Environment Setup and Deployment

Prerequisites: none

Result: Running CAP application in a local environment, deploying the app to Cloud Foundry.

1. GIT installation

This project uses Git as a source control system.

1. Check whether git is already installed.

   ```
   Shell/Bash
     1 |    git version
   ```

2. The output should look like:

   ```
   Shell/Bash
     1 |    git version 2.x.x
   ```

   If not, navigate to the Git webpage and follow installation instructions for your specific system.

2. Node.js installation

Backend application is based on CAP (SAP Cloud Application Programming Model). There are two runtimes a developer can choose, Node.js and Java. We will be using Node.js throughout our project.

1. Check whether Node.js version >= 16 is already installed.

   ```
   Shell/Bash
     1 |    node -v
   ```

2. The output should look like:

   ```
   Shell/Bash
     1 |    v16.x.x
   ```

   If not or the version number <= 16, navigate to the Node.js webpage and follow installation instructions for your specific system.

3. Cloud Foundry command line interface installation

CF CLI is used to deploy and manage your apps.

1. On macOS:

```
Shell/Bash
 1 |    brew install cloudfoundry/tap/cf-cli
```

For Windows, download a binary installer as described here.
For Linux, follow the steps described here.

2. Check if installation was successful. Based on your operating system the command might be different. For example, in Windows with Cloud Foundry CLI version 8, use cf8 --version.

```
Shell/Bash
 1 |    cf --version
```

4. Creating a CAP application

If the application already exists, clone the repository into your chosen folder and continue with step 7. Otherwise follow the tutorial. For overview, you can preview also the intermediate steps.

1. Navigate to your chosen directory and create an initial CAP project. The directory name cannot contain invalid characters such as spaces.

```
Shell/Bash
 1 |    cds init
```

You should see something like this.

```
Shell/Bash
 1 |    [init] – creating new project in current folder
 2 |    [init] – adding feature 'nodejs' …
 3 |    [init] – done.
 4 |    . . .
```

2. Open the generated project in your preferred editor. We will be using Microsoft Visual Studio Code since it provides useful SAP extensions and syntax highlighting. For installation, visit the webpage here. The generated code structure in VS Code looks like this:
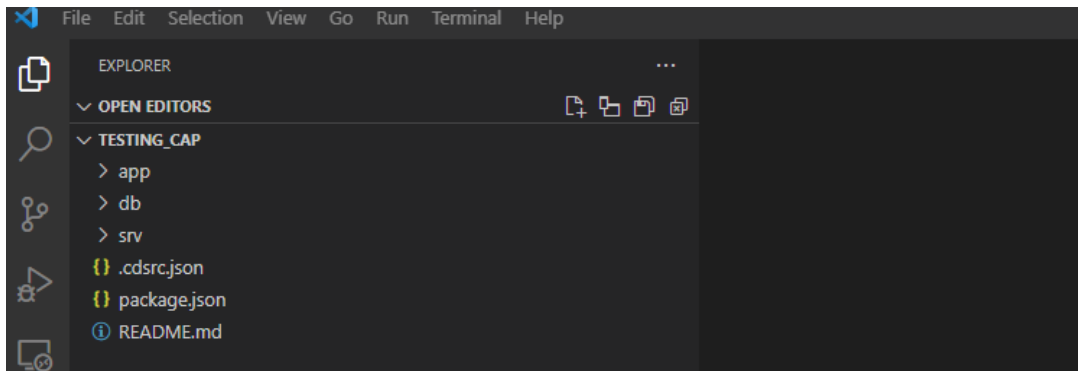
3. When developing with VS Code, go to extensions and search for SAP CDS Language Support. Install the extension to enable syntax highlighting, code completion and more.
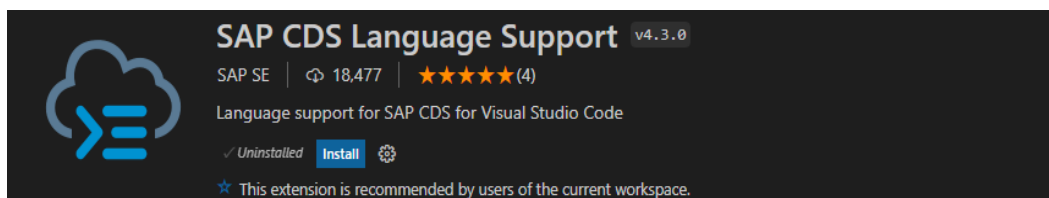
4. Run the following command in the terminal from the project directory to install necessary packages.

Shell/Bash

```
1 |     npm install
```

5. To verify that everything went correctly, start a CAP server from the project directory using following command in the terminal:

Shell/Bash

```
1 |     cds watch
```

You should see the following:

There are for now no defined models that the server could serve. You can shut down the server by pressing Command+C/Ctrl+C based on your OS.

5. Add a service model to the application

We are going to define an example service model which will serve as a basis for our CAP application that we will deploy in the next step.

1. Create a file named *example-model.cds* and place it into *db* folder. Paste the following content into the file.

```
namespace my.sales;

entity SaleOrders {
  key ID : Integer;
  title  : localized String;
  productManager : Association to ProductManagers;
  stock  : Integer;
  shippingDate : Date;
}

entity ProductManagers {
  key ID : Integer;
  name   : String;
  saleOrders  : Association to many SaleOrders on saleOrders.productManager = $self;
}
```

2. Create a file named *example-service.cds* and place it into *srv* folder. Paste the following content into the file.

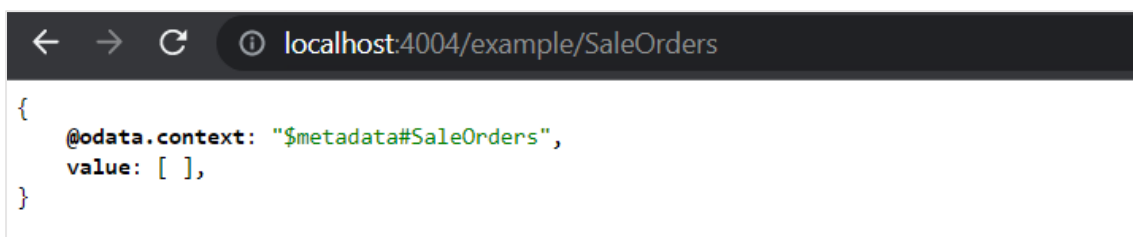```
using my.sales as my from '../db/example-model';

service ExampleService {
  entity SaleOrders @readonly as projection on my.SaleOrders;
  entity ProductManagers @readonly as projection on my.ProductManagers;
}
```

3. Run the server with:

```
Shell/Bash
1 |    cds watch
```

4. Navigate to http://localhost:4004/example/SaleOrders. You should see following screen. There is no data yet, but the GET request was executed successfully.

6. Prepare the project for build and deployment

We need to specify deployment options for an MTA application.

1. Declare which Node.js version should be used in the Cloud Foundry in the *package.json*.

package.json
```
 1 |        {
 2 |          ...
 3 |          "devDependencies": {
 4 |            ...
 5 |          },
 6 |          "engines": {
 7 |              "node": ">=14"
 8 |          },
```

2. Generate MTA deployment descriptor (*mta.yaml*) file. Run the following command from the directory of your project.

Shell/Bash
```
 1 |      cds add mta
```

3. Specify disk and memory quota in the *mta.yaml* file based on your available resources in the Cloud Foundry space.

mta.yaml
```
 1 |     modules:
 2 |       - name: sap_chatbot-srv
 3 |         type: nodejs
 4 |         path: gen/srv
 5 |         parameters:
 6 |             buildpack: nodejs_buildpack
 7 |             disk-quota: 512M
 8 |             memory: 512M
 9 |         provides:
10 |            . . .
```

7. Build and deploy CAP application to SAP BTP

To deploy a CAP application, you will need a SAP BTP global account, a subaccount, and a Cloud Foundry space with the required entitlements. Follow this tutorial if you don't have these accounts yet.

4. Find the API Endpoint of your Cloud Foundry subaccount environment.

Subaccount: ▆▆▆▆▆▆▆▆ - Overview

General   Cloud Foundry Environment   Entitlements

Cloud Foundry Environment

API Endpoint: ▆▆▆▆▆▆▆▆▆▆▆▆   Spaces (1)
Org Name: ▆▆▆▆▆▆▆
Org ID: ▆▆▆▆▆▆▆▆

Name	Applications	Service Instances
▆▆▆▆▆▆	0	0

5. Set the Cloud Foundry API endpoint.

```
Shell/Bash
1 |    cf api <API endpoint>
```

6. Log into the Cloud Foundry account using SAP BTP credentials.

```
Shell/Bash
1 |    cf login
```

If you are getting an Unauthorized error and you typed your credentials correctly, log in via SSO.

```
Shell/Bash
1 |    cf login --sso
```

7. Check if Cloud MTA Build Tool is already installed.

```
Shell/Bash
1 |    mbt --version
```

If not, install it with the following command:

```
Shell/Bash
1 |    npm install --global mbt
```

8. On Windows, install *make* from http://gnuwin32.sourceforge.net/packages/make.htm. Choose the download **Complete package, except sources**. Run the installer, find the location where it was installed and add the folder *GnuWin32\bin* to the *Path* environment user variable.

9. Install *multiapps* plugin that is necessary for deploying a CAP application.

    Shell/Bash
    ```
    1 |     cf install-plugin multiapps
    ```

10. Build the MTA module from the project root folder.
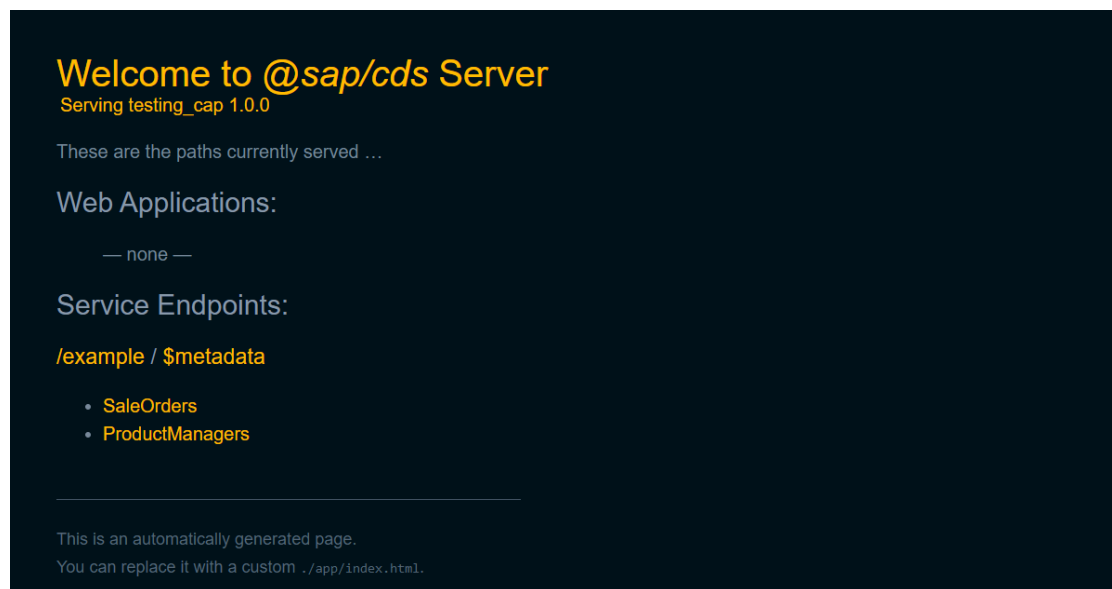
    Shell/Bash
    ```
    1 |     mbt build -t ./
    ```

11. This generated *.mtar* file in the project root folder. Deploy the module to the Cloud Foundry space using:

    Shell/Bash
    ```
    1 |     cf deploy <mtar file>
    ```

    If the deployment was successful, you should get a link which points to your CAP application. After opening the link, you should see the following screen:

    You can see two service endpoints. The service that we defined as *ExampleService* and *$metadata* service that gets added by default.