# 1 BTP Deployment Guide

# BTP accesses

Each customer following these instructions will be allowed access to their instances which are required to deploy all our applications.

All the following steps must be taken for a correct and functional deployment.

If the customer does not have sufficient knowledge of working in the BTP Cockpit, it is possible to request a consultation with a developer with whom this configuration will be carried out.

# Global Account

Admin rights are granted if the customer is interested in the management of his Global and Subaccount by the developers.

ADMIN

only if the customer needs and wants to create and edit a sub account and the entire space.

This is how your left bar will look when you have admin rights to global account.

VIEWER

It is enough if the customer can prepare the environment himself, so this permission is enough for us to get to the subaccount.

This is how your left bar will look like when you have viewer rights to global account.

# Subaccount(multi-Environment)

Here we will be interested in two main factors to set the right approaches.

Permissions are granted for the given Subaccount on which the application will be deployed (TEST/PROD)

Security

Cloud Foundry

SECURITY

When the Users tab opens, there is a Create button in the upper right corner, click on it and fill in the form where you enter the email of the developer you want to add.

in most cases the default identity provider is assigned, unless otherwise specified.

the Role Collections must also be assigned.

After creating the user, click on the row that was created with the new user.

Expand the overview of the user on the right side of the screen, click on the three dots in the Role Collections section to assign Role Collections

When you click on the Role Collections assignment you will see a form like this where you will be given the choice to check every role that will be available

Cloud Foundry

In Cloud Foundry, click on the Org Members button.

Once the Org Members tab opens, the procedure is very similar to Security, there is an Add Members button in the top right corner, click on it and fill in the developer's email and you have the option to fill in the rights for the developer.

In the E-mails field fill in the email of the developer you want to add

In Assign Org Roles we assign in most cases all the roles offered by

Once the developers and all the Roles and Role Collections are assigned, we can click on the space where the applications will be deployed.

Click on the Add Members button.

You will see a form that you fill out very similarly to all the previous ones.

Enter the developer's email.

assign roles according to the screenshot, in most cases all.

These permissions are set for the specific space to which the application will be deployed.

# Success Factors

It is necessary to enable accesses to SF for TEST/PROD, where in each instance we create API key, Certification, Private Key, copy Company key via Manage OAuth Data

Via Manage Homepage we add a tile for the application to the main page in SF.

Access to SF either as Admin or for instances only - Manage OAuth Data and Manage Homepage

The creation of objects for the application and import is handled by the Consultants.

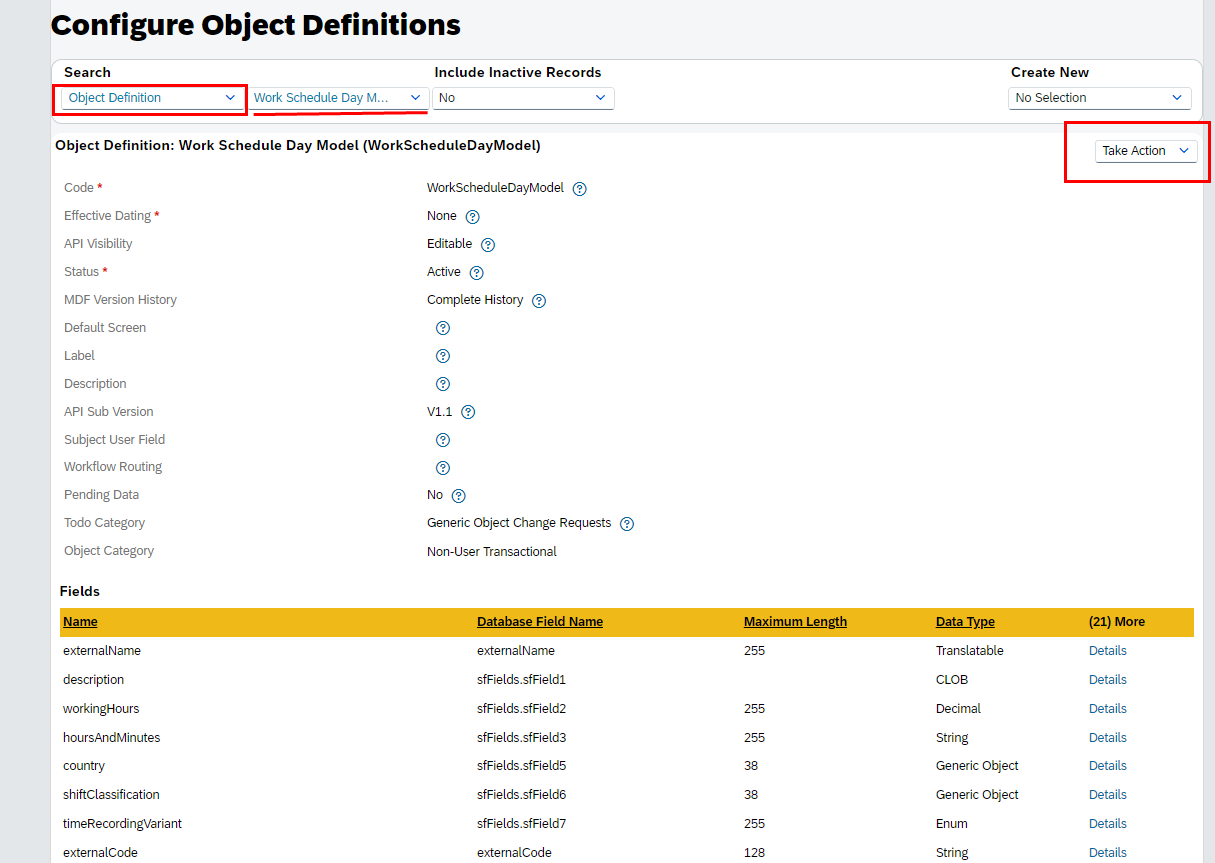
More information with print screen are here Guides.

# Setup SF preparation of MDF object

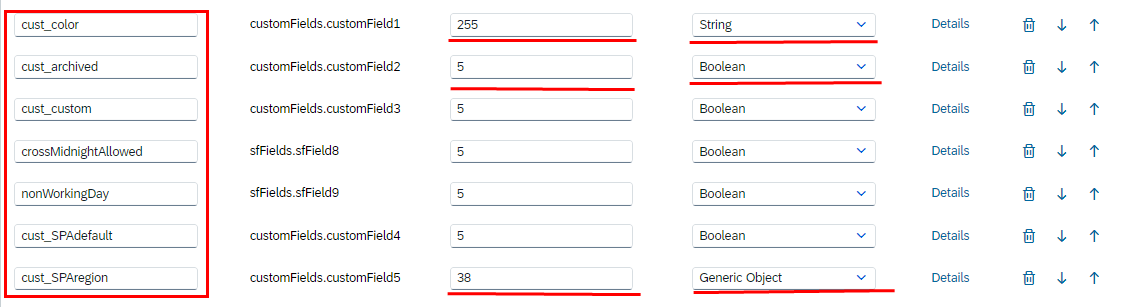
If we are deploying for the first time, we need to set up MDF objects (with or without TMA, spa proxy, work schedule and customizations and prepare some objects. Everything is configured in SF on the customer side.

# Configure Object Definitions

In SF go to configure Object Definitions in the first dropdown select Object definitions and in the second dropdown select Work Schedule Day Model.



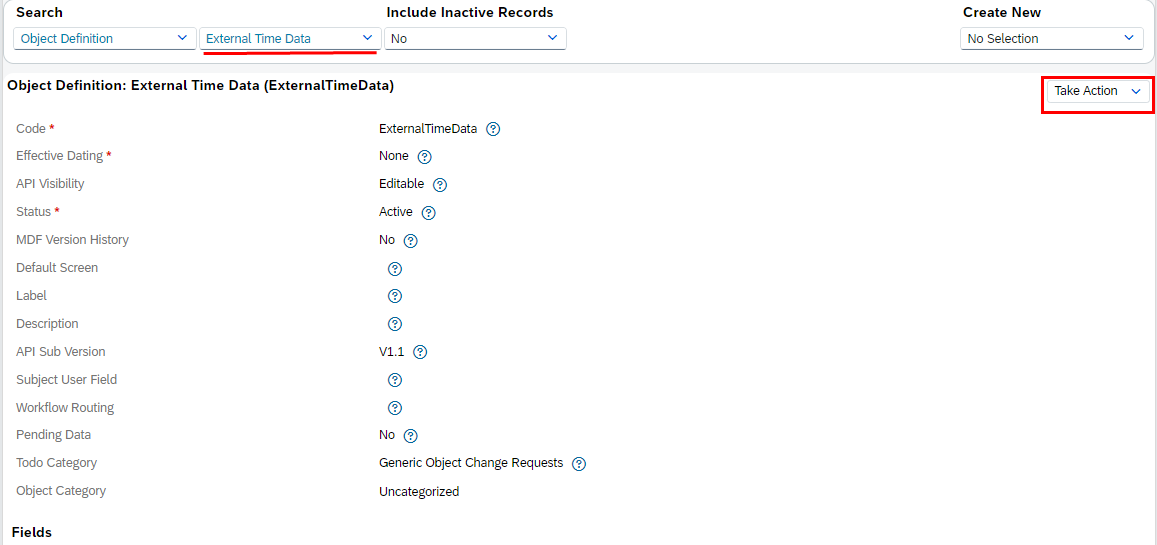
When we are on the ready tab, we click on “Take action” and then on Make correction to edit or create new objects.



If we don't find them, we have to prepare these 7 objects exactly as specified these are: cust\_color, cust\_archived, cust\_costum, crossMidnightAllowed, nonWorkingDay,cust\_Spadefault, cust\_Sparegion.

!!exactly according to the attached screenshot!!!

Another object that needs to be modified is External Time Data. Where again we click on Make Correction in the “Take action” Button and edit the objects



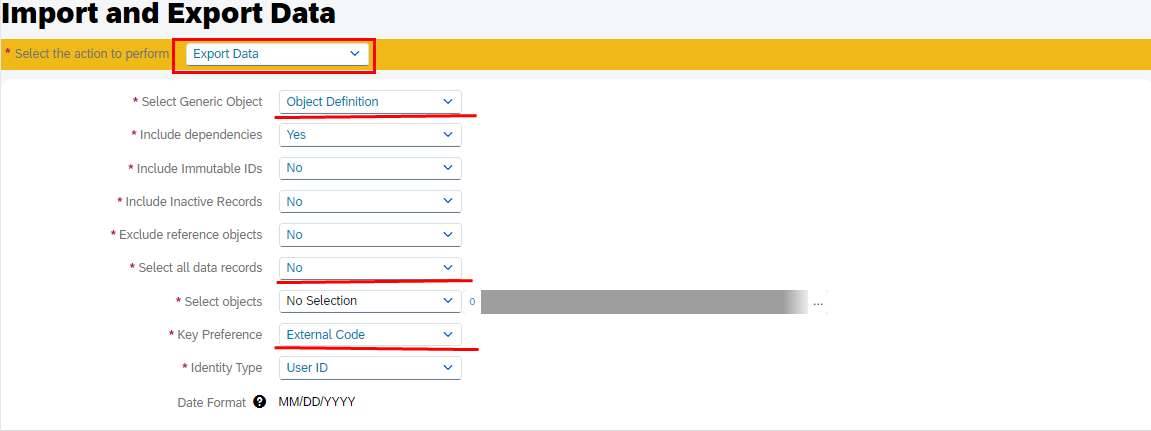
Then find the object “endTime” in the list and click on detail.

The object datasheet will open where we find the Visibility field and that field must be set to Editable then the Status field which must be Active and when we edit it, we click done.

# Import and Export Data

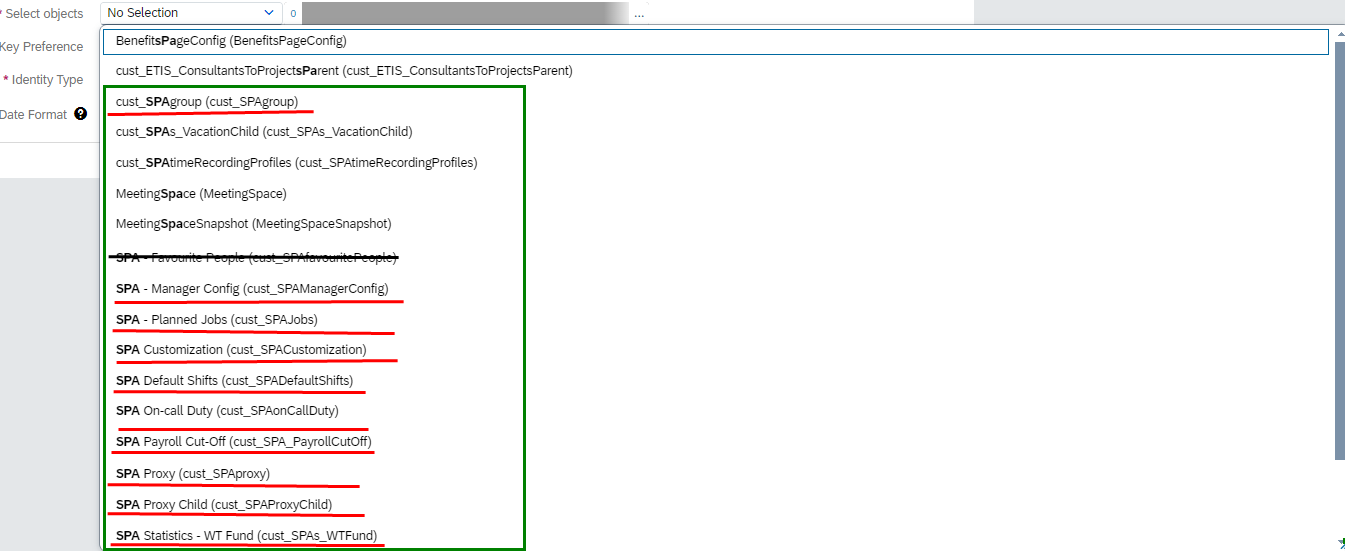
Here you need to export MDF data from the system where these objects are already prepared. For export we can use 2SDemo Sf.

!!!If the customer where we deploy SPA, have TMA (which are not created on 2SDEMO => Telecome have a completed TMA mdf object) we can ask someone from the consultancy who has access to the customer SF that has TMA to export and send us the data that we will then import!!!



Then we go to the Select Object field where we type SPA and we will see all MDF objects for SPA.

After clicking them we add them to our selection, we must select all the objects we need SPA-Favourite People is not used anymore and it is replaced by cust\_Spagroup.



After selecting all the objects that are needed, click on export and the selected data will be exported.

Then go to the Scheduled Job manager where you find your exported object go to the action column where you click on View Details and then download the file where the exported objects.

If we have downloaded all the objects, go to Import and Export data again (But this time to customer SF) in the Select field, select Import, click on ZIP file, select the file we exported and click on Import.

If we want to see if the data is uploaded well, we can click on Validate where we can look at the folder to see if it is empty or we can go to Export data and as we selected SPA objects in the previous step to make sure that they can be exported and that select will offer them to us.

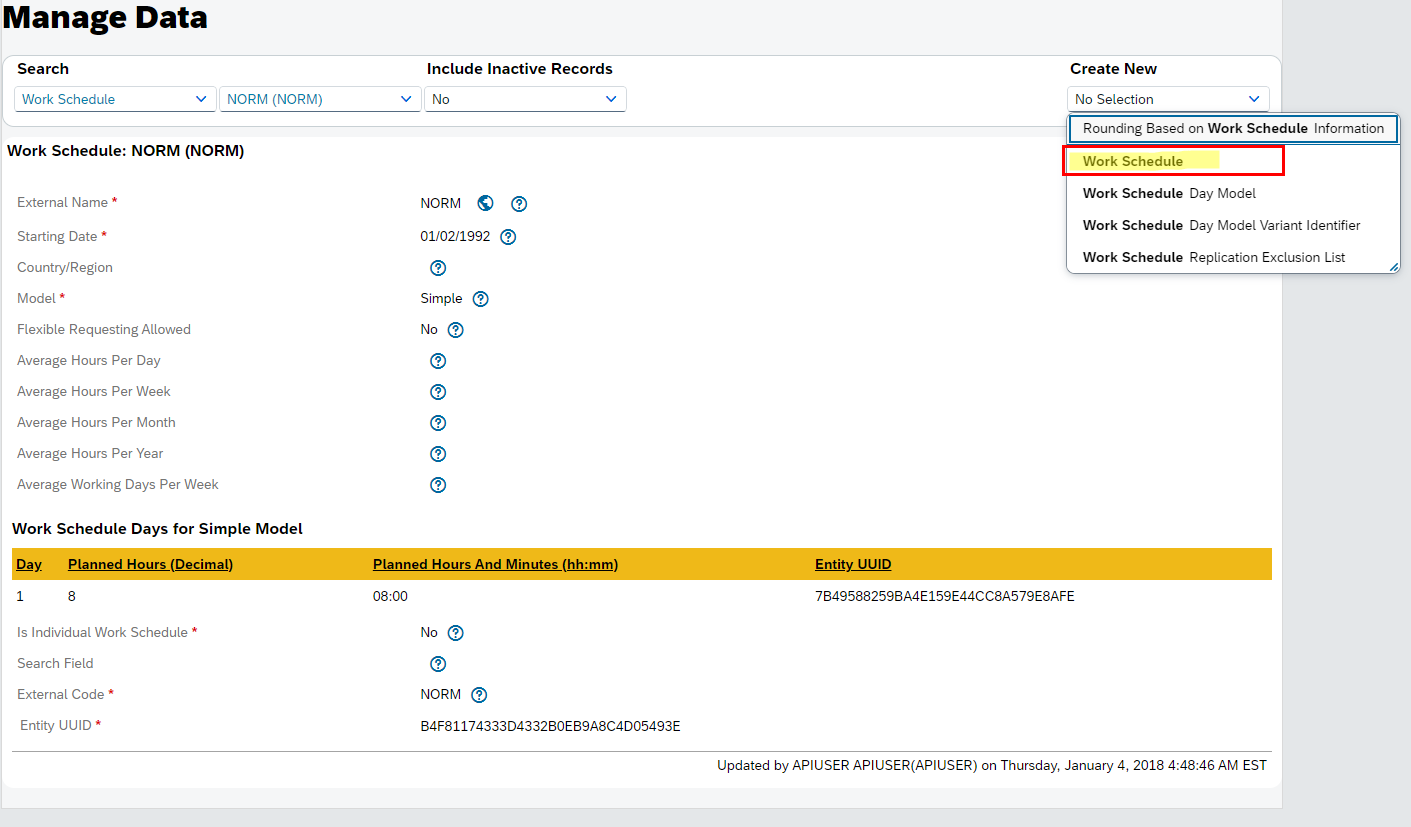
# Manage Data

First, we go to Manage Data if we have imported the MDF objects correctly we should find SPA Customization in the Create New field.

Each customer has one specific customizing which has an external code SpaCostumizing and an external name SpaCostumizing and then everything necessary is filled in according to the customer's specifications in the workbook.

The next step is to create a Work Schedule: NORM

this norm is used only once, and we can create it according to the already created norm in 2S. Go to the Create new field and find Work Schedule and create it. External Name => NORM and then add everything according to the prepared template.



!! It is needed to create Work Schedule in SPA and without this standard the application would not work.!!!

# Manage Configuration UI

Go to Configure UI click on Create New in Id type “SPAproxy” and in the selector if we have correctly imported MDF objects select SPA Proxy.

You only need to create this object to be stored in SF and save the other things such as naming the object and the name is then taken care of by the consultant, who modifies it according to the customer's specifications.

# Configure People Profile

If the customer has it, this is where the MDFs already created in Manage UI are pulled into certain sections of the user's profile. If he has permissions to do so he will see everything that is set. Consultant configuration

# Preparing the App for Deployment

The shift planning application has only two independent parts.

Shiftplanning

Shiftplanning-display

# Shiftplanning

In this part we must prepare the application for deployment. We need to go to the app folder using the cd app command and then run npm run build (we don't need to run npm install first because there is already a command in package.json that when we enter npm run build it will run the node module installation first and then the build itself).

The dist folder we need is automatically created in the app folder so we don't have to copy or move anything, after npm run build we have the application ready for the next steps.

# Shiftplanning-Display

Here we can continue in the same way as in the previous case.

In this part we must prepare the application for deployment. We need to go to the app folder using the cd app command and then run npm run build (we don't need to run npm install first because there is already a command in package.json that when we enter npm run build it will run the node module installation first and then the build itself).

The dist folder we need is automatically created in the app folder, so we don't have to copy or move anything, after npm run build, we have the application ready for the next steps.

If there is any problem in installing Node-modules. You can delete npm install from package.json and install it as needed.

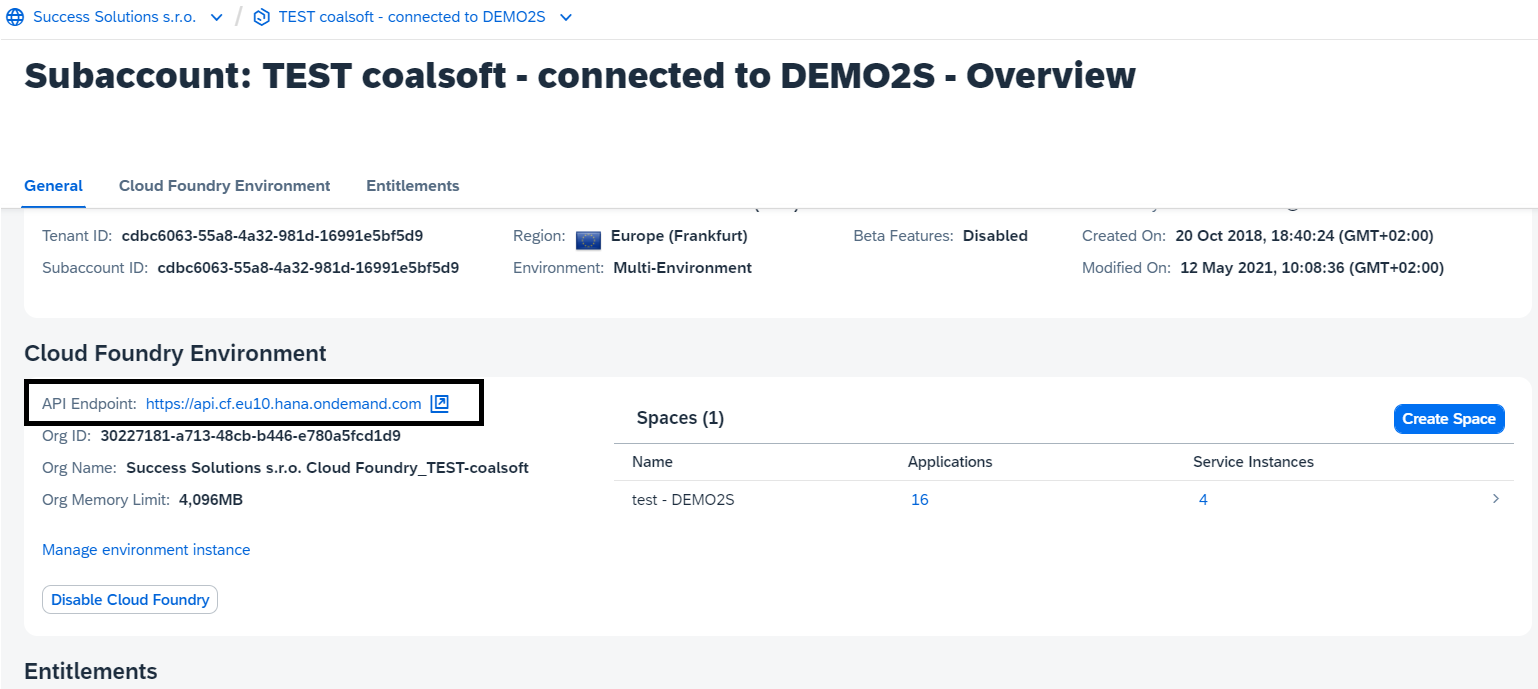
For example, for mac with sudo npm install.

For windows, the best option is to delete package-lock.json and run npm ci which should install all necessary node modules and their versions exactly as they are. The versions should be the same never newer otherwise the application just won't run.

!!!The problem in versions and module installation is sometimes a problem but it can be worked around!!!

# Logging in and setting CF.

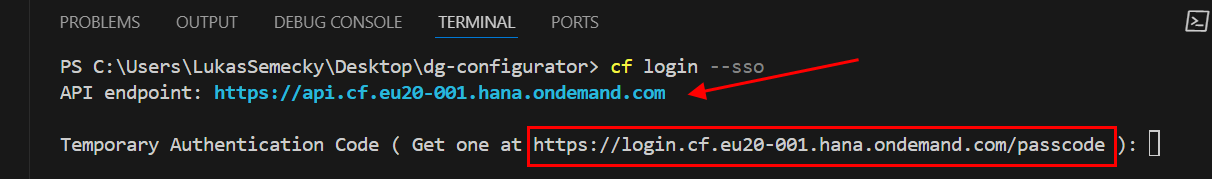
To access CF we first need to find out the API to which the respective subaccount is connected, we can find this out from the BTP. The address is needed to connect to the space.



The Api address is the same in most cases, but it is good to make sure you cannot connect to the subaccount without the correct Api address.

To log in to CF you need an E-mail and password to BTP.

After connecting to the cloud foundry, you use the cf login command and then enter your email and password or cf login --sso where you click on the subcommand, and it generates a login code that you enter the terminal.



If you need to log into another Endpoint Api, just type cf api and copy the appropriate Endpoint.

You will only see the accounts that are assigned to a given endpoint, you will never see a list of all of them unless you are on their endpoint Api.

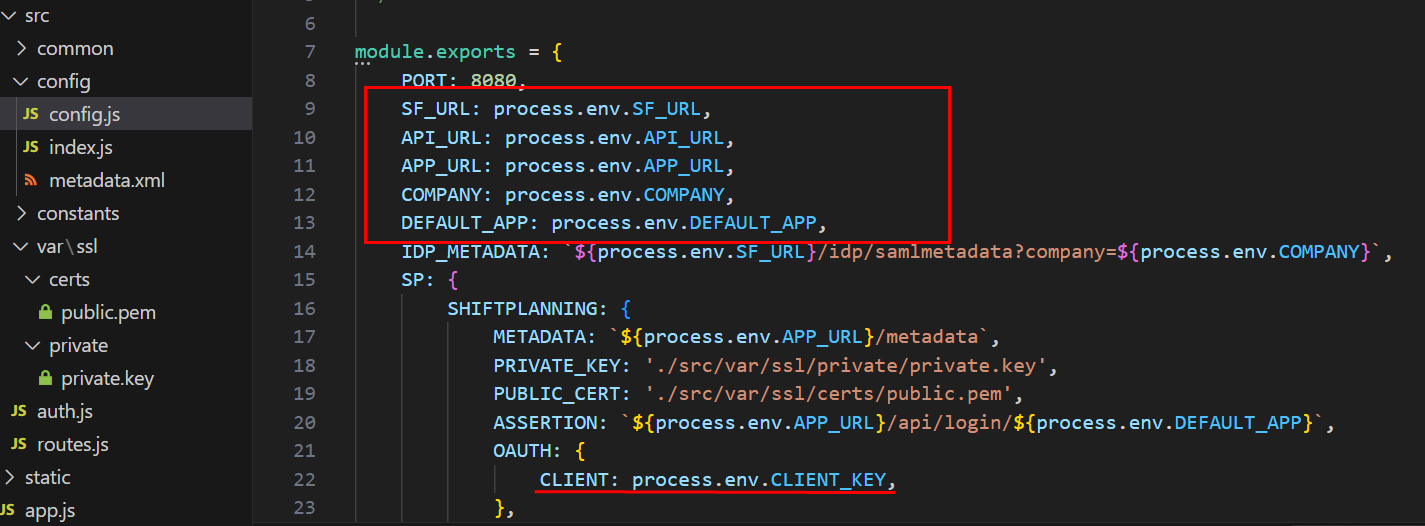
# Preparing for deployment(deploy.yaml)

The yaml file controls the deployment of the application, adding details such as how many instances to create, how much memory the application will need and other necessary variables to make the application work.

For multi-customer deployments it is a good idea to create your own yaml file to add variables to, this will allow you to not overwrite the root file each time a new deployment is made.

When these variables are added, the information is automatically overwritten in the main file.

!! The names in the created file must match those in brackets ((API\_URL)) it is case sensitive!!



In the application in the config.js file you can make sure that you have all the necessary env. that need to be inserted or created if you miss one in the yaml file.

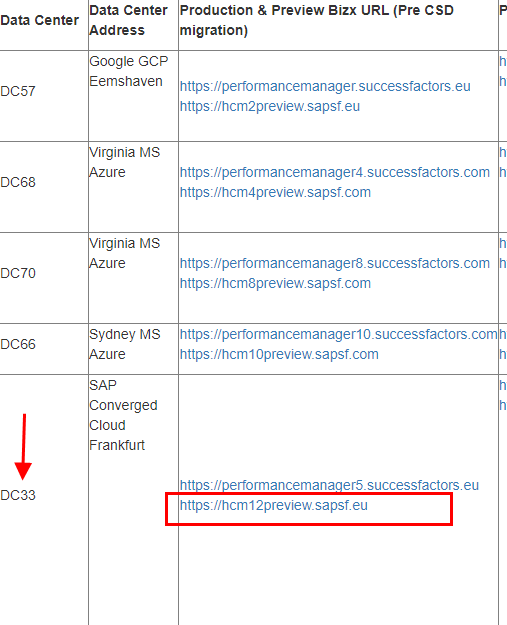
# SF\_URL

What is the SF\_URL I need can be found from the SF address of the customer, after logging into SF we investigate the URL address and from there we find out what sf URL the customer uses.

It is always https:// <<datacentrum>>.eu Everything beyond this address is no longer relevant for us.

# API\_URL

This address comes directly from the SF\_URl on this page to find the corresponding SF\_URL in the third column.



When we find the appropriate address, we focus on the first column DATA CENTER where is the code designation of the address. This is necessary to find the API\_URL on this page, we find the row that corresponds to our code name, and this is our API\_URL.

PREVIEW is for the test instance.

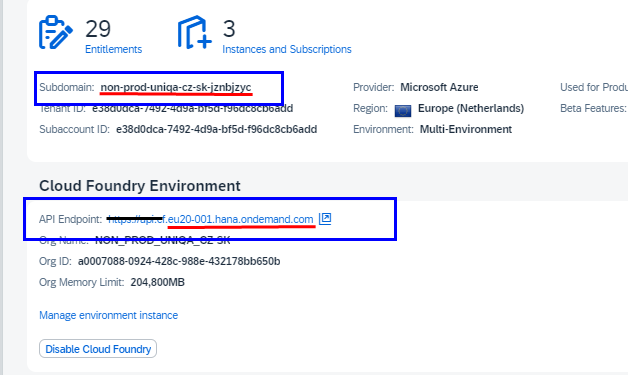
Production is for production.

# ROUTE

This route is created according to the rule:

<<application name>> - <<subdomain>>.cfapps. <<api endpoint>>

the name of the application is found in package.json or directly in manifest.yaml followed by a dash subdomain which is on the BTP of the respective subaccount followed by .cfapps and the noon part of the Api endpoint as seen in the pictures



this way we create the application address which is created both in routes and in APP\_URL

# COMPANY\_ID

Company ID in some cases can be found directly in BTP from the subaccount, but in most cases, you can find it either from the client itself or directly in SF when you have access.

After logging into the SF client, enter Manage Oauth2 Client Application in the search box.

Here you will see a table of applications logged in, just click on any of the view buttons.

And after clicking you will see a table where the first column shows the CompanyID we need

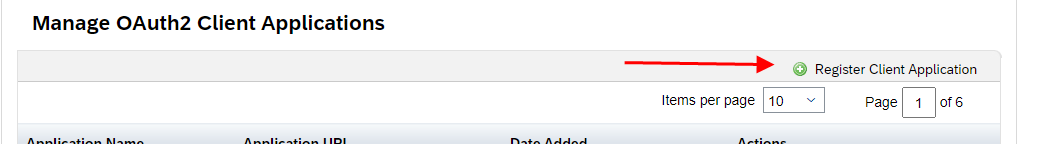
# CLIENT\_KEY

The client key is retrieved from the same place as the Company\_ID again by entering it in the search box in SF Manage OAuth2 Client Application which redirects you to the next page.

Here are 2 points:

1. we can find an already registered application by name (usually when creating it you enter the same name as the application itself, for better searching) and when we click on view, we will see in the middle of the page Api Key = CLIENT KEY

2. If the application is not yet registered here, we must first create it, click on REGISTER CLIENT APPLICATION



Here we enter the name of the application, we can look in the manifest.yaml and copy the name.

Then enter the URL it is our created route that we did in the previous step supplemented with https:// and Certificate, usually we use the same for all applications.

We have the certificate stored because the certificate includes the private key which is paired and without these two important things the application will not work.

When we fill everything in, we click on Register.

When we open the created and registered application via the view button, we will see a new window with our CLIENT\_KEY = API\_KEY in the middle.

For each part of the application, we must create a new CLIENT\_KEY the only thing that remains is always Certificates, we use the same.

# CERTIFICATION AND PRIVATE KEY

These keys are necessary for the functioning of the reposted application, they must always be the same and both must be paired together, the certificate does not work without the private key and vice versa.

If we don't have either key the key is registered as an application in Manage OAuth2 Client application in SF where we first create the keys when we create them.

Both keys can be downloaded, after creating the Certificate you must download the private key as well because it is the only way to get it then you will never get it again and you would be forced to make a new certificate.

The keys are added directly to the code, in APP/SRC/VARSSL, and this opens certs/cert.pem for the certification and private/private.key for the private key.

We always need to be careful that both keys are copied correctly otherwise the application will give us an error. It is always a good idea to check the first and last few characters of both codes.

# Deploying the application

When we have everything set up and all the manifest variables ready and the keys copied in the code, we can deploy the application.

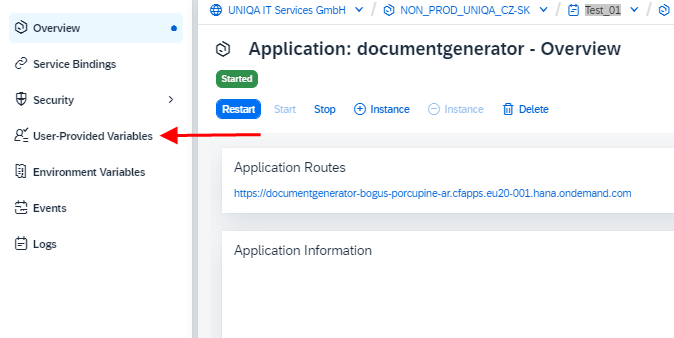
we connect to the CF of the respective customer for deployment.

After logging in to the customer and his space, enter the command.

cf push --vars-file <<filename>> cflinuxfs4

Then the application starts deploying to the given BTP

In this way we deploy both parts of the application, shiftplanning and shiftplanning-display.



In BTP we can still check all variables if they match what we have in the manifest to make sure that everything is correctly assigned, if something is missing or not correctly assigned, we can edit the variables directly in BTP without having to re-deploy the application, just edit, or add the desired value and restart the application. The only exceptions are Certification and Private Key if there is an error in them the application must be redeployed.

When we have everything set up and checked, and all deployments have taken place, the green start button is lit for all applications. We can test their functionality. When we start all 3 Frontend applications, and the screen shows this:

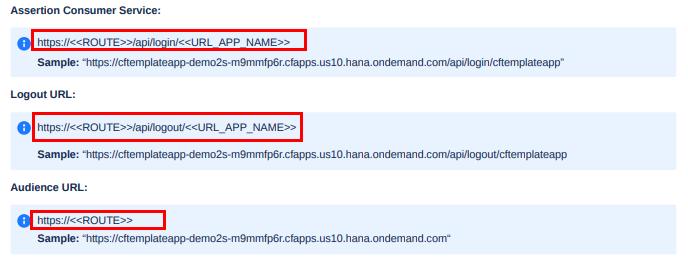
This means that the application is ready, and we are halfway done by this stage. The application is successfully deployed, but until the Provisioning addresses are created, we don't know if it is fully functional.

# Address Configuration(provisioning)

This step is necessary for the functioning of the application, the created addresses must be sent to the appropriate consultant who oversees the application to the customer.

Addresses are always created 3 and from each frontend. So, for DG we will send 9 https addresses which the consultant will enter SF.

# creating an address



1- address is for login.

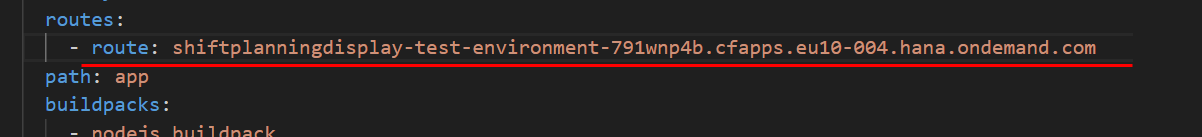
https:// <<ROUTE>>/api/login/<<APP\_NAME>>

2- address is for logout.

https:// <<ROUTE>>/api/logout/<<APP\_NAME>>

3 - only adress

https:// <<ROUTE>>



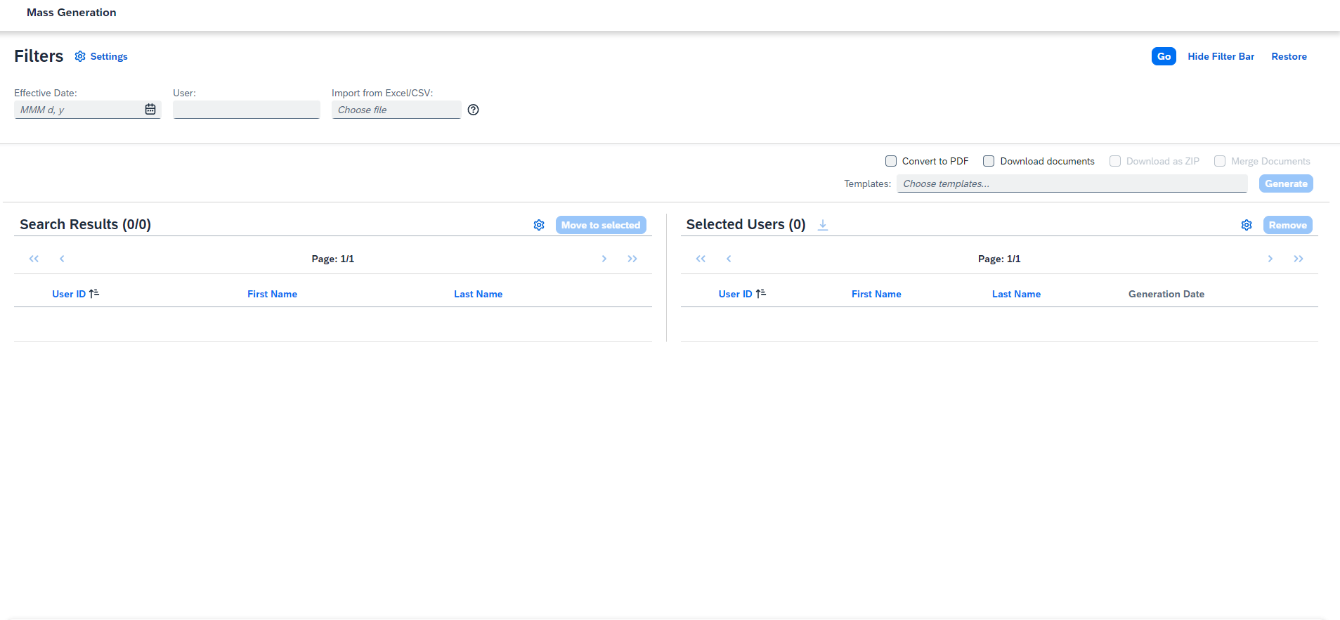
For this application it is necessary to create 6 addresses for each part of application 3. for both shiftplanning and shiftplanningdisplay

So, for example, the first address might look like this.

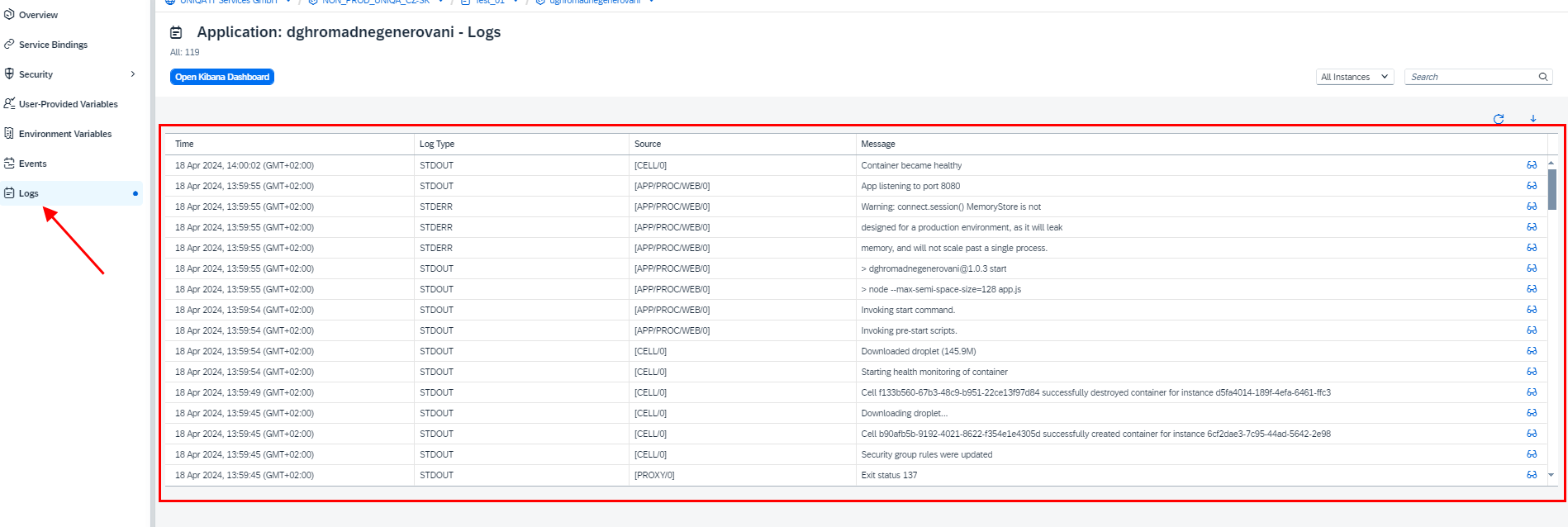
https://shiftplanningdisplay-test-enviroment-791wnp4b.cfapps.eu10-004.hana.ondemand.com/api/login/shiftplanningdisplay

# Configurations and tests

After deploying and saving the address to provisioning, the application should start fine like here:



If the application does not start and either the screen remains white or nothing is displayed, we must look for possible errors. We can look in the BTP logs which may tell us the possible error.



Check all the variables to see if they are entered correctly and there is no error, or if the corresponding Certificates or Private Keys match.

We can try to run the application on the guest locale unless there is a bug in the application itself or a bug in its build, we will test all functionality. If the application is running, there may be an error somewhere in the deployment or in one of the possible versions.

Or all MDF objects or Picklists that are needed for the application to work properly are not specified yet.