Installing 3scale

Install and configure 3scale API Management.
Install and configure 3scale API Management.
Abstract

This guide provides the information to install and configure 3scale API Management.
# Table of Contents

## PREFACE ............................................................................. 4

## MAKING OPEN SOURCE MORE INCLUSIVE .......................... 5

## CHAPTER 1. REGISTRY SERVICE ACCOUNTS FOR 3SCALE .......... 6

1.1. CREATING A REGISTRY SERVICE ACCOUNT ......................... 6
1.2. MODIFYING A REGISTRY SERVICE ACCOUNT ....................... 6
1.3. ADDITIONAL RESOURCES ............................................. 7

## CHAPTER 2. INSTALLING APICAST ......................................... 8

2.1. APICAST DEPLOYMENT OPTIONS .................................... 8
2.2. APICAST ENVIRONMENTS ............................................. 8
2.3. CONFIGURING THE INTEGRATION SETTINGS ..................... 9
2.4. CONFIGURING YOUR SERVICE ....................................... 9
2.4.1. Declaring the API backend ....................................... 9
2.4.2. Configuring the authentication settings ....................... 10
2.4.3. Configuring the API test call .................................. 11
2.5. INSTALLING THE APICAST OPERATOR ............................ 12
2.6. DEPLOYING AN APICAST GATEWAY SELF-MANAGED SOLUTION USING THE OPERATOR .......................... 12
2.6.1. APICast deployment and configuration options ................ 13
2.6.1.1. Providing a 3scale system endpoint ....................... 13
2.6.1.1.1. Verifying the APIcast gateway is running and available 14
2.6.1.1.2. Exposing APIcast externally via a Kubernetes Ingress 14
2.6.1.2. Providing a configuration secret ............................ 15
2.6.1.2.1. Verifying APIcast gateway is running and available 16
2.7. WEB_SOCKET PROTOCOL SUPPORT FOR APICAST ............ 17
2.7.1. WebSocket protocol support .................................... 17
2.8. HTTP/2 IN THE APICAST GATEWAY ................................ 17
2.8.1. HTTP/2 protocol support ..................................... 18
2.9. ADDITIONAL RESOURCES ........................................ 18

## CHAPTER 3. APICAST HOSTED .............................................. 19

3.1. DEPLOYING YOUR API WITH APICAST HOSTED IN A STAGING ENVIRONMENT ........................................ 19
3.2. DEPLOYING YOUR API WITH THE APICAST HOSTED INTO PRODUCTION ........................................ 19
3.3. ADDITIONAL INFORMATION .......................................... 20

## CHAPTER 4. RUNNING APICAST ON RED HAT OPENSIFT ............. 21

4.1. SETTING UP RED HAT OPENSIFT .................................. 21
4.1.1. Installing the Docker containerized environment ........... 21
4.1.2. Starting the OpenShift cluster .................................. 22
4.1.3. Setting up the OpenShift cluster on a remote server (Optional) ........................................ 23
4.2. DEPLOYING APICAST USING THE OPENSIFT TEMPLATE .... 23
4.3. CREATING ROUTES VIA THE OPENSIFT CONSOLE .......... 24

## CHAPTER 5. DEPLOYING APICAST ON THE DOCKER CONTAINERIZED ENVIRONMENT ............................... 27

5.1. INSTALLING THE DOCKER CONTAINERIZED ENVIRONMENT .... 27
5.2. RUNNING THE DOCKER CONTAINERIZED ENVIRONMENT GATEWAY ........................................ 28
5.2.1. The docker command options ................................ 28
5.2.2. Testing APIcast .................................................. 29
5.3. ADDITIONAL RESOURCES ........................................ 29

## CHAPTER 6. DEPLOYING APICAST ON PODMAN ........................ 30

6.1. INSTALLING THE PODMAN CONTAINER ENVIRONMENT .... 30
6.2. RUNNING THE PODMAN ENVIRONMENT
   6.2.1. Testing APIcast with Podman
6.3. THE PODMAN COMMAND OPTIONS
6.4. ADDITIONAL RESOURCES
PREFACE

This guide will help you to install and configure 3scale.
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. REGISTRY SERVICE ACCOUNTS FOR 3SCALE

To use container images from registry.redhat.io in a shared environment with 3scale 2-saas, you must use a Registry Service Account instead of an individual user’s Customer Portal credentials.

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.
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 APICAST

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

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

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

- A 3scale Hosted account.

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

- Section 2.1, “APICAST deployment options”
- Section 2.2, “APICAST environments”
- Section 2.3, “Configuring the integration settings”
- Section 2.4, “Configuring your service”
- Section 2.5, “Installing the APIcast operator”
- Section 2.6, “Deploying an APIcast gateway self-managed solution using the operator”
- Section 2.7, “WebSocket protocol support for APIcast”
- Section 2.8, “HTTP/2 in the APIcast gateway”

2.1. 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:

- Hosted APIcast: 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.

- 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.

2.2. APICAST 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 a 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`.

### 2.3. CONFIGURING THE INTEGRATION SETTINGS

As a 3scale administrator, configure the integration settings for the environment you require 3scale to run in.

**Prerequisites**

A 3scale account with administrator privileges.

**Procedure**

1. Navigate to [Your_API_name] > Integration > Settings

2. Under **Deployment**, the default options are as follows:
   - Deployment Option: hosted APIcast
   - Authentication mode: API key.

3. Change to your preferred option.

4. To save your changes, click **Update Product**.

### 2.4. 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

#### 2.4.1. 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"
  ```

### 2.4.2. Configuring the authentication settings

You can configure authentication settings for your API in the **AUTHENTICATION** section under `[Your_product_name] > Integration > Settings`.

**Table 2.1. Optional authentication fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auth user key</strong></td>
<td>Set the user key associated with the credentials location.</td>
</tr>
<tr>
<td><strong>Credentials location</strong></td>
<td>Define whether credentials are passed as HTTP headers, query parameters or as HTTP basic authentication.</td>
</tr>
<tr>
<td><strong>Host Header</strong></td>
<td>Define a custom Host request header. This is required if your API backend only accepts traffic from a specific host.</td>
</tr>
</tbody>
</table>
Furthermore, you can configure the **GATEWAY RESPONSE** error codes under [Your_product_name] > **Integration** > **Settings**. Define the **Response Code**, **Content-type**, and **Response Body** for the errors: Authentication failed, Authentication missing, and No match.

**Table 2.2. Response codes and default response body**

<table>
<thead>
<tr>
<th>Response code</th>
<th>Response body</th>
</tr>
</thead>
<tbody>
<tr>
<td>403</td>
<td>Authentication failed</td>
</tr>
<tr>
<td>403</td>
<td>Authentication parameters missing</td>
</tr>
<tr>
<td>404</td>
<td>No Mapping Rule matched</td>
</tr>
<tr>
<td>429</td>
<td>Usage limit exceeded</td>
</tr>
</tbody>
</table>

### 2.4.3. Configuring the API test call

Configuring the API involves testing the backends with a product and promoting the APIcast configuration to staging and production environments to make tests based on request calls.

For each product, requests get redirected to their corresponding backend according to the path. This path is configured when you add the backend to the product. For example, if you have two backends added to a product, each backend has its own path.

**Prerequisites**

- One or more **backends added to a product**.
- A **mapping rule** for each backend added to a product.
- An **application plan** to define the access policies.
- An **application** that subscribes to the application plan.

**Procedure**

1. Promote an APIcast configuration to Staging, by navigating to [Your_product_name] > **Integration** > **Configuration**.

2. Under **APIcast Configuration**, you will see the mapping rules for each backend added to the product. Click **Promote v.[n] to Staging APIcast**
3. Once promoted to staging, you can promote to Production. Under Staging APIcast, click Promote v.[n] to Production APIcast

- v.[n] indicates the version number to be promoted.

4. To test requests to your API in the command line, use the command provided in Example curl for testing.

- The curl command example is based on the first mapping rule in the product.

When testing requests to your API, you can modify the mapping rules by adding methods and metrics.

Every time you modify the configuration and before making calls to your API, make sure you promote to the Staging and Production environments. When there are pending changes to be promoted to the Staging environment, you will see an exclamation mark in the Admin Portal, next to the Integration menu item.

3scale Hosted APIcast gateway does the validation of the credentials and applies the rate limits that you defined for the application plan of your API. If you make a call without credentials, or with invalid credentials, you will see the error message, Authentication failed.

### 2.5. 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.

### 2.6. 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

2.6.1. 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

2.6.1.1. Providing a 3scale system endpoint

Procedure

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

   ```bash
   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
   ```

   ```bash
   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

   ```yaml
   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:

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

2.6.1.1. 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`:

   ```bash
   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:

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

2.6.1.2. 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`:

```yaml
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 2.3. APIcastExposedHost reference table**

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

### 2.6.1.2. Providing a configuration secret

#### Procedure

1. Create a secret with the configuration file:

   ```bash
   $ 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:

   ```yaml
   $ 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:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
   ```
3. Set the following content when creating the APIcast object:

```yaml
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
```

2.6.1.2. 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:

```bash
$ 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"
}
```

### 2.7. WEBSOCKET PROTOCOL SUPPORT FOR APICAST

Red Hat 3scale 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.

#### 2.7.1. WebSocket protocol support

The APIcast configuration policy chain is as follows:

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

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

### 2.8. HTTP/2 IN THE APICAST GATEWAY

Red Hat 3scale 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.

### 2.8.1. 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:

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

### 2.9. 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
- For the updates on the hosted APIcast version please refer to Red Hat 3scale API Management Platform SaaS Release Notes.
CHAPTER 3. APICAST HOSTED

Once you complete this tutorial, you will 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.

Prerequisites

- You have reviewed Chapter 3, APICAST hosted for deployment alternatives and decided to use APICAST hosted to integrate your API with 3scale.
- Your API backend service is accessible over the public Internet. 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.
- Section 3.1, “Deploying your API with APICAST hosted in a staging environment”
- Section 3.2, “Deploying your API with the APICAST hosted into production”

3.1. DEPLOYING YOUR API WITH APICAST HOSTED IN A STAGING ENVIRONMENT

The first step is to configure your API and test it in your staging environment:

Procedure

1. Define the private base URL and its endpoints.
2. Choose the placement of credentials and other configuration details. For more information see the APIcast Hosted documentation.
3. Save the configuration settings by clicking Update Product. These settings go through the APICAST staging instance to your API.

You will see a green confirmation message when your configuration is complete.

Before moving on to the next step, make sure that you have configured a secret token to that you backend service validates. You can define the value for the secret token under Authentication Settings. This will ensure that nobody can bypass APICAST access control.

3.2. DEPLOYING YOUR API WITH THE APICAST HOSTED INTO PRODUCTION

At this point, you are ready to take your API configuration to a production environment. To deploy your 3scale Hosted APIcast instance:

Procedure

1. Go back to the Integration and Configuration page and click on the 'Promote to v.x to Production' button.
2. 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. API calls may return different responses depending on which instance serves the call. Deployment has been successful when 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 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`

### 3.3. ADDITIONAL INFORMATION

- 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 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 Admin Portal as per Chapter 2, 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 OPENSSHIFT TEMPLATE

Use the following to deploy APIcast using 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/2.11.0.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 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 product 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 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 2, 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.
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:

```
+  

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```
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 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 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 2, *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:

  ```bash
  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:
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-{RHELVersionAPIcast}:3scale2.11
```

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 Supported Configurations.

- For information about getting started with Podman, see Basic Setup and Use of Podman.