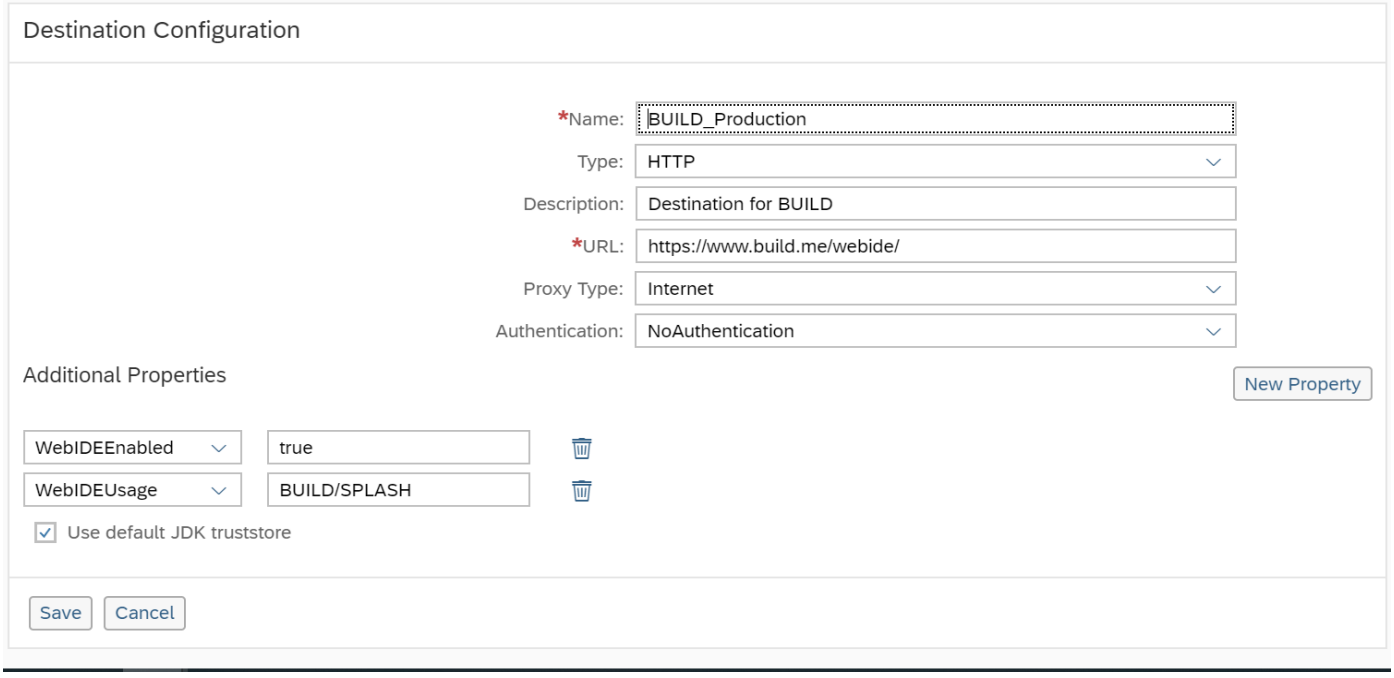
Full-stack Blockchain application development using SAP Build and SAP Web IDE

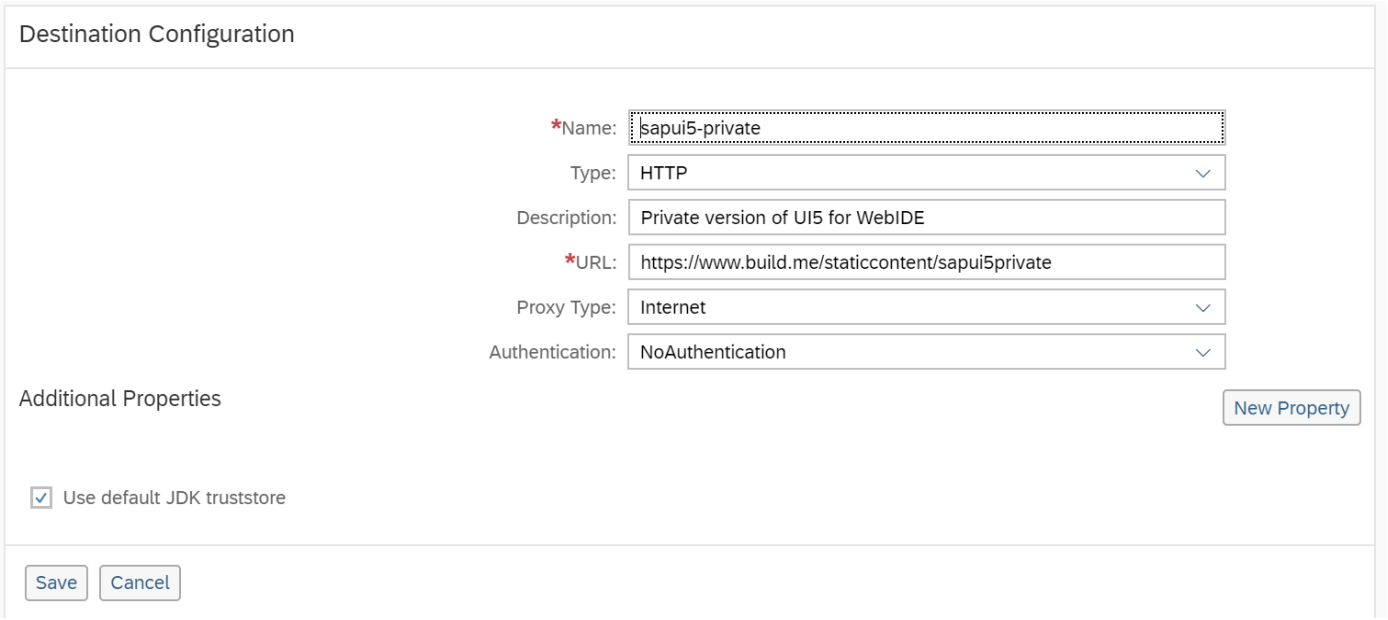
Learn to create a full-stack blockchain application from scratch using Hyperledger fabric, SAP Build and, SAP Web IDE on SAP Cloud Platform

Step 1: Pre-requisite

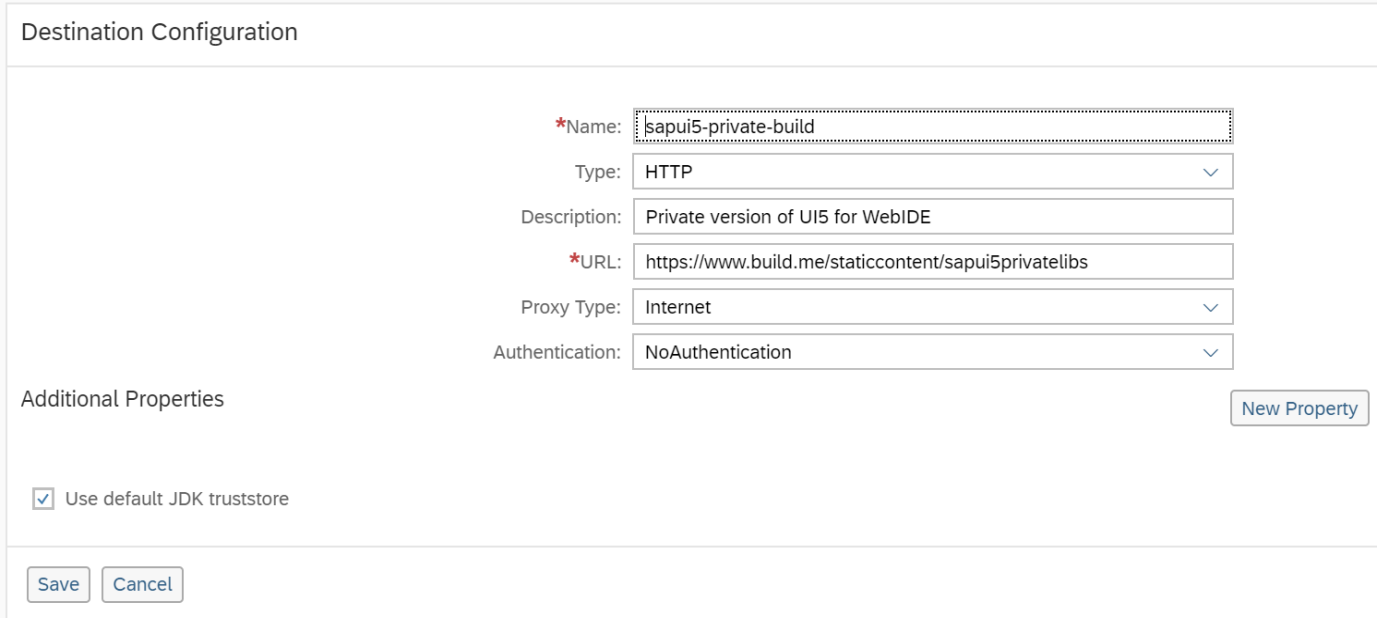
1. Logon to your SAP portal.
2. Go to Neo Workspace
3. Click on services (in the left-side menu)
4. Search for Build in search bar of the service
5. Enable SAP Build service under User Experience section
6. Search for IDE in the search bar of the service
7. Enable SAP Web IDE Full-stack under Developer Experience section
8. Click on connectivity (in the left side menu)
9. Click on destinations
10. If below listed destinations are not present in your account, add the destination by clicking New destination as shown in the screenshots below.



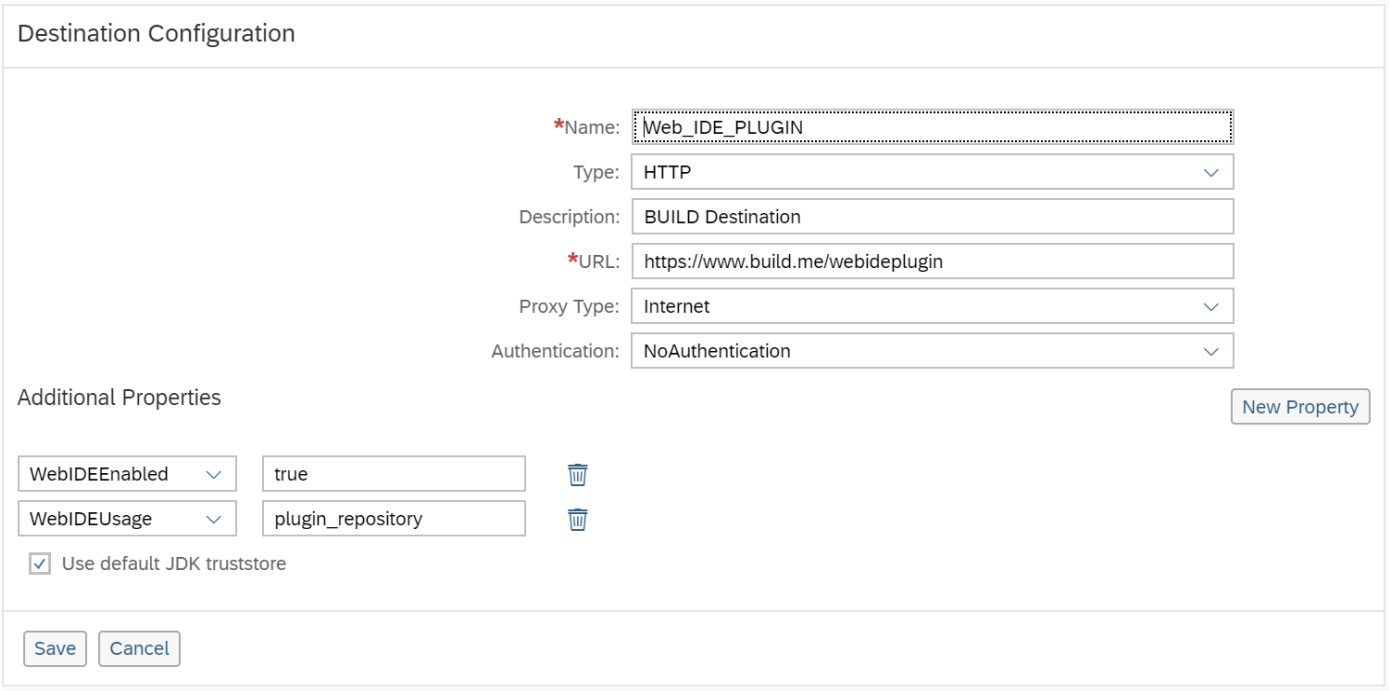
Pre-requisite Screenshot 1: Destination for BUILD



Pre-requisite Screenshot 2: Private version of UI5 for WebIDE



Pre-requisite Screenshot 3: Private version of UI5 library



Pre-requisite Screenshot 4: Web IDE plugin

More details about pre-requisite could be found here

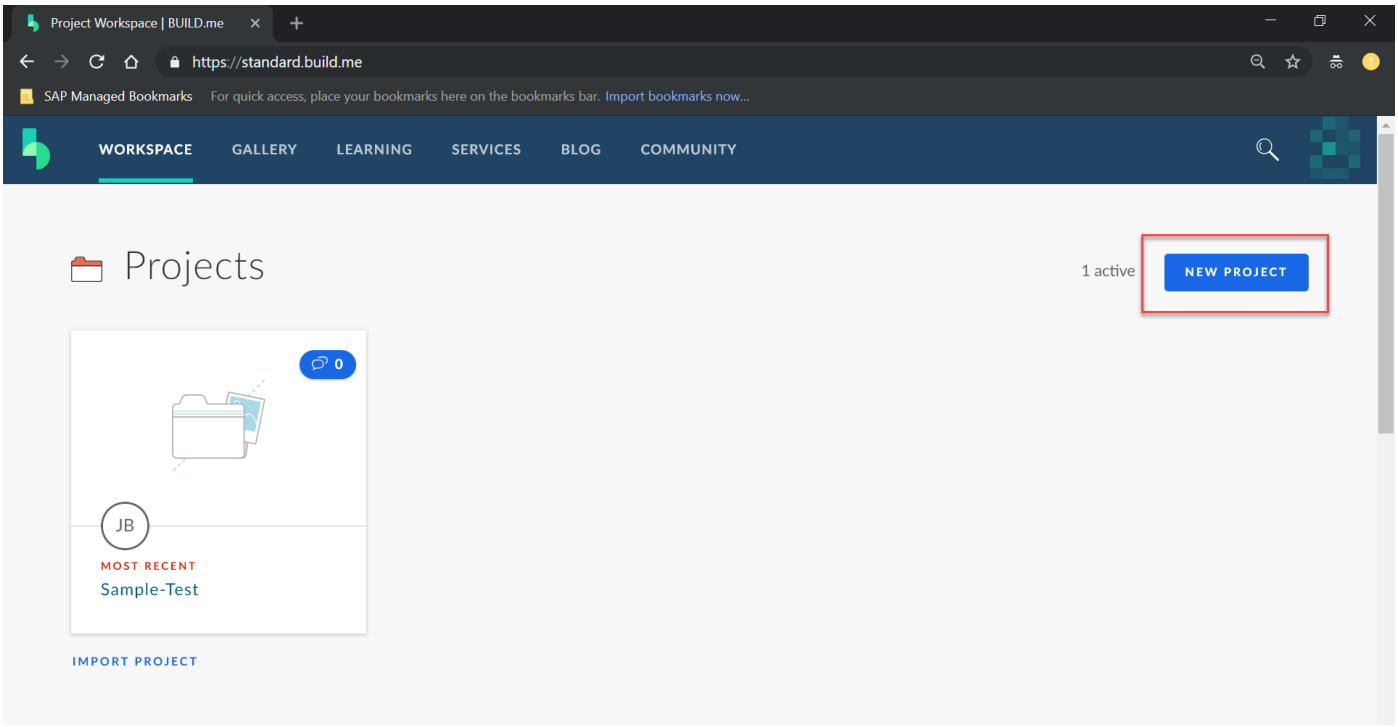
Use-Case:

There are 3 companies A, B and C. Each company could transfer the balance to other company. Once the balance for a specific company reaches -1000, they could not transfer the balance to other companies unless it settles the balances among all. The focus is not on settling the balance but how we could use SAP tools to achieve functionality of sending/receiving the balance among different parties involved. Tools used to develop this functionality is:

- SAP Build
- SAP Web IDE
- SAP Hyper-ledger fabric service

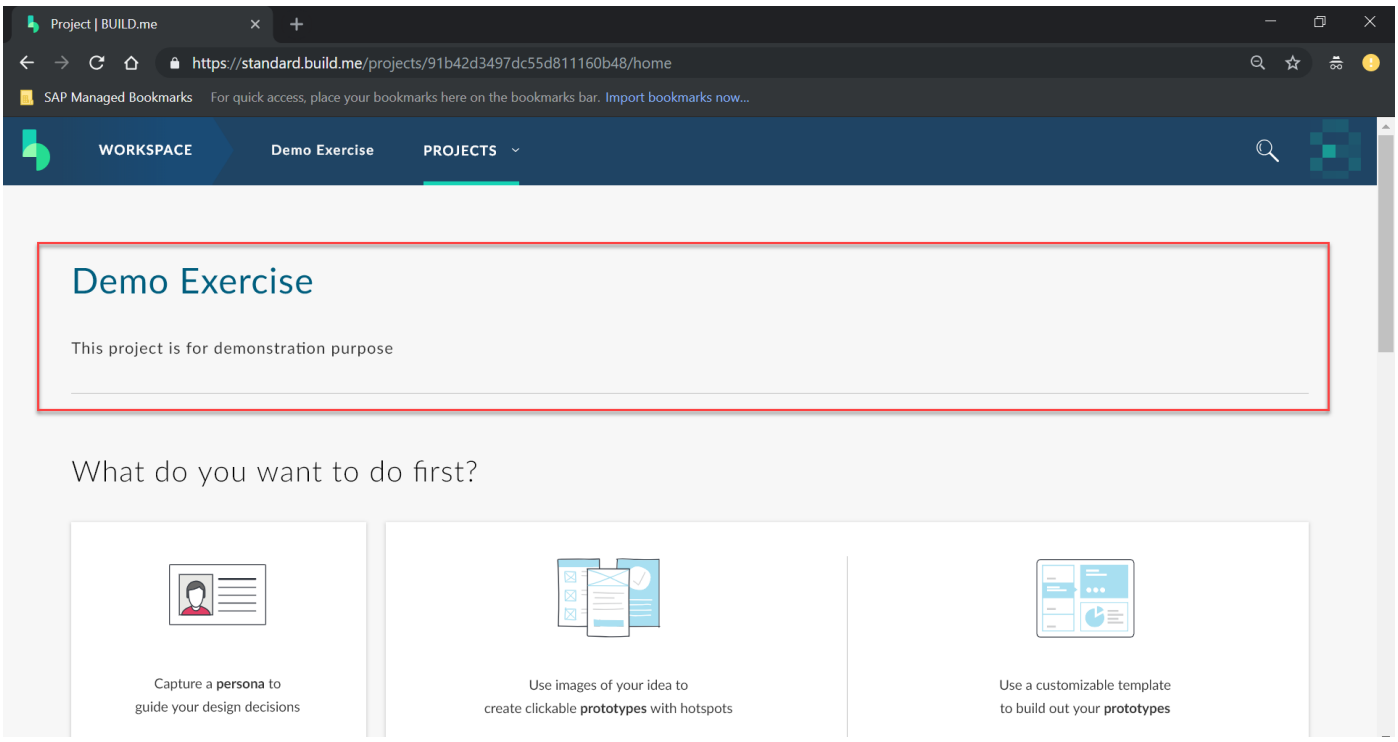
Step 2: Create the Prototype using SAP Build tool. (Future tutorial will cover interaction between different screen/pages using build tool)

1. Visit SAP build prototype website and create the account or login.
2. For detailed information about how to use SAP build tool, please visit here
3. Click on New project -> Create a new project on the right-side upper corner of the screen, as shown in the Screenshot 1.



Screenshot 1: Click on new project

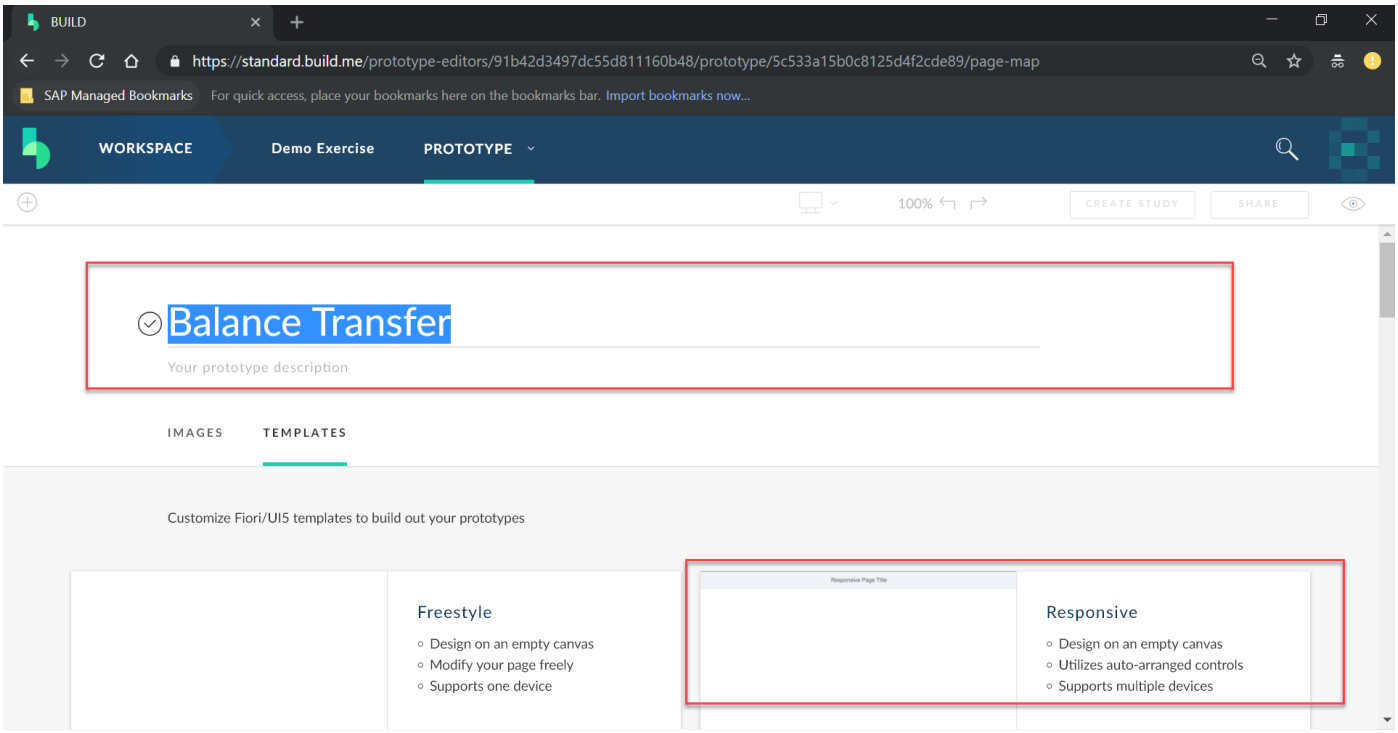
4. Write project name and project description as shown in Screenshot 2. You could give any name or description as you want. I have project name as “Demo Exercise” and project description as “This project is for demonstration purpose”. Click on “start with template”



Screenshot 2: Project Name and Project Description.

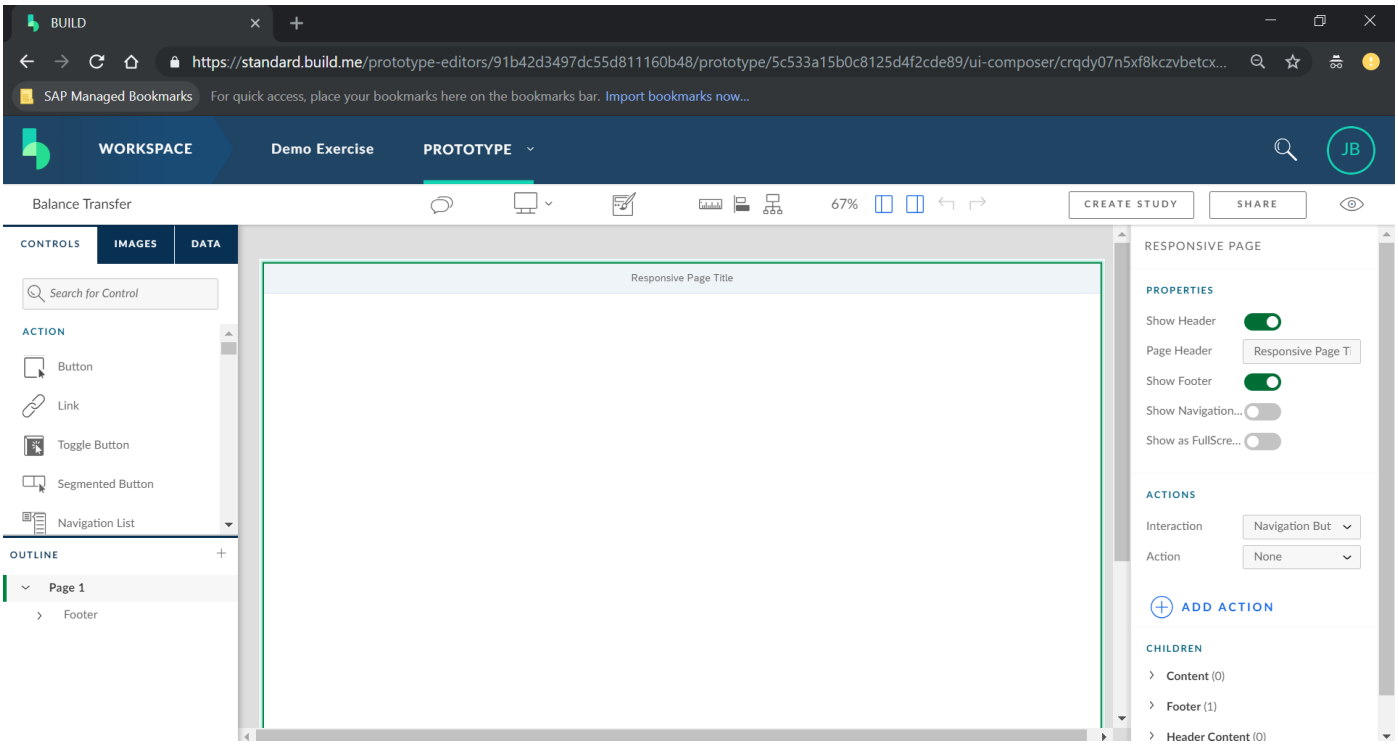
5. There will be couple of templates displayed in the page, such as, Freestyle, responsive, object page, worklist and many more.

6. Write prototype name and select the template. I have template name as “Balance Transfer” and have selected responsive template as shown in Screenshot 3.



Screenshot 3: Prototype name and select responsive design

7. Click on edit page, it will load up the screen where you could design your prototype as shown in the Screenshot 4.

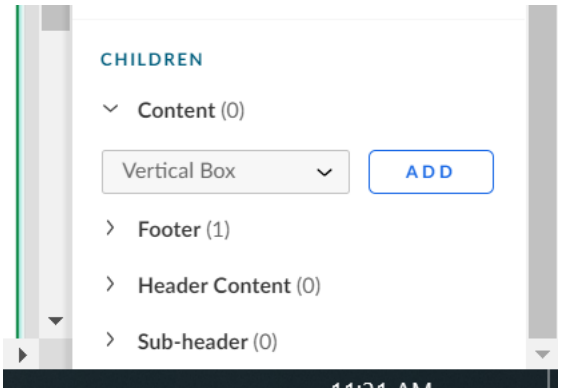


Screenshot 4: Prototype design screen

8. Change the page title. Click on the screen where it says “Responsive Page title” in the prototype and write the page title you want. I have given page title as “Netting Scenario”.

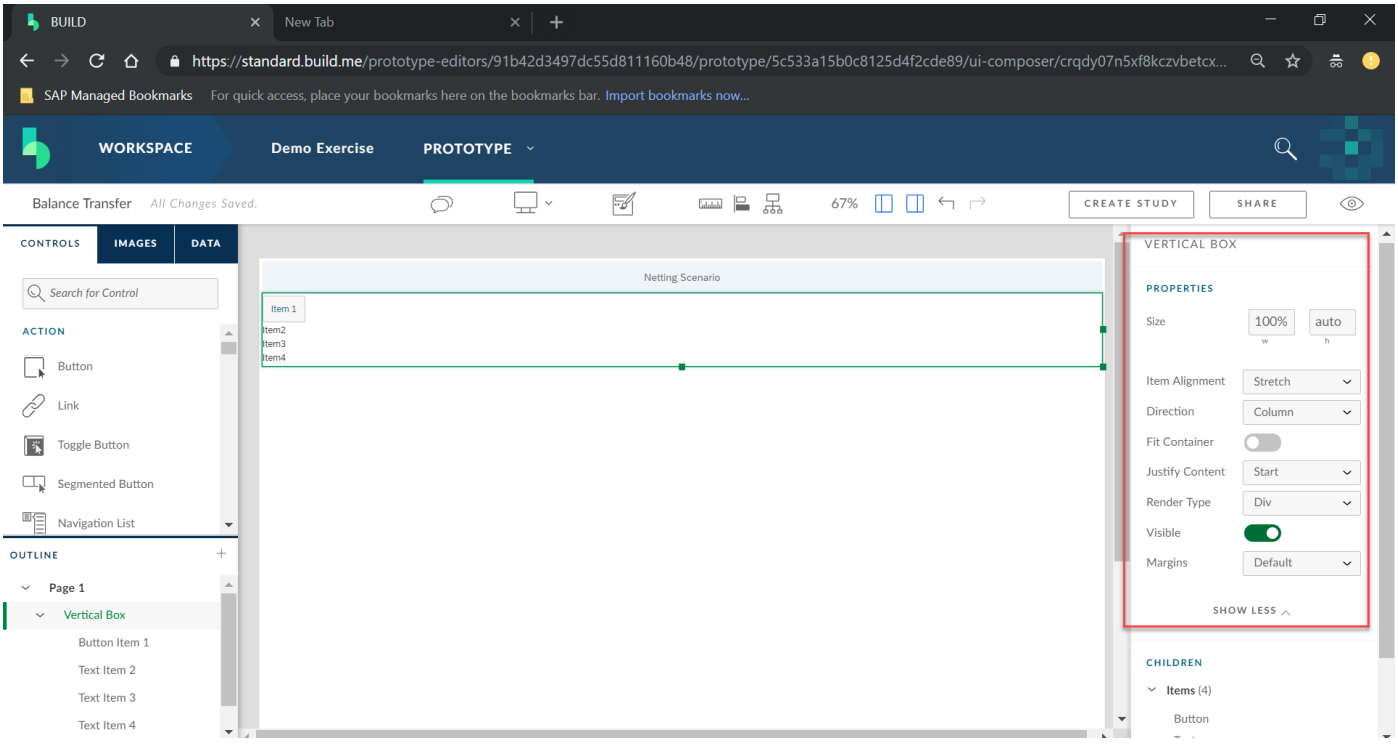
9. You could also change header, footer and content from the children listed on right bottom corner.

10. To create a prototype, I have added vertical box in the content first by selecting content in the children section in the right bottom corner as shown in the Screenshot 5. Click on Add.



Screenshot 5: Select Vertical box

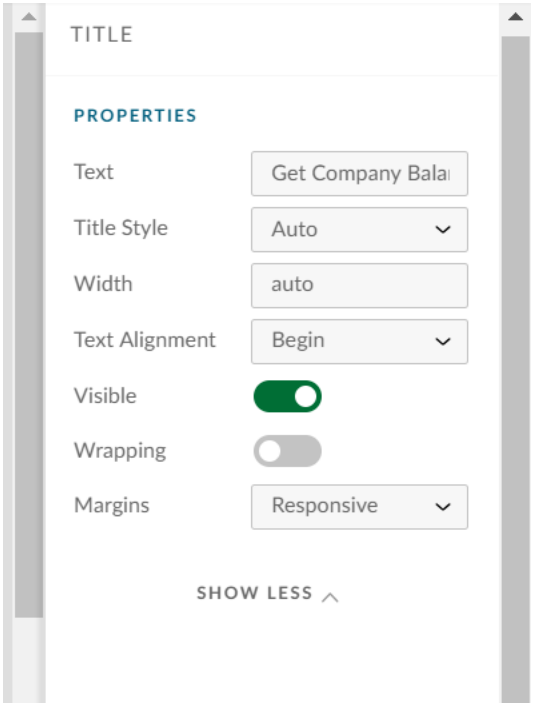
11. Change the properties of the vertical box as required. By selecting the vertical box in the prototype, you could view and edit the vertical box properties on the right side of the screen as shown in Screenshot 6.



Screenshot 6: Edit properties of vertical box

12. After editing the properties, add content to vertical box under the children section in the bottom right corner. Just as we did for adding a vertical box.

13. Add title as the content for the vertical box and edit properties for the same, as shown in screenshot 7. Property “text” of the title displays title text which I have set to “Get Company balance”



Screenshot 7: Add properties for the title

14. Now add the horizontal box to inside the vertical box. So, the order would be,

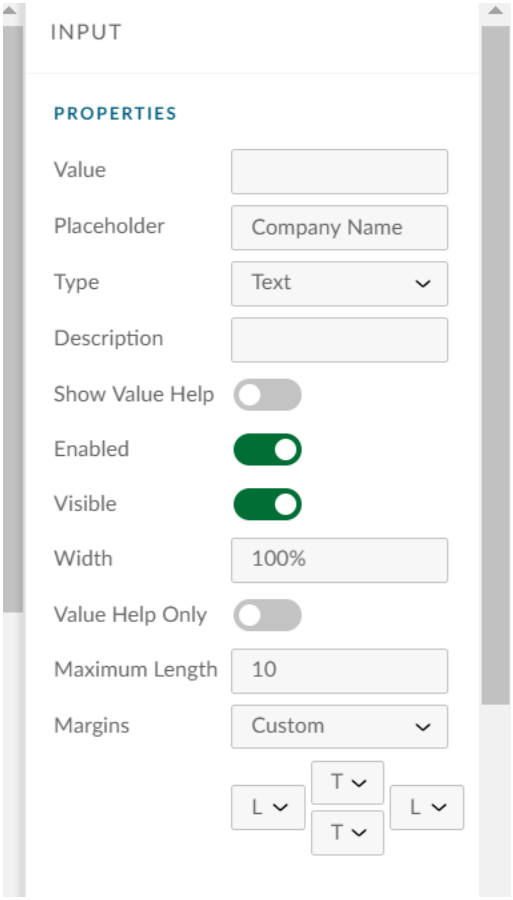
    Vertical box (parent)

        title (1st child)

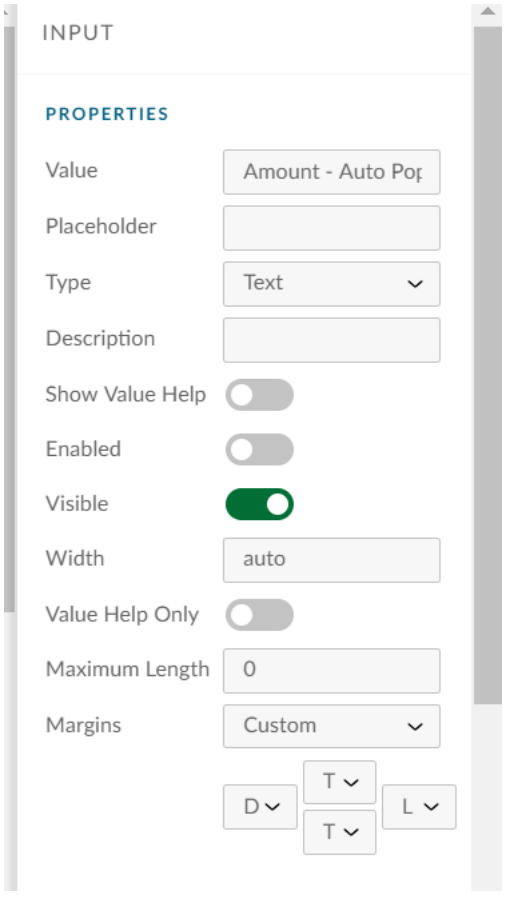
        horizontal box (2nd child)

    You could add horizontal box, by selecting vertical box in prototype screen (interactive UI) and adding content in the children section of the bottom right corner of the page.

15. Add 2 input box inside horizontal box, one to take user input for company name, and other to display the balance for that specific company. Screenshot 8, shows properties of first input box, Screenshot 9 shows properties of second input box.

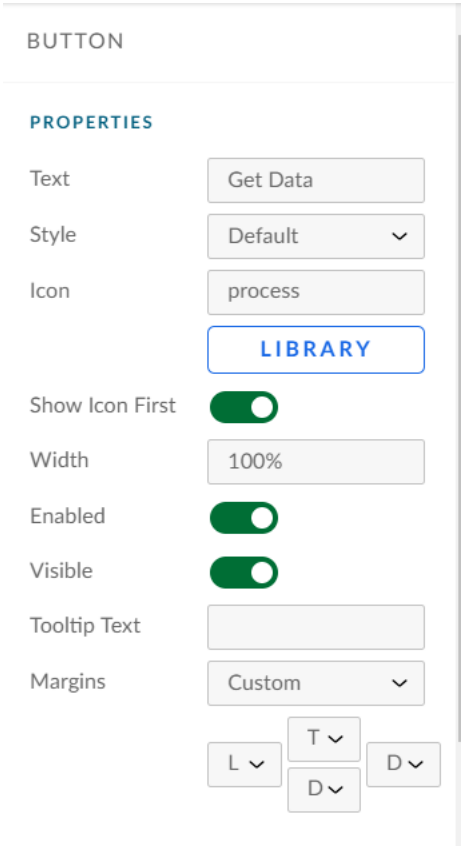


Screenshot 8: Properties of company name input box

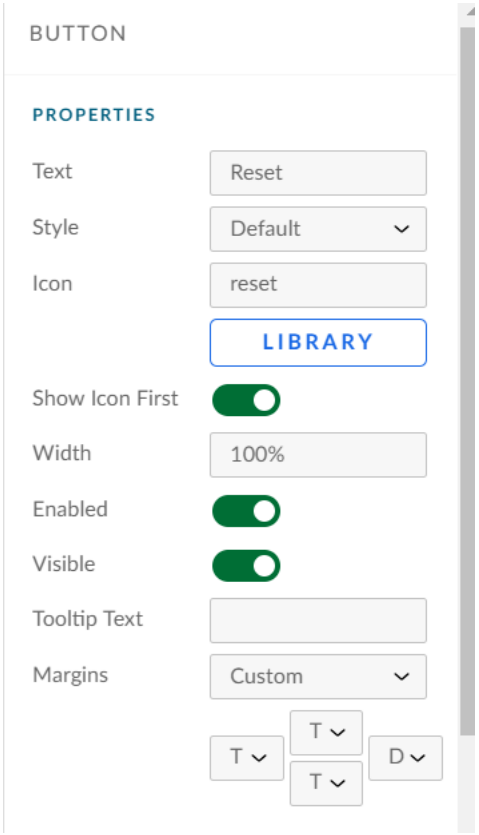


Screenshot 9: Properties of Amount to be displayed for a specific company (Auto-populated)

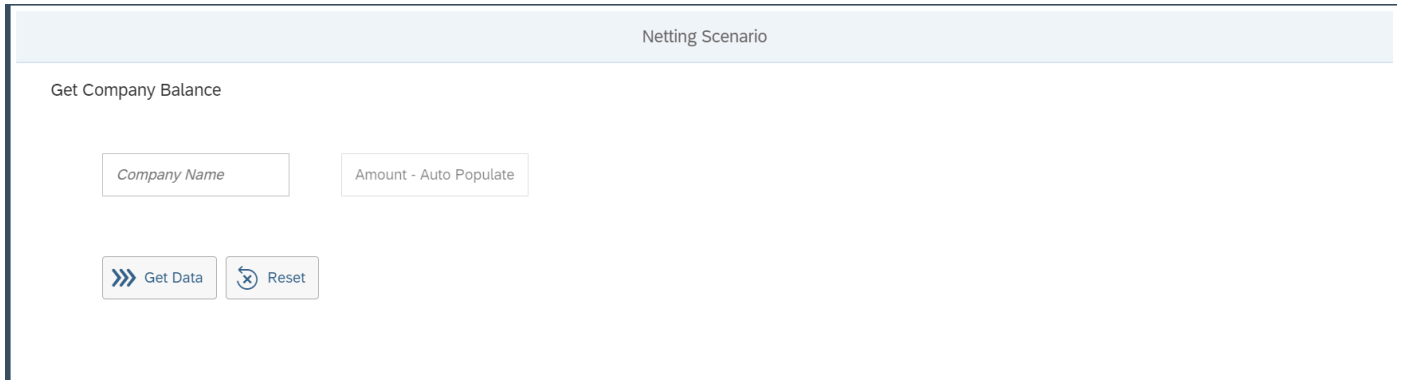
16. Adding buttons to get balance for a specific company or to reset the input box data. Select vertical box on the prototype, add a horizontal box and add 2 buttons inside horizontal box. Of, course you could set properties for horizontal box as required. In my example, I did not change any properties for horizontal box. Screenshot 10 and Screenshot 11 shows the properties of button to get balance data and to reset the input box data respectively. Screenshot 12, shows the prototype we have designed so far.



Screenshot 10: Properties for the button which gets balance for a specific company

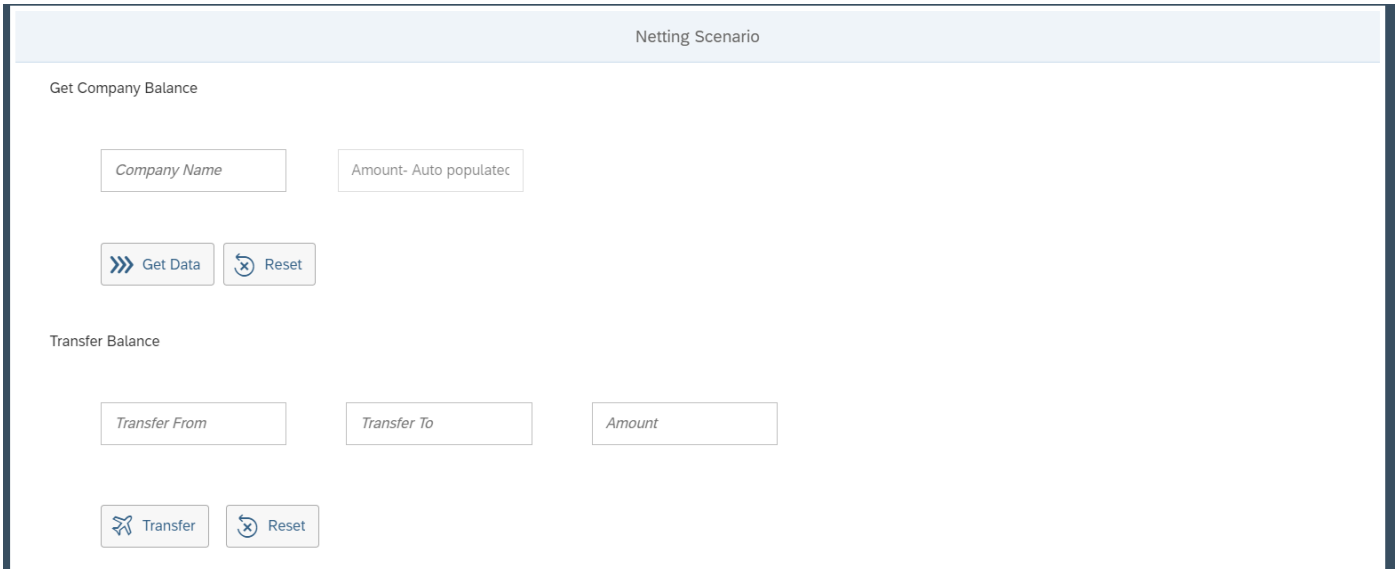


Screenshot 11: Properties for button which reset the input data



Screenshot 12: Prototype Designed (Partial)

17. Similarly, we would add elements for transferring the balance. Final prototype is as shown in the Screenshot 13.

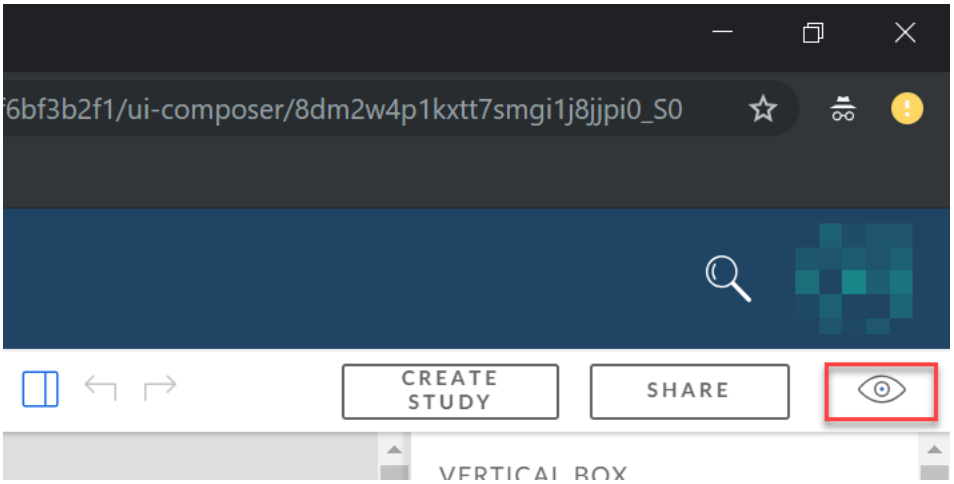


Screenshot 13: Final Prototype

18. This prototype would have following Children in right bottom corner of the build tool.

→ Vertical Box ("main container")
- Title ("Get Company Balance")
- Horizontal Box ("container to hold input box")
  - Company Name - input box
  - Amount – auto populated - input box
- Horizontal Box ("container to hold buttons")
  - Get Data – button
  - Reset – button
- Title ("Transfer Balance")
- Horizontal box ("container to hold input box")
  - Transfer from - input box
  - Transfer to - input box
  - Amount to be transferred - input box
- Horizontal Box ("container to hold buttons")
  - Transfer – button
  - Reset - button

19. You could click on preview button ("eye like symbol") just under your profile name in the top right corner as shown in screenshot 14.

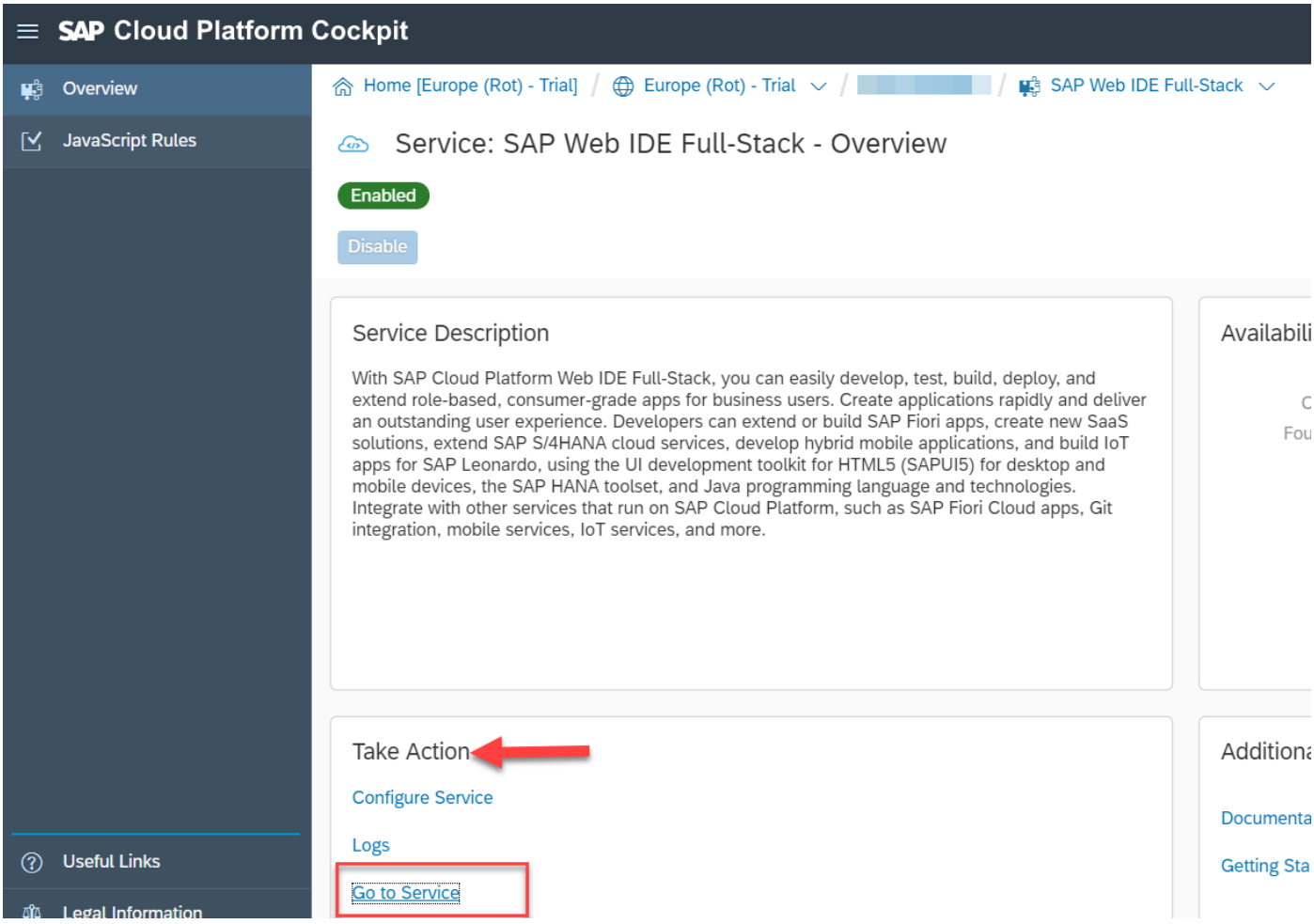


Screenshot 14: Preview button with Eye symbol

20. Click on share button as shown in screenshot 14, your prototype will be published and ready for colleagues and user to explore. You would have 3 options, share the link, share via slack or download as zip file. I have downloaded the zip file which will be useful when we import the project into SAP web IDE.

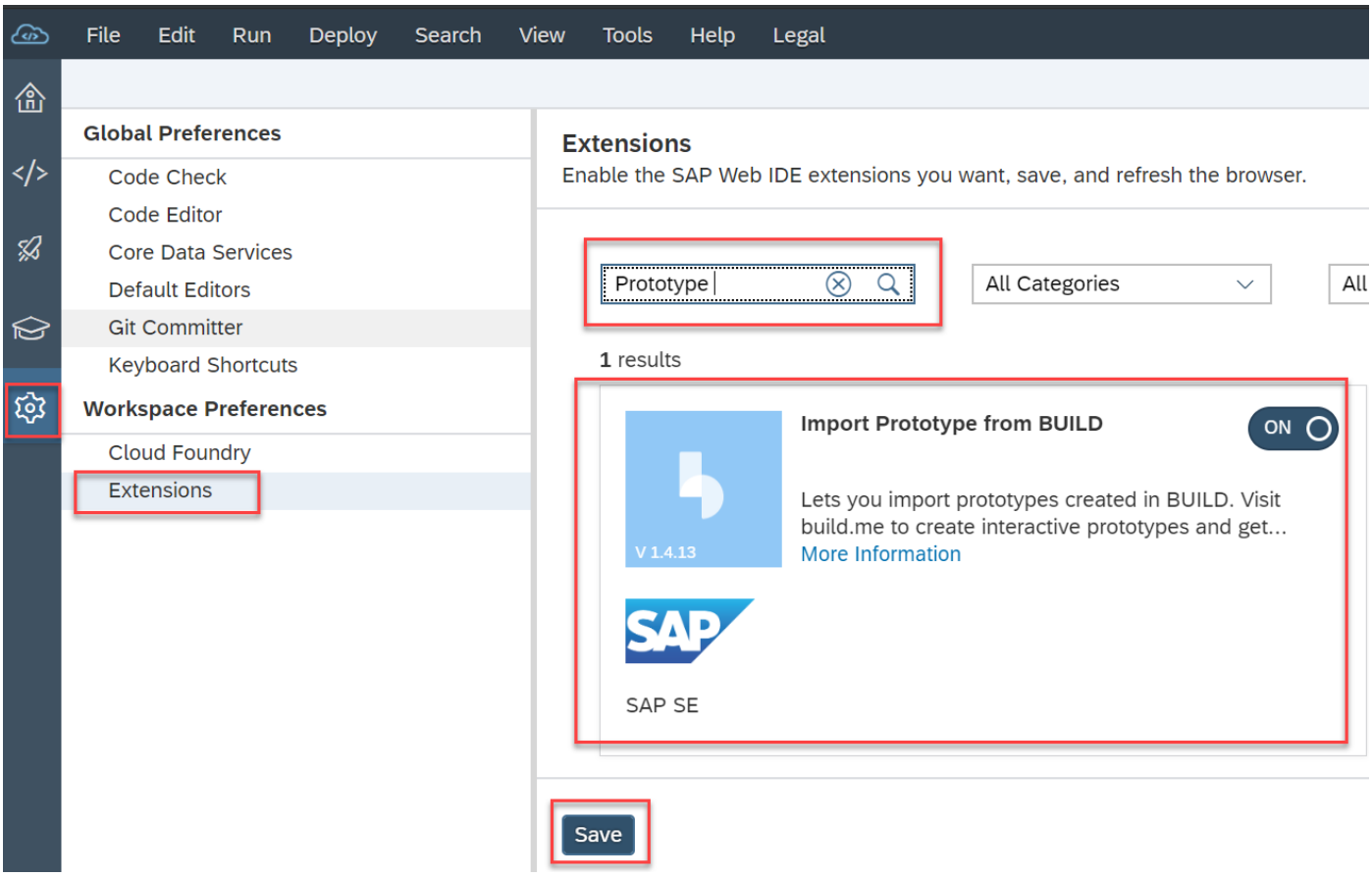
Import the Build project in the SAP Web IDE

1. Logon to your SAP portal.
2. Go to Neo Workspace
3. Click on services (in the left-side menu)
4. Search for SAP Web IDE and Click on "Go to Service" under Take Action Section as shown in the Screenshot 1.



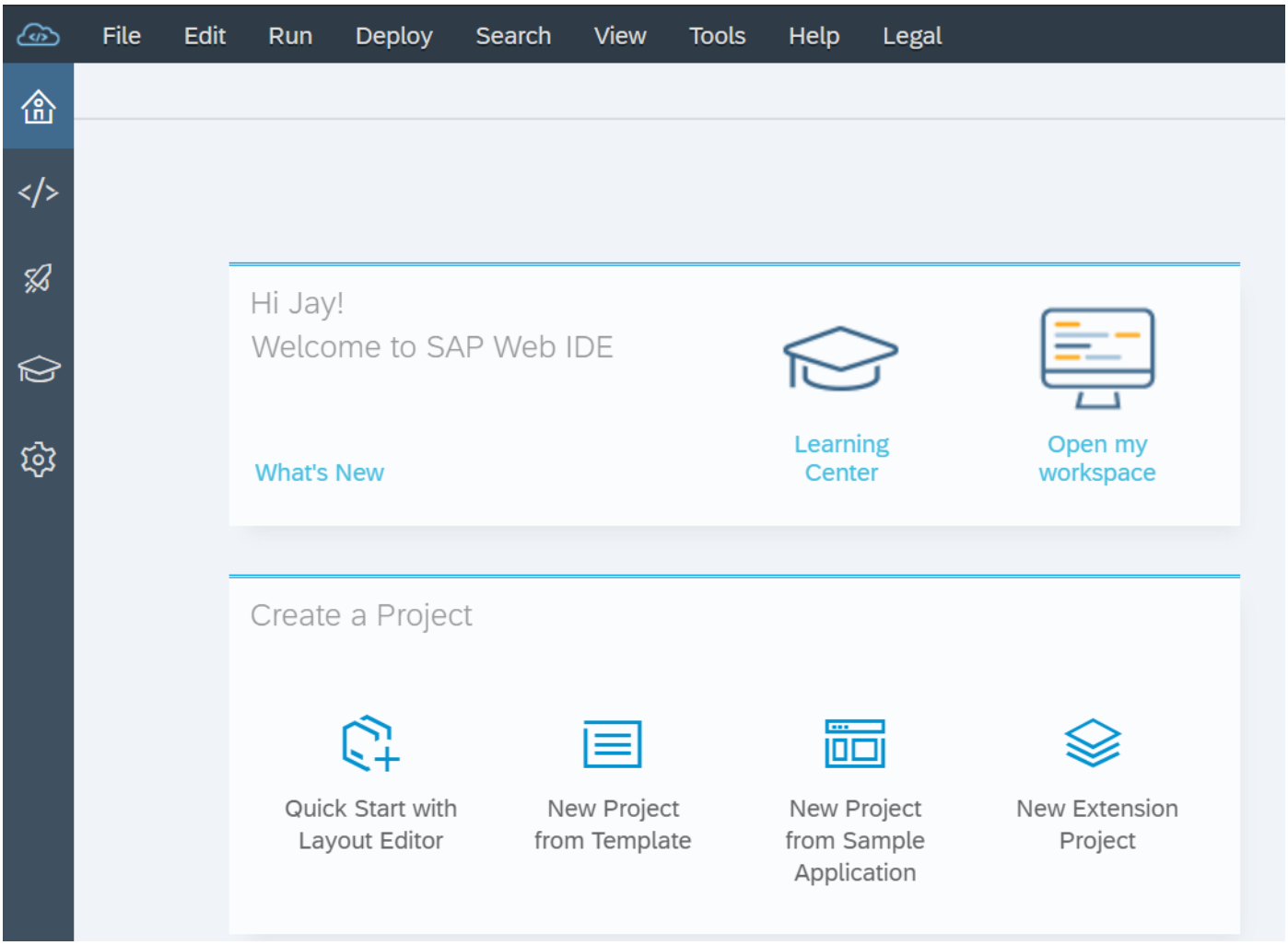
Screenshot 1: SAP Web IDE – Click on Go to Service

5. Click on "settings" Icon (last Icon in the left pane). In the workspace preference, click on extensions. Enter "Prototype" in the search bar and enable "Import Prototype from BUILD" plugin and click on save as shown in screenshot 2.



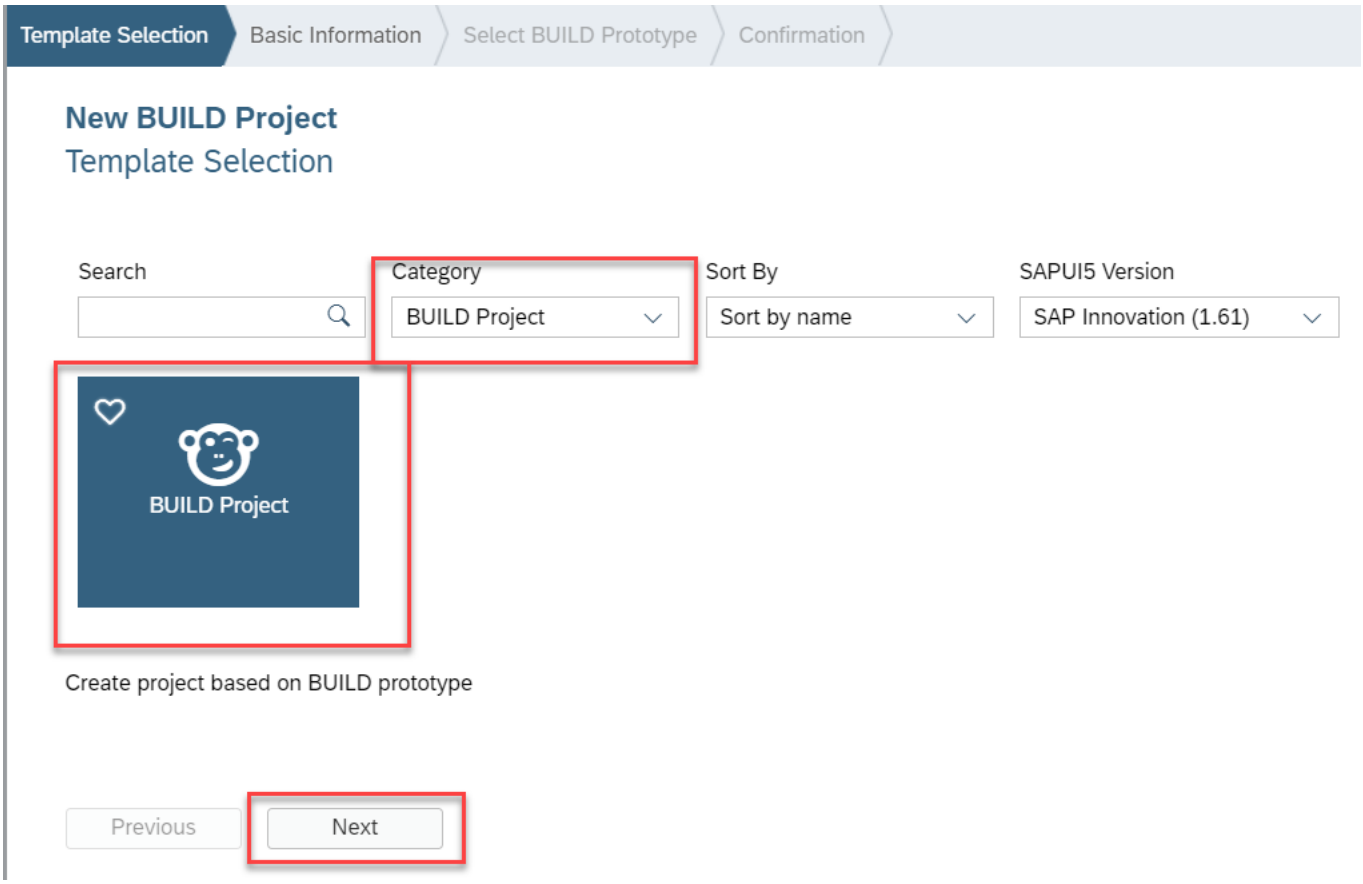
Screenshot 2: Enable plugin Import Prototype from BUILD

6. Click on the home icon, in the left pane and select “New Project from Template” from Create a project section.



Screenshot 3: New project from Template

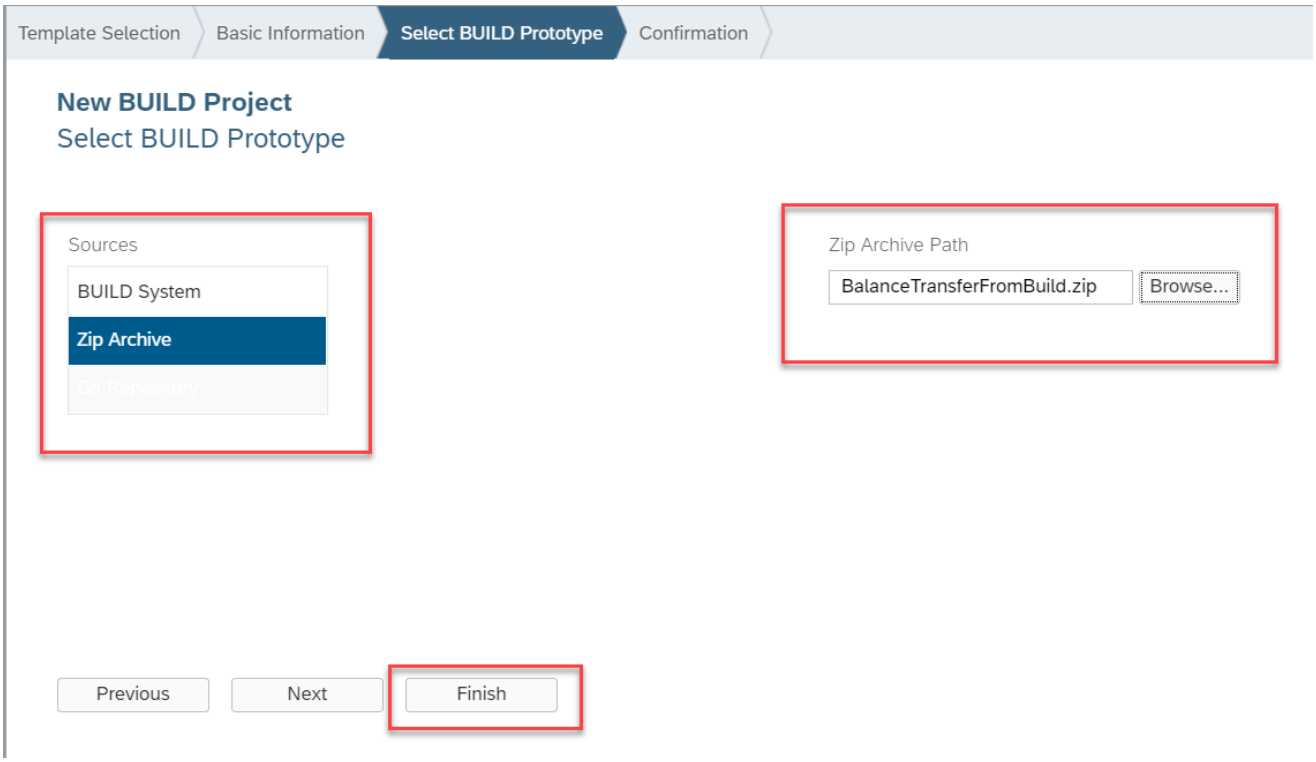
7. Search “Build” in the category and select the build project as shown in screenshot 4 and click on Next.



Screenshot 4: Build project

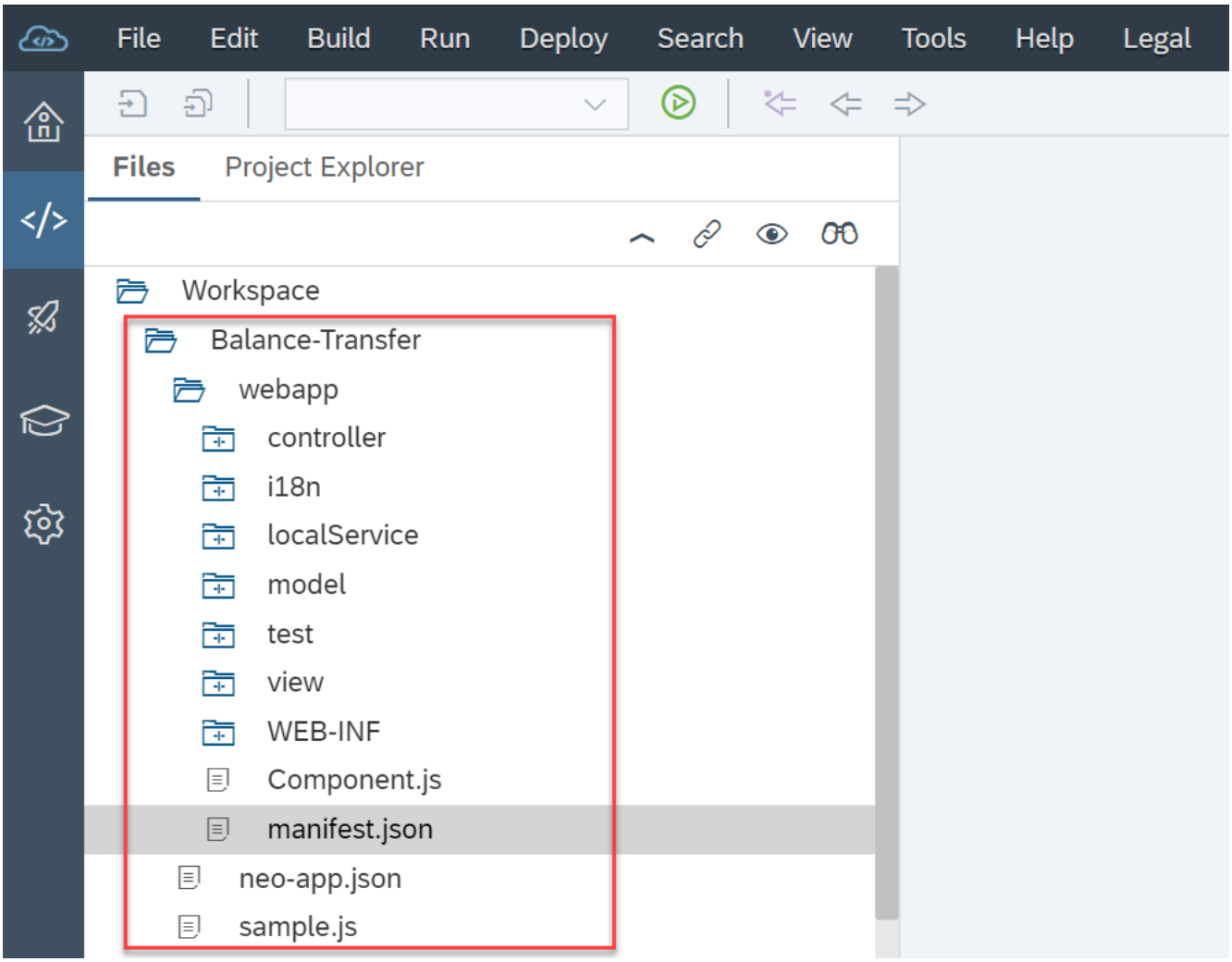
8. Write the name of the project. I have given name as “Balance-Transfer” and click on next.

9. There are 2 sources, from where a build project could be imported. Build system and Zip Archive. Click on Zip Archive and browse the project from the local file system as shown in the screenshot 5. Click on Finish.



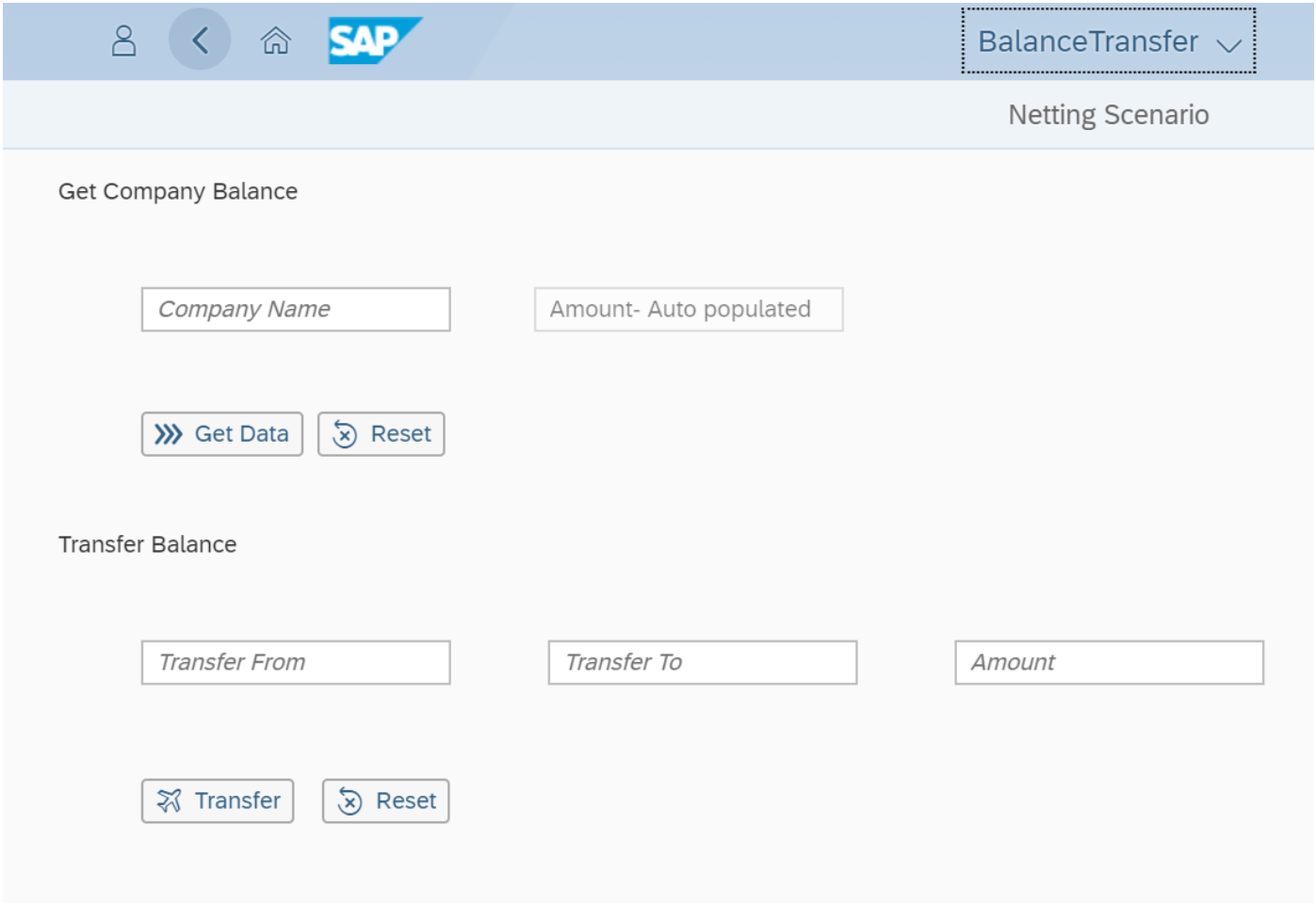
Screenshot 5: Upload Build prototype

10. You would have the file structure in your SAP Web IDE workspace as shown in the screenshot 6.



Screenshot 6: Imported Build Project

11. Right click on project name, click on Run followed by run as web-application and click on testFLPService.html. A new tab will load the prototype you designed using SAP Build tool as shown in screenshot 7.

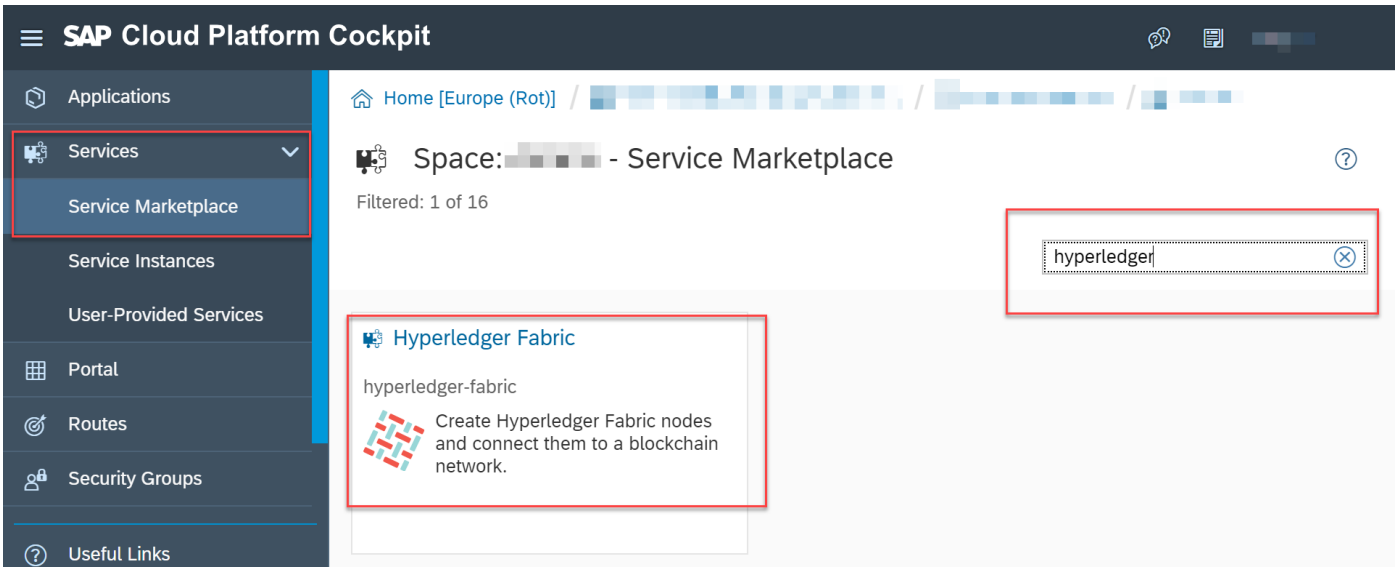


Screenshot 7: Build project deployed on SAP Web IDE

Step 3: Create Application in SAP Web IDE

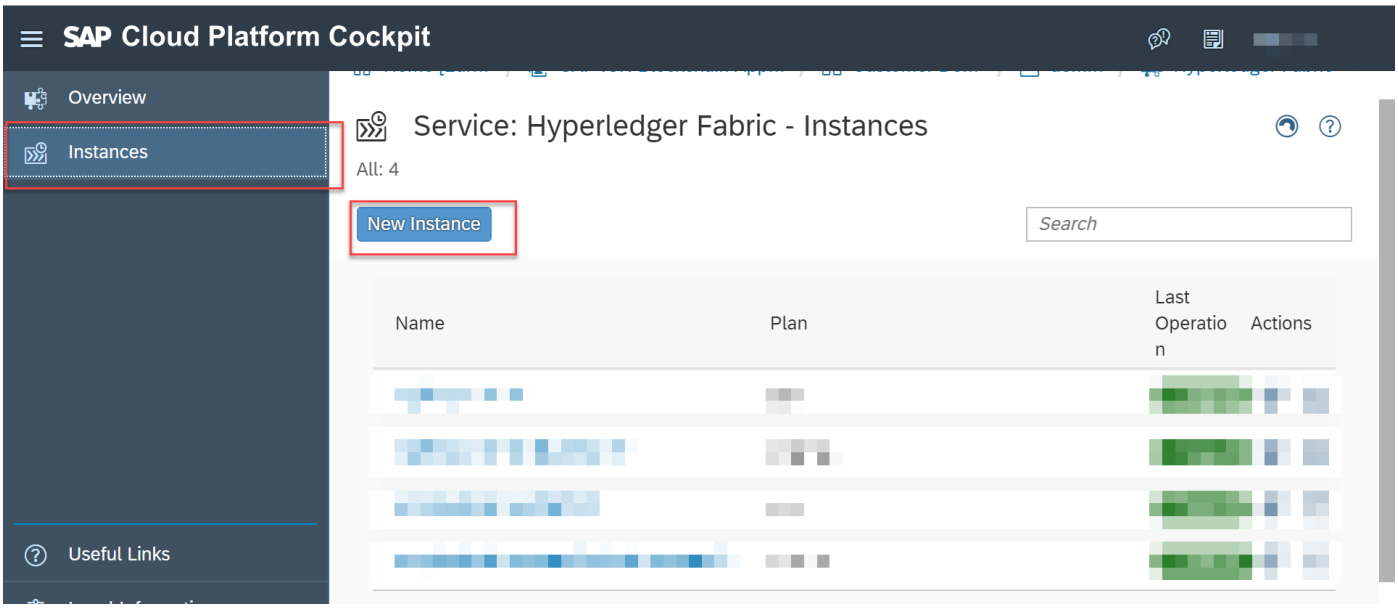
Part a: Hyper-ledger service: Create hyper-ledger service and smart contract code for the specific use case described in pre-requisite.

1. Logon to SAP Cloud Platform.
2. Go to Cloud Foundry landscape.
3. Select the subaccount and space.
4. Select Service Marketplace and search for Hyperledger-fabric service as shown in screenshot 1.



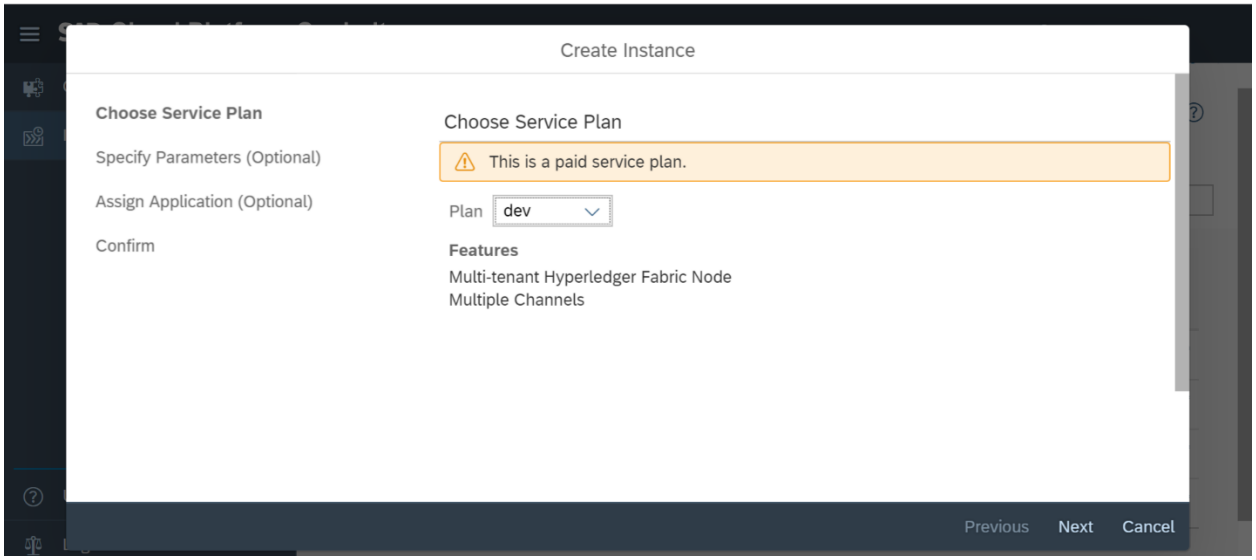
Screenshot 1: Select Hyperledger-Fabric service.

5. Selecting Hyperledger fabric service would allow us to create hyperledger-fabric service instance. Select instances in the left menu as shown in the screenshot 2. This is the place where all the created service instance would be listed.

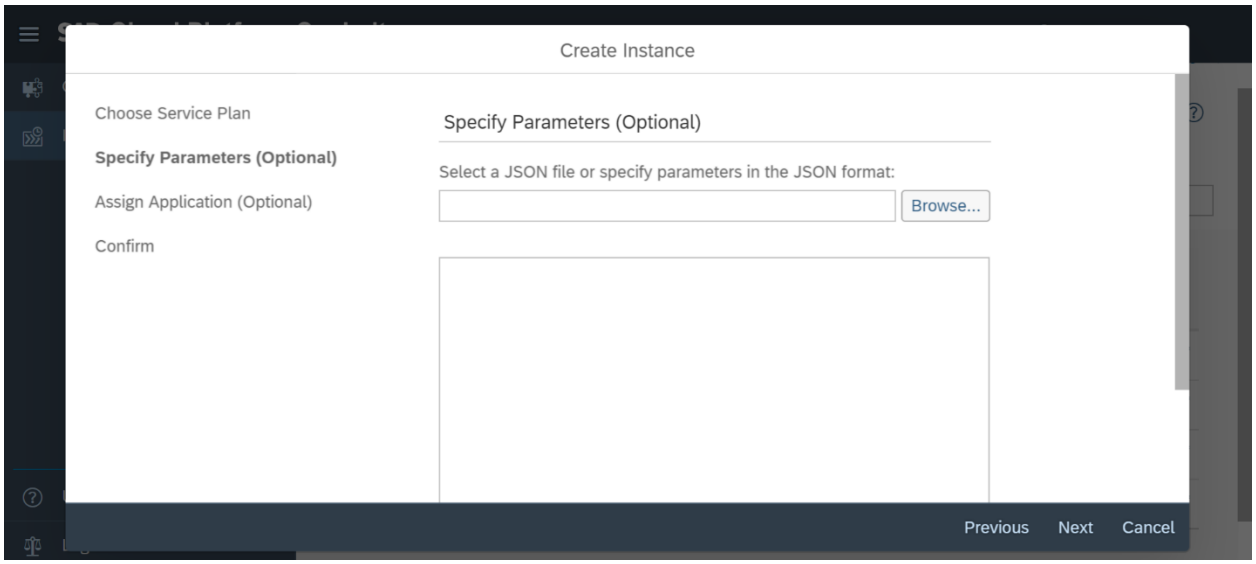


Screenshot 2: Select instances from left menu

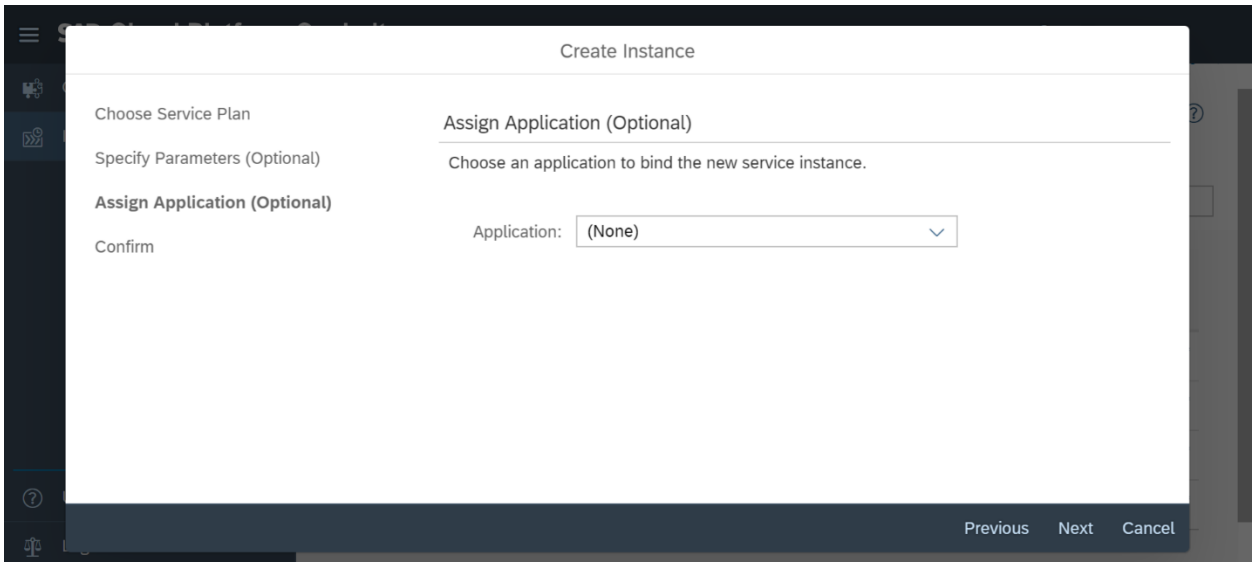
6. Click on “New Instance” which will open a dialog-box. Follow Screenshot 3a, 3b, 3c and, 3d to create the service instance.



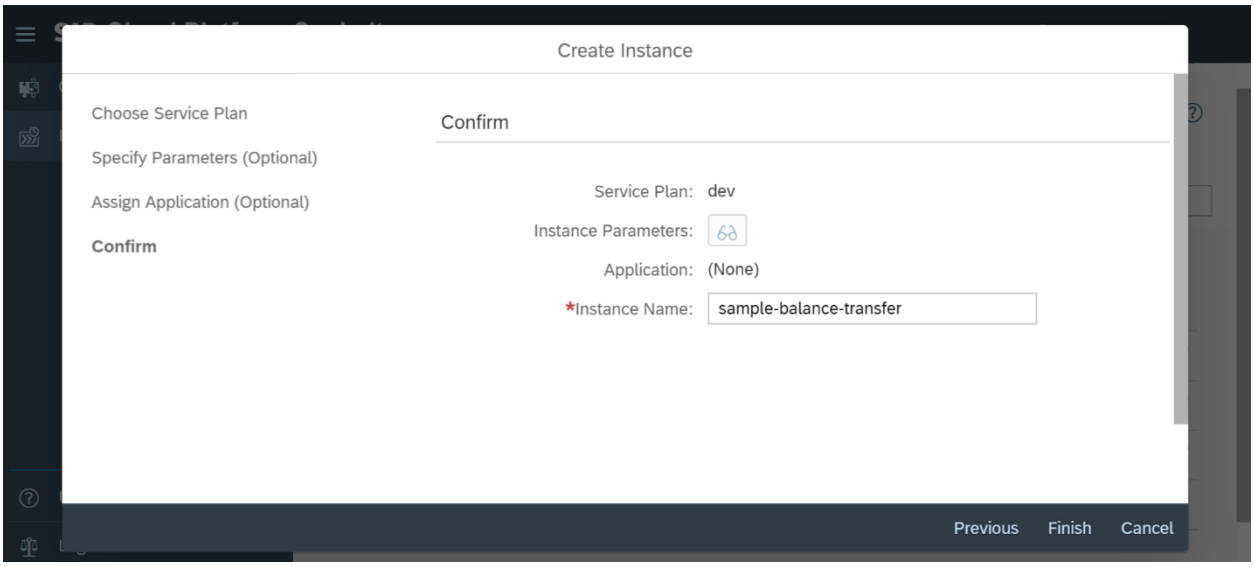
Screenshot 3a. Select the service plan



Screenshot 3b. Specify Parameters. (Keep empty as of now)

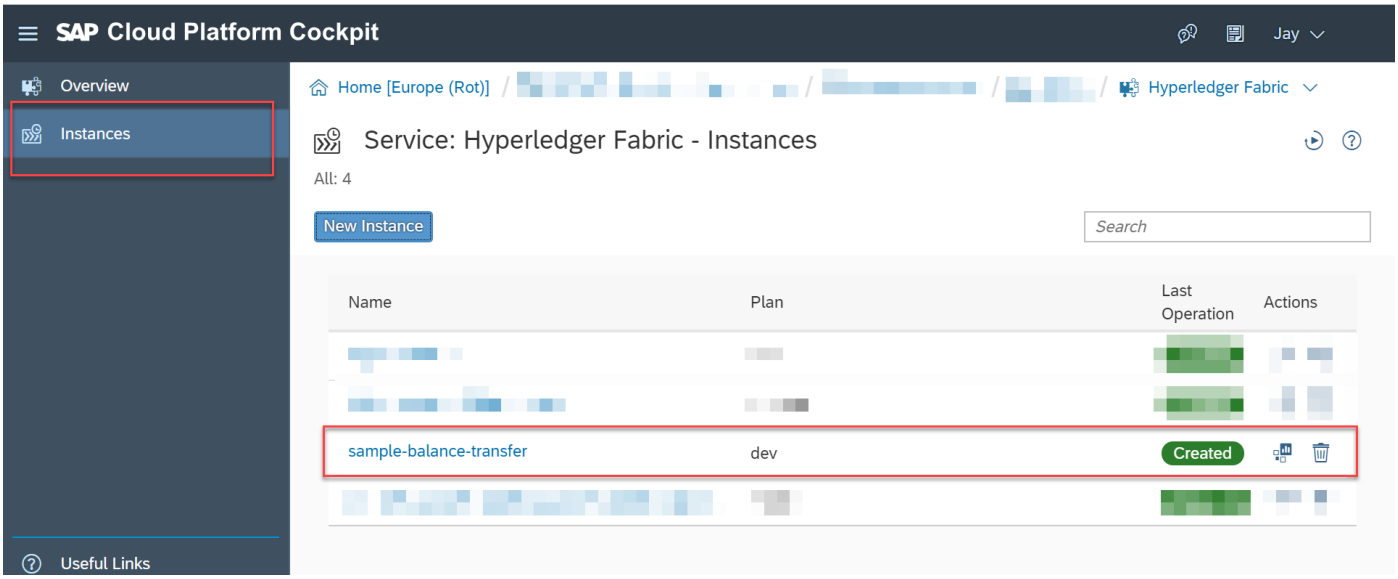


Screenshot 3c. Bind service instance to existing application



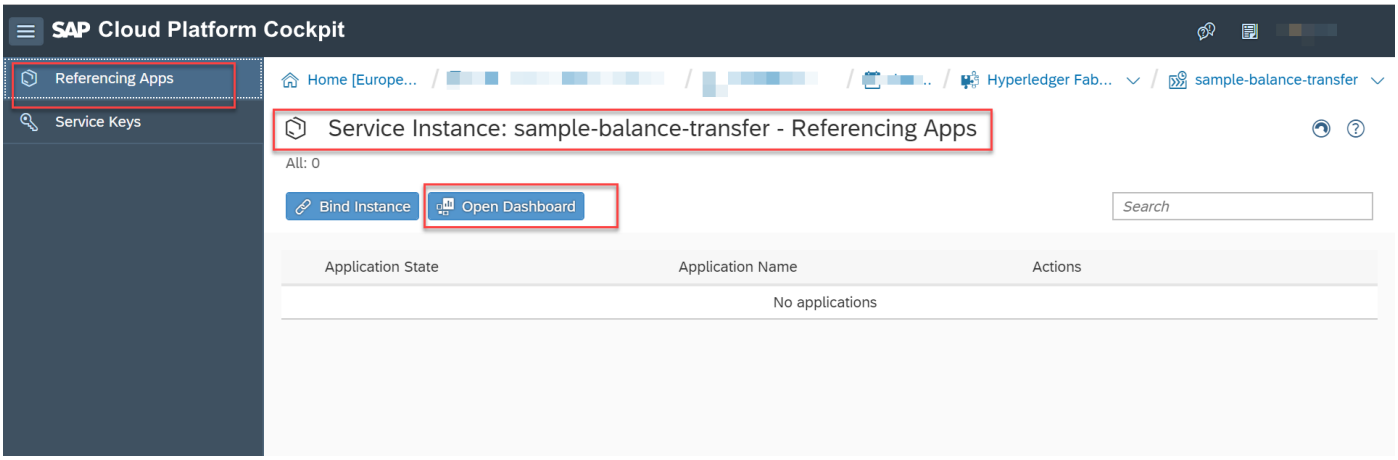
Screenshot 3d. Give Instance name and click finish.

7. Sample-balance-transfer instance with “dev” service plan will be created as shown in screenshot 4.



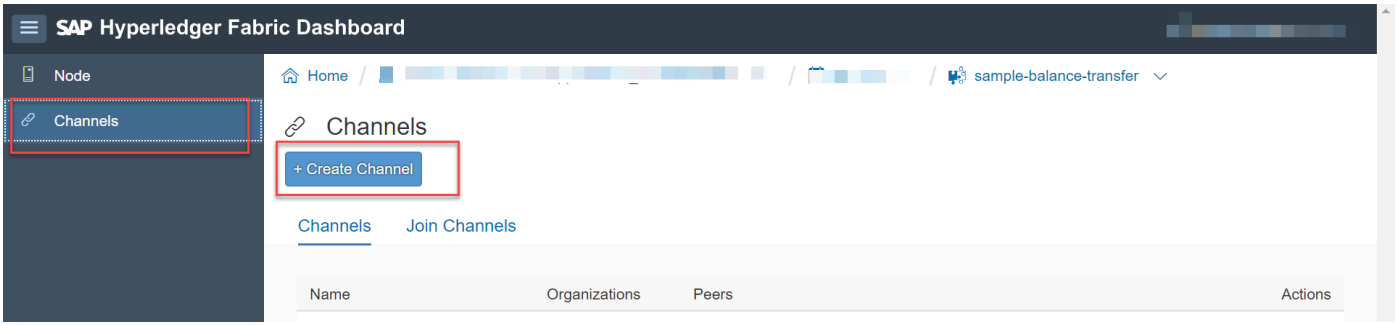
Screenshot 4: Hyper-ledger fabric service instance listed

8. Select the sample-balance-transfer service instance which you just created. And click on open dashboard as shown in screenshot 5.



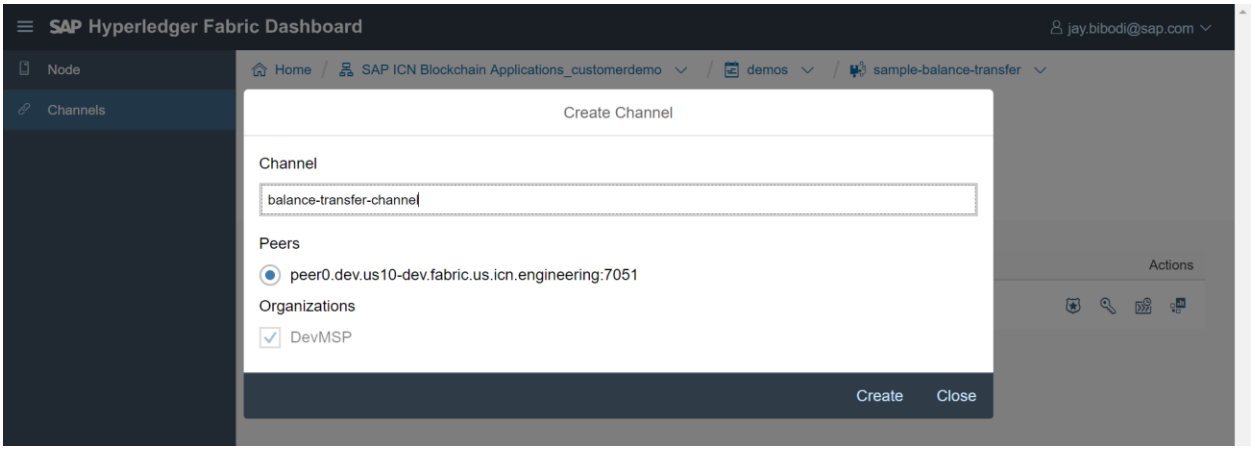
Screenshot 5: Click on Open Dashboard

9. Dashboard will open in the new tab. It contains Node and Channel information. Click on channel from the menu on the left side as shown in Screenshot 6.



Screenshot 6: All channels will be listed under channels tab

10. Click on create channel which will open a dialog box. Follow the screenshot 7 to create channel associated with the node with dev service plan created in point #7

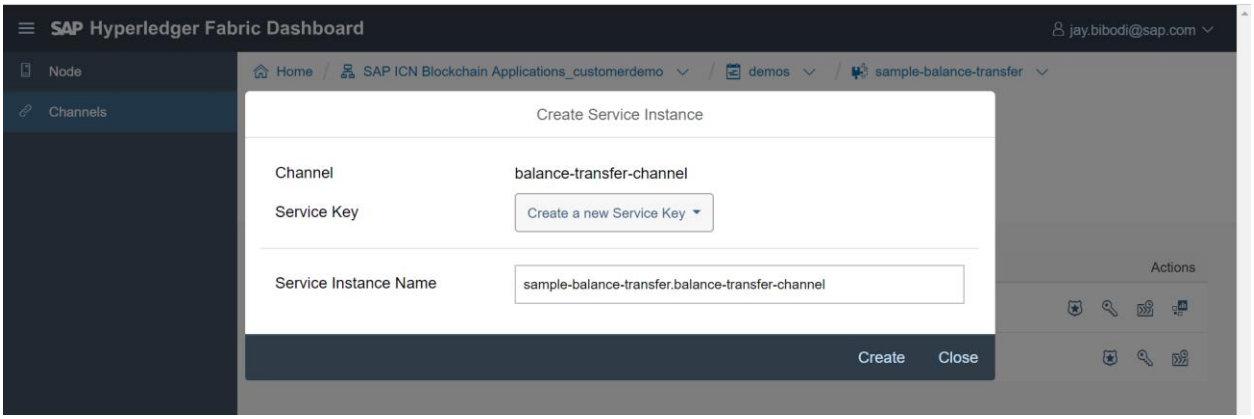


Screenshot 7: Create channel

11. Created channel will be listed under channel's tab, as shown in screenshot 8.

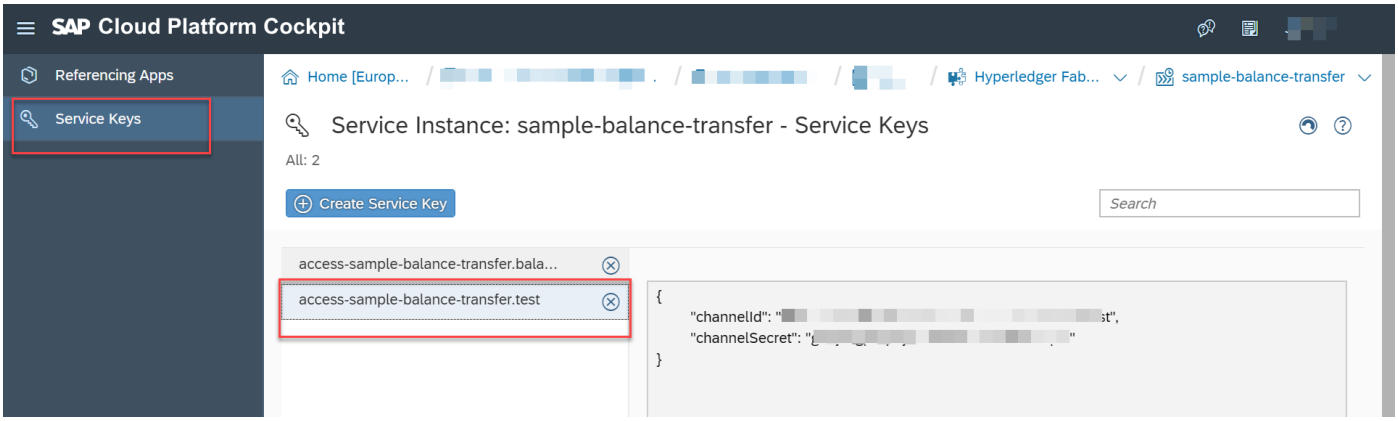
Screenshot 8: All channels listed for a specific Node

12. Select 3rd button under the Actions tab which could be seen in screenshot 8. It signifies creating a channel service instance in the same space. On clicking 3rd button under actions tab, you will see a dialog box fill details as shown in Screenshot 9.



Screenshot 9: Creating channel service instance

13. This created new service key in screenshot 9, is associated with the specific node. It will help node know which channels are associated with that specific node. You could verify this by visiting dashboard of node service instance (Instance which we created using dev service plan) and selecting Service key tab in the left menu as shown in screenshot 10.

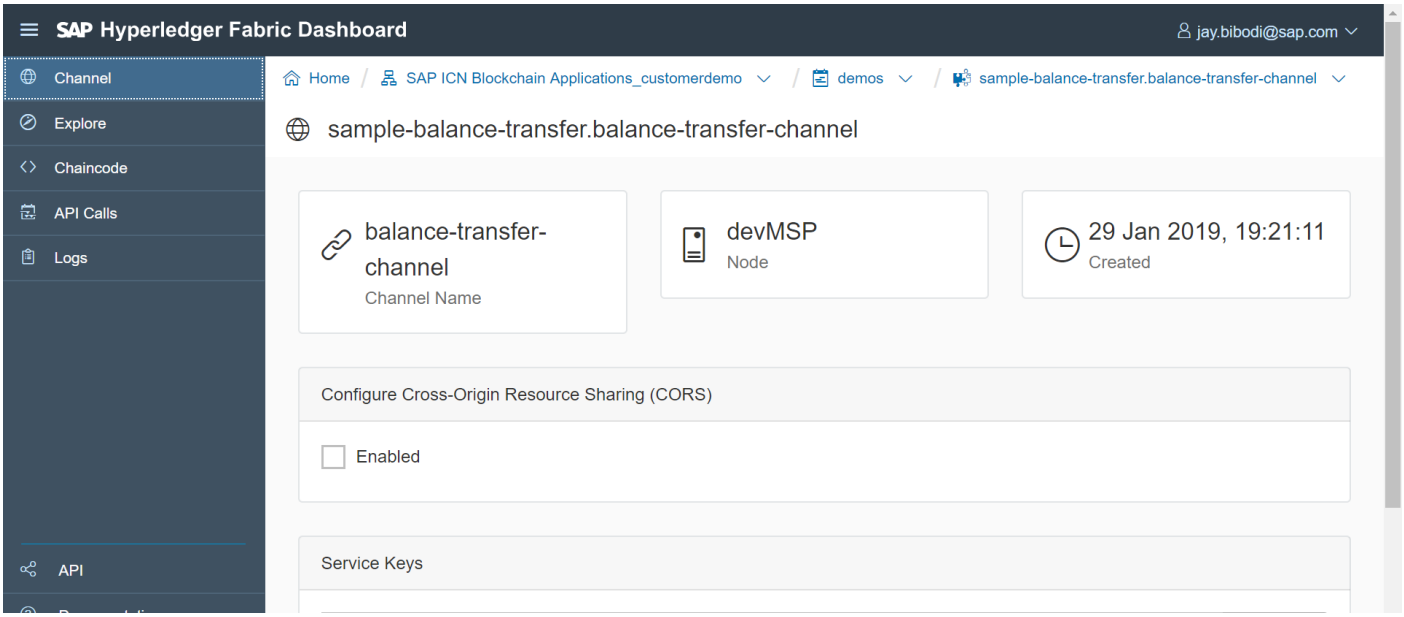


Screenshot 10: Service keys for the channel service instance

14. Open channel service instance dashboard, you can find this, in the left menu of the node dashboard. Click on 4th button under the actions tab as shown in screenshot 11

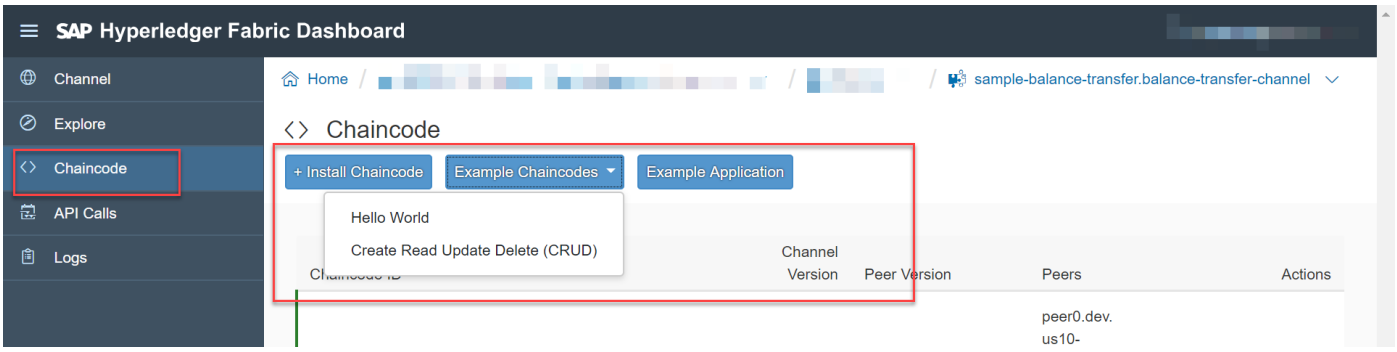
Screenshot 11: Open channel service instance dashboard

15. Channel service dashboard is shown in the screenshot 12. Names might differ based on the names of the node and channel you entered



Screenshot 12: Channel service instance dashboard

16. You could download the example chaincode by selecting chaincode option from the left menu as shown in the screenshot 13.



Screenshot 13: Example chaincodes

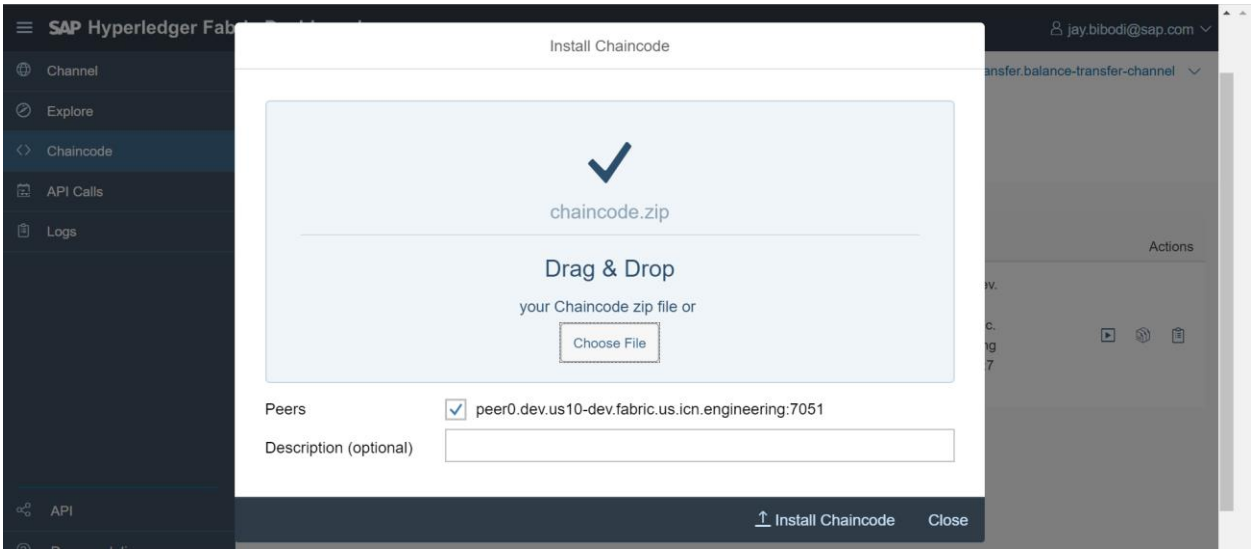
17. To install chaincode, we have a specific structure. This structure is listed below:

- src/netting_scenario.go
- src/netting_scenario.yaml

- chaincode.yaml

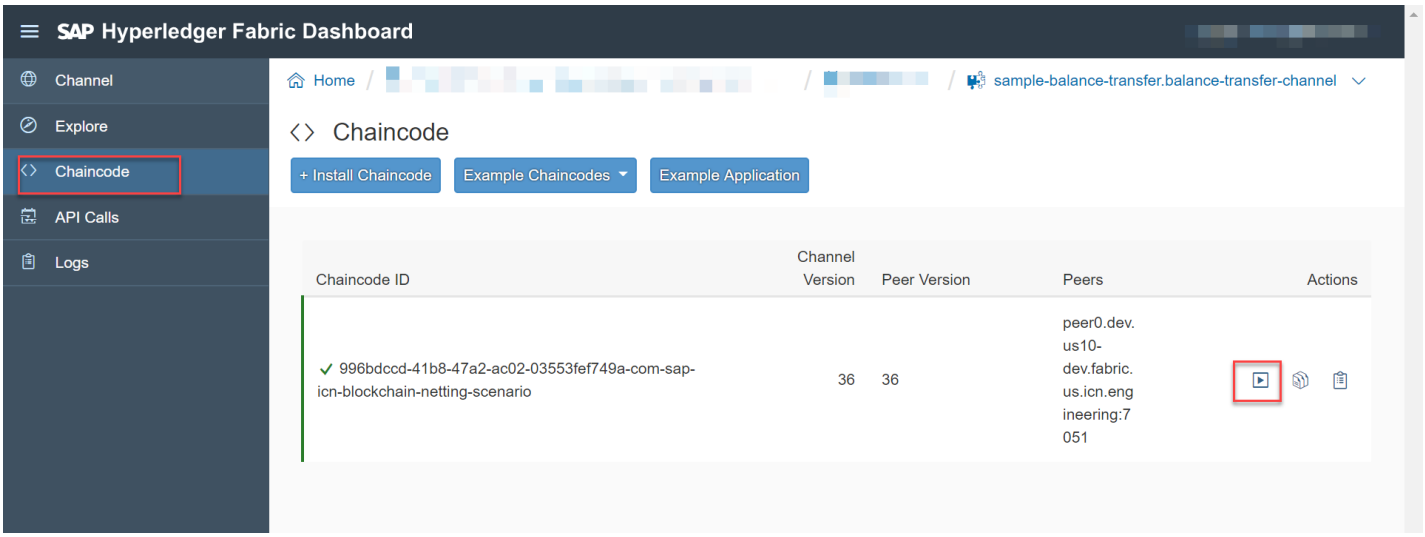
create a zip file and upload it to dialog box while installing a chaincode as discussed in next point. Zip file for the use case I am describing could be found here

18. Install the chain code by clicking on install chaincode in screenshot 13. Upload zip file you downloaded in point 17. Dialog box would look like screenshot 14



Screenshot 14: Installing chain code

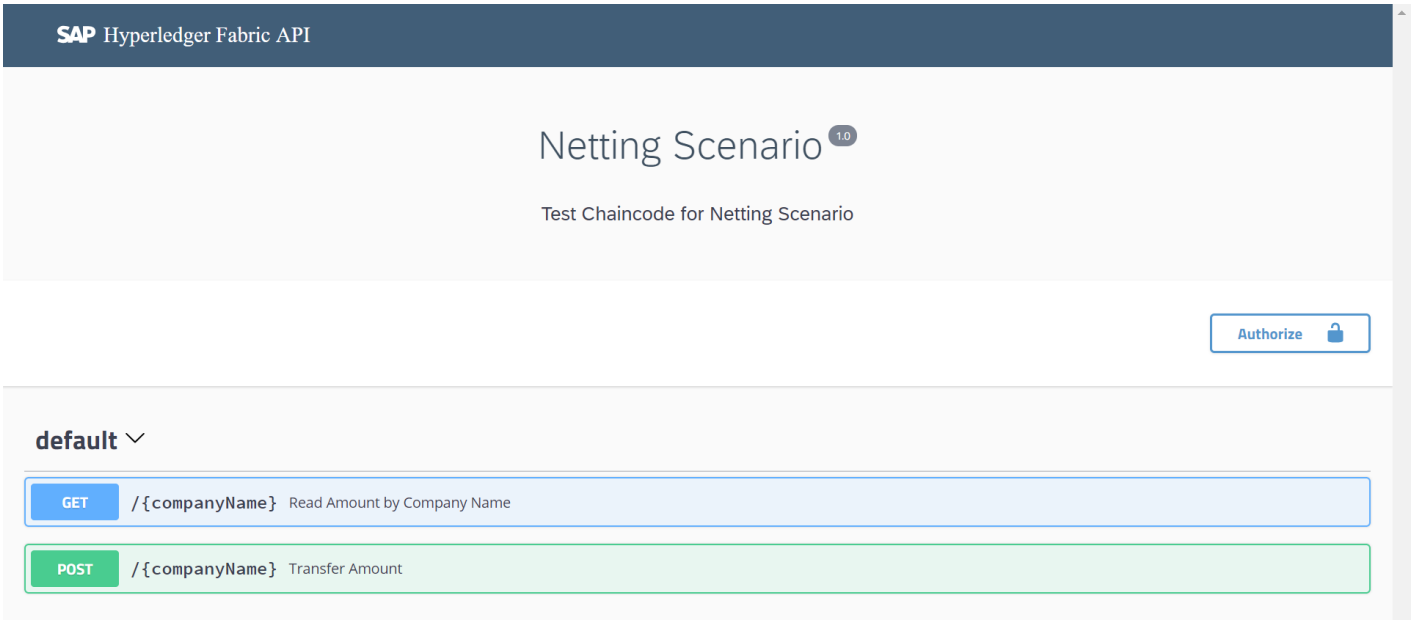
19. Once chain code is installed, you could see chain code listed as shown in screenshot 15.



Screenshot 15: All the chaincode installed are listed here for a specific channel

20. Under the actions tab, in the screenshot 15, select on 1st button. This will allow us to test the chaincode we installed via swagger API's.

Note: Important details for a service key's will be discussed in the next section when we configure Node JS service with the hyper-ledger fabric channel.



Screenshot 16: Authorize and test chaincode deployed

Part b: Backend layer: Create NodeJS application using SAP Web IDE and write required code to make use case work as expected.

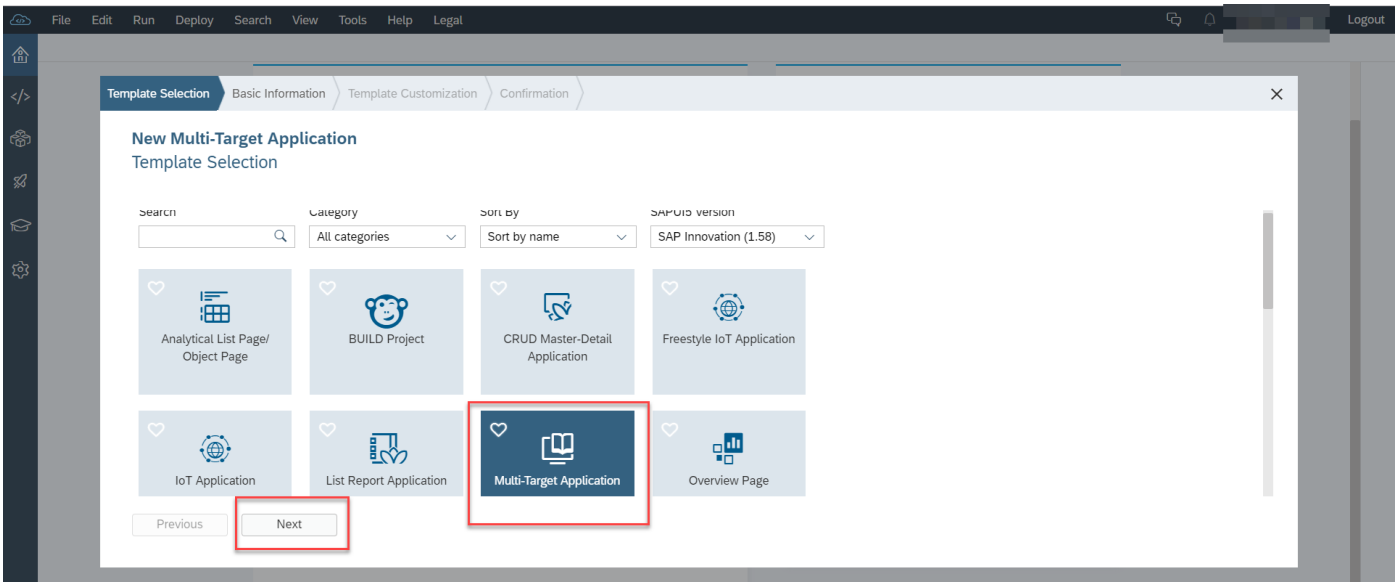
You could create application in two ways,

a. Multi-Target Application

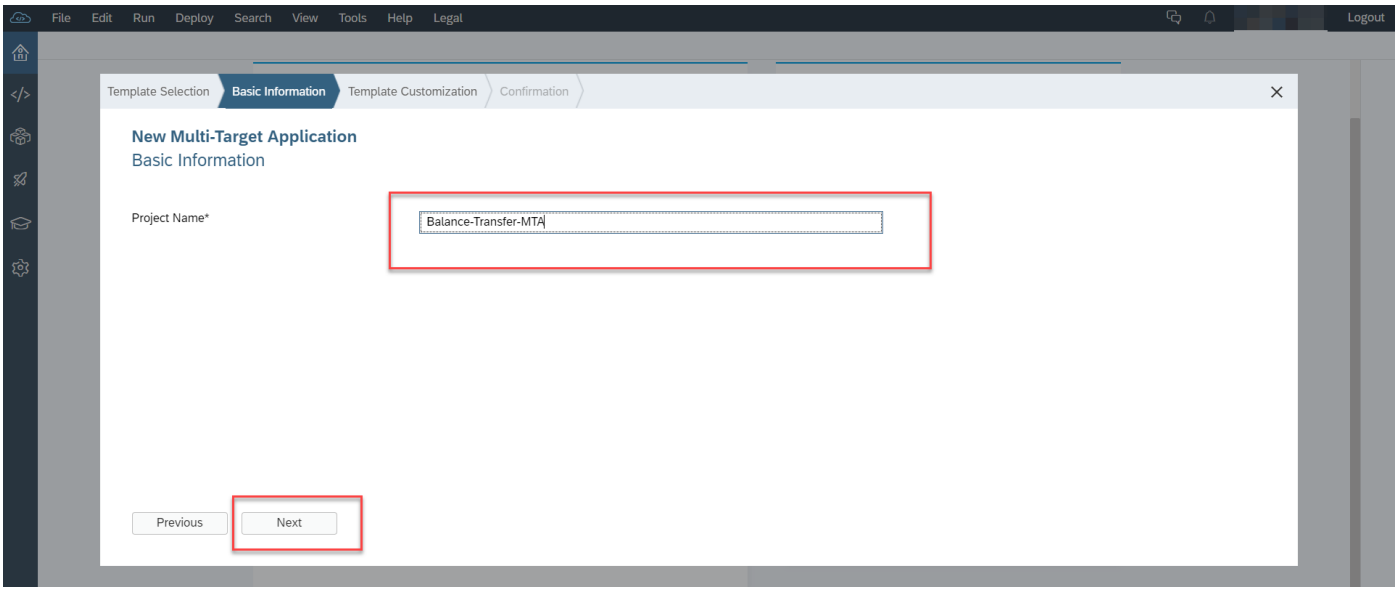
b. SAP Cloud Platform Business Application

In this tutorial, I have worked with Multi-Target Application abbreviated as MTA because I wanted to learn how could we use and interact with different modules using MTA.

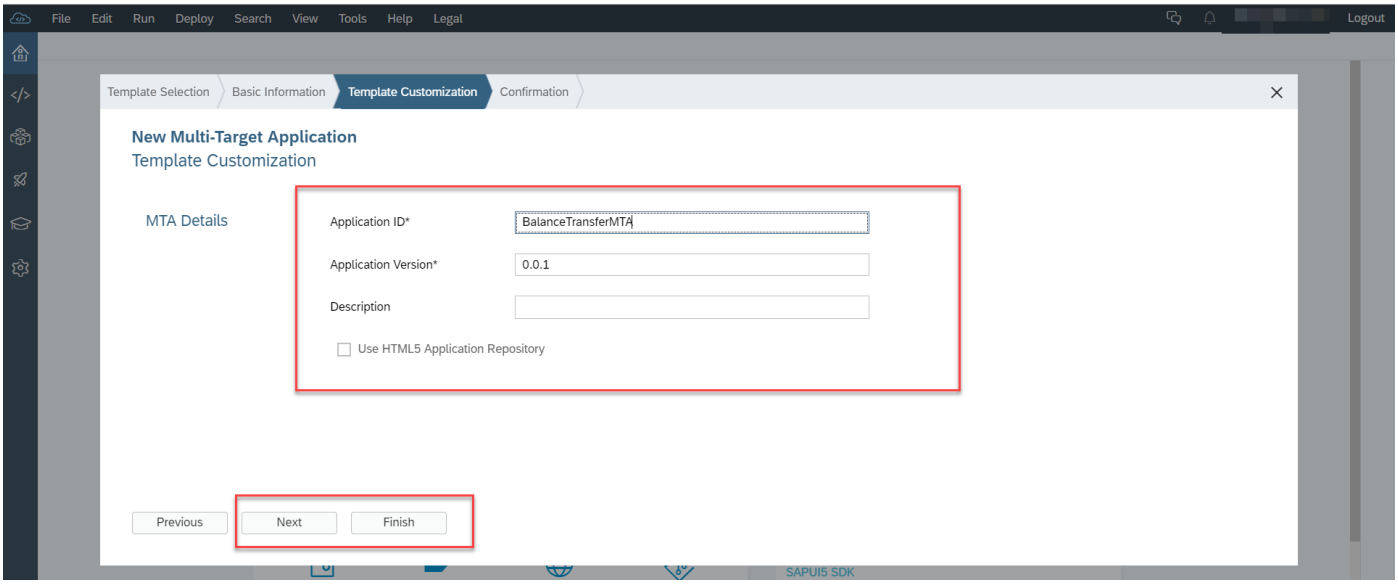
1. Go to SAP Web IDE where you already BUILD project.
2. Click on “New Project from Template” or press Ctrl+Alt+Shift+O. Fill up the details as shown in the screenshot 1a, 1b, 1c and, 1d.



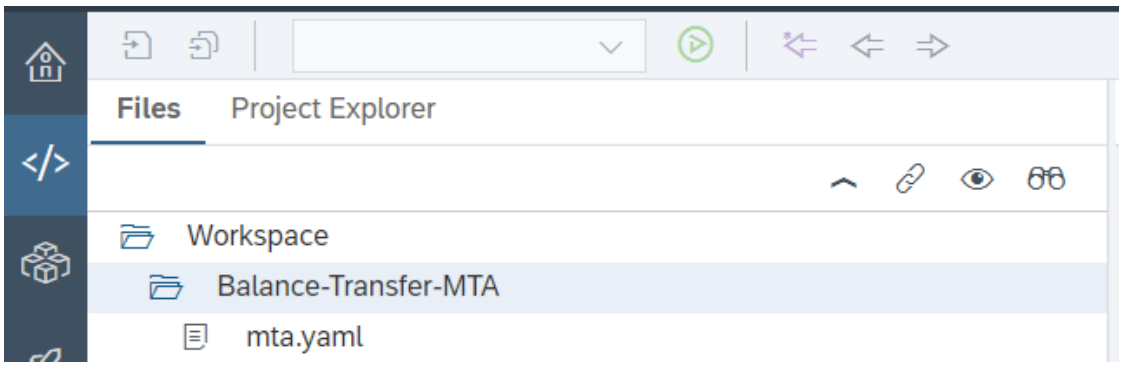
Screenshot 1a: Select template



Screenshot 1b: Give Project name (Ex: Balance-Transfer-MTA)

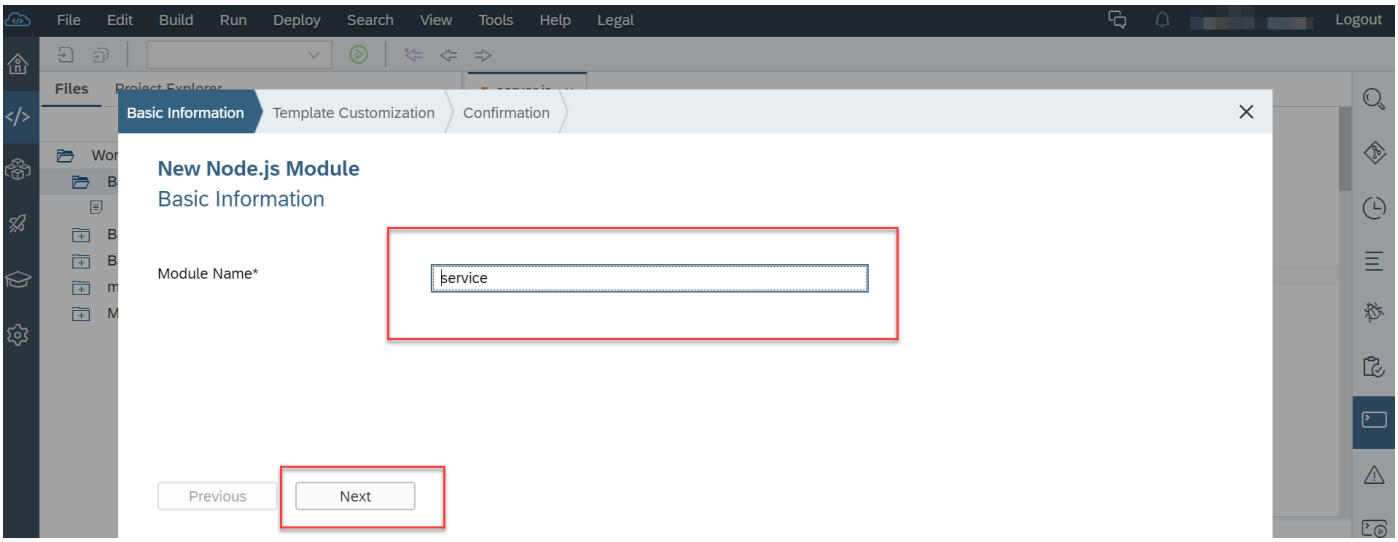


Screenshot 1c: Enter application ID, version and description

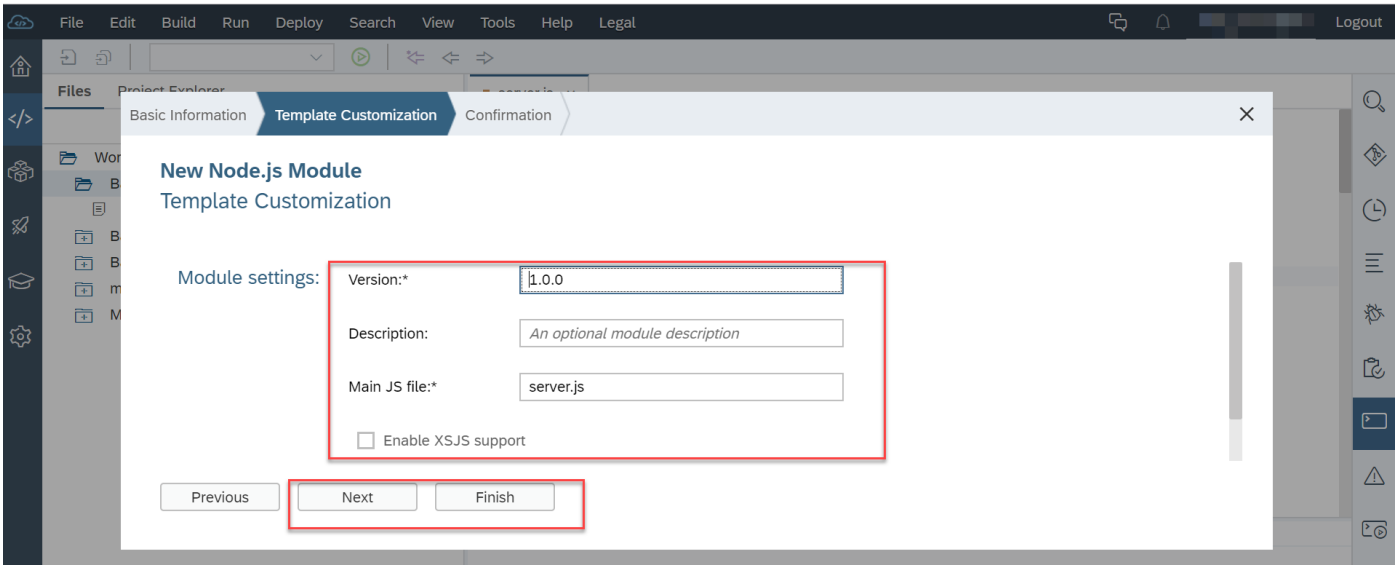


Screenshot 1d: Workspace with MTA project structure

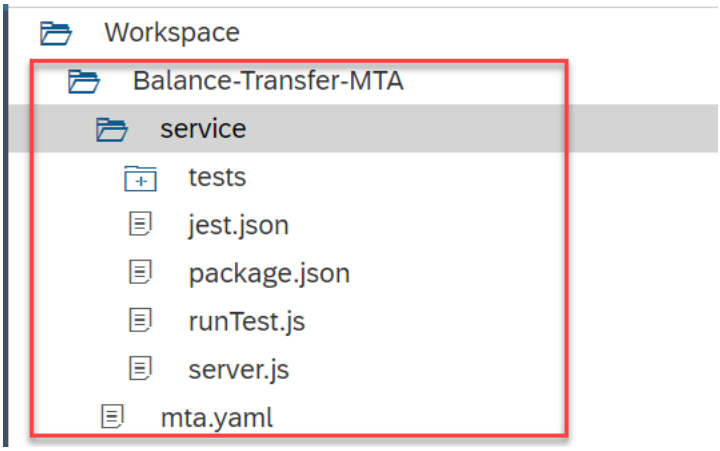
3. Right click project name, select New -> Node.js Module. This will open a dialog box where we will enter module details as shown in screenshot 2a, 2b and, 2c.



Screenshot 2a: Enter Node.js module name



Screenshot 2b: Enter version number and main js file name



Screenshot 2c: Project structure after adding Node.js module

4. Observe your mta.yaml file, it should contain code as shown in screenshot 3 after creating Node.js module. It will change when we add other modules and services.

mta.yaml

```yaml
ID: BalanceTransferMTA
_schema-version: '2.1'
version: 0.0.1

modules:
 - name: service
   type: nodejs
   path: service
   provides:
    - name: service_api
      properties:
        url: ${default-url}
```

Screenshot 3: Structure of mta.yaml file

5. I am comfortable with express framework of node js. Also, there are couple of extra node modules required to develop our application which we will add to package.json file under service module. Our package.json file would contain following dev-dependencies. All these dependencies will be installed when we deploy MTA on SCP.

"jest": "^23.6.0",

"jest-junit": "^5.2.0",

"btoa": "^1.2.1",

"cookie-parser": "~1.4.3",

"debug": "~2.6.9",

"express": "~4.16.0",

"http-errors": "~1.6.2",

"morgan": "~1.9.0",

"pug": "2.0.0-beta11",

"request": "^2.88.0",

"request-promise": "^4.2.2",

"rimraf": "^2.6.2",

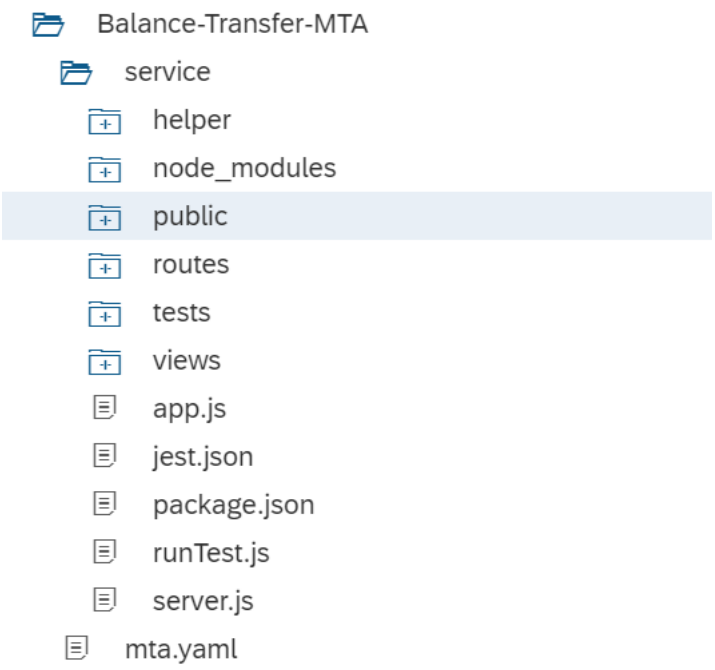
"@sap/grunt-sapui5-bestpractice-build": "1.3.62",

"@sap/grunt-sapui5-bestpractice-test": "2.0.0",

"@sap/approuter": "5.7.0"

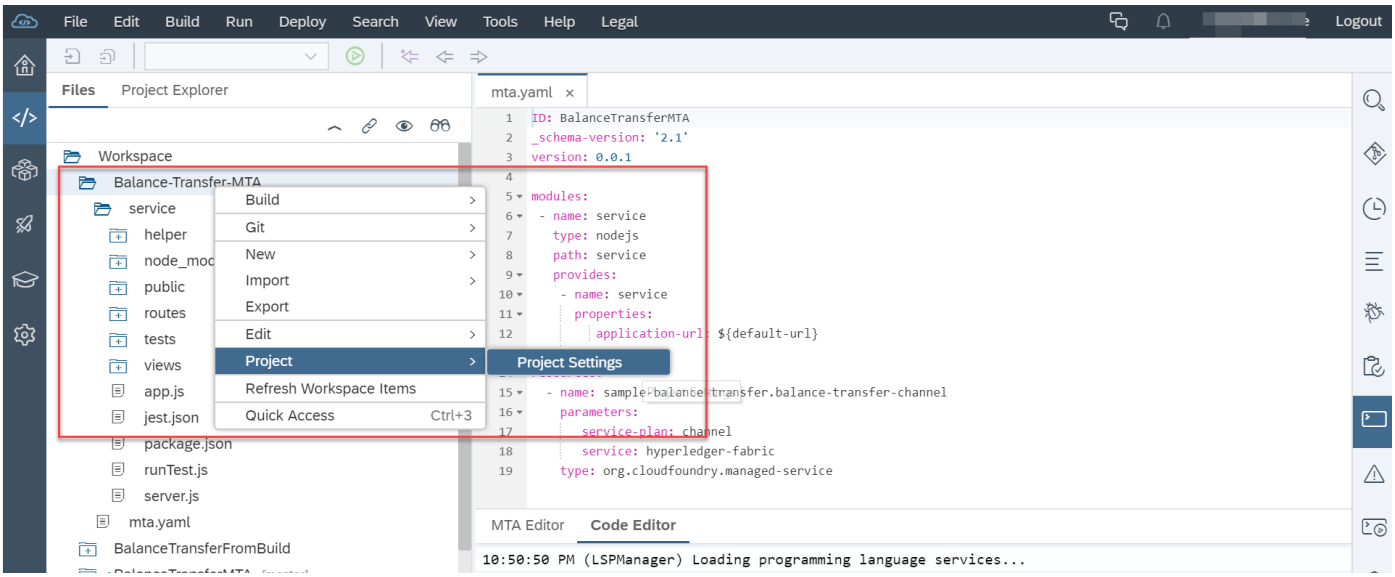
Integrating blockchain service with node js module is explained in Part 4.

Structure of express node js module is as shown in screenshot 4. Detailed code for node js module is found here



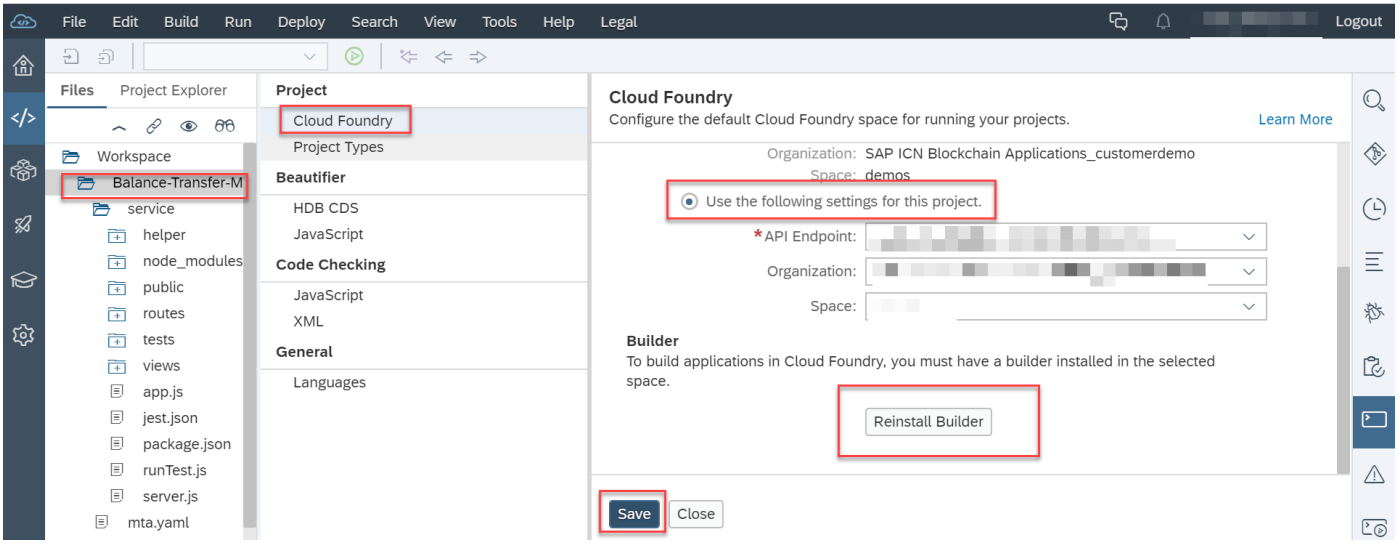
Screenshot 4: Structure of Node js application

6. Configure Cloud Foundry with the project or the workspace. Right click on the project name select project setting as shown in screenshot 5.



Screenshot 5: Select Project Settings

7. Select Cloud Foundry as shown in screenshot 6.



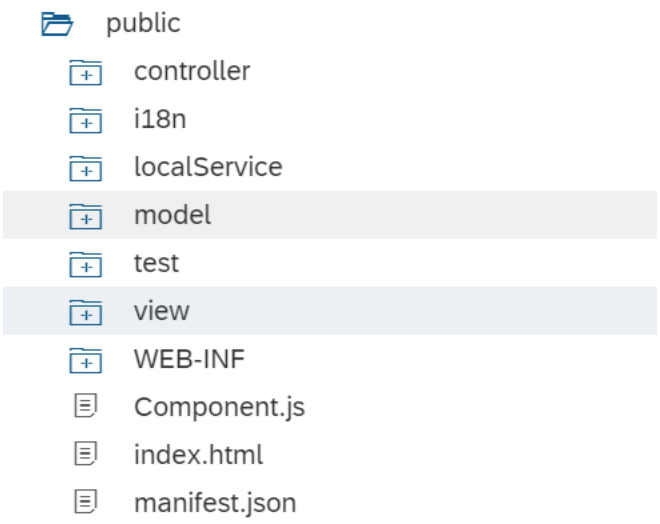
Screenshot 6: Set up Cloud Foundry and save the setting

Part c: Front-end layer: Integrating prototype project in the application using SAP Web IDE

1. Code generated by importing a build project in web ide will be used in our application.
2. The build project deployed in SAP web ide contains the folder Webapp. Copy all the files and folder from Webapp folder and copy it inside public folder of the NodeJS module.
3. “Index.html” file is not present in the webapp folder. Create index.html file and copy code from here for index.html
4. Two changes which you might have to make in index.html file is, on line 16 and 26.

   data-sap-ui-resourceroots='{"ui-space.ui": "./"}' , ui-space.ui should be replaced. Go to public/view/TransferBalance.view.xml and check for path of a controller name.

   Example: <mvc:View xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" controllerName="com.sap.build.standard.balanceTransfer.controller.TransferBalance"> if this is first line of your view.xml, copy and paste red color path in index.html on line 16 and line 26.
5. Structure of public folder in express node js module is as shown in screenshot 1. It contains UI5 code.



Screenshot 1: Structure of public folder contains UI5 code

Step 4: Integrating all 3 layers

Part a: Integrating backend NodeJS application to hyper-ledger service

1. Open mta.yaml file To add the hyper-ledger channel service instance to a nodejs application. You could define the environment variables defined in the mta.yaml file if you know it or whenever you deploy the MTA application, hyper-ledger service keys will be bind to a node js application because, application will be created only when hyper-ledger service instance exists.

   Note: In mta.yaml file, under service module, properties which specify all the keys and required url are not actually required to be specified in mta.yaml file. It will be part of application environment variables when application gets deployed in CF and hyper-ledger service instance is attached to it.

mta.yaml file structure

ID: BalanceTransferMTA

_schema-version: '2.1'

version: 0.0.1

modules:

- name: service

  type: nodejs

  path: service

  properties:

    blockchainServiceName: sample-balance-transfer.balance-transfer-channel

    ACCOUNT_URL: https://customerdemo.authentication.us10.hana.ondemand.com

    …..

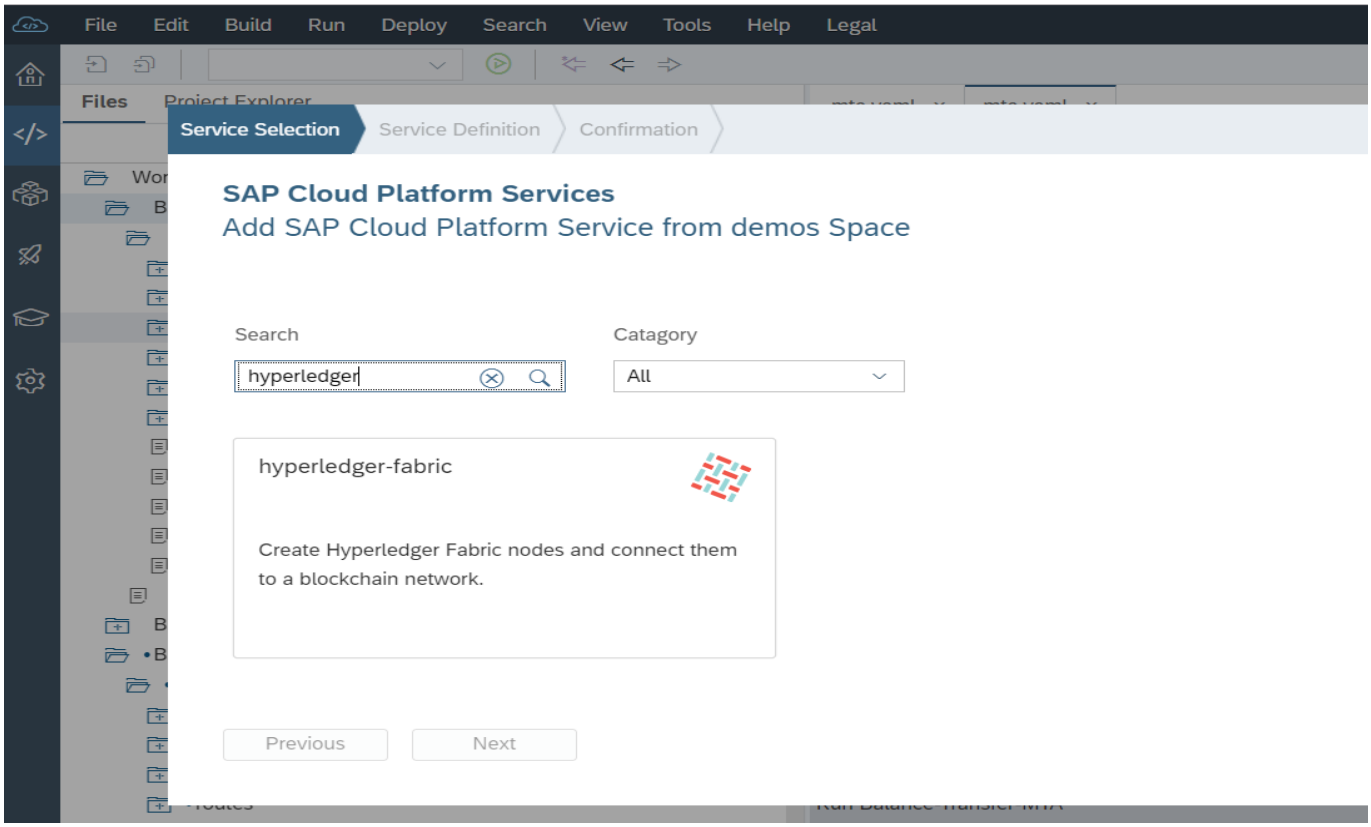
    …..

    …..

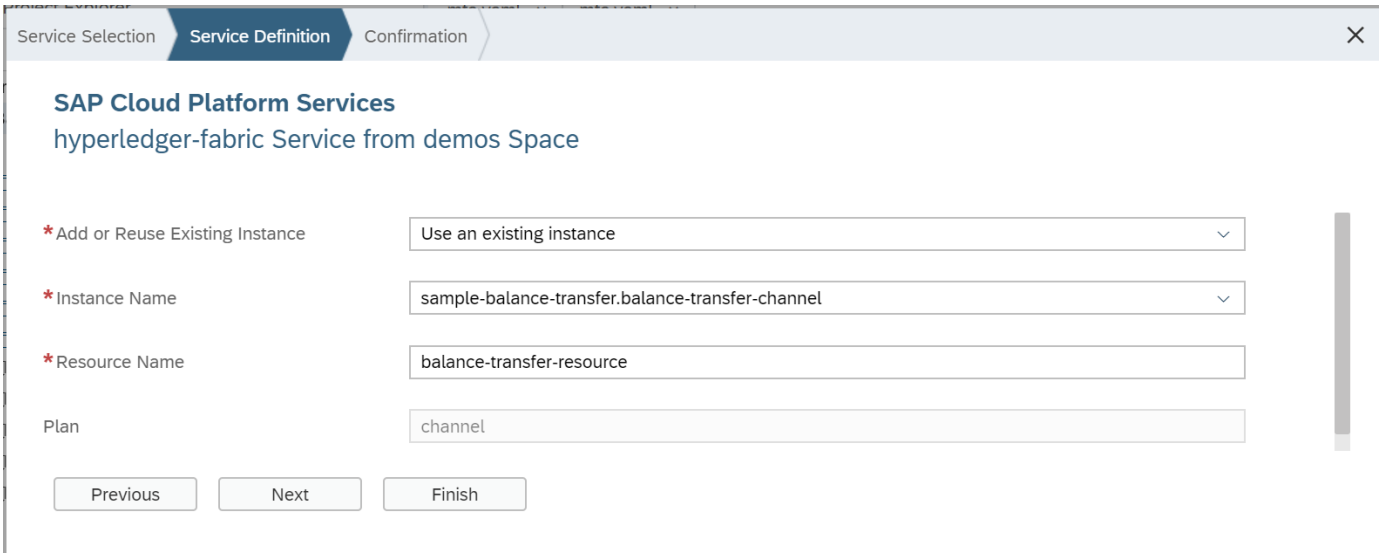
```
provides:
 - name: service
   properties:
     application-url: ${default-url}
requires:
   - name: sample-balance-transfer.balance-transfer-channel


resources:
 - name: sample-balance-transfer.balance-transfer-channel
   parameters:
     service-plan: channel
     service: hyperledger-fabric
   type: org.cloudfoundry.managed-service
```

2. Another way, of adding a blockchain service instance is via GUI, right click on MTA project, select New-> SAP Cloud Platform Service. A dialog box will open which shown in screenshot 1a and 1b. All service will be listed but make sure, your CF account is configured in workspace or for a specific MTA project.



Screenshot 1a: Select hyperledger-fabric service add to MTA project



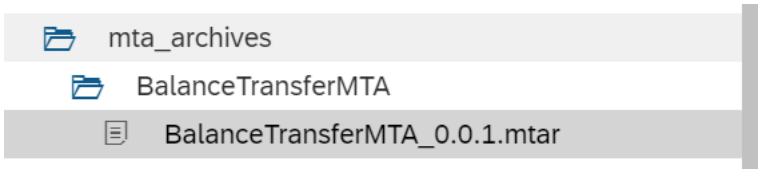
Screenshot 1b: Fill required details about the service instance

Part b: Integrating front-end SAPUI5 application to the backend NodeJS application

Controller and View are two files where we will make changes. View have onclick events which will trigger function in controller. You must make ajax or xhr request from UI5 to node js application to process the request. If you want to look at the code, please visit here.

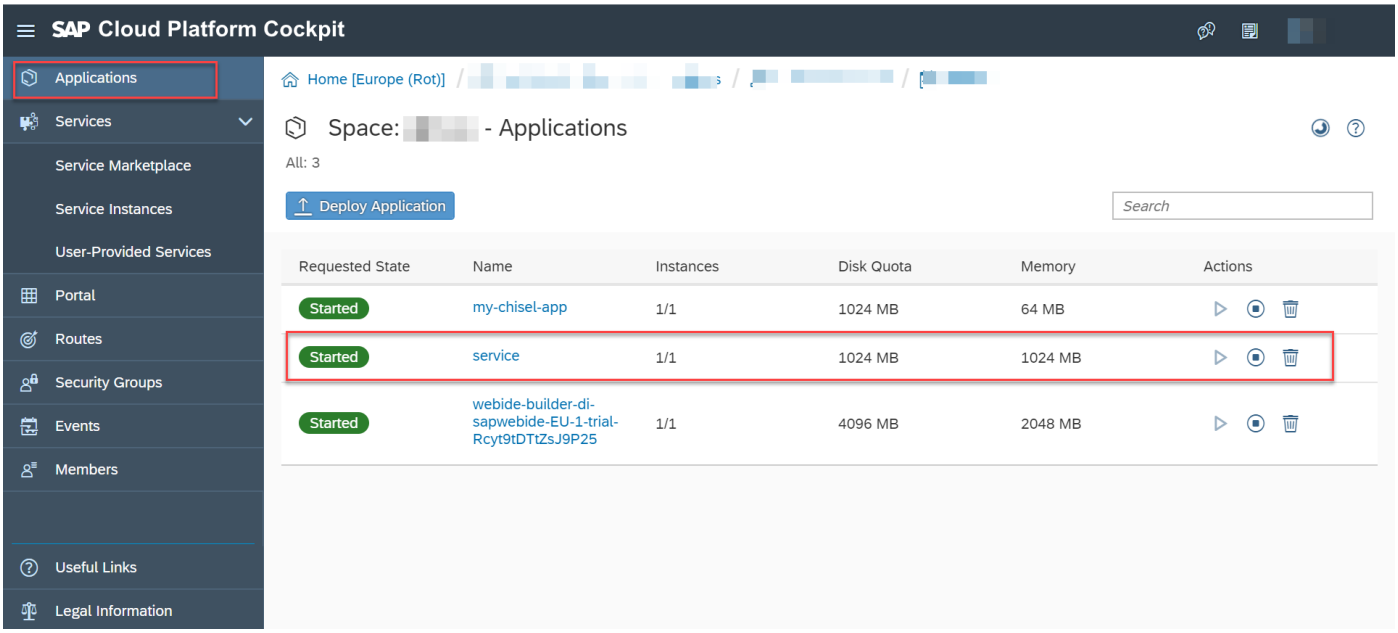
Step 5: Deploy and Test Application

1. Right click on MTA project, select Build -> Build. Build process takes couple of minutes. Once build process is completed a mta_archive folder as shown in the screenshot 1.



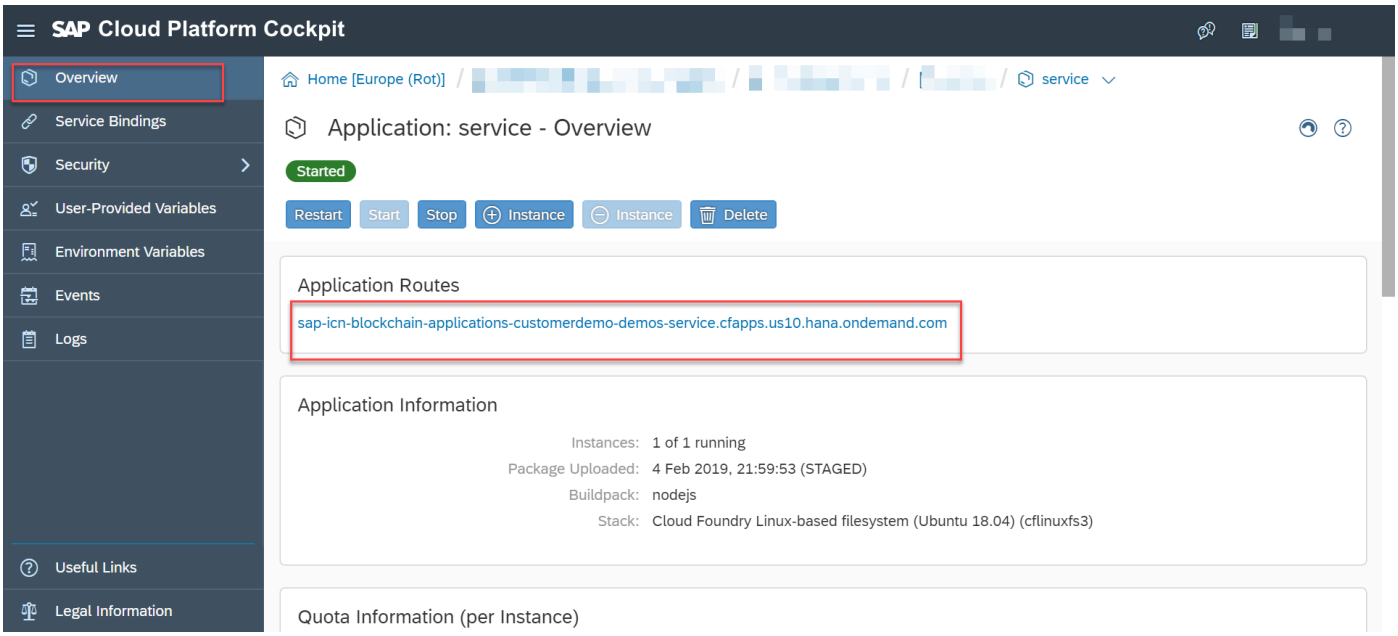
Screenshot 1: mta_archives/BalanceTransferMTA/BalanceTransferMTA_0.0.1.mtar

2. Right click on mtar file shown in screenshot 1 and select Deploy-> Deploy to SAP Cloud Platform.
3. Deployment will finish successfully after few minutes.
4. To test the application, logon to your SCP account. Go to CF instance and click on application. Name of the application will be same as name of module for MTA application unless you change the same in mta.yaml file. Screenshot 2 shows list of all applications deployed in CF.



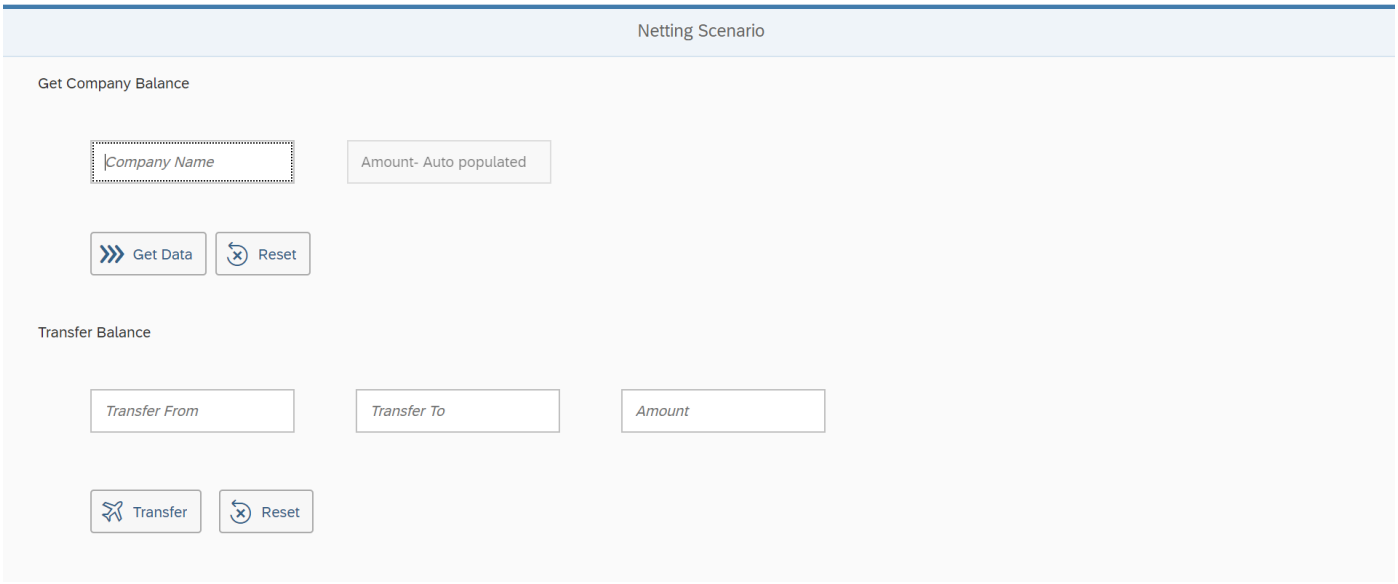
Screenshot 2: List of all applications in CF

5. Click on the application named service, it will display all the information about the specific application.



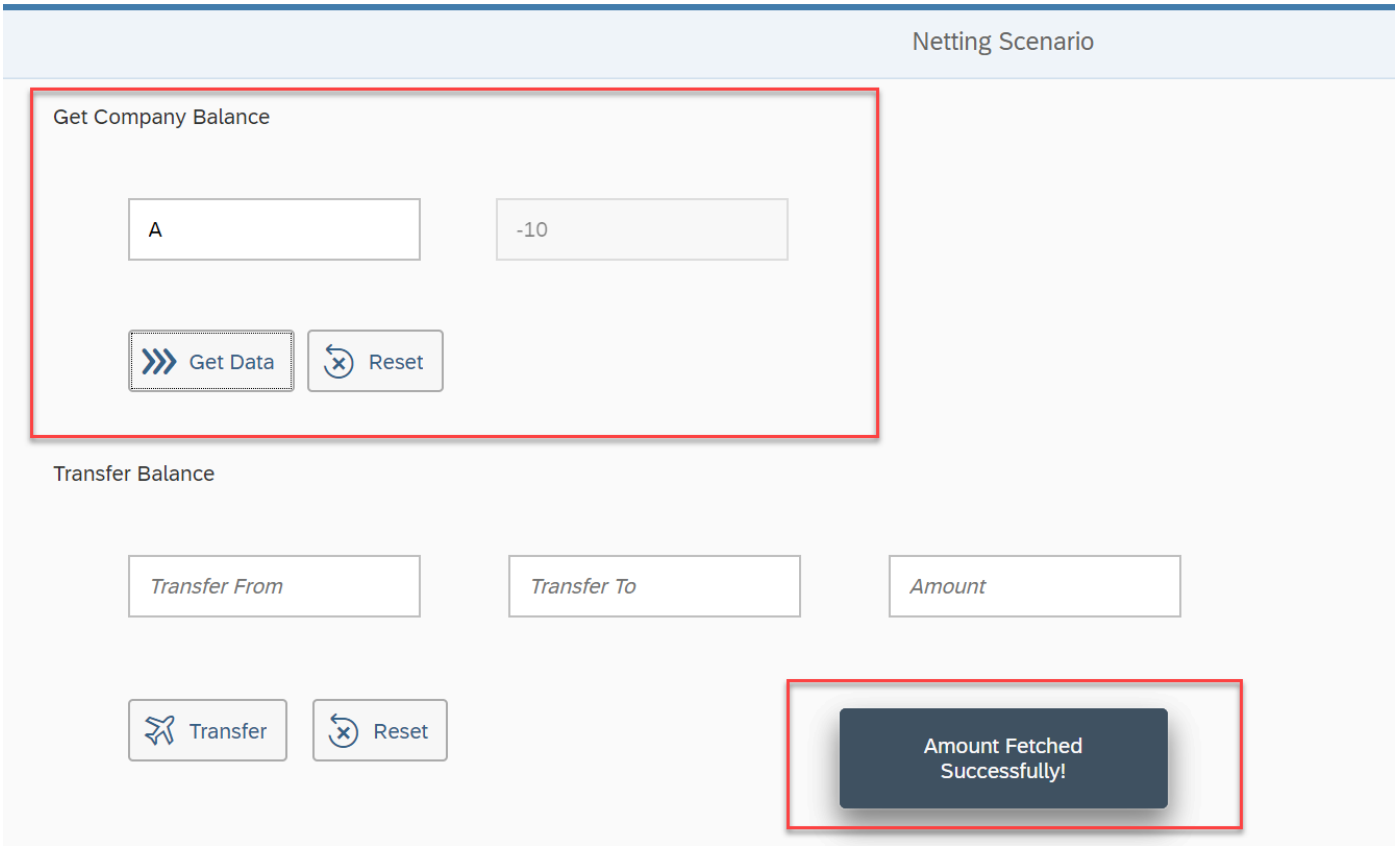
Screenshot 3: Details about the “service” named application

6. Click on the url as displayed in the screenshot 3. This will open up the index.html page of the UI which we embedded while integrating node and UI5 application as shown in screenshot 4.



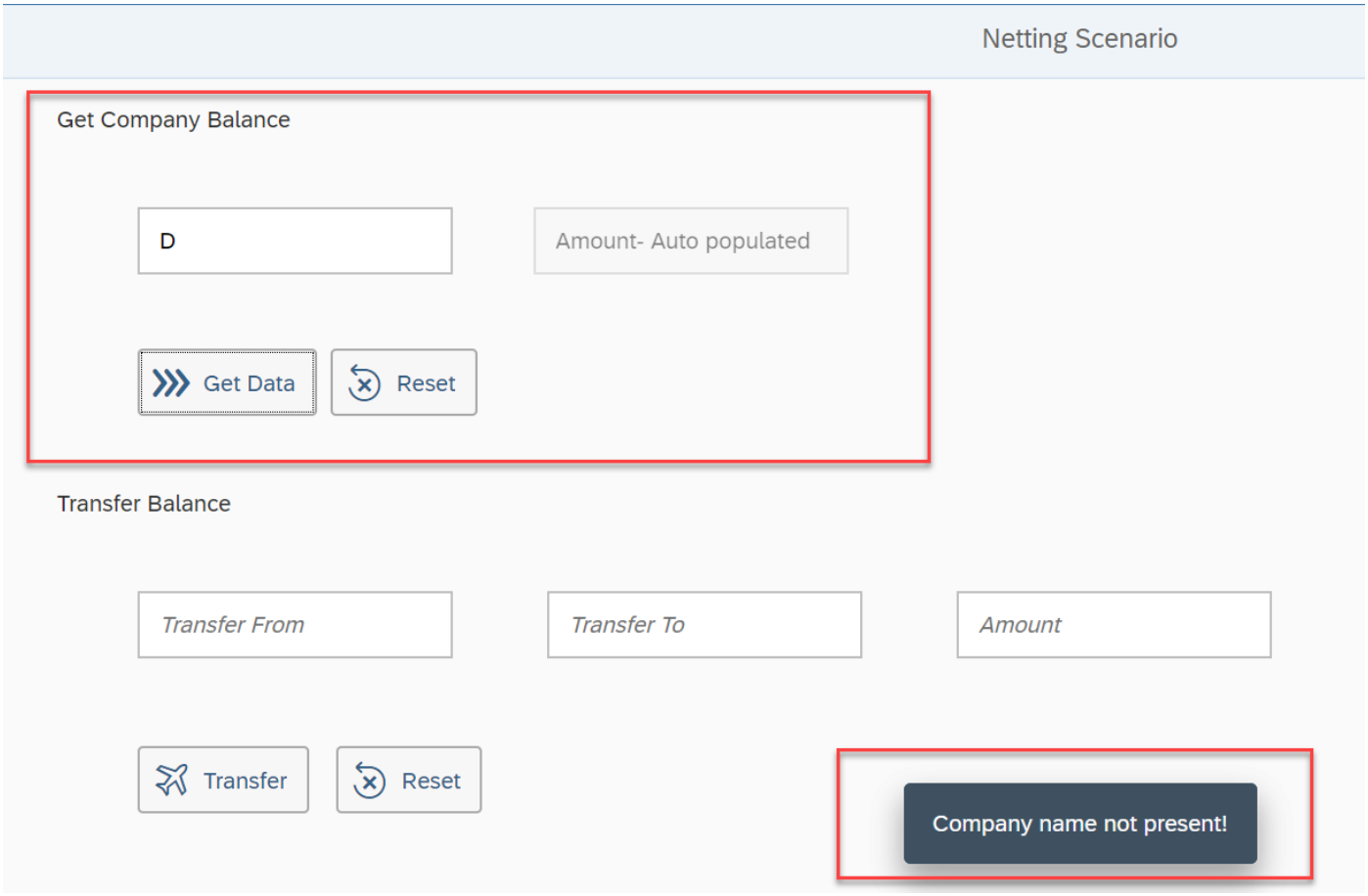
Screenshot 4: Netting Application index.html

7. In “Get Company Balance” enter value A, it will give corresponding amount or balance associated with Company A. Example is shown in the screenshot 5.



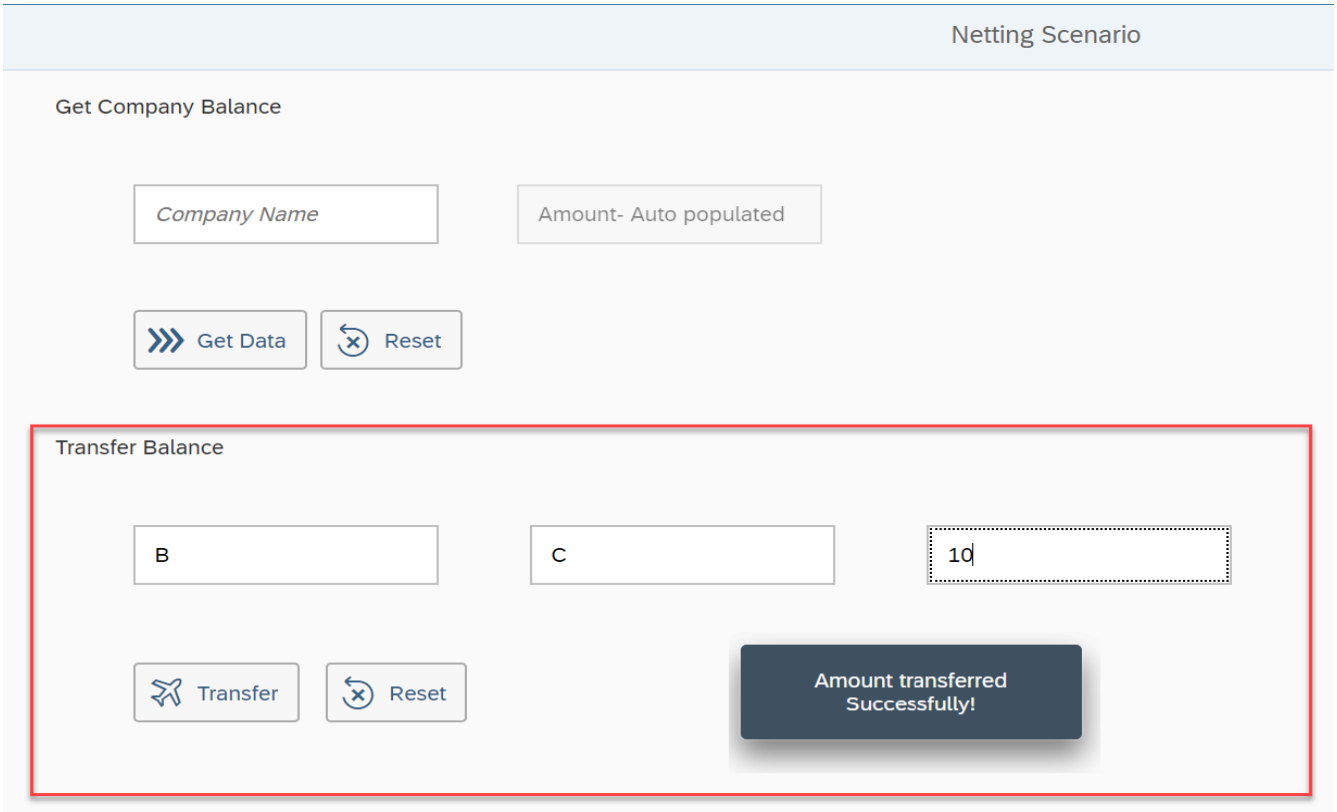
Screenshot 5: Balance for Company A

8. When we try to fetch a company name which is not present, we get the response as shown in screenshot 6.



Screenshot 6: Company name not present, No Balance fetched!

9. Transfer the Balance from company B to company C as shown in screenshot 7.



Screenshot 7: Amount transferred Successfully