Monitoring Docker

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: HOW DOES EG ENTERPRISE MONITOR DOCKER? 3

2.1 How does eG Enterprise monitor Docker Hosts managed by Kubernetes Cluster? 3

2.2 How does eG Enterprise monitor a Standalone Docker Host? 5

2.3 Managing the Docker Hosts 6

CHAPTER 3: MONITORING THE DOCKER HOSTS 10

3.1 The Docker Engine Layer 11

3.1.1 Docker Service Status Test 12

3.1.2 Docker Storage Test 14

3.1.3 Docker Events Test 17

3.1.4 Docker Images Test 20

3.1.5 Docker Images - Performance Test 23

3.2 The Docker Containers Layer 27

3.2.1 Docker Containers - Status Test 27

3.2.2 Docker Containers - Uptime Test 31

3.2.3 Docker Containers - Connectivity Test 33

3.2.4 Docker Containers - Performance Test 35

ABOUT EG INNOVATIONS 44

Table of Figures

Figure 1.1: The Docker architecture 1
Figure 2.1: Identifying the value of the EG_AGENT_IDENTIFIER ID parameter 5
Figure 2.2: Adding a Docker component in an agent based manner 7
Figure 2.3: Adding a Docker component in an agentless manner 8
Figure 2.4: List of unconfigured tests to be configured for the Docker component 9
Figure 2.5: Configuring the Docker - Containers Performance test 9
Figure 3.1: The layer model of Docker host 10
Figure 3.2: The tests mapped to the Docker Engine layer 12
Figure 3.3: The detailed diagnosis of the Total Events measure 20
Figure 3.4: The detailed diagnosis of the Images mapped to containers measure 23
Figure 3.5: The detailed diagnosis of the Total containers measure 26
Figure 3.6: The detailed diagnosis of the Running containers measure 27
Figure 3.7: The tests mapped to the Docker Containers layer 27
Figure 3.8: The detailed diagnosis of the Total Containers measure 30

Chapter 1: Introduction

Docker is an open platform for developing, shipping, and running applications. Docker is designed to deliver your applications faster. With Docker you can separate your applications from your infrastructure and treat your infrastructure like a managed application. Docker helps you ship code faster, test faster, deploy faster, and shorten the cycle between writing code and running code. Docker does this by combining a lightweight container virtualization platform with workflows and tooling that help you manage and deploy your applications.

At its core, Docker provides a way to run almost any application securely isolated in a container. The isolation and security allow you to run many containers simultaneously on your host. The lightweight nature of containers, which run without the extra load of a hypervisor, means you can get more out of your hardware.

Docker has two major components:

- **Docker**: the open source container virtualization platform.
- **Docker Hub**: the Software-as-a-Service platform for sharing and managing Docker containers.

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing the Docker containers. Both the Docker client and the daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate via sockets or through a RESTful API.

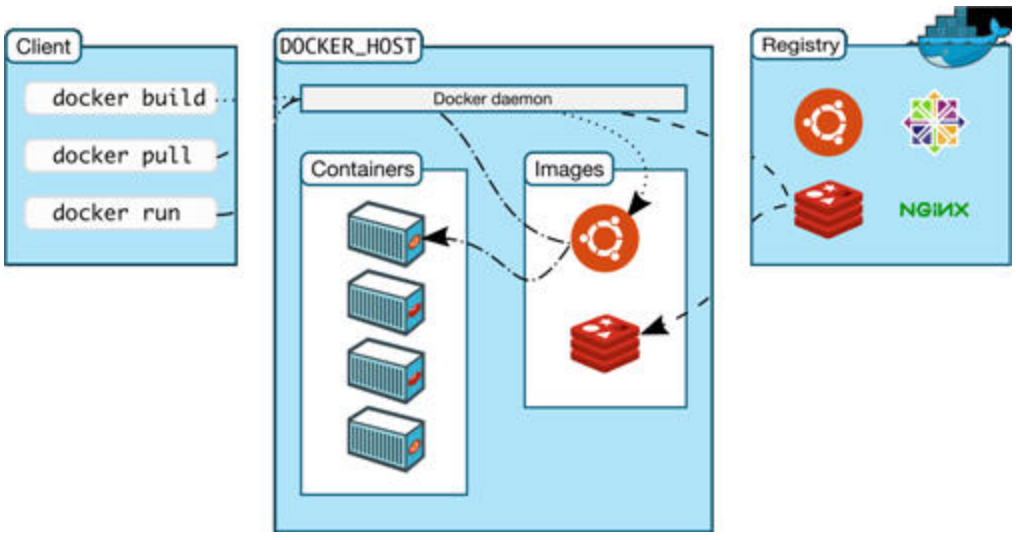

Figure 1.1: The Docker architecture

The Docker daemon runs on a host machine. The user does not directly interact with the daemon, but instead through the Docker client.

The Docker client, in the form of the docker binary, is the primary user interface to Docker. It accepts commands from the user and communicates back and forth with a Docker daemon.

To understand Docker's internals, you need to know about three components:

- **Docker images**: A Docker image is a read-only template. For example, an image could contain an Ubuntu operating system with Apache and your web application installed. Images are used to create Docker containers. Docker provides a simple way to build new images or update existing images, or you can download Docker images that other people have already created. Docker images are the **build** component of Docker.
- **Docker registries**: Docker registries hold images. These are public or private stores from which you upload or download images. The public Docker registry is provided with the Docker Hub. It serves a huge collection of existing images for your use. These can be images you create yourself or you can use images that others have previously created. Docker registries are the **distribution** component of Docker.
- **Docker containers**: Docker containers are similar to a directory. A Docker container holds everything that is needed for an application to run. Each container is created from a Docker image. Docker containers can be run, started, stopped, moved, and deleted. Each container is an isolated and secure application platform. Docker containers are the **run** component of Docker.

Due to the lightweight architecture of the docker and fast accessibility of the applications, Docker is gaining a rapid foothold among IT giants. As continuous access to the applications is the key in such environments, even the smallest slip in the performance of the Docker would result in huge losses. To ensure the 24x7 availability of the Docker and high performance rate, administrators need to closely monitor the performance and status of the Docker and its associated components, promptly detect abnormalities, and fix them before end-users notice. This is where eG Enterprise helps administrators.

Note:

eG Enterprise provides monitoring support to Docker on Linux platforms only, and not on Windows.

Chapter 2: How does eG Enterprise Monitor Docker?

eG Enterprise is capable of monitoring both Standalone Docker Host as well as the Docker Hosts managed by a Kubernetes Cluster by installing an eG agent as a container on the Docker Hosts. To monitor a Standalone Docker Host, administrators need to install the eG agent container published in the public Docker repository on the Docker Host. To monitor Docker Hosts managed by a Kubernetes Cluster, administrators need to deploy a DaemonSet which will automatically install the eG agent as a container on all the Docker Hosts in the Cluster. A DaemonSet ensures that all (or some) Nodes run a copy of the eG agent container. As Docker Hosts are added to the Kubernetes cluster, the DaemonSet automatically runs a copy of the eG agent container on those Docker Hosts.

2.1 How does eG Enterprise monitor Docker Hosts managed by Kubernetes Cluster?

To monitor the Docker Hosts managed by a Kubernetes Cluster, administrators need to deploy a DaemonSet which will automatically install the eG agent container (published on the Docker repository) on all the Docker Hosts.

The steps that follow will discuss in detail on how the eG agent container helps in monitoring the Docker Hosts managed by a Kubernetes cluster:

1. Login to the master node of the Kubernetes Cluster.
2. Access the following URL to download the **agent.yaml** file: https://raw.githubusercontent.com/egmonitoring/container/master/agent.yaml
3. The next step is to edit the downloaded **agent.yaml** file so as to enable the eG agent container to communicate with the eG manager. In the **env:** section of the **agent.yaml** file, specify the **Host IP/Fully Qualified Domain name of the eG manager** against the value option for the **EG_ MANAGER** parameter.

```
env:
- name: EG_MANAGER
value: "<Host IP/Fully qualified domain name of the eG manager>"
```

For example, if the Host IP of the eG manager is 192.168.8.72, then, specify the IP address against the value parameter.

```
env:
- name: EG_MANAGER
```

```
value: "192.168.8.72"
```

For example, if the Fully qualified domain name of the eG manager is *cloud.eginnovations.com*, then, specify the same against the value parameter.

```
env:
- name: EG_MANAGER
value: "cloud.eginnovations.com"
```

4. Then, specify the port at which the eG manager listens to. If the eG manager listens on port *7077*, then, specify *7077* against the **value** below the **EG_MANAGER_PORT** parameter.

```
- name: EG_MANAGER_PORT
value: "7077"
```

5. Next, indicate whether the eG agent container is to use SSL for communicating with the eG manager. If yes, specify **true** against the value option below the **EG_MANAGER_SSL** parameter. By default, **false** is set against the **value** option.

```
- name: EG_MANAGER_SSL
value: "true"
```

6. The next step is to specify an eG Agent Identifier ID. In an Enterprise setup or a multi-tenant setup, once a user/tenant - e.g., a user representing a customer / a department / a domain - registers with eG Enterprise to use its monitoring services, eG automatically generates a unique Agent Identifier ID and assigns the same to that user/tenant. If that user/tenant later logs into the eG management console using the registered credentials (email ID and password) and downloads the eG agent container, each eG agent container so downloaded is automatically tagged with that Agent Identifier ID. The downloaded eG agent container, once installed and configured, will automatically start discovering the Docker Hosts. eG Enterprise auto-manages the Docker Hosts and auto-assigns the Docker Hosts to the user/tenant who has the same eG Agent Identifier ID as that of the eG agent container that discovered the host.

   To obtain the eG Agent Identifier ID, the user/tenant needs to log in to the eG management console and click the **Docker** tile in the **What would you like to Discover /Monitor?** page of the eG administrative interface. In Figure 2.1 that appears, the eG Agent Identifier ID associated with the user/tenant is available.

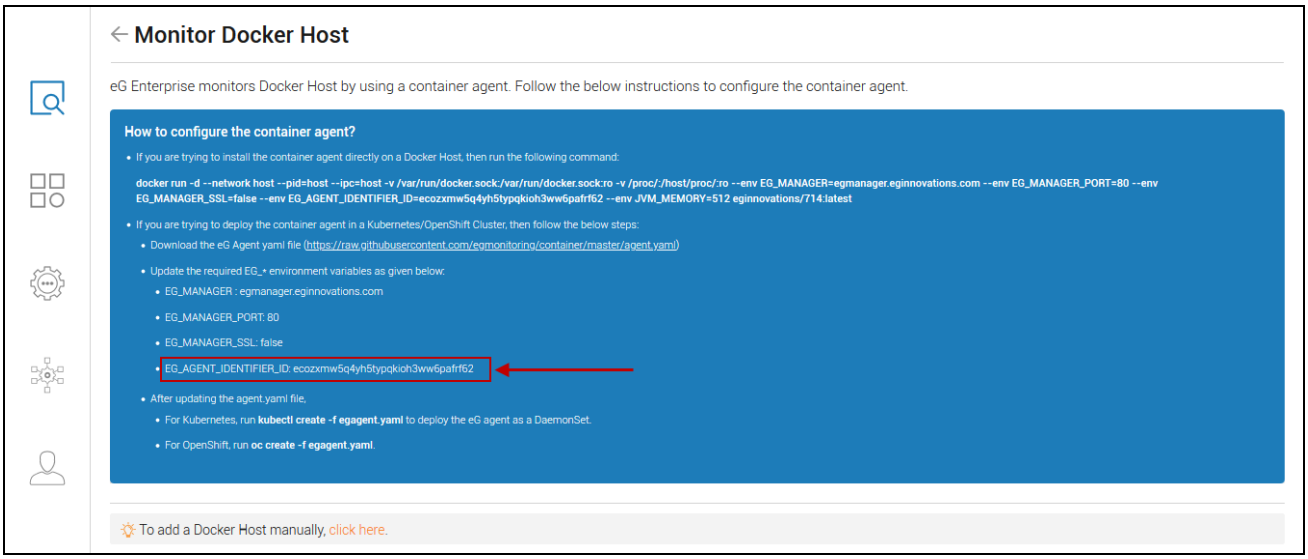

Figure 2.1: Identifying the value of the EG_AGENT_IDENTIFIER ID parameter

This eG Agent Identifier ID should be specified against the **value** option of the **EG_AGENT_IDENTIFIER_ID** parameter.

```
- name: "EG_AGENT_IDENTIFIER_ID"
value: "<Unique ID associated with the user/tenant>"
```

7. After the required credentials are specified, execute the following command on the master node of the Kubernetes Cluster:

   kubectl create -f egagent.yaml

2.2 How does eG Enterprise monitor a Standalone Docker Host?

If you wish to install the eG agent container directly on a standalone Docker Host, then, you need to execute the following command on the Docker Host:

docker run -d --network host --pid=host --ipc=host -v /var/run/docker.sock:/var/run/docker.sock:ro -v /proc/:/host/proc/:ro --env EG_MANAGER=<Host IP/Fully qualified domain name of the eG manager> --env EG_MANAGER_PORT=<port on which the eG manager listens> --env EG_MANAGER_SSL=<true/false> --env EG_AGENT_IDENTIFIER_ID=<Unique ID associated with the user/tenant> --env JVM_MEMORY=512 eginnovations/<version of the eG manager>:latest

An example command is mentioned below:

docker run --network host --pid=host --ipc=host -v /var/run/docker.sock:/var/run/docker.sock:ro -v /proc/:/host/proc/:ro --env EG_MANAGER=192.168.8.72 --env EG_MANAGER_PORT=7077 --

env EG_MANAGER_SSL=true --env EG_AGENT_IDENTIFIER_ ID=ecozxmw5q4yh5typqkioh3ww6pafrf62 --env JVM_MEMORY=512 eginnovations/714:latest

Note:

eG Enterprise provides monitoring support to Docker hosts on Linux platforms only, and not on Windows.

2.3 Managing the Docker Hosts

The eG Enterprise cannot automatically discover the Docker host. This implies that you need to manually add the component for monitoring. To manage a Docker component, do the following:

1. Log into the eG administrative interface.
2. Follow the Components ->Add/Modify menu sequence in the **Infrastructure** tile of the **Admin** menu.
3. In the **COMPONENT** page that appears next, select *Docker* as the **Component type**. Then, click the **Add New Component** button. This will invoke Figure 2.2.

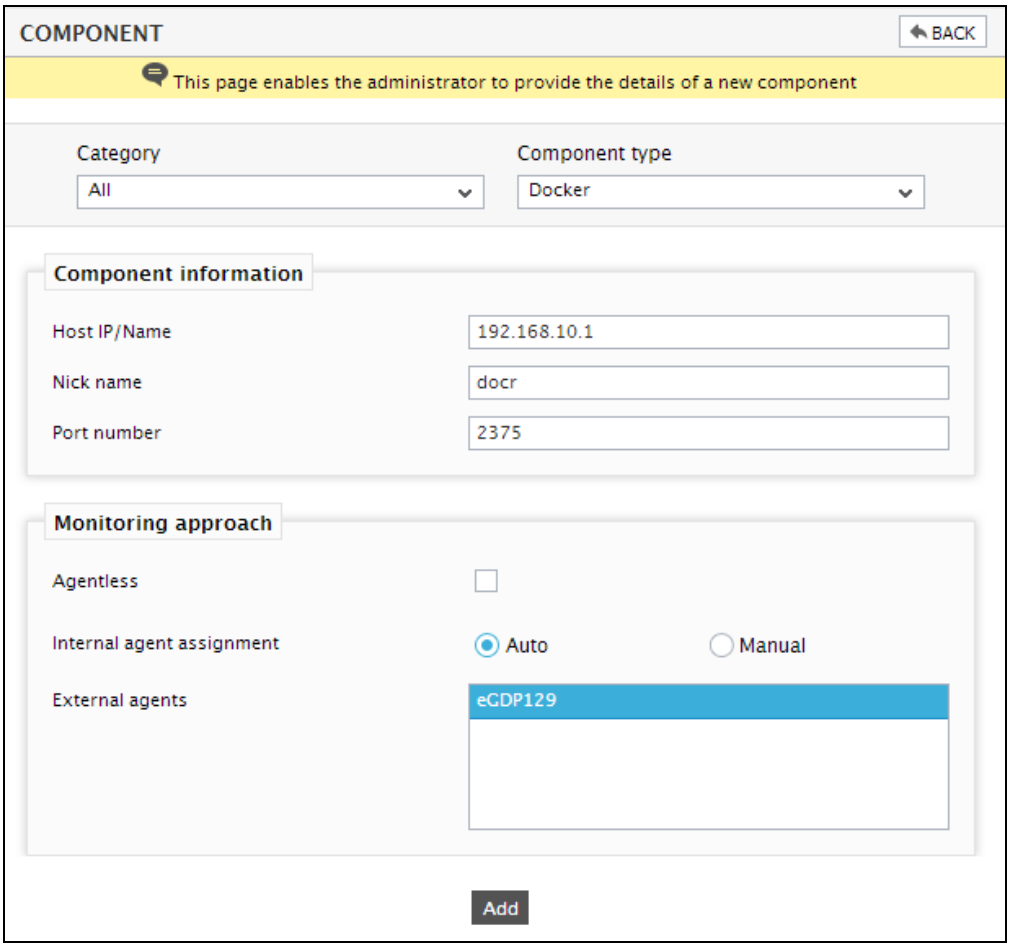

Figure 2.2: Adding a Docker component in an agent based manner

4. Specify the **Host IP/Name** and the **Nick name** of the Docker in Figure 2.2.

   Note:

   eG Enterprise provides monitoring support to Docker hosts on Linux platforms only, and not on Windows.

5. In case you are monitoring the Docker in an agent-based manner, just pick an external agent from the **External agents** list box and click the **Add** button to add the component for monitoring.
6. On the other hand, if you are monitoring the Docker host in an agentless manner, then do the following:

   - Select the **Agentless** check box.
   - Pick the **OS** on which the Docker is running. This is **Linux**.

- Set the **Mode** to **SSH**.
- Set the **Encryption type** to **Password** and the **Remote port** to 22.
- Specify the name of the Unix user who is part of the *docker group* in the **User** text box. To know how to create a *docker group* and add a Unix user to it, follow the steps discussed in Section **Chapter 2**.
- Specify the **Password** corresponding to the **User**.
- Select the Remote agent that will be monitoring the Docker. **Note that the Remote agent you choose should run on a Windows host.**
- Choose an external agent for the server by picking an option from the **External agents** list box.
- Finally, click the **Add** button to add the Docker host for monitoring.

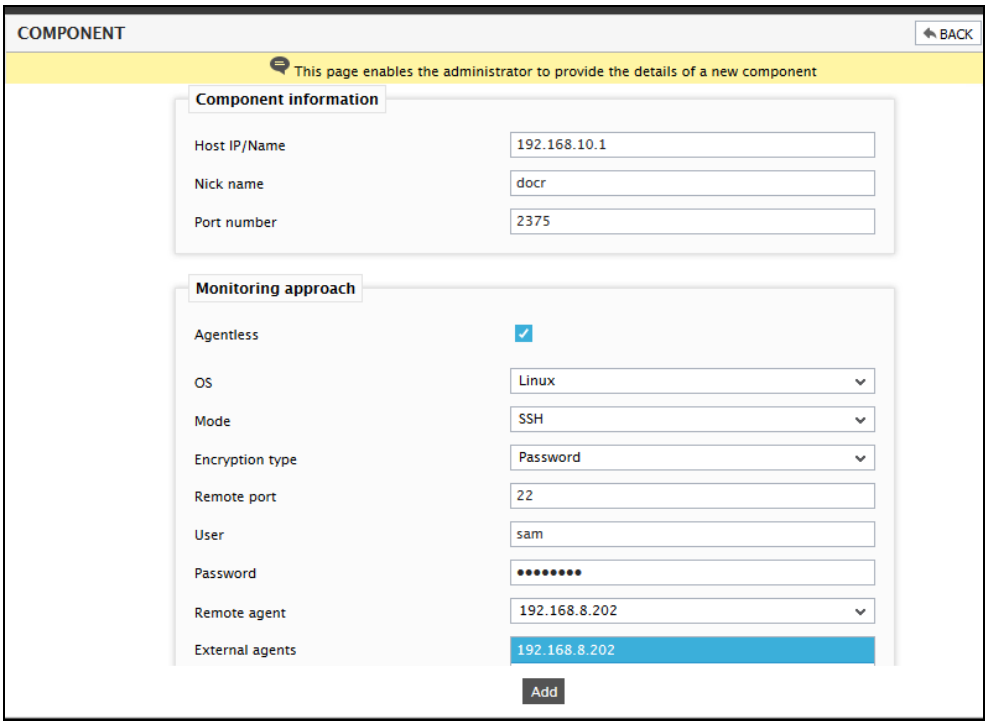

Figure 2.3: Adding a Docker component in an agentless manner

7. When you attempt to sign out, a list of unconfigured tests will appear as shown in Figure 2.4.

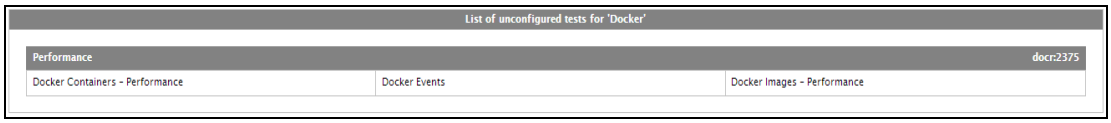

Figure 2.4: List of unconfigured tests to be configured for the Docker component

8. Click on any test in the list of unconfigured tests. For instance, click on the **Docker - Containers Performance** test to configure it. In the page that appears, specify the parameters as shown in Figure 2.5.

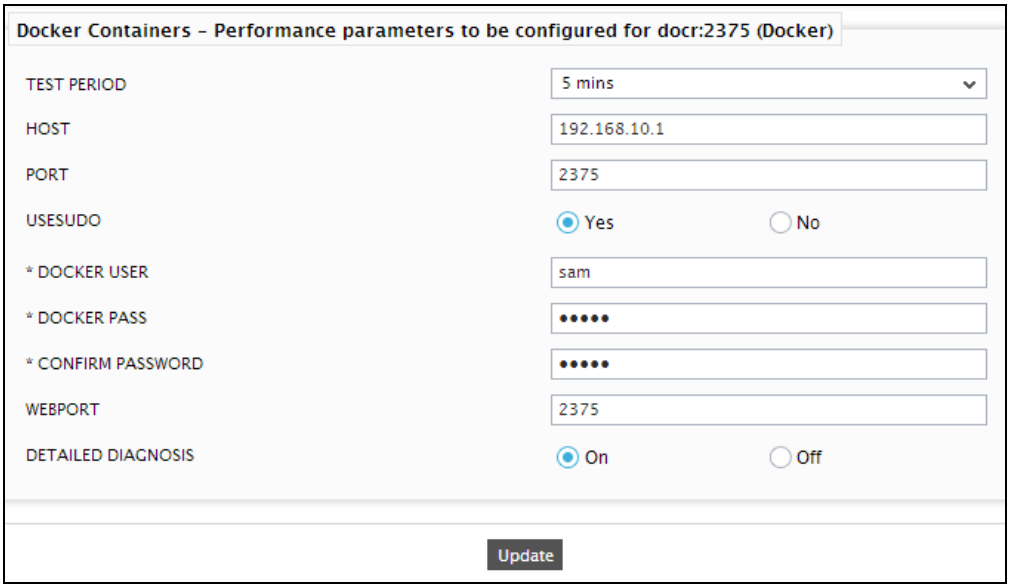

Figure 2.5: Configuring the Docker - Containers Performance test

9. To know how to configure parameters, refer to the **Monitoring the Docker Hosts** chapter.
10. Finally, signout of the eG administrative interface.

Chapter 3: Monitoring the Docker Hosts

eG Enterprise offers a specialized *Docker* monitoring model that monitors each of the key indicators of the performance of Docker hosts - such as the images, containers etc.- and proactively alerts administrators to potential performance bottlenecks, so that administrators can resolve the issues well before end-users complain.

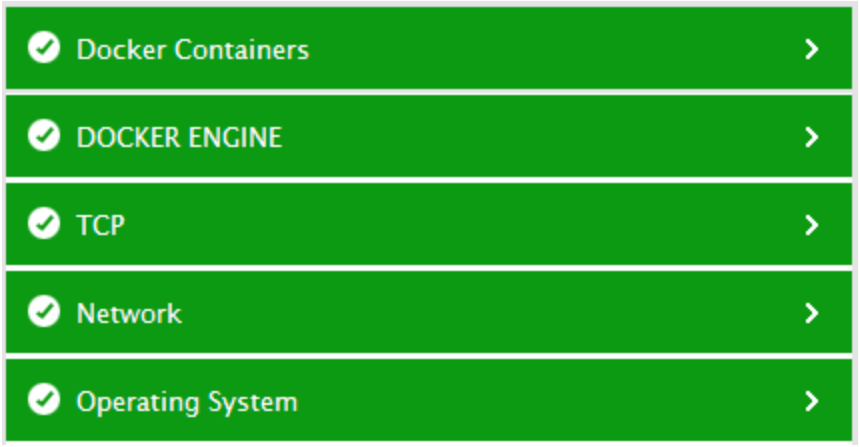

Figure 3.1: The layer model of Docker host

Each layer of Figure 3.1 above is mapped to a variety of tests, each of which report a wealth of useful information related to the docker server. Using these metrics, administrators can find quick and accurate answers to the following performance queries:

- Is the docker service installed?
- What is the current status of the docker service?
- How well the data space and metadata space of the docker server is utilized?
- How many events were triggered on the containers of the docker server? Which type of event was triggered the maximum and the minimum? – is it the create events or start events or stop events or die events?
- What is the total count of the docker images in the server? How many images are used to create the containers?
- What is the disk space utilization of the images that are mapped to the containers?

- How many containers are created from each image of the docker server? How many containers are actually running and how well the CPU, disk and memory resources are utilized by those containers?
- How many containers are available in the server and among them how many containers are currently running? How many containers are added recently and how many are actually removed?
- What is the uptime of each container?
- Is the container available over the network?
- What is the disk space utilization of each container?
- How well data is transmitted and received from each container?
- What is the rate of errors that are transmitted through each container?
- How well the memory is utilized by each container?
- How well data is read from and written to each container?

This chapter deep dives into every layer of the Docker monitoring model, the tests mapped to each layer, and the measures every test reports. Since the **Operating System**, **Network** and **TCP** layers of the Docker are similar to that of a Linux server and had been dealt in extensively in *Monitoring Unix and Windows Servers* document, let us now discuss the layers that are exclusive to the Docker in the forthcoming sections.

3.1 The Docker Engine Layer

The tests mapped to this layer reports the current state of the docker service, the utilization of data and metadata space of the docker, events triggered on the containers of the docker server, image available in the docker, the statistics revealing the number of images mapped to containers and those that are not mapped, individual resource utilization of the containers that are created based on each image etc.

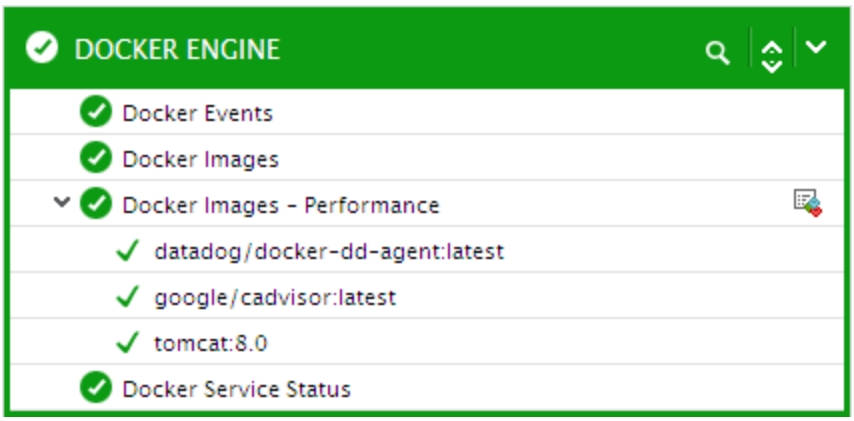

Figure 3.2: The tests mapped to the Docker Engine layer

3.1.1 Docker Service Status Test

A Docker container created from a Docker image holds everything that is needed for an application to run. In order to create a Docker container, the Docker service should be up and running without a glitch. If the Docker containers cannot be created or if the containers become inaccessible, then the applications running on the containers will be rendered inaccessible to the users causing inconvenience. To avoid such inconvenience to the users accessing the Docker containers, administrators need to constantly monitor the current state of the Docker service. The **Docker Service Status** test exactly helps administrators in this regard!

This test helps administrators to continuously monitor the Docker service and reports the current state of the service. In addition, this test helps administrators to figure out whether the Docker service is installed and loaded.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the Docker server to be monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified host listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**

Measurements made by the test

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Is service installed?</td><td>Indicates whether/not the Docker service is installed.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>
Note:
This measure reports the Measure Values listed in the table above while indicating whether the Docker service is installed or not. However, in the graph of this measure, this measure is indicated using only the Numeric Values listed in the above table.</td></tr>
<tr><td>Is service loaded?</td><td>Indicates whether/not the Docker service is loaded.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>
Note:
This measure reports the Measure Values listed in the table above while indicating whether the Docker service is loaded or not. However, in the graph of this measure, this measure is indicated using only the Numeric Values listed in the above table.</td></tr>
<tr><td>Docker service</td><td>Indicates the current state</td><td></td><td>If the Docker service is currently</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>status</td><td>of the Docker service.</td><td></td><td>running, then the value of this measure is reported as Active and if the Docker service is not running, then the value of this measure is reported as In active.

The values reported by this measure and their numeric equivalents are available in the table below:

<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>In Active</td><td>0</td></tr><tr><td>Active</td><td>1</td></tr></table>
Note:

This measure reports the Measure Values listed in the table above while indicating whether the Docker service is active or not. However, in the graph of this measure, the state is indicated using only the Numeric Values listed in the above table.</td></tr>
</table>

3.1.2 Docker Storage Test

When the Docker service runs for the first time in the Docker server, a defined quantity of data space and metadata space is allocated from the Docker volume, using which data can be stored in the Docker container outside of the boot volume (but within the root file system). By default, the Docker offers thin provisioning facility of volumes which means that you have a big pool of available storage blocks within the volume and can create block devices such as virtual disks, of arbitrary size from that pool. The storage blocks will be marked as used or taken from the pool only when you actually write to it. The storage blocks are mapped to the data space and the metadata space for storage needs. Here, the Docker images are stored in the data space and the definite data about the docker images are stored in the metadata space. Since the Docker images are the basic building blocks of the Docker, if the data space or the metadata space is not sufficient, then new images and new containers based on the images cannot be created. If sufficient images are not available, then users cannot create new containers and simultaneously applications will also become inaccessible or the users will have only a limited choice of applications. To avoid such eventualities, it is inevitable that

the data space and metadata space be monitored at regular intervals. The **Docker Storage** test helps you in this regard!

This test reports the utilization of data and metadata space within a Docker server and proactively alerts administrators to potential space crunch, if any.

Note:

This test will run only if the Docker server is installed with the *Device Mapper*, which is a Linux kernels framework for mapping physical block devices onto higher-level virtual block devices.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the Docker server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified host listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Data space used	Indicates the amount of data space currently utilized by this server.	GB	A low value is desired for this measure. If the value of this measure is closer to the Data space total measure, then it indicates that the Docker server is currently running short of space. Administrators have to ensure that sufficient data space is allocated to the Docker server.
Data space total	Indicates the total amount of data space allocated to this server.	GB	

Measurement	Description	Measurement Unit	Interpretation
Data space utilization	Indicates the percentage of data space utilized by this server.	Percent	A high value for this measure indicates that the Docker server is currently running out of space.
Metadata space used	Indicates the amount of metadata space currently utilized by this server.	GB	Metadata is data that describes other data. Meta is a prefix that in most information technology usages means "an underlying definition or description." Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified and file size are examples of very basic document metadata. Having the ability to filter through that metadata makes it much easier for someone to locate a specific document. In addition to document files, metadata is used for images, videos, spreadsheets and web pages. The use of metadata on web pages can be very important. Metadata for web pages contain descriptions of the page's contents, as well as keywords linked to the content. These are usually expressed in the form of metatags.
Metadata space total	Indicates the total amount of metadata space allocated to this server.	GB	
Metadata space utilization	Indicates the percentage of metadata space utilized by this server.	Percent	A low value is desired for this measure.

3.1.3 Docker Events Test

By default, the Docker containers can be created, run, started, stopped, moved, and deleted. These actions are called events in the Docker events API. By constantly monitoring the events triggered by the Docker, administrators can identify how many create events were triggered while creating the containers, how many events were triggered to delete the containers etc. The **Docker Events** test helps administrators to figure out the various events triggered by the Docker server.

This test monitors the Docker server and reports the total number of events triggered on the containers of the Docker server. In addition, this test helps administrators to figure out how many times the start and stop events were triggered to start and stop the containers. Alongside, administrators can also identify the create events, delete events and die events that were triggered on the server. Using this test, administrators can easily identify how well the containers are managed within the Docker server and proactively be alerted to unpleasant eventualities caused due to the slowdowns in application processing as well as processing bottlenecks.

Note:

This test will run and report metrics, only if the following pre-requisites are fulfilled:

- The Docker server should be of v1.5 or above.
- Remote REST API should be enabled on the Docker host. To know how to enable remote REST API, follow the procedure discussed in Section **Chapter 2**
- The eG agent should be 'allowed' to make remote REST API calls to pull metrics. For this purpose, make sure you configure this test with the credentials of a user who has permissions to connect to remote REST API and invoke its methods.
- Make sure that the **WEBPORT** parameter of this test is configured with the port on which remote REST API has been enabled.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the target Docker server being monitored .

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.

Parameter	Description
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	**This flag is not applicable to this test.**
Docker User and Docker Password	Specify the credentials of the user who is authorized to access the remote REST API and invoke its methods for metrics collection.
Confirm Password	Confirm the Password by retyping it here.
Webport	By default, the remote REST API is enabled on port 2375. This implies that by default, this test connects to port 2375 to access the remote REST API and make API calls for metrics collection. In some environments however, the remote REST API can be enabled on a different port. To know how to enable the remote REST API on a different port, follow the procedure discussed in Section **Chapter 2** Make sure you configure this parameter with the exact port on which the remote REST API has been enabled. To know which port number that is, do the following: • Open the /lib/system/system/docker.service file on the Docker host. • In the file, find the line which starts with **ExecStart**. In that line, look for the following entry: *-H=tcp://0.0.0.0:<Remote_API_Port>* The number that appears after ':' in the entry above, is the remote REST API port.
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability

Parameter	Description
	• Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total events	Indicates the total number of events that occurred during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which each event occurred.
Create events	Indicates the number Create events that occurred during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which each event was created.
Delete events	Indicates the number of Delete events that occurred during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which each event was deleted.
Start events	Indicates the number of Start events that occurred during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which the events were started.
Stop events	Indicates the total number of Stop events that occurred during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which the events were stopped.
Die events	Indicates the number of die events during the last measurement period.	Number	The detailed diagnosis of this measure if enabled, lists the Container ID, Container Name, Image, Event type and Time at which the die events occurred.

The detailed diagnosis of the *Total Events* measure lists the containers on which the events occurred.

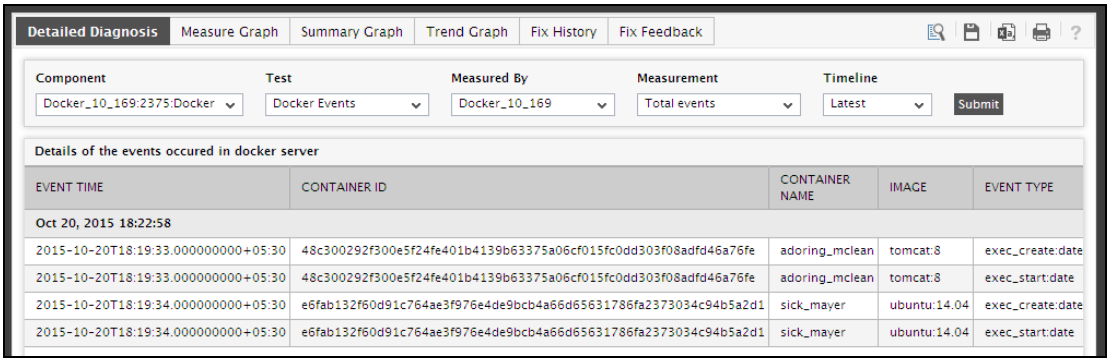

Figure 3.3: The detailed diagnosis of the Total Events measure

3.1.4 Docker Images Test

Images are the basic building blocks of the Docker, and are organized in a layered manner. The images are utilized as *read-only* templates for building multiple Docker containers using layered Docker filesystems. The containers so created will share common files and enhance disk usage and downloads on the containers. Once the containers are created, multiple applications can be installed on them. The applications can also be updated to a new version by simply building a new layer on the existing images rather than replacing the whole image or entirely rebuilding the containers. The containers can be mapped to a single image or to multiple images. If a single image is alone over-utilized in creating the containers or if any image remains unmapped to the containers, then such unmapped images will remain under-utilized but still will occupy unnecessary disk space. If too many such unmapped images exists, then the creation of new images may be impacted and applications cannot be bundled to the existing images/containers resulting in a decreased reusability of disk space, increased disk usage and reduction in the building speed of the containers. This is why, administrators need to frequently check for the images and the disk space occupied by those images. The **Docker Images** test does this check.

This test reports the total number of images that are available on the Docker host/server. In addition, this test helps administrators to compare the numerical statistics of the images that are mapped to the containers and those that are not mapped to any container. Likewise, administrators can also be able to figure out the disk space utilization of images that are mapped to the containers and the disk space utilization of the images that are not mapped to the containers. Using this test, administrators can figure out the images that are sparsely utilized, the images that are consuming too much of disk space etc and take remedial actions to restrict the disk space utilization of the images.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the target Docker server that is being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total Images	Indicates the total number of images available in the Docker server.	Number	

Measurement	Description	Measurement Unit	Interpretation
Images mapped to containers	Indicates the number of images that are mapped to the Docker containers.	Number	Ideally, the value of this measure should be high. A low value of this measure indicates that more number of images are left idle and more disk space is occupied by unused images. This may cause potential space crunch in the disk. The detailed diagnosis of this measure if enabled, lists the images and the containers to which the images are mapped.
Images not mapped to any container	Indicates the number of images that are not mapped to any Docker container.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure if enabled, lists the images that are not mapped to any Docker containers.
Disk space used by images mapped to containers	Indicates the amount of disk space utilized by the images that are mapped to the Docker containers.	MB	The detailed diagnosis of this measure if enabled, lists the images and the containers.
Disk space used by images not mapped to any container	Indicates the amount of disk space utilized by the images that were not mapped to the Docker containers.	MB	A high value of this measure indicates space crunch in the disk. The detailed diagnosis of this measure if enabled, lists the images, which were not mapped with containers.

The detailed diagnosis of the *Images mapped to containers* measure lists the Image, Image ID, the containers mapped to the image, the time at which the container was created and size of the image. Using this measure administrators can figure out the image that is widely used to create a container.

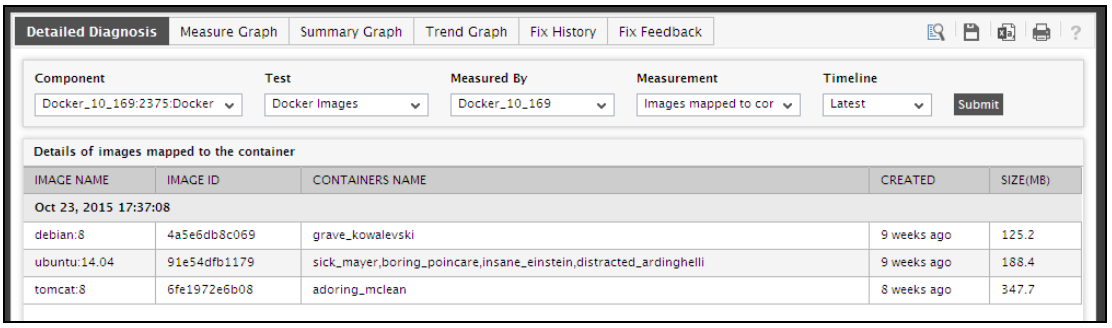

Figure 3.4: The detailed diagnosis of the Images mapped to containers measure

3.1.5 Docker Images - Performance Test

A Docker server can have multiple images using which containers are created. In large environments where the requirements of users vary from minute to minute, it is important for an administrator to cater to their specific needs by providing the images of their choice so that the users can create the containers and load applications accordingly. Often administrators may be confused while identifying the needs of the users. An image that is needed today may not be needed tomorrow to create a container. In such cases, the images may be unutilized but still hold on valuable disk space. To optimize the disk space and eliminate unused images, administrators are required to analyze the performance of the images as well as identify the images that are most commonly used by the users. The **Docker Images – Performance** test helps administrators in this regard!

This test monitors the images of the Docker server and reports the number of containers created using each image. By analyzing the number of running containers created from each image and the resource utilization of those containers, administrators can figure out the image that is widely used by the users to create the containers and analyze how well the resources are utilized by those containers.

Note:

This test will run and report metrics, only if the following pre-requisites are fulfilled:

- The Docker server should be of v1.5 or above.
- Remote REST API should be enabled on the Docker host. To know how to enable remote API, follow the procedure discussed in Section **Chapter 2**
- The eG agent should be 'allowed' to make remote REST API calls to pull metrics. For this purpose, make sure you configure this test with the credentials of a user who has permissions to connect to remote REST API and invoke its methods.

- Make sure that the **WEBPORT** parameter of this test is configured with port on which remote REST API has been enabled.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each image available on the target Docker server that is being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	**This flag is not applicable to this test.**
Docker User and Docker Password	Specify the credentials of the user who is authorized to access remote REST API and invoke its methods for metrics collection.
Confirm Password	Confirm the Password by retyping it here.
Webport	By default, the remote REST API is enabled on port 2375. This implies that by default, this test connects to port 2375 to access the remote REST API and make API calls for metrics collection. In some environments however, the remote REST API can be enabled on a different port. To know how to enable the remote REST API on a different port, follow the procedure discussed in Section **Chapter 2** Make sure you configure this parameter with the exact port on which the remote REST API has been enabled. To know which port number that is, do the following: • Open the /lib/system/system/docker.service file on the Docker host. • In the file, find the line which starts with **ExecStart**. In that line, look for the following entry: *-H=tcp://0.0.0.0:<Remote_API_Port>* The number that appears after ':' in the entry above, is the remote REST API port.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are

Parameter	Description
	detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total containers	Indicates the total number of containers that were mapped to this image.	Number	The detailed diagnosis of this measure if enabled, lists the name of the containers. Compare the value of this measure across the images to identify the image to which maximum number of containers are mapped.
Running containers	Indicates the number of containers that are created from this image and are currently in Running state.	Number	Ideally, the value of this measure should be high. A low value for this measure is a cause of concern. The detailed diagnosis of this measure if enabled, lists the containers that are currently in Running state and the resource utilization of the containers.
CPU utilization of containers	Indicates the percentage of CPU utilized by the containers (created from this image) that are currently in Running state.	Percent	This measure is only applicable for the Docker server 1.5 and above. A sudden increase or decrease in the value of this measure could be a cause of concern.
Memory used by containers	Indicates the amount of memory utilized by the containers (created from this image) that are currently in Running state.	MB	This measure is only applicable for Docker server 1.5 and above. A low value is desired for this measure. A high value of this measure indicates that the Docker is running out of

Measurement	Description	Measurement Unit	Interpretation
			space.
Memory utilization of containers	Indicates the percentage of memory utilized by the containers (created from this image) that are currently in Running state.	Percent	This measure is only applicable for the Docker server 1.5 and above. A value close to 100% indicates that the memory is running out of space.
Disk space used by image	Indicates the amount of disk space utilized by this image.	MB	Ideally, the value of this measure should be low. A significant increase in the value of this measure could be a cause of concern.

The detailed diagnosis of the *Total containers* measure lists the containers that were mapped to an image.

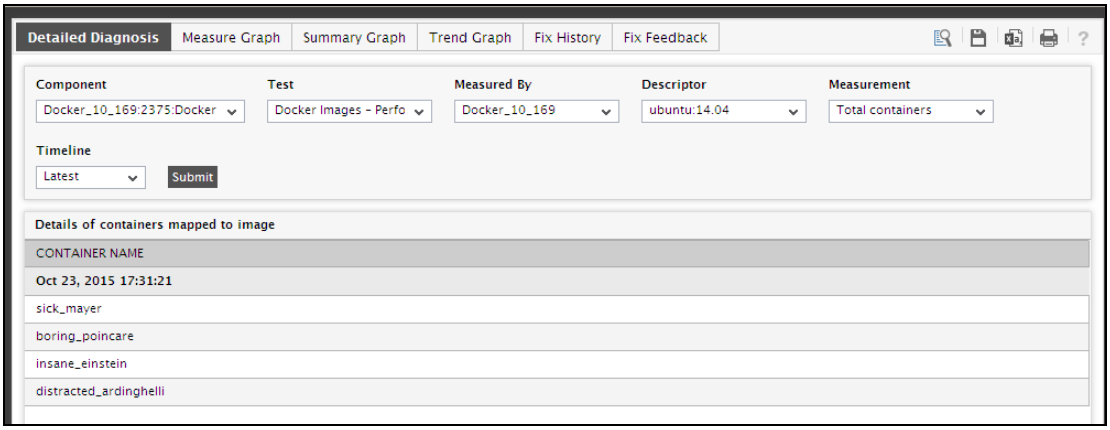

Figure 3.5: The detailed diagnosis of the Total containers measure

The detailed diagnosis of the *Running containers* lists the containers that are currently running and the split up resource utilization (CPU, memory and memory usage) for each of the listed containers.

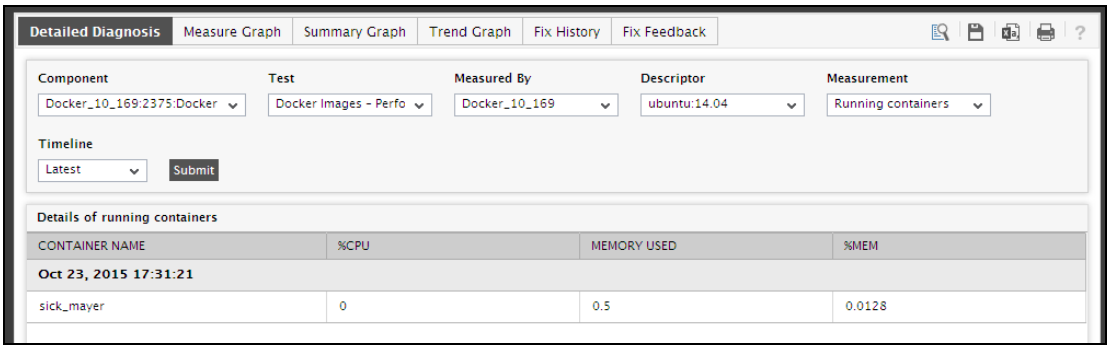

Figure 3.6: The detailed diagnosis of the Running containers measure

3.2 The Docker Containers Layer

This layer monitors the docker containers and reports the performance of each container, the uptime of the container, the network connectivity to each container and the resource and memory utilization of each container etc.

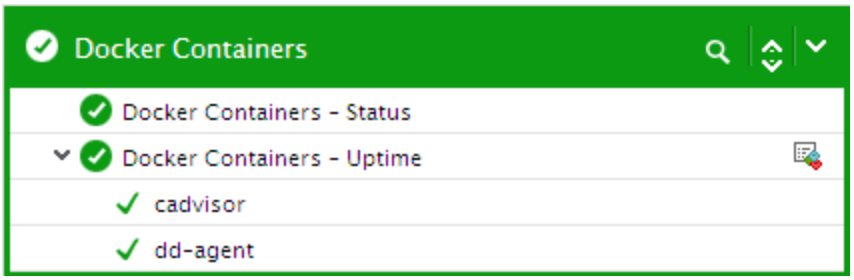

Figure 3.7: The tests mapped to the Docker Containers layer

3.2.1 Docker Containers - Status Test

Docker containers can be built within seconds based on a Docker image. In large environments where multiple containers are created simultaneously, administrators may actually want to keep track on the containers that are currently running, the containers that are currently stopped, and the containers that were removed etc so that containers that are sparsely used and containers that crashed can be identified easily. The **Docker Containers - Status** test helps administrators to optimize the containers and enhance the overall performance of the Docker server.

This test monitors the containers available in the Docker server and reports the numerical statistics of the total containers, the containers in Running state, the containers that were added, the containers that were removed etc. In addition, this test reports the containers that are not running for a time duration above the specified limit and the containers that are utilizing disk space above the

specified limit. By analyzing this, administrators can figure out the overall performance of the Docker server and identify bottlenecks if any.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the Docker server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**
Docker User and Docker Password	Specify the credentials of the user who is authorized to execute the docker commands on the target server
Size Limit in MB	By default, the disk space utilized by each container may vary according to the size of the container. Sometimes, the containers may be over-utilizing the disk space which when left unnoticed may hamper the creation of new containers. To figure out such containers, you can specify a disk space limit beyond which the containers can be termed as large sized container. For example, if you wish to monitor the number of containers that are utilizing disk space above 50 MB, then specify 50 against this text box.
Time Limit in Weeks	For this test to report the numerical statistics of the containers that were not started/running, set a valid value against this parameter. For example, if you wish to report the containers that were not started for more than 3 weeks, then set 3 against this text box.
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be

Parameter	Description
	configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total containers	Indicates the total number of containers in this server.	Number	The detailed diagnosis of this measure if enabled, lists the name of the container, the container ID, the image used to create the container, the container creation time, the status of the container and the size of the container.
Running containers	Indicates the number of containers that are in Running state.	Number	A high value of this measure indicates that more number of containers are in running state as a result more number of applications can be accessed actively. To know which containers are currently running, use the detailed diagnosis capability of this measure.
Stopped containers	Indicates the number of containers that are currently not running in this server.	Number	To know more about the containers that are currently not running, use the detailed diagnosis capability of this measure (if enabled).
Added containers	Indicates the number of containers that were created during the last measurement period.	Number	The detailed diagnosis of this measure, if enabled, lists the names of the containers that were created.
Removed containers	Indicates the number of	Number	The detailed diagnosis of this

Measurement	Description	Measurement Unit	Interpretation
	containers that were removed from this server during the last measurement period.		measure, if enabled, lists the containers that were removed. If too many containers are removed, then the users may not be able to access the hosted applications thus leading to a performance bottleneck of the Docker server. Administrators are therefore required to keep a check on the containers that are removed from the Docker server.
Large size containers	Indicates the number of containers that were using more disk space than the limit configured against the Size Limit in MB parameter.	Number	A low value is desired for this measure. The detailed diagnosis of this measure if enabled, lists the names of the containers that are over-utilizing the disk space.
Containers not started for long time	Indicates the number of containers that were not running for more than the configured Time Limit in Weeks.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure if enabled, lists the containers that are not running for longer time.

The detailed diagnosis of the *Total Containers* measure lists all the containers that were created and the images that were used to create those containers.

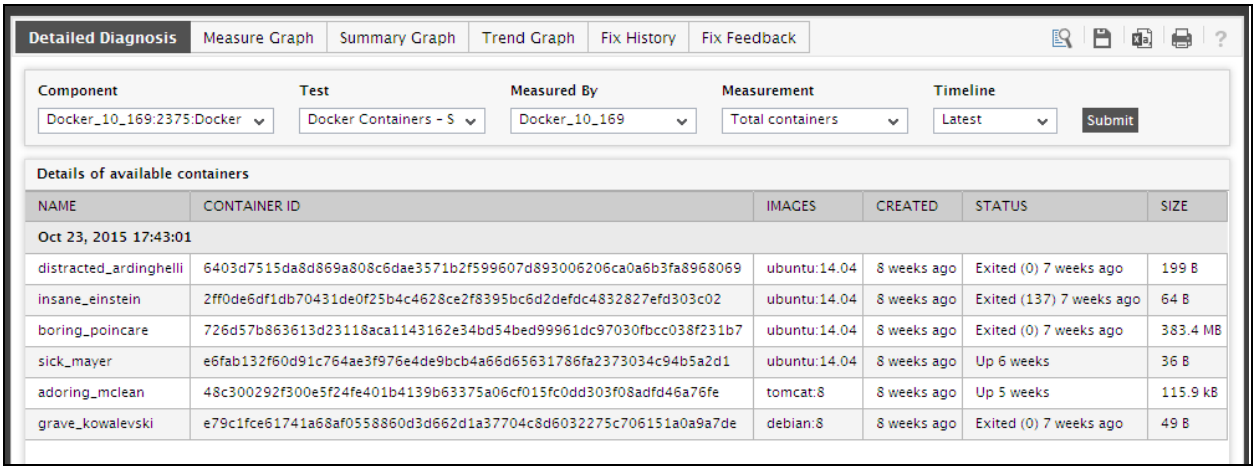

Figure 3.8: The detailed diagnosis of the Total Containers measure

3.2.2 Docker Containers - Uptime Test

In environments where Docker server is used extensively, it is essential to monitor the uptime of critical containers within the Docker server. By tracking the uptime of each of the containers, administrators can determine what percentage of time a container has been up. Comparing this value with service level targets, administrators can determine the most trouble-prone areas of the infrastructure.

In some environments, administrators may schedule periodic reboots of their containers. By knowing that a specific container has been up for an unusually long time, an administrator may come to know that the scheduled reboot task is not working on a container.

The **Docker Containers - Uptime** test included in the eG agent monitors the uptime of critical containers in the target Docker server.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each container available in the Docker server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified host listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**
Docker User and Docker Password	Specify the credentials of the user who is authorized to execute the docker commands on the target server
DD Frequency	Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency.

Parameter	Description
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Has the container been restarted?	Indicates whether/not this container was rebooted.		The values reported by this measure and their numeric equivalents are available in the table below: Measure Value / Numeric Value: No = 0; Yes = 1 **Note:** This measure reports the **Measure Value**s listed in the table above while indicating whether /not this container was rebooted. However, in the graph of this measure, the measure is indicated using only the **Numeric Value**s listed in the above table.
Uptime of the container during the last measure period	Indicates the time duration for which this container has been up since the last time this test ran	Secs	If this container has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to

Measurement	Description	Measurement Unit	Interpretation
			the measurement period. If this container was rebooted during the last measurement period, this value will be less than the measurement period of the test. The accuracy of this metric is dependent on the measurement period - the smaller the measurement period, greater the accuracy.
Total uptime of the container	Indicates the total time that this container has been up since its last reboot.	Mins	Administrators may wish to be alerted if a container has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

3.2.3 Docker Containers - Connectivity Test

When Docker starts, it creates a virtual interface called “Ethernet bridge” on the Docker server. The Ethernet bridge automatically forwards packets to/from the Docker to the external network interfaces. Every time, Docker creates the containers, it creates a pair of “peer” interfaces that are like opposite ends of a pipe. The packets sent through one of the peer interfaces will be received by the other peer interface. By binding the peer interfaces to the Ethernet bridge, Docker creates a virtual subnet shared between the Docker server and every Docker container so as to ensure uninterrupted communication between the Docker server and the Docker containers. For a connection to be uninterrupted, it is essential to keep track of the network delay as well as the packet loss. The **Docker Containers – Connectivity** test helps administrators to keep track of such delays and loss.

This test monitors the network connectivity to the Docker containers from an external location.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each container available in the target Docker server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	By default, this flag is set to **Yes**. **You are advised not to change the default setting of this flag.**
PacketSize	Here, specify the size of the packets used for the test.
PacketCount	Here, specify the number of packets to be transmitted during the test.
Timeout	Specify the maximum time (in seconds) that the test should wait for the response to a transmitted packet. A response received after the TIMEOUT period is ignored by the test. The default timeout period is 100 seconds.
PacketInterval	Represents the interval (in milliseconds) between successive packet transmissions during the execution of the test for a specific target.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Network availability of container	Indicates whether/not the network connection to this container is available.	Percent	A value of 100 indicates that this container is accessible over the network. The value 0 indicates that the container is inaccessible. Typically, the value 100 corresponds to a Packet loss of 0.
Avg network delay	Indicates the average delay between transmission of packets to this container and receipt of the response to the packet at the source.	Secs	Ideally, the value of this measure should be low.
Min network delay	The minimum time between transmission of a packet and receipt of the response back.	Secs	A significant increase in the minimum round-trip time is often a sure sign of network congestion.

Measurement	Description	Measurement Unit	Interpretation
Packet loss	Indicates the percentage of packets lost during transmission from source to target and back.	Percent	Packet loss is often caused by network buffer overflows at a network router or by packet corruptions over the network. The detailed diagnosis for this measure provides a listing of routers that are on the path from the external agent to target server, and the delays on each hop. This information can be used to diagnose the hop(s) that could be causing excessive packet loss/delays.

3.2.4 Docker Containers - Performance Test

This test monitors each container available in Docker and reports the CPU utilization, I/O processing, memory related statistics such as memory utilization, paging in/paging outs, errors that were detected etc. Using this test, administrators can easily figure out processing/memory bottlenecks and rectify the same before the users complain of slow responsiveness of the containers.

Note:

- The Docker server should be of v1.5 or above.
- Remote REST API should be enabled on the Docker host. To know how to enable remote REST API, follow the procedure discussed in Section **Chapter 2**
- The eG agent should be 'allowed' to make remote REST API calls to pull metrics. For this purpose, make sure you configure this test with the credentials of a user who has permissions to connect to REST API and invoke its methods.
- Make sure that the **WEBPORT** parameter of this test is configured with the port on which remote REST API has been enabled.

Target of the test : A Docker server

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each container available in the Docker server being monitored.

Configurable parameters for the test

Parameter	Description
Test Period	How often should the test be executed.
Host	The IP address of the host for which this test is to be configured.
Port	The port number at which the specified HOST listens. The default is 2375.
UseSUDO	**This flag is not applicable to this test.**
Docker User and Docker Password	Specify the credentials of the user who is authorized to access the remote REST API and invoke its methods for metrics collection.
Confirm Password	Confirm the Password by retyping it here.
Webport	By default, the remote REST API is enabled on port 2375. This implies that by default, this test connects to port 2375 to access the remote REST API and make API calls for metrics collection. In some environments however, the remote REST API can be enabled on a different port. To know how to enable the remote REST API on a different port, follow the procedure discussed in Section **Chapter 2** Make sure you configure this parameter with the exact port on which the remote REST API has been enabled. To know which port number that is, do the following: • Open the /lib/system/system/docker.service file on the Docker host. • In the file, find the line which starts with **ExecStart**. In that line, look for the following entry: *-H=tcp://0.0.0.0:<Remote_API_Port>* The number that appears after ':' in the entry above, is the remote REST API port.
Detailed Diagnosis	To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option. The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled: • The eG manager license should allow the detailed diagnosis capability • Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Disk space usage	Indicates the amount of disk space utilized by this container.	MB	
Data received rate	Indicates the rate at which the data was received by this container.	Mbps	A sudden increase or decrease in the value of this measure could be a cause of concern.
Incoming traffic	Indicates the rate at which packets were received by this container.	Pkts/sec	A significant increase or decrease in the value of this measure may alter traffic condition in the Docker server.
Errors received	Indicates the number of errors occurred while data was received by this container.	Number	Ideally, the value of this measure should be zero.
Packets dropped during reception	Indicates the number of packets dropped by this container during reception.	Number	Ideally, the value of this measure should be zero.
Data transmit rate	Indicates the rate at which the data was transmitted by this container.	Mbps	
Outgoing traffic	Indicates the rate at which packets were transmitted by this container.	Pkts/sec	
Errors transmitted	Indicates the number of errors occurred while data was transmitted by this container.	Number	Ideally, the value of this measure should be zero.
Packets dropped during transmission	Indicates the number of packets dropped by this container during transmission.	Number	Ideally, the value of this measure should be zero.
CPU utilization	Indicates the percentage of CPU that is currently utilized by this container.	Percent	The detailed diagnosis of this measure lists the processes that are consuming the CPU utilized by the container. Comparing the value of this measure

Measurement	Description	Measurement Unit	Interpretation
			across the containers will enable you to accurately identify the container on which CPU-intensive applications are executing.
CPU utilization in kernelmode	Indicates the percentage of CPU utilized by this container in kernel mode.	Percent	A processor in a server has two different modes: user mode and kernel mode. The processor switches between the two modes depending on what type of code is running on the processor. Applications run in user mode, and core operating system components run in kernel mode. In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system. Crashes in kernel mode are catastrophic; hence they will halt the entire container. A high value for this measure indicates that the container is taking too much of CPU resources to execute processes in kernel mode.
CPU utilization in usermode	Indicates the percentage of CPU utilized by this container in user mode.	Percent	In User mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory. Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable. Most of the code running on the containers will execute in user mode. User mode processes communicate and use the

Measurement	Description	Measurement Unit	Interpretation
			kernel through the Kernel API and system calls.
Memory used	Indicates the amount of memory that is currently utilized by this container.	MB	
Memory limit	Indicates the maximum amount of memory that is allocated to this container.	MB	
Memory utilization	Indicates the percentage of memory utilized by this container.	Percent	The detailed diagnosis of this measure lists the processes that are consuming the memory utilized by the container. A high value for this measure indicates that the memory resources of the container is depleting alarmingly.
Memory max usage	Indicates the maximum amount of memory utilized by this container.	MB	
Active anonymous memory	Indicates the amount of anonymous memory that has been identified as active by the kernel.	MB	Anonymous memory is the large, zero-filled block of memory that is directly mapped by kernel from the anonymous memory region when large memory with ideally multiples of page sizes is required for the containers. The pages of anonymous memory do not link to any file on the disk, and are the part of program's data area or stack.
Inactive anonymous memory	Indicates the amount of anonymous memory that has been identified as inactive by the kernel.	MB	All anonymous pages are initially active. In that, some pages will be tagged as inactive when the kernel sweeps over the memory at regular intervals. Whenever the inactive pages are accessed, they are immediately retagged as active. The inactive pages will be swapped when the kernel is almost out of memory, and time comes to swap out to disk.

Measurement	Description	Measurement Unit	Interpretation
Cache memory	Indicates the amount of memory used by processes of the control group that can be associated precisely with a storage block on this container.	MB	When you read from and write to files on the disk, the amount of cache memory will increase. Size of the cache memory depends on the number of read operations and write operations performed on this container.
Active file	Indicates the amount of cache memory that has been identified as active by the kernel.	MB	Pages in the cache memory can be swapped between active and inactive states similar to the anonymous memory but the exact rules used by the kernel to move memory pages between active and inactive sets are different from the rules used for the anonymous memory. The cache memory pages can be immediately retrieved in a cheaper way when the kernel needs to reclaim memory while the anonymous pages and dirty/modified pages have to be written to disk first before retrieving process.
Inactive file	Indicates the amount of cache memory that has been identified as inactive by the kernel.	MB	
Memory mapped	Indicates the amount of memory mapped by the processes in the control group.	MB	Docker on Linux also makes use of another technology called cgroups or control groups. A key to running applications in isolation is to have them only use the resources you want. This ensures containers are good multi-tenant citizens on a host. Control groups allow Docker to share available hardware resources to containers and, if required, set up limits and constraints. For example, limiting the memory available to a specific container.
Page faults	Indicates the number of times a process in this container triggered a page fault.	Number	A page fault occurs when a process accesses a part of its virtual memory space which is nonexistent or

Measurement	Description	Measurement Unit	Interpretation
			protected. The former can happen if the process is buggy and tries to access an invalid address (it will then be sent a SIGSEGV signal, typically killing it with the famous Segmentation fault message). The latter can happen when the process reads from a memory zone which has been swapped out, or which corresponds to a mapped file: in that case, the kernel will load the page from disk, and let the CPU complete the memory access. It can also happen when the process writes to a copy-on-write memory zone: likewise, the kernel will preempt the process, duplicate the memory page, and resume the write operation on the process` own copy of the page. The major page faults are regular faults occur when the kernel actually has to read the data from disk, duplicate an existing page, or allocate an empty page.
Page major faults	Indicates the number of times a process in this container triggered a major page fault.	Number	
Page ins	Indicates the number of pages that were added to the control group of this container.	Number	
Page outs	Indicates the number of pages that were not billed to the control group of this container.	Number	
Resident set size	Indicates the amount of memory that doesn't correspond to anything on disk such as stacks and heaps, and anonymous memory maps of this	MB	

Measurement	Description	Measurement Unit	Interpretation
	container.		
Huge resident set size	Indicates the amount of maximum memory that doesn't correspond to anything on disk such as stacks and heaps, and anonymous memory maps of this container.	MB	
Swap memory	Indicates the amount of swap memory that is currently utilized by the processes in the control group of this container.	MB	An unusually high value for the swap memory can indicate a memory bottleneck.
Memory unevictable	Indicates the amount of memory that cannot be reclaimed by this container.	MB	Generally, the unevictable memory has been locked with mlock, and is often used by crypto frameworks to make sure that secret keys and other sensitive material never gets swapped out to disk.
Writeback	Indicates the amount of memory that was written back in this container.	MB	A high value is preferred for this measure. Normally, the data are first written into the cache before writing it into the memory or disk that supports caching. During idle machine cycles, the data are written from the cache into the memory or onto the disk at high speed. In this way, the Write back caches improve performance and writing speed than the normal RAM or disk.
Failures	Indicates the number of memory failures that occurred in this container.	Number	Ideally, the value of this measure should be low.
Data reads	Indicates the rate at which the data was read from this container.	Mbps	Compare the values of these measures across the containers to identify the slowest container in terms of processing read and write

Measurement	Description	Measurement Unit	Interpretation
			operations (respectively).
Data writes	Indicates the rate at which the data were written into this container.	Mbps	
Data sync	Indicates the rate at which the synchronous I/O operations were performed on this container.	Mbps	In synchronous I/O file, a thread starts a I/O operation and immediately enters a wait state until the I/O request has completed such that the I/O operations of this container are performed in one-by-one manner so as to prevent unwanted data traffic.
Data async	Indicates the rate at which the asynchronous I/O operations were performed on this container.	Mbps	A thread performing asynchronous file I/O sends an I/O request to the kernel by calling an appropriate function. If the request is accepted by the kernel, the calling thread continues processing another job until the kernel signals to the thread that the I/O operation is complete. Therefore, speed of the I/O operations will be increased.
Total data	Indicates the rate at which the total I/O operations were performed on this container.	Mbps	

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.