

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.openbanking-0807.openshiftworkshop.com>

End Users credentials:

user[1..100] / openshift

3scale dedicated instances here:

<https://userX-admin.apps.openbanking-0807.openshiftworkshop.com>

access using username and password

dedicated gateway

<https://userX.amp.apps.openbanking-0807.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.openbanking-0807.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).

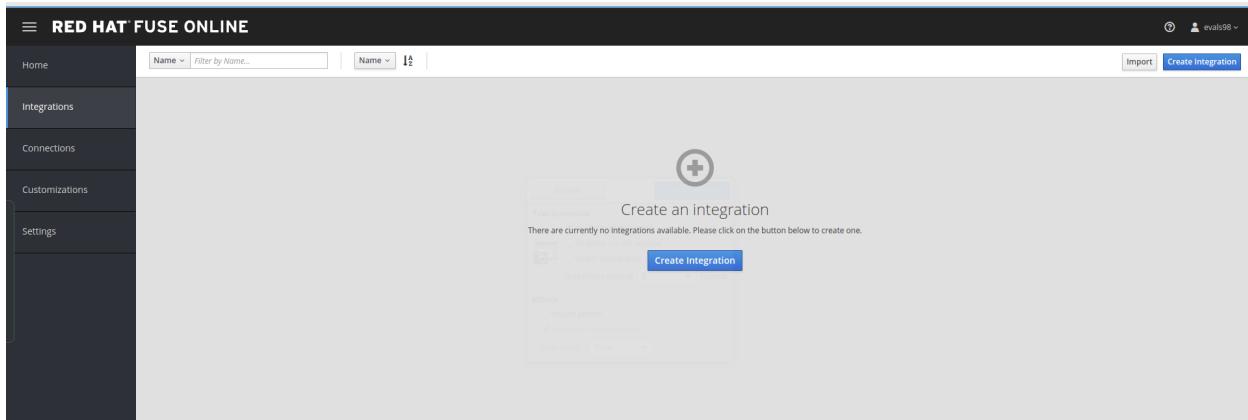
We will see how integr8ly environment works and what you get. It provides out of the box integration between products but it doesn't change the base components that come with it. You can install yourself this type of environment starting from a clean or empty OpenShift installation.

Fuse Online

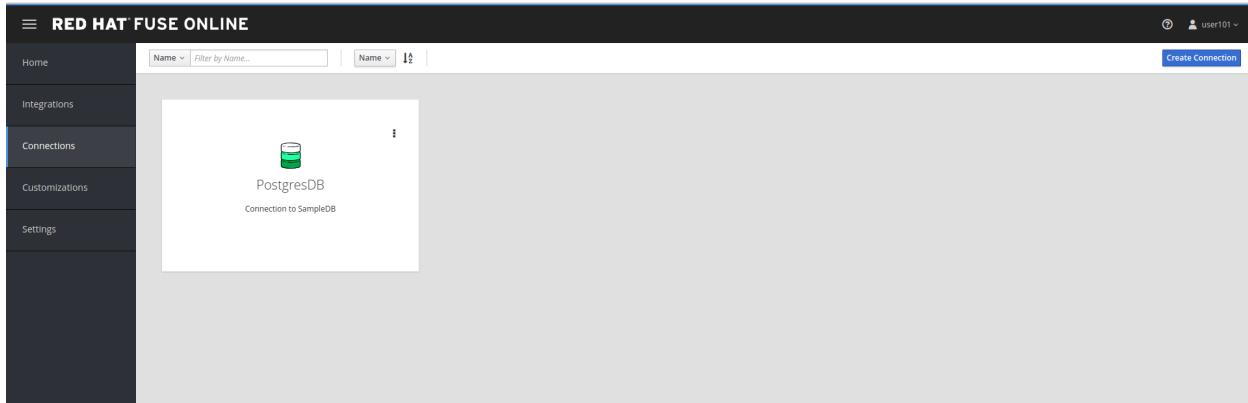
Let's start our first lab session, by opening Red Hat Fuse Online. You will get there by clicking [Red Hat Fuse](#) link in the tutorial dashboard. We will see the main functionalities of it and then use it to build a simple integration block.

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 a 'System Metrics' section with four boxes: '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, accompanied by the text 'There are currently no Integrations. Click the button below to create one.' A 'Create Integration' button is also present. At the bottom, there is a 'Connections' section with four icons: Database (Database icon), File (File icon), Timer (Timer icon), and Webhook (Webhook icon). Buttons for 'View All Connections' and 'Create Connection' are located at the top right of this section.

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.

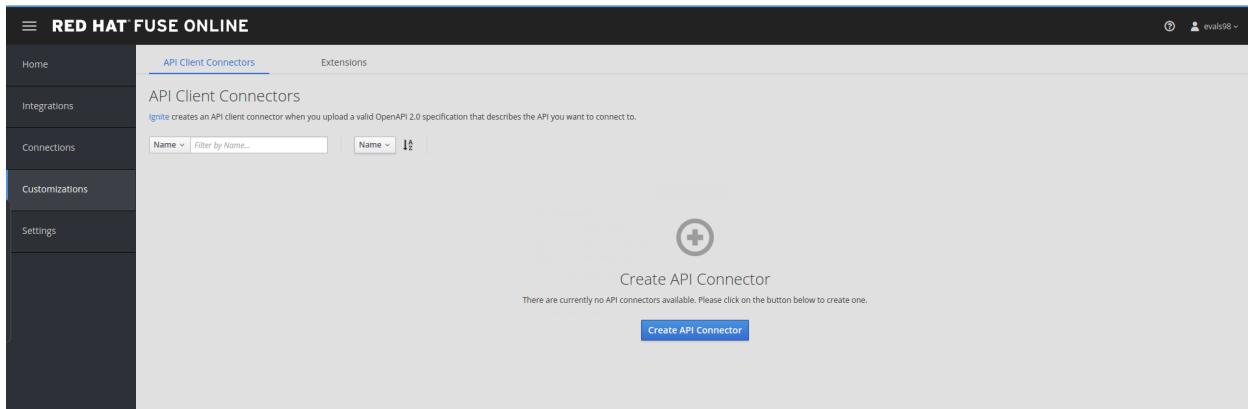


The integration window (**Integrations**) allows you to have a graphical representation of the mediation/transformation and is the first tool to onboard the citizen developers. Behind the curtains there is Camel (which is a proven technology and deployed in highly transactional environment as well) and 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> .



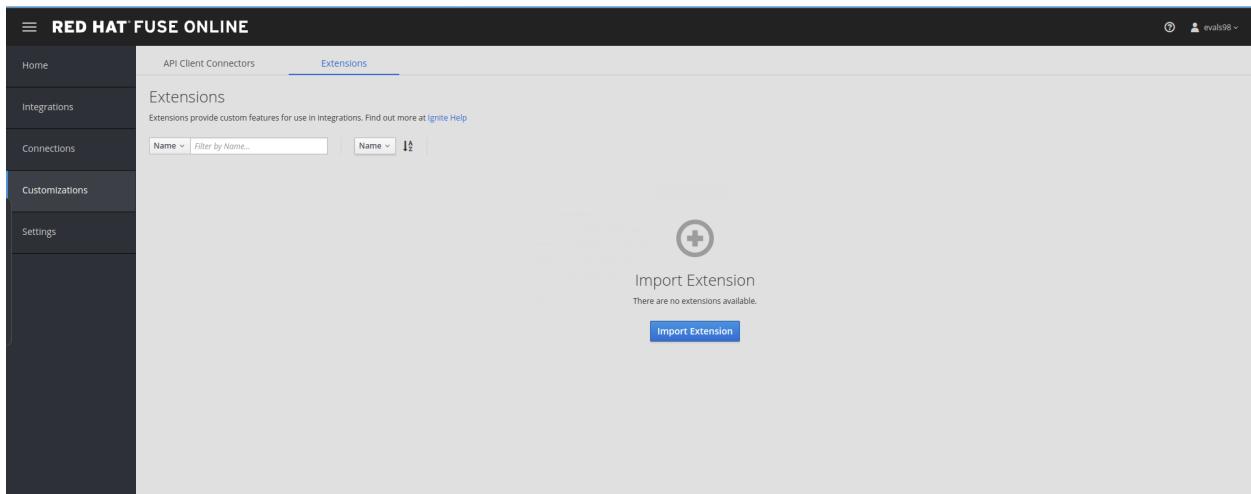
(**Connections**) you can start or end (or sometimes interpose) connections which are fully configured connectors. We will be showing the configuration of one connector during today's lab.

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.

A screenshot of the Red Hat Fuse Online web application. The top navigation bar includes 'RED HAT FUSE ONLINE' and a user icon. The left sidebar has links for 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 is a section titled 'Extensions' with a note: 'Extensions provide custom features for use in integrations. Find out more at [Ignite Help](#)'. There are two dropdown filters: 'Name' and 'Filter by Name...'. A large central area features a plus sign icon inside a circle, labeled 'Import Extension', with the text 'There are no extensions available.' and a blue 'Import Extension' button.

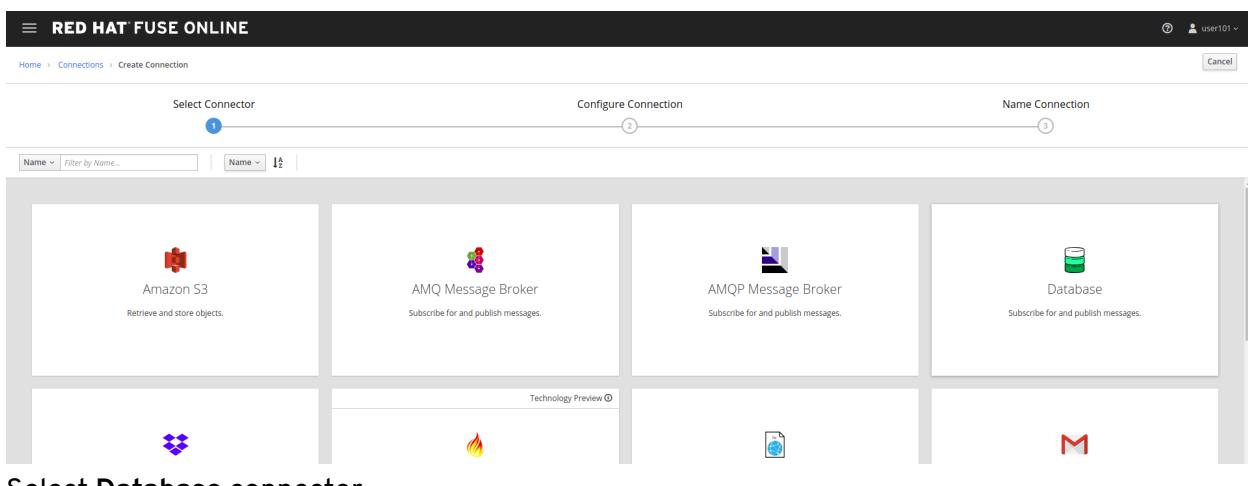
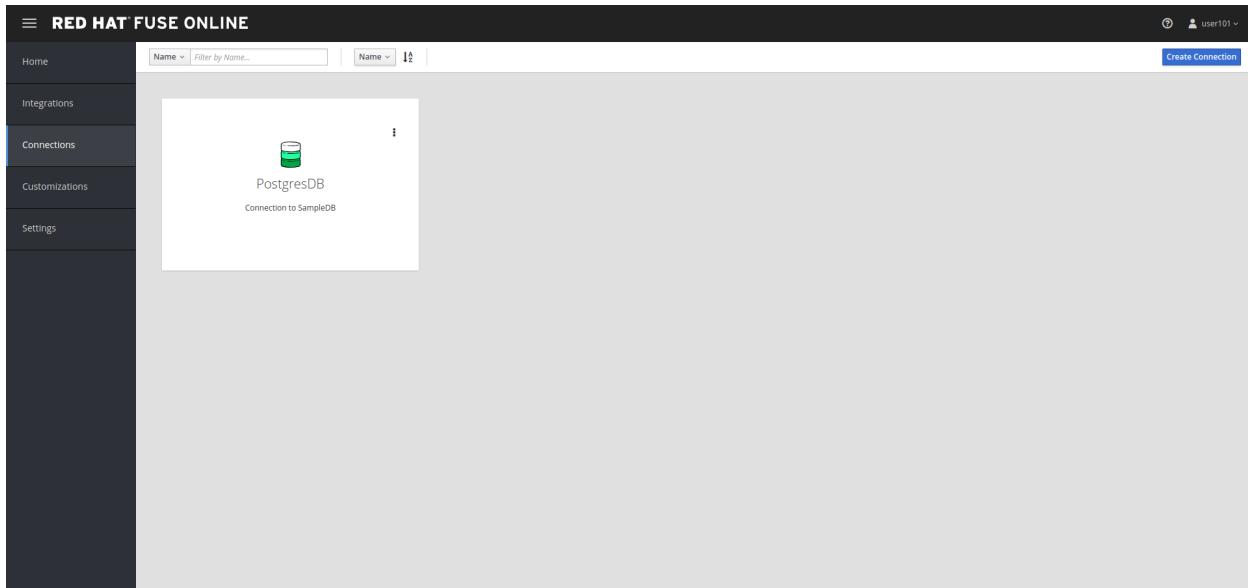
(**Extensions** panel) this is the part where you can extend the functionalities of the platform. You can compile a custom or community available extension to perform a specific functionality or connect to a specific system and in this easy way add transformation functionalities to the product.

LAB BEGINS

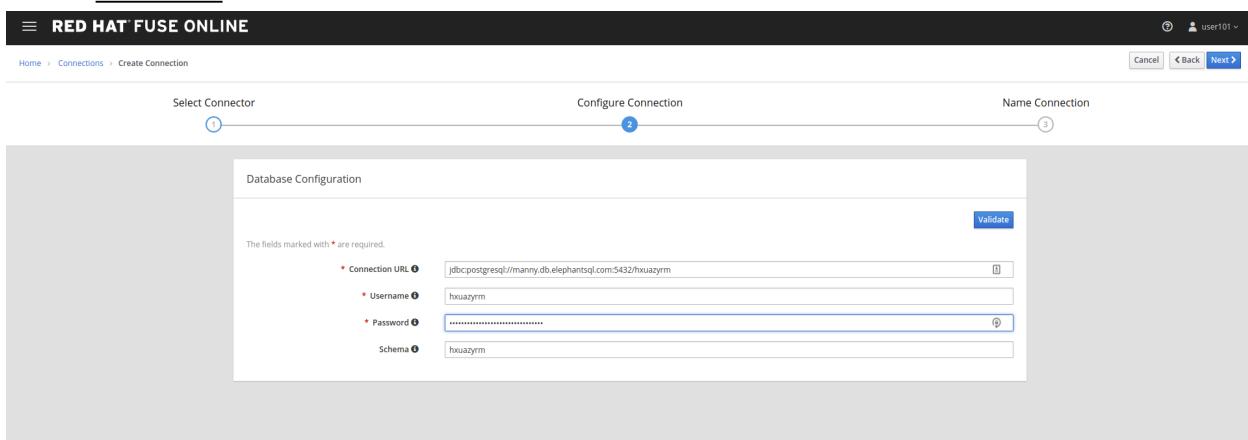
Let's start INTEGRATING now!

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:

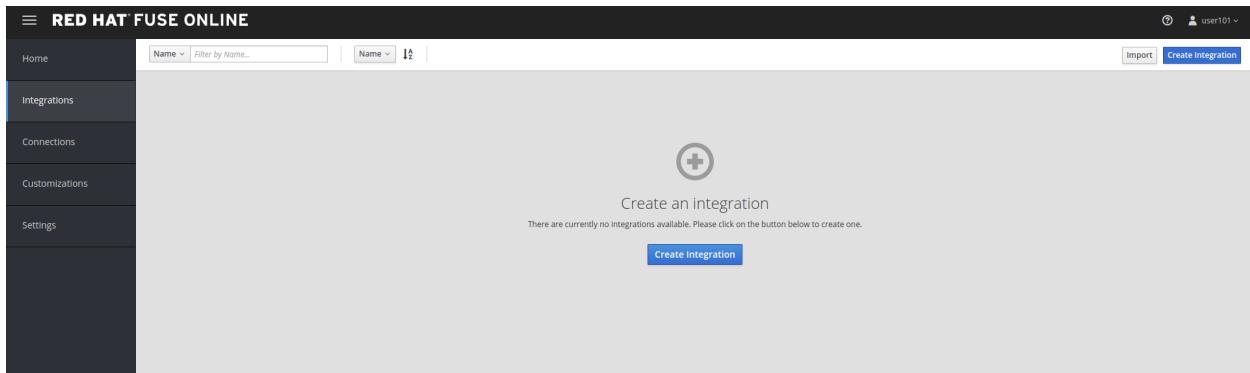
- Connection Name: open-data-banks
- Description: (empty)

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.

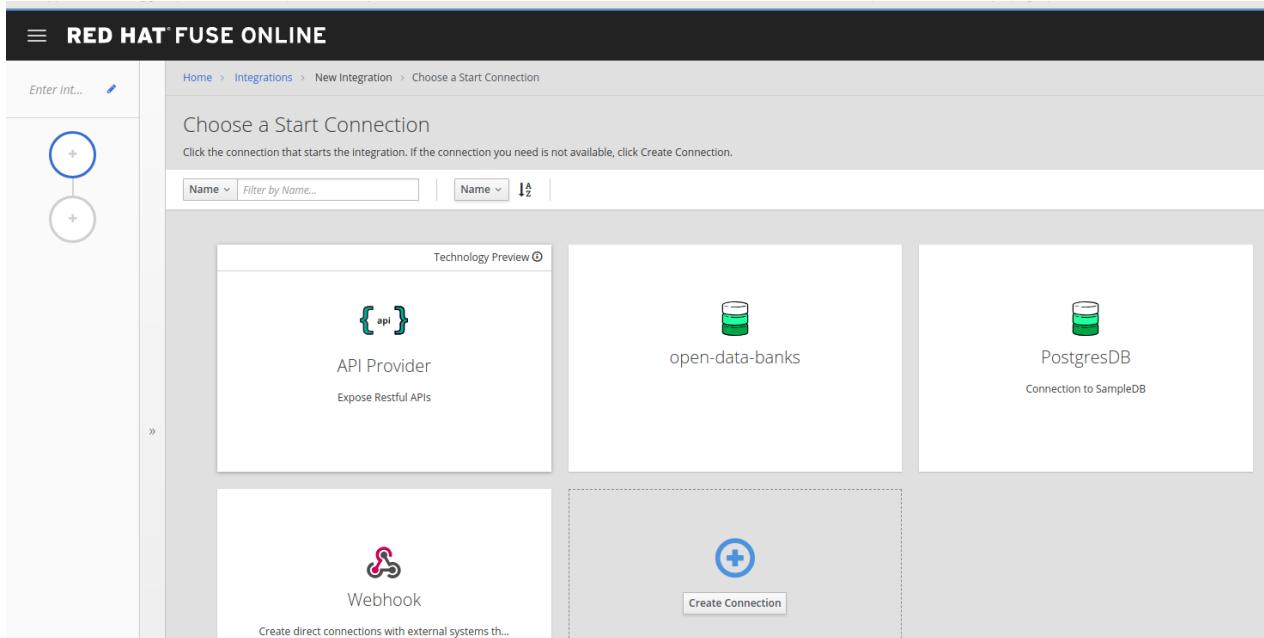
Now that we have configured the end of the integration path we want to build and we will see where to find the start and we will put the two pieces together. **Integrations -> create integration**



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 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 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. The top right corner shows 'Connectir'.

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.

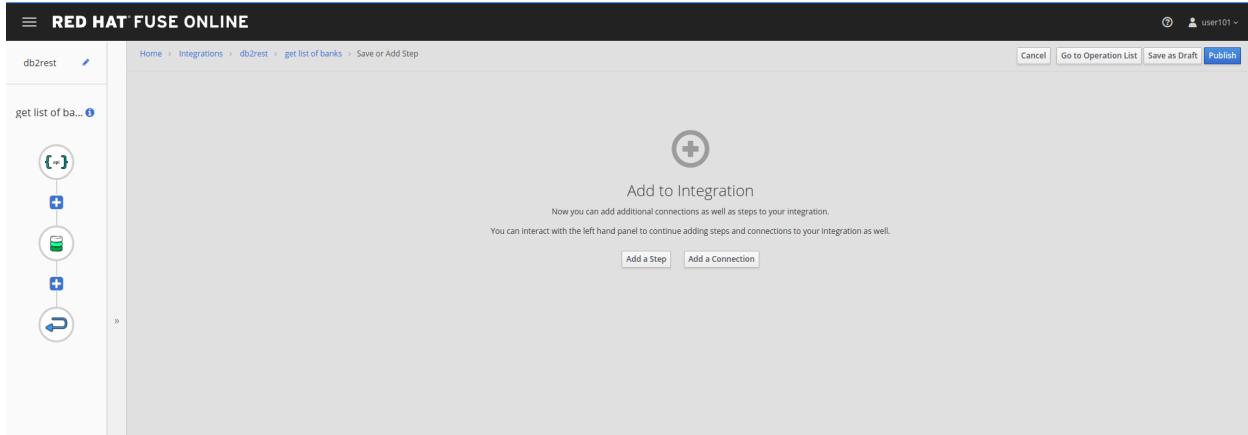
And we will invoke a simple SQL statement to return the data from a single table.

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.

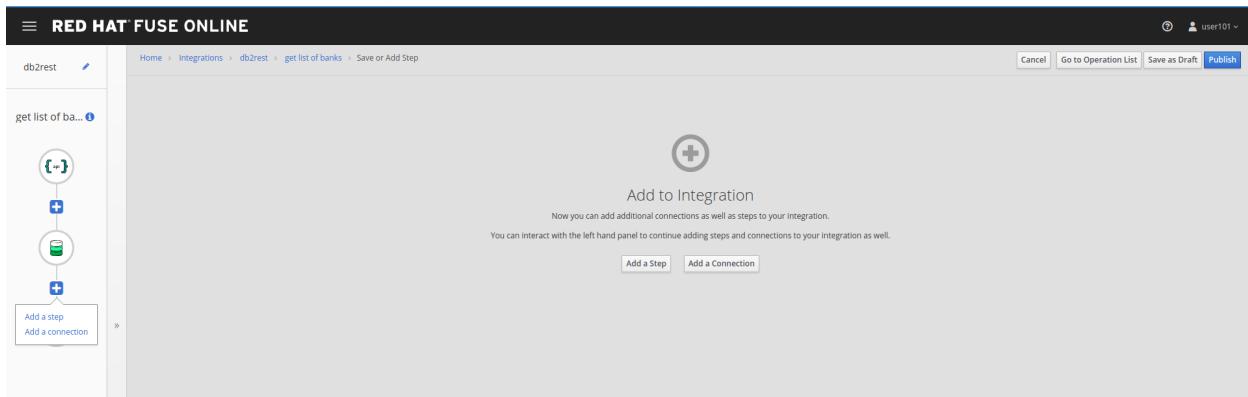
The simple statement to introduce is:

*select * from banks*

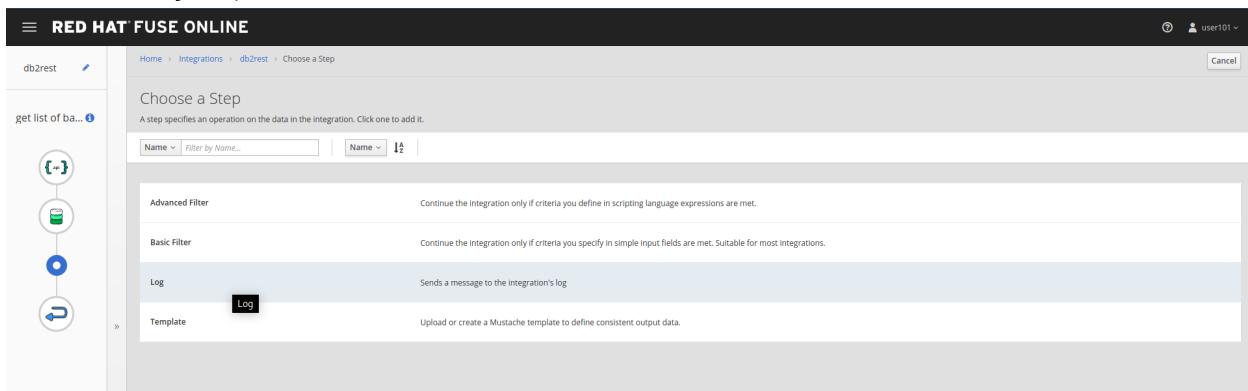
When you click done the statement will validate and you should be able to proceed with the configuration of the integration.



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. The sidebar lists 'db2rest' and 'get list of banks'. The main area features a large blue '+' button with the text 'Add to Integration'. Below the button, there's explanatory text: 'Now you can add additional connections as well as steps to your integration.' and 'You can interact with the left hand panel to continue adding steps and connections to your integration as well.' At the bottom of this section are 'Add a Step' and 'Add a Connection' buttons. The top right corner shows user information: 'user101'.

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 its status 'Assembling (1 / 4)'. Under the 'Details' tab, it shows 'API Provider' and '1 Flow'. There's a note 'no description set...'. At the bottom, there are buttons for 'Draft', 'Publish', and 'Edit'. The 'History' section indicates 'Version: 1'. The top right corner shows 'user101'.

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 <https://i-db2rest-fuse-c831e33d-23ee-11e9-9425-0a580a010007.apps.openbanking-0807.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 main 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 configuration for a 'GET' request to the URL 'https://open-data.b9ad.pro-us-east-1.openshiftapps.com/open-data/banks'. There is a 'Headers' section with a '+ Add Request Header' button. Below the step is a 'Collapse / Expand' button. At the bottom of the step panel are buttons for 'Test' (highlighted in blue), 'Save to Account', and 'Share Test Config'. A note below the buttons says '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'. The footer includes a copyright notice '© 2019 RIGOR. ALL RIGHTS RESERVED' and a 'Powered by RIGOR' logo.

Populate the fields with the following URL

<https://i-db2rest-fuse-c831e33d-23ee-11e9-9425-0a580a010007.apps.openbanking-0807.openshiftworkshop.com/open-data/banks>

and hit **Test**

The screenshot shows the results of a test run. At the top, a green bar indicates a 'PASS' status. To the right, it shows 'N. Virginia' and 'Seconds elapsed: 11'. Below the bar, it says 'Results 6.29 s' and 'Viewing a Request Step'. The results are presented in sections: 'Message' (Step 1: GET http://i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com/open-data/banks passed), 'Connection Information', 'Request' (Request Headers: GET /open-data/banks HTTP/1.1, Host: i-db2rest-fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007.apps.openbanking-4f6a.openshiftworkshop.com, Accept: */*, User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)), 'Response' (Response Headers: HTTP/1.1 200 OK, Transfer-Encoding: chunked, X-Application-Context: application, Date: Mon, 21 Jan 2019 18:20:33 GMT, Set-Cookie: 12b28c7c44ae5989665ef8abec492fd0=08271c1b765ada49ab2c2c05b8b72917; path=/; HttpOnly, Cache-control: private), and 'Response Body' (The response body content: {"short_name": "Bank X", "id": 1, "address": "psd201-bank-x-uk", "long_name": "The Bank of X"}). The 'Response Body' section has an eye icon that can be clicked to expand the content.

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

The screenshot shows a browser window with the URL <https://uptime-diagnostics-response-bodies-production.s3.amazonaws.com/578a-627ab9b000020137d7e0024ac110002/aef86bc2c7>. The page content is a JSON object:

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

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

3scale

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

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, it says "Signed in successfully". Below that, there are two main sections: "AUDIENCE" and "APIS".

AUDIENCE:

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

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

APIS:

- NEW API

API:

- OVERVIEW
- ANALYTICS
- 1 APPLICATION
- 1 ACTIVE DOC
- INTEGRATE THIS API

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

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

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

The screenshot shows the Red Hat 3SCALE API Management interface. At the top, there's a navigation bar with the 3SCALE logo, 'RED HAT 3SCALE API MANAGEMENT', and a 'Dashboard' dropdown. On the right of the dashboard are three icons: a gear, a question mark, and a user profile. Below the navigation is a section titled 'New API'. Underneath this, there are two radio buttons: 'Define manually' (unchecked) and 'Import from OpenShift' (checked). A horizontal line labeled 'SERVICE' follows. Below it are two input fields: 'Namespace' and 'Name', each with a dropdown arrow. To the right of these fields is a blue 'Create Service' button.

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

Authorize Access

3scale is requesting permission to access your account (user101)

Requested permissions

user:full

Full read/write access with all of your permissions

Includes any access you have to escalating resources like secrets

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

[Allow selected permissions](#)

[Deny](#)

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

New API

Define manually
 Import from OpenShift

SERVICE

Namespace
fuse-b5ddd58b-1b5e-11e9-9d4e-0a580a010007

Name
i-mydata

Create Service

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

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

AUDIENCE

1 Signups
last 30 days

POTENTIAL UPGRADES

0 today

ACCOUNTS THAT HIT THEIR USAGE LIMITS IN THE LAST 30 DAYS

BILLING

DEVELOPER PORTAL (0 DRAFTS)

MESSAGES

TODAY

Test created on API

Application Test has been deleted

Test created on API

Application Test has been deleted

Test created on API

Application Developer's App has been deleted

BEFORE TODAY

The sound of silence

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

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

API-> Integration-> Configuration

The screenshot shows the 'Integration' configuration page. On the left sidebar, under 'Integration', 'Configuration' is selected. The main content area is titled 'Integration settings'. It displays two sections: 'Deployment Option' set to 'APIcast' and 'Authentication' set to 'API Key (user_key)'. Below these, a message says 'To get started with this service on APIcast, add the base URL of your API and save the configuration.' A blue button labeled 'add the base URL of your API and save the configuration.' is visible. 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. On the left sidebar, under 'Integration', 'Configuration' is selected. The main content area is titled 'Integration'. It has a section 'Configure & test immediately in the staging environment' with a link to 'documentation'. Below this, there are two main sections: 'API' and 'API GATEWAY'. Under 'API', the 'Private Base URL' is set to 'https://echo-api.3scale.net:443'. Under 'API GATEWAY', the 'Staging Public Base URL' is set to 'https://api-user1-apicast-staging.apps.openbanking-4f6a.openshiftworkshop.com:443' and the 'Production Public Base URL' is set to 'https://api-user1-apicast-production.apps.openbanking-4f6a.openshiftworkshop.com:443'. There are also sections for 'MAPPING RULES' and 'AUTHENTICATION SETTINGS'.

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-0807.openshiftworkshop.com>)

The screenshot shows the Red Hat 3Scale API Management interface. On the left, a sidebar menu includes Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area has tabs for API GATEWAY and MAPPING RULES. Under API GATEWAY, fields for Private Base URL (https://echo-api.3scale.net:443), Staging Public Base URL (https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443), and Production Public Base URL (https://api-user101-apicast-production.apps.openbanking-4f6a.openshiftworkshop.com:443) are shown. Under MAPPING RULES, a table lists a single rule: Verb GET, Pattern /open-data/banks, Metric or Method (Define) 1 hits. A green 'Add Mapping Rule' button is at the bottom.

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.

The screenshot shows the Red Hat 3Scale API Management interface. The sidebar menu is identical to the previous screenshot. The main content area is titled 'AUTHENTICATION SETTINGS'. It contains three sections: Host Header (empty input field), Secret Token (input field containing Shared_secret_sent_from_proxy_to_API_backend_5c9f6c92c06c2ef0, with a note about blocking direct developer requests), and Credentials location (radio buttons for 'As HTTP Headers' (selected) and 'As query parameters (GET) or body parameters (POST/PUT/DELETE)'). Below these is an 'Auth user key' input field containing 'key'. At the bottom is an 'ERRORS' section.

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 under "Policy Chain". One policy is listed: **3scale APICAST** (builtin - Main functionality of APICAST to work with the 3scale API mana...).
- CLIENT**: A section for testing the client. It includes a text input for "API test GET request" containing a placeholder URL and curl command.

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.

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 (no latest applications), Latest alerts (no alerts), and Published Application Plans. A green button labeled "Create Application Plan" is visible. To the right, there is a box titled "Configuration, Methods and Settings" containing instructions about adding an API base URL and deployment status, along with several user permissions listed under "Authenticated by API key".

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

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

The screenshot shows the "Create Application Plan" form. The sidebar is identical to the previous screenshot. The form fields include:

- Name: Input field
- System name: Input field
- Only ASCII letters, numbers, dashes and underscores are allowed.
- Applications require approval?: Description: Set whether or not applications can be created on demand or if approval is required from you before they are activated.
- Trial Period (days): Input field
- Setup fee: Input field (0.00) with USD unit
- Cost per month: Input field (0.00) with USD unit
- Create Application Plan** button

The screenshot shows the Application Plans section of a developer portal. On the left is a sidebar with links: Overview, Analytics, Applications (Listing, Application Plans), ActiveDocs, and Integration. The 'Application Plans' link is highlighted. The main content area has a title 'Application Plans'. It says: 'Application Plans establish the rules (limits, pricing, features) for using your API; every developer's application accessing your API will be accessing it within the constraints of an Application Plan. From a business perspective, Application Plans allow you to target different audiences by using multiple plans (i.e. "basic", "pro", "premium") with different sets of rules.' Below this is a 'Default Plan' dropdown set to 'Default Plan'. A note says 'Default application plan (if any) is selected automatically upon service subscription.' A table lists one plan:

Name	Applications	State	
Basic	0	hidden	Publish Copy Delete

[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 title 'Application Plan Basic'. It contains fields for 'Name' (Basic) and 'System name' (basic). There is a checkbox for 'Applications require approval?' which is unchecked. A note below it says 'Set whether or not applications can be created on demand or if approval is required from you before they are activated.' Below are fields for 'Trial Period (days)' (empty), 'Setup fee' (0.00 USD), and 'Cost per month' (0.00 USD). At the bottom right is a blue button '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

The screenshot shows the 'Metrics, Methods, Limits & Pricing Rules' section under the 'Application Plans' tab. It includes a table for defining metrics and methods, and another table for listing features.

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

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

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

The screenshot shows the 'Accounts' tab. It displays a list of accounts and provides bulk operations options like sending emails, changing account plans, and changing states.

Group/Org.	Admin	Signup Date	Apps	State	Create
Developer	John Doe	18 Jan, 2019	1	Approved	Create

Search for Group/Org., User login, First name, Last name, email, :

[Export all Accounts](#)

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

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

Account: Developer [Edit](#)

Organization/Group Name: Developer [Send message](#)

Administrator: John Doe (admin+test@user101.com)

Signed up on: January 18, 2019 23:47

Status: Approved

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

Application

Name: test
Service: API
Plan: Basic
State: Live

Hits: 0 hits

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

Let's click on **0 Applications** and **Create application**

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

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

New Application

Application plan: Basic

Service plan: Default

Name:

Description:

[Create Application](#)

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

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

CLIENT

API test GET request

```
/
```

Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:

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

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

[Update & test in Staging Environment](#)

[Back to Integration & Configuration](#)

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

CLIENT

API test GET request

```
/
```

Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:

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

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

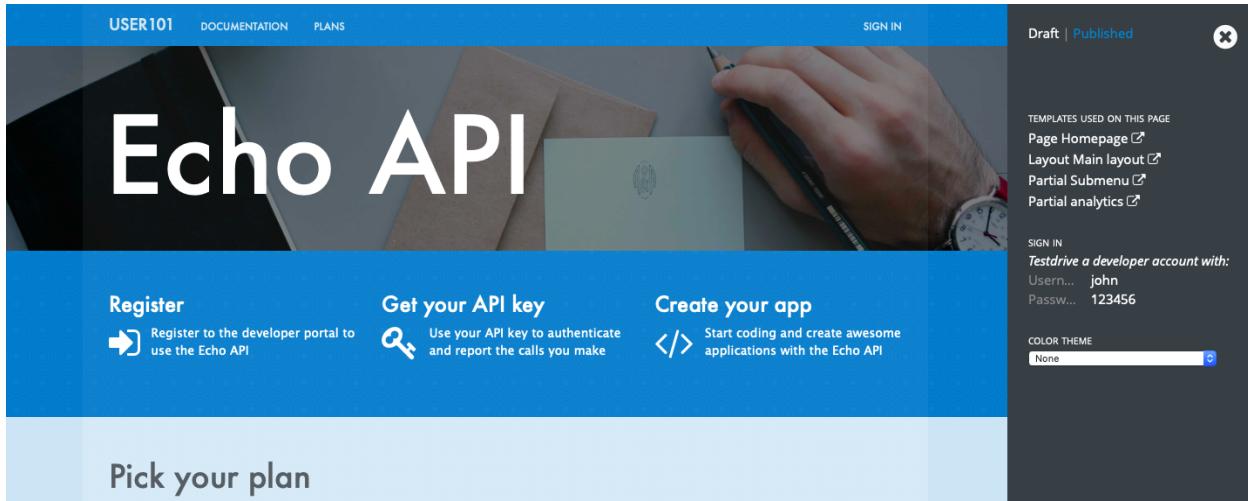
[Update & test in Staging Environment](#)

[Back to Integration & Configuration](#)

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

Let's switch to the developers' point of view by accessing the Developer portal. **Developer portal -> Visit Developer Portal**

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



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

The screenshot shows the developer's dashboard under the 'USER101' account. The top navigation bar includes 'API CREDENTIALS', 'STATISTICS', 'DOCUMENTATION', 'MESSAGES', 'SETTINGS', and a sign-out icon. A success message 'SIGNED IN SUCCESSFULLY' is displayed. The main content area features a large 'Echo API' logo. Below it, a section titled 'Your API Key' contains the text: 'This is your API key that should be kept secret. Use it to authenticate and report the calls you make to the Echo API.' To the right, a modal window titled 'CREDENTIALS' shows the API key details: 'App name: Test' and 'Key: af44089cd553a3b0515772972b5f9ced'. It also includes a note: 'Add this as a `user_key` parameter to your API calls to authenticate.'

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

The screenshot shows the API Management application's interface. At the top, there are tabs for 'USER101', 'API CREDENTIALS', 'STATISTICS', 'DOCUMENTATION', 'MESSAGES', 'SETTINGS', and a help icon. Below the tabs, a section titled 'Applications' contains a single entry for a new application. The application details are as follows:

- Name:** Test
- Description:** Test
- Plan:** Basic > [Review/Change](#)
- Status:** Live (indicated by a green button)
- User Key:** af44089cd553a3b0515772972b5f9ced

Below the User Key, there is a note: "Add this as a `user_key` parameter to your API calls to report and authenticate." A blue button labeled "Regenerate" is present.

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/>

The screenshot shows the apitester.com interface. At the top, there is a search bar with the value 'key' and a button with the ID '4c571db87a82b5aedccbd8'. Below the search bar is a button labeled '+ Add Step'. At the bottom of the header are three buttons: 'Test' (blue), 'Save to Account' (green), and 'Share Test Config' (grey).

The main area displays a test result:

- Status:** PASS
- Results:** 576 ms
- Message:** [Step 1] key financial service passed
- Connection Information:** N. Virginia, Seconds elapsed: 2
- Request:**
 - Request Headers:**

```
GET /open-data/banks HTTP/1.1
Host: wt2-eval198-example-com-3scale.apps.open-banking.opentry.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: 4c571db87a82b5aedccbd8877627a899
```
- Response:**
 - Response Headers:**

```
HTTP/1.1 200 OK
Date: Fri, 11 Jan 2019 18:04:18 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5070
Set-Cookie: JSESSIONID=y09aravw644k1un81kk5f1uy; Path=/
Expires: Fri, 11 Jan 2019 18:04:18 GMT
Last-Modified: Fri, 11 Jan 2019 18:04:18 GMT
```

Use the URL for your API gateway, the following format should be configured in your service already: <https://userX.amp.apps.openbanking-0807.openshiftworkshop.com>, remember the the key Header and the associated value. As we can see we succeed with 200 OK! Let's now just test with a wrong key or path then to confirm the role of API Management.

PASS

Results 111 ms

N. Virginia
Seconds elapsed: 2

Viewing a Request Step

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: c571db87a82b5aedccbd8877627a099
```

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: GeC552533d09895c2bc361d784adc8ddbfdb85f9e6e21ee8a00f6cad673c7c9e5b; 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 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.

Let's start with RH SSO main dashboard

<http://sso.apps.openbanking-0807.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		
scale	True	Not defined	Edit	Export	Delete
account	True	/auth/realm/openshift/account	Edit	Export	Delete
admin-cl	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
launcher-openshift-users	True	Not defined	Edit	Export	Delete
openshift-client	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	/auth/admin/openshift/console/index.html	Edit	Export	Delete

Here we can configure the applications that will authenticate using RH SSO as an IDP. As we can see there are some default clients dedicated to authentication in the integr8ly environment.

Users -> View all users

ID	Username	Email	Last Name	First Name	Actions		
0edb11ae-1292-4ca9-83d5-c1565b...	admin@example.com	admin@example.com			Edit	Impersonate	Delete
fe06b695-2c4c-46a8-b1f4-e11940b...	eval01@example.com	eval01@example.com			Edit	Impersonate	Delete
31dc7463-c473-48fa-9f19-41191b6...	eval02@example.com	eval02@example.com			Edit	Impersonate	Delete
5fb5c5d695-41db-9f57-a8128bc...	eval03@example.com	eval03@example.com			Edit	Impersonate	Delete
636eaafb-91f5-4a69-8cb8-f6585374...	eval04@example.com	eval04@example.com			Edit	Impersonate	Delete
6e601bed-1f6d-4c26-97f6-459f2d3...	eval05@example.com	eval05@example.com			Edit	Impersonate	Delete
14423744-1dbd-4072-a55b-429...	eval06@example.com	eval06@example.com			Edit	Impersonate	Delete
7776e676-c68d-4ade-9a38-2b6...	eval07@example.com	eval07@example.com			Edit	Impersonate	Delete
97b191d8-1860-4bd4-a3bb-2ea...	eval08@example.com	eval08@example.com			Edit	Impersonate	Delete
b3270190-e0b3-472d-9f16-2cd...	eval09@example.com	eval09@example.com			Edit	Impersonate	Delete
8c7ab5b0-1771-4b2e-b15a-488176...	eval10@example.com	eval10@example.com			Edit	Impersonate	Delete
abaa0b00-03b9-479a-b758-7a572...	eval10@example.com	eval10@example.com			Edit	Impersonate	Delete
c7719640-2e43-409c-bc70-23725d1...	eval11@example.com	eval11@example.com			Edit	Impersonate	Delete
e949e95b-40c3-470e-a304-342018b...	eval12@example.com	eval12@example.com			Edit	Impersonate	Delete
fd54725b-546d-4505-bc51-36434f8...	eval13@example.com	eval13@example.com			Edit	Impersonate	Delete
ba20303af049-4847-9eb6-1967dc...	eval14@example.com	eval14@example.com			Edit	Impersonate	Delete
58814183-0c84-4aff-92e8-e9b0f83...	eval15@example.com	eval15@example.com			Edit	Impersonate	Delete
9cae975b-6dff-4564-a44c-97a34117...	eval16@example.com	eval16@example.com			Edit	Impersonate	Delete
332dc59c-a342-4f71-9292-21cebf07...	eval17@example.com	eval17@example.com			Edit	Impersonate	Delete
37df7c70-1bd3-4499-9f93-c1c8b249...	eval18@example.com	eval18@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 Clients and they will be able to authenticate with the . 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...
- Impersonate user:** Impersonate

Buttons for Save and Cancel are at the bottom.

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.

DEMO ONLY

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**

The screenshot shows the 'Clients' section of the Red Hat SSO interface. On the left, a sidebar lists 'Clients' as the active category. The main area is titled 'Add Client' and contains the following fields:

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

Buttons for Save and Cancel are at the bottom.

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

The screenshot shows the Red Hat Single Sign-On interface under the 'Clients' section for the 'sync-app' client. The 'Service Account Roles' tab is selected. On the left, the navigation sidebar shows 'Clients' is currently selected. The main area displays four tabs: 'Realm Roles', 'Available Roles', 'Assigned Roles', and 'Effective Roles'. Under 'Client Roles', 'realm-management' is selected. The 'Available Roles' tab lists several roles: 'create-client', 'impersonation', 'manage-authorization', 'manage-events', and 'manage-identity-providers'. The 'Assigned Roles' tab contains 'offline_access' and 'uma_authorization'. The 'Effective Roles' tab also contains 'offline_access' and 'uma_authorization'.

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

The screenshot shows the Red Hat Single Sign-On interface under the 'Users' section. The navigation sidebar shows 'Users' is currently selected. The main area displays a table of users. One user is listed: ID 'Bed23341-d96a-419b-aa9d-a8fd...', Username 'threescale', Email 'threescale@openshiftworkshop.com', Last Name 'Name', First Name 'Name', Actions: 'Edit', 'Impersonate', and 'Delete'. There is a 'Lookup' button above the table.

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

The screenshot shows the Red Hat Single Sign-On interface under the 'Users' section with the 'Add user' sub-section selected. The navigation sidebar shows 'Users' is currently selected. The main area displays an 'Add user' form. Fields include: ID (empty), Created At (empty), Username * (set to 'john'), Email (set to 'john@example.com'), First Name (set to 'John'), Last Name (set to 'Doe'), User Enabled (set to 'ON'), Email Verified (set to 'ON'), Required User Actions (dropdown menu 'Select an action...'), and Save/Cancel buttons.

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 interface. On the left, a sidebar menu includes 'Configure' (Realm Settings, Clients, Client Templates, Roles, Identity Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events, Import, Export). The main area shows 'Users > John'. The 'Credentials' tab is selected. Under 'Manage Password', fields for 'New Password' and 'Password Confirmation' are present, along with a 'Temporary' switch set to 'OFF'. A 'Reset Password' button is available. Below this is a 'Credential Reset' section with 'Reset Actions' (a dropdown menu), 'Expires In' (set to 12 Hours), and a 'Reset Actions Email' button.

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 lists 'Overview', 'Analytics', 'Applications', 'ActiveDocs', 'Integration' (selected), 'Configuration', 'Methods & Metrics', and 'Settings'. The main area is titled 'Configuration'.

- Integration settings:** Deployment Option is set to 'APIcast' and Authentication is 'API Key (user_key)'.
- APIcast Configuration:** Private Base URL is 'https://echo-api.3scale.net:443', Mapping rules are '/ => hits', Credential Location is 'query', and Secret Token is 'Shared_secret_sent_from_proxy_to_API_backend_1664772f8f98f392'.
- Environments:** A 'Staging Environment' is listed with URL 'https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443' and version 'v. 4'. A 'Promote v. 4 to Production' button is available.

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

AUTHENTICATION

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

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

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

OpenID Connect
Use OpenID Connect for any OAuth 2.0 flow.

[Update Service](#)

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

AUTHENTICATION

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

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

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

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

[Update Service](#)

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

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

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

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

<http://client-id:client-secret@<idp-public-endpoint>>

where client-id: sync-app

client secret: 05e5e757-eb63-4d4a-b7cc-cff6a7ac55df

idp-public-endpoint: <http://sso.apps.openbanking-0807.openshiftworkshop.com/auth/realm/threescale>

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. In the APIcast Configuration section, the Private Base URL is set to `https://echo-api.3scale.net:443`, Mapping rules are set to `/ => hits`, Credential Location is `query`, and Secret Token is `Shared_secret_sent_from_proxy_to_API_backend_1664772f8f98f392`. In the Environments section, there are two environments: Staging Environment and Production Environment. The Staging Environment has a URL of `https://user101.amp.apps.openbanking-4f6a.openshiftworkshop.com:443` and a version of v. 5. It features a blue button labeled "Promote v. 5 to Production". The Production Environment has no configuration saved yet.

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

The screenshot shows the Applications configuration page for USER101. The page includes tabs for USER101, API CREDENTIALS, STATISTICS, and DOCUMENTATION. Under the Applications section, there is a Test application with the following details:

Name	Test
Description	Test
Plan	Basic > Review/Change
Status	Live
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. <input type="text" value="REIRECT URL"/>

A green "Submit" button is located at the bottom of the form.

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 'USER101' dashboard with the 'API CREDENTIALS' tab selected. Under the 'Applications' section, there is a single application named 'test'. The application details are as follows:

- Name:** test
- Description:** (No description provided)
- Plan:** Basic > [Review/Change](#)
- Status:** Live (indicated by a green button)
- Client ID:** 245e2c45
This is the Client ID you should send with each API request.
- Client Secret:** 84f7fc184b5210f98bba2290a0474fd3
This is the Client Secret used to authenticate requests.
[Regenerate](#)
- Redirect URL:** This is your Redirect URL for OAuth.

A green 'Submit' button is located at the bottom of the form.

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

The screenshot shows the 'Clients' section of the Red Hat SSO developer portal. A client named '5bc94f6a' has been created with the following settings:

- Client ID:** 5bc94f6a
- Name:** my first finance app - sz
- Description:** my first finance app - sz
- Enabled:** ON
- Client Protocol:** openid-connect
- Access Type:** confidential
- Standard Flow Enabled:** ON
- Implicit Flow Enabled:** ON
- Root URL:** (empty)
- Valid Redirect UrIs:** https://openidconnect.net/callback

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-0807.openshiftworkshop.com/auth/realms/threescale/.well-known/openid-configuration>

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

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 https://openidconnect.net/callback 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"/>
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 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_32RqnRPCixM00YQePoPt5i6NiKjSe8hZWInWah
Bj4Rz0bm53WCukRf06m3LtZLQZ0t-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- FZLQP0aPD1mGwe70KDjMI_32RqnRPCixM00YQePoPt5i6NiKjSe8hZWInWah Bj4Rz0bm53WCukRf06m3LtZLQZ0t-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-
FZLQP0aPD1mGwe7OKDJMI_32RqnrcpixM00YQePoPt5i6N1cKjSe8hZWInw
Bj4RzQbM53WCukRf06m3LtZLQZ0t-XI4eJGo8GPrsFPP37PuFtkZ-
3m7oubNDyF_ZK5msqY7Ir7x8v0K3vlKCPI0F1ZRCY4_my_oyplHCbeJa.jL

HTTP/1.1 200
Content-Type: application/json
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwi
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwi
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwiia2lk
  "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

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiwiia2lkIiA6ICJRa1RJX2Vws2G0P5v5l7D-CDJpLxoCvRr9gNKpBoBb9KFvvyyW8I6l6YX6B38rJAJ5wTCXZkyfUaSruImaLWNpEPODhiYxNg7mmgjxKKYV8bMUW80yB7wSvCGotvF_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](https://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 ID 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/openid-connect", "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:

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiwiia2lkIiA6ICJRa1RJX2Vws2lwNvpFsKp3ZTd1cnFQUWtjSERNRiSmhGcEtzUJ2aCIVn0.eyJqdGkiOilyYzJmZjQ5Zs01MDY4LTQ0MjQtYTRINS05MWU3OTk3MTM0YTMIcJleHaiOjE1NDczOTc1NTlsIm5iZl6MCwiaWF0ljoxNTQ3Mzk2NjUyLCJpc3MiOjJodHRwczovL3NIY3VyZs1zc28tc3NvLmFcHMub3Blbi1YW5raW5nLm9wZWS50cnkubWuvYXV0aC9yZWFsBxMvb3B1bnN0aWZ0IiwiYXVkiJoiNWjJOTrmNmEiLCJzdW1oIiyMzZjZDRhMy01MGm2LQtQ3j0tyWUzZC05ODdiYjA1ZTk4MzY1LCJ0exXA10iJJCrcisImF6cC16IjViYzKjZjHliwiYXV0aW90aW11ijoNTQ3Mzk1ODc2LCJzJXNzaW9uX3NBYXR1IjoiZmExND11OGItZDR10s000Wjl1Tk1atOTYxM2ZmMGE1NWJkIiwiYNYiJoi1MC1sInByZWZlcnJ1Zf91c2VybmtfZSI6ImV2YXwx0ThAZXhhbXsZS5jb201CJ1bWFpbC16ImV2YXwx0ThAZXhhbXsZS5jb201fQ_dxqOTu9ATiAVGWMFFje-2G0P5v5l7D-CDJpLxoCvRr9gNkBoBb9KFvvyyW8I6l6YX6B38rJAJ5wTCXZkyfUaSruImaMh7fmPlfrFxNAGTg21cQhHTCpTx6vqCmhg5Po_Cy1Tv6dwEcrxekVvn16FKkoWOpEuGvAmA1y5Vkmlo1XR8C0TRKfdWNZF6AT9LnhZn5npHH28qRisYWTzyYGJuuQmxA272R7QKZTxXNPjZB0KPUachbDvhCmBn5YGPQLUPKKBdCQTN2z9uFngVjLQ1tka-
```

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>

API TESTER
BETA

Sign In Create Account ⓘ

Build your test

[View example](#)

Click ⚙ to add or remove steps

[Collapse / Expand](#)

Request ▾ Step Name ☰ ⌄

GET ▾

https://wt2-evals99-example-com-3scale.apps.open-banking.openentry.me

Headers

+ Add Request Header

Authorization Bearer eyJhbGciOiJSUzI1NiI

Let's hit Test

```
Request
Request Headers
GET /open-data/banks HTTP/1.1
Host: w2t-eval99-example-com-3scale.apps.open-banking.opentry.me
Accept: /*
User-Agent: Mozilla/5.0 (compatible; Riger/1.0.0; http://riger.com)
Authorization: Bearer eyJhbGciOiJSUzIiNlIsInR5cC Ig0IAiSlUiIwia2IkIA6ICJRa1RXj2VwS2IwVpFsKp3ZTd1cnFQlWtjSERNr1iSMnh6cE1tZUj2aC1Vn0.eyJqdGk
4

Response
Response Headers
HTTP/1.1 200 OK
Server: openresty/1.13.6.1
Date: Sun, 13 Jan 2019 16:26:57 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5076
Set-Cookie: JSESSIONID=hpuy86svua959siq0hh3ny; Path=/
Expires: Sun, 13 Jan 2029 16:26:57 GMT
Access-Control-Allow-Origin: *
Cache-Control: no-cache, private, no-store
Correlation-Id: hquy86svua959siq0hh3ny
Pragma: no-cache
X-Frame-Options: DENY
Content-Security-Policy: max-age=31536000; includeSubdomains
X-Options: X-OpenBankProject
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src 'self' https://static.openbankproject.
Referrer-Policy: no-referrer_when_downgrade
Set-Cookie: 4e3rdrf6527267fd1ac88b37832037=020629bc3fa9aa2edf5e883d9565451; path=/; HttpOnly; Secure
Set-Cookie: 0ced25c053039ca80739ad95d21874=f08579e621e8ad00f8cad673c7c9e5b; path=/; HttpOnly; Secure
4

Response Body
{"banks": [{"id": "psd201-bank-x-uk", "short_name": "Bank X", "full_name": "The Bank of X", "logo": "https://static.openbankproject.com/images/sandb
4

Variables
Search 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 interface. On the left, the 'Browse Catalog' section displays a grid of service icons categorized by language (.NET, .NET, .NET, .NET), deployment type (3scale, 3scale-gateway), and specific applications like amp-apicast-wildcard-router, amp-pvc, Apache HTTP Server, CakePHP + MySQL, etc. 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 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.

The image consists of two vertically stacked screenshots of the OpenShift Cluster Console. Both screenshots show the 'Workloads' section with a list of pods. A 'Delete Pod' dialog box is open in both cases.

Screenshot 1 (Top):

- The dialog title is 'Delete Pod'.
- The message inside the dialog reads: 'Are you sure you want to delete **syndesis-ui-1-c6gc9** in namespace **fuse-dcfbb062-1520-11e9-86c5-0a580a810008**?'
- At the bottom right of the dialog are 'Cancel' and 'Confirm' buttons.
- The background shows a list of pods:
 - syndesis-operator-1-b5tlr**: Running, Ready
 - syndesis-prometheus-1-6dcfz**: Running, Ready
 - syndesis-server-1-jhmn**: Running, Ready
 - syndesis-ui-1-c6gc9**: Running, Ready
 - todo-1-8bdk4**: Running, Ready

Screenshot 2 (Bottom):

- The dialog title is 'Delete Pod'.
- The message inside the dialog reads: 'Are you sure you want to delete **syndesis-ui-1-c6gc9** in namespace **fuse-dcfbb062-1520-11e9-86c5-0a580a810008**?'
- A red error box contains the message: 'pod "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'
- At the bottom right of the dialog are 'Cancel' and 'Confirm' buttons.
- The background shows a list of pods identical to the first screenshot.

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

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

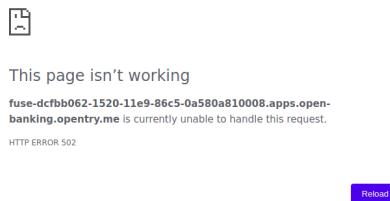
Actions

Cancel **Delete**

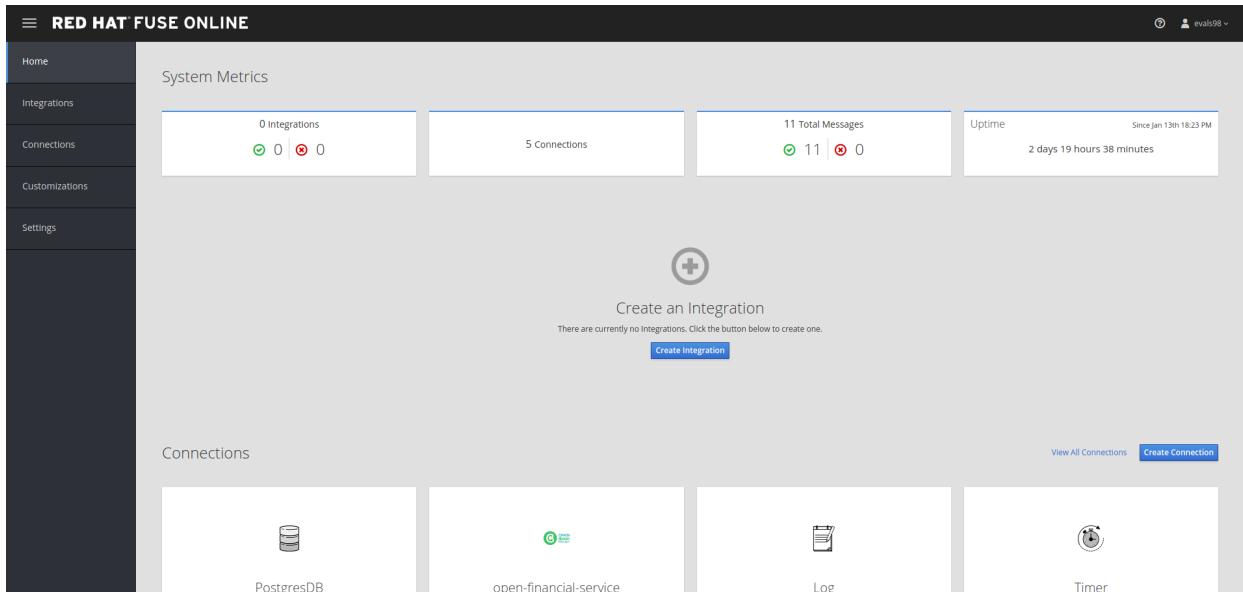
Status		Template	
Status:	Running	Containers	syndesis-ui
Deployment:	syndesis-ui, #1	Image:	fuse7/fuse-ignite-ui 0c05f43 100.0 MB
IP:	10.130.6.14	Ports:	8080/TCP
Node:	ip-172-31-3-6.ec2.internal (172.31.3.6)	Mount:	config-volume .. /usr/share/nginx/html/config read-write
Restart Policy:	Always	Mount:	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 for a project named 'fuse-dcfbb062-1520-11e9-86c5-0a580a810008'. The left sidebar includes 'Overview', 'Applications', 'Builds', 'Resources', 'Storage', 'Monitoring', and 'Catalog'. The main content area displays several deployment configurations: 'syndesis-meta, #1', 'syndesis-oauthproxy, #1', 'syndesis-operator, #1', 'syndesis-prometheus, #1', 'syndesis-server, #1', and 'syndesis-ui, #1'. The 'syndesis-ui' entry is highlighted with a yellow background and has a red 'X' icon indicating it is marked for deletion. A circular summary at the bottom right shows '1 pod'. Resource usage details for each pod are listed below their respective entries.

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



The screenshot shows the OpenShift Container Platform Application Console for the same project. The 'syndesis-ui' pod is now listed with its original resource requirements: 340 MiB Memory, < 0.01 Cores CPU, and 0.1 Kib/s Network. The circular summary at the bottom right still shows '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