Red Hat 3scale 2-saas

Installing 3scale

Install and configure 3scale API Management.

Last Updated: 2019-05-15

Red Hat 3scale 2-saas 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 API Management

CHAPTER 1. INSTALLING APICAST

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

See the articles Red Hat 3scale API Management Supported Configurations and Red Hat 3scale API Management - Component Details to get information about the latest released and supported version of APIcast. For the updates on APIcast Hosted version please refer to Red Hat 3scale API Management Platform SaaS Release Notes. In this guide you will learn about deployment options, environments provided, and how to get started.

1.1. PREREQUISITES

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

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

1.2. DEPLOYMENT OPTIONS

You can use APIcast hosted or self-managed, in both cases, it needs connection to the rest of the 3scale API management platform:

- APIcast hosted: 3scale hosts APIcast in the cloud. In this case, APIcast is already deployed for you and it is limited to 50,000 calls per day.
- **APIcast self-managed**: You can deploy APIcast wherever you want. The self-managed mode is the intended mode of operation for production environments. Here are a few recommended options to deploy APIcast:
  - the Docker containerized environment: Download a ready to use Docker-formatted container image, which includes all of the dependencies to run APIcast in a Docker-formatted container.
  - OpenShift: Run APIcast on a supported version of OpenShift. You can connect self-managed APIcasts both to a 3scale installation or to a 3scale online account.

1.3. ENVIRONMENTS

By default, when you create a 3scale account or create a new API service, you get an APIcast **hosted** 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.
- **Production**: Limited to 50,000 calls per day and supports the following out-of-the-box authentication options: API key, and App ID and App key pair, OpenID Connect.

When you use Self-managed deployment, you still have the same two environments, and you need to deploy an APIcast instance for each. You can specify which configuration (Staging or Production) the APIcast instance will use by setting the environment variable **THREESCALE_DEPLOYMENT_ENV**, which can take values **staging** or **production**.

1.4. CONFIGURING THE INTEGRATION SETTINGS

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

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

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

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

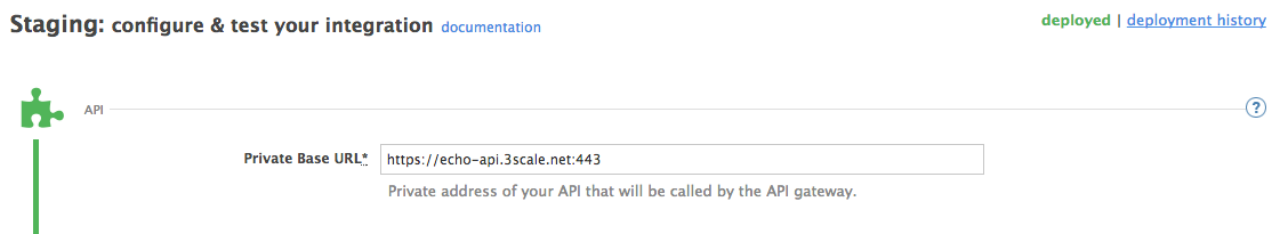
You can change these settings by clicking on **edit integration settings** in the top right corner.

1.5. CONFIGURING YOUR SERVICE

1.5.1. Declare the API backend

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

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



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"
}
```


1.5.2. Configure the authentication settings

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

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

1.5.3. Configure the API test call

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

1.5.4. Save the configuration settings

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

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

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

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

NOTE

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

Now you have your API integrated with 3scale.

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

CHAPTER 2. APICAST HOSTED

Once you complete this tutorial, you'll have your API fully protected by a secure gateway in the cloud.

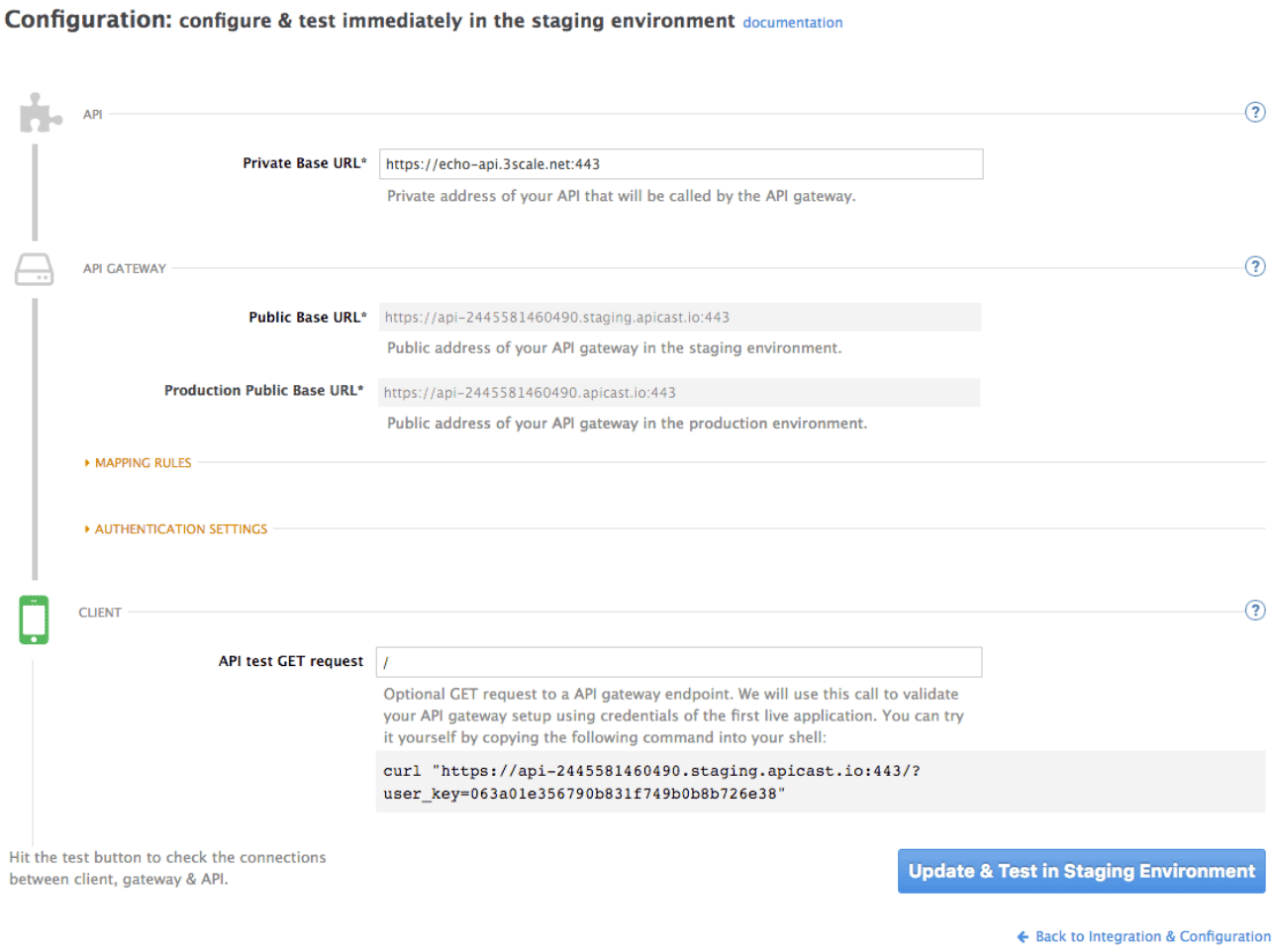
APIcast hosted is the best deployment option if you want to launch your API as fast as possible, or if you want to make the minimum infrastructure changes on your side.

2.1. PREREQUISITES

- You have reviewed the deployment alternatives and decided to use APIcast hosted to integrate your API with 3scale.
- Your API backend service is accessible over the public Internet (a secure communication will be established to prevent users from bypassing the access control gateway).
- You do not expect demand for your API to exceed the limit of 50,000 hits/day (beyond this, we recommend upgrading to the self-managed gateway).

2.2. STEP 1: DEPLOY YOUR API WITH APICAST HOSTED IN A STAGING ENVIRONMENT

The first step is to configure your API and test it in your staging environment. Define the private base URL and its endpoints, choose the placement of credentials and other configuration details that you can read about here. Once you're done entering your configuration, go ahead and click on Update & Test Staging Environment button to run a test call that will go through the APIcast staging instance to your API.



If everything was configured correctly, you should see a green confirmation message.

Before moving on to the next step, make sure that you have configured a secret token to be validated by your backend service. You can define the value for the secret token under **Authentication Settings**. This will ensure that nobody can bypass APIcast's access control.

2.3. STEP 2: DEPLOY YOUR API WITH THE APICAST HOSTED INTO PRODUCTION

At this point, you're ready to take your API configuration to a production environment. To deploy your 3scale-hosted APIcast instance, go back to the 'Integration and Configuration' page and click on the **'Promote to v.x to Production'** button. Repeat this step to promote further changes in your staging environment to your production environment.

It will take between 5 and 7 minutes for your configuration to deploy and propagate to all the cloud APIcast instances. During redeployment, your API will not experience any downtime. However, API calls may return different responses depending on which instance serves the call. You'll know it has been deployed once the box around your production environment has turned green.

Both the staging and production APIcast instances have base URLs on the apicast.io domain. You can easily tell them apart because the staging environment URLs have a staging subdomain. For example:

- staging: https://api-2445581448324.staging.apicast.io:443
- production: https://api-2445581448324.apicast.io:443

2.3.1. Bear in mind

- 50,000 hits/day is the maximum allowed for your API through the APIcast production cloud instance. You can check your API usage in the Analytics section of your Admin Portal.
- There is a hard throttle limit of 20 hits/second on any spike in API traffic.
- Above the throttle limit, APIcast returns a response code of **403**. This is the same as the default for an application over rate limits. If you want to differentiate the errors, please check the response body.

CHAPTER 3. RUNNING APICAST ON RED HAT OPENSHIFT

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

3.1. PREREQUISITES

To follow the tutorial steps below, you will first need to configure APIcast in your 3scale Admin Portal as per the APIcast Overview. 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.

3.2. SETTING UP RED HAT OPENSHIFT

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

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

In this tutorial the OpenShift cluster will be installed using:

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

3.2.1. Install the Docker containerized environment

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

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

1. List all repositories:

```
sudo yum repolist all
```

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

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

3. Install Docker-formatted container images:

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

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

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

5. Start the Docker service:

```
sudo systemctl start docker
```

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

```
sudo systemctl status docker
```

3.2.2. Start the OpenShift cluster

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

NOTE

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

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

```
oc cluster up
```

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

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

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

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

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

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

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

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

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

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

3.3. DEPLOYING APICAST USING THE OPENSHIFT TEMPLATE

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

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

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

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

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

   The response should look like this:

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

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

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

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

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

   The response should look like this:

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

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

   ```
   oc new-app -f https://raw.githubusercontent.com/3scale/3scale-amp-openshift-templates/2.5.0.GA/apicast-gateway/apicast.yml
   ```

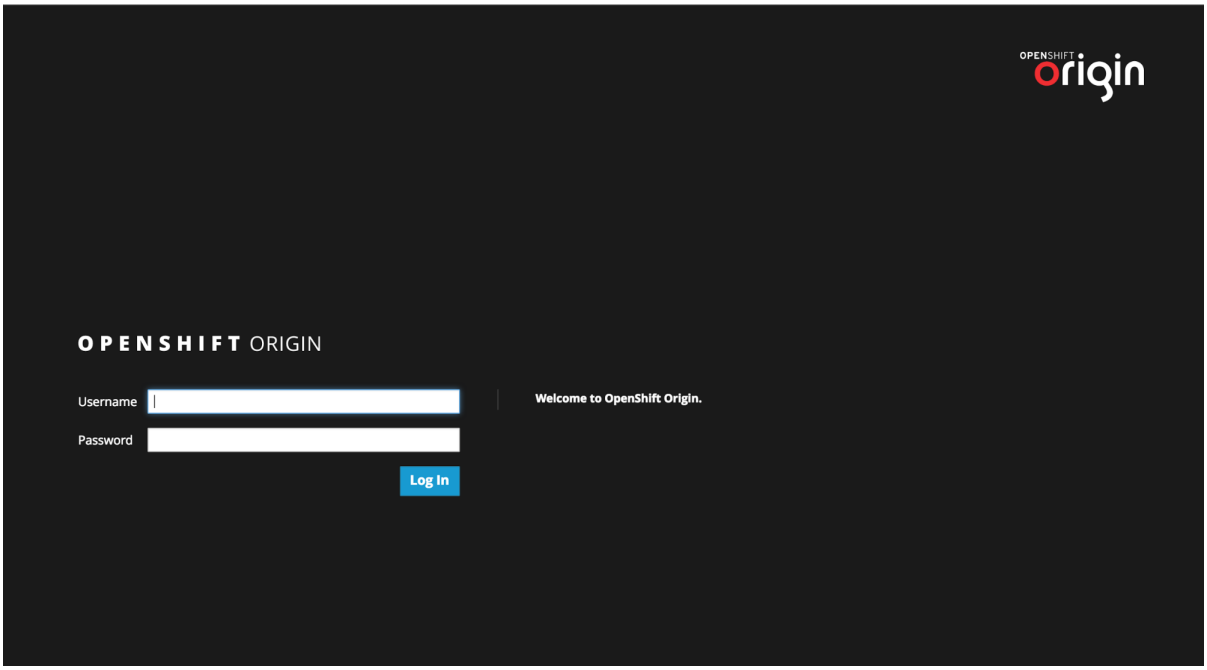
   You should see the following messages at the bottom of the output:

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

3.4. CREATING ROUTES VIA THE OPENSHIFT CONSOLE

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

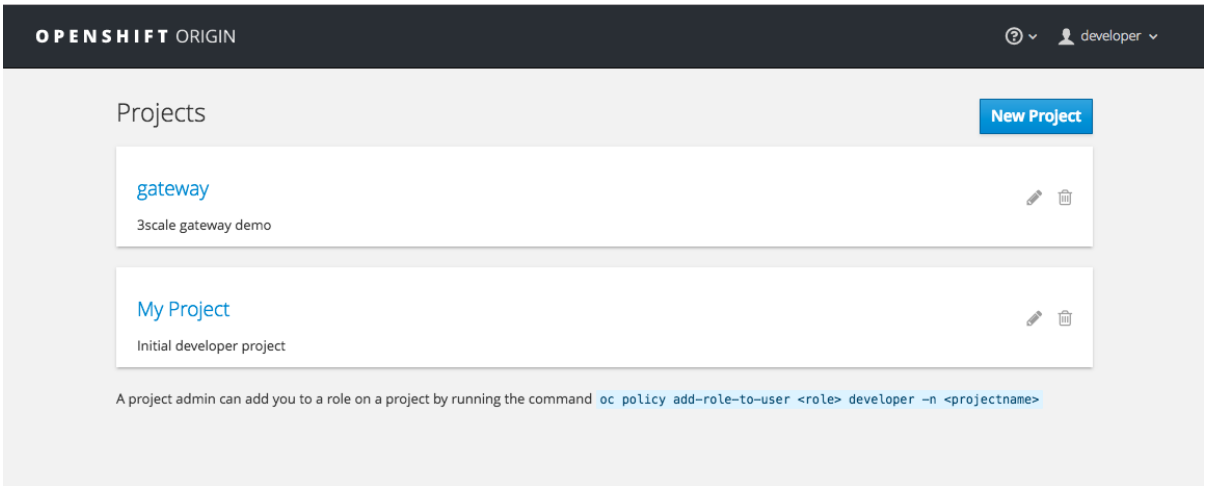
   You should see the login screen:



NOTE

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

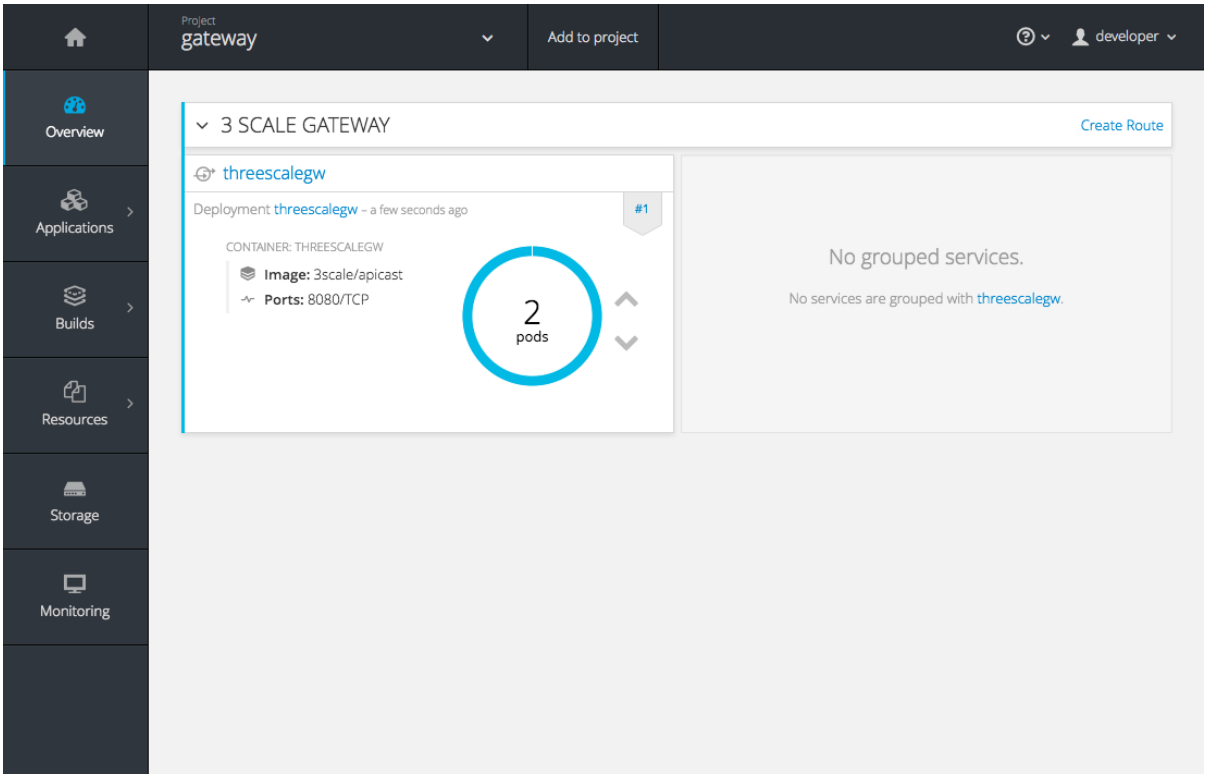
2. Log in using the *developer* credentials created or obtained in the *Setup OpenShift* section above.
   You will see a list of projects, including the *gateway* project you created from the command line above.



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

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

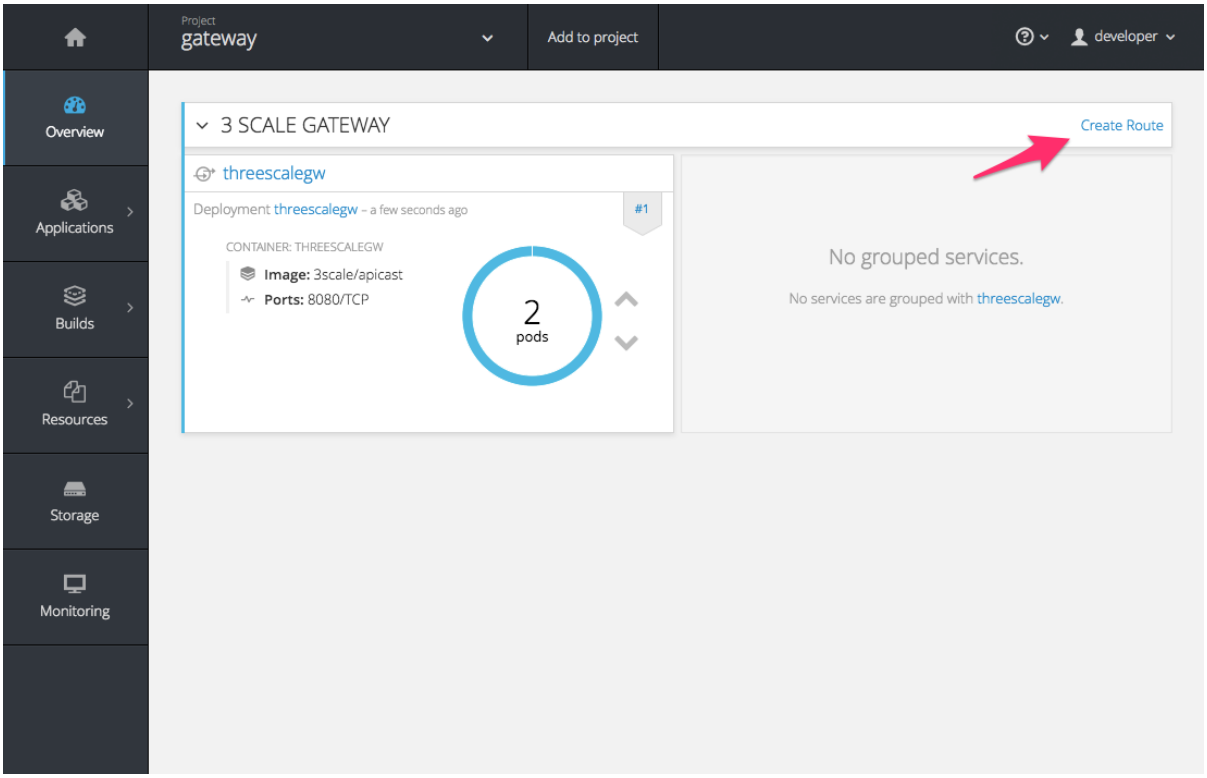
   When the build completes, the UI will refresh and show two instances of APIcast (*2 pods*) that have been started by OpenShift, as defined in the template.



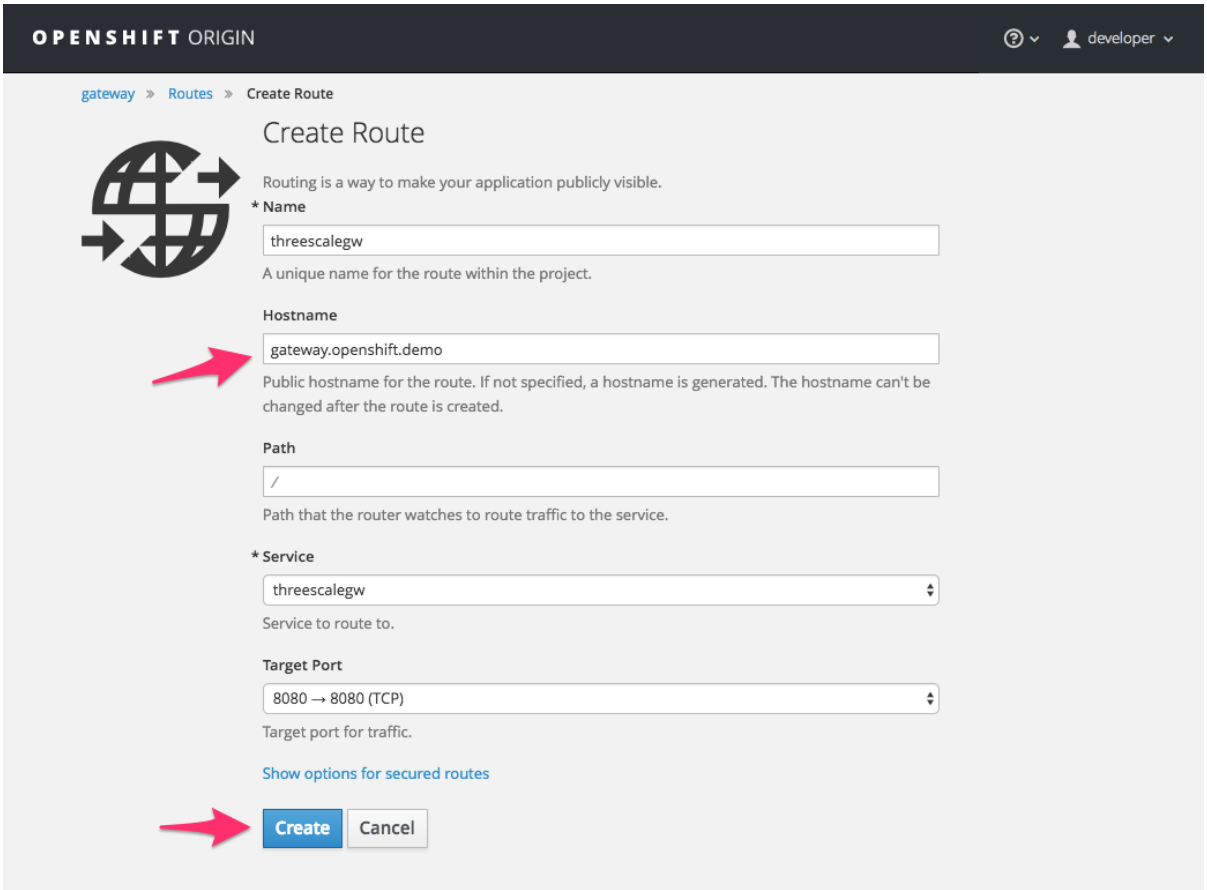
   Each APIcast instance, upon starting, downloads the required configuration from 3scale using the settings you provided on the **Integration** page of your 3scale Admin Portal.

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

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



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. Alternatively, you can configure the APIcast Built-in Wildcard Routing to avoid creating a new route for every 3scale service you define.

CHAPTER 4. APICAST ON THE DOCKER CONTAINERIZED ENVIRONMENT

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

4.1. PREREQUISITES

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

4.2. INSTALLING THE DOCKER CONTAINERIZED ENVIRONMENT

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

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

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

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

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

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

4.3. RUNNING THE DOCKER CONTAINERIZED ENVIRONMENT GATEWAY

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

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

   sudo docker pull registry.access.redhat.com/3scale-amp24/apicast-gateway.
3. Run APIcast in a Docker-formatted container:

sudo docker run --name apicast --rm -p 8080:8080 -e THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net registry.access.redhat.com/3scale-amp24/apicast-gateway.

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

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

4.3.1. The Docker command options

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

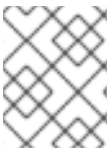
- **--rm**: Automatically removes the container when it exits.
- **-d** or **--detach**: Runs the container in the background and prints the container ID. When it is not specified, the container runs in the foreground mode and you can stop it using **CTRL + c**. When started in the detached mode, you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

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

4.4. TESTING APICAST

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

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



NOTE

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