

# 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

Environment:

Environment reachable here:

<https://tutorial-web-app-webapp.apps.openbanking-4f6a.openshiftworkshop.com/>

End Users credentials:

user[1..100] / openshift

3scale dedicated instances here:

<https://userX-admin.apps.openbanking-4f6a.openshiftworkshop.com>

access using username and password

dedicated gateway

<https://userX.amp.apps.openbanking-4f6a.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 schedule) 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.openbanking-4f6a.openshiftworkshop.com/>

The screenshot shows the Red Hat Solution Explorer interface. On the left, there's a section titled "Start with a walkthrough" featuring three cards: "Integrating event-driven and API-driven applications (AMQ)" (preview), "Integrating event-driven and API-driven applications (EnMasse)" (Get Started, 15 min), and "Integrating API-driven applications" (Get Started, 21 min). To the right, a sidebar lists "Applications" with 7 applications: Red Hat OpenShift (Ready for use), Red Hat 3scale API Management Platform (Ready for use), Red Hat AMQ (Ready for use), Eclipse Che (Ready for use), EnMasse (Ready for use), Red Hat Fuse (Ready for use), and Red Hat Developer Launcher (Ready for use). A "community" badge is next to Eclipse Che and Red Hat Fuse. At the bottom, a navigation bar includes links like "Combining Federation V2 and Isti...", "The Importance of REST APIs for 5...", "OpenShift Web Console - Google...", "Red Hat Solution Explorer - Google...", "Franz", and "FSI Roadshow lab notes - Google...".

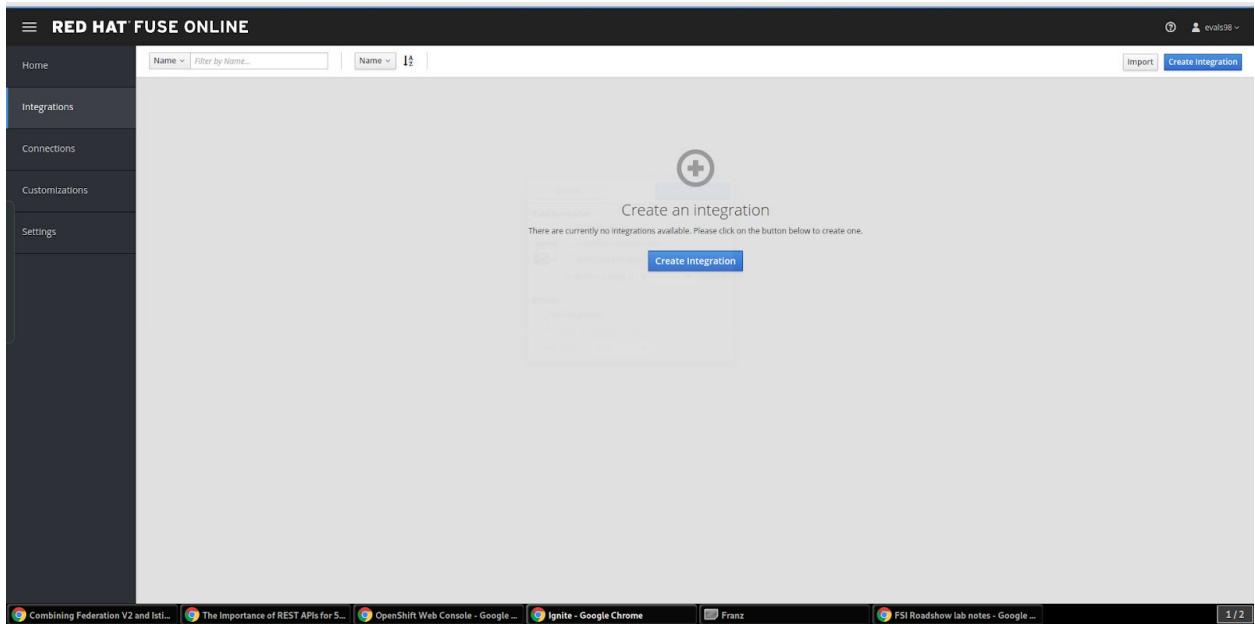
We will see how integr8ly environment works and what you get. It provides out of the box integration between products but it doesn't change the base product. You can still install this type of environment starting from a clean or empty OpenShift installation.

## Fuse Online

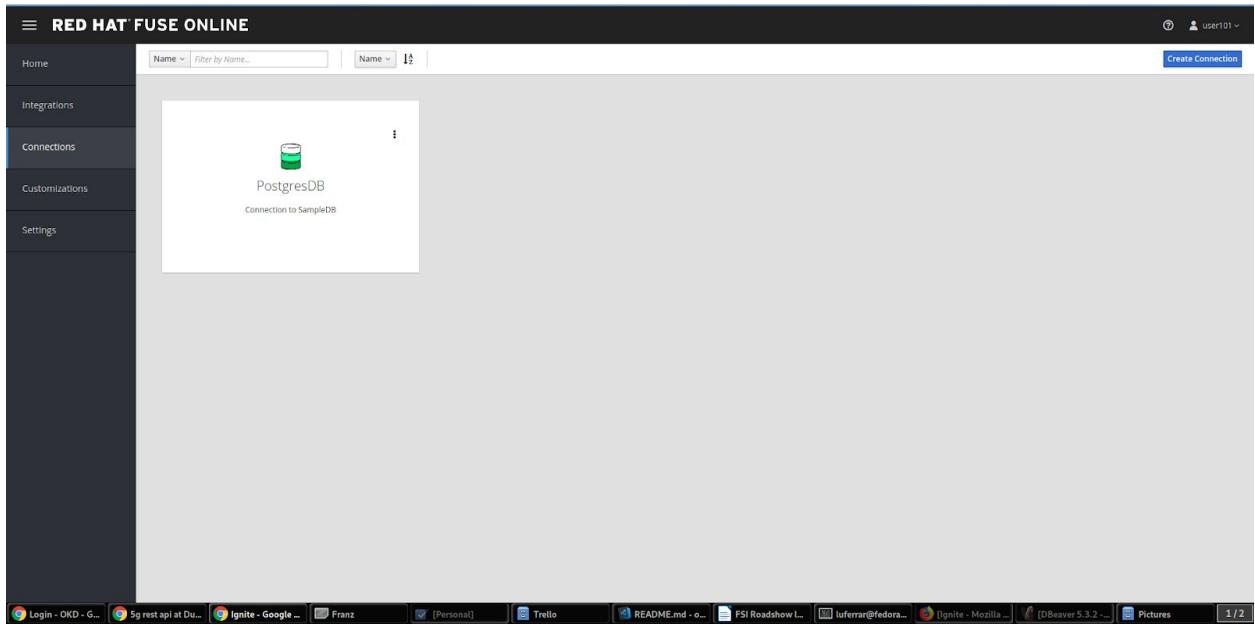
Let's start our first lab session, by opening Red Hat Fuse Online. We will see the main functionalities of it and then use it.

The screenshot shows the Red Hat Fuse Online dashboard. On the left, a sidebar menu includes Home, Integrations, Connections, Customizations, and Settings. The main area starts with "System Metrics" showing 0 Integrations (0 green, 0 red), 4 Connections, 0 Total Messages (0 green, 0 red), and Uptime (Since Jan 10th 22:44 PM, 2 minutes). Below this is a "Create an Integration" section with a large plus icon, a message stating "There are currently no Integrations. Click the button below to create one.", and a "Create Integration" button. Further down is a "Connections" section with four cards: PostgresDB (database icon), Log (document icon), Timer (clock icon), and Webhook (circular arrow icon). At the bottom, a navigation bar includes links like "Combining Federation V2 and Isti...", "The Importance of REST APIs for 5...", "OpenShift Web Console - Google...", "Ignite - Google Chrome", "Franz", and "FSI Roadshow lab notes - Google...".

Explain the main dashboard, especially important when you then show messages coming from the request being sent.



Explain the role of the integration window (seeing graphically the mediation/transformation) and the relevance for citizen developers. Little detour on what's behind the curtains (Camel) to reassure people that's not new technology but a well tested one and with 100s of possible connectors.



Connections section. Explain that you can start or end (or sometimes interpose) connections which are completely 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. Make a couple of examples about possible integrations or mashups.

The screenshot shows the 'API Client Connectors' section of the Red Hat Fuse Online interface. On the left, a sidebar menu includes 'Home', 'Integrations', 'Connections', 'Customizations' (which is currently selected), and 'Settings'. The main content area has tabs for 'API Client Connectors' (selected) and 'Extensions'. Below the tabs, there's a brief description of what Ignite does: 'ignite creates an API client connector when you upload a valid OpenAPI 2.0 specification that describes the API you want to connect to.' There are two search/filter boxes: one for 'Name' with a dropdown 'Filter by Name...' and another for 'Name' with a dropdown '12'. In the center, there's a large button with a plus sign inside a circle labeled 'Create API Connector'. Below it, a message says 'There are currently no API connectors available. Please click on the button below to create one.' At the bottom right of the content area is a blue 'Create API Connector' button. The browser's address bar at the bottom shows several tabs, including 'Combining Federation V2 and Isti...', 'The Importance of REST APIs for S...', 'OpenShift Web Console - Google...', 'Ignite - Google Chrome', 'Franz', and 'FSI Roadshow lab notes - Google...'. The page is identified as '1 / 2'.

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

The screenshot shows the 'Extensions' section of the Red Hat Fuse Online interface. The sidebar menu is identical to the previous screenshot. The main content area has tabs for 'API Client Connectors' (disabled) and 'Extensions' (selected). A message above the tabs reads: 'Extensions provide custom features for use in Integrations. Find out more at Ignite Help'. Below the tabs are two search/filter boxes: 'Name' with 'Filter by Name...' and 'Name' with '12'. In the center, there's a large button with a plus sign inside a circle labeled 'Import Extension'. Below it, a message says 'There are no extensions available.' At the bottom right of the content area is a blue 'Import Extension' button. The browser's address bar at the bottom shows several tabs, including 'Combining Federation V2 and Isti...', 'The Importance of REST APIs for S...', 'OpenShift Web Console - Google...', 'Ignite - Google Chrome', 'Franz', and 'FSI Roadshow lab notes - Google...'. The page is identified as '1 / 2'.

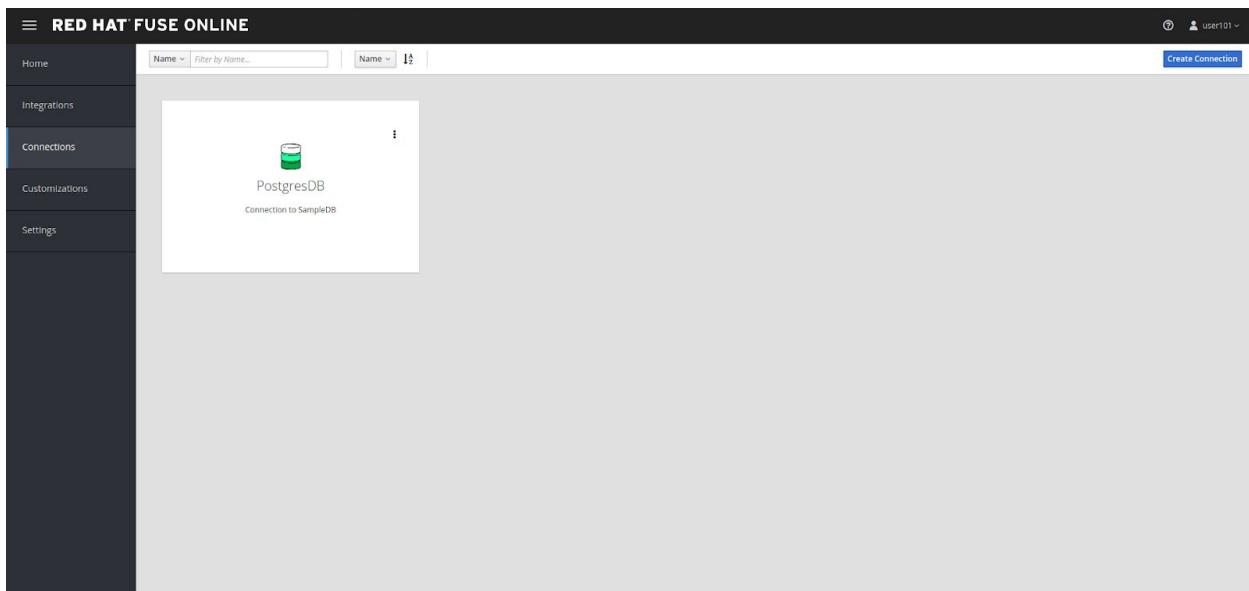
Extensions pane: this is the part where you can extend the functionalities of the platform. Ideally you would have prepared one compiled extension and you can import it to see how easy it is to add transformation functionalities to the product.

**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 ElephantSQL. You could be running this anywhere else, as

long as you have a public direct connection to the DB to use.

## Create connection



A screenshot of the "Create Connection" wizard, Step 1: Select Connector. The title bar says "RED HAT FUSE ONLINE" and "Create Connection". The sub-navigation shows "Home &gt; Connections &gt; Create Connection". The main content area is titled "Select Connector" and shows a grid of connector icons and names. The "Database" connector is highlighted. Other connectors shown include Amazon S3, AMQ Message Broker, AMQP Message Broker, Dropbox, FHIR, and Gmail. A progress bar at the top indicates steps 1, 2, and 3.

Select a Database connector

The screenshot shows the 'Configure Connection' step of the 'Create Connection' wizard. The 'Database Configuration' form displays a success message: 'Database has been successfully validated.' Below this message, there is a note: 'The fields marked with \* are required.' followed by four input fields: 'Connection URL' (value: jdbc:postgresql://manny.db.elephantsql.com:5432/hxuazym), 'Username' (value: hxuazym), 'Password' (value: eiHvH6xbZ2\_DRdeoAu\_tLbNe3des\_NUf), and 'Schema' (value: hxuazym). A 'Validate' button is visible at the top right of the form.

Use the following connection details and validate them:

Connection details: `jdbc:postgresql://manny.db.elephantsql.com:5432/hxuazym`

username: `hxuazym`

password: `eiHvH6xbZ2_DRdeoAu_tLbNe3des_NUf`

schema: `hxuazym`

Let's name the connection and finalize it.

The screenshot shows the 'Name Connection' step of the 'Create Connection' wizard. The 'Database Configuration' form displays a success message: 'Database has been successfully validated.' Below this message, there is a note: 'The fields marked with \* are required.' followed by four input fields: 'Connection URL' (value: jdbc:postgresql://manny.db.elephantsql.com:5432/hxuazym), 'Username' (value: hxuazym), 'Password' (value: eiHvH6xbZ2\_DRdeoAu\_tLbNe3des\_NUf), and 'Schema' (value: hxuazym). A 'Validate' button is visible at the top right of the form.

The screenshot shows the 'Create Connection' process in the Red Hat Fuse Online interface. The steps are:

- Select Connector**: Shows a list of connectors, with 'open-data-banks' selected.
- Configure Connection**: Shows configuration options for the selected connector.
- Name Connection**: Shows the connection name 'open-data-banks' entered.

The 'Add Connection Details' form includes fields for 'Connection Name' (set to 'open-data-banks') and 'Description'. The browser's address bar shows the URL as `https://fuse-b5ddd8b-1b5e-11e9-9d4e-0a580a010007.apps.openshiftworkshop.com/connections`.

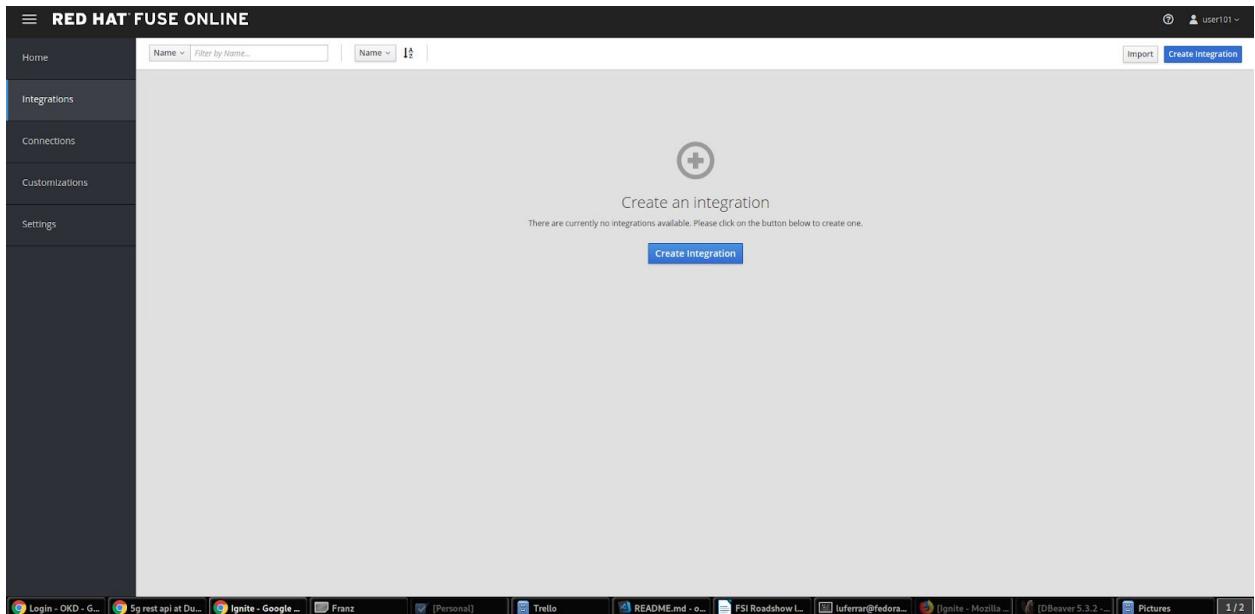
**Connections Overview:**

- open-data-banks**: Connection to SampleDB
- PostgresDB**: Connection to SampleDB

The browser's status bar indicates the date as Mon Jan 21 09:04.

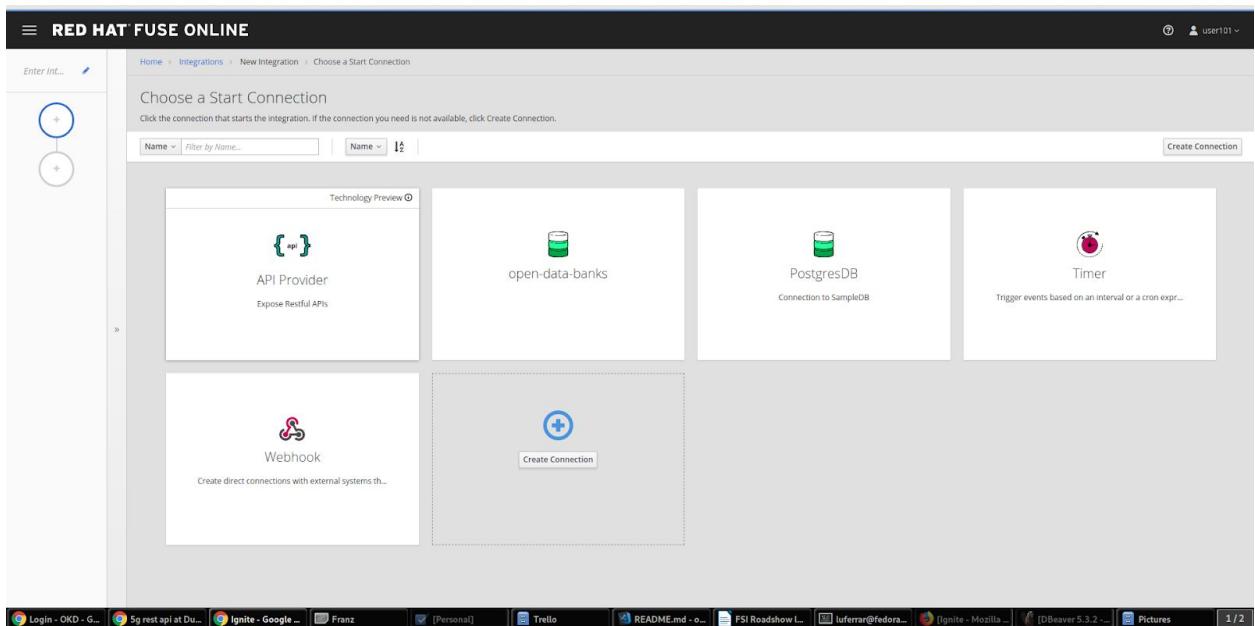
We can now use this connection as an integration starting, middle or finishing point.

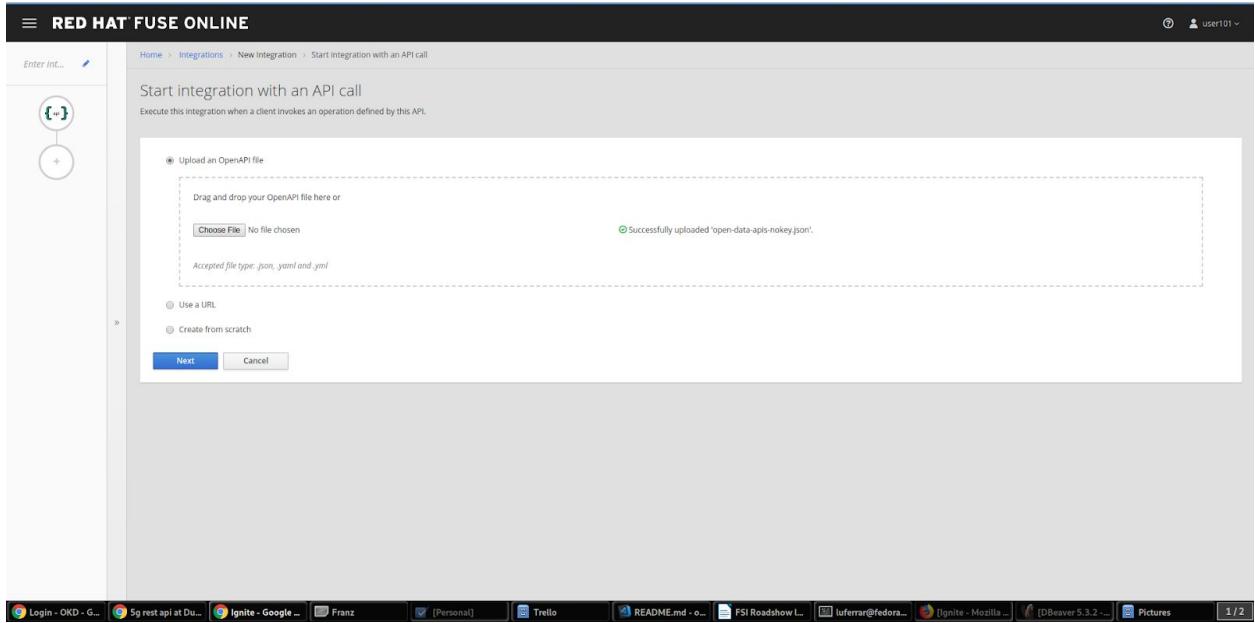
Now that we have configured the end of the integration transformation and seen where to find the start we will put the two pieces together. **Integrations -> create**



We will start by exposing a REST endpoint that will then get linked 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>





The definition is the same one used on the Open Banking solution portal for Open Data APIs as seen before.

There is a validation happening on the API definition, but no error was identified so we can proceed with the configuration of the connector.

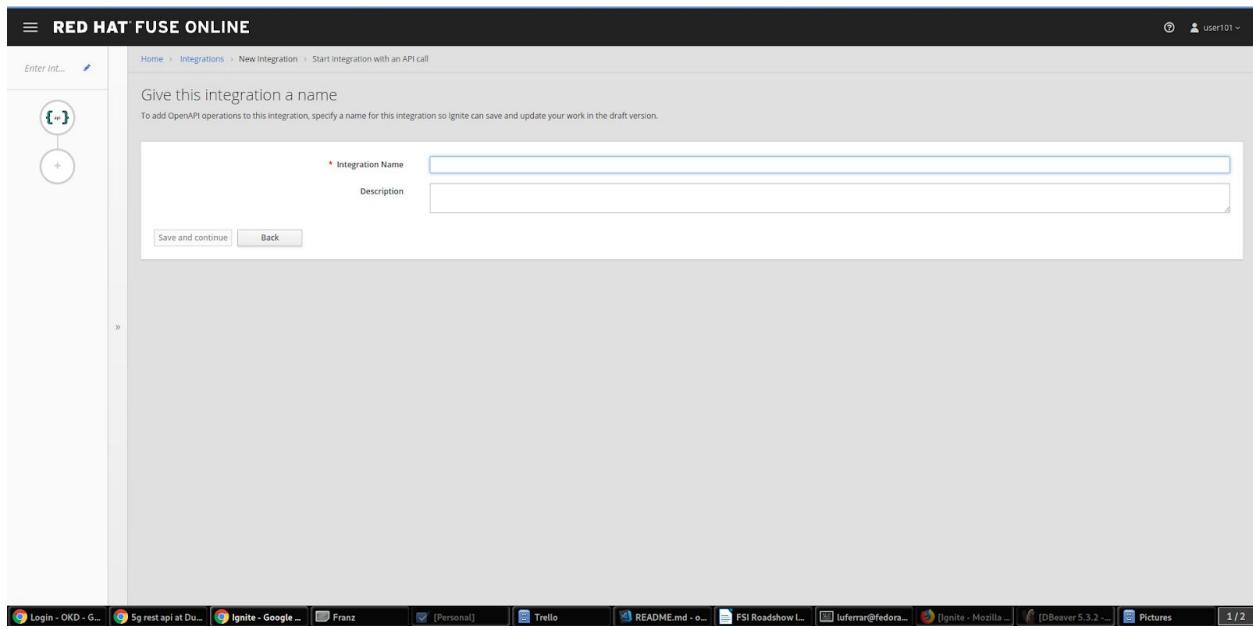
To show how easy it is to correct definitions, -> review/edit

Permissions	Conditions	Limitations
<ul style="list-style-type: none"> <li>Commercial Use</li> <li>Distribution</li> <li>Modification</li> <li>Patent Use</li> <li>Private Use</li> </ul>	<ul style="list-style-type: none"> <li>Disclose source</li> <li>License and copyright notice</li> <li>Same License</li> <li>State changes</li> </ul>	<ul style="list-style-type: none"> <li>Lubility</li> <li>Warranty</li> </ul>

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.

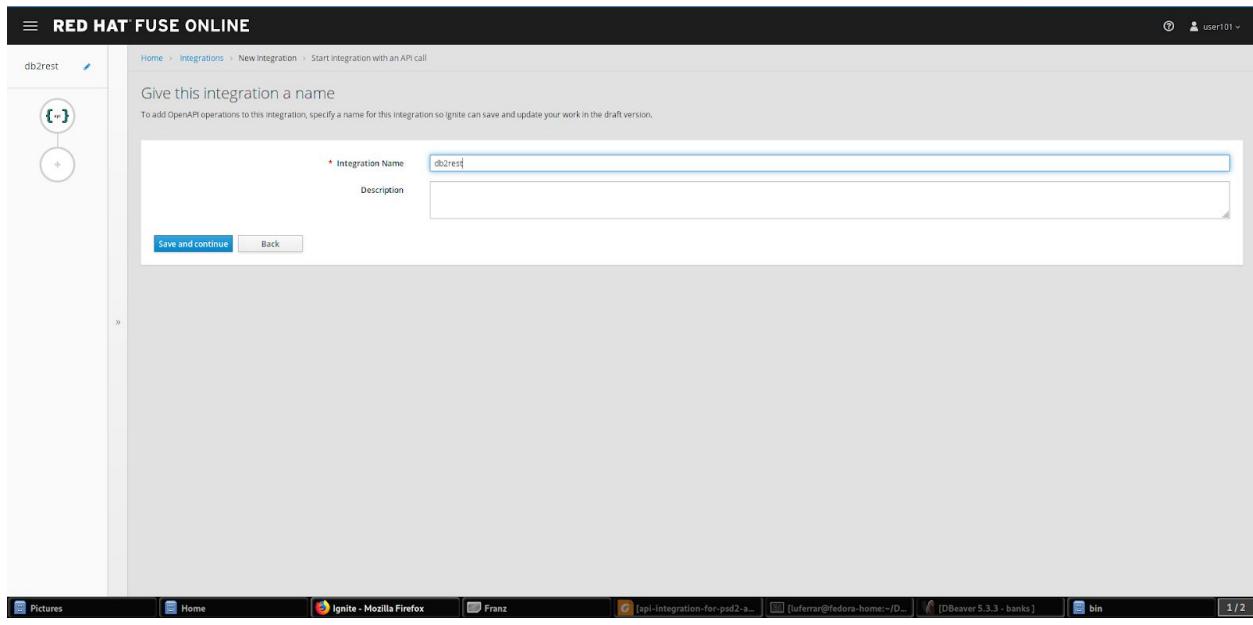
The service is exposed without any protection (that's one of the reasons we are going to be using 3scale later on) since the API definition is not adding any authentication on top of it.

Name the integration.



The screenshot shows the Red Hat Fuse Online web interface. The title bar says "RED HAT FUSE ONLINE". The URL in the address bar is "Home > Integrations > New Integration > Start integration with an API call". On the left, there's a sidebar with a "Enter Int..." button and a diagram editor showing a single node. The main content area has a heading "Give this integration a name" and a note: "To add OpenAPI operations to this integration, specify a name for this integration so Ignite can save and update your work in the draft version." Below this are two input fields: "Integration Name" (containing "db2rest") and "Description". At the bottom are "Save and continue" and "Back" buttons. The status bar at the bottom shows various browser tabs and icons.

And save and continue



This screenshot is identical to the previous one, showing the "New Integration" step. The only difference is that the "Integration Name" field now contains "db2rest" instead of the placeholder text. The "Save and continue" button is highlighted with a blue border, indicating it is the next action to be taken.

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

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with 'db2rest' and 'API Provider'. The main area is titled 'Choose operation' with the sub-instruction 'Click an operation to integrate an integration flow.' Below this is a search bar with 'Name' and 'Filter by Name...' dropdowns, and a 'Name' dropdown with a magnifying glass icon. A table lists several operations:

Operation	Endpoint	Status
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

At the bottom of the browser window, there's a toolbar with various icons and a status bar showing '1 / 2'.

Click on *get list of banks*

The screenshot shows the 'Save or Add Step' screen for the 'get list of banks' operation. The top navigation bar includes 'Home', 'Integrations', 'db2rest', 'get list of banks', 'Save or Add Step', 'Cancel', 'Go to Operation List', 'Save as Draft', and 'Publish'. The main area features a large plus sign icon and the heading 'Add to Integration'. Below it, a message says 'Now you can add additional connections as well as steps to your integration. You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom are 'Add a Step' and 'Add a Connection' buttons. The browser toolbar and status bar are visible at the bottom.

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.

Let's connect this front end to the database we previously configured as a terminating connection.

**Add a connection**

**RED HAT FUSE ONLINE**

db2rest 

Home > Integrations > db2rest > Choose a Connection

Choose a Connection

Click the connection that completes the integration. If the connection you need is not available, click Create Connection.

Name  Filter by Name... Name  

 Log  
Log the exchange with different options

 open-data-banks

 PostgresDB  
Connection to SampleDB





get list of ba... 

Click on the previously configured data source.

☰ RED HAT FUSE ONLINE

db2rest

get list of ba... ⓘ

Home > Integrations > Choose Action

open-data-banks - Choose an Action

Choose an action for the selected connection.

Name Filter by Name... Name ↕

Invoke SQL

Invoke stored procedure

Invoke SQL

Periodically invoke a stored procedure.

Cancel

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

The screenshot shows the Red Hat Fuse Online interface. The left sidebar has a tree structure with 'db2rest' expanded, showing 'get list of ba...' and other options. The main panel title is 'db2rest' and the sub-section is 'Configure Invoke SQL'. A sub-sub-section titled 'open-data-banks' is selected. Below it, there's a section for 'Invoke SQL' with a note: 'Enter a SQL statement that starts with INSERT, SELECT, UPDATE or DELETE.' A text input field contains 'select \* from banks'. At the bottom of this section are 'Cancel', 'Choose Action', and 'Done' buttons.

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.

The screenshot shows the Red Hat Fuse Online interface. The left sidebar has a tree structure with 'db2rest' expanded, showing 'get list of ba...' and other options. The main panel title is 'db2rest' and the sub-section is 'Save or Add Step'. A central area is titled 'Add to Integration' with the sub-instruction: 'Now you can add additional connections as well as steps to your integration. You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom of this area are 'Add a Step' and 'Add a Connection' buttons. The top right of the screen has buttons for 'Cancel', 'Go to Operation List', 'Save as Draft', and 'Publish'.

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

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar for the integration named "db2rest". The main area displays a flow diagram with three nodes: a "HTTP" node at the top, a "DB" node in the middle, and a "Log" node at the bottom. Below the flow diagram are two buttons: "Add a step" and "Add a connection". To the right of the flow diagram, there's a large central panel titled "Add to Integration". It contains a prominent plus sign icon and the text "Now you can add additional connections as well as steps to your integration. You can interact with the left hand panel to continue adding steps and connections to your integration as well." At the bottom of this panel are two buttons: "Add a Step" and "Add a Connection". At the very bottom of the page, there's a standard browser toolbar with various icons.

Select the log step.

The screenshot shows the "Choose a Step" screen within the Red Hat Fuse Online interface. The left sidebar still shows the "db2rest" integration. The main area is titled "Choose a Step" and contains a sub-header: "A step specifies an operation on the data in the integration. Click one to add it." Below this, there are four filter options: "Advanced Filter", "Basic Filter", "Log", and "Template". The "Log" option is selected, indicated by a blue border around its "Log" button. The "Log" section describes the step as "Sends a message to the integration's log". The "Template" section is described as "Upload or create a Mustache template to define consistent output data". At the bottom of the page, there's a standard browser toolbar.

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

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a tree view showing the integration flow: 'db2rest' -> 'get list of banks'. The main area is titled 'Configure Log' under the 'db2rest' step. A modal window is open, prompting the user to 'Fill out the fields associated with the selected step.' It contains two checkboxes: 'Message Context' (unchecked) and 'Message Body' (checked). Below these is a 'Custom Text' input field, which is currently empty. At the bottom of the modal are 'Cancel' and 'Done' buttons.

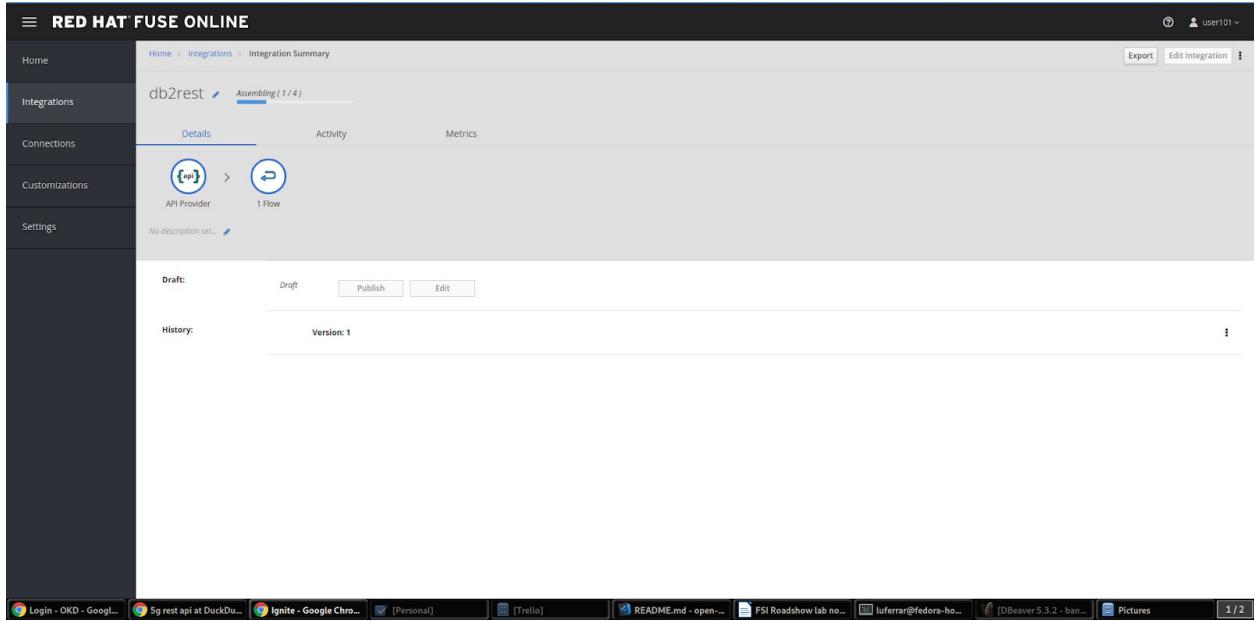
>Login - OKD - Google... | Sg rest api at DuckDu... | Ignite - Google Chro... | [Personal] | [Trello] | README.md - open... | FSI Roadshow lab no... | lufer...@fedor...-ho... | DBeaver 5.3.2 - ban... | Pictures | 1 / 2

We are going to log just the message body.

The screenshot shows the Red Hat Fuse Online interface. The left sidebar shows the integration flow: 'db2rest' -> 'get list of banks'. The main area is titled 'Save or Add Step' under the 'get list of banks' step. A modal window is open, titled 'Add to Integration'. It contains the text: 'Now you can add additional connections as well as steps to your integration. You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom of the modal are 'Add a Step' and 'Add a Connection' buttons.

>Login - OKD - Google... | Sg rest api at DuckDu... | Ignite - Google Chro... | [Personal] | [Trello] | README.md - open... | FSI Roadshow lab no... | lufer...@fedor...-ho... | DBeaver 5.3.2 - ban... | Pictures | 1 / 2

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 PROTECT IT ANYWAY.

Now that you have shown the attendees how to build the full integration block you can test the integration just built, using this online tool <https://apitester.com/>. For the attendees to also test you can share the following URL for them to use  
<http://i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com>

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

The screenshot shows the API Tester BETA interface. At the top, there's a toolbar with various icons and a search bar. Below the toolbar, the title "API TESTER BETA" is displayed. On the right side of the header, there are "Sign In" and "Create Account" buttons. The main area is titled "Build your test" and has a sub-section "View example". A "Request" step is defined, showing a GET method and the URL "https://open-data.b9sd.pro-us-east-1.openshiftapps.com/open-data/banks". There's also a "Headers" section with "+ Add Request Header". Below the request step, there's a button "+ Add Step" and three buttons at the bottom: "Test" (highlighted in blue), "Save to Account", and "Share Test Config". A callout box titled "Saving tests to your account allows you to:" lists four options: "Rerun this test", "Share test results with others", "View the run history", and "Create and re-run multiple tests".

© 2019 RIGOR. ALL RIGHTS RESERVED

Powered by RIGOR

Populate the fields with the following URL  
<http://i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com/open-data/banks>  
and hit Test

The screenshot shows the API Tester results page. At the top, it says "PASS" with a green checkmark icon. Below that, it says "Results 6.29 s" and "N. Virginia" with "Seconds elapsed: 11". There are tabs for "Request Step" (highlighted in green) and "Response Step". The "Request Step" tab shows a message "[Step 1] GET http://i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com/open-data/banks passed". Below that, there are sections for "Connection Information", "Request Headers", and "Response Headers". The "Request Headers" section shows the following 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)
    
```

The "Response Headers" section shows the following headers:

```

HTTP/1.1 200 OK
Transfer-Encoding: chunked
X-Application-Context: application
Date: Mon, 21 Jan 2019 10:20:33 GMT
Set-Cookie: 12b28c7c44e5989665ef8abec492fd0@08271c1b765ada49ab2c2c05b8b72917; path=/; HttpOnly
Cache-control: private
    
```

The "Response Body" section contains the JSON response: {"short\_name": "Bank X", "id": 1, "address": "psd201-bank-x--uk", "long\_name": "The Bank of X"}. Below the response body, there's a "Variables" section with a search bar "Search variables...".

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



Show the structure of the JSON response, with basic information given around banks (dummy data).

### 3scale

To reach 3scale as a user you can just go back to the main integr8ly dashboard and click on 3scale

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

Explain that you are now logged in 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.

**Signed in successfully**

**AUDIENCE**

**1 Signups**  
last 30 days

**Potential Upgrades**  
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.

**APIS**

**API**

**0 Hits**  
last 30 days

**Top Applications**  
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.

**NEW API**

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**. 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.

**New API**

Define manually  
 Import from OpenShift

**SERVICE**

Namespace

Name

**Create Service**

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

3scale 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.

The screenshot shows the 'New API' creation interface. At the top, there are two radio button options: 'Define manually' (unchecked) and 'Import from OpenShift' (checked). Below these, there are two input fields: 'Namespace' containing 'fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007' and 'Name' containing 'i-mydata'. At the bottom right of the form, there is a blue 'Create Service' button.

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

The dashboard summary includes:

- AUDIENCE:** 1 Signups (last 30 days)
- POTENTIAL UPDATES:** Accounts that hit their Usage Limits in the last 30 days. Note: 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.
- DRAFTS:** TODAY: Test created on API, Application Test has been deleted. BEFORE TODAY: Test created on API, Application Test has been deleted, Test created on API, Application Developer's App has been deleted.
- MESSAGES:** 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

The screenshot shows the Red Hat 3Scale API Management interface. The left sidebar is dark and contains navigation items: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area has a light background. At the top, it says "Configuration". Below that is a section titled "Integration settings" with two rows: "Deployment Option" set to "APICAST" and "Authentication" set to "API Key (user\_key)". A blue button at the bottom left of this section says "add the base URL of your API and save the configuration.". At the very bottom of the page, there's a "Support" link and a "Version 2.4.0 - Powered by 3scale" link.

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

The screenshot shows the Red Hat 3Scale API Management interface. The left sidebar is dark and contains navigation items: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area has a light background. At the top, it says "Integration". Below that is a sub-section titled "API" which includes a "Private Base URL" input field containing "https://echo-api.3scale.net:443" with a note below it: "Private address of your API that will be called by the API gateway.". Below this is another section titled "API GATEWAY" with two "Public Base URL" inputs: "Staging Public Base URL" containing "https://api-user1-apicast-staging.apps.openbanking-4f6a.openshiftworkshop.com:443" and "Production Public Base URL" containing "https://api-user1-apicast-production.apps.openbanking-4f6a.openshiftworkshop.com:443", both with notes below them: "Public address of your API gateway in the staging environment." and "Public address of your API gateway in the production environment.". There are also sections for "MAPPING RULES" and "AUTHENTICATION SETTINGS" at the bottom.

We have the section where we map the Backend API 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 address 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.openbanking-4f6a.openshiftworkshop.com>)

The screenshot shows the Red Hat 3scale API Management interface. On the left, a sidebar menu includes Overview, Analytics, Applications, ActiveDocs, Integration, Configuration (selected), Methods & Metrics, and Settings. The main content area has tabs for API: API (selected) and API GATEWAY. Under API GATEWAY, there are sections for Private Base URL (https://echo-api.3scale.net:443), Staging Public Base URL (https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443), and Production Public Base URL (https://api-user101-apicast-production.apps.openbanking-4f6a.openshiftworkshop.com:443). Below these is a section for MAPPING RULES, which contains a table with one row: Verb (GET), Pattern (/open-data/banks), and Metric or Method (Define) (hits). A button for 'Add Mapping Rule' is visible.

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).

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 resource). We can now check the configuration of authentication settings.

The screenshot shows the Red Hat 3scale API Management interface. The sidebar menu is identical to the previous screenshot. The main content area is titled 'AUTHENTICATION SETTINGS'. It includes fields for Host Header (empty), Secret Token (Shared\_secret\_sent\_from\_proxy\_to\_API\_backend\_5c9f6c92c06c2efa), and Credentials location (radio buttons for 'As HTTP Headers' (selected) and 'As query parameters (GET) or body parameters (POST/PUT/DELETE)'). Below these is a field for Auth user key (key). At the bottom, there is a section for 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 API management interface with a sidebar menu on the left. The main area displays two sections for customizing error responses:

- 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

Explain that you can also customize the error returned to the end user. **Policies**

The screenshot shows the API management interface with a sidebar menu on the left. The main area displays the Policies section and a client test request:

- POLICIES**
  - Policy Chain
  - [3scale APICAST](#) builtin - Main functionality of APICAST to work with the 3scale API mana...
- CLIENT**
  - API test GET request: /
  - 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:

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

At the bottom right, there are buttons for "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 API gateway's configuration interface. On the left, there's a sidebar with navigation links: Overview, Analytics, Applications, ActiveDocs, Integration (with sub-options Configuration, Methods & Metrics, Settings), and another Integration section. 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. Highlight the presence of SOAP policy to map SOAP services and advanced rate limit functionalities, rate limit policy. Update the API test GET request field with the same pattern you are mapping above (excluding the \$).

The screenshot shows the API gateway's configuration interface. The sidebar is identical to the previous screenshot. The main area has two sections:

- POLICIES**: Shows a 'Policy Chain' section with a link to '3scale APICAST'.
- CLIENT**: Shows an 'API test GET request' field containing a curl command:

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

Below the field is a note: "Hit the test button to check the connections between client, gateway & API."

At the bottom right are buttons: 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

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

- Update the service configuration
- 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 Red Hat 3Scale API Management interface. On the left, there's a sidebar with navigation links: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area has tabs for Response Body (No Mapping Rule matched) and Policies. Below that is a Client section titled 'APItest GET request' with the URL '/open-data/banks'. A note 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.' Below this is a curl command: 'curl "https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443/open-data/banks" -H key: USER\_KEY''. At the bottom, a note says 'Something is not quite ok. Is your private API host reachable from the API gateway?' and 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.

Now from the Service overview page click the green link *Create application plan*. Since we are creating a Service we will need to offer a way for Developers to subscribe to it and use it. Application plan are the way to do that (also called API Contracts).

The screenshot shows the Service overview page. The sidebar includes Overview, Analytics, Applications, ActiveDocs, and Integration (selected). The main content area has sections for Latest Apps (There are no latest applications), Latest alerts (There are no alerts), and Published Application Plans (There are no published application plans. Create one!). A green button labeled '+ Create Application Plan' is visible. To the right, there's a section for Configuration, Methods and Settings with a note: '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.' It also lists: 'Configure APICast', 'Authenticated by API key', 'ID for API calls is 104 and system name is fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007-l-db2rest', 'Users can manage application keys', 'Users can manage applications', and 'Users can request plan change'.

Fill out the 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.

**Create Application Plan**

Name

System name

Only ASCII letters, numbers, dashes and underscores are allowed.

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

**Create Application Plan**

**Application Plans**

Application Plans establish the rules (limits, pricing, features) for using your API; every developer's application accessing your API will be accessing it within the constraints of an Application Plan. From a business perspective, Application Plans allow you to target different audiences by using multiple plans (i.e. 'basic', 'pro', 'premium') with different sets of rules.

Name	Applications	State	
Basic	0	hidden	<a href="#">Publish</a> <a href="#">Copy</a> <a href="#">Delete</a>

**Create Application Plan**

We see that we have 1 API contract (or Application Plan), but no application associated to 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"/>

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

We can now switch to the Audience tab to create an Application to test the Configuration.

The screenshot shows the 'Accounts' listing page. On the left is a sidebar with links: 'Accounts' (selected), 'Listing' (selected), 'SETTINGS', 'Usage Rules', 'Fields Definitions', 'Applications', 'Billing', 'Developer Portal', and 'Messages'. The main area has a heading 'Accounts' and 'Bulk operations'. It says 'You have selected 1 accounts and you can make following operations with them:'. Below are three buttons: 'Send email' (Send email to selected accounts), 'Change account plan' (Transfer these accounts to different account plan), and 'Change state' (Approve, reject or make pending selected accounts). A table follows with columns: 'Group/Org.', 'Name', 'Signup Date', 'Apps', 'State', and a 'Create' button. The table shows one row for 'Developer' with 'John Doe' as the name, '18 Jan, 2019' as the signup date, '1' as the number of apps, and 'Approved' as the state. A search bar and a 'Search' button are at the bottom. A link 'Export all Accounts' is also present.

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

The screenshot shows the 'Account: Developer' details page. The sidebar is identical to the previous one. The main area has a header 'Account: Developer' with an 'Edit' link. It shows basic account information: Organization/Group Name (Developer), Administrator (John Doe, email: admin+test@user101.com), Signed up on (January 18, 2019 23:47), and Status (Approved). Below this is a 'Billing Status' section with a note that monthly billing is enabled and a 'Disable' link. To the right are two boxes: 'Account Plan: Default' (with a 'Convert to a Custom Plan' link) and 'Application' (listing Name: test, Service: API, Plan: Basic, State: Live). A note at the bottom says 'Open "https://user101-admin.apps.openbanking-4f6a.openshiftworkshop.com/p/admin/dashboard" in a new tab'.

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 **0 Applications** and **Create application**

The screenshot shows the 'Applications' listing page. The sidebar is identical to the previous ones. The main area has a header 'Account 'Developer'' > '0 Applications' | '1 User' | '0 Invitations' | '0 Group Memberships' | '0 Invoices' | '1 Service Subscription'. It has a table with columns: 'Name', 'State', 'Plan', 'Paid?', 'Created At', 'Traffic On', and a 'Create Application' button. The table is currently empty. A search bar and a 'Search' button are at the bottom.

Account 'Developer' > 1 Application | 1 User | 0 Invitations | 0 Group Memberships | 0 Invoices | 1 Service Subscription

## New Application

Application plan

Service plan

Name

Description

Create Application

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**

CLIENT

API test GET request

/

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:

```
curl "https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443/?user_key=9457a3f7a5ea6dd3e46bc5ab565385ad"
```

Hit the test button to check the connections between client, gateway & API.

Update & test in Staging Environment

Back to Integration & Configuration

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

CLIENT

API test GET request

```
/
```

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:

```
curl "https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443/?user_key=af44089cd553a3b0515772972b5f9ced"
```

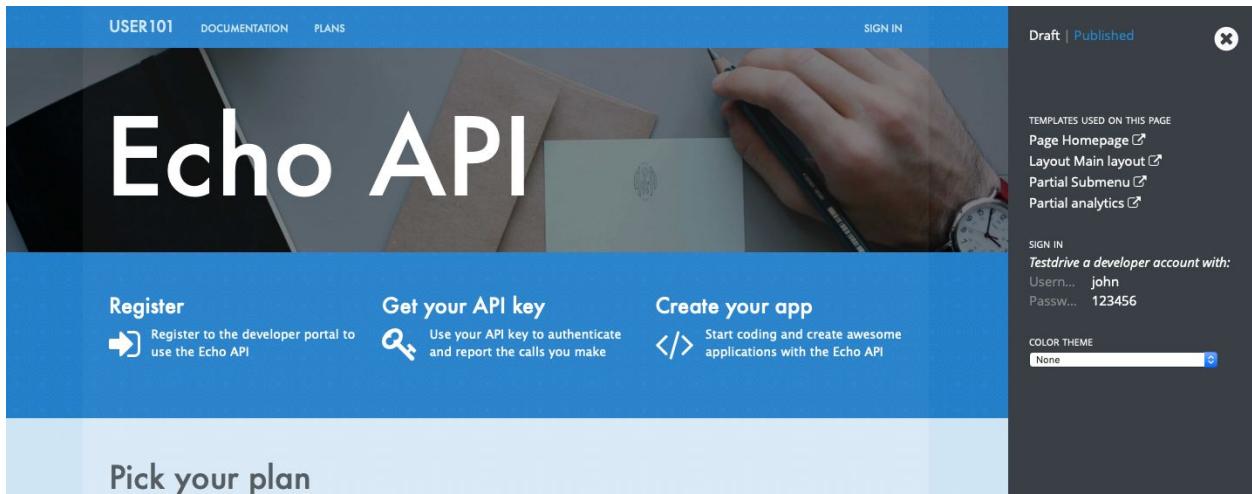
*Connection between client, gateway & API is working correctly as reflected in the analytics section.*

[Update & test in Staging Environment](#)  
[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. **Developer portal -> Visit Developer Portal**

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



Let's sign in with the default user credentials provided in the sidebar. john / 123456

The screenshot shows the Echo API developer dashboard. At the top, there's a navigation bar with links for 'USER101', 'API CREDENTIALS', 'STATISTICS', 'DOCUMENTATION', 'MESSAGES', 'SETTINGS', and a user icon. A message 'SIGNED IN SUCCESSFULLY' is displayed. Below the header, a large banner features the text 'Echo API' over a background image of a hand writing on a piece of paper. Underneath the banner, a section titled 'Your API Key' contains the text: '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 includes a note: 'Add this as a `user_key` parameter to your API calls to authenticate.' There's a 'Regenerate' button at the bottom of the modal.

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

The screenshot shows the 'Applications' section of the Echo API developer dashboard. It lists one application named 'Test' with the following details:

Name	Test
Description	Test
Plan	Basic > <a href="#">Review/Change</a>
Status	<span style="color: green;">Live</span>
User Key	<code>af44089cd553a3b0515772972b5f9ced</code>

Below the table, there's a note: 'Add this as a `user_key` parameter to your API calls to report and authenticate.' and a 'Regenerate' button.

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

key | 4c571db87a82b5aedccbd8 |

+ Add Step

**PASS**

Results 576 ms N. Virginia  
Seconds elapsed: 2

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: w2-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: opentry/1.13.6.1
Date: Fri, 11 Jan 2019 18:04:18 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 5076
Set-Cookie: JSESSIONID=y0pray2w644kluns8ikk5f1uy; Path=/
Expires: Fri, 11 Jan 2019 18:04:18 GMT
```

Use the URL for your API gateway, the following format should be configured in your service already: <https://userX.amp.apps.openbanking-4f6a.openshiftworkshop.com> the key Header and related value. 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.

key | 4c571db87a82b5aedccbd8 |

+ Add Step

**PASS**

Results 111 ms N. Virginia  
Seconds elapsed: 2

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: w2-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: opentry/1.13.6.1
Date: Fri, 11 Jan 2019 18:10:42 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: Chunked
Set-Cookie: 6ec052093408095c20c301d764adc8ddb-fd85f9e6e21ee0a00f6cad673c7c9e5b; 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 controlled service with OIDC. API key is not considered a safe method anymore and is vulnerable to many attacks.

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

SINCE AS INTEGR8LY USERS YOU DON'T HAVE ACCESS TO THE RELATED RH SSO REALM, YOU ARE GOING TO SEE HOW TO CONFIGURE A RH SSO CLIENT THAT WILL THEN BE USED BY EVERYBODY IN THEIR 3SCALE CONFIGURATION. AS ADMIN YOU WILL NEED TO GET THE USER AND PASSWORD FROM THE ENVIRONMENT VARIABLES OF SSO TO ACCESS THE ADMIN CONSOLE OF RH SSO. YOU WILL NEED TO LOGIN TO OPENSHIFT AS ADMIN.

Let's start with RH SSO main dashboard

<https://sso-test.apps.openbanking-4f6a.openshiftworkshop.com>

Alternatively

<http://sso-rhsso.apps.openshift.es>

The screenshot shows the RHSSO Admin Console interface. On the left, there is a sidebar with navigation links: 'Configure' (selected), 'Realm Settings' (selected), 'Clients', 'Client Templates', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. Under 'Manage', there are links for 'Groups', 'Users', 'Sessions', 'Events', 'Import', and 'Export'. The main content area is titled 'OpenShift' and has tabs for 'General', 'Login', 'Keys', 'Email', 'Themes', 'Cache', 'Tokens', 'Client Registration', and 'Security Defenses'. The 'General' tab is selected. It contains fields for 'Name' (set to 'openshift'), 'Display name', 'HTML Display name', 'Enabled' (set to 'ON'), and 'Endpoints' (set to 'OpenID Endpoint Configuration'). At the bottom of the form are 'Save' and 'Cancel' buttons. The top right corner shows a user icon and 'Admin'. The bottom of the screen shows a browser toolbar with multiple tabs and a page number indicator '1 / 2'.

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",
  "check_session_iframe": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/certs",
  "grant_types_supported": [
    "authorization_code",
    "implicit",
    "refresh_token",
    "password",
    "client_credentials"
  ],
  "response_types_supported": [
    "code",
    "token",
    "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		
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 applications that will authenticate using RH SSO as an IDP. As we can see there are some default clients dedicated to authentication in the integr8ly environment.

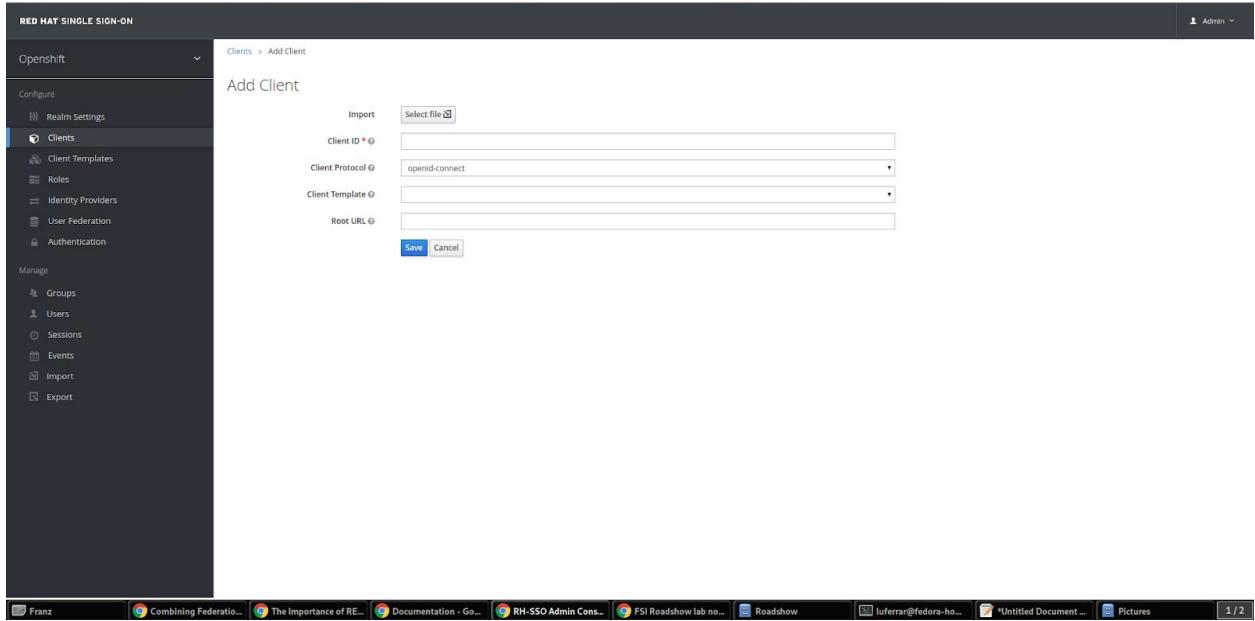
Users -> View all users

Here we can see all the users that are stored inside RH SSO, making it act as an IDM as well. These are the end users of the applications created in Clients. Let's explore one of these users.

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 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**



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

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

## Save -> Service account roles

Add manage-clients to the allowed this special client to create application on behalf of API management

And now we are ready to use these application credentials inside 3scale OIDC configuration section.

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

The screenshot shows the 'Configuration' section of the APIcast interface. On the left, a sidebar lists 'Overview', 'Analytics', 'Applications', 'ActiveDocs', 'Integration' (selected), 'Configuration', 'Methods & Metrics', and 'Settings'. The main area has a title 'Configuration' with a 'edit integration settings' link. It contains two sections: 'Integration settings' (Deployment Option: APIcast, Authentication: API Key (user\_key)) and 'APICAST Configuration' (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). Below this is an 'Environments' section with a 'Staging Environment' card (URL: https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443, version: v. 4) and a 'Promote v. 4 to Production' button.

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**

The screenshot shows the 'Authentication' selection screen. The sidebar is identical to the previous one. The main area has a heading 'AUTHENTICATION' and a 'Authentication' section with the note: 'Authentication is essential to provide Access Control. The chosen authentication mode dictates how your customers will authenticate with your API.' Three options are listed: 'API Key (user\_key)' (selected, indicated by a green checkmark), 'App\_ID and App\_Key Pair' (indicated by a grey checkmark), and 'OpenID Connect' (indicated by a grey checkmark). A 'Update Service' button is at the bottom right.

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.

**Configuration**

[edit integration settings](#)

**Integration settings**

Deployment Option	APICast
Authentication	OpenID Connect

[edit APICast configuration](#)

**APICast Configuration**

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**

[Configuration history](#)

**Staging Environment**

https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443  
v. 4

[Promote v. 4 to Production](#)

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**

**AUTHENTICATION SETTINGS**

**OpenID Connect Issuer**  
https://sso.example.com/auth/realm/gateway

Location of your OpenID Provider. The format of this endpoint is determined on your OpenID Provider setup. A common guidance would be "https://<CLIENT\_ID>:<CLIENT\_SECRET>@<HOST>:<PORT>/auth/realm/<REALM\_NAME>".

**Host Header**

Lets you 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\_1664772f8f98f392

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)

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@<rh-sso-public-endpoint>>

where client-id: sync-app

client secret: 1b6886b5-db99-4343-a6ca-73ef8fd25b95

rh-sso-public-endpoint: <http://sso-rhsso.apps.openshift.es/auth/realm/3scale-auth>

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

**APIcast Configuration**

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**

**Staging Environment**  
https://user101.amp.apps.openbanking-4f5a.openshiftworkshop.com:443  
v. 5

**Promote v. 5 to Production**

**Production Environment**  
no configuration has been saved for the production environment yet

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

The screenshot shows a web-based application management interface. At the top, there's a blue header bar with the text "USER101" and navigation links for "API CREDENTIALS", "STATISTICS", and "DOCUMENTATION". Below the header, the main content area has a light blue background. On the left, there's a sidebar with a back arrow and the text "Applications". On the right, there's an "Edit Test" button with a gear icon. The main content area displays a configuration for an application named "Test". The configuration includes:

- Name:** Test
- Description:** Test
- Plan:** Basic > [Review/Change](#)
- Status:** Live
- Client ID:** **5zdf2946**  
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.  
  
[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).

USER101 API CREDENTIALS STATISTICS DOCUMENTATION

◀ Applications Edit test

Name: test

Description:

Plan: Basic > Review/Change

---

Status: Live

---

**Client ID:** 245e2c45  
This is the Client ID you should send with each API request.

**Client Secret:** 84f7fc184b5210f98bba2290a0474fd3  
This is the Client Secret used to authenticate requests.  
Regenerate

**Redirect URL:** This is your Redirect URL for OAuth.

REDIRECT URL:

Submit

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

**Application 'Test' | Analytics**

### Test [Edit](#)

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

### API Credentials

Client ID	52df2946
Client Secret	ea2494e0efbfdd7b4ae2d77062a60f3 <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



Hits  
1 hit

**Application Plan: Basic**

[Convert to a Custom Plan](#)

**FEATURES**

- Unlimited Greetings (✓)
- 24/7 support (✗)
- Unlimited calls (✗)

**Change Plan**

[Change Plan](#)

The screenshot shows the RHSSO interface under the 'Clients' section. A client named '5bc94f6a' is selected. The configuration details are as follows:

- Client ID:** 5bc94f6a
- Name:** my first finance app - sz
- Description:** my first finance app - sz
- Enabled:** ON
- Consent Required:** OFF
- Client Protocol:** openid-connect
- Client Template:** (empty)
- Access Type:** confidential
- Standard Flow Enabled:** ON
- Implicit Flow Enabled:** OFF
- Direct Access Grants Enabled:** OFF
- Service Accounts Enabled:** OFF
- Root URL:** (empty)
- Valid Redirect URIs:** https://openidconnect.net/callback
- Base URL:** (empty)
- Admin URL:** (empty)
- Web Origins:** (empty)

All seems 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-rhsso.apps.openshift.es/auth/realms/3scale-auth/.well-known/openid-configuration>

We are going to use the client id and secret as from the application in the developer portal.

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

The screenshot shows the 'OpenID Connect Configuration' dialog box. The configuration fields are as follows:

- Server Template:** Custom
- Discovery Document URL:** https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/.well-known/openid-configuration
- Authorization Token Endpoint:** https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/token
- Token Endpoint:** https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/token
- Token Keys Endpoint:** https://secure-sso-sso.apps.open-banking.opentry.me/auth/realms/openshift/protocol/openid-connect/certs
- OIDC Client ID:** 5bc94f6a
- OIDC Client Secret:** 2b6db299110348dbae6134e97a8d0350
- Scope:** openid email

A note at the bottom of the dialog box reads: "Remember to set https://openidconnect.net/callback as an allowed callback with your application."

Start the authentication flow by hitting start. You are going to be redirected to the RH SSO login interface where you can use a default user details and password (luca / password). Once you login you will receive a temporary code to be exchanged for the final credentials or access token.

The screenshot shows a web application interface. At the top, a yellow circle with the number '2' contains the text 'Exchange Code from Token'. Below this, a section titled 'Your Code is' contains a long string of characters: eyJhbGciOiJkaXIlCJlbmM10JBMTI40BjDLUhTMjU2In0..VeAdYXzdHkLQLwD- FZLOPoP01m0we70KDjMI\_32RqnrpcixM08YQePoPt516NiKjSe8hZwlnhBj4Rz0zMS3WcukRf06m3LzLz0Zot-X14eJgobGP;SFPP37PuFtkZ-3n7aubNDyF\_ZK5msqY7Ir7xv0K3vIKCP1oF1ZRcY4\_my\_oypHCbeJa.jLF. Now, we need to turn that access code into an access token, by having our server make a request to your token endpoint.

A 'Request' box contains the following POST data:

```
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=2b6db299110348dbae0134e97a8d0359
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXIlCJlbmM10JBMTI40BjDLUhTMjU2In0..VeAdQlWd- FZLOPoP01m0we70KDjMI_32RqnrpcixM08YQePoPt516NiKjSe8hZwlnhBj4Rz0zMS3WcukRf06m3LzLz0Zot-X14eJgobGP;SFPP37PuFtkZ-3n7aubNDyF_ZK5msqY7Ir7xv0K3vIKCP1oF1ZRcY4_my_oypHCbeJa.jLF
```

At the bottom of the request box is a button labeled 'EXCHANGE'.

## Hit Exchange

The screenshot shows the result of the 'EXCHANGE' button click. The response is displayed in a 'Request' box:

```
HTTP/1.1 200
Content-Type: application/json
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInRScCig0AiAisldUiwi",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCig0AiAisldUiwi",
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCig0AiAisldUiwi2lk",
  "not-before-policy": 0,
  "session_state": "fa149bb8-d4e9-49be-95b0-9613ff0a55bd",
  "scope": ""
}
```

At the bottom of the response box is a button labeled '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. We can see that another important piece of information is shown there regarding when this credential will expire. We can hit **NEXT** and id\_token will also be shown, which contains the 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](#)

```
eyJhbGciOiJSUzIiNiIsInR5cC Ig0AiSldUIwi a2IkIA6ICJRa1RX2VwS2IwNvpFSkp3ZT d1cnFQ
20B9v5v17D-
CDJpLxOcbVRgNkqBoBb9K Fwv1y w81616X6B38rJA35WTCZkyfUs5ruIm
LNpEP0D1YxNg7mgjKKYV8bMu8yB7WsvCgtvF_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-ssso.apps.open-banking.opentrue.me/auth/realm/openshift/protocol/openid-connect/certs>, which is found in the discovery document or configuration menu options.

[VERIFY](#)

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

The screenshot shows the JUUT debugger interface with the following details:

- HEADER: ALGORITHM & TOKEN TYPE**

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "OKTI_epKb05ZEJJwe7urqPQkchDMF-R2xFpMeeBvh-U"
}
```

- PAYOUT: DATA**

```
{
  "jti": "7dcd7531-78ea-41ea-ad31-1502a8aca437",
  "exp": 1547396457,
  "nbf": 0,
  "iat": 1547396157,
  "iss": "https://secure-sso-ssso.apps.open-banking.opentrue.me/auth/realm/openshift",
  "aud": "Shc9416a",
  "sub": "33ed4a3-50c6-47b4-ac3d-987bb0e9836",
  "typ": "ID",
  "azp": "Shc9416a",
  "auth_time": 1547395876,
  "session_state": "fa199b8d-4de9-49be-95b8-9613ff0a55bd",
  "acr": "A",
  "preferred_username": "evals90@example.com",
  "email": "evals90@example.com"
}
```

- VERIFY SIGNATURE**

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to use this token
```

Let's now go back to our OpenID client and copy the access token long string.

It should look something like this:

```
eyJhbGciOiJSUzIiNiIsInR5cC Ig0AiSldUIwi a2IkIA6ICJRa1RX2VwS2IwNvpFSkp3ZT d1cnFQ
20B9v5v17D-
CDJpLxOcbVRgNkqBoBb9K Fwv1y w81616X6B38rJA35WTCZkyfUs5ruIm
LNpEP0D1YxNg7mgjKKYV8bMu8yB7WsvCgtvF_XBx9K3g
```

The long string above is the access token long string, which needs to be used as a header in the call towards the OpenID protected service.

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>

The screenshot shows the API Tester interface with a green header bar. Below it, a large central area titled "Build your test" contains a "Request" section. The request method is set to "GET" and the URL is "https://wt2-evals99-example-com-3scale.apps.open-banking.opentry.me". Under the "Headers" section, there is an "Authorization" field containing the value "Bearer eyJhbGciOiJSUzI1NiI". A "View example" link is located just above the request section.

## Let's hit Test

The screenshot shows the results of the test. The "Request" section displays the same GET request details. The "Response" section shows the full response headers and body. The response headers include standard HTTP headers like "Server", "Content-Type", and "Content-Length", along with specific OpenBankProject headers such as "Set-Cookie" and "X-Content-Type-Options". The response body is a JSON array of bank objects:

```
{
  "banks": [
    {
      "id": "psd201-bank-x--uk",
      "short_name": "Bank X",
      "full_name": "The Bank of X",
      "logo": "https://static.openbankproject.com/images/sandbox/bank_x.png",
      "website": "https://www.example.com",
      "bank_routing": {
        "scheme": "OBP",
        "address": "psd201-bank-x--uk"
      }
    },
    {
      "id": "psd201-bank-y--uk",
      "short_name": "Bank Y",
      "full_name": "The Bank of Y",
      "logo": "https://static.openbankproject.com/images/sandbox/bank_y.png",
      "website": "https://www.example.com",
      "bank_routing": {
        "scheme": "OBP",
        "address": "psd201-bank-y--uk"
      }
    },
    {
      "id": "at02-bank-x--01",
      "short_name": "Bank X",
      "full_name": "The Bank of X",
      "logo": "https://static.openbankproject.com/images/sandbox/bank_x.png",
      "website": "https://www.example.com",
      "bank_routing": {
        "scheme": "OBP",
        "address": "at02-bank-x--01"
      }
    },
    {
      "id": "at02-bank-y--01",
      "short_name": "Bank Y",
      "full_name": "The Bank of Y",
      "logo": "https://static.openbankproject.com/images/sandbox/bank_y.png",
      "website": "https://www.example.com",
      "bank_routing": {
        "scheme": "OBP",
        "address": "at02-bank-y--01"
      }
    },
    {
      "id": "at02-2080--01",
      "short_name": "Abanca",
      "full_name": "ABANCA CORPORACION BANCARIA, S.A.",
      "logo": "https://static.openbankproject.com/images/sandbox/bank_abanca.png",
      "website": "https://www.abanca.es"
    }
  ]
}
```

The "Variables" section at the bottom right contains a search bar with the placeholder "Search variables...".

And success!

The screenshot shows the raw JSON response from the previous test. The "Raw" tab is selected, displaying the entire JSON object. The "Parsed" tab is also visible. The JSON structure is identical to the one shown in the "Response Body" section of the previous screenshot.

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

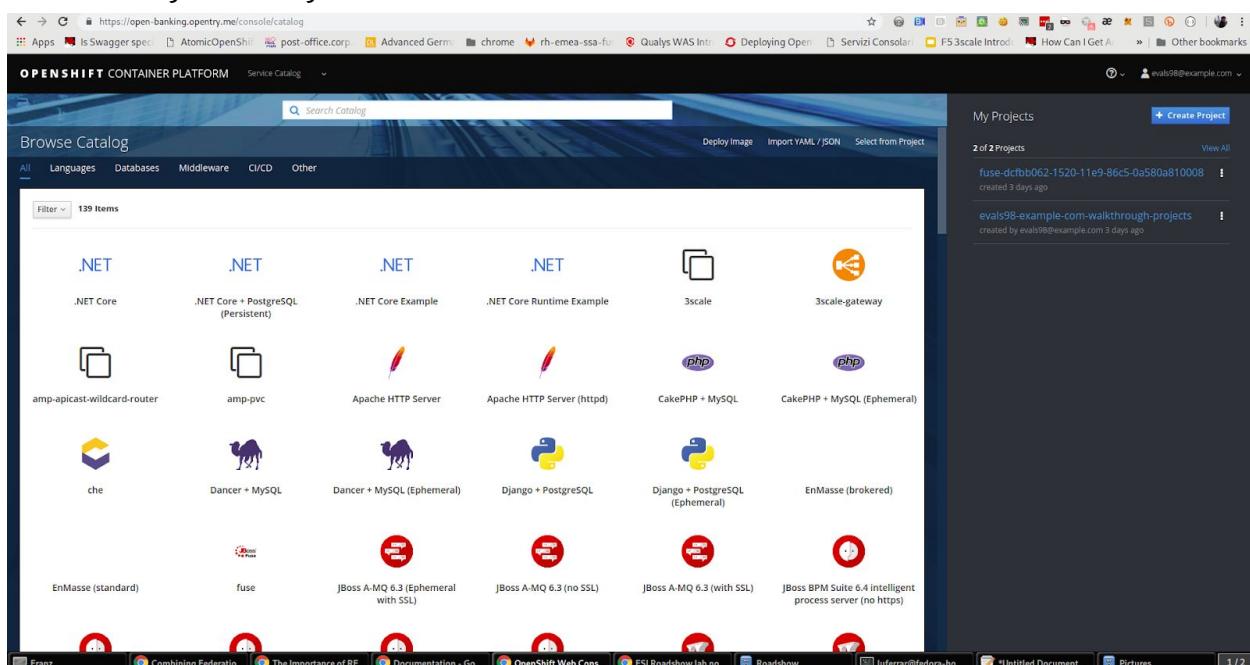
- Check for the validity of the access token credentials (not expired, legit and associated to the correct application)
- Check for rate limits on the application triggering the call
- Apply monetization rules to the call
- Apply any additional policy that might modify the call in real time

## Checkpoint

Improved security to the highest grade possible while using standards.

## OpenShift (optional)

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



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.

OPENSHIFT CONTAINER PLATFORM Cluster Console

evals98@example.com

Home Workloads Networking Storage Builds Administration Projects Service Accounts Roles Role Bindings Resource Quotas

### Projects

Create Project

Filter Projects by name...

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

Franz Combinig Federatio... The Importance of RE... Documentation - Go... Projects - OpenShift... FSI Roadshow lab no... Roadshow lufarr@example.com Untitled Document ... Pictures 1 / 2

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**

OPENSHIFT CONTAINER PLATFORM Cluster Console

evals98@example.com

Project: default

Status of default

Restricted Access

You don't have access to this section due to cluster policy.



projects.project.openshift.io "default" is forbidden: User "evals98@example.com" cannot get projects.project.openshift.io in the namespace "default": no RBAC policy matched

Home Status Search Events Workloads Networking Storage Builds Build Configs Builds Image Streams Administration Projects Service Accounts Roles Role Bindings Resource Quotas

Franz Combinig Federatio... The Importance of RE... Documentation - Go... Status of default - Op... FSI Roadshow lab no... Roadshow lufarr@example.com Untitled Document ... Pictures 1 / 2

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

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.

The screenshot shows the OpenShift Cluster Console interface. On the left, there's a sidebar with navigation links for Home, Workloads (Pods, Deployments, etc.), Networking, Storage, Builds, and Administration (Projects, Service Accounts, Roles, etc.). The main area displays a list of pods. A modal dialog box titled "Delete Pod" is open, asking if the user wants to delete the pod "syndesis-ui-1-c6gc9" from the namespace "fuse-dcfb062-1520-11e9-86c5-0a580a810008". The dialog has "Cancel" and "Confirm" buttons. Below the dialog, the pod details are shown: app=syndesis, deployment=syndesis-prometheus-1, deploymentconfig=syndesis-prometheus, syndesis.io/app=syndesis, syndesis.io/component=syndesis-prometheus, syndesis.io/type=infrastructure. The pod status is Running and Ready. Other visible pods include syndesis-operator-1-b5tr, syndesis-prometheus-1-6dcf, syndesis-server-1-jhm, syndesis-ui-1-c6gc9, and todo-1-8bd4.

This screenshot is similar to the first one, but the "Delete Pod" dialog now displays an error message: "pod \"syndesis-ui-1-c6gc9\" is forbidden: User \"evals98@example.com\" cannot delete pods in the namespace \"fuse-dcfb062-1520-11e9-86c5-0a580a810008\": no RBAC policy matched". This indicates that the user does not have the necessary permissions to delete the pod. The rest of the interface and pod list are identical to the first screenshot.

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.

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 project

Deployment Config	Mb Memory	Cores CPU	Kib Network
broker-amq_, #1	100	< 0.01	52
syndesis-db_, #1	340	< 0.01	0.1
syndesis-meta_, #1	11	< 0.01	1.5
syndesis-oauthproxy	16	< 0.01	1.8
syndesis-operator, #1	50	< 0.01	1.9
syndesis-prometheus, #1	570	0.01	41
syndesis-server, #1			
syndesis-ul_, #1			

We can also open the relative Fuse Online installation.

**OPENShift CONTAINER PLATFORM** Application Console ▾

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

APPLICATION  
syndesis

Overview Applications Builds Resources Storage Monitoring Catalog

DEPLOYMENT CONFIG broker-amq, #1

DEPLOYMENT CONFIG syndesis-db, #1

DEPLOYMENT CONFIG syndesis-meta, #1

DEPLOYMENT CONFIG syndesis-oauthproxy, #1

DEPLOYMENT CONFIG syndesis-operator, #1

DEPLOYMENT CONFIG syndesis-prometheus, #1

DEPLOYMENT CONFIG syndesis-server, #1

DEPLOYMENT CONFIG syndesis-ui, #1

https://fuse-dcfbb062-1520-11e9-86c5-0a580a810008.apps.open-banking.openshift.me:8443

Mb Memory Cores CPU Kib/s Network

100 Mb Memory < 0.01 Cores CPU 52 Kib/s Network 0 pods

100 Mb Memory < 0.01 Cores CPU 52 Kib/s Network 1 pod

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

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

16 Mb Memory < 0.01 Cores CPU 1.8 Kib/s Network 1 pod

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

570 Mb Memory 0.01 Cores CPU 41 Kib/s Network 1 pod

570 Mb Memory 0.01 Cores CPU 41 Kib/s Network 1 pod

Franz Combining Federatio... The Importance of RE... Documentation - Go... OpenShift Web Cons... FSI Roadshow lab no... Roadshow iuferrari@fedora-ho... \*Untitled Document ... Pictures 1 / 2

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.

**OPENShift CONTAINER PLATFORM** Application Console ▾

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

APPLICATION  
syndesis

Overview Applications Builds Resources Storage Monitoring Catalog

DEPLOYMENT CONFIG syndesis-oauthproxy, #1

DEPLOYMENT CONFIG syndesis-operator, #1

DEPLOYMENT CONFIG syndesis-prometheus, #1

DEPLOYMENT CONFIG syndesis-server, #1

DEPLOYMENT CONFIG syndesis-ui, #1

CONTAINERS syndesis-ui

- Image: fuse7/fuse-ignite-ui 0c05f43 100.0 MB
- Ports: 8080/TCP

Average Usage Last 15 Minutes

5.7 Mb Memory

0 Cores CPU

0.3 Kib/s Network

1 pod

NETWORKING

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

Routes - External Traffic

Create Route

DEPLOYMENT CONFIG todo, #1

Franz Combining Federatio... The Importance of RE... Documentation - Go... OpenShift Web Cons... FSI Roadshow lab no... Roadshow iuferrari@fedora-ho... \*Untitled Document ... Pictures 1 / 2

**OPENSHIFT CONTAINER PLATFORM** Application Console

Overview    Applications    Builds    Resources    Storage    Monitoring    Catalog

Pods > syndesis-ui-1-c6gc9

syndesis-ui-1-c6gc9 created 3 days ago

Status

Status:	Running
Deployment:	syndesis-ui #1
IP:	10.130.6.14
Node:	ip-172-31-3-6.ec2.internal (172.31.3.6)
Restart Policy:	Always

Container syndesis-ui

State:	Running since Jan 10, 2019 10:44:45 PM
Ready:	true
Restart Count:	0

Template

Containers

syndesis-ui

- Image: fuse7/fuse-ignite-ui:0c65f43 (100.0 MiB)
- Ports: 8080/TCP
- Mount: config-volume → /usr/share/nginx/html/config (read-write)
- Mount: default-token-qfv2w → /var/run/secrets/kubernetes.io/serviceaccount (read-only)
- Memory: 50 MiB to 255 MiB
- Readiness Probe: GET / on port 8080 (HTTP) 1s delay, 1s timeout
- Liveness Probe: GET / on port 8080 (HTTP) 30s delay, 1s timeout

Volumes

config-volume

Type: config map (populated by a config map)  
Config Map: syndesis-ui-config

default-token-qfv2w

Type: secret (populated by a secret when the pod is created)  
Secret: default-token-qfv2w

Add Storage to syndesis-ui | Add Config Files to syndesis-ui

Show Annotations

javascript:void(0)

Franz    Combining Federation...    The Importance of RE...    Documentation - Go...    OpenShift Web Cons...    FSI Roadshow lab no...    Roadshow    luferr@fedora-ho...    \*Untitled Document...    Pictures    1/2

**OPENSHIFT CONTAINER PLATFORM** Application Console

Overview    Applications    Builds    Resources    Storage    Monitoring    Catalog

Pods > syndesis-ui-1-c6gc9

syndesis-ui-1-c6gc9 created 3 days ago

Status

Status:	Running
Deployment:	syndesis-ui #1
IP:	10.130.6.14
Node:	ip-172-31-3-6.ec2.internal (172.31.3.6)
Restart Policy:	Always

Container syndesis-ui

State:	Running since Jan 10, 2019 10:44:45 PM
Ready:	true
Restart Count:	0

Confirm Delete

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!

Delete pod immediately without waiting for the processes to terminate gracefully

Template

Containers

syndesis-ui

- Image: fuse7/fuse-ignite-ui:0c65f43 (100.0 MiB)
- Ports: 8080/TCP
- Mount: config-volume → /usr/share/nginx/html/config (read-write)
- Mount: default-token-qfv2w → /var/run/secrets/kubernetes.io/serviceaccount (read-only)
- Memory: 50 MiB to 255 MiB
- Readiness Probe: GET / on port 8080 (HTTP) 1s delay, 1s timeout
- Liveness Probe: GET / on port 8080 (HTTP) 30s delay, 1s timeout

Volumes

config-volume

Type: config map (populated by a config map)  
Config Map: syndesis-ui-config

default-token-qfv2w

Type: secret (populated by a secret when the pod is created)  
Secret: default-token-qfv2w

Add Storage to syndesis-ui | Add Config Files to syndesis-ui

Show Annotations

javascript:void(0)

Franz    Combining Federation...    The Importance of RE...    Documentation - Go...    OpenShift Web Cons...    FSI Roadshow lab no...    Roadshow    luferr@fedora-ho...    \*Untitled Document...    Pictures    1/2

OPENSOURCE CONTAINER PLATFORM Application Console

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

Overview Applications Builds Resources Storage Monitoring Catalog

DEPLOYMENT CONFIG syndesis-meta, #1

DEPLOYMENT CONFIG syndesis-oauthproxy, #1

DEPLOYMENT CONFIG syndesis-operator, #1

DEPLOYMENT CONFIG syndesis-prometheus, #1

DEPLOYMENT CONFIG syndesis-server, #1

DEPLOYMENT CONFIG syndesis-ui, #1

CONTAINERS

syndesis-ui

- Image: fuse7/fuse-ignite-ui 8c05f43 1000 MB
- Ports: 8080/TCP

Average Usage Last 15 Minutes

Mib Memory Cores CPU Kib/s Network

1 pod

NETWORKING

Service - Internal Traffic

syndesis-ui

Routes - External Traffic

Create Route

Franz Combining Federatio... The Importance of RE... Documentation - Go... OpenShift Web Cons... FSI Roadshow lab no... Roadshow iuferrari@fedora-ho... \*Untitled Document ... Pictures 1 / 2

Pod syndesis-ui-1-c8gc9 was marked for deletion.

Search Catalog Add to Project



This page isn't working

fuse-dcfbb062-1520-11e9-86c5-0a580a810008.apps.openshift.opentry.me is currently unable to handle this request.

HTTP ERROR 502

Reload

Franz Combining Federatio... The Importance of RE... Documentation - Go... fuse-dcfbb062-1520... FSI Roadshow lab no... Roadshow iuferrari@fedora-ho... \*Untitled Document ... Pictures 1 / 2

**OPENSOURCE CONTAINER PLATFORM** Application Console ▾

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

Overview Applications Builds Resources Storage Monitoring Catalog

DEPLOYMENT CONFIG syndesis-meta, #1

Mb Memory < 0.01 Cores CPU 0.1 Kib/s Network

DEPLOYMENT CONFIG syndesis-oauthproxy, #1

Mb Memory < 0.01 Cores CPU 1.5 Kib/s Network

DEPLOYMENT CONFIG syndesis-operator, #1

Mb Memory < 0.01 Cores CPU 2.8 Kib/s Network

DEPLOYMENT CONFIG syndesis-prometheus, #1

Mb Memory < 0.01 Cores CPU 1.9 Kib/s Network

DEPLOYMENT CONFIG syndesis-server, #1

Mb Memory < 0.01 Cores CPU 89 Kib/s Network

DEPLOYMENT CONFIG syndesis-ui, #1

Average Usage Last 15 Minutes

CONTAINERS syndesis-ui

- Image: fuse/fuse-ignite-ui 8c05f43 1000 MB
- Ports: 8080/TCP

NETWORKING Service - Internal Traffic syndesis-ui Routes - External Traffic

create Route

1 pod

Franz Combining Federat... The Importance of RE... Documentation - Go... OpenShift Web Cons... FSI Roadshow lab no... Roadshow lufer...@fedora-ho... \*Untitled Document... Pictures 1 / 2

https://fuse-dcfbb062-1520-11e9-86c5-0a580a810008.apps.open-banking.opentrace.me

Apps Is Swagger spec AtomicOpenSh... post-office.cor... Advanced Germ... chrome rh-emea-ssa-fu Qualys WAS Int... Deploying Open... Servizi Consola... FS3scale Intro... How Can I Get... Other bookmarks evals98

**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

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

Franz Combining Federat... The Importance of RE... Documentation - Go... Ignite - Google Chro... FSI Roadshow lab no... Roadshow lufer...@fedora-ho... \*Untitled Document... Pictures 1 / 2

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

## Q&A

## Common issues

- While using Fuse Online it I have experienced some timeouts and slowness. If that happens too often you can always use the same URL that all the other attendees are using (from our open-bank demo portal) to test the API. Also make sure you use a different browser for Fuse Online scenario than the one used for Integr8ly demos
- 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>"
- The installation of RH SSO might have some certification issues, so might need to use instead a local or otherwise deployed installation