# SL1 PowerFlow Platform

Version 2.2.1

# Table of Contents

# 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, and Cherwell Service Management. The PowerFlow platform sits in the middle of SL1 and the third-party application and handles the flow of data.

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

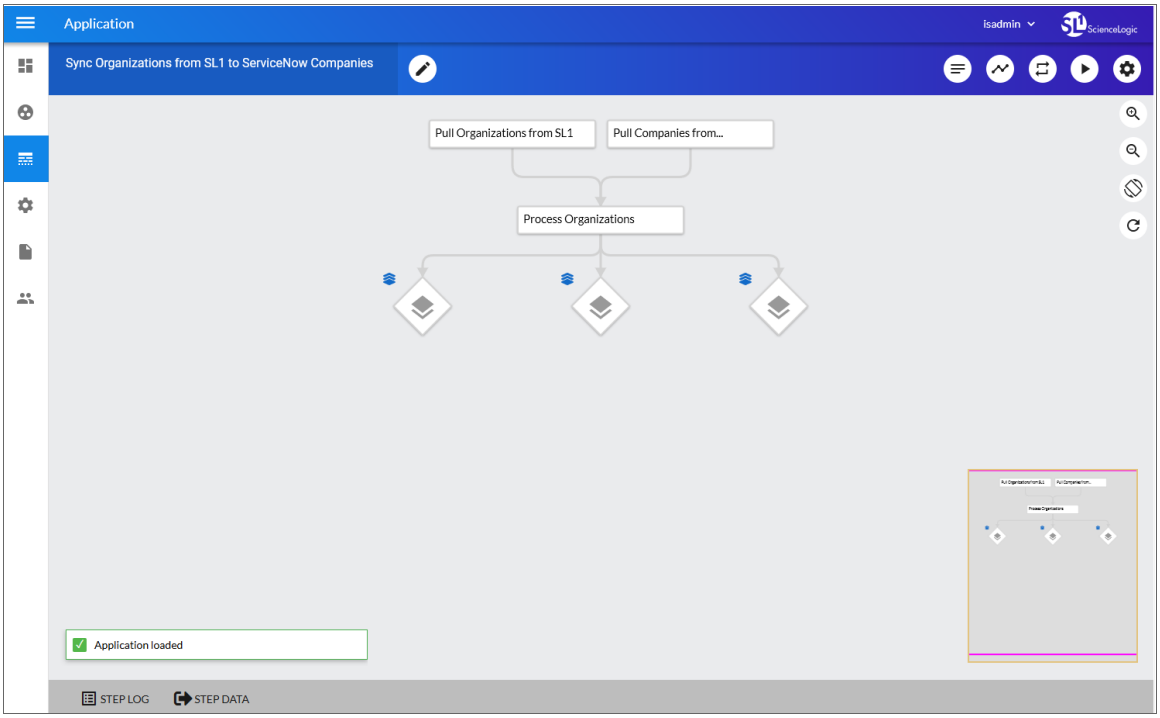
**NOTE:** After the 2.1.0 platform release, the *Integration Service* was rebranded as *SL1 PowerFlow*, and the *Automation Builder* was rebranded as *SL1 PowerFlow builder*.

This chapter covers the following topics:

# What is SL1 PowerFlow?

SL1 PowerFlow enables intelligent, bi-directional integration between SL1 and third-party applications to promote a unified management ecosystem. PowerFlow allows users to translate and share data between SL1 and other applications without the need for programming knowledge. PowerFlow is designed to provide high availability and scalability.

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




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

- ***Steps***. A step is a generic Python class that performs a single action. 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. Steps can be re-used in multiple PowerFlow applications. 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. In the image above, the steps display as part of the flowchart in the main viewing pane as well as the **Steps Registry** pane.
- ***Applications***. A PowerFlow application is a JSON object that includes all the information required for executing an integration on the PowerFlow platform. An application combines a set of 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. 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.

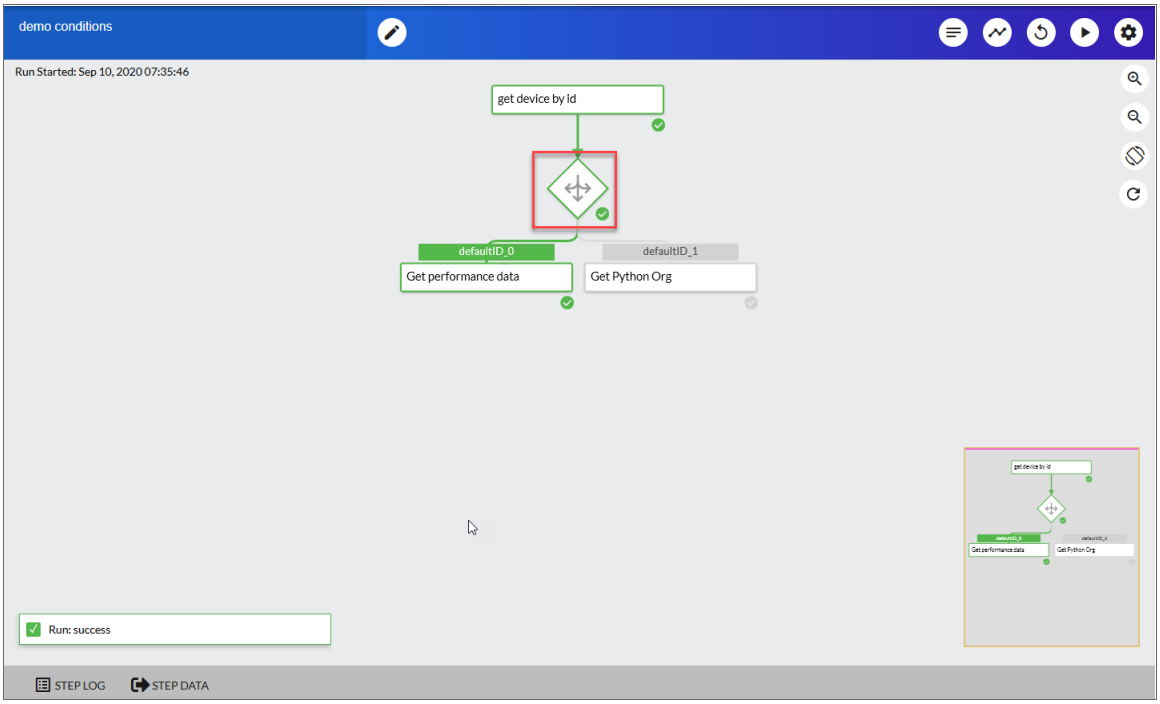
- ***Configuration Objects***. A configuration object is a stand-alone JSON file that contains a set of configuration variables used as input for an application. Configurations can include variables like hostname, user name, password, or other credential information. Configuration objects allow the same 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. You can access all configuration objects on the **Configurations** page ( ).
- ***Synchronization PowerPacks***. A Synchronization PowerPack (also called a SyncPack) contains all the code and logic needed to perform integrations on the PowerFlow platform. You can access the latest steps, applications, and configurations for PowerFlow or a third-party integration (such as ServiceNow, Cherwell, or Restorepoint) by downloading the most recent Synchronization PowerPack for that integration from ScienceLogic. You can access all Synchronization PowerPacks on the **SyncPacks** page ( ).

# What is the SL1 PowerFlow Builder?

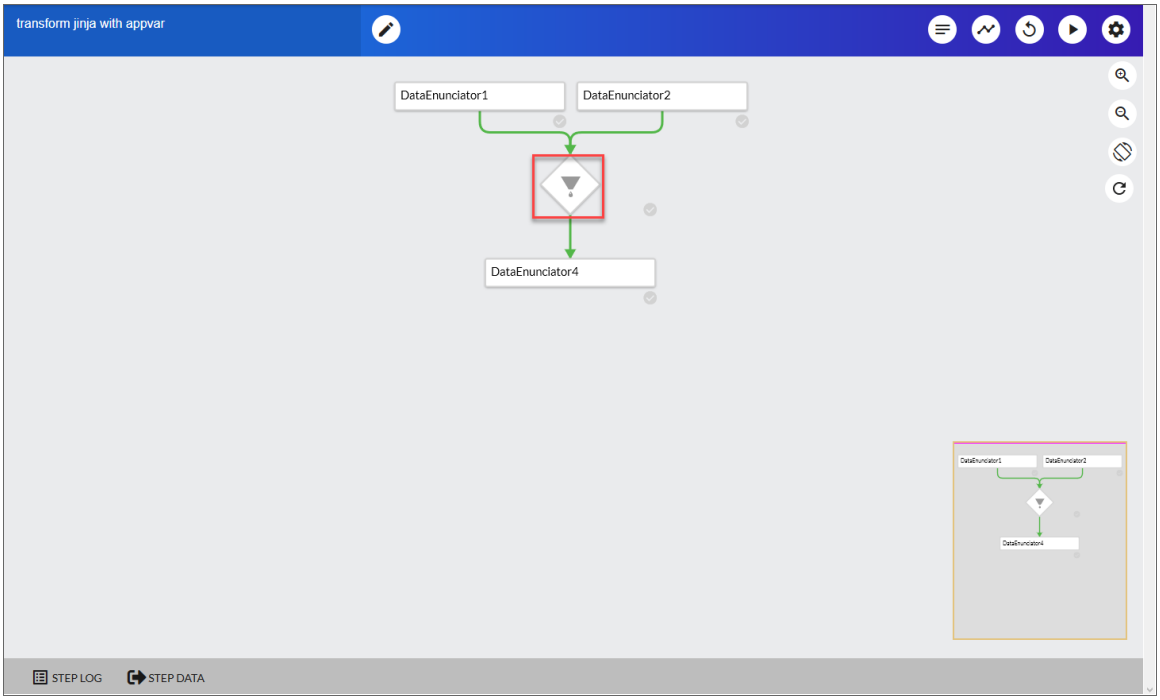
You can use the **SL1 PowerFlow builder** in the PowerFlow user interface to create complicated applications with logical branching and data transformation features using drag-and-drop components. You access the PowerFlow builder on the **Applications** page in the PowerFlow user interface.

**NOTE:** The PowerFlow builder is available only in SL1 Premium solutions. To upgrade, contact ScienceLogic Customer Support. For more information, see ScienceLogic Pricing.

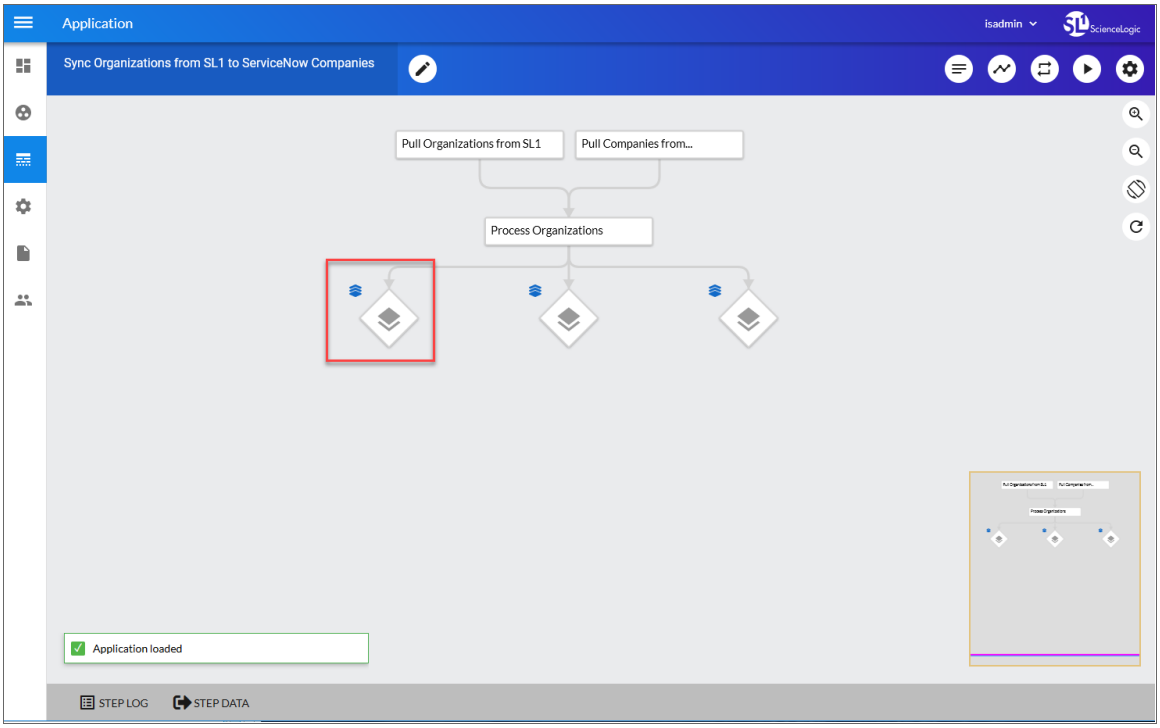
From the **Steps Registry** pane on an **Application** page, you can drag a **Condition** operator ( ) onto an application workflow to create the option for branching flows, such as If-Else or If-Then-Else statements:



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:



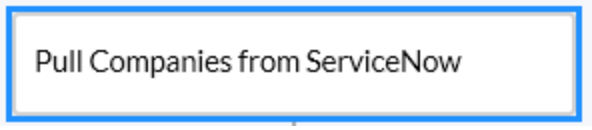
You can also use the **Trigger Application** operator ( ) to launch one or more PowerFlow applications from within a new or existing PowerFlow application. This operator uses the same functionality as the "Trigger Application" step from the Base Steps Synchronization PowerPack:




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

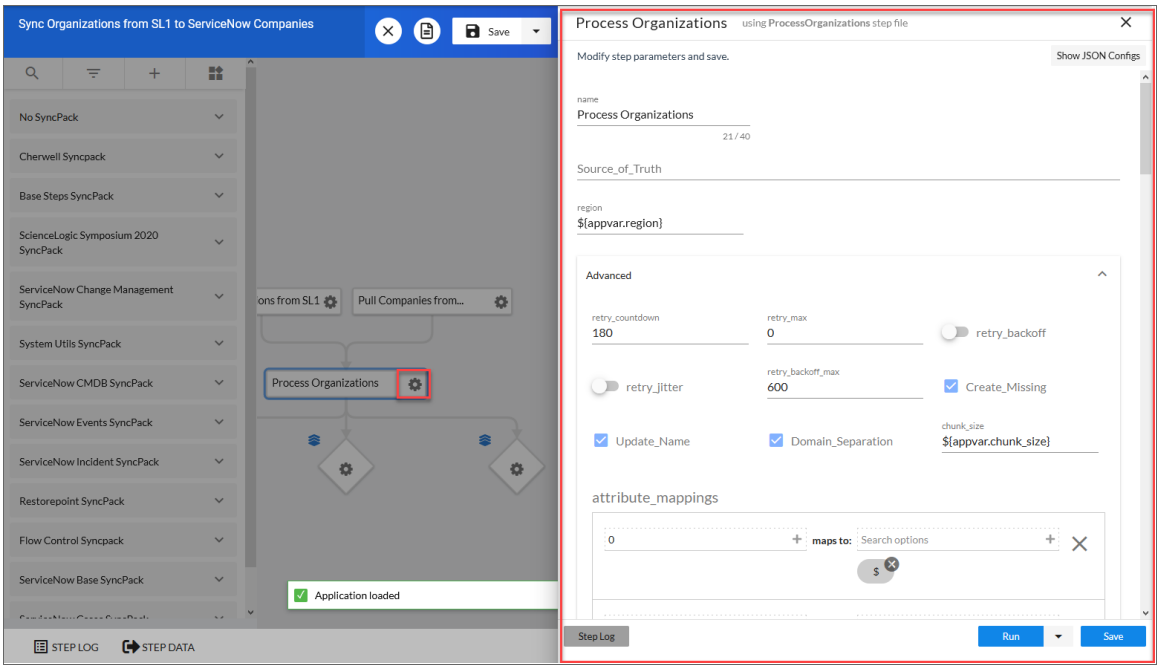
# What is a Step?

In PowerFlow, a ***step*** is a generic Python class that performs a single action, such as caching device data:




Steps accept 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.

You can view and edit the parameters for a step by opening a PowerFlow application from the **Applications** page, clicking **[Open Editor]** (), and then clicking the gear icon () on a step. The **Configuration** pane for that step appears:




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 add test data to the step and click the down arrow next to the **[Run]** button () and select *Custom Run* to run test data for that step.

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

Steps are grouped into the following types:

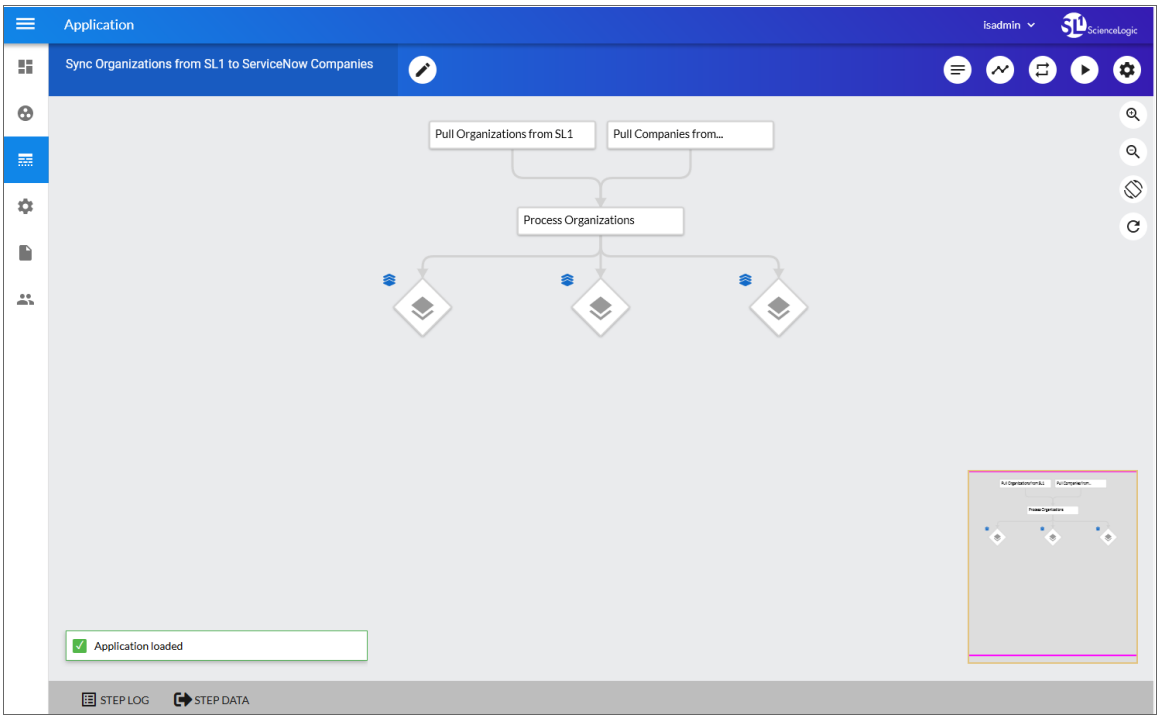
- ***Standard***. Standard steps do not require any previously collected data to perform. Standard steps are generally used to generate data to perform a transformation or a database insert. These steps can be run independently and concurrently.
- ***Aggregated***. Aggregated steps require data that 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.

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 file 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.

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.

Applications can be executed through the REST API and are processed as an asynchronous task in PowerFlow. During processing the user is provided a unique task ID for the application and each of its tasks. Using the task IDs, the user can poll for the status of the application and the status of each individual running step in the application.

Executing an application from the REST API allows the user to dynamically set one-time parameter values for the variables defined 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 applications, see *Managing PowerFlow Applications*.

# What is a Configuration Object?

Configuration variables are defined in a stand-alone JSON file called a ***configuration object*** that lives on the PowerFlow system and can be accessed by all PowerFlow applications and their steps.

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

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

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 together.

Each global variable in the configuration has the option of being 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*.

# Creating and Saving 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 the ***SL1 PowerFlow for Developers*** manual.

# Elements of the PowerFlow User Interface

With the 2.0.0 release of the PowerFlow platform, the PowerFlow user interface was updated to match the 8.12.0 and later release of the SL1 user interface, with the navigation tabs now located on the left-hand side of the window. The tabs provide access to the following pages: **Dashboard**, **SyncPacks**, **Applications**, **Configurations**, **Reports**, and **Admin Panel**.

**TIP:** To view a pop-out list of menu options, click the menu icon ( ). Click the menu icon again to close the pop-out menu.

## Logging In and Out of the PowerFlow User Interface

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

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default with 2.0.0 installations. If you are migrating from a previous version of PowerFlow to version 2.0.0, you can log in and authenticate with the same user and password.
- ***Basic Authentication***. PowerFlow 2.0.0 continues to support Basic Authentication as well. Because PowerFlowPowerPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication with PowerFlow version 2.0.0 or later.
- ***OAuth***. OAuth lets PowerFlow administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as LDAP or Active Directory, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.

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


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

**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.

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

The **SyncPacks** page ( ) lets you import, install, and view Synchronization PowerPacks, which are also called "SyncPacks". For more information, see *Managing Synchronization PowerPacks*.

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 *Viewing a Report for an Application*.

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

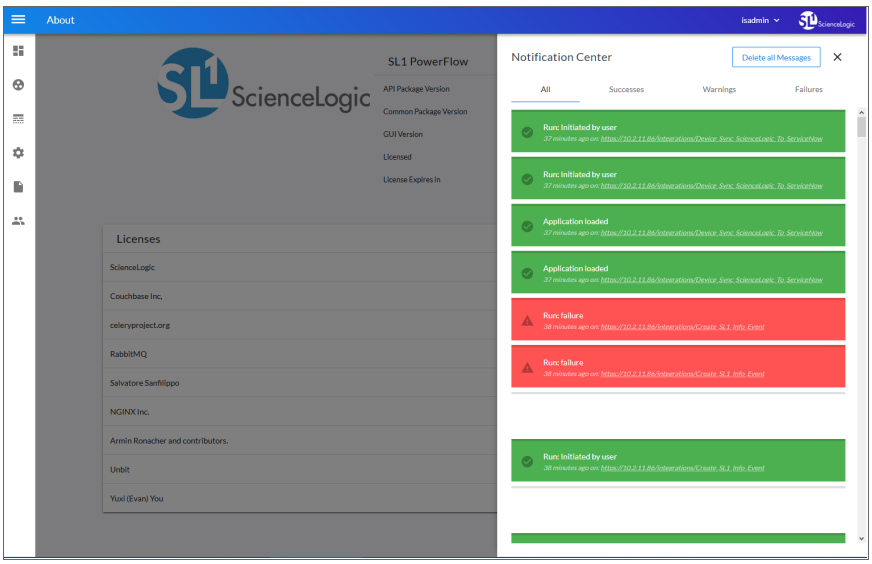
> **TIP:** While the **SyncPacks**, **Applications**, **Configurations**, **Reports** and **Admin Panel** pages are loading or running a procedure, you will see a dark green, animated line running across the top of the page until the process completes.

## Additional Navigation

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

- *About*. Displays version information about the PowerFlow version and the licenses used by PowerFlow. This page also display whether the PowerFlow system is licensed, and when the license expires.
- *Help*. Displays online Help for PowerFlow in a new browser window.

- *Notifications*. Opens the **Notification Center** pane, which contains a log of all previous notifications that appeared on the PowerFlow system about applications that were run successfully or with warnings or failures:




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 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 Messages]** button.

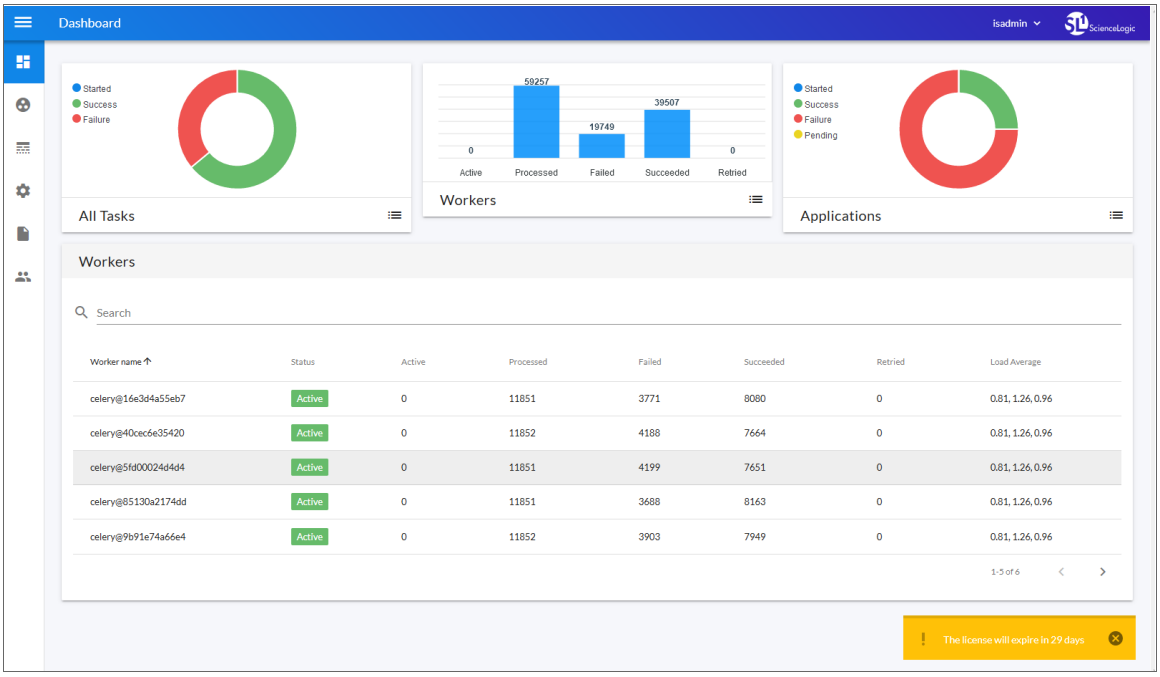
Click the Close icon (✕) to close the **Notification Center** pane.

- *Log off*. Logs you out of the PowerFlow user interface.

# Monitoring PowerFlow on the Dashboard Page

## Monitoring the Status of Tasks, Workers, and PowerFlow Applications

You can use the **Dashboard** 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:




The **Dashboards** page is the initial landing page after you log in to PowerFlow. This page displays high-level statistics about the health of the worker services that are being used by the PowerFlow instance.

To view more information on the **Dashboard** page:

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.
2. Click the List icon ( ) for the **All Tasks**, **Workers**, or **Applications** graphs to view a list of relevant tasks, workers, or applications. Use the left and right arrow icons to move through the list of items.
3. To view additional details about a specific tasks, click the List icon ( ) for the **All Tasks** graph and then click the link for the task, where relevant. The application aligned with that task appears, and you can select a step and view the **Step Log** details for that step.

**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 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:

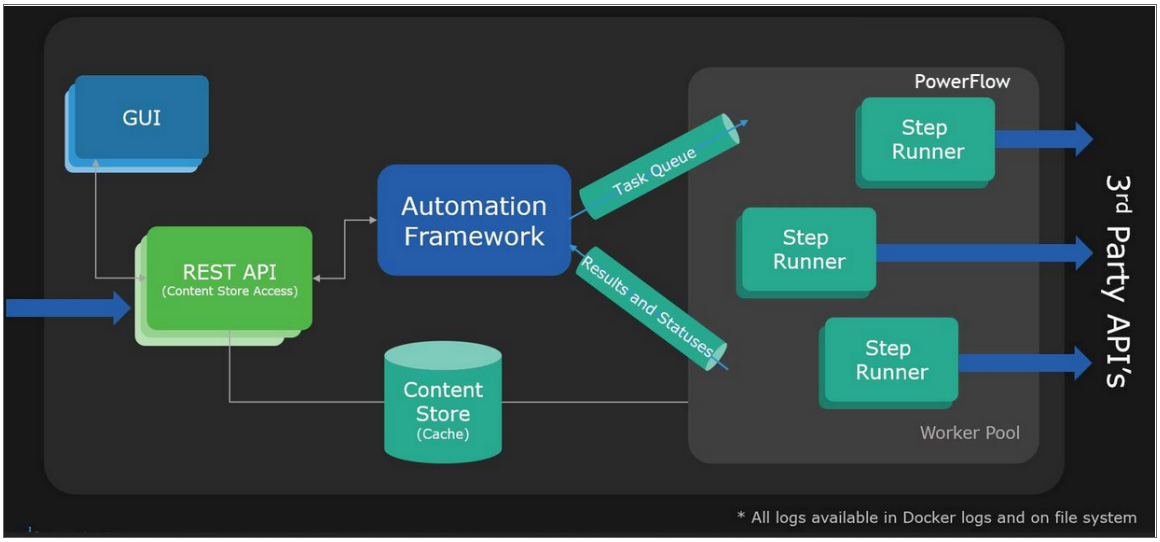
# PowerFlow Architecture

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:

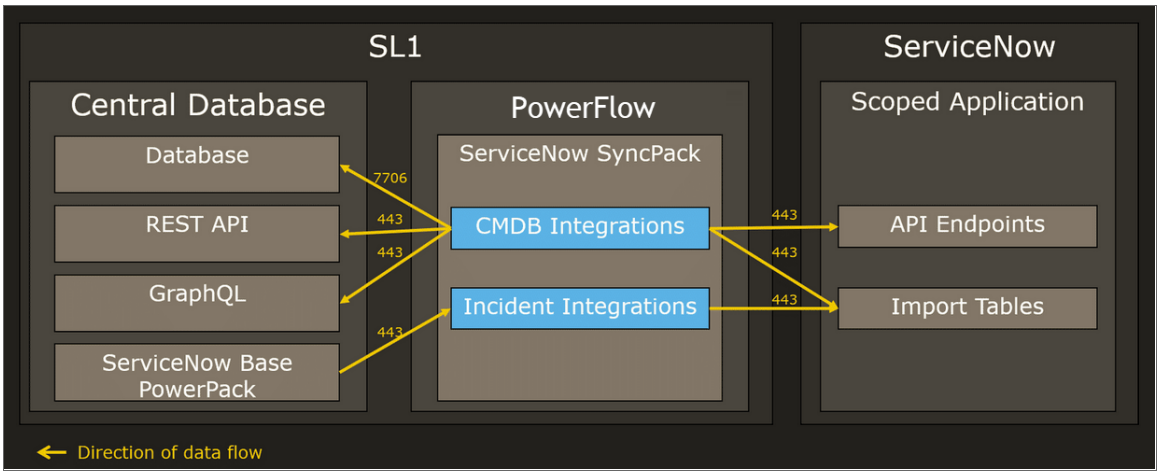



The PowerFlow includes the following containers:

- ***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.

## 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:

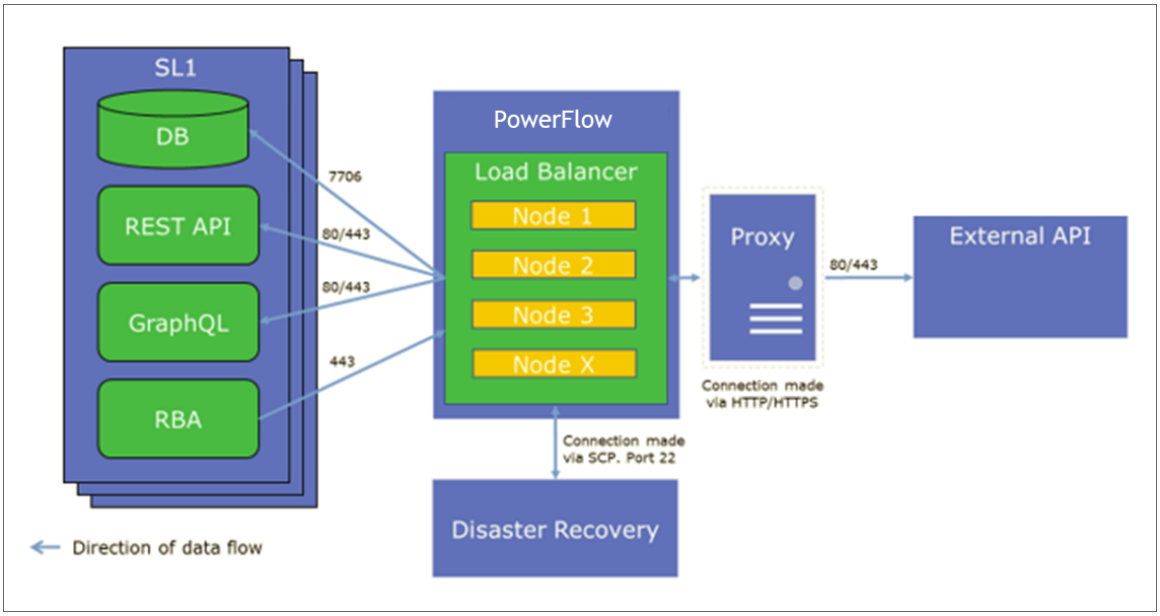
- ***SL1 Central Database***. PowerFlow communicates with the SL1 database over port 7706.
- ***SL1 REST API***. PowerFlow communicates with the SL1 REST API over port 443.
- ***GraphQL***. PowerFlow communicates with GraphQL over port 443.
- ***ServiceNow Base PowerPack***. In this example, the Run 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.

**NOTE:** 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 *Backing up Data*.
- 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 *Backing up Data*.

## 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 Synchronization PowerPacks for PowerFlow have minimum version dependencies. Please see the documentation for those Synchronization PowerPacks for more information on those dependencies.

The following table lists the port access required by PowerFlow:

| Source IP | PowerFlow Destination | PowerFlow Source Port | Destination Port | Requirement |
|---|---|---|---|---|
| PowerFlow | SL1 API | Any | TCP 443 | SL1 API Access |
| SL1 Run Book Action | PowerFlow | Any | TCP 443 | Send SL1 data to PowerFlow |
| Devpi | PowerFlow | Any | TCP 3141 | Internal Python package repository for Synchronization PowerPacks; check for self-certification for PowerFlow |
| Dex Server | PowerFlow | Any | TCP 5556 | Enable authentication for PowerFlow |
| PowerFlow | SL1 Database | Any | TCP 7706 | SL1 Database Access |
| powerflowcontrol (pfctl, formerly called iservicecontrol) command-line utility | PowerFlow | Any | 22 (on all host nodes) | Log in and perform admin tasks on nodes |

| Source IP | PowerFlow Destination | PowerFlow Source Port | Destination Port | Requirement |
|---|---|---|---|---|
| Encapsulated Security Protocol (ESP) | PowerFlow | IP Protocol 50 | n/a | Security; ESP should be open and available between cluster nodes |
| Couchbase Dashboard | PowerFlow | 8091 | n/a | Couchbase Dashboard (use your PowerFlow credentials) |
| RabbitMQ Dashboard | PowerFlow | 15672 | n/a | RabbitMQ Dashboard (use *guest*/*guest*for credentials) |

ScienceLogic highly recommends that you disable all firewall session-limiting policies. Firewalls will drop HTTPS requests, which results in data loss.

**CAUTION:** You can use only *one* SL1 system with only *one* third-party application (such as ServiceNow, Cherwell, or Restorepoint). For example, if you were using two ServiceNow systems with PowerFlow, you would need *two* SL1 systems to create two separate pairings of SL1 and ServiceNow.

**NOTE:** You can use a single PowerFlow system to manage multiple pairings of SL1 and a third-party application. The pairings must always be one to one: one SL1 system connected to one third-party application.

**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.

**NOTE:** 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.

**TIP:** For more information about system requirements for your PowerFlow environment, see the System Requirements page at the ScienceLogic Support site.

### 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.

> **NOTE:** 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. 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

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 *upgrading* an existing version of the PowerFlow, see ***Upgrading PowerFlow***.

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

# Installing PowerFlow via ISO

## Locating the ISO Image

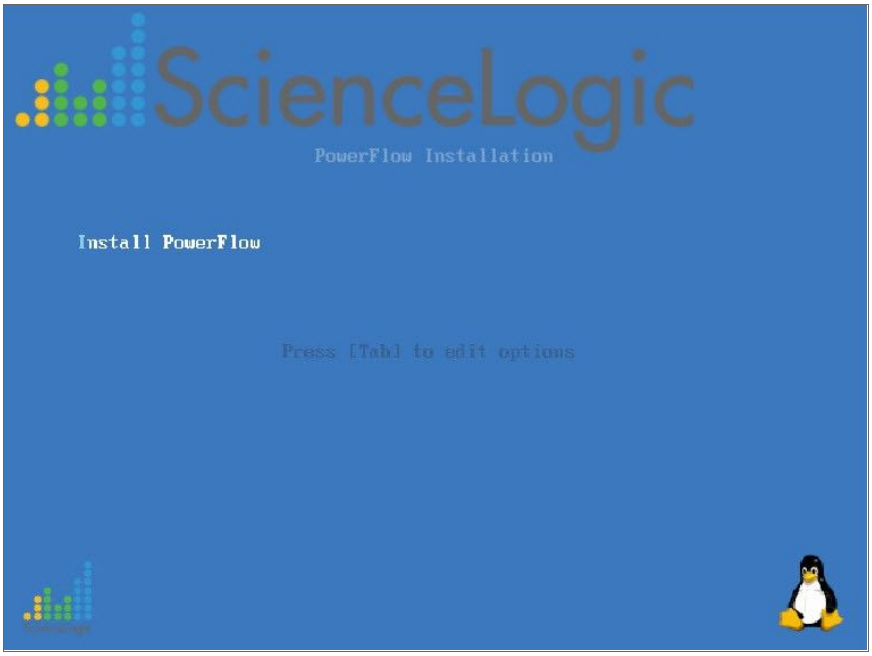
To locate the PowerFlow ISO image:

1. Go to the ScienceLogic Support site at https://support.sciencelogic.com/s/.
2. Click the **Product Downloads** tab and select *PowerFlow*. 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.

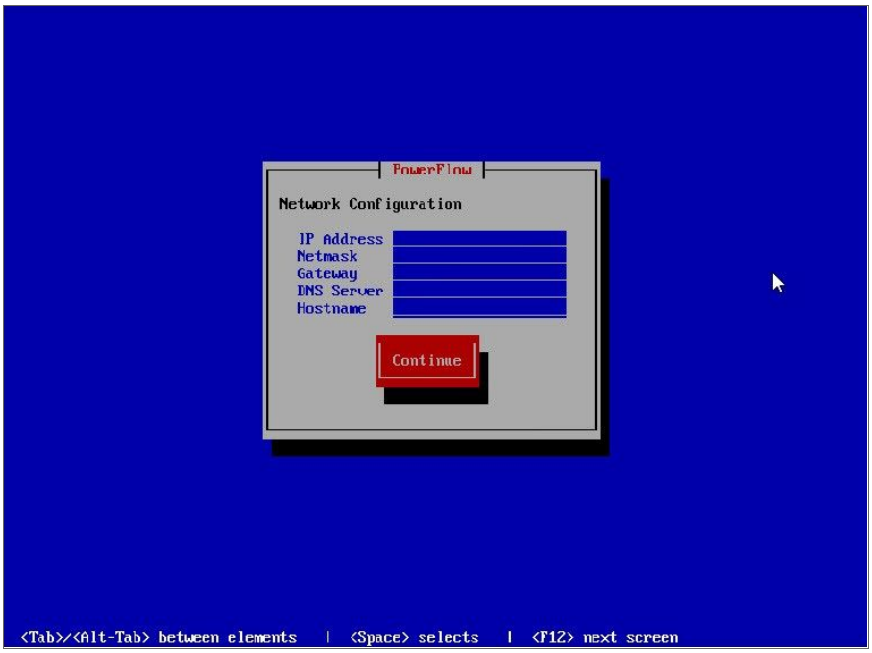
## Installing from the ISO Image

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:

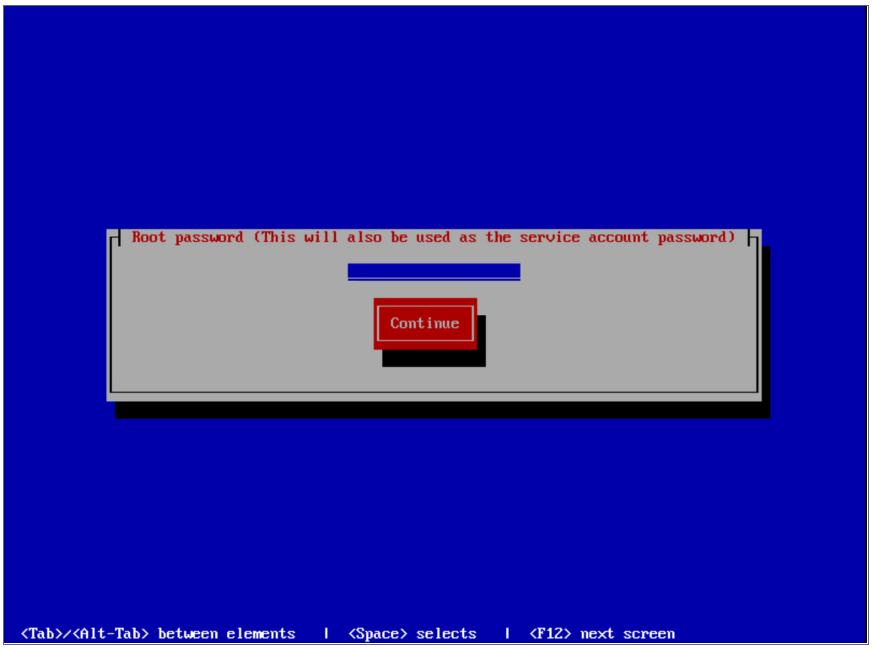


3. Select *Install Integration Service*. After the installer loads, the **Network Configuration** window appears:




4. 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.

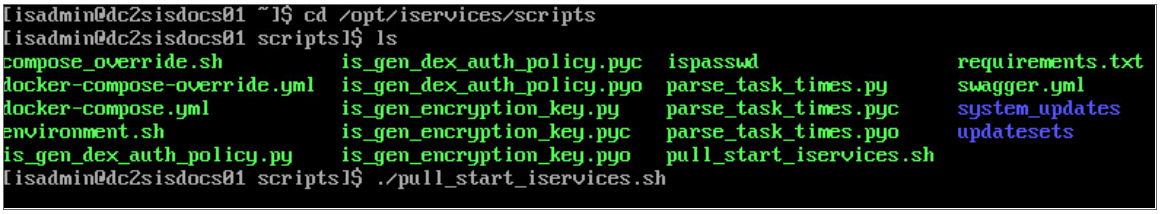
> **NOTE:** You use this password to log into the PowerFlow user interface, to SSH to the PowerFlow server, and to verify API requests and database actions. This password is set as both the "Linux host isadmin" user and in the **/etc/iservices/is_pass** file that is mounted into the PowerFlow stack as 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*.

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*.

14. 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
```

# Installing PowerFlow via RPM to a 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

The following procedure describes how to install PowerFlow via RPM to Amazon Web Service (AWS) EC2. 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 on any Oracle Linux 7 or later 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.

**WARNING:** If you install the PowerFlow system on any operating system *other* than Oracle Linux 7, 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 7.

**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:** 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.

**NOTE:** If you are installing the RPM in a cluster configuration, and you want to distribute traffic between the nodes, a load balancer is required.

**NOTE:** 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.

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

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

   **TIP:** 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 7.6 virtual machine by searching for **OL7.6-x86_64-HVM** in the ***Search for an AMI*** field.
3. Click the ***results*** link for Community AMIs.
4. Click **[Select]** for a virtual machine running Oracle Linux 7.6 or greater for installation. The following image shows an example of an **OL7.6-x86_64-HVM-*** AMI file:


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:

    - Only inbound port 443 needs to be exposed to any of the systems that you intend to integrate.
    - For PowerFlow version 1.8.2 and later, 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-2.x.x-1.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:

```
sudo yum install yum-utils
sudo yum-config-manager --enable ol7_latest
sudo yum-config-manager --enable ol7_optional_latest
```

9. Run the following commands to update and install the host-level packages, and to upgrade to Python 3.6:

```
sudo yum remove python34-pip python34-setuptools python3
sudo yum --setopt=obsoletes=0 install python36-pip python36 python36-setuptools
python36-devel openssl-devel gcc make kernel
sudo yum update
sudo yum install python36-pip
```

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

```
sudo yum install -y wget

sudo pip install --upgrade pip

sudo pip install docker-compose

wget https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-19.03.5-3.el7.x86_64.rpm && sudo yum install docker-ce-19.03.5-3.el7.x86_64.rpm
```

> **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/.

> **WARNING:** 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.

11. Create the Docker group:

```
sudo groupadd docker
```

12. Add your user to the Docker group:

```
sudo usermod -aG docker $USER
```

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:

```
sudo setenforce 0
sudo vim /etc/sysconfig/selinux
SELINUX=permissive
sudo systemctl enable docker
sudo systemctl start docker
sudo yum install yum-utils
sudo yum-config-manager –enable ol7_addons ol7_optional_latest ol7_latest
sudo yum-config-manager –disable ol7_ociyum_config
wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo rpm -Uvh epel-release-7*.rpm
sudo yum update
sudo pip install docker-compose
sudo yum install firewalld
systemctl enable firewalld
systemctl start firewalld
systemctl disable iptables
```

15. Run the following firewall commands as "sudo":

```
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`

16. Copy the PowerFlow RPM to the instance of installation and install the RPM:

```
sudo yum install sl1-powerflow
systemctl restart docker
```

17. Create a password for PowerFlow:

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

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

18. Pull and start iservices to start PowerFlow:

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

**NOTE:** 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.

**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.

**NOTE:** 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."
```

# Upgrading Oracle Linux Operating System Packages from ISO

ScienceLogic releases a major update to PowerFlow every six months via ISO and RPM. ScienceLogic also releases a monthly maintenance release (MMR) as needed to address major customer-facing bugs via ISO and RPM. 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.

All ISO builds of PowerFlow (major updates and MMRs) include the most recent, stable version of the Oracle Linux 7 operating system (OS). If there are OS vulnerabilities discovered in PowerFlow, you will need to either patch the vulnerability yourself using `yum` or wait for the next PowerFlow ISO.

Upgrading OS packages for an offline deployment require the following manual steps to mount the ISO and update the packages packaged with the ISO.

1. Mount the PowerFlow ISO onto the system:

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

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:

```
[localISISOMount]
name=Locally mounted IS ISO for packages
enabled=1
baseurl=file:///mnt/tmpISMount
gpgcheck=0
```

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

3. Optionally, you can import the latest GNU Privacy Guard (GPG) key to verify the packages by running the following commands:

```
rpm --import /mnt/repo_keys/RPM-GPG-KEY-Oracle
rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-Docker-ce
rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-EPEL-7
```

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

```
yum update
```

# Upgrading PowerFlow

Upgrading to the latest version of the PowerFlow platform *will* involve some downtime of PowerFlow. Before upgrading your version of PowerFlow, ScienceLogic recommends that you make a backup of your PowerFlow system. For more information, see *Backing up Data*.

You can upgrade PowerFlow from any 1.8.x version or later to the latest version. Select the relevant procedure below based on your upgrade path:

- *Upgrading from Version 2.x.x*
- *Upgrading from Version 1.8.x with the Upgrade Script*
- *Manually Upgrading from Version 1.8.x*

**TIP:** As a best practice, you should *always* upgrade to the most recent version of PowerFlow that is currently available at the PowerFlow Support page.

**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.

**WARNING:** If you made any customizations to default applications or steps that shipped with previous versions of PowerFlow, you will need to make those customizations compatible with Python 3.6 or later *before* upgrading to version 2.0.0 or later of PowerFlow.

**WARNING:** 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 or back up those custom configurations before upgrading, or contact ScienceLogic Support to prevent the loss of those modifications.

**NOTE:** If you are installing the RPM in a cluster configuration, and you want to distribute traffic between the nodes, a load balancer is required.

**TIP:** To help address any issues with the upgrade, see *Troubleshooting Upgrade Issues*.

## Upgrading from Version 2.x.x

To upgrade to the latest version from version 2.x.x:

1. Download the PowerFlow RPM and copy the RPM file to the PowerFlow system.
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. Type the following at the command line:

   ```
   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**.

   **NOTE**: If you are running PowerFlow in a clustered environment, install the RPM on all nodes in the cluster before continuing with the remaining steps.

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

   ```
   docker stack rm iservices
   ```

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

   ```
   systemctl restart docker
   ```

   **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 take the services slightly longer to update.

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

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

8. 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. You can use the visualizer at port 8080 to monitor the progress of the updates.
9. 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.

## Upgrading from Version 1.8.x with the Upgrade Script

**TIP:** As a best practice, you should *always* upgrade to the most recent version of PowerFlow that is currently available at the PowerFlow Support page.

To upgrade to the latest version of PowerFlow from version 1.8.x:

1. Upgrade the host packages and Python 3.6 (previous versions of PowerFlow used Python 2.6).
2. Upgrade to Oracle 7.3 or later.
3. Upgrade to Docker version 18.09.2 or later.

   **NOTE:** PowerFlow version 2.0.0 or later requires the **docker-ce 18.09.2** or later version of Docker. The PowerFlow ISO installs the **docker-ce 19.03.5-3** version of Docker by default, but if you are upgrading to this version from the RPM, you must upgrade Docker *before* you upgrade PowerFlow with the RPM.

4. Install the PowerFlow upgrade RPM.
5. Update the PowerFlow system from Basic Authentication to OAuth 2.0. For more information, see *Configuring Authentication with the PowerFlow*.

6. Set up licensing for PowerFlow. You must license your PowerFlow system to enable all of the features. If you are not deploying PowerFlow on a production or pre-production environment, you can skip licensing. For more information, see *Licensing the PowerFlow*.

> **NOTE:** If you are upgrading from a version before 1.8.3, be sure to review the release notes for the older version for any relevant update considerations before upgrading. For example, there is a small port change that you might need to apply if you are upgrading a customized cluster from a version of PowerFlow before version 1.8.3.

You will need to run the **is_upgrade_to_v2.sh** script to perform the upgrade steps automatically. The script upgrades the PowerFlow system from 1.8.x to 2.0.0 or later.

To locate the upgrade script:

1. At the ScienceLogic Support site, click the **Product Downloads** tab and select *PowerFlow*. The **PowerFlow Release** page appears.
2. Click the relevant "Integration Service 2.0" link. The **Release Version** page appears.
3. In the **Release Files** section, click the "1.8.X to 2.X.X Upgrade Script" link. A **Release File Details** page appears.
4. Click **Download File** on the **Release File Details** page. The **is_upgrade_to_v2.sh** script is in the **is_upgrade_tools.zip** file.

The upgrade script runs the following steps:

1. Checks to see if the Oracle version is greater or equal to 7.3. If not, the script stops and displays a message that you need to update to Oracle 7.3 or later.
2. Sets the requirements location and either mounts the ISO or verifies that the RPM exists. For this step, the script asks if the installation will be offline or online, and it also asks you for the location of the RPM.
3. Installs Python 3.6.
4. Installs Docker 19.03.5.
5. Installs the PowerFlow RPM.
6. Runs the **pull_start_iservices.sh** script to deploy and initialize PowerFlow.
7. If the upgrading process was offline, cleans the changes made for the upgrading process, unmounts the ISO, and removes the **localiso.repo** file.

To run the upgrade script:

1. Download the **is_upgrade_to_v2.sh** script and add it to a directory on the PowerFlow system.
2. Download the **sl1-powerflow-2.x.x-1.x86_64.rpm** file or the ISO file and add it to a directory on the PowerFlow system. Make a note of this directory, because you will need it for Step 2 in the script.

> **TIP:** Alternately, instead of downloading the RPM file, you can specify an online location for the RPM file.

3. If needed, run the following command on the PowerFlow system to give the script execution permissions:

```
sudo chmod +x is_upgrade_to_v2.sh
```

> **NOTE:** If you are installing a version of PowerFlow later than 2.0.0, you will need to update the version number in the command, above.

4. Change directory to the directory containing the **is_upgrade_to_v2.sh** script, such as **/home/isadmin/**, and then run the following command to execute the upgrade script:

```
sudo ./is_upgrade_to_v2.sh
```

> **NOTE:** If you are installing a version of PowerFlow later than 2.0.0, you will need to update the version number in the command, above.

5. For Step 2 of the script, you will need to specify if you want to run the upgrade online, or run it offline. Type "1" if you have Internet access, or "2" if you want to run the update offline.

6. For Step 2 of the script, you will need to specify the location of the RPM or ISO file for 2.0.x. You can use a location on the PowerFlow system for the RPM or ISO file, or an online location for the RPM. For example: **/home/isadmin/sl1-powerflow-2.1.0-1.x86_64.rpm**

7. After the upgrade script completes, perform the following steps to verify the upgrade:

   - Review the **docker-compose.yml** file and ensure that all environmental changes are in place.
   - If the **docker-compose.yml** file is ready to be deployed, you can re-deploy the PowerFlow stack.
   - After the PowerFlow stack is up and running, run the ***healthcheck*** action with the **powerflowcontrol** (pfctl) command-line utility to verify that you had a healthy deployment.
   - If needed, run the ***healthcheck*** action with the **powerflowcontrol** (pfctl) utility to automatically fix any remaining inconsistencies after the upgrade.

   > **TIP:** For more information about the **iservicecontrol** (pfctl) utility, see *Using the powerflowcontrol (pfctl) Command-line Utility*.

8. 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
```

9. As needed, update the PowerFlow system from Basic Authentication to OAuth 2.0. For more information, see *Configuring Authentication with PowerFlow*.

10. As needed, set up licensing for the PowerFlow. For more information, see *Licensing PowerFlow*.

> **NOTE:** If you are upgrading a clustered PowerFlow environment from 1.8.x to 2.0.0 or later, see *Updating Cluster Settings when Upgrading from 1.8.x to 2.0.0 or Later*.

## Manually Upgrading from Version 1.8.x

Instead of running the upgrade script, you can manually upgrade to the latest version from 1.8.x by following the detailed instructions below.

> **NOTE:** If you are upgrading from a version before 1.8.3, be sure to review the release notes for the older version for any relevant update considerations before upgrading. For example, there is a small port change that you might need to apply if you are upgrading a customized cluster from a version of PowerFlow before version 1.8.3.

### Step 1. Upgrading Host Packages and Python 3.6

To access the host packages online:

1. To make sure that all repositories can access the required host-level packages, enable the necessary repositories by running the following commands on the PowerFlow system:

   ```
   sudo yum install yum-utils
   sudo yum-config-manager --enable ol7_latest
   sudo yum-config-manager --enable ol7_optional_latest
   ```

2. Run the following commands to update and install the host-level packages, and to upgrade to Python 3.6:

   ```
   sudo yum remove python34-pip python34-setuptools python3
   sudo yum --setopt=obsoletes=0 install python36-pip python36 python36-setuptools
   python36-devel openssl-devel gcc make kernel
   sudo yum update
   ```

3. Continue the upgrade process by upgrading to Oracle 7.3 or later.

If you need to upgrade the host packages offline, without Internet access, you can mount the latest PowerFlow ISO file onto the system and create a yum repository configuration that points to the local mount point in **/etc/yum.repos.d**. After the ISO is mounted, you can import the latest GNU Privacy Guard (GPG) key used by the repository.

To access the host packages offline:

1. Mount the PowerFlow ISO onto the system:

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

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:

```
[localISISOMount]
name=Locally mounted IS ISO for packages
enabled=1
baseurl=file:///mnt/tmpISMount
gpgcheck=0
```

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

3. Optionally, you can import the latest GNU Privacy Guard (GPG) key to verify the packages by running the following command:

```
rpm --import /mnt/repo_keys/RPM-GPG-KEY-Oracle
rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-Docker-ce
rpm --import /mnt/tmpISMount/repo_keys/RPM-GPG-KEY-EPEL-7
```

4. If you cannot install Docker or Python offline, delete the other repository references by running the following command (ScienceLogic recommends that you back up those file first):

```
rm -rf /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel-testing.repo
/etc/yum.repos.d/public-yum-ol7.repo
```

5. Run the following commands to update and install the host-level packages, and to upgrade to Python 3.6:

```
sudo yum remove python34-pip python34-setuptools python3
sudo yum --setopt=obsoletes=0 install python36-pip python36 python36-setuptools
python36-devel openssl-devel gcc make kernel
sudo yum update
```

6. Continue the upgrade process by upgrading to Oracle 7.3 or later.

## Step 2. Upgrading to Oracle 7.3 or Later

ScienceLogic recommends that you update the version of Oracle Linux running on the PowerFlow to 64-bit version 7.3 or later.

> **NOTE:** If you want to upgrade to Oracle Linux to 7.6, see https://docs.oracle.com/en/operating-systems/oracle-linux/7/relnotes7.6/.

To upgrade to Oracle 7.3 or later:

1. To check the current version of Oracle Linux on your PowerFlow system, run the following command on the PowerFlow system:

```
cat /etc/oracle-release
```

2. To install Oracle, choose one of the following procedures:

   - Using a public Oracle Linux repository, run the `yum update` command.
   - By mounting the latest PowerFlow ISO to the system and installing the latest packages from the ISO.

3. Continue the upgrade process by updating Docker.

## Step 3. Upgrading to Docker 18.09.2 or later

PowerFlow systems before version 2.0.0 included Docker 18.06. If you are running a version of the PowerFlow before version 2.0.0, you will need to update Docker 18.09.2 or later to be able to upgrade to PowerFlow version 2.0.0 or later. For more information about the security updates included in Docker 18.09.2, see https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5736.

Before upgrading Docker, ScienceLogic recommends that you review the following information from the Docker product manual: https://docs.docker.com/ee/upgrade/.

**NOTE:** Run the following process on Docker Swarm node, starting with the manager nodes.

**WARNING:** For clustered configurations, see the information in *Installing Docker in Clustered Configurations* before running the upgrade steps below.

To upgrade to Docker 18.09.2 or later:

1. Review the steps in the Docker product manual: https://docs.docker.com/ee/docker-ee/oracle/#install-with-a-package.

2. To install docker-io, run the following command on the PowerFlow instance:

```
sudo yum install -y https://download.docker.com/linux/centos/7/x86_
64/stable/Packages/containerd.io-1.3.7-3.1.el7.x86_64.rpm
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-
19.03.5-3.el7.x86_64.rpm
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-
cli-19.03.5-3.el7.x86_64.rpm
```

**NOTE:** You will need to update the Docker versions in this command if there are more recent versions on the Docker Download page:https://download.docker.com/linux/centos/7/x86_64/stable/Packages/.

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

3. To install docker-ce offline using the IS 2.0.*x* ISO, run the following commands:

```
rpm -e --nodeps docker-ce
sudo yum install -y /mnt/tmpISMount/Packages/containerd.io-1.3.7-3.1.el7.x86_64.rpm
sudo yum install -y /mnt/tmpISMount/Packages/docker-ce-cli-19.03.5-3.el7.x86_64.rpm
sudo yum install -y /mnt/tmpISMount/Packages/docker-ce-19.03.5-3.el7.x86_64.rpm
sudo systemctl enable docker
sudo systemctl start docker
```

> **NOTE:** If the node is a member of a cluster, wait for a few minutes, and the node will automatically rejoin the swarm cluster, and re-deploy the services running on that node. Wait until all services are operational, then proceed to upgrade the next node. For more information, see ***Installing Docker in Clustered Configurations***.

4. Continue the upgrade process by ***installing the PowerFlow RPM***.

## Installing Docker in Clustered Configurations

Follow the best practices for upgrading a cluster described in the Docker product manual: https://docs.docker.com/ee/upgrade/#cluster-upgrade-best-practices.

> **NOTE:** You should upgrade all manager nodes before upgrading worker nodes. Upgrading manager nodes sequentially is recommended if live workloads are running in the cluster during the upgrade. After you upgrade the manager nodes, you should upgrade worker nodes, and then the Swarm cluster upgrade is complete.

Docker recommends that you drain manager nodes of any services running on those nodes If a live migration is expected, all workloads must be running on swarm workers, not swarm managers, or the manager under upgrade needs to be completely drained.

Also, a new Python Package Index (PyPI) service was added to the PowerFlow stack. When deploying PowerFlow in a cluster setup, and not using network-aware volumes, the PyPI server must be "pinned" to a specific node with constraints. Pinning the PyPI server to a single node ensures that its persistent volume containing the Synchronization PowerPacks will always be available to PowerFlow.

## Step 4. Installing the PowerFlow RPM

To update PowerFlow using the RPM:

1. Download the PowerFlow RPM and copy the RPM file to the PowerFlow system.
2. Log in as *isadmin* with the appropriate (root) password. You must be root to upgrade the RPM file.

3. Type the following at the command line:

   ```
   sudo rpm -Uvh full_path_of_rpm
   ```

   where *full_path_of_rpm* is the full path and name of the RPM file.

4. Run the **pull_start_iservices.sh** script to deploy and initialize 2.x.x:

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

   > WARNING: Do not run the **pull_start_iservices.sh** script if you are using PowerFlow in a clustered environment.

5. 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
   ```

6. Verify that each service now uses the new version of PowerFlow.

7. As needed, update the PowerFlow system from Basic Authentication to OAuth 2.0. For more information, see *Configuring Authentication with PowerFlow*.

8. As needed, set up licensing for PowerFlow. For more information, see *Licensing PowerFlow*.

> NOTE: If you are upgrading a clustered PowerFlow environment from 1.8.x to 2.0.0 or later, see *Updating Cluster Settings when Upgrading from 1.8.x to 2.0.0 or Later*.

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

You can use the PowerFlow command-line tool (iscli) to upload customer 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 https://127.0.0.1:3141/isadmin/dependencies
   devpi upload <location of your package dependencies>
   ```

## Updating Cluster Settings when Upgrading from 1.8.x to 2.0.0 or Later

After upgrading PowerFlow cluster nodes from 1.8.x to 2.0.0 or later, you need to verify the following details in the **docker-compose** file:

1. Verify that the pypiserver is pinned to the node that contains the Synchronization PowerPacks: **node.hostname** == ***<node hostname from docker node ls>***. pypiserver is not a critical service, so it is not replicated.
2. Verify that the dexserver is replicated three times to keep the pypiserver service from moving from one node to another and so you no longer have the persisted Synchronization PowerPack storage.

You can use the **healthcheck** action with the **powerflowcontrol** (pfctl) command-line utility to validate the **docker-compose** file. The **healthcheck** action will show a message if the pypiserver or dexserver services are not well-configured in the **docker-compose** file. You can fix these issues manually, or you can fix them using the **autoheal** action with the **powerflowcontrol** utility. The utility corrects the **docker-compose** file and copies it to all the nodes in the cluster environment.

> **NOTE:** When using the version of the **powerflowcontrol** command-line utility that comes with PowerFlow version 2.1.0 or later, the **autocluster** action validates and fixes the pypiserver and dexserver services definitions in the **docker-compose** file.

For more information, see *Using the powerflowcontrol (pfctl) Command-line Utility*.

# Troubleshooting Upgrade Issues

The following topics describe issues that might occur after the upgrade to version 2.0.0 or later, and how to address those issues.

## Cannot mount the virtual environment, or the virtual environment is not accessible

If the docker container does not properly mount the virtual environment, or the virtual environment is not accessible to the environment, you might need to remove and re-deploy the service to resolve the issue.

## To roll back to a version before PowerFlow 2.0.0 or later

After a schedule is accessed or modified on the 2.0.0 or later PowerFlow API or scheduler, that schedule will not be accessible again in a 1.x version. If you upgrade to 2.0.0 from 1.x and you want to go back to the 1.x release, you must delete the schedule and recreate the schedule in 1.x (if the schedule was modified in 2.0.0).

## Cannot access PowerFlow or an Internal Server Error occurs

PowerFlow version 2.0.0 and later use a new type of authentication session. This change might cause problems if your browser attempts to load the PowerFlow user interface using a "stale" cache from version 1.8.4. If you have issues accessing the user interface, or if you see an "Internal server error" message when you log in, be sure to clear the local cache of your browser.

### After upgrading, the syncpack_steprunner fails to run

This error flow tends to happen when the syncpack_steprunner is deployed, but the database is not yet updated with the indexes necessary for the Synchronization PowerPack 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:

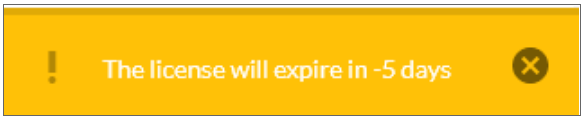
```
initialize_couchbase -s
```

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

## Licensing PowerFlow

Before users can access all of the features of version 2.0.0 or later 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 yellow dialog box appears at the bottom right of the screen that displays when the license expires. You can also track your licensing information on the **About** page (username menu > About). If the number is a negative number, that means your license is that many days past expiration. 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 Synchronization PowerPack 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>
   ```

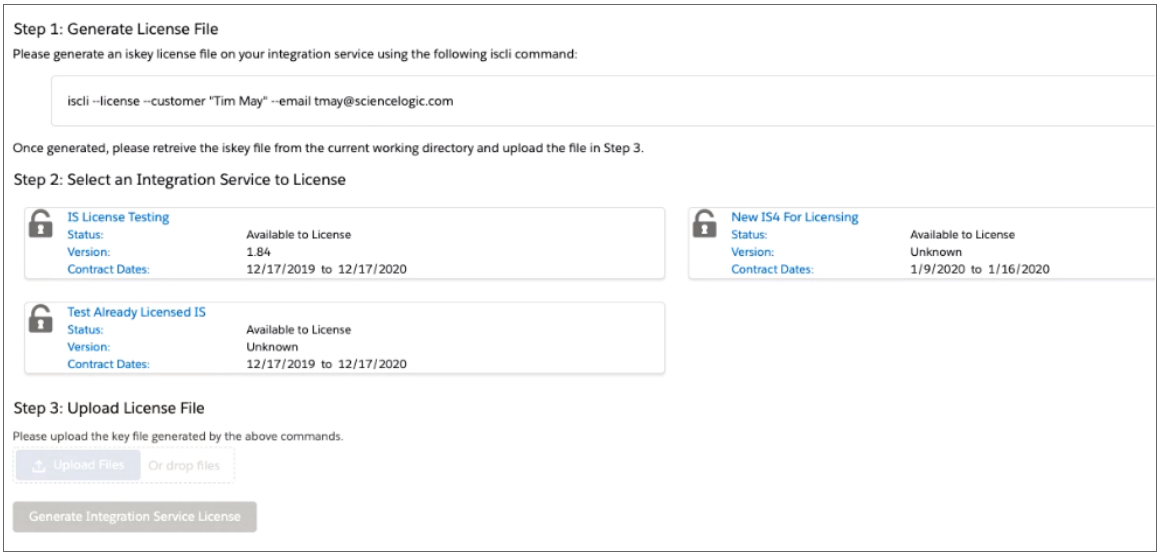
   where `<user_name>` is the first and last name of the user, and `<user_email>` is the user's email address. For example:

   ```
   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:



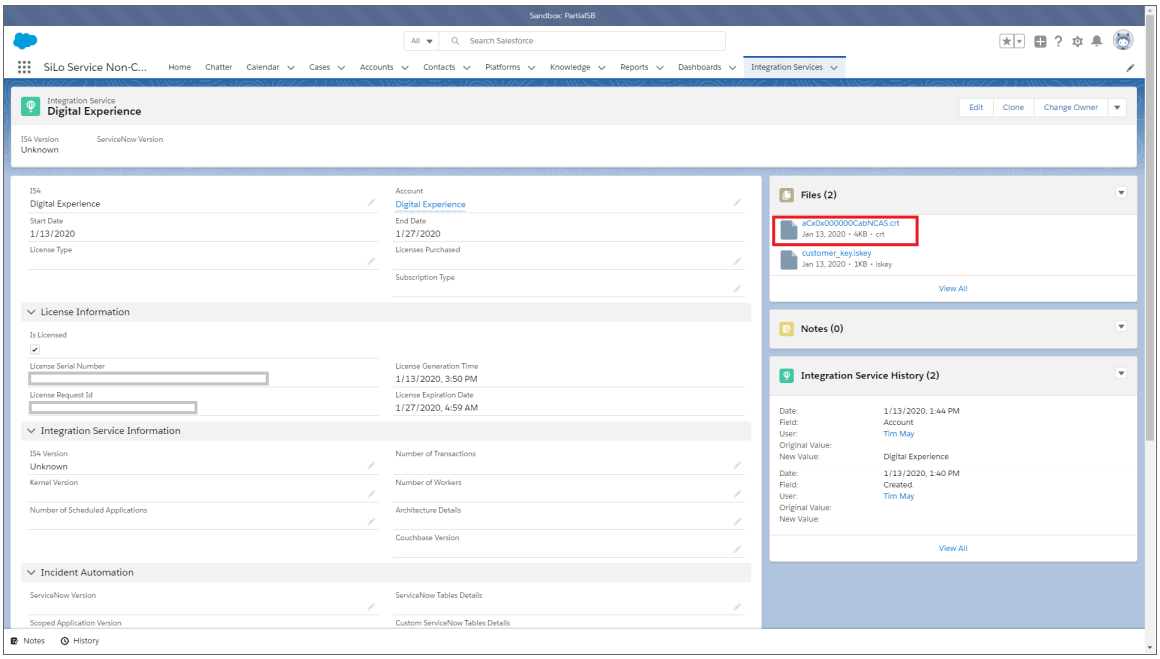

5. For Step 2 of the "Generate License File" process, select the PowerFlow record you want to license.

> **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 Synchronization PowerPacks published by ScienceLogic and ScienceLogic Professional Services, and to run and schedule PowerFlow applications from those Synchronization PowerPacks. 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 Synchronization PowerPacks 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.

**NOTE:** If you are upgrading from PowerFlow version 2.x.x to version 2.2.0, you will need to upgrade your license to the Advanced or Premium solution to be able to upload custom content (such as steps and applications) for Synchronization PowerPacks, or if you want to use the PowerFlow builder. Please note that this licensing update does not impact any solutions that are already installed on the PowerFlow system, and you can continue to run and schedule existing content as needed. For more information, see ScienceLogic Pricing.

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 Synchronization PowerPacks.

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 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.

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"
```

> **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 **syncpack_steprunner** service, add the following lines:

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

> **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.

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 into 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.

To change the PowerFlow Administrator (*isadmin*) user password:

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.

   **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.

**NOTE:** If a PowerFlow user makes multiple incorrect login attempts, PowerFlow locks out the user. To unlock the user, run the following command: `unlock_user -u <username>`

# 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) container:

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

> **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 to the PowerFlow host.
2. 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"
```

3. Run the following script to validate and apply the change:

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

4. Re-deploy the gui service by running the following commands:

```
docker service rm iservices_gui
/opt/iservices/scripts/pull_start_iservices.sh
```

## 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**

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 PowerFlow for Scalability

SL1 PowerFlow version 2.2.0 introduces the **enable_state_backend** setting in the **environments** section of the **docker-compose.yml** file. This setting is set to **true** (enabled) by default. This setting provides a more consistent, scalable location for storing task metadata, enabling a more rapid query for task states, as well as preventing any task state logs from being ejected from the system and losing the state. The **enable_state_backend** setting does this by writing metadata for tasks to the Couchbase database.

This setting addresses the following scalability issues:

- Having to increase the redis memory limit to prevent early ejection of logs and task states (logs for a previously run PowerFlow application always return pending, even though the application already ran.
- Slow lookups when querying for large-scale integration results (slow load times for Device Sync).
- Having to increase the contentapi memory limit to prevent timeouts when querying for large sets of results.

Because PowerFlow version 2.2.0 and later now writes task metadata to the Couchbase database, the user experience of working with previously run integrations and data is improved. However, this new setting also results in a slightly increased performance impact on Couchbase.

If you are using a new deployment of PowerFlow version 2.2.0 or later, no action is needed; you can install version 2.2.0 and the **enable_state_backend** setting in the **environments** section is set to **true** (enabled) by default. You can then scale up as needed.

If you are using a version of PowerFlow before version 2.2.0, or if you are upgrading to version 2.2.0, ScienceLogic recommends that you assess the current utilization of your PowerFlow system, because there will be a greater impact on the Couchbase database by default. If you find the database, or database node, is running at max CPU (80% or greater), you should be prepared to increase CPUs allocated to the Couchbase nodes, or deploy an additional database node to accommodate for the additional load.

Alternatively, if the benefits of this feature are not desired, and you cannot afford to increase CPU in a large scale environment, you can disable it by setting **enable_state_backend: false** on both the contentapi and steprunner services in the **docker-compose.yml** file. Disabling this feature will no longer store metadata in the Couchbase database, which removes the additional overhead on Couchbase, but you will not experience the benefits listed above.

This release significantly improves performance related to task status querying, reducing the API memory requirements and increasing the responsiveness of the PowerFlow user interface. When a PowerFlow system queries for the state of a task, instead of pulling all relevant data from that task, the system only pulls the metadata that is needed at that time.

## Signs that the Database Needs More CPU

The following behaviors are indications that your database needs more CPU:

- The following error occurs in dexserver logs: "Storage health check failed: create auth request: insert auth request: operation has timed out" errors in dexserver logs"
- Frequent timeout errors in logs
- Unable to consistently authenticate and log into the PowerFlow user interface, such as one request succeeds, but the next five requests keep bringing you back to authentication
- API requests are failing with 500 errors
- Couchbase CPU utilization is frequently greater than 80%

To further ensure the database is not doing more work than necessary, ScienceLogic recommends the following actions to further offset this additional database impact:

- Make sure that all of your PowerFlow applications that are running are activated and installed from a Synchronization PowerPack. Content from Synchronization PowerPacks allows PowerFlow to read code directly from virtual environments and save queries to the database.
- Reduce or remove "replica" count settings for the logs bucket. While 2 replicas is important for Synchronization PowerPack content, as it is your configuration and integration data, it is not as important for logs. Replicating bucket data between nodes can be expensive, especially for a highly active bucket like logs. For this reason, ScienceLogic recommends you have a maximum of 1 replica for the logs bucket, or disable replicas on logs completely.

## Timeout and Retry Configuration Variables

There are three configuration variables you can use to configure the number of retries for saving the application logs, step logs, and the status of both in Couchbase. To improve the access to Couchbase when saving the application and step logs, you can edit the default values of the following configuration variables in the **docker-compose.yml** file:

- ***logs_timeout_retries***. Use this configuration variable as **env_var** in the **docker-compose.yml** file for the workers, if any other number of retries is needed. The default is 3.
- ***logs_delay_retries***. The delay of retries. The default is 2.
- ***state_timeout_retries***. Number of retries for Celery. This value can be changed using the **env_var** in the **docker-compose.yml** file for the workers, if any other number of retries is needed. The default is 3.

# 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:

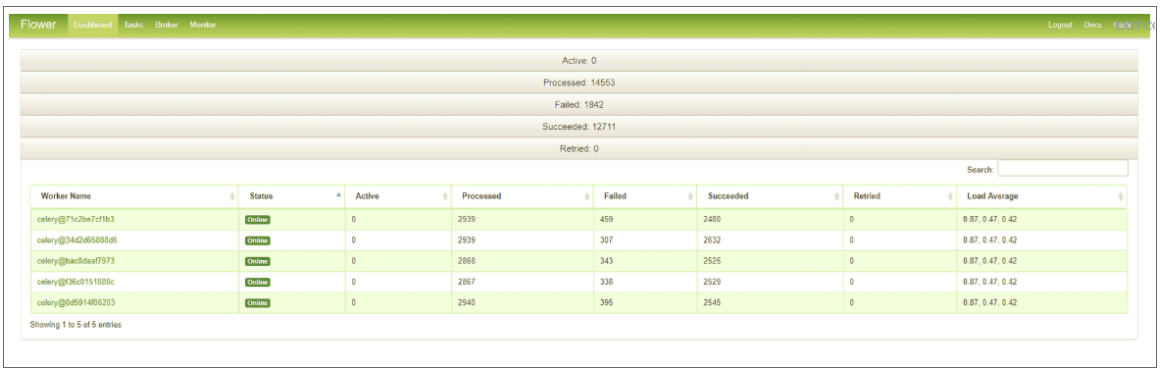
```
steprunner:
  image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
  environment:
    additional_worker_args: ' --max-memory-per-child 700000'
```

# 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. 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

To view this information in the Flower user interface navigate to `<hostname_of_PowerFlow_system>/flower`.

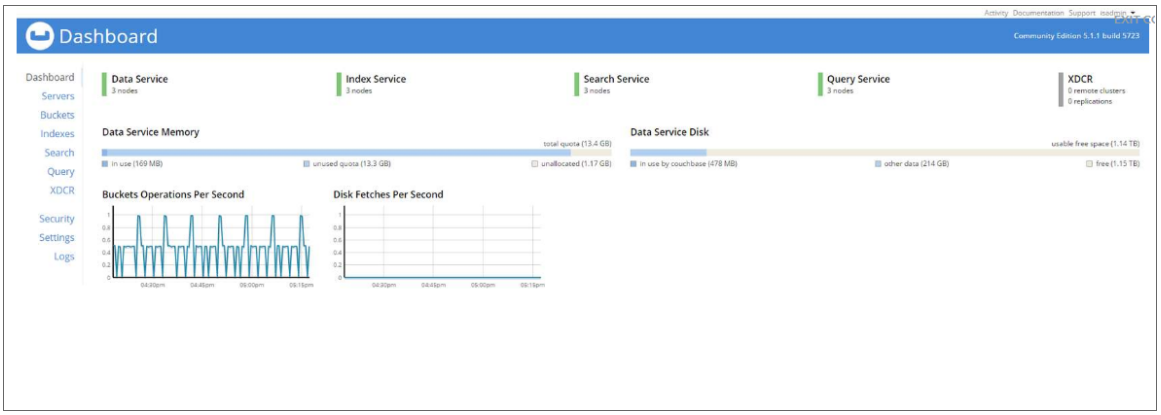
For more information, see the Flower API Reference.

**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:




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.

To view this information in the Couchbase Administrator user interface, navigate to `<hostname_of_PowerFlow_system>:8091`.

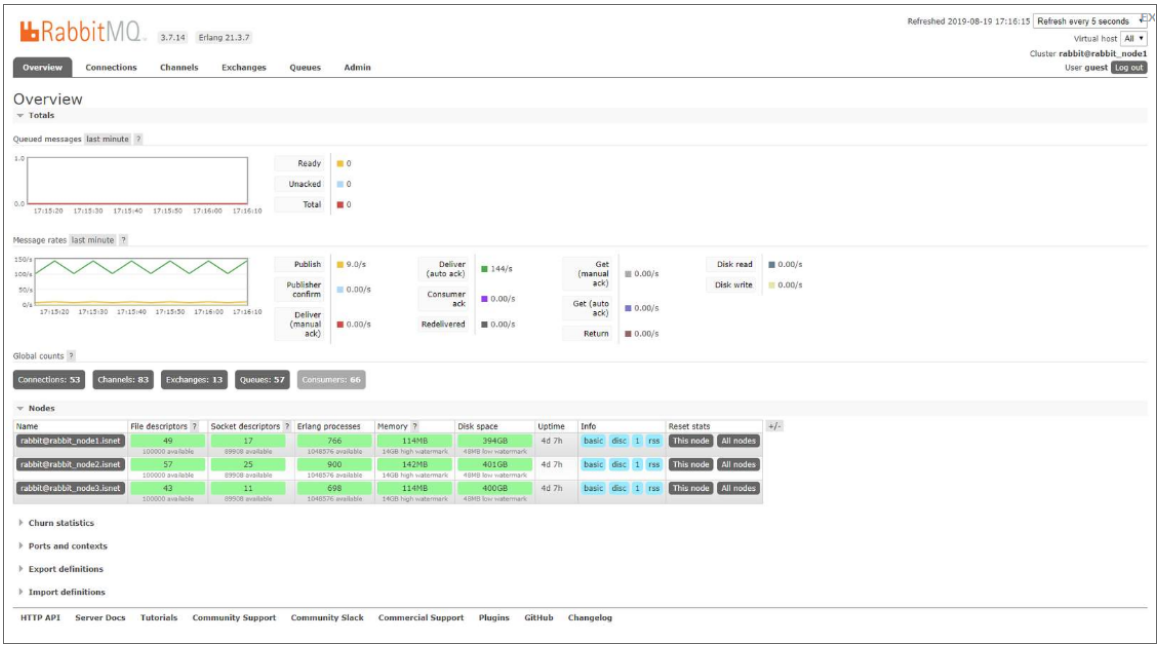
For more information, see the Couchbase API Reference.

**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:




## 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

# 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 (  ) 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

SL1 PowerFlow version 2.0.0 or later supports the following authentication methods:

- ***Local Authentication***. The same local Administrator user (*isadmin*) is supported by default with 2.0.0 or later installations. If you are migrating from a previous version of PowerFlow to version 2.0.0 or later, you can log in and authenticate with the same user and password. Local authentication only supports the isadmin administrator user.
- ***Basic Authentication***. PowerFlow 2.0.0 or later continues to support Basic Authentication as well. Because the PowerFlow PowerPacks, diagnostic scripts, and the iscli tool continue to use Basic Authentication, ScienceLogic does not recommend disabling Basic Authentication with PowerFlow version 2.0.0 or later.
- ***OAuth***. Lets PowerFlow administrators use their own authentication providers to enforce user authentication and lockout policies. Authentication using a third-party provider, such as LDAP or Active Directory, requires additional configuration. For optimal security, ScienceLogic recommends that you disable the local Administrator user (*isadmin*) and exclusively use your own authentication provider.

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*

## 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**. To change this password, run the **/opt/iservices/scripts/ispasswd** script.

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 in PowerFlow 2.0.0 or later. 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*.

**WARNING:** This Basic Authentication user is enabled by default to support backward compatibility with any scripts or queries running against the versions of PowerFlow before 2.0.0. If you disable this authentication strategy, PowerPacks and end-user scripts will not be able to query or access the API unless they are updated to use OAuth 2.0 with bearer a token.

## 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 or Active Directory (AD). 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 or SAML 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.

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 "insecureNoSSL",
       # 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 configuration.
```

## 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.

In PowerFlow 2.0.0 or later, 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 2.0.0 or later 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.

# 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.

### Assigning a Role to a Specific User

By applying a role permission to a single username in the admin panel, 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 in the admin panel, 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 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*.
- **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.
- **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.

# 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.

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 Synchronization PowerPacks, 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 definitions; and create Synchronization PowerPacks.
- ***Administrator***. The user has all of the privileges of the **Develop** permission, but the user can also add, install, activate, and delete Synchronization PowerPacks; 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

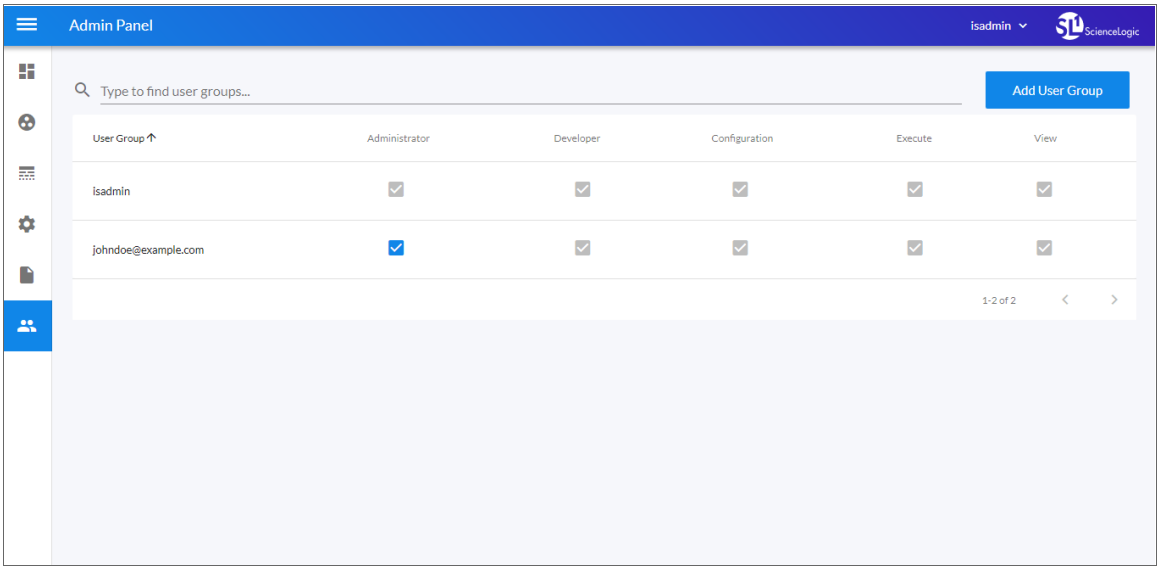
If you have the *Administrator* role, you can modify the permissions for each user group on the page. You cannot change the permissions for the user group to which you currently belong.

A user with the *Administrator* role can also create a user group and assign permissions to that group.

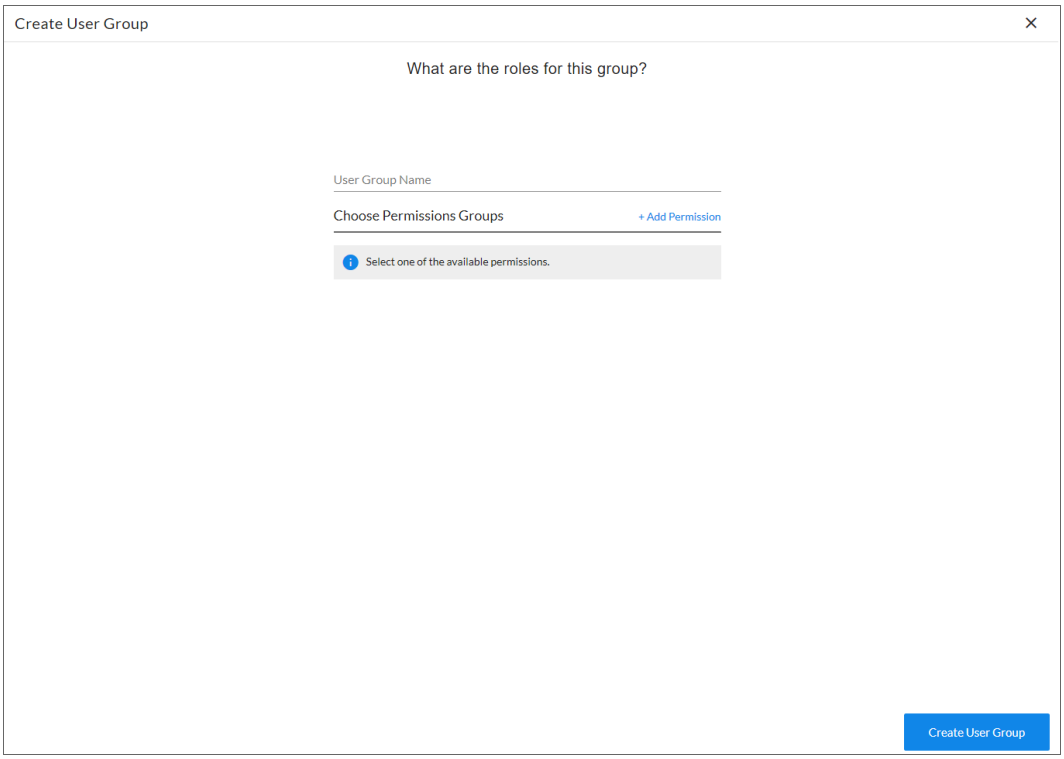
**NOTE:** The *only* password saved in the PowerFlow system is the password for the *isadmin* Administrator user. All other user passwords are saved in the third-party authentication provider configured for PowerFlow.

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 users in this group will use when logging in to PowerFlow.
   - ***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.

# Chapter 4

# Managing Synchronization PowerPacks

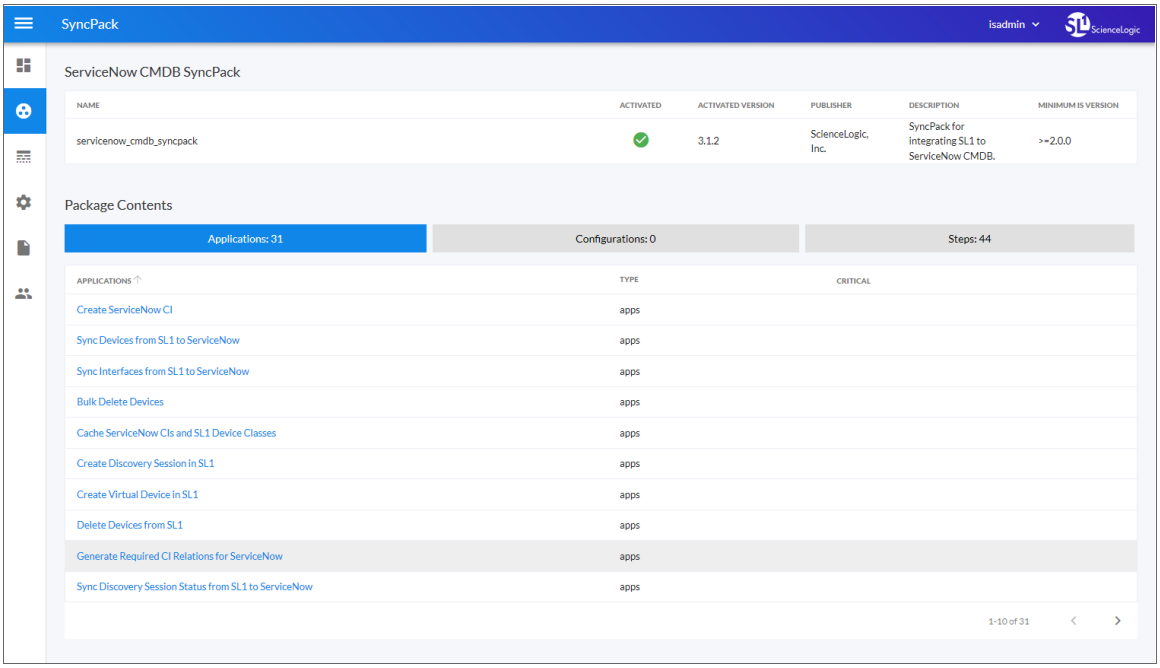
## Overview

This chapter describes how to use the **SyncPacks** page ( ) to import, install, and view Synchronization PowerPacks.

This chapter covers the following topics:

# What is a Synchronization PowerPack?

A Synchronization PowerPack contains all of the code and logic needed to perform integrations on the PowerFlow platform. A Synchronization PowerPack is saved as a Python **.whl** file, and it typically includes Python steps that are listed in applications, and then encapsulated within configurations:



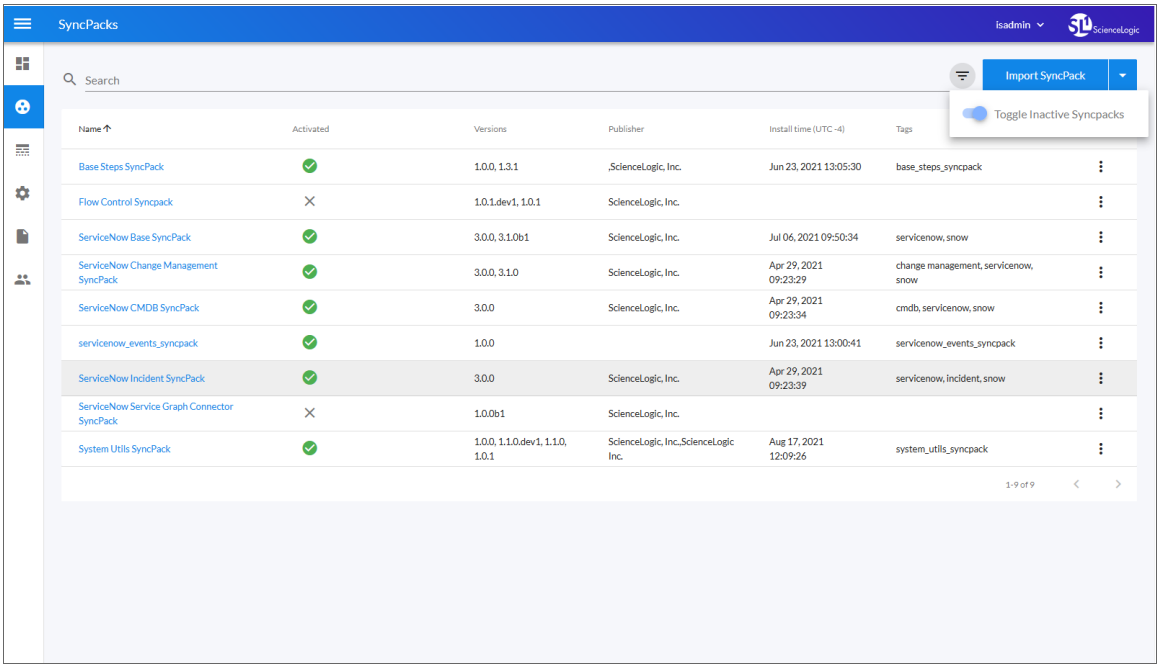

You can access the latest steps, applications, and configurations for PowerFlow or a third-party integration, such as ServiceNow, by downloading the most recent Synchronization PowerPack for that integration from ScienceLogic. You can access all Synchronization PowerPacks on the **SyncPacks** page.

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

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

# Viewing the List of Synchronization PowerPacks

The **SyncPacks** page provides a list of the Synchronization PowerPacks on your PowerFlow system. From this page you can import, install, and view Synchronization PowerPacks:



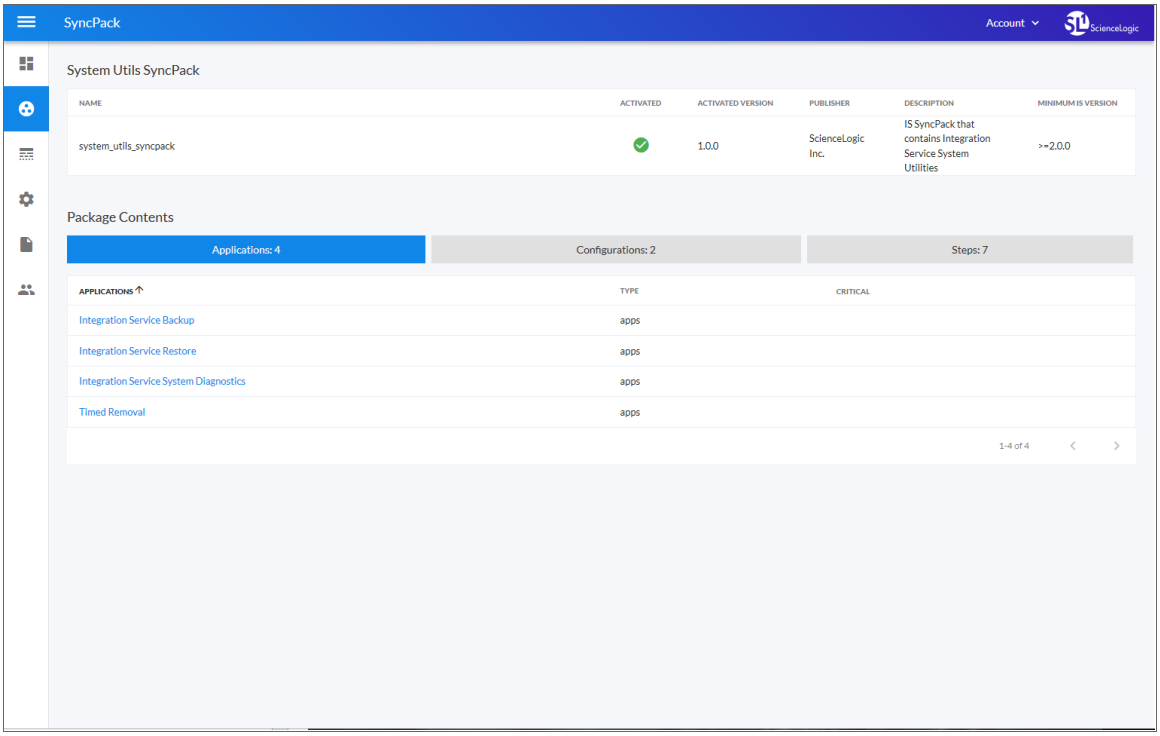

You can search for a specific Synchronization PowerPack by typing the name of that Synchronization PowerPack in the ***Search*** field at the top of the **SyncPacks** page. The user interface filters the list as you type. You can also sort the list of Synchronization PowerPacks by clicking the headers of the columns.

**TIP:** Click the Filter icon ( ) at the top right of the **SyncPacks** page and select *Toggle Inactive SyncPacks* to show all available Synchronization PowerPacks. An *activated* Synchronization PowerPack has been installed and is ready to be used.

The **[Actions]** button ( ) for a Synchronization PowerPack gives you the option to activate and install or uninstall that Synchronization PowerPack. Also, if you have older versions of the same Synchronization PowerPack, you can select *Change active version* to activate a different version other than the version that is currently running.

**TIP:** Hover over the version numbers in the **Version** column to see the "active" version of the Synchronization PowerPack currently being used by PowerFlow.

Click a Synchronization PowerPack from the list to view the **SyncPacks** detail page for the Synchronization PowerPack. From this page you can view the contents of the Synchronization PowerPack, including the applications, configuration objects, and steps included in the Synchronization PowerPack. You can also view additional metadata, including the active version, the minimum PowerFlow platform version, and other details:




# Importing and Installing a Synchronization PowerPack

The **SyncPacks** page in thePowerFlow user interface lets you import and install the latest version of a Synchronization PowerPack. After you install a Synchronization PowerPack, PowerFlow considers that Synchronization PowerPack to be ***activated*** and ready to be used. By default, the **SyncPacks** page displays all activated Synchronization PowerPacks.

After you install a Synchronization PowerPack, that Synchronization PowerPack is available on the **SyncPacks** page of the PowerFlow user interface. You can click the name of that Synchronization PowerPack to view the detail page for that Synchronization PowerPack. On the Synchronization PowerPack detail page, you can view metadata about the Synchronization PowerPack, and you can also view the applications, configuration objects, and steps for that Synchronization PowerPack. Also, the applications from that Synchronization PowerPack are added to the **Applications** page of the user interface.

You can click the Filter icon ( ) at the top of the **SyncPacks** page and select *Toggle Inactive SyncPacks* to view all versions of the available Synchronization PowerPacks. A Synchronization PowerPack must be a Python **.whl** file to upload and install in PowerFlow.

Also, you can click the dropdown arrow next to the **[Import SyncPack]** button to import or view dependencies for PowerFlow Synchronization PowerPacks.

**NOTE:** If a Synchronization PowerPack has a dependency on another specific Synchronization PowerPack version, you will need to install that Synchronization PowerPack version on the PowerFlow system before you can install the Synchronization PowerPack with the dependency.

**NOTE:** You must have the *Develop* or *Administrator* role to install a Synchronization PowerPack. 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 Synchronization PowerPacks. 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 Synchronization PowerPacks.

## Locating and Downloading a Synchronization PowerPack

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

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

To locate and download the Synchronization PowerPack:

1. Go to the ScienceLogic Support Site.
2. Click the **[Product Downloads]** tab and select *PowerPack*.
3. In the ***Search PowerPacks*** field, search for the Synchronization PowerPack and select it from the search results. The **Release Version** page appears.
4. On the **[PowerPack Versions]** tab, click the name of the Synchronization PowerPack 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 Synchronization PowerPack to start downloading the file.

**NOTE:** Synchronization PowerPacks do not require a specific license. After you download a Synchronization PowerPack, 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 Synchronization PowerPack in an offline deployment, see "Installing or Upgrading in an Offline Environment" in the Synchronization PowerPack release notes to ensure you install any external dependencies.

## Importing a Synchronization PowerPack

To import a Synchronization PowerPack 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 Synchronization PowerPack you want to install.

   **TIP:** You can also drag and drop a **.whl** file to the **SyncPacks** page.

3. Click **[Import]**. PowerFlow registers and uploads the Synchronization PowerPack. The Synchronization PowerPack is added to the **SyncPacks** page.

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

## Installing a Synchronization PowerPack

To install a Synchronization PowerPack in the PowerFlow user interface:

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

   **TIP:** By default, the **SyncPacks** page displays only activated and installed PowerPacks. 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 uninstalled PowerPacks.

2. Click **[Yes]** to confirm the activation and installation. When the Synchronization PowerPack is activated, the **SyncPacks** page displays a green check mark icon ( ) for that Synchronization PowerPack. If the activation or installation failed, then a red exclamation mark icon ( ) appears.

   > **TIP:** While the Synchronization PowerPack is installing, you cannot click any of the options that appear when you click the **[Actions]** button ( ).

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 Synchronization PowerPack. 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 Synchronization PowerPack on your PowerFlow system, you can click the **[Actions]** button ( ) for that Synchronization PowerPack and select *Change active version* to activate a different version other than the version that is currently running.

> **TIP:** The most common error that occurs when installing a Synchronization PowerPack is that the Synchronization PowerPack dependencies are not installed. Review the "System Requirements" section of the release notes for that Synchronization PowerPack to ensure that you have installed all of the required applications for that Synchronization PowerPack. To view a list of additional PowerFlow and SL1 products that are required by the various Synchronization PowerPacks, see the SL1 PowerFlow Dependency Matrix.

# Default Synchronization PowerPacks

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

- *Base Steps* Synchronization PowerPack
- *System Utils* Synchronization PowerPack

> **NOTE:** The *Flow Control* Synchronization PowerPack contains the "IfStep" step that enables the logical branching used by the PowerFlow builder. This Synchronization PowerPack is included in PowerFlow.

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

## Base Steps Synchronization PowerPack

PowerFlow Platform version 2.2.1 and later requires version 1.3.1 or later of the *Base Steps* Synchronization PowerPack. This version includes an update to the "Query REST" step that prevents issues with scheduled PowerFlow applications.

**TIP:** You can download the latest version of this Synchronization PowerPack from the PowerPacks page of the ScienceLogic Support Site.

The *Base Steps* Synchronization PowerPack contains a set of steps that are used in a variety of different Synchronization PowerPacks, including "Query REST", "Query GQL", and "MySQL Select" steps, which are used throughout PowerFlow for a variety of use cases.

This Synchronization PowerPack contains the "Template App" PowerFlow application. You can use the "Template App" application as a template for building new PowerFlow applications.

The *Base Steps* Synchronization PowerPack includes the following steps, which you can use in new and existing applications:

- Cache Delete
- Cache Read
- Cache Save
- Direct Cache Read
- Jinja Template Data Render
- MS-SQL Describe
- MS-SQL Insert
- MS-SQL Select
- MySQL Describe
- MySQL Insert
- MySQL Select
- Query GraphQL
- Query REST (updated in version 1.2.0, 1.3.0, and 1.3.1)
- QueryRest: BearerAuth (new in version 1.3.0)
- QueryRest: OAuth (new in version 1.2.0)
- Run a command through an SSH tunnel
- Trigger Application

> **TIP:** To view the code for a step, select a Synchronization PowerPack from the **SyncPacks** page, click the **[Steps]** tab, and select the step you want to view.

You can configure the existing "Query REST" step to use bearer authentication by adding the bearer token to the request headers in the **Configuration** pane for that step. The ***headers*** field should look like the following:

```
{
  "Authorization": "Bearer <token_id>",
  "accept": "application/json",
  "content-type": "application/json"
  }
```

## System Utils Synchronization PowerPack

The *System Utils* Synchronization PowerPack is a Standard Synchronization PowerPack that contains applications, a configuration object, and steps. Version 1.1.0 of this Synchronization PowerPack requires SL1 PowerFlow version 2.2.0 or later.

> **NOTE:** You must install the *Base Steps* Synchronization PowerPack before you can install this Synchronization PowerPack.

The *System Utils* Synchronization PowerPack version 1.1.0 includes the following PowerFlow applications:

- ***Integration Service Backup***. Creates a backup file of the Couchbase database used by PowerFlow and sends the file to a remote location. For more information, see *Backing up Data*.
- ***Integration Service Restore***. Restores a Couchbase backup file that was created up with the "PowerFlow Backup" application. For more information, see *Restoring a Backup*.
- ***Integration Service System Diagnostics***. Displays a report of platform diagnostics for the services used by PowerFlow. For more information, see *Viewing PowerFlow System Diagnostics*.
- ***Timed Removal***. Removes logs from Couchbase on a regular schedule. For more information, see *Removing Logs on a Regular Schedule*.

This Synchronization PowerPack includes the following configuration objects:

- ***IS - System Backup Configuration Example***. Contains the structure needed for the "PowerFlow Backup" and "PowerFlow Restore" applications.
- ***PF - System Diagnostic Configuration Example***. Contains the structure needed for the "PowerFlow System Diagnostics" application.

> **TIP:** You should do a ***Save As*** with the example configuration objects, above, to make a new configuration object that you can customize for your specific PowerFlow system. Do *not* use the example configuration objects to run PowerFlow applications.

This Synchronization PowerPack also includes the following steps, which are used in the four PowerFlow applications listed above:

- Collect IS diagnostics
- Create Integration Service Backup
- Diagnostic reporter
- Query an IS Node
- Query Application Logs and Remove
- Query IS Manager node
- Restore Couchbase From Backup

# Chapter 5

# Managing PowerFlow Applications

## Overview

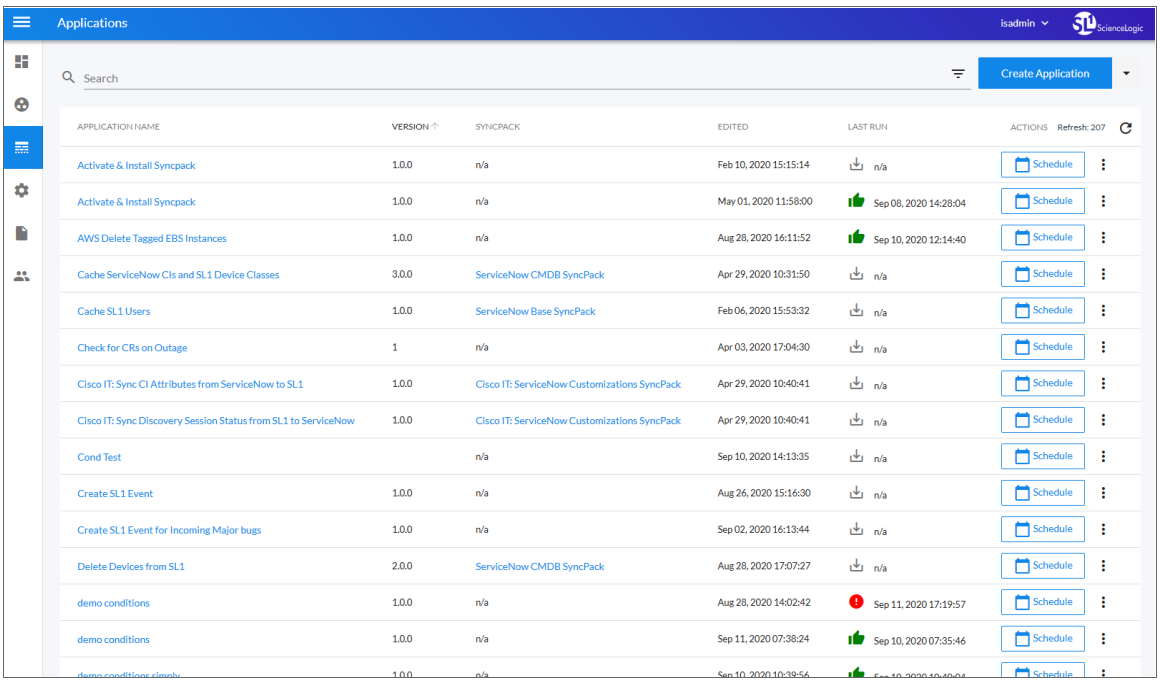
This chapter describes how to use the **Applications** page ( ) of the PowerFlow user interface to view, run, and schedule PowerFlow applications. You can use the PowerFlow builder to create custom applications, and those applications can use "flow control" operators that enable logical branching and data transformation between steps.

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

The **Applications** page ( ) provides a list of available PowerFlow applications on your PowerFlow system. From this page you can schedule, edit, view, and (for Premium solution users) create applications and steps:




You can search for a specific application by typing the name of that in the ***Search*** field at the top of the **Applications** page. The user interface filters the list as you type. You can also sort the list of applications by clicking the headers of the columns.

Select an application name to view the **Application** detail page.

Some of the applications on the **Applications** page of the PowerFlow user interface 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 Synchronization PowerPack or some of the Synchronization PowerPacks, click the Filter icon ( ) and select the Synchronization PowerPacks that contain the applications 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**.

To manually refresh the **Applications** page, click the refresh icon ( ) next to the "countdown timer". Otherwise, PowerFlow automatically refreshes the page after 300 seconds (five minutes).

The **SyncPack** column on the **Applications** page lists the name of the Synchronization PowerPack to which a specific application belongs, where relevant. You can click the name of the Synchronization PowerPack to which an application belongs to go to the Synchronization PowerPack page for that pack.

The **Last Run** column shows the current status of all of the PowerFlow applications:

| Icon | Status |
|---|---|
| | The application ran successfully. The date and time of the last run appears next to the icon. |
| | The application is currently running. An application can have one of the following statuses:<br>• ***Started***. The application is currently running.<br>• ***Pending***. The application has not run or will not run.<br>If you stop the application, the next tasks will not run. |
| | The application failed to run successfully. |
| | The application has not been run. |

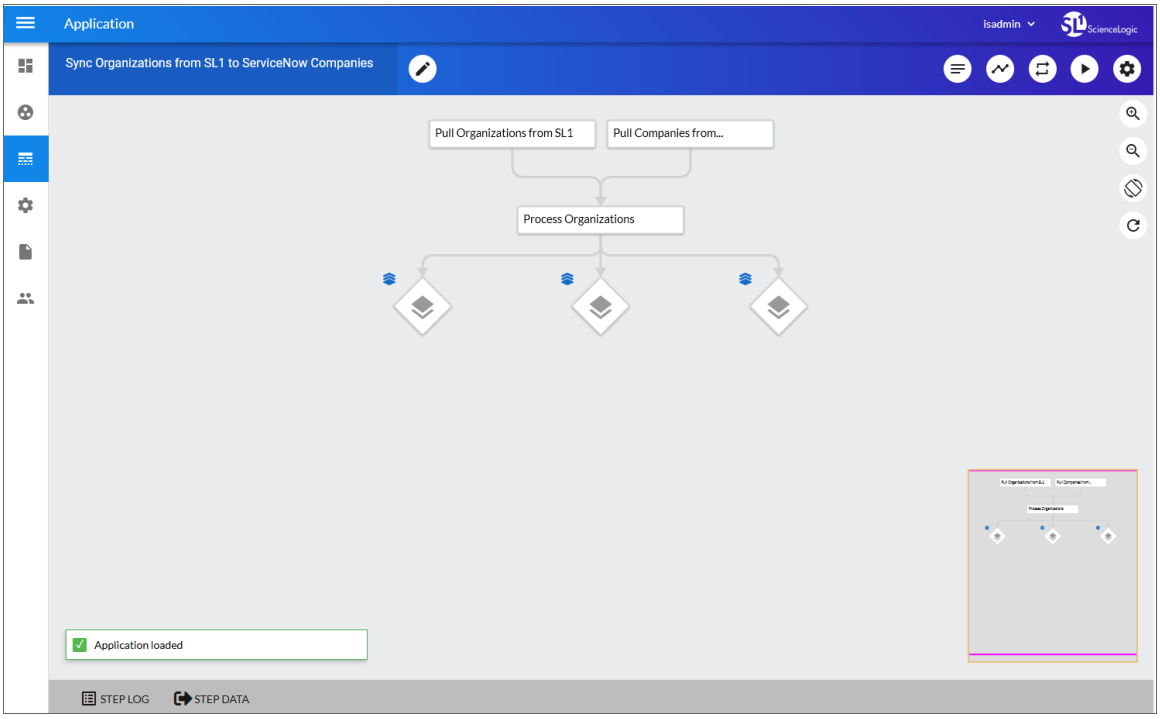
The **[Schedule]** button lets you use the Scheduler to define how often or at what time to run an application. A scheduled application displays a check mark in the **[Schedule]** button on this page. For more information, see *Scheduling a PowerFlow Application*.

The **[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.

> 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** 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".

## 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. After you click the button, the following buttons appear in the top navigation bar:
    - **[Close Editor]** (  ). Closes the **Steps Registry** pane.
    - **[Edit Metadata]** (  ). Opens the **Integration Metadata** window for that application so you can update the description or version of the application.
    - **[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.

  For more information, see ***Editing an Application***.

> **NOTE:** The PowerFlow builder is available only in SL1 Premium solutions. To upgrade, contact ScienceLogic Customer Support. For more information, see ScienceLogic Pricing.

- **[Reports]**(  ). Displays a report of the results of this application, where applicable. For more information, see ***Viewing a Report for an Application***.
- **[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:
    - *Debug 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.

  If this application does not support Debug or Custom Replays, this icon (  ) appears instead.

- **[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:
    - *Debug 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.

> **TIP:** After you start a run, you can click **[Stop]** (  ) to stop that application and end all running tasks for that application. The **[Stop]** button only appears while an application is running.

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 fields 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.

> **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". After a few seconds, the pop-up message disappears.

## 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.

## Application Thumbnail

In the bottom right-hand corner of the page is a smaller, thumbnail version of the application. You can click and drag on this version to move or scroll through the steps in the main viewing pane.

# Creating a Basic PowerFlow Application

On the **Applications** page, you can create new applications using the PowerFlow builder interface.

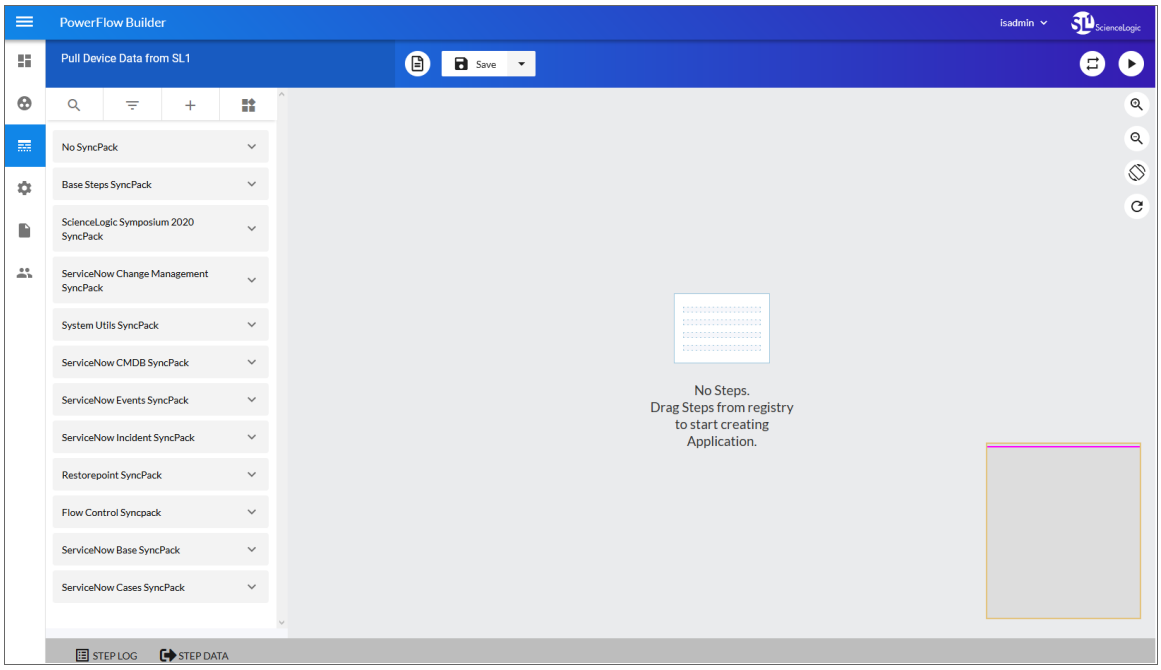
> **TIP:** To add a flow control operator to an application, such as a **Condition**, a **Transform**, or a **Trigger Application** operator, see *Working with Flow Control Operators*.

> **NOTE:** The PowerFlow builder is available only in SL1 Premium solutions. To upgrade, contact ScienceLogic Customer Support. For more information, see ScienceLogic Pricing.

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, 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 view the steps by Synchronization PowerPack or 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 Synchronization PowerPack for that step, the version, and creator of the step. You can also click **[Edit Step Code]** to edit the code for that step.

5. On the **Steps Registry** pane, click the step you want to add and drag it to the main viewing pane ("canvas"). The step is highlighted, and the **Parameters** pane for that step appears.
6. On the **Parameters** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed.

   **TIP:** 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. If needed, click the down arrow on the **Advanced** section to update the advanced fields.

> **TIP:** For example, if you are getting data from SL1, you would type the relevant SL1 information in the ***username*** and ***password*** fields in the Advanced section.

8. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** to run the step. Click the **[Step Log]** button if you want to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
9. On the **Parameters** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.

> **NOTE:** Clicking **[Save]** on the **Parameters** pane only saves the settings for this specific step; it does not save the new application.

10. Click **[Save]** (💾) to save your work in progress.
11. Repeat steps 4-9 to add more steps to the application.

> **TIP:** To connect one step to another, click and drag the mouse from the first step to the second step. An arrow appears between the two steps, which you can click and drag to reposition.

> **TIP:** 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, right-click the step and select *Delete*.

12. 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*.

13. On the **Application** detail page, add any additional steps and operators to the new application, and then click **[Save]** (💾) and the Close icon (⊗). 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* Synchronization PowerPack contains the "IfStep" step that enables the logical branching used by the PowerFlow builder. This Synchronization PowerPack is included in PowerFlow.

For more information about how to use PowerFlow builder to create low-code automated workflows that use "flow control" operators, watch the video at https://www.youtube.com/watch?v=isHImAGyQQc

## 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:

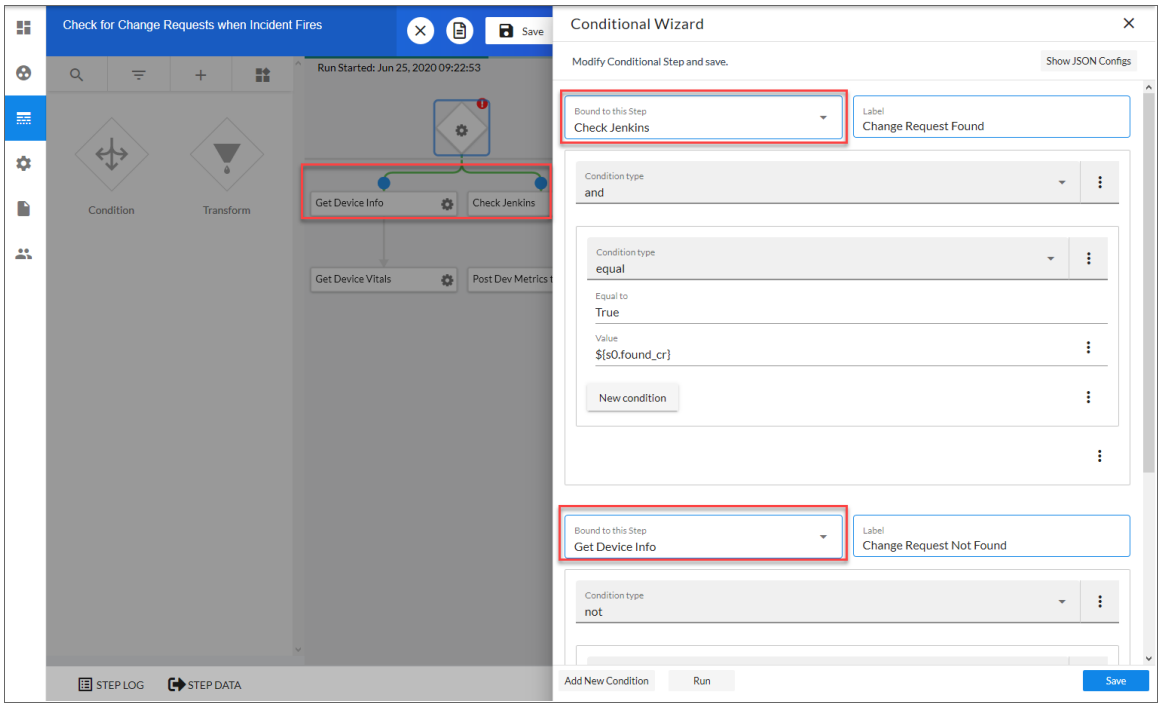
1. From the **Applications** page (), click **[Create Application]**. A **Create Application** window appears.
2. Complete the following fields:
   - ***Friendly Name***. The name that you want users to see for this application. Required.
   - ***Description***. A short description of what this application 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 **Parameters** pane for that step appears.
6. On the **Parameters** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed.
7. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
8. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** to run the step. Click the **[Step Log]** button if you want to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
9. On the **Parameters** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
10. Click **[Save]** ( ) to save your work in progress.
11. Repeat steps 4-9 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 **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.
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.

> **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.

17. Click **[Run]** ( ) to gather data again for all of the steps that will send data to the **Transform** operator.

18. Click the gear icon (⚙) on the **Condition** operator. The **Conditional Wizard** pane for that operator appears, this time with fields related to the steps you connected to the **Condition** operator:




19. Click **[Save]** on the **Conditional Wizard** pane to save your work so far.
20. Click **[Add New Condition]** for each new condition you want to configure.
21. Complete the following fields on the **Conditional Wizard** pane for each step you want to include in the branching:
    - ***Bound to this Step***. Change 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:
    - *equal*. In the ***Equal to*** field, type *True* or *False*. In the ***Value*** field, specify the variable that should be true or false.
    - *numeric*. In the ***Value*** field, specify the variable that should fall within the range specified by the ***Above*** and ***Below*** fields .
    - *template*. In the ***Value*** field, type a condition that can include numeric or another kind or comparison. For example: `${step_name.stop_step} == true`
    - *and*. Joins two or more conditions with the AND logic operator, and these conditions can be simple or complex conditions.
    - *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.

22. 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.
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 **[Save]** (💾) and the Close icon (⊗). 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 **Parameters** pane for that step appears.
6. On the **Parameters** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed.
7. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
8. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** to run the step. Click the **[Step Log]** button if you want to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
9. On the **Parameters** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
10. Click **[Save]** ( ) to save your work in progress.
11. Repeat steps 4-9 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.
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.

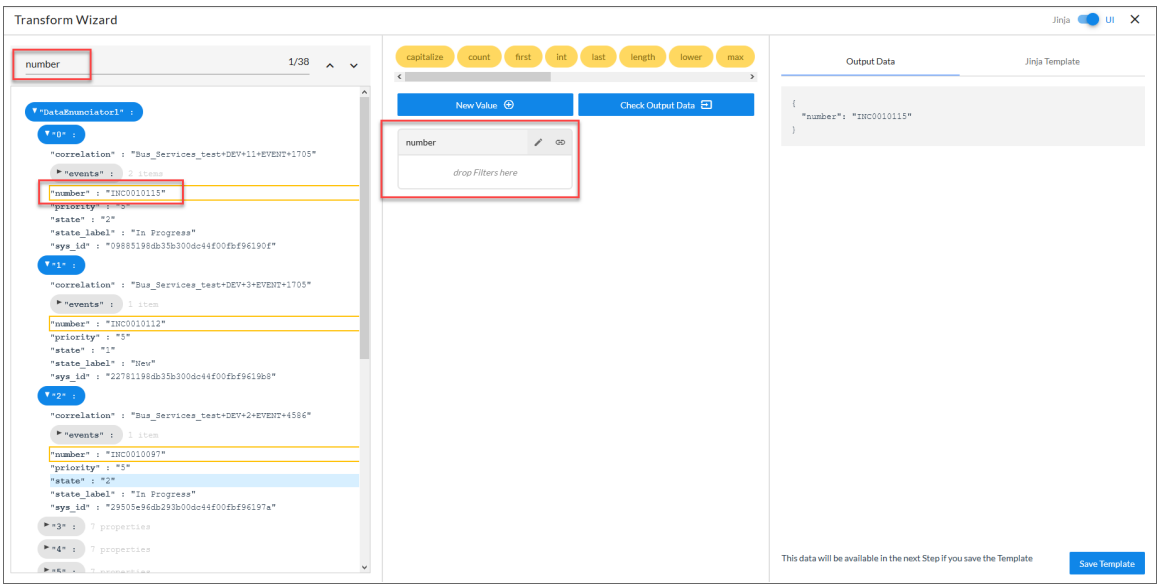
> **TIP:** The arrows connecting the steps and the **Transform** operator display as green to show the flow of the data, based on the values in the application.

18. Click **[Save]** ( )and then click **[Run]** ( ) to gather data from the step or steps that will send data to the **Transform** operator.

19. Click the gear icon (⚙) on the **Transform** operator. The **Transform Wizard** page appears again.
20. As needed, click the ***Jinja/UI*** toggle 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 22 to 27.
    - If you are using the ***Jinja*** code-only view, see steps 28 to 29.

> **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:

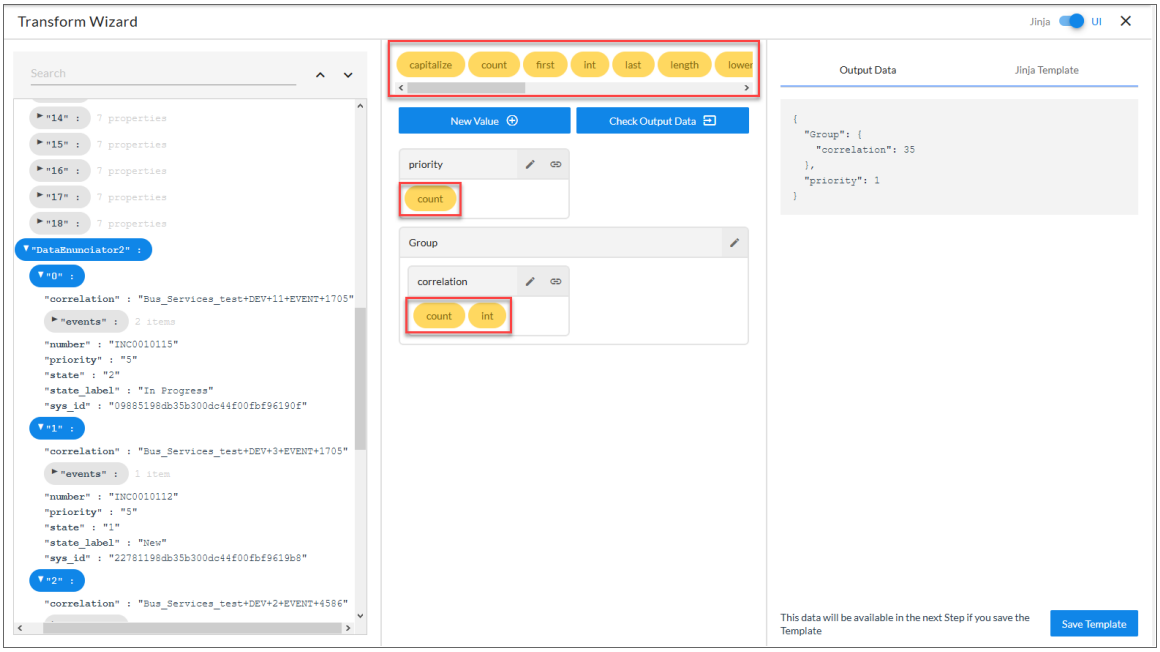



You can drag and drop add multiple values from the first pane onto the middle pane. PowerFlow uses these values to create the Jinja template for this **Transform** step. You can also drag and drop a group of values onto the middle pane by clicking the blue oval at the top of the list of values.

> **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 **[New Value]** and typing 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 31.
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 **[Save Template]**.
30. Close the **Transform Wizard** page. The **Application** detail page appears again.
31. Click **[Save]** ( ) to save your work.

> **TIP:** Click **[Run]** ( ) 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 **[Save]** ( ) and the Close icon ( ). The application is added to the **Applications** page.

## Creating an Application that Uses a Trigger Application Operator

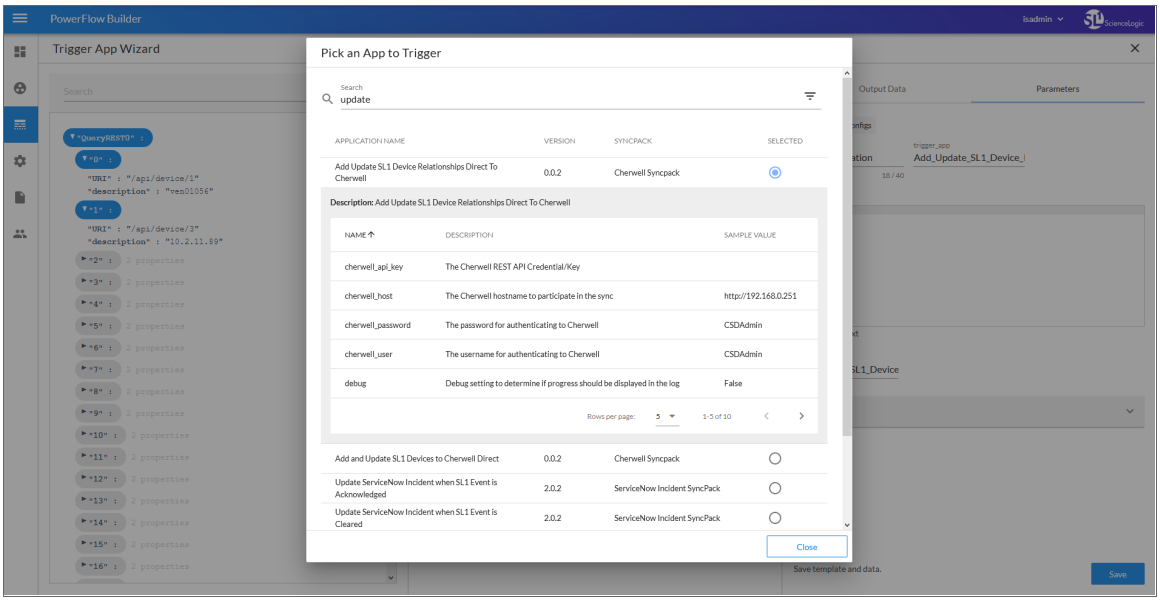
In the PowerFlow builder interface, you can use the **Trigger Application** operator ( ) to launch one or more PowerFlow applications from within a new or existing PowerFlow application. This operator uses the same functionality as the "Trigger Application" step from the Base Steps Synchronization PowerPack.

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 **Parameters** pane for that step appears.
6. On the **Parameters** pane, type a new name for the step and update the other fields on the **Configuration** pane as needed.
7. If needed, click the down arrow on the **Advanced** section to update the advanced fields.
8. To verify that the parameters you specified are correct and the step is configured correctly, click **[Run]** to run the step. Click the **[Step Log]** button if you want to view the results of the run on the **[Step Log]** and **[Step Data]** tab.
9. On the **Parameters** pane, click **[Save]** to save the parameters for the new step. The **Application** detail page appears again.
10. Click **[Save]** ( ) to save your work in progress.
11. Repeat steps 4-9 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 **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.
16. Connect the 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 **[Save]** ( ).
18. Click **[Run]** ( ) to run the new application and gather data for the **Trigger Application** operator.

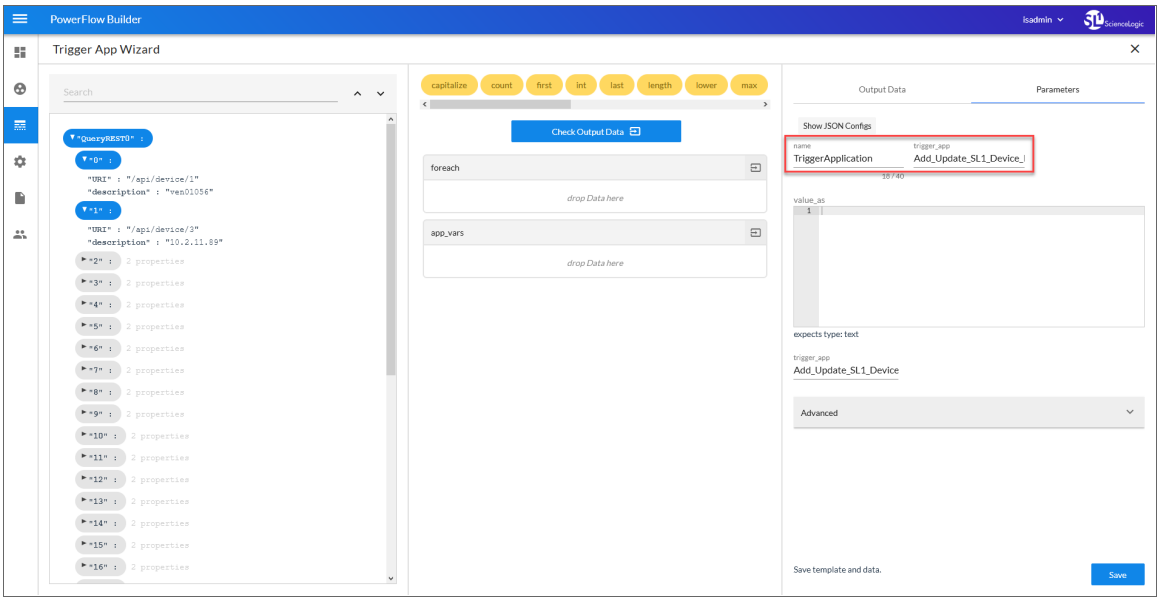
19. Click the gear icon (⚙) on the **Trigger Application** operator. The **Pick an App to Trigger** window for the **Trigger App Wizard** page appears:




20. Select the PowerFlow application that you want to trigger with this step.

> **NOTE:** When you select the application, a table appears below the application name. This table contains a description of the application you want to trigger, along with the required application variables from the **Configuration** pane for that application.

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 **[Save]** ( ) to save your work.

> **TIP:** Click **[Run]** ( ) 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 **[Save]** ( ) and the Close icon ( ). 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 "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.<br>• 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.<br>• Set to "false" if you only want to trigger one instance of the triggered application. |

| Parameter Name | Default Value | Description |
|---|---|---|
| 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 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.

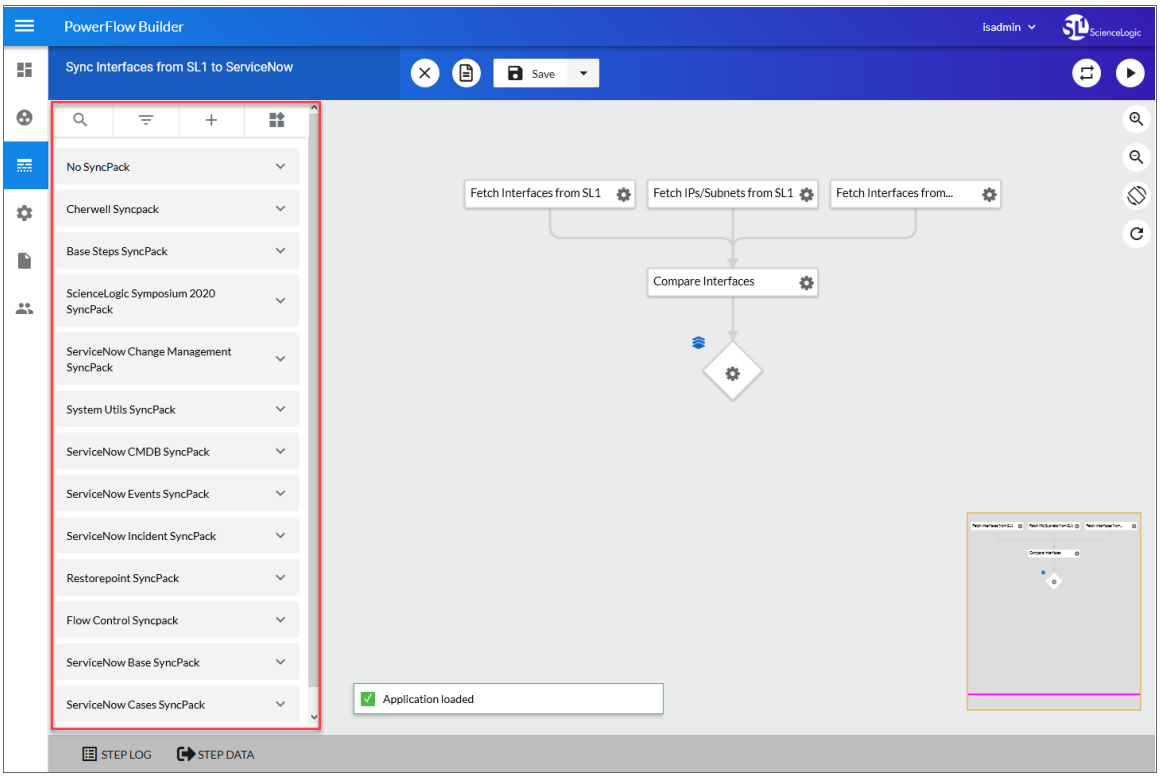
# Editing a PowerFlow Application

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 Synchronization PowerPack, 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 Synchronization PowerPack for that step, the version, and creator of the step. You can also click **[Edit Step Code]** to edit the code for that step.

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.
5. 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 gear icon ( ) on the step and update the fields as needed on the **Configuration** pane.

> **TIP:** You can also right-click the gear icon ( ) on a step to edit the step or 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 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.
12. To stop editing and close the **Search Steps Registry** pane, click the **[Close Editor]** button ( ).

# Creating a Step

On the **Applications** page, you can create new steps using Python that you can add to new or existing PowerFlow applications.

> **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 **[Create Application]** and select *Create Step*. A **Create Step** window appears.

   > **TIP:** You can also create a step from an **Application** page by clicking the **[Open Editor]** button (  ) and then clicking **[Create Step]** (+) 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. Click **[Save]**. 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

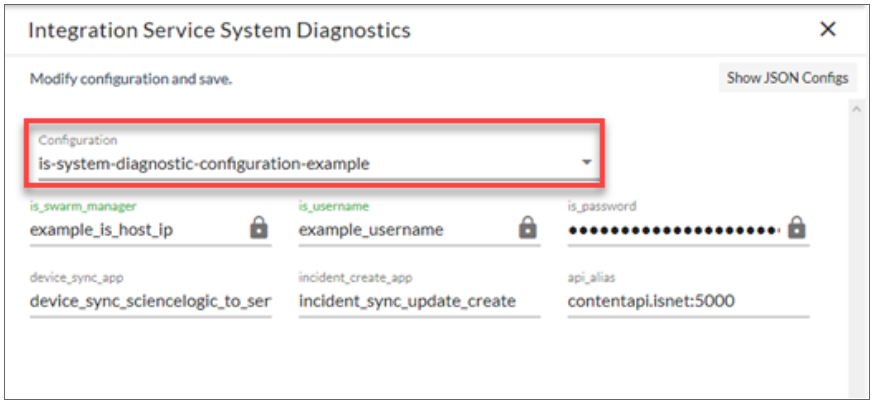
Before you can run a PowerFlow application, you must align the application with a configuration object from the **Configurations** page. A ***configuration object*** defines global variables, such as endpoints and credentials, that can be used by multiple steps and applications. Each variable in a configuration 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 **[Configure]** ( ). The **Configuration** pane opens on the right side of the **Application** page:




> **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 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. Click **[Show JSON Configs]** to view the JSON configuration data for the configuration object. Click **[Hide JSON Editor]** again to view the fields instead.
5. As needed, edit the other configuration values for the application. Press **[Enter]** after editing an item to make sure your changes are saved.

> **NOTE:** To prevent potential issues with security and configuration, any fields that are encrypted and any configuration-specific fields containing a padlock icon ( ) on the **Configuration** pane cannot be edited.

6. Click **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.

# 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:

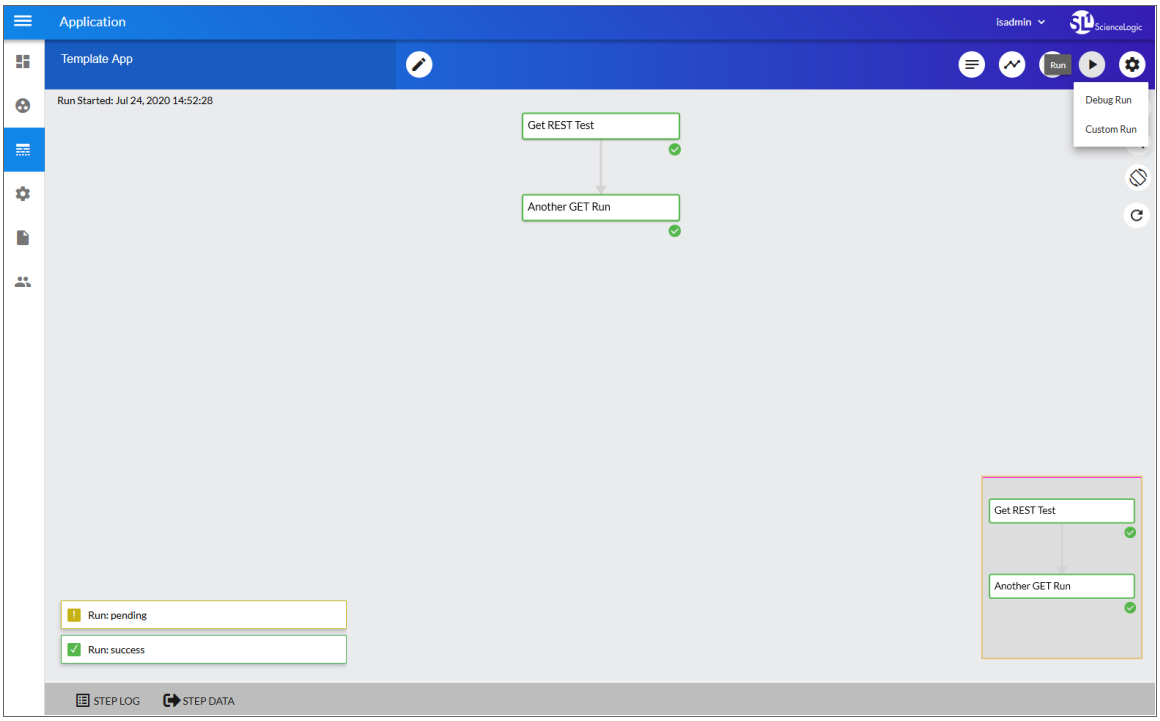
- ***Run***. Executes the application normally, with a log level of 1. This is the default, and it is the same as the *Run Now* option from the **Applications** page.
- ***Debug Run***. Executes the application in Debug Mode with a log level of 10.
- ***Custom Run***. Executes the application using a 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.

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 **[Run]** () 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]** ( ) 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** page for a selected application contains a **[Timeline]** button (  ) that displays a history of previous runs of that application.

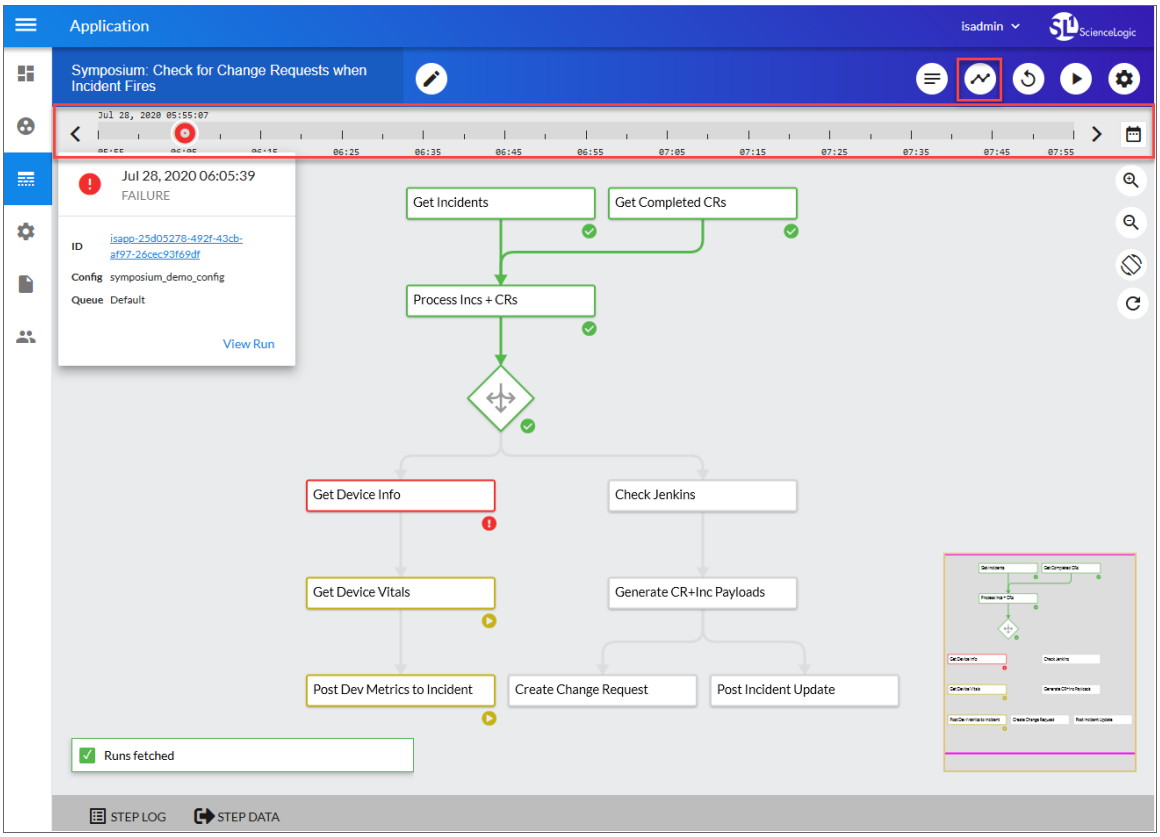
Also, you can click **[Replay]** (  ) to replay the last run of an application, such as when an application failed. You can also choose from the following options

- *Debug 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.

If this application does not support Debug or Custom Replays, this icon appears instead: (  ).

To view and filter the Timeline:

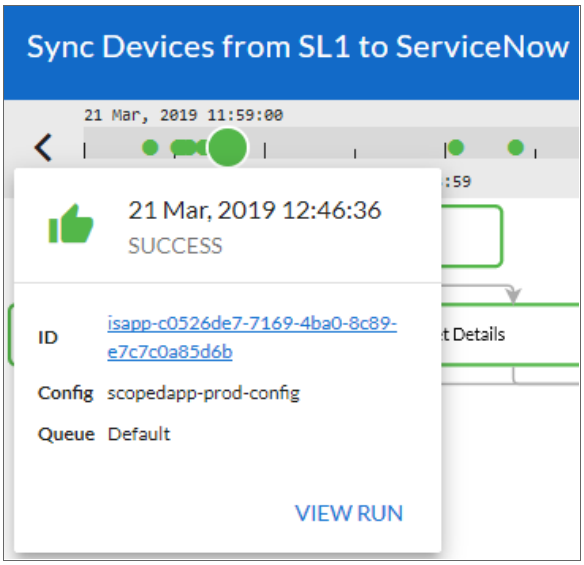
1. From an **Application** page, click **[Timeline]** ( ). The Timeline displays above the steps for that application:




2. The default view for the Timeline shows the last two hours of runs for that application. The image above shows the last four hours of runs. Use the left arrow icon ( ) and the right arrow icon ( ) to move through the Timeline in 15-minute increments:

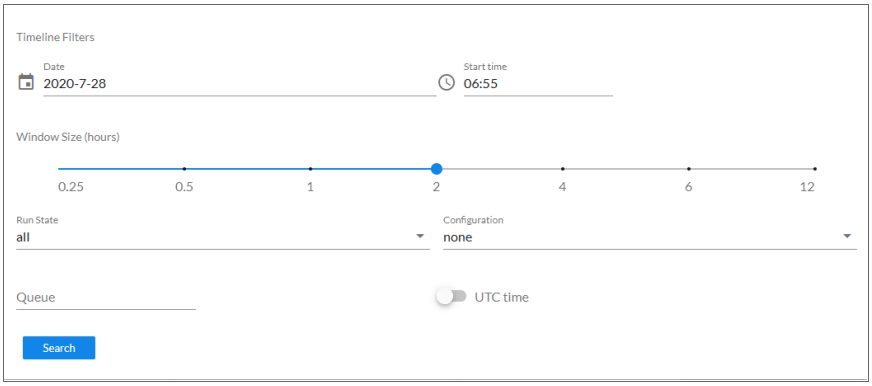

**NOTE:** The Timeline displays colored dots at a specific time that represent the last time this application was run. A green icon means a run was successful, and a red icon means that a run failed.

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 and the configuration and 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.

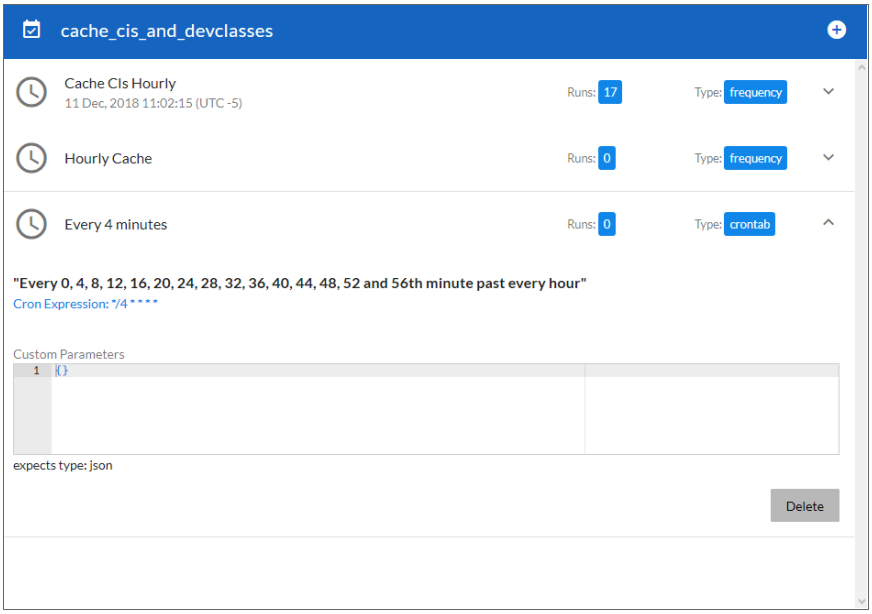
> **TIP:** If the Timeline is open and you want to close it, click the **[Timeline ]**button (  ).

# 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.

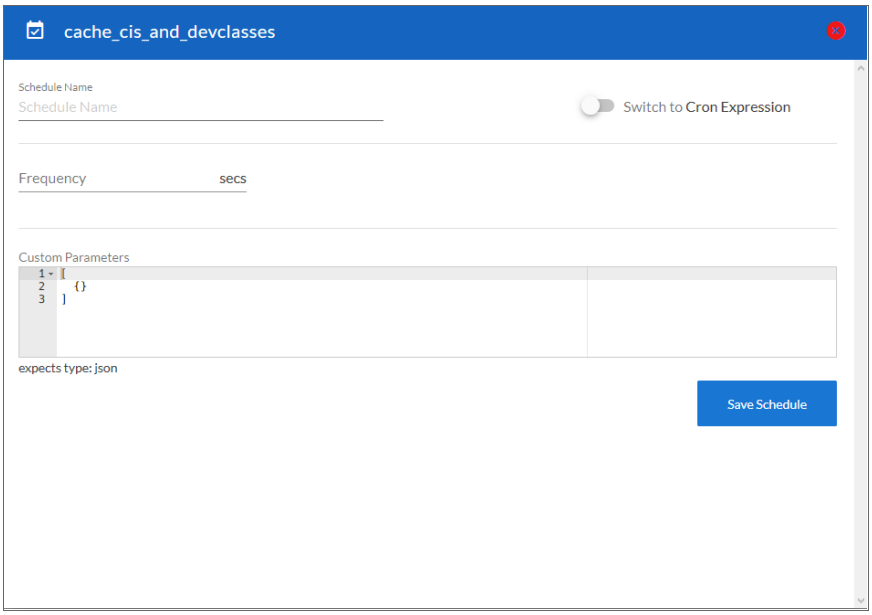
To schedule an application:

1. On the **Applications** page (), click the **[Schedule]** button for the application you want to schedule. The **Schedule** window appears, displaying any existing schedules for that application:




> **NOTE:** If you set up a schedule using a cron expression, the details of that schedule display in a more readable format in this list. For example, if you set up a cron expression of ***/4 * * * ***, the schedule on this window includes the cron expression along with an explanation of that expression: "*Every 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, and 56th minute past every hour*".

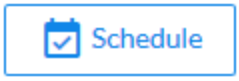
2. Select a schedule from the list to view the details for that schedule.
3. Click the + icon to create a schedule. A blank **Schedule** window appears:




4. In the **Schedule** window, complete the following fields:

   - ***Schedule Name***. Type a name for the schedule.
   - ***Switch to***. Use this toggle to switch between a cron expression and setting the frequency in seconds.
     - ***Cron expression***. Select this option to schedule the application using a cron expression. 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 *Expression: "Every 0 and 30th minute past every hour on the 1 and 31st of every month"*, based on ***/30 * * /30 * ***.
     - ***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.

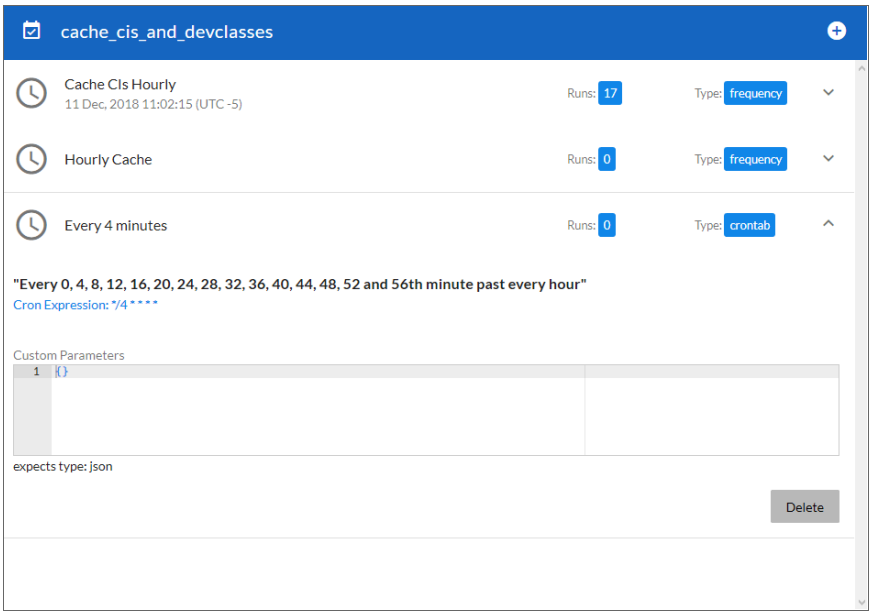
5. Click **[Save Schedule]**. The schedule is added to the list of schedules on the initial **Schedule** window. Also, on the **Applications** page, the word "Scheduled" appears in the **Scheduled** column for this application, and the **[Schedule]** button contains a check mark:



**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:

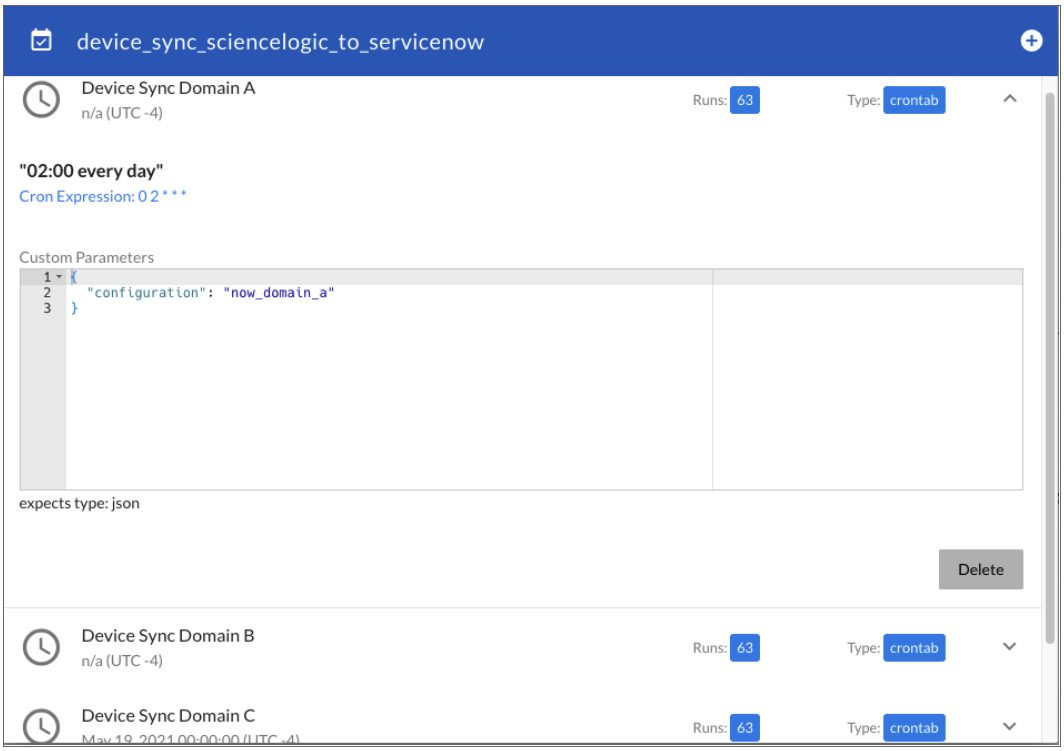
1. On the **Applications** page, click the **[Schedule]** button for the application that contains a schedule you want to delete. The **Schedule** window appears.
2. Click the down arrow icon ( ) to view the details of an existing schedule:




3. To delete the selected schedule, click **[Delete]**. The schedule is removed.

**NOTE:** When either multiple SL1 stacks or multiple ServiceNow systems are involved with PowerFlow, you should create an individual configuration object for each SL1 stack or ServiceNow system. Next, create an individual schedule for each configuration object. Each schedule should use a configuration object that is specific to that single SL1 stack or ServiceNow system. Creating copies of a PowerFlow application from a Synchronization PowerPack for the purpose of distinguishing between domains is not supported, and will result in issues on upgrades.

The following image shows how you can schedule PowerFlow applications for multiple ServiceNow domains:




# Viewing PowerFlow System Diagnostics

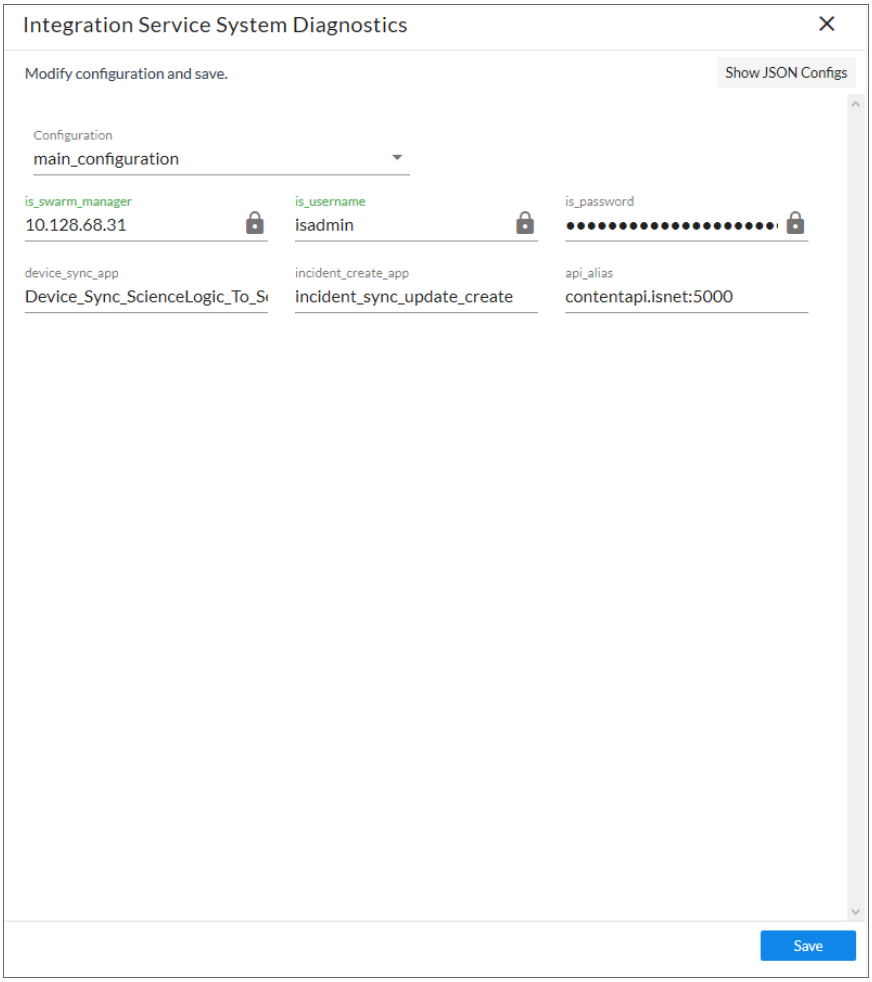
The "PowerFlow System Diagnostics" application lets you view platform diagnostics for the PowerFlow system. You can use the information displayed in these diagnostics to help you troubleshoot issues with the different tools used by PowerFlow.

**NOTE:** Older versions of this application are named "Integration Service System Diagnostics".

To view the diagnostics report:

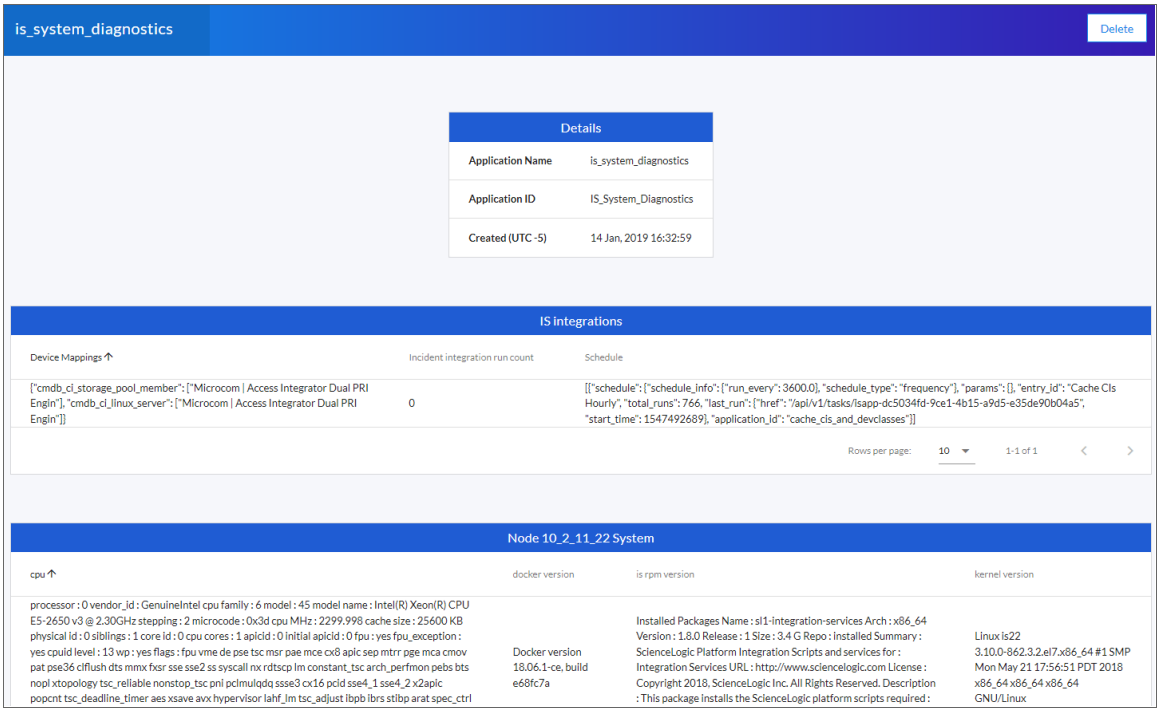
1. From the **Applications** page, select the "PowerFlow System Diagnostics" application. The **Application** page appears.

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***.
   - ***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.
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.

# Backing up and Restoring PowerFlow Data

You can use PowerFlow to back up and recover data in the Couchbase database.

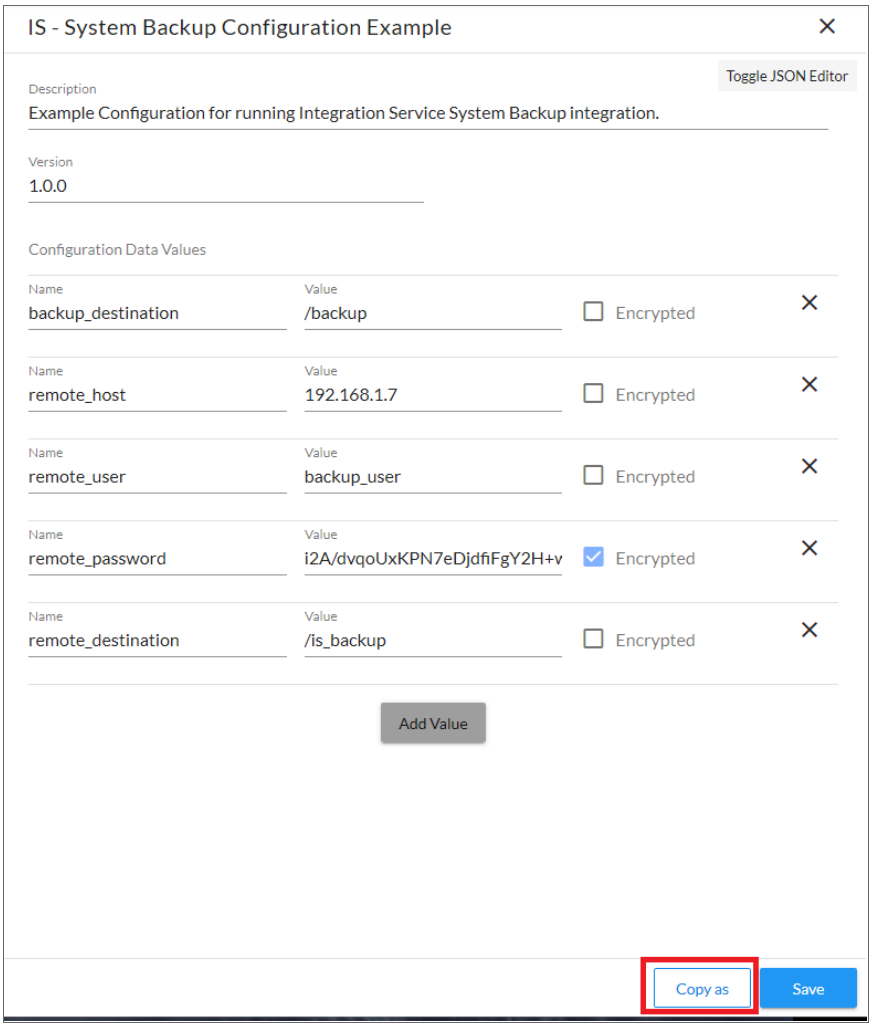
This option uses the "PowerFlow Backup" application in the PowerFlow user interface to create a backup file and send that file using secure copy protocol (SCP) to a destination system. You can then use the "PowerFlow Restore" application to get a backup file from the remote system and restore its content.

**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.

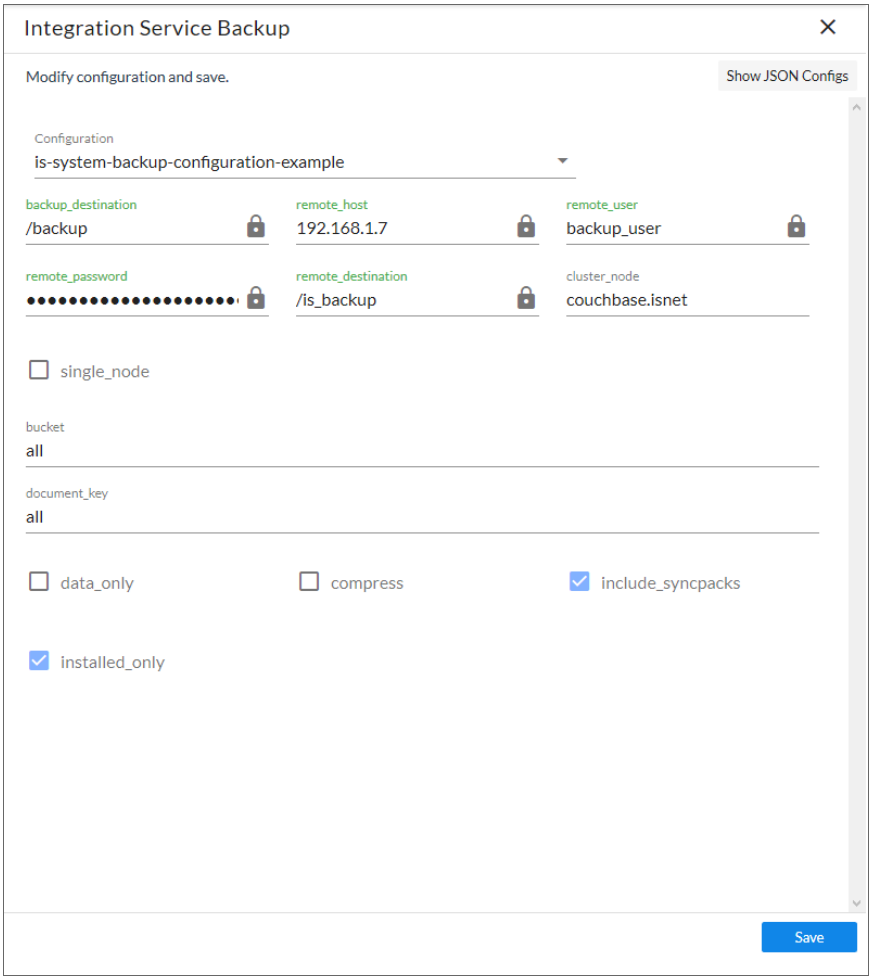
## Creating a Backup

To create a backup:

1. To add the relevant configuration information, go to the **Configurations** page ( ), click the **[Actions]** button ( ) for the "IS - System Backup Configuration Example", and select *Edit*. The **Configuration** pane for that configuration object appears:



2. Click **[Copy as]** and provide values for the following fields:
    - ***backup_destination***. The location where the backup file is created.
    - ***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.
3. Click **[Save]**.
4. Go to the **Applications** page and select the "PowerFlow Backup" application. The **Application** page appears.
5. Click **[Configure]** (  ). The **Configuration** pane appears:

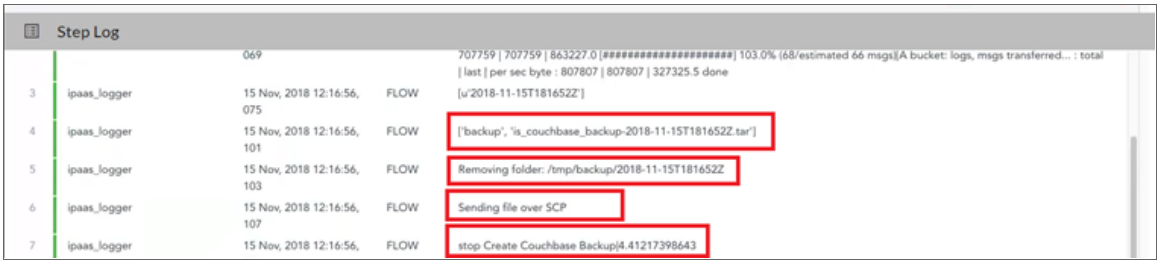


6. In the configuration file, provide values for the following fields:

   - ***Configuration***. Select a 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).
   - ***single_node***. Select this option if you want to back up from a single node in the cluster. Specify the node in the ***cluster_node*** field.
   - ***bucket***. Select which bucket in Couchbase you want to back up. Your options include all buckets, content-only buckets, or logs. The default is *all*.
   - ***document_key***: Select whether you want to back up all record or CI and device cache records.
   - ***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 Synchronization PowerPacks on the PowerFlow system.
   - ***installed_only***. Select this option if you want to back up only Synchronization PowerPacks that have been installed. If you do not select this option, the application will also back up Synchronization PowerPacks that have been uploaded to the PowerFlow system.

   > 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 **[Save]** and wait for the "App & Config modifications saved" pop-up message to appear. The **Configuration** pane automatically closes after this message appears.
8. On the **Application** page, click **[Run Now]** (  ). 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*.

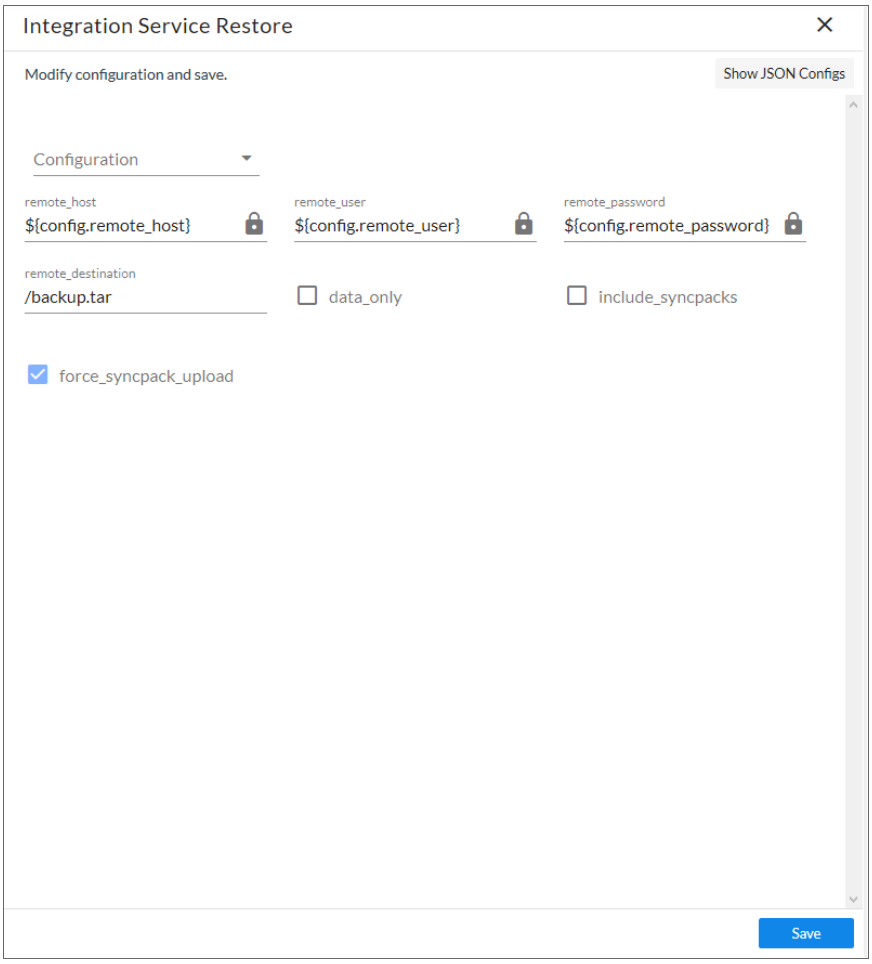
## Restoring a Backup

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:

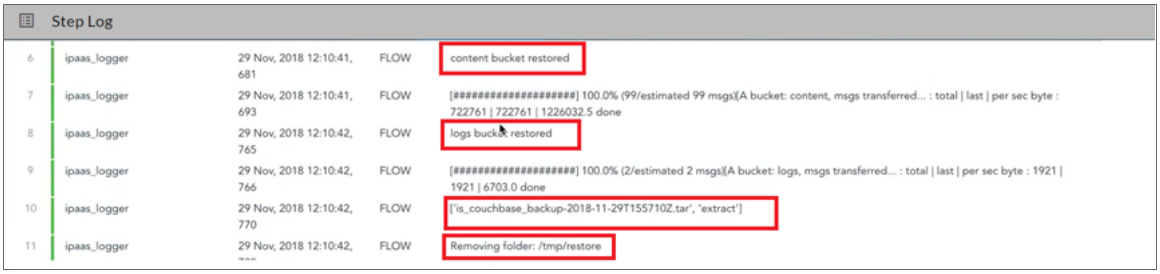
1. In the PowerFlow user interface, go to the **Applications** page and select the **PowerFlow Restore** application. The **Application** page appears.
2. Click **[Configure]** ( ). The **Configuration** pane appears:




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.
   - ***remote_destination***. Type the name of the backup file created by the "PowerFlow Backup" application.
   - ***data_only***. Select this option if you only want to restore bucket data.
   - ***include_syncpacks***. Select this option if you want to restore the Synchronization PowerPacks you backed up with the "PowerFlow Backup" application.
   - ***force_syncpack_upload***. Select this option if you want to force upload Synchronization PowerPacks if the files already exist in the PowerFlow 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 Now]** ( ).
6. To ensure that the backup was restored, click to open the **Step Log** section and look for entries related to restoring the backup:

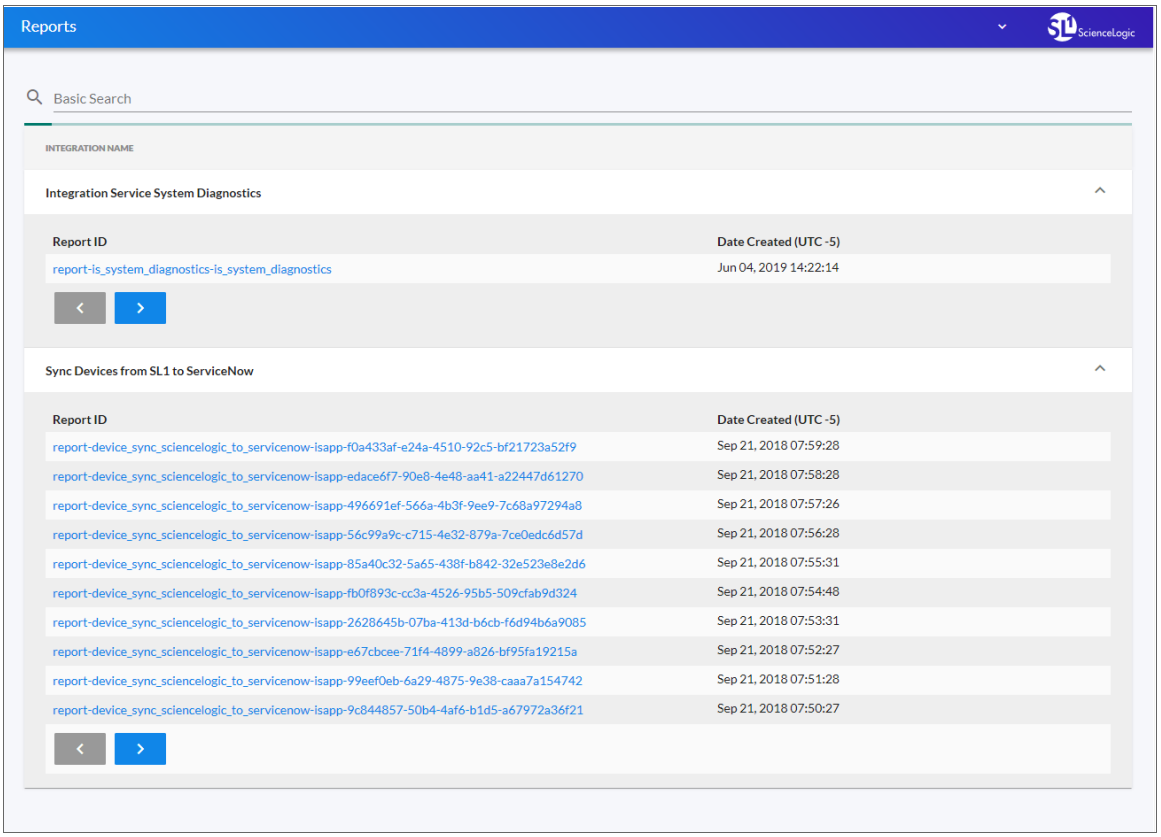



**NOTE:** After running the "PowerFlow Restore" application, the "PowerFlow Backup" application might display as "Run status pending". This issue occurs because at the time of the last backup from Couchbase, the logs for the "PowerFlow Backup" application showed a pending state. This message is addressed during the next run, and it does not cause any issues with the backup or restore processes.

# Viewing a Report for a PowerFlow Application

In the PowerFlow user interface, the **Reports** page ( ) contains a list of reports associated with applications. If an application has the reporting feature enabled and the application supports reports, PowerFlow will generate a report each time you run the application.

An individual report displays data only from the most recent run of the application; a report is not an aggregation of all previous runs.




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.

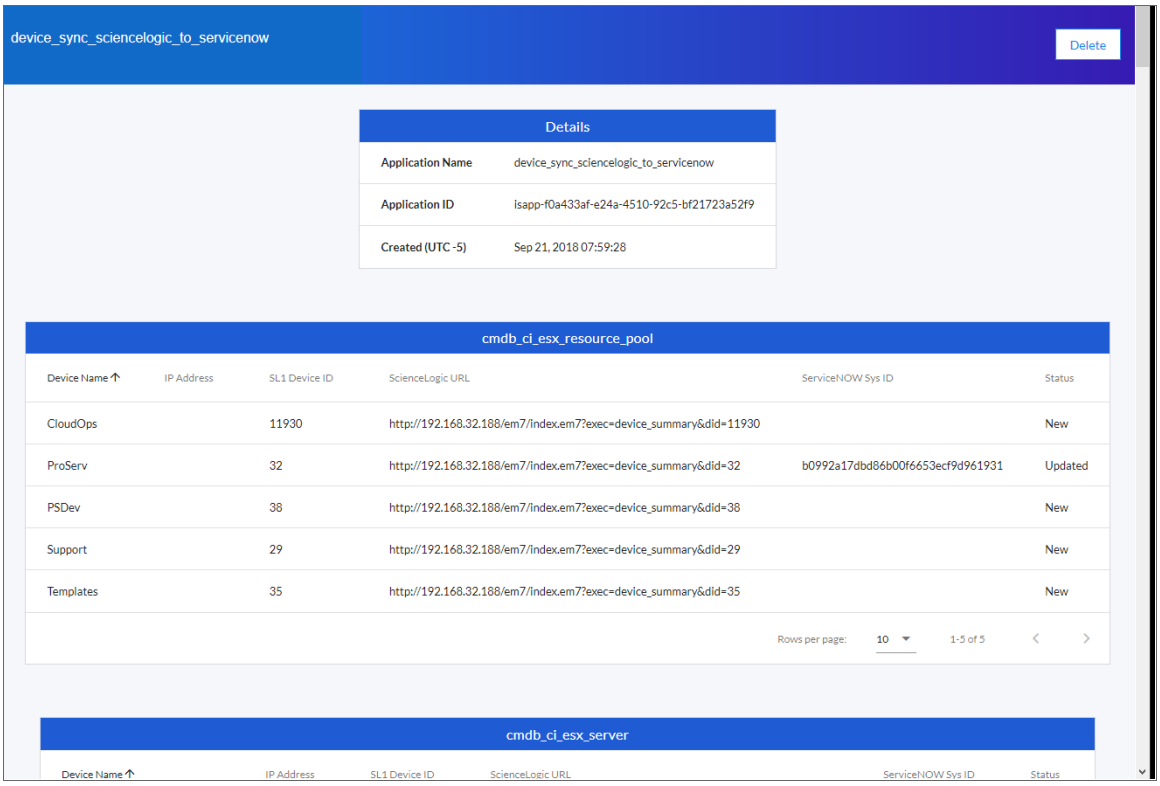
**NOTE**: Not all applications generate reports. Currently, only the "PowerFlow System Diagnostics" applications support the generation of reports.

To view details for an application report:

1. On the **Reports** page ( ), click the name of the application to expand the list of reports for that application.

> **TIP:** Click the arrow buttons to scroll forward and back through the list of reports.

2. Click a report name in the **Report ID** column. The **Report Details** page appears:




> **TIP:** The ***Status*** field in a report displays the current state of the synced item, which can include *New*, *Removed*, *Updated*, or *Unchanged*.

3. To view the detail page for the application on the **Applications** page, click the **Application Name** link.

> **TIP:** From the detail page for the application, click **[Reports]** ( ) to return to the **Reports** page.

4. To delete a report, open the report and click **[Delete]**. Click **[OK]** to delete the report.

# Chapter 6

# Managing Configuration Objects

## Overview

This chapter describes how to use the **Configurations** page ( ) 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?

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 configurations.

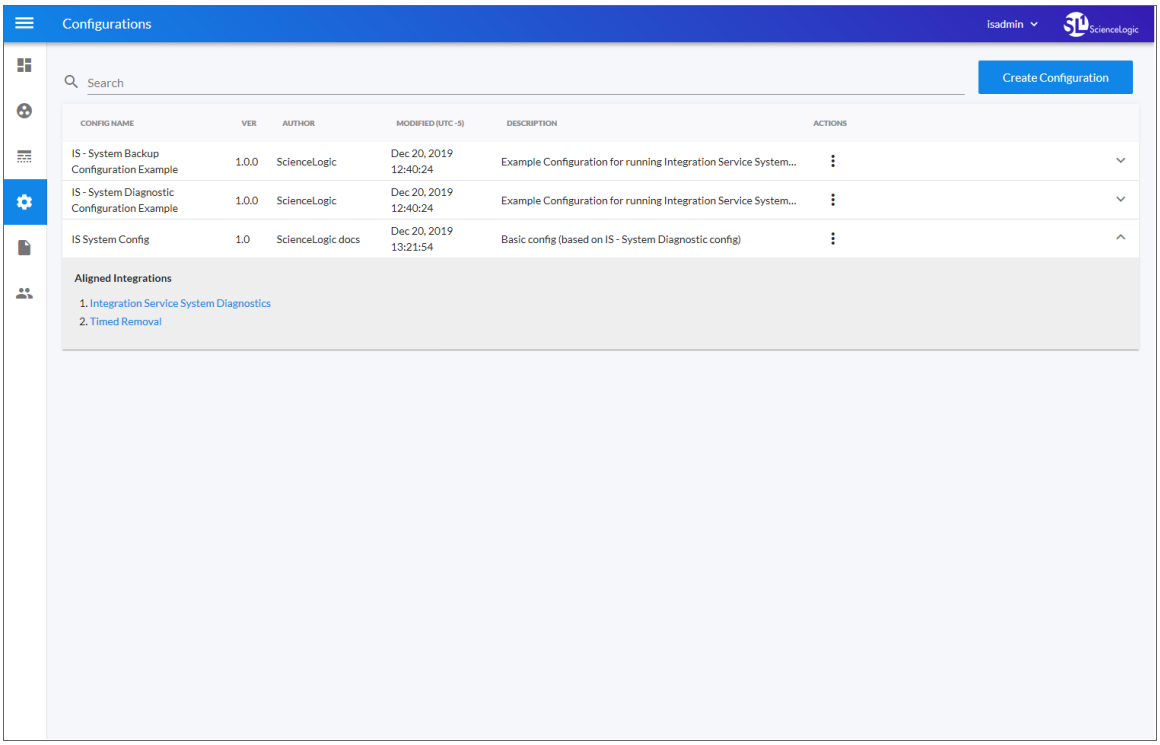
Configuration objects can map variables from SL1 to a third-party platform. For instance, SL1 has device classes and various third-party platforms have CI classes; the configuration object would map these two variables.

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 the **Applications** page. Before you can run an 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).

You can include the **config.** prefix with a variable to tell PowerFlow to use a configuration file to resolve the variable. If you want to re-use the same settings between applications, such as hostname and credentials, define configuration variables.

The **Configurations** page ( ) displays a list of available configurations. From this page you can create and edit configuration objects:




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. Click the **[Actions]** button ( ) for a configuration object to edit or delete that object.

**TIP:** Click the down arrow icon ( ) for a configuration object to see which applications are currently "aligned" with that configuration object.

# 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 Synchronization PowerPack will not overwrite a user configuration.

> **TIP:** Many Synchronization PowerPacks 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:** 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]**, 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.

# Editing a Configuration Object

To edit an existing configuration object:

1. Go to the **Configurations** page.
2. Click the **[Actions]** button ( ⋮ ) for the configuration object you want to edit and select *Edit*. The **Configuration** pane appears.
3. 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. 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.

# Chapter 7

# 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

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.

# 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. | debug/info (varies between development and product environments) |
| celery_log_level | The log level for Celery-related components and modules. | debug/info (varies between development and product environments) |

# PowerFlow Log Files

In PowerFlow version 2.1.0 and later, additional logging options were added for **gui**, **api**, and **rabbitmq** services. The logs from a service 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

Logs for the rabbitmq service were not previously persisted, but with PowerFlow 2.1.0, 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.

# Viewing the Step Logs and Step Data for a PowerFlow Application

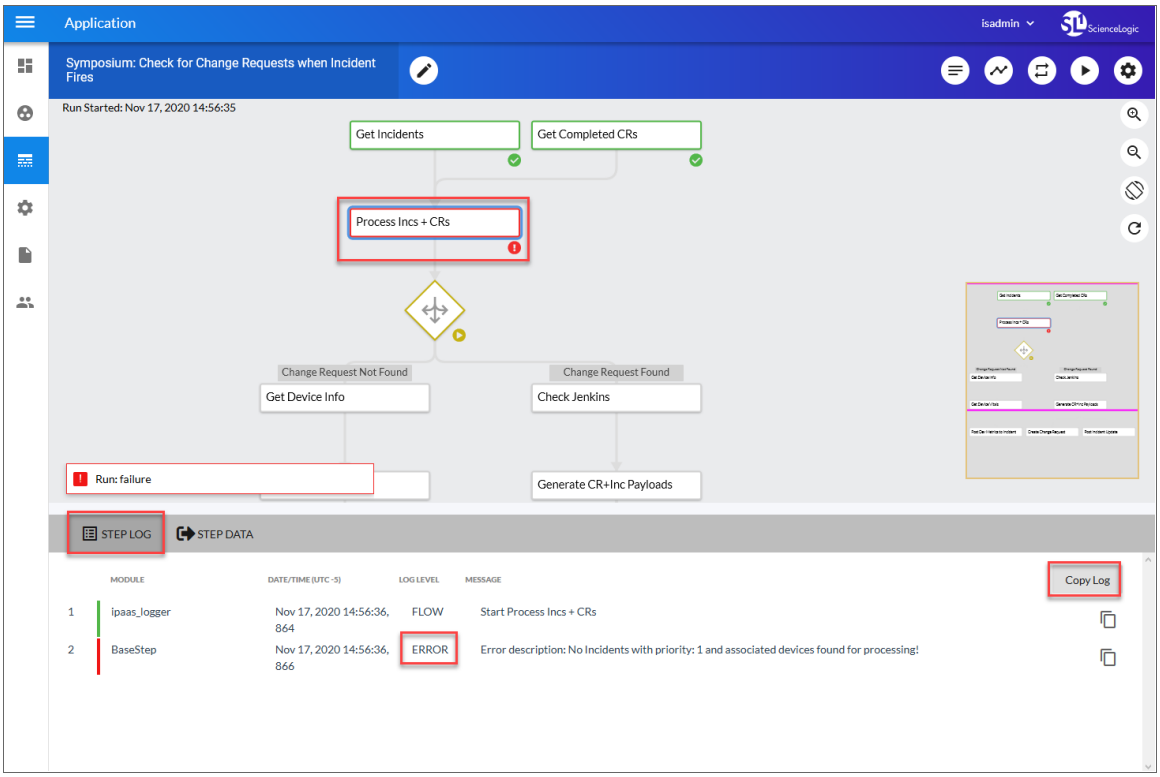
The **[Step Log]** tab on an **Application** page displays the time, the type of log, and the log messages from a step that you selected in the main pane. All times that are displayed in the **Message** column of this pane are in seconds, such as "stop Query and Cache ServiceNow CI List|***5.644190788269043***".

> **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:




**TIP:** For longer log messages, click the down arrow icon ( ) in the **Message** column of the **[Step Log]** tab to open the message.

4. 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.
5. To copy the entire log, click the **[Copy Log]** button.
6. Click the **[Step Data]** tab to display the JSON data that was generated by the selected step.
7. Click the **[Step Log]** tab to close the tab.
8. To generate more detailed logs when you run this application, hover over **[Run]** ( ) 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.
4. Click the **[Save]** button and close the **Configuration** pane.
5. Click the **[Run Now]** button ( ) to run the "Timed Removal" application.

# Chapter 8

# 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 the menu options, click the Advanced menu icon ( ••• ).

This chapter covers the following topics:

# 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 **Dashboard** 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** page of the ScienceLogic Support Site 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*.
- **SL1 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 SL1 PowerFlow PowerPack*.

  **NOTE:** Versions 105 and earlier of this PowerPack were named the *ScienceLogic: Integration Service PowerPack*.

- **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 the Couchbase PowerPack*.
- **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*.

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.

The following sub-topics describe the configuration steps you need to take for each PowerPack. For best results, complete these configuration steps in the given order to set up monitoring of PowerFlow within SL1.

# 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:

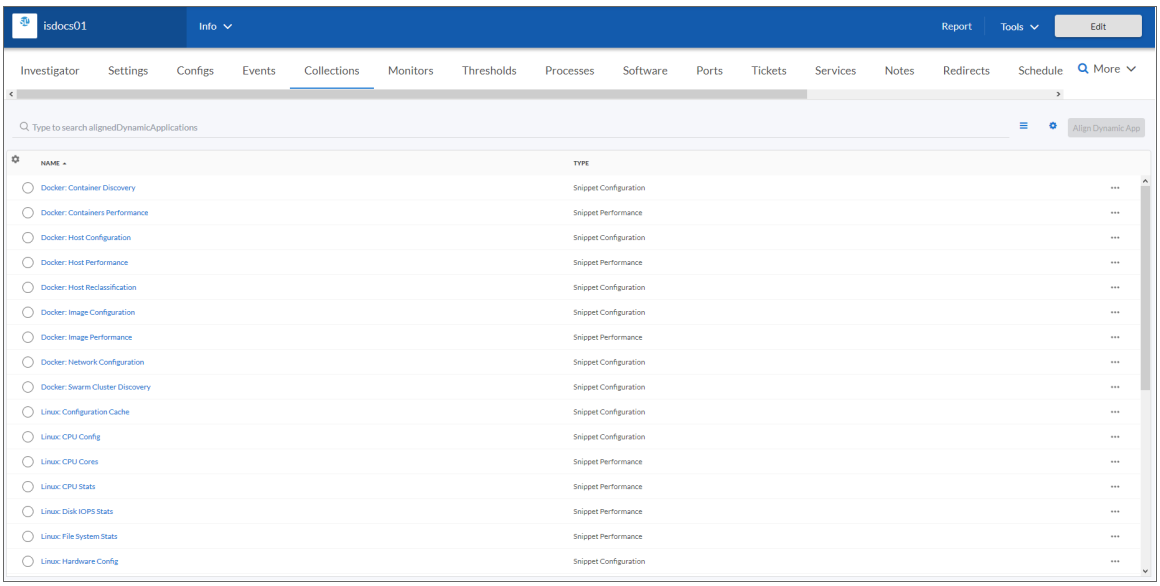
1. Make sure that you have already installed the Linux Base Pack PowerPack and the Docker PowerPack.
2. In SL1, go to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (  ) for the example **Docker Basic - Dev ssh** credential. The **Credential Editor** modal page appears.
3. Complete the following fields, and keep the other fields at their default settings:
   - ***Credential 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 As]** and close the **Credential Editor** modal page.
5. On the **Devices** page, click **[Add Devices]** to discover your PowerFlow server using the new Docker SSH new credential.

   TIP: 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.

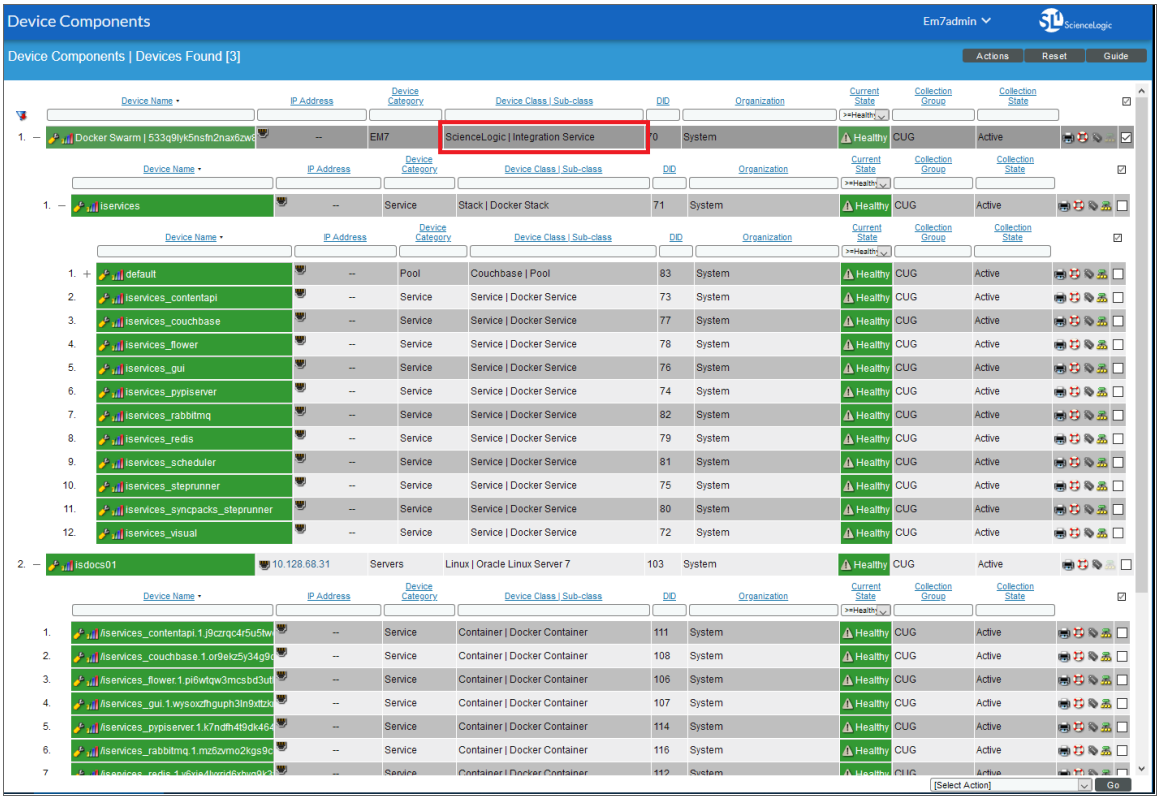
   **NOTE:** Select ***Discover Non-SNMP*** and ***Model Devices*** in the **Advanced options** section.

   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.

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:




**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 | Integration Service* as the Device Class and click **[Set Class]**. Click **[Save]** on the **Device Investigator** page to save your changes.

**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 SL1 PowerFlow PowerPack

**NOTE:** Versions 105 and earlier of this PowerPack were named the *ScienceLogic: Integration Service PowerPack*.

The *SL1 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.

To configure SL1 to monitor PowerFlow, you must first create a SOAP/XML credential. This credential allows the Dynamic Applications in the *PowerFlow* PowerPack to communicate with PowerFlow.

In addition, before you can run the Dynamic Applications in the *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

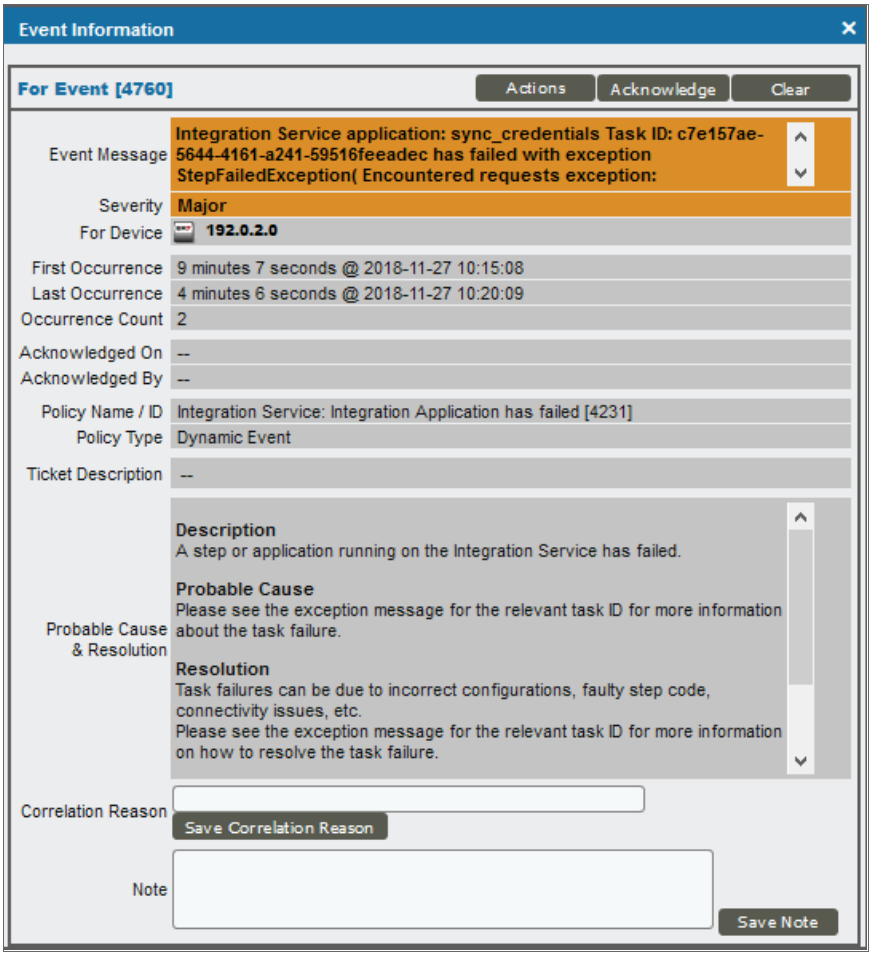
To configure the PowerFlow PowerPack:

1. In SL1, make sure that you have already installed the Linux Base PowerPack, the Docker PowerPack, and the SL1 PowerFlow PowerPack on your SL1 system.
2. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (  ) for the **IS - Example** credential. The **Credential Editor** modal page appears.
3. Complete the following fields, and keep the other fields at their default settings:
    - ***Profile Name***. Type a 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
    - ***Timeout (seconds)***. Type "20".
    - ***Embed Value [%1]***. Type "False".
4. Click the **[Save As]** button and close the **Credential Editor** modal page. You will use this new credential to manually align the following Dynamic Applications:
    - REST: Performance Metrics Monitor
    - ScienceLogic: Integration Service Queue Configuration
    - ScienceLogic: Integration Service Workers

5. Go to the **Devices** page, select the device representing your PowerFlow server, and click the **[Collections]** tab.
6. Click **[Edit]**, click **[Align Dynamic App]**, 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

The "ScienceLogic: Integration Service Queue Configuration" Dynamic Application generates 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
```

# Configuring the Couchbase PowerPack

Couchbase stores all cache and configuration data on PowerFlow. Monitoring the performance of your PowerFlow is critical in ensuring the health of your PowerFlow instance.

After you install the Couchbase PowerPack in SL1, create a new Couchbase SOAP/XML credential. Using that credential, you need to manually align the "Couchbase Component Count" and "Couchbase Pool Discovery" Dynamic Applications with the Docker Swarm root device. These steps are covered in detail below.

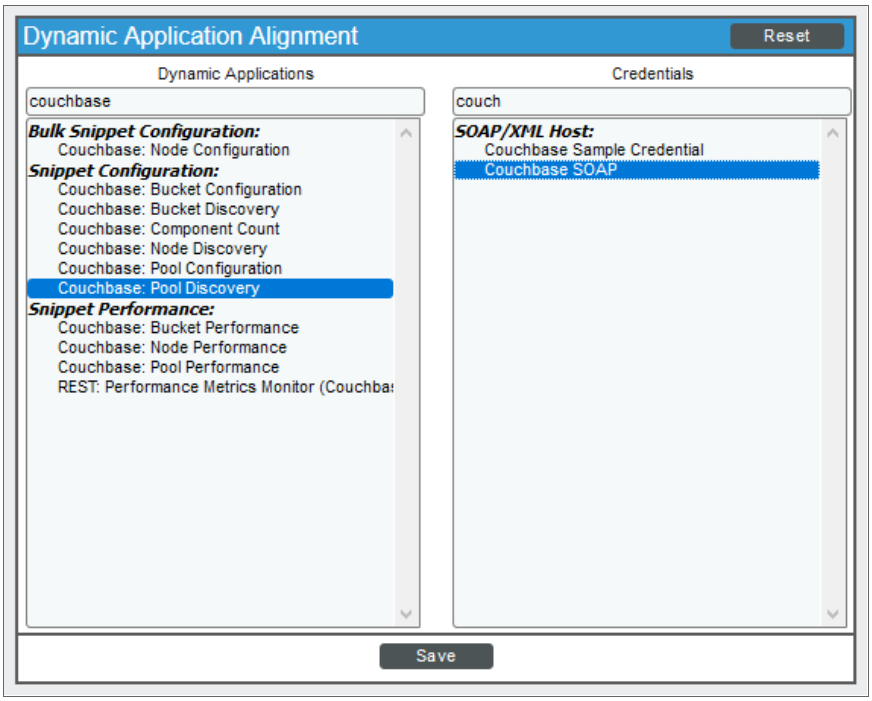
To configure the Couchbase PowerPack:

1. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (  ) for the **Couchbase Sample** credential. The **Credential Editor** modal page appears.
2. Complete the following fields, and keep the other fields at their default settings:
   - ***Profile Name***. Type a new descriptive name for the credential.
   - ***URL***. Type the full URL for PowerFlow. Ensure that the port 8091 is appended to the hostname, and use *https* for this URL.
   - ***HTTP Auth User***. Type the username for your PowerFlow instance.
   - ***HTTP Auth Password***. Type the password for your PowerFlow instance.

   **NOTE:** For a clustered PowerFlow environment, point the Couchbase credentials at the load balancer for PowerFlow. The example above is for a single node deployment.

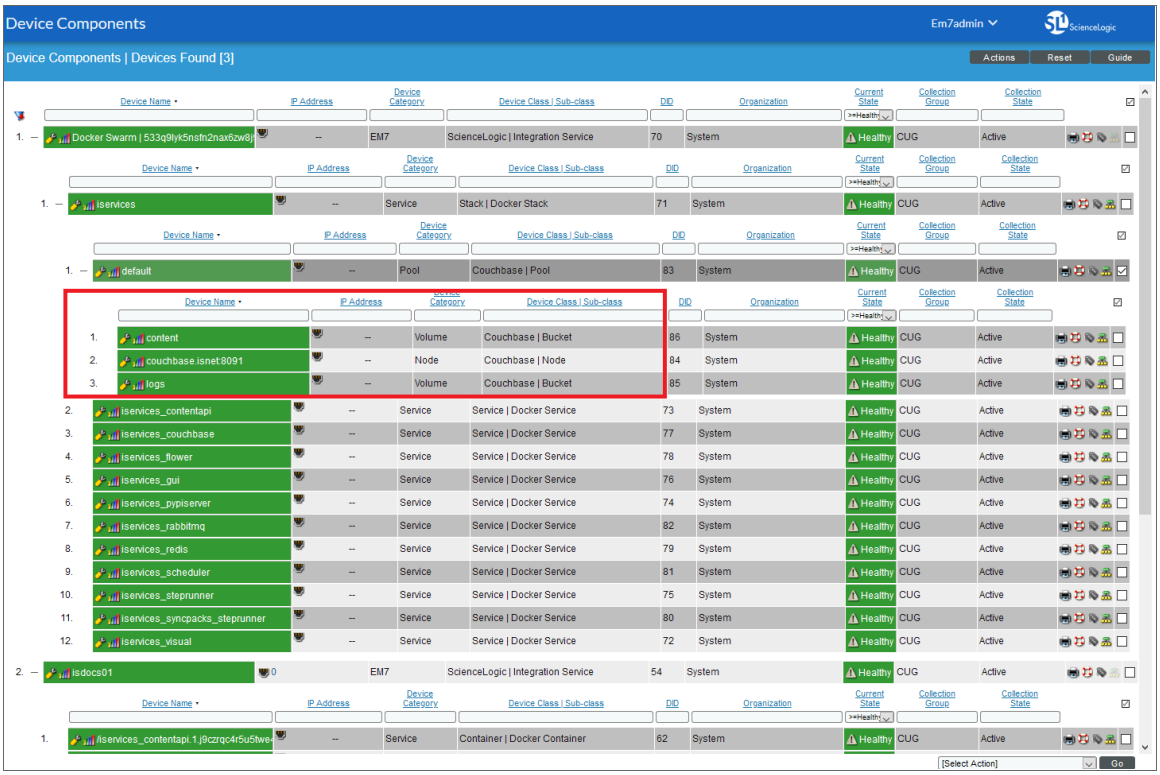
3. Click **[Save As]** and close the **Credential Editor** modal page.
4. Go to the **Device Components** page (Devices > Device Components) and expand the Docker Swarm root device by clicking the + icon.
5. Click the wrench icon (  ) for the "Stack | Docker Stack" component device (iservices) and click the **[Collections]** tab.

6. Align the "Couchbase: Pool Discovery" Dynamic Application by clicking the **[Actions]** button and selecting *Add Dynamic Application*. The **Dynamic Application Alignment** modal appears:




7. Select the "Couchbase: Pool Discovery" Dynamic Application and select the Couchbase credential that you created in steps 1-3. Click **[Save]**.
8. Click the **[Actions]** button, select *Add Dynamic Application* and align the "Couchbase: Component Count" Dynamic Application and the Couchbase credential. Click **[Save]**.
9. Select the "Couchbase: Component Count" Dynamic Application and select the IS credential that you created in steps 2-3. Click **[Save]**. SL1 models out your Couchbase components and provides you with additional information about the usage of the Couchbase service.

10. Navigate to the **Device Components** page (Devices > Device Components) to see the Couchbase components:




# Configuring the RabbitMQ PowerPack

You can monitor the RabbitMQ service with the AMQP: RabbitMQ PowerPack. This PowerPack monitors RabbitMQ configuration data and performance metrics using Dynamic Applications, and the PowerPack creates a major event in SL1 for any applications in PowerFlow that are in a *Failed* state.

After you install the RabbitMQ PowerPack in SL1, create a new SOAP/XML credential. Using that credential, you need to manually align the following Dynamic Applications:

- ScienceLogic: Integration Service Queue Configuration
- AMQP: RabbitMQ Configuration
- AMQP: RabbitMQ Performance

To configure the RabbitMQ PowerPack:

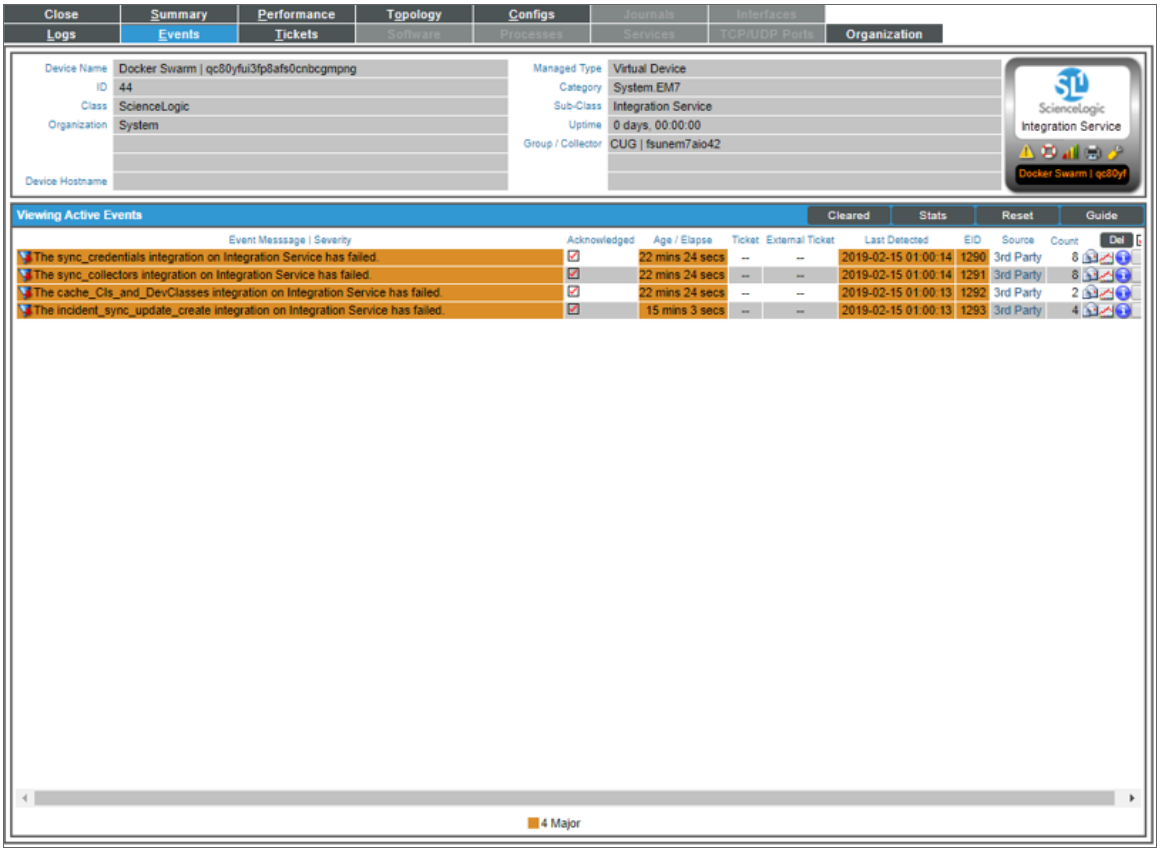
1. In SL1, navigate to the **Credential Management** page (System > Manage > Credentials) and click the wrench icon (  ) for the **IS Sample** credential. You can also create a new SOAP/XML credential. The **Credential Editor** modal page appears.

2. Complete the following fields, and keep the other fields at their default settings:
    - ***Profile Name***. Type a new descriptive name for the credential.
    - ***URL***. Type the full URL for your PowerFlow and use *https* for this URL.
    - ***HTTP Auth User***. Type the username for your PowerFlow instance.
    - ***HTTP Auth Password***. Type the password for your PowerFlow instance.

    > **NOTE:** For a clustered PowerFlow environment, point the credentials at the load balancer for the PowerFlow system. The example above is for a single node deployment.

3. Click **[Save As ]** and close the **Credential Editor** modal page.
4. Go to the **Device Components** page (Devices > Device Components) and click the wrench icon (  ) for the Docker Swarm root device.
5. Click the **[Collections]** tab on the **Device Properties** window.
6. Align the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application by clicking the **[Actions]** and selecting *Add Dynamic Application*. The **Dynamic Application Alignment** modal appears.

7. Select the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application and select the credential that you created in step 2. Click **[Save]**. This Dynamic Application queries PowerFlow every 15 minutes by default to retrieve information about any failed integrations, which generates a Major event in SL1 (the events auto-expire after 90 minutes):




**TIP:** The events generated by this Dynamic Application include the **Integration ID**, which you can use to find the relevant application on your PowerFlow instance. Copy the name in the event message and navigate to *https://<PowerFlow>/integrations/<integration_ID>*.

8. To view more information about your failed applications, navigate to the **[Configurations]** tab for the device and click the report for the "ScienceLogic: Integration Service Queue Configuration" Dynamic Application. This configuration report shows you more information about the failed integrations on your Integration Service instance. For example, you can use the ***Last Run ID*** field to find the exact logs for a specific execution of the application. To do this, copy the *Integration ID* and the *Last Run ID* and navigate to *https://<PowerFlow>/integrations/<integration_ID>?runid=<last_run_id>*
9. Align the "AMQP: RabbitMQ Configuration" and "AMQP: RabbitMQ Performance" Dynamic Applications using the same process as steps 6-7.

# 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 9

# 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".

> **NOTE:** The different actions available with the **powerflowcontrol** utility should replace the need to configure and run the different PowerPacks described in *Using SL1 to Monitor PowerFlow*.

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?

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 PowerFlow version 2.0.0 or later. If you are using an older version of PowerFlow, you can also access the latest version at the ScienceLogic Support site.

To download the latest version of the **powerflowcontrol** (pfctl) command-line 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 to the current release. The **Release Version** page appears.
4. In the **Release Files** section, click the link for the **powerflowcontrol** (pfctl) command-line utility. A **Release File** page appears.
5. Click **[Download File]** at the bottom of the **Release File** page.

**NOTE:** The **powerflowcontrol** command-line utility requires port 22 on all host nodes.

**NOTE:** You can use key-based authentication instead of username and password authentication for the **powerflowcontrol** command-line utility.

The **powerflowcontrol** command-line utility was updated to allow any user provided to run the **powerflowcontrol** utility. Please note that the default **isadmin** user already meets these requirements, and this update is relevant only if your PowerFlow environment uses custom users or processes.

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)

WARNING: 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: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`

For a list of all of the actions you can run on a single node, SSH to the PowerFlow server and run the following command:

```
powerflowcontrol node-action --help
```

For a list of all of the actions you can run on a clustered system, run the following command:

```
powerflowcontrol cluster-action --help
```

# 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.

## healthcheck

The following commands show the formatting for a **healthcheck** action for a *single node*, followed by an example:

```
pfctl --host <host> <username>:<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 <host> <username>:<password> --host <host> <username>:<password> --host
<host> <username>:<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
```

## autoheal

The following commands show the formatting for an **autoheal** action for a *single node*, followed by an example:

```
pfctl --host <host> <username>:<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>:<password> --host <host> <username>:<password> --host
<host> <username>:<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_contentapi 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_contentapi 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 missing indeces:
Missing the following indexes: {'couchbase.isnet': ['idx_casbin'], 'couchbase-work-
er2.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 <PowerFlow_host1>  <username>:<password> --host <PowerFlow_host2>
<username>:<password> --host <PowerFlow_host3>  <username>:<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 cluster-action --action generate_haproxy_config`

# upgrade

This topic explains how to use the **upgrade** action in the **powerflowcontrol** utility to upgrade a clustered environment from PowerFlow version 1.8.x to 2.x.x. The **upgrade** action is *only* for upgrading a cluster from a 1.8.x installation, and should not be run otherwise.

**NOTE:** The **powerflowcontrol** utility cannot be installed on a 1.8.4 PowerFlow system because the utility is not compatible with Python 2.7. If you want to use the utility on a 1.8.4 PowerFlow system, you will need to run the **is_upgrade_to_v2.sh** on any of the cluster nodes. This script will update the nodes to Python 3.x, and then you can download and install the **powerflowcontrol** utility on the upgraded node, and upgrade of the other nodes from that node.

**NOTE:** If you have an environment that has Python 3.6 or later available, the powerflowcontrol package can be installed in that system (a local environment, virtual machine, or another PowerFlow system). That environment must have an SSH connection to the PowerFlow nodes that you want to upgrade. Also, be aware that only certain actions, like the **upgrade** action, can be run from an external system, but most of the actions that the **powerflowcontrol** use will only run inside the cluster nodes.

To run the **upgrade** action in a clustered environment:

1. Back up your PowerFlow data. For more information, see *Backing up Data*.
2. Run the **upgrade** action with the **powerflowcontrol** utility:

> **NOTE:** You will need to know if the upgrade process will be done offline or online, and you will need the url of the PowerFlow RPM file or the local path of the PowerFlow RPM or ISO file.

For an *offline* upgrade, run the following command on the PowerFlow instance:

```
pfctl --host <swarm-node1-ip> <is-username>:<is-password> --host <swarm-node2-ip> <is-username>:<is-password> node-action --action upgrade --upgrade_args offline <PowerFlow-iso-local-path>
```

where *<PowerFlow-iso-local-path>* is a local path such as `/home/isadmin/sl1-powerflow-2.1.2.iso`.

For an *online* upgrade, run the following command on the PowerFlow instance:

```
pfctl --host <swarm-node1-ip> <is-username>:<is-password> --host <swarm-node2-ip> <is-username>:<is-password> node-action --action upgrade --upgrade_args online <PowerFlow-rpm-url-or-local-path>
```

3. Manually open the firewall ports you need, or use the powerflowcontrol *open-firewall-ports* action.
4. Synchronize NTP in all the nodes. For more information, see *Preparing the PowerFlow System for High Availability*.
5. Manually verify that NTP is synchronized in all the nodes manually, or use the **powerflowcontrol** node action to verify that.
6. To deploy PowerFlow, run the **autocluster** action using the **powerflowcontrol** utility if your clustered environment contains three nodes.

> **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.

7. Run the **healthcheck** cluster action using the **powerflowcontrol** utility to check that the cluster environment was deployed correctly.
8. Run the **autoheal** cluster action using the **powerflowcontrol** utility to fix any errors found during the **healthcheck** action run. You should only run the autoheal action if the **healthcheck** action found errors.

# open_firewall_ports

To open firewall ports for a single node, SSH to the PowerFlow server and run the following command:

```
pfctl --host <is_host> isadmin:<password> node-action --action open_firewall_ports
```

> **TIP:** Many of the other **powerflowcontrol** actions use the same format as the **open_firewall_ports** action, above. The only change you need to make for those commands is to replace the name of the action at the end of the command. For example: `pfctl --host <is_host> isadmin:<password> node-action --action pull_latest_images`

# 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
```

To view the current node password unencrypted, run the following command:

```
pfctl password decrypt
```

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.

# Chapter 10

# Troubleshooting SL1 PowerFlow

## Overview

This chapter includes troubleshooting resources, procedures, and frequently asked questions related to working with SL1 PowerFlow.

This chapter covers the following topics:

# 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.
2. Note the Docker container version, and verify that the Docker services are running.
3. If a certain service is failing, make a note the service name and version.
4. 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`.
5. 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 Synchronization PowerPack version, if applicable.
2. Make a note of whether the user is running an update set or a version of the Certified/Scoped application, if relevant.
3. Make a note of the ServiceNow application that is failing on 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. Use *guest/guest* for the login.
- **https://<*IP of PowerFlow*>/flower**. Provides access to Flower, a tool for monitoring and administrating Celery clusters.

**NOTE:** For version 2.0.0 and later of PowerFlow, 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 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.

## 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
```

First, verify that your services are up. If there is an issue with your RabbitMQ volume, you can clear the volume with the following commands:

```
docker service rm iservices_rabbitmq
docker volume rm iservices_rabbitdb
```

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"
    }
  }
]
```

### 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"
```

## 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
```

```
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* 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
```

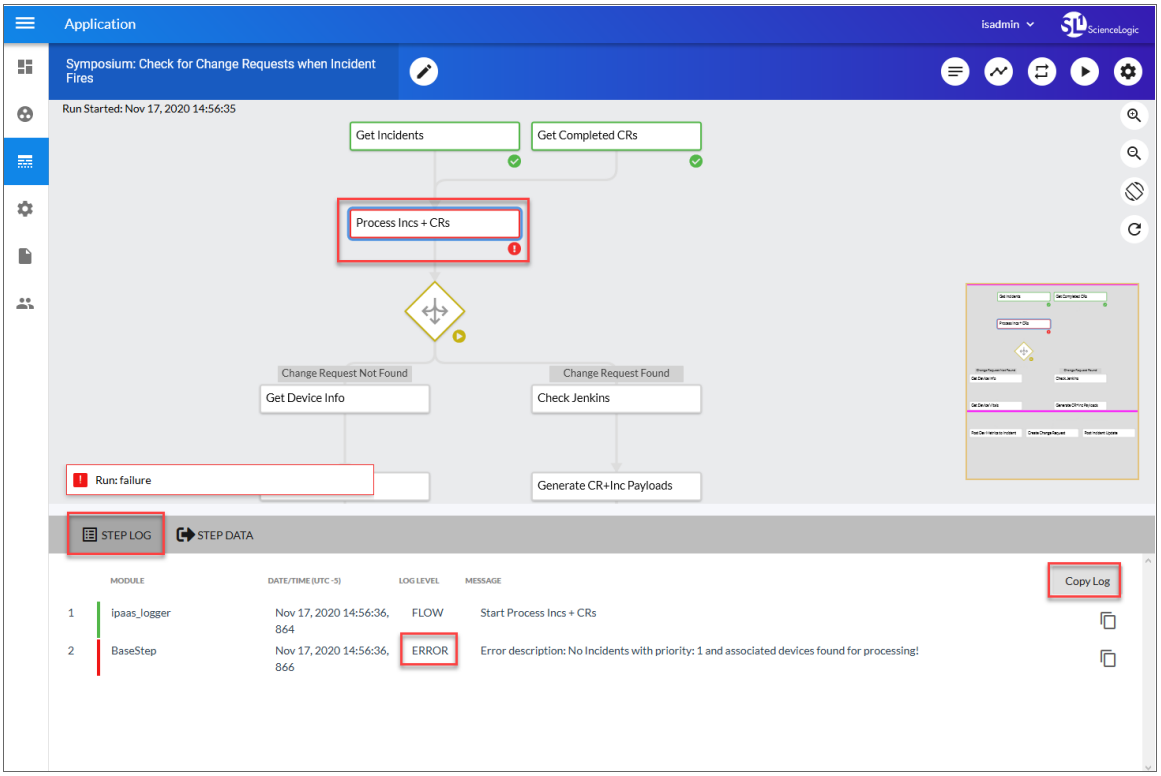
```
"error creating external connectivity network: Failed to Setup IP tables: Unable to
enable SKIP DNAT rule: (iptables failed: iptables --wait -t nat -I DOCKER -i docker_
gwbridge -j RETURN: iptables: No chain/target/match by that name."
```

# 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 with "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]** (  ) from and select *Debug Run* from the pop-up menu. PowerFlow executes the application in Debug Mode with a log level of 10.

To run an application in Debug Mode using the API:

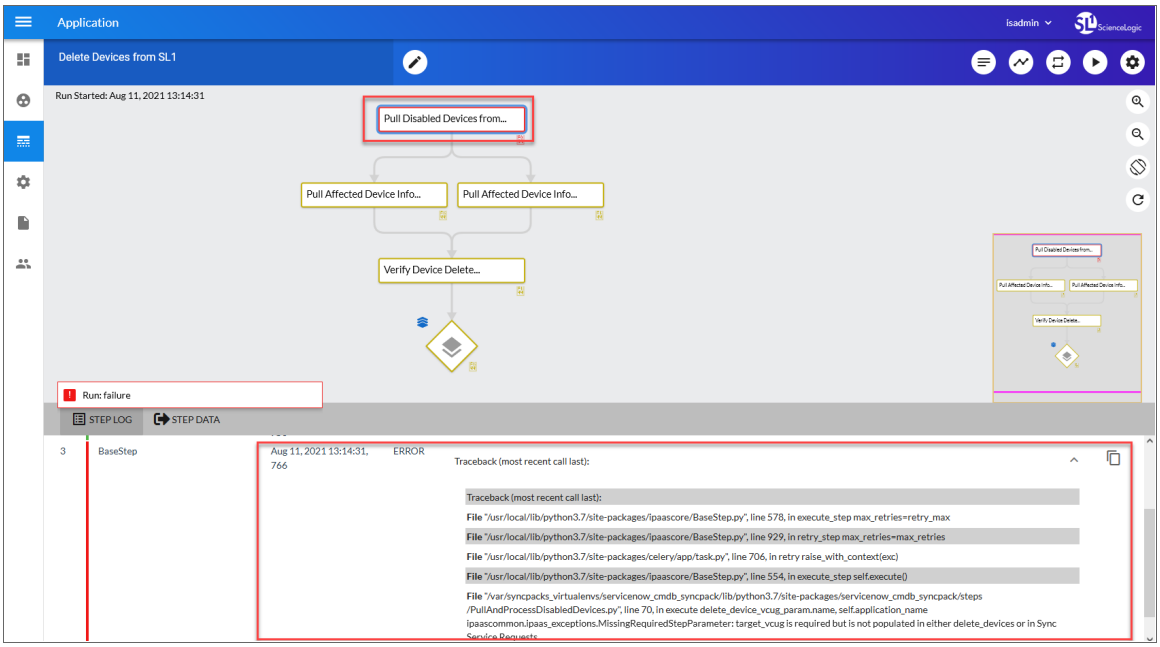
1. POST the following to the API endpoint:

```
https://<PowerFlow>/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:em7admin -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.

## 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 Synchronization PowerPacks 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 steprunner_syncpacks service can be limited to just workers?

The syncpack_steprunners are responsible for keeping all Synchronization PowerPack virtual environments in sync on all nodes. As you install or upgrade Synchronization PowerPacks, 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 Synchronization PowerPacks onto the nodes of a system.

## What is the difference between the steprunner_syncpacks and the steprunner services?

The syncpack_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 Synchronization PowerPack on the SyncPacks page?

If you have uploaded a Synchronization PowerPack, 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* Synchronization PowerPack has been installed and is ready to be used.

Click the Filter icon ( ) at the top right of the **SyncPacks** page and click the ***Show all SyncPacks*** toggle to show only the activated Synchronization PowerPacks. Deselect he ***Show all SyncPacks*** toggle to see all uploaded Synchronization PowerPacks.

## Why can't I see or upload a Synchronization PowerPack?

If your PowerFlow system uses self-signed certificates, you will need to manually accept the certificate before you can upload Synchronization PowerPacks.

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 Synchronization PowerPacks.

Also, if a Synchronization PowerPack failed to load or activate, a red exclamation mark icon ( ) appears next to that Synchronization PowerPack 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 Synchronization PowerPack is that the Synchronization PowerPack dependencies are not installed. Review the "System Requirements" section of the release notes for that Synchronization PowerPack to ensure that you have installed all of the required applications for that Synchronization PowerPack.

## Why do I get a "Connection error" message when I try to install the System Utils Synchronization PowerPack?

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 Synchronization PowerPack 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-worker2.isnet
   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?

The PowerFlow system 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:

- ***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.

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.yml**.
4. You can define the environment variables for one or more worker processes. The docker-compose.yml file contains definitions for worker processes. For example, you might see something like this:

```
services:
  steprunner:
    image: sciencelogic/is-worker:latest
    environment:
    LOGLEVEL: 10
    celery_log_level: 10
    logdir: /var/log/iservices
    broker_url: 'pyamqp://guest@rabbit//'
    result_backend: 'redis://redis:6379/0'
    db_host: 'couchbase,localhost'
    secrets:
      - is_pass
      - encryption_key
    deploy:
    replicas: 2
    networks:
      - isnet
    volumes:
      - "/var/log/iservices:/var/log/iservices"
      - "statedb:/var/run/celery/states/"
```

```
steprunner_1:
  image: : sciencelogic/is-worker:latest
  environment:
  LOGLEVEL: 10
  celery_log_level: 10
  task_soft_time_limit: 30
  optimization: ""
  task_acks_late: 'False'
  logdir: /var/log/iservices
  broker_url: 'pyamqp://guest@rabbit//'
  result_backend: 'redis://redis:6379/0'

  db_host: 'couchbase,localhost'
  secrets:
    - is_pass
    - encryption_key
  deploy:
  replicas: 2
  networks:
    - isnet
  volumes:
    - "/var/log/iservices:/var/log/iservices"
    - "statedb:/var/run/celery/states/"

steprunner_2:
  image: : sciencelogic/is-worker:latest
  environment:
  LOGLEVEL: 10
  celery_log_level: 10
  task_soft_time_limit: 30
  optimization: '-Ofair'
  task_acks_late: 'False'
  logdir: /var/log/iservices
  broker_url: 'pyamqp://guest@rabbit//'
  result_backend: 'redis://redis:6379/0'
  db_host: 'couchbase,localhost'
  secrets:
    - is_pass
    - encryption_key
  deploy:
  replicas: 2
  networks:
    - isnet
  volumes:
    - "/var/log/iservices:/var/log/iservices"
    - "statedb:/var/run/celery/states/"
```

5. The services with names that start with "steprunner" are the workers for the PowerFlow system. To define the optimization variables, enter the variables and values in the definition of the worker, under the ***environment*** section. See the example in step #3 to see the syntax for environment variables.

6. After you have updated the docker-compose file, you can update and re-deploy the 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
```

## 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 remove a schedule that does not have a name?

If you encounter a schedule in PowerFlow that was created without a name, PowerFlow cannot update or delete that schedule using the API.

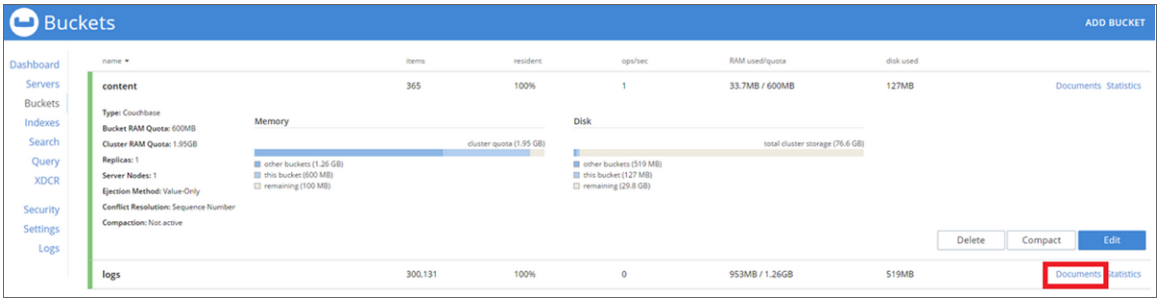
**NOTE:** This issue only affects versions of PowerFlow prior to version 1.8.2.

To address this issue, you will need to delete *all* schedules on PowerFlow, which involves going into Couchbase and deleting a file.

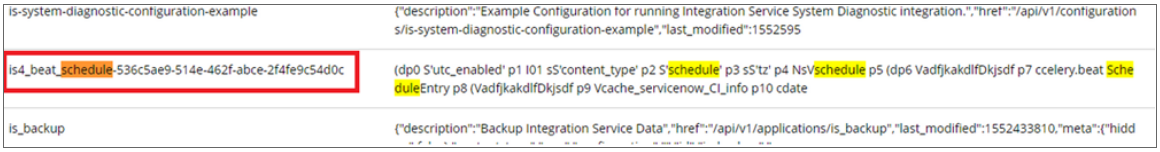
**WARNING:** Exercise extreme caution when performing this procedure, as deleting the wrong file will cause your PowerFlow instance to stop working.

To delete all schedules, including a schedule without a name:

1. Log in to the Couchbase management interface at **https://*<localhost>*:8091**.
2. Navigate to the **[Buckets]** tab.
3. Click the **Documents** link on the **content** bucket:




4. On the **content > documents** page, extend the results to show 100 records per page
5. Search for "schedule". You will see a file similar to the following example:




6. Delete the file.

**WARNING:** Deleting this file deletes all schedules on your PowerFlow instance.

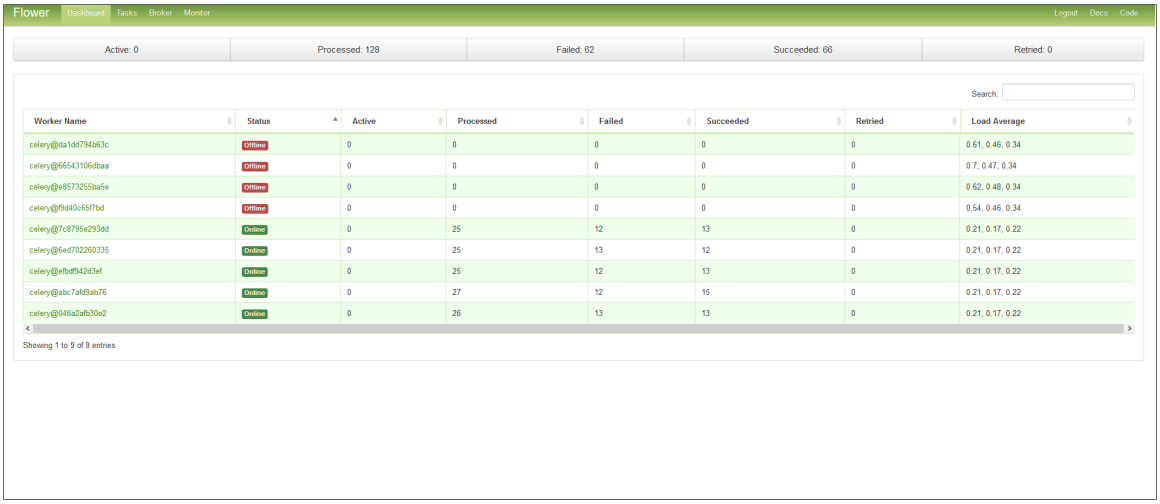
## How do I identify and fix a deadlocked state?

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** at the end of the URL, such as **https://192.0.2.0/flower/**.
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
   ```

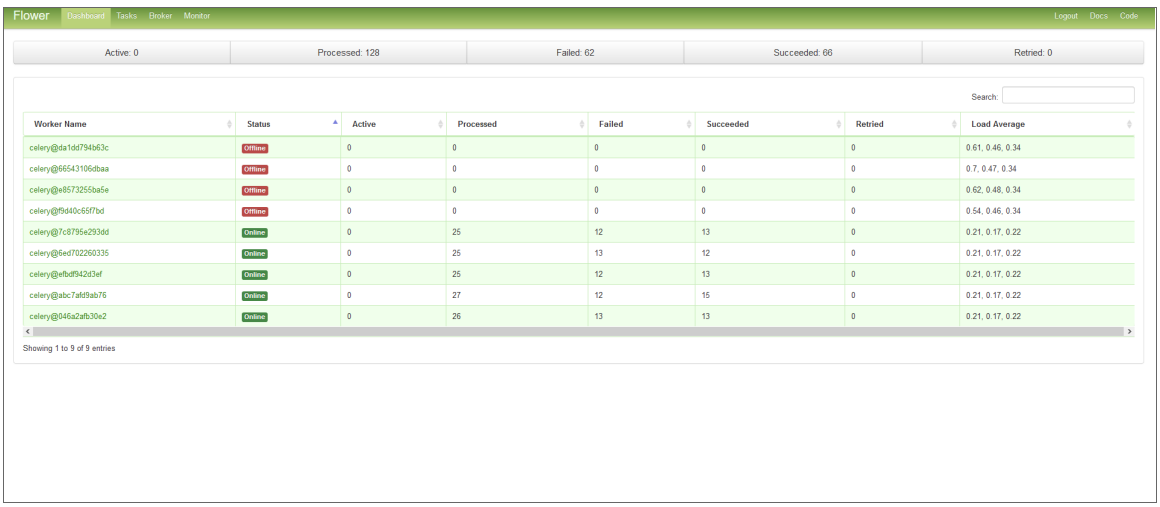
   where *number_greater_than_3* is an integer greater than 3.

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** at the end of the URL, such as **https://192.0.2.0/flower/**.
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 (such as version 1.8.0 force-installed on 1.8.0), 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 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 Synchronization PowerPacks from the PowerFlow user interface to install the environments on the new nodes. Applications from those Synchronization PowerPacks 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.

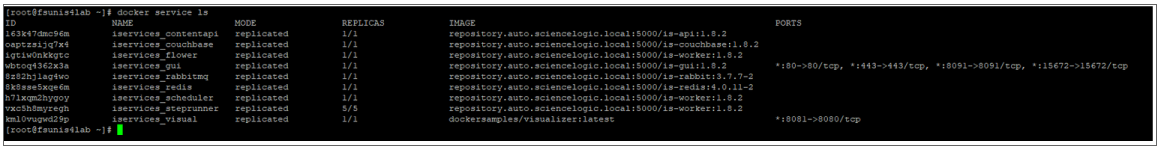
## What do I do if I get a Code 500 Error when I try to access the PowerFlow user interface?

To address this issue:

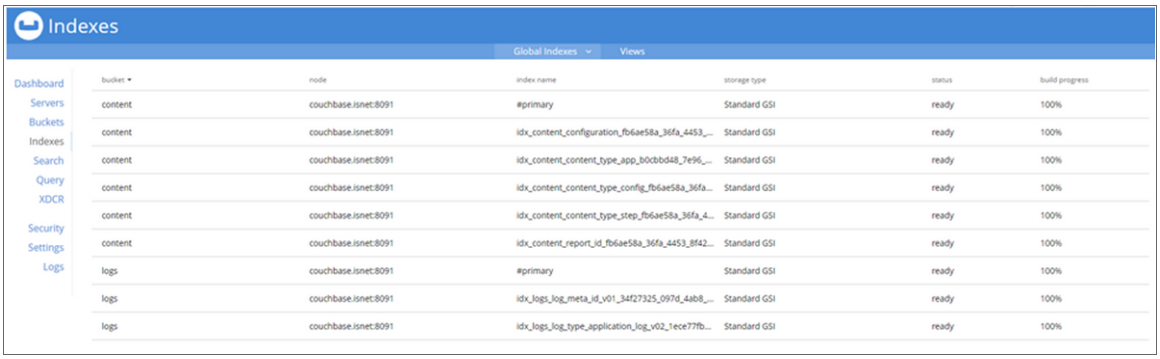
1. SSH to your PowerFlow instance.
2. Check your Docker services with the following command:

   ```
   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 em7admin
iscli -uaf <APPLICATION_FILE.JSON> -U isadmin -p em7admin
iscli -ucf <CONFIG_FILE.JSON> -U isadmin -p em7admin
iscli -usf <STEP_FILE.PY> -U isadmin -p em7admin -H <IS_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
   ```

# Chapter 11

# 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

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.

# Available Endpoints

## POST

**/applications**. Add a new application or overwrite an existing application.

**/applications/{appName}/run**. Run a single application by name with saved or provided configurations.

**/applications/run**. Run a single application by name. For more information, see *Querying for the State of a PowerFlow Application*.

**/configurations**. Add a new configuration or overwrite an existing configuration.

**/roles/owner**. Add a new owner assigned a specific role.

**/steps**. Add a new step or overwrite an existing step.

**/steps/run**. Run a single step by name.

**/schedule**. Add a new scheduled PowerFlow application.

**/syncpacks/{syncpackName}/install**. Install a specific Synchronization PowerPack version by name.

**/tasks/{taskId}/replay**. Replay a specific PowerFlow application. Replayed applications run with the same application variables, configuration, and queue as the originally executed application.

**/tasks/{taskId}/revoke**. Revoke or terminate a specific task or application. By default, this command will not terminate the current running task. If an application ID is provided, all tasks associated with that application are revoked.

## 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

**/about**. Retrieve version information about the packages used by this PowerFlow system, including the version of PowerFlow.

**/applications**. Retrieve a list of all available applications on this PowerFlow system.

**/applications/{appName}**. Retrieve a specific application.

**/applications/{appName}/logs**. Retrieve the logs for the specified application.

**/cache/{cache_id}**. Retrieve a specific cache to gather information about the user interface and the PowerFlow applications.

**/configurations**. Retrieve a list of all configurations on this PowerFlow system.

**/configurations/{configName}**. Retrieve a specific configuration.

**/license?type=platform**. Retrieve license data for this PowerFlow system.

**/reports**. Retrieve a list of paginated reports.

**/reports/{reportId}**. Retrieve a specific report by ID.

**/roles**. Retrieve a list of available roles on this PowerFlow system.

**/roles/owner**. Retrieve a list of roles assigned to owners on this PowerFlow system.

**/roles/owner/{owner}**. Retrieve the role assigned to a specific owner.

**/sessions**. Retrieve a list of sessions for this PowerFlow system.

**/sessions/status**. Retrieve the Session Management status for this PowerFlow system.

**/sessions/username/{username}**. Retrieve the session IDs for a specific user.

**/sessions/{session_id}**. Retrieve a specific session from Session Management for this PowerFlow system.

**/schedule**. Retrieve a list of all scheduled applications on this PowerFlow system.

**/steps**. Retrieve a list of all steps on this PowerFlow system.

**/steps/{stepName}**. Retrieve a specific step.

**/syncpacks**. Retrieve a list of all Synchronization PowerPacks on this PowerFlow system.

**/syncpacks/{synpackName}**. Retrieve the full details about a specific Synchronization PowerPack.

**/api/v1/syncpacks?only_installed=true**. Retrieve a list of only the installed Synchronization PowerPacks on this system.

**/api/v1/syncpacks?only_activated=true**. Retrieve a list of only the activated Synchronization PowerPacks on this system.

**/tasks/{taskId}**. Retrieve a specific task.

## REST

**/tasks**. Terminate all running tasks.

**/tasks/{taskId}**. Terminate a specific running task.

## DELETE

**/applications/{appName}**. Delete a PowerFlow application by name.

**/cache/{cache_id}**. Delete a cache entry by name.

**/configurations/{configName}**. Delete a configuration by name.

**/license?type=platform**. Delete license data for this PowerFlow system.

**/roles/owner**. Delete a specific owner role.

**/schedule**. Delete a scheduled PowerFlow application by ID.

**/sessions**. Delete a list of sessions for this PowerFlow system.

**/sessions?all=true**. Delete all sessions for this PowerFlow system.

**/sessions/status**. Delete the Session Management status for this PowerFlow system.

**/sessions/username/{username}**. Delete the session IDs for a specific user.

**/sessions/{session_id}**. Delete a specific session from Session Management for this PowerFlow system.

**/reports/{appName}**. Delete a specific report by name.

**/reports/{reportId}**. Delete a specific report by report ID.

**/steps/{stepName}**. Delete a specific step by name.

**/syncpacks/{spName}**. Delete a specific Synchronization PowerPack by name.

# Appendix

# A

# Configuring the PowerFlow System for High Availability

## Overview

This appendix describes how to create High Availability deployments to protect the data in PowerFlow.

This appendix covers the following topics:

# 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.

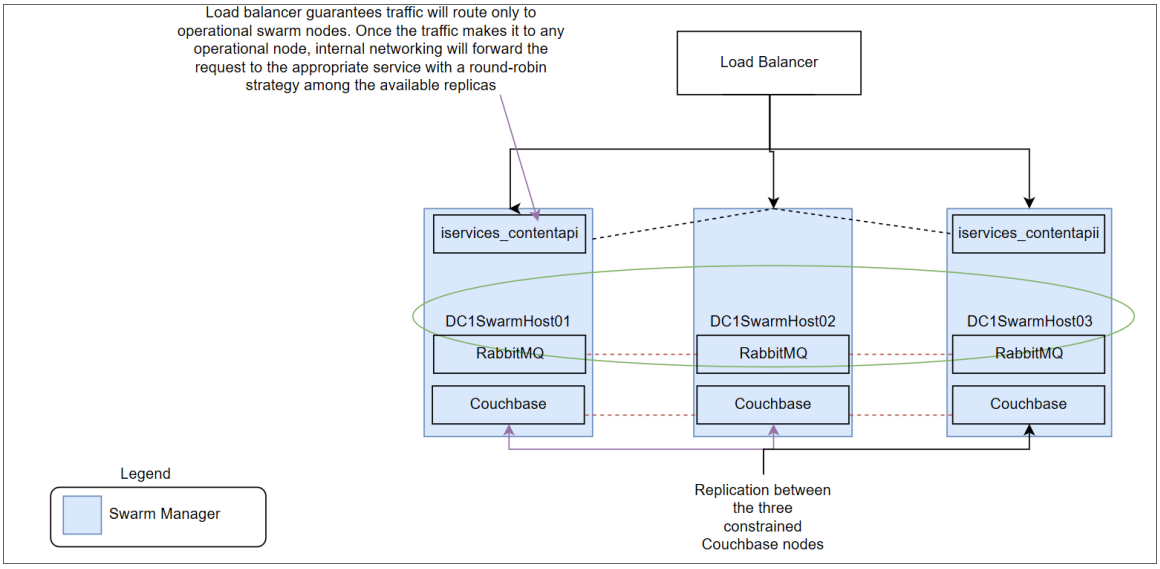
### Configuration

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.

Below is an example **docker-compose-override.yml** file for PowerFlow 2.0.0 or later:

```
services:
  contentapi:
    environment:
      db_host: "couchbase.isnet,couchbase-worker.isnet,couchbase-worker2.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/is-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/is-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-worker2.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/is-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/is-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
    volumes:
      - "rabbitdb2:/var/lib/rabbitmq"
  rabbitmq3:
    deploy:
      placement:
        constraints:
          - "node.hostname == <Swarm node hostname3>"
    hostname: rabbit_node3.isnet
    image: "sciencelogic/is-rabbit:3.7.14-3"
    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node3.isnet
    volumes:
      - "rabbitdb3:/var/lib/rabbitmq"
  scheduler:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-worker.isnet"
  steprunner:
    environment:
      db_host: "couchbase.isnet,couchbase-worker2.isnet,couchbase-worker.isnet"
version: "3.4"
volumes:
  rabbitdb2:
  rabbitdb3:
```

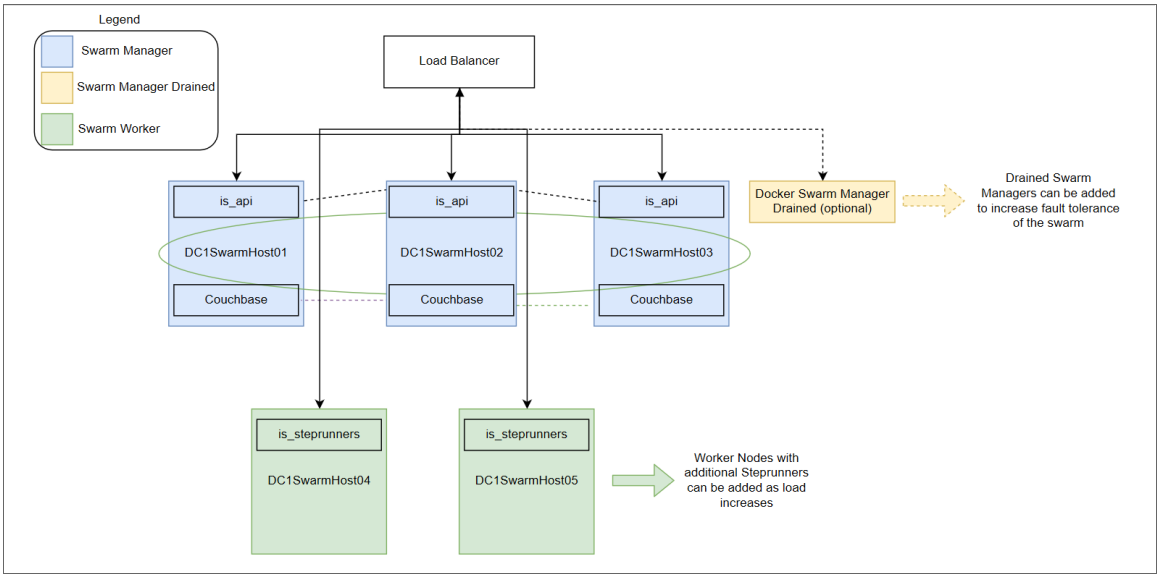
# 3+ Node Cluster with Separate Workers (4 or More Nodes)

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:
    ...
```

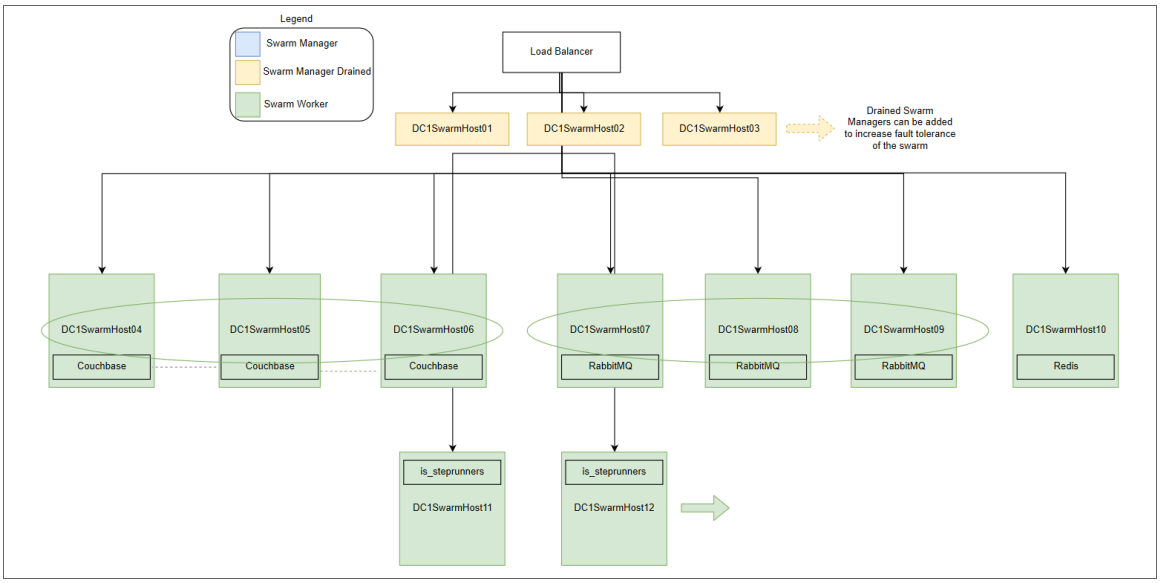
> **NOTE:** The `--max-replica-per-node` option is available with docker-compose 3.8 or later. Add `version: '3.8'` at the start of the **docker-compose** file to ensure compatibility.

# 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)*.

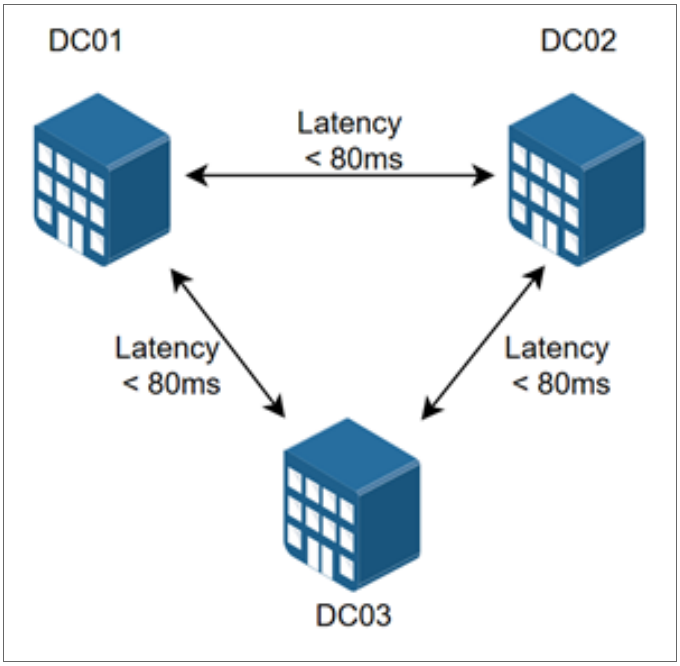
Next, add the additional three nodes to the cluster as managers, and drain them of all services (see *Using Drained Managers to Maintain Swarm Health*). Promote the drained nodes to Swarm Managers, and make all other nodes workers.

## Additional Deployment Options

The following diagrams show additional High Availability deployment architectures that are supported for PowerFlow.

### Cross-Data Center Swarm Configuration

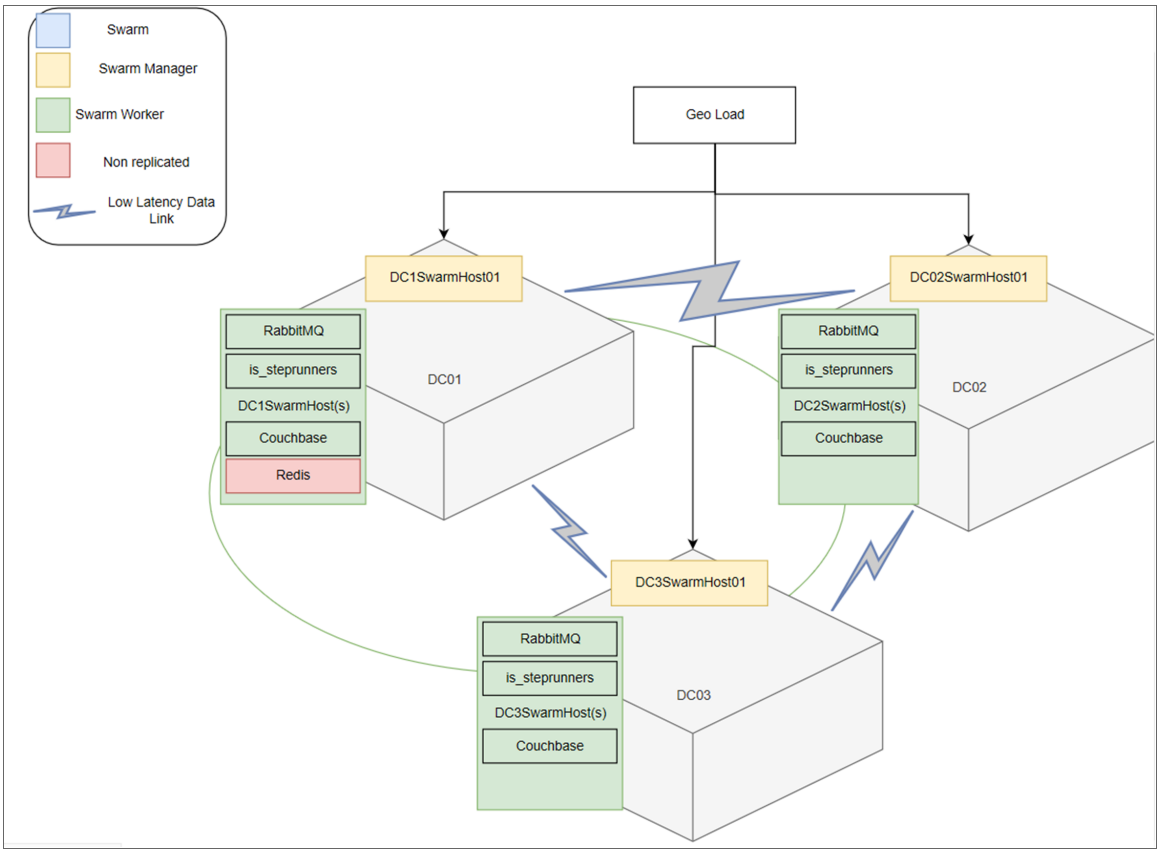
Docker Swarm requires three data centers to maintain quorum of the swarm in the event of a full data center outage. Each data center must have a low-latency connection between the data centers.




**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. The default user interface login is *guest/guest*.

## 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 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 have to run the following commands:
>
> ```
> firewall-cmd --add-service docker-swarm --permanent
> firewall-cmd --reload
> ```

> **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
```

# 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 <PowerFlow_host1>  <username>:<password> --host <PowerFlow_host2>
<username>:<password> --host <PowerFlow_host3>  <username>:<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 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:

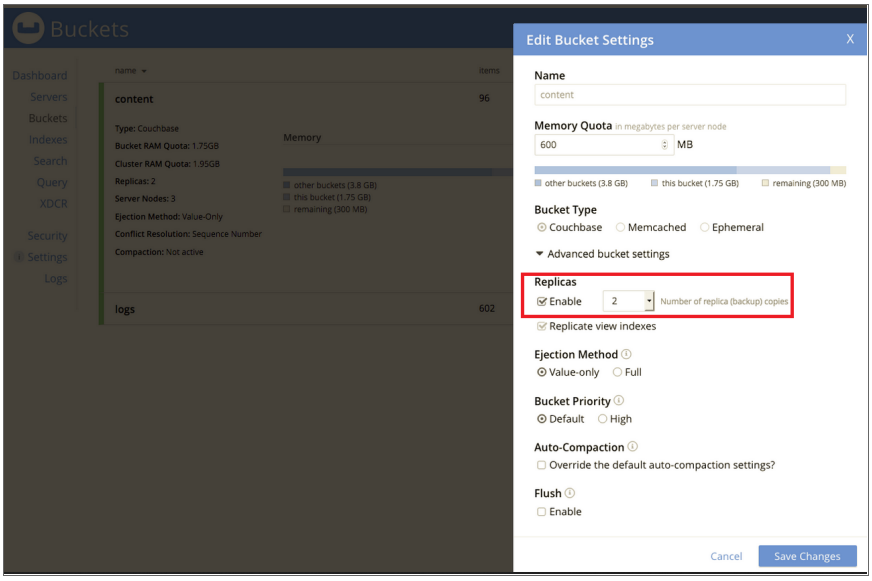
```
docker stack deploy -c <location of compose file> iservices
```

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:

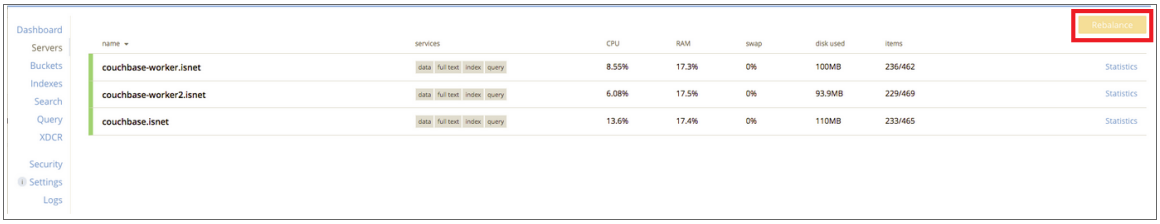
```
docker stack deploy -c <location of compose file> iservices
```

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.

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-worker.isnet

  scheduler:
    environment:
      db_host: couchbase.isnet,couchbase-worker2.isnet,couchbase-worker.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/is-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/is-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/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 == <swarm node hostname>
  rabbitmq2:
     image: sciencelogic/is-rabbit:3.7.7-1
     hostname: rabbit_node2.isnet
```

```
    volumes:
      - "rabbitdb2:/var/lib/rabbitmq"

    networks:
      isnet:
        aliases:
          - rabbit
          - rabbit_node2.isnet
    deploy:
      placement:
        constraints:
          - node.hostname == <swarm node hostname>
rabbitmq3:
    image: sciencelogic/is-rabbit:3.7.7-1
    hostname: rabbit_node3.isnet
    volumes:
      - "rabbitdb3:/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-worker2.isnet'

pypiserver:
    image: sciencelogic/is-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:
  rabbitdb2:
  rabbitdb3:
  devpi:
configs:
  dex_config:
    file: /etc/iservices/dexConfiguration.yaml
```

# Scaling iservices-contentapi

To scale out the iservices-contentapi to distribute the service across the three nodes, run the following command:

```
docker service scale iservices-contentapi=3
```

# 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 (**https://*<IP of PowerFlow>*:8091**) 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:

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
   ```
3. Remove any failed manager nodes from the cluster by running the following Docker commands:

   ```
   docker swarm init --force-new-cluster
   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:

```
docker exec -i -t <container id> /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`

7. One of the previous commands will show a failed node. Copy the IP address and port number of the failed node for step 7.

8. Use the currently running cluster and the failed node's IP address and port to run the following command to failover:

```
couchbase-cli failover -c <operating-couchbase-worker> -u isadmin -p <password>
--server-failover <isnet name: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 -u isadmin -p <password> --server-
failover couchbase.isnet:8091 --force
```

> **NOTE:** If the surviving node is couchbase-worker2 and the contentapi is in a waiting state, restart couchbase-worker2 to reset the connection and resolve the API waiting. Run the following command: `docker service update --force iservices_couchbase-worker2`

9. Rebalance the cluster using the functioning container name:

```
couchbase-cli rebalance -c <operating-couchbase-worker> -u isadmin -p
<password>
```

10. In the unlikely event that a failover occurs and no queries can be performed, validate that the indexes exist, and if not, rebuild them. To rebuild the primary indexes, run the following commands:

```
cbq -u isadmin
CREATE PRIMARY INDEX ON content;
CREATE PRIMARY INDEX ON logs;
```

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

## Recovering a Docker Swarm Node

To recover a Docker Swarm node:

1. Restart the node.
2. If manual failover actions were taken while this node was offline, be sure to force the node to leave the swarm now that the node is back online by running the following command:

   ```
   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).

If the surviving node is a worker node, you will need to update **docker-compose.yml** to restore the node. The following updates to **docker-compose.yml** ensure that the **couchbase** service is told to join the node that is currently up and running:

```
couchbase:
  ...
  environment:
    JOIN_ON: "couchbase-worker2"
```

> NOTE: This process defaults to couchbase-worker. As a result, this update is only needed when the only surviving node is couchbase-worker2, couchbase-worker3, couchbase-worker4, and so on. Also, this process only relates to re-clustering nodes that have been removed.

To restore the failed Couchbase node:

1. Log in to the node where the failed Couchbase cluster node was pinned.
2. Run *one* of the following commands, depending on the Couchbase node being recovered:
   - `docker service scale iservices_couchbase=0`
   - `docker service scale iservices_couchbase-worker=0`

3. If the Docker Swarm node was restored and not rebuilt, remove files from the old container:

```
rm -rf /var/data/couchbase/*
docker service scale iservices_couchbase=1
```

A new node is added to Couchbase that will automatically re-balance the cluster after it is added.

> **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 s**healthcheck** and **autoheal** action with the **powerflowcontrol** (pfctl) command-line utility.

4. 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 Synchronization PowerPacks 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 Synchronization PowerPacks.

# Additional Configuration Information

## 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 over TLS

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.

# 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:

```
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 Network Alias is incorrect

If you experience issues with the iservices_contentapi container, the Alias IP might be incorrect.

To address this issue, run the following commands on the relevant node:

```
docker service scale iservices_contentapi=0
docker service scale iservices_contentapi=1 (or another number as needed)
```

> **NOTE:** This issue was addressed in the **docker-ce 18.06** version of Docker, which is required for version 1.8.0 of PowerFlow.

## 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 --permanant
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 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 appendix covers the following topics:

# 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 IS 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 IS 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

By default RabbitMQ contains the default credentials *guest/guest*.PowerFlow uses these credentials by default 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, or if you are using credentials other than *guest/guest* to authenticate, 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 a Custom Configuration Object

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.

## 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
  ...
```

## Dedicating Queues Per Integration or Customer

Dedicating a separate queue for each customer is beneficial in 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.

**Example 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:
    - 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 as it's expecting a very heavy load
    - steprunner-acmequeue-catchup will get 3 workers as it's not often that this will run

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.

After these changes have been made, your docker-compose entries for the new steprunners should look similar to the following:

```
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:

| Worker Name |
| --- |
| celery@acme-queue-ikb81pi54ver91sxoysltcgbi |
| celery@acme-queue-9bkuebtrnfi349krlw8jc0tn6 |
| celery@acme-queue-koo0bg18bpsw4uvc2iz2l0vf9 |
| celery@acme-queue-gilsccrfx1giosh14pye73ui8 |
| celery@acme- queue-u5kbd5m977wjfdcbofhl230x3 |

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:

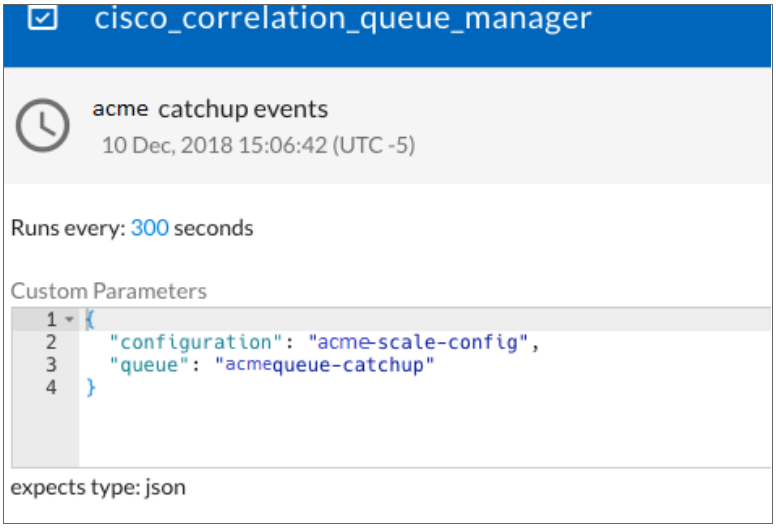
| Name | Messages |
| --- | --- |
| celery | 0 |
| acmequeue | 886 |
| acme queue-catchup | 0 |
| priority.high | 0 |

## 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 "params" override available in the scheduler. Below is an example of the scheduled integration which 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 Automations 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'

# 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. By default, PowerFlow version 1.8.1 and later 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 *journalctl –no-page |grep docker |grep err* logs.

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

## Example of a PowerFlow 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": "<arecord>.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": "${config.em7_db_password}"
  },
  {
    "encrypted": false,
    "name": "em7_user",
    "value": "<username>"
  },
  {
    "encrypted": false,
    "name": "em7_db_user",
    "value": "root"
  },
  {
    "encrypted": false,
    "name": "em7_db_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "snow_user",
    "value": "<username>"
  },
  {
    "encrypted": true,
    "name": "snow_password",
    "value": "<password>"
  },
  {
    "encrypted": false,
    "name": "Domain_Credentials",
    "value": {
      "c9818d2c4a3623120162443385l894bb": {
        "password": "3m7Admin!",
        "user": "is4DomainUser2"
      }
    }
  },
  {
    "name": "region",
    "value": "ACMEScaleStack"
  },
  {
    "encrypted": false,
    "name": "em7_id",
    "value": "${config.region}"
  },
  {
    "encrypted": false,
    "name": "generate_report",
    "value": "true"
  }
```

```
]
```

## Example of a Schedule Configuration

```
[
  {
    "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 *1.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 *1.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.

# 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.<br>**Impact**:<br>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*.<br>**To prevent:**<br>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.<br>Log in to the RabbitMQ admin user interface, which displays a message similar to "RabbitMQ experienced a network partition and the cluster is paused".<br>**Impact**:<br>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)<br>**Note**: ScienceLogic sets the *autoheal* policy by default | Handle the split-brain partition state and *resynchronize your RabbitMQ queues*.<br>**Note**: This is enabled by default.<br>**To prevent:**<br>Set a partition handling policy. See the *Resiliency* section for more information. |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| RabbitMQ, ***continued*** | | Execing into the RabbitMQ container and running **rabbitmqcli cluster-status** shows nodes in a partition state like the following:<br>`[{nodes,`<br>`[{disc,`<br>`['rabbit@rabbit_node1.isnet','rabbit@rabbit_node2.isnet', 'rabbit@rabbit_node3.isnet','rabbit@rabbit_node4.isnet', 'rabbit@rabbit_node5.isnet','rabbit@rabbit_node6.isnet']}]},`<br>`{running_nodes, ['rabbit@rabbit_node4.isnet']},`<br>`{cluster_name,<<"rabbit@rabbit_node1">>},`<br>`{partitions,`<br>`[{'rabbit@rabbit_node4.isnet', ['rabbit@rabbit_node1.isnet','rabbit@rabbit_node2.isnet', 'rabbit@rabbit_node3.isnet','rabbit@rabbit_node5.isnet', 'rabbit@rabbit_node6.isnet']}]},`<br>`{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.<br>**Impact**:<br>One queue's workers will not be processing. | Multi-node failure in RabbitMQ, when it loses majority and can not failover.<br>Queues go out of sync because of broken swarm. | ***Recreate queues*** for the particular worker.<br>***Resynchronize queues***.<br>**To prevent**:<br>Deploy enough nodes to ensure quorum for failover. |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| Couchbase | Couchbase node is unable to restart due to indexer error. | This issue can be monitored in the Couchbase logs:<br>`Service 'indexer' exited with status 134. Restarting. Messages:`<br>`sync.runtime_Semacquire (0xc4236dd33c)`<br>**Impact**:<br>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.<br>**To prevent:**<br>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".<br>**Impact**:<br>Couchbase nodes cannot rebalance and provide even replication. | Network issues: missing firewall rules or blocked ports.<br>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.<br>Reset the Docker swarm network status by *electing a new swarm leader.*<br>**To prevent:**<br>Validate the firewall rules before deployment.<br>*Use drained managers to maintain swarm* |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| 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".<br>**Impact**:<br>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 .<br>Validate couchbase settings, ensure that the proper aliases, hostname, and environment variables are set.<br>Stale docker network. | Validate the deployment configuration and network settings of your **docker-compose**. Redeploy with valid settings.<br>In the event of a swarm failure, or stale swarm network, reset the Docker swarm network status by *electing a new swarm leader.*<br>**To prevent:**<br>Validate hostnames, aliases, and environment settings before deployment.<br>*Use drained managers to maintain swarm* |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| 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.<br>**Impact**:<br>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`<br>You can also remove the `--max-tasks-per-child` setting in the steprunners. |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| 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.<br>**Impact**:<br>The node is not running anything. | One of the following situations could cause this issue:<br>Invalid label deployment configuration.<br>The node does not have the containers you are telling it to deploy.<br>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.<br>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.<br>Check the node status for errors:<br><br>`docker node ls`<br>**To prevent:**<br>Validate your configuration before deploying. |

| Tool | Issue | Symptoms | Cause | Solution |
|---|---|---|---|---|
| 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.<br>The system logs indicate an error like: `"error="rpc error: code = DeadlineExceeded desc = context deadline exceeded"`<br>**Impact**:<br>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.<br>**To prevent**:<br>*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:<br>Indexes do not exist on a particular Couchbase node.<br>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.<br>The indexer can't keep up to a large number of requests, and Couchbase requires additional resources to service the requests. | Solutions:<br>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<br>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

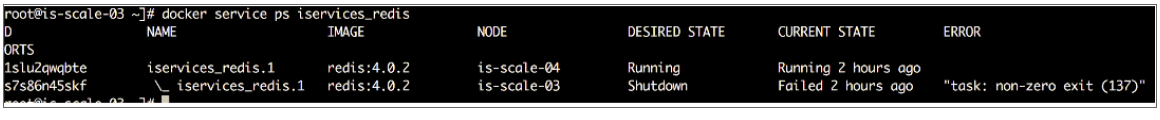
### Step 1: Obtain the ID of the failed container for the service

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
```



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

### Backup

1. Exec into the Couchbase container.
2. Run the following command to back up the data. You can add optional flags to only back up the content bucket as the logs bucket can be large.

   ```
   cbbackup http://couchbase.isnet:8091 /opt/couchbase/var/backup -u [user] -p
   [password] -x data_only=1
   ```
3. Exit the Couchbase shell and then copy the backup file in **/var/data/couchbase/backup** to a safe location, such as **/home/isadmin**.

### 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.

## Perform 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 fs 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>`.

# Prepare the Systems

### Install the new RPM

The first step of upgrading is to install the new RPM on all systems in the stack. 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 installing the RPM which 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:

```
rpm -Uvh <new-rpm-file>
```

### Compare compose file changes and resolve differences

After the RPM is installed, you will notice a new docker-compose file is placed at **/etc/iservices/scripts/docker-compose.yml**. 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 <old_compose_with_small_changes> iservices
   ```

3. Monitor the update, and wait until all services are up and running before proceeding.

**Example image definition of this upgrade group:**

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:1.8.1
  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1
  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1
  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3
  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:1.8.1
  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-1.8.3
  steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
  couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1
  steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```

### 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.

### Flower Dashboard

Due to a known issue addressed in version 1.8.3 of PowerFlow, the Flower Dashboard might not display any workers. Flower eventually picks up the new workers when they are restarted in the worker group. If this is a concern, you can perform the Flower upgrade in the same group as the workers.

## Upgrade Core Services (Rabbit 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.

### Special GUI consideration with 1.8.3

In the upgrade to version 1.8.3 of PowerFlow, the Couchbase and RabbitMQ user interface ports will be exposed through PowerFlow user interface with HTTPS. To ensure there is no port conflict between services and PowerFlow user interface, ensure that the Couchbase and RabbitMQ user interface port mappings are removed or modified from the default (8091) admin port. To avoid conflicts, make sure the new PowerFlow user interface definition does not conflict with the Couchbase or RabbitMQ definitions.

- Modifying the port mapping (exposing to a different port than 8091) means it will still be exposed via HTTP through a port other than 8091 until the PowerFlow user interface is upgraded, at which point that port can then be closed.
- Removing the port means no port mapping will be defined to 8091 for the Couchbase service, and the Couchbase user interface will be inaccessible until the PowerFlow user interface is updated.

The PowerFlow user interface will not update until all port conflicts are resolved. You can upgrade the PowerFlow user interface at any time after this has been done, but be sure to first review the *Update the GUI* topic, below.

**NOTE:** You can manually remove port mappings from a service with the following command, though the command will restart the service: `docker service update --publish-rm published=8091,target=8091 iservices_couchbase`

### Example docker-compose with images and JOIN_ON for updating the first node:

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:1.8.1
  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
    environment:
      JOIN_ON: "couchbase-worker2"
  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1
  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3
  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-1.8.3
  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-1.8.3
  steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
  couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:1.8.1
  steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```

### 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

You can update the GUI service along with any other group, but due to the port mapping changes in version 1.8.3 of PowerFlow, you should update this service *after* the databases and RabbitMQ nodes have been updated, and their port mappings no longer conflict.

Since 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.

**Example docker-compose definition with one of two worker nodes and contentapi updated**:

```
services:
  contentapi:
    image: repository.auto.sciencelogic.local:5000/is-api:hotfix-1.8.3
    deploy:
        replicas: 3
  couchbase:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
    environment:
      JOIN_ON: "couchbase-worker2"
  couchbase-worker:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
  flower:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3
  gui:
    image: repository.auto.sciencelogic.local:5000/is-gui:hotfix-1.8.3
  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-1.8.3
  steprunner:
    image: repository.auto.sciencelogic.local:5000/is-worker:hotfix-1.8.3
  couchbase-worker2:
    image: repository.auto.sciencelogic.local:5000/is-couchbase:hotfix-1.8.3
  steprunner2:
    image: repository.auto.sciencelogic.local:5000/is-worker:1.8.1
```

## 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

800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010