



Monitoring Docker

Beta Version

Docker PowerPack version 104

Table of Contents

Overview	3
What is Docker?	3
What Does the Docker PowerPack Monitor?	4
Installing the Docker PowerPack	4
Configuring Docker Monitoring	6
Prerequisites for Monitoring Docker	6
Enabling the Docker API	7
Configuring a Docker Credential	9
Creating a Basic/Snippet Credential	9
Creating an SSH/Key Credential	10
Discovering Docker Components	11
Manually Aligning Dynamic Applications	13
Viewing Docker Component Devices	15
Relationships Between Component Devices	17
Docker Dashboards	18
Device Dashboards	18
Docker: Container	19
Docker: Host	20
Docker: Service	21
Docker: Stack	22
Docker: Swarm	23

Chapter 1

Overview

Introduction

This manual describes how to monitor the Docker platform in SL1 using the *Docker PowerPack*.

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

<i>What is Docker?</i>	3
<i>What Does the Docker PowerPack Monitor?</i>	4
<i>Installing the Docker PowerPack</i>	4

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 Docker?

Docker is a platform that automates the process of deploying applications using software containers. These containers include individual deployment components (e.g., software code, system libraries, etc.) that, when combined, contain everything needed to run the application.

What Does the Docker PowerPack Monitor?

The *Docker PowerPack* includes:

- An example Basic/Snippet Credential and an example SSH/Key Credential for discovering Docker devices
- Dynamic Applications and Run Book Actions to discover, model, and monitor the following Docker component devices:
 - Hosts
 - Containers
 - Swarms
 - Stacks
 - Services
- Device Classes for each type of Docker component device SL1 monitors
- Event Policies that are triggered when Docker component devices meet certain status criteria

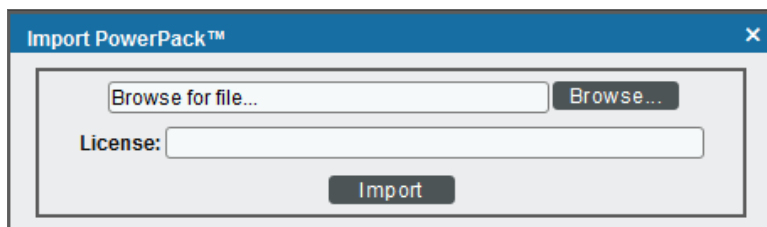
Installing the Docker PowerPack

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

TIP: By default, installing a new version of a PowerPack overwrites all content in 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 page appears, click the **[Install]** button to install the PowerPack.

NOTE: If you exit the **PowerPack Installer** modal page 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 page. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Configuring Docker Monitoring

Overview

The following sections describe how to configure and discover the Docker platform and its component devices for monitoring by SL1 using the *Docker PowerPack*:

<i>Prerequisites for Monitoring Docker</i>	6
<i>Enabling the Docker API</i>	7
<i>Configuring a Docker Credential</i>	9
<i>Creating a Basic/Snippet Credential</i>	9
<i>Creating an SSH/Key Credential</i>	10
<i>Discovering Docker Components</i>	11
<i>Manually Aligning Dynamic Applications</i>	13
<i>Viewing Docker Component Devices</i>	15
<i>Relationships Between Component Devices</i>	17

Prerequisites for Monitoring Docker

Before you can monitor the Docker platform and its component devices in SL1 using the *Docker PowerPack*, you must first follow the instructions in the [Enabling the Docker API](#) section. These steps enable the Dynamic Applications in the *Docker PowerPack* to communicate with and gather data from the Docker API.

If you are using Secure Shell (SSH) to monitor Docker or Kubernetes nodes in conjunction with the *Kubernetes PowerPack*, you must also install cURL 7.40 or greater on all of the Docker hosts that you want to monitor, prior to discovery. You must then run the following cURL commands on each of those hosts:

- `curl --unix-socket /var/run/docker.sock http://docker/containers/json`
- `curl --unix-socket /var/run/docker.sock http://docker/containers/[container_id]/json`
- `curl --unix-socket /var/run/docker.sock http://docker/containers/[container_id]/stats?stream=0`

Enabling the Docker API

Before you discover Docker components using the *Docker PowerPack*, you must first enable the Docker API. This section describes how to do so for Windows, CentOS, Red Hat Enterprise Linux (RHEL), and Oracle Linux operating systems.

Windows

To enable the Docker API for Windows using the Docker Toolbox:

1. Start Docker Quickstart Terminal.
2. To determine the IP address of the Docker host machine, type the following command:

```
$ docker-machine ip
```

3. Log in to the host machine:

```
$ docker-machine ssh
```

4. Navigate to Boot2Docker:

```
$ cd /var/lib/boot2docker
```

5. Edit the Boot2Docker profile:

```
$ sudo vi profile
```

6. In the profile, change "DOCKER_HOST" to "DOCKER_HOST='-H tcp://0.0.0.0:[port number]'", and set DOCKER_TLS=no.

7. Exit the SSH session, and then restart Docker:

```
$ exit
$ docker-machine restart
```

8. To verify that the Docker API is accessible, open a browser and navigate to `http://[IP address]:[port number]/version`.

If the Docker API is successfully enabled, the version returns something similar to the following:

```
{ "Version": "17.10.0-ce", "ApiVersion": "1.33", "MinAPIVersion": "1.12", "GitCommit": "f4ffd25", "GoVersion": "gol.8.3", "Os": "linux", "Arch": "amd64", "KernelVersion": "4.4.93-boot2docker", "BuildTime": "2017-10-17T19:05:23.000000000+00:00" }
```

CentOS

To enable the Docker API for CentOS:

1. Log in to the command-line interface of the server running Docker and navigate to systemd/system:

```
$ cd /etc/systemd/system
```

2. Create a new "docker.service.d" folder, then navigate to that folder:

```
$ mkdir docker.service.d
$ cd docker.service.d
```

3. Create a new docker.conf file:

```
$ vi docker.conf
```

4. Type the following:

```
INSERT
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:[port number] -H
unix://var/run/docker.sock
```

5. Reload daemon, restart Docker, and open the port on the firewall by typing the following:

```
$ systemctl daemon-reload
$ systemctl restart docker
$ firewall-cmd --add-port=[port number]/tcp
```

6. Verify that the Docker API is accessible by typing the following:

```
$ *curl http://localhost:[port number]/version*
```

If the Docker API is successfully enabled, the version returns something similar to the following:

```
{"Version":"17.06.1-ce","ApiVersion":"1.30","MinAPIVersion":"1.12","GitCommit":
:"874a737","GoVersion":"go1.8.3","Os":"linux","Arch":"amd64","KernelVersion":
:"3.10.0-514.26.2.el7.x86_64","BuildTime":"2017-08-17T23:01:50.155177940+00:00"}
```

RHEL 7 and Oracle Linux 7

To enable the Docker API for RHEL 7 or Oracle Linux 7:

1. Log in to the command-line interface of the server running Docker and navigate to systemd/system:

```
$ cd /usr/lib/systemd/system
```

2. Edit the service.docker file:

```
$ sudo vi service.docker
```


3. Find the line that starts with "ExecStart=/usr/bin/dockerd" and add "-H tcp://0.0.0.0: [port number]" so that the updated line looks like this:

```
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243
```

4. Reload daemon, restart Docker, and open the port on the firewall by typing the following:

```
$ systemctl daemon-reload
$ systemctl restart docker
$ firewall-cmd --add-port=[port number]/tcp
```

5. Verify that the Docker API is accessible by typing the following:

```
$ *curl http://[IP address]:[port number]/version*
```

If the Docker API is successfully enabled, the version returns something similar to the following:

```
{"Version": "17.06.2-ee-4", "ApiVersion": "1.30", "MinAPIVersion": "1.12", "GitCommit": "dd2c358", "GoVersion": "go1.8.3", "Os": "linux", "Arch": "amd64", "KernelVersion": "3.10.0-514.el7.x86_64", "BuildTime": "2017-10-12T16:19:56.386620861+00:00"}
```


Configuring a Docker Credential

The *Docker PowerPack* includes an example Basic/Snippet Credential and an example SSH/Key Credential for your use. You can modify these to create your own Credentials that will enable SL 1 to discover your Docker devices.

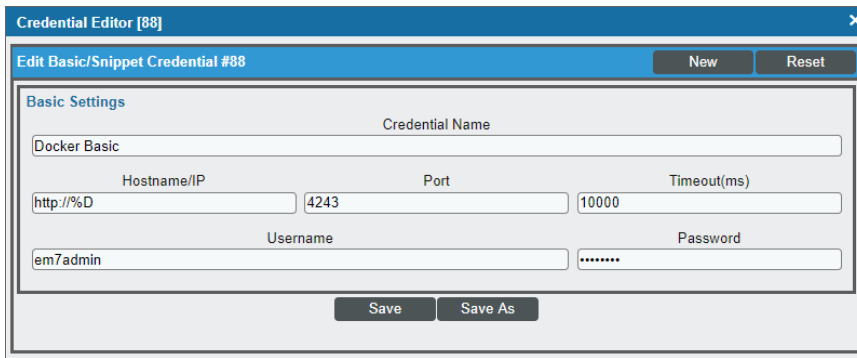
Creating a Basic/Snippet Credential

To configure SL 1 to monitor the Docker platform using the Docker API, you must create a Basic/Snippet credential that allows the Dynamic Applications in the *Docker PowerPack* to connect with Docker hosts or swarms. An example Basic/Snippet credential that you can edit for your own use is included in the *Docker PowerPack*.

To create a Basic/Snippet credential to access Docker hosts or swarms:

1. Go to the **Credential Management** page (System > Manage > Credentials).
2. Locate the example **Docker Basic** credential, and then click its wrench icon (). The **Edit Basic/Snippet Credential** modal page appears.

3. Complete the following fields:



The screenshot shows a window titled "Credential Editor [88]". Inside, there's a sub-window "Edit Basic/Snippet Credential #88" with "New" and "Reset" buttons. The "Basic Settings" section contains the following fields:

- Credential Name:** Docker Basic
- Hostname/IP:** http://%D
- Port:** 4243
- Timeout(ms):** 10000
- Username:** em7admin
- Password:** (masked with dots)

At the bottom, there are "Save" and "Save As" buttons.

- **Credential Name.** Type a new name for the Docker credential.
- **Hostname/IP.** Type "%D".
- **Port.** Type the port number you specified when you [enabled the Docker API](#).
- **Timeout(ms).** Type "10000".
- **Username.** Type a value for the username.
- **Password.** Type a value for the password.

NOTE: The Docker platform does not require a specific username and password to access the platform, but SL1 does require the **Username** and **Password** fields to have values when using Basic/Snippet credentials to monitor Docker. Therefore, those fields must have entries, but the values themselves do not matter.

4. Click the **[Save As]** button.
5. When the confirmation message appears, click **[OK]**.

Creating an SSH/Key Credential

If you are using SSH to monitor Docker swarms, then you must create an SSH/Key credential that allows the Dynamic Applications in the *Docker PowerPack* to connect with Docker swarms. An example SSH/Key credential that you can edit for your own use is included in the *Docker PowerPack*.

NOTE: You can also use an SSH credential in conjunction with the *Kubernetes PowerPack* to monitor the Docker infrastructure for a Kubernetes cluster.

To create an SSH/Key credential to monitor Docker containers:

1. Go to the **Credential Management** page (System > Manage > Credentials).
2. Locate the example **Docker Basic - SSH** credential, and then click its wrench icon (🔧). The **Edit SSH/Key Credential** modal page appears.
3. Complete the following fields:

The screenshot shows a 'Credential Editor' window with the following fields and values:

- Credential Name:** Docker Basic - ssh
- Hostname/IP:** %D
- Port:** 22
- Timeout(ms):** 10000
- Username:** sl1admin
- Password:** [masked]
- Private Key (PEM Format):** [blank]

Buttons: New, Reset, Save, Save As

- **Credential Name.** Type a new name for the Docker credential.
 - **Hostname/IP.** Type "%D".
 - **Port.** Type the SSH port number for the Docker swarm you want to monitor.
 - **Timeout(ms).** Type "10000".
 - **Username.** Type the username for a user with SSH access to the Docker swarm command line interface.
 - **Password.** Type the user's password.
 - **Private Key (PEM Format).** Keep this field blank.
4. Click the **[Save As]** button.
 5. When the confirmation message appears, click **[OK]**.

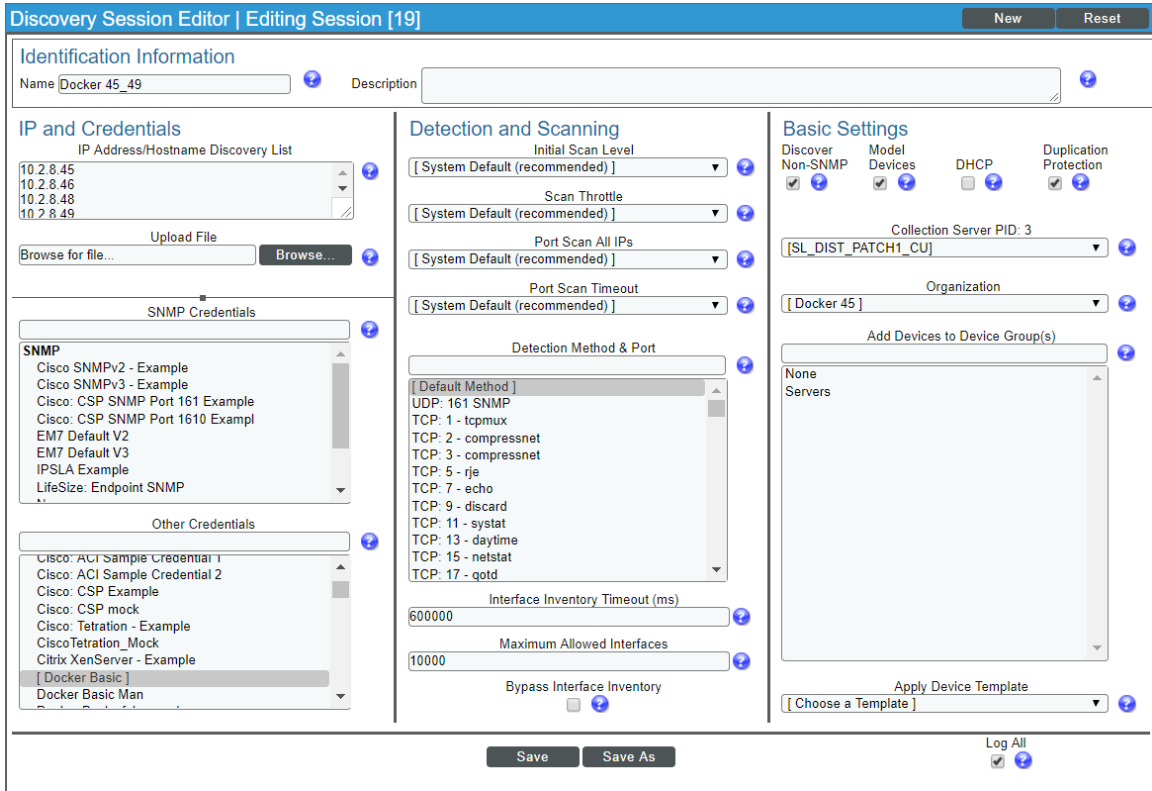
Discovering Docker Components

To discover and model your Docker component devices for monitoring, you must run a discovery session. The discovery session will discover the Docker hosts and swarms that SL1 will use as the root devices for monitoring the Docker components.

Several minutes after the discovery session has completed, the Dynamic Applications in the Docker PowerPack will automatically align to the Docker root devices. These Dynamic Applications will discover, model, and monitor the remaining components in your Docker system.

To discover Docker components, perform the following steps:



1. Go to the **Discovery Control Panel** page (System > Manage > Discovery), and then click the **[Create]** button. The **Discovery Session Editor** page appears.
2. In the **Discovery Session Editor** page, complete the following fields:



- **Name.** Type a name for your discovery session.
- **IP Address/Hostname Discovery List.** Type the IP addresses for all of the Docker hosts in the swarm that you want to discover.


NOTE: Swarms are created only when the swarm leader is discovered.

- **Other Credentials.** Select the **Basic/Snippet or SSH/Key credential(s)** you created for Docker.
- **Discover Non-SNMP.** Select this checkbox.
- **Model Devices.** Select this checkbox.

3. Optionally, you can enter values in the other fields on this page. For more information about the other fields on this page, see the **Discovery & Credentials** manual.
4. Click the **[Save]** button to save the discovery session, and then close the **Discovery Session Editor** window.
5. The discovery session you created displays at the top of the **Discovery Control Panel** page. Click its lightning-bolt icon () to run the discovery session.
6. The **Discovery Session** window appears. When a root device is discovered, click its device icon () to view the **Device Properties** page for that device.

Manually Aligning Dynamic Applications

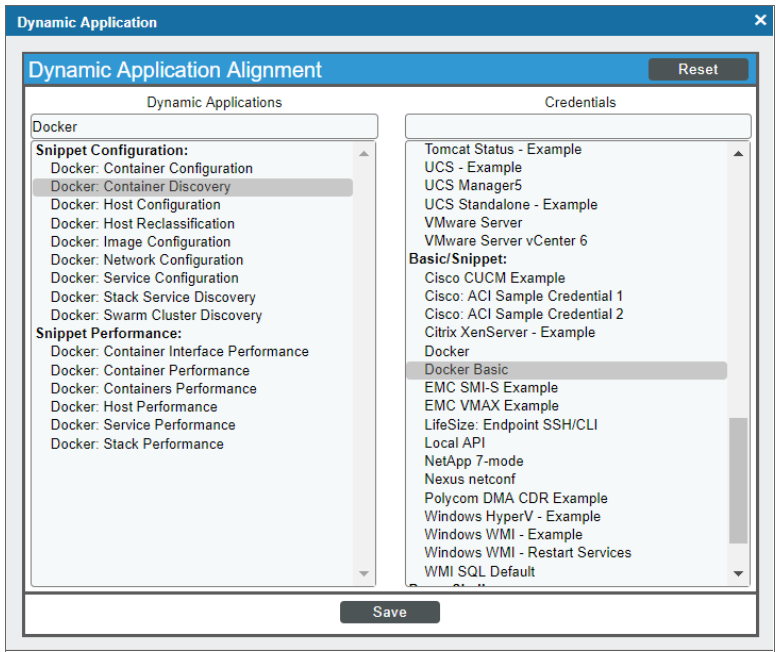
To verify that SL1 has automatically aligned the correct Dynamic Applications during discovery:

1. From the **Device Properties** page (Registry > Devices > wrench icon () for the Docker root device, click the **[Collections]** tab. The **Dynamic Application Collections** page appears.
2. The following Dynamic Applications should appear in the list of aligned Dynamic Applications:
 - For Docker Hosts:
 - Docker: Container Discovery
 - Docker: Containers Performance
 - Docker: Host Configuration
 - Docker: Host Performance
 - Docker: Host Reclassification
 - Docker: Image Configuration
 - Docker: Image Performance
 - Docker: Network Configuration
 - Docker: Swarm Cluster Discovery
 - For Docker Swarms:
 - Docker: Stack Discovery
 - Docker: Swarm Configuration
 - Docker: Swarm Performance
 - Docker: Swarm Service Discovery

NOTE: It can take several minutes after discovery for Dynamic Applications to display on the **Dynamic Application Collections** page. If the listed Dynamic Applications do not display on this page, try clicking the **[Reset]** button.

If the Dynamic Applications have not been automatically aligned, you can align them manually. To do so, perform the following steps:

1. Go to the **Device Properties** page (Registry > Devices > wrench icon(🔧)) for the Docker root device and click the **[Collections]** tab. The **Dynamic Application Collections** page appears.
2. On the **Dynamic Application Collections** page, click the **[Action]** button and then select *Add Dynamic Application* from the menu. The **Dynamic Application Alignment** page appears.

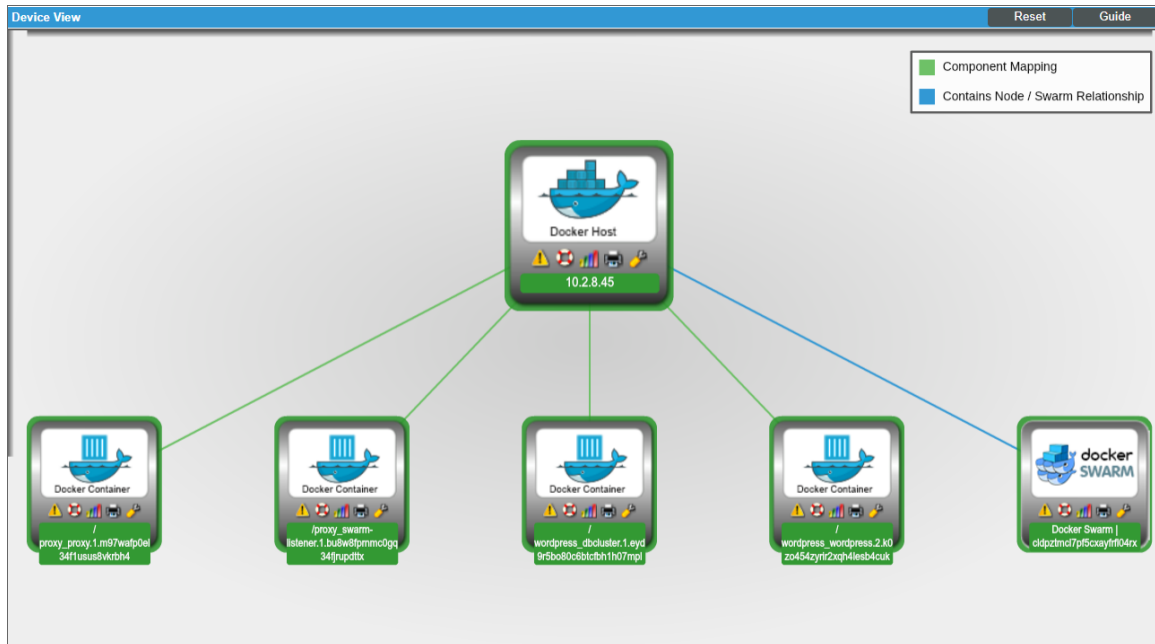


3. In the **Dynamic Applications** field, select a Dynamic Application to align.
4. In the **Credentials** field, select the **Basic/Snippet credential** you created for Docker.
5. Click the **[Save]** button.
6. Repeat steps 2-5 as needed to align any additional Dynamic Applications.

Viewing Docker Component Devices

In addition to the **Device Manager** page (Registry > Devices > Device Manager), you can view the Docker platform and all of its component devices in the following places in the user interface:

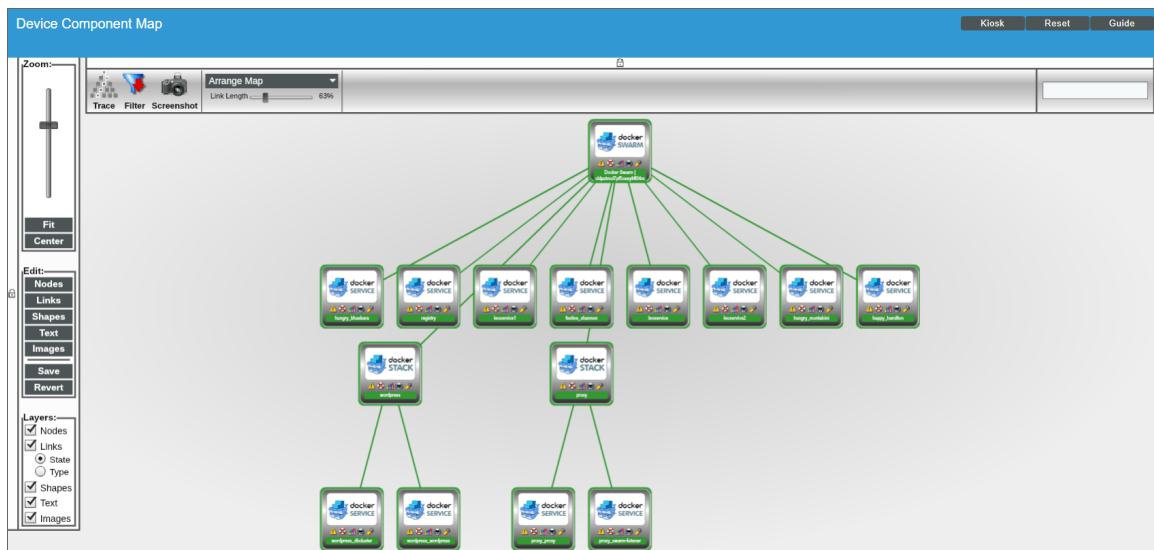
- The **Device View** modal page (Registry > Devices > Device Manager > graph icon > Topology) displays a map of a particular device and all of the devices with which it has parent-child relationships. Double-clicking any of the devices listed reloads the page to make the selected device the primary 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 Docker, find the Docker Host or Docker Swarm device and click its plus icon (+):

Device Name	IP Address	Device Category	Device Class Sub-class	IID	Organization	Current State	Collection Group	Collection State																																																																																											
10.2.8.45	10.2.8.45	Compute	Host Docker Host	4112	Docker 45	Healthy	CUG1	Active																																																																																											
<table border="1"> <thead> <tr> <th>Device Name</th> <th>IP Address</th> <th>Device Category</th> <th>Device Class Sub-class</th> <th>IID</th> <th>Organization</th> <th>Current State</th> <th>Collection Group</th> <th>Collection State</th> </tr> </thead> <tbody> <tr> <td>/proxy_proxy.1.m97wafp0e1341fus8</td> <td>--</td> <td>Service</td> <td>Container Docker Container</td> <td>4118</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>/proxy_swarm_listener.1.bu6w8fprmc</td> <td>--</td> <td>Service</td> <td>Container Docker Container</td> <td>4117</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>/wordpress_dbcluster.1.eyd9f5bo80cd</td> <td>--</td> <td>Service</td> <td>Container Docker Container</td> <td>4115</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>/wordpress_wordpress.2.k0zo454zyr</td> <td>--</td> <td>Service</td> <td>Container Docker Container</td> <td>4116</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> </tbody> </table>										Device Name	IP Address	Device Category	Device Class Sub-class	IID	Organization	Current State	Collection Group	Collection State	/proxy_proxy.1.m97wafp0e1341fus8	--	Service	Container Docker Container	4118	Docker 45	Healthy	CUG1	Active	/proxy_swarm_listener.1.bu6w8fprmc	--	Service	Container Docker Container	4117	Docker 45	Healthy	CUG1	Active	/wordpress_dbcluster.1.eyd9f5bo80cd	--	Service	Container Docker Container	4115	Docker 45	Healthy	CUG1	Active	/wordpress_wordpress.2.k0zo454zyr	--	Service	Container Docker Container	4116	Docker 45	Healthy	CUG1	Active																																													
Device Name	IP Address	Device Category	Device Class Sub-class	IID	Organization	Current State	Collection Group	Collection State																																																																																											
/proxy_proxy.1.m97wafp0e1341fus8	--	Service	Container Docker Container	4118	Docker 45	Healthy	CUG1	Active																																																																																											
/proxy_swarm_listener.1.bu6w8fprmc	--	Service	Container Docker Container	4117	Docker 45	Healthy	CUG1	Active																																																																																											
/wordpress_dbcluster.1.eyd9f5bo80cd	--	Service	Container Docker Container	4115	Docker 45	Healthy	CUG1	Active																																																																																											
/wordpress_wordpress.2.k0zo454zyr	--	Service	Container Docker Container	4116	Docker 45	Healthy	CUG1	Active																																																																																											
<table border="1"> <thead> <tr> <th>Device Name</th> <th>IP Address</th> <th>Device Category</th> <th>Device Class Sub-class</th> <th>IID</th> <th>Organization</th> <th>Current State</th> <th>Collection Group</th> <th>Collection State</th> </tr> </thead> <tbody> <tr> <td>festive_shannon</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4122</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>happy_hamilton</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4126</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>hungry_bhaskara</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4119</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>hungry_montalcini</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4125</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>jeoservice</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4123</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>jeoservice1</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4121</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>jeoservice2</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4124</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>proxy</td> <td>--</td> <td>Service</td> <td>Stack Docker Stack</td> <td>4128</td> <td>Docker 45</td> <td>Healthy</td> <td>CUG1</td> <td>Active</td> </tr> <tr> <td>registry</td> <td>--</td> <td>Service</td> <td>Service Docker Service</td> <td>4120</td> <td>Docker 45</td> <td>Healthy</td> <td>PI IP1</td> <td>Active</td> </tr> </tbody> </table>										Device Name	IP Address	Device Category	Device Class Sub-class	IID	Organization	Current State	Collection Group	Collection State	festive_shannon	--	Service	Service Docker Service	4122	Docker 45	Healthy	CUG1	Active	happy_hamilton	--	Service	Service Docker Service	4126	Docker 45	Healthy	CUG1	Active	hungry_bhaskara	--	Service	Service Docker Service	4119	Docker 45	Healthy	CUG1	Active	hungry_montalcini	--	Service	Service Docker Service	4125	Docker 45	Healthy	CUG1	Active	jeoservice	--	Service	Service Docker Service	4123	Docker 45	Healthy	CUG1	Active	jeoservice1	--	Service	Service Docker Service	4121	Docker 45	Healthy	CUG1	Active	jeoservice2	--	Service	Service Docker Service	4124	Docker 45	Healthy	CUG1	Active	proxy	--	Service	Stack Docker Stack	4128	Docker 45	Healthy	CUG1	Active	registry	--	Service	Service Docker Service	4120	Docker 45	Healthy	PI IP1	Active
Device Name	IP Address	Device Category	Device Class Sub-class	IID	Organization	Current State	Collection Group	Collection State																																																																																											
festive_shannon	--	Service	Service Docker Service	4122	Docker 45	Healthy	CUG1	Active																																																																																											
happy_hamilton	--	Service	Service Docker Service	4126	Docker 45	Healthy	CUG1	Active																																																																																											
hungry_bhaskara	--	Service	Service Docker Service	4119	Docker 45	Healthy	CUG1	Active																																																																																											
hungry_montalcini	--	Service	Service Docker Service	4125	Docker 45	Healthy	CUG1	Active																																																																																											
jeoservice	--	Service	Service Docker Service	4123	Docker 45	Healthy	CUG1	Active																																																																																											
jeoservice1	--	Service	Service Docker Service	4121	Docker 45	Healthy	CUG1	Active																																																																																											
jeoservice2	--	Service	Service Docker Service	4124	Docker 45	Healthy	CUG1	Active																																																																																											
proxy	--	Service	Stack Docker Stack	4128	Docker 45	Healthy	CUG1	Active																																																																																											
registry	--	Service	Service Docker Service	4120	Docker 45	Healthy	PI IP1	Active																																																																																											

- The **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. The platform also updates each map with the latest status and event information. To view the map for Docker, go to the **Device Component Map** page (Views > Device Maps > Components) 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 parent/child relationships between component devices, SL1 also creates relationships between the following component devices:

- Swarms and Nodes
- Services and Containers

You can also use the *Docker PowerPack* in conjunction with the *Kubernetes PowerPack* when monitoring Kubernetes systems. When you do so, SL1 creates relationships between Docker Swarms and Containers and their underlying Kubernetes Nodes.

Chapter

3

Docker Dashboards

Overview

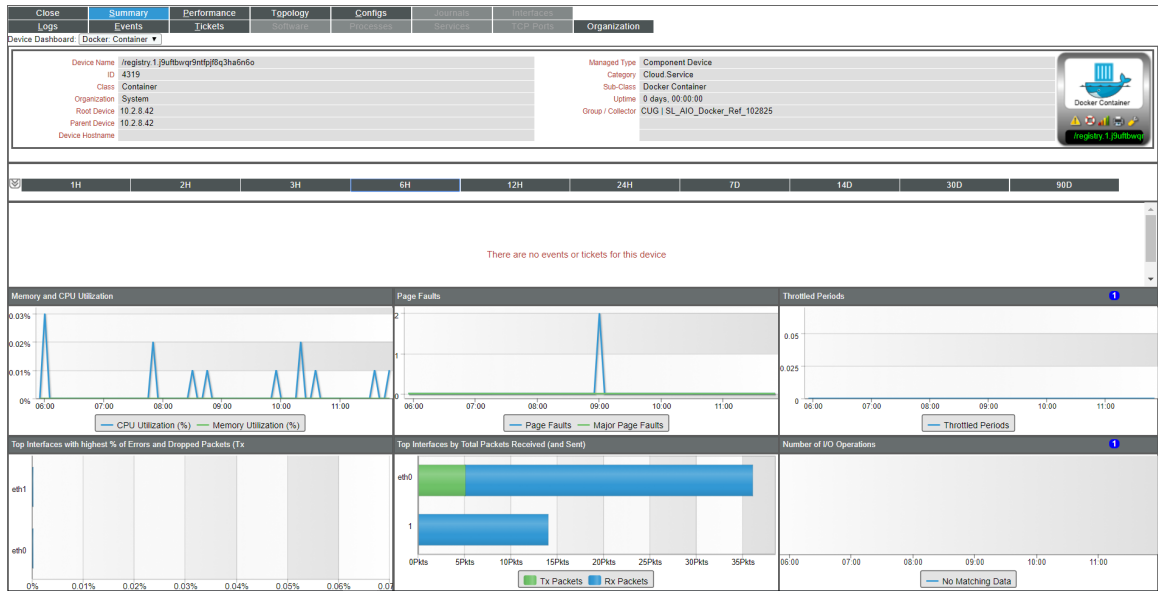
The following sections describe the device dashboards that are included in the *Docker PowerPack*:

<i>Device Dashboards</i>	18
<i>Docker: Container</i>	19
<i>Docker: Host</i>	20
<i>Docker: Service</i>	21
<i>Docker: Stack</i>	22
<i>Docker: Swarm</i>	23

Device Dashboards

The *Docker PowerPack* includes device dashboards that provide summary information for Docker component devices. Each of the device dashboards in the *Docker PowerPack* is set as the default device dashboard for the equivalent device class.

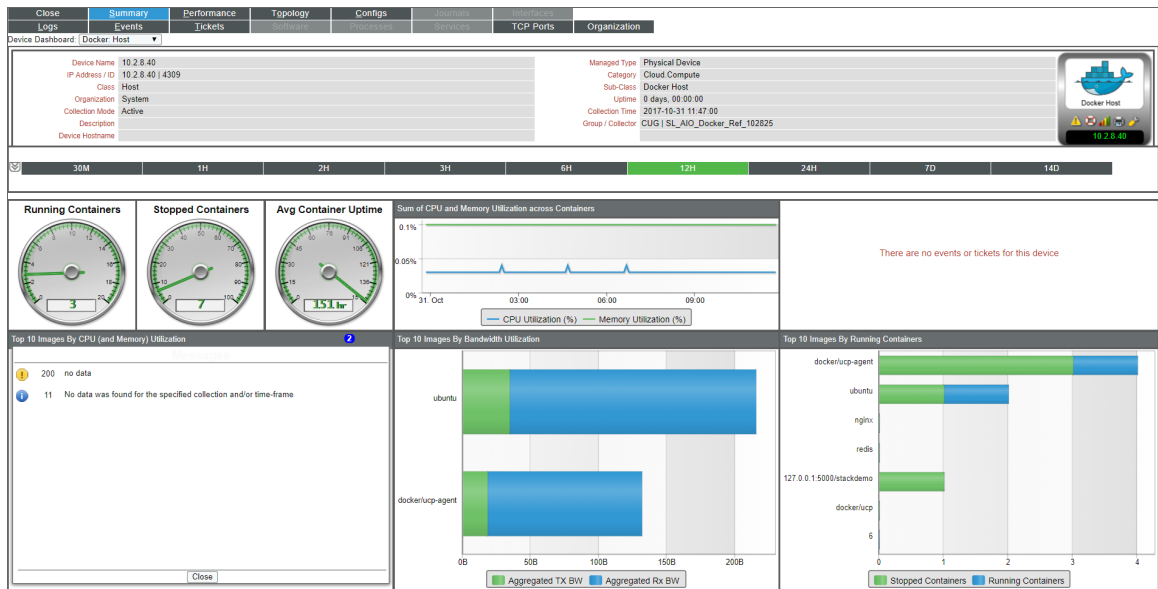
Docker: Container



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

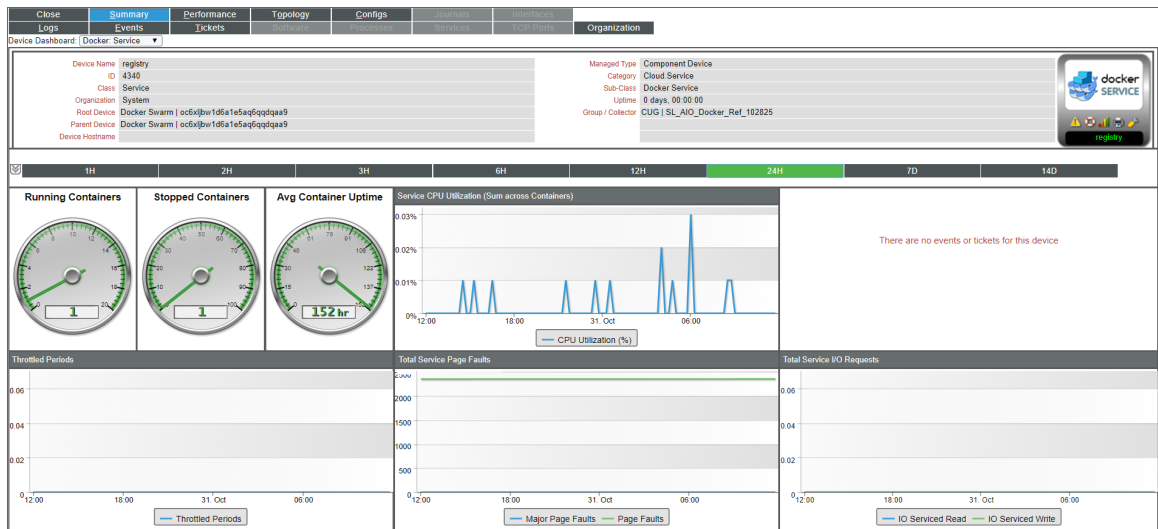
Docker: Host



The Docker: Host device dashboard displays the following information:

- Containers running and stopped over a specified period of time, as well as average container uptime
- Total CPU utilization and memory across all containers over a specified period of time
- Events and tickets for the device
- Top 10 images by CPU and memory utilization over a specified period of time
- Top 10 images by bandwidth utilization over a specified period of time
- Top 10 images by running containers over a specified period of time

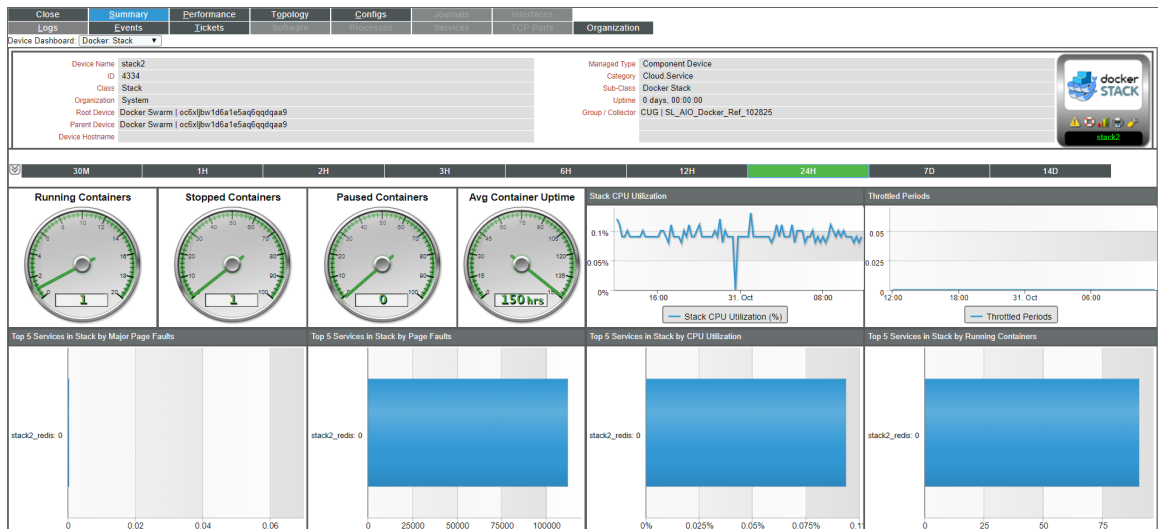
Docker: Service



The Docker: Service device dashboard displays the following information:

- Containers running and stopped over a specified period of time, as well as average container uptime
- Service CPU utilization over a specified period of time
- Events and tickets for the device
- Throttled periods over a specified period of time
- Top service page faults over a specified period of time
- Total service input and output requests over a specified period of time

Docker: Stack



The Docker: Stack device dashboard displays the following information:

- Containers running, stopped, and paused over a specified period of time, as well as average container uptime
- Stack CPU utilization over a specified period of time
- Throttled periods over a specified period of time
- Top 5 services in the stack over a specified period of time, based on the number of major page faults
- Top 5 services in the stack over a specified period of time, based on the number of page faults
- Top 5 services in the stack over a specified period of time, based on CPU utilization
- Top 5 services in the stack over a specified period of time, based on the number of running containers

Docker: Swarm



The Docker: Swarm device dashboard displays the following information:

- Containers running, stopped, and paused over a specified period of time, as well as average container uptime
- Number of stacks, services, images, total nodes, and available nodes over a specified period of time
- Top 5 stacks in the swarm over a specified period of time, based on CPU utilization
- Top 5 services in the swarm over a specified period of time, based on CPU utilization
- Top 5 services in the swarm over a specified period of time, based on the number of major page faults and total page faults
- Overall swarm CPU utilization
- Overall swarm memory utilization
- Overall swarm input and output

© 2003 - 2019, 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