DEVELOPING MULTITENANT APPLICATIONS ON SAP CLOUD PLATFORM

CPL282

Exercises / Solutions
Anuj Mehta / SAP
Mahipal Ramachandran / SAP
Ashitha MS / SAP

SAP TECHED

TABLE OF CONTENTS

Introduction 3

Part 1: Persistence Service multitenancy – Data initialization and isolation 6
- **Step 1: Review that data isolation using a Tenant Discriminator Column has been setup.** 8
- **Step 2: Review that the data initialization servlet has been setup and exposed.** 11
- **Step 3: Use the data initialization servlet to initialize pollution data for consumer ABC PetroCorp** 13

Part 2: Identity Management – Assign application roles to users in SAP CP 17
- **Step1: Application logic for Plant Management based upon different roles** 17
- **Step2: Roles assignment in SAP Cloud Platform cockpit** 19
- **Step3: Validation of Identity Management setup** 21

Part 3: On-Premise Connectivity service – On premise setup and configure connectivity service 23
- **Step 1: Expose on-premise services securely using SAP Cloud Connector** 24
  - **Section 1: Simulate on-premise setup by running a mock service on your machine** 24
  - **Section 2: Expose the mock service using SAP Cloud Connector** 30
- **Step 2: Configure connectivity between exposed on-premise services and the cloud application using SAP CP destinations** 36
  - **Section 1: Configure Destinations from SAP CP** 36
  - **Section 2: Access the on-premise service from the cloud Java application** 39

Part 4: Configure a Connectivity service and Test the multitenant application 43
- **Step 1: Configure connectivity on the consumer account (to service running on SAP CP)** 43
- **Step 2. Test the Pollution Monitoring dashboard by login to HTML5 Application** 49

Part 5: Fiori Launchpad – Configuring tile for multi-tenant application 50
- **Step1: Login to SAP CP Portal Service and Import SAP Fiori Launchpad content into ABC PetroCorp Subaccount** 50
- **Step2: Publish and Launch SAP Fiori Launchpad site from ABC PetroCorp subaccount** 55

INTRODUCTION

'ITeLO Consulting' is a trusted SAP partner company that specializes in providing solutions for the **Oil and Gas industry**. They have adopted the **SAP Cloud Platform** as the innovation platform to build and run their applications on the cloud. They choose to deliver their solutions via the **Software as a service (SaaS) model** in which the software is deployed as a hosted service and is available to customers over the internet.

ITeLO Consulting has developed a **pollution monitoring dashboard** that helps Oil & Gas companies view a comparison of the air pollution levels onsite (at the plant) vs. the air pollution levels at the city (where the plant is located). They have found interest for this solution from certain Oil & Gas companies that want to adopt measures for proactive pollution prevention.

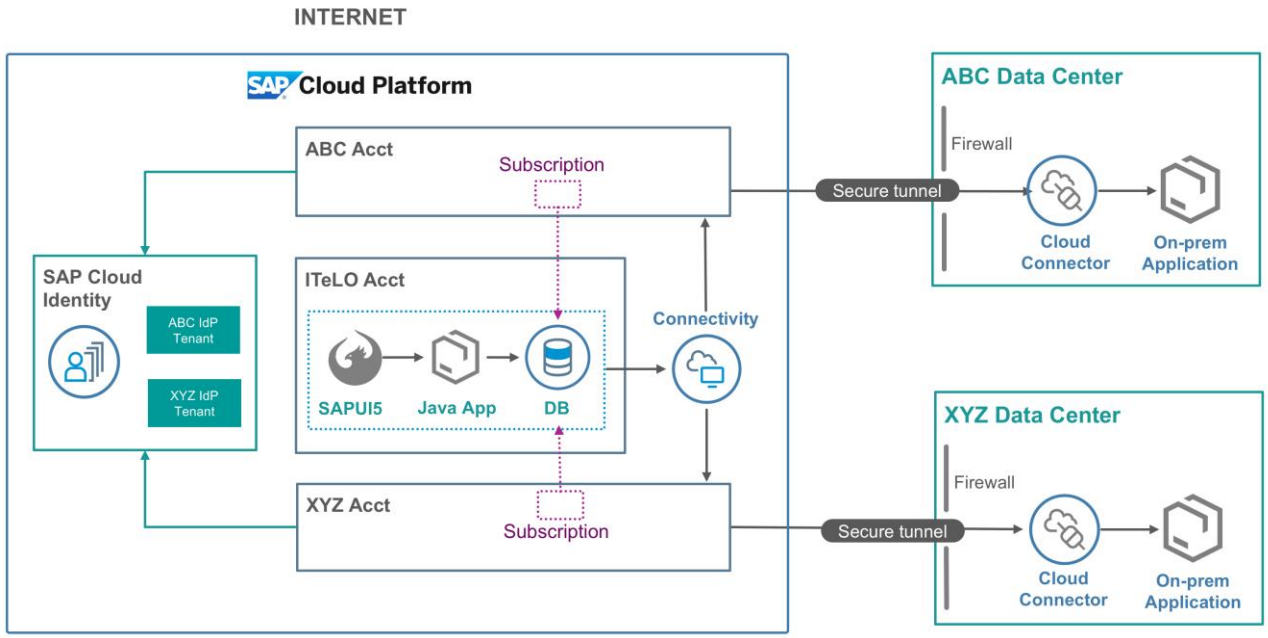
'ABC PetroCorp' and **'XYZ EnergyCorp'** are two such Oil & Gas companies based out of Canada and US respectively, that are buying this solution from ITeLO.

'Robert' works for ITeLO Consulting and is the architect for this project. ITeLO has bought an SAP Cloud Platform package enabled with the services required to build this application.

Robert's scenario is to develop a true **mutli-tenant SaaS application** and deliver it to his customers.

'Emily' is an employee of the IT department of ABC PetroCorp. She has been assigned to work on this project with the Robert.
For XYZ EnergyCorp, **'Stephen'** is an employee of the IT department

The architecture of this multitenant application looks like this:



ITeLO Consulting (Provider)' subaccount
This is the provider account in which the database, business logic and user interface applications are running.
Database
The air pollution data of each plant is stored in the HANA DB using **SAP CP Persistence Service**. The data of different tenants are kept separate by using the tenant isolation features provided by JPA.
Business Logic
The business logic is defined in the Java application which runs in the provider subaccount. The *same* instance of the Java application is run for different tenants. Each tenant creates a subscription to this Java application and gets a unique consumer access URL. This is the URL that will be used by end users to access the application. Whenever the application is accessed using this URL, SAP CP can extract the tenant ID that can be used to identify the tenant in the application logic.
User Interface
The UI logic is defined in an HTML5 application that runs on the provider account. Just like the Java application, the same instance of the HTML5 application is run for different tenants. And the application is accessed using a unique consumer access URL.

'ABC PetroCorp (Consumer)' subaccount

This is the consumer subaccount for ABC PetroCorp in which subscriptions are created and tenant specific configurations – such as the Identity Provider and connectivity to on-premise services – are made. Remember that the Java/HTML5 applications are *not* deployed or copied to this account.

Identity and Access Management
Only employees of ABC PetroCorp should be allowed to access the subscribed applications.
ABC PetroCorp has its own Identity Provider (SAP HANA Cloud Identity Authentication service) that contains information about their employees such as name, email id, organization, role etc. This Identity Provider will be configured in the *ABC PetroCorp (Consumer)* subaccount such that the subscribed applications can be accessed only by their employees.

Connectivity
ABC PetroCorp has on –premise services running within their landscape that provide information about their plants – details like no. of workers at each plant, area of the plant etc. All this information must be pulled from their landscape and provided to the subscribed application using the SAP Cloud Connector.

'XYZ EnergyCorp (Consumer)' subaccount
Consumer account for XYZ EnergyCorp. Similar to the ABC PetroCorp (Consumer) subaccount.

As outlined that ITeLO Consulting has developed a pollution monitoring SaaS application. To provide this as a multi-tenant SaaS to its customers, ITeLO must make use of the multi-tenant capabilities of SAP Cloud Platform. Let us see how this can be achieved. In the interest of saving your time, following tasks have already been done for you.

1. Project and sub-accounts set up and create subscriptions

Robert has created a subaccount for ITeLo Consulting and ABC PetroCorp.
Robert has deployed applications (Java, HTML5) build by development team of ITeLo Consulting in ITeLo Consulting (provider) subaccount
Robert has done the SAP HANA Database binding to the Java application in ITeLo Consulting (provider) subaccount.
Robert has subscribed ABC PetroCorp(consumer) subaccount to the Java and HTML5 applications running on the ITeLo Consulting (provider) subaccount.

2. Configure Open source service in the provider subaccount

As outlined above pollution monitoring dashboard application helps consumers to compare plant's pollution data with city's pollution data.
Here plant data comes from the plants but what's about city's pollution data?

The answer is air quality service from OpenAQ which is open source and publicly available over the internet. Robert has configured this service as a destination in the ITeLo Consulting (provider) subaccount.

3. Identity Management -Setup of Identity Authentication service

To provide security and identity management to Pollution monitoring dashboard application Emily has established trust between ABC PetroCorp(consumer) account and the ABC PetroCorp's IDP. She has also imported following end users required for the application in her ABC PetroCorp's IDP.

1. User:Johan ID: (Area manager)
2. User ID:Smith (PlantSupervisor)

Now we will focus on 'how Emily collaborate with Robert to do customer-specific configuration in the ABC PetroCorp (consumer) subaccount. This will be covered in the following parts:

Part1. Persistence service multitenancy – Data initialization and isolation

Part2. Identity Management – Assign application roles to users in SAP CP

Part3. On-Premise Connectivity service – On premise setup and configure connectivity service

Part4. Configure a Connectivity service and Test the multitenant application

Part5. Fiori Launchpad – Configuring tile for multi-tenant application

PART 1: PERSISTENCE SERVICE MULTITENANCY – DATA INITIALIZATION AND ISOLATION

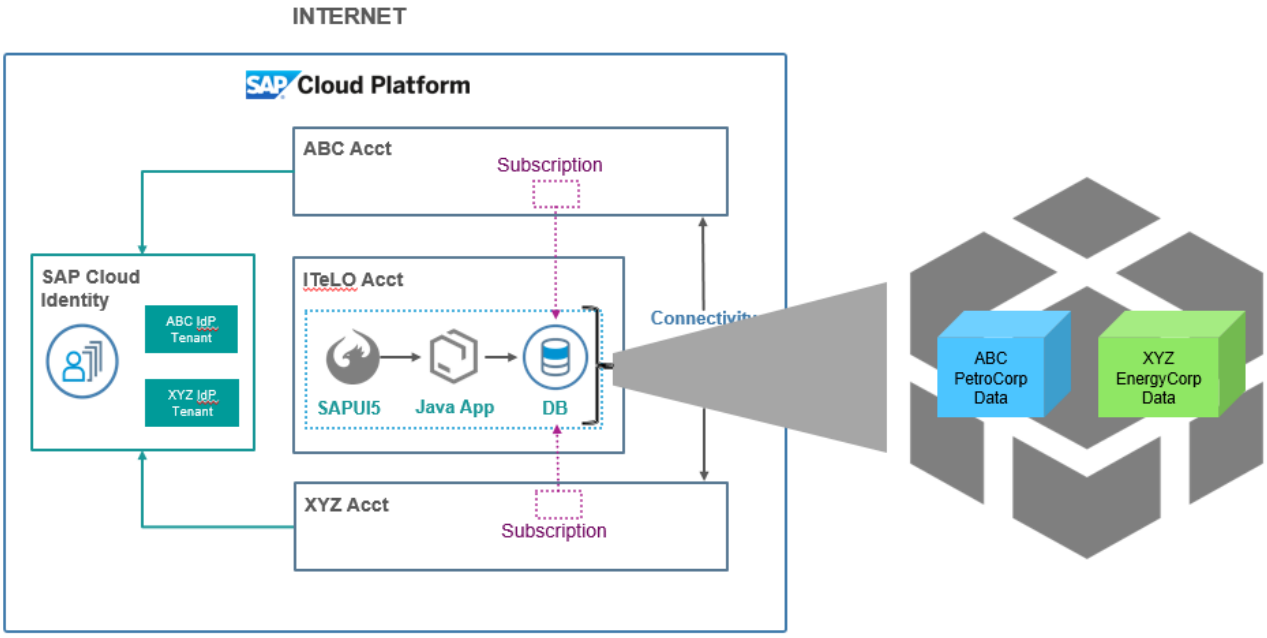
Explore how to achieve persistence multitenancy on the SAP Cloud Platform.
Initialize the Pollution data required for the application using an initialization servlet.

Time for completion: 10min

'**ITeLO Consulting**'is a trusted SAP partner company that specializes in providing solutions for the **Oil and Gas industry.** They have adopted the **SAP Cloud Platform** as the innovation platform to build and run their applications on the cloud.

As outlined earlier, ABC PetroCorp and XYZ EnergyCorp are Oil and Gas companies who are interested in purchasing ITeLO Consulting's innovative **Pollution Monitoring** solution on the SAP Cloud Platform. As the foundation for a true multitenant application, ITeLO Consulting should ensure that the data of ABC PetroCorp is completely isolated from the data of XYZ EnergyCorp in the database.

Robert from ITeLO Consulting wants his application to isolate the data storage of his customers, ABC PetroCorp and XYZ EnergyCorp. This is very important to ensure the security of customer's data and the application. Robert is wondering how this can be achieved?



To achieve this data isolation, we need to decide the granularity of isolation i.e. Table-level (store customer data in different tables in a single schema), Schema-level (store customer data in different schema), Tenant-level (store customer data in different database tenants). The SAP Cloud Platform Persistence Service, in conjunction with JPA (Java Persistence API) and EclipseLink, supports the implementation of these data-isolation methods.

There are three possible methods to achieve database/persistence multitenancy.

Using Tenant Discriminator Column: Uses a column in the table to ensure data isolation. In this approach a single DB schema is shared between all application consumers. The tenant identifier provided by the Tenant Runtime can be used as the value in the discriminator column.

Code Snippet:

```
@Entity
@Table(name = "PERSON")
@Multitenant
@TenantDiscriminatorColumn(name = "TenantId", contextProperty = "eclipselink-tenant.id", length = 36)
@NamedQuery(name = "AllPersons", query = "select p from Person p")
public class Person {
...
}
```

Using Table-per-tenant multitenancy: Uses tenant-specific tables to ensure data isolation. In this approach a new table or schema in the database is created for each customer. A tenant table discriminator specifies how to discriminate one customer's table from the other. The tables can be in the same schema, using a prefix or suffix naming pattern to distinguish them; or they can be in separate schema, using a schema tenant table discriminator.

Code Snippet:

```
@Entity
@Table(name = "EMPLOYEE")
@Multitenant(TABLE_PER_TENANT)
@TenantTableDiscriminator(type=SCHEMA, contextProperty="eclipselink-tenant.id")
public class Employee {
    ...
}
```

Using Multitenant Database Containers: Uses different database containers/tenants for data isolation. In this approach a database new tenant is created for each customer.

In this lesson, let's focus on how to achieve data isolation using a Tenant Discriminator Column in the database. In addition, as a preparatory step, we will also initialize the data required for the Pollution Monitoring application being built as a part of the hands-on.

Robert will need to perform the following steps:

Step1: Review that data isolation using a Tenant Discriminator Column has been setup.

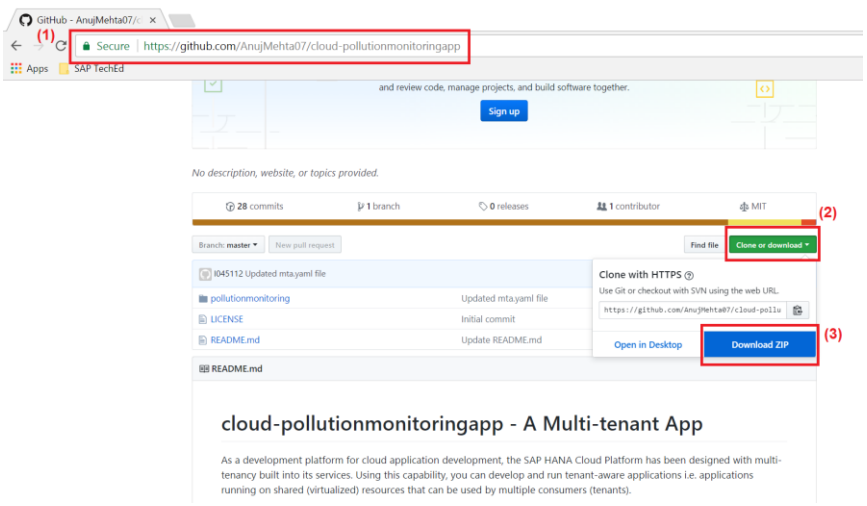
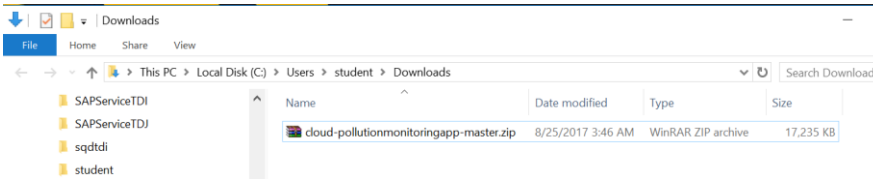
Step2: Review that the data initialization servlet has been setup so that customers can use it for their account.

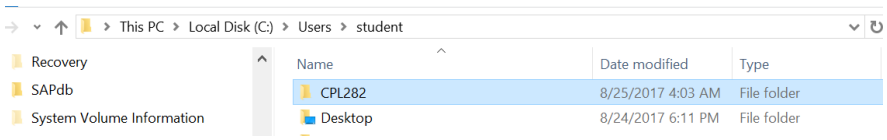
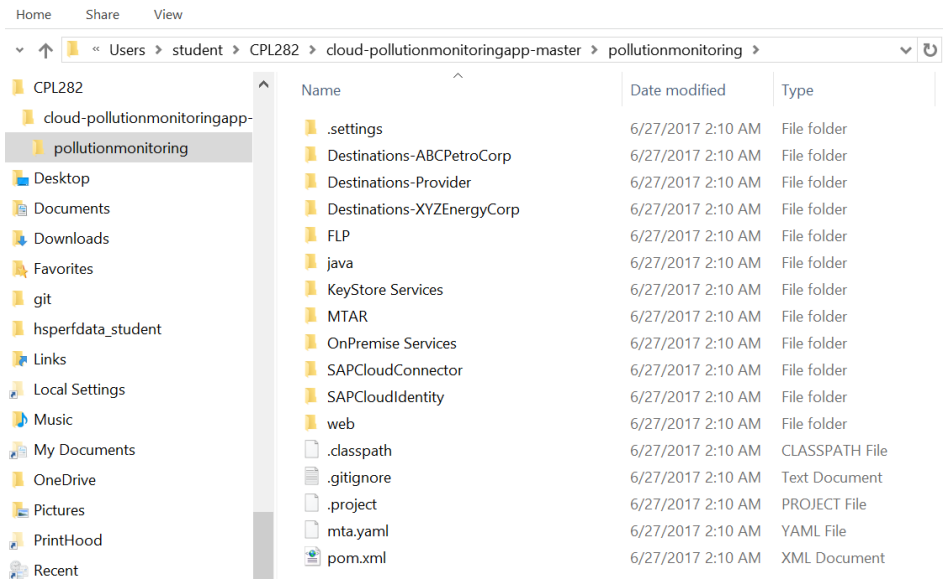
Emily will need to perform the following steps:

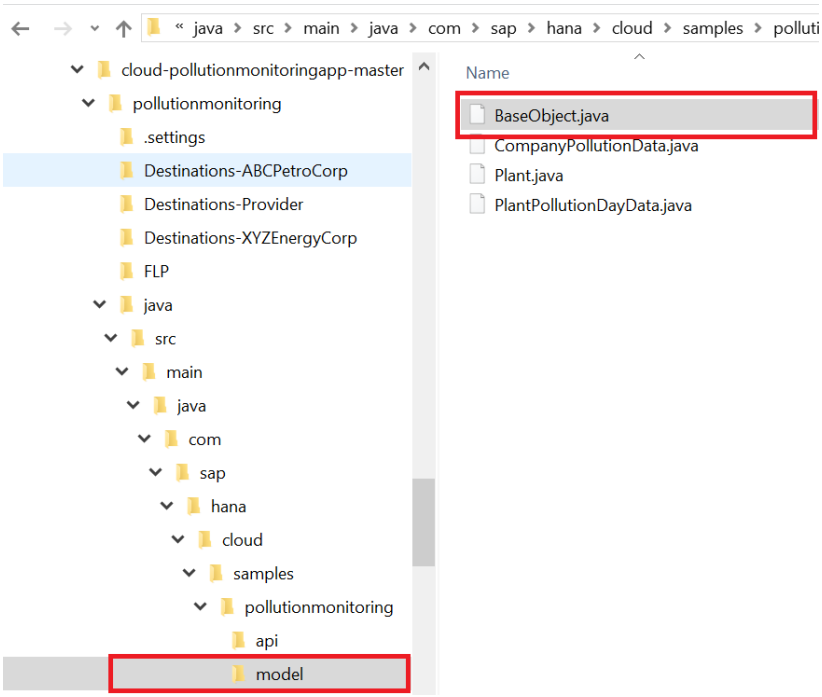
Step3: Use the data initialization servlet provided by Robert to initialize the data for ABC PetroCorp

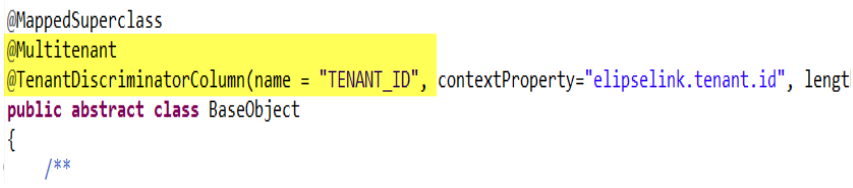
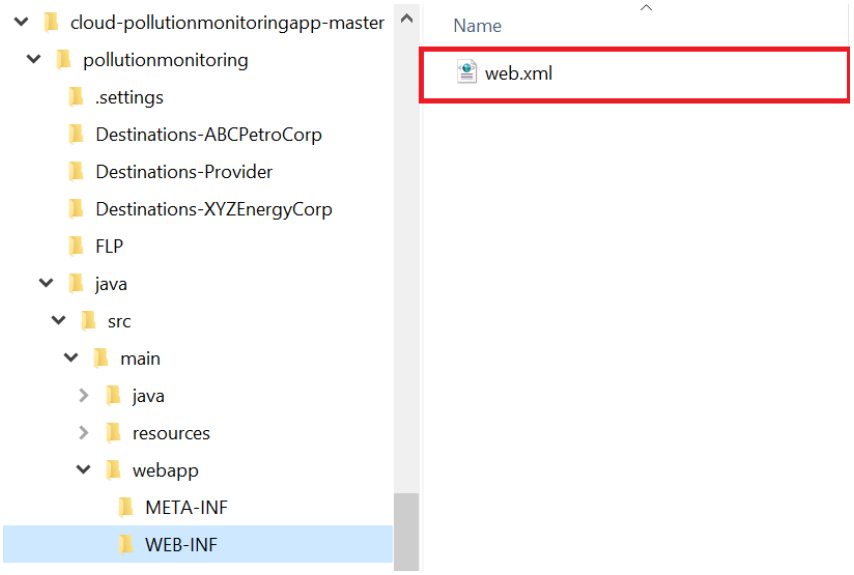
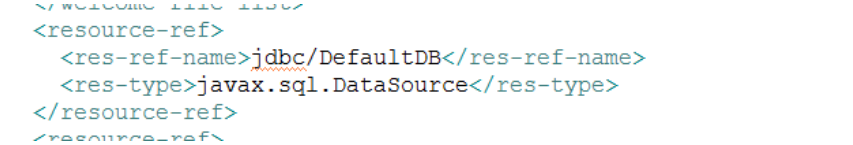
Step 1: Review that data isolation using a Tenant Discriminator Column has been setup.

In this step, you will import the pollution monitoring application to Eclipse and check how the data model of the application has been used to bring in persistence multitenancy.

Explanation	Screenshot
1. Open the following URL https://github.com/AnujMehta07/cloud-pollutionmonitoringapp in the browser. Click on the **Clone or download** and **Download ZIP** to download the content of project **pollutionmontoring.**	
2. This will download the project into **cloud-pollutionmonitoringapp-master.zip** file in the Downloads folder **(C:/Users/student/Downloads**)	

Explanation	Screenshot
3.) Create a folder with name **CPL282** in the location **C:/Users/student**	
3.) Unzip the content of **cloud-pollutionmonitoringapp-master.zip** file under the **CPL282** folder. The folder structure will look like this.	

Explanation	Screenshot
3. In the imported project, navigate to **/pollutionmonitoring/java/src /main/java/ com/sap/hana /cloud/samples /pollutionmonitoring/model /BaseObject.java**. Open the java file with notepad++	

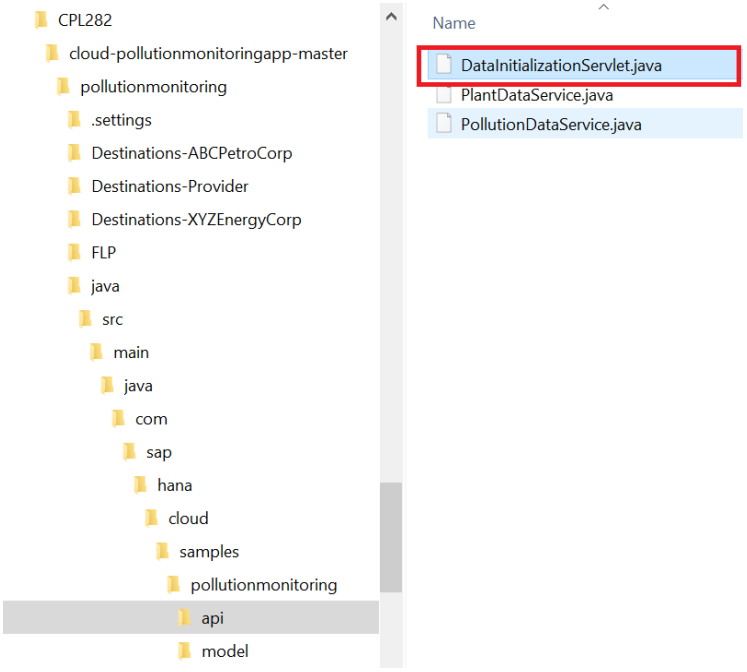
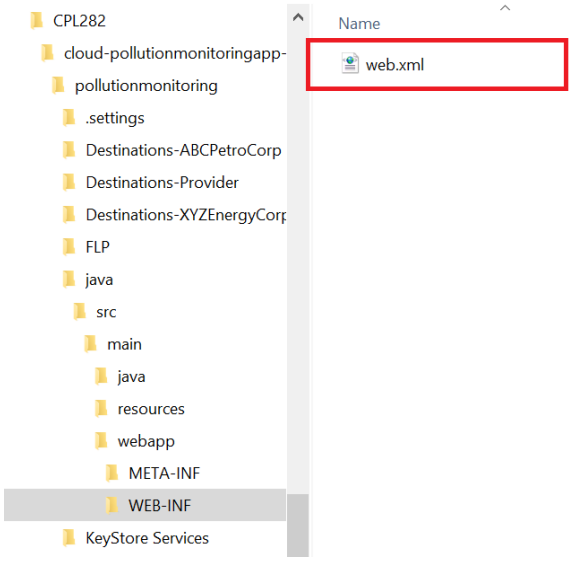
Explanation	Screenshot
4. Check how persistence multitenancy has been achieved using the **Tenant Discriminator Column** method.	
5. Navigate to **/pollutionmonitoring/java/src /main/webapp/WEB-INF/web.xml**. Open the web.xml file with notepad++	
6. Check that the **data source** (db) has been bound to the application as the default db.	

Result: We have now reviewed how multitenancy has been implemented for the Pollution Monitoring application.

Step 2: Review that the data initialization servlet has been setup and exposed.

Robert is aware that consumers of his application would require to load data for them to use it. So, he has built and exposed a data initialization servlet, which can be used by customers (like Emily) to push data into the Pollution Monitoring app.

In this step, you will check out how the data initialization servlet has been setup and how it has been exposed for consumption.

Explanation	Screenshot
1. Check the servlet **/pollutionmonitoring/java/src/main/java /com/sap/hana/cloud/samples /pollutionmonitoring/api /DataInitializationServlet.java**	
2. Navigate to ***/pollutionmonitoring/java/src/main /webapp/WEB-INF/web.xml***.	
3. Verify that the servlet is exposed using the url path '**/initialize'**	

Result: We have now reviewed how the data initialization servlet to load data into the Pollution Monitoring application has been setup and exposed. The setup on the Provider account is complete!

Step 3: Use the data initialization servlet to initialize pollution data for consumer ABC PetroCorp

Emily, from ABC PetroCorp will now access the Pollution Monitoring application using the subscription created (already setup for you).

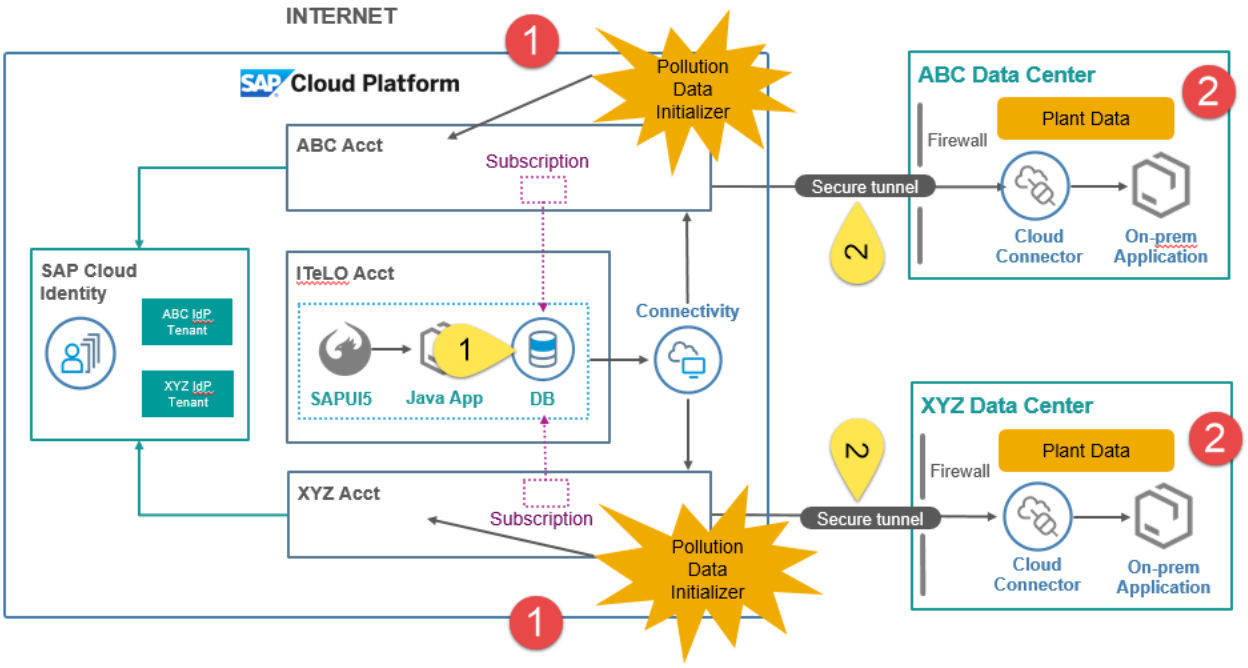
The Pollution Monitoring application, majorly deals with 2 types of data:

Pollution Data: which is read from sources like external APIs, Pollution sensors, etc.

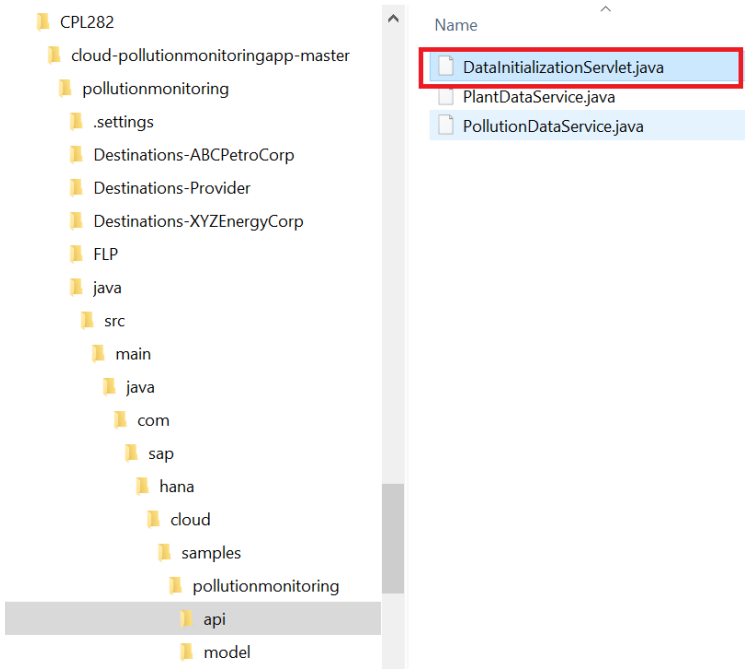
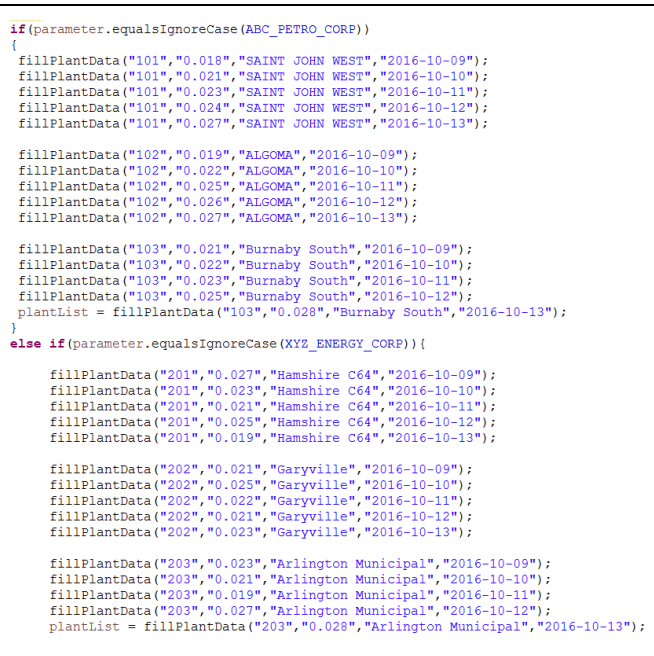
Plant Data: which is generally read from Systems of record like ERP systems, which are mostly On-Premise within the consumer private network.

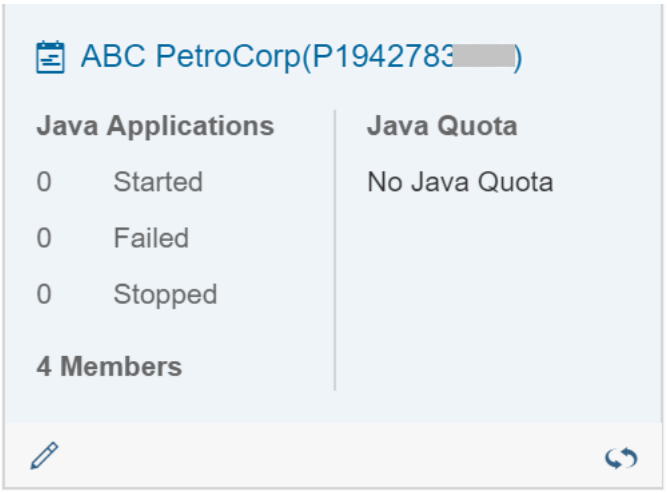
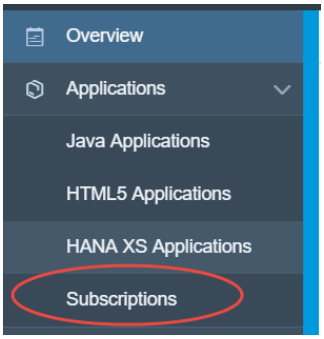
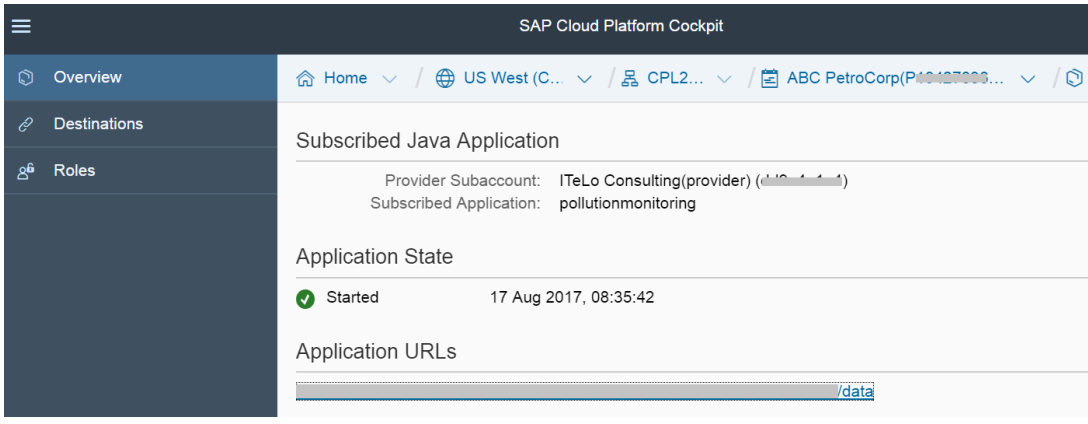
1) => In this step, you will create Pollution data using the Data Initialization servlet, which is included in the project. This data will be stored in the HANA DB on SAP CP, which is bound to the application in the provider account.

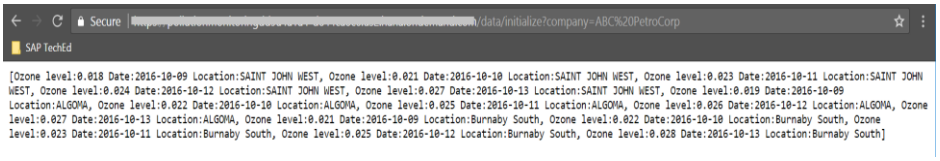
2) => The plant data is read 'on-demand' by the application, from the consumer system, i.e. it is not stored in the HANA DB on SAP CP. You will try this in Lesson 3 'On-Premise Connectivity service – On premise setup and configure connectivity service'.



In this step, you will load the Pollution Data into the subscribed application. To do this you will use the Data Initialization servlet in the project, which can be utilized to push simulated (prepared) data to the HANA database. Let us check how the prepared data looks like and how this can be initialized using the servlet.

Explanation	Screenshot
1. Navigate to ***/pollutionmonitoring/java/src /main/java/com /sap/hana/cloud /samples/pollutionmonitoring /api/DataInitializationServlet.java***.Open the java file.	
2. The prepared data for both the **ABC PetroCorp** and **XYZ EnergyCorp** can be seen in the code.	

Explanation	Screenshot
You will initialize the pollution data for **ABC PetroCorp** by calling the Data Initialization servlet	
3. Browse the URL https://account.hana.ondemand.com/ to open SAP Cloud Platform cockpit, go to the **ABC Petro Corp (P194278****)** account.	
4. Navigate to **Applications -> Subscriptions**	
5. In the Subscribed Java Application **Overview** page, copy the Application URLs.	

Explanation	Screenshot
6. In a new browser window, paste the copied URL and add **/initialize?company=ABC PetroCorp** to it. Initialization should complete and the data that was pushed to the application should be displayed.	

Result: We have now initialized the pollution data for ABC PetroCorp to ITeLO's Pollution Monitoring application.

Congratulations! You have introduced Persistence Multitenancy in our Pollution Monitoring application on SAP Cloud Platform and initialized the application with the pollution data it needs.

PART 2: IDENTITY MANAGEMENT – ASSIGN APPLICATION ROLES TO USERS IN SAP CP

For Identity management, set up of Identity authentication service tenant is already done for us. To know more how to setup this please refer here. In this part we will be assigning application roles to end users in our consumer account of SAP CP.

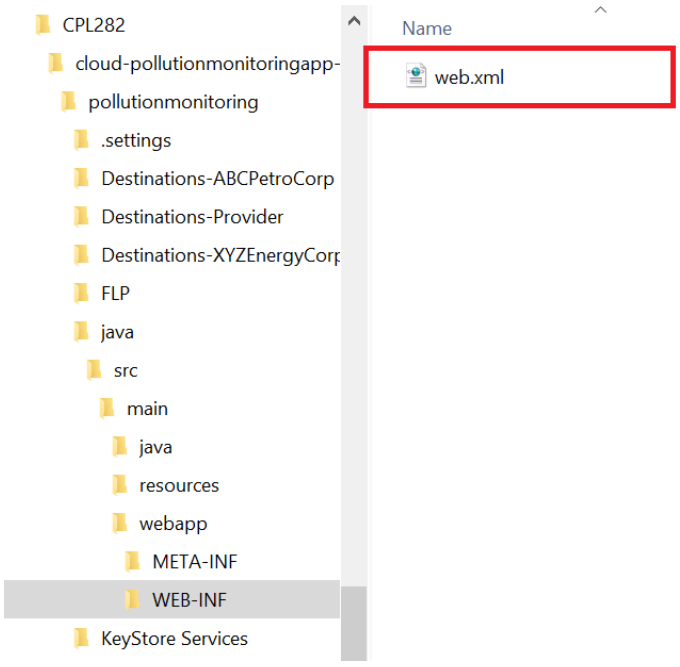
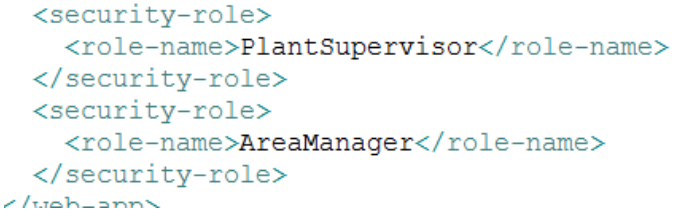
*The Pollution Monitoring multitenant application built by Robert from ITeLO Consulting (Provider), provides two predefined roles **PlantSupervisor** and **AreaManager** which control the authorization of the application and decide what the end-user actually sees in the application.*

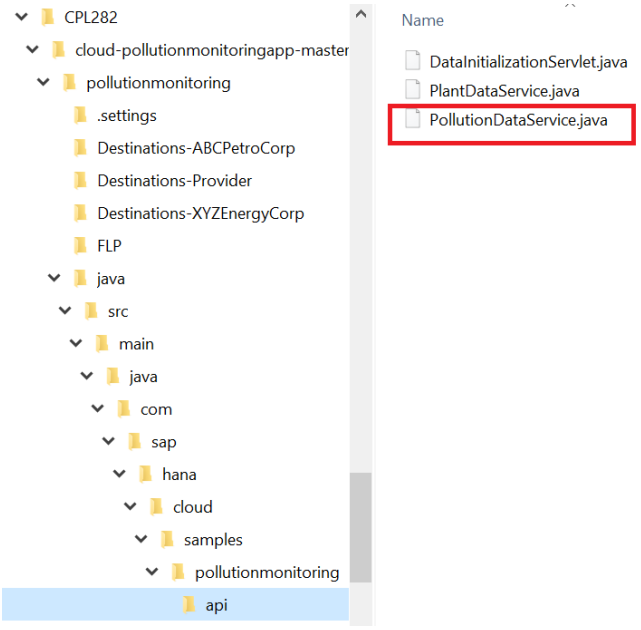
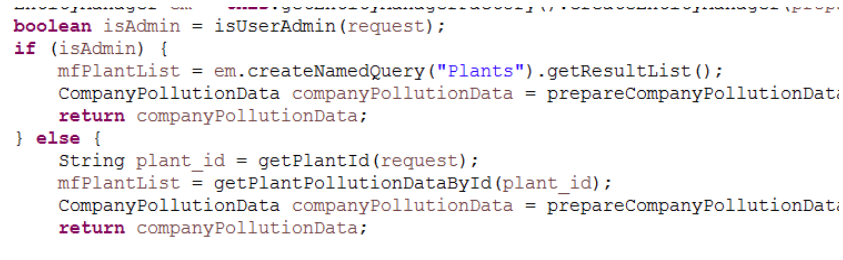
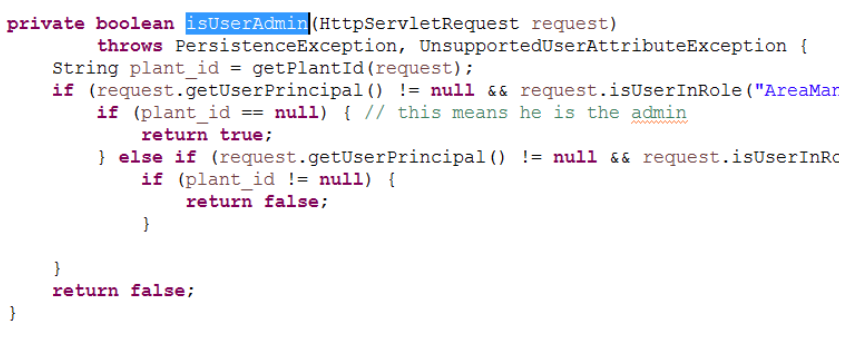
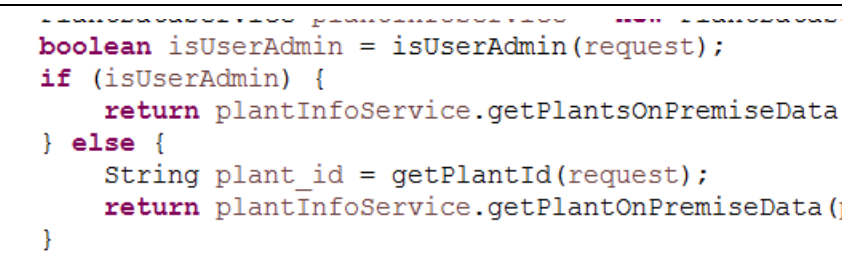
PlantSupervisor**: Users who are assigned this role will be able to view the data of **only the plant maintained in his User details (Company Information) in Identity authentication service tenant.

***AreaManager**: Users who are assigned this role will be able to view the data of **all plants in his area**.*

Let us check out how this is achieved in the code.

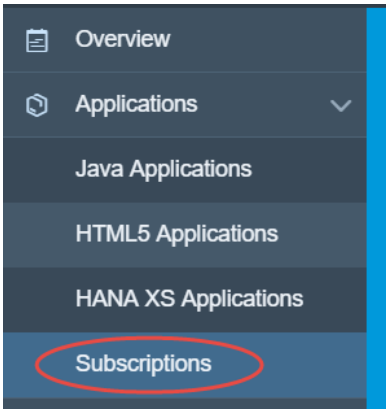
Step1: Application logic for Plant Management based upon different roles

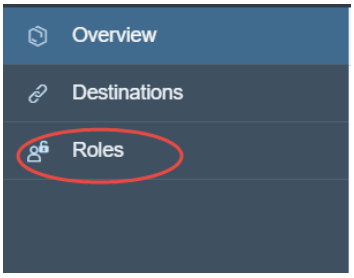
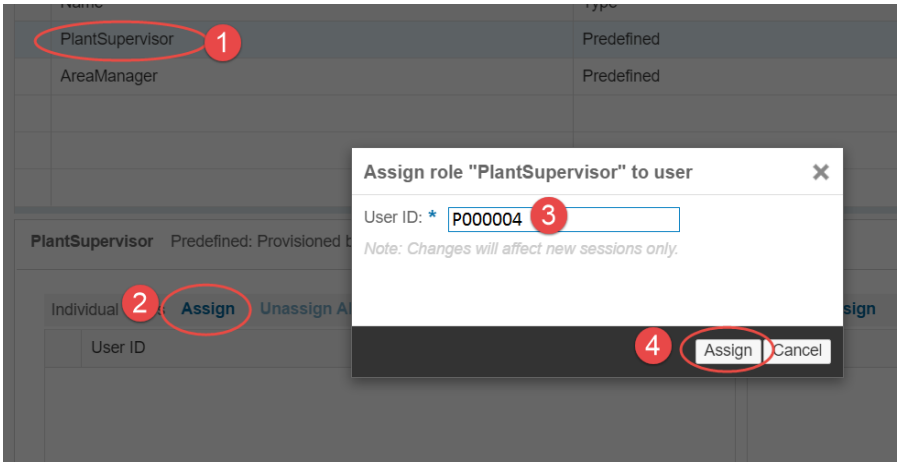
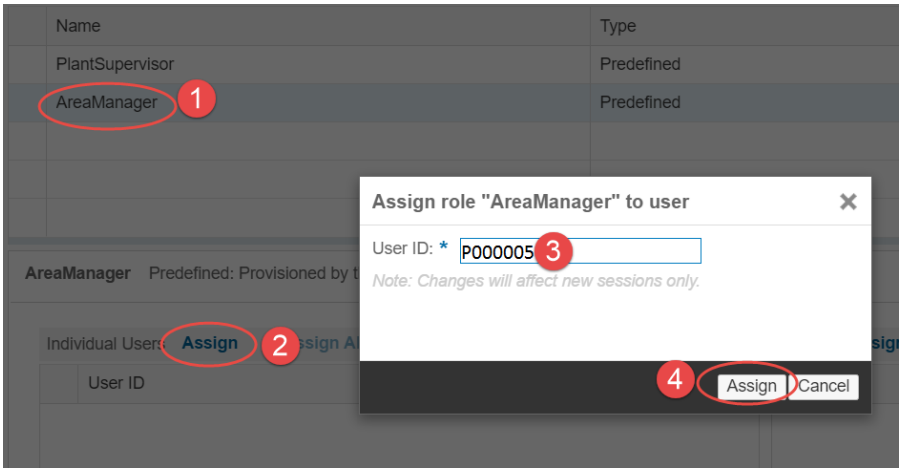
Explanation	Screenshot
1. In the imported project, navigate to **/pollutionmonitoring/java/src/main/webapp/WEB-INF/web.xml**.Open the web.xml file.	
2. Check out the two security roles created in the application.	

Explanation	Screenshot
3. Navigate to the java file **/pollutionmonitoring/java/src/main/java/com/sap/hana/cloud/samples/pollutionmonitoring/api/PollutionDataService.java**. Open the java file with notepad++.	
4. Check the method **getCompanyPollutionData().** Here we check if the user is an **admin (manager).** If yes, we show the complete Resultlist, if not we filter by **plant_id** and show only those results.	
5. For more clarity, you can also check the method **isUserAdmin()** where we check the role and decipher if he is an **Admin (manager)** or not.	
6. A similar logic is also followed for retrieving the **plant data** from the on-premise systems .	

Step2: Roles assignment in SAP Cloud Platform cockpit

On the consumer side, Emily will now have to assign these roles to the Area Managers and Plant Supervisors in her company ABC PetroCorp i.e. ABCPlantSupervisor and ABCAreaManager.

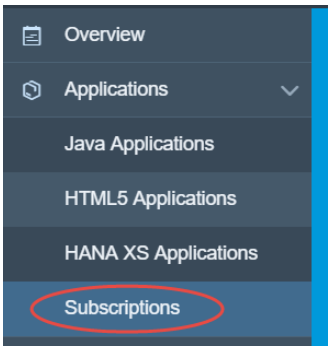
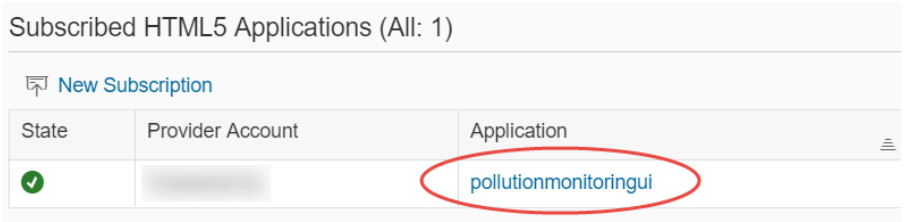
Explanation	Screenshot
1. In the SAP CP cockpit of the consumer, ABC PetroCorp, navigate to **Applications - > Subscriptions**.	
2. Click on the **pollutionmonitoring** subscribed Java application, provided by ITeLO Consulting.	

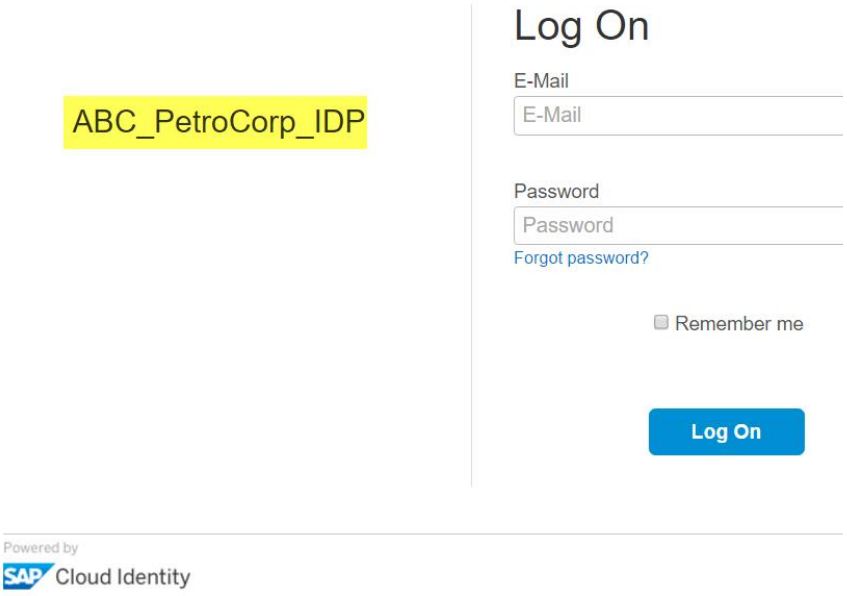
Explanation	Screenshot
3. In the application overview page, navigate to **Roles**. The list of application roles will be displayed.	
4. Select the **PlantSupervisor** role and click on Assign. In the pop-up enter the User ID **P00004** of the ABCPlantSupervisor and click on **Assign**.	
5. Similarly, select the **AreaManager** role and click on Assign. In the pop-up enter the User ID **P000005** of the **ABCAreaManager** (noted in the previous blog after import) and click on Assign.	

Completed! We have successfully mapped the application roles to the corresponding users of **ABC PetroCorp** from Identity authentication service tenant.

Step3: Validation of Identity Management setup

Emily can now check if the setup of the SAP CP account and the application with the corporate IDP, in our case Identity authentication service tenant, works as expected.

Explanation	Screenshot
1. In the SAP CP cockpit of the consumer, ABC PetroCorp, navigate to **Applications -> Subscriptions.**	
2. Click on the link of the subscribed HTML5 application **pollutionmonitoringui**	

Explanation	Screenshot
3. Copy the **Application URL** link in the Active Version section.	
4. Open a browser in *Incognito mode/Private mode*, paste the URL and click on Go.The login page from the application configuration in SAP Cloud Identity is shown with the correct name.	

You should be able to use the User ID of the **AreaManager** or **PlantSupervisor**

Area manager:
User ID Johan
Password Test@12345

PlantSupervisor:
User ID Smith
Password Test@12345

and login to the application and use it without any issues. *(Although, you may find that you do not find any data as the setup is still not complete. This is okay!)*

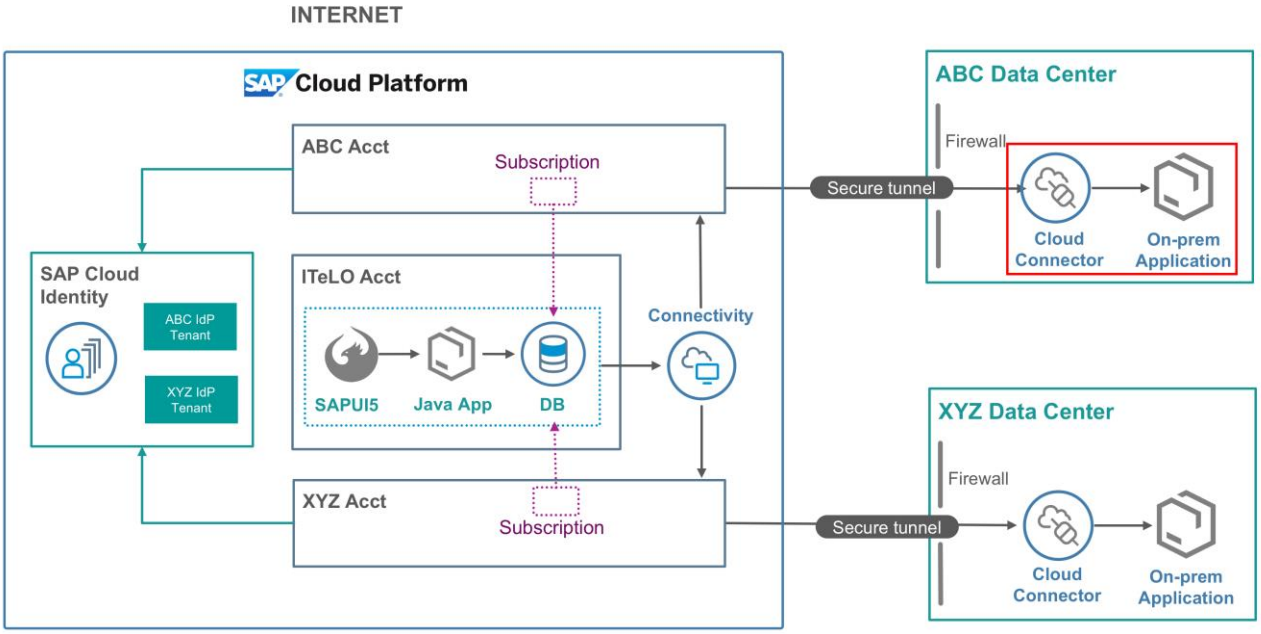
PART 3: ON-PREMISE CONNECTIVITY SERVICE – ON PREMISE SETUP AND CONFIGURE CONNECTIVITY SERVICE

'ITeLO Consulting' is a trusted SAP partner company that specializes in providing solutions for the **Oil and Gas industry**. They have adopted the **SAP Cloud Platform** as the innovation platform to build and run their applications on the cloud

As it is already outlined, ABC PetroCorp –premise service running within their landscape that provide information about its plants – details like no. of workers at each plant, area of the plant etc. The pollution monitoring shows this information in the dashboard.

To achieve this, the on-premise service need to be exposed from their landscape and provided to the cloud application. This can be achieved using the SAP CP Connectivity service

Let's understand how Emily will set up the connectivity between the on-premise services running within ABC PetroCorp network and the application running in the ABC PetroCorp (Consumer)account. Refer to the section highlighted in red below.



Emily will perform the following steps.

Step1: Expose on-premise services securely using SAP Cloud Connector

Step2: Configure connectivity between exposed on-premise services and the cloud application using SAP CP destinations

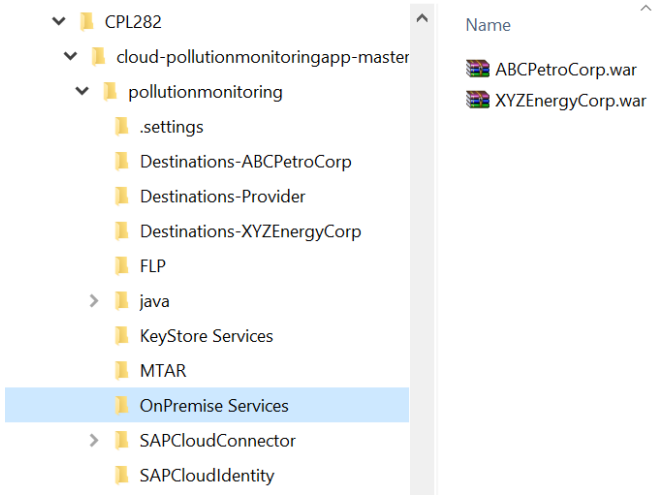
Step 1: Expose on-premise services securely using SAP Cloud Connector

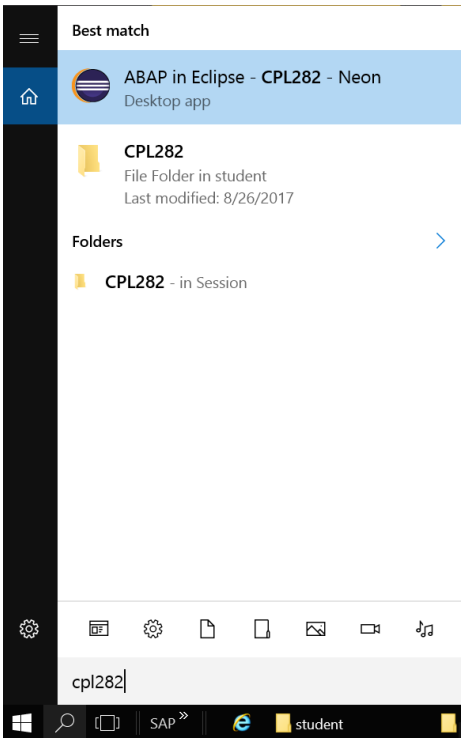
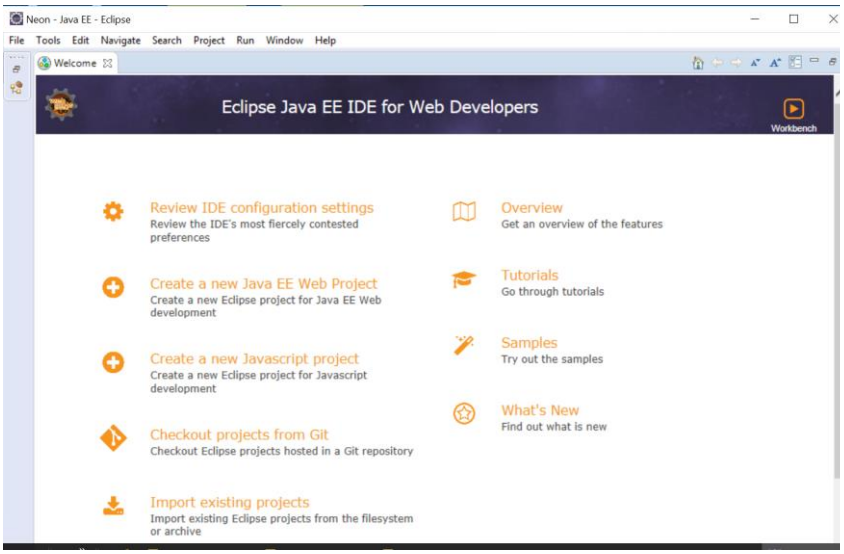
Note: To make it easy to execute the tutorial, we will simulate the on-premise setup by running a service locally on a Tomcat server installed on our machine. And then use the SAP Cloud Connector installed on our machine to securely expose this service to the cloud.
In practice, Emily will work with on-premise services running within the company landscape.

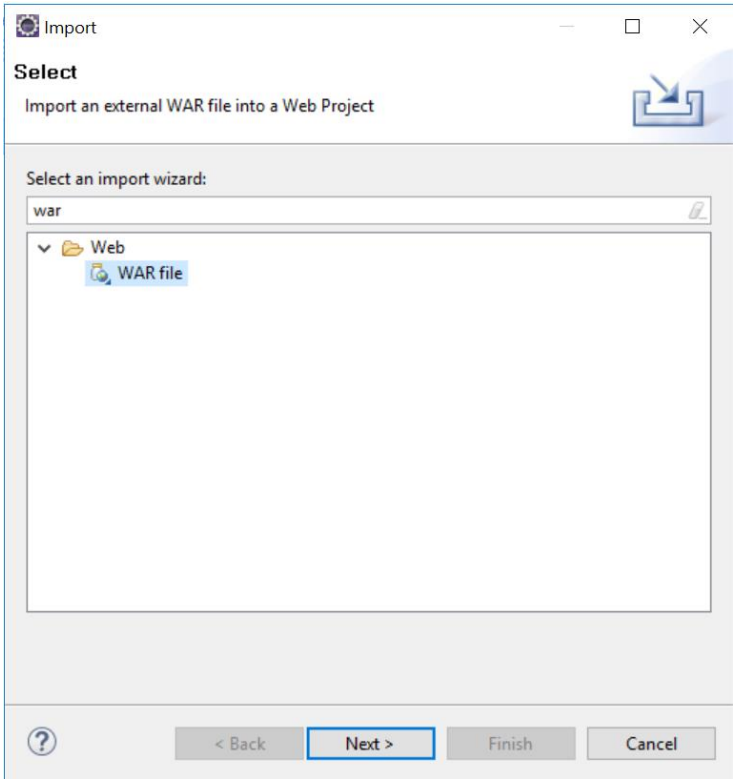
Section 1: Simulate on-premise setup by running a mock service on your machine

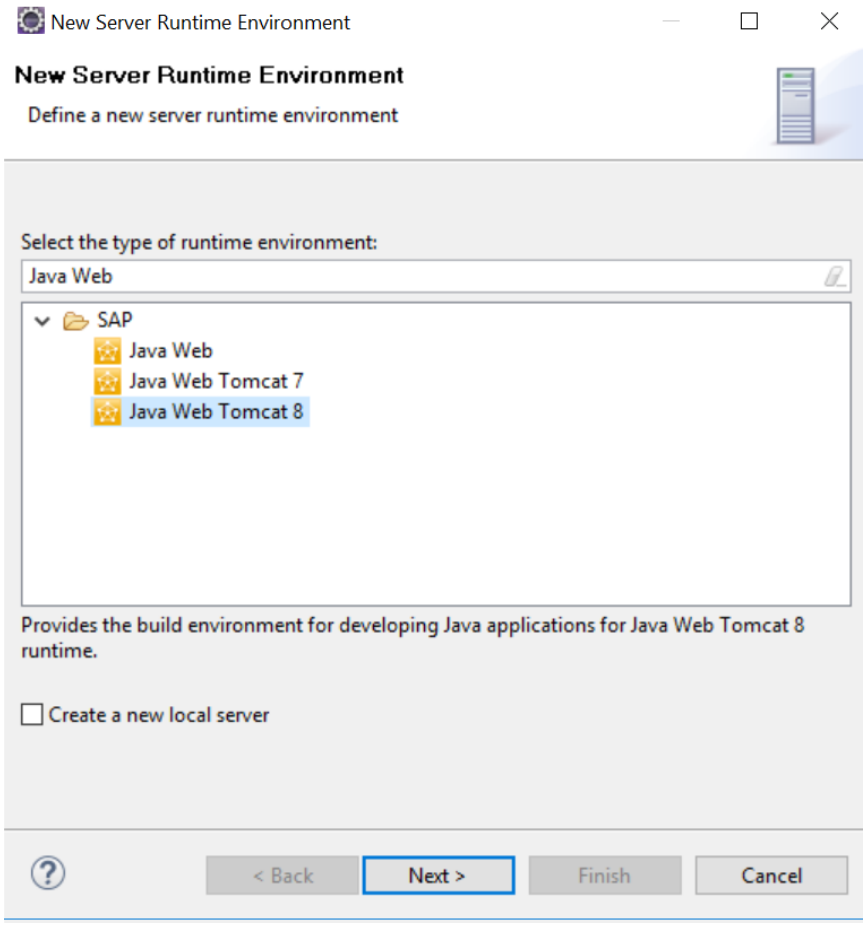
A mock service is available in the Git repository. In this step, you will import, deploy and run this mock service on a local Java Web Tomcat 8 server.

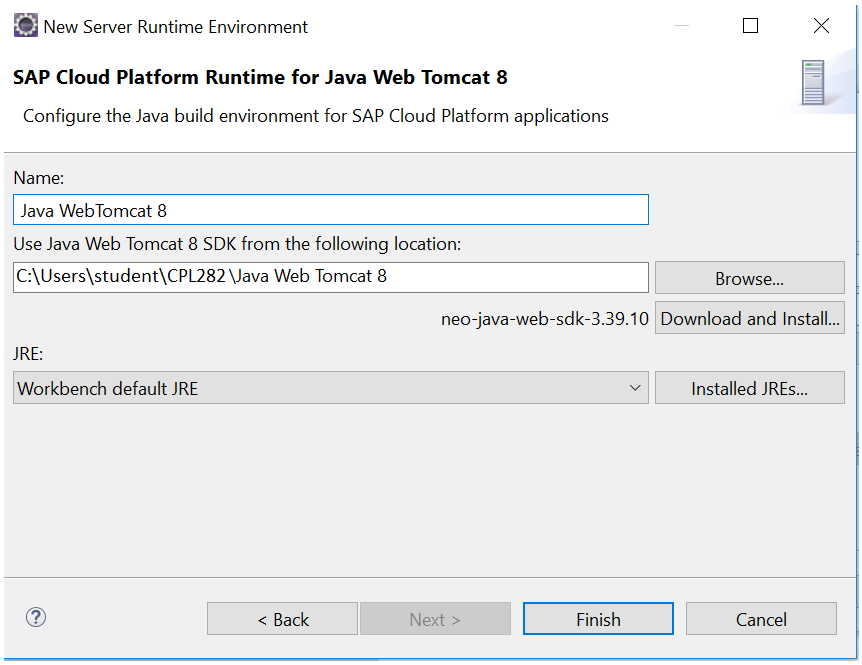
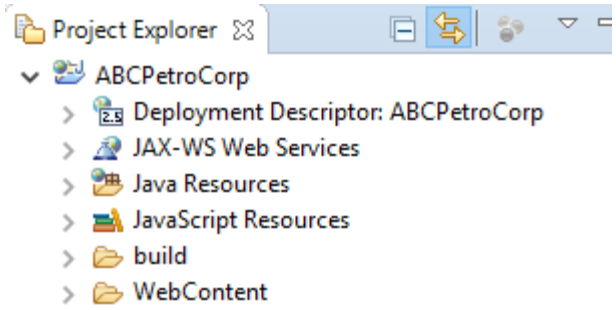
Import project containing mock service into Eclipse

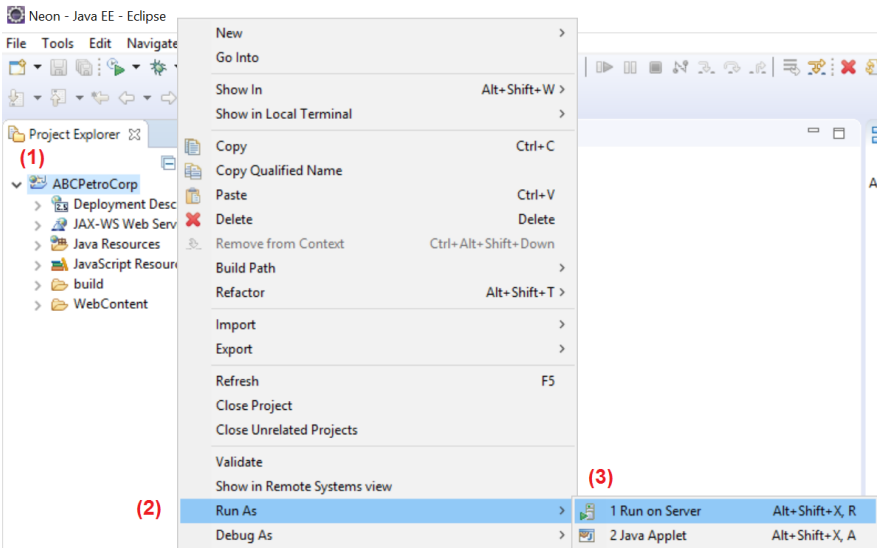
Explanation	Screenshot
1. The mock service has been uploaded as a WAR file under **On-Premise Services** folder (see screenshot).	

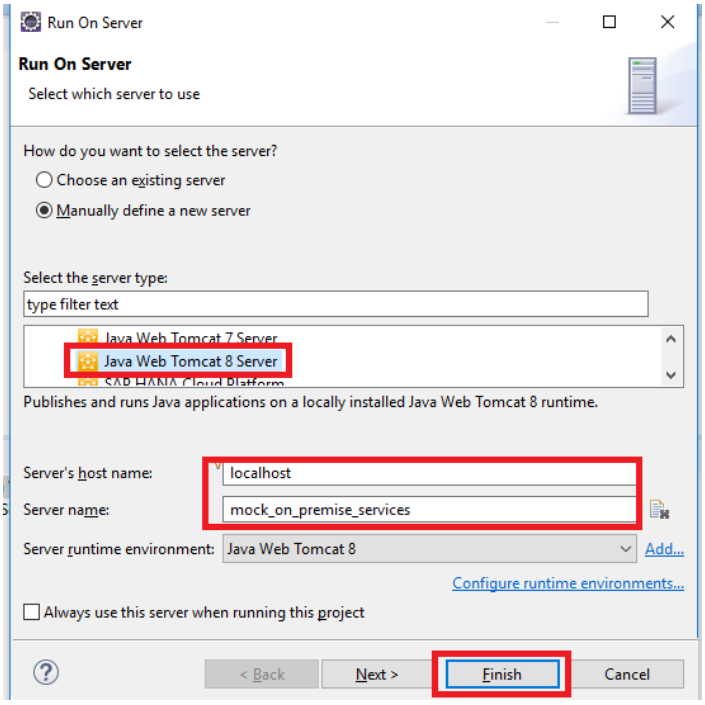
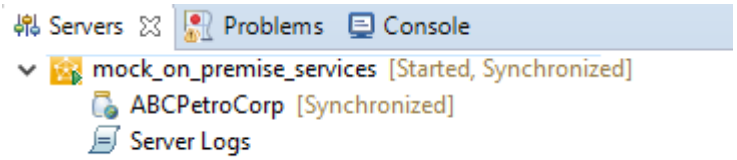
Explanation	Screenshot
2. Open the eclipse by searching **cpl 282** in the Search windows and select the Best match **ABAP in eclipse-CPL282-Neon** as shown in the screenshot.	
3. This will open the eclipse in **Java EE perspective** with a **Welcome View**. Close the Welcome view.	

Explanation	Screenshot
4. Click on File>Import option in and search **war** in an search bar to select an import wizard. Click on **Next.**	

Explanation	Screenshot
5. In the WAR Import dialog, provide the following info: **WAR file:** File location of the **ABC PetroCorp.war** file **Web project:** ABCPetroCorp 6. To select **Target runtime** click on **New** and search **Java Web** in the search bar. 7. Select **Java Web Tomcat 8** server under **SAP** Folder and click on **Next**.	

Explanation	Screenshot
8. Browse for Java Web Tomcat 8 SDK stored on your system. **Note: The Zip file of JAVA Web Tomcat 8 will be available at** **Studentshare/CPL282/JavaWebTomcat8.zip** **Copy this file and paste into C:\Users\student\CPL282 and extract the content.** In case if it is not available please download from this site. https://tools.hana.ondemand.com/#cloud under **SAP Cloud Platform Neo Environment** section. Click on **Finish**.	
9. After a **successful import**, in the Project Explorer view, the project structure should look like this.	
Deploy and run the mock service on local Java Web Tomcat 8 server	

Explanation	Screenshot
10. Deploy imported **on-premise service** on to local server (Java Web Tomcat 8). In the Project Explorer, right click on **ABCPetroCorp** node. Select **Run As>Run on Server**.	

Explanation	Screenshot
11. In the Run on Server dialog, choose “Manually define a new server”. Enter the following details: **Server’s host name:** localhost **Server name:** mock_on_premise_services Click on **Finish**.	
12. Once the deployment has been done, you should see the following structure under **Servers** view.	
13. This mock service can be accessed at the URL **http://localhost:8080/ABCPetroCorp/** **Note:**It could be possible that your server **does not run on port 8080**. In that case please check the port number of your server and execute **http://localhost:<your port number>/ABCPetroCorp**.	This would return: *{“plants”:[[{“numberOfWorkers”:”7543”,”country”:”CANADA”,” crudeOilCapacity”:”7857 gallons”,”plantId”:”101”,”location”:”New BrunsWick”,”plantArea”:”1212 Sq feet”,”plantName”:”LPG Production”},{“numberOfWorkers”:”10000”,”country”:”CANADA”,”crudeOilCapacity”:”20000 gallons”,”plantId”:”102”,”location”:”Alberta”,”plantArea”:”5252 Sq feet”,”plantName”:”Oilfield Production”},{“numberOfWorkers”:”8000”,”country”:”CANADA”,”crudeOilCapacity”:”18000 gallons”,”plantId”:”103”,”location”:”British Columbia”,”plantArea”:”4846 Sq feet”,”plantName”:”Gas Processing and Compression Plant”}]],”companyName”:[“ABC PetroCorp, CANADA”]}*

Section 2: Expose the mock service using SAP Cloud Connector

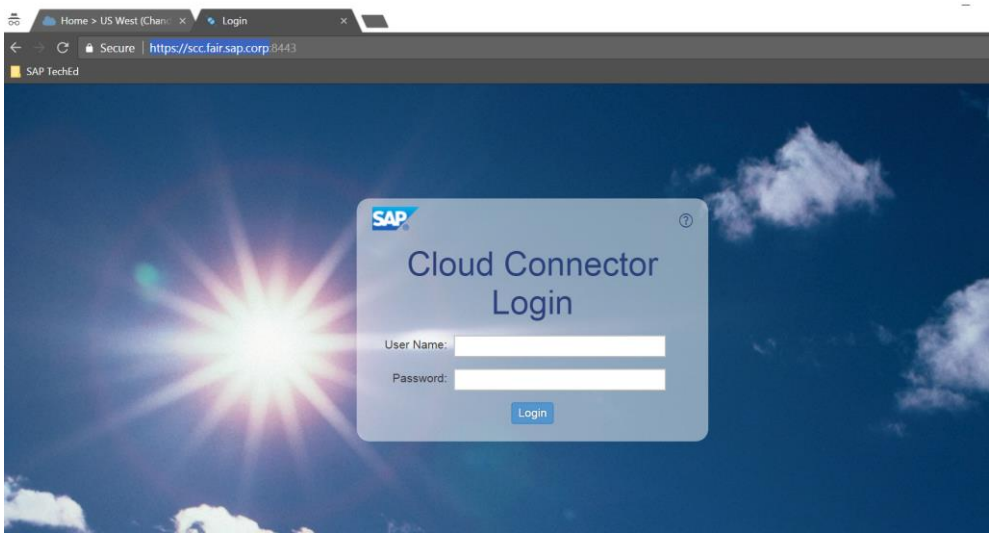
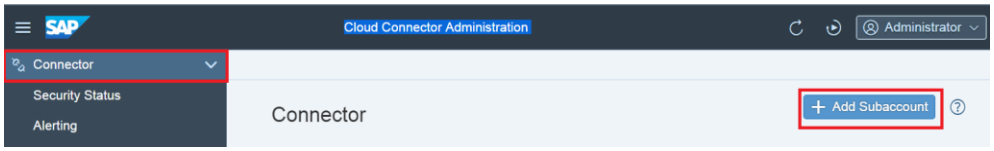
The ABC Petro Corp mock service can be exposed to the application running on SAP CP in a secure and easy way using SAP Cloud Connector.
For executing this tutorial easily, the SAP Cloud Connector will be installed on your local machine.
In the actual scenario, Emily will install and configure the Cloud Connector within the company landscape.

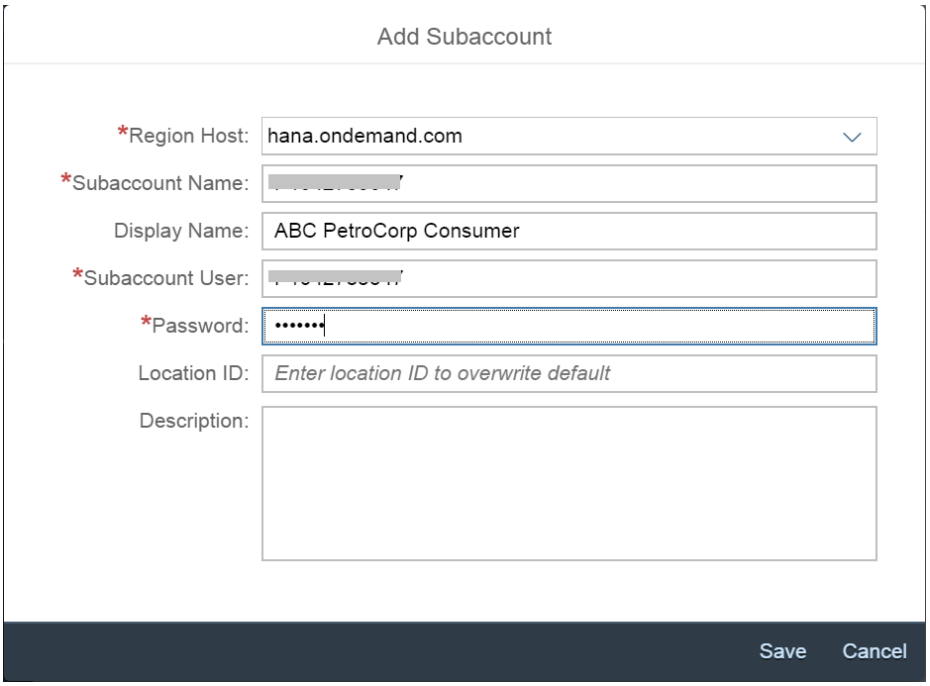
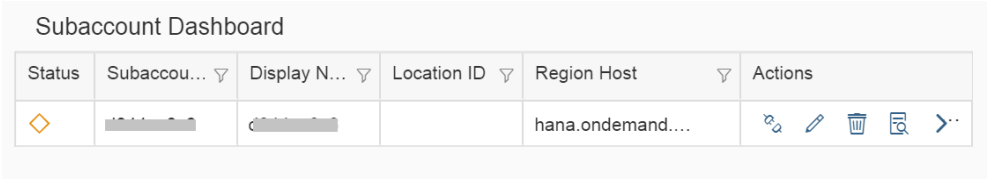
Pre-requisites:

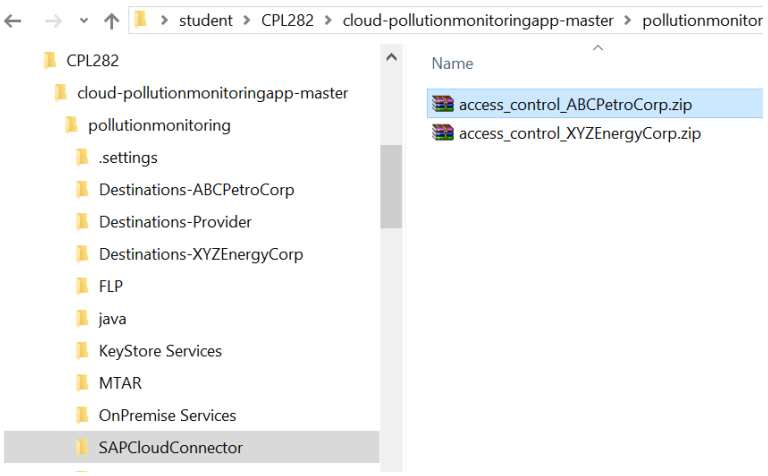
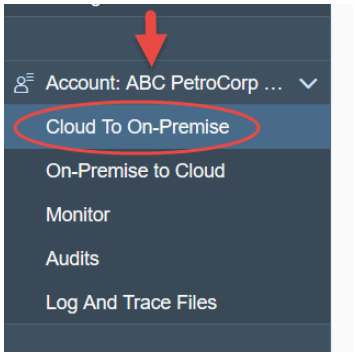
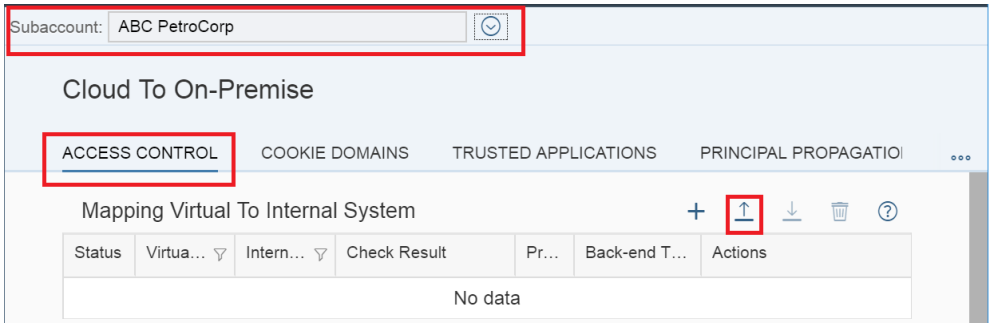
1. You have SAP HANA Cloud Connector installed
2. You have performed the initial configuration on SAP Cloud Connector

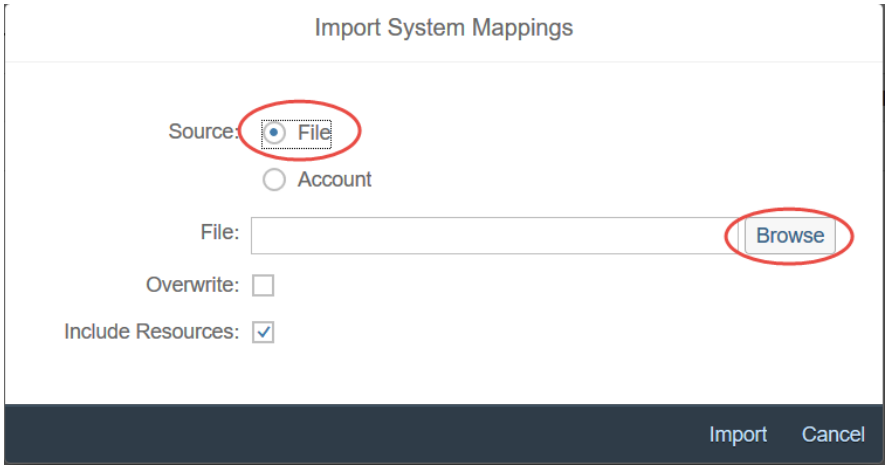
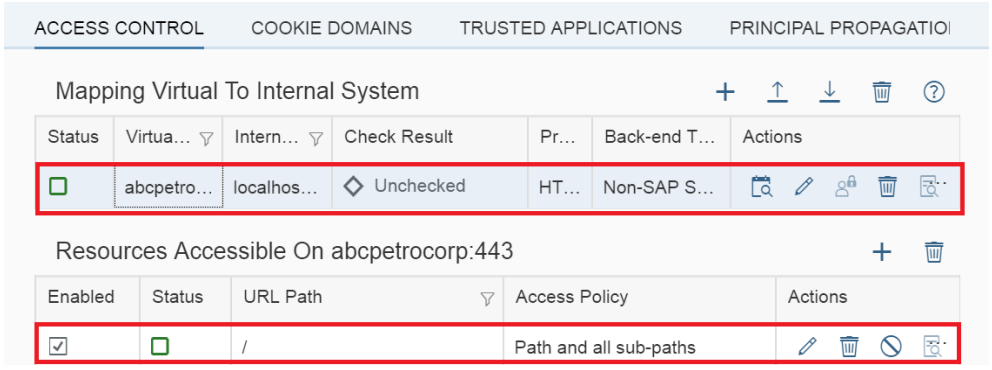
Configuring Access Control for consumer account

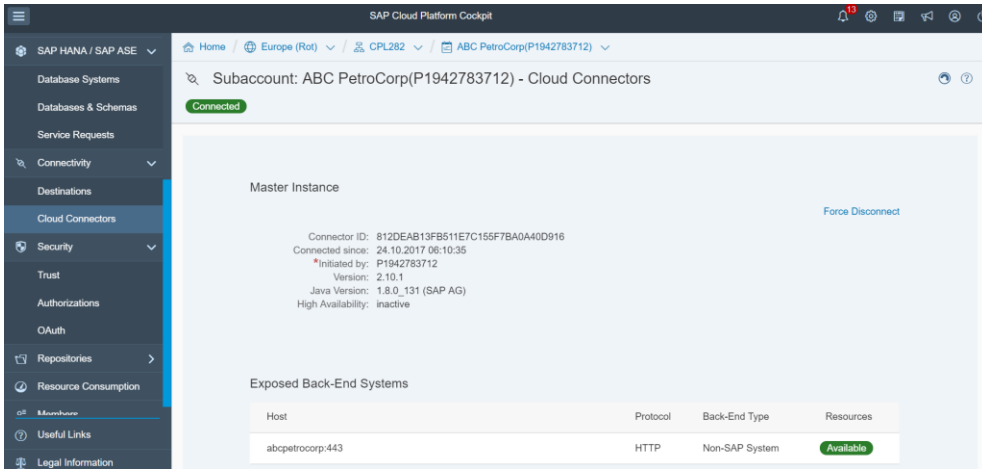
The internal on-premise system will be exposed using a virtual system. This virtual system then will be used in the HTTP destination configuration in SAP SAP Cloud Platform in the next step (Step 2). More info on configure access control can be found here.

Explanation	Screenshot
1. Log in to the SAP Cloud connector with URL https://<hostname>:<port> https://scc.fair.sap.corp:8443 **User Name:** Administrator **Password:** welcome	
2. Add the SAP CP account to which you want to expose the on-premise mock services. Here we are connecting to the **ABC PetroCorp (Consumer)** Navigate to the **Connector** tab and click on **Add Subaccount**	

Explanation	Screenshot
3. Enter details of the **ABC PetroCorp (Consumer)** account and click Save. **Region Host**: hana.ondemand.com **Subaccount Name:** <Subaccount Name> To find subaccount name, switch to your **SAP Cloud Platform Cockpit** and navigate to **overview** page of your **ABC PetroCorp** subaccount. You will find subaccount name ABC PetroCorp(P1942783xxx) under **Subaccount Information** **Subaccount User:** P1942783xxx **Password:** Welcome17 Click on **Save.** Use HTTPS Proxy Host: proxy Port: 8080	 
4. Once added, it will be listed in the **Subaccount Dashboard**.	

Explanation	Screenshot
5. Next step is to configure **access control**. There has to be a mapping created between the local system and the virtual system. This has been made available as **access_control_<<consumer name>>.zip file** under the **SAPCloudConnector** folder. You can import this file to configure **access control** in the SAP Cloud connector as described in the upcoming steps.	
6. Navigate to **Cloud To On-Premise** section and under the **ABC PetroCorp** Account.	
7. Check that the **Subaccount** dropdown shows the correct account and go to **ACCESS CONTROL** tab. Click on **Import** corresponding under the **Mapping Virtual to Internal System** section as shown in the screenshot.	

Explanation	Screenshot
8. This opens **Import System Mappings** dialog. Select **Source as File** and choose **Browse**. Select the location of the access_control_ABCPetroCorp.zip' file and click on **Import**.	
9. This will import the access control setting on to your SAP Cloud connector. **Note:** The **access_control_ABCPetroCorp.zip** file assumes port number **8080** for the on-premise service. In case your mock service runs on different port, then Edit the mapping with the correct port number.	
10. Check that **the access control** has been configured correctly as shown in the screenshot.	
11. In the SAP CP cloud cockpit for the **ABC PetroCorp** (P19442783xxx) account, navigate to the	

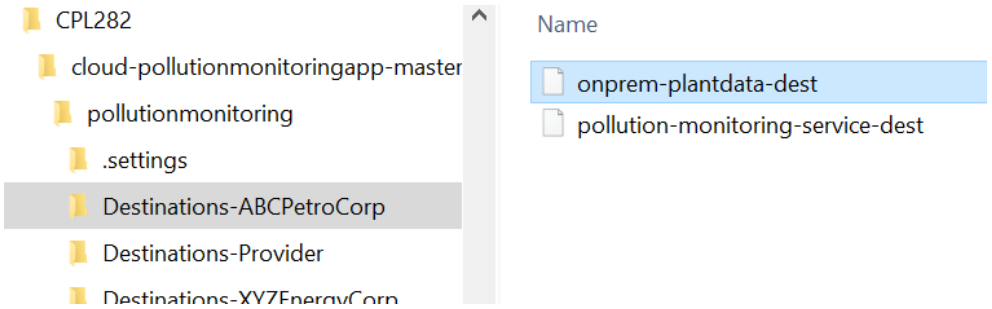
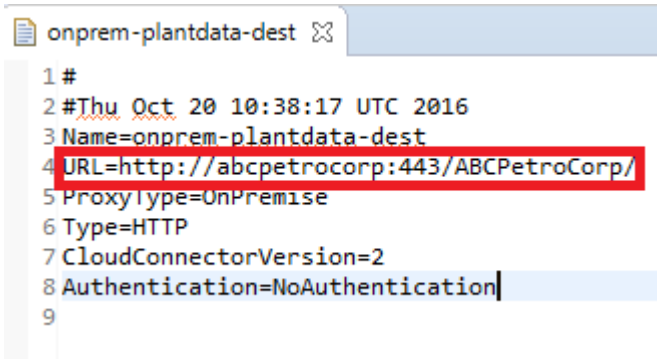
Explanation	Screenshot
Connectivity>Cloud Connectors tab. You would see the status of the SAP Cloud Connector as shown in the screenshot.	

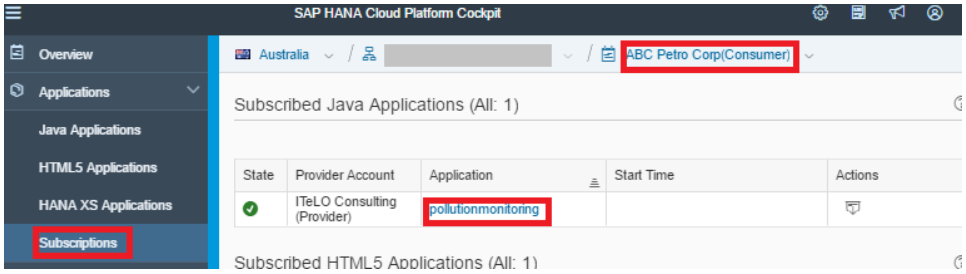
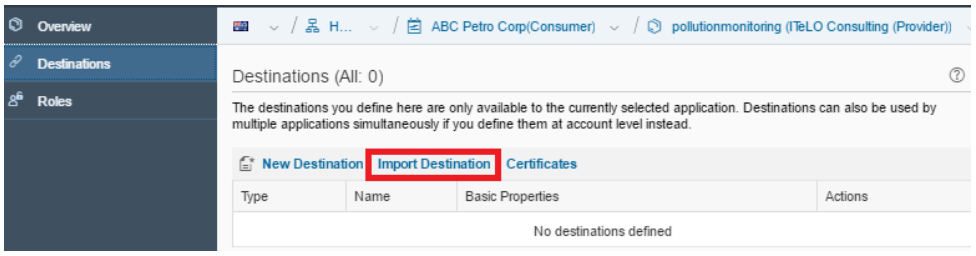
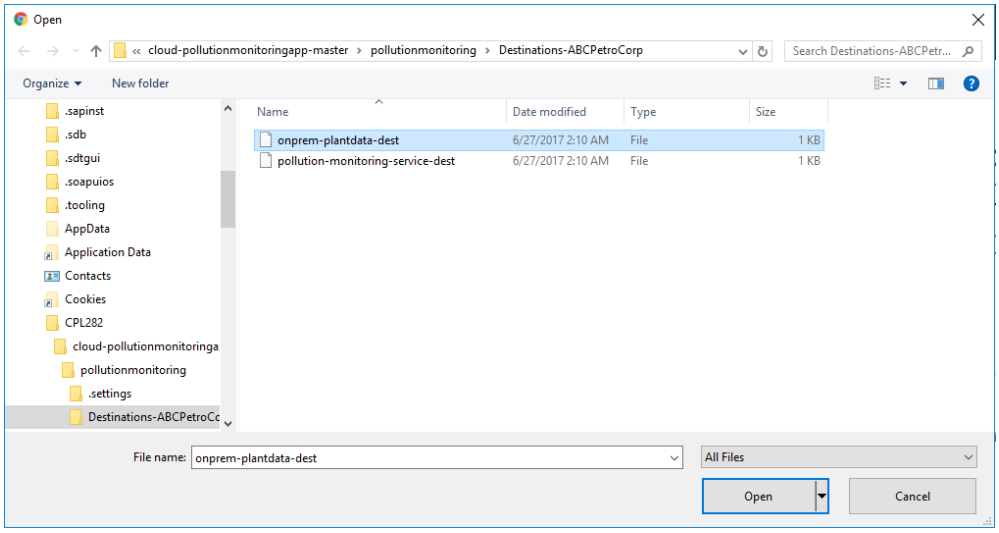
With this step, the mock **on-premise service** have been exposed to the **ABC PetroCorp** (Consumer) account.

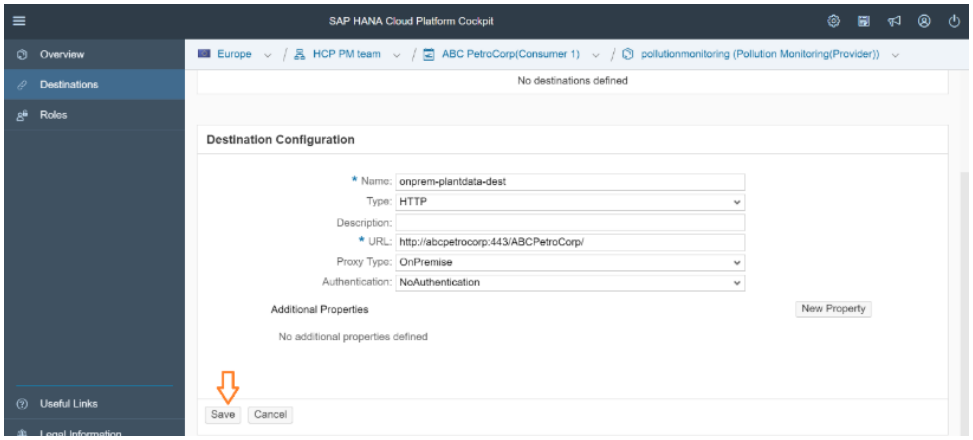
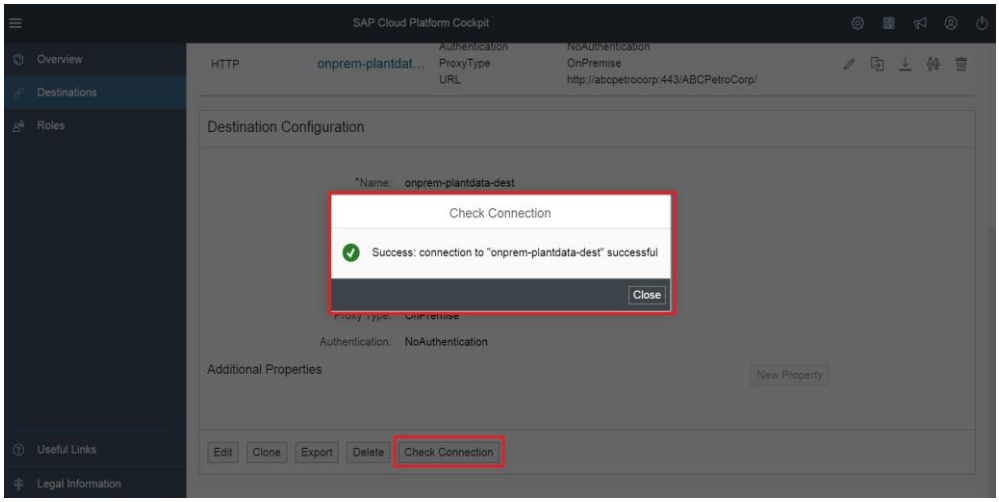
Step 2: Configure connectivity between exposed on-premise services and the cloud application using SAP CP destinations

In this step, the Java cloud application on SAP CP will connect to the exposed mock on-premise services through HTTP destinations in SAP CP.

Section 1: Configure Destinations from SAP CP

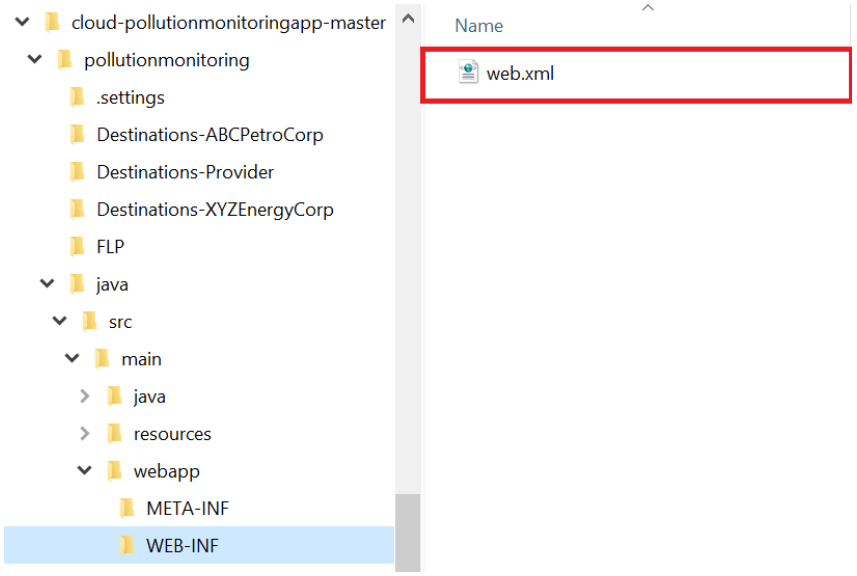
Explanation	Screenshot
1. There is a destination file **onprem-plandata-dest** file that is available under the folder **Destinations-ABCPetroCorp**	
2. This destination points to the mock **on-premise service URL** which has been already exposed to the **ABC PetroCorp** Subaccount in the previous step. This destination needs to be uploaded to the **ABC PetroCorp** (Consumer) Subaccount as shown in the steps below.	

Explanation	Screenshot
3. Navigate to **Subscriptions** pane of **ABC PetroCorp** (Consumer) Subaccount and click on **pollutionmonitoring** Java application subscription	
4. Navigate to **Destinations** pane of your subscribed **pollutionmonitoring** Java application and click on **Import Destination**.	
5. Select the file location of **onprem-plandata-dest** file from your Project.	

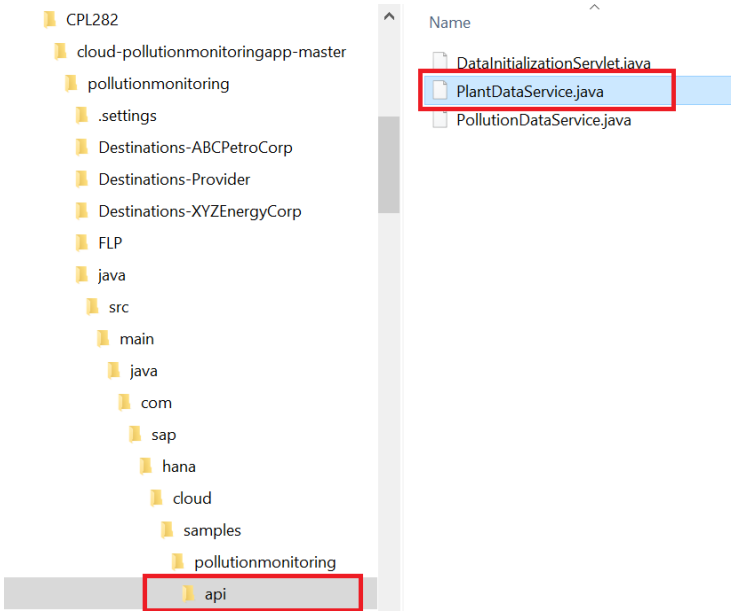
Explanation	Screenshot
6. The destination would be added in your Subaccount as shown in the screenshot click on **Save**.	
7. Verify that on click on **Check Connection** you will get following message in a dialog **Success: connection to 'onprem-Plant-dest' successful**.	

Section 2: Access the on-premise service from the cloud Java application

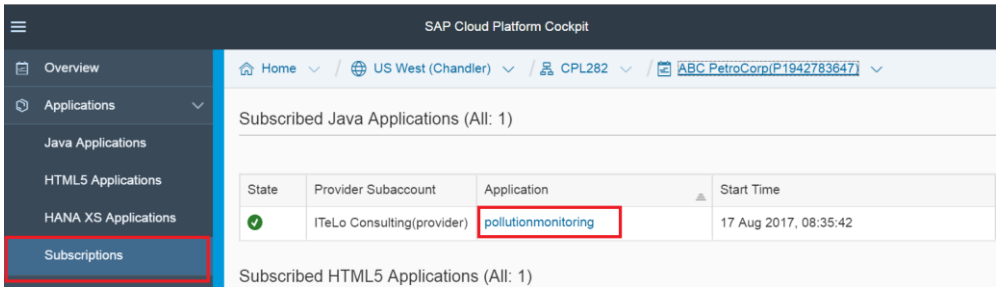
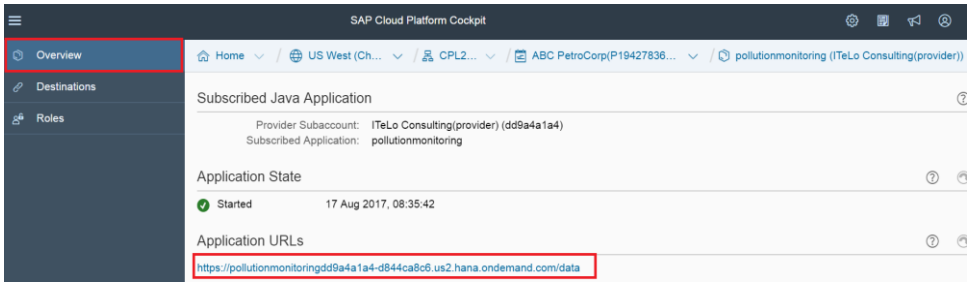
If the on-premise connectivity setup is working fine, then the **on-premise** service would be accessible via the pollutionmonitoring Java application.

Explanation	Screenshot
1. Navigate to **/pollutionmonitoring/java/src /main/webapp/WEB-INF/web.xml**. Open the web.xml file with notepad++	

2. In the Java application, the destination **onprem-plantdata-dest** has been configured in the **web.xml** file.	**web.xml**
	
	`<resource-ref>`
	`<res-ref-name>onprem-plantdata-dest</res-ref-name>`
	`<res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>`
	`</resource-ref>`

Explanation	Screenshot
3. Navigate to **/pollutionmonitoring/java/src /main/java/com/sap/hana/cloud/samples/pollutionmonitoring/api** to edit PlantDataService.java with notepad++.	

4. This destination **onprem-plantdata-dest** is being looked up in the **PlantDataService.java**.
 	
Once the destination look up has been done, all calls from the Java code to the mock service are triggered via this destination.
	PlantDataService.java
	
	`ConnectivityConfiguration configuration = (ConnectivityConfiguration) ctx`
	
	`.lookup("java:comp/env/connectivityConfiguration");`
	`// Get destination configuration for "destinationName"`
	`DestinationConfiguration destConfiguration = configuration.getConfiguration("onprem-plantdata-dest");`

Explanation	Screenshot
5. We will test this service call from the Java application. in the **ABC PetroCorp** subaccount, navigate to the **Subscriptions** tab click on the **Subscribed java application** pollutionmonitoring.	
6. From the **Overview** pane, copy the **pollutionmonitoring** Java Application URL.	
7. Append **'/api/v1/plantdata'** to the the URL and run the service call in a browser. The URL will look like: https://pollutionmonitoring<provider account id>-<consumer account id>.<<landscape_host>>/data/api/v1/plantdata The plant details returned from the mock **on-premise services** (invoked by the Java application) are shown in the browser.	

That's it! The pollution monitoring application in SAP CP is connected to the on-premise services running within the **ABC PetroCorp** network. The important point to note here is that the connectivity was established at the subscription(tenant) level.

In the next part, we will look at connecting the SAP CP's HTML5 application to the Java service running on the SAP Cloud Platform

PART 4: CONFIGURE A CONNECTIVITY SERVICE AND TEST THE MULTITENANT APPLICATION

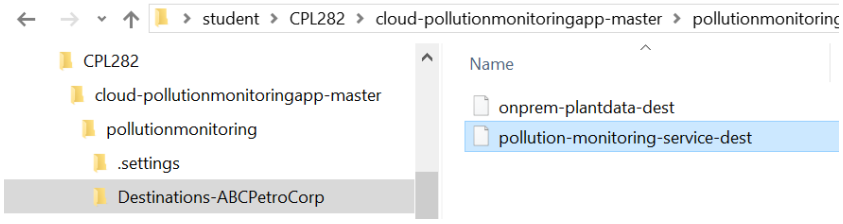
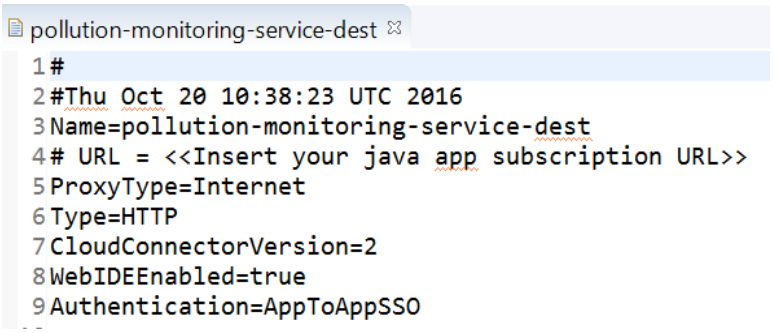
Step 1: Configure connectivity on the consumer account (to service running on SAP CP)
Step 2: Test the Pollution Monitoring dashboard by login to HTML5 Application

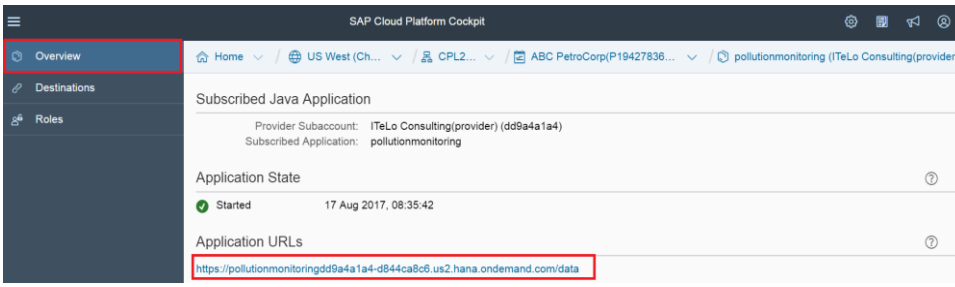
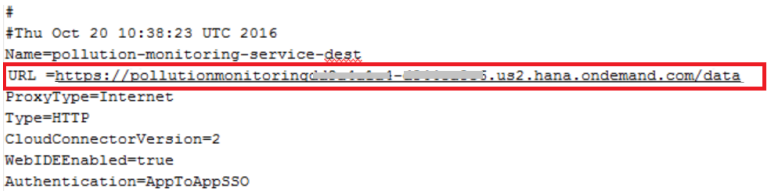
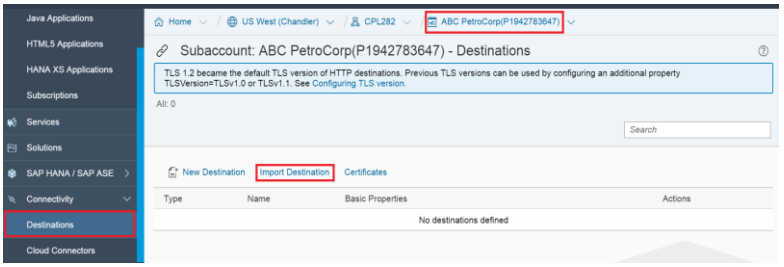
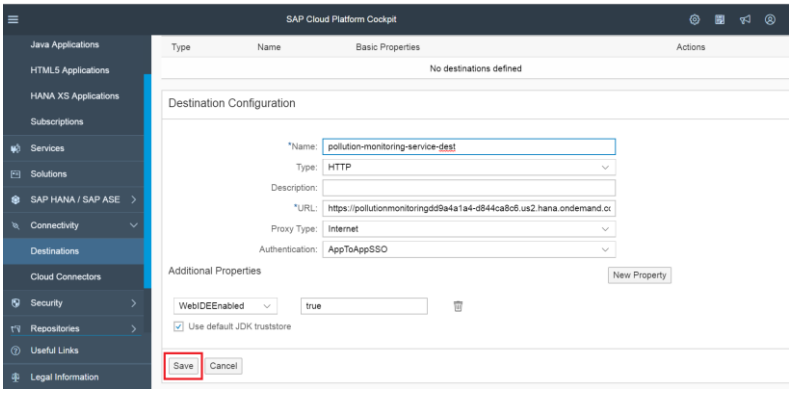
Let's look at these steps in detail.

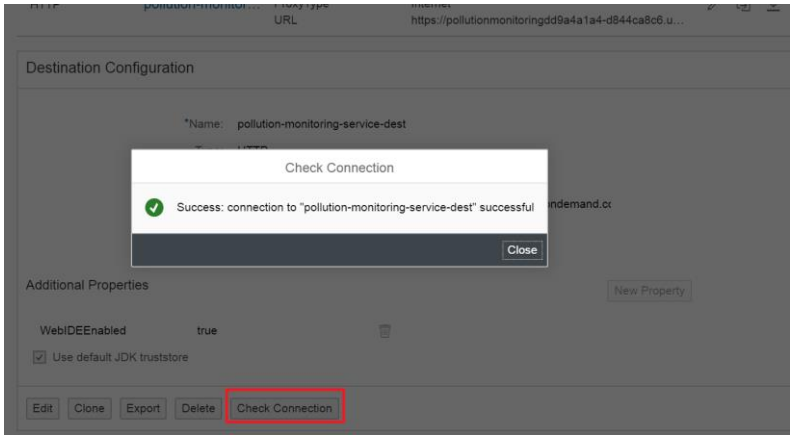
Step 1: Configure connectivity on the consumer account (to service running on SAP CP)

In consumer account, HTML5 application uses services for displaying plant and pollution data from Java application. Hence, there is need to configure destination in the consumer account that points to subscribed Java application. This destination will be consumed by the HTML5 application running in same consumer account.

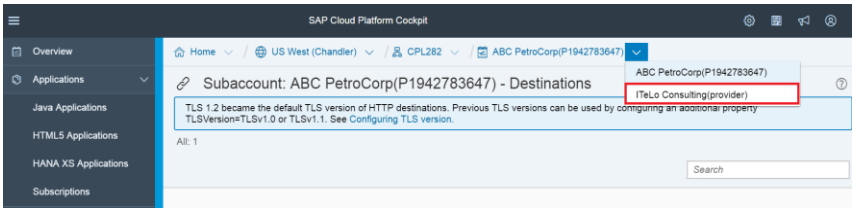
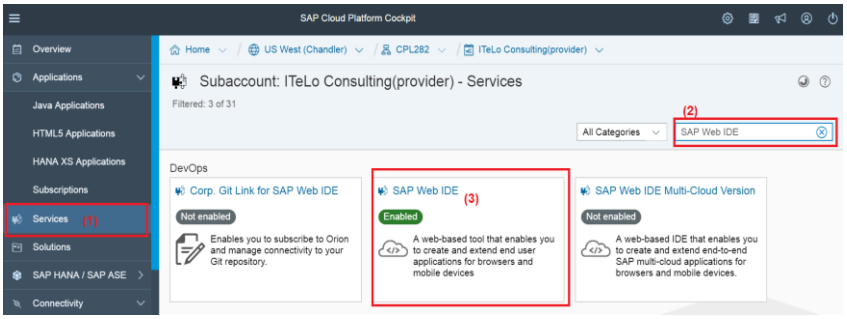
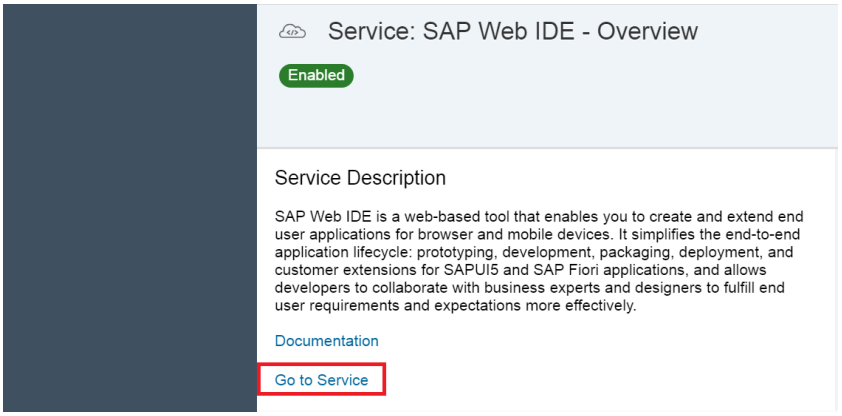
Let's understand how Emily will configure this destination that points to the subscribed pollution monitoring Java application running in **ABC PetroCorp** Subaccount.

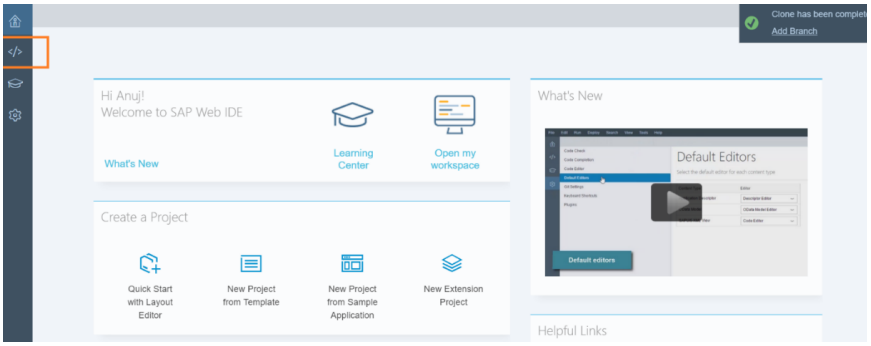
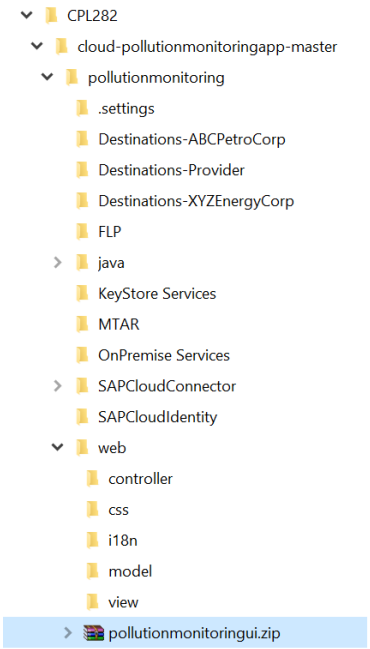
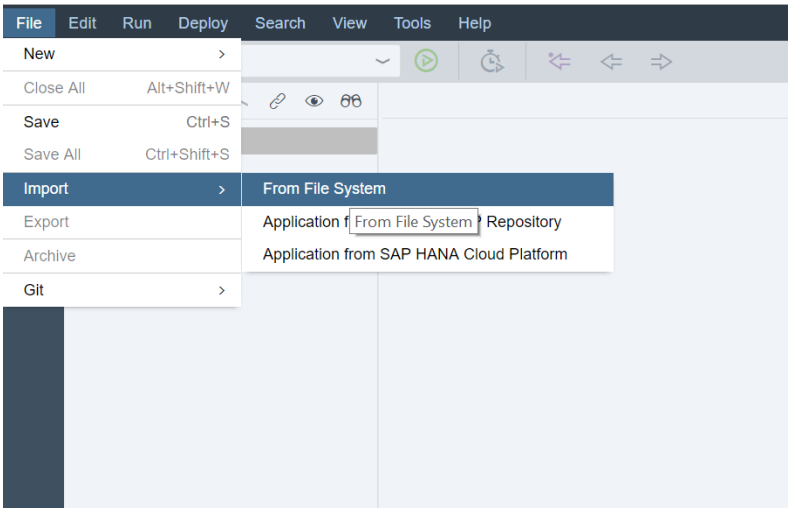
Explanation	Screenshot
1. There is a destination file **pollution-monitoring-service-dest** file that is available under the folder **Destinations-ABCPetroCorp.**	
2. Here, the destination URL points to the **pollutionmonitoring** Java application which **ABC PetroCorp Subaccount** is subscribed to. Since this destination is to be consumed by HTML5 Application, **WebIDEEnabled** is **true** and **Authentication is AppToAppSSO** (Application to Application Single Sign on) **URL** needs to be copied from your Java application subscribed URL as mention in the **next step.**	

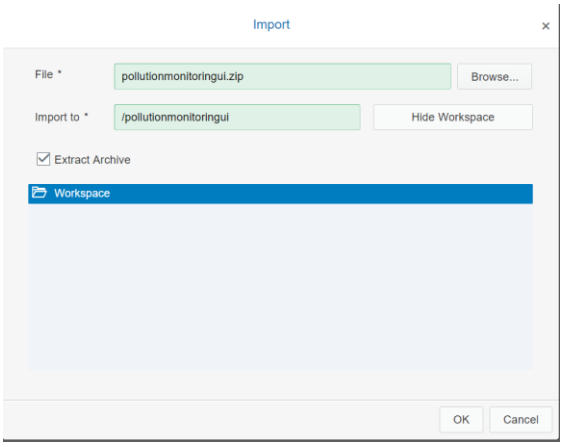
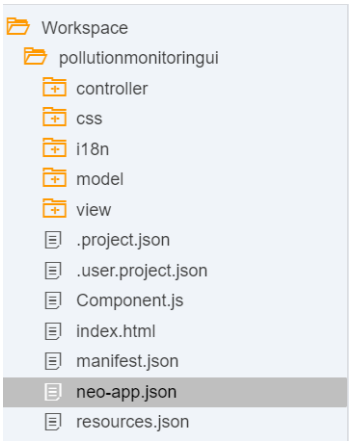
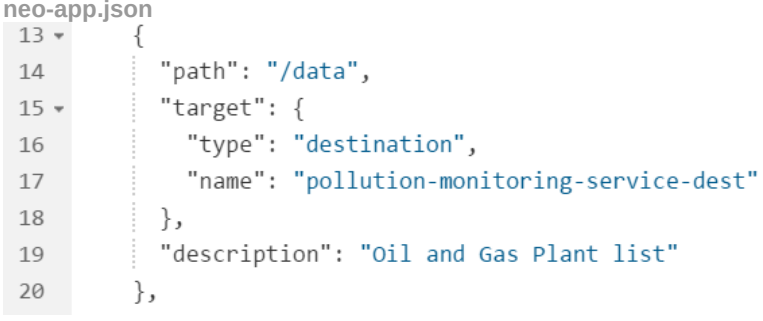
Explanation	Screenshot
3. Copy the **pollutionmonitoring** Java application subscribed URL from Application URLs section in overview pane.	
4. Paste this URL into the **URL** parameter and **remove** # in the **pollution-monitoring-service-dest** file**.**	
5. Navigate to **Destinations** pane of your **ABC PetroCorp Subaccount** click on 'Import Destination'.	
6. Select the file location of **pollution-monitoring-service-dest** file. The destination would be added in your account as shown in the screenshot click on **Save.**	

Explanation	Screenshot
7. Verify that on click on **Check Connection** you will get following message in a dialog **Success: connection to 'pollution-monitoring-service-dest' successful**.	

Let's understand that how **pollutionmonitoringui** HTML5 application consume subscribed **pollutionmonitoring** Java application. For this you need to import the **pollutionmonitoringui** into the SAP Web IDE workspace.

Explanation	Screenshot
1. Switch to **the ITeLo Consuting (provider) Subaccount** through dropdown in the breadcrumb.	
2. Navigate to **Services** pane of **ITeLo Consulting (Provider) Subaccount** and search for **SAP Web IDE** service and click on **SAP Web IDE**.	
3. Click on **Go to service.**	

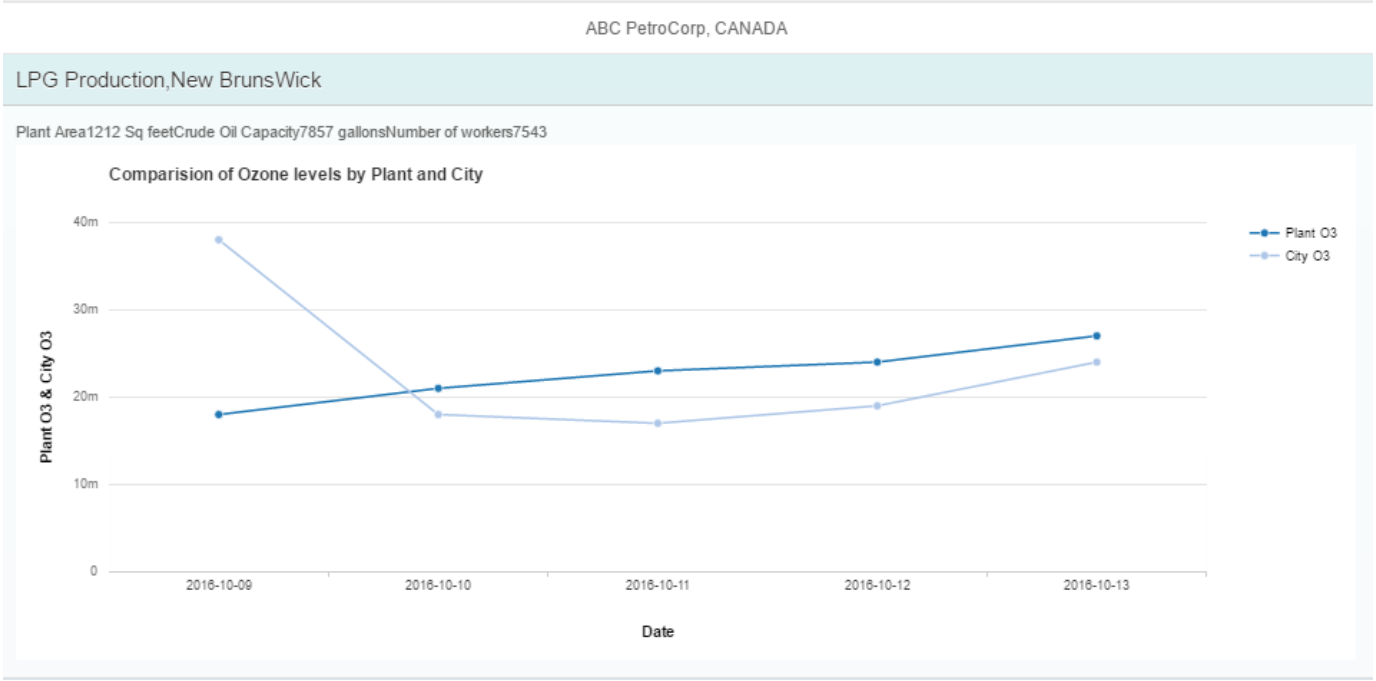
Explanation	Screenshot
4. **SAP Web IDE** Editor opens in new Tab of the same browser. 5. Click on the **"</>"** Development pane.	
6. Navigate the location of HTML5 app **pollutionmonitoringui.zip** file defined under the folder **web.**	
7. Import the **pollutionmonitoringui** HTML5 app using File->Import->From File System.	

Explanation	Screenshot
8. Click on Browse and choose the **pollutionmonitoringui.zip** file from the location mention above. Import to /**pollutionmonitoringui** folder under Workspace. Select **Extract Archive** check Box. Click on **OK.**	
9. HTML5 app **Pollutionmonitoringui** imported into the SAP WEB IDE workspace and looks like this.	
10. Open the **neo-app.json** file.	
11. The destination **pollution-monitoring-service-dest** is defined in the **neo-app.json** file(between line number 13 to 20) which invokes **pollutionmonitoring** Ja va application.	neo-app.json 

Step 2. Test the Pollution Monitoring dashboard by login to HTML5 Application

Emily login to **pollutionmonitoringui HTML5 Application** by following instructions of Part 2-Step3 **(Validating the Identity Management setup)** and test whether Pollution Monitoring dashboard

- displays plant and pollution data
- compare pollution (ozone) level by plant and city
- authorize users to display UI based upon their roles (all plants or single plant information)



PART 5: FIORI LAUNCHPAD – CONFIGURING TILE FOR MULTI-TENANT APPLICATION

Robert has created the SAP Fiori Launchpad (FLP) site for the pollution monitoring dashboard application running on the provider account. He has also exported this site for his customers.More information on this can be found here.

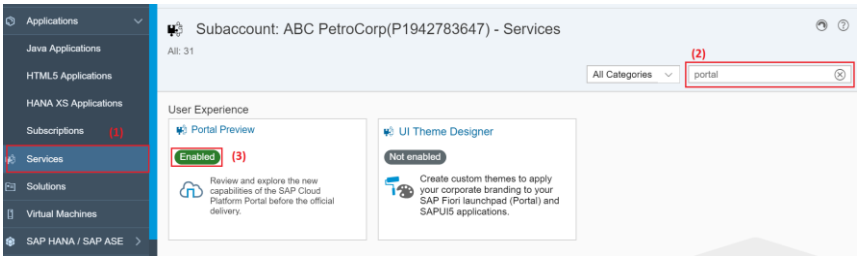
Emily now can import and run this SAP FLP site into ABC PetroCorp subaccount and will perform the following steps:

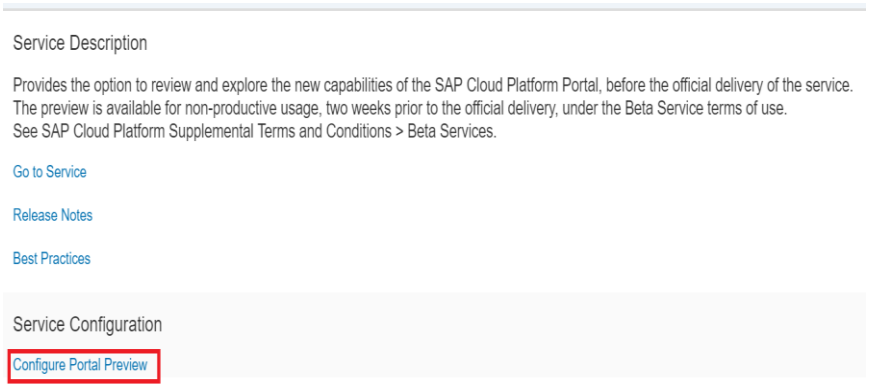
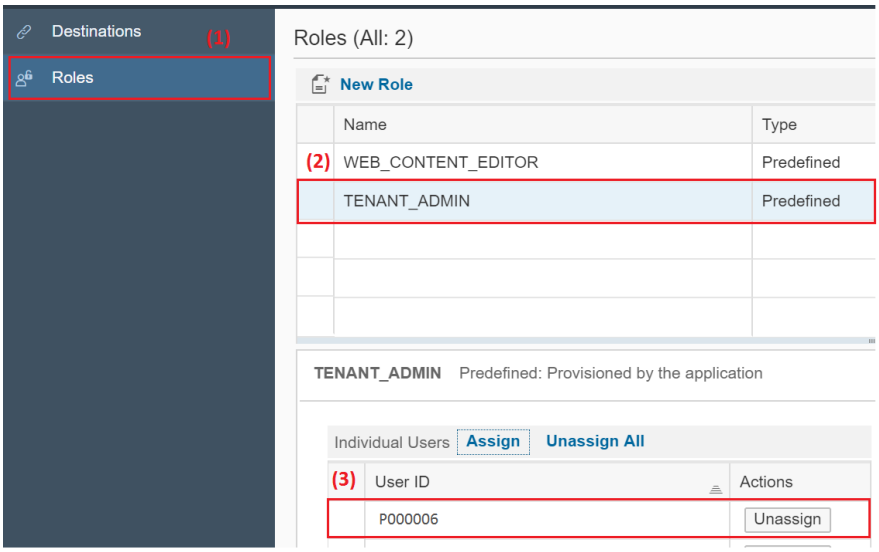
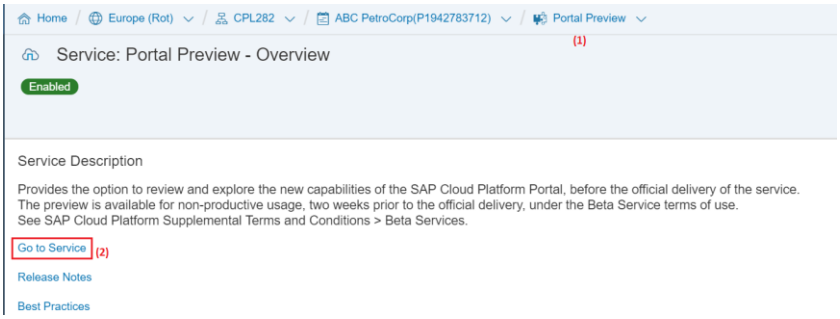
Step 1: Login to SAP CP Portal Service and Import SAP Fiori Launchpad content into ABC PetroCorp Subaccount

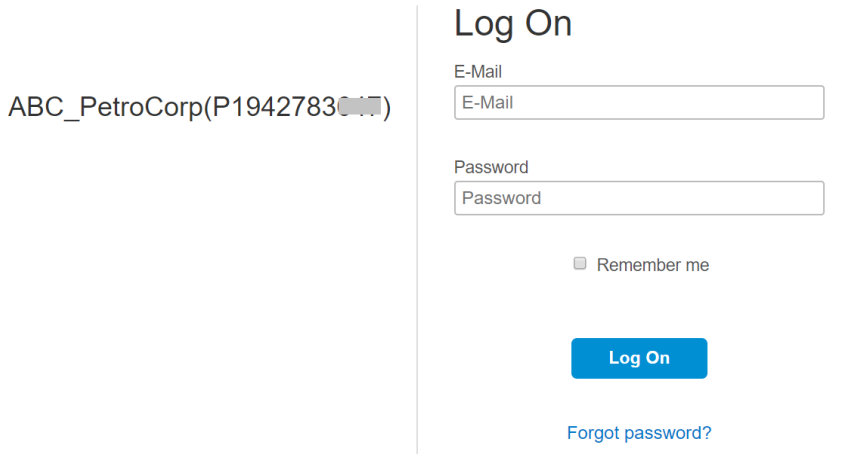
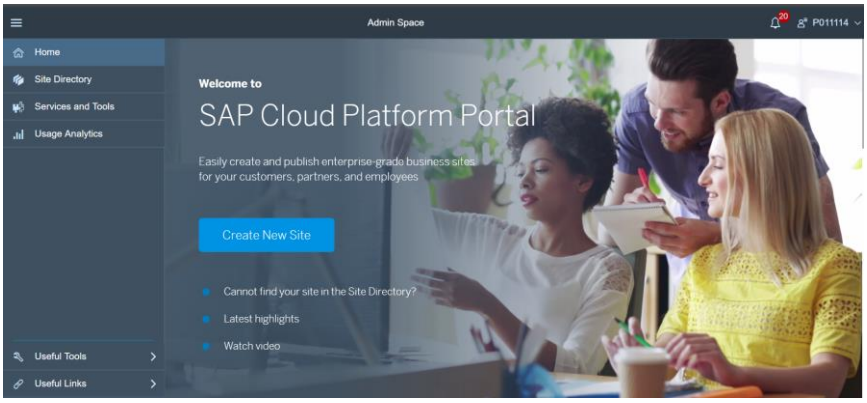
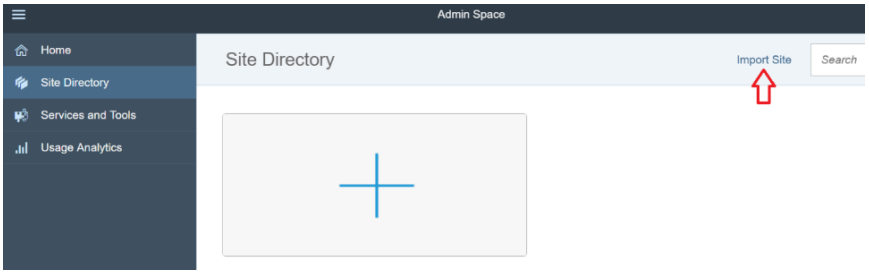
Step 2: Publish and Launch SAP Fiori Launchpad site from the ABC PetroCorp Subaccount

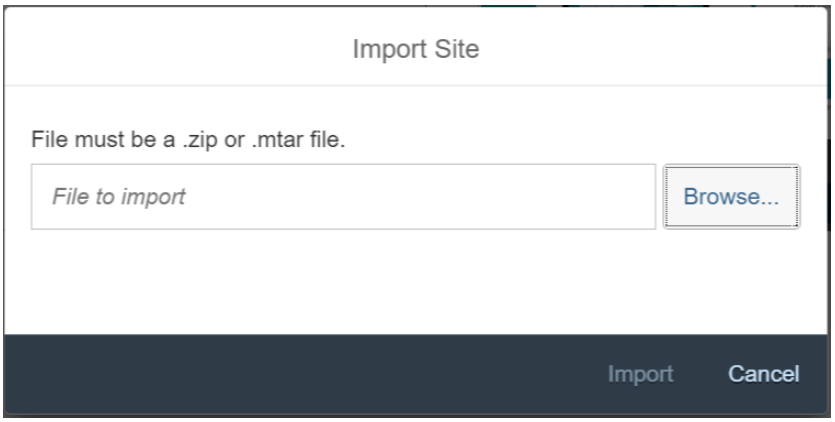
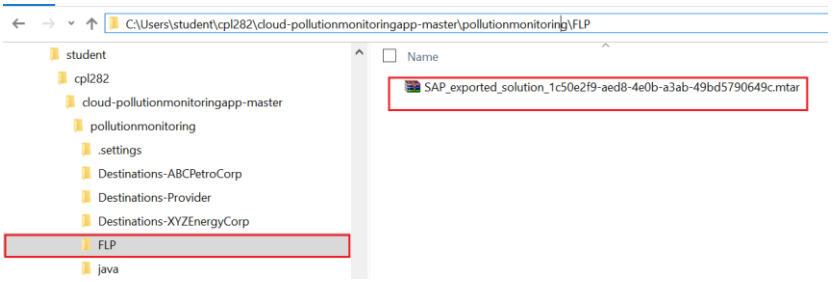
Note: Currently SAP FLP does not support subscriptions. It is not possible to subscribe to an FLP site running in the provider account but for the sake of completeness we are importing SAP FLP site **containing configuration of HTML5 application** into the consumer Subaccount.

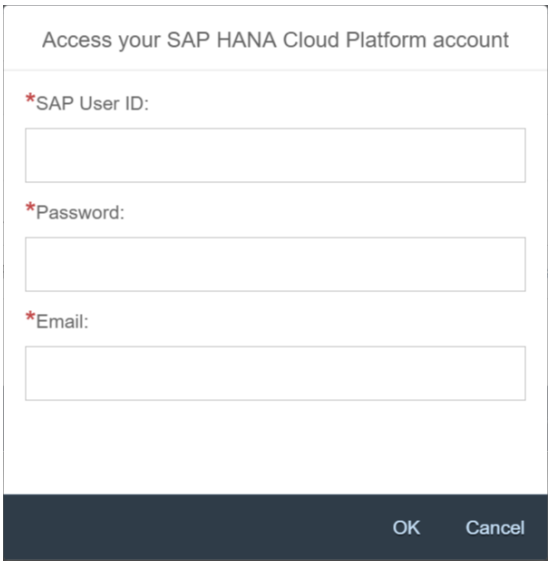
Step1: Login to SAP CP Portal Service and Import SAP Fiori Launchpad content into ABC PetroCorp Subaccount

Explanation	Screenshot
1. In the **ABC PetroCorp** (P1942783xxx) Subaccount Click on **Services** to view the complete list of services offered by the SAP Cloud Platform for your account. Search for portal under **All Categories** to locate the **Portal Preview**. Ensure that the service is **Enabled** for your account.	

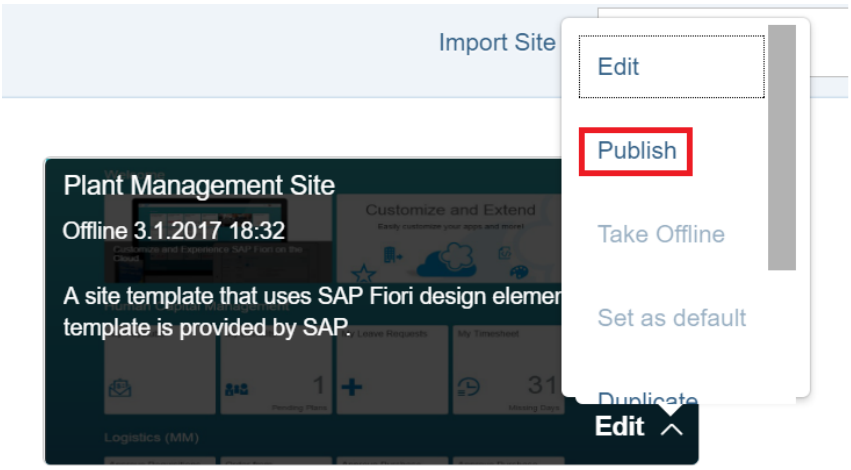
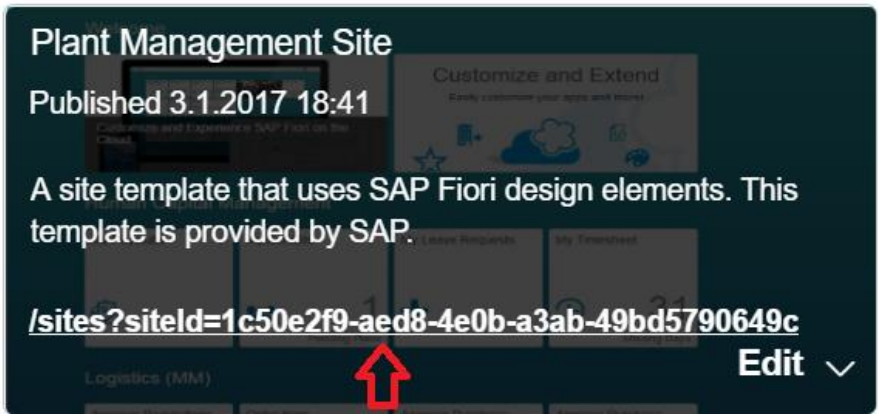
Explanation	Screenshot
2. This will open the **Service Description** page for Portal service. Click on the **Configure Portal Preview** to assign the predefined **TENANT_ADMIN** role for administrative permissions of the SAP FLP.	
3. Navigate to **Roles** pane and Select **TENAT_ADMIN** Role And assign to **Emily User_ID in the IDP that is P000006**.	
4. Click on **Portal** to switch back to the Service Description Page. Click on **Go to Service** to launch **SAP Cloud Platform, Portal Preview** in a new browser tab.	

Explanation	Screenshot
5. Login to ABC_PetroCorp(PXXXXXX) IDP with Emily's credentials **Emily/Test@12345**	
6. This will open the **SAP Cloud Platform Portal welcome** screen.	
7. Navigate to **Site Directory** from the left-hand side panel of SAP Cloud Platform Portal. Click on **Import Site** to open the import site dialog.	

Explanation	Screenshot
8. Click on **Browse** to select mtar file.	
9. Choose the **SAP_exported_solution_.<... >. mtar** file from the PollutionMontoring project under the folder **FLP** folder as shown in the screenshot	
10. Click on the **Import**	

Explanation	Screenshot
11. Provide **SAP User ID**: **P1942783XXX** **Password: Welcome17** **Email:** cpl282-XXX@teched.cloud.sap to access SAP HANA Cloud Platform account and click on **OK.**	

Step2: Publish and Launch SAP Fiori Launchpad site from ABC PetroCorp subaccount

Explanation	Screenshot
1. Click on the **Edit** menu of the newly created tile for Plant Management site. 2. Click on **Publish**.	
3. In the **Publish Site** confirmation dialog Click on **Publish**. This will publish the site and make it available to use.	
4. Click on the **SAP FLP Plant Management Site URL** This will launch the SAP FLP site in a new browser tab.	

Done!
With this, employees of **ABC PetroCorp** will be able to access the Pollution monitoring dashboard application using SAP Fiori Launchpad.

www.sap.com/contactsap