Step 1: Access SAP Business Application Studio

i) Begin by logging into your **SAP BTP Trial Account**.
Navigate to **Instances and Subscriptions** and click on **SAP Business Application Studio (BAS)** to launch the development environment.

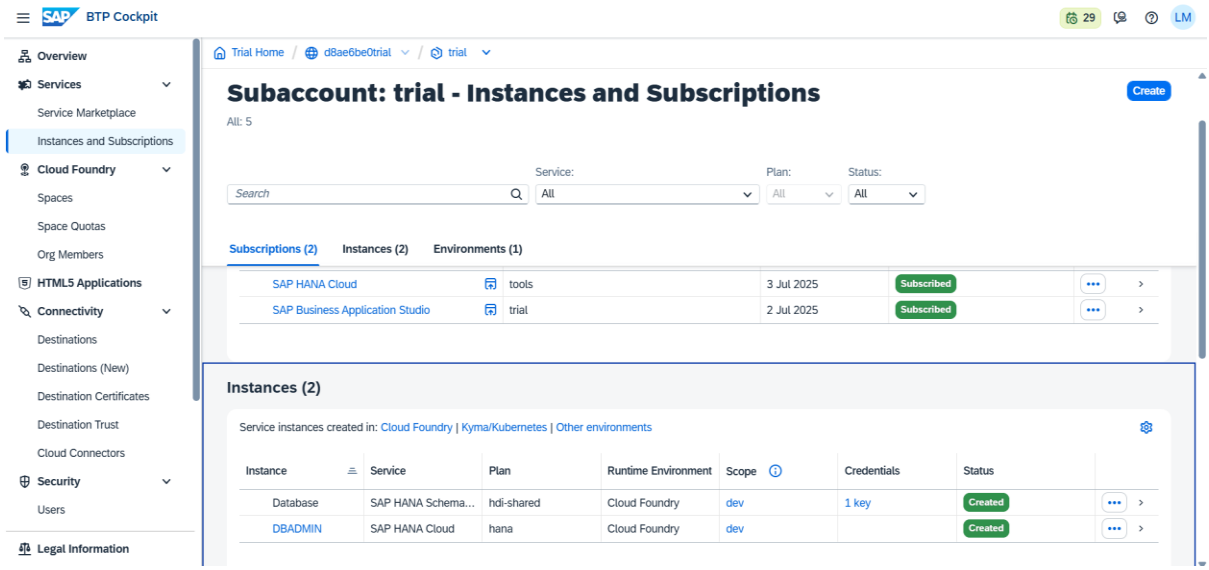


Image 1.1 - Accessing SAP BAS from Instances and Subscriptions in SAP BTP

ii) Once SAP Business Application Studio (BAS) is launched, click on **"Create Dev Space"** to begin setting up your development environment.

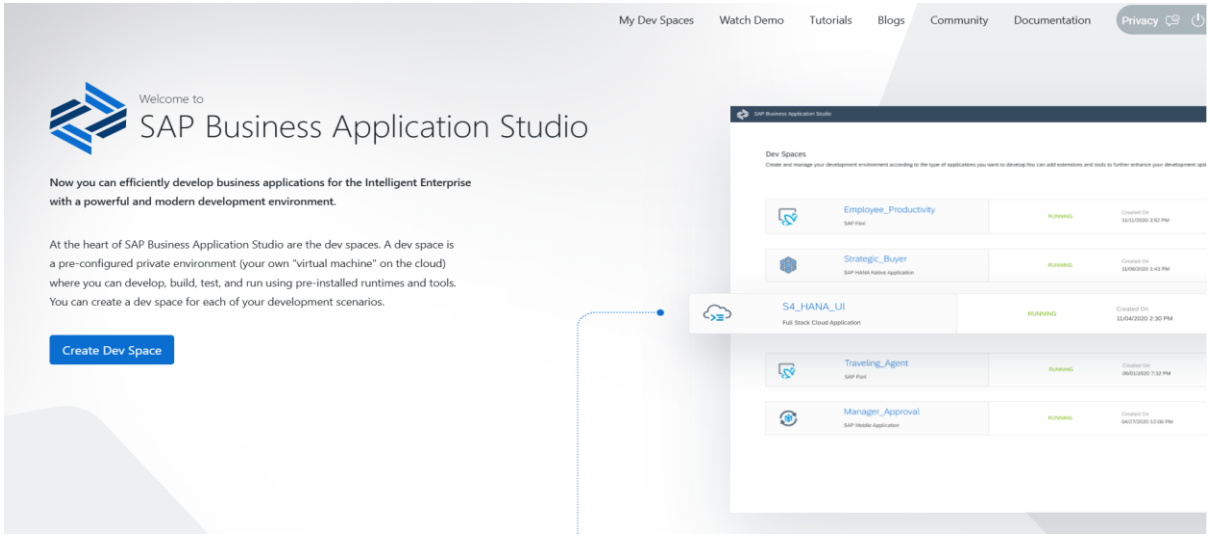


Image 1.2 - Creating a new Dev Space in SAP BAS

Step 2: Configure Your Dev Space

i)Enter a **Dev Space name** (e.g., dev), select **Full Stack Cloud Application** as the application type, and check the box for **Additional SAP Extensions** to enable relevant tools and features.

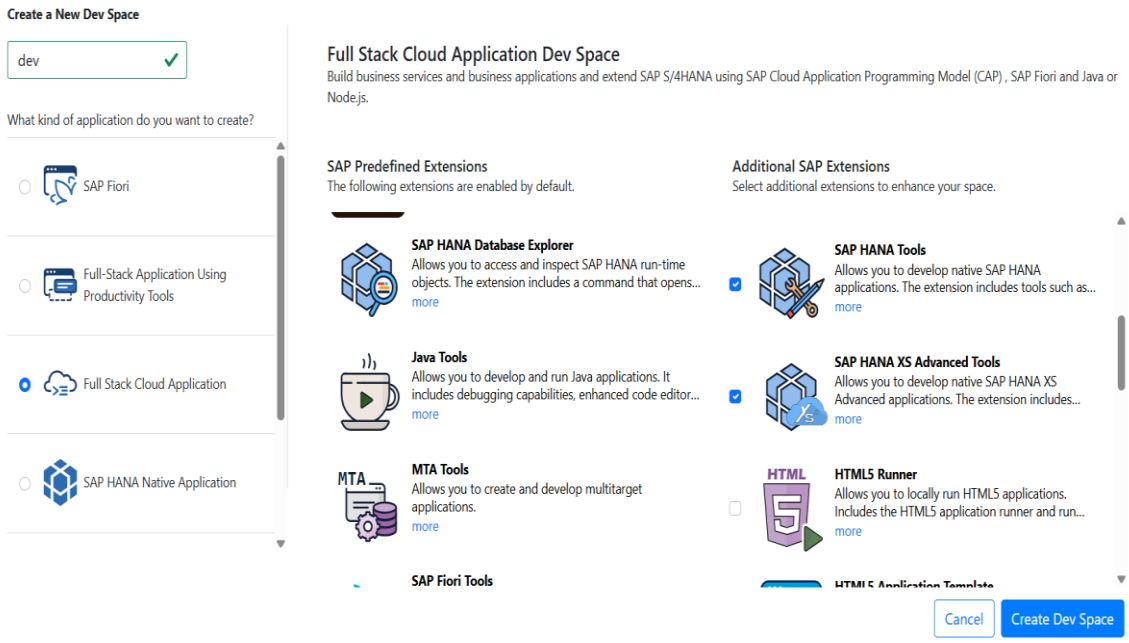
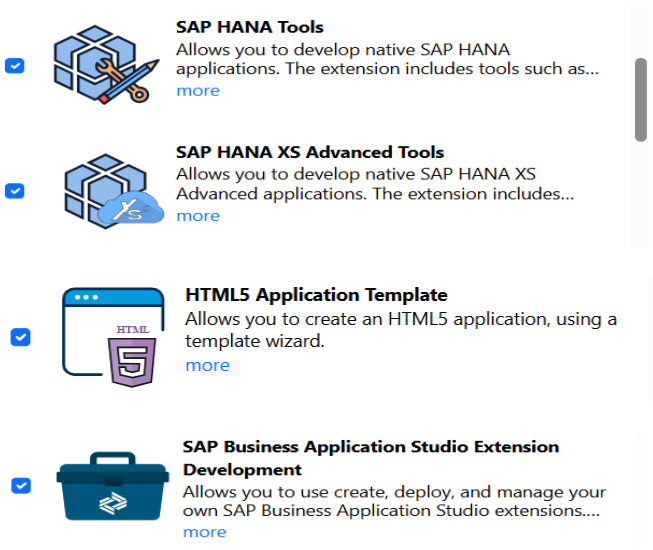


Image 2.1- Naming and configuring the Dev Space in SAP BAS

ii) **Note:** Below are the most essential **Additional SAP Extensions** you should select while creating your Dev Space:



Step 3: Dev Space Creation & Status

Once the Dev Space is created, it will appear with a status of **“STARTING”**. After a few moments, it will change to **“RUNNING”**, indicating that it’s ready for use.

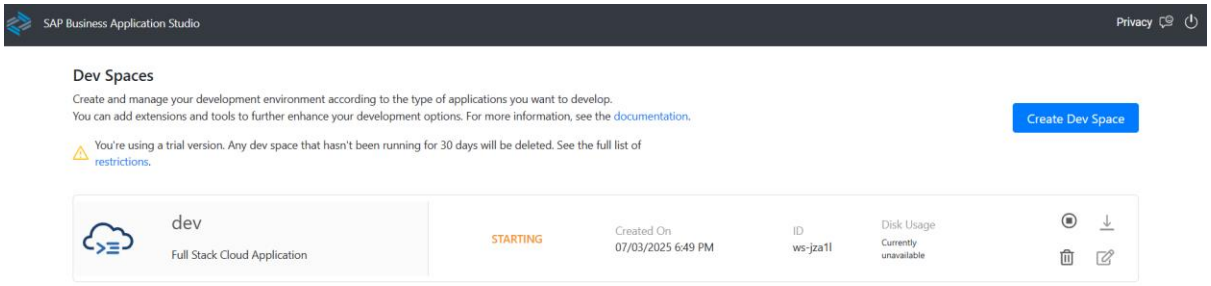


Image 3.1- Dev Space status – STARTING

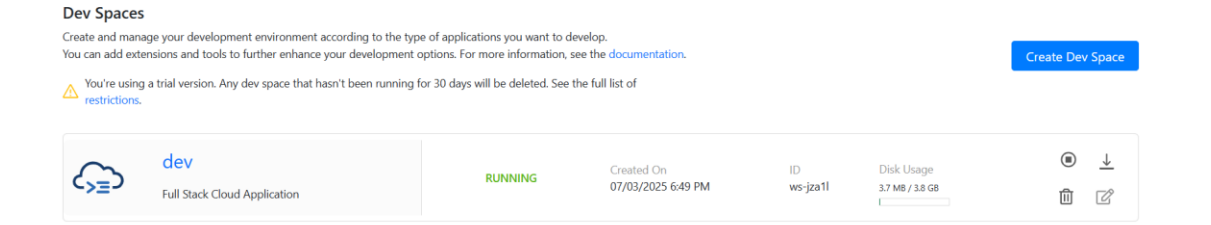


Image 3.2- Dev Space status – RUNNING

Step 4: Start a New Project from Template

i)Once your development environment is ready, click on **“Start from Template”** to create a new project using the guided wizard provided by SAP Business Application Studio.

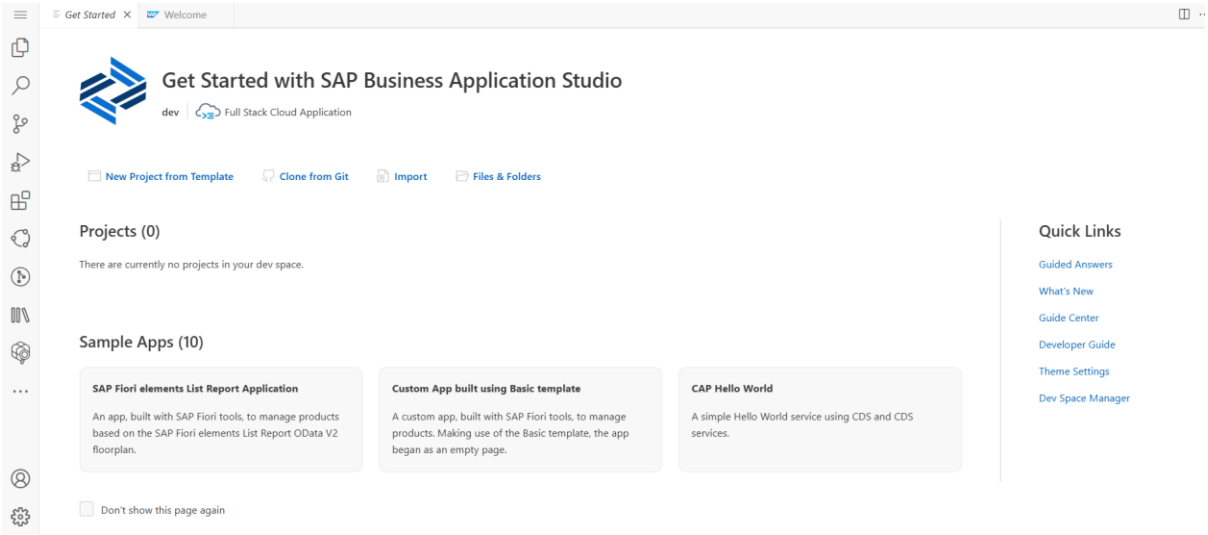


Image 4.1- Launching a new project from template in SAP BAS

ii) From the list of available templates, choose **“CAP Project”** (Cloud Application Programming Model) to start building your full-stack application using the CAP framework.

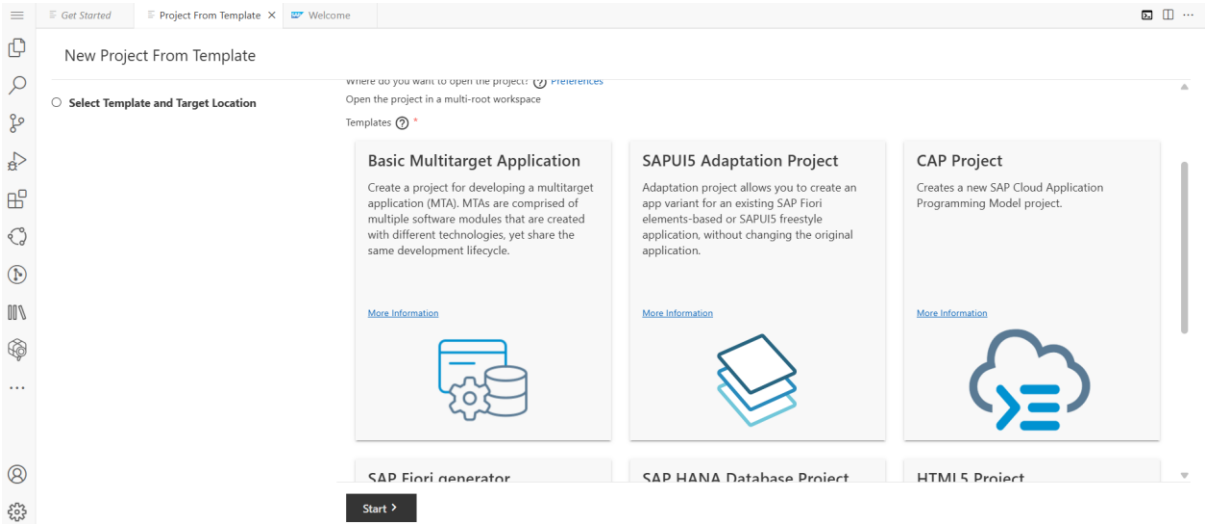


Image 4.2- Selecting the CAP Project template in the template wizard and click on start

Step 5: Configure CAP Project Settings

Enter your **Project Name** (e.g., `project`), select **Java** as the runtime, and choose **SAP HANA Cloud** as the database.

Then, select your preferred **deployment method**, and click **"Finish"** to generate the project.

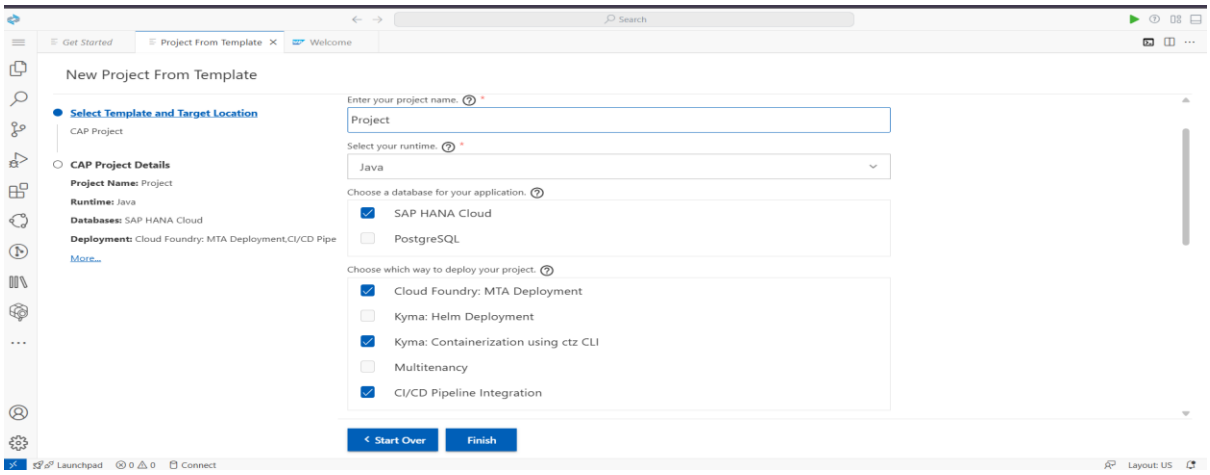
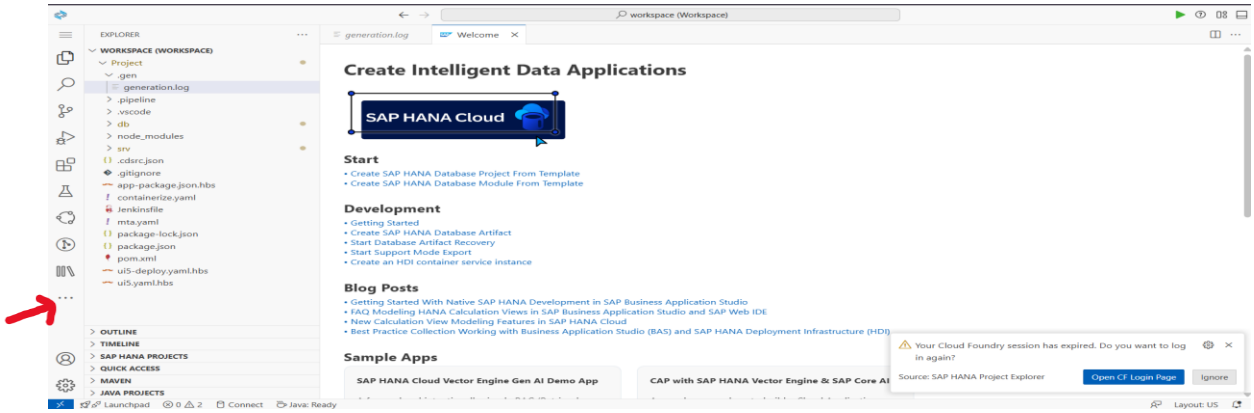


Image 5.1- Configuring CAP project with Java runtime and SAP HANA Cloud

Step 6: Sign In to Cloud Foundry

Once your workspace is created, click on the **three dots (...)** in the left panel (as shown by the arrow), and select **"Sign in to Cloud Foundry"** to authenticate your environment with the SAP BTP Cloud Foundry instance.



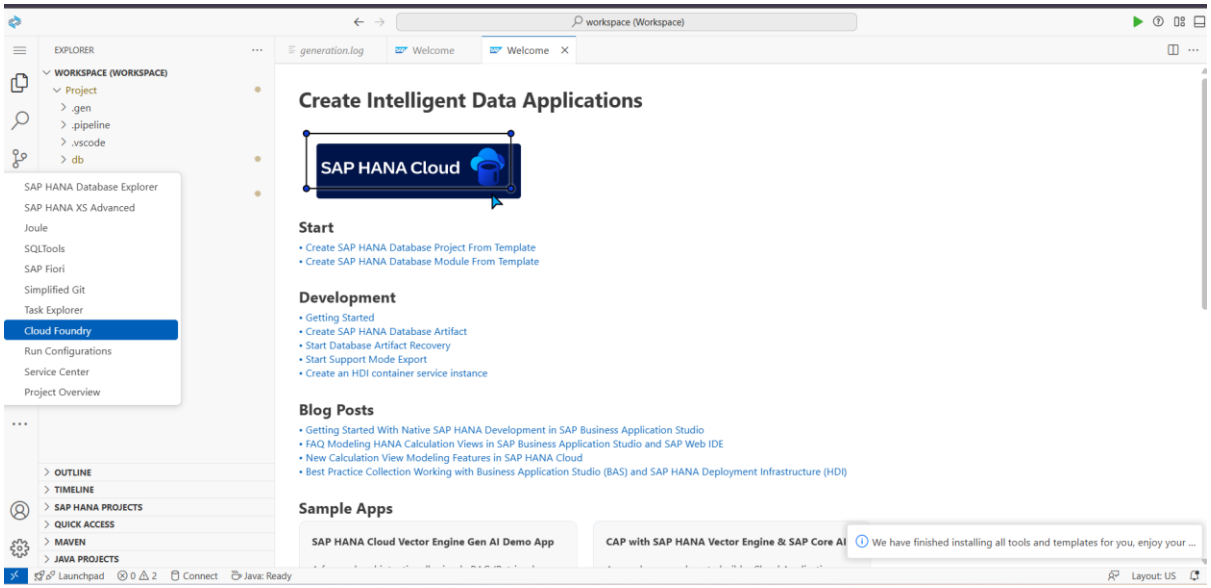


Image 6.2- Click on the three dots and sign in to Cloud Foundry

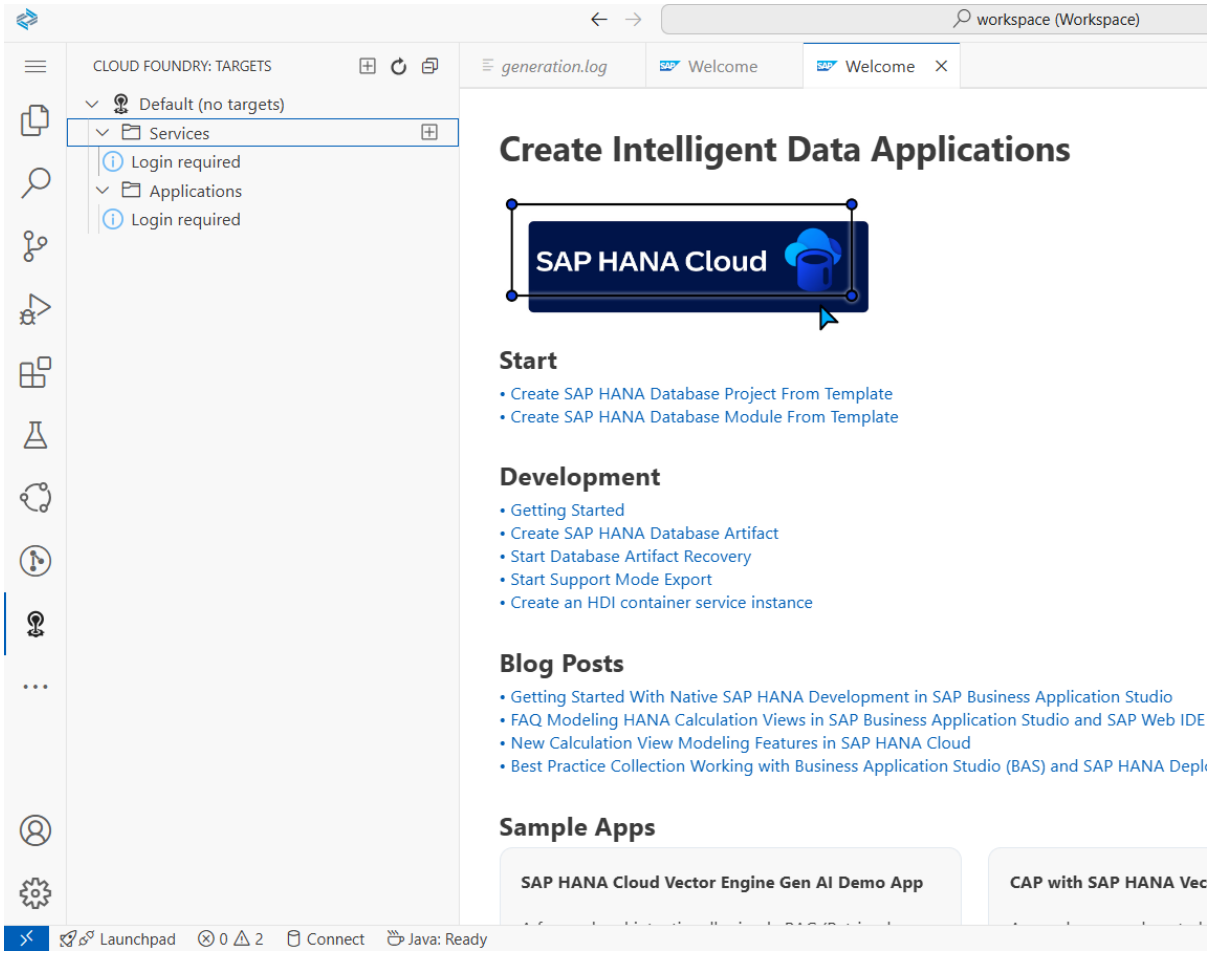


Image 6.3 - Click Login Required

ii) **Authenticate to Cloud Foundry**

Choose your **authentication method** (typically username and password), then enter your **SAP BTP Cloud Foundry credentials** to complete the sign-in process.

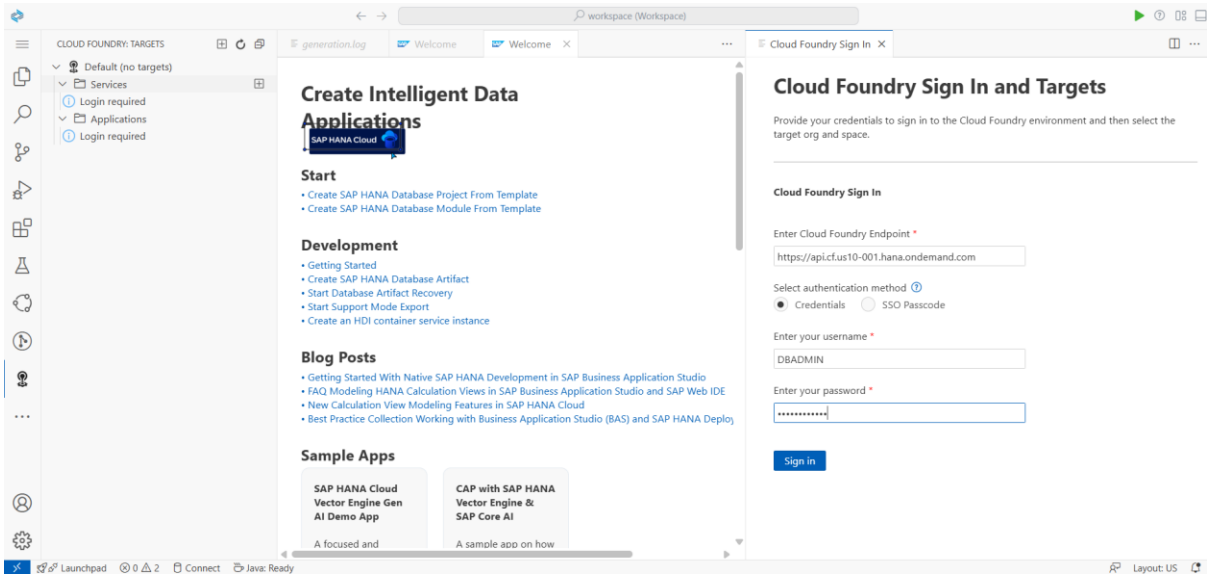


Image 6.4- Entering Cloud Foundry username and password for authentication

◆ Step 7: Clean Up `pom.xml` File

After signing in, navigate to the **`pom.xml`** file inside the **`srv`** folder and **remove the existing plugin** step that is not required for your CAP Java project setup.

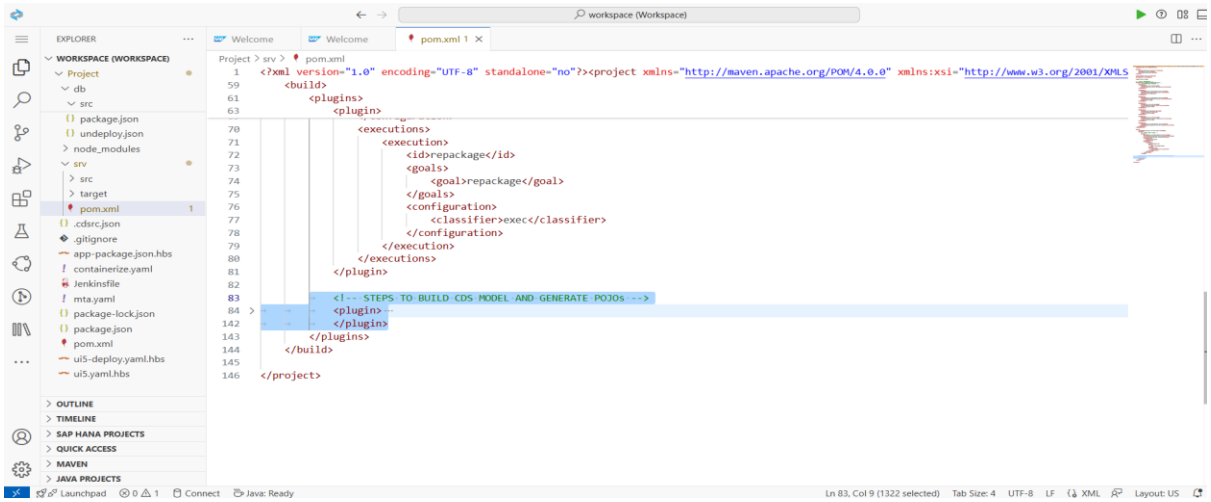


Image 7.1- Removing the unnecessary plugin from `pom.xml` inside `srv`

🔹 Step 8: Temporarily Delete the db Folder

For now, right-click on the **db** folder and select **Delete**. This step is temporary — you can recreate the database module later as needed for your project.

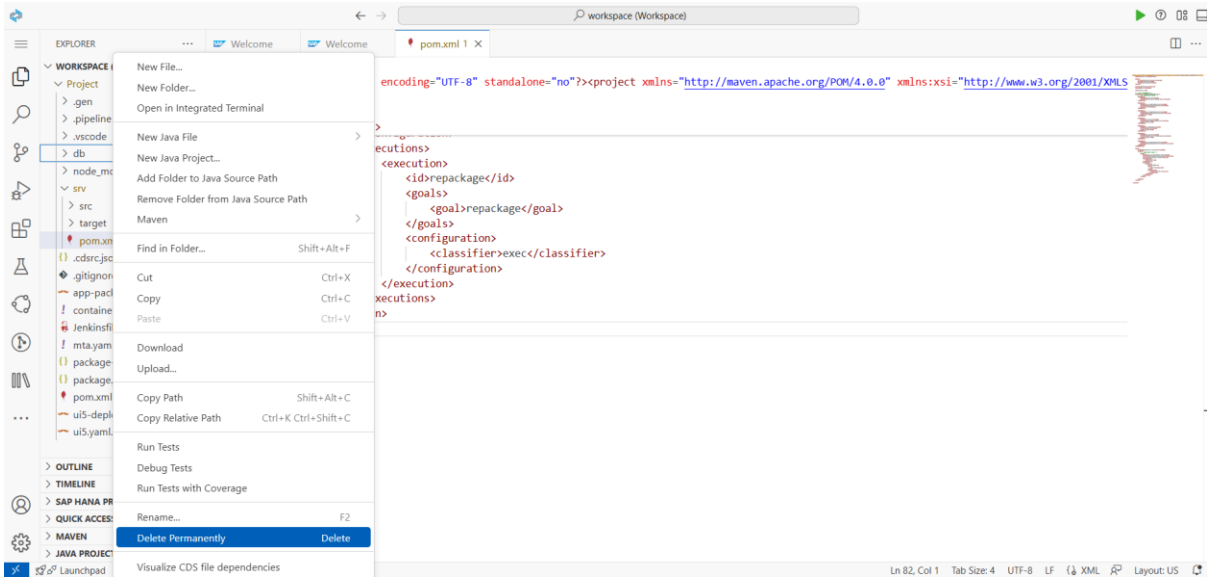


Image 8.1- Deleting the db folder temporarily – will recreate later

🔹 Clear the `application.yaml` File

Navigate to the **`application.yaml`** file and remove its existing content. This will help you start fresh with your own custom configuration later in the project.

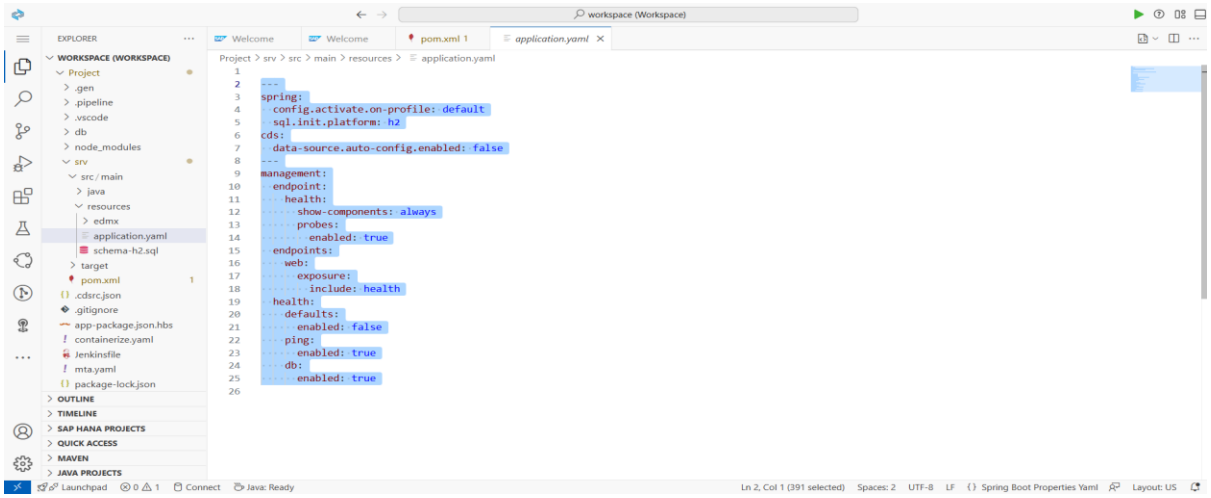


Image 8.2- Clearing contents of `application.yaml` to start with a fresh configuration

🔹 Delete `schema-h2.sql` File

Since it's not required for the current setup, right-click on the **`schema-h2.sql`** file and select **Delete** to remove it from the project.

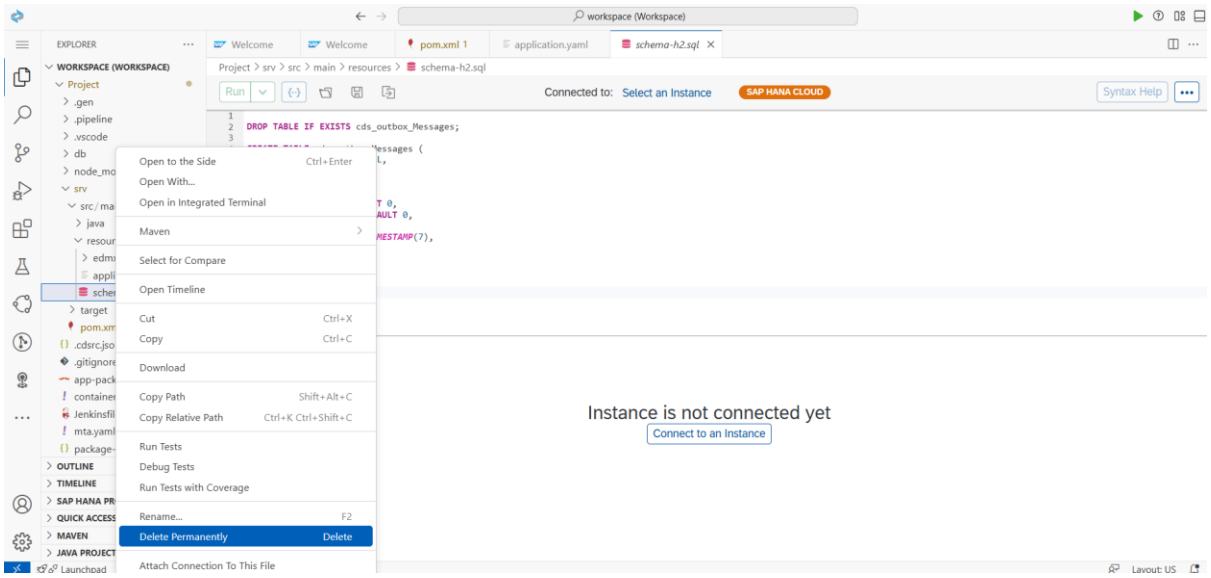


Image 8.3- Deleting the unnecessary `schema-h2.sql` file

🔹 Create MTA Module from Template

Right-click on the **`mta.yaml`** file and select **"Create MTA Module from Template"** to start adding a new module to your CAP project.

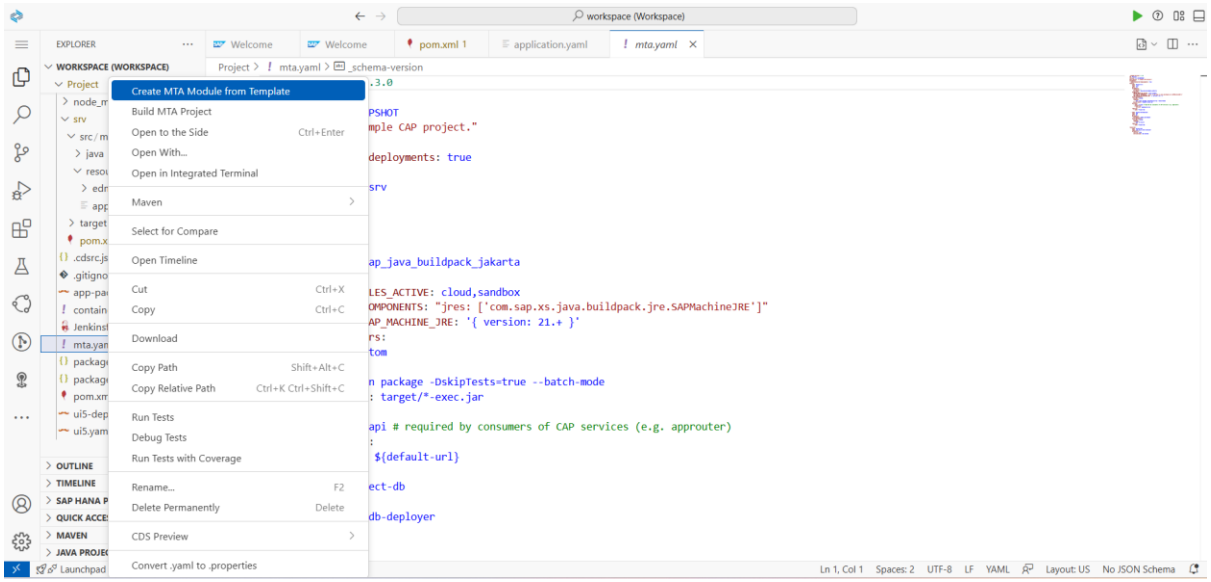


Image 8.4- Creating an MTA module from template by right-clicking on `mta.yaml`

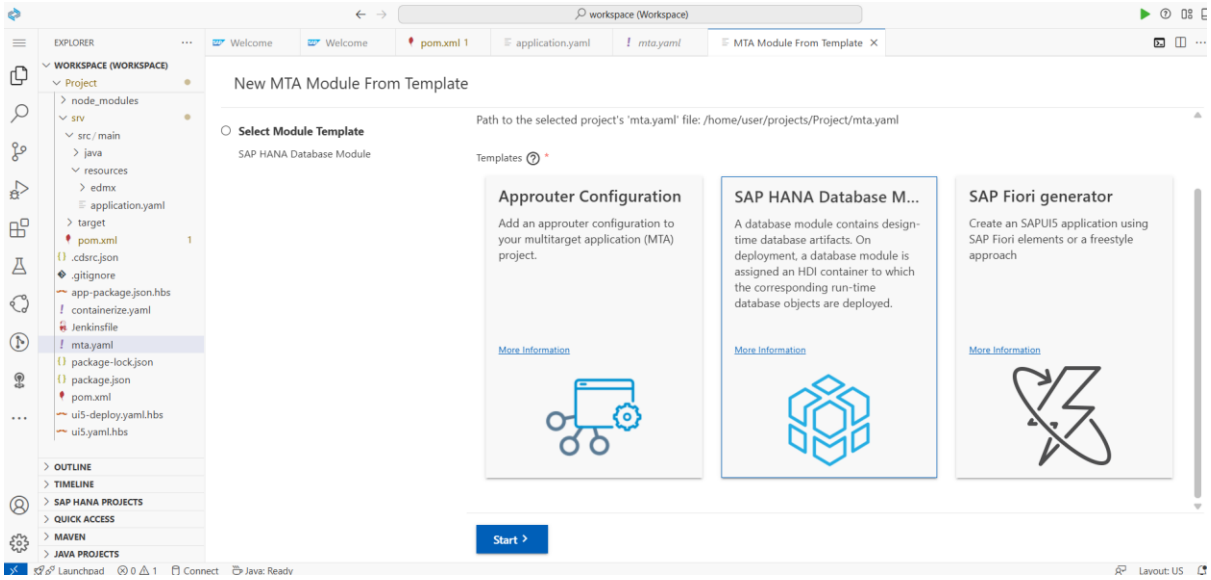


Image 8.5- Select “SAP HANA Database Module” in the templates and click “Start”

- **Configure HANA Module Properties**

Set the basic properties for the new module and provide the **Module Name** as **db** and click next

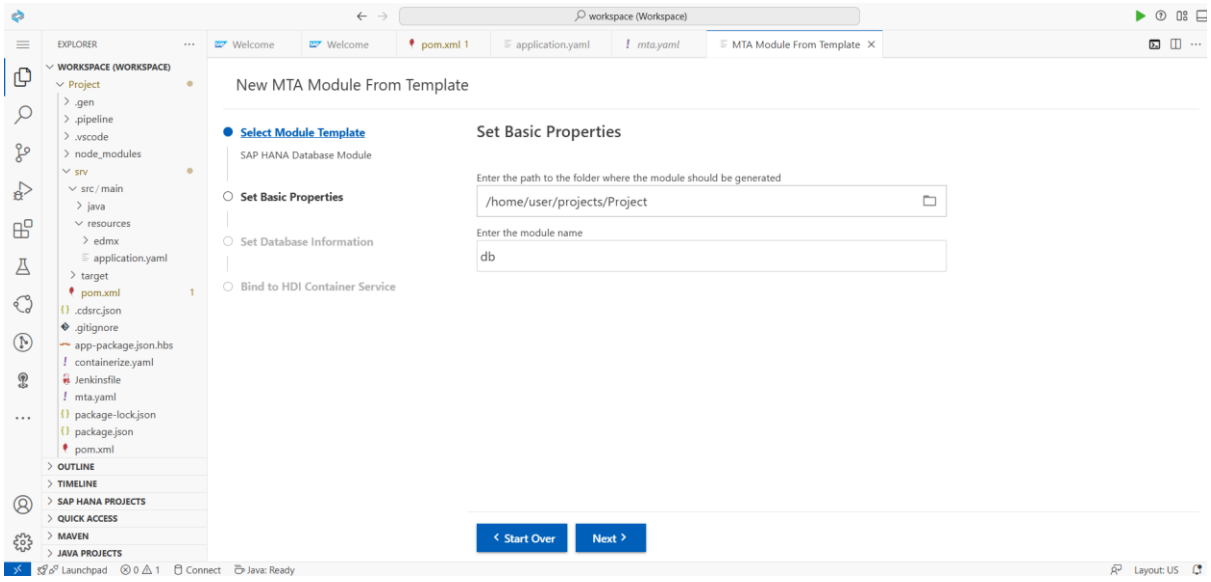


Image 8.6- Setting basic properties and naming the module as db

- **Set the Database Information**

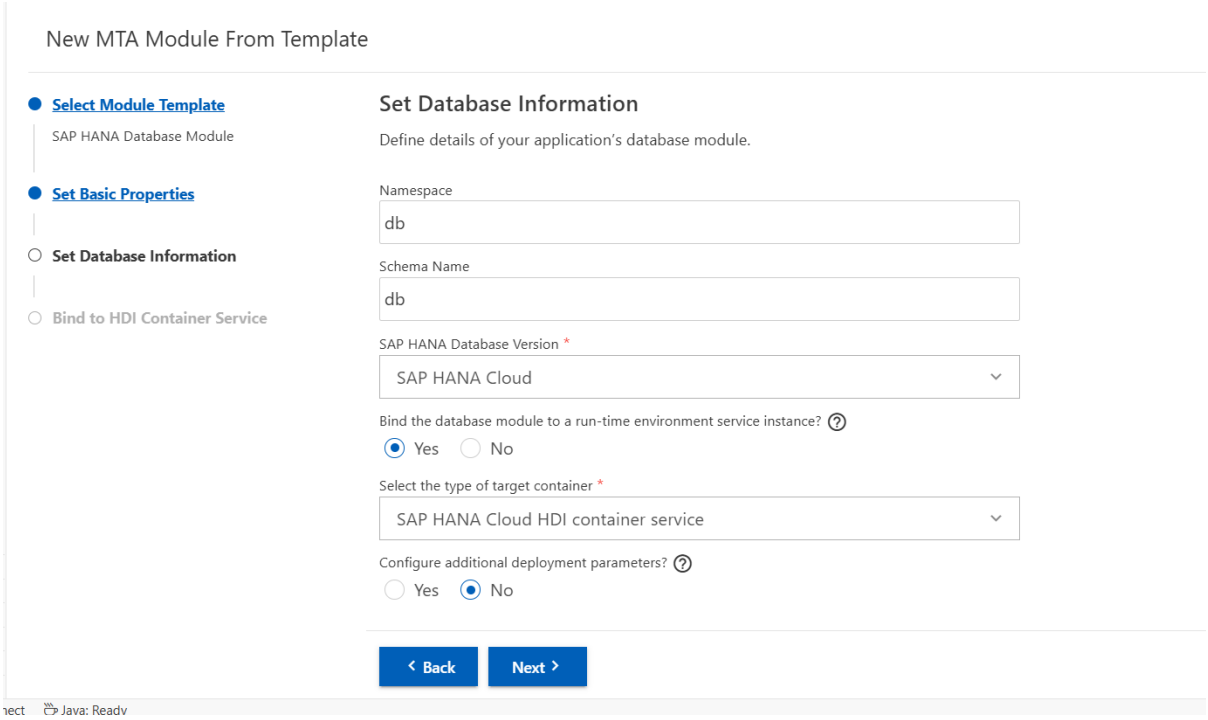


Image 8.7- Setting database information for the SAP HANA module

- **Bind to HDI Container Service**

In this step, **bind the SAP HANA Database Module to an HDI Container Service** to manage your database artifacts securely within the SAP BTP environment.

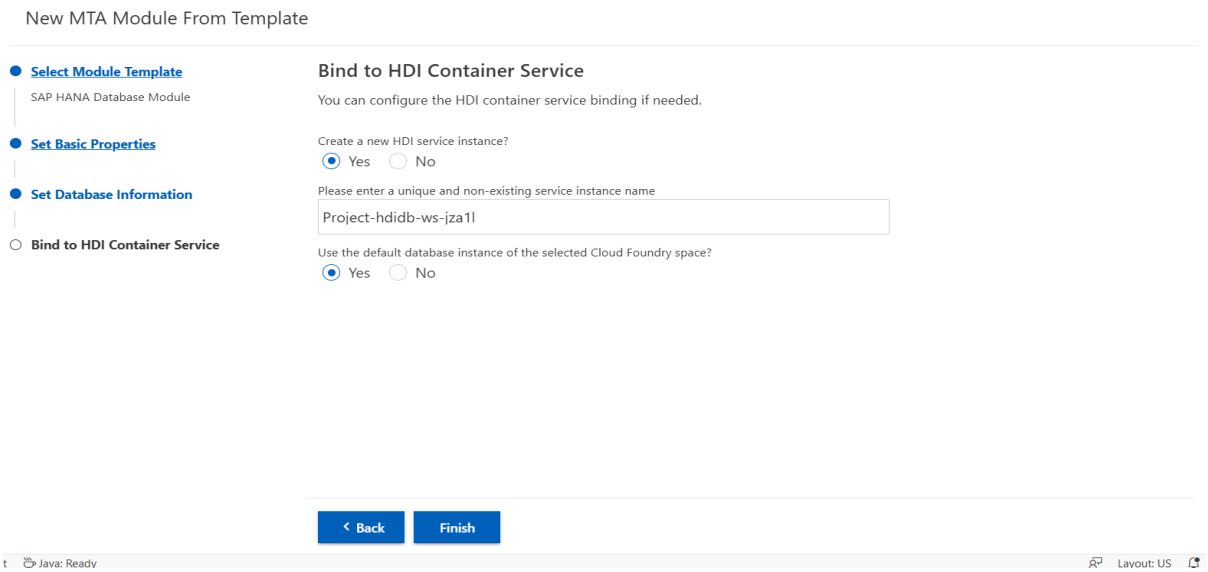


Image 8.8- Binding the module to an HDI Container Service

Step 9: Run the Application

Click on **“Run”** or select **“Run and Debug”** from the left panel to start your CAP application locally in SAP Business Application Studio.

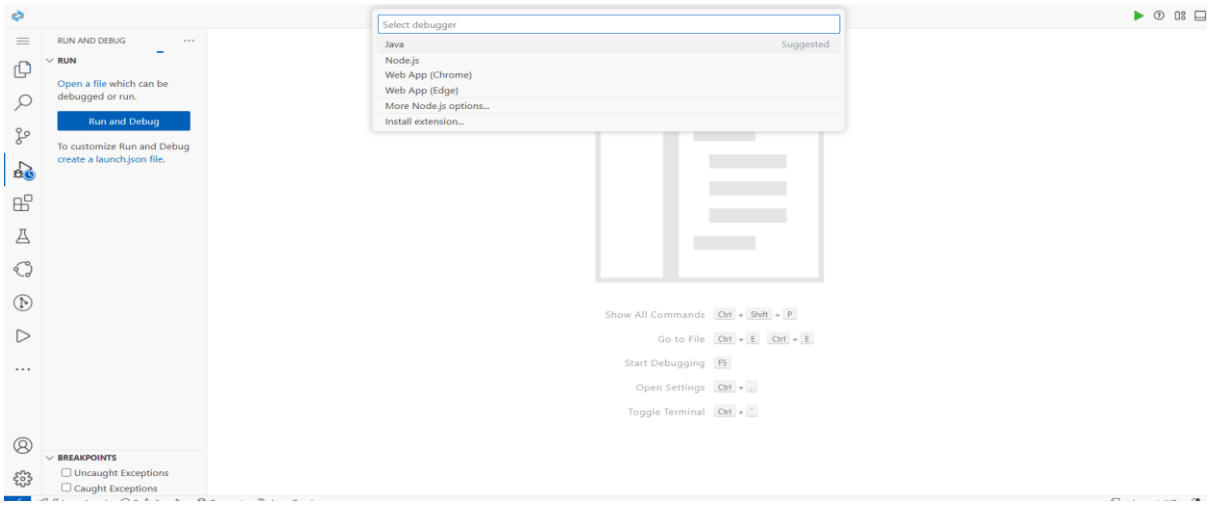


Image 9.1- Running the CAP application using “Run and Debug

- **Select Java and Project**

When prompted, choose **Java** as the runtime environment and select your CAP **project folder** to run the application.

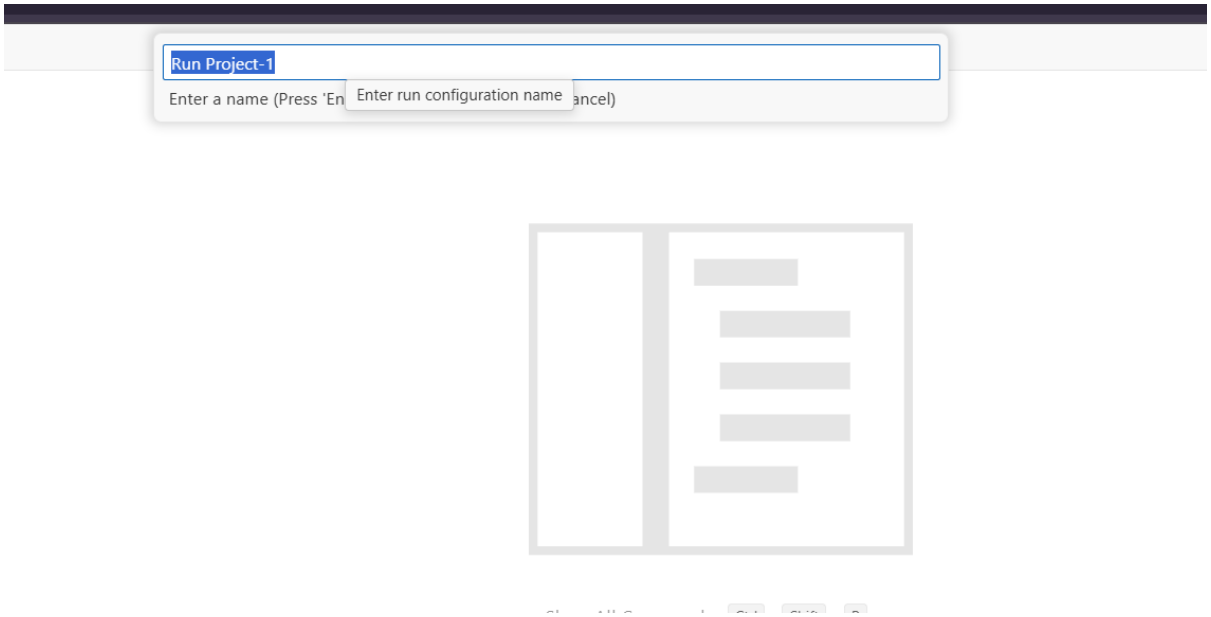


Image 9.2- Selecting Java runtime and the CAP project folder to run the app

- **Service Started Successfully**

Once everything is configured correctly, your **CAP service will start running**, and you’ll see confirmation in the terminal that the application is up and running.

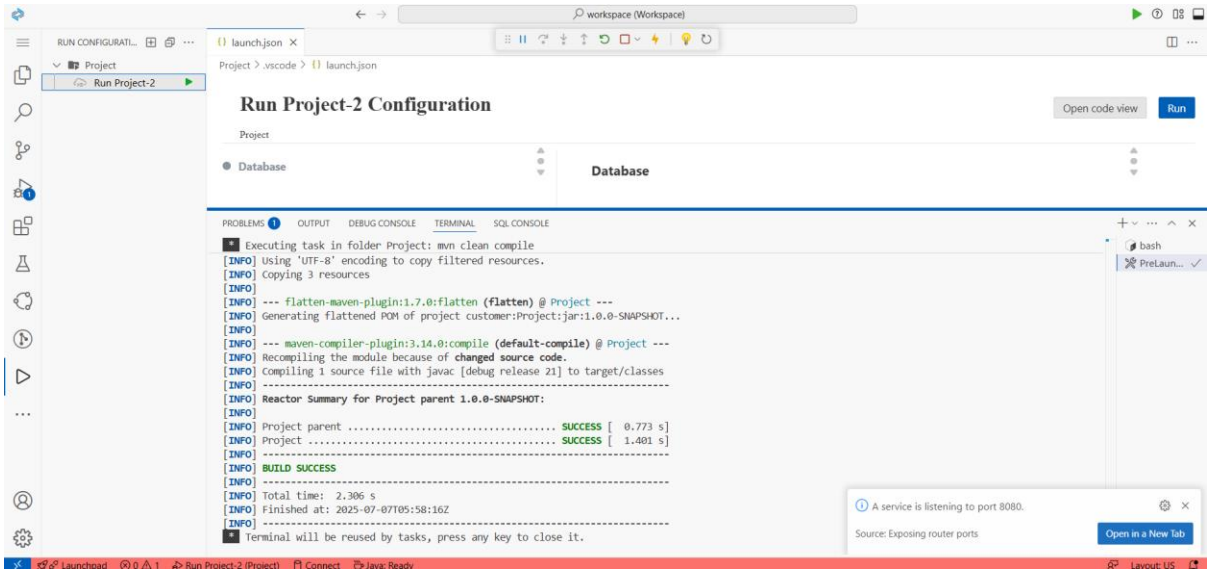


Image 9.3- CAP service started successfully and running in the terminal

◇ Part 2: Database Connection Setup

In this section, we'll configure and connect the CAP project to the SAP HANA Cloud database using the HDI container.

◆ Step 1: Navigate to SAP HANA Cloud

In your **SAP BTP Trial Account**, select **SAP HANA Cloud** to begin setting up your database instance.

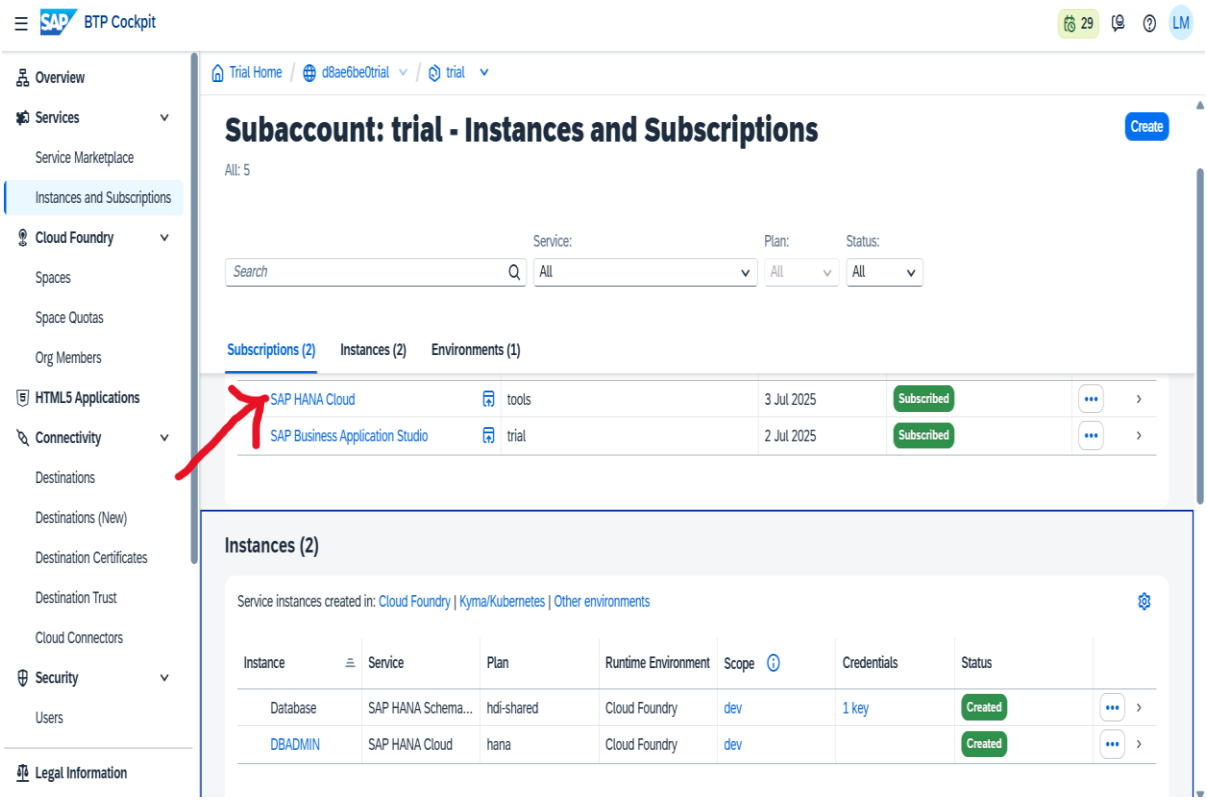


Image 1.1- Selecting SAP HANA Cloud from the trial account navigation menu

◆ Step 2: Ensure the Database is Running

Before proceeding, **make sure your SAP HANA Cloud database instance is in the "Running" state**. This is essential for successful binding and data operations.

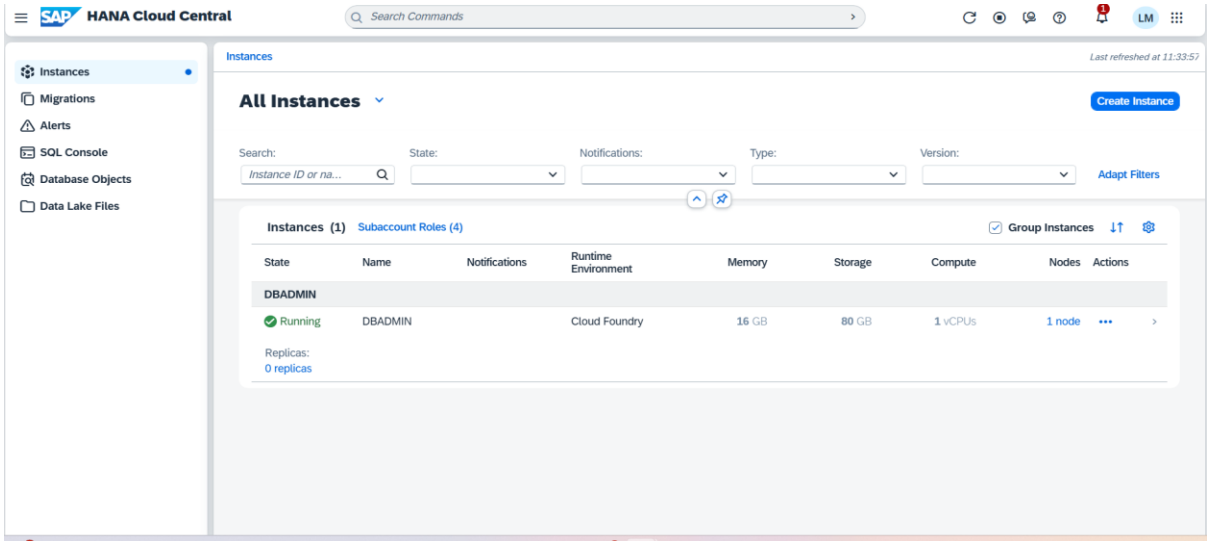


Image 2.1- Verifying that the SAP HANA Cloud database is running

Step 3: Open Database Explorer

Navigate to the **Database Explorer** and click on the **“+” (plus) symbol** to add your HDI container and start exploring or managing database artifacts.

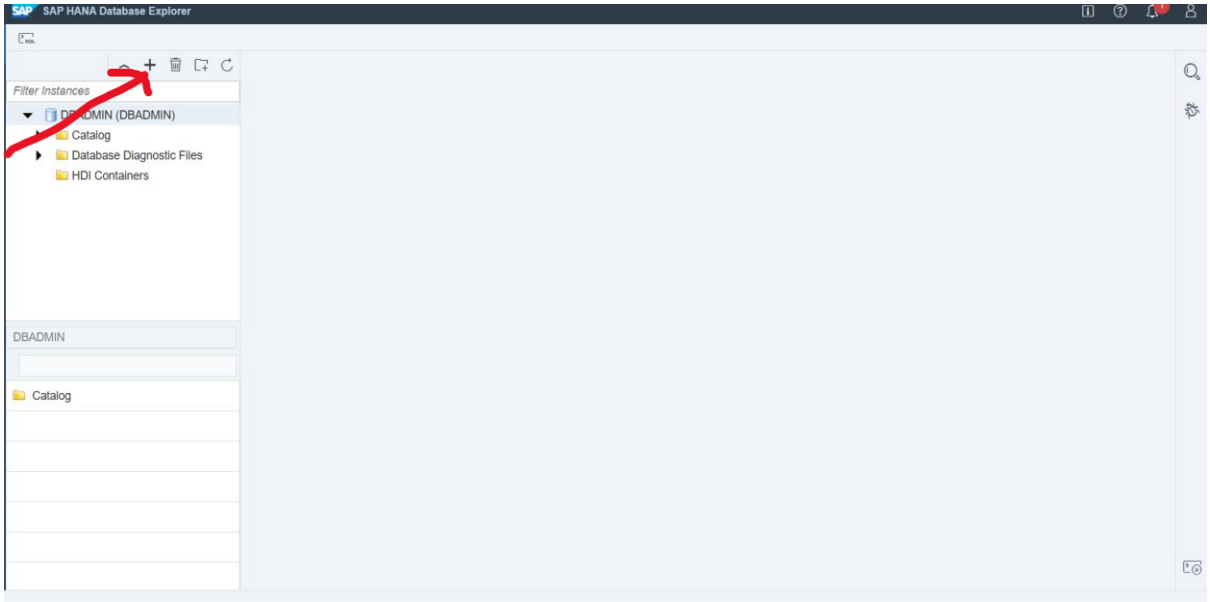


Image 3.1- Opening the Database Console and clicking the “+” icon to add HDI container

Step 4: Select HDI Container Instance Type

In the popup window, set the **Instance Type** to **HDI Container** to connect your CAP

project's database module to the correct SAP HANA schema and select the database so it will automatically generate key .

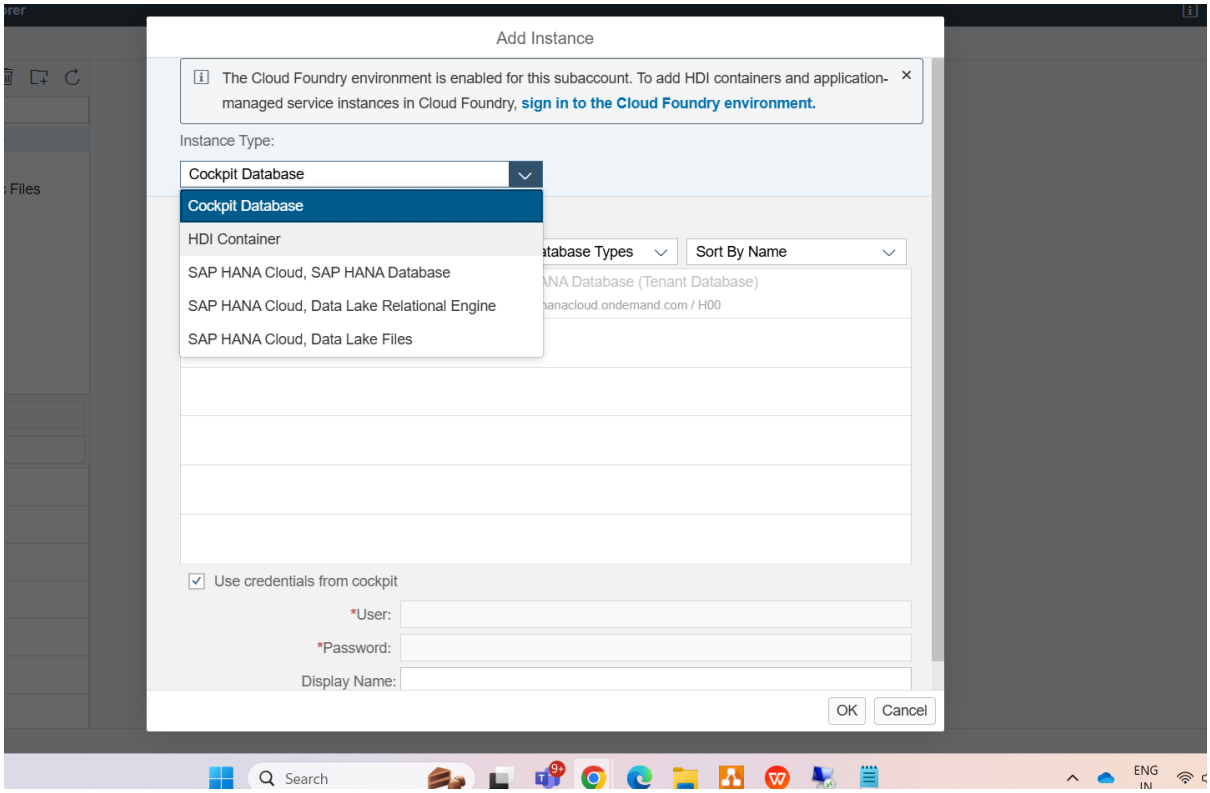


Image 4.1- "HDI Container" as the instance type in the Database Explorer and select the database you created

◆ Step 5: Convert `application.yaml` to `application.properties`

Right-click on the **`application.yaml`** file inside the `resources` folder and select the option **"Convert to Properties"**. This will automatically generate the equivalent `application.properties` file for easier configuration.

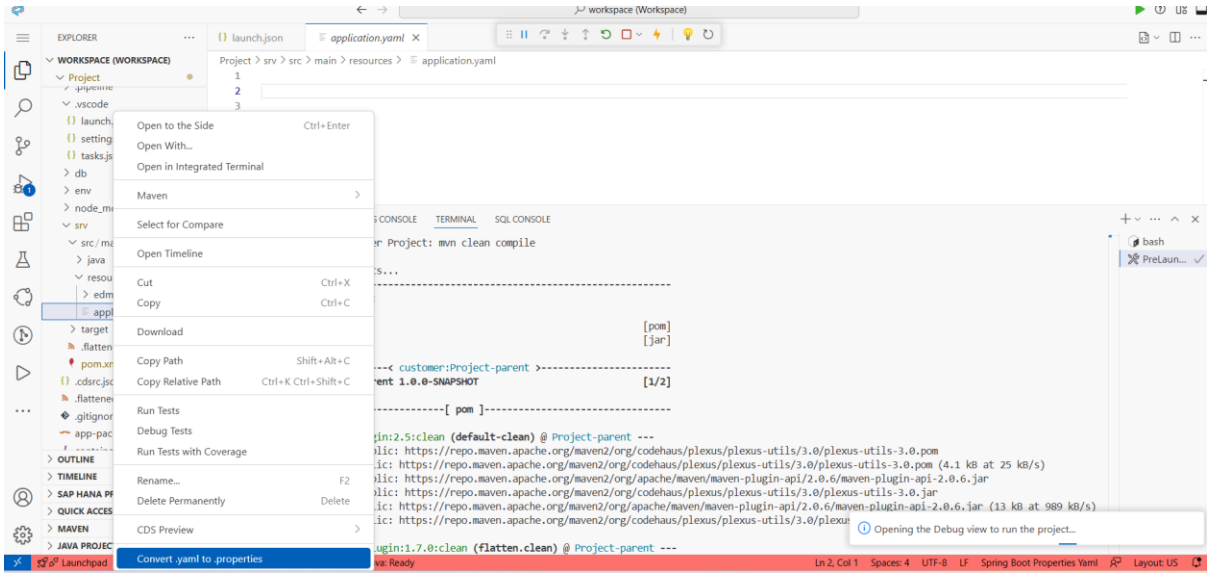


Image 5.1- Right-clicking `application.yaml` and selecting “Convert to Properties”

◆ Step 6: Verify Automatically Created Credentials

Once the **HDI Container** is set in the database, navigate to the **Instance** section. You’ll notice that a **Service Key** is automatically generated under **Credentials**, which will be used for secure connectivity between your CAP application and the database.

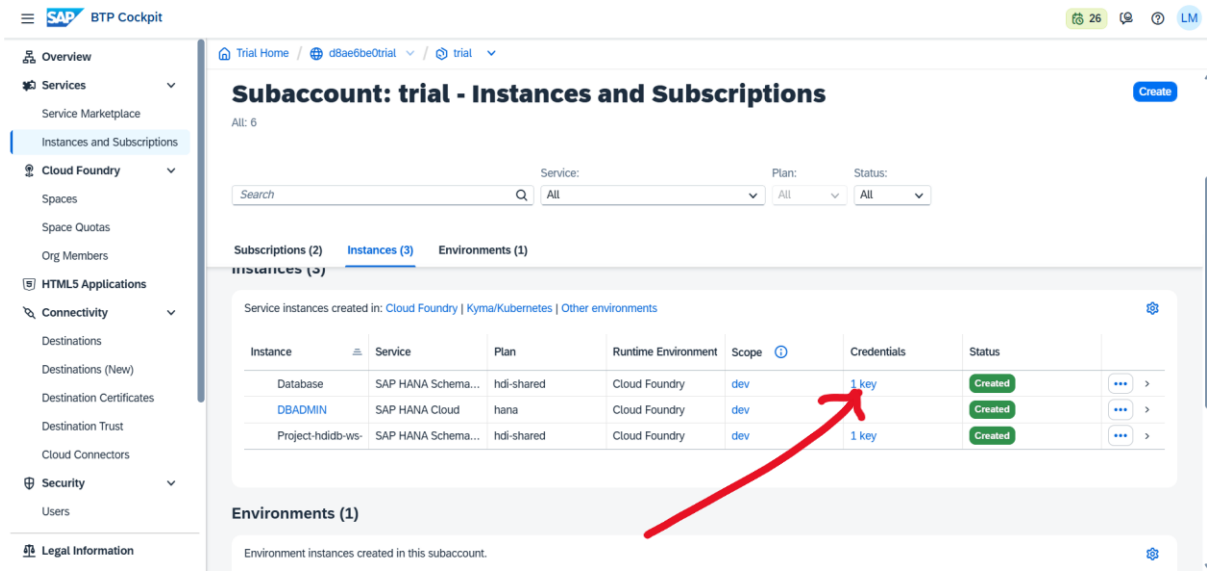


Image 6.1- Viewing the auto-generated key under credentials in the HDI container instance

Step 7: View HDI Container Credentials

Click on the **Service Key** created under your HDI Container instance. You'll be able to see the detailed **credentials**, including the **username**, **password**, **URL**, and other connection details required to integrate with your CAP application.

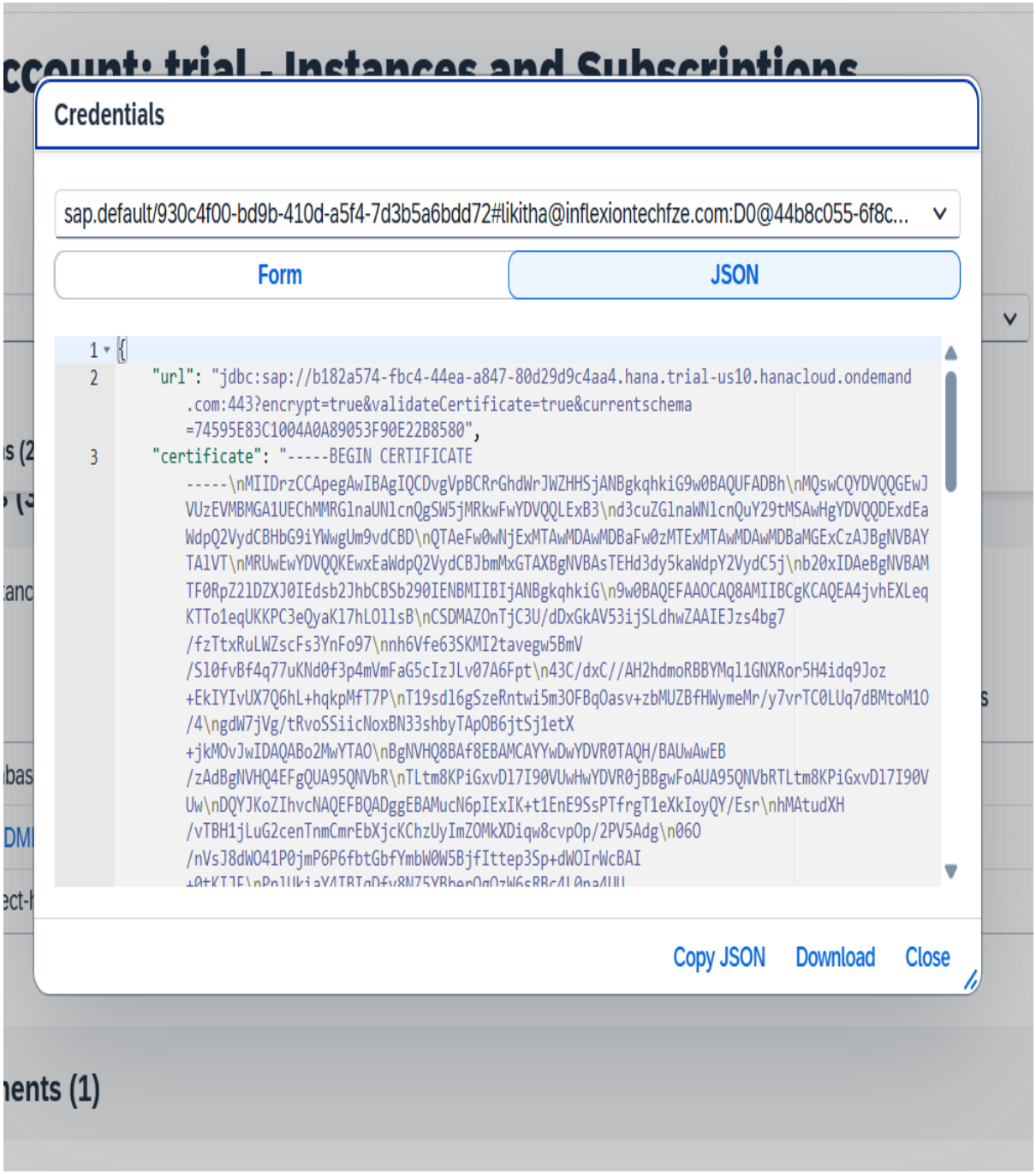


Image 7.1- Viewing detailed credentials from the HDI container service key

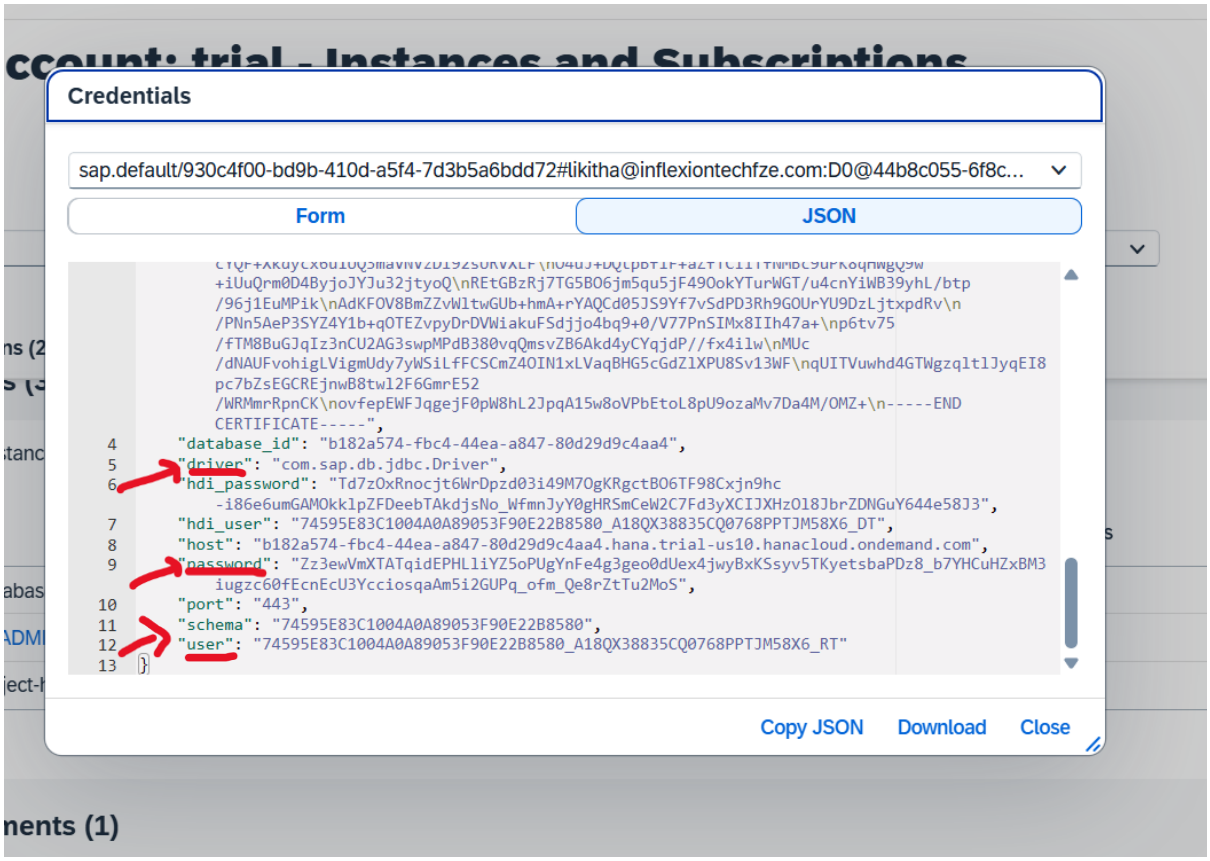


Image 7.2- Viewing detailed credentials from the HDI container service key

🔹 Step 8: Configure Credentials in `application.properties`

Copy the required details (like **URL**, **username**, and **password**) from the HDI container service key and set them in the **`application.properties`** file using the basic syntax format below:

1) spring.datasource.url=your_hdi_ur

2) spring.datasource.username=your_username

3) spring.datasource.password=your_password

4) spring.datasource.driver-class-name=com.sap.db.jdbc.Driver

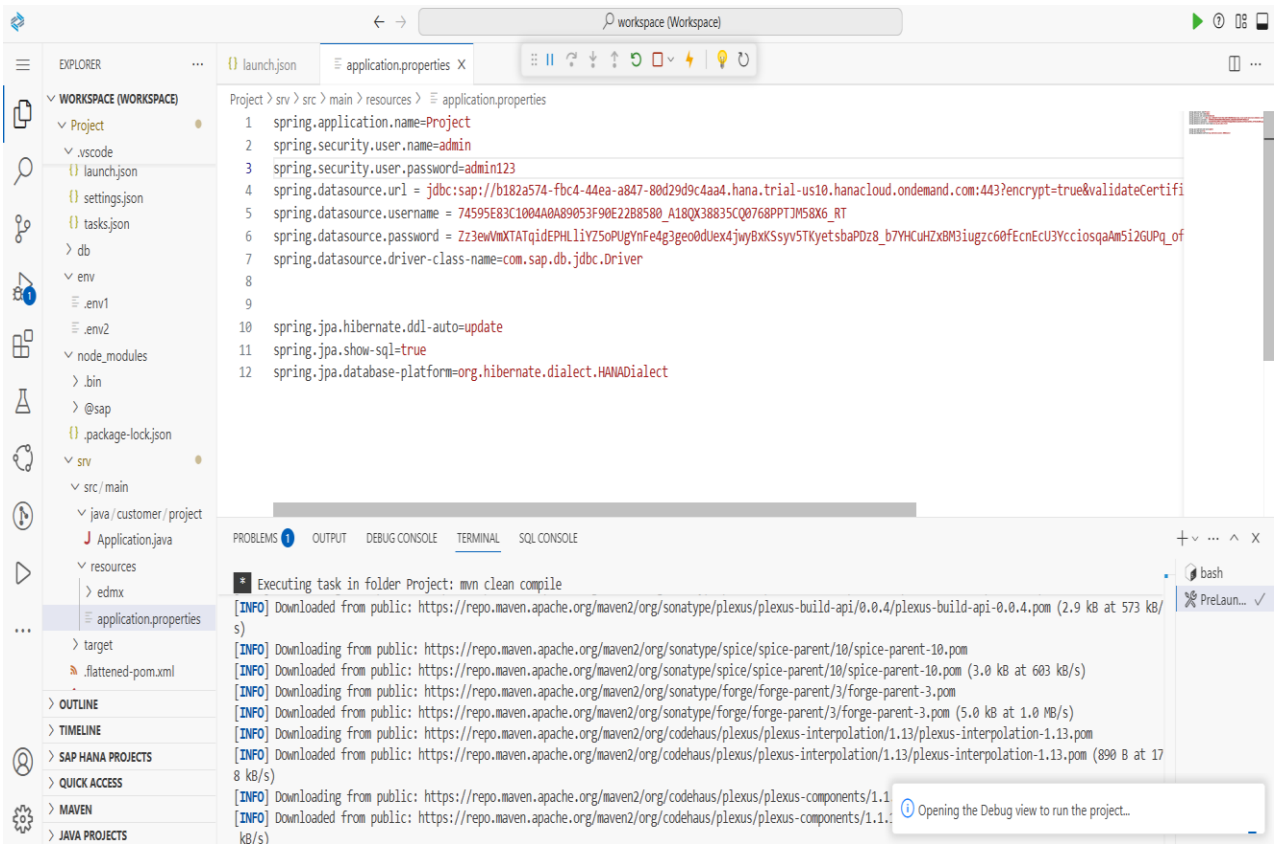


Image 8.1- Setting HDI container credentials in `application.properties`

🔹 Step 9: Add Required Dependencies to pom.xml

To ensure smooth development and handle runtime issues, add the following dependencies to your **pom.xml** file:

1) <!-- Enables live reload and developer convenience tools -->

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>
```

2)<!-- Resolves HANA JDBC connection errors like:
org.hibernate.exception.GenericJDBCException:
unable to obtain isolated JDBC connection [SAP DBTech JDBC: [1890]: HANA Database instance is stopped] -->

```
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
</dependency>
```

3)<!-- Useful for testing Spring Security features -->

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

🔹 Step 10: Add JPA Files – Entity, Repository, Service, and Controller

To structure your CAP Java project following standard Spring Boot practices, create the required **JPA components**:

- **Entity** – Represents your database table.
- **Repository** – Extends `JpaRepository` for CRUD operations.
- **Service** – Contains business logic.
- **Controller** – Exposes REST APIs.

These components help in building a clean and maintainable backend using Spring Data JPA.

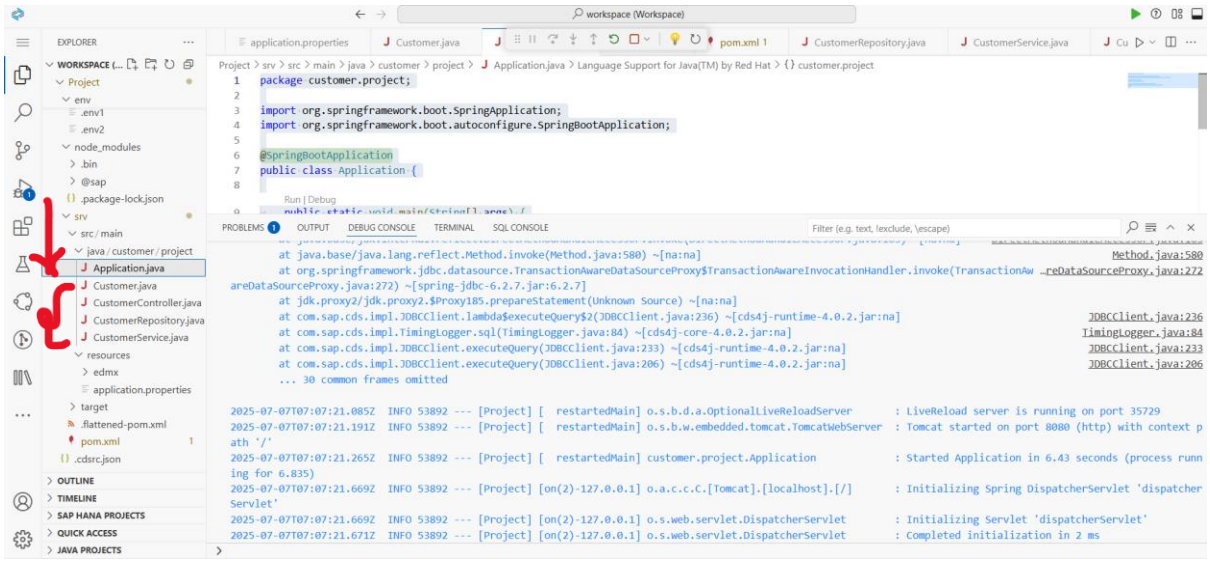


Image 10.1- Adding Entity, Repository, Service, and Controller layers for the CAP project

◆ Step 11: Auto-Creation of Tables from Entity Classes

When you run the application, the **JPA Entity classes** will automatically generate corresponding **tables in the SAP HANA Cloud database**.

To enable this behavior, make sure to include the following property in your **`application.properties`** file:

spring.jpa.hibernate.ddl-auto=update

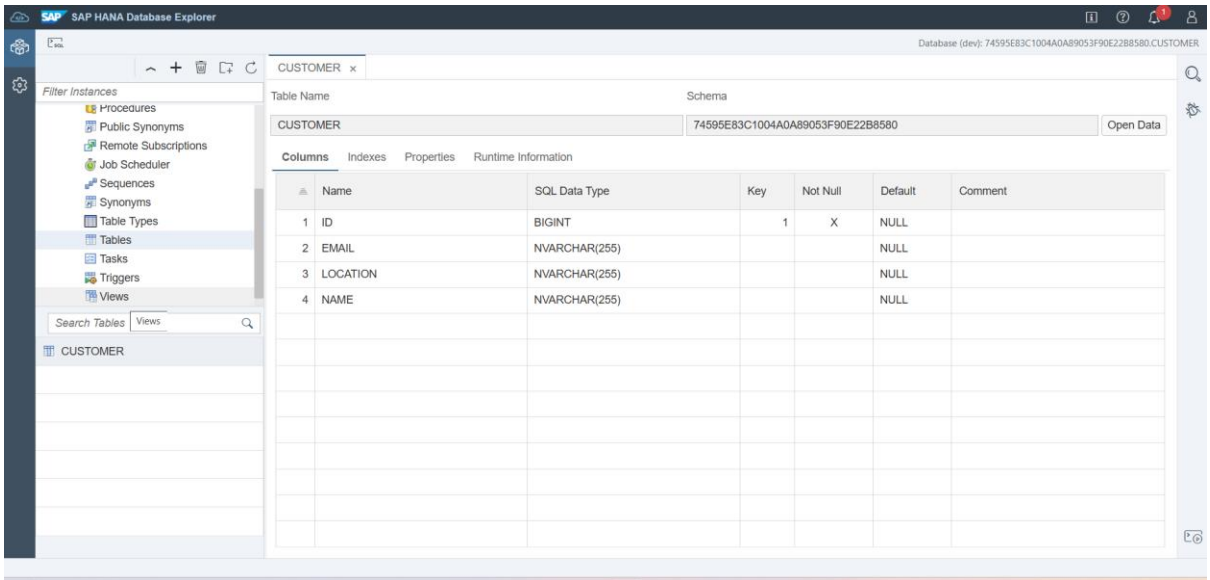


Image 11.1- Caption: Auto-creating tables in SAP HANA from JPA entity classes using `spring.jpa.hibernate.ddl-auto=update`

◆ Part 3: Creation of `webapp` Folder (UI Layer)

In this part, we'll create the **`webapp`** folder which serves as the **UI layer** of your CAP application, typically used to build SAP Fiori or other frontend modules.

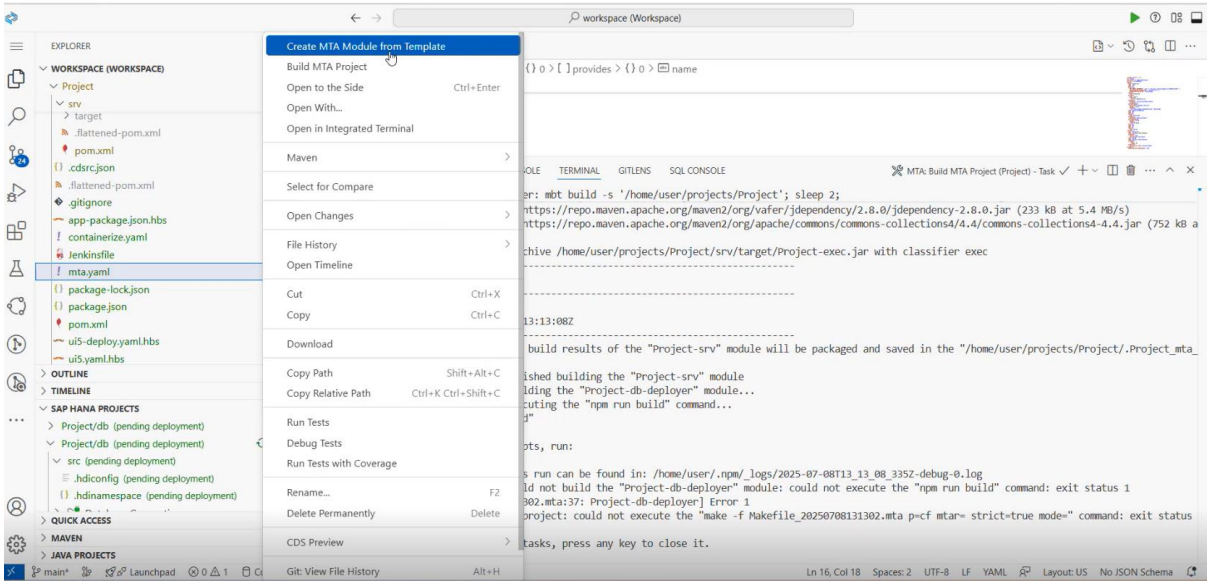


Image 1- click on `mta.yaml` and select "Create MTA Module from Template"

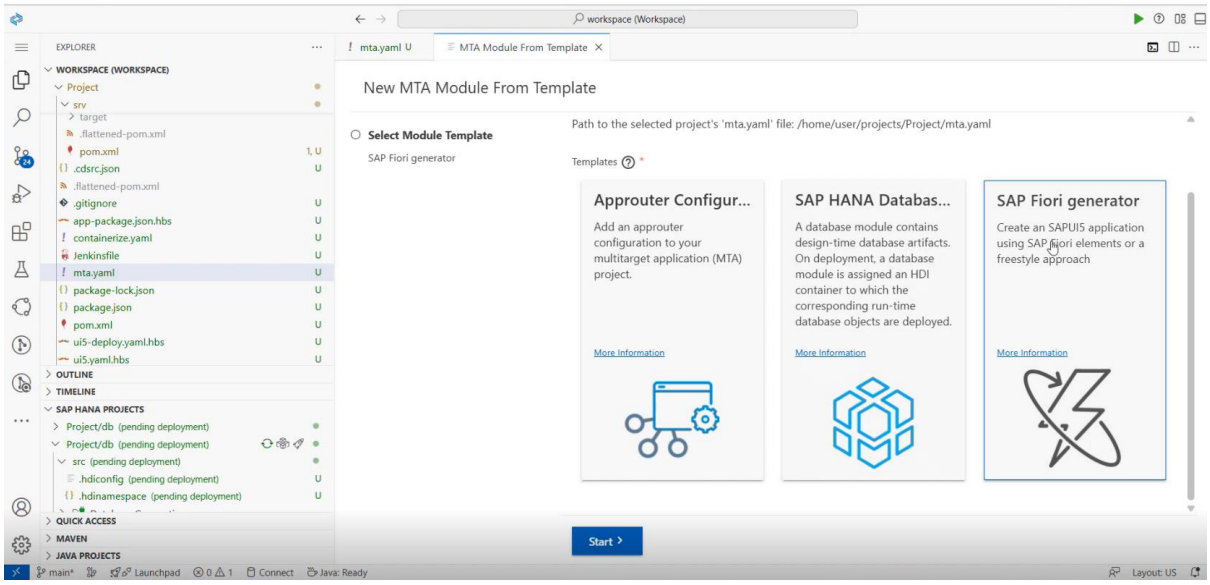


Image 2- Select "SAP Fiori Application" from the list of available templates

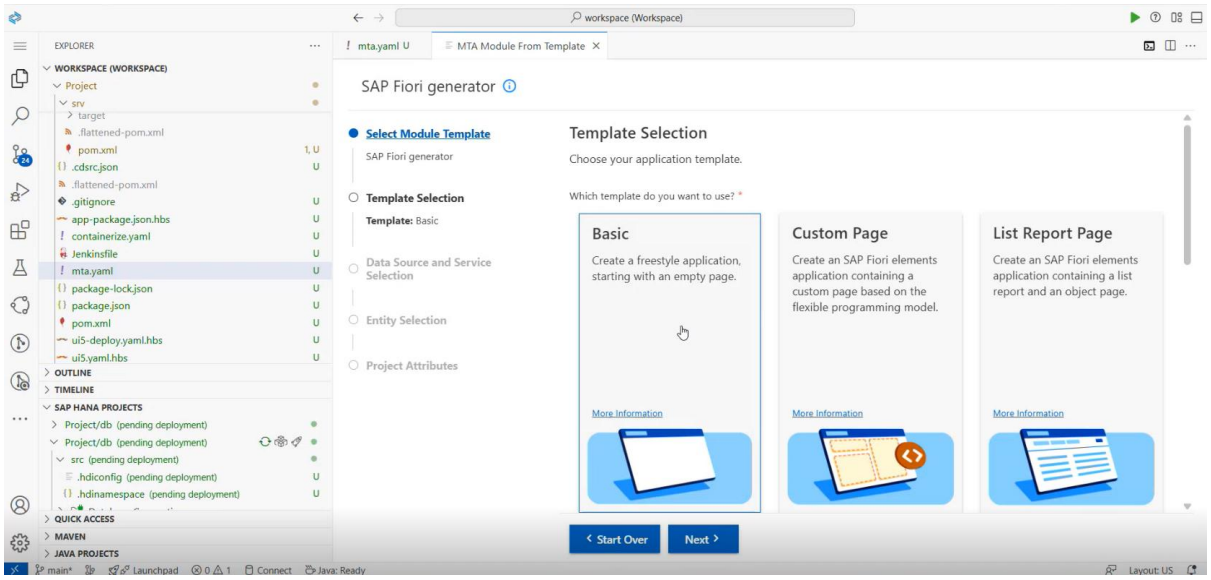


Image 3- Select the “Basic” template for the SAP Fiori application

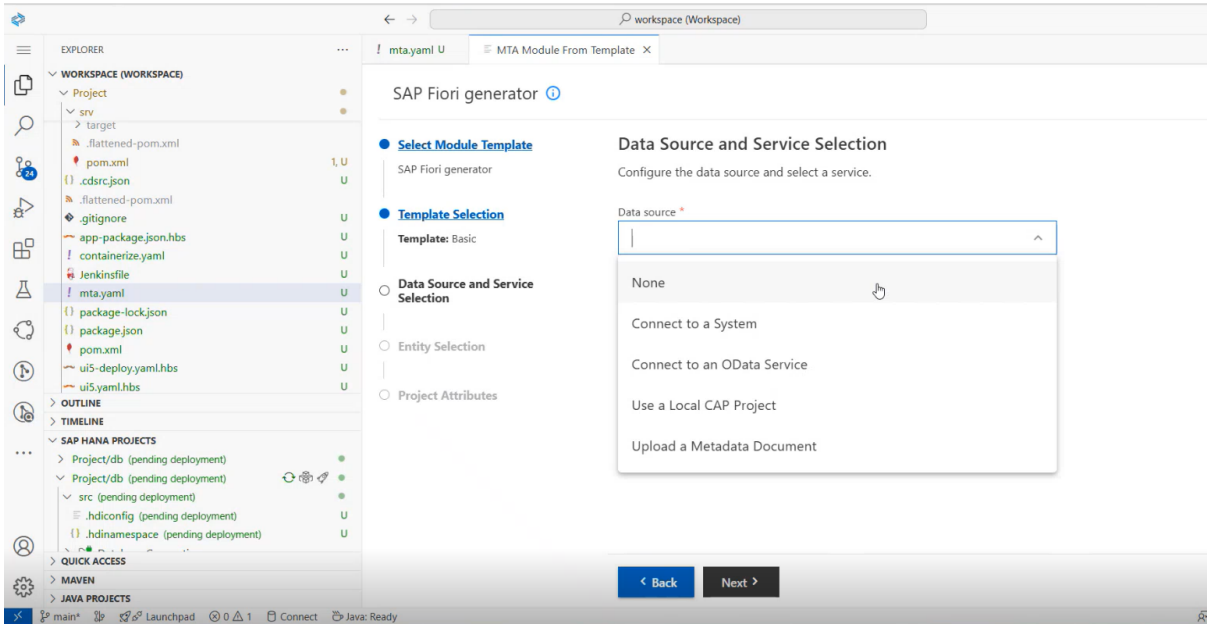


Image 4- Select “None” as the data source and click “Next”

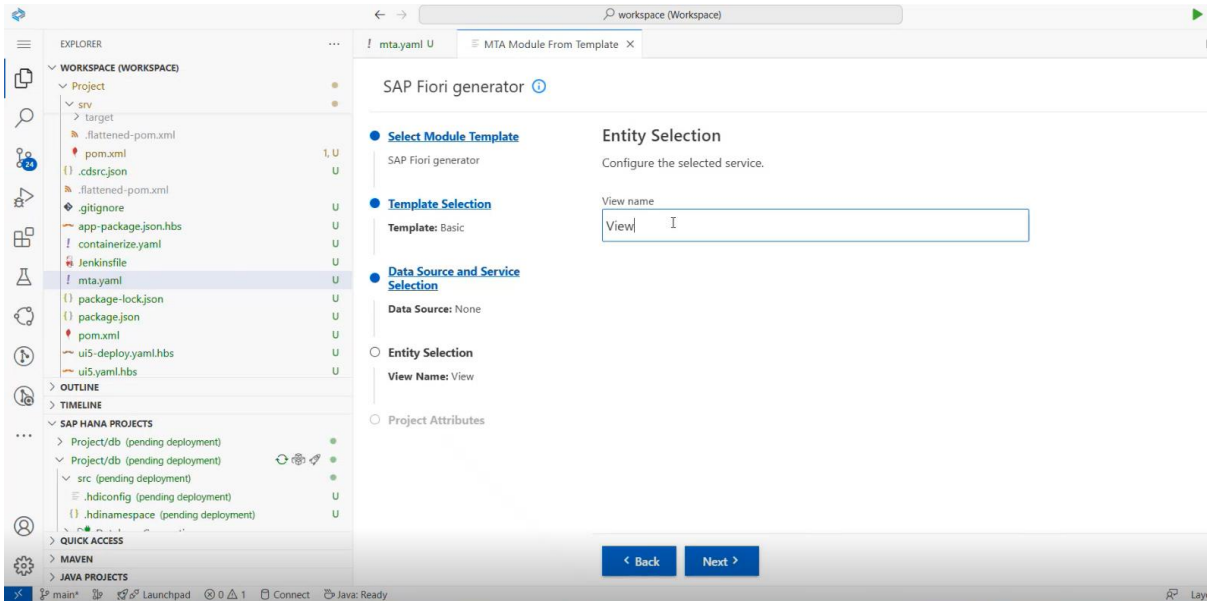


Image 5- Enter the view name as `view` and proceed

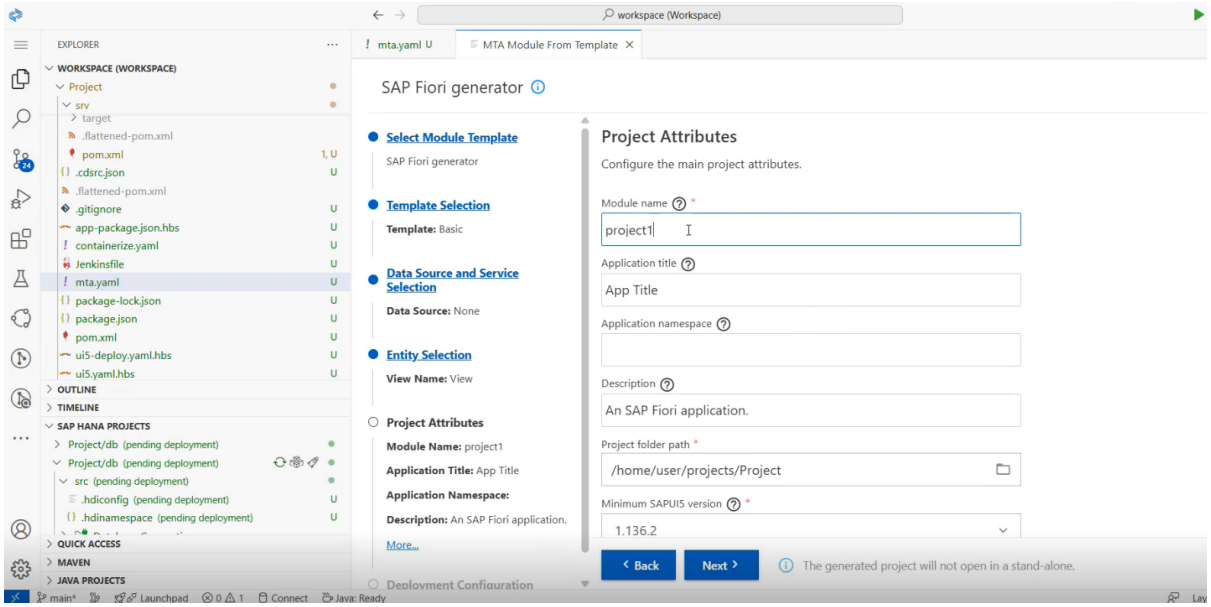


Image 6- Click “Next” to continue with the Fiori module creation

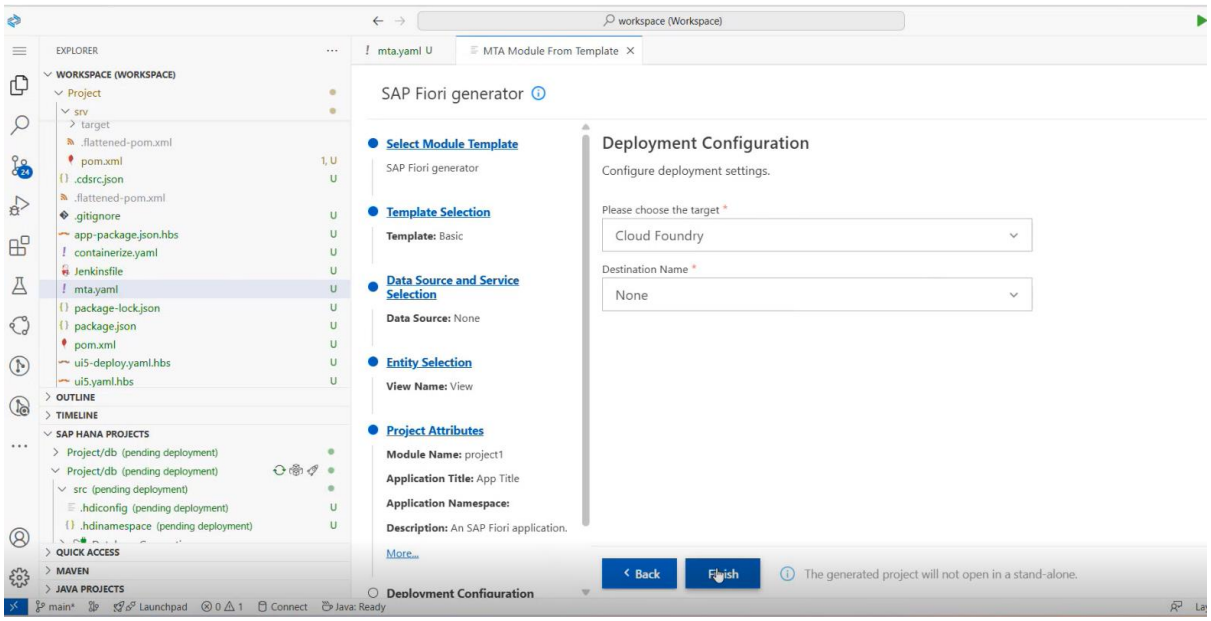


Image 7- Click “Next” to continue with the Fiori module creation

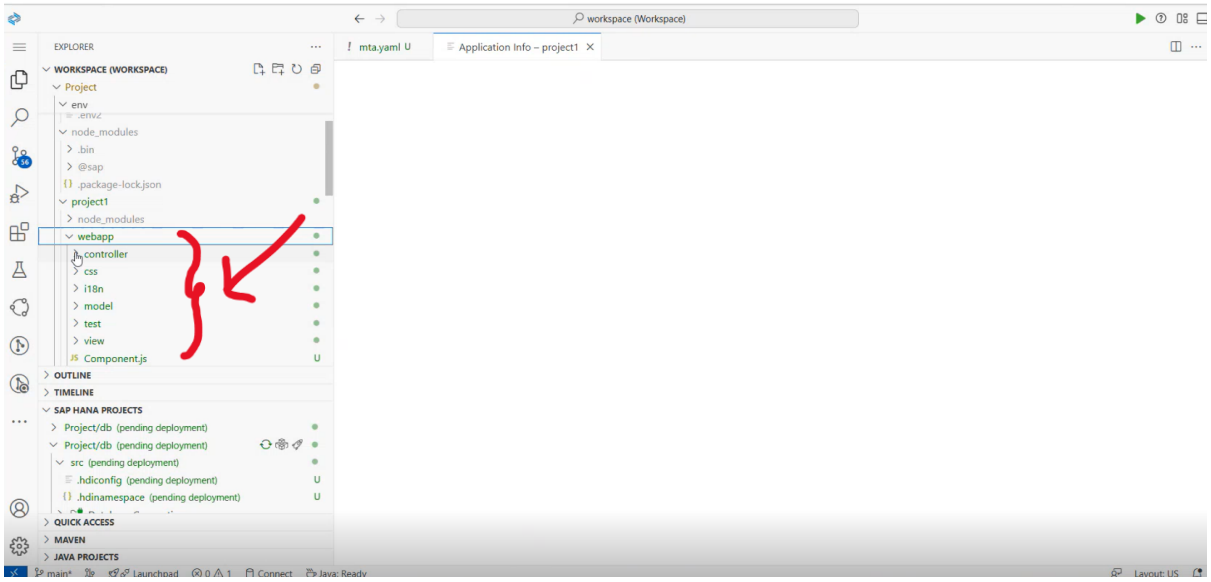


Image 8- webapp folder is successfully created after completing the Fiori module setup

◇ Part 5: Deploying the Project to GitHub

To continue development locally or customize the application further, you can now **deploy or open the project in Visual Studio Code**.

To deploy your CAP project from SAP Business Application Studio (BAS) to GitHub, follow these command-line steps:

1️⃣ **Initialize Git in Your Project Folder**

```
git init
```

2️⃣ **Add Remote GitHub Repository**

```
git remote add origin https://github.com/username/projectname.git
```

3️⃣ **Verify Remote Setup**

```
git remote  -v
```

4️⃣ **Stage Your Changes**

```
git add .
```

5️⃣ **Commit Your Changes**

```
git commit -m "Initial commit"
```

6️⃣ **Publish Your Branch to GitHub**

Click on **"Publish Branch"** in the BAS Git panel.

👉 A browser window will open prompting you to **copy a verification code**.

7️⃣ **Authorize SAP BAS to Access GitHub**

Paste the copied verification code into GitHub and **authorize SAP Business Application Studio**.

8️⃣ **Pull Latest Changes from Main Branch (If Needed)**

```
git pull origin main --rebase
```

9️⃣ **Or Simply Pull**

```
git pull
```