'
    bindPW: '{{BIND_PW}}'
       # The DN and password for an application service account. The connector
uses
       # these credentials to search for users and groups. Not required if the
LDAP
       # server provides access for anonymous auth.
       # Please note that if the bind password contains a `$`, it has to be
saved in an
       # environment variable which should be given as the value to the bindPW
field.

    usernamePrompt: silo credentials
       # The attribute to display in the provided password prompt. If unset,
will
       # display "Username"

    userSearch:
       # User search maps a username and password entered by a user to a LDAP
entry.
      baseDN: OU=Domain User Accounts,DC=ScienceLogic,DC=local
         # BaseDN to start the search from.
      filter: "(objectClass=user)"
         # Optional filter to apply when searching the directory.
      username: userPrincipalName
         # username attribute used for comparing user entries.
         # The following three fields are direct mappings of attributes on the
user entry.
      idAttr: DN
         # String representation of the user.
      emailAttr: userPrincipalName
         # Required. Attribute to map to email.
      nameAttr: cn
         # The nameAttr field maps to the display name of users. No default
value.
    groupSearch:
       # Group search queries for groups given a user entry. Group searches must
match a user
       # attribute to a group attribute.
```

```
      baseDN: OU=Domain Groups,DC=ScienceLogic,DC=local
         # BaseDN to start the search from.
      filter: "(objectClass=group)"
         # Optional filter to apply when searching the directory.
         # The following list contains field pairs that are used to match a user
to a group.
         # It adds an additional requirement to the filter that an attribute in
the group
         # must match the user's attribute value.
      userAttr: DN
         # The userAttr field is the attribute used from the user search query.
         # to match to an element of the group search.
      groupAttr: member
         # The groupAttr field is the attribute of the group results that should
match the userAttr value.
      nameAttr: cn
         #  This value must exactly match the group defined in the IS Group con-
figuration.
```

OAuth Client Authentication Using a Third-party Provider

OAuth Client Authentication provides user authentication with a configured third-party provider, such as LDAP or AD. This allows users to use clients authenticating with OAuth 2.0 bearer tokens to authenticate against their configured provider.

The OAuth Client Authentication strategy is automatically enabled when a authentication provider is configured. You can configure this strategy using the same parameters as the *User Interface Login Using a Third-party Authentication Provider*.

The PowerFlow system provides discovery endpoints for all OAuth 2.0 required endpoints. The discovery address is: **https://IS-IP:5556/dex/.well-known/openid-configuration**.

Using these discovery endpoints, you can use an OAuth 2.0 client to generate a secure token using a third-party authentication provider. You can use the generated secure token as a bearer authentication token (specified in request headers) to authenticate and make requests.

The **client_secret** and **session_secret** are unique, randomly generated strings generated for each IS deployment. To obtain an OAuth 2.0 token, you will need these values, which you can find in the **/etc/iservices/isconfig.yml** file.

Basic Authentication Lockout Removal

PowerFlow supports OAuth 2.0 with OpenID Connect, which lets PowerFlow administrators use third-party authentication providers, such as LDAP or Active Directory, to enforce user authentication and lockout policies. PowerFlow continues to support Basic Authentication as well, but PowerFlow no longer enforces automatically locking out the *isadmin* user if that user has too many failed login attempts.

If you are concerned about removing the lockout functionality for the *isadmin* user, you can perform one of the following actions:

1. Change the default username from *isadmin* to a different name to prevent brute force type attacks on the *isadmin* user.
2. Disable Basic Authentication and use third-party authentication providers, such as your company's LDAP server, to enforce user authentication and lockout policies.

> **NOTE:** Before disabling Basic Authentication, make sure that all scripts or tools that query the PowerFlow API have been updated to use OAuth 2.0.

Common Access Card (CAC) Authentication

Enabling Common Access Card (CAC) authentication lets a PowerFlow user provide a CAC card through a browser to the PowerFlow root IP address. After identifying the CAC card, the ingress proxy verifies and authenticates the user. CAC authentication bypasses Dex authentication and does not use OIDC protocols.

Applying CAC Authorization

To enable Common Access Card (CAC) authorization on the PowerFlow system:

1. Copy your CA certificate to the PowerFlow system; For example, **server-ca.crt**.
2. Add the corresponding `server-ca` secret into the **gui** container in the **docker-compose** file:

```
gui:
  secrets
   - source: server-ca.crt
secrets:
 server-ca.crt:
   file: /etc/iservices/server-ca.crt
```

3. Add the following line to **/etc/iservices/isconfig.yml**:

```
CAC_AUTH: 'true'
```

Adding CRL to CAC Authentication

To add a certificate revocation list (CRL) to CAC authentication:

1. If a PEM-formatted CRL file needs to be added, copy that file to all PowerFlow nodes; For example, **/var/crl**.

2. Add the corresponding `volume` information to `gui` in the **docker-compose** file:

```
gui:
...
volumes:
    - read_only: true
      source: ca_crl
      target: /var/crl
      type: volume
volumes:
...
  ca_crl: {}
```

3. After the CRL is applied to the volume, update the configuration of **/etc/iservices/isconfig.yml** with the following line and then re-deploy:

```
CAC_CRL: 'true'
```

CAC Authentication with LDAP

CAC authentication with LDAP uses the standard CAC authentication process, and it additionally searches the LDAP server for relevant groups of the user, which can be used for role-based access control (RBAC) checking later.

This authentication process uses the same DEX_CONNECTORS configuration settings for the configured LDAP/AD server and associated certificates to perform a search of the users' LDAP server. Using the search strings configured for Dex, if the CAC logic identifies group membership, the user session will be updated with their groups.

Environment Expectations

CAC authentication with LDAP requires the following:

- The user must provide an Admin Bind DN and password to provide a search of the LDAP system.
- The user from CAC will not attempt to be bound and authenticated to LDAP.
- The end user has necessary certificates to form a trusted **ldaps//** connection. SSL certificate verification can be disabled, but is not recommended.
- The administrator is aware of the LDAP environment, search attributes, and object classes necessary to link either a CAC user's SAN or CN to their group membership
- Only SAN or CN is supported for group membership search.

Add LDAP to CAC Query

CAC authentication with LDAP requires **ldapCA.pem**, which is a file with the internal LDAP server CA chain. This will get concatenated to **tls-ca-bundle.pem**, the ca-trust bundle of the **is_gui** container.

To copy an LDAP CA certificate for verification:

1. Create **tls-ca-bundle.pem** by running the following commands:

```
docker cp $(docker ps -q --filter name=iservices_gui):/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem /etc/iservices/tls-ca-bundle.pem
```

```
cat ldapCA.pemt >> /etc/iservices/tls-ca-bundle.pem
```

2. Use an SCP tool to move **tls-ca-bundle.pem** to **/etc/iservices/**.

Update the Docker configuration:

```
configs:
  isconfig:
    file: /etc/iservices/isconfig.yml
  tlsbundle:
    file: /etc/iservices/tls-ca-bundle.pem
```

```
  dexserver:
    configs:
    - source: isconfig
      target: /etc/iservices/isconfig.yml
    - source: tlsbundle
      target: /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
    environment:
      db_host: couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.isnet
    networks:
      isnet:
        aliases:
        - dexserver
        - dexserver.isnet
    secrets:
    - source: is_pass
```

```
gui:
    read_only: true
    configs:
    - source: isconfig
      target: /etc/iservices/isconfig.yml
```

```
    - source: tlsbundle
      target: /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
...
```

Update the **isconfig.yml** configuration:

```
CAC_AUTH: 'true'
CAC_LDAP_VERIFY: 'true'
DEX_CONNECTORS:
- type: ldap
  name: ldap
  id: ldap

  config:
    host: rstcsdc01.sciencelogic.local
    rootCA: /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem

    bindDN: CN=svc-ldap-commander,OU=_Service,OU=Domain User Account-
s,DC=sciencelogic,DC=local
    bindPW: <pass removed>
    usernamePrompt: silo credentials
    userSearch:
      baseDN: OU=Domain User Accounts,DC=ScienceLogic,DC=local
      filter: "(objectClass=user)"
      username: userPrincipalName
      idAttr: DN
      emailAttr: userPrincipalName
      baseDN: OU=Domain User Accounts,DC=ScienceLogic,DC=local
      filter: "(objectClass=user)"
      username: userPrincipalName
      idAttr: DN
      emailAttr: userPrincipalName
      nameAttr: cn

    groupSearch:
      baseDN: OU=Domain Groups,DC=ScienceLogic,DC=local
      filter: "(objectClass=group)"
      userAttr: DN
      groupAttr: member
      nameAttr: cn
```

To make sure that user sessions are terminated upon account deletion from the Active Directory server, the following configuration can be set in the **isconfig.yml** file. This setting ensures that the user account validation is executed during every user request:

```
FORCE_CAC_LDAP_REVALIDATION: 'true'
```

This configuration is disabled by default, as it can cause overhead in the Active Directory server. For more information, see *Configuring Authentication Settings in PowerFlow*.

CAC Authentication with LDAP and SAN

CAC authentication with LDAP and SAN allows the user to use their Subject Alternative Name (SAN) value when performing group lookup.

To implement CAC authentication with LDAP and SAN , use the configuration variable called **CAC_LDAP_SEARCH_BY**. This environment variable specifies whether to use Subject Alternative Name (SAN) or Common Name (CN) for LDAP group membership searches when LDAP is enabled and when CAC_LDAP_VERIFY is enabled. The default value of this environment variable is *san*; use *cn* for Common Name in the environment variable section.

API Key Authentication

The API key authentication strategy gives users access to the PowerFlow API using API keys, which are also called authentication tokens. This strategy provides access to the PowerFlow API in a controllable manner, with options to restrict which hosts may or may not use certain tokens.

You can create an API key on the **API Keys** page ( ) of the PowerFlow user interface. After you create the API key, you can use it to create requests to the PowerFlow API endpoints based on the role you assigned to the key. The role defines which resources the key has access to in the PowerFlow API.

For more information about using API keys, see *Creating and Using API Keys in SL1 PowerFlow*.

Role-based Access Control (RBAC) Configuration

Regardless of the authentication method you have chosen to use, the role-based permissions assigned to users is applied in the same way. PowerFlow uses the existing users and user groups that you have already set up in your authentication system.

Assigning a Role to a Specific User

By applying a role permission to a single username on the **Admin Panel** page, or through the API, permissions will be granted to any user who logs in through a provider matching that username.

Assigning Roles to a Specific User Group

By applying a role permission to a group name on the **Admin Panel** page, or through the API, permissions will be granted to any user who logs in and is a member of the specified group.

For authentication to work properly, **group_search** must be configured in the authentication provider's connector information. When requesting authentication tokens, be sure to also request the **groups** claim.

> **NOTE:** If a user belongs to multiple groups, with varying permissions defined to each group, the user will be permitted to do all actions provided by the group that provides the most roles.

Viewing User and Group Information

After configuring your third-party authentication connector in Dex, you can run the following command to view the search strategy and group results for any user that attempts to authenticate by looking at the Dexserver logs:

```
docker service logs -f iservices_dex
```

Changing Roles and Permissions

To change roles and permissions through the PowerFlow user interface, go to the **Admin Panel** page ( ) to create and edit the roles and permissions of the users or user groups. For more information, see *Creating a User Group in PowerFlow*.

To change roles and permissions through the API, refer to the **swagger.yml** file for API required parameters and endpoints to update roles and permissions.

Configuring Authentication Settings in PowerFlow

The following configuration settings can be configured and used with the PowerFlow authentication and authorization strategies:

- **DEX_CONNECTORS**. This environment variable specifies which authentication providers Dex will use. For more information, see *User Interface Login Using a Third-party Authentication Provider*.
- **BASIC_AUTH_USER**. This environment variable specifies the basic_auth and admin login username. For more information, see *User Interface Login Administrator User*.
- **BASIC_AUTH_PASSWORD**. This environment variable specifies the basic_auth and admin login password. For more information, see *User Interface Login Administrator User*.
- **BASIC_AUTH**. This environment variable specifies whether the local Administrator user will be enabled. For more information, see *User Interface Login Administrator User*.
- **BIND_DN**. Use this encrypted value in the **dex_connectors** section when using LDAP as a connector.
- **BIND_PW**. Use this encrypted value in the **dex_connectors** section when using LDAP as a connector.
- **CLIENT_ID**. The IS client_id in use by default is *isproxy*, and you should not change this value.
- **CLIENT_SECRET**. The client_secret is a randomly generated string unique to each PowerFlow deployment. In most configurations, you do not need to change this secret.
- **CAC_AUTH**. This environment variable specifies whether to enable CAC certificate checking. If this is used, there should be a public CA certificate added as a **server-ca.crt** secret to the **gui** container. For more information, see *CAC Authentication*.

- **CAC_CRL**. This environment variable specifies whether certificate revocation list (CRL) checking is enabled for the CAC client certificate. If this is true and updated, the user should must add a **ca_crl:/var/ca_crl** volume to both the **gui** and **syncpacks_steprunner** services.
- **CAC_LDAP_VERIFY**. This environment variable specifies whether to enable lookup of group permissions based on the user Subject Alternative Name (SAN) or Common Name (CN) to the configured LDAP server. If this is configured, the user must configure an LDAP or AD server under the DEX_CONNECTORS environment variable.
- **CAC_LDAP_SEARCH_BY**. This environment variable specifies whether to use Subject Alternative Name (SAN) or Common Name (CN) for LDAP group membership searches when LDAP is enabled and when CAC_LDAP_VERIFY is enabled. The default value of this environment variable is *san*; use *cn* for Common Name in the environment variable section.
- **FORCE_CAC_LDAP_REVALIDATION**: This environment variable ensures that user sessions are terminated upon account deletion from the AD server when using CAC authentication. This variable will only work if the **CAC_LDAP_VERIFY** variable is enabled. This configuration is disabled by default, as this setting can cause overhead in the AD server.

User Groups, Roles, and Permissions

On the **Admin Panel** page ( ), you can edit and create ***user groups*** that define the different roles and permissions for your users. Depending on their assigned permissions, users have access to certain features, or they are blocked from certain features.

On this page, you can allocate permissions to users and user groups that exist in your authentication system. The ***User Group*** field in the PowerFlow user interface can specify either an individual user or a group that already exists in your authentication system.

The available roles and permissions for PowerFlow include the following:

- ***View***. The user can view and get via the API the list of applications, the list of installed SyncPacks, the dashboards, the reports, the configuration objects, and the results of application runs.
- ***Execute***. The user has all of the privileges of the **View** permission, but the user can also trigger application runs through the user interface or the API.
- ***Configuration***. The user has all of the privileges of the **Execute** permission, but the user can also add and edit configuration objects and modify the application variables.
- ***Developer***. The user has all of the privileges of the **Configure** permission, but the user can also add, copy, and edit step definitions; add, copy, and edit application variables; and create SyncPacks.
- ***Administrator***. The user has all of the privileges of the **Develop** permission, but the user can also add, install, activate, and delete SyncPacks; delete applications, steps, and configuration objects; and access to (but not authorization to) the user interfaces for Couchbase, Flower, and RabbitMQ.

Additional information about roles and permissions in PowerFlow:

- Roles in PowerFlow are automatically assigned to a user based on the user's group with the external provider (such as LDAP or AD) .
- If a PowerFlow system does not have an external provider configured, such as LDAP or Active Directory, that PowerFlow system can only support a single *isadmin* Administrator user.

- The *only* password saved in the PowerFlow system is the password for the *isadmin* Administrator user. All other user passwords are saved in the third-party authentication provider configured for PowerFlow.
- The authentication configuration used by PowerFlow is also supported in SL1.
- API queries made by users will also be checked for proper authorization.
- The PowerFlow administrator can configure which users are in which roles.
- The PowerFlow administrator can test and configure the LDAP and Active Directory settings to ensure that the settings work before proceeding.
- The PowerFlow administrator can always log in with the *isadmin* Administrator user, even if the underlying LDAP provider is inaccessible.

Creating a User Group in PowerFlow

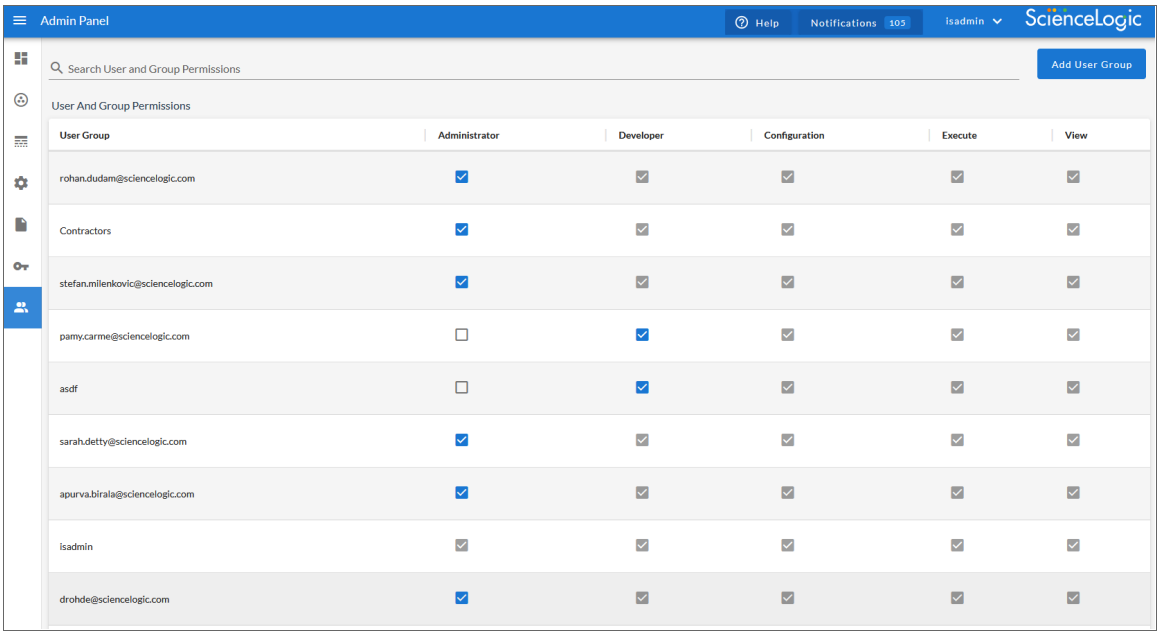
On the **Admin Panel** page, you can allocate permissions to users and user groups that exist in your authentication system.

If you have the *Administrator* role, you can modify the permissions for each user group on the page. You cannot change the permissions for the user group to which you currently belong. A user with the *Administrator* role can also create a user group and assign permissions to that group.

NOTE: The *only* password saved in the PowerFlow system is the password for the *isadmin* Administrator user. All other user passwords are saved in the third-party authentication provider configured for PowerFlow.

To create a user group:

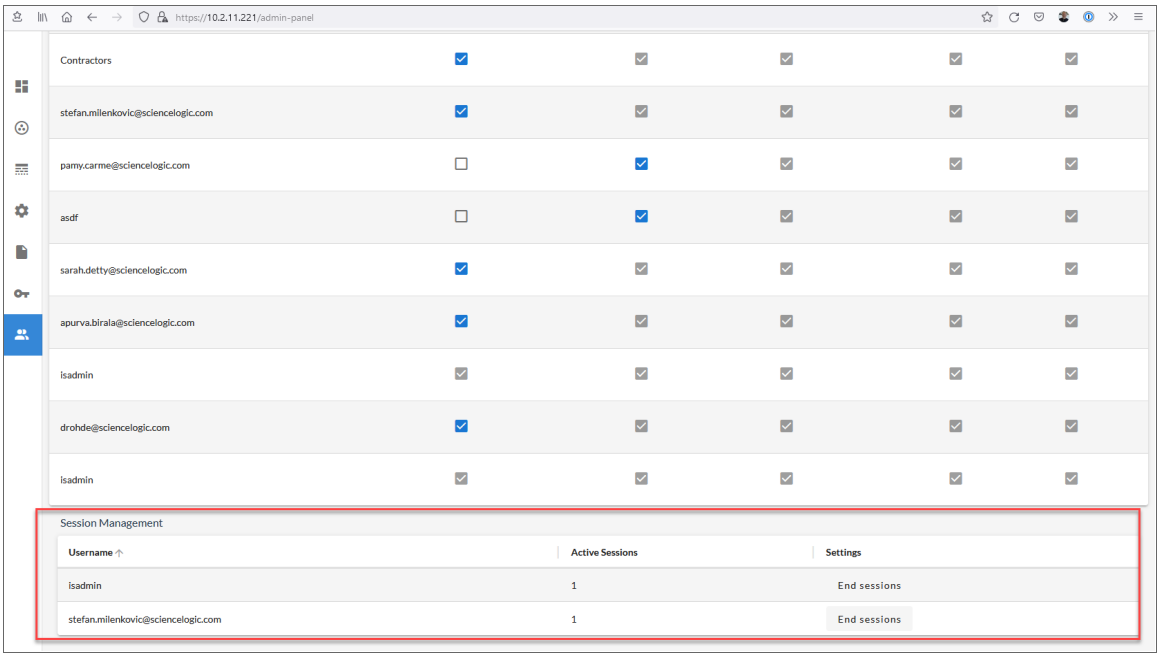
1. In the PowerFlow user interface, go to the **Admin Panel** page ( ):



2. Click **[Add User Group]**. The **Create User Group** window appears.
3. Complete the following fields:
   - ***User Group Name***. Type a name for this user group, *without* spaces. This is the name that the user or the users in the user group will use when logging in to PowerFlow. You can specify either an individual user or a user group that already exists in your authentication system.
   - ***Choose Permissions Group***. Click **Add Permission** to select the permission to assign to this new user group. Your choices include *Developer*, *Configuration*, *View*, *Execute*, and *Administrator*, and these choices are defined in ***User Groups, Roles, and Permissions***.
4. Click **[Create User Group]**. The group is added to the **Admin Panel** page.
5. If you want to give a user group *more* permissions, select the empty checkbox ( ) for that role from the **Admin Panel** page. You must have the *Administrator* role to change these settings.
6. If you want to *remove* permissions from a user group, select the highest level role, which has a blue check mark ( ). The role to the right (with fewer permissions) becomes the new role for that user group.

Managing User Sessions

A user with the *Administrator* role can use the **Session Management** pane of the **Admin Panel** page ( ) to monitor which users have been logged into the PowerFlow system. The admin user can end active sessions as needed. Non-admin users can view user sessions, but they cannot end those sessions.



When an administrator ends a session for a user by clicking the **[End sessions]** button, that user is redirected to the PowerFlow login page.

Enabling Session Management

Session management is disabled by default. You can use the following set of configuration variables to enable and configure session management. You can set these variables in the PowerFlow configuration file **/etc/iservices/isconfig.yml** or on the services environment variables:

- **ENABLE_SESSION_STORAGE**. Set to **true** to enable session management with default values. This variable is set to false by default.
- **SESSION_IDLE_TIME**: Specify the idle time in seconds. The default is 900 seconds. This value and the following values should be declared as a string.
- **SESSION_RENEW_EVERY**: Specify the automatic session regeneration time in seconds. The default is 600 seconds.
- **SESSION_ABSOLUTE_TIME**: Specify the longest time allowed for a session, in hours, such as **25h**. You can also use minutes, such as **60m**. This variable uses Dex notation. The default is **25h** (25 hours).
- **CONCURRENT_SESSIONS_BY_USER**. Specify the number of sessions that can be running at the same time, by number of users. The default is 30 users.

> **NOTE:** When the number of user sessions is exceeded, the user trying to log in is redirected to a "Session limit has been reached" page, and then the user is returned to the login page.

You can use the following configurations in the **/etc/iservices/isconfig.yml** file to improve load balancer compatibility if the load balancer sends requests to the client in proxy protocol format like AWS ELB:

- **LOAD_BALANCED: true**. Setting this value to **true** specifies that the load balancer will send requests to the client in proxy protocol format. This value is **false** by default.
- **RATE_LIMITED**. Setting this value to **true** enables rate limiting. This value is **false** by default.
- **RATE_LIMIT_REQUESTS_PER_SECOND**. This value specifies the number of rate limit requests per second. The default is '**50**'.
- **RATE_LIMIT_BURST**. This value specifies the rate limit burst. The default is '**100**'.

IMPORTANT: You will need to re-deploy the PowerFlow stack for any changes to the **docker-compose.yml** file to take place.

In addition, the exposed ports in the **docker-compose.yml** file are set to **mode: host** to let PowerFlow capture the proper client IP address of the requests being sent into PowerFlow. This setting lets PowerFlow set the proper rate limits and log transactions. This feature does not allow using the Swarm ingress; as a result, you will need to scale the gui container and place the container in the nodes that will be expecting ingress traffic.

Authentication and Authorization for Services Used by PowerFlow

The PowerFlow administrator can control the level of access to the specific PowerFlow services, including Couchbase and RabbitMQ. Authentication for these services is provided by Dex authentication, which is already used for role-based access control (RBAC) in PowerFlow.

NOTE: This feature requires LDAP/AD authentication for the PowerFlow system. For more information, see *OAuth Client Authentication Using a Third-party Provider*.

Couchbase

- Couchbase authentication. To access the Couchbase user interface, a user must log in to PowerFlow first, using his or her PowerFlow credentials. If the user is authorized to access the Couchbase user interface, the user can add port "8091" to the PowerFlow URL, and the user will be automatically redirected to the Couchbase user interface.
- Couchbase authorization. The roles and user groups defined in PowerFlow are applied to the Couchbase user interface based on the default user group policies. The PowerFlow administrator can update these user policies to specify which groups can access Couchbase.

  Couchbase authorization uses the following default permissions:

  - *Administrator*. The user has access to all resources at all levels.
  - *Developer*. The user can add and edit buckets and documents, but the user cannot delete anything.
  - *Configuration*. The user can add and delete indexes and add nodes to Couchbase.

- *Execute*. The user has read-only access.
- *View*. The user cannot login to the Couchbase user interface. This was explicitly set that way as Couchbase is the main database for PowerFlow.

RabbitMQ

- RabbitMQ authentication. RabbitMQ authentication works the same as PowerFlow authentication and Couchbase authentication.
- RabbitMQ authorization. RabbitMQ authorization uses the following default permissions:
  - *Administrator*: The user has access to all resources, at all levels, and the user can create internal users and policies. These policies do not impact PowerFlow users.
  - *Developer*: The user can create resources and read all resources on all vhosts.
  - *Configuration*: The user can create queues and exchanges only in the default vhost, but the user can read queues and exchanges on all vhosts.
  - *Execute*: The user can read queues and exchanges on all vhosts, but the user cannot create or configure any resources.
  - *View*: The user can only view queues and exchanges on the default vhost.

NOTE: If you want to disable the auto-login feature for RabbitMQ and Couchbase, you can set the **force_auth_validation** environment variable to "true" under the GUI container configurations in the **docker-compose** file. Setting this variable to "true" allows you to access the Couchbase or RabbitMQ user interface to address issues without needed to authorize. If the flag is missing or set to "false", the auto-login feature continues to work.

Chapter 10

Viewing Logs in SL1 PowerFlow

Overview

This chapter describes the different types of logging available in PowerFlow.

This chapter covers the following topics:

Logging Data in PowerFlow 219
Logging Configuration 220
PowerFlow Log Files 221
Working with Log Files 221
Viewing the Step Logs and Step Data for a PowerFlow Application 224
Removing Logs on a Regular Schedule 225

Logging Data in PowerFlow

PowerFlow allows you to view log data locally, remotely, or through Docker.

Local Logging

PowerFlow writes logs to files on a host system directory. Each of the main components, such as the process manager or Celery workers, and each application that is run on the platform, generates a log file. The application log files use the application name for easy consumption and location.

In a clustered environment, the logs must be written to the same volume or disk being used for persistent storage. This ensures that all logs are gathered from all hosts in the cluster onto a single disk, and that each application log can contain information from separately located, disparate workers.

You can also implement log features such as rolling, standard out, level, and location setting, and you can configure these features with their corresponding environment variable or setting in a configuration file.

NOTE: Although it is possible to write logs to file on the host for persistent debug logging, as a best practice, ScienceLogic recommends that you utilize a logging driver and write out the container logs somewhere else.

TIP: You can use the "Timed Removal" application in the PowerFlow user interface to remove logs from Couchbase on a regular schedule. On the **Configuration** pane for that application, specify the number of days before the next log removal. For more information, see *Removing Logs on a Regular Schedule*.

Remote Logging

If you use your own existing logging server, such as Syslog, Splunk, or Logstash, PowerFlow can route its logs to a customer-specified location. To do so, attach your service, such as logspout, to the microservice stack and configure your service to route all logs to the server of your choice.

CAUTION: Although PowerFlow supports logging to these remote systems, ScienceLogic does not officially own or support the configuration of the remote logging locations. Use the logging to a remote system feature at your own discretion.

Viewing Logs in Docker

You can use the Docker command line to view the logs of any current running service in the PowerFlow cluster. To view the logs of any service, run the following command:

```
docker service logs -f iservices_<service_name>
```

Some common examples include the following:

```
docker service logs –f iservices_couchbase
```

```
docker service logs –f iservices_steprunner
```

```
docker service logs –f iservices_contentapi
```

NOTE: *Application* logs are stored on the central database as well as on all of the Docker hosts in a clustered environment. These logs are stored at **/var/log/iservices** for both single-node or clustered environments. However, the logs on each Docker host only relate to the services running on that host. For this reason, using the Docker service logs is the best way to get logs from all hosts at once.

NOTE: The application logs stored locally on individual node filesystems can now be collected using the **powerflowcontrol** (pfctl) command-line utility **collect_pf_logs** action. For more information, see collect_pf_logs in the "Using the powerflowcontrol (pfctl) Command-line Utility" chapter of the ***SL1 PowerFlow Platform*** manual.

Logging Configuration

The following table describes the variables and configuration settings related to logging in PowerFlow:

Environment Variable/Config Setting	Description	Default Setting
logdir	The directory to which logs will be written.	/var/log/iservices
stdoutlog	Whether logs should be written to standard output (stdout).	True
loglevel	Log level setting for PowerFlow application modules.	**warning(30)**. You can use **info (20)** or **debug(10)** to see more details about the operations executed over the API and steprunners services. This setting should be only enabled while troubleshooting issues.
celery_log_level	The log level for Celery-related components and modules.	**warning(30)**. You can use **info (20)** or **debug(10)** to see more details about the operations executed over the API and steprunners services. This setting should be only enabled while troubleshooting issues.

PowerFlow Log Files

The logs from the **gui**, **api**, and **rabbitmq** services are available in the **/var/log/iservices** directory on which that particular service is running.

To aggregate logs for the entire cluster, ScienceLogic recommends that you use a tool like Docker Syslog: https://docs.docker.com/config/containers/logging/syslog/.

Logs for the gui Service

By default all nginx logs are written to **stdout**. Although not enabled by default, you can also choose to write access and error logs to file in addition to stdout.

To log to a persistent file, simply mount a file to **/host/directory:/var/log/nginx**, which will by default log both access and error logs.

For pypiserver logs, the logs are not persisted to disk by default. You can choose to persist pypiserver logs by setting the **log_to_file** environment variable to true.

If you choose to persist logs, you should mount a host volume to **/var/log/devpi** to access logs from a hos. For example: **/var/log/iservices/devpi:/var/log/devpi**.

Logs for the api Service

By default all log messages related to PowerFlow are written out to **/var/log/iservices/contentapi**.

PowerFlow-specific logging is controlled by existing settings listed in *Logging Configuration*. Although not enabled by default, you can also choose to write nginx access and error logs to file in addition to stdout.

To log to a persistent file, simply mount a file to **/file/location/host:/var/log/nginx/nginx_error.log** or **/host/directory:/var/log/nginx** depending if you want access or error logs.

Logs for the rabbitmq Service

All **rabbitmq** service logs are written to stdout by default. You can choose write to a logfile (**stdout** or **logfile**, not both).

To write to the logfile:

1. Add the following environment variable for the rabbitmq service:

```
RABBITMQ_LOGS: "/var/log/rabbitmq/rabbit.log"
```

2. Mount a volume: **/var/log/iservices/rabbit:/var/log/rabbitmq**

The retention policy of these logs is 10 MB, for a total of five maximum logs written to file.

Working with Log Files

Use the following procedures to help you locate and understand the contents of the various log files related to PowerFlow.

Accessing Docker Log Files

The Docker log files contain information logged by all containers participating in PowerFlow. The information below is also available in the PowerPacks listed above.

To access Docker log files:

1. Use SSH to connect to the PowerFlow instance.
2. Run the following Docker command:

   ```
   docker service ls
   ```

3. Note the Docker service name, which you will use for the `<service_name>` value in the next step.
4. Run the following Docker command:

   ```
   docker service logs -f <service_name>
   ```

Accessing Local File System Logs

The local file system logs display the same information as the Docker log files. These log files include debug information for all of the PowerFlow applications and all of the Celery worker nodes.

To access local file system logs:

1. Use SSH to connect to the PowerFlow instance.
2. Navigate to the **/var/log/iservices** directory to view the log files.

Understanding the Contents of Log Files

The pattern of deciphering log messages applies to both Docker logs and local log files, as these logs display the same information.

The following is an example of a message in a Docker log or a local file system log:

```
"2018-11-05 19:02:28,312","FLOW","12","device_sync_sciencelogic_to_
servicenow","ipaas_logger","142","stop Query and Cache ServiceNow
CIs|41.4114570618"
```

You can parse this data in the following manner:

```
'YYYY-MM-DD' 'HH-MM-SS,ss' 'log-level' 'process_id' 'is_app_name' 'file'
'lineOfCode' 'message'
```

To extract the runtime for each individual task, use regex to match on a log line. For instance, in the above example, there is the following sub-string:

```
"stop Query and Cache ServiceNow CIs|41.4114570618"
```

Use regex to parse the string above:

```
"stop ..... | ..."
```

where:

- Everything after the `|` is the time taken for execution.
- The string between `"stop"` and `|` represents the step that was executed.

In the example message, the "Query and Cache ServiceNow CIs" step took around 41 seconds to run.

Managing journald Settings

The journald volatile storage takes part of the memory based on the environment memory size, which might cause undesired behavior in environments where the memory is highly used by PowerFlow services. PowerFlow uses journald volatile storage, which means that all logs are kept only in memory.

Total Memory	Maximum memory used by journald
16 GB	About 800 MB
24 GB	About 1.2 GB
32 GB	About 1.6 GB
64 GB	About 3.2 GB

To check the size of journal logs on any PowerFlow version 2.2.x or later single node, run the following command:

```
du -sh /run/log/journal
```

For PowerFlow version 2.2.x, you can control those settings by updating the **/etc/docker/daemon/json** file and setting the `log-opts` max size in the `json-file` logging driver. For more information, see https://docs.docker.com/config/containers/logging/json-file/.

For PowerFlow version 2.3 or later nodes, you can clear logs with the following command (this is automatically done when you run the **healthcheck** action):

```
journalctl --vacuum-time=7d
```

You can also configure journald logs settings by using the following command to enforce small size and time limits:

```
sudo sed -i -e '/RuntimeMaxUse=/s/.*/RuntimeMaxUse=800M/' -e
'/MaxRetentionSec/s/.*/MaxRetentionSec=2week/' /etc/systemd/journald.conf
&& sudo systemctl restart systemd-journald
```

NOTE: PowerFlow updates journald volatile limits to the following values, which can be changed if you want retain fewer or more logs:

```
RuntimeMaxUse=800M

MaxRetentionSec=2week
```

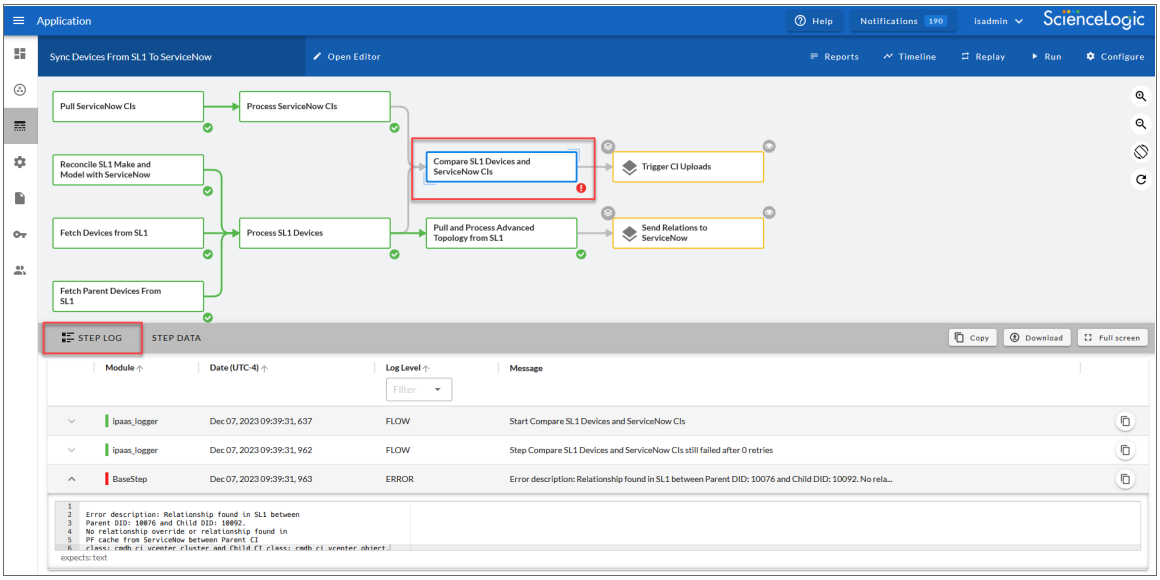
Viewing the Step Logs and Step Data for a PowerFlow Application

The **[Step Log]** tab on an **Application** page displays the time, the type of log, and the log messages from a step that you selected in the main pane. All times that are displayed in the **Message** column of this pane are in seconds, such as "stop Query and Cache ServiceNow CI List|***5.644190788269043***".

TIP: You can view log data for a step on the **[Step Log]** tab while the **Configuration** pane for that step is open.

To view logs for the steps of an application:

1. From the **[Applications]** tab, select an application. The **Application** page appears.
2. Select a step in the application.
3. Click the **[Step Log]** tab in the bottom left-hand corner of the screen. The **[Step Log]** tab appears at the bottom of the page, and it displays log messages for the selected step:



4. For longer log messages, click the down arrow icon ( ) to open the message.
5. To copy a single log, click the Copy Message icon ( ) next to that message.
6. To copy all messages in the log, click the **[Copy]** button on the gray bar.
7. To download all messages in the log, click the **[Download]** button on the gray bar. The file is saved in JSON format.
8. To view the logs in full-screen mode, click the **[Full screen]** button on the gray bar. Click the **[Full screen]** button again to turn off full-screen mode.
9. Click the **[Step Data]** tab to display the JSON data (where relevant) that was generated by the selected step.
10. Click the **[Step Log]** tab to close the tab.
11. To generate more detailed logs when you run this application, hover over the **[Run]** button and select *Debug Run*.

TIP: Log information for a step is saved for the duration of the ***result_expires*** setting in the PowerFlow system. The ***result_expires*** setting is defined in the **opt/iservices/scripts/docker-compose.yml** file. The default value for log expiration is 7 days. This environment variable is set in seconds.

Removing Logs on a Regular Schedule

The "Timed Removal" application in the PowerFlow user interface lets you remove logs from Couchbase on a regular schedule.

To schedule the removal of logs:

1. In the PowerFlow user interface, go to the **[Applications]** tab and select the "Timed Removal" application.
2. Click the **[Configure]** button. The **Configuration** pane appears.
3. Complete the following fields:
   - ***Configuration***. Select the relevant configuration object to align with this application. Required.
   - ***time_in_days***. Specify how often you want to remove the logs from Couchbase. The default is 7 days. Required.
   - ***remove_dex_sessions***. Select this option if you want to remove Dex Server sessions from the logs.
4. Click the **[Save]** button and close the **Configuration** pane.
5. Click the **[Run]** button to run the "Timed Removal" application.

Chapter 11

Using the powerflowcontrol (pfctl) Command-line Utility

Overview

This chapter describes how to use the **powerflowcontrol** (pfctl) command-line utility to run automatic cluster **healthcheck** and **autoheal** actions that will verify the configuration of your PowerFlow cluster or a single PowerFlow node. The **powerflowcontrol** utility also includes an **autocluster** action that performs multiple administrator-level actions on either the node or the cluster. You can use this action to automate the configuration of a three-node cluster.

> **NOTE:** The **powerflowcontrol** command-line utility was called **iservicecontrol** in previous release of SL1 PowerFlow. You can use either "iservicecontrol" or "pfctl" in commands, but "iservicecontrol" will eventually be deprecated in favor of "pfctl".

For more information about using the **powerflowcontrol** (pfctl) command-line utility, watch the video at https://www.youtube.com/watch?v=IWTACuLhepA.

This chapter covers the following topics:

What is the powerflowcontrol (pfctl) Utility? 227
healthcheck and autoheal 229
autocluster 233
apply_<n>GB_override, verify_<n>GB_override 234
check_dex_connectivity 235
check_docker_service_update_status 235
check_redis_maxmemory, fix_redis_maxmemory 235

logcollect *236*
open_firewall_ports *237*
Increasing the PowerFlow Docker Swarm Heartbeat in Cluster Environments *237*
password *238*
Disabling TLS Verification *239*

What is the powerflowcontrol (pfctl) Utility?

The **powerflowcontrol** (pfctl) command-line utility included in PowerFlow contains automatic cluster **healthcheck** and **autoheal** actions that will verify the configuration of your cluster or single node. The utility also includes an **autocluster** action that performs multiple administrator-level actions on either the node or the cluster.

The **powerflowcontrol** utility is included in the latest release of PowerFlow. If you need a newer version, you can download the latest version from the ScienceLogic Support site at https://support.sciencelogic.com/s/.

Notes about installing the utility:

- The **powerflowcontrol** command-line utility requires port 22 on all host nodes.
- You can use key-based authentication instead of username and password authentication for the **powerflowcontrol** command-line utility.
- If the isadmin (host) password contains a special character, such as an "@" or "#" symbol, the password must be escaped in the **iservicecontrol** commands by adding single quotes, such as `'user:host_password'`. For example: `pfctl --host 10.10.10.100 'isadmin:testing@is' --host 10.10.10.102 'isadmin:testing@is' --host 10.10.10.105 'isadmin:testing@is' autocluster`

The **powerflowcontrol** command-line utility was updated to let any user run the **powerflowcontrol** utility. The default **isadmin** user already meets these requirements, and this update is relevant only if your PowerFlow environment uses custom users or processes.

If yourPowerFlow system is hosted in AWS, or if your company does not permit password-based SSH access, you can create a configuration object in PowerFlow to use with the **powerflowcontrol** utility. When you *create the configuration object*, specify the private key to use to SSH into each of the nodes. For example:

```
hosts:
  10.2.11.101:
    user: isadmin
    key_file: /home/ec2-user/key
  10.2.11.102:
    user: isadmin
    key_file: /home/ec2-user/testkey
  10.2.11.103:
    user: isadmin
    key_file: /home/ec2-user/testkey
```

Then you can run **pfctl** with the `--config` option:

```
pfctl --config <path_to_config> autocluster
```

User Requirements for using the powerflowcontrol (pfctl) utility

The user requirements for working with **powerflowcontrol** include the following:

- The user must belong to the **iservices** group.
- The user must belong to the **docker** group.
- The user must belong to the **systemd-journal** group, or have permission to view **journalctl** logs (to check for errors in Docker services).
- The user must have **sudo** permission (to set PowerFlow configuration file group ownership).

> IMPORTANT: The pfctl utility does not require **sudo** permission to *execute* cluster and node actions. If you do run pfctl once as **sudo**, it is expected that you would need to continue using **sudo** to modify files. ScienceLogic recommends that you interact with pfctl without **sudo**, by using a non-root user (like **isadmin**) that is part of the **iservices** group. To reset the file ownership, clear out the files from **/tmp** and re-run the pfctl utility without **sudo**, as a user that belongs to the **iservices** group.

Installing the powerflowcontrol (pfctl) utility

When updating or installing the powerflowcontrol utility, you will need to do so as the root user.

To download and install the **powerflowcontrol** utility:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **[Product Downloads]** tab and select *PowerFlow*. The **PowerFlow** page appears.
3. Click the link for the current release. The **Release Version** page appears.
4. In the **Release Files** section, click the link for the version of PowerFlow Control you want to download. The **Release File Details** page appears.
5. Click the **[Download File]** button to download the **.whl** file for the **powerflowcontrol** utility.
6. Using WinSCP or another file-transfer utility, copy the **.whl** file to a directory on the PowerFlow system.
7. Go to the console of the PowerFlow system or use SSH to access the PowerFlow system.
8. To install the utility, run the following command:

   ```
   sudo pip3 install iservicecontrol-x.x.x-py3-none-any.whl
   ```

   where `x.x.x` is the pfctl version number.
9. To check the version number of the utility, run the following command:

   ```
   pip3 show iservicecontrol
   ```

Getting Help with the powerflowcontrol (pfctl) utility

For a detailed list of all of the actions you can run on a single node, SSH to the PowerFlow server and run the following command:

```
pfctl node-action --help
```

For a detailed list of all of the actions you can run on a clustered system, run the following command:

```
pfctl cluster-action --help
```

To view updated and expanded help text, run the following command :

```
pfctl --help
```

To check the installed pfctl version, run the following command:

```
pfctl --version
```

NOTE: If you get the "Error: No such option: --version Did you mean --json?" error message when running `pfctl --version`, you might have an older version of pfctl that was installed as a different user. To resolve this, be sure to install the most recent version of the powerflowcontrol (pfctl) utility version as root with sudo, and remove any other versions installed by other users (isadmin or ec2-user):

```
su isadmin
```

```
pip3 uninstall -y iservicecontrol
```

healthcheck and autoheal

The **powerflowcontrol** (pfctl) command-line utility performs multiple administrator-level actions in a clustered PowerFlow environment. The **powerflowcontrol** utility contains automatic cluster **healthcheck** and **autoheal** capabilities that you can use to prevent issues with your PowerFlow environment:

- The ***healthcheck*** action executes various commands to verify configurations, proxies, internal connectivity, queue cluster, database cluster, indexes, NTP settings, Docker versions on all clusters, and more. Any previously reported troubleshooting issues are addressed with the healthcheck action.
- The ***autoheal*** action automatically takes corrective action on your cluster.

After deploying any clusters in a PowerFlow system, or if you are troubleshooting an existing cluster, you should first run the **healthcheck** action to generate immediate diagnostics of the entire cluster and all services and containers associated with the cluster. If the **healthcheck** action finds any issues, you can run the **autoheal** action to attempt to address those issues.

You can view the current PowerFlow version and the installed pfctl version if you add `--json` at the start of the `healthcheck` command.

healthcheck

The following commands show the formatting for a **healthcheck** action for a *single node*, followed by an example:

```
pfctl --host <pf_host_ip_address> <username>:<host_password> node-action --action healthcheck
```

```
pfctl --host 10.2.11.222 isadmin:isadmin222 node-action --action healthcheck
```

The following commands show the formatting for a **healthcheck** action for a *clustered environment*, followed by an example:

```
pfctl --host <pf_host_ip_address> <username>:<host_password> --host <host> <username>:<host_password> --host <pf_host_ip_address> <username>:<host_password> cluster-action --action healthcheck
```

```
pfctl --host 10.2.11.222 isadmin:isadmin222 --host 10.2.11.232 isadmin:isadmin232 --host 10.2.11.244 isadmin:isadminpass cluster-action --action healthcheck
```

TIP: As a best practice, run the **healthcheck** action once a day on your PowerFlow to identify and address any potential issues with the system before those issues impact operations.

Additional Features with the healthcheck Action

Starting with version 2.7.4 of the pfctl utility, the **healthcheck** node-actions and cluster-actions include the following features:

- **check_debug_run**. Checks if you have run any debug-level runs of PowerFlow applications in the past day and provides a notification if you have.
- **check_schedule_debug_enable**. Checks if you have scheduled any debug-level runs of PowerFlow applications and provides a notification if you have.

autoheal

The following commands show the formatting for an **autoheal** action for a *single node*, followed by an example:

```
pfctl --host <host> <username>:<host_password> node-action --action autoheal
```

```
pfctl --host 10.2.11.222 isadmin:isadmin222 node-action --action autoheal
```

The following commands show the formatting for an **autoheal** action for a *clustered environment*, followed by an example:

```
pfctl --host <host> <username>:<host_password> --host
<host> <username>:<host_password> --host <host> <username>:<host_password>
cluster-action --action autoheal
```

```
pfctl --host 10.2.11.222 isadmin:isadmin222 --host 10.2.11.232
isadmin:isadmin232 --host 10.2.11.244 isadmin:isadminpass cluster-action -
-action autoheal
```

Example Output

The following section lists example healthcheck output:

```
verify db host for cluster 10.2.11.222...........[OK]
check dex connectivity 10.2.11.222..............[OK]
check rabbit cluster count 10.2.11.222...........[OK]
check rabbit cluster alarms 10.2.11.222..........[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.222.............[OK]
verify cmd in container 10.2.11.232.............[Skipped - iservices_
steprunner not found on 10.2.11.232]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.232.............[Skipped - iservices_con-
tentapi not found on 10.2.11.232]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.232.............[Skipped - iservices_
steprunner not found on 10.2.11.232]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.232.............[Skipped - iservices_con-
tentapi not found on 10.2.11.232]
verify cmd in container 10.2.11.232.............[OK]
verify cmd in container 10.2.11.244.............[OK]
verify cmd in container 10.2.11.244.............[OK]
verify cmd in container 10.2.11.244.............[OK]
verify cmd in container 10.2.11.244.............[OK]
verify cmd in container 10.2.11.244.............[OK]
verify cmd in container 10.2.11.244.............[OK]
```

```
get file hash 10.2.11.222.....................[OK]
get file hash 10.2.11.232.....................[OK]
/etc/iservices/isconfig.yml does not match between 10.2.11.222 and
10.2.11.232
get file hash 10.2.11.222.....................[OK]
get file hash 10.2.11.232.....................[OK]
get file hash 10.2.11.244.....................[OK]
/opt/iservices/scripts/docker-compose.yml does not match between
10.2.11.222 and 10.2.11.244
get file hash 10.2.11.222.....................[OK]
get file hash 10.2.11.232.....................[OK]
get file hash 10.2.11.244.....................[OK]
get file hash 10.2.11.222.....................[OK]
get file hash 10.2.11.232.....................[OK]
get file hash 10.2.11.244.....................[OK]
get file hash 10.2.11.222.....................[OK]
get file hash 10.2.11.232.....................[OK]
get file hash 10.2.11.244.....................[OK]
check cpu 10.2.11.222.........................[OK]
check disk 10.2.11.222........................[OK]
check memory 10.2.11.222......................[Failed]
check cpu 10.2.11.232.........................[OK]
check disk 10.2.11.232........................[OK]
check memory 10.2.11.232......................[OK]
check cpu 10.2.11.244.........................[OK]
check disk 10.2.11.244........................[OK]
check memory 10.2.11.244......................[OK]
Utilization warnings in the cluster:
{'10.2.11.222': ['There is less than 2000mb memory available']}
verify ntp sync 10.2.11.222...................[OK]
verify ntp sync 10.2.11.232...................[OK]
verify ntp sync 10.2.11.244...................[OK]
check replica count logs 10.2.11.222..........[OK]
check replica count content 10.2.11.222.......[Failed]
Identified missing replicas on some buckets: ['Replica count for bucket:
content is not the expected 2']
verify pingable addr 10.2.11.222..............[OK]
verify pingable addr 10.2.11.232..............[OK]
verify pingable addr 10.2.11.244..............[OK]
get exited container count 10.2.11.222........[OK]
```

```
get exited container count 10.2.11.232..........[OK]
get exited container count 10.2.11.244..........[OK]
6 exited (stale) containers found cluster-wide
verify node indexes 10.2.11.222................[Failed]
Some nodes are missing required indexes. Here are the nodes with the miss-
ing indeces:
Missing the following indexes: {'couchbase.isnet': ['idx_casbin'], 'couch-
base-worker2.isnet':
['idx_content_configuration']}
```

Using powerflowcontrol healthcheck on the docker-compose file

You can also validate the **docker-compose** file with the **powerflowcontrol healthcheck** action. The action will show a message if pypiserver or dexserver services are not configured properly in the docker-compose file. You can fix these settings manually or with the **powerflowcontrol autoheal** action, which corrects the **docker-compose** file and copies it to all the nodes in the clustered environment.

When using version 1.3.0 or later of the **powerflowcontrol** (pfctl) command-line utility, the **autocluster** action validates and fixes the pypiserver and dexserver services definitions in the **docker-compose** file.

> **NOTE:** The **healthcheck** action in the **powerflowcontrol** command-line utility for PowerFlow clusters will check the Docker version for each cluster to ensure that the Docker version is the same in all the hosts.

autocluster

You can use the **powerflowcontrol** (pfctl) command-line utility to perform multiple administrator-level actions on your PowerFlow cluster. You can use the **autocluster** action with the **powerflowcontrol** command to automate the configuration of a three-node cluster.

> **NOTE:** If you are using another cluster configuration, the deployment process should be manual, because the **powerflowcontrol** utility *only* supports the automated configuration of a three-node cluster.

> **WARNING:** The **autocluster** action will completely reset and remove all data from the system. When you run this action, you will get a prompt verifying that you want run the action and delete all data.

To automate the configuration of a three-node cluster, run the following command:

```
pfctl --host <pf_host1> <username>:<host_password> --host <pf_host2>
<username>:<host_password> --host <pf_host3>  <username>:<host_password>
autocluster
```

For example:

```
pfctl --host 192.11.1.1 isadmin:passw0rd --host 192.11.1.2
isadmin:passw0rd --host 192.11.1.3 isadmin:passw0rd autocluster
```

Running this command will configure your PowerFlow three-node cluster without any additional manual steps required.

NOTE: You can use the **generate_haproxy_config** cluster-action in the **powerflowcontrol** (pfctl) utility to create an HAProxy configuration template that lets you easily set an HAProxy load balancer for a three-node cluster.

For example:

```
pfctl --host <host_IP_1> user:host_password --host <host_IP_2> user:host_
password --host <host_IP_3> user:host_password cluster-action --action
generate_haproxy_config
```

or

```
pfctl --config pfctl.yml cluster-action --action generate_haproxy_config
```

apply_<n>GB_override, verify_<n>GB_override

IMPORTANT: The actions in this topic are available in the powerflowcontrol (pfctl) utility version 2.7.4 and later.

You can use the following cluster-actions to apply and verify 16 GB, 32 GB, and 64 GB overrides to SaaS PowerFlow systems only. These actions let you control the memory allocation of the PowerFlow nodes and ensure full replication of all services in any failover scenario. In addition, when you run these actions on a SaaS PowerFlow system, the **docker-compose.yml** file is updated with deployment configurations specific to a SaaS environment.

- **apply_16GB_override** and **verify_16GB_override**. These settings support up to 25,000 to 30,000 devices, depending on the relationship depth of the devices.
- **apply_32GB_override** and **verify_32GB_override**. These settings support up to approximately 70,000 devices.
- **apply_64GB_override** and **verify_64GB_override**.

The following command is an example of a pfctl command to apply the 32 GB override:

```
pfctl --host 10.10.10.100 'isadmin:testing@is' --host 10.10.10.102
'isadmin:testing@is' --host 10.10.10.105 'isadmin:testing@is' cluster-
action --action apply_32GB_override
```

When you run the override actions listed above, the updates are applied automatically to the PowerFlow server as well as to the **docker-compose.yml** file. You do not need to redeploy the whole stack.

For more information, see *Recommended Memory Allocation of PowerFlow Nodes*.

check_dex_connectivity

> IMPORTANT: The **check_dex_connectivity** action is available in the powerflowcontrol (pfctl) utility version 2.7.10 and later.

This node action checks if the Dex server is reachable for each node of the cluster. You can use this check to verify that all the nodes in a cluster can reach the load balancer successfully.

```
pfctl cluster-action --action check_dex_connectivity
```

check_docker_service_update_status

> IMPORTANT: The **check_docker_service_update_status** action is available in the powerflowcontrol (pfctl) utility version 2.7.4 and later.

The **check_docker_service_update_status** action iterates over all the running services in PowerFlow and checks the status of the Docker service after running a `docker service update` command. You can run this action as a node-action or a cluster-action.

For example:

```
pfctl --config config.yml cluster-action --action check_docker_service_
update_status
```

In addition, you can use the **--update-parallelism** option with the `docker service update` command to along with a value of 0, to update all Docker services at once.

Use the following format for the `docker service update` command:

```
docker service update --update-parallelism <uint> <configurations_to_
update><service_name>
```

where *<uint>* is the number of replicas that you want to update in parallel. Use a value of 0 to update all Docker services at once. For example:

```
docker service update --update-parallelism 0 iservices_couchbase-worker --
env-add AUTO_REBALANCE=true
```

check_redis_maxmemory, fix_redis_maxmemory

You can use the pfctl node actions **check_redis_maxmemory** and **fix_redis_maxmemory** to verify the **MAXMEMORY** variable in the **docker-compose.yaml** file. In release of PowerFlow before 2.7.0, the PowerFlow

Redis service expected a value of **MAXMEMORY** to be passed using Docker environment variables, but the override templates in the pfctl utility supplied **maxmemory** instead.

The **check_redis_maxmemory** action determines if the Redis service requires an update to the environment variable, while the **fix_redis_maxmemory** action applies that change. The user will need to delete and recreate the Redis service after the fix action runs.

In addition, starting with PowerFlow version 2.7.0, the PowerFlow Redis image can accept either **MAXMEMORY** or **maxmemory** as an environment variable.

logcollect

Starting with PowerFlow version 2.7.0, you can use the **logcollect** action to collect relevant files and logs of a PowerFlow node and PowerFlow stack docker services to troubleshoot issues.

> **NOTE:** In previous versions of PowerFlow, the **logcollect** action was called **collect_pf_logs**; the older name will be deprecated in upcoming releases.

To collect additional logs for troubleshooting, SSH to the PowerFlow server and run the following commands.

For clustered environments, use the **logcollect cluster-action**. For example:

```
pfctl --host 10.2.11.222 isadmin:isadmin222 --host 10.2.11.232
isadmin:isadmin232 --host 10.2.11.244 isadmin:isadminpass cluster-action -
-action logcollect
```

To collect logs from each node individually, use **node-action** instead of **cluster-action**. Use **node-action** for 1-node deployments as well. For example:

```
pfctl --host 10.2.11.222 isadmin:isadmin222 --host 10.2.11.232
isadmin:isadmin232 --host 10.2.11.244 isadmin:isadminpass node-action --
action logcollect
```

> **NOTE:** Be aware that PowerFlow logs will be collected in every node, and the compressed file will be located in each of them when using **node-action**.

You can also use the following extra arguments:

- `--since` or `-s` . This argument lets you specific when Docker and journald logs will be collected. The default value is 24 hours. Only hours are accepted.
- `--includelog` or `-l`. This argument allows including extra logs. The acceptable values are `varlog`, which collects and compresses the **/var/log** directory, `healthcheck`, which collects the **pfctl healthcheck** output, and `serviceslogs` which collects Docker service logs for the services running in the PowerFlow stack. These are not included by default
- `--help`. View additional details.

You can set the following extra environment variables for further configurations:

- `PFCTL_LOGCOLLECT_ROOT_DIR`. The directory where the logs compressed file will be saved. The default value is **/tmp**. Valid existing directories should be set for this environment variable.
- `PFCTL_LOGCOLLECT_TIMEOUT`. The timeout for collecting docker services logs. Its default value is 40.

logservicescollect

If you want to collect only Docker services logs, you can use the following **node-action**. This action also takes the `--since` and `--includelog` serviceslogs arguments, and also uses the environment variables mentioned above:

```
pfctl --host 10.2.11.222 isadmin:isadmin222 --host 10.2.11.232
isadmin:isadmin232 node-action --action logservicescollect -l serviceslogs
```

open_firewall_ports

To open firewall ports for a single node, SSH to the PowerFlow server and run the following command:

```
pfctl --host <pf_host_IP_address> isadmin:<host_password> node-action --
action open_firewall_ports
```

Increasing the PowerFlow Docker Swarm Heartbeat in Cluster Environments

Starting with PowerFlow version 3.1.0, you can use the **update_swarm_heartbeat_period** and **check_swarm_heartbeat_period** cluster actions to improve control over PowerFlow Docker Swarm heartbeat configuration.

update_swarm_heartbeat_period

This cluster action increases the PowerFlow Docker Swarm heartbeat to 20 seconds. For deployments in which network disruption or timeout is present, the default timeout of five seconds can be increased using the following action:

```
pfctl --cluster-action --action update_swarm_heartbeat_period
```

When creating a new cluster using **autocluster**, the PowerFlow Docker Swarm heartbeat will be set to 20 seconds by default. The default will also be set to 20 seconds when running **autoheal**.

check_swarm_heartbeat_period

This cluster action allows you to check the current PowerFlow Docker Swarm heartbeat configuration. This action is also called when calling the **healthcheck** action.

```
pfctl --cluster-action --action check_swarm_heartbeat_period
```

password

This section covers how to encypt and change passwords in PowerFlow.

Encrypting a PowerFlow Password

To encrypt a password using the **powerflowcontrol** (pfctl) command-line utility, SSH to the PowerFlow server and run the following command:

```
pfctl password encrypt
```

This command displays the decrypted password on standard output. It does not alter the contents of **/etc/iservices/is_pass** in place, but just decrypts to **stdout**.

This command is used locally when the decrypted password is needed for certain tasks that are, in turn, remotely started. On certain systems where the password is encrypted, this involves decrypting the password using the **encryption_key** file contents. For remote nodes, the nodes must also support the same version or later of the **powerflowcontrol** (pfctl) command-line utility, as that command is executed locally.

Changing the isadmin User Password

Starting in PowerFlow version 3.0.0, you can also use the following command to update the PowerFlow Administrator (*isadmin*) user password for a node environment:

```
pfctl --host IP isadmin:host_password password set -p 'new_password'
```

You can use the following command to update the PowerFlow Administrator (*isadmin*) user password for a cluster environment:

```
pfctl --host 10.10.10.100 'isadmin:testing@is' --host 10.10.10.102
'isadmin:testing@is' --host 10.10.10.105 'isadmin:testing@is' password set
```

After the password is correctly updated, make sure the stack is removed and redeployed:

1. Remove the stack using the following command:

   ```
   docker stack rm iservices
   ```

2. Wait until the services are down. Check with the following command:

   ```
   watch docker ps
   ```

3. Redeploy the stack using the following command:

   ```
   docker stack deploy --resolve-image=never -c
   /opt/iservices/scripts/docker-compose.yml iservices
   ```

Disabling TLS Verification

To ensure optimal functionality, all PowerFlow systems must be configured to use HTTPS certificates. For more information, see *Changing the HTTPS Certificate*. When making requests to the PowerFlow system, pfctl checks will now verify the certificate associated with the system. As a result, validation of the default self-signed certificate configured for PowerFlow systems upon initial deployment and intended for temporary use will fail.

WARNING: Disabling TLS verification should only be implemented temporarily while configuring the PowerFlow system to use valid HTTPS certificates.

To allow pfctl to bypass TLS verification:

1. Use the `disable-ssl-verification` argument when executing pfctl node and cluster actions:

   ```
   pfctl --disable-ssl-verification --host <host> <username>:<host_password> node-action --action healthcheck
   ```

2. If a pfctl configuration .yml file is in use, include the `disable-ssl-verification` setting in the file. For example:

   ```
   disable_ssl_verification: 'true'
   hosts:
     10.2.11.102:
       user: isadmin
       password: password
   ```

3. Set the environment variable in the PowerFlow host using `PFCTL_DISABLE_SSL_VERIFICATION=true` :

   ```
   PFCTL_DISABLE_SSL_VERIFICATION=true pfctl init-sps
   ```

NOTE: If you see the check `verify steprunner running` fail with the message `Connection refused. Unable to get response from the URL https://PF/api/v1/steps/run?wait=30` it is likely that the TLS Certificate check failed, and needs to be addressed by skipping temporarily and applying valid HTTPS Certificates.

Chapter 12

Using SL1 to Monitor SL1 PowerFlow

Overview

This chapter describes the various ScienceLogic PowerPacks that you can use to monitor the components of the PowerFlow system. This chapter also describes the suggested settings, metrics, and situations for healthy SL1 and PowerFlow systems.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon ( ).
- To view a page containing all of the menu options, click the Advanced menu icon (•••).

This chapter covers the following topics:

Monitoring PowerFlow *241*

Configuring the Docker PowerPack *242*

Configuring the ScienceLogic: PowerFlow PowerPack *244*

Stability of the PowerFlow Platform *247*

Monitoring PowerFlow

You can use a number of ScienceLogic PowerPacks to help you monitor the health of your PowerFlow system. This section describes those PowerPacks and additional resources and procedures you can use to monitor the components of PowerFlow.

TIP: You can also use the **PowerFlow Control Tower** page in the PowerFlow user interface to monitor the status of the various tasks, workers, and applications that are running on your PowerFlow system. You can use this information to quickly determine if your PowerFlow instance is performing as expected.

You can download the following PowerPacks from the **PowerPacks & SyncPacks** page of the ScienceLogic Support Site at https://support.sciencelogic.com/s/ to help you monitor your PowerFlow system:

- **Linux Base Pack PowerPack**: This PowerPack monitors your Linux-based PowerFlow server with SSH (the PowerFlow ISO is built on top of an Oracle Linux Operating System). This PowerPack provides key performance indicators about how your PowerFlow server is performing. The only configuration you need to do with this PowerPack is to install the latest version of it.
- **Docker PowerPack**: This PowerPack monitors the various Docker containers, services, and Swarm that manage the PowerFlow containers. This PowerPack also monitors PowerFlow when it is configured for High Availability. Use version 103 or later of the Docker PowerPack to monitor PowerFlow services in SL1. For more information, see *Configuring the Docker PowerPack*.
- **ScienceLogic: PowerFlow PowerPack**. This PowerPack monitors the status of the applications in your PowerFlow system. Based on the events generated by this PowerPack, you can diagnose why applications failed on PowerFlow. For more information, see *Configuring the ScienceLogic: PowerFlow PowerPack*.

  IMPORTANT: The "ScienceLogic: PowerFlow" PowerPack is the main PowerPack that you can use to monitor the critical health of a PowerFlow system.

- **Couchbase PowerPack**: This PowerPack monitors the Couchbase database that PowerFlow uses for storing the cache and various configuration and application data. This data provides insight into the health of the databases and the Couchbase servers. For more information, see *Configuring Couchbase for Monitoring* in the SL1 Product Documentation.
- **AMQP: RabbitMQ PowerPack**. This PowerPack monitors RabbitMQ configuration data and performance metrics using Dynamic Applications. You can use this PowerPack to monitor the RabbitMQ service used by PowerFlow. For more information, see *Configuring the RabbitMQ PowerPack* in the SL1 Product Documentation.

You can use each of the PowerPacks listed above to monitor different aspects of PowerFlow. Be sure to download and install the latest version of each PowerPack.

Configuring the Docker PowerPack

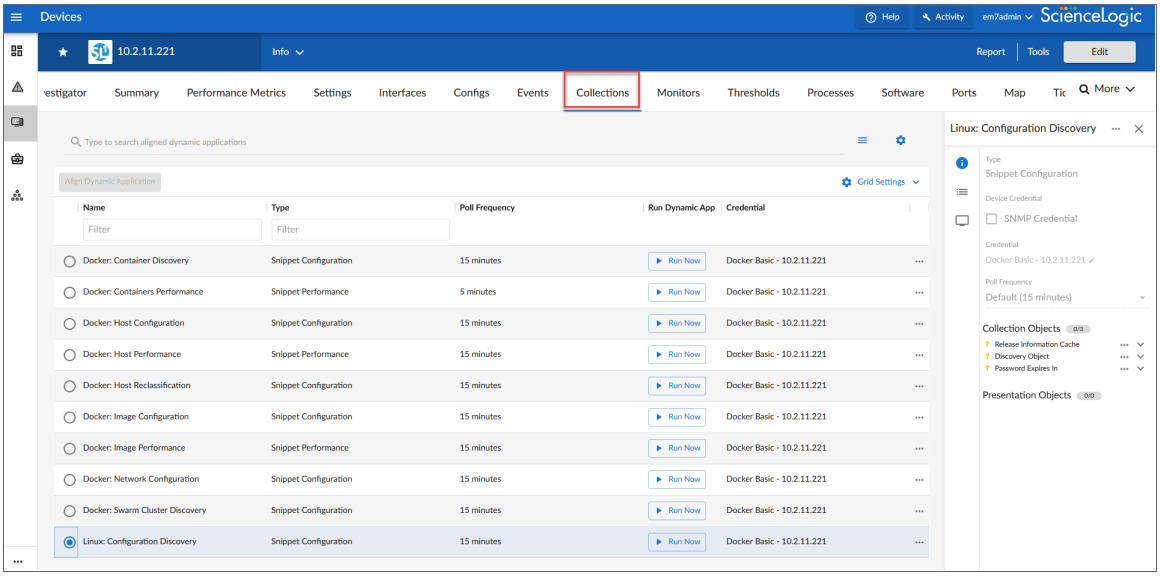
The "Docker" PowerPack monitors the various Docker containers, services, and Swarm that manage the PowerFlow containers. This PowerPack also monitors PowerFlow when it is configured for High Availability. Use version 103 or later of the Docker PowerPack to monitor PowerFlow services in SL1.

To configure the "Docker" PowerPack to monitor PowerFlow:

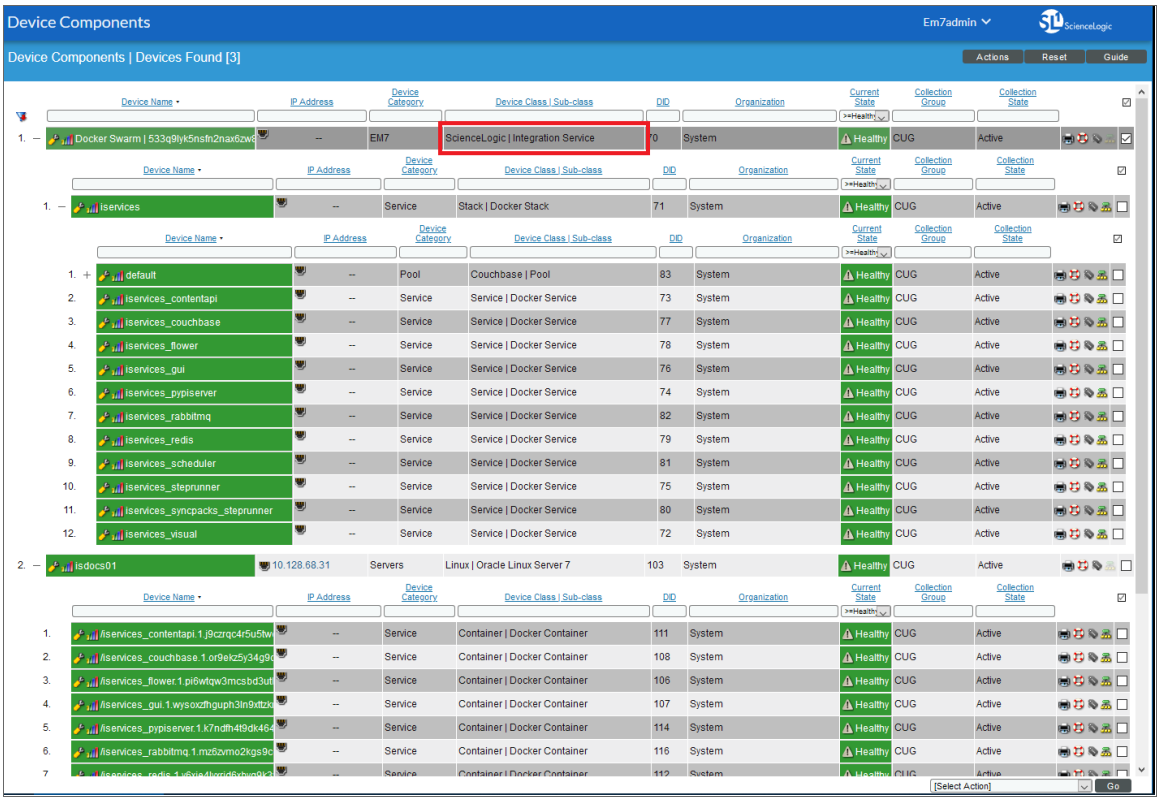
1. Make sure that you have already installed the "Linux Base Pack" PowerPack and the "Docker" PowerPack.
2. In SL1, go to the **Credential Management** page (Manage > Credentials or System > Manage > Credentials in the classic user interface) and selct the **Docker Basic - Dev ssh** credential. The **Edit Credential** page appears.
3. Complete the following fields, and keep the other fields at their default settings:
    - ***Name***. Type a new name for the credential.
    - ***Hostname/IP***. Type the hostname or IP address for the PowerFlow instance, or type "%D".
    - ***Username***. Type the username for the PowerFlow instance.
    - ***Password***. Type the password for the PowerFlow instance.
4. Click **[Save & Close]**.
5. On the **Devices** page, click **[Add Devices]** to discover your PowerFlow server using the new Docker SSH new credential.
    - Use the ***Unguided Network Discovery*** option and search for the new Docker credential on the **Choose credentials** page of the Discovery wizard. For more information, see the ***Discovery and Credentials***manual.
    - Select ***Discover Non-SNMP*** and ***Model Devices*** in the **Advanced options** section.
    - Click **[Save and Run]**. After the discovery is complete, SL1 creates a new Device record for the PowerFlow server and new Device Component records for Docker containers.
6. Go to the **Devices** page and select the new device representing your PowerFlow server.

> **NOTE:** If the Docker Swarm root device is modeled with a different device class, go to the **Devices** page and select the Docker Swarm root device. Click the **[Edit]** button on the **Device Investigator** page , click the **Info** drop-down, and edit the *Device Class* field. From the **Select a Device Class** window, select *ScienceLogic PowerFlow* as the Device Class and click **[Set Class]**. Click **[Save]** on the **Device Investigator** page to save your changes.

7. Go to the **[Collections]** tab of the **Device Investigator** page for the new device and make sure that all of the Docker and Linux Dynamic Applications have automatically aligned. This process usually takes a few minutes. A group of Docker and Linux Dynamic Applications should now appear on the **[Collections]** tab:



8. To view your newly discovered device components, navigate to the **Device Components** page (Devices > Device Components). If you do not see your newly discovered Docker Host, wait for the dynamic applications on the Docker host to finish modeling out its component devices. A Docker Swarm virtual root device will also be discovered. After discovery finishes, you should see the following devices representing your PowerFlow system on the **Device Components** page (Devices > Device Components):



NOTE: At times, the advertised host IP for a Docker node might display as "0.0.0.0" instead of the actual external address. This is a known issue in Docker. To work around this issue, remove and rejoin the nodes of the swarm one by one, and use the following argument to add them: `--advertise-addr <ip-to-show>`. For example, `docker swarm join --advertise-addr ...`. Do not remove a leader node unless there are at least two active leaders available to take its place.

Configuring the ScienceLogic: PowerFlow PowerPack

The "ScienceLogic: PowerFlow" PowerPack monitors the status of the applications in your PowerFlow system. Based on the events generated by this PowerPack, you can diagnose why applications failed in PowerFlow.

IMPORTANT: The "ScienceLogic: PowerFlow" PowerPack is the main PowerPack that you can use to monitor the critical health of a PowerFlow system.

To configure SL1 to monitor PowerFlow, you must first create a SOAP/XML credential. This credential allows the Dynamic Applications in the "ScienceLogic: PowerFlow" PowerPack to communicate with PowerFlow.

In addition, before you can run the Dynamic Applications in the "ScienceLogic: PowerFlow" PowerPack, you must manually align the Dynamic Applications from this PowerPack to your PowerFlow device in SL1. These steps are covered in detail below.

Configuring the PowerPack

To configure the PowerFlow PowerPack:

1. In SL1, make sure that you have already installed the "Linux Base" PowerPack, the "Docker" PowerPack, and the "ScienceLogic: PowerFlow" PowerPack on your SL1 system.
2. In SL1, navigate to the **Credentials** page (Manage > Credentials or System > Manage > Credentials in the classic user interface) and select the "ScienceLogic: PowerFlow Example" SOAP/XML credential. The **Edit Credential** page appears.
3. Complete the following fields, and keep the other fields at their default settings:
    - ***Name***. Type a new name for the credential.
    - ***URL***. Type the URL for your PowerFlow system.
    - ***HTTP Auth User***. Type the PowerFlow administrator username.
    - ***HTTP Auth Password***. Type the PowerFlow administrator password

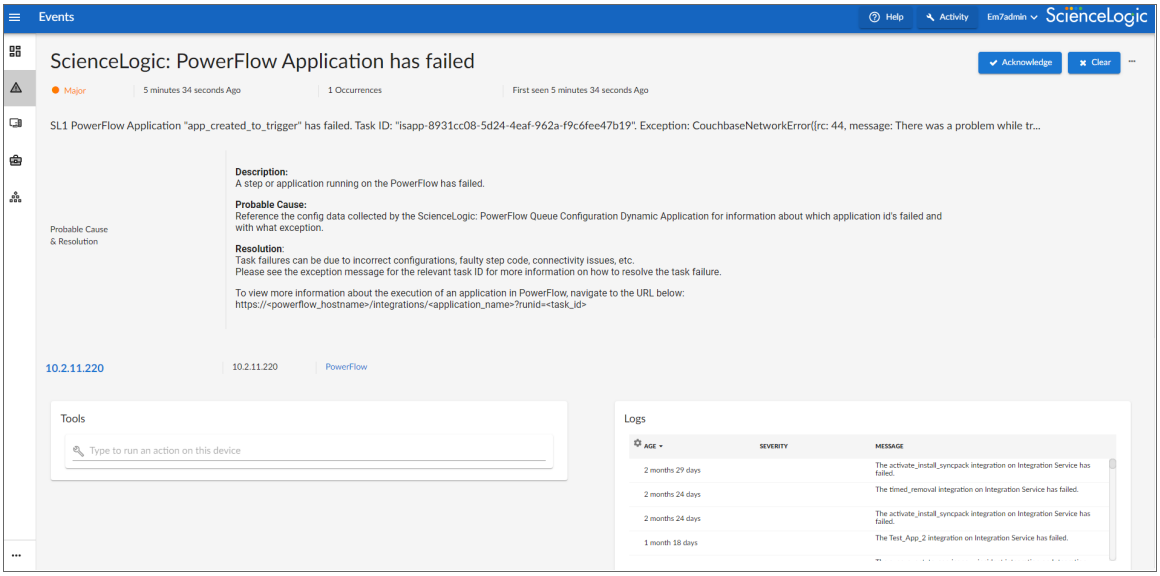
NOTE: If you upgrade the PowerPack to version 107, be sure to remove the "False" value in the Embed Value [%1] field. If this field has the "False" value populated, it will trigger a Snippet Framework error.

4. Click the **[Save & Close]** button. You will use this new credential to manually align the following Dynamic Applications:
    - ScienceLogic: PowerFlow Queue Configuration
    - ScienceLogic: PowerFlow Workers Configuration

5. Go to the **Devices** page, select the device representing your PowerFlow server, and click the **[Collections]** tab.
6. Click **[Edit]**, click **[Align Dynamic Application]**, and select *Choose Dynamic Application*. The **Choose Dynamic Application** window appears.
7. In the ***Search*** field, type the name of the first of the PowerFlow Dynamic Applications. Select the Dynamic Application and click **[Select]**.
8. Select *Choose Dynamic Application*. The **Choose Credential** window appears.
9. In the ***Search*** field, type the name of the credential you created in steps 2-4, select the new credential, and click **[Select]**. The **Align Dynamic Application** window appears.
10. Click **[Align Dynamic App]**. The Dynamic Application is added to the **[Collections]** tab.
11. Repeat steps 6-10 for each remaining Dynamic Application for this PowerPack, and click **[Save]** when you are done aligning Dynamic Applications.

Events Generated by the PowerPack

After you align the "ScienceLogic: PowerFlow Queue Configuration" Dynamic Application in SL1, that Dynamic Application will generate a Major event in SL1 if an application fails in PowerFlow:



The related event policy includes the name of the application, the Task ID, and the traceback of the failure. You can use the application name to identify the application that failed in PowerFlow. You can use the Task ID to determine the exact execution of the application that failed, which you can then use for debugging purposes.

To view more information about the execution of an application in PowerFlow, navigate to the relevant page in PowerFlow by formatting the URL in the following manner:

```
https://<PowerFlow_hostname>/integrations/<application_
name>?runid=<task_id>
```

For example:

```
https://192.0.2.0/integrations/sync_credentials?runid=c7e157ae-5644-
4161-a241-59516feeadec
```

For additional monitoring options, see *Configuring Monitoring for SL1 PowerFlow* in the SL1 Product Documentation.

Stability of the PowerFlow Platform

This topic defines what a healthy SL1 system and a healthy PowerFlow system look like, based on the following settings, metrics, and situations.

What makes up a healthy SL1 system?

To ensure the stability of your SL1 system, review the following settings in your SL1 environment:

- The SL1 system has been patched to a version that has been released by ScienceLogic within the last 12 months. ScienceLogic issues a software update at least quarterly. It is important for the security and stability of the system that customers regularly consume these software updates.
- The user interface and API response times for standard requests are within five seconds:
    - Response time for a specific user interface request.
    - Response time for a specific API request.
- At least 20% of local storage is free and available for new data. ***Free space*** is a combination of unused available space within InnoDB datafiles and filesystem area into which those files can grow
- The central system is keeping up with all collection processing:
    - Performance data stored and available centrally within three minutes of collection
    - Event data stored and available centrally within 30 seconds of collection
    - Run book automations are completing normally
- Collection is completing normally. Collection tasks are completing without early termination (sigterm).
- All periodic maintenance tasks are completing successfully:
    - Successfully completing daily maintenance (pruning) on schedule
    - Successfully completing backup on schedule
- High Availability and Disaster Recovery are synchronized (where used):
    - Replication synchronized (except when halted / recovering from DR backup).
    - Configuration matches between nodes.

What makes up a healthy PowerFlow system?

To ensure the stability of the PowerFlow system, review the following settings in your environment:

- The settings from the previous list are being met in your SL1 system.
- You are running a supported version of PowerFlow.
- The memory and CPU percentage of the host remains less than 80% on core nodes.
- Task workloads can be accepted by the API and placed onto the queues for execution.
- The PowerFlow API is responding to POST calls to run applications within the default timeout of 30 seconds. For standard applications triggers, this is usually sub-second.
- The PowerFlow Scheduler is configured correctly. For example, there are no tasks accidentally set to run every minute or every second.
- Task workloads are actively being pulled from queues for execution by workers. Workers are actively processing tasks, and not just leaving items in queue.
- Worker nodes are all up and available to process tasks.
- Couchbase does not frequently read documents from disk. You can check this value with the "Disk Fetches per second" metric in the Couchbase user interface.
- The Couchbase Memory Data service memory usage is not using all allocated memory, forcing data writes to disk. You can check this value with the "Data service memory allocation" metric in the main Couchbase dashboard.
- Container services are not restarting.
- The RabbitMQ memory usage is not more than 2-3 GB per 10.000 messages in queues. The memory usage might be a little larger if you are running considerably larger tasks.
- RabbitMQ mirrors are synchronized.
- RabbitMQ is only mirroring the dedicated queues, not temporary or TTL queues.
- All Couchbase indexes are populated on all Couchbase nodes.
- The Couchbase nodes are fully rebalanced and distributed.
- The Docker Swarm cluster has at least three active managers in a High Availability cluster.
- For any Swarm node that is also a swarm manager, and that node is running PowerFlow services :
    - At least one CPU with 4 GB of memory is available on the host to actively manage the swarm cluster.
    - Any PowerFlow services running on this host are not able to consume all of the available resources, causing cluster operations to fail.

Some of the following PowerFlow settings might vary, based on your configuration:

- The number of applications sitting in queue is manageable. A large number of applications sitting in queue could indicate either a large spike in workload, or no workers are processing.
- The number of failed tasks is manageable. A large number of failed tasks could be caused by ServiceNow timeouts, expected failure conditions, and other situations.
- ServiceNow is not overloaded with custom table transformations that cause long delays when PowerFlow is communicating with ServiceNow.

Chapter 13

Troubleshooting SL1 PowerFlow

Overview

This chapter includes troubleshooting resources, procedures, and frequently asked questions related to working with SL1 PowerFlow.

> WARNING: PowerFlow and SyncPacks content not created by ScienceLogic is *not* supported by ScienceLogic. This includes custom steps, PowerFlow applications, and SyncPacks.

This chapter covers the following topics:

Initial Troubleshooting Steps 250
Resources for Troubleshooting 250
Identifying Why a Service or Container Failed 261
Identifying Why a PowerFlow Application Failed 263
Troubleshooting Clustering and Node Failover 265
Frequently Asked Questions 268

Initial Troubleshooting Steps

PowerFlow acts as a middle server between data platforms. For this reason, the first steps should always be to ensure that there are no issues with the data platforms with which PowerFlow is talking. There might be additional configurations or actions enabled on ServiceNow or SL1 that result in unexpected behavior. For detailed information about how to perform the steps below, see *Resources for Troubleshooting*.

SL1 PowerFlow

1. Run `docker service ls` on the PowerFlow server:
    - Note the Docker container version.
    - Verify that the Docker services are running.
2. If a certain service is failing, make a note of the service name and version.
3. If a certain service is failing, run `docker service ps <service_name>` to see the historical state of the service and make a note of this information. For example: `docker service ps iservices_contentapi`.
4. Make a note of any logs impacting the service by running `docker service logs <service_name>`. For example: `docker service logs iservices_couchbase`.

ServiceNow

1. Make a note of the ServiceNow version and SyncPack version, if applicable.
2. Make a note if you are running a ServiceNow certified application or a Service Graph SyncPack.
3. Make a note of the SyncPack application that is failing in PowerFlow.
4. Make a note of what step is failing in the application, try running the application in debug mode, and capture any traceback or error messages that occur in the step log.

Resources for Troubleshooting

This section contains port information for PowerFlow and troubleshooting commands for Docker, Couchbase, and the PowerFlow API.

Useful PowerFlow Ports

- **https://<*IP of PowerFlow*>:8091**. Provides access to Couchbase, a NoSQL database for storage and data retrieval.
- **https://<*IP of PowerFlow*>:15672**. Provides access to the RabbitMQ Dashboard, which you can use to monitor the service that distributes tasks to be executed by PowerFlow workers.
- **https://<*IP of PowerFlow*>/flower/workers**. Provides access to Flower, a tool for monitoring and administrating Celery clusters.
- **https://<*IP of PowerFlow*>:3141**. Provides access to the pypiserver service. which you can use to see if SyncPacks have been correctly uploaded to Devpi container.

IMPORTANT: Port 5556 must be open for both PowerFlow and the client.

powerflowcontrol healthcheck and autoheal actions

SL1 PowerFlow includes a command-line utility called **powerflowcontrol** (pfctl) that performs multiple administrator-level actions in a clustered PowerFlow environment. The **powerflowcontrol** utility contains automatic cluster **healthcheck** and **autoheal** actions that you can use to prevent issues with your PowerFlow environment.

For more information, see *Using the powerflowcontrol (pfctl) Command-line Utility*.

Helpful Docker Commands

PowerFlow is a set of services that are containerized using Docker. For more information about Docker, see the Docker tutorial.

Use the following Docker commands for troubleshooting and diagnosing issues with PowerFlow:

Viewing Container Versions and Status

To view the PowerFlow version, SSH to your instance and run the following command:

```
rpm -qa | grep powerflow
```

To view the individual services with their respective image versions, SSH to your PowerFlow instance and run the following command:

```
docker service ls
```

In the results, you can see the container ID, name, mode, status (see the *replicas* column), and version (see the *image* column) for all the services that make up PowerFlow:

Restarting a Service

Run the following command to restart a single service:

```
docker service update --force <service_name>
```

Stopping all PowerFlow Services

Run the following command to stop all PowerFlow services:

```
docker stack rm iservices
```

Restarting Docker

Run the following command to restart Docker:

```
systemctl restart docker
```

NOTE: Restarting Docker does not clear the queue.

Viewing Logs for a Specific Service

You can use the Docker command line to view the logs of any current running service in the PowerFlow cluster. To view the logs of any service, run the following command:

```
docker service logs -f iservices_<service_name>
```

Some common examples include the following:

```
docker service logs –f iservices_couchbase
```

```
docker service logs –f iservices_steprunner
```

```
docker service logs –f iservices_contentapi
```

NOTE: *Application* logs are stored on the central database as well as on all of the Docker hosts in a clustered environment. These logs are stored at **/var/log/iservices** for both single-node or clustered environments. However, the logs on each Docker host only relate to the services running on that host. For this reason, using the Docker service logs is the best way to get logs from all hosts at once.

NOTE: The application logs stored locally on individual node filesystems can now be collected using the **powerflowcontrol** (pfctl) command-line utility **collect_pf_logs** action. For more information, see collect_pf_logs in the "Using the powerflowcontrol (pfctl) Command-line Utility" chapter of the ***SL1 PowerFlow Platform*** manual.

Clearing RabbitMQ Volume

RabbitMQ is a service that distributes tasks to be executed by PowerFlow workers. This section covers how to handle potential issues with RabbitMQ.

The following error message might appear if you try to run a PowerFlow application via the API:

```
Internal error occurred: Traceback (most recent call last):\n File
\"./content_api.py\", line 199,
in kickoff_application\n task_status = ... line 623, in _on_close\n
(class_id, method_id),
```

```
ConnectionError)\nInternalError: Connection.open: (541) INTERNAL_ERROR -
access to vhost '/'
refused for user 'guest': vhost '/' is down
```

If your PowerFlow system is in a cluster, in this situation you should scale down services, clear the volumes, and then scale them back up one by one. If your PowerFlow is a single node, you would just have on node and only **iservices_rabbitmq**.

To clear the RabbitMQ volume:

1. Run the following command to scale down all RabbitMQ services:

   ```
   docker service scale iservices_rabbitmq=0 iservices_rabbitmq2=0
   iservices_rabbitmq3=0
   ```

2. Find and clear all RabbitMQ-related volumes from a node:

   ```
   docker volume rm $(docker volume ls -q --filter name="i*rabbit*")
   ```

3. Scale back up the first RabbitMQ node. Nodes must be scaled up one node at a time:

   ```
   docker service scale iservices_rabbitmq=1
   ```

4. Wait for the first RabbitMQ node to come up in the RabbitMQ user interface (Port 15672) and connect to workers.
5. Run the following command to scale back up RabbitMQ node 2:

   ```
   docker service scale iservices_rabbitmq2=1
   ```

6. Then run the following command to scale back up RabbitMQ node 3:

   ```
   docker service scale iservices_rabbitmq3=1
   ```

If you get a message stating that the volume is in use, run the following command:

```
docker rm <id of container using volume>
```

Re-deploy PowerFlow by running the following command:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml iservices
```

> **NOTE:** Restarting Docker does not clear the queue, because the queue is persistent. However, clearing the queue with the commands above might result in data loss due to the tasks being removed from the queue.

Viewing the Process Status of All Services

Run the following command:

```
docker ps
```

Deploying Services from a Defined Docker Compose File

Run the following command:

```
docker stack deploy -c <compose-file> iservices
```

Dynamically Scaling for More Workers

Run the following command:

```
docker service scale iservices_steprunner=10
```

Completely Removing Services from Running

Run the following command:

```
docker stack rm iservices
```

Helpful Couchbase Commands

Checking the Couchbase Cache to Ensure an SL1 Device ID is Linked to a ServiceNow Sys ID

You can determine how an SL1 device links to a ServiceNow CI record by using the respective device and sys IDs. You can retrieve these IDs from the PowerFlow Couchbase service.

First, locate the correlation ID with the following Couchbase query:

```
select meta().id from logs where meta().id like "lookup%"
```

This query returns results similar to the following:

```
[
  {
    "id": "lookup-ScienceLogicRegion+DEV+16"
  },
  {
    "id": "lookup-ScienceLogicRegion+DEV+17"
  }
]
```

After you locate the correlation ID, run the following query:

```
select cache_data from logs where meta().id = "lookup-
ScienceLogicRegion+DEV+16"
```

This query returns the following results:

```
[
  {
    "cache_data": {
      "company": "d6406d3bdbc72300c40bdec0cf9619c2",
      "domain": null,
      "snow_ci": "u_cmdb_ci_aws_service",
      "sys_id": "0c018f14dbd36300f3ac70adbf9619f7"
    }
  }
]
```

Clearing the Internal PowerFlow Cache

Some applications pull information from ScienceLogic or ServiceNow and store that data in a cache stored on PowerFlow. Examples of these applications include "Cache ScienceLogic Devices using GraphQL" and "Cache ServiceNow CIs and ScienceLogic Device Classes". While not often, at time these caches could get stale and as a result, they might not get auto-cleared or updated in PowerFlow. The steps below cover how to clear the cache to force PowerFlow to re-build that cache.

Clearing the internal PowerFlow cache can be done using the command-line interface or the Couchbase user interface.

Clearing the Cache using the Command-Line Interface

1. To clear the cache using the command-line interface, you will need to first get the ID of the Couchbase container using the `docker ps` or the `docker service ls` command. Use this ID for the *`<container_id>`* value in the next step.

2. Run the following commands on the PowerFlow system:

```
docker exec -it <container_id> /bin/bash
```

```
cbq -u <username> -p <password> -e "http://<IP_of_PowerFlow>:8091"
```

```
delete from logs where log_type == "cache"
```

Alternately, you can run the following command instead:

```
docker exec -i -t $(docker ps --filter name=iservices_couchbase -q)
/bin/bash
```

Clearing the Cache using the Couchbase user interface

1. Navigate to **http://<*IP_of_PowerFlow*>:8091** and sign in using your regular PowerFlow credentials.
2. Go to the *Query* tab and run the following query:

```
delete from logs where log_type == "cache"
```

Accessing Couchbase with the Command-line Interface

If you don't have access to port 8091 on your PowerFlow instance, you can connect to the Couchbase container by using the command-line interface (CLI).

To access Couchbase by using the CLI, run the following commands:

```
docker exec -it <container_id> /bin/bash
```

```
cbq -u <username> -p <password> -e "https://<localhost>:8091"
```

Exposing Couchbase Secondary Nodes User Interfaces

You can enable the secondary Couchbase user interfaces for PowerFlow by using Dex authentication. This setup allows users to access the Couchbase user interface through any node within the Couchbase cluster. As a result, you still have the capability to check the status of the entire Couchbase cluster, even in scenarios where the primary node is down or experiencing issues. This allows you to perform necessary actions on the nodes to restore their health and ensure that your system remains operational and healthy.

To expose secondary Couchbase user interfaces:

1. In the **/etc/iservices/isconfig.yml** file, update or add the following line:

```
OPEN_SECONDARY_CB_PORTS: true
```

2. Update the **/opt/iservices/scripts/docker-compose-override.yml** file by adding the following ports to the **gui** service:

```
gui:
 ... ...
  ports:
  - published: 8100
    target: 8100
    mode: host
  - published: 8101
    target: 8101
    mode: host
```

3. Run the script **/opt/iservices/scripts/compose_override.sh** file to validate and add the changes above to the **docker-compose.yml** file.
4. If a load balancer is being used, update the load balancer configuration to route to 8100, 8101, and open the port in the load balancer.
5. Re-deploy the stack, and you will now see that couchbase-worker and couchbase-worker2 are accessible via 8100 and 8101, respectively.

```
docker stack deploy --resolve-image=never -c
/opt/iservices/scripts/docker-compose.yml iservices
```

Temporarily Exposing Couchbase and RabbitMQ User Interfaces for Troubleshooting

To troubleshoot issues with Couchbase and RabbitMQ, you can temporarily expose the user interfaces of both services without requiring Dex authentication. This is important since Dex requires Couchbase to be operational. In case of outages or manual failover scenarios, access to the Couchbase user interface is essential for executing actions on the nodes to restore stability and health.

> **WARNING:** Disable this configuration after you finish the troubleshooting process, as exposing services without going through the PowerFlow login process is not secure.

Disabling Dex Authentication

To disable Dex authentication:

1. Update the **/opt/iservices/scripts/docker-compose.yml** file by adding the `DISABLE_DEX_AUTH: 'true'` environment variable to the **gui** service. This is a temporary change, so do it only in the node (commonly the main one) from where the stack is redeployed. For example:

```
gui:
    environment:
      DISABLE_DEX_AUTH: 'true'
      ... ..
```

2. Run the following command to remove the **gui** service:

```
docker service rm iservices_gui
```

3. Redeploy the stack so the **gui** service can be recreated with the added environment variable:

```
docker stack deploy --resolve-image=never -c
/opt/iservices/scripts/docker-compose.yml iservices
```

4. To access the Couchbase user interface, use port 8091. For the RabbitMQ user interface, use port 15672. Both can be accessed directly without going through Dex authentication. Use the corresponding default credentials to access the services user interfaces.

> **NOTE:** If the Couchbase secondary nodes on ports 8100 and 8101 are configured and open in the load balancer, they will also be accessible without Dex authentication.

Temporarily Exposing Couchbase Secondary Nodes User Interface for Troubleshooting

In addition to the configuration for disabling Dex authentication, you can temporarily expose Couchbase secondary nodes through port 8091. This is useful if the primary Couchbase node becomes inaccessible or is down, and the secondary nodes user interface on ports 8100 and 8101 is not configured.

> **WARNING:** Disable this configuration after you finish the troubleshooting process, as exposing services without going through the PowerFlow login process is not secure.

1. Update the **/opt/iservices/scripts/docker-compose.yml** file by adding the `ENABLE_SECONDARY_CB_UI: 'true'` environment variable to the **gui** service. As this is a temporary change do it only in the node (commonly the main one) from where the stack is redeployed.

```
  gui:
    environment:
      ENABLE_SECONDARY_CB_UI: 'true'
      DISABLE_DEX_AUTH: 'true'
      .... ...
```

2. Run the following command to remove the **gui** service:

```
docker service rm iservices_gui
```

3. Redeploy the stack so the **gui** service can be recreated with the added environment variable:

```
docker stack deploy --resolve-image=never -c
/opt/iservices/scripts/docker-compose.yml iservices
```

4. Access the Couchbase user interface directly using port 8091 without going through Dex authentication and using the default credentials.

Useful API Commands

Getting PowerFlow Applications from the PowerFlow API

You can use the API or cURL to retrieve the application code, which is useful when you are troubleshooting potential code-related issues. You cannot access these API endpoints with a browser, but you can request these API endpoints by using an application such as Postman:

```
https://<PowerFlow>/api/v1/applications/<application_name>
```

If you do not have access to Postman, you can use cURL to get the same information.

```
curl -iku <username>:<password> -H "Accept: application/json" -H "Content-Type: application/json" -X GET https://<PowerFlow>/api/v1/applications/<application_name>
```

Creating and Retrieving Schedules with the PowerFlow API

You can define and retrieve schedules using the PowerFlow API. You can define all of these schedules in the PowerFlow user interface as well.

To create a schedule via the API, POST the following payload to the API endpoint:

`https://<PowerFlow>/api/v1/schedule`

```
{
  "application_id": "APP_ID",
  "entry_id": "SCHEDULE_NAME",
  "params": {"a":"B"},
  "schedule": {
    "schedule_info": {
      "day_of_month": "*",
      "day_of_week": "*",
      "hour": "*",
      "minute": "*",
```

```
            "month_of_year": "*"
        },
        "schedule_type": "crontab"
    },
    "total_runs": 0
}
```

You can also specify the schedule to run on a frequency in seconds by replacing the schedule portion with the following:

```
"schedule": {
        "schedule_info": {
            "run_every": FREQUENCY_IN_SECONDS
        },
        "schedule_type": "frequency"
    }
```

Diagnosis Tools

Multiple diagnosis tools exist to assist in troubleshooting issues with the PowerFlow platform:

- ***Docker PowerPack***. This PowerPack monitors your Linux-based PowerFlow server with SSH (the PowerFlow ISO is built on top of an Oracle Linux Operating System). This PowerPack provides key performance indicators about how your PowerFlow server is performing. For more information on the Docker PowerPack and other PowerPacks that you can use to monitor PowerFlow, see the "Using SL1 to Monitor SL1 PowerFlow" chapter in the ***SL1 PowerFlow Platform*** manual.
- ***Flower.*** This web interface tool can be found at the /flower endpoint. It provides a dashboard displaying the number of tasks in various states as well as an overview of the state of each worker. This tool shows the current number of active, processed, failed, succeeded, and retried tasks on the PowerFlow platform. This tool also shows detailed information about each of the tasks that have been executed on the platform. This data includes the UUID, the state, the arguments that were passed to it, as well as the worker and the time of execution. Flower also provides a performance chart that shows the number of tasks running on each individual worker.
- ***Debug Mode***. All applications can be run in "debug" mode via the PowerFlow API. Running applications in debug mode may slow down the platform, but they will result in much more detailed logging information that is helpful for troubleshooting issues. For more information on running applications in Debug Mode, see *Retrieving Additional Debug Information*.

- ***Application Logs***. All applications generate a log file specific to that application. These log files can be found at **/var/log/iservices** and each log file will match the ID of the application. These log files combine all the log messages of all previous runs of an application up to a certain point. These log files roll over and will get auto-cleared after a certain point.
- ***Step Logs***. Step logs display the log output for a specific step in the application. These step logs can be accessed via the PowerFlow user interface by clicking on a step in an application and bringing up the **Step Log** tab. These step logs display just the log output for the latest run of that step.
- ***Service Logs***. Each Docker service has its own log. These can be accessed via SSH by running the following command:

```
docker service logs -f <service_name>
```

Identifying Why a Service or Container Failed

This section outlines the troubleshooting steps necessary to determine the underlying root cause of why a service or container was restarted. For this section, we use the *iservices_redis* service as an example.

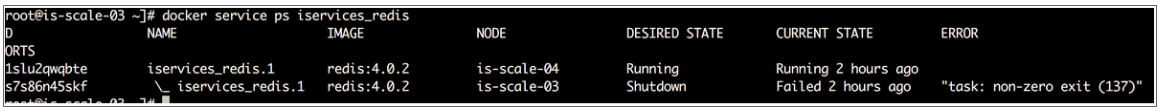
Step 1: Obtain the ID of the failed container for the service

Run the following command for the service that failed previously:

```
docker service ps --no-trunc <servicename>
```

For example:

```
docker service ps --no-trunc iservices_redis
```



From the command result above, we see that one container with the id *3s7s86n45skf* had failed previously while running on node *is-scale-03*, with the error "non-zero exit", and another container was restarted in its place.

At this point, we can ask the following questions:

- When you run `docker service ps --no-trunc`, is the error something obvious? Does the error say that it cannot mount a volume, or that the image is not found? If so, that's most likely the root cause of the issue and what needs to be addressed
- Did the node on which that container was running go down? Or is that node still up?
- Are the other services running on that node running fine? Was only this single service affected?
- If other services are running fine on that same node, it is probably a problem with the service itself. If all services on that node are not functional, it could mean a node failure.

At this point, we should be confident that the cause of the issue is not a deploy configuration issue, it is not an entire node failure, and the problem exists within the service itself. Continue to Step 2 if this is the case.

Step 2: Check for any error messages or logs indicating an error

Using the id obtained from step 1 we can collect the logs from the failed container with the following commands:

```
docker service logs <failed-id>
```

For example:

```
docker service logs 3s7s86n45skf
```

Search the service logs for any explicit errors or warning messages that might indicate why the failure occurred.

Usually, you can find the error message in those logs, but if the container ran out of memory, it may not be seen here. Continue to Step 3 if the logs provide nothing fruitful.

Step 3: Check for out of memory events

If there were no errors in the logs, or anywhere else that can be seen, a possible cause for a container restart could be that the system ran out of memory.

Perform the following steps to identify if this is the case:

1. Log in to the node where the container failed in our example. As seen in step 1, the container failed on *is-scale-03*.
2. From the node where the container failed, run the following command:

   ```
   journalctl -k | grep -i -e memory -e oom
   ```

3. Check the result for any out of memory events that caused the container to stop. Such an event typically looks like the following:

   ```
   is-scale-03 kernel: Out of memory: Kill process 5946 (redis-server)
   score 575 or sacrifice child
   ```

Troubleshooting a Cloud Deployment of PowerFlow

After completing the AWS setup instructions, if none of the services start and you see the following error during troubleshooting, you will need to restart Docker after installing the RPM installation.

```
sudo docker service ps iservices_couchbase --no-trunc
```

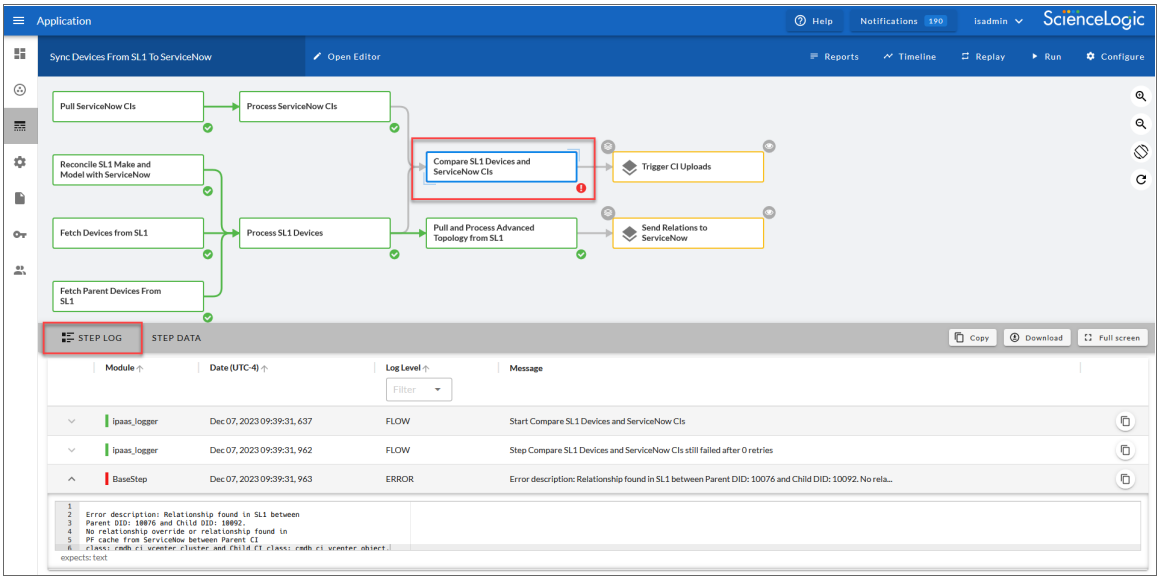
```
"error creating external connectivity network: Failed to Setup IP tables:
Unable to enable SKIP DNAT rule: (iptables failed: iptables --wait -t nat
-I DOCKER -i docker_gwbridge -j RETURN: iptables: No chain/target/match by
that name."
```

Identifying Why a PowerFlow Application Failed

The following topics describe how to determine where a PowerFlow application failed, and how to retrieve more log information related to that failure from the logs.

Determining Where an Application Failed

If a PowerFlow application fails, a red failure icon appears under that application on the **Application** detail page.



To determine where the application is failing:

1. Open the application and locate which step is failing. A failed step is highlighted in red in the image above.
2. Select the step and click the **Step Log** tab to view the logs for that step.
3. Review the error message to determine the next steps.

Retrieving Additional Debug Information (Debug Mode)

The logs in PowerFlow use the following **loglevel** settings, from most verbose to least verbose:

- ***10***. Debug Mode.
- ***20***. Informational.
- ***30***. Warning. This is the default settings if you do not specify a loglevel.
- ***40***. Error.

WARNING: If you run applications in Debug Mode ("loglevel": 10), those applications will take longer to run because of increased I/O requirements. Enabling debug logging using the following process is the only recommended method. ScienceLogic does not recommend setting "loglevel": 10 for the whole stack with the **docker-compose** file.

To run an application in Debug Mode using the PowerFlow user interface:

1. Select the PowerFlow application from the **Applications** page.
2. Hover over the **[Run]** button and select *Custom Run* from the pop-up menu. The **Custom Run** window appears.
3. Select the Logging Level. *Debug* is the most verbose and will take longer to run.
4. Specify the configuration object for the custom run in the ***Configuration*** field, and add any JSON parameters in the ***Custom Parameters*** field, if needed.
5. Click **[Run]**.

To run an application in Debug Mode using the API:

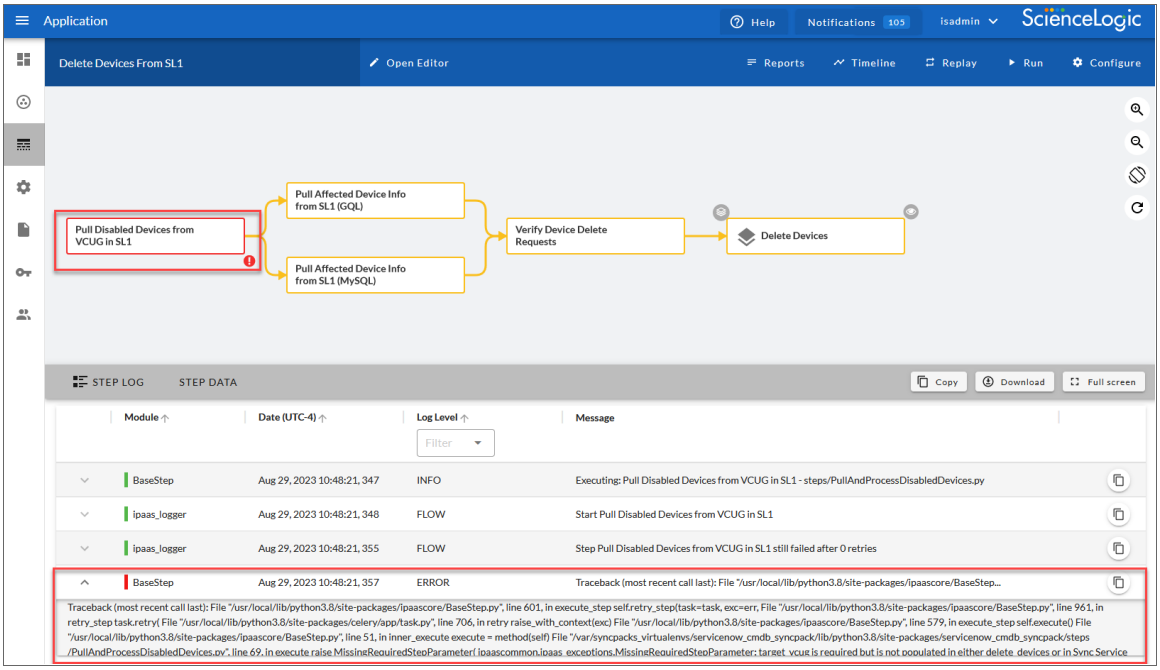
1. POST the following to the API endpoint:

```
https://<PowerFlow_IP>/api/v1/applications/run
```

2. Include the following in the request body:

```
{
  "name": "<application_name>",
  "params": {
  "loglevel": 10
  }
}
```

After running the application in Debug Mode, review the step logs in the PowerFlow user interface to see detailed debug output for each step in the application. This information is especially helpful when trying to understand why an application or step failed:



You can also run an application in debug using curl via SSH:

1. SSH to the PowerFlow instance.
2. Run the following command:

```
curl -v -k -u isadmin:<password> -X POST "https://<your_
hostname>/api/v1/applications/run"
 -H 'Content-Type: application/json' -H 'cache-control: no-cache' -d
'{"name":
"interface_sync_sciencelogic_to_servicenow","params": {"loglevel":
10}}'
```

Troubleshooting Clustering and Node Failover

This section covers how to troubleshoot a clustered environment and what to expect with a node failover in a PowerFlow system.

After a failover, Couchbase or the PowerFlow user interface are not available

In this situation, a failover occurred in a three-node cluster. Docker was stopped on the Swarm Leader, and another node in the cluster took the Swarm Leader role. After that, you could not access the PowerFlow user

interface, and Couchbase was not running.

The reason that the master of Couchbase on that node was not running was that the node was down. The other two Couchbase systems in the cluster were still running, and you should have been able to access the system through the PowerFlow user interface.

If you could not access the PowerFlow user interface, check to make sure you have a load balancer in front of the cluster. Each instance of Couchbase is "pinned" to a specific node so that it can persist data to that local node. The locally persisted data is then replicated between all three Couchbase nodes.

After a cluster or node failover, PowerFlow will not start

In this situation, you had a cluster or node failover, and now PowerFlow will not start. For example, you might have 0/3 contentapi 0/3 dexserver, or you cannot log in to the user interface. The following steps cover how to reset the database nodes and and reliably re-create indexes to resolve this issue.

> **NOTE:** These steps are for PowerFlow 2.3.0 and later.

1. Remove the stack by running the following command:

   ```
   docker stack rm iservices
   ```

2. In the **docker-compose.yml** file, add **AUTO_REBALANCE: "true"** to the **couchbase-worker** and **couchbase-worker2** services.
3. On each node, clear the indexes:

   ```
   rm -rf /var/data/couchbase/lib/couchbase/data/@2i
   ```

4. On the main node, run the following command to re-deploy the stack:

   ```
   docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
   iservices
   ```

5. Wait a few minutes, and then run `pfctl --host hostname user:host_password --host hostname2 user:host_password --host hostname3 user:host_password autoheal`.
6. If the **autoheal** action fails before it can validate and re-create indexes, create them on each node manually `pfctl --host <good-node> user:host_password --host <good-node2> user:host_password node-action --action create_node_indexes`.
7. Log in to the Couchbase user interface and validate the cluster health.

8. If one of the nodes is not shown in the Couchbase user interface:

    - If the rebalancing process with only 2 nodes is still in progress, wait for that process to finish.
    - If one of the following error messages is displayed on the missing node Docker logs, and the rebalancing process is not in progress, try forcing that service to restart so that it can try joining the cluster again using the following command:

      ```
      docker service update --force couchbase_worker_node_name
      ```

      Error messages:

      - "Error initializing couchbasedb: rc: 1 error:b'ERROR: Node addition is disallowed while rebalance is in progress"
      - "cluster_check='ERROR: Unable to connect to host at <http://localhost:8091>: HTTPConnectionPool(host='\\"localhost'\\", port=8091)"

I get a 502 error when I try to log in using the load balancer IP address

Make sure that the **HOST_ADDRESS** in the **/etc/iservices/isconfig.yml** file is pointing to that load balancer. If needed, update that value and re-deploy the PowerFlow system.

> **NOTE:** You should only use the load balancer IP when you want to log into or otherwise access the system.

Also, you should make sure that your contentapi service is properly configured in the **docker-compose.yml** file. If this service only specifies one of the Couchbase nodes, such as `db_host: couchbase.isnet,localhost`, you will not get API responses when that Couchbase node is down.

To properly configure the contentapi service in the **docker-compose.yml** file, use the following example as a template:

```
db_host:couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.isnet
```

> **TIP:** The **healthcheck** and **autoheal** actions in the **powerflowcontrol**(pfctl) command-line utility will report problems like this to you, and it will also creates relevant templates. The **healthcheck** and **autoheal** ations are supported in PowerFlow 2.0.0 and later. For more information, see *Using the powerflowcontrol (pfctl) Command-line Utility*.

After a node goes down, the SyncPacks page does not display the expected content

In this situation, the node that went down was the node that was hosting the PyPi server, which is not an operationally critical service. The PyPi server is only used to install new SyncPacks onto the system.

After a node goes down, I cannot access the db port for that instance of Couchbase :8091 directly

You will not be able to access the db port for that instance of couchbase :8091 directly. Because that node is down, the Couchbase server is not available to provide the Administrator IP. By default, only the user interface for that node is exposed, but you can expose the admin user interfaces on the other systems as well, and those admin user interfaces will be available even when the primary node goes down.

If you fail over one of the other nodes instead of that master, you will see that the Couchbase user interface stays up, because the node hosting that Couchbase instance is still up).

Couchbase fails to properly initialize or keeps trying to initialize

Couchbase might fail to properly initialize or keep trying to initialize if the Couchbase database is in a highly virtualized environment, where the CPUs are over-provisioned, and high CPU ready wait and costop times are occurring. In this situation, you will not be able to access the Couchbase user interface at **https://*<IP of PowerFlow>*:8091**.

To address this issue, make sure that the VMware environment is optimized and ready waits are minimized. For more information about CPU over-allocation in VMware, see Maximizing VMware Performance and CPU Utilization.

The system will not be as optimal as possible due to high ready wait times at the vCPU level. As a result, you will need to retry to initialize Couchbase on the first startup. Also, rebalances might need to be done multiple times to eventually succeed in over-provisioned environments.

Frequently Asked Questions

This section contains a set of frequently asked questions (FAQs) and the answers to address those situations.

> **TIP:** For additional troubleshooting information specific to multi-tenant environments, see ***Common Problems, Symptoms, and Solutions*** in the *PowerFlow for Multi-tenant Environments* appendix.

What is the first thing I should do when I have an issue with PowerFlow?

Ensure that all the services are up and running by running the following command:

```
docker service ls
```

If any service is not running, such as display a `0/1`, run the following commands:

```
docker service ps <service not running>
```

```
docker service logs <service not running>
```

Can the steprunners_syncpack service can be limited to just workers?

The syncpacks_steprunner service is responsible for keeping all SyncPack virtual environments in sync on all nodes. As you install or upgrade SyncPacks, these steprunners consistently maintain the virtual environment changes throughout the cluster without network storage. Disabling these steprunners will not affect the system operationally unless you need to update, modify, or install new SyncPacks onto the nodes of a system.

What is the difference between the steprunner_syncpacks and the steprunner services?

The syncpacks_steprunners accept no other tasks, beside create and update tasks for the virtual environment.

What is the minimal image required for workers?

At a minimum, worker nodes that will only be used to process tasks will need just the worker image.

If the GUI server is constrained to use only the manager nodes, do the worker nodes need to have their isconfig.yml file updated with the correct HOST value?

The **isconfig.yml** file is applied to all nodes in the cluster when the stack is deployed. The config used is created from the config on the system where you actually run Docker stack deploy and applied to all other nodes. In other words, only the managers from which you are deploying the stack need to have the **isconfig.yml** value correct.

Can I unload unwanted images from a worker node?

Removing unwanted images is part of typical Docker operations, and it uses the following command: `docker image rm <image name>`

For more information, see https://docs.docker.com/engine/reference/commandline/image_rm/.

If I dedicated workers to one SL1 stack, how are jobs configured to run only on those workers?

You create a worker service with the **user_queues** variable set, with a list of queues that you want that worker to process only. The queues are auto-created. That worker service can also be pinned to run on specific nodes. You can tell an integration to run on a specific queue at runtime, on a schedule, or in the application configuration. The queue will be processed by the workers that you previously defined. If no workers are listening to the queue, the task will not process.

Approximately how much data is sent between distributed PowerFlow nodes?

The amount of data sent between distributed PowerFlow nodes largely depends on the applications that are currently running, the size of the syncs for those applications, and how much of the cache those syncs use. In general, only configuration files, cache, and some logs are stored or replicated between database nodes, while

the queue service mirrors messages between its nodes as well. As a result, not a lot of information is being replicating at any given time, as only necessary cluster and vital data are replicated.

Why can't I find a SyncPack on the SyncPacks page?

If you have uploaded a SyncPack, but you cannot see it on the SyncPack page, make sure that the SyncPack filter is showing all SyncPacks. By default, the **SyncPacks** page will only show SyncPacks that are activated. An *activated* SyncPack has been installed and is ready to be used.

Click the Filter icon ( ) at the top right of the **SyncPacks** page and enable the *Toggle Inactive SyncPack* toggle to show all of theSyncPacks.

Why can't I see or upload a SyncPack?

If your PowerFlow system uses self-signed certificates, you will need to manually accept the certificate before you can upload SyncPacks.

Go to **https://*<IP address of PowerFlow>*:3141/isadmin**, accept the certificate, and then exit out of the tab. When you log into PowerFlow again, you will be able to upload SyncPacks.

Also, if a SyncPack failed to load or activate, a red exclamation mark icon ( ) appears next to that SyncPack on the **SyncPacks** page. Click the red icon ( ) to view errors logs related to the loading or activation process.

TIP: The most common error that occurs when installing a SyncPack is that the SyncPack dependencies are not installed. Review the "System Requirements" section of the release notes for that SyncPack to ensure that you have installed all of the required applications for that SyncPack. To view a list of additional PowerFlow and SL1 products that are required by the various SyncPacks, see *Dependencies for SL1 PowerFlow SyncPacks*.

Why do I get a "Connection error" message when I try to install the System Utils SyncPack?

In this situation, you get the following error message:

```
Connection error, please verify that the steprunners can request data from
pypiserver and that the syncpack system_utils_syncpack==1.0.1 uploaded to
the local pypiserver is a valid syncpack.
```

To address this issue, review the following checklist

- Review the logs for the "Activate & Install SyncPacks" PowerFlow application for relevant error details.
- Make sure that the cluster nodes are not offline.
- If the error is related to the "Activate & Install SyncPacks" application, make sure that the SyncPack that you uploaded to pypi does not have extra characters at the end, such as a "(1)" or anything else that does not follow the correct name syntax.

- If the failure is related to the "Activate & Install SyncPacks" application, then the environment might be preventing the connection, such as the `no_proxy` setting in the **docker-compose.yml** file.
- Make sure that the pypiserver has a `pypiserver.isnet` alias in the **docker-compose.yml** file.
- Validate all of the required settings in the **docker-compose.yml** file, including `no_proxy` and Couchbase database settings.

The following is an example of a portion of a **docker-compose.yml** file for this issue:

```
syncpacks_steprunner:
                        .... ...
                        environment:
                        ... ....
                        db_host: couchbase.isnet,couchbase-worker.isnet,couchbase-worke
                        http_proxy: http://is.cms.com:8080/
                        https_proxy: https://is.cms.com:8080/
                no_proxy: .isnet, 172.1.1.1/16
```

How can I optimize workers, queues, and tasks?

PowerFlow uses Celery to spawn and manage worker processes and queues. You can define environment variables to optimize these worker processes.

You can use the following environment variables to optimize worker processes:

- ***broker_load_from_backend***. Lets you send a task ID to the broker and load the task data from redis. If you set this variable to *True*, Celery will save the payload it would have sent through the broker (RabbitMQ, SQS) and sends it to the backend (Redis) instead, reducing the size of the message. When set to False, Celery will work as it normally does. The default value for this setting is *False*.

  IMPORTANT: If the `broker_load_from_backend` variable is set in the **docker-compose.yml** file, it must be set in `contentapi` and all `steprunner` services (including custom steprunners, and `syncpacks_steprunner`), as well as the `scheduler` service where relevant.

- ***task_soft_time_limit***. Enforces global timeout for all tasks in the PowerFlow system. If a task exceeds the specified timeout limit, the PowerFlow system terminates the task so that the next task in the queue can be processed. Possible values are:
  - Integer that specifies the time in seconds.
  - Default value is "3600" (1 hour).
- ***optimization***. Determines how tasks are distributed to worker processes. Possible values are:
  - *-Ofair*. Celery distributes a task only to the worker process that is available for work.
  - " " (double-quotation mark, space, double-quotation mark). Distributes and queues all tasks to all available workers. Although this increases performance, tasks in queues might be delayed waiting for

long-running tasks to complete.

- Default value is *-Ofair*.

- ***task_acks_late***. Specifies that if a worker process crashes while executing a task, Celery will redistribute the task to another worker process. Possible values are:
  - *True*. Enables the environment variable.
  - *False*. disabled the environment variable.
  - Default value is "False".

> **NOTE**: Because many applications run at regular intervals or are scheduled, the PowerFlow system re-executes tasks even if the ***task_acks_late*** environment variable is disabled. in the event of a worker crash, if you want to ensure that tasks are completed, you can enable the ***task_acks_late*** variable. However, be aware that if tasks are not idempotent, the ***task_acks_late*** variable can cause unpredictable results.

- ***max_result_size***. Organizes the step result data into chunks of the specified size, in MB. Enabled by default. Possible values are:
  - *Any number greater than 0, but less than 512*. The recommended value is 250 MB. The value should by specified in bytes, such as "250000000" for 250 MB.
  - *0*. Turns off this option.
  - Default value is "250000000" (for 250 MB).

To define these environment variables:

1. Either go to the console of the PowerFlow system or use SSH to access the server.
2. Log in as ***isadmin*** with the appropriate password.
3. Use a text editor like vi to edit the file **/opt/iservices/scripts/docker-compose-override.yml**.

> **NOTE:** PowerFlow uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be reapplied when the **/opt/iservices/scripts/docker-compose.yml** file is replaced on an RPM upgrade. This ensures that no user-specific configurations are lost. By default, only core services are included in the **docker-compose-override.yml** file. If you want to include extra services, add them as needed.

4. The services with names that start with `steprunner` are the workers for PowerFlow. Most of the configurations mentioned above need to be applied only to steprunners, but if the configuration mentions that the environment variable also needs to be applied to other services, such as `contentapi`, `scheduler`, and `syncpacks_steprunner`, you should also update those services. To define the optimization variables, enter the variables and values in the definition of the worker, under the ***environment*** section. See the example below to see the syntax for environment variables.

```
steprunner:
```

```
    environment:
      additional_worker_args: ' --max-tasks-per-child 1 '
      broker_url: pyamqp://guest@rabbit//
      db_host: couchbase.isnet,localhost
      logdir: /var/log/iservices
      result_backend: redis://redis:6379/0
      PIP_CONFIG_FILE: /usr/tmp/pip.conf
      broker_load_from_backend: 'true'
  steprunner_custom:
    environment:
      .... ..
      LOGLEVEL: 10
      celery_log_level: 10
      task_soft_time_limit: 30
      optimization: '-Ofair'
      task_acks_late: 'False'
```

5. Save the settings in the **/opt/iservices/scripts/docker-compose-override.yml** file and then run the **/opt/iservices/scripts/compose_override.sh** script.

> **NOTE:** The **compose_override.sh** script validates that the configured **docker-compose.yml** and **docker-compose-override.yml** files have the correct syntax. If the settings are correct, the script applies the settings to the existing **docker-compose.yml** file that is used to actually deploy.

6. Remove only the updated services for a faster update process:

```
docker service rm iservices_contentapi iservices_steprunner
iservices_steprunner_custom
```

7. After you have updated the **docker-compose** file, you can update and re-deploy PowerFlowthe PowerFlow system to pick up your changes to the file. To do this, execute the following command:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

> **NOTE:** If you did not remove the services referenced in step 6six, the stack may take some time to apply the changes to the services.

8. After the services are up and running and have the expected replicas, verify that the updated services have the environment variables in place by executing the following command

```
docker exec -u root -i -t $(docker ps -q -n 1 --filter name=<name of
service>) env
```

where `<name of service>` is the service you want to verify.

The output should include the environment variables defined in the **docker-compose** file.

Why do I get a "Connection refused" error when trying to communicate with Couchbase?

If you get a "Connection refused to Couchbase:8091" error when you are trying to communicate with Couchbase, check the firewalld service by running the following command:

```
systemctl status firewalld
```

Firewalld is responsible for all of the internal communications between the various Docker services on the Docker Swarm. If firewalld is not active, there will be no communications between the services, and you might see an error like "Connection refused to Couchbase:8091".

To start the firewalld service, run the following command:

```
systemctl start firewalld
```

Why do I have client-side timeouts when communicating with Couchbase?

If you are running an intensive application, or if you are running in Debug Mode, you might see the following stack trace error:

```
(generated, catch TimeoutError): <RC=0x17[Client-Side timeout exceeded for
operation. Inspect network conditions or increase the timeout], HTTP
Request failed. Examine 'objextra' for full result, Results=1, C Source=
(src/http.c,144), OBJ=ViewResult<rc=0x17[Client-Side timeout exceeded for
```

```
operation. Inspect network conditions or increase the timeout],
value=None, http_status=0, tracing_context=0, tracing_output=None>,
Tracing Output={":nokey:0": null}>
```

This error occurs when there is too much load going into the Couchbase database. If you're running with Debug Mode, that mode creates a large number of extra log messages in the database, which can contributeto this error.

To work around this issue, increase the timeout being used by setting the *db_host* environment variable in the steprunner service:

```
db_host: 'couchbase.isnet,localhost?n1ql_timeout=100000000.00'
```

If you increase the timeout, the timeout errors should go away.

> **NOTE:** Increasing the timeout might not always be the correct action. If you are using an especially large system, you might want to allocate additional resources to the Couchbase services, including more memory for indexing and search. If you are encountering timeouts in a non-temporary fashion, such as only running Debug Mode for an application to determine what went wrong, you might want to add more resources instead of increasing the timeout.

What should I do if the Couchbase disk is full, the indexer is failing, and the database is unusable?

If the Couchbase database stops unexpectedly and the disk is full:

1. Check the size of the **/var/data/couchbase/lib/couchbase/data/@2i** directory.
2. If the contents of this directory are causing the disk to fill up, stop the couchbase service and remove the **@2i** directory.
3. Restart Couchbase to return the PowerFlow system to normal operations.

Use the following optimizations to ensure that the indexer directory **@2i** does not overuse disk space:

- When running PowerFlow in production, do not run applications in Debug Mode by default.
- Remove any **#primary** index that exists in content or logs. Primary indexes are useful for troubleshooting, but they should not be used in production. In prior versions of PowerFlow, the primary index was automatically created, which caused no problems with typical workloads. However, with large workloads, the primary index adds unnecessary overhead that might impact the system. ScienceLogic recommends removing primary indexes that exist on systems in production.
- Qualys scans running against a whole cluster at once have been known to affect the performance of the database, and these scans have caused crashes resulting in index corruption in the past. If you plan on running Qualys scans on the PowerFlow system, run them at an off-peak time, and only on one node of the cluster.

- Under heavy workloads, the default auto-compaction settings might occur too frequently, causing issues due to heavy operations and compaction activities running at the same time. To reduce this possibility, select a specific day and time of the week for the database compaction to occur. Ideally compaction is configured to occur during a time of lower workloads, and not at peak. You can configure these settings in the Couchbase user interface (Settings > Auto-Compaction), or through the API: Configure Auto-Compaction with the CLI.

> **NOTE:** Needing to change auto-compaction defaults might be an early indication that the PowerFlow system needs additional resources with more load. For more information, see the following list of workload considerations.

Sizing the Couchbase database for workload considerations:

- **indexer memory**. Under heavy workloads, the indexer might be constrained and causing a bottleneck in the system due to its default 500 MB memory allocation. In real-time you can monitor the index memory percentage used through the Couchbase Administrator user interface (https://<*IP of PowerFlow*>:8091). If memory is constantly at 90% or greater, the system might benefit from an increase in memory allocation to the indexer service (Settings > Service Memory Quotas).
- Make sure that VMotion is disabled in VMware, and make sure that PowerFlow nodes will not be VMotioned while running.
- **db disk size**: Depending on the workload being performed and how much data is being saved into the system via logs, you might need to increase the database disk space. In general, you can expect the database to use at most twice the size of a bucket fully compacted. For more information, see Indexes size and compaction. The disk size available to the Couchbase system should be sufficient for whatever size workloads are expected.

What causes a Task Soft Timeout?

The following error might occur when you see a long-running task fail:

```
raise SoftTimeLimitExceeded() SoftTimeLimitExceeded: SoftTimeLimitExceeded
()
```

This error message means that the default timeout for a task on your PowerFlow system is too low. By default the task timeout, which is set by the environment variable *task_soft_time_limit*, is set to 3600 seconds (1 hour).

If you intend to have tasks executing for longer than an hour at a time, you can increase this setting by changing the *task_soft_time_limit* environment variable in your steprunners. Note that the value is set in seconds.

How do I address an "Error when connecting to DB Host" message when access is denied to user "root"?

In this situation, you get an error similar to the following:

```
Error when connecting to DB Host... (1045, "Access denied for user
'root'@'10.86.21.224' (using password: NO)"
```

This issue occurs when the **encryption_key** file from one PowerFlow system does not match the **encryption_key** file on another system, such as when you take data from a production PowerFlow system to a test PowerFlow system.

The encryption key must be identical between two PowerFlow systems if you plan to migrate from one to another. The encryption key must be identical between High Availability or Disaster Recovery systems as well.

To address this issue:

1. Copy the **encryption_key** file from the **/etc/iservices/** folder on the production system to the **/etc/iservices/** folder on the test system.
2. Re-upload all applications and configurations from the production system to the test system.
3. Remove and redeploy the stack on the test PowerFlow after copying the key by running the following command on the test system:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

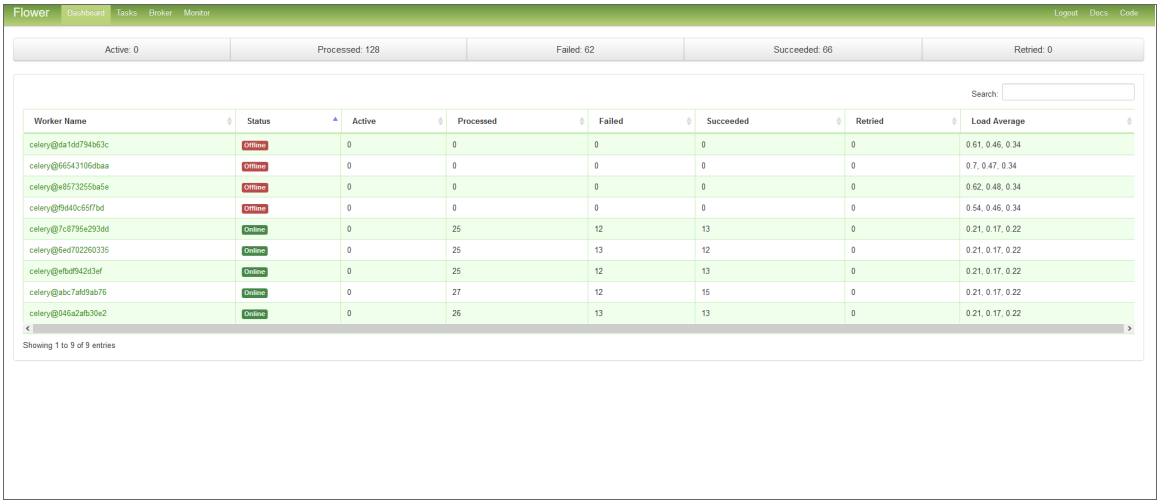
How do I identify and fix a deadlocked state?

If PowerFlow appears to be running, but it is not processing any new applications or tasks, then PowerFlow could be in a ***deadlocked*** state.

A deadlocked state occurs when one or more applications include steps that are either ordered improperly or that contain syntax errors. In this situation, tasks are waiting on subsequent tasks to finish executing, but the worker pool is exhausted. As a result, PowerFlow is not able to execute those subsequent tasks.

To identify a deadlocked state with a PowerFlow system:

1. Navigate to the Celery Flower interface for PowerFlow by typing the URL or IP address for your PowerFlow and adding **/flower/workers** at the end of the URL, such as **https://192.0.2.0/flower/workers**.
2. Click the **[Dashboard]** tab for Flower. A list of workers appears:



3. Review the number of active or running tasks for all workers. If all workers have the maximum number of tasks, and no new tasks are being consumed, then you might have a deadlock state.

To fix a deadlocked state in a PowerFlow system, perform one of the following steps:

1. Go to the console of the PowerFlow system or use SSH to access the server.
2. Log in as ***isadmin*** with the appropriate password.
3. Increase the number of workers by either:

   A. Running the following command at the shell prompt:

   ```
   docker service scale iservices_steprunner=x
   ```

   where *x* is the number of workers.

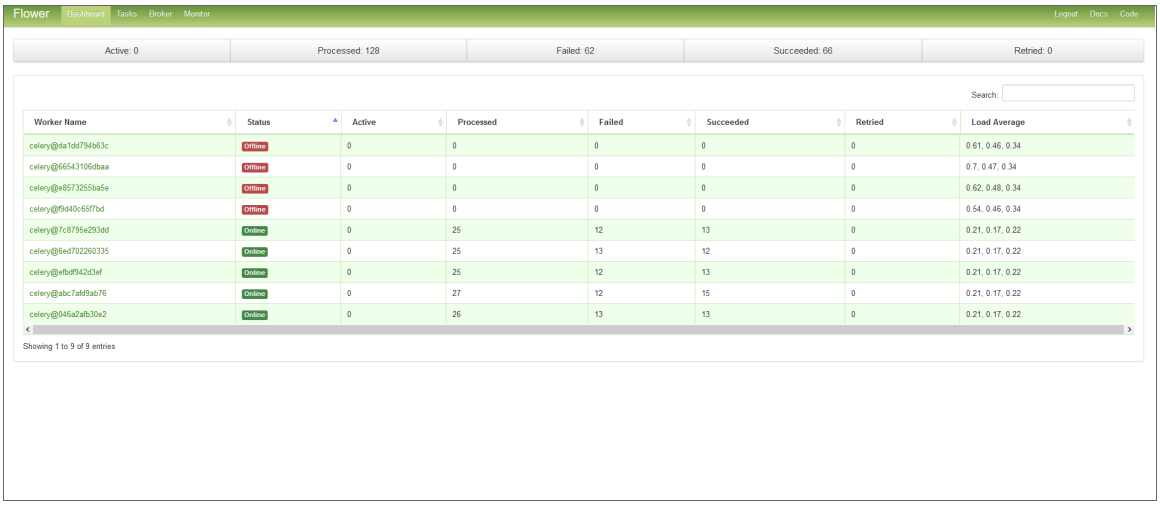
   B. Using a text editor like vi to edit the file **/opt/iservices/scripts/docker-compose.yml**. In the **environment:** section at the top of the file, add the following:

   ```
   worker_threads: number_greater_than_3
   ```

   where `number_greater_than_3` is an integer greater than 3.
4. After you have updated the docker-compose file, you can update and re-deploy the PowerFlow system to pick up the changes in the docker-compose.yml file To do this, execute the following at the shell prompt:

   ```
   docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
   iservices
   ```

   The PowerFlow system should now include additional workers.
5. Navigate to the Celery Flower interface for PowerFlow by typing the URL or IP address for PowerFlow and adding **/flower/workers** at the end of the URL, such as **https://192.0.2.0/flower/workers**.
6. Click the **[Dashboard]** tab for Flower. A list of workers appears:



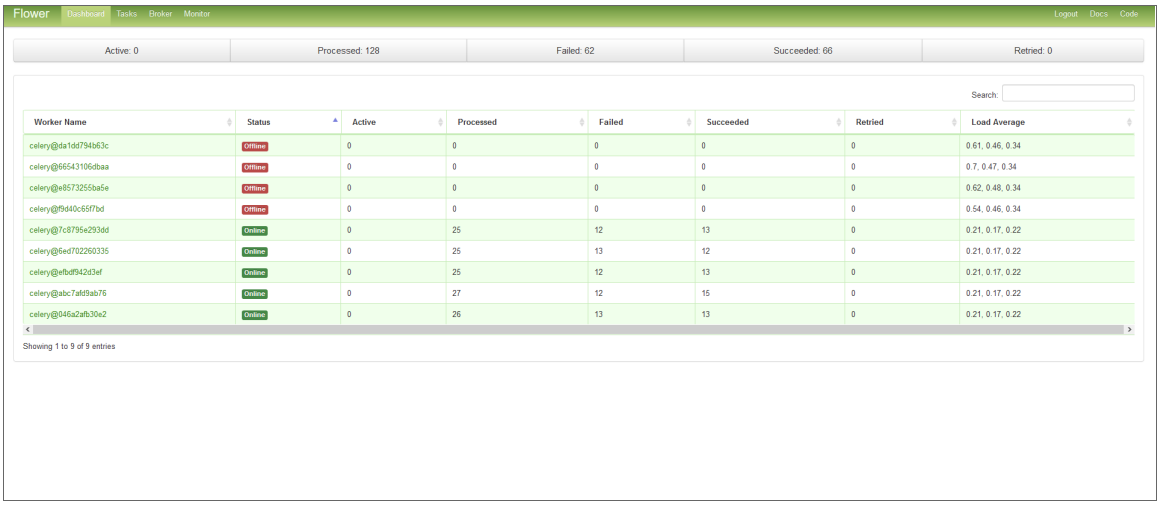
7. Review the number of active or running tasks for all workers.

If PowerFlow appears to be running, but it is not processing any new applications or tasks, then PowerFlow could be in a ***deadlocked*** state.

A deadlocked state occurs when one or more applications include steps that are either ordered improperly or that contain syntax errors. In this situation, tasks are waiting on subsequent tasks to finish executing, but the worker pool is exhausted. As a result, PowerFlow is not able to execute those subsequent tasks.

To identify a deadlocked state with a PowerFlow system:

1. Navigate to the Celery Flower interface for PowerFlow by typing the URL or IP address for your PowerFlow and adding **/flower/workers** at the end of the URL, such as **https://192.0.2.0/flower/workers**.
2. Click the **[Dashboard]** tab for Flower. A list of workers appears:



3. Review the number of active or running tasks for all workers. If all workers have the maximum number of tasks, and no new tasks are being consumed, then you might have a deadlock state.

To fix a deadlocked state in a PowerFlow system, perform one of the following steps:

1. Go to the console of the PowerFlow system or use SSH to access the server.
2. Log in as ***isadmin*** with the appropriate password.
3. Increase the number of workers by either:

   A. Running the following command at the shell prompt:

   ```
   docker service scale iservices_steprunner=x
   ```

   where *x* is the number of workers.

   B. Using a text editor like vi to edit the file **/opt/iservices/scripts/docker-compose.yml**. In the **environment:** section at the top of the file, add the following:

   ```
   worker_threads: number_greater_than_3
   ```

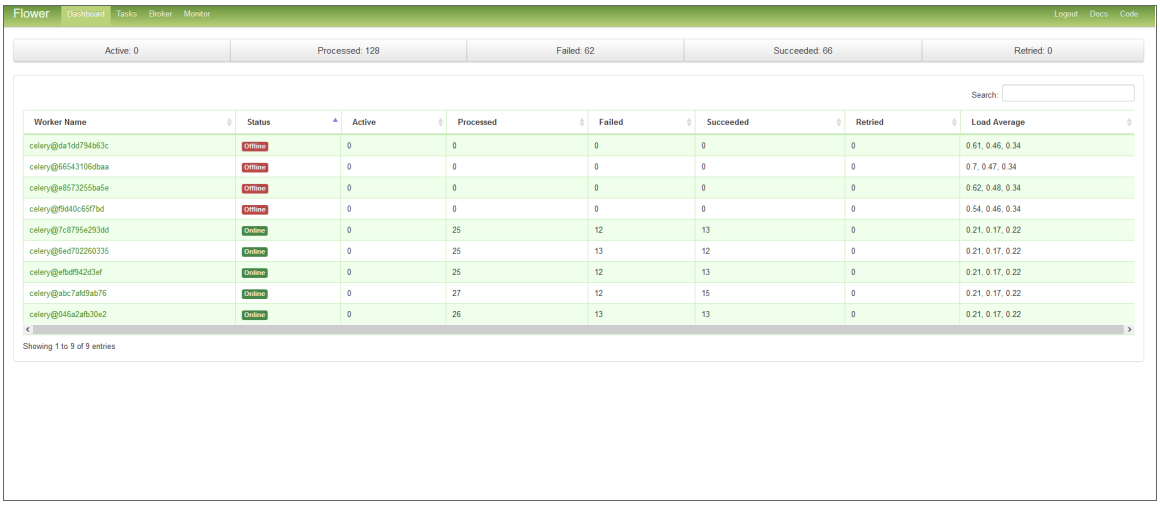
   where `number_greater_than_3` is an integer greater than 3.

4. After you have updated the docker-compose file, you can update and re-deploy the PowerFlow system to pick up the changes in the docker-compose.yml file To do this, execute the following at the shell prompt:

   ```
   docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
   iservices
   ```

   The PowerFlow system should now include additional workers.

5. Navigate to the Celery Flower interface for PowerFlow by typing the URL or IP address for PowerFlow and adding **/flower/workers** at the end of the URL, such as **https://192.0.2.0/flower/workers**.
6. Click the **[Dashboard]** tab for Flower. A list of workers appears:



7. Review the number of active or running tasks for all workers.

How can I point the "latest" container to my latest available images for PowerFlow?

If you force-upgraded an RPM on top of an existing PowerFlow RPM of the same version, and you have custom worker types pointing to specific images, the *latest* tag gets created incorrectly.

To address this issue:

1. Modify the **docker-compose.yml** file and update all SL1 images to point to the correct version that you expect to be latest.
2. Change any custom worker or custom services using SL1 containers to point to: *latest*.
3. Re-install the RPM of the same version via force.

Why does the "latest" tag not exist after the initial ISO installation?

This situation only affects users with custom services that point to the latest tag. To work around this issue, run the tag latest script manually after running the `./pull_start_iservices.sh` command:

```
python /opt/iservices/scripts/system_updates/tag_latest.py
/opt/iservices/scripts/docker-compose.yml
```

How do I address permissions errors with SyncPack virtual environments?

This situation is relevant only for upgrades to PowerFlow version 2.3.0. Permission errors might occur if you restarted Docker before the stack was redeployed, using the old stack services definition. In this situation:

- The permission that was properly set during the RPM installation for PowerFlow version 2.3.0 was reset to root. This change was not expected for PowerFlow version 2.3.0, because containers were rootless starting with version 2.3.0 for more security.
- The PowerFlow version 2.3.0 Docker containers (rootless users) cannot recreate the syncpacks virtual environments, because containers in versions of PowerFlow before 2.3.0 used root as user.

To address this issue, you need to force the steprunners to recreate the syncpacks virtual environments:

1. Run the following commands:

```
pfctl --host pf-node-ip '<username>:<host_password>' node-action --
action modify_iservices_volumes_owner
```

```
docker service update --force iservices_syncpacks_steprunner
```

2. Verify that the syncpacks virtual environments owner is not root and that the environments were recently recreated:

```
sudo ls -la /var/lib/docker/volumes/iservices_syncpacks_virtualenvs/_
data/sudo ls -la /var/lib/docker/volumes/iservices_syncpacks_
virtualenvs/_data/
```

3. If the syncpacks virtual environments were not recreated, perform steps 4-5.

> **WARNING:** Use caution while performing these next two steps.

4. Remove the syncpacks virtualenvironments volume on every node:

```
sudo rm -r /var/lib/docker/volumes/iservices_syncpacks_virtualenvs/_data/*
```

5. Restart the **syncpacks_steprunners** service using either of the following actions:

```
docker service update --force iservices_syncpacks_steprunner
```

or

```
docker service rm iservices_syncpacks_steprunner
```

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml --resolve-image never iservices
```

How do I address intermittent user access when using single sign-on?

Prior to version 3.1.1 of PowerFlow, you might experience intermittent access issues when using SSO. There are multiple options to address this issue:

1. Upgrade your PowerFlow system to version 3.1.1 or later. This will permanently resolve the issue.
2. If you are on PowerFlow version 2.7.0 or later, you can make a get request to the `/api/v1/roles/reload` API endpoint.
3. If you are on a version prior to 2.7.0, you must restart the API service using the following command:

```
docker service update --force iservices_contentapi
```

How do I keep from losing incidents or events if my PowerFlow system is down?

To keep from losing incidents or events, make sure that the Run Book Automation Retry Queue is enabled. This feature is available in PowerFlow version 2.3.0 or later and SL1 version 11.1.0 or later. For more information, see *Enabling Run Book Automation Queue Retries*.

For ServiceNow Events, Incidents, and Cases, you can also manually request that tickets be created for any events that did not get picked up. For more information, see *Configuring the "ServiceNow: Click to Create" Automation Policy*.

How do I restore an offline backup of my PowerFlow system?

To completely restore a PowerFlow system from an existing backup using a fresh PowerFlow installation, copy the following files and make sure that they match what existed on the previous PowerFlow system:

- **/etc/iservices/encryption_key**. This file ensures that the restored data can be decrypted as it was on the previous system.
- **/etc/iservices/is_pass**. Use this file if you wish to re-use the same password from the old system.
- **/etc/iservices/isconfig.yml**. This file contains authentication related settings, if used, and it sets the hostname to the load balancer, if it's a clustered environment.
- **/opt/iservices/scripts/docker-compose-override.yml**. This file contains your PowerFlow container versions and environment settings.
- **/opt/iservices/scripts/docker-compose.yml**. This file contains your PowerFlow container versions and environment settings.

The swarm cluster should have the same amount of nodes and node labels applied as in the previous system to ensure identically matching Docker environments. All nodes in the cluster must have the PowerFlow images in your **docker-compose.yml** file loaded and available (use **[docker image ls]**).

After the PowerFlow system is started, you can run the "PowerFlow Restore" application to restore all previously used applications, cache, and config data. For more information, see *Backing up Data*.

NOTE: If the workers/api are now running on completely new nodes after the restore, you will have to re-install the SyncPacks from the PowerFlow user interface to install the environments on the new nodes. Applications from those SyncPacks will already be configured with the restored settings.

NOTE: This process is not considered a "Passive Disaster Recovery failover scenario. Active/Passive relationships between PowerFlow clusters is not supported.

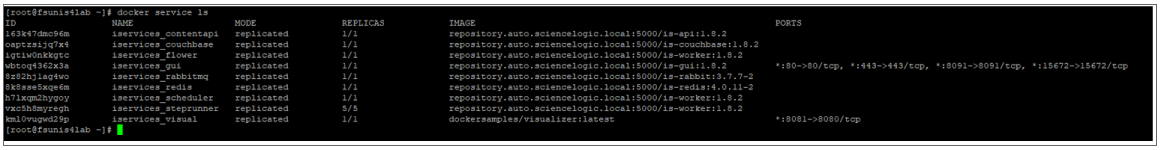
What do I do if I get a Code 500 Error when I try to access the PowerFlow user interface?

To address this issue:

1. SSH to your PowerFlow instance.
2. Check your Docker services with the following command:

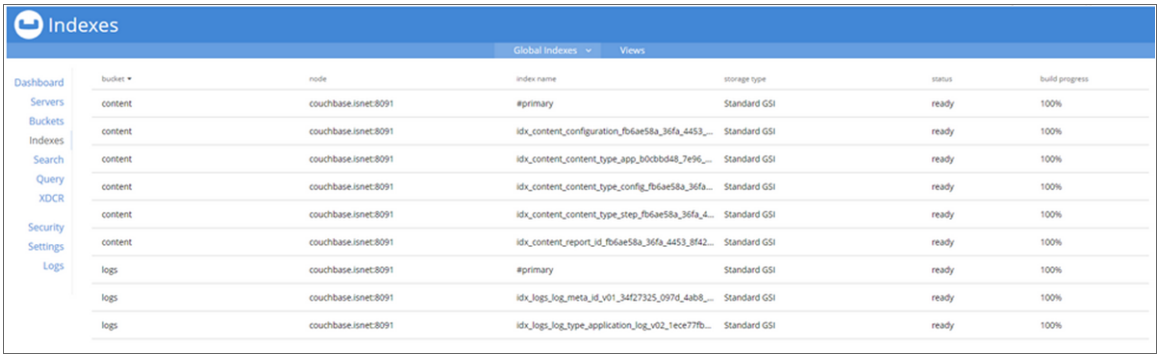
```
docker service ls
```

3. Ensure that all of your services are up and running:



4. If all of your services are up and running, but you are still getting Code 500 Errors, navigate to the Couchbase management portal of your PowerFlow server at port 8091 over HTTPS.

5. In the Couchbase portal, navigate to the **[Indexes]** tab and verify that all of your indexes are in a *ready* state:



6. Wait until all **status** entries are *ready* and all **build progress** entries are *100%*, and then navigate back to your PowerFlow user interface.
7. Verify that the Code 500 Error no longer exists.
8. If an index is stuck in a non-ready state, find the index name, copy that value and execute the following command in the Couchbase Query Editor:

```
BUILD INDEX ON content(INDEX_NAME_HERE)
```

What should I do if I get a 500 Error?

A 500 Internal Server Error will always have some kind of stack trace in the contentapi logs. Run the following command to find the cause of the 500 Error:

```
docker service logs -t iservices_contentapi
```

A 502 or 504 Error might mean that the user interface cannot reach an API container on a node, or the API container cannot reach a database node in the cluster. To address this issue:

- For a cluster, make sure the cluster is healthy, and all database nodes are balanced.
- For a cluster, make sure that the firewall ports are open on all nodes.
- Run the following commands to check the logs for 502 or 504 Errors:

```
docker service logs -t iservices_gui
```

```
docker service logs -t iservices_contentapi
```

The logs will specify which container caused a timeout when trying to reach that container.

What are some common examples of using the iscli tool?

The PowerFlow system includes a command line utility called the iscli tool. You can use the iscli tool to upload components such as steps, configurations, and applications from the local file system onto PowerFlow.

For more information on how to use this tool, SSH to your PowerFlow instance and type the following command:

```
iscli --help
```

You can use the iscli tool to add drop files or additional content onto the PowerFlow. You can also use the utility to upload content to a remote host. Examples of common syntax include the following:

```
iscli -usf <STEP_FILE.PY> -U isadmin -p <password>
```

```
iscli -uaf <APPLICATION_FILE.JSON> -U isadmin -p <password>
```

```
iscli -ucf <CONFIG_FILE.JSON> -U isadmin -p <password>
```

```
iscli -usf <STEP_FILE.PY> -U isadmin -p <password> -H <PF_HOST>
```

NOTE: The password for the iscli tool should be the same password as the PowerFlow Administrator (*isadmin*) user password. For more information, see ***Changing the PowerFlow Password***.

How do I view a specific run of an application in PowerFlow?

To view the log results of a previous execution or run of an application in the PowerFlow:

1. Use Postman or another API tool to locate the appID and the name of the application.
2. In the PowerFlow, update the PowerFlow URL with the appID, in the following format:

```
https://<PowerFlow>/integrations/<application_name>?runid=<App_ID>
```

For example:

```
https://<PowerFlow>/integrations/CreateServiceNowCI?runid=isapp-d8d1afad-74f8-42d4-b3ed-4a2ebcaef751
```

Why am I getting an "ordinal not in range" step error?

If you get an "ordinal not in range" error, check your CI Class Mappings to make sure the mappings do not contain international or "special" characters.

For example:

```
AWS | Availability Zone - São Paulo
```

If you find a class mapping with a special character like the above example, remove the class mapping, or rename the device class in SL1 to not include the special characters. Then you can sync the CI classes again.

How do I clear a backlog of Celery tasks in Flower?

To clear a backlog of Celery tasks:

1. docker exec into a bash shell in a worker process. For example:

   ```
   docker exec -it e448db31aaec /bin/bash
   ```

   where `e448db31aaec` is the container ID of the is-worker process on your system

2. Run the Python interpreter.
3. Run the following commands:

   ```
   from ipaascommon.celeryapp import app
   ```

   ```
   app.control.purge()
   ```

Why does traffic from specific subnets not get a response from PowerFlow?

In this situation, you can see traffic going into the host and into the Docker network, the traffic is not being routed back out. Responses were lost in the Docker ingress network, and the client timed out.

To address this issue:

1. Remove the Docker service by running the following command:

   ```
   docker stack rm iservices
   ```

2. Remove the default ingress network:

   ```
   docker network rm ingress
   ```

3. Add a newly addressed ingress network:

   ```
   docker network create --driver overlay --ingress --
   subnet=172.16.0.0/16 --gateway=172.16.0.1 ingress
   ```

4. Redeploy PowerFlow:

   ```
   docker stack deploy -c docker-compose.yml iservices
   ```

If the containers have an exposed port and you find the following error in the logs, you might need to remove **/var/lib/docker/network/files/local-kv.db**:

```
error="failed to detect service binding for container iservices_gui…"
```

To address this issue:

1. Remove the Docker service:

   ```
   docker stack rm iservices
   ```

2. Remove the .db file:

   ```
   rm /var/lib/docker/network/files/local-kv.db
   ```

3. Restart the docker daemon:

```
systemctl restart docker
```

4. Redeploy PowerFlow:

```
docker stack deploy -c docker-compose.yml iservices
```

What should I do if the number of tasks listed in the dashboards is not accurate?

To address an issue where the number of tasks listed in the PowerFlow and Flower dashboards do not match the Task List, you can set the FLOWER_MAX_TASKS environment variable in the PowerFlow **docker-compose** file to 20,000 tasks or higher.

For example:

```
flower:

environment:

... ....

worker_type: flower

FLOWER_MAX_TASKS: 20000
```

Why do I get "context deadline exceeded due to node exhaustion" when checking docker journalctl logs?

In unstable network environments where nodes are getting disconnected from the Docker Swarm, the error message "context deadline exceeded due to node exhaustion" might indicate that there is a network interruption. After confirming that the message is displayed when executing `journalctl --no-page | grep dockerd |grep error` in any of the cluster nodes, run the following pfctl action to increase the default Docker Swarm heartbeat period from five seconds to 20 seconds. For more information, see the section on *Increasing the PowerFlow Docker Swarm Heartbeat in Cluster Environments*.

```
pfctl --cluster-action --action update_swarm_heartbeat_period
```

Why do I get the following error when updating the PowerFlow administrator user password (isadmin)?

```
pfctl password set

.... ..

self.primary_manager = self.nodes[0]

IndexError: list index out of range
```

This error occurs because the correct syntax was not used when attempting to change the password. Use the following valid syntax to successfully update the password:

```
pfctl --host <ip1> <username1:host_password1> --host <ip2>
<username2:host_password2> password [ARGS]
```

For pfctlversions after 2.7.10, error handling has been improved to provide clearer feedback when attempting to change the password with incorrect syntax.

Why is the Monitor tab for Flower no longer visible?

The Flower user interface no longer displays the Monitor tab. This is because the new Flower 2 version generates Prometheus metrics. PowerFlow does not expose those metrics, since most of that information is already present in the PowerFlow Control Tower user interface.

Chapter 14

API Endpoints in SL1 PowerFlow

Overview

SL1 PowerFlow includes an API that is available after you install the PowerFlow system.

This chapter covers the following topics:

Interacting with the API *290*

Available Endpoints *290*

Interacting with the API

To view the full documentation for the PowerFlow API:

1. From the PowerFlow system, copy the **/opt/iservices/scripts/swagger.yml** file to your local computer.
2. Open a browser session and go to editor.swagger.io.
3. In the Swagger Editor, open the **File** menu, select **Import File**, and import the file **swagger.yml**. The right pane in the Swagger Editor displays the API documentation.

As of PowerFlow version 3.1.0 or greater, you can also access the full documentation for the PowerFlow API by going to the endpoint **/api/v1/docs**. The documentation can be downloaded if needed by going to **/api/v1/swagger?download=true**.

You can also access the full documentation for the PowerFlow API in the PowerFlow user interface:

1. Click on your username in the top right corner of PowerFlow.
2. Select *About* from the drop-down menu.
3. Click ***API Documentation***. This will open a new tab with the full PowerFlow API documentation.

Available Endpoints

POST

/api/v1/apikeys/. Add a new API key.

/api/v1/applications. Add a new application or overwrite an existing application.

/api/v1/applications/{appName}/run. Run a single application by name with saved or provided configurations.

/api/v1/applications/run. Run a single application by name.

/api/v1/configurations. Add a new configuration or overwrite an existing configuration.

/api/v1/roles/owner. Add a new owner assigned a specific role.

/api/v1/schedule. Add a new scheduled PowerFlow application.

/api/v1/status. Runs the "PowerFlow Control Tower HealthCheck" application to generate health status data.

/api/v1/steps. Add a new step or overwrite an existing step.

/api/v1/steps/run. Run a single step by name.

/api/v1/syncpacks/{syncpackName}/install. Install a specific SyncPack version by name.

/api/v1/tasks/{taskId}/replay. Replay a specific PowerFlow application. Replayed applications run with the same application variables, configuration, and queue as the originally executed application.

/api/v1/tasks/{taskId}/revoke. Terminate a specific task or application. By default, this command will not terminate the current running task.

/api/v1/tasks/{appId}/revoke. Terminate all tasks associated with a specific application.

/api/v1/me/widgets/{widget_id}. Creates a new widget or updates a existing widget used on the **PowerFlow Control Tower** page.

Querying for the State of a PowerFlow Application

When triggering PowerFlow application from the **applications/run** endpoint, you can query for the state of that application in two ways:

1. **Asynchronously**. When you POST a run of a PowerFlow application to **/applications/run**, the response is an integration status with a Task ID, such as: *isap-23233-df2f24-etc*. At any time, you can query for the current state of that task from the endpoint **/api/v1/tasks/isap-23233-df2f24-etc**. The response includes all of the steps run by the application, along with the status of the steps, and URL links to additional info, such as logs for each step.
2. **Synchronously**. When you POST a run of an application, you can tell PowerFlow to wait responding until the application is complete by adding the **wait** argument. For example, **/api/v1/applications/run?wait=20** will wait for 20 seconds before responding. The maximum wait time is 30 seconds. When the application completes, or 30 seconds has passed, the API returns the current status of the integration run. This process works the same as if you had manually queried **/api/v1/tasks/isapp-w2ef2f2f**. Please note that while the API is waiting for your application to complete, you are holding on to a thread. If you have multiple applications that run for a long period of time, do not use a synchronous query unless you have no other option. ScienceLogic recommends using an *asynchronous* query whenever possible.

GET

/api/v1/about. Retrieve version information about the packages used by this PowerFlow system, including the version of PowerFlow.

/api/v1/apikeys. Retrieve all available API keys saved in the PowerFlow system.

/api/v1/apikeys/{api_key}. Get details of a single API key.

/api/v1/applications. Retrieve a list of all available applications on this PowerFlow system.

/api/v1/applications/{appName}. Retrieve a specific application.

/api/v1/applications/{appName}/logs. Retrieve the logs for the specified application.

/api/v1/cache/{cache_id}. Retrieve a specific cache to gather information about the user interface and the PowerFlow applications.

/api/v1/cache/{cache_key}. Retrieve cache documents, but only if this cache document was explicitly saved to be exposed to the API. You will need to save the cache document using the latest version of the "SaveToCache" step in the *Base Steps* SyncPack. This step has a step parameter called "read_from_api" that lets you decide whether the cache document can be requested from the API.

/api/v1/configurations. Retrieve a list of all configurations on this PowerFlow system.

/api/v1/configurations/{configName}. Retrieve a specific configuration.

/api/v1/license?type=platform. Retrieve license data for this PowerFlow system.

/api/v1/reports. Retrieve a list of paginated reports.

/api/v1/reports/{reportId}. Retrieve a specific report by ID.

/api/v1/roles. Retrieve a list of available roles on this PowerFlow system.

/api/v1/roles/owner. Retrieve a list of roles assigned to owners on this PowerFlow system.

/api/v1/roles/owner/{owner}. Retrieve the role assigned to a specific owner.

/api/v1/sessions. Retrieve a list of sessions for this PowerFlow system.

/api/v1/sessions/status. Retrieve the Session Management status for this PowerFlow system.

/api/v1/sessions/username/{username}. Retrieve the session IDs for a specific user.

/api/v1/sessions/{session_id}. Retrieve a specific session from Session Management for this PowerFlow system.

/api/v1/schedule. Retrieve a list of all scheduled applications on this PowerFlow system.

/api/v1/status. Retrieve all the health status cache documents without running the "PowerFlow Control Tower HealthCheck" application.

/api/v1/status?all=true. Retrieve all health metrics for PowerFlow services and merge all the health status cache documents to return only one JSON response.

/api/v1/status/{service}. Retrieve all health metrics for a specific PowerFlow service, including the following services: **contentapi**, **couchbase**, **dexserver**, **iservices_syncpack_steprunner**, **iservices_syncpacks_steprunner**, **pfctl_output**, **rabbitmq**, **redis**, **steprunner**. Starting with PowerFlow version 2.6.0, you can run a `GET api/v1/status` operation that returns all the health status cache documents without running the HealthCheck application.

/api/v1/steps. Retrieve a list of all steps on this PowerFlow system.

/api/v1/steps/{stepName}. Retrieve a specific step.

/api/v1/docs/. Swagger user interface to display PowerFlow API documentation

/api/v1/swagger. Swagger file in JSON format that is consumed by the Swagger UI

/api/v1/swagger?download=true. Swagger in .yml format to download it as a file

/api/v1/swagger?download=true&force_reload=true. Force reload the Swagger file in .yml format to download. If this is not used, the file is cached.

/api/v1/syncpacks. Retrieve a list of all SyncPacks on this PowerFlow system.

/api/v1/syncpacks/{synpackName}. Retrieve the full details about a specific SyncPack.

/api/v1/syncpacks?only_installed=true. Retrieve a list of only the installed SyncPacks on this system.

/api/v1/syncpacks?only_activated=true. Retrieve a list of only the activated SyncPacks on this system.

/api/v1/tasks/{taskId}. Retrieve a specific task.

/api/v1/webhooks. Retrieve all available webhooks saved in the PowerFlow system.

/api/v1/me/widgets. Returns a list of all installed widgets used on the **PowerFlow Control Tower** page.

/api/v1/me/widgets/{widget_id}. Returns a specific widget using the specified widget ID.

DELETE

/api/v1/apikeys/{api_key}. Delete an API key.

/api/v1/applications/{appName}. Delete a PowerFlow application by name.

/api/v1/cache/{cache_id}. Delete a cache entry by name.

/api/v1/configurations/{configName}. Delete a configuration by name.

/api/v1/license?type=platform. Delete license data for this PowerFlow system.

/api/v1/me/widgets/{widget_id}. Delete the specified widget used on the **PowerFlow Control Tower** page.

/api/v1/roles/owner. Delete a specific owner role.

/api/v1/schedule. Delete a scheduled PowerFlow application by ID.

/api/v1/sessions. Delete a list of sessions for this PowerFlow system.

/api/v1/sessions?all=true. Delete all sessions for this PowerFlow system.

/api/v1/sessions/status. Delete the Session Management status for this PowerFlow system.

/api/v1/sessions/username/{username}. Delete the session IDs for a specific user.

/api/v1/sessions/{session_id}. Delete a specific session from Session Management for this PowerFlow system.

/api/v1/reports/{appName}. Delete a specific report by name.

/api/v1/reports/{reportId}. Delete a specific report by report ID.

/api/v1/steps/{stepName}. Delete a specific step by name.

/api/v1/syncpacks/{spName}. Delete a specific SyncPack by name.

Appendix

A

Configuring the SL1 PowerFlow System for High Availability

Overview

This appendix describes how to create High Availability deployments to protect the data in PowerFlow.

This chapter covers the following topics:

Types of High Availability Deployments for PowerFlow 295
Additional Deployment Options 305
Requirements Overview 306
Preparing the PowerFlow System for High Availability 311
Configuring Clustering and High Availability 312
Scaling iservices_contentapi 322
Single Manager Failure - Automatic Failover 322
Manual Failover 323
Additional Configuration Information 328
Known Issues 335

Types of High Availability Deployments for PowerFlow

The following table contains a set of ratings that depict the level of resiliency enabled by various PowerFlow deployment types. The higher the rating, the more resilient the PowerFlow system, not just from a node failure perspective, but also from a throughput and load-balancing regard.

Deployment Type	Resiliency Rating	Typical Audience
Single-node deployment	F	Users who want PowerFlow running, but do not care about failover.
Three-node cluster	B+	Users who want PowerFlow running, and also want support for automatic failover for one-node failure.
3+ node cluster with separate workers (at least 4 nodes)	A-	Users who want automatic failover for one-node failure, and intend to have very CPU- or memory-intensive tasks executing on the workers constantly.
3+ node cluster with separate workers, and drained manager nodes (at least 6 nodes)	A	Users who want automatic failover for one-node failure, intend to have very CPU- or memory-intensive tasks executing on the workers, and want to completely mitigate risks of resource contention between services.

You can start with any deployment type, and at a later time scale up to any other deployment type as needed. For example, a you can start with a single-node deployment, then at a later date add three more nodes to enable a 3+ node cluster with separate workers.

The deployments listed in the table are just the standards for deployment. For very high-scale customers, a more advanced deployment might be necessary. For deployment requirements like this, please contact ScienceLogic Support.

WARNING: If you are deploying PowerFlow without a load balancer, you can only use the deployed IP address as the management user interface. If you use another node to log in to the PowerFlow system, you will get an internal server error. Also, if the deployed node is down, you must redeploy the system using the IP address for another active node to access the management user interface.

NOTE: There is no support for active or passive Disaster Recovery. ScienceLogic recommends that your PowerFlow Disaster Recovery plans include regular backups and restoring from backup. For more information, see *Backing up Data*.

The standard deployments are listed below in the following topics:

- *Standard Single-node Deployment (1 Node)*
- *Standard Three-node Cluster (3 Nodes)*
- *3+ Node Cluster with Separate Workers (4 or More Nodes)*
- *3+ Node Cluster with Separate Workers and Drained Manager Nodes (6 or More Nodes)*

NOTE: You can use a command-line utility called **powerflowcontrol** (pfctl) that performs multiple administrator-level actions on either the node or the cluster. You can use this script to automate the configuration of a three-node cluster. For more information, see *Automating the Configuration of a Three-Node Cluster*.

Standard Single-node Deployment (1 Node)

Single-node deployment is the standard deployment that comes along with the ISO and RPM installation. This is the default deployment if you install the ISO and run the **pull_start_iservices.sh** script.

This deployment provides a single node running the PowerFlow system. If this node fails, the system will not be operational.

Requirements

One node, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see *System Requirements*.

Risks

A single node supports no data replication, no queue mirroring, and no failover capabilities.

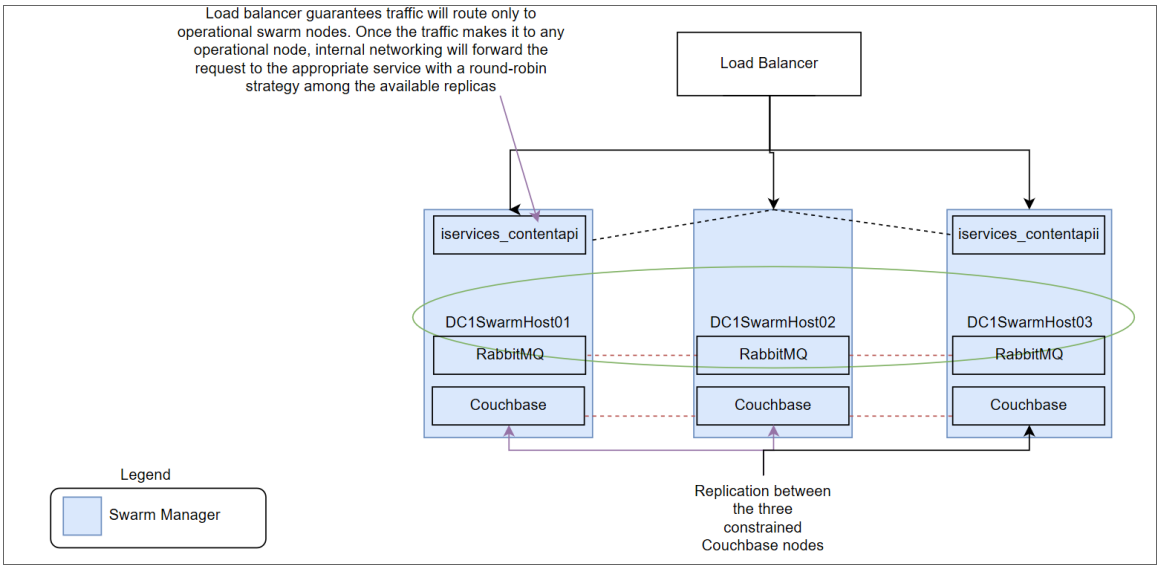
Configuration

This configuration is available as a default deployment with the docker-compose included in the PowerFlow 2.0.0 or later ISO or RPM.

Standard Three-node Cluster (3 Nodes)

The following High Availability deployment is an example of a three-node cluster:

- Each node in the Swarm is a Swarm Manager.
- All Swarm nodes are located within the same data center.



The three-node cluster is the most basic option providing full High Availability and data replication support among three nodes. In this deployment, each of the three nodes are running the same services in a clustered environment, which provides failover and data loss prevention capabilities. This deployment option will satisfy most High Availability needs, but it does not mitigate risks with the potential for worker operations to affect and degrade the database and queue services, because all services are running on the same nodes.

This deployment provides:

- **Automatic failover for one out of three node failure**: If one node in the cluster fails, automatic failover occurs, and the PowerFlow system will continue to be operational running on two out of three of the nodes.
- **Full data replication between all three nodes**. All nodes have a copy of the same data replicated across all three nodes. If one or two nodes fail, you will not experience data loss in the database or in the queues.
- **Full queue mirroring between all three nodes**. All nodes have a mirror of the queues defined in the PowerFlow environment. If one or two nodes fail, the system still retains messages in queues using the autoheal policy by default. For more information about autoheal behavior in RabbitMQ, see *The RabbitMQ Split-brain Handling Strategy*.

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see *System Requirements*.

Risks

When only three nodes are allocated used for High Availability, the following risks are present:

- **Over-utilization of nodes causing clustering issues**. In a three node cluster, worker containers, and Docker Swarm Managers are running on the same node as the database and queue services. As a result, if the node is not provisioned correctly, there could be some resource contention. If a node reaches 100% CPU, Docker Swarm cluster operations might fail, causing a node to completely restart, and causing a failover or other unexpected behavior.
- **Over-utilization of workers nodes causing database or queue issues**. Since all services are sharing the same nodes in this configuration, if worker operations become extremely CPU- or memory-intensive, the system might try to use resources needed from the database or queue. If this happens, you might encounter failures when querying the database or using the queues.

Mitigating Risks

The above risks can be mitigated by ensuring that the node is deployed with adequate CPU and memory for the workloads that you plan to run on the node. Memory limits are placed on containers by default. If needed, you could also add CPU limits to worker containers to further prevent resource contention.

Configuration

PowerFlow uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/scripts/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost. By default only main core services are included in the **docker-compose-override.yml** file, if extra services need to be added they should be included as needed.

Below is an example **docker-compose-override.yml** file for PowerFlow:

```
services:
  contentapi:
    environment:
      db_host: "couchbase.isnet,couchbase-worker.isnet,couchbase-work-
er2.isnet"
  couchbase:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
    environment:
      db_host: couchbase.isnet
    hostname: couchbase.isnet
    networks:
      isnet:
        aliases:
          - couchbase
          - couchbase.isnet
```

```
couchbase-worker:
  container_name: couchbase-worker
  deploy:
    placement:
      constraints:
        - "node.hostname == <Swarm node hostname2>"
    replicas: 0
  environment:
    AUTO_REBALANCE: "true"
    TYPE: WORKER
    db_host: couchbase
  hostname: couchbase-worker.isnet
  image: "sciencelogic/pf-couchbase:1.7.0"
  networks:
    isnet:
      aliases:
        - couchbase-worker
        - couchbase-worker.isnet
  ports:
    - "8100:8091"
  secrets:
    - is_pass
    - encryption_key
  volumes:
    - "/var/data/couchbase:/opt/couchbase/var"
couchbase-worker2:
  container_name: couchbase-worker2
  deploy:
    placement:
      constraints:
        - "node.hostname == <Swarm node hostname3>"
    replicas: 0
  environment:
    AUTO_REBALANCE: "true"
    TYPE: WORKER
    db_host: couchbase
  hostname: couchbase-worker2.isnet
  image: "sciencelogic/pf-couchbase:1.7.0"
  networks:
    isnet:
```

```
        aliases:
          - couchbase-worker2
          - couchbase-worker2.isnet
    ports:
      - "8101:8091"
    secrets:
      - is_pass
      - encryption_key
    volumes:
      - "/var/data/couchbase:/opt/couchbase/var"
  dexserver:
    deploy:
      replicas: 2
    environment:
      db_host: "couchbase.isnet,couchbase-worker.isnet,couchbase-work-
er2.isnet"
  pypiserver:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
  rabbitmq:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname1>"
    hostname: rabbit_node1.isnet
    image: "sciencelogic/pf-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node1.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
  rabbitmq2:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname2>"
```

```
    hostname: rabbit_node2.isnet
    image: "sciencelogic/pf-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
  rabbitmq3:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname3>"
    hostname: rabbit_node3.isnet
    image: "sciencelogic/pf-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node3.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"
  scheduler:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet"
  steprunner:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet"
version: "3.4"
volumes:
  rabbitdb:
```

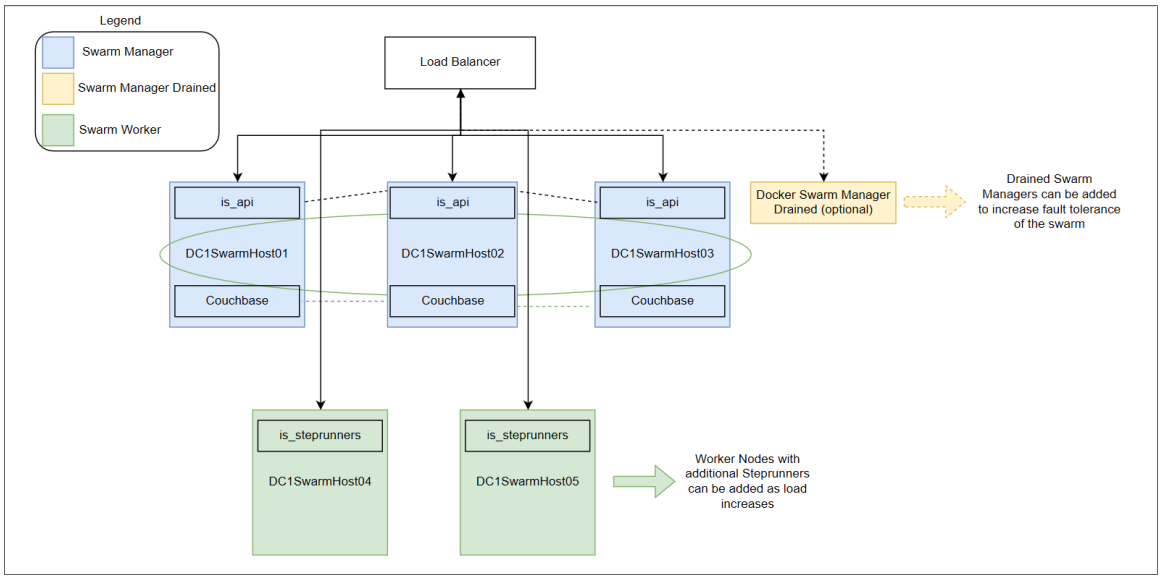
3+ Node Cluster with Separate Workers (4 or More Nodes)

The three-node cluster with separate workers is a slight variation of the standard three-node cluster. With this deployment strategy, all worker operation load is run by a separate independent node. This is preferable over the standard three-node deployment, because it completely prevents worker operations from stealing resources from the databases or queues.

Since steprunner workload is entirely on dedicated servers, you have greater ability to scale up to more workers, or even add additional nodes of workers to the system, without affecting critical database or queue operations.

This deployment provides a complete separation of worker processing from the database and queue processing, which is very helpful for users which have very CPU-intensive tasks that execute frequently.

The following High Availability deployment adds Docker Swarm worker nodes where steprunners can be constrained. This lets you continue to scale out new worker nodes as the load increases. This also lets you distribute steprunners based on workloads. Core services include ContentAPI, RabbitMQ, and Couchbase.



You can add drained Docker Swarm Manager nodes to increase fault tolerance of the Swarm, and to ensure that the orchestration of the Swarm is not impeded by large workloads on the core nodes.

The maximum Couchbase cluster with fully replicated nodes is four. Anything greater than four will not have a full replica set and will auto-shard data across additional nodes. There is no way as of this version of Couchbase to set the placement of the replicas. Redis replication and clustering is not currently supported in this version of PowerFlow.

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see ***System Requirements***.

One or more worker node with your choice of sizing.

Worker Node Sizing

Worker nodes can be sized to any CPU or memory constraints, though the greater the memory and CPU, more workers the node can run. The minimum size of a worker node is 2 CPU, 4 GB memory.

Risks

Core Node over-utilization could cause Swarm clustering problems. Because the Swarms are the same nodes as the core managers, there is a possibility for heavily loaded databases and queues to contend with the Swarm hosts for resources. In this case the Swarm may restart itself and the services running on that node. This is not as likely to occur with workers running on their own dedicated nodes.

Mitigating Risks

The above risks can easily be mitigated by ensuring the node is deployed with adequate CPU and memory for the workloads it is expected to run. Additionally, you can apply CPU and memory limits to the database or queue containers so that there will always be enough resources allocated to the host to prevent this scenario. For more information, see *Configuring Additional Elements of PowerFlow*.

Configuration

Using this configuration consists of:

- Joining the standard three-node Swarm cluster with one or more nodes as a Swarm worker.
- Labeling each additional "worker" node with a Swarm label "worker". For more information, see *Creating a Node Label*. You can also use the worker node role to restrict the steprunners to run only in the Swarm worker nodes using `node.role==worker` in the `constraints` section in the **docker-compose** file.
- In addition to the standard three-node deployment, you should update the steprunners to run on a dedicated node in the **docker-compose** file:

```
steprunner3:
  deploy:
    placement:
      constraints:
      - node.labels.types == worker
```

- You can edit the value of `--max-replica-per-node` in the **docker-compose-override** file to restrict the number of replicas that will run in each Swarm node. The default value is 5:

```
steprunner:
  deploy:
    replicas: 15
    ...
    placement:
      max_replicas_per_node: 5
  environment:
    ...
```

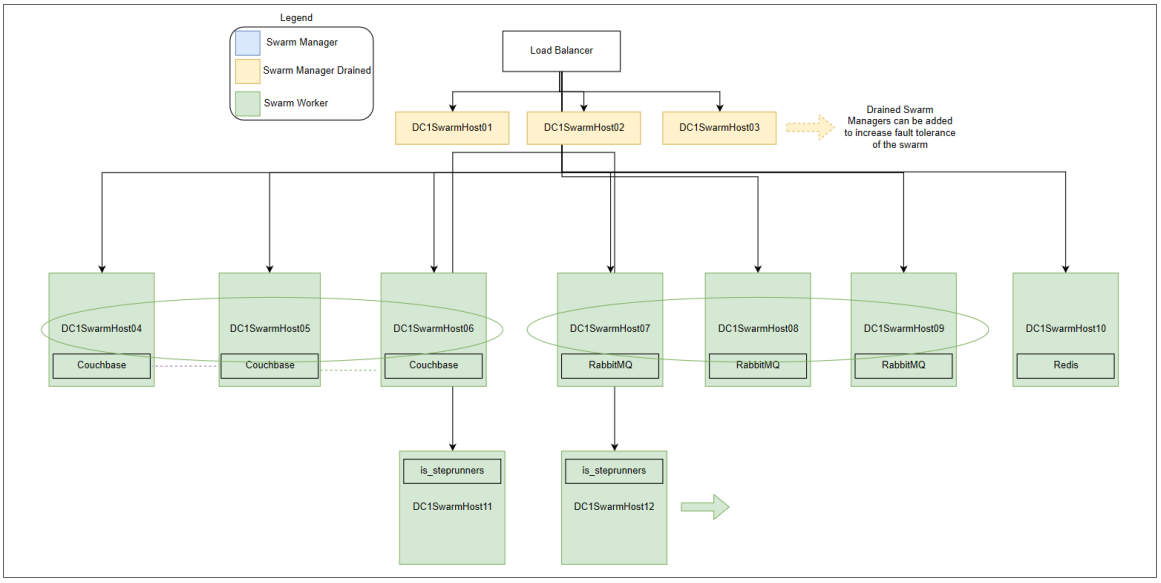
NOTE: The `--max-replica-per-node` option is available with docker-compose 3.8 or later. Add `version: '3.8'` at the start of the **docker-compose** file to ensure compatibility.

3+ Node Cluster with Separate Workers and Drained Manager Nodes (6 or More Nodes)

This deployment option is the most robust of the one-node auto-failover deployments, and completely mitigates known risks for resource contention in clusters.

This configuration provides everything that the 3+ node cluster with dedicated workers provides, with the addition of drained Swarm Managers. The drained Swarm Managers mitigate the risk of database or queue processing causing contention of resources for the Swarm clustering operations at the host level.

This deployment should only be used for large deployments of PowerFlow. This deployment separates out all the core services onto their own dedicated worker node and lets you distribute steprunners based on workloads:



You can add drained Docker Swarm Manager nodes to increase fault tolerance of the Swarm, and to ensure that the orchestration of the Swarm is not impeded by large workloads on the core nodes.

The maximum Couchbase cluster with fully replicated nodes is four. Anything greater than four will not have a full replica set and will auto-shard data across additional nodes. There is no way as of this version of Couchbase to set the placement of the replicas. Redis replication and clustering is not currently supported in this version of PowerFlow.

Requirements

Three nodes, 8 CPU, 24 GB memory minimum, preferably 34 GB to 56 GB memory, depending on workload sizes. For more information, see ***System Requirements***.

Also, three nodes, 2 CPU, 4 GB memory for the Swarm Manager.

Risks

None.

Configuration

Use the same **docker-compose-override.yml** file found in *Standard Three-node Cluster (3 Nodes)*.

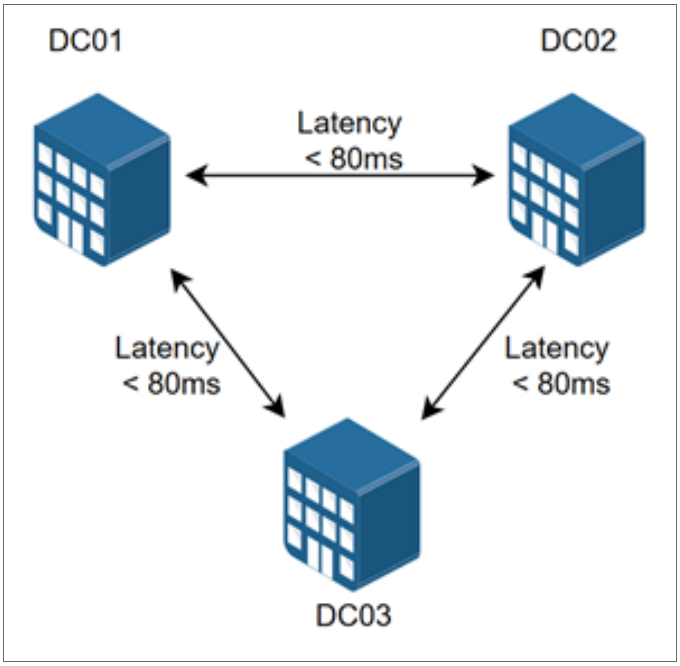
Next, add the additional three nodes to the cluster as managers, and drain them of all services (see *Using Drained Managers to Maintain Swarm Health*). Promote the drained nodes to Swarm Managers, and make all other nodes workers.

Additional Deployment Options

The following diagrams show additional High Availability deployment architectures that are supported for PowerFlow.

Cross-Data Center Swarm Configuration

Docker Swarm requires three data centers to maintain quorum of the swarm in the event of a full data center outage. Each data center must have a low-latency connection between the data centers.

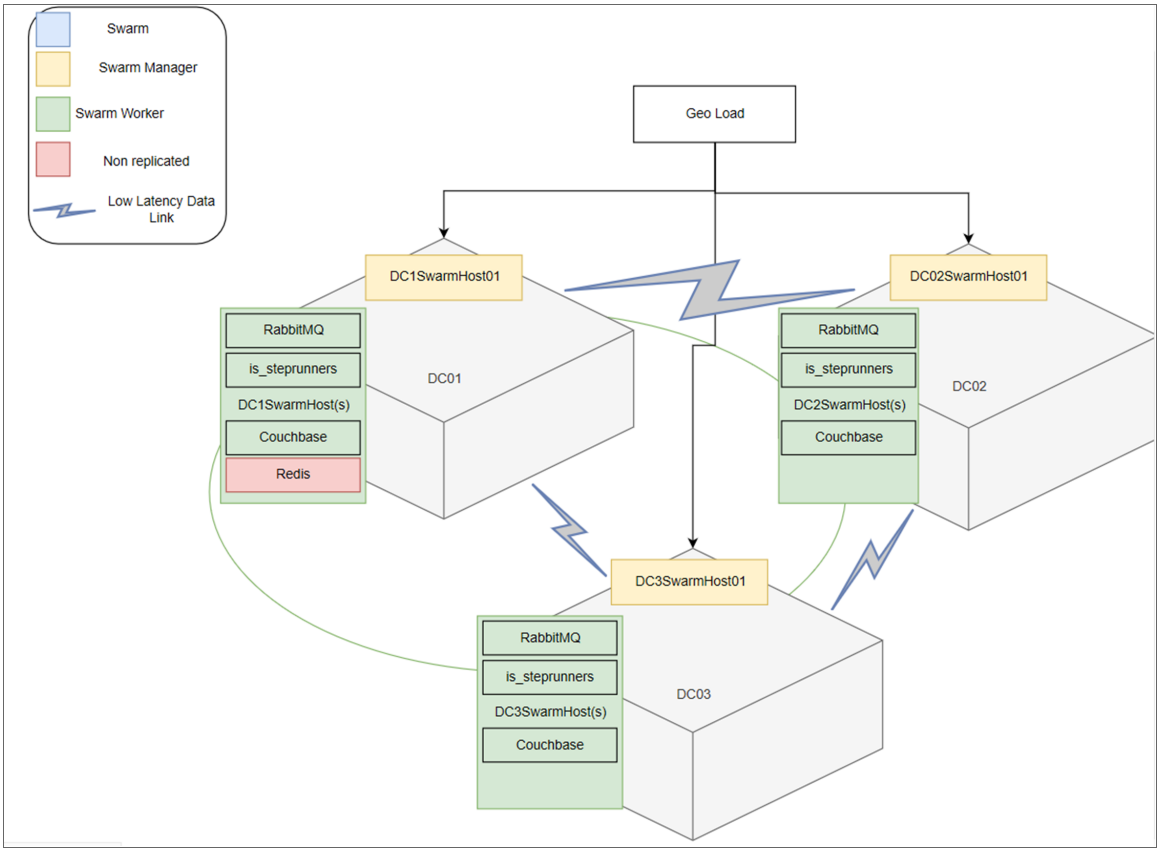


NOTE: Implementing clustering across links with a latency that is greater than 80 ms is not supported, and may cause one or more of the following situations: nodes dropping out of the cluster, or automatically failover, failed data replication, and potential cluster communication issues resulting in timeouts and significantly increased overhead.

The cross-data center configuration has the following limitation: the Redis service cannot be deployed in High Availability. As a result, all task results saved by any steprunner will have to be saved within that data center. Upon

a failure of that data center, a new Redis service will be created, but an application in the middle of its run would have to retry.

The following High Availability deployment shows a cross-data center swarm configuration:



Additional Notes

Tagging and constraints in the Docker compose file should be used to ensure proper placement. Example compose files are not available at this time.

Configuration management solutions such as Ansible should be used to update and manage large swarm environments.

For an easy upgrade of PowerFlow, use Docker Hub to pull the latest images or use an internal Docker registry.

Requirements Overview

Because PowerFlow uses the Docker Swarm tool to maintain its cluster and automatically re-balance services across nodes, ScienceLogic strongly recommends that you implement the following best practices from Docker, Couchbase, and RabbitMQ. The topics in this section describe those best practices, along with requirements and frequently asked questions.

IMPORTANT: To support automatic failover of the Couchbase database without manual intervention, you must set up at least **three nodes** for automatic failover of a single node, **five nodes** for automatic failover of two nodes, and so on.

NOTE: For a clustered PowerFlow environment, you must install the PowerFlow RPM on every server that you plan to cluster the PowerFlow. You can load the Docker images for the services onto each server locally by running **/opt/iservices/scripts/pull_start_iservices.sh**. Installing the RPM onto each server ensures that the PowerFlow containers and necessary data are available on all servers in the cluster. For more information, see *Installing PowerFlow via RPM*.

NOTE: You can use a command-line utility called **powerflowcontrol** (pfctl) that performs multiple administrator-level actions on either the node or the cluster. You can use this script to automate the configuration of a three-node cluster. For more information, see *Automating the Configuration of a Three-Node Cluster*.

Docker Swarm Requirements for High Availability

After implementing Docker Swarm High Availability, if a node goes down, all the services on that failed node can be dynamically re-provisioned and orchestrated among the other nodes in the cluster. High Availability for Swarm also facilitates network connections with the various other High Availability components.

Docker Swarm requires the following:

- The cluster contains at least **three nodes** running as managers. With three nodes, there can be a quorum vote between managers when a node is failed over.
- A load balancer with a virtual IP running in front of all nodes in the cluster. The load balancer allows user interface requests to be distributed among each of the hosts in the case one of the hosts fails for ports 443:HTTPS, 3141:Devpi and 5556:Dex.

An example of why a load balancer is needed in front of the virtual IP is the ServiceNow ticketing workflow. If you're only directing the request to a single node and that node goes down, your ticketing will stop even if the other PowerFlow nodes are still up and functional. The load balancer will account for the downed node and automatically route to the other nodes in the cluster.

For more information, see the Docker High Availability Documentation.

What happens if I use three nodes and two of the nodes fail?

Docker fault tolerance is limited to one failure in a three-node cluster. If more than one node goes down in a three-node cluster, automatic High Availability and failover cannot be guaranteed, and manual intervention may be required. Adding more nodes is the only way to increase the fault tolerance.

In the event of a two out of three failure, after you perform manual failover actions, the PowerFlow system will be back up and running.

For more information about the manual failover steps, see the *Failover* section.

Couchbase Database Requirements for High Availability

Couchbase High Availability ensures that no application, configuration, or step data from the PowerFlow system will be lost in the event of a node failure.

To support automatic failover, Couchbase requires at least **three nodes** in the high availability cluster.

Each node will have an independent and persistent storage volume that is replicated throughout the cluster. Alternatively, shared storage can be used instead of independent persistent volumes. This replication ensures that data is replicated in all places, and if a single node goes down, no data will be lost.

For more information, see the Couchbase documentation.

What if I have three nodes and two of them fail?

In the event of a failure of two out of three nodes, no data will be lost, because the data is being replicated.

If multiple Couchbase data nodes go down at the same time, automatic failover might not occur (not even nodes for quorum to failover). You will then need to perform manual failover steps. After you perform these manual actions, the PowerFlow system will be operational again. For more information about the manual failover steps, see the *Failover* section.

RabbitMQ Clustering and Persistence for High Availability

Implementing RabbitMQ High Availability ensures that if any integrations or tasks are waiting in the Rabbit queue, those tasks will not be lost if a node containing the Rabbit queue fails.

> **NOTE**: You can switch between both single-node and cluster options at any time during deployment.

RabbitMQ clustering requires a Docker Swarm configuration with multiple nodes. For more information, see *Configuring Docker Swarm*.

As a best practice for security, enable the user interface only temporarily during cluster configuration.

RabbitMQ Option 1: Persisting Queue to Disk on a Single Node (Default Configuration)

With this configuration, the PowerFlow queue runs on a single node, and the queue is persisted on disk. As a result, if the PowerFlow stack is removed and re-deployed, no messages are lost during the downtime. Any messages that exist in the queue before the stack is stopped continue to exist after the stack is re-deployed.

Potential Risks and Mitigations

Because the queue runs on a single node, if that node fails, then the queue and its related data might be lost.

You can mitigate data loss by persisting the queues on your choice of network shared storage, so that if the queue fails on one node, the queue and its messages can be brought back up on another node.

Requirements/Setup (Enabled by Default)

- You must define a static hostname for the RabbitMQ host in the docker-compose file. The default is *rabbit_node1.isnet*.
- You must mount a volume to **/var/lib/rabbitmq** in the docker-compose file.

Example docker-compose Definition

```
rabbitmq:
  image: sciencelogic/is-rabbit:3.7.7-1
  hostname: rabbit_node1.isnet
  volumes:
    - "rabbitdb:/var/lib/rabbitmq"
  networks:
    isnet:
      aliases:
        - rabbit
        - rabbit_node1.isnet
```

RabbitMQ Option 2: Clustering Nodes with Persistent Queues on Each Node

This configuration lets multiple nodes join a RabbitMQ cluster. When you include multiple nodes int he RabbitMQ cluster, all queue data, messages, and other necessary information is automatically replicated and persisted on all nodes in the cluster. If any node fails, then the remaining nodes in the cluster continue maintaining and processing the queue.

Because the RabbitMQ cluster includes disk-persisted queues, if all nodes in the Rabbit cluster fail, or if the service is removed entirely, then no data loss should occur. Upon restart, the nodes will resume with the same cluster configuration and with the previously saved data.

If you include multiple nodes in a RabbitMQ cluster, PowerFlow automatically applies an HA policy of all-node replication, with retroactive queue synchronization disabled. For more information, refer to the RabbitMQ documentation.

Potential Risks and Mitigations

If you create a Docker Swarm cluster with only two nodes, the cluster might stop functioning if a single node fails. To prevent this situation, include at least **three nodes** in each cluster.

Requirements/Setup

For a Docker Swarm configuration with multiple independent nodes:

- Both RabbitMQ services must be "pinned" to each of the two nodes. See the **Example Compose Definition** below.
- You must add a new RabbitMQ service to the docker-compose.yml file. This new service should have a hostname and alias following the designated pattern. The designated pattern is: *rabbit_nodex.isnet*, where *x* is the node number. This configuration supports up to 20 clustered nodes by default.
- After you update the docker-compose.yml file, the nodes will auto-cluster when you perform a deployment.

Example Code: docker-compose Definition of Two Clustered Rabbit Services

```
rabbitmq:
  image: sciencelogic/is-rabbit:3.7.7-1
  hostname: rabbit_node1.isnet
  volumes:
     - "rabbitdb:/var/lib/rabbitmq"
  networks:
     isnet:
       aliases:
         - rabbit
         - rabbit_node1.isnet
  deploy:
     placement:
       constraints:
         - node.hostname == node-number-1.domain
rabbitmq2:
   image: sciencelogic/is-rabbit:3.7.7-1
   hostname: rabbit_node2.isnet
   volumes:
     - "rabbitdb:/var/lib/rabbitmq"

   networks:
     isnet:
       aliases:
         - rabbit
         - rabbit_node2.isnet
  deploy:
     placement:
       constraints:
         - node.hostname == node-number-2.domain
```

Checking the Status of a RabbitMQ Cluster

This section contains commands and additional resources for administering your clusters.

To check the status of your clustered RabbitMQ environment:

1. Run `docker ps` and locate the **iservices_rabbit** container.
2. Run the following command on the RabbitMQ container:

```
docker exec -it [container_id] /bin/bash
```

You can run the following commands for more information:

- `rabbitmqctl cluster_status`. Returns information about the current cluster status, including nodes in the cluster, and failed nodes.
- `rabbitmqctl list_policies`. Returns information about current policies. Ensure that the ha-all policy is automatically set for your cluster.

For additional cluster-related administrative commands, see the RabbitMQ Cluster Management documentation page.

Preparing the PowerFlow System for High Availability

You need to prepare your PowerFlow system in the following ways before configuring the High Availability solution:

1. Make sure that your PowerFlow system has been updated with `yum upgrade`.
2. Run the following commands to open up the proper firewall ports for Docker Swarm on each swarm node:

```
firewall-cmd --add-port=2376/tcp --permanent

firewall-cmd --add-port=2377/tcp --permanent

firewall-cmd --add-port=7946/tcp --permanent

firewall-cmd --add-port=7946/udp --permanent

firewall-cmd --add-port=4789/udp --permanent

firewall-cmd --add-protocol=esp --permanent
```

> **NOTE:** If your system is fully yum-updated, you only need to run the following commands:
>
> ```
> firewall-cmd --add-service docker-swarm --permanent
>
> firewall-cmd --reload
>
> systemctl restart docker
> ```

> **TIP:** To view a list of all ports, run the following command: `firewall-cmd --list-all`

3. Make sure that the **/etc/iservices/is_pass** and **/etc/iservices/encryption_key** are identical on all clustered nodes.
4. Make sure that NTP is properly configured on all nodes:

- Edit the **/etc/chrony.conf file** to add NTP servers. If you want to use the pool.ntp.org NTP servers, remove the **.ol.** from the domain names.
- Enable chronyd by running the following commands:

```
systemctl start chronyd
```

```
systemctl enable chronyd
```

```
timedatectl #ensure ntp is enabled is yes and ntp sync is yes
```

Troubleshooting Ports and Protocols

If you have trouble with cluster configuration, make sure that all of the following ports and protocols are enabled between each of the nodes in the PowerFlow cluster within your network:

- 2376/tcp
- 2377/tcp
- 7946/tcp
- 7946/udp
- 4789/udp
- protocol=esp (IP Protocol 50)

Firewall rules are automatically configured within the PowerFlow operating system. Failing to allow required ports between nodes block PowerFlow clustering and networking.

Additionally, when using a load balancer, ensure that each of the PowerFlow cluster nodes are able to communicate to the load balancer listening ports (443, 5556, 3141).

Configuring Clustering and High Availability

This section describes how to configure clustering and High Availability with Docker Swarm and the Couchbase database, using three or more nodes.

> **NOTE:** This topic assumes you are using PowerFlow ISOs for each node, which includes an initial Docker Swarm node configuration. The use of the PowerFlow ISO is not required, however. You could instead deploy another node (without using the PowerFlow ISO) and configure a Linux operating system based on Red Hat. You could then add that system to the swarm.

TIP: When configuring a three-node clustered environment, you can set the **OPEN_SECONDARY_CB_PORTS** configuration variable to "true" to expose Couchbase secondary ports through the main node IP or host name. You can set this configuration variable as a GUI environment variable in the **docker-compose.yml** file, or you can set it in the **isconfig.yml** file in the host. If **OPEN_SECONDARY_CB_PORTS** is set to "true", the GUI service exposes the Couchbase secondary ports in the compose file. The **autocluster** cluster-action in the **powerflowcontrol** (pfctl) utility was updated to automatically expose Couchbase secondary ports when creating a three-node clustered environment.

For more information about troubleshooting issues with clustering, see *Troubleshooting Clustering and Node Failover*.

Automating the Configuration of a Three-Node Cluster

You can use the **powerflowcontrol** (pfctl) command-line utility to perform multiple administrator-level actions on your PowerFlow cluster. You can use the **autocluster** action with the **powerflowcontrol** command to automate the configuration of a three-node cluster.

NOTE: If you are using another cluster configuration, the deployment process should be manual, because the **powerflowcontrol** utility *only* supports the automated configuration of a three-node cluster.

WARNING: The **autocluster** action will completely reset and remove all data from the system. When you run this action, you will get a prompt verifying that you want run the action and delete all data.

To automate the configuration of a three-node cluster, run the following command:

```
pfctl --host <pf_host1> <username>:<host_password> --host <pf_host2>
<username>:<host_password> --host <pf_host3>  <username>:<host_password>
autocluster
```

For example:

```
pfctl --host 192.11.1.1 isadmin:passw0rd --host 192.11.1.2
isadmin:passw0rd --host 192.11.1.3 isadmin:passw0rd autocluster
```

Running this command will configure your PowerFlow three-node cluster without any additional manual steps required.

NOTE: You can use the **generate_haproxy_config** cluster-action in the **powerflowcontrol** (pfctl) utility to create an HAProxy configuration template that lets you easily set an HAProxy load balancer for a three-node cluster.

For example:

```
pfctl --host <host_IP_1> user:host_password --host <host_IP_2> user:host_password --host <host_IP_3> user:host_password cluster-action --action generate_haproxy_config
```

or

```
pfctl --config pfctl.yml cluster-action --action generate_haproxy_config
```

TIP: For more information about other actions you can perform with the **powerflowcontrol** utility, see *Using the powerflowcontrol (pfctl) Command-line Utility*.

Configuring Docker Swarm

To configure Docker Swarm for clustering (three or more nodes) and High Availability:

NOTE: Two-Node High Availability is not possible because Docker Swarm requires an odd number of nodes (3+) for quorum and consensus.

1. If you do not already have PowerFlow running in your environment, install PowerFlow on a single node. Doing this creates a single-node Docker Swarm manager.
2. Ensure that NTP is configured on all swarm nodes. For more information, see *Preparing PowerFlow System for High Availability*.
3. SSH to the Docker Swarm manager (leader) and run the following command to retrieve the join token. Make note of the token, because you will need it to join a node to the swarm in step 4, below:

   ```
   docker swarm join-token manager
   ```

4. Run the following commands on each Docker Swarm node that you want to join to the cluster:

   ```
   docker swarm init
   ```

   ```
   docker swarm join --token <join token> <swarm manager ip>:<port>
   ```

   where `<join token>` is the value from step 3. For example:

   ```
   docker swarm join --token SWMTKN-1-
   5e8skxby61cthkfkv6gzhhil89v0og2m7lx014tvvv42n7m0rz-
   an0fdam5zj0v7d471co57d09h 10.7.3.21:2377
   ```

5. Run the following command to verify that the nodes have been added:

   ```
   docker node ls
   ```

6. If you are using local images and not connecting to Docker Hub, load docker images on the other swarm nodes:

```
for i in $(ls -1 /opt/iservices/images/); do docker load -i
/opt/iservices/images/$i; done
```

Configuring the Couchbase Database

To add a Couchbase worker node:

1. In the **docker-compose-override.yml** file, add the following line to constrain the Couchbase container to a single Docker Swarm node at the bottom of the **couchbase** section:

```
deploy:
...
 hostname: couchbase.isnet
 deploy:
   placement:
     constraints:
       - node.hostname == <name of Docker Swarm node>

  networks:
    isnet:
      aliases:
        - couchbase
        - couchbase.isnet

   environment:
      db_host: couchbase.isnet
```

2. Add the couchbase-worker and couchbase-worker2 section. deploy > replicas on the workers should be set to 0:

```
couchbase-worker:
 image: repository.auto.sciencelogic.local:5000/is-couchbase:feature-
INT-1208-HA-IS-Services
 container_name: couchbase-worker.isnet
 volumes:
   - "/var/data/couchbase:/opt/couchbase/var"
 deploy:
   placement:
     constraints:
       - node.hostname == <name of Docker Swarm node>
 networks:
   isnet:
     aliases:
       - couchbase-worker
       - couchbase-worker.isnet
 hostname: couchbase-worker.isnet

 ports:
   - "8095:8091"
 secrets:
   - is_pass
   - encryption_key
 ulimits:
   nofile: 80000
   core: 100000000
   memlock: 100000000
 environment:
   TYPE: 'WORKER'
   AUTO_REBALANCE: 'true'
   db_host: 'couchbase'
```

NOTE: This deployment makes the Couchbase worker user interface available on port 8095 of the Docker Swarm stack. If the master node goes down, or if the primary Couchbase user interface is not available on port 8091, you can still access the secondary Couchbase user interface through port 8095.

3. Add couchbase-worker to the db_host setting for contentapi:

```
contentapi:
...
  environment:
    ...
    db_host: 'couchbase,couchbase-worker,couchbase-worker2'
```

4. All db_host variables in docker-compose should be in the following format:

```
db_host: 'couchbase,couchbase-worker,couchbase-worker2'
```

5. If you are using the override file, run the **/opt/iservices/compose_override.sh** script to validate and update the **docker-compose.yml** file with your changes.

6. Deploy the stack with only the Couchbase node by editing the replicas on couchbase-worker to 1 and running the following command:

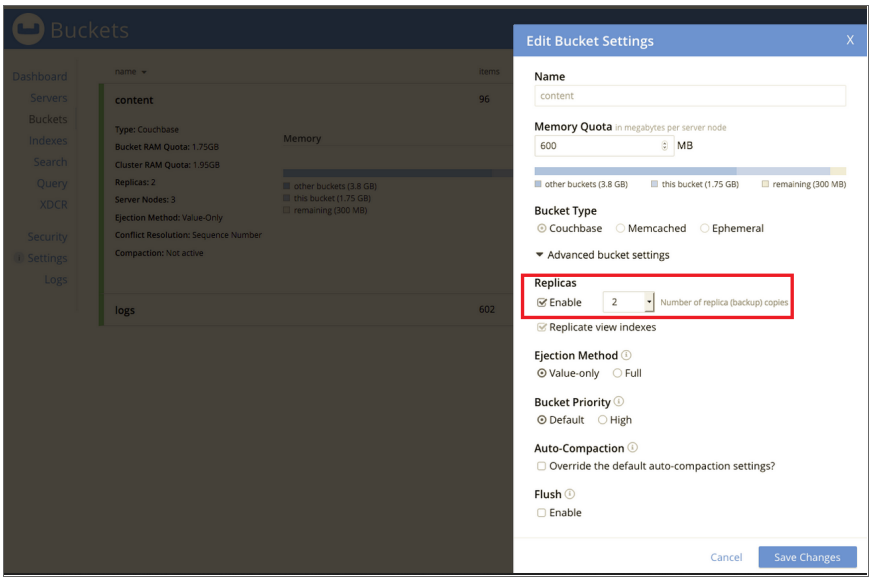
```
docker stack deploy -c <location of compose file> iservices
```

7. After the two-node Couchbase cluster has been successfully deployed and the secondary indexes are successfully added, edit the replicas on couchbase-worker2 to 1 and run the following command:

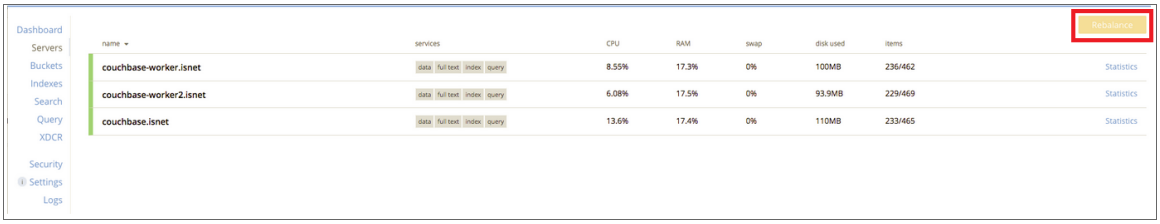
```
docker stack deploy -c <location of compose file> iservices
```

8. Set the replicas in the **docker-compose-override.yml** file as well.

9. After the second worker is added, set the number of replicas to "2" on each bucket (content and logs) in the Couchbase Administrator user interface and click **[Save Changes]**:



10. Rebalance the cluster by navigating to the **Servers** section of the Couchbase Administrator user interface and clicking the **Rebalance** button:



Code Example: docker-compose-override.yml

PowerFlow uses a **docker-compose-override.yml** file to persistently store user-specific configurations for containers, such as proxy settings, replica settings, additional node settings, and deploy constraints. The user-specific changes are kept in this file so that they can be re-applied when the **/opt/iservices/scripts/docker-compose.yml** file is completely replaced on an RPM upgrade, ensuring that no user-specific configurations are lost. By default only main core services are included in the **docker-compose-override.yml** file, if extra services need to be added they should be included as needed.

If you are running PowerFlow in a cluster, these files should always be the same between all manager nodes. With this in place, if any manager node dies, you can re-deploy with the same settings from any other manager node.

The following section includes a complete example of the **/opt/iservices/scripts/docker-compose-override.yml** file for a three-node Couchbase and RabbitMQ clustered deployment:

NOTE: If shared volumes are available in the cluster, the deploy placement can be omitted and removed.

```
version: '3.2'
services:
  steprunner:
    environment:
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet

  scheduler:
    environment:
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-work-
er.isnet

  couchbase:
    environment:
       db_host: 'couchbase.isnet'
    deploy:
```

```
    placement:
      constraints:
        - node.hostname == <swarm node hostname>
  networks:
    isnet:
      aliases:
        - couchbase
        - couchbase.isnet
  hostname: couchbase.isnet

couchbase-worker:
  image: sciencelogic/pf-couchbase:1.7.0
  container_name: couchbase-worker
  volumes:
    - "/var/data/couchbase:/opt/couchbase/var"
  ports:
    - "8100:8091"
  deploy:
    replicas: 0
    placement:
      constraints:
        - node.hostname == <swarm node hostname>
  networks:
    isnet:
      aliases:
        - couchbase-worker
        - couchbase-worker.isnet
  hostname: couchbase-worker.isnet
  secrets:
    - is_pass
    - encryption_key
  environment:
     TYPE: 'WORKER'
     AUTO_REBALANCE: 'true'
     db_host: 'couchbase'

couchbase-worker2:
  image: sciencelogic/pf-couchbase:1.7.0
  container_name: couchbase-worker2
  ports:
```

```
    - "8101:8091"
  volumes:
    - "/var/data/couchbase:/opt/couchbase/var"
  deploy:
    replicas: 0
    placement:
      constraints:
        - node.hostname == <swarm node hostname>
  networks:
    isnet:
      aliases:
        - couchbase-worker2
        - couchbase-worker2.isnet
  hostname: couchbase-worker2.isnet
  secrets:
    - is_pass
    - encryption_key
  environment:
     TYPE: 'WORKER'
     AUTO_REBALANCE: 'true'
     db_host: 'couchbase'
rabbitmq:
  image: sciencelogic/pf-rabbit:3.7.7-1
  hostname: rabbit_node1.isnet
  volumes:
     - "rabbitdb:/var/lib/rabbitmq"
  networks:
     isnet:
       aliases:
         - rabbit
         - rabbit_node1.isnet
  deploy:
     placement:
       constraints:
         - node.hostname == <swarm node hostname>
rabbitmq2:
   image: sciencelogic/pf-rabbit:3.7.7-1
   hostname: rabbit_node2.isnet
   volumes:
     - "rabbitdb:/var/lib/rabbitmq"
```

```
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
    deploy:
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
  rabbitmq3:
    image: sciencelogic/pf-rabbit:3.7.7-1
    hostname: rabbit_node3.isnet
    volumes:
      - "rabbitdb:/var/lib/rabbitmq"

    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node3.isnet
    deploy:
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
  contentapi:
    environment:
      db_host: 'couchbase.isnet,couchbase-worker.isnet,couchbase-work-
er2.isnet'

  pypiserver:
    image: sciencelogic/pf-pypi:4.8.0-1
    hostname: devpi
    container_name: devpi
    volumes:
       - "devpi:/data"
    networks:
       isnet:
          aliases:
             - pypiserver
```

```
    secrets:
       - is_pass
  dexserver:
    image: scr.sl1.io/is-dex:2.18.0-1
    ports:
      - "5556:5556"
      - "5558:5558"
    command: ["serve", "/dexConfiguration.yaml"]
    networks:
      isnet:
        aliases:
          - dexserver
    configs:
      - source: dex_config
        target: /dexConfiguration.yaml
volumes:
  rabbitdb:
  devpi:
configs:
  dex_config:
    file: /etc/iservices/dexConfiguration.yaml
```

Scaling iservices_contentapi

To scale out iservices_contentapi to distribute the service across the three nodes, run the following command:

```
docker service scale iservices_contentapi=3
```

Single Manager Failure - Automatic Failover

When one node in the cluster fails (Node1, Node2, or Node3), the remaining nodes maintain quorum and fail-over happens automatically.

1. Log in to PowerFlow.
2. Access the Couchbase Administrator user interface using one of the Couchbase Node Ports that is still up:
   - Couchbase: 8091
   - Couchbase Worker 1: 8100 or Couchbase Worker 2: 8100

   ```
   https://<IP of PowerFlow>:Port
   ```

3. Once the downed node comes back, it should rejoin the cluster automatically. If it does not, and needs to be reset, follow the steps below:
    a. Force the node to leave the swarm.
    b. Get the swarm token from the other nodes.
    c. Force the node to join the swarm again. Execute the following command:

```
docker swarm leave --force
```

> **NOTE:** In a three-node cluster, a single failed node will be automatically removed. You will still need to perform a re-balance.

Manual Failover

If you have a cluster with three or more nodes that is not configured with automatic failover, you must perform the following manual failover steps.

> **NOTE**: If you can access the Couchbase Administrator user interface (**Couchbase: 8091, Couchbase Worker 1: 8100, or Couchbase Worker 2: 8100**) on the node that is still running, you can simply click the **[Failover]** button in the Couchbase Administrator user interface instead of manually running the couchbase-cli commands below.

> **NOTE:** In a three-node cluster, a single failed node will be automatically removed. You will still need to perform a re-balance.

Initiating Manual Failover

To initiate a manual failover and promote the only Couchbase node that is up:

1. Log in to the Docker Swarm node where the node that is running resides.
2. Run the following command on that node to see which node IDs exist:

```
docker node ls
```

> **NOTE:** If this command failed, and the error message states "The swarm does not have a leader. It's possible that too few managers are online. Make sure more than half of the managers are online.", run the following command to restart the cluster creation. This will restart the running services. Go to step 4 if this command was run and wait until the Couchbase container starts.
>
> ```
> docker swarm init --force-new-cluster
> ```

3. Remove any failed manager nodes from the cluster by running the following Docker command:

```
docker node rm <failed node id>
```

4. Run the following command to identify the Container ID of the running Couchbase container:

```
docker ps
```

5. Connect to the Docker container that is still up:

```
docker exec -u root -i -t $(docker ps -q -n 1 --filter
name=iservices_couchbase) /bin/bash
```

6. Use the instance of Couchbase that is up by running the following commands:

```
couchbase-cli server-list -c <operating-couchbase-node> -u isadmin -p
<password>
```

   where `<operating-couchbase-node>` could be one of the following:

   - couchbase.isnet
   - couchbase-worker.isnet
   - couchbase-worker2.isnet

   and the password is the PowerFlow Administrator user interface password.

7. One of the previous commands will show one or two failed nodes. Copy the Couchbase node names for step 8.

8. Use the currently running node (operating-couchbase-node) and the failed node's names to run the following command to failover:

```
couchbase-cli failover -c <operating-couchbase-node>:8091 -u isadmin
-p <password> --server-failover <failed-couchbase-node>:8091 --force
```

For example, if the operating node is `couchbase-worker`, and the isnet name:port of the failed service is `couchbase.isnet:8091`, then the command would be:

```
couchbase-cli failover -c couchbase-worker:8091 -u isadmin -p
<password> --server-failover couchbase.isnet:8091 --force
```

If the command fails and suggests using the `--hard` argument, then the command would be:

```
couchbase-cli failover -c couchbase-worker:8091 -u isadmin -p
<password> --server-failover couchbase.isnet:8091 --force --hard
```

NOTE: Execute the command above for both failed nodes. If the command returns an error, run the following commands to sendboth failed nodes as part of the command. This is likely to happen if both unhealthy nodes went down at the same time.

```
couchbase-cli failover -c <operating-couchbase-node>:8091 -u
isadmin -p <password> --server-failover <failed-couchbase-
node>:8091,<second-failed-couchbase-node>:8091 --force
```

```
couchbase-cli failover -c <operating-couchbase-node>:8091 -u
isadmin -p <password> --server-failover <failed-couchbase-
node>:8091,<second-failed-couchbase-node>:8091 --force --hard
```

9. Rebalance the cluster using the functioning container name:

```
couchbase-cli rebalance -c <operating-couchbase-node>:8091 --server-
remove <failed-couchbase-node>:8091 --server-remove <second- failed-
couchbase-node>:8091 -u isadmin -p <password>
```

10. Remove and recreate the indexes to avoid issues:

```
couchcontrol -c <operating-couchbase-node> index remove-secondary
```

```
couchcontrol -c  <operating-couchbase-node> index create-secondary -f
/tmp/scripts/couchbase_index.json -b
```

For example, if the operating node is couchbase-worker, then the command would be:

```
couchcontrol -c couchbase-worker index remove-secondary
```

```
couchcontrol -c couchbase-worker index create-secondary -f
/tmp/scripts/couchbase_index.json -b
```

11. Scale down the unhealthy and inactive couchbase services that were just removed from the cluster

```
docker service scale iservices_<couchbase-failed-node>=0
```

```
docker service scale iservices_<second-couchbase-failed-node>=0
```

12. If the contentapi service is in a waiting state, restart the couchbase node that was just promoted to reset the connection and resolve the API waiting. Run the following command:

```
docker service update --force iservices_<couchbase-node-name>
```

13. Force the Dex server service to restart:

```
docker service update --force iservices_dexserver
```

14. Log in to the PowerFlow user interface and validate that your data still exists.

NOTE: Some documents may be lost. If Couchbase lost its quorum, multiple documents, including the scheduler document, might be lost. The applications that were still queued to run should still be able to run if the applications and their configurations were not affected.

15. Go to the Couchbase user interface, which should be available at one of the following ports, depending on the active node :8091, :8100, or :8101. If the removed nodes are still there and waiting for a rebalance action, click the **[Rebalance]** button.

Recovering a Docker Swarm Node

If a node does not join the swarm automatically, follow the steps below to recover a Docker Swarm node:

1. Restart the node.
2. If manual failover actions were taken while this node was offline, run the following command to force the node to leave the swarm now that the node is back online:

```
docker swarm leave --force
```

3. Follow the steps in *Configuring Docker Swarm* to add the node back to the existing swarm by obtaining the join-token from the manager.

Restoring a Couchbase Node

CAUTION: You should take the restoration actions in this topic *only* after a manual or automatic failover has been performed and the node has been completely removed from the cluster(the node should not be visible in the user interface or server-list).

IMPORTANT: If the logs bucket has more than 10,000 documents, the rebalance actions could take more time when the new nodes rejoin the cluster. To avoid this, if the logs documents are not critical, you can flush the logs bucket in the Couchbase user interface.

To restore the failed Couchbase node:

1. Log in to Couchbase Administrator user interface using the port of the node that is still up (8091 or 8100)
2. If the Docker Swarm node was restored and not rebuilt, remove files from the old container:

```
rm -rf /var/data/couchbase/*

docker volume rm iservices_tmp_couchbase # only for MUD environments

docker service scale iservices_couchbase=1
```

   A new node is added to the Couchbase cluster. If the rebalance environment variable is set, the balancer process will start automatically. If not, click the **[Rebalance]** button in the Couchbase user interface so the reset node can be added to the cluster.

   NOTE: If the server is not completely removed from the cluster and is just waiting to be added back, you may do so using the Couchbase user interface, or by running **healthcheck** and **autoheal** actions with the **powerflowcontrol** (pfctl) command-line utility.

3. After all nodes in a cluster are running, be sure to perform **healthcheck** and **autoheal** actions with the **powerflowcontrol** (pfctl) command-line utility to re-validate the cluster and re-set configurations such as replication and index counts. For more information, see ***healthcheck and autoheal***.

NOTE: If the master node goes down, the SyncPacks for the PowerFlow system might not display. This is because the pypiserver is constrained by default to one master node, so it does not start on workers if that master node goes down. To address this issue after completing the failover steps, above, you can re-import the SyncPacks.

NOTE: If two Couchbase nodes were reset, documents form the database such as applications, configurations, scheduler and others may be lost. ScienceLogic recommends restoring from a recent backup to have all documents in place.

Restoring RabbitMQ

RabbitMQ nodes automatically join the cluster and sync data, but if big workloads were desynchronized, there could be some issues and the unhealthy RabbitMQ nodes may need to be reset.

The powerflowcontrol (pfctl) healthcheck executes the actions below, but ScienceLogic also recommends checking manually:

1. Log in to the RabbitMQ user interface and check that all the nodes are clustered together.
2. Go to the **Queues** tab and check that the queues for the application are synchronized. If there is a red "+1" in the Node list, enter that queue and click the **[Synchronize]** button.
3. If the synchronization above fails, try clearing out the volumes of the nodes that do not want to be syncronized. For more information, see the Troubleshooting SL1 PowerFlow chapter in the SL1 PowerFlow Platform manual.
4. Run powerflowcontrol (pfctl) **healthcheck** and **autoheal** to make sure the system is healthy and has the corresponding configurations. For more information, see *healthcheck and autoheal*.

Additional Configuration Information

Load Balancer Recommended Settings

Configurations to Improve Load Balancer Compatibility

You can use the following configurations in the **/etc/iservices/isconfig.yml** file to improve load balancer compatibility if the load balancer sends requests to the client in proxy protocol format like AWS ELB:

- **LOAD_BALANCED: true**. Setting this value to **true** specifies that the load balancer will send requests to the client in proxy protocol format. This value is **false** by default.
- **RATE_LIMITED**. Setting this value to **true** enables rate limiting. This value is **false** by default.
- **RATE_LIMIT_REQUESTS_PER_SECOND**. This value specifies the number of rate limit requests per second. The default is **'50'**.
- **RATE_LIMIT_BURST**. This value specifies the rate limit burst. The default is **'100'**.

IMPORTANT: You will need to re-deploy the PowerFlow stack for any changes to the **docker-compose.yml** file to take place.

In addition, the exposed ports in the **docker-compose.yml** file are set to **mode: host** to let PowerFlow capture the proper client IP address of the requests being sent into PowerFlow. This setting lets PowerFlow set the proper rate limits and log transactions. This feature does not allow using the Swarm ingress; as a result, you will need to scale the gui container and place the container in the nodes that will be expecting ingress traffic.

Recommended Load Balancer Modes

Use TCP mode or HTTP mode, plus the recommended healthcheck endpoints listed below. ScienceLogic recommends that you use TCP instead of HTTP (which requires specific endpoints).

NOTE: If you use HTTP mode, make sure that the SSL cipher configurations are in place to work with OpenSSL 1.1.1K FIPS ciphers. If needed, run some openssl commands, such as `openssl ciphers ...` against the PowerFlow system.

Recommended HealthCheck Endpoints

When using HTTP mode with the load balancer, configure the following **healthcheck** endpoints to make sure that the PowerFlow nodes are responding correctly.

PowerFlow 2.5.0 or later

> **NOTE:** This version includes **healthcheck** endpoints for easy verification.

https://pf-node:5556/healthcheck. Should respond with a "200 status, with an "ok" response.

or

https://pf-node:5556/dex/theme/styles.css . Should respond with a "200" status code.

https://pf-node/discovery. Should respond with a "200 status, with an "ok" response.

https://pf-node:15672/healthcheck. Should respond with a "200 status, with an "ok" response.

https://pf-node:8091/healthcheck. Should respond with a "200 status, with an "ok" response.

PowerFlow 2.4.1

https://pf-node:5556/dex/theme/styles.css. Should respond with a "200" status code.

https://pf-node/discovery. Should respond with a "200" status code.

https://pf-node:15672.Should respond with a "302" status code.

https://pf-node:8091. Should respond with a "301" status code

cURL Commands

To verify that PowerFlow nodes can reach the Load Balancer, you can execute the following cURL commands from the PowerFlow nodes to the Load Balancer, and from the Load Balancer to the PowerFlow nodes:

`curl https://IP:5556/dex/theme/styles.css`. Should respond with a "200" status code.

`curl https://IP/discovery`. Should respond with a "200" status code.

`curl https://IP:15672`. Should respond with a "302" status code.

`curl https://IP:8091` . Should respond with a "301" status code.

Optimization Settings to Improve RabbitMQ Reclustering

To avoid a potential race condition between three RabbitMQ nodes, and to improve how the nodes recluster after the PowerFlow stack is redeployed. ScienceLogic recommends setting the following configurations:

- PowerFlow RabbitMQ node 1 = 20-second grace period. The configuration `stop_grace_period: 20s` is required in the `rabbitmq` service definition in **docker-compose.yml**. For example:

```
rabbitmq:
  stop_grace_period: 20s
```

- PowerFlow RabbitMQ node 2 = 10-second grace period. No configuration is required as this is the default value.
- PowerFlow RabbitMQ node 3 = 10-second grace period. No configuration is required as this is the default value.

In environments where the latency between nodes is higher, you can use the following settings:

- PowerFlow RabbitMQ node 1 = 30-second grace period. The configuration `stop_grace_period: 20s` is required in the `rabbitmq` service definition in **docker-compose.yml**.
- PowerFlow RabbitMQ node 2 = 20-second grace period. The configuration `stop_grace_period: 20s` is required in the `rabbitmq` service definition in **docker-compose.yml**. .
- PowerFlow RabbitMQ node 3 = 10-second grace period. No configuration is required as this is the default value.

TIP: The `stop_grace_period` setting allows Docker Swarm to stop the container after the configured time, which is why the first node must have a longer grace period. For more information, see https://docs.docker.com/compose/compose-file/05-services/#stop_grace_period.

Optimization Settings to Improve Performance of Large-Scale Clusters

In large-scale clusters, one of the root causes of abnormal memory and CPU usage is from inter-worker communication overhead, or overly "chatty" workers, and their event queues. You can completely disable inter-worker eventing to significantly reduce overhead on the queuing system and prevent the symptoms associated with abnormal memory usage.

Also, to improve the performance of large-scale clusters by default, the following optimization settings were added to the **docker-compose.yml** file for all workers in version 2.0.1 of the PowerFlow platform:

```
steprunner-<worker-x>:
  environment:
    additional_worker_args: "--max-tasks-per-child 1 --without-gossip --
without-mingle"
```

In addition to the default optimization settings above, you can further reduce system overhead by setting the `--without-heartbeat` environment variable in `additional_worker_args`. Please note that this setting will reduce the memory and CPU utilization of the system, but it will come at the cost of preventing the Flower service from getting an accurate depiction of current worker states.

If you want to disable these new configuration settings, set the environment variable `"disable_default_optimizations"` to `"True"` for all workers.

NOTE: Workers will continue to generate events for consumption from monitoring tools like Flower even with the new default configuration settings. In some extremely large clusters, you might want to completely disable eventing of workers completely, especially if Flower is not in use. To completely disable worker eventing, set the environment variable `"disable_events"` to `"True"`.

For more information, see https://docs.celeryproject.org/en/latest/reference/celery.bin.worker.html#cmdoption-celery-worker-without-gossip

Additional suggestions for improving performance in large-scale clusters:

- Assess the impact of using Flower before keeping it enabled for a long period of time. Running Flower can cause increased overhead on the RabbitMQ nodes, but the overhead is not significant initially. However, the overhead generated by Flower will continue to increase as more workers are added to the stack, and those workers send events to Flower.
- ScienceLogic recommends that you monitor memory and queue utilization before and after running Flower with your current environment size to determine whether the extra overhead provided is worth the task information it provides.
- If a system event causes workers to restart, it is possible that all workers constantly restarting at the same time, every 0 seconds will generate increased load on the system, making it difficult for other services to start up. To prevent this, it is recommended to add a **restart_delay** to workers to prevent a "rush" of hundreds of workers trying to re-connect over the network all at once. For example:

```
steprunner-<worker-x>:
  deploy
    restart_policy:
      delay: 30s
```

Exposing Additional Couchbase Cluster Node Management Interfaces overTLS

The **is_gui** container acts as a reverse proxy to the internal services and their individual management interfaces. This container configured in **/etc/nginx/conf.d/default.conf**.

To expose the management interfaces of additional Couchbase nodes within a cluster:

1. Copy the configuration from the gui container:

```
docker cp <container id>:/etc/nginx/conf.d/default.conf ./
```

2. Edit the configuration to include the desired services:

```
server {
    listen       8092 ssl;
    server_name  couchbase-worker;

location = / {
  return 301 https://$host:8092/ui/index.html;
}

 location / {
  resolver 127.0.0.11 valid=5s;
  set $upstream couchbase-worker.isnet;
  proxy_pass http://$upstream:8092$request_uri;
  proxy_pass_header Server;
  proxy_pass_header Cache-Control;
  proxy_pass_header Content-Length;
  proxy_pass_header Connection;
  proxy_pass_header Pragma;
  proxy_pass_header ns-server-ui;
  proxy_pass_header invalid-auth-response;
}
    ssl_certificate     /etc/iservices/is_cert.pem;
    ssl_certificate_key /etc/iservices/is_key.pem;
    ssl_protocols       TLSv1.2;
    ssl_ciphers         'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-GCM-
SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES128-
GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:20m;
    ssl_session_timeout 180m;
    add_header Strict-Transport-Security "max-age=31536000" always;
}
```

3. Create the following Dockerfile:

```
FROM sciencelogic/is_gui

COPY ./default.conf /etc/nginx/conf.d/default.conf
```

4. Build the container with the new configurations:

```
docker build -t <customer>/is_gui:<PowerFlow version>-1 -f Dockerfile
```

5. Add the image name to the **is_gui** section in the **docker-compose-override.yml** file, and do a Docker stack deploy to enable the new **is_gui** container.

Restricting the Number of Replicas

If you use the `max-replica-per-node` option in the **docker-compose-override** file to restrict the number of replicas that will run in each Swarm node, you should also specify more than one replica for the `replicas` option.

ScienceLogic recommends using three replicas for the **gui** service: one running on each of the core nodes. The default `max-replica-per-node` value is 1 for the **gui** service.

For the **gui** service, the `max_replicas_per_node` option is set, but the **gui** service is not pinned to the core nodes using labels. If there are more than three nodes (core and worker nodes) the **gui** service replicas will run in any node, so you will need to add a restriction to the **docker-compose-override** file, such as the following:

```
placement:
  constraints:
    - node.labels.types == master
```

If you update the **docker-compose-override** file, you will need to redeploy the corresponding service. In the example below, you will need to redeploy the **gui** service after you update the override file:

```
gui:
  deploy:
    replicas: 3
    ...
    placement:
      max_replicas_per_node: 1
  environment:
    ...
```

TIP: You can also configure the `max_replicas_per_node` option for the **contentapi**, **dexserver** and **steprunners** services, and any other PowerFlow services that have more than one replica.

NOTE: The `max-replica-per-node` option is available with docker-compose 3.8 or later. Add `version: '3.8'` at the start of the **docker-compose** file to ensure compatibility.

HAProxy Configuration (Optional)

CAUTION: As a convenience, ScienceLogic provides an example configuration for the HAProxy load balancer below. Please note that it is your responsibility to configure the load balancer. ScienceLogic cannot be held responsible for any deployments that deviate from the example HAProxy load balancer configuration.

The following example configuration describes using HAProxy as a load balancer:

Code Example: HAProxy as Load Balancer

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option                  redispatch
    retries                 3
    timeout http-request    1m
    timeout queue           1m
    timeout connect         1m
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 6000

frontend http_front
  bind *:80
  bind *:443
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
  default_backend http_back

frontend dex_front
  bind *:5556
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
```

```
  default_backend dex_back
frontend  devpi_front
  bind *:3141
  option tcplog
  mode tcp
  tcp-request inspect-delay 5s
  default_backend  devpi_back

backend http_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:443 check
  server master2 <docker swarm node 2 ip>:443 check
  server master3 <docker swarm node 3 ip>:443 check

backend dex_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:5556 check
  server master2 <docker swarm node 2 ip>:5556 check
  server master3 <docker swarm node 3 ip>:5556 check


backend devpi_back
  mode tcp
  balance roundrobin
  server master1 <docker swarm node 1 ip>:3141 check
  server master2 <docker swarm node 2 ip>:3141 check
  server master3 <docker swarm node 3 ip>:3141 check
```

Known Issues

The following section describes the known issues you might encounter with the High Availability solution and how to address those issues.

Docker container on last swarm node cannot communicate with other swarm nodes

This is an issue with the Encapsulating Security Payload (ESP) protocol not being enabled in firewalld. You can enable the ESP protocol with the firewalld docker-swarm script.

To address this issue, add the following firewall rule to each node:

```
firewall-cmd --add-protocol=esp --permanent
```

```
firewall-cmd --reload
```

Couchbase service does not start, remains at nc -z localhost

To address this issue, stop the container where this is happening and remove its persistent volume:

```
rm -rf /var/data/couchbase
```

Couchbase-worker fails to connect to master

A connection failure might happen a few times when a stack is freshly deployed. You can ignore these messages, and the worker should eventually connect to the master.

Couchbase database stops unexpectedly and the disk is full

If you are running large or customized workloads, you might encounter a situation where Couchbase stops unexpectedly because the disk is full. To prevent this situation, review the considerations in *What should I do if the Couchbase disk is full, indexer is crashing, and the database is unusable?*.

Couchbase rebalance fails with "Rebalance exited" error

In this situation, you received the following error:

```
Rebalance exited with reason {service_rebalance_failed,index,
```

```
{linked_process_died,<12807.821.0>,
```

```
{no_connection,"index-service_api"}
```

```
}}
```

If the Couchbase rebalance fails on the initial rejoin of a failed node into a cluster, you should check the index states and wait until the indexes are no longer in a warmup state. After the indexes are created on that node, the rebalance should succeed.

When setting up a three-node High Availability Couchbase cluster, the second node does not appear

In this situation, if you have cloned any of the nodes, the nodes might think that there is a split-brain condition.

To address this issue, delete the Couchbase data on the newly added nodes by running the following command on each node:

```
rm -rf /var/data/couchbase/*
```

The PowerFlow user interface fails to start after a manual failover of the swarm node

To address this issue, run the following commands on the relevant node:

```
docker stack rm iservices
```

```
systemctl restart docker
```

```
docker stack deploy -c docker-compose.yml iservices
```

The PowerFlow user interface returns 504 errors

Ensure that your PowerFlow systems have been updated with `yum upgrade`.

NTP should be used, and all node times should be in sync

If all nodes time are not in sync, you might experience issues with the iservices_steprunners.

The following is an example of a Docker Swarm error caused by the time not being in sync:

```
Error response from daemon: certificate (1 - 2v4umws4pxag6kbxaelwfl3vf)
not valid before Fri, 30 Nov 2018 13:47:00 UTC, and it is currently Fri,
30 Nov 2018 06:41:24 UTC: x509: certificate has expired or is not yet
valid
```

For more information, see *Preparing the PowerFlow System for High Availability*.

Example Logs from Flower

```
iservices_flower.1.jg6glaf298d2@is-scale-05 | [W 181023 20:17:40
state:113] Substantial drift from celery@1ee384863e37 may mean clocks are
out of sync. Current drift is iservices_flower.1.jg6glaf298d2@is-scale-05
| 18 seconds. [orig: 2018-10-23 20:17:40.090473 recv: 2018-10-23
20:17:58.486666]
```

Appendix

B

Configuring the SL1 PowerFlow System for Multi-tenant Environments

Overview

This appendix describes the best practices and troubleshooting solutions for deploying PowerFlow in a multi-tenant environment that supports multiple customers in a highly available fashion. This section also covers how to perform an upgrade of PowerFlow with minimal downtime.

This chapter covers the following topics:

Quick Start Checklist for Deployment 339
Deployment 339
Advanced RabbitMQ Administration and Maintenance 343
Creating Specific Queues for Customers 344
PowerFlow Queue FAQs 351
Failure Scenarios 355
Examples and Reference 359
Test Cases 373
Backup Considerations 374
Resiliency Considerations 375
Additional Sizing Considerations 377
Scaling the PowerFlow Devpi Server 378
Node Placement Considerations 382
Common Problems, Symptoms, and Solutions 383
Common Resolution Explanations 389

Quick Start Checklist for Deployment

1. Deploy and cluster the initial High Availability stack. Label these nodes as "core".
2. Create the PowerFlow configuration object for the new PowerFlow systems. The configuration object includes the SL1 IP address, the ServiceNow user and domain, and other related information.
3. Deploy and cluster the worker node or nodes for the customer.
4. Label the worker node or nodes specifically for the customer.
5. Update the **docker-compose.yml** file on a core node:
    - Add two steprunner services for each customer, one for real-time eventing, and one for backlogged events, labeled based on the organization name: *acme* and *acme-catchups*.
    - Update the new steprunner hostnames to indicate who the steprunner works for.
    - Update the new steprunner deploy constraints to deploy only to the designated labels.
    - Update the new steprunner *user_queues* environment variable to only listen on the desired queues.
6. Schedule the required PowerFlow integrations:
    - Run Device Sync daily, if desired
    - Correlation queue manager running on the catchup queue
7. Modify the Run Book Automations in SL1 to trigger the integration to run on the queue for this customer:
    - Modify the IS_PASSTHROUGH dictionary with "queue" setting.
    - Specify the configuration object to use in PowerFlow for this SL1 instance.

Deployment

The following sections describe how to deploy PowerFlow in a multi-tenant environment. After the initial High Availability (HA) core services are deployed, the multi-tenant environment differs in the deployment and placement of workers and use of custom queues.

Core Service Nodes

For a multi-tenant deployment, ScienceLogic recommends that you dedicate at least three nodes to the ***core*** PowerFlow services. These core PowerFlow services are shared by all workers and customers. As a result, it is essential that these services are clustered to handle failovers.

Because these core services are critical, ScienceLogic recommends that you initially allocate a fairly large amount of resources to these services. Allocating more resources than necessary to these nodes allows you to further scale workers in the future. If these nodes become overly taxed, you can add another node dedicated to the core services in the cluster.

These core services nodes are dedicated to the following services:

- API
- UI
- RabbitMQ
- Couchbase
- Redis

It is critical to monitor these core service nodes, and to always make sure these nodes have enough resources for new customers and workers as they are on-boarded.

To ensure proper failover and persistence of volumes and cluster information, the core services must be pinned to each of the nodes. For more information, see *Configuring Core Service Nodes*, below.

Requirements

Three nodes (or more for additional failover support) with six CPUs and 56 GB memory each.

Configuring Core Service Nodes

- Install the PowerFlow RPM on your core three nodes.
- See the *High Availability section* for information about how to join the cluster as a manager, and copy **the /etc/iservices/encryption_key** and **/etc/iservices/is_pass** file from a core service node to the new worker node (same location and permissions).
- *Create a label on the node* and label these nodes as "core node".
- See the *Configuring Clustering and High Availability* section for details on clustering Couchbase and RabbitMQ, and an example compose file of this setup.
- *Update the contentapi, UI, and redis services* so that those services are only ever deployed onto the core nodes.

Critical Elements to Monitor on Core Nodes

- Memory utilization: Warnings at 80%
- CPU utilization: Warnings at 80%
- RabbitMQ queue sizes (can also be monitored from the Flower API, or the PowerFlow user interface)

Worker Service Nodes

Separate from the core services are the ***worker services***. These worker services are intended to be deployed on nodes separate from the core services, and other workers, and these worker services aim to provide processing only for specified dedicated queues. Separating the VMs or modes where worker services are deployed will ensure that one customer's workload, no matter how heavy it gets, will not negatively affect the other core services, or other customer workloads.

Requirements

The resources allocated to the worker nodes depends on the worker sizing chosen, the more resources provided to a worker, the faster their throughput. Below is a brief guideline for sizing. Please note that even if you exceed the number of event syncs per minute, events will be queued up, so the sizing does not have to be exact. The below sizing just provides a suggested guideline.

Event Sync Throughput Node Sizing

CPU	Memory	Worker count	Time to sync a queue full of 10,000 events	Events Synced per second
2	16 GB	6	90 minutes	1.3
4	32 GB	12	46 minutes	3.6
8	54 GB	25	16.5 minutes	10.1

Test Environment and Scenario

- Each Event Sync consists of PowerFlow workers reading from the pre-populated queue of 10000 events. The sync interprets, transforms, and then POSTS the new event as a correlated ServiceNow incident into ServiceNow. This process goes on to then query ServiceNow for the new sysID generated for the incident, transforms it, and then POSTs it back to SL1 as an external ticket to complete the process.
- Tests were performed on a node of workers only.
- Tests were performed with a 2.6 GHz virtualized CPU in a vCenter VM. Both SL1 and ServiceNow were responding quickly when doing so.
- Tests were performed with a pre-populated queue of 10000 events.
- Tests were performed with the current deployed version of Cisco custom integration. Data will again be gathered for the next version when it is completed by Pro Services.
- Each event on the queue consisted of a single correlated event.

Configuring the Worker Node

- Install the PowerFlow RPM on the new node.
- See the *High Availability section* for information about how to join the cluster as a manager or worker, and copy the **/etc/iservices/encryption_key** and **/etc/iservices/is_pass** file from a core service node to the new worker node (same location and permissions).
- By default, the worker will listen on and accept work from the default queue, which is used primarily by the user interface, and any integration run without a custom queue.
- To configure this worker to run customer-specific workloads with custom queues, see *Onboarding a Customer*.
- Modify the **docker-compose.yml** on a core service node accordingly.
- If you just want the node to accept default work, the only change necessary is to increase worker count using the table provided in the requirements section

- If you want the node to be customer specific, be sure to add the proper labels and setup custom queues for the worker in the docker-compose when deploying. This information is contained in the Onboarding a customer section.

Initial Worker Node Deployment Settings

It is required that there is always at least one worker instance listening on the default queue for proper functionality. The default worker can run in any node.

Worker Failover Considerations and Additional Sizing

When deploying a new worker, especially if it is going to be a custom queue dedicated worker, it is wise to consider deploying an extra worker listening on the same queues. If you have on a single worker node listening to a dedicated customer queue, there is potential for that queue processing to stop completely if that single node worker fails.

For this reason, ScienceLogic recommends that for each customer dedicated worker you deploy, you deploy a second one as well. This way there are two nodes listening to the customer dedicated queue, and if one node fails, the other node will continue processing from the queue with no interruptions.

When deciding on worker sizing, it's important to take this into consideration. For example, if you have a customer that requires a four-CPU node for optimal throughput, an option would be to deploy two nodes with two CPUs, so that there is failover if one node fails.

- How to know when more resources are necessary
- Extra worker nodes ready for additional load or failover

Knowing When More Resources are Necessary for a Worker

Monitoring the memory, CPU and pending integrations in queue can give you an indication of whether more resources are needed for the worker. Generally, when queue times start to build up, and tickets are not synced over in an acceptable time frame, more workers for task processing are required.

Although more workers will process more tasks, they will be unable to do so if the memory or CPU required by the additional workers is not present. When adding additional workers, it is important to watch the memory or CPU utilization, so long as the utilization is under 75%, it should be okay to add another worker. If utilization is consistently over 80%, then you should add more resources to the system before addling additional workers.

Keeping a Worker Node on Standby for Excess Load Distribution

Even if you have multiple workers dedicated to a single customer, there are still scenarios in which a particular customer queue spikes in load, and you'd like an immediate increase in throughput to handle this load. In this scenario you don't have the time to deploy a new PowerFlow node and configure it to distribute the load for greater throughput, as you need increased load immediately.

This can be handled by having a node on standby. This node has the same PowerFlow RPM version installed, and sits idle in the stack (or is turned off completely). When a spike happens, and you need more resources to distribute the load, you can then apply the label to the corresponding to the customer who's queues spiked. After setting the label on the standby node, you can scale up the worker count for that particular customer. Now, with the stand-alone node labeled for work for that customer, additional worker instances will be distributed to and started on the standby node.

When the spike has completed, you can return the node to standby by reversing the above process. Decrease the worker count to what it was earlier, and then remove the customer specific label from the node.

Critical Elements to Monitor in a Steprunner

- Memory utilization: Warnings at 80%
- CPU utilization: Warnings at 80%
- Successful, failed, active tasks executed by steprunner (retrievable from Flower API or PowerPack)
- Pending tasks in queue for the worker (retrievable by Flower API or PowerPack)
- Integrations in queue (similar information here as in pending tasks in queue, but this is retrievable from the PowerFlow API).

Advanced RabbitMQ Administration and Maintenance

This section describes how multi-tenant deployments can use separate virtual hosts and users for each tenant.

Using an External RabbitMQ Instance

In certain scenarios, you might not want to use the default RabbitMQ queue that is prepackaged with PowerFlow. For example, you might already have a RabbitMQ production cluster available that you just want to connect with PowerFlow. You can do this by defining a new virtual host in RabbitMQ, and then you configure the PowerFlow broker URL for contentapi, steprunner, scheduler services so that they point to the new virtual host.

> Any use of an external RabbitMQ server will not be officially supported by ScienceLogic if there are issues in the external RabbitMQ instance.

Setting a User other than Guest for Queue Connections

When communicating with RabbitMQ in the swarm cluster, all communication is encrypted and secured within the overlay Docker network.

To add another user, or to change the user that PowerFlow uses when communicating with the queues:

1. Create a new user in RabbitMQ that has full permissions to a virtual host. For more information, see the RabbitMQ documentation.
2. Update the **broker_url** environment variable with the new credentials in the **docker-compose** file and then re-deploy.

Configuring the Broker (Queue) URL

When using an external RabbitMQ system, you need to update the ***broker_url*** environment variable in the contentapi, steprunner, and scheduler services. You can do this by modifying the environment section of the services in **docker-compose** and changing ***broker_url***. The following line is an example:

```
broker_url: 'pyamqp://username:password@rabbitmq-hostname/v-host'
```

Creating Specific Queues for Customers

When a new SL1 system is to be onboarded into PowerFlow, by default their integrations are executed on the default queue. In large multi-tenant environments, ScienceLogic recommends separate queues for each customer. If desired, each customer can also have specific queues.

For more information about queues, see *PowerFlow Queue FAQs*.

Create the Configuration Object

The first step to setting up a new PowerFlow system is to create a configuration object with variables that will satisfy all PowerFlow applications. The values of these should be specific to the new system (such as SL1 IP address, username, password).

See the *example configuration* for a template you can fill out for new system.

Because integrations might update their variable names from EM7 to SL1 in the future, ScienceLogic recommends to cover variables for both **em7_** and **sl1_**. The *example configuration* contains this information.

Label the Worker Node Specific to the Customer

For an example label, if you want a worker node to be dedicated to a customer called "acme", you could create a node label called "customer" and make the value of the label "acme". Setting this label now makes it easier to cluster in additional workers and distribute load dynamically in the future.

Creating a Node Label

This topic outlines creating a label for a node. Labels provide the ability to deploy a service to specific nodes (determined by labels) and to categorize the nodes for the work they will be performing. Take the following actions to set a node label:

```
# get the list of nodes available in this cluster (must run from a manager node)

docker node ls

# example of adding a label to a docker swarm node

docker node update --label-add customer=acme <node id>
```

Placing a Service on a Labeled Node

After you create a node label, refer to the example below for updating your **docker-compose-override.yml** file and ensuring the desired services deploy to the matching labeled nodes:

```
# example of placing workers on a specific labeled node

steprunner-acme:
```

```
...
deploy:
  placement:
    constraints:
    - node.labels.customer == acme
  resources:
    limits:
      memory: 1.5G
  replicas: 15
...
```

Creating a Queue Dedicated to a Specific Application or Customer

You can create a new queue that is specific to a PowerFlow application or to a customer to ensure that work and events created from one system will not affect or slow down work created from another system, provided the multi-tenant system has enough resources allocated.

In the example below, we created two new queues in addition to the default queue, and allocated workers to it. Both of these worker services use separate queues as described below, but run on the same labeled worker node.

New Queues to Deploy:

- ***acmequeue***. The queue we use to sync events specific from a customer called "acme". Only events syncs and other integrations for "acme" will run on this queue.
- ***acmequeue-catchup***. The queue where any old events that should have synced over already (but failed due to PowerFlow not being available or other reason) will run. Running these catchup integrations on a separate queue ensures that real-time event syncing isn't delayed in favor of an older event catching up.

Add Workers for the New Queues

First, define additional workers in our stack that are responsible for handling the new queues. All modifications are made in **docker-compose-override.yml**:

1. Copy an existing steprunner service definition.
2. Change the steprunner service name to something unique for the stack. For this example, use the following names:
    - steprunner-acmequeue
    - steprunner-acmequeue-catchup
3. Modify the `replicas` value to specify how many workers should be listening to this queue:

- steprunner-acmequeue will get 15 workers because it is expecting a very heavy load
- steprunner-acmequeue-catchup will get three workers because it will not run very often

4. Add a new environment variable labeled `user_queues`. This environment variable tells the worker what queues to listen to:
    - steprunner-acmequeue will set user_queues= "acmequeue"
    - steprunner-acmequeue-catchup will set user_queues="acmequeue-catchup"
5. To ensure that these workers can be easily identified for the queue to which they are assigned, modify the hostname setting:
    - Hostname: "acmequeue-{{.Task.ID}}"
    - Hostname "acmequeue-catchup-{{.Task.ID}}"
6. After the changes have been made, run **/opt/iservices/script/compose-override.sh** to validate that the syntax is correct.
7. When you are ready to deploy, re-run the docker stack deploy with the new compose file.

Code Example: docker-compose entries for new steprunners

After these changes have been made, your docker-compose entries for the new steprunners should look similar to the following (the values relevant to this procedures display in **bold**, below).

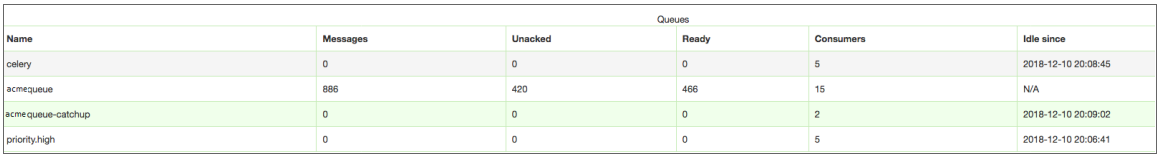
```
steprunner-acme-catchup:
  image: sciencelogic/is-worker:latest
  hostname: "acme-catchup-{{.Task.ID}}"
  deploy:
    placement:
      constraints:
        - node.labels.customer == acme
    resources:
      limits:
        memory: 2G
    replicas: 3
  environment:
    user_queues: 'acmequeue-catchup'
  ..
  ..
  ..

steprunner-acme:
  image: sciencelogic/is-worker:latest
  hostname: "acmequeue-{{.Task.ID}}"
  deploy:
    placement:
```

```
      constraints:
        - node.labels.customer == acme
    resources:
      limits:
        memory: 2G
    replicas: 15
environment:
    user_queues: 'acmequeue'
..
..
..
```

Once deployed via Docker Stack deploy, you should see the new workers in Flower, as in the following image:

You can verify the queues being listened to by looking at the "broker" section of Flower, or by clicking into a worker and clicking the **[Queues]** tab:



Adding a PowerFlow Application to a Specific Queue

To add a PowerFlow application to a specific queue:

1. Use Postman or cURL to do a GET to load the list of PowerFlow applications:

   ```
   GET <PowerFlow_hostname>/api/v1/applications
   ```

   where `<PowerFlow_hostname>` is the IP address or URL for your PowerFlow system.

2. Locate the "id" value for the PowerFlow application name you want to use, and include that value to load the specific application by name:

   ```
   GET <PowerFlow_hostname>/api/v1/applications/<application_name>
   ```

   For example:

   ```
   GET 10.1.1.22/api/v1/applications/interface_sync_sciencelogic_to_
   servicenow
   ```

3. Copy the entire JSON code and save it to a file with the same name as the application from step 2.

4. Edit the JSON code for the application by adding the following line to the initial code block, after `"id"` or `"progress"`:

   ```
   "queue": "<queue_name>"
   ```

   For example:

   ```
   "queue": "acmequeue"
   ```

5. Upload the updated application using the iscli tool:

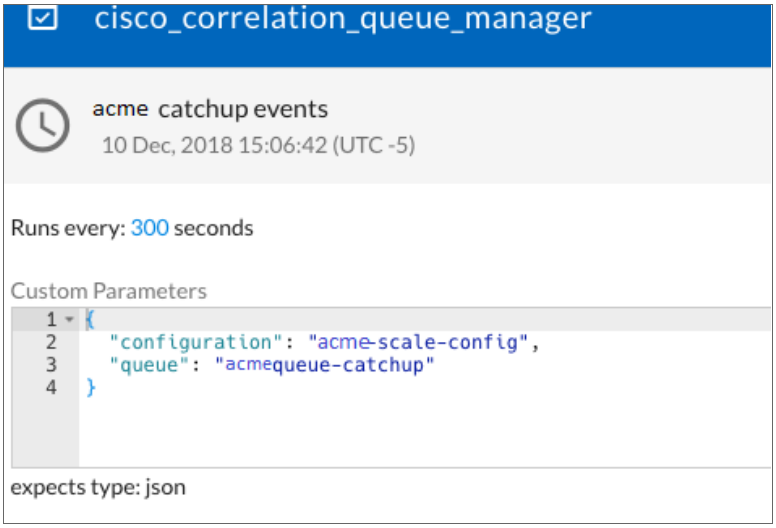
   ```
   -uaf <application-name> -H PowerFlow_hostname> -p <password>
   ```

Create Application Schedules and Automation Settings to Utilize Separate Queues

After the workers have been configured with specific queue assignments, schedule your PowerFlow applications to run on those queues, and configure Run Book Automations (RBAs) to place the applications on those queues.

Scheduling an Application with a Specific Queue and Configuration

To run an application on a specific queue using a configuration for a particular system, you can use the ***Custom Parameters*** override available in the scheduler. Below is an example of the scheduled application that utilizes the **acmecqueue-catchup** queue:



In the example above, the **cisco_correlation_queue_manager** is scheduled to run every 300 seconds, using the acme configuration, and will run on the **acmequeue**. You can have any number of scheduled runs per application. If we were to add additional customers, we would add a new schedule entry with differing configurations, and queues for each.

Configuring Applications to Utilize a Specific Queue and Configuration

The last step to ensuring integrations for your newly onboarded SL1 system is to update the Run Book Automations in SL1 to provide the configuration and queue to use when the Run Book Automation triggers an event.

Modify the Event-correlation policy with the following changes:

1. IS4_PASSTHROUGH= {"queue":"acmequeue"}
2. CONFIG_OVERRIDE= 'acme-scale-config'

PowerFlow Queue FAQs

This section contains a list of frequently asked questions and answers about queues in PowerFlow.

What is RabbitMQ, and what messages are placed in it?

RabbitMQ is the queueing service used for most PowerFlow deployments. When a PowerFlow application is run, each step of the application (and a few other internal steps) gets placed in the queue for processing.

Some customers might have multiple queues corresponding to specific workloads.

What does it mean when the queue reports a high message count?

When a queue reports a high message count, it means that there are many tasks in the queue waiting to be processed. When this occurs, additional tasks placed in the queue might be delayed in their execution until the previous tasks have completed. A high message count is not always an actionable concern.

When should I be concerned about a high message count?

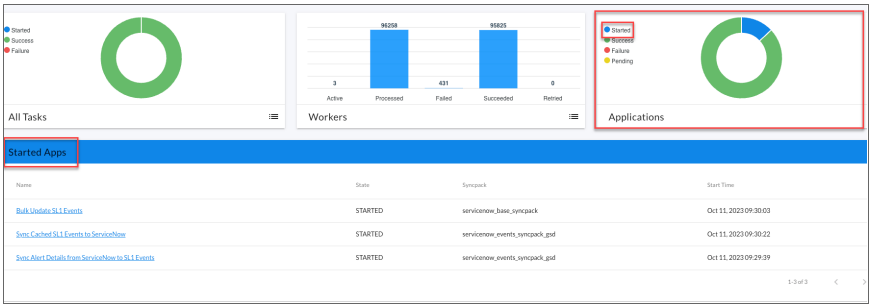
A high message count can simply mean that there is a large workflow being processed on a periodic schedule for a short period of time, like syncing many devices. If the queue goes up and then back down periodically, especially at the same time every day, it could be normal behavior depending on the customer's schedules, and as a result should not be a cause for concern.

A high message count could be an issue if:

- The queue has been increasing steadily over a duration of time and never reducing.
- There is no pattern of high queue count at this time of day for that system.

How can I tell what is currently in queue to be processed?

In the PowerFlow user interface, go to the **Control Tower** page and scroll down to the **All Applications** circle graph. If you click the *Pending* (yellow) or *Started* (blue) elements on the circle graph, a list of applications in that state displays below the graphs:



Alternatively, you can check the steprunner logs to see exactly what they are processing, or you can access Flower (the tool used for monitoring PowerFlow tasks and workers) at **https://*<IP of PowerFlow>*/flower/workers**.When viewing tasks on the **[Tasks]** tab, look in the **kwargs** column for the `sn (step name)` and `an (app name)` values to see what the task is for.

How can I tell what caused the queue backlog?

A queue backlog might be caused by two possible scenarios:

1. **Over-scheduling applications in PowerFlow (most likely)**. In this situation, PowerFlow applications have been scheduled more frequently than they actually take to complete a run. In other words, if an application is scheduled to run every minute, but the app actually takes five minutes to run, there will inevitably be a backlog.

   To check if this is the case, review the current schedules for your PowerFlow applications. If there are any schedules that are set to run frequently (multiple times an hour), check how long they are taking to run by adding up the steps' run time (visible in step logs),

2. **Event Flood triggered from SL1**. In this situation, an event flood of more events than typical coming from SL1 causes of a high message count.

   To check if this is the case, log in to SL1 and check to make sure that the run book automation policies are reasonable, and ensure that any run book automations for ServiceNow are not configured to run on notice events, for example.

   The following is an example of a n SL1 database query that can be used to show run book automation triggers over time:

```
select date_format(notify_stamp, '%Y-%m-%d %H') as date, count(*)
from master_events.events_cleared

where notify_count>0 and notify_stamp > now() - interval 34 hour

group by date order by date desc;
```

What do I do if the high message count was caused by over-scheduling?

If the cause of the queue backlog is due to over-scheduling, then you must assess which PowerFlow applications are taking too long, and then correct the schedule for those applications. Additionally, you should try to understand why these applications were scheduled so frequently in the first place:

- Did you simply make a mistake and over-schedule unintentionally?
- Is there a reason that your apps are now taking longer to run than before? A typical cause of this is a large amount of orphaned open incidents in ServiceNow; reducing the number of these events will reduce app run time. Another potential cause is if you recently onboarded many additional devices, and your schedules need to be adjusted for that.
- If you are using custom applications that need to be run often, you should investigate why these applications need to run so frequently, and you should assess whether your PowerFlow instance needs to be upsized.

What do I do if the high message count was caused by an SL1 event flood?

If the cause of the queue backlog was due to an SL1 event flood, the best thing to do is to determine what events caused it, and whether policies need to be changed:

- Assess the run book automation policies that triggered the event flood to see if there is a recommended update, such as not triggering an event for notice level events

- If you feel that your automation policies need to be extremely granular and should not change, you might need to assess whether your PowerFlow instance needs to be upsized.

After the event flood has been reconciled, you can clear the queue to return to fast processing.

How can I clear messages from the queue?

You can clear messages from the queue in two ways (eventually this will be possible from the **Control Tower** page):

1. **From the RabbitMQ user interface**:
    - Click the **[Queues]** tab and select the queue you would like to purge.
    - Scroll down to the ***purge*** drop down.
    - Select and execute the **[purge messages]** button.
2. **From the PowerFlow node**, run the following command:

```
docker exec $(docker ps -q --filter name=iservices_steprunner*|head -n 1) celery --app=ipaascommon.celeryapp:app purge -f -Q celery
```

> **TIP:** Change `-Q celery` to `-Q <other-queue-name>` to purge a different queue than the default queue.

Why are PowerFlow applications still showing as "Pending" after I cleared the queue?

PowerFlow applications that are initially placed in the queue, but have not yet run will be placed in the "Pending" status. Should those tasks be forcefully purged from the queue, the application will never have a chance to change to another state.

After clearing a queue, it is expected that previously queued applications (which are now cleared) will display as "Pending", and they will never run.

Rather than looking at the previous runs, you should validate processing behavior by triggering a new run of any application. If the queue backlog is cleared, you should see the application go from "Pending" to "Started" almost instantaneously.

Why are messages stuck in the broadcast queue in RabbitMQ?

The syncpack_steprunners use the broadcast queue only to install SyncPacks across all nodes. Every syncpack_steprunner (which runs on every node) creates its own broadcast queue.

If a syncpack_steprunner is restarted, it creates a new broadcast queue, leaving the old one from the previous replica around. When those broadcast queues from old syncpack_steprunner containers are left around, and a user clicks "activate/install syncpack", that message gets placed on all broadcast queues, including the ones left around from old workers.

You can use the `pfctl remove_rabbit_non_consumer_queues` to clean up these unwanted queues, and that action is also called by the **autoheal** action.

Failure Scenarios

This topic cover how PowerFlow handles situations where certain services fail.

Worker Containers

In case of failure, when can the worker containers be expected to restart?

- The worker containers have a strict memory limit of 2 GB. These containers may be restarted if the service requests more memory than the 2 GB limit.
- The restart is done by a SIGKILL of the OOM_Killer on the Linux system.

What happens when a worker container fails?

- Worker containers are ephemeral, and simply execute the tasks allotted to them. There is no impact to a worker instance restarting.

What processing is affected when service is down?

- The task_reject_on_worker_lost environment variable dictates whether the task being executed at the time the worker was restarted will be re-queued for execution by another worker. (default false)
- For more information about Celery, the task-processing framework used by PowerFlow): http://docs.celeryproject.org/en/latest/userguide/configuration.html#task-reject-on-worker-lost

What data can be lost?

- Workers contain no persistent data, so there is no data to lose, other than the data from the current task that is being executed on that worker when its shut down (if there is one)
- Any PowerFlow application that fails can be replayed (and re-executed by the workers) on demand with the **/api/v1/tasks/<task-id>/replay** endpoint.

API

When can the API be expected to restart?

- The API also has a default memory limit. As with the steprunners (worker containers), if the memory limit is reached, the API is restarted by a SIGKILL of the OOM_Killer on the Linux system to prevent a memory leak.

What happens when it fails?

- On the clustered system, there are always three contentapi services, so if one of the API containers fails, API requests will still be routed to the functioning containers through the internal load balancer.

What processing is affected when service is down?

- If none of the API services are up and running, any Run Book Automation calls to sync an incident through PowerFlow results in an error. The next time that scheduled integration runs, the integration recognizes the events that failed to send to PowerFlow, and the integration will process them so that the events sync.

- Even if the API is down, the events that were generated while it was down will be synced by the scheduled application. PowerFlow will reach out to SL1 for those items that SL1 failed to post to the PowerFlow.

What data can be lost?

- The API contains no persistent data, so there is no risk of data loss.

Couchbase

If a core service node running Couchbase fails, the database should continue to work normally and continue processing events, as long as a suitable number of clustered nodes are still up and running. Three core service nodes provides automatic failover handling of one node, five core service nodes provides automatic failover handling of two nodes, and so on. See the *High Availability section* for more information.

If there are enough clustered core nodes still running, the failover will occur with no interruptions, and the failing node can be added back at any time with no interruptions.

> **NOTE**: For optimal performance and data distribution after rejoining a cluster, you can click the **[Re-balance]** button from the Couchbase user interface, if needed.

If there are not enough clustered core nodes still running, then you will manually have to fail over the Couchbase Server. In this scenario, since automatic failover could not be performed (due to too few nodes available), there will be disruption in event processing. For more information, see the *Manual Failover section*.

In case of failure, when can Couchbase be expected to restart?

- In ideal conditions, the Couchbase database should not restart, although Couchbase might be restarted when the node it is running on is over-provisioned. For more information, see the *known issue*.

What happens when it fails?

- Each Couchbase node in the cluster contains a fully replicated set of data. If any single node fails, automatic failover will occur after the designated time (120 seconds by default). Automatic failover, processing, and queries to the database will continue without issue.
- If the database simply is restarted and not down for a long period of time (120 seconds), then the system will not automatically fail over, and the cluster will still be maintained.
- If two out of three of the database nodes fail for a period of time, processing may be paused until a user takes manual failover action. These manual actions are documented in the *Manual Failover section*.

What processing is affected when service is down?

- In the event of an automatic failover (1/3 node failure), no processing will be affected and queries to the database will still be functional.
- In the event of a large failure (2/3 node failure) automatic failover will not happen, and manual intervention may be needed to so you can query the database again.

What data can be lost?

- Every Couchbase node has full data replication between each of the three nodes. In the event of a failure of any of the nodes, no data is lost, as a replicated copy exists across the cluster.

RabbitMQ

RabbitMQ clustered among all core service nodes provides full mirroring to each node. So long as there is at least one node available running RabbitMQ, the queues should exist and be reachable. This means that a multiple node failure will have no effect on the RabbitMQ services, and it should continue to operate normally.

In case of failure, when can RabbitMQ be expected to restart?

- Similar to the Couchbase database, in a smooth-running system, RabbitMQ should never really restart.
- As with other containers, RabbitMQ might be restarted when the node it is running on is over-provisioned. For more information, see the ***known issue***.

What happens when RabbitMQ fails?

- All RabbitMQ nodes in the cluster mirror the other queues and completely replicate the data between them. The data is also persisted.
- In the event of any RabbitMQ node failure, the other nodes in the cluster will take over responsibility for processing its queues.
- If all RabbitMQ nodes are restarted, their messages are persisted to disk, so any tasks or messages sitting in queue at the time of the failure are not lost, and are reloaded once the system comes back up.
- In the event of a network partition (***"split-brain" state***) RabbitMQ will follow the configured partition handling strategy (default autoheal).
- For more information, see https://www.rabbitmq.com/partitions.html#automatic-handling.

What processing is affected when service is down?

- When this service is down completely (no nodes running), the API may fail to place event sync tasks onto the queue. As such, any Run Book Automation calls to sync an incident through PowerFlow will result in an error.
- These failed event syncs are then placed in a database table in SL1 which PowerFlow queries on a schedule every few minutes. The next time that scheduled integration runs, the integration recognizes the events that failed to send to PowerFlow, and the integration will process them so that the events sync.

What data can be lost?

- All data is replicated between nodes, so there is little risk of data loss.
- The only time there may be loss of tasks in queues is if there is a network partition, also called a ***"split-brain" state***.

PowerFlow User Interface

In case of failure, when can the user interface be expected to restart?

- The PowerFlow user interface (GUI) should never be seen as restarted unless a user forcefully restarted the interface.
- The PowerFlow user interface might be restarted when the node it is running on is over-provisioned. For more information, see the ***known issue***.

What happens when it fails?

- The GUI service provides the proxy routing throughout the stack, so if the GUI service is not available, Run Book Automation POSTS to PowerFlow will fail. However, as with an API failure, if the Run Book Actions can not POST to PowerFlow, those events will be placed in a database table in SL1 that PowerFlow queries on a schedule every few minutes. The next time that scheduled integration runs, the integration recognizes the events that failed to send to PowerFlow, and the integration will process them so that the events sync.
- When the GUI service is down and SL1 cannot POST to it, the syncing of the events might be slightly delayed, as the events will be pulled in and created with the next run of the scheduled integration.

What data can be lost?

- The PowerFlow user interface persists no data, so there is no risk of any data loss.

Redis

If the Redis service fails, it will automatically be restarted, and will be available again in a few minutes. The impact of this happening, is that task processing in PowerFlow is delayed slightly, as the worker services pause themselves and wait for the Redis service to become available again.

Consistent Redis failures

Consistent failures and restarts in Redis typically indicate your system has too little memory, or the Redis service memory limit is set too low, or not low at all. PowerFlow ships with a default memory limit of 8 GB to ensure that the Redis service only ever uses 8 GB of memory, and it ejects entries if it is going to go over that limit. This limit is typically sufficient, though if you have enough workers running large enough integrations to overfill the memory, you may need to increase the limit.

Before increasing Redis memory limit, be sure that there is suitable memory available to the system.

Known Issue for Groups of Containers

If you see multiple containers restarting at the same time on the same node, it indicates an over-provisioning of resources on that node. This only occurs on Swarm manager nodes, as the nodes are not only responsible for the services they are running, but also for maintaining the Swarm cluster and communicating with other manager nodes.

If resources become over-provisioned on one of those manager nodes (as they were with the core nodes when we saw the failure), the Swarm manager will not be able to perform its duties and may cause a docker restart on that particular node. This failure is indicated by "context deadline exceeded", and "heartbeat failures" in the logs from running `journalctl --no-page |grep docker |grep err`.

This is one of the reasons why docker recommends running "manager-only" nodes, in which the manager nodes are only responsible for maintaining the Swarm, and not responsible for running other services. If any nodes that are running PowerFlow services are also a Swarm manager, make sure that the nodes are not over-provisioned, otherwise the containers on that node may restart. For this reason, ScienceLogic recommends monitoring and placing thresholds at 80% utilization.

To combat the risk of over-provisioning affecting the docker Swarm manager, apply resource constraints on the services for the nodes that are also Swarm managers, so that docker operations always have some extra memory or CPU on the host to do what they need to do. Alternatively, you can only use drained nodes, which are not running any services, as Swarm managers, and not apply any extra constraints.

For more information about Swarm management, see https://docs.docker.com/engine/Swarm/admin_guide/.

Examples and Reference

Code Example: A Configuration Object

```
[
  {
    "encrypted": false,
    "name": "em7_host",
    "value": "<ip_address>"
  },
  {
    "encrypted": false,
    "name": "sl1_host",
    "value": "${config.em7_host}"
  },
  {
    "encrypted": false,
    "name": "sl1_id",
    "value": "${config.em7_id}"
  },
  {
    "encrypted": false,
    "name": "sl1_db_port",
    "value": 7706
  },
```

```
{
  "encrypted": false,
  "name": "snow_host",
  "value": "<instance>.service-now.com"
},
{
  "encrypted": true,
  "name": "em7_password",
  "value": "<password>"
},
{
  "encrypted": false,
  "name": "sl1_user",
  "value": "${config.em7_user}"
},
{
  "encrypted": false,
  "name": "sl1_password",
  "value": "${config.em7_password}"
},
{
  "encrypted": false,
  "name": "sl1_db_user",
  "value": "${config.em7_db_user}"
```

```
},
{
  "encrypted": false,
  "name": "sl1_db_password",
  "value": "${config.em7_db_password}"
},
{
  "encrypted": false,
  "name": "em7_user",
  "value": "<username>"
},
{
  "encrypted": false,
  "name": "em7_db_user",
  "value": "root"
},
{
  "encrypted": false,
  "name": "em7_db_password",
  "value": "<password>"
},
{
  "encrypted": false,
  "name": "snow_user",
```

```
    "value": "<username>"
  },
  {
    "encrypted": true,
    "name": "snow_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "Domain_Credentials",
    "value": {
      "c9818d2c4a3623120162443385189 4bb": {
        "password": "3m7Admin!",
        "user": "is4DomainUser2"
      }
    }
  },
  {
    "name": "region",
    "value": "ACMEScaleStack"
  },
  {
    "encrypted": false,
    "name": "em7_id",
```

```
    "value": "${config.region}"
  },
  {
    "encrypted": false,
    "name": "generate_report",
    "value": "true"
  }
]
```

Code Example: A Schedule Configuration Object

```
[
  {
    "application_id": "device_sync_sciencelogic_to_servicenow",
    "entry_id": "dsync every 13 hrs acme",
    "last_run": null,
    "params": {
      "configuration": "acme-scale-config",
      "mappings": {
        "cmbd_ci_ip_router": [
          "Cisco Systems | 12410 GSR",
          "Cisco Systems | AIR-AP1141N",
          "Cisco Systems | AP 1200-IOS",
          "Cisco Systems | Catalyst 5505"
        ],
        "cmdb_ci_esx_resource_pool": [
```

```
        "VMware | Resource Pool"
    ],
    "cmdb_ci_esx_server": [
        "VMware | ESXi 5.1 w/HR",
        "VMware | Host Server",
        "VMware | ESX(i) 4.0",
        "VMware | ESX(i) w/HR",
        "VMware | ESX(i) 4.0 w/HR",
        "VMware | ESX(i)",
        "VMware | ESX(i) 4.1 w/HR",
        "VMware | ESXi 5.1 w/HR",
        "VMware | ESXi 5.0 w/HR",
        "VMware | ESX(i) 4.1",
        "VMware | ESXi 5.1",
        "VMware | ESXi 5.0"
    ],
    "cmdb_ci_linux_server": [
        "ScienceLogic, Inc. | EM7 Message Collector",
        "ScienceLogic, Inc. | EM7 Customer Portal",
        "ScienceLogic, Inc. | EM7 All-In-One",
        "ScienceLogic, Inc. | EM7 Integration Server",
        "ScienceLogic, Inc. | EM7 Admin Portal",
        "ScienceLogic, Inc. | EM7 Database",
        "ScienceLogic, Inc. | OEM",
```

```
    "ScienceLogic, Inc. | EM7 Data Collector",
    "NET-SNMP | Linux",
    "RHEL | Redhat 5.5",
    "Virtual Device | Content Verification"
  ],
    "cmdb_ci_vcenter": [
    "VMware | vCenter",
    "Virtual Device | Windows Services"
  ],
    "cmdb_ci_vcenter_cluster": [
    "VMware | Cluster"
  ],
    "cmdb_ci_vcenter_datacenter": [
    "VMware | Datacenter"
  ],
    "cmdb_ci_vcenter_datastore": [
    "VMware | Datastore",
    "VMware | Datastore Cluser"
  ],
    "cmdb_ci_vcenter_dv_port_group": [
    "VMware | Distributed Virtual Portgroup"
  ],
    "cmdb_ci_vcenter_dvs": [
    "VMware | Distributed Virtual Switch"
```

```
        ],
          "cmdb_ci_vcenter_folder": [
          "VMware | Folder"
        ],
          "cmdb_ci_vcenter_network": [
          "VMware | Network"
        ],
          "cmdb_ci_vmware_instance": [
          "VMware | Virtual Machine"
        ]
      },
      "queue": "acmequeue",
      "region": "ACMEScaleStack"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 47200
      },
      "schedule_type": "frequency"
    },
    "total_runs": 0
  },
  {
    "application_id": "device_sync_sciencelogic_to_servicenow",
```

```
"entry_id": "dsync every 12 hrs on .223",
"last_run": null,
"params": {
  "configuration": "em7-host-223",
  "mappings": {
    "cmdb_ci_esx_resource_pool": [
      "VMware | Resource Pool"
    ],
    "cmdb_ci_esx_server": [
      "VMware | ESXi 5.1 w/HR",
      "VMware | Host Server",
      "VMware | ESX(i) 4.0",
      "VMware | ESX(i) w/HR",
      "VMware | ESX(i) 4.0 w/HR",
      "VMware | ESX(i)",
      "VMware | ESX(i) 4.1 w/HR",
      "VMware | ESXi 5.1 w/HR",
      "VMware | ESXi 5.0 w/HR",
      "VMware | ESX(i) 4.1",
      "VMware | ESXi 5.1",
      "VMware | ESXi 5.0"
    ],
    "cmdb_ci_linux_server": [
      "ScienceLogic, Inc. | EM7 Message Collector",
```

```
    "ScienceLogic, Inc. | EM7 Customer Portal",
    "ScienceLogic, Inc. | EM7 All-In-One",
    "ScienceLogic, Inc. | EM7 Integration Server",
    "ScienceLogic, Inc. | EM7 Admin Portal",
    "ScienceLogic, Inc. | EM7 Database",
    "ScienceLogic, Inc. | OEM",
    "ScienceLogic, Inc. | EM7 Data Collector",
    "NET-SNMP | Linux",
    "RHEL | Redhat 5.5",
    "Virtual Device | Content Verification"
  ],
  "cmdb_ci_vcenter": [
    "VMware | vCenter",
    "Virtual Device | Windows Services"
  ],
  "cmdb_ci_vcenter_cluster": [
    "VMware | Cluster"
  ],
  "cmdb_ci_vcenter_datacenter": [
    "VMware | Datacenter"
  ],
  "cmdb_ci_vcenter_datastore": [
    "VMware | Datastore",
    "VMware | Datastore Cluser"
```

```
      ],
      "cmdb_ci_vcenter_dv_port_group": [
        "VMware | Distributed Virtual Portgroup"
      ],
      "cmdb_ci_vcenter_dvs": [
        "VMware | Distributed Virtual Switch"
      ],
      "cmdb_ci_vcenter_folder": [
        "VMware | Folder"
      ],
      "cmdb_ci_vcenter_network": [
        "VMware | Network"
      ],
      "cmdb_ci_vmware_instance": [
        "VMware | Virtual Machine"
      ]
    }
  },
  "schedule": {
    "schedule_info": {
      "run_every": 43200
    },
    "schedule_type": "frequency"
  },
```

```
    "total_runs": 0
  },
  {
    "application_id": "cisco_correlation_queue_manager",
    "entry_id": "acme catchup events",
    "last_run": {
      "href": "/api/v1/tasks/isapp-a20d5e08-a802-4437-92ef-32d643c6b777",
      "start_time": 1544474203
    },
    "params": {
      "configuration": "acme-scale-config",
      "queue": "acmequeue-catchup"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
      "schedule_type": "frequency"
    },
    "total_runs": 33
  },
  {
    "application_id": "cisco_incident_state_sync",
    "entry_id": "incident sync every 5 mins on .223",
```

```
    "last_run": {
      "href": "/api/v1/tasks/isapp-52b19097-e0bf-450b-948c-487aff33fc3b",
      "start_time": 1544474203
    },
    "params": {
      "configuration": "em7-host-223"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
      "schedule_type": "frequency"
      },
      "total_runs": 2815
  },
  {
    "application_id": "cisco_incident_state_sync",
    "entry_id": "incident sync every 5 mins acme",
    "last_run": {
      "href": "/api/v1/tasks/isapp-dde1dba5-2343-4026-8801-35a02e4e57a1",
      "start_time": 1544474202
    },
      "params": {
        "configuration": "acme-scale-config",
```

```
        "queue": "acmequeue"
      },
      "schedule": {
        "schedule_info": {
          "run_every": 300
        },
        "schedule_type": "frequency"
    },
    "total_runs": 1587
},
{
    "application_id": "cisco_correlation_queue_manager",
    "entry_id": "qmanager .223",
    "last_run": {
      "href": "/api/v1/tasks/isapp-cb7cc2e5-eab1-474a-907a-055f26dbc36d",
      "start_time": 1544474203
    },
    "params": {
      "configuration": "em7-host-223"
    },
    "schedule": {
      "schedule_info": {
        "run_every": 300
      },
```

```
            "schedule_type": "frequency"
        },
        "total_runs": 1589
    }
]
```

Test Cases

Load Throughput Test Cases

Event throughput testing with PowerFlow only:

The following test cases can be attempted with any number of dedicated customer queues. The expectation is that each customer queue will be filled with 10,000 events, and then you can time how long it takes to process through all 10,000 events in each queue.

1. Disable any steprunners.
2. Trigger 10,000 events through SL1, and let them build up in the PowerFlow queue.
3. After all 10,000 events are waiting in queue, enable the steprunners to begin processing.
4. Time the throughput of all event processing to get an estimate of how many events per second and per minute that PowerFlow will handle.
5. The results from the ScienceLogic test system are listed in the *sizing section of worker nodes*.

Event throughput testing with SL1 triggering PowerFlow:

This test is executed in the same manner as the event throughput test described above, but in this scenario you never disable the steprunners, and you let the events process through PowerFlow as they are alerted to by SL1.

1. Steprunners are running.
2. Trigger 10,000 events through SL1, and let the steprunners handle the events as they come in.
3. Time the throughput of all event processing to get an estimate of how many events per second and per minute that PowerFlow will handle.

The difference between the timing of this test and the previous test can show how much of a delay the SL1 is taking to alert PowerFlow about an event, and subsequently sync it.

Failure Test Cases

1. Validate that bringing one of the core nodes down does not impact the overall functionality of the PowerFlow system. Also, validate that bringing the core node back up rejoins the cluster and the system continues to be operational.

2. Validate that bringing down a dedicated worker node only affects that specific workers processing. Also validate that adding a new "standby" node is able to pick up the worker where the previous failed worker left off.
3. Validate that when the Redis service fails on any node, it is redistributed and is functional on another node.
4. Validate that when a PowerFlow application fails, you can view the failure in the PowerFlow Timeline.
5. Validate that you can query for and filter only for failing tasks for a specific customer.

Separated queue test scenarios

1. Validate that scheduling two runs of the same application with differing configurations and queues works as expected:
    - Each scheduled run should be placed on the designated queue and configuration for that schedule.
    - The runs, their queues, and configurations should be viewable from the PowerFlow Timeline, or can be queried from the log's endpoint.
2. Validate that each SL1 triggering event is correctly sending the appropriate queue and configuration that the event sync should be run on:
    - This data should be viewable from the PowerFlow Timeline.
    - The queue and configuration should be correctly recognized by PowerFlow and executed by the corresponding worker.
3. Validate the behavior of a node left "on standby" waiting for a label to start picking up work. As soon as a label is assigned and workers are scaled, that node should begin processing the designated work.

Backup Considerations

This section covers the items you should back up in your PowerFlow system, and how to restore backups. For more information, see *Backing up Data*.

What to Back Up

When taking backups of the PowerFlow environment, collect the following information from the host level of your primary manager node (this is the node from which you control the stack):

Files in /**opt/iservices/scripts**:

- **/opt/iservices/scripts/docker-compose.yml**
- **/opt/iservices/scripts/docker-compose-override.yml**

All files in **/etc/iservices/**:

- **/etc/iservices/is_pass**
- **/etc/iservices/encryption_key**

In addition to the above files, make sure you are storing Couchbase dumps somewhere by using the *cbbackup* command, or the "PowerFlow Backup" application.

Fall Back and Restore to a Disaster Recovery (Passive) System

You should do a data-only restore if:

- The system you are restoring to is a different configuration or cluster setup than the system where you made the backup.
- The backup system has all the indexes added and already up to date.

You should do a *full* restore if:

- The deployment where the backup was made is identical to the deployment where it is being restored (same amount of nodes).
- There are no indexes defined on the system you're backing up.

Once failed over, be sure to disable the "PowerFlow Backup" application from being scheduled.

Resiliency Considerations

The RabbitMQ Split-brain Handling Strategy (SL1 Default Set to Autoheal)

If multiple RabbitMQ cluster nodes are lost at once, the cluster might enter a "Network Partition" or "Split-brain" state. In this state, the queues will become paused if there is no auto-handling policy applied. The cluster will remain paused until a user takes manual action. To ensure that the cluster knows how to handle this scenario as the user would want, and not pause waiting for manual intervention, it is essential to set a partition handling policy.

For more information on RabbitMQ Network partition (split-brain) state, how it can occur, and what happens, see: http://www.rabbitmq.com/partitions.html.

By default, ScienceLogic sets the partition policy to *autoheal* in favor of continued service if any nodes go down. However, depending on the environment, you might wish to change this setting.

For more information about the automatic split-brain handling strategies that RabbitMQ provides, see: http://www.rabbitmq.com/partitions.html#automatic-handling.

autoheal is the default setting set by SL1, and as such, queues should always be available, though if multiple nodes fail, some messages may be lost.

NOTE: If you are using pause_minority mode and a "split-brain" scenario occurs for RabbitMQ in a single cluster, when the split-brain situation is resolved, new messages that are queued will be mirrored (replicated between all nodes once again).

ScienceLogic Policy Recommendation

ScienceLogic's recommendations for applying changes to the default policy include the following:

- If you care more about **continuity of service** in a data center outage, with queues always available, even if the system doesn't retain some messages from a failed data center, use *autoheal*. This is the SL1 default setting.
- If you care more about **retaining message data** in a data center outage, with queues that might not be available until the nodes are back, but will recover themselves once nodes are back online to ensure that no messages are lost, use *pause_minority*.
- If you prefer **not to have RabbitMQ handle this scenario automatically**, and you want to manually recover your queues and data, where queues will be paused and unusable until manual intervention is made to determine where to fallback, use *ignore*.

Changing the RabbitMQ Default Split-brain Handling Policy

The best way to change the SL1 default split-brain strategy is to make a copy of the RabbitMQ config file from a running rabbit system, add your change, and then mount that config back into the appropriate place to apply your overrides.

1. Copy the config file from a currently running container:

```
docker cp <container-id>:/etc/rabbitmq/rabbitmq.conf
/destination/on/host
```

2. Modify the config file:

```
change cluster_partition_handling value
```

3. Update your **docker-compose.yml** file to mount that file over the config for all rabbitmq nodes:

```
mount "<path/to/config>/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf"
```

Using Drained Managers to Maintain Swarm Health

To maintain Swarm health, ScienceLogic recommends that you deploy some swarm managers that do not take any of the workload of the application. The only purpose for these managers is to maintain the health of the swarm. Separating these workloads ensures that a spike in application activity will not affect the swarm clustering management services.

ScienceLogic recommends that these systems have 2 CPU and 4 GB of memory.

To deploy a drained manager node:

1. Add your new manager node into the swarm.
2. Drain it with the following command:

```
docker node update --availability drain <node>
```

Draining the node ensures that no containers will be deployed to it. For more information, see https://docs.docker.com/engine/swarm/admin_guide/.

Updating the PowerFlow Cluster with Little to No Downtime

There are two potential update workflows for updating the PowerFlow cluster. The first workflow involves using a Docker registry that is connectable to swarm nodes on the network. The second workflow requires manually copying the PowerFlow RPM or containers to each individual node.

Updating Offline (No Connection to a Docker Registry)

1. Copy the PowerFlow RPM over to all swarm nodes.
2. Only install the RPM on all nodes. Do not stack deploy. This RPM installation automatically extracts the latest PowerFlow containers, making them available to each node in the cluster.
3. From the primary manager node, make sure your **docker-compose** file has been updated, and is now using the appropriate version tag: either *latest* for the latest version on the system, or *2.x.x*.
4. If all swarm nodes have the RPM installed, the container images should be runnable and the stack should update itself. If the RPM was missed installing on any of the nodes, it may not have the required images, and as a result, services might not deploy to that node.

Updating Online (All Nodes Have a Connection to a Docker Registry)

1. Install the PowerFlow RPM only onto the master node.
2. Make sure the RPM doesn't contain any host-level changes, such as Docker daemon configuration updates. If there are host level updates, you might want to make that update on other nodes in the cluster
3. Populate your Docker registry with the latest PowerFlow images.
4. From the primary manager node, make sure your **docker-compose** file has been updated, and is now using the appropriate version tag: either *latest* for the latest version on the system, or *2.x.x*.
5. Docker stack deploy the services. Because all nodes have access to the same Docker registry, which has the designated images, all nodes will download the images automatically and update with the latest versions as defined by the **docker-compose** file.

Additional Sizing Considerations

This section covers the sizing considerations for the Couchbase, RabbitMQ, Redis, contentapi, and GUI services.

Sizing for Couchbase Services

The initial sizing provided for Couchbase nodes in the multi-tenant cluster for 6 CPUs and 56 GB memory should be more than enough to handle multiple customer event syncing workloads.

ScienceLogic recommends monitoring the CPU percentage and Memory Utilization percentage of the Couchbase nodes to understand when a good time to increase resources is, such as when Memory and CPU are consistently above 80%.

Sizing for RabbitMQ Services

The only special considerations for RabbitMQ sizing is how many events you will plan for in the queue at once.

Every 10,000 events populated in the PowerFlow queue will consume approximately 1.5 GB of memory.

NOTE: This memory usage is drained as soon as the events leave the queue.

Sizing for Redis Services

The initial sizing deployment for redis should be sufficient for multiple customer event syncing.

The only time memory might need to be increased to Redis is if you are attempting to view logs from a previous run, and the logs are not available. A lack of run logs from a recently run integration indicates that the Redis cache does not have enough room to store all the step and log data from recently executed runs.

Sizing for contentapi Services

The contentapi services sizing should remain limited at 2 GB memory, as is set by default.

If you notice timeouts, or 500s when there is a large load going through the PowerFlow system, you may want to increase the number of contentapi replicas.

For more information, see *placement considerations*, and ensure the API replicas are deployed in the same location as the redis instance.

Sizing for the GUI Service

The GUI service should not need to be scaled up at all, as it merely acts as an ingress proxy to the rest of the PowerFlow services.

Sizing for Workers: Scheduler, Steprunner, Flower

Refer to the worker sizing charts provided by ScienceLogic for the recommended steprunner sizes.

Flower and Scheduler do not need to be scaled up at all.

Scaling the PowerFlow Devpi Server

For large environments, you can replicate the PowerFlow Devpi Server, which is the internal Python package repository. Creating Devpi Server replicas prevents multiple syncpacks_steprunners from attempting to access a single Devpi Server at the same time, which might cause failures when creating or recreating SyncPack virtual environments.

NOTE: The Devpi Server is deployed as the **pypiserver** service on a PowerFlow stack.

When to Add a New Devpi Server Replica to the PowerFlow Stack

ScienceLogic recommends that you add replicas if you have more than 75 syncpack_steprunners, or add retries to the SyncPack installation process.

Number of syncpack_steprunners	Devpi Server Replicas
75 or more	0
100 or more	1
150 or more	2

Adding a New Devpi Server Replica to the Stack

If you want to add a Devpi Server replica to the PowerFlow stack, you will need to add a new service to the **docker-compose-override** file, using the same configuration as the code block, below.

NOTE: You can add any number of replicas to the stack, but each replica must have its own unique alias and volume, as the Devpi Servermaster volume cannot be used by a replica.

Code Example: docker-compose-override file

```
pypiserver_replica:
    container_name: devpi_replica
      deploy:
        replicas: 1
        placement:
          constraints:
          - node.hostname == pf-node2 # name of the node where this rep-
lica is running
    environment:
      devpi_role: 'replica'
      devpi_threads: 200
    hostname: devpi_replica
    image: scr.sl1.io/pf-pypi:6.3.1-7
    networks:
      isnet:
        aliases:
        - pypiserver_replica
        - pypiserver_replica.isnet
    secrets:
    - source: encryption_key
```

```
    - source: is_pass
    volumes:
    - devpi_replica:/data:rw
.. . ..
volumes:
  devpi_replica: {}
  ... . ..
```

Considerations

- To initialize Devpi Server replicas, the master Devpi Server service should be running and healthy. Replicas have the same information as the master, because the replicas are constantly syncing with their master.
- To allow the Devpi Server and its replicas to receive more than 200 concurrent requests, you can increase the number of threads by setting the **devpi_threads** environment variable in the Devpi Server and its replicas.
- When a Devpi Server replica is running, the replica makes a request to the Devpi Server service every 30 seconds to sync SyncPacks and their dependencies, which means that the Devpi Server can be busier than its replicas receiving requests from the steprunners.

Configuring Steprunners to Consume Data from Devpi Server Replicas

To allow **steprunner** and **syncpacks_steprunner** services to use Devpi Server replicas:

1. Set the **devpi_trusted_host** environment variable for **syncpacks_steprunner** and **steprunner** services with a string that contains the aliases of the Devpi Server and its replicas separated by a comma.

   Other custom configurations related to the **devpi_trusted_host** include the following:

   - **devpi_trusted_host**. The default value is **pypiserver.isnet**.
   - **devpi_random_order**. The default value is **false**. This configuration let you mix the order of the **devpi_trusted_host** list.
   - **devpi_random_host_number**. The default value is the **devpi_trusted_host** length. This configuration defines how many hosts will be chosen randomly from the **devpi_trusted_host** list.

   The following example uses two Devpi Server replicas:

```
syncpacks_steprunner:
  environment:
    devpi_trusted_host: pypiserver.isnet,pypiserver_
replica.isnet,pypiserver_
replica2.isnet
    devpi_random_order: true
    devpi_random_host_number: 2
```

2. For setting a Devpi Server replica as a main resource for a syncpack_steprunner, define the following environment variable:

```
syncpacks_steprunner:
  environment:
    devpi_host: pypiserver_replica.isnet
```

> **NOTE:** You only need to do this step if you want to completely restrict a steprunner from calling the master Devpi Server service.

3. To assign custom Devpi Server replicas to steprunners in different nodes, you can use a **pip.conf** file. The following example shows how to mount the custom **pip.conf** file as a volume.

```
syncpacks_steprunner:
  environment:
    PIP_CONFIG_FILE: /usr/tmp/pip.conf
.. ....
volumes:
  - /tmp/pip.conf:/usr/tmp/pip.conf
```

Because volumes are owned by every node, this file can contain a different configuration based on the node where the syncpack_steprunners are running. This is not recommended, as managing different versions of **pip.conf** in different nodes can be difficult.

Additional Considerations

In environments where more than 75 syncpack_steprunners are running, ScienceLogic recommend the following configurations:

- The number of Devpi Server threads **devpi_threads** should be increased from the default value of 200. Start with 500 and increase it to 1000 if needed:

```
pypiserver_replica:
  container_name: devpi_replica
    deploy:
      replicas: 1
      placement:
        constraints:
        - node.hostname == pf-node2 # name of the node where this
replica is running
  environment:
    devpi_role: 'replica'
    devpi_threads: 1000
```

> **NOTE:** When the number of Devpi Server threads is increased, that service's memory consumption is also slightly increased.

- When PowerFlow is running offline, more calls can occur to Devpi Server and its replicas, so take that situation into account that when setting replicas and its threads.
- Retries for pip should be set by setting by the environment variable **PIP_RETRIES** to 3. This configuration should be set on the syncpack_steprunners.
- Retries for the SyncPack installation application is configured as an environment variable for the **using sp_installation_retries**, which has default value of 3.

```
syncpacks_steprunner:
   environment:
     devpi_trusted_host: pypiserver.isnet,pypiserver_
replica.isnet,pypiserver_replica2.isnet
     PIP_RETRIES: 3 # default value is 0
     sp_installation_retries: 5
     PIP_TIMEOUT: 10 # default value is 5
```

Node Placement Considerations

Preventing a Known Issue: Place contentapi and Redis services in the Same Physical Location

An issue exists where if there latency exists between the contentapi and redis, the Applications page may not load. This issue is caused by the API making too many calls before returning. The added latency for each individual call can cause the overall endpoint to take longer to load than the designated timeout window of thirty seconds.

The only impact of this issue is the Applications page won't load. There is no operational impact on the integrations as a whole, even if workers are in separate geos than redis.

There is also no risk to High Availability (HA) by placing the API andRredis services on the same geo. If for whatever reason that geo drops out, the containers will be restarted automatically in the other location.

Common Problems, Symptoms, and Solutions

Tool	Issue	Symptoms	Cause	Solution
Docker Visualizer	Docker Visualizer shows some services as "undefined".	When viewing the Docker Visualizer user interface, some services are displayed as "undefined", and states aren't accurate. **Impact**: Cannot use Visualizer to get the current state of the stack.	Failing docker stack deployment: https://github.com/dockersamples/docker-swarm-visualizer/issues/110	Ensure your stack is healthy, and services are deployed correctly. If no services are failing and things are still showing as undefined, *elect a new swarm leader*. **To prevent:** Ensure your configuration is valid before deploying.
RabbitMQ	RabbitMQ queues encountered a node failure and are in a "Network partition state" (split-brain scenario).	The workers are able to connect to the queue, and there are messages on the queue, but the messages are not being distributed to the workers. Log in to the RabbitMQ admin user interface, which displays a message similar to "RabbitMQ experienced a network partition and the cluster is paused". **Impact**: The RabbitMQ cluster is paused and waiting for user intervention to clean the split-brain state.	Multi-node failure occurred, and rabbit wasn't able to determine who the new master should be. This also will only occur if there is NO partition handling policy in place (see the resiliency section for more information) **Note**: ScienceLogic sets the *autoheal* policy by default	Handle the split-brain partition state and *resynchronize your RabbitMQ queues*. **Note**: This is enabled by default. **To prevent:** Set a partition handling policy. See the

Tool	Issue	Symptoms	Cause	Solution
				Resiliency section for more information.

RabbitMQ, ***continued***		Execing into the RabbitMQ container and running **rabbitmqcli cluster-status** shows nodes in a partition state like the following:
		`[{nodes,`
		`[{disc,`
		`['rabbit@rabbit_node1.isnet','rabbit@rabbit_node2.isnet','rabbit@rabbit_node3.isnet','rabbit@rabbit_node4.isnet','rabbit@rabbit_node5.isnet','rabbit@rabbit_node6.isnet']}]},`
		`{running_nodes,['rabbit@rabbit_node4.isnet']},`
		`{cluster_name,<<"rabbit@rabbit_node1">>},`
		`{partitions,`
		`[{'rabbit@rabbit_node4.isnet',['rabbit@rabbit_node1.isnet','rabbit@rabbit_node2.isnet','rabbit@rabbit_node3.isnet','rabbit@rabbit_node5.isnet','rabbit@rabbit_node6.isnet']}]},`		

Tool	Issue	Symptoms	Cause	Solution
		`{alarms,[{'rabbit@rabbit_node4.isnet',[]}]}]`		
PowerFlow steprunners and RabbitMQ	Workers constantly restarting, no real error message.	Workers of a particular queue are not stable and constantly restart. **Impact**: One queue's workers will not be processing.	Multi-node failure in RabbitMQ, when it loses majority and can not failover. Queues go out of sync because of broken swarm.	*Recreate queues* for the particular worker. *Resynchronize queues*. **To prevent**: Deploy enough nodes to ensure quorum for failover.

Couchbase	Couchbase node is unable to restart due to indexer error.	This issue can be monitored in the Couchbase logs:
		`Service 'indexer' exited with status 134. Restarting. Messages:`
		`sync.runtime_Semacquire (0xc4236dd33c)`
		Impact:
		One couchbase node becomes corrupt.
			Memory is removed from the database while it is in operation (memory must be dedicated to the VM running Couchbase).
			The Couchbase node encounters a failure, which causes the corruption.
				Ensure that the memory allocated to your database nodes is dedicated and not shared among other VMs.
				To prevent:
				Ensure that the memory allocated to your database nodes is dedicated and not shared among other VMs.

Couchbase	Couchbase is unable to rebalance.	Couchbase nodes will not rebalance, usually with an error saying "exited by janitor". **Impact**: Couchbase nodes cannot rebalance and provide even replication.	Network issues: missing firewall rules or blocked ports. The Docker swarm network is stale because of a stack failure.	Validate that all firewall rules are in place, and that no external firewalls are blocking ports. Reset the

Tool	Issue	Symptoms	Cause	Solution
				Docker swarm network status by *electing a new swarm leader.* **To prevent:** Validate the firewall rules before deployment. *Use drained managers to maintain swarm*
PowerFlow steprunners to Couchbase	Steprunners unable to communicate to Couchbase	Steprunners unable to communicate to Couchbase database, with errors like "client side timeout", or "connection reset by peer". **Impact**: Steprunners cannot access the database.	Missing Environment variables in compose: Check the db_host setting for the steprunner and make sure they specify all Couchbase hosts available . Validate couchbase settings, ensure that the proper aliases, hostname, and environment variables are set. Stale docker network.	Validate the deployment configuration and network settings of your **docker-compose**. Redeploy with valid settings. In the event of a swarm failure, or stale swarm network, reset the Docker swarm network status by *electing a new swarm leader.* **To prevent:** Validate hostnames, aliases, and environment settings before deployment. *Use drained managers to maintain*

Tool	Issue	Symptoms	Cause	Solution
				swarm
Flower	Worker display in flower is not organized and hard to read, and it shows many old workers in an offline state.	Flower shows all containers that previously existed, even if they failed, cluttering the dashboard. **Impact**: Flower dashboard is not organized and hard to read.	Flower running for a long time while workers are restarted or coming up/coming down, maintaining the history of all the old workers. Another possibility is a known issue in task processing due to the `--max-tasks-per-child` setting. At high CPU workloads, the `max-tasks-per-child` setting causes workers to exit prematurely.	Restart the flower service by running the following command: `docker service update --force iservices_flower` You can also remove the `--max-tasks-per-child` setting in the steprunners.
All containers on a particular node	All containers on a particular node do not deploy.	Services are not deploying to a particular node, but instead they are getting moved to other nodes. **Impact**: The node is not running anything.	One of the following situations could cause this issue: Invalid label deployment configuration. The node does not have the containers you are telling it to deploy. The node is missing a required directory to mount into the container.	Make sure the node that you are deploying to is labeled correctly, and that the services you expect to be deployed there are properly constrained to that system. Go through the troubleshooting steps of "When a docker service doesn't deploy" to check that the service is not missing a requirement on the host. Check the node status for errors:

Tool	Issue	Symptoms	Cause	Solution
				`docker node ls` **To prevent:** Validate your configuration before deploying.
All containers on a particular node	All containers on a particular node periodically restart at the same time.	All containers on a particular node restart at the same time. The system logs indicate an error like: `"error="rpc error: code = DeadlineExceeded desc = context deadline exceeded"` **Impact**: All containers restart on a node.	This issue only occurs in single-node deployments when the only manager allocates too many resources to its containers, and the containers all restart since the swarm drops. The manager node gets overloaded by container workloads and is not able to handle swarm management, and the swarm loses quorum.	Use some drained manager nodes for swarm management to separate the workloads. **To prevent**: *Use drained managers to maintain swarm*.
General Docker service	Docker service does not deploy. Replicas remain at 0/3.	Docker service does not deploy.	There are a variety of reasons for this issue, and you can reveal most causes by checking the service logs to address the issue.	*Identify the cause of the service not deploying*.
PowerFlow user interface	The Timeline or the **Applications** page do not appear in the user interface.	The Timeline is not showing accurate information, or the **Applications** page is not rendering.	One of the following situations could cause these issues: Indexes do not exist on a particular Couchbase node. Latency between the API and the redis service is too great for the API to collect all the data it needs before the 30-second timeout is reached.	Solutions: Verify that indexes exist. Place the API and redis containers in the same geography so there is little latency. This issue will be fixed in a future IS release

Tool	Issue	Symptoms	Cause	Solution
			The indexer can't keep up to a large number of requests, and Couchbase requires additional resources to service the requests.	Increase the amount of memory allocated to the Couchbase indexer service.

Common Resolution Explanations

This section contains a set of solutions and explanations for a variety of issues.

Elect a New Swarm Leader

Sometimes when managers lose connection to each other, either through latency or a workload spike, there are instances when the swarm needs to be reset or refreshed. By electing a new leader, you can effectively force the swarm to redo service discovery and refresh the metadata for the swarm. This procedure is highly preferred over removing and re-deploying the whole stack.

To elect a new swarm leader:

1. Make sure there at least three swarm managers in your stack.
2. To identify which node is the current leader, run the following command:

```
docker node ls
```

3. Demote the current leader with the following command:

```
docker node demote <node>
```

4. Wait until a new node is elected leader:

```
docker node ls
```

5. After a new node is elected leader, promote the old node back to swarm leader:

```
docker node promote <node>
```

Recreate RabbitMQ Queues and Exchanges

> **NOTE:** If you do not want to retain any messages in the queue, the following procedure is the best method for recreating the queues. If you do have data that you want to retain, you can *resynchronize RabbitMQ queues*.

To recreate RabbitMQ queues:

1. Identify the queue or queues you need to delete:
    - If default workers are restarting, you need to delete queues celery and priority.high.
    - If a custom worker cannot connect to the queue, simply delete that worker's queue.
2. Delete the queue and exchange through the RabbitMQ admin console:
    - Log in to the RabbitMQ admin console and go to the **[Queues]** tab.
    - Find the queue you want to delete and click it for more details.
    - Scroll down and click the **[Delete Queue]** button.
    - Go to the **[Exchanges]** tab and delete the exchange with the same name as the queue you just deleted.
3. Delete the queue and exchange through the command line interface:
    - exec into a rabbitmq container
    - Delete the queue needed:

        ```
        rabbitmqadmin delete queue name=name_of_queue
        ```

    - Delete the exchange needed:

        ```
        rabbitmqadmin delete exchange name=name_of_queue
        ```

    After you delete the queues, the queues will be recreated the next time a worker connects.

Resynchronize RabbitMQ Queues

If your RabbitMQ cluster ends up in a "split-brain" or partitioned state, you might need to manually decide which node should become the master. For more information, see http://www.rabbitmq.com/partitions.html#recovering.

To resynchronize RabbitMQ queues:

1. Identify which node you want to be the master. In most cases, the master is the node with the most messages in its queue.
2. After you have identified which node should be master, scale down all other RabbitMQ services:

    ```
    docker service scale iservices_rabbitmqx=x0
    ```

3. After all other RabbitMQ services except the master have been scaled down, wait a few seconds, and then scale the other RabbitMQ services back to 1. Bringing all nodes but your new master down and back up again forces all nodes to sync to the state of the master that you chose.

Identify the Cause of a Service not Deploying

Step 1: Obtain the ID of the failed container for the service

Run the following command for the service that failed previously:

```
docker service ps --no-trunc <servicename>
```

For example:

```
docker service ps --no-trunc iservices_redis
```

```
root@is-scale-03 ~]# docker service ps iservices_redis
D                   NAME                    IMAGE             NODE              DESIRED STATE       CURRENT STATE            ERROR
ORTS
1slu2qwqbte        iservices_redis.1       redis:4.0.2       is-scale-04       Running             Running 2 hours ago
s7s86n45skf         \_ iservices_redis.1   redis:4.0.2       is-scale-03       Shutdown            Failed 2 hours ago       "task: non-zero exit (137)"
```

From the command result above, we see that one container with the ID **3s7s86n45skf** failed previously running on node **is-scale-03** (non-zero exit) and another container was restarted in its place.

At this point, you can ask the following questions:

- Is the error when using `docker service ps --no-trunc` something obvious? Does the error say that it cannot mount a volume, or that the image was not found? If so, that is most likely the root cause of the issue and needs to be addressed.
- Did the node on which that container was running go down? Or is that node still up?
- Are the other services running on that node running fine, and was only this service affected? If other services are running fine on that same node, it is probably a problem with the service itself. If all services on that node are not functional, it could mean a node failure.

At this point, the cause of the issue is not a deploy configuration issue, and it is not an entire node failure. The problem exists within the service itself. Continue to Step 2 if this is the case.

Step 2: Check for any interesting error messages or logs indicating an error

Using the ID obtained in Step 1, collect the logs from the failed container with the following command:

```
docker service logs <failed-id>
```

For example:

```
docker service logs 3s7s86n45skf
```

Review the service logs for any explicit errors or warning messages that might indicate why the failure occurred.

Repair Couchbase Indexes

Index stuck in “created” (not ready) state

This situation usually occurs when a node starts creating an index, but another index creation was performed at the same time by another node. After the index is created, you can run a simple query to build the index which will change it from created to “ready”:

```
BUILD index on 'content'('idx_content_content_type_config_a3f867db_7430_
4c4b_b1b6_138f06109edb') using GSI
```

Deleting an index

If you encounter duplicate indexes, such as a situation where indexes were manually created more than once, you can delete an index:

```
DROP index content.idx_content_content_type_config_d8a45ead_4bbb_4952_
b0b0_2fe227702260
```

Recreating all indexes on a particular node

To recreate all indexes on a particular Couchbase node, exec into the couchbase container and run the following command:

```
Initialize_couchbase -s
```

NOTE: Running this command recreates all indexes, even if the indexes already exist.

Add a Broken Couchbase Node Back into the Cluster

To remove a Couchbase node and re-add it to the cluster:

1. Stop the node in Docker.
2. In the Couchbase user interface, you should see the node go down, failover manually, or wait the appropriate time until it automatically fails over.
3. Clean the Couchbase data directory on the necessary host by running the following command:

   ```
   rm -rf /var/data/couchbase/*
   ```

4. Restart the Couchbase node and watch it get added back into the cluster.
5. Click the **Rebalance** button to replicate data evenly across nodes.

Restore Couchbase Manually

NOTE: If you created the backup with the "PowerFlow Backup" application in PowerFlow, you will need to decompress the backup file. The Couchbase backup is in the **couchbase** folder, and you will need to use the backup in that folder to restore the backup.

Backup

1. Run the following command on each manager node:

   ```
   docker ps
   ```

2. Find the container with the Couchbase name and make a note of that container's ID.
3. Run the following command on that manager node, inserting the container ID for that node:

   ```
   docker exec -it <container_id> /bin/bash
   ```

4. Execute into the Couchbase container by running the following command:

   ```
   cbbackup http://couchbase.isnet:8091 /opt/couchbase/var/backup -u
   <user> -p <password> -x data_only=1
   ```

5. Exit the Couchbase shell and then copy the backup file in **/var/data/couchbase/backup** to a safe location, such as **/home/isadmin**.
6. Repeat these steps on each PowerFlow node.

Delete Couchbase

```
rm -f /var/data/couchbase/*
```

Restore

1. Copy the backup file into **/var/data/couchbase/backup**.
2. Execute into the Couchbase container.
3. Run the following command to restore the content:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b
content -u <user> -p <password>
```

4. Run the following command to restore the logs:

```
cbrestore /opt/couchbase/var/backup http://couchbase.isnet:8091 -b
logs -u <user> -p <password>
```

PowerFlow Multi-tenant Upgrade Process

This section describes how to upgrade PowerFlow in a multi-tenant environment with as little downtime as possible.

Performing Environment Checks Before Upgrading

Validate Cluster states

- Validate that all Couchbase nodes in the cluster are replicated and fully balanced.
- Validate that the RabbitMQ nodes are all clustered and queues have *ha-v1-all* policy applied.
- Validate that the RabbitMQ nodes do not have a large number of messages backed up in queue.

Validate Backups exist

- Ensure that you have a backup of the database before upgrading.
- Ensure that you have a copy of your most recently deployed docker-compose file. If all user-specific changes are only populated in docker-compose-override, this is not necessary, but you might want a backup copy.
- Make sure that each node in Couchbase is fully replicated, and no re-balancing is necessary.

Clean out old container images if desired

Before upgrading to the latest version of PowerFlow, check the local file system and see if there are any older versions taking up space that you might want to remove. These containers exist both locally on the file system and the internal Docker registry. To view any old container versions, check the **/opt/iservices/images** directory.

ScienceLogic recommends that you keep at a minimum the last version of containers, so you can downgrade if necessary.

Cleaning out images is not mandatory, but it is just a means of clearing out additional space on the system if necessary.

To remove old images:

1. Delete any unwanted versions in **/opt/iservices/images**.
2. Identify any unwanted images known to Docker with docker images.
3. Remove the images with the ID `docker rmi <id>`.

Installing the PowerFlow RPM

The first step of upgrading is to install the new RPM on all nodes in the cluster. Doing so will ensure that the new containers are populated onto the system (if using that particular RPM), and any other host settings are changed. RPM installation does not pause any services or affect the Docker system in any way, other than using some resources.

PowerFlow has two RPMs, one with containers and one without. If you have populated an internal Docker registry with Docker containers, you can install the RPM without containers built in. If no internal Docker repository is present, you must install the RPM which has the containers built in it. Other than the containers, there is no difference between the RPMs.

For advanced users, installing the RPM can be skipped. However this means that the user is completely responsible for maintaining the docker-compose and host level configurations.

To install the RPM:

1. SSH into each node.
2. If you are installing the RPM that contains the container images built in, you may want to upgrade each core node one by one, so that the load of extracting the images doesn't affect all core nodes at once
3. Run the following command:

   ```
   sudo rpm -Uvh <full_path_of_rpm>
   ```

   where `full_path_of_rpm` is the name and path of the RPM file, such as **/home/isadmin/sl1-powerflow-2.x.x-1.x86_64**.

Compare docker-compose file changes and resolve differences

After the RPM is installed, you will notice a new **docker-compose.yml** file is placed in **/opt/iservices/scripts/**. As long as your environment-specific changes exist solely in the **compose-override** file, all user changes and new version updates will be resolved into that new docker-compose.yml file.

ScienceLogic recommends that you check the differences between the two docker-compose files. You should validate that:

1. All environment-specific and custom user settings that existed in the old docker-compose also exist in the new docker-compose file.

2. The image tags reference the correct version in the new docker-compose. If you are using an internal Docker registry, be sure these image tags represent the images from your internal registry.
3. Make sure that any new environment variables added to services are applied to replicated services. To ensure these updates persist through the next upgrade, also make the changes in docker-compose-override.yml. In other words, if you added a new environment variable for Couchbase, make sure to apply that variable to couchbase-worker1 and couchbase-worker2 as well. If you added a new environment variable for the default steprunner, make sure to set the same environment variable on each custom worker as well.
4. If you are using the *latest* tag for images, and you are using a remote repository for downloading, be sure that the *latest* tag refers to the images in your repository.
5. The old docker-compose is completely unchanged, and it matches the current deployed environment. This enables PowerFlow to update services independently without restarting other services.
6. After you resolve any differences between the compose files has been resolved, proceed with the upgrade using the old **docker-compose.yml** (the one that matches the currently deployed environment).

Make containers available to systems

After you apply the host-level updates, you should make sure that the containers are available to the system.

If you upgraded using the RPM with container images included, the containers should already be on all of the nodes, you can run Docker images to validate the new containers are present. If this is the case you may skip to the next section.

If the upgrade was performed using the RPM which did *not* contain the container images, ScienceLogic recommends that you run the following command to make sure all nodes have the latest images:

```
docker-compose -f <new_Docker_compose_file> pull
```

This command validates that the containers specified by your compose file can be pulled and reached from the nodes. While not required, you might to make sure that the images can be pulled before starting the upgrade. If the images are not pulled manually, they will automatically be pulled by Docker when the new image is called for by the stack.

Perform the Upgrade

To perform the upgrade on a clustered system with little downtime, PowerFlow re-deploys services to the stack in groups. To do this, PowerFlow gradually makes the updates to groups of services and re-runs *docker stack deploy* for each change. To ensure that no unintended services are updated, start off using the same docker-compose file that was previously used to deploy. Reusing the same docker-compose file and updating only sections at a time ensures that only the intended services to be updated are affected at any given time.

Avoid putting all the changes in a single docker-compose file, and do a new *docker stack deploy* with all changes at once. If downtime is not a concern, you can update all services, but updating services gradually allows you to have little or no downtime.

WARNING: Before upgrading any group of services, be sure that the **docker-compose** file you are deploying from is *exactly* identical to the currently deployed stack (the previous version). Start with the *same* **docker-compose** file and update it for each group of services as needed,

Upgrade Redis, Scheduler, and Flower

The first group to update includes Redis, Scheduler and Flower. If desired, this group can be upgraded along with any other group.

To update:

1. Copy the service entries for Redis, Scheduler and Flower from the new compose file into the old **docker-compose** file (the file that matches the currently deployed environment). Copying these entries makes it so that the only changes in the docker-compose file (compared to the deployed stack) are changes for Redis, Scheduler and Flower.
2. Run the following command:

```
docker stack deploy -c /opt/iservices/scripts/docker-compose.yml
iservices
```

3. Monitor the update, and wait until all services are up and running before proceeding.

Code Example: Image definition of this upgrade group

```
services:

  contentapi:

    image: repository.auto.sciencelogic.local:5000/is-api:2.4.0

  couchbase:

    image: repository.auto.sciencelogic.local:5000/is-couchbase:2.4.0

  couchbase-worker:

    image: repository.auto.sciencelogic.local:5000/is-couchbase:2.4.0

  flower:

    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.0

  gui:

    image: repository.auto.sciencelogic.local:5000/is-gui:2.4.0

  rabbitmq:
```

```
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2
scheduler:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1
steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:2.4.0
couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:2.4.0
steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:2.4.0
```

Redis Version

As the Redis version might not change with every release of PowerFlow, there might not be any changes needed in the upgrade for Redis. This can be expected and is not an issue.

You can configure Redis to let the **contentapi** container iterate through multiple potential Redis result stores to find the correct result id for a task. To enable this option in the **docker-compose.yml** file, set the **result_backend** environment variable of the **contentapi** container to a comma-delimited list of URLs for Redis instances, such as **redis://redis:6378/0,redis:///redis2:6380/0**. To deploy multiple Redis instances, make sure that the stack deploys the instances with different aliases, ports, and hostnames. Also, multiple backends are only supported on **contentapi**, not the **steprunners**. Steprunners can only write to a single backend.

Upgrade Core Services (RabbitMQ and Couchbase)

The next group of services to update together are the RabbitMQ/Couchbase database services, as well as the GUI. Because the core services are individually defined and "pinned" to specific nodes, upgrade these two

services at the same time, on a node-by-node basis. In between each node upgrade, wait and validate that the node rejoins the Couchbase and Rabbit clusters and re-balances appropriately.

Because there will always be two out of three nodes running these core services, this group should not cause any downtime for the system.

Rabbit/Couchbase Versions

The Couchbase and RabbitMQ versions used might not change with every release of PowerFlow. If there is no update or change to be made to the services, you can ignore this section for RabbitMQ or Couchbase upgrades, or both. Assess the differences between the old and new docker-compose files to check if there is an image or environment change necessary for the new version. If not, you can move on to the next section.

Update Actions (assuming three core nodes)

To update first node services:

1. Update just core *node01* by copying service entries for couchbase, rabbitmq1 from the new compose file (compared and resolved as part of above prepare steps) into the old docker-compose file. At this point, the compose file you use to deploy should also contain the updates for the previous groups
2. Before deploying, access the Couchbase user interface, select the first server node, and click "failover". Select "graceful failover". Manually failing over before updating ensures that the system is still operational when the container comes down.
3. For the failover command that can be run through the command-line interface if the user interface is not available, see the ***Manual Failover section***.
4. Run the following command:

   ```
   docker stack deploy -c <compose_file>
   ```

5. Monitor the process to make sure the service updates and restarts with the new version. To make sure that as little time as possible is used when updating the database, the database containers should already be available on the core nodes.
6. After the node is back up, go back to the Couchbase UI and add the node back, and rebalance the cluster to make it whole again.
7. For more information on how to re-add the node and rebalance the cluster if the user interface is not available, see the ***Manual Failover section***.

First node Couchbase update considerations

- When updating the first couchbase node, be sure to set the environment variable **JOIN_ON: "couchbase-worker2"**, so that the couchbase master knows to rejoin the workers after restarting.
- Keep in mind by default, only the primary Couchbase node user interface is exposed. Because of this, when the first Couchbase node is restarted, the Couchbase admin user interface will be inaccessible. If you would like to have the Couchbase user interface available during the upgrade of this node, ensure that at least one other Couchbase-worker services port is exposed.

Code Example: docker-compose with images and JOIN_ON for updating the first node

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:2.4.0
  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-
    2.4.1
    environment:
      JOIN_ON: "couchbase-worker2"
  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:2.4.0
  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1
  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-2.4.1
  rabbitmq:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
  rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
  rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
  redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2
  scheduler:
```

```
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1

steprunner:

    image: repository.auto.sciencelogic.local:5000/is-worker:2.4.0

couchbase-worker2:

    image: repository.auto.sciencelogic.local:5000/is-couchbase:2.4.0

steprunner2:

    image: repository.auto.sciencelogic.local:5000/is-worker:2.4.0
```

Update second and third node services

To update the second and third node services, repeat the steps from the first node on each node until all nodes are re-clustered and available. Be sure to check the service port mappings to ensure that there are no conflicts (as described above), and remove any HTTP ports if you choose.

Update the GUI

Because the GUI service provides all ingress proxy routing to the services, there might be a very small window where PowerFlow might not receive API requests as the GUI (proxy) is not running. This downtime is limited to the time it takes for the GUI container to restart.

To update the user interface:

1. Make sure that any conflicting port mappings are handled and addressed.
2. Replace the docker-compose GUI service definition with the new one.
3. Re-deploy the docker-compose file, and validate that the new GUI container is up and running.
4. Make sure that the HTTPS ports are accessible for Couchbase/RabbitMG.

Update Workers and contentapi

You should update the workers and contentapi last. Because these services use multiple replicas (multiple steprunner or containerapi containers running per service), you can rely on Docker to incrementally update each replica of the service individually. By default, when a service is updated, it will update one container of the service at a time, and only after the previous container is up and stable will the next container be deployed.

You can utilize additional Docker options in docker-compose to set the behavior of how many containers to update at once, when to bring down the old container, and what happens if a container upgrade fails. See the *update_config* and *rollback_config* options available in Docker documentation: https://docs.docker.com/compose/compose-file/.

Upgrade testing was performed by ScienceLogic using default options. An example where these settings are helpful is to change the parallelism of *update_config* so that all worker containers of a service update at the same time.

The update scenario described below takes extra precautions and only updates one node of workers per customer at a time. If you decide, you can also safely update all workers at once.

To update the workers and contentapi:

1. Modify the docker-compose file, the contentapi, and "worker_node1" services of all customers to use the new service definition.
2. Run a `docker stack deploy` of the new compose file. Monitor the update, which should update the API container one instance at a time, always leaving a container available to service requests. The process updates the workers of node1 one container instance at a time by default.
3. After workers are back up and the API is fully updated, modify the docker-compose file and update the second node's worker's service definitions.
4. Monitor the upgrade, and validate as needed.

Code Example: docker-compose definition with one of two worker nodes and contentapi updated:

```
services:

  contentapi:

    image: repository.auto.sciencelogic.local:5000/is-api:hotfix-2.4.1

    deploy:

        replicas: 3

  couchbase:

    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-
    2.4.1

    environment:

      JOIN_ON: "couchbase-worker2"

  couchbase-worker:

    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-
    2.4.1

  flower:

    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1

  gui:
```

```
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-2.4.1
rabbitmq:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
rabbitmq2:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
rabbitmq3:
    image: repository.auto.sciencelogic.local:5000/is-rabbit:3.7.7-2
redis:
    image: repository.auto.sciencelogic.local:5000/is-redis:4.0.11-2
scheduler:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1
steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-2.4.1
couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-
    2.4.1
steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:2.4.0
```

© 2003 - 2025, ScienceLogic, Inc.

All rights reserved.

LIMITATION OF LIABILITY AND GENERAL DISCLAIMER

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information on this Site, information on this Site may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described in this Site at any time without notice.

Copyrights and Trademarks

ScienceLogic, the ScienceLogic logo, and EM7 are trademarks of ScienceLogic, Inc. in the United States, other countries, or both.

Below is a list of trademarks and service marks that should be credited to ScienceLogic, Inc. The ® and ™ symbols reflect the trademark registration status in the U.S. Patent and Trademark Office and may not be appropriate for materials to be distributed outside the United States.

- ScienceLogic™
- EM7™ and em7™
- Simplify IT™
- Dynamic Application™
- Relational Infrastructure Management™

The absence of a product or service name, slogan or logo from this list does not constitute a waiver of ScienceLogic's trademark or other intellectual property rights concerning that name, slogan, or logo.

Please note that laws concerning use of trademarks or product names vary by country. Always consult a local attorney for additional guidance.

Other

If any provision of this agreement shall be unlawful, void, or for any reason unenforceable, then that provision shall be deemed severable from this agreement and shall not affect the validity and enforceability of any remaining provisions. This is the entire agreement between the parties relating to the matters contained herein.

In the U.S. and other jurisdictions, trademark owners have a duty to police the use of their marks. Therefore, if you become aware of any improper use of ScienceLogic Trademarks, including infringement or counterfeiting by third parties, report them to Science Logic's legal department immediately. Report as much detail as possible about the misuse, including the name of the party, contact information, and copies or photographs of the potential misuse to: legal@sciencelogic.com. For more information, see https://sciencelogic.com/company/legal.

800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010