Red Hat 3scale API Management 2.9

Installing 3scale

Install and configure 3scale API Management.

Last Updated: 2020-11-24

Red Hat 3scale API Management 2.9 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

This guide will help you to install and configure 3scale

CHAPTER 1. REGISTRY SERVICE ACCOUNTS FOR 3SCALE

To use container images from **registry.redhat.io** in a shared environment with 3scale 2.9, you must use a *Registry Service Account* instead of an individual user's *Customer Portal* credentials.

IMPORTANT

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

To create and modify a registry service account, perform the steps outlined in the following sections:

- Section 1.1, "Creating a registry service account"
- Section 1.2, "Modifying a registry service account"

1.1. CREATING A REGISTRY SERVICE ACCOUNT

To create a registry service account, follow the procedure below.

Procedure

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 numerical string before the form field.
   b. Enter a *Description*.
   c. Click **Create**.
4. Navigate back to your *Service Accounts*.
5. Click the *Service Account* you created.
6. Make a note of the username, including the prefix string, for example **12345678|username**, and your password. 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.

1.2. MODIFYING A REGISTRY SERVICE ACCOUNT

You can edit or delete service accounts from the *Registry Service Account* page, by 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: You cannot modify the username for the *Service Account*.
- **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*.

1.3. ADDITIONAL RESOURCES

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

CHAPTER 2. INSTALLING 3SCALE ON OPENSHIFT

This section walks you through steps to deploy Red Hat 3scale API Management 2.9 on OpenShift.

The Red Hat 3scale API Management 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 2.3, "Deploying 3scale on OpenShift using a template"
- Section 2.6, "Deploying 3scale using the operator"

NOTE

- Whether deploying 3scale using the operator or via templates, you must first configure registry authentication to the Red Hat Ecosystem Catalog. For more details, see Section 2.3.1, "Configuring registry authentication in OpenShift".
- The 3scale Istio Adapter is available as an optional adapter that allows labeling a service running within the Red Hat OpenShift Service Mesh, and integrate that service with Red Hat 3scale API Management. Refer to 3scale adapter documentation for more information.

Prerequisites

- You must configure 3scale servers for UTC (Coordinated Universal Time).
- Create user credentials using the step in Chapter 1, *Registry service accounts for 3scale*

To install 3scale on OpenShift, perform the steps outlined in the following sections:

- Section 2.1, "System requirements for installing 3scale on OpenShift"
- Section 2.2, "Configuring nodes and entitlements"
- Section 2.3, "Deploying 3scale on OpenShift using a template"
- Section 2.4, "Parameters of the 3scale template"
- Section 2.5, "Using APIcast with 3scale on OpenShift"
- Section 2.6, "Deploying 3scale using the operator"
- Section 2.8, "Troubleshooting common 3scale installation issues"

2.1. SYSTEM REQUIREMENTS FOR INSTALLING 3SCALE ON OPENSHIFT

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

Environment requirements

Red Hat 3scale API Management 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

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

Hardware requirements

Hardware requirements depend on your usage needs. Red Hat recommends that you test and configure your environment to meet your specific requirements. The 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 back-end listener to the number of cores in your compute node.

2.2. CONFIGURING NODES AND ENTITLEMENTS

Before deploying 3scale on OpenShift, you must configure the necessary nodes and the entitlements for the environment to fetch images from the Red Hat Ecosystem Catalog . Perform the following steps to configure the nodes and entitlements:

Procedure

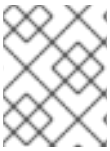
1. Install Red Hat Enterprise Linux (RHEL) on each of your nodes.
2. Register your nodes with Red Hat using the Red Hat Subscription Manager (RHSM), via the interface or the command line.
3. Attach your nodes to your 3scale subscription using RHSM.
4. Install OpenShift on your nodes, complying with the 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-rpms** repository using the subscription manager:

```
sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2-rpms
```

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

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

2.3. DEPLOYING 3SCALE ON OPENSHIFT USING A TEMPLATE



NOTE

OpenShift Container Platform (OCP) 4.x supports deployment of 3scale using the operator only. See Deploying 3scale using the operator .

Prerequisites

- An OpenShift cluster configured as specified in the Configuring nodes and entitlements section.
- A domain that resolves to your OpenShift cluster.
- Access to the Red Hat Ecosystem Catalog .
- (Optional) A working SMTP server for email functionality.

NOTE

Deploying 3scale on OpenShift using a template is based on OpenShift Container Platform 3.11

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

- Configuring registry authentication in OpenShift
- Importing the 3scale template
- Getting the Admin Portal URL
- Configuring SMTP Variables (Optional)

2.3.1. Configuring registry authentication in OpenShift

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

Procedure

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
```

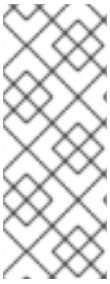
3. Create a **docker-registry** secret using the credentials you created in Creating a Registry Service Account.

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"
```

2.3.2. Importing the 3scale template



NOTE

- Wildcard routes have been removed as of 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:

Procedure

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 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.

2.3.3. 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 3scale 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}

2.3.4. 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:

Procedure

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

```
oc login
```

   a. Using the **oc patch** command, specify the **secret** type where **system-smtp** is the name of the secret, followed by the **-p** option, and write the 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)

Variable	Description
openssl.verify.mode	Specify how OpenSSL checks certificates when using TLS. Allowed values: **none** or **peer**.

Example

```
oc patch secret system-smtp -p '{"stringData":{"address":"<your_address>"}}'
oc patch secret system-smtp -p '{"stringData":{"username":"<your_username>"}}'
oc patch secret system-smtp -p '{"stringData":{"password":"<your_password>"}}'
```

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

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

3. Check the status of the rollout to ensure it has finished:

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

2.4. PARAMETERS OF THE 3SCALE TEMPLATE

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

Table 2.1. Template parameters

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

Name	Description	Default Value	Required?
AMP_RELEASE	3scale release tag.	**2.8.0**	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
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

Name	Description	Default Value	Required?
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-5-rhel7:5.0**	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-amp2/memcached-rhel7:3scale2.9**	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-amp2/system-rhel7:3scale2.9**	yes
AMP_BACKEND_IMAGE	3scale Backend image to use	**registry.redhat.io/3scale-amp2/backend-rhel7:3scale2.9**	yes
AMP_APICAST_IMAGE	3scale APIcast image to use	**registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9**	yes
AMP_ZYNC_IMAGE	3scale Zync image to use	**registry.redhat.io/3scale-amp2/zync-rhel7:3scale2.9**	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

Name	Description	Default Value	Required?
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
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

2.5. 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.

- Deploying APIcast templates on an existing OpenShift cluster containing 3scale
- Connecting APIcast from a different OpenShift cluster
- Changing the default behavior for embedded APIcast
- Connecting multiple APIcast deployments on a single OpenShift cluster over internal service routes
- Connecting APIcast on other deployments

2.5.1. 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:

Procedure

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 Configuring nodes and entitlements.

2.5.2. 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:

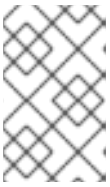
Procedure

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
```

2.5.3. 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.

2.5.4. 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 Software-Defined Networking (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 SDN plugin, perform the following steps to connect over internal routes:

Procedure

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 route backend-listener
```

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:

Procedure

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

```
oc login
```

2. As administrator, 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 route backend-listener
```

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

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

Additional resources

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

2.5.5. 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.

Additional resources

For more details, see Deploying APIcast on the Docker containerized environment.

2.6. DEPLOYING 3SCALE USING THE OPERATOR

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

NOTE

- Wildcard routes have been removed since 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.

Prerequisites

- Section 2.3.1, “Configuring registry authentication in OpenShift”
- Deploying 3scale using the operator first requires that you follow the steps in Chapter 7, *Installing the 3scale Operator on OpenShift*
- OpenShift Container Platform 4
  - A user account with administrator privileges in the OpenShift cluster.
  - **Note:** OCP 4 supports deployment of 3scale using the operator only.
  - For more information about supported configurations, see the Red Hat 3scale API Management Supported Configurations page.

Follow these procedures to deploy 3scale using the operator:

- Deploying the APIManager custom resource
- Getting the Admin Portal URL
- Getting the APIManager administrator credentials

2.6.1. 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 **Operators > 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

Use the *wildcardDomain* parameter with a name of your choice that resolves to an IP address with a valid DNS domain. Remove the placeholder marks when you are adding your parameter: < >.

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

2.6.2. 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 3scale 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}**

2.6.3. 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_USER | 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.

Additional resources

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

2.6.4. 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 3scale 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}**

2.7. DEPLOYMENT CONFIGURATION OPTIONS FOR 3SCALE ON OPENSHIFT USING THE OPERATOR

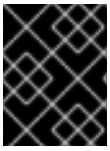
This section provides information about the deployment configuration options for Red Hat 3scale API Management on OpenShift using the operator.

Prerequisites

- Section 2.3.1, “Configuring registry authentication in OpenShift”
- Deploying 3scale using the operator first requires that you follow the steps in Chapter 7, *Installing the 3scale Operator on OpenShift*
- OpenShift Container Platform 4.x
  - A user account with administrator privileges in the OpenShift cluster.

2.7.1. Proof of concept for evaluation deployment

The following sections describe the configuration options applicable to the proof of concept for an evaluation deployment of 3scale. This deployment uses internal databases as default.



IMPORTANT

The configuration for external databases is the standard deployment option for production environments.

2.7.1.1. Default deployment configuration

- Containers will have Kubernetes resource limits and requests .
  - This ensures a minimum performance level.
  - It limits resources to allow external services and allocation of solutions.
- Deployment of internal databases.

- File storage will be based on Persistence Volumes (PV).
  - One will require read, write, execute (RWX) access mode.
  - OpenShift configured to provide them upon request.
- Deploy MySQL as the internal relational database.

The default configuration option is suitable for proof of concept (PoC) or evaluation by a customer.

One, many, or all of the default configuration options can be overriden with specific field values in the *APIManager* custom resource. The 3scale operator allows all available combinations whereas templates allow fixed deployment profiles. For example, the 3scale operator allows deployment of 3scale in evaluation mode and external databases mode. Templates do not allow this specific deployment configuration. Templates are only available for the most common configuration options.

2.7.1.2. Evaluation installation

For and evaluation installtion, containers will not have kubernetes resource limits and requests specified. For example:

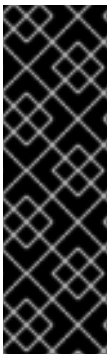
- Small memory footprint
- Fast startup
- Runnable on laptop
- Suitable for presale/sales demos

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  wildcardDomain: lvh.me
  resourceRequirementsEnabled: false
```

Check *APIManager* custom resource for reference.

2.7.2. External databases installation

An external databases installation is suitable for production use where high availability (HA) is a requirement or where you plan to reuse your own databases.



IMPORTANT

When enabling the 3scale external databases installation mode, all of the following databases are externalized:

- **backend-redis**
- **system-redis**
- **system-database** (**mysql** or **postgresql**)

3scale 2.8 and above has been tested is supported with the following database versions:

Database	Version
Redis	5.0
MySQL	5.7
PostgreSQL	10.6

Before creating *APIManager custom resource* to deploy 3scale, you must provide the following connection settings for the external databases using OpenShift secrets.

2.7.2.1. Backend Redis secret

Deploy two external Redis instances and fill in the connection settings as shown in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-redis
stringData:
  REDIS_STORAGE_URL: "redis://backend-redis-storage"
  REDIS_STORAGE_SENTINEL_HOSTS: "redis://sentinel-0.example.com:26379,redis://sentinel-
1.example.com:26379, redis://sentinel-2.example.com:26379"
  REDIS_STORAGE_SENTINEL_ROLE: "master"
  REDIS_QUEUES_URL: "redis://backend-redis-queues"
  REDIS_QUEUES_SENTINEL_HOSTS: "redis://sentinel-0.example.com:26379,redis://sentinel-
1.example.com:26379, redis://sentinel-2.example.com:26379"
  REDIS_QUEUES_SENTINEL_ROLE: "master"
type: Opaque
```

The *Secret* name must be **backend-redis**.

See Backend redis secret for reference.

2.7.2.2. System Redis secret

Deploy two external Redis instances and fill in the connection settings as shown in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: system-redis
stringData:
  URL: "redis://system-redis"
  SENTINEL_HOSTS: "redis://sentinel-0.example.com:26379,redis://sentinel-1.example.com:26379,
redis://sentinel-2.example.com:26379"
  SENTINEL_ROLE: "master"
  NAMESPACE: ""
```

```
  MESSAGE_BUS_URL: "redis://system-redis-messagebus"
  MESSAGE_BUS_SENTINEL_HOSTS: "redis://sentinel-0.example.com:26379,redis://sentinel-
1.example.com:26379, redis://sentinel-2.example.com:26379"
  MESSAGE_BUS_SENTINEL_ROLE: "master"
  MESSAGE_BUS_NAMESPACE: ""
type: Opaque
```

The *Secret* name must be **system-redis**.

See System redis secret for reference.

2.7.2.3. System database secret

Deploy a system database secret MySQL or PostgreSQL database instance and fill in the connection settings as shown in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: system-database
stringData:
  URL: "mysql2://root:password0@system-mysql/system"
  DB_USER: "mysql"
  DB_PASSWORD: "password1"
type: Opaque
```

The *Secret* name must be **system-database**.

See System database secret for reference.

2.7.2.4. *APIManager* custom resource

Lastly, create *APIManager* custom resource to deploy 3scale.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  wildcardDomain: lvh.me
  highAvailability:
    enabled: true
```

Check *APIManager* HighAvailabilitySpec for reference.

2.7.3. Amazon Simple Storage Service 3scale *Filestorage* installation

The following examples show 3scale *FileStorage* using Amazon Simple Storage Service (Amazon S3) instead of persistent volume claim (PVC).

Before creating *APIManager* custom resource to deploy 3scale, connection settings for the S3 service needs to be provided using an openshift secret.

2.7.3.1. Amazon S3 secret

In the following example, *Secret* name can be anyone, as it will be referenced in the *APIManager* custom resource.

```
kind: Secret
metadata:
  creationTimestamp: null
  name: aws-auth
stringData:
  AWS_ACCESS_KEY_ID: 123456
  AWS_SECRET_ACCESS_KEY: 98765544
  AWS_BUCKET: mybucket.example.com
  AWS_REGION: eu-west-1
type: Opaque
```

Lastly, create the *APIManager* custom resource to deploy 3scale.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
  spec:
    wildcardDomain: lvh.me
    system:
      fileStorage:
        simpleStorageService:
          configurationSecretRef:
            name: aws-auth
```

NOTE

Amazon S3 region and Amazon S3 bucket settings are provided directly in the *APIManager* custom resource. The Amazon S3 secret name is provided directly in the *APIManager* custom resource.

Check *APIManager SystemS3Spec* for reference.

2.7.4. PostgreSQL installation

A MySQL internal relational database is the default deployment. This deployment configuration can be overriden to use PostgreSQL instead.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  wildcardDomain: lvh.me
  system:
    database:
      postgresql: {}
```

Check *APIManager DatabaseSpec* for reference.

2.7.5. Reconciliation

Once 3scale has been installed, the 3scale operator enables updating a given set of parameters from the custom resource to modify system configuration options. Modifications are made by *hot swapping*, that is, without stopping or shutting down the system.

Not all the parameters of the *APIManager* custom resource definitions (CRDs) are reconcilable.

The following is a list of reconcilable parameters:

- Section 2.7.5.1, “Resources”
- Section 2.7.5.2, “Backend replicas”
- Section 2.7.5.3, “APIcast replicas”
- Section 2.7.5.4, “System replicas”

2.7.5.1. Resources

Resource limits and requests for all 3scale components.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  ResourceRequirementsEnabled: true/false
```

2.7.5.2. Backend replicas

Backend components pod count.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  backend:
    listenerSpec:
      replicas: X
    workerSpec:
      replicas: Y
    cronSpec:
      replicas: Z
```

2.7.5.3. APIcast replicas

APIcast staging and production components pod count.

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
```

```
spec:
  apicast:
    productionSpec:
      replicas: X
    stagingSpec:
      replicas: Z
```

2.7.5.4. System replicas

System app and system *sidekiq* components pod count

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  system:
    appSpec:
      replicas: X
    sidekiqSpec:
      replicas: Z
```

2.8. TROUBLESHOOTING COMMON 3SCALE INSTALLATION ISSUES

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

- Previous deployment leaving dirty persistent volume claims
- Incorrectly pulling from the Docker registry
- Permission issues for MySQL when persistent volumes are mounted locally
- Unable to upload logo or images
- Test calls not working on OpenShift
- APIcast on a different project from 3scale failing to deploy

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

Procedure

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 Ecosystem Catalog.

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

Solution

Procedure

Use the appropriate version of the Docker containerized environment.

2.8.3. Permission issues for MySQL when persistent volumes are mounted locally

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

Procedure

1. The directories used for the persistent volumes MUST have the write permissions for the root group. Having read-write 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

Procedure

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

2.8.5. Test calls not working 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

Procedure

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

2.8.6. APIcast on a different project from 3scale failing to deploy

Problem

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

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

You see the following error 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

Procedure

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 Red Hat 3scale API Management 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.

Prerequisites

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

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

To install APIcast, perform the steps outlined in the following sections:

- Section 3.1, "APIcast deployment options"
- Section 3.2, "APIcast environments"
- Section 3.3, "Configuring the integration settings"
- Section 3.4, "Configuring your service"
- Section 3.5, "Installing the APIcast operator"
- Section 3.6, "Deploying an APIcast gateway self-managed solution using the operator"
- Section 3.7, "WebSocket protocol support for APIcast"
- Section 3.8, "HTTP/2 in the APIcast gateway"

3.1. APICAST 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. Here are a few recommended options to deploy APIcast:
  - Deploying APIcast on 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.
  - ** Running APIcast on Red Hat OpenShift : Run APIcast on a supported version of OpenShift. You can connect self-managed APIcasts to a 3scale On-premises installation or to a 3scale Hosted (SaaS) account.

3.2. APICAST 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.

3.3. CONFIGURING THE INTEGRATION SETTINGS

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

On the Configuration page you will see the *Integration settings*.

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.

3.4. CONFIGURING YOUR SERVICE

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

This section will guide you through configuring your service:

- Declaring the API backend
- Configuring the authentication settings
- Configuring the API test call

3.4.1. Declaring the API backend

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, you 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**.

Procedure

- 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"
  }
```


3.4.2. Configuring the authentication settings

You can configure authentication settings for your API in the **AUTHENTICATION SETTINGS** section.

The following fields are all optional:

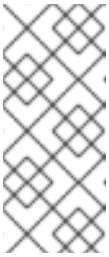
Field	Description
Host Header	Define a custom Host request header. This is required if your API backend only accepts traffic from a specific host.
Secret Token	Used to block direct developer requests to your API backend. Set the value of the header here, and ensure your backend only allows calls with 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

Field	Description
Errors	Define the response code, content type, and response body, for the following errors: authentication failed, authentication missing, no match.

3.4.3. Configuring the API test call

Procedure

1. Configure the test call for the hosted staging environment.
2. Enter a path existing in your API in the **API test GET request field** (for example, **/v1/word/good.json**).
3. Save the settings by clicking on the **Update Product** button in the bottom right part of the page.
    a. 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.



NOTE

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.

4. 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.

3.5. INSTALLING THE APICAST OPERATOR

This guide provides steps for installing the APIcast operator through the OpenShift Container Platform (OCP) console.

Procedure

1. Log in to the OCP console using an account with administrator privileges.
2. Create new project **operator-test** in **Projects > Create Project**
3. Click **Operators > Installed Operators**
4. Type *apicast* in the *Filter by keyword* box to find the APIcast operator. Do not use the community version.
5. Click the APIcast operator. You will see information about the APIcast operator.
6. Click *Install*. The *Create Operator Subscription* page opens.
7. Click *Subscribe* to accept all of the default selections on the *Create Operator Subscription* page.
    a. The subscription upgrade status is shown as *Up to date*.
8. Click **Operators > Installed Operators** to verify that the APIcast operator *ClusterServiceVersion* (CSV) status displays to *InstallSucceeded* in the **operator-test** project.

3.6. DEPLOYING AN APICAST GATEWAY SELF-MANAGED SOLUTION USING THE OPERATOR

This guide provides steps for deploying an APIcast gateway self-managed solution using the APIcast operator via the Openshift Container Platform console.

Prerequisites

- OpenShift Container Platform (OCP) 4.x or later with administrator privileges.
- You must first follow the steps in Installing the APIcast operator .

Procedure

1. Log in to the OCP console using an account with administrator privileges.
2. Click **Operators > Installed Operators**.
3. Click the *APIcast Operator* from the list of *Installed Operators*.
4. Click the **APIcast > Create APIcast**

3.6.1. APICast deployment and configuration options

You can deploy and configure an APIcast gateway self-managed solution using two approaches:

- Providing a 3scale system endpoint
- Providing a configuration secret

3.6.1.1. Providing a 3scale system endpoint

Procedure

1. Create an OpenShift secret that contains 3scale System Admin Portal endpoint information:

```
oc create secret generic ${SOME_SECRET_NAME} --from-
literal=AdminPortalURL=${MY_3SCALE_URL}
```

   - **${SOME_SECRET_NAME}** is the name of the secret and can be any name you want as long as it does not conflict with an existing secret.
   - **${MY_3SCALE_URL}** is the URI that includes your 3scale access token and 3scale System portal endpoint. For more details, see **THREESCALE_PORTAL_ENDPOINT**

     Example

```
oc create secret generic 3scaleportal --from-literal=AdminPortalURL=https://access-
token@account-admin.3scale.net
```

     For more information about the contents of the secret see the Admin portal configuration secret reference.

2. Create the OpenShift object for APIcast

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
spec:
  adminPortalCredentialsRef:
    name: SOME_SECRET_NAME
```

   The **spec.adminPortalCredentialsRef.name** must be the name of the existing OpenShift secret that contains the 3scale system Admin Portal endpoint information.

3. Verify the APIcast pod is running and ready, by confirming that the **readyReplicas** field of the OpenShift Deployment associated with the APIcast object is *1*. Alternatively, wait until the field is set with:

```
$ echo $(oc get deployment apicast-example-apicast -o jsonpath='{.status.readyReplicas}')
1
```

3.6.1.1.1. Verifying the APIcast gateway is running and available

Procedure

1. Ensure the OpenShift Service APIcast is exposed to your local machine, and perform a test request. Do this by port-forwarding the APIcast OpenShift Service to **localhost:8080**:

```
oc port-forward svc/apicast-example-apicast 8080
```

2. Make a request to a configured 3scale service to verify a successful HTTP response. Use the domain name configured in **Staging Public Base URL** or **Production Public Base URL** settings of your service. For example:

```
$ curl 127.0.0.1:8080/test -H "Host: myhost.com"
```

3.6.1.1.2. Exposing APIcast externally via a Kubernetes Ingress

To expose APIcast externally via a Kubernetes Ingress, set and configure the **exposedHost** section. When the **host** field in the **exposedHost** section is set, this creates a Kubernetes Ingress object. The Kubernetes Ingress object can then be used by a previously installed and existing Kubernetes Ingress Controller to make APIcast accessible externally.

To learn what Ingress Controllers are available to make APIcast externally accessible and how they are configured see the Kubernetes Ingress Controllers documentation .

The following example to expose APIcast with the hostname **myhostname.com**:

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
spec:
  ...
  exposedHost:
    host: "myhostname.com"
  ...
```

The example creates a Kubernetes Ingress object on the port 80 using HTTP. When the APIcast deployment is in an OpenShift environment, the OpenShift default Ingress Controller will create a Route object using the Ingress object APIcast creates which allows external access to the APIcast installation.

You may also configure TLS for the **exposedHost** section. Details about the available fields in the following table:

Table 3.1. APIcastExposedHost reference table

json/yaml field	Type	Required	Default value	Description
host	string	Yes	N/A	Domain name being routed to the gateway
tls	[]extensions.IngressTLS	No	N/A	Array of ingress TLS objects. See more on TLS.

3.6.1.2. Providing a configuration secret

Procedure

1. Create a secret with the configuration file:

```
$ curl
https://raw.githubusercontent.com/3scale/APIcast/master/examples/configuration/echo.json -
o $PWD/config.json

oc create secret generic apicast-echo-api-conf-secret --from-file=$PWD/config.json
```

The configuration file must be called **config.json**. This is an APIcast CRD reference requirement.

For more information about the contents of the secret see the Admin portal configuration secret reference.

2. Create an APIcast custom resource:

```
$ cat my-echo-apicast.yaml
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: my-echo-apicast
spec:
  exposedHost:
    host: YOUR DOMAIN
  embeddedConfigurationSecretRef:
    name: apicast-echo-api-conf-secret

$ oc apply -f my-echo-apicast.yaml
```

a. The following is an example of an embedded configuration secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: SOME_SECRET_NAME
type: Opaque
stringData:
  config.json: |
    {
      "services": [
        {
          "proxy": {
            "policy_chain": [
              { "name": "apicast.policy.upstream",
                "configuration": {
                  "rules": [{
                    "regex": "/",
                    "url": "http://echo-api.3scale.net"
                  }]
                }
              }
            ]
          }
        }
      ]
    }
```

3. Set the following content when creating the APIcast object:

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
spec:
  embeddedConfigurationSecretRef:
    name: SOME_SECRET_NAME
```

The **spec.embeddedConfigurationSecretRef.name** must be the name of the existing OpenShift secret that contains the configuration of the gateway.

4. Verify the APIcast pod is running and ready, by confirming that the **readyReplicas** field of the OpenShift Deployment associated with the APIcast object is *1*. Alternatively, wait until the field is set with:

```
$ echo $(oc get deployment apicast-example-apicast -o jsonpath='{.status.readyReplicas}')
1
```

3.6.1.2.1. Verifying APIcast gateway is running and available

Procedure

1. Ensure the OpenShift Service APIcast is exposed to your local machine, and perform a test request. Do this by port-forwarding the APIcast OpenShift Service to **localhost:8080**:

```
oc port-forward svc/apicast-example-apicast 8080
```

2. Make a request to a configured 3scale Service to verify a successful HTTP response. Use the domain name configured in **Staging Public Base URL** or **Production Public Base URL** settings of your service. For example:

```
$  curl 127.0.0.1:8080/test -H "Host: localhost"
{
  "method": "GET",
  "path": "/test",
  "args": "",
  "body": "",
  "headers": {
    "HTTP_VERSION": "HTTP/1.1",
    "HTTP_HOST": "echo-api.3scale.net",
    "HTTP_ACCEPT": "*/*",
    "HTTP_USER_AGENT": "curl/7.65.3",
    "HTTP_X_REAL_IP": "127.0.0.1",
    "HTTP_X_FORWARDED_FOR": ...
    "HTTP_X_FORWARDED_HOST": "echo-api.3scale.net",
    "HTTP_X_FORWARDED_PORT": "80",
    "HTTP_X_FORWARDED_PROTO": "http",
    "HTTP_FORWARDED": "for=10.0.101.216;host=echo-api.3scale.net;proto=http"
  },
  "uuid": "603ba118-8f2e-4991-98c0-a9edd061f0f0"
```

3.7. WEBSOCKET PROTOCOL SUPPORT FOR APICAST

Red Hat 3scale API Management provides support in the APIcast gateway for WebSocket protocol connections to backend APIs.

The following list are points to consider in if you are planning to implement WebSocket protocols:

- The WebSocket protocol does not support JSON Web Token (JWT).
- The WebSocket standard does not allow extra-headers.
- The WebSocket protocol is not part of the HTTP/2 standard.

3.7.1. WebSocket protocol support

The APIcast configuration policy chain is as follows:

```
"policy_chain": [
  { "name": "apicast.policy.websocket" },
  { "name": "apicast.policy.apicast" }
],
```

The API backend can be defined as **http[s]`** or **ws[s]**.

3.8. HTTP/2 IN THE APICAST GATEWAY

Red Hat 3scale API Management provides APIcast gateway support for HTTP/2 and Remote Procedure Calls (gRPC) connections. The HTTP/2 protocol controls enables data communication between APIcast and the API backend.

NOTE

- You cannot use **api_key** authorization. Use JSON Web Token (JWT) or Headers instead.
- gRPC endpoint terminates Transport Layer Security (TLS).
- The gRPC policy (HTTP/2) must be above the APIcast policy in the policy chain.

3.8.1. HTTP/2 protocol support

With HTTP/2 termination, APICast enabled HTTP/2 and backends can be HTTP/1.1 plaintext or TLS.

In HTTP/2 endpoints, where the policy is used, there are some constraints:

- The endpoint needs to listen on TLS in case this policy does not work expected.
- gRPC full flow will only work if the TLS policy is enabled.

The APIcast configuration policy chain is as follows:

```
"policy_chain": [
  { "name": "apicast.policy.grpc" },
  { "name": "apicast.policy.apicast" }
```

```
],
```

3.9. ADDITIONAL RESOURCES

To get information about the latest released and supported version of APIcast, see the articles:

- Red Hat 3scale API Management Supported Configurations
- Red Hat 3scale API Management - Component Details .

CHAPTER 4. RUNNING APICAST ON RED HAT OPENSHIFT

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

Prerequisites

- You must configure APIcast in your Red Hat 3scale API Management Admin Portal as per Chapter 3, *Installing APIcast*.
- 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.

To run APIcast on Red Hat OpenShift, perform the steps outlined in the following sections:

- Section 4.1, "Setting up Red Hat OpenShift"
- Section 4.2, "Deploying APIcast using the OpenShift template"
- Section 4.3, "Creating routes via the OpenShift console"

4.1. SETTING UP RED HAT OPENSHIFT

If you already have a running OpenShift cluster, you can skip this section. 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

Use the following section to set up Red Hat OpenShift:

- Installing the Docker containerized environment
- Starting the OpenShift cluster
- Setting up the OpenShift cluster on a remote server (Optional)

4.1.1. Installing 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:

Procedure

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
```

6. Verify that the container service is running with the following command:

```
sudo systemctl status docker
```

4.1.2. Starting the OpenShift cluster

To start the OpenShift cluster, do the following:

Procedure

1. 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

The docker command runs as the **root** user, so you will need to run any **oc** or docker commands with root privileges.

2. 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
```

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

4.1.3. Setting up the OpenShift cluster on a remote server (Optional)

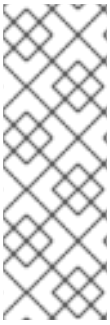
If you are deploying the OpenShift cluster on a remote server, you will need to explicitly specify a public hostname and a routing suffix on starting the cluster, so that you will be able to access the OpenShift web console remotely.

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.

4.2. DEPLOYING APICAST USING THE OPENSHIFT TEMPLATE



NOTE

- You can only deploy APIcast on OpenShift Container Platform (OCP) 3.11 when using templates.
- Operator-based installations are only supported on OCP version 4.1 and 4.2.
- For more information about supported configurations, see the Red Hat 3scale API Management Supported Configurations page.

Use the following to deploy APIcast unsing the OpenShift template:

Procedure

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 for the 3scale account, 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/2.8.0.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.
```

4.3. CREATING ROUTES VIA THE OPENSHIFT CONSOLE

To create routes via the OpenShift console, do the following:

Procedure

1. Open the web console for your OpenShift cluster in your browser: https://OPENSHIFT-SERVER-IP:8443/console/

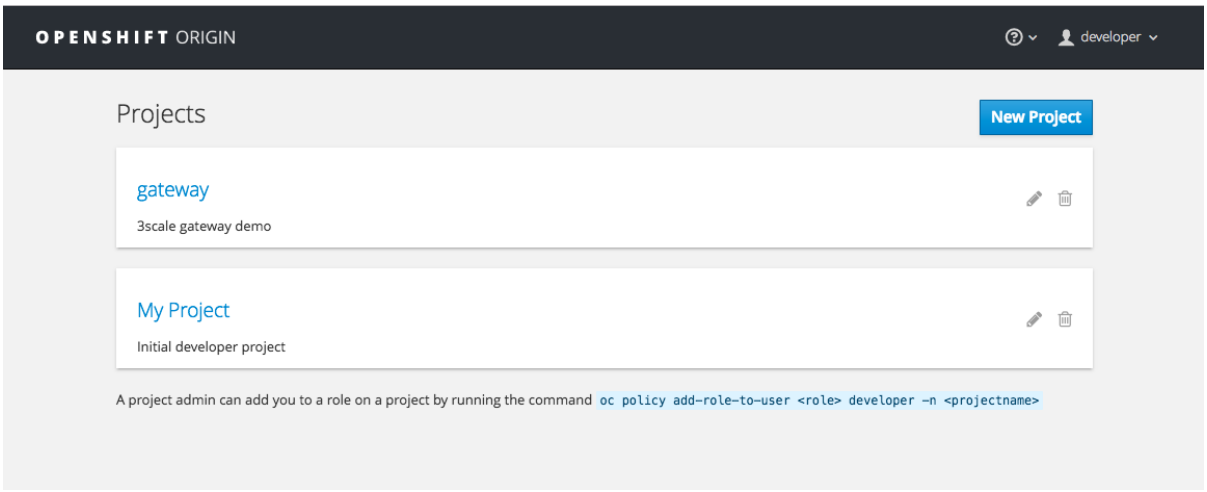
   Use the value specified in **--public-hostname** instead of **OPENSHIFT-SERVER-IP** if you started OpenShift cluster on a remote server.

   You will see the login screen for OpenShift.

NOTE

You may receive a warning about an untrusted website. 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.

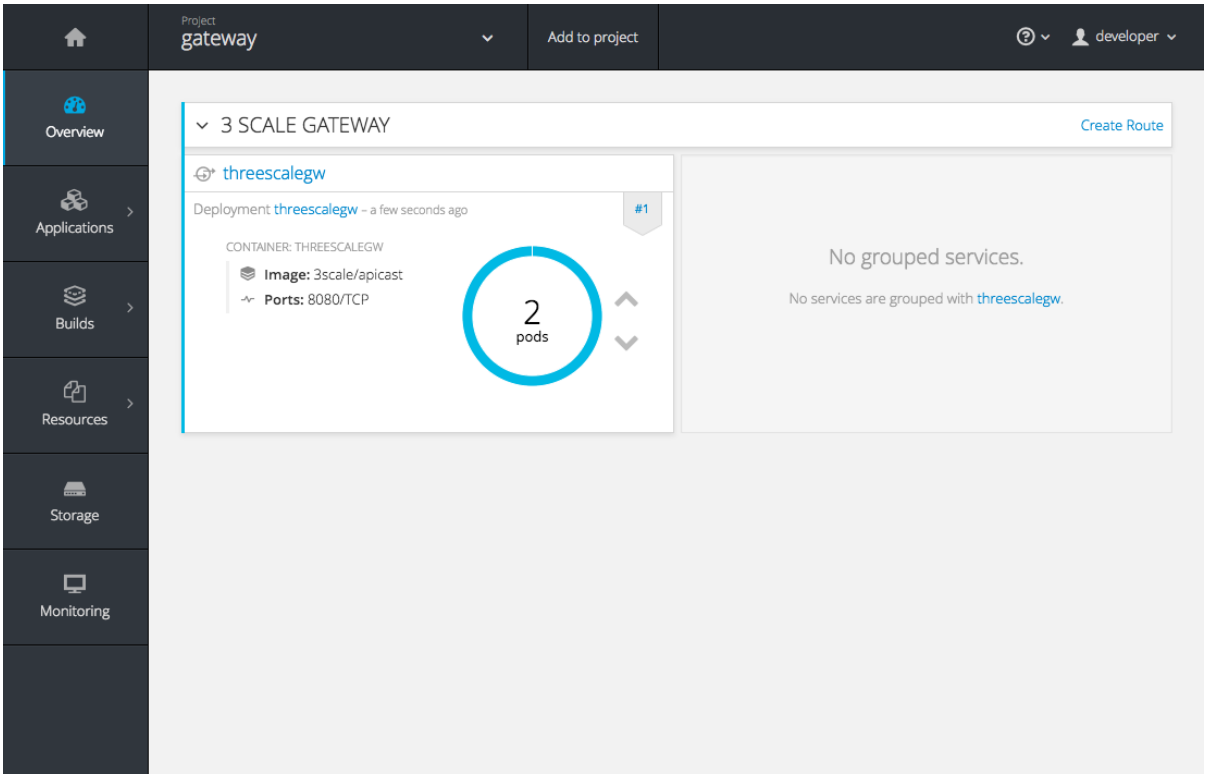
2. Log in using the *developer* credentials created or obtained in the Setting up Red Hat OpenShift section.
   You will see a list of projects, including the *gateway* project you created from the command line above.



   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.

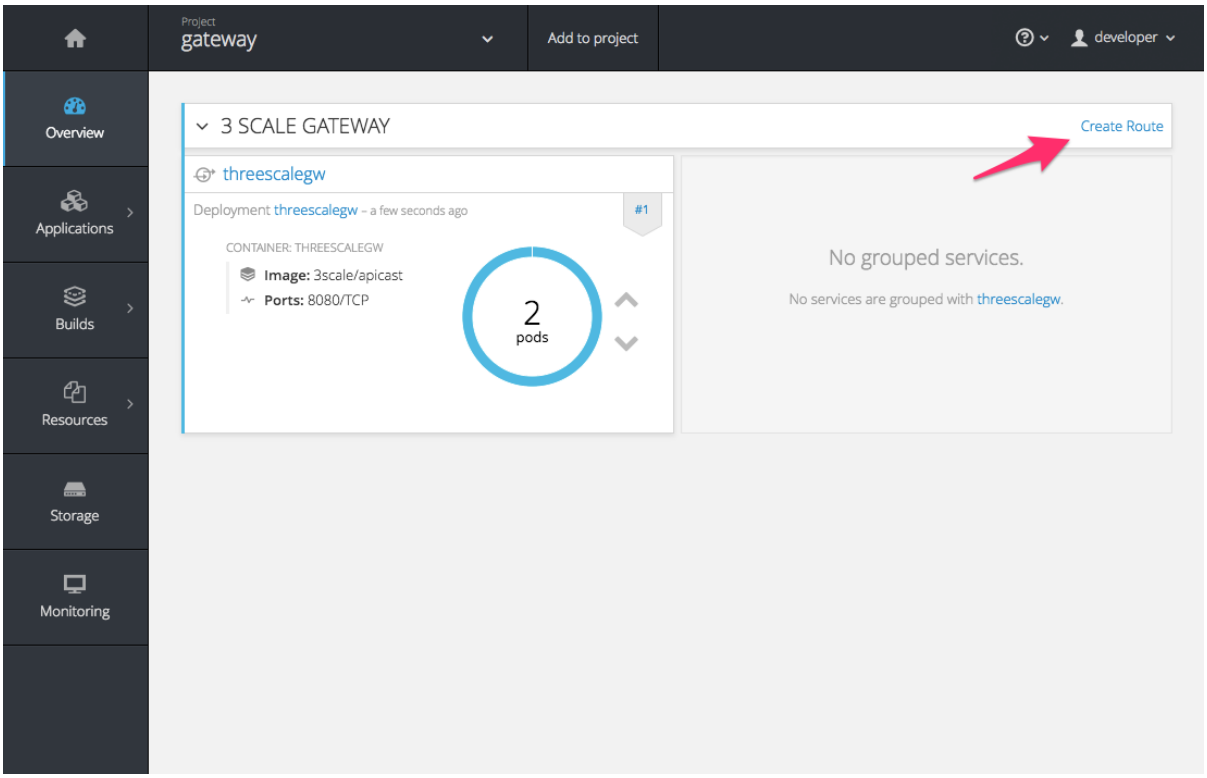
   When the build completes, the user interface (UI) will refresh and show two instances of APIcast (*2 pods*) that have been started by OpenShift, as defined in the template.



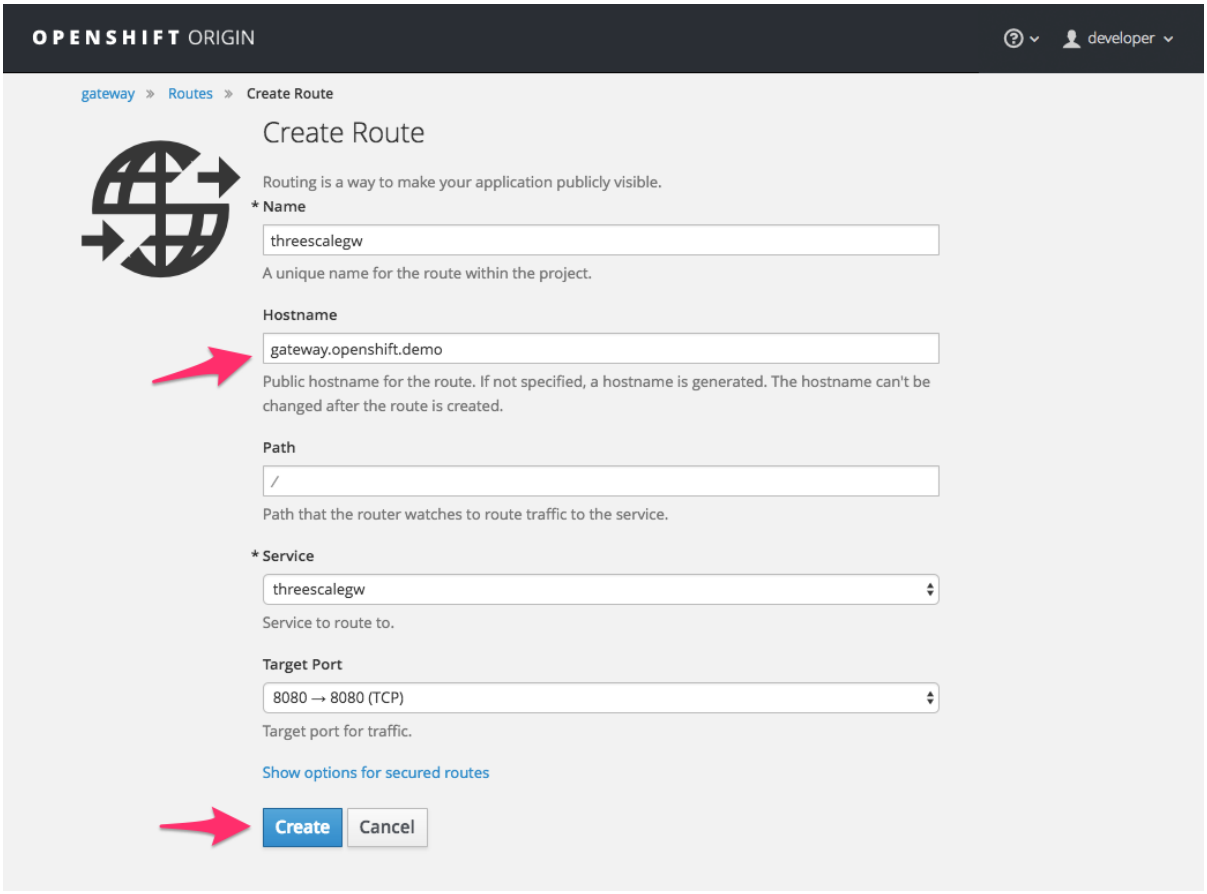
Each APIcast instance, upon starting, downloads the required configuration from 3scale using the settings you provided on the **Integration** page of your 3scale Admin Portal.

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**.



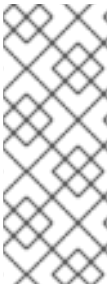
Enter the same host you set in 3scale above in the section *Public Base URL* (without the *http://* and without the port) , e.g. **gateway.openshift.demo**, then click the **Create** button.



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

CHAPTER 5. DEPLOYING APICAST ON THE DOCKER CONTAINERIZED ENVIRONMENT

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



NOTE

When deploying APIcast on the Docker containerized environment, the supported versions of Red Hat Enterprise Linux (RHEL) and Docker are as follows:

- RHEL 7.7
- Docker 1.13.1

Prerequisites

- You must configure APIcast in your 3scale Admin Portal as per Chapter 3, *Installing APIcast*.
- Access to the Red Hat Ecosystem Catalog .
  - To create a registry service account, see Creating and modifying registry service accounts .

To deploy APIcast on the docker containerized environment, perform the steps outlined in the following sections:

- Section 5.1, “Installing the Docker containerized environment”
- Section 5.2, “Running the Docker containerized environment gateway”

5.1. INSTALLING THE DOCKER CONTAINERIZED ENVIRONMENT

This guide covers the steps to set up the Docker containerized environment on RHEL 7.x.

The Docker container engine provided by Red Hat is 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* option. For details, see the RHEL product documentation .

To deploy RHEL 7.x on an Amazon Web Services (AWS), Amazon Elastic Compute Cloud (Amazon EC2) instance, take the following steps:

Procedure

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**.

Additional resources

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

5.2. RUNNING THE DOCKER CONTAINERIZED ENVIRONMENT GATEWAY

To run the docker containerized environment gateway, do the following:

Procedure

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 container engine image from the Red Hat registry:

+

```
sudo docker pull registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9
```

1. Run APIcast in a Docker container engine:

```
sudo docker run --name apicast --rm -p 8080:8080 -e
THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net
registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9
```

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 container engine called *"apicast"* on port **8080** and fetches the JSON configuration file from your 3scale Admin Portal. For other configuration options, see Installing APIcast.

5.2.1. 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

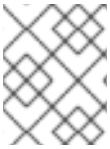
publish this port by adding **-p 8090:8090** to the **docker run** command.

- **-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**.

5.2.2. Testing APIcast

The preceding steps ensure that your Docker container engine is running with your own configuration file and the Docker container 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.

Additional resources

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

5.3. ADDITIONAL RESOURCES

- For more information about tested and supported configuration, see Red Hat 3scale API Management Supported Configurations

CHAPTER 6. DEPLOYING APICAST ON PODMAN

This is a step-by-step guide for deploying APIcast on a Pod Manager (Podman) container environment to be used as a Red Hat 3scale API Management API gateway.

NOTE

When deploying APIcast on a Podman container environment, the supported versions of Red Hat Enterprise Linux (RHEL) and Podman are as follows:

- RHEL 8.x
- Podman 1.4.2

Prerequisites

- You must configure APIcast in your 3scale Admin Portal as per Chapter 3, *Installing APIcast*.
- Access to the Red Hat Ecosystem Catalog .
  - To create a registry service account, see Creating and modifying registry service accounts .

To deploy APIcast on the Podman container environment, perform the steps outlined in the following sections:

- Section 6.1, "Installing the Podman container environment"
- Section 6.2, "Running the Podman environment"

6.1. INSTALLING THE PODMAN CONTAINER ENVIRONMENT

This guide covers the steps to set up the Podman container environment on RHEL 8.x. Docker is not included in RHEL 8.x, therefore, use Podman for working with containers.

For more details about Podman with RHEL 8.x, see the Container command-line reference.

Procedure

- Install the Podman container environment package:
  sudo dnf install podman

Additional resources

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

- Podman Installation Instructions

6.2. RUNNING THE PODMAN ENVIRONMENT

To run the Podman container environment, follow the procedure below.

Procedure

1. Download a ready to use Podman container image from the Red Hat registry:

```
podman pull registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9
```

2. Run APIcast in a Podman:

```
podman run --name apicast --rm -p 8080:8080 -e
THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net
registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9
```

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 Podman container engine called *"apicast"* on port **8080** and fetches the JSON configuration file from your 3scale Admin Portal. For other configuration options, see Installing APIcast.

6.2.1. Testing APIcast with Podman

The preceding steps ensure that your Podman container engine is running with your own configuration file and the Podman container 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.

6.3. THE PODMAN COMMAND OPTIONS

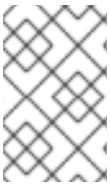
You can use the following option examples with the **podman** command:

- **-d**: Runs the container in *detached mode* 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 **podman attach** command, for example, **podman attach apicast**.
- **ps** and **-a**: Podman **ps** is used to list creating and running containers. Adding **-a** to the **ps** command will show all containers, both running and stopped, for example, **podman ps -a**.
- **inspect** and **-l**: Inspect a running container. For example, use **inspect** to see the ID that was assigned to the container. Use **-l** to get the details for the latest container, for example, **podman inspect -l | grep Id\":**.

6.4. ADDITIONAL RESOURCES

- For more information about tested and supported configurations, see Red Hat 3scale API Management Supported Configurations.
- For information about getting started with Podman, see Basic Setup and Use of Podman.

CHAPTER 7. INSTALLING THE 3SCALE OPERATOR ON OPENSHIFT



NOTE

3scale supports the last two general availability (GA) releases of OpenShift Container Platform (OCP). For more information, see the Red Hat 3scale API Management Supported Configurations page.

This documentation shows you how to:

- Create a new project.
- Deploy a Red Hat 3scale API Management 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 to a supported version of an OpenShift Container Platform 4 cluster using an account with administrator privileges.
  - For more information about supported configurations, see the Red Hat 3scale API Management Supported Configurations page.

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.

To install the 3scale operator on OpenShift, perform the steps outlined in the following sections:

- Section 7.1, “Creating a new OpenShift project”
- Section 7.2, “Installing and configuring the 3scale operator using the OLM”

7.1. CREATING A NEW OPENSHIFT PROJECT

This procedure explains how to create a new OpenShift project named **3scale-project**. Replace this project name with your own.

Procedure

To create a new OpenShift project:

- Indicate a valid name using alphanumeric characters and dashes. As an example, run the command below to create **3scale-project**:

```
oc new-project 3scale-project
```

This creates the new *OpenShift project* where the operator, the *APIManager* custom resource (CR), and the *Capabilities* custom resources will be installed. The operator manages the custom resources through OLM in that project.

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

Use Operator Lifecycle Manager (OLM) to install the 3scale operator on an OpenShift Container Platform (OCP) 4.3 cluster through the OperatorHub in the OCP console.

NOTE

- When using the OCP on a restricted network or a disconnected cluster, OLM can no longer use the OperatorHub. Follow the instructions for setting up and using the OLM in the guide titled Using Operator Lifecycle Manager on restricted networks.

Prerequisites

- You must install and deploy the 3scale operator in the project that you defined in Creating a new OpenShift project.

Procedure

1. In the OpenShift Container Platform console, log in using an account with administrator privileges.
2. The menu structure depends on the version of OpenShift you are using:
   - Click **Operators > 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. Confirm that the subscription **upgrade status** is shown as **Up to date**.

8. Verify that the 3scale operator ClusterServiceVersion (CSV) is displayed, and the **Status** of the operator ultimately resolves to **InstallSucceeded** in the project you defined in Creating a new OpenShift project:
   - Click **Operators > Installed Operators**. In this case, successful installation will register the *APIManager* CRD, and the CRDs related to the *Capabilities* functionality of the operator in the *OpenShift API server*.
9. 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
```

10. You should see the following output:

```
No resources found.
```

7.2.1. Allowing domains on restricted networks

IMPORTANT

When configuring the 3scale operator on restricted networks, database images are missing and will not be automatically mirrored. They must be mirrored manually. Refer to the OCP documentation guide on Using Operator Lifecycle Manager on restricted networks.

To manually mirror database images, do the following:

- Find the images you need to mirror.
- Use the **oc image mirror** command to mirror the images to the bastion registry.
- Create an **imageContentSourcePolicy** object in OCP to make the mirrored images recognized.

Procedure

1. Create a file with the following contents and name it **databases_images.txt**:

```
registry.redhat.io/rhscl/redis-32-
rhel7@sha256:a9bdf52384a222635efc0284db47d12fbde8c3d0fcb66517ba8eefad1d4e9dc9=
<BASTION_REGISTRY>/threescale-db/redis-32-rhel7

registry.redhat.io/rhscl/postgresql-10-
rhel7@sha256:de3ab628b403dc5eed986a7f392c34687bddafee7bdfccfd65cecf137ade3dfd=
<BASTION_REGISTRY>/threescale-db/postgresql-10-rhel7

registry.redhat.io/rhscl/mysql-57-
rhel7@sha256:9a781abe7581cc141e14a7e404ec34125b3e89c008b14f4e7b41e094fd3049fe=
<BASTION_REGISTRY>/threescale-db/mysql-57-rhel7
```

   The **BASTION_REGISTRY** is the address of your bastion registry concrete digest (**sha256:**), which may differ.

2. Get your bastion registry concrete digest by invoking bastion registry in the OCP project where you already deployed the 3scale operator using OLM:

```
oc get csv 3scale-operator.v0.6.0 -o yaml | grep 'redis-32-rhel7\|postgresql-10-rhel7\|mysql-57-rhel7'
```

3. Run the command below. In this case, **REG_CREDS_FILE** is the file containing Docker-like formatted registry credentials for **registry.redhat.io** and their bastion registry.

```
oc image mirror -f databases_images.txt -a <REG_CREDS_FILE>
```

4. Create the following file **additional-dbs-icspolicy.yml**:

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: additional-threescale-dbs
spec:
  repositoryDigestMirrors:
  - mirrors:
    - <bastion registry>/threescale-db/redis-32-rhel7
    source: registry.redhat.io/rhscl/redis-32-rhel7
  - mirrors:
    - <bastion registry>/threescale-db/postgresql-10-rhel7
    source: registry.redhat.io/rhscl/postgresql-10-rhel7
  - mirrors:
    - <bastion registry>/threescale-db/mysql-57-rhel7
    source: registry.redhat.io/rhscl/mysql-57-rhel7
```

5. Run the following command:

```
oc create -f additional-dbs-icspolicy.yml
```

   This command will restart all OCP cluster nodes one after another. Wait for all nodes to restart.

6. Run the following command to see the status of each node. Every node must be in ready state.

```
oc get nodes
```

When you are creating the **threescale-registry-auth** secret for the **apimanager** object, include the secret credentials for your bastion registry so that the 3scale operator pulls the mirrored images.

Continue with creating the **apimanager** object according to the Installing and configuring the 3scale operator using the OLM documentation.

Additionally, while using OCP on restricted networks, you will need to create a list of the allowed domains you intend to use in the 3scale Developer Portal. Consider the following examples:

- Any link you intend to add to the Developer Portal.
- SSO integrations through third party SSO providers such as GitHub.
- Billing.
- Webhooks that trigger an external URL.

Additional resources

- For troubleshooting information, see the OpenShift Container Platform documentation.
- For more information about using the OLM on retricted networks, see Using Operator Lifecycle Manager on restricted networks.
- For more information about preparing your installation on restricted networks, see Creating a mirror registry for installation in a restricted network.
- For more information about supported configurations, see the Red Hat 3scale API Management Supported Configurations page.

CHAPTER 8. 3SCALE HIGH AVAILABILITY AND EVALUATION TEMPLATES

This document describes the templates for High Availability and Evaluation used by Red Hat 3scale API Management 2.9 installation.

Prerequisites

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

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 .

To deploy High Availability and Evaluation templates, perform the steps outlined in the following sections:

- Section 8.1, “High Availability template”
- Section 8.2, “Evaluation template”

8.1. HIGH AVAILABILITY TEMPLATE

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

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.

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.

8.1.1. Setting RWX_STORAGE_CLASS for high availability

ReadWriteMany (RWX) PersistentVolumeClaims (PVCs) uses the storage class RWX_STORAGE_CLASS.

required: false

value: **null**

- Set this to **null** to signal OpenShift that you want the storage class to be auto-discovered (no value).
- If you set this to an empty string or no default value, it signals OpenShift that you want the string storage empty. This is an invalid setting.

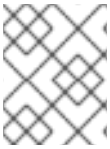
8.2. EVALUATION TEMPLATE

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 9. REDIS HIGH AVAILABILITY (HA) SUPPORT FOR 3SCALE

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 .



NOTE

The information and procedures in this section is not officially supported by Red Hat. It is for reference only.

The database components for HA in Red Hat 3scale API Management 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.

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

If the Redis pod comes to a stop, or if the OpenShift Container Platform stops it, a new pod is automatically created. Persistent storage will restore the data so the pod continues to work. In these scenarios, there will be a small amount of downtime while the new pod starts. This is due to a limitation in Redis that does not support a multi-master setup. You can reduce downtime by preinstalling the Redis images onto all nodes that have Redis deployed to them. This will speed up the pod restart time.

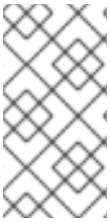
To set up Redis for zero downtime and configure back-end components for 3scale, perform the steps outlined in the following sections:

- Section 9.1, “Setting up Redis for zero downtime”
- Section 9.2, “Configuring back-end components for 3scale”

9.1. 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.

9.2. CONFIGURING BACK-END COMPONENTS FOR 3SCALE

There are configuration settings in 3scale 2.9 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 Redis with sentinels, 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.

9.2.1. Creating backend-redis and system-redis secrets

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

- Deploying a fresh installation of 3scale for HA
- Migrating a non-HA deployment of 3scale to HA

9.2.2. Deploying a fresh installation of 3scale for HA

1. Create the **backend-redis** and **system-redis** secrets with the 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 Deploying 3scale on OpenShift using a template , using the latest version of the templates.

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

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

1. Edit the **backend-redis** and **system-redis** secrets with all fields as shown in 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 using templates.

9.2.3.1. 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**.

9.2.3.2. 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.

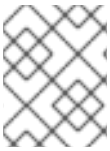
9.3. ADDITIONAL INFORMATION

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

CHAPTER 10. CONFIGURING AN EXTERNAL MYSQL DATABASE

This guide provides information for externalizing the MySQL database for Chapter 8, *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 8, *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.



NOTE

Red Hat supports 3scale configurations that use an external MySQL database. However, the database itself is not within the scope of support.

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, *Installing 3scale on OpenShift*.

To configure an external MySQL database for High Availability (HA), perform the steps outlined in the following sections:

- Section 10.1, "External MySQL database limitations"
- Section 10.2, "Externalizing the MySQL database"
- Section 10.3, "Rolling back"

10.1. 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*.

10.2. EXTERNALIZING THE MYSQL DATABASE

Use the following steps to fully externalize the MySQL database.

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.

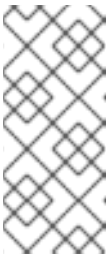
2. Follow the steps below in the order shown to scale down all the pods. This will avoid loss of data.

   ### 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 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. Send **system-mysql-dump.sql** to the remote database server and import the dump into it. Use the command to import it:

12. 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
```

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

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

14. 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.

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

16. 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.

10.3. 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 14.

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.

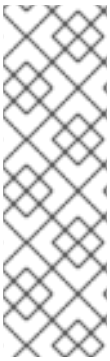
2. Scale down all the pods and then scale them back up again, including **system-mysql**. The **system-app** pod and the other pods to be started after it should be up and running again. Run the following command to confirm all pods are back up and running:

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

10.4. ADDITIONAL INFORMATION

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

CHAPTER 11. SETTING UP YOUR 3SCALE SYSTEM IMAGE WITH AN ORACLE DATABASE



NOTE

- The Oracle Database is only supported with OpenShift Container Platform (OCP) 3.11 when you are performing a template-based installation of 3scale.
- The Oracle Database is not supported with OCP version 4.2 and 4.3 when you are performing an operator-only installation of 3scale.
- For more information about supported configurations, see the Red Hat 3scale API Management Supported Configurations page.

This section explains how a Red Hat 3scale API Management administrator sets up the 3scale system image with an Oracle Database. By default, 3scale 2.9 has a component called system that stores configuration data in a MySQL database. You can override the default database and store your information in an external Oracle Database. Follow the steps in this chapter to build a custom system container image with your own Oracle Database client binaries and deploy 3scale to OpenShift.

Prerequisites

- Download the *basic-lite* or *basic* client, the *ODBC driver*, and the *SDK* for either Oracle Database 12c or Oracle Database 19c from the Instant Client Downloads for Linux x86-64 (64-bit) page.

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

Oracle 12c example packages for 3scale 2.9 and 2.9.1

- Oracle Instant Client Package - Basic: instantclient-basic-linux.x64-12.2.0.1.0.zip or instantclient-basiclite-linux.x64-12.2.0.1.0.zip
- Oracle Instant Client Package - SDK: instantclient-sdk-linux.x64-12.2.0.1.0.zip
- Oracle Instant Client Package - ODBC: instantclient-odbc-linux.x64-12.2.0.1.0-2.zip

Oracle 19c example packages for 3scale 2.9.1

- Oracle Instant Client Package - Basic or Basic Light: instantclient-basic-linux.x64-19.8.0.0.0dbru.zip or instantclient-basiclite-linux.x64-19.8.0.0.0dbru.zip
- Oracle Instant Client Package - SDK: instantclient-sdk-linux.x64-19.8.0.0.0dbru.zip
- Oracle Instant Client Package - ODBC: instantclient-odbc-linux.x64-19.8.0.0.0dbru.zip

To set up your 3scale system image with and Oracle Database, perform the steps outlined in the following sections:

- Section 11.1, “Preparing the Oracle Database”
- Section 11.2, “Building the system image”

11.1. PREPARING THE ORACLE DATABASE

This section provides steps for preparing your Oracle Database.

Prerequisites

- A supported version of the Oracle Database accessible from your OpenShift cluster.
- Access to the Oracle Database **system** user for installation procedures.

Procedure

1. Create a new database.
   To configure 3scale with Oracle Database, use the following settings:

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

2. Collect the database details.
   You need the following information for 3scale configuration:

   - The Oracle Database URL address.
   - The Oracle Database. service name
   - The 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.

11.2. BUILDING THE SYSTEM IMAGE

This section provides steps to build the system image.

Prerequisites

- You should have already carried out the steps in Preparing the Oracle Database.

Procedure

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

```
git clone --branch 2.9.1.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 2.9 *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 Preparing the 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 Preparing the Oracle Database:

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

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

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

9. 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.

11.2.1. Updating ImageChange triggers

Update the ImageChange triggers of the DeploymentConfigs that use the System image so they use the new Oracle-based System image.

Prerequisites

- First carry out the steps in Building the system image.

Procedure

1. Save the current 3scale release in an environment variable:

```
$ export THREESCALE_RELEASE=2.9
```

2. Update the **system-app** ImageChange trigger:

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

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

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

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

```
$ oc set triggers dc/system-sidekiq --from-image=amp-system:${THREESCALE_RELEASE} --containers=system-sidekiq,check-svc --remove

$ oc set triggers dc/system-sidekiq --from-image=amp-system:${THREESCALE_RELEASE}-oracle --containers=system-sidekiq,check-svc
```

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

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

```
$ oc set triggers dc/system-sphinx --from-image=amp-system:${THREESCALE_RELEASE} --containers=system-sphinx,system-master-svc --remove

$ oc set triggers dc/system-sphinx --from-image=amp-system:${THREESCALE_RELEASE}-oracle --containers=system-sphinx,system-master-svc
```

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

Additional resources

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