



Monitoring Kubernetes

Kubernetes PowerPack version 101

Table of Contents

Introduction	3
What is Kubernetes?	4
What Does the Kubernetes PowerPack Monitor?	4
Installing the Kubernetes PowerPack	5
Configuration and Discovery	7
Prerequisites for Monitoring Kubernetes Clusters	8
Required Permissions for the Service Account Token	8
Creating Credentials for Kubernetes Clusters	9
Viewing the Kubernetes Credentials	13
Master SSH/Key Credential	14
SOAP/XML Credential	15
Node SSH/Key Credential	15
Specifying the Kubernetes Topology for Discovery	15
Example: Defining the Topology	16
Example: Defining the Application and Tier Labels	18
Example: Creating the Application and Tier Device Classes	19
Metric Aggregation	22
Filtering	22
Discovering a Kubernetes Cluster	23
Customizing Event Policies	25
Viewing Component Devices	27
Relationships Between Component Devices	29
Dashboards	30
Device Dashboards	30
Kubernetes Cluster	31
Kubernetes Node	32
Kubernetes Namespace	33
Kubernetes Controllers	34
Docker: Container	35

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 of the menu options, click the Advanced menu icon (.

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

What is Kubernetes?	4
What Does the Kubernetes PowerPack Monitor?	4
Installing the Kubernetes PowerPack	5

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**, and **CronJobs**.

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 leverages the capabilities of the *Linux Base Pack* PowerPack and *Docker* PowerPack to provide a comprehensive view of the Kubernetes cluster, including its underlying hardware and Docker infrastructure. You must install and run the most recent versions of these PowerPacks before you install the *Kubernetes* PowerPack. For more information about using these PowerPacks, see the **Monitoring Linux and Solaris** and **Monitoring Docker** manuals.

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
 - Discover and monitor Docker containers
 - 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 *Docker* and *Linux Base Pack* PowerPacks 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

WARNING: Before installing the *Kubernetes* PowerPack, you must first import and install the *Linux Base Pack* PowerPack version 101 or greater and the *Docker* PowerPack version 102 or greater. The *Kubernetes* PowerPack leverages both of these PowerPacks and will not work properly if they are not also installed.

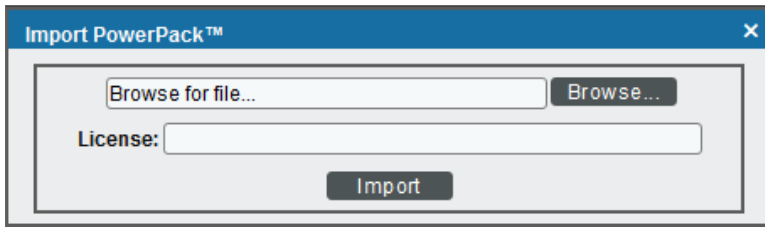
Before completing the steps in this manual, you must import and install the latest version of the *Kubernetes* PowerPack.

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 Customer Portal](#).
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 of 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	8
<i>Required Permissions for the Service Account Token</i>	8
Creating Credentials for Kubernetes Clusters	9
<i>Viewing the Kubernetes Credentials</i>	13
Master SSH/Key Credential	14
SOAP/XML Credential	15
Node SSH/Key Credential	15
Specifying the Kubernetes Topology for Discovery	15
<i>Example: Defining the Topology</i>	16
<i>Example: Defining the Application and Tier Labels</i>	18
<i>Example: Creating the Application and Tier Device Classes</i>	19
<i>Metric Aggregation</i>	22
<i>Filtering</i>	22
Discovering a Kubernetes Cluster	23

Customizing Event Policies	25
Viewing Component Devices	27
Relationships Between Component Devices	29

Prerequisites for Monitoring Kubernetes Clusters

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

1. Import and install the *Linux Base Pack PowerPack* version 101 or greater and the *Docker PowerPack* version 102 or greater.
2. Create a Kubernetes service account that SL1 can use to communicate with the Kubernetes API. This service account must, at a minimum, have the following roles assigned to it:
 - view
 - cluster-reader
 - cluster-admin
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 as well as the Docker API.

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

CAUTION: ScienceLogic recommends monitoring a maximum of 2,000 containers in a single cluster.

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
- replicaset
- daemonsets

```



```
- cronjobs
- jobs
- componentstatuses
- namespaces
- services
- persistentvolumes
- persistentvolumeclaims
- events
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*.

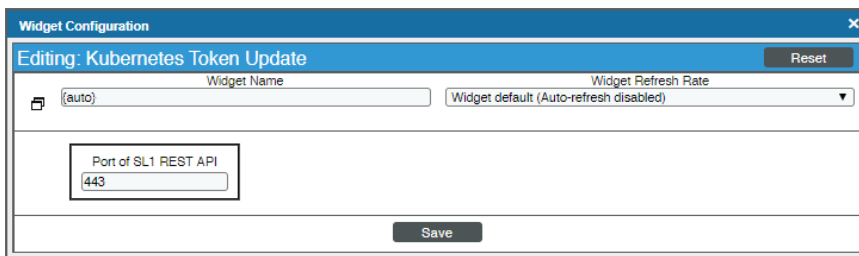
- In the **Topology Configuration** field:
 - In the **Enter Label** field, type a topology definition that you want SL1 to use when discovering Kubernetes component devices. (For more information, see the [Specifying the Kubernetes Topology for Discovery](#) section.)
 - Click **[Add]**.
 - Repeat the 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.

- To customize the IP address port for the Kubernetes cluster, go to the **Dashboards** page (System > Customize > Dashboards).
- In the **Dashboard Name** column, type "Kubernetes Token Entry". Click the wrench icon for the dashboard to open the **Dashboard Editor** page.
- 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 customized port number into the **Port of SL1 REST API** field.

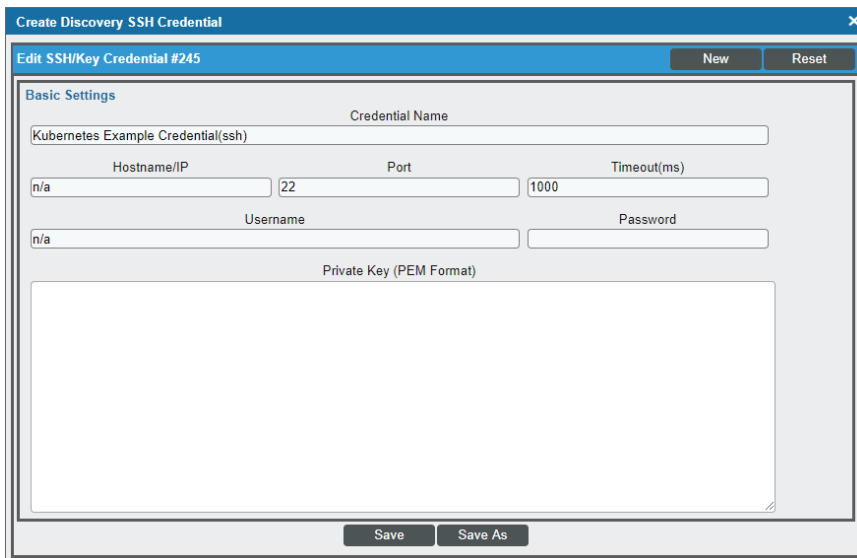


- Click the **[Save]** button.
- In the **Token** field, paste your Kubernetes service account token.

- 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:



- **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 the IP address of the Kubernetes cluster.
- **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.
- **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.

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).

SOAP/XML Credential

The screenshot shows the 'Credential Editor [247]' window for editing a 'SOAP/XML Credential #247'. The interface includes several sections: 'Basic Settings' with fields for Profile Name, Content Encoding, Method, HTTP Version, and URL; 'Soap Options' with an Embedded Password field and four Embed Value fields; 'Proxy Settings' with Hostname/IP, Port, User, and Password fields; 'CURL Options' with a list of options and arrows; and 'HTTP Headers' with a list of headers. 'Save' and 'Save As' buttons are at the bottom.

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. For more information, see the section on [Metric Aggregation](#).
- **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.

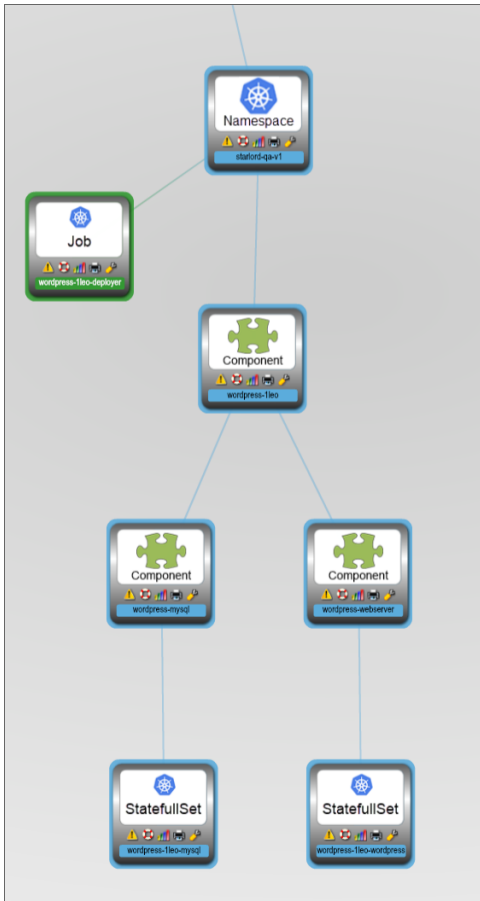
You can other API responses from the payload if they are meaningful to you. For example, if you wanted to have a component based on an application and another based on the field "dnsPolicy", then your topology definition would be "TOPOLOGY:Application:DNSPolicy", and your component definitions would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

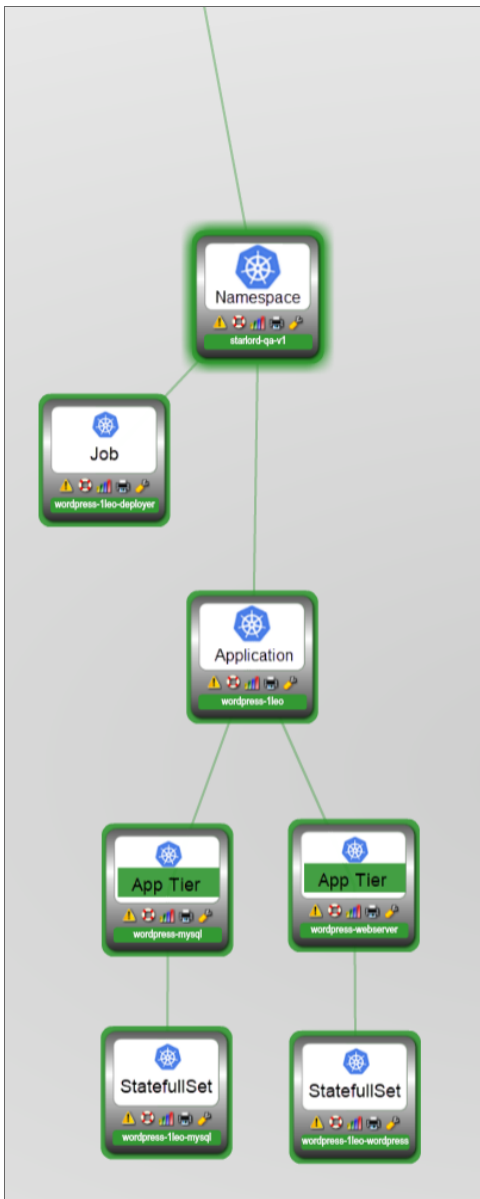
Device Type [Component]	Device Class Kubernetes	Description Application	Dynamic App Alignment (none) Cisco: VOS Component to Physical Merge Cisco: CUCM SIP Trunk Cache Cisco: CUCM MGCP Gateway Cache Cisco: CUCM BRI Gateway Cont. Discove Cisco: UCS Root Cache Cisco: TelePresence Conductor Bridge Po	Device Dashboard [None] Save Save As
Root Device <input type="checkbox"/>	Class Identifier 1 kubernetes	Device Icon [KubernetesApplication.png] All in Class <input type="checkbox"/>		
Device Class Tier No Tier	Class Identifier 2 Application	Device Category [Unknown] All in Class <input type="checkbox"/>		
Weight [1]				

Device Type [Component]	Device Class Kubernetes	Description App Tier	Dynamic App Alignment (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	Device Dashboard [None] Save Save As
Root Device <input type="checkbox"/>	Class Identifier 1 kubernetes	Device Icon [KubernetesAppTier.png] All in Class <input type="checkbox"/>		
Device Class Tier No Tier	Class Identifier 2 Tier	Device Category [Unknown] All in Class <input type="checkbox"/>		
Weight [1]				

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:



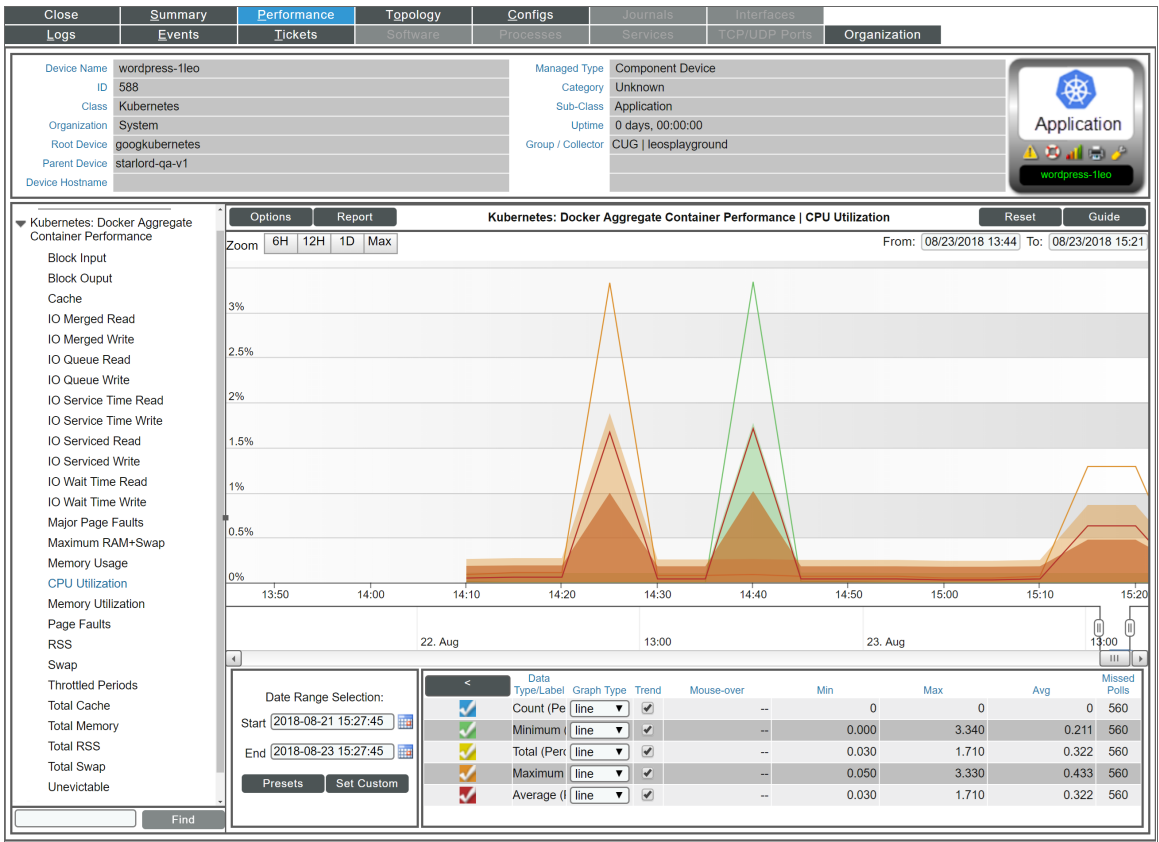
NOTE: If you previously discovered your Kubernetes system prior to defining device classes that were specific to your topology definition, then after you have defined the device classes, you must delete any of the generic devices that were used in lieu of those device classes in order for SL1 to rebuild 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.

Metric Aggregation

The *Kubernetes* PowerPack also enables you to aggregate Docker Container metrics on any of the Aggregation components. To enable Docker Container aggregation, simply add the "@" symbol to the component definition. For example:

```
Application@:metadata:labels:app.kubernetes.io/name
```

The following screenshot illustrates the metrics that will be collected after you have added the "@" symbol to the definition:



NOTE: These Dynamic Applications are always aligned to the component device; however, they do not collect data unless you include the "@" symbol in the component definition.

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.

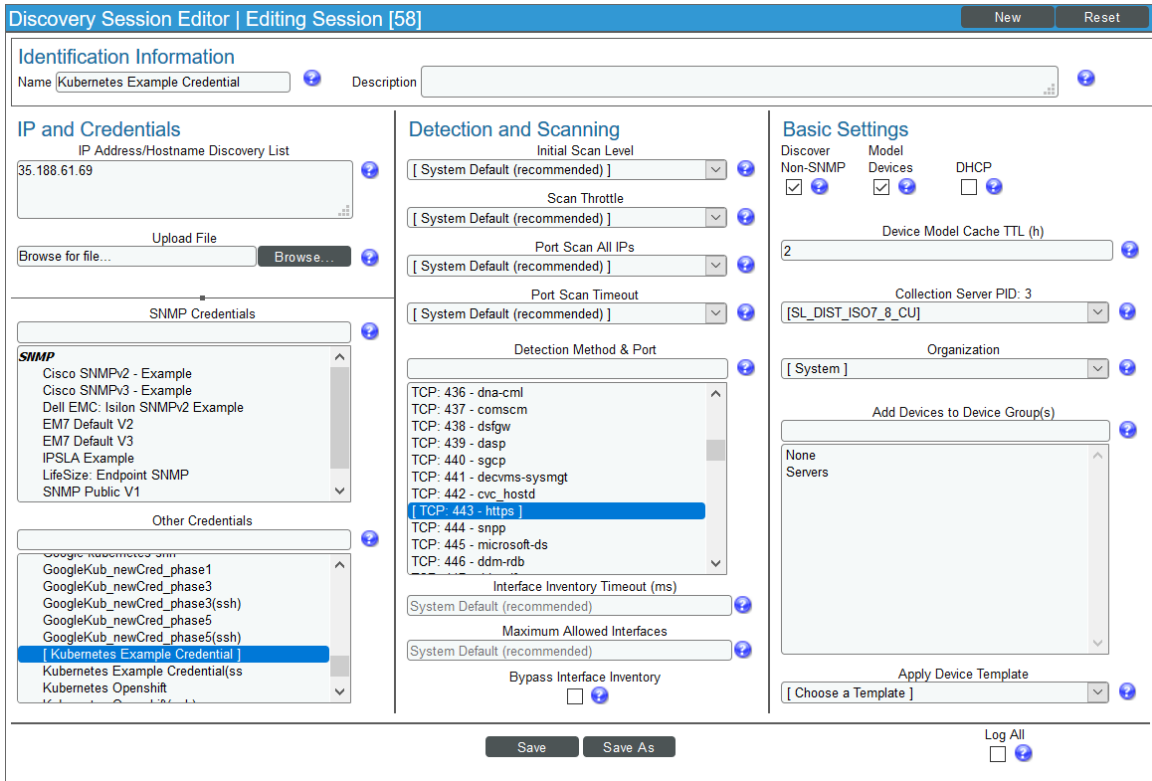
Discovering a Kubernetes Cluster

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 > 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:



- **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.


- Click the discovery session's lightning-bolt icon () to run discovery. The **Discovery Session** window appears.
- 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:

- Go to the **Event Policy Manager** page (Registry > Events > Event Manager).
- Search for the "Kubernetes: Normal event" in the **Event Policy Name** field.
- Select the wrench icon () for the event policy to open the **Event Policy Editor** page.
- In the **Operational State** drop-down, select *Disabled* and then click the **[Save]** button. Repeat these steps for the "Kubernetes: Warning event" event policy.
- 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	Enabled
Kubernetes: Warning event	Any	Major	Enabled
Kubernetes: Error Image Never Pull	Pod	Major	Disabled
Kubernetes: Failed to Create Pod Container	Pod	Major	Disabled
Kubernetes: Failed to Kill Pod	Pod	Major	Disabled
Kubernetes: Failed Start Hook	Pod	Major	Disabled
Kubernetes: Failed Sync	Pod	Major	Disabled


Event Policy	Device	Severity	State
Kubernetes: Failed Validation	Pod	Major	Disabled
Kubernetes: Free Disk Space Failed	Pod	Major	Disabled
Kubernetes: Image Pull Backoff	Pod	Major	Disabled
Kubernetes: Pod Container Created	Pod	Notice	Disabled
Kubernetes: Pod Exceeded Grace Period	Pod	Major	Disabled
Kubernetes: Pod Failed	Pod	Notice	Disabled
Kubernetes: Pod Image Inspect Failed	Pod	Notice	Disabled
Kubernetes: Pod Image Pulled	Pod	Notice	Disabled
Kubernetes: Pod Image Pulling	Pod	Major	Disabled
Kubernetes: Pod Killing	Pod	Major	Disabled
Kubernetes: Pod Network Not Ready	Pod	Major	Disabled
Kubernetes: Pod Preempting	Pod	Major	Disabled
Kubernetes: Pod Started	Pod	Notice	Disabled
Kubernetes: Pod Unhealthy	Pod	Major	Disabled
Kubernetes: Prestop Hook	Pod	Major	Disabled
Kubernetes: Probe Warning	Pod	Major	Disabled
Kubernetes: Already Mounted Volume	Node	Notice	Disabled
Kubernetes: Container GC Failed	Node	Major	Disabled
Kubernetes: Failed Attach Volume	Node	Critical	Disabled
Kubernetes: Failed Create Pod Sandbox	Node	Notice	Disabled
Kubernetes: Failed Map Volume	Node	Major	Disabled
Kubernetes: Failed Mount	Node	Major	Disabled
Kubernetes: Failed Node Allocatable Enforcement	Node	Major	Disabled
Kubernetes: Failed Pod Sandbox Status	Node	Notice	Disabled
Kubernetes: File System Resize Failed	Node	Major	Disabled
Kubernetes: File System Resize Successful	Node	Notice	Disabled
Kubernetes: Image GC Failed	Node	Major	Disabled
Kubernetes: Invalid Disk Capacity	Node	Major	Disabled
Kubernetes: Kubelet Setup Failed	Node	Critical	Disabled

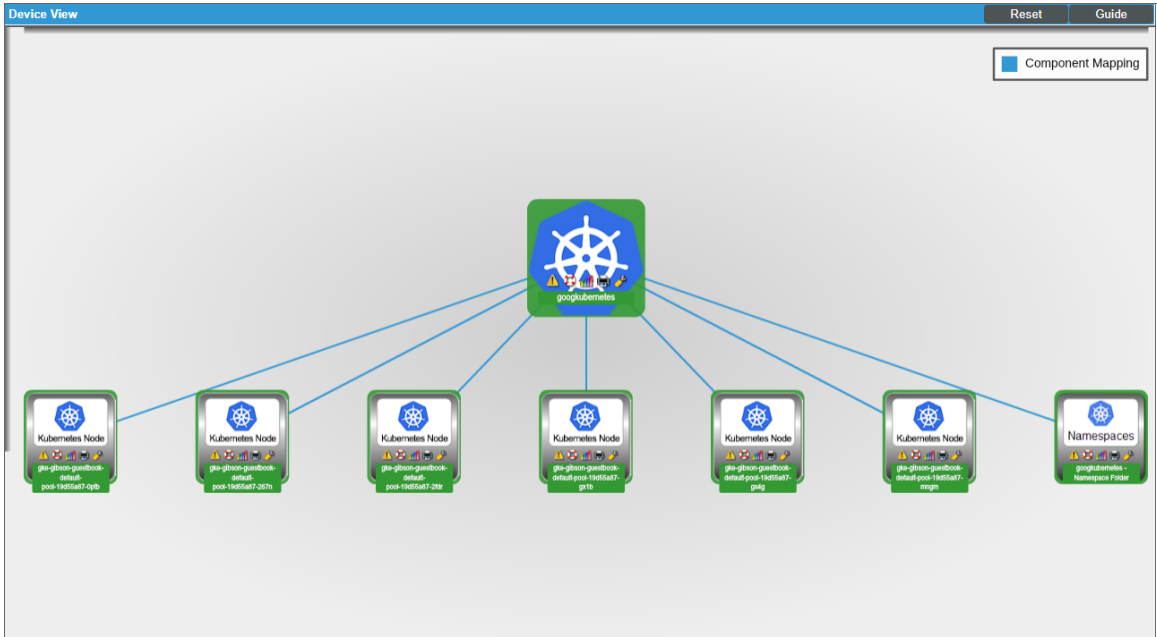
Event Policy	Device	Severity	State
Kubernetes: Node Allocatable Enforced	Node	Notice	Disabled
Kubernetes: Node Not Ready	Node	Major	Disabled
Kubernetes: Node Not Schedulable	Node	Major	Disabled
Kubernetes: Node Ready	Node	Notice	Disabled
Kubernetes: Node Schedulable	Node	Notice	Disabled
Kubernetes: Rebooted	Node	Critical	Disabled
Kubernetes: Sandbox Changed	Node	Notice	Disabled
Kubernetes: Starting	Node	Notice	Disabled
Kubernetes: Successful Attach Volume	Node	Notice	Disabled
Kubernetes: Successful Mount Volume	Node	Notice	Disabled
Kubernetes: Volume Resize Failed	Node	Major	Disabled
Kubernetes: Volume Resize Successful	Node	Notice	Disabled

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 **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 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:

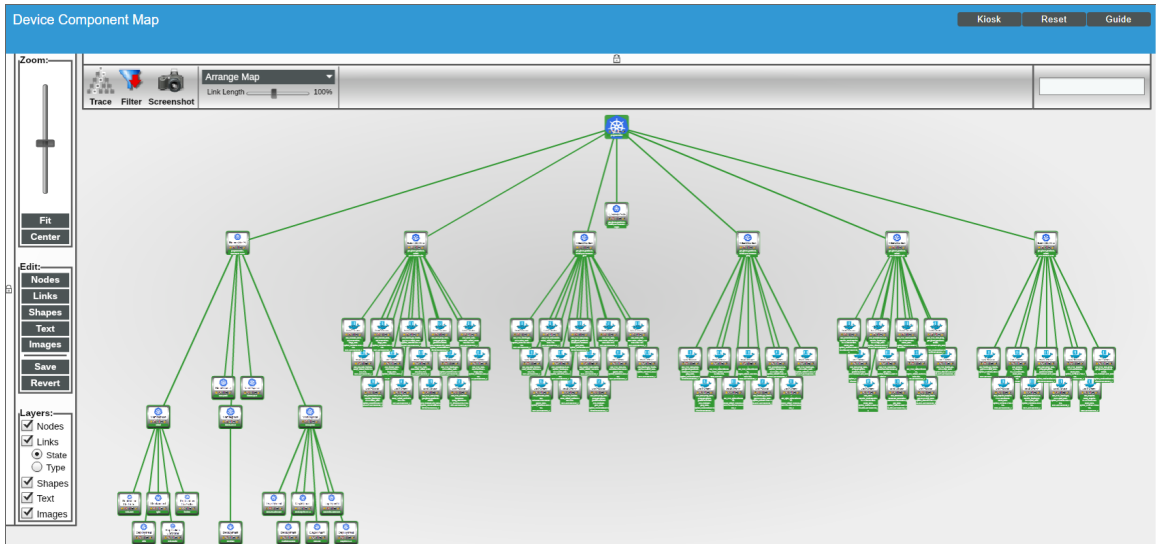


- The **Device Components** page (Registry > Devices > Device Components) displays a list of all root devices and component devices discovered by SL1 in an indented view, so you can easily view the hierarchy and relationships between child devices, parent devices, and root devices. To view the component 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	DIID	Organization	Current State	Collection Group	Collection State	Actions
1. - googkubernetes	--	Cloud	Kubernetes Cluster	1382	System	Healthy	CUG1	Active	[Icons]
1. - gke-gibson-guestbook-default-pool-19	--	Servers	Kubernetes Node	1392	System	Healthy	CUG1	Active	[Icons]
1. 1. gke-event-exporter-event-exporter	--	Service	Container Docker Container	1446	System	Healthy	CUG1	Active	[Icons]
2. gke-fluentd-gcp-fluentd-gcp-v2.0.9	--	Service	Container Docker Container	1448	System	Healthy	CUG1	Active	[Icons]
3. gke-kube-proxy-kube-proxy-gke-gibson-g	--	Service	Container Docker Container	1442	System	Healthy	CUG1	Active	[Icons]
4. gke-kubernetes-dashboard-kubern	--	Service	Container Docker Container	1447	System	Healthy	CUG1	Active	[Icons]
5. gke-POD_cattle-75f6ccbd78-mgbvt	--	Service	Container Docker Container	1444	System	Healthy	CUG1	Active	[Icons]
6. gke-POD_event-exporter-v0.1.8-59	--	Service	Container Docker Container	1440	System	Healthy	CUG1	Active	[Icons]
7. gke-POD_fluentd-gcp-v2.0.9-cr4jh	--	Service	Container Docker Container	1445	System	Healthy	CUG1	Active	[Icons]
8. gke-POD_kube-proxy-gke-gibson-g	--	Service	Container Docker Container	1441	System	Healthy	CUG1	Active	[Icons]
9. gke-POD_kubernetes-dashboard-6	--	Service	Container Docker Container	1443	System	Healthy	CUG1	Active	[Icons]
10. gke-prometheus-to-sd-exporter_eve	--	Service	Container Docker Container	1438	System	Healthy	CUG1	Active	[Icons]
11. gke-prometheus-to-sd-exporter_flue	--	Service	Container Docker Container	1439	System	Healthy	CUG1	Active	[Icons]
2. + gke-gibson-guestbook-default-pool-19	--	Servers	Kubernetes Node	1394	System	Healthy	CUG1	Active	[Icons]
3. gke-gibson-guestbook-default-pool-19	--	Servers	Kubernetes Node	1396	System	Healthy	CUG1	Active	[Icons]

- The **Device Component Map** page (Views > 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.



Relationships Between Component Devices

In addition to the parent/child relationships between component devices, relationships are automatically created by the Dynamic Applications in the *Kubernetes PowerPack* between each controller device and its underlying Docker container.

Chapter

3

Dashboards

Overview

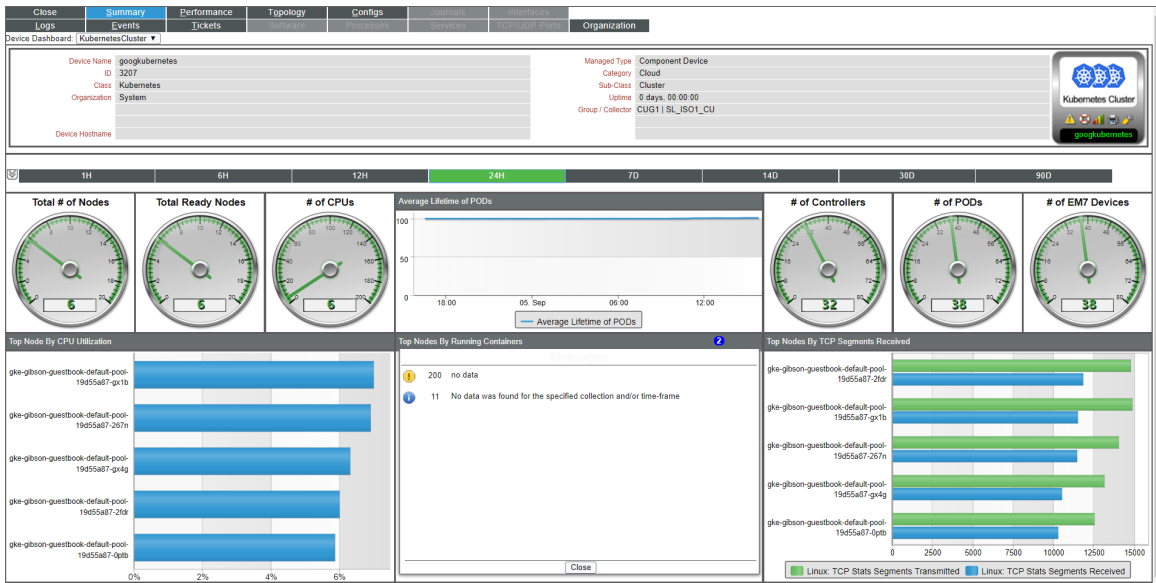
The following sections describe the device dashboards that are included in the *Kubernetes* PowerPack:

<i>Device Dashboards</i>	30
<i>Kubernetes Cluster</i>	31
<i>Kubernetes Node</i>	32
<i>Kubernetes Namespace</i>	33
<i>Kubernetes Controllers</i>	34
<i>Docker: Container</i>	35

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.

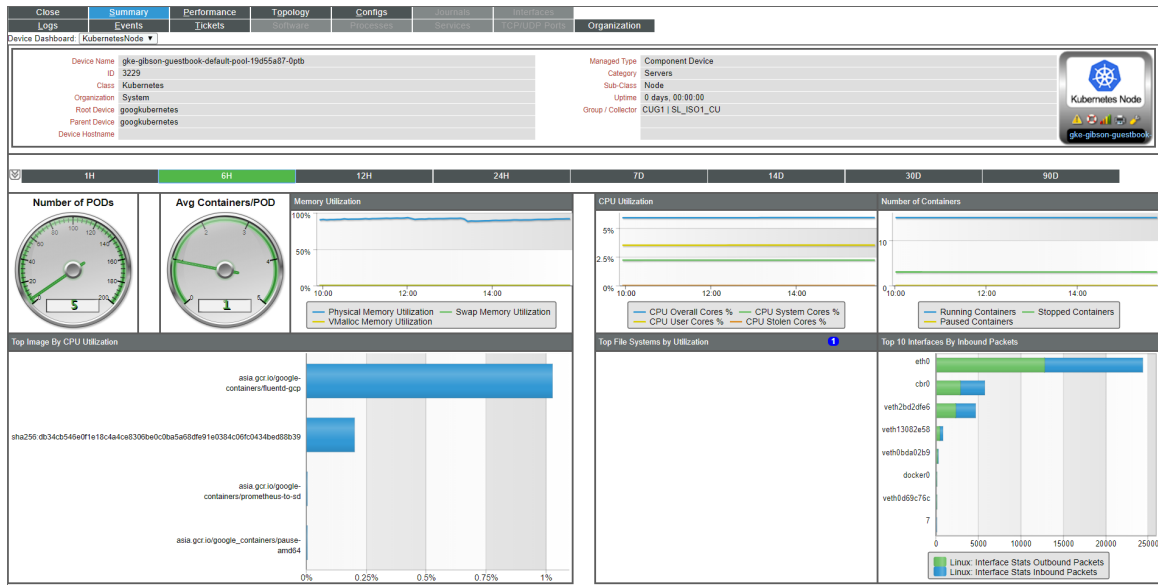
Kubernetes Cluster



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 running containers
- Top nodes, sorted by the number of TCP segments received

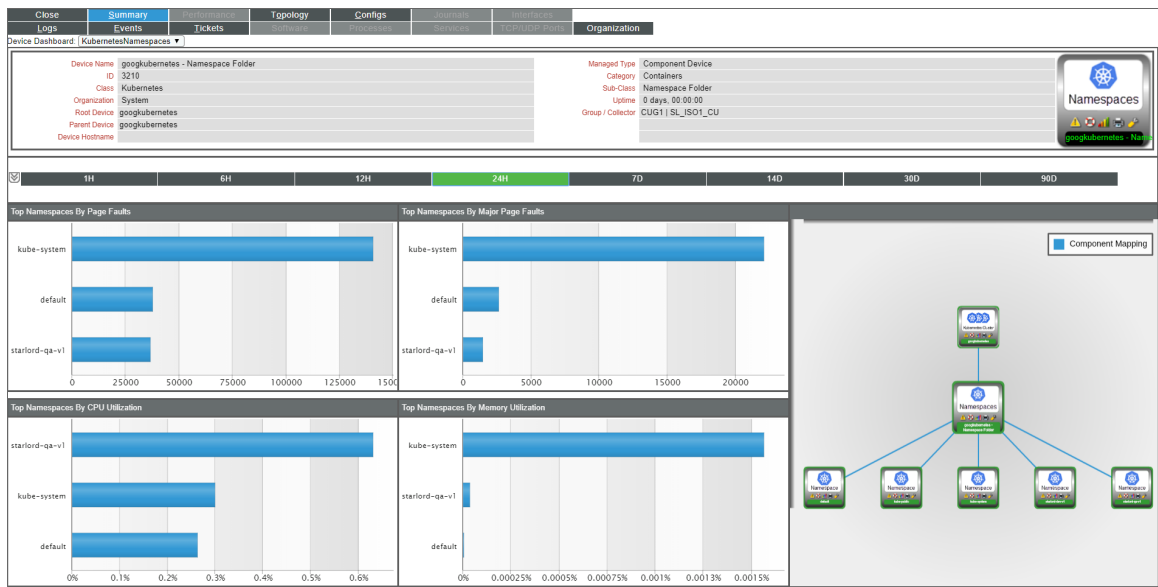
Kubernetes Node



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
- Number of containers
- Top images, sorted by CPU utilization
- Top file systems, sorted by utilization
- Top 10 interfaces, sorted by the number of inbound packets.

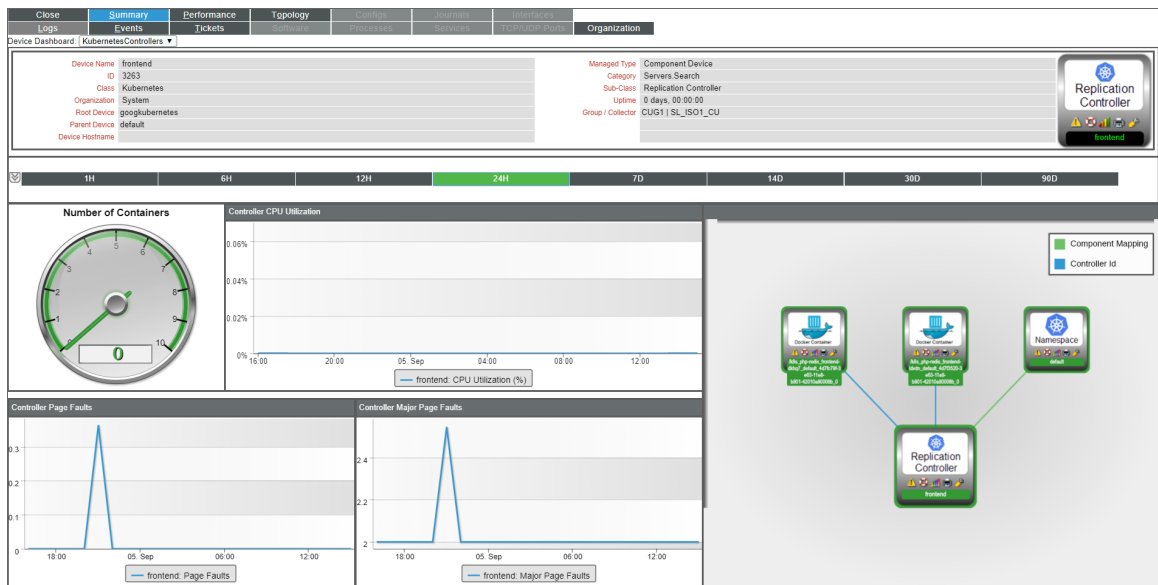
Kubernetes Namespace



The Kubernetes Namespace device dashboard displays the following information:

- The basic information about the device
- Top namespaces, sorted by the number of page faults
- Top namespaces, sorted by the number of major page faults
- Top namespaces, sorted by CPU utilization
- Top namespaces, sorted by memory utilization
- A dynamic component map showing the device's relationships

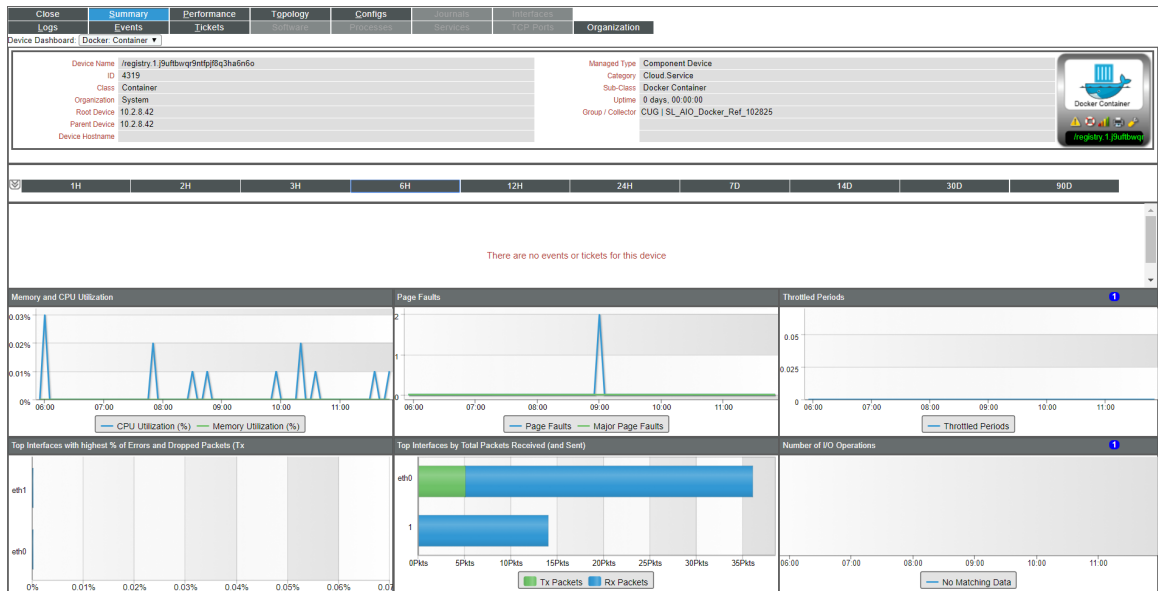
Kubernetes Controllers



The Kubernetes Controllers device dashboard displays the following information:

- The basic information about the device
- Number of active containers
- Controller CPU utilization
- Controller page faults
- Controller major page faults
- A dynamic component map showing the device's relationships

Docker: Container



The Docker: Container device dashboard displays the following information:

- Events and tickets for the device
- Memory and CPU utilization over a specified period of time
- Page faults over a specified period of time
- Throttled periods over a specified period of time
- Top interfaces with the highest percentage of errors and dropped packets over a specified period of time
- Top interfaces by total packets received over a specified period of time
- Number of input and output operations over a specified period of time

© 2003 - 2020, ScienceLogic, Inc.

All rights reserved.

LIMITATION OF LIABILITY AND GENERAL DISCLAIMER

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

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

Copyrights and Trademarks

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

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

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

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

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

Other

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

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



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

International: +1-703-354-1010