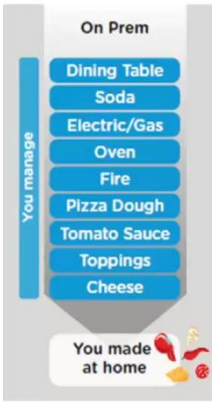
What is the difference between ON-PREMISE & CLOUD

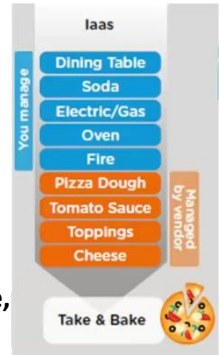
On-premise software is installed and runs on a company's own hardware infrastructure, and accessed through local host

whereas **cloud** software is installed and runs on the provider's servers, and accessed through a web browser or other interface.

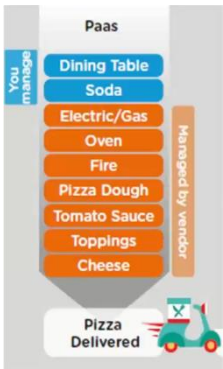
So to give you better example lets compare it to a pizza delivery. So I am very hungry today and definitely I am going to eat pizza today. So the Options I have



Option-1 : I will go to market I purchase everything raw materials which is used to prepare a pizza. And I will prepare a pizza by own. Now imagine in the world of software, If your company want to use software or build software, they need hardware, they need everything and everything is managed by your own company. That's exactly is **On-premise solution.** So right from managing of physical hardware, ram, computer, computing-resources, servers, networking, security, firewall, people, server room, cooling, development, code, infrastructure everything is your own company.



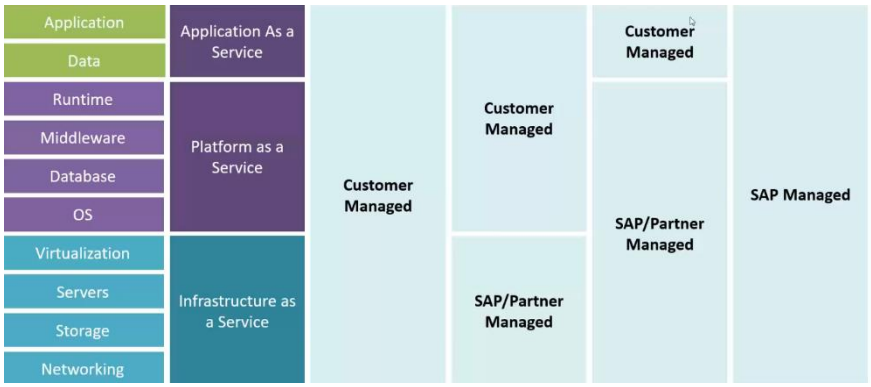
Option-2 : On the other side I would say, I don't have that much of time and I am very hungry and I would want some part of these as out-source. I would go back to market and say what ever infrastructure is required which include **cheese, Toppings, Tomato Sauce, Pizza Dough** all of these I will outsource, I will get it from market and perhaps I just to take that out and bake it and eat it. I get most of the materials from outside just need to bring it my oven and I cook it and I will eat it. So in this case I just need to manage **Fire, Oven, Electric/Gas, Soda, Dining Table.** Rest of the items I will outsource, So this is similar to a infrastructure as a service, where your company says I would want somebody else to own the hardware I don't want to manage **hardware** , **security, data-centre, database, storage, ram, and computing resources.** Let me outsource that to somebody. And the somebody is AWS, AZURE. So this technique is like **Infrastructure as a service.**



Option-3 : Then you say maybe I don't want to manage my fire cooking and oven in that case I will go with **Platform as a Service.** In this case what I would do is I would order pizza online and let them deliver at my home. In that case you are not cooking. That is again managed by the vendor who cooks it and use their own raw materials and get you the pizza at home delivered. And just you need to manage your **dining table, soda** to enjoy your pizza. That is platform as a service where the required platform to developed application is managed by the somebody else.
So here both **infrastructure** and **platform** are managed by the third party vendor.

Option-4 : And finally you decided I don't want anything to managed by myside. I just will go out to somebody and say I will sit there and I will eat pizza and I will comeback home.
I don't want to manage I just go, eat and come, That is called software as a service. Where everything or you can say entire software are managed by the vendor.

Type of cloud offerings



So what is cloud foundry.

Cloud Foundry is an **Open Source cloud platform as a service** (PaaS) on which developers can build, deploy, run and scale applications. **VMware** created Cloud Foundry, which is now part of **Pivotal Software**, whose parent company is Dell Technologies.

So in cloud foundry everything managed by the SAP [**Iaas, Paas --- Managed**] , you as a developer just need to work with **Application and Data.**

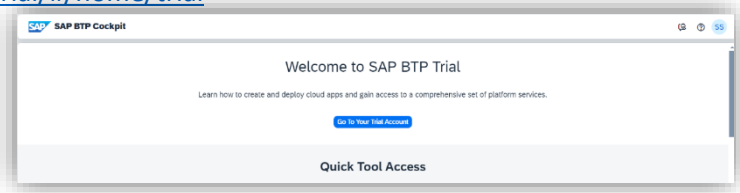
Setup Cloud Platform free trial account

go to : https://www.sap.com/products/technology-platform.html

click on create account button. And by filling the necessary details create account.

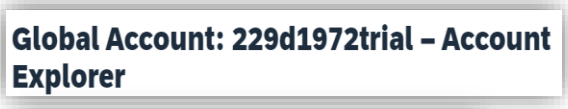
I have already an account : soumiksahacodedevpro@gmail.com password : Sapfiori@123

For login : https://account.hanatrial.ondemand.com/trial/#/home/trial

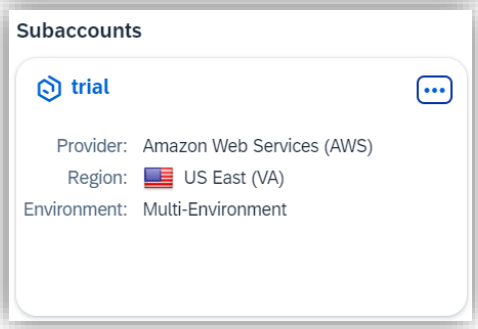


So when you setup your sap cloud foundry account. You will see a global account. A global account from sap point of view represents a customer for billing purpose. Suppose I am a **Deloitte or Accenture**. So I go to sap and say I want to use your platform and I want services to build application for our customers. So sap will charge the amount which need to pay based on my resources which I bought. So when I buy multiple resources for my organisation. And all the resources are given into the global account or assigned into the global account. And that we call it as **entitlement**. So after I got the all entitlements, now it is my decision how to distribute the resources into different departments, So to do that I need to have a sub-account. So a sub-account is per region. So lets say I am **Accenture.** And I have right now 2 departments, one is HR and another one is Finance**.** In finance department I want to give ABAP and hana. In the HR I will give Java, Node Js . So I have a flexibility to decide how I want to utilize my resource across multiple department. So I can create N number of subaccount inside the global account. And to create subaccount we need to choose the region. And based on the region it will automatically select the infrastructure provider. So depending on the region we will see the different services. And also we can create space. So space is like provide the access of the shared location for the department for application development.

Global Account and Sub Account

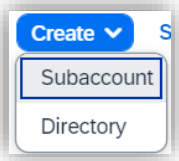


So as you can see here, this is my global account.

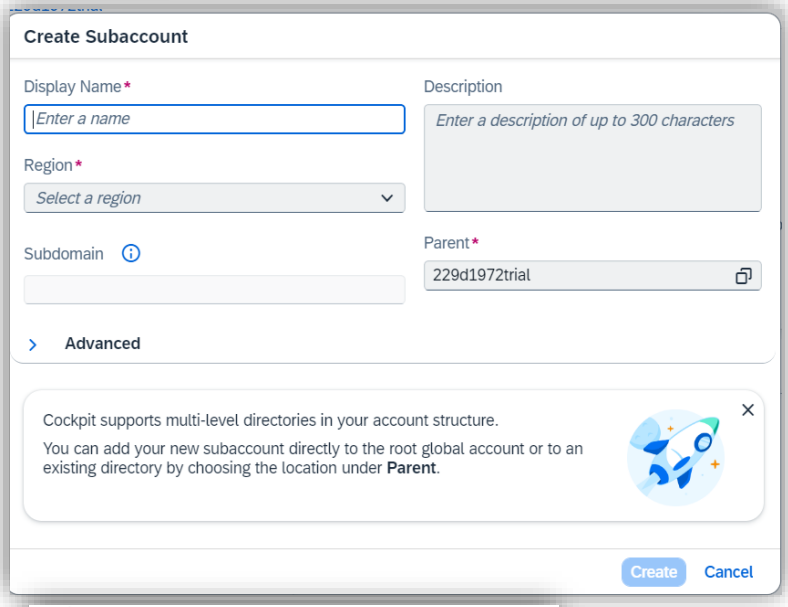


And this is my sub-account, which is inside my global account. And in this subaccount, the region is **US** and the infrastructure provider is AWS. Basically while creating thew subaccount we need to provide the region And based on the region it will automatically select the infrastructure provider. We can create **N** number of subaccount in a global account.

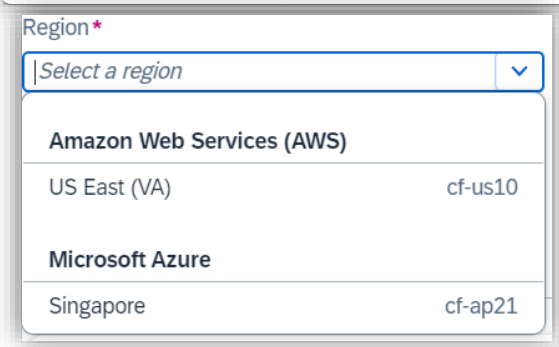
Create subaccount



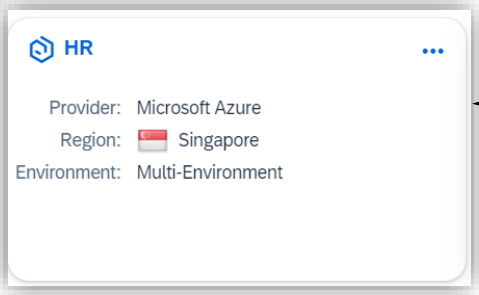
So for creating the subaccount we will click the **Create** dropdown and from there we will choose the **Subaccount.** And we will click on the Subaccount.



So at the time of creating the subaccount, We need to provide the some mandatory field like **Display Name** i.e. we need to give a name to our Subaccount. **Region** i.e. we need to give region to our subaccount. And **Description** is the option field. After fill the All mandatory filed we will click on create button and It will create the Subaccount.



So while selecting the region from the dropdown, there based on the region, it will select the different infrastructure provider.



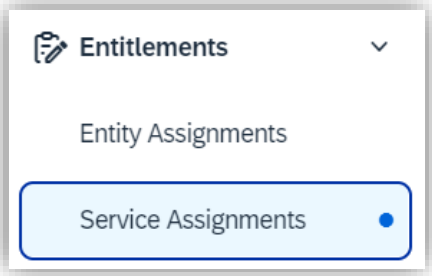
*So here as you can see I have created a subaccount named **HR** and I have choose the Singapore region, that's why It automatically choose the **Microsoft Azure** infrastructure provider*

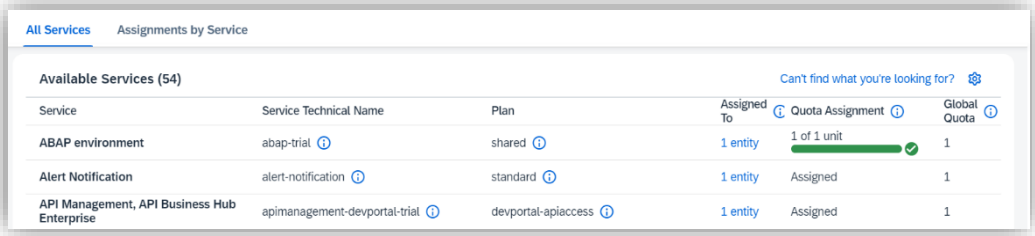
Entitlements

So all the services or resources are assigned into the global account, that we call it as entitlements. So in the **entitlements**, there you will see the **service assignments,** basically which are all the services are available that it will show in the service assignment. And the services will be vary based on different regions

Is service or resources is vary based on the region ?

Yes service or resources are vary based on the region.

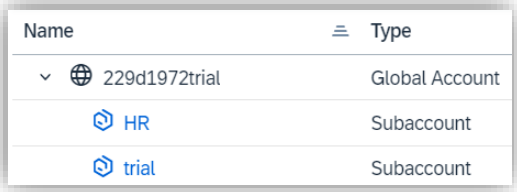


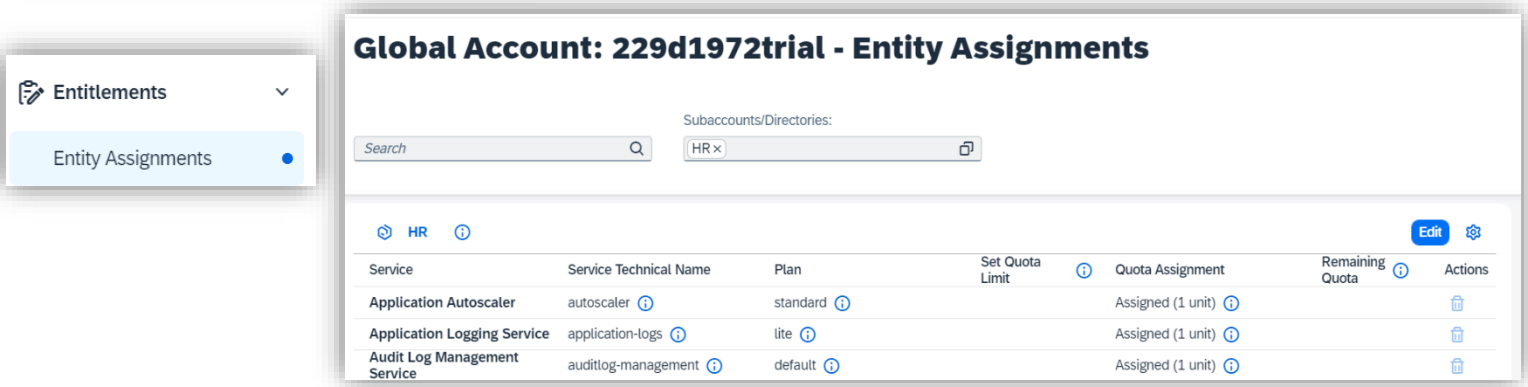


Entity assignment

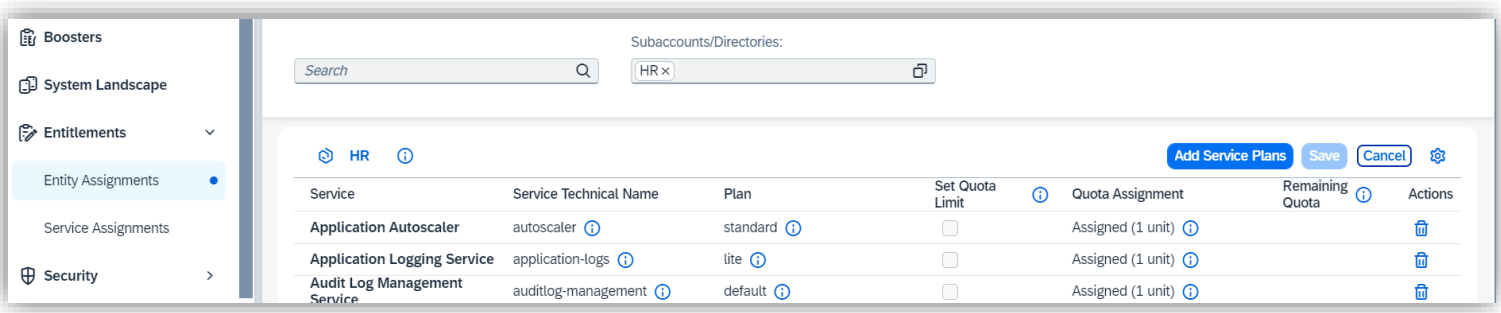
So in the **entitlements** section you will see the **Entity assignment**, basically there you can check how many services or resources are assigned to my subaccount. So remember the services or the resources are distributed from the main account to subaccount.

So I have total 2 subaccount name **HR, trial.** So I want to see in my HR subaccount what are all the services are assigned and I also want to see in my trial subaccount what are all the services are assigned, that we can see in the **Entity assignment** section.



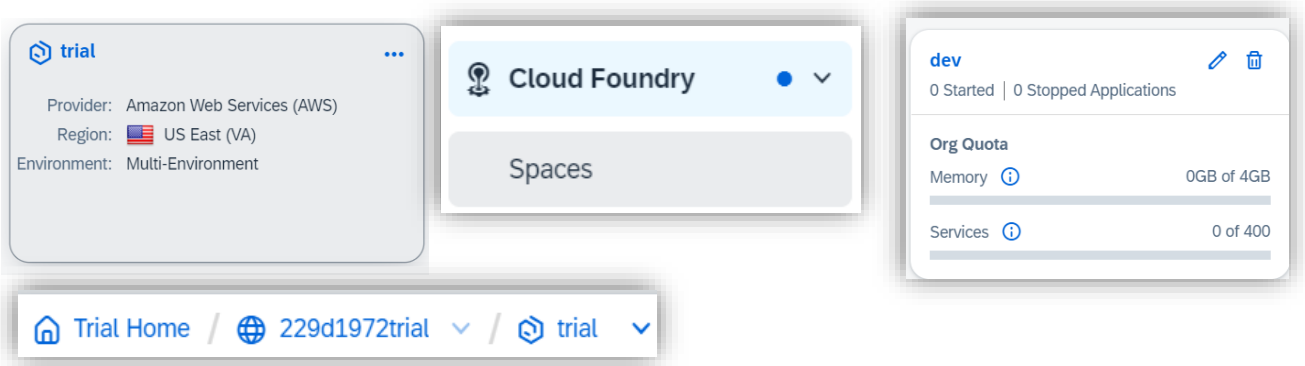


So when we have multiple custom subaccount, so what resource need to assign to the subaccount, that we can distribute by going to the entity assignments, by clicking on edit button.



So in this episode, we are going to go and deploy first fiori application in cloud foundry and will understand what is going to happen behind the scenes, when an application deploy in cloud foundry.

So when we create our first cloud foundry account, it automatically creates a dev space. If you want to see the dev space, which got created automatically, for that you need to go inside the sub-account by click on the sub-account, then it will show a **cloud foundry** folder inside that folder there is a file called **space**, if you click on that you will see the dev space.



So I had a subaccount named **trial,** I went inside the **trial.** And after that in the left side I saw a **could foundry** dropdown, I clicked on that and I saw file called **spaces** and after clicked on that **dev space** got open, which created automatically when we created cloud foundry account. Now we will allocate the space in the dev space, with the help of space quota.

What is space quota ?

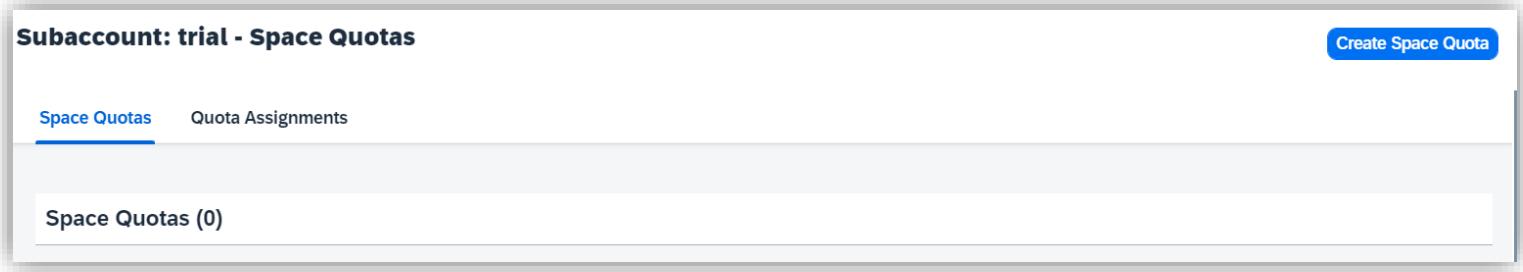
Space quota in SAP BTP (Business Technology Platform) refers to the allocated amount of storage space that a user or application is allowed to consume within a specific service or environment.

So now I will create a space quota for the dev-space. So when I initially go to the space quota, we see there is no space quote assign to the dev-space, that's why in the below picture **Space Quotas(0)**.

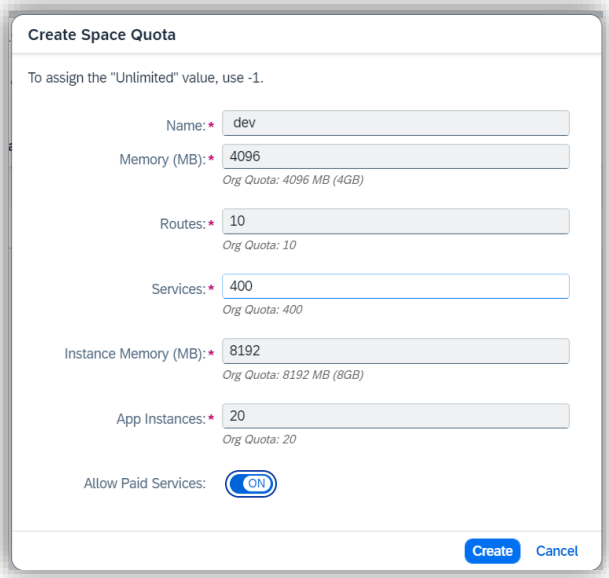
If there is no space-quota assigned to the dev-space what will happen ?

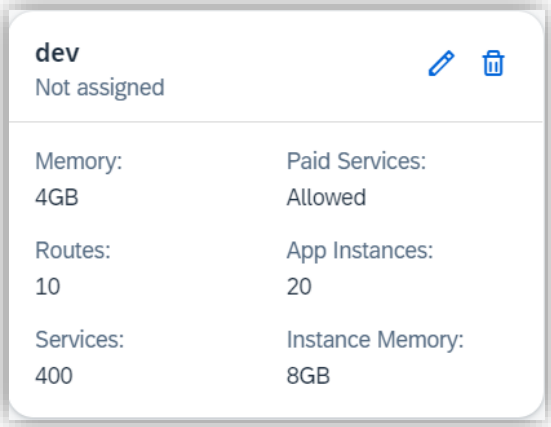
That means there is no memory allocated to the dev-space. And we can not deploy application to dev-space.

So to create the Space Quota I will click on the button Create Space Quota.



So Now I am creating the Space Quota services for the dev-space, which is part of the subaccount **trial.** After fill all the necessary details, I will click on Save button.

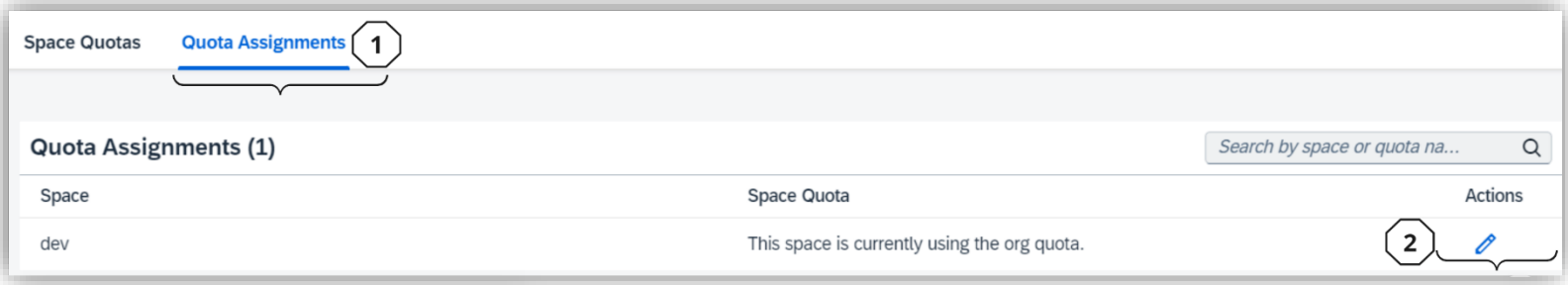


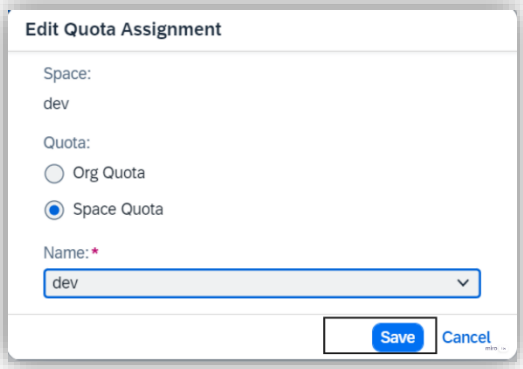


So here as you can see I have created a Space Quota. But this Space Quota is not assigned to the dev-space. That's why it is Showing Not Assigned.

NOTE Suppose you want to delete any Space Quota, which you created, that space quota can only be deleted, if it not assign to any dev-space. If it is already assign to dev-space, It can not be deleted

So now I will assign the Space Quota to the Dev Space. To assign that we will click on **Quota Assignment** Button And we will click on **Edit button** and after that one dialog box will open and we will select the **Space Quota** and then we will select the name of the Space Quota from the drop down. After select the name of the Space Quota We will click on Save button.

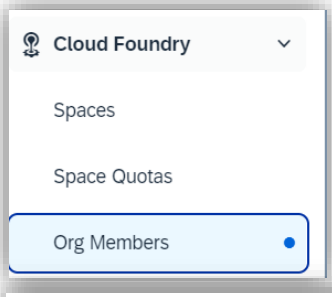


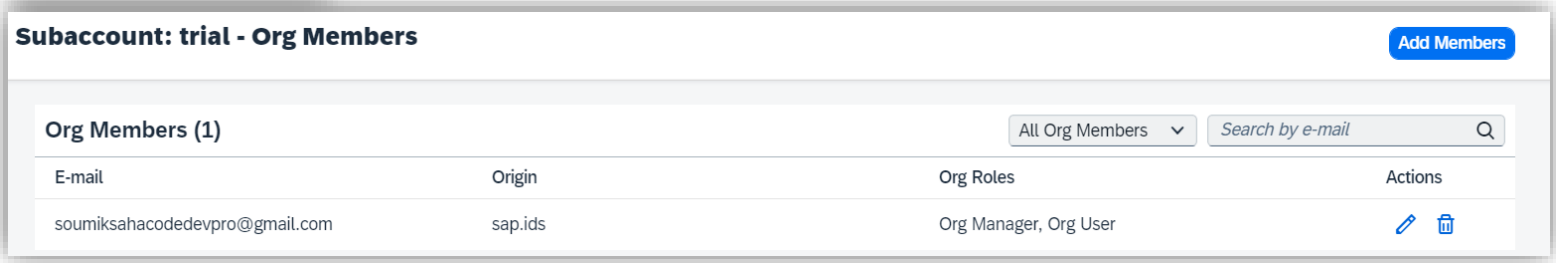


So once I assign the Space Quota to dev-space, that means the service or resources which is offered by the sap in the free trail, every thing I assigned to the dev-space.

NOTE Suppose you want to delete any Space Quota, which you created, that space quota can only be deleted, if it not assign to any dev-space. If it is already assign to dev-space, It can not be deleted

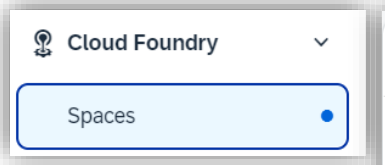
Org Members : So here we can add other members by clicking on the Add Members button and to add members we need to give the member email-id. So that user also can login my cloud foundry account and also can deploy application to my cloud foundry account.

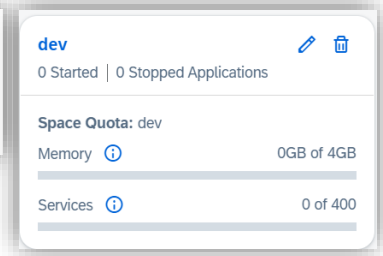


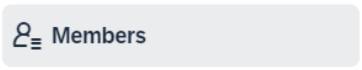


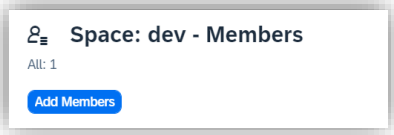
And now within the dev-space, who are all can deploy application that permission you can give by going inside the dev-space.

So to give permission in the dev-space first we will go inside the dev-space. And we will click on Members and we will add members by clicking on add members. So that members can be able to deploy applications within the dev-space.



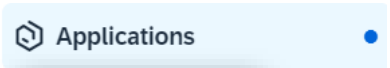


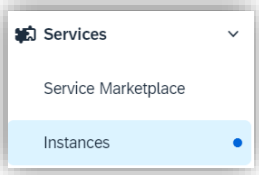




The application which you are deploying will show in the Application section.

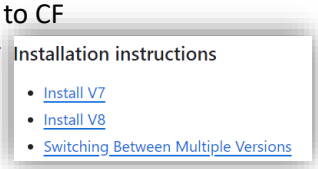
We can start or run the service, for that we need to go to instances section.

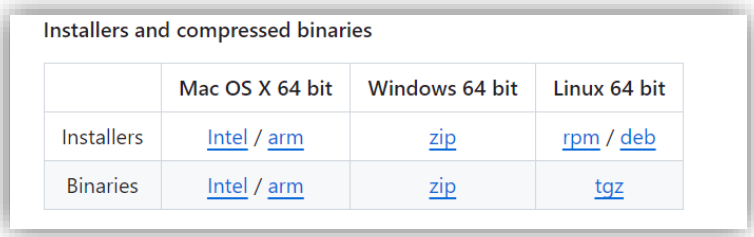




Tools required to develop applications :

1. Git hub
2. Git Steup ----- https://git-scm.com/
3. Cloud Foundry (Command Line Interface) – allows you to communicate/connect to CF directly from the command prompt. -------- So to install we will go and search for
   https://github.com/cloudfoundry/cli
   https://github.com/cloudfoundry/cli/wiki/V7-CLI-Installation-Guide
   So here I am going to install v7. So I will click on that v7





I will install the zip file of installer.

To check the version of the CF CLI, we will write the command **cf --version**

The runtime which cloud foundry supports : Python, C, .NET, Ruby, Java, Java Specific to SAP, Node JS, these are all the runtimes

How do I know what are all the runtime it supports ?

So for that you need to login the cloud foundry account locally through the command prompt. So in the command prompt we will type **cf login.** And after that it will ask for the endpoint. To get the endpoint you need to go to the cloud foundry account through the browser. And you need to go inside a subaccount by clicking on that subaccount. In my case there is a subaccount named **trial**, so I will go inside the **trail** subaccount. Once you go inside, you will get a **API endpoint**. That endpoint you will copy and paste it to the cmd. And after that it will ask username and password. So need to provide the cloud foundry credentials. After that you will successfully login to the cloud foundry account through the cmd. Below showed the steps, how to login to the cloud foundry through the cmd.

```
PS C:\Users\Soumik> cf login
API endpoint:
```


```
API endpoint: https://api.cf.us10-001.hana.ondemand.com

Email: soumiksahacodedevpro@gmail.com
Password:
```

```
PS C:\Users\Soumik> cf login
API endpoint: https://api.cf.us10-001.hana.ondemand.com

Email: soumiksahacodedevpro@gmail.com
Password:

Authenticating...
OK

Targeted org 229d1972trial.

Targeted space dev.

API endpoint:   https://api.cf.us10-001.hana.ondemand.com
API version:    3.158.0
user:           soumiksahacodedevpro@gmail.com
org:            229d1972trial
space:          dev
PS C:\Users\Soumik>
```

So thus we can login to the sap cloud foundry account from the command prompt with the help of **CF CLI.**

So now I want to see, what are all the available runtimes which is supported by the cloud foundry, So for that the command would be **cf buildpacks**

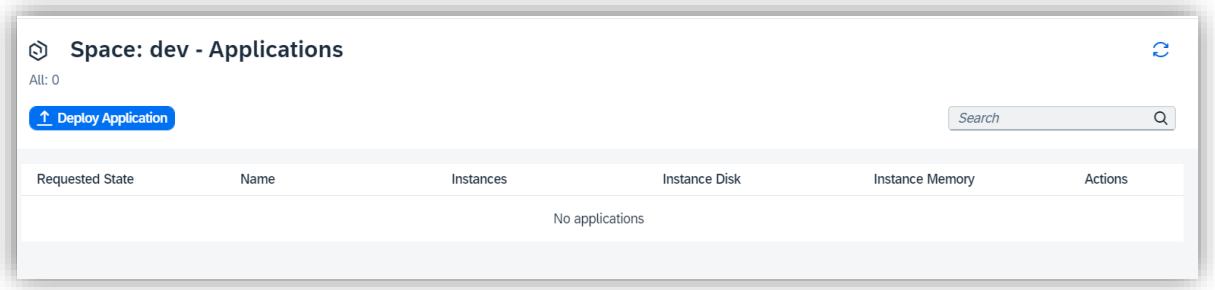
```
PS C:\Users\Soumik> cf buildpacks
Getting buildpacks as soumiksahacodedevpro@gmail.com...

position   name                                stack         enabled   locked   filename
1          staticfile_buildpack                cflinuxfs4    true      false    staticfile_buildpack-cached-cflinuxfs4-v1.6.9.zip
2          java_buildpack                      cflinuxfs4    true      false    java_buildpack-cached-cflinuxfs4-v4.66.0.zip
3          staticfile_buildpack                cflinuxfs3    true      false    staticfile_buildpack-cached-cflinuxfs3-v1.6.2.zip
4          java_buildpack                      cflinuxfs3    true      false    java_buildpack-cached-cflinuxfs3-v4.58.zip
5          nodejs_buildpack                    cflinuxfs4    true      false    nodejs_buildpack-cached-cflinuxfs4-v1.8.21.zip
6          go_buildpack                        cflinuxfs4    true      false    go_buildpack-cached-cflinuxfs4-v1.10.15.zip
7          python_buildpack                    cflinuxfs4    true      false    python_buildpack-cached-cflinuxfs4-v1.8.19.zip
8          php_buildpack                       cflinuxfs4    true      false    php_buildpack-cached-cflinuxfs4-v4.6.14.zip
9          nginx_buildpack                     cflinuxfs4    true      false    nginx_buildpack-cached-cflinuxfs4-v1.2.10.zip
10         sap_java_buildpack_1_86                           true      false    sap_java_buildpack-v1.86.0.zip
11         binary_buildpack                    cflinuxfs4    true      false    binary_buildpack-cached-cflinuxfs4-v1.1.9.zip
12         sap_java_buildpack                                true      false    sap_java_buildpack-v1.88.0.zip
13         sap_java_buildpack_1_88                           true      false    sap_java_buildpack-v1.88.0.zip
14         sap_java_buildpack_1_87                           true      false    sap_java_buildpack-v1.87.0.zip
15         nodejs_buildpack                    cflinuxfs3    true      false    nodejs_buildpack-cached-cflinuxfs3-v1.8.9.zip
16         sap_java_buildpack_jakarta                        true      false    sap_java_buildpack-v2.2.0.zip
17         sap_java_buildpack_jakarta_2_2                    true      false    sap_java_buildpack-v2.2.0.zip
18         go_buildpack                        cflinuxfs3    true      false    go_buildpack-cached-cflinuxfs3-v1.10.8.zip
19         python_buildpack                    cflinuxfs3    true      false    python_buildpack-cached-cflinuxfs3-v1.8.9.zip
20         php_buildpack                       cflinuxfs3    true      false    php_buildpack-cached-cflinuxfs3-v4.6.4.zip
21         nginx_buildpack                     cflinuxfs3    true      false    nginx_buildpack-cached-cflinuxfs3-v1.2.2.zip
22         binary_buildpack                    cflinuxfs3    true      false    binary_buildpack-cached-cflinuxfs3-v1.1.4.zip
23         zero_trust_sidecar_buildpack                      true      false    zero_trust_sidecar-v1.0.2.zip
24         zero_trust_sidecar_buildpack_1_1                  true      false    zero_trust_sidecar-v1.0.2.zip
25         zero_trust_sidecar_buildpack_1_0                  true      false    zero_trust_sidecar-v1.0.0.zip
PS C:\Users\Soumik>
```

I want to see the applications are running or not in the cloud foundry, what would be the command for that?

So the command would be the **cf apps**

```
PS C:\Users\Soumik> cf apps
Getting apps in org 229d1972trial / space dev as soumiksahacodedevpro@gmail.com...

No apps found
```

We can also check by going to the cloud foundry account in **Applications** section. To go to the Application file, we need to go inside the dev-space.



If you want to see all the CLI commands then you can visit the documentations : https://docs.cloudfoundry.org/cf-cli/

Also you can get all the cli commands, by enter the command in CMD **cf -help**

So what is important for me when I deploy the app to cloud foundry [NR]

At the time of deploy applications

1. how much memory you want me to allocate,
2. how much computing resources does this app needs .
3. What is the name of this app.
4. Is this app needs some extra services.
5. What also we need to tell which build-pack is needed to run the application.
6. And when you want to run this app, do you want to assign special your own name or do you want cloud foundry to assign some name and path to this application.

So this are the minimum things which cloud foundry will ask you. And all this information we maintain a file name **manifest.yml** file. So whenever you deploy application, cloud foundry CLI will look for this manifest.yml file. If it get this yml file then it will fetch the information from this file. And then it is going to deploy application on cloud foundry. And in the yml file also we need to tell which build-pack is needed, If we don't mention which build-pack is needed to run this application, then cloud foundry will automatically detect.

So on the above points all the optional, only the **3rd** one is mandatory.

```yaml
---
applications:
  - name: soumik-go-app
    memory: 256M
    instances: 1
    random-route: true
```

So here we have mentioned the application name, we have assigned the memory to the application, and also we have mentioned the **random-route** true that means I don't want to decide the URL. Let the cloud foundry decided what will be the URL of the application. If I want to decide the URL, then I have to decide the route and I have to decide the route name.

So what are all the properties need to pass in the manifest.yml file how shall I get to know? [NR]

For that we will visit the documentation of the manifest.yml file
https://docs.cloudfoundry.org/deploying/index.html

I am deploying the **go application**. So I have 3 files with me that is **go.mod**, **main.go** & **manifest.yml** file so I have also mentioned the name of the application in the manifest.yml file. So now I want to push my application with the command **cf push.** But when I will do the push it will look for the manifest file. So in the command prompt I need to go to that folder where this **manifest file** is present. And from that location, we need to do **cf push.** But while doing push if it don't get the manifest file it will throw an error.

So when I will push the application with the command cf push, internally it will deploy the application to cloud foundry. And it will identify the runtime which is used in the application. And once it identify the runtime, then it will automatically download that buildpacks and deploy the application.

Note : There are 5 applications and suppose Java runtime fails. Then it will not impact go application. Because go application is running separately I different container. So each application has their own environment.

Deploying a Go application using cloud foundry [NR]

1. So first to deploy the go application, we need to download the go file and will keep it in a folder [**C:\SAP BTP\projects\test-app]**
2. Command prompt go to that folder location and run **cf push.**
3. Then it will automatically deploy the application and also start running the application

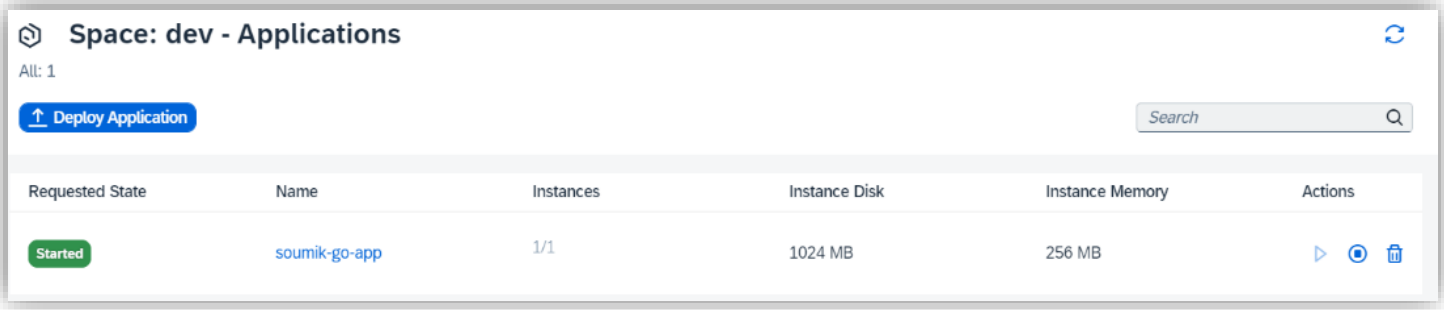
```
name:                soumik-go-app
requested state:     started
isolation segment:   trial
routes:              soumik-go-app-accountable-camel-av.cfapps.us10-001.hana.ondemand.com
last uploaded:       Sun 10 Mar 08:51:36 IST 2024
stack:               cflinuxfs4
buildpacks:
isolation segment:   trial
       name           version   detect output   buildpack name
       go_buildpack   1.10.15   go              go

type:            web
sidecars:
instances:       1/1
memory usage:    256M
start command:   test-app
     state     since                  cpu    memory   disk     details
#0   running   2024-03-10T03:21:45Z   0.0%   0 of 0   0 of 0
PS C:\SAP BTP\projects\test-app>
```

So here as you can see application is running. The URL which is used to open the application that will be mentioned in the **routes**

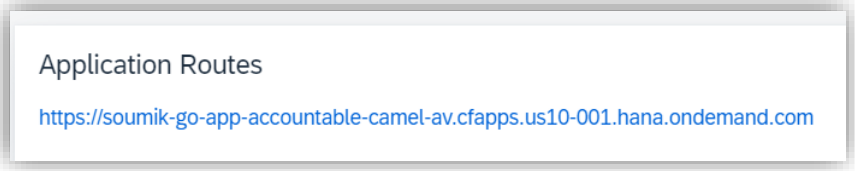
4. To verify that whether application deploy or not, we will use the command : **cf apps**

```
PS C:\SAP BTP\projects\test-app> cf apps
Getting apps in org 229d1972trial / space dev as soumiksahacodedevpro@gmail.com...

name           requested state   processes   routes
soumik-go-app   started           web:1/1     soumik-go-app-accountable-camel-av.cfapps.us10-001.hana.ondemand.com
```

or manually by going to application we can check.

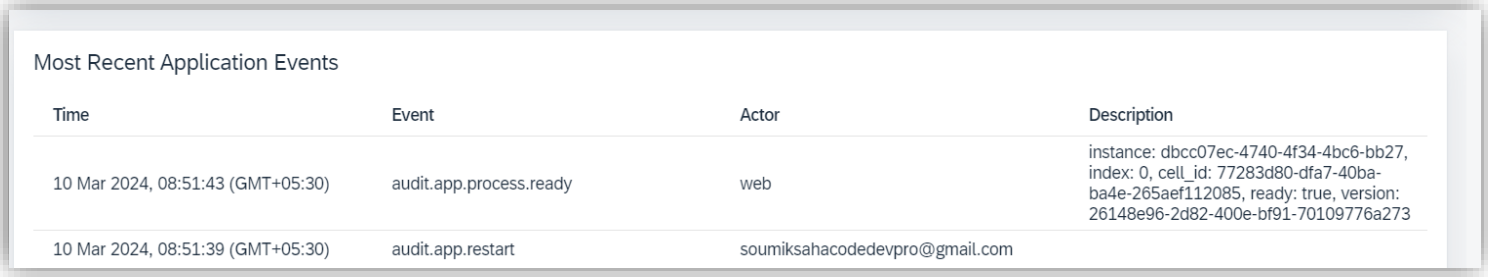


Also we can get the application routes by going inside the application



We can also get the log of the application, by using the command **cf logs {appName} --recent**

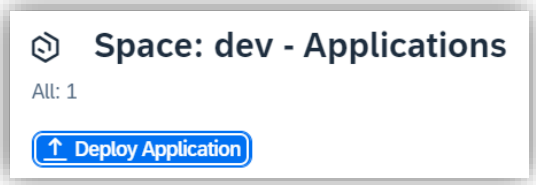
Or manually by going inside the application, you can check the logs.

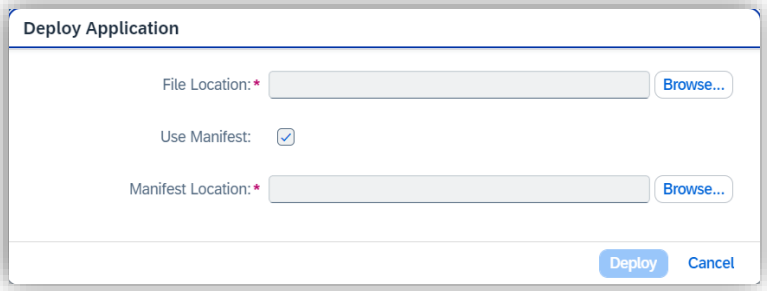


Is there any option to deploy application from desktop, without using cf cli ?

Yes there is a option where you can do directly from desktop.

So first go to the application and click on **Deploy Application.** Then give the necessary file, which it will ask, thus you can deploy application.





So what happens behind the scene, when we deploy the application [NR]

So for the deploy the application, when we run the command **cf push** then the cloud foundry Cli first check my computer folder and it will look for the **manifest.yml** file. And after that it reads all the properties from the yml file. Then it send a signal to cloud foundry. And in the cloud foundry there is something called cloud controller. The cloud controller listening to the request which are coming related to deployment of an application. So it picks up the work and from the cloud controller the work is pushed to Diego Core. So Diego Core it is a engine of cloud foundry, which is able to understand and run application. In Diego core there is something called Diego Bulletin board, which is a API layer, to take work from cloud controller. And This work is going to give Auctioneer. Auctioneer is like a dispatcher it going to get all the request. So this is the brain of the Diego core. So there are 2 main function of auctioneer. First it get all the request which come for the deployment. And then it checks which is the suitable build pack for this application, if you don't provide that in the manifest file. After that this work is given to a cell, where the execution of the work is happen. A cell is consist of something called garden, and garden consist of something called container. The actual execution happen inside garden of a cell. In different container different applications are running. A cell also have component called rap, It is also known as cell rap. So this rap will control and tell the other cell and Auctioneer what is running inside the cell. And also we have database, which cloud

foundry use by itself to track all the information about the internal process. And also we have something called CF Router, takes all the request which comes from the client. So whenever the user access the final application, first it comes to cloud foundry router. It routes the application to correct cells. That's how it works behind the scenes, when you deploy applications.

Scale up application instances

So there is 1 instance of my application currently running, so I can scale up and increase the application instances. So the syntax of the command is **cf scale {appName} -i {number of instances}**

So here how many instances you want that you need to give. I had 1 instances and I wanted 3 that's why I have given **cf scale soumik-go-app -i 3,** and here i refer to instances

```
C:\SAP BTP\projects\test-app>cf scale soumik-go-app -i 3
```

```
name           requested state   processes   routes
soumik-go-app  started           web:3/3     soumik-go-app-accountable-camel-av.cfapps.us10-001.hana.ondemand.com
```

Scale up Application memory.

So there is 256M memory in my application and I want to scale up the memory. That means I will increase the memory size. So for that the syntax of the command is
cf scale {appName} -m {memory size} , here m is for the memory and size we need to provide.
So I have 256M memory and I need 512M memory for that I will write the command
cf scale soumik-go-app -m 512M. So for increase the memory, service will restart then it will increase.

```
C:\SAP BTP\projects\test-app>cf scale soumik-go-app -m 512M
```

We can also delete the application

Command : **cf delete {appName}**

So when do we increase the instances of the Application ?

I developed an app and suppose 100 users are using my application. So suddenly my application tie up with a big company and users increase from 100 to 5000. So now in this situation my application can not handle this many request which will come from the user. So it like a ticket counter, so when we go to airport or railway station if there are only one ticket counter then the queue size will be big and people need to wait for the long time. But if there multiple counter runs parallelly then the queue size will be low and don't need to wait for long time. So to handle the 5000 user request we will increase the application instances.

What is Node JS ?

Node JS is an open-source, cross platform, JavaScript runtime environment used for executing JavaScript code outside web browser.

So when the company started development the web development, they used to use a programming language called JavaScript. So JavaScript was language which is only supported by the browser. So then people came up with an idea, can we use JavaScript as server side development, So that we no need to learn 2 different languages one for web-development and another for server side development.

If we talk about the client server architecture. So we have a browser or mobile device which is acts like a client. And this client sends a request to the server and the server send back the response to the client. Son in the server side we have the programming languages like Java, ABAP. On the client side people used to build application using JavaScript programming language. Now to execute JavaScript we don't need separate engines, we execute the JavaScript directly in the browser engine. So a developer what he had done is he developed a open source cross platform JavaScript runtime environment, which is named as node JS. So that means if you know JavaScript to build the web-application, you can run also JavaScript on the server side. So the great advantage here is as a developer you don't need to learn 2 programming languages. So node JS is the runtime environment to run the JavaScript on the server side.

The advantage of using the node JS as a server side development

1. It is open source,
2. It is cross platform, it works on any operating system
3. It is a asynchronous programming language, which make the execution very fast

What Node JS Capable of

1. Node JS can build application logic
2. It is used to build web server
3. Used to send emails
4. Do database lookups
5. We can so automations
6. We can send outputs
7. We can host web pages

And the Programming language, which Node JS is use is JavaScript.

Development Tool ?

So we can use BAS -- we can use it to built node JS application.
Also we can use VS code, which is free. The Vs Code give thousands of free plugins which we can install and make the developer life easy.

So now I will start the node JS development with vs code

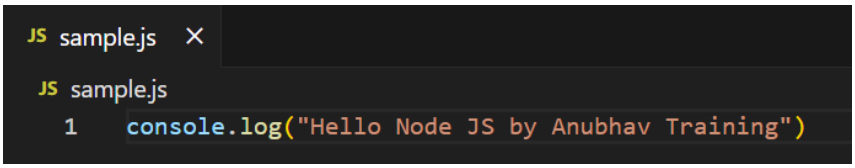
1. Install the Vs code, if you don't have. So I installed Vs Code and created a workspace name **BasicNodeJs [C:\SAP BTP\BasicNodeJs]** So will start node Js development in this workspace.

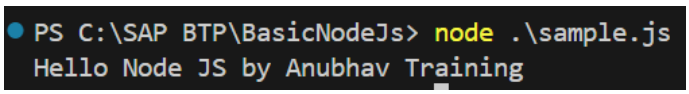
2. Install node js : https://nodejs.org/en Verify the download of the node js : node -v

Funda fox

1. JavaScript is a case sensitive language
2. Every statement ends with semicolon
3. To run any node js file we need to run the command **node {filename}**
4. Every variable or function follows JavaScript naming convention (camel-case). The first letter is small and the next consecutive word, first letter is capital.
   Iloveindia === iLoveIndia mylifeiscool === myLifeIsCool.

Print a hello world program in Node js





So execute the program we need to open the terminal and write the command **node sample.js** **[**So here sample.js is the filename, which I executed**].**

What is variable

A variable is a container that stores a value. Variable can be declared with **var, let, const** keyword.

The value of the JavaScript variable can be changed during the execution of program [**only for let & var**], below example is given

Example :

```
1  console.log("Storing value to the variable")
2
3  let a = "Soumik"  // a  contains Soummik
4  console.log(a)
5  a = "Saha"
6  // Java script allows to change the variable type at the runtime.
7  console.log(a)
8
```

```
Storing value to the variable
Soumik
Saha
```

var vs let vs const in JavaScript

- var is a globally scoped while let and const are block scoped

```
console.log("Var and Let and Constant")

var a = 100
console.log(a)

// Var are globally scooped
console.log(a)

{
  var a = "Soumik"
  console.log(a)
}
console.log(a)
```

```
Var and Let and Constant
100
100
Soumik
Soumik
```

So in the program as you can see we declared **"a"** variable a which value is declared **100**, and inside the block we again declared **"a"** variable which value is **"Soumik".** So if we come out of the block and again print the value of **"a"** we will get the value as "**Soumik**", that's why **var** is globally scoped.

```
let b = 100
console.log(b)

{
  let b = "It is blocked scoope."
  console.log(b)
}
console.log(b)
```

```
100
It is blocked scoope.
100
```

So in the program as you can see we declared **"b"** variable a which value is declared **100**, and inside the block we again declared **"b"** variable which value is **"It is blocked scoope".** So if we come out of the block and again print the value of **"b"** we will get the value as **100**, that's why **let** is blocked scoped.

```
const author = "Soumik"
console.log(author)

{
  const author = "It is blocked scoope."
  console.log(author)
}

console.log(author)
```

```
Soumik
It is blocked scoope.
Soumik
```

So in the program as you can see we declared **"author"** variable a which value is declared **"Soumik"**, and inside the block we again declared **"author"** variable which value is **"It is blocked scoope".** So if we come out of the block and again print the value of **"author"** we will get the value as **"Soumik"**, that's why **const** is blocked scoped.

- **var** can be updated and re-declared within its scope, **let** can be updated but not re-declared. **const** can neither be updated nor be re-declared.

```
var a = 100
console.log(a)

var a = 200  // Var can be re-declared and updated within its scope and Var are globally scooped
console.log(a)
```

```
const author = "Soumik"
console.log(a)
// const author = "Soumik"  Can not re-declared.

// author = "Saha" Can not re-assign the constant value to variable as it is constant.
```

```
let b = 100
// let b = 200 ---> let can not be re-declared.
console.log(b)

b = 200  // let value only can be updated
```

- **const** must be initialize the value during declaration unlike **var** and **let.**

```
//const Name  const must be initialized during declaration.

var Name // Var can be initialized during declaration.

let Name1 // let can be initialized during declaration.
```

Use a typeof operator to find the data type of the variable.

```
let num = 10
console.log(typeof num)

let fName = "Soumik Saha"
console.log(typeof fName)
```

```
number
string
```

Convert the String to a Number

Number.parseInt("Enter the String")

```
1  let age = "25"
2  console.log(typeof(age)) //When it was string it was showing string
3  age = Number.parseInt(age)
4  console.log(typeof(age)) //Now it is showing number
```

---------->

```
string
number
```

Primitive data type and Object

Total 7 primitive data type are there in JavaScript **– Null, Number, Boolean, BigInt, String, Symbol, Undefined**

```
// NN BB SS U. [Shortcut for primitive data type]
// N = number, null
// B = boolean, big-int
// S = String, Symbol
// U = undefined.

let a = null;
let b = 123;

let c = false;
let d = BigInt("123456789")

let e = "Soumik Saha";
let f = Symbol(" I am a symbol");

let g = undefined;
let h; // This also we can call it as undefined as we did not defined anything

console.log(a, b, c, d, e, f, g, h)
console.log(typeof a, typeof b, typeof c, typeof d, typeof e, typeof f, typeof g, typeof
h)
```

So this is the example of all primitive data type. We can Also get the type of the variable with the help of **typeof** operator

```
null 123 false 123456789n Soumik Saha Symbol( I am a symbol) undefined undefined
object number boolean bigint string symbol undefined undefined
```

Object are the non-primitive data type in JavaScript.

```
// Non-primitive data type is ---> object.

const Student_Data = {
  "Name": "Soumik",
  "Class": 12,
  "Roll_number": 2,
  "Section": "B"


}
console.log(Student_Data.Name) // this way we can fetch the value from object. "."
console.log(Student_Data["Name"]) // this way also we can fetch the value from object,
with the help of "[]"
console.log(Student_Data["Class"])
```

```
Soumik
Soumik
12
```

JavaScript Arrays

Arrays are variable which can hold more than one value. And values can be similar or different type.

```
//can be similar data types
const fruits =["banana" , "apple" , "orange" , "grapes"]

//can be different data types
const a1 = [7 , "harry" , false]
```

Accessing the array values : we can access the array values with the help of index number.

```
const animals = ["cat" , "dog" , "cow"]

console.log(animals[0])
console.log(animals[1])
console.log(animals[2])
```

```
cat
dog
cow
```

Finding the length of the array : We can find the length of an array using **length** property

```
const books = ["maths" , "physics" , "chemistry" , "biology"]
console.log(books.length)
```

```
4
```

pop() : Removes last element from an array. And also this method return the element which got removed.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let popedElement = numbers.pop();
console.log(popedElement); // pop method return
console.log(numbers); // after remove the element the updated array
```

```
10
[
  1, 2, 3, 4, 5,
  6, 7, 8, 9
]
```

push() : Add a new element at the end of an array. And also this method return the new array length.

```
const singers = ["Arijit", "Shreya", "Amit", "Neha"];

let pushedElement = singers.push("Ritika");

console.log(pushedElement); // push method return
console.log(singers); // after add the element the updated array
```

```
5
[ 'Arijit', 'Shreya', 'Amit', 'Neha', 'Ritika' ]
```

shift : Removes first element from an array. This method return the element which got removed

```
const player = ["Rohit", "Virat", "Dhoni", "Rahul"];

let shiftedElement = player.shift();

console.log(shiftedElement); // shift method return
console.log(player); // after remove the element the updated array
```

```
Rohit
[ 'Virat', 'Dhoni', 'Rahul' ]
```

unshit : Add a new element to the first on an array. This method returns the new array length.

```
const company = ["Google", "Microsoft", "Apple", "IBM"];

let unshiftedElement = company.unshift("Amazon");

console.log(unshiftedElement); // unshift method return
console.log(company); // after add the element the updated array
```

```
5
[ 'Amazon', 'Google', 'Microsoft', 'Apple', 'IBM' ]
```

splice in array : Splice method add/remove array element. It changes the original array.

Syntax of Splice : `array.splice(index, howmany, item1, ....., itemX)`

index	Required. The index (position) to add or remove items.
How many	Optional. Number of items to be removed.
item1, ..., itemX	Optional. New elements(s) to be added.

Add element in the array with the help of Splice

```
let number3 = [51,222,4443,94,85,11116,87,98]

number3.splice(2,0,1021,1022,1023) // it add the element in the array.

console.log(number3)
```

```
[
    51,  222,  1021,
  1022, 1023,  4443,
    94,   85, 11116,
    87,   98
]
```

Delete element in the array with the help of splice.

```
let number4 = [51,222,4443,94,85]
number4.splice(2,2) // it delete the element in the array.

console.log(number4)
```

```
[ 51, 222, 85 ]
```

Delete element and add element in the array with the help of splice.

```
let number5 = [51,222,4443,94,85]

number5.splice(2,2,1021,1022,1023) // it delete the element and add element
console.log(number5)
```

```
[ 51, 222, 1021, 1022, 1023, 85 ]
```

Looping through an array

Array can be looped through using the classical JavaScript For loop or through some other methods

Array can be looped through for loop :

```
var cricketer = ["Rohit", "Virat", "Dhoni", "Sachin", "Rahul"]

for (var i = 0; i < cricketer.length; i++)
  {
    console.log(cricketer[i])
  }
```

Array can be looped through for each loop

Syntax of ForEach loop :

```
array.forEach(function(currentValue, index, arr), thisValue)
```

currentValue	Required. The value of the current element.
index	Optional. The index of the current element.
arr	Optional. The array of the current element.

```
var singer = ["Arijit", "Neha", "Amit", "Sonu", "Kishore"];

singer.forEach((element, index, array) => {

  console.log(element, index, array);
});
```

```
Arijit 0 [ 'Arijit', 'Neha', 'Amit', 'Sonu', 'Kishore' ]
Neha 1 [ 'Arijit', 'Neha', 'Amit', 'Sonu', 'Kishore' ]
Amit 2 [ 'Arijit', 'Neha', 'Amit', 'Sonu', 'Kishore' ]
Sonu 3 [ 'Arijit', 'Neha', 'Amit', 'Sonu', 'Kishore' ]
Kishore 4 [ 'Arijit', 'Neha', 'Amit', 'Sonu', 'Kishore' ]
```

Array can be looped through for of loop : Basically for of loop, loops through the value of an iterate able object

```
var dancer = ["Kishore", "Arijit", "Neha", "Amit", "Sonu"];

for (let key of dancer)
  {
    console.log(key);
  }
```

```
Kishore
Arijit
Neha
Amit
Sonu
```

Array can be loop through for in loop :

```
var actor = ["Kishore", "Arijit", "Neha", "Amit", "Sonu"];

for (let key in actor)
  {
    console.log(actor[key])
  }
```

```
Kishore
Arijit
Neha
Amit
Sonu
```

Array loop through the map function : JavaScript map returns a new array element.

Syntax :

```
array.map(function(currentValue, index, arr), thisValue)
```

```
const alphabet=["A", "B", "C", "D", "E", "F", "G", "H", "I"]

alphabet.map((element, index, array)=>{

  console.log(element)
})
```

```
A
B
C
D
E
F
G
H
I
```

```
// write a map function which returns an array
let arr = [1,2,3,4,5];
let value =arr.map((Element,Array,Index)=>
  {
    return Element;
  })
console.log(value)
```

```
[ 1, 2, 3, 4, 5 ]
```

What is the difference between map function and for each loop

So, the main difference between **map** and **forEach** is that map returns a array, while forEach does not return anything and only modifies the original array.

How to modify array with the help of for each loop

```
const numbers=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

 numbers.forEach((element, index, array)=>{
 array[index]=element*element
})
console.log(numbers)
```

```
[
   1,  4,  9, 16,  25,
  36, 49, 64, 81, 100
]
```

Here as you can see in the array index we are storing the square of each element. Same also can be done with the help of Map function and Filter.

filter method : JavaScript Array filter() Method is used to create a new array from a given array consisting of only those elements from the given array that satisfy a condition. The filter method **return** array.

```
array.filter(function(currentValue, index, arr), thisValue)
```

```
const Numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const evenNumbers = Numbers.filter((number) =>
  {
    return number % 2 === 0;
  })

console.log(evenNumbers)
```

```
[ 2, 4, 6, 8 ]
```

We can also use filter with this way also:

```
const Numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const evenNumbers = Numbers.filter((checkCondition));

function checkCondition(number)
  {
     return number % 2 === 0;
  }

console.log(evenNumbers)
```

```
[ 2, 4, 6, 8 ]
```

Objects

Objects are variables too. But objects can contain many values. Object values are written as **name : value** pairs

Will show an object, do all the operations with the object.

```
//creating a employee object

var employee = {
  name: "John",
  age: 30,
  designation: "Software Engineer",
  salary: 50000
}

//Print the object
console.log(employee);

//Access the properties of the object
console.log("Name:", employee.name);

//Change the properrty of the object
employee.age = 31;
console.log("Age:", employee.age);

//Loop through the properties of the object
for (let key in employee)
  {
    console.log(key + ":", employee[key]);
  }

//Delete a property from the object
delete employee.salary;
console.log(employee);
```

```
{
  name: 'John',
  age: 30,
  designation: 'Software Engineer',
  salary: 50000
}
Name: John
Age: 31
name: John
age: 31
designation: Software Engineer
salary: 50000
{ name: 'John', age: 31, designation: 'Software Engineer' }
```

JSON

Basically it is JavaScript Object Notation.

JSON Syntax Rules

Data is in name/value pairs `"firstName":"John"`

Data is separated by commas

Curly braces hold objects

Square brackets hold arrays

Create Json Object

```
//Create a JSON Object
var jsonObject= {
  "name": "Ram",
  "age": "25",
  "hobbies": ["coding", "reading"]
}
console.log(jsonObject)
```

Create a Json Array

```
//create a JSON array
var jsonArray = [
  {
    "id": 1,
    "name": "John",
    "age": 25
  },
  {
    "id": 2,
    "name": "Jane",
    "age": 30
  }
]
console.log(jsonArray)
```

Convert the JSON to String

```
//create a JSON array
var jsonArray = [
  {
    "id": 1,
    "name": "John",
    "age": 25
  },
  {
    "id": 2,
    "name": "Jane",
    "age": 30
  }
]
console.log(JSON.stringify(jsonArray))
```

So we can convert the **Json** to **String.**
JSON.stringify(Json file name)

Convert the String to JSON

```
//Convert tha String to JSON
var jsonString = '{"id":1,"name":"John","age":25}'
var jsonObj = JSON.parse(jsonString)
console.log(jsonObj)
```

We can also convert the **String** to **Json**
JSON.parse(String file name)

Function in JavaScript

A JavaScript function is a block of code, designed to perform a particular task. If we don't call the function it will not execute the code inside the function.

Syntax of function :

Function **myFunction**() {

code

}

Function without parameter

Function **myFunction**(parameter1, parameter2) {

code

}

This is called function with parameter.

We can declare function below ways :

Function as a statement

```
function Add(num1,num2){
    let sum = num1+ num2;
    return sum;
}

let res = Add(7,8);
console.log(res); // 15
```

Function as an expression

Function also can be declared by function expression. A function expression can be stored in a variable. And that variable is used to call the function. This type of function is called anonymous function. [**a function without name].** Like below

```
let sum = function(a, b)
{
  alert("Result is", (a + b) / 2);
}

let number1 = prompt("Enter first number");
number1 = parseInt(number1);

let number2 = prompt("Enter second number");
number2 = parseInt(number2);

sum(number1, number2);
```

```
Enter first number> 1
Enter second number> 3
Result is 2
```

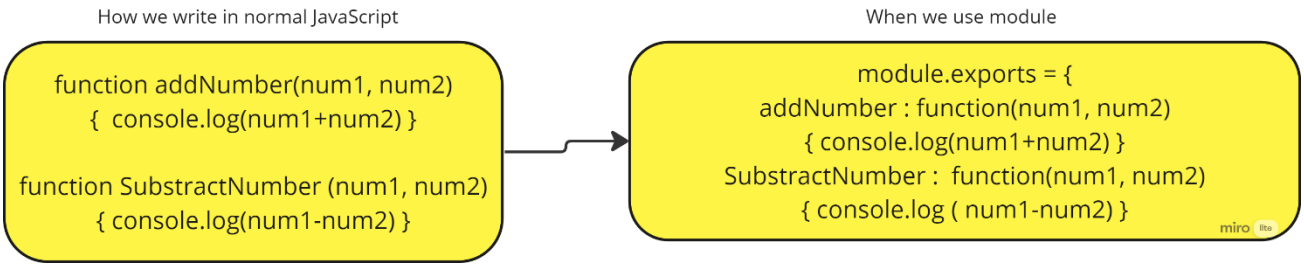
Module in Node JS

In simple terms, a module is a piece of reusable JavaScript code. It could be a .js file or a directory containing .js files. You can export the content of these files and use them in other files.

Modules help developers to follow the DRY (Don't Repeat Yourself) principle in programming.

Create Your Custom Modules

You can create our own modules, and easily include them in our applications.
Suppose there is **function** : **addNumber, SubstractNumber** you want to reuse the function then you will create a module and code will be JavaScript code.



Import Module (to reuse the function)

You load the module with the **`require`** function. You need to pass the name of the module you're loading as an argument to the **`require`** function.
Syntax : let variable = require('nameOfModule')

Write a program to manipulate 2 numbers using custom module in node js

```
module.exports = {

    addNumber :  function( num1, num2)
    {
        console.log(num1 + num2)
    },
    SubstractNumber :  function( num1, num2)
    {
        console.log(num1 - num2)
    }
}
```

```
let myReuse = require ("./Reuse");

myReuse.addNumber(1,2)
myReuse.SubstractNumber(4,2)
```

```
3
2
```

NPM [node package manager]

NPM is a package manager for Node.js . www.npmjs.com hosts thousands of free packages to download and use. The NPM program is installed on your computer when you install Node.js.

What is a Package?

A package in Node.js contains all the files you need for a module.

If you want to download any package the command will be : **npm install {package name}**

Once you install the package, you can use that package with the help of require function.

```
let downloadedPackage = require ('Name of the package')
```

Arrow function in JavaScript

Arrow function allow us to write shorter function syntax.

```
//Arrow Function

let sum = (a, b) =>   //Arrow function with 2 parameter
{
  alert("Result is", (a + b) / 2);
}

let number1 = prompt("Enter first number");
number1 = parseInt(number1);

let number2 = prompt("Enter second number");
number2 = parseInt(number2);

sum(number1, number2);
```

```
Enter first number> 1
Enter second number> 3
Result is 2
```

What is the difference between the arrow function and the normal function.

If we create any global variable in the program, then that variable can be accessible in the arrow function, but can not accessible in the normal function. That means **this** keyword can be accessible inside the arrow function, but can not accessible inside the normal function.

What is asynchronous & synchronous programming mean ?

When in the application user try to save data, then until save the data in the backend, user need to wait. Basically users are locked, they can not perform any action on UI, until the processing getting completed. This is call **Synchronous programming**. But it is wasting time to user, because user can not perform action

On the other hand, in the **Asynchronous programming** user no need to wait until the process complete. In this case user is un-blocked, User can do other task parallelly. once the process of saving the data is finished then user get notified.

This **asynchronous programming** can be done with the help of **callback** and **promise**

Why JavaScript called asynchronous programming?

```
// Write a program with use of setTimeout
setTimeout(function(){
  console.log("1st");
},2000)

console.log("2nd")
console.log("3rd")
```

In this program as you can see that 2nd and 3rd execute first and 1st execute later, because JavaScript is asynchronous, it does not wait for the execution, it runs parallelly.

What is setTimeout()

setTimeout() is an asynchronous function, that means this timer function will not pause execution until the other task get finish.

This function takes a *callback function* and *millisecond* as an argument

```
setTimeout(function () {
  console.log("Soumik");
}, 3000);

console.log("Saha");
```

```
Saha
Soumik
```

So here as you can see, we have used a callback function and 3000 milliseconds as an argument. So, this function is called callback function because we passed this function as an argument to **setTimeout** function.

JavaScript Callback function

A JavaScript callback is a function which is to be executed after another function has finished execution. Another Simple definition would be - Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function.

Why callback function is required, what is the benefit of it.

So I will give you 2 example

1

```
function myDisplayer(total) {
  console.log(total)
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  return sum;
}

let result = myCalculator(5, 5);
myDisplayer(result);
```

```
10
```

2

```
function myDisplayer(total) {
  console.log(total)
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  myDisplayer(sum);
}

myCalculator(5, 5);
```

```
10
```

The problem with the first example above, is that you must call two functions to display the result.

The problem with the second example, is that you cannot prevent the calculator function from displaying the result.

Now it is time to bring in a callback.

```
function myDisplayer(total) {
  console.log(total)
}

function myCalculator(num1, num2, callback) {
  let sum = num1 + num2;
  callback(sum);
}

myCalculator(5, 5, myDisplayer);
```

```
10
```

In the example above, `myDisplayer` is a called a **callback function** It is passed to **myCalculator**() as an argument.
NOTE : When you pass a function as an argument, remember not to use parenthesis. Right: **myCalculator(5, 5, myDisplayer);**
Wrong: ~~myCalculator(5, 5, myDisplayer());~~

Issue with the Callback functions

One issue that can arise when using callback functions is the problem of "**callback hell**" or "**pyramid of doom**." This occurs when you have multiple nested callbacks, making the code difficult to read and understand. And also another problem is inversion of control.

JavaScript Promises

The solution of the **callback hell is promises.** A promises is a "**promise of code execution**". The code either executes or fail. In the both cases the subscriber will be notified.

The **syntax of promise :**

let promise = new Promise(function(resolve, rejected)

{

// code execution

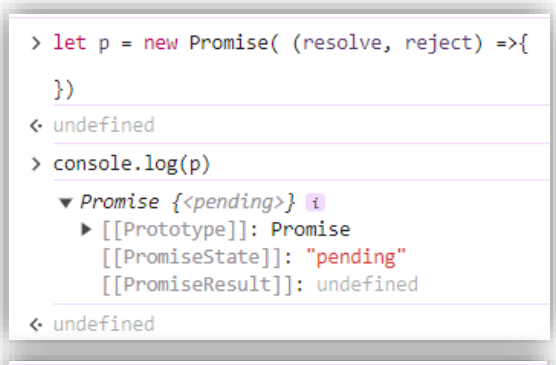
}

function(resolve & reject) callback function provided by the JavaScript. They are called like this.

resolve(value) -------- the promise resolves successfully and returns a value.
reject(error) ------- the promise fails with an error.

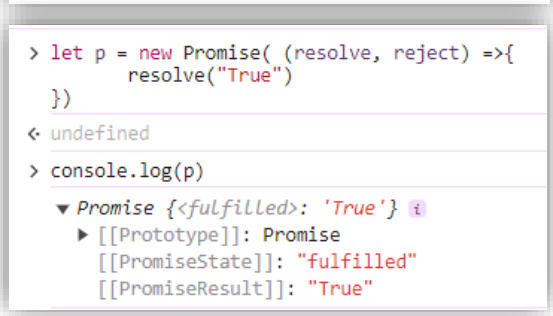
The promise has **PromiseState** which is initially **pending**. Change to either **fulfilled,** when resolve is called or either **rejected,** when reject is called.

The promise has **PromiseResult** which is initially **undefined**. Then changes to **value,** if resolved. or convert to **error**, when rejected.

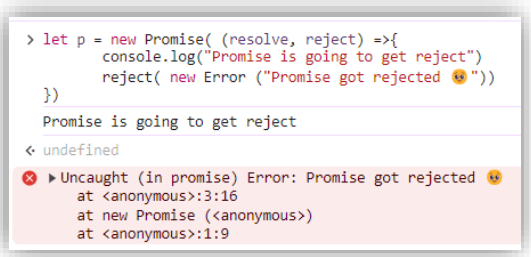


Here we have defined a promise and the **promise-state** is pending and the **promise-result** is undefined, because we haven't called

The **reject or resolve.**



Here we have defined a promise and the **promise-state** is **fulfilled** and the **promise-result** we got, because we have called the **resolve.**



In this promise we have used **reject.** That's why promise got

Rejected.

Suppose I have multiple promises, it will get executed parallelly, not sequentially

```
> let promise1 = new Promise ((resolve,reject)=>{

    setTimeout(()=>
        {
            console.log("I am a promise which got rejected")
            reject(new Error("I got rejected"))
        }, 5000)
})

let promise2 = new Promise ((resolve,reject)=>{

    setTimeout(()=>
        {
            console.log("I am a promise which got selected")
            resolve("Selected")
        }, 5000)
})

    console.log(promise1, promise2)
```

Here as you can see 2 promises one is **resolved** and another one **reject**

```
▼ Promise {<pending>} i
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "rejected"
  ▶ [[PromiseResult]]: Error: I got rejected at <anonymous>
```

```
▼ Promise {<pending>} i
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "Selected"
```

The states of promises in JavaScript

pending: the promise is still in the works.
fulfilled: the promise executed successfully.
rejected: the promise fails.

What is then() and catch() in promise?

when a promise is successful, you can then use the resolved data with the help of **then()**

when a promise fails, you catch the error, and do something with the error information **catch().**

When promise is fulfilled?

When a promise is fulfilled, you can access the resolved data in the **`then`** method of the promise:

When a promise is rejected

When a promise is rejected (that is, the promise fails), you can access the error information returned in the **`catch`** method of the promise:

```
> let promise1 = new Promise ((resolve,reject)=>{

    setTimeout(()=>
        {

            reject(new Error("I got rejected"))
        }, 5000)
})

let promise2 = new Promise ((resolve,reject)=>{

    setTimeout(()=>
        {

            resolve("Got resolved")
        }, 5000)
})

promise1.catch((error)=>
    {
        console.log("in promise1 error occured")
    })
promise2.then((value)=>
    {
       console.log(value)
    })
◇ ▶ Promise {<pending>}
in promise1 error occured
Got resolved
```

So here as you can see the Promise1 got failed with error and to handle this error data we used **catch().**

And as you can see the promise2 got executed and to get the resolve data we will use **then().**

Promise Chaining

Promise Chaining is a simple concept by which we may initialize another promise inside our **.then()** method and accordingly we may execute our results.

```
let promise = new Promise((resolve, reject)=>
    {
        resolve("I am the first promise")
    })
    promise.then((value)=>
    {
        console.log(value)
        return new Promise((resolve, reject)=>
            {
                resolve("I am the nested promise")
            })

    })
    .then((value)=>
    {
        console.log(value)
    })
```

```
I am the first promise
I am the nested promise
```

JavaScript Async/Await

Async and await make the promise easier to write.

async function always returns a promise. A function can be made async by using **async** keyword.

Example:

```
async function harry() {
    return 7; }
```

So here we can do like this

```
harry().then((x)=> {
console.log(x) })
```

So async ensures that the function returns promise and wrap the non-promises on it.

```
async function myDisplay()
{
   return "Soumik Das"
}
myDisplay().then((a) =>
{
  console.log(a)
})
```

```
Soumik Das
```

So there is a another keyword called **await.** That works only inside async function.

let value = await promise

The await keyword makes JavaScript wait until the promise settles and return value.

```
async function weather() {
  let delhiWeather = new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("27 Deg");
    }, 3000);
  });

  let bangaloreWeather = new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("21 Deg");
    }, 6000);
  });

  console.log("Fetching Delhi Weather Please wait ...");
  let delhiW = await delhiWeather;
  console.log("Fetched Delhi Weather: " + delhiW);

  console.log("Fetching Bangalore Weather Please wait ...");
  let bangaloreW = await bangaloreWeather;
  console.log("Fetched Bangalore Weather: " + bangaloreW);
  return [delhiW, bangaloreW];
}

weather().then((value) => {
  console.log(value);
});
```

```
Fetching Delhi Weather Please wait ...
Fetched Delhi Weather: 27 Deg
Fetching Bangalore Weather Please wait ...
Fetched Bangalore Weather: 21 Deg
[ '27 Deg', '21 Deg' ]
```

```
let weather = async ()=>
  {
     let delhiWeather = new Promise((resolve, reject) => {
       setTimeout(() => {
         resolve("27 Deg");
       }, 3000);
     });

     let bangaloreWeather = new Promise((resolve, reject) => {
       setTimeout(() => {
         resolve("21 Deg");
       }, 0000);
     });

    console.log("Fetching Delhi Weather Please wait ...");
    let delhiW = await delhiWeather;
    console.log("Fetched Delhi Weather: " + delhiW);

    console.log("Fetching Bangalore Weather Please wait ...");
    let bangaloreW = await bangaloreWeather;
    console.log("Fetched Bangalore Weather: " + bangaloreW);
    console.log('\n')

  }
let bioDataOfSoumik = () =>
  {
    console.log("Fetching Bio Data of Soumik Please wait ...");
    setTimeout(()=>{
      console.log("Hey I am Soumik, I am a Web Developer");
    },2000)
    console.log('\n')

  }
let adressOfSoumik = () =>
  {
    console.log("Fetching Adress of Soumik Please wait ...");
    setTimeout(()=>
      {
        console.log("Hey I am Soumik, I am from Bangalore");
      },2000)
      console.log('\n')
  }

let main1 = async () =>
  {
    let a = await weather();
    let b = await bioDataOfSoumik();
    let c = await adressOfSoumik();
  }
main1();
```

```
Fetching Delhi Weather Please wait ...
Fetched Delhi Weather: 27 Deg
Fetching Bangalore Weather Please wait ...
Fetched Bangalore Weather: 21 Deg


Fetching Bio Data of Soumik Please wait ...


Fetching Adress of Soumik Please wait ...


Hey I am Soumik, I am a Web Developer
Hey I am Soumik, I am from Bangalore
```

So in this program we have created a async function name **weather** and inside we created 2 promise and also we used **await** in all the promise. So that's why It executes sequentially.

On the other hand we have created 2 function **bioDataOfSoumik, adressOfSoumik** that we created normal function.

We created another async function named **main1** and we called 3 function inside **main1** function. And we have used **await** keyword. Since the **bioDataOfSoumik, adressOfSoumik** are the normal function that's why they are executed parallelly.

Event concept in Node Js

what is events and EventEmitter

If you worked with the JavaScript or html, you know whenever user click a button or anything then an event will be triggered, to handle the event we write the event-handler function. Event is like a signal.

But in the backend side, Node.js offers us the option to build a similar system. So there is a class name **EventEmitter**. So EventEmitter which is mainly used to create the events. So this class is come from the **events** module. And **emit** is used to trigger events, [similarly when we click a button event will be trigger]. Now we need to handle the events as well, typically we write code to handle events within **event listeners**[on] or callback functions.

For example: Below I have create a event named **speak.** So when we trigger the speak event with the help of **emit.** Then to handle this event we write the on event listener.

```
let events = require("events");

// create an instance of EventEmitter
let eventEmitter = new events.EventEmitter();

// Define the event handler
eventEmitter.on("speak",()=>{
    console.log("Soumik is speaking")
})

// Emit the event
eventEmitter.emit("speak")
```

So here first I am importing the events module. Then we are creating the object of the EventEmitter class. And we are creating a event name **speak.** To handle the event we have defined the event handler

```
Soumik is speaking
```

How node js events works in real life

Write a program where event will be triggered when user hit the API

```
// write a code where events will be triggered when the user hit
the API
const express = require('express');
const events = require('events');
const app = express();
app.listen(4000)
let eventEmitter = new events.EventEmitter();

eventEmitter.on('hitApi', (data) => {
  console.log('Api got hit: ', data);
})
app.get("/", (req, res) => {
  res.send("Api called");
  eventEmitter.emit('hitApi', "general api")
})

app.get("/update", (req, res) => {
  res.send("Update api called");
  eventEmitter.emit('hitApi', "update api")
})

app.get("/put", (req, res) => {
  res.send("put api called");
  eventEmitter.emit('hitApi', "put api")
})
```

```
Api got hit:  general api
Api got hit:  update api
Api got hit:  put api
Api got hit:  general api
```

What is webserver

So in the client server architecture, One side we have a client which is nothing but a mobile, browser running on a desktop. And from the client side we are going to send a request to the server. So there has to be somebody who should be respond to the request and that is server. So we built the server, with the help of Node JS. So I can built a webserver which somebody sends a request and server will be capable to handle the request. Server is like a house. Just the way to enter into a house we need a door. Similarly to enter into a server we need a port. So when we will create a webserver we need to create a port number also. On which request can be sent and server constantly listening to this request and respond accordingly. So we will use a module named **Express** to create webserver.

Installing Express

So we can install in 2 ways

1. Keep the module within the project : **npm install {module name}**
2. Keep the module globally in your system, so that all project can share **npm install -g {module name}**

So here the recommended option to download is option-1 . Because when you deliver this project to production and quality system or different system or your friends computer. So built a node js program and you wanted to run that program on friends computer, How your friend know that this module needs to be install. So the right practise is keep the module within the project. And also store the information of all the modules related to project inside a special file called package.json

Package.json / What Package.json file Contains?

It holds the information about our project like Project name, license, author name, description, version, It hold all the dependencies required by node project.

So in this way when you give your project to your friend to colleague they can get to know what are all dependencies are required to run this project by seeing the package.json file.

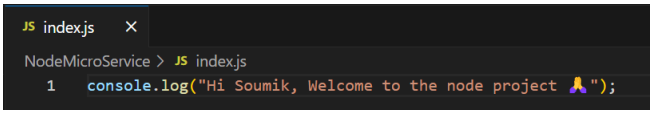
So when we deploy our project in another system/ machine we can use **npm install** command. So this command checks the package.json file. It go and scan the package.json file. It reads all the module which is required by the project and download one by one.

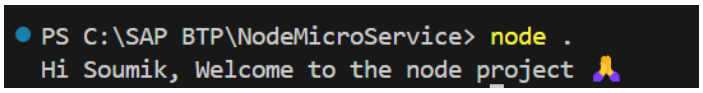
Within the project there will be folder which will be created called **node_modules.** So in that folder system will install all the dependencies.

Steps to initialize a fresh node project

So I will create a new Project Named **NodeMicroService.** Where we will do all our node project.

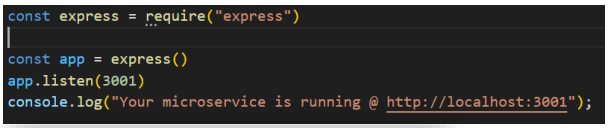
1. Run command **npm init** to initialize a new node project and need to answers some questions and after that project gets created automatically. it will create a package.json file automatically.
2. So as we mentioned **index.js** will be my starting file.
3. So we will create the **index.js** file inside our project
4. To test our node project, we can use command called **node .** So when we use this command it will run the default file **index.js**



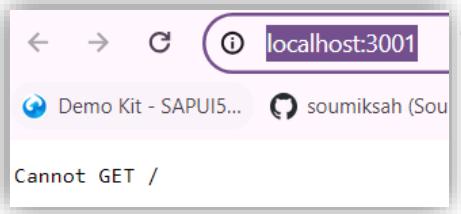


Create a web server and microservice using Express

1. Install express module and add that as dependency and also to our project **npm install express --save**
2. So after installing the express module it will add an entry for **`express`** in your **`package.json`** file under the **`dependencies`** section, along with the version number installed. And also within the project it will create a folder called **node_modules.**
3. Use express module to create web server (refer index.js)

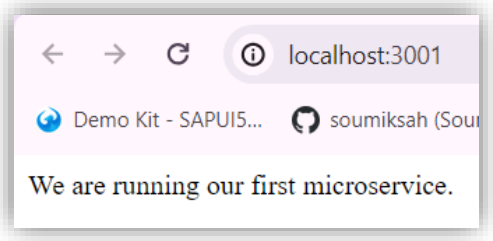


So I created the web server, with the help of this code, but when I opened the link http://localhost:3001/ then I got an



Cannot GET / So the reason is we have created a server but we did not handle anything for response. There is not code written to handle **post, get, put etc.**

So now we will handle get request, I will update the above code



This code is for creating a basic web server using the Express.js framework in Node.js.

Here's a breakdown:

It imports the 'express' library.

It initializes an Express application. It defines a route for handling HTTP GET requests to the root URL ('/') with a callback function that sends the response '**We are running our first microservice**.'.

It starts the server on port 3001. In summary, when you run this code and access http://localhost:3001 in your web browser, you will see **We are running our first microservice**' displayed in the browser. This is a simple example of creating a web server using Express.js.

So when we do code changes, then we need to restart the service. So for that I can use a another module given by node is called **nodemon** with the help of this I don't require to do the code change every time. So in my project this module is not required. This module is only for the developer.

npm install nodemon --save-dev

--save-dev: This flag tells npm to save the package as a development dependency in your project's **package.json** file. Development dependencies are packages that are only needed during development and testing, not in production. This means that when someone else clones your project and runs npm install, npm will install the necessary development dependencies as well.

```
npm install nodemon --save-dev
```

```
"devDependencies": {
  "nodemon": "^3.1.0"
}
```

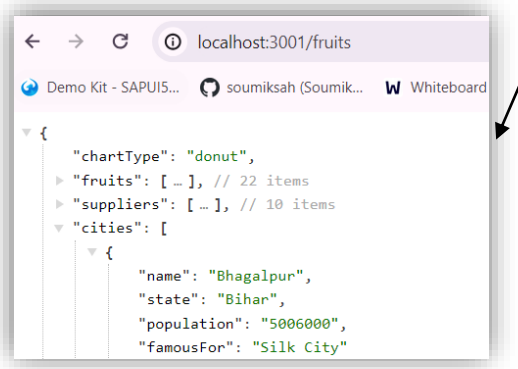
So you can use this dependency by using the command : **nodemon .**

Showing a Json in the webserver

```
app.get("/fruits", (req,res)=>{
    res.json({
        "chartType": "donut",
        "fruits": [ ···
        ],
        "suppliers": [ ···
        ],
        "cities": [ ···
        ]
    })
})
```

app.get("/fruits", (req,res)=>{ ... }) : This defines a route handler for GET requests to the "/fruits" endpoint. When a client sends a GET request to this endpoint, the callback function `(req,res)=>{ ... }` will be executed.

res.json({ ... }) : This method sends a JSON response back to the client



So this is the first microservice with the hardcoded data on Node Js.

Exercise :

1. Prepare a sample JSON payload for demo vendor data, and return that with endpoint/vendors

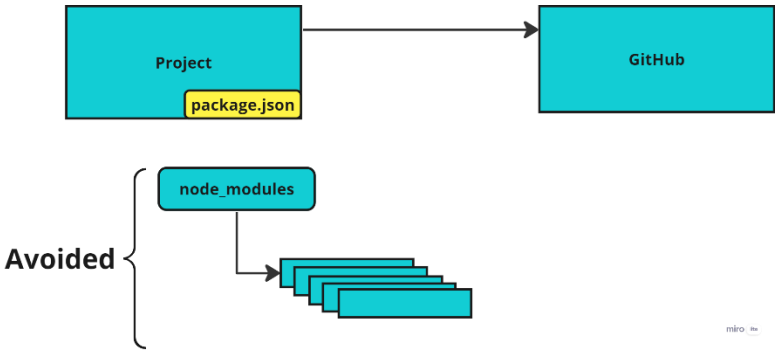
2. Create a new endpoint with /vendor/:id and just check in a collection of vendors if there is a vendor with this id passed by browser, and return that single vendor record.

How do you push the code changes to the git repository when it comes to node.

In our project we have the **package.json** file where we have all the name of the dependencies. And also we have the **node_modules** where we have the modules downloaded. So there are many modules are downloaded, because one module are dependent on other. But when we push he code changes to the GitHub, we will not push the **node_modules** to GitHub. Because if we push the node modules then the GitHub will be flooded with the huge amount of unnecessary code.

Where as the person who is going to use this Project can actually run a **npm install** command in their computer after getting the code from the GitHub. So I will only push the project and package.json to the GitHub. I will avoid sending the node_modules to the GitHub.

So for that I will create a file called **.gitignore [**. means it is a hidden file**]** And inside the file I will write the folder name, so that folder will be greyed out that indicates that when I push my code to GitHub this folder will be abandoned and will not push to GitHub.



Prepare a sample JSON payload for demo vendor data, and return that with endpoint/vendors

So I have created a new file **server.js** and defines a route handler for GET requests to the "/vendor" endpoint. When client sends a get request the callback function will be executed. And **res.json()** sends a Json response back to the client. Now to run the file, the command would be **node server.js**

Create a new endpoint with /vendor/:id and just check in a collection of vendors if there is a vendor with this id passed by browser, and return that single vendor record.

```
//create a vendor json array and declared as a global variable
this.aVendor = [
  {5 hidden lines},
  {5 hidden lines},
  {5 hidden lines},
];
```

So I will declare a global variable with **this** keyword. So here in this picture as you can see I have declared a global variable named **aVendor** where I kept an array of the Json of the vendor details. And everywhere instead of writing the same Json multiple times, I will reuse this global variable.

```
app.get("/vendors/:id", (req, res) => {
  let Element = this.aVendor.filter((element) => {
    return element.id == req.params.id;
  });
  if (Element !== undefined) {
    res.send(Element);
  } else {
    res.send("Not found");
  }
});
```

So **app.get("/vendors/:id", (req, res) => { ... }):** This line sets up a route handler for the GET request method to the "/vendors/:id" endpoint. The **:id** portion in the endpoint is a parameter, which means it can be any value and will be accessible through **req.params.id** in the handler function.

Array.filter() method iterates through each element of the array and returns the element that satisfies the condition. So the condition is compares the **id** property of each element with the value of **req.params.id**.

After that we check if the **Element** is not undefined then send it as response using **res.send()**
If **Element** is undefined, it sends a response with a message indicating that **Not found**.

Below is the complete code

```
const express = require("express");
const app = express();
app.listen(3001);

//create a vendor json array and declared as a global variable
this.aVendor = [
  {5 hidden lines},
  {5 hidden lines},
  {5 hidden lines},
];

app.get("/vendors", (req, res) => {
  res.send(this.aVendor);
});


app.get("/vendors/:id", (req, res) => {
  let Element = this.aVendor.filter((element) => {
    return element.id == req.params.id;
  });
  if (Element !== undefined) {
    res.send(Element);
  } else {
    res.send("Not found");
  }
});
```

So the reason for writing the **:id** like this, because we want to use **id** as a parameter that means in future we can change the value of the id

We can also define a set of easy to use start script, this we can setup in **package.json** file in the **scripts** section.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "main" : "node .",
  "server" : "node ./server.js"
},
```

So now if I want to run **index.js** then we will run the command **npm run main** If we want to run **server.js** then we will run command **npm run server**

Middleware

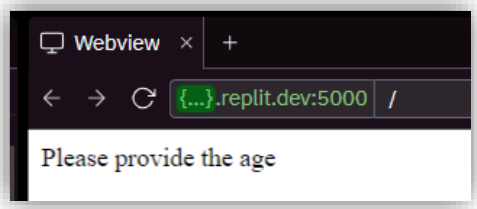
So middleware is a function, which accepts 3 parameters **req resp next**. With the help of middleware we can access the request and response and also modifies them. So middleware is used to Check authorization of an request, Add security to the request. So we can create a middleware and we can reuse that to all the routes. So while creating the middleware it accepts 3 mandatory parameter **req**, **resp**, **next**. with the help of **req** and **resp** parameter we can modify or access the routes. And **next** basically it is a function, so when we sends a request to fulfil the request the **next** function helps. If we don't call the **next** function then the request which we send can not fulfilled rather it will be loading continuously. So after create the middleware if we want to use that we will write the code **app.use(middleware name)**

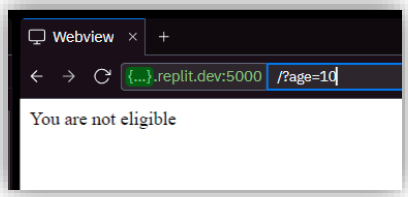
```
const express = require('express');
const app = express();

const middleWare =(req,res,next)=>{
  console.log("middleware is called");

}

app.use(middleWare)

app.get( '/', (req, res) => {
  res.send('Welcome to Home Page');

})

app.get( '/user', (req, res) => {
  res.send('Welcome to User Page');
})
app.listen(5000)
```

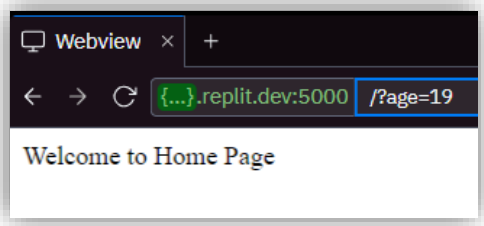
So here as you can see the request are not executed rather it is loading because we did not call the **next()** function.

Give a real example of middleware: If user age has 18 or above then only access the page otherwise not.

```
const express = require("express");
const app = express();
const middleware = (req, res, next) => {
  if (!req.query.age) {
    res.send("Please provide the age");
  } else if (req.query.age < 18) {
    res.send("You are not eligible");
  } else {
    next();
  }
};
app.use(middleware);

app.get("/", (req, res) => {
  res.send("Welcome to Home Page");
});

app.listen(5000);
```







```
app.post("/vendors", (req, res)=>
{
    var postReqBody = req.body
    postReqBody.id = "1"
    res.json(postReqBody)
})
```

So here we are creating a post request sent to the **/vendors** endpoint. So whatever post request body we will pass that will show as response. So now I want to pass Json as post request body but it will not because express don't allow the Json request body.

So for that I will use a middleware for the Express, to tell that if we pass the Json as a post request express need to allow the Json also. So we google it and found a middleware name **express.json()**

Express.json() : So this middleware parse the incoming request with JSON payloads.

To use this middleware the code will be `app.use(express.json());`

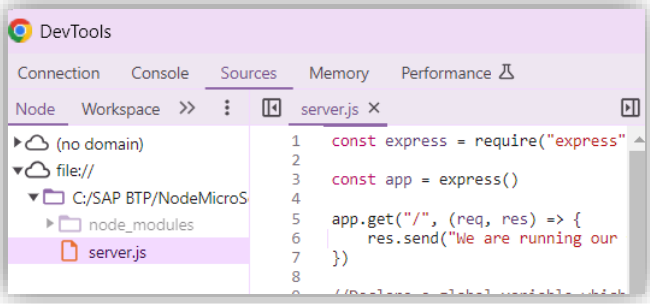
So now after adding this middleware, we will be able to post the Json request. So we will validate the post request in the Postman.

```
app.post("/vendors", (req, res) => {
    var postReqBody = req.body;

    //Code for generate randon uuid number
    function e1() {
        var u='',i=0;
        while(i++<36) {
            var c='xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'[i-1],r=Math.random()*16|0,v=c=='x'?r:(r&0x3|0x8);
            u+=(c=='-'||c=='4')?c:v.toString(16)
        }
        return u;
    }

    let randomId = e1()
    postReqBody.id = randomId
    res.json(postReqBody)
})
```

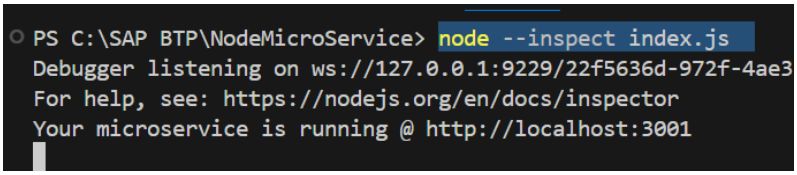
How to debug in Node Js

For that in the terminal we need to write a code **node --inspect {filename}** After that go to the browser and click on the **F12.** You will get a **node js** icon.

So to debug in node js click on the node js icon then go to the source, and there in the code you can add breakpoint.



After adding the breakpoint, if we click on the F10 it will move to the next line.



YEOMAN [NR]

So yeoman is a node framework, built on node which is used to built web application. So go to the yeoman generator https://yeoman.io/generators/ search **anubhav-basicfiori** So this creates a fiori app automatically. So to use this generator [**anubhav-basicfiori]** First we need yeoman tool in my computer. So first we will create a Folder name **yeoman** [C:\yeoman] then inside this we will install the yeoman, the command is **npm install -g yo** and then to install the generator the command is **npm install -g generator-anubhav-basicfiori** After done this to create the fiori app, we need to run the command **yo.** And choose the Anubhav Basicfiori option. After that it will ask you some question and give the answer to that question and after that the fiori application generated automatically. Now I will copy the webapp folder and paste it to my project folder **NodeMicroService**

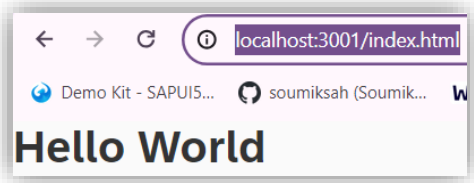
So for testing the fiori app, we will go to index.html file and comment the code, which is inside the script tag. And I will put a **Hello World** inside the body tag. And in the server.js file I will add piece of code, which is given below.

```
<body class="sapUiBody">
    <div id="content"></div>
    <h1>Hello World</h1>
</body>
```

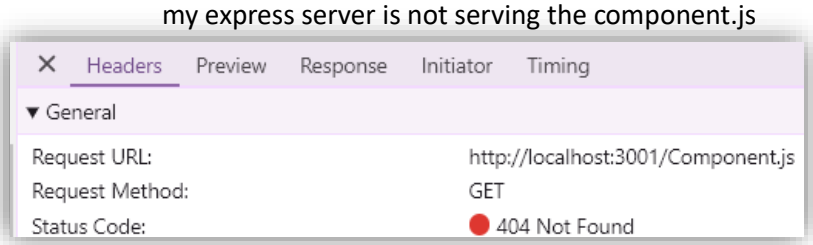
```
app.get("/index.html", (req,res)=>{

    res.sendFile(__dirname + '/webapp/index.html')
})
```

Now I will run the endpoint http://localhost:3001/index.html

It will print **Hello World**



Now I will test the actual fiori app which Anubhav has given me. So I will uncomment the code and Remove the Hello World from the index.html file. After that I hit the URL http://localhost:3001/index.html it is now not loading anything. So now to see what happened I will go to the network and check the error it is showing that can not get the component.js file because my express server is not serving the component.js



Now I will add code in the server.js file for serve the component.js file. After add the code again I run and showing the error in the network that can not get the manifest.json file because my express server is not serving the manifest.json file. Now I will add code in the server.js file for serve the manifest.json file. Again I run and showing the error says can not get App.view.xml. So if there is 100 folder/file we need to add all in the server.js file. So this will be a repetitive task, So to solve this problem, we will use a another middleware which mainly servers everything from the webapp folder, so we don't need to serve one by one.

app.use("/", express.static("webapp/")) : So this is the middleware so any file which needs to load that it will search in the webapp folder and it will load. Suppose I want **/manifest.json, /server.js, /App.view.xml** so that file it will look in the webapp folder and it will load. So now everything got load and my fiori app will run nicely.

Exercise : Deploy the above created fiori app in the cloud foundry by adding the manifest.yml file.

So first I will create a manifest.yml file and with the necessary details and I want to push my application with the command **cf push.** But when I will do the push it will look for the manifest file. So in the command prompt I need to go to that folder where this manifest file is present [**C:\SAP BTP\NodeMicroService**]. And from that location, we need to do **cf push.**

Note : So before doing the **cf push** we need to do **cf login.** Otherwise, it will throw an error "**token expired"**

What is an SAP AppRouter anyway? [NR]

The AppRouter is a Node.js library used as a single entry point for an application running in the Cloud Foundry, SAP BTP.

Let's say a user is trying to access an application by calling the URL; there are a few things that would happen:

The request comes to the AppRouter of the application.

The AppRouter checks if authentication is required (configured in xsapp.json of the application) and sends information like client location, cookies, and requested page details to the browser.

If authentication is required, the AppRouter redirects to the SAP's IDP Login page.

Once the login is successful, the AppRouter redirects to the index.html of the application.

26-27 got skipped Started from 28

Challenges for developer in BTP

App router implementation was difficult ---> It is laborious work to setup project and security.
What is SAP's recommendation to build cloud native apps in BTP ----> Best practise.
Are there any gold standard which industry should follow w.r.t SAP BTP
Many times as a developer, I need to take care of loading all required dependencies
Reusability of code which can speed up my development
As a developer I want to focus more on functionality then the project setup
Tight coupling to DB or technology
Dependency on SQL

So above all are the challenges, which you faced,
The solution of all these problem is one - using a framework so that's where there are two main powerful framework which is supported in SAP BTP.
1st : RAP [Restful ABAP Programming Language**]** So this for those who are working on ABAP, who has knowledge on ABAP. They can use RAP on BTP

2nd : CAPM [Cloud Application Programming Language] Those who are developing open technology like **Java, Node js** with HANA or non HANA in this case CAPM is more relevant framework for us, which is solve all the above problem.

What is CAPM?

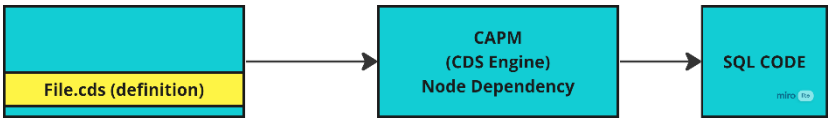
It is a framework which is collection of languages, libraries, and tools for building enterprise grade cloud native application in SAP BTP. This is recommended framework to built application. It takes most of the load away from the developer, so that the developer can focus more on the domain rather than focusing on the application structure, application dependency, application setup, all the required connections and settings. It speed up development like anything.

The official document of capm : https://cap.cloud.sap/docs/

Concept of CDS

CDS stands for Core Data & Services. So CDS says that anything you are developing, your design time definition will be a CDS file. It means that when you are designing or developing a software, you will define its components, such as data models and services, using CDS files. We will create **.cds** file for almost everything e.g. **service**, DB artifact, annotations. This is a design time file and SAP's framework (CAPM, RAP) will convert this design time definition after compilation to the runtime object.

Example : Suppose I have an Application. And on that application I have a file called **File.cds.** So now we give this **File.cds** to an engine basically to CAPM framework. And CAPM will have a engine called CDS engine, which is a node dependency. So when we give the file to CDS engine of CAPM, it does a beautiful job. So it check which database the application trying to talk and that information is stored inside the file called Package.json**.** And the CDS engine is going to convert the **CDS file** into a SQL code accordingly. And the database is dynamically generated by the engine depending on the database which is maintained in the Package.json**.** So in the Package.json if I mentioned HANA, then it will create the SQL code of HANA database. And also you can do same thing for services as well let say OData services. You write a .cds file which is a definition and then you tell which version of OData you want to generate V2 or V4 and the CDS engine then convert V2 or V4 metadata accordingly.



What is CDS-DK

So this contains the tools required to develop CAPM application including a command line tool called cds.

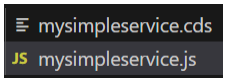
First CAPM Application

1. Install CDS-DK (Development Kit) , this contains the tools required to develop CAPM application including a command line tool called cds. **npm install -g @sap/cds-dk**
2. So I created a new project for building the CAPM application. [**C:\SAP BTP\CAPM1**]. So as we already installed the CDS-DK, so we got a command line. So we if write the command **cds --help,** then we can see all the list of commands.
3. So we will go inside the folder [**C:\SAP BTP\CAPM1**] and I will initialize the project for that I will run the command **cds init.** So it will give you a basic node JS project automatically.
4. So I will create simple rest service today. So everything in CDS, first you have to define and then you have to implement. So we need to follow a Mantra **Definition – Implementation.**
5. So in the SRV folder I will create a definition to define a service so I created the service which is **.cds** extension **[mysimpleservice.cds]** So when you run the command **cds run** it will look for the cds file and it will compile the cds file.
6. So before write the code in cds file we need to install a extension from the marketplace **SAP CDS Language Support.** So this will help to show the code completion.
7. So we will define a service in the **cds** file. And I will define a definition inside the service.

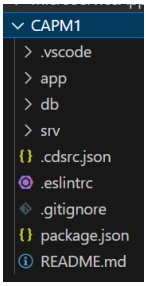
```
service mySrvDemo {

    function myFunction(msg: String)returns String;
    }
```

So **mySrvDemo** is the Service. And **function myFunction(msg: String)returns String;** this I have defined a definition. And I have defined a parameter of type String and name of the parameter is **msg.** And this function signature is a typescript syntax.

8. Now we have to implementation **.js** file for the service **mySrvDemo** which is in cds file. So the Cds file we only define the definition. The main implementation happen only in the JS file.
9. And one thing you have to remember that the cds file name and the js file name need to be same name, otherwise it will throw an error. So name has to be match otherwise system won't be able to detect that **mysimpleservice.js** file is the implementation of the **mysimpleservice.cds** file



10. So once both cds and js file is implemented, then we will run the project, So the command is **cds run.** And every code changes I don't want to restart the services for that I will use the command **cds watch**

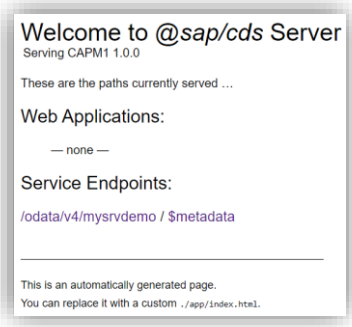


This section defines a service called mysrvdemo. Inside this service, there is a function called somesrv which takes a single parameter msg of type String and returns a String.

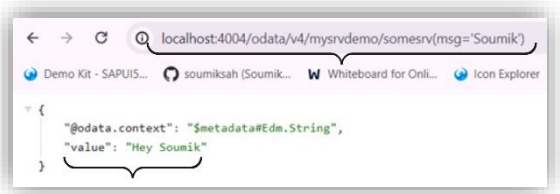
```
service mysrvdemo {

    function somesrv(msg: String)returns String;
    }
```

This section of code is the implementation of the mysrvdemo. It's written in JavaScript and exports a function named mysrvdemo. This function takes one parameter srv. Inside mysrvdemo, it registers an event handler for the event 'somesrv' using srv.on(). When this event is triggered, it expects req (request) and res (response) objects.

```
const mysrvdemo = function(srv){
    srv.on('somesrv', (req, res)=>{

        return "Hey " + req.data.msg;
    });
}

module.exports = mysrvdemo;
```

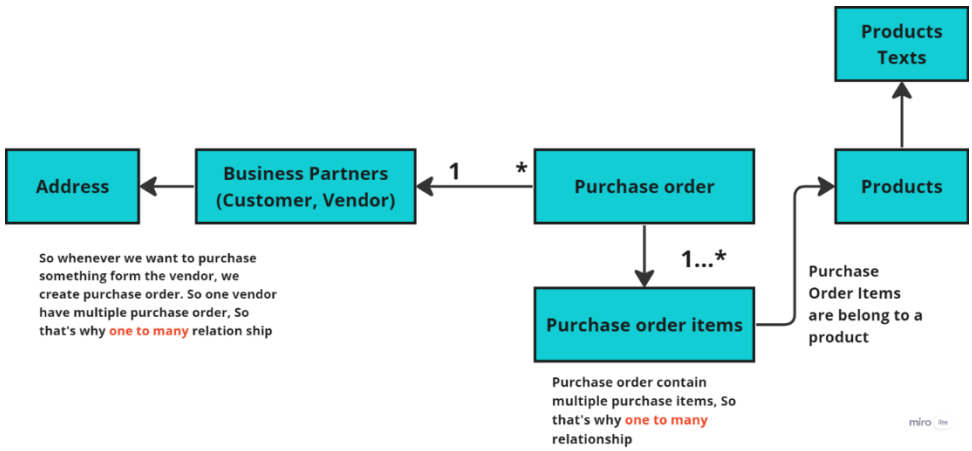


Run this service:



Creating DB table with CAPM

1. Add local database which helps us building and testing CAPM app locally in our computer, using a **SQLite** database which is free and file system based database which we can use.
2. Command is **npm install sqlite@3 [**compatible version of node 10 for SQLite**]** and this will be going to use only during development, that's why we save it as development dependency. **npm install sqlite@3 --save-dev** **[In my case it did not worked that's why I used command npm install sqlite3 --save-dev]**
3. Now after done installation the SQLite, I will create a database table. I will create a new file in the DB folder name **datamodel.cds** Inside this file we will define a **namespace soumik.db** So basically it is a name of the file, which is used to import or use this file content in another cds file.
4. So we will create EPM data model [**Enterprise Procurement Model]** So it is a real life scenario, where you have lots of table which contains data. This table structures are given below. So I will be creating this table one by one.



Suppose I have multiple table, I want to section them for that I will use **context.** So all my tables which hold the master-data, that I can keep inside the **master context.** And the table which contain transactional data will keep them in **transactional** context. So this way I can group my table in a section. So one **cds** file can have 100 of tables. But recommendation not more than 20-30 table.

So now I will create **business partner, address** table inside the master context. I will add the field/column name of **business partner, address** table**.**

```
namespace soumik.db ;

context master {

    entity businesspartner {

        key NODE_KEY: String(32);
        BP_ROLE: String(2);
        EMAIL_ADDRESS: String(128);
        PHONE_NUMBER: String(14);
        FAX_NUMBER: String(14);
        WEB_ADDRESS: String(64);
        ADDRESS_GUID: String(32);
        BP_ID: String(16);
        COMPANY_NAME: String(80)
    }

    entity address {

        key NODE_KEY: String(32);
        CITY: String(64);
        POSTAL_CODE: String(14);
        STREET: String(64);
        BUILDING: String(64);
        COUNTRY: String(2);
        VAL_START_DATE: Date;
        VAL_END_DATE: Date;
        LATITUDE: Decimal;
        LONGITUDE: Decimal;


    }
}
```

So here in this picture I have defined the columns for the table **businesspartner, address.** And wherever I have mentioned the **key** attribute, that I make a primary key. So there is no data in the table now, I have just created a skeleton of the Table. So now I will show a very interesting thing if I compile this table it will generate a SQL code. So if you want to see which code it generates for that you need to go to the **db** folder where this **datamodel.cds** file is present [**C:\SAP BTP\CAPM1\db]** and I will run the command **cds compile datamodel.cds -2 sql**

How to compile the entities?

For that I have to go to that folder, where all the entities are mentioned and from there I need to run time command

cds compile {cds file name where all the entities are mentioned} -2 sql

```
CREATE TABLE soumik_db_master_businesspartner (
  NODE_KEY NVARCHAR(32) NOT NULL,
  BP_ROLE NVARCHAR(2),
  EMAIL_ADDRESS NVARCHAR(128),
  PHONE_NUMBER NVARCHAR(14),
  FAX_NUMBER NVARCHAR(14),
  WEB_ADDRESS NVARCHAR(64),
  ADDRESS_GUID NVARCHAR(32),
  BP_ID NVARCHAR(16),
  COMPANY_NAME NVARCHAR(80),
  PRIMARY KEY(NODE_KEY)
);

CREATE TABLE soumik_db_master_address (
  NODE_KEY NVARCHAR(32) NOT NULL,
  CITY NVARCHAR(64),
  POSTAL_CODE NVARCHAR(14),
  STREET NVARCHAR(64),
  BUILDING NVARCHAR(64),
  COUNTRY NVARCHAR(2),
  VAL_START_DATE DATE_TEXT,
  VAL_END_DATE DATE_TEXT,
  LATITUDE DECIMAL,
  LONGITUDE DECIMAL,
  PRIMARY KEY(NODE_KEY)
);
```

So this is the SQL code, which got generated after compile. Here one thing I noticed whatever my **namespace, context name, table name** they all got joined with **"_"** and that became my actual database table name, which is generated by the system. **soumik_db_master_businesspartner, soumik_db_master_address.** And here you can not duplicate the **table name**.

And now we will deploy our CDS file to SQLite database.

Deploy our CDS file to SQLite Database

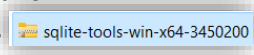
Now to deploy the cds file to SQLite Database,

1. One thing we need to remember that if we are inside the **db folder,** we have to come out from the db folder **[cd ../]**, that's means we have to be inside the project folder, but not inside the **db folder.**
2. So when we will be inside the project folder, then we will run the command
   cds deploy --to sqlite:{name to your database which you want to create}
   So in my case I want to create the database with the same name **soumik.db**
   so my command will be **cds deploy --to sqlite:soumik.db**
   So if we are inside the db folder and run the deploy command, then it will throw error, so we have to be outside of the db folder.

```
PS C:\SAP BTP\CAPM1> cds deploy --to sqlite:soumik.db
WARNING:

  Node.js v16 has reached end of life. Please upgrade to v18 or higher.

/> successfully deployed to soumik.db
```

3. Now I have successfully deployed and also created a database **soumik.db** inside the Project folder. Now to need to access the file **soumik.db** we need to have command line tool for SQLite.

4. So to install the SQLite command line tool we will go https://sqlite.org/download.html this and we will download the command line tool from **Precompiled Binaries for Windows.**



   After that unzip the zip file kept [**C:\sqlite-tools-win-x64-3450200**]. Now I need to set path, for that I will go to environment variable and there in the System Variable I will set the path in the path section.

5. After install, now I want to see my database table, for that I will write the command **sqlite3 soumik.db**. In SQLite command line tool every command starts with **"."**
   .help command is used to show all the commands.
   .tables command is used to show all the table.

```
sqlite> .tables
soumik_db_master_address          soumik_db_master_businesspartner
```

   So as you can see after run **.tables** command I got 2 table details. Now I want to read the data from a table, So for that the command would be **select * from {table name}**
   select * from soumik_db_master_businesspartner
   After run this command I saw there is no data printing, that means the table is empty.

```
PS C:\SAP BTP\CAPM1> sqlite3 soumik.db
SQLite version 3.45.2 2024-03-12 11:06:23 (UTF-16 console I/O)
Enter ".help" for usage hints.
```

6. Now I will fill the data in the table. So for that in the **db** folder we need to create a **csv** folder. And there I will add the csv file, which is having the data and also the csv file column name should matches the table column name. And **Be Careful !!** the csv file name should follow the pattern **: namespace-tableName.csv [** if context is not there]
namespace.contextName-tableName.csv [if context is there]

```
namespace soumik.db ;

context master {

    entity businesspartner {···
    }

    entity address {···
    }
}
```

So here I have 2 tables so the csv file name should be
soumik.db.master-businesspartner.csv
soumik.db.master-address.csv

If this name does not follow the pattern, then it will not work.

7. In the **6th** point you saw how we created the csv folder and add the csv file manually. Now I don't want to create this csv folder and csv file manually, so for that there is a command provided by sap is **cds add data.** So this command will automatically add a **data** folder inside the **db** folder. And it will create the csv file also automatically inside the data folder. So basically it checks how many entities are there and based on that it creates the csv file. But there is some condition we need to follow in order to get the csv folder :
    a. The cds file where all the entities are mentioned that file name should be **data-model.cds**
    b. So once we execute the command **cds add data** it will create a folder name **data** inside the **db** folder and inside that data folder it will create the csv file automatically.
    c. After creating the csv file automatically by the command, now suppose I have created a new entity and now if I run the command **cds add data** then whichever csv file previously created it will not overwrite them, it will just skip them and which is not created yet, it will create them automatically.

8. After adding the csv file, now I will deploy this file to SQLite database, for that the command is **cds deploy --to sqlite:soumik.db**

```
PS C:\SAP BTP\CAPM1> cds deploy --to sqlite:soumik.db
WARNING:

  Node.js v16 has reached end of life. Please upgrade to v18 or higher.

  > init from db\csv\soumik.db.master-businesspartner.csv
  > init from db\csv\soumik.db.master-address.csv
/> successfully deployed to soumik.db
```

9. So now after deploy the data I need to fetch the table data, So I will connect to my database using the command **sqlite3 soumik.db** and after that I will fetch the data from the table so I will use the command **select * from soumik_db_master_businesspartner**;

Building a foreign key relationship between businesspartner and address table

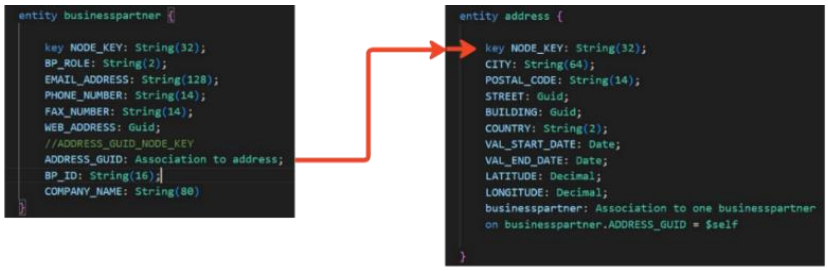
10. In the two tables **soumik_db_master_address & soumik_db_master_businesspartner** data kept in a such a way that there is a foreign key relationship between the two tables so that means the foreign key in the **businesspartner [ADDRESS_GUID]** table used to point the primary key in the **address [NODE_KEY]** table. So I will do a inner join.

select * from soumik_db_master_businesspartner inner join soumik_db_master_address on soumik_db_master_businesspartner.ADDRESS_GUID = soumik_db_master_businesspartner.NODE_KEY;

11. **We can also build a primary and foreign key relationship another way without writing the sql query.**

So in the **businesspartner** table if we do like this **ADDRESS_GUID: Association to address;** That means it will automatically check the primary key of the **address** table, and also built foreign key relationship from **businesspartner** table to **address** table. So now we built a association between **businesspartner** table and **address** table. So we can achieve this association with **?$expand.** So my association name is **ADDRESS_GUID.**
So when we go to the businesspartner table [BPSet] from that we can fetch the address table data with the help of this code : localhost:4004/CatalogService/BPSet?$expand=ADDRESS_GUID

```
entity businesspartner {
    key NODE_KEY: String(32);
    BP_ROLE: String(2);
    EMAIL_ADDRESS: String(128);
    PHONE_NUMBER: String(14);
    FAX_NUMBER: String(14);
    WEB_ADDRESS: Guid;
    //ADDRESS_GUID_NODE_KEY
    ADDRESS_GUID: Association to address;
    BP_ID: String(16);
    COMPANY_NAME: String(80)
}
```



So here we did association from **businesspartner table to address table.** And **ADDRESS_GUID** is my association which will help me to navigate from **businesspartner table to address table** and maintain the relationship between **businesspartner & address table**

So here we built a association from **businesspartner** table to **address** table. So we did it from the **ADDRESS_GUID** attribute to **NODE_KEY** attribute.
So the **ADDRESS_GUID** attribute value should match with the NODE_KEY attribute value, then the association will work perfectly.

12. Similarly we can do vice versa also that means from the **address** table we can go to the **businesspartner** table for that we need to build an association which will be from address table to businesspartner table. So the code will be
businesspartner: Association to one businesspartner on businesspartner.ADDRESS_GUID = $self

```
entity address {
    key NODE_KEY: String(32);
    CITY: String(64);
    POSTAL_CODE: String(14);
    STREET: Guid;
    BUILDING: Guid;
    COUNTRY: String(2);
    VAL_START_DATE: Date;
    VAL_END_DATE: Date;
    LATITUDE: Decimal;
    LONGITUDE: Decimal;
    businesspartner: Association to one businesspartner
    on businesspartner.ADDRESS_GUID = $self
}
```

So here in this **address** entity there is not such column name **businesspartner.** We just written this name for to built association from the address table to businesspartner table. So this type of association where there is not column name exists and we are the **on condition** manually we called **unmanaged association.**

```
businesspartner: Association to one businesspartner
on businesspartner.ADDRESS_GUID = $self
```

This is an **unmanaged association** because businesspartner column is not exists in csv and we write the **on-condition** manually

```
ADDRESS_GUID: Association to address;
```

This is called managed association because this **ADDRESS_GUID** column is available in the csv file and we did not write any on condition manually

So here we did association from the **address** table to **businesspartner** table and **businesspartner** is my association name which is an unmanaged association and help me to navigate from address table to businesspartner table. unmaged association because there is no column exists in the csv file named **businesspartner.**

businesspartner: Association to one businesspartner on businesspartner.ADDRESS_GUID = $self

So when we do the unmanaged association : Suppose there are 2 entity C and D. Now I did association from C to D. Now if I want to do vice-versa that means from D to C we want to association, then we will go for unmanaged association

13. So the above code defines association from address table to businesspartner table. This specifies the type of association which is **unmanaged** association, this is a one-to-one relationship from address table to businesspartner table. The meaning of the association is "**ADDRESS_GUID**" attribute in the "businesspartner" table must be equal to the "**NODE_KEY**" attribute of the "address" table. And **$Self** indicates the "**NODE_KEY**" attribute of the "address" table. So now both tables are connected. So to reflect the changes we need to again deploy the file to SQLite database.

14. What is the meaning of **$Self** ?
    So in a table if we mention $Self it indicates the primary key of that table. So we have used $Self in the address table, so it indicates the primary key of that address table which is NODE_KEY

15. So if we deploy this code will give an error, because in the businesspartner table there is a column name **ADDRESS_GUID** which built an association with primary key of the address table which is **NODE_KEY**. So the column name will be auto converted to **ADDRESS_GUID_NODE_KEY.** So now in the csv file we need to change the column name from **ADDRESS_GUID** to **ADDRESS_GUID_NODE_KEY**. After that we can deploy it will work.

16. As you can see we have used many places **String(64)** as a key, So in future if I want to change from **64** to **32,** then I have to change all the place. So rather I will define it one place and reuse everywhere. So I will create a datatype and I will reuse that datatype everywhere.
    type Guid : String(64)

```
ity address {

 key NODE_KEY: String(32);
 CITY: String(64);
 POSTAL_CODE: String(14);
 STREET: String(64);
 BUILDING: String(64);
 COUNTRY: String(2);
 VAL_START_DATE: Date;
 VAL_END_DATE: Date;
 LATITUDE: Decimal;
 LONGITUDE: Decimal;
 businesspartner: Association to one businesspartn
 on businesspartner.ADDRESS_GUID = $self
```

```
entity businesspartner {

    key NODE_KEY: String(32);
    BP_ROLE: String(2);
    EMAIL_ADDRESS: String(128);
    PHONE_NUMBER: String(14);
    FAX_NUMBER: String(14);
    WEB_ADDRESS: String(64);
    //ADDRESS_GUID_NODE_KEY
    ADDRESS_GUID: Association to address;
    BP_ID: String(16);
    COMPANY_NAME: String(80)
}
```

```
type Guid : String(64);

context master {

    entity businesspartner {

        key NODE_KEY: String(32);
        BP_ROLE: String(2);
        EMAIL_ADDRESS: String(128);
        PHONE_NUMBER: String(14);
        FAX_NUMBER: String(14);
        WEB_ADDRESS: Guid;
        //ADDRESS_GUID_NODE_KEY
        ADDRESS_GUID: Association to address;
        BP_ID: String(16);
        COMPANY_NAME: String(80)
    }

    annotate businesspartner with{···
    }

    entity address {

        key NODE_KEY: String(32);
        CITY: Guid;
        POSTAL_CODE: String(14);
        STREET: Guid;
        BUILDING: Guid;
        COUNTRY: String(2);
```

Exercise : Create remaining tables and add the relationship between them.

Aspect – Reuse of types

Aspect is reuse of types.

There are 2 types of aspects

1. Standard Aspects, which is provided by SAP : **@sap/cds/commons**
2. Custom Aspects.

So why we use aspects ?

1. Basically it is use for reuse. Just take an example I want to create 3 tables employee data, supplier data, customer data. So all these 3 tables if you look at they need a certain common fields like address, So address contains like city, country, street, Postal Code, region etc. So this fields I need to use all the 3 tables. So what we do is we create the address field as aspect. And we include the aspect to all the tables and at runtime all the address fields automatically comes to the all 3 tables. So here there is big advantage that is reusability that means you don't need to manually add all the fields to the all 3 tables. So tomorrow if a new field **landmark** add in the address filed, then it will automatically add in all the 3 tables as already address filed are included. So here the address filed we can call it as aspects**.** But sap has one level beyond that, so sap provides some ready to use aspects, so that we can use them directly. So we use this aspect with the table so the syntax is : **entity {entity name} : {aspects name} { }** So the aspect could be the standard aspect or could be the custom aspects. The standard aspects are come from the dependency **@sap/cds/commons**

Adding the standard aspect cuid & temporal with the employee table

So we created a employee table. So I will use a aspect name **cuid** in the employee table**.** This is a standard aspect. And that aspect will come from **@sap/cds/commons** So I need to import the aspects. So from this cuid aspects we will get the **ID** field which is a primary key. So rather than manually add the ID field in employee table, we will use this **cuid** aspect, ID field will automatically add by the cuid aspect. And also we can use the **temporal** aspect which will give me **validFrom** and **validTo** field automatically. So now I will compile the employee table and I will see all the fields which will come from the aspects. I will go inside the **db** folder where this **datamodel.cds** file is present [**C:\SAP BTP\CAPM1\db**] and I will run the command **cds compile datamodel.cds -2 sql**

```
using { cuid, temporal } from '@sap/cds/common';
```

So here we are importing the **standard** aspect from the @sap/cds/common

```
entity employee : cuid, temporal {

}
```

So here we are using the aspect with the entity name employee

```
CREATE TABLE soumik_db_master_employee (
  ID NVARCHAR(36) NOT NULL,
  validFrom TIMESTAMP_TEXT NOT NULL,
  validTo TIMESTAMP_TEXT,
  PRIMARY KEY(ID, validFrom)
);
```

So after compile the employee table we have seen that **ID**, **ValidFrom**, **validTo** field came from the standard aspect

So now I am adding the fields one by one in the employee table. So there is a field called **sex** where I will define gender of employee. So I know that the value of **sex** field is fixed constant value which can not be change. So in file called **common.cds** I will define the gender as Enum.

What is enum ?

An enumeration (Enum for short) is a special data type which contains a set of predefined constants. (unchangeable variable). To create an Enum, we use the **enum** keyword.

```
//create a gender aspects
type Gender : String(1) enum{

    male ='M';
    female = 'F';
    noBinary = 'N';
    noDisclosure= 'D';
    selfDescribe='S'

};
```

So this is my custom **Gender** I will define it as Enum as in the gender there are fixed values as you can see [**M,F,N,D,S**] constants. That's means the value are not required to change.

So in the **datamodel.cds** file in the employee table in the **sex** field I will use this custom field Gender, first I will import this. And then we will use.

```
using {soumik.common} from './common';
```

So inside the parenthesis I will give the namespace of the cds file. Where I have mentioned the custom field. And common is my file name where all the custom field mentioned

```
sex: common.Gender
```

So in the employee table there i s a field name **sex,** there I want to reuse my custom field. So the syntax will be : {file name}. {custom field name} ---> common.Gender

What is the regular expression to maintain the format phone number & mail :

Phone : '((?:\+|00)[17](?: |\-)?|(1 -)?|(?:\+|00)[1-9]\d{0,2}(?: |\-)?|(?:\+|00)1\-\d{3}(?: |\-)?)?(0\d|\([0-9]{3}\)|[1-9]{0,3})(?:((?: |\-)[0-9]{2}){4}|((?:[0-9]{2}){4})|((?: |\-)[0-9]{3}(?: |\-)[|\-)[0-9]{4})\([0-9]{7}))'

Mail : ^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$

```
type PhoneNumber : String(30) @assert.format: '((?:\\+|00)[17](?: |\\-)?|(
```

So here in the common.cds file I am added the PhoneNumber as it may be use in the multiple tables and I am added also a regular expression to maintain the format of my phone number

```
type Email       : String(255) @assert.format: '^([a-zA-Z0-9_\-\.]
```

So here in the common.cds file I am added the Emailas it may be use in the multiple tables and I am added also a regular expression to maintain my Email format

```
aspect Amount {

  CURRENCY_CODE : String(4);
  GROSS_AMOUNT  : AmountT;
  NET_AMOUNT    : AmountT;
  TAX_AMOUNT    : AmountT;
}
```

So here I have created a custom aspect called **Amount** in the common.cds file. So basically for reuse purpose I did this. So whenever I need all this field in my table, I will not add all of them manually, rather I will reuse my custom aspect**.**

To use the custom aspect first we need to import the custom aspect and after that we need to use the custom aspect in the table syntax is : **entity {entity name} : {aspects name}**

```
using {soumik.common} from './common';
```

```
entity purchaseorder : common.Amount {
```

```
//Create a amount format for reuse
type AmountT     : Decimal(15, 2);
```

So in almost every table there will be a amount field and the amount which I will put should a certain format. So that's why in the common.cds file I created a reuseable field **AmountT** so this format I will follow whenever I will use the amount. Decimal(15,2)

```
entity employee : cuid, temporal {

    nameFirst     : String(40);
    nameMiddle    : String(40);
    nameLast      : String(40);
    nameInitials  : String(40);
    sex           : common.Gender;
    language      : String(1);
    phoneNumber   : common.PhoneNumber;
    email         : common.Email;
    loginName     : String(12);
    currency      : Currency;
    salaryAmount  : common.AmountT;
    accountNumber : String(16);
    bankId        : String(8);
    bankName      : String(64)
}
```

So this is my employee table with all standard aspect and also the custom

Fields like **Gender, PhoneNumber, AmountT**.

I have added all the remaining tables also. And after that I redeploy **cds deploy --to sqlite:soumik.db**

```
type Currency : Association to sap.common.Currencies;
```

```
entity Currencies : CodeList {
  key code      : String(3) @(title : '{i18n>CurrencyCode}');
      symbol    : String(5) @(title : '{i18n>CurrencySymbol}');
      minorUnit : Int16     @(title : '{i18n>CurrencyMinorUnit}');
}
```

So in the employee table **currency** attribute value we have given a Standard aspect **Currency.** So this Currency aspect is build a association with the Currencies entity, which Primary key is code. That's why in the csv file instead of **currency** attribute name should be modified as **currency_code.**

Now I will run the command cds add data, So it will create csv file. So first it created employee csv file, because we added employee entity. And also additionally it created 2 csv file **sap.common-Currencies.csv sap.common-Currencies.texts.csv** automatically, because we added the Currency standard aspect that's why it created 2 csv file automatically

Built an OData Service.

So now inside the **srv** folder I will create a cds file name **CatalogService.cds** So inside this file I will expose my entities.

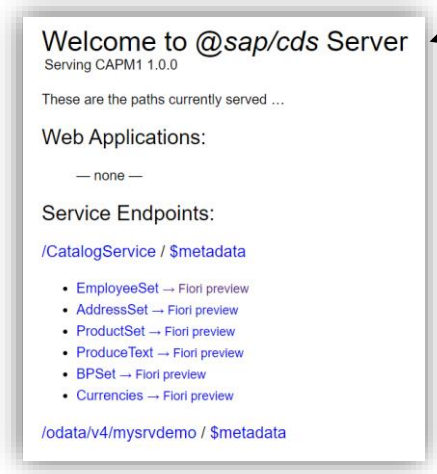
CatalogService.cds 1 ×

CAPM1 > srv > CatalogService.cds > {} CatalogService

```
using { soumik.db.master, soumik.db.transaction } from '../db/datamodel';


service CatalogService @(path:'/CatalogService'){

   entity EmployeeSet as projection on master.employee;
   entity AddressSet as projection on master.address;
   entity ProductSet as projection on master.product;
   entity ProduceText as projection on master.prodtext;
   entity BPSet as projection on master.businesspartner;

}
```

So here first I have import my data model file, because whatever data tables I have created that I am importing to create OData service.

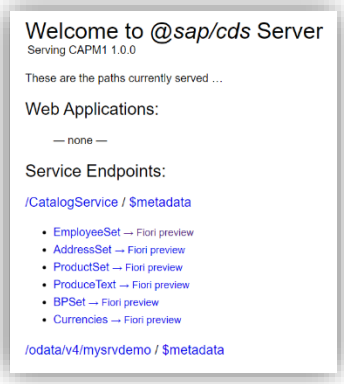
So now I will first expose the Master tables. So for that I will write the commands and also I am giving the new entity name which will show in the OData Service. After creation of all the Entity, Now I will run the command **cds run**



So now what are the entities I have exposed that all are showing in the OData service. And you noticed that while defining the Service I have added a annotation **@(path:'/CatalogService')** basically the use of this piece of code is whatever we have written inside the ' ' that will populate in the OData service. /CatalogService / $metadata

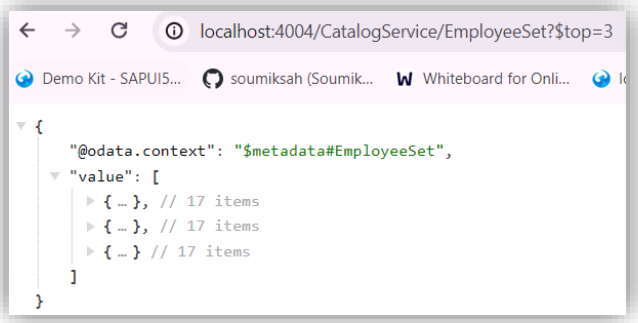
So if we did not use the annotation then it could show /odata/v4/catalog / $metadata . because after the **Catalog** when it see the new word **Service** then It ignore it.

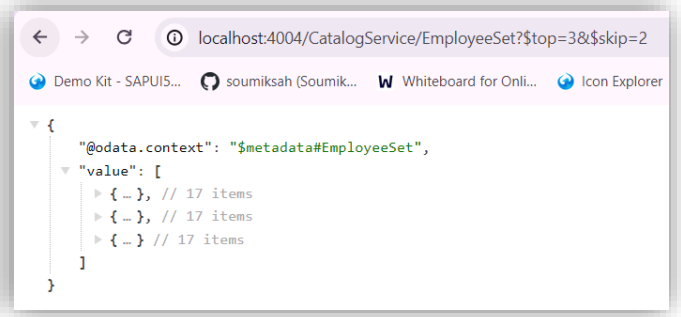
So to keep the whole Name as it is we will define it in the annotations.



So now if we access the Entity it will not show the data, because the Entity which we created It also creates the **View** in the database, since we did not deploy the code the View did not create, since view not created we will not be able to access the Entity, so for that reason we need to redeploy **cds deploy --to sqlite:soumik.db** after That we can able to access the data of the entity.

So in this OData I can also do the operation **top & skip**





The Standard aspect **temporal** has 2 attribute validForm & validTo and it expect the date and date format should be **YYYY-MM-DD,** so in the csv file we should give this format otherwise it will throw an error.

So OData supports following common operations:

Filtering: OData allows clients to filter data based on specific criteria using the **$filter** query option.
Example : **/EmployeeSet?$filter=bankName eq 'My Bank of San Francisco'**

Sorting: Sorting allows clients to retrieve data in a specified order using the **$orderby** query option.
Example : **/EmployeeSet?$orderby=nameFirst asc** or **/EmployeeSet?$orderby=salaryAmount desc**

Selecting: The **$select** query option allows clients to specify which properties of an entity to include in the response.
Example : /EmployeeSet?$select=sex,language so in this example it will show only this 2 attribute.

Top: The **$top** query option allows clients to limit the number of items returned in the result set.
Example : **/EmployeeSet?$top=5**

Skip: The **$skip** query option allows clients to skip a specified number of items in the result set.
Example : **/EmployeeSet?$skip=1**

Top & Skip : /EmployeeSet?$top=10&$skip=20 $top=10 specifies that you want to retrieve the top 10 items. $skip=20 specifies that you want to skip the first 20 items. This query will return items 21 to 30 from the **EmployeeSet.**
/EmployeeSet?$top=10&$skip=30 This query will return items 31 to 40 from the **EmployeeSet.**
/EmployeeSet?$top=10&$skip=40 This query will return items 41 to 50 from the **EmployeeSet.**

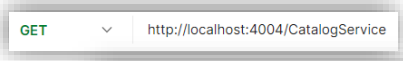
Count: Returns the count of entities matching a query using the **$count** query option.
Example **/EmployeeSet/$count**

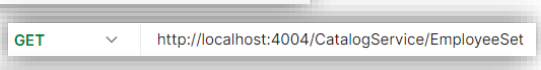
Expand: The **$expand** query option allows clients to retrieve related entities inline with the requested entities.
Example: **/ EmployeeSet?$expand=Customer**

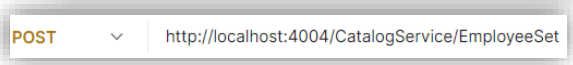
Search: the $search query is used to search the entity value
Example: **/EmployeeSet?$search=John**

Substring : To search for data using a substring in an OData query, you can use the **$filter** query option along with the **substringof** function.
Example : **/Products?$filter=substringof('phone', Name).** This query will return all products where the Name property contains the substring "phone".

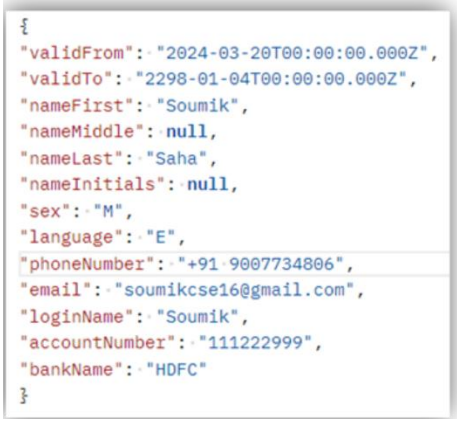
Insert some new data to my OData

Get all entities : 

Fetch Employee Data : 

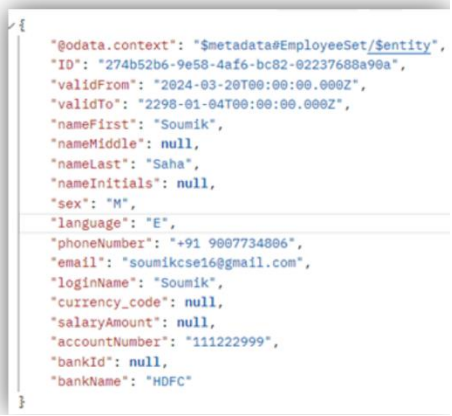
Create Employee : 

REQUEST



```
{
"validFrom": "2024-03-20T00:00:00.000Z",
"validTo": "2298-01-04T00:00:00.000Z",
"nameFirst": "Soumik",
"nameMiddle": null,
"nameLast": "Saha",
"nameInitials": null,
"sex": "M",
"language": "E",
"phoneNumber": "+91 9007734806",
"email": "soumikcse16@gmail.com",
"loginName": "Soumik",
"accountNumber": "111222999",
"bankName": "HDFC"
}
```

RESPONSE



```
{
  "@odata.context": "$metadata#EmployeeSet/$entity",
  "ID": "274b52b6-9e58-4af6-bc82-02237688a90a",
  "validFrom": "2024-03-20T00:00:00.000Z",
  "validTo": "2298-01-04T00:00:00.000Z",
  "nameFirst": "Soumik",
  "nameMiddle": null,
  "nameLast": "Saha",
  "nameInitials": null,
  "sex": "M",
  "language": "E",
  "phoneNumber": "+91 9007734806",
  "email": "soumikcse16@gmail.com",
  "loginName": "Soumik",
  "currency_code": null,
  "salaryAmount": null,
  "accountNumber": "111222999",
  "bankId": null,
  "bankName": "HDFC"
}
```

So here in the request payload I haven't pass the **currency_code bankId salaryAmount** because they are causing some issue, so without that I created a employee. And in the response you can notice that **ID** generated automatically, because we have use **cuid** aspect.

In the product table we will add a Column name **DESCRIPTION** and the value of the column would be ***localized String(250).*** So here a new table will be created automatically, and the name of the product will be **product_texts**. The reason behind the creation of a table automatically is we used localized. So in which table we will use this localized then a table will be created automatically and the name of the table will be **tableName_texts** and the field would be

```
CREATE TABLE soumik_db_master_product_texts (
  locale NVARCHAR(14) NOT NULL,
  NODE_KEY NVARCHAR(64) NOT NULL,
  DESCRIPTION NVARCHAR(250),
  PRIMARY KEY(locale, NODE_KEY)
);
```

So why we used localized: So I want that my product table DESCRIPTION attribute value adopt different language, so that when the language change it can show the description on that language. So for that we use the concept of localized **DESCRIPTION : localized String(250)**

So what is the meaning of localized here ?

So in my **product** table I have added a attribute called DESCRIPTION, which will give me the description of the product, now I want that this description can change based on different languages. So for that reason I will use the concept of localized. `DESCRIPTION : localized String(250);` Now after use this localized, it will automatically create a table name **product_texts** . So in the product_texts table there will be column name **locale, NODE_KEY, description.** And in the **locale** column we need to keep the different language code. and **description** column will write the description based on that language code. And here **NODE_KEY** column came from the product table, as we used NODE_KEY as a primary key in product table.

	NODE_KEY	locale	DESCRIPTION
1			
2	74867AD200E41EDBA5D8B06B26EB4052	de	Notebook Basic 15 r
3	74867AD200E41EDBA5D8B06B26EB6052	de	Notebook Basic 17 r
4	74867AD200E41EDBA5D8B06B26EB8052	de	Notebook Basic 18 r
5	74867AD200E41EDBA5D8B06B26EBA052	de	Notebook Basic 19 r

So the biggest advantage is before loading the description it will automatically fetch your browser language and that browser language it will compare with the **locale column** language-code [column of the product_texts] and if it is matches then it will fetch the data from the **DESCRIPTION** column of the product_texts table. If It does not match with the language code, then it will display the default **DESCRIPTION** which we maintain in the main table called **product.** So in the product table there is a column called **DESCRIPTION** so there we have maintained the description in English language that is default language.

And the changes we need to do in our project, in the csv file. So we will add the csv file **product_texts** and would be 3 column **locale, NODE_KEY, description**. In the locale we will mention the language-code. In the description column we will write the description based on the language code. And the product_texts table node_key value should match with the product table node_key value, then only it will work on different language. Now once all the changes done, we will deploy the code **cds deploy --to sqlite:soumik.db**

Now I want to give the proper readable name to my column name, that we can do with the help of **annotations.**

```
annotate businesspartner with{
    NODE_KEY @title : '{i18n>bp_key}';
    BP_ROLE @title : '{i18n>bp_role}';
    EMAIL_ADDRESS @title : '{i18n>email_address}';
    PHONE_NUMBER @title : '{i18n>phone_number}';
    FAX_NUMBER @title :'{i18n>fax_number}';
    WEB_ADDRESS @title :'{i18n>web_address}';
    ADDRESS_GUID @title : '{i18n>address_guid}';
    BP_ID @title : '{i18n>bp_id}';
    COMPANY_NAME @title : '{i18n>company_name}';

}
```

So here as you can see in the **businesspartner** table there are multiple columns but column name are not perfect for read. I want to give simple and meaning full column name, so for that I will use annotate.

i18n.properties

CAPM1 > db > i18n > i18n.properties

```
1  bp_key=Business Partner Guid
2  bp_role=Partner Type
```

Create a **i18n** folder, Where inside that there will be **i18n.properties** file. Where inside I will define the columns name.

So here I will also translate for German Language.

i18n_de.properties

CAPM1 > db > i18n > i18n_de.properties

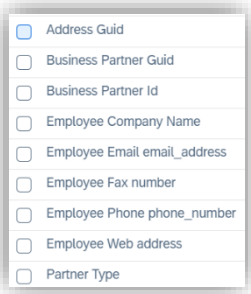
```
1  bp_key=Leitfaden für Geschäftspartner
2  bp_role=Partnertyp
```

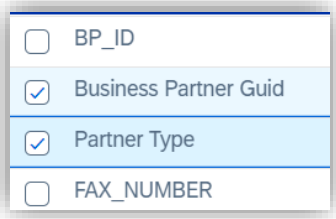
Now I will deploy the code, to reflect the changes. And I will again do CDS RUN. Now I will open the metadata and there I will see that the modified name of the column reflect in the metadata

```
<Annotations Target="CatalogService.BPSet/NODE_KEY">
    <Annotation Term="Common.Label" String="Business Partner Guid"/>
  </Annotations>
<Annotations Target="CatalogService.BPSet/BP_ROLE">
    <Annotation Term="Common.Label" String="Partner Type"/>
  </Annotations>
```

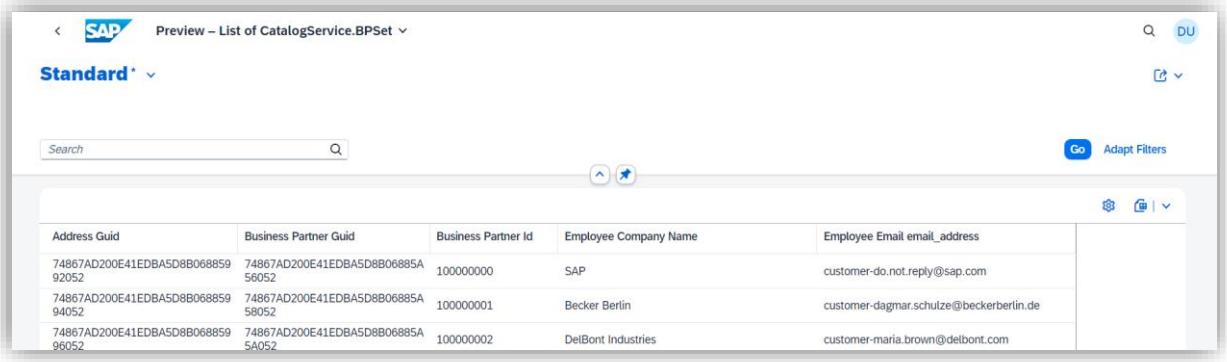
The modified name of the column name also will reflect in the fiori preview. If you go to the fiori preview of the **BPSet** you will see that columns.

```
annotate businesspartner with{
    NODE_KEY @title : '{i18n>bp_key}';
    BP_ROLE @title : '{i18n>bp_role}';
    EMAIL_ADDRESS @title : '{i18n>email_address}';
    PHONE_NUMBER @title : '{i18n>phone_number}';
    FAX_NUMBER @title :'{i18n>fax_number}';
    WEB_ADDRESS @title :'{i18n>web_address}';
    ADDRESS_GUID @title : '{i18n>address_guid}';
    BP_ID @title : '{i18n>bp_id}';
    COMPANY_NAME @title : '{i18n>company_name}';

}
```

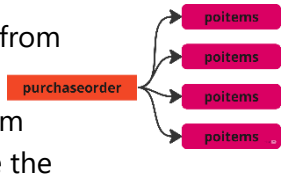




Now I will see the fiori preview of the **BPSet**



Now we will build the relationship between the **purchaseorder** & **poitems** table.



So here as we know that for one purchaseorder there may have multiple poitems. So from **purchaseorder** to **poitems** I want to build **1 to many relationship**. So if we delete the purchaseorder, there is no point in keeping the poitems. So that means from purchaseorder to poitems the relation should be tight couple, so anytime if we delete the purchaseorder it will automatically delete the poitems associated with the purchaseorder. So for that I will use composition from **purchaseorder---->poitems**. And if I delete any poitems it should not delete the purchaseorder, so for that I will use association from **poitems----->purchaseorder**

```
entity purchaseorder : common.Amount {
    key NODE_KEY         : Guid;
        PO_ID            : String(24);
        PARTNER_GUID     : Association to master.businesspartner;
        LIFECYCLE_STATUS : String(1);
        OVERALL_STATUS   : String(1);
        Criticality      : Integer;
        poitems          : Composition of many poitems
                               on poitems.PARENT_KEY = $self;
}
```

one to many relationship

many to one relationship

```
entity poitems : common.Amount {

    key NODE_KEY     : Guid;
        PARENT_KEY   : Association to purchaseorder;
        PO_ITEM_POS  : Integer;
        PRODUCT_GUID : Association to master.product;
}
```

```
entity purchaseorder : common.Amount {
    key NODE_KEY         : Guid;
        PO_ID            : String(24);
        PARTNER_GUID     : Association to master.businesspartner;
        LIFECYCLE_STATUS : String(1);
        OVERALL_STATUS   : String(1);
        Criticality      : Integer;
        poitems          : Composition of many poitems
                               on poitems.PARENT_KEY = $self;
}
```

So from the **purchaseorder** table we did one to one relationship with the **businesspartner** table, that means from **PARTNER_GUID** attribute we did a association to the primary key of the **businesspartner**.

And also from the **purchaseorder** table we did one to many relationship with the **poitems** table. In the purchaseorder table we have defined an attribute name poitems and from there we did a composition with the poitems table. And we have defined the on condition manually that means we have written a unmanaged association from **purchaseorder** table to **poitems** table. Now there is tight coupling from purchaseorder table to poitems table because of **Composition.** So now if I delete the purchaseorder then the poitems related to the purchaseorder will be deleted.

```
entity poitems : common.Amount {

    key NODE_KEY     : Guid;
        PARENT_KEY   : Association to purchaseorder;
        PO_ITEM_POS  : Integer;
        PRODUCT_GUID : Association to master.product;
}
```

So in this **poitems** table we did 2 managed associations from the 1st is PARENT_KEY to purchaseorder table. 2nd is PRODUCT_GUID to product, That means 1st we did one to one relationship from **poitems --->purchaseorder** table and 2nd we did one to one relationship from **poitems --->product** table. can achieve the association with the help of **expand** and my association attribute will be **PARENT_KEY & PRODUCT_GUID.**

So now for a purchaseorder I want see all the purchase items

So as you noticed that we have defined a one to many relationship from purchaseorder to poitems. So now if I want see for a particular purchaseorder what are all are my poitems, so for that I will go to the purchaseorder table and from there I will call the poitems.

```
port4004-workspaces-ws-9lzv9.us10.trial.applicationstudio.cloud.sap/CatalogService/purchaseorder?$expand=poitems
```


```
{
  "@odata.context": "$metadata#purchaseorder(poitems())",
  "value": [
    {
      "ID": "d0d02cc5-1f65-4e83-aa49-58c160b898dd",
      "CURRENCY_CODE": "EUR",
      "GROSS_AMOUNT": 3913.91,
      "NET_AMOUNT": 3289,
      "TAX_AMOUNT": 624.91,
      "PURCHASE_ORDER_ID": "300000000",
      "PARTNER_ID_ID": "2c57b324-c600-4526-9b3f-5fa406a7a566",
      "LIFECYCLE_STATUS": "N",
      "OVERALL_STATUS": "P",
      "Criticality": 0,
      "poitems": [
        { … }, // 8 items
        { … } // 8 items
      ]
    }
  ]
}
```


So here in the picture as you can see we got all the **poitems** for a particular **purchaseorder,** and also you have noticed that the poitems came as an array, because we did one to many relationship from purchaseorder to poitems.

Now for a particular purchase items I want to see what will be the purchase order and what will be the product details.

```
port4004-workspaces-ws-9lzv9.us10.trial.applicationstudio.cloud.sap/CatalogService/poitems?$expand=PRODUCT_ID,PARENT_KEY
```

What is CDS views ?

In SAP CAPM, CDS views are primarily used for defining the structure and behaviour of data. CDS views are defined using the CDS language. CDS views are designed such a way that we can represent of data from one or more underlying tables or other data sources. They can include calculations, filtering conditions, and associations between different entities.

So inside the DB folder I will create a file name **CDSViews.cds** after that inside that folder I am going to put a namespace **soumik.db** and I will export all my table which I created from **datamodel.cds** file Now I will define a context **CDSViews** and I will define a view which will expose a non normalize data of my purchase-order and purchase-items together. Now while defining view if we give view-name case sensitive then we will use a new option **![ ].** Now all the Attribute name I will modify and will give more meaningful name, with the help of new option **![ ]** Once I created the CDS view I will export the CDS view for that I will create new file **CDSServices.cds** inside the srv folder. And from there I will export the **POWorklist**

```
namespace soumik.db;

using { soumik.db.master, soumik.db.transaction } from './datamodel';

context CDSViews {

    define view![POWorklist] as
    select from transaction.purchaseorder{

        key PO_ID as ![PurchaseOrderId],
            PARTNER_GUID.BP_ID as ![PartnerId],
            PARTNER_GUID.COMPANY_NAME as ![CompanyName],
            GROSS_AMOUNT as ![POGrossAmount],
            CURRENCY_CODE as ![POCurrencyCode],
            LIFECYCLE_STATUS as ![POStatus],
        key Items.PO_ITEM_POS as ![ItemPosition],
            Items.PRODUCT_GUID.PRODUCT_ID as ![ProductId],
            Items.PRODUCT_GUID.DESCRIPTION as ![ProductName],
            PARTNER_GUID.ADDRESS_GUID.CITY as ![City],
            PARTNER_GUID.ADDRESS_GUID.COUNTRY as ![Country],
            Items.GROSS_AMOUNT as ![GrossAmount],
            Items.NET_AMOUNT as ![NetAmount],
            Items.TAX_AMOUNT as ![TaxAmount],
            Items.CURRENCY_CODE as ![CurrencyCode],


    }
}
```

CDSServices.cds

CAPM1 > srv > CDSServices.cds > {} CDSService

```
1   using { soumik.db.CDSViews } from '../db/CDSViews';
2
3   service CDSService @(path:'/CDSService'){
4
5       entity POWorklist as projection on CDSViews.POWorklist;
6
7   }
```


```
{
    "PurchaseOrderId": "300000000",
    "PartnerId": "100000087",
    "CompanyName": "South American IT Company",
    "POGrossAmount": null,
    "POCurrencyCode": "EUR",
    "POStatus": "N",
    "ItemPosition": 10,
    "ProductId": "HT-6110",
    "ProductName": "CD-RW, DVD+R/RW, DVD-R/RW, MPEG 2 (Video-DVD), MPEG 4, VCD, SVCD, DivX, Xvid
    ",
    "City": "Peking",
    "Country": "CN",
    "GrossAmount": 154.7,
    "NetAmount": 130,
    "TaxAmount": 24.7,
    "CurrencyCode": "ARS"
},
```

I will add some more CDS views

```
define view ProductValueHelp as
 select from master.product{

   @EndUserText.label:
   [
       {
           language: 'EN',
           text: 'Product ID'
       },
       {
           language: 'DE',
           text: 'Prodekt ID'
       }
   ]
   PRODUCT_ID as ![ProductId],
    @EndUserText.label:
   [
       {
           language: 'EN',
           text: 'Product Description'
       },
       {
           language: 'DE',
           text: 'Prodekt Description'
       }
   ]
   DESCRIPTION as ![Description]
 };
```

```
define view ![ItemView] as
  select from transaction.poitems{

  PARENT_KEY.PARTNER_GUID.NODE_KEY as ![Partner],
  PRODUCT_GUID.NODE_KEY as ![ProductId],
  CURRENCY_CODE as ![CurrencyCode],
  GROSS_AMOUNT as ![GrossAmount],
  NET_AMOUNT as ![NetAmount],
  TAX_AMOUNT as ![TaxAmount],
  PARENT_KEY.OVERALL_STATUS as ![POStatus]

  };

   define view ProductViewSub as
    select from master.product as prod {
        PRODUCT_ID as ![ProductId],
        texts.DESCRIPTION as![Description],
        (
            select from transaction.poitems as a{
                SUM(a.GROSS_AMOUNT) as SUM
            }
            where a.PRODUCT_GUID.NODE_KEY = prod.NODE_KEY
        ) as PO_SUM : Decimal(10, 2)
    };
```

```
define view ProductView as select from master.product
mixin{
    PO_ORDERS: Association[*] to ItemView on
                    PO_ORDERS.ProductId = $projection.ProductId
}
into{
    NODE_KEY as![ProductId],
    DESCRIPTION,
    CATEGORY as![Category],
    PRICE as![Price],
    TYPE_CODE as![TypeCode],
    SUPPLIER_GUID.BP_ID as![BPId],
    SUPPLIER_GUID.COMPANY_NAME as![CompanyName],
    SUPPLIER_GUID.ADDRESS_GUID.CITY as![City],
    SUPPLIER_GUID.ADDRESS_GUID.COUNTRY as![Country],
    //Exposed association, which means when someone read the view
    //the data for orders wont be ready by default
    //until unless someone consume the association
    PO_ORDERS

};

define view CProductValuesView as
    select from ProductView{

        ProductId,
        Country,
        PO_ORDERS.CurrencyCode as ![CurrencyCode],
        sum(PO_ORDERS.GrossAmount) as ![POGrossAmount]: Decimal(10, 2)
    }
    group by ProductId,Country,PO_ORDERS.CurrencyCode
```

```
service CDSService @(path:'/CDSService'){

    entity POWorklist as projection on CDSViews.POWorklist;
    entity ProductOrders as projection on CDSViews.ProductViewSub;
    entity ProductAggregation as projection on CDSViews.CProductValuesView;


}
```

/CDSService / $metadata

- POWorklist → Fiori preview
- ProductOrders → Fiori preview
- ProductAggregation → Fiori preview

localhost:4004/CDSService/ProductOrders

Demo Kit - SAPUI5... | soumiksah (Soumik... | Whiteboard for Onli... | Icon Explorer | Flipkart | ODATA

Pretty-print


```
{
    "@odata.context": "$metadata#ProductOrders",
    "value": [
        {
            "ProductId": "HT-1000",
            "Description": "Notebook Basic 15 con 2,80 GHz de cuatro n◆cleos, pantalla LCD de 15 \"
            "PO_SUM": 17
        },
        {
            "ProductId": "HT-1000",
            "Description": "Notebook Basic 15 mit 2,80 GHz Quad-Core, 15 \"LCD, 4 GB DDR3-RAM, 500
            "PO_SUM": 17
        },
        {
```


localhost:4004/CDSService/ProductAggregation

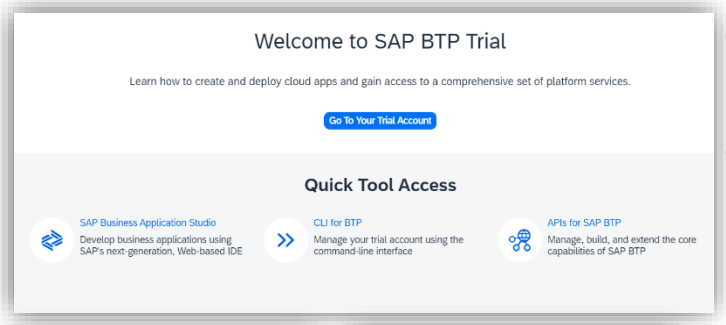
Demo Kit - SAPUI5... | soumiksah (Soumik... | Whiteboard for Onli...

Pretty-print

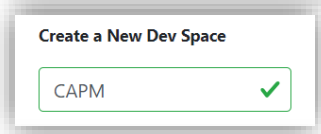

```
{
    "@odata.context": "$metadata#ProductAggregation",
    "value": [
        {
            "ProductId": "74867AD200E41EDBA5D8B06B26EB4052",
            "Country": "BR",
            "CurrencyCode": "EUR",
            "POGrossAmount": 17
        },
        {
            "ProductId": "74867AD200E41EDBA5D8B06B26EB6052",
            "Country": "CA",
            "CurrencyCode": "EUR",
            "POGrossAmount": 56
        },
```

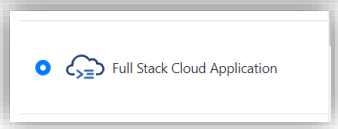
SAP BAS [Business Application Studio]

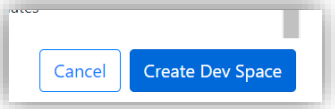
First we will go to the trial account : https://account.hanatrial.ondemand.com/trial/#/home/trial and there you will see the BAS, and there I will click and will redirect to the BAS. Now I will click on the



Create Dev Space and then I will choose **Full Stack Cloud Application** and I will choose the Dev-Space name. So I named it **CAPM.** Now I will click **Create Dev Space.**







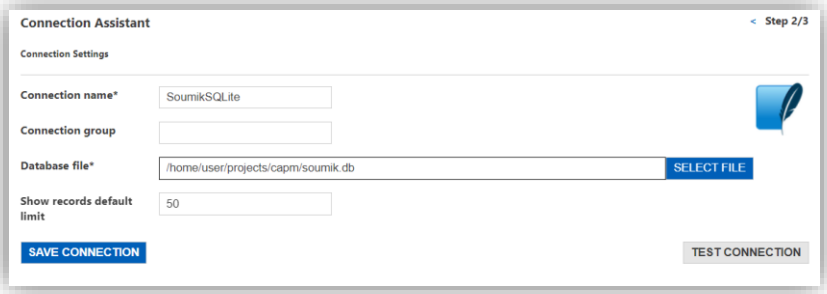
Now I will work with the BAS [I will take entire development into the BAS]

Now I will go to the BAS there already created dev space named **CAPM,** I will go there and now I will create a Project for creating the project I will go to the terminal and I should be in the Project directory. `user: projects $` Now I will create a new directory **capm.** `user: projects $ mkdir capm` And now I will go inside the **capm** and there I will initialize a capm project by running the command **cds init**

```
user: projects $ cd capm/
user: capm $ cds init
```

after that I will install sqlite**. npm install sqlite3 --save-dev.**
Now we will do **npm install,** so that system will install all the necessary dependencies. Now from the local all the files which we will created will copy and paste it to BAS in the **capm** project. Once copy all the files in the **capm** project of BAS, now we will deploy to sqlite **cds deploy --to sqlite:soumik.db**

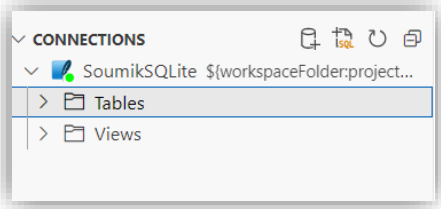
So in BAS, there is a amazing tool which is called **sql tools**  so click on this button and after that click on **CONNECTIONS** you can add connection to SQLite in the graphical mode. So click on **Add New Connection** from there select **SQLite** and give the **connection-name** , **Database file name**



So here I have given the Connection Name **SoumikSQLite** & Database file name : **/home/user/projects/capm/soumik.db**
So this I got from

```
user: capm $ pwd
/home/user/projects/capm
```

So once fill all this things I will click on the **TEST CONNECTION** It will show Successfully connected! and after that I will click on **SAVE CONNECTION** after that we will see all the tables and views after save the connection.



So you can go inside the **Views** and **Tables** and you can show the Table records.

There is a very interesting thing which I love the most about BAS that is you can develop, deploy tables, structures, views in Graphical mode. So in the **capm** Project go inside the **db** folder. There you will see the cds file. **datamodel.cds** So right click on the cds file and open with the **CDS Graphical Modeler** and you will see a dropdown which will initially be your database name :**soumik.db** from the dropdown choose your table **soumik.db.master** or **soumik.db.transaction** there you will see the graphical mode of the table which is created and also you can see the association between the tables graphically. And there is a button called **Add Entity.** Where you can And new Table graphically. And also you can add Property on that table.

Test or Run the Application in the Bas tool

So now I am running my application in the BAS tool. So now I want to do CRUD operations with my entities, so for that which url I am getting with that I will not be able to do operations in postman, because postman does not support this url. So that means in the bas only we have to do operations.. So for that in our project we will go inside the **srv** folder and create a file with **.http** extension. So I created a file name **test.http** and inside this I will do the operations.

```
Send Request
GET http://localhost:4004/CatalogService/
```

So this is the get request for the Catalog-Service

```
###
Send Request
GET http://localhost:4004/CatalogService/$metadata
```

So this is the get request for the Service Metadata. So we have to provide 3 hash **(#)** to show the send request.

```
###
Send Request
GET http://localhost:4004/CatalogService/EmployeeSet?$top=2
```

Get the First 2 employee.

```
Read Purchase Order
Send Request
http://localhost:4004/CatalogService/POs?$top=2&$expand=Items,PARTNER_GUID
```

So this is the get request of the purchase order and Also we are doing the association with the help of expand.

```
### Create new Employee
Send Request
POST http://localhost:4004/CatalogService/EmployeeSet
Content-Type: application/json

{
"validFrom": "2024-03-20T00:00:00.000Z",
"validTo": "2298-01-04T00:00:00.000Z",
"nameFirst": "Soumik",
"nameMiddle": null,
"nameLast": "Saha",
"nameInitials": null,
"sex": "M",
"language": "E",
"phoneNumber": "+91 9007734806",
"email": "soumikcse16@gmail.com",
"loginName": "Soumik",
"accountNumber": "111222999",
"bankName": "HDFC"
}
```

So here I have created an Employee with the POST request. Where we have to give the **Content-Type** and we have to pass the Request body.

Now I want that the attribute **nameFirst, namLast** attribute can not be null that means while creating the employee we should pass the value of those attributes, otherwise it will not create the employee. It will throw the error. For that I will modify the employee entity.

```
entity employee : cuid, temporal {

nameFirst: String(40) not null;
nameMiddle: String(40);
nameLast: String(40) not null;
nameInitials: String(40);
sex: common.Gender;
language: String(1);
phoneNumber: common.PhoneNumber ;
email: common.Email ;
loginName: String(12);
currency: Currency;
salaryAmount: common.AmountT;
accountNumber: String(16);
bankId: String(20);
bankName: String(64)
}
```

So here as you can see the 2 attributes I have set as **not null.** And after that I will deploy the code by run the command **cds deploy --to sqlite:soumik.db** And now while creating the employee if don't pass the attribute, it will throw error.

Now while creating the employee I want that The phone-number and email of the employees can not be duplicate, it should be unique. So for that I will write some code. I want that the **email, phoneNumber** attribute should be different for different employees.

```
@assert.unique: {
    phoneNumber: [phoneNumber],
    email      : [email]
}
entity employee : cuid, temporal {

    nameFirst     : String(40) not null;
    nameMiddle    : String(40);
    nameLast      : String(40) not null;
    nameInitials  : String(40);
    sex           : common.Gender;
    language      : String(1);
    phoneNumber   : common.PhoneNumber;
    email         : common.Email;
    loginName     : String(12);
    currency      : Currency;
    salaryAmount  : common.AmountT;
    accountNumber : String(16);
    bankId        : String(20);
    bankName      : String(64)
}
```

So this code I have added. So this code tells that the phone-number and email cannot be duplicate, It should be unique otherwise it will throw the error **"Entity already exists"**

So the **nameFirst, nameLast** may be common to multiple entity. So rather write the same attribute in multiple table I will create a custom aspect and I will reuse them. So I will create a custom aspect now. I will create the custom aspect in the **common.cds** file.

```
aspect fullName {
    nameFirst  : String(40) not null;
    nameMiddle : String(40);
    nameLast   : String(40) not null;
}
```

So I created a custom aspect name **fullName** now I will use this custom aspect to the employee table.

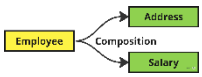
```
entity employee : cuid, temporal, common.fullName {

    nameInitials  : String(40);
    sex           : common.Gender;
    language      : String(1);
    phoneNumber   : common.PhoneNumber;
    email         : common.Email;
    loginName     : String(12);
    currency      : Currency;
    salaryAmount  : common.AmountT;
    accountNumber : String(16);
    bankId        : String(20);
    bankName      : String(64)
}
```

So here I have used the custom aspect.

Association & Composition

Suppose I have some tables **Employee, Address, Salary, Designation.** So now I want to built a relationship between these tables. Think about if I removed an employee then there is not point in keeping the Address, Salary of an employee. And We should keep the designation after delete the employee, because there are may be other employee with same designation. So I will use composition and association.

Composition : It is **tight coupling**. So if there is a composition from **Employee to Address & Salary**, then If I delete the Employee then the Salary and the Address of the employee would get deleted automatically. It can be achieved with the help of composition. For doing the composition we use **of**



Association : It means **loose coupling**. So if there is a association from **Employee to Designation**, then If I delete the Employee then the Designation of the employee would not get deleted. For doing the association we use **to**

```
entity Employee : cuid, managed {

  fName           : String;
  lName           : String;
  gender          : String;
  DOB             : Date;
  age             : Integer;
  contractStarted : Date;
  email           : types.Email not null;
  phone           : types.PhoneNumber not null;
  address         : Composition of many Address
                      on address.employee = $self;
  salary          : Composition of Salary;
  department      : Association to Department;
  designation     : Association to Designation;
}
```

So after doing association and composition the column name would be **salary_ID, department_ID, designation_ID** so here salary, department, designation are the foreign key and the ID is the primary key of those table to which this foreign key built the relationship. So to do the navigation we use `?$expand=association_name` And one thing we should remember the foreign key value should be same with the primary key of the table.

So in the employee table the foreign key **salary** which built an composition with **ID** which is a primary key of the Salary table. So in the data file the foreign key would be renamed to **salary_ID** and its value would be same as the ID of the Salary table. So that we can easily navigate. Similar would be happen with **department & designation**

So in the above example we have seen that `department : Association to Department;` so here an Employee has only one department that's why we have write this. Or another way we can write this means **one to one relationship.** If you don't mention **one** by default it takes one `department : Association to one Department;`

we can do vice versa also, that means from the Department table also we can built relation to the Employee table.

```
Employee    : Association to one Employee
                on Employee.address = $self
```

Here we have written unmanaged association

`salary : Composition of Salary;` In this example you can see that the relation between the employee and the salary is composition. That means it is a tight coupling. So if I delete the employee the salary also will be deleted automatically. Suppose instead of the Composition it is a Association then after delete also the employee the salary will not be deleted. So this is the difference between the Composition and Association.

we can do vice versa also, that means from the Salary table also we can built relation to the Employee table.

```
Employee    : Association to one Employee
                on Employee.address = $self
```

Here we have written unmanaged association

And there is an one more advantage of Composition over the Association. So if we don't expose the Salary entity but still there is an composition between the Employee and the Salary. So, we can access the Salary through the **?$expand=salary.** But we can not directly access the Salary table.

But if the same case happen with the Association. So in that case we can not access the Salary table through the **?$expand=salary** and also we can not directly access the salary table. `salary : Association to Salary;`

Building a one to many relationship using Composition

So there is a employee table and a address table. So in one employee may be there is multiple address. So I want to build relationship between the employee and the address.

Name	Gender	Age	Email	Address_ID
Soumik	Male	26	soumik@123	123
Soumik	Male	26	soumik@123	456
Soumik	Male	26	soumik@123	789

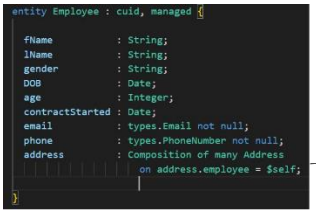
ID	Address	Pin code
123	Santir Bazar	799144
456	Marathalli	560037
789	Jigani	560105

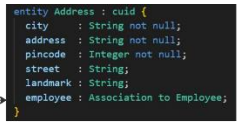
So here as you can see for a employee there are 3 address and we connected through the Address_ID for that we are duplicating the entire row, so we will avoid this relation.

ID	Name	Gender	Age	Email
123	Soumik	Male	26	soumik@123

ID	Address	Pin Code	Employee_ID
123	Santir Bazar	799144	**123**
456	Marathalli	560037	**123**
789	Jigani	560105	**123**

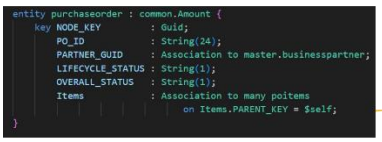
So here also we did the relationship but instead of duplicating the the entire row we only duplicate a column





So here we built a one to many relationship from **Employee** to **Address** table that menas one employee have multiple address. And also we write the condition.

So now as you can see here we built a one to many relationship and One employee having multiple address . So if we delete the employee then the address related to the employee will be deleted automatically as it is **Composition.** So in the Employee table the address attribute is a unmanaged association.



one to many relationship

So here In a purchaseorder there can be multiple poitems. So that's why we are building the one to many relationship. And the Items are the unmanaged association.

So similarly we are created another one to many relationship between the purchaseorder and poitems. And we did association. And here if we delete any purchaseorder the poitems which is associated with the purchaseorder will not be deleted as we did **Association**

So from one entity, how we can fetch multiple entity?

First of all relation should be built between the entities.

port4004-workspaces-ws-9lzv9.us10.trial.applicationstudio.cloud.sap/CatalogService/POItems/74867AD200E41EDBA5D8B0CBD35A0052?$expand=PARENT_KEY,PRODUCT_GUID


```
{
  "@odata.context": "$metadata#POItems(PARENT_KEY(),PRODUCT_GUID
  "CURRENCY_CODE": "ARS",
  "GROSS_AMOUNT": 154.7,
  "NET_AMOUNT": 130,
  "TAX_AMOUNT": 24.7,
  "NODE_KEY": "74867AD200E41EDBA5D8B0CBD35A0052",
  "PARENT_KEY_NODE_KEY": "74867AD200E41EDBA5D8B0C98DC28052",
  "PO_ITEM_POS": 10,
  "PRODUCT_GUID_NODE_KEY": "74867AD200E41EDBA5D8B06B26F60052",
  "PARENT_KEY": {…}, // 9 items
  "PRODUCT_GUID": {…} // 16 items
}
```


Convert ODATA V4 ----------> ODATA V2

So the service which are generated by the CAPM it is come as **ODATA V4** version. But sometimes we need the **ODATA V2** version also, because some element does not support **ODATA V2.** So for that we can convert the V4 to V2 as well. For that we need to download a node module called odata-v2-adapter-proxy (cov2ap). The command for download is `npm i @sap/cds-odata-v2-adapter-proxy` after install I will create a **server.js** file in the **srv** folder. And in that file I will copy the piece of code, which I will get from the page where I got the node module. After install the module and after adding the code I will deploy the code **cds deploy --to sqlite:soumik.db**

Before convert to V4 ----------->

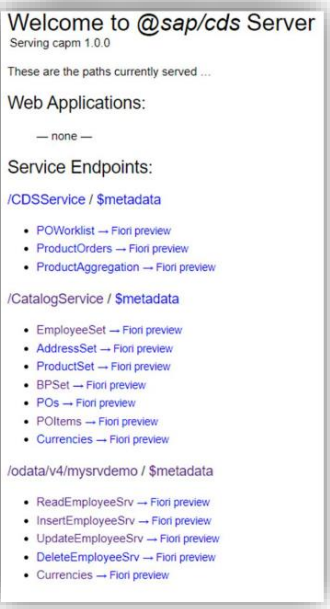
server.js

capm > srv > server.js > ...

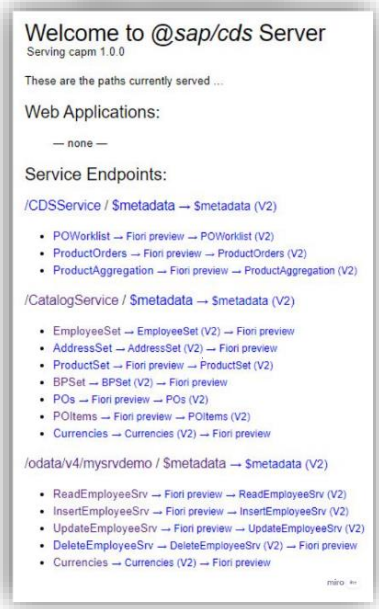
```
1  const cds = require("@sap/cds");
2  const cov2ap = require("@sap/cds-odata-v2-adapter-proxy");
3  cds.on("bootstrap", (app) => app.use(cov2ap()));
4  module.exports = cds.server;
```

So in this code we have imported 2 modules one is **cds** and another is **v2-adopter-proxy**. And we have created a event call **bootstrap** and to handle this event we have written an event-handler. So before starting the application we do bootstrap, while doing the bootstrap this bootstrap event will be called and inside that the callback function will execute. And that callback function will convert the V4 model to V2 model

Before Convert to V2



After Convert to V2



So here in the picture you can see after convert V2 both V2 and V4 are available. V4 was by default. So now to access the V2 we have to provide **/V2** in the URL

Access the **EmployeeSet** in both V2 and V4

V4: port4004-workspaces-ws-9lzv9.us10.trial.applicationstudio.cloud.sap/CatalogService/EmployeeSet

V2: https://port4004-workspaces-ws-9lzv9.us10.trial.applicationstudio.cloud.sap/v2/CatalogService/EmployeeSet

So previously we faced some problem during the creation of employee, so I have made some changes. So in the employee table there is a attribute **bankId** so there previously the size was **String(8)** and I have changed to **String(20).** And there is a attribute **currency** in the employee table and there we have used **Currency** aspect. And this **Currency** aspect contains 4 fields code, symbol, name, descr. So while doing POST, PUT, PATCH if we pass the Currency attribute then we would need the Currency table maintained properly with the Demo data. So we will create a csv file of currency with all the 4 fields. So when we will insert data in the currency field, at that time it will match the data with the demo data, which is stored in the csv file.

```
### Read the data of the employee which is created above
Send Request
GET http://localhost:4004/CatalogService/EmployeeSet/cf0dece5-2824-48d9-9121-31ab3203a574
```

So here I am fetching the employee with the help of Employee-Id

What is the difference between PUT & PATCH ?

So **PUT** means update everything. All the field it will update. Basically complete update method that replaces an existing record with a new one. While doing PUT if you don't pass any attribute, then it will consider attribute value as NULL.

PATCH — means partial update method. While doing PATCH if you don't pass any attribute, then it will consider existing attribute value only.

Update the Employee details using PUT.

```
### Update the salary for the employee
Send Request
PUT http://localhost:4004/CatalogService/EmployeeSet/56e0a289-86da-49d5-a09c-fba17259c7ba
Content-Type: application/json; IEEE754Compatible=true

{
    "validFrom": "2024-03-20T00:00:00.000Z",
    "validTo": "2298-01-04T00:00:00.000Z",
    "salaryAmount" : "22478.00",
    "currency_code" : "INR"
}
```


```
{
  "@odata.context": "$metadata#EmployeeSet/$entity",
  "ID": "56e0a289-86da-49d5-a09c-fba17259c7ba",
  "validFrom": "2024-03-20T00:00:00.000Z",
  "validTo": "2298-01-04T00:00:00.000Z",
  "nameFirst": null,
  "nameMiddle": null,
  "nameLast": null,
  "nameInitials": null,
  "sex": null,
  "language": null,
  "phoneNumber": null,
  "email": null,
  "loginName": null,
  "currency_code": "INR",
  "salaryAmount": 22478,
  "accountNumber": null,
  "bankId": null,
  "bankName": null
}
```


So here as you can see I updated the employee details using PUT, but after update in the response you can see that whatever we have passed in the request-body that means **validFrom, validTo, salaryAmount, currency_code** that value got updated in the response other attributes value is showing **null.** And one this we need to remember that while doing PUT or PATCH we have to pass `IEEE754Compatible=true` in the Content-type, otherwise it will throw error. And before doing the PUT one thing again we have to make sure that in the request body we are passing the **currency_code** and in the table for this **currency_code** you have passed the **Currency** aspect. And this Currency aspect contains 4 fields *code, symbol, name, descr*. So while doing POST, PUT, PATCH if we pass the **currency_code** attribute in the request body then we would need the Currency table maintained properly with the Demo data. So we will create a csv file named **sap.common-Currencies** with all the 4 fields. So when we will insert data in the **currency_code** field, at that time it will match the data with the demo data, which is stored in the csv file.

sap.common-Currencies.csv

capm > db > csv > sap.common-Currencies.csv

```
code;symbol;name;descr;
EUR;€;Euro;European Euro
USD;$;US Dollar;United States Dollar
CAD;$;Canadian Dollar;Canadian Dollar
AUD;$;Australian Dollar;Australian Dollar
GBP;£;British Pound;Great Britain Pound
ILS;₪;Shekel;Israeli New Shekel
INR;₹;Rupee;Indian Rupee
QAR;﷼;Riyal;Katar Riyal
SAR;﷼;Riyal;Saudi Riyal
JPY;¥;Yen;Japanese Yen
CNY;¥;Yuan;Chinese Yuan Renminbi
```

Update the Employee details using PATCH

```
### Update the salary of the employee using PATCH
Send Request
PATCH http://localhost:4004/CatalogService/EmployeeSet/77c6a2e9-8217-41d9-898b-7805897cbce3
Content-Type: application/json; IEEE754Compatible=true

{
    "salaryAmount" : "22478.00",
    "currency_code" : "INR"
}
```


```
{
  "@odata.context": "$metadata#EmployeeSet/$entity",
  "ID": "77c6a2e9-8217-41d9-898b-7805897cbce3",
  "validFrom": "2024-03-20T00:00:00.000Z",
  "validTo": "2298-01-04T00:00:00.000Z",
  "nameFirst": "Soumik",
  "nameMiddle": null,
  "nameLast": "Saha",
  "nameInitials": null,
  "sex": "M",
  "language": "E",
  "phoneNumber": "+91 9007734806",
  "email": "soumikcse16@gmail.com",
  "loginName": "Soumik",
  "currency_code": "INR",
  "salaryAmount": 22478,
  "accountNumber": "111222999",
  "bankId": null,
  "bankName": "HDFC"
}
```


So here I have updated the **salaryAmount, currency_code** using PATCH and PATCH is a partial update, so the attribute which we will not update the value of that attribute will be remain as it is.

Capabilities Annotation

Suppose there is a requirement from the client, where you want to restrict somebody to do Update or Somebody to do a insert or Somebody to do a delete on an entity, That we can handle with the help of Capabilities.

@readonly : If we use this capability, that means we can only read, but we can not do any operations like **create, update, delete.** If you do create, update, delete you will get error.

@Capabilities: {Insertable, Updatable : false, Deletable, Readable} : the meaning of this capabilities is you can insert, delete, read, but you **can not update**.

```
@Capabilities: {Insertable,Updatable : false,Deletable,Readable}
entity EmployeeSet as projection on master.employee;
entity AddressSet as projection on master.address;
entity ProductSet as projection on master.product;
//  entity ProduceText as projection on master.prodtext;
entity BPSet as projection on master.businesspartner;
```

In this case the **Capabilities** are specified outside of all entity definition. This suggests that the capabilities apply globally to all entities. Therefore, the capabilities specified (Insertable, Updatable, Deletable, Readable) apply to all entities.

Another way we can add capabilities for a particular entity. So for that I will create a **.cds** file inside the **srv** folder. And there I will write the annotations for a particular entity. Eg: **EmployeeSetAnnotations.cds**

Example : Suppose for the EmployeeSet entity I want to add some capabilities. So for that I will create a file inside the srv folder name **EmployeeSetAnnotations.cds** and inside that we will write capabilities for the particular entity.

EmployeeSetAnnotations.cds

capm > srv > EmployeeSetAnnotations.cds

```
using {CatalogService} from './CatalogService';

annotate CatalogService.EmployeeSet with @(Capabilities: {

    InsertRestrictions: {
        $Type     : 'Capabilities.InsertRestrictionsType',
        Insertable: true
    },

    UpdateRestrictions: {
        $Type    : 'Capabilities.UpdateRestrictionsType',
        Updatable: true
    },
    DeleteRestrictions: {
        $Type    : 'Capabilities.DeleteRestrictionsType',
        Deletable: true
    },
    ReadRestrictions  : {
        $Type    : 'Capabilities.ReadRestrictionsType',
        Readable: true
    },

});

```

So here in this picture you can see that inside the srv folder we have created cds file for employee entity. That means whatever operations are allowed for the employee entity we will define here.

Generic Handlers

The CAPM framework offers us **CURDQ** operations on the entity. Suppose now I want to do below operations :

1. I want to take full control on CURDQ
2. I want to add my custom validations before the data is updated
3. Before data is save in the database we want to change the data with enrichment.
4. Before read the data we want to add extra information to the data.

All this things we can achieve with the help of **Event Handlers.** And which is provided by the CAPM by default is **Generic Handlers.**

Event Handling

Your company is planning to build an extension application using the SAP Cloud Application Programming Model (CAP). The generic service handlers that the framework provides for standard CRUD operations (CREATE, READ, UPDATE, DELETE) do not fully satisfy the application's requirements. You want to implement custom business logic on top of the standard functionality. For that, you need to understand the concept of event handlers in CAP.

So basically CAPM provided us 3 events **on, before, after**

So when user send the request it first goes to the **before** event, then from before it goes to **on** event and last it goes to the **after** event and after that the results will go back to the user.

before : Suppose if you want to validate data or enrich data before the insert or post happens then you can go with before events. In this event no database operations happens like CRUDQ.

on : After validate or enrich the data from the before event, the data comes to the on event. So basically in on we do the database operations. Basically all the CRUDQ operations are happen in the on event.

after : Once all the database operations done and data saved in the on event, then it comes to the after event. So basically if I want to show some extra data before presented to the user, then I will go with the after event.

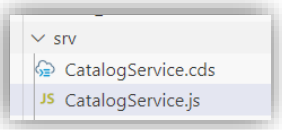
So in the **before** event what you will be changing that will be saved in the database using **on** event. So generally in very rare case we use this **on** event. In 99% case we don't use the **on** event. So we do all our operations in before and after only.

How to Register Event Handlers

So here we can choose Option 1 or Option 2 , but the Option 1 is recommended.

Option 1

In this method, the JavaScript file is placed next to the CDS (.cds) file used to define the service, the JavaScript file needs to have the same name as the .cds file, this way the framework hooks up the implementation to the service file.



Option 2

Here we set the link through the impl annotation in your CDS model file (.cds), where the respective service implementation can be found. This is useful if you have diverging file names or you want to make it very explicit that the two files belong together.

```
service CatalogService@(path:'/CatalogService', impl:'MyCode.js') {
```

So now I will create an employee, So I want that before create the employee the before event trigger and after create the employee the after event trigger.

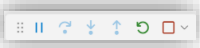
```
const cds = require("@sap/cds")

module.exports = cds.service.impl(async (srv)=>{
    //All your implementation
    srv.before("CREATE", "EmployeeSet", async (req)=>{

        console.log("Before event is triggered")
    })

    srv.after("CREATE", "EmployeeSet", async (req)=>{

        console.log("After event is triggered")
    })
})
```

So here I have written the before and after event. So before create the employee the before event will be triggered and it will print the statement. And after employee create after event will be triggered and it will print the statement. While debug

```
[odata] - POST /CatalogService/employee
Before event got triggered
After event got triggered
```

How to debug the JavaScript file in BAS

So to debug the JavaScript file in the bas. First we need to terminate the port if running in the terminal. Then we will click on the  button. Then after that we will click 

start debugging button. And once it start in the debug console it will show the port number. We can add breakpoint in our code to see the flow. We can close the debug  by click on the stop button.

Now I want that while creating the employee if I give a big salary amount, then it will throw error.

```
const cds = require("@sap/cds")

module.exports = cds.service.impl(async (srv) => {
    //All your implementation
    srv.before("CREATE", "EmployeeSet", async (req) => {

        if (parseFloat(req.data.salaryAmount) >= 1000000) {
            req.error(500, "Salary must be below 1000000")
        }

    })

})
```

So here as you can see that we have written a condition that while creating the employee if the salary of the employee exceeds 1000000 then will throw an error which status code is 500 and throw an error message.

I want that while update the employee if I give a big salary amount, then it will throw error.

So already in the code we have written a condition that while create the employee if the salary exceeds more than 1000000 then it will throw error. Now I want to write same condition but in different condition, that is while update the employee. So instead of write new condition again an again we will do modify in our existing code, so that code will work for both create and update.

```
const cds = require("@sap/cds")

module.exports = cds.service.impl(async (srv) => {
    //All your implementation
    srv.before(["CREATE","UPDATE"], "EmployeeSet", async (req) => {

        if (parseFloat(req.data.salaryAmount) >= 1000000) {
            req.error(500, "Salary must be below 1000000")
        }

    })

})
```

So here as you can see that we modified our code, so basically in the **before** event we have mentioned an array where we have written CREATE, UPDATE that means while create or update the employee the before event will be triggered.

```
Send Request
POST http://localhost:4004/CatalogService/EmployeeSet
Content-Type: application/json

{
"validFrom": "2024-03-20T00:00:00.000Z",
"validTo": "2298-01-04T00:00:00.000Z",
"salaryAmount" : "1000000.00",
"nameFirst": "Soumik",
"nameMiddle": null,
"nameLast": "Saha",
"nameInitials": null,
"sex": "M",
"language": "E",
"phoneNumber": "+91 9007734806",
"email":"soumikcse16@gmail.com",
"loginName": "Soumik",
"accountNumber": "111222999",
"bankName": "HDFC"
}
```

```
{
  "error": {
    "code": "500",
    "message": "Salary must be below 1000000",
    "@Common.numericSeverity": 4
  }
}
```

```
### Update the salary, validation date for the employee using PUT
Send Request
PUT  http://localhost:4004/CatalogService/EmployeeSet/7519ea02-10d
Content-Type: application/json; IEEE754Compatible=true

{
    "validFrom": "2024-03-20T00:00:00.000Z",
    "validTo": "2298-01-04T00:00:00.000Z",
    "salaryAmount" : "1000000.00",
    "currency_code" : "INR"
}
```

miro

I want that while creating the employee I will pass DOB and it will calculate the age and will give in the response with the help of before event.

So for that I will add DOB, AGE field in the employee entity and I will add those field in the employee csv file. And after when creating a new employee we will pass the DOB and AGE value will pass **0** and in the before event we will write our logic. So based on the DOB it will calculate the AGE and it will show in the response.

```
{
"validFrom": "2024-03-20T00:00:00.000Z",
"validTo": "2298-01-04T00:00:00.000Z",
"salaryAmount" : "100000.00",
"nameFirst": "Soumik",
"nameMiddle": null,
"nameLast": "Saha",
"nameInitials": null,
"DOB" : "1998-03-15",
"AGE" : 0,
"sex": "M",
"language": "E",
"phoneNumber": "+91 9007734806",
"email":"soumikcse16@gmail.com",
"loginName": "Soumik",
"accountNumber": "111222999",
"bankName": "HDFC"
}
```

```
{
  "@odata.context": "$metadata#EmployeeSet/$entity",
  "ID": "b3baf055-775b-4f4e-bf45-4ecc8b31a0b1",
  "validFrom": "2024-03-20T00:00:00.000Z",
  "validTo": "2298-01-04T00:00:00.000Z",
  "nameFirst": "Soumik",
  "nameMiddle": null,
  "nameLast": "Saha",
  "nameInitials": null,
  "DOB": "1998-03-15",
  "AGE": 26,
  "sex": "M",
  "language": "E",
  "phoneNumber": "+91 9007734806",
  "email": "soumikcse16@gmail.com",
  "loginName": "Soumik",
  "currency_code": null,
  "salaryAmount": 100000,
  "accountNumber": "111222999",
  "bankId": null,
  "bankName": "HDFC"
}
```

miro

```
const cds = require("@sap/cds")

module.exports = cds.service.impl(async (srv) => {
    //All your implementation
    srv.before(["CREATE", "UPDATE"], "EmployeeSet", async (req) => {

        const DOB = req.data.DOB
        const iage = calculateAge(DOB);
        req.data.AGE = iage;

        if (parseFloat(req.data.salaryAmount) >= 1000000) {
            req.error(500, "Salary must be below 1000000")
        }

    })

    function calculateAge(date)
    {
        const today = new Date()
        const givenDate = new Date(date)
        const age = today.getFullYear() - givenDate.getFullYear();
        return age;
    }
```

One thing you remember that what ever data modification or addition we do in the **before** event that data comes to the **on** event and from there CRUDQ operations happens.

Now I want that after creating the employee in the response the in the First Name of the employee I want to add Mr as prefix. With the help of After event.

So we will do this with the help of after event. So after event is mainly used to show some extra data before it displays to the user.

```
srv.after("READ", "employee", async (req) => {
    req.forEach(element => {
        element.nameFirst = `Mr. ${element.nameFirst}`;

    });

})
```

So this is the code, So we have used **after** event and inside the after event we have defined the **READ** because when you create an employee by default you first read only, that's why READ also it will work instead of CREATE. So what req we are getting that we are iterating using the for-each loop.

And we are adding Mr in the First-Name. So after create the employee when we will read the employee data, then we will see that Mr is added as a prefix in the First-Name.

What is fiori elements?

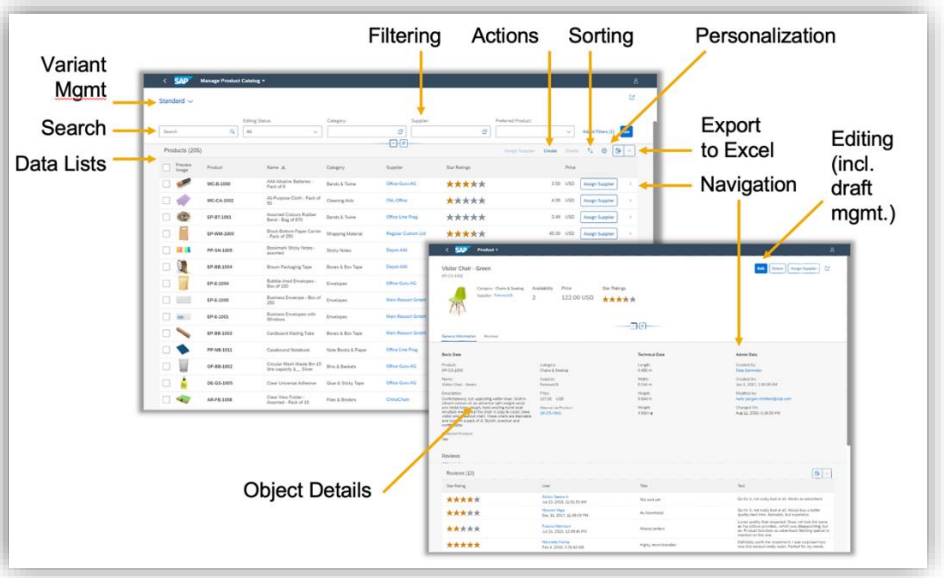
SAP Fiori elements is a framework that consists of the most commonly used floorplan templates and is designed to :

- Speed up development by reducing the amount of frontend code needed to build SAP Fiori apps.
- Drive UX consistency and compliance with the latest SAP Fiori design guidelines.

There are different fiori templates are available

1. List Report
2. Worklist
3. Object Page
4. Overview Page
5. Analytical List Page

So we will be developed a list report and Object Page using fiori elements.



So here in the list report on the click of the button it will navigate to the Object Page.

What is facet in object page? Facets are essentially sections or tabs within an object page that organize the information related to a particular object or entity. Facets help to structure the information. Users can to easily navigate from one facet to another facet.

What is line items in object page ?

In SAP Fiori, a line item refers to row of data displayed on an object page. Object pages in SAP Fiori typically present detailed information about a particular object or entity, such as a sales order, purchase order, customer, product, etc.

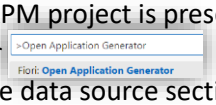
What is object header in the object page ?

In SAP Fiori, an "object header" typically refers to a UI component used to display key information about an object at the top of a page or within a header section. This component is often used to provide a quick summary or overview of the object being displayed.

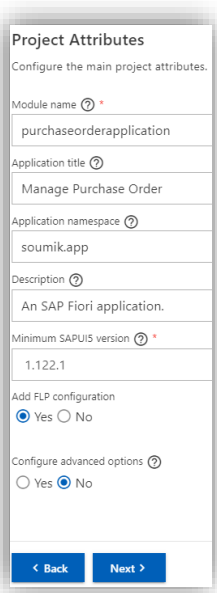
Difference between freestyle SAPUI5 and SAP Fiori elements

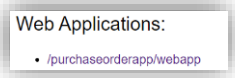
1. Freestyle fiori app gives the flexibility and fiori elements gives the efficiency
2. Freestyle fiori app needs the design requirements from the architects and in the fiori elements we get the design requirement from the SAP templates
3. To build the freestyle fiori app we need the web-development knowledge. And to build the fiori elements we need to know the annotations.
4. In the freestyle fiori app we need to write the code and UI logic by own that means we are the owner and we will take care of the maintenance and in the fiori elements the annotations are the owner and they will take care of the maintenance.
5. The total cost of development and maintenance of the freestyle fiori app is Higer. For the fiori elements the total cost of development and maintenance is lower.

So we will start developing the fiori elements

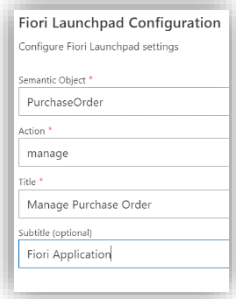
I will go the BAS where is my CAPM project is present. There I will click **Ctrl+Shift+P** and I will select the Open Application Generator 

and from there I will choose the **List Report Page** and will click on NEXT. And in the data source section I will choose ***Local CAP Project***. And I will choose my CAP project name which is **capm.** And after that I will choose the OData Service as **CatalogService** and after that I will click on next. And I will choose the main-entity as **Pos** (Purchase order) and I will choose the navigation-entity as **Items.** So that means when I click on the Purchase Order I will navigate to the Purchase Items. After that I will configure my project attributes. After that I will click on next and I will fill the Fiori Launchpad configuration. And I will click on finish. And after that In the App section the Project will be create automatically. And now I will hit the command **cds watch** and I will open the port. And there apart from the service I will see the web applications also. So that will be my fiori application.

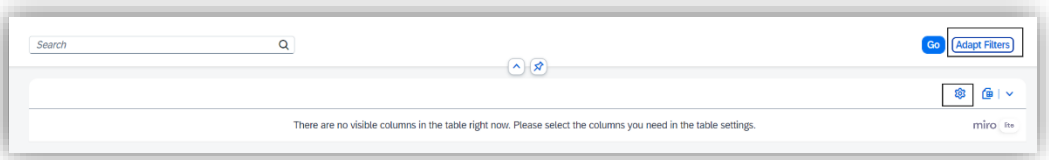




----- So I will click on that it will open the fiori application.



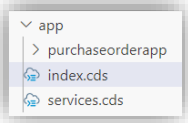
One thing remember that in the csv file in th amount column what amount we will pass that we should not use **comma.** If we use comma it will not work in the fiori application



So after you open the fiori app you see like this. So here if you want to show the columns then for that you will click on the **settings** button and will select the column which you want to see the data. And If you want to apply filter then you will click on the **Adapt Filter** and there you will select the filter.

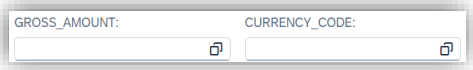
Now I want that when I open the fiori application I don't want to apply the filter manually and I don't want to add the column manually. I want that it should come automatically out of the box.

So for that in the app folder I will create a file name **index.cds** and inside the fiori app (purchaseorderapp) we will create a file name **annotation.cds** file. The use of the **index.cds** file is just point to the **annotation.cds** file which is inside the fiori app. So in that **index.cds** file we will just give the path of the **annotation.cds** file so whenever the application load then it will load the **index.cds** file and from there it will load the **annotation.cds** file. So many annotations file we will create inside the fiori app the path of all the file we will keep inside the **index.cds** file.



And now inside the **annotation.cds** file we will write the annotations to solve the problem ,which is mentioned in the question.

SelectionFields : this fiori elements define the filter fields that are to be displayed in the filter panel of the value help dialog.



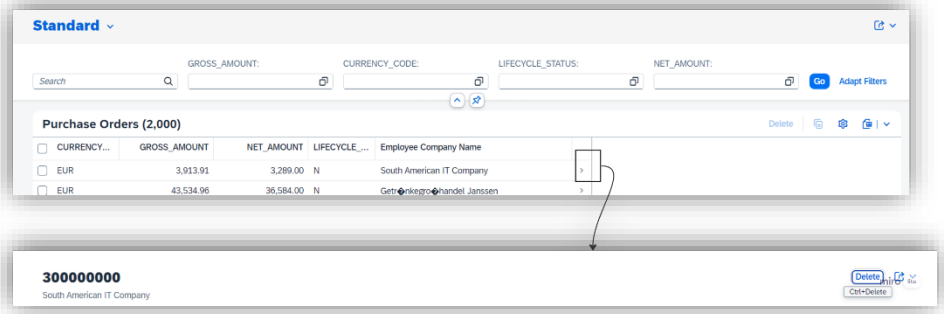
LineItem : this fiori elements this annotations define the column of the table.

HeaderInfo : this fiori element this annotations define an entity and its title, and an optional short description, the name of its entity in singular and plural form,

annotations.cds

app > purchaseorderapp > annotations.cds

```
using {CatalogService} from '../../srv/CatalogService';

annotate CatalogService.POs with @(UI: {
    SelectionFields: [
        GROSS_AMOUNT,
        CURRENCY_CODE,
        LIFECYCLE_STATUS,
        NET_AMOUNT
    ],
    LineItem       : [
        {
            $Type: 'UI.DataField',
            Value: CURRENCY_CODE,
        },
        {
            $Type: 'UI.DataField',
            Value: GROSS_AMOUNT,
        },
        {
            $Type: 'UI.DataField',
            Value: NET_AMOUNT,
        },
        {
            $Type: 'UI.DataField',
            Value: LIFECYCLE_STATUS,
        },
        {
            $Type: 'UI.DataField',
            Value: PARTNER_GUID.COMPANY_NAME,
        },
    ],
    HeaderInfo     : {
        $Type         : 'UI.HeaderInfoType',
        TypeName      : '{i18n>PurchaseOrder}',
        TypeNamePlural: '{i18n>PurchaseOrders}',
        Title         : {
            Label: '{i18n>POID}',
            Value: PO_ID
        },
        Description   : {
            Label: '{i18n>DESC}',
            Value: PARTNER_GUID.COMPANY_NAME
        }
    },

});
```



What is criticality in fiori elements

In Fiori elements, "criticality" used to visually indicate the importance of certain data by highlighting them.

criticality has following values

So criticality has 4 values 0----[Neutral], 1-----[**Red**], 2-----[**Orange**], 3------[**Green**]

So in my Purchase Order entity there is a attribute called **LIFECYCLE_STATUS** it has 3 values N, B, D So I want that when we will display the column then the column value should show full name like this New, Blocked, Delivered without changing the data in the DB file. And also I want to give some colour to this 3 values with the help of criticality.

So we will use for the after event. As because we are not changing the original data, we are only showing it to the user. So for that

```
//Below code for Purchase Order
srv.after("READ", "POs", async (req) => {
    req.forEach(element => {
        if (element.LIFECYCLE_STATUS == 'N') {
            element.LIFECYCLE_STATUS = 'New'

        }
        else if (element.LIFECYCLE_STATUS == 'D') {
            element.LIFECYCLE_STATUS = 'Delivered'

        }
        else if (element.LIFECYCLE_STATUS == 'B') {
            element.LIFECYCLE_STATUS = 'Blocked'

        }
    });
})
```

So here in this code as you can see after data is displayed to the user, the after event will be triggered. So we are writing the condition inside the after event. So basically we are iterate through all the elements. So when condition match we will convert the value.

net amount ...	Gross amount of pos	Gross amount of pos	Lifecycle statusof POS	Employee Company Name
EUR	3,913.91	3,289.00	New	South American IT Company
EUR	43,534.96	36,584.00	Delivered	Getr\ufffdnkegro\ufffdhandel Janssen
EUR	18,142.74	15,246.00	Blocked	Danish Fish Trading Company

Now I want to highlight and add icon of the LIFECYCLE_STATUS column value, with the help of Criticality.

So basically Criticality has accept some values, based on that it changes the colour. So first in the purchase-order entity we will add the field name Criticality as Integer also in the csv file we will add the Criticality column. So now in the Criticality field I need to initialize the values, so that it can highlight. So we want to highlight the LIFECYCLE_STATUS column value. So we will use the Criticality for that particular column

```
//Below code for Purchase Order
srv.after("READ", "POs", async (req) => {
    req.forEach(element => {
        if (element.LIFECYCLE_STATUS == 'N') {
            element.LIFECYCLE_STATUS = 'New'
            element.Criticality = 2

        }
        else if (element.LIFECYCLE_STATUS == 'D') {
            element.LIFECYCLE_STATUS = 'Delivered'
            element.Criticality = 3
        }
        else if (element.LIFECYCLE_STATUS == 'B') {
            element.LIFECYCLE_STATUS = 'Blocked'
            element.Criticality = 1
        }
    });
})
```

	net amount ...	Gross amount of pos	Gross amount of pos	Lifecycle stat...	Employee Company Name
○	EUR	3,913.91	3,289.00	⚠ New	South American IT Company
○	EUR	43,534.96	36,584.00	✓ Delivered	Getr\ufffdnkegro\ufffdhandel Janssen
○	EUR	18,142.74	15,246.00	⊗ Blocked	Danish Fish Trading Company

Creating a Facet:

Facets are essentially sections or tabs within an object page that organize the information related to a particular object or entity. Facets help to structure the information. Users can to easily navigate from one facet to another facet. So basically to create facet we will use facet annotations.

So we have group of fields that's why we create Facet. So basically when we define an facet it is an array, because we can add multiple facet. `Facets : [],`

Types of facet

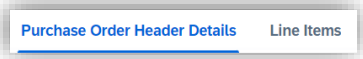
UI.ReferenceFacet : Facet that are refers the things that are Reference facet **e.g.** LineItems

UI.ReferenceURLFacet : Facet that refers to a URL

UI.CollectionFacet : It is a collection of facets.

So while creating a facet we will add 3 things **Type**, **Label**, **Target**.

- So Type is basically which type of facet it is.
- And the Label would be the name of my facets.



- So for reference purpose we will use Target. Example I want to add the LineItems in my facet so for that we will create the LineItem in somewhere outside and we will refer them in the facet with the help of Target.

Creating the **1st facet**: and here I have taking

```
Facets                    : [
    {
        $Type : 'UI.ReferenceFacet',
        Label : '{i18n>POGeneralInfo}',
        Target: ![@UI.FieldGroup#HeaderGeneralInformation]
    },
```

So, this is my first facet which I have created the reference of Field Groups. So basically Field Group is basically a group of fields where we can have group of data.

```
FieldGroup#HeaderGeneralInformation: {
    $Type: 'UI.FieldGroupType',
    Data : [
        {
            $Type: 'UI.DataField',
            Value: PO_ID,
        },
        {
            $Type: 'UI.DataField',
            Value: PARTNER_GUID_NODE_KEY,
        },
```

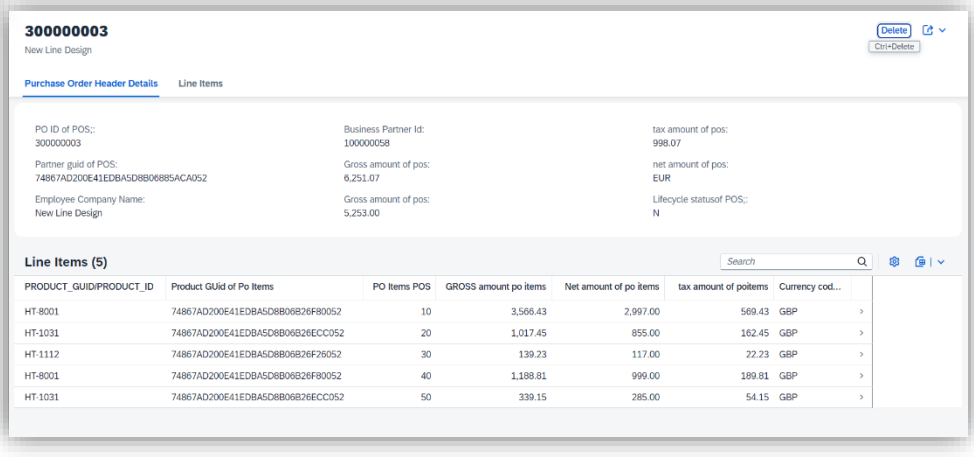
So here I have defined my FieldGroup and inside the Data there we kept the attribute. So If tomorrow there are multiple FieldGroup, so to identify uniquely we have given the Name **FieldGroup#HeaderGeneralInformation** and same name we have referenced in the Target.

Creating the **2nd facet**: created and here I have taking

```
{
    $Type : 'UI.ReferenceFacet',
    Label : '{i18n>POITemDetails}',
    Target: 'Items/@UI.LineItem'
}
```

So, this is my second facet which I have the reference of Line-Items. So basically Line-Items is a data in tubular format.

```
annotate CatalogService.POItems with @(UI: {LineItem: [
    {
        $Type: 'UI.DataField',
        Value: PRODUCT_GUID.PRODUCT_ID,
    },
    {
        $Type: 'UI.DataField',
        Value: PRODUCT_GUID_NODE_KEY,
    },
```

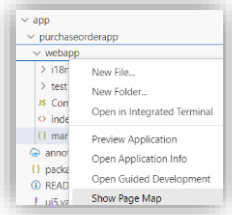


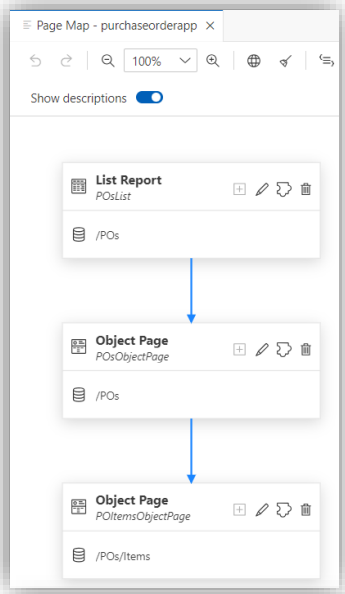
So in the Second Facet I have added a Line Items, from that I want to further navigate.

Now I want to do further navigation from the Line Items. For that First I will define a Header info. So after the Line Items are done we will define a Header Info. So when I will click any crow of the line item, It will navigate to the 3rd view.

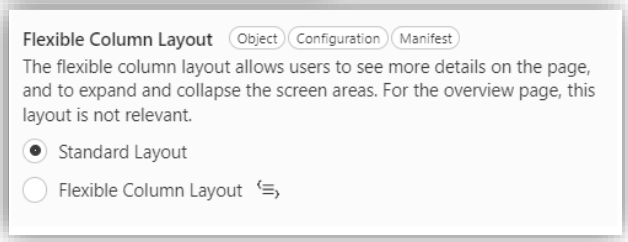
```
HeaderInfo: {
    $Type          : 'UI.HeaderInfoType',
    TypeName       : 'PO Item',
    TypeNamePlural: 'PO Items',
    Title          : {
        $Type: 'UI.DataField',
        Value: NODE_KEY
    },
    Description    : {
        $Type: 'UI.DataField',
        Value: PO_ITEM_POS
    }
},
```

So now I will show you a very interesting thing. So in the fiori application if you come you will see a **manifest.json** file so inside this file there are defined the **routes** and **targets** so initially when the fiori app was created automatically then there was only 3 routes and 3 targets are configured but now I want to add another view that means I will be having 4 view. So for that I will have to add another view in the routes and targets. But sap gives a facility that without adding anything in the **manifest.json** file we can add the views. So for that inside the fiori app there will be a **webapp** folder and we need to right click on that folder and then you will see an option called **Show Page Map.** Click on that and you will see a graphical representation of all the views.



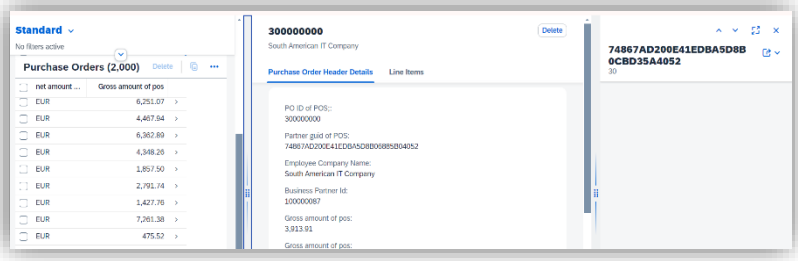


So this is the graphical representation of all the views. So here you can see the **edit** button to edit the view. And you will see the **delete** button to delete the views. And also there is a **Plus** button to add the view. So with the help of this we can add the mul tiple views, without writing any code in the manifest.json file.



So you will also see the different column layout that is **Standard Layout & Flexible Column Layout.** So if you select the Standard then after navigate from one view to another. It will take to you to another page to show the next view.

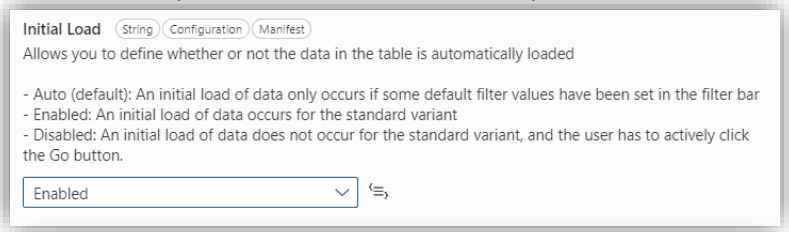
So if you select the Flexible Column Layout then it will allow you to collapse the screen areas. And you will see the all of your views in a single page.

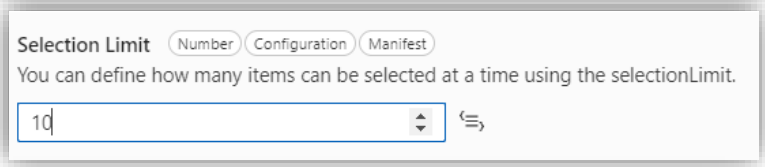


So this is the flexible column layout. So here you can see all of my views are came into the single page. And we can collapse the views

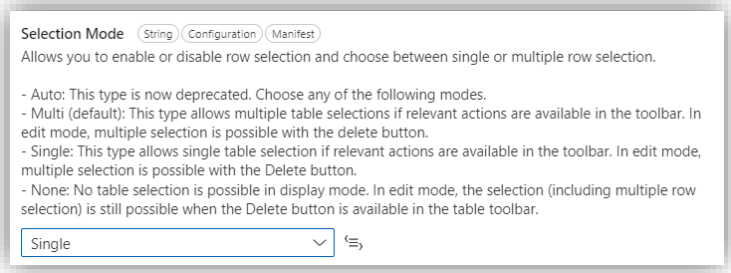
So when the application load the column data not come. So I want that the column data should come.

So for that I will go to the graphical representation of the views. I will click on edit button of the first view, in which column data is not loading. After that you will see the Table option if you click on that you will see the **Initial L oad** option set that **Enabled .** After that column data will load.





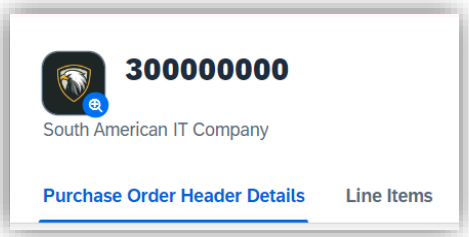
And this **selection limit** use to show that how many column data you want to show at a time. So I given 10 that means it will show 10 records at a time.



And this selection mode will tell you that how many row you want to select at a time. So I have given single that means at a time I can select One row.

I will add a icon to the application : So in the **HeaderInfo** section after the title and description I will add a image URL. And that URL will show the image.

```
HeaderInfo: {
    $Type         : 'UI.HeaderInfoType',
    TypeName      : 'PO Item',
    TypeNamePlural: 'PO Items',
    Title         : {
        $Type: 'UI.DataField',
        Value: NODE_KEY
    },
    Description   : {
        $Type: 'UI.DataField',
        Value: PO_ITEM_POS
    },
    ImageUrl      : 'https://www.adobe.com/cont
    mascot-logo-design_fb-img_1200x800.jpg'
},
```

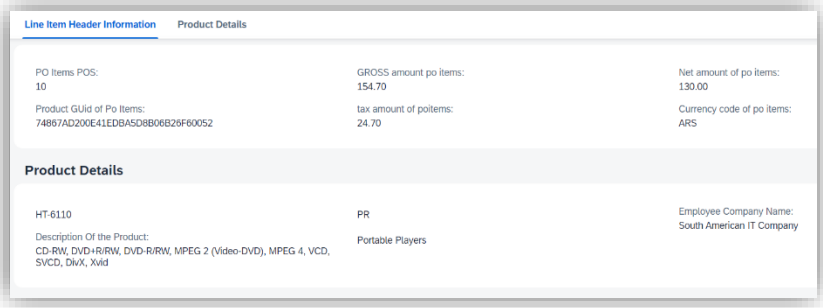


Now I want to implement my third view :

I have implemented the **HeaderInfo** in my third view and also added the Title & Description & Image. Now I will add the Facet. So I have added 2 Facet.

```
Facets  : [
    {
        Label : '{i18n>lineItemHeader}',
        $Type : 'UI.ReferenceFacet',
        Target: '@UI.FieldGroup#LineItemHeader',
    },
    {
        Label : 'Product Details',
        $Type : 'UI.ReferenceFacet',
        Target: 'PRODUCT_GUID/@UI.FieldGroup#ProductDetails',
    }
```

So here as you can see I have added 2 facet. And both 2 facet I have added the Field Group



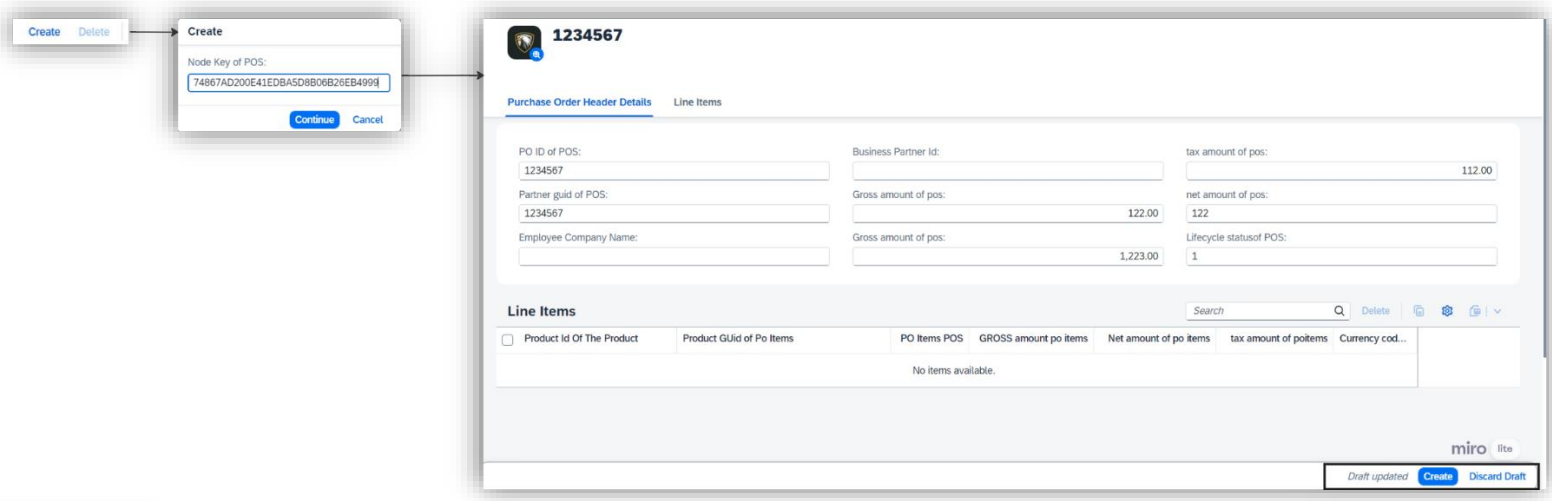
One thing remember that in Facet when you want to refer the data from the different entity, then while referring you have to provide that name of the **Association**. So in the above facet example you can see that I want to show the Product entity data from the poitems entity. So that's why I passed the PRODUCT_KEY attribute. Because from this attribute we connected to the product entity through association

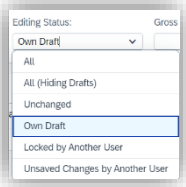
Now I want to insert & delete the Data, and also I want to enable the draft functionality.

So basically **draft** is like if you take an example of Gmail when you write a mail or don't send then it then that mail will save as draft, so next time when you again come to Gmail and try to send or modify the same mail, then you don't need to write a new mail from the beginning. So there is a section called draft, where the mail which you last written it will se save as here. So from there again you can modify that mail.

So similarly here when you are going to insert a data but unfortunately your system got shut down, then again you start your system, you don't need to insert the data from the scratch. So data will be saved as draft and you can continues from that. So there will be a table created named as DRAFT and there your unsaved data will be stored. To enable the draft and insert, delete functionality we need to go inside the SRV folder, there will be a cds file where all of our entity are exposed. And there you need to write the command for the annotation.

So the syntax of the annotation is **annotate {entity-name} with @odata.draft.enabled;**





So in the above picture as you can see we are trying to insert data, and data saved as draft. So to access the draft we will go back to the home page and there will a dropdown field in the header name **Editing Status.** And from the dropdown you will select a value name **Own Draft** and click on **GO**, so that draft data will populate.

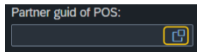
Now if you want to change the theme of the fiori application, that means you want to change it from light to dark, then in the **index.html** file you need to add the theme name which you want.

```
data-sap-ui-theme="sap_fiori_3_dark"
```

Now While creating the Purchase Order data everytime it ask me the ID, so every time I need to pass the Id, but I want Id create automatically.So we will solve this problem by using cuid aspect so in the Pruchase Order table and also I will use in the Purchase Order Items table. And after that in the both purchaseorder, poitems csv file I will modify the column name to ID and association name also I will modify **PARENT_KEY_ID.**

After doing this necessary changes, and deploy, when we will go to the UI and create purchase order then it will not ask for the Id, it will automatically create **UUID** as we will use **cuid** aspect.

Now I want that while creating the Purchase Order data, in the **Partner guid** field we want to add a value help



So for that we need to write the code in the **annotations.cds** file.

```
//We will implement value help for the Partner_Guid for Purchase order table
annotate CatalogService.POs with {
    PARTNER_GUID @(
        Common          : {Text: PARTNER_GUID.COMPANY_NAME, },
        ValueList.entity: CatalogService.BPSet
    )
};

@cds.odata.valuelist
annotate CatalogService.BPSet with @(UI.Identification: [{
    $Type: 'UI.DataField',
    Value: COMPANY_NAME,
}, ]);
```

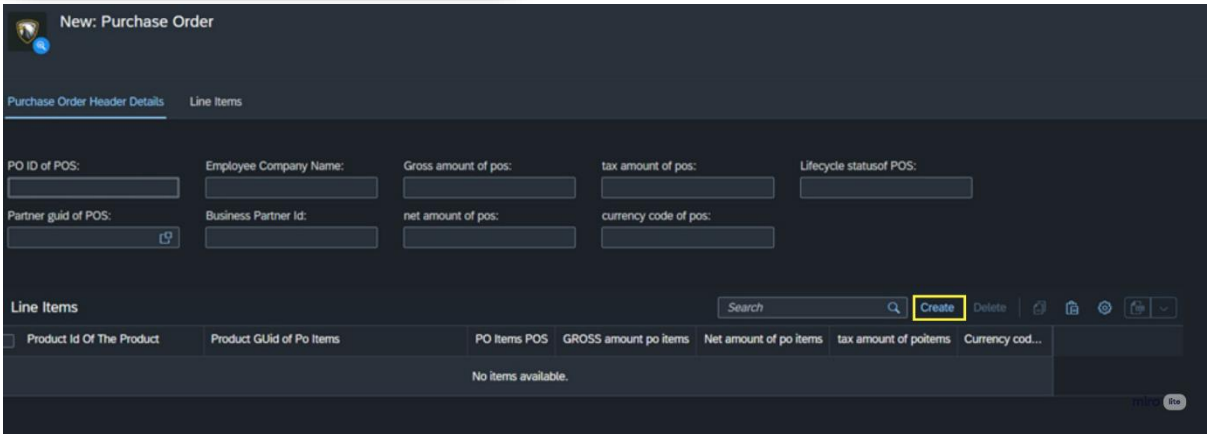
Now I am able to create the Purchase Order, but I also want to create the Purchase Order Item

So to do that I need to set a one to many relationship between the **Purchase-Order** to **Purchase-Items**. So I need do Composition from **Purchase-Order** to **Purchase-Items.**

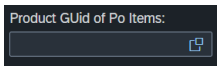
The Purchase-Items can not be complete without the Purchase-Order. So that's why we need tight coupling. That's why we make it Composition. Then only we can create the Purchase-Items for the Purchase-Order.

```
entity purchaseorder : common.Amount {
    key NODE_KEY         : Guid;
        PO_ID            : String(24);
        PARTNER_GUID     : Association to master.businesspartner;
        LIFECYCLE_STATUS : String(1);
        OVERALL_STATUS   : String(1);
        Criticality      : Integer;
        Items            : Composition of  many poitems
                               on Items.PARENT_KEY = $self;
}
```

```
entity poitems : common.Amount {
    key NODE_KEY     : Guid;
        PARENT_KEY   : Association to purchaseorder;
        PO_ITEM_POS  : Integer;
        PRODUCT_GUID : Association to master.product;
}
```



While creating the Purchase-Item I want to add the Value Help for the Product Guid of Purchase Items




```
//We will implement value help for the Product_Guid for poitems table
annotate CatalogService.POItems with {
    PRODUCT_GUID @(
        Common          : {Text: PRODUCT_GUID.DESCRIPTION,
        },
        ValueList.entity: CatalogService.ProductSet
    )
};

@cds.odata.valuelist
annotate CatalogService.ProductSet with @(UI.Identification: [{
    $Type: 'UI.DataField',
    Value: DESCRIPTION,
}, ]);
```

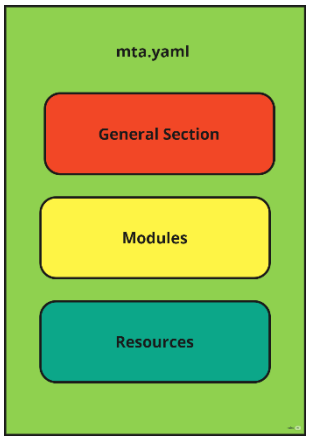
Deploy Application to Hana Database

So now in the **App** folder assume there are 2 application. And we have **db** folder, **srv** folder. Now I want to deploy all this things into BTP. All this files or folder we need to configure somewhere before deploying to BTP, so to do that there is a file called **mta.yaml** file. mta stands for multi target application.

What is yaml file?

YAML is a human-readable data serialization language that is often used for writing configuration files. The full form of yaml file is yet another markup language.

So basically in this mta.yaml file there would be 3 section



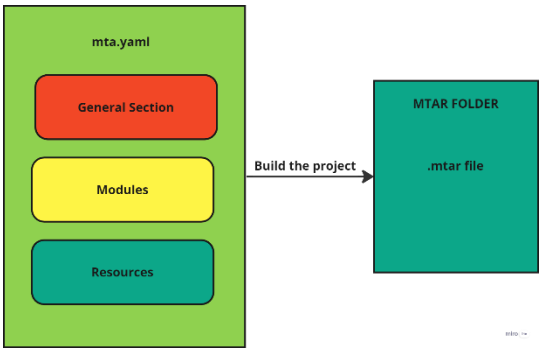
So in the general section there will be containing the details of the application and description.

In the Modules section we will defined the modules which we are going to deploy in BTP, like **db**, **srv**, **app** this things we want to deploy in our BTP that we will mention in our modules section.

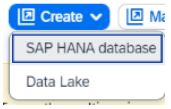
In the resources section we will defined which service we are going to use and what is my service plans.

And this **mta.yaml** file we no need to create manually. We will write a command to create the **mta.yaml** file.

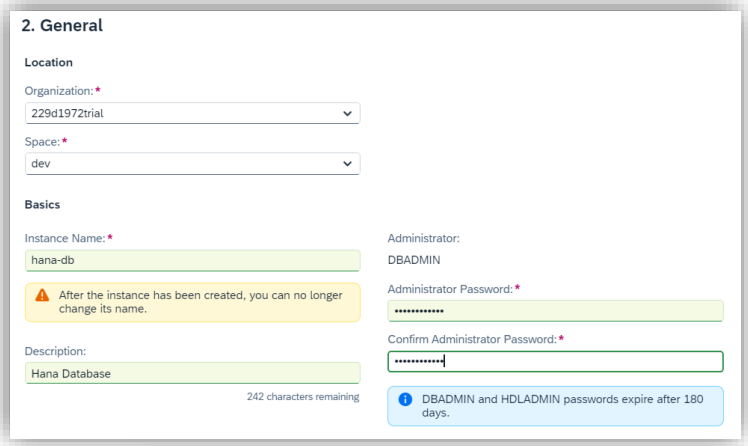
First we need to build our project. Once build is done this **.mtar** file will create Inside **mta_archives** folder **.mtar** file will be created. And after that we will deploy this **.mtar** file. This **.mtar** file contains all the section which is present in the mta.yaml file.



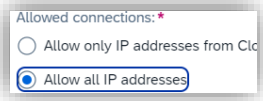
We will use **hana db,** for that we need to create the db. So if you go inside your subaccount there you will see a **Cloud Foundry** folder, inside that **Spaces** will be present. Once you go to the Spaces you will see there is a **dev space**. You need to go inside the dev space. Once you go inside the dev space, there you will see a **SAP HANA cloud** in the left side. So you need to click on that **SAP HANA Cloud** and there a **Create** button will present. So if you click on that Button you will see a drop-down, from that drop-down you need to click the **SAP HANA database.** After that a new page will open and from there you will select the option **Sign in with default identity provider.** Then it will ask for sign-in you will sign-in by providing the BTP credentials. After that you will click on **AUTHORIZE**

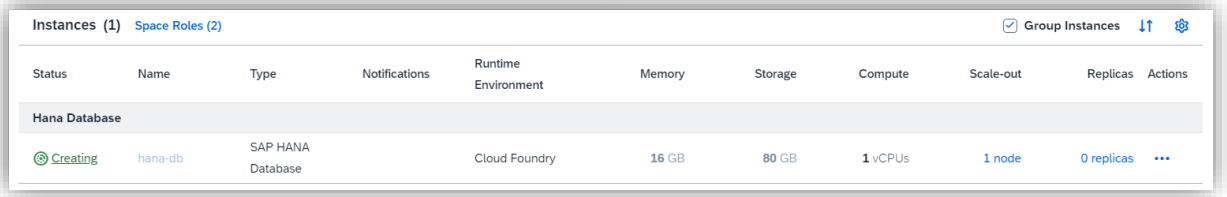


Now you will redirect to a new page. There you will see different SAP HANA cloud instance. From there you need to choose a cloud instance. So I will choose **SAP HANA Cloud, SAP HANA Database.** After that I will click on the **Next Step** button. After that you will get a general section there you need to fill the mandatory section like Instance name & Administrator Password**.** So I have given the **hana-db** instance name. And I have given the Description(optional field) **Hana Database**. And I have given the Administrator password **Sapfiori@123.** And the username is **DBADMIN.**



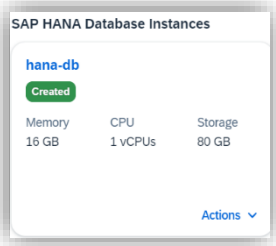
Now after that click on **Next Step** button. Again, click on **Next Step.** Again, click on **Next Step.** After that you will go to the Hana database advance settings page. There you will select the **Allow all IP addresses** dropdown. Otherwise it will start restricting the IP addresses. After that click on **Next Step.** After that click on **review and Create.** After that review all the data which you submitted. After that click on **Create Instance.** It takes time to create instance.

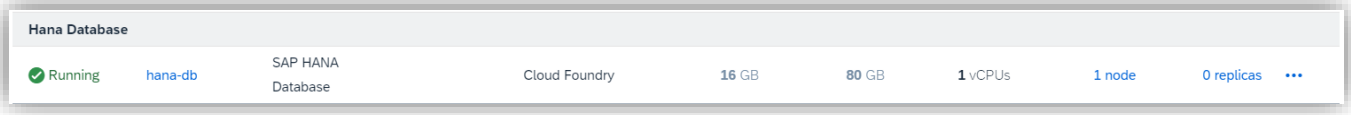




So if you want to check you hana-db is running or not how will you check

For that you will go to the SAP HANA Cloud SAP HANA Cloud there you will see the hana-database is created. If you see more details on hana-db then you need click on the Manage SAP HANA Cloud there you need to check whether the hana-db is running or not. If it is running then it is fine otherwise you need to start it by click the **3 dot**

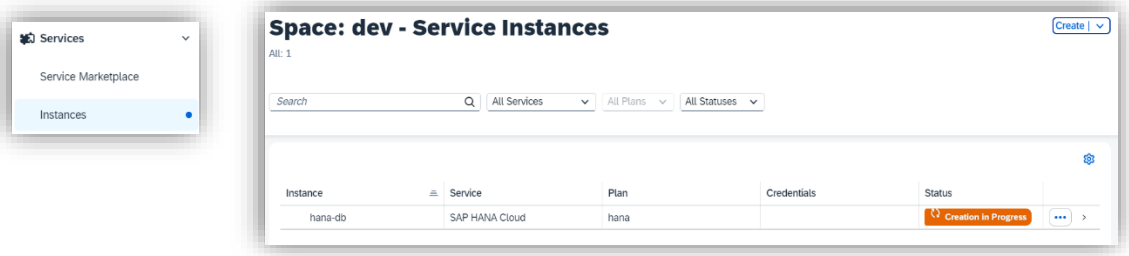




How to start the hana-db in case it is stopped.

For that you need to SAP HANA Cloud there you will see Manage SAP HANA Cloud and from there you will start the hana-db by clicking **3 dot**.

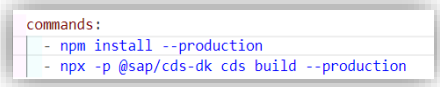
Now if you go to instances, which is inside the Service folder, there also you can check what are instances are present and what is the status of that.



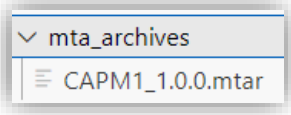
Now I want to add hana database for that the command will be **cds add hana.** Now I want to add mta.yaml file for that the command will be **cds add mta.** After adding the mta.yaml file in the resources section we will see the **hana-db** this came because we add the hana database.

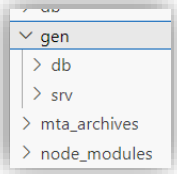
In my module section now currently have **Srv** and **db-deployer**. So this db-deployer job is deploy the database as it is in the BTP.

Now I want to built our project for that the command will be **mbt build,** before build I need to add following commands in the **mta.yaml** file, otherwise it will throw error while building the project. After build the project, it will



- create a **gen** folder automatically.
- And also it create the mta-archives folder, inside that mtar file will be present.





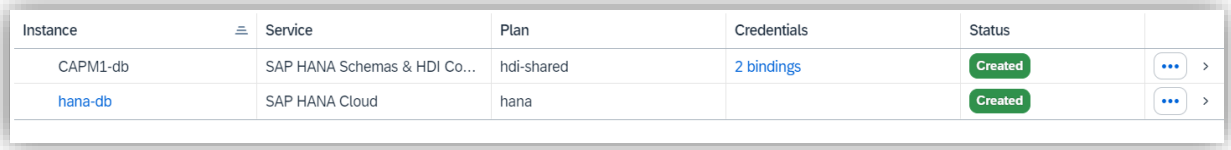
If you go inside the gen folder you will see the **db** & **srv** folder .In the gen folder inside the **db** folder the data will be store as table. Because hana **db** don't understand the csv file that's why it convert to table.

Now we will deploy our **mtar** file to BTP. There are 2 options are there to deploy the **mtar** file.

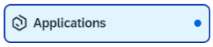
1st option is right click on the mtar file. And select the option **Deploy MTA Archive**
2nd option is deploy by using the terminal using **cf login**

we will go with the 2nd option. So in the terminal I will run the command **cf login.** After that it will ask for a endpoint. I will paste it. After that it will ask for username and password. I will give the details. And now I successfully logged in. Now to deploy the command will be
cf deploy {my mtar file path} in my case the command is **cf deploy mta_archives/CAPM1_1.0.0.mtar** and after that click on enter. Now it will start creating the services.

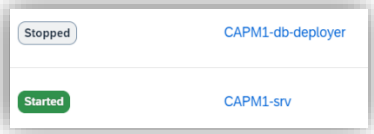
After deploy successfully, if you in the Intance and you will see that **CAPM1-db** instance is created. And which is having 2 binding. And another is **hana-db** which is created by us.



Now in the above diagram you will notice that the **CAPM1-db** which got created, having 2 binding. That means. If you go to your **mta.yaml** file, there you will see the **modules** section 2 modules which requires this **CAPM1-db** that's why its showing **2 bindings.** So in my case `CAPM1-srv, CAPM1-db-deployer` need this **CAPM1-db** resource, that's why showing 2 bindings

Now if you go to the **Applications** 

you will see it created the application CAPM1-srv and CAPM1-db-deployer.



Now during deploying the application, I will get some error in the console. So to see the errors clearly I will go to the logs of the application. So for that first of all I will go to the Applications. And there I will see my application. I will click on the application **CAPM1-srv** and after go inside the application in the left side you will see the **Logs.** So click on the logs you will get the full logs and from that you can easily see the error.

So one error ye found that -----> Error: Cannot find module '@sap/xssec'. Make sure to install it with 'npm i @sap/xssec'

So here it is telling that module don't found and it is recommend to install the module. So we will install that. For that I will write the command **npm install @sap/xssec passport**

After installation, we will again built the MTA project. So to build that in the mta.yaml file we will do right-click. And click on the option **Built MTA Project.** So it will replace **mta_archives** folder with new one. After that we will built the mta file. For that we will go to the mta file, which is inside the **mta_archives** folder and right click on it. And click on the option **Deploy MTA Project.**

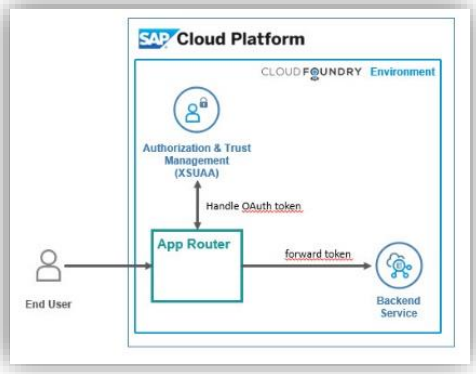
Again we will check the latest logs and we will get the errors in the logs. Besically in the error it is telling that no XSUAA instance bound to application. That means we need to add XSUAA.

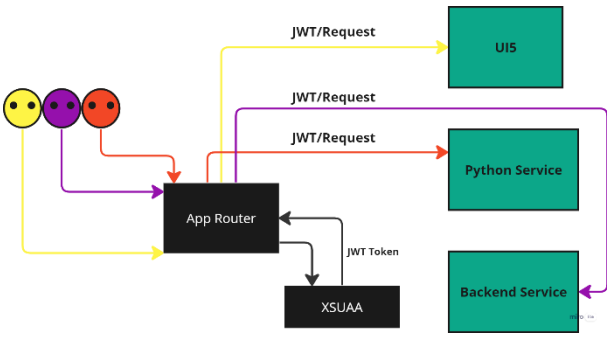
So earlier it was easily possible to deploy the application into BTP. But now SAP says that without **Security** you can not deploy the application. So to do this security we add something called **XSUAA.**

Now we will discuss a concept called app-router.

Suppose you are working in a IT company. So everyday go and enter into the office. So there is a main door through which you enter into the office. So there you swipe your card then doors open and then enter into the office. So this is where you are authenticate yourself that you are a person who is working in this organisation. And this biometric/swipe will check that the user is valid or not.

So similarly whenever user send some request, instead of directly going to the target service, the request first goes to the **App Router.** The App Router will go to the **XSUAA.** So this XSUAA has something called Client-Id ,Client-Secret kind of mechanism. And then using that Client-Id , Client-Secret it generates a authentication token called **JWT Token** and give it back to the App Router. And after that along with the JWT token, the request goes to the target service. So once we get the JWT token, it attached with the Request and goes to the target service and there it authenticated and give the response back. So this is how the App Router concept works.





So here in this picture as you can see there is some user who is trying to access the service by sending the request. But there we need have some security, otherwise anyone can access the service. So that's why App Router concept came into the picture. So instead of directly going to the target service, the request first comes to the App Router. So the App Router goes To the **XSUAA.** So this XSUAA has something called Client-Id ,Client-Secret kind of mechanism. And then using that Client-Id , Client-Secret it generates a authentication token called **JWT Token** and give it back to the App Router. So now App Router contains the request so the JWT token attached to the request and goes to the targeted service. and there it authenticated and give the response back. So this is how the App Router concept works.

Now we will create the instance of XSUAA into our application.

So for that first in the BTP we will go inside our Subaccount. There in the left side inside the **Services** folder we will go **Instances and Subscriptions.** And there in the left side you will see **create** button. With the help of that you will create the **XSUAA.**

So it will ask for provide some details, so we will give that accordingly.

Service --- Authorization and Trust Management Service
Plan --- application
Runtime Environment --- Cloud Foundry
Space --- dev
Instance Name --- CAPM1-xsuaa

After that click on **Next,** and then click on **Create** button. After sometime It will create, which we will see it **Instance and Subscriptions.**

So after created the XSUAA, If we click on that you will **View Credentials.** Which is disable. So enable the **View Credentials** we need to create a Service Key. For that we will click on **Create** button, Once we click on the Create button it will ask for the **application name** so generally we provide the xsuaa instance name and add **"-key"** with them. So I have given the name CAPM1-xsuaa-key and clicked on the create button. And Once key got created you will see the **View Credentials** also enable. Now if you click on the View Credentials you will see the everything client-id, client-secret etc.…
Now from this it will create **JWT Token.**

So we will go to the **POSTMAN** application. We will create a request name is **getJwtToken.** And there you will go to the Authorization. And there you will Select the Authorization type **OAuth2.0**

Now in the **Configure New Token** section you will provide the Token Name. I have given token name as **xsuaa.** And Grant Type I have selected **Client Credentials,** because I have the client details with me.

Now you will go to the View Credentials and there you will see a **URL** you will copy this URL and paste it somewhere and in the last of the URL you have to add **/oauth/token** now this complete URL you will paste it **Access Token URL** which is in the postman. After that you will fill Client ID, Client Secret and after that you will come down and click on **Get New Access Token** and it will create a token automatically.

We can also decoded the JWT auth token. So for that we will go to the https://jwt.io/ and there you will paste the token.

What is JWT token ?

A JSON web token(JWT) is JSON Object which is used to securely transfer information over the web(between two parties). It can be used for an authentication system and can also be used for information exchange. The token is mainly composed of header, payload, signature. These three parts are separated by dots(.).

Now we will create a AppRouter. [In unmanaged approuter we create approuter manually]
So to install the AppRouter we will use the command **cds add approuter.** After add this a new file will be created **xs-security.json** on that file I will add roles, for security purpose we will do this. And also a **router** folder is created inside the **app** folder.

Due to add **approuter**, may be there can be added some dependencies in the package.json file, so for that we will do **npm install.** It will install if any thing is not yet install.

After this above changes, now we will build the application using **mbt build** command. And after that we will deploy the mtar file. Right click on the mtar file and choose the option **Deploy MTA Archive**.

After deploy the **.mtar file** a new instance would be created name **CAPM1-auth.** Because we have earlier created a **capm-xsuaa** instance, for that.

So what are all Modules and Resources are available currently ?
So currently we have 3 module 1st is srv module, name is **CAPM1-srv,** 2nd is **CAPM1-db-deployer** 3rd module is AppRouter whose name is **CAPM1**

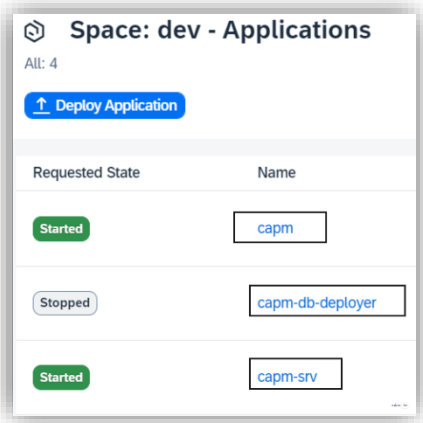
And we have 2 resources 1st is hana database, name is **CAPM1-db.** 2nd resource is xsuaa for security purpose whose name is **CAPM1-auth.**

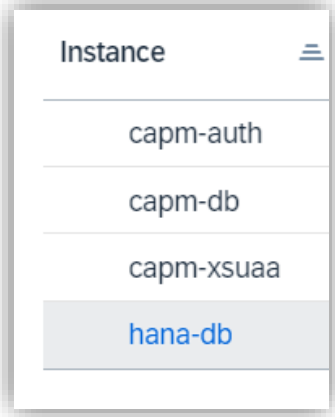
What is the meaning of requires attribute in the Module

So basically what we want to deploy that we will keep on the module section. Suppose I want to deploy **srv,** so that I will keep on the module. And Module requires some extra service which help the module to successfully deploy. So in the module section we will mention those service names with the help of **requires** attribute.

What will happen with my Modules & Resources in mta.yaml file after deploy the applications ?

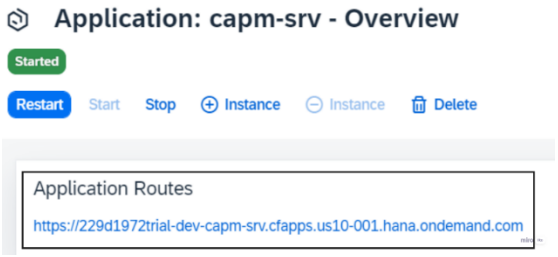
So in our mta.yaml file we had 3 modules ---- **capm-srv, capm-db-deployer, capm** so after deploy the applications, in the BTP Applications section you will see all of you module as Applications





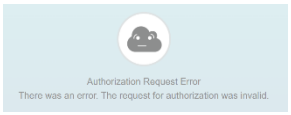
And what are all the resources we have created that will be present in the **Instance** section.

Now if we want to access the application services, for that we will go to **Applications**, from there we will click on the service, so my service name is **CAPM1-srv.** Once we will go inside the service there we will see Application Routes, so we will click on the Application Routes link. So it will display all of my Entities which we defined, but we can not access the data of that entity, it will show unauthorized.



So while accessing the service **capm-srv** we are getting Unauthorized, because we can not directly access the services, we need to go through via AppRouter.

Now to go to the AppRouter **capm** , when we try to access then we will get error



So to remove this error, from the App router, we need to modify our code, so for that we will go to our **xs-security.json** file and there will add the piece of code

```
"oauth2-configuration":{
  "token-validity": 3600,
  "redirect-uris": ["https://*/**"]
}
```

And after that we will go to the mta.yaml file and there in the resources section you will **capm-auth** resource. From that resource you will cut the **config** part.

```
config:
  xsappname: capm-${org}-${space}
  tenant-mode: dedicated
```

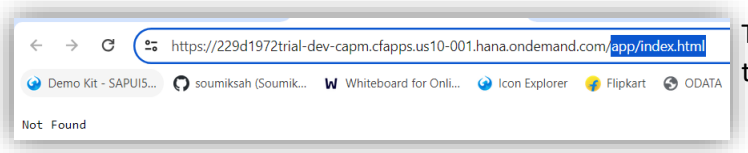
And after that you need to go to the **xs-security.json** file and there on the top you will paste the code. And modify that code.

```
{
  "xsappname": "capm-xs-app",
  "tenant-mode": "dedicated",
  "scopes": [],
  "attributes": [],
  "role-templates": [],
  "role-collections":[],
  "oauth2-configuration":{
    "token-validity": 3600,
    "redirect-uris": ["https://*/**"]
  }
}
```

Now After doing this modification, we need to re-build and after that re-deploy the code. But before redeploy the code we need to do one thing, that is we need to delete the **CAPM1-auth** instance, because we did modification, for that reason this instance not able to update properly, so that's why need to delete this instance and after that we need to re-build and re-deploy.

The command for build is : **mbt build.** And for deploy Right click on the mtar file and choose the option **Deploy MTA Archive.**

So once the deploy is done we will try to access the App-Router that is **CAPM1,** but we will get error as not found.



The reason for this error is it is searching the **index.html** file inside the **app** folder,

But there is not index.html file in the app folder. So to remove this issue we will go to the **xs-app.json** file which is inside the **app/router.** In that **xs-app.json** file there is a attribute name is `welcomeFile` whose value is `app/index.html.` So that's why this problem is happening. So to solve this problem we will remove this attribute. And again we will re-build and re-deploy.

Now we can access all the entity and also the data if the entity.

approuterouter Implementations

1. To add approuter into the project, to do that we use the command **cds add approuter.**
2. It generates below listed items
    a. **New** xs-security.json file : contains Roles, Scope, Role-Collections
    b. **Update** package.json file : it adds **"auth": "xsuaa"**
    c. **Update** mta.yaml file : it adds resource

```
- name: EmployeeDetails-auth
  type: org.cloudfoundry.managed-service
  parameters:
    service: xsuaa
    service-plan: application
    path: ./xs-security.json
```

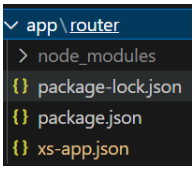
So this a resource whose name is **EmployeeDetails-auth** and the service is **xsuaa.** So this xsuaa job is create the roles and role-collections, that's why we gave the path as **xs-security.json** basically it takes the reference.

    d. **Update** mta.yaml file : it adds modules.

```
- name: EmployeeDetails
  type: approuter.nodejs
  path: app/router
  parameters:
    keep-existing-routes: true
    disk-quota: 256M
    memory: 256M
  requires:
    - name: srv-api
      group: destinations
      properties:
        name: srv-api # must be used in xs-app.json as well
        url: ~{srv-url}
        forwardAuthToken: true
    - name: EmployeeDetails-auth
```

So it adds this **approuter** module.

3. Create a new **router** folder, inside the **app** folder. And this router folder contains this many files.

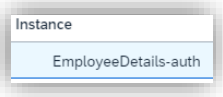


How to access the services through the postman.

So we saw that we can not directly access the services. So we can access through the approuter. So when user send the request, it goes to the approuter. And then approuter goes to the xsuaa and get the access-token and after that the request attached with the access-token and goes to services. And get the response from the particular service.

But if we want to access the service from the postman, then we need to manually create the access-token. And after that we can access the service.

So we will fetch the client details from the instance : **EmployeeDetails-auth**

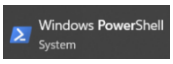


How to deploy the application from Vs code to Cloud.

As of now we deployed our application from BAS to Cloud, but there is a problem that you can not deploy multiple times, because as it is an trial account. But from VS code you can deploy multiple times, there is no restriction.

But to deploy the application from VS we need to install following things :

1. We need to go to **cmd** and there we need to install **mbt** by entering the command **npm install -g mbt**
2. After that we need to install **Chocolatey** to install that we need to go to **windows PowerShell** and we need to run as an Administrator. And then we need to enter the command:




```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

https://help.sap.com/docs/SAP_BUSINESS_ONE_WEB_CLIENT/e6ac71d18c7543828bd4463f77d67ff7/6ec59d6131234d1dbcddbc879737c56e.html?version=10.0_FP_2305

3. After download the Chocolatey we will download the **make**, so we will go to **windows PowerShell** and we need to run as an Administrator. And then we need to enter the command: **choco install make**

4. After that we need to install MultiApps Cloud Foundry CLI Plugin, for that we need to go to cmd and we need to execute the command:

```
cf install-plugin -f https://github.com/cloudfoundry-incubator/multiapps-cli-plugin/releases/latest/download/multiapps-plugin.win64.exe
```

So this MultiApps Cloud Foundry CLI Plugin allow us to execute cf operations in cf cli for vs code.

Once done this you can now deploy the application from Vs code.

While deploying the application if you get error Service plan hdi-shared not found

For that you need to activate the services. For that you need to go to the **Entity Assignment** which is inside the **Entitlements.** After that you need to click on **Add service plans.** After that you need to search hana. It will show the services : **SAP HANA Cloud, SAP HANA Schemas & HDI Containers, SAP HANA Schemas & HDI Containers Trial**. Make sure all the services are added. Once all the services are added this issue will resolve. The below service and plans should be added.

Service	Service Technical Name	Plan	Set Quota Limit	Subaccount Assignment	Remaining Global Quota	Actions
SAP HANA Cloud	hana-cloud-trial	hana-cloud-connection	☐	1 shared units	1 shared units	🗑
		hana	☐	1 shared units	1 shared units	🗑
		relational-data-lake	☐	1 shared units	1 shared units	🗑
	hana-cloud-tools-trial	tools (Application)	☐	1 shared units	1 shared units	🗑
SAP HANA Schemas & HDI Containers	hana	hdi-shared	☐	10 shared units	10 shared units	🗑
		schema	☐	10 shared units	10 shared units	🗑
		sbss	☐	10 shared units	10 shared units	🗑
		securestore	☐	10 shared units	10 shared units	🗑
SAP HANA Schemas & HDI Containers Trial	hanatrial	securestore	☑	— 1 + Units	0 units	🗑

Now if you don't want to deploy your local data into btp how will you do that ?

So the local data which you have in the csv format, you don't want to deploy in the btp, for that in the **mta.yaml** file you need to add commands :
npx rimraf gen/db/src/gen/data **npx rimraf gen/srv/sdc/db/data**

```
commands:
  - npm install --production
  - npx -p @sap/cds-dk cds build --production
  # To stop uploading local Data
  - npx rimraf gen/db/src/gen/data
  - npx rimraf gen/srv/sdc/db/data
```

When I try to do cds watch its throwing error why ?

So after added the xsuaa, in the **package.json** file it is showing that the auth : xsuaa so that's why if I do **cds watch** it shows the error **Authentication kind "xsuaa" configured, but no XSUAA instance bound to application.** To make it work we need to create **default-env.json** file for that we need to run the command **cf default-env {srv module name in the mta.yaml file}.** After run this command the **default-env.json** file will be created. And after that the **cds watch** will work. But make sure that the default-env is installed, after that only you can run the command.

Another option we can do to work the command **cds watch,** we can delete the two attribute **db, auth.**

```
"requires": {
  "db": "hana",
  "auth": "xsuaa"
}
```

Managed & Unmanaged AppRouter

So when you go to the mta.yaml file, in the module section if you see a **AppRouter** module. That means AppRouter is created by us and we are handling this AppRouter. So basically we create the AppRouter manually by entering the command. That is called **Unmanaged AppRouter**.

So when you go to the mta.yaml file, in the module section if you don't see a **AppRouter** module. That means AppRouter is created by SAP and also handling by SAP. That is called **Managed AppRouter**.

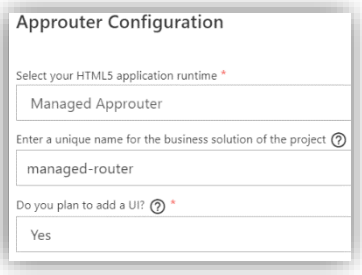
In the Unmanaged AppRouter after deploy the application, in the **Applications** section the AppRouter module will show. It will show Service, AppRouter, db-deployer.
In the Unmanaged AppRouter to access the Entity-Data, we will got to AppRouter link and from there we will directly access the Entity-data.

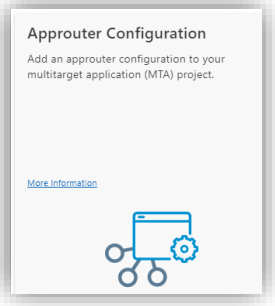
In the Managed AppRouter after deploy the application, in the **Applications** section the AppRouter module will not show. Only it will show Service and db-deployer. In the Managed AppRouter to access the Entity-Data we need to get the clients credentials to create the auth-token and with the help of the auth-token we will access the Entity-data.

Implementation of Managed AppRouter : So we created a new directory called managed-approuter, and inside that we initialize a cloud project by using the command **cds init.** After that we created a demo OData service by running the command **cds add tiny-sample.** So it created a dummy book entity. After that we **deployed it to SQLite**. After that we will add hana database by using the command **cds add hana.** After that we will install xssec module by running the command **npm install @sap/xssec passport.** After that we will add mta file by using the command **cds add mta.** After that in the mta.yaml file we need to add 2 commands Basically we will use managed approuter most of the cases. We don't use much unmanaged approuter. Unmanaged approuter will be used like when on the top of the approuter we want to perform some action. Like I want to execute some code, before execute the approuter, then we will go for the unmanaged approuter. So whatever the router ar controlled by the sap they are called managed app-router.

```
commands:
  - npm install --production
  - npx -p @sap/cds-dk cds build --production
```

Creation of Managed AppRouter : So we will right click on the **mta.yaml** file and we will choose the option **Create MTA Module from Template.** After that we will select the **Approuter Configuration** and click om **Start**. After that we will select **Managed Approuter** from the dropdown. And Last dropdown value we will select **Yes**. After that click on Next. So it will add Managed approuter. Now if you go to the **mta.yaml** file you will see in the resources section it will create a resources name `managed-approuter_html_repo_host, managed-approuter-destination-service` so this 2 resources for UI. And also `uaa_managed-approuter` this resource they have added for approuter. And they have added modules `managed-approuter-destination-content` for UI.





```
modules:
- name: managed-approuter-srv ···
- name: managed-approuter-db-deployer ···
- name: managed-approuter-destination-content ···
resources:
- name: managed-approuter-db ···
- name: managed-approuter-destination-service ···
- name: managed-approuter_html_repo_host ···
- name: uaa_managed-approuter ···
```

So after created the manage approuter successfully, now you don't need to think of handling the approuter by yourself. Sap will take care of this things.

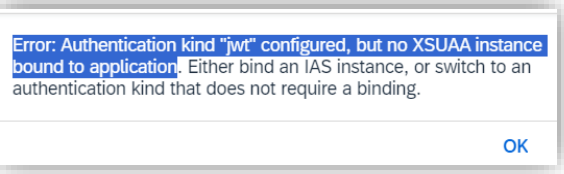
We will do **cf loging.** After that we will built the project using the command **mbt build.** But before building make sure that in the mta.yaml file this 2 commands should be the configured : `npm ci` `npx cds build --production` otherwise it will throw error

```
parameters:
  enable-parallel-deployments: true
build-parameters:
  before-all:
  - builder: custom
    commands:
    - npm ci
    - npx cds build --production
```

After building is done, now we will deploy our project. So for that we will go inside the mtar file, which is inside the **mta_archives** folder. After that we will right click on the **mtar** file and select the option **Deploy MTA Archive.**

I got error during the deployment, the error is : Error starting application "managed-approuter-srv": Some instances have crashed. Check the logs of your application for more information.

So I went to the **managed-approuter-srv** and there in the logs I saw the error



So for that I went to the **mta.yaml** file and there I went to **managed-approuter-srv,** and there in the requires section I added the **uaa_managed-approuter** resource. Previously there was no approuter added, that's why error is showing. Now I added the resource.

```
- name: managed-approuter-srv
  type: nodejs
  path: gen/srv
  requires:
  - name: uaa_managed-approuter
  - name: managed-approuter-db
```

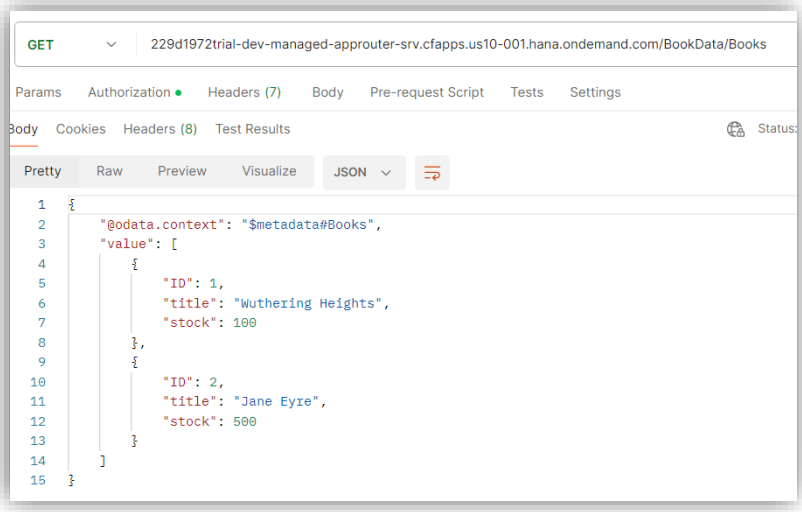
And another thing in the **package.json** file there was not **xsuaa authentication** defined, that's why there also I have added the auth as xsuaa.

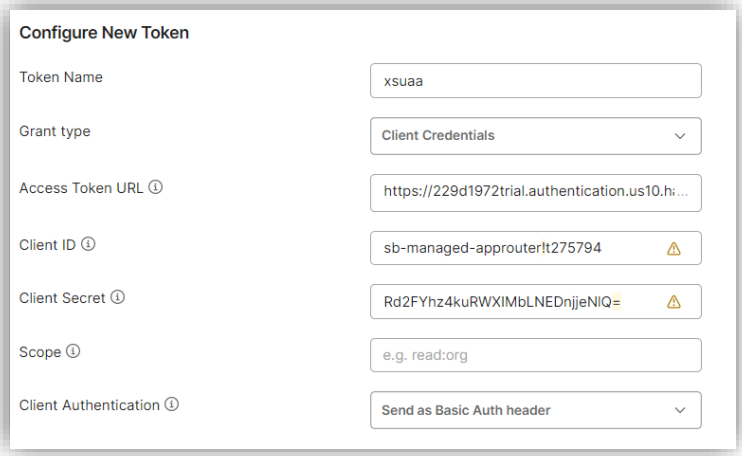
```
"cds": {
  "requires": {
    "db": "hana",
    "auth": "xsuaa"
  }
}
```

Now the application deployed successfully. After deployed we can see that it only created the 2 application : managed-approuter-srv managed-approuter-db-deployer but it did not create the app-router.

And also it created the instance : managed-approuter-db managed-approuter-destination-service managed-approuter-html5-app-host-service managed-approuter-xsuaa-service

Now to access the entity data, we have to go to the instance **managed-approuter-xsuaa-service** from there I have to fetch the client-Id, client-secret, client-url and we have to go inside the service managed-approuter-srv and from there we will copy the application routes link. And after that I will go to the **postman** and I will create the auth-token and after that I will access the entities and their data.



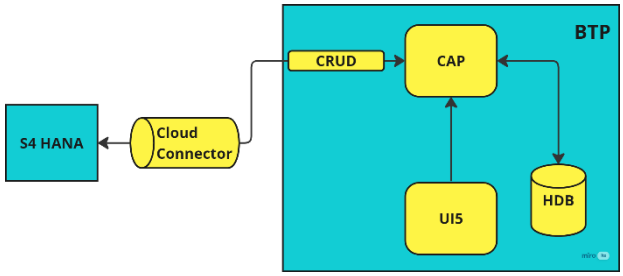


Consume external application in SAP CAP

Let say I have some code available in S/4 HANA System, and which is my old code. Now I want to add some new code on the top of this, basically I want to enhance and for that I will write BTP code for that. And I don't want to lose or I don't want to scrap my old code. So we can do extension in this case.

So suppose I have a **CAP** application in **BTP** and also I have a **UI** application. So basically the UI application connected to my CAP application. And also I have old **S4-HANA** code with me. So now I want to execute the old code from CAP. So from the CAP we can do CRUD operation to the S4 HANA to execute the old code or Also we can do vice-versa. So both way this operations are possible.

So to connect the **CAP** with **S4 HANA,** we have something called Tunnel, because you can not directly connect with S4 HANA there has to be a connection, so this is called **Cloud Connector.**



So what is Cloud Connector ?

Cloud Connector is an application that can be installed on a Windows, Linux, Mac OS operating system, which creates a secure connection between the SAP "cloud" and on-premise system.

So in BTP when you go inside your subaccount, then in the left side you will see a **Connectivity.** Inside that there will be Destinations and Cloud Connectors.

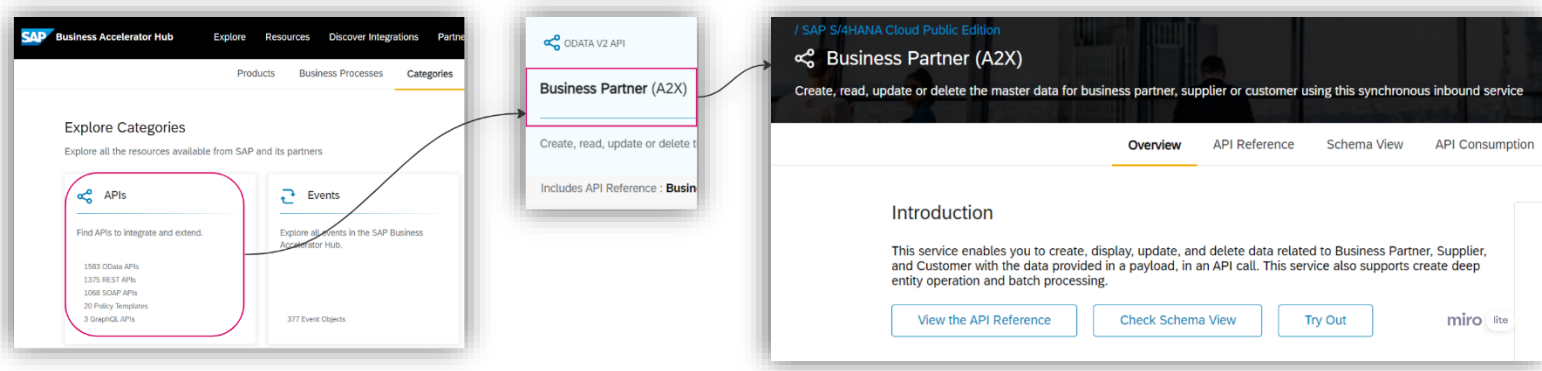
So if we want to connect sap related things with the sap cloud, then we need Cloud Connector and Destination both.

But if we want to connect non-sap related things with the sap cloud, then Destination is more than enough.

Sap built many standard services and all this kept in a place called **sap business accelerator hub** https://api.sap.com/ So now lets assume my company need a service which is present in this place, so rather creating from scratch I will reuse this in my company.

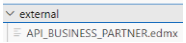
So now I want to use a API from this sap business accelerator hub. I want to use Business Partner API

So to get that First we will go to the **Categories** section. And there I will go inside the **APIs** and there I will search business partner and result will come. So I will use **Business Partner (A2X) [ODATA-V2 API].**



So I will go inside this API. There you will see Overview, API Reference section of this API. And in the API Reference section you will see all the Endpoint of this Business Partner service. And also we can execute the endpoint, because they have provided **Swagger UI**.

Now I want to consume this service into our cap service. So to consume this one we have something called **metadata** file. So in the Overview section if you come down you will see there is something called **API Specification,** so this is nothing but you metadata file. So this API Specification we can download. It has 3 format JSON, YAML, EDMX. So most of time we use EDMX format, that's why we will download the EDMX format.

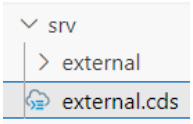
So now I will go to my directory : **SAP-CAP-EXTERNAL-API** which is created on the dev-space : **CAP.** And first I will go inside my directory and will do **npm install.** So the EDMX file which we downloaded that I want to have in the directory. So for that I will create a folder inside my directory name **external.** And in this folder I will keep my **edmx** file 

Now I will import this **edmx** file into my service for that the command is **cds import {relative path of this edmx file}**
So in my case the command is **cds import external/API_BUSINESS_PARTNER.edmx**
After run this command inside the **srv** folder, a new folder got created named **external.** And inside this one **csn** file and one **edmx** file created.

So in the **csn** file we have the entity attribute details in csn format. So CAP basically use this **csn** file to consume this things inside the file.

And also in the package.json file it got added some piece of code.

```
"cds": {
  "requires": {
    "API_BUSINESS_PARTNER": {
      "kind": "odata-v2",
      "model": "srv/external/API_BUSINESS_PARTNER"
    }
  }
```



Now in the srv folder I will create a cds file name **external.cds** basically to define the service. Now in the **external.cds** file I will import the `API_BUSINESS_PARTNER.csn` file which is inside the external folder of srv folder. Now I will define a service name as **External** service. And I will expose an Entity which is `A_BusinessPartner.` So in the Business-Partner API this entity are present. And That entity has many fileds but I don't want to expose all the fields, so some selected fields I will expose. And I have to give the URL of this Service in the package.json file, so that attribute name will be

```
using {API_BUSINESS_PARTNER as external} from './external/API_BUSINESS_PARTNER';

service ExternalService {

    entity A_BusinessPartner as
        projection on external.A_BusinessPartner {
            BusinessPartner,
            Customer,
            BusinessPartnerFullName,
            BusinessPartnerGrouping,
            BusinessPartnerUUID,
            OrganizationBPName1
        };
}
```

```
"cds": {
  "requires": {
    "API_BUSINESS_PARTNER": {
      "kind": "odata-v2",
      "model": "srv/external/API_BUSINESS_PARTNER",
      "credentials": {
        "url": "https://sandbox.api.sap.com/s4hanacloud/sap/opu/odata/sap/API_BUSINESS_PARTNER"
      }
    }
  }
}
```

So in the swagger UI, If I execute any one endpoint, from there I will cpoy the URL

Now I will execute this using the **cds watch** command.

So here when I try to execute the Entity, then it will not work it will throw below error.

```
'<error xmlns="http://docs.oasis-open.org/odata/ns/metadata">
   <code>501</code>
   <message>Entity "ExternalService.A_businessPartner" is annotated with "@cds.persistence.skip" and cannot be served generically.</message>
 </error>
```

So basically the CAP service is consuming the external service. But CAP does not know what service it is, So that's why data is not loading. So to solve this problem we will write **on** event. So for that we will create the **external.js** file next to the cds file.

```
const cds = require("@sap/cds")
module.exports = cds.service.impl (async function(){

    //Connect the external service
    const bp = await cds.connect.to("API_BUSINESS_PARTNER");

    this.on("READ","A_BusinessPartner", async (req)=>{
        //Code implementation
        return bp.run(req.query)
    })
})
```

So first we will import the **@sap/cds.** After that we will export the **impl** function. We will connect to the external service which name is API_BUSINESS_PARTNER. And we will get this name from the **external** folder. After that we will write the **on** event. So basically while reading the entity **A_BusinessPartner** this on event will be triggered and a callback back function will be executed. So we whatever request comes from the user that will execute and return. So **req.query** means the query which user are passing.

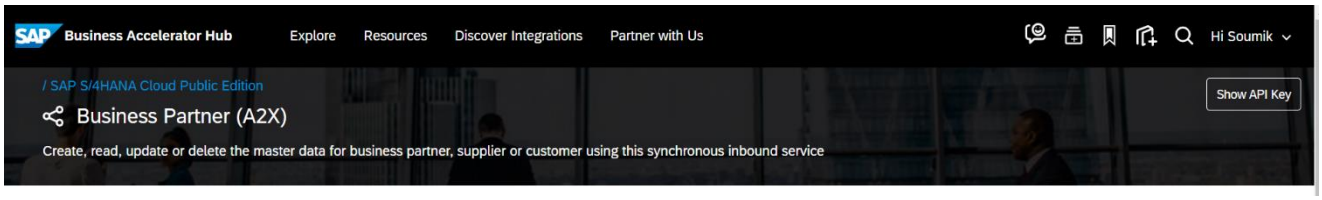
After implement this code, when we will do cds watch we will get and error **Cannot find module '@sap-cloud-sdk/resilience'** to resolve this issue we need to install this module, so the command would be **npm install @sap-cloud-sdk/resilience.**

After that when we try to access the entity it will show again error :
Cannot find module '@sap-cloud-sdk/http-client' so we need to solve this issue, so for that we will run the command **npm install @sap-cloud-sdk/http-client**

Now again it will show a new error, while accessing the entity, the error is displayed below

```
<error xmlns="http://docs.oasis-open.org/odata/ns/metadata">
  <code>502</code>
  <message>Error during request to remote service: Request failed with status code 401</message>
</error>
```

So here we are trying to access the External services from the CAP, but without authentication or authorization we can not access any services. Authentication is required. So for that reason there is something called **API Key** which is present in the external service. We need to fetch that key and we need to use that key in the package.json file.




```
"credentials": {
  "url": "https://sandbox.api.sap.com/s4hanacloud/sap/opu/odata/sap/API_BUSINESS_PARTNER",
  "headers" :
  {
    "APIKey" : "GN8g3E5vgOoHfjbuqn1dXCMgxTkWm11a"
  }
}
```

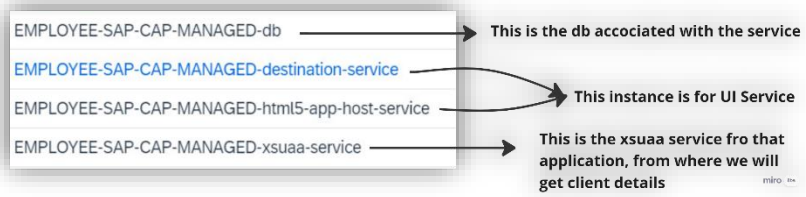
After doing all this changes now we can successfully see the entity data. Thus we can access the external data.

So We have created a new directory called **EMPLOYEE-SAP-CAP-MANAGED** in the CAP dev space. So basically we have created a managed approuter, that means approuter is managed by SAP. So we build this service using the command **mbt build.** And after that we deployed the service, for that right click on the mtar file and choose the option **Deploy MTA Archive.**

After deployed the service, now in BTP, Application section we can see it created 2 application : 1st **db-deployer** and 2nd **srv**

Here Approuter did not created, as it is managed approuter

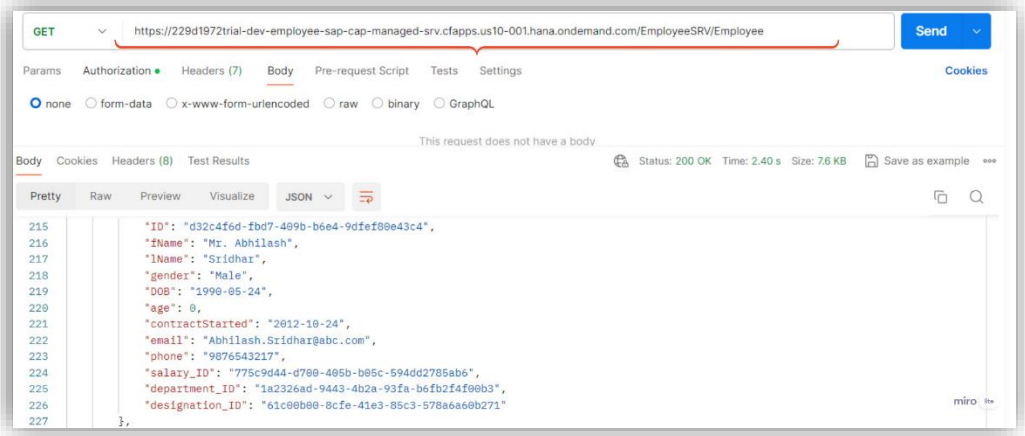
After deployed the service, now in BTP, Instance section we can see it created Instances :



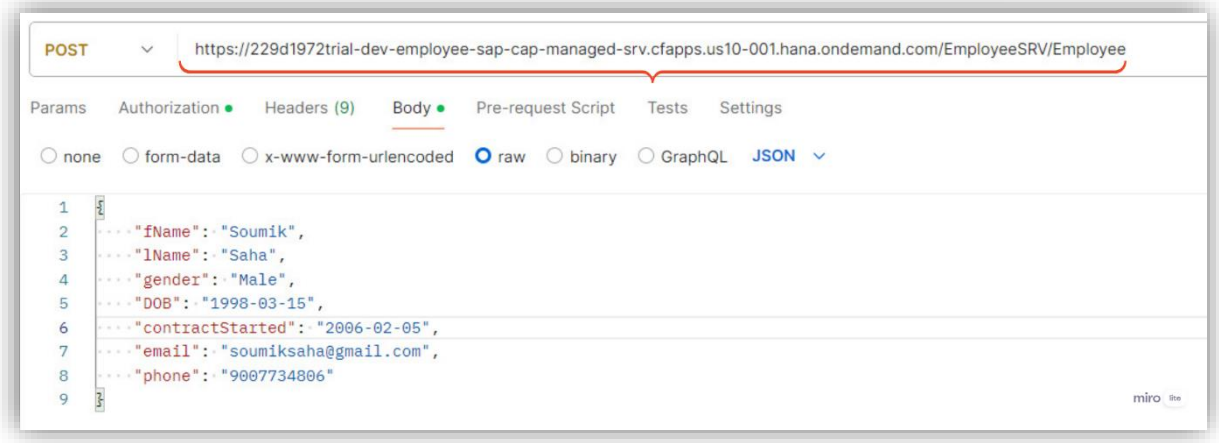
we can do CRUD operation to the entities which are stored in Cloud, from locally through POSTMAN. For that we need to create the **default-env.json** file. After that we will run the **cds watch** command, when the **cds watch** is running then we can do CRUD operation.

So we have employee entity, which is deployed in the cloud. Now I can create a new employee locally through POSTMAN. That means we can access the BTP layer locally

So first of all if the default-env.json is not present, we will create that. Once it creates after that we will run the cds watch, it will work perfectly as we have default-env.json file. After that we will go to the postman and there we will create the auth-token with the help of client-credentials. And we will pass the URL which is for create the employee. And the URL would be the Cloud URL as you can see in the below picture. Thus we can access the BTP layer locally.



You can do **UPDATE** or **DELETE** operations also



Now we will see how to add Role for Authentication in local system, not in BTP

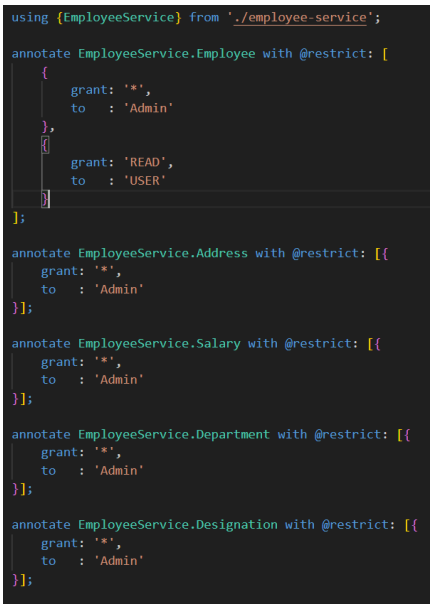
I have all the entities and also have the entities data, now I want to add some restriction while accessing the entity-data.

So the file, where we will expose the entity, that file we will add the restriction . So I have the directory name **EMPLOYEE-SAP-CAP-MANAGED** inside that I have the **srv** folder inside that **employee-service.cds** file present. So there I will add my restriction.

```
using employee.details as db from '../db/data-model';

@path: '/EmployeeSRV'
service EmployeeService {

    @restrict: [{
        grant: '*',
        to   : 'Admin'
    }]
    entity Employee    as projection on db.Employee;
    entity Address     as projection on db.Address;
    entity Salary      as projection on db.Salary;
    entity Department  as projection on db.Department;
    entity Designation as projection on db.Designation;
}
```

So this is my **employee-service.cds** file and I have exposed the entity. And also I have added **restriction.** So it is telling that grant all the operations like CREATE, READ, UPDATE, DELETE. For a user who is having Admin scope. As this restriction are defined above the Employee entity, that's why it is only applicable for Employee entity.

If we want to restrict all the entity, for that I will create a another file in the **srv** folder name **ServiceRestriction.cds** and there I will defined the restriction for all the entity.



```
using {EmployeeService} from './employee-service';

annotate EmployeeService.Employee with @restrict: [
    {
        grant: '*',
        to   : 'Admin'
    },
    {
        grant: 'READ',
        to   : 'USER'
    }
];

annotate EmployeeService.Address with @restrict: [{
    grant: '*',
    to   : 'Admin'
}];

annotate EmployeeService.Salary with @restrict: [{
    grant: '*',
    to   : 'Admin'
}];

annotate EmployeeService.Department with @restrict: [{
    grant: '*',
    to   : 'Admin'
}];

annotate EmployeeService.Designation with @restrict: [{
    grant: '*',
    to   : 'Admin'
}];
```

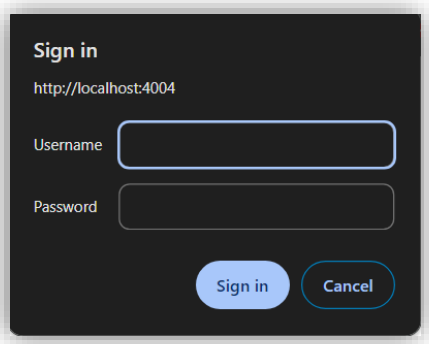
So here as you can see this is my **ServiceRestriction.cds** file. And we have set the restrictions for all the entities.

So here for the **Employee** entity we have set 2 restrictions.

1st is the user which is having the **Admin** scope can access and do all the operations[CRUD] with the entity-data.

2nd is the user which is having the **USER** scope can access do read operations with the entity-data.

After adding the restrictions, Now we will run this application locally with the help of **cds watch.** After that when we try to access the Entity, it will show a alert box with username and password. So we have to enter the correct username and password and also scope of that user should match with the defined **restrictions.**



So this alert box will show, when we try to access the Entity.
So for that we need to create the user. I mean we need to create the dummy user for testing purpose.

We will create the User and Role associated with the user, for accessing the Entity-data

So we have added the restriction and we have given the condition that the user which is having the Admin scope, can access the entity and do the **CRUD** operation. So that why we will create dummy user which is having the Admin role.

So in the **package.json** file we will create the dummy user, in the **auth** section.

So here as you can see that in the **package.json** file we have added the user, in the **auth** section.

"kind": "mocked" : kind means :- type of the user. And mocked means :- dummy user. And we have defined **password**, **ID** of the user. And we have defined the roles associated with that user.

So here we have defined 2 users, 1st one associated with the Admin role. And 2nd one is associated with the USER role.

Now after implemented all this, When we try to access the entity, it will ask for username and password. So we will pass the correct username and password which satisfy the restriction condition, that means the role should match, which defined in the restrictions.

Now we will create a **.http** file to do the operations, inside the **srv** folder

So now I will create a test folder inside my directory. And inside that test folder we will create a file with **.http** extension. So created a file named : **EmployeeCapManaged.http**

So here as you can see I defined 2 GET operations and 1 POST operation.

And for every operations I have to pass the Authorization. So basically the format of passing the authorization would be ---> **Authorization: Basic username:password**

So instead of writing like this above, we encoded this Authorization to Base64. And for that we need to go https://www.base64encode.org/ There we need to encode the Authorization. After encoded them we kept that in variable. In this picture may be you can see in the top we created the variable **@SoumikCredentials @AlaxCredentials** and kept the encoded value inside the variable. And where we have to pass the Authorization we will pass the variable name. So this variable is here used for reuseable purpose.

While creating the employee, we have used **Soumik credentials** as authorization, because Soumik has the permission to do all the CRUD operations, But **Alax credentials** only have permission to READ operation. So if we use Alax credentais it will throw error.

xs-security.json [Both same implementation for managed and unmanaged approuter.]

```
{
  "xsappname": "employee-sap-cap-managed",
  "tenant-mode": "dedicated",
  "scopes": [],
  "attributes": [],
  "role-templates": [],
  "role-collections": [],
  "oauth2-configuration": {
    "token-validity": 3600,
    "redirect-uris": [
      "https://*/**"
    ]
  }
}
```

So this is my xs-security.json file, which is empty. I will implement this file. So in this file

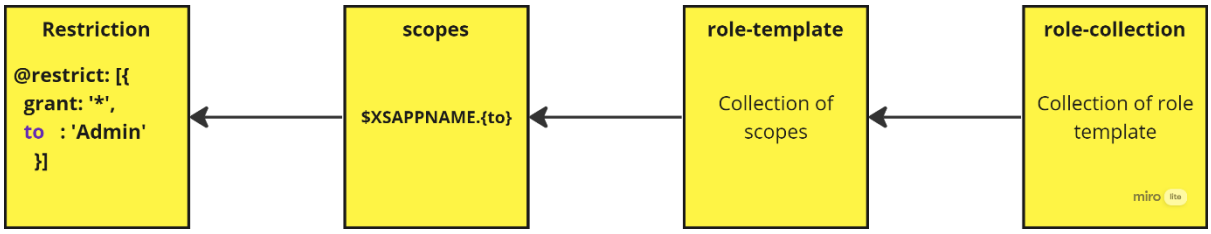
scopes : [] - the scopes property accepts an array. You can define multiple scopes. Each scope has a **name** and a **description**.

```
"scopes": [
  {
    "name": "$XSAPPNAME.Admin",
    "description": "edit"
  }
],
```

The value of the **name** attribute is

role-templates: [] – So this property accepts an array. Basically role-templates is the collection of scopes.

role-collections:[] -- So this property accepts an array. This role-collections is the collection of role-templates.



Now we will implement the xs-security.json [both same for managed & unmanaged approuter]
Now we will implement the **xs-security.json** for adding the application security in cloud.

```
"scopes": [
  {
    "name": "$XSAPPNAME.Admin",
    "description": "edit"
  },
  {
    "name": "$XSAPPNAME.USER",
    "description": "read"
  }
],
```

So in the **xs-security.json** file we have added this 2 scopes, which will apply for the application. So basically one Scope is **Admin** and another one is **USER**.

So this 2 scope we have written from the **restrictions,** which we defined for the entity.

```
"role-templates": [
  {
    "name": "Viewer",
    "description": "Read the data",
    "scope-references": [
      "$XSAPPNAME.USER"
    ]
  }
],
```

So basically the role-template is the collection of **scope.** We define multiple scope in the **role-template.** As you can see we have defined **USER** scope in the **scope-references**. Like this we can define multiple scope in that **scope-references** section. And name of the role is **Viewer.**

```
"role-collections": [
  {
    "name": "Employee_role_collection",
    "description": "Employee roles",
    "role-template-references": [
      "$XSAPPNAME.Viewer"
    ]
  }
],
```

And in the role-collections is the collection of role-template. We can define multiple role-template in the role-collection **role-template-references** section

```
{
  "xsappname": "employee-sap-cap-managed",
  "tenant-mode": "dedicated",
  "scopes": [
    {
      "name": "$XSAPPNAME.Admin",
      "description": "edit"
    },
    {
      "name": "$XSAPPNAME.USER",
      "description": "read"
    }
  ],
  "role-templates": [
    {
      "name": "Viewer",
      "description": "Read the data",
      "scope-references": [
        "$XSAPPNAME.USER"
      ]
    }
  ],
  "role-collections": [
    {
      "name": "Employee_role_collection",
      "description": "Employee roles",
      "role-template-references": [
        "$XSAPPNAME.Viewer"
      ]
    }
  ],
  "oauth2-configuration": {
    "token-validity": 3600,
    "redirect-uris": [
      "https://*/**"
    ]
  }
}
```

So this is the whole **xs-security.json** file. So we have given the xsappname. And we have defined 2 scopes. This 2 scopes came from the restriction, which we defined in the **ServiceRestriction.cds** file. So one scope is Admin who is having permission to do all **CRUD** operation. And another scope is USER who is having permission to do only **READ** operation.

role-template which is the collection of scope. So we have defined only USER scope in the role template. Which is only have the permission to READ.

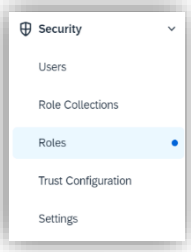
Another is role-collections which is the collection of role-templates. So we have defined the USER role template.

Removed the user details from the **auth** section, which is mentioned in the **package.json** file. And instead of that write **“auth” : “xsuaa”.**

And one thing you remember that suppose you have more than one project-directory. And inside the both project directory, in the **xs-security.json** file if you defined same **xsappname**. Then it will through error, because **xsappname** should be unique.

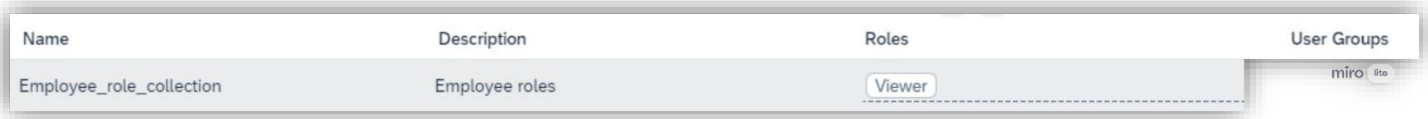
Now we will deploy our application.

After deploy we will see go the BTP, So there if you go inside the sub-account you will see **Security** folder. Inside that security folder **Roles, Role Collection** etc will be there.

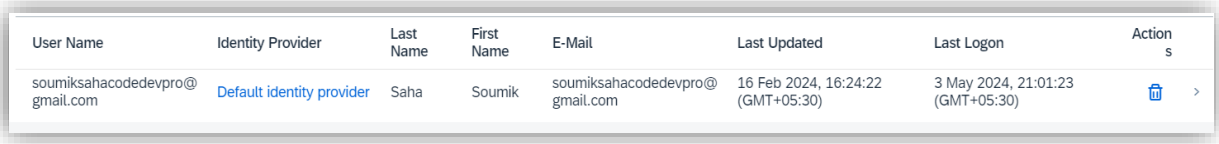


Now if you go inside the **Roles** you will see the role which I created in the **xs-security.json** name is **USER.**

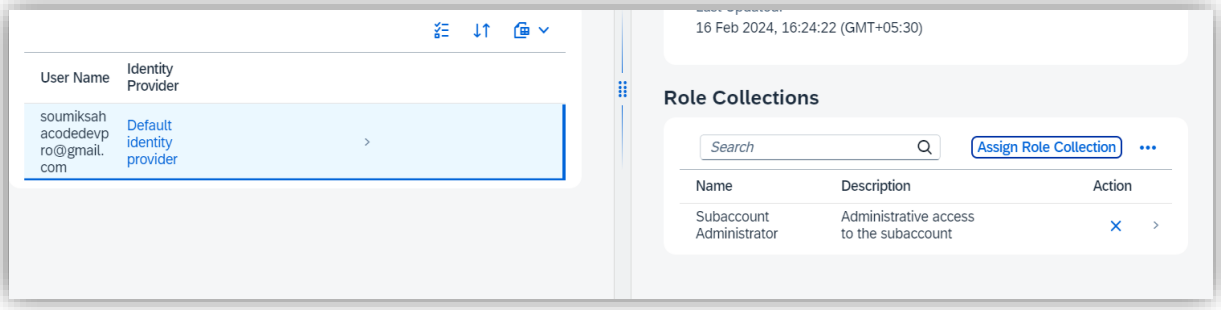
If you go inside the role-collections, you will see the name of the role-collection and the Roles is **USER**, which associated with role-collections.



Now you need to assign this role-collection to the user. So to assign this you need to come to the Users, which is inside the Same folder only i.e. **Security** folder. Once you come to the User section you will see a user.



After that you will click on the user, it will navigate to another view. And to assign role-collection with the user for that you will click the option **Assign Role Collection,** which you will see in the right side. After that you will click on that and select the name of the role-collection and assign it.



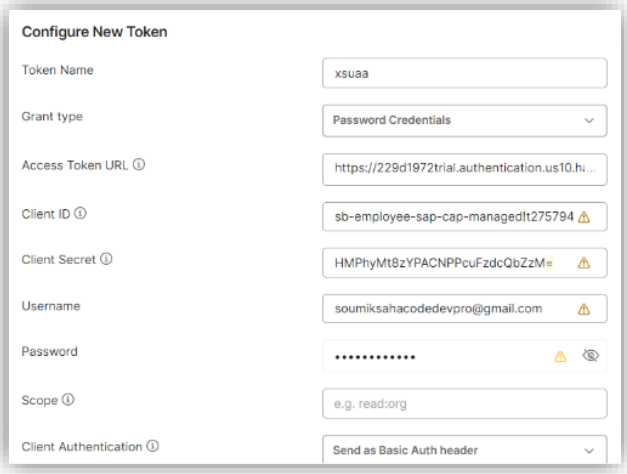
Now role-collection got assign to the user.

So the user have only the read options available. That means user only can access the entity, but can not do other option on the entity.

We will access and do operation on the entity from postman:

Now first we will create the auth-token, with the help of client-id ,client-secret, URL and with the help of that auth-token I will try to access the Employee entity from the **srv**. But it will show me **forbidden** in the response as because after assign the **role-collections** to user Now to create the auth-token we have to pass username and password also in authorization.

So for that we have to select **Grant Type as Password Credentials.**



So here in this picture as you can see we have choose **Grant type** as ------> **Password Credentials**.

And after that we have defined Access Token URL, Client ID, Client Secret, Username, Password

And with this above attribute I have created the auth-token and used that token to access the entity-data. Now we will be able to access the entity-data successfully.

Now I will try to create new employee, it will show me error-message **Forbidden,** because this user has only read permission, that's why this user cannot create employee.

Now if you want that the user want to do all the CRUD operation, for that we need to modify our **xs-security.json** file basically we need to add the scope to the role-template, and that scope should have all the permission.

```
"role-templates": [
  {
    "name": "USER",
    "description": "Read the data",
    "scope-references": [
      "$XSAPPNAME.USER",
      "$XSAPPNAME.Admin"
    ]
  }
],
```

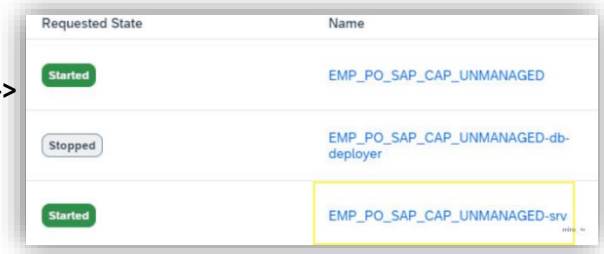
So as you can see here I am adding the another permission which name is **Admin.** And this scope having all the permission. After that I will build and deploy this modified code, and after that you will able to create employee. Similar way you will create the token and by using that token you will create employee, like you did for get employee.

```
@restrict: [
    {
        grant: '*',
        to   : 'Admin'
    },
    {
        grant: 'READ',
        to   : 'USER'
    }
];
```

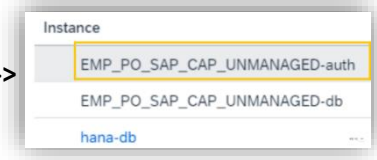
So in the unmanaged app router, how to do CRUD operation

So to do the operation in the unmanaged app router, we have to go to the postman. And in the postman we will do the operation, for that we need to have the URL and for that URL we will refer the Application which name is having the **srv** as postfix. which is present in the Applications section. And we will create the **Jwt** token for that we have to have the client credentials, with the help of that we will be able to create the token and use that. And to get the client credentials we will use the Instance which name is having **auth** as postfix.

srv ---->



Client Credentials --------->



In the unmanaged **approuter** we can add the **scopes**, **role-template**, **role-collections** similar way how we can do in the managed **approuter**.

AFTER ADDING THE ROLES, WHEN WE TRY TO ACCESS THE ENTITY DATA TROUGH THE POSTMAN, THEN WE HAVE TO PASS THE **BASE-URL** IN THE POSTMAN. SO THAT URL WE HAVE TO TAKE FROM THE **SRV,** WHICH IS PRESENT IN THE APPLICATION SECTION.

So I have used 2 databases in my project 1st is local database which is SQLite and 2nd is hana database. Now I want segregate databases based on the condition

So in the **package.json** file I will do the segregate. So if the **db** name is **development**, then it will indicate the **sqlite**. And if the **db** name is **production** then it will indicate the **hana**. For **squlite** we will added the authentication by adding the **mocked user**. For **hana** we will add the authentication by adding the **xsuaa**.

```
"cds": {
  "requires": {
    "db": {
      },
      "[production]": {
        "kind": "xsuaa"
      }
    },
    "[production]": {
      "auth": "xsuaa"
    },
    "[development]": {
      "auth": {
        "kind": "mocked",
        "users": {
          "soumiksaha@123": {
            "password": "password",
            "ID": "soumiksaha@123",
            "roles": [
              "Admin"
            ]
          },
          "chandansaha@123": {
            "password": "password",
            "ID": "chandansaha@123",
            "roles": [
              "USER"
            ]
          }
        }
      }
    }
  }
}
```

So this is the code for segregation of Database.

Now if I do any changes, to deploy the changes to my local database **i.e. sqlite** database, the command would be **cds deploy --profile development**

And to run my local database the command would be
cds watch --profile development

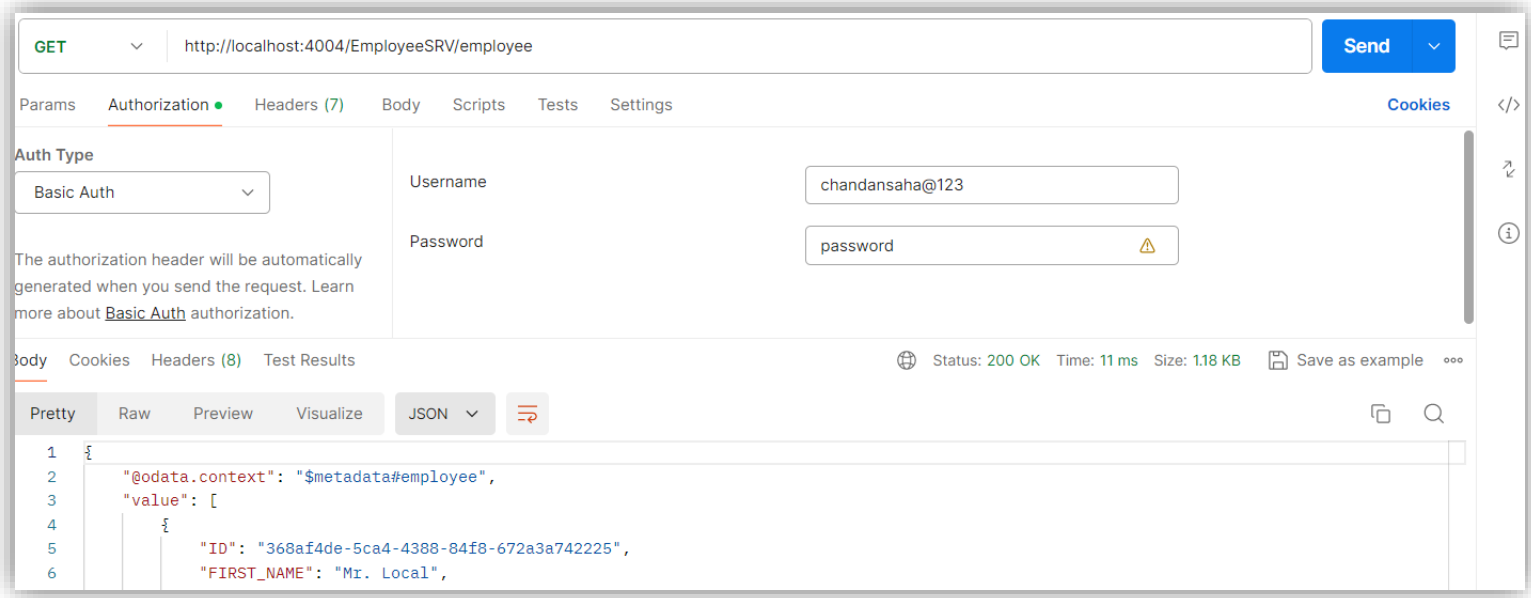
When you try to open the entity data it will ask for username and password. So give the proper credentials, which roles matches with the entity scope.

Now if you want to create data then you will go to postman and will create the auth-token by giving the client-credentials along with username and password.

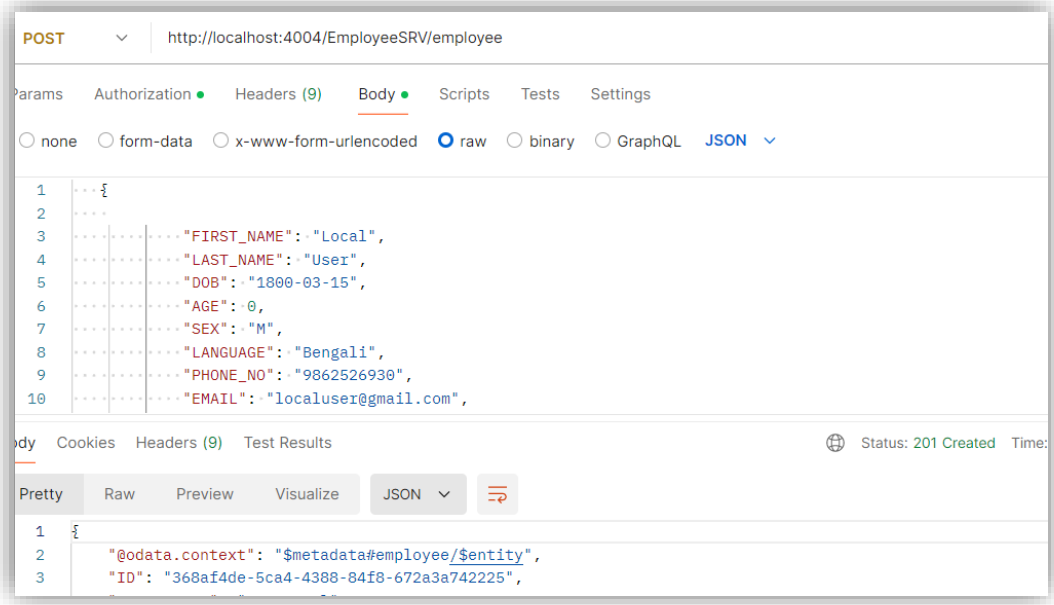
Doing the crud operations in the sqlite database.

Now after adding the Authentication in the SQLite database, now I want to access the data and create the data to the SQLite database. For that I will go to postman. And I will pass the base URL. And the type of the Authorization would be the **Basic Auth** because we will be passing the username and password. And in the base URL the URL would be containing **localhost,** as it is local database SQLite.

Getting the employee from the SQLite



Creating the employee in the SQLite database

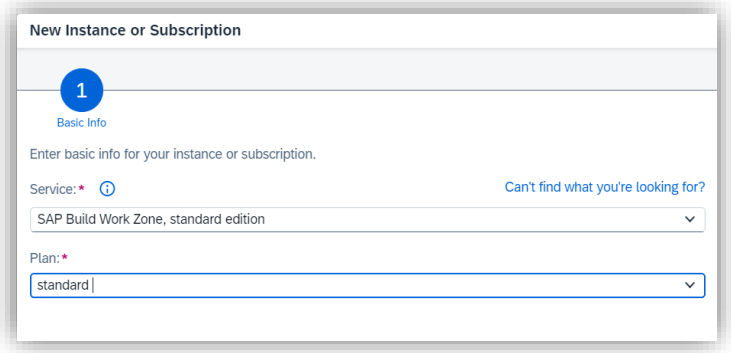


Deploy the fiori application in BTP

After deploy, the deployed UI will appear to the **HTML5 Applications** section. So to go that section, you need to go inside the subaccount. Once you go inside the subaccount in the left side you will see the **HTML5 Application** section. So all the backend services are sitting in one place and the UI are sitting another place. Basically we are segregating the UI and Backend.

So first we have to subscribe the **SAP Build Work Zone, standard edition.** So to subscribe that we have to come to the **Instances and Subscription,** which is inside the **Services** folder. So there you will see a **Create** button. You will click on that button. And after that you will select the Service name from the dropdown and Select the plan also from the dropdown and will click on Create button. So now **SAP Build Work Zone, standard edition** will be subscribed.

If you don't find the Build Work zone in the **Instance or Subscription** & **Service Marketplace** then you need to come to entitlements and from there you need to add this service for the subaccount.



After subscribe the **build work zone** , Now I will go to the role-collection section. In this section we will see some role-collection automatically added.
So this **Launchpad_Admin, Launchpad_Advanced_Theming, Launchpad_External_User** automatically added. Now I will assign this role-collection to the user.

If you don't add this role-collection, then you will not be able to access this **Build work zone.**

So now we will deploy our UI. As we already know that how to deploy the service, so first we will run **mbt build** and after the I will deploy the **mtar file.**