Table of Contents

1. Prerequisites for this document 2

2. CAPM Project 3

2.1 Create a Dev Space 3

2.2 Configure Dev Space 4

2.3 Create CAPM Project 5

2.4 Deploy CAPM Project 7

2.5 Run CAPM Project Locally 9

2.6 Integrate REST / SAP OData Services in CAPM as External Service 11

2.7 Maintain Config JSON in BTP (User Defined Variable) and expose using CAPM 14

2.8 Create a Destination in BTP to consume CAPM Service 16

2.9 Consume CAPM Service in UI5 Application with Destination 17

2.10 Reference Links 18

3. AutoScaler 19

3.1 Introduction 19

3.2 What is Application AutoScaler 19

3.3 Dynamic Scaling 19

3.4 Define AutoScaler Service in MTA.yaml 20

3.5 Reference Links on AutoScaler 20

4. Access SAP OData Service with No Authentication in UI5 Project 21

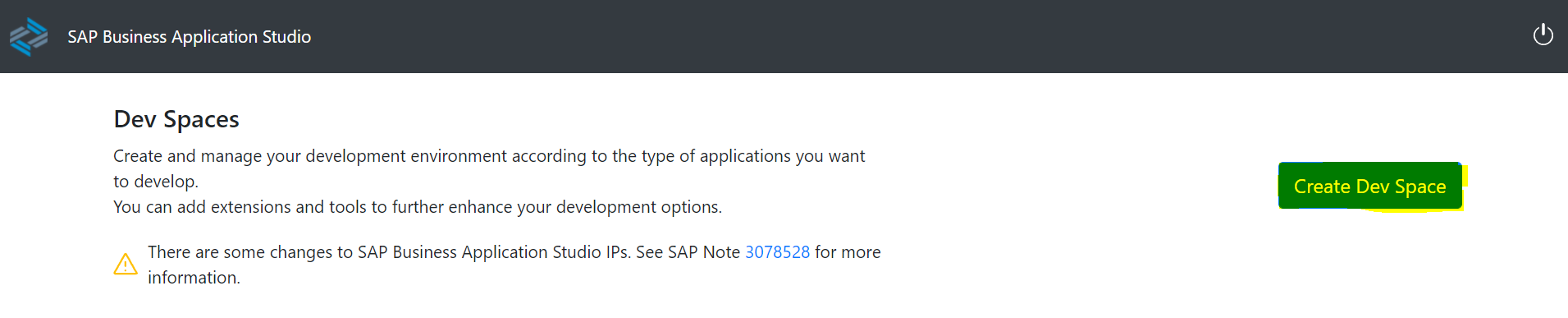
# Prerequisites for this document

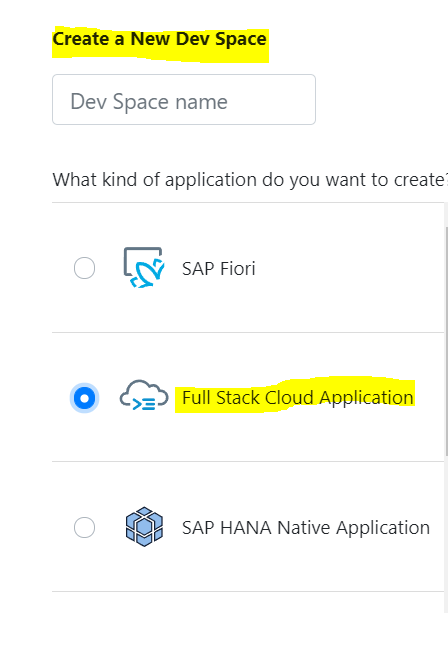
Code Samples and steps mentioned in this document are created using SAP Business Application Studio and SAP Cloud Foundry

# CAPM Project

# Create a Dev Space

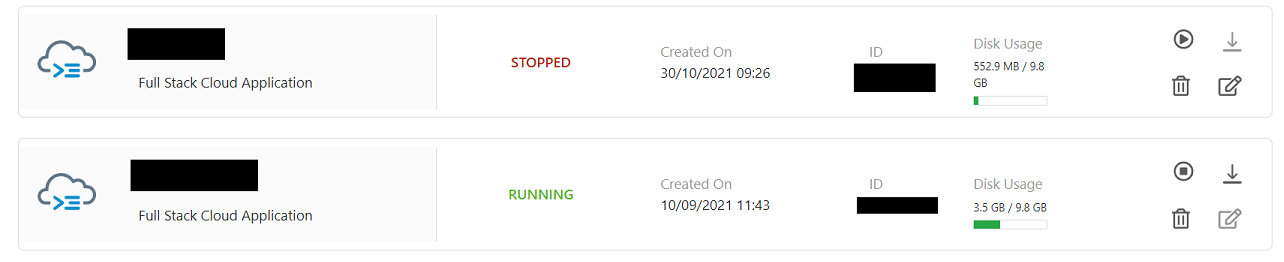
Login to your Business Application Studio (BAS), and choose Create Dev Space, enter any name you wish for your dev space and select **FULL STACK CLOUD APPLICATION**





Once you click on Create Dev Space, it will be configured with most common tools, we need for the type of application we choose, Dev Space will then begin starting and the process will take few minutes or so, as cloud environment is being created.

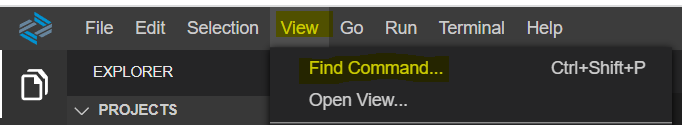
Once the Dev Space reaches Green Status of **RUNNING**. We can click of the name of the Dev Space and it will load the editor in browser

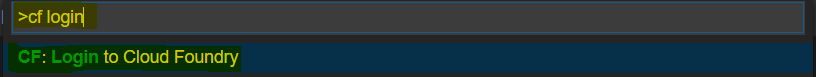


We will be redirected to newly created SAP Business Application Studio Dev Space.

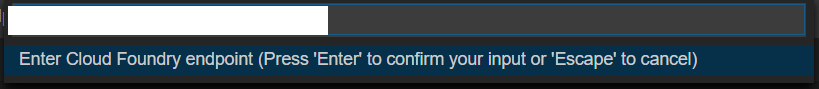
# Configure Dev Space

Go to **View Menu** and Select **Find Command**





Command window will open at the top of the Business Application Studio, input will prompt you for the API end point

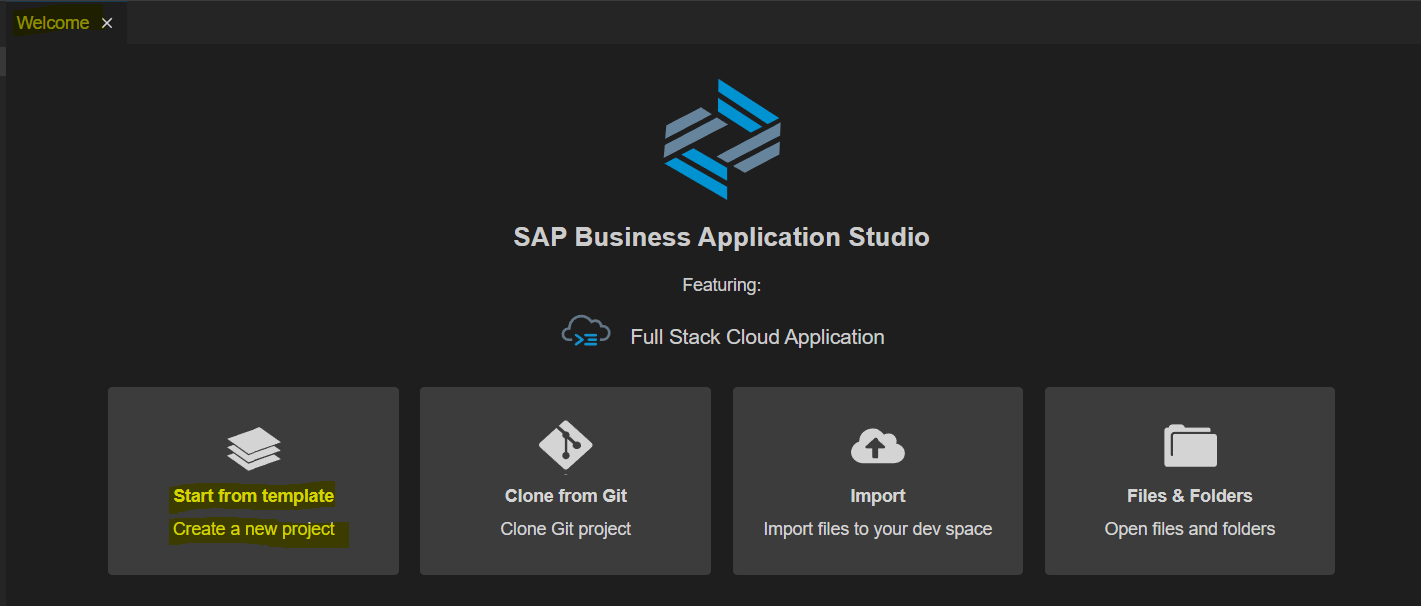


Default is likely the correct value, if you need to confirm you can find it in SAP BTP cockpit at the subaccount level.

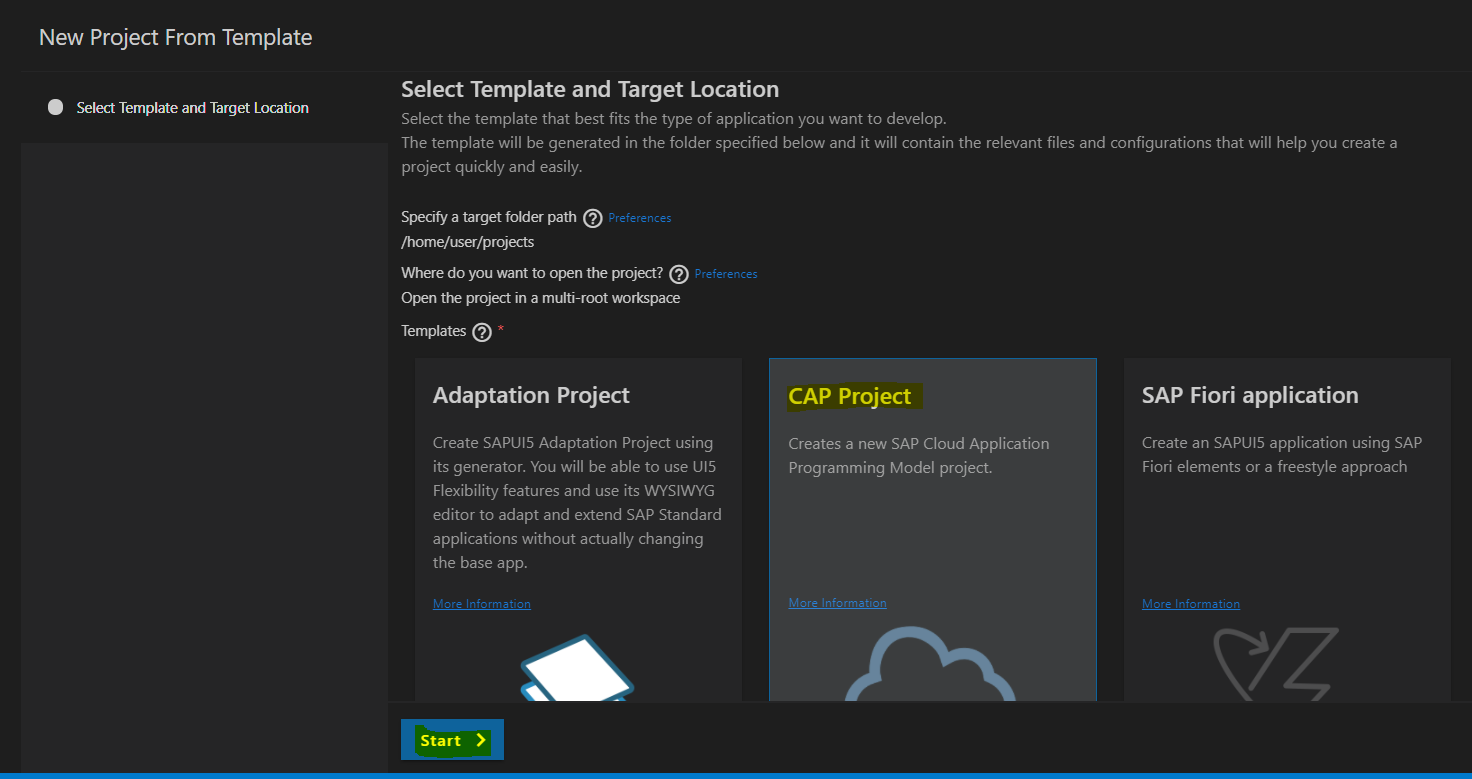
Press enter to confirm your API end point, in the next input field enter your email address and password. Once Authentication successful, in the next input you will be asked for your Organization, and final input you will be asked to select your space.

# Create CAPM Project

From SAP Business Application Studio >> Welcome Tab, Click on **START FROM TEMPLATE CREATE A NEW PROJECT**.



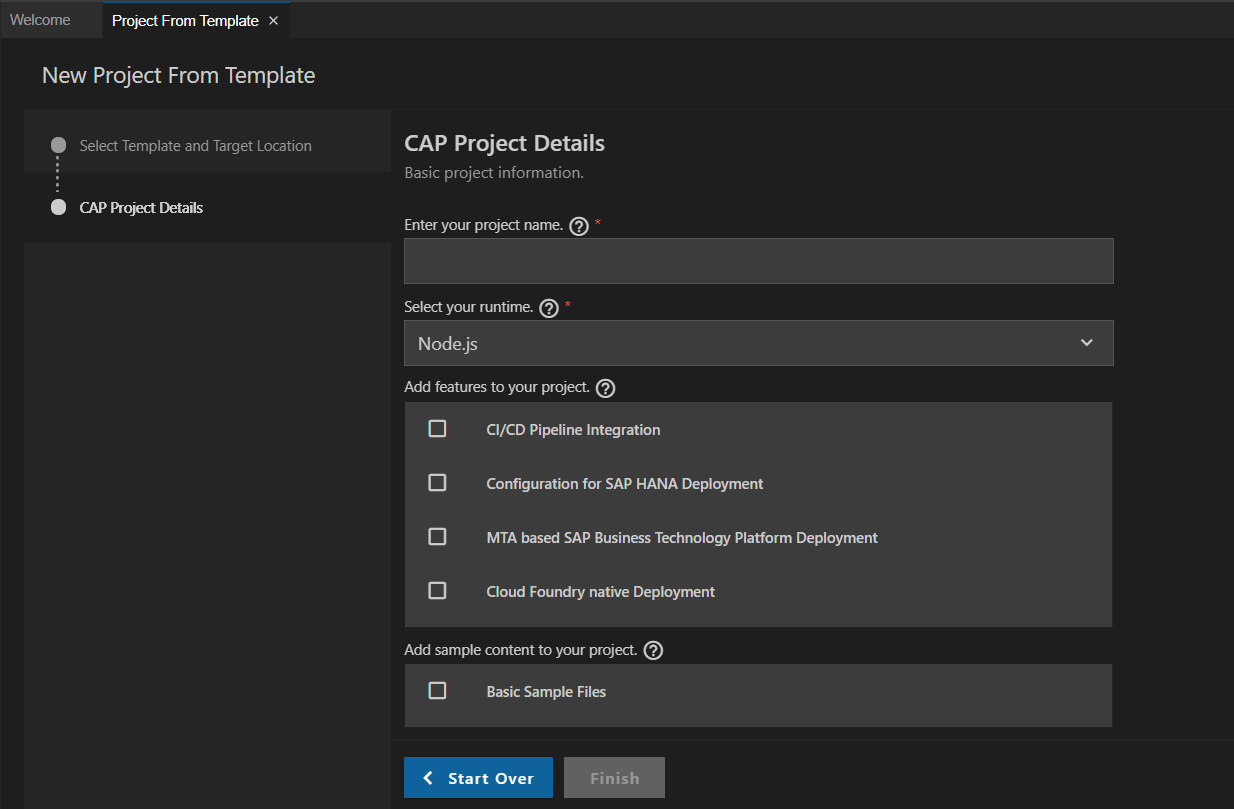
Choose **CAP Project** and click **Start**



Enter **Project Name**,

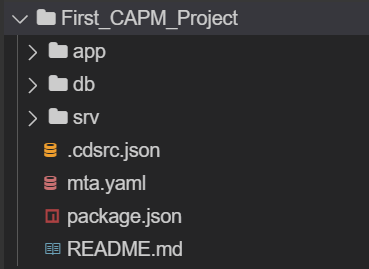
Choose **NodeJS** as **runtime**,

Select **MTA based SAP Business Technology Platform Deployment** as **features** and Press Finish



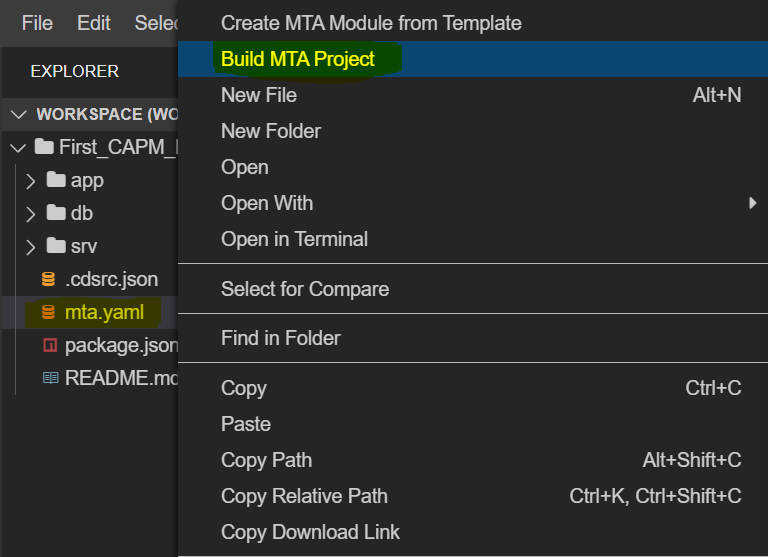
Choose to open the project in a New Workspace, once the generation is complete

Wizard has generated MTA with two modules, **Database Module** (Folder Name: **db**) and **Service Module** (Folder Name: **srv**), Expand them to check the folder structure (Check the screenshot below)

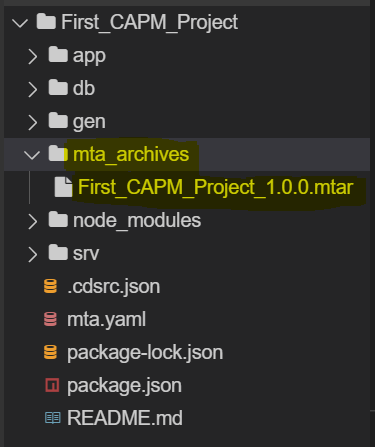


# Deploy CAPM Project

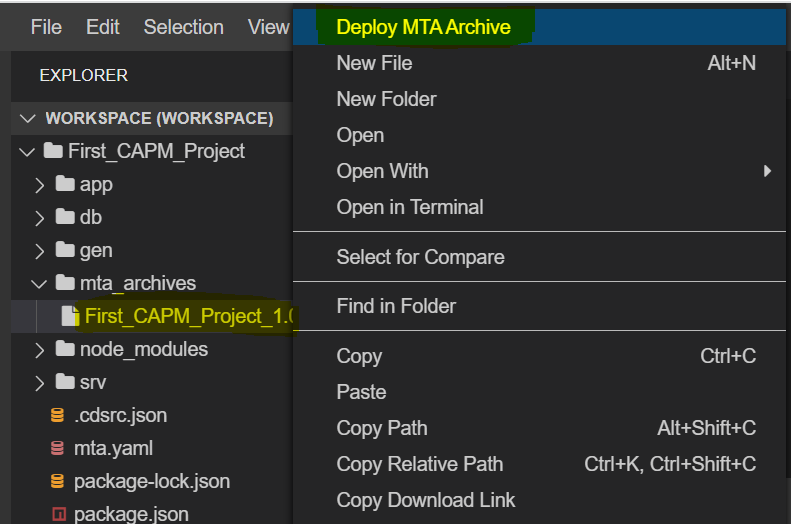
Right Click on mta.yaml and select **Build MTA Project**



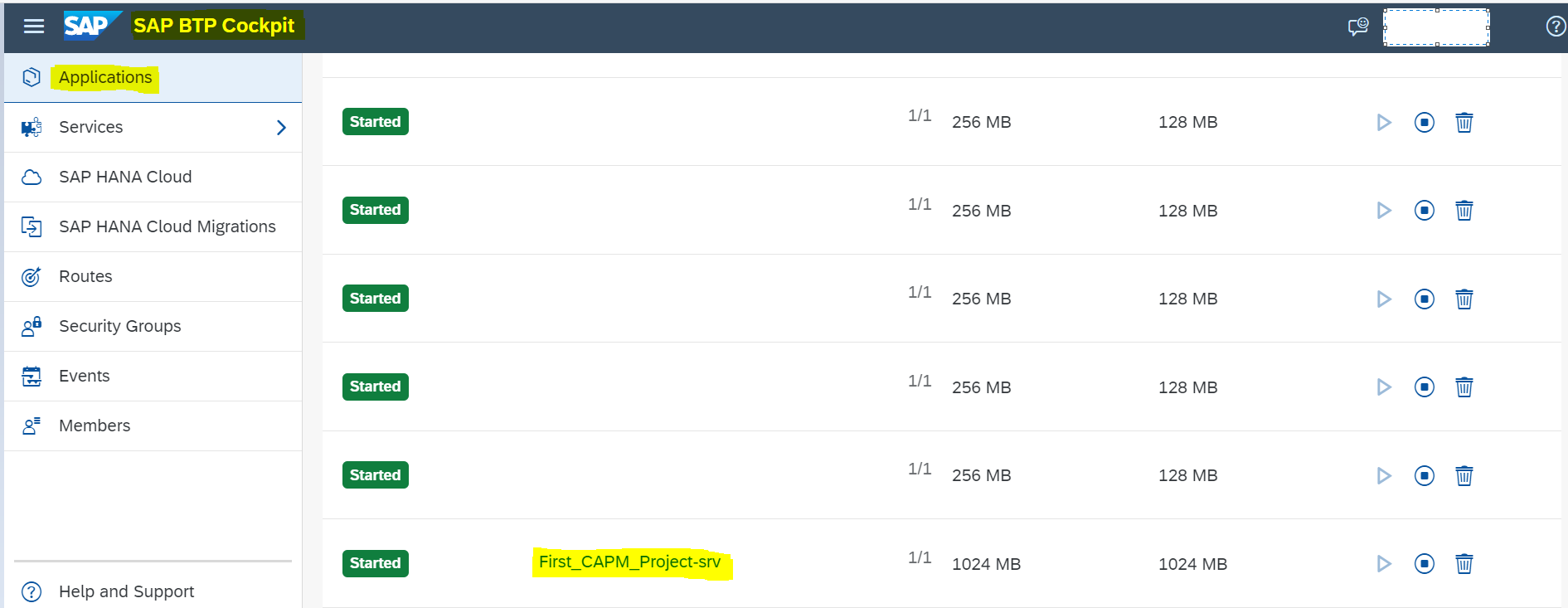
Upon Successful build new folder **mta\_archives** gets created; this folder contains a MTAR file.



Right click on MTAR file and select **DEPLOY MTA ARCHIVE**

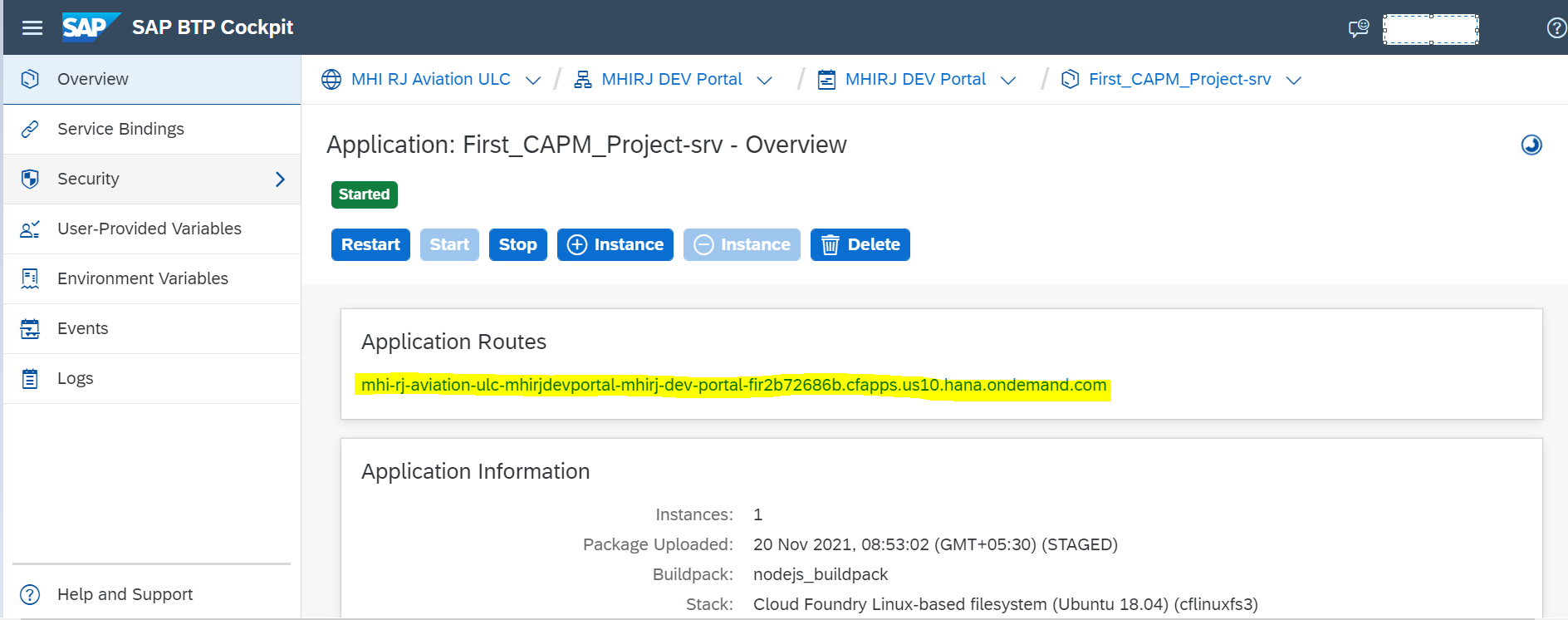


After successful deployment login to BTP Cockpit, under Applications you can see the CAPM Project deployed



Click on the **CAPM Project Link**, to get the URL and access the application

URL under Application Routes Section, is the URL for accessing the application



# Run CAPM Project Locally

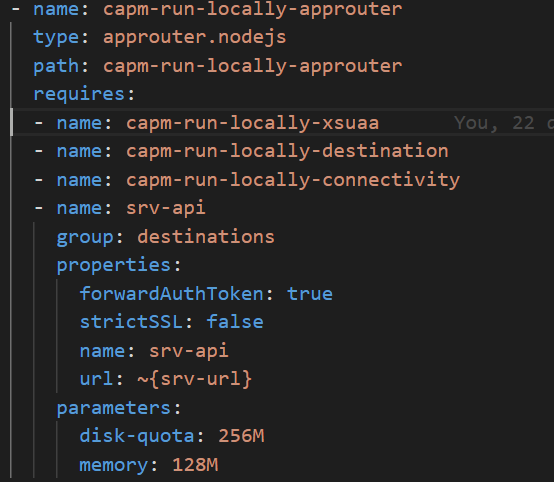
To run the CAPM Project Locally (from Local Environment), and to know which user has logged on we need to get the authentication token (JWT Token) injected in the Service. This is done by Approuter, Approuter will get the token from uaa.service and add it to the Authorization header in the request to the service.

To achieve this, we need to use default-env.json

**Approuter:** Approuter is configured in “Approuter” folder in the project xs-app.json



**MTA.yaml**:



This Approuter module requires 2 services:

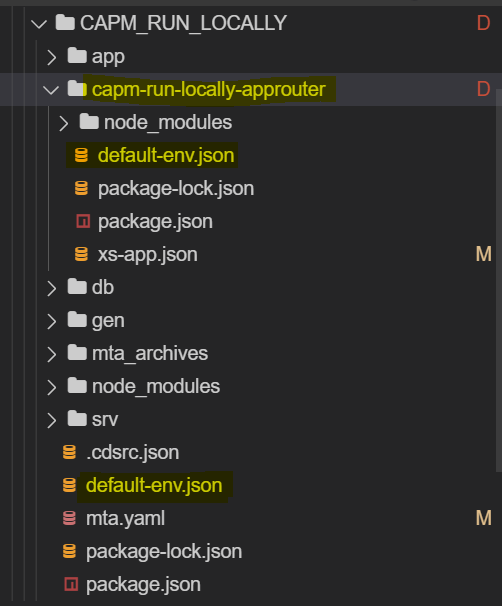
1. **capm-run-locally-xsuaa** (which is uaa service)
2. **srv-api** destination (which is provided by capm-run-locally-srv module)

In the srv-api destination, we set forwardAuthToken to true. This will forward JWT Token from Approuter to the destination

Create a default-env.json file in Approuter folder, Get the environment variables for the approuter app, we can get environment variables from BTP cockpit Environment Variables or by using CLI Commands



We also need to copy default-env.json to the root directory of our project because in the service , we need to validate JWT Token

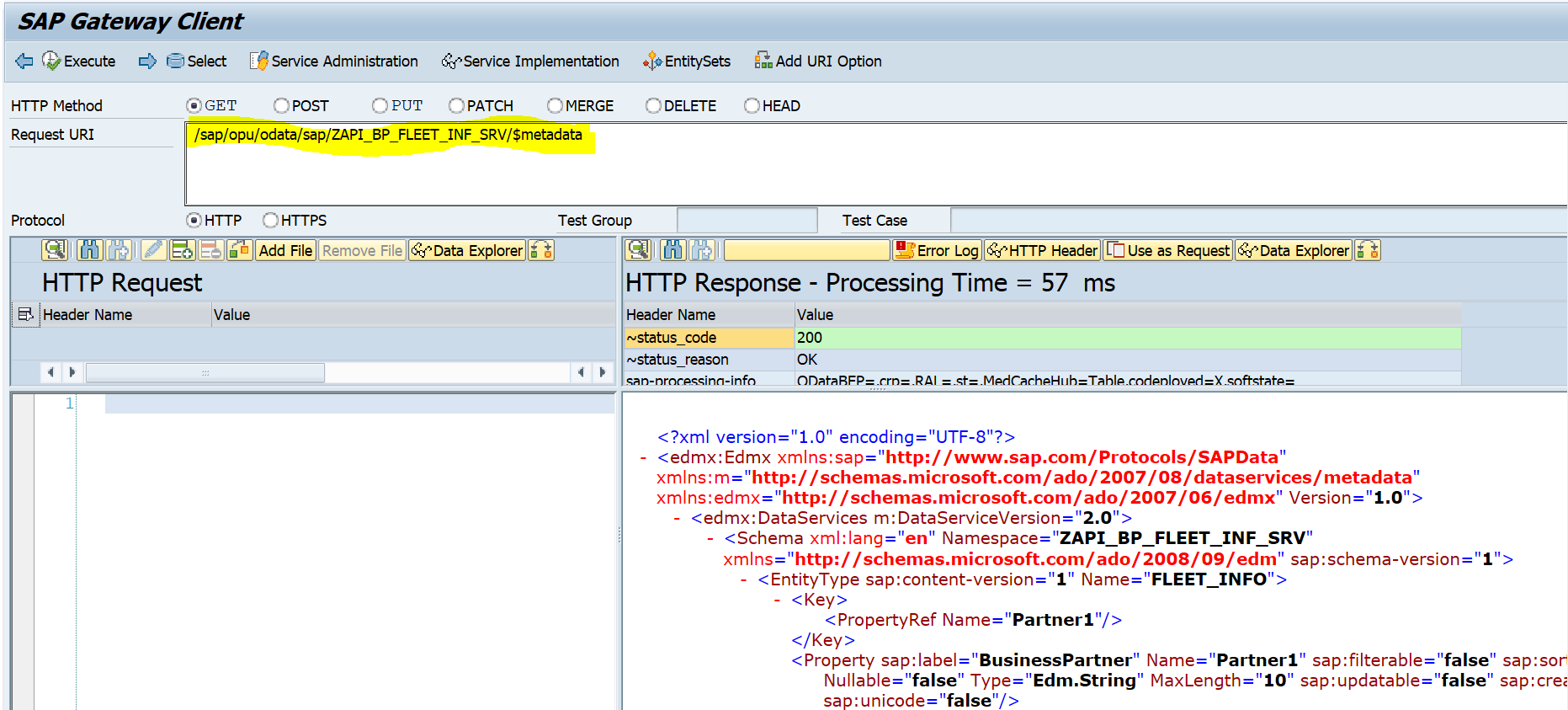


# Integrate REST / SAP OData Services in CAPM as External Service

Follow the steps in Section 2.3 and create a new CAPM Project. Check the **srv folder** in project explorer, and make sure that you have two files with same name and different extensions. For example: cat-service.cds and cat-service.js, if not create them manually.

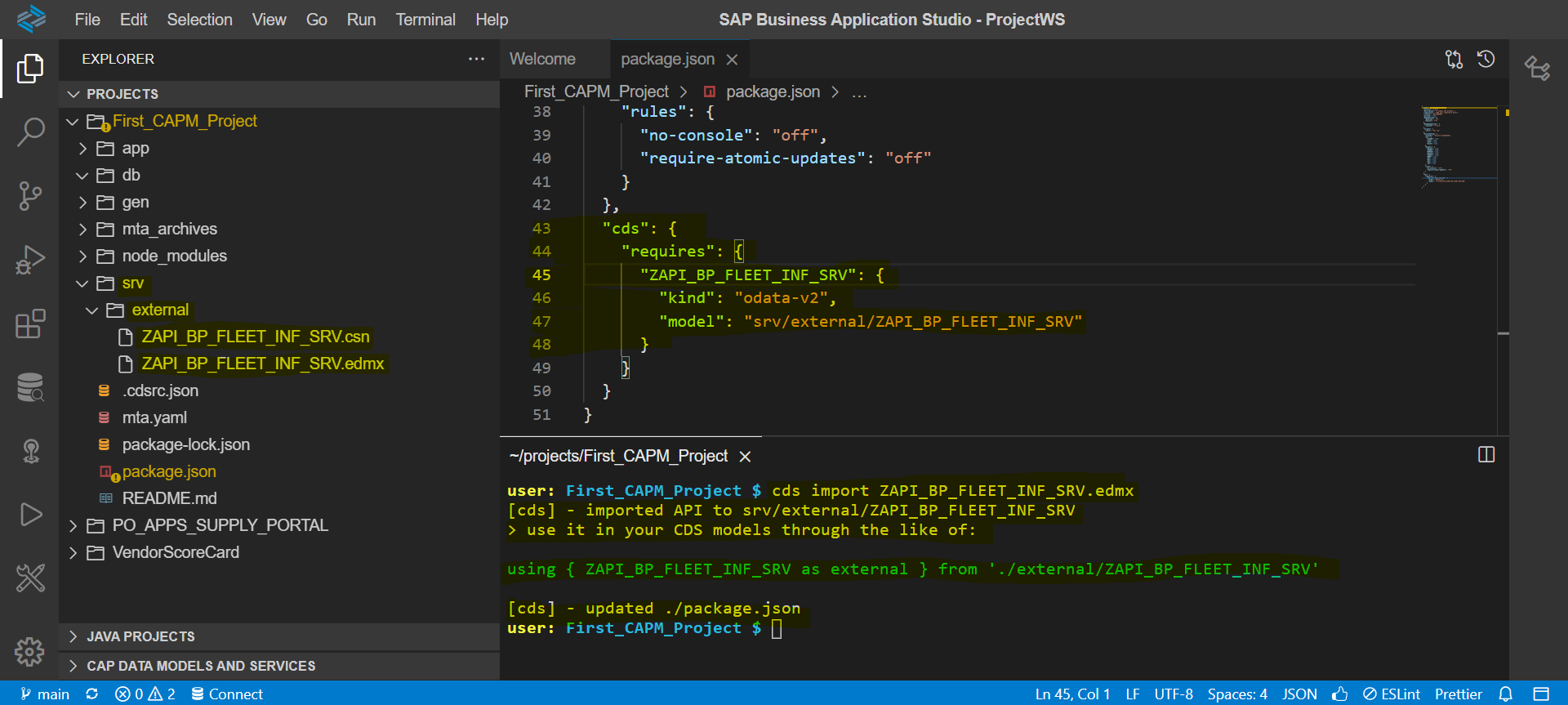
CDS file contains Service and Entity Declarations, and JS file contains its implementation.

Get the service document or EDMX file of OData Service by using $metadata tag, and then save the xml data with .edmx extension (**I have taken an example of /sap/opu/odata/sap/ZAPI\_BP\_FLEET\_INF\_SRV**)



Save the metadata in notepad file with .edmx extension

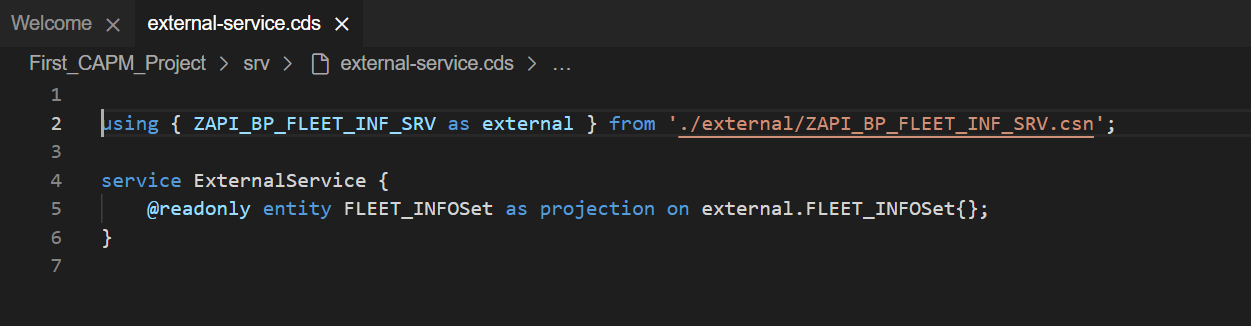
Copy the .edmx file into Project Root Folder and then Use **cds import** command to import .edmx file into your project



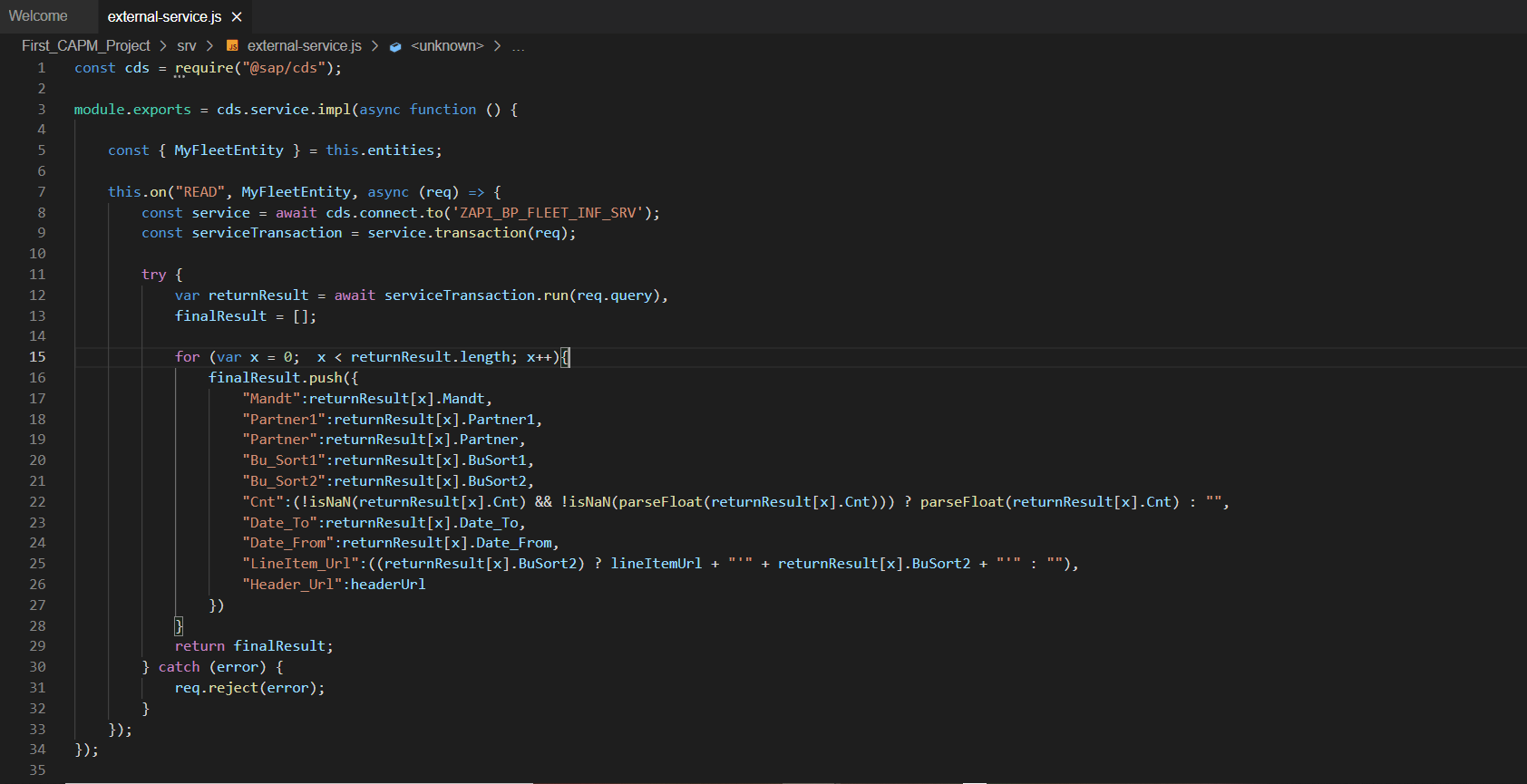
Few things that happened during import

1. .edmx file was imported to folder **srv/external**
2. .csn file was generated based on the edmx file and also created inside folder **srv/external. CSN** stands for **Core Schema Notation**, this is the schema definition that is understood and used by the CDS framework.
3. Package.json is updated with cds.requires configurations for the imported external service

Create a external-service.cds file to define our ExternalService



Create custom handler for our **ExtenalService** and implement the READ event of **FLEET\_INFOSet** Entity

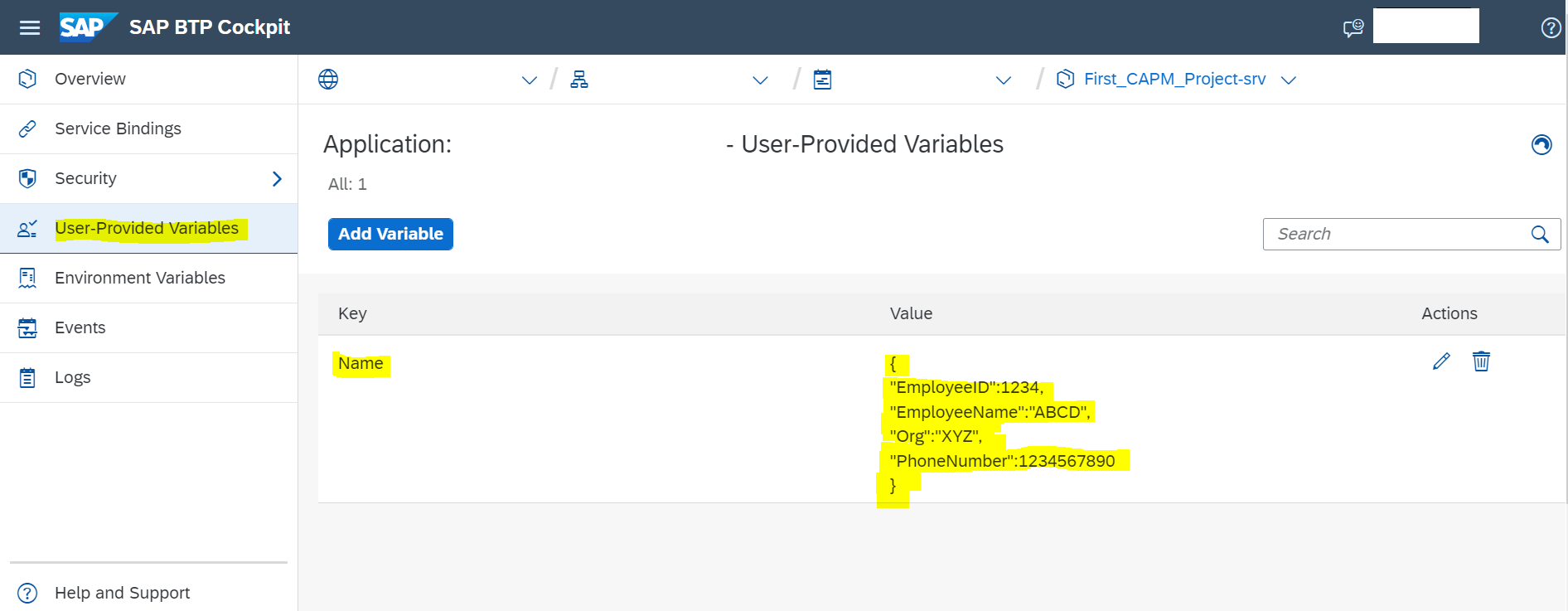


Now we are done with coding, follow the steps section 2.4 to deploy and test the application

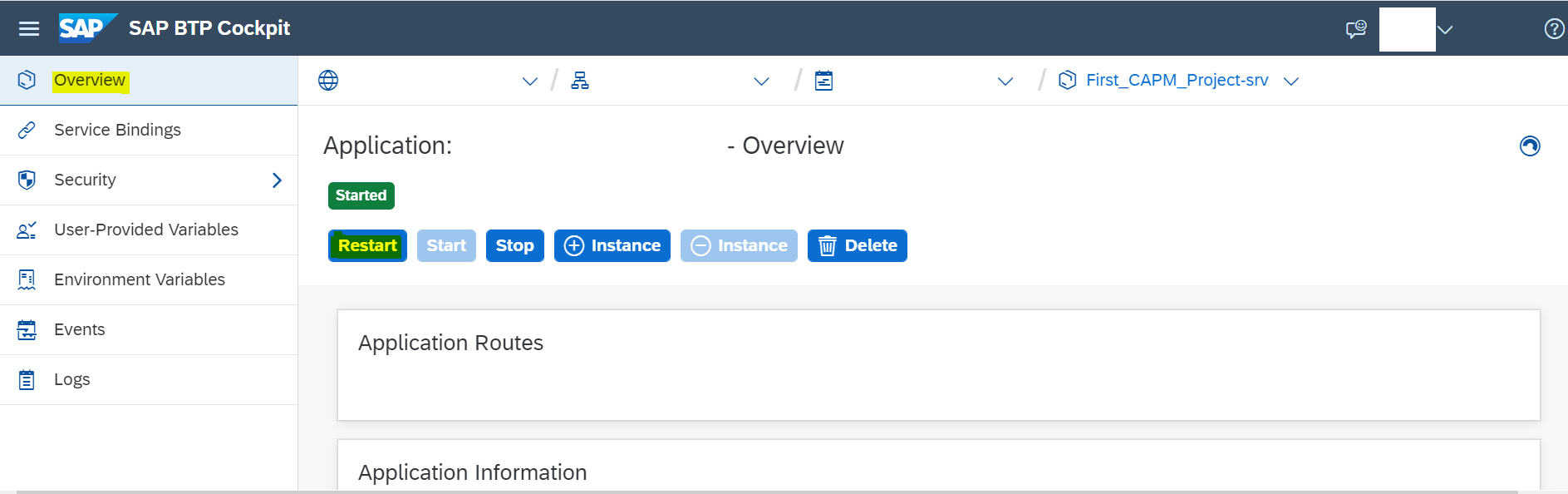
# Maintain Config JSON in BTP (User Defined Variable) and expose using CAPM

If we maintain Local JSON files / models in UI5 Applications, we need to redeploy UI5 application, each time we change anything in JSON files / models, this creates unnecessary over head for the project.

To eliminate this, we can use the combination of User-Provided Variables in BTP Cloud Foundry and CAPM Project.

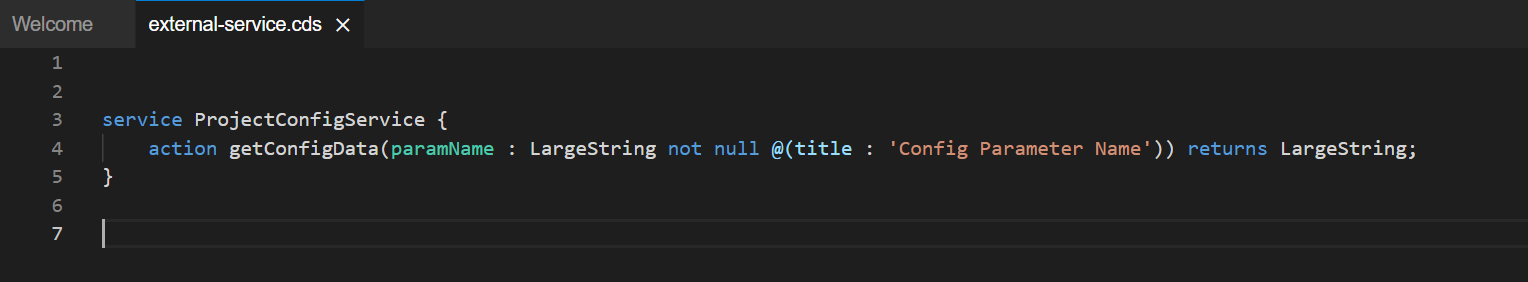


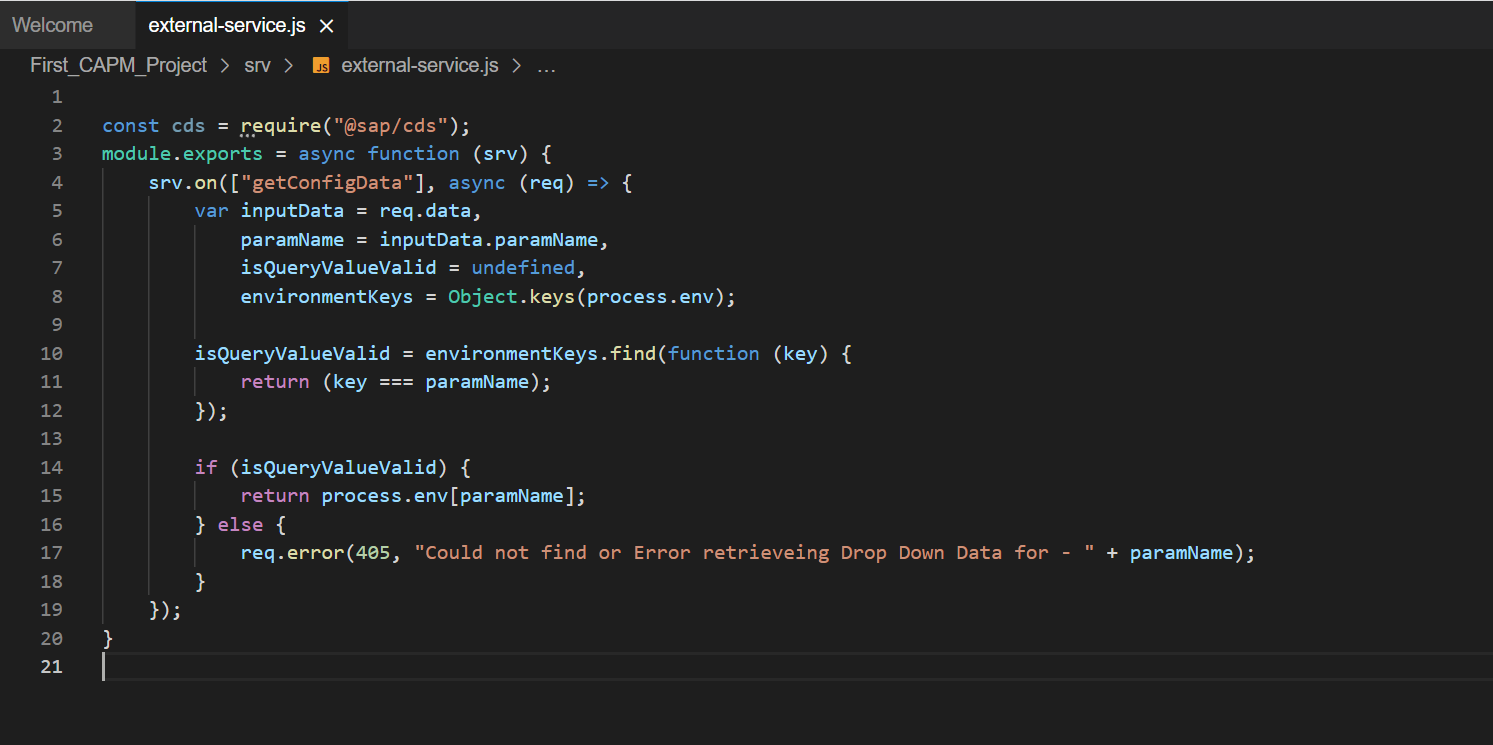
Once you have created all the necessary **User-Provided Variables**, go to **Overview** section, and **Restart** the application



Now Follow the steps in Section 2.3 and create a new CAPM Project. Check the srv folder in project explorer, and make sure that you have two files with same name and different extensions. For example: cat-service.cds and cat-service.js

CDS file contains Service and Entity Declarations, and JS file contains its implementation.





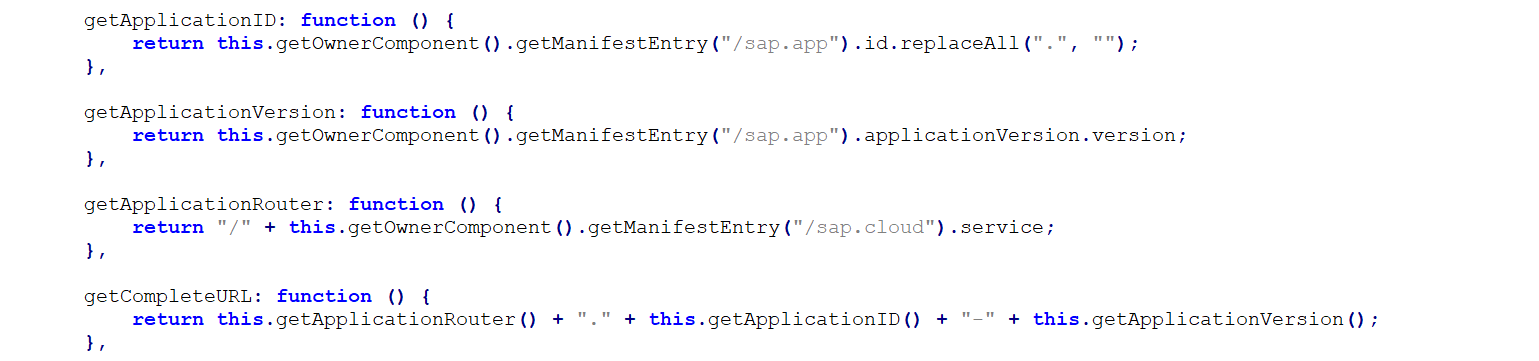
Now we are done with coding, follow the steps section 2.4 to deploy and test the application

# Create a Destination in BTP to consume CAPM Service

Destinations

# Consume CAPM Service in UI5 Application with Destination

We Can Consume, CAPM by making AJAX Call, as below





# Reference Links

Few Reference Links

1. <https://github.com/SAP-samples/cloud-cap-samples/tree/openSAP-week1-unit4-final>
2. <https://developers.sap.com/mission.btp-application-cap-e2e.html>
3. <https://www.youtube.com/playlist?list=PL6RpkC85SLQBFi4SK77b2y4EwlXMVG0XJ>
4. <https://github.com/SAP-samples/cf-mta-examples/blob/main/app-routes/modelled_with_hosts_and_domains/mta.yaml>
5. <https://github.com/gregorwolf/bookshop-demo>
6. <https://github.com/jowavp/SAP-CAPM-Nodejs-Authorisation-example/tree/master/approuter>
7. <https://blogs.sap.com/2020/05/01/sap-cloud-application-programming-model-demo1/>
8. <https://blogs.sap.com/2020/05/03/sap-cloud-application-programming-model-demo2/>
9. <https://blogs.sap.com/2020/05/10/sap-cloud-application-programming-model-demo3/>
10. <https://blogs.sap.com/2020/05/16/sap-cloud-application-programming-model-demo4/>
11. <https://blogs.sap.com/2020/06/07/sap-cloud-application-programming-model-demo5/>
12. <https://blogs.sap.com/2020/07/12/sap-cloud-application-programming-model-demo6/>
13. <https://blogs.sap.com/2021/06/16/how-to-implement-bound-functions-and-unbound-functions-in-sap-capm/>
14. <https://blogs.sap.com/2019/03/11/sap-cloud-platform-backend-service-tutorial-9-cds-how-to-use-select-statement/>
15. <https://blogs.sap.com/2020/12/27/consuming-a-rest-service-with-the-sap-cloud-application-programming-model/>

# AutoScaler

# Introduction

In cloud applications, workload is a very important factor. Workload can be measured as CPU usage, memory utilized, response times, network traffic etc. There are mainly two types of workloads as follows:

1. **Static Workload:** It’s the normal workload of an application. When workload is static then application behaviour is always the same and stable. To manage this workload, proper resources need to be assigned to the application.
2. **Dynamic Workload:** It’s the workload that keeps on changing overtime. In this case, workload can grow or shrink randomly at specific times. To deal with this kind of workload, there are two options. First option is to assign a lot of resources to the application. But the problem is, most of time application won’t be using or needing that much resource. Second option is to use the Application AutoScaler service provided by Cloud Foundry.

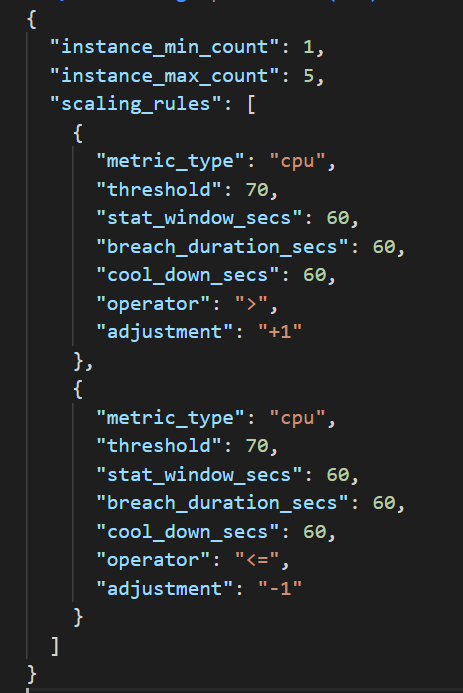
# What is Application AutoScaler

Automatically Scale your application to meet their dynamic resource needs.

Application AutoScaler lets you automatically increase or decrease the number of your application instances based on the policies you have defined.

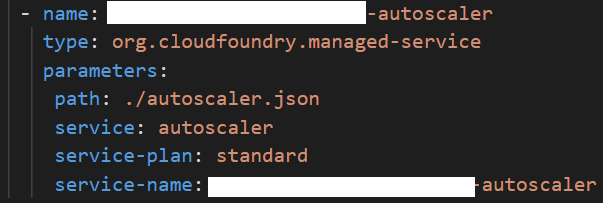
# Dynamic Scaling

Application instances are scaled up and down based on the values of various parameters like memory consumed, response time, throughput and so on. The scaling rule that we will be writing will be triggered based on CPU usage. We must define minimum and maximum instance count of the application. The configuration is defined in autoscaler.json (Which is on the same level as mta.yaml in project folder)



# Define AutoScaler Service in MTA.yaml

Define AutoScaler service in MTA.yaml, so that during deployment AutoScaler service gets created on need basis and binded to application



# Reference Links on AutoScaler

1. <https://help.sap.com/viewer/7472b7d13d5d4862b2b06a730a2df086/Cloud/en-US/45341f37cf6e4738a4b7cd20f18350de.html>
2. <https://blogs.sap.com/2019/01/29/managing-workload-in-sap-cloud-platform-with-autoscaler-service/>

# Access SAP OData Service with No Authentication in UI5 Project

**Requirement:** We need to create a UI5 Application, which is open to public (no login required), this application in turn calls on-premises SAP OData Service

Creating a MTA project with UI5 application and Standalone Approuter, and creating a Connectivity Service in MTA.yaml, we were able to achieve above requirement

