Red Hat Mobile Application Platform 4.7

Troubleshooting Guide

For Red Hat Mobile Application Platform 4.7

Last Updated: 2018-10-09

Red Hat Mobile Application Platform 4.7 Troubleshooting Guide

For Red Hat Mobile Application Platform 4.7

Legal Notice

Abstract

This document provides troubleshooting procedures for common RHMAP problems.

Table of Contents

CHAPTER 1. TROUBLESHOOTING RHMAP

If you encounter an issue with RHMAP, use the following procedure to help solve the issue:

1. Make sure all the Nagios checks are passing. If not, see Troubleshooting Core and MBaaS using Nagios.

2. Make sure that template apps are building and working correctly. Red Hat provide many template apps that can verify your configuration and network access. If a simple template app works, try testing with a complex template app that most closely resembles your App.

3. If the template apps work, but your App is not working, check for any known issues and workarounds in the Release Notes and on the Red Hat support site.

CHAPTER 2. TROUBLESHOOTING THE RHMAP CORE

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 Core on OpenShift 3.

2.1. CHECKING THE NAGIOS DASHBOARD

The first step to check whether the Core is running correctly is to check whether the Nagios dashboard is reporting any issues, see how to access Nagios. You will find more information on how to troubleshoot any issues discovered by Nagios in the Nagios troubleshooting information.

2.2. ANALYZING THE CORE LOGS

To see the logging output of individual Core components, you must configure centralized logging in your OpenShift cluster. See Centralized Logging for Core Components for a detailed procedure.

The section Identifying Issues in a Core provides guidance on discovering Core failures by searching and filtering its logging output.

The section Identifying Issues in an MBaaS provides guidance in discovering if the issues you are facing are originating in the MBaaS.

2.3. TROUBLESHOOTING COMMON PROBLEMS

OpenShift Cleans Persistent Volume too Early

When a pod is started that takes some time to get running, the automatic PV cleaner can see the unused mount points in the pod and delete them even though are in fact going to be used in the future.

This issue is caused due to Persistent Volume Claims being added to the list of volumes to preserve rather than the actual name of the Persistent Volume associated with the Persistent Volume Claim.

As a result, the periodic cleanup process would unmount the volume if a pod utilizing the Persistent Volume Claim had not yet entered running state.

The issue is logged in Bugzilla and has been marked as closed so should be resolved in a future release of Kubernetes.

Kubernetes Pods Stuck Terminating

For example,

```
oc get po
NAME                    READY     STATUS        RESTARTS   AGE
fh-messaging-1-gzlw6    0/1       Terminating   0          1h
mongodb-1-1-2pqbq       0/1       Terminating   0          1h
mongodb-initiator       1/1       Terminating  3 0          1h
```

Solution is to force delete the pods:

```
$ oc delete pod fh-messaging-1-gzlw6 --grace-period=0
pod "fh-messaging-1-gzlw6" deleted
$ oc delete pod mongodb-1-1-2pqbq --grace-period=0
```

```
pod "mongodb-1-1-2pqbq" deleted
$ oc delete pod mongodb-initiator --grace-period=0
pod "mongodb-initiator" deleted
```

Unable to mount volumes for pod - unsupported volume type

The full error in the oc get events log is:

```
1m   3s  8  mongodb-2-1-qlgwl Pod  FailedMount  {kubelet 10.10.0.99}
Unable to mount volumes for pod "mongodb-2-1-qlgwl_qe-3node-
rhmap4(f7898df8-41ce-11e6-9064-0abb8905d551)": unsupported volume type
1m   3s  8  mongodb-2-1-qlgwl Pod  FailedSync   {kubelet 10.10.0.99} Error
syncing pod, skipping: unsupported volume type
```

The problem here is the PVC is being bound to an already bound Persistent Volume (in another namespace)

```
$ oc get pvc mongodb-claim-2 -n qe-3node-rhmap4
mongodb-claim-2   <none>    Bound     pveleven   50Gi       RWO
3h

$ oc get pv pveleven
NAME       CAPACITY   ACCESSMODES   STATUS    CLAIM
REASON    AGE
pveleven   50Gi       RWO           Bound     qe-3node-rhmap410-
1/mongodb-claim-2             4d
```

Notice the CLAIM is a different namespace i.e qe-3node-rhmap410-1, not qe-3node-rhmap4. This is usually the result of a project being created with a name that already existed, but was deleted. When deleting a project in OpenShift, it doesn't do a hard delete straight away. Rather, it marks a bunch of resources for deletion. This can cause unusual behaviour (such as this) if some resources still exist when the new project with the same name is created, and kubernetes thinks the resources marked for deletion actually belong to the new project.

The solution in this case was to remove the PVC, then recreate it, allowing it to Bind to a new PV.

```
oc export pvc mongodb-claim-2 -o yaml > mongodb-claim-2.yaml
oc delete pvc mongodb-claim-2
oc create -f ./mongodb-claim-2.yaml
```

Studio Reports "no bizobjects exist"

After installation or upgrade, you might see an error **no bizobjects exist**. To resolve this issue, redeploy fh-aaa and millicore as follows:

1. Redeploy fh-aaa and wait for that process to complete:

   ```
   oc deploy dc/fh-aaa --latest
   ```

2. After the deployment is complete, redeploy millicore:

   ```
   oc deploy dc/millicore --latest
   ```

Projects Fail to Appear in Studio

You might notice projects failing to load in Studio and pop-ups in your browser. If you encounter this issue, restart fh-supercore using the OpenShift Console, or using the following command:

```
oc scale --replicas=0 dc fh-supercore && oc scale --replicas=1 dc fh-supercore
```

2.4. CONTACTING SUPPORT

If all else fails, you can contact Red Hat Support for more assistance. Including a **`fh-system-dump-tool`** report will help the support team arrive at a solution more rapidly.

The **`fh-system-dump-tool`** will search your installation for frequently found issues and also create a tar.gz archive containing all the recent logs and errors it can find. More information on this can be found here.

CHAPTER 3. TROUBLESHOOTING THE RHMAP MBAAS

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.

3.1. CHECKING 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.

3.2. ANALYZING THE MBAAS 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.

3.3. TROUBLESHOOTING COMMON PROBLEMS

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.

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 app=mongodb-3-node-replica-set
```

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.

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.

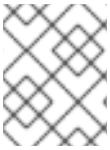
SUMMARY

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.



FIX

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
```

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.

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)"
}
```

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
```

Result

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

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`**

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.

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.

Summary

If your OpenShift installation only has outbound internet access via a proxy server, you may have issues when the node.js builder in OpenShift attempts to do an npm installation. Also, if you have a custom npm registry, you need to configure npm to avoid issues with node.js builder.

To overcome this, create a **`.npmrc`** file in the root directory of your Cloud App. Use this file to specify the proxy location and possibly the npm registry location. For more information, see the npm configuration documentation. For example, to use a http(s) proxy and registry for **`company.com`**, you might create the following **`.npmrc`** file:

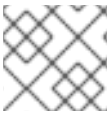
```
proxy=http://proxy.company.com:8080/
https-proxy=http://proxy.company.com:8080
registry=http://registry.company.com/
```

Result

You can now deploy your app to the RHMAP MBaaS.

Summary

If you create an MBaaS target in Studio and the DNS URL does not properly form, the route in the OpenShift project is not created.



NOTE

The DNS URL is set in the 'OpenShift Router DNS' field.

To validate this issue:

1. Open RHMAP Studio
2. Click on **Admin**
3. Click on **MBaaS Targets**
4. Choose your MBaaS
5. Click on **Check the MBaaS Status**
6. Click on **Raw MBaaS Health Response**
7. The output is similar to the following:

```
---
{"userDetail":"getaddrinfo ENOTFOUND",
"systemDetail":"getaddrinfo ENOTFOUND -
{ [Error: getaddrinfo ENOTFOUND] code: 'ENOTFOUND', errno:
'ENOTFOUND', syscall: 'getaddrinfo' }
","code":"ENOTFOUND"}
---
```

Fix

To resolve this issue, create a new MBaaS using the following format for the 'OpenShift Router DNS' field:

```
---
https://*.<domain>
---
```

where <domain> is your domain or subdomain, for example, **https://*.openshift.example.com**.

Result

Check the MBaaS status. The output is similar to the following:

```
---
 "status": "ok",
 "summary": "No issues to report. All tests passed without error.",
 "details": [
---
```

CHAPTER 4. TROUBLESHOOTING CORE AND MBAAS USING NAGIOS

If you encounter issues when working with Core or MBaaS, Nagios can assist with troubleshooting the following

- CPU - CPU check
- Memory - Memory check
- Storage - Disk Usage check
- MongoDB - MongoDB check
- MySQL - MySQL check
- Overall health - Health Check
- Network - Ping Check

NOTE

CPU and Memory are described in the same section

Each Nagios check provides a status of the current situation. For example, the status for the CPU is OK.

There are four different status levels

- OK
- WARNING
- CRITICAL
- UNKNOWN

4.1. TROUBLESHOOTING CPU/MEMORY ISSUES

4.1.1. Reviewing CPU and Memory Status

This section applies to Core and MBaaS components.

The status levels for CPU and Memory are

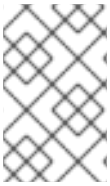
Status	Description
OK	The CPU/Memory usage for all containers is within the set limits
WARNING	One or more of the container's CPU/Memory usage is over the WARNING threshold of 80%

Status	Description
CRITICAL	One or more of the container's CPU/Memory usage is over the CRITICAL threshold of 90%
UNKNOWN	An internal error has occurred and the check was unable to determine a correct status for this check

4.1.2. Resolving CRITICAL and WARNING Issues

The following describes how to check that CPU/Memory usage is within the desired threshold for all containers in the current project.

In order to calculate the current containers CPU/Memory usage, the check needs to run twice in order to gather adequate data and make the calculation. Every check records the current usage and uptime of the process and then compares it to the previous check.



NOTE

This check is only applied to containers that have a CPU/Memory limit set and ignores any containers that do not have a limit set. To monitor a component, it must have a recommended starting limit set for CPU/Memory in the corresponding template.

This check must be executed twice before it will provide CPU/Memory usage for containers. A status of UNKNOWN is displayed for the first check.

An occasional spike in CPU/Memory usage for a particular component might not be a cause for concern, especially if the value slightly exceeds the WARNING threshold. For example, it is common for a higher CPU/Memory usage to be reported on component startup or after an upgrade, in which case no action is required.

If the check is reporting issues, consider one of the following actions

4.1.2.1. Increasing Limits

All RHMAP components are created with a recommended limit for CPU/Memory, but these limits might need to be increased depending on individual usage of the platform.

1. Check the CPU and Memory limits

```
$ oc describe dc/<component>
```

   The output is similar to the following

```
  Limits:
    CPU:      800m
    Memory :   800Mi
```

2. Modify the CPU and Memory limits in the components deploy config. For example to change both limits from 800 to 900

```
$ oc edit dc/<component>
```

Make the following edits

```
spec:
  containers:
  ...
    resources:
      limits:
        CPU: 900m
        Memory: 900Mi
```

You might want to only change one limit (CPU or Memory), depending on the Nagios status of CPU and Memory. Containers are automatically updated after changing the deployment configuration.

3. Verify the new limits

```
$ oc describe dc/<component>
```

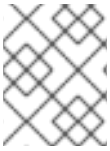
The output is similar to the following

```
Limits:
  CPU:      900m
  Memory:   900Mi
```

4.1.2.2. Troubleshooting Component Container

If a high CPU/Memory usage is **regularly** reported for a component's container, especially after an update, it might indicate a problem with the component. Checking the container's logs can give some indication of internal errors:

```
$ oc logs -f <pod id> -c <container id>
```



NOTE

The <pod id> and <container id> can be retrieved from the status information of the Nagios check

```
CPU example -  OK: mysql-1-8puwv:mysql: - usage: 0.17%
Memory example -  OK: mysql-1-8puwv:mysql: - usage: 57.1%
```

In this example, the <pod id> is **`mysql-1-8puwv`** and the <container id> is **`mysql`**. To scale the component down and up again.

```
$ oc delete pod <component pod id>
$ watch 'oc get pods | grep <component>'
```

The output is similar to the following

```
<component>-2-9xeva    1/1       Running   0          10s
```

4.1.3. Resolving UNKNOWN Issues for CPU

The CPU check must be executed at least twice before it can gather adequate data to calculate the usage of a container. If a container has not previously been checked, usage cannot be calculated and Nagios displays a status of UNKNOWN. Forcing service checks on all containers ensures that all service checks are executed at least twice.

To force host service checks

```
$ oc exec -ti "$(oc get pods | grep nagios | awk '{print $1}')" --
/opt/rhmap/host-svc-check
```

4.1.4. Resolving UNKNOWN Issues for Memory

An UNKNOWN status should never occur and usually indicates an internal error in the check. Contact Support and include the status information displayed for the check.

4.1.5. Resolving Reported High Memory Usage

Millicore loads some of the binary objects into memory and stores them in mysql when completing app builds and also when using the app store. This can spike the reported memory usage. This issue is not related to JVM memory usage, but to the reported container memory usage and will be addressed in a future release.

If the reported memory usage stays high or gets to critical you may want to do a redeploy of millicore.

```
$ oc deploy dc/millicore --latest
$ oc deploy dc/mysql --latest
```

4.2. TROUBLESHOOTING DISK USAGE ISSUES

4.2.1. Reviewing Disk Usage Status

This section applies to Core and MBaaS components.

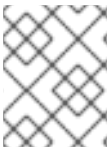
The status levels for Disk Usage are

Status	Description
OK	Successfully retrieve disk usage information for all containers without error and all are under the warning threshold of 80%
WARNING	One or more volumes are over the WARNING threshold of 80%
CRITICAL	One or more volumes are over the CRITICAL threshold of 90%
UNKNOWN	An internal error has occurred and the check was unable to determine a correct status for this check. The error code and message are provided which may help troubleshooting

4.2.2. Resolving CRITICAL and WARNING Issues

The following describes how to check disk usage for **all** volumes that are mounted in all containers and are also within the desired threshold for the current project.

For each volume, retrieve the disk usage from the container as a percentage for both the blocks and inodes. Compare the highest percentage against the threshold to determine the status.



NOTE

Due to the large number of volumes checked, volumes with usage below 10% are not reported in the detail.

4.2.2.1. Increasing Size of Persistent Volume

Many factors can affect storage in OpenShift, the following suggestions can help to resolve common issues:

1. Moving a container to another node with more resources can resolve issues where a non-persistent volume has insufficient space on the node.

2. Redeploying a pod and containers can resolve issues associated with temporary/shared memory volumes.

4.2.2.2. Troubleshooting Component Container

If a particularly high disk usage is continually indicated in a component's container, especially after an update, it may be an indication that there is a problem with the component. Checking the container's logs can give some indication of internal errors.

```
$ oc logs -f <pod id> -c <container id>
```

4.3. TROUBLESHOOTING MONGODB ISSUES

4.3.1. Reviewing MongoDB Status

The following applies to MBaaS only.

The status levels for MongoDB are

Status	Description
OK	The MongoDB replicaset is running, can be connected to, and has a single primary member
WARNING	The service is running, with non-critical issues present
CRITICAL	The service is not functioning correctly
UNKNOWN	No pods running MongoDB were located

4.3.2. Resolving CRITICAL Issues

The following describes how to check that the MongoDB replicaset is correctly configured and healthy. The course of action required is determined by the current status and the information returned by the check. For each individual check, view this information using the Nagios dashboard.

4.3.2.1. Troubleshooting a Failed State

A member has a state of:

1. Recovering or Startup2
   This can happen shortly after a restart of the replicaset, in which case it will usually correct itself after a short time. If the problem persists, delete the malfunctioning pod and allow time for the deploy config to recreate.

   ```
   $ oc delete pod <mongodb-pod-name>
   ```

2. Fatal, Unknown, Down or Rollback
   This indicates a problem with the node in the replicaset. Delete the pod to force the deploy config to recreate resulting in a new pod which should correct the issue.

   ```
   $ oc delete pod <mongodb-pod-name>
   ```

3. Removed
   This is possibly due to a pod connecting to the replicaset before the pod is able to resolve DNS. Restart the replicaset config inside the pod and wait a short time for the member to join the replicaset.

   ```
   $ oc rsh <mongodb-pod-name>
   $ mongo admin -u admin -p ${MONGODB_ADMIN_PASSWORD}
   $ rs.reconfig(rs.config(), {force: true});
   ```

To determine the appropriate value for the **`MONGODB_ADMIN_PASSWORD`** environment variable, view the details for the primary member container in the OpenShift console:

1. Log into the OpenShift Console
2. Navigate to the **`mongodb-1`** pod
3. Click on **`Details`** to view the values of environment variables, including **`MONGODB_ADMIN_PASSWORD`**.

4.3.2.2. Troubleshooting Multiple Failed States

Remove one of the primary nodes and allow the deploy config to restart which should result in the node joining as a secondary node.

```
$ oc delete pod <mongodb-pod-name>
```

4.3.3. Resolving WARNING Issues

4.3.3.1. Troubleshooting a Startup or Arbiter State

Remove the node whose state is Startup or Arbiter and allow the deploy config to restart.

```
$ oc delete pod <mongodb-pod-name>
```

4.3.3.2. Troubleshooting an Even Number of Voters

The MBaaS is configured to have an odd number of MongoDB services. An even number indicates that a MongoDB service has been removed or is is incorrectly connected to the replicaset. Check that each MongoDB-service has a replica count of 1.

```
$ for dc in $(oc get dc | grep mongodb-[0-9] | awk '{print $1}'); do echo
$dc":"$(oc get dc $dc -o yaml | grep replicas); done
```

If any MongoDB services are not set to 1, modify the dc and set it to 1.

```
$ oc edit dc <mongodb-dc-name>
```

```
  spec:
    replicas: 1
```

4.3.3.3. Misconfiguration of Component Resources

The component's resources, that is service, deploy configuration, etc, do not match that of the original template. For example: ports, env vars or image versions are incorrect.

4.3.4. Resolving UNKNOWN Issues

An UNKNOWN status should never occur and usually indicates an internal error in the check. Contact Support and include the status information displayed for the check.

4.4. TROUBLESHOOTING MYSQL ISSUES

4.4.1. Reviewing MySQL Status

The following applies to Core only.

The status levels for MySQL are

Status	Description
OK	Connection was successful and all checked metrics were gather successfully
WARNING	Connection most likely failed because MySQL is not yet ready to fully serve requests
CRITICAL	Connection failed due to major issue, could be out of memory, unable to open socket, etc. The additional text may help point to what is wrong

Status	Description
UNKNOWN	Deployment config for MySQL scaled to 0 or an internal error has occurred and the check was unable to determine a correct status for this check. The error code and message are provided which may help troubleshooting

4.4.2. Resolving CRITICAL and UNKNOWN Issues

The following describes how to check a connection to MySQL and also gather various metric values that are useful for diagnosing issues. Although the check does gather statistics, it does **not** validate the stats against a threshold. If you can connect, authenticate and run the metric collection commands against MySQL - it will return OK.

Sample output

```
  Status Information: Uptime: 1461116 Threads: 8 Questions: 758388 Slow
queries: 0 Opens: 588 Flush tables: 1 Open tables: 100 Queries per second
avg: 0.519
  Performance Data: Connections=175748c;;; Open_files=20;;;
  Open_tables=100;;; Qcache_free_memory=0;;; Qcache_hits=0c;;;
  Qcache_inserts=0c;;; Qcache_lowmem_prunes=0c;;; Qcache_not_cached=0c;;;
  Qcache_queries_in_cache=0;;; Queries=758389c;;; Questions=753634c;;;
  Table_locks_waited=0c;;; Threads_connected=8;;; Threads_running=1;;;
  Uptime=1461116c;;;
```

4.4.2.1. Testing UNKNOWN status

To force an UNKNOWN status, execute.

```
$ oc scale --replicas=0 dc mysql
```

4.4.2.2. Testing CRITICAL status

To force a CRITICAL status after achieving an UNKNOWN status (force recheck of Nagios before it has a chance to fully start up).

```
$ oc scale --replicas=1 dc mysql
```

A status of OK should appear once startup is complete.

4.5. TROUBLESHOOTING HEALTH CHECK ISSUES

4.5.1. Reviewing Health Check Status

The following applies to Core and MBaaS.

The status levels for Health Check are

Status	Description
OK	The component is responding to the health check and reporting that it can successfully communicate with all dependent components
CRITICAL	The component health endpoint has responded but has reported an issue with a dependent component
UNKNOWN	The component health endpoint has not responded

4.5.2. Resolving CRITICAL Issues

The component health endpoint is responding, that is an output can be viewed, but an issue with a dependent component is reported in the output.

4.5.2.1. Troubleshooting a Critical Test Item Error

The reported error is

```
A critical test item encountered an error. Please investigate this. See
the "details" object for specifics.
```

Access the Nagios dashboard and examine the component's state. View the output of the component health endpoint by selecting the Service named *<component>::Health* from the *Services* screen of the dashboard. This output should display which component is experiencing issues.

The fh-supercore health endpoint is reporting an issue connecting to fh-metrics. The full error is

```
Check: fh-supercore::health

A critical test item encountered an error. Please investigate this. See
the "details" object for specifics.
Test: Check fh-metrics - Status: crit - Details: {u'status': u'crit',
u'id': u'fh-fhmetrics', u'error': {u'syscall': u'connect', u'errno':
u'EHOSTUNREACH', u'code': u'EHOSTUNREACH', u'port': 8080, u'address':
u'172.30.112.65'}}
```

Check the Nagios dashboard for issues with the component's Ping check by following the Troubleshooting guide in the Ping Health section.

4.5.3. Resolving UNKNOWN Issues

The component health endpoint has failed to respond.

4.5.3.1. Troubleshooting No Route to Host Error

The reported error is

```
RequestError: The health endpoint at "<component-health-endpoint>" is not
contactable. Error: <urlopen error [Errno 113] No route to host>
```

Check the Nagios dashboard for issues with the component's Ping check by following the Troubleshooting guide in Ping Health section.

4.6. TROUBLESHOOTING PING CHECK ISSUES

4.6.1. Reviewing Ping Check Status

The following applies to Core and MBaaS.

The status levels for Ping Check are

Status	Description
OK	The service is running and a HTTP connection can be established
WARNING	The service is running, but there was some issue with the connection. For example: Authentication failure, Bad Request
CRITICAL	The service is down and a HTTP connection cannot be established
UNKNOWN	An internal error has occurred and the check was unable to determine a correct status for this check

4.6.2. Resolving CRITICAL Issues

The following describes how to check that a HTTP connection can be established to the RHMAP component service.

Check the current status and the status information returned by the check using the Nagios dashboard.

4.6.2.1. Troubleshooting a TCP Socket Error

An attempt was made to connect to address <component> on port 8080 and results in an error.

The full error is

```
No Route To host HTTP CRITICAL - Unable to open TCP socket
```

The service is not running or not contactable. This could be an indication of:

1. No component deployment config

   ```
   $ oc get dc <component>
   ```

   The solution is to recreate the deployment configuration for this component from the installer templates.

   ```
   $ oc get dc <component>
   ```

```
NAME        REVISION   REPLICAS   TRIGGERED BY
<component> 1       1       config
```

2. No replicas for component deployment config (scaled to 0).

```
$ oc get dc <component>
```

```
NAME        REVISION   REPLICAS   TRIGGERED BY
<component> 1       0       config
```

The solution is to scale the component so it has at least 1 replica.

```
oc scale --replicas=1 dc/<component>
oc get dc <component>
```

```
NAME        REVISION   REPLICAS   TRIGGERED BY
<component> 1       1       config
```

3. No component service

```
$ oc get svc <component>
```

The solution is to re-create the service for this component from the installer templates.

```
$ oc get svc <component>
```

```
NAME        CLUSTER-IP    EXTERNAL-IP   PORT(S) AGE
<component> 172.30.204.178   <none>     8080/TCP   1d
```

4.6.2.2. Misconfiguration of Component Resources

The component's resources, that is service, deploy configuration, etc, do not match that of the original template. For example: ports, environment variables or image versions are incorrect.

The solution is to validate each resource against the original template ensuring that all relevant attributes match.

4.6.3. Resolving WARNING Issues

A WARNING status will most commonly happen if the service is running and can be contacted but the client appears to have failed (4xx HTTP status).

4.6.3.1. Troubleshooting HTTP 4xx status

The Client has failed and the result was a 4xx HTTP status. This could be an indication of:

1. Misconfiguration of Component Resources
   The component's resources, that is service, deploy configuration, etc, do not match that of the original template. For example: ports, environment variables or image versions are incorrect.

The solution is to validate each resource against the original template and ensure all relevant attributes match.

2. Performance Issues
   The Ping check might enter a WARNING state if the component under test has performance related issues and the request is taking too long to respond.

   The solution is to ensure that that the CPU and Memory thresholds meet the desired threshold for the component. Check this by following the Troubleshooting CPU/Memory Issues.

4.6.4. Resolving UNKNOWN Issues

An UNKNOWN status should never occur and usually indicates an internal error in the check. Contact Support and include the status information displayed for the check.

CHAPTER 5. TROUBLESHOOTING NFS ISSUES

Ensure that the NFS server and shares are configured according to the OpenShift documentation for configuring NFS PersistentVolumes.

Consider the following when reviewing your NFS configuration:

1. Ensure the NFS mounts are accessible to the OpenShift nodes
   Check this by manually mounting an NFS share to an OpenShift node. If the mount is unsuccessful, ensure that the correct ports are open on the NFS server. More information on the ports required for either NFSv3 or NFSv4 can be found in the NFS section of the OpenShift guide.

2. Ensure SELinux allows Pods to write to a remote NFS server
   There are SELinux settings that must be configured to allow writing. For more information, see the NFS section of the OpenShift guide

3. Check the NFS export options
   The NFS export options in the **`/etc/exports`** file on the NFS server must be set to **`rw,root_squash,no_wdelay`**. If this file is changed, the NFS server may need to be restarted.

4. Ensure the exported directories are writable
   The directories exported from the NFS server which are used by the Pods in OpenShift should be writable. The easiest way to achieve this is to use the **`chmod`** command to set the directory permissions to **`777`**:

   ```
   chmod 777 /var/export/export1
   ```

   The NFS Volume Security section of the NFS OpenShift guide has more information on other options.

CHAPTER 6. TROUBLESHOOTING ANSIBLE INSTALL/UPGRADE ISSUES

6.1. 3-NODE MBAAS RE-RUN STALLING

Once the initial creation of the MongoDB ReplicaSet in the rhmap-installer 3-node-mbaas playbook is complete, the MongoDB-Initiator Pod is deleted. Should the playbook now fail, it is not possible to re-run the playbook in an attempt to complete it - the playbook is not idempotent. The playbook will stall at the point of checking the MongoDB Initiator Pod has successfully completed.

To work-around this:

1. Delete the existing project
2. Re-run the **3-node-mbaas.yml** playbook

6.2. ERROR EXCEPTION DURING PLAYBOOK EXECUTION

In the event of an error exception in the installation or upgrade of RHMAP using the Ansible installer, use the following steps

- Re-run the installer to see if the error persists (i.e that it's not a timing issue).
- Use the playbooks/seed-images.yaml playbook to seed the images.
- Check the **`ansible.log`** file for error information.
- Check the file **`/tmp/events_error.log`** for details (it saves the OpenShift event log).
- View the current event log either via the web console or use the cli command **`oc get events -w`**
- If the problems persist try running the Ansible installer with strict_mode=false **`-e strict_mode=false`**
- Make use of the **`--skip-tags`** this will help isolating the problem and also decrease debugging time.
- Add **`- debug: var="{{ variable_to_debug }}"`** in the template to help with debugging.
- Enable the debug strategy - https://docs.ansible.com/ansible/playbooks_debugger.html

1. If you cannot find the Ansible logs or if Ansible gives you a warning like following, it means Ansible cannot create the file specified in Ansible configuration because of insufficient user permissions.

```
[WARNING]: log file at /some/path/ansible.log is not writeable
and we cannot create it, aborting
```

You can use a custom path for Ansible logs. In **`/opt/rhmap/x.x/rhmap-installer/ansible.cfg`**, you can change the **`log_path`** property to a value of your own choice.

You can also use the **`ANSIBLE_LOG_PATH`** environment variable to achieve the same goal.

```
ANSIBLE_LOG_PATH="/another/path/ansible.log" ansible-playbook
some-playbook.yml
```

Note that **ANSIBLE_LOG_PATH** environment variable has a precedence over the **ansible.cfg log_path** property.