



Red Hat Mobile Application Platform 4.7

Administrator's Guide

For Red Hat Mobile Application Platform 4.7

Red Hat Mobile Application Platform 4.7 Administrator's Guide

For Red Hat Mobile Application Platform 4.7

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides information on administering RHMAP.

Table of Contents

CHAPTER 1. POST INSTALLATION PROXY SET-UP	5
1.1. CONFIGURING OPENSIFT	5
1.2. CONFIGURING RHMAP	5
1.2.1. Update BuildConfig Proxy Defaults	6
CHAPTER 2. CONFIGURING RHMAP CORE TO USE NON-WILDCARD SSL ROUTING	8
2.1. PREREQUISITES	8
2.2. PROVISIONING THE RHMAP PROXY	8
2.3. DISABLING THE RHMAP PROXY	9
CHAPTER 3. SCALING RHMAP CORE APPLICATION	10
3.1. SCALING PODS	10
3.2. EXAMPLE REPLICHA CONFIGURATION FOR CORE	10
3.3. LIMITATIONS	10
3.3.1. Database and cache pods	10
3.3.2. Storage and monitoring components	11
CHAPTER 4. MODIFYING SMTP SERVER SETUP IN THE CORE	12
4.1. SETTING SMTP ENVIRONMENT VARIABLES ON MILLICORE	12
4.2. VERIFYING SMTP SETTINGS FOR CORE	12
4.3. TROUBLESHOOTING	13
CHAPTER 5. BACKING UP A CORE	14
5.1. OVERVIEW	14
5.2. REQUIREMENTS	14
5.3. EXPECTED STORAGE REQUIREMENTS	14
5.4. WHAT DATA IS BACKED UP	14
5.5. BACKING UP THE MONGODB DATA	14
5.6. BACKING UP THE MYSQL DATA	15
5.7. BACKING UP FILESYSTEM DATA	15
5.8. BACKUP FREQUENCY	16
CHAPTER 6. RESTORING A CORE BACKUP	17
6.1. OVERVIEW	17
6.2. REQUIREMENTS	17
6.3. WHAT DATA IS RESTORED	17
6.4. PREPARING FOR RESTORATION	17
6.5. RESTORING MONGODB DATA	18
6.6. RESTORING MYSQL DATA	18
6.7. RESTORING COMPONENTS PERSISTENT DATA	19
6.8. CONFIRMING RESTORATION	20
CHAPTER 7. RHMAP 4.X MBAAS	21
7.1. OVERVIEW	21
7.2. ARCHITECTURE OF THE MBAAS	21
7.3. SECURITY CONSIDERATIONS	21
CHAPTER 8. PROVISIONING AN RHMAP 4.X MBAAS IN OPENSIFT 3	23
8.1. PREREQUISITES	23
8.2. INSTALLATION	23
8.2.1. Before The Installation	24
8.2.1.1. Network Configuration	24
8.2.1.1.1. Making Project Networks Global	24

8.2.1.2. Persistent Storage Setup	25
8.2.1.3. Apply Node Labels	25
8.2.1.3.1. Labelling for MBaaS components	25
8.2.1.3.2. Labelling for MongoDB replicas	26
8.2.1.3.2.1. Why are MongoDB replicas spread over multiple nodes?	27
8.2.2. Installing the MBaaS	27
8.2.3. Verifying The Installation	27
8.3. CREATING AN MBAAS TARGET	29
8.4. AFTER INSTALLATION	30
CHAPTER 9. ADJUSTING SYSTEM RESOURCE USAGE OF THE MBAAS AND CLOUD APPS	31
9.1. OVERVIEW	31
9.2. PREREQUISITES	31
9.3. ADJUSTING RESOURCE USAGE OF THE MBAAS	31
9.3.1. Calculating the Appropriate Resource Requests and Limits	31
9.3.1.1. Example	32
9.3.2. Overview of Resource Usage of MBaaS Components	32
9.4. ADJUSTING RESOURCE USAGE OF CLOUD APPS	32
9.4.1. Overview of Resource Usage of Cloud App Components	32
9.5. SETTING RESOURCE REQUESTS AND LIMITS	33
9.6. USING CLUSTER METRICS TO VISUALIZE RESOURCE CONSUMPTION	34
CHAPTER 10. SETTING UP SMTP FOR CLOUD APP ALERTS	35
10.1. OVERVIEW	35
10.2. PREREQUISITES	35
10.3. CONFIGURING SMTP SETTINGS IN FH-MBAAS	35
10.4. VERIFYING SMTP SETTINGS FOR MBAAS	35
10.5. TROUBLESHOOTING	35
CHAPTER 11. BACKING UP AN MBAAS	37
11.1. REQUIREMENTS	37
11.2. EXPECTED STORAGE REQUIREMENTS	37
11.3. WHAT DATA IS BACKED UP	37
11.4. BACKING UP THE MONGODB DATA	37
11.5. BACKING UP THE NAGIOS DATA	37
11.6. BACKUP FREQUENCY	38
11.7. EXAMPLE BACKUP SCRIPT	38
CHAPTER 12. RESTORING AN MBAAS BACKUP	39
12.1. OVERVIEW	39
12.2. REQUIREMENTS	39
12.3. WHAT DATA IS RESTORED	39
12.4. RESTORING NAGIOS DATA	39
12.5. RESTORING MONGODB DATA	40
12.6. CONFIRMING RESTORATION	41
CHAPTER 13. TROUBLESHOOTING THE RHMAP MBAAS	42
13.1. OVERVIEW	42
13.2. CHECK THE HEALTH ENDPOINT OF THE MBAAS	42
13.3. ANALYZE LOGS	43
13.4. COMMON PROBLEMS	43
13.4.1. A replica pod of mongodb-service is replaced with a new one	43
13.4.1.1. Summary	43
13.4.1.2. Fix	43

13.4.1.3. Result	45
13.4.2. MongoDB doesn't respond after repeated installation of the MBaaS	45
13.4.2.1. Summary	45
13.4.2.2. Fix	46
13.4.2.3. Result	50
13.4.3. MongoDB replica set stops replicating correctly	50
13.4.3.1. Summary	50
13.4.3.2. Fix	51
13.4.3.3. Result	51
13.4.4. An MBaaS component fails to start because no suitable nodes are found	51
13.4.4.1. Summary	51
13.4.4.2. Fix	52
13.4.4.3. Result	53
13.4.4.4. Result	53

CHAPTER 1. POST INSTALLATION PROXY SET-UP

After installing the Core, configure it to use a proxy.

1.1. CONFIGURING OPENSIFT

For steps to configure OpenShift see [OpenShift documentation](#) To determine the <docker-registry-ip> run the following commands:

```
oc project default
oc rsh <docker-registry-pod>
printenv DOCKER_REGISTRY_SERVICE_HOST
```

See the section "[Working with HTTP Proxies](#)" for information about configuring your OpenShift installation to use a proxy.

1.2. CONFIGURING RHMAP

1. Configure the node-proxy ConfigMap:

```
oc edit configmap node-proxy
```

This opens the node-proxy config map in your default editor.

Edit the following values in the relevant data keys, only including the username and password if the proxy requires authentication:

```
http-proxy: "http://<username>:<password>@<proxy-host>:<proxy-port>"
https-proxy: "http://<username>:<password>@<proxy-host>:<proxy-
port>"
```



NOTE

The https-proxy setting refers to internal traffic, use the http protocol, for example [http://user:pass@myproxy.com:80](#).

2. Configure the millicore-proxy ConfigMap:

```
oc edit configmap millicore-proxy
```

This opens the millicore-proxy config map in your default editor. Edit the following values in the relevant data keys, only including the millicore-proxy-user and millicore-proxy-pass if your proxy requires authentication:

```
millicore-proxy: <proxy-host>
millicore-proxy-password: <proxy-password>
millicore-proxy-port: '<proxy-port>'
millicore-proxy-user: <proxy-user>
```

3. Configure the ups-proxy ConfigMap:

```
oc edit configmap ups-proxy
```

■

This opens the ups-proxy config map in your default editor. Edit the following values in the relevant data keys, only including the http-proxy-user and http-proxy-pass if your proxy requires authentication:

```
http-proxy-host: <proxy-host>
http-proxy-pass: <proxy-password>
http-proxy-port: '<proxy-port>'
http-proxy-user: <proxy-user>
https-proxy-host: <proxy-host>
https-proxy-port: '<proxy-port>'
socks-proxy-host: '<socks-proxy-host>'
socks-proxy-port: '<socks-proxy-port>'
```

4. Redeploy all the relevant pods to pick up the new configuration. The services that require a pod redeploy are listed below. If you are running OpenShift 3.2 or 3.3, use the following command for each of the services:

```
for i in fh-aaa fh-metrics fh-messaging fh-ngui fh-scm fh-supercore
ups fh-appstore gitlab-shell millicore; do oc deploy ${i} --latest;
done
```

If you are running OpenShift 3.4, use the following command for each of the services:

```
for i in fh-aaa fh-metrics fh-messaging fh-ngui fh-scm fh-supercore
ups fh-appstore gitlab-shell millicore; do oc rollout latest ${i};
done
```

1.2.1. Update BuildConfig Proxy Defaults

In order for RHMAP to pull the github repo from an external URL without having to manually add the proxy details to the buildconfig, edit the `/etc/origin/master/master-config.yaml` file and configure the default proxy values.

At the top of this file is a section named **admissionConfig**. Note that this should be a root element, not a similarly named element under the `kubernetesMasterConfig` element. Add the following configuration:

Note: Exclude entries for `<username>` and `<password>` if proxy authentication is not required.

```
pluginConfig:
  BuildDefaults:
    configuration:
      apiVersion: v1
      kind: BuildDefaultsConfig
      gitHTTPProxy: http://<username>:<password>@<proxy-host>:<proxy-
port>
      gitHTTPSProxy: http://<username>:<password>@<proxy-host>:<proxy-
port>
      gitNoProxy: <nodes-ip-range>,<infra-ip-range>,<containers-ip-
range>,<docker-registry.default.svc.cluster.local
      env:
        - name: HTTP_PROXY
          value: http://<username>:<password>@<proxy-host>:<proxy-port>
        - name: HTTPS_PROXY
```

```
    value: http://<username>:<password>@<proxy-host>:<proxy-port>
  - name: NO_PROXY
    value: "<nodes-ip-range>,<infra-ip-range>,<containers-ip-
range>,docker-registry.default.svc.cluster.local"
```

Once this is complete, restart openshift master to make the changes take effect:

```
systemctl restart atomic-openshift-master.service
```

CHAPTER 2. CONFIGURING RHMAP CORE TO USE NON-WILDCARD SSL ROUTING

For a standard RHMAP installation, we recommend using a wild card SSL certificate installed on the OpenShift router. However, in situations where this is not desirable, RHMAP can be exposed using a single URL using the following procedure.

2.1. PREREQUISITES

- A running MBaaS, tested with a Cloud App
- A tested public IP address and domain with an attached certificate

2.2. PROVISIONING THE RHMAP PROXY

RHMAP Proxy is a separate component which routes requests from a single external host to Cloud Apps and RHMAP Core instances deployed on OpenShift.

For example, when RHMAP Proxy receives a request for https://rhmapproxy.internal.com/cloud_app_id/hello, that request is translated to http://cloud_app_id.internal.com/hello.

The RHMAP Proxy component is bundled within the RHMAP RPM in the form of an OpenShift template. Provisioning this template creates an OpenShift route and exposes this service.

Deploy an RHMAP Proxy for each MBaaS project:

1. Update the `/opt/rhmap/4.5/rhmap-installer/roles/non-wildcard-proxy/defaults/main.yml` file with the following information

- `project_name` - name for your new RHMAP Proxy
- `rhmap_core_project_name` - name of existing RHMAP Core project_name
- `base_host` - domain name of the OpenShift cluster
- `platform_url` - URL for RHMAP Studio
- `non_wildcard_external_host` - URL exposed to the Internet that retrieves the API URL when an application starts

For example:

```
project_name: "rhmap-non-wildcard-ssl-proxy"
rhmap_core_project_name: "rhmap-core"
base_host: "internal.domain.com"
platform_url: "https://rhmap.internal.domain.com"
non_wildcard_external_host: "proxy-route.internal.domain.com"
```

2. Run the playbook:

```
ansible-playbook -i <inventory-file> non-wild-card-proxy.yml
```

3. Consider the following if you deploy applications behind the RHMAP Proxy

If you serve static content from your application, it is important to consider how paths are written within the application's source code. It is recommended to use relative paths with dot notation.

For example where the URL displayed in a browser takes the format of <https://rhmapproxy.mydomain.com/myAppId/contacts>, in the source code of that application `Contact us` should be written as `Contact us` as appropriate.

Server side applications must have a trailing slash appended to the URL if one does not exist when viewed in a browser.

4. Configure existing proxies

If there are existing proxies in your infrastructure, you must configure these appropriately.

- **Reverse proxy**
A reverse proxy which is an entry point to your infrastructure and exposed to the Internet must be configured to point to the RHMAP Proxy OpenShift route.
- **HTTP proxy**
When using the RHMAP proxy in conjunction with a HTTP proxy, ensure that the wildcard DNS record that your MBaaS is using refers to a routable IP address that the RHMAP proxy can communicate with, that is the RHMAP proxy must be able to communicate with the IP address of the Cloud Apps deployed to your MBaaS.

5. Configure DNS

By default, RHMAP Proxy uses the default OpenShift DNS server to resolve internal domain names. If you use a custom DNS server within your network, run the following command to specify your DNS server IP address in the RHMAP Proxy deployment:

```
oc env dc nginx-proxy DNS_SERVER=<ip-address>
```

2.3. DISABLING THE RHMAP PROXY

If you have provisioned the RHMAP Proxy as described in [Provisioning the RHMAP Proxy](#) and you later decide that you do not require it, complete the following procedure:

1. Delete the non wildcard ssl proxy project:

```
oc delete project <rhmap-non-wildcard-ssl-proxy>
```

2. Unset the EXTERNAL_HOST environment variable:

```
oc env dc millicore -n rhmap-core EXTERNAL_HOST=''
```

3. Open Studio and navigate to the *Admin > MBaaS Targets* section.
4. For each MBaaS Target, remove the value for the **External MBaaS Host** field and save the changes.
5. Edit any static content so that all links resolve as expected.

CHAPTER 3. SCALING RHMAP CORE APPLICATION

This guide explains how to horizontally scale RHMAP Core components. OpenShift pod scaling allows to react to changes in traffic and allocate the necessary resources to handle current demand. Pods can be scaled manually or automatically, based on resource usage. For more information please refer to [OpenShift documentation](#).

3.1. SCALING PODS

RHMAP Core pods can be scaled using either oc command tool or web console.

To scale pods execute following commands:

1. Target oc tool to RHMAP OpenShift master

```
oc login http://yourcluster.url
```

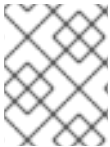
2. Fetch all RHMAP Core deployment configurations to get name of component you want to scale

```
oc get dc
```

3. To scale any of deployment configurations execute:

```
oc scale --replicas=2 dc name
```

Where **name** represents deployment configuration name returned by previous command.



NOTE

OpenShift3 also supports autoscaling that would allow components to scale based on resource usage. For more information please refer to [OpenShift documentation](#).

3.2. EXAMPLE REPLICA CONFIGURATION FOR CORE

Example commands that would scale most used components:

```
oc scale --replicas=2 dc fh-ngui
oc scale --replicas=2 dc millicore
oc scale --replicas=2 dc fh-supercore
oc scale --replicas=2 dc fh-aaa
oc scale --replicas=2 dc ups
```

3.3. LIMITATIONS

Not all RHMAP Core pods can be scaled using OpenShift api.

3.3.1. Database and cache pods

Increasing the replica count to more than one for any of these pods is not supported. Scaling following pods to more than one replica may cause loss of data and make platform unstable.

Component	Reason
mysql	Mysql is currently a standalone instance without support for master slave replication.
mongodb	Mongodb is a standalone instance that is not part of a replica set.
memcached	Not configured for replication. Core components are configured use single memcached instance.
redis	Not configured for replication. Core components are configured use single redis instance.

3.3.2. Storage and monitoring components

Component	Reason
gitlab-shell	Git repository data cannot be shared between nodes
fh-scm	Git repository data cannot be shared between nodes
nagios	Monitoring service - only one instance needed. Multiple instances of this pod may cause alert duplication.

CHAPTER 4. MODIFYING SMTP SERVER SETUP IN THE CORE

The platform sends emails for user account activation, password recovery, form submissions, and other events.

This procedure shows how to set up email support in the Core if it hasn't been set up during installation in the *Frontend setup* step in the [Installation Guide](#), or if you need to modify the SMTP server setup.

4.1. SETTING SMTP ENVIRONMENT VARIABLES ON MILLICORE

1. Edit the **millicore** deployment configuration and modify the environment variables with values for your SMTP server.

```
oc project <core-project-id>
oc edit dc/millicore

...
spec:
  ...
  template:
    ...
    spec:
      containers:
      - env:
        ...
        - name: SMTP_SERVER
          value: localhost
        - name: SMTP_PORT
          value: "25"
        - name: SMTP_AUTH
          value: "false"
        - name: SMTP_TLS
          value: "false"
        - name: SMTP_USERNAME
          value: ${SMTP_USERNAME}
        - name: SMTP_PASSWORD
          value: ${SMTP_PASSWORD}
        ...
```

2. Save your changes and close the editor.

After modifying the deployment configuration, a redeploy of the **millicore** pod is triggered automatically. Once the pod is running again, you can verify the changes.

4.2. VERIFYING SMTP SETTINGS FOR CORE

Verify the SMTP server setup by trying the forgotten password procedure:

1. Navigate to the URL of the Studio.
2. If you're logged in, log out.
3. Under the login form, click *Forgot your password?*

4. Fill in your email address.

5. Click *Reset my Password*.

You should receive an email with instructions for password recovery.

4.3. TROUBLESHOOTING

If the password recovery email doesn't arrive, verify the SMTP settings in the running **millicore** pod.

```
oc env pod -l name=millicore --list | grep SMTP
```

It may help to view the **millicore** logs while attempting the password recovery, looking for any errors related to SMTP or email.

```
oc logs -f millicore-<deploy-uuid>
```

If the test email seems to be sent, but fails to arrive, check it hasn't been placed in your spam or junk folder.

CHAPTER 5. BACKING UP A CORE

5.1. OVERVIEW

You can back up a Core by following this procedure. After completing the procedure and storing the backup data safely, you can restore a Core to the state at the time of backup using the [Restoring a Core Backup](#) procedure.

5.2. REQUIREMENTS

- A self-managed Core installation on an OpenShift platform
- A local installation of the **oc** binary
- The **oc** binary has a logged in user on the platform you wish to back up
- The **oc** binary has a logged in user with permission to run the **oc get pc** command

5.3. EXPECTED STORAGE REQUIREMENTS

Other factors that have an impact on how much storage is required for backups include:

- how often you backup
- what compression is used
- the length of time you store the backups

5.4. WHAT DATA IS BACKED UP

You must back up the following items to back up a Core:

- Mongodb Replica Set data
- MySQL data
- Nagios historical data
- Core metrics files
- Git files
- Core SCM files

5.5. BACKING UP THE MONGODB DATA

Back up the Mongodb data using the **mongodump** command in combination with the **oc exec** command:

```
oc exec `oc get po --selector='deploymentconfig=mongodb-1' --template="{{(index .items 0).metadata.name}}"` bash -- -c '/opt/rh/rh-mongodb32/root/usr/bin/mongodump -u admin -p ${MONGODB_ADMIN_PASSWORD} --gzip --archive' > ./core7_mongodb.gz
```

5.6. BACKING UP THE MYSQL DATA

Back up the MySQL data using the **mysqldump** command in combination with the **oc exec** command:

```
oc exec `oc get po --selector='deploymentconfig=mysql' --template="{{(index .items 0).metadata.name}}"` bash -- -c
'/opt/rh/mysql55/root/usr/bin/mysqldump -h mysql -u ${MYSQL_USER} -p${MYSQL_PASSWORD} --all-databases' > mysql-backup.sql
```

5.7. BACKING UP FILESYSTEM DATA

Back up the Nagios, metrics, git and scm files by copying the files on the associated persistent volumes:

1. Determine the Volume names that need to be backed up, by entering the following command:

```
oc get pvc
```

The output of this command displays the volume name, for example:

NAME	STATUS	VOLUME	CAPACITY
ACCESSMODES AGE			
git-data 20h	Bound	pv5g32	5Gi RWO
metrics-backup-log-data 20h	Bound	pv5g39	5Gi RWO
mongodb-claim-1 20h	Bound	pv25g95	25Gi RWO
mysql 20h	Bound	pv5g173	5Gi RWO
nagios-claim-1 2m	Bound	pv1g18	1Gi RWO
scm-data 20h	Bound	pv25g45	25Gi RWO

This shows that only the pv5g32, pv5g39, pv1g18 and pv25g45 persistent volumes need to be backed up, corresponding to the persistent volumes named git-data, metrics-backup-log-data, nagios-claim-1 and scm-data.

2. Determine the location of each of the filesystems that you want to back up using the **oc describe pv** command. For example to determine the location of the filesystems associated with the **pv5g32** persistent volume in step 1, enter the following:

```
oc describe pv pv5g32
```

The output of this command displays the filesystem location, for example:

```
Name: pv5g32
Labels: <none>
Status: Bound
Claim: core7/git-data
Reclaim Policy: Delete
Access Modes: RWO
Capacity: 5Gi
Message:
```

Source:

Type: HostPath (bare host directory volume)

Path: /home/vagrant/exp5g32

3. Create an archive of the files. For example, to archive the files for the git-data volume in the steps above, enter the following command:

```
cd /home/vagrant/exp5g32 && tar -zcf  
~/core_backup_2016_09_26_1300_001/core7_git-data.tar.gz .
```

5.8. BACKUP FREQUENCY

Red Hat recommends backing up at least once per day, but you might decide to back up critical data more frequently.

CHAPTER 6. RESTORING A CORE BACKUP

6.1. OVERVIEW

You can back up a Core by following the [Backing up a Core](#) procedure. Once you have created the backup, you can restore a Core to the state at the time of backup.

6.2. REQUIREMENTS

- A self-managed Core installation on an OpenShift platform
- The **oc** binary has a logged in user with permission to edit deployment configurations and view persistent volumes.
- A backup of the data for Mongodb, MySQL and all the components persistent volumes as described in [Backing up Filesystem Data](#).

6.3. WHAT DATA IS RESTORED

- Mongodb Replica Set data
- MySQL data
- Nagios historical data
- Core metrics files
- Git files
- Core SCM files

6.4. PREPARING FOR RESTORATION

Before restoring the data:

1. Scale the Mongodb and MySQL to run 1 replica.
2. Scale all the other pods to 0, waiting for the scaling to take effect.
You can scale the components down through the web interface or using the following commands:

```
oc scale --replicas=0 dc/fh-aaa
oc scale --replicas=0 dc/fh-appstore
oc scale --replicas=0 dc/fh-messaging
oc scale --replicas=0 dc/fh-metrics
oc scale --replicas=0 dc/fh-ngui
oc scale --replicas=0 dc/fh-scm
oc scale --replicas=0 dc/fh-supercore
oc scale --replicas=0 dc/gitlab-shell
oc scale --replicas=0 dc/memcached
oc scale --replicas=0 dc/millicore
oc scale --replicas=0 dc/nagios
oc scale --replicas=0 dc/redis
oc scale --replicas=0 dc/ups
```

-
- 3. Verify all pods have been scaled down:

```
oc get pods
```

Only **mysql** and **mongo** should be listed.

6.5. RESTORING MONGODB DATA



NOTE

If you did not create a gzipped archive dump as described in [Backing up Mongodb Data](#), you might need to change the mongorestore arguments used in this procedure.

1. Copy the backup file into the **tmp** directory of the mongodb pod using the **oc rsync** command:

```
oc rsync /path/to/backups/dir <mongo-pod-name>:/tmp
```

TIP

rsync copies everything in a directory. To save time put the Mongodb dump file into an empty directory before using rsync.

You can ignore errors similar to the following:

```
rsync: failed to set permissions on "/tmp/.": Operation not
permitted (1)
rsync error: some files/attrs were not transferred (see previous
errors)
(code 23) at main.c(1052) [sender=3.0.9] error: exit status 23
```

2. Connect to the remote shell of the Mongodb pod:

```
oc rsh <mongo-pod-name>
```

3. Run **mongorestore** on the backup data:

```
mongorestore -u admin -p ${MONGODB_ADMIN_PASSWORD} --gzip --
archive=mongodb-backup.gz
```

6.6. RESTORING MYSQL DATA

1. Copy the MySQL dump file into the **tmp** directory of the mysql pod using the **oc rsync** command:

```
oc rsync /path/to/backups/dir <mysql-pod-name>:/tmp
```

TIP

rsync copies everything in a directory. To save time put the MySQL dump file into an empty directory before using rsync.

You can ignore errors similar to the following:

```
rsync: failed to set permissions on "/tmp/.": Operation not
permitted (1)
rsync error: some files/attrs were not transferred (see previous
errors)
(code 23) at main.c(1052) [sender=3.0.9] error: exit status 23
```

2. Connect to the remote shell of the MySQL pod:

```
oc rsh <mysql-pod-name>
```

3. Restore the backed up data:

```
mysql -h mysql -u ${MYSQL_USER} -p${MYSQL_PASSWORD} < mysql-
backup.sql
```

6.7. RESTORING COMPONENTS PERSISTENT DATA

TIP

Every component other than MySQL and MongoDB is restored using this procedure and the components can be restored in any order

1. Take note of which persistent volume each component is using, by entering the following command:

```
oc get pvc
```

2. For each volume-name, other than MySQL and MongoDB:

```
oc describe pv <volume-name>
```

The output contains a section similar to the following, use this to determine the server and directory the data is stored in.:

```
Source:
  Type:          NFS (an NFS mount that lasts the lifetime of a pod)
  Server:        1.2.3.4
  Path:          /data/nfs/pv
  ReadOnly:      false
```

TIP

Take note of the permissions and ownership of any files in the data directory, and make sure that the restored files match these permissions and ownership

To restore the data:

1. Delete the contents of the pod's current persistent volume.
2. Extract the backed up data into that directory.

Once all the components data has been restored, you can scale the components up through the web interface or start the components using the following commands:

```
oc scale --replicas=1 dc/fh-aaa
oc scale --replicas=1 dc/fh-appstore
oc scale --replicas=1 dc/fh-messaging
oc scale --replicas=1 dc/fh-metrics
oc scale --replicas=1 dc/fh-ngui
oc scale --replicas=1 dc/fh-scm
oc scale --replicas=1 dc/fh-supercore
oc scale --replicas=1 dc/gitlab-shell
oc scale --replicas=1 dc/memcached
oc scale --replicas=1 dc/millicore
oc scale --replicas=1 dc/nagios
oc scale --replicas=1 dc/redis
oc scale --replicas=1 dc/ups
```

After running this command, wait until all pods have been scaled up. Verify they are running using the following command:

```
oc get pods
```

All the components should be listed, including **mysql** and **mongo** and the left and right values in the **READY** column should be equal.

6.8. CONFIRMING RESTORATION

Log in to the Studio and ensure that all the projects, environments and MBaaS targets are correctly restored.

CHAPTER 7. RHMAP 4.X MBAAS

7.1. OVERVIEW

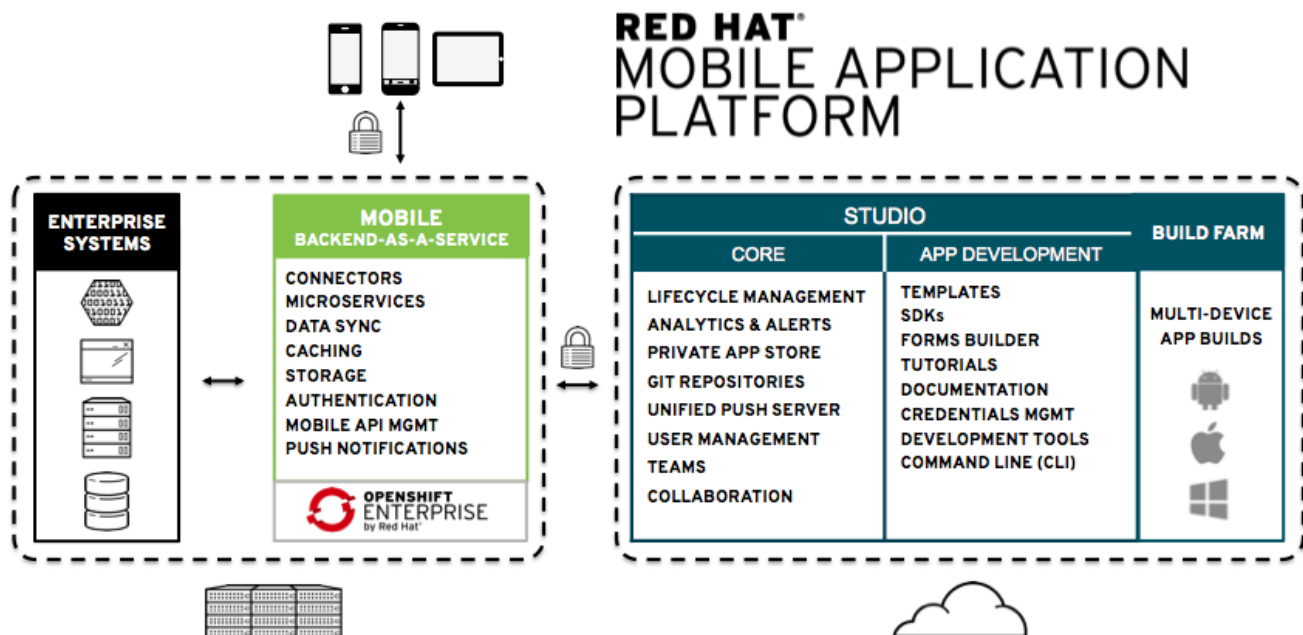
Red Hat Mobile Application (RHMAP) 4.x has a hybrid deployment model — the Core and the MBaaS are deployed in different locations.

- Development and management of apps occurs in the multi-tenant cloud instance of the RHMAP Core hosted by Red Hat.
- Application data, runtime, and integrations are deployed to the RHMAP MBaaS installed in a private or public instance of OpenShift Enterprise 3.

The *Mobile Backend-as-a-Service* (MBaaS) is a core component of RHMAP – the back-end platform hosting containerized Cloud App in conjunction with database storage (MongoDB). The Cloud Apps deployed in an MBaaS can make use of RHMAP APIs, such as data synchronization, caching, or push notifications, and integrate with enterprise systems or other Cloud Services.

7.2. ARCHITECTURE OF THE MBAAS

The RHMAP MBaaS 4.x is built on top of several technologies, including OpenShift v3, Kubernetes, Containers, and Red Hat Software Collections. The MBaaS consists of several components, each running in its own container. Similarly, every Cloud App deployed to the MBaaS runs in a container. Those containers are deployed and orchestrated by Kubernetes.



In the MBaaS, the users can configure multiple isolated runtime and storage environments to support software development life-cycle stages, such as development, testing, and production. Each environment can host multiple Cloud Apps.

7.3. SECURITY CONSIDERATIONS

Since the MBaaS is not hosted in Red Hat's public multi-tenant cloud, the data transmitted between the mobile device and the Cloud App does not pass through any servers operated by Red Hat or any other third party. Private data from backend systems is transmitted directly between mobile devices and the MBaaS.

The following data still resides in the RHMAP Core:

- User names and passwords of RHMAP accounts
- Master database of the Core, with entries for projects, apps, and their IDs
- Git repositories hosting the source code of Client and Cloud Apps
- App store containing the built binaries of Client Apps

CHAPTER 8. PROVISIONING AN RHMAP 4.X MBAAS IN OPENSIFT 3

An OpenShift 3 cluster can serve as an MBaaS target and host your Cloud Apps and Cloud Services. This guide provides detailed steps to deploy the RHMAP 4.x MBaaS on an OpenShift 3 cluster.

Follow the [Installation](#) procedure, which results in an MBaaS with the following characteristics:

- three replicas defined for each MBaaS component (with the exception of **fh-statsd**)
- three MongoDB replicas with a 50GB persistent storage requirement each
 - nodeSelectors of **mbaas_id=mbaas1**, **mbaas_id=mbaas2**, and **mbaas_id=mbaas3** for the MongoDB replicas

8.1. PREREQUISITES

This guide assumes several prerequisites are met before the installation:

- All nodes in the cluster must be registered with the Red Hat Subscription Manager and have RHMAP entitlement certificates downloaded. See [Preparing Infrastructure for Installation](#) for detailed steps.
- The MBaaS requires outbound internet access to perform npm installations, make sure that all relevant nodes have outbound internet access before installation.
- An existing OpenShift installation, version 3.7, 3.9 or 3.10.
- The OpenShift master and router must be accessible from the RHMAP Core.
- A wildcard DNS entry must be configured for the OpenShift router IP address.
- A trusted wildcard certificate must be configured for the OpenShift router. See [Using Wildcard Certificates](#) in OpenShift documentation.
- Image streams and images in the **openshift** namespace must be updated to the latest version. Refer to sections [3.3.8. Updating the Default Image Streams and Templates](#) and [3.3.9. Importing the Latest Images](#) in the OpenShift 3.2 Installation and Configuration guide.
- You must have administrative access to the OpenShift cluster using the **oc** CLI tool, enabling you to:
 - Create a *project*, and any resource typically found in a project (for example, *deployment configuration*, *service*, *route*).
 - Edit a *namespace* definition.
 - Create a *security context constraint*.
 - Manage nodes, specifically *labels*.

For information on installation and management of an OpenShift cluster and its users, see the [official OpenShift documentation](#).

8.2. INSTALLATION

The installation of an MBaaS in OpenShift 3 results in a resilient three-node cluster:

- MBaaS components are spread across all three nodes.
- MongoDB replica set is spread over three nodes.
- MongoDB data is backed by persistent volumes.
- A Nagios service with health checks and alerts is set up for all MBaaS components.

The installation consists of several phases. Before the installation, you must prepare your OpenShift cluster:

- [Set up persistent storage](#) - you need to create Persistent Volumes with specific parameters in OpenShift.
- [Label the nodes](#) - nodes need to be labeled in a specific way, to match the node selectors expected by the OpenShift template of the MBaaS.
- [Network Configuration](#) - configuring the SDN network plugin used in OpenShift so that Cloud Apps can communicate with MongoDB in the MBaaS.

Follow these steps after configuring the OpenShift cluster:

- [Install the MBaaS from a template](#)
- [Verify the installation](#)

8.2.1. Before The Installation

The installation procedure poses certain requirements on your OpenShift cluster in order to guarantee fault tolerance and stability.

8.2.1.1. Network Configuration

Cloud Apps in an MBaaS communicate directly with a MongoDB replica set. In order for this to work, the OpenShift SDN must be configured to use the **ovs-subnet** SDN plugin. For more detailed information on configuring this, see [Migrating Between SDN Plug-ins](#) in the OpenShift Enterprise documentation.

8.2.1.1.1. Making Project Networks Global

If you cannot use the **ovs-subnet** SDN plugin, you must make the network of the MBaaS project global after installation. For example, if you use the **ovs-multitenant** SDN plugin, projects must be configured as global. The following command is an example of how to make a project global:

```
oadm pod-network make-projects-global live-mbaas
```

To determine if projects are global, use the following command:

```
oc get netnamespaces
```

In the output, any projects that are configured global have namespaces with a value of "0"

**NOTE**

If a project network is configured as global, you cannot reconfigure it to reduce network accessibility.

For further information on how to make projects global, see [Making Project Networks Global](#) in the OpenShift Enterprise documentation.

8.2.1.2. Persistent Storage Setup

**NOTE**

Ensure that any NFS server and shares that you use are configured according to the [OpenShift documentation for configuring NFS PersistentVolumes](#). See the [Troubleshooting NFS Issues](#) section for more information on configuring NFS.

Some components of the MBaaS require persistent storage. For example, MongoDB for storing databases, and Nagios for storing historical monitoring data.

As a minimum, make sure your OpenShift cluster has the following persistent volumes in an **Available** state, with at least the amount of free space listed below:

- Three **50 GB** persistent volumes, one for each MongoDB replica
- One **1 GB** persistent volume for Nagios

For detailed information on persistent volumes and how to create them, see [Persistent Storage](#) in the OpenShift Enterprise documentation.

8.2.1.3. Apply Node Labels

By applying labels to OpenShift nodes, you can control which nodes the MBaaS components, MongoDB replicas and Cloud Apps will be deployed to.

This section describes the considerations for:

- [Section 8.2.1.3.1, “Labelling for MBaaS components”](#)
- [Section 8.2.1.3.2, “Labelling for MongoDB replicas”](#)

Cloud apps get deployed to nodes labeled with the default **nodeSelector**, which is usually set to **type=compute** (defined in the OpenShift master configuration).

8.2.1.3.1. Labelling for MBaaS components

It is recommended, but not required, to deploy the MBaaS components to dedicated nodes, separate from other applications (such as RHMAP Cloud Apps).

Refer to [Infrastructure Sizing Considerations for Installation of RHMAP MBaaS](#) for the recommended number of MBaaS nodes and Cloud App nodes for your configuration.

For example, if you have 12 nodes, the recommendation is:

- Dedicate three nodes to MBaaS and MongoDB.

- Dedicate three nodes to Cloud Apps.

To achieve this, apply a label, such as **type=mbaas** to the three dedicated MBaaS nodes.

```
oc label node mbaas-1 type=mbaas
oc label node mbaas-2 type=mbaas
oc label node mbaas-3 type=mbaas
```

Then, when creating the MBaaS project, as described later in [Section 8.2.2, “Installing the MBaaS”](#), set this label as the **nodeSelector**.

You can check what **type** labels are applied to all nodes with the following command:

```
oc get nodes -L type
```

NAME	STATUS	AGE	TYPE
ose-master	Ready,SchedulingDisabled	27d	master
infra-1	Ready	27d	infra
infra-2	Ready	27d	infra
app-1	Ready	27d	compute
app-2	Ready	27d	compute
app-3	Ready	27d	compute
mbaas-1	Ready	27d	mbaas
mbaas-2	Ready	27d	mbaas
mbaas-3	Ready	27d	mbaas

In this example, the deployment would be as follows:

- Cloud apps get deployed to the three dedicated Cloud App nodes **app-1**, **app-2**, and **app-3**.
- The MBaaS components get deployed to the three dedicated MBaaS nodes **mbaas-1**, **mbaas-2**, and **mbaas-3** (if the **nodeSelector** is also set on the MBaaS Project).

8.2.1.3.2. Labelling for MongoDB replicas

In the production MBaaS template, the MongoDB replicas are spread over three MBaaS nodes. If you have more than three MBaaS nodes, any three of them can host the MongoDB replicas.

To apply the required labels (assuming the three nodes are named **mbaas-1**, **mbaas-2**, and **mbaas-3**):

```
oc label node mbaas-1 mbaas_id=mbaas1
oc label node mbaas-2 mbaas_id=mbaas2
oc label node mbaas-3 mbaas_id=mbaas3
```

You can verify the labels were applied correctly by running this command:

```
oc get nodes -L mbaas_id
```

NAME	STATUS	AGE	MBAAS_ID
10.10.0.102	Ready	27d	<none>
10.10.0.117	Ready	27d	<none>
10.10.0.141	Ready	27d	<none>
10.10.0.157	Ready	27d	mbaas3

10.10.0.19	Ready,SchedulingDisabled	27d	<none>
10.10.0.28	Ready	27d	mbaas1
10.10.0.33	Ready	27d	<none>
10.10.0.4	Ready	27d	<none>
10.10.0.99	Ready	27d	mbaas2

See [Updating Labels on Nodes](#) in the OpenShift documentation for more information on how to apply labels to nodes.

8.2.1.3.2.1. Why are MongoDB replicas spread over multiple nodes?

Each MongoDB replica is scheduled to a different node to support failover.

For example, if an OpenShift node failed, data would be completely inaccessible if all three MongoDB replicas were scheduled on this failing node. Setting a different **nodeSelector** for each MongoDB **DeploymentConfig**, and having a corresponding OpenShift node in the cluster matching this label will ensure the MongoDB pods get scheduled to different nodes.

In the production MBaaS template, there is a different **nodeSelector** for each MongoDB **DeploymentConfig**:

- **mbaas_id=mbaas1** for **mongodb-1**
- **mbaas_id=mbaas2** for **mongodb-2**
- **mbaas_id=mbaas3** for **mongodb-3**

Excerpt of DeploymentConfig of mongodb-1

```
{
  "kind": "DeploymentConfig",
  "apiVersion": "v1",
  "metadata": {
    "name": "mongodb-1",
    "labels": {
      "name": "mongodb"
    }
  },
  "spec": {
    ...
    "template": {
      ...
      "spec": {
        "nodeSelector": {
          "mbaas_id": "mbaas1"
        }
      }
    }
  }
}
```

8.2.2. Installing the MBaaS

For information about installing the MBaaS, see the [Installing RHMAP](#) Guide.

8.2.3. Verifying The Installation

1. Ping the health endpoint.

If all services are created, all pods are running, and the route is exposed, the MBaaS health endpoint can be queried as follows:

```
curl `oc get route mbaas --template "
{{.spec.host}}"`/sys/info/health
```

The endpoint responds with health information about the various MBaaS components and their dependencies. If there are no errors reported, the MBaaS is ready to be configured for use in the Studio. Successful output will resemble the following:

```
{
  "status": "ok",
  "summary": "No issues to report. All tests passed without error",
  "details": [
    {
      "description": "Check Mongoddb connection",
      "test_status": "ok",
      "result": {
        "id": "mongoddb",
        "status": "OK",
        "error": null
      },
      "runtime": 33
    },
    {
      "description": "Check fh-messaging running",
      "test_status": "ok",
      "result": {
        "id": "fh-messaging",
        "status": "OK",
        "error": null
      },
      "runtime": 64
    },
    {
      "description": "Check fh-metrics running",
      "test_status": "ok",
      "result": {
        "id": "fh-metrics",
        "status": "OK",
        "error": null
      },
      "runtime": 201
    },
    {
      "description": "Check fh-statsd running",
      "test_status": "ok",
      "result": {
        "id": "fh-statsd",
        "status": "OK",
        "error": null
      },
      "runtime": 7020
    }
  ]
}
```


2. Verify that all Nagios checks are passing.

Log in to the Nagios dashboard of the MBaaS by following the steps in the [Accessing the Nagios Dashboard](#) section in the Operations Guide.

After logging in to the Nagios Dashboard, all checks under the left-hand-side **Services** menu should be indicated as **OK**. If any of the checks are not in an **OK** state, consult the Troubleshooting and Debugging guide, which explains the likely causes and fixes for common problems.

After verifying that the MBaaS is installed correctly, you must create an MBaaS target for the new MBaaS in the Studio.

8.3. CREATING AN MBAAS TARGET

1. In the Studio, navigate to the *Admin > MBaaS Targets* section. Click *New MBaaS Target*.
2. Enter the following information

- **MBaaS Id** - a unique ID for the MBaaS, for example, **live**. The ID must be equal to the OpenShift project name chosen in the *Installing the MBaaS* section, without the **-mbaas** suffix.
- **OpenShift Master URL** - the URL of the OpenShift master, for example, <https://master.openshift.example.com:8443>.
- **OpenShift Router DNS** - a wildcard DNS entry of the OpenShift router, for example, ***.cloudapps.example.com**.
- **MBaaS Service Key**
Equivalent to the value of the **FHMBaaS_KEY** environment variable, which is automatically generated during installation. To find out this value, run the following command:

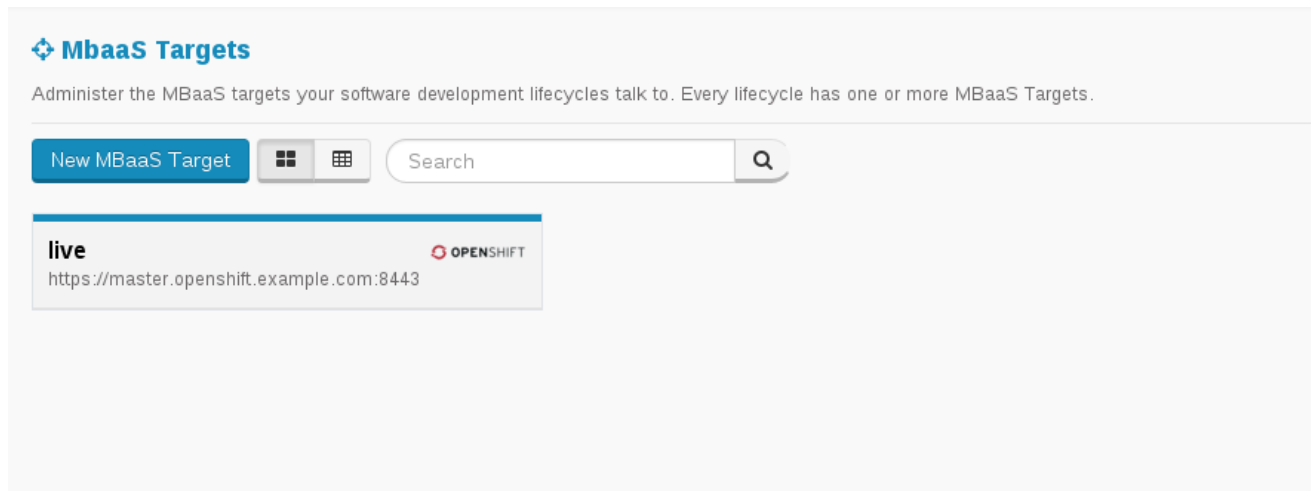
```
oc env dc/fh-mbaas --list | grep FHMBaaS_KEY
```

Alternatively, you can find the value in the OpenShift Console, in the *Details* tab of the **fh-mbaas** deployment, in the *Env Vars* section.

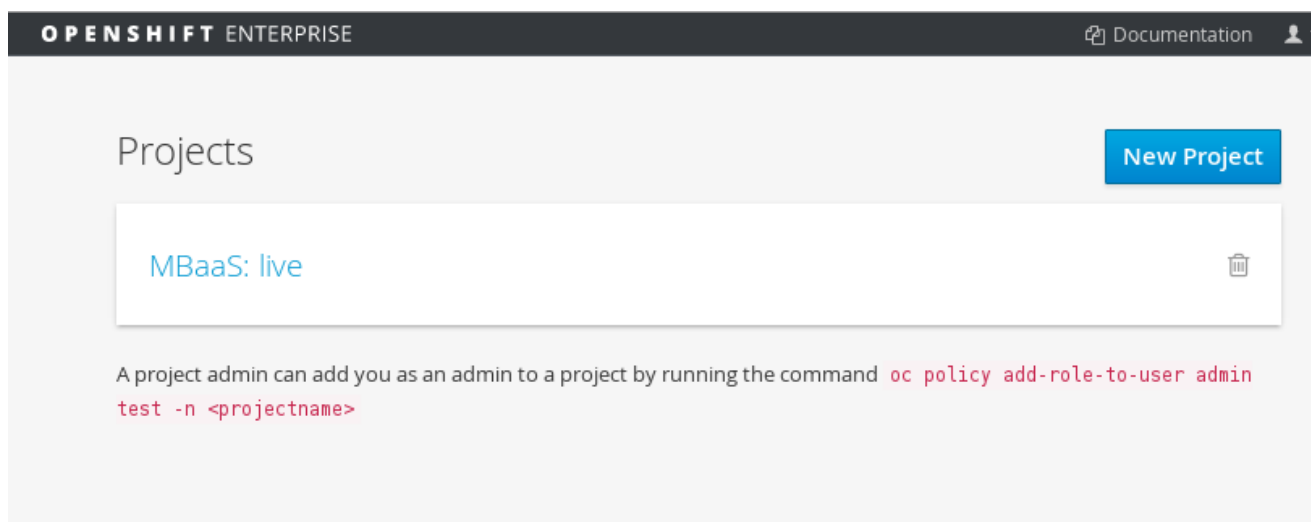
- **MBaaS URL**
A URL of the route exposed for the **fh-mbaas-service**, including the *https* protocol prefix. This can be retrieved from the OpenShift web console, or by running the following command:
- ```
echo "https://"$(oc get route/mbaas -o template --template {{.spec.host}})
```
- **MBaaS Project URL** - (Optional) URL where the OpenShift MBaaS project is available e.g. <https://mbaas-mymbaas.openshift.example.com:8443/console/project/my-mbaas/overview>.
  - **Nagios URL** - (Optional) Exposed route where Nagios is running in OpenShift 3 e.g. <https://nagios-my-mbaas.openshift.example.com>.

3. Click *Save MBaaS* and you will be directed to the MBaaS Status screen. For a manual installation, the status should be reported back in less than a minute.

Once the process of creating the MBaaS has successfully completed, you can see the new MBaaS in the list of MBaaS targets.



In your OpenShift account, you can see the MBaaS represented by a project.



## 8.4. AFTER INSTALLATION

- [Create an Environment](#) - you must create at least one environment for the MBaaS to be usable by Cloud Apps and Cloud Services
- [Adjusting System Resource Usage of the MBaaS and Cloud Apps](#) - we **strongly recommend** that you adjust the system resource usage of MBaaS components as appropriate for your production environment
- Optional: [Enabling Centralized Logging](#) - deploy a centralized logging solution based on ElasticSearch, Fluentd, and Kibana to debug issues with the MBaaS

## CHAPTER 9. ADJUSTING SYSTEM RESOURCE USAGE OF THE MBAAS AND CLOUD APPS

### 9.1. OVERVIEW

In the RHMAP 4.x MBaaS based on OpenShift 3, each Cloud App and each MBaaS component runs in its own container. This architecture allows for granular control of CPU and memory consumption. A fine level of control of system resources helps to ensure efficient use of nodes, and to guarantee uninterrupted operation of critical services.

An application can be prepared for various situations, such as high peak load or sustained load, by making decisions about the resource limits of individual components. For example, you could decide that MongoDB must keep working at all times, and assign it high, guaranteed amount of resources. At the same time, if the availability of a front-end Node.js server is less critical, the server can be assigned less initial resources, with the possibility to use more resources when available.

### 9.2. PREREQUISITES

The system resources of MBaaS components and Cloud Apps in the MBaaS can be regulated using the mechanisms available in OpenShift – resource *requests*, *limits*, and *quota*. Before proceeding with the instructions in this guide, we advise you to read the [Quotas and Limit Ranges](#) section in the OpenShift documentation.

### 9.3. ADJUSTING RESOURCE USAGE OF THE MBAAS

The RHMAP MBaaS is composed of several components, each represented by a single container running in its own pod. Each container has default resource requests and limits assigned in the MBaaS OpenShift template. See the section [Overview of Resource Usage of MBaaS Components](#) for a complete reference of the default values.

Depending on the deployment model of the MBaaS, you may have to adjust the resource limits and requests to fit your environment. If the MBaaS components are deployed on the same nodes as the Cloud Apps, there is no adjustment required.

However, when the MBaaS components are deployed on nodes dedicated to running the MBaaS only, it is strongly recommended to adjust the resource limits to take full advantage of the available resources on the dedicated nodes.

#### 9.3.1. Calculating the Appropriate Resource Requests and Limits



#### NOTE

This section refers to CPU resources in two different terms – the commonly used term *vCPU* (virtual CPU), and the term *millicores* used in OpenShift documentation. The unit of *1 vCPU* is equal to *1000 m (millicores)*, which is equivalent to 100% of the time of one CPU core.

The resource limits must be set accordingly for your environment and depend on the characteristics of load on your Cloud Apps. However, the following rules can be used as a starting point for adjustments of resource limits:

- Allow 2 GiB of RAM and 1 vCPU for the underlying operating system.

- Split the remainder of resources in equal parts amongst the MBaaS Components.

### 9.3.1.1. Example

Given a virtual machine with 16 GiB of RAM and 4 vCPUs, we allow 2 GiB of RAM and 1 vCPU for the operating system. This leaves 14GB RAM and 3 vCPUs (equal to 3000 m) to distribute amongst the 5 MBaaS components.

$$14 \text{ GiB} / 5 = 2.8 \text{ GiB of RAM per component}$$

$$3000 \text{ m} / 5 = 600 \text{ m per component}$$

In this example, the resource **limit** for each MBaaS component would be 2.8 GiB of RAM and 600 millicores of CPU. Depending on the desired level of quality of service of each component, set the resource **request** values as described in the section [Quality of service tiers](#) in the OpenShift documentation.

### 9.3.2. Overview of Resource Usage of MBaaS Components

The following table lists the components of the MBaaS, their idle resource usage, default resource request, and default resource limit.

| MBaaS component | Idle RAM usage | RAM request | RAM limit | Idle CPU usage | CPU request | CPU limit |
|-----------------|----------------|-------------|-----------|----------------|-------------|-----------|
| fh-mbaas        | 160 MiB        | 200 MiB     | 800 MiB   | <1%            | 200 m       | 800 m     |
| fh-messaging    | 160 MiB        | 200 MiB     | 400 MiB   | <1%            | 200 m       | 400 m     |
| fh-metrics      | 120 MiB        | 200 MiB     | 400 MiB   | <1%            | 200 m       | 400 m     |
| fh-statsd       | 75 MiB         | 200 MiB     | 400 MiB   | <1%            | 200 m       | 400 m     |
| mongodb         | 185 MiB        | 200 MiB     | 1000 MiB  | <1%            | 200 m       | 1000 m    |

## 9.4. ADJUSTING RESOURCE USAGE OF CLOUD APPS

The resource requests and limits of Cloud Apps can be set the same way as for MBaaS components. There is no particular guideline for doing the adjustment in Cloud Apps.

### 9.4.1. Overview of Resource Usage of Cloud App Components

| Cloud App component | Idle RAM usage | RAM request | RAM limit | Idle CPU usage | CPU request | CPU limit |
|---------------------|----------------|-------------|-----------|----------------|-------------|-----------|
| nodejs-frontend     | 125 MiB        | 90 MiB      | 250 MiB   | <1%            | 100 m       | 500 m     |
| redis               | 8 MiB          | 100 MiB     | 500 MiB   | <1%            | 100 m       | 500 m     |

| Cloud App component | Idle RAM usage | RAM request | RAM limit | Idle CPU usage | CPU request | CPU limit |
|---------------------|----------------|-------------|-----------|----------------|-------------|-----------|
|---------------------|----------------|-------------|-----------|----------------|-------------|-----------|

## 9.5. SETTING RESOURCE REQUESTS AND LIMITS

The procedure for setting the resource requests and limits is the same for both MBaaS components and Cloud App components.

1. Identify the name of the deployment configuration.
  - **MBaaS components:** Names of deployment configurations match the names of components listed in the table in [Section 9.3.2, “Overview of Resource Usage of MBaaS Components”](#).
  - **Cloud apps:** Find the OpenShift project for your RHMAP environment and edit the deployment configuration for your app in there.
2. Open the editor for the deployment configuration, for example **fh-mbaas**:

```
oc edit dc fh-mbaas
```

3. Edit the **requests** and **limits**.

The DeploymentConfig contains two **resources** sections with equivalent values: one in the **spec.strategy** section, and another in the **spec.template.spec.containers** section. Set the **cpu** and **memory** values of **requests** and **limits** as necessary, making sure the values stay equivalent between the two sections, and save the file.

```
apiVersion: v1
kind: DeploymentConfig
metadata:
 name: fh-mbaas
 ...
spec:
 ...
 strategy:
 resources:
 limits:
 cpu: 800m
 memory: 800Mi
 requests:
 cpu: 200m
 memory: 200Mi
 ...
 spec:
 containers:
 ...
 resources:
 limits:
 cpu: 800m
 memory: 800Mi
```

```
requests:
 cpu: 200m
 memory: 200Mi
```



## NOTE

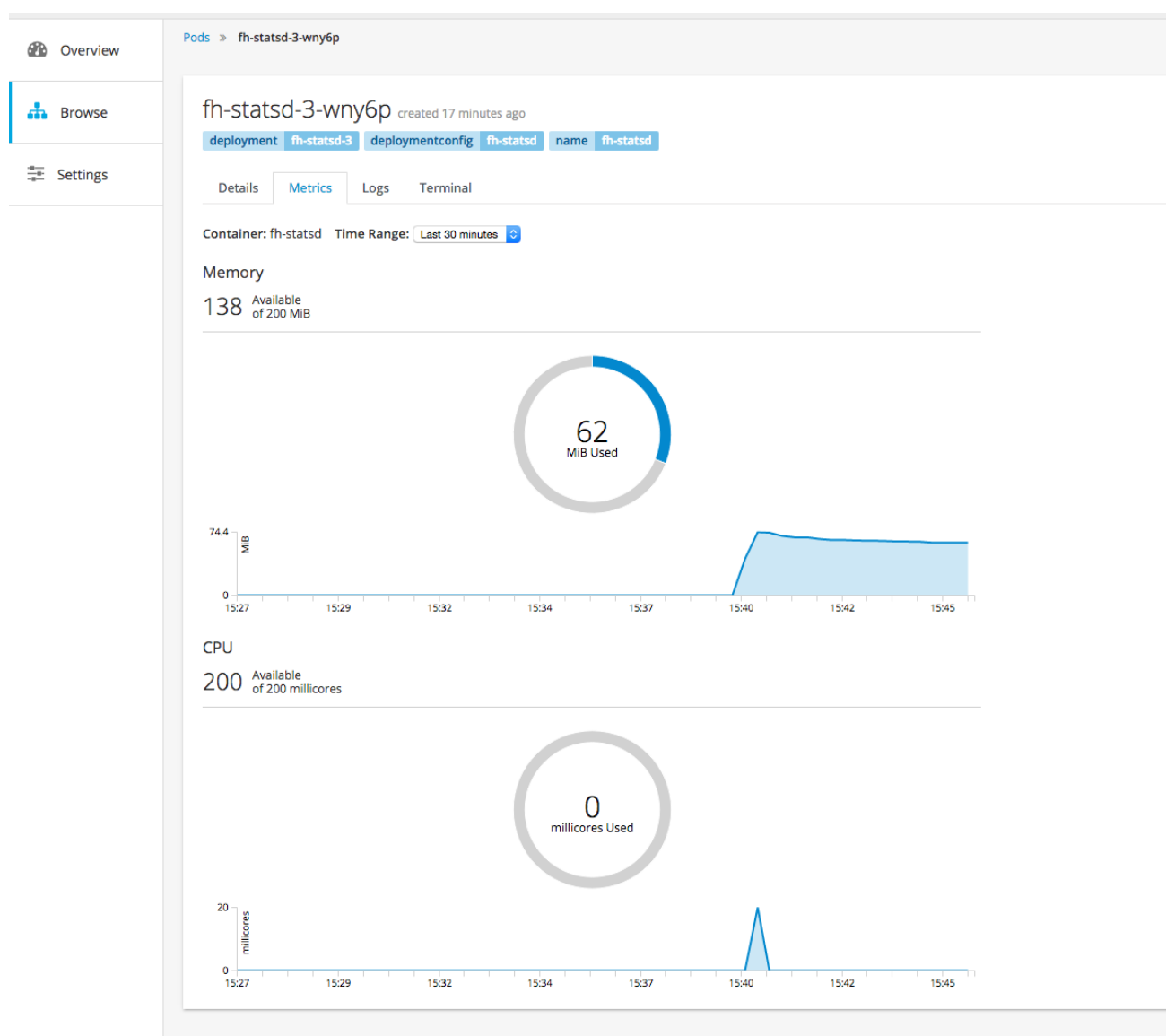
Changing the deployment configuration triggers a new deployment. Once the deployment is complete, the resource limits are updated.

For more information on resources, see [Deployment Resources](#) in the OpenShift documentation.

## 9.6. USING CLUSTER METRICS TO VISUALIZE RESOURCE CONSUMPTION

It is possible to view the immediate and historical resource usage of pods and containers in the form of donut charts and line charts using the *Cluster Metrics* deployment in OpenShift. Refer to [Enabling Cluster Metrics](#) in the OpenShift documentation for steps to enable cluster metrics.

Once cluster metrics are enabled, in the OpenShift web console, navigate to *Browse > Pods* and click on the component of interest. Click on the *Metrics* tab to see the visualizations.



## CHAPTER 10. SETTING UP SMTP FOR CLOUD APP ALERTS

### 10.1. OVERVIEW

Each Cloud App can automatically send alerts by e-mail when specified events occur, such as when the Cloud App gets deployed, undeployed, or logs an error. See [Alerts & Email Notifications](#) for more information.

For the RHMAP 4.x MBaaS based on OpenShift 3, the e-mail function is not available immediately after installation. You must configure an SMTP server to enable e-mail support.

### 10.2. PREREQUISITES

- An RHMAP 4.x MBaaS running in OpenShift Enterprise 3
- An account on an SMTP server through which notification alerts can be sent
- An email address where alerts should be sent
- A deployed Cloud App

### 10.3. CONFIGURING SMTP SETTINGS IN FH-MBAAS

The **FH\_EMAIL\_SMTP** and **FH\_EMAIL\_ALERT\_FROM** environment variables in the fh-mbaas DeploymentConfig need to be set, using the below commands:

```
oc project <mbaas-project-id>
oc env dc/fh-mbaas FH_EMAIL_SMTP="smtps://username:password@localhost"
FH_EMAIL_ALERT_FROM="rhmap-admin@example.com"
```

After modifying the DeploymentConfig, a redeploy of the fh-mbaas pod should be triggered automatically. Once the pod is running again, you can verify the changes.

### 10.4. VERIFYING SMTP SETTINGS FOR MBAAS

1. In the Studio, navigate to a deployed Cloud App.
2. Go to the *Notifications > Alerts* section.
3. Click *Create An Alert*.
4. In the *Emails* field, enter your e-mail address.
5. Click *Test Emails*.

You should receive an e-mail from the e-mail address set as **FH\_EMAIL\_ALERT\_FROM**.

### 10.5. TROUBLESHOOTING

If the test email fails to send, verify the SMTP settings in the running fh-mbaas Pod.

```
oc env pod -l name=fh-mbaas --list | grep EMAIL
```

It may help to view the fh-mbaas logs while attempting to send an email, looking for any errors related to SMTP or email.

```
oc logs -f fh-mbaas-<deploy-uuid>
```

Ensure the Cloud App you are using to send a test mail with is running. If the test email sends OK, but fails to arrive, check it hasn't been placed in your spam or junk folder.



## CHAPTER 11. BACKING UP AN MBAAS

You can back up an MBaaS by following this procedure. After completing the procedure and storing the backup data safely, you can restore an MBaaS to the state at the time of backup.

### 11.1. REQUIREMENTS

- A self-managed MBaaS installation on an OpenShift platform
- A local installation of the **oc** binary
- The **oc** binary has a logged in user on the platform you wish to back up
- The **oc** binary has a logged in user with permission to run the **oc get pc** command

### 11.2. EXPECTED STORAGE REQUIREMENTS

As most of the data being backed up is the data stored in the platform's Cloud Apps, the amount of backup storage space required is proportional to the amount of data stored by the Cloud Apps.

Other factors that have an impact on how much storage is required for backups include:

- how often you backup
- what compression is used
- the length of time you store the backups

### 11.3. WHAT DATA IS BACKED UP

You must back up the following items to back up an MBaaS:

- MongoDB replica set data
- Nagios historical data

### 11.4. BACKING UP THE MONGODB DATA

Back up the MongoDB data using the **mongodump** command in combination with the **oc exec** command:

```
oc exec `oc get po --selector='deploymentconfig=mongodb-1' --template="{{(index .items 0).metadata.name}}"` bash -- -c '/opt/rh/rh-mongodb32/root/usr/bin/mongodump -u admin -p ${MONGODB_ADMIN_PASSWORD} --gzip --archive' > ./mbaas_mongodb.gz
```

### 11.5. BACKING UP THE NAGIOS DATA

Back up the Nagios data by copying the files from the Nagios pod using the **oc exec** command:

```
oc exec <nagios-pod-name> bash -- -c 'tar -zcf - /var/log/nagios/' > nagios-backup.tar.gz
```

For example, if the <nagios-pod-name> is nagios-1:

```
oc exec nagios-1 bash -- -c 'tar -zcf - /var/log/nagios/' > nagios-
backup.tar.gz
```

## 11.6. BACKUP FREQUENCY

Red Hat recommends backing up at least once per day, but you might decide to back up critical data more frequently.

## 11.7. EXAMPLE BACKUP SCRIPT

The example script below shows how you can back up an MBaaS from a backup server, assuming that the user on the backup server has permission to execute commands via the **oc** binary:

```
ts=$(date +%y-%m-%d-%H-%M-%S)
Backup the Mongo services
project=mbaas
for pod in $(oc get pods -n $project | grep mongodb-[0-9]-[0-9]-[0-9a-zA-Z] | awk '{print $1}'); do
 service=$(echo $pod | cut -f1,2 -d"-");
 oc exec $pod bash -- -c '/opt/rh/rh-mongodb32/root/usr/bin/mongodump -
u admin -p ${MONGODB_ADMIN_PASSWORD} --gzip --archive' >
$project-$service-$ts.gz
done

Backup the Nagios service
oc exec -n $project <nagios-pod-name> bash -- -c 'tar -zcf -
/var/log/nagios/' > nagios-backup-$ts.tar.gz
```

## CHAPTER 12. RESTORING AN MBAAS BACKUP

### 12.1. OVERVIEW

You can back up an MBaaS by following the [Backing up an MBaaS](#) procedure. Once you have created the backup, you can restore a MBaaS to the state at the time of backup.

### 12.2. REQUIREMENTS

- A self-managed MBaaS installation on an OpenShift platform
- The **oc** binary has a logged in user with permission to edit deployment configurations and view persistent volumes.
- A backup of the data for mongodb and Nagios services as described in [Backing up an MBaaS](#).

### 12.3. WHAT DATA IS RESTORED

- Mongodb replicaset data
- Nagios historical data

### 12.4. RESTORING NAGIOS DATA

1. Locate the Nagios persistent volume claim-name from the Nagios pod:

```
oc describe pod <nagios-pod-name> | grep "ClaimName:"
claimName: <claim-name>
```

2. Determine the volume-name using the claim-name from step 1:

```
oc describe pvc <claim-name> | grep "Volume:"
Volume: <volume-name>
```

3. Using the volume-name, run the following command:

```
oc describe pv <volume-name>
```

The output contains a **Source** section, similar to:

```
Source:
 Type: NFS (an NFS mount that lasts the lifetime of a pod)
 Server: nfs-server.local
 Path: /path/to/nfs/directory
 ReadOnly: false
```

This information describes where the data for the Nagios persistent volume is located.

4. Stop the Nagios pod by editing the Nagios deployment config and set **replicas** to **0**:

```
oc edit dc nagios
```

Change replicas to 0:

```
spec:
 replicas: 0
```

5. Restore the Nagios data by deleting the contents of the pod's current persistent volume, then extract your backed up data into that directory. Ensure that after extraction the **status.dat** and related files are at the root of the persistent volume.
6. Once the restore is complete, start the Nagios pod by editing the Nagios deployment config and set **replicas** to **1**:

```
oc edit dc nagios
```

Change replicas back to 1:

```
spec:
 replicas: 1
```

## 12.5. RESTORING MONGODB DATA

To restore the MongoDB data, the replicaset must be operational. If this is not the case, see the [MongoDB doesn't respond after repeated installation of the MBaaS](#) section of this guide.

Should you encounter issues with MongoDB, see [Troubleshooting MongoDB Issues](#).

1. Locate the primary MongoDB node. You must restore using the primary member of the MongoDB replicaset, you can find this by logging into any of the MongoDB nodes via remote shell:

```
oc rsh <mongodb-pod-name>
```

Connect to the mongo shell in this pod:

```
mongo admin -u admin -p ${MONGODB_ADMIN_PASSWORD}
```

Check the replicaset status:

```
rs.status()
```

The node with the "stateStr" key set to **PRIMARY** is the node to use in the following restoration procedure.

2. Upload MongoDB data to the primary pod using the **oc rsync** command:

```
oc rsync /path/to/backup/directory/ <primary-mongodb-pod>:/tmp/backup
```

### TIP

rsync copies everything in a directory. To save time put the MongoDB data files into an empty directory before using rsync.

3. Import data into the Mongodb replicaset using the 'mongorestore' tool from inside the mongodb pod. Log into the pod:

```
oc rsh <mongodb-pod-name>
```

Restore the data:

```
mongorestore -u admin -p ${MONGODB_ADMIN_PASSWORD} --gzip --
archive=/tmp/backup/mongodb-backup.gz
```

For more information on this tool, see the [mongorestore documentation](#).

## 12.6. CONFIRMING RESTORATION

Log in to the Studio and ensure that all the projects, environments and MBaaS targets are correctly restored.

## CHAPTER 13. TROUBLESHOOTING THE RHMAP MBAAS

### 13.1. OVERVIEW

This document provides information on how to identify, debug, and resolve possible issues that can be encountered during installation and usage of the RHMAP MBaaS on OpenShift 3.

In [Common Problems](#), you can see a list of resolutions for problems that might occur during installation or usage of the MBaaS.

### 13.2. CHECK THE HEALTH ENDPOINT OF THE MBAAS

The first step to check whether an MBaaS is running correctly is to see the output of its health endpoint. The HTTP health endpoint in the MBaaS reports the health status of the MBaaS and of its individual components.

From the command line, run the following command:

```
curl https://<MBaaS URL>/sys/info/health
```

To find the MBaaS URL:

1. Navigate to the MBaaS that you want to check, by selecting **Admin**, then **MBaaS Targets**, and then choosing the MBaaS.
2. Copy the value of the **MBaaS URL** field from the MBaaS details screen.

If the MBaaS is running correctly without any errors, the output of the command should be similar to the following, showing a **"test\_status": "ok"** for each component:

```
{
 "status": "ok",
 "summary": "No issues to report. All tests passed without error",
 "details": [
 {
 "description": "Check fh-statsd running",
 "test_status": "ok",
 "result": {
 "id": "fh-statsd",
 "status": "OK",
 "error": null
 },
 "runtime": 6
 },
 ...
]
}
```

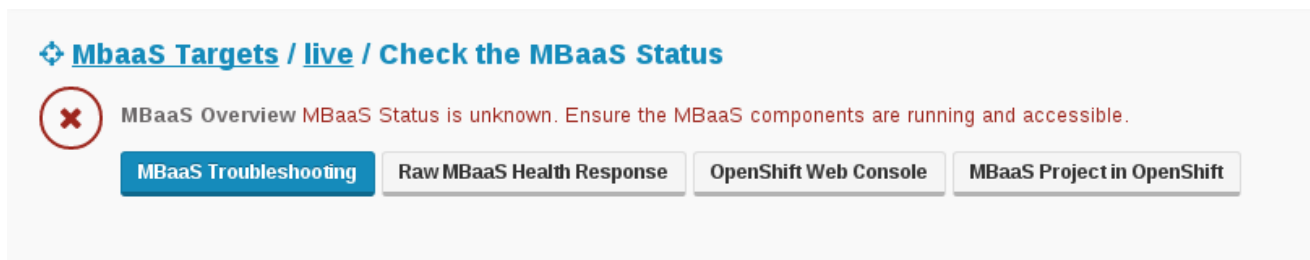
If there are any errors, the output will contain error messages in the **result.error** object of the **details** array for individual components. Use this information to identify which component is failing and to get information on further steps to resolve the failure.

You can also see a HTTP *503 Service Unavailable* error returned from the health endpoint. This can happen in several situations:

- The MBaaS hasn't finished deploying.
- The URL of the MBaaS is not reachable on port 80. Check your network configuration.
- Provisioning of the MBaaS has failed. See the [Prerequisites](#) and [Before the Installation](#) (for manual installation) sections of the [Provisioning an MBaaS in Red Hat OpenShift Enterprise 3](#) guide and make sure your OpenShift cluster fulfills all the listed requirements.

Alternatively, you can see the result of this health check in the Studio. Navigate to the *Admin > MBaaS Targets* section, select your MBaaS target, and click *Check the MBaaS Status*.

If there is an error, you are presented with a screen showing the same information as described above. Use the provided links to *OpenShift Web Console* and the associated *MBaaS Project in OpenShift* to help with debugging of the issue on the OpenShift side.



## 13.3. ANALYZE LOGS

To see the logging output of individual MBaaS components, you must configure centralized logging in your OpenShift cluster. See [Enabling Centralized Logging](#) for a detailed procedure.

The section [Identifying Issues in an MBaaS](#) provides guidance on discovering MBaaS failures by searching and filtering its logging output.

## 13.4. COMMON PROBLEMS

### 13.4.1. A replica pod of mongodb-service is replaced with a new one

#### 13.4.1.1. Summary

The replica set is susceptible to down time if the replica set members configuration is not up to date with the actual set of pods. There must be at least two members active at any time, in order for an election of a primary member to happen. Without a primary member, the MongoDB service won't perform any read or write operations.

A MongoDB replica may get terminated in several situations:

- A node hosting a MongoDB replica is terminated or evacuated.
- A re-deploy is triggered on one of the MongoDB Deployment objects in the project – manually or by a configuration change.
- One of the MongoDB deployments is scaled down to zero pods, then scaled back up to one pod.

To learn more about replication in MongoDB, see [Replication](#) in the official MongoDB documentation.

#### 13.4.1.2. Fix

The following procedure shows you how to re-configure a MongoDB replica set into a fully operational state. You must synchronize the list of replica set members with the actual set of MongoDB pods in the cluster, and set a primary member of the replica set.

1. Note the MongoDB endpoints.

```
oc get ep | grep mongo
```

Make note of the list of endpoints. It is used later to set the replica set members configuration.

| NAME      | ENDPOINTS        |
|-----------|------------------|
| AGE       |                  |
| mongodb-1 | 10.1.2.152:27017 |
| 17h       |                  |
| mongodb-2 | 10.1.4.136:27017 |
| 17h       |                  |
| mongodb-3 | 10.1.5.16:27017  |
| 17h       |                  |

2. Log in to the oldest MongoDB replica pod.  
List all the MongoDB replica pods.

```
oc get po -l name=mongodb-replica
```

In the output, find the pod with the highest value in the **AGE** field.

| NAME              | READY | STATUS  | RESTARTS | AGE |
|-------------------|-------|---------|----------|-----|
| mongodb-1-1-4nsrv | 1/1   | Running | 0        | 19h |
| mongodb-2-1-j4v3x | 1/1   | Running | 0        | 3h  |
| mongodb-3-2-7tezv | 1/1   | Running | 0        | 1h  |

In this case, it is **mongodb-1-1-4nsrv** with an age of 19 hours.

Log in to the pod using **oc rsh**.

```
oc rsh mongodb-1-1-4nsrv
```

3. Open a MongoDB shell on the primary member.

```
mongo admin -u admin -p ${MONGODB_ADMIN_PASSWORD} --host
${MONGODB_REPLICA_NAME}/localhost
```

```
MongoDB shell version: 2.4.9
connecting to: rs0/localhost:27017/admin
[...]
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
Questions? Try the support group
http://groups.google.com/group/mongodb-user
rs0:PRIMARY>
```



4. List the configured members.

Run **rs.conf()** in the MongoDB shell.

```
rs0:PRIMARY> rs.conf()
{
 "_id" : "rs0",
 "version" : 56239,
 "members" : [
 {
 "_id" : 3,
 "host" : "10.1.0.2:27017"
 },
 {
 "_id" : 4,
 "host" : "10.1.1.2:27017"
 },
 {
 "_id" : 5,
 "host" : "10.1.6.4:27017"
 }
]
}
```

5. Ensure all hosts have either **PRIMARY** or **SECONDARY** status.

Run the following command. It may take several seconds to complete.

```
rs0:PRIMARY> rs.status().members.forEach(function(member)
{print(member.name + ' :: ' + member.stateStr)})
```

```
mongodb-1:27017 :: PRIMARY
mongodb-2:27017 :: SECONDARY
mongodb-3:27017 :: SECONDARY
rs0:PRIMARY>
```

There must be exactly one **PRIMARY** node. All the other nodes must be **SECONDARY**. If a member is in a **STARTUP**, **STARTUP2**, **RECOVERING**, or **UNKNOWN** state, try running the above command again in a few minutes. These states may signify that the replica set is performing a startup, recovery, or other procedure potentially resulting in an operational state.

### 13.4.1.3. Result

After applying the fix, all three MongoDB pods will be members of the replica set. If one of the three members terminates unexpectedly, the two remaining members are enough to keep the MongoDB service fully operational.

## 13.4.2. MongoDB doesn't respond after repeated installation of the MBaaS

### 13.4.2.1. Summary

**NOTE**

The described situation can result from an attempt to create an MBaaS with the same name as a previously deleted one. We suggest you use unique names for every MBaaS installation.

If the **mongodb-service** is not responding after the installation of the MBaaS, it is possible that some of the MongoDB replica set members failed to start up. This can happen due to a combination of the following factors:

- The most likely cause of failure in MongoDB startup is the presence of a **mongod.lock** lock file and journal files in the MongoDB data folder, left over from an improperly terminated MongoDB instance.
- If a MongoDB pod is terminated, the associated persistent volumes transition to a *Released* state. When a new MongoDB pod replaces a terminated one, it may get attached to the same persistent volume which was attached to the terminated MongoDB instance, and thus get exposed to the files created by the terminated instance.

**13.4.2.2. Fix****NOTE**

SSH access and administrator rights on the OpenShift master and the NFS server are required for the following procedure.

**NOTE**

This procedure describes a fix only for persistent volumes backed by NFS. Refer to [Configuring Persistent Storage](#) in the official OpenShift documentation for general information on handling other volume types.

The primary indicator of this situation is the **mongodb-initiator** pod not reaching the *Completed* status.

Run the following command to see the status of **mongodb-initiator**:

```
oc get pod mongodb-initiator
```

| NAME              | READY | STATUS  | RESTARTS | AGE |
|-------------------|-------|---------|----------|-----|
| mongodb-initiator | 1/1   | Running | 0        | 5d  |

If the status is any other than *Completed*, the MongoDB replica set is not created properly. If **mongodb-initiator** stays in this state too long, it may be a signal that one of the MongoDB pods has failed to start. To confirm whether this is the case, check logs of **mongodb-initiator** using the following command:

```
oc logs mongodb-initiator
```

```
=> Waiting for 3 MongoDB endpoints ...mongodb-1
mongodb-2
mongodb-3
```

```
=> Waiting for all endpoints to accept connections...
```

If the above message is the last one in the output, it signifies that some of the MongoDB pods are not responding.

Check the event log by running the following command:

```
oc get ev
```

If the output contains a message similar to the following, you should continue with the below procedure to clean up the persistent volumes:

```
FailedMount {kubelet ip-10-0-0-100.example.internal} Unable to mount
volumes for pod "mongodb-1-1-example-mbaas": Mount failed: exit status 32
```

The following procedure will guide you through the process of deleting contents of existing persistent volumes, creating new persistent volumes, and re-creating persistent volume claims.

#### 1. Find the NFS paths.

On the OpenShift master node, execute the following command to find the paths of all persistent volumes associated with an MBaaS. Replace **<mbaas-project-name>** with the name of the MBaaS project in OpenShift.

```
list=$(oc get pv | grep <mbaas-project-name> | awk '{ print $1}');
for pv in ${list[@]} ; do
 path=$(oc describe pv ${pv} | grep Path: | awk '{print $2}' | tr -
d '\r')
 echo ${path}
done
```

Example output:

```
/nfs/exp222
/nfs/exp249
/nfs/exp255
```

#### 2. Delete all contents of the found NFS paths.

Log in to the NFS server using **ssh**.

Execute the following command to list contents of the paths. Replace **<NFS paths>** with the list of paths from the previous step, separated by spaces.

```
for path in <NFS paths> ; do
 echo ${path}
 sudo ls -l ${path}
 echo " "
done
```

Example output:

```
/nfs/exp222
-rw-----. 1001320000 nfsnobody admin.0
-rw-----. 1001320000 nfsnobody admin.ns
-rw-----. 1001320000 nfsnobody fh-mbaas.0
```

```
-rw-----. 1001320000 nfsnobody fh-mbaas.ns
-rw-----. 1001320000 nfsnobody fh-metrics.0
-rw-----. 1001320000 nfsnobody fh-metrics.ns
-rw-----. 1001320000 nfsnobody fh-reporting.0
-rw-----. 1001320000 nfsnobody fh-reporting.ns
drwxr-xr-x. 1001320000 nfsnobody journal
-rw-----. 1001320000 nfsnobody local.0
-rw-----. 1001320000 nfsnobody local.1
-rw-----. 1001320000 nfsnobody local.ns
-rwxr-xr-x. 1001320000 nfsnobody mongod.lock
drwxr-xr-x. 1001320000 nfsnobody _tmp
```

```
/nfs/exp249
```

```
drwxr-xr-x. 1001320000 nfsnobody journal
-rw-----. 1001320000 nfsnobody local.0
-rw-----. 1001320000 nfsnobody local.ns
-rwxr-xr-x. 1001320000 nfsnobody mongod.lock
drwxr-xr-x. 1001320000 nfsnobody _tmp
```

```
/nfs/exp255
```

```
drwxr-xr-x. 1001320000 nfsnobody journal
-rw-----. 1001320000 nfsnobody local.0
-rw-----. 1001320000 nfsnobody local.ns
-rwxr-xr-x. 1001320000 nfsnobody mongod.lock
drwxr-xr-x. 1001320000 nfsnobody _tmp
```



### WARNING

Make sure to back up all data before proceeding. The following operation may result in irrecoverable loss of data.

If the listed contents of the paths resemble the output shown above, delete all contents of the found NFS paths. Replace **<NFS paths>** with the list of paths from step 1, separated by spaces.

```
for path in <NFS paths>
do
 if [-z ${path+x}]
 then
 echo "path is unset"
 else
 echo "path is set to '$path'"
 cd ${path} && rm -rf ./*
 fi
done
```

3. Re-create persistent volumes.  
Log in to the OpenShift master node using **ssh**.

Navigate to the directory which contains the YAML files that were used to create the persistent volumes.

Execute the following command to delete and re-create the persistent volumes. Replace **<mbaas-project-name>** with the name of the MBaaS project in OpenShift.

```
list=$(oc get pv | grep <mbaas-project-name> | awk '{ print $1}');
for pv in ${list}; do
 oc delete pv ${pv}
 oc create -f ${pv}.yaml
done
```

The persistent volumes are now re-created and in *Available* state.



## NOTE

The re-created persistent volumes will not be used by OpenShift again for the same persistent volume claims. Make sure you have at least three additional persistent volumes in *Available* state.

4. Re-create persistent volume claims for MongoDB.  
Create three JSON files, with the following names:

- **mongodb-claim-1.json**
- **mongodb-claim-2.json**
- **mongodb-claim-3.json**

Copy the following contents into each file. Change the **metadata.name** value to match the name of the file without the suffix. For example, the contents for the **mongodb-claim-1.json** file are as follows:

```
{
 "kind": "PersistentVolumeClaim",
 "apiVersion": "v1",
 "metadata": {
 "name": "mongodb-claim-1"
 },
 "spec": {
 "accessModes": ["ReadWriteOnce"],
 "resources": {
 "requests": {
 "storage": "50Gi"
 }
 }
 }
}
```

Run the following command to re-create the persistent volume claims.

```
for pvc in mongodb-claim-1 mongodb-claim-2 mongodb-claim-3; do
 oc delete pvc ${pvc}
 oc create -f ${pvc}.json
```

```
done
```

5. Verify that **mongodb-initiator** proceeds with initialization.

Run the following command to see the logs of **mongodb-initiator**.

```
oc logs mongodb-initiator -f
```

After **mongodb-initiator** completes its work, the log output should contain the following message, indicating that the MongoDB replica set was successfully created.

```
=> Successfully initialized replSet
```

### 13.4.2.3. Result

The MongoDB service is fully operational with all three replicas attached to their persistent volumes. The persistent volumes left in *Released* state from the previous installation are now in the *Available* state, ready for use by other persistent volume claims.

## 13.4.3. MongoDB replica set stops replicating correctly

### 13.4.3.1. Summary

If some of the MBaaS components start to crash, this may be because they can not connect to a primary member in the MongoDB replica set. This usually indicates that the replica set configuration has become inconsistent. This can happen if a majority of the member pods get replaced and have new IP addresses. In this case, data cannot be written to or read from MongoDB replica set in the MBaaS project.

To verify the replica set state as seen by each member, run the following command in the shell of a user logged in to OpenShift with access to the MBaaS project:

```
for i in `oc get po -a | grep -e "mongodb-[0-9]\+" | awk '{print $1}'`;
do
 echo "## ${i} ##"
 echo mongo admin -u admin -p \${MONGODB_ADMIN_PASSWORD} --eval
 "printjson(rs.status\(\)\)" | oc rsh --shell='/bin/bash' $i
done
```

For a fully consistent replica set, the output for each member would contain a **members** object listing details about each member. If the output resembles the following, containing the **"ok" : 0** value for some members, proceed to the fix in order to make the replica set consistent.

```
mongodb-1-1-8syid
MongoDB shell version: 2.4.9
connecting to: admin
{
 "startupStatus" : 1,
 "ok" : 0,
 "errmsg" : "loading local.system.replset config (LOADINGCONFIG)"
}
mongodb-2-1-m6ao1
MongoDB shell version: 2.4.9
connecting to: admin
{
```

```

 "startupStatus" : 1,
 "ok" : 0,
 "errmsg" : "loading local.system.replset config (LOADINGCONFIG)"
 }
mongodb-3-2-e0a11
MongoDB shell version: 2.4.9
connecting to: admin
{
 "startupStatus" : 1,
 "ok" : 0,
 "errmsg" : "loading local.system.replset config (LOADINGCONFIG)"
}

```

### 13.4.3.2. Fix

You can make the replica set consistent by forcing a re-deploy.

1. Note the MongoDB endpoints which are in an error status.

```
oc get po
```

Example:

```

```bash
NAME                                READY    STATUS    RESTARTS   AGE
mongodb-1-1-pu0fz                  1/1      Error      0           1h
```

```

2. Force a deploy of this Pod

```
oc deploy mongodb-1 --latest
```

### 13.4.3.3. Result

The replica starts replicating properly again and dependent MBaaS components start working again.

## 13.4.4. An MBaaS component fails to start because no suitable nodes are found

### 13.4.4.1. Summary

If some of the MBaaS components are not starting up after the installation, it may be the case that the OpenShift scheduler failed to find suitable nodes on which to schedule the pods of those MBaaS components. This means that the OpenShift cluster doesn't contain all the nodes required by the MBaaS OpenShift template, or that those nodes don't satisfy the requirements on system resources, node labels, and other parameters.

Read more about the OpenShift [Scheduler](#) in the OpenShift documentation.

To verify that this is the problem, run the following command to list the event log:

```
oc get ev
```

If the output contains one of the following messages, you are most likely facing this problem – the nodes in your OpenShift cluster don't fulfill some of the requirements.

- **Failed for reason MatchNodeSelector and possibly others**
- **Failed for reason PodExceedsFreeCPU and possibly others**

#### 13.4.4.2. Fix

To fix this problem, configure nodes in your OpenShift cluster to match the requirements of the MBaaS OpenShift template.

- Apply correct labels to nodes.  
Refer to [Apply Node Labels](#) in the guide *Provisioning an MBaaS in Red Hat OpenShift Enterprise 3* for details on what labels must be applied to nodes.
- Make sure the OpenShift cluster has sufficient resources for the MBaaS components, Cloud Apps, and Cloud Services it runs.  
Configure the machines used as OpenShift nodes to have more CPU power and internal memory available, or add more nodes to the cluster. Refer to the guide on [Overcommitting](#) and [Compute Resources](#) in the OpenShift documentation for more information on how containers use system resources.
- Clean up the OpenShift instance.  
Delete unused projects from the OpenShift instance.

Alternatively, it is also possible to correct the problem from the other side — change the deployment configurations in the MBaaS OpenShift template to match the setup of your OpenShift cluster.



#### WARNING

Changing the deployment configurations may negatively impact the performance and reliability of the MBaaS. Therefore, this is not a recommended approach.

To list all deployment configurations, run the following command:

```
oc get dc
```

| NAME         | TRIGGERS     | LATEST |
|--------------|--------------|--------|
| fh-mbaas     | ConfigChange | 1      |
| fh-messaging | ConfigChange | 1      |
| fh-metrics   | ConfigChange | 1      |
| fh-statsd    | ConfigChange | 1      |
| mongodb-1    | ConfigChange | 1      |
| mongodb-2    | ConfigChange | 1      |
| mongodb-3    | ConfigChange | 1      |

To edit a deployment configuration, use the **oc edit dc <deployment>** command. For example, to edit the configuration of the **fh-mbaas** deployment, run the following command:

```
■
```



```
oc edit dc fh-mbaas
```

You can modify system resource requirements in the **resources** sections.

```
...
resources:
 limits:
 cpu: 800m
 memory: 800Mi
 requests:
 cpu: 200m
 memory: 200Mi
...
```

Changing a deployment configuration triggers a deployment operation.

#### 13.4.4.3. Result

If you changed the setup of nodes in the OpenShift cluster to match the requirements of the MBaaS OpenShift template, the MBaaS is now fully operational without any limitation to quality of service.

If you changed the deployment configuration of any MBaaS component, the cluster should now be fully operational, with a potential limitation to quality of service.

#### 13.4.4.4. Result

You can now deploy your app to the RHMAP MBaaS.