

# Preparing to deliver the workshop

Instructor requirements:

- Being very comfortable with 3scale both in terms of configuration and of operations.
- Knowing the basics of the Developer Portal
- Being comfortable with RH SSO and especially around OpenID Connect protocol
- Being able to use Postman (or similar REST client) to build REST request and REST authentication
- Having a basic knowledge of OpenShift

Attendees requirements:

- To use the GUI of OpenShift: [https://docs.openshift.com/container-platform/3.11/architecture/infrastructure\\_components/web\\_console.html#browser-requirements](https://docs.openshift.com/container-platform/3.11/architecture/infrastructure_components/web_console.html#browser-requirements)
- Basic knowledge of REST protocol
- Basic knowledge of containers
- Basic understanding of OAuth social applications

GUID variable value will be provided to you by the instructors and will need to be exchanged everywhere in the URL provided.

Environment:

Environment reachable here:

<https://tutorial-web-app-webapp.apps.open-banking-GUID.openshiftworkshop.com>

End Users credentials:

user[1..100] / openshift

3scale dedicated instances here [X is your user id]:

<https://userX-admin.apps.open-banking-GUID.openshiftworkshop.com>

access using username and password above

dedicated gateway

<https://userX.amp.apps.open-banking-GUID.openshiftworkshop.com>

## Duration and format

This is a workshop designed for approximately 2 hours delivery. The focus is on the adoption of the products to build a comprehensive financial solution. This is not an introduction to the different products used in the open banking solution portal, but it is focused on the modules and functionalities relevant for the FSI industry. Typically, instructors would talk through the introduction slides, and then for each hands-on lab, explain the steps needed to achieve the objective of the lab calling on the main functionalities of the products. At the end of each activity there will be a checkpoint with the attendees.

## Timeline

30 minutes	Architecture and key features
30 minutes	<i>Start of Practical Part 1</i> Financial Backend service demo and building blocks on Fuse Online [hands-on of the attendees in parallel]
25 minutes	Basics of Service Configuration and Integration in 3scale. Basics of Developer portal content publishing [hands-on of the attendees in parallel]
5 minutes	<b>CHECKPOINT</b>
15 minutes	<b>BREAK</b>
10 minutes	<i>Start of Practical Part 2</i> Introducing OIDC & OAuth [slides]
30 minutes	Basics of RH SSO and 3scale OIDC API authentication [hands-on of the attendees in parallel]
5 minutes	<b>CHECKPOINT</b>
10 minutes	(optional according to timing) OpenShift high level walkthrough [simple demo]
5 minutes	<b>Q&amp;A and closing</b>

## Practical Part 1

### Integr8ly

Login into the integr8ly environment main page

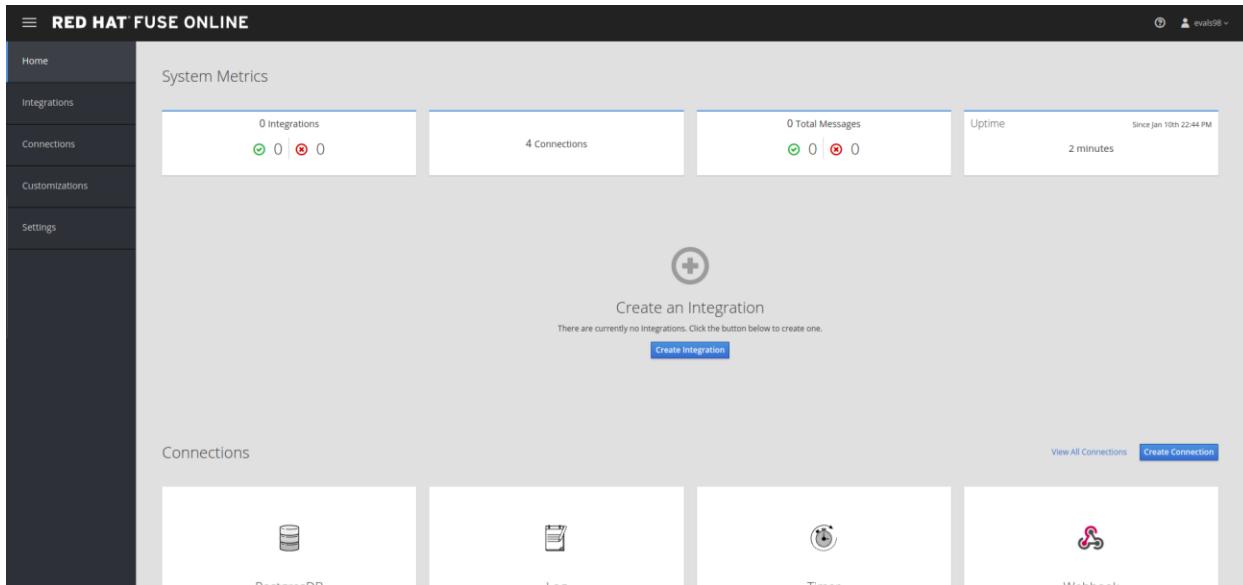
<https://tutorial-web-app-webapp.apps.open-banking-GUID.openshiftworkshop.com>

The screenshot shows the Red Hat Solution Explorer interface. On the left, there's a sidebar titled 'Start with a walkthrough' containing three items: 'Integrating event-driven and API-driven applications (AMQ)', 'Integrating event-driven and API-driven applications (EnMasse)', and 'Integrating API-driven applications'. Each item has a 'Get Started' button and a duration (15 min or 21 min). On the right, there's a section titled 'Applications' listing various Red Hat products: Red Hat OpenShift, Red Hat 3scale API Management Platform, Red Hat AMQ, Eclipse Che, EnMasse, Red Hat Fuse, and Red Hat Developer Launcher. Each product entry includes a status indicator ('Ready for use') and a 'preview' button.

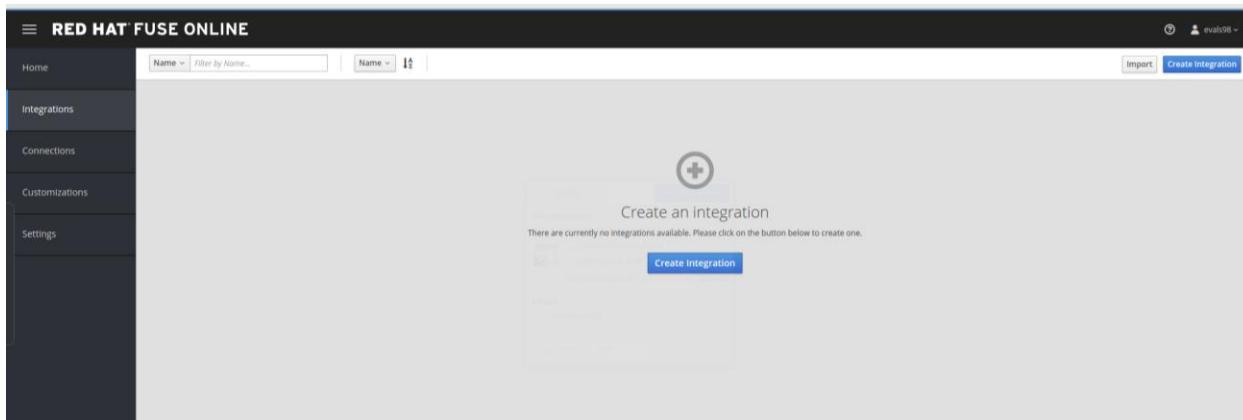
We will see how integr8ly environment works and what you get. It provides out of the box integration between products but it doesn't change the base components that come with it. You can install yourself this type of environment starting from a clean or empty OpenShift installation.

### Fuse Online

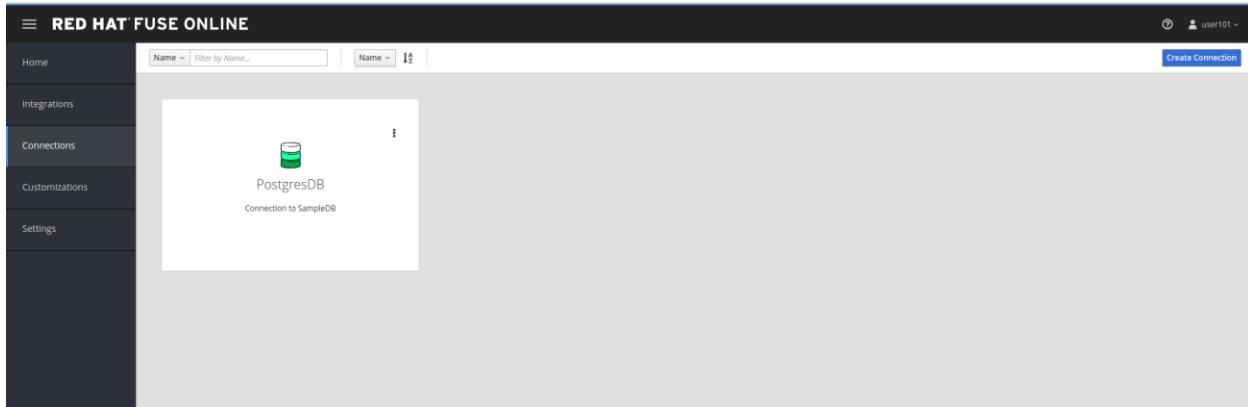
Let's start our first lab session, by opening Red Hat Fuse Online. You will get there by clicking [Red Hat Fuse Online](#) link in the tutorial dashboard. When using Fuse Online for the first time, you will have to authorize access by clicking on "Allow selected permissions" and having both permissions checkboxes selected. We will see the main functionalities of it and then use it to build a simple integration block.



The main dashboard (**Home**) is the landing page and it is especially important when you need to check messages coming from the requests being sent and in general to give an overview of the deployed integration blocks.

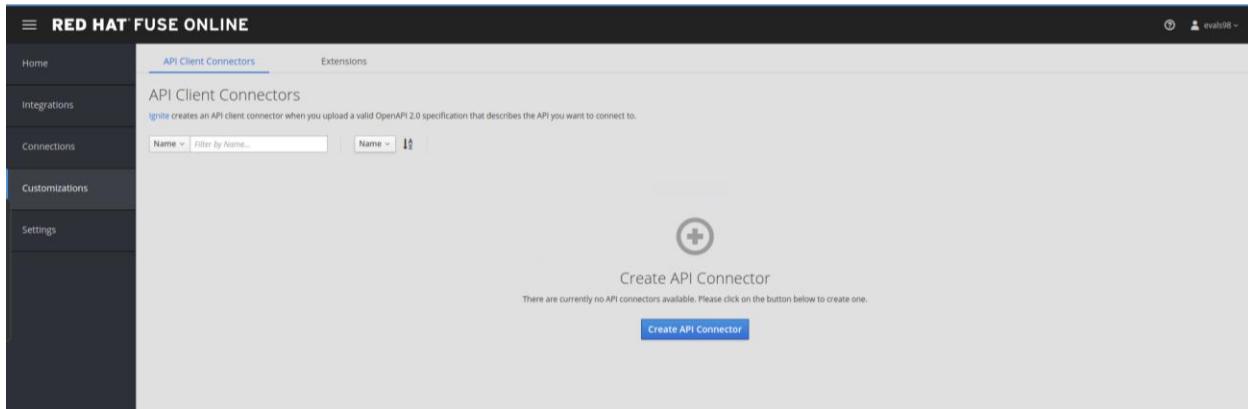


The integration window (**Integrations**) allows you to have a graphical representation of the mediation/transformation and is the first tool to onboard the citizen developers. Behind the curtains there is Camel (which is a proven technology and deployed in highly transactional environment as well) and although the connectors available right now are just few the number will grow dramatically as this is the repository <http://camel.apache.org/components.html> .

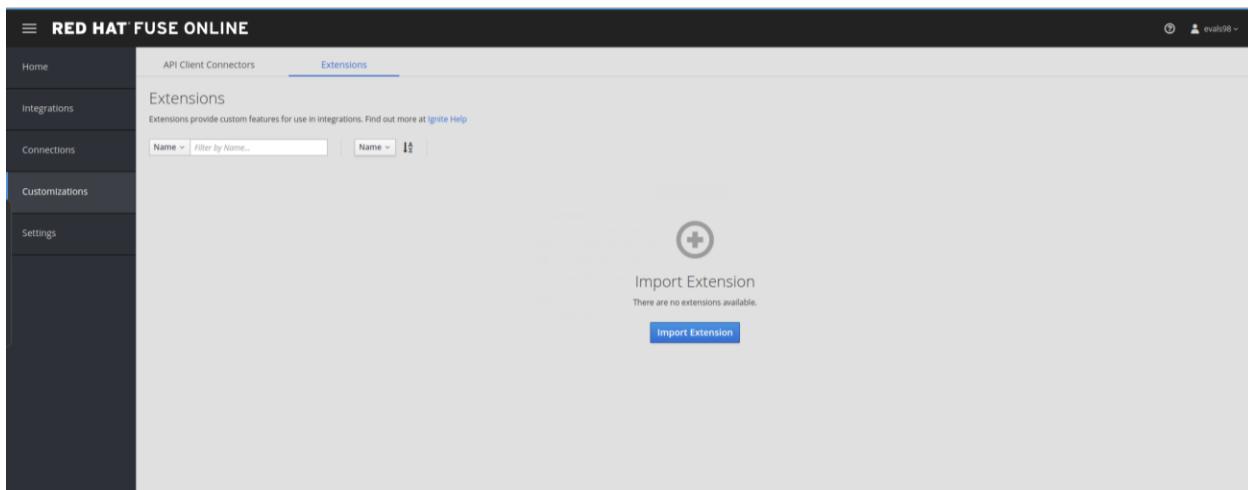


(Connections) you can start or end (or sometimes interpose) connections which are fully configured connectors. We will be showing the configuration of one connector during today's lab.

Various connectors are available and new connectors are coming in the future and will be ported onto Fuse Online.



(Customizations) there are two panes available here, the first one dedicated to API connections, which we will be using afterwards as Connection in the Integration.



(Extensions panel) this is the part where you can extend the functionalities of the platform. You can compile a custom or community available extension to perform a specific functionality

or connect to a specific system and in this easy way add transformation functionalities to the product.

## LAB BEGINS

### Let's start INTEGRATING now!

This first session will take us to the creation of a simple REST service from a database source. This is quite a common scenario and one that goes well for quick prototyping service scenarios. Let's start by creating the connection to an existing DB that I previously created. It is a DBaaS built on PostgreSQL and hosted on this environment. You could be running this anywhere else, as long as you have a public direct connection to the DB to use.

#### Connections -> Create connection

The screenshot shows two views of the Red Hat Fuse Online interface. The top view is a summary page with a single connection named 'PostgresDB' listed. The bottom view is a detailed 'Create Connection' wizard:

- Step 1: Select Connector**: Shows a grid of connector options:
  - Amazon S3: Retrieve and store objects.
  - AMQ Message Broker: Subscribe for and publish messages.
  - AMQP Message Broker: Subscribe for and publish messages.
  - Database: Subscribe for and publish messages.
  - Technology Preview: Includes options for CloudWatch Metrics, CloudWatch Metrics Insights, and CloudWatch Metrics Insights Metrics Processor.
- Step 2: Configure Connection**: Not visible in the screenshot.
- Step 3: Name Connection**: Not visible in the screenshot.

A callout at the bottom left points to the 'Database' connector icon with the text "Select Database connector".

The fields marked with \* are required.

- \* Connection URL: jdbc:postgresql://manny.db.elephantsql.com:5432/hxuazymr
- \* Username: hxuazymr
- \* Password:
- Schema: hxuazymr

Validate

Use the following connection details and validate them:

Connection details: jdbc:postgresql://postgreSQL.shared.svc:5432/sampledb

username: dbuser

password: password

Database has been successfully validated.

The fields marked with \* are required.

- \* Connection URL: jdbc:postgresql://manny.db.elephantsql.com:5432/hxuazymr
- \* Username: hxuazymr
- \* Password:
- Schema: hxuazymr

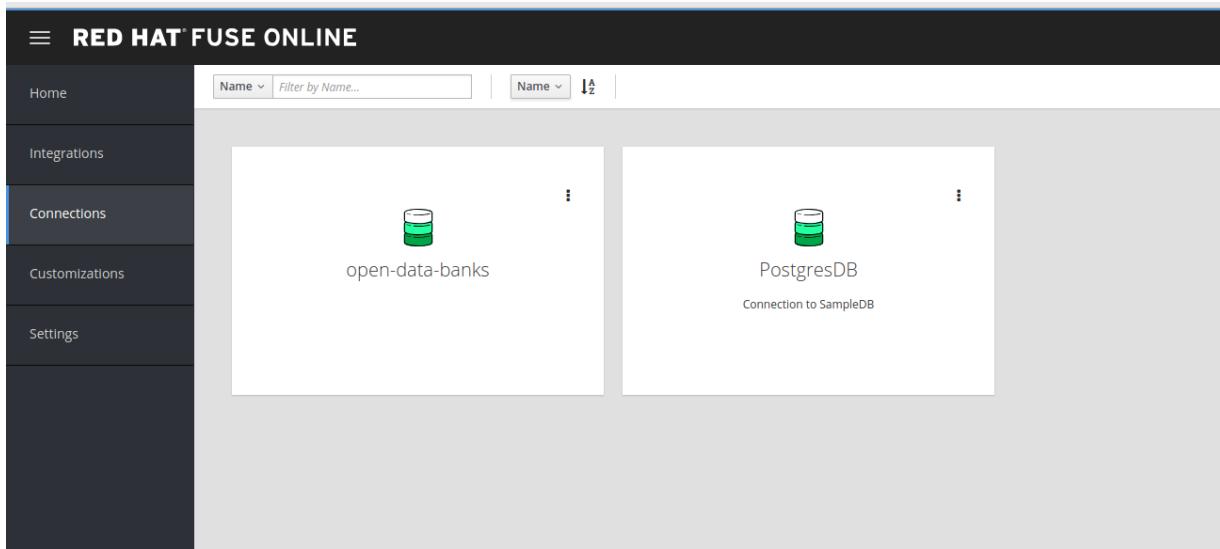
Validate

Let's name the connection and finalize the configuration of the connector ([Create](#)).

The fields marked with \* are required.

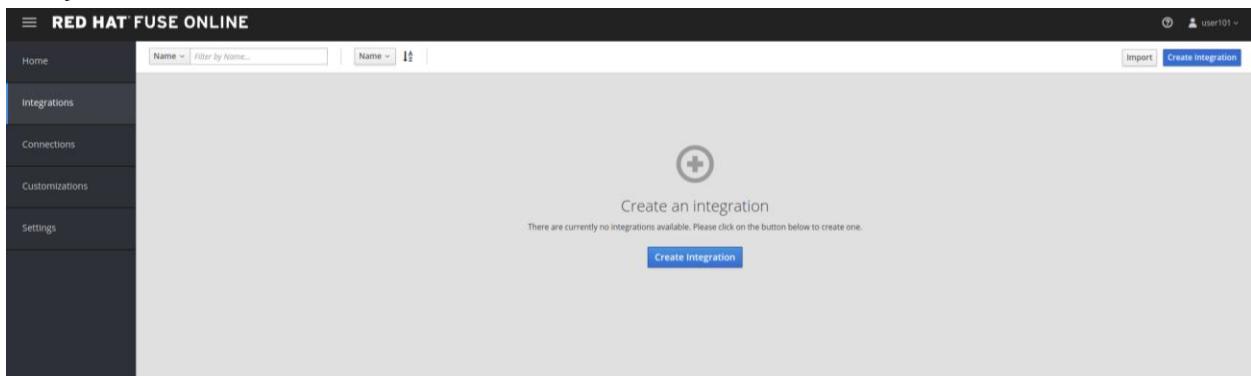
- \* Connection Name: open-data-banks
- Description:

Create



We can now use this connection as an integration starting, middle or finishing point.

Now that we have configured the end of the integration path we want to build and we will see where to find the start and we will put the two pieces together. **Integrations -> create integration**



We will start by exposing a REST endpoint that will then get mapped to the backend datasource.

Use the [API Provider](#) connector with the following OpenAPI file:

<https://raw.githubusercontent.com/lucamaf/open-banking-roadshow/master/open-data-apis-nokey.json>

**RED HAT FUSE ONLINE**

Enter Int...

Home > Integrations > New Integration > Choose a Start Connection

Choose a Start Connection  
Click the connection that starts the integration. If the connection you need is not available, click Create Connection.

Name Filter by Name... Name

Technology Preview

API Provider  
Expose Restful APIs

open-data-banks

PostgresDB  
Connection to SampleDB

Webhook  
Create direct connections with external systems th...

Create Connection

**RED HAT FUSE ONLINE**

Enter Int...

Home > Integrations > New Integration > Start Integration with an API call

Start integration with an API call  
Execute this integration when a client invokes an operation defined by this API.

Upload an OpenAPI file  
Drag and drop your OpenAPI file here or No file chosen

Accepted file type: json, yaml and yml

Use a URL  
 Create from scratch

**Next** **Cancel**

The definition is the same one seen on the Open Banking solution portal for Open Data APIs. There is a validation happening on the API definition, but no error was identified so we can proceed with the configuration of the connector.

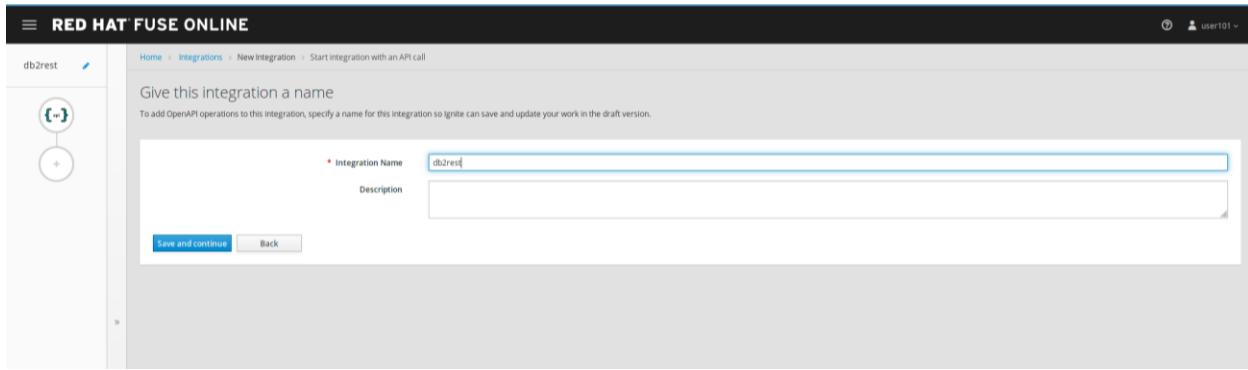
To show how easy it is to correct definitions -> **review/edit**

This opens a window on Apicurito which is a scaled down version of Apicurio, our API Design platform. It is fairly easy to change elements graphically and also with the help of this tool an API team can start with a Design First approach when configuring the API. Also the same team doesn't need to know about the rules around OpenAPI specifications thanks to this tool.

The service is exposed without any protection (that's one of the reasons we are going to be using 3scale later on) since the API definition is not adding any authentication on top of it by default.

Name the integration.

And save and continue



We are going to map just one of the endpoints exposed, in particular the *get banks* (/banks) one.

Operation	Method	Description
get atm info	GET /banks/(bank_id)/atms(atm_id)	501 Not Implemented
get bank atms	GET /banks/(bank_id)/atms	501 Not Implemented
get bank details	GET /banks/(bank_id)	501 Not Implemented
get branch info	GET /banks/(bank_id)/branches/(branch_id)	501 Not Implemented
get branches available by a specific bank	GET /banks/(bank_id)/branches	501 Not Implemented
get list of banks	GET /banks	501 Not Implemented
get product info	GET /banks/(bank_id)/products/(product_id)	501 Not Implemented
get products	GET /banks/(bank_id)/products	501 Not Implemented

Click on get list of banks

We now have a dumb pipe which connects an endpoint to receive user requests and return always 200 (all OK) - the return blue arrow symbol. Not very useful integration but a starter! Let's connect this front end to the database we previously configured as a terminating connection.

**Add a connection**

The screenshot shows the Red Hat Fuse Online interface. The left sidebar has a tree view with 'db2rest' selected. The main content area is titled 'Choose a Connection' with the sub-tutorial 'Click the connection that completes the integration. If the connection you need is not available, click Create Connection.' Below this are two connection options: 'Log' (represented by a notepad icon) and 'open-data-banks' (represented by a database icon). A third option, 'PostgreSQL', is partially visible on the right.

Click on the previously configured data source.

The screenshot shows the Red Hat Fuse Online interface after selecting 'open-data-banks'. The left sidebar now shows 'get list of bank accounts' under 'db2rest'. The main content area is titled 'open-data-banks - Choose an Action' with the sub-tutorial 'Choose an action for the selected connection.' Below this are two action options: 'Invoke SQL' (represented by a database icon) and 'Invoke stored procedure' (represented by a gear icon). The 'Invoke SQL' button is highlighted in blue.

And we will invoke a simple SQL statement to return the data from a single table.

The simple statement to introduce is:  
`select * from banks`

When you click **done** the statement will validate and you should be able to proceed with the configuration of the integration.

And now let's add a simple log of the requests coming through. Add a step.

Select the log step.

The screenshot shows the 'Choose a Step' dialog in the Red Hat Fuse Online interface. The dialog title is 'Choose a Step' and it says 'A step specifies an operation on the data in the integration. Click one to add it.' There are four options listed:

- Advanced Filter**: Continue the integration only if criteria you define in scripting language expressions are met.
- Basic Filter**: Continue the integration only if criteria you specify in simple input fields are met. Suitable for most integrations.
- Log**: Sends a message to the integration's log. This option is selected and highlighted in blue.
- Template**: Upload or create a Mustache template to define consistent output data.

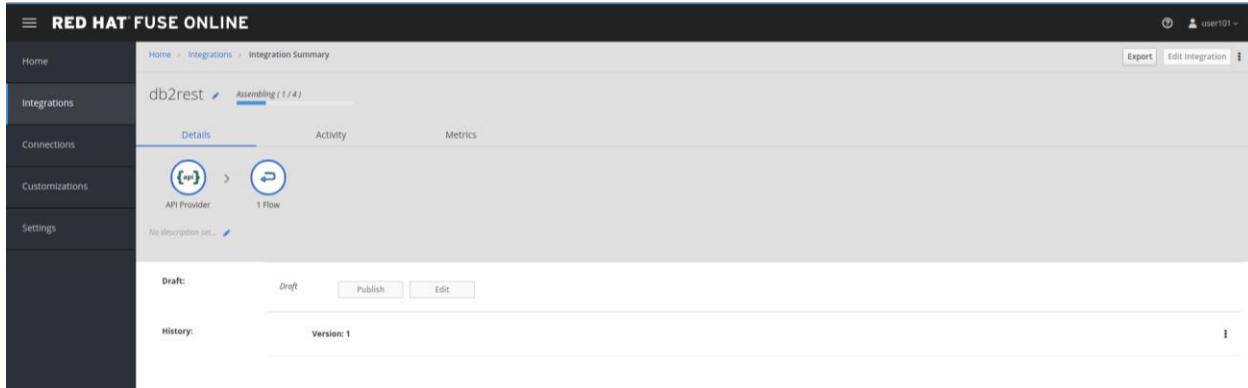
We are going to be sending a copy of the responses coming through to the integration log.

The screenshot shows the 'Configure Log' dialog in the Red Hat Fuse Online interface. The dialog title is 'Configure Log' and it says 'Fill out the fields associated with the selected step.' There are two options under 'Custom Text': 'Message Context' (unchecked) and 'Message Body' (checked). The 'Custom Text' field is empty. At the bottom are 'Cancel' and 'Done' buttons.

We are going to log just the message body.

The screenshot shows the main integration builder interface in the Red Hat Fuse Online platform. On the left, there is a vertical toolbar with icons for different step types: { } (Script), + (Add Step), and various connection icons. The main workspace has a large plus sign icon in the center, with the text 'Add to Integration' above it. Below the plus sign, it says 'Now you can add additional connections as well as steps to your integration.' and 'You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom of the workspace are 'Add a Step' and 'Add a Connection' buttons. The top navigation bar shows the path 'Home > Integrations > db2rest > get list of banks' and includes buttons for 'Cancel', 'Go to Operation List', 'Save as Draft', and 'Publish'.

We are now ready to deploy and expose this integration in our platform, to use it. Hit [Publish](#)



You can check the progress in building the integration changing through phases. We can notice the platform is getting the required components and constructing the block. When the building is completed we can test the Integration block.

*SINCE AUTO DISCOVERY FEATURE IS ACTIVE WE WILL NOT GET AUTOMATICALLY A URL WITH THE INTEGRATION BUILDING PROCESS, BUT API MANAGEMENT WILL BE ABLE TO SEE IT AND EXPOSE IT IN ANY CASE*

Now that we have seen how to build the full integration you can test the integration just built, using this online tool <https://apitester.com/>. You can use the following URL to test it <http://i-db2rest.apps.open-banking-GUID.openshiftworkshop.com/open-data/banks>

API tester works as a full Web or Mobile application but stripped down of the GUI.

## Build your test

[View example](#)Click  to add or remove steps[Collapse / Expand](#)

Request no auth financial service

GET

1 https://open-data.b9ad.pro-us-east-1.openshiftapps.com/open-data/banks

Headers + Add Request Header

+ Add Step

 Saving tests to your account allows you to:

- Rerun this test
- Share test results with others
- View the run history
- Create and re-run multiple tests

© 2019 RIGOR. ALL RIGHTS RESERVED

Powered by **Populate the fields with the following URL****http://i-db2rest.apps.open-banking-GUID.openshiftworkshop.com/open-data/banks**  
**and hit Test**

 **PASS**

N. Virginia  
Seconds elapsed: 11

Results 6.29 s Viewing a Request Step 1 < >

Message [Step 1] GET http://i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com/open-data/banks passed

Connection Information >

Request Request Headers

```
GET /open-data/banks HTTP/1.1
Host: i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
```

Response Response Headers

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
X-Application-Context: application
Date: Mon, 21 Jan 2019 18:20:33 GMT
Set-Cookie: 12b28c7c44ae5989665ef8abec492fd0=08271c1b765ada49ab2c2c05b8b72917; path=/; HttpOnly
Cache-control: private
```

Response Body

```
{"short_name": "Bank X", "id": 1, "address": "psd201-bank-x--uk", "long_name": "The Bank of X"}
```

**Show that everything went 200 fine and open the Response Body (eye icon)**

The screenshot shows a browser window with the URL <https://uptime-diagnostics-response-bodies-production.s3.amazonaws.com/578a-6>. The page content is a JSON object:

```
{"short_name": "Bank X", "id": 1, "address": "psd201-bank-x--uk", "long_name": "The Bank of X"}
```

The response is in JSON format, with basic information given around banks (dummy data).

## 3scale

To reach 3scale as an admin user you can just go back to the main integr8ly tutorial dashboard and click on [3scale admin dashboard link](#) (something like this <https://userX-admin.apps.open-banking-GUID.openshiftworkshop.com/>). You will be presented with the login window and can use you previously used user credentials.

Introducing now 3scale, for the purpose of managing these exposed APIs, securing them and tracking their usage. **Dashboard**

Everybody will be logged in act as an administrator or API provider. The dashboard will show a summary of the trends in usage of the platform, in terms of developers signups, usage of APIs and message sent and received.

The screenshot shows the Red Hat 3scale API Management dashboard. At the top, it displays:

- RED HAT 3SCALE API MANAGEMENT
- Dashboard
- Signed in successfully

**AUDIENCE** section:

- 1 ACCOUNT
- 1 APPLICATION
- BILLING
- DEVELOPER PORTAL (0 DRAFTS)
- 0 MESSAGES

**Potential Upgrades** section (last 30 days):

- 0 today
- Accounts that hit their Usage Limits in the last 30 days
- In order to show Potential Upgrades, add 1 or more usage limits to your [Application Plans](#).
- Furthermore, [Web Alerts for Admins of this Account of 100% \(and up\)](#) should be enabled for service(s) with usage limits.

**TODAY** section:

- The sound of silence

**BEFORE TODAY** section:

- No news, good news.

**APIS** section:

- NEW API

**API** section:

- OVERVIEW
- ANALYTICS
- 1 APPLICATION
- 1 ACTIVEDOC
- INTEGRATE THIS API

**Top Applications** section (last 30 days):

- 0 today
- Apps with consistently high traffic in the last 30 days
- In order to show Top Applications you need to have at least one application sending traffic to the API.
- Consider [making some test calls](#) from the Integration page to get a feel for what you'd see here.

Let's start managing and protecting the API we just created on Fuse Online.

Click on the green button **New API** and select **Import from OpenShift** (click on **Authenticate to enable this option**). We are going to be using the [auto-discovery](#) feature we saw before. We would be helped by this feature and would save time configuring the service.

The screenshot shows the 'New API' creation page. At the top, there's a navigation bar with the Red Hat 3SCALE logo, 'RED HAT 3SCALE API MANAGEMENT', and a 'Dashboard' dropdown. Below the navigation, the title 'New API' is displayed. There are two radio button options: 'Define manually' (selected) and 'Import from OpenShift'. A note below the second option says '(Authenticate to enable this option)'. The main form area is labeled 'SERVICE'. It contains three fields: 'Name' (a text input field), 'System name' (a text input field with a note 'Only ASCII letters, numbers, dashes and underscores are allowed.'), and 'Description' (a large text area). The entire form is set against a light gray background.

You are now presented with the permissions screen and you should click the *Allow selected permissions* button to proceed. This is to allow API management to go and look for services in our container platform.

#### Authorize Access

Bscale is requesting permission to access your account (user101)

##### Requested permissions

###### user:full

Full read/write access with all of your permissions  
Includes any access you have to escalating resources like secrets

You will be redirected to [https://master.apps.openbanking-4f6a.openshiftworkshop.com/auth/service-discovery/callback?referrer=/api/config/services/new&self\\_domain=user101-admin.apps.openbanking-4f6a.openshiftworkshop.com](https://master.apps.openbanking-4f6a.openshiftworkshop.com/auth/service-discovery/callback?referrer=/api/config/services/new&self_domain=user101-admin.apps.openbanking-4f6a.openshiftworkshop.com)

[Allow selected permissions](#)

[Deny](#)

You will now see the auto-discovered Fuse service appear in the *Namespace* field. Click the blue **Create Service** button and proceed in starting the service configuration.

New API

Define manually  
 Import from OpenShift

SERVICE

Namespace

Name

**Create Service**

You will be redirected to the Dashboard view where a success message should appear as shown in the following screenshot.

The service will be imported shortly. You will receive a notification when it is done.

**AUDIENCE**

1 Signups  
last 30 days

**POTENTIAL UPGRADES**  
today | 0 Accounts that hit their Usage Limits in the last 30 days

In order to show Potential Upgrades, add 1 or more usage limits to your Application Plans.

Furthermore, Web Alerts for Admins of this Account of 100% (and up) should be enabled for service(s) with usage limits.

**DEVELOPER PORTAL (0 DRAFTS)**

**MESSAGES (0)**

**TODAY**

- Test created on API
- Application Test has been deleted
- Test created on API
- Application Test has been deleted
- Test created on API
- Application Developer's App has been deleted

**BEFORE TODAY**

The sound of silence

Once the service is created you will see that in the dropdown menu where you can also see Audience.

We can now proceed on changing the details of the configuration of the API and publish the update on the Developer Portal so that the public Developers can sign up for the open financial API.

**API-> Integration-> Configuration**

The screenshot shows the 'Integration' section of the APIcast configuration interface. On the left is a sidebar with links like Overview, Analytics, Applications, ActiveDocs, Integration (Configuration, Methods & Metrics, Settings), and Support. The main area is titled 'Configuration' and contains a 'Integration settings' section. It shows 'Deployment Option' set to 'APIcast' and 'Authentication' set to 'API Key (user\_key)'. Below this is a note: 'To get started with this service on APIcast, add the base URL of your API and save the configuration.' A blue button labeled 'add the base URL of your API and save the configuration.' is visible. In the top right corner, there's a link 'edit integration settings'.

This is where you configure the rest of the details of the mapped and protected Service. The private base URL should already be filled with the details coming from the auto-discovery feature.

The screenshot shows a dark header bar with 'API MANAGEMENT' on the left and a dropdown menu 'API: i-db2rest' on the right. This indicates the specific API being configured.

## Integration

Configure the staging environment [documentation](#)

The screenshot shows two configuration sections. The first, under 'API', has a 'Private Base URL' input field containing 'http://i-db2rest.fuse-06b94c72-2572-11e9-9425-0a580a010007.svc.cluster.local:8080'. A 'Use Echo API' link is next to it. Below is a note: 'Private address of your API that will be called by the API gateway. For end-to-end encryption your private base URL scheme should be https.' The second section, under 'API GATEWAY', has a 'Staging Public Base URL' input field containing 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443'. A note below says 'Public address of your API gateway in the staging environment. Make sure to [add the correct route](#) [Use Default URL](#)'. Below is a 'Production Public Base URL' input field containing 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443'. A note below says 'Public address of your API gateway in the production environment. Make sure to [add the correct route](#) [Use Default URL](#)'.

We have a section where we map the Backend API (or in this case Integration service) and then 2 URLs where we expose the managed API on the staging first and production gateways or infrastructure. We will be changing the Staging and Public addresses of the gateway. In this case we are not going to use separate staging or public infrastructure so it can be the same address (please notice the format of the staging and public base url for the gateways <https://userX.amp.apps.open-banking-GUID.openshiftworkshop.com>)

Scheme should be https.

API GATEWAY

Staging Public Base URL  
https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443  
Public address of your API gateway in the staging environment. Make sure to [add the correct route](#) [Use Default URL](#)

Production Public Base URL  
https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443  
[Use Default URL](#)  
Public address of your API gateway in the production environment. Make sure to [add the correct route](#)

MAPPING RULES

Verb	Pattern	+	Metric or Method (Define)
GET	/open-data/banks\$	1	hits

[Add Mapping Rule](#)

AUTHENTICATION SETTINGS

We will now make sure we map a single endpoint or resource in 3scale and disallow any other endpoint (i.e. the other endpoints have not been implemented yet so we are protecting them from being exposed).

The endpoint you want to map is /open-data/banks\$ (notice the \$ at the end of the path that will allow us to make sure users cannot improperly access any other sub-resource). We can now check and change the configuration of authentication settings.

APPLICATIONS

ActiveDocs

Integration

Configuration

Methods & Metrics

Settings

AUTHENTICATION SETTINGS

Host Header

Let's define a custom Host request header. This is needed if your API backend only accepts traffic from a specific host.

Secret Token

Shared\_secret\_sent\_from\_proxy\_to\_API\_backend\_5c9f6c92c06c2efa

Enables you to block any direct developer requests to your API backend; each 3scale API gateway call to your API backend contains a request header called X-3scale-proxy-secret-token. The value of this header can be set by you here. It's up to you ensure your backend only allows calls with this secret header.

Credentials location

As HTTP Headers

As query parameters (GET) or body parameters (POST/PUT/DELETE)

Auth user key

key

ERRORS

We see that we have already api key protection enabled, but we might want to pass this information as HTTP Header instead of HTTP parameter. We will also change the header name to 'key'

The screenshot shows the 3scale APIcast configuration interface. On the left is a sidebar with navigation links: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main area displays two error configurations:

- AUTHENTICATION FAILED ERROR**
  - Response Code: 403
  - Content-type: text/plain; charset=us-ascii
  - Response Body: Authentication failed
- AUTHENTICATION MISSING ERROR**
  - Response Code: 403
  - Content-type: text/plain; charset=us-ascii
  - Response Body: Authentication parameters missing

You can also customize the error returned to the end user. **Policies**

The screenshot shows the Policies section of the 3scale APIcast interface. The sidebar on the left remains the same. The main area shows:

- POLICIES**: A list of policies, with "3scale APICAST" selected. It includes a "Policy Chain" button and an "Add Policy" button.
- CLIENT**: A section for testing the client. It shows an "API test GET request" input field containing a curl command:
 

```
curl "https://api-user1-apicast-staging.apps.openbanking-4f6a.openshiftworkshop.com:443/" -H
'key: 177e86c845f6e642c756fb4f39697adb'
```

At the bottom, there is a note: "Hit the test button to check the connections between client, gateway & API." and two buttons: "Update & test in Staging Environment" and "Back to Integration & Configuration".

These are like additional plugin that you can configure to adapt the service to your own preference. **Add Policy**

The screenshot shows the 3scale APIcast configuration interface. On the left, there's a sidebar with navigation links: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main area is titled 'POLICIES' and contains a list of available policies:

- OAuth 2.0 Token Introspection**: builtin - Configures OAuth 2.0 Token Introspection.
- Conditional policy [Tech preview]**: builtin - Executes a policy chain conditionally.
- Upstream**: builtin - Allows to modify the upstream URL of the request based on its ...
- Anonymous access**: builtin - Provides default credentials for unauthenticated requests.
- URL rewriting with captures**: builtin - Captures arguments in a URL and rewrites the URL using them.
- Logging**: builtin - Controls logging.
- SOAP**: builtin - Adds support for a small subset of SOAP.
- Header modification**: builtin - Allows to include custom headers.
- 3scale batcher**: builtin - Caches auths from 3scale backend and batches reports.
- Edge limiting**: builtin - Adds rate limit.

Several policies are available and the list is always increasing. Two policies / functionalities are of importance: SOAP policy to map SOAP services and advanced rate limit functionalities with edge limiting policy. Update the API test GET request field with the same pattern you are mapping above (excluding the \$).

Hitting the big blue button will allow you to do two things at once:

- Update the service configuration on the platform
- Test the configuration just uploaded to the gateway.

The second one will fail since we are not providing any valid key, so we will get unauthorized request but the gateway will receive the updated configuration in any case.

The screenshot shows the 3scale APIcast configuration interface. The top part shows the 'POLICIES' section with a single policy listed:

Policy Chain	
<b>3scale APIcast</b> builtin - Main functionality of APIcast to work with the 3scale API m...	

The bottom part is titled 'CLIENT' and contains a 'API test GET request' field with the value '/'. A note below it says: 'Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:'. A red warning message follows: 'In order to send a valid test request, please create an application that subscribes to an application plan of this service. Start with creating an application plan.' Below this is a code snippet:

```
curl "https://api-user101-apicast-staging.amp.apps.open-banking-2a38.openshiftworkshop.com:443" -H"user_key: USER_KEY"
```

A red error message at the bottom states: 'Something is not quite ok. Is your private API host reachable from the API gateway?'. At the bottom right are buttons for 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

We will now fix the test request error as advised by the warning message.

Let's switch to explaining the role of API contracts of Application Plans.

Within the red error message a link is generated “[create application plan](#)”. Click this link and you will be redirected to the **create application plan** form.

Fill out the **Name** and **System Name** fields on the [create application plan](#) form and click the blue button to submit the form.

You can safely ignore for now the monetization options and use whichever name you prefer.

The screenshot shows the 'Create Application Plan' form. On the left is a dark sidebar with navigation links: Overview, Analytics, Applications (selected), Listing, Application Plans (selected), ActiveDocs, and Integration. The main area is titled 'Create Application Plan'. It contains fields for 'Name' and 'System name', both of which are currently empty. There is a note below the system name field stating "Only ASCII letters, numbers, dashes and underscores are allowed." Below these are two checkboxes: 'Applications require approval?' (unchecked) and 'Set whether or not applications can be created on demand or if approval is required from you before they are activated.' Underneath are fields for 'Trial Period (days)', 'Setup fee' (0.00 USD), and 'Cost per month' (0.00 USD). At the bottom right is a blue 'Create Application Plan' button.

The screenshot shows the 'Application Plans' page. The sidebar is identical to the previous screenshot. The main area is titled 'Application Plans' and contains a paragraph about what Application Plans are. Below is a table with one row:

Name	Applications	State	Actions		
Basic	0	hidden	Publish	Copy	Delete

A blue 'Create Application Plan' button is located at the top right of the table area.

We see that we have 1 API contract (or Application Plan), but no application associated with it. The application plans are in **hidden** state by default, so let's publish this one so that it is usable and visible on the Developer portal. Let's open the application plan.

**Application Plan Basic**

Name: Basic

System name: basic

Applications require approval?  
Set whether or not applications can be created on demand or if approval is required from you before they are activated.

Trial Period (days):

Setup fee: 0.00 USD

Cost per month: 0.00 USD

**Update Application plan**

### Main elements:

- Monetization settings (trial, setup, cost per month)
- Endpoint mapped (in this case generic Hits) and relative monetization and rate limiting settings

Metric or Method (Define)	Enabled ?	Visible ?	Text only ?
Hits	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Metrics, Methods, Limits & Pricing Rules**

Metric or Method (Define)	Pricing (0)	Limits (0)	Enabled?	Visible?	Text only?
Hits	<input type="button" value="Pricing (0)"/>	<input type="button" value="Limits (0)"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Features**

Name	Description	Enabled?	New feature?
Unlimited Greetings		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
24/7 support		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
Unlimited calls		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete

We can now switch to the Audience tab to create an Application to test the Configuration, by clicking on Listing.

**Accounts**

Group/Org.	Admin	Signup Date	Apps	State
Developer	John Doe	29 Jan, 2019	3	Approved

Search for Group/Org., User login, First name, Last name, email, user\_key, app\_id or App.

From here we can see how we can, as Provider, approve or deny Developers' Accounts registrations. Let's click on the default **Developer** Account

The screenshot shows the developer account details page. On the left, a sidebar navigation includes 'Accounts' (selected), 'Listing', 'SETTINGS', 'Usage Rules', 'Fields Definitions', 'Applications' (selected), 'Billing', 'Developer Portal', and 'Messages'. The main content area has a header 'Account: Developer' with an 'Edit' link. It displays the following information:

- Organization/Group Name:** Developer  Send message
- Administrator:** John Doe ([admin+test@user101.com](mailto:admin+test@user101.com))
- Signed up on:** January 18, 2019 23:47
- Status:** Approved

**Billing Status:** Monthly billing is enabled. Disable

**Application:** Name: test, Service: API, Plan: Basic, State: Live. Hits 0 hits

We can see that the Developer has the default application associated, but it's subscribed to the default Service. We can also see the Developer user details.

Let's click on Applications in the top level navigation and Create application

The screenshot shows the 'New Application' creation form. The sidebar navigation is identical to the previous screen. The main form includes the following fields:

Name	State	Plan	Paid?	Created At	Traffic On	<input checked="" type="radio"/> Create Application
<input type="text"/>	All	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Search"/>

**New Application**

Application plan: Basic

Service plan: Default

Name:

Description:

Here we can now subscribe the application to the **Application plan** we created on our new Service from the drop down field available. Let's fill in the rest of the fields with some basic details and click the big blue button: Create Application.

We now have an assigned key so we can go back to the Configuration window of the API service and make a successful test call. **API -> Integration -> edit Apicast configuration**

The screenshot shows the 'Integration & Configuration' section of the OpenShift web console. In the 'CLIENT' tab, there is a 'Policy Chain' section containing a single policy named '3scale APICAST'. Below it, there is a 'Test' section for an 'API test GET request' to '/open-data/banks'. The curl command is pre-populated with the key value: 'curl "https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks" -H'key: 13ece073549b898fb089a9493b4c1cae''. A note below the curl command says: 'Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:'. At the bottom right, there are buttons for 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

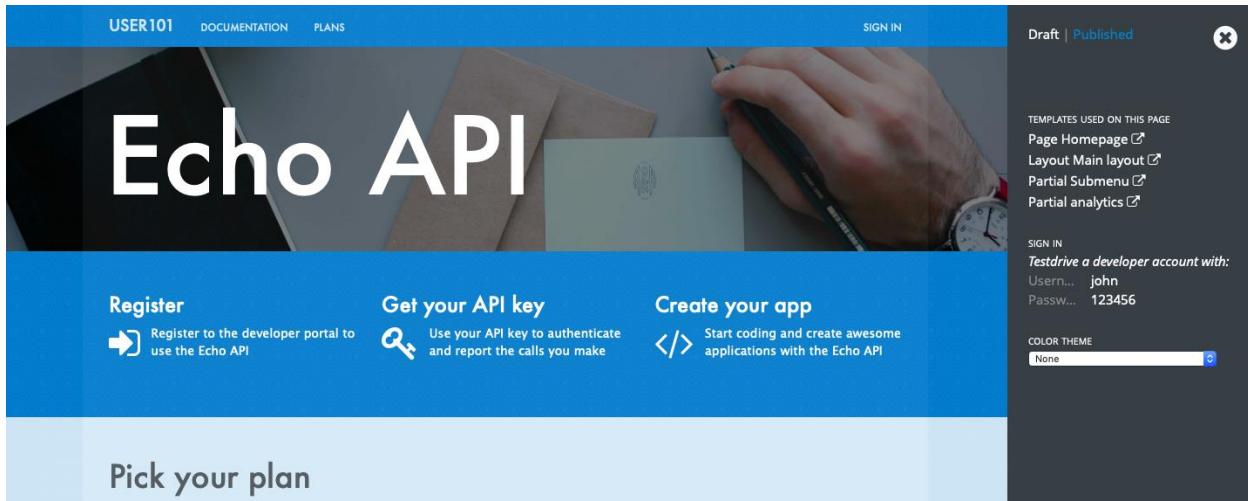
We now have a pre-populated key in the example curl statement, let's try again testing the deployed configuration.

This screenshot is identical to the one above, showing the 'Integration & Configuration' interface. It displays the same 'CLIENT' tab with the '3scale APICAST' policy and the pre-populated curl command. The note about the optional GET request is also present. At the bottom right, there are buttons for 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

As we can see we turned the testing into a success.

Let's switch to the developers' point of view by accessing the Developer portal. You can access it by selecting in the top menu in **Audience -> Developer portal -> Visit Portal**

The sidebar allows us to edit pages of the Developer Portal live, but we are not interested in it now so we can close it.



Let's sign in with the default user credentials provided in the sidebar. This is the default developer user, created for the default developer account [john / 123456]

The screenshot shows the Echo API developer portal after signing in. The top navigation bar now includes 'MESSAGES', 'SETTINGS', and a user icon. A message 'SIGNED IN SUCCESSFULLY' is displayed. The main content area features the 'Echo API' banner and a section titled 'Your API Key' with the subtext: 'This is your API key that should be kept secret. Use it to authenticate and report the calls you make to the Echo API.' To the right, a modal window titled 'CREDENTIALS' shows an 'App name' of 'Test' and a 'Key' of 'af44089cd553a3b0515772972b5f9ced'. It also contains the instruction: 'Add this as a `user_key` parameter to your API calls to authenticate.'

We are now logged in the developer's dashboard. Let's see the Applications I have created

I can now use the credential that I have associated with the application and test the protected service. Let's move to the online API testing tool, <https://apitester.com/>

Use the URL for your API gateway, the following format should be configured in your service already: `https://userX.amp.apps.open-banking-GUID.openshiftworkshop.com`, remember the key Header and the associated value.

**PASS** N. Virginia  
Seconds elapsed: 2

Results 576 ms Viewing a Request Step 1 < >

Message [Step 1] key financial service passed

Connection Information

Request

Request Headers

```
GET /open-data/banks HTTP/1.1
Host: wt2-evals98-example-com-3scale.apps.open-banking.opentry.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: 4c571db87a82b5aedccbd8877627a090
```

Response

Response Headers

```
HTTP/1.1 200 OK
Server: openresty/1.13.6.1
Date: Fri, 11 Jan 2019 18:04:18 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5078
Set-Cookie: JSESSIONID=yo9rav2w644k1uns81kk5fiuy;Path=/
Expires: Fri, 11 Jan 2019 18:04:18 GMT
```

As we can see we succeed with 200 OK!

Let's now just test with a wrong key or path then to confirm the role of API Management.

Connection Information

Request

Request Headers

```
GET /open-data/banks HTTP/1.1
Host: wt2-evals98-example-com-3scale.apps.open-banking.opentry.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: c571db87a82b5aedccbd8877627a090
```

Response

Response Headers

```
HTTP/1.1 403 Forbidden
Server: openresty/1.13.6.1
Date: Fri, 11 Jan 2019 18:16:42 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: chunked
Set-Cookie: 6ec552533d9895c2bc361d784adc8ddb=fd85f9e6e21ee8a00f6cad673c7c9e5b; path=/; HttpOnly; Secure
```

Response Body

As expected we receive a Forbidden error.

## Checkpoint



## Break

## Practical Part 2

### RH SSO and 3SCALE OIDC

Let's now improve the security of the managed integration service with OIDC. API key is not really considered a safe API authentication protocol anymore and it is vulnerable to many attacks.

After introducing content around OAuth and OIDC, let's see the main elements of RH SSO itself.

### LAB BEGINS

Let's start with RH SSO main dashboard (HTTP reachable one)

<http://sso-http-sso.apps.open-banking-GUID.openshiftworkshop.com/auth/admin/userX/console/#/realms/userX>

The screenshot shows the RHSSO interface with the 'openshift' realm selected. The left sidebar has sections for 'Configure' (with 'Realm Settings' selected), 'Manage' (Groups, Users, Sessions, Events, Import, Export), and 'OpenShift'. The main panel shows the 'General' tab of the 'openshift' realm configuration. It includes fields for Name (set to 'openshift'), Display name, HTML Display name, Enabled (set to ON), and Endpoints (set to 'OpenID Endpoint Configuration'). There are 'Save' and 'Cancel' buttons at the bottom.

The realms are like separate instances of the platform, dedicated to separating users and applications. As we can see we can customize several aspects of the realm like the theme of the login page or the tokens' default parameters. **Endpoints -> OpenID Endpoint Configuration**

```

    "issuer": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift",
    "authorization_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/auth",
    "token_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/token",
    "token_introspection_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/token/introspect",
    "userinfo_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/userinfo",
    "end_session_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/logout",
    "jwks_url": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/certs",
    "check_session_iframe": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/login-status-iframe.html",
    "grant_types_supported": [
        "authorization_code",
        "implicit",
        "refresh_token",
        "password",
        "client_credentials"
    ],
    "response_types_supported": [
        "code",
        "none",
        "id_token",
        "token",
        "id_token token",
        "code id_token",
        "code token",
        "code id_token token"
    ],
    "subject_types_supported": [
        "public",
        "pairwise"
    ],
    "id_token_signing_alg_values_supported": [
        "RS256"
    ],
    "userinfo_signing_alg_values_supported": [
        "RS256"
    ],
    "request_object_signing_alg_values_supported": [
        "none",
        "RS256"
    ],
    "response_modes_supported": [
        "query",
        "fragment",
        "form_post"
    ],
    "registration_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/clients-registrations/openid-connect",
    "token_endpoint_auth_methods_supported": [
        "private_key_jwt",
        "client_secret_basic"
    ]
}

```

This is where we can find the public endpoints of the Realm exposed by RH SSO (we are going to be using this later).

Let's now take a look at the Clients section.

Client ID	Enabled	Base URL	Actions		
3scale	True	Not defined	Edit	Export	Delete
account	True	/auth/realms/openshift/account	Edit	Export	Delete
admin-cl	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
launcher-openshift-users	True	Not defined	Edit	Export	Delete
openshift-client	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	/auth/admin/openshift/console/index.html	Edit	Export	Delete

Here we can configure the web or mobile applications that will authenticate using RH SSO as an IDP (corresponding to applications in 3scale). As we can see there are some default clients dedicated to authentication in the integr8ly environment.

Users -> View all users

ID	Username	Email	Last Name	First Name	Actions
c50f0d72-114d-40b6-b318-ad7112e...	user101	user101@openshiftworkshop.com			<a href="#">Edit</a> <a href="#">Impersonate</a> <a href="#">Delete</a>

Here we can see all the end users that are stored inside RH SSO, making it act as an IDM as

well. This is the end user of the applications created in the Clients section and he will be able to authenticate through them. It is also the same user that each one is using to authenticate to the PaaS we are sharing. Let's open the users' details.

The screenshot shows the 'Details' tab of a user profile for 'evals98@example.com'. The user has an ID of 236c0443-50cb-47b4-aed3-987b805e9836 and was created on 1/9/19 12:48:50 PM. The username and email are both evals98@example.com. The first name is listed as 'evals98' and the last name as 'example'. Both 'User Enabled' and 'Email Verified' are set to 'on'. There is a 'Required User Actions' dropdown set to 'Select an action...' and an 'Impersonate' button. At the bottom are 'Save' and 'Cancel' buttons.

We can see here the type of information stored along with basic user details. The user profile can be customized with additional attributes as well.

We will take advantage of one of the features available in OIDC and not in OAUTH which is dynamic client registration.

Normally to make sure an API web application authenticates with RH SSO, we would need to manually create the application on both platforms. With this feature, we let 3scale sync the applications to RH SSO, as well as obviously authenticating our API calls. Let's create a special type of such Client in RH SSO. **Clients -> Create**

The screenshot shows the 'Add Client' page. It includes fields for 'Client ID' (with a required asterisk), 'Client Protocol' (set to 'openid-connect'), 'Client Template' (with a dropdown menu), and 'Root URL'. An 'Import' section with a 'Select file' button is also present. At the bottom are 'Save' and 'Cancel' buttons.

Let's call it sync-app and configure the other details required to let it communicate with 3scale.

We are going to give it only the rights to create applications on behalf of 3scale (*service accounts enabled only*).

### Save -> Service account roles

Add manage-clients to the assigned roles in this window, by picking realm-management in the Client roles menu, this special role allows it to create application on behalf of API management. Then click add selected

And now we are ready to use the client credentials inside 3scale OIDC configuration section. To authenticate as we were an end user, we can just re-use our predefined user (already used to login in the rest of the components)

We have now all the elements to proceed with the corresponding configuration on API management to authenticate calls using our RH SSO.

Let's now switch back to 3scale to configure the API management side of OIDC authentication.

We can see that we have a fully configured API with API key as the Authentication method. We are going to change it to the more secure OpenID Connect, to ensure our financial data are protected from attacks performed when a key is compromised. **Edit integration settings**

**AUTHENTICATION**

**Authentication**  
Authentication is essential to provide Access Control. The chosen authentication mode dictates how your customers will authenticate with your API.

API Key (user\_key)  
The application is identified & authenticated via a single string. ✓

App\_ID and App\_Key Pair  
The application is identified via the App\_ID and authenticated via the App\_Key.

OpenID Connect  
Use OpenID Connect for any OAuth 2.0 flow. ✓

[Update Service](#)

We are going to change it to OpenID Connect. [Update service](#)

**AUTHENTICATION**

**Authentication**  
Authentication is essential to provide Access Control. The chosen authentication mode dictates how your customers will authenticate with your API.

API Key (user\_key)  
The application is identified & authenticated via a single string.

App\_ID and App\_Key Pair  
The application is identified via the App\_ID and authenticated via the App\_Key.

OpenID Connect  
Use OpenID Connect for any OAuth 2.0 flow. ✓

[Update Service](#)

Clearly the platform is warning us that we have customers using this API and it might break their application, changing the authentication method. In a real world case, we would inform the developer in advance by using the messaging and notification functionality available within the platform.

We have now changed the authentication method, we are just left with configuring the correct IdP inside 3scale to make sure it is authenticating the requests with RH SSO. [edit apicast configuration](#)

As we see we have a dedicated field for this purpose now: **OpenID Connect Issuer**

Let's build a url of this format to use it:

`http://client-id:client-secret@<idp-public-endpoint>`  
where client-id: sync-app  
client secret: <defined-during-configuration>  
idp-public-endpoint: sso-http-sso.apps.open-banking-GUID.openshiftworkshop.com/auth/realms/<userX>

Lastly, change the Credentials location to As HTTP Headers

And update the staging environment and promote the configuration to production by clicking the blue button *Promote to production*.

The screenshot shows the 3scale API Management interface. On the left, a sidebar menu includes ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main area has two tabs: 'APIcast Configuration' and 'Environments'.  
**APIcast Configuration Tab:**  
- Private Base URL: https://echo-api.3scale.net:443  
- Mapping rules: / => hits  
- Credential Location: query  
- Secret Token: Shared\_secret\_sent\_from\_proxy\_to\_API\_backend\_1664772f8f98f392  
**Environments Tab:**  
- **Staging Environment:** https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443 v. 5. A blue button labeled 'Promote v. 5 to Production' is visible.  
- **Production Environment:** no configuration has been saved for the production environment yet.  
A 'Configuration history' link is located above the environments section.

Let's now switch user perspective and get in the shoes of the developer and open their Applications section.

The screenshot shows the 'USER101' dashboard with the 'API CREDENTIALS' tab selected. Under 'API CREDENTIALS', the 'Applications' section is active. A new application named 'Test' is being configured. The application details are as follows:

- Name:** Test
- Description:** Test
- Plan:** Basic > Review/Change
- Status:** Live (indicated by a green button)
- Client ID:** 52df2946  
This is the Client ID you should send with each API request.
- Client Secret:** This is the Client Secret used to authenticate requests.  
[Create secret](#)
- Redirect URL:** This is your Redirect URL for OAuth.  
REDIRECT URL:   
[Submit](#)

We can see the secret of their application is absent as is the redirect URL. We are going to generate the first and add as redirect url the following <https://openidconnect.net/callback> (we are going to explain why in a moment).

The screenshot shows the 'Client Secret' field populated with the value '52df2946'. Below the field, there is a 'Regenerate' button and a 'Submit' button.

Let's make sure that the application is now aligned in terms of credentials both in 3scale and RH SSO.

**Overview**

Application 'Test' | Analytics

## Test [Edit](#)

Account	Developer
Description	Test
Service	API
State	Live <a href="#">Suspend</a>

### API Credentials

Client ID	52df2946
Client Secret	ea2494e0efbfddde7b4ae2d77062a60f3 <a href="#">Regenerate</a>
Redirect URL	<a href="https://openidconnect.net/callback">https://openidconnect.net/callback</a> <a href="#">Edit</a>

### Usage in last 30 Days

Application Plan: Basic

[Convert to a Custom Plan](#)

FEATURES

- Unlimited Greetings [Edit](#) (green)
- 24/7 support [Edit](#) (orange)
- Unlimited calls [Edit](#) (red)

### Change Plan

[Change Plan](#)

#### SIGN-ON

##### Clients [?](#)

Client ID	Enabled	Base URL	Actions	
3scale	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
5bc94f6a	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
account	True	/auth/realm/openshift/account	<a href="#">Edit</a>	<a href="#">Export</a>
admin-cli	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
broker	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
launcher-openshift-users	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
openshift-client	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
realm-management	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>
security-admin-console	True	/auth/admin/openshift/console/index.html	<a href="#">Edit</a>	<a href="#">Export</a>
sync-app	True	Not defined	<a href="#">Edit</a>	<a href="#">Export</a>

All looks good! Let's now try to authenticate the end user, using OpenID Connect.

We are going to need a special web client, a little bit more intelligent than just the API tester:

<https://openidconnect.net/>

Let's configure it with the correct parameters from the previous steps. **Configuration**

Let's change the server template to custom and input in the discovery URL the one we opened before in our RH SSO realm

`http://sso-http-sso.apps.open-banking-`

`GUID.openshiftworkshop.com/auth/realm/<userX>/.well-known/openid-configuration`

And click on USE DISCOVERY DOCUMENT

We are going to use the client id and secret as from the application created in the 3scale developer portal / 3scale admin portal or RH SSO since they are all the same.

And lastly as scope we are going to add **openid** and **email**. **SAVE**

### OpenID Connect Configuration

Server Template: Custom

Discovery Document URL: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm...> USE DISCOVERY DOCUMENT  
Use a discovery document to populate your server urls

Authorization Token Endpoint: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm...>

Token Endpoint: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm...>

Token Keys Endpoint: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm...>

Remember to set <https://openidconnect.net/callback> as an allowed callback with your application!

OIDC Client ID: 5bc94f6a

OIDC Client Secret: 2b6db299110348dbae6134e97a8d0359

Scope: openid email

**SAVE**

Hey, just a friendly note: we store stuff like your keys in LocalStorage so that when you redirect to authenticate, you don't lose them. You can clear them by clicking on this button: **CLEAR LOCALSTORAGE**

Start the authentication flow by hitting start. You are going to be redirected to the RH SSO login interface where you can use your user details and password we saw before ([userX](#) / [openshift](#)). Once you login you will receive a temporary code to be exchanged for the final credentials or access token.

## 2 Exchange Code from Token

Your Code is

```
eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAdYXdZhKLt  
QLwD-  
FZL0POaPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiCkjSe8hZWInWah  
Bj4RzQbm53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-  
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jLtf
```

Now, we need to turn that access code into an access token, by having our server make a request to your token endpoint

Request
POST https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/token grant_type=authorization_code &client_id=5bc94f6a &client_secret=2b6db299110348dbae6134e97a8d0359 &redirect_uri=https://openidconnect.net/callback &code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAd QLwD- FZL0POaPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiCkjSe8hZWInWah Bj4RzQbm53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ- 3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jL
<b>EXCHANGE</b>

Hit Exchange

```
Request

POST https://secure-sso-sso.apps.open-
banking.opentry.me/auth/realm/openid-connect/token
grant_type=authorization_code
&client_id=5bc94f6a
&client_secret=2b6db299110348dbae6134e97a8d0359
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAd
QLwD-
FZLQP0aPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiCk5e8hZWInW
Bj4RzQbM53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1zRCY4_my_oyplHCbeJa.jL

HTTP/1.1 200
Content-Type: application/json
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiwi
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiwi
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiwi
  "not-before-policy": 0,
  "session_state": "fa149b8b-d4e9-49be-95b0-9613ff0a55bd",
  "scope": ""
}
```

NEXT

You will receive the “access\_token” which is an expiring credential that we will be using to authenticate with 3scale to get access to the configured API using OpenID Connect. We can see that another important piece of information is shown there regarding when this credential will expire “expires\_in”.

We can hit **NEXT** and id\_token will also be shown, which contains more user related details.

### 3 Verify User Token

Now, we need to verify that the ID Token sent was from the correct place by validating the JWT's signature

Your "id\_token" is

[VIEW ON JWT.IO](#)

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRa1RJX2Vs  
2G0P5v5l7D-  
CDJpLxoCbVr9gNKpBoBb9KFwviyW8I6l6YX6B38rJAJ5WTCXZkyfUaSruIma  
LWNpEPODhiYNg7mmgjxKKYV8bMUW80yB7WSvCGotvF_XBx9K3g
```

This token is cryptographically signed with the RS256 algorithm. We'll use the public key of the OpenID Connect server to validate it. In order to do that, we'll fetch the public key from <https://secure-sso-sso.apps.open-banking.opentrace.me/auth/realm/openid-connect/certs>, which is found in the discovery document or configuration menu options.

[VERIFY](#)

We can decode the information on the website [JWT.io](#) and found our user details once again as passed to the Backend service.

The screenshot shows the JWT.io interface with a decoded JSON Web Token (JWT). The token is a long string of characters. The decoder shows the following fields:

- HEADER: ALGORITHM & TOKEN TYPE**: Contains the following JSON:

```
{"alg": "RS256", "typ": "JWT", "kid": "0kT1_epKb852EJwe7urqPQkhMF-R2xFpMneBvh-U"}
```
- PAYOUT: DATA**: Contains the following JSON:

```
{"jti": "7dcc7531-70ea-41ea-ad31-1502a8aca437", "exp": 1547396457, "nbf": 0, "iat": 1547396157, "iss": "https://secure-sso-sso.apps.open-banking.opentrace.me/auth/realm/openshift", "aud": "5be94f6a", "sub": "236cd4d3-58c6-47b4-ae3d-987bb5e9836", "typ": "ID", "azp": "5be94f6a", "auth_time": 1547395876, "session_state": "fa14988b-d4e9-49be-95b8-9613ff0a55bd", "acr": "0", "preferred_username": "evals98@example.com", "email": "evals98@example.com"}
```
- VERIFY SIGNATURE**: Contains the following code:

```
RSASHA256(  
base64URLEncode(header) + "." +  
base64URLEncode(payload),  
Public Key or Certificate. Enter
```

Let's now go back to <https://openidconnect.net/> website and copy the "access\_token" value in the step 2 (the long string).

Request

```

POST https://secure-sso-sso.apps.open-
banking.opentry.me/auth/realm/openid-connect/token
grant_type=authorization_code
&client_id=5bc94f6a
&client_secret=2b6db299110348dbae6134e97a8d0359
&redirect_url=https://openiddconnect.net/callback
&code=eyJhbGciOiJkaXilCJlbmMioiJBMTI400JDLUhTMju2In0..VeAd
QLwD-
FZLQP0aPD1mGwe70KDJM1_32RqnrcixM00YQePoPt5i6NiCkjSe8hzWinw
Bj4RzobM53WcukRf06m3LtZL0Z0t-X14eJgoGPrSFP37PuFkz-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3v1KCp10F1ZRCY4_my_oyplHCbeJa.jl

```

HTTP/1.1 200

Content-Type: application/json

```
{
  "access_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiIiwi
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiIiwi
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiIiia2lk
  "not-before-policy": 0,
  "session_state": "fa149b8b-d4e9-49be-95b0-9613ff0a55bd",
  "scope": ""
}
```

NEXT

It should look something like this:

```

eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiIiwiA6ICJRa1RXJ2VwS2lwNVpFSkp3ZTdlcnFQUWtjSERnRiSMNhGcE1tZUJ2aC1Vln0.eyJqdGkiOiLyZJmZjQ5ZS01MDY4LTQ0
MjQtYTRINS05MWU3OTk3MTM0YTMIcJleHAIOjE1NDczOTc1NTlsIm5iZl6MCwiaWF0joxtNQ3Mzk2NjUyLCJpc3MiOjodHRwczovL3NIY3VyzS1zc28tc3NvLmFwcHMuB3Bl
biIiYW5raW5nLm9wZW50cnkubWUvYXVoCa9yZWFsbXMb3BlbnNoaWZ0liwYXVkjoiNWJ0tRmNmElCJzdWliOilyMzzjZDRhMy01MGML2LTQ3YjQtYWUzZC05ODdiYjA1ZT
K4MzYiLCJ0eXAiOjCJhenAiOjYm5NGY2YSiSmfIdGhfdGhtZS16MTU0Nz5NTg3Niwc2Vzclvb19zdgF0ZSl6lmZhMTQ5YjhiLWQ0ZTktnDliZS05NWlwLtk2
MTNmZjBhNTViZC1stmfjciil6jAiLCJhbGxvd2VklW9yaWdpbnMi0ltfLCJyZWFsbV9hY2Ni3MiOnsicm9sZXMiOlsidWhx2F1dGhvcml6YXRpb24iXOslnJlc291cmNIX2FjY2Vzcy
l6eyJhY2NvdW50lp7lnJvbGVzljpbilmhbmnZS1hY2NvdW50liwibWFuYWDlWFjY291bnQtbGlua3MiLCJ2aWV3LXByb2ZpbGUiiXX19LCJwcmVmZXJyZWRfdXNlcm5hbWUiOj
IdmFsczk4QGV4YW1wbGUuY29tlwiwzW1haWwiOjJdmFsczk4QGV4YW1wbGUuY29tln0.07y6GDFq5CajAT0DkywEuQqEuD5H7_YMqrVC4AMPthZ-
m_xZ_DAPBEqj3mmzp1o1J0o0_4pMxNgKpyyqCQifY79GRS5lJE6aVrZK53rQkud5diaZAE1-ryiD8Ctp_MrQtsTS7bVKbaFyCXNyFfx3c-
TER8GnGG900IYPxpy5M954slcp4CWxhA7ZwVEuQNRrs5w2G2TCjrFyQjCzslnFwDRtADjbMiY7kq1cwRB5qm9ipdEEigDnH8dietiOZgY24sK10vtowjz_CHuWr5W3474dAZVF
C7utwStl_bNcojigENRcz5cP7fH7Nm8e4itWoSVPRVYcfDHyYb9zixQ

```

We are going to use this as a Header in our call towards the OpenID protected service.

Let's go back to our api tester and add this as an Authorization header. The format is

Authorization      Bearer <access\_token\_value\_here>

Build your test

Click *+* to add or remove steps

Step Name

Request GET https://wt2-evals99-example-com-3scale.apps.open-banking.opentry.me

Headers

+ Add Request Header

Authorization: Bearer eyJhbGciOiJSUzIINi...

Sign In Create Account ⓘ

Collapse / Expand

Let's hit Test

```
Request
Request Headers
GET /open-data/banks HTTP/1.1
Host: wt2-evalis99-example-com-3scale.apps.open-banking.opentrue.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; +http://rigor.com)
Authorization: Bearer eyJhbGciOiJSUzIiNlInRscICgIaISiUiwia2ikIia2iCJRkJX2vS2IwNvpFSkp32Td1cnFQuWtjSERNR115MnhcE1tZUJ2aCIVInO.eYjqdGK

Response
Response Headers
HTTP/1.1 200 OK
Server: openentry/1.13.6.1
Date: Sun, 13 Jan 2019 16:26:57 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5978
Set-Cookie: JSESSIONID=hcqbu9y65vuua9591q06hh3ny; Path=/
Expires: Sun, 13 Jan 2019 16:26:57 GMT
Access-Control-Allow-Origin: *
Cache-Control: no-cache, private, no-store
Correlation-Id: hcqbu9y65vuua9591q06hh3ny
Pragma: no-cache
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src 'self' https://static.openbankproject.com/images/sandboxes/bank-x/logo.png
Referrer-Policy: no-referrer-when-downgrade
Set-Cookie: 4e0f3adfe552726737d1c68037832937e20029bc3faaa2bd456003d95b0e541; path=/; HttpOnly; Secure
Set-Cookie: 0cecd25c053b95cab8739dead952187+fd85f8e621ee8a00fcad673c7c9e50; path=/; HttpOnly; Secure

Response Body


{"banks": [{"id": "psd291-bank-x--uk", "short_name": "Bank X", "full_name": "The Bank of X", "logo": "https://static.openbankproject.com/images/sandboxes/bank-x/logo.png"}]}


Search variables...

```

And success!

The work done by the API management behind the curtain is quite impressive:

- Check for the validity of the access token credentials (not expired, legit and associated to the correct application)
  - Check for rate limits on the application triggering the call
  - Apply monetization rules to the call
  - Apply any additional policy that might modify the call in real time
  - Report the traffic back to the analytics component

## Checkpoint

Improved security to the highest grade possible while using standards.

## OpenShift (optional)

### LAB BEGINS

As user you will login into openshift and it already looks evident that the end user has been profiled as developer on OpenShift as he has access only to Objects and Projects he created.

The screenshot shows the OpenShift Container Platform Service Catalog. At the top, there's a search bar labeled "Search Catalog". Below it, a "My Projects" section shows 2 of 2 projects: "evals98@example.com-walkthrough-projects" (created 3 days ago) and "fuse-dcfbb062-1520-11e9-86c5-0a580a810008" (created 3 days ago). The main area is titled "Browse Catalog" and shows a grid of 139 items under categories like ".NET", "Apache", "Django", "EnMasse", "JBoss", and "PHP". Each item has a small icon and a name like ".NET Core", "amp-apicast-wildcard-router", "che", "fuse", etc.

If we click on the fuse project we will be able to access to the Fuse Online installation dedicated to the user. We would also be able to see any integration project running alongside Fuse installation.

If we switch to the **Cluster console**, this will give us some Operations details on the project created or assigned to our user.

The screenshot shows the OpenShift Container Platform Cluster Console. On the left, a sidebar menu includes "Home", "Workloads", "Networking", "Storage", "Builds", "Administration" (which is expanded to show "Projects", "Service Accounts", "Roles", "Role Bindings", and "Resource Quotas"). The main area is titled "Projects" and shows a table with two rows:

NAME	STATUS	REQUESTER	LABELS
evals98@example.com-walkthrough-projects	Active	evals98@example.com	No labels
fuse-dcfbb062-1520-11e9-86c5-0a580a810008	Active	No requester	No labels

This type of console is also used by Operations administrators to check the health of OpenShift. We can see the RBAC in action if we click on **Home -> Status**

The **Project default** is excluded from the scope of any evals users, since it can contain system components and privileged objects.

We can just switch to the Fuse project to see if there anything wrong with it in the cluster.

We will now try as bad intentioned user to change some parameters around the installed products.

**OPENSHIFT CONTAINER PLATFORM Cluster Console**

The screenshot shows the OpenShift Container Platform Cluster Console interface. On the left, there's a navigation sidebar with sections like Home, Workloads (Pods, Deployments, Deployment Configs, Stateful Sets, Secrets, Config Maps), Cron Jobs, Jobs, Daemon Sets, Replica Sets, Replication Controllers, HPAs, Networking, Storage, Builds, and Administration (Projects, Service Accounts, Roles). The main area displays a list of pods:

- syndesis-operator-1-b5tlr**: IP 172-31-4-79.ec2.internal, Running, Ready. Labels: syndesis.io/com...=syndesis, syndesis.io/type=operator.
- syndesis-prometheus-1-6dczf**: IP 172-31-0-6.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-prometheus-1, deploymentconfig=syndesis-prometheus-1, syndesis.io/app=syndesis, syndesis.io/com...=syndesis-prometheus-1, syndesis.io/type=infrastructure.
- syndesis-server-1-jhmn**: IP 172-31-1-208.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-server-1, deploymentconfig=syndesis-server-1, syndesis.io/app=syndesis, syndesis.io/com...=syndesis-server-1, syndesis.io/type=infrastructure.
- syndesis-ui-1-c6gc9**: This pod has a context menu open with options: View Environment, Edit Labels, Edit Annotations, Edit Pod, and Delete Pod. IP 172-31-3-6.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-ui-1, deploymentconfig=syndesis-ui-1, syndesis.io/app=syndesis, syndesis.io/component=syndesis-ui-1, syndesis.io/type=infrastructure.
- todo-1-8bd4k**: IP 172-31-3-6.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=todo-1.

**CONTAINER PLATFORM Cluster Console**

The screenshot shows the Container Platform Cluster Console interface. The left sidebar is identical to the OpenShift one. The main area shows a list of pods, with the 'Delete Pod' dialog box overlaid:

- syndesis-operator-1-b5tlr**: IP 172-31-4-79.ec2.internal, Running, Ready. Labels: syndesis.io/type=operator.
- syndesis-prometheus-1-6dczf**: IP 172-31-0-6.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-prometheus-1, deploymentconfig=syndesis-prometheus-1, syndesis.io/app=syndesis, syndesis.io/com...=syndesis-prometheus-1, syndesis.io/type=infrastructure.
- syndesis-server-1-jhmn**: IP 172-31-1-208.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-server-1, deploymentconfig=syndesis-server-1, syndesis.io/app=syndesis, syndesis.io/com...=syndesis-server-1, syndesis.io/type=infrastructure.
- syndesis-ui-1-c6gc9**: IP 172-31-3-6.ec2.internal, Running, Ready. Labels: app=syndesis, deployment=syndesis-ui-1, deploymentconfig=syndesis-ui-1, syndesis.io/app=syndesis, syndesis.io/component=syndesis-ui-1, syndesis.io/type=infrastructure.

The 'Delete Pod' dialog box contains the message: "Are you sure you want to delete syndesis-ui-1-c6gc9 in namespace fuse-dcfbb062-1520-11e9-86c5-0a580a810008?" with "Cancel" and "Confirm" buttons.

As we can see we tried to kill one of the running components of our integration platform with no success, because of the roles assigned to my user.

## DEMO ONLY

Let's see the magic introduced by OpenShift and login as administrator of the platform once again.

We now have full access to all the platforms from all users. We will open as admin one of the Fuse projects and open one of the components of Fuse Online.

Deployment Config	Pod Count	Memory	Cores CPU	Network
DEPLOYMENT CONFIG syndesis-db, #1	1	100 Mb Memory	< 0.01 Cores CPU	52 Kib/s Network
DEPLOYMENT CONFIG syndesis-meta, #1	1	340 Mb Memory	< 0.01 Cores CPU	0.1 Kib/s Network
DEPLOYMENT CONFIG syndesis-oauthproxy, #1	1	11 Mb Memory	< 0.01 Cores CPU	1.5 Kib/s Network
DEPLOYMENT CONFIG syndesis-operator, #1	1	16 Mb Memory	< 0.01 Cores CPU	1.8 Kib/s Network
DEPLOYMENT CONFIG syndesis-prometheus, #1	1	50 Mb Memory	< 0.01 Cores CPU	1.9 Kib/s Network
DEPLOYMENT CONFIG syndesis-server, #1	1	570 Mb Memory	0.01 Cores CPU	41 Kib/s Network
DEPLOYMENT CONFIG syndesis-ui, #1	1	580 Mb Memory	0.02 Cores CPU	53 Kib/s Network

We are going to test the auto healing capabilities of the platform by killing one of its running components, in particular the one providing the UI service.

Container	Average Usage (Last 15 Minutes)
syndesis-ui	5.7 Mb Memory, 0 Cores CPU, 0.3 Kib/s Network

**CONTAINERS**

- syndesis-ui
  - Image: fuse7/fuse-ignite-ui 0c85f43 100.0 MB
  - Ports: 8080/TCP

**NETWORKING**

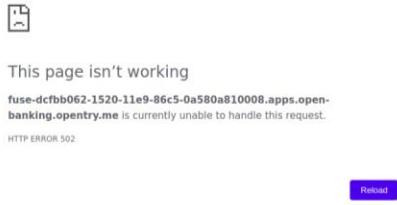
Service - Internal Traffic  
syndesis-ui  
80/TCP → 8080

Routes - External Traffic  
[Create Route](#)

The image consists of three vertically stacked screenshots from the OpenShift Container Platform Application Console. Each screenshot shows a different stage of the pod deletion process:

- Screenshot 1:** Shows the pod details page for 'syndesis-ui-1-c6gc9'. The status is 'Running'. In the top right corner, there is a 'Actions' dropdown menu with options: 'Add Storage', 'Edit YAML', and 'Delete'. The 'Delete' option is highlighted.
- Screenshot 2:** A 'Confirm Delete' dialog box is displayed. It asks, 'Are you sure you want to delete the pod 'syndesis-ui-1-c6gc9'? It cannot be undone. Make sure this is something you really want to do!'. There is a checkbox labeled 'Delete pod immediately without waiting for the processes to terminate gracefully'. Below the dialog are 'Cancel' and 'Delete' buttons.
- Screenshot 3:** The pod list page after the deletion. The pod 'syndesis-ui-1-c6gc9' is listed with a status of 'Deleted'. A message at the top right says 'Pod 'syndesis-ui-1-c6gc9' was marked for deletion.' The pod details page is shown again, but the pod has been removed from the list.

As you can see we just deleted a Pod and we will verify that UI is broken by accessing the interface of Fuse Online



**OPENSHIFT CONTAINER PLATFORM Application Console**

**fuse-dcfbb062-1520-11e9-86c5-0a580a810008**

- Overview
- Applications
- Builds
- Resources
- Storage
- Monitoring
- Catalog

**DEPLOYMENT CONFIG**  
syndesis-meta, #1

340 Mb Memory < 0.01 Cores CPU 0.1 Kib/s Network 1 pod

**DEPLOYMENT CONFIG**  
syndesis-oauthproxy, #1

12 Mb Memory < 0.01 Cores CPU 1.5 Kib/s Network 1 pod

**DEPLOYMENT CONFIG**  
syndesis-operator, #1

16 Mb Memory < 0.01 Cores CPU 2.8 Kib/s Network 1 pod

**DEPLOYMENT CONFIG**  
syndesis-prometheus, #1

50 Mb Memory < 0.01 Cores CPU 1.9 Kib/s Network 1 pod

**DEPLOYMENT CONFIG**  
syndesis-server, #1

570 Mb Memory < 0.01 Cores CPU 89 Kib/s Network 1 pod

**DEPLOYMENT CONFIG**  
syndesis-ui, #1

Average Usage Last 15 Minutes

**CONTAINERS**  
syndesis-ui  
Image: fuse7/fuse-ignite-ui 0c85f43 100.0 MB  
Ports: 8080/TCP

**NETWORKING**  
Service - Internal Traffic  
syndesis-ui

Routes - External Traffic  
Create Route

**RED HAT FUSE ONLINE**

- Home
- Integrations
- Connections
- Customizations
- Settings

**System Metrics**

0 Integrations	5 Connections	11 Total Messages	Uptime Since Jan 13th 18:23 PM 2 days 19 hours 38 minutes
0 0	0 0	11 0	

**Create an Integration**  
There are currently no integrations. Click the button below to create one.  
[Create Integration](#)

**Connections**

PostgresDB	open-financial-service	Log	Timer
------------	------------------------	-----	-------

[View All Connections](#) [Create Connection](#)

As we can see the component auto-healed thanks to OpenShift features and in a few seconds we have a GUI running once again for the integration platform.

## Q&A

### Common issues

- openidconnect.net client might have an additional space in the redirect\_uri field. That's a client bug, you can fix it by adding an additional redirect URIs in RH SSO with a space preceding the URL: “<https://openidconnect.net/callback>”<sup>[17]</sup>
- The installation of RH SSO might have some certificate issues, so might need to use instead a RH SSO deployed somewhere else or using the HTTP only route as suggested in the tutorial