

# 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-fe8e.openshiftworkshop.com>

End Users credentials:

user[1..100] / openshift

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

<https://userX-admin.apps.openbanking-fe8e.openshiftworkshop.com>

access using username and password above

dedicated gateway

<https://userX.amp.apps.openbanking-fe8e.openshiftworkshop.com>

## Duration and format

This is a workshop designed for approximately 2 hours delivery. The focus is on the adoption of the products to build a comprehensive financial solution. This is not an introduction to the

different products used in the open banking solution portal, but it is focused on the modules and functionalities relevant for the FSI industry. Typically, instructors would talk through the introduction slides, and then for each hands-on lab, explain the steps needed to achieve the objective of the lab calling on the main functionalities of the products. At the end of each activity there will be a checkpoint with the attendees.

## Timeline

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

## Practical Part 1

Integr8ly

Login into the integr8ly environment main page

<https://tutorial-web-app-webapp.apps.openbanking-fe8e.openshiftworkshop.com>

The screenshot shows the Red Hat Solution Explorer interface. On the left, under 'Start with a walkthrough', there are 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). On the right, under 'Applications', there is a list of seven items: Red Hat OpenShift (Ready for use), Red Hat 3scale API Management Platform (Ready for use), Red Hat AMQ (Ready for use, preview), Eclipse Che (Ready for use, community), EnMasse (Ready for use), Red Hat Fuse (Ready for use), and Red Hat Developer Launcher (Ready for use, community).

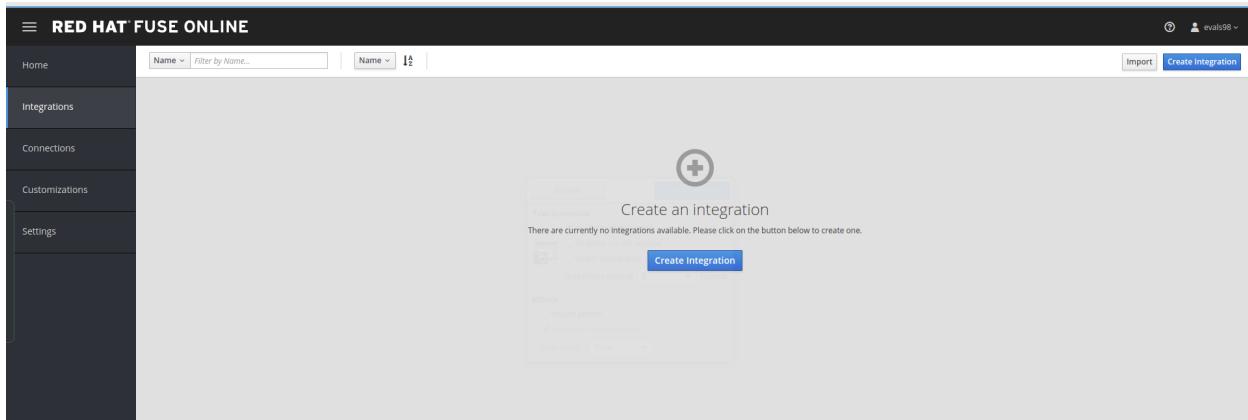
We will see how integr8ly environment works and what you get. It provides out of the box integration between products but it doesn't change the base components that come with it. You can install yourself this type of environment starting from a clean or empty OpenShift installation.

## Fuse Online

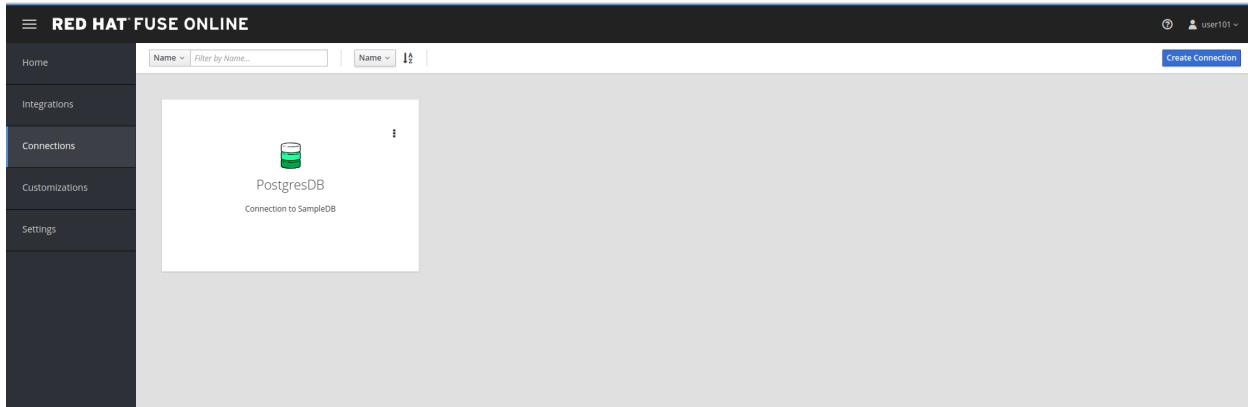
Let's start our first lab session, by opening Red Hat Fuse Online. You will get there by clicking [Red Hat Fuse](#) link in the tutorial dashboard. We will see the main functionalities of it and then use it to build a simple integration block.

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 has two sections: 'System Metrics' and 'Connections'. 'System Metrics' displays 0 Integrations (0 green, 0 red), 4 Connections, 0 Total Messages (0 green, 0 red), and Uptime (2 minutes, Since Jan 10th 22:44 PM). Below this is a 'Create an Integration' button with a plus sign icon. 'Connections' shows four categories: Database (Database icon), File (File icon), Timer (Timer icon), and Webhook (Webhook icon). There are buttons for 'View All Connections' and 'Create Connection'.

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

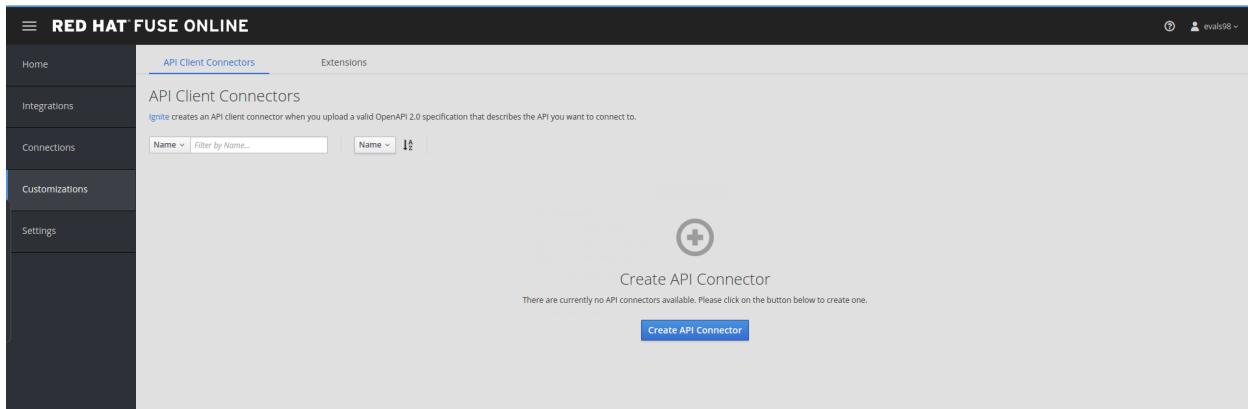


The integration window (**Integrations**) allows you to have a graphical representation of the mediation/transformation and is the first tool to onboard the citizen developers. Behind the curtains there is Camel (which is a proven technology and deployed in highly transactional environment as well) and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



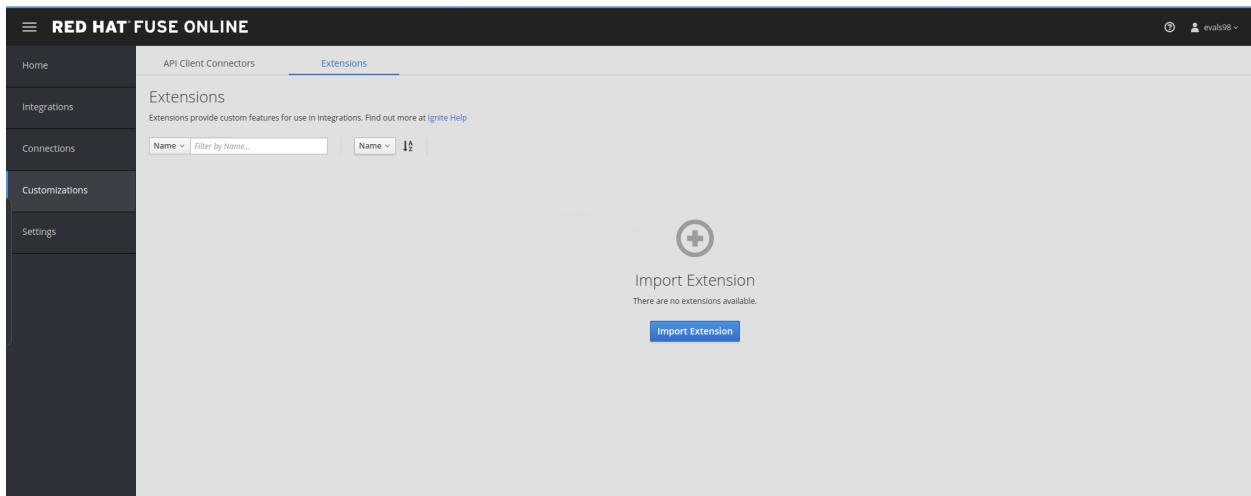
(**Connections**) you can start or end (or sometimes interpose) connections which are fully configured connectors. We will be showing the configuration of one connector during today's lab.

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



(**Customizations**) there are two panes available here, the first one dedicated to API

connections, which we will be using afterwards as Connection in the Integration.



The screenshot shows the Red Hat Fuse Online web interface. The left sidebar has navigation links: Home, Integrations, Connections, Customizations (which is selected), and Settings. The main content area is titled 'Extensions' and contains a sub-header 'API Client Connectors'. Below this, there's a search bar with dropdowns for 'Name' and 'Filter by Name...', and a button labeled 'Import Extension'. A large circular icon with a plus sign is centered above the text 'Import Extension'. Below it, a message says 'There are no extensions available.' and a blue 'Import Extension' button is visible.

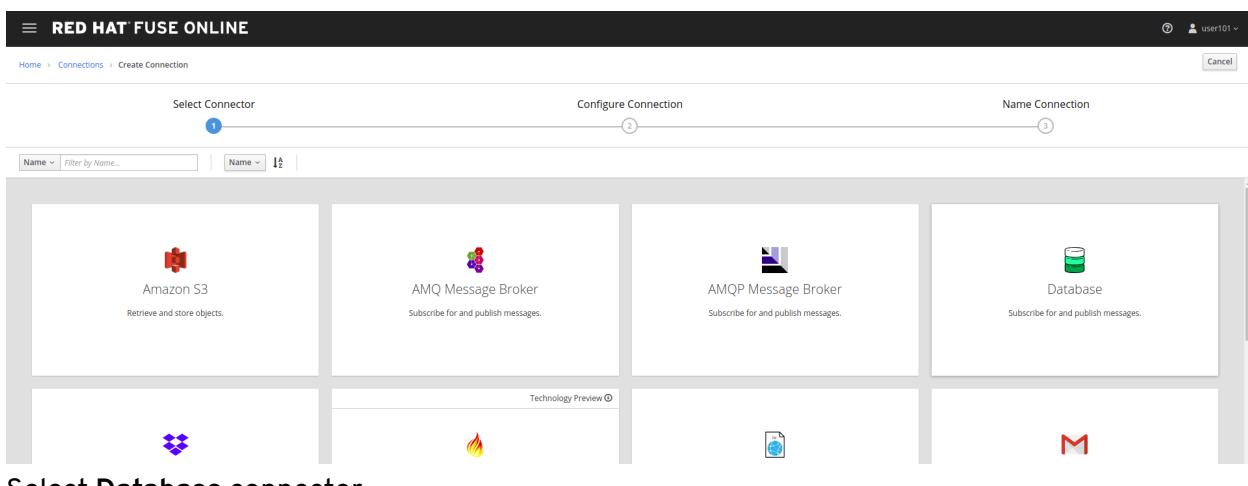
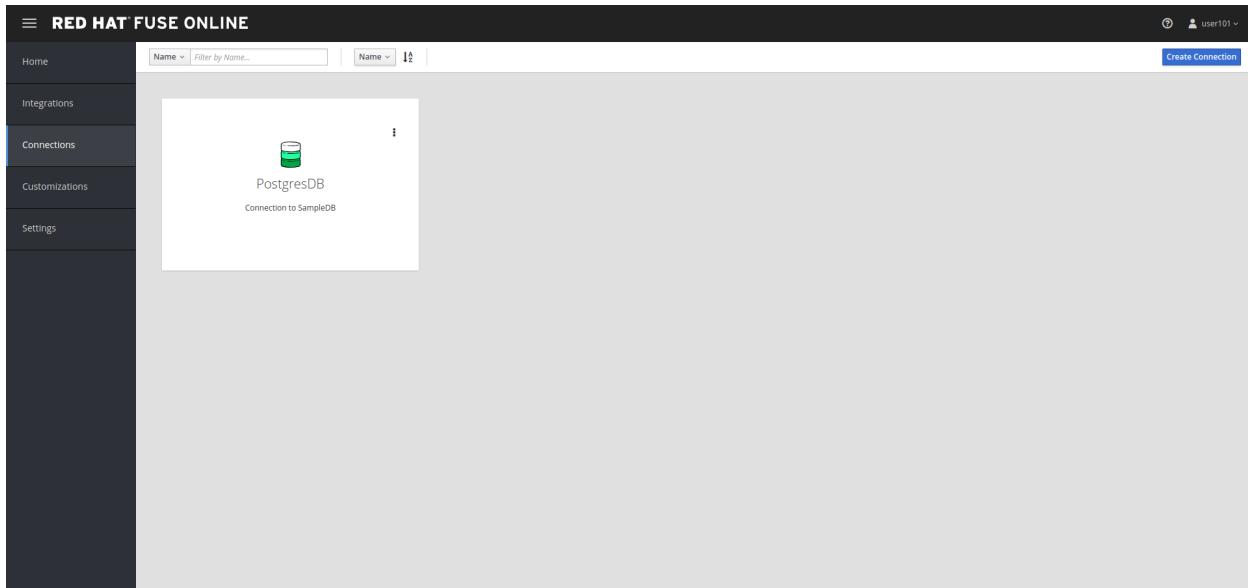
([Extensions](#) panel) this is the part where you can extend the functionalities of the platform. You can compile a custom or community available extension to perform a specific functionality or connect to a specific system and in this easy way add transformation functionalities to the product.

## LAB BEGINS

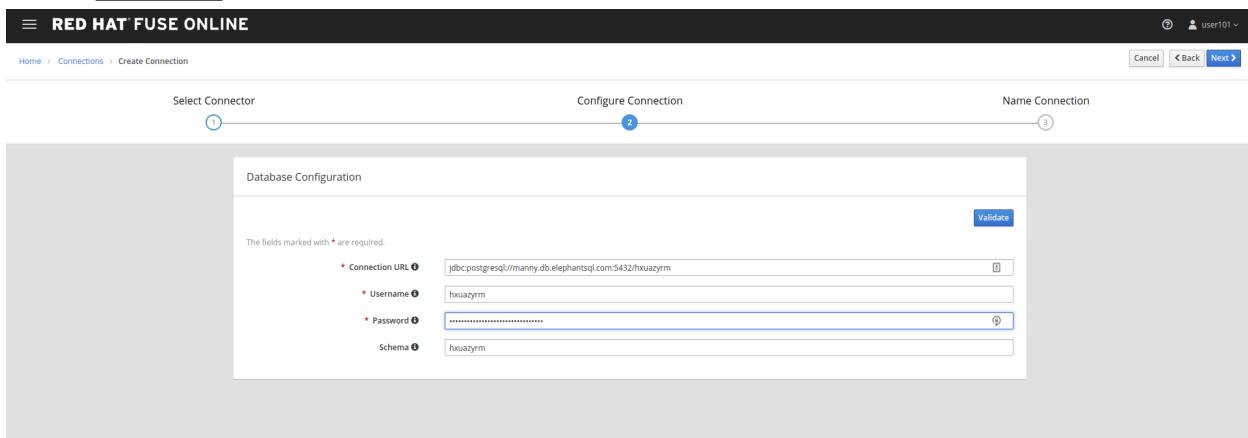
Let's start INTEGRATING now!

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

[Connections](#) -> Create connection



### Select Database connector



Use the following connection details and validate them:

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

username: `hxuazymr`

password: eiHvH6xbZ2\_DRdeoAu\_tLbNe3des\_NUf  
schema: hxuazym

The screenshot shows the 'Configure Connection' step of the connector creation process. It displays a 'Database Configuration' form with the following fields:

- Connection URL: jdbcpostgresql://manny.db.elephantsql.com:5432/hxuazym
- Username: hxuazym
- Password: (redacted)
- Schema: hxuazym

A green success message at the top indicates that the database has been successfully validated. The 'Validate' button is visible in the top right corner.

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

The screenshot shows the 'Name Connection' step of the connector creation process. It displays an 'Add Connection Details' form with the following fields:

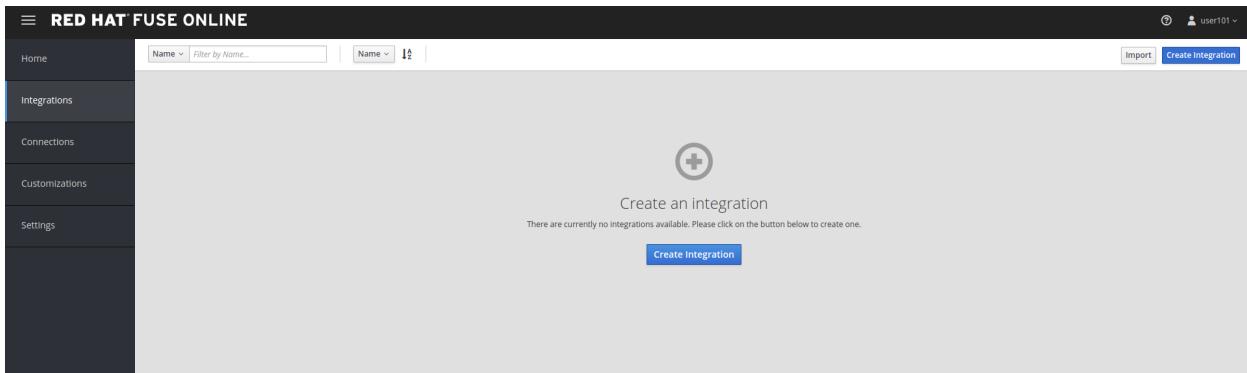
- Connection Name: open-data-banks
- Description: (empty)

The screenshot shows the Red Hat Fuse Online dashboard under the 'Connections' tab. Two connections are listed:

- open-data-banks: A connection to a PostgreSQL database named 'PostgresDB' with a description 'Connection to SampleDB'.
- PostgresDB: A connection to a PostgreSQL database named 'PostgresDB' with a description 'Connection to SampleDB'.

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

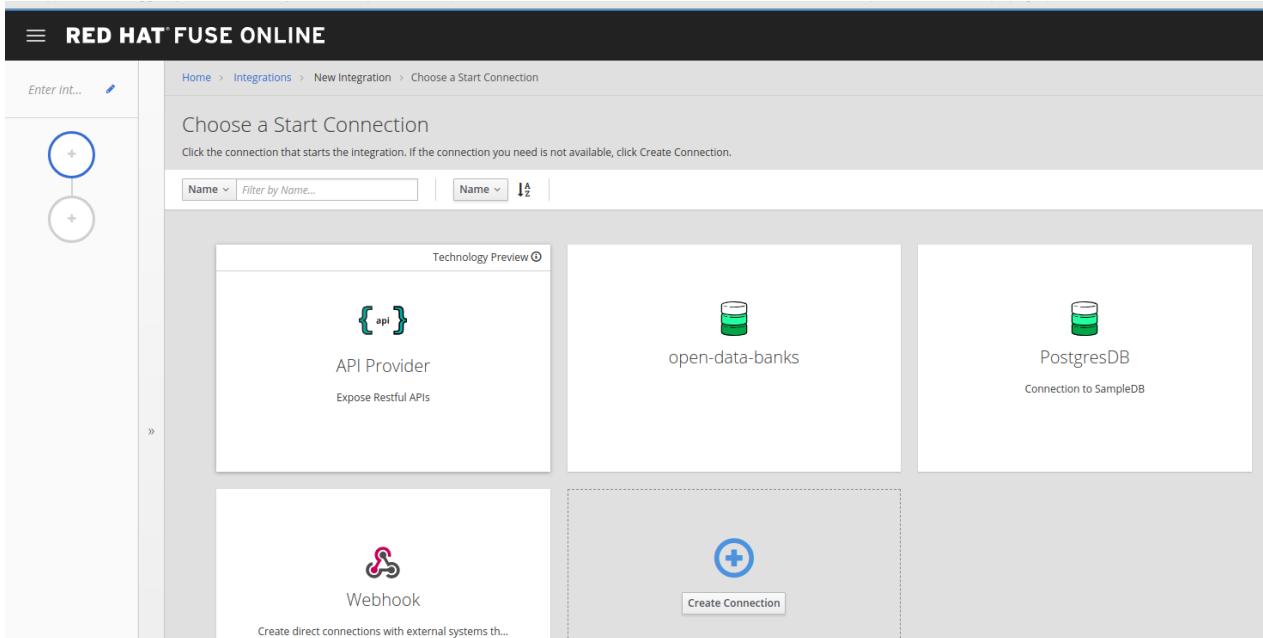
Now that we have configured the end of the integration path we want to build and we will see where to find the start and we will put the two pieces together. **Integrations -> create integration**



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

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

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



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

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

Tag	Description
ATM	No description.
Branch	No description.
Open Banking	No description.
Product	No description.
PSD2	No description.

This opens a window on Apicurito which is a scaled down version of Apicurio, our API Design platform. It is fairly easy to change elements graphically and also with the help of this tool an API team can start with a Design First approach when configuring the API. Also the same team doesn't need to know about the rules around OpenAPI specifications thanks to this tool. The service is exposed without any protection (that's one of the reasons we are going to be using 3scale later on) since the API definition is not adding any authentication on top of it by default.

Name the integration.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a search bar 'Enter int...' and a diagram icon. The main area shows the path 'Home > Integrations > New Integration > Start integration with an API call'. A title 'Give this integration a name' is followed by 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 fields for 'Integration Name' (marked with a red asterisk) and 'Description', both currently empty. At the bottom are 'Save and continue' and 'Back' buttons.

And save and continue

The screenshot shows the same interface after saving the integration name. The 'Integration Name' field now contains 'db2rest'. The other fields ('Description') are still empty. The 'Save and continue' button is highlighted in blue.

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

The screenshot shows the 'Choose operation' step for the 'db2rest' API provider. The sidebar shows 'db2rest' and 'API Provider'. The main area shows the path 'Home > Integrations > db2rest > Choose operation'. A title 'Choose operation' with a note 'Click an operation to integrate an integration flow.' is followed by a table of operations:

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

Click on get list of banks

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a tree view showing 'db2rest' and 'get list of ba...'. The main panel displays a flow diagram with a single step represented by a blue arrow pointing right. Above the diagram is a large plus sign icon and the text 'Add to Integration'. Below it, a message says 'Now you can add additional connections as well as steps to your integration.' There are buttons for 'Add a Step' and 'Add a Connection'. At the top right, there are buttons for 'Cancel', 'Go to Operation List', 'Save as Draft', and 'Publish'. The top bar also shows the user 'user101'.

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

### Add a connection

The screenshot shows the 'Choose a Connection' screen in Red Hat Fuse Online. The left sidebar shows the same integration flow as before. The main area has a heading 'Choose a Connection' and a sub-instruction 'Click the connection that completes the integration. If the connection you need is not available, click Create Connection.' Below this are two dropdown menus: 'Name' and 'Filter by Name...'. To the right, there are three connection options: 'Log' (represented by a notepad icon), 'open-data-banks' (represented by a cylinder icon), and 'PostgreSQL' (partially visible). Each option has a brief description below it. The 'open-data-banks' connection is selected.

Click on the previously configured data source.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a connection named 'db2rest' and a 'get list of ba...' entry. The main area is titled 'open-data-banks - Choose an Action' with the sub-instruction 'Choose an action for the selected connection.' Below this are two buttons: 'Invoke SQL' (which is highlighted in blue) and 'Invoke stored procedure'. There are also 'Name' dropdowns and a 'Filter by Name...' input field.

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

This screenshot shows the 'Configure Invoke SQL' step for the 'db2rest' integration. The left sidebar is identical to the previous one. The main area has a title 'open-data-banks' and a sub-instruction 'Invoke SQL'. It says 'Enter a SQL statement that starts with INSERT, SELECT, UPDATE or DELETE.' Below this is a text input field containing the SQL statement 'select \* from banks'. At the bottom right are 'Cancel', '< Choose Action', and a blue 'Done' button.

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. On the left, there's a vertical panel for the integration 'db2rest' with a step named 'get list of banks'. The main area displays a flow diagram with several nodes: a source node (green circle with a minus sign), a transformation node (blue square with a plus sign), a target node (green circle with a plus sign), and a sink node (red circle with a right-pointing arrow). A large circular button with a plus sign is centered above the flow, labeled 'Add to Integration'. Below it, text says 'Now you can add additional connections as well as steps to your integration.' and 'You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom are buttons for 'Add a Step' and 'Add a Connection'.

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

This screenshot is similar to the previous one, but the 'Add a step' dialog is open on the left side of the interface. It contains a single button labeled 'Add a step'.

Select the log step.

The screenshot shows the 'Choose a Step' dialog box. It has a search bar at the top and a list of step types below. The 'Log' step is highlighted with a black border. Other steps listed include 'Advanced Filter', 'Basic Filter', and 'Template'. The background of the main interface shows the same integration flow as the previous screenshots.

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

We are going to log just the message body.

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

You can check the progress in building the integration changing through phases.

We can notice the platform is getting the required components and constructing the block.  
When the building is completed we can test the Integration block.

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

Now that we have seen how to build the full integration you can test the integration just built,

using this online tool <https://apitester.com/>. You can use the following URL to test it <http://i-db2rest-fuse-7df78d66-28d4-11e9-aad0-0a580a010007.apps.openbanking-fe8e.openshiftworkshop.com>

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

The screenshot shows the API TESTER BETA interface. At the top, there are links for 'Sign In' and 'Create Account'. Below the header, the title 'Build your test' is displayed with a 'View example' link. A note says 'Click ⚙ to add or remove steps'. The main area contains a step editor with a single step labeled '1'. The step details are: Request type 'GET', endpoint 'no auth financial service', and URL 'https://open-data.b9ad.pro-us-east-1.openshiftapps.com/open-data/banks'. There is also a 'Headers' section with a '+ Add Request Header' button. A 'Collapse / Expand' button is located in the top right corner of the step editor. Below the step editor is a dashed box containing a '+ Add Step' button. At the bottom of the page, there are buttons for 'Test' (blue), 'Save to Account' (green), and 'Share Test Config'. A sidebar on the left lists the benefits of saving tests to the account: 'Rerun this test', 'Share test results with others', 'View the run history', and 'Create and re-run multiple tests'. The footer includes copyright information '© 2019 RIGOR. ALL RIGHTS RESERVED' and a 'Powered by RIGOR' logo.

Populate the fields with the following URL

<http://i-db2rest-fuse-7df78d66-28d4-11e9-aad0-0a580a010007.apps.openbanking-fe8e.openshiftworkshop.com/open-data/banks>

and hit **Test**

**PASS**

N. Virginia  
Seconds elapsed: 11

Results 6.29 s Viewing a Request Step 1 < >

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

Connection Information >

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

Response Response Headers HTTP/1.1 200 OK Transfer-Encoding: chunked X-Application-Context: application Date: Mon, 21 Jan 2019 18:20:33 GMT Set-Cookie: 12b28c7c44e5989665ef8abec492fd0=08271c1b765ada49ab2c2c05b8b72917; path=/; HttpOnly Cache-control: private

Response Body    

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

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

     https://uptime-diagnostics-response-bodies-production.s3.amazonaws.com/578a-6

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

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

### 3scale

To reach 3scale as a user you can just go back to the main integr8ly tutorial dashboard and click on 3scale. You will be presented with the login window and can use you previously used user credentials.

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

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

The screenshot shows the Red Hat 3Scale API Management dashboard. At the top, there are navigation links: RED HAT 3SCALE API MANAGEMENT, Dashboard, and a user icon. Below the header, there are several sections:

- AUDIENCE**: Shows 1 Signups last 30 days.
- POTENTIAL UPDATES**: Shows 0 accounts today that hit their usage limits in the last 30 days. It includes instructions to add usage limits to application plans and enables web alerts for admins.
- TODAY**: Shows "The sound of silence".
- BEFORE TODAY**: Shows "No news, good news."
- APIS**: Shows 0 Hits last 30 days.
- TOP APPLICATIONS**: Shows 0 top applications with consistently high traffic in the last 30 days. It includes instructions to have at least one application sending traffic to the API and to make test calls.
- NEW API**: A green button to start creating a 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** (click on **Authenticate to enable this option**). We are going to be using the [auto-discovery](#) feature we saw before. We would be helped by this feature and would save time configuring the service.

The screenshot shows the "New API" creation form. At the top, there are two radio button options:

- Define manually
- Import from OpenShift ([Authenticate to enable this option](#))

Below these options, there are fields for defining the API:

- SERVICE**: A section with a "Name" input field.
- System name**: An input field with a note: "Only ASCII letters, numbers, dashes and underscores are allowed."
- Description**: A text area for describing the API.

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

## Authorize Access

Bscale is requesting permission to access your account (user101)

### Requested permissions

#### user:full

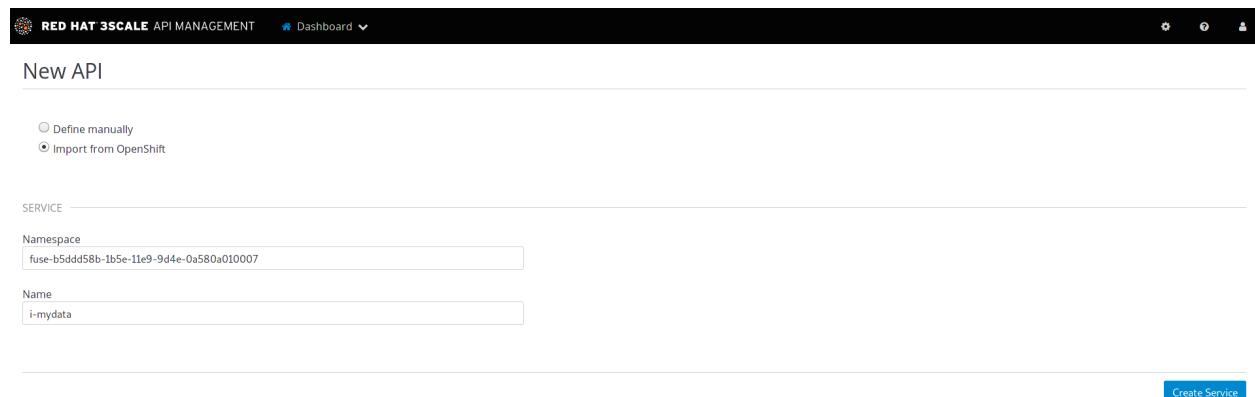
Full read/write access with all of your permissions  
Includes any access you have to escalating resources like secrets

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

[Allow selected permissions](#)

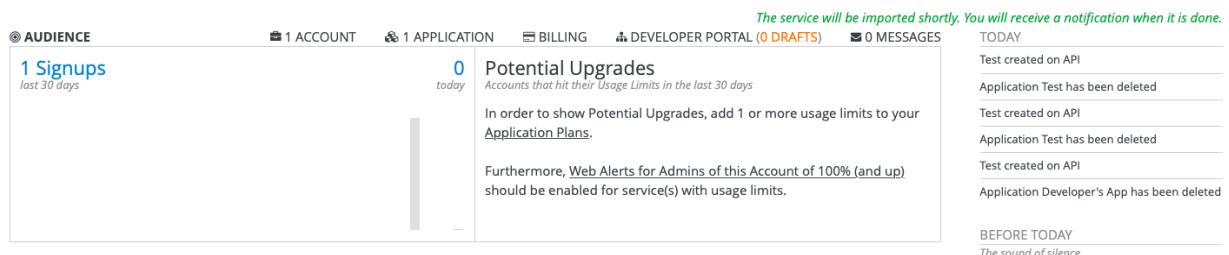
[Deny](#)

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



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, the 'Namespace' field is populated with the value 'fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007'. The 'Name' field is populated with '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 screenshot shows the Red Hat 3Scale API Management dashboard. In the top navigation bar, there are links for 'ACCOUNT', 'APPLICATION', 'BILLING', 'DEVELOPER PORTAL (0 DRAFTS)', and 'MESSAGES'. The main area has sections for 'AUDIENCE' (showing 1 Signup in the last 30 days), 'POTENTIAL UPDATES' (0 accounts today), and 'TODAY' (listing several test events). The 'BEFORE TODAY' section is empty.

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

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

## API-> Integration-> Configuration

The screenshot shows the configuration page for an integration service. 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 is titled 'Configuration' and contains a section for 'Integration settings'. It shows 'Deployment Option' set to 'APIcast' and 'Authentication' set to 'API Key (user\_key)'. A note at the bottom says 'To get started with this service on APIcast, add the base URL of your API and save the configuration.' There's also a link to 'edit integration settings'. At the bottom right, it says 'Version 2.4.0 - Powered by 3scale'.

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

The screenshot shows the 'Integration' configuration section. It includes a 'Private Base URL' input field containing 'http://i-db2rest.fuse-06b94c72-2572-11e9-9425-0a580a010007.svc.cluster.local:8080'. Below it is a note about using https. There are also sections for 'Staging Public Base URL' and 'Production Public Base URL', both currently set to 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443'. Each has a 'Use Default URL' checkbox.

We have the section where we map the Backend API (or in this case Integration service) and then 2 URLs where we expose the managed API on the staging first and production gateways or infrastructure. We will be changing the Staging and Public addresses of the gateway. In this case we are not going to use separate staging or public infrastructure so it can be the same address (please notice the format of the staging and public base url for the gateways

[https://userX.amp.apps.openbanking-fe8e.openshiftworkshop.com\)](https://userX.amp.apps.openbanking-fe8e.openshiftworkshop.com)

scheme should be https.

API GATEWAY

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

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

MAPPING RULES

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

[Add Mapping Rule](#)

AUTHENTICATION SETTINGS

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

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

Applications >

ActiveDocs

Integration

Configuration

Methods & Metrics

Settings

AUTHENTICATION SETTINGS

Host Header

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\_5c9f6c92c06c2efa

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

Credentials location

As HTTP Headers

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

Auth user key

key

ERRORS

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

The screenshot shows a sidebar menu on the left with options like Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main area displays two sections for error configuration:

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

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

The sidebar menu is identical to the previous screenshot. The main area shows 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...)
  - Add Policy
- 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, there is a note: "Hit the test button to check the connections between client, gateway & API." and 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**

Select a Policy Cancel

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

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

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

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

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

RED HAT 3SCALE API MANAGEMENT API: IDb2rest

Response Body  
No Mapping Rule matched

POLICIES

CLIENT

API test GET request  
/open-data/banks  
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:  
*In order to send a valid test request, please create an application that subscribes to an application plan of this service.*  
curl "https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443/open-data/banks" -H'key: USER\_KEY'

Something is not quite ok. Is your private API host reachable from the API gateway?

Update & test in Staging Environment  
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 known as API Contracts).

The screenshot shows the Service Overview page with a sidebar containing links for Overview, Analytics, Applications, ActiveDocs, and Integration. The main content area displays sections for Latest Apps, Latest alerts, and Published Application Plans. The Published Application Plans section includes a green 'Create Application Plan' button. To the right, there is a sidebar titled 'Configuration, Methods and Settings' with various configuration options and status information.

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

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

The screenshot shows the 'Create Application Plan' form. The sidebar on the left is identical to the previous screenshot. The main form has fields for 'Name' and 'System name'. Below these are sections for 'Applications require approval?' (with a note about setting whether applications can be created on demand or if approval is required), 'Trial Period (days)', 'Setup fee' (set to 0.00 USD), and 'Cost per month' (set to 0.00 USD). A large blue 'Create Application Plan' button is located at the bottom right of the form.

The screenshot shows the 'Application Plans' section of a developer portal. On the left is a sidebar with 'Overview', 'Analytics', 'Applications' (selected), 'Listing', 'ActiveDocs', and 'Integration'. The main area has a heading 'Application Plans' with a sub-note about establishing rules for API access. It shows a 'Default Plan' dropdown set to 'Default Plan'. Below is a table:

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.

The screenshot shows the 'Application Plan Basic' configuration page. The sidebar is identical to the previous screenshot. The main area has a heading 'Application Plan Basic' and fields for 'Name' (Basic) and 'System name' (basic). There is a checkbox for 'Applications require approval?' which is unchecked. Below are fields for 'Trial Period (days)', 'Setup fee' (0.00 USD), and 'Cost per month' (0.00 USD). A blue button at the bottom right says '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

Don't modify anything beside filling *Name* and *System name*.

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

Metric or Method (Define)	Enabled	Visible	Text only
Hits	Green	Green	Green
Pricing (0)	Green	Green	Green
Limits (0)	Green	Green	Green

**Features**

Name	Description	Enabled?	New feature
Unlimited Greetings		Green	<a href="#">Edit</a> <a href="#">Delete</a>
24/7 support		Red	<a href="#">Edit</a> <a href="#">Delete</a>
Unlimited calls		Red	<a href="#">Edit</a> <a href="#">Delete</a>

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

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

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

Account 'Developer' > 1 Application | 1 User | 0 Invitations | 0 Group Memberships | 0 Invoices | 2 Service Subscriptions

**Account: Developer** [Edit](#)

Organization/Group Name	Developer	<a href="#">Send message</a>
Administrator	John Doe ( <a href="mailto:admin+test@user101.com">admin+test@user101.com</a> )	
Signed up on	January 18, 2019 23:47	
Status	Approved	

**Billing Status**  
Monthly billing is enabled. [Disable](#)

**Account Plan: Default**  
[Convert to a Custom Plan](#)

**Application**

Name	test
Service	API
Plan	Basic
State	Live

**Hits**  
0 hits

We can see that the Developer has the default application associated, but it's subscribed to the

default Service. We can also see the Developer user details.

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

The screenshot shows the 'Create Application' page. On the left, there is a sidebar with navigation links: 'Accounts' (Listing, SETTINGS, Usage Rules, Fields Definitions), 'Applications' (Listing, SETTINGS, Usage Rules, Fields Definitions), 'Billing', 'Developer Portal', and 'Messages'. The main content area has a breadcrumb path: 'Account 'Developer'' > '1 Application' | '1 User' | '0 Invitations' | '0 Group Memberships' | '0 Invoices' | '1 Service Subscription'. Below this, the title 'New Application' is displayed. There are four input fields: 'Application plan' (set to 'Basic'), 'Service plan' (set to 'Default'), 'Name' (empty), and 'Description' (empty). At the bottom right is a blue 'Create Application' button.

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

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

The screenshot shows the 'edit Apicast configuration' page. It has two main sections: 'POLICIES' and 'CLIENT'. The 'POLICIES' section contains a 'Policy Chain' table with one item: '3scale APICAST builtin - Main functionality of APICAST to work with the 3scale API m...'. The 'CLIENT' section contains a 'CLIENT' icon and a 'CLIENT' tab. Under the 'CLIENT' tab, there is a 'Test' section with a 'GET request' URL: '/open-data/banks'. Below the URL is a note: '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 code block shows the curl command: `curl "https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks" -H 'key: 13ec073549b898fb089a9493b4c1cae'`. At the bottom, there is a note: 'Hit the test button to check the connections between client, gateway & API.' and two buttons: 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

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

The screenshot shows the 3scale APIcast interface. At the top, there is a 'Policy Chain' section with a button to 'Add Policy'. Below it, a '3scale APIcast' section displays the message: 'builtin - Main functionality of APIcast to work with the 3scale API m...'. In the 'CLIENT' section, there is a 'API test GET request' input field containing '/open-data/banks'. Below this, 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:' followed by a curl command. A green success message at the bottom states: 'Connection between client, gateway & API is working correctly as reflected in the analytics section.' On the right, there are buttons for 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

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

Let's switch to the developers' point of view by accessing the Developer portal. You can access it by selecting in to the top menu in **Audience -> Developer portal -> Visit Developer Portal**. The sidebar allows us to edit pages of the Developer Portal live, but we are not interested in it now so we can close it.

The screenshot shows the Echo API developer portal homepage. The top navigation bar includes 'USER101', 'DOCUMENTATION', 'PLANS', and 'SIGN IN'. The main content area features a large 'Echo API' logo with a background image of a person writing on a notepad. Below the logo are three calls-to-action: 'Register', 'Get your API key', and 'Create your app'. The 'Get your API key' section includes a note: 'Use your API key to authenticate and report the calls you make'. The 'Create your app' section includes a note: 'Start coding and create awesome applications with the Echo API'. A sidebar on the right is titled 'Draft | Published' and contains sections for 'TEMPLATES USED ON THIS PAGE' (listing 'Page Homepage', 'Layout Main layout', 'Partial Submenu', and 'Partial analytics'), 'SIGN IN' (with fields for 'User...' set to 'john' and 'Passw...' set to '123456'), and 'COLOR THEME' (set to 'None'). A large blue button at the bottom says 'Pick your plan'.

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

The screenshot shows the Echo API developer dashboard. At the top, there's a navigation bar with links for 'USER101', 'API CREDENTIALS', 'STATISTICS', 'DOCUMENTATION', 'MESSAGES', 'SETTINGS', and a user icon. A success message 'SIGNED IN SUCCESSFULLY' is displayed. The main content area features a large 'Echo API' logo with a background image of a hand writing on a notepad. Below the logo, the heading 'Your API Key' is shown. A text block explains that this is the API key for authentication. To the right, a modal window titled 'CREDENTIALS' shows the app details: 'App name: Test', 'Key: af44089cd553a3b0515772972b5f9ced', and instructions to add it as a 'user\_key' parameter. There's also a 'Regenerate' button.

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

The screenshot shows the 'Applications' section of the Echo API developer dashboard. It lists one application named 'Test'. The details for 'Test' are: Name: Test, Description: Test, Plan: Basic > [Review/Change](#), Status: Live (indicated by a green square). Below the application details, the 'User Key' is listed as 'af44089cd553a3b0515772972b5f9ced'. A note says to add this as a 'user\_key' parameter. A 'Regenerate' button is available. The top navigation bar is identical to the previous screenshot.

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

Sign In   Create Account   ⓘ

## Build your test

[View example](#)

Click ⚙ to add or remove steps      Collapse / Expand

Request
Step Name

GET

1 https://user100.amp.apps.openbanking-0807.openshiftworkshop.com

Headers

+ Add Request Header

key	13ece073549b898fb089a94
-----	-------------------------

+ Add Step

↻ Test
Save to Account
↗ Share Test Config

⚠ Saving tests to your account allows you to:

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

Use the URL for your API gateway, the following format should be configured in your service already: <https://userX.amp.apps.openbanking-fe8e.openshiftworkshop.com>, remember the the key Header and the associated value.

The screenshot shows the API Tester interface with a successful test run. The top section displays the configuration for a single step: a GET request to https://user100.amp.apps.openbanking-0807.openshiftworkshop.com with a key header set to 13ece073549b898fb089a94. Below this are buttons for Test, Save to Account, and Share Test Config. A note about saving tests to the account is present. The main results area shows a green bar indicating PASS, with a duration of 576 ms. It includes sections for Message (Step 1] key financial service passed), Connection Information, Request (Request Headers including Host, User-Agent, and key), and Response (Response Headers including Content-Type, Content-Length, and Set-Cookie). The response body shows a JSON object with a cookie.

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.

The screenshot shows a test results page from a cloud provider. At the top, there's a green bar indicating a 'PASS' status. Below it, the 'Results' section shows a duration of 111 ms. The 'Message' section contains the text '[Step 1] key financial service passed'. The 'Request' section displays the following details:

```

GET /open-data/banks HTTP/1.1
Host: wt2-evals98-example-com-3scale.apps.open-banking.opentry.me
Accept: /
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: c571db87a82b5aedccbd8877627a099

```

The 'Response' section shows the following headers and body:

```

HTTP/1.1 403 Forbidden
Server: openresty/1.13.6.1
Date: Fri, 11 Jan 2019 18:16:42 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: chunked
Set-Cookie: GeC552533d09895c2bc361d784adc8ddbfdb85f9e6e21ee8a00f6cad673c7c9e5b; path=/; HttpOnly; Secure

```

The 'Response Body' section shows a grid of small icons representing different data types.

As expected we receive a Forbidden error.

## Checkpoint



## Break

## Practical Part 2

### RH SSO and 3SCALE OIDC

Let's now improve the security of the managed integration service with OIDC. API key is not really considered a safe 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 OIDC CONFIGURATION.

DEMO ONLY

Let's start with RH SSO main dashboard

<http://sso.apps.openbanking-fe8e.openshiftworkshop.com/auth/>

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

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

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

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

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

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

## Users -> View all users

ID	Username	Email	Last Name	First Name	Actions		
0edb11ae-1292-4ca9-83d5-c156c5ba...	admin@example.com	admin@example.com			Edit	Impersonate	Delete
fe06b690-204c-46a8-b1f6-e119a40b...	eval01@example.com	eval01@example.com			Edit	Impersonate	Delete
31dc7465-c473-48fa-9f19-41191b63...	eval02@example.com	eval02@example.com			Edit	Impersonate	Delete
5fb5c5d69a5-41db-4f57-a8128bc7...	eval03@example.com	eval03@example.com			Edit	Impersonate	Delete
636eaafb9-81f5-4a69-8cb8-f6585574...	eval04@example.com	eval04@example.com			Edit	Impersonate	Delete
6e601bed-106d-4c26-af7f-459f2d3d...	eval05@example.com	eval05@example.com			Edit	Impersonate	Delete
14423744-1dbd-4072-a55b-429a209...	eval06@example.com	eval06@example.com			Edit	Impersonate	Delete
7776e67-6c8d-4ade-9a38-2b66382...	eval07@example.com	eval07@example.com			Edit	Impersonate	Delete
97b191c8-1860-4bd4-a3bb-2ea379...	eval08@example.com	eval08@example.com			Edit	Impersonate	Delete
b3270190-e03-472d-9f16-2cdf0fb...	eval09@example.com	eval09@example.com			Edit	Impersonate	Delete
8c7ab5b0-1771-4b2e-b15a-488176...	eval10@example.com	eval10@example.com			Edit	Impersonate	Delete
aba40b40-33b9-479b-b758-7a57f2...	eval11@example.com	eval11@example.com			Edit	Impersonate	Delete
c7719640-2e43-409c-bc70-23725d1...	eval12@example.com	eval12@example.com			Edit	Impersonate	Delete
e949e95b-40c3-470e-a304-342d18b...	eval13@example.com	eval13@example.com			Edit	Impersonate	Delete
fd54725b-546d-4505-bc51-36434f8...	eval14@example.com	eval14@example.com			Edit	Impersonate	Delete
ba420303ef049-4847-9eb6-1967c0cc...	eval15@example.com	eval15@example.com			Edit	Impersonate	Delete
58814183-0c84-4aff-92e8-e9b0a83...	eval16@example.com	eval16@example.com			Edit	Impersonate	Delete
9cae975b-6aff-4564-a44c-97a34117...	eval17@example.com	eval17@example.com			Edit	Impersonate	Delete
332dc59c-a342-4f71-9292-12cebf07...	eval18@example.com	eval18@example.com			Edit	Impersonate	Delete
37df70190-1bd3-4499-9f93-c1c8b249...	eval19@example.com	eval19@example.com			Edit	Impersonate	Delete

Here we can see all the end 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 the Clients section and they will be able to authenticate through them. Let's open one of these users' details.

The screenshot shows the 'Details' tab of a user profile. Key fields include:

- ID:** 238cd4a3-50cd-47b4-ae3d-987hb05e9836
- Created At:** 1/9/19 12:48:50 PM
- Username:** evals98@example.com
- Email:** evals98@example.com
- First Name:** [empty]
- Last Name:** [empty]
- User Enabled:** ON
- Email Verified:** ON
- Required User Actions:** Select an action... (dropdown)
- Impersonate user:** Impersonate button
- Buttons:** Save, Cancel

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

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

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

Clients > Add Client

### Add Client

The 'Add Client' form contains the following fields:

- Import:** Select file (button)
- Client ID \***: [empty]
- Client Protocol**: openid-connect
- Client Template**: [dropdown menu]
- Root URL**: [dropdown menu]
- Buttons:** Save, Cancel

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

Clients > sync-app

### Sync-app

- [Settings](#)
- [Roles](#)
- [Mappers](#)
- [Scope](#)
- [Revocation](#)
- [Sessions](#)
- [Offline Access](#)
- [Installation](#)

Client ID: sync-app

Name:

Description:

Enabled:  ON

Consent Required:  OFF

Client Protocol: openid-connect

Client Template:

Access Type: public

Standard Flow Enabled:  ON

Implicit Flow Enabled:  OFF

Direct Access Grants Enabled:  ON

Root URL:

\* Valid Redirect URIs: +

Base URL:

Admin URL:

Web Origins: +

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

Sync-app

- [Settings](#)
- [Roles](#)
- [Mappers](#)
- [Scope](#)
- [Revocation](#)
- [Sessions](#)
- [Offline Access](#)
- [Installation](#)

Client ID: sync-app

Name:

Description:

Enabled:  ON

Consent Required:  OFF

Client Protocol: openid-connect

Client Template:

Access Type: confidential

Standard Flow Enabled:  OFF

Implicit Flow Enabled:  OFF

Direct Access Grants Enabled:  OFF

Service Accounts Enabled:  ON

Root URL:

Base URL:

Admin URL:

> Fine Grain OpenID Connect Configuration

> OpenID Connect Compatibility Modes

Save  Cancel

#### Save -> Service account roles

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

Clients > sync-app

Sync-app

Service Account Roles

sync-app Service Accounts

Realm Roles

Available Roles

Assigned Roles

Effective Roles

Add selected >

< Remove selected

Client Roles

realm-management

Available Roles

Assigned Roles

Effective Roles

Add selected >

< Remove selected

And now we are ready to use the client credentials inside 3scale OIDC configuration section. To authenticate as we were an end user, we will need to create one test user, so let's go to the Users section and add a user

RED HAT SINGLE SIGN-ON

Admin

Threescale

Configure

- Realm Settings
- Clients
- Client Templates
- Roles
- Identity Providers
- User Federation
- Authentication

Manage

- Groups
- Users**
- Sessions
- Events
- Import
- Export

Users

Lookup

ID	Username	Email	Last Name	First Name	Actions
Bed23341-d96a-419b-a99d-a8fcddd...	threescale	threescale@openshiftworkshop.com			<a href="#">Edit</a> <a href="#">Impersonate</a> <a href="#">Delete</a>

We will fill in all the user details and switch to user email verified

RED HAT SINGLE SIGN-ON

Threescale

Configure

- Realm Settings
- Clients
- Client Templates
- Roles
- Identity Providers
- User Federation
- Authentication

Manage

- Groups
- Users**
- Sessions
- Events
- Import
- Export

Users > Add user

Add user

ID

Created At

Username \*

john

Email

john@example.com

First Name

john

Last Name

Doe

User Enabled

ON

Email Verified

ON

Required User Actions

Select an action...

Save Cancel

Now we will set the password, by going to credentials and setting it to password and reset

password.

The screenshot shows the Red Hat Single Sign-On (SSO) interface. In the top navigation bar, it says "RED HAT SINGLE SIGN-ON" and "Threescale". Below this, there's a breadcrumb trail: "Users > john". The main area is titled "John" with a trash icon. A navigation bar below the title includes "Details", "Attributes", "Credentials" (which is selected), "Role Mappings", "Groups", "Consents", and "Sessions". On the left, a sidebar titled "Configure" lists "Realm Settings", "Clients", "Client Templates", "Roles", "Identity Providers", "User Federation", and "Authentication". Under "Manage", there are links for "Groups", "Users" (which is selected), "Sessions", "Events", "Import", and "Export". The "Credentials" section contains two tabs: "Manage Password" and "Credential Reset". The "Manage Password" tab has fields for "New Password" and "Password Confirmation", both set to "\*\*\*\*\*". There's also a "Temporary" switch set to "OFF" and a "Reset Password" button. The "Credential Reset" tab has a "Reset Actions" dropdown set to "Select an action...", a "Expires In" field set to "12 Hours", and a "Reset Actions Email" button. A "Send email" button is also present.

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

## LAB BEGINS

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

The screenshot shows the 3scale API Management interface. The left sidebar includes "Overview", "Analytics", "Applications", "ActiveDocs", "Integration" (which is selected), "Configuration", "Methods & Metrics", and "Settings". The main area is titled "Configuration". It has three main sections: "Integration settings", "APIcast Configuration", and "Environments".  
**Integration settings:** Deployment Option is set to "APIcast" and Authentication is set to "API Key (user\_key)".  
**APIcast Configuration:** Private Base URL is "https://echo-api.3scale.net:443", Mapping rules is "/ => hits", Credential Location is "query", and Secret Token is "Shared\_secret\_sent\_from\_proxy\_to\_API\_backend\_1664772f8f98f392".  
**Environments:** It shows a "Staging Environment" with the URL "https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443 v. 4". There are buttons for "edit integration settings", "edit APIcast configuration", "Configuration history", and "Promote v. 4 to Production".

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

**AUTHENTICATION**

**Authentication**

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

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

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

OpenID Connect  
Use OpenID Connect for any OAuth 2.0 flow.

[Update Service](#)

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

**AUTHENTICATION**

**Authentication**

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

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

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

OpenID Connect  
Use OpenID Connect for any OAuth 2.0 flow.

[Update Service](#)

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

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

**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**  
[empty input field]

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\_8a412887b1f48430

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 to 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)

**ERRORS**

**AUTHENTICATION FAILED ERROR**

**Response Code**  
403

**Content-type**

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

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

`http://client-id:client-secret@<idp-public-endpoint>`  
where client-id: sync-app  
client secret: 5f8f3908-dc47-4b24-a9aa-44f6860e1144  
idp-public-endpoint:

[sso.apps.openbanking-fe8e.openshiftworkshop.com/auth/realms/threescale](https://sso.apps.openbanking-fe8e.openshiftworkshop.com/auth/realms/threescale)

Lastly, change the Credentials location to As HTTP Headers

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

The screenshot shows the Threescale APIcast Configuration interface. On the left, a sidebar menu is visible with options: ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area is divided into two sections: 'APIcast Configuration' and 'Environments'.

**APIcast Configuration** (top section):

- 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** (bottom section):

- Staging Environment**:  
URL: https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443  
Version: v. 5  
Actions: Configuration history, Promote v. 5 to Production
- Production Environment**:  
Message: no configuration has been saved for the production environment yet

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

The screenshot shows the 3scale API Management platform interface. At the top, there's a navigation bar with links for 'USER101', 'API CREDENTIALS', 'STATISTICS', and 'DOCUMENTATION'. Below this, a header bar has a back arrow labeled 'Applications' and a 'Edit Test' button. The main content area displays an application configuration for 'Test'. It includes fields for 'Name' (Test), 'Description' (Test), 'Plan' (Basic > Review/Change), 'Status' (Live), 'Client ID' (52df2946), and 'Client Secret' (with a 'Create secret' button). A 'Redirect URL' field is present with a placeholder 'This is your Redirect URL for OAuth.' and a 'Submit' button.

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).

This screenshot shows the 'Edit Test' screen for the 'Test' application. It features a 'Client Secret' section with a 'Regenerate' button and a 'Redirect URL' section containing the value 'https://openidconnect.net'. Both sections have a 'Submit' button at the bottom.

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

**Overview**

Application 'Test' | Analytics

## Test [Edit](#)

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

### API Credentials

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

### Usage in last 30 Days

**Application Plan: Basic**

[Convert to a Custom Plan](#)

**FEATURES**

- Unlimited Greetings [Edit](#) (Green)
- 24/7 support [Edit](#) (Red)
- Unlimited calls [Edit](#) (Red)

**Change Plan**

[Change Plan](#)

**SIGN-ON**

### Clients [?](#)

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

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

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

<https://openidconnect.net/>

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

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

<http://sso.apps.openbanking-fe8e.openshiftworkshop.com/auth/realms/threescale/.well-known/openid-configuration>

And click on USE DISCOVERY DOCUMENT

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

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

### OpenID Connect Configuration

([X](#))

Server Template	Custom
Discovery Document URL	<input type="text" value="https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm..."/> USE DISCOVERY DOCUMENT Use a discovery document to populate your server urls
Authorization Token Endpoint	<input type="text" value="https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm..."/>
Token Endpoint	<input type="text" value="https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm..."/>
Token Keys Endpoint	<input type="text" value="https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm..."/>
Remember to set <a href="https://openidconnect.net/callback">https://openidconnect.net/callback</a> as an allowed callback with your application!	
OIDC Client ID	<input type="text" value="5bc94f6a"/>
OIDC Client Secret	<input type="text" value="2b6db299110348dbae6134e97a8d0359"/>
Scope	<input type="text" value="openid email"/>
<a href="#">SAVE</a>	
Hey, just a friendly note: we store stuff like your keys in LocalStorage so that when you redirect to authenticate, you don't lose them. You can clear them by clicking on this button: <a href="#">CLEAR LOCALSTORAGE</a>	

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

2

## Exchange Code from Token

Your Code is

```
eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAdYXdZhKLt
QLwD-
FZLQP0aPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiCkjSe8hZWInWah
Bj4Rz0bm53WCukRf06m3LtzLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jLtF
```

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

Request
<pre>POST https://secure-sso-sso.apps.open- banking.opentrace.me/auth/realms/openshift/protocol/openid- connect/token grant_type=authorization_code &amp;client_id=5bc94f6a &amp;client_secret=2b6db299110348dbae6134e97a8d0359 &amp;redirect_url=https://openidconnect.net/callback &amp;code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAd QLwD- FZLQP0aPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiCkjSe8hZWInWah Bj4Rz0bm53WCukRf06m3LtzLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ- 3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jLtF</pre>
<b>EXCHANGE</b>

**Hit Exchange**

```
Request

POST https://secure-sso-sso.apps.open-
banking.opentry.me/auth/realms/openshift/protocol/openid-
connect/token
grant_type=authorization_code
&client_id=5bc94f6a
&client_secret=2b6db299110348dbae6134e97a8d0359
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.VeAd
QLwD-
FZLQP0aPD1mGwe7OKDJMI_32RqnrcpixM00YQePoPt5i6N1cKjSe8hZWInw
Bj4RzQbM53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrsFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vlKCPl0F1ZRCY4_my_oyplHCbeJa.jL

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

NEXT

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

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

### 3 Verify User Token

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

Your "id\_token" is

[VIEW ON JWT.IO](#)

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiwiia2lkIiA6ICJRa1RJX2Vws
2G0P5v5l7D-
CDJpLxoCvRr9gNKpBoBb9KFvvyyW8I6l6YX6B38rJAJ5wTCXZkyfUaSruIma
LWNpEPODhiYxNg7mmgjxKKYV8bMUW80yB7wSvCGotvF_XBx9K3g
```

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

VERIFY

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

The screenshot shows the jwt.io interface with a decoded JWT token. The token is split into three sections: HEADER, PAYLOAD, and SIGNATURE.

**HEADER:** ALGORITHM & TOKEN TYPE

```
{ "alg": "RS256", "typ": "JWT", "kid": "OKT1_epKb852EJJwe7urqPQkchDMF-R2xFpMmeBvh-U"}
```

**PAYOUT:** DATA

```
{ "jti": "7dcc7531-70ea-41ea-ad31-1502a8aca437", "exp": 1547396457, "nbf": 0, "iat": 1547396157, "iss": "https://secure-sso-sso.apps.open-banking.opentrue.me/auth/realm/openshift", "aud": "5bc94f6a", "sub": "236cd4a3-50c6-47b4-ae3d-987bb85e9836", "typ": "ID", "azp": "5bc94f6a", "auth_time": 1547395876, "session_state": "fa149bb8-d4e9-49be-95b0-9613ff0a55bd", "acr": "0", "preferred_username": "evals98@example.com", "email": "evals98@example.com" }
```

**SIGNATURE:**

```
RSASHA256(
base64URlEncode(header) + "." +
base64URlEncode(payload),
Public Key or Certificate. Ent
```

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

It should look something like this:

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiwiia2lkIiA6ICJRa1RJX2VwsIwNvpFSkp3ZTd1cnFQUWtjSERNRiSmhGcEtZUJ2aCIVn0.eyJqdGkiOilyYzJmZjQ5Zs01MDY4LTQ0MjQtYTRINS05MWU3OTk3MTM0YTMIcJleHaiOjE1NDczOTc1NTlsIm5iZl6MCwiaWF0IjoxNTQ3Mzk2NjUyLCJpc3MiOjodHRwczovL3NIY3Vys1zc28tc3NvLmFcHMub3Bib11YW5raW5nLm9wZWS05cnkubWuvYXV0aC9yZWFsbxMvb3B1bnN0aWZ0IiwiYXVkiJoiNWJjOTrmNmEiLCJzdW1oIiyMzZjZDRhMy01MGm2L7TQ3j0tyWUzZC05ODdiYJA1ZTk4MzY1LCJ0exXA10iJJCrcisImF6cC16IjViYzKjZjHiiwiYXV0aW90aW11ijoNTQ3Mzk1ODc2LCJzZXNzaW9uX3NBYXR1IjoiZmExND11OGItZDR10s000WJ1lTk1AtOTYxM2ZmMGE1NWJkIiwiYNYiJoiMCIsInByZWZlcnJ1Zf91c2VybmfZSI6ImV2YXwx0ThAZXhhbXsZS5jb201CJ1bWFpbC16ImV2YXwx0ThAZXhhbXsZS5jb201fQ.DxqOTu9ATiVAWGWMFFje-2G0P5v5l7D-CDJpLxoCvRr9gNkBoBb9KFvvyyW8I6l6YX6B38rJAJ5wTCXZkyfUaSruImaMh7fmPlfrFxNAGTg21cQhHTCpTx6vqCmhg5Po_Cy1Tv6dwEcrxekVvn16FKkoWOpEuVamA1y5Vkmlo1XR8C0TRKfdWNZF6AT9LnhZn5npHH28qRisYWTzyYGJuuQmxA272R7QKZTxXNPjZB0KPUachbDvhMcBn5YGPQLUPKKBdCQTN2z9uFngVjLQ1tka-
```

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. A test step is being configured with a 'Request' type (GET) and the URL https://wt2-evals99-example-com-3scale.apps.open-banking.opentry.me. An 'Authorization' header is added with the value Bearer eyJhbGciOiJSUzI1Ni... . The tool has tabs for Request, Response, and Variables.

## Let's hit Test

The screenshot shows the 'Response' tab of the API Tester tool. The status is HTTP/1.1 200 OK. The response body is a JSON array of bank objects:

```
[{"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" }}]
```

And success!

```
{
  "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"
      }
    }
  ]
}
```

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

- Check for the validity of the access token credentials (not expired, legit and associated to the correct application)

- Check for rate limits on the application triggering the call
- Apply monetization rules to the call
- Apply any additional policy that might modify the call in real time
- Report the traffic back to the analytics component

## Checkpoint

Improved security to the highest grade possible while using standards.

## OpenShift (optional)

### LAB BEGINS

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

The screenshot shows the OpenShift Container Platform Service Catalog interface. On the left, the 'Browse Catalog' section displays a grid of project icons categorized by language (.NET, PHP, Java, etc.) and type (e.g., .NET Core, MySQL, PostgreSQL). On the right, the 'My Projects' section shows a list of two projects: 'fuse-dcfbb062-1520-11e9-86c5-0a580a810008' and 'evals98-example-com-walkthrough-projects'. Both projects were created by 'evals98@example.com' 3 days ago.

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.

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

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

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

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

Are you sure you want to delete **syndesis-ui-1-c6gc9** in namespace **fuse-dcfbb062-1520-11e9-86c5-0a580a810008**?

**Cancel** **Confirm**

Are you sure you want to delete **syndesis-ui-1-c6gc9** in namespace **fuse-dcfbb062-1520-11e9-86c5-0a580a810008**?

**Cancel** **Confirm**

**✖** pods "syndesis-ui-1-c6gc9" is forbidden: User "evals98@example.com" cannot delete pods in the namespace "fuse-dcfbb062-1520-11e9-86c5-0a580a810008": no RBAC policy matched

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

## DEMO ONLY

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

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

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**

- 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

**NETWORKING**

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

Routes - External Traffic  
Create Route

1 pod

**OPENSHIFT CONTAINER PLATFORM Application Console**

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

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

Pods > syndesis-ui-1-c6gc9

syndesis-ui-1-c6gc9 created 3 days ago

app syndesis deployment syndesis-ui-1 deploymentconfig syndesis-ui More labels...

**Actions**

- Add Storage
- Edit YAML
- Delete

Status		Template	
Status:	Running	Containers	syndesis-ui
Deployment:	syndesis-ui, #1	Image:	fuse7/fuse-ignite-ui 0c05f43 100.0 MB
IP:	10.130.6.14	Ports:	8080/TCP
Node:	ip-172-31-3-6.ec2.internal (172.31.3.6)	Mount:	config-volume .. /usr/share/nginx/html/config read-write
Restart Policy:	Always	Mount:	default-token-qv2w .. /var/run/secrets/kubernetes.io/serviceaccount read-only
Container syndesis-ui		Memory:	50 MiB to 255 MiB
State:	Running since Jan 10, 2019 10:44:45 PM	Type:	Readiness Probe: GET / on port 8080 (HTTP) 1s delay, 1s timeout
Ready:	true	Config Map:	Liveness Probe: GET / on port 8080 (HTTP) 30s delay, 1s timeout
Restart Count:	0	Add Storage to syndesis-ui   Add Config Files to syndesis-ui	

**OPENSHIFT CONTAINER PLATFORM Application Console**

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

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

Pods > syndesis-ui-1-c6gc9

syndesis-ui-1-c6gc9 created 3 days ago

app syndesis deployment syndesis-ui-1 deploymentconfig

**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

**Actions**

**Cancel** **Delete**

Status		Template	
Status:	Running	Containers	syndesis-ui
Deployment:	syndesis-ui, #1	Image:	fuse7/fuse-ignite-ui 0c05f43 100.0 MB
IP:	10.130.6.14	Ports:	8080/TCP
Node:	ip-172-31-3-6.ec2.internal (172.31.3.6)	Mount:	config-volume .. /usr/share/nginx/html/config read-write
Restart Policy:	Always	Mount:	default-token-qv2w .. /var/run/secrets/kubernetes.io/serviceaccount read-only
Container syndesis-ui		Memory:	50 MiB to 255 MiB
State:	Running since Jan 10, 2019 10:44:45 PM	Type:	Readiness Probe: GET / on port 8080 (HTTP) 1s delay, 1s timeout
Ready:	true	Config Map:	Liveness Probe: GET / on port 8080 (HTTP) 30s delay, 1s timeout
Restart Count:	0	Add Storage to syndesis-ui   Add Config Files to syndesis-ui	

The screenshot shows the OpenShift Container Platform Application Console. In the center, a message box indicates that the 'Pod "syndesis-ui-1-c5gr9" was marked for deletion.' Below this, a list of deployment configurations is shown:

- 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

Under the 'CONTAINERS' section, it shows the 'syndesis-ui' container with the following details:

- Image: fuse7/fuse-ignite-ui 8c05f43 100.0 MB
- Ports: 8080/TCP

Resource usage statistics are provided for each pod:

Mib Memory	Cores CPU	Kib/s Network
340	< 0.01	0.1
12	< 0.01	1.5
16	< 0.01	2.8
50	< 0.01	1.9
570	< 0.01	89
1	1	1

A circular progress bar indicates '1 pod'.

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

This page isn't working

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

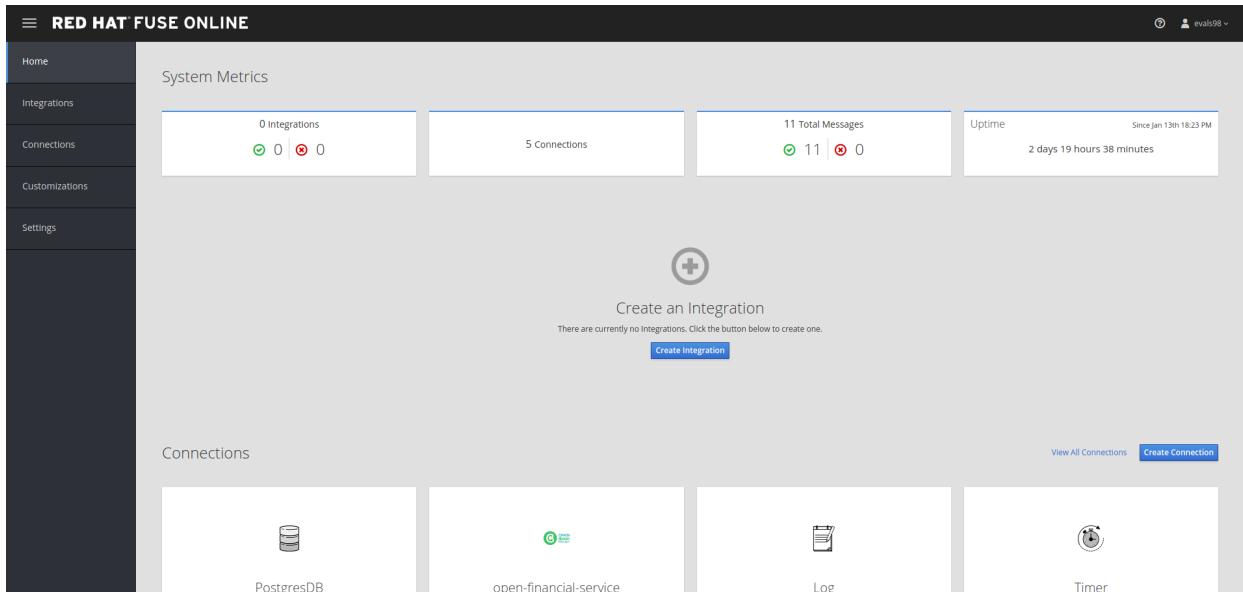
HTTP ERROR 502

[Reload](#)

The screenshot shows the OpenShift Container Platform Application Console. The 'syndesis-ui' pod is no longer listed in the deployment configuration list. The resource usage table now reflects the removal of the pod:

Mib Memory	Cores CPU	Kib/s Network
340	< 0.01	0.1
12	< 0.01	1.5
16	< 0.01	2.8
50	< 0.01	1.9
570	< 0.01	89
1	1	1

A circular progress bar indicates '1 pod'.



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

## Q&A

## Common issues

- openidconnect.net client might have an additional space in the redirect\_uri field. That's a client bug, you can fix it by adding an additional redirect URIs in RH SSO with a space preceding the URL: "<https://openidconnect.net/callback>"
- The installation of RH SSO might have some certificate issues, so might need to use instead a RH SSO deployed somewhere else or using the HTTP only route as suggested in the tutorial
- The database is deployed on ElephantSQL on a free tier, which allows only so many concurrent connections. You might receive a limit reached if too many users are trying the Open Data Bank API at the same time