Quest™

Foglight® for Container Management 2.0.0

User and Administration Guide

© 2019 Quest Software Inc.

ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software Inc.

The information in this document is provided in connection with Quest Software products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Quest Software products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST SOFTWARE ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL QUEST SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF QUEST SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Quest Software makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Quest Software does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept.
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our website (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal.

Trademarks

Quest, the Quest logo, and Join the Innovation are trademarks and registered trademarks of Quest Software Inc. For a complete list of Quest marks, visit https://www.quest.com/legal/trademark-information.aspx. "Apache HTTP Server", Apache, "Apache Tomcat" and "Tomcat" are trademarks of the Apache Software Foundation. Google is a registered trademark of Google Inc. Android, Chrome, Google Play, and Nexus are trademarks of Google Inc. Red Hat, JBoss, the JBoss logo, and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the U.S. and other countries. CentOS is a trademark of Red Hat, Inc. in the U.S. and other countries. Fedora and the Infinity design logo are trademarks of Red Hat, Inc. Microsoft, .NET, Active Directory, Internet Explorer, Hyper-V, Office 365, SharePoint, Silverlight, SQL Server, Visual Basic, Windows, Windows Vista and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. AIX, IBM, PowerPC, PowerVM, and WebSphere are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Java, Oracle, Oracle Solaris, PeopleSoft, Siebel, Sun, WebLogic, and ZFS are trademarks or registered trademarks of Oracle and/or its affiliates in the United States and other countries. SPARC is a registered trademark of SPARC International, Inc. in the United States and other countries. Products bearing the SPARC trademarks are based on an architecture developed by Oracle Corporation. OpenLDAP is a registered trademark of the OpenLDAP Foundation. HP is a registered trademark that belongs to Hewlett-Packard Development Company, L.P. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. MySQL is a registered trademark of MySQL AB in the United States, the European Union and other countries. Novell and eDirectory are registered trademarks of Novell, Inc., in the United States and other countries. VMware, ESX, ESXi, vSphere, vCenter, vMotion, and vCloud Director are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. Sybase is a registered trademark of Sybase, Inc. The X Window System and UNIX are registered trademarks of The Open Group. Mozilla and Firefox are registered trademarks of the Mozilla Foundation. "Eclipse", "Eclipse Foundation Member", "EclipseCon", "Eclipse Summit", "Built on Eclipse", "Eclipse Ready" "Eclipse Incubation", and "Eclipse Proposals" are trademarks of Eclipse Foundation, Inc. IOS is a registered trademark or trademark of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. Apple, iPad, iPhone, Mac OS, Safari, Swift, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Ubuntu is a registered trademark of Canonical Ltd. Symantec and Veritas are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. OpenSUSE, SUSE, and YAST are registered trademarks of SUSE LCC in the United States and other countries. Citrix, AppFlow, NetScaler, XenApp, and XenDesktop are trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries. AlertSite and DéjàClick are either trademarks or registered trademarks of Boca Internet Technologies, Inc. Samsung, Galaxy S, and Galaxy Note are registered trademarks of Samsung Electronics America, Inc. and/or its related entities. MOTOROLA is a registered trademarks of Motorola Trademark Holdings, LLC. The Trademark BlackBerry Bold is owned by Research In Motion Limited and is registered in the United States and may be pending or registered in other countries. Quest is not endorsed, sponsored, affiliated with or otherwise authorized by Research In Motion Limited. Ixia and the Ixia four-petal logo are registered trademarks or trademarks of Ixia. Opera, Opera Mini, and the O logo are trademarks of Opera Software ASA. Tevron, the Tevron logo, and CitraTest are registered trademarks of Tevron, LLC. PostgreSQL is a registered trademark of the PostgreSQL Global Development Group. MariaDB is a trademark or registered trademark of MariaDB Corporation Ab in the European Union and United States of America and/or other countries. Vormetric is a registered trademark of Vormetric, Inc. Intel, Itanium, Pentium, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries. Debian is a registered trademark of Software in the Public Interest, Inc. OpenStack is a trademark of the OpenStack Foundation. Amazon Web Services, the "Powered by Amazon Web Services" logo, and "Amazon RDS" are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries. Infobright, Infobright Community Edition and Infobright Enterprise Edition are trademarks of Infobright Inc. POLYCOM®, RealPresence® Collaboration Server, and RMX® are registered trademarks of Polycom, Inc. All other trademarks and registered trademarks are property of their respective

owners.

Legend

WARNING: A WARNING icon indicates a potential for property damage, personal injury, or death.

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

IMPORTANT NOTE, **NOTE**, **TIP**, **MOBILE**, or **VIDEO:** An information icon indicates supporting information.

Foglight for Container Management User and Administration Guide
Updated - August 2019
Foglight Version - 5.9.5
Software Version - 2.0.0

Contents

Understanding Foglight for Container Management . . . 6
About Foglight for Container Management . . . 6
Architecture . . . 7
Sizing Your Monitored Environment . . . 7
Foglight Management Server Requirements . . . 8
Kubernetes Agent Requirements . . . 8
Docker Swarm Agent Requirements . . . 8
Getting Started . . . 9
Prerequisite . . . 9
Creating and Activating Agent . . . 19
Configuring data collection interval . . . 24

Using Foglight for Container Management . . . 25
Kubernetes . . . 26
Monitoring Kubernetes Pods . . . 26
Monitoring Kubernetes Nodes . . . 28
Monitoring Kubernetes Clusters . . . 32
Monitoring Kubernetes Other Components . . . 34
Alarms . . . 37
Capacity Management . . . 37
Optimizer . . . 39
Administration . . . 43
Docker Swarm . . . 44
Monitoring Docker Containers . . . 44
Monitoring Docker Hosts . . . 46
Monitoring Docker Swarm Clusters . . . 50
Monitoring Docker Swarm Services . . . 50
Alarms . . . 51
Analytics . . . 52
Kubernetes analytics . . . 52
Docker Swarm analytics . . . 56
Domains and Object Groups . . . 58
Domains . . . 58
Object Groups . . . 59
Metrics . . . 60
Kubernetes metrics . . . 60
Docker Swarm metrics . . . 61
Rules . . . 62
Kubernetes . . . 62
Docker Swarm . . . 71
Customization . . . 74

About Us . . . 78
We are more than just a name . . . 78

Our brand, our vision. Together. 78
Contacting Quest 78
Technical support resources 78

1

Understanding Foglight for Container Management

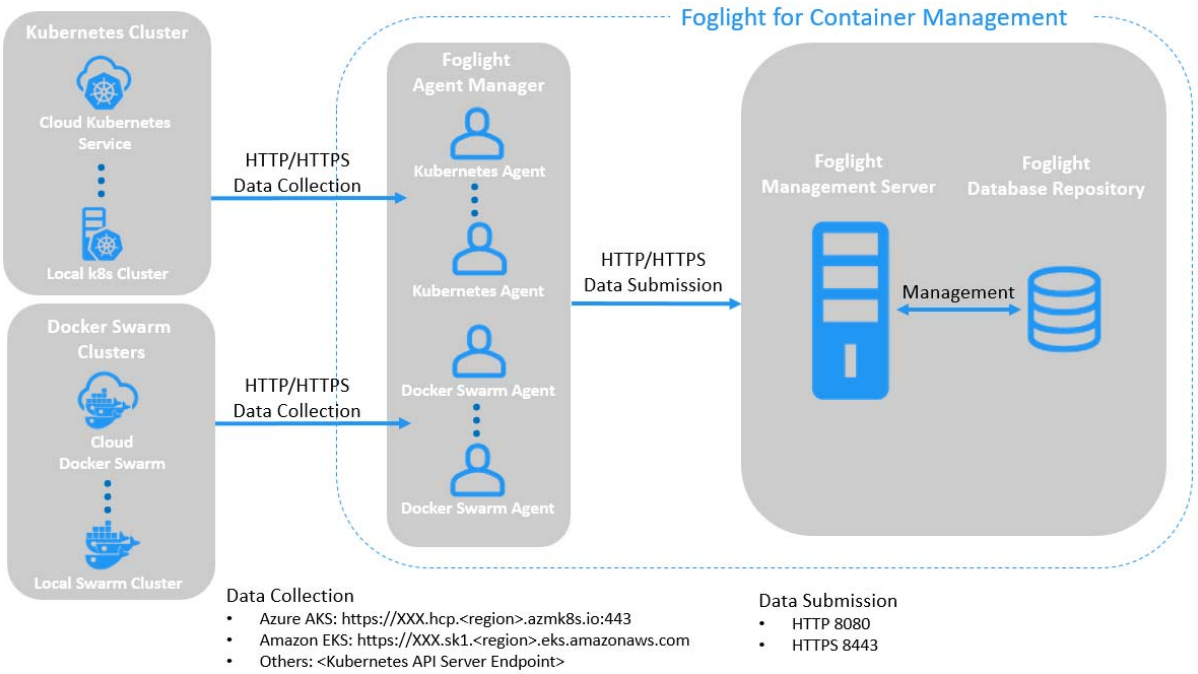
- About Foglight for Container Management
- Architecture
- Sizing Your Monitored Environment
  - Foglight Management Server Requirements
  - Kubernetes Agent Requirements
  - Docker Swarm Agent Requirements
- Getting Started
  - Prerequisite
  - Creating and Activating Agent
  - Configuring data collection interval

About Foglight for Container Management

Containers are a method of operating system virtualization that allow you to run an application and its dependencies in resource-isolated processes. Foglight® for Container Management simplifies this process by tracking each container, the resources it consumes, and the remaining compute of the container host, as well as providing you with the cluster information and pre-configured rules with notifications identifying the problem of your clusters.

Architecture

Figure 1. Components of Foglight for Container Management



Foglight for Container Management consists of three main components:

- Foglight Management Server and Foglight Database Repository — Responsible for managing, alerting, and viewing the collected data. Both components can be set to run on the same machine or reside on separate machines.
- Agent Manager — Hosts the monitoring Kubernetes agents.
- Docker Swarm clusters — Manages containerized applications in a clustered environment.
- Kubernetes clusters — Manages containerized applications in a clustered environment.

Sizing Your Monitored Environment

Consider the possibility of a great amount of objects being collected, analyzed, and maintained by the application, several aspects of the underlying server must be taken into account. The sizing of the supporting clusters and containers depends on the complexity of the underlying environment. Sufficient processing power and CPU memory are required to support effective collection, server data handling, and analytics.

> **NOTE:** Currently Quest validates the environment with up to 10000 containers. If your environment beyonds this scale, contact Quest Support.

Foglight Management Server Requirements

The minimum system requirements of the Foglight Management Server vary from the scale of clusters. The scale of clusters is determined by running containers.

Table 1. Foglight Management Server requirements

Operating System	Maximum Containers	Foglight		Agent Manager	
		JVM Settings	# of CPUs	JVM Settings	# of CPUs
Windows 64-bit	1000	Xms|Xmx=4G	2	Xms|Xmx=4G	2
	5000	Xms|Xmx=8G	4	Xms|Xmx=8G	4
	10000	Xms|Xmx=12G	6	Xms|Xmx=12G	6
Linux 64-bit	1000	Xms|Xmx=4G	2	Xms|Xmx=4G	2
	5000	Xms|Xmx=8G	4	Xms|Xmx=8G	4
	10000	Xms|Xmx=12G	6	Xms|Xmx=12G	6

If you are using an embedded Agent Manager, make sure to use the sum resources of both Foglight and Agent Manager.

Kubernetes Agent Requirements

Kubernetes Agent collects inventory and metrics every 5 minutes by default. Refer to Configuring data collection interval for details about how to change the collection interval.

Table 2. Kubernetes Agent requirements

Maximum Containers	Kubernetes Agent Collection Interval (minutes)	
	Inventory	Metrics
500	5	5
1000	10	10
5000	30	30
10000	60	60

Table 2 is the recommendations for local Kubernetes clusters. If you deploy Kubernetes clusters on the Cloud Provider Kubernetes Service, consider your network rate and change your configurations based on different Cloud Provider and different region/zone of your cluster.

Docker Swarm Agent Requirements

Docker Swarm Agent collects inventory and metrics every 5 minutes by default. Refer to Configuring data collection interval for details about how to change the collection interval.

Table 3. Docker Swarm Agent requirements

Maximum Containers	Docker Swarm Agent Collection Interval (minutes)	
	Inventory	Metrics
500	5	5
1000	10	10
5000	30	30

Table 3 is the recommendations for local Docker Swarm clusters. For cloud environment, consider network rate and change configurations based on different Cloud Provider and different region/zone.

Getting Started

- Prerequisite
  - Kubernetes Agent
    - Preparing the Kubernetes credential
    - Enabling Heapster service in monitored environment
  - Docker Swarm Agent
    - Preparing Docker Swarm Agent credentials
    - Enabling Docker Remote API for monitored docker host
    - Uploading Docker Swarm Agent credentials
- Creating and Activating Agent
  - Creating and Activating a Kubernetes Agent
  - Creating and Activating a Docker Swarm Agent
- Configuring data collection interval

Prerequisite

Kubernetes Agent

Each Kubernetes Agent monitors the assets inside the selected Kubernetes Service Providers. To enable the data collection, complete the following prerequisites before create agent.

- Preparing the Kubernetes credential
- Enabling Heapster service in monitored environment

Preparing the Kubernetes credential

The Kubernetes configuration file named *KubeConfig* is a standard configuration of Kubernetes and is required for Kubernetes agents to access the cluster. Foglight for Container Management verifies and supports the local Kubernetes and the following Cloud Kubernetes Service Providers. Based upon your environment, select either of approaches to get your *KubeConfig* file:

NOTE: Data from different Kubernetes Agents with the same cluster name will be merged into one cluster.

- Local Kubernetes
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (EKS)
- Google Cloud Platform Container Engine (GKE)
- IBM Cloud Kubernetes Service
- Openshift Origin

Local Kubernetes

If you build a Kubernetes cluster locally, find this *KubeConfig* file under the `/etc/kubernetes/admin.kubeconfig` on your master node.

Azure Kubernetes Service (AKS)

Before generating the Kubernetes credentials, record the following information:

- Azure Username
- Azure Password
- Azure Subscription Number
- The name of your AKS Cluster Resource Group
- The name of your AKS cluster

Download the Azure Command Line Interface and install it in your local platform, and then follow steps below to generate your Kubernetes credential:

1. Run the command *`az login`*.

   Then a browser shows up, directing you to the Azure Portal where you should enter your Azure Username and Password to complete the authentication.

2. Run the command: *`az account set --subscription <azure subscription number>`*
3. Run the command: *`az aks get-credentials --resource-group <azure resource group name> --name <azure cluster name>`*
4. Find the Kubernetes configuration file under *`<USER_HOME>/.kube/config`* on your local platform.

   NOTE: The token in this Kubernetes configuration file will get expired after two years. If you don't want the credential gets expired, refer to Foglight Container Tools for detail.

Amazon Elastic Container Service for Kubernetes (EKS)

Follow the Amazon EKS offical guide Getting Started with Amazon EKS. Follow the guide and complete Create a kubeconfig for Amazon EKS. in the end of the guide.

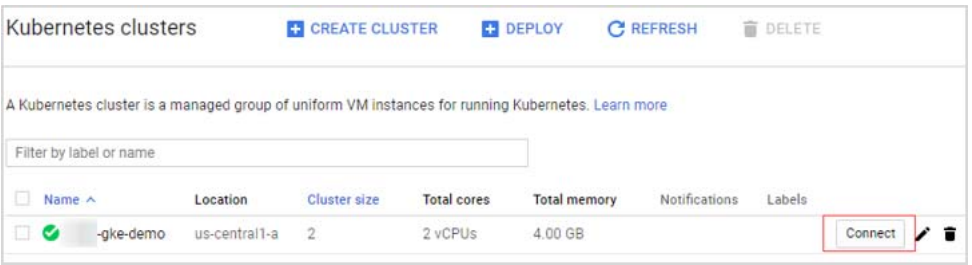
NOTE: If you don't want the credential gets expired, refer to Foglight Container Tools for detail.

Google Cloud Platform Container Engine (GKE)

Download the Google Cloud Client tool and install it in your local platform, and then follow steps below to generate your Kubernetes credential:

1. Generate the intermediate Kubernetes credential for your cluster.

   a. Log into your Kubernetes cluster, click **Connect** next to your cluster name.



   b. Click to copy the command below, and then run this command.

c Find the intermediate Kubernetes configuration file under *<USER_HOME>/.kube/config* on your local platform. The following is the example of this intermediate Kubernetes configuration file.

> **NOTE:** This Kubernetes configuration file cannot be used as the agent credential because the token in this file will get expired soon and "*cmd-path*" of the token directs to your local platform.

```
kind: Config
preferences: {}
users:
- name: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
  user:
    auth-provider:
      config:
        access-token: ya29.GlzuBVkzkoVc1VUV_7yXM50DpQ7z7ahGzFA_f2o08FhxZjDICXRFdAw5ytBc9dHBT90yYk
        cmd-args: config config-helper --format=json
        cmd-path: C:\Users\jwang7\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd
        expiry: 2018-07-04T06:44:59Z
        expiry-key: '{.credential.token_expiry}'
        token-key: '{.credential.access_token}'
      name: gcp
```

d Open Google Cloud Client tool and run the following commands to create a Kubernetes service account that grants with the *cluster-admin* role and the access to your Google Kubernetes Engine (GKE) cluster.

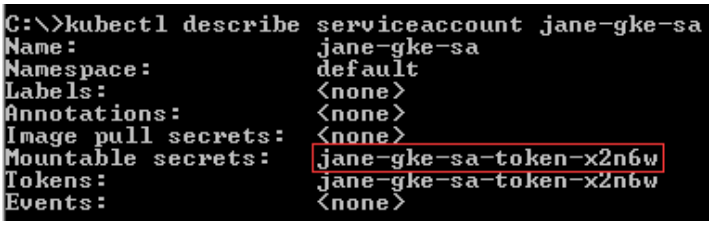
a *kubectl create serviceaccount <service account name>*

b *kubectl create clusterrolebinding <cluster role binding name> --clusterrole=cluster-admin -serviceaccount=default:<service account name>*

"default" in the above command is the namespace name of this service account name. The name space name will be "default" if you do not change it. You can also change to other namespace names, as needed.

c *kubectl describe serviceaccount <service account name>*

You will get the response similar as below. Record the <secret name> for later use.



```
C:\>kubectl describe serviceaccount jane-gke-sa
Name:                jane-gke-sa
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   jane-gke-sa-token-x2n6w
Tokens:              jane-gke-sa-token-x2n6w
Events:              <none>
```

d *kubectl describe secret <secret name>*

You will get response similar as below. Record the token value (exclude "token:") for later use.

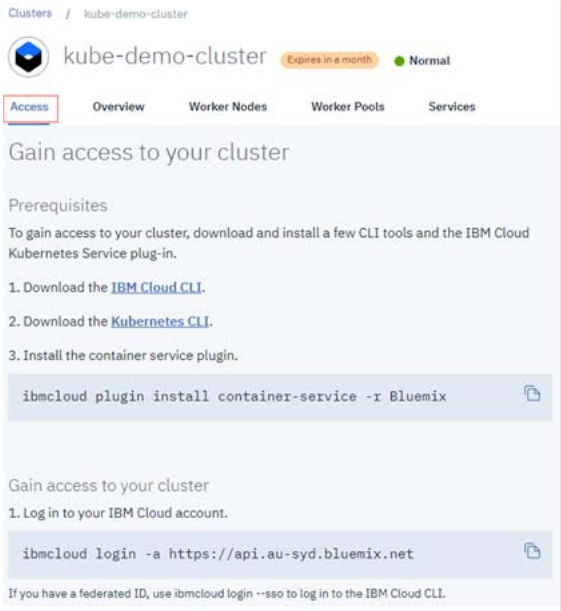
```
C:\>kubectl describe secret jane-gke-sa-token-x2n6w
Name:         jane-gke-sa-token-x2n6w
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=jane-gke-sa
              kubernetes.io/service-account.uid=9f16c355-7f4f-11e8-9c03-42010a80
0032

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1115 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZ
pY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx
0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImphbnUtZ2t1LXNhLXR
va2VuLXgybjZ3Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmF
tZSI6ImphbnUtZ2t1LXNhIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY29
1bnQudWlkIjoiOWYxNmMzNTUtN2Y0Zi0xMWU4LTljMDMtNDIwMTBhODAwMDMyIiwic3ViIjoic3lzdGV
tOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6amFuZS1na2Utc2EifQ.VdRPWHLCqU3taWV5dyjZf4PPcQmx
7XrXZDflLT2t3tVPkgoyEWRQZOK9ebPCfOm-4h3loIBSNuq8TokcZiU5K6nf6gToJ1iNAQEuNBwJ5zjp
JumCzfiIpOjMPcTm6kF2BhhrvFkKnXNCaJRSmCIhHLBqtB4rqWKCdtJU8yfJTR253c8n2-gtT3XwPLxD
9ELLUzEXFr-vjaK7wDD_jFbLEyTRSmKQGiixIWlTfea8scwops9_cJx6LRY92FnucdwMrpnxU3aIXWVR
g9BO7EU5vdFXKJreabM-VAxMksgrq3tIcwS1mU5V7ujSXcpk1j-YOP2JYwFusytjP1ptZMrS0w
```

e Open the intermediate Kubernetes configuration file under <USER_HOME>/.kube/config, and then add the user and change the token to the new one.

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURDekNDQWZPZ0F3SUJBZ01RV2(
    server: https://35.193.204.217
  name: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
contexts:
- context:
    cluster: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
    user: jane-gke-sa
  name: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
current-context: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
kind: Config
preferences: {}
users:
- name: gke_dulcet-bucksaw-208510_us-central1-a_jane-gke-demo
  user:
    auth-provider:
      config:
        access-token: ya29.GlzuBVkzkoVc1VUV_7yXM50DpQ7z7ahGzFA_f2o08FhxZjDICXRFdAw5ytBc9dHBT90yYkB4
        cmd-args: config config-helper --format=json
        cmd-path: C:\Users\jwang7\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd
        expiry: 2018-07-04T06:44:59Z
        expiry-key: '{.credential.token_expiry}'
        token-key: '{.credential.access_token}'
      name: gcp
- name: jane-gke-sa
  user:
    token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3V
```

IBM Cloud Kubernetes Service

If you have created your cluster on IBM Cloud Kubernetes Service, get the access from the console as described on the cluster's *Access* view. You will get a .pem file and a .yml file after you performing the steps.



By default IBM Cloud Kubernetes Service uses certificate authority file and token/refresh token. However, certificate authority data and service account token should be used in the Kubernetes Agent credential. After you successfully test your connection through "kubectl get nodes", follow the steps below to generate the Kubernetes Agent credential.

1 Run the command *kubectl config view -minify=true -flatten -o json*. You will get an output similar as below, then record the *<certificate authority data>* for later use.

```
C:\Users\jwang7\.kube>kubectl config view --minify=true --flatten -o json
{
    "kind": "Config",
    "apiVersion": "v1",
    "preferences": {},
    "clusters": [
        {
            "name": "kube-demo-cluster",
            "cluster": {
                "server": "https://130.198.66.34:30244",
                "certificate-authority-data": "LS0tLS1CRUdJTiBDRUJUSUZJQ0FURS0tL
S0tCk1JSUZSVENDQXkyZ0F3SUJBZ0lKQUx5SmdFOVA2NmkrTUEwR0NIcUdISWIzRFFFQkN3UUFNRGt4T
npBMUJnTlYKQkFNTUxtUXpNall4VVdZeE5HRXpPVFExTTJGaU1XUTJabU16TXpjd09UaGhZVEF6TFd0M
UltUnlibVYwWlhNdApZMkV3SGhjTk1UZ3dOekEyTURVd05UUTNXaGNOTkRVeE1USXhNRFU3TlRRM1dqQ
TUNUGN3TlFZRFZRUUREQzVrCk16STJNV0ZtTVRSaE16azBOVE5oWVpGbE5tWmpNek0zTURrNFlXRXdNe
TFyZFdKbGNtNWxkR1Z6IFdOaE1JSUMKSWpBTkJna3Foa2lHOXcvQkFRRUZBQU9DQWc4QU1JSUNDZ0tDQ
WdFQXllIdk4xL1pwc09CckJHIHlPT0wvenBpRwo5M3RtYVUyeGtYMnIvcjMzc1ZUTU5aY1R6a21xQjUMU
zV0RkFkIUFQNEVYRGJmZmpHck3M20zZkxDL2huMnhvCjNRQTQ0U254ZlRRU1ErK0R2MUk3WkQzcDIpT
zZzY2JWanErckFUL3g1NXPtamRRcmprVGoxNlUzTjZ2SzEvdUsKYipwSUdQOU5WSEMxOWFaZXNkQ3Ija
FNIdEw0K2N3NXQwNEZyQ3dRb1N1TWUnVXBYVFREMkxTd05yeXpXOVxZRupyM2PraHQ3OSs5Y2lIU1BGR
3BQb2FBZXdOSjBQY0x5eDJDemFsNGlDTHJxREZ6M3UMVM3TFhtNEFhMys2OEVxCkM3Uzk0NUhBTV96V
GpySXZUZTlvOUE3UFBDc0pKYT1PM09IQ110b25mQnlEK1BZcExkRzdTZ0N3WTFJaXc1ZzQKaU10dU04S
zRJcktPSnluK2ltR1d6QUpPci94cjk2dy94Tm14UklIUkNPc2RzTVg1N3hPS3pqNlJvb2E3bGpPZAoyL
3Znb1Z0QkpxTlVPWU9QUEhybitjS09ZZiQ5cmh2cVxXemtQT1JzOFplRDlqMmk4WURNeEd2L2ZybThRT
npUCkduYkxuckdUZUZCR0c3cGitbXRZSVBwOCtGYmFxbmtGUTZ5d2NXOTFqSnZ6STlvZEp6ZnQ4Zk9IZ
kRNcjBteFgKR05aNlpHSU1FU1ZZU3JNTUt2RCtMT01lSzBqRH1hRWpaYm80dU14R3YrU3NLMWZrc0JUW
UJubWdBaDIvRC9VcApwNUU3SXI1NkcvL2Y3UGFHc3NDMTZBK2J0Z1ZONk9oOUUsTXpHQUhTVWR6ZndRR
3dxcjB4NHRMUlU2c3A2U1dFC1BZSEUmR0JMbEY4Mm5XcEMzePUDQXdFQUFhT1FNRTR3SFFZRFZSME9CQ
l1FRkcxCUEyazQxZ2lTcndIdXltRkMKdzdsbWtlcE5NQjhHQTFUZE13UU1NQmFBRkcxCUEyazQxZ2lTc
ndIdXltRkN3N2xta2UwTk1Bd0dBMUVkRXdRRgpNQU1CQWY4d0RRWUpLbIpJaHZjTkFRRUxCUUFEZ2dJQ
kFGc29JbWFydjJRck4TTNKajQ5andUb2NOd2lUU1JTClNFd1RwanRobUtGdEdOdU9EU1UMbUMwSjRWa
3g5Um9nRkUxZ0YzbG3ZPbINFUPI4T3l6UGsxI24eZy9kbWUsQUwKajBHUUFoUEhCdGlHUUdtSy91ckITa
2IZOUJjQ3RuYU1oeUpuZTdnOXhlU3p2MzRQeTExOUtnazE5YUZSYkRrQuo1UmpuQ01icUpCTWdZOHZtU
VlieTMwMDA1Ni9pNjlZR1RXdTlFU1UZOS9VTGZnajNqOXA2Q21zMzIrelIwTVQ5CjZPcjdlbU0yL0YyM
0U1dnRSRGNtbjZQN1dnaDE4dVRDd2tnVno0T0ppbzJ0ODBMOUdESnRILytuNzFOUUUHNGcKMUU5TTEzT
FlIOW1KdTUwSllHb3p0S08zc0UCNmg4cTUZY2JoMXhhZmEvR1Byc1N2SnNBNEgwVFd2Rk8yUnhNcQp5U
0ZIMjBRbzlaK3hHQ2ULUk1OYjFkY2x6aktxOFQwK3JXK25uTi9FU0ZIZlBIWmdiWW9UR3djYit6SjZha
HR1CnJWTVRzaUZSK0pCQUdnUjlXTE0wcFZnTG1YSDdrZkM4QjFaQ1NOcEZ2d0ZOeXNyQTk2RDlId2tmS
GpvZlJ2eUAKaWxSRFZ4eHB6OGowVTRkc2hDancwZmFyeWiEL3UMTVNZRjFTU2NZTlBaTUwvUGN2cUNFe
Fd1Z293REZqcjJIMQp2YnhpL3lxcnplZUJNMjgrZ1UWRERIazJXWczUlZqQnU3b3Y4QUNUbFhxY2NQU
0ZoNU10R1djeUdVdjhFZUN4CkR0dEVHTk1hR1ZMSnY1eUhUb25TM0J3Z25RUTh2RWYzNGw3a29TZk1LR
lkxZWM0SDhvaWZaT3pyMEt0UEIxakMKOUBoNFZTS1Ixa2ZrCi0tLS0tRU5EIENFUlRJRklDQURFLS0tL
S0K"
```

2 Run the command *kubectl create serviceaccount <service account>*.

3 Run the command *kubectl describe serviceaccount <service account>*. You will get a response similar as below, then record <service account secret> (in this sample, it is jane-sa-token-xkqrk) for later use.

```
C:\Users\jwang7\.kube>kubectl describe serviceaccount jane-sa
Name:                jane-sa
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   jane-sa-token-xkqrk
Tokens:              jane-sa-token-xkqrk
Events:              <none>
```

4 Run the command *kubectl describe secret <service account secret>*. You will get a response similar as below, then record <service account token> for later use.

```
C:\Users\jwang7\.kube>kubectl describe secret jane-sa-token-xkqrk
Name:         jane-sa-token-xkqrk
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=jane-sa
              kubernetes.io/service-account.uid=f9a842fb-80e0-11e8-9481-82908083
3fed

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1887 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGUzL3NlcnZ
yV2UhY2NudU50IiwiaJUiZXJuZXRlcy5pbv9zZXJ2aWNlYWNjb3UudC9zYVt1c3BhY2UiOiJkZWZhdUx
0Iiwia3UiZXJuZXRlcy5pbv9zZXJ2aWNlYWNjb3UudC9zZWNyZXQubnFtZSI6ImphbnUtc2EtdG9rZW4
teGtxcnsiLCJrdWJlcm5ldGUzLmlvL3NlcnZpY2UhY2NvdU50L3NlcnZpY2UtYWNjb3UudC5uYW1lIjo
iamFuZS1zYSIsImt1YnVybnV0ZXMuaW8vc2UydmljZWFjY291bnQvc2Vydmljes1hY2NvdU50LnVpZCI
6ImY5YTg0MmZiLTgwZTAtMTF1OC05NDgxLTgyOTA4MDgzM2Z1ZCIsInNiYiI6InN5c3RlbTpzZXJ2aWN
lYWNjb3UudDpkZWZhdWx0OmphbnUtc2EifQ.Q08FFAaDTQBdM1Ic_FEE6jY3LBzeuZsdstFHCyzd5uM6
cRR5exC_uCnsI17jSBuvnFUvpYuAA3zp_DLnBltxe0773FqR7aGhqGgtKkgyuS9yM3QHu_xAEhwd9Etc
Mbc2RxfA30UjxZcvHUyQd48OcyyYzQkGM8Iat5nZS9hSU5b-hiveqzL-_CBZBW8xuMDqFIfUWmIlaYEf
htyfQnLumnMYNVUpQ8KgpP0kyza6YknSvKOot6WGuU-YvYFHjv1OgLQxuhhGrw01AYy8EuyF7kzuhnxU
UUrxONBao2P1004DxuHQjay07Z50NgmU8hq0URUQMDTkGRnwslnQK9m0xioUT4TcvDKS15brCR84cYH
-ZR3PHp4biFuxqqKuc6YSFZcCZ4LWU4A2zMAbLuFgZaNozvLIwpdoGBcnldhisuds9Hughlq_X9Napkg
lujU-8PJqSExMACuAQPtsP0JWO5CewA3Dz7Z4cKaX7Z-qLtHqfqK6wYvyKiOFLin3pAwroM59-5lCurC
```

5 Open the .yml file generated previously, which looks like below.

```
apiVersion: v1
clusters:
- name: kube-demo-cluster
  cluster:
    certificate-authority: ca-mel01-kube-demo-cluster.pem
    server: https://130.198.66.34:30244
contexts:
- name: kube-demo-cluster
  context:
    cluster: kube-demo-cluster
    user: Jane.Wang@quest.com
    namespace: default
current-context: kube-demo-cluster
kind: Config
users:
- name: Jane.Wang@quest.com
  user:
    auth-provider:
      name: oidc
      config:
        client-id: bx
        client-secret: bx
        id-token: eyJraWQiOiIyMDE3MTAzMC0wMDowMDowMCIsImFsZyI6IlJTMjU2I
        idp-issuer-url: https://iam.ng.bluemix.net/kubernetes
        refresh-token: J1B9J8mzGdOecQbYvbmh7csxFZ5jiC9QEJxgWP7Ei57db_w
```

6 Change the certificate authority to the data <certificate authority data> of this authority and change the users section to use <service account token>. Save your changes, and then you will get a credential file like below. This file will be used as the Kubernetes Agent credential to connect to your IBM cloud Kubernetes service cluster.

```
apiVersion: v1
clusters:
- name: kube-demo-cluster
  cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0
    server: https://130.198.66.34:30244
contexts:
- name: kube-demo-cluster
  context:
    cluster: kube-demo-cluster
    user: ibmclouduser
    namespace: default
current-context: kube-demo-cluster
kind: Config
users:
- name: ibmclouduser
  user:
    token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm
```

Openshift Origin

If you could access the `/etc/origin/master/admin.kubeconfig` on the master node, download this file which can be used as the Kubernetes Agent credential.

If you could not access the `/etc/origin/master/admin.kubeconfig` on the master node, follow instructions below to generate a permanent credential file.

Before generating the permanent Kubernetes credentials, record the following information and ensure you have granted the privilege for accessing the cluster-wide resources:

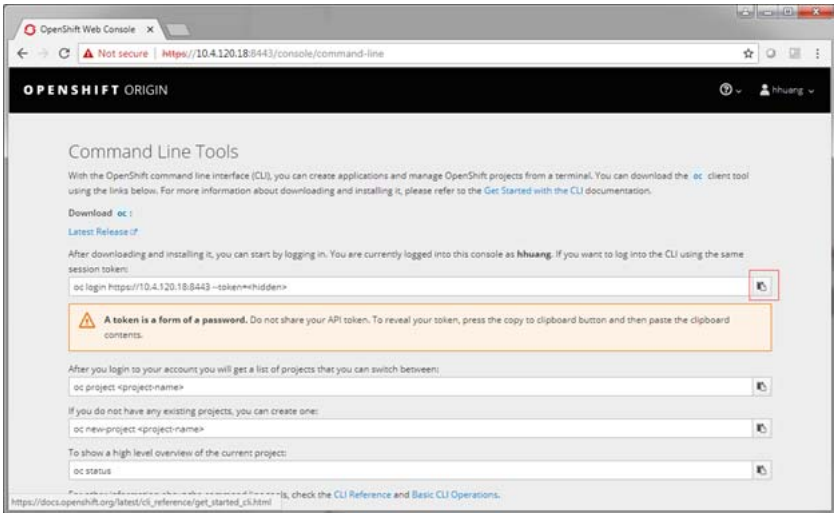
- Openshift Username
- Openshift Password

Download the Openshift Command Line Interface and install it in your local platform, and then follow steps below to generate your Kubernetes credential:

1 Log into Openshift and generate an intermediate Kubernetes configuration file.

   1 After logging into Openshift, click **Command Line Tools** on the upper right.

   2 Click the button next to the *Session token* field, copy the command, and then paste it in your local Command Line Tool. Make sure to find the intermediate Kubernetes configuration file under `<USER_HOME>/.kube/config` on your local platform.



   3 On your local platform, browse to open this configuration file. You may see the context similar to the following. Record ***<config-cluster-name>*** for later use.

```
1  apiVersion: v1
2  clusters:
3  - cluster:
4      insecure-skip-tls-verify: true
5      server: <cluster-ip-port>
6    name: <config-cluster-name>
7  contexts:
8  - context:
9      cluster: <config-cluster-name>
10     namespace: <project-name>
11     user: <config-user-name>
12   name: <config-context-name>
13 current-context: <config-context-name>
14 kind: Config
15 preferences: {}
16 users:
17 - name: <config-user-name>
18   user:
19     token: <access-token>
```

2 The token generated in step 1 will be expired after 4 hours, however Foglight for Container Management needs a permanent Kubernetes credential. So you need to create a service account with "**cluster-admin**" role, and then get the authorization code (not expired) of this service account to generate our permanent Kubernetes credential.

   1 Run the command `oc project` ***<project-name>***.

   2 Run the command `oc create serviceaccount` ***<service-account-name>***.

   You can check if your service account has been created successfully using the command: `kubectl get serviceaccounts`

   3 Run the command `oc serviceaccounts get-token` ***<service-account-name>***. Then you will get a token *<service-account-token>* like below. Record this token for later use.

"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Im9zLWFkb

*WluLXRva2VuLWY0a2ZsIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLW
FjY291bnQubmFtZSI6Im9zLWFkbWluIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9
zZXJ2aWNlLWFjY291bnQudWlkIjoiODMzNGU0NTQtNzQ1Yy0xMWU4LWFmNmEtMDA1MDU2YjY3
NDFhIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6b3MtYWRtaW4ifQ.RW
H_AoXy2U1elkHN_Bs9IR1xo0zNCJlwcY0h3zuQnrkOFi8gVpX1I77uhAPp7oIjPqDSWkUAN9F
6mP_tNdGwJsqRmHYEMOtCLnnIM61BYxIcABvwr66aOZ3Gn0D7EM5M_7XgKDCl6ON3W5NaH0D8
DpVTYqxkQ49u3qt4gqrcjVCaSsDNWlgGxY4KOIDrUbKkdgaRKzeD9o4Bv9VbYICqyxwoUebku
JAcHiXGIcSe-ozS_zroPi1tT5HW-RY0Pn3Fp3zBnydiokna0-mXot5lqoYc-
R6E1U9YSrAOhWm9Q8ipiut6OczXbmLPM4DYve6dmHi_j5FquCqhod-QlA7aPw"*

4 Run the following command to grant your service account with the "**cluster-admin**" privilege: *kubectl create clusterrolebinding* ***<cluster-role-binding-name>*** *--clusterrole=cluster-admin --serviceaccount=default:****<service-account-name>***.

3 Generate a permanent Kubernetes configuration file and save it under *<USER_HOME>/.kube/config file/credential*.

1 Open and edit the intermediate configuration file.

2 Use kubectl to add user credentials, create new context, in the end change the existing contexts to the ones that you added in step 2. For example,

```
kubectl config set-credentials <credential-name> --token=<service-account-token>
```

```
kubectl config set-context <new-context-name> --cluster=<config-cluster-name> --user=<credential-name> --namespace=<project-name>
```

```
kubectl config use-context <new-context-name>
```

3 Save the current Kubernetes configuration file.

Enabling Heapster service in monitored environment

There are various approaches to enable Heapster on your Kubernetes cluster. Visit Heapster official website to determine the approach that you are going to deploy your Heapster service, or you can follow instructions in https://github.com/foglight/container to deploy your service.

Some of the cloud platform Kubernetes service has enabled Heapster service for the cluster. If you have connected to the cluster, run the following command to check: *kubectl cluster-info*

Docker Swarm Agent

Each Docker Swarm Agent monitors the assets in one docker host. Docker Remote API needs to be enabled for the Docker Swarm Agent collecting data from the docker host. If TLS is enabled to secure the Docker Remote API, credential for Docker Swarm Agent needs to be prepared. Complete the following prerequisites before create agent.

- Preparing Docker Swarm Agent credentials
- Enabling Docker Remote API for monitored docker host
- Uploading Docker Swarm Agent credentials

Preparing Docker Swarm Agent credentials

If TLS enabled to secure Docker Remote API, then complete the following guide to get the credentials for Docker Swarm Agent for the docker host. Otherwise, continue with Enabling Docker Remote API for monitored docker host on page 17

Refer to the official guide to generate the keys. Be aware that, during generating the keys, the Foglight Agent Manager host address should be in the allow access list.

Docker Swarm Agent needs following credentials, you can get them when you finish the official guide.

- CA Public Key (ca.pem in official guide)
- Client Public Key (cert.pem in official guide)
- Client Private Key (key.pem in official guide)

Enabling Docker Remote API for monitored docker host

Change *ExecStart* in docker service startup script as below.

Non-TLS secured

```
ExecStart=/usr/bin/dockerd –H tcp://0.0.0.0:2375 –H
unix:///var/run/docker.sock
```

NOTE: Access should be allowed to the TCP port 2375

TLS secured

If TLS enabled, complete Preparing Docker Swarm Agent credentials on page 16 first, then you will get the ca.pem, server-cert.pem and server-key.pem mentioned in the official guide.

```
ExecStart=/usr/bin/dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-
cert.pem --tlskey=server-key.pem -H tcp://0.0.0.0:2375 -H
unix:///var/run/docker.sock
```

NOTE: Access should be allowed to the TCP port 2375

Then restart docker service.

Uploading Docker Swarm Agent credentials

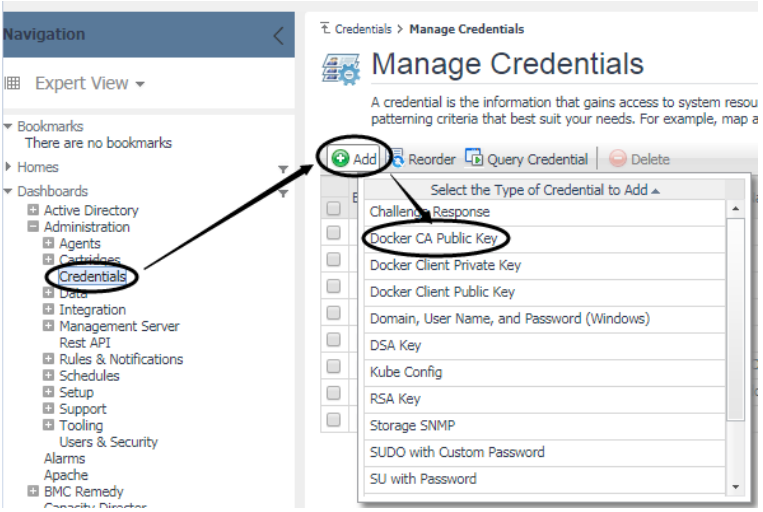
If TLS is enabled to secure Docker Remote API, go through this section to upload the credential for Docker Swarm Agent. Otherwise, skip this section.

When complete Preparing Docker Swarm Agent credentials on page 16, following credentials should be generated.

- CA Public Key
- Client Public Key
- Client Private Key

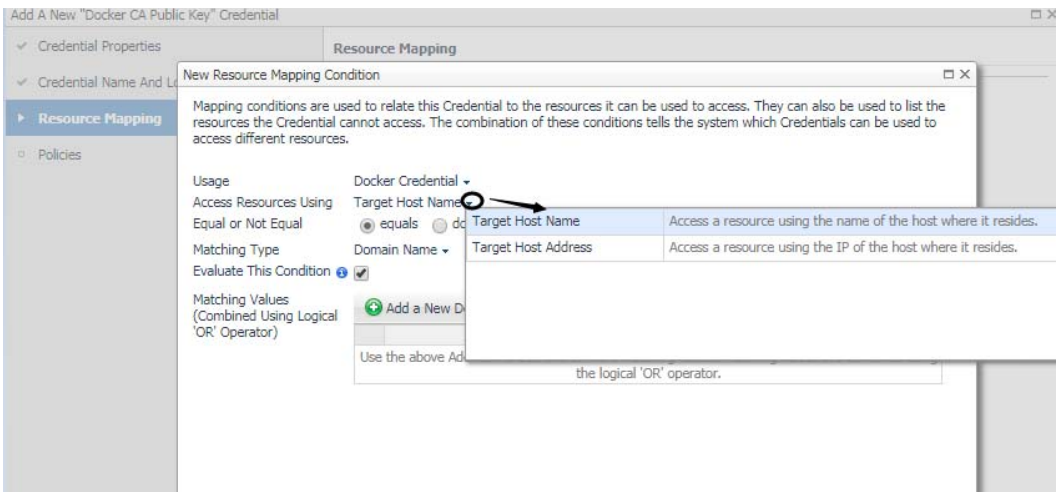
These are the credentials for Docker Swarm Agent, complete the following steps to upload the credentials.

On the **Administration > Credentials > Manage Credentials** dashboard, click **Add**, and then select Docker CA Public Key or Docker Client Public Key or Docker Client Private Key to upload related credentials. Take Docker CA Public Key as an example.

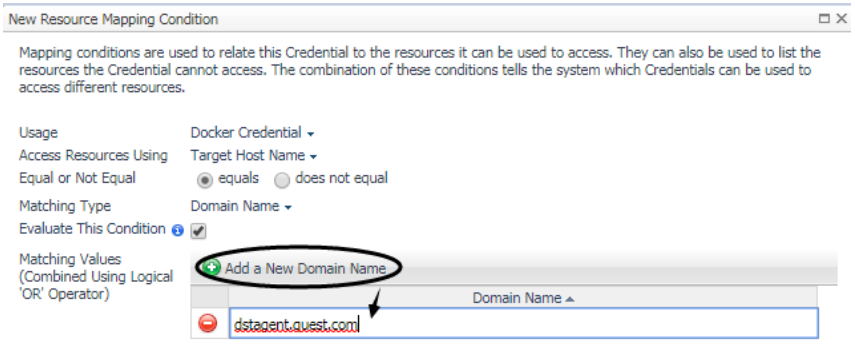


In the **Add a New "Docker CA Public Key" Credential** dialog box, specify the following values:

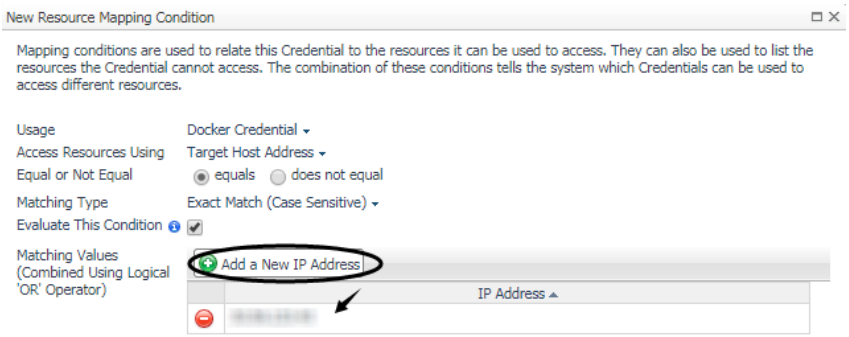
- Credential Properties: Click **Load from file** to import Docker CA Public Key, and then click **Next**.
- Credential Name And Lockbox: Specify a unique name for this credential, and then click **Next**.
- Resource Mapping: Click **Add**. In the **New Resource Mapping Condition** dialog box, choose Target Host Name or Target Host Address for the monitored docker host.



If choose **Target Host Name**, then enter the host name of the monitored docker host.



If choose Target Host Address, then enter the IP address of the monitored docker host.



Then click **Add** to finish editing **New Resource Mapping Condition** and back to **Resource Mapping**. Then click **Finish**.

Then **Docker CA Public Key** has been uploaded and mapped to the docker host. To monitor this docker host, **Docker Client Public Key** and **Docker Client Private Key** also need to be uploaded following the above steps.

Creating and Activating Agent

Foglight for Container Management supports Kubernetes Agent and Docker Swarm Agent.

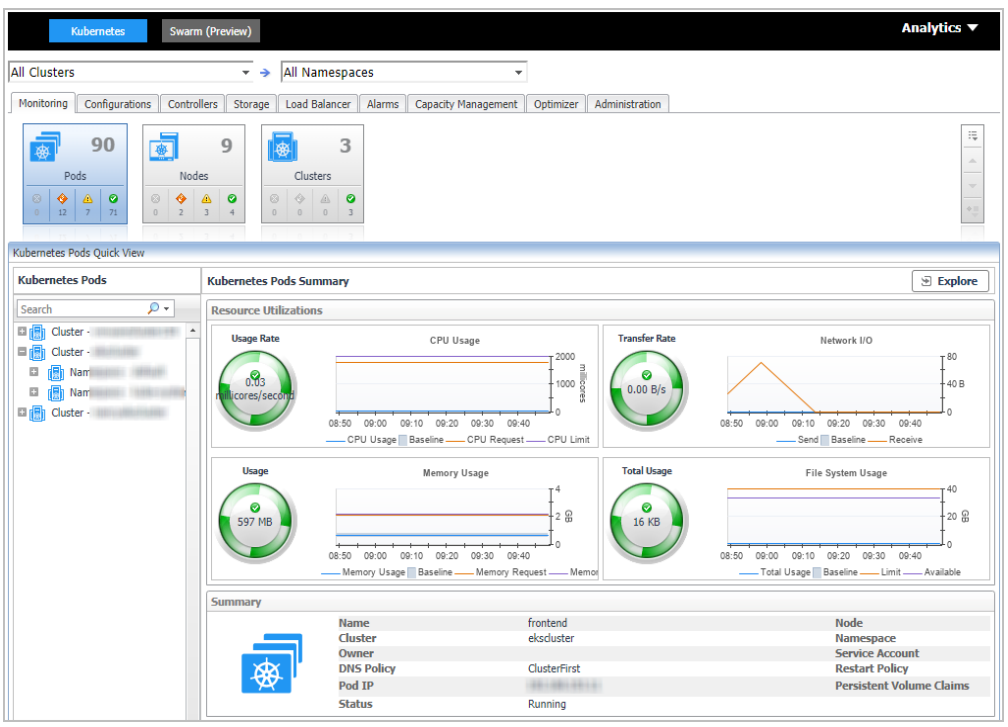
- Creating and Activating a Kubernetes Agent
- Creating and Activating a Docker Swarm Agent

Creating and Activating a Kubernetes Agent

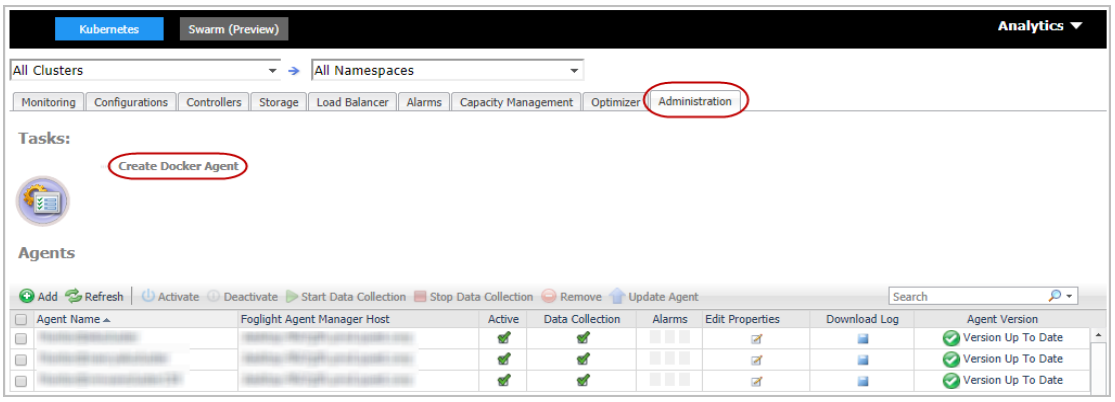
To create a Kubernetes agent on a monitored host:

1 Log in to the Foglight browser interface and make sure the left Navigation panel is open.

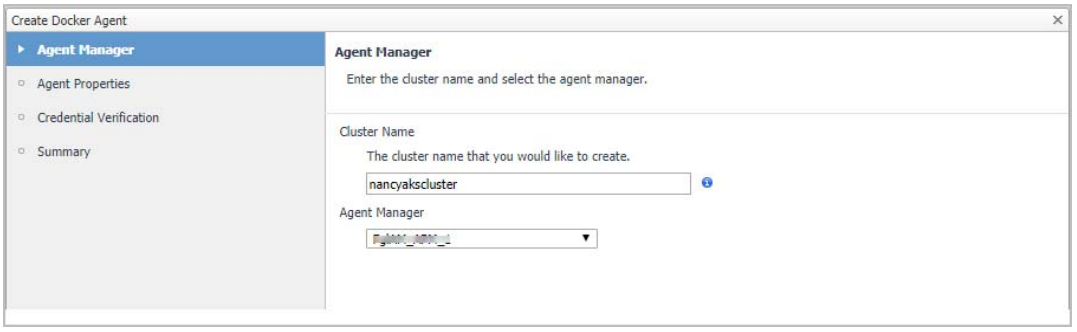
2 On the navigation panel, from **Standard View** click **Container Environment** or from **Expert View** click **Dashboards > Container**. Then the Container dashboard will display as below.



3 In the Container dashboard, click **Administration** tab, and then click **Create Docker Agent**. The **Create Docker Agent** wizard opens.

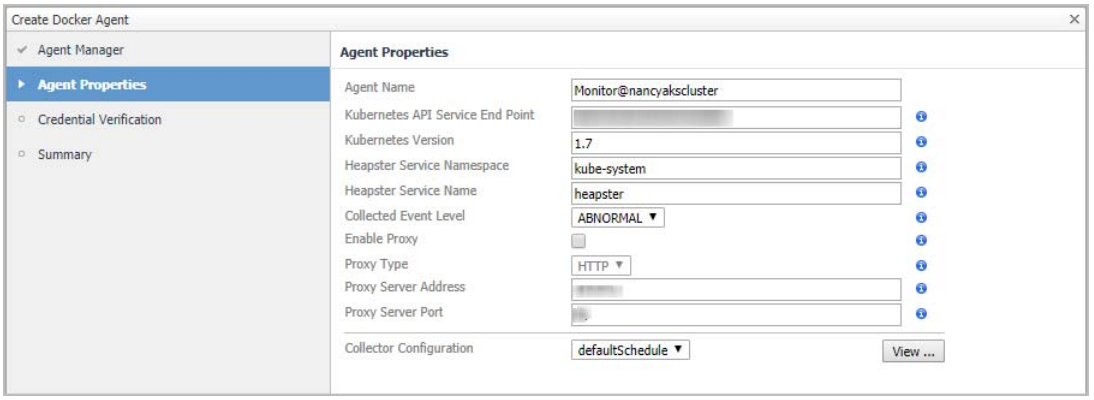


4 *Agent Manager*: specify the following values, and then click **Next**.



- *Cluster Name*: unique name for the monitored cluster.
- *Agent Manager*: select an Agent Manager which manages the agent.

5 *Agent Properties*



- *Kubernetes API Service End Point*: Get this information from the *KubeConfig* file. For more information, see Enabling Heapster service in monitored environment on page 16.
- *Kubernetes Version: 1.7 by default.*

  NOTE: Only need to change for OpenShift clusters.

- *Heapster Service Namespace/Heapster Service Name*: Get both values from the Heapster service configuration. For more information, see Enabling Heapster service in monitored environment on page 16.

NOTE: If you monitor OpenShift Origin and enable the Hawkular metrics, then the Heapster Service Namespace should be openshift-infra, and the Heapster Service Name should be https:heapster:.

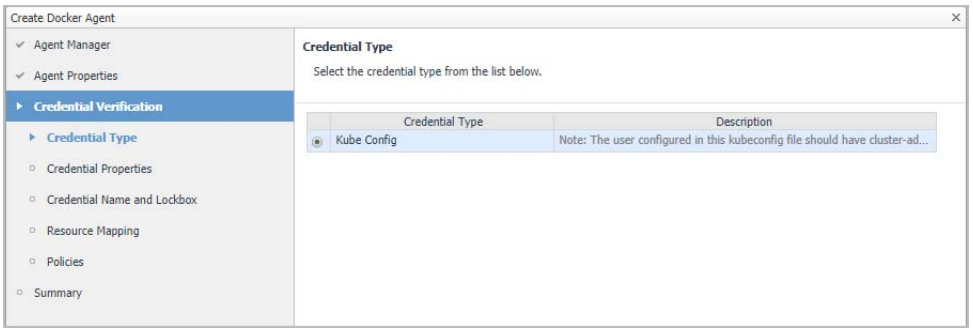
- *Collected Event Level:* Set the collected event level, including *ABNORMAL* and *ALL*. *ALL* will collect both abnormal and normal events while *Abnormal* only collects abnormal events.
- *Enable Proxy*: To enable the proxy, select the checkbox. Enter the *Proxy Server Address* and *Proxy Server Port* information.

6 *Credential Verification*

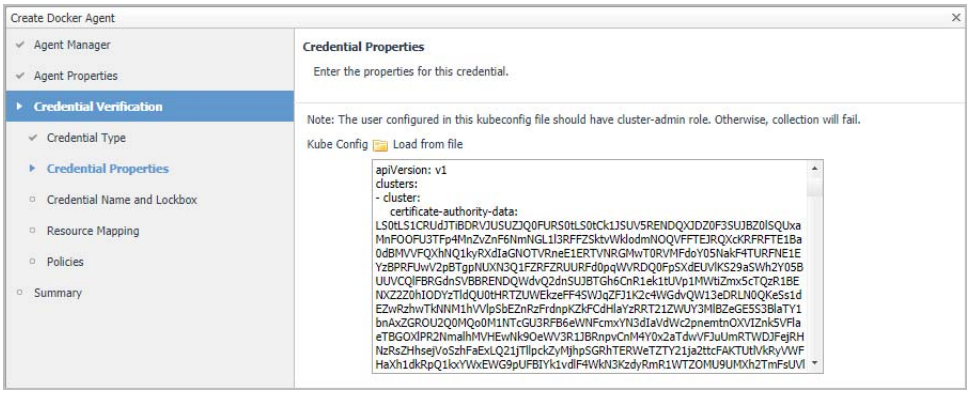
- Do not configure a credential: click **Next**.

NOTE: If an existing credential is detected, the screen will prompt, "*An existing credential has been detected for this resource and the lockbox won't be released to the selected Agent manager. To continue the Agent creation steps, manually release the lockbox. Ignore this message if the lockbox has been released and click 'Next' to continue.*".

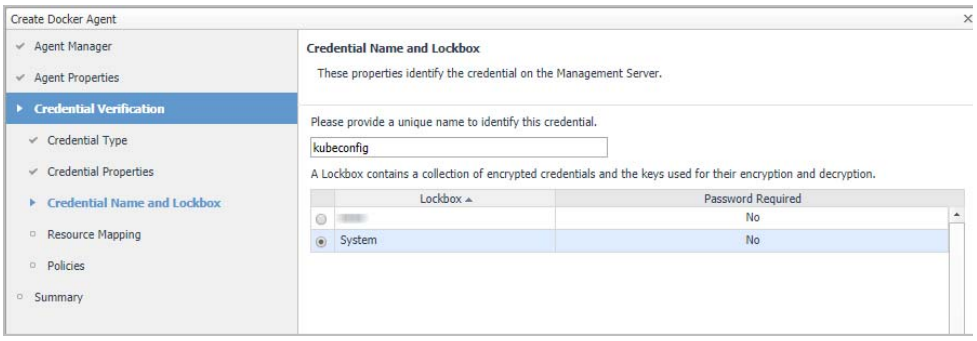
- *Add cluster to a new credential.*
  - Credential Type: Select the credential type from the list and click **Next**.



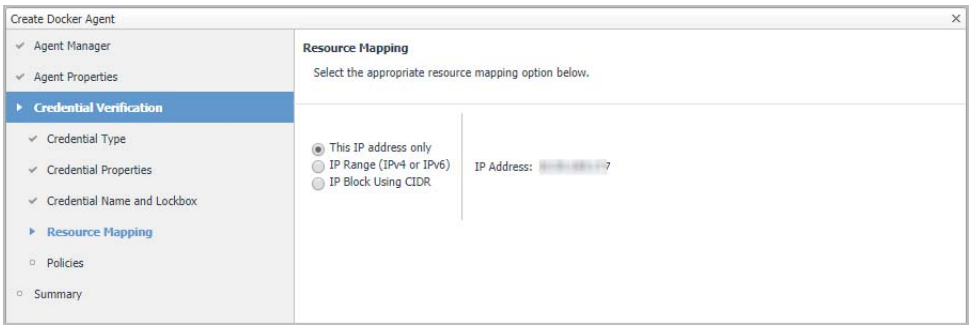
  - Credential Properties: Click **Load from file** to upload the credential and click **Next**.



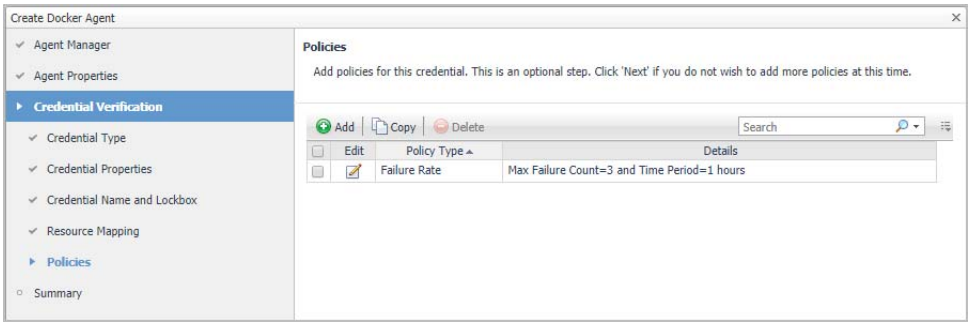
  - Credential Name and Lockbox: give a name for the credential, choose a lockbox, then click **Next**.



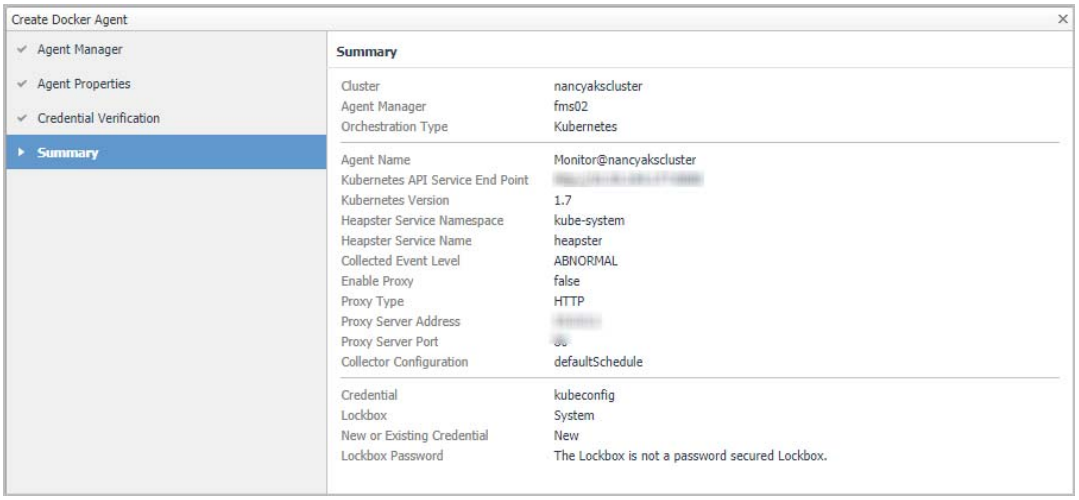
- Resource Mapping: click **Next**.



- Policies: Add policies for this credential. This is an optional step. Click **Next** if you do not wish to add more policies at this time.



7 *Summary*: click **Finish**.



8 Then, the agent will be created and activated automatically.

Creating and Activating a Docker Swarm Agent

Each Docker Swarm Agent monitored one docker host. If the docker host belongs to a Docker Swarm cluster, it will be considered as a manager/worker node. Otherwise, it will be considered to be a standalone docker host.

> **NOTE:** For a Docker Swarm cluster, you should create one Docker Swarm Agent for one host in the cluster, and if you want to monitor the whole cluster environment, you need to create all the Docker Swarm Agents for all the hosts in the cluster.

To create a Docker Swarm agent on a monitored host:

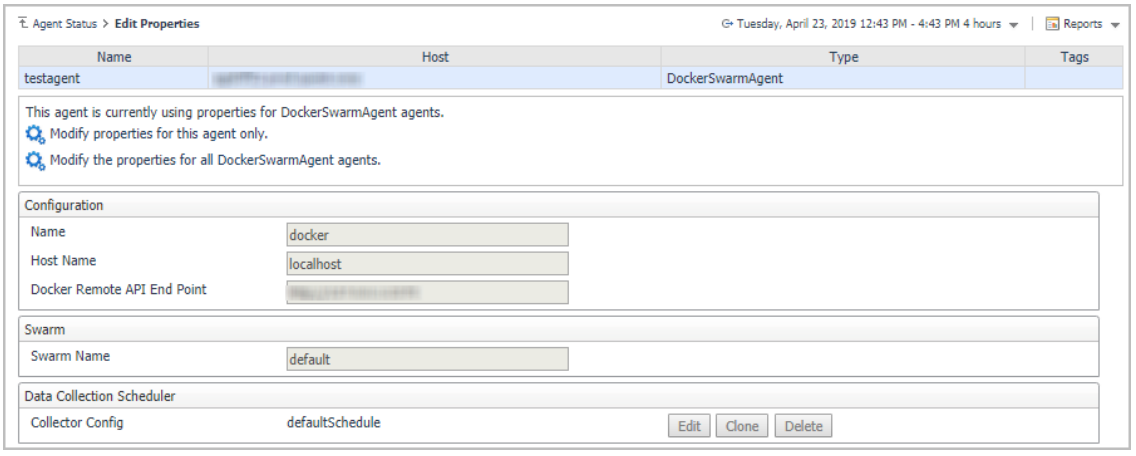
1. Login in to the Foglight browser interface and make sure the left navigation panel is open.
2. On the navigation panel, under **Dashboards**, click **Administration > Agents > Agent Status**.

   The **Agent Status** dashboard opens.
3. In the **Agent Status** dashboard, click **Create Agent**.

   The **Create Agent** wizard opens.
4. *Host Selector*: Select the monitored host that you want to monitor with the Docker Swarm agent instance that you are about to create, and then click **Next**.

   > **NOTE:** In order to select the host, the Foglight Agent Manager must be installed and running on the monitored host.
5. *Agent Type and Instance Name*: Specify the following values, and then click **Next**.
   - *Agent Type*: Select DockerSwarmAgent from the agent type list.
   - *Agent Name*: Specify the name of the agent instance that you are about to create using either of the following approaches:
     - *Generic Name*: This option is selected by default. A generic name is a combination of the host name and the agent type and uses the following syntax: `agent_type@host_name`.
     - *Specify Name*: Type that name in the *Name* field. For example, `MyAgent`.
6. On the **Summary** page, review the choices you have made, and then click **Finish**.

   The *Agents* table refreshes automatically, showing the new Docker Swarm Agent.
7. On the *Agents* table, select the Docker Swarm Agent that you create, click **Edit Properties**, and then click **Modify the private properties for this agent**.
8. In the *Agents* properties view, check if the following values have been configured based upon your environment:



   - *Name*: give a name to the monitored docker host, it should be unique.
   - *Host Name*: IP address or host name of the monitored docker host.

- *Docker Remote API End Point*: Docker Remote API endpoint of the monitored docker host. For more information, see Enabling Docker Remote API for monitored docker host on page 17.
- *Swarm Name*: specify the swarm cluster name for display. If the swarm name is kept as “default”, then the cluster name will be displayed as “default (cluster ID)” on the dashboard. If a customized name is input here, then the customized cluster name will be displayed on the dashboard.

NOTE: Ensure that the docker host inside the same cluster has the same configuration for Swarm Name.

9 Return back to the *Agents* table, select the above property changed Docker Swarm Agent, and then click **Activate**.

The new Docker Swarm Agent is created and data will be shown on the **Monitoring** tab after a few minutes.

Configuring data collection interval

The default data collection interval of agents is set to 5 minutes by default. Foglight for Container Management enables you to change this collection interval as needed.

NOTE: Changing the data collection interval will take effect for all Kubernetes agents and Docker Swarm agents.

To configure the data collection interval:

1 On the navigation panel, under **Dashboards**, select **Administration > Agents > Agent Status**.

2 On the *Agent Status* dashboard, select the Kubernetes agent that you use to monitoring the container environment, and then click **Edit Properties**.

3 In the *Edit Properties* dashboard, click **Edit** next to the *Collector Config* field.

4 In the KubernetesAgent or DockerSwarmAgent Collector Config dialog box, change the following values, as needed:

- *Inventory Collector*: Specifies the interval for collecting components.
- *Metrics Collector*: Specifies the interval for collecting metrics.

5 Click **Save**.

2

Using Foglight for Container Management

- Kubernetes
  - Monitoring Kubernetes Pods
  - Monitoring Kubernetes Nodes
  - Monitoring Kubernetes Clusters
  - Monitoring Kubernetes Other Components
  - Alarms
  - Capacity Management
  - Optimizer
  - Administration
- Docker Swarm
  - Monitoring Docker Containers
  - Monitoring Docker Hosts
  - Monitoring Docker Swarm Clusters
  - Monitoring Docker Swarm Services
  - Alarms
- Analytics
  - Kubernetes analytics
    - Heatmap analytics
    - Scatter Plot analytics
  - Docker Swarm analytics
    - Heatmap analytics
    - Scatter Plot analytics
- Domains and Object Groups
  - Domains
  - Object Groups
- Metrics
  - Kubernetes metrics
  - Docker Swarm metrics
- Rules
  - Kubernetes
  - Docker Swarm

- Customization

Kubernetes

Monitoring Kubernetes Pods

A pod contains one or multiple containers, such as Docker containers, which contains storage/network and the specification about how to run the containers. The *Kubernetes Pods Quick View*, which appears after clicking **Monitoring > Pods**, shows the data collected about the selected clusters and namespaces. This view consists of the following two panes:

- The **Kubernetes Pods** tree view, which appears on the left of *Kubernetes Pods Quick View*, lists the pods existing in the monitored Kubernetes environment.
- The Kubernetes Pods Summary view, which appears on the right after you select an individual pod in the **Kubernetes Pods** tree view.

Kubernetes Pods Summary view

The **Kubernetes Pods Summary** view appears on the right when you select a cluster in the **Kubernetes Pods** tree view.

Figure 2. Kubernetes Pods Summary view

The **Kubernetes Pods Summary** view displays the following data:

- *Resource Utilizations*: The resource utilization for the selected Kubernetes Pod over a selected period of time, which includes the following:
  - *CPU Usage*: Shows the CPU utilization summary for the selected Kubernetes Pod based on its total capacity during a selected time period.
  - *Transfer Rate*: Shows the network utilization summary for the selected Kubernetes Pod, including the average rate of network throughput, and the amounts of data sent to and received from the network.
  - *Memory Usage*: Shows the physical memory utilization summary for the selected Kubernetes Pod, broken into the amounts of memory that is swapped to disk, actively used, and allocated, all during a selected time period.

- File System Usage: Shows the file system resource utilization summary for the selected Kubernetes Pod, including the available/total/limited file system resource.

- *Summary*: Displays the detailed information about the selected Kubernetes Pod, including *Name, Node, Cluster, Namespace, Owner, Pod IP, Service Account, DNS Policy, Restart Policy,* and *Status*.

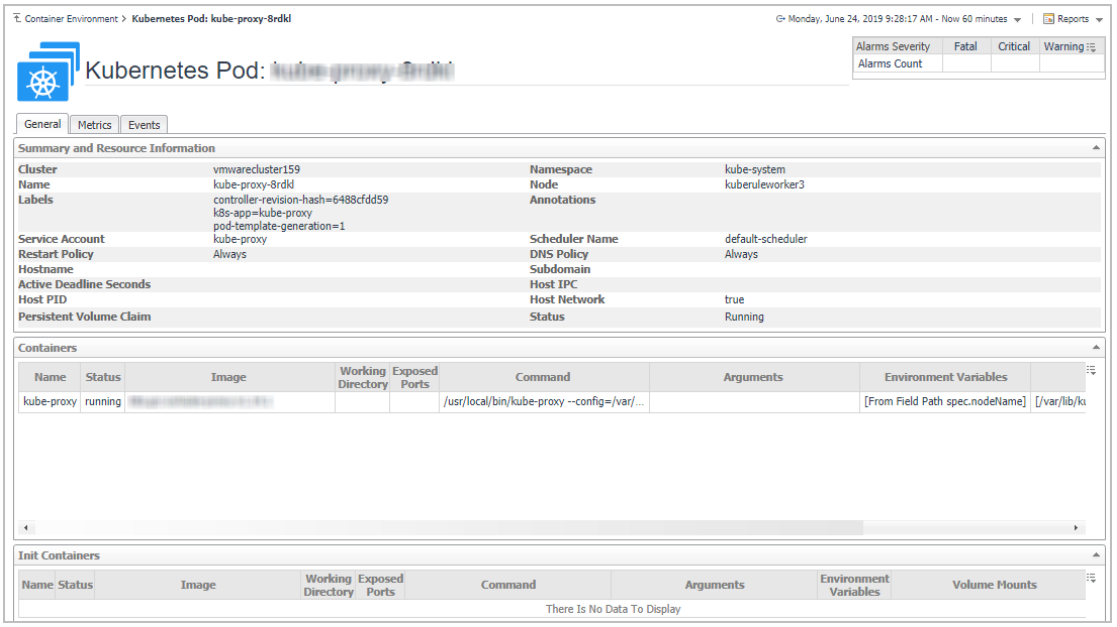
Click **Explore** on the upper right of the **Kubernetes Pods Summary** view to open the Pods Explorer view, which shows more detailed information about this Kubernetes cluster.

Pods Explorer view

The *Pods Explorer* view opens when you click **Explore** in the Kubernetes Pods Summary view, which includes the following tabs:

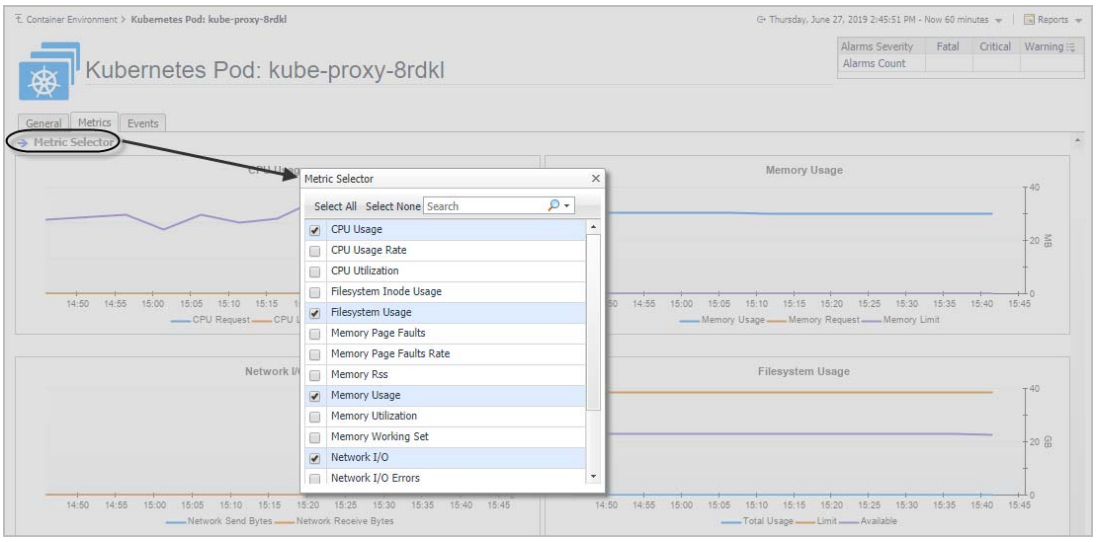
- *General tab:* The *General* tab displays the overall information of the selected Kubernetes Pod over a selected period of time, including the *Summary and Resource Information* table, the *Containers* table, and the *Init Containers* table. For more information, see Pod metrics *on page 60*.

Figure 3. Kubernetes Pods Explorer view General Tab



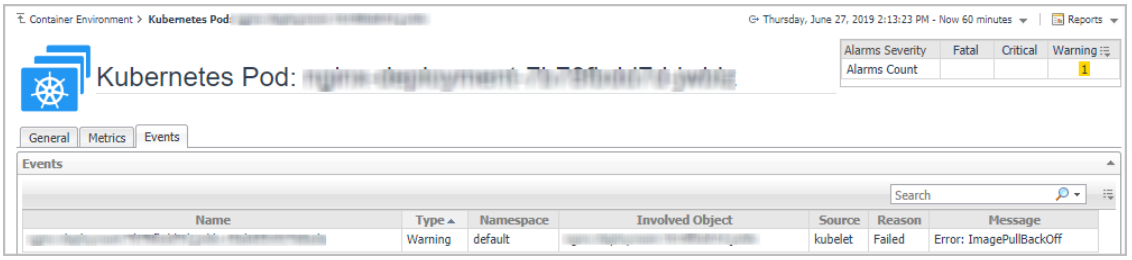
- *Metrics tab:* The *Metrics* tab displays a *Metric Selector* allowing you to choose the metrics to be plotted on this dashboard. Charts of *CPU Usage, Memory Usage,* and *Network I/O* are presented by default.

Figure 4. Kubernetes Pods Explorer view Metrics Tab



- *Event* tab: The *Event* tab lists all the events occur on the pods.
  - Name: name of the event.
  - Type: type of the event, Warning or Normal.
  - Namespace: namespace of where this event happens.
  - Kind: type of the Kubernetes component on which this event occurs.
  - Involved Object: name of the Kubernetes component on which this event occurs.
  - Source: where this event has been triggered from.
  - Reason: reason of this event.
  - Message: detailed message of this event.

Figure 5. Kubernetes Pods Explorer view Events tab



Monitoring Kubernetes Nodes

A node, previously known as a minion, is a worker machine in Kubernetes. A node may be a VM or physical machine, depending on the cluster. Each node has the services necessary to run pods and is managed by the master components. The *Kubernetes Nodes Quick View*, which appears after clicking **Monitoring > Nodes**, shows the data collected about the selected clusters and namespaces. This view consists of the following two panes:

- The **Kubernetes Nodes** tree view, which appears on the left of *Kubernetes Nodes Quick View*, lists the nodes existing in the monitored Kubernetes environment.
- The Kubernetes Nodes Summary view, which appears on the right after you select an individual node in the **Kubernetes Nodes** tree view.

Kubernetes Nodes Summary view

The **Kubernetes Nodes Summary** view appears on the right when you select a node in the **Kubernetes Nodes** tree view.

Figure 6. Kubernetes Nodes Summary view

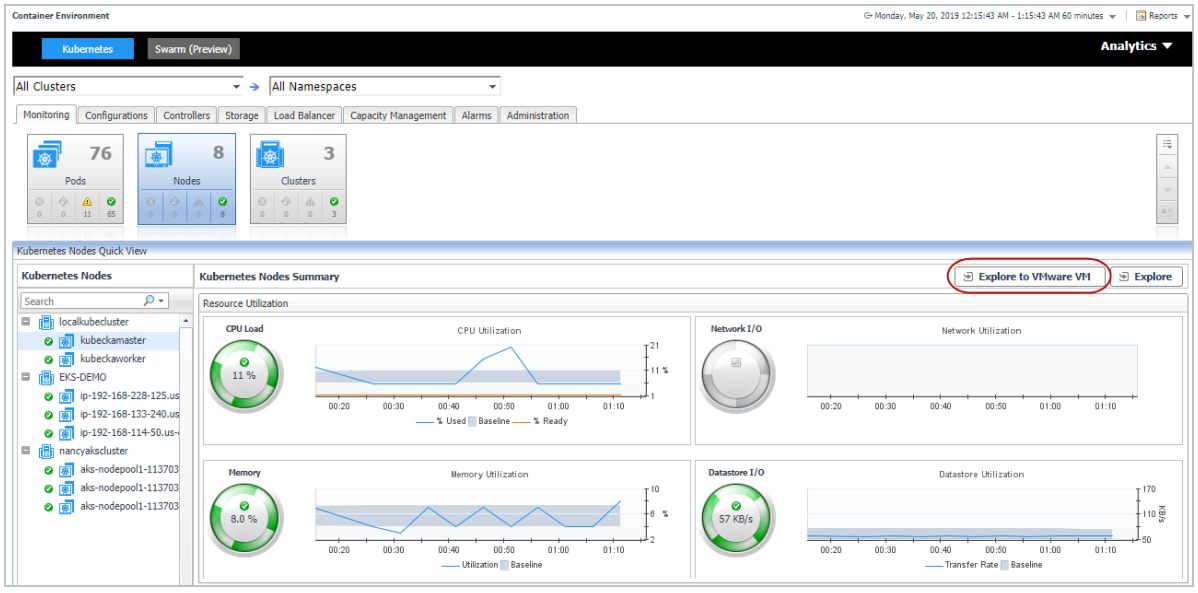
The **Kubernetes Nodes Summary** view displays the following data:

- *Resource Entitlement*: The resource allocation for the selected Kubernetes node over a selected period of time, which includes the following:
  - *CPU Allocatable*: Shows the current allocatable CPU resources of this node.
  - *Memory Allocatable*: Shows the current allocatable Memory resources of this node.
  - *CPU Request*: Shows the trend of CPU request, limit, and capacity of this node.
  - *Memory Request*: Shows the trend of Memory request, limit, and capacity of this node.
- *Resource Utilizations*: The resource utilization for the selected Kubernetes node over a selected period of time, which includes the following:
  - *CPU Utilization*: Shows the CPU utilization summary for the selected Kubernetes node based on its total capacity during a selected time period.
  - *Transfer Rate*: Shows the network utilization summary for the selected Kubernetes node, including the average rate of network throughput, and the amounts of data sent to and received from the network.
  - *Memory Utilization*: Shows the physical memory utilization summary for the selected Kubernetes node, broken into the amounts of memory that is swapped to disk, actively used, and allocated, all during a selected time period.
  - *File System Usage*: Shows the file system resource utilization summary for the selected Kubernetes Pod, including the available/total/limited file system resource.
- *Summary*: Displays the detailed information about the selected Kubernetes node, including *Name, Pod CIDR, OS, Architecture, OS Image, Address, Capacity, Allocatable,* and *Status*.

Click **Explore** on the upper right of the **Kubernetes Nodes Summary** view to open the Nodes Explorer view, which shows more detailed information about this Kubernetes node.

> **NOTE:** If the virtual machines belong to the Container cluster, and they are monitored by other cartridges at the same time, for example, VMware, Infrastructure, AWS, or Azure. Then, the collected data will come from that cartridge directly and the **Kubernetes Nodes Summary** view will be different from the screenshot above. See Figure 7 for detailed information.

Figure 7. Kubernetes Nodes Summary view for VMware



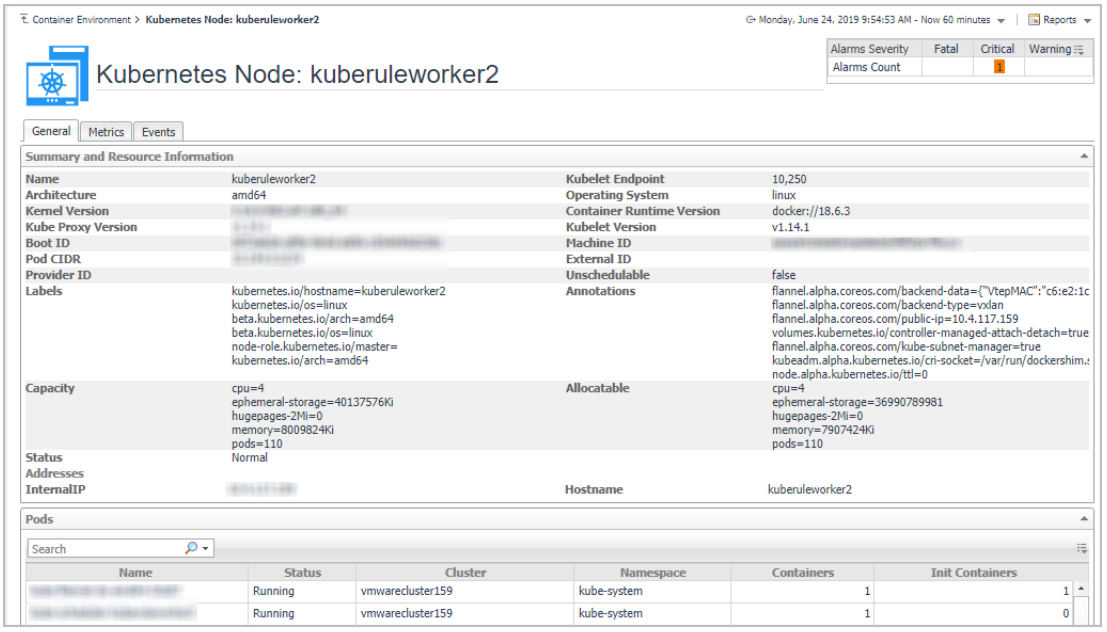
- *Explore to VMware VM*: Click the button to open the *VMware Explorer* view, which is the same view from VMware cartridge. The *Explore to xx* button varies from the cartridge that is monitoring the machines. Currently, the supported cartridges include: VMware, Infrastructure, AWS, and Azure.
- *Resource Utilizations*: The displayed metrics will be slight different among different monitoring cartridges.

Nodes Explorer view

The *Nodes Explorer* view opens when you click **Explore** in the Kubernetes Nodes Summary view, which includes the following tabs:

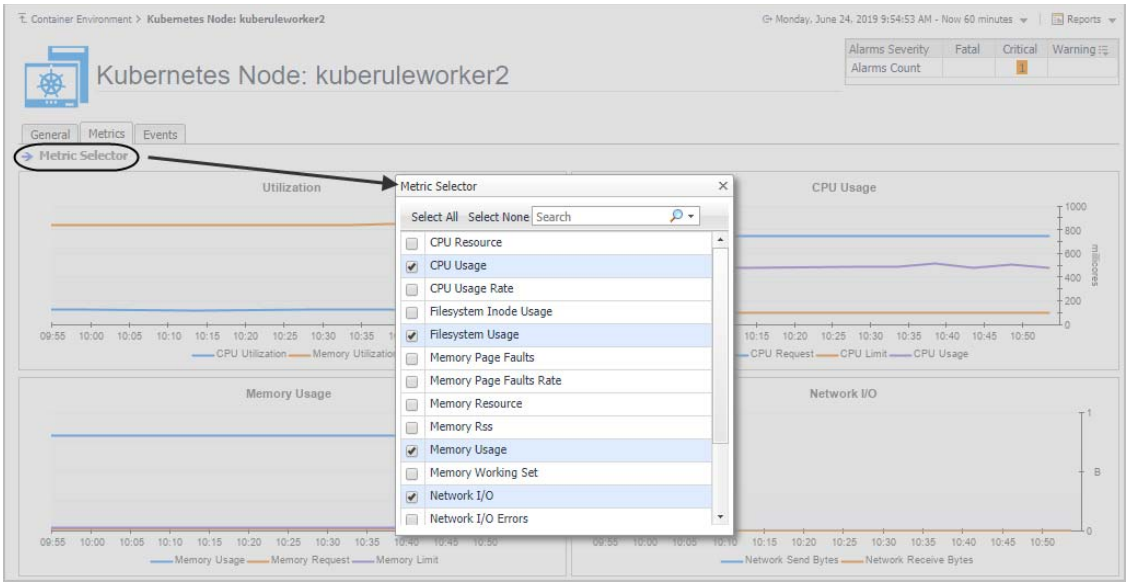
- *General tab:* The *General* tab displays the overall information of the selected Kubernetes node over a selected period of time, including the *Summary and Resource Information* table and the *Pods* table. For more information, see Node metrics *on page 61*.

Figure 8. Kubernetes Nodes Explorer view General Tab



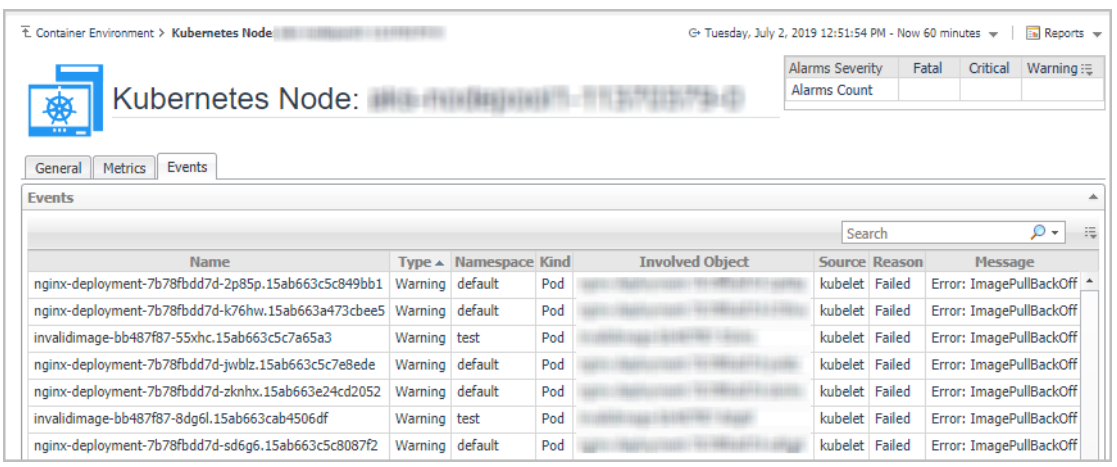
- *Metrics tab:* The *Metrics* tab displays a *Metric Selector* allowing you to choose the metrics to be plotted on this dashboard. Charts of *CPU Usage*, *Utilization*, *Memory Usage*, and *Network I/O* are presented by default.

Figure 9. Kubernetes Nodes Explorer view Metrics Tab



- *Event* tab: The *Event* tab lists all the events occur on the nodes.
  - Name: name of the event.
  - Type: type of the event, Warning or Normal.
  - Namespace: namespace of where this event happens.
  - Kind: type of the Kubernetes component on which this event occurs.
  - Involved Object: name of the Kubernetes component on which this event occurs.
  - Source: where this event has been triggered from.
  - Reason: reason of this event.
  - Message: detailed message of this event.

Figure 10. Kubernetes Nodes Explorer view Events tab



Monitoring Kubernetes Clusters

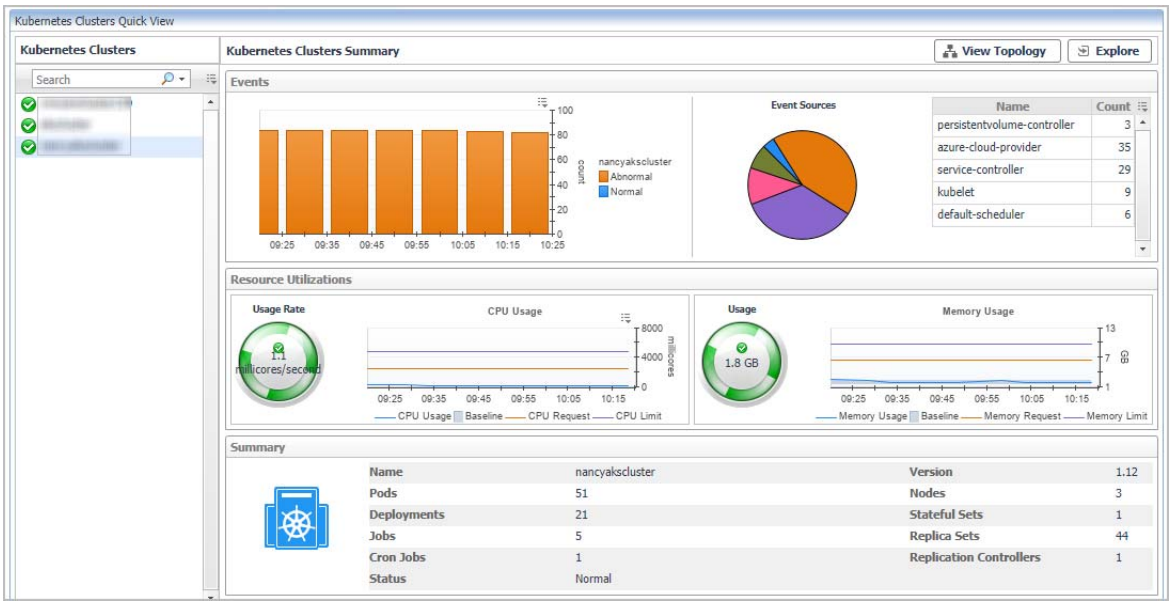
Kubernetes cluster is a group of kubernetes resources. There are two kinds of nodes inside a cluster, Kubernetes master and Kubernetes nodes. Kubernetes master is responsible for maintaining the desired state of your cluster which Kubernetes node is responsible to run your application and cloud workflows.The *Kubernetes Cluster Quick View*, which appears after clicking **Monitoring > Clusters**, shows the data collected about the selected clusters and namespaces. This view consists of the following two panes:

- The **Kubernetes Clusters** tree view, which appears on the left of *Kubernetes Clusters Quick View*, lists the clusters existing in the monitored Kubernetes environment.
- The Kubernetes Clusters Summary view, which appears on the right after you select an individual cluster in the **Kubernetes Clusters** tree view.

Kubernetes Clusters Summary view

The **Kubernetes Clusters Summary** view appears on the right when you select a node in the **Kubernetes Clusters** tree view.

Figure 11. Kubernetes Clusters Summary view



The **Kubernetes Clusters Summary** view displays the following data:

- *Events*: The events occur on the selected Kubernetes cluster over a selected period of time, which includes:
  - The column chart on the left: Shows the timeline of the occurred events, which indicates at what time and how many events have occurred.
  - The pie chart on the right- Event Sources: Shows the events distribution for different event source.
- *Resource Utilizations*: The resource utilization for the selected Kubernetes cluster over a selected period of time, which includes the following:
  - *Usage Rate*: Shows the CPU usage summary for the selected Kubernetes cluster based on its total capacity during a selected time period.
  - *Memory Usage*: Shows the physical memory utilization summary for the selected Kubernetes node, broken into the amounts of memory that is swapped to disk, actively used, and allocated, all during a selected time period.
- *Summary*: Displays the detailed information about the selected Kubernetes cluster, including *Name, Version, Pods, Nodes, Deployments, Stateful Sets, Jobs,* and *Replica Sets*.

Click **Explore** on the upper right of the **Kubernetes Clusters Summary** view to open the Cluster Explorer view, which shows more detailed information about this Kubernetes cluster.

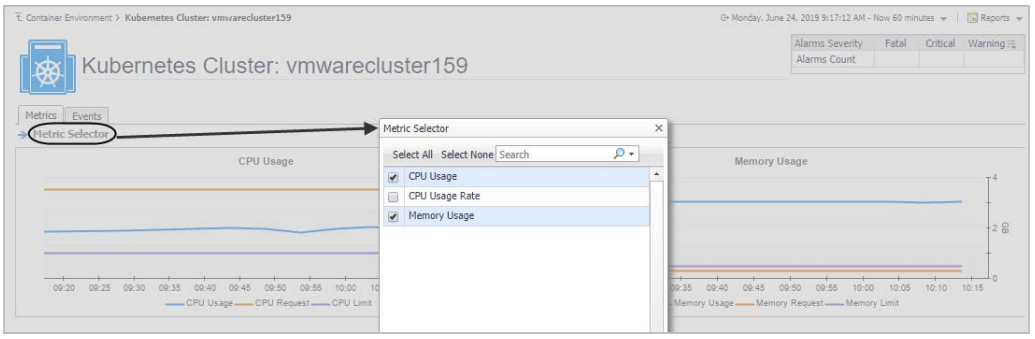
Click **View Topology** on the upper right of the **Kubernetes Clusters Summary** view to open the Cluster Topology view, which shows the topology graph from the application accessible aspect.

Cluster Explorer view

The *Cluster Explorer* view opens when you click **Explore** in the Kubernetes Clusters Summary view, which includes the following tabs:

- *Metrics tab:* The *Metrics* tab displays a *Metric Selector* allowing you to choose the metrics to be plotted on this dashboard. Charts of *CPU Usage* and *Memory Usage* are presented by default.

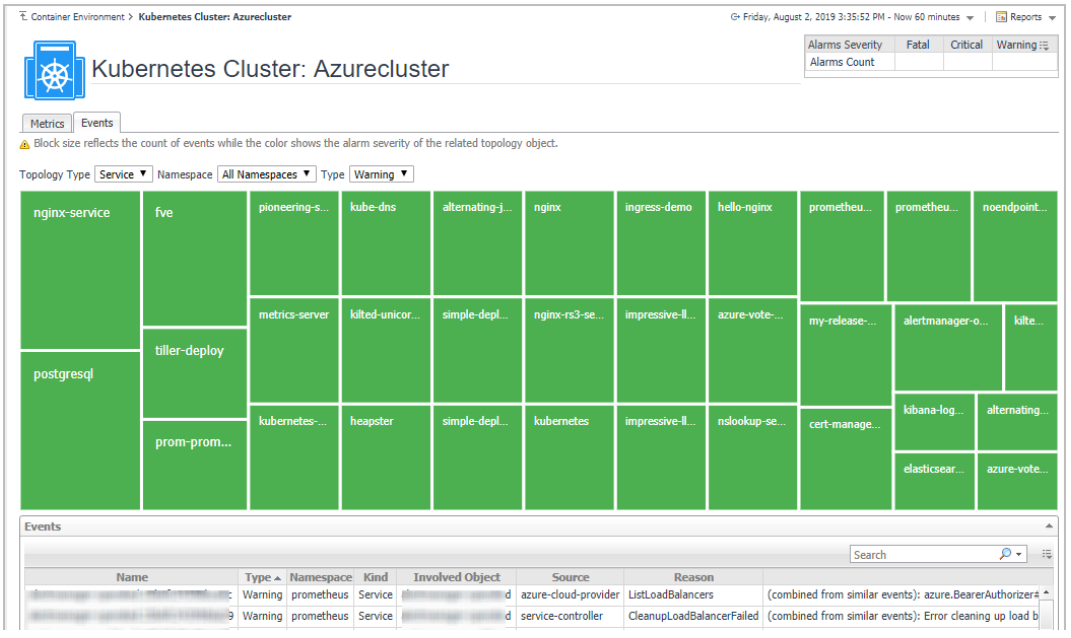
Figure 12. Kubernetes Clusters Explorer view Metrics tab



- Events tab: The Events tab shows a Heat Map of the events occur in this cluster. Heat maps will be refreshed automatically when you change either of the following fields:
  - *Topology Type*: Indicates the Kubernetes components on which the event occurs, including Pod, Node, and Service.
  - *Namespace*: Use the namespaces to filter the events.
  - *Type*: Indicates the severity of the event, including warning and normal.

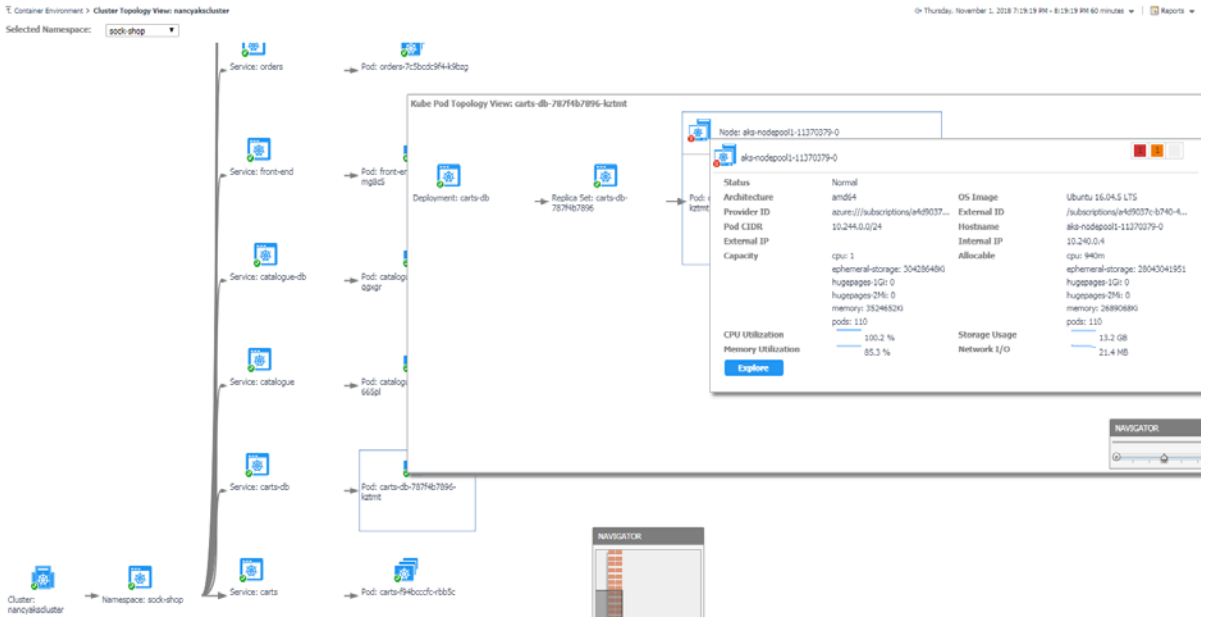
NOTE: The color in the heatmap indicates the severity of component alarms.
Green: indicates normal. Yellow: indicates warning. Orange: indicates critical. Red: indicates fatal.

Figure 13. Kubernetes Clusters Explorer view Events tab



Cluster Topology view

Figure 14. Kubernetes Clusters Topology view



The *topology view* visualizes the relationships between the objects from the pods accessible aspect in your environment through an interactive dependency map. The map illustrates how different components relate to each other, and the levels of the available resources available to them. Click on Pod, another sub topology view will popup to show the relationship from pods controller to storage for the selected Pod. Click other components or click the Pod inside the sub topology view, an information view will popup to show alarms, basic information, some metrics. From the information popup view of Pod, Node and Cluster, click the Explore button will navigate to the explorer view of the selected Pod/Node/Cluster. The **NAVIGATOR** in the bottom-right corner allows you to easily set the zoom level by dragging the slider into the appropriate position.

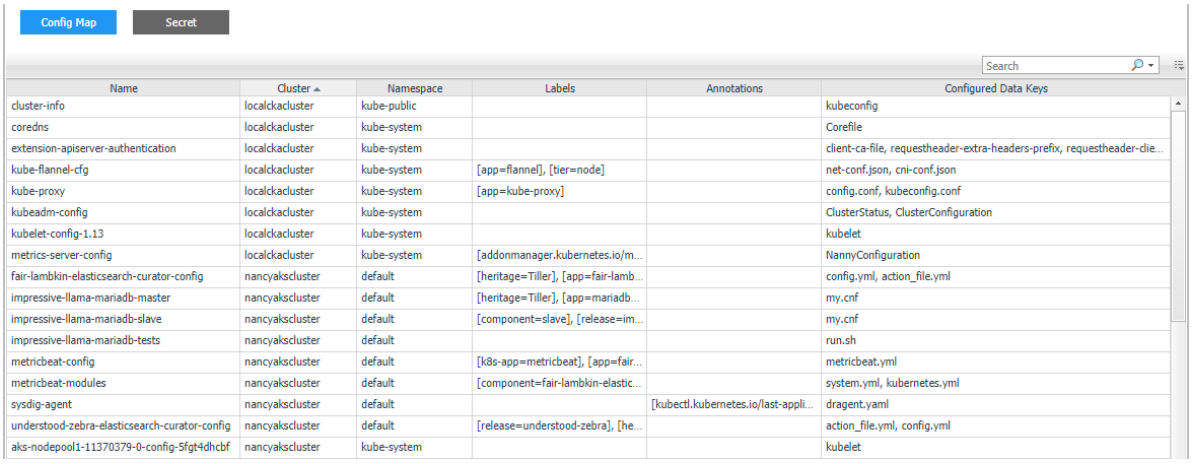
Monitoring Kubernetes Other Components

Kubernetes other components here including pods controllers, services, ingresses, persistent volumes, secrets and so on. All these components are grouped and displayed in tabs.

- Configurations
- Controllers
- Storage
- Load Balancer

Configurations

Figure 15. Kubernetes Configuration Dashboard



Config Map | Secret

Search

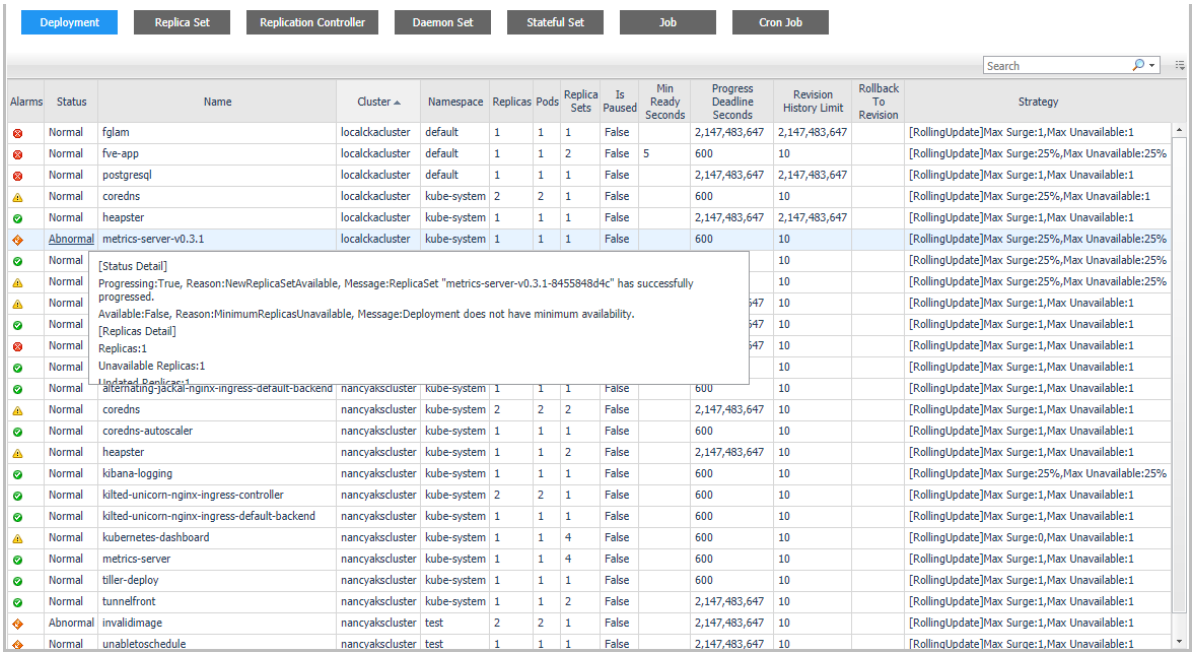
Name	Cluster	Namespace	Labels	Annotations	Configured Data Keys
cluster-info	localckacluster	kube-public			kubeconfig
coredns	localckacluster	kube-system			Corefile
extension-apiserver-authentication	localckacluster	kube-system			client-ca-file, requestheader-extra-headers-prefix, requestheader-clie...
kube-flannel-cfg	localckacluster	kube-system	[app=flannel], [tier=node]		net-conf.json, cni-conf.json
kube-proxy	localckacluster	kube-system	[app=kube-proxy]		config.conf, kubeconfig.conf
kubeadm-config	localckacluster	kube-system			ClusterStatus, ClusterConfiguration
kubelet-config-1.13	localckacluster	kube-system			kubelet
metrics-server-config	localckacluster	kube-system	[addonmanager.kubernetes.io/m...		NannyConfiguration
fair-lambkin-elasticsearch-curator-config	nancyakscluster	default	[heritage=Tiller], [app=fair-lamb...		config.yml, action_file.yml
impressive-llama-mariadb-master	nancyakscluster	default	[heritage=Tiller], [app=mariadb...		my.cnf
impressive-llama-mariadb-slave	nancyakscluster	default	[component=slave], [release=im...		my.cnf
impressive-llama-mariadb-tests	nancyakscluster	default			run.sh
metricbeat-config	nancyakscluster	default	[k8s-app=metricbeat], [app=fair...		metricbeat.yml
metricbeat-modules	nancyakscluster	default	[component=fair-lambkin-elastic...		system.yml, kubernetes.yml
sysdig-agent	nancyakscluster	default		[kubectl.kubernetes.io/last-appli...	dragent.yaml
understood-zebra-elasticsearch-curator-config	nancyakscluster	default	[release=understood-zebra], [he...		action_file.yml, config.yml
aks-nodepool1-11370379-0-config-5fgt4dhcbf	nancyakscluster	kube-system			kubelet

The *Configurations* dashboard includes Kubernetes Secret and Config Map.

- A Kubernetes Secret is an object that contains a small amount of sensitive data, such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in an image; putting it in a Secret object allows for more control over how it is used, and reduces the risk of accidental exposure.
- A Kubernetes Config Map binds configuration files, command-line arguments, environment variables, port numbers, and other configuration artifacts to your Pods' containers and system components at runtime. Config maps allow you to separate your configurations from your Pods and components, which helps keep your workloads portable, makes their configurations easier to change and manage, and prevents hardcoding configuration data to Pod specifications.

Controllers

Figure 16. Kubernetes Controllers Dashboard



Deployment | Replica Set | Replication Controller | Daemon Set | Stateful Set | Job | Cron Job

Search

Alarms	Status	Name	Cluster	Namespace	Replicas	Pods	Replica Sets	Is Paused	Min Ready Seconds	Progress Deadline Seconds	Revision History Limit	Rollback To Revision	Strategy
	Normal	fglam	localckacluster	default	1	1	1	False		2,147,483,647	2,147,483,647		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	fve-app	localckacluster	default	1	1	2	False	5	600	10		[RollingUpdate]Max Surge:25%,Max Unavailable:25%
	Normal	postgresql	localckacluster	default	1	1	1	False		2,147,483,647	2,147,483,647		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	coredns	localckacluster	kube-system	2	2	1	False		600	10		[RollingUpdate]Max Surge:25%,Max Unavailable:1
	Normal	heapster	localckacluster	kube-system	1	1	1	False		2,147,483,647	2,147,483,647		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Abnormal	metrics-server-v0.3.1	localckacluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:25%,Max Unavailable:25%
	Normal										10		[RollingUpdate]Max Surge:25%,Max Unavailable:25%
	Normal										10		[RollingUpdate]Max Surge:25%,Max Unavailable:25%
	Normal									647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal									647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal									647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal										10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	alternating-jackal-nginx-ingress-default-backend	nancyakscluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	coredns	nancyakscluster	kube-system	2	2	2	False		2,147,483,647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	coredns-autoscaler	nancyakscluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	heapster	nancyakscluster	kube-system	1	1	2	False		2,147,483,647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	kibana-logging	nancyakscluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:25%,Max Unavailable:25%
	Normal	kilted-unicorn-nginx-ingress-controller	nancyakscluster	kube-system	2	2	1	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	kilted-unicorn-nginx-ingress-default-backend	nancyakscluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	kubernetes-dashboard	nancyakscluster	kube-system	1	1	4	False		600	10		[RollingUpdate]Max Surge:0,Max Unavailable:1
	Normal	metrics-server	nancyakscluster	kube-system	1	1	4	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	tiller-deploy	nancyakscluster	kube-system	1	1	1	False		600	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	tunnelfront	nancyakscluster	kube-system	1	1	2	False		2,147,483,647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Abnormal	invalidimage	nancyakscluster	test	2	2	1	False		2,147,483,647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1
	Normal	unabletoschedule	nancyakscluster	test	1	1	1	False		2,147,483,647	10		[RollingUpdate]Max Surge:1,Max Unavailable:1

[Status Detail]
Progressing:True, Reason:NewReplicaSetAvailable, Message:ReplicaSet "metrics-server-v0.3.1-8455848d4c" has successfully progressed.
Available:False, Reason:MinimumReplicasUnavailable, Message:Deployment does not have minimum availability.
[Replicas Detail]
Replicas:1
Unavailable Replicas:1
Updated Replicas:1

A controller manages a set of pods and ensures that the cluster is in the specified state. Instead of manually creating a pod, controllers can be used to create pods and to manage the pods. For example, the pods maintained by a replication controller are automatically replaced if they fail, get deleted, or are terminated. The *Controllers* dashboard presents the information related to the following controller types: *Deployment, Replica Set, Replication Controller, Daemon Set, Stateful Set, Job,* and *Cron Job*.

Storage

Figure 17. Kubernetes Storage Dashboard

Persistent Volume | Persistent Volume Claim | Storage Class

Search

Alarms	Status	Name	Cluster	Reclaim Policy	Claim	Storage Class	Source Type
	Bound	pv-sc-no-sc-customize	localckacluster	Retain	pvc-sc-pv-customize-sc	no-sc-customize	HostPath
	Available	pv-sc-default	localckacluster	Retain		default	HostPath
	Available	pv-sc-invalid-provisioner	localckacluster	Retain		sc-invalid-provisio...	HostPath
	Bound	pv-invalid-nfs	localckacluster	Recycle	pvc-invalid-sc-pv	slow	NFS
	Available	pv-pvc-oversize	localckacluster	Retain		sc-oversize	HostPath
	Available	pv-pvc-acm1	localckacluster	Retain		sc-pvc-acm1	HostPath
	Bound	pvc-2b95e22d-dc28-11e8-b2ed-befa22179703	nancyakscluster	Delete	data-mehdb-1	default	AzureDisk
	Bound	pvc-45f1fe1e-5f54-11e9-b660-16063de8b09f	nancyakscluster	Delete	data-elasticsearch-2	default	AzureDisk
	Bound	pvc-59cb23a5-fd17-11e8-adf4-de8994810bc3	nancyakscluster	Delete	data-elasticsearch-0	default	AzureDisk
	Bound	pvc-7049bcb8-fd17-11e8-adf4-de8994810bc3	nancyakscluster	Delete	data-elasticsearch-1	default	AzureDisk
	Bound	pvc-90f76a94-2e94-11e9-810c-0a130f143c9f	nancyakscluster	Delete	alertmanager-prom-prometheus-operator-alertmanager-db-al...	default	AzureDisk
	Bound	pvc-9727fba6-2e94-11e9-810c-0a130f143c9f	nancyakscluster	Delete	prometheus-prom-prometheus-operator-prometheus-db-prom...	default	AzureDisk

The Kubernetes storage contains volumes, storage class, persistent volume, and persistent volume claim. Volumes are on-disk files used by the containers for persistent their data as well as sharing with other containers.The *Storage* dashboard shows the information about the following storage classes:

- *Storage Class* provides a way for the administrator to describe the "class" of storage they offer.
- *Persistent Volume* subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed.
- *Persistent Volume Claim* is used for dynamic volume provisioning which allow storage volumes to be created on-demand.

Load Balancer

Figure 18. Kubernetes Load Balancer Dashboard

Service | Ingress | Endpoint

Alarms	Name	Cluster	Namespace	Type	Cluster IP	External IPs	External Name	IP Address
	fve	localckacluster	default	NodePort	10.103.174.0			
	fve-app	localckacluster	default	ClusterIP	10.111.36.15	[redacted]		
	kubernetes	localckacluster	default	ClusterIP	10.96.0.1			
	postgresql	localckacluster	default	ClusterIP	10.98.70.60			
	test	localckacluster	default	ClusterIP	10.99.240.162			
	heapster	localckacluster	kube-system	ClusterIP	10.110.175.65			
	kube-dns	localckacluster	kube-system	ClusterIP	10.96.0.10			
	metrics-server	localckacluster	kube-system	ClusterIP	10.109.149.172			
	kubernetes	localkubecluster159	default	ClusterIP	10.96.0.1			
	kube-dns	localkubecluster159	kube-system	ClusterIP	10.96.0.10			
	azure-vote-back	nancyakscluster	default	ClusterIP	10.0.71.228			
	azure-vote-front	nancyakscluster	default	LoadBalancer	10.0.116.152			
	hello-nginx	nancyakscluster	default	NodePort	10.0.28.128			

The *Load Balancer* dashboard includes information about Kubernetes service, endpoint, and ingress. A Kubernetes ingress can provide load balancing, SSL termination, and name-based virtual hosting. A Kubernetes service is an abstraction which defines a logical set of pods and a policy by which to access them - sometime called micro-services. Kubernetes will update the endpoint whenever the set of pods in a service changes.

Alarms

Figure 19. Kubernetes Alarms Dashboard

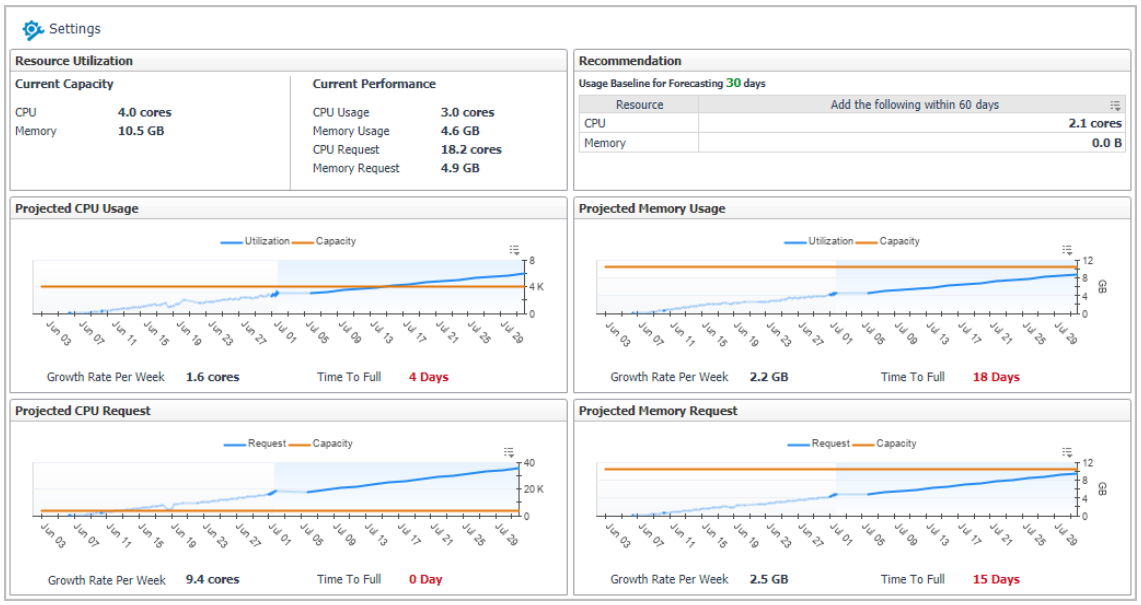
The *Alarms* dashboard displays a list of alarms generated against the monitored Kubernetes environment. Use this view to quickly identify any potential problems related to a specific Kubernetes component.

Capacity Management

Foglight for Container Management provide capacity management feature for Kubernetes. This feature uses historical data to predict the trend and usage within a specific future period.

> **NOTE:** If the Capacity Management tab is not displayed, ensure the following:
> 1. You have purchased a license for Capacity Management. If not, contact Quest Support to purchase a license.
> 2. You have the Container Administrator role.

Figure 20. Capacity Management for Kubernetes



The Capacity Management dashboard contains the following fields:

- *Setting*: Click to change the following values:
  - *Baseline for Forecasting*: Defines the historical period used for the calculations of metric views, current capacity, and recommended resources in the Resource Utilization view. The default value is *60 Days Trending*.
  - *Time Frame*: Defines the predicted period for calculating metric views, current capacity, and recommended resources in the Resource Utilization view. The default value is *Next 30 Days*.
- *Resource Utilization*:
  - *Current Capacity*: current resource capacity.
  - *Current Performance*: current resource usage.
- *Recommendation*:

  In this section, it shows how many resources are recommended to be added in the current trend, so as to meet the predicated usage.
- *Projected CPU/Memory Usage*: Shows the historical data and the predicted usage trend within the configured future period.
- *Projected CPU/Memory Request*: Shows the historical data and the predicted request trend within the configured future period.
  - *Utilization*: usage.
  - *Capacity*: upper bound which the usage might reach.
  - *Growth Rate per Week*: growth amount of the resource.
  - *Time to Full*: how many days the resource usage/request will reach the capacity.

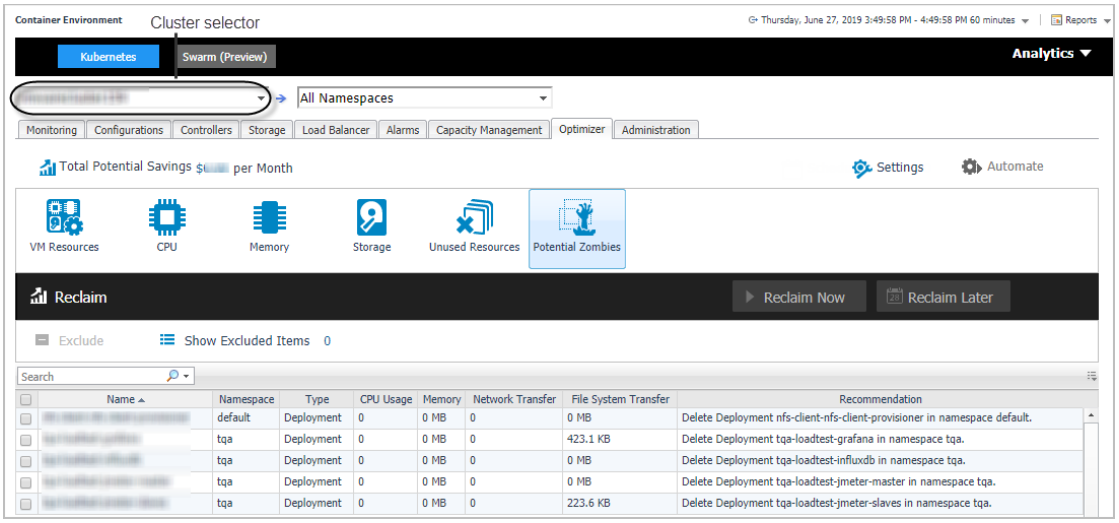
  NOTE: If a value *Never* is displayed at *Time to Full*, which means the usage/request trend is declining and the usage/request will never reach the capacity.

Optimizer

The *Optimizer* view appears after clicking **Container** > **Kubernetes** > **Optimizer**.

> **NOTE:** If the Optimizer tab is not displayed, ensure the following:
> 1. You have purchased a license for Optimizer. If not, contact Quest Support to purchase a license.
> 2. You have the Container Administrator role.
>
> **NOTE:** The displayed views are varied if the cluster hosts are monitored by the agents of VMware, AWS, or Azure.

Figure 21. Kubernetes Optimizer Dashboard



The Optimizer view includes the following elements:

- **Cluster Selector**. The cluster selector is located at the top of the Optimizer view and allows you to select the environment that you want to optimize.

  > **NOTE:** The Namespace selector doesn't work for Optimizer dashboard.

- **Settings**. The Settings dialog box is used to change the time period and properties that are used for calculation. For more information, see Settings on page 40.
- **Automate**. Use the Automate menu to set the criteria for automatically sending recommendations for improvements. Currently, this button only functions for **CPU** and **Memory** when a VMware cluster is selected.
- **Reclaim Now** and **Reclaim Later** buttons. System administrator can select a VM from the list and review the Reclaiming Savings bar for information about how many resources can be reclaimed.

> **NOTE:** The **Reclaim Now** and **Reclaim Later** buttons are enabled only after selecting a checkbox from the table. Currently, the two buttons only function for **VM Resources**, **CPU**, and **Memory** when a VMware cluster is selected.
> The **Automate**, **Reclaim Now**, and **Reclaim Later** buttons are displayed only when a VMware cluster is selected.

- **Exclude** button. Select an object you want to exclude from the table to enable the Exclude button, and click Exclude. Then, this object is added to the list of excluded objects under a specific category.
- **Show Excluded Items** button. Click the Show Excluded Items button to view the excluded objects. The Settings dialog box appears. For more information, see
- **VM Resources/VM Resizing**. Shows instance or virtual machine name, utilization, recommendations for both CPU and memory resources, and savings.
- **CPU**. Shows instance or virtual machine name, utilization, recommendations for CPU resource, and estimated savings.

- **Memory**. Shows instance or virtual machine name, utilization, recommendations for memory resource, and estimated savings.
- **Storage**. Shows virtual machine name, utilization, storage and modify recommendations, and savings.

> **NOTE: VM Resizing** will be displayed when a cloud cluster is selected. **VM Resources**, **CPU**, **Memory**, and **Storage** will be displayed when a VMware cluster is selected.

- **Unused Resources** table. Detects and shows those unused resources in container environment.

  For example, persistent volume stays unused for more than 3 months. persistent volume stays in unbound status. This is due to the Unused Resources configuration in Settings.
- **Potential Zombies** table. Detects and shows the potential pod controllers in container environment, including Deployment, Daemon Set, Stateful Set, Replication Controller, as well as Pod that is not managed by any Pod Controller.

  For example, if all pods managed by a pod controller are zombies, then we might suggest you to delete the whole pod controller.

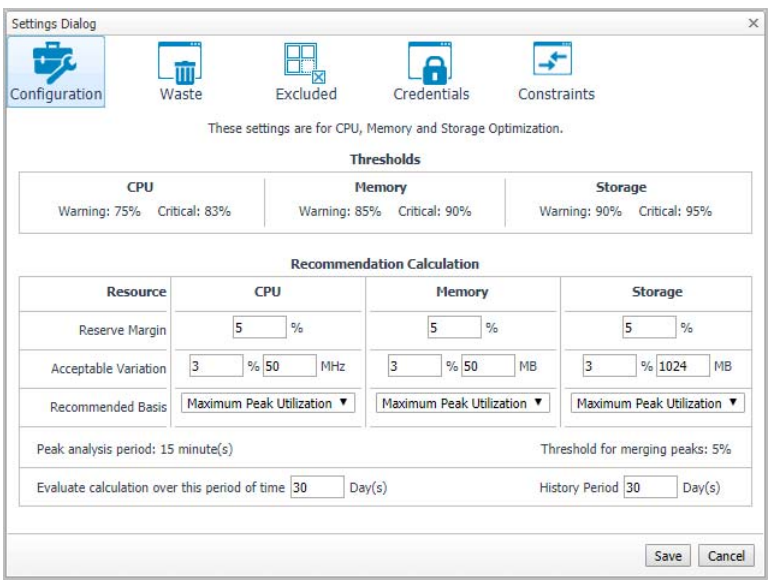
Settings

Use the Settings menu to define the default optimization settings for your environment. The Settings Dialog box provides information about the following components:

- Configuration tab
- Waste tab
- Excluded tab
- Credentials tab
- Constraints tab

Configuration tab

Figure 22. Configuration tab



The **Configuration** tab provides the recommended settings for CPU, memory, and storage optimization.

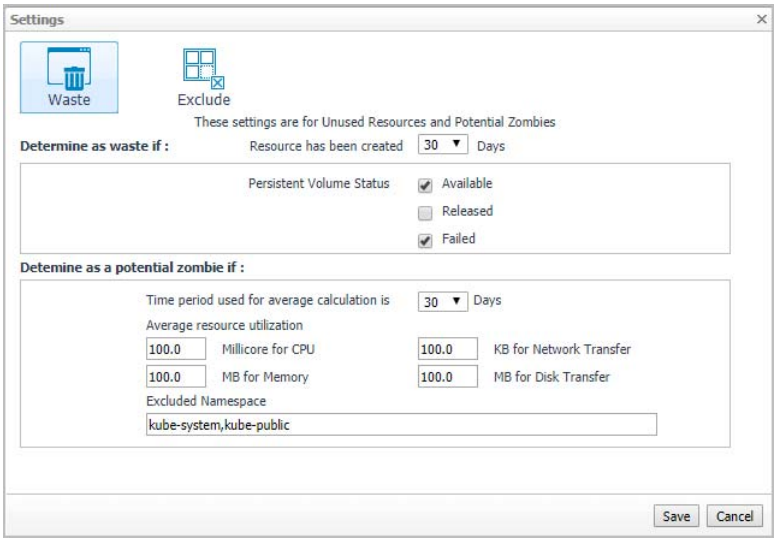
- **Thresholds**. Provides the values of a resource metric that define the Warning and Critical levels (for CPU, memory, and storage).

- **Recommendation Calculation** area. Allows you to define the following parameters for optimizing the CPU, memory resources in your environment, Storage resources not supported at current version:

To save any changes made to the **Configuration** settings, click **Save** at the bottom of the tab.

Waste tab

Figure 23. Waste tab



The **Waste** tab allows you to configure the settings for determining resources wasted in your environment. These include unused resources and potential zombie Pod controllers.

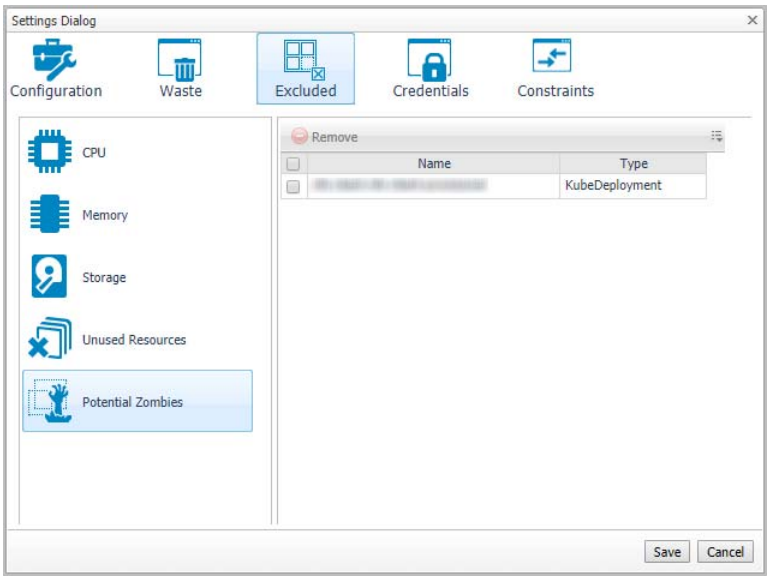
- *Determine as waste if:* used to filter Unused Resources.
  - *Resource has been created [time] days*: Resources that has been created more than the set days will be considered here.
  - *Persistent Volume Status*: By default, select *Available* and *Failed*. For detailed information, go to https://kubernetes.io/docs/concepts/storage/persistent-volumes/#phase.
- *Determine as a potential zombie if*: used to filter Potential Zombies.

  In container environment, Potential Zombie Pod Controller is considered here, including Deployment, Daemon Set, Stateful Set, Replication Controller, and Pod that is not managed by any Pod Controller. Settings work for single pod managed by Pod Controller. If all pods or partial pods of a Pod Controller are considered as zombies, different recommendations will be generated.
  - *Time period used for average calculation is [time] days*: The average metrics for the pods are calculated, so a time range should be set to calculate the average value.
  - *Average resource utilization*: only if a pod's metrics satisfy all the conditions, it will be considered to be a potential zombie pod.
  - *Excluded Namespace*: pods in the namespace can be excluded in the Potential Zombies check.

To save any changes made to the **Waste** settings, click **Save** at the bottom of the tab.

Excluded tab

Figure 24. Excluded tab



The **Excluded** tab allows you to remove a resource from the list of excluded objects. The **Excluded** tab includes the following information:

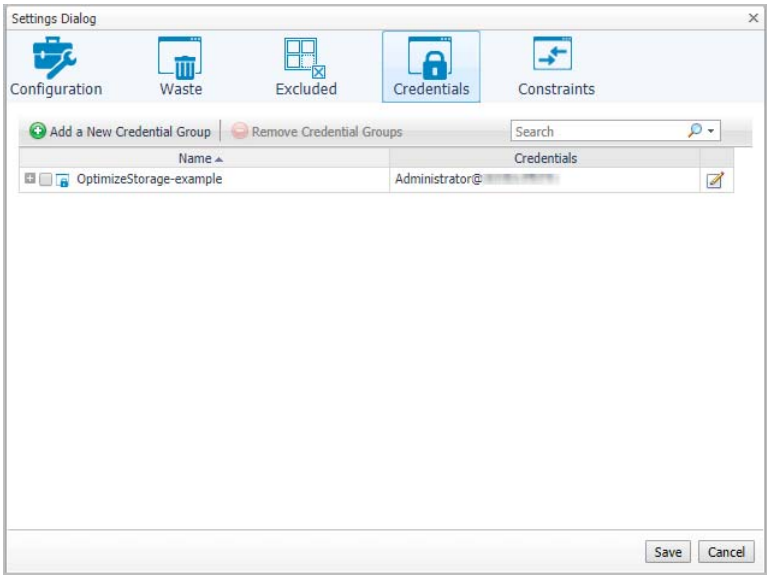
- On the left side, a navigation tree, that allows you to select the resource category.
- On the right side, the list of resources excluded from the selected category.

To remove resources from the list of **Excluded** objects, select the check boxes for these resources and click **Remove**. To save any changes made to the **Excluded** settings, click **Save** at the bottom of the tab.

The **Excluded** tab can also be accesses by clicking **Show Excluded Items** on the **Optimizer** tab.

Credentials tab

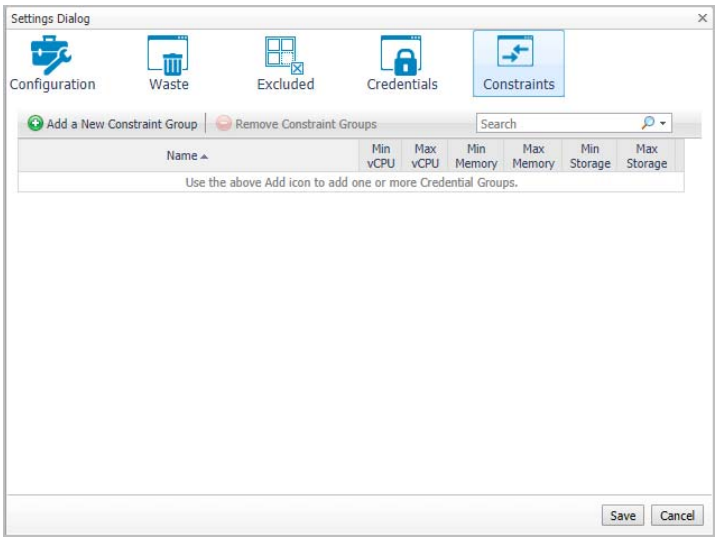
Figure 25. Credentials tab



This tab is available in VMware environment. The **Credentials** tab allows you to add, edit, and remove credentials groups. This tab is only for the Storage rightsizer.

Constraints tab

Figure 26. Constraints tab

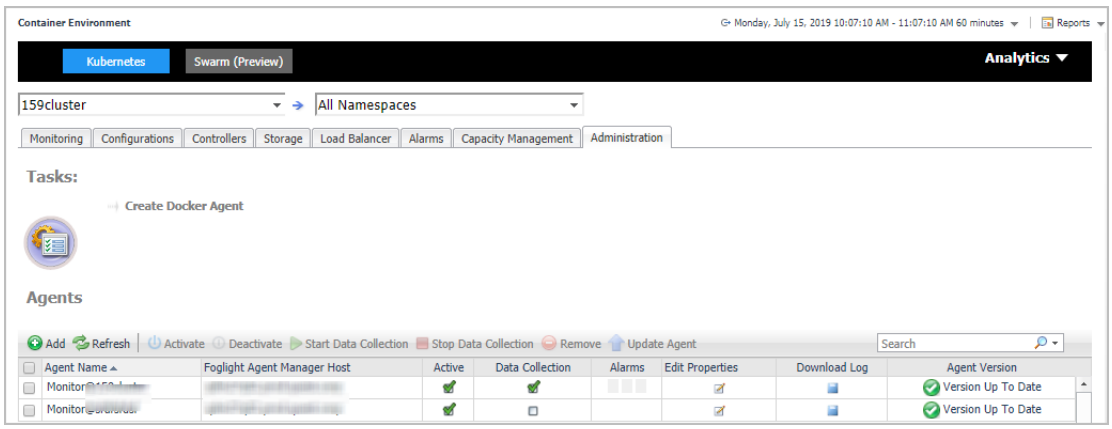


This tab is available in VMware environment. The **Constraints** tab allows you to set custom thresholds for select objects in the environment. These recommendations are displayed in the **Optimizer** tab > **VM Configuration/ CPU/ Memory/ Storage** views > **Modify Recommendation** column. Use this tab to add, edit, and remove constraints groups.

> **IMPORTANT:** A virtual machine may have several partitions. VM environment makes recommendations for each partition separately, but the custom constraints can be set only for the entire VM (not for individual partitions). Therefore, the custom constraint for storage are applied to all partitions on the selected VM.

Administration

Figure 27. Kubernetes Administration Dashboard



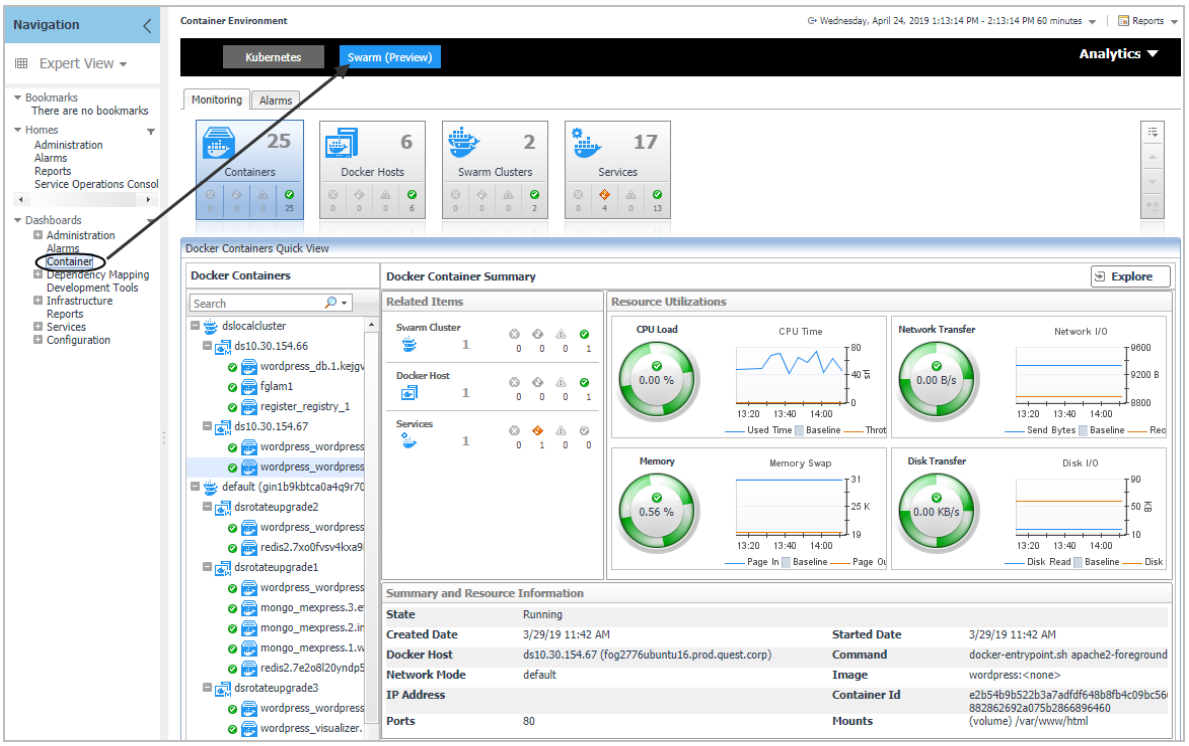
The *Administration* dashboard supports new agent creation and existing agents management. Use *Create Docker Agent* or *Add* to create a new agent. See Creating and Activating a Kubernetes Agent on page 19 for more information. Use *Activate*, *Deactivate*, *Start Data Collection*, *Stop Data Collection, Remove*, and *Update Agent* to manage the agent. Click *Edit Properties* to update the properties of the chosen agent.

> **NOTE:** The *Administration* dashboard can be accessed only when the users have both the Administrator role and the Container Administrator role. To grant the users with the Container Administrator role, go to **Administration** > **Users & Security** management under *Administer Server* > **Manage Users, Groups, Roles** > **Roles** tab.

Docker Swarm

The *Docker Container Quick View* appears after clicking **Monitoring > Containers**. Click **Swarm (Preview)** from the header on top to switch to Docker Swarm dashboard.

Figure 28. Docker Swarm Dashboard



Monitoring Docker Containers

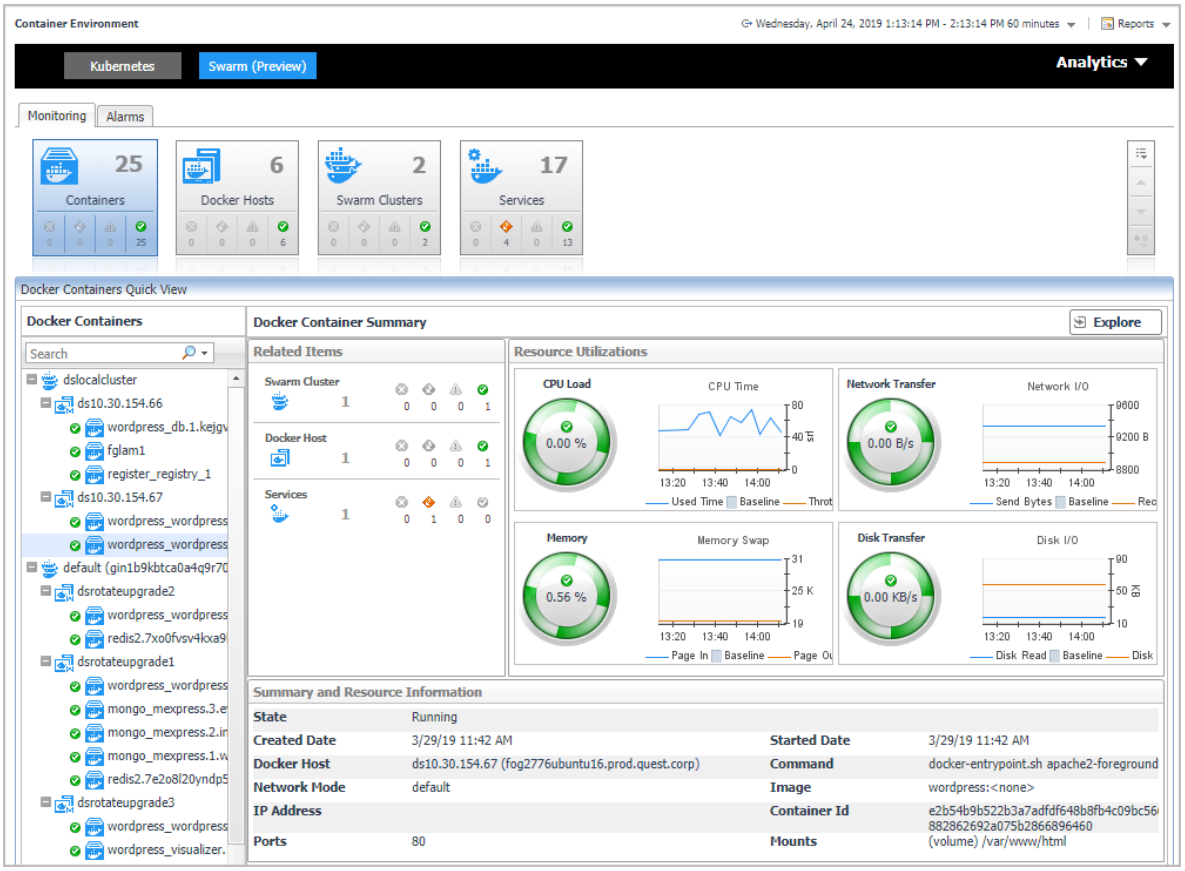
This view consists of the following two panes:

- The **Docker Containers** tree view, which appears on the left of *Docker Containers Quick View*, lists the containers existing in the monitored *Docker* environment. The containers in the tree view are grouped by **cluster > docker host > container**.
- The Docker Container Summary view, which appears on the right after you select an individual container in the **Docker Containers** tree view.

Docker Container Summary view

The **Docker Container Summary** view appears on the right when you select a container in the **Docker Containers** tree view.

Figure 29. Docker Container Summary view



The **Docker Container Summary** view displays the following data:

- *Related Items: Shows the related Docker components grouped by type as well as the associated alarms.*
- *Resource Utilizations*: The resource utilization for the selected Docker Container over a selected period of time, which includes the following:
  - *CPU Load*: Shows the CPU utilization of the selected container.
  - CPU Time: Shows the used time and throttled time of the selected container.
  - *Network Transfer*: Shows the transfer bytes rate of the selected container over a selected period of time.
  - Network I/O: Shows the total send/receive bytes of the selected container.
  - *Memory*: Shows the memory utilization of the selected container.
  - Memory Swap: Shows the mounts of memory pages that are swapped to disk.
  - *Disk Transfer*: Shows the disk transfer bytes rate of the selected container over a selected period of time.
  - Disk I/O: Shows the disk read/write bytes of the selected container.
- *Summary and Resource Information*: Displays the detailed information about the selected Container, including *State, Command, Created Time, Started Time, Image,* and *so on*.

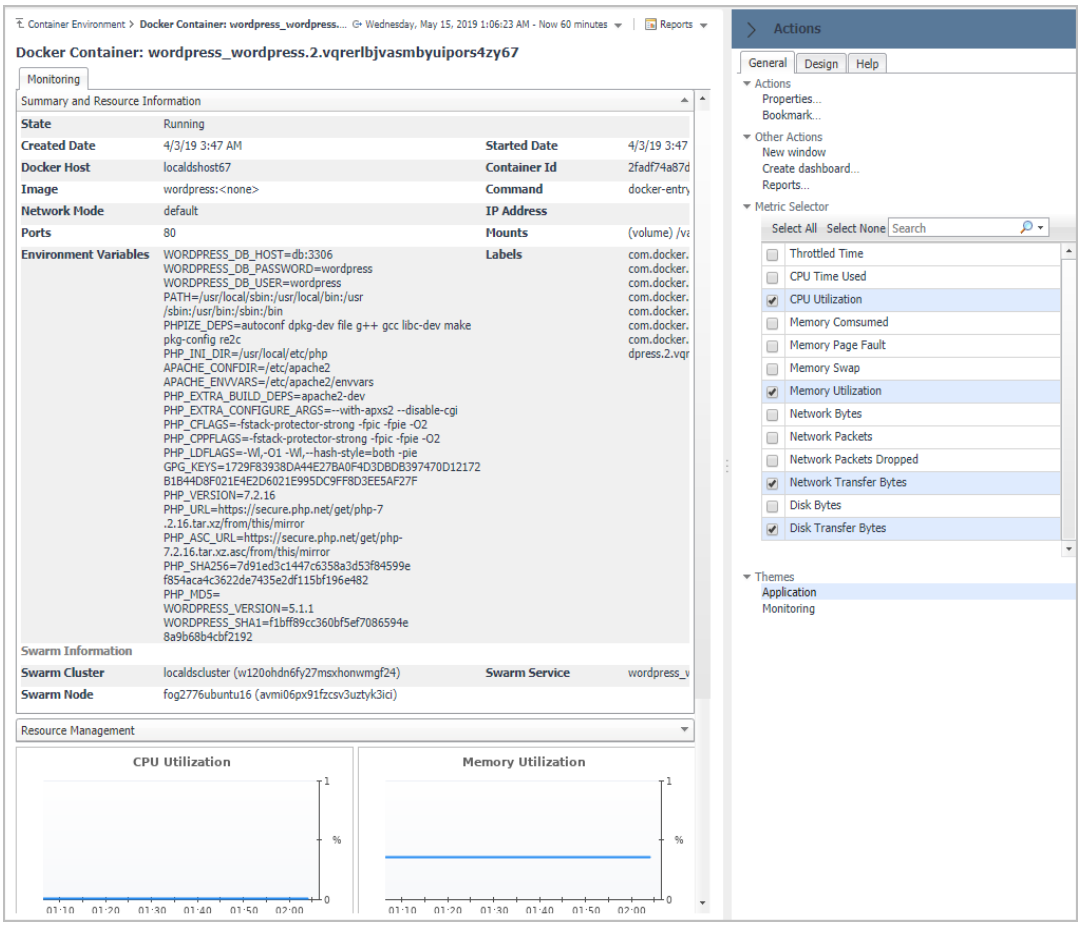
Click **Explore** on the upper right of the **Docker Container Summary** view to open the Container Explorer view, which shows more detailed information about this container.

Container Explorer view

The *Container Explorer* view opens when you click **Explore** in the Docker Container Summary view, which includes the following tabs:

Monitoring tab: The *Monitoring* tab displays the overall information of the selected container over a selected period of time, including the *Summary and Resource Information* table, Resource Management table as well as the Metrics list. To set the Metrics list displayed, go to **Action** > **General** > **Metric Selector**. For more information, see Container metrics on page 61.

Figure 30. Docker Container Explorer view Monitoring Tab



Monitoring Docker Hosts

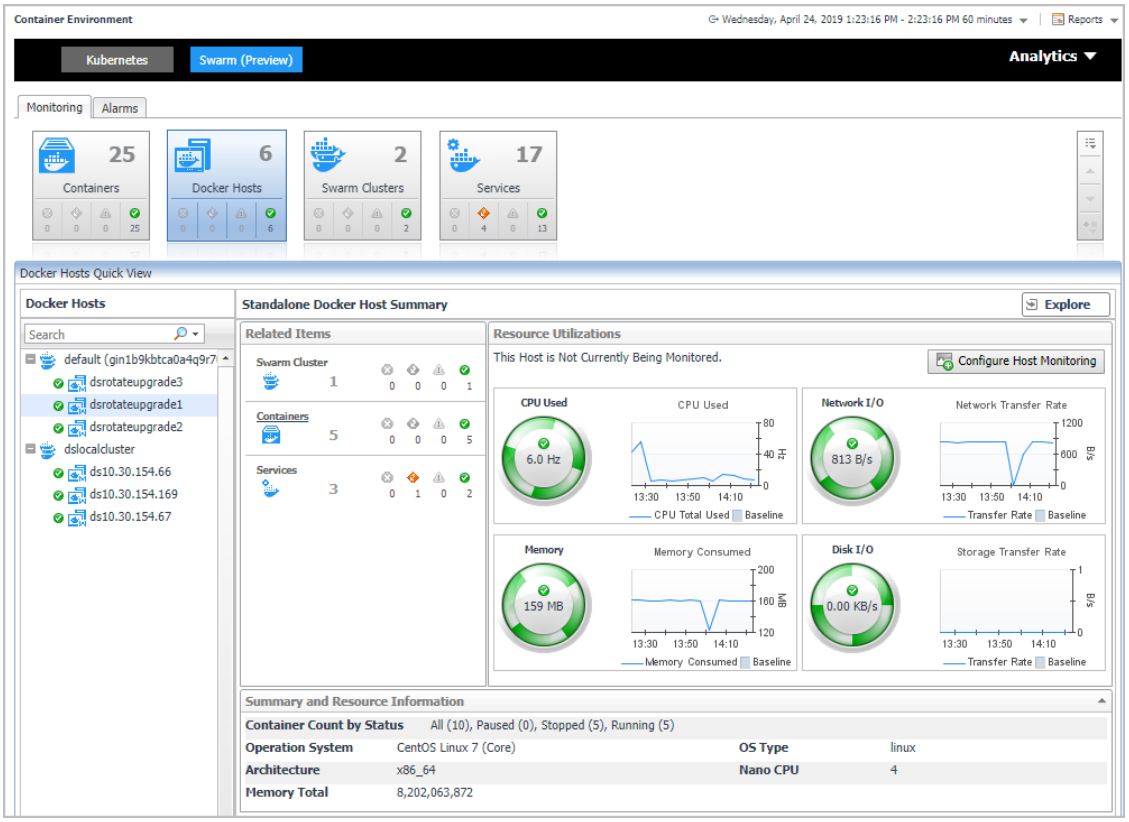
This view consists of the following two panes:

- The **Docker Hosts** tree view, which appears on the left of *Docker Hosts Quick View*, lists the docker hosts existing in the monitored *Docker* environment. The docker hosts in the tree view are grouped by **cluster > docker host**.
- The Docker Host Summary view appears on the right after you select an individual docker host in the **Docker Hosts** tree view.

Docker Host Summary view

The **Docker Host Summary** view appears on the right when you select a docker host in the **Docker Hosts** tree view.

Figure 31. Docker Host Summary view



The **Docker Host Summary** view displays the following data:

- *Related Items: Shows the related Docker components grouped by type as well as the associated alarms.*
- *Resource Utilizations*: The resource utilization for the selected docker host over a selected period of time, which includes the following:
  - *CPU Load*: Shows the CPU utilization of the selected docker host.
  - *CPU Used*: Shows the used CPU resources aggregated from the containers running on the docker host.
  - *Network I/O* and Network Transfer Rate: Shows the transfer bytes rate of the selected docker host aggregated from the containers running on the docker host over a selected period of time.
  - *Memory* and Memory Consumed: Shows the memory consumed bytes aggregated from the containers running on the docker host.
  - *Disk I/O* and Disk Transfer: Shows the disk transfer bytes rate of the selected docker host aggregated from the containers running on the docker host over a selected period of time.
- *Summary and Resource Information*: Displays the detailed information about the selected docker host, including *Container Count by Status, Operating System, Memory Total,* and *so on*.

Click **Explore** on the upper right of the **Docker Host Summary** view to open the Docker Host Explorer view, which shows more detailed information about this container.

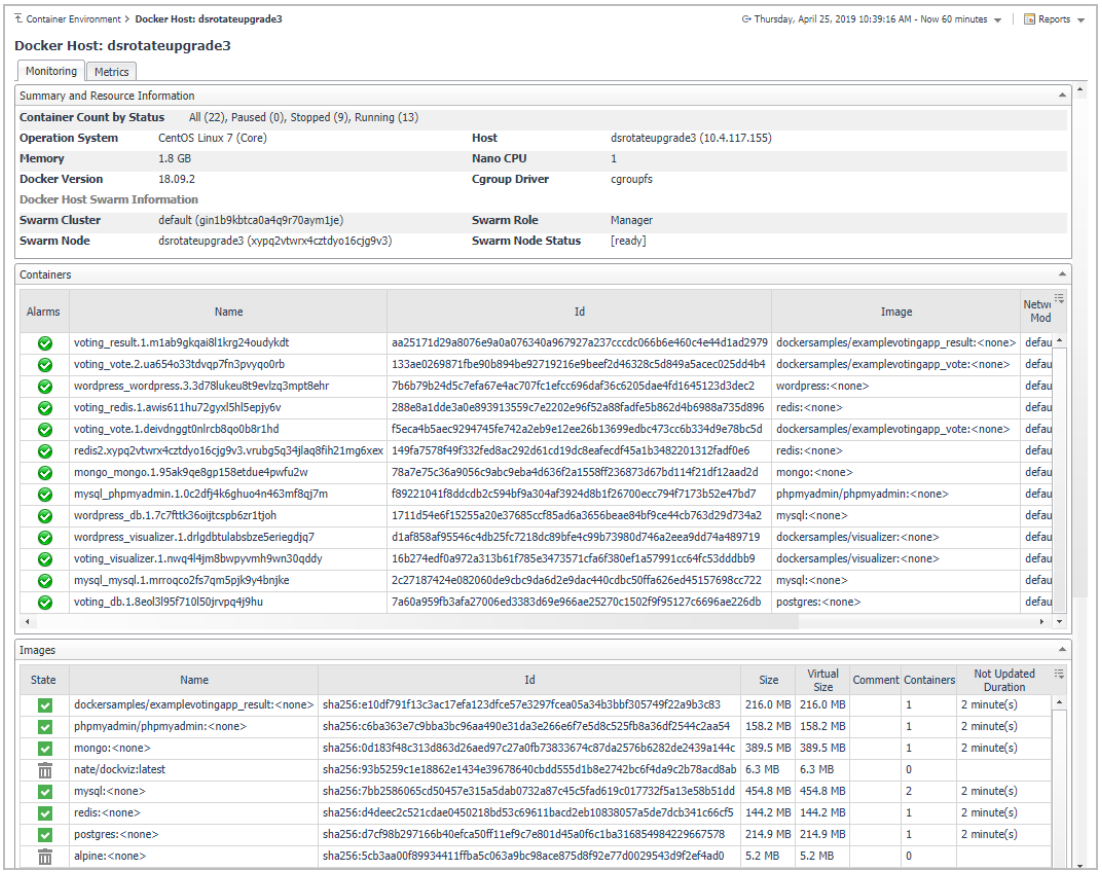
Docker Host Explorer view

The *Docker Host Explorer* view opens when you click **Explore** in the Docker Host Summary view, which includes the following tabs:

- *Monitoring* tab*:* The *Monitoring* tab displays the overall information of the selected docker host over a selected period of time, including the *Summary and Resource Information* table, *Containers* table, *Images* table, and *Volumes* table.

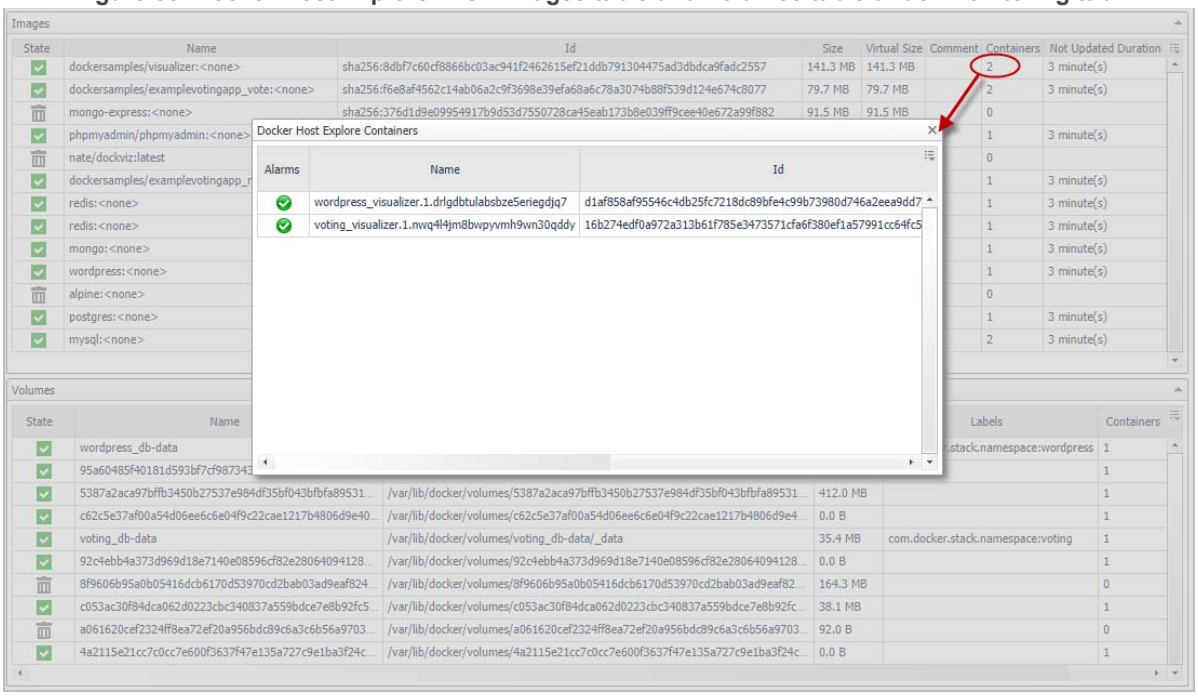
  NOTE: All the docker host metrics are calculated from the aggregated metrics of the containing containers on the docker host.

Figure 32. Docker Host Explorer view Monitoring Tab



  - *Containers* table: Includes the containers on this docker host.
  - *Images* table: Includes the images pulled onto this docker host.
    - : Indicates this image is using by a container.
    - : Indicates no container is using this image and the image can be recycled.
  - *Volumes* table: Includes the volumes created on this docker host.
    - : Indicates this volume is using by a container.
    - : Indicates no container is using this volume and the volume can be recycled.

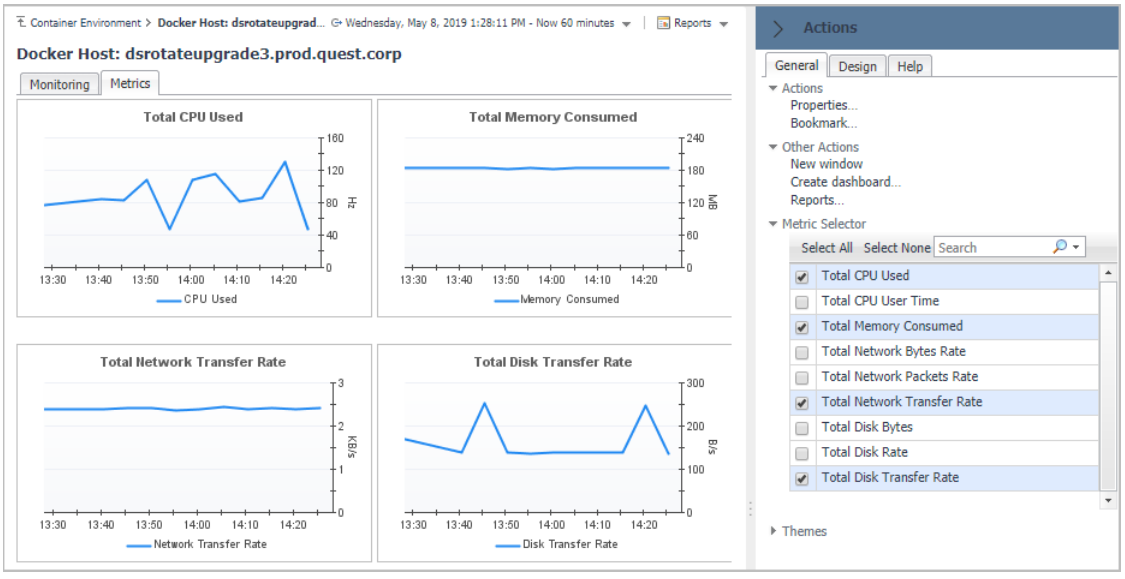
Figure 33. Docker Host Explorer view Images table and Volumes table under Monitoring tab



By clicking the number in the *Containers* column, a *Docker Host Explore Containers* view will open, which lists the containers using this image or this volume. Click the Name or ID of the container and an explore page of the container will appear.

- *Metrics* tab: The *Metrics* tab displays the Metrics list. To set the Metrics list displayed, go to **Action** > **General** > **Metric Selector**. For more information about the description of the metrics, see Container metrics on page 61.

Figure 34. Docker Host Explorer view Metrics Tab



Monitoring Docker Swarm Clusters

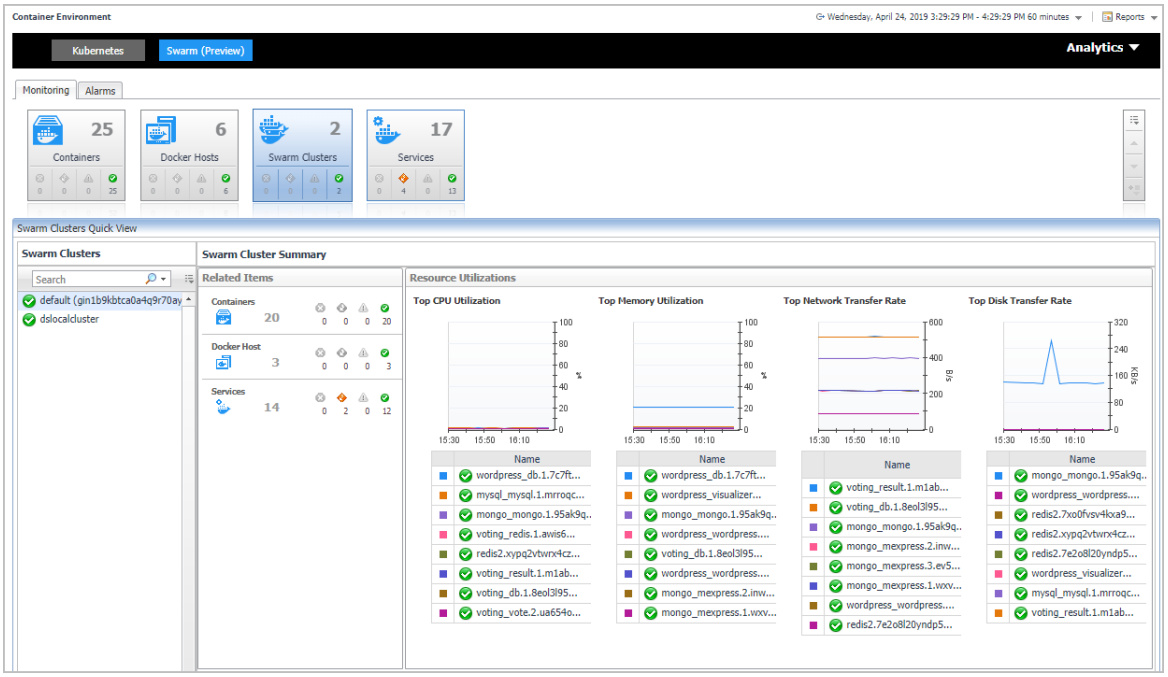
This view consists of the following two panes:

- The **Swarm Clusters** tree view, which appears on the left of *Swarm Clusters Quick View*, lists the docker swarm clusters existing in the monitored *Docker* environment.
- The Docker Swarm Cluster Summary view, which appears on the right after you select an individual docker swarm cluster in the **Swarm Clusters** tree view.

Docker Swarm Cluster Summary view

The **Docker Swarm Cluster Summary** view appears on the right when you select a docker swarm cluster in the **Swarm Clusters** tree view.

Figure 35. Docker Swarm Cluster Summary view



The **Docker Swarm Cluster Summary** view displays the following data:

- *Related Items*: Shows the related Docker components grouped by type as well as the associated alarms.
- *Resource Utilizations*: Shows CPU Utilization, Memory Utilization, Network Transfer Rate, Disk Transfer Rate metrics of the containers running in this docker swarm cluster in descending order.

Monitoring Docker Swarm Services

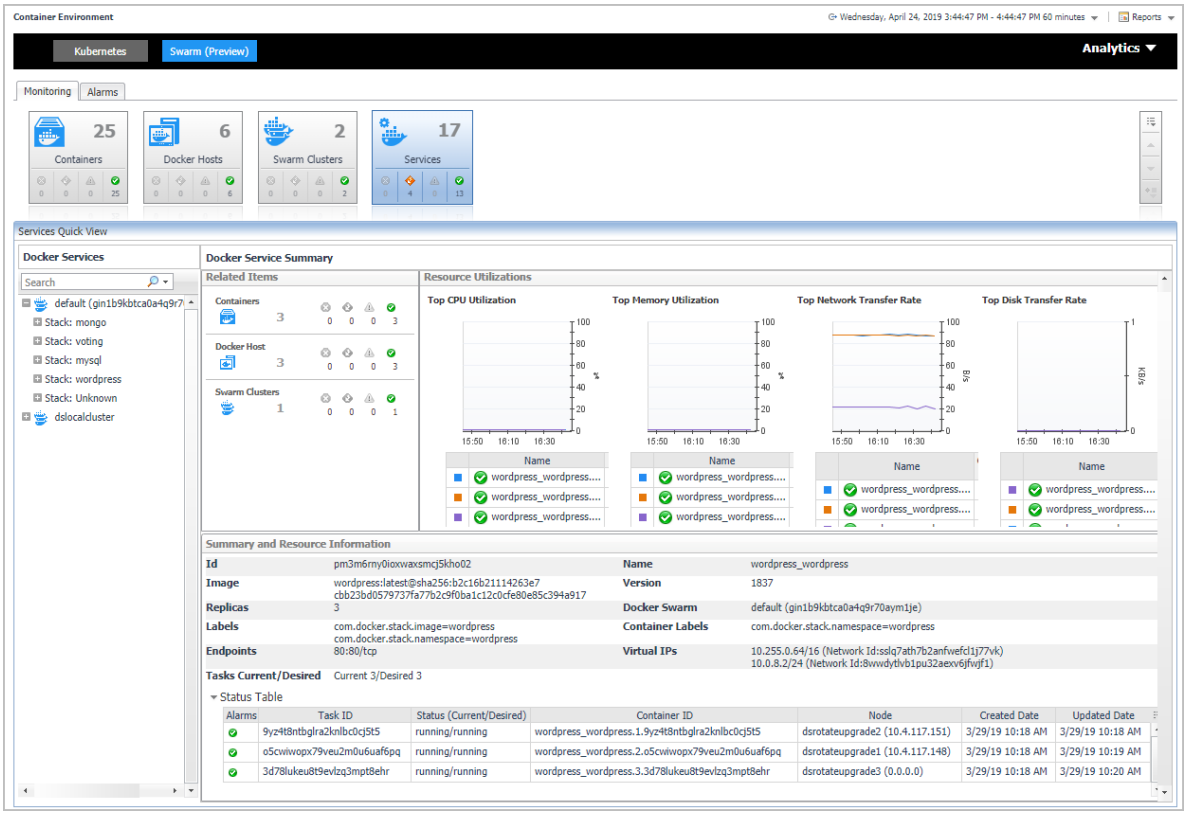
This view consists of the following two panes:

- The **Swarm Services** tree view, which appears on the left of *Swarm Services Quick View*, lists the docker swarm services existing in the monitored *Docker* environment.
- The Docker Swarm Service Summary view, which appears on the right after you select an individual docker swarm service in the **Swarm Services** tree view.

Docker Swarm Service Summary view

The **Docker Service Summary** view appears on the right when you select a docker swarm service in the **Docker Services** tree view.

Figure 36. Docker Service Summary view

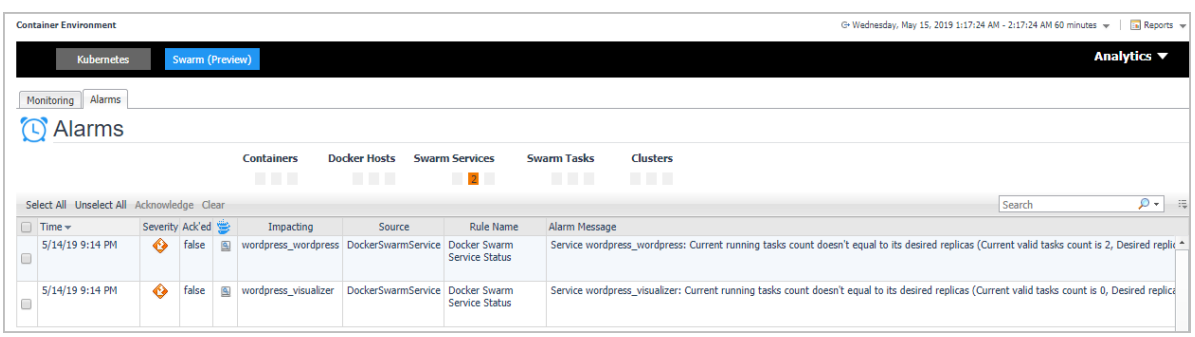


The **Docker Service Summary** view displays the following data:

- *Related Items*: Shows the related Docker components grouped by type as well as the associated alarms.
- *Resource Utilizations*: Shows CPU Utilization, Memory Utilization, Network Transfer Rate, Disk Transfer Rate metrics of the containers running in this docker swarm service in descending order.
- *Summary and Resource Information*: Shows the summary information of the docker swarm service, including Labels, Image, Mount Volumes, Ports, Container Status and so on.

Alarms

Figure 37. Docker Swarm Alarms Dashboard



The *Alarms* dashboard displays a list of alarms generated against the monitored Docker environment. Use this view to quickly identify any potential problems related to a specific Docker component.

Analytics

Foglight for Container Management provide analytics feature for Kubernetes and Docker Swarm.

Heat Map is a two-dimensional representation of data in which values are represented by colors. Showing collected metrics with elaborate heat maps allows you to understand complex data sets and the monitored cluster environment well.

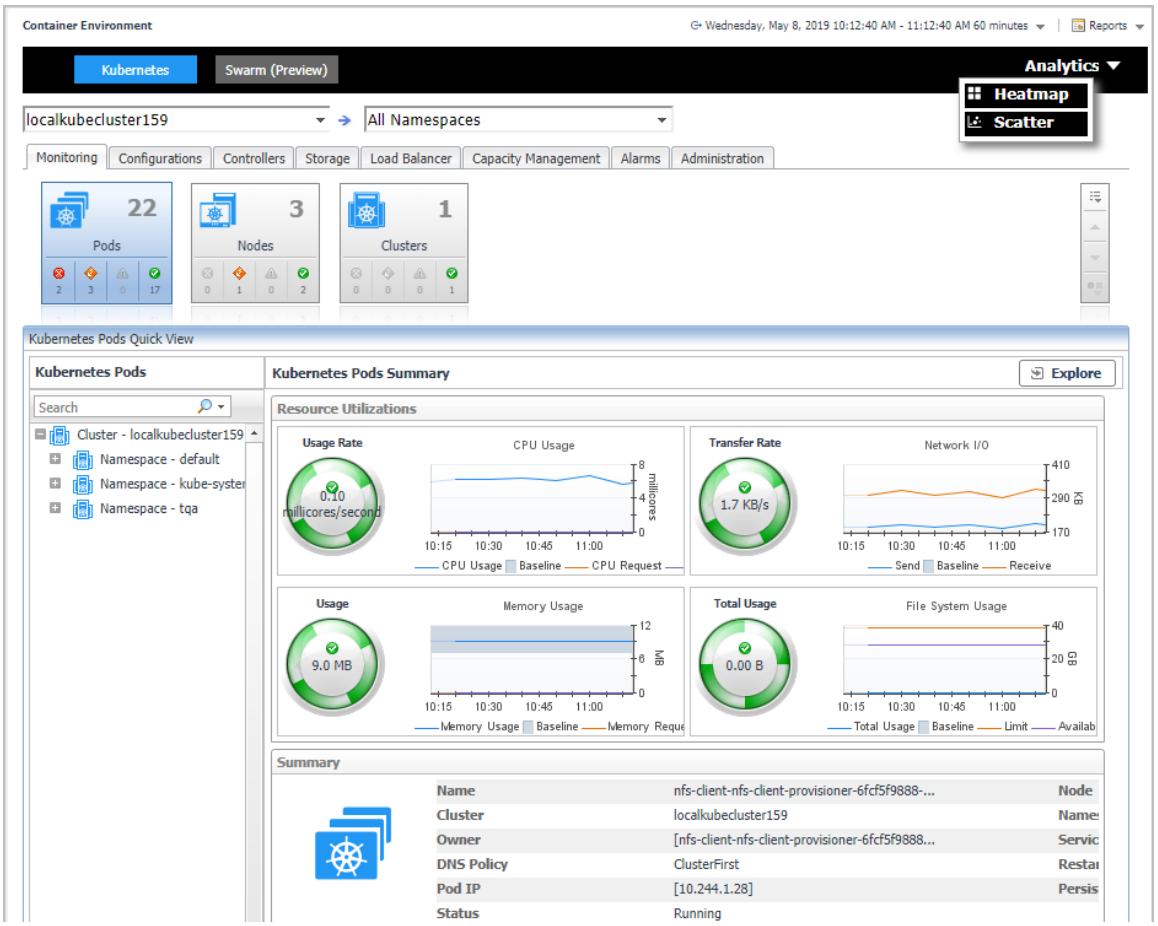
Scatter Plot is used to display values in points using two variables for a set of data. The points is color-coded also, Color Metric can be used to display one additional variable.

- Kubernetes analytics
  - Heatmap analytics
  - Scatter Plot analytics
- Docker Swarm analytics
  - Heatmap analytics
  - Scatter Plot analytics

Kubernetes analytics

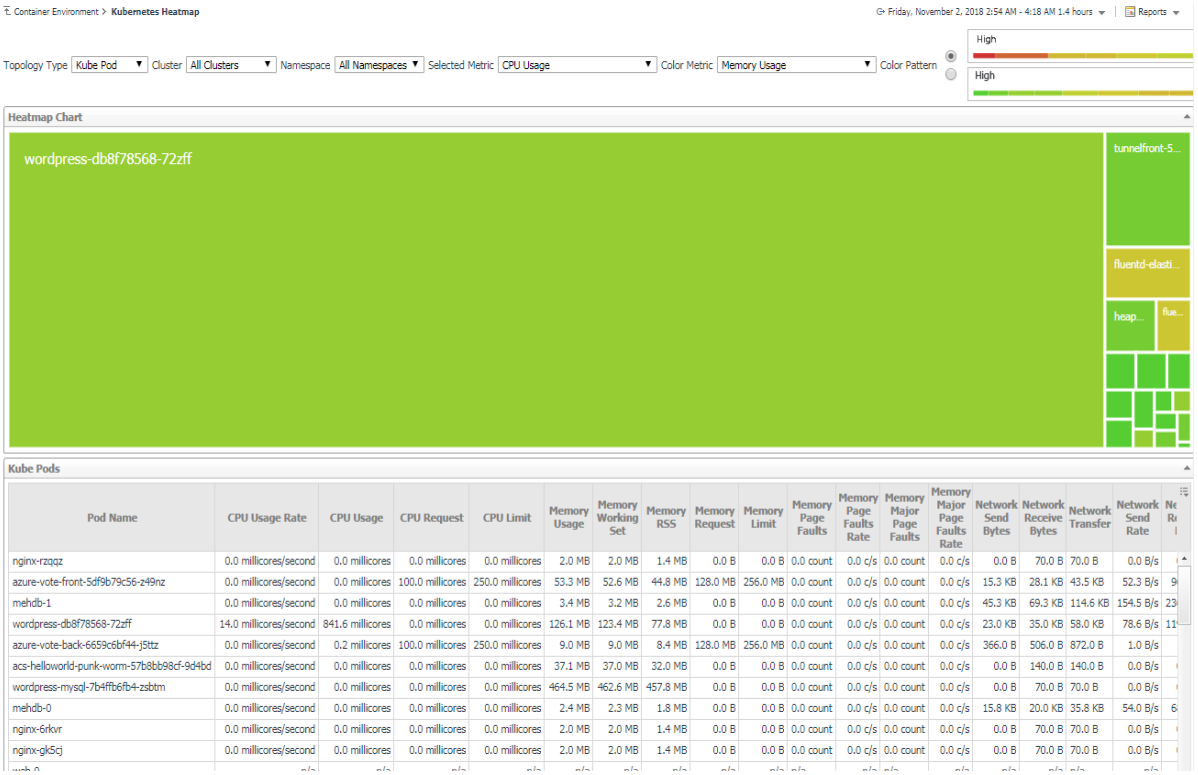
In the Container dashboard, choose **Kubernetes** from the header. Then click **Analytics** from the header, a drop down view will display with **Heatmap** and **Scatter** on it. Click **Heatmap** will navigate to the Kubernetes **Heatmap Analytics** dashboard, while click **Scatter** will navigate to the Kubernetes **Scatter Plot Analytics** dashboard.

Figure 38. Kubernetes analytics Navigation



Heatmap analytics

Figure 39. Kubernetes Heatmap Analytics Dashboard



Pod Name	CPU Usage Rate	CPU Usage	CPU Request	CPU Limit	Memory Usage	Memory Working Set	Memory RSS	Memory Request	Memory Limit	Memory Page Faults	Memory Page Faults Rate	Memory Major Page Faults	Memory Major Page Faults Rate	Network Send Bytes	Network Receive Bytes	Network Transfer	Network Send Rate
nginx-rzqqz	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	2.0 MB	2.0 MB	1.4 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	0.0 B	70.0 B	70.0 B	0.0 B/s
azure-vote-front-5df9b79c56-z49nz	0.0 millicores/second	0.0 millicores	100.0 millicores	250.0 millicores	53.3 MB	52.6 MB	44.8 MB	128.0 MB	256.0 MB	0.0 count	0.0 c/s	0.0 count	0.0 c/s	15.3 KB	28.1 KB	43.5 KB	52.3 B/s
mehdb-1	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	3.4 MB	3.2 MB	2.6 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	45.3 KB	69.3 KB	114.6 KB	154.5 B/s
wordpress-db8f78568-72zff	14.0 millicores/second	841.6 millicores	0.0 millicores	0.0 millicores	126.1 MB	123.4 MB	77.8 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	23.0 KB	35.0 KB	58.0 KB	78.6 B/s
azure-vote-back-6659c6bf44-j5ttz	0.0 millicores/second	0.2 millicores	100.0 millicores	250.0 millicores	9.0 MB	9.0 MB	8.4 MB	128.0 MB	256.0 MB	0.0 count	0.0 c/s	0.0 count	0.0 c/s	366.0 B	506.0 B	872.0 B	1.0 B/s
acs-helloworld-punk-worm-57b8bb98cf-9d4bd	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	37.1 MB	37.0 MB	32.0 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	0.0 B	140.0 B	140.0 B	0.0 B/s
wordpress-mysql-7b4ffb6fb4-zsbtm	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	464.5 MB	462.6 MB	457.8 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	0.0 B	70.0 B	70.0 B	0.0 B/s
mehdb-0	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	2.4 MB	2.3 MB	1.8 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	15.8 KB	20.0 KB	35.8 KB	54.0 B/s
nginx-6rkvr	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	2.0 MB	2.0 MB	1.4 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	0.0 B	70.0 B	70.0 B	0.0 B/s
nginx-gk5cj	0.0 millicores/second	0.0 millicores	0.0 millicores	0.0 millicores	2.0 MB	2.0 MB	1.4 MB	0.0 B	0.0 B	0.0 count	0.0 c/s	0.0 count	0.0 c/s	0.0 B	70.0 B	70.0 B	0.0 B/s

Heat maps will be refreshed automatically when you change either of the following fields:

- *Topology Type:* Indicates the monitored topology object, including Kubernetes Pod, Kubernetes Node, and Kubernetes Cluster.
- *Cluster:* Lists all clusters available in the monitored Kubernetes environment.
- *Namespace:* Lists all namespaces available in the monitored Kubernetes environment.
- *Selected Metric:* Populates a rectangle based upon the selected metrics. For example, if you select *Memory Usage* from the *Selected Metric* drop-down list, the rectangle area will be populated based on the used memory for the selected topology object. For more information about metrics, refer to Kubernetes metrics *on page 60*.
- Rendering related metrics: For example, if you select *CPU Usage Rate* and Red to Green, the rectangle of the topology object that has larger value of CPU Usage Rate will be rendered in red.
  - *Color Metric*: Renders the color of rectangle based upon the selected color metric.
  - *Color Pattern*: Offers two patterns, Red to Green (larger value shows in red) or Green to Red (larger value shows in green).

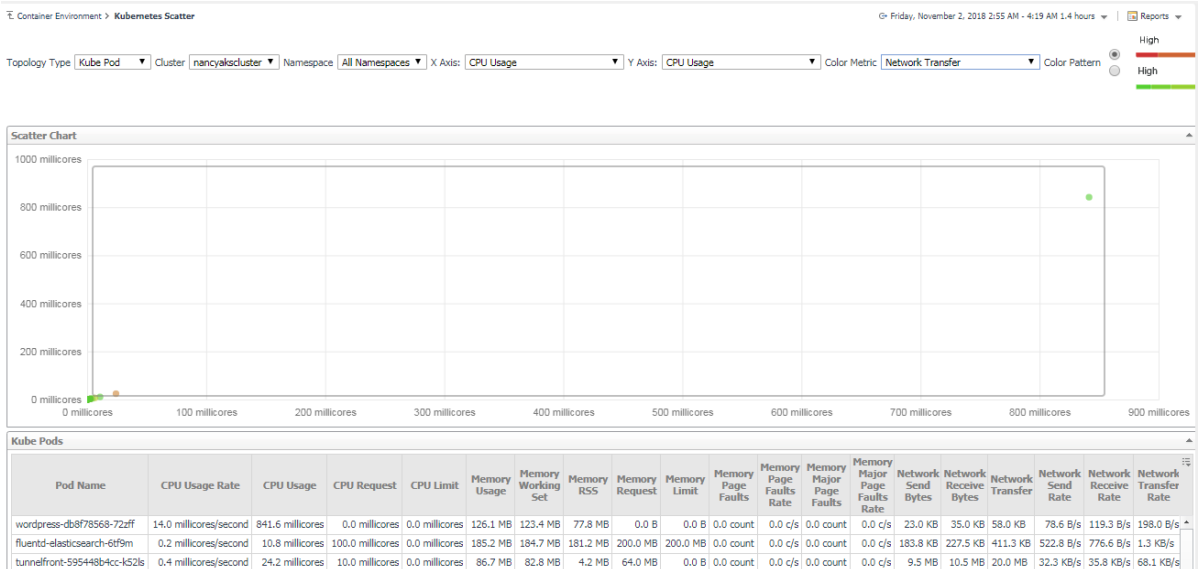
Figure 39 shows an example of heat map. This sample diagram represents the “wordpress-db8f78568-72zff” has the maximum amounts of CPU usage, while “fluentd-elastic-ef455uh68-72cfe” has a higher Memory Usage. If you switch the Color Pattern, then “wordpress-db8f78568-72zff” will turn to red. Clicking the object name on the heat map directs you to the relevant object *Explorer* dashboard. For more information, see:

- Pods Explorer view on page 27
- Pod metrics on page 60
- Nodes Explorer view on page 30
- Node metrics on page 61
- Cluster Explorer view on page 33

- Cluster metrics on page 61

Scatter Plot analytics

Figure 40. Kubernetes Scatter Plot Analytics Dashboard



The points on the chart will be refreshed automatically when you change either of the following fields:

- *Topology Type:* Indicates the monitored topology object, including Kubernetes Pod, Kubernetes Node, and Kubernetes Cluster.
- *Cluster:* Lists all clusters available in the monitored Kubernetes environment.
- *Namespace:* Lists all namespaces available in the monitored Kubernetes environment.
- *X Axis:* Indicates which metrics will be plotted on X axis.
- *Y Axis:* Indicates which metrics will be plotted on Y axis.
- Rendering related metrics:
  - Color Metric: Renders the color of circle based upon the selected metrics.
  - *Color Pattern*: Offers two patterns, Red to Green (larger value shows in red) or Green to Red (larger value shows in green).

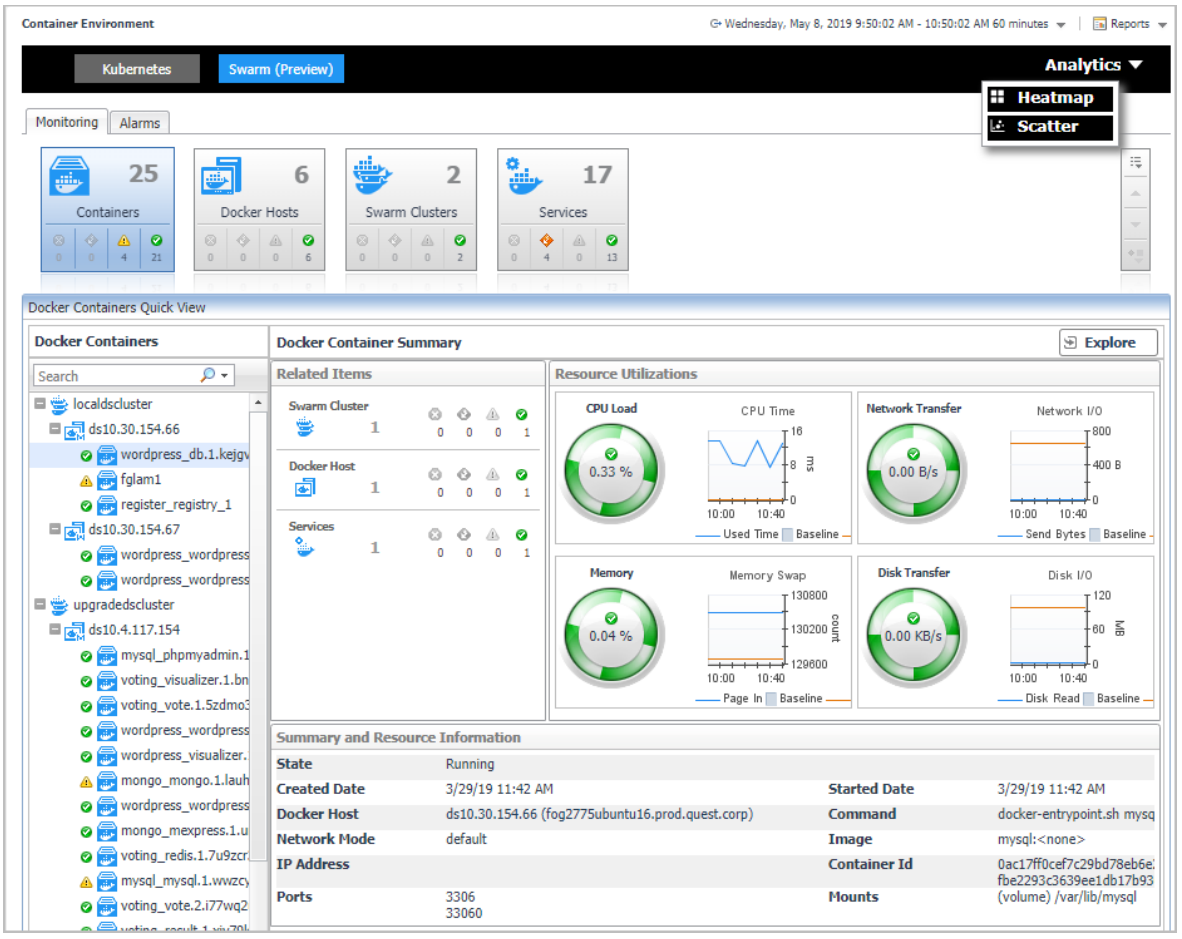
Figure 40 shows an example of Scatter Plot analytics. The purple circle in the middle represents the following: “wordpress-db8f78568-72zff” CPU Usage is around 0.85 cores, its Memory Usage is around 121MB, and its value of Network Transfer Bytes is not high. For more information, see:

- Pods Explorer view on page 27
- Pod metrics on page 60
- Nodes Explorer view on page 30
- Node metrics on page 61
- Cluster Explorer view on page 33
- Cluster metrics on page 61

Docker Swarm analytics

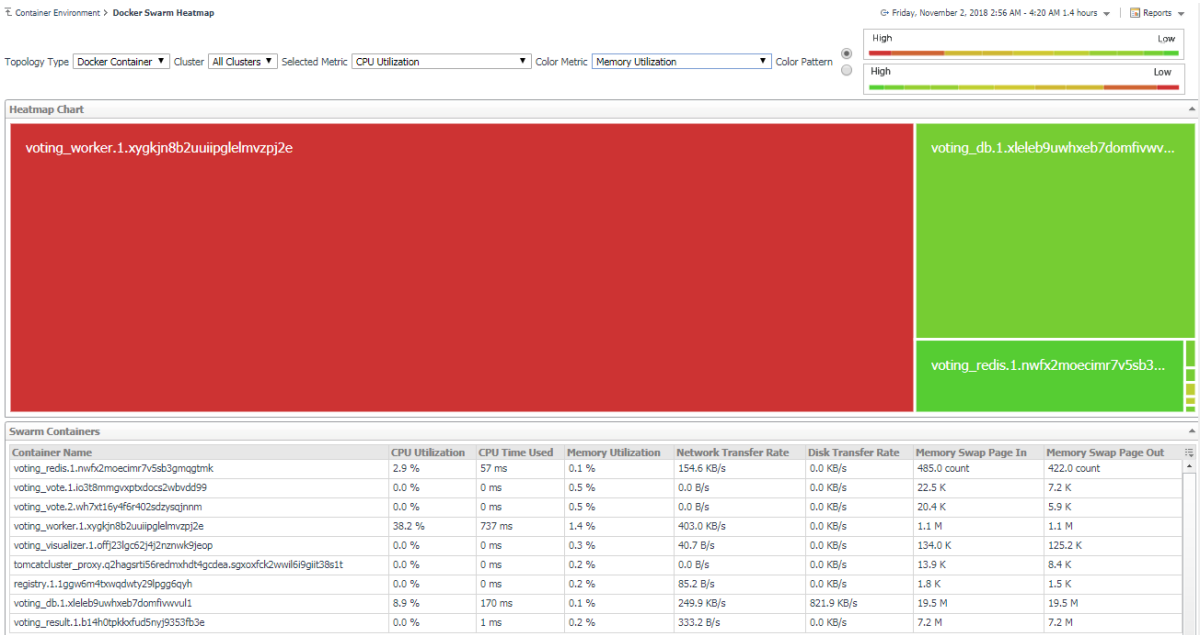
In the Container dashboard, choose **Docker Swarm** from the header. Then click **Analytics** from the header, a drop down view will display with **Heatmap** and **Scatter** on it. Click **Heatmap** will navigate to the Docker Swarm **Heatmap Analytics** dashboard, while click **Scatter** will navigate to the Docker Swarm **Scatter Plot Analytics** dashboard.

Figure 41. Docker Swarm Analytics Navigation



Heatmap analytics

Figure 42. Docker Swarm Heatmap Analytics Dashboard



Container Name	CPU Utilization	CPU Time Used	Memory Utilization	Network Transfer Rate	Disk Transfer Rate	Memory Swap Page In	Memory Swap Page Out
voting_redis.1.nwfx2moecimr7v5sb3gmqgtmk	2.9 %	57 ms	0.1 %	154.6 KB/s	0.0 KB/s	485.0 count	422.0 count
voting_vote.1.io3t8mmgvxptxdocs2wbvdd99	0.0 %	0 ms	0.5 %	0.0 B/s	0.0 KB/s	22.5 K	7.2 K
voting_vote.2.wh7xt16y4f6r402sdzysqjnnm	0.0 %	0 ms	0.5 %	0.0 B/s	0.0 KB/s	20.4 K	5.9 K
voting_worker.1.xygkjn8b2uuiipglelmvzpj2e	38.2 %	737 ms	1.4 %	403.0 KB/s	0.0 KB/s	1.1 M	1.1 M
voting_visualizer.1.offj23lgc62j4j2nznwk9jeop	0.0 %	0 ms	0.3 %	40.7 B/s	0.0 KB/s	134.0 K	125.2 K
tomcatcluster_proxy.q2hagsrti56redmxhdt4gcdea.sgxoxfck2wwil6i9giit38s1t	0.0 %	0 ms	0.2 %	0.0 B/s	0.0 KB/s	13.9 K	8.4 K
registry.1.1ggw6m4txwqdwty29lpgg6qyh	0.0 %	0 ms	0.2 %	85.2 B/s	0.0 KB/s	1.8 K	1.5 K
voting_db.1.xleleb9uwhxeb7domfivwvul1	8.9 %	170 ms	0.1 %	249.9 KB/s	821.9 KB/s	19.5 M	19.5 M
voting_result.1.b14h0tpkkxfud5nyj9353fb3e	0.0 %	1 ms	0.2 %	333.2 B/s	0.0 KB/s	7.2 M	7.2 M

Heat maps will be refreshed automatically when you change either of the following fields:

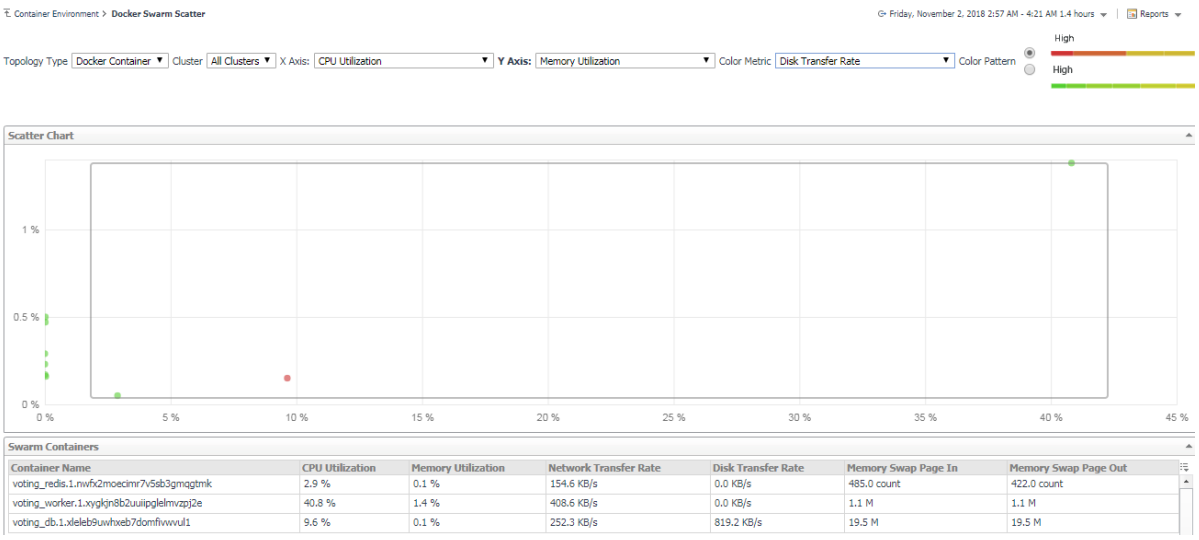
- *Topology Type:* Indicates the monitored topology object, including Docker Container and Docker Host.
- *Cluster:* Lists all clusters available in the monitored Docker Swarm environment.
- *Selected Metric:* Populates a rectangle based upon the selected metrics. For example, if you select *Memory Time Used* from the *Selected Metric* drop-down list, the rectangle area will be populated based on the used CPU time for the selected topology object. For more information about metrics, refer to Docker Swarm metrics *on page 61*.
- Rendering related metrics: For example, if you select *CPU Utilization* and Red to Green, the rectangle of the topology object that has larger value of CPU Utilization will be rendered in red.
  - *Color Metric*: Renders the color of rectangle based upon the selected color metric.
  - *Color Pattern*: Offers two patterns, Red to Green (larger value shows in red) or Green to Red (larger value shows in green).

Figure 42 shows an example of heat map. This sample diagram represents the “voting_redis.1.nwfx2moecimr7v5sb3gmqgtmk” has the maximum amounts of CPU Utilization which is the largest in size, and also it has the higher Memory Utilization since it is in Red. If you switch the Color Pattern, then “voting_redis.1.nwfx2moecimr7v5sb3gmqgtmk” will turn to green. Clicking the object name on the heat map directs you to the relevant object *Explorer* dashboard. For more information, see:

- Container Explorer view on page 46
- Docker Host Explorer view on page 47
- Container metrics on page 61

Scatter Plot analytics

Figure 43. Docker Swarm Scatter Plot Analytics Dashboard



The points on the chart will be refreshed automatically when you change either of the following fields:

- *Topology Type:* Indicates the monitored topology object, including Docker Container and Docker Host.
- *Cluster:* Lists all clusters available in the monitored Docker Swarm environment.
- *X Axis:* Indicates which metrics will be plotted on X axis.
- *Y Axis:* Indicates which metrics will be plotted on Y axis.
- Rendering related metrics:
  - Color Metric: Renders the color of circle based upon the selected metrics.
  - *Color Pattern*: Offers two patterns, Red to Green (larger value shows in red) or Green to Red (larger value shows in green).

Figure 43 shows an example of Scatter Plot analytics. The purple circle in the middle represents the following: “voting_redis.1.nwfx2moecimr7v5sb3gmqgtmk” CPU Utilization is 2.9%, its Memory Usage is 0.1%, and its value of Network Transfer Bytes is not high. For more information, see:

- Container Explorer view on page 46
- Docker Host Explorer view on page 47
- Container metrics on page 61

Domains and Object Groups

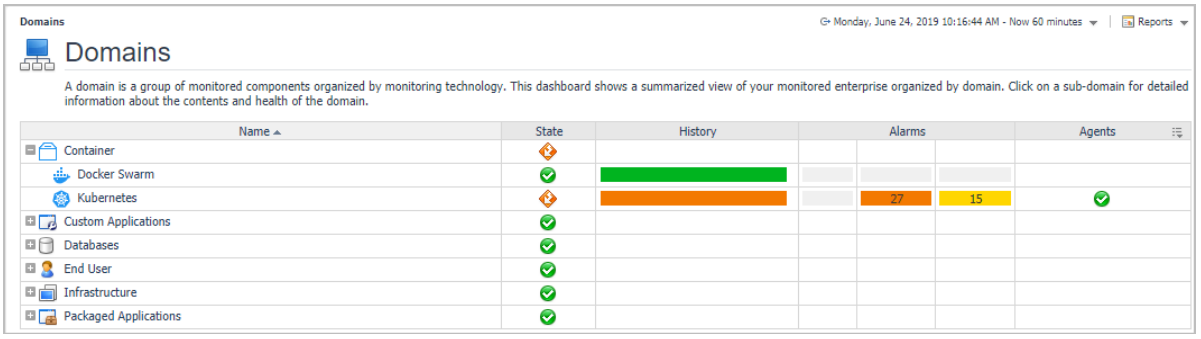
Domains

A domain is a group of monitored components organized by monitoring technology. This dashboard shows a summarized view of your monitored enterprise organized by domain. Click on a sub-domain for detailed information about the contents and health of the domain.

To access the Domains dashboard, on the Navigation panel, click **Dashboards** > **Services** > **Domains**.

Click the + icon to display the components under Container.

Figure 44. Container Components in Domains dashboard



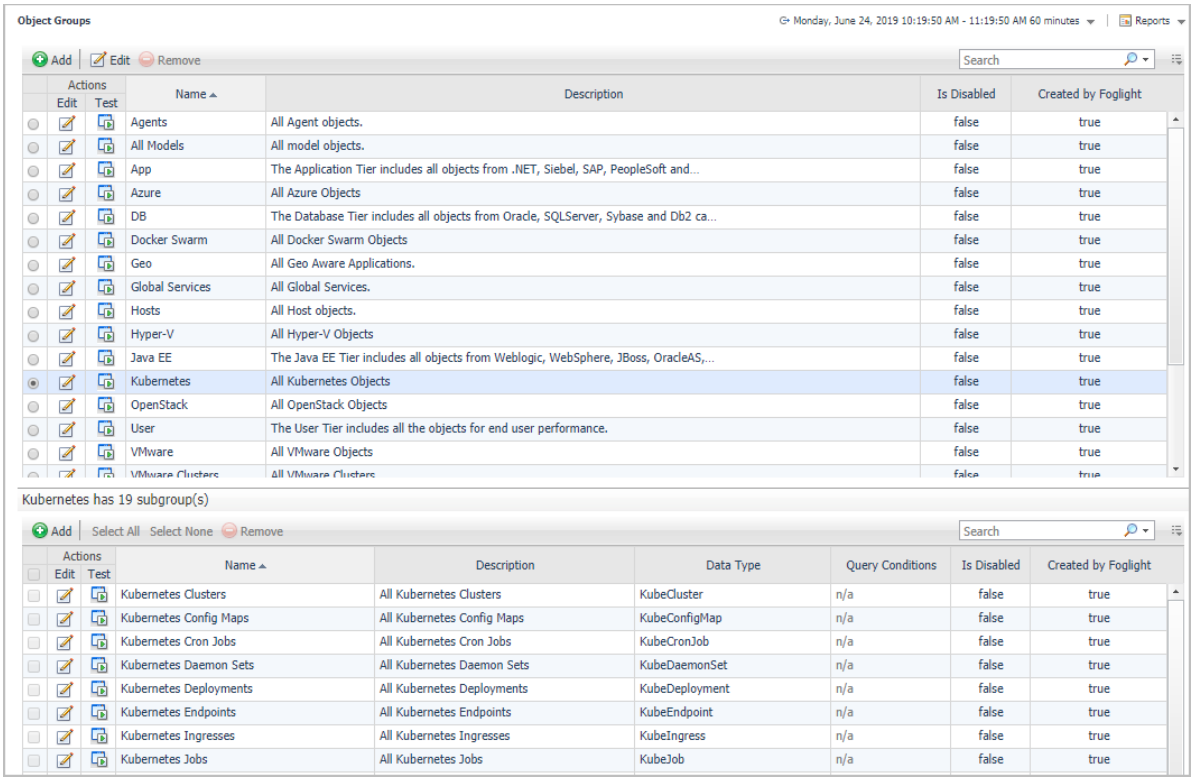
Click the State, History, Alarms, and Agents column, for detailed alarms and health information.

Object Groups

An object group is a mapping to a certain set of data types of the objects you are interested in.

To access the Object Groups dashboard, on the Navigation panel, click **Dashboards** > **Services** > **Object Groups**.

Figure 45. Object Groups for Container



Select *Docker Swarm* or *Kubernetes* to display the subgroups.

Metrics

Kubernetes metrics

Pod metrics

Table 4. Pod metrics

Metric name	Description
CPU Usage	CPU usage on all cores in millicores.
CPU Usage Rate	CPU usage rate on all cores in millicores/second.
CPU Request	CPU request (the guaranteed amount of resources) in millicores.
CPU Limit	CPU hard limit in millicores.
CPU Utilization	Percentage of CPU usage / CPU limit if user configured CPU limit for this pod.
Memory Usage	Total memory usage in bytes.
Memory Working Set	Total working set usage. Working set is the memory being used and not easily dropped by the kernel.
Memory Rss	RSS memory usage.
Memory Request	Memory request (the guaranteed amount of resources) in bytes.
Memory Limit	Memory hard limit in bytes.
Memory Page Faults	Number of page faults.
Memory Major Page Faults	Number of major page faults.
Memory Page Faults Rate	Number of page faults per second.
Memory Major Page Faults Rate	Number of major page faults per second.
Memory Utilization	Percentage of Memory usage / Memory limit if user configured Memory limit for this pod.
Network Send	Total send bytes.
Network Receive	Total receive bytes.
Network Send Rate	Total send bytes per second.
Network Receive Rate	Total receive bytes per second.
Network Send Errors	Total send errors count.
Network Receive Errors	Total receive errors count.
Network Send Errors Rate	Total send errors count per second.
Network Receive Errors Rate	Total receive errors count per second.
Network Transfer	Total send and receive bytes.
Network Transfer Rate	Total send and receive bytes per second.

Node metrics

Table 5. Node metrics

Metric name	Description
CPU Usage	CPU usage on all cores in millicores.
CPU Usage Rate	CPU usage rate on all cores in millicores/second.
CPU Request	CPU request (the guaranteed amount of resources) in millicores.
CPU Limit	CPU hard limit in millicores.
CPU Utilization	CPU utilization as a share of node allocatable.
Memory Usage	Total memory usage in bytes.
Memory Working Set	Total working set usage. Working set is the memory being used and not easily dropped by the kernel.
Memory RSS	RSS memory usage.
Memory Request	Memory request (the guaranteed amount of resources) in bytes.
Memory Limit	Memory hard limit in bytes.
Memory Page Faults	Number of page faults.
Memory Major Page Faults	Number of major page faults.
Memory Page Faults Rate	Number of page faults per second.
Memory Major Page Faults Rate	Number of major page faults per second.
Memory Utilization	Memory utilization as a share of memory allocatable.

Cluster metrics

Table 6. Cluster metrics

Metric name	Description
CPU Usage	CPU usage on all cores in millicores.
CPU Usage Rate	CPU usage rate on all cores in millicores/second.
CPU Request	CPU request (the guaranteed amount of resources) in millicores.
CPU Limit	CPU hard limit in millicores.
Memory Usage	Total memory usage in bytes.
Memory Request	Memory request (the guaranteed amount of resources) in bytes.
Memory Limit	Memory hard limit in bytes.

Docker Swarm metrics

Container metrics

Table 7. Container metrics

Metric name	Description
CPU Utilization	CPU utilization.
CPU Time Used	Total CPU time that a container used.
CPU Throttled Time	Total time that a container's CPU usage was throttled.
Memory Page Fault	Total page fault count of a container's Memory.
Memory Consumed	Total memory consumed of a container in bytes.

Table 7. Container metrics

Metric name	Description
Memory Utilization	Memory utilization.
Memory PageIn Rate	Total page in count of a container's Memory.
Memory PageOut Rate	Total page out count of a container's Memory.
Disk Read Bytes	Total disk read bytes.
Disk Write Bytes	Total disk write bytes.
Disk Transfer Rate	Sum of total disk read and write bytes.
Network Send Packets	Total network send packets count.
Network Receive Packets	Total network receive packets count.
Network Send Bytes	Total network send bytes.
Network Receive Bytes	Total network receive bytes.
Network Inbound Dropped Packets	Total dropped packet count of all the packets coming into the container.
Network Outbound Dropped Packets	Total dropped packet count of all the packets going out from the container.
Network Transfer Rate	Sum of network send bytes and receive bytes per seconds during a specific period.

Rules

Foglight for Container Management allows you to create flexible rules that can be applied to complex interrelated data from multiple sources within your clusters. You can associate several different actions with a rule, configure a rule so that it does not fire repeatedly, and associate a rule with schedules to define when it should be evaluated or not.

Different types of data can be used in rules, including registry variables, raw metrics, derived metrics, and topology object properties.

There are two types of rules: simple rules and multiple-severity rules. A simple rule has a single condition, and can be in one of three states: *Fire*, *Undefined*, or *Normal*. A multiple-severity rule can have up to five severity levels: *Undefined*, *Fatal*, *Critical*, *Warning*, and *Normal*.

Rule conditions are regularly evaluated against monitoring data (metrics and topology object properties collected from your monitored environment and transformed into a standard format). Therefore, the state of the rule can change if the data changes. For example, if a set of monitoring data matches a simple rule's condition, the rule enters the *Fire* state. If the next set does not match the condition, the rule exits the *Fire* state and enters the *Normal* state.

Rules can be configured to send emails, pager messages, or perform other actions you define. Performance data can be viewed and analyzed using Foglight for Container Management.

Foglight for Container Management includes a number of predefined rules used to monitor the health of your container clusters. You are allowed to modify these rules to satisfy your different requirements. Many of these rules listed and described in this section have thresholds defined within them. Those thresholds include standard deviations, utilization percentages, and so on, are default values predefined in the registry.

Kubernetes

All rules are controlled by registry variable Kubernetes:AlertSensitivity. If the value is 0, then no alarm can be fired. If the value is 1, warning level alarm can be fired. If the value is above 1, then all level alarm can be fired.

Kubernetes Administrator email address can be configured in Registry Variable KubernetesAdmin.

Health Check

Kubernetes Pod Health Check

Purpose

This rule detects abnormal Pod health status and fires alarm for different severity abnormal health status.

Scope

KubePod

Conditions and Severities

Conditions	Severity	Action
Pods that is in Failed or Unknown status. Or the node which is running the pod gets disconnected.	Critical	Send email to Kubernetes Administrator.
Pods that is in CrashLoopBackOff status.	Warning	None

Kubernetes Pod Health Check (Pending Phase)

Purpose

This rule detects Pods that stays in pending phase for an abnormal long time.

Scope

KubePod

Conditions and Severities

Conditions	Severity	Action
Pods that is pending for two continuous data submission periods because of Failed to schedule to Node.	Critical	Send email to Kubernetes Administrator.
Pods that is pending for two continuous data submission periods because container is not ready.	Warning	None

Kubernetes Container Health Check

Purpose

This rule detects abnormal Container health status and fires alarm for different severity abnormal health status.

Scope

KubeContainer

Conditions and Severities

Conditions	Severity	Action
Container that is terminated for abnormal reasons.	Critical	Send email to Kubernetes Administrator.

Kubernetes Node Health Check

Purpose

This rule detects abnormal Node health status and fires alarm for different severity abnormal health status.

Scope

KubeNode

Conditions and Severities

Conditions	Severity	Action
Nodes that is not Ready or out of disk or network unavailable.	Critical	Send email to Kubernetes Administrator.
Nodes whose memory or disk is under pressure.	Warning	None

Kubernetes Deployment Health Check

Purpose

This rule detects abnormal Deployment health status and fires alarm for different severity abnormal health status.

Scope

KubeDeployment

Conditions and Severities

Conditions	Severity	Action
Deployment is not available.	Critical	Send email to Kubernetes Administrator.
Deployment has failed to create some of the replicated pods.	Warning	None

Kubernetes Daemon Set Health Check

Purpose

This rule detects abnormal Daemon Set health status and fires alarm for different severity abnormal health status.

Scope

KubeDaemonSet

Conditions and Severities

Conditions	Severity	Action
Some of the pods created by the Daemon Set is not available or mis-scheduled.	Critical	Send email to Kubernetes Administrator.
The daemon set doesn’t have enough replicated pods running that meets its desired replicated pods count.	Warning	None

Kubernetes Job Health Check

Purpose

This rule detects abnormal Job health status and fires alarm for different severity abnormal health status.

Scope

KubeJob

Conditions and Severities

Conditions	Severity	Action
Job that is failed.	Warning	None

Kubernetes Persistent Volume Health Check

Purpose

This rule detects abnormal Persistent Volume health status and fires alarm for different severity abnormal health status.

Scope

KubePersistentVolume

Conditions and Severities

Conditions	Severity	Action
Persistent Volume that is in failed status.	Warning	None

Kubernetes Persistent Volume Claim Health Check

Purpose

This rule detects abnormal Persistent Volume Claim health status and fires alarm for different severity abnormal health status.

Scope

KubePersistentVolumeClaim

Conditions and Severities

Conditions	Severity	Action
Persistent Volume Claim that is in failed status.	Warning	None

Kubernetes Persistent Volume Claim Health Check (Long Pending)

Purpose

This rule detects abnormal long pending Persistent Volume Claim and fires alarm for different severities.

Scope

KubePersistentVolumeClaim

Conditions and Severities

Conditions	Severity	Action
Persistent Volume Claim that is pending for two continuous data submission periods.	Critical	None

Usage

Kubernetes Pod CPU Utilization

Purpose

This rule detects abnormal CPU Utilization for Pods, and fires alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities. This rule only works for those Pods that configures CPU limit.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is about to reach the limit, the ration is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodCpu UtilizationFatal	Send email to Kubernetes Administrator
Pods whose usage is about to reach the limit, the ration is above the value configured in critical Threshold.	Critical	Kubernetes:PodCpu UtilizationCritical	None
Pods whose usage is about to reach the limit, the ration is above the value configured in warning Threshold.	Warning	Kubernetes:PodCpu UtilizationWarning	None

*Note: the unit is percentage.

Kubernetes Pod Memory Utilization

Purpose

This rule detects abnormal Memory Utilization for Pods, and fires alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities. This rule only works for those Pods that configures Memory limit.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is about to reach the limit, the ration is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodMemory UtilizationFatal	Send email to Kubernetes Administrator
Pods whose usage is about to reach the limit, the ration is above the value configured in critical Threshold.	Critical	Kubernetes:PodMemory UtilizationCritical	None
Pods whose usage is about to reach the limit, the ration is above the value configured in warning Threshold.	Warning	Kubernetes:PodMemory UtilizationWarning	None

*Note: the unit is percentage.

Kubernetes Pod CPU Usage

Purpose

This rule detects abnormal CPU Usage for Pods, and fires alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodCpu UsageFatal	Send email to Kubernetes Administrator
Pods whose usage is above the value configured in critical Threshold.	Critical	Kubernetes:PodCpu UsageCritical	None
Pods whose usage is above the value configured in warning Threshold.	Warning	Kubernetes:PodCpu UsageWarning	None

*Note: the unit is percentage.

Kubernetes Pod Memory Usage

Purpose

This rule detects abnormal Memory Usage for Pods, and fires alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodMemory UsageFatal	Send email to Kubernetes Administrator
Pods whose usage is above the value configured in critical Threshold.	Critical	Kubernetes:PodMemory UsageCritical	None
Pods whose usage is above the value configured in warning Threshold.	Warning	Kubernetes:PodMemory UsageWarning	None

*Note: the unit is percentage.

Kubernetes Pod Network Receive

Purpose

This rule detects abnormal Network Receive in bytes for Pods, and fire alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodNetwork ReceiveFatal	Send email to Kubernetes Administrator
Pods whose usage is above the value configured in critical Threshold.	Critical	Kubernetes:PodNetwork ReceiveCritical	None
Pods whose usage is above the value configured in warning Threshold.	Warning	Kubernetes:PodNetwork ReceiveWarning	None

*Note: the unit is percentage.

Kubernetes Pod Network Send

Purpose

This rule detects abnormal Network Send in bytes for Pods, and fire alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubePod.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Pods whose usage is above the value configured in fatal Threshold.	Fatal	Kubernetes:PodNetwork SendFatal	Send email to Kubernetes Administrator
Pods whose usage is above the value configured in critical Threshold.	Critical	Kubernetes:PodNetwork SendCritical	None
Pods whose usage is above the value configured in warning Threshold.	Warning	Kubernetes:PodNetwork SendWarning	None

*Note: the unit is percentage.

Kubernetes Node CPU Utilization

Purpose

This rule detects abnormal CPU Utilization in bytes for Nodes, and fire alarm on different severities. It is enabled by default. You can change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubeNode.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Nodes whose utilization is above the value configured in fatal Threshold.	Fatal	Kubernetes:NodeCpu UtilizationFatal	Send email to Kubernetes Administrator
Nodes whose utilization is above the value configured in critical Threshold.	Critical	Kubernetes:NodeCpu UtilizationCritical	None
Nodes whose utilization is above the value configured in warning Threshold.	Warning	Kubernetes:NodeCpu UtilizationWarning	None

*Note: the unit is percentage.

Kubernetes Node Memory Utilization

Purpose

This rule detects abnormal Memory Utilization in bytes for Nodes, and fire alarm on different severities. It is enabled by default. You can change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubeNode.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Nodes whose utilization is above the value configured in fatal Threshold.	Fatal	Kubernetes:NodeMemory UtilizationFatal	Send email to Kubernetes Administrator
Nodes whose utilization is above the value configured in critical Threshold.	Critical	Kubernetes:NodeMemory UtilizationCritical	None
Nodes whose utilization is above the value configured in warning Threshold.	Warning	Kubernetes:NodeMemory UtilizationWarning	None

*Note: the unit is percentage.

Kubernetes Node Network Receive

Purpose

This rule detects abnormal Network Receive in bytes for Nodes, and fire alarm on different severities. It is enabled by default. You can change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubeNode.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Nodes whose utilization is above the value configured in fatal Threshold.	Fatal	Kubernetes:NodeNetwork ReceiveFatal	Send email to Kubernetes Administrator
Nodes whose utilization is above the value configured in critical Threshold.	Critical	Kubernetes:NodeNetwork ReceiveCritical	None
Nodes whose utilization is above the value configured in warning Threshold.	Warning	Kubernetes:NodeNetwork ReceiveWarning	None

*Note: the unit is percentage.

Kubernetes Node Network Send

Purpose

This rule detects abnormal Network Send in bytes for Nodes, and fire alarm on different severities. It is enabled by default. You can change value of the registry variables or use your own value to change the threshold of each severities.

Scope

KubeNode.metrics

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Nodes whose utilization is above the value configured in fatal Threshold.	Fatal	Kubernetes:NodeNetwork SendFatal	Send email to Kubernetes Administrator
Nodes whose utilization is above the value configured in critical Threshold.	Critical	Kubernetes:NodeNetwork SendCritical	None
Nodes whose utilization is above the value configured in warning Threshold.	Warning	Kubernetes:NodeNetwork SendWarning	None

*Note: the unit is percentage.

Docker Swarm

All rules are controlled by registry variable Docker:AlertSensitivity. If the value is 0, then no alarm can be fired. If the value is 1, warning level alarm can be fired. If the value is above 1, then all level alarm can be fired.

Docker Swarm Administrator email address can be configured in Registry Variable Docker:DockerAdmin.

Health Check

Docker Container Status

Purpose

This rule detects abnormal Container health status and fires alarm for different severity abnormal health status.

Scope

DockerContainer

Conditions and Severities

Conditions	Severity	Action
Container that is already stopped for abnormal reason.	Critical	Send email to Docker Swarm Administrator

Docker Container Status - Paused

Purpose

This rule detects abnormal long-time paused Container and fires alarm for different severity abnormal health status.

Scope

DockerContainer

Conditions and Severities

Conditions	Severity	Action
Container paused for two continuous data submission periods.	Warning	None

Docker Service Status

Purpose

This rule detects abnormal Docker Swarm Service health status and fires alarm for different severity abnormal health status.

Scope

DockerService

Conditions and Severities

Conditions	Severity	Action
Missing some of the replicated task running for this service.	Critical	Send email to Docker Swarm Administrator

Docker Task Status

Purpose

This rule detects abnormal Docker Swarm Task health status and fires alarm for different severity abnormal health status.

Scope

DockerTask

Conditions and Severities

Conditions	Severity	Action
Task that is in failed, orphaned or remove status.	Critical	Send email to Docker Swarm Administrator

Docker Task Status -- pending

Purpose

This rule detects abnormal long-time pending Docker Swarm Task and fires alarm for different severity abnormal health status.

Scope

DockerTask

Conditions and Severities

Conditions	Severity	Action
Task that is in pending status for two continuous data submission periods.	Warning	None

Usage

Docker Swarm Container CPU Utilization

Purpose

This rule detects abnormal CPU Utilization for Docker Swarm Containers, and fire alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities. This rule only works for those Containers that configures CPU limit.

Scope

DockerContainerCPU

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Container whose usage is about to reach the limit, the ration is above the value configured in fatal Threshold.	Fatal	Docker:ContainerCpu UtilizationFatal	Send email to Kubernetes Administrator
Pods whose usage is about to reach the limit, the ration is above the value configured in critical Threshold.	Critical	Docker:ContainerCpu UtilizationCritical	None
Pods whose usage is about to reach the limit, the ration is above the value configured in warning Threshold.	Warning	Docker:ContainerCpu UtilizationWarning	None

*Note: the unit is percentage.

Docker Swarm Container Memory Utilization

Purpose

This rule detects abnormal Memory Utilization for Docker Swarm Containers, and fire alarm on different severities. It is disabled by default. You can customize it and enable it based on your different requirements. For more details about customization, refer to Customization on page 74. You can also change value of the registry variables or use your own value to change the threshold of each severities. This rule only works for those Containers that configures Memory limit.

Scope

DockerContainerMemory

Conditions and Severities

Conditions	Severity	Threshold (Registry Variable)*	Action
Container whose usage is about to reach the limit, the ration is above the value configured in fatal Threshold.	Fatal	Docker:ContainerMemory UtilizationFatal	Send email to Kubernetes Administrator
Pods whose usage is about to reach the limit, the ration is above the value configured in critical Threshold.	Critical	Docker:ContainerMemory UtilizationCritical	None
Pods whose usage is about to reach the limit, the ration is above the value configured in warning Threshold.	Warning	Docker:ContainerMemory UtilizationWarning	None

*Note: the unit is percentage.

Customization

To customize a rule, *Rule Scope* and *Condition* will be used frequently.

To access Rule Scope and Condition, do the following:

1 Under **Dashboards**, click **Administration** > **Rules & Notifications** > **Rules**, then click on the rule and select *View and Edit*.

2 Click **Rule Editor** on the *Rule Detail* popup dialog box. Then click **Continue** on the *Confirm Edit Rule* popup dialog box.

3 On the **Rule Editor** dashboard, *Rule Scope* can be located on the **Rule Definition** tab and Condition can be located on the **Condition & Actions** tab.

Figure 46. Rule Scope

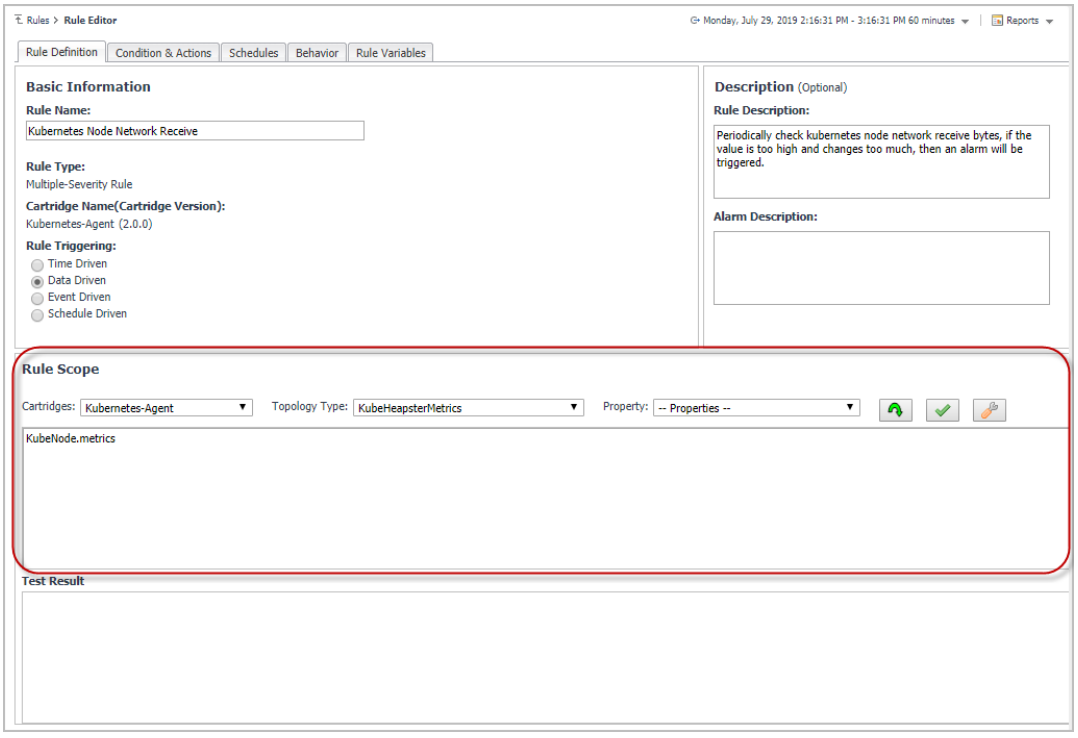
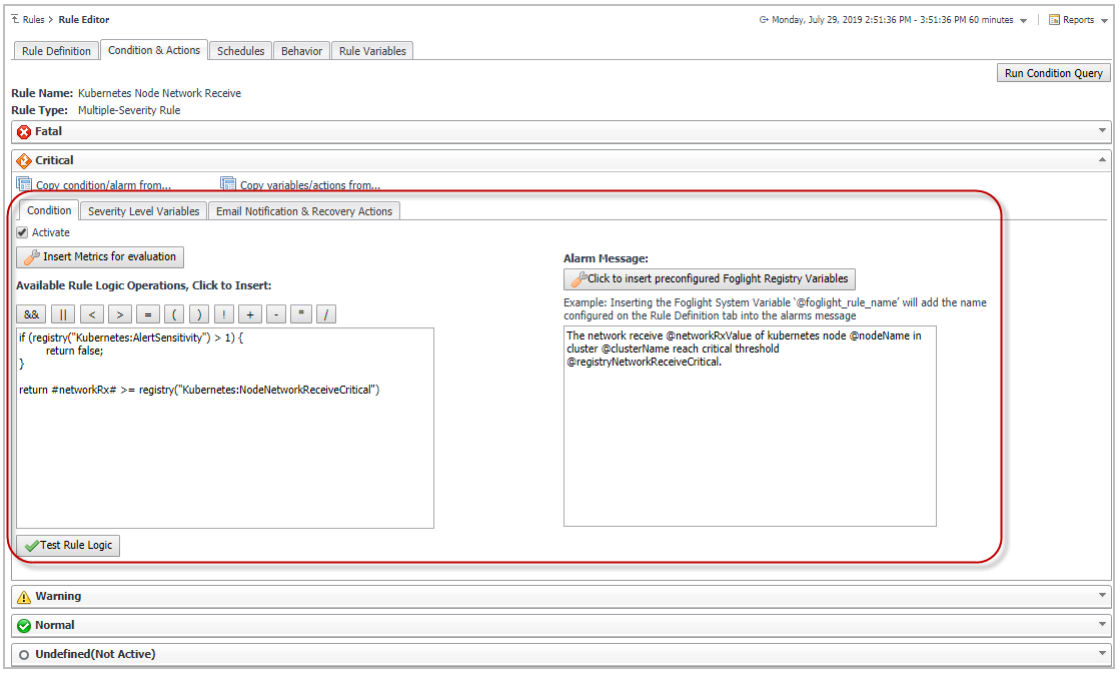


Figure 47. Condition



Kubernetes

Filter Pods by Cluster

Finding Pods inside cluster “kubecluster”, enter following statement in the Scope of a rule, and choose KubePod as the Topology Type in the Rule Scope.

```
KubePod where namespace.cluster.name='kubecluster'
```

Filter Pods by Namespace

Finding Pods inside namespace “default” of Cluster “kubecluster”, enter following statement in the Scope of a rule, and choose KubePod as the Topology Type in the Rule Scope.

```
KubePod where namespace.cluster.name='kubecluster' and namespace.name='test'
```

Filter Nodes by Cluster

Finding Nodes inside cluster “kubecluster”, enter following statement in the Scope of a rule, and choose KubeNode as the Topology Type in the Rule Scope.

```
KubeNode where cluster.name='nancyakscluster'
```

Filter Pod by Labels

Find Pods with labels “run=nginx” and “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose KubePod as the Topology Type in the Rule Scope.

```
KubePod where labels.key='run' and labels.value='nginx-rollingupdate' and
labels.key='env' and labels.value='prod'
```

If you want to find Pods by labels in namespace “test” of cluster “kubecluster”, you can append *and* `namespace.cluster.name='kubecluster' and namespace.name='test'` to the end of above statement.

Filter Node by Labels

Find Nodes with labels “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose KubeNode as the Topology Type in the Rule Scope.

```
KubeNode where labels.key='env' and labels.value='prod'
```

If you want to find Nodes by labels in cluster “kubecluster”, you can append `and cluster.name='kubecluster'` to the end of above statement.

Filter Pod Metrics by Pod Labels

Find Pods Metrics with labels “run=nginx” and “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose KubeHeapsterMetrics as the Topology Type in the Rule Scope.

```
KubePod.metrics where object.labels.key='run' and object.labels.value='nginx' and
object.labels.key='env' and object.labels.value='prod'
```

If you want to find Pods by labels in namespace “test” of cluster “kubecluster”, you can append `and namespace.cluster.name='kubecluster' and namespace.name='test'` to the end of above statement.

Filter Nodes Metrics by Node Labels

Find Node Metrics with labels “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose KubeHeapsterMetrics as the Topology Type in the Rule Scope.

```
KubeNode.metrics where object.labels.key='env' and object.labels.value='prod'
```

If you want to find Nodes by labels in cluster “kubecluster”, you can append `and cluster.name='kubecluster'` to the end of above statement.

Docker Swarm

Filter Container by Swarm Cluster

Find Containers in cluster “dockercluster”, enter following statement in the Scope of a rule, and choose DockerContainer as the Topology Type in the Rule Scope.

```
DockerContainer where dockerSwarm.service.cluster.name='kicakdscluster'
```

Filter Container by Labels

Find Containers with labels “com.docker.stack.namespace=nginx” and “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose DockerContainer as the Topology Type in the Rule Scope.

```
DockerContainer where labels.key='com.docker.stack.namespace' and
labels.value='nginx' and labels.key='env' and labels.value='prod'
```

If you want to find Containers by labels in cluster “swarmcluster”, you can append `and dockerSwarm.service.cluster.name='kicakdscluster'` to the end of above statement.

Filter Docker Host by Swarm Cluster

Find Docker Hosts in cluster “dockercluster”, enter following statement in the Scope of a rule, and choose DockerHost as the Topology Type in the Rule Scope.

```
DockerHost where dockerSwarmNodeInfo.node.cluster.name='kicakdscluster'
```

Filter Container CPU Usage by Container Labels

Find Container CPU Usage by container labels “com.docker.stack.namespace=nginx” and “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose DockerContainerCPU as the Topology Type in the Rule Scope.

```
DockerContainerCPU where container.labels.key='com.docker.stack.namespace' and
container.labels.value='nginx' and container.labels.key='env' and
container.labels.value='prod'
```

If you want to find Containers by labels in cluster “swarmcluster”, you can append *and* `container.dockerSwarm.service.cluster.name='kicakdscluster'` to the end of above statement.

Filter Container Memory Usage by Container Labels

Find Container CPU Usage by container labels “com.docker.stack.namespace=nginx” and “env=prod” among all clusters, enter following statement in the Scope of a rule, and choose DockerContainerMemory as the Topology Type in the Rule Scope.

```
DockerContainerMemory where container.labels.key='com.docker.stack.namespace' and
container.labels.value='nginx' and container.labels.key='env' and
container.labels.value='prod'
```

If you want to find Containers by labels in cluster “swarmcluster”, you can append *and* `container.dockerSwarm.service.cluster.name='kicakdscluster'` to the end of above statement.

About Us

We are more than just a name

We are on a quest to make your information technology work harder for you. That is why we build community-driven software solutions that help you spend less time on IT administration and more time on business innovation. We help you modernize your data center, get you to the cloud quicker and provide the expertise, security and accessibility you need to grow your data-driven business. Combined with Quest's invitation to the global community to be a part of its innovation, and our firm commitment to ensuring customer satisfaction, we continue to deliver solutions that have a real impact on our customers today and leave a legacy we are proud of. We are challenging the status quo by transforming into a new software company. And as your partner, we work tirelessly to make sure your information technology is designed for you and by you. This is our mission, and we are in this together. Welcome to a new Quest. You are invited to Join the Innovation™.

Our brand, our vision. Together.

Our logo reflects our story: innovation, community and support. An important part of this story begins with the letter Q. It is a perfect circle, representing our commitment to technological precision and strength. The space in the Q itself symbolizes our need to add the missing piece—you—to the community, to the new Quest.

Contacting Quest

For sales or other inquiries, visit https://www.quest.com/company/contact-us.aspx/.

Technical support resources

Technical support is available to Quest customers with a valid maintenance contract and customers who have trial versions. You can access the Quest Support Portal at https://support.quest.com.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request.
- View Knowledge Base articles.
- Sign up for product notifications.
- Download software and technical documentation.
- View how-to-videos.
- Engage in community discussions.
- Chat with support engineers online.
- View services to assist you with your product.