



Open Banking Workshop

Red Hat Workshop
March 2019

Table of Contents

Introduction	3
Practical Lab - Part 1	4
FUSE ONLINE	4
3SCALE API MANAGEMENT PLATFORM	17
Practical Lab - Part 2	35
RED HAT SINGLE SIGN ON AND 3SCALE OIDC	35

Introduction

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 Lab - Part 1

Log into the Red Hat Solution Explorer with your user id and the password “openshift”.

The URL has the following structure:

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

(with the GUID replaced with the value provided by your instructor).

The screenshot shows the Red Hat Solution Explorer interface. On the left, there's a sidebar with "Start with a walkthrough" and "Writing Walkthroughs". The main area displays three walkthrough cards: "Publishing walkthroughs to a cluster" (25 min), "Integrating message-oriented middleware with a RESTful API using AMQ" (15 min), and "Integrating message-oriented middleware with a RESTful API using AMQ Online" (21 min). To the right, there's a section titled "Applications" listing various services: Red Hat OpenShift (Ready for use), Red Hat 3scale API Management Platform (Ready for use), Eclipse Che (community, Ready for use), Red Hat Fuse Online (Ready for use), Red Hat Developer Launcher (community, Ready for use), 3scale Developer Dashboard (custom, Ready for use), and 3scale Admin Dashboard (custom, Ready for use).

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

FUSE ONLINE

Let's start our first lab session, by opening Red Hat Fuse Online. You will get there by clicking [Red Hat Fuse Online](#) link in the tutorial dashboard.

When using Fuse Online for the first time, you will have to authorize access by clicking on "Allow selected permissions" and having both permissions checkboxes selected.

We will see the main functionalities of it and then use it to build a simple integration block.

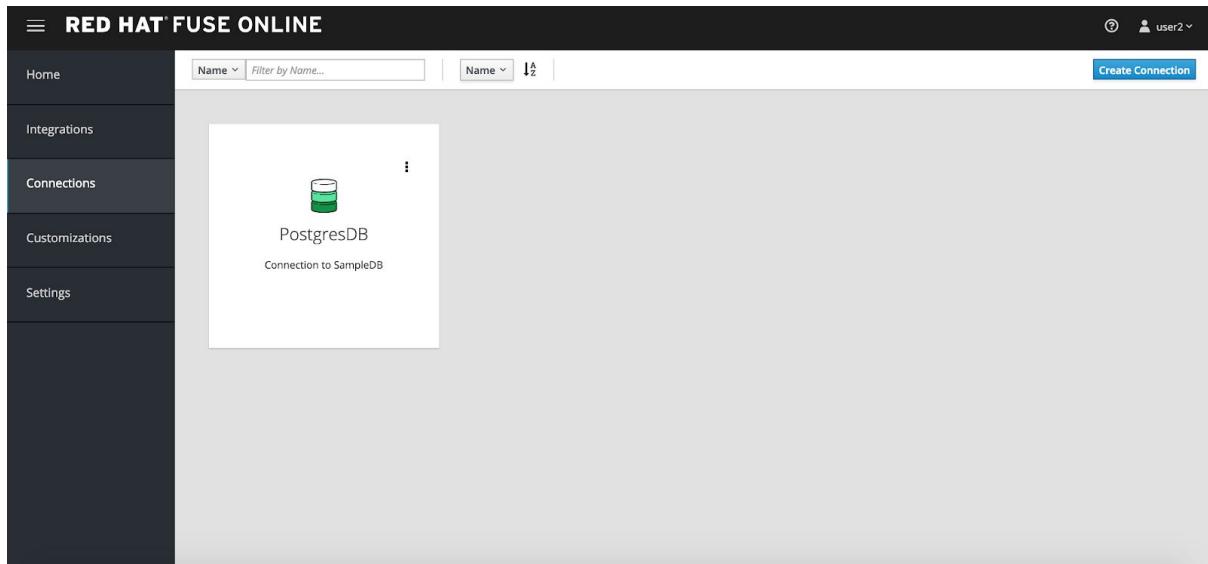
The screenshot shows the Red Hat Fuse Online interface. At the top, there's a navigation bar with the title 'RED HAT FUSE ONLINE' and a user icon 'user2'. On the left, a vertical sidebar lists 'Home', 'Integrations', 'Connections', 'Customizations', and 'Settings'. The main area is titled 'System Metrics' and displays four cards: '0 Integrations' (green circle with 0, red circle with 0), '1 Connections' (green circle with 1, red circle with 0), '0 Total Messages' (green circle with 0, red circle with 0), and 'Uptime Since Mar 11th 15:34 7 minutes'. Below these metrics is a large central button with a plus sign and the text 'Create an Integration'. A smaller message below it says 'There are currently no Integrations. Click the button below to create one.' and a blue 'Create Integration' button.

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 screenshot shows the 'Integrations' window of Red Hat Fuse Online. It has a similar layout to the home page, with a sidebar on the left and a main content area. The main content includes a search bar with 'Name' dropdowns and 'Filter by Name...', a 'Create Integration' button, and a large central 'Create an integration' button with a plus sign. A message below it states 'There are currently no integrations available. Please click on the button below to create one.' There are also 'Import' and 'Create Integration' buttons at the top right of the main area.

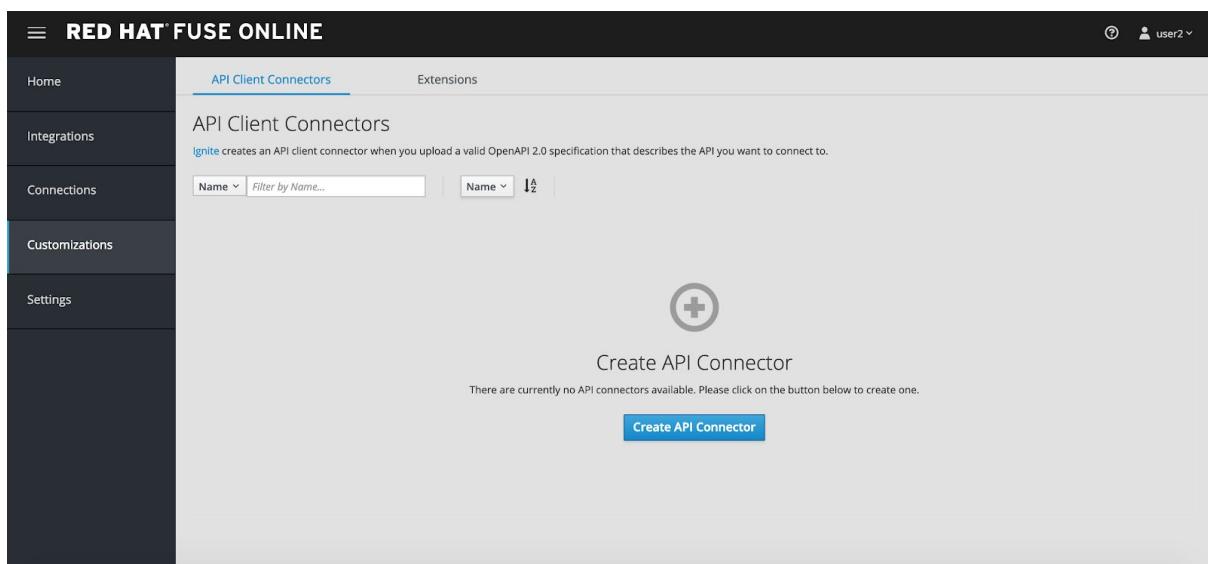
The integration window ([Integrations](#)) allows you to have a graphical representation of the mediation/transformation and is the first tool to onboard the citizen developers.

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> .

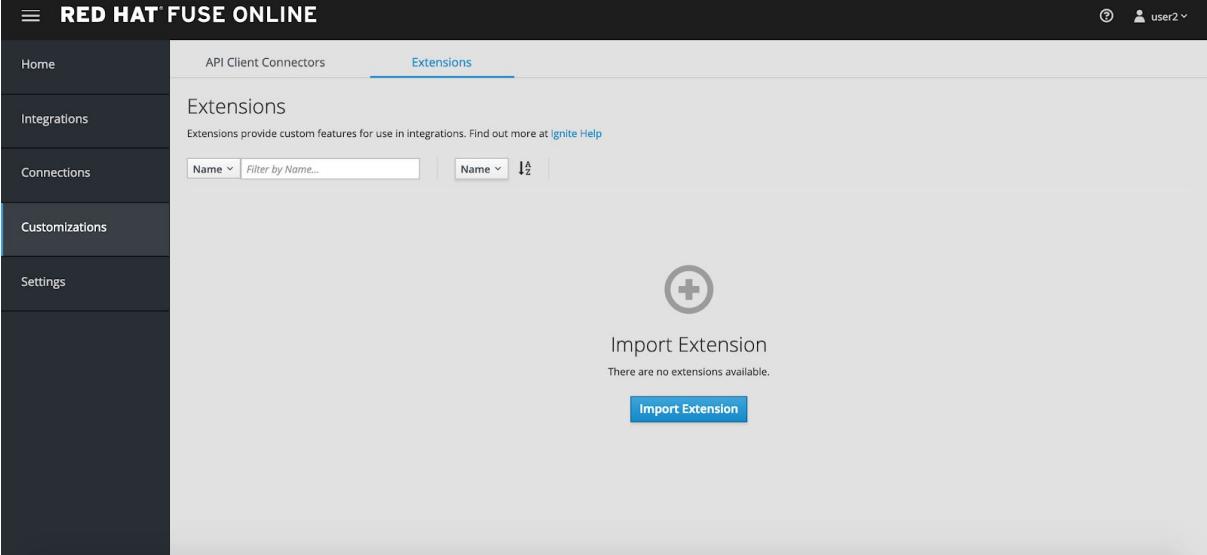


(**Connections**) you can start or end (or sometimes interpose) connections which are fully configured connectors. We will be showing the configuration of one connector during today's lab.

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 shows "RED HAT FUSE ONLINE" and a user icon for "user2". The left sidebar has menu items: Home, Integrations, Connections, Customizations (which is selected and highlighted in blue), and Settings. The main content area has tabs: "API Client Connectors" and "Extensions" (which is active and highlighted in blue). Below the tabs, it says "Extensions provide custom features for use in integrations. Find out more at [Ignite Help](#)". There are two search/filter input fields: "Name" and "Filter by Name...". In the center, there is a large circular button with a plus sign and the text "Import Extension". Below it, a message says "There are no extensions available." and a blue "Import Extension" button.

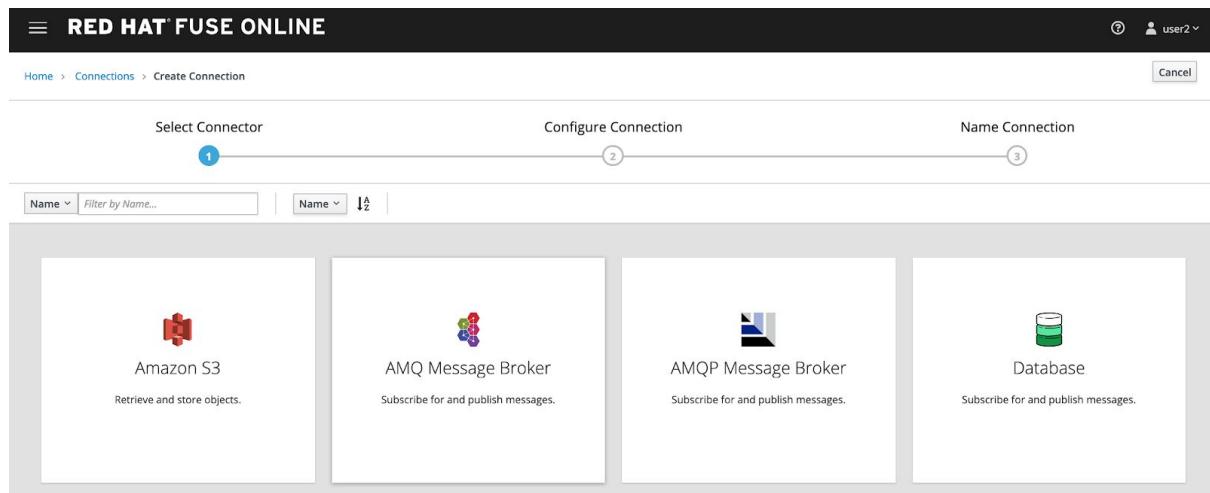
(Extensions panel) this is the part where you can extend the functionalities of the platform. You can compile a custom or community available extension to perform a specific functionality or connect to a specific system and in this easy way add transformation functionalities to the product.

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.

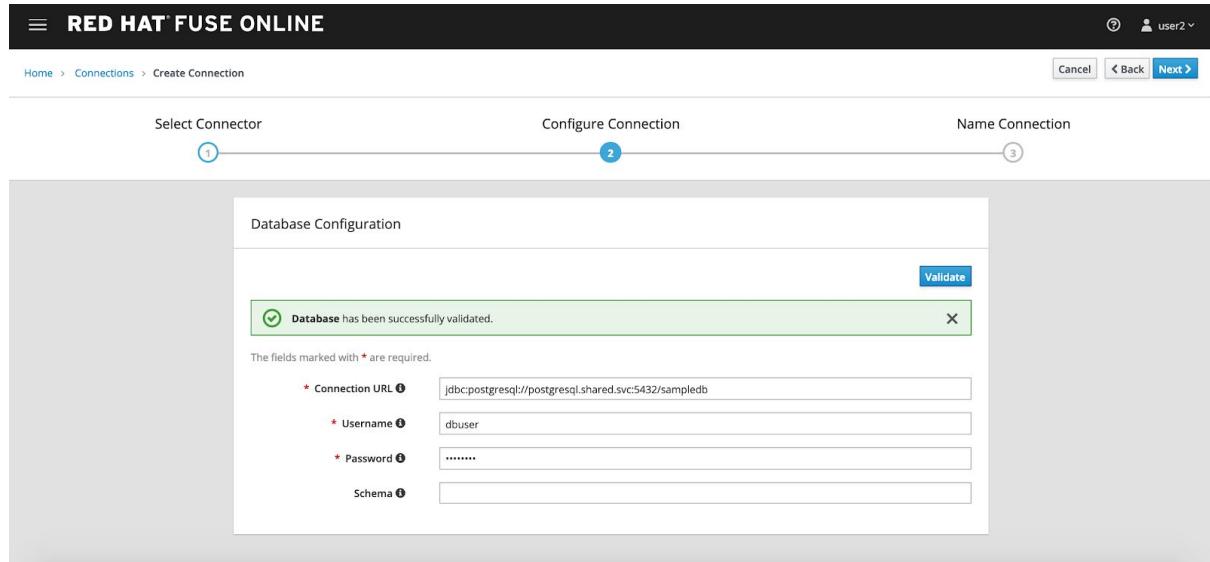
Let's start by creating the connection to an existing DB that I previously created. It is a DBaaS built on PostgreSQL and hosted on this environment. You could be running this anywhere else, as long as you have a public direct connection to the DB to use.

Under “Connections” click on “Create Connection” and select the “Database” connector.



Use the following connection details and validate them by clicking on “Validate”:

Connection URL: jdbc:postgresql://postgresql.shared.svc:5432/sampledb
Username: dbuser
Password: password



Click “Next” and provide “open-data-banks” as the “Connection Name”. Then click on “Create” to create the new connection.

The screenshot shows the 'Create Connection' wizard in progress. The top navigation bar indicates 'RED HAT FUSE ONLINE', the user 'user2', and the current step 'Configure Connection'. The main form is titled 'Add Connection Details' and contains fields for 'Connection Name' (set to 'open-data-banks') and 'Description'. A note at the top states: 'The fields marked with * are required.' There are three numbered steps at the top: 1. Select Connector, 2. Configure Connection, and 3. Name Connection.

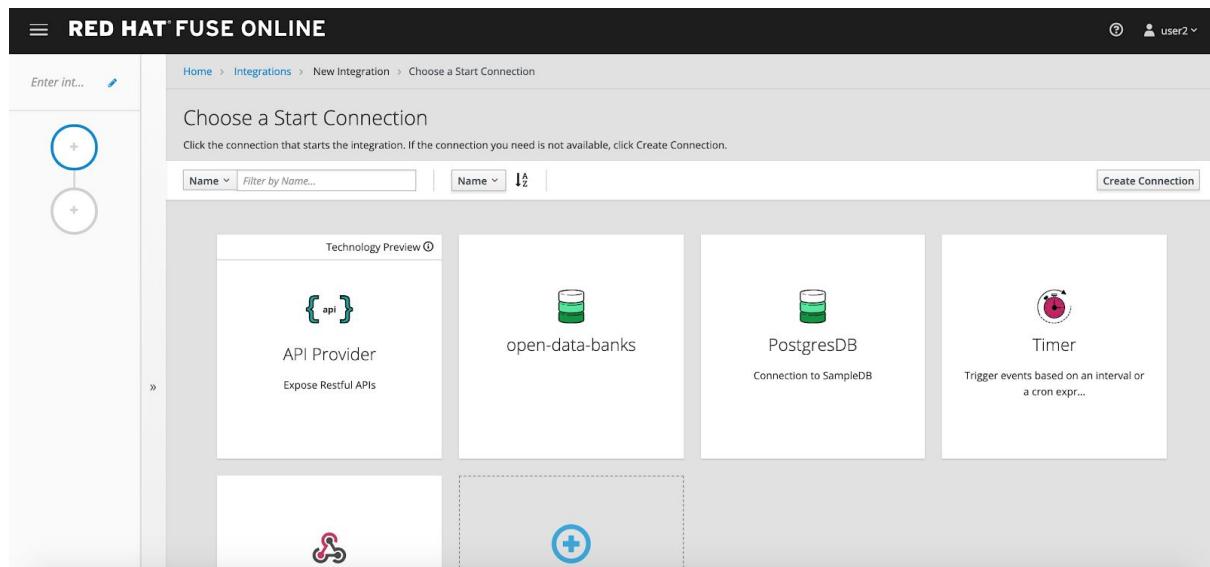
We can now use this connection as an integration starting, middle or finishing point.

The screenshot shows the 'Connections' page in the Red Hat Fuse Online interface. The left sidebar includes links for Home, Integrations, Connections (which is selected), Customizations, and Settings. The main area displays two connections: 'open-data-banks' and 'PostgresDB'. Each connection is represented by a icon and a label. A 'Create Connection' button is located in the top right corner. A 'Trello' button is visible in the bottom right corner of the main area.

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.

Under “Integrations” click on “Create Integration”

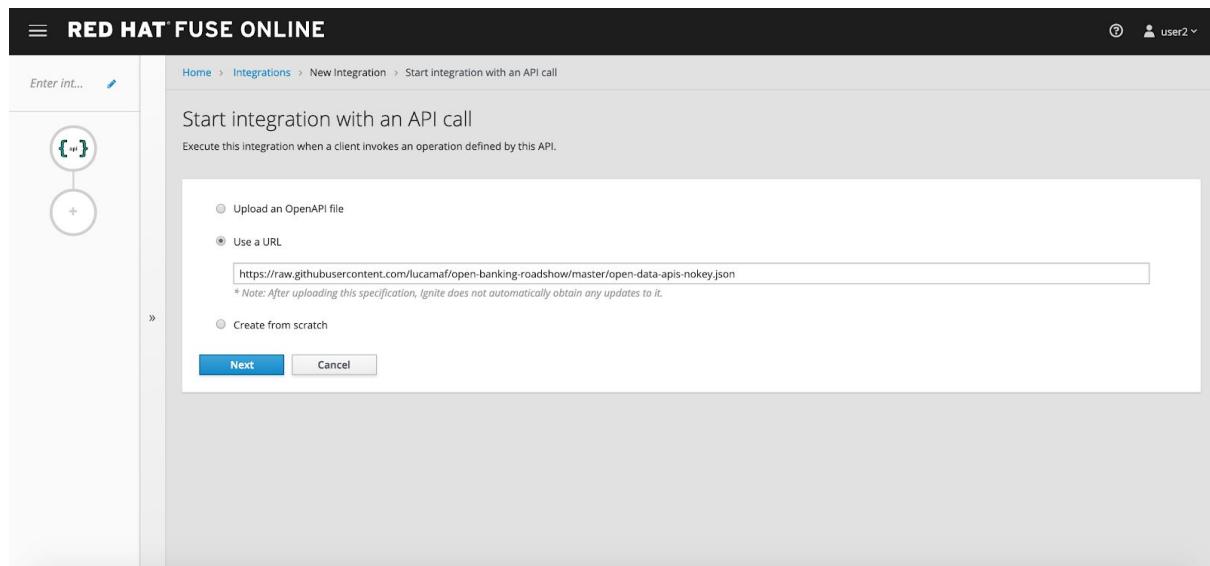
We will start by exposing a REST endpoint that will then get mapped to the backend datasource.



Use the “**API Provider**” connector with the following OpenAPI file (under “Use a URL”):

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

Then click on “Next”.



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 click on “Review/Edit”.

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

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

Click on “Back” and then on “Next”. Provide “open-banking” as the “Integration Name”

Click on “Save and continue”.

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

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

Click on get list of banks

Add to Integration

Now you can add additional connections as well as steps to your integration.

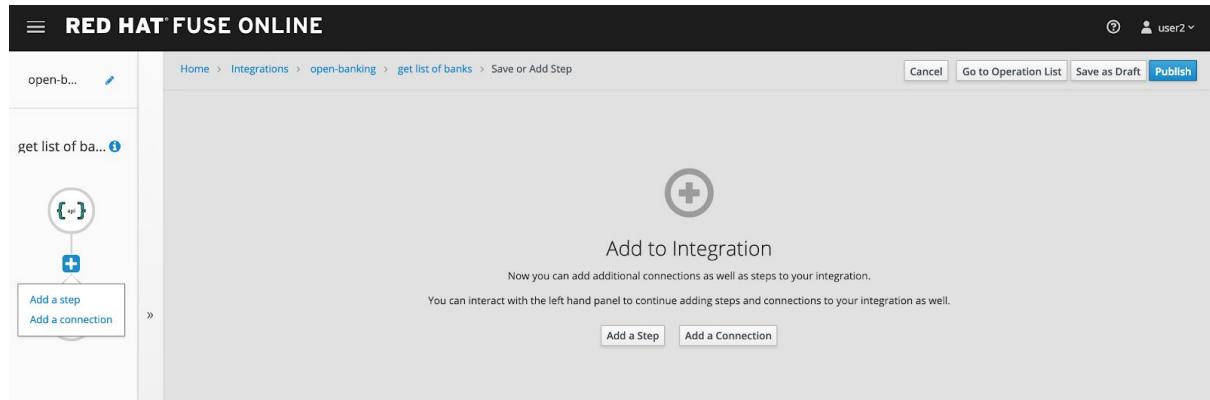
You can interact with the left hand panel to continue adding steps and connections to your integration as well.

Add a Step Add a Connection

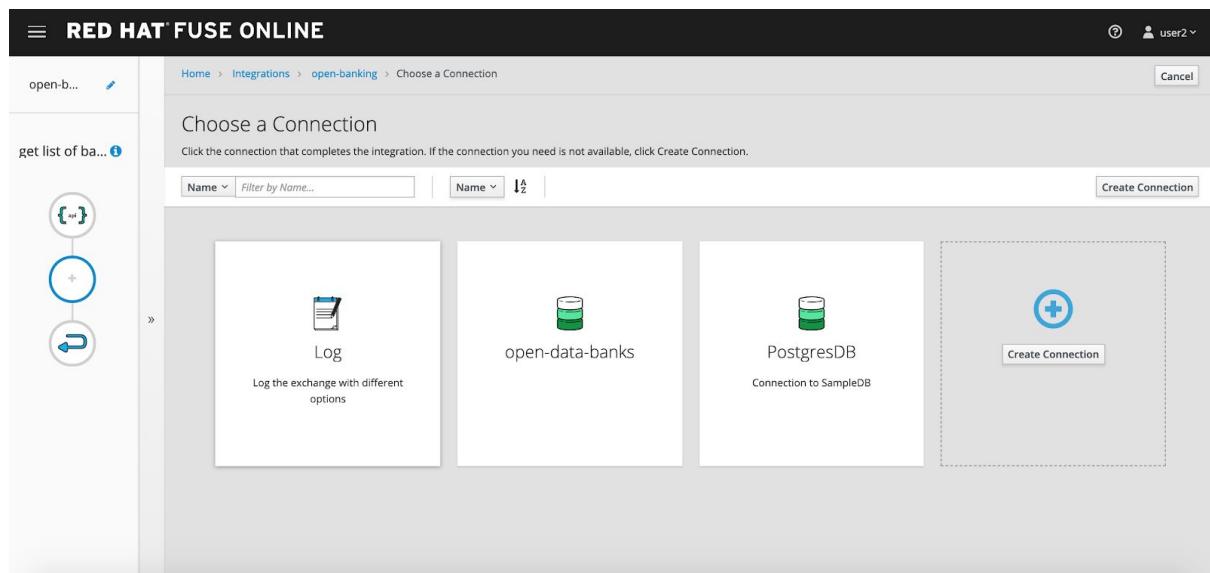
We now have a dumb pipe which connects an endpoint to receive user requests and return always 200 (all OK) - the return blue arrow symbol. Not very useful integration but a starter!

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

Hover over the blue + symbol and click on “Add a connection”.



Click on the previously configured data source “open-data-banks”.



And click on “Invoke SQL” to invoke a simple SQL statement to return the data from a single table.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a connection named "open-b...". The main area is titled "open-data-banks - Choose an Action" and says "Choose an action for the selected connection.". There are two options: "Invoke SQL" (selected) and "Invoke stored procedure".

Add the statement “**select * from banks**” into the “SQL statement” field.

And click on “Done”.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a connection named "open-b...". The main area is titled "open-data-banks" and "Configure Invoke SQL". It says "Enter a SQL statement that starts with INSERT, SELECT, UPDATE or DELETE.". The "SQL statement" field contains "select * from banks".

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

Click on “Add a step” underneath the second + symbol.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a tree view showing a project named "open-b..." and a step named "get list of ba...". On the right, the main panel displays the "Add to Integration" screen. It features a large central area with a plus sign icon and the text "Add to Integration". Below this, it says "Now you can add additional connections as well as steps to your integration." and "You can interact with the left hand panel to continue adding steps and connections to your integration as well." At the bottom of this area are two buttons: "Add a Step" and "Add a Connection". At the very bottom of the page, there are navigation links: "Home", "Integrations", "open-banking", "get list of banks", "Save or Add Step", "Cancel", "Go to Operation List", "Save as Draft", and "Publish".

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

Select the “Log” step.

The screenshot shows the "Choose a Step" screen in Red Hat Fuse Online. On the left, the sidebar shows the same project and step structure as the previous screenshot. The main panel has a title "Choose a Step" and a subtitle "A step specifies an operation on the data in the integration. Click one to add it." Below this is a search bar with "Name" and "Filter by Name..." fields. A list of step options is displayed:

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

At the top right of the main panel, there is a "Cancel" button. The bottom of the page includes the standard navigation links: "Home", "Integrations", "open-banking", "get list of banks", "Save or Add Step", "Cancel", "Go to Operation List", "Save as Draft", and "Publish".

Select the “Message Body” option and click on “Done”.

The screenshot shows the Red Hat Fuse Online interface. On the left, there's a sidebar with a tree view of integration steps: 'open-b...' (with a pencil icon), 'get list of ba...', and three other nodes represented by icons (a person, a green circle, and a blue circle). The main area is titled 'Configure Log' and displays the sub-tutorial 'Fill out the fields associated with the selected step.' It contains a configuration dialog with the following fields:

- Message Context
- Message Body

A 'Custom Text' input field is present, and at the bottom are 'Cancel' and 'Done' buttons.

We are now ready to deploy and expose this integration in our platform, to use it.

Hit “Publish”.

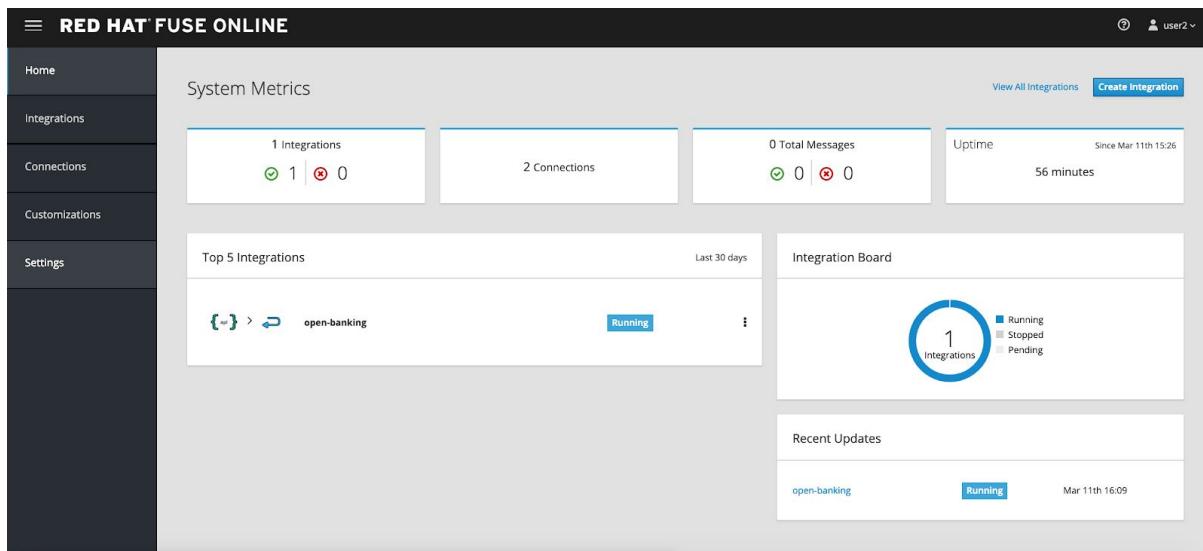
The screenshot shows the Red Hat Fuse Online interface on the 'Integration Summary' page for the 'open-banking' integration. The sidebar on the left is visible with the 'Integrations' item selected. The main content area shows the integration details:

- open-banking** (with a pencil icon) - Status: Assembling (1 / 4)
- Details** tab is active, showing 'API Provider' and '1 Flow'.
- Activity** and **Metrics** tabs are also present.
- A note says 'No description set...' with a pencil icon.
- At the bottom, there are buttons for **Draft**, **Publish** (highlighted in blue), and **Edit**.
- Below that, a **History:** section shows 'Version: 1'.

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

This will take about 5 to 10 minutes.

Go back to “Home” to view the recently deployed integration.



3SCALE API MANAGEMENT PLATFORM

Now let's manage our newly created integration endpoint in 3scale.

Open 3scale by clicking [Red Hat 3scale API Management Platform](#) link in the tutorial dashboard of the [Red Hat Solution Explorer](#).

The screenshot shows the Red Hat Solution Explorer dashboard. It features a 'Start with a walkthrough' section with three items: 'Publishing walkthroughs to a cluster', 'Integrating message-oriented middleware with a RESTful API using AMQ', and 'Integrating message-oriented middleware with a RESTful API using AMQ Online'. Each item has a 'Get Started' button and a duration (25 min, 15 min, 21 min). To the right is a '5 walkthroughs' summary. On the far right is an 'Applications' section listing 10 applications: Red Hat OpenShift (Ready for use), Red Hat 3scale API Management Platform (Ready for use), Eclipse Che (community, Ready for use), Red Hat Fuse Online (Ready for use), Red Hat Developer Launcher (community, Ready for use), 3scale Developer Dashboard (custom, Ready for use), and 3scale Admin Dashboard (custom, Ready for use).

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's a header with the 3scale logo, 'RED HAT 3SCALE API MANAGEMENT', and a 'Dashboard' dropdown. Below the header, there are three main sections:

- AUDIENCE:** Shows 1 Signup in the last 30 days.
- POTENTIAL UPGRADES:** Shows 0 accounts that hit usage limits today. It includes instructions to add usage limits to Application Plans and enable Web Alerts for Admins.
- MESSAGES:** Shows 0 messages. A section for 'TODAY' shows 'Inbox Zero'.

In the bottom section, there's a 'NEW API' button. Below it, under 'APIS', there's a section for 'API' with 0 hits in the last 30 days. This section includes tabs for 'OVERVIEW', 'ANALYTICS', 'APPLICATION', 'ACTIVEDOC', and 'INTEGRATE THIS API'. It also shows 'Top Applications' with 0 apps having high traffic in the last 30 days, with instructions to make test calls from the integration page.

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

In the dashboard 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.

RED HAT 3SCALE API MANAGEMENT [Dashboard](#)

New API

Define manually Import from OpenShift

SERVICE

Name

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

Description

[Add API](#)

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

Authorize Access

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.

RED HAT 3SCALE API MANAGEMENT [Dashboard](#)

New API

Define manually Import from OpenShift

SERVICE

Namespace

Name

[Create Service](#)

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

The 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 sections for AUDIENCE (1 Signups last 30 days), ACCOUNT (1 ACCOUNT today), APPLICATION (1 APPLICATION), BILLING, DEVELOPER PORTAL (0 DRAFTS), and MESSAGES (0 MESSAGES). A message box displays: "The service will be imported shortly. You will receive a notification when it is done." Below this, there are two news items: "TODAY No news, good news." and "BEFORE TODAY The sound of silence." In the center, there's a section titled "Potential Upgrades" which says "Accounts that hit their Usage Limits in the last 30 days". It includes instructions to add usage limits to Application Plans and enable Web Alerts for Admins. At the bottom, there are links for "All APIs" and "NEW API".

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.

In the menu click on the newly created API “i-open-banking”.

The screenshot shows the Red Hat 3SCALE API Management dashboard with a sidebar menu. The menu items are: Dashboard, Audience, Type the API name, API, and i-open-banking. The i-open-banking item is highlighted, indicating it is selected.

Click on “Integration” and “Configuration”.

The screenshot shows the 'Configuration' section of the Red Hat 3Scale API Management interface. It displays the following details:

- Integration settings:** Deployment Option: APICAST; Authentication: API Key (user_key).
- APICAST Configuration:**
 - Private Base URL: http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080
 - Mapping rules: / => hita
 - Credential Location: query
 - Secret Token: Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e
- Environments:**
 - Staging Environment:** https://fuse-d6d2a761-43b5-11e9-9d29-0a580a010007-i-open-banking-3scale-apicast-staging.apps.openbanking-ea09.openshiftworkshop.com:443 v. 1. Configuration history. Promote v. 1 to Production.
 - Production Environment:** no configuration has been saved for the production environment yet.

Click on the link “edit APICAST configuration”.

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' section of the Red Hat 3Scale API Management interface. It displays the following details:

- API:** Private Base URL: http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080. Use Echo API.
- API GATEWAY:**
 - Staging Public Base URL:** https://fuse-d6d2a761-43b5-11e9-9d29-0a580a010007-i-open-banking-3scale-apicast-staging.apps.i...
 - Production Public Base URL:** https://fuse-d6d2a761-43b5-11e9-9d29-0a580a010007-i-open-banking-3scale-apicast-production.ap...

We have the section where we map the Backend API (or in this case Integration service) as the Private Base URL.

And we define two URLs where we expose the managed API on the staging fist and production gateways.

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.GUID.openshiftworkshop.com:443`

where you replace **userX** with your user ID and the **GUID** with the provided value from your instructor under “Staging Public Base URL” and “Production Public Base URL”.

Integration

Configure & test immediately in the staging environment [documentation](#)

The screenshot shows the 3scale API configuration interface. It has two main sections: 'API' and 'API GATEWAY'.
API Section:
- **Private Base URL:** A text input field containing `http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080`. To its right is a blue link labeled 'Use Echo API'.
- Below the input field is a note: 'Private address of your API that will be called by the API gateway. For end-to-end encryption your private base URL scheme should be https.'
API GATEWAY Section:
- **Staging Public Base URL:** A text input field containing `https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443`. To its right is a blue link labeled 'Use Default URL'.
- Below the input field is a note: 'Public address of your API gateway in the staging environment. Make sure to add the correct route'.
- **Production Public Base URL:** A text input field containing `https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443`. To its right is a blue link labeled 'Use Default URL'.
- Below the input field is a note: 'Public address of your API gateway in the production environment. Make sure to add the correct route'.

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).

Under “Mapping Rules” click the green edit symbol.

- Under “Verb” select GET
- Under “Pattern” enter `/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).
- Under “Metrics or Method” leave hits

The screenshot shows the 'MAPPING RULES' section of the 3scale API gateway configuration. It displays a single mapping rule with the following details:

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

At the bottom right of the rule card, there are edit and delete icons, and a green button labeled 'Add Mapping Rule'.

We can now check and change the configuration of authentication settings under the menu item “Authentication Settings”.

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

The screenshot shows the 'AUTHENTICATION SETTINGS' configuration page. It includes the following sections:

- Host Header:** An empty input field with a placeholder: "Lets you define a custom `Host` request header. This is needed if your API backend only accepts traffic from a specific host."
- Secret Token:** An input field containing "Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e". A detailed description below it states: "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:** A radio button group where "As HTTP Headers" is selected (indicated by a blue circle).
- Auth user key:** An input field containing "key".

Under “Policies” you can add existing policies executed against API calls at the gateway level. Click on “Add Policy” to view the policies. For now we are not going to add new policies though, so click “Cancel” after that.

POLICIES

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 o...
Anonymous access builtin - Provides default credentials for unauthenticated requests.
URL rewriting with captures builtin - Captures arguments in a URL and rewrites the URL using t...
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.
3scale Referrer builtin - Sends the 'Referer' to 3scale backend so it can be validated.

Update the “API test GET request” field with the same pattern you are mapping above (excluding the \$):

/open-data/banks

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:

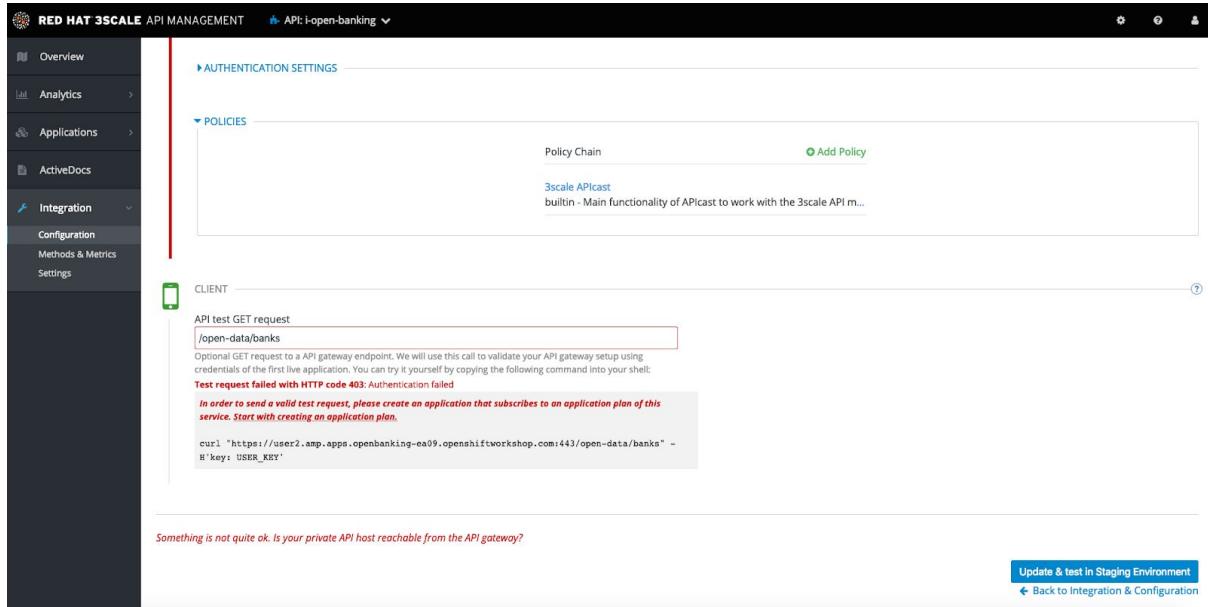
In order to send a valid test request, please create an application that subscribes to an application plan of this service. Start with creating an application plan.

```
curl "https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443/open-data/banks" -H
'key: USER_KEY'
```

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

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

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



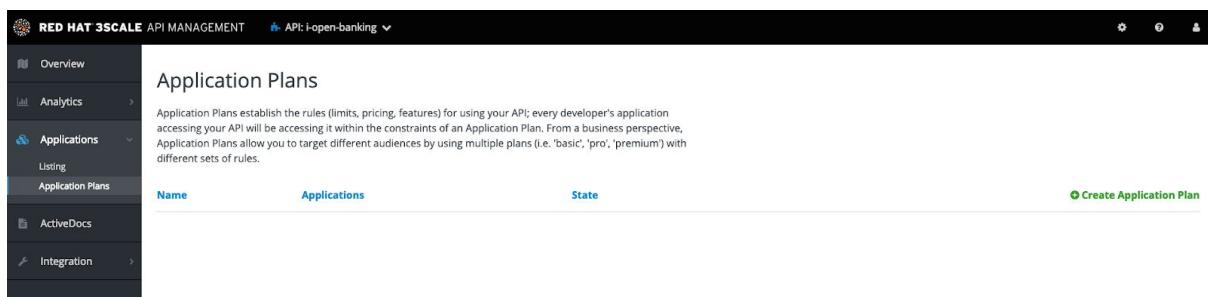
The screenshot shows the Red Hat 3Scale API Management interface. On the left, there's a navigation sidebar with options like Overview, Analytics, Applications (which is expanded to show 'Listing' and 'Application Plans'), ActiveDocs, Integration, Configuration (Methods & Metrics, Settings), and Help. The main content area is titled 'API: i-open-banking'. It has sections for 'AUTHENTICATION SETTINGS' and 'POLICIES'. Under 'POLICIES', there's a 'Policy Chain' section with a link to 'Add Policy' and a note about '3scale APICAST'. Below that is a 'CLIENT' section with a mobile phone icon. It contains a 'Test request' input field with the URL '/open-data/banks'. A red box highlights the error message: 'Test request failed with HTTP code 403: Authentication failed'. Below the error message, there's a note: '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.' At the bottom of the page, there's a note: 'Something is not quite ok. Is your private API host reachable from the API gateway?' and two buttons: 'Update & test in Staging Environment' and 'Back to Integration & Configuration'.

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

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

Within the red error message a link is generated "*Start with creating an application plan*". Click this link and you will be redirected to the Application Plan overview.

Click on the green link "Create Application Plan".



The screenshot shows the 'Application Plans' overview page. The left sidebar includes 'Overview', 'Analytics', 'Applications' (with 'Listing' and 'Application Plans' sub-options), 'ActiveDocs', and 'Integration'. The main content area has a heading 'Application Plans' and a descriptive text: '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 table with columns 'Name', 'Applications', 'State', and a green 'Create Application Plan' button. A red box highlights the 'Create Application Plan' button.

Fill out the Name field ("Basic") and System Name field ("basic") form and click the blue button to submit the form.

You can safely ignore for now the monetization options.

Create Application Plan

Name

System name

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

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

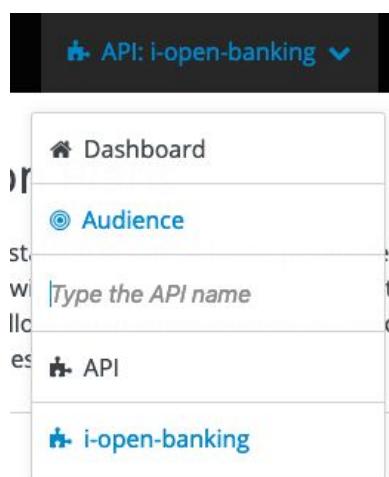
Trial Period (days)

Setup fee
 USD

Cost per month
 USD

We see that we have one 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 (by clicking on “Publish”) so that it is usable and visible on the Developer portal.

In the menu click on “Audience”.



From here we can see how we can - as Provider - approve or deny Developers’ Accounts registrations.

RED HAT 3SCALE API MANAGEMENT Audience

Accounts

Group/Org.	Admin	Signup Date	Apps	State	Create
Developer	John Doe	6 Mar, 2019	1	Approved	Create

[Export all Accounts](#)

Let's click on the default "Developer" Account.

RED HAT 3SCALE API MANAGEMENT Audience

Account: Developer [Edit](#)

Organization/Group Name	Developer	Send message
Administrator	John Doe (admin+test@example.com)	
Signed up on	March 06, 2019 22:43	
Status	Approved	

Billing Status

Monthly billing is enabled. [Disable](#)

Account Plan: Default

[Convert to a Custom Plan](#)

Application

Name	Developer's App
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 "1 Application" in the top level navigation and then on "Create Application".

RED HAT 3SCALE API MANAGEMENT Audience

Applications

Name	State	Service	Plan	Paid?	Created At	Traffic On	Create Application
Developer's App	live	API	Basic	free	March 06, 2019		Create Application

Here we can now subscribe the application to the Application plan "Basic" we created on our new Service "i-open-banking" from the drop down field available.

The screenshot shows the Red Hat 3SCALE API Management interface. The top navigation bar includes the logo, 'RED HAT 3SCALE API MANAGEMENT', and a 'Audience' dropdown. The left sidebar has a dark theme with white text and icons, listing 'Accounts' (selected), 'Applications' (with a sub-item 'i-open-banking'), 'Billing', 'Developer Portal', and 'Messages'. The main content area shows the URL 'Account 'Developer' > 1 Application | 1 User | 0 Invitations | 0 Group Memberships | 0 Invoices | 1 Service Subscription'. Below this is the title 'New Application'. A 'Basic' application plan is selected under 'Application plan'. Under 'API', 'Basic' is chosen from 'Unlimited' and 'i-open-banking'. The 'i-open-banking' section is highlighted with a blue background. A 'Basic' service plan is selected under 'Service plan'. The 'Name' field contains 'Open Banking App' and the 'Description' field also contains 'Open Banking App'. At the bottom right is a blue 'Create Application' button.

Let's fill in the rest of the fields with the name "Open Banking App" and click the blue button "Create Application".

The screenshot shows the 'New Application' form. The URL at the top is 'Account 'Developer' > 1 Application | 1 User | 0 Invitations | 0 Group Memberships | 0 Invoices | 1 Service Subscription'. The form fields are: 'Application plan' (Basic), 'Service plan' (Default), 'Name' (Open Banking App), 'Description' (Open Banking App). At the bottom right is a blue 'Create Application' button.

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.

In the menu on the overview window click on "Integration" and "Configuration".

The screenshot shows the Red Hat 3SCALE API Management interface. The left sidebar has sections for Overview, Analytics, Applications (Listing, Application Plans), ActiveDocs, Integration (Configuration, Methods & Metrics, Settings). The main content area shows the 'Open Banking App' application details. It includes fields for Account (Developer: Open Banking App), Description (Open Banking App), Service (i-open-banking), and State (Live, Suspend button). A 'Regenerate' button is available for the User Key (90eeffbd8c833b7e52c6735d2d111699). To the right, there's a box for 'Application Plan: Basic' with a 'Convert to a Custom Plan' link. Below the main application details are sections for 'API Credentials' (User Key) and 'Usage in last 30 Days' (0 hits). At the bottom, there's a 'Current Utilization' section stating 'This is an unmetered application, there are no limits defined'.

Click on “edit APIcast configuration”.

The screenshot shows the 'APIcast Configuration' page. It contains fields for Private Base URL (http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080), Mapping rules (/open-data/banks\$ => hits), Credential Location (headers), and Secret Token (Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e). A red box highlights the 'edit APIcast configuration' link at the top right of the form.

Scroll down to the bottom of the page and click on “Update & test in Staging Environment” again.

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

API: l-open-banking

AUTHENTICATION SETTINGS

POLICIES

Policy Chain: 3scale APICAST
builtin - Main functionality of APICAST to work with the 3scale API m...

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:
curl "https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443/open-data/banks" -
H 'key: 90eefb0d8c833b7e532c6735d2d111699'

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. 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.

PROVIDER NAME DOCUMENTATION PLANS SIGN IN

Draft | Published

Echo API

REGISTER
Register to the developer portal to use the Echo API

Get your API key
Use your API key to authenticate and report the calls you make

Create your app
Start coding and create awesome applications with the Echo API

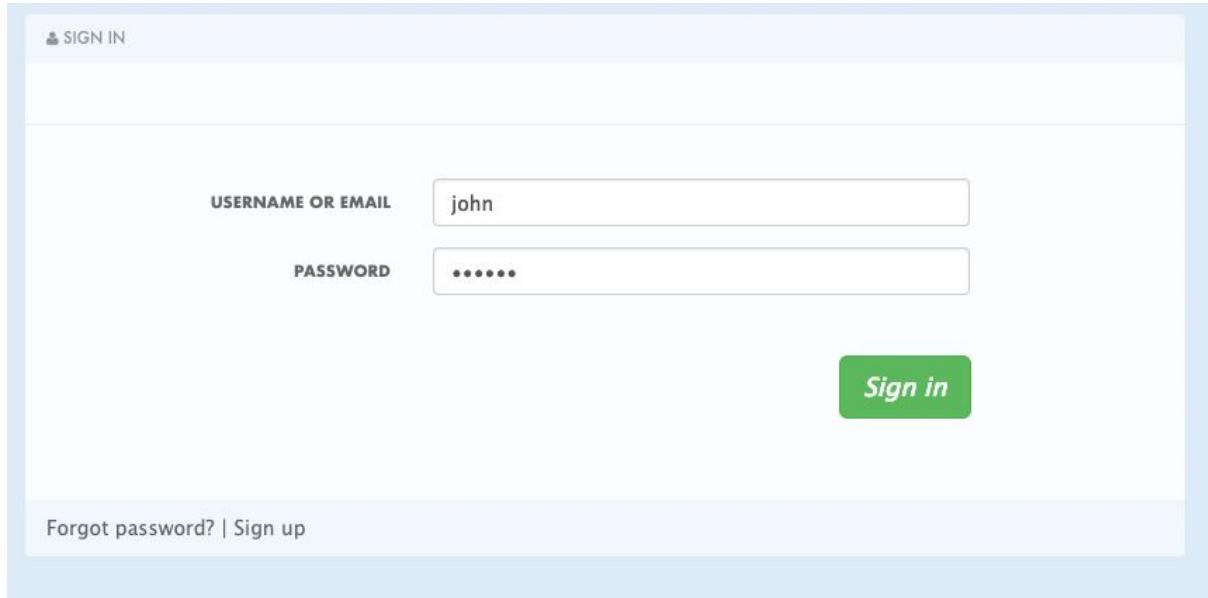
PICK YOUR PLAN

BASIC	UNLIMITED
Features ✓ Unlimited Greetings	Features ✓ Unlimited Greetings ✓ 24/7 support ✓ Unlimited calls
Limits No limits	Limits No limits
Signup to plan Basic	Signup to plan Unlimited

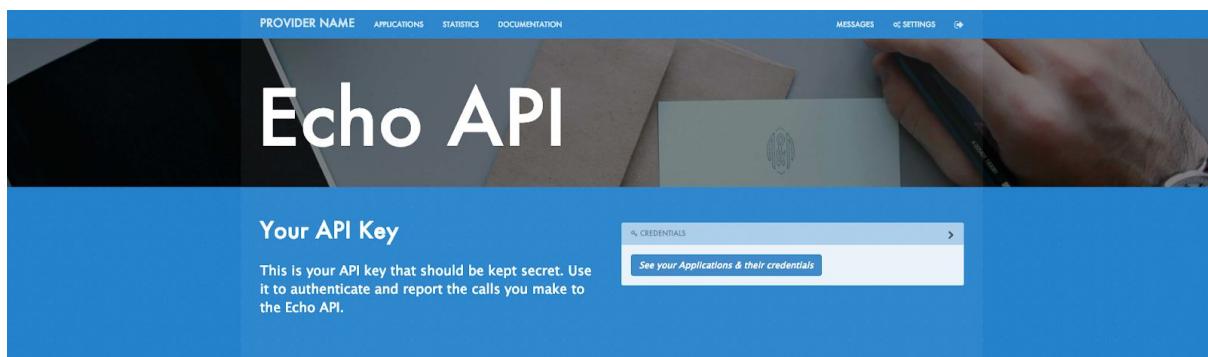
Let's sign in with the default user credentials provided in the sidebar. This is the default developer user, created for the default developer account.

The username is john

The password is 123456



We are now logged in the developer's dashboard.



Let's see the applications by clicking on "See your applications & their credentials".

NAME	SERVICE	CREDENTIALS	STATE
Open Banking App	i-open-banking	90eefbbd8c833b7e52c6735d2d111699	live
Developer's App	API	0d403c2d4d0afd5fa92e3dfb6006cc44	live

[Create new application](#)

Click on “Open Banking App”.

Name: Open Banking App
Description: Open Banking App
Plan: Basic > [Review/Change](#)
Status: Live
User Key: **90eefbbd8c833b7e52c6735d2d111699**
Add this as a `user_key` parameter to your API calls to report and authenticate.
[Regenerate](#)

You can now use the credentials that were associated with the application and test the protected service.

Let's move to the online API testing tool, <https://apitester.com>

Use the URL for your API gateway:

`https://userX.amp.apps.GUID.openshiftworkshop.com:443/open-data/banks`

where you replace **userX** with your user ID and the **GUID** with the provided value from your instructor.

Also remember the the key Header and the associated value.

The easiest way to test APIs

Make HTTP requests, extract values from the responses, assert the values are correct, reuse variables across steps, or inject custom logic using JavaScript. Build a single test to quickly validate an endpoint or build a whole suite of tests to run at the click of a button.

[View example](#)[Visit FAQ](#)[X Don't show this again.](#)

Build your test

[View example](#)

Click to add or remove steps

[Collapse / Expand](#)

Request	Step Name
GET	https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443/open-data/banks
1	Headers + Add Request Header <input type="text" value="key"/> <input type="text" value="90eefbbd8c833b7e52c"/>
+ Add Step	

[Test](#)[Save to Account](#)[Share Test Config](#)

Click on “Test”.

As we can see we succeed with 200 OK !

 PASS

N. Virginia
Seconds elapsed: 2

Results 1.10s

Viewing a Request Step 1 < >

Message

[Step 1] GET https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443/open-data/banks passed

Connection Information

Request

Request Headers

```
GET /open-data/banks HTTP/1.1
Host: user2.amp.apps.openbanking-ea09.openshiftworkshop.com
Accept: */*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
key: 90eefbbd8c833b7e52c6735d2d111699
```

Response

Response Headers

```
HTTP/1.1 200 OK
Server: openresty/1.13.6.1
Date: Tue, 12 Mar 2019 05:33:12 GMT
Transfer-Encoding: chunked
X-Application-Context: application
Set-Cookie: Sedba771e9ee85040f19fe8af69fd058=2573d43e5703f514b749872ff136af35; path=/; HttpOnly
Cache-control: private
```

Response Body

```
{"short_name": "387832 5350", "id": 1, "address": "425-4002 Urna, Road", "long_name": "Molestie Inc."}
```



Practical Lab - Part 2

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.

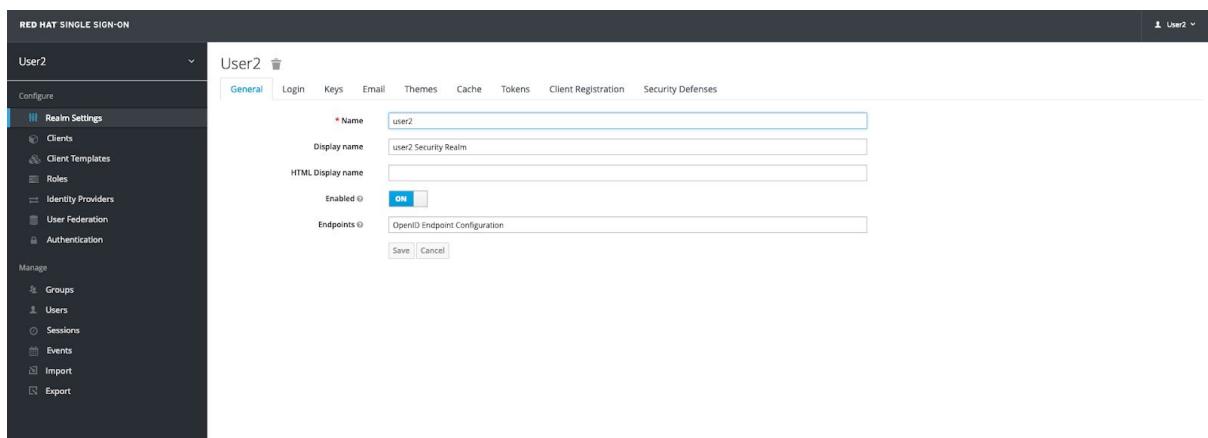
RED HAT SINGLE SIGN ON AND 3SCALE OIDC

Let's start with RH SSO main dashboard

It has the following URL:

**[https://sso-sso.apps.GUID.openshiftworkshop.com/auth/admin/userX/console
#/realms/userX](https://sso-sso.apps.GUID.openshiftworkshop.com/auth/admin/userX/console/#/realms/userX)**

where you replace the GUID with the provided GUID and userX with your user ID.



The screenshot shows the RHSSO interface. On the left, there is a sidebar with a dark background containing navigation links such as 'Configure', 'Realm Settings' (which is currently selected), 'Clients', 'Client Templates', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. Below this is a 'Manage' section with links for 'Groups', 'Users', 'Sessions', 'Events', 'Import', and 'Export'. The main content area has a light gray background and displays the 'User2' realm settings. At the top of this area, there are tabs for 'General', 'Login', 'Keys', 'Email', 'Themes', 'Cache', 'Tokens', 'Client Registration', and 'Security Defenses'. The 'General' tab is active. The 'General' settings include fields for 'Name' (set to 'user2'), 'Display name' (set to 'user2 Security Realm'), 'HTML Display name' (empty), 'Enabled' (set to 'ON'), and 'Endpoints' (set to 'OpenID Endpoint Configuration'). At the bottom of the form are 'Save' and 'Cancel' buttons. In the top right corner of the main window, there is a small dropdown menu labeled 'User2'.

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.

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

Client ID	Enabled	Base URL	Actions		
3scale-admin	True	Not defined	Edit	Export	Delete
account	True	/auth/realm/User2/account	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	/auth/admin/user2/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.

Click on “Users” and the link “View all users”.

ID	Username	Email	Last Name	First Name	Actions	Unlock users	Add user
e7a74fc0-0a47-4906-b9f1-3f3ea0f41...	user2	user2@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.

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

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

Normally to make sure an API web application authenticates with RH SSO, we would need to manually create the application on both platforms. With this feature, we let 3scale sync the applications to RH SSO, as well as obviously authenticating our API calls.

Let's create a special type of such Client in RH SSO.

Under “Client” click on on “Create”.

Under Client ID provide the name “sync-app” and click on “Save”.

Now configure the other details required to let it communicate with 3scale.

- Under “Access Type” select “confidential”
- Set “Service Accounts Enabled” to “ON”

- Set “Standard Flow Enabled” to “OFF”
- Set “Direct Access Grants Enabled” to “OFF”

Click on “Save”.

Move to the newly activated “Service Account Roles” tab.

Pick “realm-management” under “Client Roles”

Add “manage-clients” to the assigned roles in this window and by clicking “add selected”.

Finally, click on the “Credentials” tab and note down the “Secret”.

Clients > sync-app

Sync-app

Settings Credentials Roles Mappers Scope Revocation Sessions Offline Access Clustering Installation Service Account Roles

Client Authenticator: Client Id and Secret

Secret: b517a14b-57be-458c-be91-cf53ab482a84 | Regenerate Secret

Registration access token: [empty] | Regenerate registration access token

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.

RED HAT 3SCALE API MANAGEMENT API: i-open-banking

Overview Analytics Applications ActiveDocs Integration Configuration Methods & Metrics Settings

Configuration

Integration settings

Deployment Option: APICAST

Authentication: API Key (user_key)

edit integration settings

APICAST Configuration

Private Base URL: http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080

Mapping rules: /open-data/banks\$ => hits

Credential Location: headers

Secret Token: Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e

edit APICAST configuration

Environments

Staging Environment: https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443 v. 3

Configuration history

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.

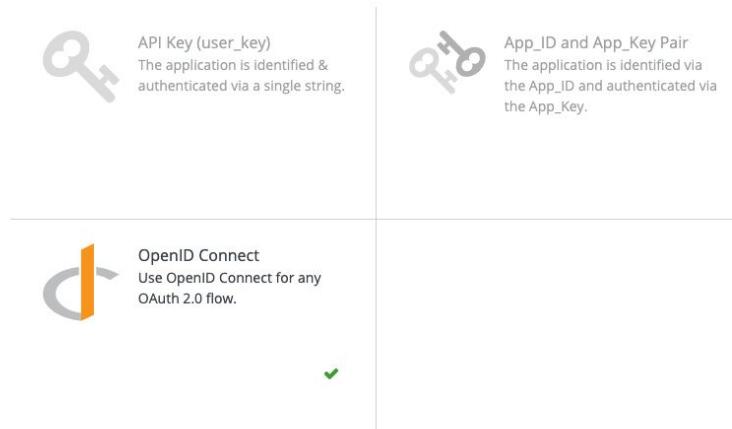
Click on “Edit integration settings”

Under “Authentication” we are going to change the selected value to “OpenID Connect” and click “Update service”.

AUTHENTICATION

Authentication

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



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.

Click on “edit APIcast configuration”.

The screenshot shows the 'Configuration' page in the Red Hat 3scale API Management interface. On the left, there's a sidebar with navigation links like Overview, Analytics, Applications, ActiveDocs, Integration, Configuration (which is selected), Methods & Metrics, and Settings. The main area has a header 'Configuration' and two sections: 'Integration settings' and 'APIcast Configuration'. Under 'Integration settings', 'Deployment Option' is set to 'APICAST' and 'Authentication' is set to 'OpenID Connect'. Under 'APIcast Configuration', 'Private Base URL' is 'http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080', 'Mapping rules' is '/open-data/banks\$ => hits', 'Credential Location' is 'headers', and 'Secret Token' is 'Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e'. There are 'edit' buttons for both sections.

As we see we have a dedicated field for this purpose now under “Authentication Settings” (**OpenID Connect Issuer**).

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

```
http://client-id:client-secret@idp-public-endpoint
```

where

client-id:

sync-app

client secret:

<defined-during-configuration>

idp-public-endpoint:

sso-*http-sso.apps.GUID.openshiftworkshop.com/auth/realm*/*<userX>*

(where you replace GUID with the GUID from the instructor and your user id)

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.

Configuration

The screenshot shows the configuration interface for an APIcast integration. It includes sections for 'Integration settings' (Deployment Option: APICAST, Authentication: OpenID Connect), 'APICAST Configuration' (Private Base URL: http://i-open-banking.fuse-d6d2a761-43b5-11e9-9d29-0a580a010007.svc.cluster.local:8080, Mapping rules: /open-data/banks\$ => hits, Credential Location: headers, Secret Token: Shared_secret_sent_from_proxy_to_API_backend_d1aeed05897d8e7e), and 'Environments' (Staging Environment: https://user2.amp.apps.openbanking-ea09.openshiftworkshop.com:443). A blue 'Promote v. 5 to Production' button is visible in the bottom right corner of the environments section.

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

The screenshot shows a web application interface for managing an Open Banking application. At the top, there's a navigation bar with tabs: PROVIDER NAME, APPLICATIONS, STATISTICS, and DOCUMENTATION. Below the navigation, a header bar has a back arrow labeled 'Applications' and a 'Edit Open Banking App' button.

The main content area displays the following application details:

- Name:** Open Banking App
- Description:** Open Banking App
- Plan:** Basic > [Review/Change](#)
- Status:** Live
- Client ID:** **44835500**
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 URL:

<https://openidconnect.net/callback>

Client Secret

This is the Client Secret used to authenticate requests.

[Create secret](#)

Redirect URL This is your Redirect URL for OAuth.

REDIRECT URL

[Submit](#)

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

The screenshot shows the Red Hat 3Scale API Management interface. On the left, there is a sidebar with navigation links: Overview, Analytics, Applications (with Listing and Application Plans), ActiveDocs, and Integration. The main content area displays details for an API named "i-open-banking".

API Details:

Account	Developer
Description	Open Banking App
Service	i-open-banking
State	Live
Suspend	

API Credentials:

Client ID	036d8064
Client Secret	d4c26ac6a9ec2d2e1980d004be1ca56e
Regenerate	
Redirect URL	https://openidconnect.net/callback
Edit	

Usage in last 30 Days:

A line chart showing usage over the last 30 days. A single blue spike is visible, indicating 2 hits.

Hits
2 hits

Check the “Clients” section and ensure that the automatically created client ID is now present. In the screenshot below it is “036d8064”.

Clients

Client ID	Enabled	Base URL	Actions		
036d8064	True	Not defined	Edit	Export	Delete
3scale-admin	True	Not defined	Edit	Export	Delete
account	True	/auth/realm/user1/account	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	/auth/admin/user1/console/index.html	Edit	Export	Delete
sync-app	True	Not defined	Edit	Export	Delete

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 by clicking on “Configuration”:

- Under “Server Template” select “Custpm”
- Under “Discovery Document URL” type the following URL:
`http://sso-http-sso.apps.GUID.openshiftworkshop.com/auth/realm/userX/.well-known/openid-configuration`
- Click “Use Discovery Document”
- Fill in the “OIDC Client Id” with the client id from RHSSO
- Fill in the “OIDC Client Secret” with the secret from RHSSO
- Under Scope type in “openid email”
- Select “Save”

OpenID Connect Configuration



Server Template	Custom	▲
Discovery Document URL	<input type="text" value="http://sso-http-sso.apps.openbanking-8005.openshiftworkshop.cc"/> USE DISCOVERY DOCUMENT	
Use a discovery document to populate your server urls		
Authorization Token Endpoint	<input type="text" value="http://sso-http-sso.apps.openbanking-8005.openshiftworkshop.com/auth/realms/user1/protocol/ope"/>	
Token Endpoint	<input type="text" value="http://sso-http-sso.apps.openbanking-8005.openshiftworkshop.com/auth/realms/user1/protocol/ope"/>	
Token Keys Endpoint	<input type="text" value="http://sso-http-sso.apps.openbanking-8005.openshiftworkshop.com/auth/realms/user1/protocol/ope"/>	
Remember to set https://openidconnect.net/callback as an allowed callback with your application!		
OIDC Client ID	<input type="text" value="036d8064"/>	
OIDC Client Secret	<input type="text" value="d4c26ac6a9ec2d2e1980d004be1ca56e"/>	
Scope	<input type="text" value="openid email"/>	
<input type="button" value="SAVE"/>		

Now you are on the start page with the prefilled Request document:

1

Redirect to OpenID Connect Server

Request

```
http://sso-http-sso.apps.openbanking-
8005.openshiftworkshop.com/auth/realms/user1/protocol/openi
connect/auth?

client_id=036d8064
&redirect_uri= https://openidconnect.net/callback
&scope=openid email
&response_type=code
&state=bb175a14238a4c69e72c5df6fc39ba9bac576124
```

START

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..VTYGpUa-
DVZHGXcukw_0hA.38CdI0wjX-
AvF02Ah0D3WpvJCbxfwpvWw0CosuX8maHzXjiGLrH-Y-
PZltQ_fn41yhWkKjB0Br37wGi20zxj0tG351Vou4V698VyEoeFkWMnb1E6src
x4imFVrbgf48DdRh4lBzXJ_VyCsXudyDY8tUGgyDXQz9P48w9R4m0vR2AXIXD
I1heVHxTiNnTpK2z8i1dqVeG87oKSQT9iEdxkRzfuzj8fpYqGaia37.8nxaId
```

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 http://sso-http-sso.apps.openbanking-
8005.openshiftworkshop.com/auth/realms/user1/protocol/openi
connect/token
grant_type=authorization_code
&client_id=036d8064
&client_secret=d4c26ac6a9ec2d2e1980d004be1ca56e
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VTYG
DVZHGXcukw_0hA.38CdI0wjX-
AvF02Ah0D3WpvJCbxfwpvWw0CosuX8maHzXjiGLrH-Y-
PZltQ_fn41yhWkKjB0Br37wGi20zxj0tG351Vou4V698VyEoeFkWMnb1E6s
x4imFVrbgf48DdRh4lBzXJ_VyCsXudyDY8tUGgyDXQz9P48w9R4m0vR2AXI
I1heVHxTiNnTpK2z8i1dqVeG87oKSQT9iEdxkRzfuzj8fpYqGaia37.8nxa
```

EXCHANGE

Hit “Exchange”

Request

```
POST http://sso-http-sso.apps.openbanking-
8005.openshiftworkshop.com/auth/realms/user1/protocol/openi
connect/token
grant_type=authorization_code
&client_id=036d8064
&client_secret=d4c26ac6a9ec2d2e1980d004be1ca56e
&redirect_url=https://openidconnect.net/callback
&code=eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..VTYG
DVZHGVcukw_0hA.38CdI0wjX-
AvF02Ah0D3WpvJCbxfwpvWw0CosuX8maHzXjiGLrH-Y-
PZltQ_fn41yhWkKjB0Br37wGi20zxj0tG351Vou4V698VyEoeFkwMnb1E6s
x4imFVrbgf48DdRh4lBzXJ_VyCsXudyDY8tUGgyDXQz9P48w9R4m0vR2AXI
I1heVHxTiNnTpK2z8i1dqVeG87oKSQT9iEdxkRzfuzj8fpYqGaia37.8nxz
```

HTTP/1.1 200

Content-Type: application/json

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiw
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiw
  "token_type": "bearer",
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk
  "not-before-policy": 0,
  "session_state": "45c16117-8802-4ca6-aa00-b535d9a26273",
  "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](#)

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldeiwia2lkIiA6ICJMdVRQlEwT  
UFrXYs3nUmK2nt9NjyMtRJNSUTDeRrihEfG4YfuZ_sEumafHT6EKPDlHaGVyq  
FL_5hD-wuU9cRi3nJ6iH4Rse8nAtgRKf-  
wt1NozI7fH1I8uW0oGbiPW_KwUi0DM0XWgd9JNLJr5Ef7mA0ksTXakZq4ACU  
DKia8u4NRv3mvBED8llzyefKJ9Tvl2bDEkFzHK5e213WMipBvLIsDoPvvWj-  
Fzl5MTE96sPZaokRxo4hQnJssk_WSwzoIC2X7yxDAYqNKIglH3xs709lSzdhK
```

Copy the “access_token” value in the step 2 (the long string).

```

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": ""

}

```

It should look something like this:

```

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwiia2lkA6ICJRa1RXJ2VwS2lwNVpFSkp3ZTd1cnFQUWtjSERNRi1SMnhGcE1tZUJ2aC1Vln0.eyJqdGkiOiYzJmZjQ5ZS01MDY4
LTQ0MjQtYTRiNS05MWU3OTk3MTM0YTMiLCJleHAIoJE1NDczOTc1NTlsIm5iZil6MCwiaWF0ljoxNTQ3Mzk2NjUyLCJpc3MiOjJodHRwczovL3NIY3VzS1zc28tc3NvLmF
wcHMub3BlbiIiYW5raW5nLm9wZW50cnkubWUvYXV0aC9yZWfsbXMvc3BlbnNoaWZ0liwYXVkljoiNWJjOTRmnNeiLCJzdWIoiMzJzDRhMy01MGm2LTQ3jQtYwuZ
ZC05ODdiYjA1ZTk4MzYiLCJ0eXAiOiJCZWFyZXliLCJhenAiOiiYmM5NGY2YSIsImF1dGhfdGltZSl6MTU0NzM5NTq3Niwc2Vzc2IvbI9zdGF0ZSl6ImZhMTQ5YjhILWQ0Z
TktNDliZS05NWlwLTK2MTNmZjBhNTViZCIsImFjci6ijAiLCJhbGxvd2VklW9yaWdpbnMiOltLCJyZWfsbV9hY2Nic3MiOnsicm9sZXMiOlsidWhx2FidGhvcmI6XRpb24i
XXOsInJlc291cmNIX2FjY2Vzcyl6eyJhY2NvdW50lpj7InJvbGVzlpblm1hbmFnZS1hY2NvdW50liwibWFuYWdIlfWFjY291bnQtbGlua3MiLCJ2aWV3LXByb2ZpbGUixX19LCJ
wcmVmZXJyZWRfdXNlcm5hbWUiOjJdmFsczk4QGV4YWiwbGUuY29tiwiZWhaWwiOjJdmFsczk4QGV4YWiwbGUuY29tln0.07y6GDFq5CajATODkywEuQqEuD5H7_Y
MqrVC4AMPthZ-m_xZ_DAPBEqj3nmzp1o1Joo0_4pMxNgKpyqCQIFY79GRS5lJE6aVrZK53rQkud5dlaZAE1-ryiD8CTP_MrQtsTS7bVkaFyCXNyFfxy3c-TER8GnGG90
OIYPxpy5M954slcp4CWxxA7zwVeQuNRRs5w2G2TCjrFyQjCzslnFwDrtdAdjbMiY7kq1cwRB5qM9ipdEEligDnH8dietiOZgY24sK10vtowjz_CHuWr5W3474dAZVFC7utwSt
I_bNcoj1gENRcz5cP7H7Nm8e4itWoSVPRVYcfDHyb9zixQ

```

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 (<https://apitester.com>) and add this as an Authorization header.

Use the URL for your API gateway:

```

https://userX.amp.apps.GUID.openshiftworkshop.com:443/open-data/banks

```

where you replace **userX** with your user ID and the **GUID** with the provided value from your instructor.

Under Header add the following values:

Name: **Authorization**

Value: **Bearer** < enter the access_token >

Request	Step Name
GET	https://user1.amp.apps.openbanking-8005.openshiftworkshop.com:443/open-data/banks
1	Headers + Add Request Header Authorization : Bearer eyJhbGciOiJSUzI

Let's hit **Test**

```
Request
Request Headers
GET /open-data/banks HTTP/1.1
Host: user1.amp.apps.openbanking-8005.openshiftworkshop.com
Accept: */*
User-Agent: Mozilla/5.0 (compatible; Rigor/1.0.0; http://rigor.com)
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJMdkVRQlEwT2RqTEszWXRVdzRtd09adXZ3ZFBHZ0tBc0tVby1qaUNhcUFrIn0.eyJqdG

Response
Response Headers
HTTP/1.1 200 OK
Server: openresty/1.13.6.1
Date: Thu, 21 Mar 2019 07:51:35 GMT
Transfer-Encoding: chunked
X-Application-Context: application
Set-Cookie: Sedba771e9ee85040f19fe8af69fd058=45ebfaa7d2323f90956b74c000c5abeb; path=/; HttpOnly
Cache-control: private

Response Body
{"short_name":"387832 5350","id":1,"address":"425-4002 Urna, Road","long_name":"Molestie Inc."}
```

You should see a 200 response and the values in the “Response Body”.

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