Monitoring Kubernetes

Kubernetes PowerPack version 103

Table of Contents

Chapter

1

Introduction

Overview

This manual describes how to monitor Kubernetes clusters in SL1 using the *Kubernetes* PowerPack.

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 (•••).

The following sections provide an overview of the Kubernetes platform and the *Kubernetes* PowerPack:

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

What is Kubernetes?

Kubernetes is an open-source platform that automates the deployment, scaling, and operation of application containers. The Kubernetes platform is deployed in ***clusters*** that consist of compute nodes. These nodes can take on the following roles:

- ***Master***. The master runs on one of the physical computers in the cluster and manages the cluster. It oversees all cluster activities such as scheduling, maintaining, and scaling applications, as well as executing updates.
- ***Nodes***. Nodes are physical computers or virtual machines (VMs) that run applications and perform other tasks in a Kubernetes cluster. Nodes are controlled by the master.

Kubernetes manages containers through a series of objects that represent your system, including ***Pods***, ***Services***, ***Volumes***, and ***Namespaces***. Kubernetes also uses a series of Controller objects that provide additional features and functionality; these include ***ReplicaSets***, ***Deployments***, ***StatefulSets***, ***DaemonSets***, ***Jobs***, ***CronJobs***, and ***IngressControllers***.

NOTE: For more information about these Kubernetes concepts, consult the Kubernetes documentation.

What Does the Kubernetes PowerPack Monitor?

The *Kubernetes* PowerPack enables you to monitor Kubernetes clusters, nodes, and objects.

NOTE: The *Kubernetes* PowerPack can leverage the capabilities of the *Linux Base Pack* PowerPack to provide a comprehensive view of the Kubernetes cluster, including its underlying hardware. If you would like to do this, you must install and run the most recent version of this PowerPack before you install the *Kubernetes* PowerPack. For more information about using this PowerPack, see the ***Monitoring Linux*** manual.

NOTE: The *Kubernetes* PowerPack has been validated on the Cloud Native Computing Foundation (CNCF) version of Kubernetes.

The *Kubernetes* PowerPack includes the following features:

- Dynamic Applications that perform the following tasks:
  - Discover and monitor the Kubernetes cluster, nodes, and objects
  - Collect and present data about the underlying Linux operating system

- Device Classes for each of the Kubernetes devices the *Kubernetes* PowerPack models
- Event Policies and corresponding alerts that are triggered when Kubernetes devices meet certain status criteria
- A dashboard and widget that you must use to create Credentials for discovering Kubernetes devices
- An SSH/Key Credential that the Kubernetes Token Entry Dashboard can use as a template for creating additional SSH/Key Credentials for monitoring Kubernetes clusters
- Run Book Action and Automation policies do the following:
  - Automatically create Kubernetes clusters whenever SL1 discovers a Kubernetes host
  - Align Dynamic Applications from the *Linux Base Pack* PowerPack to Kubernetes nodes and report back to the ScienceLogic Data Collector or All-in-One Appliance if the Dynamic Applications were successfully aligned
  - Ensure that Namespaces (and their children) have a 1-hour vanishing timer, to properly reflect topology changes

NOTE: You must not edit the SSH/Key Credential that is included in the *Kubernetes* PowerPack.

Installing the Kubernetes PowerPack

NOTE: If you would like to leverage the capabilities of the*Linux Base Pack* PowerPack with the *Kubernetes* PowerPack, you must first import and install the *Linux Base Pack* PowerPack version 103 before installing the *Kubernetes* PowerPack.

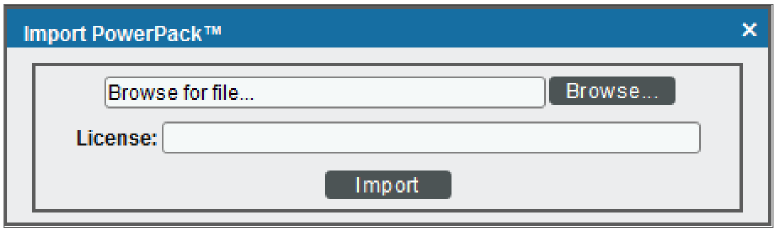
Before completing the steps in this manual, you must import and install the latest version of the *Kubernetes* PowerPack.

TIP: By default, installing a new version of a PowerPack overwrites all content from a previous version of that PowerPack that has already been installed on the target system. You can use the ***Enable Selective PowerPack Field Protection*** setting in the **Behavior Settings** page (System > Settings > Behavior) to prevent new PowerPacks from overwriting local changes for some commonly customized fields. (For more information, see the ***System Administration*** manual.)

To download and install a PowerPack:

1. Download the PowerPack from the ScienceLogic Support Site.
2. Go to the **PowerPack Manager** page (System > Manage > PowerPacks).
3. In the **PowerPack Manager** page, click the **[Actions]** button, then select *Import PowerPack*.

4. The **Import PowerPack** dialog box appears:



5. Click the **[Browse]** button and navigate to the PowerPack file.
6. When the **PowerPack Installer** modal appears, click the **[Install]** button to install the PowerPack.

NOTE: If you exit the **PowerPack Installer** modal without installing the imported PowerPack, the imported PowerPack will not appear in the **PowerPack Manager** page. However, the imported PowerPack will appear in the **Imported PowerPacks** modal. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Chapter

2

Configuration and Discovery

Overview

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 (⋯).

The following sections describe how to configure and discover Kubernetes clusters for monitoring by SL1 using the *Kubernetes* PowerPack:

Prerequisites for Monitoring Kubernetes Clusters

Before you can monitor Kubernetes clusters using the *Kubernetes* PowerPack, you must first do the following:

1. If you will be using Dynamic Applications from the *Linux Base Pack* PowerPack, import and install version 103.
2. Create a Kubernetes service account that SL1 can use to communicate with the Kubernetes API. This service account must have the minimum permissions set in the ***Required Permissions for the Service Account Token*** section.
3. Extract the service account token.
4. Ensure that cURL 7.40 or greater is installed on all Kubernetes nodes that you want to monitor.
5. Configure SSH credentials on the Kubernetes nodes. These credentials must be the same on all nodes, and are used to retrieve data from the underlying Linux OS.

For more information about any of these steps, see https://kubernetes.io/docs/reference/access-authn-authz/rbac/.

Required Permissions for the Service Account Token

The minimum required permissions are required for the service account token:

~~~
```

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:
  name: cluster-limited

rules:

- apiGroups:

- '*'

resources:

- nodes

- pods

- replicationcontrollers

- deployments

- statefulsets

- replicasets
~~~

~~~
- daemonsets
- cronjobs
- jobs
- componentstatuses
- namespaces
- persistentvolumes
- persistentvolumeclaims
- services
- events
- ingresses
- horizontalpodautoscalers
verbs:
- get
- list
- watch
```
~~~

Creating Credentials for Kubernetes Clusters

Unlike other PowerPacks, you ***must not*** create a credential for Kubernetes using the **Credential Management** page (System > Manage > Credentials).

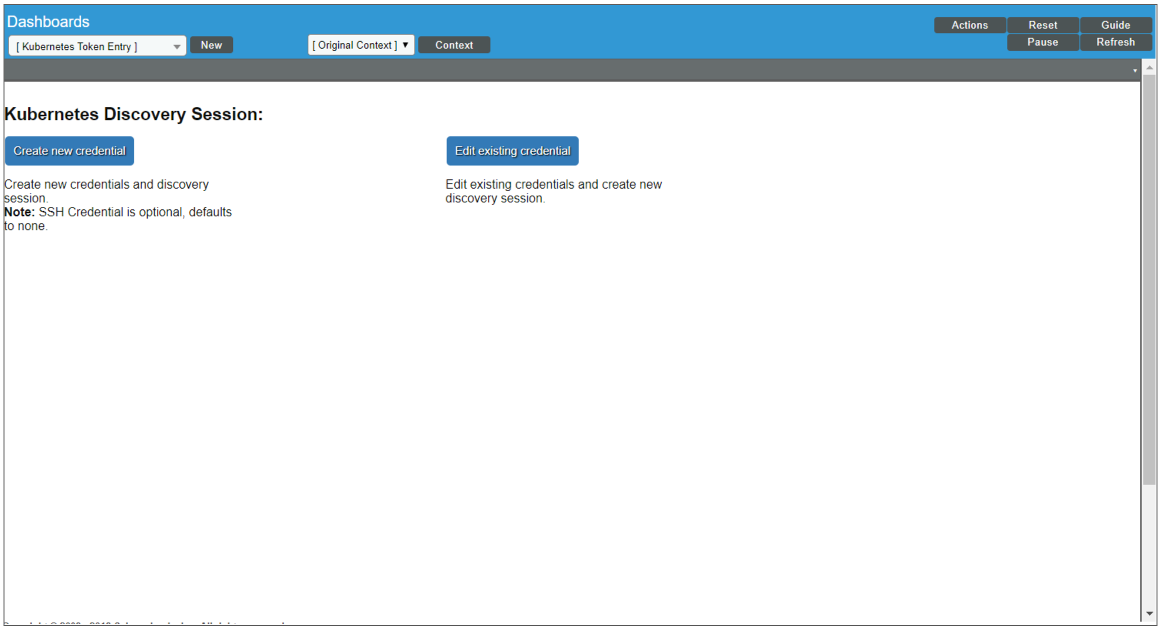
Instead, you must use the **Kubernetes Token Entry** dashboard that is included in the *Kubernetes* PowerPack. This dashboard automatically creates the following credentials based on your input:

- ***A master SSH/Key Credential***. This credential enables SL1 to communicate with the Kubernetes API so that it can monitor the Kubernetes master. It includes a host IP address, port number, and the service account token.
- ***A SOAP/XML Credential***. This credential includes HTTP headers that enable you to specify the Kubernetes topology that you want SL1 to discover.
- ***A node SSH/Key Credential***. This credential enables SL1 to monitor and run Dynamic Applications on Kubernetes nodes.

To create credentials for Kubernetes clusters:

1. Click the **Dashboards tab**. In the drop-down field in the upper-left corner of the page, select *Kubernetes Token Entry*.

2. On the **Kubernetes Token Entry** dashboard page, click **[Create new credential]**.



Kubernetes Discovery Session:

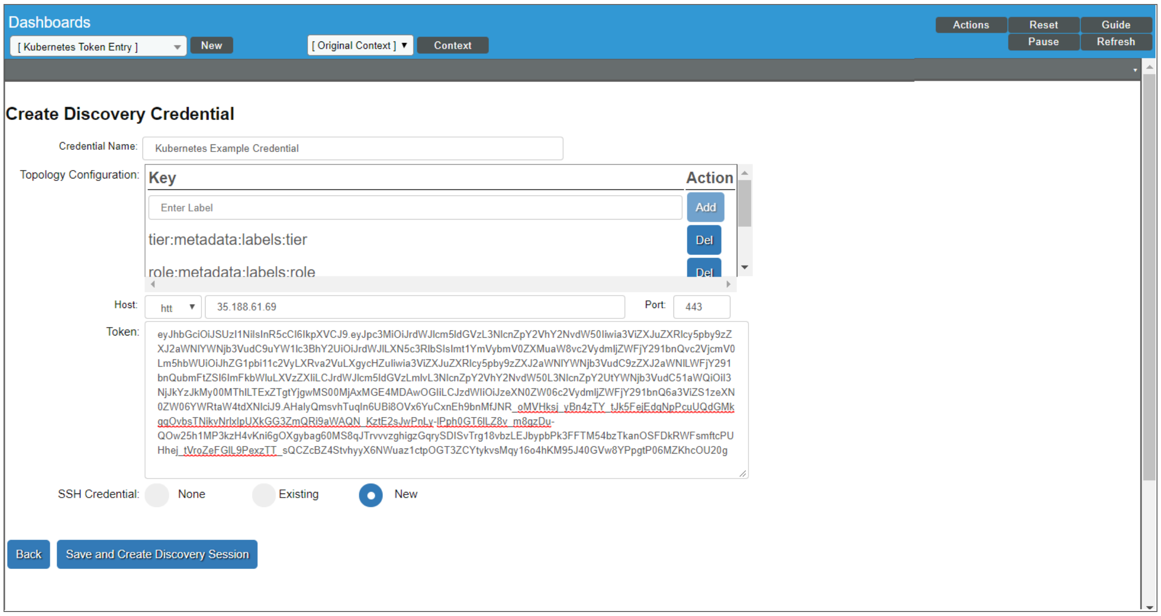
Create new credential

Create new credentials and discovery session.
Note: SSH Credential is optional, defaults to none.

Edit existing credential

Edit existing credentials and create new discovery session.

3. On the **Create Discovery Credential** dashboard page:



Create Discovery Credential

Credential Name: Kubernetes Example Credential

Topology Configuration:

Key	Action
Enter Label	Add
tier:metadata:labels:tier	Del
role:metadata:labels:role	Del

Host: htt 35.188.61.69 Port: 443

Token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZ
XJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0
Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXgycHZuIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291
bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI3
NjJkYzJkMy00MThlLTExZTgtYjgwMS00MjAxMGE4MDAwOGIiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN
0ZW06YWRtaW4tdXNlciJ9.AHalyQmsvhTuqIn6UBi8OVx6YuCxnEh9bnMfJNR_oMVHksj_yBn4zTY_tJk5FejEdqNpPcuUQdGMk
gqOvbsTNikvNrlxlpUXkGG3ZmQRi9aWAQN_KztE2sJwPnLy-lPph0GT6lLZ8v_m8gzDu-
QOw25h1MP3kzH4vKni6gOXgybag60MS8qJTrvvvzghigzGqrySDISvTrg18vbzLEJbypbPk3FFTM54bzTkanOSFDkRWFsmftcPU
Hhej_tVroZeFGIL9PexzTT_sQCZcBZ4StvhyyX6NWuaz1ctpOGT3ZCYtykvsMqy16o4hKM95J40GVw8YPpgtP06MZKhcOU20g
```

SSH Credential: ○ None ○ Existing ● New

- In the ***Credential Name*** field, type a name for the Kubernetes master SSH/Key credential.

- In the ***Topology Configuration*** field:
  - In the ***Enter Label*** field, type a topology definition that you want SL1 to use when discovering Kubernetes component devices. (For more information, see the ***Specifying the Kubernetes Topology for Discovery*** section.)
  - Click **[Add]**.
  - Repeat the previous two steps as needed until you have fully defined the discovery topology configuration.

> **NOTE:** Specifying the discovery topology configuration creates a SOAP/XML credential that uses cURL commands for topology discovery.

- In the ***Host*** field, select *https://* if the IP address is secure or *http://* if it is not, and then type the IP address of the Kubernetes cluster.
- In the ***Port*** field, type the IP address port for the Kubernetes cluster.

> **NOTE:** Ports 443 or 8443 are typically used for HTTPS.

> **NOTE:** For steps on how to configure a customized IP port and to edit the snippet code in some Run Book Actions to use that IP port, see the ***Configuring Customized IP Ports*** section.

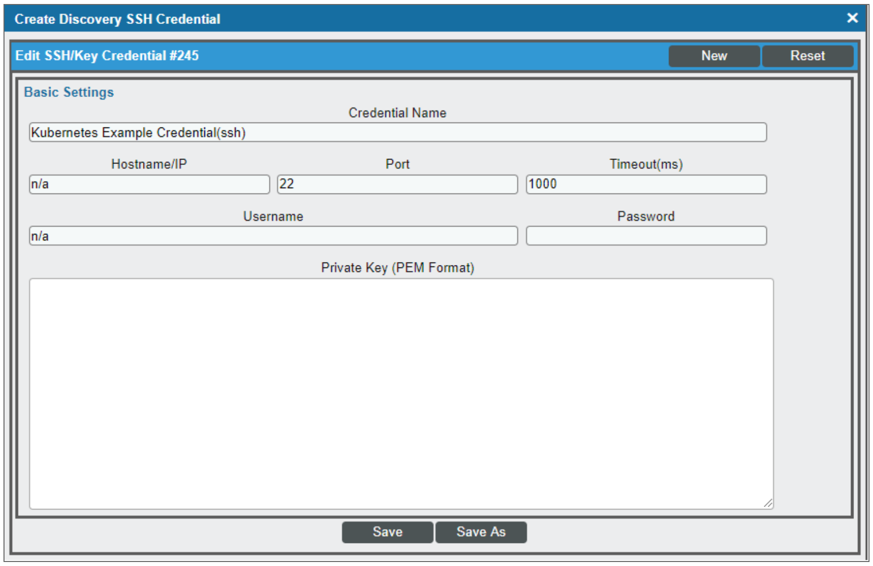
- In the ***Token*** field, paste your Kubernetes service account token.
- In the ***SSH Credential*** field:
  - Select *None* if you do not want to create or use an additional SSH/Key credential to monitor Kubernetes nodes.

> **NOTE:** To fully monitor Kubernetes, a standard SSH/Key credential is required to communicate with the Kubernetes node.

  - Select *Existing* if you want to use an existing SSH/Key credential to monitor Kubernetes nodes, and then select that credential from the ***Existing SSH Key*** drop-down field.
  - Select *New* if you want to create a new SSH/Key credential to monitor Kubernetes nodes.

4. Click the **[Save and Create Discovery Session]** button.
   - If you selected *None* or *Existing* in the ***SSH Credential*** field, then SL1 saves your Kubernetes credentials and creates a new discovery session. Proceed to the ***Discovering a Kubernetes Cluster*** section.
   - If you selected *New* in the ***SSH Credential*** field, then the **Edit SSH/Key Credential** modal page appears. Proceed to step 5.

5. On the **Edit SSH/Key Credential** modal page, make entries in the following fields:



Create Discovery SSH Credential

Edit SSH/Key Credential #245

Basic Settings

Credential Name: Kubernetes Example Credential(ssh)

Hostname/IP: n/a

Port: 22

Timeout(ms): 1000

Username: n/a

Password:

Private Key (PEM Format):

- ***Credential Name***. Defaults to the same credential name as the Kubernetes master SSH/Key credential name that you entered in step 3, followed by "(ssh)".
- ***Hostname/IP***. Type "%D".
- ***Port***. Type the IP port of the Kubernetes nodes.

NOTE: Port 22 is typically used for SSH.

- ***Timeout (ms)***. Type the time, in milliseconds, after which SL1 will stop trying to communicate with the Kubernetes nodes. ScienceLogic recommends setting this field to a value of at least 1,000; however, you can increase the value if you experience high network latency.
- ***Username***. Type the SSH account username.

  If you want to monitor multiple EKS clusters, you must type the AWS EKS token account ID, the region name, and the cluster name in this field in the following format:

  ```
  AWS-EKS-token-account-id:region-name:cluster-name
  ```

  For example:

  ```
  1234567890121:us-east-1:ekscluster
  ```

- ***Password***. Type the password for the SSH account.
- ***Private Key (PEM Format)***. Type the SSH private key that you want SL1 to use to monitor the Kubernetes nodes.

NOTE: Most systems will require ***either*** a password or a private key for authentication.

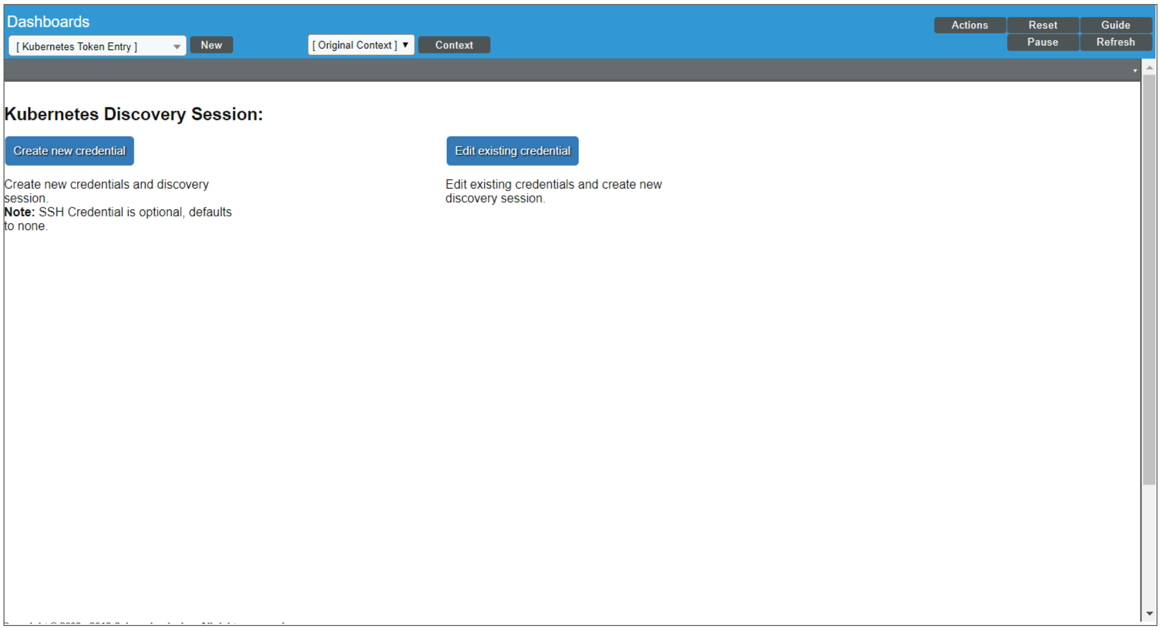
6. Click **[Save]** to save the SSH/Key credential for monitoring Kubernetes nodes. The **Create Discovery Credential** dashboard page appears again.
7. In the ***Existing SSH Key*** drop-down field, select the SSH/Key credential that you created in steps 5 and 6.
8. Click the **[Save and Create Discovery Session]** button.

WARNING: If you created an SSH/Key credential for Kubernetes nodes in steps 5 and 6, you must click the **[Save and Create Discovery Session]** button, as indicated in step 8, even though you already clicked the button once in step 4. Clicking the button the first time saved your Kubernetes master SSH/Key and SOAP/XML credentials; clicking it the second time will link the node SSH/Key credential to the master SSH/Key credential.

Configure the Discovery Session to Exclude Dynamic Applications from the Linux Base Pack

To configure the discovery session to exclude aligning *Linux Base Pack* Dynamic Applications:

1. On the **Kubernetes Token Entry** dashboard page, click **[Create new credential]**.

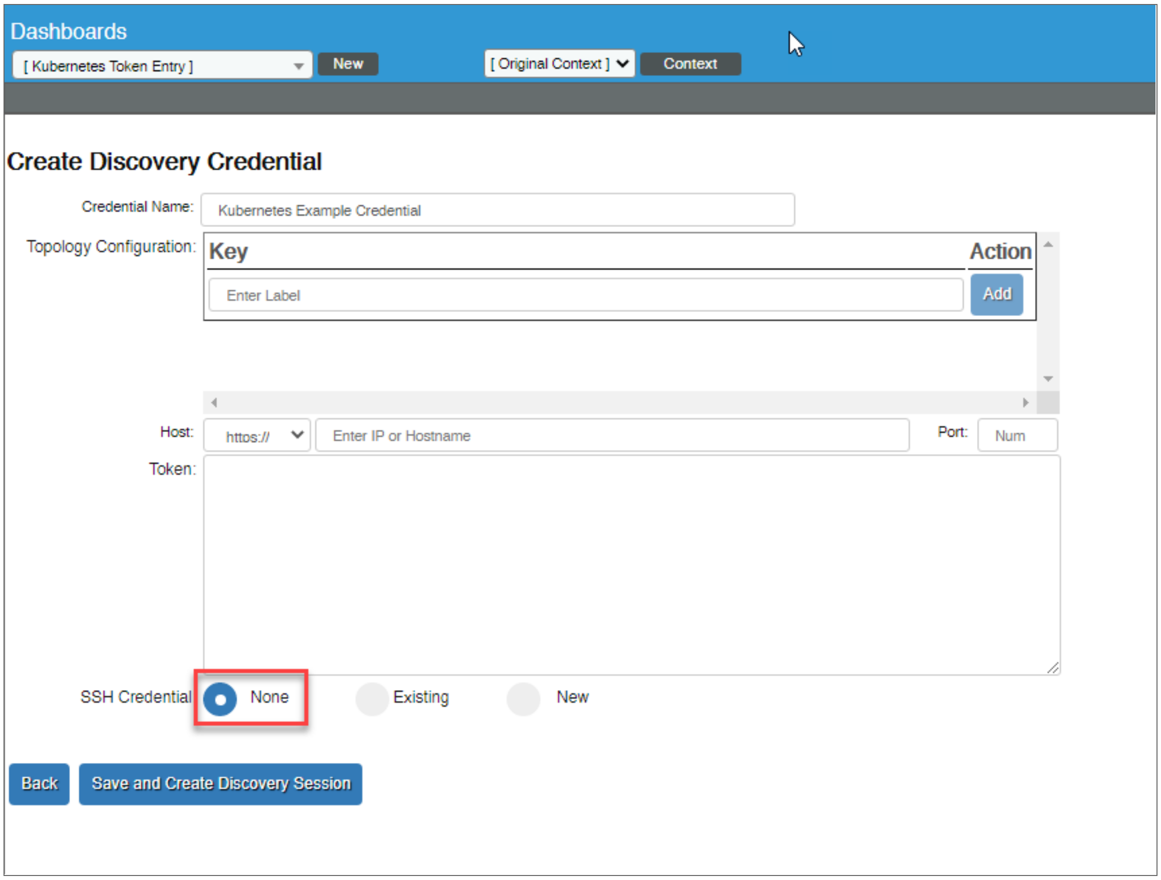


Kubernetes Discovery Session:

Create new credentials and discovery session.
Note: SSH Credential is optional, defaults to none.

Edit existing credentials and create new discovery session.

2. On the **Create Discovery Credential** dashboard page, fill out the fields as described in the *Creating Credentials for Kubernetes Clusters* section, then select **None** in the ***SSH Credential*** field.



Create Discovery Credential

Credential Name: Kubernetes Example Credential

Topology Configuration:

Key	Action
Enter Label	Add

Host: https:// Enter IP or Hostname Port: Num

Token:

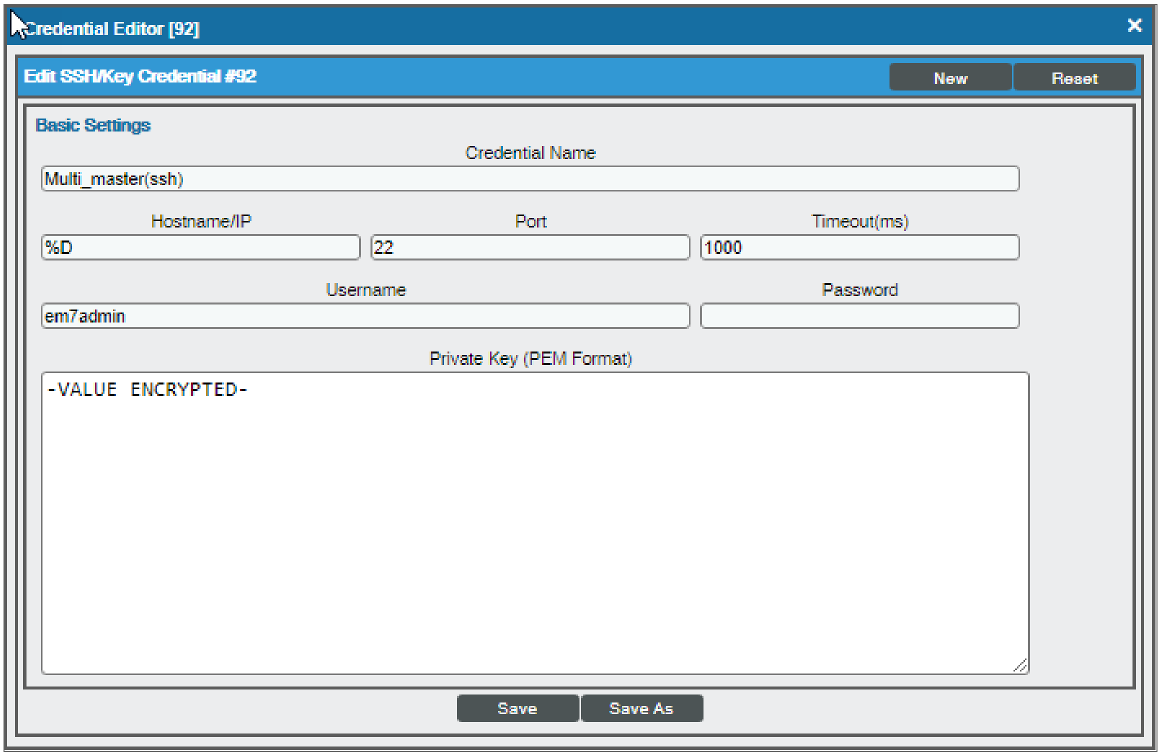
SSH Credential ● None ○ Existing ○ New

4. Click the **[Save and Create Discovery Session]** button.
5. Since you selected *None* in the ***SSH Credential*** field, SL1 saves your Kubernetes credentials and creates a new discovery session. Proceed to the ***Discovering a Kubernetes Cluster*** section to finish the process. The Kubernetes cluster will be discovered and there will not be any *Linux Base Pack* Dynamic Applications aligned.

Enabling Data Collection for Linux Base Pack Dynamic Applications

If you previously disabled auto-aligning the Linux Dynamic Applications and need to monitor a Linux machine that your nodes are deployed on, you can enable and align the *Linux Base Pack* Dynamic Applications by performing the following steps:

1. Create an SSH/Key credential. When naming the credential, use the name of the SSH/Key credential that you have already created for Kubernetes, but add "SSH" in the title. For example, if your credential name is "Multi_master", name it "Multi_master(ssh)":



Credential Name: Multi_master(ssh)

Hostname/IP: %D

Port: 22

Timeout(ms): 1000

Username: em7admin

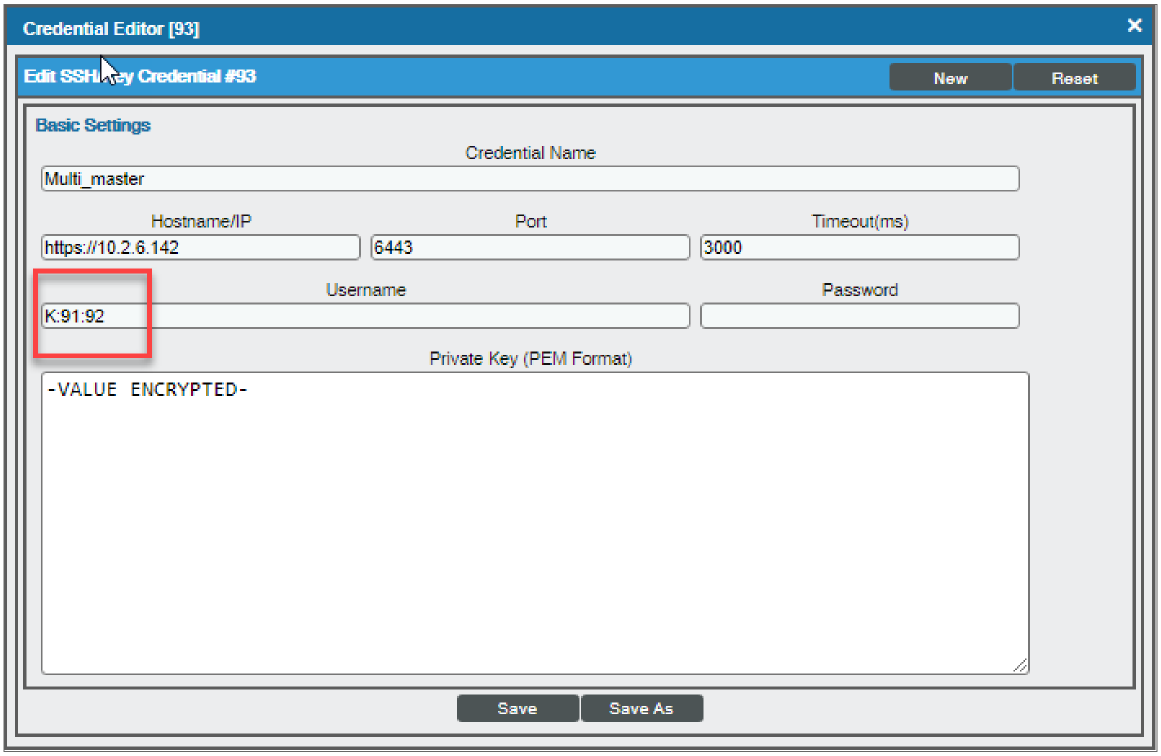
Password:

Private Key (PEM Format): -VALUE ENCRYPTED-

2. Go to the **Kubernetes Token Entry** dashboard page, and click **[Edit existing credential]**.
3. Select the *original credential you created* (in our example, "Multi_master") in the ***Credential*** field.
4. In the ***SSH Credential*** field, select *Existing*.
5. In the ***Existing SSH Key*** field, select the SSH credential that you created in step 1 (in our example, "Multi_master(ssh)").
6. Click **[Save and Create Discovery Session]**.

NOTE: A copy of the original discovery session will appear in the **Discovery Control Panel**, but it will not automatically run. You may delete this discovery session.

7. Locate original Kubernetes credential in the **Credential Management** page (System > Manage > Credentials) and click its wrench icon (🔧).
8. Validate that the Kubernetes credential has been updated by confirming that the ***Username*** field now contains the ID of the new SSH credential:



Credential Name: Multi_master

Hostname/IP: https://10.2.6.142

Port: 6443

Timeout(ms): 3000

Username: K:91:92

Password:

Private Key (PEM Format): -VALUE ENCRYPTED-

9. After about 15 minutes all the *Linux Base Pack* Dynamic Applications will align to the Kubernetes nodes.

If the Linux Base Pack Dynamic Applications are not aligning, you may need to clear the cache on the Data Collector that contains the cluster. To do this, execute the following query on the Data Collector:

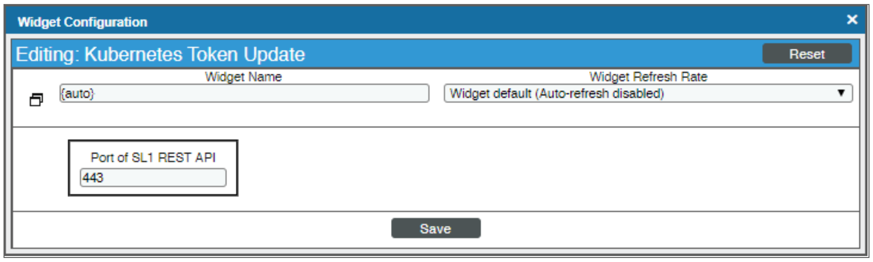
```
DELETE

FROM cache.dynamic_app

WHERE 'key' LIKE 'KUBERNETES_NODE_APP_ALIGN_STATUS_%'
```

Configuring Customized IP Ports

To use a custom IP address port with your Kubernetes cluster, perform the following steps:

1. Go to the **Dashboards** page (System > Customize > Dashboards).
2. In the ***Dashboard Name*** column, type "Kubernetes Token Entry". Click the wrench icon (🔧) for the dashboard to open the **Dashboard Editor** page.

3. In the **Dashboard Editor** page, go to the top-right corner of the widget and click ***Options*** > ***Configure*** to open the **Widget Configuration** window. In the ***Widget Configuration*** window, type your custom port number into the ***Port of SL1 REST API*** field.



Widget Configuration

Editing: Kubernetes Token Update

Widget Name: {auto}

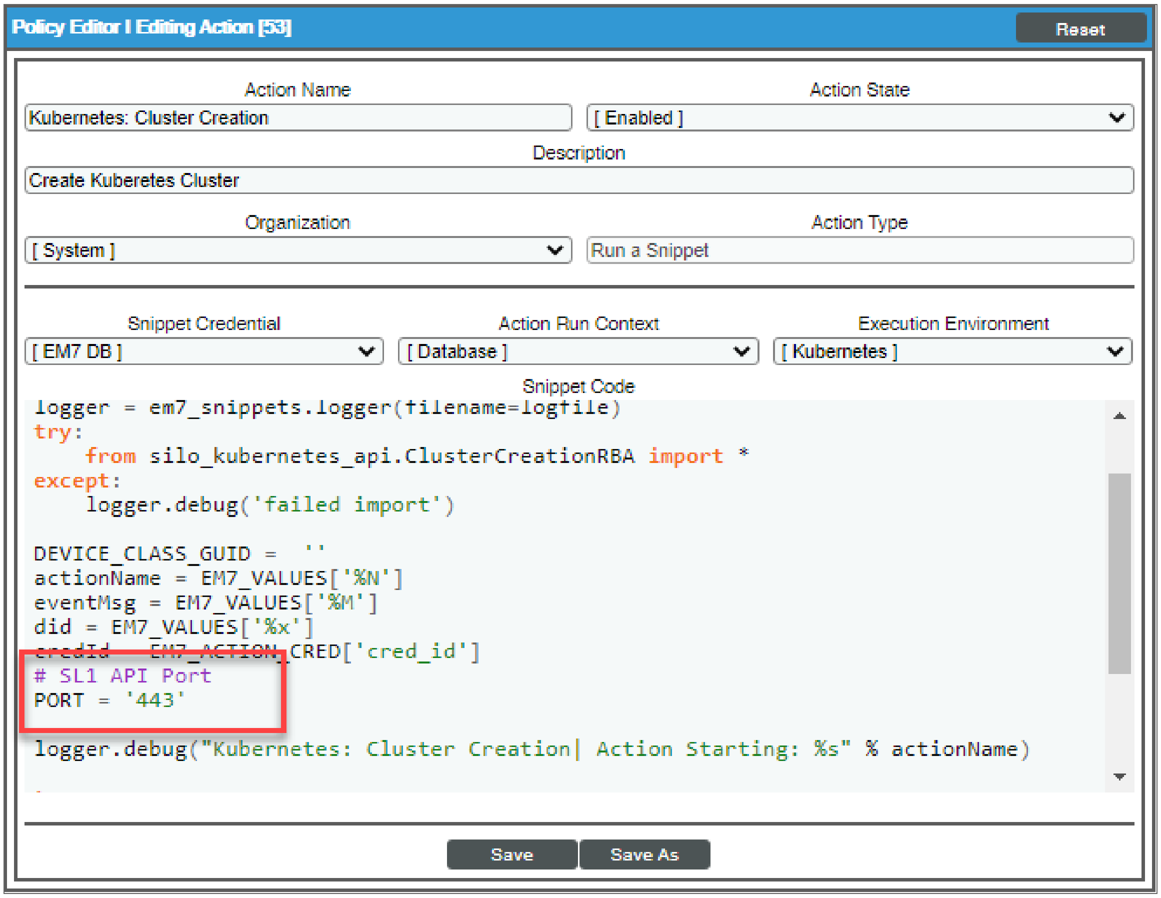
Widget Refresh Rate: Widget default (Auto-refresh disabled)

Port of SL1 REST API: 443

4. Click the **[Save]** button.

If you are using a customized IP port for your Kubernetes cluster, you will need to edit the snippet code in some of the PowerPack's run book actions for discovery to run successfully. To do so:

1. Go to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. Locate the "Kubernetes: Cluster Creation" run book action and click its wrench icon (🔧).
3. In the **Action Policy Editor** window, find the entry for `PORT` in the ***Snippet Code*** field and update the entry to your customized port number.



Policy Editor | Editing Action [53]

Action Name	Action State
Kubernetes: Cluster Creation	[Enabled]

Description: Create Kuberetes Cluster

Organization	Action Type
[System]	Run a Snippet

Snippet Credential	Action Run Context	Execution Environment
[EM7 DB]	[Database]	[Kubernetes]

Snippet Code

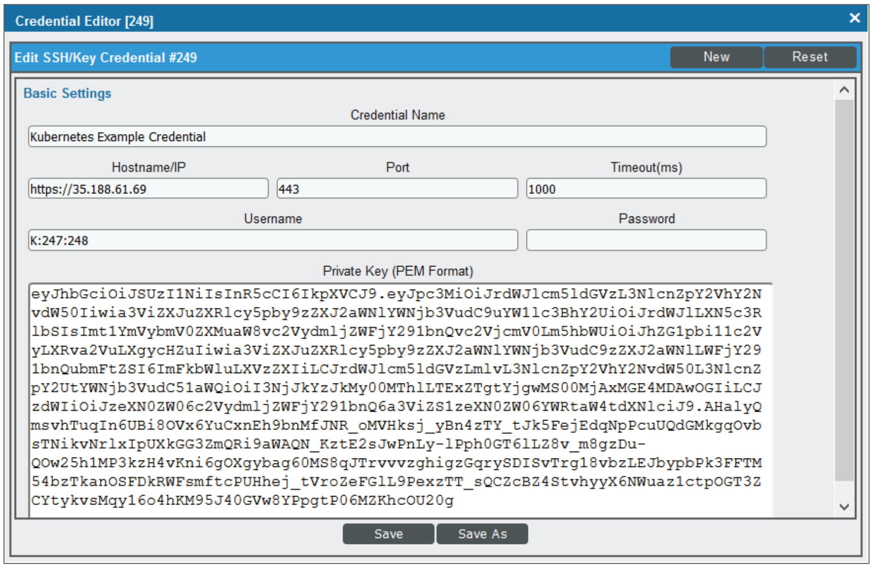
```
logger = em7_snippets.logger(filename=logfile)
try:
    from silo_kubernetes_api.ClusterCreationRBA import *
except:
    logger.debug('failed import')

DEVICE_CLASS_GUID = ''
actionName = EM7_VALUES['%N']
eventMsg = EM7_VALUES['%M']
did = EM7_VALUES['%x']
credId = EM7_ACTION_CRED['cred_id']
# SL1 API Port
PORT = '443'

logger.debug("Kubernetes: Cluster Creation| Action Starting: %s" % actionName)
```

4. Click **[Save]**.
5. Repeat these steps for the following run book actions:
    - Kubernetes: GCP Cluster Update
    - Kubernetes: Node App Alignment
    - Kubernetes: Set Namespace Vanishing Timer

Viewing the Kubernetes Credentials

After you have created the Kubernetes credentials using the **Kubernetes Token Entry** dashboard, you can view the credentials on the **Credential Management** page (System > Manage > Credentials).

Master SSH/Key Credential



Credential Name: Kubernetes Example Credential

Hostname/IP: https://35.188.61.69

Port: 443

Timeout(ms): 1000

Username: K:247:248

Password:

Private Key (PEM Format)

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2N
vdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3R
lbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2V
yLXRva2VuLXgycHZuIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY29
1bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZ
pY2UtYWNjb3VudC51aWQiOiI3NjJkYzJkMy00MThlLTExZTgtYjgwMS00MjAxMGE4MDAwOGIiLCJ
zdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.AHalyQ
msvhTuqIn6UBi8OVx6YuCxnEh9bnMfJNR_oMVHksj_yBn4zTY_tJk5FejEdqNpPcuUQdGMkgqOvb
sTNikvNrlxIpUXkGG3ZmQRi9aWAQN_KztE2sJwPnLy-lPph0GT6lLZ8v_m8gzDu-
QOw25h1MP3kzH4vKni6gOXgybag60MS8qJTrvvvzghigzGqrySDISvTrg18vbzLEJbypbPk3FFTM
54bzTkanOSFDkRWFsmftcPUHhej_tVroZeFGlL9PexzTT_sQCZcBZ4StvhyyX6NWuaz1ctpOGT3Z
CYtykvsMqy16o4hKM95J40GVw8YPpgtP06MZKhcOU20g
```

WARNING: You must not modify the master SSH/Key credential from this page. You must use the **Kubernetes Token Entry** dashboard to edit the master credential.

The master SSH/Key credential will include the ***Credential Name***, ***Hostname/IP***, ***Port***, and ***Private Key*** (service account token) values that you entered on the **Create Discovery Credential** dashboard page.

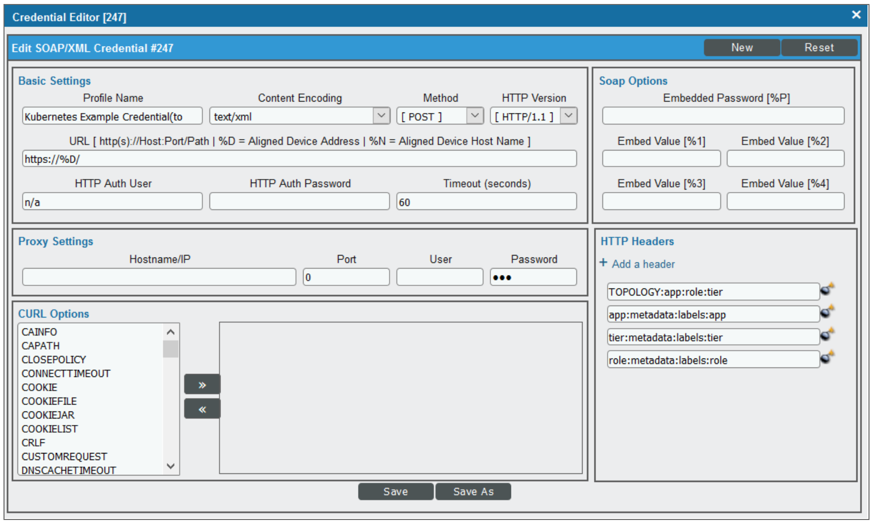
The ***Timeout(ms)*** field displays the default timeout value. You can edit this value if needed.

The ***Username*** field indicates the SOAP/XML credential and optional node SSH/Key credential that are associated with this master SSH/Key credential. The ***Username*** value is auto-generated in the following format:

```
K:SOAP/XML Credential ID #:Node SSH/Key Credential ID #
```

The ***Password*** field should remain blank.

SOAP/XML Credential



Profile Name: Kubernetes Example Credential(to
Content Encoding: text/xml
Method: [POST]
HTTP Version: [HTTP/1.1]
URL [http(s)://Host:Port/Path | %D = Aligned Device Address | %N = Aligned Device Host Name]: https://%D/
HTTP Auth User: n/a
HTTP Auth Password:
Timeout (seconds): 60
Embedded Password [%P]:
Embed Value [%1]:
Embed Value [%2]:
Embed Value [%3]:
Embed Value [%4]:
Hostname/IP:
Port: 0
User:
Password: •••

TOPOLOGY:app:role:tier
app:metadata:labels:app
tier:metadata:labels:tier
role:metadata:labels:role

The SOAP/XML credential will include the ***HTTP Headers*** field values that you entered in the ***Topology Configuration*** field on the **Create Discovery Credential** dashboard page.

The ***Profile Name*** field defaults to the same credential name as the Kubernetes master SSH/Key credential name, followed by "(topo)".

All other fields in this credential use the default values needed for monitoring Kubernetes clusters.

Node SSH/Key Credential

The node SSH/Key credential will include the values that you defined on the **Edit SSH/Key Credential** modal page, as described in the ***Creating Credentials for Kubernetes Clusters*** section.

Specifying the Kubernetes Topology for Discovery

The *Kubernetes* PowerPack utilizes a ***flexible device topology***. This flexible topology enables you to specify the device component map hierarchy that you want SL1 to create when discovering and modeling your Kubernetes devices, rather than using a hierarchy that is pre-defined in the PowerPack.

When ***creating your Kubernetes credentials***, you can specify the device topology that you want to be modeled upon discovery. The topology is based on labels used in Kubernetes.

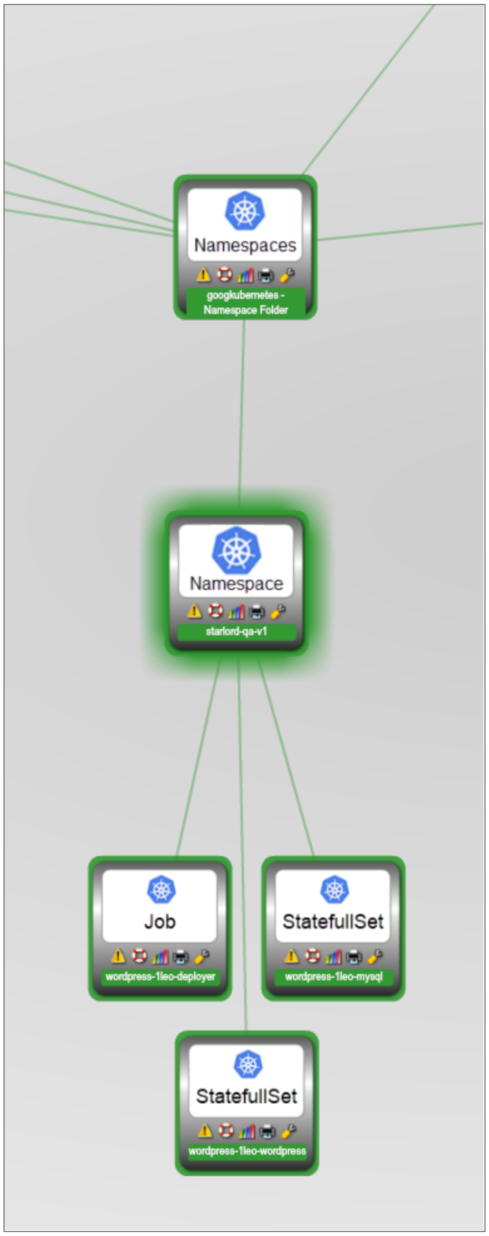
For example, if you want to separate your production and development environments, you could use labels such as "environment=prod" and "environment=dev" in Kubernetes. When discovering your Kubernetes system, SL1 could then use those labels to utilize the following features:

- ***Aggregation***. This enables SL1 to model the aggregation point and create a component device in the platform that represents grouping based on these labels. If just aggregation is required, then the device component map would display a component device for "dev" and another component device for "prod". All of the components with the "dev" and "prod" labels would then appear under those two component devices.
- ***Filtering***. This enables SL1 to create additional components that match a Kubernetes label. So in the case of the environment label, SL1 could selectively model only the production environment. (In this scenario, all of the controllers that do not match the production label would still appear in the device component map under the namespace.)

Example: Defining the Topology

The first step to utilizing the flexible topology is to define the topology and the components that you want to appear on the device component map.

The following example shows an application and its tiers modeled on a device component map ***without*** a defined topology:



To define the Kubernetes topology, you must first type a topology definition into the ***Topology Configuration*** field when *using the Kubernetes Token Entry dashboard to create your Kubernetes discovery credentials*. For example:

TOPOLOGY:Application:Tier

This definition declares that additional components will be created in the device component map to reflect Applications and Application Tiers.

When defining the topology, keep the following important rules in mind:

- "TOPOLOGY" must be capitalized.
- The labels included in your topology definition are used to identify the component devices that will appear on the device component map. These labels ***must*** match a device class's ***Class Identifier 2*** component identifier. For more information about component identifiers and the ***Class Identifier 2*** identifier, see the ***Dynamic Application Development*** manual.
- The labels in your topology definition must be preceded by a colon, without a space.

Example: Defining the Application and Tier Labels

The next step is to define which Kubernetes labels will be used to identify the application and the tier. Once again, you will do so by adding entries to the ***Topology Configuration*** field in the ***Kubernetes Token Entry dashboard***. For example, you could enter the following to define the application:

```
Application:metadata:labels:app
```

This definition states that the Kubernetes label "app" will be used to create the application component.

Finally, the same is done for the Tier component, as follows:

```
Tier:metadata:labels:tier
```

This definition will use the label "tier" to create the application tier component.

You must define each of the components listed in your topology definition. Therefore, if the topology definition in the previous example included other components in addition to "Application" and "Tier", those would need to be defined similar to the application and tier definition examples in this section.

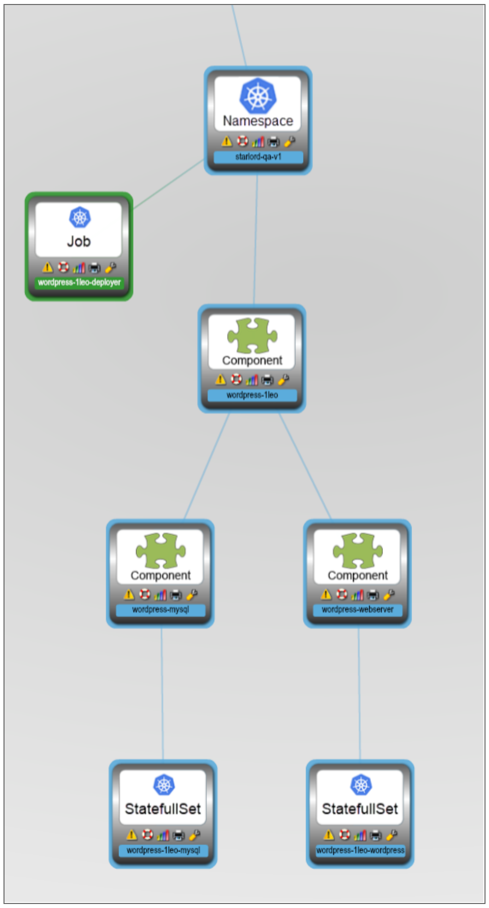
When defining these components, the first terms in the definitions must match the labels used in the topology definition. The middle terms in the definition represent the list of keys in the API response for a pod that identifies the members of that tier. The final term in the definition is the component's name.

You can other API responses from the payload if they are meaningful to you. For example, if you wanted to have a component based on an application and another based on the field "dnsPolicy", then your topology definition would be "TOPOLOGY:Application:DNSPolicy", and your component definitions would be as follows:

```
Application:metadata:labels:app

DNSPolicy:dnsPolicy
```

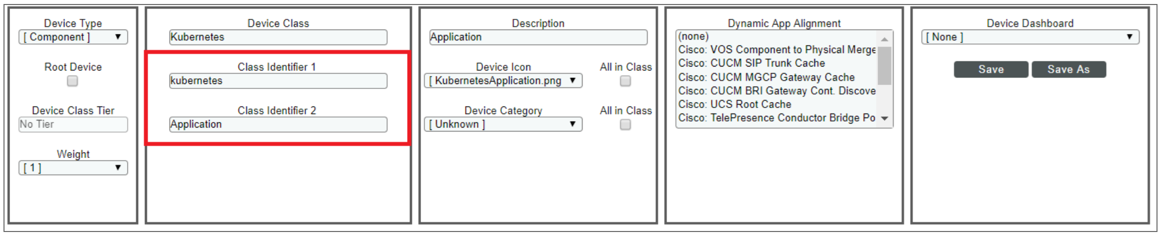
Example: Creating the Application and Tier Device Classes

The following example illustrates what the device component map might look like after the topology definitions are entered, as described in the previous two sections:



In this example, the generic "Component" device appears in the device component map. This is because no device classes match the names used in the topology definition and thus, SL1 uses a default device class.

For the purposes of this example, the following device classes are created:



Device Type: [Component]

Root Device: □

Device Class Tier: No Tier

Weight: [1]

Device Class: Kubernetes

Class Identifier 1: kubernetes

Class Identifier 2: Application

Description: Application

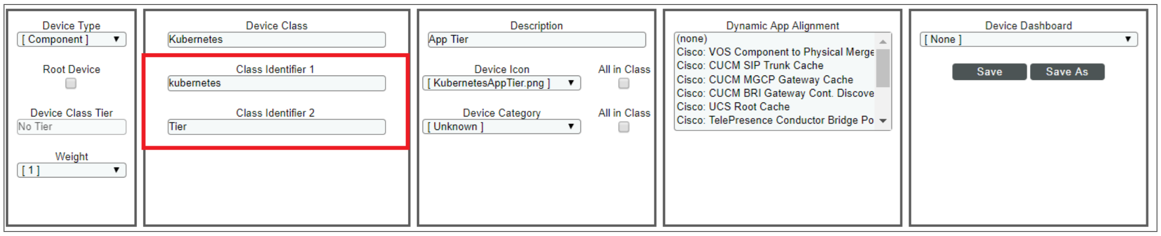
Device Icon: [KubernetesApplication.png] All in Class □

Device Category: [Unknown] All in Class □

Dynamic App Alignment

Device Dashboard: [None]

(none)
Cisco: VOS Component to Physical Merge
Cisco: CUCM SIP Trunk Cache
Cisco: CUCM MGCP Gateway Cache
Cisco: CUCM BRI Gateway Cont. Discove
Cisco: UCS Root Cache
Cisco: TelePresence Conductor Bridge Po
Save
Save As



Device Type: [Component]

Root Device: □

Device Class Tier: No Tier

Weight: [1]

Device Class: Kubernetes

Class Identifier 1: kubernetes

Class Identifier 2: Tier

Description: App Tier

Device Icon: [KubernetesAppTier.png] All in Class □

Device Category: [Unknown] All in Class □

Dynamic App Alignment

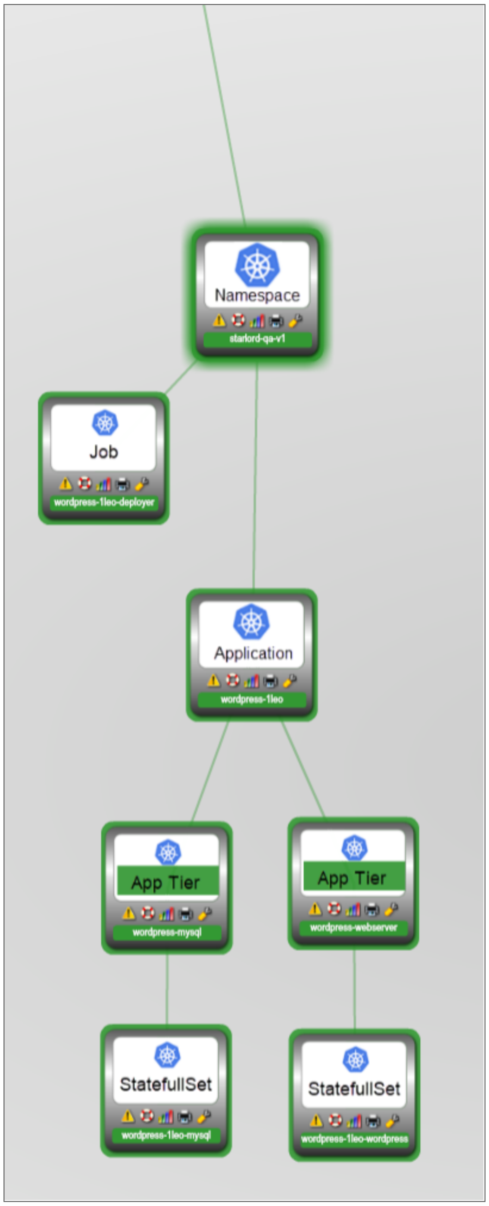
Device Dashboard: [None]

(none)
Cisco: VOS Component to Physical Merge
Cisco: CUCM SIP Trunk Cache
Cisco: CUCM MGCP Gateway Cache
Cisco: CUCM BRI Gateway Cont. Discove
Cisco: UCS Root Cache
Cisco: TelePresence Conductor Bridge Po
Save
Save As

NOTE: For information on how to create device classes, see the ***Device Management*** manual.

For this example, when creating the device classes, the ***Class Identifier 1*** field must always be "Kubernetes" and the ***Class Identifier 2*** field value ***must*** match the fields in the credential's topology definition—in this case, "Application" and "Tier".

After you have created these device classes, the device component map tree will appear as follows:



NOTE: If you previously discovered your Kubernetes system prior to defining device classes that were specific to your topology definition, then after you have defined the device classes, you must delete any of the generic devices that were used in lieu of those device classes in order for SL1 to rebuild the device component map with the new device classes. For instance, in our example above, the devices that were initially discovered with the generic "Component" device class had to be deleted from the **Device Manager** page (Registry > Devices > Device Manager) so that they could be automatically rediscovered with the newly defined device classes.

Filtering

Filtering enables the creation of specific aggregation components. In the preceding examples, application components would be created for ***all*** applications with that label. If you wanted to model only a set of applications or one specific application, you can do so using filtering.

For example, to create only a single "example-1" application component for the example application in the previous sections, you could modify the ***Topology Configuration*** as follows:

TOPOLOGY:Application:Tier

Application:metadata:labels:app.kubernetes.io/name=example-1

Tier:metadata:labels:app.kubernetes.io/component

TIP: You can include lists when filtering. For example, you could include multiple application names, separated by comma.

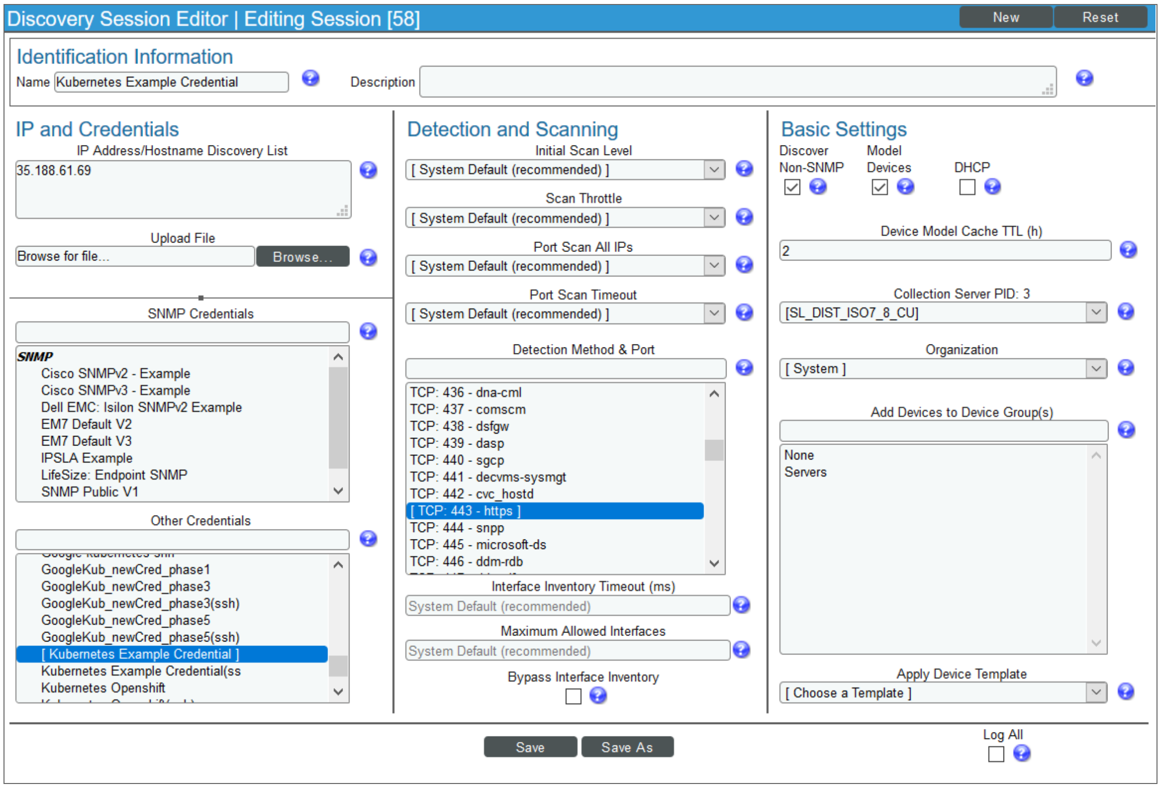
Discovering a Kubernetes Cluster

When you use the **Kubernetes Token Entry** dashboard to create credentials for the Kubernetes cluster that you want to monitor, SL1 automatically creates a discovery session that will discover your Kubernetes cluster and component devices.

To discover your Kubernetes cluster:

1. Go to the **Discovery Control Panel** page (System > Manage > Classic Discovery).
2. Locate the discovery session with the ***Session Name*** that matches the name of the Kubernetes master SSH/Key credential that you created. (For example, if your SSH/Key credential is called "Kubernetes Example Credential", then the discovery ***Session Name*** will also be "Kubernetes Example Credential".)
3. If you want to run the discovery session immediately, proceed to step 6. Otherwise, to view the discovery session, click its wrench icon (🔧) and continue to step 4.

4. The **Discovery Session Editor** window appears. It includes the following information:



Discovery Session Editor | Editing Session [58]

Identification Information

Name: Kubernetes Example Credential

Description:

IP and Credentials

IP Address/Hostname Discovery List: 35.188.61.69

Upload File: Browse for file... Browse...

SNMP Credentials

SNMP
Cisco SNMPv2 - Example
Cisco SNMPv3 - Example
Dell EMC: Isilon SNMPv2 Example
EM7 Default V2
EM7 Default V3
IPSLA Example
LifeSize: Endpoint SNMP
SNMP Public V1

Other Credentials

Google Kubernetes SSH
GoogleKub_newCred_phase1
GoogleKub_newCred_phase3
GoogleKub_newCred_phase3(ssh)
GoogleKub_newCred_phase5
GoogleKub_newCred_phase5(ssh)
[Kubernetes Example Credential]
Kubernetes Example Credential(ss
Kubernetes Openshift

Detection and Scanning

Initial Scan Level: [System Default (recommended)]

Scan Throttle: [System Default (recommended)]

Port Scan All IPs: [System Default (recommended)]

Port Scan Timeout: [System Default (recommended)]

Detection Method & Port

TCP: 436 - dna-cml
TCP: 437 - comscm
TCP: 438 - dsfgw
TCP: 439 - dasp
TCP: 440 - sgcp
TCP: 441 - decvms-sysmgt
TCP: 442 - cvc_hostd
[TCP: 443 - https]
TCP: 444 - snpp
TCP: 445 - microsoft-ds
TCP: 446 - ddm-rdb

Interface Inventory Timeout (ms): System Default (recommended)

Maximum Allowed Interfaces: System Default (recommended)

Bypass Interface Inventory: □

Basic Settings

Discover Non-SNMP: ☑ Model Devices: ☑ DHCP: □

Device Model Cache TTL (h): 2

Collection Server PID: 3: [SL_DIST_ISO7_8_CU]

Organization: [System]

Add Devices to Device Group(s)

None
Servers

Apply Device Template: [Choose a Template]

Log All: □

- ***Name***. Displays the name of the discovery session, which matches the name of the Kubernetes master SSH/Key credential that you created. If you want to edit this discovery session, you should type a new name in this field.
- ***IP Address/Hostname Discovery List***. Displays the IP address from the Kubernetes master SSH/Key credential that you created.
- ***Other Credentials***. The Kubernetes master SSH/Key credential that you created is selected.
- ***Detection Method & Port***. The port from the Kubernetes master SSH/Key credential that you created is selected.
- ***Discover Non-SNMP***. This checkbox is selected.

NOTE: For more information about the other fields on this page, see the ***Discovery & Credentials*** manual.

5. If you did not make any changes to the discovery session, you can close the window without saving and then proceed to the next step. If you did make changes, click **[Save As]** and then close the **Discovery Session Editor** window. The discovery session you created will appear at the top of the **Discovery Control Panel** page.

6. Click the discovery session's lightning-bolt icon ( ) to run discovery. The **Discovery Session** window appears.
7. When you run the discovery session, a Run Book Action in the *Kubernetes* PowerPack creates a virtual device that acts as the root device in your Kubernetes cluster. When the Kubernetes root device is discovered, you can click its device icon ( ) to view the cluster's device properties.

NOTE: SL1 might take several minutes to discover the component devices for your cluster.

Customizing Event Policies

The "Kubernetes: Event Configuration" Dynamic Application is a Journal Dynamic Application that collects the events reported by Kubernetes. The "Kubernetes: Normal event" and "Kubernetes: Warning event" are general event policies that are enabled by default.

Users can enable more specific event policies in the PowerPack after disabling the ""Kubernetes: Normal event" and "Kubernetes: Warning event" policies. To enable these event policies, perform the following steps:

1. Go to the **Event Policy Manager** page (Registry > Events > Event Manager).
2. Search for the "Kubernetes: Normal event" in the ***Event Policy Name*** field.
3. Select the wrench icon ( ) for the event policy to open the **Event Policy Editor** page.
4. In the ***Operational State*** drop-down, select *Disabled* and then click the **[Save]** button. Repeat these steps for the "Kubernetes: Warning event" event policy.
5. Once these event policies are disabled, find the event policies you want to use and enable them. You can enable more than one event policy at a time by selecting their checkboxes in the **Event Policy Manager** page, selecting *ENABLE these event policies* in the ***Select Action*** menu, and then clicking the **[Go]** button.

The following event policies are available:

Event Policy	Device	Severity	State
Kubernetes: Normal event	Any	Notice	Disabled
Kubernetes: Warning event	Any	Major	Enabled
Kubernetes: Network Failure	Any	Critical	Enabled
Kubernetes: Error Image Never Pull	Pod	Major	Enabled

Event Policy	Device	Severity	State
Kubernetes: Failed to Create Pod Container	Pod	Major	Enabled
Kubernetes: Failed to Kill Pod	Pod	Major	Enabled
Kubernetes: Failed Start Hook	Pod	Major	Enabled
Kubernetes: Failed Sync	Pod	Major	Enabled
Kubernetes: Failed Validation	Pod	Major	Enabled
Kubernetes: Free Disk Space Failed	Pod	Major	Enabled
Kubernetes: Image Pull Backoff	Pod	Major	Enabled
Kubernetes: Pod Container Created	Pod	Notice	Disabled
Kubernetes: Pod Exceeded Grace Period	Pod	Major	Enabled
Kubernetes: Pod Failed	Pod	Notice	Enabled
Kubernetes: Pod Image Inspect Failed	Pod	Notice	Enabled
Kubernetes: Pod Image Pulled	Pod	Notice	Disabled
Kubernetes: Pod Image Pulling	Pod	Major	Disabled
Kubernetes: Pod Killing	Pod	Major	Enabled
Kubernetes: Pod Network Not Ready	Pod	Major	Enabled
Kubernetes: Pod Preempting	Pod	Major	Enabled
Kubernetes: Pod Started	Pod	Notice	Disabled
Kubernetes: Pod Unhealthy	Pod	Major	Enabled
Kubernetes: Prestop Hook	Pod	Major	Enabled

Event Policy	Device	Severity	State
Kubernetes: Probe Warning	Pod	Major	Enabled
Kubernetes: Already Mounted Volume	Node	Notice	Enabled
Kubernetes: Container GC Failed	Node	Major	Disabled
Kubernetes: Failed Attach Volume	Node	Critical	Enabled
Kubernetes: Failed Create Pod Sandbox	Node	Notice	Enabled
Kubernetes: Failed Map Volume	Node	Major	Enabled
Kubernetes: Failed Mount	Node	Major	Enabled
Kubernetes: Failed Node Allocatable Enforcement	Node	Major	Enabled
Kubernetes: Failed Pod Sandbox Status	Node	Notice	Enabled
Kubernetes: File System Resize Failed	Node	Major	Enabled
Kubernetes: File System Resize Successful	Node	Notice	Enabled
Kubernetes: Image GC Failed	Node	Major	Enabled
Kubernetes: Invalid Disk Capacity	Node	Major	Enabled
Kubernetes: Kubelet Setup Failed	Node	Critical	Enabled
Kubernetes: Node Allocatable Enforced	Node	Notice	Enabled
Kubernetes: Node Not Ready	Node	Major	Enabled
Kubernetes: Node Not Schedulable	Node	Major	Enabled

Event Policy	Device	Severity	State
Kubernetes: Node Ready	Node	Notice	Enabled
Kubernetes: Node Schedulable	Node	Notice	Disabled
Kubernetes: Rebooted	Node	Critical	Enabled
Kubernetes: Sandbox Changed	Node	Notice	Enabled
Kubernetes: Starting	Node	Notice	Enabled
Kubernetes: Successful Attach Volume	Node	Notice	Disabled
Kubernetes: Successful Mount Volume	Node	Notice	Disabled
Kubernetes: Volume Resize Failed	Node	Major	Enabled
Kubernetes: Volume Resize Successful	Node	Notice	Enabled
Kubernetes: Node Condition Healthy	Node	Healthy	Enabled
Kubernetes: Node Condition Unhealthy	Node	Major	Enabled
Kubernetes: Node Condition Unknown	Node	Major	Enabled
Kubernetes: Cluster Creation	Pingable	Notice	Enabled
Kubernetes: Component Healthy State	Cluster	Healthy	Enabled
Kubernetes: Component No Healthy State	Cluster	Major	Enabled
Kubernetes: Node Status Changed	Cluster	Notice	Enabled
Kubernetes: Persistent Volume status Healthy	Cluster	Healthy	Enabled
Kubernetes: Persistent Volume Status Unhealthy	Cluster	Major	Enabled

Event Policy	Device	Severity	State
Kubernetes: Restart Count Exceeded Threshold	Deployment/Daemon Set	Minor	Enabled
Kubernetes: Restart Count Returned to Normal	Deployment/Daemon Set	Healthy	Enabled

Viewing Component Devices

When SL1 performs collection for the Kubernetes cluster, SL1 will create component devices that represent each device and align other Dynamic Applications to those component devices. Some of the Dynamic Applications aligned to the component devices will also be used to create additional component devices. All component devices appear in the **Devices** page just like devices discovered using the ScienceLogic discovery process.

In addition to the **Devices** page, you can view the Kubernetes cluster and all associated component devices in the following places in the user interface:

- The **Device Investigator** Map page (click **Map** in the **Device Investigator** page) displays a map of a particular device and all of the devices with which it has parent-child relationships. Double-clicking any of the listed devices reloads the page to make the selected device the primary device

- The **Device Components** page (Devices > Device Components) displays a list of all root devices and component devices discovered by SL1 in an indented view, so you can easily view the hierarchy and relationships between child devices, parent devices, and root devices. To view the component devices associated with a Kubernetes cluster, find the cluster device and click its plus icon (+).

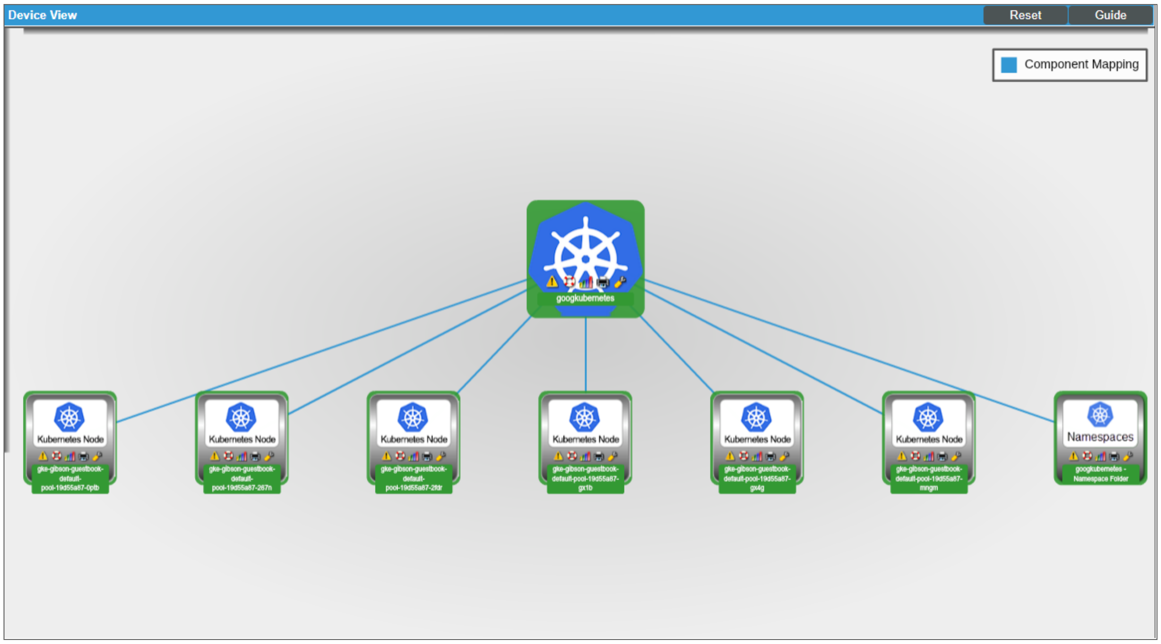
- The **Component Map** page (Classic Maps > Device Maps > Components) allows you to view devices by root node and view the relationships between root nodes, parent components, and child components in a map. This makes it easy to visualize and manage root nodes and their components. SL1 automatically updates the **Component Map** as new component devices are discovered. SL1 also updates each map with the latest status and event information. To view the map for a Kubernetes cluster, go to Classic Maps > Device Maps > Components, and select the map from the list in the left NavBar. To learn more about the **Component Map** page, see the ***Views*** manual.

Viewing Component Devices in the SL1 Classic User Interface

When SL1 performs collection for the Kubernetes cluster, SL1 will create component devices that represent each device and align other Dynamic Applications to those component devices. Some of the Dynamic Applications aligned to the component devices will also be used to create additional component devices. All component devices appear in the **Device Manager** page just like devices discovered using the ScienceLogic discovery process.

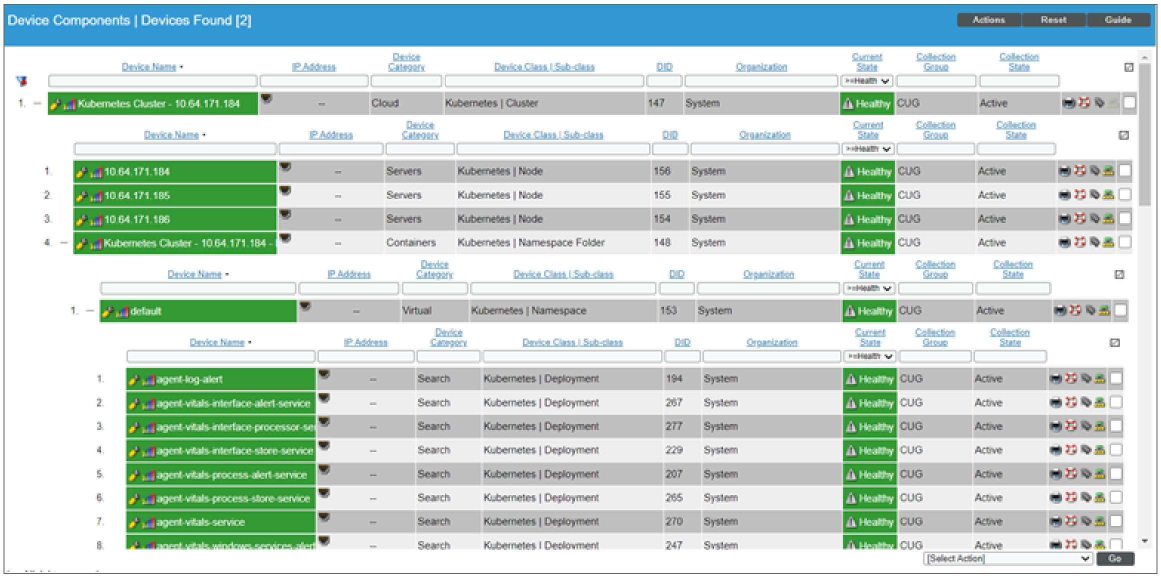
In addition to the **Device Manager** page, you can view the Kubernetes cluster and all associated component devices in the following places in the user interface:

- The **Device View** modal page (click the bar-graph icon [📶] for a device, then click the **Topology** tab) displays a map of a particular device and all of the devices with which it has parent-child relationships. Double-clicking any of the devices listed reloads the page to make the selected device the primary device:



Device View
Reset
Guide
Component Mapping
googkubernetes
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-0ptb
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-267n
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-2fdr
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-gx1b
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-gx4g
Kubernetes Node
gke-gibson-guestbook-default-pool-19d55a87-mngm
Namespaces
googkubernetes - Namespace Folder

- The **Device Components** page (Registry > Devices > Device Components) displays a list of all root devices and component devices discovered by SL1 in an indented view, so you can easily view the hierarchy and relationships between child devices, parent devices, and root devices. To view the component devices associated with a Kubernetes cluster, find the cluster device and click its plus icon (**+**):


Device Components | Devices Found [2]

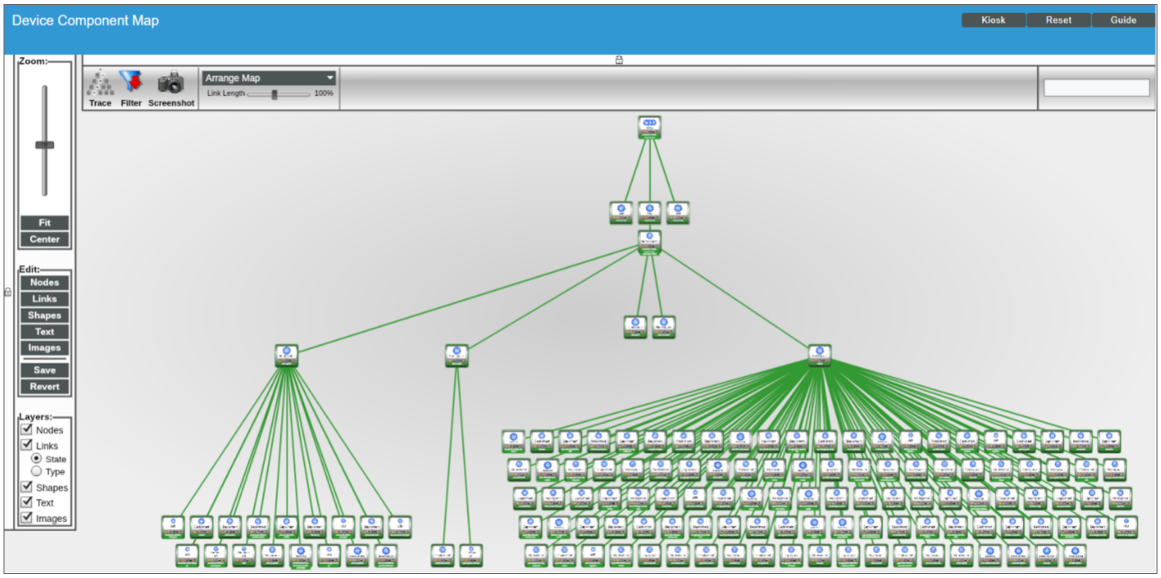
	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	Kubernetes Cluster - 10.64.171.184	--	Cloud	Kubernetes | Cluster	147	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	10.64.171.184	--	Servers	Kubernetes | Node	156	System	Healthy	CUG	Active
2.	10.64.171.185	--	Servers	Kubernetes | Node	155	System	Healthy	CUG	Active
3.	10.64.171.186	--	Servers	Kubernetes | Node	154	System	Healthy	CUG	Active
4.	Kubernetes Cluster - 10.64.171.184 -	--	Containers	Kubernetes | Namespace Folder	148	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	default	--	Virtual	Kubernetes | Namespace	153	System	Healthy	CUG	Active

	Device Name	IP Address	Device Category	Device Class | Sub-class	DID	Organization	Current State	Collection Group	Collection State
1.	agent-log-alert	--	Search	Kubernetes | Deployment	194	System	Healthy	CUG	Active
2.	agent-vitals-interface-alert-service	--	Search	Kubernetes | Deployment	267	System	Healthy	CUG	Active
3.	agent-vitals-interface-processor-se	--	Search	Kubernetes | Deployment	277	System	Healthy	CUG	Active
4.	agent-vitals-interface-store-service	--	Search	Kubernetes | Deployment	229	System	Healthy	CUG	Active
5.	agent-vitals-process-alert-service	--	Search	Kubernetes | Deployment	207	System	Healthy	CUG	Active
6.	agent-vitals-process-store-service	--	Search	Kubernetes | Deployment	265	System	Healthy	CUG	Active
7.	agent-vitals-service	--	Search	Kubernetes | Deployment	270	System	Healthy	CUG	Active
8.	agent-vitals-windows-services-alert	--	Search	Kubernetes | Deployment	247	System	Healthy	CUG	Active

- The **Device Component Map** page (Classic Maps > Device Maps > Components) allows you to view devices by root node and view the relationships between root nodes, parent components, and child components in a map. This makes it easy to visualize and manage root nodes and their components. SL1 automatically updates the **Device Component Map** as new component devices are discovered. SL1 also updates each map with the latest status and event information. To view the map for a Kubernetes cluster, go to the **Device Component Map** page and select the map from the list in the left NavBar. To learn more about the **Device Component Map** page, see the ***Views*** manual.



Chapter

3

Dashboards

Overview

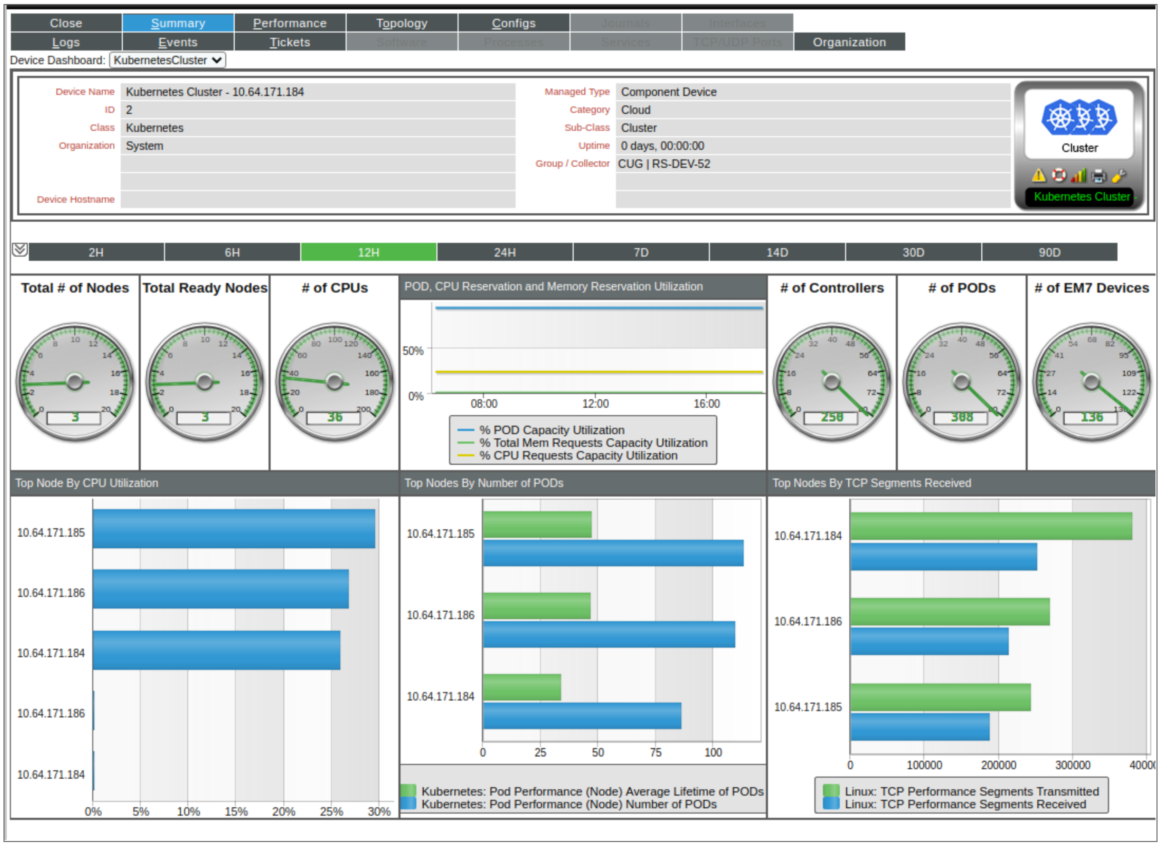
The following sections describe the device dashboards that are included in the *Kubernetes* PowerPack:

Device Dashboards

The *Kubernetes* PowerPack includes device dashboards that provide summary information for Kubernetes component devices. The following device dashboards in the *Kubernetes* PowerPack are aligned as the default device dashboard for the equivalent device class.

NOTE: If the device dashboards are not populating data for your selected time frame, change the time frame then click back to your desired time frame and the data will populate.

Kubernetes Cluster



Device Dashboard: KubernetesCluster

Device Name: Kubernetes Cluster - 10.64.171.184

ID: 2

Class: Kubernetes

Organization: System

Device Hostname:

Managed Type: Component Device

Category: Cloud

Sub-Class: Cluster

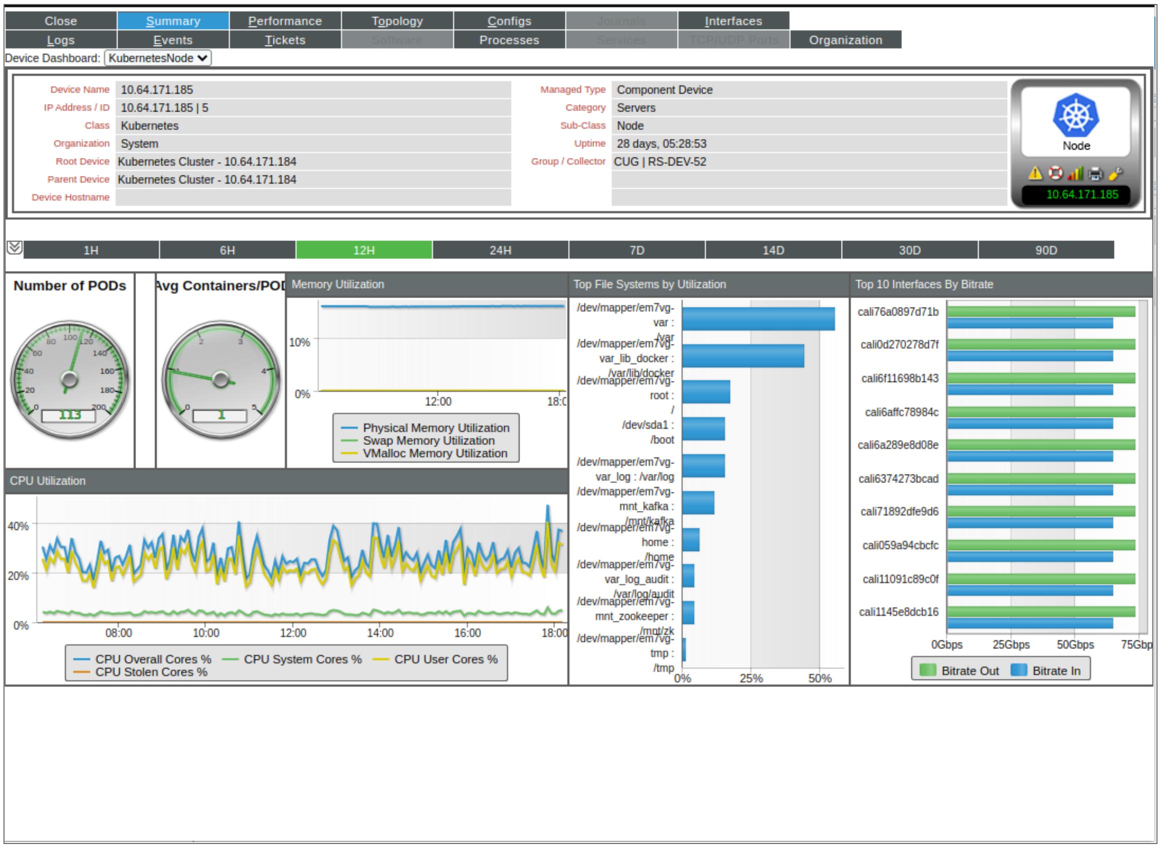
Uptime: 0 days, 00:00:00

Group / Collector: CUG | RS-DEV-52

The Kubernetes Cluster device dashboard displays the following information:

- The basic information about the device
- Six gauges that display the following metrics:
  - Total number of nodes
  - Total number of ready nodes
  - Number of CPUs
  - Number of Controllers
  - Number of Pods
  - Number of ScienceLogic devices
- The average pod lifetime
- Top nodes, sorted by CPU utilization
- Top nodes, sorted by the number of Pods
- Top nodes, sorted by the number of TCP segments received

Kubernetes Node



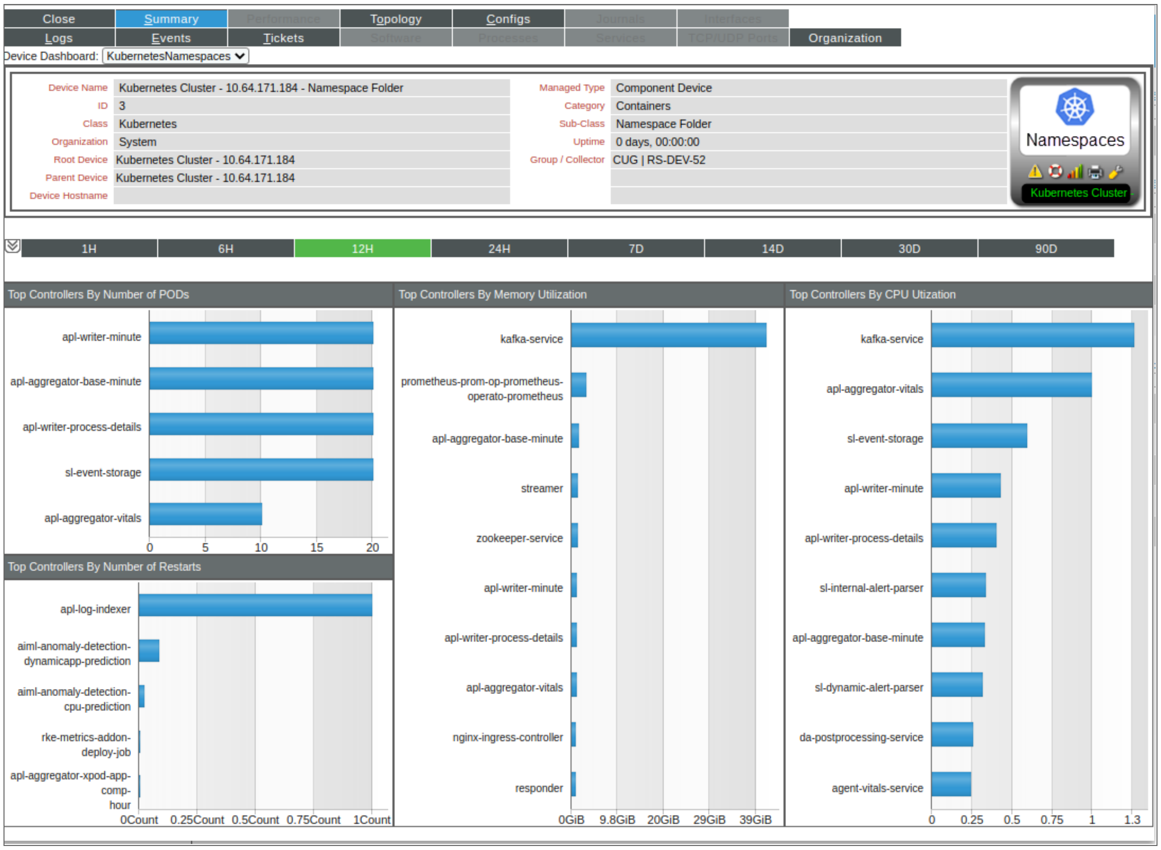
Device Dashboard: KubernetesNode

Device Name	10.64.171.185	Managed Type	Component Device
IP Address / ID	10.64.171.185 | 5	Category	Servers
Class	Kubernetes	Sub-Class	Node
Organization	System	Uptime	28 days, 05:28:53
Root Device	Kubernetes Cluster - 10.64.171.184	Group / Collector	CUG | RS-DEV-52
Parent Device	Kubernetes Cluster - 10.64.171.184		
Device Hostname			

The Kubernetes Node device dashboard displays the following information:

- The basic information about the device
- Number of active pods
- Average number of containers per pod
- Memory utilization
- CPU utilization
- Top file systems, sorted by utilization
- Top 10 interfaces, sorted by the number of inbound packets.

Kubernetes Namespace



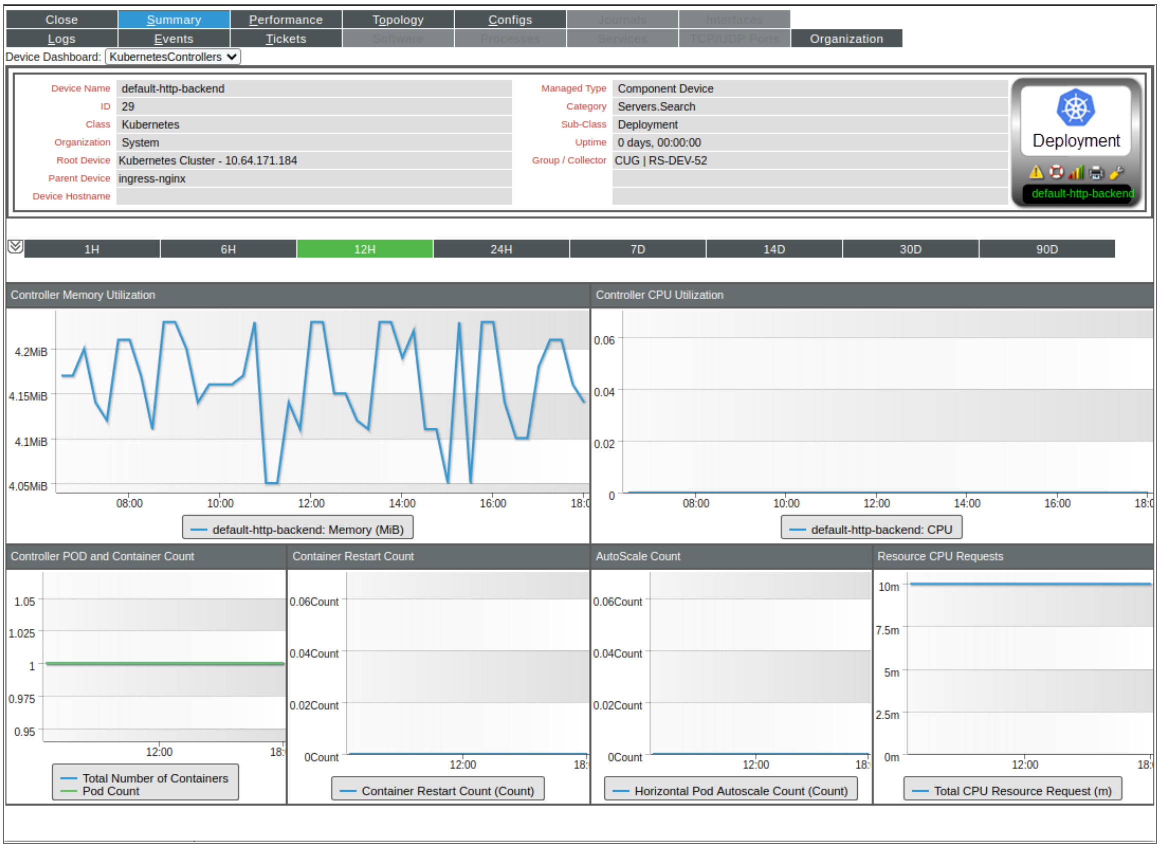
Device Dashboard: KubernetesNamespaces

Device Name	Kubernetes Cluster - 10.64.171.184 - Namespace Folder	Managed Type	Component Device
ID	3	Category	Containers
Class	Kubernetes	Sub-Class	Namespace Folder
Organization	System	Uptime	0 days, 00:00:00
Root Device	Kubernetes Cluster - 10.64.171.184	Group / Collector	CUG | RS-DEV-52
Parent Device	Kubernetes Cluster - 10.64.171.184		
Device Hostname			

The Kubernetes Namespace device dashboard displays the following information:

- The basic information about the device
- Top controllers, sorted by the number of pods
- Top controllers, sorted by the number of restarts
- Top controllers, sorted by memory utilization
- Top controllers, sorted by CPU utilization

Kubernetes Controllers



Device Dashboard: KubernetesControllers

Device Name	default-http-backend	Managed Type	Component Device
ID	29	Category	Servers.Search
Class	Kubernetes	Sub-Class	Deployment
Organization	System	Uptime	0 days, 00:00:00
Root Device	Kubernetes Cluster - 10.64.171.184	Group / Collector	CUG | RS-DEV-52
Parent Device	ingress-nginx		
Device Hostname			

The Kubernetes Controllers device dashboard displays the following information:

- The basic information about the device
- Controller memory utilization
- Controller CPU utilization
- Controller pod and container count
- Container restart count
- AutoScale count
- Resource CPU requests

LIMITATION OF LIABILITY AND GENERAL DISCLAIMER

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information on this Site, information on this Site may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described in this Site at any time without notice.

Copyrights and Trademarks

ScienceLogic, the ScienceLogic logo, and EM7 are trademarks of ScienceLogic, Inc. in the United States, other countries, or both.

Below is a list of trademarks and service marks that should be credited to ScienceLogic, Inc. The ® and ™ symbols reflect the trademark registration status in the U.S. Patent and Trademark Office and may not be appropriate for materials to be distributed outside the United States.

- ScienceLogic™
- EM7™ and em7™
- Simplify IT™
- Dynamic Application™
- Relational Infrastructure Management™

The absence of a product or service name, slogan or logo from this list does not constitute a waiver of ScienceLogic's trademark or other intellectual property rights concerning that name, slogan, or logo.

Please note that laws concerning use of trademarks or product names vary by country. Always consult a local attorney for additional guidance.

Other

If any provision of this agreement shall be unlawful, void, or for any reason unenforceable, then that provision shall be deemed severable from this agreement and shall not affect the validity and enforceability of any remaining provisions. This is the entire agreement between the parties relating to the matters contained herein.

In the U.S. and other jurisdictions, trademark owners have a duty to police the use of their marks. Therefore, if you become aware of any improper use of ScienceLogic Trademarks, including infringement or counterfeiting by third parties, report them to Science Logic's legal department immediately. Report as much detail as possible about the misuse, including the name of the party, contact information, and copies or photographs of the potential misuse to: legal@sciencelogic.com

ScienceLogic

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

International: +1-703-354-1010