Abstract

This reference architecture will demonstrate 3scale API Management Platform on-premises installation using API gateway secure JBoss Fuse API on OpenShift.
# Table of Contents

**COMMENTS AND FEEDBACK**  ................................................................. 3

**CHAPTER 1. EXECUTIVE SUMMARY**  ...................................................... 4

**CHAPTER 2. REFERENCE ARCHITECTURE ENVIRONMENT**  .......................... 5

**CHAPTER 3. INSTALL GUIDE**  .............................................................. 6

  3.1. OPENSHIFT LOGIN  .......................................................... 6

  3.2. ON-PREMISE INSTALLATION  ............................................... 6

  3.3. E-COMMERCE SERVICES  .................................................. 8

    3.3.1. Create Project  ................................................... 8

    3.3.2. Template Population  ........................................... 8

    3.3.3. Persistence Data Population  ................................... 8

    3.3.4. Set Secret Token  ................................................ 8

  3.4. LOG IN TO 3SCALE AMP ADMIN PORTAL  .................................. 8

  3.5. CREATE NEW ACTIVEDOCS  ............................................... 9

  3.6. CONFIGURE NEW APIS  .................................................. 11

    3.6.1. Create new APIs  ............................................... 11

    3.6.1.1. Add methods  ............................................... 13

    3.6.2. Delete default API  ........................................... 15

    3.6.3. Set up integration  ............................................ 16

    3.6.3.1. Create routes in OpenShift  .................................... 17

    3.6.3.2. Private Base URL  ....................................... 18

    3.6.3.3. Create Application Plans  ................................ 19

    3.6.3.4. Add mapping rules  ....................................... 20

    3.6.3.5. Set up rate limits for public plan  ....................... 21

  3.7. CREATE NEW ACCOUNTS  .................................................. 22

    3.7.1. Create Applications using the new APIs  .................... 24

  3.8. FINISH INTEGRATION  .................................................. 25

  3.9. DEVELOPER PORTAL SETUP  ........................................... 26

  3.10. DEPLOY THE PRESENTATION APPLICATION  .................................. 27

**CHAPTER 4. DESIGN AND DEVELOPMENT**  .......................................... 28

  4.1. USING DEVELOPER PORTAL  ........................................... 28

  4.2. ACCESSING THE WEB APPLICATION  .................................... 32

    4.2.1. Browser Access  ............................................. 32

    4.2.2. User Registration  ........................................... 33

  4.3. RATE LIMITS  .......................................................... 34

  4.4. ANALYTICS FEATURE  ................................................ 35

  4.5. ACCESS CONTROL USING USER_KEY  .................................... 37

  4.6. PREVENT BYPASSING API GATEWAY THROUGH SECRET TOKEN  ............ 39

  4.7. UPDATING SWAGGER FILES  .......................................... 41

  4.8. DEVELOPER PORTAL DOCUMENTATION PAGE  ............................ 43

**APPENDIX A. REVISION HISTORY**  .................................................. 44

**APPENDIX B. CONTRIBUTORS**  .......................................................... 45

**APPENDIX C. REVISION HISTORY**  .................................................. 46
In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.
CHAPTER 1. EXECUTIVE SUMMARY

APIs are the building blocks of the digital economy. Because they are key to seizing business value, you need a world-class API infrastructure that delivers now and into the future.

Red Hat® 3scale API Management Platform makes it easy to manage your APIs for internal or external users. Share, secure, distribute, control, and monetize your APIs on an infrastructure platform built with performance, customer control, and future growth in mind. And now, with the release of 3scale API Management Platform 2, you can place any 3scale components on-premise, in the cloud, or on any combination of the two.

Refer to the official Red Hat documentation for more introduction on API use cases.

This reference architecture describes the on-premise installation of Red Hat 3scale API Management Platform on OpenShift Container Platform and demonstrates important capabilities of the product by enhancing the functionality of an existing E-Commerce application through the introduction of an API gateway.
CHAPTER 2. REFERENCE ARCHITECTURE ENVIRONMENT

This reference architecture relies on a previous reference architecture application and environment thoroughly described in the paper titled *Building Microservices on OpenShift Container Platform with Fuse Integration Services*.

This environment installs Red Hat 3scale API Management Platform on OpenShift Container Platform and uses 3scale API Management Platform’s functions to control access to the Fuse gateway service, set rate limits on certain service calls, develop *ActiveDocs* from the *swagger* file and display them on *Developer Portal* of 3scale API Management Platform. A separately deployed Presentation application on OpenShift is used to make various calls to the Fuse gateway service through 3scale API Management Platform and demonstrate the end to end integration.

The configuration of 3scale API Management Platform for this reference architecture includes creating three APIs in the Admin Portal, based on the Fuse gateway service. These three APIs will be used by three different type of users:

- **Internal user**: Access to every API exposed by Fuse gateway service, including product, sales, billing and fulfillment.

- **Partner user (third party)**: Access is limited to only product calls, and exclusively to inquiry, add, delete and update of products.

- **Public user**: Access is limited to only product inquiry, and further limited to just 5 HTTP GET requests per minute

OpenShift Container Platform may be deployed with either a single, or three master hosts for this reference architecture. In both cases, it is assumed that *ocp-master1* refers to one (or the only) OpenShift master host and that the environment includes two OpenShift node hosts with the host names of *ocp-node1* and *ocp-node2*.

It is further assumed that OpenShift has been installed by the *root* user and that a regular user has been created with basic access to the host machine, as well as access to OpenShift through its *identity providers*.
CHAPTER 3. INSTALL GUIDE

3.1. OPENSOURCES LOGIN

Log in to the master host machine as the regular user, and use the `oc` utility to authenticate against OpenShift:

```
$ oc login -u ocuser --certificate-authority=/etc/origin/master/ca.crt
--server=https://apimgmt-master.apimgmt.example.com:8443
Authentication required for https://apimgmt-master.apimgmt.example.com:8443 (openshift)
Username: ocuser
Password: PASSWORD
Login successful.
```

Welcome! See 'oc help' to get started.

3.2. ON-PREMISE INSTALLATION

Please follow the On-premises Installation Guide to install 3scale API Management Platform (AMP) 2.0 on OpenShift.

Create a new project called 3scale-api.

```
oc new-project 3scale-api --display-name="3scale APM on OpenShift" --description="This project demonstrates the use of 3scale APM on OpenShift"
```

Next, download `amp.yml` and proceed with the installation. In the OpenShift environment used for this reference architecture, the wildcard domain is set up as `apimgmt.example.com`.

```
oc new-app --file amp.yml --param WILDCARD_DOMAIN=apimgmt.example.com
--> Deploying template "3scale-api/system" for "amp.yml" to project 3scale-api

system
--------
Login on https://3scale-admin.apimgmt.example.com as admin/bptp5gec
```

* With parameters:
  * AMP_RELEASE=2.0.0-CR2-redhat-1
  * ADMIN_PASSWORD=bptp5gec # generated
  * ADMIN_USERNAME=admin
  * APICAST_ACCESS_TOKEN=sjvkpnun # generated
  * ADMIN_ACCESS_TOKEN=kc2tdwob1wijaori # generated
  * WILDCARD_DOMAIN=apimgmt.example.com
  * TENANT_NAME=3scale
  * MySQL User=mysql
  * MySQL Password=ovixy777 # generated
  * MySQL Database Name=system
* MySQL Root password=hclunkck # generated
* SYSTEM_BACKEND_USERNAME=3scale_api_user
* SYSTEM_BACKEND_PASSWORD=ogboxf8d # generated
* REDIS_IMAGE=rhsc1/redis-32-rhel7:3.2-5.7
* MYSQL_IMAGE=rhsc1/mysql-56-rhel7:5.6-13.14
* SYSTEM_BACKEND_SHARED_SECRET=sip7ekg4 # generated
* SYSTEM_APP_SECRET_KEY_BASE=b0b2ec645aa60c2d47bebc817dada0dba73ea5183401185e0060728e7e0c4a52adb72a112cca22568084b502c2212dc78b8e213187dabeb747a0d55c8ec5c87 # generated
* APICAST_MANAGEMENT_API=status
* APICAST_OPENSSL_VERIFY=false
* APICAST_RESPONSE_CODES=true

--> Creating resources ...
persistentvolumeclaim "system-storage" created
persistentvolumeclaim "mysql-storage" created
persistentvolumeclaim "system-redis-storage" created
persistentvolumeclaim "backend-redis-storage" created
deploymentconfig "backend-cron" created
deploymentconfig "backend-redis" created
deploymentconfig "backend-listener" created
service "backend-redis" created
service "backend-listener" created
service "system-provider" created
service "system-developer" created
deploymentconfig "backend-worker" created
service "system-mysql" created
service "system-redis" created
deploymentconfig "system-redis" created
service "system-sphinx" created
deploymentconfig "system-sphinx" created
service "system-memcache" created
deploymentconfig "system-memcache" created
route "system-provider-admin-route" created
route "backend-route" created
route "system-developer-route" created
deploymentconfig "apiCast-staging" created
service "apiCast-staging" created
deploymentconfig "apiCast-production" created
service "apiCast-production" created
route "api-apiCast-staging-route" created
route "api-apiCast-production-route" created
deploymentconfig "system-app" created
deploymentconfig "system-resque" created
deploymentconfig "system-sidekiq" created
deploymentconfig "system-mysql" created
configmap "redis-config" created
configmap "smtp" created

--> Success
Run 'oc status' to view your app.

Please record the username (admin) and password, as these will later be used to log in into 3scale API Management Platform Admin Portal.
Below are the commands to reset and clean up, if you ever need to do a re-installation of the 3scale-api project.

```bash
oc delete all --all
oc delete configmap redis-config smtp
oc delete pvc system-storage mysql-storage system-redis-storage backend-redis-storage
```

### 3.3. E-COMMERCE SERVICES

#### 3.3.1. Create Project

Create the new project which will house the E-Commerce Services:

```bash
oc new-project ecom-services --display-name="E-Commerce Services Suite" --description="Back-end logic microservice suite"
```

#### 3.3.2. Template Population

Within the new project, execute the provided YAML template to configure and instantiate the full services suite:

```bash
oc new-app -f https://raw.githubusercontent.com/RHsyseng/3scale-apm/master/ecom-svcs/project-template.yaml
```

#### 3.3.3. Persistence Data Population

Once all services are deployed and running, instantiate the e-commerce data set via a GET request to the gateway service route:

```bash
$ curl -i http://ecom.rhmap.example.com/demo/testApi
HTTP/1.1 200 OK
...
```

#### 3.3.4. Set Secret Token

Then use the following command to set the Secret Token for the gateway service.

```bash
oc env dc/gateway-service -e GATEWAY_TOKEN_VALUE=Shared_secret_sent_from_proxy_to_API_backend_12345678987654321
```

### 3.4. LOG IN TO 3SCALE AMP ADMIN PORTAL

The console URL, username and the generated password are logged during the processing of the application template by OpenShift. Use this information to log in to the 3scale AMP console.

Login on https://3scale-admin.apimgmt.example.com as admin/bptp5gecn
### 3.5. CREATE NEW ACTIVEDOCS

ActiveDocs make API documentation user friendly, interactive, intuitive, and clear. With 3scale ActiveDocs, based on the Swagger Framework, developers can explore APIs live, straight from the documentation web pages.

To take advantage of this feature, it is best to set up ActiveDocs before further defining the API. Click on top menu item called API, then select ActiveDocs to bring up the ActiveDocs page. There is one demo service spec, called Echo, already defined.

Click the link to Create a new spec:

After filling out all the required fields and clicking the Update Service button, the exposed services can be viewed and tested on the next preview screen:
Click on each service to reveal details, including the schema and/or model for the request and response, and the newly added \textit{user_key}, used for 3scale API Management Platform’s authentication.

Use the following values to create three sets of ActiveDocs. For API JSON Spec content, use the corresponding swagger files from the \textit{github repository}:

**Service 1:**

- Name: Fuse-gateway-internal
- System Name: Fuse-gateway-internal

**Service 2:**

- Name: Fuse-gateway-partner
- System Name: Fuse-gateway-partner
3.6. CONFIGURE NEW APIs

3.6.1. Create new APIs

Used the defined ActiveDocs as references for creating services. Click on the APIs menu, and then click on Create Service.

Figure 3.1. Create Service

Enter Name and System Name, then press the Create Service button. For this reference architecture, the default configuration is valid; keep using APIcast as the gateway choice and API Key (user_key) as the authentication option.
Create three services as follows:

- Fuse-gateway-partner
- Fuse-gateway-public
- Fuse-gateway-internal

Use the following values:

**Service 1:**

Name: Fuse-gateway-internal
System Name: Fuse-gateway-internal
Service 2:
Name: Fuse-gateway-partner
System Name: Fuse-gateway-partner

Service 3:
Name: Fuse-gateway-public
System Name: Fuse-gateway-public

3.6.1.1. Add methods

The next step is to add methods for the newly added APIs. Click on the Definition link.

▼ Fuse-gateway-internal

Definition, Integration and Settings

Integrated through APIcast
Authenticated by API key
ID for API calls is Fuse-gateway-internal
Users can manage application keys
Users can manage applications
Users can request plan change
Users cannot select a plan when creating an application

Published Application Plans

There are no published application plans. Create one!

The methods are defined in the Definition page, click on the New method link.

Fuse-gateway-service > Definition

Methods
Add the methods of this API to get data on their individual usage. Method calls trigger the built-in metrics. Usage limits and pricing rules for individual methods are defined from within each Application Plan. A method needs to be mapped to one or more URL patterns in the Mapping Rules section of the Integration page so specific calls to your API up the count of specific methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>System Name</th>
<th>Unit</th>
<th>Description</th>
<th>Mapped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You have no methods

Enter appropriate values for Friendly name and System name.
Based on the Fuse gateway service ActiveDocs, create the following methods for these three APIs:

Fuse-gateway-internal API, no limit, with access to all services:

<table>
<thead>
<tr>
<th>Method</th>
<th>System Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>billing-get</td>
<td>billing-get</td>
</tr>
<tr>
<td>billing-post</td>
<td>billing-post</td>
</tr>
<tr>
<td>customer-get</td>
<td>customer-get</td>
</tr>
<tr>
<td>customer-post</td>
<td>customer-post</td>
</tr>
<tr>
<td>customer-put</td>
<td>customer-put</td>
</tr>
<tr>
<td>customer-patch</td>
<td>customer-patch</td>
</tr>
<tr>
<td>customer-delete</td>
<td>customer-delete</td>
</tr>
<tr>
<td>product-get</td>
<td>product-get</td>
</tr>
<tr>
<td>product-post</td>
<td>product-post</td>
</tr>
<tr>
<td>product-put</td>
<td>product-put</td>
</tr>
<tr>
<td>product-patch</td>
<td>product-patch</td>
</tr>
<tr>
<td>product-delete</td>
<td>product-delete</td>
</tr>
</tbody>
</table>

Fuse-gateway-partner API, limited to only to the product service:
3.6.2. Delete default API

Red Hat 3scale API Management Platform installation configures a default API called echo; delete this default API to make future configuration steps more clear, otherwise all new accounts created will automatically point to the default API's application plan.

- Click on the APIs menu and then click the API Definition link:

  ![API Definition](image)

  **Definition, Integration and Settings**
  
  On the Integration page, add your API base URL to the Private Base URL field in the staging box and hit Update & Test. Once the staging box is green, hit Deploy in the production box and you have completed a basic integration.

  Authenticated by API key
  ID for API calls is 2 and system name is api
  Users can manage application keys
  Users can manage applications
  Users can request plan change
  Users cannot select a plan when creating an application

- On the next screen, click the edit link:
3.6.3. Set up integration

Click on the APIs menu, and then click on Integration for each of the new services.

Fuse-gateway-internal

Definition, Integration and Settings

Integrated through APICAST
Authenticated by API key
ID for API calls is 6 and system name is Fuse-gateway-internal

Users can manage application keys
Users can manage applications
Users can request plan change
Users cannot select a plan when creating an application

Published Application Plans

There are no published application plans. Create one!

On the integration screen, click the button to add the base URL.
On this setup page, there are three base URLs:

**Fuse-gateway-service > Integration & Configuration**

Integration settings

- **Deployment Option:** APIcast
- **Authentication:** API Key (user_key)

To get started with this service on APIcast, add the base URL of your API and save the configuration.

On this setup page, there are three base URLs:

**Fuse-gateway-service > Integration**

**Configuration:** configure & test immediately in the staging environment [documentation](#)

<table>
<thead>
<tr>
<th>API</th>
<th>Private Base URL</th>
<th>Staging Public Base URL</th>
<th>Production Public Base URL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="https://echo-api.3scale.net:443">https://echo-api.3scale.net:443</a></td>
<td><a href="https://api">https://api</a> gast-ing-3scale-api-gateway-staging.middlewared.3scale-ee.com</td>
<td><a href="https://api-gateway-production.3scale-ee.com">https://api-gateway-production.3scale-ee.com</a></td>
</tr>
</tbody>
</table>

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

*Public address of your API gateway in the staging environment.*

*Public address of your API gateway in the production environment.*

**Staging Public Base URL and Production Public Base URL** are two important values that need to be recorded; these two addresses have to be added to the 3scale-api project's router in order for requests from clients to reach OpenShift's 3scale API gateway staging and production pod.

<table>
<thead>
<tr>
<th>po/apicast-production-1-3zv6c</th>
<th>1/1</th>
<th>Running</th>
<th>4</th>
<th>3d</th>
</tr>
</thead>
<tbody>
<tr>
<td>po/apicast-staging-1-9kf20</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>3d</td>
</tr>
</tbody>
</table>

**3.6.3.1. Create routes in OpenShift**

Use the route option of the oc utility to create all three routes for staging. Setting up the routes for production follows the same pattern.

```
oc expose service apicast-staging --hostname=fuse-gateway-public-3scale-apicast-staging.apimgmt.example.com --name=fuse-gateway-staging-public-route
```
oc expose service apicast-staging --hostname=fuse-gateway-partner-3scale-apicast-staging.apimgmt.example.com --name=fuse-gateway-staging-partner-route

oc expose service apicast-staging --hostname=fuse-gateway-internal-3scale-apicast-staging.apimgmt.example.com --name=fuse-gateway-staging-internal-route

Edit each new route to secure it:

oc edit route/fuse-gateway-staging-public-route
oc edit route/fuse-gateway-staging-partner-route
oc edit route/fuse-gateway-staging-internal-route

This is done by adding two lines to the spec that defines the TLS requirement:

```
spec:
  host: deploy.example.com
  port:
    targetPort: gateway
  tls: termination: edge
to:
  kind: Service
  name: apicast-staging
  weight: 100
  wildcardPolicy: None
```

To avoid creating one route for each service, check out the option to Configure Wildcard Domains. This feature is currently in tech preview and does not have full support yet.

### 3.6.3.2. Private Base URL

The Private Base URL is the address of the Fuse gateway service. Use CLI commands to get the IP address and port of the OpenShift service:

```
[czhu@apimgmt-master ~]$ oc projects
You have access to the following projects and can switch between them with 'oc project <projectname>':

* 3scale-api
default
ecom-services
kube-system
lambdaair
management-infra
openshift
openshift-infra

Using project "3scale-api" on server "https://apimgmt-master.apimgmt.example.com:8443".
```

```
[czhu@apimgmt-master ~]$ oc get svc -n ecom-services
NAME           CLUSTER-IP       EXTERNAL-IP
```
In our reference environment, http://172.30.124.155:9091 is the address of the service and this value can be used as Private Base URL for all three new APIs, since they all point to the same Fuse gateway service.

3.6.3.3. Create Application Plans

Click on the APIs menu, and then click on Create Application Plan for each of the services.

Fuse-gateway-service

Definition, Integration and Settings

On the Integration page, add your API base URL to the Private Base URL field in the staging box and hit Update & Test. Once the staging box is green, hit Deploy in the production box and you have completed a basic integration.

Published Application Plans

There are no published application plans. Create one!

You have 0 application plans (0 published) with a total of 0 live applications.

Use the following values to add four plans, including two plans for the Fuse-gateway-internal API:
Name:           desktop-plan  
System Name:    desktop-plan

Name:           mobile-plan  
System Name:    mobile-plan

One plan for the Fuse-gateway-partner API:

Name:           partner-plan  
System Name:    partner-plan

One plan for the Fuse-gateway-public API:

Name:           public-plan  
System Name:    public-plan

Also select a Default Plan for APIs. For example, two plans are defined for the Fuse-gateway-internal API. Select desktop-plan as the Default Plan, so that new applications created from this point on, automatically associate with the desktop-plan for this API.

3.6.3.4. Add mapping rules

Next, click the Add a mapping rule link to go to Integration page and add new mapping rules for the methods. This step is quite straight forward; just repeat it for all three APIs.
3.6.3.5. Set up rate limits for public plan

Set up *Rate Limits* for the public plan, so that anonymous public requests will not overload the infrastructure capacity. Click on Fuse-gateway-public API’s *Application Plans* link.

**Fuse-gateway-public**

*Definition, Integration and Settings*

- Integrated through APIcast
- Authenticated by API key
- ID for API calls is 5 and system name is *Fuse-gateway-public*
- Users can manage application keys
- Users can manage applications
- Users can request plan change
- Users cannot select a plan when creating an application

*Published Application Plans* [Create Application Plan]

There are no published application plans. Create one.

You have 1 application plan (0 published) with a total of 1 live application.

Then click on *New usage limit* for the *product-get* method and add a *Usage limit*. For example, set a maximum rate of 5 calls per minute:
3.7. CREATE NEW ACCOUNTS

Click on the top menu item, titled Developers.

A default developer by the name of John Doe has been created during installation; it is best to remove this developer to avoid future confusion.

Click on the Create link to navigate to the Create new Account page.

With each new user, should an existing API with an application plan exist, the system will create an application to point to them. The created application will select the first available API plan by default, which may not be appropriate, unless a Default Plan is already chosen for the API. Review the
corresponding screen and create the needed applications, once the right APIs are defined.

Click on the application name to display detailed information about the application, and make changes.

The API Credentials for this application is also displayed on this page:

```
API Credentials

User Key
032acee45c560dc6b510b0ab7d30be4d

Regenerate
Set Custom Key
```

Note the associated application plan:

```
Application Plan: desktop-plan
```

Use below values to create four accounts:

**Account 1:**
- Username: desktop-user
- Email: desktop-user@test.com
- Password: password
- Organization/Group Name: desktop-group

**Account 2:**
- Username: mobile-user
- Email: mobile-user@test.com
- Password: password
- Organization/Group Name: mobile-group

**Account 3:**
Username: partner-user
Email: partner-user@test.com
Password: password
Organization/Group Name: partner-group

Account 4:
Username: public-user
Email: public-user@test.com
Password: password
Organization/Group Name: public-group

3.7.1. Create Applications using the new APIs

This step is required to create an associated user key for each API. Click on the top menu item, titled Developers and choose one of the group defined earlier.

Click on Application, then click the Create Application link:

Select the right corresponding application plan and type an application name:

Repeat this step to create the following applications:
3.8. FINISH INTEGRATION

Finally, click on the top API menu item and from there, go to the Integration section of each defined API. Click to edit APIcast configuration and add two more values.

First, add `Shared_secret_sent_from_proxy_to_API_backend_12345678987654321` to Secret Token under AUTHENTICATION SETTINGS. For further explanation of this setting, refer to the next chapter.

Then, add `/products?featured=1` to the input field for API test Get request at the bottom of the page. Notice that the `curl` command will already include the proper `user_key` at the end of the URL. This is the new application's `user_key`.

Press the button to Update & test in staging Environment. If everything is set up properly, the test will go through. The automatically generated `curl` command can also be used in a shell environment to verify.

```
[czhu@apimgmt-master ~]$ curl "https://fuse-gateway-service-3scale-apicast-staging.ml888888888888888.example.com:443/products?featured=1&user_key=032acee45c569d6b5c6b14b7930be45"
```
3.9. DEVELOPER PORTAL SETUP

The final step is to set up the Developer Portal. The goal is to let users sign up in the developer portal and immediately get access to the APIs, so they can get started using the API interactive documentation (ActiveDocs, based on Swagger spec), with their own application key.

Click on the top menu item, titled Developer Portal, then select Documentation from the left side panel:

Replace the page source with the Documentation content found in the project repository, and save your changes.

Next, click on Homepage in the left side panel.
3.10. DEPLOY THE PRESENTATION APPLICATION

Deploy the Presentation service, which exposes a web tier for the desktop-user to access the application.

Create a new project for this service:

```
$ oc new-project desktop-project --display-name="Desktop project for Fuse ecom-services" --description="Desktop project for Fuse ecom-services"
```

Next, use OpenShift Source-To-Image (S2I) in this project to deploy the presentation service. The `presentation_template.yaml` file can be found in the project repository. Note that the `SERVICE_ADDRESS` is passed in as environment variables, while `user_key` is referenced as an OpenShift Secret in the template; the value for `user_key` is passed to the template as a parameter.

```
$ oc new-app -p USER_KEY_VALUE=9331104b50c0385cc64ebd72a38d4a56 -e SERVICE_ADDRESS=fuse-gateway-internal-3scale-apicast-staging.middleware.ocp.cloud.lab.eng.bos.redhat.com -f https://raw.githubusercontent.com/RHsyseng/3scale-apm/master/template/presentation_template.yaml
```
CHAPTER 4. DESIGN AND DEVELOPMENT

4.1. USING DEVELOPER PORTAL

To navigate to the Developer Portal, click on the top menu item titled Developer Portal and then select the Visit Developer Portal link on the next row. This will open the Developer Portal page in a new tab.

By default, the Developer Portal is not visible to the public. Users other than admin will be challenged for the Developer Portal Access Code.

Access code [ ] Enter

You can setup the site access code in your admin dashboard under ‘Settings > Developer Portal’.

Please make sure that you have cookies enabled!

This Access Code can be found in the 3scale API Management Platform Admin Portal. Click on the top menu item, titled Settings, and then choose Developer Portal.

Clearing the value for the Developer Portal Access Code makes the site public, and the address can then be bookmarked for quick access without going through the 3scale API Management Platform Admin Portal.

The Developer Portal allows registered users (desktop-user, mobile-user, partner-user and public-user) to access their own corresponding ActiveDocs. Anonymous users will see the following page, asking them to register and get an API key:
A further attempt by an anonymous user to access the documentation will result in the below page, prompting the user to sign in to see ActiveDocs, but also informing the user of the credentials for the public user account.

Registered users who have logged in will see a page displaying their API key (user_key):
Clicking the DOCUMENTATION link takes an authenticated user to a page that displays the ActiveDocs corresponding to their application plan. For desktop-user and mobile-user, as internal users, this means access to the Fuse-gateway-internal ActiveDoc defined in last chapter, which has access to all services exposed by Fuse gateway service:
For partner-user, access is limited to *Fuse-gateway-partner* ActiveDocs, which only exposes the product service.

For public-user, only access to *Fuse-gateway-public* ActiveDoc is provided, limiting such users to only product queries.
The ActiveDocs displayed on the DOCUMENTATION page accelerate development by allowing users to explore services, their parameters and responses.

4.2. ACCESSING THE WEB APPLICATION

The presentation application is accessible by the desktop-user and calls Fuse gateway service services. The desktop-user is associated with the internal plan, and this application also makes use of all services.

4.2.1. Browser Access
To run the web application, simply point your browser to the address exposed by the route. This address should ultimately resolve to the IP address of an OpenShift host where the router is deployed.

**Figure 4.1. Application Homepage before initialization**

Demo data is populated by hitting the demo.jsp page, for example at http://msa.example.com/demo.jsp. After initialization, this page will show the featured products:

**Figure 4.2. Application Homepage after initialization**

<table>
<thead>
<tr>
<th>ABC HD32CS5002 32-inch LED TV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Dimensions:</strong> 29 x 3 x 17</td>
</tr>
<tr>
<td><strong>Inches</strong></td>
</tr>
<tr>
<td>$249.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ABC HD42CS5002 42-inch LED TV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Dimensions:</strong> 37 x 2 x 22</td>
</tr>
<tr>
<td><strong>Inches</strong></td>
</tr>
<tr>
<td>$424.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microtech MM-733N Microwave Oven; 1.6 Cubic Feet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Dimensions:</strong> 22 x 19 x 12</td>
</tr>
<tr>
<td><strong>Inches</strong></td>
</tr>
<tr>
<td>$178.0</td>
</tr>
</tbody>
</table>

### 4.2.2. User Registration

Anonymous users are constrained to browsing inventory, viewing featured products and searching the catalog. To use other features that result in calls to the Sales and Billing services, a valid customer must be logged in.

To register a customer and log in, click on the Register button in the top-right corner of the screen and fill out the registration form:
After registration, the purchase button allows customers to add items to their shopping cart, to subsequently visit the shopping cart and check out, and review their order history.

4.3. RATE LIMITS

Rate Limit is set up for the public plan. The imposed constraint is a maximum of five requests in every one minute.

To see the effect of this Rate Limit, use the curl command example from the Fuse-gateway-public API integration page.

Click on APIs, choose Fuse-gateway-public, click on Integration link, then click edit APICast configuration link.

The curl command is at the bottom.

Execute this command multiple times in succession to exceed the limit and a message of Authentication failed will appear instead of the response data.
4.4. ANALYTICS FEATURE

Click on the top menu item titled Analytics to view the Monitoring page, containing usage data for all three APIs.

Click on an API name to view more details about it. For example, clicking on Fuse-gateway-public should reveal a spike that results from the multiple curl command invocations intended to the rate limit in the previous section.
The usage graph for Fuse-gateway-internal will show multiple calls for different services. These service calls were triggered by the web application on different Fuse gateway service operations.

Note that registered users can view associated and accessible Analytics data from the Developer Portal. For example, after signing in to the Developer Portal as the desktop-user, click on STATISTICS to view a graph similar to the one displayed below:
4.5. ACCESS CONTROL USING USER_KEY

This reference architecture uses API Key (user_key) for authentication.

Each application has its own user_key. The keys can be found in respective application pages.

API Credentials

User Key

032ceee45c5d6dc6b510b0ab7d30be4d

Regenerate

Set Custom Key

Each calling application uses the assigned user_key, which corresponds to an API's application plan. The screenshot below shows an example of an HTTP GET call:
The `user_key` and base URL should not be hard-coded in the application source code, as they are prone to later changes. One good example is the use of OpenShift Secret to managing the `user_key` in OpenShift, which is referenced in the deployment of the presentation service, for example:

The `user_key` value is stored in an OpenShift `Secret`, which is defined in the `presentation_template.yaml` file:

```yaml
- apiVersion: v1
  kind: Secret
  metadata:
    name: 3scale-secret
    namespace: desktop-project
  stringData:
    user_key: ${USER_KEY_VALUE}
```

Further down in the `presentation_template.yaml` file, content from the `Secret` is referenced and made available as an environment variable.

```yaml
spec:
  containers:
    - env:
      - name: USER_KEY
        valueFrom:
          secretKeyRef:
            name: 3scale-secret
            key: user_key
```

The actual value for `user_key` is passed in as a parameter during deployment. The use of a parameter instead of an environment variable has the advantage that parameters won’t be kept in memory, so credentials won’t be exposed. Since the base URL is not sensitive information, it can be passed in as an environment variable during pod creation.

```
$ oc new-app -p USER_KEY_VALUE=9331104b50c0385cc64ebd72a38d4a56 -e SERVICE_ADDRESS=fuse-gateway-internal-3scale-apicast-staging.middleware.ocp.cloud.lab.eng.bos.redhat.com --file=presentation_template.yaml
```

Then `RestClient.java` code in the service can then retrieve both data as environment variable values and use them in the web address:

```java
public class RestClient {
    public static String userKey = System.getenv("USER_KEY");
    public static String serviceAddress = System.getenv("SERVICE_ADDRESS");
```
For troubleshooting purpose, this reference architecture’s presentation layer uses the JavaScript console to log the two variables’ runtime value. This allows users to verify the actual values. To see these two values in Google Chrome, go to setting, More tools, Developer Tools, and click Console.

4.6. PREVENT BYPASSING API GATEWAY THROUGH SECRET TOKEN

The network topology is often effective in blocking unauthorized access to internal services. When Red Hat 3scale API Management Platform is used to introduce a gateway, direct access to the underlying API is often blocked through network tools. In cases where such configuration is impossible or insufficient, the secret token feature of 3scale API Management Platform can be used to achieve similar results.

When enabled, 3scale API Management Platform hard codes a token in the HTTP header of all requests passing through the gateway. Applications exposing the managed API would then have to check for the existence of this token and reject any requests that do not contain it. As long as the value of this token is kept private, the only way to access such API would be through the gateway.

There are two steps to turn this feature on.

First, in 3scale API Management Platform Admin Portal, click on the top menu item titled API, then choose Integration for the API that needs to be secured. Click the AUTHENTICATION SETTINGS drop-down list and give Secret Token a value.

Second, in the application code for the managed API, check all request for a token name of X-3scale-proxy-secret-token in the HTTP header. Reject any requests that do not provide a value for this token, or have an incorrect value.

For example, the following code snippet from the Fuse gateway service shows an example of how to enforce the Secret Token.

```java
@Value('${gateway.token.header:X-3scale-proxy-secret-token}')
String tokenHeader
```
@Value('${gateway.token.value:#{null}}')
String tokenValue

Logger logger = Logger.getLogger(AppRoute.class.name)

@Override
void configure() throws Exception {
    if (tokenValue != null && tokenHeader != null) {
        logger.info("AUTH TOKEN REQUIREMENT DETECTED: 
[${tokenHeader}] adding security route interceptor")
        interceptFrom().id("auth token interceptor").process(new Processor() {
            @Override
            void process(Exchange exchange) throws Exception {
                if (exchange.in.getHeader(tokenHeader) == null ||
                    exchange.in.getHeader(tokenHeader) !=
                    tokenValue) {
                    logger.info("Authorization token required, but 
header missing or invalid")
                    exchange.out.setHeader(Exchange.HTTP_RESPONSE_CODE, 403)
                    exchange.out.setBody("Unauthorized [missing or
invalid token"])
                    exchange.setProperty(Exchange.ROUTE_STOP, Boolean.TRUE)
                }
            }
        });
    }
}

The secret token in inject into the Fuse gateway service pod as an environment variable to avoid hard-coding it.
In this example, the token value is set to 
*Shared_secret_sent_from_proxy_to_API_backend_12345678987654321*. Changing this value in 
the 3scale API Management Platform Admin Portal without updating it in the Fuse gateway service 
code would result in the API integration test failing, since the value passed from 3scale API 
Management Platform to Fuse gateway service will not match.

To demonstrate this, try the following `curl` commands to the same Fuse gateway service API, 
where the first one bypasses the 3scale API Management Platform and uses the Fuse gateway 
service gateway IP address directly, while the second request goes through 3scale API 
Management Platform as expected.

```

curl "https://fuse-gateway-internal-3scale-apicast-
staging.middlereap.cloud.lab.eng.bos.redhat.com:443/products?
?featured=1&user_key=876286f7c3cf24899c6f390464c41429"
```

The first requests fails with a message of *missing or invalid token*, while the second command 
returns the expected response.

4.7. UPDATING SWAGGER FILES

To create a new service spec, the swagger file for the new service is required. The reference 
architecture’s Fuse gateway service provided a swagger file at *http://api.example.com/apidoc/swagger.json* 
after installation. Several updates are required to this file before it can be used in 
this project:

- Update the host to the correct value. Red Hat 3scale API Management Platform doesn’t allow IP 
  address, so please use either the *Staging Public Base URL* or the *Production Public Base URL* 
  instead:

```
"host" : "fuse-gateway-service-3scale-apicast-staging.apimgmt.example.com:443",
```
Update the scheme to HTTPS, as it is the default protocol used by 3scale API Management Platform:

```
"schemes" : [ "https" ],
```

For each service, add one more parameter called `user_key` for authentication. This parameter is not included in the Swagger file originally provided by the Fuse gateway service.

Note that only one parameter called `customerId` is included in the file:

```
"/customers/{customerId}" : {
  "get" : {
    "tags" : [ "customers/{customerId}" ],
    "summary" : "get customer",
    "produces" : [ "application/json" ],
    "parameters" : [ {
      "name" : "customerId",
      "in" : "path",
      "description" : "id of customer to fetch",
      "required" : true,
      "type" : "string"
    } ],
    "responses" : {
      "200" : {
        "description" : "customer fetched",
        "schema" : {
          "$ref" : "#/definitions/Customer"
        }
      }
    },
    "x-camelContextId" : "camel-1",
    "x-routeId" : "route7"
  },
}
```

Add the new parameter:

```
"/customers/{customerId}" : {
  "get" : {
    "tags" : [ "customers/{customerId}" ],
    "summary" : "get customer",
    "produces" : [ "application/json" ],
    "parameters" : [ {
      "name" : "customerId",
      "in" : "path",
      "description" : "id of customer to fetch",
      "required" : true,
      "type" : "string"
    } ],
    { 
      "name" : "user_key",
      "in" : "query",
      "description" : "Your API access key",
      "required" : true,
      "x-data-threescale-name" : "user_keys",
      "type" : "string"
    }
  },
}
```
4.8. DEVELOPER PORTAL DOCUMENTATION PAGE

**Liquid** is a simple programming language used for displaying and processing most of the data from the 3scale system available for API providers.

Use Liquid control flow tags **if** and **elsif** to limit access for each registered user to their own ActiveDocs:

{% if current_user.username == 'desktop-user' or current_user.username == 'mobile-user' %}
{%- active_docs version: "2.0" services: "Fuse-gateway-internal" %}
.............
{% elif current_user.username == 'partner-user' %}
{%- active_docs version: "2.0" services: "Fuse-gateway-partner" %}
.............
{% elif current_user.username == 'public-user' %}
{%- active_docs version: "2.0" services: "Fuse-gateway-public" %}
.............
{% endif %}

Refer to the product [documentation](#) on how to publish ActiveDocs in the Developer Portal.

```javascript
$(function () {
    window.swaggerUi.options['docExpansion'] = 'none';
    window.swaggerUi.options['url'] = "https://3scale.apimgmt.example.com/swagger/spec/Fuse-gateway-internal.json";
    window.swaggerUi.load();
});
```

For more information on using Liquid and Developer Portal, please refer to [Liquids: Developer Portal](#) and the [Liquid Reference](#) pages.
## APPENDIX A. REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Release Date</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>August 2017</td>
<td>Calvin Zhu</td>
</tr>
</tbody>
</table>
## APPENDIX B. CONTRIBUTORS

We would like to thank the following individuals for their time and patience as we collaborated on this process. This document would not have been possible without their many contributions.

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Title</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Babak Mozaffari</td>
<td>Manager, Software Engineering &amp; Consulting Engineer</td>
<td>Technical Content Review</td>
</tr>
<tr>
<td>Jeremy Ary</td>
<td>Senior Software Engineer</td>
<td>Technical Content</td>
</tr>
<tr>
<td>Andrew Mackenzie</td>
<td>Director, Software Engineering</td>
<td>Technical Content Review</td>
</tr>
<tr>
<td>Vanessa Ramos</td>
<td>Senior Product Manager</td>
<td>Technical Content Review</td>
</tr>
<tr>
<td>Hugo Guerrero</td>
<td>Senior Principal Product Marketing Manager</td>
<td>Technical Content Review</td>
</tr>
</tbody>
</table>
APPENDIX C. REVISION HISTORY

| Revision 1.0-0 | August 2017 | CZ |