EXERCISE - SECURING YOUR CLOUD FOUNDRY APPLICATION

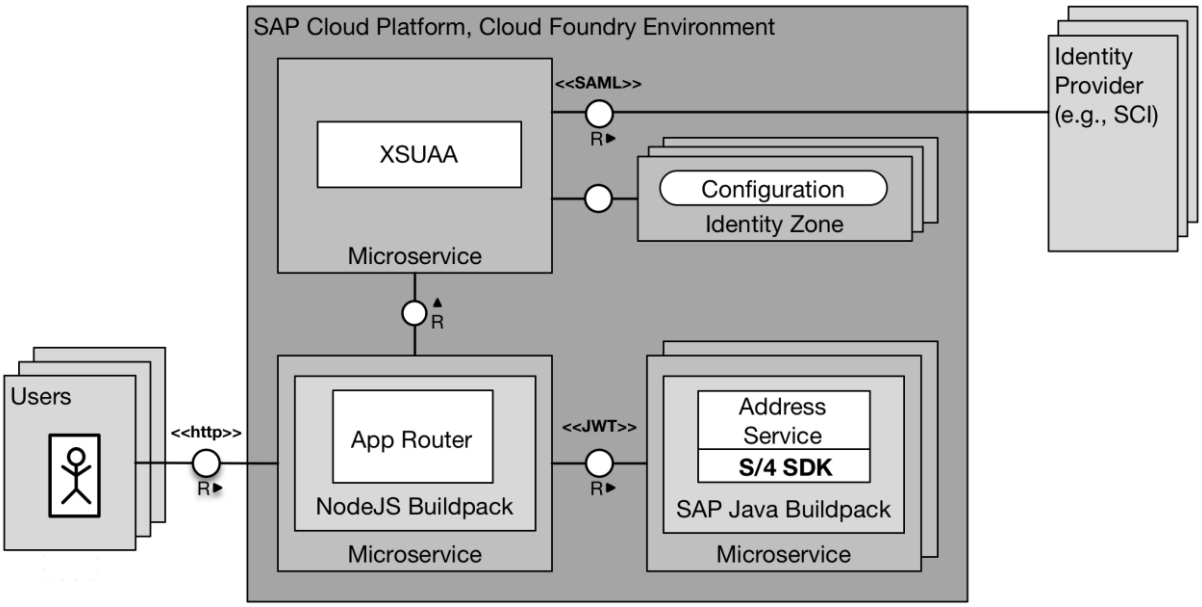
Duration: 60 min

Description

In this exercise, you'll learn how to

- introduce authentication into your application using the SAP app router component
- protect your Java microservice so that it only accepts requests based on a valid JSON Web Token (JWT), received from the app router
- assign roles and scopes to your application users and let your back-end deal with authorization information.

Let's give a quick look to how all this works considering the following diagram:



On one hand, the app router is a general entry point into the world of microservices. The main idea is that you can split an application into multiple microservices with independent deployability, polyglot runtimes, persistence, and independent teams. Therefore, a central entry component is required that hides the complexity of the microservice landscape from the end customer.

On the other hand, the app router is mainly responsible for managing authentication flows. The app router takes incoming, unauthenticated requests from users and initiates an OAuth2 flow with the XSUAA service. The XSUAA service is an SAP-specific fork of CloudFoundry's UAA service to deal with authentication and authorization. If a user authenticates at the XSUAA, it will respond with a JSON Web Token (JWT) containing the authenticated users as well as all scopes that he or she has been granted.

Goal

The goal of this exercise is to secure your application in SAP Cloud Platform, Cloud Foundry using the SAP app router component.

Prerequisites

Here below are prerequisites for this exercise.

- A trial account on the SAP Cloud Platform. You can get one by registering here

- Apache Maven 3.3.9+ download it here
- Java JDK 8
- Latest LTS version of NodeJS download it here
- Cloud Foundry CLI download it here
- A S/4HANA system or a mock server available

Steps

1. Setting up the app router and reviewing the file structure
2. Deployment of the new application version and the app router
3. Check scopes and assign users to application roles

Setting up the app router and reviewing the file structure

1. Firstly, clone the prepared repository and switch to the branch learning/security:

```
git clone https://github.com/SAP/cloud-s4-sdk-book.git
git checkout learning/security
```

2. Go to the folder ***<your project folder>/approuter*** and take a look at the file *package.json:*

```
{
  "name": "approuter",
  "dependencies": {
    "@sap/approuter": "*"
  },
  "scripts": {
    "start": "node node_modules/@sap/approuter/approuter.js"
  }
}
```

This file will be now used to install the latest version of the app router using npm.

3. To install the app router, execute the following commands in your command line:

```
cd <your project folder>/approuter
npm config set @sap:registry https://npm.sap.com
npm install
```

If successful, you should find a *node_modules* folder besides your *package.json*.

4. Let us now analyze some important files that were added to the app router folder

The app router folder should now look as follows:

```
.
├── approuter
│   ├── node_modules
│   ├── package.json
│   └── xs-app.json
├── manifest.yml
└── xs-security.json
```

4a. File ***xs-app.json*** is located under ***<your project folder>/approuter/approuter*** and contains information regarding routes and logout url.

4b. **M*anifest.yml*** file is located inside ***<your project folder/approuter>*** and contains data to successfully deploy the app router in SAP Cloud Platform, Cloud Foundry:

Note: Replace the route parameter with your subaccount's subdomain as well as the URL parameter of the destinations variable with the correct base URL of your previously deployed Address Manager microservice. For example, if you have created a TRIAL account on SAP Cloud Platform previously with a P-User, the route of the app router will have a form like *approuter-p1942765239trial* and the

URL to the microservice might be *address-manager-unsparse-subutopian* based on the random route generated in previous sessions.

4c. In order to create an instance of the XSUAA service, you require the ***xs-security.json*** descriptor file. You can find this file in ***<your project folder/approuter>***.

The file contains the unique application name, tenant-mode (shared) and scopes and role templates that will be used for authorization customizing.

5. Analyze changed ***web.xml*** file in the example application. It now contains the security constraint for the API ***/api/business-partners.***

Deployment of the new application version and the app router

1. Drop and create a service instance called `my-xsuaa` of the XSUAA service by issuing the following command and using the *xs-security.json* file as configuration input.

```
cf unbind-service example my-xsuaa
cf delete-service my-xsuaa
cf create-service xsuaa application my-xsuaa -c xs-security.json
```

This is required to consider the new security configurations provided in the ***xs-security.json***.

2. Deploy the secured example application and remove the set ALLOW_MOCKED_AUTH_HEADER, which was used to mock the user and tenant information. Now the application can retrieve these data from the JWT provided by the approuter:

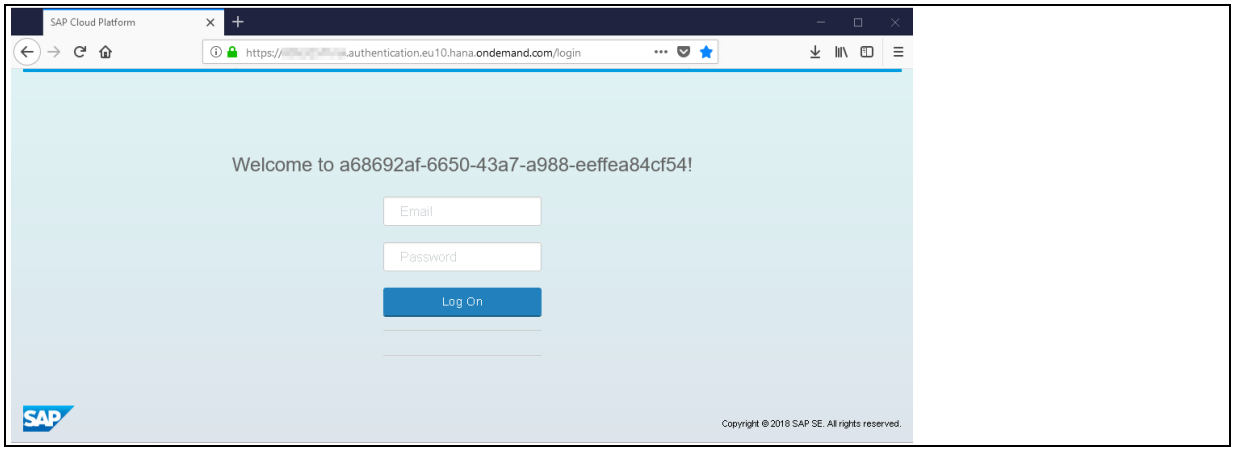
```
cd <your project folder>
cf api https://api.cf.eu10.hana.ondemand.com
cf login
cf push
cf unset-env example ALLOW_MOCKED_AUTH_HEADER
```

3. Deploy the app router using the following commands:

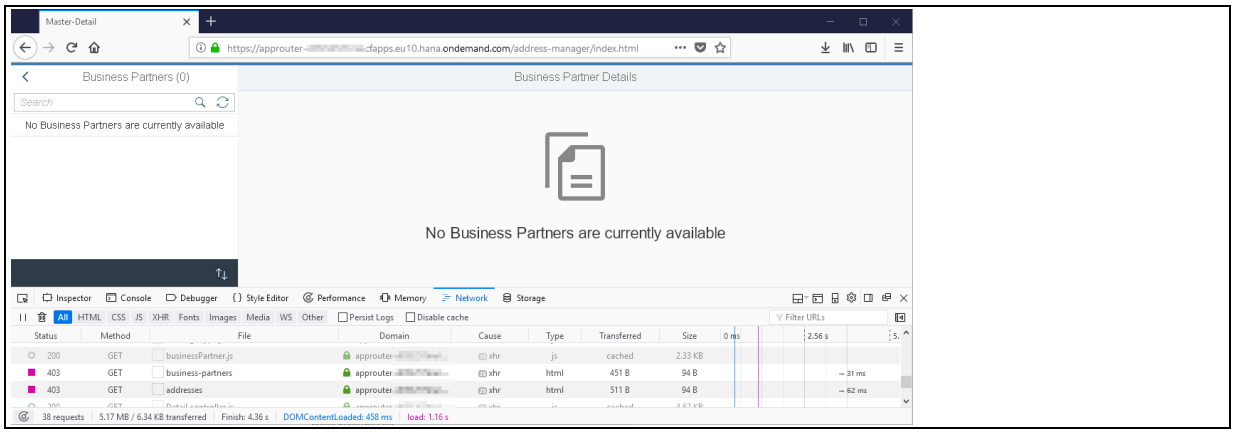
```
cd <destLocation/approuter>
cf push
```

Validating results:

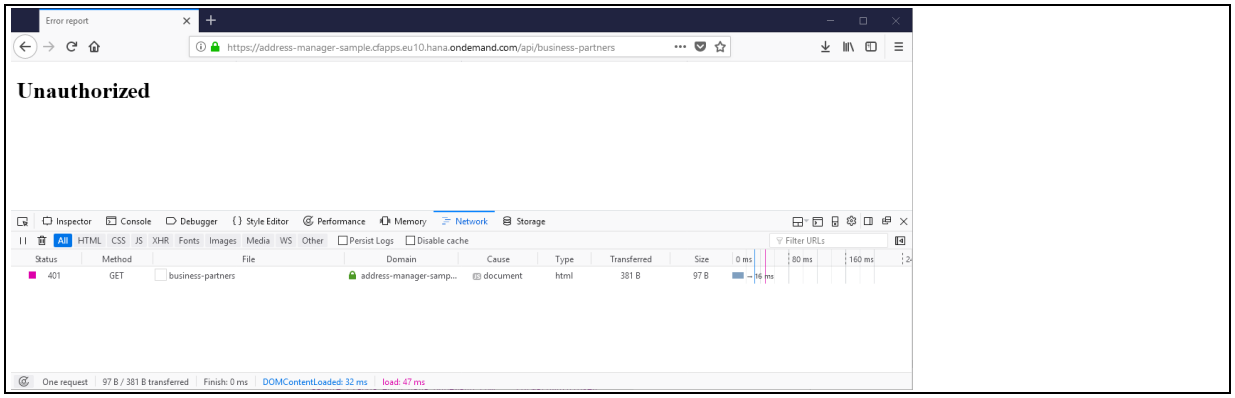
After this, you should be able to locate the app router from within your browser using the host name of your deployment as well as the subaccount's subdomain following the pattern as specified before. For example, using our exemplary user P1942765239, the final URL would be https://approuter-p1942765239trial.cfapps.eu10.hana.ondemand.com/address-manager/index.html. This should present a page similar to the figure below where you can use your e-mail address and password to login.



After logging on, you will get 403 (Forbidden), as authentication has worked but the user is lacking the authorization we required to protect the endpoints accordingly (you can use “inspect panel” in your Chrome browser to check the response code).



Accessing the backend service directly will reject any requests with a 401 HTTP status (Unauthorized) code because we do not provide any valid JWT (you can use “inspect panel” in your Chrome browser).

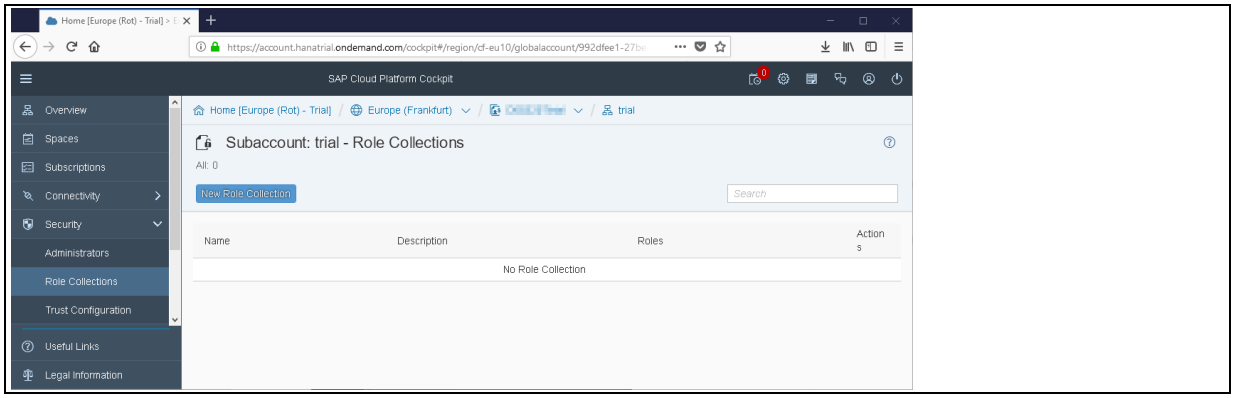


Check scopes and assign users to application roles

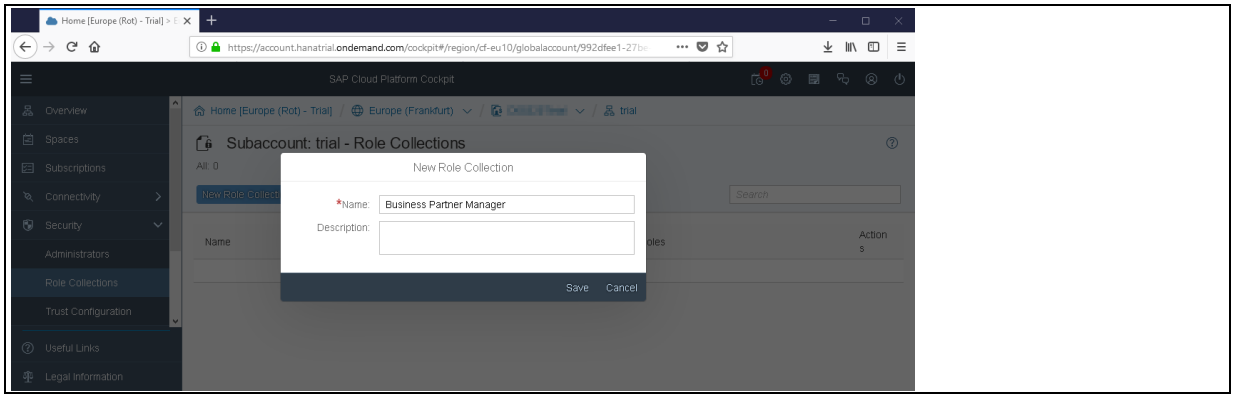
In this part of the exercise, we set up business authorizations that can be checked as part of the application semantics.

Now that we have a new XSUAA configuration in-place and protected our backend microservices, we need to assign the corresponding roles to our users. For that purpose, use SAP Cloud Platform cockpit.

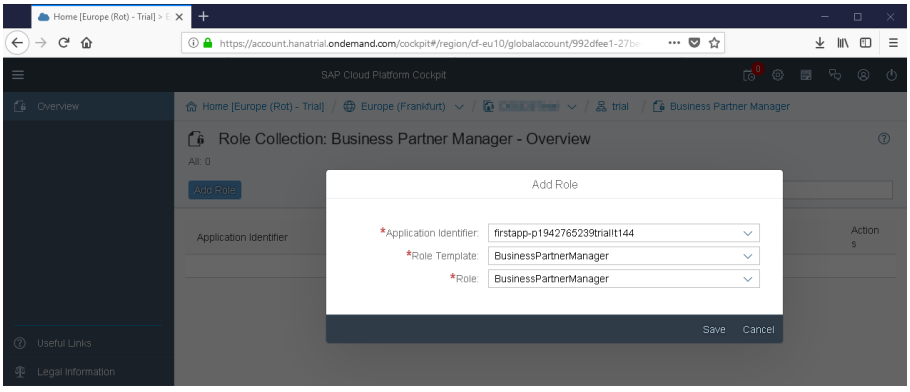
2. Create a role collection: Open the *Role Collections* tab under the *Security* ribbon of your Cloud Foundry subaccount as shown in the figure below.



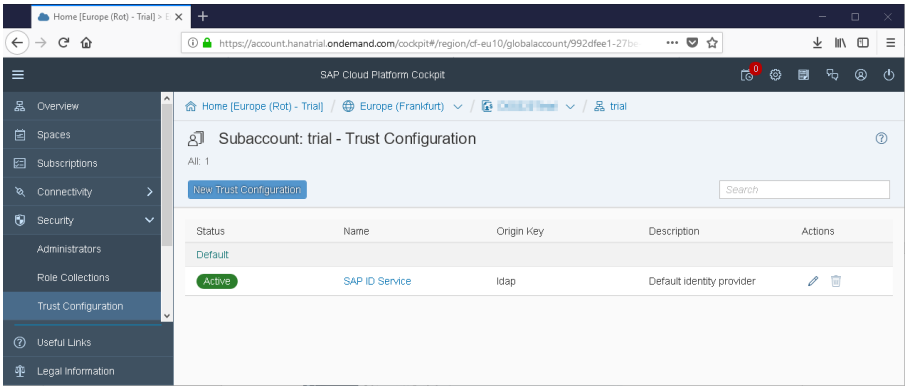
Create a new role collection with the name "*Business Partner Manager*" and click *Save.*



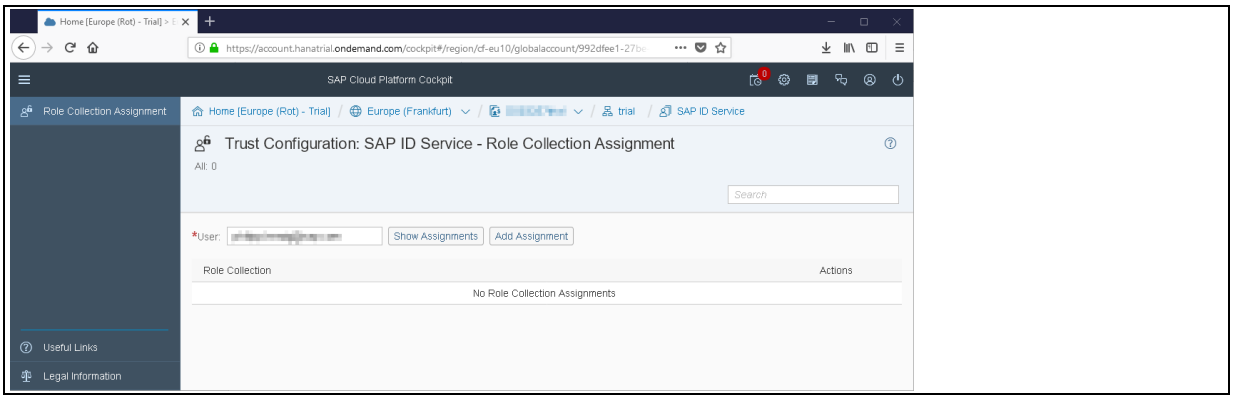
3. Assign a role to the role collection: Select the newly created role collection "*Business Partner Manager*" and select *Add Role*. From the menu, select your application presented as the application identifier `xsappname` from your *xs-security.json* and the corresponding role template defined earlier.



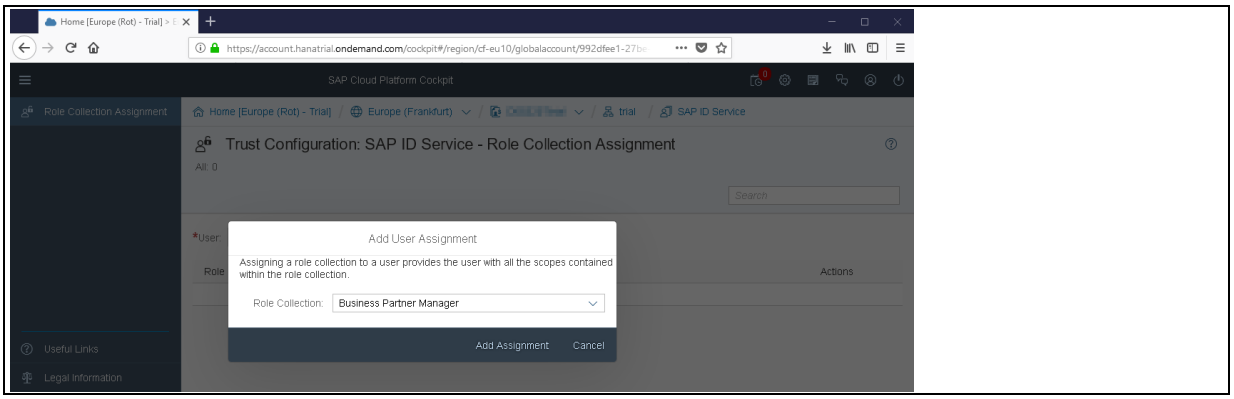
4. Assign a user to the role: Select the *Trust Configuration* from the *Security* menu and select the *SAP ID Service* from the list



In the opening dialog, enter your e-mail into the user field and click *Show Assignments*.



Then, click *Add Assignment* and choose the "*Business Partner Manager*" role collection from the menu to assign it to your user.



Based on this role collection assignment, you are now able to use the application as before using our authenticated and authorized user.

Validating results:

Logout using the following URL: *<app router URL>/logout* and then login using your credentials again.

Access the application via app router: *<App router URL>/address-manager/index.html*. The application should show the business partner data again.