



# Red Hat 3scale API Management 2.10

## Migrating 3scale

Migrate or upgrade 3scale API Management and its components



# Red Hat 3scale API Management 2.10 Migrating 3scale

---

Migrate or upgrade 3scale API Management and its components

## Legal Notice

Copyright © 2021 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, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

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

The OpenStack<sup>®</sup> 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

Migrate 3scale from a template to an operator-based installation. Also, find the information to upgrade 3scale and its components to the latest version.

## Table of Contents

<b>PREFACE</b> .....	<b>3</b>
<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>4</b>
<b>CHAPTER 1. 3SCALE MIGRATION GUIDE: FROM TEMPLATE TO OPERATOR-BASED DEPLOYMENTS</b> ....	<b>5</b>
1.1. PREREQUISITES TO PERFORM THE MIGRATION	5
1.2. MIGRATING 3SCALE TEMPLATE TO OPERATOR-BASED DEPLOYMENTS	5
<b>CHAPTER 2. 3SCALE TEMPLATE-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10</b> .....	<b>6</b>
2.1. PREREQUISITES TO PERFORM THE UPGRADE	6
2.1.1. Configurations	6
2.1.2. Preliminary tasks	6
2.1.3. Tools	6
2.2. UPGRADING FROM 2.9 TO 2.10 IN A TEMPLATE-BASED INSTALLATION	6
2.2.1. Creating a backup of the 3scale project	7
2.2.2. Updating 3scale version number	7
2.2.3. Upgrading 3scale images	8
2.2.3.1. Patch the system image	8
2.2.3.1.1. Patching the system image: 3scale with Oracle Database	8
2.2.3.1.2. Patching the system image: 3scale with other databases	9
2.2.3.2. Patch the apicast image	10
2.2.3.3. Patch the backend image	11
2.2.3.4. Patch the zync image	12
2.2.3.5. Patch the system-memcached image	13
2.2.3.6. Patch the zync-database-postgresql image	13
2.2.3.7. Additional image changes	14
2.2.3.8. Confirm image URLs	16
2.3. UPGRADING 3SCALE WITH AN ORACLE DATABASE IN A TEMPLATE-BASED INSTALLATION	17
2.3.1. Upgrading 3scale with Oracle 19c	17
<b>CHAPTER 3. 3SCALE OPERATOR-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10</b> .....	<b>19</b>
3.1. PREREQUISITES TO PERFORM THE UPGRADE	19
3.2. UPGRADING FROM 2.9 TO 2.10 IN AN OPERATOR-BASED INSTALLATION	19
<b>CHAPTER 4. APICAST OPERATOR-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10</b> .....	<b>21</b>
4.1. PREREQUISITES TO PERFORM THE UPGRADE	21
4.2. UPGRADING APICAST FROM 2.9 TO 2.10 IN AN OPERATOR-BASED INSTALLATION	21



## PREFACE

This guide provides the information to migrate Red Hat 3scale API Management from a template to an operator-based installation, the details required to upgrade your 3scale installation from 2.9 to 2.10, as well as the steps to upgrade APIcast in an operator-based deployment.

To migrate from a template-based to an operator-based deployment, refer to the procedures listed in the [3scale migration guide](#).

To upgrade your 3scale On-premises deployment from 2.9 to 2.10, refer to one of the following guides depending on the installation type:

- [3scale template-based upgrade guide](#)
- [3scale operator-based upgrade guide](#)

To upgrade APIcast in an operator-based deployment, refer to the steps listed in the [APIcast upgrade guide](#).

## MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our [CTO Chris Wright's message](#).



# CHAPTER 1. 3SCALE MIGRATION GUIDE: FROM TEMPLATE TO OPERATOR-BASED DEPLOYMENTS

This section contains information about migrating Red Hat 3scale API Management from a template-based deployment using Red Hat OpenShift 3.11, to an operator-based deployment using Red Hat OpenShift 4.x.



## WARNING

In order to understand the required conditions and procedure, read the entire migration guide before applying the listed steps. The migration process disrupts the provision of the service until the procedure finishes. Due to this disruption, make sure to have a maintenance window.

## 1.1. PREREQUISITES TO PERFORM THE MIGRATION

Before migrating your 3scale installation from a template to an operator-based deployment, confirm that your deployment is supported by consulting the following guides:

- [Backing up 3scale a template-based deployment](#).
- [Restoring the backup in an operator-based deployment](#).

## 1.2. MIGRATING 3SCALE TEMPLATE TO OPERATOR-BASED DEPLOYMENTS

The basic setup before migration is that 3scale points to the OCP3 domain: **3scale.example.com** → **ocp3.example.com**

To migrate 3scale from a template-based deployment using Red Hat OpenShift 3.11 to an operator-based deployment using Red Hat OpenShift 4.1, follow these steps:

1. [Create a 3scale backup from the template-based deployment](#).
2. [Deploy 3scale using the operator](#).
3. [Restore the backup in the operator-based deployment](#).
4. Point the 3scale WILDCARD\_DOMAIN, in this case **3scale.example.com**, to **ocp4.example.com**.

After you have performed all the listed steps, 3scale migration from a template to an operator-based deployment is now complete.

## CHAPTER 2. 3SCALE TEMPLATE-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10

This section contains information about upgrading Red Hat 3scale API Management from version 2.9 to 2.10, in a template-based deployment.



### IMPORTANT

In order to understand the required conditions and procedure, read the entire upgrade guide before applying the listed steps. The upgrade process disrupts the provision of the service until the procedure finishes. Due to this disruption, make sure to have a maintenance window.

## 2.1. PREREQUISITES TO PERFORM THE UPGRADE

This section describes the required configurations, tasks and tools to upgrade 3scale from 2.9 to 2.10 in a template-based installation.

### 2.1.1. Configurations

- 3scale supports upgrade paths from 2.9 to 2.10 with templates on OpenShift 3.11.

### 2.1.2. Preliminary tasks

- Ensure your OpenShift CLI tool is configured in the same project where 3scale is deployed.
- Perform a backup of the database you are using with 3scale. The procedure of the backup is specific to each database type and setup.

### 2.1.3. Tools

You need these tools to perform the upgrade:

- 3scale 2.9 deployed with templates in an OpenShift 3.11 project.
- Bash shell: To run the commands detailed in the upgrade procedure.
- base64: To encode and decode secret information.
- jq: For JSON transformation purposes.

## 2.2. UPGRADING FROM 2.9 TO 2.10 IN A TEMPLATE-BASED INSTALLATION

Follow the procedure described in this section to upgrade 3scale 2.9 to 2.10 in a template-based installation.

To start with the upgrade, go to the project where 3scale is deployed.

```
$ oc project <3scale-project>
```

Then, follow these steps in this order:

1. [Section 2.2.1, "Creating a backup of the 3scale project"](#)
2. [Section 2.2.2, "Updating 3scale version number"](#)
3. [Section 2.2.3, "Upgrading 3scale images"](#)

## 2.2.1. Creating a backup of the 3scale project

### Previous step

None.

### Current step

This step lists the actions necessary to create a backup of the 3scale project.

1. Depending on the database used with 3scale, set `SYSTEM_DB` with one of the following values:
  - If the database is MySQL, **SYSTEM\_DB=system-mysql**.
  - If the database is PostgreSQL, **SYSTEM\_DB=system-postgresql**.
2. Create a back-up file with the existing DeploymentConfigs:

```
THREESCALE_DC_NAMES="apicast-production apicast-staging backend-cron backend-listener backend-redis backend-worker system-app system-memcache ${SYSTEM_DB} system-redis system-sidekiq system-sphinx zync zync-database zync-que"
```

```
for component in ${THREESCALE_DC_NAMES}; do oc get --export -o yaml dc ${component} > ${component}_dc.yml ; done
```

3. Backup all existing OpenShift resources in the project that are exported through the **export all** command:

```
oc get -o yaml --export all > threescale-project-elements.yaml
```

4. Create a back-up file with the additional elements that are not exported with the **export all** command:

```
for object in rolebindings serviceaccounts secrets imagestreamtags cm rolebindingrestrictions limitranges resourcequotas pvc templates cronjobs statefulsets hpa deployments replicaset poddisruptionbudget endpoints; do oc get -o yaml --export $object > $object.yaml; done
```

5. Verify that all of the generated files are not empty, and that all of them have the expected content.

### Next step

[Section 2.2.2, "Updating 3scale version number"](#)

## 2.2.2. Updating 3scale version number

## Previous step

[Section 2.2.1, "Creating a backup of the 3scale project"](#)

## Current step

This step updates the 3scale release version number from 2.9 to 2.10 in the **system-environment** ConfigMap. AMP\_RELEASE is a ConfigMap entry referenced in some DeploymentConfig container environments.

1. To patch AMP\_RELEASE, run this command:

```
oc patch cm system-environment --patch '{"data": {"AMP_RELEASE": "2.10"}}'
```

2. Confirm that the AMP\_RELEASE key in the system-environment ConfigMap has the **2.10** value:

```
oc get cm system-environment -o json | jq '.data["AMP_RELEASE"]'
```

## Next step

[Section 2.2.3, "Upgrading 3scale images"](#)

## 2.2.3. Upgrading 3scale images

### Previous step

[Section 2.2.2, "Updating 3scale version number"](#)

### Current step

This step updates the 3scale images required for the upgrade process.

#### 2.2.3.1. Patch the system image

1. Create the new image stream tag:

```
oc patch imagestream/amp-system --type=json -p '{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "AMP system 2.10"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/system-rhel7:3scale2.10"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}'
```

2. To continue the procedure, consider the database used with your 3scale deployment:
  - If the database is Oracle DB, follow the steps listed in [Section 2.2.3.1.1, "Patching the system image: 3scale with Oracle Database"](#)
  - If the database is different from Oracle DB, follow the steps listed in [Section 2.2.3.1.2, "Patching the system image: 3scale with other databases"](#)

##### 2.2.3.1.1. Patching the system image: 3scale with Oracle Database

1. To start patching the system image of 3scale with an Oracle Database, perform the following procedure: [3scale 2.9 to 2.10 using Oracle 19c](#) .
2. Patch the system-app ImageChangeTrigger:

- a. Remove the old **2.9-oracle** trigger:

```
oc set triggers dc/system-app --from-image=amp-system:2.9-oracle --
containers=system-master,system-developer,system-provider --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-app --from-image=amp-system:2.10-oracle --
containers=system-master,system-developer,system-provider
```

This triggers a redeployment of **system-app**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

3. Patch the **system-sidekiq** ImageChange trigger:

- a. Remove the old **2.9-oracle** trigger:

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.9-oracle --
containers=system-sidekiq,check-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.10-oracle --
containers=system-sidekiq,check-svc
```

This triggers a redeployment of **system-sidekiq**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

4. Patch the **system-sphinx** ImageChange trigger:

- a. Remove the old **2.9-oracle** trigger:

```
oc set triggers dc/system-sphinx --from-image=amp-system:2.9-oracle --
containers=system-sphinx,system-master-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-sphinx --from-image=amp-system:2.10-oracle --
containers=system-sphinx,system-master-svc
```

This triggers a redeployment of **system-sphinx**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

5. Scale 3scale back if you scaled it down.

#### 2.2.3.1.2. Patching the system image: 3scale with other databases

1. Patch the **system-app** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-app --from-image=amp-system:2.9 --containers=system-
master,system-developer,system-provider --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-app --from-image=amp-system:2.10 --containers=system-master,system-developer,system-provider
```

This triggers a redeployment of **system-app**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

2. Patch the **system-sidekiq** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.9 --containers=system-sidekiq,check-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.10 --containers=system-sidekiq,check-svc
```

This triggers a redeployment of **system-sidekiq**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

3. Patch the **system-sphinx** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-sphinx --from-image=amp-system:2.9 --containers=system-sphinx,system-master-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-sphinx --from-image=amp-system:2.10 --containers=system-sphinx,system-master-svc
```

This triggers a redeployment of **system-sphinx**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

### 2.2.3.2. Patch the apicast image

1. Patch the **amp-apicast** image stream:

```
oc patch imagestream/amp-apicast --type=json -p '{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "AMP APICast 2.10"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.10"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}'
```

2. Patch the **apicast-staging** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/apicast-staging --from-image=amp-apicast:2.9 --containers=apicast-staging --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/apicast-staging --from-image=amp-apicast:2.10 --containers=apicast-staging
```

This triggers a redeployment of **apicast-staging**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

3. Patch the **apicast-production** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/apicast-production --from-image=amp-apicast:2.9 --containers=apicast-production,system-master-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/apicast-production --from-image=amp-apicast:2.10 --containers=apicast-production,system-master-svc
```

This triggers a redeployment of **apicast-production**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

### 2.2.3.3. Patch the backend image

1. Patch the **amp-backend** image stream:

```
oc patch imagestream/amp-backend --type=json -p [{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "AMP Backend 2.10"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/backend-rhel7:3scale2.10"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}]
```

2. Patch the **backend-listener** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/backend-listener --from-image=amp-backend:2.9 --containers=backend-listener --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/backend-listener --from-image=amp-backend:2.10 --containers=backend-listener
```

This triggers a redeployment of **backend-listener**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

3. Patch the **backend-worker** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/backend-worker --from-image=amp-backend:2.9 --containers=backend-worker,backend-redis-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/backend-worker --from-image=amp-backend:2.10 --
containers=backend-worker,backend-redis-svc
```

This triggers a redeployment of **backend-worker**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

4. Patch the **backend-cron** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/backend-cron --from-image=amp-backend:2.9 --containers=backend-
cron,backend-redis-svc --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/backend-cron --from-image=amp-backend:2.10 --containers=backend-
cron,backend-redis-svc
```

This triggers a redeployment of **backend-cron**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

#### 2.2.3.4. Patch the zync image

1. Patch the **amp-zync** image stream:

```
oc patch imagestream/amp-zync --type=json -p '{"op": "add", "path": "/spec/tags/-", "value":
{"annotations": {"openshift.io/display-name": "AMP Zync 2.10"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/3scale-amp2/zync-rhel7:3scale2.10"}, "name":
"2.10", "referencePolicy": {"type": "Source"}}}'
```

2. Patch the **zync** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/zync --from-image=amp-zync:2.9 --containers=zync,zync-db-svc --
remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/zync --from-image=amp-zync:2.10 --containers=zync,zync-db-svc
```

This triggers a redeployment of **zync**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

3. Patch the **zync-que** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/zync-que --from-image=amp-zync:2.9 --containers=que --remove
```

- b. Add the new version-specific trigger:



```
oc set triggers dc/zync-que --from-image=amp-zync:2.10 --containers=que
```

This triggers a redeployment of **zync-que**. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

### 2.2.3.5. Patch the **system-memcached** image

1. Patch the **system-memcached** image stream:

```
oc patch imagestream/system-memcached --type=json -p '{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "System 2.10 Memcached"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/memcached-rhel7:3scale2.10"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}'
```

2. Patch the **system-memcache** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-memcache --from-image=system-memcached:2.9 --containers=memcache --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-memcache --from-image=system-memcached:2.10 --containers=memcache
```

This triggers a redeployment of the **system-memcache** DeploymentConfig. Wait until it is redeployed, its corresponding new pods are ready, and the old ones terminated.

### 2.2.3.6. Patch the **zync-database-postgresql** image

1. Patch the **zync-database-postgresql** image stream:

```
oc patch imagestream/zync-database-postgresql --type=json -p '{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "Zync 2.10 PostgreSQL"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/rhscl/postgresql-10-rhel7"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}'
```

- This patch command updates the **zync-database-postgresql** image stream to contain the **2.10** tag. You can verify that the **2.10** tag has been created with these steps:

- a. Run this command:

```
oc get is zync-database-postgresql
```

- b. Check that the *Tags* column shows the **2.10** tag.

2. Patch the **zync-database** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/zync-database --from-image=zync-database-postgresql:2.9 --containers=postgresql --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/zync-database --from-image=zync-database-postgresql:2.10 --
containers=postgresql
```

In case there are new updates on the image, this patch might also trigger a redeployment of the **zync-database** DeploymentConfig. If this happens, wait until the new pods are redeployed and ready, and the old pods are terminated.

### 2.2.3.7. Additional image changes

If one or more of the following DeploymentConfigs are available in your 3scale 2.10 installation, click the links that apply to obtain more information on how to proceed:

- [backend-redis DeploymentConfig](#)
- [system-redis DeploymentConfig](#)
- [system-mysql DeploymentConfig](#)
- [system-postgresql DeploymentConfig](#)

#### backend-redis DeploymentConfig

If the **backend-redis** DeploymentConfig exists in your current 3scale installation, patch the **redis** image for **backend-redis**:

1. Patch the **backend-redis** image stream:

```
oc patch imagestream/backend-redis --type=json -p [{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "Backend 2.10 Redis"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/rhsc/redis-32-rhel7:3.2"}, "name": "2.10",
"referencePolicy": {"type": "Source"}}}]
```

This patch updates the backend-redis image stream to contain the **2.10** tag. With the command below, you can confirm that the tag has been created if the *Tags* column shows **2.10**:

```
oc get is backend-redis
```

2. Patch the **backend-redis** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/backend-redis --from-image=backend-redis:2.9 --containers=backend-
redis --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/backend-redis --from-image=backend-redis:2.10 --
containers=backend-redis
```

In case there are new updates on the image, this patch might also trigger a redeployment of the **backend-redis** DeploymentConfig. If this happens, wait until the new pods are redeployed and ready, and the old pods are terminated.

## system-redis DeploymentConfig

If the **system-redis** DeploymentConfig exists in your current 3scale installation, patch the **redis** image for **system-redis**.

1. Patch the **system-redis** image stream:

```
oc patch imagestream/system-redis --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "System 2.10 Redis"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/rhsc/redis-32-rhel7:3.2"}, "name": "2.10",
"referencePolicy": {"type": "Source"}}}]'
```

This patch updates the **system-redis** image stream to contain the **2.10** tag. With the command below, you can confirm that the tag has been created if the *Tags* column shows **2.10**:

```
oc get is system-redis
```

2. Patch the **system-redis** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-redis --from-image=system-redis:2.9 --containers=system-
redis --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-redis --from-image=system-redis:2.10 --containers=system-
redis
```

In case there are new updates on the image, this patch might also trigger a redeployment of the **system-redis** DeploymentConfig. If this happens, wait until the new pods are redeployed and ready, and the old pods are terminated.

## system-mysql DeploymentConfig

If the **system-mysql** DeploymentConfig exists in your current 3scale installation, patch the MySQL image for **system-mysql**.

1. Patch the **system-mysql** image stream:

```
oc patch imagestream/system-mysql --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "System 2.10 MySQL"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/rhsc/mysql-57-rhel7:5.7"}, "name": "2.10",
"referencePolicy": {"type": "Source"}}}]'
```

This patch updates the **system-mysql** image stream to contain the **2.10** tag. With the command below, you can confirm that the tag has been created if the *Tags* column shows **2.10**:

```
oc get is system-mysql
```

2. Patch the **system-mysql** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-mysql --from-image=system-mysql:2.9 --containers=system-mysql --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-mysql --from-image=system-mysql:2.10 --containers=system-mysql
```

In case there are new updates on the image, this patch might also trigger a redeployment of the **system-mysql** DeploymentConfig. If this happens, wait until the new pods are redeployed and ready, and the old pods are terminated.

### system-postgresql DeploymentConfig

If the **system-postgresql** DeploymentConfig exists in your current 3scale installation, patch the PostgreSQL image for **system-postgresql**.

1. Patch the **system-postgresql** image stream:

```
oc patch imagestream/system-postgresql --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "System 2.10 PostgreSQL"}, "from": {"kind": "DockerImage", "name": "registry.redhat.io/rhsc/postgresql-10-rhel7"}, "name": "2.10", "referencePolicy": {"type": "Source"}}}]'
```

This patch updates the **system-postgresql** image stream to contain the **2.10** tag. With the command below, you can confirm that the tag has been created if the *Tags* column shows **2.10**:

```
oc get is system-postgresql
```

2. Patch the **system-postgresql** ImageChange trigger:

- a. Remove the old **2.9** trigger:

```
oc set triggers dc/system-postgresql --from-image=system-postgresql:2.9 --containers=system-postgresql --remove
```

- b. Add the new version-specific trigger:

```
oc set triggers dc/system-postgresql --from-image=system-postgresql:2.10 --containers=system-postgresql
```

In case there are new updates on the image, this patch might also trigger a redeployment of the **system-postgresql** DeploymentConfig. If this happens, wait until the new pods are redeployed and ready, and the old pods are terminated.

#### 2.2.3.8. Confirm image URLs

Confirm that all the image URLs of the DeploymentConfigs contain the new image registry URLs with a hash added at the end of each URL address:

```
THREESCALE_DC_NAMES="apicast-production apicast-staging backend-cron backend-listener backend-redis backend-worker system-app system-memcache system-mysql system-redis system-sidekiq system-sphinx zync zync-database zync-que"
```

```
for component in ${THREESCALE_DC_NAMES}; do echo -n "${component} image: " && oc get dc
$component -o json | jq .spec.template.spec.containers[0].image ; done
```

### Next step

None. After you have performed all the listed steps, 3scale upgrade from 2.9 to 2.10 in a template-based deployment is now complete.

## 2.3. UPGRADING 3SCALE WITH AN ORACLE DATABASE IN A TEMPLATE-BASED INSTALLATION

This section explains how to upgrade Red Hat 3scale API Management when you are using a 3scale system image with an Oracle Database, in a template-based installation with OpenShift 3.11.

### Prerequisites

A 3scale installation with the Oracle Database. See [Setting up your 3scale system image with an Oracle Database](#).

To upgrade your 3scale system image with an Oracle Database in a template-based installation, perform the procedure below:

- [Section 2.3.1, “Upgrading 3scale with Oracle 19c”](#)

### 2.3.1. Upgrading 3scale with Oracle 19c

This procedure guides you through an Oracle Database 19c update for 3scale 2.10 from an existing 3scale 2.9 installation.

**IMPORTANT:** Loss of connection to the database can potentially corrupt 3scale. Make a backup before proceeding to perform the upgrade. See the official Oracle Database documentation: [Oracle Database Backup and Recovery User’s Guide](#).

### Prerequisites

- A 3scale 2.9 installation.
- An Oracle Database 19c installation.
  - For more details about configuring 3scale with Oracle, see [Preparing the Oracle Database](#)

### Procedure

1. Clone the [OpenShift Templates for 3scale 2.10](#)

```
git clone --branch 3scale-2.10.0-GA https://github.com/3scale/3scale-amp-openshift-
templates.git
```

2. Place your Oracle Database Instant Client Package files into the **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** directory.
3. Run the **oc process** command with the **-f** option and specify the **build.yml** OpenShift template:

```
oc process -f build.yml | oc apply -f -
```

4. Enter the **oc start-build** command to build the new system image:

```
oc start-build 3scale-amp-system-oracle --from-dir=.
```

5. Wait until the build completes. To see the state of the build, run the following command:

```
$ oc get build <build-name> -o jsonpath="{.status.phase}"
```

- a. Wait until the build is in a Complete state.

### Additional resources

For more information about 3scale and Oracle Database support, see [Red Hat 3scale API Management Supported Configurations](#).

## CHAPTER 3. 3SCALE OPERATOR-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10

This section contains information about upgrading Red Hat 3scale API Management from version 2.9 to 2.10, in an operator-based deployment.

To automatically obtain a micro-release of 3scale, make sure automatic updates is on. To check this, see [Setting up the 3scale operator for micro releases](#).



### IMPORTANT

In order to understand the required conditions and procedure, read the entire upgrade guide before applying the listed steps. The upgrade process disrupts the provision of the service until the procedure finishes. Due to this disruption, make sure to have a maintenance window.

### 3.1. PREREQUISITES TO PERFORM THE UPGRADE

This section describes the required configurations to upgrade 3scale from 2.9 to 2.10 in an operator-based installation.

- 3scale 2.9 previously deployed via the 3scale operator.
- An OpenShift Container Platform (OCP) 4.x cluster with administrator access.

### 3.2. UPGRADING FROM 2.9 TO 2.10 IN AN OPERATOR-BASED INSTALLATION

To upgrade 3scale from version 2.9 to 2.10 in an operator-based deployment:

1. Log in to the OCP console using the account with administrator privileges.
2. Select the project where the *3scale-operator* has been deployed.
3. Click **Operators > Installed Operators**
4. Select **Red Hat Integration - 3scale > Subscription > Channel**
5. Edit the channel of the subscription by selecting *threescale-2.10* and save the changes.
  - This will start the upgrade process.
  - Wait until the upgrade process finishes for *APIManager*.
6. Query the pods status on the project:

```
oc get pods
```

- Wait until all the new versions are running and ready without errors.
- They might have temporary errors during the upgrade process.

**NOTE**

Times can vary from 5-10 minutes approximately. Be sure to keep checking the state of the pods until all of them are running, ready, and without errors.

7. Confirm the upgrade process has been successful, by logging in to the 3scale Admin Portal and check that it works as expected.
8. Check the status of the *APIManager* objects and get the *YAML* content by running the following command:

```
oc get apimanager <myapimanager> -o yaml
```

- The new annotations with the values should be as follows:

```
apps.3scale.net/apimanager-threescale-version: "2.10"  
apps.3scale.net/threescale-operator-version: "0.7.0"
```

After you have performed all the listed steps, 3scale upgrade from 2.9 to 2.10 in an operator-based deployment is now complete.



## CHAPTER 4. APICAST OPERATOR-BASED UPGRADE GUIDE: FROM 2.9 TO 2.10

This section contains information about upgrading APIcast from version 2.9 to 2.10, in an operator-based deployment.



### IMPORTANT

In order to understand the required conditions and procedure, read the entire upgrade guide before applying the listed steps. The upgrade process disrupts the provision of the service until the procedure finishes. Due to this disruption, make sure to have a maintenance window.

### 4.1. PREREQUISITES TO PERFORM THE UPGRADE

This section describes the required configurations to upgrade APIcast from 2.9 to 2.10 in an operator-based installation.

- APIcast 2.9 previously deployed via the APIcast operator.
- An OpenShift Container Platform (OCP) 4.x cluster with administrator access.

### 4.2. UPGRADING APICAST FROM 2.9 TO 2.10 IN AN OPERATOR-BASED INSTALLATION

To upgrade APIcast from version 2.9 to 2.10 in an operator-based deployment:

1. Log in to the OCP console using the account with administrator privileges.
2. Select the project where the *APIcast operator* has been deployed.
3. Click **Operators > Installed Operators**
4. In **Subscription > Channel** select *Red Hat Integration - 3scale APIcast gateway* .
5. Edit the channel of the subscription by selecting the *threescale-2.10* channel and save the changes.
  - This will start the upgrade process.
  - Wait until the upgrade process finishes for *APIcast*.
6. Query the pods status on the project:

```
oc get pods
```

- Wait until all the new versions are running and ready without errors.
- They might have temporary errors during the upgrade process.



### NOTE

Times can vary from 5-10 minutes approximately. Be sure to keep checking the state of the pods until all of them are running, ready, and without errors.

7. Check the status of the *APIcast* objects and get the *YAML* content by running the following command:

```
oc get apicast <myapicast> -o yaml
```

- The new annotations with the values should be as follows:

```
apicast.apps.3scale.net/operator-version: "0.4.0"
```

After you have performed all the listed steps, APIcast upgrade from 2.9 to 2.10 in an operator-based deployment is now complete.