Monitoring RabbitMQ Cluster

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: HOW DOES EG ENTERPRISE MONITOR A RABBITMQ CLUSTER? 3

CHAPTER 3: HOW TO MONITOR A RABBITMQ CLUSTER USING EG ENTERPRISE? 8

3.1 Managing a RabbitMQ Cluster 8

3.2 Configuring Tests for the RabbitMQ Cluster 11

CHAPTER 4: MONITORING THE RABBITMQ CLUSTER 13

4.1 The RabbitMQ Servers Layer 14

4.1.1 RabbitMQ Node Garbage Collection Test 15

4.1.2 RabbitMQ Nodes Test 17

4.2 The RabbitMQ Cluster Layer 25

4.2.1 RabbitMQ Cluster Status Test 26

4.2.2 RabbitMQ Cluster Workload Test 27

4.3 The RabbitMQ Messages Layer 39

4.3.1 RabbitMQ Exchanges Test 40

4.3.2 RabbitMQ Queues Test 43

4.4 The RabbitMQ Service Layer 55

4.4.1 RabbitMQ Application Connections Test 55

4.4.2 RabbitMQ Client Connections Test 58

4.4.3 RabbitMQ Channels Test 61

ABOUT EG INNOVATIONS 69

Table of Figures

Figure 1.1: Architecture of a RabbitMQ server 1
Figure 2.1: Logging into the console as an administrator 3
Figure 2.2: The Admin console of a cluster node 4
Figure 2.3: The Admin tab page 4
Figure 2.4: Adding a user 5
Figure 2.5: The newly added user displayed in the All users section 6
Figure 2.6: Viewing virtual hosts and users assigned to them 6
Figure 2.7: Granting access permissions to the new 'monitoring' user 7
Figure 3.1: Newly discovered RabbitMQ clusters listed in the Unmanaged components list 9
Figure 3.2: Managing a RabbitMQ cluster 9
Figure 3.3: Adding a RabbitMQ Cluster to the eG Enterprise system 10
Figure 3.4: The list of unconfigured tests for the RabbitMQ cluster 11
Figure 3.5: Configuring the RabbitMQ Application Connections test 11
Figure 4.1: Layer model of the RabbitMQ Cluster 13
Figure 4.2: The tests mapped to the RabbitMQ Servers layer 14
Figure 4.3: The tests mapped to the RabbitMQ Cluster layer 25
Figure 4.4: The detailed diagnosis of the Unacknowledged messages measure 38
Figure 4.5: The detailed diagnosis of the Messages returned measure 38
Figure 4.6: The detailed diagnosis of the Disk reads measure 39
Figure 4.7: The detailed diagnosis of the Disk writes measure 39
Figure 4.8: The tests mapped to the RabbitMQ Messages layer 39
Figure 4.9: The detailed diagnosis of the Unacknowledged messages measure 54
Figure 4.10: The detailed diagnosis of the Message return rate measure 54
Figure 4.11: The detailed diagnosis of the Disk read rate measure 54
Figure 4.12: The detailed diagnosis of the Disk write rate measure 55
Figure 4.13: The tests mapped to the RabbitMQ Service layer 55
Figure 4.14: The detailed diagnosis of the Number of application connections measure 58
Figure 4.15: The detailed diagnosis of the Number of clients connected measure 60

Chapter 1: Introduction

RabbitMQ is an open source message broker software that implements the Advanced Message Queueing Protocol (AMQP). RabbitMQ server is written in the Erlang programming language.

The basic architecture of a message queue is simple:

1. Client applications called producers create messages and publishes them to an exchange on a Rabbit.
2. An exchange accepts messages from the producer application and routes them to message queues. The exchange takes the routing decision by taking different message attributes into accoun, such as routing key, depending on the exchange type.
3. Bindings have to be created from the exchange to queues. The exchange routes the message into the queues, depending on message attributes such as routing keys and bindings.
4. The messages stay in the queue until other applications, called consumers, connect to that queue and subscribe to the messages.
5. The consumer(s) subscribes to the messages.

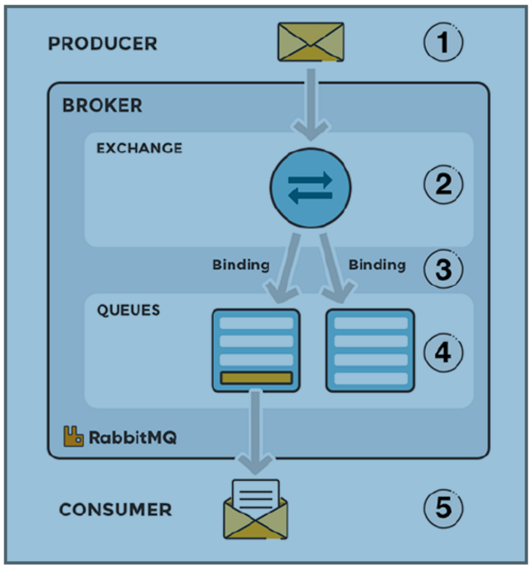
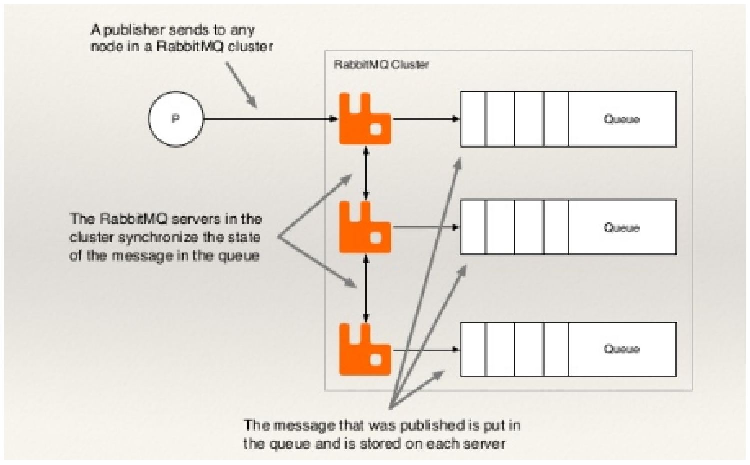


Figure 1.1: Architecture of a RabbitMQ server

A RabbitMQ Cluster connects multiple nodes to form a single logical broker. Virtual hosts, exchanges, queues, bindings, users and permissions are mirrored across all nodes in a cluster.

Typically, a publisher sends a message to any node in the cluster. The message is then added to a queue attached to that node. By default, contents of a queue within a RabbitMQ cluster are located on a single node (the node on which the queue was declared). This is in contrast to exchanges and bindings, which can always be considered to be on all nodes. Queues can optionally be made mirrored across multiple nodes. Each mirrored queue consists of one master and one or more mirrors. The master is hosted on one node commonly referred as the master node. Each queue has its own master node. All operations for a given queue are first applied on the queue's master node and then propagated to mirrors. This involves enqueueing publishes, delivering messages to consumers, tracking acknowledgements from consumers and so on.


A publisher sends to any node in a RabbitMQ cluster
RabbitMQ Cluster
P
Queue
Queue
Queue
The RabbitMQ servers in the cluster synchronize the state of the message in the queue
The message that was published is put in the queue and is stored on each server

Messages published to the queue are replicated to all mirrors. Consumers are connected to the master regardless of which node they connect to, with mirrors dropping messages that have been acknowledged at the master.

The high availability of queues and messages and the balancing of load across workers are some of the reasons why RabbitMQ clusters are widely used in mission-critical environments today. In such environments, the sudden unavailability of the cluster, irregularities in load balancing across nodes in the cluster, and slowness in delivery of messages by queues, can significantly delay inter-application communication, badly hit user productivity, and negatively impact both user experience and revenue. To avoid this, administrators must closely monitor the availability and overall performance of the RabbitMQ cluster and its core components, proactively detect anomalies, and promptly initiate measures to pre-empt them. This is where eG Enterprise helps!

eG Enterprise is capable of monitoring RabbitMQ clusters out-of-the-box. eG's intelligent agents can instantly detect and promptly alert administrators to issues in the availability, status, and performance of a cluster as a whole and also of individual nodes, exchanges, queues, and virtual hosts in the cluster. This document details how eG Enterprise monitors the RabbitMQ cluster and what metrics it reports.

Chapter 2: How Does eG Enterprise Monitor a RabbitMQ Cluster?

eG Enterprise monitors a RabbitMQ cluster in an agentless manner. An eG agent on a remote Windows host in the environment should be configured to connect to the Management Interface of the rabbitmq-management plugin of any node in the cluster. Once in, the eG agent uses the HTTP-based API provided by the plugin to run commands on the console and pull metrics of interest. To be able to connect and collect metrics using the rabbitmq-management plugin, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then feel free to configure the eG tests with the credentials of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for user creation are detailed below:

1. First, open a browser and connect to the Management Interface of any node in the cluster, using the URL: http://<NodeIP/Name>:<ManagementPort> or https://<NodeIP/Name>:<ManagementPort>/
2. Once Figure 2.1 appears, specify the credentials of an administrator in Figure 2.1 to login to the console.

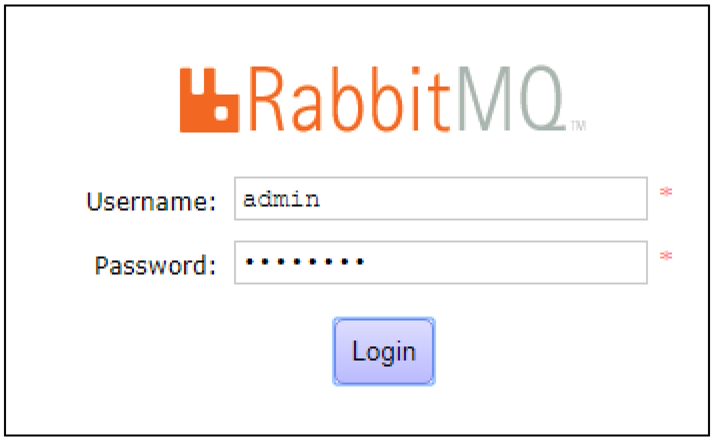
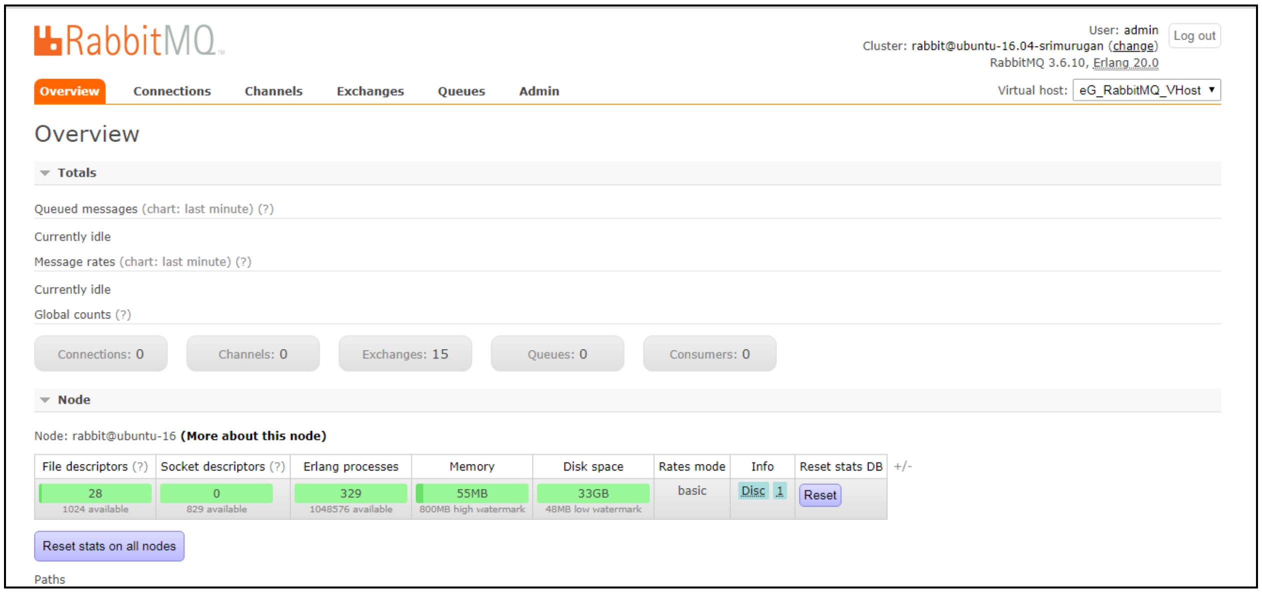

Username: admin
Password: ••••••••

Figure 2.1: Logging into the console as an administrator

3. Upon a successful login, Figure 2.2 will appear.



User: admin
Cluster: rabbit@ubuntu-16.04-srimurugan (change)
RabbitMQ 3.6.10, Erlang 20.0
Virtual host: eG_RabbitMQ_VHost

Overview

Totals

Queued messages (chart: last minute) (?)

Currently idle

Message rates (chart: last minute) (?)

Currently idle

Global counts (?)

Connections: 0 | Channels: 0 | Exchanges: 15 | Queues: 0 | Consumers: 0

Node

Node: rabbit@ubuntu-16 **(More about this node)**

File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Rates mode	Info	Reset stats DB
28 1024 available	0 829 available	329 1048576 available	55MB 800MB high watermark	33GB 48MB low watermark	basic	Disc 1	Reset

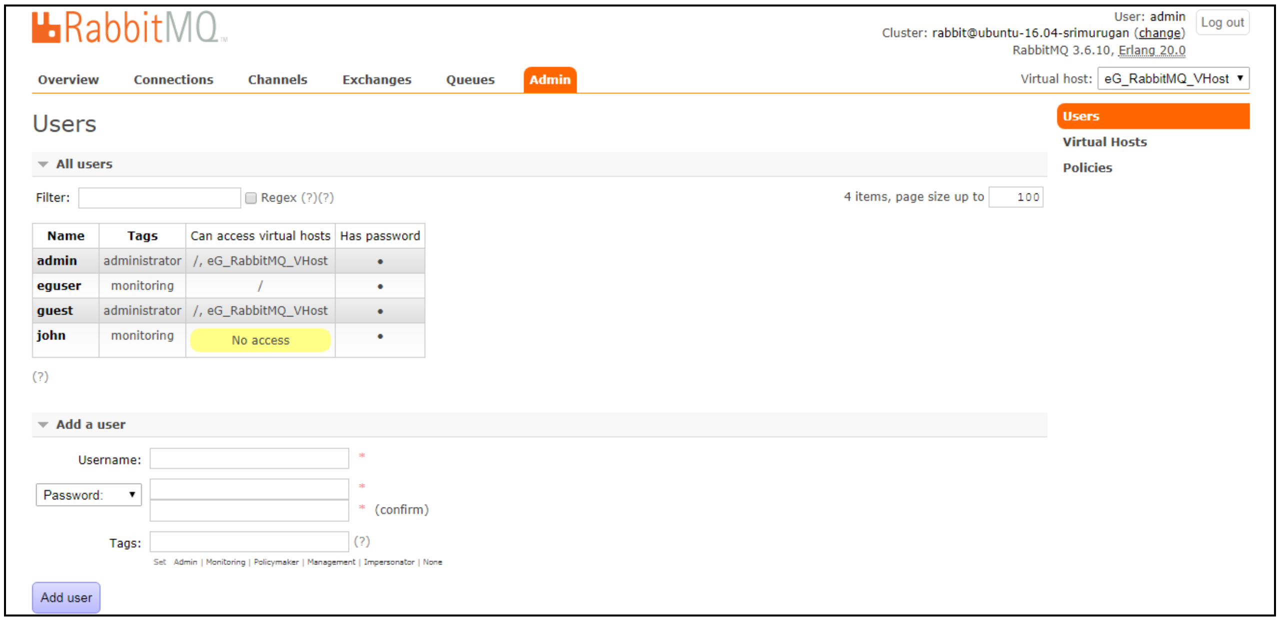
+/-

Reset stats on all nodes

Paths

Figure 2.2: The Admin console of a cluster node

4. Now, click on Admin tab page in Figure 2.2. Figure 2.3 will then appear.



User: admin
Cluster: rabbit@ubuntu-16.04-srimurugan (change)
RabbitMQ 3.6.10, Erlang 20.0
Virtual host: eG_RabbitMQ_VHost

Users

All users

Filter: Regex (?)(?)

4 items, page size up to 100

Name	Tags	Can access virtual hosts	Has password
admin	administrator	/, eG_RabbitMQ_VHost	•
eguser	monitoring	/	•
guest	administrator	/, eG_RabbitMQ_VHost	•
john	monitoring	No access	•

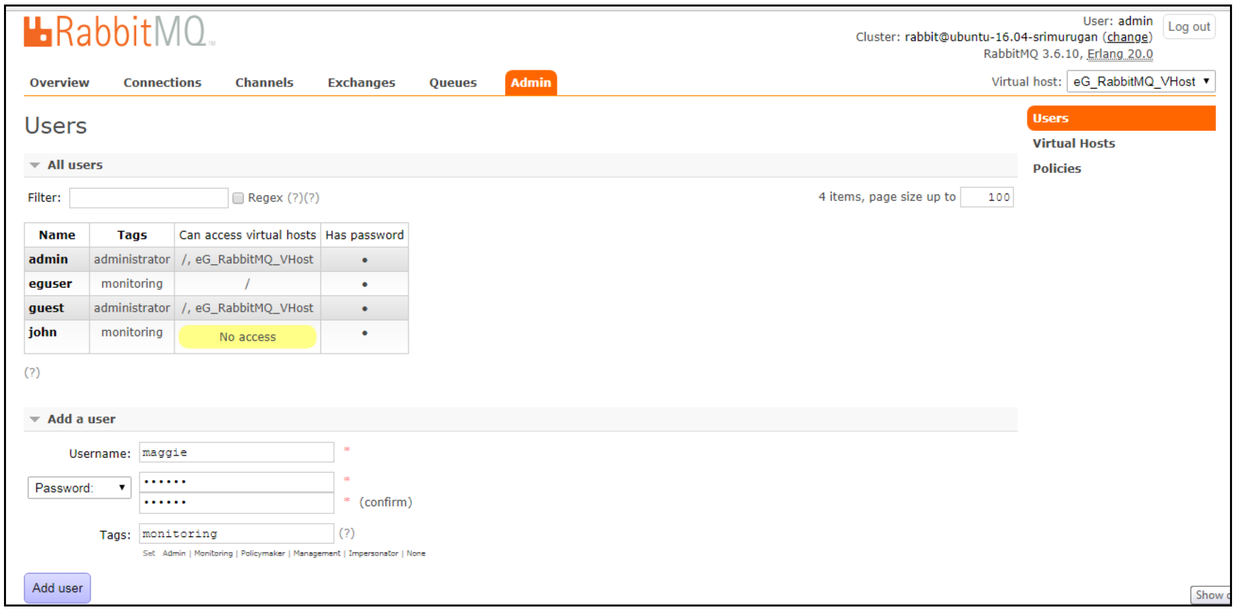
(?)

Add a user

Username: *
Password: *
* (confirm)
Tags: (?)
Set Admin | Monitoring | Policymaker | Management | Impersonator | None

Figure 2.3: The Admin tab page

5. Using the **Add a user** section of Figure 2.3, create a new user. For that, first specify the **Username** and **Password** for the new user (see Figure 2.4). Then, from the list of tags listed below the **Tags** field, select the **Monitoring** tag to assign it to the user.


RabbitMQ
User: admin
Log out
Cluster: rabbit@ubuntu-16.04-srimurugan (change)
RabbitMQ 3.6.10, Erlang 20.0
Overview
Connections
Channels
Exchanges
Queues
Admin
Virtual host: eG_RabbitMQ_VHost
Users
Virtual Hosts
Policies

Users

All users

Filter: Regex (?)(?) 4 items, page size up to 100

Name	Tags	Can access virtual hosts	Has password
admin	administrator	/, eG_RabbitMQ_VHost	•
eguser	monitoring	/	•
guest	administrator	/, eG_RabbitMQ_VHost	•
john	monitoring	No access	•

(?)

Add a user

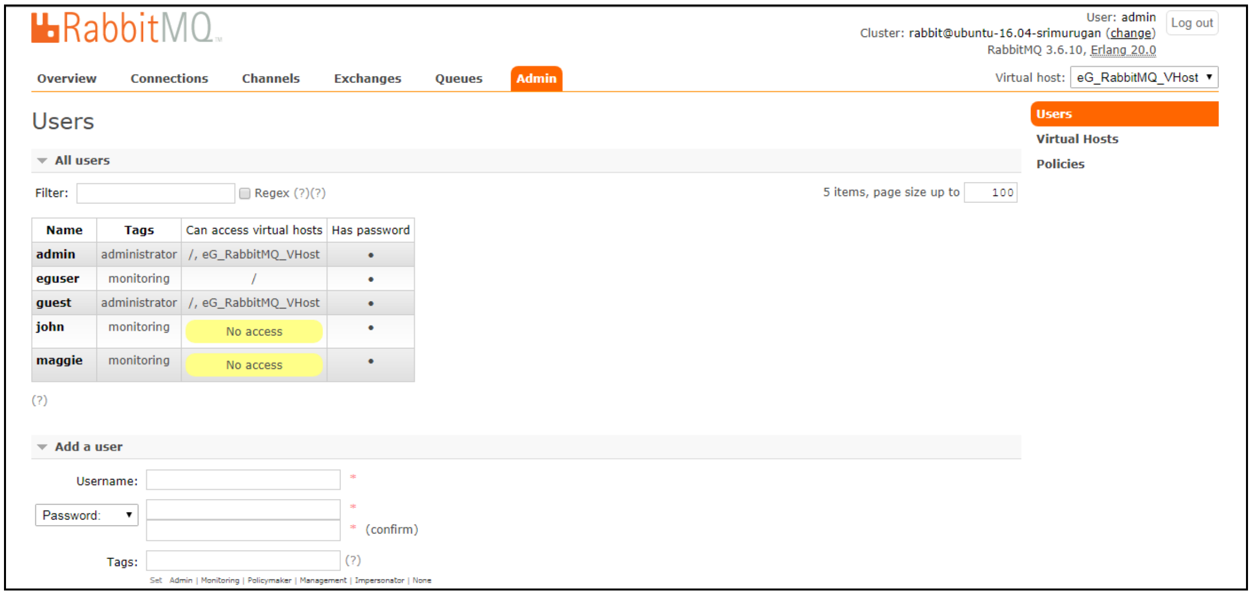
Username:	maggie *
Password:	•••••• *
	•••••• * (confirm)
Tags:	monitoring (?)
	Set Admin | Monitoring | Policymaker | Management | Impersonator | None

Add user

Figure 2.4: Adding a user

6. Then, click the **Add user** button in Figure 2.4.

7. Figure 2.5 will then appear displaying the newly created user and the tag assigned to that user. Make a note of the credentials of this user, and make sure that you specify these credentials when configuring eG tests for a target RabbitMQ Cluster.


RabbitMQ
User: admin
Log out
Cluster: rabbit@ubuntu-16.04-srimurugan (change)
RabbitMQ 3.6.10, Erlang 20.0
Overview
Connections
Channels
Exchanges
Queues
Admin
Virtual host: eG_RabbitMQ_VHost
Users
Virtual Hosts
Policies

Users

All users

Filter: Regex (?)(?) 5 items, page size up to 100

Name	Tags	Can access virtual hosts	Has password
admin	administrator	/, eG_RabbitMQ_VHost	•
eguser	monitoring	/	•
guest	administrator	/, eG_RabbitMQ_VHost	•
john	monitoring	No access	•
maggie	monitoring	No access	•

(?)

Add a user

Username: *

Password: *

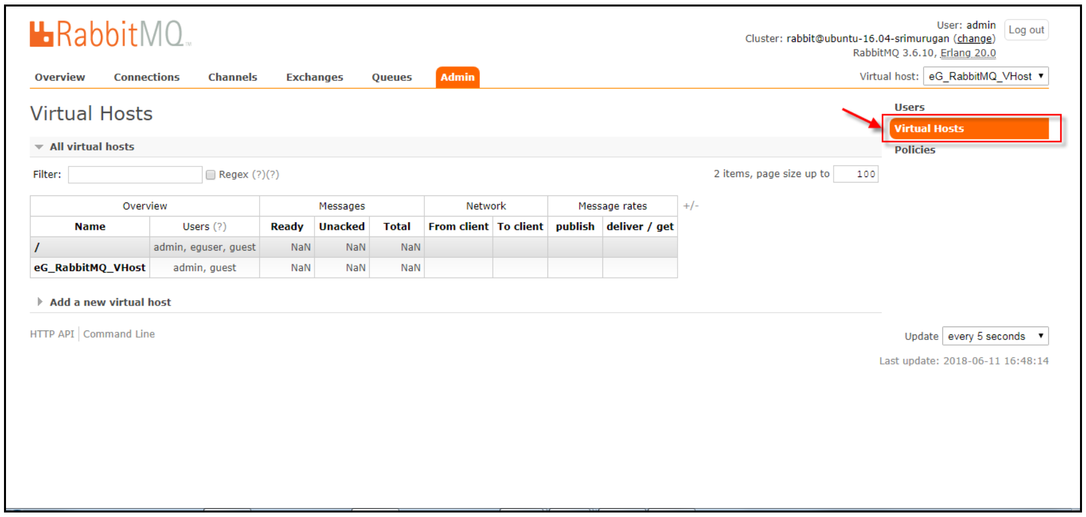
* (confirm)

Tags: (?)

Set Admin | Monitoring | Policymaker | Management | Impersonator | None

Figure 2.5: The newly added user displayed in the All users section

8. RabbitMQ requires that any new user created on a node be associated with one/more virtual hosts. Even from a monitoring standpoint, if you configure the eG tests with the credentials of a 'monitoring' user who is not assigned any virtual host, then the eG agent will not be able to run any test or collect metrics. Therefore, proceed to assign one/more virtual hosts to the 'monitoring' user you just created. For that, click on the Virtual Hosts option, indicated by Figure 2.6. You will then be able to view the virtual hosts that pre-exist and the users who can access them (see Figure 2.6).



Virtual Hosts

All virtual hosts

Filter: Regex (?)(?) 2 items, page size up to 100

Overview		Messages			Network		Message rates	
Name	Users (?)	Ready	Unacked	Total	From client	To client	publish	deliver / get
/	admin, eguser, guest	NaN	NaN	NaN				
eG_RabbitMQ_VHost	admin, guest	NaN	NaN	NaN				

+/-

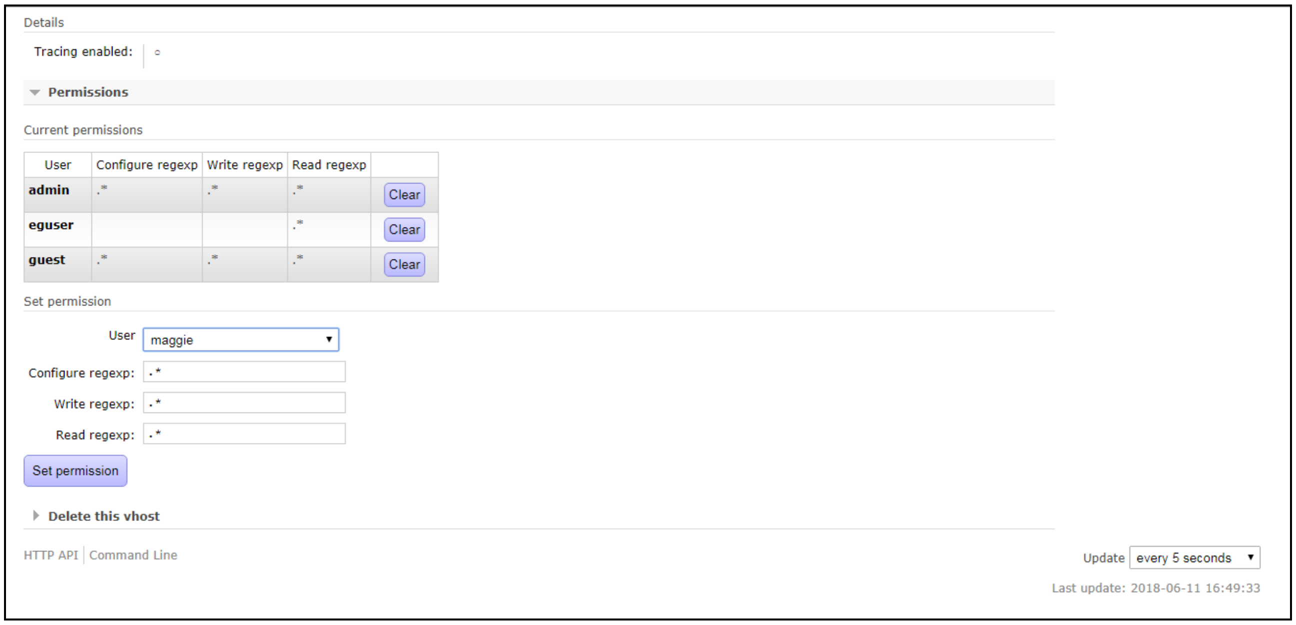
Add a new virtual host

Update every 5 seconds

Last update: 2018-06-11 16:48:14

Figure 2.6: Viewing virtual hosts and users assigned to them

9. In the list of virtual hosts in Figure 2.6, click on the virtual host to which you want to assign the new user.
10. Next, keep scrolling down the Virtual Hosts page (depicted by Figure 2.6) until you view the **Permissions** section (see Figure 2.7). Under **Set Permission** in Figure 2.7, select the name of the new **User**. To grant the Configure, Read, and Write permissions to the new user, enter .* against the **Configure regexp**, **Write regexp**, and **Read regexp** text boxes. Finally, click on the **Set permission** button therein.



User	Configure regexp	Write regexp	Read regexp	
admin	.*	.*	.*	Clear
eguser			.*	Clear
guest	.*	.*	.*	Clear

User	maggie
Configure regexp:	.*
Write regexp:	.*
Read regexp:	.*

Figure 2.7: Granting access permissions to the new 'monitoring' user

Chapter 3: How to Monitor a RabbitMQ Cluster Using eG Enterprise?

Once the 'monitoring' user is created and assigned to a virtual host, proceed to monitor the RabbitMQ cluster using eG Enterprise. The broad steps in this regard are as follows:

1. Manage the RabbitMQ cluster using the eG admin interface
2. Configure tests for the RabbitMQ cluster

Each of these steps have been elaborately discussed in the sections that will follow :

3.1 Managing a RabbitMQ Cluster

eG Enterprise is capable of automatically discovering a RabbitMQ cluster using a port-based scanning technique. To manage an auto-discovered RabbitMQ cluster, follow the steps below:

1. Login to the eG admin interface as admin with password admin.
2. Follow the Infrastructure -> Components -> Manage/Unmanage/Delete menu sequence in the Admin home page.
3. Figure 1 will then appear. Select RabbitMQ Cluster as the Component type in Figure 1. All auto-discovered RabbitMQ clusters will be displayed in the Unmanaged Components list, with an asterisk (*) symbol alongside. This 'asterisk' denotes that the RabbitMQ cluster has been 'newly' discovered by eG Enterprise. To manage a cluster, select it from the Unmanaged Components list and click the < button in Figure 1. This will transfer the selection to the Managed Components list (see Figure 2). Finally, click the Update button .

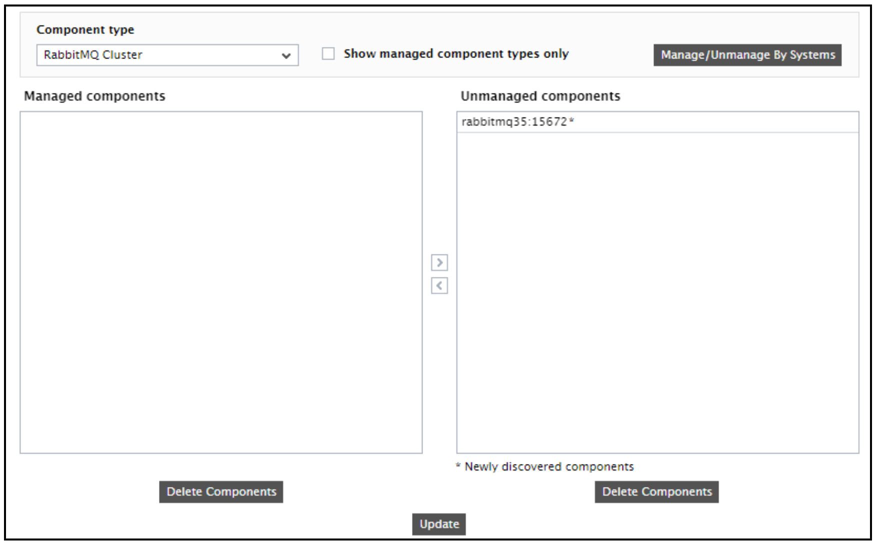


Figure 3.1: Newly discovered RabbitMQ clusters listed in the Unmanaged components list

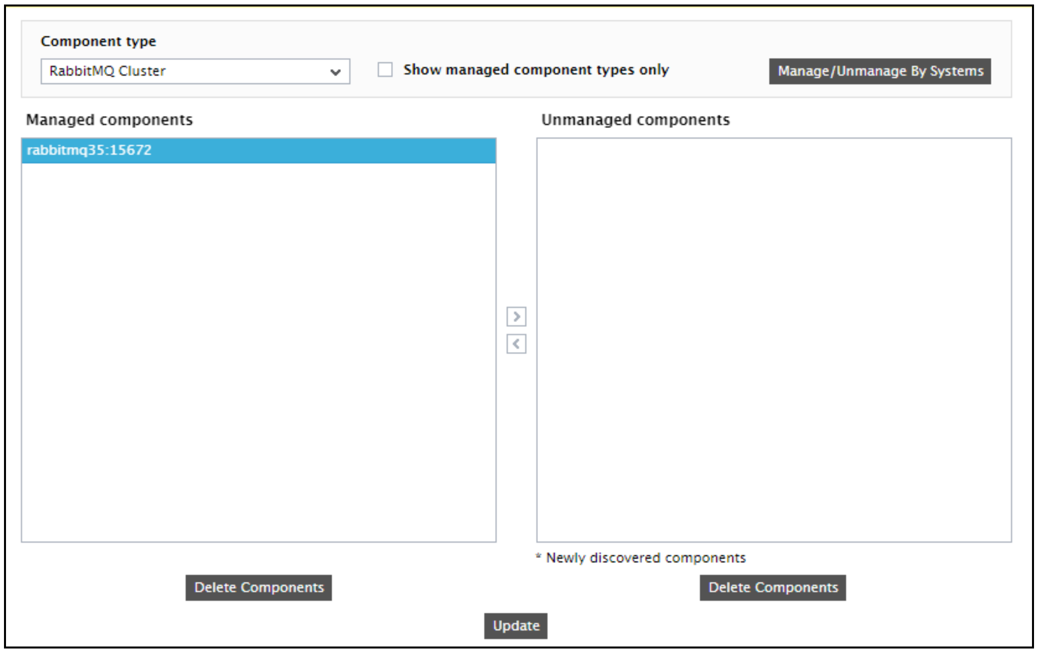


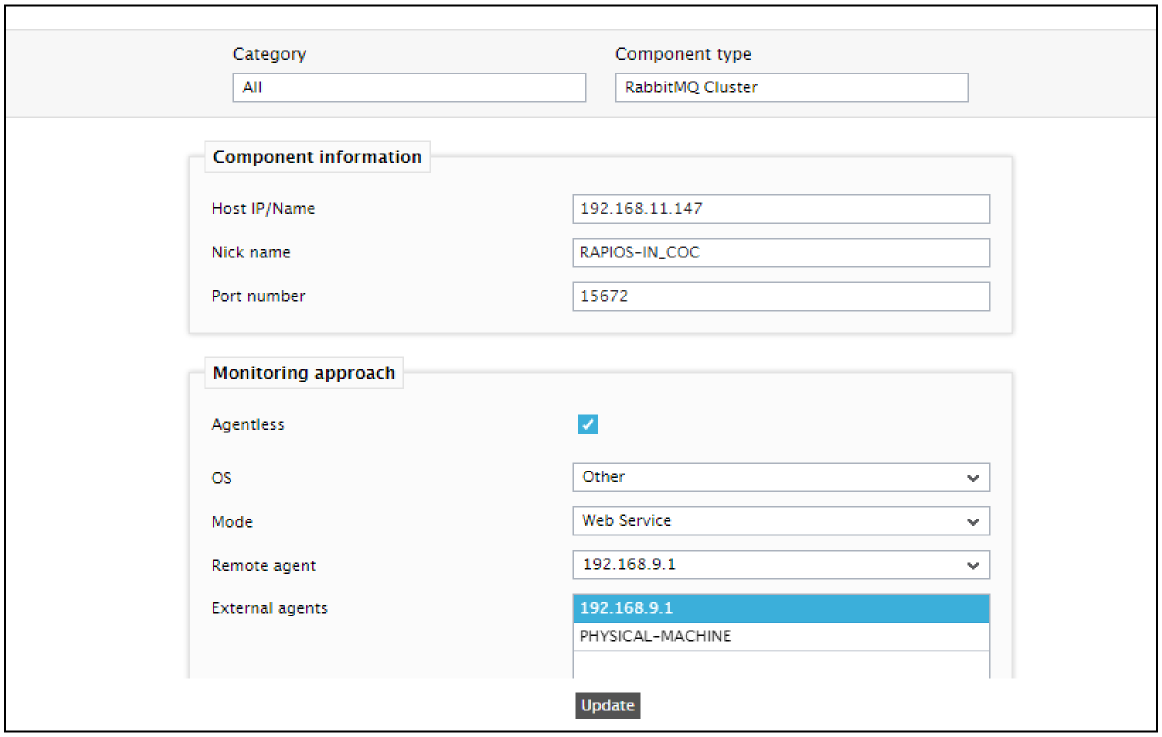
Figure 3.2: Managing a RabbitMQ cluster

If for some reason, eG Enterprise is unable to auto-discover the RabbitMQ cluster, then, you will have to manually add the cluster to the eG Enterprise system. For this, follow the steps below:

1. Login to the eG admin interface as admin with password admin.
2. Follow the menu sequence Infrastructure -> Components -> Add/Modify in the Admin Home page.

3. In the page that appears next, select RabbitMQ Cluster as the Component type and click the Add New Component button therein.

4. Figure 3 will then appear.



Category	All
Component type	RabbitMQ Cluster

Component information

Host IP/Name	192.168.11.147
Nick name	RAPIOS-IN_COC
Port number	15672

Monitoring approach

Agentless	✓
OS	Other
Mode	Web Service
Remote agent	192.168.9.1
External agents	192.168.9.1 PHYSICAL-MACHINE

Figure 3.3: Adding a RabbitMQ Cluster to the eG Enterprise system

5. In Figure 3, provide the Host IP/Name of any node in the cluster. Assign a Nick name to the cluster.

6. Since the RabbitMQ Cluster can be monitored in an Agentless manner alone, the Agentless flag will be enabled by default.

7. Set Other as the OS and Web Service as the Mode.

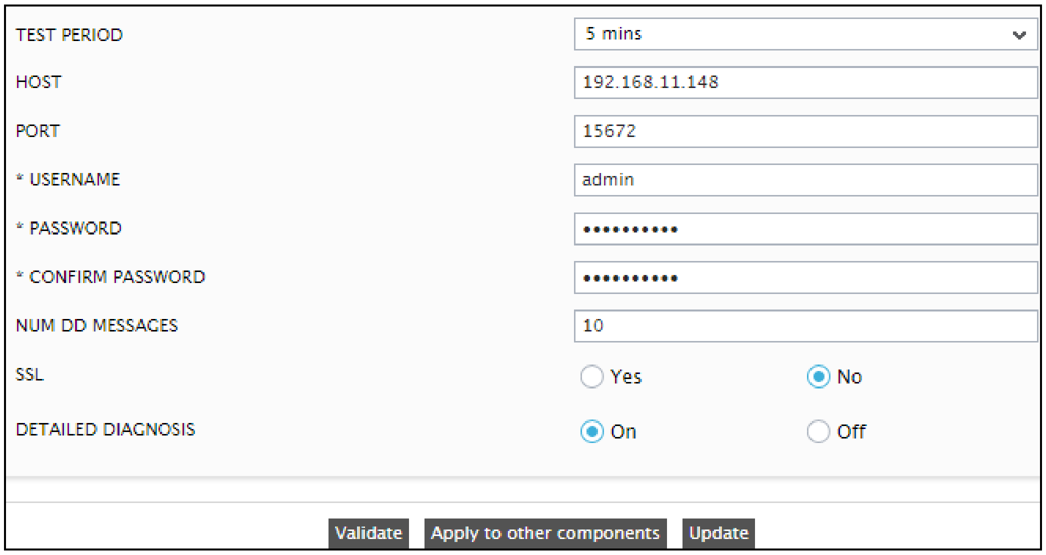
8. Then, pick the Remote Agent that should do agentless monitoring of the cluster.

9. Next, assign a External Agent to the cluster.

10. Finally, click the Add button to add the cluster to the eG Enterprise system. Components manually added will be automatically managed by eG Enterprise.

3.2 Configuring Tests for the RabbitMQ Cluster

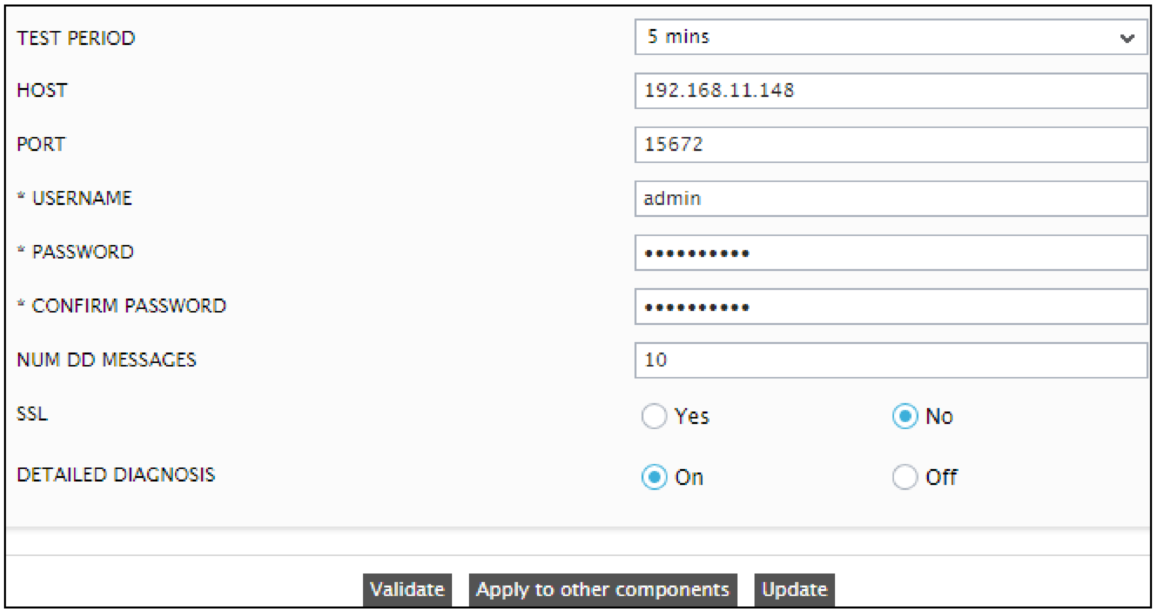
Once the RabbitMQ cluster is managed, sign out of the eG admin interface. This will invoke Figure 1, listing all the unconfigured tests for the cluster.



TEST PERIOD	5 mins
HOST	192.168.11.148
PORT	15672
* USERNAME	admin
* PASSWORD	••••••••••
* CONFIRM PASSWORD	••••••••••
NUM DD MESSAGES	10
SSL	Yes [✓ No]
DETAILED DIAGNOSIS	[✓ On] Off

Figure 3.4: The list of unconfigured tests for the RabbitMQ cluster

Click on any test to configure it. Say, you click on the RabbitMQ Application Connections test. Doing so will invoke Figure 2.



TEST PERIOD	5 mins
HOST	192.168.11.148
PORT	15672
* USERNAME	admin
* PASSWORD	••••••••••
* CONFIRM PASSWORD	••••••••••
NUM DD MESSAGES	10
SSL	Yes [✓ No]
DETAILED DIAGNOSIS	[✓ On] Off

Figure 3.5: Configuring the RabbitMQ Application Connections test

To know how to configure this test, refer to the RabbitMQ Application Connections Test topic.

Once the test is configured, sign out of the eG admin interface.

Chapter 4: Monitoring the RabbitMQ Cluster

Now that the RabbitMQ cluster s monitoring- ready, proceed to view the performance results reported by the eG agent for the RabbitMQ cluster. For that, login to the eG user interface as any user with monitoring rights to the cluster.

eG Enterprise provides a specialized monitoring model for the RabbitMQ cluster.

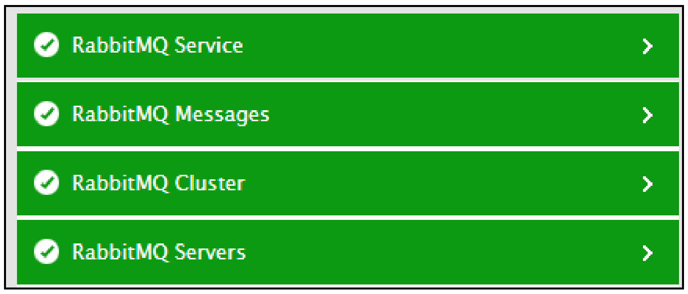


Figure 4.1: Layer model of the RabbitMQ Cluster

Each layer of Figure 4.1 is mapped to tests that report on the health of the cluster, the servers in the cluster, and the message queues, channels, and client connections to the cluster. Using the metrics reported by these tests, administrators can find quick and accurate answers to certain persistent performance queries, such as the following:

- Are any nodes in the cluster not running presently? If so, which nodes are these?
- Is any node in the cluster consuming file and/or socket descriptors abnormally? Which node is this?
- Is any node consuming memory excessively? Which node is this?
- Are Erlang processes been over-utilized by any node? Which node is this?
- Is any node's bandwidth usage unusually high?
- Are disk IOPS abnormally high on any node?
- Has garbage collection occurred very frequently on any node? Which node is it? How much heap memory was reclaimed during the garbage collection on that node?
- What is the current message load on the cluster?
- Which type of messages are being delivered slowly - messages to consumers requiring acknowledgement? or messages to consumers not requiring acknowledgement?

- Are too many messages delivered to consumers using manual acknowledgement? If so, which queue delivered such messages the most?
- Are any messages being redelivered?
- Have any messages been returned?
- Are too many messages read from and written to disks every second? If so, which queue is generating this high level of disk activity?
- Is any queue abnormally long in terms of number of messages it contains?
- Are there any idle queues?
- Is any queue consuming memory excessively? What could be causing the memory drain - a large number of unacknowledged messages in the queue?
- How are the virtual hosts the configured user has access to, performing? Is any virtual host handling too many unacknowledged messages? Which virtual host is seeing an unusual number of redelivered and returned messages?
- Over which application and client connection is the too much data transmitted?
- What type of exchanges are operational? Is any exchange processing messages published to it slowly?
- What is the overall connection and channel load on the cluster? Which connection/channel is imposing the maximum load in terms of the number of reductions invoked?
- Are consumers of any channel prefetching messages at an abnormal rate? Could this be owing to an improper Prefetch count setting?

4.1 The RabbitMQ Servers Layer

With the help of the tests mapped to this layer, you can monitor the state, resource usage, and garbage collection activity on each of the Erlang nodes in your cluster.

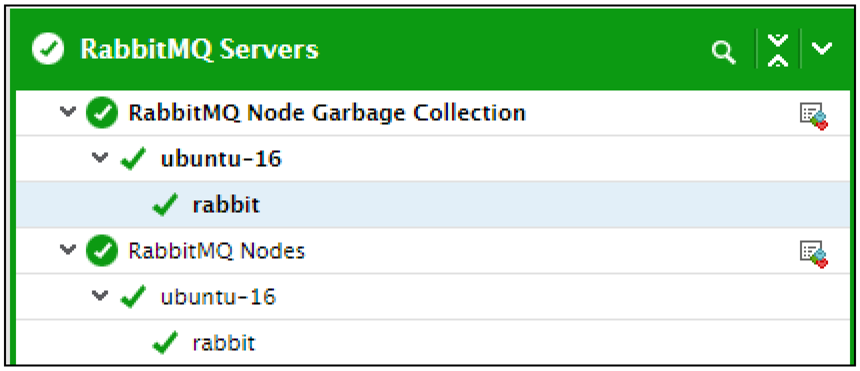


Figure 4.2: The tests mapped to the RabbitMQ Servers layer

4.1.1 RabbitMQ Node Garbage Collection Test

RabbitMQ server is written in the Erlang programming language. Each Erlang process has its own stack and heap which are allocated in the same memory block and grow towards each other. When the stack and the heap meet, the garbage collector is triggered and memory is reclaimed. If the garbage collector does not reclaim enough memory, the heap will grow to accomodate more data. If heap growth is not controlled by efficient garbage collection, it can degrade the performance of the RabbitMQ node, and consequently, slowdown cluster operations as well.

Using the RabbitMQ Node Garbage Collection test, you can keep tabs on garbage collection activity on each node of a cluster and identify the node from which the least memory was reclaimed. When a cluster under-performs, you can use this test to figure out if the dip in cluster performance is owing to excessive heap growth on a node caused by inefficient garbage collection.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each node in the monitored RabbitMQ Cluster

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Garbage collects	Indicates the number of garbage collection operations that occurred on this node during the last measurement period.	Number	Compare the value of this measure across nodes to identify the node on which garbage collection has happened very often. Such a node could have experienced rapid and abnormal heap growth, thus triggering garbage collection frequently. You may want to investigate the reasons for heap growth on that node. Typically, heaps grow in two stages, first a variation of the Fibonacci sequence is used starting at 233 words. Then at about 1 mega words the heap only grows in 20% increments. There are two occasions when the heap grows: • If the total size of the heap + message and heap fragments exceeds the current heap size; • If after a fullsweep, the total amount of live objects is greater than 75% Either way, you may want to resize the heap to avoid frequent garbage collections. This is because, every time garbage collection happens, the garbage collector must suspend the execution of the node to ensure the integrity of the object trees. The more live objects are found, the longer the suspension, which has a direct impact on response time and throughput. This in turn may impact overall cluster performance as well.
Garbage collects	Indicates the rate at which	Operations/Sec	A high value is indicative of frequent

Measurement	Description	Measurement Unit	Interpretation
data reclaimed	garbage collections take place on this node.		garbage collections. This could be owing to rapid and significant heap growth on the node. Frequent garbage collections on a node may degrade its performance. To avoid this, you may want to consider resizing the heap on that node.
Data reclaimed	Indicates the memory reclaimed by the garbage collector on this node.	MB	Compare the value of this measure across nodes to know on which node the garbage collector reclaimed the maximum memory and on which node the least memory was reclaimed.
Data reclamation rate	Indicates the rate at which memory was reclaimed by the garbage collector on this node.	MB/Sec	
Context switch operations	Indicates the rate at which runtime context switching takes place on this node.	Operations/Sec	

4.1.2 RabbitMQ Nodes Test

A RabbitMQ cluster is a logical grouping of one or several Erlang nodes, each running the RabbitMQ application and sharing users, virtual hosts, queues, exchanges, bindings, and runtime parameters.

A client can connect to any node and perform any operation. Nodes will route operations to the queue master node transparently to clients. In case of a node failure, clients will be able to reconnect to a different node, recover their topology and continue operation. Regardless of which node is serving client requests, at any point in time, administrators should be able to tell the operational state of each node in the cluster, so that the failed nodes can be identified.

Moreover, client connections, channels, and queues are distributed across cluster nodes. This means that all nodes in a cluster should be sized with adequate resources such as memory, bandwicth, disk space, file/socket handlers, and Erlang processes. Administrators should be able to track the usage of these critical resources on each node, and pinpoint the node that is under-sized.

Additionally, administrators should observe the reads from and writes to the queue index journals, message store, and disk of each node to gauge the level of activity on each node and measure a node's ability to handle these activity levels.

The RabbitMQ Nodes test enables administrators to perform all the above! This test auto-discovers the nodes in a target cluster. For each node, the test then reports the state of that node, its uptime, and how its memory, file descriptors, socket descriptors, bandwidth resources and Erlang processes have been utilized. Nodes that are down and those that are running out of resources are revealed in the process. Furthermore, the test reports the rate at which reads, writes, seeks, and syncs were performed on the disk of each node, thus revealing the I/O processing ability of each node. The time taken by every node to perform these I/O operations is also reported, so that latent nodes can be identified.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each node in the monitored RabbitMQ Cluster

Configurable parameters for the test

Parameters	**Description**
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Status	Indicates the current state of this node.	Number	The values that this measure can report and their corresponding numeric values are listed in the table below: **Measure Value** — **Numeric Value** Running — 1 Stopped — 0 **Note:** This test reports the **Measure Value**s listed in the table above to indicate the current operational state of a node. In the graph of this measure however, the same will be represented using the numeric equivalents.
Uptime	Indicates the uptime of this node (in days).	Days	Compare the value of this measure across nodes to identify the node that has been down for the longest time.
Maximum file descriptors	Indicates the maximum number of file descriptors that this node can use.	Number	By default, a node can use upto a maximum of 1024 file descriptors. A file descriptor (FD, less frequently fildes) is an abstract indicator (handle) used to access a file or other input/output resource, such as a pipe or network socket.
Used file descriptors	Indicates the number of file descriptors that have been utilized on this node.	Number	This count includes both file and socket descriptors. A file descriptor (FD, less frequently fildes) is an abstract indicator (handle) used to access a file or other input/output resource, such as a pipe or network socket.

Measurement	Description	Measurement Unit	Interpretation
			A socket is an abstraction of a communication endpoint. Just as they would use file descriptors to access a file, applications use socket descriptors to access sockets. Socket descriptors are implemented as file descriptors in the UNIX System.
File descriptor usage	Indicates what percentage of the maximum number of file descriptors configured for this node is currently in use.	Percent	A value close to 100% is a cause for concern, as it indicates that the node is running out of file handles. If the value reaches 100%, then the node will block all incoming connections. To avoid this, you may want to consider increasing the maximum file descriptor configuration of the node.
Maximum socket descriptors	Indicates the maximum number of socket descriptors that this node can use.	Number	A socket is an abstraction of a communication endpoint. Just as they would use file descriptors to access a file, applications use socket descriptors to access sockets.
Used socket descriptors	Indicates the number of socket descriptors this node is using.	Number	
Socket descriptors usage	Indicates what percentage of the maximum number of socket descriptors configured for this node is presently in use.	Percent	A value close to 100% is a cause for concern, as it indicates that the node is running out of socket descriptors. If a node exhausts socket descriptors, then that node will block all incoming connections. To avoid this, you may want to consider increasing the maximum socket descriptor configuration of the node.
Maximum erlang processes	Indicates the maximum number of Erlang processes configured for	Number	

Measurement	Description	Measurement Unit	Interpretation
	this node.		
Used erlang processes	Indicates the number of Erlang processes that this node is currently using.	Number	
Erlang process usage	Indicates what percentage of the maximum number of Erlang processes configured for this node is currently in use.	Percent	Queues, connections, and channels are the main components of RabbitMQ that consume processes. This means, higher the number of queues, connections, and channels, higher will be the usage of Erlang processes. If the value of this measure is close to 100% for a node, it implies that that node is running out of Erlang processes. This could cause databases to hang and messages to start piling up on RabbitMQ. To avoid this, you may want to consider increasing the maximum number of Erlang processes configured for the node.
Maximum memory	Indicates the maximum amount of memory this node can use.	MB	
Used memory	Indicates the amount of memory in use on this node.	MB	
Memory usage	Indicates the percentage of memory that this node is using.	Percent	A value close to 100% indicates that the node is running out of memory. If the situation is allowed to persist, then the node may soon exhaust its memory completely. This could bring messaging operations to a standstill. At this juncture, you can start by looking at some of the common memory consumers on a node are as follows:

Measurement	Description	Measurement Unit	Interpretation
			• Connections • Channels • Queue masters, indices, and messages kept in memory • Queue mirrors, indices, and messages kept in memory • Binaries containing message bodies and metadata • Plugins • Mnesia tables and other ETS tables that keep an in-memory copy of their data • Memory used by code and atoms If any of the afore-mentioned factors are large in number, then memory consumption too is likely to increase. In such a situation, to conserve memory, see if you can control the memory consumption of the aforesaid components by decreasing their count. For instance, see if you can optimize the number of channels your applications typically use and bring that number down. Alternatively, you may want to consider increasing the memory allocation to the node.
RAM-only Mnesia transactions	Indicates the rate at which RAM-only Mnesia transactions take place on this node.	Transactions/Sec	An Mnesia transaction is a mechanism by which a series of database operations can be executed as one functional block.

Measurement	Description	Measurement Unit	Interpretation
			If the transaction is performed on data stored exclusively in memory, it is a RAM-only Mnesia transaction. An example of such a transaction is creation/deletion of transient queues.
Disk Mnesia transactions	Indicates the rate at which Mnesia transactions take place on this node's disk.	Transactions/Sec	If the transaction is performed on data stored in disk, it a disk transaction. An example of such a transaction is creation/deletion of durable queues.
Index journals	Indicates the rate at which message information is written to queue index journals on this node.	Messages/Sec	Each record in a queue index journal represents a message being published to a queue, being delivered from a queue, and being acknowledged in a queue. If the value of this measure keeps increasing consistently, it could indicate that one/more queues reside on the target node and that there is a high level of messaging activity on that node.
Store reads	Indicates the rate at which messages are read from the message store on this node.	Messages/Sec	Messages (the body, and any properties and / or headers) can either be stored directly in the queue index, or written to the message store. The message store is a key-value store for messages, shared among all queues in the server. There are technically two message stores (one for transient and one for persistent messages) but they are usually considered together as "the message store". If the value of the *Store writes* measure increases consistently, it could indicate the presence of many

Measurement	Description	Measurement Unit	Interpretation
			lazy queues on the node with many messages. To accommodate all these messages in the message store, the node will have to be sized with sufficient disk space and file descriptors.
Store writes	Indicates the rate at which messages are written to the message store on this node.	Messages/Sec	
Index reads	Indicates the rate at which segment files are read from the queue index on this node.	Messages/Sec	The queue index is responsible for maintaining knowledge about where a given message is in a queue, along with whether it has been delivered and acknowledged. There is therefore one queue index per queue. If the values reported by these measures are consistently high for a node, it could mean that one/more queues reside on the node and many messages are stored in the queues.
Index writes	Indicates the rate at which segment files are read from written to the queue index on this node.	Messages/Sec	
Disk reads	Indicates the rate at which read operations are performed on the disk on this node.	IOPS	Compare the values of these measures across nodes to identify the node that is experiencing a high level of disk activity. In the process, you can identify which type of I/O operations are very common on that node - reads? writes? seeks? or syncs?
Disk writes	Indicates the rate at which write operations are performed on the disk on this node.	IOPS	
Disk seeks	Indicates the rate at which seek operations are performed on the disk on this node. A seek operation happens when the disk physically locates a piece of data on it, when reading/writing.	IOPS	
Disk syncs	Indicates the rate at which the node invokes fsync() to ensure data is flushed to the disk.	IOPS	

Measurement	Description	Measurement Unit	Interpretation
Read bandwidth	Indicates the rate at which data is read by this node.	MB/Sec	These measures are good indicators of the bandwidth used by a node when reading/writing. You can compare the value of this measure across nodes to know which node is consuming maximum bandwidth.
Write bandwidth	Indicates the rate at which data is written by this node.	MB/Sec	
Disk read time	Indicates the average time taken by this node to perform a read operation.	Millisecs	Compare the values of these measures across nodes to identify the most latent node. In the process, you can identify which type of I/O operations are taking the longest on each node - reads? writes? seeks? or syncs?
Disk write time	Indicates the average time taken by this node to perform a write operation.	Millisecs	
Disk seek time	Indicates the average time taken by this node to perform a seek operation.	Millisecs	
Disk sync time	Indicates the average time taken by this node to complete an fsync().	Millisecs	

4.2 The RabbitMQ Cluster Layer

Using the tests mapped to this layer, you can be:

- Proactively alerted to a sudden change in the operational state of the nodes in your cluster;
- Consistent rise in the workload of your cluster;
- Problems in the functioning of your virtual hosts

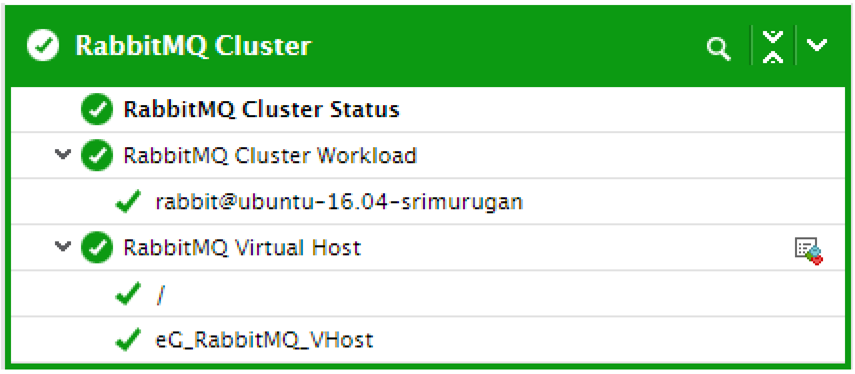


Figure 4.3: The tests mapped to the RabbitMQ Cluster layer

4.2.1 RabbitMQ Cluster Status Test

Typically, at least one cluster node should be up and running at a given point in time, to keep the cluster alive. Client applications will be unable to access a RabbitMQ cluster if all its nodes are down. To ensure high cluster availability therefore, administrators should keep an eye on the running state of every node in the cluster and promptly identify the nodes that are not running, so that they can rapidly initiate measures to start the nodes that are down. This is where the RabbitMQ Cluster Status test helps!

This test monitors the status of nodes in a cluster, and reports the count of nodes that are running and those that are down. Also, the test promptly notifies administrators if even a single node is rendered unavailable. Detailed diagnostics provided by this test reveal the name of the unavailable node(s), thus enabling administrators to start that node(s) and ensuring cluster availability.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for the cluster being monitored

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.

Parameters	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 enable/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 nodes	Indicates the total number of nodes in the cluster.	Number	
Running nodes	Indicates the number of nodes in the cluster that are currently running.	Number	Ideally, the value of this measure should be equal to that of the Total nodes measure.
Stopped nodes	Indicates the number of nodes in the cluster that are not running presently.	Number	Ideally, the value of this measure should be 0. If this measure reports a non-zero value, then use the detailed diagnosis of the measure to know which nodes in the cluster are not running.

4.2.2 RabbitMQ Cluster Workload Test

The workload of a RabbitMQ cluster is defined by the messages it receives from publishers and delivers to consumers, the connections, exchanges, and channels it handles, and the message read-write activity on its disk. Understanding the current workload of a cluster and measuring how well the cluster handles the workload is important to figure out if the cluster has been sized and configured right to perform well at peak load. The RabbitMQ Cluster Workload test enables administrators to gain this understanding.

This test tracks the messages published to and delivered by the cluster, measures the rate at which the cluster processes the messages, and thus reveals bottlenecks (if any) in message processing. Additionally, the test reports the count of connections, exchanges, and channels on the cluster, and also reports the level of I/O activity on the cluster disks. The test also reveals the count of unacknowledged and redelivered messages, and thus enables administrators evaluate how such messages may impact cluster performance. This way, the test helps administrators gauge the current workload on the cluster, understand the present processing power of the cluster, and thus figure out how the cluster size/configuration can be tweaked to enhance its processing ability.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for the monitored RabbitMQ Cluster

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.
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.

Parameters	Description
	The option to selectively enable/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
Ready messages	Indicates the number of messages that are available to be delivered now.	Number	Use the detailed diagnosis of this measure to receive an overview of the RabbitMQ cluster setup. The RabbitMQ version, management version, and Erlang version will be displayed as part of the detailed diagnostics.
Unacknowledged messages	Indicates the number of messages for which the cluster is waiting for acknowledgement.	Number	A low value is desired for this measure. This is because, all unacknowledged messages have to reside in RAM on the servers. If you have too many unacknowledged messages you will run out of memory. An efficient way to limit unacknowledged messages is to limit how many messages your clients prefetch. To know which queue attached to which node in the cluster has the maximum number of unacknowledged messages, use the detailed diagnosis of this measure.
Current messages	Indicates the total number of messages on the cluster currently.	Number	This is the sum total of the value of the Ready messages and Unacknowledged messages. The value of this measure is a good

Measurement	Description	Measurement Unit	Interpretation
			indicator of the current message load on the cluster.
Published message rate	Indicates the rate at which publishers are publishing messages on the server.	Messages/Sec	
Publisher confirmation rate	Indicates the rate at which the cluster confirms the receipt of a message to a publisher.	Messages/Sec	A 'Publish Confirm' is nothing but a acknowledgement sent by the cluster to a publisher confirming the receipt of a message from that publisher. Publish Confirms have a performance impact. This means, the lower the value of this measure, the better. However, one should keep in mind that a Publish Confirm is required if the publisher needs at-least-once processing of messages.
Manually acknowledged message delivery rate	Indicates the rate at which messages are delivered to consumers that use manual acknowledgements.	Messages/Sec	Messages in transit might get lost in an event of a connection failure, and such a message might need to be retransmitted. Acknowledgements let the server and clients know when to retransmit messages. A manual acknowledgement is an 'explicit' acknowledgement that is received from the consumer. Manually sent acknowledgements can be positive or negative. Positive acknowledgements simply instruct RabbitMQ to record a message as delivered and can be discarded. Negative acknowledgements with *basic.reject* have the same effect. The difference is primarily in the semantics: positive acknowledgements assume a message was successfully processed while their negative counterpart suggests that a delivery

Measurement	Description	Measurement Unit	Interpretation
			wasn't processed but still should be deleted. Whether positive or negative, manual acknowledgements deliver low throughput and hence, should be avoided. A low value is therefore desired for this measure.
Auto-acknowledged message delivery rate	Indicates the rate at which messages are delivered to consumers that use automatic acknowledgements.	Messages/Sec	
Consumer acknowledgement rate	Indicates the rate at which messages are being acknowledged by consumers.	Messages/Sec	If the *Delivery rate of messages requiring acknowledgement* measure registers an abnormally low value, then, you may want to check the value of this measure at around the same time to determine whether the delay by consumers in acknowledging the messages was what caused the delivery delay.
Message redelivery rate	Indicates the rate at which messages with the 'redelivered' flag set are being delivered.	Messages/Sec	
Delivery rate of messages requiring acknowledgement	Indicates the rate at which messages requiring acknowledgement are being delivered in response to basic.get.	Messages/Sec	Compare the value of this measure with that of the *Delivery rate of messages not requiring acknowledgement* measure to figure out what type of messages are being delivered much slower than the rest.
Delivery rate of messages not requiring acknowledgement	Indicates the rate at which messages not requiring acknowledgement are being delivered in response to basic.get.	Messages/Sec	Compare the value of this measure with that of the *Delivery rate of messages requiring acknowledgement* measure to figure out what type of messages are

Measurement	Description	Measurement Unit	Interpretation
			being delivered much slower than the rest.
Message return rate	Indicates the rate at which unrouteable messages with 'mandatory' flag set to 'true', were sent to publishers.	Messages/Sec	An unroutable message is a message without a destination. For example, a message sent to an exchange without any bound queue. If the 'mandatory' flag is set to 'true', then the cluster return an unroutable message to the producer with a `basic.return` AMQP method.
Disk read rate	Indicates the rate at which queues read messages from disk.	Messages/Sec	A high value could indicate that messages are frequently read from the disk and not from the RAM. This could be owing to high memory pressure, which may have forced RabbitMQ to move messages from RAM to disk.
Disk write rate	Indicates the rate at which queues wrote messages to disk.	Messages/Sec	A high value for this measure could indicate any of the following: • Many messages have been published in such a way that they must be written to disk; • A very high memory pressure on RabbitMQ has caused the cluster to move majority of messages from RAM to disk;
Connections	Indicates the number of connections for all virtual hosts that the current user has access to.	Number	Each connection uses about 100 KB of RAM (and even more, if TLS is used). Which means, if the value of Connections measure is over 1000, it can be a heavy burden on a RabbitMQ server. In the worst case, the server can crash due to out-of-memory. The AMQP protocol has a

Measurement	Description	Measurement Unit	Interpretation
			mechanism called channels that "multiplexes" a single TCP connection. It's recommended that each process only creates one TCP connection, and uses multiple channels in that connection for different threads. Its also recommended that both connections and channels are kept at a minimum. This is because, an unusually high value for the *Connections* and the *Channels* measures can adversely impact the performance of the RabbitMQ management interface.
Channels	Indicates the total number of channels for all virtual hosts the current user has access to.	Number	
Exchanges	Indicates the total number of exchanges for all virtual hosts the current user has access to.	Number	An exchange is responsible for the routing of the messages to the different queues. An exchange accepts messages from the producer application and routes them to message queues with the help of bindings and routing keys.
Queues	Indicates the total number of queues for all virtual hosts the current user has access to.	Number	A queue is a buffer that stores messages. Messages are published to exchanges, which distribute them to queues using rules called bindings. Queues are single-threaded in RabbitMQ, and one queue can handle up to about 50k messages/s. You will achieve better throughput on a multi-core system if you have multiple queues and consumers. You will achieve optimal throughput if you have as many queues as cores on the underlying node(s). This means, ideally, the value of this measure should be equal to the number of cores on the monitored node.

Measurement	Description	Measurement Unit	Interpretation
			If the value of this measure is over 1000, it is a cause for concern. This is because, the RabbitMQ management interface will keep information about all queues and this might slow down the server. The CPU and RAM usage may also be affected in a negative way if you have too many queues (thousands of queues). The RabbitMQ management interface collects and calculates metrics for each and every queue which uses some resources and CPU and disk contention can occur if you have thousands up on thousands of active queues and consumers.
Consumers	Indicates the total number of consumers for all virtual hosts the current user has access to.	Number	A consumer is a user application that receives messages. You will achieve better throughput on a multi-core system if the value of this measure is more than one. However, if there are a large number of consumers, CPU and disk contention can occur on the RabbitMQ management interface.
Published messages	Indicates the total number of messages entering the server.	Number	
Publisher confirmed messages	Indicates the total number of messages that the server is confirming to publisher.	Number	A 'Publish Confirm' is nothing but a acknowledgement sent by the cluster to a publisher confirming the receipt of a message from that publisher. Publish Confirms have a performance impact. This means, the lower the value of this measure, the better. However, one should keep in mind that a Publish Confirm is required if the publisher needs at-least-once

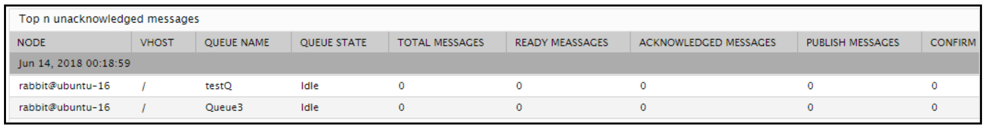
Measurement	Description	Measurement Unit	Interpretation
			processing of messages.
Messages delivered with manual acknowledgement	Indicates the total number of messages that this virtual host delivered to consumers that use manual acknowledgements.	Number	Messages in transit might get lost in an event of a connection failure, and such a message might need to be retransmitted. Acknowledgements let the server and clients know when to retransmit messages. A manual acknowledgement is an 'explicit' acknowledgement that is received from the consumer. Manually sent acknowledgements can be positive or negative. Positive acknowledgements simply instruct RabbitMQ to record a message as delivered and can be discarded. Negative acknowledgements with *basic.reject* have the same effect. The difference is primarily in the semantics: positive acknowledgements assume a message was successfully processed while their negative counterpart suggests that a delivery wasn't processed but still should be deleted. Whether positive or negative, manual acknowledgements deliver low throughput and hence, should be avoided. A low value is therefore desired for this measure.
Messages delivered with auto-acknowledgement	Indicates the total number of messages that this virtual host delivered to consumers that use automatic acknowledgements.	Number	In automatic acknowledgement mode, a message is considered to be successfully delivered immediately after it is sent. This mode trades off higher throughput (as long as the consumers can keep up) for reduced safety of delivery and consumer

Measurement	Description	Measurement Unit	Interpretation
			processing. This mode is often referred to as "fire-and-forget". Unlike with manual acknowledgement model, if consumers's TCP connection or channel is closed before successful delivery, the message sent by the server will be lost. Therefore, automatic message acknowledgement should be considered unsafe and not suitable for all workloads. The value 0 is hence ideal for this measure.
Messages delivered with consumer acknowledgement	Indicates the total number of messages that are being acknowledged by consumers of this virtual host.	Number	
Redelivered messages	Indicates the total number of messages that are being delivered by this virtual host, with the 'redelivered' flag set.	Number	If a message is delivered to a consumer and then requeued (because it was not acknowledged before the consumer connection dropped, for example) then RabbitMQ will set the 'redelivered' flag on it when it is delivered again (whether to the same consumer or a different one). This is a hint that a consumer may have seen this message before (although that's not guaranteed, the message may have made it out of the broker but not into a consumer before the connection dropped). Conversely if the redelivered flag is not set then it is guaranteed that the message has not been seen before.
Messages requiring acknowledgements	Indicates the total number of messages requiring acknowledgement that are	Number	For best performance and high throughput, the value of this measure should be low.

Measurement	Description	Measurement Unit	Interpretation
	being delivered in response to basic.get .		
Messages not requiring acknowledgements	Indicates the total number of messages not requiring acknowledgement that are being delivered in response to basic.get on this virtual host.	Number	
Messages returned	Indicates the rate at which this virtual host sent unrouteable messages with 'mandatory' flag set to 'true', to publishers .	Messages/Sec	An unroutable message is a message without a destination. For example, a message sent to an exchange without any bound queue. If the 'mandatory' flag is set to 'true', then an unroutable message is returned to the producer with a \`basic.return\` AMQP method. To know which nodes in the cluster returned the maximum number of messages to publishers, use the detailed diagnosis of this measure.
Disk reads	Indicates the total number of messages read from disk on this virtual host.	Number	A high value could indicate that many messages are read from the disk and not from the RAM. This could be owing to high memory pressure, which may have forced RabbitMQ to move messages from RAM to disk. If this measure reports an abnormally high value, then use the detailed diagnosis of this measure to know which nodes in the cluster are performing the maximum reads from the disk. Such nodes could be running out of memory.
Disk writes	Indicates the total number of messages written to disk on this virtual host.	Number	A high value for this measure could indicate any of the following: • Many messages have been

Measurement	Description	Measurement Unit	Interpretation
			published in such a way that they must be written to disk; • A very high memory pressure on RabbitMQ has caused the cluster to move majority of messages from RAM to disk; If this measure reports an abnormally high value, then use the detailed diagnosis of this measure to know which nodes in the cluster are performing the maximum reads from the disk. Such nodes could be running out of memory.

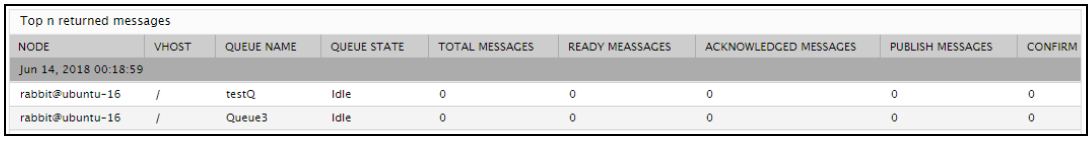
To know which queue attached to which node in the cluster has the maximum number of unacknowledged messages, use the detailed diagnosis of the *Unacknowledged messages* measure. The state of the queue is also revealed as part of detailed diagnostics. If your cluster is running out of memory, you can use the detailed statistics to identify which queue attached to which node is draining the memory by holding many unacknowledged messages.


Top n unacknowledged messages

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.4: The detailed diagnosis of the Unacknowledged messages measure

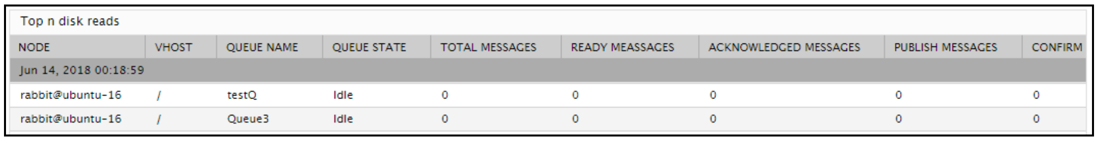
To know which nodes in the cluster returned the maximum number of messages to publishers, use the detailed diagnosis of the *Messages returned* measure.


Top n returned messages

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.5: The detailed diagnosis of the Messages returned measure

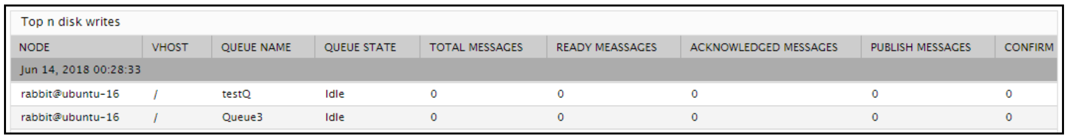
Using the detailed diagnosis of the *Disk reads* and *Disk writes* measures, you can easily identify the exact node in the cluster that is performing many reads from the disk and/or numerous writes to the disk. Excessive read-write activity on disk is a sign of a memory contention in the cluster. Under such circumstances, you can use the detailed diagnostics to identify the node and the queue that could be contributing to the contention.



Top n disk reads

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.6: The detailed diagnosis of the Disk reads measure



Top n disk writes

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:28:33								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.7: The detailed diagnosis of the Disk writes measure

4.3 The RabbitMQ Messages Layer

Use the tests mapped to this layer to know which exchanges and queues are operational in the virtual hosts the 'configured user' (i.e., the user you have configured for the eG tests) has access to. The tests will also help you instantly capture processing bottlenecks that any exchange may be experiencing and abnormal increase in the length of any queue.

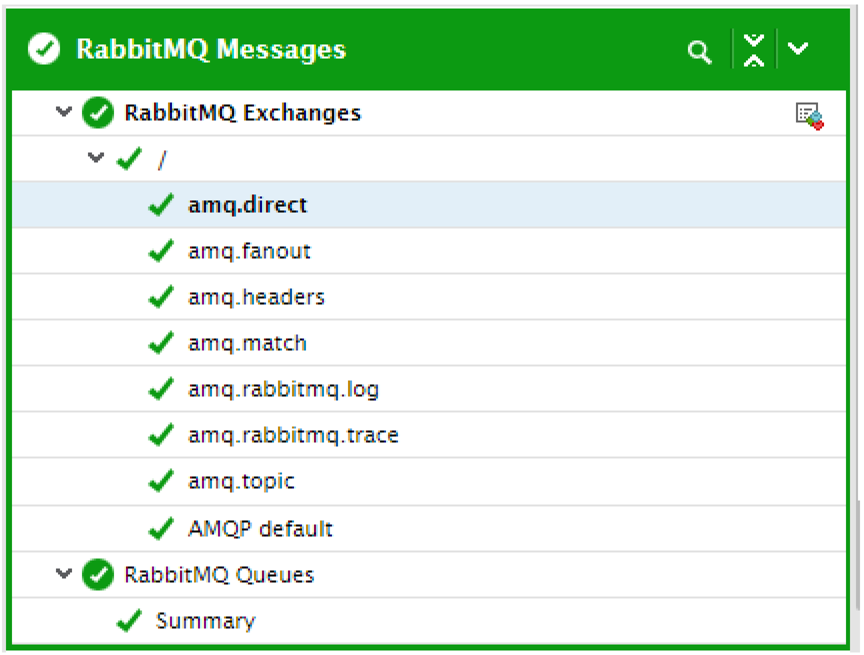


Figure 4.8: The tests mapped to the RabbitMQ Messages layer

4.3.1 RabbitMQ Exchanges Test

Messages are not published directly to a queue, instead, the producer sends messages to an exchange. An exchange is responsible for the routing of the messages to the different queues. An exchange accepts messages from the producer application and routes them to message queues with the help of bindings and routing keys.

To understand the load on each exchange and measure the ability of the exchange to process the load, use the RabbitMQ Exchanges test. This test auto-discovers the exchanges in a cluster and reports how quickly each exchange sends out the messages it receives from producers/publishers to queues. In the process, the test points to those exchanges that are unable to route messages to queues as fast as they receive them.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each exchange

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Type	Indicates the type of this exchange.		An exchange can be of one of the following types: • **Direct:** A direct exchange delivers messages to queues based on a message routing key. In a direct exchange, the message is routed to the queues whose binding key exactly matches the routing key of the message. If the queue is bound to the exchange with the binding key pdfprocess, a message published to the exchange with a routing key pdfprocess will be routed to that queue. • **Fanout:** A fanout exchange routes messages to all of the queues that are bound to it. • **Topic:** The topic exchange does a wildcard match between the routing key and the routing pattern specified in the binding. • **Headers:** Headers exchanges use the message header attributes for routing. The values that this measure can take and their corresponding numeric values are as follows:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td><table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Direct</td><td>1</td></tr><tr><td>Fanout</td><td>2</td></tr><tr><td>Headers</td><td>3</td></tr><tr><td>Topic</td><td>4</td></tr></table><b>Note:</b>
This test reports the <b>Measure Value</b>s listed in the table above to indicate the exchange type. In the graph of this measure however, the same will be represented using the numeric equivalents.</td></tr>
<tr><td>Features</td><td>Indicates the features of this exchange.</td><td></td><td>The value of this measure can be any of the following:<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Durable</td><td>0</td></tr><tr><td>Internal</td><td>1</td></tr><tr><td>Durable, Internal</td><td>10</td></tr></table>Durable exchanges will survive server restarts and will last until they are explicitly deleted. The setting called "Internal" is for exchanges that do not have an external client connecting to them.
<b>Note:</b>
This test reports the <b>Measure Value</b>s listed in the table above to indicate the exchange features. In the</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			graph of this measure however, the same will be represented using the numeric equivalents.
Published in	Indicates the rate at which messages are published into this exchange.	Messages/Sec	
Published out	Indicates the rate at which this exchange sends messages out to queues.	Messages/Sec	If the value of the *Published in* measure is abnormally higher than the value of the *Published out* measure, it implies that the corresponding exchange does not have the processing power to process all the messages that come into it..

4.3.2 RabbitMQ Queues Test

A queue is best defined as a buffer that stores messages. The basic architecture of a message queue is simple, there are client applications called producers that create messages and deliver them to the broker (the message queue). Other applications, called consumers, connects to the queue and subscribes to the messages to be processed. A software can be a producer, or consumer, or both a consumer and a producer of messages. Messages placed onto the queue are stored until the consumer retrieves them.

Since contents of a queue are typically located on a single node in the cluster, failure of that node can deny consumers access to that queue and its contents. Administrators should therefore continuously track the status of the queues, identify queues that are down, and get the down queues up and running before users complain. Also, too many messages in a queue can cause a contention for memory resources. This in turn can slow down message processing. To avoid this, administrators should track the length of each queue, promptly identify the queues that are consistently growing in length , and rapidly initiate measures to curb queue growth. The RabbitMQ Queues test helps administrators with all of the above!

This test auto-discovers the queues on a target node. For each queue, the test then reports the status of the queue, the total number of messages in the queue, and the type of messages - eg., unacknowledged, published, confirmed - in each queue, so administrators can precisely pinpoint queues that are growing in length at an alarming rate, know what type of messages are in such queue, and accordingly decide on how to control the growth in queue length. Additionally, the test reports on the memory usage of each queue, so you can easily assess the impact of queue length on memory. Moreover, the test also reports the rate at which the different types of messages in a queue are delivered to consumers, so that any bottleneck in delivery can be proactively detected and promptly fixed.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each queue in the target node

First-level descriptor: Node name

Second-level descriptor: Queue name

Configurable parameters for the test

Parameters	**Description**
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.
Num DD Messages	By default, this parameter is set to 10. This means that, by default, the detailed diagnosis of this test will report the details of the **top-10** queues in terms of the number of messages in each queue. To view the details of more queues as part of detailed metrics, you will have to increase the value of this parameter. Likewise, to view the details of less than 10 queues, reduce the value of this parameter.
Individual Queues	If you want the test to report metrics for every queue, then set this flag to **Yes**. In this case, each queue process will be a descriptor of this test. On the other hand, if you

Parameters	Description
	want to receive an overview of the queue load on the cluster, you can set this flag to **No**. In this case, the test will report metrics for a *Summary* descriptor alone.
Queue group	Provide a comma-separated list of queue groups you want monitored.
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 enable/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
State	Indicates the state of this queue		The values that this measure can take and their corresponding numeric values are as follows: **Measure Value** / **Numeric Value** Running / 0 Idle / 1 **Note:** This test reports the **Measure Value**s listed in the table above to indicate the queue status. In the graph of this measure however, the same will be represented using the numeric equivalents. **This measure is not available for**

Measurement	Description	Measurement Unit	Interpretation
			the Summary descriptor.
Running queues	Indicates the current count of running queues.	Number	**This measure is reported only for the Summary descriptor.** Use the detailed diagnosis of this measure to know which queues in the cluster are running.
Idle/down queues	Indicates the current count of idle/down queues.	Number	**This measure is reported only for the Summary descriptor.** Use the detailed diagnosis of this measure to know which queues in the cluster are not running. Ideally, the value of this measure should be 0. If one/more queues go down frequently, you may want to consider mirroring the queues across multiple nodes in a cluster to ensure high availability of the queues. Each mirrored queue consists of one master and one or more mirrors. The master is hosted on one node commonly referred as the master node. Each queue has its own master node. All operations for a given queue are first applied on the queue's master node and then propagated to mirrors. This involves enqueueing publishes, delivering messages to consumers, tracking acknowledgements from consumers and so on. Messages published to the queue are replicated to all mirrors. Consumers are connected to the master regardless of which node they

Measurement	Description	Measurement Unit	Interpretation
			connect to, with mirrors dropping messages that have been acknowledged at the master. Queue mirroring therefore enhances availability, but does not distribute load across nodes (all participating nodes each do all the work). If the node that hosts queue master fails, the oldest mirror will be promoted to the new master as long as it synchronised. Unsynchronised mirrors can be promoted, too, depending on queue mirroring parameters.
Unacknowledged messages	Indicates the number of messages in this queue that are waiting for acknowledgement. For the Summary descriptor, this measure will report the total number of messages across all queues that are waiting for acknowledgement.	Number	A low value is desired for this measure. This is because, all unacknowledged messages have to reside in RAM on the servers. If you have too many unacknowledged messages you will run out of memory. An efficient way to limit unacknowledged messages is to limit how many messages your clients prefetch. To know which queue attached to which node in the cluster has the maximum number of unacknowledged messages, use the detailed diagnosis of this measure.
Messages in queue	Indicates the total number of messages currently in queue. For the Summary descriptor, this measure indicates the total number of messages across all queues in the cluster.	Number	The value of this measure indicates the current queue length. Ideally, the value of this measure should be small for any queue. This is because, short queues are the fastest. When a queue is empty, and it has consumers ready to receive messages, then as soon as a

Measurement	Description	Measurement Unit	Interpretation
			message is received by the queue, it goes straight out to the consumer. Many messages in a queue can put a heavy load on RAM usage. When this happens, RabbitMQ will start flushing (page out) messages to disk in order to free up RAM, and when that happens queueing speeds will deteriorate. Some common problems with long queues are as follows: • Small messages embedded in queue index • Take a long time to sync between nodes • Time-consuming to start a server with many messages • RabbitMQ management interface collects and stores stats for all queues There are many ways by which you can limit queue size. For starters, you can limit the maximum length of a queue to a set number of messages, or a set number of bytes (the total of all message body lengths, ignoring message properties and any overheads), or both. By default, when a maximum queue length or size is set and the maximum is reached is to drop or dead-letter messages from the front of the queue (i.e. the oldest messages in the queue). Queue size can also be limited using

Measurement	Description	Measurement Unit	Interpretation
			a Time-To-Live (TTL) extension. RabbitMQ allows you to set TTL (time to live) for both messages and queues. When TTL is set for a queue, then any message that has been in the queue for longer than the configured TTL is said to be dead.The server guarantees that dead messages will not be delivered using basic.deliver (to a consumer) or included into a basic.get-ok response (for one-off fetch operations). Further, the server will try to remove messages at or shortly after their TTL-based expiry. A TTL can be specified on a per-message basis, by setting the expiration field in the basic AMQP class when sending a basic.publish. The value of the expiration field describes the TTL period in milliseconds. The same constraints as for x-message-ttl apply. Since the expiration field must be a string, the broker will (only) accept the string representation of the number.
Published message rate	Indicates the rate at which publishers are publishing messages to this queue. For the Summary descriptor, this measure indicates the average rate at which publishers are publishing messages across queues.	Messages/Sec	
Publisher confirmation rate	Indicates the rate at which the receipt of a message	Messages/Sec	A 'Publish Confirm' is nothing but a acknowledgement sent by the cluster

Measurement	Description	Measurement Unit	Interpretation
	into this queue is confirmed to a publisher. For the Summary descriptor, this measure indicates the average rate at which the receipt of messages across all queues is confirmed to a publisher.		to a publisher confirming the receipt of a message from that publisher. Publish Confirms have a performance impact. This means, the lower the value of this measure, the better. However, one should keep in mind that a Publish Confirm is required if the publisher needs at-least-once processing of messages.
Manually acknowledged message delivery rate	Indicates the rate at which this queue delivers messages to consumers that use manual acknowledgements. For the Summary descriptor, this measure indicates the average rate at which the queues in the cluster deliver messages to consumers that use manual acknowledgements.	Messages/Sec	Messages in transit might get lost in an event of a connection failure, and such a message might need to be retransmitted. Acknowledgements let the server and clients know when to retransmit messages. A manual acknowledgement is an 'explicit' acknowledgement that is received from the consumer. Manually sent acknowledgements can be positive or negative. Positive acknowledgements simply instruct RabbitMQ to record a message as delivered and can be discarded. Negative acknowledgements with *basic.reject* have the same effect. The difference is primarily in the semantics: positive acknowledgements assume a message was successfully processed while their negative counterpart suggests that a delivery wasn't processed but still should be deleted. Whether positive or negative, manual acknowledgements deliver low throughput and hence, should be avoided. A low value is therefore desired for this measure.

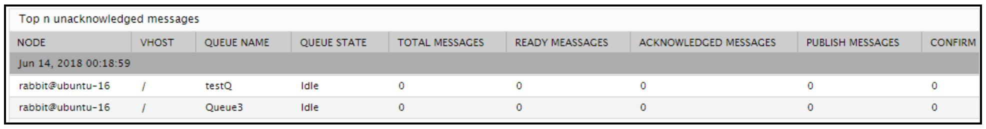
Measurement	Description	Measurement Unit	Interpretation
Auto-acknowledged message delivery rate	Indicates the rate at which this queue delivers messages to consumers that use automatic acknowledgements. For the Summary descriptor, this measure indicates the average rate at which the queues in the cluster deliver messages to consumers that use automatic acknowledgements.	Messages/Sec	
Consumer acknowledgement rate	Indicates the rate at which messages in this queue are being acknowledged by consumers. For the Summary descriptor, this measure indicates the average rate at which the messages across queues are being acknowledged by consumers.	Messages/Sec	If the *Delivery rate of messages requiring acknowledgement* measure registers an abnormally low value, then, you may want to check the value of this measure at around the same time to determine whether the a delay by consumers in acknowledging the messages was what caused the delivery delay.
Message redelivery rate	Indicates the rate at which this queue delivers messages with the 'redelivered' flag set. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster deliver messages with the 'redelivered' flag set.	Messages/Sec	
Delivery rate of messages requiring	Indicates the rate at which this queue delivers	Messages/Sec	Compare the value of this measure with that of the *Delivery rate of*

Measurement	Description	Measurement Unit	Interpretation
acknowledgement	messages requiring acknowledgement in response to basic.get. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster deliver messages requiring acknowledgement in response to basic.get.		*messages not requiring acknowledgement* measure to figure out what type of messages are being delivered much slower than the rest.
Delivery rate of messages not requiring acknowledgement	Indicates the rate at which messages not requiring acknowledgement are being delivered by this queue in response to basic.get. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster deliver messages not requiring acknowledgement in response to basic.get.	Messages/Sec	Compare the value of this measure with that of the *Delivery rate of messages requiring acknowledgement* measure to figure out what type of messages are being delivered much slower than the rest.
Message return rate	Indicates the rate at which this queue sent unrouteable messages with 'mandatory' flag set to 'true', to publishers. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster sent unrouteable messages with 'mandatory' flag set to 'true', to publishers.	Messages/Sec	An unroutable message is a message without a destination. For example, a message sent to an exchange without any bound queue. If the 'mandatory' flag is set to 'true', then the cluster return an unroutable message to the producer with a `basic.return` AMQP method.

Measurement	Description	Measurement Unit	Interpretation
Disk read rate	Indicates the rate at which this queue reads messages from disk. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster read messages from disk.	Messages/Sec	A high value could indicate that messages are frequently read from the disk and not from the RAM. This could be owing to high memory pressure, which may have forced RabbitMQ to move messages from RAM to disk.
Disk write rate	Indicates the rate at which this queue wrote messages to disk. For the Summary descriptor, this measure reports the average rate at which the queues in the cluster wrote messages to disk.	Messages/Sec	A high value for this measure could indicate any of the following: • Many messages have been published in such a way that they must be written to disk; • A very high memory pressure on RabbitMQ has caused the cluster to move majority of messages from RAM to disk;
Queues	Indicates the total number of queues currently in the cluster.	Number	**This measure is reported only for the Summary descriptor.** Use the detailed diagnosis of the measure to know which are the queues in the target cluster.
Used memory	Indicates the total memory used by this queue. For the Summary descriptor, this measure reports the total memory used up by all queues in cluster.	MB	Ideally, the value of this measure should be low. A consistent increase in this value could indicate excessive memory usage, and could hint at a potential memory contention. The key factor impacting memory usage is message queue length. Time-correlate the changes in the Messages in queue measure with that of the Used memory measure to

Measurement	Description	Measurement Unit	Interpretation
			figure out if the queue length is indeed increasing the memory pressure. In which case, you may want to initiate measures to limit the queue size.

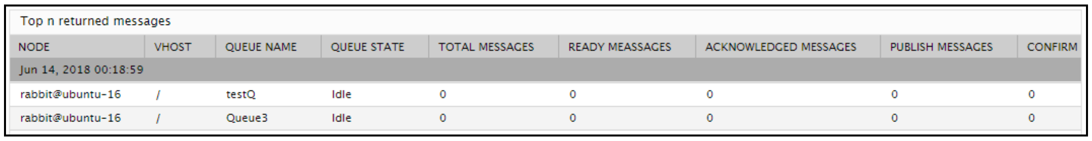
To know which queue attached to which node in the cluster has the maximum number of unacknowledged messages, use the detailed diagnosis of the *Unacknowledged messages* measure. The state of the queue is also revealed as part of detailed diagnostics. If your cluster is running out of memory, you can use the detailed statistics to identify which queue attached to which node is draining the memory by holding many unacknowledged messages.


Top n unacknowledged messages

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.9: The detailed diagnosis of the Unacknowledged messages measure

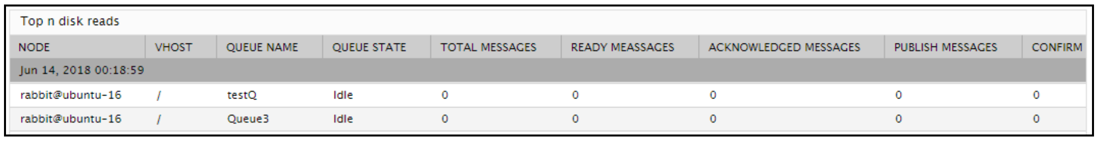
To know which queues returned the maximum number of messages to publishers, use the detailed diagnosis of the *Message return rate* measure.


Top n returned messages

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

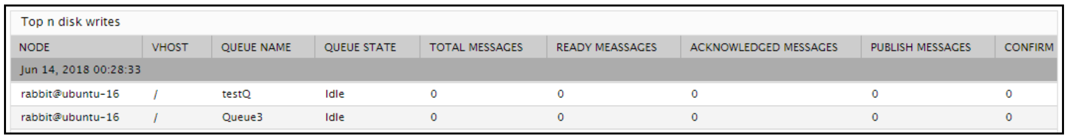
Figure 4.10: The detailed diagnosis of the Message return rate measure

Using the detailed diagnosis of the *Disk read rate* and *Disk write rate* measures, you can easily identify the exact queue that is performing many reads from the disk and/or numerous writes to the disk, and which node that queue is attached to. Excessive read-write activity on disk is a sign of a memory contention in the cluster. Under such circumstances, you can use the detailed diagnostics to identify the node and the queue that could be contributing to the contention.


Top n disk reads

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:18:59								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.11: The detailed diagnosis of the Disk read rate measure


Top n disk writes

NODE	VHOST	QUEUE NAME	QUEUE STATE	TOTAL MESSAGES	READY MEASSAGES	ACKNOWLEDGED MESSAGES	PUBLISH MESSAGES	CONFIRM
Jun 14, 2018 00:28:33								
rabbit@ubuntu-16	/	testQ	Idle	0	0	0	0	0
rabbit@ubuntu-16	/	Queue3	Idle	0	0	0	0	0

Figure 4.12: The detailed diagnosis of the Disk write rate measure

4.4 The RabbitMQ Service Layer

Using the tests mapped to this layer, you can measure the load on and processing ability of your TCP connections, application connections, and channels.

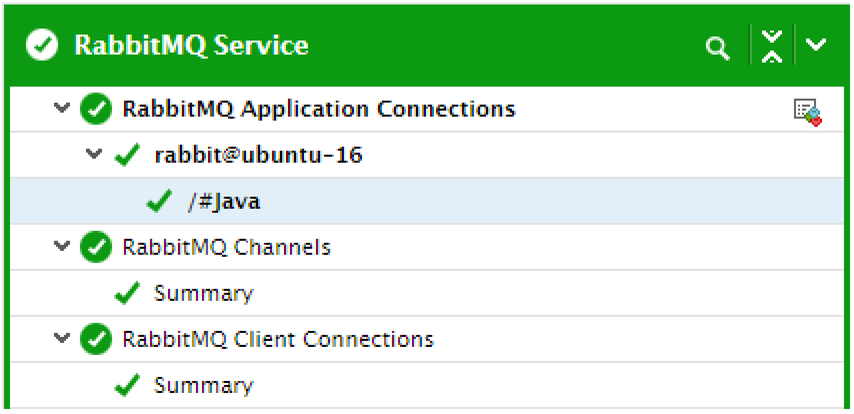


Figure 4.13: The tests mapped to the RabbitMQ Service layer

4.4.1 RabbitMQ Application Connections Test

Applications communicate with each other via a RabbitMQ server/cluster. By tracking connections from each application and measuring the data load that every application imposes on the cluster, you can easily identify which applications are overloading the cluster. This is what you can achieve with the RabbitMQ Application Connections test.

This test auto-discovers the applications that are currently connected to the cluster. For each auto-discovered application, the test then reports the count of connections established by that application on the cluster and the amount of data received and sent by the cluster. From this, we can quickly and accurately infer which application is overloading the cluster.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each application currently connected to every node in the target cluster

First-level descriptor: Node name

Second-level descriptor: Application process

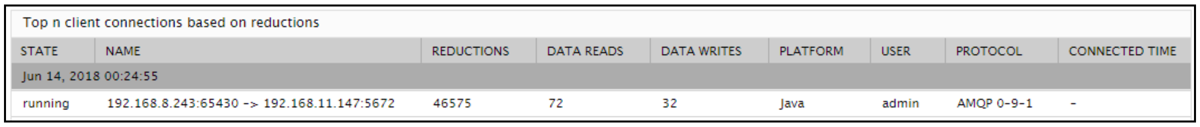
Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.
Num DD Messages	By default, this parameter is set to 10. This means that, by default, the detailed diagnosis of the *Number of application connections* measure will report the details of the **top-10** connections (of an application) in terms of their reduction count. To view the details of more connections as part of detailed metrics, you will have to increase the value of this parameter. Likewise, to view the details of less than 10 connections, reduce the value of this parameter.
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 enable/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
Number of application connections	Indicates the total number of instances of this application process currently running.	Number	This is a good indicator of the connection load imposed by an application on a node. Use the detailed diagnosis of this measure to view the top-10 (by default) connections made by this application, in terms of their reduction count.
Average data from client	Indicates the rate at which data was received from this application.	Bytes/Sec	
Average data to client	Indicates the rate at which data was sent to this application.	Bytes/Sec	
Average reductions	Indicates the rate at which reductions take place on this application.	Reductions/Sec	The reduction is a counter per process that is normally incremented by one for each function call. It is used for preempting processes and context switching them when the counter of a process reaches the maximum number of reductions. For example in Erlang/OTP R12B this maximum number was 2000 reductions. The value of this measure represents the rate at which an application makes function calls. This is the real indicator of the workload generated by an application on the cluster.

Use the detailed diagnosis of the *Number of application connections* measure to view the top-10 (by default) connections made by an application, in terms of their reduction count. The name of each connection, its current state, and the I/O overheads imposed by that connection are revealed. You can also know which user initiated that connection, when, and over what protocol.



Top n client connections based on reductions

STATE	NAME	REDUCTIONS	DATA READS	DATA WRITES	PLATFORM	USER	PROTOCOL	CONNECTED TIME
Jun 14, 2018 00:24:55								
running	192.168.8.243:65430 -> 192.168.11.147:5672	46575	72	32	Java	admin	AMQP 0-9-1	-

Figure 4.14: The detailed diagnosis of the Number of application connections measure

4.4.2 RabbitMQ Client Connections Test

Using this test, administrators can determine the different types of clients (Java clients, PHP clients, etc.) currently connected to the cluster and the number of current connections per type. This way, the test provides near-accurate indicators of the client connection load on the cluster. Also, by reporting the rate at which reductions and data are processed over the different types of client connections, the test reveals early to administrators if a specific type of clients are experiencing processing difficulties.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each type of client currently connected to the cluster

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME** and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.
Num DD Messages	By default, this parameter is set to 10. This means that, by default, the detailed

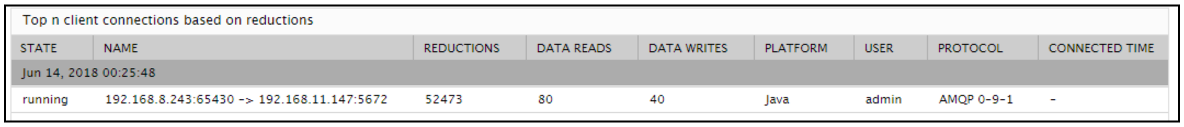
Parameters	Description
	diagnosis of the *Number of clients connected* measure will report the details of the **top-10** connections in terms of their reduction count. To view the details of more connections as part of detailed metrics, you will have to increase the value of this parameter. Likewise, to view the details of less than 10 connections, reduce the value of this parameter.
Individual Connections	If you want the test to report metrics for every client connection, then set this flag to **Yes**. In this case, each client connection process will be a descriptor of this test. On the other hand, if you want to receive an overview of the connection load on the cluster and measure the overall processing ability of the cluster, you can set this flag to **No**. In this case, the test will report metrics for a *Summary* descriptor alone.
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 enable/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
Number of clients connected	Indicates the number of clients of this type currently connected.	Number	Compare the value of this measure across client types to know which type of clients imposed the maximum connection load on the cluster. For the Summary descriptor, this measure will report the total number of connections across client types. Use the detailed diagnosis of this measure to view the top-10 (by default) connections, in terms of their reduction count.

Measurement	Description	Measurement Unit	Interpretation
Average data from client	Indicates the rate at which data was received from clients of this type.	Bytes/Sec	For the Summary descriptor, this measure will report the average number of bytes of data that was received and sent every second across all types of clients. Using the values reported by these measure for the Summary descriptor, you can evaluate the cluster throughput.
Average data to client	Indicates the rate at which data was sent by clients of this type.	Bytes/Sec	
Average reductions	Indicates the rate at which reductions take place on clients of this type.	Reductions/Sec	The reduction is a counter per process that is normally incremented by one for each function call. It is used for preempting processes and context switching them when the counter of a process reaches the maximum number of reductions. For example in Erlang/OTP R12B this maximum number was 2000 reductions. The value of this measure represents the rate at which clients of a type function calls. This is the real indicator of the workload generated by a client type on the cluster. To understand the workload of the cluster as a whole, use the value this measure reports for the Summary descriptor.

Use the detailed diagnosis of the *Number of clients connected* measure to view the top-10 (by default) connections, in terms of their reduction count. The name of each connection, its current state, and the I/O overheads imposed by that connection are revealed. You can also know which user initiated that connection, when, and over what protocol.



Top n client connections based on reductions

STATE	NAME	REDUCTIONS	DATA READS	DATA WRITES	PLATFORM	USER	PROTOCOL	CONNECTED TIME
Jun 14, 2018 00:25:48								
running	192.168.8.243:65430 -> 192.168.11.147:5672	52473	80	40	Java	admin	AMQP 0-9-1	-

Figure 4.15: The detailed diagnosis of the Number of clients connected measure

4.4.3 RabbitMQ Channels Test

A connection is a TCP connection between your application and the RabbitMQ broker. A channel is a virtual connection inside a connection. In other words, a channel multiplexes a TCP connection. Typically, each process only creates one TCP connection, and uses multiple channels in that connection for different threads. When you are publishing or consuming messages from a queue, it's all done over a channel.

Since unacknowledged messages are a resource- drain, RabbitMQ limits the number of unacknowledged messages a channel can hold at any point in time, using a Prefetch count configuration. Depending upon the unacknowledged message traffic on your channels and the count of consumers for those messages, you may want to fine-tune this Prefetch count time and again. By setting the *global* flag to *true*, you may even decide to configure a Consumer Prefetch count, which will be shared across all consumers on a channel.

The RabbitMQ channels test reports useful metrics on the message traffic and message prefetching on a channel. This way, the test provides administrators with effective pointers on how to tweak the Prefetch count setting and optimize RabbitMQ performance.

Target of the test : A RabbitMQ Cluster

Agent deploying the test : A remote agent

Outputs of the test : One set of results for each channel on every node in the target cluster

First-level descriptor: Node name

Second-level descriptor: Channel name

Configurable parameters for the test

Parameters	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the configured Host listens; by default, this is 15672
Username, Password, and Confirm Password	The eG agent connects to the Management Interface of the rabbitmq-management plugin of the target node, and runs HTTP-based API commands on the node using the plugin to pull metrics of interest. To connect to the plugin and run the API commands, the eG agent requires the privileges of a user on the cluster who has been assigned the 'monitoring' tag. If such a user pre-exists, then configure this test with the **USERNAME**

Parameters	Description
	and **PASSWORD** of that user. On the other hand, if no such user exists, then you will have to create a user for this purpose using the Management Interface. The steps for this have been detailed in Section **Chapter 2** In this case, make sure you configure this test with the **USERNAME** and **PASSWORD** of the new user. Finally, confirm the password by retyping it in the **CONFIRM PASSWORD** text box.
SSL	By default, this flag is set to **No**, as the target node is not SSL-enabled by default. If the node is SSL-enabled, then set this flag to **Yes**.
Num DD Messages	By default, this parameter is set to 10. This means that, by default, the detailed diagnosis of the *Number of channels connected* measure will report the details of the **top-10** channels in terms of their reduction count. To view the details of more channels as part of detailed metrics, you will have to increase the value of this parameter. Likewise, to view the details of less than 10 channels, reduce the value of this parameter.
Individual Channels	If you want the test to report metrics for every channel on a node, then set this flag to **Yes**. In this case, each channel will be a descriptor of this test. On the other hand, if you want to understand the overall message load across all channels and all nodes in a cluster, you can set this flag to **No**. In this case, the test will report metrics for a *Summary* descriptor alone.
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 enable/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
Number of channels connected	Indicates the number of channels connected.	Number	Use the detailed diagnosis of this measure to view the top-10 (by

Measurement	Description	Measurement Unit	Interpretation
	For the Summary descriptor, this measure will indicate the total number of channels connected.		default) channels, in terms of their reduction count.
Message prefetch rate for consumer	Indicates the rate at which each consumer on this channel prefetched messages. For the Summary descriptor, this measure will indicate the rate at which the consumers across all channels on all nodes prefetched messages.	Messages/Sec	If each consumer on a channel consumes messages at a steady rate at all times - i.e., if the value of this measure does not change much over time - it could be owing to a Prefetch setting per consumer. Prefetch allows you to limit the number of unacknowledged messages for a channel and/or a consumer. Once the number reaches the configured count, RabbitMQ will stop delivering more messages on the channel and/or to that consumer unless at least one of the outstanding messages is acknowledged. With the default Prefetch setting, which gives consumers an unlimited buffer, Rabbit will push all messages in a queue to a consumer as fast as the network and the consumer allow. The consumer will balloon in memory as they buffer all the messages in their own RAM. The queue may appear empty if you ask Rabbit, but there may be millions of messages unacknowledged as they sit in the consumers ready for processing by the client application. If you add a new consumer, there are no messages left in the queue to be sent to the new consumer. Messages are just being buffered in the existing

Measurement	Description	Measurement Unit	Interpretation
			consumer, and may be there for a long time, even if there are other consumers that become available to process such messages sooner. This means that with the default Prefetch setting, Rabbit performance will be poor. The goal is to keep the consumers saturated with work, but to minimise the client's buffer size so that more messages stay in Rabbit's queue and are thus available for new consumers or to just be sent out to consumers as they become free. Let's say it takes 50ms for Rabbit to take a message from a queue, put it on the network and for it to arrive at the consumer. It takes 4ms for the client to process the message. Once the consumer has processed the message, it sends an ack back to Rabbit, which takes a further 50ms to be sent to and processed by Rabbit. So we have a total round trip time of 104ms. If we have a prefetch setting of 1 message then Rabbit will not send out the next message until after this round trip completes. Thus the client will be busy for only 4ms of every 104ms, or 3.8% of the time. The goal is to keep the client busy 100% of the time. Here are some guidelines for setting the correct prefetch value: • If you have one single or few consumers processing messages quickly, we recommend

Measurement	Description	Measurement Unit	Interpretation
			prefetching many messages at once. Try to keep your client as busy as possible. If you have about the same processing time all the time and network behavior remains the same - you can simply take the total round trip time / processing time on the client for each message, to get an estimated prefetch value. • If you have many consumers, and a short processing time, we recommend a lower prefetch value than for one single or few consumers. A too low value will keep the consumers idling a lot since they need to wait for messages to arrive. A too high value may keep one consumer busy, while other consumers are being kept in an idling state. • If you have many consumers, and/or a long processing time, we recommend you to set prefetch count to 1 so that messages are evenly distributed among all your workers. Please note that if your client auto-ack messages, the prefetch value will have no effect.
Global message	Indicates the rate at which	Messages/Sec	If this measure reports a non-zero

Measurement	Description	Measurement Unit	Interpretation
prefetch rate	this channel prefetched messages across all its consumers. For the Summary descriptor, this measure will indicate the rate at which messages were prefetched across all channels and nodes.		value, it could indicate that the global Prefetch count configuration is active. By default, the Prefetch count configuration applies to each new consumer on a channel. If required, you can set the global flag in the basic.qos method to true, so that the Prefetch count configuration applies per channel (i.e., across all consumers of a channel). For instance, say that there are two consumers of a channel, with a global prefetch count of 15. In this case, both these consumers will only ever have 15 unacknowledged messages between them. Note that a per-channel Prefetch count and a per-consumer Prefetch count can even co-exist. For instance, say that there are two consumers of a channel. Also, assume that the per-channel Prefetch count is 15 and the per-consumer Prefetch count is 10. In this case, the two consumers will only ever have 15 unacknowledged messages between them, with a maximum of 10 messages for each consumer. This will be slower than the above example, due to the additional overhead of coordinating between the channel and the queues to enforce the global limit. A global Prefetch setting is ideal because a single channel may

Measurement	Description	Measurement Unit	Interpretation
			consume from multiple queues, thus requiring coordination between the channel and the queue(s) for every message sent to ensure they don't go over the limit. This coordination can be slow on a single machine, and very slow when consuming across a cluster. With the global setting, there is no need for this coordination.
Unacknowledged message rate	Indicates the rate of unacknowledged messages on this channel. For the Summary descriptor, this measure will indicate the rate of unacknowledged messages across all channels and nodes.	Messages/Sec	Ideally, the channel should hold very few unacknowledged messages, as such messages are resource-hungry. If the value of this measure keeps varying over time, it could imply that the default Prefetch count setting is at play. In this case, use the guidelines discussed in the **Interpretation** column of the *Message prefetch rate* and *Global message prefetch rate* measures to know how to fine-tune the Prefetch count configuration and limit the unacknowledged messages on a channel.
Unconfirmed message rate	Indicates the rate of unconfirmed messages on this channel. For the Summary descriptor, this measure will indicate the rate of unconfirmed messages across all channels and nodes.	Messages/Sec	Unconfirmed messages refer to those messages for which the broker is yet to send a receipt confirmation to the producer.
Uncommitted message rate	Indicates the rate of uncommitted messages on this channel.	Messages/Sec	Uncommitted messages are those that are received by the consumer for transactions that are not yet

Measurement	Description	Measurement Unit	Interpretation
	For the Summary descriptor, this measure will indicate the rate of uncommitted messages across all channels and nodes.		committed.
Uncommitted acknowledgement rate	Indicates the rate of uncommitted acknowledgements on this channel. For the Summary descriptor, this measure will indicate the rate of uncommitted acknowledgement across all channels and nodes.	Messages/Sec	Uncommitted acknowledgements are acknowledgements that are received by the node for transactions that are not yet committed.
Reduction rate	Indicates the rate at which reductions take place on this channel. For the Summary descriptor, this measure will indicate the rate of reductions across all channels and nodes.	Reductions/Sec	The reduction is a counter per process that is normally incremented by one for each function call. It is used for preempting processes and context switching them when the counter of a process reaches the maximum number of reductions. For example in Erlang/OTP R12B this maximum number was 2000 reductions. The value of this measure represents the rate at which a channel makes function calls. This is the real indicator of the workload generated by channel on a particular node. To understand the workload of the cluster as a whole, use the value this measure reports for the Summary descriptor.

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.