# Red Hat 3scale API Management 2.3

## Infrastructure

Learn more about deploying Red Hat 3scale API Management on different platforms.

Last Updated: 2019-10-22

# Red Hat 3scale API Management 2.3 Infrastructure

Learn more about deploying Red Hat 3scale API Management on different platforms.

## Legal Notice

Copyright © 2019 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

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

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This guide documents deployment and infrastructure management with Red Hat 3scale API Management 2.3.

# Table of Contents

CHAPTER 1. UPGRADE 3SCALE API MANAGEMENT 2.2 TO 2.3 ..... 4
1.1. PREREQUISITES 4
1.2. SELECT THE PROJECT 4
1.3. GATHER THE NEEDED VALUES 4
1.4. PATCH APICAST 5
1.5. VERIFY UPGRADE 9
1.6. UPGRADE APICAST IN OPENSHIFT 10

CHAPTER 2. BUILDING A 3SCALE API MANAGEMENT SYSTEM IMAGE WITH THE ORACLE DATABASE RELATIONAL DATABASE MANAGEMENT SYSTEM ..... 11
2.1. BEFORE YOU BEGIN 11
2.1.1. Obtain Oracle software components 11
2.1.2. Meet prerequisites 11
2.2. PREPARING ORACLE DATABASE 11
2.3. BUILDING THE SYSTEM IMAGE 12

CHAPTER 3. 3SCALE API MANAGEMENT ON-PREMISES INSTALLATION GUIDE ..... 14
3.1. PREREQUISITES 14
3.2. 3SCALE AMP OPENSHIFT TEMPLATES 14
3.3. SYSTEM REQUIREMENTS 14
3.3.1. Environment Requirements 14
3.3.2. Hardware Requirements 14
3.4. CONFIGURE NODES AND ENTITLEMENTS 15
3.5. DEPLOY THE 3SCALE AMP ON OPENSHIFT USING A TEMPLATE 15
3.5.1. Prerequisites 15
3.5.2. Import the AMP Template 16
3.5.3. Configure SMTP Variables (Optional) 17
3.6. 3SCALE AMP TEMPLATE PARAMETERS 18
3.7. USE APICAST WITH AMP ON OPENSHIFT 21
3.7.1. Deploy APIcast Templates on an Existing OpenShift Cluster Containing your AMP 22
3.7.2. Connect APIcast from an OpenShift Cluster Outside an OpenShift Cluster Containing your AMP 22
3.7.3. Connect APIcast from Other Deployments 23
3.7.4. Change Built-In APIcast Default Behavior 23
3.7.5. Connect Multiple APIcast Deployments on a Single OpenShift Cluster over Internal Service Routes 24
3.8. 7. TROUBLESHOOTING 25
3.8.1. Previous Deployment Leaves Dirty Persistent Volume Claims 25
3.8.2. Incorrectly Pulling from the Docker Registry 25
3.8.3. Permissions Issues for MySQL when Persistent Volumes are Mounted Locally 26
3.8.4. Unable to Upload Logo or Images because Persistent Volumes are not Writable by OpenShift 26
3.8.5. Create Secure Routes on OpenShift 27
3.8.6. APIcast on a Different Project from AMP Fails to Deploy due to Problem with Secrets 27

CHAPTER 4. 3SCALE API MANAGEMENT ON-PREMISES OPERATIONS AND SCALING GUIDE ..... 28
4.1. INTRODUCTION 28
4.1.1. Prerequisites 28
4.1.2. Further Reading 28
4.2. RE-DEPLOYING APICAST 28
4.3. APICAST BUILT-IN WILDCARD ROUTING 29
4.3.1. Modify Wildcards 29
4.4. SCALING UP AMP ON PREMISES 29
4.4.1. Scaling up Storage 29
4.4.1.1. Method 1: Backup and Swap Persistent Volumes 30

4.4.1.2. Method 2: Back up and Redeploy AMP 30
4.4.2. Scaling up Performance 30
4.4.2.1. Configuring 3scale On-Premises Deployments 30
4.4.2.2. Vertical and Horizontal Hardware Scaling 31
4.4.2.3. Scaling Up Routers 31
4.4.2.4. Further Reading 31
4.5. OPERATIONS TROUBLESHOOTING 31
4.5.1. Access Your Logs 32
4.5.2. Job Queues 32

**CHAPTER 5. HOW TO DEPLOY A FULL-STACK API SOLUTION WITH FUSE, 3SCALE, AND OPENSHIFT 33**
5.1. PART 1: FUSE ON OPENSHIFT SETUP 34
5.1.1. Step 1 34
5.1.2. Step 2 35
5.1.3. Step 3 35
5.1.4. Step 4 36
5.1.5. Step 5 37
5.1.6. Step 6 38
5.1.7. Step 7 38
5.1.8. Step 8 39
5.1.9. Step 9 39
5.1.10. Step 10 40
5.1.11. Step 11 41
5.2. PART 2: CONFIGURE 3SCALE API MANAGEMENT 41
5.2.1. Step 1 41
5.2.2. Step 2 41
5.2.3. Step 3 42
5.2.4. Step 4 43
5.2.5. Step 5 43
5.3. PART 3: INTEGRATION OF YOUR API SERVICES 44
5.4. PART 4: TESTING THE API AND API MANAGEMENT 44
5.4.1. Step 1 44
5.4.2. Step 2 45
5.4.3. Step 3 45
5.4.4. Step 4 46
5.4.5. Step 5 46

# CHAPTER 1. UPGRADE 3SCALE API MANAGEMENT 2.2 TO 2.3

The 2.3 version of 3scale API Management only updates the APIcast component of the product.

Perform the steps in this document to upgrade the APIcast to version 2.3.

## 1.1. PREREQUISITES

- You must be running 3scale On-Premises 2.2
- OpenShift CLI

**WARNING**

Red Hat recommends that you establish a maintenance window when performing the upgrade because this process may cause a disruption in service.

## 1.2. SELECT THE PROJECT

1. From a terminal session, log in to your OpenShift cluster using the following command. Here, <YOUR_OPENSHIFT_CLUSTER> is the URL of your OpenShift cluster.

```
oc login https://<YOUR_OPENSHIFT_CLUSTER>:8443
```

2. Select the project you want to upgrade using the following command. Here, <3scale-22-project> is the name of your project.

```
oc project <3scale-22-project>
```

## 1.3. GATHER THE NEEDED VALUES

1. Gather the following values from the APIcast component of your current 2.2 deployment:
   - APICAST_MANAGEMENT_API
   - OPENSSL_VERIFY
   - APICAST_RESPONSE_CODES
2. Export these values from the current deployment into the active shell.

```
export `oc env dc/apicast-production --list | grep -E
'^(APICAST_MANAGEMENT_API|OPENSSL_VERIFY|APICAST_RESPONSE_CODES)=' |
tr "\n" ' ' `
```

3. Optionally, to query individual values from the OpenShift CLI, run the following **oc get** command, where **<variable_name>** is the name of the variable you want to query.

```
oc get "-o=custom-columns=NAMES:.spec.template.spec.containers[0].env[?(.name==\"
<variable_name>\")].value" dc/apicast-production
```

4. Set the value for the new version of the 3scale API Management release.

```
export AMP_RELEASE=2.3.0
```

5. Set the values for the new environment variables.

```
export AMP_APICAST_IMAGE=registry.access.redhat.com/3scale-amp23/apicast-gateway
```

6. Set the **APICAST_ACCESS_TOKEN** environment variable with the valid Access Token for Account Management API. You can extract it from the **THREESCALE_PORTAL_ENDPOINT** environment variable.

```
oc env dc/apicast-production --list | grep THREESCALE_PORTAL_ENDPOINT
```

This will return the following output:

```
THREESCALE_PORTAL_ENDPOINT=http://<ACCESS_TOKEN>@system-
master:3000/master/api/proxy/configs
```

7. Export the **<ACCESS_TOKEN>** value to the environment variable **APICAST_ACCESS_TOKEN**.

```
export APICAST_ACCESS_TOKEN=<ACCESS_TOKEN>
```

8. Confirm that the necessary values are exported to the active shell.

```
echo AMP_RELEASE=$AMP_RELEASE
echo AMP_APICAST_IMAGE=$AMP_APICAST_IMAGE
echo APICAST_ACCESS_TOKEN=$APICAST_ACCESS_TOKEN
echo APICAST_MANAGEMENT_API=$APICAST_MANAGEMENT_API
echo OPENSSL_VERIFY=$OPENSSL_VERIFY
echo APICAST_RESPONSE_CODES=$APICAST_RESPONSE_CODES
```

## 1.4. PATCH APICAST

1. To patch the **apicast-staging** deployment configuration, run the following **oc patch** command.

```
oc patch dc/apicast-staging -p "
metadata:
  name: apicast-staging
  labels:
    app: APIcast
    3scale.component: apicast
    3scale.component-element: staging
spec:
  replicas: 1
  selector:
    deploymentConfig: apicast-staging
  strategy:
```

```
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 1800
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      labels:
        deploymentConfig: apicast-staging
        app: APIcast
        3scale.component: apicast
        3scale.component-element: staging
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '9421'
    spec:
      containers:
      - env:
        - name: THREESCALE_PORTAL_ENDPOINT
          value: \"http://${APICAST_ACCESS_TOKEN}@system-
master:3000/master/api/proxy/configs\"
        - name: APICAST_CONFIGURATION_LOADER
          value: \"lazy\"
        - name: APICAST_CONFIGURATION_CACHE
          value: \"0\"
        - name: THREESCALE_DEPLOYMENT_ENV
          value: \"sandbox\"
        - name: APICAST_MANAGEMENT_API
          value: \"${APICAST_MANAGEMENT_API}\"
        - name: BACKEND_ENDPOINT_OVERRIDE
          value: http://backend-listener:3000
        - name: OPENSSL_VERIFY
          value: '${APICAST_OPENSSL_VERIFY}'
        - name: APICAST_RESPONSE_CODES
          value: '${APICAST_RESPONSE_CODES}'
        - name: REDIS_URL
          value: \"redis://system-redis:6379/2\"
        image: amp-apicast:latest
        imagePullPolicy: IfNotPresent
        name: apicast-staging
        resources:
          limits:
            cpu: 100m
            memory: 128Mi
          requests:
            cpu: 50m
            memory: 64Mi
        livenessProbe:
          httpGet:
            path: /status/live
            port: 8090
          initialDelaySeconds: 10
          timeoutSeconds: 5
          periodSeconds: 10
```

```
        readinessProbe:
          httpGet:
            path: /status/ready
            port: 8090
          initialDelaySeconds: 15
          timeoutSeconds: 5
          periodSeconds: 30
        ports:
        - containerPort: 8080
          protocol: TCP
        - containerPort: 8090
          protocol: TCP
        - name: metrics
          containerPort: 9421
          protocol: TCP
  triggers:
  - type: ConfigChange
  - type: ImageChange
    imageChangeParams:
      automatic: true
      containerNames:
      - apicast-staging
      from:
        kind: ImageStreamTag
        name: amp-apicast:latest
"
```

2. To patch the **apicast-production** deployment configuration, run the following **oc patch** command.

```
oc patch dc/apicast-production -p "
metadata:
  name: apicast-production
  labels:
    app: APIcast
    3scale.component: apicast
    3scale.component-element: production
spec:
  replicas: 1
  selector:
    deploymentConfig: apicast-production
  strategy:
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 1800
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      labels:
        deploymentConfig: apicast-production
        app: APIcast
        3scale.component: apicast
        3scale.component-element: production
```

```
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '9421'
    spec:
      initContainers:
      - name: system-master-svc
        image: amp-apicast:latest
        command: ['sh', '-c', 'until \$(curl --output /dev/null --silent --fail --head http://system-
master:3000/status); do sleep $SLEEP_SECONDS; done']
        activeDeadlineSeconds: 1200
        env:
        - name: SLEEP_SECONDS
          value: \"1\"
      containers:
      - env:
        - name: THREESCALE_PORTAL_ENDPOINT
          value: \"http://${APICAST_ACCESS_TOKEN}@system-
master:3000/master/api/proxy/configs\"
        - name: APICAST_CONFIGURATION_LOADER
          value: \"boot\"
        - name: APICAST_CONFIGURATION_CACHE
          value: \"300\"
        - name: THREESCALE_DEPLOYMENT_ENV
          value: \"production\"
        - name: APICAST_MANAGEMENT_API
          value: \"${APICAST_MANAGEMENT_API}\"
        - name: BACKEND_ENDPOINT_OVERRIDE
          value: http://backend-listener:3000
        - name: OPENSSL_VERIFY
          value: '${APICAST_OPENSSL_VERIFY}'
        - name: APICAST_RESPONSE_CODES
          value: '${APICAST_RESPONSE_CODES}'
        - name: REDIS_URL
          value: \"redis://system-redis:6379/1\"
        image: amp-apicast:latest
        imagePullPolicy: IfNotPresent
        name: apicast-production
        resources:
          limits:
            cpu: 1000m
            memory: 128Mi
          requests:
            cpu: 500m
            memory: 64Mi
        livenessProbe:
          httpGet:
            path: /status/live
            port: 8090
          initialDelaySeconds: 10
          timeoutSeconds: 5
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /status/ready
            port: 8090
          initialDelaySeconds: 15
```

```
        timeoutSeconds: 5
        periodSeconds: 30
      ports:
      - containerPort: 8080
        protocol: TCP
      - containerPort: 8090
        protocol: TCP
      - name: metrics
        containerPort: 9421
        protocol: TCP
  triggers:
  - type: ConfigChange
  - type: ImageChange
    imageChangeParams:
      automatic: true
      containerNames:
      - system-master-svc
      - apicast-production
      from:
        kind: ImageStreamTag
        name: amp-apicast:latest
"
```

3. To patch the **amp-apicast** image stream, run the following **oc patch** command.

```
oc patch is/amp-apicast -p "
metadata:
  name: amp-apicast
  labels:
    app: APIcast
    3scale.component: apicast
  annotations:
    openshift.io/display-name: AMP APIcast
spec:
  tags:
  - name: latest
    annotations:
      openshift.io/display-name: AMP APIcast (latest)
    from:
      kind: ImageStreamTag
      name: "${AMP_RELEASE}"
  - name: "${AMP_RELEASE}"
    annotations:
      openshift.io/display-name: AMP APIcast ${AMP_RELEASE}
    from:
      kind: DockerImage
      name: ${AMP_APICAST_IMAGE}
    importPolicy:
      insecure: false
"
```

4. Set **importPolicy.insecure** to **true** if the server is allowed to bypass certificate verification or connect directly over HTTP during image import.

## 1.5. VERIFY UPGRADE

After you have performed the upgrade procedure, verify the success of the upgrade by making test API calls to the updated APIcast.

NOTE

It may take some time for the redeployment operations to complete in OpenShift.

## 1.6. UPGRADE APICAST IN OPENSHIFT

If you deployed APIcast outside of the complete 3scale API Management on-premises installation using the **apicast.yml** OpenShift template, take the following steps to upgrade your deployment. The steps assume that the name of the deployment configuration is **apicast**, which is the default value in the **apicast.yml** template of the 3scale version 2.2. If you used a different name, you must adjust the commands accordingly.

1. Update the container image

```
oc patch dc/apicast --patch='{"spec":{"template":{"spec":{"containers":[{"name": "apicast",
"image":"registry.access.redhat.com/3scale-amp23/apicast-gateway"}]}}}}'
```

2. Add the port definition for port **9421** used for Prometheus metrics

```
oc patch dc/apicast --patch='{"spec": {"template": {"spec": {"containers": [{"name":
"apicast","ports": [{"name": "metrics", "containerPort": 9421, "protocol": "TCP"}]}]}}}}'
```

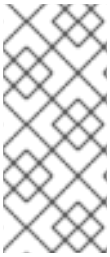
3. Add Prometheus annotations

```
oc patch dc/apicast --patch='{"spec": {"template": {"metadata": {"annotations":
{"prometheus.io/scrape": "true", "prometheus.io/port": "9421"}}}}}'
```

4. Remove the **APICAST_WORKERS** environment variable

```
oc env dc/apicast APICAST_WORKERS-
```

**APICAST_WORKERS** allows specifying the value for the directive **worker_processes**. By default, APIcast uses the value **auto**, which triggers auto-detection of the best number of workers, when running in OpenShift or Kubernetes environments. Thus, it is recommended not set the **APICAST_WORKERS** value explicitly and let APIcast perform auto-detection.



NOTE

The **APICAST_WORKERS** parameter is no longer present in the **apicast.yml** OpenShift template. In case you are using scripts that deploy the template with **APICAST_WORKERS** parameter, make sure you remove this parameter from the scripts, otherwise the deployment will fail with the following error: **error: unexpected parameter name "APICAST_WORKERS"**

# CHAPTER 2. BUILDING A 3SCALE API MANAGEMENT SYSTEM IMAGE WITH THE ORACLE DATABASE RELATIONAL DATABASE MANAGEMENT SYSTEM

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

## 2.1. BEFORE YOU BEGIN

### 2.1.1. Obtain Oracle software components

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

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

### 2.1.2. Meet prerequisites

You must also meet the following prerequisites:

- A supported version of Oracle Database accessible from your OpenShift cluster
- Access to the Oracle Database **system** user for installation procedures
- Possess the Red Hat 3scale 2.3 amp.yml template

## 2.2. PREPARING ORACLE DATABASE

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

   ```
   ALTER SYSTEM SET max_string_size=extended SCOPE=SPFILE;
   ```

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

2. Collect the database details.
   Get the following information that will be needed for 3scale configuration:
   - Oracle Database URL
   - Oracle Database service name
   - Oracle Database **system** user name and password
   - Oracle Database service name

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

## 2.3. BUILDING THE SYSTEM IMAGE

1. clone the 3scale-amp-openshift-templates github repository
2. place your Oracle Database Instant Client Package files into the **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** directory
3. run the **oc new-app** command with the -f option and specify the **build.yml** OpenShift template

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

4. run the **oc new-app** command with the -f option, specifying the **amp.yml** OpenShift template, and the -p option, specifying the **WILDCARD_DOMAIN** parameter with the domain of your OpenShift cluster

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

5. enter the following shell **for** loop command, specifying the following information you collected in the Preparing Oracle Database section previously:
   - **{USER}**: the username that will represent 3scale in your Oracle Database
   - **{PASSWORD}**: the password for **USER**
   - **{ORACLE_DB_URL}**: the URL of your Oracle Database
   - **{DATABASE}**: the service name of the database you created in Oracle Database
   - **{PORT}**: the port number of your Oracle Database

```
for dc in system-app system-resque system-sidekiq system-sphinx; do oc env dc/$dc --overwrite DATABASE_URL="oracle-enhanced://{USER}:{PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"; done
```

6. enter the following **oc patch** command, specifying the same **USER**, **PASSWORD**, **ORACLE_DB_URL**, **PORT**, and **DATABASE** values that you provided in the previous step above:

```
$ oc patch dc/system-app -p '[{"op": "replace", "path": "/spec/strategy/rollingParams/pre/execNewPod/env/1/value", "value": "oracle-enhanced://{USER}:{PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"}]' --type=json
```

7. enter the following **oc patch** command, specifying your own Oracle Database **system** user password in the **SYSTEM_PASSWORD** field:

```
$ 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
```

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

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

# CHAPTER 3. 3SCALE API MANAGEMENT ON-PREMISES INSTALLATION GUIDE

This guide walks you through steps to install 3scale 2.3 (on-premises) on OpenShift using OpenShift templates.

## 3.1. PREREQUISITES

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

## 3.2. 3SCALE AMP OPENSHIFT TEMPLATES

Red Hat 3scale API Management Platform (AMP) 2.3 provides an OpenShift template. You can use this template to deploy AMP onto OpenShift Container Platform.

The 3scale AMP template is composed of the following:

- Two built-in APIcast API gateways
- One AMP admin portal and developer portal with persistent storage

## 3.3. SYSTEM REQUIREMENTS

This section lists the requirements for the 3scale API Management OpenShift template.

### 3.3.1. Environment Requirements

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

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

### 3.3.2. Hardware Requirements

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

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

- Set the **PUMA_WORKERS** variable of the backend listener to the number of cores in your compute node.

## 3.4. CONFIGURE NODES AND ENTITLEMENTS

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

Perform the following steps to configure the entitlements:

1. Install Red Hat Enterprise Linux (RHEL) on each of your nodes.
2. Register your nodes with Red Hat using the Red Hat Subscription Manager (RHSM), via the interface or the command line.
3. Attach your nodes to your 3scale subscription using RHSM.
4. Install OpenShift on your nodes, complying with the following requirements:
   - Use a supported OpenShift version.
   - Configure persistent storage on a file system that supports multiple writes.
5. Install the OpenShift command line interface.
6. Enable access to the **rhel-7-server-3scale-amp-2.3-rpms** repository using the subscription manager:

```
sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.3-rpms
```

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

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

## 3.5. DEPLOY THE 3SCALE AMP ON OPENSHIFT USING A TEMPLATE

### 3.5.1. Prerequisites

- An OpenShift cluster configured as specified in the Chapter 3, *Configure Nodes and Entitlements* section.
- A domain, preferably wildcard, that resolves to your OpenShift cluster.
- Access to the Red Hat container catalog.
- (Optional) A working SMTP server for email functionality.

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

- Import the AMP Template
- Configure SMTP Variables (Optional)

### 3.5.2. Import the AMP Template

Perfrom the following steps to import the AMP template into your OpenShift cluster:

1. From a terminal session log in to OpenShift:

```
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 the configure nodes and entitlements section .

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

   c. Optionally, specify the **--param** option with the **WILDCARD_POLICY** parameter set to **subdomain** to enable wildcard domain routing:
   *Without Wildcard Routing:*

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

   *With Wildcard Routing:*

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

   The terminal shows the master and tenant URLs and credentials for your newly created AMP 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'
Access your application via route 'apicast-wildcard.3scale-project.example.com'
```

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

**NOTE**

You may need to wait a few minutes for AMP to fully deploy on OpenShift for your login and credentials to work.

**More Information**

For information about wildcard domains on OpenShift, visit Using Wildcard Routes (for a Subdomain) .

### 3.5.3. Configure SMTP Variables (Optional)

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

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

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

```
oc login
```

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

| Variable | Description |
|---|---|
| address | Allows you to specify a remote mail server as a relay |
| username | Specify your mail server username |
| password | Specify your mail server password |
| domain | Specify a HELO domain |

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

**Example**

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

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

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

## 3.6. 3SCALE AMP TEMPLATE PARAMETERS

Template parameters configure environment variables of the AMP yml template during and after deployment.

| Name | Description | Default Value | Required? |
|---|---|---|---|
| APP_LABEL | Used for object app labels | "3scale-api-management" | yes |
| ZYNC_DATABASE_PASSWORD | Password for the PostgreSQL connection user. Generated randomly if not provided. | N/A | yes |
| ZYNC_SECRET_KEY_BASE | Secret key base for Zync. Generated randomly if not provided. | N/A | yes |

| | | | |
|---|---|---|---|
| ZYNC_AUTHENTICATION_TOKEN | Authentication token for Zync. Generated randomly if not provided. | N/A | yes |
| AMP_RELEASE | AMP release tag. | 2.3.0 | yes |
| ADMIN_PASSWORD | A randomly generated AMP administrator account password. | N/A | yes |
| ADMIN_USERNAME | AMP 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 |
| WILDCARD_POLICY | Enable wildcard routes to built-in APIcast gateways by setting the value as "Subdomain" | **None** | yes |
| TENANT_NAME | Tenant name under the root that Admin UI will be available with -admin suffix. | 3scale | yes |
| MYSQL_USER | Username for MySQL user that will be used for accessing the database. | mysql | yes |
| MYSQL_PASSWORD | Password for the MySQL user. | N/A | yes |
| MYSQL_DATABASE | Name of the MySQL database accessed. | system | yes |

| | | | |
|---|---|---|---|
| MYSQL_ROOT_PASSWORD | Password for Root user. | N/A | yes |
| SYSTEM_BACKEND_USERNAME | Internal 3scale API username for internal 3scale api auth. | 3scale_api_user | yes |
| SYSTEM_BACKEND_PASSWORD | Internal 3scale API password for internal 3scale api auth. | N/A | yes |
| REDIS_IMAGE | Redis image to use | registry.access.redhat.com/rhscl/redis-32-rhel7:3.2 | yes |
| MYSQL_IMAGE | Mysql image to use | registry.access.redhat.com/rhscl/mysql-57-rhel7:5.7 | yes |
| MEMCACHED_IMAGE | Memcached image to use | registry.access.redhat.com/3scale-amp20/memcached:1.4.15 | yes |
| POSTGRESQL_IMAGE | Postgresql image to use | registry.access.redhat.com/rhscl/postgresql-95-rhel7:9.5 | yes |
| AMP_SYSTEM_IMAGE | 3scale System image to use | registry.access.redhat.com/3scale-amp22/system | yes |
| AMP_BACKEND_IMAGE | 3scale Backend image to use | registry.access.redhat.com/3scale-amp22/backend | yes |
| AMP_APICAST_IMAGE | 3scale APIcast image to use | registry.access.redhat.com/3scale-amp23/apicast-gateway | yes |
| AMP_ROUTER_IMAGE | 3scale Wildcard Router image to use | registry.access.redhat.com/3scale-amp22/wildcard-router | yes |
| AMP_ZYNC_IMAGE | 3scale Zync image to use | registry.access.redhat.com/3scale-amp22/zync | yes |
| SYSTEM_BACKEND_SHARED_SECRET | Shared secret to import events from backend to system. | N/A | yes |

| | | | |
|---|---|---|---|
| SYSTEM_APP_SECRET_KEY_BASE | System application secret key base | N/A | yes |
| APICAST_MANAGEMENT_API | Scope of the APIcast Management API. Can be disabled, status or debug. At least status required for health checks. | status | no |
| APICAST_OPENSSL_VERIFY | Turn on/off the OpenSSL peer verification when downloading the configuration. Can be set to true/false. | false | no |
| APICAST_RESPONSE_CODES | Enable logging response codes in APIcast. | true | no |
| APICAST_REGISTRY_URL | A URL which resolves to the location of APIcast policies | http://apicast-staging:8090/policies | yes |
| MASTER_USER | Master administrator account username | master | yes |
| MASTER_NAME | The subdomain value for the master admin portal, will be appended with the **-master** suffix | master | yes |
| MASTER_PASSWORD | A randomly generated master administrator password | N/A | yes |
| 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 |

## 3.7. USE APICAST WITH AMP ON OPENSHIFT

APIcast with AMP on OpenShift differs from APIcast with AMP hosted and requires unique configuration procedures.

This section explains how to deploy APIcast with AMP on OpenShift.

### 3.7.1. Deploy APIcast Templates on an Existing OpenShift Cluster Containing your AMP

AMP OpenShift templates contain two built-in APIcast API gateways by default. If you require more API gateways, or require separate APIcast deployments, you can deploy additional APIcast templates on your OpenShift cluster.

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

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

```
oc login
```

3. Create a secret that allows APIcast to communicate with AMP. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your AMP 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 UI will be available with. The default value for **TENANT_NAME** 3scale. If you used a custom value in your AMP deployment then you must use that value here.

4. Import the APIcast template by downloading the apicast.yml, located on the 3scale GitHub, and running the **oc new-app** command, specifying the **--file** option with the **apicast.yml** file:

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

### 3.7.2. Connect APIcast from an OpenShift Cluster Outside an OpenShift Cluster Containing your AMP

If you deploy APIcast on a different OpenShift cluster, outside your AMP cluster, you must connect over the public route.

1. Create an access token with the following configurations:
   - Coped 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 AMP. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your AMP 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 UI will be available with. The default value for`TENANT_NAME` is **3scale**. If you used a custom value in your AMP deployment then you must use that value here.

4. Deploy APIcast on an OpenShift cluster outside of the OpenShift Cluster with the oc new-app command. Specify the **--file** option and the file path of your **apicast.yml** file:

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

5. Update the apicast **BACKEND_ENDPOINT_OVERRIDE** environment variable set to the URL **backend.** followed by the wildcard domain of the OpenShift Cluster containing your AMP deployment:

```
oc env dc/apicast --overwrite BACKEND_ENDPOINT_OVERRIDE=https://backend-
<TENANT_NAME>.<WILDCARD_DOMAIN>
```

### 3.7.3. Connect APIcast from Other Deployments

After you have deployed APIcast on other platforms, you can connect them to AMP on OpenShift by configuring the **BACKEND_ENDPOINT_OVERRIDE** environment variable in your AMP OpenShift Cluster:

1. Log in to your AMP OpenShift Cluster:

```
oc login
```

2. Configure the system-app object **BACKEND_ENDPOINT_OVERRIDE** environment variable:

   - If you are using a native installation: BACKEND_ENDPOINT_OVERRIDE=https://backend.<your_openshift_subdomain> bin/apicast
   - If are using the Docker containerized environment: docker run -e BACKEND_ENDPOINT_OVERRIDE=https://backend.<your_openshift_subdomain>

### 3.7.4. Change Built-In APIcast Default Behavior

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

In built-in APIcast deployments, AMP 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 built-in APIcast deployments.

### 3.7.5. Connect Multiple APIcast Deployments on a Single OpenShift Cluster over Internal Service Routes

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

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

**ovs-subnet**

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

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

```
oc login
```

2. Enter the **oc new-app** command with the path to the **apicast.yml** file:
    a. Specify the **--param** option with the **BACKEND_ENDPOINT_OVERRIDE** parameter set to the domain of your OpenShift cluster's AMP project:

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.<AMP_PROJECT>.svc.cluster.local:3000
```

**ovs-multitenant**

If you are using the 'ovs-multitenant' Openshift SDN plugin, take the following steps to connect over the internal routes:

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

```
oc login
```

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

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

3. Enter the **oc new-app** option with the path to the **apicast.yml** file:
    a. Specify the **--param** option with the **BACKEND_ENDPOINT_OVERRIDE** parameter set to the domain of your OpenShift cluster's AMP project:

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.<AMP_PROJECT>.svc.cluster.local:3000
```

**More information**

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

## 3.8. 7. TROUBLESHOOTING

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

- Previous Deployment Leaves Dirty Persistent Volume Claims
- Incorrectly Pulling from the Docker Registry
- Permissions Issues for MySQL when Persistent Volumes are Mounted Locally
- Unable to Upload Logo or Images Because Persistent Volumes are not Writable by OpenShift
- Create Secure Routes on OpenShift
- APIcast on a Different Project from AMP Fails to Deploy Due to Problem with Secrets

### 3.8.1. Previous Deployment Leaves Dirty Persistent Volume Claims

**Problem**

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

**Cause**

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

**Solution**

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

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

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

### 3.8.2. 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.access.redhat.com** Red Hat container registry.

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

**Solution**

Use the appropriate version of the Docker containerized environment.

### 3.8.3. Permissions 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] Can't start server : on unix socket: Permission denied
[ERROR] Do you already have another mysqld server running on socket: /var/lib/mysql/mysql.sock ?
[ERROR] Aborting
```

**Cause**

The MySQL process is started with inappropriate user permissions.

**Solution**

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

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

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

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

### 3.8.4. Unable to Upload Logo or Images because Persistent Volumes are not Writable by OpenShift

**Problem**

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

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

**Cause**

Persistent volumes are not writable by OpenShift.

**Solution**

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

### 3.8.5. Create Secure Routes on OpenShift

**Problem**

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

**Cause**

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

**Solution**

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

### 3.8.6. APIcast on a Different Project from AMP Fails to Deploy due to Problem with Secrets

**Problem**

APIcast deploy fails (pod doesn't turn blue). The following error appears in the logs:

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

The following error appears in the pod:

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

**Cause**

The secret was not properly set up.

**Solution**

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

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

# CHAPTER 4. 3SCALE API MANAGEMENT ON-PREMISES OPERATIONS AND SCALING GUIDE

## 4.1. INTRODUCTION

This document describes operations and scaling tasks of a Red Hat 3scale AMP 2.3 On-Premises installation.

### 4.1.1. Prerequisites

An installed and initially configured AMP On-Premises instance on a supported OpenShift version.

This document is not intended for local installations on laptops or similar end user equipment.

### 4.1.2. Further Reading

- Health and Liveness Monitoring
- OpenShift Documentation

## 4.2. RE-DEPLOYING APICAST

After you have deployed AMP On-Premises and your chosen APIcast deployment method, you can test and promote system changes through your AMP dashboard. By default, APIcast deployments on OpenShift, both built-in and on other OpenShift clusters, are configured to allow you to publish changes to your staging and production gateways through the AMP UI.

Redeploy APIcast on OpenShift:

1. Make system changes.
2. In the UI, deploy to staging and test.
3. In the UI, promote to production.

By default, APIcast retrieves and publishes the promoted update once every 5 minutes.

If you are using APIcast on the Docker containerized environment or a native installation, you must configure your staging and production gateways, and configure how often your gateway retrieves published changes. After you have configured your APIcast gateways, you can redeploy APIcast through the AMP UI.

To redeploy APIcast on the Docker containerized environment or a native installations:

1. Configure your APIcast gateway and connect it to AMP On-Premises.
2. Make system changes.
3. In the UI, deploy to staging and test.
4. In the UI, promote to production.

APIcast retrieves and publishes the promoted update at the configured frequency.

## 4.3. APICAST BUILT-IN WILDCARD ROUTING

The built-in APIcast gateways that accompany your on-preimses AMP deployment support wildcard domain routing at the subdomain level. This feature allows you to name a portion of your subdomain for your production and staging gateway public base URLs. To use this feature, you must have enabled it during the on-premises installation.

**NOTE**

Ensure that you are using the OpenShift Container Platform version that supports Wildcard Routing. For information on the supported versions, see Supported Configurations.

The AMP does not provide DNS capabilities, so your specified public base URL must match the DNS configuration specified in the **WILDCARD_DOMAIN** parameter of the OpenShift cluster on which it was deployed.

### 4.3.1. Modify Wildcards

Perform the following steps to modify your wildcards:

1. Log in to your AMP.
2. Navigate to your API gateway settings page: **APIs** → your API → **Integration** → **edit APIcast configuration**
3. Modify the staging and production public base URLs with a string prefix of your choice, adhere to these requirements:
    - API endpoints must not begin with a numeric character

The following is an example of a valid wildcard for a staging gateway on the domain **example.com**:

```
apiname-staging.example.com
```

**More Information**

For information on routing, see the OpenShift documentation.

## 4.4. SCALING UP AMP ON PREMISES

### 4.4.1. Scaling up Storage

As your APIcast deployment grows, you may need to increase the amount of storage available. How you scale up storage depends on which type of file system you are using for your persistent storage.

If you are using a network file system (NFS), you can scale up your persistent volume using the **oc edit pv** command:

```
oc edit pv <pv_name>
```

If you are using any other storage method, you must scale up your persistent volume manually using one of the methods listes in the following sections.

#### 4.4.1.1. Method 1: Backup and Swap Persistent Volumes

1. Back up the data on your existing persistent volume.
2. Create and attach a target persistent volume, scaled for your new size requirements.
3. Create a pre-bound persistent volume claim, specify: The size of your new PVC The persistent volume name using the **volumeName** field.
4. Restore data from your backup onto your newly created PV.
5. Modify your deployment configuration with the name of your new PV:

```
oc edit dc/system-app
```

6. Verify your new PV is configured and working correctly.
7. Delete your previous PVC to release its claimed resources.

#### 4.4.1.2. Method 2: Back up and Redeploy AMP

1. Back up the data on your existing persistent volume.
2. Shut down your 3scale pods.
3. Create and attach a target persistent volume, scaled for your new size requirements.
4. Restore data from your backup onto your newly created PV.
5. Create a pre-bound persistent volume claim. Specify:
   a. The size of your new PVC
   b. The persistent volume name using the **volumeName** field.
6. Deploy your AMP.yml.
7. Verify your new PV is configured and working correctly.
8. Delete your previous PVC to release its claimed resources.

### 4.4.2. Scaling up Performance

#### 4.4.2.1. Configuring 3scale On-Premises Deployments

By default, 3scale deployments run one process per pod. You can increase performance by running more processes per pod. Red Hat recommends running 1-2 processes per core on each node.

Perform the following steps to add more processes to a pod:

1. Log in to your OpenShift cluster.

```
oc login
```

2. Switch to your 3scale project.

```
oc project <project_name>
```

3. Set the appropriate environment variable to the desired number of processes per pod.

   a. **APICAST_WORKERS** for APIcast pods (Red Hat recommends to keep this environment variable unset to allow APIcast to determine the number of workers by the number of CPUs available to the APIcast pod)

   b. **PUMA_WORKERS** for backend pods

   c. **UNICORN_WORKERS** for system pods

```
oc env dc/apicast --overwrite APICAST_WORKERS=<number_of_processes>
```

```
oc env dc/backend --overwrite PUMA_WORKERS=<number_of_processes>
```

```
oc env dc/system-app --overwrite UNICORN_WORKERS=<number_of_processes>
```

#### 4.4.2.2. Vertical and Horizontal Hardware Scaling

You can increase the performance of your AMP deployment on OpenShift by adding resources. You can add more compute nodes as pods to your OpenShift cluster (horizontal scaling) or you can allocate more resources to existing compute nodes (vertical scaling).

**Horizontal Scaling**

You can add more compute nodes as pods to your OpenShift. If the additional compute nodes match the existing nodes in your cluster, you do not have to reconfigure any environment variables.

**Vertical Scaling**

You can allocate more resources to existing compute nodes. If you allocate more resources, you must add additional processes to your pods to increase performance.

NOTE

Red Hat does not recommend mixing compute nodes of a different specification or configuration on your 3scale deployment.

#### 4.4.2.3. Scaling Up Routers

As your traffic increases, you must ensure your OCP routers can adequately handle requests. If your routers are limiting the throughput of your requests, you must scale up your router nodes.

#### 4.4.2.4. Further Reading

- Scaling tasks, adding hardware compute nodes to OpenShift
- Adding Compute Nodes
- Routers

## 4.5. OPERATIONS TROUBLESHOOTING

### 4.5.1. Access Your Logs

Each component's deployment configuration contains logs for access and exceptions. If you encounter issues with your deployment, check these logs for details.

Follow these steps to access logs in 3scale:

1. Find the ID of the pod you want logs for:

```
oc get pods
```

2. Enter **oc logs** and the ID of your chosen pod:

```
oc logs <pod>
```

The system pod has two containers, each with a separate log. To access a container's log, specify the **--container** parameter with the **system-provider** and **system-developer**:

```
oc logs <pod> --container=system-provider
oc logs <pod> --container=system-developer
```

### 4.5.2. Job Queues

Job Queues contain logs of information sent from the **system-resque** and **system-sidekiq** pods. Use these logs to check if your cluster is processing data. You can query the logs using the OpenShift CLI:

```
oc get jobs
```

```
oc logs <job>
```

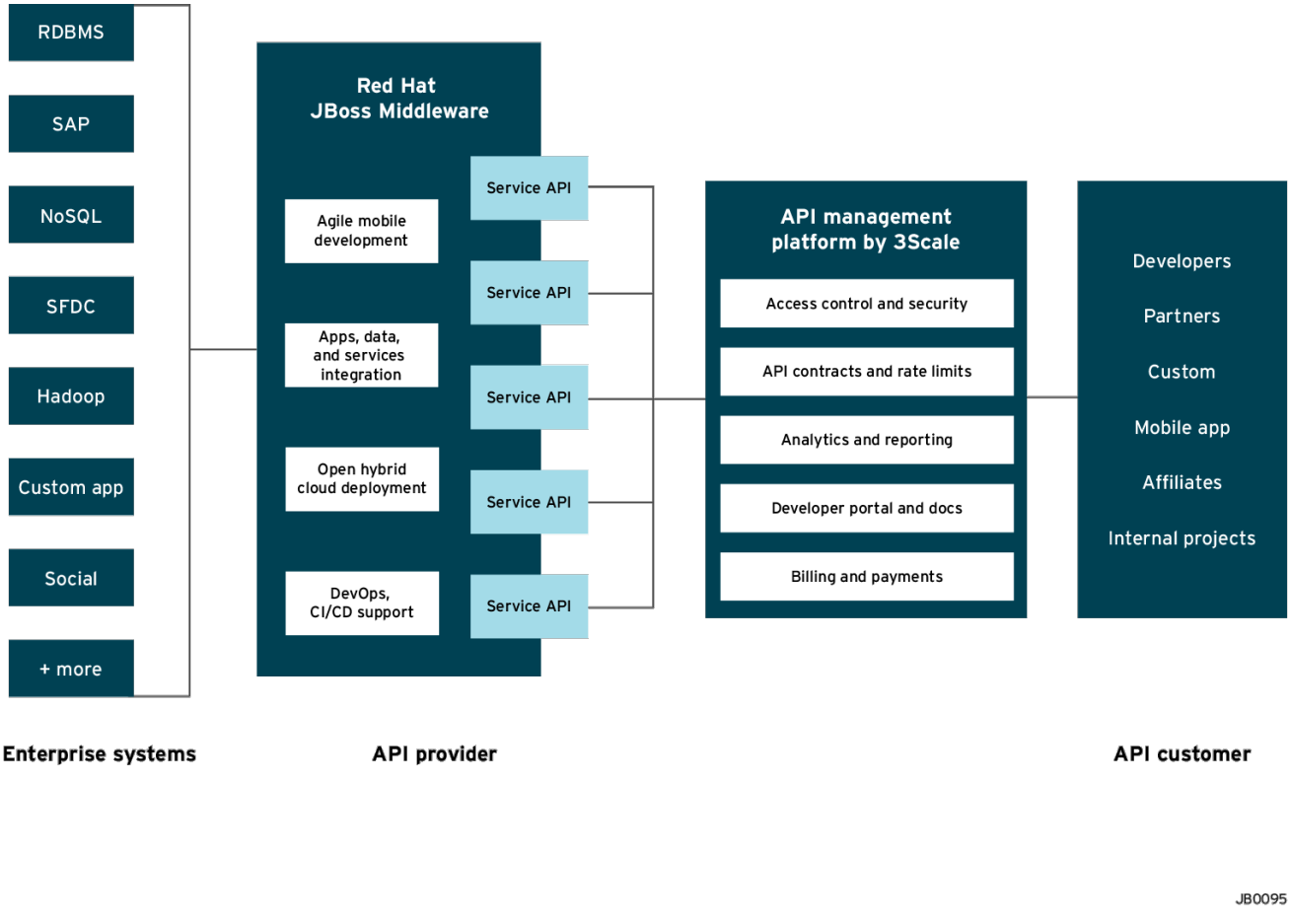
# CHAPTER 5. HOW TO DEPLOY A FULL-STACK API SOLUTION WITH FUSE, 3SCALE, AND OPENSHIFT

This tutorial describes how to get a full-stack API solution (API design, development, hosting, access control, monetization, etc.) using Red Hat JBoss xPaaS for OpenShift and 3scale API Management Platform - Cloud.

The tutorial is based on a collaboration between Red Hat and 3scale to provide a full-stack API solution. This solution includes design, development, and hosting of your API on the Red Hat JBoss xPaaS for OpenShift, combined with the 3scale API Management Platform for full control, visibility, and monetization features.

The API itself can be deployed on Red Hat JBoss xPaaS for OpenShift, which can be hosted in the cloud as well as on premise (that's the Red Hat part). The API management (the 3scale part) can be hosted on Amazon Web Services (AWS), using 3scale APIcast or OpenShift. This gives a wide range of different configuration options for maximum deployment flexibility.

The diagram below summarizes the main elements of this joint solution. It shows the whole integration chain including enterprise backend systems, middleware, API management, and API customers.




For specific support questions, please contact support.

This tutorial shows three different deployment scenarios step by step:

1. Scenario 1 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on Amazon Web Services (AWS) using the 3scale AMI.

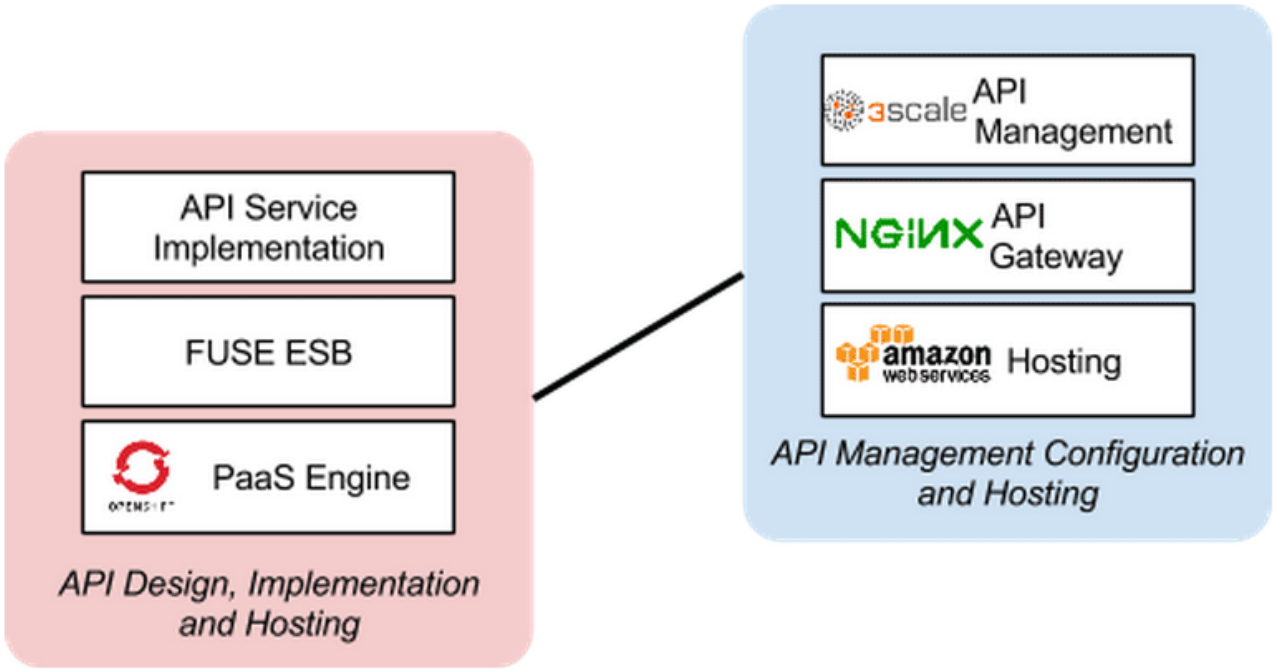
2. Scenario 2 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on APIcast (3scale's cloud hosted API gateway).

3. Scenario 3 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on OpenShift

This tutorial is split into four parts:

- Part 1: Fuse on OpenShift setup to design and implement the API
- Part 2: Configuration of 3scale API Management
- Part 3: Integration of your API services
- Part 4: Testing the API and API management

The diagram below shows the roles the various parts play in this configuration.

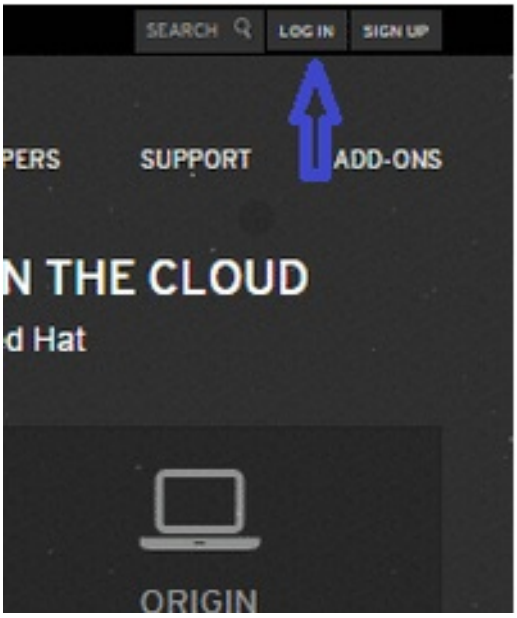



## 5.1. PART 1: FUSE ON OPENSHIFT SETUP

You will create a Fuse on OpenShift application that contains the API to be managed. You will use the REST quickstart that is included with Fuse 6.1. This requires a medium or large gear, as using the small gear will result in memory errors and/or horrible performance.

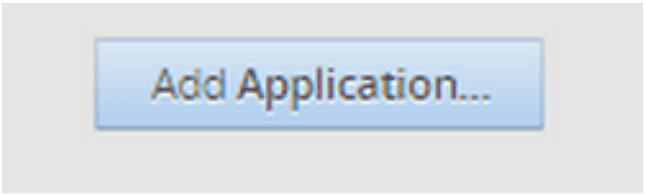
### 5.1.1. Step 1

Sign in to your OpenShift online account. Sign up for an OpenShift online account if you don't already have one.



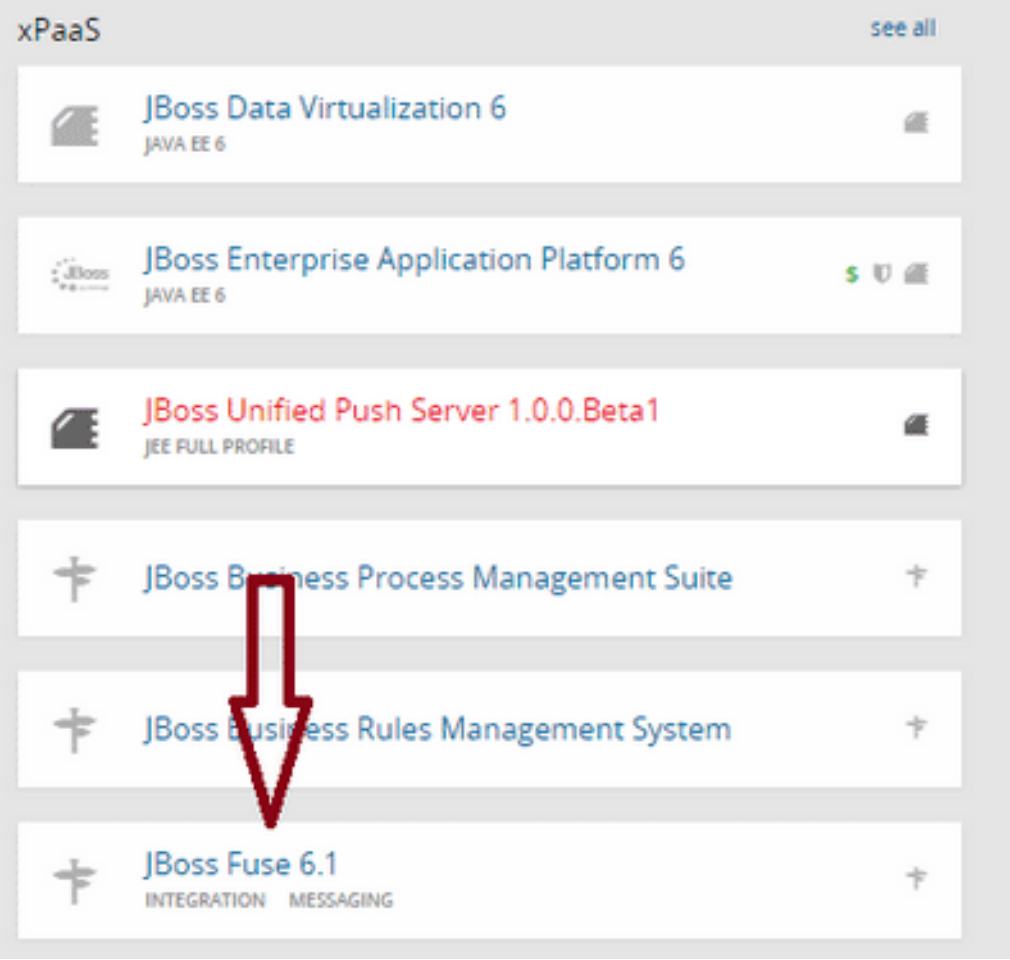

### 5.1.2. Step 2

Click the "add application" button after signing in.



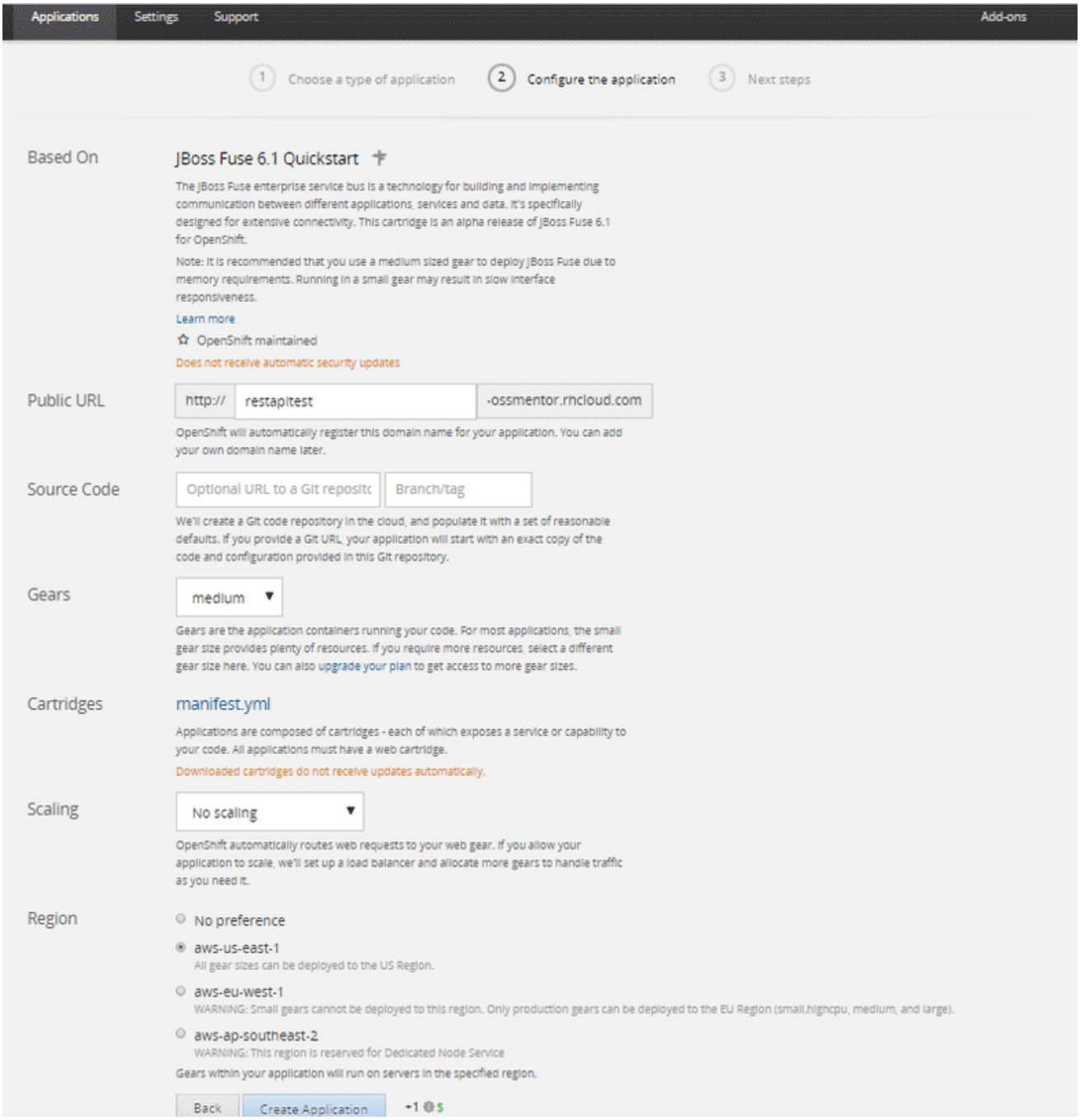

### 5.1.3. Step 3

Under xPaaS, select the Fuse type for the application.



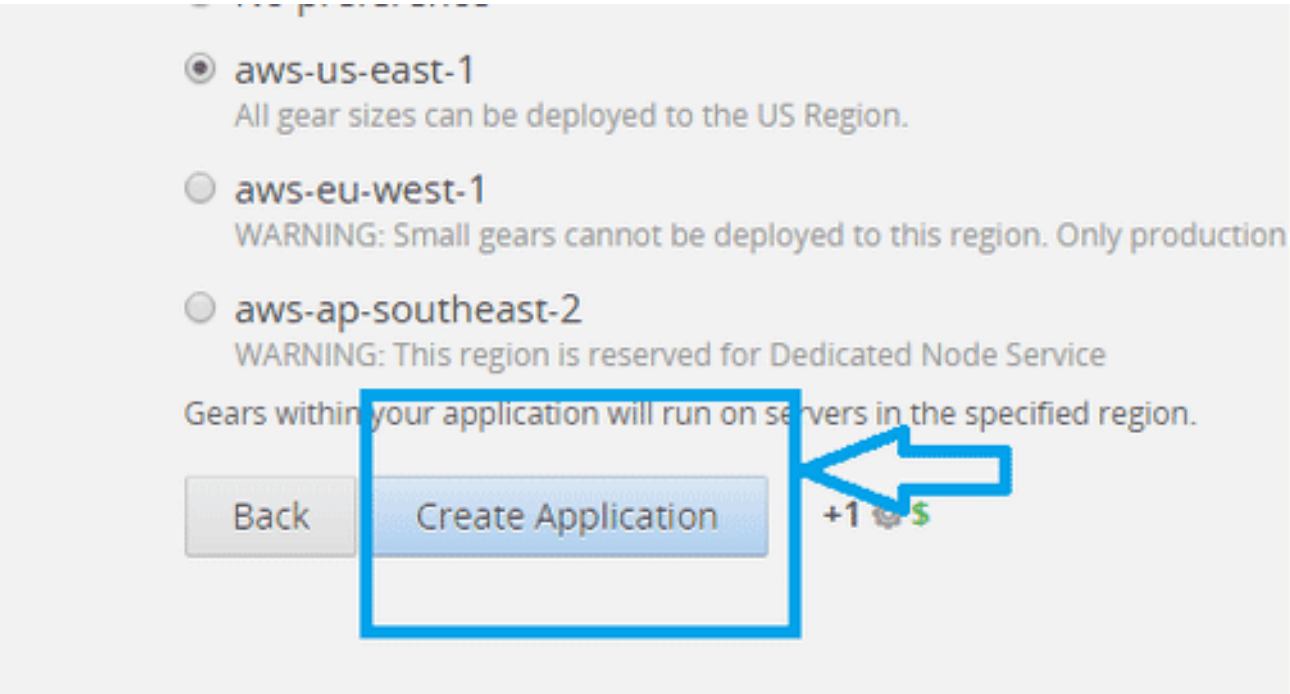

### 5.1.4. Step 4

Now configure the application. Enter the subdomain you'd like your application to show up under, such as "restapitest". This will give a full URL of the form "appname-domain.rhcloud.com" – in the example below "restapitest-ossmentor.rhcloud.com". Change the gear size to medium or large, which is required for the Fuse cartridge. Now click on "create application".



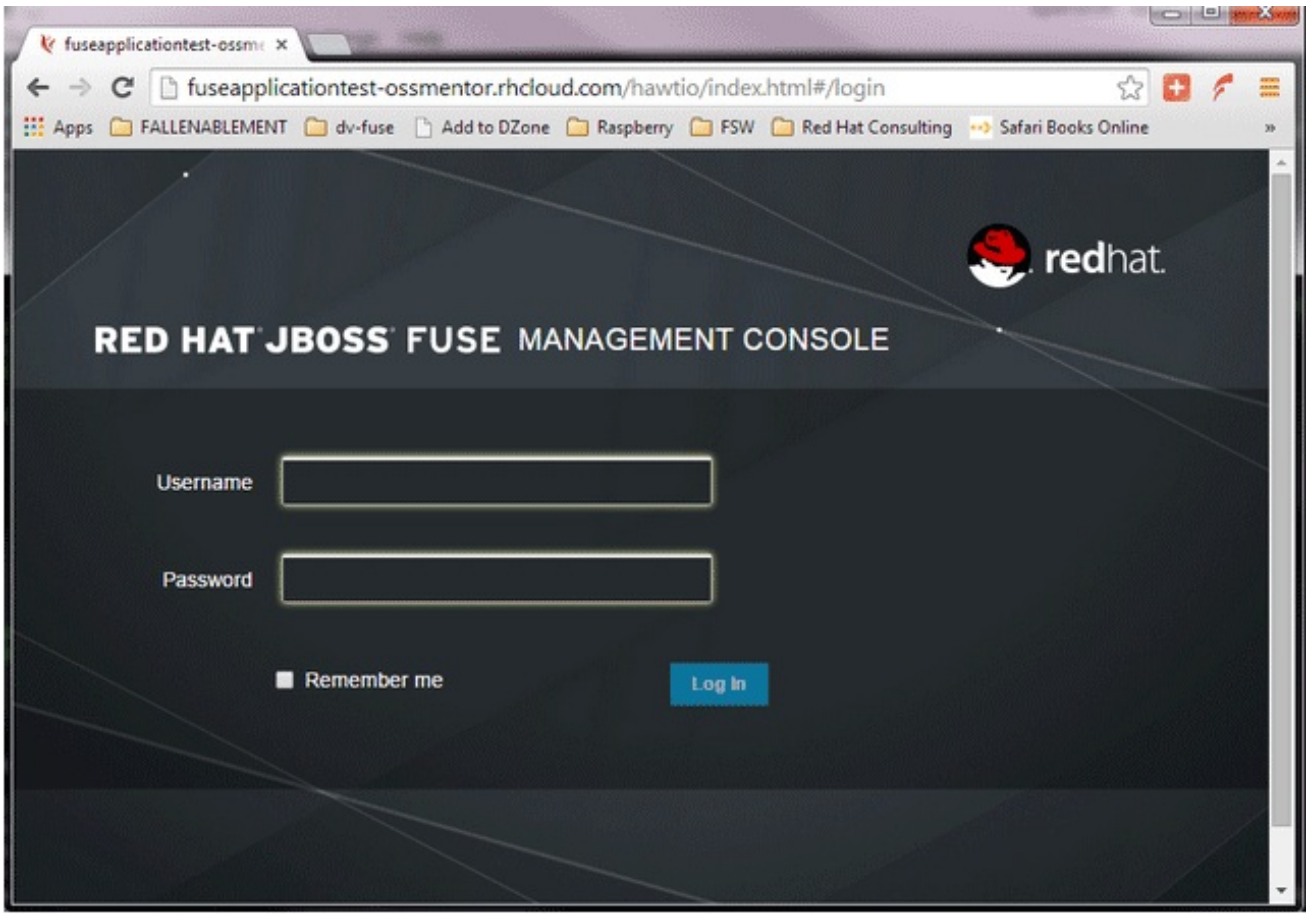

### 5.1.5. Step 5

Click "create application".



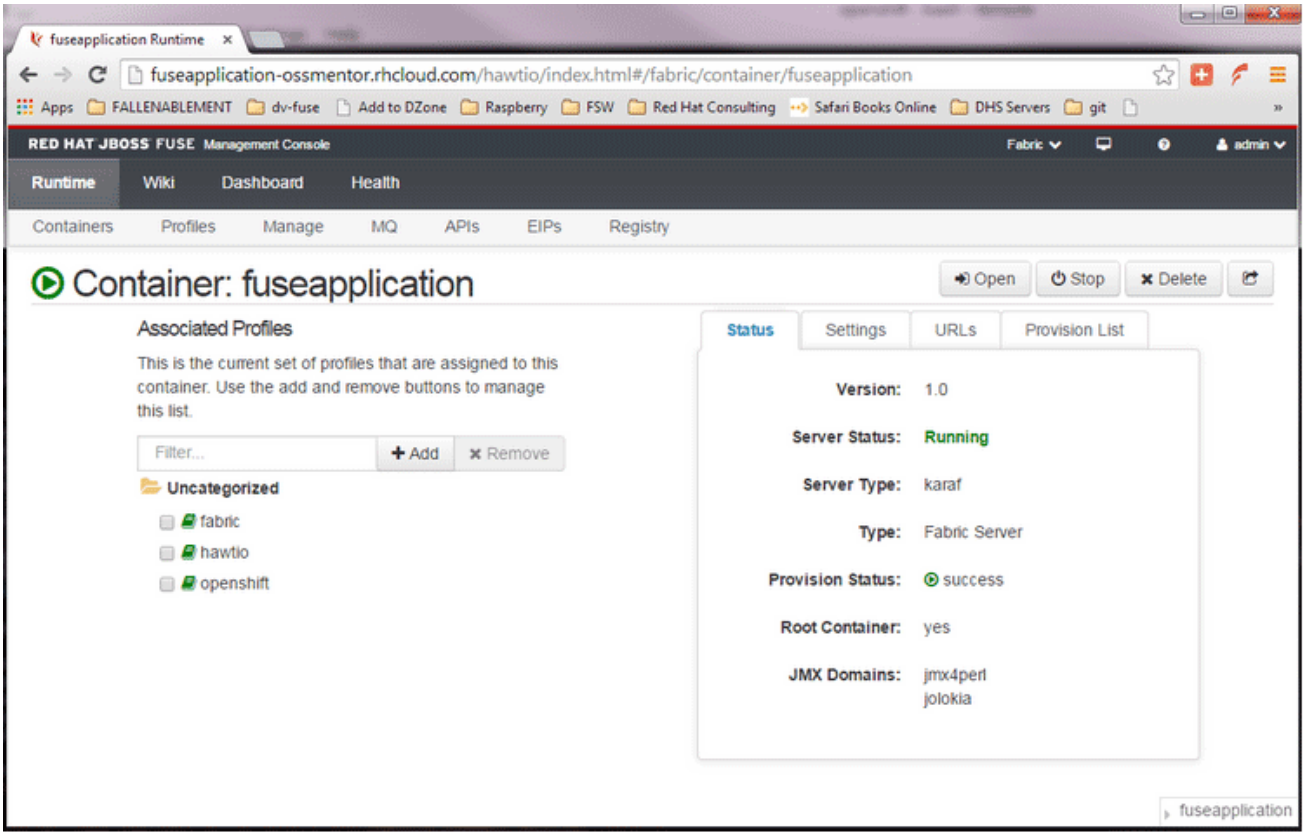

### 5.1.6. Step 6

Browse the application hawtio console and sign in.



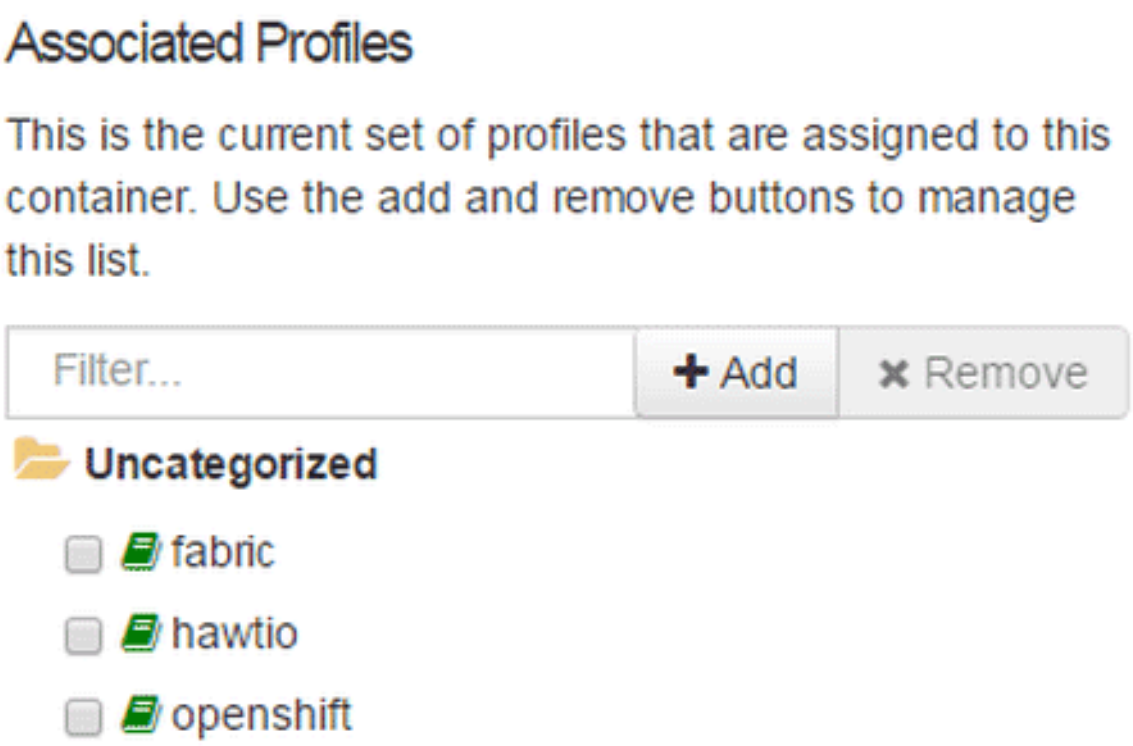

### 5.1.7. Step 7

After signing in, click on the "runtime" tab and the container, and add the REST API example.



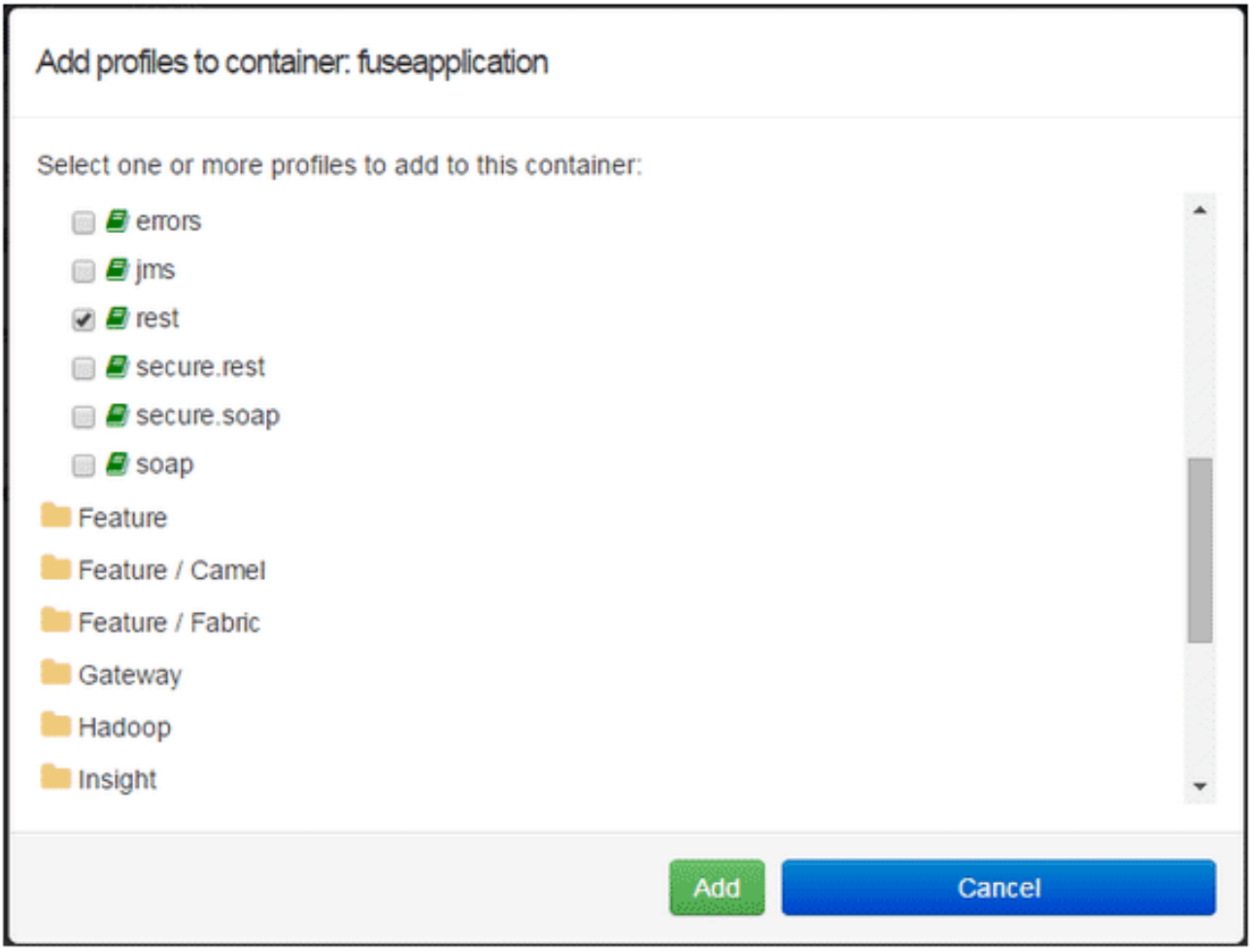

### 5.1.8. Step 8

Click on the "add a profile" button.



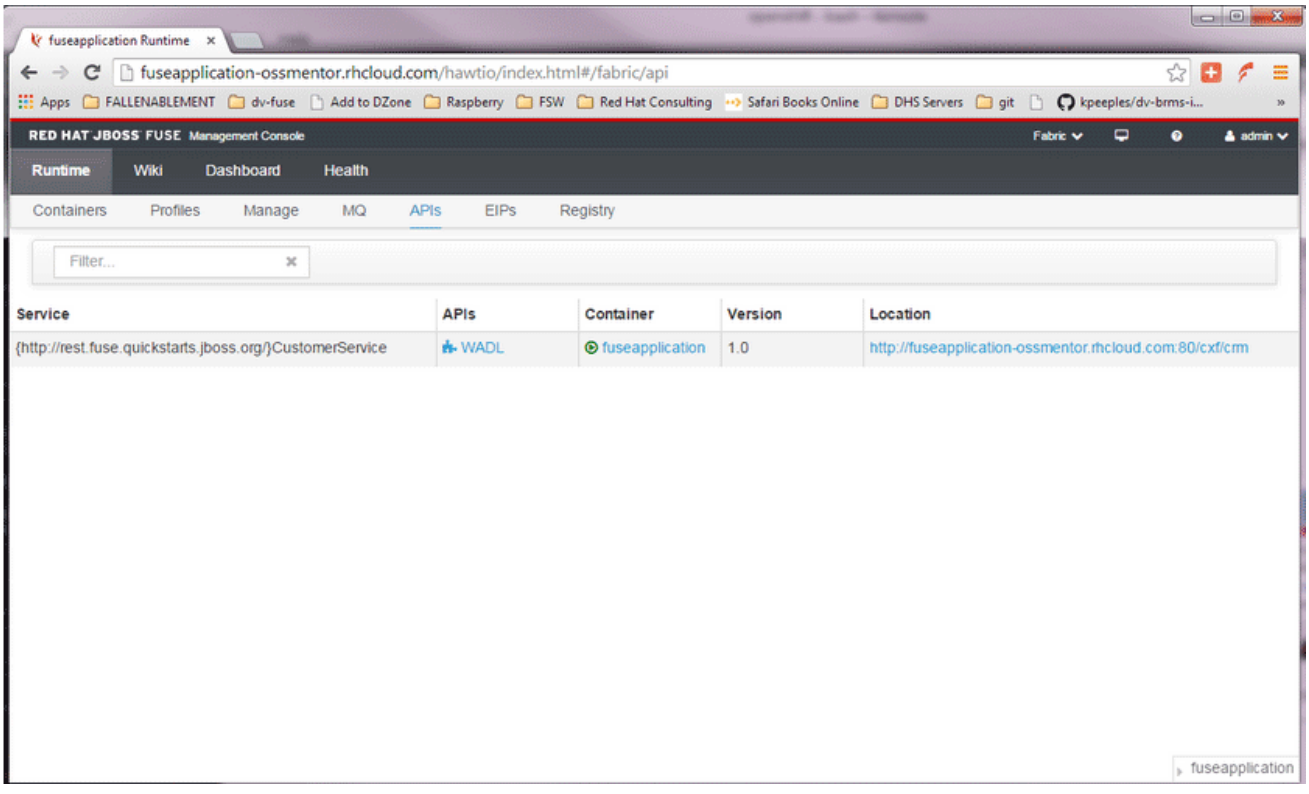

### 5.1.9. Step 9

Scroll down to examples/quickstarts and click the "REST" checkbox, then "add". The REST profile should show up on the container associated profile page.




### 5.1.10. Step 10

Click on the runtime/APIs tab to verify the REST API profile.



### 5.1.11. Step 11

Verify the REST API is working. Browse to customer 123, which will return the ID and name in XML format.

## 5.2. PART 2: CONFIGURE 3SCALE API MANAGEMENT

To protect the API that you just created in Part 1 using 3scale API Management, you first must conduct the according configuration, which is then later deployed according to one of the three scenarios presented.

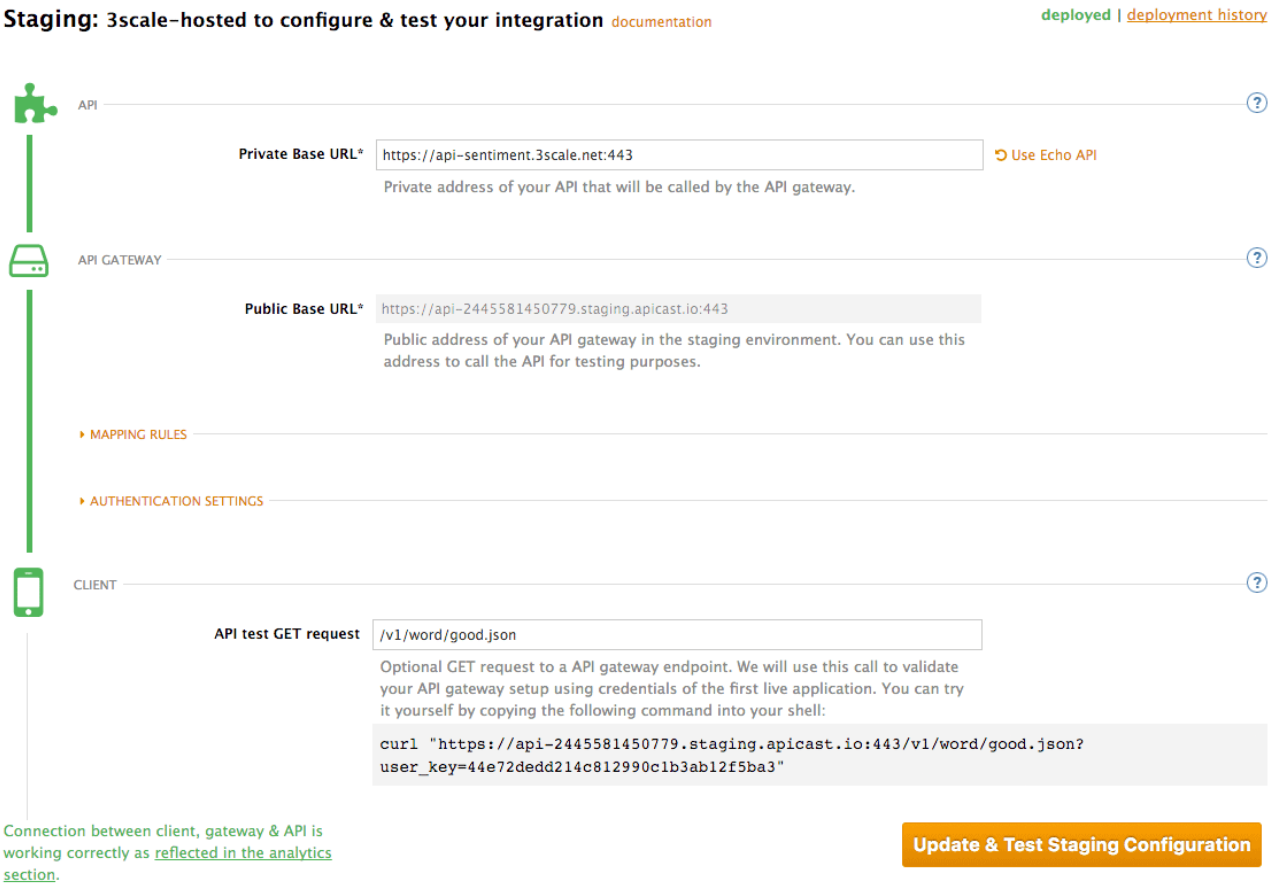
Once you have your API set up on OpenShift, you can start setting it up on 3scale to provide the management layer for access control and usage monitoring.

### 5.2.1. Step 1

Log in to your 3scale account. You can sign up for a 3scale account at www.3scale.net if you don't already have one. When you log in to your account for the first time, follow the wizard to learn the basics about integrating your API with 3scale.

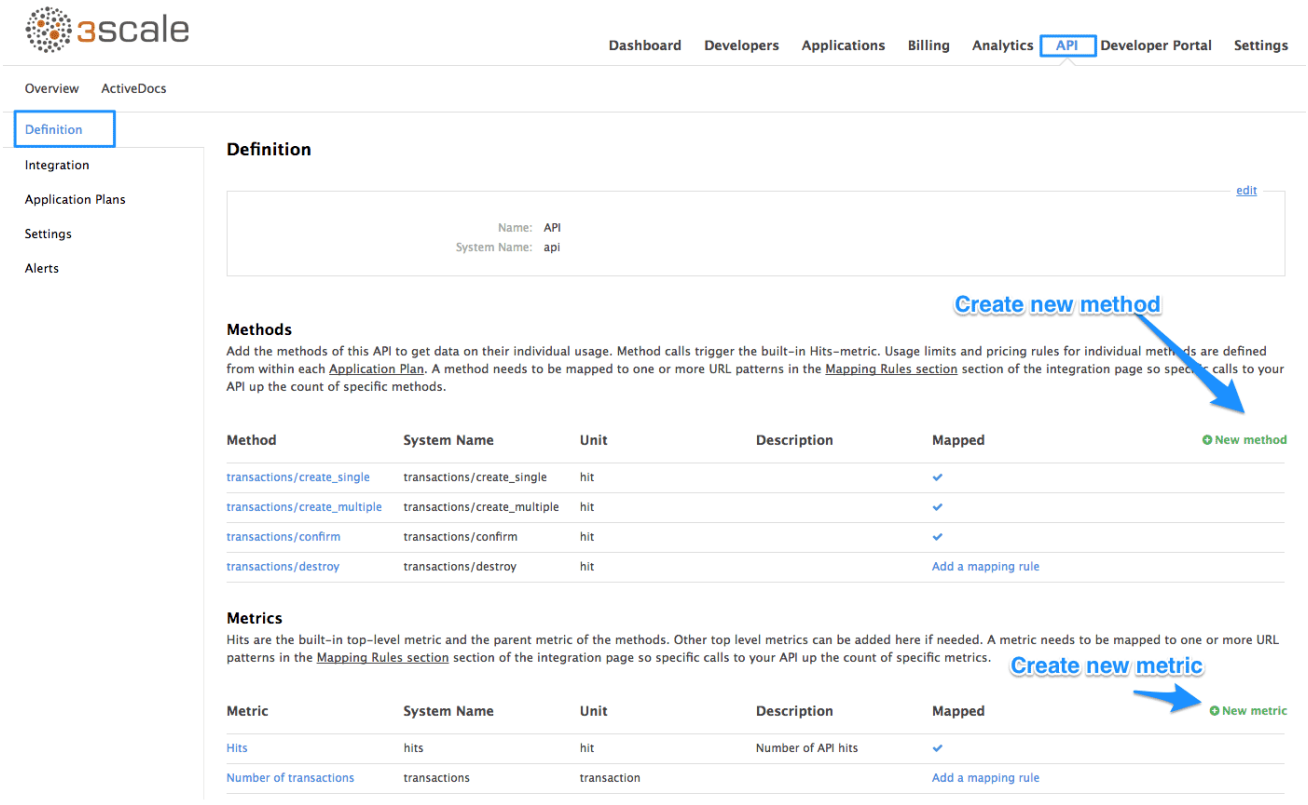
### 5.2.2. Step 2

In **API > Integration**, you can enter the public URL for the Fuse application on OpenShift that you just created, e.g. "restapitest-ossmentor.rhcloud.com" and click on **Test**. This will test your setup against the 3scale API Gateway in the staging environment. The staging API gateway allows you to test your 3scale setup before deploying your proxy configuration to AWS.




### 5.2.3. Step 3

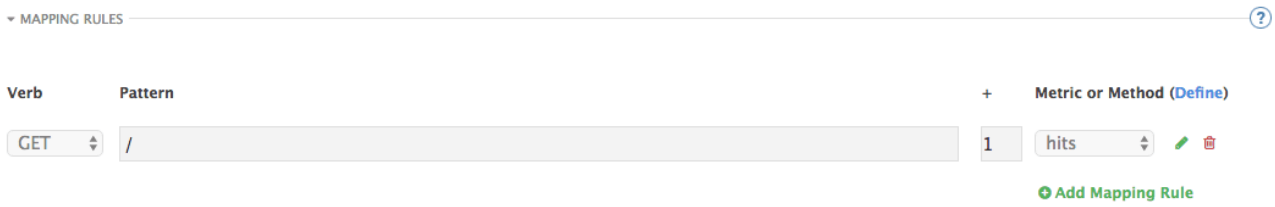
The next step is to set up the API methods that you want to monitor and rate limit. To do that go to **API > Definition** and click on 'New method'.



For more details on creating methods, visit our API definition tutorial.

### 5.2.4. Step 4

Once you have all of the methods that you want to monitor and control set up under the application plan, you'll need to map these to actual HTTP methods on endpoints of your API. Go back to the integration page and expand the "mapping rules" section.



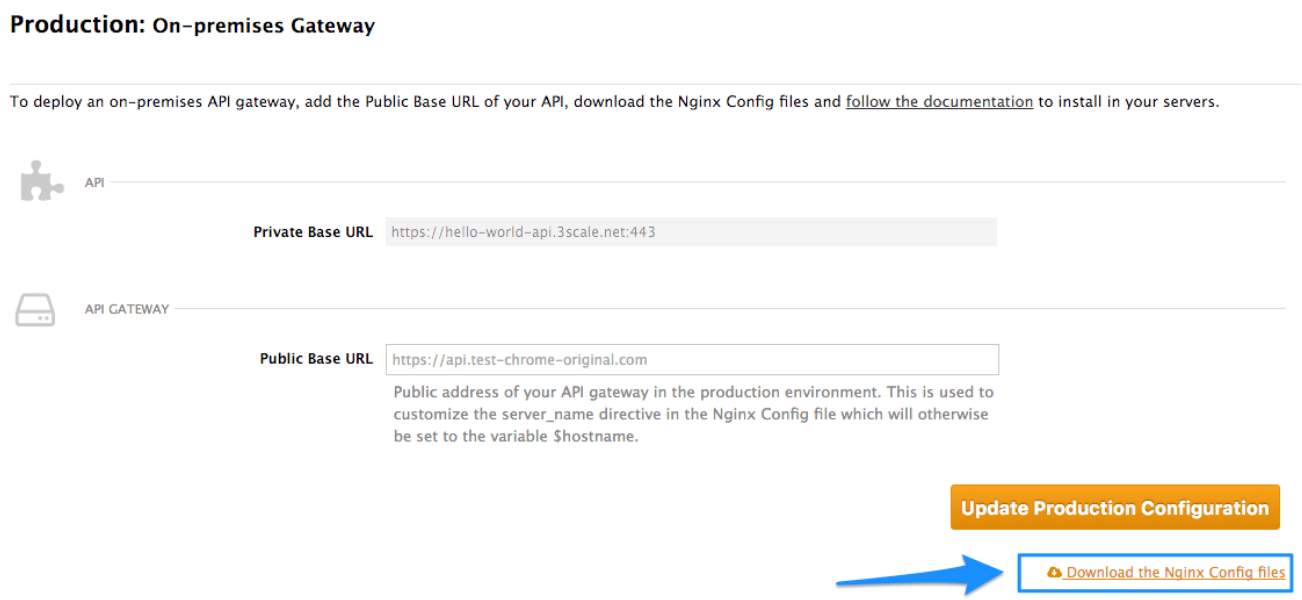

Create mapping rules for each of the methods you created under the application plan.

Once you have done that, your mapping rules will look something like this:

For more details on mapping rules, visit our tutorial about mapping rules.

### 5.2.5. Step 5

Once you've clicked "update and test" to save and test your configuration, you are ready to download the set of configuration files that will allow you to configure your API gateway on AWS. For the API gateway, you should use a high-performance, open-source proxy called nginx. You will find the necessary configuration files for nginx on the same integration page by scrolling down to the "production" section.




The next section will now take you through various hosting scenarios.

## 5.3. PART 3: INTEGRATION OF YOUR API SERVICES

There are different ways in which you can integrate your API services in 3scale. Choose the one that best fits your needs:

- APIcast hosted on AWS
- APIcast hosted
- APIcast on OpenShift

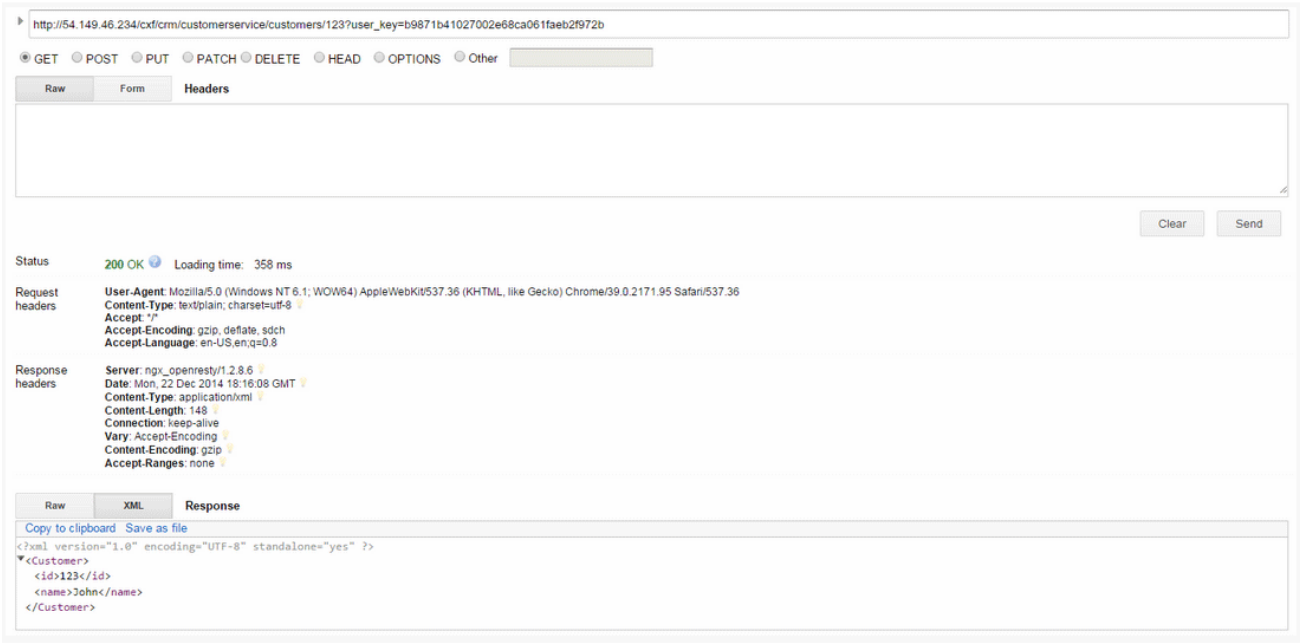
## 5.4. PART 4: TESTING THE API AND API MANAGEMENT

Testing the correct functioning of the API and the API Management is independent from the chosen scenario. You can use your favorite REST client and run the following commands.

### 5.4.1. Step 1

Retrieve the customer instance with id 123.

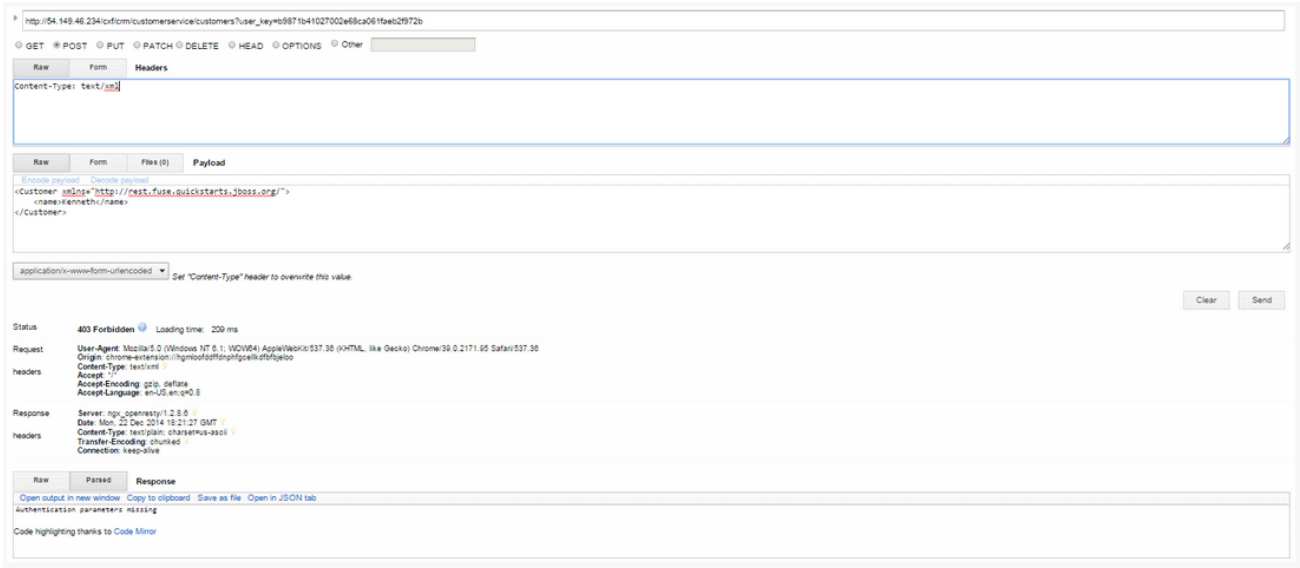
```
http://54.149.46.234/cxf/crm/customerservice/customers/123?
user_key=b9871b41027002e68ca061faeb2f972b
```




## 5.4.2. Step 2

Create a customer.

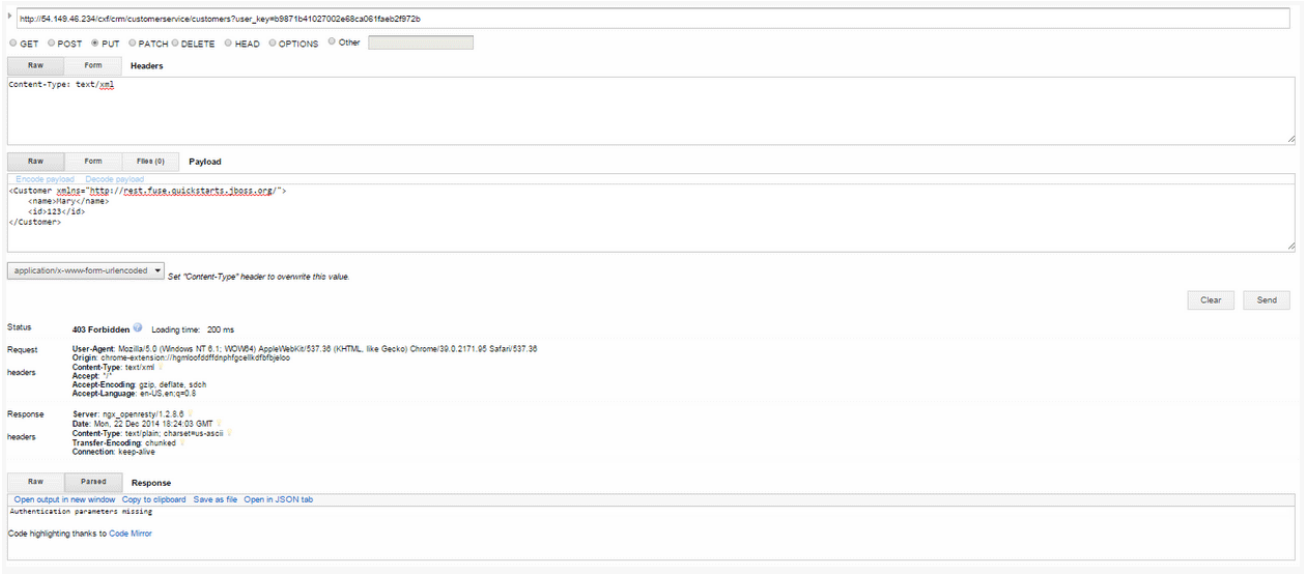
> http://54.149.46.234/cxf/crm/customerservice/customers?
> user_key=b9871b41027002e68ca061faeb2f972b




## 5.4.3. Step 3

Update the customer instance with id 123.

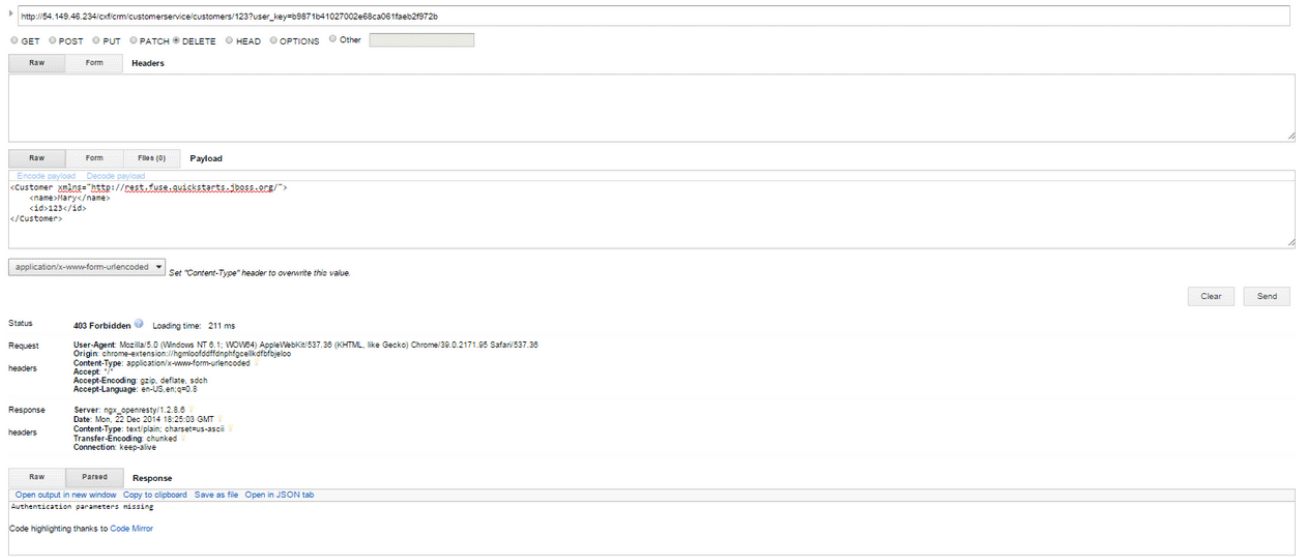
> http://54.149.46.234/cxf/crm/customerservice/customers?
> user_key=b9871b41027002e68ca061faeb2f972b




## 5.4.4. Step 4

Delete the customer instance with id 123.

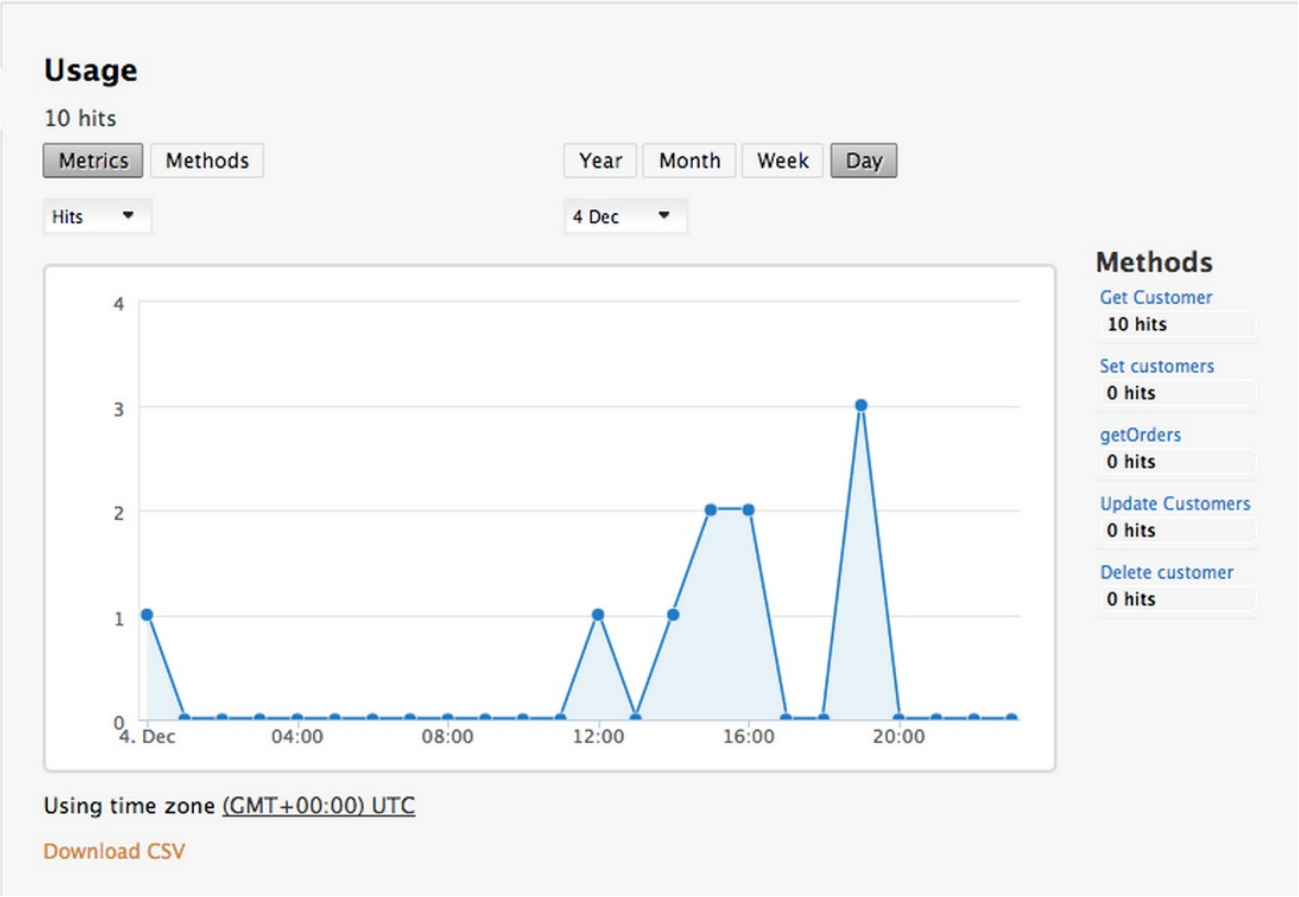
> http://54.149.46.234/cxf/crm/customerservice/customers/123?
> user_key=b9871b41027002e68ca061faeb2f972b




## 5.4.5. Step 5

Check the API Management analytics of your API.

If you now log back in to your 3scale account and go to Monitoring > Usage, you can see the various hits of the API endpoints represented as graphs.




This is just one element of API Management that brings you full visibility and control over your API. Other features include:

1. Access control
2. Usage policies and rate limits
3. Reporting
4. API documentation and developer portals
5. Monetization and billing

For more details about the specific API Management features and their benefits, please refer to the 3scale API Management Platform product description.

For more details about the specific Red Hat JBoss Fuse product features and their benefits, please refer to the JBOSS FUSE Overview.

For more details about running Red Hat JBoss Fuse on OpenShift, please refer to the Getting Started with JBoss Fuse on OpenShift.