SAP BTP - PBC - Workshop

Hands-on Guide

Contents

[Clone Template Project from GitHub 2](#_Toc190200402)

[Create the Database Layer 7](#_Toc190200403)

[Create the OData Service Layer 8](#_Toc190200404)

[Build and Deploy Application to SAP BTP Cloud Foundry Runtime 8](#_Toc190200405)

[Build 8](#_Toc190200406)

[Log in to Cloud Foundry 8](#_Toc190200407)

[Deploy 11](#_Toc190200408)

[Assign the App Role Collections 13](#_Toc190200409)

[Pushing Mails to the Application 14](#_Toc190200410)

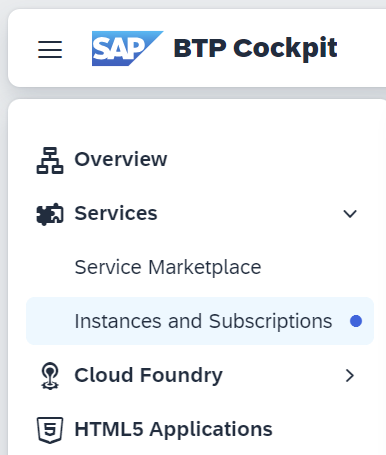
[The information required will be found in the XSUAA Service instance. 14](#_Toc190200411)

[Testing the GenAI Mail Insights Application 20](#_Toc190200412)

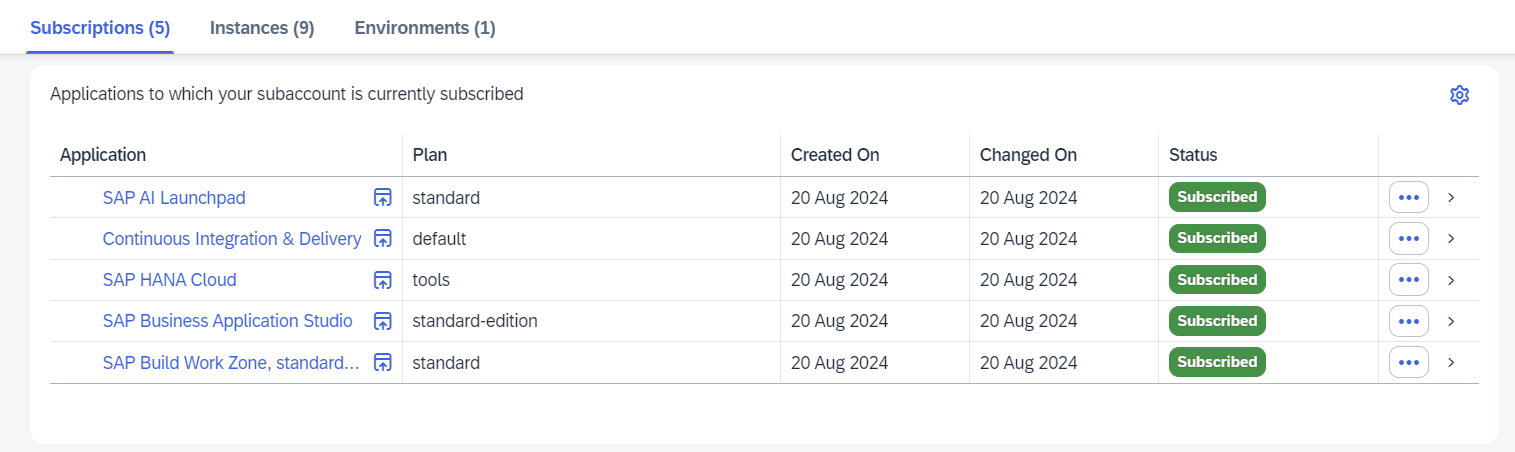
# Clone Template Project from GitHub

**SAP BAS Dev Space**: <https://help.sap.com/docs/bas/sap-business-application-studio/dev-spaces-in-sap-business-application-studio>

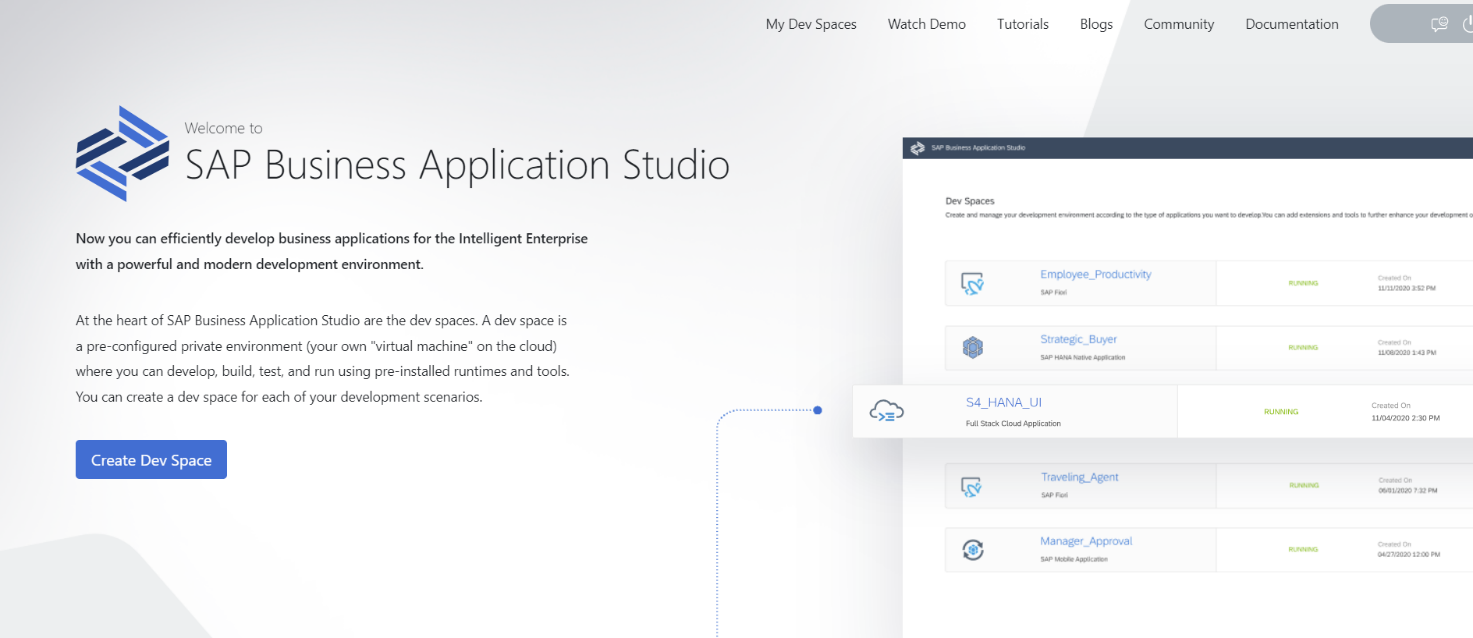
In the SAP BTP Cockpit, navigate to **Services** > **Instances and Subscriptions**.



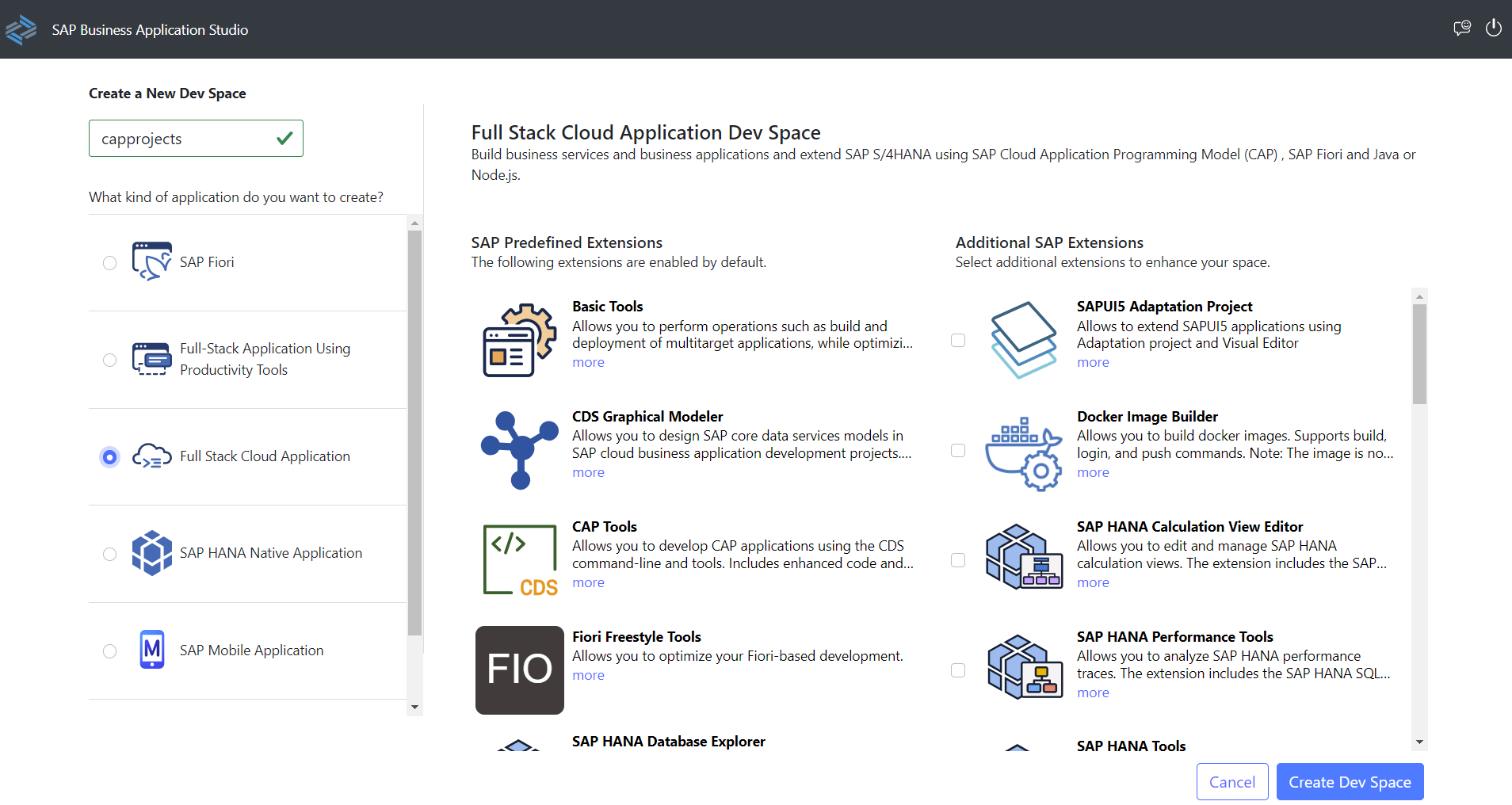
On the **Subscriptions tab**, click on SAP Business Application Studio (BAS) to launch it in a new tab.



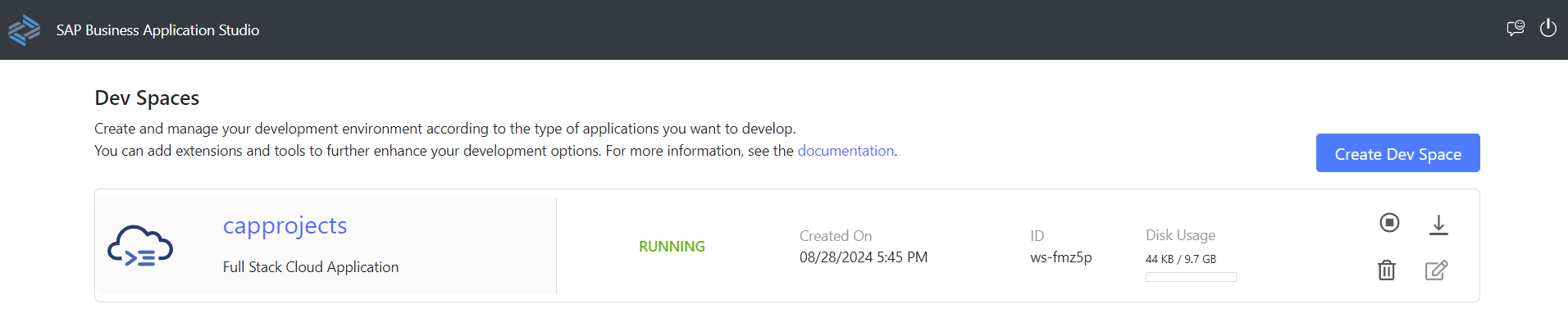
If it is the first time you are accessing it, the welcome page will be displayed, and you will need to create a **Developer Space**. Click on the **Create Dev Space button**. This step is executed one time only.



Give the **Dev Space** a **name**, select the **Full Stack Cloud Application** option, and click on **Create Dev Space button**.

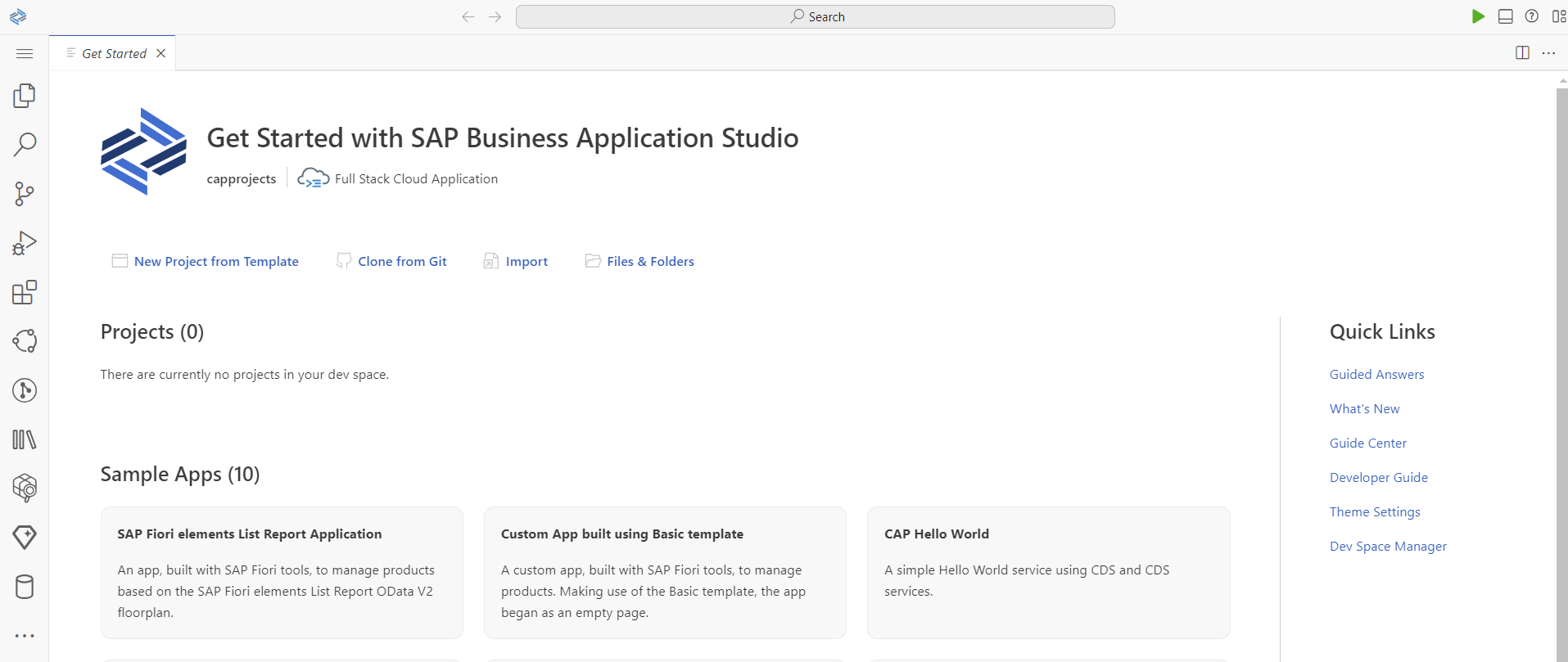


Once the **Dev Space** is created and it is **RUNNING**, click on its name to launch **BAS**.

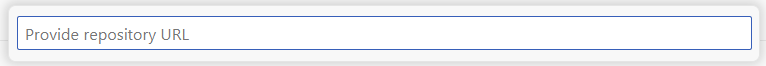


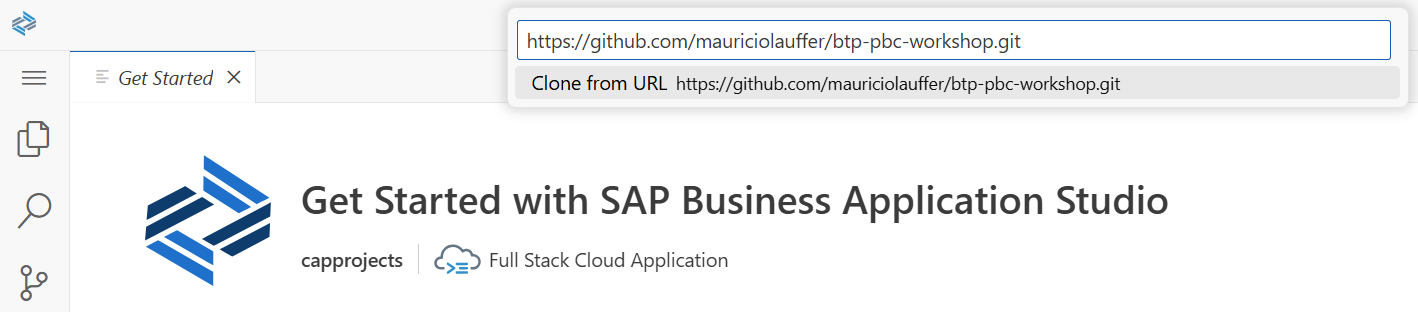
We will not start a project from scratch. We will clone a template project from GitHub.

Click on **Clone from Git**.

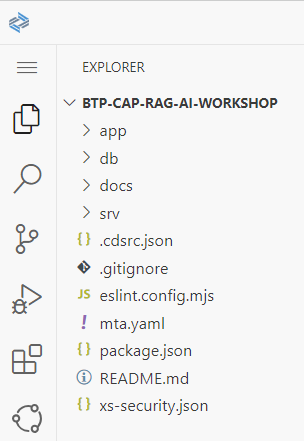


Enter the **git repository** URL (<https://github.com/mauriciolauffer/btp-pbc-workshop.git>) and press **ENTER**.

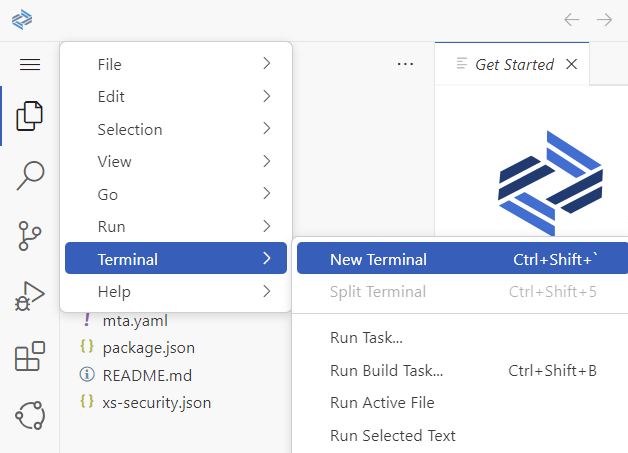




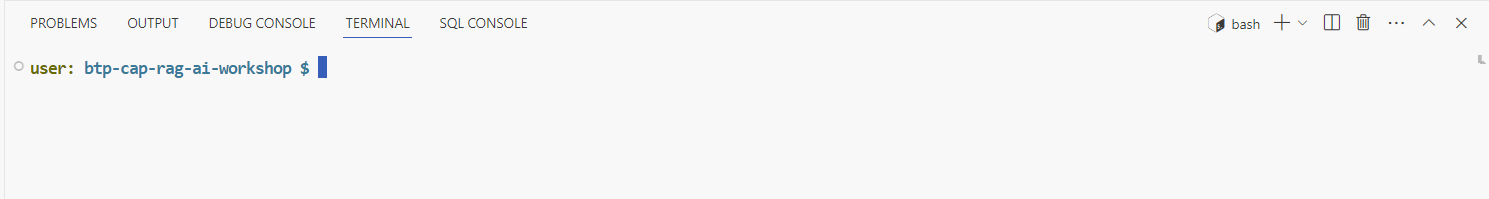
The template project will be cloned and opened. You can start exploring the codebase in the **Explorer** section.

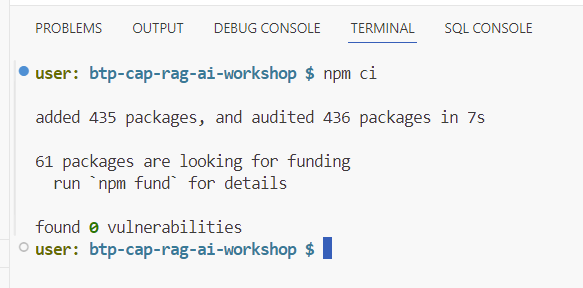


Open the Terminal to install the project dependencies. Click on the **hamburger button** (3 stacked lines icon), then **Terminal** > **New Terminal**.



The Terminal should open at the bottom of the screen, install the project dependencies with the command:   
**$ npm ci**





Now, install the global dependencies used to build and test the application with the command:  
**$ npm i -g ts-node tsx mbt**

# Create the Database Layer

Open the file **db/data-models.cds** and create the tables:

entity Translations : cuid {

      sender            :      String;

      subject           :      String;

      body              :      LargeString;

      summary           :      String;

      responseBody      :      LargeString;

      keyFacts          : many KeyFact;

      requestedServices : many String;

}

entity Mails : managed, cuid {

      subject                :      String;

      body                   :      LargeString;

      senderEmailAddress     :      String;

      sender                 :      String;

      responded              :      Boolean default false;

      category               :      String;

      sentiment              :      Integer;

      urgency                :      Integer;

      summary                :      String;

      responseBody           :      LargeString;

      languageNameDetermined :      String;

      languageMatch          :      Boolean;

      embedding              :      Vector(1536);

      translation            :      Composition of Translations;

      requestedServices      : many String;

      suggestedActions       : many Action;

      keyFacts               : many KeyFact;

}

# Create the OData Service Layer

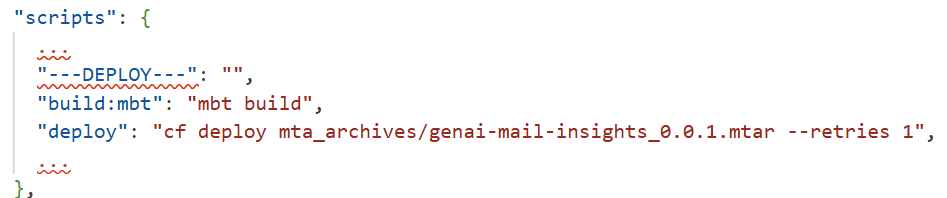
Open the file **srv/mail-insights-service.cds** and explore the OData Service definition.

Open the file **srv/ mail-insights -service.ts** and explore the service implementation. The code is written in Node.js using TypeScript which gives us better code quality and protects against common JavaScript errors.

The AI prompting and SAP AI Core configuration can be found in the file **srv/utils/ai-core.ts**. This file is responsible for handling everything SAP AI Core related, open this up and explore its content.

# Build and Deploy Application to SAP BTP Cloud Foundry Runtime

Access the Terminal to execute pre-defined NPM scripts to build and deploy the application to SAP BTP Cloud Foundry Runtime.



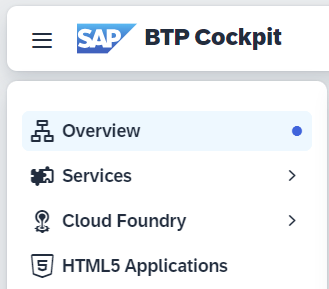
### Build

In the terminal, execute $ **npm run build:mbt**



### Log in to Cloud Foundry

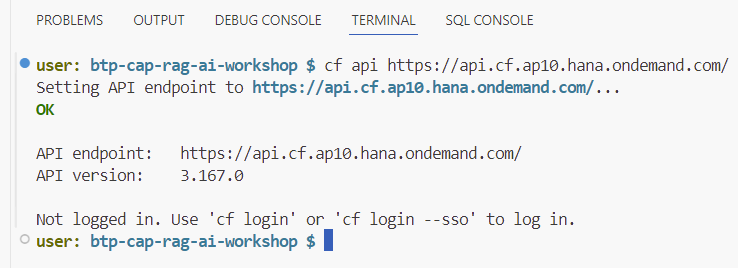
Before deploying the application to the **Cloud Foundry Runtime**, you need to log in to the target system. You will need the target system **CF API Endpoint**. This information can be found on the **SAP BTP Cockpit Overview** page. Click on the **Overview** menu to navigate to it.



In the Cloud Foundry Environment tab, copy the **API Endpoint** value (<https://api.cf.ap10.hana.ondemand.com>). The value will vary based on the subaccount region.

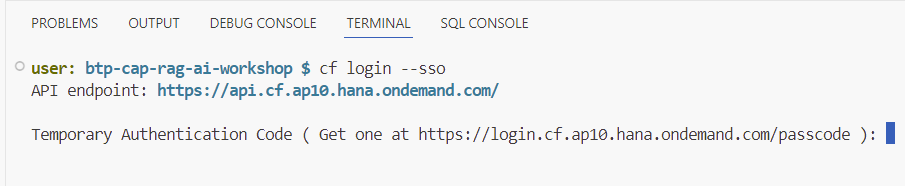


Now, go back to **BAS** to set the **CF API Endpoint** to be used for deployment. In the terminal, execute the command **$ cf api YOUR\_API\_ENDPOINT\_GOES\_HERE**

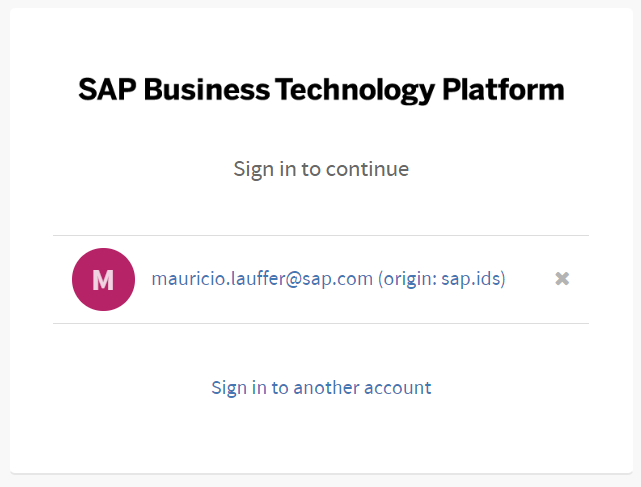


After setting the API Endpoint you can log into the SAP BTP Subaccount Cloud Foundry runtime. In the terminal, execute the command **$ cf login --sso**

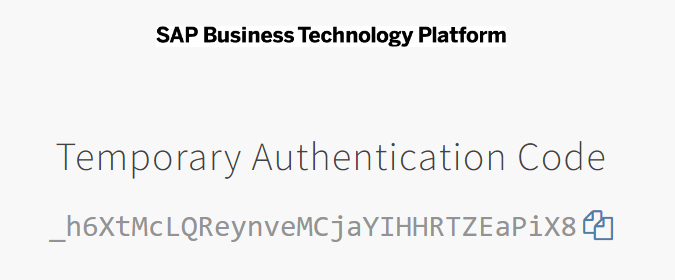
This will show you a link to the **Login page**. Click on the link to open it in a new tab.



The **Login page** will be displayed. Log into the **SAP BTP Subaccount** with your credentials.



Copy the **Temporary Authentication Code**.



Go back to **BAS**. In the terminal, paste or type the **Temporary Authentication** Code and press Enter. A list of CF ORGs will be listed, select the one used for the workshop, type its number and press **ENTER**.



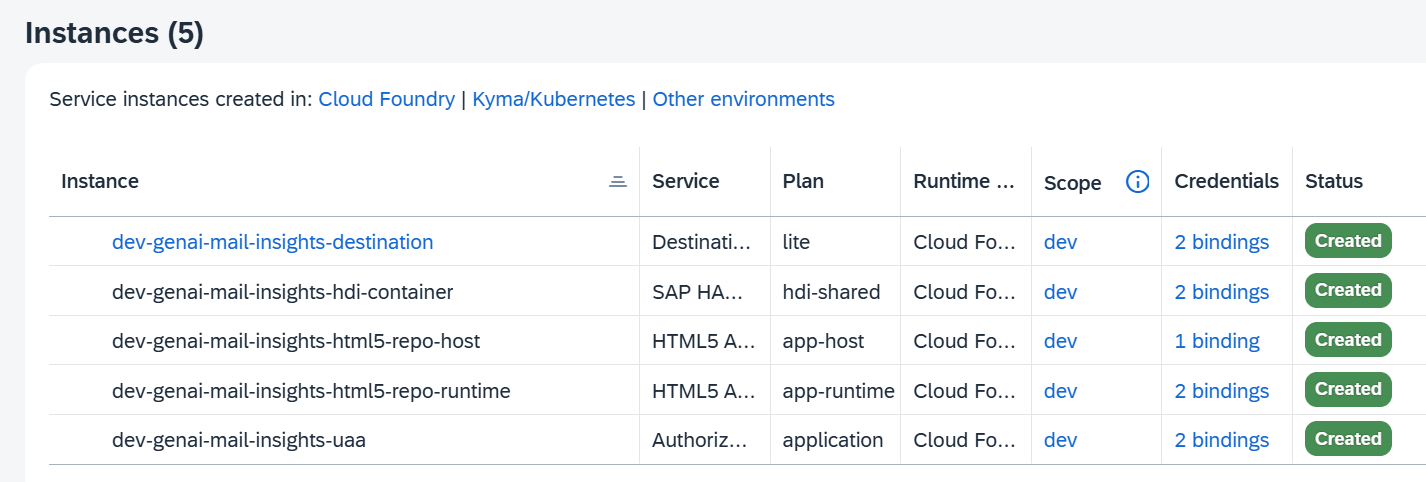


### Deploy

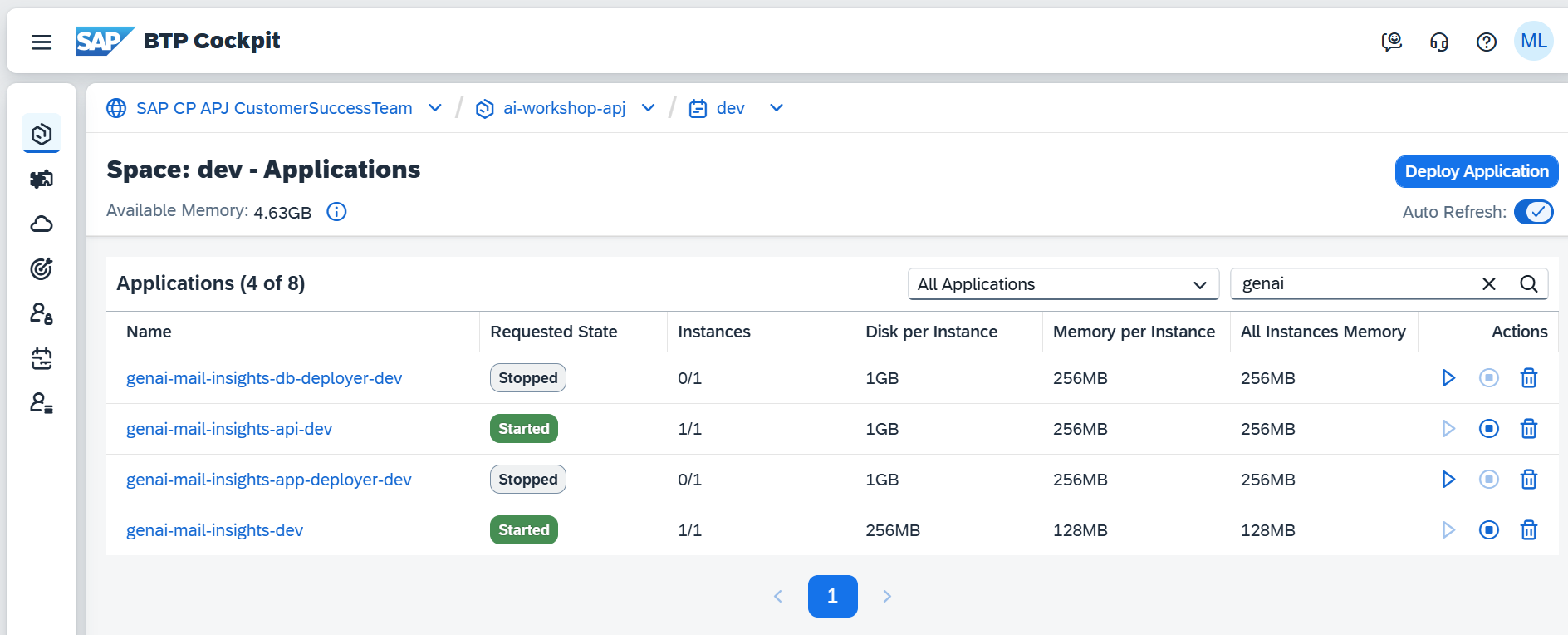
Now, execute the command $ **npm run deploy**



The deployment process will take a while. Wait for the process to finish successfully. The newly created services instances should be visible in **SAP BTP Cockpit**.



And the applications, in the **Cloud Foundry space** used for deployment.

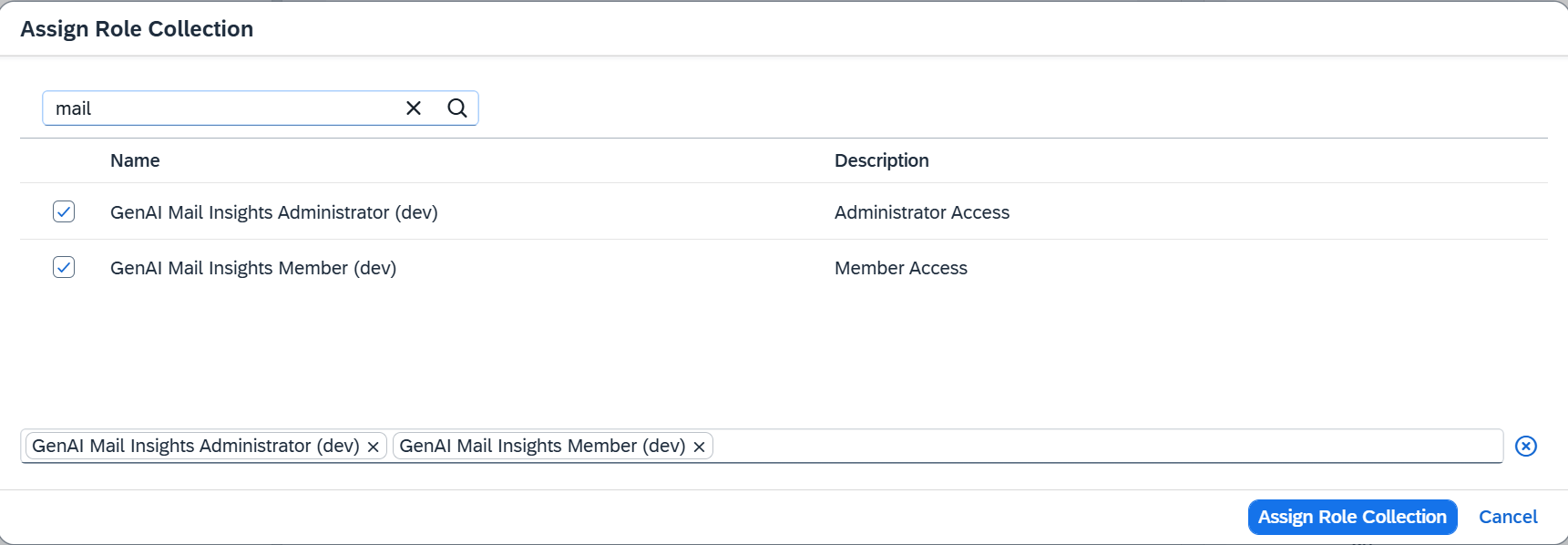


# Assign the App Role Collections

In your SAP BTP Cockpit, switch to **Security > Users** menu and assign the **GenAI Mail Insights Administrator** or **Member** role collection to the users who need to access this application.



Select both roles **GenAI Mail Insights Administrator** or **GenAI Mail Insights Member** and click on the **Assign Role Collection** button.

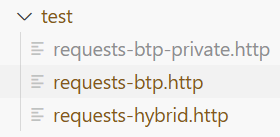


# Pushing Mails to the Application

Now, we will push sample mails to **GenAI Mail Insights** using prepared HTTP requests.

Find the provided **.http** sample file **test/requests-btp.http**, containing a few test mails for processing.

Duplicate and rename the file to **requests-btp-private.http** to ensure that your credentials are not accidentally being committed to GitHub.



Update the variables in the very beginning of the **http** test file using the existing **XSUAA** Service Instance and **route** created for your CAP Application.

@xsuaaHostname = <uaa.url>

@btpXsuaaClient = <uaa.clientid>

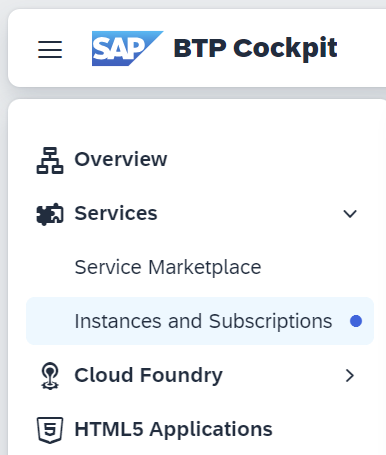
@btpXsuaaSecret = <uaa.clientsecret>

@btpAppHostname = <app.url>

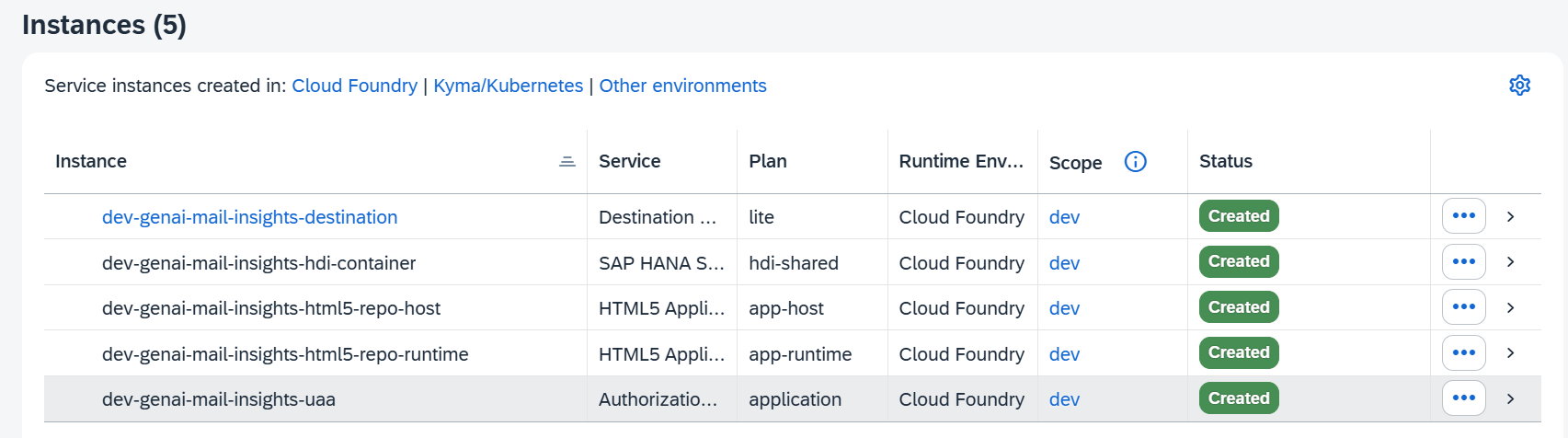


### The information required will be found in the XSUAA Service instance.

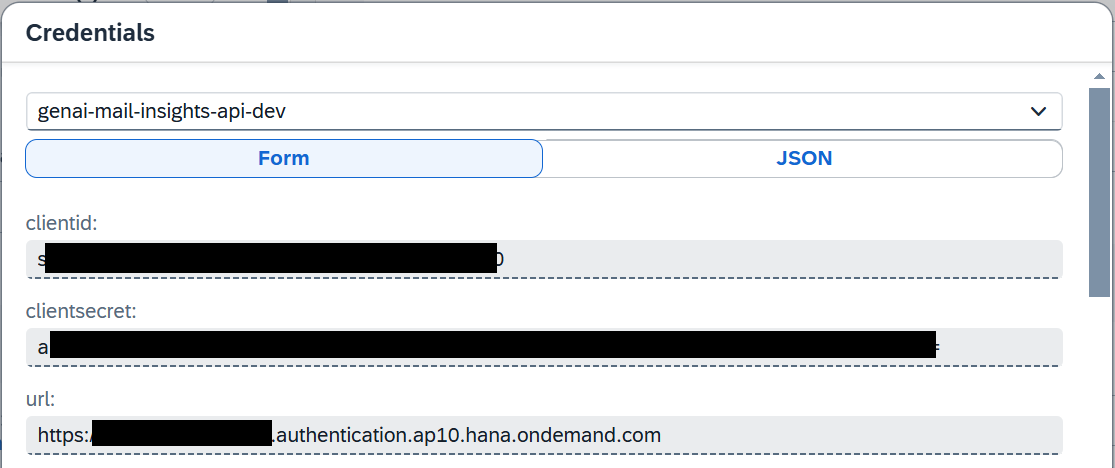
In the **SAP BTP Cockpit**, navigate to **Services** > **Instances and Subscriptions**.



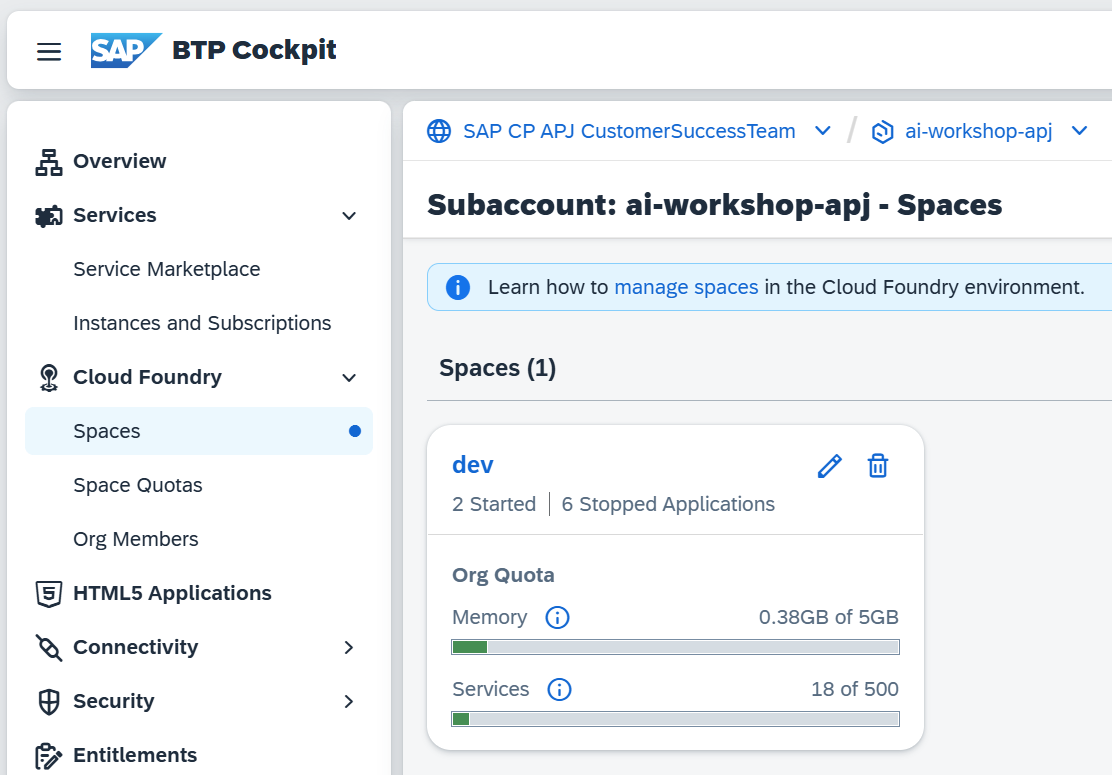
On the **Instances tab**, click on the XSUAA service instance (**dev-genai-mail-insights-uaa** in the image).

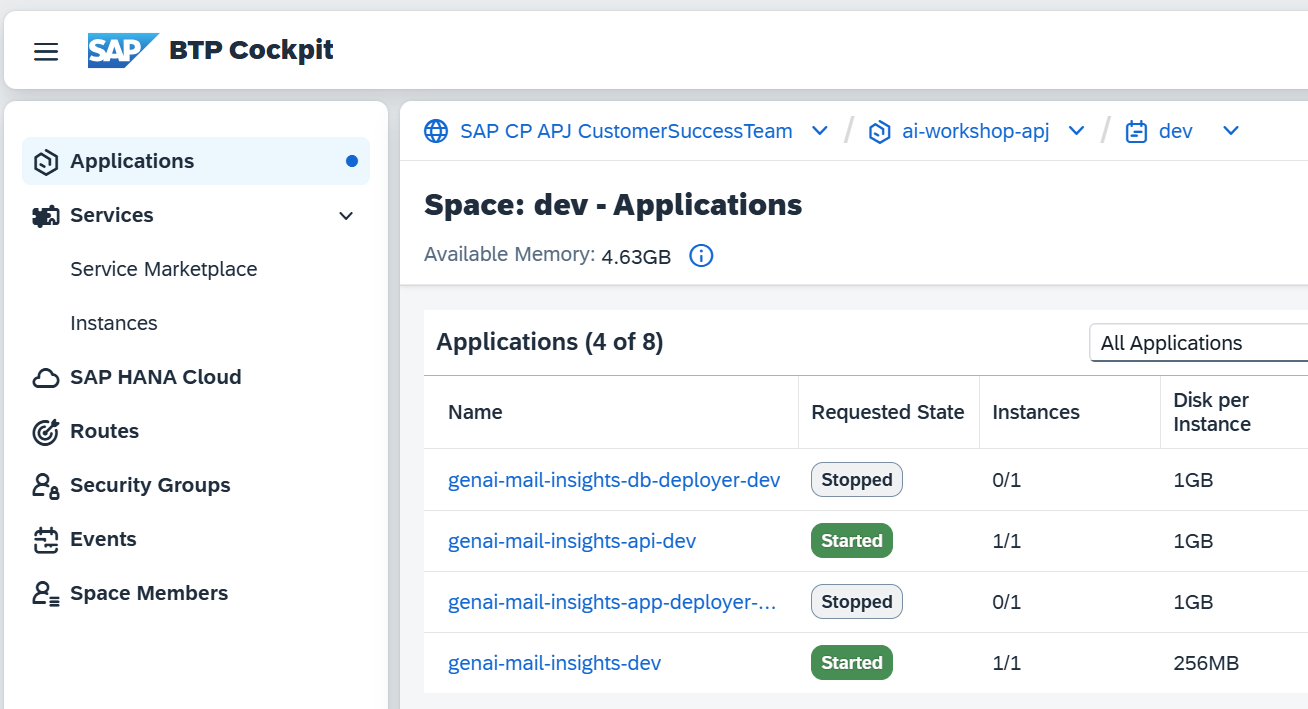


Click on **View Credentials** at the top, or, on the **Service Key Name** to open a dialog with the information required. Here you can find **<uaa.url>** + **<uaa.clientid>** + **<uaa.clientsecret>**

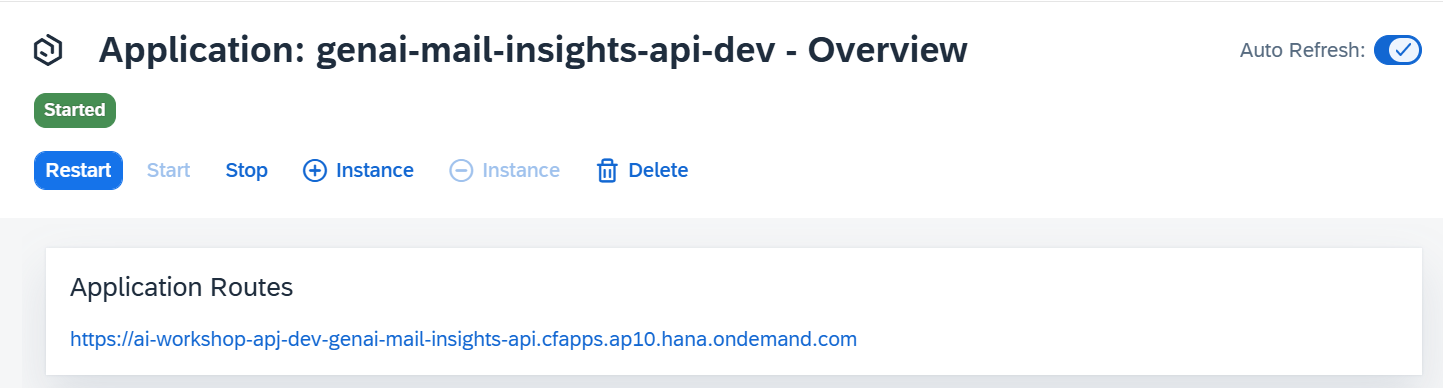


To get the <app.url> value, navigate to **the Cloud Foundry space** where the application is running, and click on the application **genai-mail-insights-api-dev**.



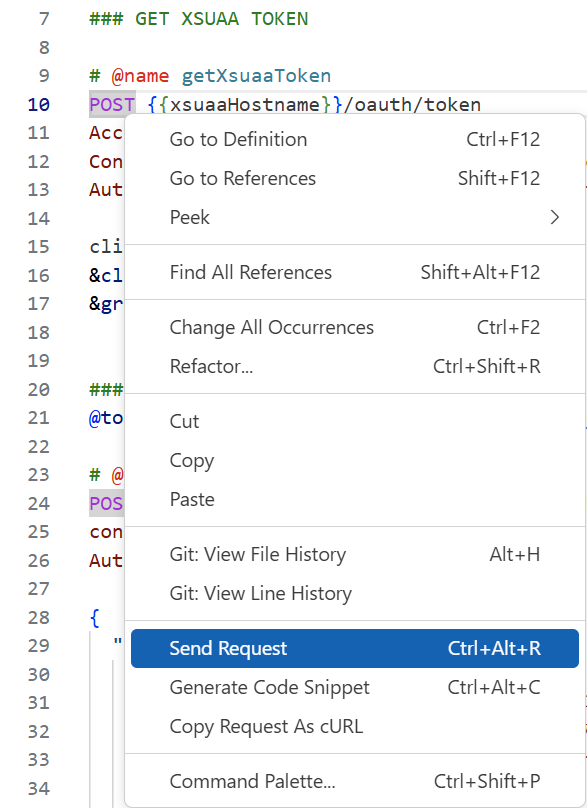


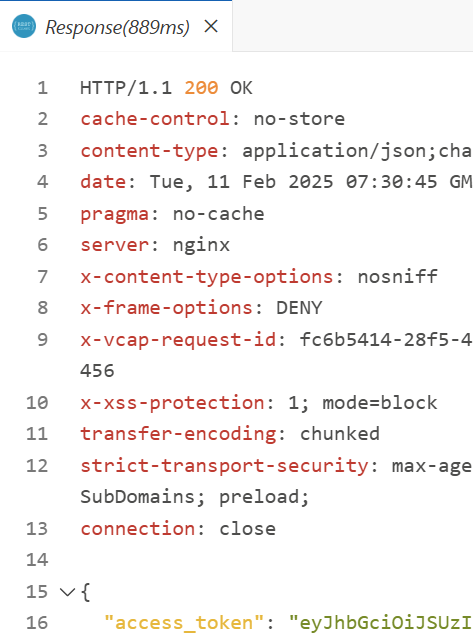
The Application Routes is the **<app.url>** value you need.



Now that you have update the file with the proper credentials and target values, it is time to push mails to the application.

Execute the **### GET XSUAA TOKEN** request by clicking on **Send Request** or right-click on **POST** and click on **Send Request**.





Once you successfully retrieved a token issued by XSUAA, please scroll down and execute the **### ADD MAILS I** request. The processing will take a while, so please wait until the request returns a result. The request will automatically inject the token retrieved from XSUAA.



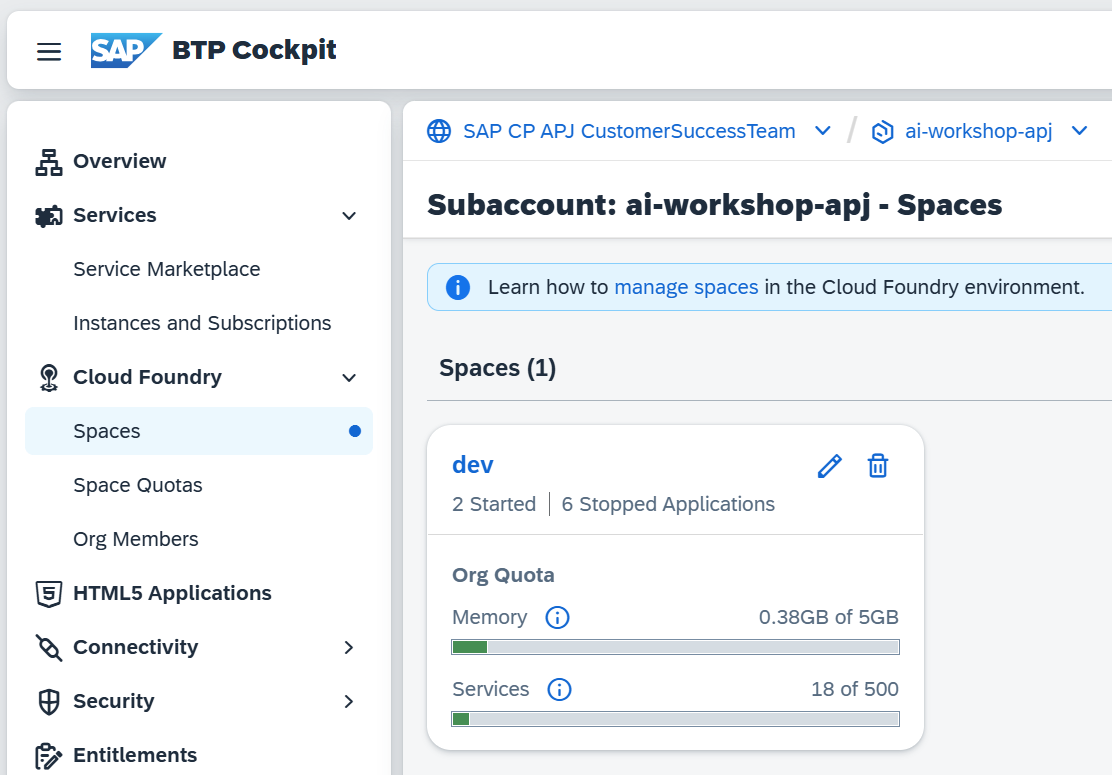
Continue with the **### ADD MAILS II** and **### ADD MAILS III** requests to add further sample mails to your application. Injecting the sample mails has been split into three parts, given the token limit of the Large Language Model being used. Please run the three **ADD MAILS** requests sequentially.

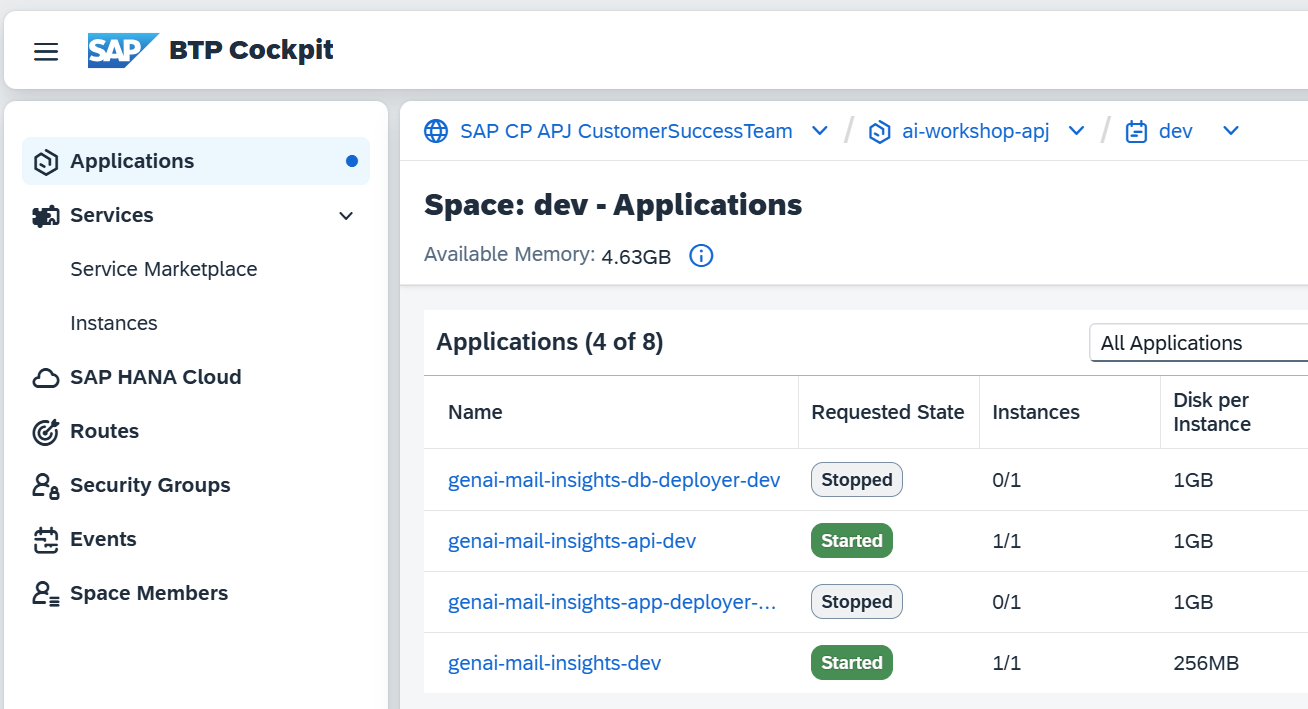
Once all sample mails have been processes successfully, please continue with the next chapter to assign the required role collection before opening the GenAI Mail Insights application ([click here](./4-TestApplication.md)).

# Testing the GenAI Mail Insights Application

You can test the test the **GenAI Mail Insights** application now.

Navigate to **the Cloud Foundry space** where the application is running, and click on the application **genai-mail-insights-dev**.





Click on the **Application Routes** link, the app will be launched in a new tab.

