Red Hat 3scale API Management 2.6

Installing 3scale

Install and configure 3scale API Management.

Last Updated: 2022-12-13

Red Hat 3scale API Management 2.6 Installing 3scale

Install and configure 3scale API Management.

Legal Notice

Abstract

This guide provides the information to install and configure 3scale API Management.

Table of Contents

PREFACE 4

CHAPTER 1. 3SCALE SYSTEM IMAGE WITH ORACLE DATABASE 5
1.1. BEFORE YOU BEGIN 5
1.1.1. Obtaining Oracle software components 5
1.1.2. Prerequisites 5
1.2. PREPARING ORACLE DATABASE 5
1.3. BUILDING THE SYSTEM IMAGE 6

CHAPTER 2. INSTALLATION GUIDE FOR 3SCALE ON OPENSHIFT 8
2.1. INTRODUCTION 8
2.1.1. Prerequisites 8
2.2. SYSTEM REQUIREMENTS 8
2.2.1. Environment requirements 8
2.2.2. Hardware requirements 8
2.3. CONFIGURING NODES AND ENTITLEMENTS 9
2.4. DEPLOYING 3SCALE ON OPENSHIFT USING A TEMPLATE 9
2.4.1. Prerequisites 9
2.4.2. Configuring registry authentication in OpenShift 10
2.4.3. Creating registry service accounts 10
2.4.4. Modifying registry service accounts 11
2.4.5. Importing the 3scale template 12
2.4.6. Getting the Admin Portal URL 13
2.4.7. Configuring SMTP Variables (Optional) 13
2.5. 3SCALE TEMPLATE PARAMETERS 15
2.6. USING APICAST WITH 3SCALE ON OPENSHIFT 18
2.6.1. Deploying APIcast templates on an existing OpenShift cluster containing 3scale 18
2.6.2. Connecting APIcast from a different OpenShift cluster 19
2.6.3. Changing the default behavior for embedded APIcast 19
2.6.4. Connecting multiple APIcast deployments on a single OpenShift cluster over internal service routes 20
2.6.5. Connecting APIcast on other deployments 20
2.7. DEPLOYING 3SCALE USING THE OPERATOR 21
2.7.1. Prerequisites 21
2.7.2. Deploying the APIManager custom resource 21
2.7.3. Getting the APIManager administrator credentials 22
2.7.4. Getting the Admin Portal URL 22
2.8. TROUBLESHOOTING 22
2.8.1. Previous deployment leaving dirty persistent volume claims 23
2.8.2. Incorrectly pulling from the Docker registry 23
2.8.3. Permission issues for MySQL 24
2.8.4. Unable to upload logo or images 24
2.8.5. Test calls do not work on OpenShift 24
2.8.6. APIcast on a different project from 3scale 25

CHAPTER 3. INSTALLING APICAST 26
3.1. PREREQUISITES 26
3.2. DEPLOYMENT OPTIONS 26
3.3. ENVIRONMENTS 26
3.4. CONFIGURING THE INTEGRATION SETTINGS 27
3.5. CONFIGURING YOUR SERVICE 27
3.5.1. Declare the API backend 27
3.5.2. Configure the authentication settings 28

3.5.3. Configure the API test call 28
3.5.4. Save the configuration settings 28

CHAPTER 4. RUNNING APICAST ON RED HAT OPENSHIFT 30
4.1. PREREQUISITES 30
4.2. SETTING UP RED HAT OPENSHIFT 30
4.2.1. Install the Docker containerized environment 30
4.2.2. Start the OpenShift cluster 31
4.2.3. Set up the OpenShift cluster on a remote server (Optional) 31
4.3. DEPLOYING APICAST USING THE OPENSHIFT TEMPLATE 32
4.4. CREATING ROUTES VIA THE OPENSHIFT CONSOLE 32

CHAPTER 5. APICAST ON THE DOCKER CONTAINERIZED ENVIRONMENT 36
5.1. PREREQUISITES 36
5.2. INSTALLING THE DOCKER CONTAINERIZED ENVIRONMENT 36
5.3. RUNNING THE DOCKER CONTAINERIZED ENVIRONMENT GATEWAY 36
5.3.1. The Docker command options 37
5.4. TESTING APICAST 37

CHAPTER 6. INSTALLING THE 3SCALE OPERATOR ON OPENSHIFT 4.X 38
6.1. CREATING A NEW OPENSHIFT PROJECT 38
6.2. INSTALLING AND CONFIGURING THE 3SCALE OPERATOR USING THE OLM 38

CHAPTER 7. 3SCALE HIGH AVAILABILITY AND EVALUATION TEMPLATES 40
7.1. INTRODUCTION 40
7.2. PREREQUISITES 40
7.3. HIGH AVAILABILITY TEMPLATE 40
7.3.1. Prerequisites 40
7.3.2. Using the HA template 40
7.4. EVALUATION TEMPLATE 41

CHAPTER 8. REDIS HIGH AVAILABILITY (HA) SUPPORT FOR 3SCALE 42
8.1. SETTING UP REDIS FOR ZERO DOWNTIME 42
8.2. CONFIGURING BACK-END COMPONENTS FOR 3SCALE 42
8.2.1. Creating backend-redis and system-redis secrets 43
8.2.1.1. Deploying a fresh installation of 3scale for HA 43
8.2.1.2. Migrating a non-HA deployment of 3scale to HA 44
8.2.2. Using Redis Enterprise 45
8.2.3. Using Redis Sentinel 45

CHAPTER 9. CONFIGURING AN EXTERNAL MYSQL DATABASE 47
9.1. EXTERNAL MYSQL DATABASE LIMITATIONS 47
9.2. EXTERNALIZING THE MYSQL DATABASE 47
9.3. ROLLING BACK 51

PREFACE

This guide will help you to install and configure 3scale

CHAPTER 1. 3SCALE SYSTEM IMAGE WITH ORACLE DATABASE

By default, Red Hat 3scale API Management 2.6 has a component called **system** that stores configuration data in a MySQL database. You have the option to override the default database and store your information in an external Oracle Database. Follow the steps in this document to build a custom system container image with your own Oracle Database client binaries and deploy 3scale to OpenShift.

1.1. BEFORE YOU BEGIN

1.1.1. Obtaining Oracle software components

Before you can build the custom 3scale system container image, you must acquire a supported version of the following Oracle software components:

- Oracle Instant Client Package Basic or Basic Light
- Oracle Instant Client Package SDK
- Oracle Instant Client Package ODBC

1.1.2. Prerequisites

- Supported version of Oracle Database accessible from your OpenShift cluster
- Access to the Oracle Database **system** user for installation procedures
- The 3scale 2.6 **amp.yml** template

1.2. PREPARING ORACLE DATABASE

1. Create a new database.
   The following settings are required for the Oracle Database to work with 3scale:

   ```
   ALTER SYSTEM SET max_string_size=extended SCOPE=SPFILE;
   ALTER SYSTEM SET compatible='12.2.0.1' SCOPE=SPFILE;
   ```

2. Collect the database details.
   Get the following information that will be needed for 3scale configuration:

   - Oracle Database URL
   - Oracle Database service name
   - Oracle Database **system** password
     The **DATABASE_URL** parameter must follow this format:

```
oracle-enhanced://USER:PASSWORD@HOST:PORT/DATABASE
```

Example

```
DATABASE_URL="oracle-enhanced://USER:PASSWORD@my-oracle-
database.com:1521/threescalepdb"
```

Additional resources

For information on creating a new database in Oracle Database, see the Oracle documentation.

1.3. BUILDING THE SYSTEM IMAGE

NOTE

After you download the system Oracle *build.yml* file, you will have to manually change the **AMP_RELEASE** value from **2.5.0** to **2.6.0**.

This section provides steps to build the system image.

Prerequisites

- The Oracle Database must be configured. For more details, follow the steps in Section 1.2, “Preparing Oracle Database”.

Procedure

1. Clone the 3scale OpenShift Templates GitHub repository. Use the following command:

   ```
   $ git clone --branch 2.6.0.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. Download the 3scale 2.6 *amp.yml* template.

4. Run the **oc new-app** command with the **-f** option and specify the **build.yml** OpenShift template:

   ```
   $ oc new-app -f build.yml
   ```

5. Run the **oc new-app** command with the **-f** option to indicate the **amp.yml** OpenShift template, and the **-p** option to specify the **WILDCARD_DOMAIN** parameter with the domain of your OpenShift cluster:

   ```
   $ oc new-app -f amp.yml -p WILDCARD_DOMAIN=mydomain.com
   ```

6. Enter the following **oc patch** commands, replacing **SYSTEM_PASSWORD** with the Oracle Database **system** password you set up in Section 1.2, “Preparing Oracle Database”:

   ```
   $ oc patch dc/system-app -p '[{"op": "add", "path":
   "/spec/strategy/rollingParams/pre/execNewPod/env/-", "value": {"name":
   "ORACLE_SYSTEM_PASSWORD", "value": "SYSTEM_PASSWORD"}}]' --type=json

   $ oc patch dc/system-app -p '{"spec": {"strategy": {"rollingParams": {"post":{"execNewPod":
   {"env": [{"name": "ORACLE_SYSTEM_PASSWORD", "value":
   "SYSTEM_PASSWORD"}]}}}}}}'
   ```

7. Enter the following command, replacing **DATABASE_URL** to point to your Oracle Database, specified in Section 1.2, "Preparing Oracle Database":

```
$ oc patch secret/system-database -p '{"stringData": {"URL": "DATABASE_URL"}}'
```

8. Link the pull secret to the builder with the following command:

```
$ oc secrets link builder threescale-registry-auth
```

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

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

Additional resources

- For more information about 3scale and Oracle Database support, see Red Hat 3scale API Management Supported Configurations.

CHAPTER 2. INSTALLATION GUIDE FOR 3SCALE ON OPENSHIFT

2.1. INTRODUCTION

This guide walks you through steps to deploy Red Hat 3scale API Management - On-premises 2.6 on OpenShift.

The 3scale solution for on-premises deployment is composed of:

- Two API gateways: embedded APIcast
- One 3scale Admin Portal and Developer Portal with persistent storage

There are two ways to deploy a 3scale solution:

- Section 2.4, “Deploying 3scale on OpenShift using a template”
- Section 2.7, “Deploying 3scale using the operator”

NOTE

Whether deploying 3scale using the operator or templates, you must first Section 2.4.2, “Configuring registry authentication in OpenShift”.

2.1.1. Prerequisites

- You must configure 3scale servers for UTC (Coordinated Universal Time).

2.2. SYSTEM REQUIREMENTS

This section lists the requirements for the 3scale - OpenShift template.

2.2.1. Environment requirements

3scale requires an environment specified in supported configurations.

Persistent volumes:

- 3 RWO (ReadWriteOnce) persistent volumes for Redis and MySQL persistence
- 1 RWX (ReadWriteMany) persistent volume for CMS and System-app Assets

The RWX persistent volume must be configured to be group writable. For a list of persistent volume types that support the required access modes, see the OpenShift documentation .

2.2.2. Hardware requirements

Hardware requirements depend on your usage needs. Red Hat recommends that you test and configure your environment to meet your specific requirements. Following are the recommendations when configuring your environment for 3scale on OpenShift:

- Compute optimized nodes for deployments on cloud environments (AWS c4.2xlarge or Azure Standard_F8).
- Very large installations may require a separate node (AWS M4 series or Azure Av2 series) for Redis if memory requirements exceed your current node's available RAM.
- Separate nodes between routing and compute tasks.
- Dedicated computing nodes for 3scale specific tasks.
- Set the **PUMA_WORKERS** variable of the backend listener to the number of cores in your compute node.

2.3. CONFIGURING NODES AND ENTITLEMENTS

Before you can deploy 3scale on OpenShift, you must configure your nodes and the entitlements required for your environment to fetch images from the Red Hat Container Registry .

Perform the following steps to configure the entitlements:

1. Install Red Hat Enterprise Linux (RHEL) on each of your nodes.
2. Register your nodes with Red Hat using the Red Hat Subscription Manager (RHSM), via the interface or the command line.
3. Attach your nodes to your 3scale subscription using RHSM.
4. Install OpenShift on your nodes, complying with the following requirements:
   - Use a supported OpenShift version.
   - Configure persistent storage on a file system that supports multiple writes.
5. Install the OpenShift command line interface.
6. Enable access to the **rhel-7-server-3scale-amp-2.6-rpms** repository using the subscription manager:

   ```
   sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.6-rpms
   ```

7. Install the 3scale template called **3scale-amp-template**. This will be saved at **/opt/amp/templates**.

   ```
   sudo yum install 3scale-amp-template
   ```

2.4. DEPLOYING 3SCALE ON OPENSHIFT USING A TEMPLATE

This section describes how to deploy 3scale on OpenShift using a template.

2.4.1. Prerequisites

- An OpenShift cluster configured as specified in the Section 2.3, "Configuring nodes and entitlements" section.
- A domain that resolves to your OpenShift cluster.

  - **Note:** OpenShift Container Platform (OCP) 3.11 supports deployment of 3scale using templates only.

- Access to the Red Hat container catalog.
- (Optional) A working SMTP server for email functionality.

Follow these procedures to install 3scale on OpenShift using a **.yml** template:

- Section 2.4.2, “Configuring registry authentication in OpenShift”
- Section 2.4.3, “Creating registry service accounts”
- Section 2.4.5, “Importing the 3scale template”
- Section 2.4.7, “Configuring SMTP Variables (Optional)”

2.4.2. Configuring registry authentication in OpenShift

You must configure registry authentication to the Red Hat container registry before you can use Red Hat 3scale API Management OpenShift image stream. Follow the instruction below to configure the registration to container registry.

1. Log in to the OpenShift server as an administrator, as follows:

```
oc login -u system:admin
```

2. Log in to the OpenShift project where you will be installing the image streams. Red Hat recommends that you use the **openshift** project for the 3scale OpenShift image streams.
Note: It will have a prefix that is a fixed, random string.

```
oc project your-openshift-project
```

3. Create a **docker-registry** secret using the credentials you created in Section 2.4.3, “Creating registry service accounts”.

NOTE

- Replace **your-registry-service-account-username** with the username created in the format, **12345678|username**.
- Replace **your-registry-service-account-password** with the password string below the username, under the *Token Information* tab.
- Create a **docker-registry** secret for every new **namespace** where the image streams reside and which use *registry.redhat.io*.

```
$ oc create secret docker-registry threescale-registry-auth \
  --docker-server=registry.redhat.io \
  --docker-username="your-registry-service-account-username" \
  --docker-password="your-registry-service-account-password"
```

2.4.3. Creating registry service accounts

To use container images from **registry.redhat.io** in a shared environment with 3scale 2.6 deployed on OpenShift, you must use a *Registry Service Account* instead of an individual user's *Customer Portal* credentials.

NOTE

It is a requirement for 3scale 2.6 that you follow the steps outlined below before deploying either on OpenShift using a template or via the operator, as both options use registry authentication.

1. Navigate to the Registry Service Accounts page and log in.
2. Click **New Service Account**
3. Fill in the form on the *Create a New Registry Service Account* page.
    a. Add a name for the *service account*.
    Note: You will see a fixed-length, randomly generated number string before the form field.
4. Enter a *Description*.
5. Click **Create**.
6. Navigate back to your *Service Accounts*.
7. Click the *Service Account* you created.
8. Make a note of the username, including the prefix string, for example **12345678|username**, and your password.
    a. This username and password will be used to log in to **registry.redhat.io**.

NOTE

There are tabs available on the *Token Information* page that show you how to use the authentication token. For example, the *Token Information* tab shows the username in the format **12345678|username** and the password string below it.

2.4.4. Modifying registry service accounts

Service accounts can be modified or deleted. This can done from the *Registry Service Account* page using the pop-up menu to the right of each authentication token in the table.

WARNING

The regeneration or removal of *service accounts* will impact systems that are using the token to authenticate and retrieve content from **registry.redhat.io**.

A description for each function is as follows:

- **Regenerate token:** Allows an authorized user to reset the password associated with the *Service Account*.
  Note: The username for the *Service Account* cannot be changed.
- **Update Description:** Allows an authorized user to update the description for the *Service Account*.
- **Delete Account:** Allows an authorized user to remove the *Service Account*.

For more information see:

- Red Hat Container Registry Authentication
- Authentication enabled Red Hat registry

2.4.5. Importing the 3scale template

NOTE

- Wildcard routes have been removed in 3scale 2.6.
  - This functionality is handled by Zync in the background.
- When API providers are created, updated, or deleted, routes automatically reflect those changes.

Perform the following steps to import the 3scale template into your OpenShift cluster:

1. From a terminal session log in to OpenShift as the cluster administrator:

```
oc login
```

2. Select your project, or create a new project:

```
oc project <project_name>
```

```
oc new-project <project_name>
```

3. Enter the **oc new-app** command:

   a. Specify the **--file** option with the path to the *amp.yml* file you downloaded as part of Section 2.3, “Configuring nodes and entitlements”.

   b. Specify the **--param** option with the **WILDCARD_DOMAIN** parameter set to the domain of your OpenShift cluster:

   ```
   oc new-app --file /opt/amp/templates/amp.yml --param WILDCARD_DOMAIN=
   <WILDCARD_DOMAIN>
   ```

   The terminal shows the master and tenant URLs and credentials for your newly created 3scale Admin Portal. This output should include the following information:

   - master admin username

- master password
- master token information
- tenant username
- tenant password
- tenant token information

4. Log in to https://user-admin.3scale-project.example.com as admin/xXxXyz123.

```
* With parameters:

 * ADMIN_PASSWORD=xXxXyz123 # generated
 * ADMIN_USERNAME=admin
 * TENANT_NAME=user

 * MASTER_NAME=master
 * MASTER_USER=master
 * MASTER_PASSWORD=xXxXyz123 # generated

--> Success
Access your application via route 'user-admin.3scale-project.example.com'
Access your application via route 'master-admin.3scale-project.example.com'
Access your application via route 'backend-user.3scale-project.example.com'
Access your application via route 'user.3scale-project.example.com'
Access your application via route 'api-user-apicast-staging.3scale-project.example.com'
Access your application via route 'api-user-apicast-production.3scale-project.example.com'
```

5. Make a note of these details for future reference.

NOTE

Wait for 3scale to fully deploy on OpenShift for your login and credentials to work.

2.4.6. Getting the Admin Portal URL

When you deploy 3scale using the template, a default tenant is created, with a fixed URL: **3scale-admin.${wildcardDomain}**

The dashboard shows the new portal URL of the tenant. As an example, if the *<wildCardDomain>* is **3scale-project.example.com**, the Admin Portal URL is: **https://3scale-admin.3scale-project.example.com**.

The **wildcardDomain** is the *<wildCardDomain>* parameter you provided during installation. Open this unique URL in a browser using the this command:

```
xdg-open https://3scale-admin.3scale-project.example.com
```

Optionally, you can create new tenants on the *MASTER portal URL*: **master.${wildcardDomain}**

2.4.7. Configuring SMTP Variables (Optional)

OpenShift uses email to send notifications and invite new users. If you intend to use these features, you must provide your own SMTP server and configure SMTP variables in the SMTP config map.

Perform the following steps to configure the SMTP variables in the SMTP config map:

1. If you are not already logged in, log in to OpenShift:

```
oc login
```

   a. Configure variables for the SMTP config map. Use the **oc patch** command, specify the **configmap** and **smtp** objects, followed by the **-p** option and write the following new values in JSON for the following variables:

Variable	Description
address	Allows you to specify a remote mail server as a relay
username	Specify your mail server username
password	Specify your mail server password
domain	Specify a HELO domain
port	Specify the port on which the mail server is listening for new connections
authentication	Specify the authentication type of your mail server. Allowed values: **plain** (sends the password in the clear), **login** (send password Base64 encoded), or **cram_md5** (exchange information and a cryptographic Message Digest 5 algorithm to hash important information)
openssl.verify.mode	Specify how OpenSSL checks certificates when using TLS. Allowed values: **none**, **peer**, **client_once**, or **fail_if_no_peer_cert**.

Example

```
oc patch configmap smtp -p '{"data":{"address":"<your_address>"}}'
oc patch configmap smtp -p '{"data":{"username":"<your_username>"}}'
oc patch configmap smtp -p '{"data":{"password":"<your_password>"}}'
```

2. After you have set the configmap variables, redeploy the **system-app** and **system-sidekiq** pods:

```
oc rollout latest dc/system-app
oc rollout latest dc/system-sidekiq
```

2.5. 3SCALE TEMPLATE PARAMETERS

Template parameters configure environment variables of the 3scale (*amp.yml*) template during and after deployment.

Name	Description	Default Value	Required?
APP_LABEL	Used for object app labels	**3scale-api-management**	yes
ZYNC_DATABASE_PASSWORD	Password for the PostgreSQL connection user. Generated randomly if not provided.	N/A	yes
ZYNC_SECRET_KEY_BASE	Secret key base for Zync. Generated randomly if not provided.	N/A	yes
ZYNC_AUTHENTICATION_TOKEN	Authentication token for Zync. Generated randomly if not provided.	N/A	yes
AMP_RELEASE	3scale release tag.	**2.6.0**	yes
ADMIN_PASSWORD	A randomly generated 3scale administrator account password.	N/A	yes
ADMIN_USERNAME	3scale administrator account username.	**admin**	yes
APICAST_ACCESS_TOKEN	Read Only Access Token that APIcast will use to download its configuration.	N/A	yes
ADMIN_ACCESS_TOKEN	Admin Access Token with all scopes and write permissions for API access.	N/A	no
WILDCARD_DOMAIN	Root domain for the wildcard routes. For example, a root domain **example.com** will generate **3scale-admin.example.com**.	N/A	yes

Name	Description	Default Value	Required?
TENANT_NAME	Tenant name under the root that Admin Portal will be available with - admin suffix.	**3scale**	yes
MYSQL_USER	Username for MySQL user that will be used for accessing the database.	**mysql**	yes
MYSQL_PASSWORD	Password for the MySQL user.	N/A	yes
MYSQL_DATABASE	Name of the MySQL database accessed.	**system**	yes
MYSQL_ROOT_PASSWORD	Password for Root user.	N/A	yes
SYSTEM_BACKEND_USERNAME	Internal 3scale API username for internal 3scale api auth.	**3scale_api_user**	yes
SYSTEM_BACKEND_PASSWORD	Internal 3scale API password for internal 3scale api auth.	N/A	yes
REDIS_IMAGE	Redis image to use	**registry.redhat.io/rhscl/redis-32-rhel7:3.2**	yes
MYSQL_IMAGE	Mysql image to use	**registry.redhat.io/rhscl/mysql-57-rhel7:5.7**	yes
MEMCACHED_IMAGE	Memcached image to use	**registry.redhat.io/3scale-amp20/memcached:1.4.15**	yes
POSTGRESQL_IMAGE	Postgresql image to use	**registry.redhat.io/rhscl/postgresql-10-rhel7**	yes
AMP_SYSTEM_IMAGE	3scale System image to use	**registry.redhat.io/3scale-amp26/system**	yes
AMP_BACKEND_IMAGE	3scale Backend image to use	**registry.redhat.io/3scale-amp26/backend**	yes

Name	Description	Default Value	Required?
AMP_APICAST_IMAGE	3scale APIcast image to use	**registry.redhat.io/3scale-amp26/apicast-gateway**	yes
AMP_ZYNC_IMAGE	3scale Zync image to use	**registry.redhat.io/3scale-amp26/zync**	yes
SYSTEM_BACKEND_SHARED_SECRET	Shared secret to import events from backend to system.	N/A	yes
SYSTEM_APP_SECRET_KEY_BASE	System application secret key base	N/A	yes
APICAST_MANAGEMENT_API	Scope of the APIcast Management API. Can be disabled, status or debug. At least status required for health checks.	**status**	no
APICAST_OPENSSL_VERIFY	Turn on/off the OpenSSL peer verification when downloading the configuration. Can be set to true/false.	**false**	no
APICAST_RESPONSE_CODES	Enable logging response codes in APIcast.	true	no
APICAST_REGISTRY_URL	A URL which resolves to the location of APIcast policies	**http://apicast-staging:8090/policies**	yes
MASTER_USER	Master administrator account username	**master**	yes
MASTER_NAME	The subdomain value for the master Admin Portal, will be appended with the **-master** suffix	**master**	yes
MASTER_PASSWORD	A randomly generated master administrator password	N/A	yes

Name	Description	Default Value	Required?
MASTER_ACCESS_TOKEN	A token with master level permissions for API calls	N/A	yes
IMAGESTREAM_TAG_IMPORT_INSECURE	Set to true if the server may bypass certificate verification or connect directly over HTTP during image import.	**false**	yes

2.6. USING APICAST WITH 3SCALE ON OPENSHIFT

APIcast is available with API Manager for 3scale Hosted, and in on-premises installations in OpenShift Container Platform. The configuration procedures are different for both. This section explains how to deploy APIcast with API Manager on OpenShift.

2.6.1. Deploying APIcast templates on an existing OpenShift cluster containing 3scale

3scale OpenShift templates contain two embedded APIcast by default. If you require more API gateways, or require separate APIcast deployments, you can deploy additional APIcast templates on your OpenShift cluster.

Perform the following steps to deploy additional API gateways on your OpenShift cluster:

1. Create an access token with the following configurations:
   - Scoped to Account Management API
   - Having read-only access
2. Log in to your APIcast cluster:

   ```
   oc login
   ```

3. Create a secret that allows APIcast to communicate with 3scale. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your 3scale deployment:

   ```
   oc secret new-basicauth apicast-configuration-url-secret --
   password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
   <WILDCARD_DOMAIN>
   ```

   NOTE

   TENANT_NAME is the name under the root that the Admin Portal will be available with. The default value for **TENANT_NAME** is **3scale**. If you used a custom value in your 3scale deployment, you must use that value here.

4. Import the APIcast template using the **oc new-app** command, specifying the **--file** option with the **apicast.yml** file:

```
oc new-app --file /opt/amp/templates/apicast.yml
```

> **NOTE**
>
> First install the APIcast template as described in Section 2.3, "Configuring nodes and entitlements".

2.6.2. Connecting APIcast from a different OpenShift cluster

If you deploy APIcast on a different OpenShift cluster, outside your 3scale cluster, you must connect through the public route:

1. Create an access token with the following configurations:
   - Scoped to Account Management API
   - Having read-only access
2. Log in to your APIcast cluster:

```
oc login
```

3. Create a secret that allows APIcast to communicate with 3scale. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your 3scale deployment:

```
oc secret new-basicauth apicast-configuration-url-secret --
password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
<WILDCARD_DOMAIN>
```

> **NOTE**
>
> **TENANT_NAME** is the name under the root that the Admin Portal will be available with. The default value for **TENANT_NAME** is **3scale**. If you used a custom value in your 3scale deployment, you must use that value.

4. Deploy APIcast on a different OpenShift cluster using the **oc new-app** command. Specify the **--file** option and the to path to your **apicast.yml** file:

```
oc new-app --file /path/to/file/apicast.yml
```

2.6.3. Changing the default behavior for embedded APIcast

In external APIcast deployments, you can modify default behavior by changing the template parameters in the APIcast OpenShift template.

In embedded APIcast deployments, 3scale and APIcast are deployed from a single template. You must modify environment variables after deployment if you wish to change the default behavior for the embedded APIcast deployments.

2.6.4. Connecting multiple APIcast deployments on a single OpenShift cluster over internal service routes

If you deploy multiple APIcast gateways into the same OpenShift cluster, you can configure them to connect using internal routes through the backend listener service instead of the default external route configuration.

You must have an OpenShift SDN plugin installed to connect over internal service routes. How you connect depends on which SDN you have installed:

ovs-subnet

If you are using the **ovs-subnet** OpenShift Software-Defined Networking (SDN) plugin, perform the following steps to connect over internal routes:

1. If not already logged in, log in to your OpenShift cluster:

   ```
   oc login
   ```

2. Enter the following command to display the **backend-listener** route URL:

   ```
   oc get route backend
   ```

3. Enter the **oc new-app** command with the path to **apicast.yml**:

   ```
   oc new-app -f apicast.yml
   ```

ovs-multitenant

If you are using the **ovs-multitenant** OpenShift SDN plugin, perform the following steps to connect over internal routes:

1. If not already logged in, log in to your OpenShift cluster:

   ```
   oc login
   ```

2. As admin, specify the **oadm** command with the **pod-network** and **join-projects** options to set up communication between both projects:

   ```
   oadm pod-network join-projects --to=<3SCALE_PROJECT> <APICAST_PROJECT>
   ```

3. Enter the following command to display the **backend-listener** route URL:

   ```
   oc get route backend
   ```

4. Enter the **oc new-app** command with the path to **apicast.yml**:

   ```
   oc new-app -f apicast.yml
   ```

More information

For information on OpenShift SDN and project network isolation, see Openshift SDN.

2.6.5. Connecting APIcast on other deployments

If you deploy APIcast on Docker, you can connect APIcast to 3scale deployed on OpenShift by setting the **THREESCALE_PORTAL_ENDPOINT** parameter to the URL and access token of your 3scale Admin Portal deployed on OpenShift. You do not need to set the **BACKEND_ENDPOINT_OVERRIDE** parameter in this case.

For more details, see Chapter 5, *APIcast on the Docker containerized environment* .

2.7. DEPLOYING 3SCALE USING THE OPERATOR

This section will take you through installing and deploying the 3scale solution via the 3scale operator, using the *APIManager* custom resource.

NOTE

- Wildcard routes have been removed in 3scale 2.6.
  - This functionality is handled by Zync in the background.
- When API providers are created, updated, or deleted, routes automatically reflect those changes.

2.7.1. Prerequisites

- Section 2.4.2, "Configuring registry authentication in OpenShift"
- Deploying 3scale using the operator first requires that you follow the steps in Chapter 6, *Installing the 3scale operator on OpenShift 4.x*
- OpenShift Container Platform (OCP) 4.x
  - A user account with administrator privileges in the OpenShift cluster
  - **Note:** OCP 4.x supports deployment of 3scale using the operator only.

2.7.2. Deploying the *APIManager* custom resource

Deploying the *APIManager* custom resource will make the operator begin processing and will deploy a 3scale solution from it.

Procedure

1. Click **Catalog > Installed Operators**.
   a. From the list of *Installed Operators*, click *3scale Operator*.
2. Click the *API Manager* tab.
3. Click **Create APIManager**.
4. Clear the sample content and add the following *YAML* definitions to the editor, then click **Create**.

NOTE

The *wildcardDomain* parameter can be any desired name you wish to give that resolves to an IP address, which is a valid DNS domain. Be sure to remove the placeholder marks for your parameters: < >.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  wildcardDomain: <wildcardDomain>
  resourceRequirementsEnabled: true
```

NOTE

For more information about the *APIManager* fields, refer to the Reference documentation.

2.7.3. Getting the *APIManager* administrator credentials

To log in to 3scale after the operator-based deployment, you need the administrator credentials. To get these credentials, perform the steps below:

1. Run these commands:

```
oc get secret system-seed -o json | jq -r .data.ADMIN_USERNAME | base64 -d
oc get secret system-seed -o json | jq -r .data.ADMIN_PASSWORD | base64 -d
```

2. Login as the *APIManager* administrator to verify these credentials are working.

2.7.4. Getting the Admin Portal URL

When you deploy 3scale using the operator, a default tenant is created, with a fixed URL: **3scale-admin.${wildcardDomain}**

The dashboard shows the new portal URL of the tenant. As an example, if the *<wildCardDomain>* is **3scale-project.example.com**, the Admin Portal URL is: **https://3scale-admin.3scale-project.example.com**.

The **wildcardDomain** is the *<wildCardDomain>* parameter you provided during installation. Open this unique URL in a browser using the this command:

```
xdg-open https://3scale-admin.3scale-project.example.com
```

Optionally, you can create new tenants on the *MASTER portal URL*: **master.${wildcardDomain}**

2.8. TROUBLESHOOTING

This section contains a list of common installation issues and provides guidance for their resolution.

- Section 2.8.1, “Previous deployment leaving dirty persistent volume claims”

- Section 2.8.2, “Incorrectly pulling from the Docker registry”
- Section 2.8.3, “Permission issues for MySQL”
- Section 2.8.4, “Unable to upload logo or images”
- Section 2.8.5, “Test calls do not work on OpenShift”
- Section 2.8.6, “APIcast on a different project from 3scale”

2.8.1. Previous deployment leaving dirty persistent volume claims

Problem

A previous deployment attempt leaves a dirty Persistent Volume Claim (PVC) causing the MySQL container to fail to start.

Cause

Deleting a project in OpenShift does not clean the PVCs associated with it.

Solution

1. Find the PVC containing the erroneous MySQL data with the **oc get pvc** command:

```
# oc get pvc
NAME                  STATUS    VOLUME    CAPACITY   ACCESSMODES   AGE
backend-redis-storage   Bound     vol003    100Gi      RWO,RWX       4d
mysql-storage           Bound     vol006    100Gi      RWO,RWX       4d
system-redis-storage    Bound     vol008    100Gi      RWO,RWX       4d
system-storage          Bound     vol004    100Gi      RWO,RWX       4d
```

2. Stop the deployment of the system-mysql pod by clicking **cancel deployment** in the OpenShift UI.
3. Delete everything under the MySQL path to clean the volume.
4. Start a new **system-mysql** deployment.

2.8.2. Incorrectly pulling from the Docker registry

Problem

The following error occurs during installation:

```
svc/system-redis - 1EX.AMP.LE.IP:6379
  dc/system-redis deploys docker.io/rhscl/redis-32-rhel7:3.2-5.3
    deployment #1 failed 13 minutes ago: config change
```

Cause

OpenShift searches for and pulls container images by issuing the **docker** command. This command refers to the **docker.io** Docker registry instead of the **registry.redhat.io** Red Hat container registry.

This occurs when the system contains an unexpected version of the Docker containerized environment.

Solution

Use the appropriate version of the Docker containerized environment.

2.8.3. Permission issues for MySQL

Problem

The system-msql pod crashes and does not deploy causing other systems dependant on it to fail deployment. The pod log displays the following error:

```
[ERROR] Cannot start server : on unix socket: Permission denied
[ERROR] Do you already have another mysqld server running on socket: /var/lib/mysql/mysql.sock ?
[ERROR] Aborting
```

Cause

The MySQL process is started with inappropriate user permissions.

Solution

1. The directories used for the persistent volumes MUST have the write permissions for the root group. Having rw permissions for the root user is not enough as the MySQL service runs as a different user in the root group. Execute the following command as the root user:

   ```
   chmod -R g+w /path/for/pvs
   ```

2. Execute the following command to prevent SElinux from blocking access:

   ```
   chcon -Rt svirt_sandbox_file_t /path/for/pvs
   ```

2.8.4. Unable to upload logo or images

Problem

Unable to upload a logo - **system-app** logs display the following error:

```
Errno::EACCES (Permission denied @ dir_s_mkdir - /opt/system/public//system/provider-name/2
```

Cause

Persistent volumes are not writable by OpenShift.

Solution

Ensure your persistent volume is writable by OpenShift. It should be owned by root group and be group writable.

2.8.5. Test calls do not work on OpenShift

Problem

Test calls do not work after creation of a new service and routes on OpenShift. Direct calls via curl also fail, stating: **service not available**.

Cause

3scale requires HTTPS routes by default, and OpenShift routes are not secured.

Solution

Ensure the **secure route** checkbox is clicked in your OpenShift router settings.

2.8.6. APIcast on a different project from 3scale

Problem

APIcast deploy fails (pod does not turn blue). The following error appears in the logs:

```
update acceptor rejected apicast-3: pods for deployment "apicast-3" took longer than 600 seconds to
become ready
```

The following error appears in the pod:

```
Error synching pod, skipping: failed to "StartContainer" for "apicast" with RunContainerError:
"GenerateRunContainerOptions: secrets \"apicast-configuration-url-secret\" not found"
```

Cause

The secret was not properly set up.

Solution

When creating a secret with APIcast v3, specify **apicast-configuration-url-secret**:

```
oc secret new-basicauth apicast-configuration-url-secret  --
password=https://<ACCESS_TOKEN>@<TENANT_NAME>-admin.<WILDCARD_DOMAIN>
```

CHAPTER 3. INSTALLING APICAST

APIcast is an NGINX based API gateway used to integrate your internal and external API services with the 3scale Platform. APIcast does load balancing by using round-robin.

In this guide you will learn about deployment options, environments provided, and how to get started.

To get information about the latest released and supported version of APIcast, see the articles Red Hat 3scale API Management Supported Configurations and Red Hat 3scale API Management - Component Details.

3.1. PREREQUISITES

APIcast is not a standalone API gateway. It needs connection to 3scale API Manager.

- You will need a working 3scale On-Premises instance.

3.2. DEPLOYMENT OPTIONS

You can use hosted or self-managed APIcast. In both cases, APIcast must be connected to the rest of the 3scale API management platform:

- **Embedded APIcast**: Two APIcast gateways (staging and production) come by default with the 3scale API Management installation. They come pre-configured and ready to use out-of-the-box.
- **Self-managed APIcast**: You can deploy APIcast wherever you want. The self-managed mode is the intended mode of operation for production environments. Here are a few recommended options to deploy APIcast:
  - the Docker containerized environment: Download a ready to use Docker-formatted container image, which includes all of the dependencies to run APIcast in a Docker-formatted container.
  - OpenShift: Run APIcast on a supported version of OpenShift. You can connect self-managed APIcasts both to a 3scale installation or to a 3scale online account.

3.3. ENVIRONMENTS

By default, when you create a 3scale account, you get embedded APIcast in two different environments:

- **Staging**: Intended to be used only while configuring and testing your API integration. When you have confirmed that your setup is working as expected, then you can choose to deploy it to the production environment. The OpenShift template sets the parameters of the Staging APIcast in a way that the configuration is reloaded on each API call (**APICAST_CONFIGURATION_LOADER: lazy**, **APICAST_CONFIGURATION_CACHE: 0**). It is useful to test the changes in APIcast configuration quickly.
- **Production**: This environment is intended for production use. The following parameters are set for the Production APIcast in the OpenShift template: **APICAST_CONFIGURATION_LOADER: boot**, **APICAST_CONFIGURATION_CACHE: 300**. This means that the configuration will be fully loaded when APIcast is started, and will be cached for 300 seconds (5 minutes). After 5 minutes the configuration will be reloaded. This means that when you promote the configuration to production, it may take up to 5 minutes to be applied, unless you trigger a new deployment of APIcast.

3.4. CONFIGURING THE INTEGRATION SETTINGS

Go to **[your_API_name] > Integration > Configuration**

At the top of the Integration page you will see the integration options. By default, the deployment option is hosted APIcast, and the authentication mode is API key. You can change these settings by clicking on **edit integration settings** in the upper right corner.

At the top of the Integration page you will see the integration options. By default, you find these values:

- Deployment option: embedded APIcast.
- Authentication mode: API key.

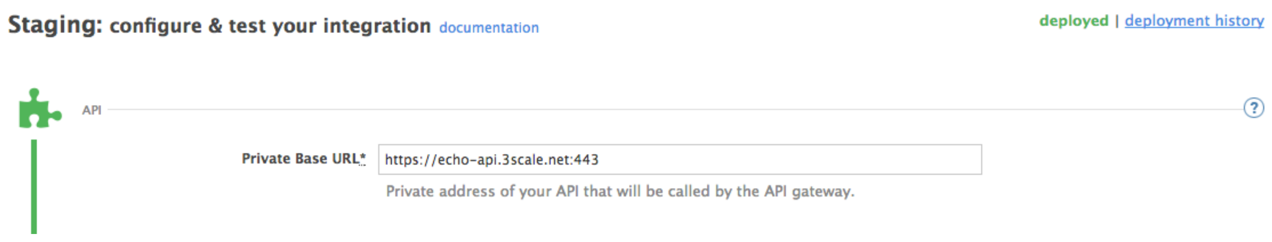
You can change these settings by clicking on **edit integration settings** in the upper right corner.

3.5. CONFIGURING YOUR SERVICE

3.5.1. Declare the API backend

You need to declare your API backend in the Private Base URL field, which is the endpoint host of your API backend. APIcast will redirect all traffic to your API backend after all authentication, authorization, rate limits and statistics have been processed.

Typically, the Private Base URL of your API will be something like **https://api-backend.yourdomain.com:443**, on the domain that you manage (**yourdomain.com**). For instance, if you were integrating with the Twitter API the Private Base URL would be **https://api.twitter.com/**. In this example will use the Echo API hosted by 3scale – a simple API that accepts any path and returns information about the request (path, request parameters, headers, etc.). Its Private Base URL is **https://echo-api.3scale.net:443**


Staging: configure & test your integration
documentation
deployed | deployment history
API
?

Private Base URL*: https://echo-api.3scale.net:443

Private address of your API that will be called by the API gateway.

Test your private (unmanaged) API is working. For example, for the Echo API you can make the following call with **curl** command:

```
curl "https://echo-api.3scale.net:443"
```

You will get the following response:

```
{
    "method": "GET",
    "path": "/",
    "args": "",
    "body": "",
    "headers": {
      "HTTP_VERSION": "HTTP/1.1",
      "HTTP_HOST": "echo-api.3scale.net",
      "HTTP_ACCEPT": "*/*",
    "HTTP_USER_AGENT": "curl/7.51.0",
    "HTTP_X_FORWARDED_FOR": "2.139.235.79, 10.0.103.58",
    "HTTP_X_FORWARDED_HOST": "echo-api.3scale.net",
    "HTTP_X_FORWARDED_PORT": "443",
    "HTTP_X_FORWARDED_PROTO": "https",
    "HTTP_FORWARDED": "for=10.0.103.58;host=echo-api.3scale.net;proto=https"
  },
  "uuid": "ee626b70-e928-4cb1-a1a4-348b8e361733"
}
```

3.5.2. Configure the authentication settings

You can configure authentication settings for your API in the **AUTHENTICATION SETTINGS** section. The following fields are all optional:

Field	Description
Host Header	Define a custom Host request header. This is required if your API backend only accepts traffic from a specific host.
Secret Token	Used to block direct developer requests to your API backend. Set the value of the header here, and ensure your backend only allows calls with this secret header.
Credentials location	Define whether credentials are passed as HTTP headers, query parameters or as HTTP basic authentication.
Auth user key	Set the user key associated with the credentials location
Errors	Define the response code, content type, and response body, for the following errors: authentication failed, authentication missing, no match.

3.5.3. Configure the API test call

You need to configure the test call for the hosted staging environment. Enter a path existing in your API in the **API test GET request field** (for example, **/v1/word/good.json**).

3.5.4. Save the configuration settings

Save the settings by clicking on the **Update & Test Staging Configuration**button in the bottom right part of the page. This will deploy the APIcast configuration to the 3scale hosted staging environment. If everything is configured correctly, the vertical line on the left should turn green.

If you are using one of the Self-managed deployment options, save the configuration from the GUI and make sure it is pointing to your deployed API gateway by adding the correct host in the staging or production Public base URL field. Before making any calls to your production gateway, do not forget to click on the **Promote v.x to Production**button.

Find the sample **curl** at the bottom of the staging section and run it from the console:

```
curl "https://XXX.staging.apicast.io:443/v1/word/good.json?user_key=YOUR_USER_KEY"
```

NOTE

You should get the same response as above, however, this time the request will go through the 3scale hosted APIcast instance. Note: You should make sure you have an application with valid credentials for the service. If you are using the default API service created on sign up to 3scale, you should already have an application. Otherwise, if you see **USER_KEY** or **APP_ID** and **APP_KEY** values in the test curl, you need to create an application for this service first.

Now you have your API integrated with 3scale.

3scale-hosted APIcast gateway does the validation of the credentials and applies the rate limits that you defined for the application plan of the application. If you try to make a call without credentials, or with invalid credentials, you will see an error message.

CHAPTER 4. RUNNING APICAST ON RED HAT OPENSHIFT

This tutorial describes how to deploy the APIcast API Gateway on Red Hat OpenShift.

4.1. PREREQUISITES

To follow the tutorial steps below, you will first need to configure APIcast in your 3scale Admin Portal as per the Installing APIcast guide. Make sure *Self-managed Gateway* is selected as the deployment option in the integration settings. You should have both Staging and Production environment configured to proceed.

4.2. SETTING UP RED HAT OPENSHIFT

If you already have a running OpenShift cluster, you can skip this step. Otherwise, continue reading.

For production deployments you can follow the instructions for OpenShift installation.

In this tutorial the OpenShift cluster will be installed using:

- Red Hat Enterprise Linux (RHEL) 7
- Docker containerized environment v1.10.3
- OpenShift Origin command line interface (CLI) - v1.3.1

4.2.1. Install the Docker containerized environment

Docker-formatted container images provided by Red Hat are released as part of the Extras channel in RHEL. To enable additional repositories, you can use either the Subscription Manager, or yum config manager. See the RHEL product documentation for details.

For a RHEL 7 deployed on a AWS EC2 instance you will use the following the instructions:

1. List all repositories:

   ```
   sudo yum repolist all
   ```

2. Find and enable the ***-extras** repository.

   ```
   sudo yum-config-manager --enable rhui-REGION-rhel-server-extras
   ```

3. Install Docker-formatted container images:

   ```
   sudo yum install docker docker-registry
   ```

4. Add an insecure registry of **172.30.0.0/16** by adding or uncommenting the following line in **/etc/sysconfig/docker** file:

   ```
   INSECURE_REGISTRY='--insecure-registry 172.30.0.0/16'
   ```

5. Start the Docker service:

   ```
   sudo systemctl start docker
   ```

With the following command, you can verify that the container service is running:

```
sudo systemctl status docker
```

4.2.2. Start the OpenShift cluster

Download the latest stable release of the client tools (**openshift-origin-client-tools-VERSION-linux-64bit.tar.gz**) from OpenShift releases page, and place the Linux **oc** binary extracted from the archive in your **PATH**.

NOTE

- Please be aware that the **oc cluster** set of commands are only available in the 1.3+ or newer releases.
- the docker command runs as the **root** user, so you will need to run any **oc** or docker commands with root privileges.

Open a terminal with a user that has permission to run docker commands and run:

```
oc cluster up
```

At the bottom of the output you will find information about the deployed cluster:

```
    -- Server Information ...
      OpenShift server started.
      The server is accessible via web console at:
      https://172.30.0.112:8443

      You are logged in as:
        User:     developer
        Password: developer

      To login as administrator:
        oc login -u system:admin
```

Note the IP address that is assigned to your OpenShift server. You will refer to it in the tutorial as **OPENSHIFT-SERVER-IP**.

4.2.3. Set up the OpenShift cluster on a remote server (Optional)

If you are deploying the OpenShift cluster on a remote server, you will need to explicitly specify a public hostname and a routing suffix on starting the cluster, so that you will be able to access the OpenShift web console remotely.

For example, if you are deploying on an AWS EC2 instance, you should specify the following options:

```
oc cluster up --public-hostname=ec2-54-321-67-89.compute-1.amazonaws.com --routing-suffix=54.321.67.89.xip.io
```

where **ec2-54-321-67-89.compute-1.amazonaws.com** is the Public Domain, and **54.321.67.89** is the IP of the instance. You will then be able to access the OpenShift web console at https://ec2-54-321-67-89.compute-1.amazonaws.com:8443.

4.3. DEPLOYING APICAST USING THE OPENSHIFT TEMPLATE

1. By default you are logged in as *developer* and can proceed to the next step.
   Otherwise login into OpenShift using the **oc login** command from the OpenShift Client tools you downloaded and installed in the previous step. The default login credentials are *username = "developer"* and *password = "developer"*:

   ```
   oc login https://OPENSHIFT-SERVER-IP:8443
   ```

   You should see **Login successful.** in the output.

2. Create your project. This example sets the display name as *gateway*

   ```
   oc new-project "3scalegateway" --display-name="gateway" --description="3scale gateway
   demo"
   ```

   The response should look like this:

   ```
   Now using project "3scalegateway" on server "https://172.30.0.112:8443".
   ```

   Ignore the suggested next steps in the text output at the command prompt and proceed to the next step below.

3. Create a new secret to reference your project by replacing **<access_token>** and **<domain>** with your own credentials. See below for more information about the **<access_token>** and **<domain>**.

   ```
   oc create secret generic apicast-configuration-url-secret --from-
   literal=password=https://<access_token>@<admin_portal_domain>  --
   type=kubernetes.io/basic-auth
   ```

   Here **<access_token>** is an Access Token (not a Service Token) for the 3scale Account Management API, and **<domain>-admin.3scale.net** is the URL of your 3scale Admin Portal.

   The response should look like this:

   ```
   secret/apicast-configuration-url-secret
   ```

4. Create an application for your APIcast gateway from the template, and start the deployment:

   ```
   oc new-app -f https://raw.githubusercontent.com/3scale/3scale-amp-openshift-
   templates/2.6.0.GA/apicast-gateway/apicast.yml
   ```

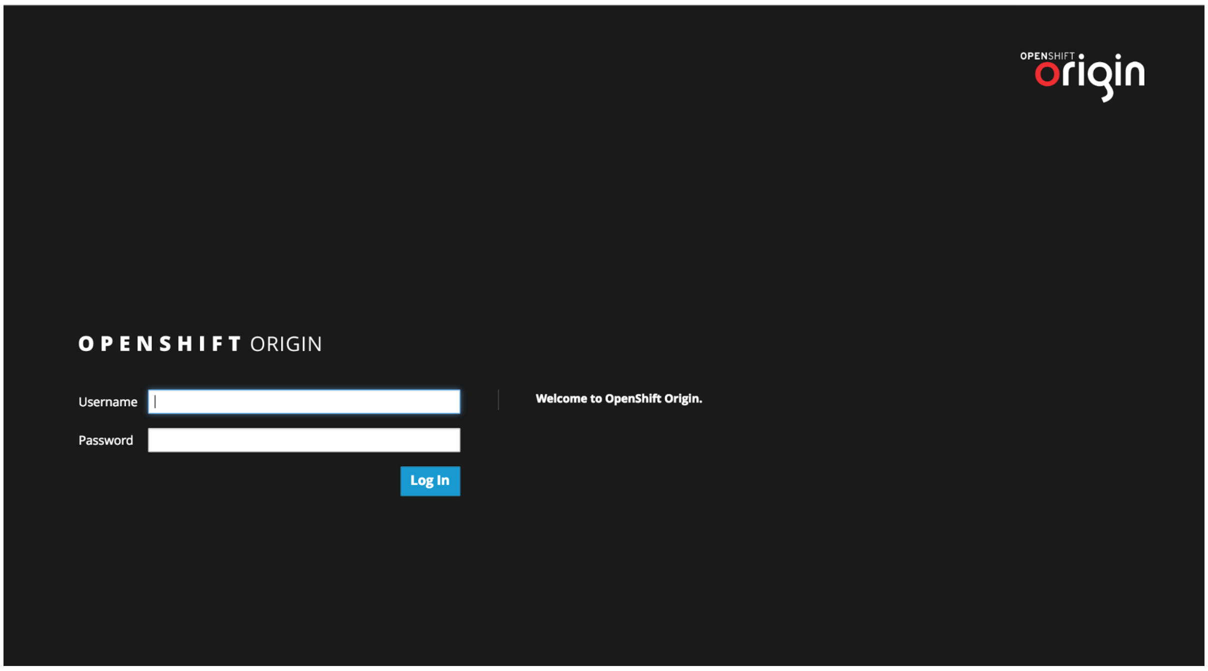
   You should see the following messages at the bottom of the output:

   ```
   --> Creating resources with label app=3scale-gateway ...
       deploymentconfig "apicast" created
       service "apicast" created
   --> Success
       Run 'oc status' to view your app.
   ```

4.4. CREATING ROUTES VIA THE OPENSHIFT CONSOLE

1. Open the web console for your OpenShift cluster in your browser: https://OPENSHIFT-SERVER-IP:8443/console/
   Use the value specified in **--public-hostname** instead of **OPENSHIFT-SERVER-IP** if you started OpenShift cluster on a remote server.

   You should see the login screen:

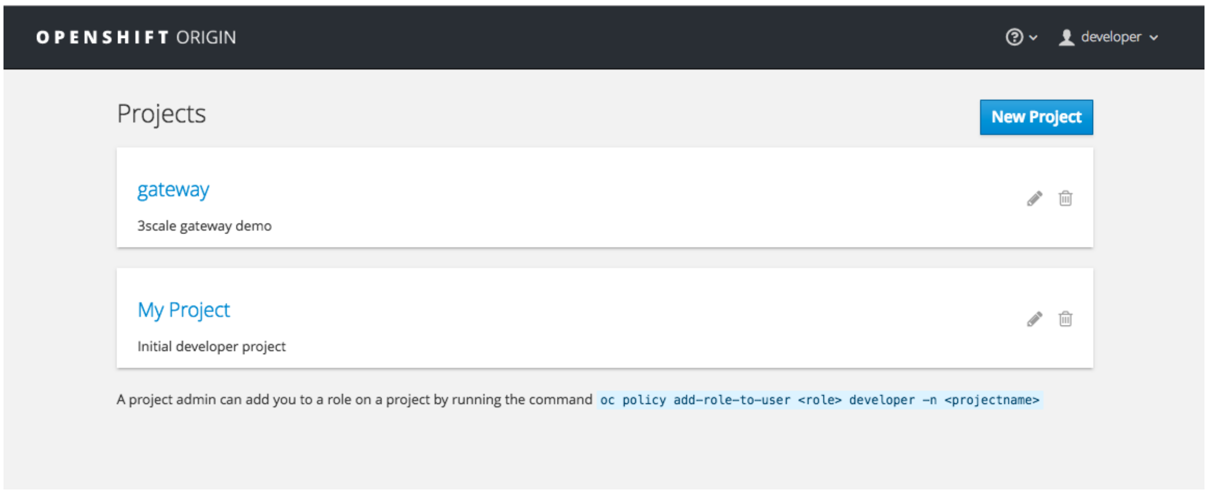


Welcome to OpenShift Origin.

NOTE

You may receive a warning about an untrusted web-site. This is expected, as you are trying to access the web console through secure protocol, without having configured a valid certificate. While you should avoid this in production environment, for this test setup you can go ahead and create an exception for this address.

2. Log in using the *developer* credentials created or obtained in the *Setup OpenShift* section above.
   You will see a list of projects, including the *gateway* project you created from the command line above.



gateway
3scale gateway demo

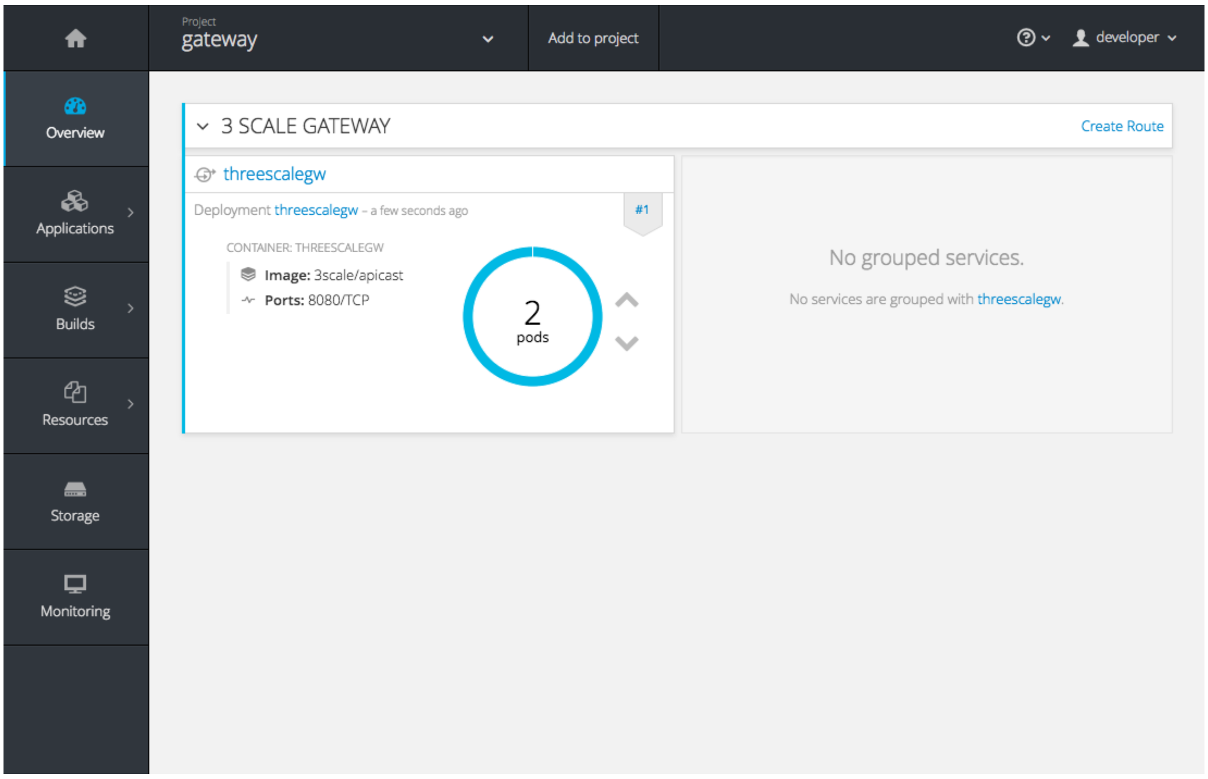
My Project
Initial developer project

A project admin can add you to a role on a project by running the command `oc policy add-role-to-user <role> developer -n <projectname>`

If you do not see your gateway project, you probably created it with a different user and will need to assign the policy role to to this user.

3. Click on the *gateway* link and you will see the *Overview* tab.
   OpenShift downloaded the code for APIcast and started the deployment. You may see the message *Deployment #1 running* when the deployment is in progress.

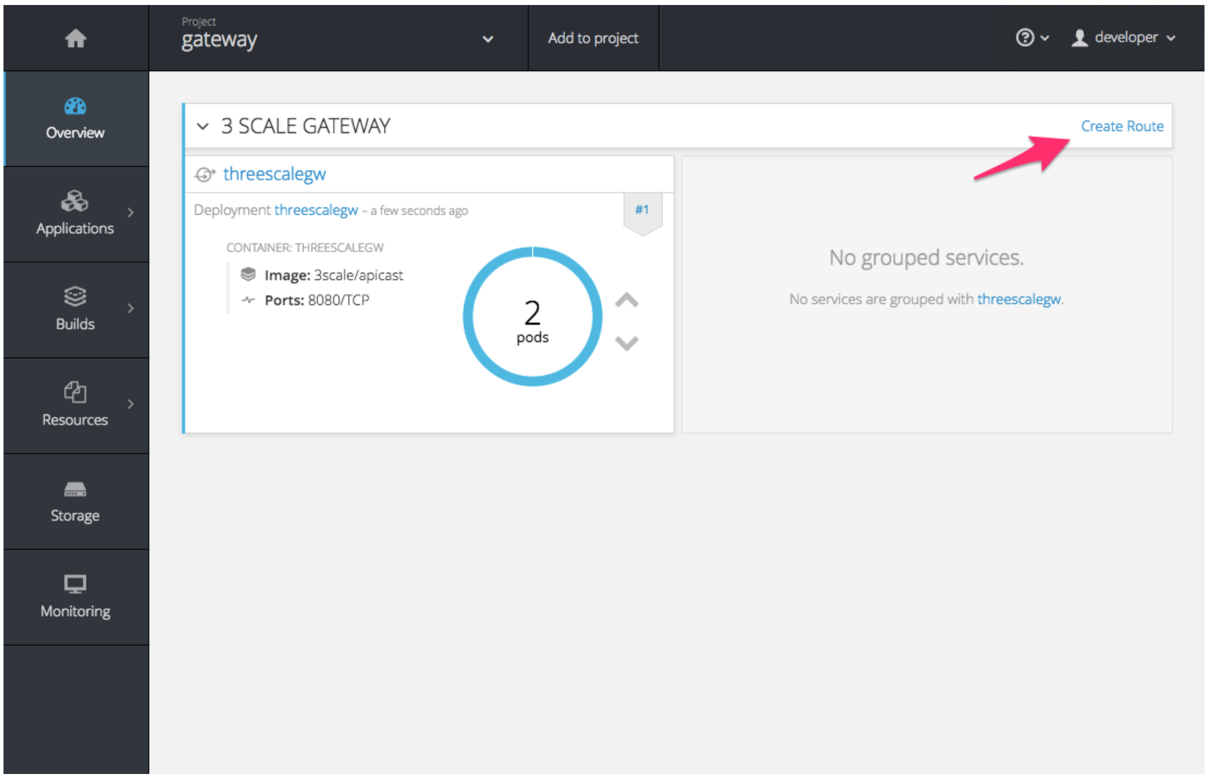
   When the build completes, the UI will refresh and show two instances of APIcast (*2 pods*) that have been started by OpenShift, as defined in the template.



   Each APIcast instance, upon starting, downloads the required configuration from 3scale using the settings you provided on the **Integration** page of your 3scale Admin Portal.

   OpenShift will maintain two APIcast instances and monitor the health of both; any unhealthy APIcast instance will automatically be replaced with a new one.

4. To allow your APIcast instances to receive traffic, you need to create a route. Start by clicking on **Create Route**.



Deployment threescalegw – a few seconds ago

CONTAINER: THREESCALEGW

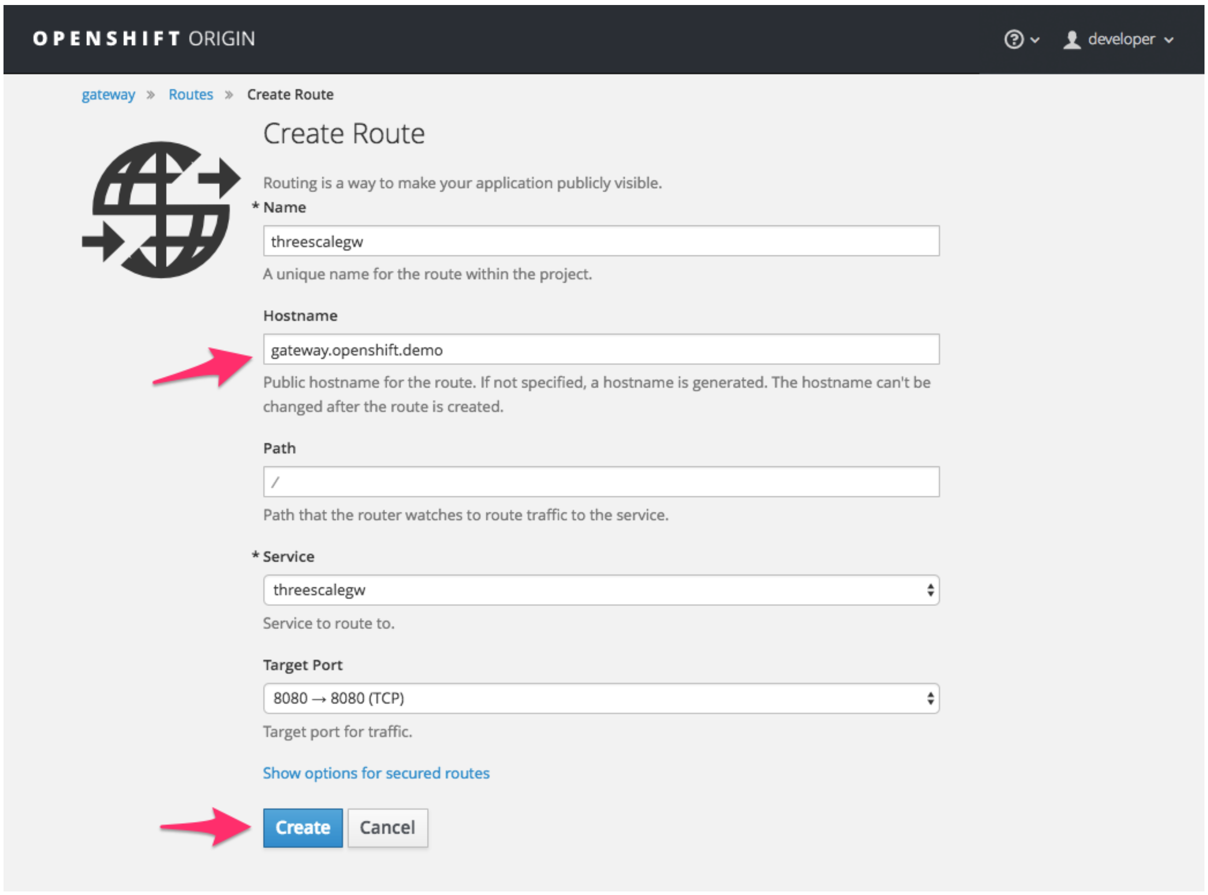
Image: 3scale/apicast

Ports: 8080/TCP

No grouped services.

No services are grouped with threescalegw.

Enter the same host you set in 3scale above in the section **Public Base URL** (without the *http://* and without the port) , e.g. **gateway.openshift.demo**, then click the **Create** button.


OPENSHIFT ORIGIN
developer
gateway » Routes » Create Route
Show options for secured routes
Create
Cancel

Create Route

Routing is a way to make your application publicly visible.

* Name

threescalegw

A unique name for the route within the project.

Hostname

gateway.openshift.demo

Public hostname for the route. If not specified, a hostname is generated. The hostname can't be changed after the route is created.

Path

/

Path that the router watches to route traffic to the service.

* Service

threescalegw

Service to route to.

Target Port

8080 → 8080 (TCP)

Target port for traffic.

For every 3scale service you define, you must create a new route.

CHAPTER 5. APICAST ON THE DOCKER CONTAINERIZED ENVIRONMENT

This is a step-by-step guide to deploy APIcast inside a Docker-formatted container that is ready to be used as a 3scale API gateway.

5.1. PREREQUISITES

You must configure APIcast in your 3scale Admin Portal as per the the Installing APIcast guide.

5.2. INSTALLING THE DOCKER CONTAINERIZED ENVIRONMENT

This guide covers the steps to set up the Docker containerized environment on Red Hat Enterprise Linux (RHEL) 7.

Docker-formatted containers provided by Red Hat are released as part of the Extras channel in RHEL. To enable additional repositories, you can use either the Subscription Manager or the yum config manager. For details, see the RHEL product documentation.

To deploy RHEL 7 on an AWS EC2 instance, take the following steps:

1. List all repositories: **sudo yum repolist all**.
2. Find the ***-extras** repository.
3. Enable the **extras** repository: **sudo yum-config-manager --enable rhui-REGION-rhel-server-extras**.
4. Install the Docker containerized environment package: **sudo yum install docker**.

For other operating systems, refer to the following Docker documentation:

- Installing the Docker containerized environment on Linux distributions
- Installing the Docker containerized environment on Mac
- Installing the Docker containerized environment on Windows

5.3. RUNNING THE DOCKER CONTAINERIZED ENVIRONMENT GATEWAY

1. Start the Docker daemon:
   sudo systemctl start docker.service.
2. Check if the Docker daemon is running:
   sudo systemctl status docker.service.

   You can download a ready to use Docker-formatted container image from the Red Hat registry:

   sudo docker pull registry.access.redhat.com/3scale-amp26/apicast-gateway.
3. Run APIcast in a Docker-formatted container:

sudo docker run --name apicast --rm -p 8080:8080 -e THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net registry.access.redhat.com/3scale-amp26/apicast-gateway.

Here, **<access_token>** is the Access Token for the 3scale Account Management API. You can use the Provider Key instead of the access token. **<domain>-admin.3scale.net** is the URL of your 3scale admin portal.

This command runs a Docker-formatted container called *"apicast"* on port **8080** and fetches the JSON configuration file from your 3scale portal. For other configuration options, see the Installing APIcast guide.

5.3.1. The Docker command options

You can use the following options with the **docker run** command:

- **--rm**: Automatically removes the container when it exits.
- **-d** or **--detach**: Runs the container in the background and prints the container ID. When it is not specified, the container runs in the foreground mode and you can stop it using **CTRL + c**. When started in the detached mode, you can reattach to the container with the **docker attach** command, for example, **docker attach apicast**.
- **-p** or **--publish**: Publishes a container's port to the host. The value should have the format **<host port="">:<container port="">**, so **-p 80:8080** will bind port **8080** of the container to port **80** of the host machine. For example, the Management API uses port **8090**, so you may want to publish this port by adding **-p 8090:8090** to the **docker run** command.
- **-e** or **--env**: Sets environment variables.
- **-v** or **--volume**: Mounts a volume. The value is typically represented as **<host path="">:<container path="">[:<options>]**. **<options>** is an optional attribute; you can set it to **:ro** to specify that the volume will be read only (by default, it is mounted in read-write mode). Example: **-v /host/path:/container/path:ro**.

For more information on available options, see Docker run reference.

5.4. TESTING APICAST

The preceding steps ensure that your Docker-formatted container is running with your own configuration file and the Docker-formatted image from the 3scale registry. You can test calls through APIcast on port **8080** and provide the correct authentication credentials, which you can get from your 3scale account.

Test calls will not only verify that APIcast is running correctly but also that authentication and reporting is being handled successfully.

NOTE

Ensure that the host you use for the calls is the same as the one configured in the **Public Base URL** field on the **Integration** page.

CHAPTER 6. INSTALLING THE 3SCALE OPERATOR ON OPENSHIFT 4.X

This documentation shows you how to:

- Create a new project.
- Deploy a 3scale instance.
- Create the **threescale-registry-auth** secret in the project.
- Install the 3scale operator through Operator Lifecycle Manager (OLM).
- Deploy the custom resources once the operator has been deployed.

Prerequisites

- Access with administrator privileges to an OpenShift Container Platform (OCP) 4.1 cluster.
  - **Note:** OCP 4.x supports deployment of 3scale using the operator only.

WARNING

Deploy the 3scale operator and custom resource definitions (CRDs) in a separate newly created, empty *project*. If you deploy them in an existing *project* containing infrastructure, it could alter or delete existing elements.

6.1. CREATING A NEW OPENSHIFT PROJECT

Run the command shown below to create a new OpenShift project. Create the new project, and install the operator and the custom resources. The operator manages the custom resources through OLM in that project.

NOTE

This command shows an example of a project name, **operator-test**. Replace this project name with your own.

```
oc new-project operator-test
```

This will create a new *OpenShift project* where the operator, the *APIManager* custom resource (CR) and the *Capabilities* custom resources will be installed.

6.2. INSTALLING AND CONFIGURING THE 3SCALE OPERATOR USING THE OLM

Use OLM to install the 3scale Operator on an OpenShift Container Platform 4.1 cluster by using OperatorHub in the OpenShift Container Platform console.

NOTE

You must install and deploy the 3scale operator in the **operator-test** project.

Procedure

1. In the OpenShift Container Platform console, log in using an account with administrator privileges.
2. Click **Catalog** > **OperatorHub**.
3. In the **Filter by keyword** box, type *3scale operator* to find the 3scale operator.
4. Click the 3scale operator. Information about the Operator is displayed.
5. Read the information about the operator and click **Install**. The Create Operator Subscription page opens.
6. On the **Create Operator Subscription** page, accept all of the default selections and click **Subscribe**.

   NOTE

   The operator will only be available in the specific single namespace on the cluster that you have selected.

   The *3scale-operator* details page is displayed, where you can see the *Subscription Overview*.
7. After the subscription **upgrade status** is shown as **Up to date**, click **Catalog** > **Installed Operators** to verify that the 3scale operator ClusterServiceVersion (CSV) is displayed and its Status ultimately resolves to InstallSucceeded in the **operator-test** project.
   a. Successful installation will register the *APIManager* CRD and the CRDs related to the *Capabilities* functionality of the operator in the *OpenShift API server*.
8. After successful installation, query the resource types defined by the CRDs via **oc get**.
   a. For example, to verify that the *APIManager* CRD has been correctly registered, execute the following command:

      ```
      oc get apimanagers
      ```
9. You should see the following output:

   ```
   No resources found.
   ```

   For troubleshooting information, see the OpenShift Container Platform documentation.

CHAPTER 7. 3SCALE HIGH AVAILABILITY AND EVALUATION TEMPLATES

7.1. INTRODUCTION

This document describes the templates for High Availability and Evaluation used by Red Hat 3scale API Management 2.6 On-Premises installation.

IMPORTANT

The 3scale High Availability and Evaluation templates are a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

7.2. PREREQUISITES

- You need to have an available OpenShift cluster to deploy elements of the High Availability and Evaluation templates.

7.3. HIGH AVAILABILITY TEMPLATE

The High Availability (HA) template allows you to have a HA setting for critical databases.

7.3.1. Prerequisites

- Before deploying the HA template, you must deploy and configure the external databases, and configure them in a HA configuration with a load-balanced endpoint.

7.3.2. Using the HA template

For HA, the template named **amp-ha-tech-preview.yml** allows you to deploy critical databases externally to OpenShift. This excludes:

- Memcached
- Sphinx
- Zync

Differences between the standard **amp.yml** template and **amp-ha-tech-preview.yml** include:

- Removal of the following elements:
  - backend-redis and its related components
  - system-redis and its related components
  - system-mysql and its related components

    - Redis and MySQL related ConfigMaps
    - MYSQL_IMAGE, REDIS_IMAGE, MYSQL_USER, MYSQL_ROOT_PASSWORD parameters
- By default, increased from 1 to 2 the number of replicas for non-database **DeploymentConfig** object types.
- Addition of the following mandatory parameters, allowing you the control of the location of external databases:
    - BACKEND_REDIS_STORAGE_ENDPOINT
    - BACKEND_REDIS_QUEUES_ENDPOINT
    - SYSTEM_REDIS_URL
    - APICAST_STAGING_REDIS_URL
    - APICAST_PRODUCTION_REDIS_URL
    - SYSTEM_DATABASE_URL

With **amp-ha-tech-preview.yml**, you need to configure database connections (excluding **system-memcache**, **zync-database** and **system-sphinx** that do not contain permanent data) out of the cluster via the newly added mandatory parameters. The endpoints require database load-balanced connection strings, including authentication information. Also, for the non-database deployments, the number of pod replicas is increased to 2 by default to have redundancy at application-level.

7.4. EVALUATION TEMPLATE

For evaluation purposes, there is a template named **amp-eval-tech-preview.yml** that deploys a 3scale environment without resource requests nor limits.

The only functional difference compared to the standard **amp.yml** template is that the resource limits and requests have been removed. This means that in this version the minimum hardware requirements have been removed on the pods at CPU and Memory level. This template is intended only for evaluation, testing, and development purposes as it tries to deploy the components in a best-effort way with the given hardware resources.

CHAPTER 8. REDIS HIGH AVAILABILITY (HA) SUPPORT FOR 3SCALE

NOTE

There are known issues with the operator and **backend-redis**. For more information, see the Red Hat 3scale API Management 2.6 release notes, Chapter 6. Known issues in the release notes.

High availability (HA) is provided for most components by the OpenShift Container Platform (OCP). For more information see OpenShift Container Platform 3.11 Chapter 30. High Availability .

The database components for HA in 3scale include:

- **backend-redis**: used for statistics storage and temporary job storage.
- **system-redis**: provides temporary storage for background jobs for 3scale and is also used as a message bus for *Ruby* processes of **system-app** pods.

NOTE

Both **backend-redis** and **system-redis** include supported Redis high availability variants for Redis Sentinel and Redis Enterprise.

8.1. SETTING UP REDIS FOR ZERO DOWNTIME

If zero downtime is required, you must configure Redis outside of OCP. There are several ways to set it up using the configuration options of 3scale pods:

- Set up your own self-managed Redis
- Use Redis Sentinel: Reference Redis Sentinel Documentation
- Redis provided as a service:
  For example by:
  - Amazon ElastiCache
  - Redis Labs

NOTE

Red Hat does not provide support for the above mentioned services. The mention of any such services does not imply endorsement by Red Hat of the products or services. You agree that Red Hat is not responsible or liable for any loss or expenses that may result due to your use of (or reliance on) any external content.

8.2. CONFIGURING BACK-END COMPONENTS FOR 3SCALE

There are configuration settings in 3scale 2.6 to configure Redis HA (failover) for the **back-end** component. You can configure these settings as environment variables in the following deployment configurations: **backend-cron**, **backend-listener**, and **backend-worker**.

NOTE

If you want to use sentinels instead of Redis, you must create the **system-redis** secret with all fields in order to configure the Redis you want to point to before deploying 3scale. The fields are not provided as parameters in the back end as of 3scale 2.6.

8.2.1. Creating backend-redis and system-redis secrets

Follow these steps to create **backend-redis** and **system-redis** secrets accordingly if

- Section 8.2.1.1, "Deploying a fresh installation of 3scale for HA"
- Section 8.2.1.2, "Migrating a non-HA deployment of 3scale to HA"

8.2.1.1. Deploying a fresh installation of 3scale for HA

1. Create the **backend-redis** and **system-redis** secrets with all fields below:

   backend-redis

   ```
   REDIS_QUEUES_SENTINEL_HOSTS
   REDIS_QUEUES_SENTINEL_ROLE
   REDIS_QUEUES_URL
   REDIS_STORAGE_SENTINEL_HOSTS
   REDIS_STORAGE_SENTINEL_ROLE
   REDIS_STORAGE_URL
   ```

   system-redis

   ```
   MESSAGE_BUS_NAMESPACE
   MESSAGE_BUS_SENTINEL_HOSTS
   MESSAGE_BUS_SENTINEL_ROLE
   MESSAGE_BUS_URL
   NAMESPACE
   SENTINEL_HOSTS
   SENTINEL_ROLE
   URL
   ```

   - When configuring for Redis with sentinels, the corresponding **URL** fields in **backend-redis** and **system-redis** refer to the Redis group in the format **redis://[:redis-password@]redis-group[/db]**`, where [x] denotes optional element **x** and **redis-password**, **redis-group**, and **db** are variables to be replaced accordingly:

     Example

     ```
     redis://:redispwd@mymaster/5
     ```

   - The **SENTINEL_HOSTS** fields are comma-separated lists of sentinel connection strings in the following format:

     ```
     [redis://][:sentinel-password@]sentinel-hostname-or-ip[:port]
     ```

- For each element of the list, [x] denotes optional element **x** and **sentinel-password**, **sentinel-hostname-or-ip**, and **port** are variables to be replaced accordingly:

  Example

  ```
  :sentinelpwd@123.45.67.009:2711,:sentinelpwd@other-sentinel:2722
  ```

- The **SENTINEL_ROLE** fields are either **master** or **slave**.

2. Deploy 3scale as indicated in the Installation guide for 3scale 2.6 on OpenShift, using the latest version of the templates.

   a. Ignore the errors due to **backend-redis** and **system-redis** already present.

8.2.1.2. Migrating a non-HA deployment of 3scale to HA

1. Edit the **backend-redis** and **system-redis** secrets with all fields as shown in Section 8.2.1.1, "Deploying a fresh installation of 3scale for HA".

2. Make sure the following **backend-redis** environment variables are defined for the back-end pods.

```
name: BACKEND_REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: REDIS_STORAGE_SENTINEL_HOSTS
      name: backend-redis
name: BACKEND_REDIS_SENTINEL_ROLE
  valueFrom:
    secretKeyRef:
      key: REDIS_STORAGE_SENTINEL_ROLE
      name: backend-redis
```

3. Make sure the following **system-redis** environment variables are defined for the **system-(app|sidekiq|sphinx)** pods.

```
name: REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: SENTINEL_HOSTS
      name: system-redis
name: REDIS_SENTINEL_ROLE
  valueFrom:
    secretKeyRef:
      key: SENTINEL_ROLE
      name: system-redis
name: MESSAGE_BUS_REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: MESSAGE_BUS_SENTINEL_HOSTS
      name: system-redis
name: MESSAGE_BUS_REDIS_SENTINEL_ROLE
  valueFrom:
```

```
    secretKeyRef:
      key: MESSAGE_BUS_SENTINEL_ROLE
      name: system-redis
```

4. Proceed with instructions to continue upgrading 3scale.

8.2.2. Using Redis Enterprise

1. Use Redis Enterprise deployed in OpenShift, with three different **redis-enterprise** instances:
   a. Edit **system-redis** secret:
      i. Set distinct values to **MESSAGE_BUS_NAMESPACE** and **NAMESPACE**.
      ii. Set **URL** and **MESSAGE_BUS_URL** to the same database.
   b. Set the back-end database in **backend-redis** to **REDIS_QUEUES_URL**
   c. Set the third database to **REDIS_STORAGE_URL** for **backend-redis**.

8.2.3. Using Redis Sentinel

1. Use Redis Sentinel, with three or four different Redis databases:
   a. Edit **system-redis** secret:
      i. Set distinct values to **MESSAGE_BUS_NAMESPACE** and **NAMESPACE**.
      ii. Set **URL** and **MESSAGE_BUS_URL** to the proper Redis group, for example: **redis://:redispwd@mymaster/5**
      iii. Set **SENTINEL_HOSTS** and **MESSAGE_BUS_SENTINEL_HOSTS** to a comma-separated list of sentinels hosts and ports, for example: **:sentinelpwd@123.45.67.009:2711,:sentinelpwd@other-sentinel:2722**
      iv. Set **SENTINEL_ROLE** and **MESSAGE_BUS_SENTINEL_ROLE** to *master*
2. Set the **backend-redis** secret for back-end with the values:
   - **REDIS_QUEUES_URL**
   - **REDIS_QUEUES_SENTINEL_ROLE**
   - **REDIS_QUEUES_SENTINEL_HOSTS**
3. Set the variables in the third database as follows:
   - **REDIS_STORAGE_URL**
   - **REDIS_STORAGE_SENTINEL_ROLE**
   - **REDIS_STORAGE_SENTINEL_HOSTS**

Notes

- The *system-app* and *system-sidekiq* components connect directly to **back-end** Redis for retrieving statistics.

  - As of 3scale 2.7, these system components can also connect to **back-end** Redis (storage) when using sentinels.

- The *system-app* and *system-sidekiq* components uses **only backend-redis** storage, not **backend-redis** queues.

  - Changes made to the system components support **backend-redis** storage with sentinels.

CHAPTER 9. CONFIGURING AN EXTERNAL MYSQL DATABASE

This guide provides information for externalizing the MySQL database for Chapter 7, *3scale High Availability and Evaluation templates*. This can be done by using the default *amp.yml* file. This is useful where there are several infrastructure issues, such as network or filesystem, using the default **system-mysql** pod.

The difference between this approach and the one in Chapter 7, *3scale High Availability and Evaluation templates* is that this provides a way for externalizing a MySQL database in case Red Hat 3scale API Management was initially using the default *amp.yml* template.

9.1. EXTERNAL MYSQL DATABASE LIMITATIONS

There are limitations with the process of externalizing your MySQL database:

3scale On-premises versions

It has only been tested and verified on the 2.5 On-premises and 2.6 On-premises versions from 3scale.

MySQL database user

With the *mysql2://* formatted URL, you must use *'root'@'%'* or the connection to the database will fail. Using any combination of *username* and *password* is not supported since 3scale uses *'root'@'%'*.

MySQL host

Use the *IP address* from the external MySQL database instead of the *hostname* or it will not resolve. For example, use *1.1.1.1* instead of *mysql.mydomain.com*.

System database

The remote MySQL server must not have a currently existing database named **system**.

9.2. EXTERNALIZING THE MYSQL DATABASE

Use the following steps to fully externalize the MySQL database.

Prerequisites

- Access to an OpenShift Container Platform 3.11 cluster using an account with administrator privileges.
- A 3scale instance installation on the OpenShift cluster. See Chapter 2, *Installation guide for 3scale on OpenShift*.

WARNING

This will cause downtime in the environment while the process is ongoing.

Procedure

1. Login to the OpenShift node where your 3scale On-premises instance is hosted and change to its project:

   ```
   oc login -u <user> <url>
   oc project <3scale-project>
   ```

   Replace **<user>**, **<url>**, and **<3scale-project>** with your own credentials and the project name.

2. Follow the steps below in the order shown to scale down all the pods. This will avoid loss of data.

   ### Stop 3scale On-premises

   From the OpenShift web console or from the command line interface (CLI), scale down all the deployment configurations to zero replicas in the following order:

   - **apicast-wildcard-router** and **zync** for versions before 3scale 2.6 or **zync-que** and **zync** for 3scale 2.6 and above.
   - **apicast-staging** and **apicast-production**.
   - **system-sidekiq**, **backend-cron**, and **system-sphinx**.
     - 3scale 2.3 includes **system-resque**.
   - **system-app**.
   - **backend-listener** and **backend-worker**.
   - **backend-redis**, **system-memcache**, **system-mysql**, **system-redis**, and **zync-database**. The following example shows how to perform this in the CLI for **apicast-wildcard-router** and **zync**:

     ```
     oc scale dc/apicast-wildcard-router --replicas=0
     oc scale dc/zync --replicas=0
     ```

     NOTE

     The deployment configuration for each step can be scaled down at the same time. For example, you could scale down **apicast-wildcard-router** and **zync** together. However, it is better to wait for the pods from each step to terminate before scaling down the ones that follow. The 3scale instance will be completely inaccessible until it is fully started again.

3. To confirm that no pods are running on the 3scale project use the following command:

   ```
   oc get pod
   ```

   The command should return *No resources found*.

4. Scale up the database level pods again using the following command:

   ```
   oc scale dc/{backend-redis,system-memcache,system-mysql,system-redis,zync-database} --
   replicas=1
   ```

5. Ensure that you are able to login to the external MySQL database through the **system-mysql** pod before proceeding with the next steps:

```
oc rsh system-mysql-<system_mysql_pod_id>
mysql -u root -p -h <host>
```

- *<system_mysql_pod_id>*: The identifier of the system-mysql pod.
- The user should always be root. For more information see Section 9.1, "External MySQL database limitations".

    a. The CLI will now display **mysql>**. Type *exit*, then press *return*. Type *exit* again at the next prompt to go back to the OpenShift node console.

6. Perform a full MySQL dump using the following command:

```
oc rsh system-mysql-<system_mysql_pod_id> /bin/bash -c "mysqldump -u root --single-transaction --routines --triggers --all-databases" > system-mysql-dump.sql
```

- Replace *<system_mysql_pod_id>* with your unique **system-mysql** pod *ID* .
- Validate that the file **system-mysql-dump.sql** contains a valid MySQL level dump as in the following example:

```
$ head -n 10 system-mysql-dump.sql
-- MySQL dump 10.13  Distrib 5.7.24, for Linux (x86_64)
--
-- Host: localhost    Database:
-- ------------------------------------------------------
-- Server version   5.7.24

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8 */;
```

7. Scale down the **system-mysql** pod and leave it with 0 (zero) replicas:

```
oc scale dc/system-mysql --replicas=0
```

8. Find the *base64* equivalent of the URL **mysql2://root:<password>@<host>/system**, replacing <password> and <host> accordingly:

```
echo "mysql2://root:<password>@<host>/system" | base64
```

9. Create a default *'user'@'%'* on the remote MySQL database. It only needs to have SELECT privileges. Also find its *base64* equivalents:

```
echo "user" | base64
echo "<password>" | base64
```

- Replace <password> with the password for *'user'@'%'*.

10. Perform a backup and edit the OpenShift secret **system-database**:

    ```
    oc get secret system-database -o yaml > system-database-orig.bkp.yml
    oc edit secret system-database
    ```

    - **URL**: Replace it with the value from [step-8].
    - **DB_USER** and **DB_PASSWORD**: Use the values from the previous step for both.

11. Use the command below to send **system-mysql-dump.sql** to the remote database server and import the dump into the server:

    ```
    mysql -u root -p < system-mysql-dump.sql
    ```

12. Ensure that a new database called *system* was created:

    ```
    mysql -u root -p -se "SHOW DATABASES"
    ```

13. Use the following instructions to *Start 3scale On-premises*, which scales up all the pods in the correct order.

    Start 3scale On-premises

    - **backend-redis**, **system-memcache**, **system-mysql**, **system-redis**, and **zync-database**.
    - **backend-listener** and **backend-worker**.
    - **system-app**.
    - **system-sidekiq**, **backend-cron**, and **system-sphinx**
        - 3scale 2.3 includes **system-resque**.
    - **apicast-staging** and **apicast-production**.
    - **apicast-wildcard-router** and **zync** for versions before 3scale 2.6 or **zync-que** and **zync** for 3scale 2.6 and above.
      The following example shows how to perform this in the CLI for **backend-redis**, **system-memcache**, **system-mysql**, **system-redis**, and **zync-database**:

      ```
      oc scale dc/backend-redis --replicas=1
      oc scale dc/system-memcache --replicas=1
      oc scale dc/system-mysql --replicas=1
      oc scale dc/system-redis --replicas=1
      oc scale dc/zync-database --replicas=1
      ```

      The **system-app** pod should now be up and running without any issues.

14. After validation, scale back up the other pods in the order shown.

15. Backup the **system-mysql** *DeploymentConfig* object. You may delete after a few days once you are sure everything is running properly. Deleting **system-mysql** *DeploymentConfig* avoids any future confusion if this procedure is done again in the future.

9.3. ROLLING BACK

Perform a rollback procedure if the **system-app** pod is not fully back online and the root cause for it could not be determined or addressed after following step 13.

1. Edit the secret **system-database** using the original values from **system-database-orig.bkp.yml**. See [step-10]:

   ```
   oc edit secret system-database
   ```

   Replace *URL*, *DB_USER*, and *DB_PASSWORD* with their original values.

2. Scale down all the pods and then scale them back up again, including **system-mysql**. The **system-app** pod and the other pods to be started after it should be up and running again. Run the following command to confirm all pods are back up and running:

   ```
   oc get pods -n <3scale-project>
   ```