

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
- 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.open-banking-2a38.openshiftworkshop.com>

End Users credentials:

user[1..100] / openshift

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

<https://userX-admin.apps.open-banking-2a38.openshiftworkshop.com>

access using username and password above

dedicated gateway

<https://userX.amp.apps.open-banking-2a38.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	CHECKPOINT
15 minutes	BREAK
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	CHECKPOINT
10 minutes	(optional according to timing) OpenShift high level walkthrough [simple demo]
5 minutes	Q&A and closing

Practical Part 1

Integr8ly

Login into the integr8ly environment main page

<https://tutorial-web-app-webapp.apps.open-banking-2a38.openshiftworkshop.com>

Welcome to the Red Hat Solution Explorer

Get started with an end-to-end solution walkthrough or use any of the available application services to create custom integrations.

Start with a walkthrough

- Integrating event-driven and API-driven applications (AMQ) preview Get Started 15 min
- Integrating event-driven and API-driven applications (EnMasse) Get Started 21 min
- Integrating API-driven applications preview Get Started 21 min

Applications 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 community
- EnMasse Ready for use
- Red Hat Fuse Ready for use
- 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 Online](#) link in the tutorial dashboard. When using Fuse Online for the first time, you will have to authorize access by clicking on "Allow selected permissions" and having both permissions checkboxes selected. We will see the main functionalities of it and then use it to build a simple integration block.

System Metrics

- 0 Integrations 0 | 0
- 4 Connections
- 0 Total Messages 0 | 0
- Uptime Since Jan 10th 22:44 PM 2 minutes

Create an Integration

There are currently no integrations. Click the button below to create one.

[Create Integration](#)

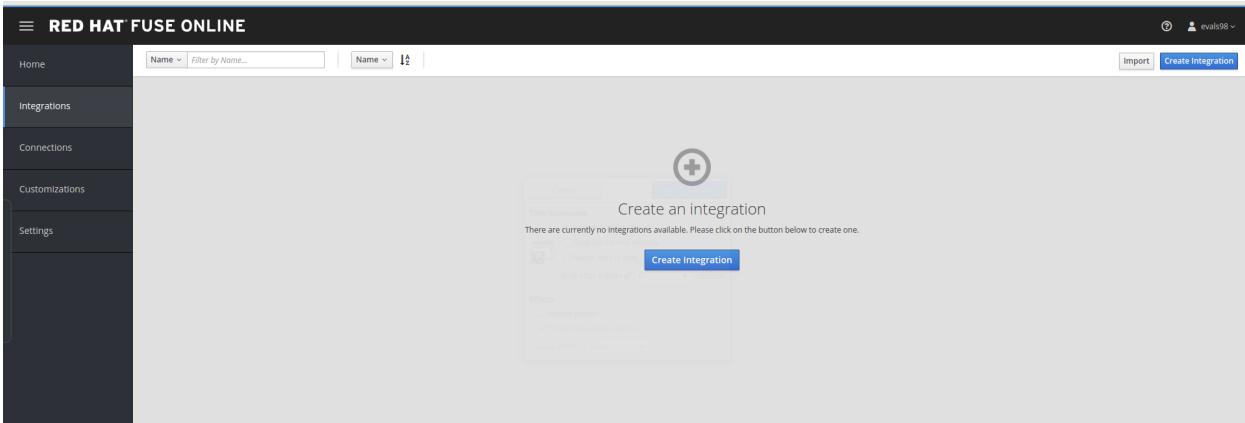
Connections

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

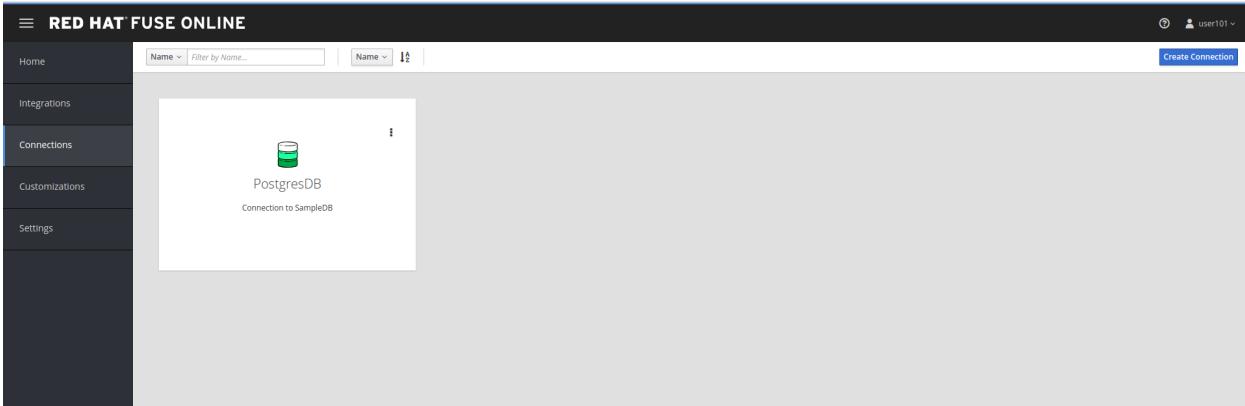
Icons for Datafeed, Log, Timer, and Webhook

The main dashboard ([Home](#)) is the landing page and it is especially important when you need to

check messages coming from the requests being sent and in general to give an overview of the deployed integration blocks.

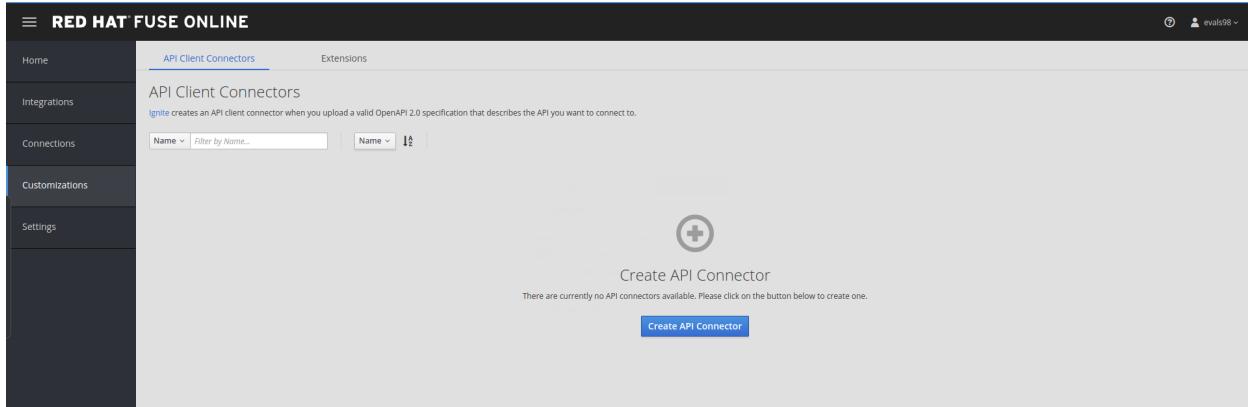


The integration window (**Integrations**) allows you to have a graphical representation of the mediation/transformation and is the first tool to onboard the citizen developers. Behind the curtains there is Camel (which is a proven technology and deployed in highly transactional environment as well) and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

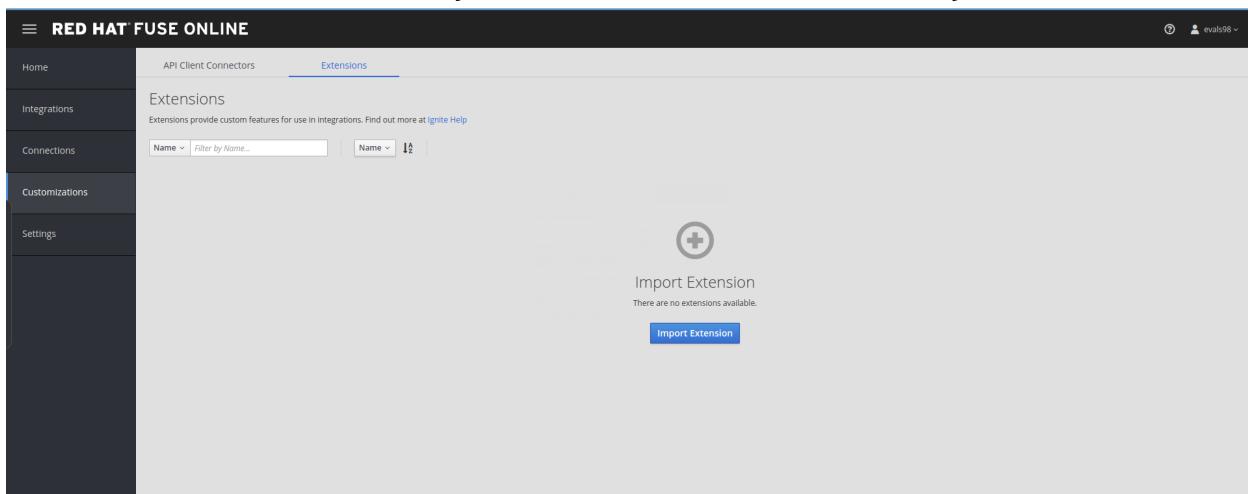


(**Connections**) you can start or end (or sometimes interpose) connections which are fully configured connectors. We will be showing the configuration of one connector during today's lab.

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



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



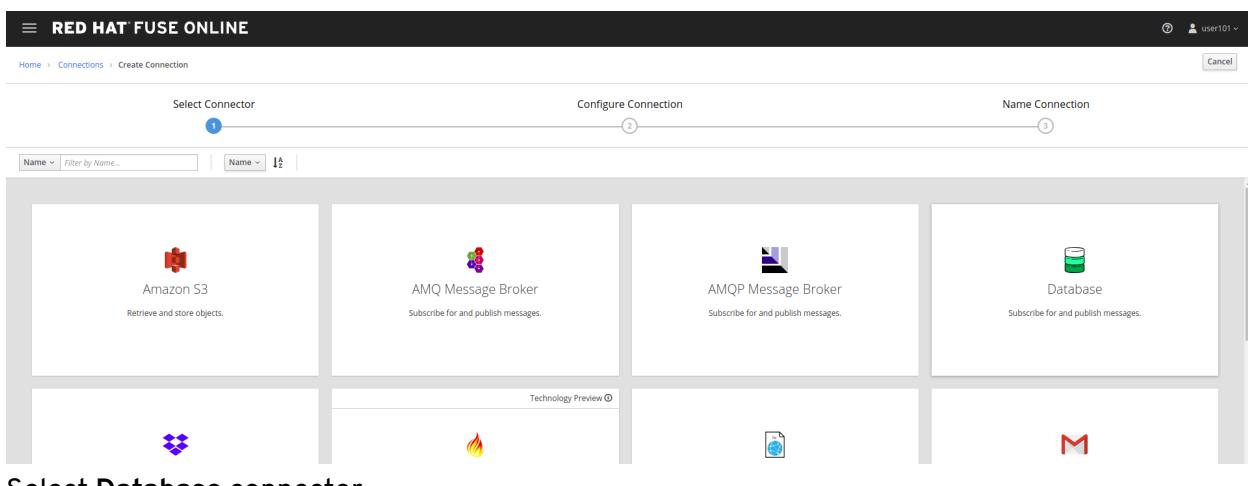
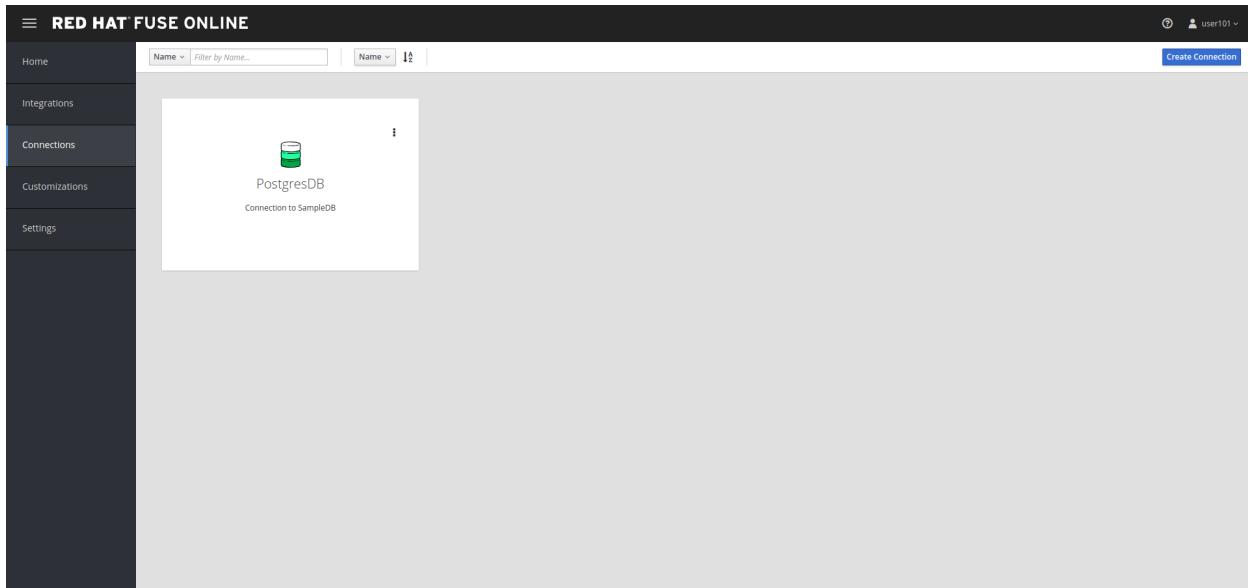
(**Extensions** panel) this is the part where you can extend the functionalities of the platform. You can compile a custom or community available extension to perform a specific functionality or connect to a specific system and in this easy way add transformation functionalities to the product.

LAB BEGINS

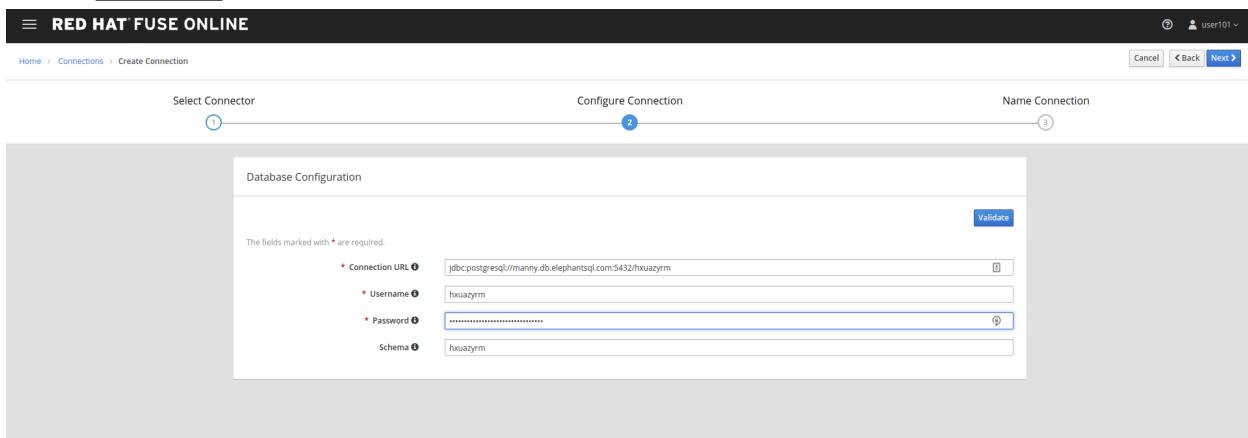
Let's start INTEGRATING now!

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

Connections -> Create connection



Select Database connector



Use the following connection details and validate them:

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

username: dbuser

password: password

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

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

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 connection creation process. It displays an 'Add Connection Details' form with the following fields:

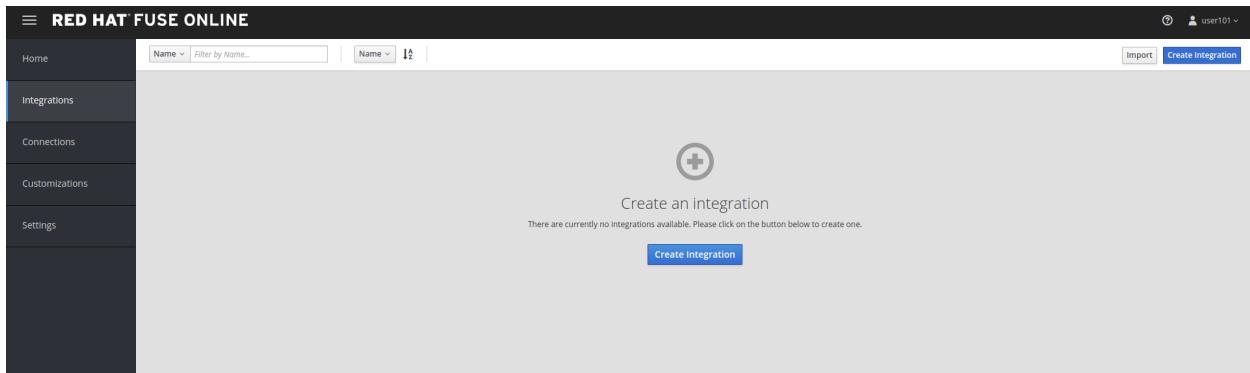
- Connection Name: open-data-banks
- Description: (empty)

The screenshot shows the main interface of Red Hat Fuse Online, specifically the 'Connections' section. The left sidebar lists 'Home', 'Integrations', 'Connections' (which is selected), 'Customizations', and 'Settings'. The main area displays a list of connections:

- open-data-banks
- PostgresDB
Connection to SampleDB

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

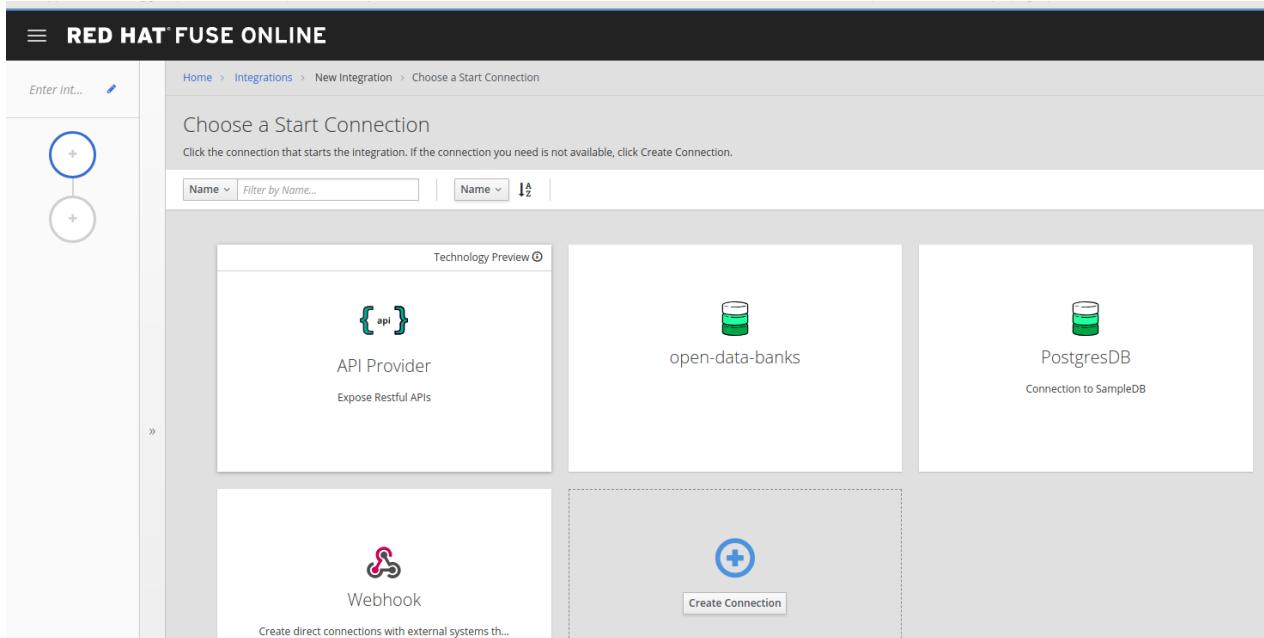
Now that we have configured the end of the integration path we want to build and we will see where to find the start and we will put the two pieces together. **Integrations -> create integration**



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 screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a 'Enter Int...' button and a diagram icon. The main area is titled 'Start integration with an API call'. It contains three options: 'Upload an OpenAPI file' (with a 'Choose File' button and a message saying 'Successfully uploaded "open-data-apis-nokey.json"'), 'Use a URL', and 'Create from scratch'. At the bottom are 'Next' and 'Cancel' buttons.

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

The screenshot shows the Apicurito interface for editing an API definition. The top navigation bar includes 'Back', 'Edit open-data-apis API Definition', and user info. The main content area is titled 'open-data-apis'. It features sections for 'Paths' (listing /banks, /banks/(bank_id), etc.), 'Definitions' (showing 'None Found Reusable types are useful'), and 'Tags' (listing ATM, Branch, Open Banking, Product, PSD2). To the right, detailed sections include 'INFO' (version 1.0.0, OpenBanking open data apis), 'CONTACT' (information about Luca Ferrari), 'LICENSE' (GNU GPL3, with a note about strong copyleft), 'Permissions' (Commercial Use, Distribution, Modification, Patent Use, Private Use), 'Conditions' (Disclose source, License and copyright notice, Same License, State changes), and 'Limitations' (Liability, Warranty). A 'View full GNU General Public License v3.0' link is also present.

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 has a breadcrumb 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'. 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') and buttons ('Save and continue', 'Back') remain the same.

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. The left sidebar shows 'db2rest' and 'API Provider'. The main area shows 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 'db2rest' and 'get list of ba...'. The main area has a heading 'Choose a Connection' with 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. At the bottom right, there's a 'Create Connection' button.

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's also a small note: 'Periodically invoke a stored procedure.'

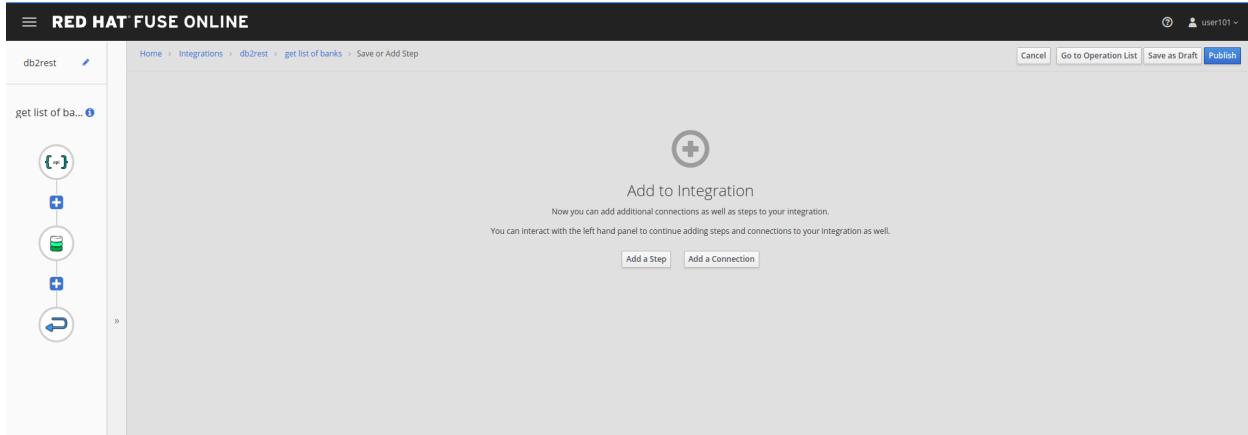
And we will invoke a simple SQL statement to return the data from a single table.

This screenshot shows the 'Configure Invoke SQL' screen. 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 of the input field are three buttons: 'Cancel', '< Choose Action', and 'Done' (which is highlighted in blue).

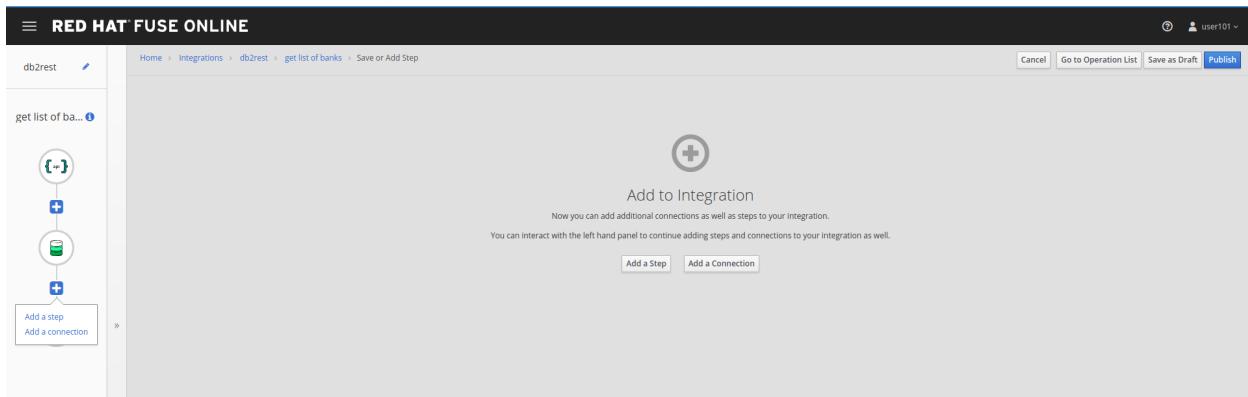
The simple statement to introduce is:

*select * from banks*

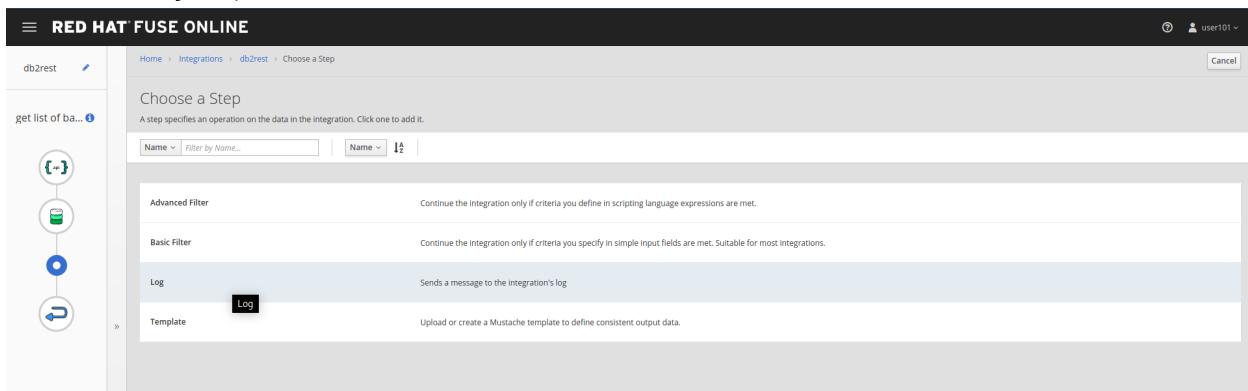
When you click done the statement will validate and you should be able to proceed with the configuration of the integration.



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



Select the log step.



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

The screenshot shows the 'Configure Log' dialog in the Red Hat Fuse Online interface. On the left, there's a sidebar with a tree view showing 'db2rest' and 'get list of banks'. The main area has a title 'Configure Log' and a sub-instruction 'Fill out the fields associated with the selected step.' Below this is a configuration panel with two checkboxes: 'Message Context' (unchecked) and 'Message Body' (checked). There's also a 'Custom Text' input field which is currently empty. At the bottom of the dialog are 'Cancel' and 'Done' buttons.

We are going to log just the message body.

The screenshot shows the integration builder interface in Red Hat Fuse Online. The sidebar lists 'db2rest' and 'get list of banks'. The main workspace features a large '+' icon with the text 'Add to Integration' and the sub-instruction 'Now you can add additional connections as well as steps to your integration.' Below the '+' icon are buttons for 'Add a Step' and 'Add a Connection'.

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

The screenshot shows the 'Integration Summary' page for the 'db2rest' integration. The top navigation bar includes 'Home', 'Integrations' (which is selected), 'Connections', 'Customizations', and 'Settings'. The main content area shows the integration name 'db2rest' and the status 'Assembling (1 / 4)'. Below this are tabs for 'Details', 'Activity', and 'Metrics'. In the 'Details' tab, it shows 'API Provider' and '1 Flow'. There's also a note 'no description set...'. At the bottom, there's a 'Draft' section with 'Draft' and 'Publish' buttons, and a 'History' section showing 'Version: 1'.

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

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

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

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

using this online tool <https://apitester.com/>. You can use the following URL to test it <http://i-db2rest.apps.open-banking-2a38.openshiftworkshop.com/open-data/banks>

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 main title is 'Build your test' with a 'View example' link. A note says 'Click ⚙ to add or remove steps'. The test configuration panel shows a single step: a 'Request' type set to 'no auth financial service' with a 'GET' method. The URL field contains 'https://open-data.b9ad.pro-us-east-1.openshiftapps.com/open-data/banks'. Underneath the URL, there are 'Headers' and '+ Add Request Header' options. To the right of the URL field is a 'Collapse / Expand' button. Below the configuration are three buttons: 'Test' (blue), 'Save to Account' (green), and 'Share Test Config' (grey). A tooltip for 'Save to Account' explains its benefits: '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'. At the bottom left is a copyright notice: '© 2019 RIGOR. ALL RIGHTS RESERVED'. At the bottom right is a 'Powered by' logo for RIGOR.

Populate the fields with the following URL

<http://i-db2rest.apps.open-banking-2a38.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: 12b28c7c44ae5989665ef8abec492fd0=08271c1b765ada49ab2c2c05b8b72917; path=/; HttpOnly Cache-control: private

Response Body  
 
 

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

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

     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 admin dashboard link(something like this <https://userX-admin.apps.openbanking-2a38.openshiftworkshop.com/>). You will be presented with the login window and can use you previously used user credentials.

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

Everybody 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 consider making 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** (Name field is empty)
- System name** (Field is empty, note: Only ASCII letters, numbers, dashes and underscores are allowed.)
- Description** (Field is empty)

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

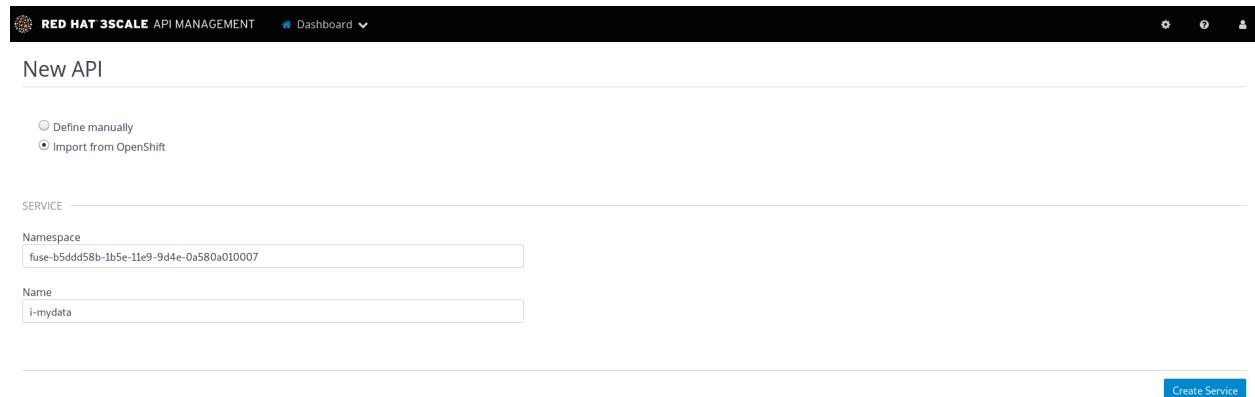
Full read/write access with all of your permissions
Includes any access you have to escalating resources like secrets

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

[Allow selected permissions](#)

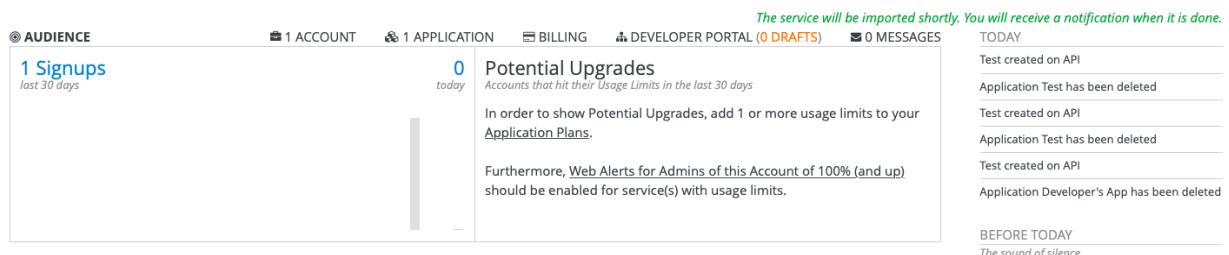
[Deny](#)

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 'SERVICE' section has fields for 'Namespace' (containing 'fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007') and 'Name' (containing 'i-mydata'). At the bottom right of the form 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)
ACCOUNT: 1 ACCOUNT
APPLICATION: 1 APPLICATION
BILLING: 0 today (Accounts that hit their Usage Limits in the last 30 days)
DEVELOPER PORTAL: 0 DRAFTS
MESSAGES: 0 MESSAGES

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

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

BEFORE TODAY:
The sound of silence

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

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

API-> Integration-> Configuration

The screenshot shows the 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 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 page. At the top, it says 'API MANAGEMENT' and 'API: i-db2rest'. Below that, it has sections for 'Private Base URL' (set to 'http://i-db2rest.fuse-06b94c72-2572-11e9-9425-0a580a010007.svc.cluster.local:8080') and 'Staging Public Base URL' (set to 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443'). There are also sections for 'Production Public Base URL' (set to 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443') and 'Mapping Rules'.

Integration

Configure the staging environment [documentation](#)

The screenshot shows the 'Integration' configuration page with more detailed settings. Under 'Private Base URL', there's a note about using https. Under 'Staging Public Base URL', it says 'Public address of your API gateway in the staging environment. Make sure to [add the correct route](#) [Use Default URL](#)'. Under 'Production Public Base URL', it says 'Public address of your API gateway in the production environment. Make sure to [add the correct route](#) [Use Default URL](#)'.

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.open-banking-2a38.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\$	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

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

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

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

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

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

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


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

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

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

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.

POLICIES

Policy Chain Add Policy

3scale APIcast
builtin - Main functionality of APIcast to work with the 3scale API m...

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:

In order to send a valid test request, please create an application that subscribes to an application plan of this service. Start with creating an application plan.

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

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

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

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

The screenshot shows the 'Create Application Plan' form. On the left is a dark sidebar with navigation links: Overview, Analytics, Applications (selected), Listing, Application Plans (selected), ActiveDocs, and Integration. The main area is titled 'Create Application Plan'. It contains fields for 'Name' (text input with 'Test Plan'), 'System name' (text input with 'testplan'), a note about ASCII letters, a checkbox for 'Applications require approval?' (unchecked), and fields for 'Trial Period (days)', 'Setup fee' (0.00 USD), and 'Cost per month' (0.00 USD). At the bottom is a blue 'Create Application Plan' button.

The screenshot shows the 'Application Plans' page. The sidebar is identical to the previous screenshot. The main area is titled 'Application Plans' and contains a table with one row. The table columns are 'Name', 'Applications', 'State', and actions. The row shows 'Basic' in 'Name', '0' in 'Applications', 'hidden' in 'State', and buttons for 'Publish', 'Copy', and 'Delete'. At the top right of the table area is a blue 'Create Application Plan' button.

Name	Applications	State	
Basic	0	hidden	Publish Copy Delete

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

Metrics, Methods, Limits & Pricing Rules

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

Features

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

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

Accounts

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

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

The screenshot shows the 'Account: Developer' page. On the left, a sidebar navigation includes 'Accounts' (selected), 'Listing', 'SETTINGS', 'Usage Rules', 'Fields Definitions', 'Applications' (selected), 'Billing', 'Developer Portal', and 'Messages'. The main content area displays account details: Organization/Group Name (Developer), Administrator (John Doe, john.doe@example.com), Signed up on (January 18, 2019 23:47), and Status (Approved). A 'Send message' button is present. To the right, there are two sections: 'Account Plan: Default' (with a 'Convert to a Custom Plan' link) and 'Application' (with fields for Name, Service, Plan, and State, and a 'Hits' counter showing 0 hits).

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

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

The screenshot shows the 'New Application' creation form. The sidebar navigation is identical to the previous screenshot. The main form has a header 'New Application' and includes fields for 'Application plan' (set to 'Basic'), 'Service plan' (set to 'Default'), 'Name' (empty), and 'Description' (empty). A 'Create Application' button is at the bottom right.

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 Apicast configuration interface. At the top, there's a 'POLICIES' section with a 'Policy Chain' and an 'Add Policy' button. Below it is a '3scale APIcast' section with a link to the documentation. The main area is labeled 'CLIENT' and contains a 'Test' button. A text input field shows the URL 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks'. Below the input field is a command-line example:

```
curl "https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks" -H'key: 13ece073549b898fb089a9493b4c1cae'
```

Below the client section, a note says: 'Hit the test button to check the connections between client, gateway & API.' To the right, there are 'Update & test in Staging Environment' and 'Back to Integration & Configuration' buttons.

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

The screenshot shows the Apicast configuration interface. At the top, there's a 'POLICIES' section with a 'Policy Chain' and an 'Add Policy' button. Below it is a '3scale APIcast' section with a link to the documentation. The main area is labeled 'CLIENT' and contains a 'Test' button. A text input field shows the URL 'https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks'. Below the input field is a command-line example:

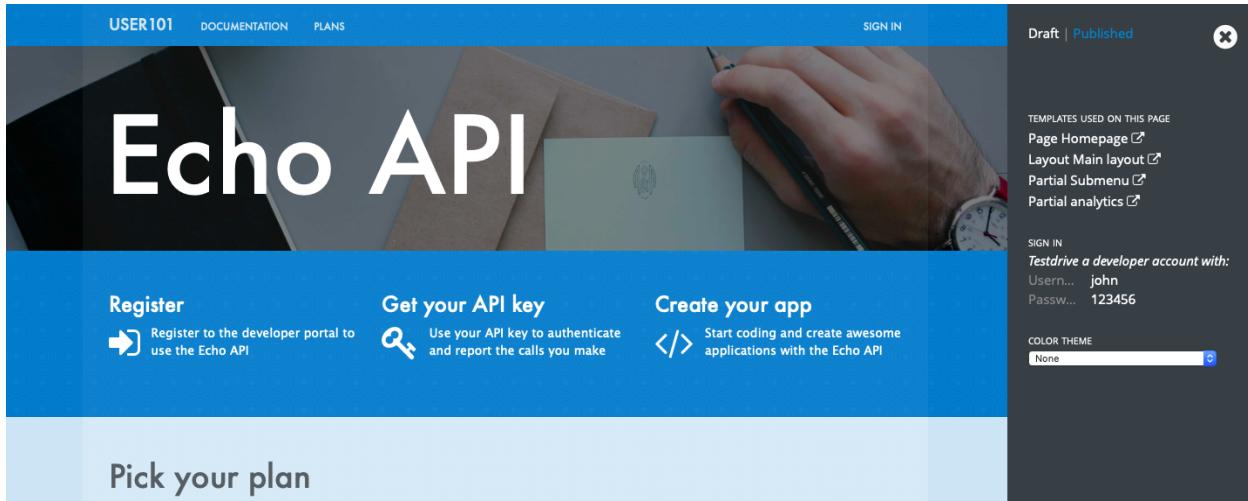
```
curl "https://user100.amp.apps.openbanking-0807.openshiftworkshop.com:443/open-data/banks" -H'key: 13ece073549b898fb089a9493b4c1cae'
```

Below the client section, a note says: 'Connection between client, gateway & API is working correctly as reflected in the analytics section.' To the right, there are 'Update & test in Staging Environment' and 'Back to Integration & Configuration' buttons.

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 Portal**

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



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

The screenshot shows the developer dashboard under the 'USER101' account. The top navigation includes 'API CREDENTIALS', 'STATISTICS', 'DOCUMENTATION', 'MESSAGES', 'SETTINGS', and a close button. A success message 'SIGNED IN SUCCESSFULLY' is visible. The main area features the 'Echo API' logo and a 'Your API Key' section. This section contains a warning: '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 is a modal window titled 'CREDENTIALS' showing an app named 'Test' with the key 'af44089cd553a3b0515772972b5f9ced'. It instructs to add this as a 'user_key' parameter to API calls.

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

USER101 API CREDENTIALS STATISTICS DOCUMENTATION MESSAGES SETTINGS

Applications

Name: Test
Description: Test
Plan: Basic > [Review/Change](#)

Status: Live

User Key: **af44089cd553a3b0515772972b5f9ced**
Add this as a `user_key` parameter to your API calls to report and authenticate.

[Regenerate](#)

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/>

API TESTER BETA

Sign In Create Account

Build your test

View example

Click to add or remove steps

Step Name:

Method:

URL:

Headers:

- + Add Request Header
-

[+ 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.open-banking-2a38.openshiftworkshop.com`, remember the the key Header and the associated value.

PASS N. Virginia
Seconds elapsed: 2

Results 576 ms Viewing a Request Step 1 < >

Message [Step 1] key financial service passed

Connection Information

Request

Request Headers

```
GET /open-data/banks HTTP/1.1
Host: wt2-evals98-example-com-3scale.apps.open-banking.opentry.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: 4c571db87a82b5aedccbd8877627a099
```

Response

Response Headers

```
HTTP/1.1 200 OK
Server: openresty/1.13.6.1
Date: Fri, 11 Jan 2019 18:04:18 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5078
Set-Cookie: JSESSIONID=y09rav2w644k1uns81kk5fiuy;Path=/
Expires: Fri, 11 Jan 2019 18:04:18 GMT
Access-Control-Allow-Origin: *
```

As we can see we succeed with 200 OK!

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

Connection Information

Request

Request Headers

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

Response

Response Headers

```
HTTP/1.1 403 Forbidden
Server: openresty/1.13.6.1
Date: Fri, 11 Jan 2019 18:16:42 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: chunked
Set-Cookie: 6ec552533d9895c2bc361d784adc8db=fd85f9e6e21ee8a00f6cad673c7c9e5b; path=/; HttpOnly; Secure
```

Response Body

As expected we receive a Forbidden error.

Checkpoint



Break

Practical Part 2

RH SSO and 3SCALE OIDC

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

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

LAB BEGINS

Let's start with RH SSO main dashboard

<https://sso-sso.apps.open-banking-2a38.openshiftworkshop.com/auth/admin/userX/console/#/realms/userX>

The screenshot shows the RHSSO interface. On the left, there is a sidebar with a dark background containing navigation links: 'Configure' (selected), 'Realm Settings' (selected), 'Clients', 'Client Templates', 'Roles', 'Identity Providers', 'User Federation', 'Authentication', 'Manage' (selected), 'Groups', 'Users', 'Sessions', 'Events', 'Import', and 'Export'. The main content area has a light background and is titled 'Openshift'. It contains several input fields and controls:

- 'Name': A text input field containing 'openshift'.
- 'Display name': An empty text input field.
- 'HTML Display name': An empty text input field.
- 'Enabled': A toggle switch set to 'ON'.
- 'Endpoints': A dropdown menu currently set to 'OpenID Endpoint Configuration'.
- 'Save' and 'Cancel' buttons at the bottom right of the form.

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/realm/openshift",
  "authorization_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/auth",
  "token_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/token",
  "token_introspection_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/token/introspect",
  "userinfo_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/userinfo",
  "end_session_endpoint": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/logout",
  "jwks_url": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/certs",
  "check_session_iframe": "https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/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/realm/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/realm/openshift/account	Edit	Export	Delete
admin-cli	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		
c50f7d72-11ad-40c6-b318-d7112e...	user101	user101@openshiftworkshop.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. This is the end user of the applications created in the Clients section and he will be able to authenticate through them. It is also the same user that each one is using to authenticate to the PaaS we are sharing. Let's open the users' details.

The screenshot shows the 'Details' tab of a user profile. The user is 'evals98@example.com'. Key fields include:

- ID:** 236ccda3-50cb-47b4-ae3d-987bb05e9836
- 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...
- Impersonate user:** Impersonate

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

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

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

The screenshot shows the 'Add Client' form. Fields include:

- Import:** Select file
- Client ID ***: (Required)
- Client Protocol:** openid-connect
- Client Template:** (dropdown menu)
- Root URL:** (empty)

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

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

Save -> Service account roles

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

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

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

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

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

AUTHENTICATION

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

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

App_ID and App_Key Pair
The application is identified via the App_ID and authenticated via the App_Key.

OpenID Connect
Use OpenID Connect for any OAuth 2.0 flow.

[Update Service](#)

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

AUTHENTICATION

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

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

App_ID and App_Key Pair
The application is identified via the App_ID and authenticated via the App_Key.

OpenID Connect
Use OpenID Connect for any OAuth 2.0 flow.

[Update Service](#)

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

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

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_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
application/json

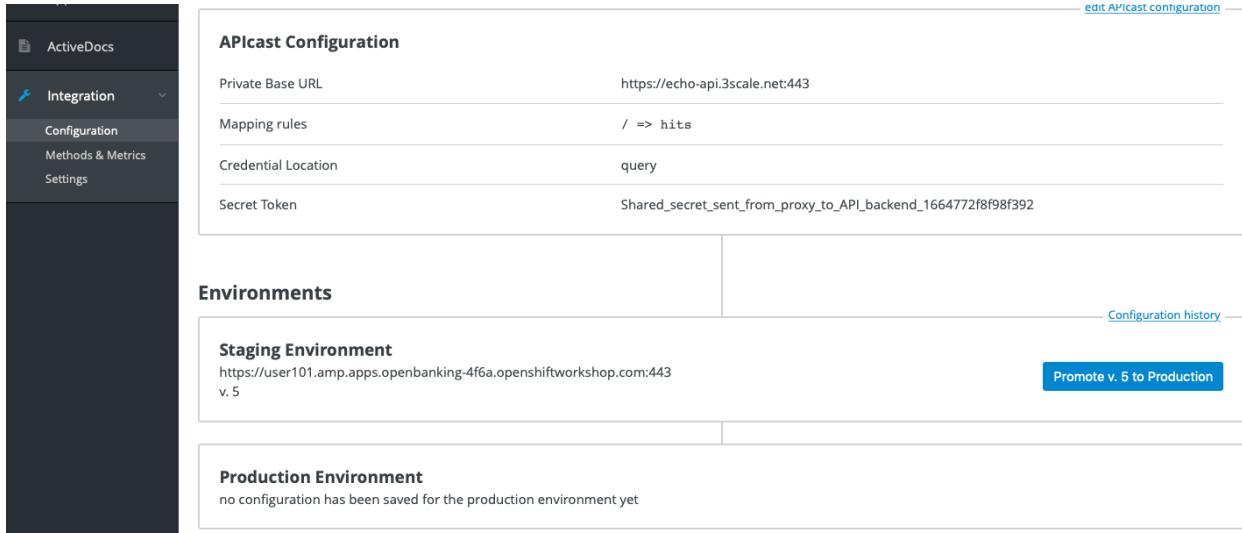
As we see we have a dedicated field for this purpose now: **OpenID Connect Issuer**

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

`https://client-id:client-secret@<idp-public-endpoint>`
where client-id: sync-app
client secret: <defined-during-configuration>
idp-public-endpoint:
`sso-sso.apps.open-banking-2a38.openshiftworkshop.com/auth/realms/<userX>`

Lastly, change the Credentials location to As HTTP Headers

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



The screenshot shows the APIcast Configuration and Environments sections of the OpenShift interface.

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:**
 - URL: `https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443`
 - Version: v. 5
 - Action: [Promote v. 5 to Production](#)
- Production Environment:**
 - 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 section titled 'Applications' is shown with a back arrow and an 'Edit Test' button. The application details are listed:

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

[Submit](#)

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

The screenshot shows the application configuration page again. The 'Client Secret' field now contains a regenerated value: '52df2946'. The 'Redirect URL' field has been updated to 'https://openidconnect.net'. The 'Submit' button is visible 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 Suspend

API Credentials

Client ID	52df2946
Client Secret	ea2494e0efbfddde7b4ae2d77062a60f3 Regenerate
Redirect URL	https://openidconnect.net/callback Edit

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	Edit Export
5bc94f6a	True	Not defined	Edit Export
account	True	/auth/realm/openShift/account	Edit Export
admin-cl	True	Not defined	Edit Export
broker	True	Not defined	Edit Export
launcher-openShift-users	True	Not defined	Edit Export
openShift-client	True	Not defined	Edit Export
realm-management	True	Not defined	Edit Export
security-admin-console	True	/auth/admin/openShift/console/index.html	Edit Export
sync-app	True	Not defined	Edit Export

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

`https://sso-sso.apps.open-banking-2a38.openshiftworkshop.com/auth/realms/<userX>/.well-known/openid-configuration`

And click on USE DISCOVERY DOCUMENT

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

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

OpenID Connect Configuration

Server Template: Custom

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

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

Token Endpoint: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/token>

Token Keys Endpoint: <https://secure-sso-sso.apps.open-banking.opentry.me/auth/realm/openshift/protocol/openid-connect/certs>

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

OIDC Client ID: 5bc94f6a

OIDC Client Secret: 2b6db299110348dbae6134e97a8d0359

Scope: openid email

SAVE

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

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

2

Exchange Code from Token

Your Code is

```
eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VeAdYXdZhKLt
QLwD-
FZLQP0aPD1mGwe70KDJKM_32RqnRPCixM00YQePoPt5i6NiKjSe8hZWInWah
Bj4RzQbm53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jL
```

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

Request

```
POST https://secure-sso-sso.apps.open-
banking.opentrace.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-
FZLQP0aPD1mGwe70KDJKM_32RqnRPCixM00YQePoPt5i6NiKjSe8hZWInWah
Bj4RzQbm53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vLKCPi0F1ZRCY4_my_oyplHCbeJa.jL
```

EXCHANGE

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-
FZLQP0aPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NICKjSe8hZWInW
Bj4RzQbM53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrSFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vlKCPi0F1ZRCY4_my_oyplHCbeJa.jL

HTTP/1.1 200
Content-Type: application/json
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk
  "not-before-policy": 0,
  "session_state": "fa149b8b-d4e9-49be-95b0-9613ff0a55bd",
  "scope": ""
}
```

NEXT

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

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

3 Verify User Token

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

Your "id_token" is

[VIEW ON JWT.IO](#)

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRa1RJX2Vs  
2G0P5v5l7D-  
CDJpLxoCbVrr9gNKpBoBb9KFwviyW8I6l6YX6B38rJAJ5WTCXZkyfUaSruIma  
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.opentrace.me/auth/realm/openid-connect/certs>, which is found in the discovery document or configuration menu options.

[VERIFY](#)

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

The screenshot shows the jwt.io interface with the following details:

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "RS256",  
  "typ": "JWT",  
  "kid": "OKT1i_epKb852EJwe7urqPQkchDMF-R2xFpMmeBvh-U"
```

PAYOUT: DATA

```
{  
  "jti": "7dcc7531-70ea-41ea-ad31-1502a8aca437",  
  "exp": 1547396457,  
  "nbf": 0,  
  "iat": 1547396157,  
  "iss": "https://secure-sso-sso.apps.open-banking.opentrace.me/auth/realm/openshift",  
  "aud": "Sbc94f6a",  
  "sub": "236cd4a3-50c6-47b4-ae3d-987bb05e9836",  
  "typ": "ID",  
  "azp": "Sbc94f6a",  
  "auth_time": 1547395876,  
  "session_state": "fa1498bb-d4e9-49be-95b0-9613ff0a55bd",  
  "acr": "0",  
  "preferred_username": "evals98@example.com",  
  "email": "evals98@example.com"  
}
```

VERIFY SIGNATURE

```
RSASHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  Public Key or Certificate. Ent
```

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

Request

```

POST https://secure-sso-sso.apps.open-
banking.opentry.me/auth/realms/openshift/protocol/openid-
connect/token
grant_type=authorization_code
&client_id=5bc94f6a
&client_secret=2b6db299110348dbe6134e97a8d0359
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXilCJlbmMioiJBMTI400JDLUhTMju2In0..VeAd
QLwD-
FZLQP0aPD1mGwe70KDJM_32RqnrcpixM00YQePoPt5i6NIcKjSe8hZWinw
Bj4Rz0bM53WcukRf06n3LtZL0Z0T-X14eJGobGPrSFP37PuFtkz-
3m7oubNDyF_Zk5msqY7ir7x8v0K3vIKCp0F1ZRCY4_my_oyplHCbeJa.jl

HTTP/1.1 200
Content-Type: application/json
{
  "access_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiwi",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiwi",
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiwiia2lk",
  "not-before-policy": 0,
  "session_state": "fa149b8b-d4e9-49be-95b0-9613ff0a55bd",
  "scope": ""
}

```

NEXT

It should look something like this:

```

eyJhbGciOiJSUzIINiIsInR5cCIgOiAiSldUiwiia2IkliA6ICJRa1RX2VwS2lwNVpFSkp3ZTd1cnFQUWtjSERNRIiSMNhGcE1tZU2aC1Vln0.eyJqdGkiOiLyZJmZjQ5ZS01MDY4LTQ0
MjQtYTRINS05MWU3OTk3MTM0YTMIcJheHAIoIE1ndczOtc1NTlsIm5iZl6MCwiaWF0joxtQ3Mzk2NjUyLCJpc3MIOJodHRwcovL3NIY3VyzS1zc28tc3NvLmFwcHMub3Bl
biIiYW5raW5nLm9wZW50cnkubWuYXV0aC9zWFsbXMb3B1bnNaoWZ0liwiYXVkljoiNWJ0tRmNmElCJzdWliOilyMzzjZDRhMy01MGMLTQ3YjQtYWUzZC05ODdiYjA1ZT
k4MzViLCJ0eXAiOjCZWfyzXliLCJhenAiOiiYmM5NGY2YSIsImFlGhfdGtzS16MTU0Nz5NTg3Niwick2Vzc2Vb19zdGF0ZS16lmZhMTQ5YjhiLWQ0ZTktnDliZS05NwlwLtk2
MTNmZjBhNTViZCIsImfcjliljAiLCJhbGxvd2VklW9yaWdpbnMiOltfLCJyZWFsbV9h2Nic3M0nsicm9sZXMiOlsidWlhX2F1dGhvcml6YXRpb24iX0slJlc291cmNlX2FjY2vzcy
l6eyJhY2NvdW50lpj7InJvbGVzljpb1mhbmnZS1hY2NvdW50liwbWFuYWDILWFjY291bnQtbGlua3MiLCJ2aWV3LXByb2ZpbGUixX19LCJwcmVmZXJyZWRfdXNlcm5hbWUiOj
Idmfsczk4QGV4YWlwbgUuY29tliwiZWhaWwiOjldmfsczk4QGV4YWiwbGuuY29tn0.07y6GDFq5CajATODkywEuQqEuD5H7_YMqrVC4AMPthZ-
m_xz_DAPBEqj3mmzp1oJ0oo_4pMxNgKpyyqCQifY79GRS5lJE6aVrZK53rQkud5diaZAЕ1-ryiD8Ctp_MrQtsTS7bVKbaFyCXNyFxy3c-
TER8GnGG900IYPxpy5M954sicp4CWxhA7ZwVeUqNRs5w2G2TCjrFyQjCzsINfwDRtADjbMiY7kq1cwRB5qm9ipdEEligDnH8dietiOzgY24sK10vtowjz_CHuWr5W3474dAZVF
C7utwStl_bNcojigENRcz5cP7fH7Nim8e4itWoSVPRVYcfDHyYb9zixQ

```

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>

Let's hit Test

Request

Request Headers

```
GET /open-banking/banks HTTP/1.1
Host: wt2-eval199-example-com-3scale.apps.open-banking.opentrace.me
Accept: */*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCig0IAisIdUiwia2lkIlA6ICJRa1RJX2VwS2lWvpFSkp3ZTd1cnFQuWtjSERNRI15MnH6cE1tZUJ2aC1ViN0eyJqdG
```

Response

Response Headers

```
HTTP/1.1 200 OK
Server: openbankproject/1.13.6.1
Date: Sun, 13 Jan 2019 16:26:57 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5970
Set-Cookie: JSESSIONID=9qbuuy865vuua959siq86bh3ny; Path=/
Expires: Sun, 13 Jan 2019 16:26:57 GMT
Access-Control-Allow-Origin: *
Cache-Control: no-cache, private, no-store
Correlation-Id: 9qbuuy865vuua959siq86bh3ny
Pragma: no-cache
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Firebase-Project: SAMEPROJECT
X-Xss-Protection: mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src 'self' https://static.openbankproject.org
Referrer-Policy: no-referrer-when-downgrade
Set-Cookie: 4e0f3df6272873fd1ac68b3782307-029029bcfa9a2bdf608b3d955e541; path=/; HttpOnly; Secure
Set-Cookie: 0cee625c053b039ca8b739deed9521874+fdb8579e62ee8a080f6cad673c79e5b; path=/; HttpOnly; Secure
```

Response Body

Variables

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.

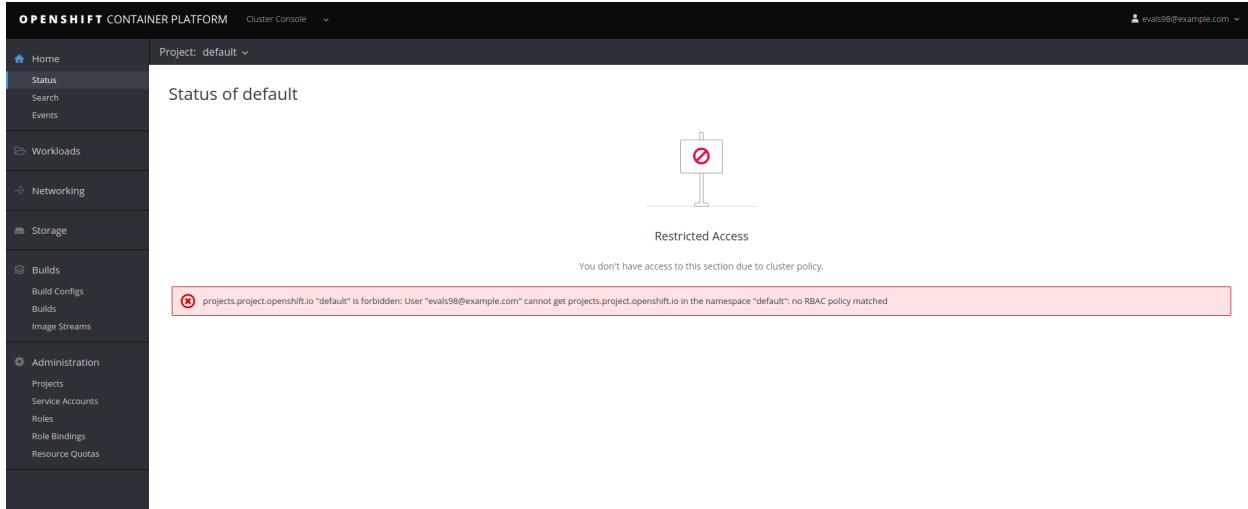
The screenshot shows the OpenShift Container Platform Service Catalog. At the top, there's a search bar labeled "Search Catalog". Below it, a navigation bar includes "Service Catalog", "Deploy Image", "Import YAML / JSON", and "Select from Project". On the left, a sidebar lists categories: All, Languages, Databases, Middleware, CI/CD, and Other. A "Filter" dropdown shows "139 Items". The main area displays a grid of service icons categorized by language (.NET, PHP, Java, etc.) and type (e.g., .NET Core, Apache HTTP Server, Django). Each item includes a small icon, the name, and a brief description. To the right, a sidebar titled "My Projects" shows two projects: "evals98@example.com-walkthrough-projects" and "fuse-dcfbb062-1520-11e9-86c5-0a580a810008".

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

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

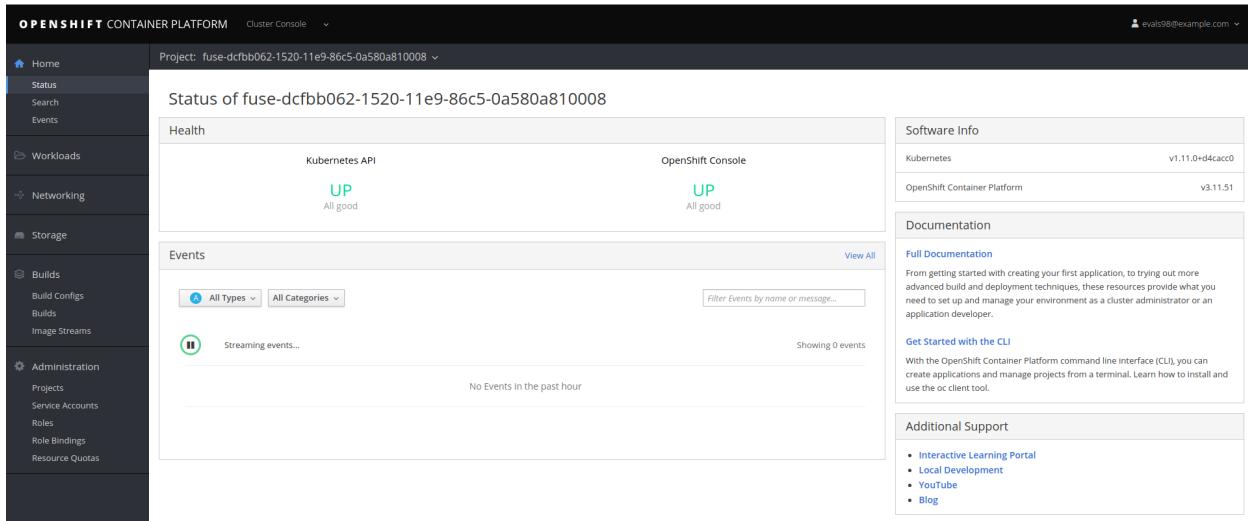
The screenshot shows the OpenShift Container Platform Cluster Console. The left sidebar includes "Home", "Workloads", "Networking", "Storage", "Builds", "Administration" (selected), "Projects" (selected), "Service Accounts", "Roles", "Role Bindings", and "Resource Quotas". The main area is titled "Projects" and shows a table of existing projects. The table columns are NAME, STATUS, REQUESTER, and LABELS. Two projects are listed: "evals98@example.com-walkthrough-projects" (Active, requester: evals98@example.com, no labels) and "fuse-dcfbb062-1520-11e9-86c5-0a580a810008" (Active, requester: No requester, no labels). A "Create Project" button is located at the top left of the table area.

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



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

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



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

OPENSHIFT CONTAINER PLATFORM Cluster Console

Workloads

- Pods
- Deployments
- Deployment Configs
- Stateful Sets
- Secrets
- Config Maps
- Cron Jobs
- Jobs
- Daemon Sets
- Replica Sets
- Replication Controllers
- HPAs

Networking

Storage

Builds

Administration

- Projects
- Service Accounts
- Roles

Pod Name	Namespace	Annotations	Status	Ready
syndesis-operator-1-b5tlr	use-dcfbb062-1520-11e9-86c5-0a580a810008	syndesis.io/type=operator	Running	Ready
syndesis-prometheus-1-6dczf	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-prometheus-1 deploymentconfig=syndesis-prometheus syndesis.io/app=syndesis syndesis.io/component=syndesis-prometheus syndesis.io/type=infrastructure	Running	Ready
syndesis-server-1-jjhrn	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-server-1 deploymentconfig=syndesis-server syndesis.io/app=syndesis syndesis.io/component=syndesis-server syndesis.io/type=infrastructure	Running	Ready
syndesis-ui-1-c6gc9	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-ui deploymentconfig=syndesis-ui syndesis.io/app=syndesis syndesis.io/component=syndesis-ui syndesis.io/type=infrastructure	Running	Ready
todo-1-8bd4k	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=todo-1	Running	Ready

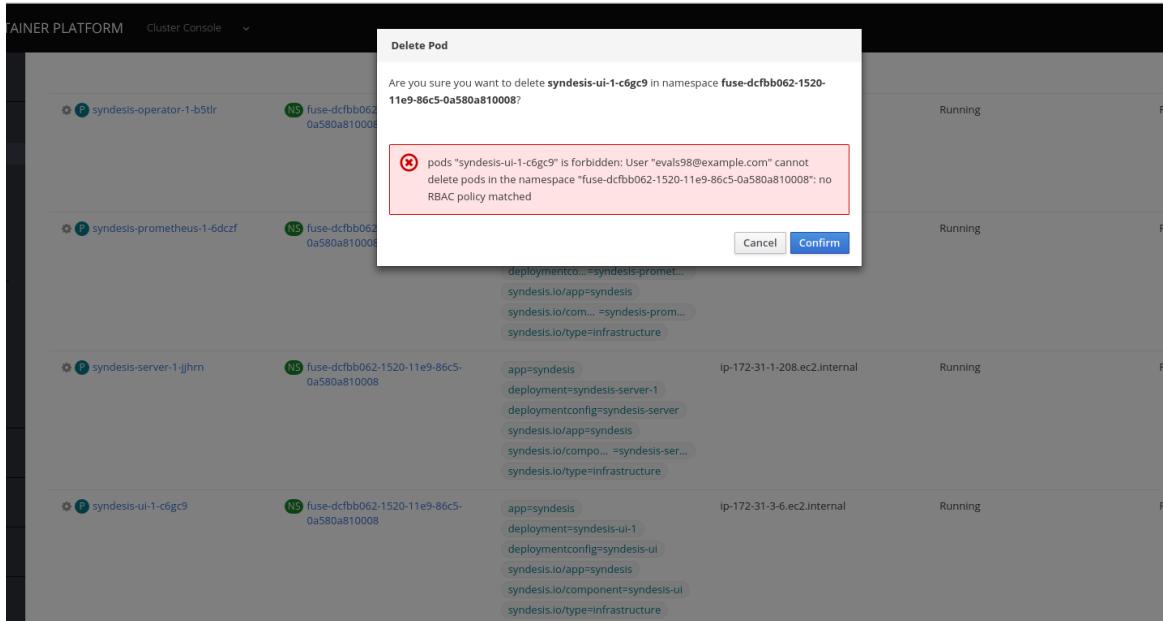
CONTAINER PLATFORM Cluster Console

Delete Pod

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

Cancel **Confirm**

Pod Name	Namespace	Annotations	Status	Ready
syndesis-operator-1-b5tlr	use-dcfbb062-1520-11e9-86c5-0a580a810008	syndesis.io/type=operator	Running	Ready
syndesis-prometheus-1-6dczf	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-prometheus-1 deploymentconfig=syndesis-prometheus syndesis.io/app=syndesis syndesis.io/component=syndesis-prometheus syndesis.io/type=infrastructure	Running	Ready
syndesis-server-1-jjhrn	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-server-1 deploymentconfig=syndesis-server syndesis.io/app=syndesis syndesis.io/component=syndesis-server syndesis.io/type=infrastructure	Running	Ready
syndesis-ui-1-c6gc9	use-dcfbb062-1520-11e9-86c5-0a580a810008	app=syndesis deployment=syndesis-ui deploymentconfig=syndesis-ui syndesis.io/app=syndesis syndesis.io/component=syndesis-ui syndesis.io/type=infrastructure	Running	Ready



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.

APPLICATION
syndesis

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

	Mib Memory	Cores CPU	Kib/s Network	Pods
DEPLOYMENT CONFIG broker-amq, #1	100	< 0.01	52	0
DEPLOYMENT CONFIG syndesis-db, #1	340	< 0.01	0.1	1
DEPLOYMENT CONFIG syndesis-meta, #1	11	< 0.01	1.5	1
DEPLOYMENT CONFIG syndesis-oauthproxy, #1	16	< 0.01	1.8	1
DEPLOYMENT CONFIG syndesis-operator, #1	50	< 0.01	1.9	1
DEPLOYMENT CONFIG syndesis-prometheus, #1	570	0.01	41	1
DEPLOYMENT CONFIG syndesis-server, #1	12	< 0.01	7.9	1
DEPLOYMENT CONFIG syndesis-ui, #1	16	< 0.01	0.9	1

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.

APPLICATION
syndesis

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

Mib Memory	Cores CPU	Kib/s Network
5.7	0	0.3
0	0	0

NETWORKING

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

Routes - External Traffic
Create Route

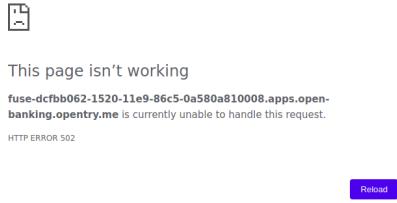
The image consists of three vertically stacked screenshots of the OpenShift Container Platform Application Console.

Screenshot 1: Shows the details of a pod named "syndesis-ui-1-c6gc9". The "Actions" dropdown menu is open, showing options like "Add Storage", "Edit YAML", and "Delete".

Screenshot 2: A "Confirm Delete" dialog box is displayed, asking if you want to delete the pod. It states: "Are you sure you want to delete the pod 'syndesis-ui-1-c6gc9'? It cannot be undone. Make sure this is something you really want to do!". There is a checkbox for "Delete pod immediately without waiting for the processes to terminate gracefully".

Screenshot 3: The main application view shows the deployment configuration "syndesis-ui, #1" has been marked for deletion. The status bar at the top right says "Pod 'syndesis-ui-1-c6gc9' was marked for deletion." Below, it lists other deployment configurations: "syndesis-meta, #1", "syndesis-oauthproxy, #1", "syndesis-operator, #1", "syndesis-prometheus, #1", and "syndesis-server, #1".

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



OPENSHIFT CONTAINER PLATFORM Application Console

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

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

DEPLOYMENT CONFIG
syndesis-meta, #1

340	< 0.01	0.1
Mib Memory	Cores CPU	Kib/s Network

DEPLOYMENT CONFIG
syndesis-oauthproxy, #1

12	< 0.01	1.5
Mib Memory	Cores CPU	Kib/s Network

DEPLOYMENT CONFIG
syndesis-operator, #1

16	< 0.01	2.8
Mib Memory	Cores CPU	Kib/s Network

DEPLOYMENT CONFIG
syndesis-prometheus, #1

50	< 0.01	1.9
Mib Memory	Cores CPU	Kib/s Network

DEPLOYMENT CONFIG
syndesis-server, #1

570	< 0.01	89
Mib Memory	Cores CPU	Kib/s Network

DEPLOYMENT CONFIG
syndesis-ui, #1

1	Average Usage Last 15 Minutes	
Mib Memory	Cores CPU	Kib/s Network

CONTAINERS
syndesis-ui

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

NETWORKING
Service - Internal Traffic
syndesis-ui

Routes - External Traffic
[Create Route](#)

RED HAT FUSE ONLINE

- Home
- Integrations
- Connections
- Customizations
- Settings

System Metrics

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

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

Connections

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

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

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

Q&A

Common issues

- openidconnect.net client might have an additional space in the redirect_uri field. That's a client bug, you can fix it by adding an additional redirect URIs in RH SSO with a space preceding the URL: “<https://openidconnect.net/callback>”
- 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