

# HA 450 - XS ADVANCED

(NATIVE HANA DEVELOPMENT SPS04)

Ramana Yellapu

IBM | SAP NextGen UX Team

23 May 2020

## TOPICS COVERED

OVERVIEW

ARCHITECTURE

TOOLS

CONCEPTS

AUTHORIZATION

TUTORIALS– INITIAL SETUP

TUTORIALS ILLUSTRATIONS -END TO END

REFERENCES

## Overview

### How it evolved?

- 1.Load data into SAP Hana- for legacy systems and build Hana web apps [initial target for this landscape]
- 2.XS engine classic- sp01 sps11 to xs advanced sp04 multi tenant (container based) database
- 3.To improve performance - using this setup with load balancing we can achieve.
- 4.Migrate functionality available to implement
- 5.Cloud based app development hana platform
- 6.Platform to build for SAP cloud platform/HCP/SCP/Analytics cloud
- 7.Neo platform to cloud foundry

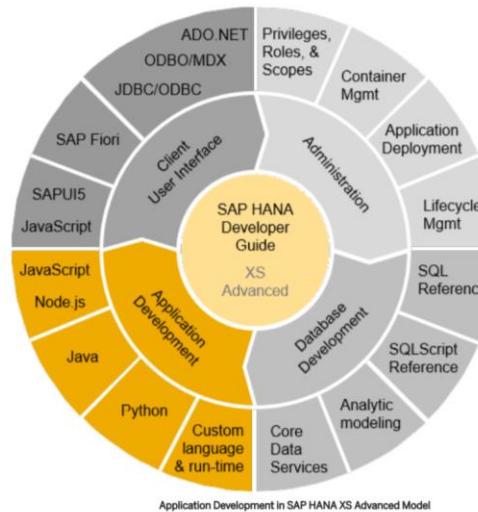
### Why is it Essential?

support extended till 2030 and all services in erp are not yet avaialbe in s4 hana

XS Advanced platform allows to access data for native hana development

Use cases:

1. For Group organization to view all transactions data app for complete overview of all data bases
2. Sap Analytical cloud
3. Sectoral industrial apps- public sector, retail
4. MTA Based (Mutli Target Applications) , below scenario depicts how XS Advanced platform fits in as a HANA System.



### Two approaches for development

Cloud Foundry - cloud environment

XS Advanced - onpremise

## ARCHITECTURE

### Architecture Overview

As illustrated in the following diagram, the basic system architecture has a classic 3-tier approach:

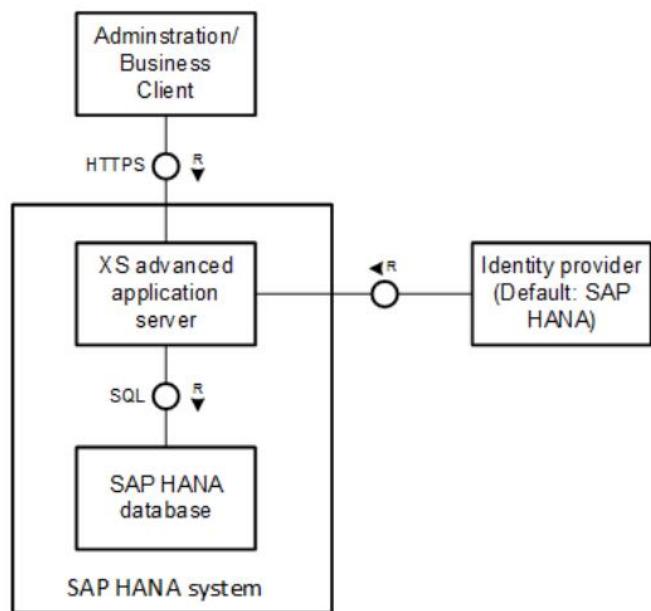


Figure 3: 3-Tier Architecture of SAP HANA with XS Advanced

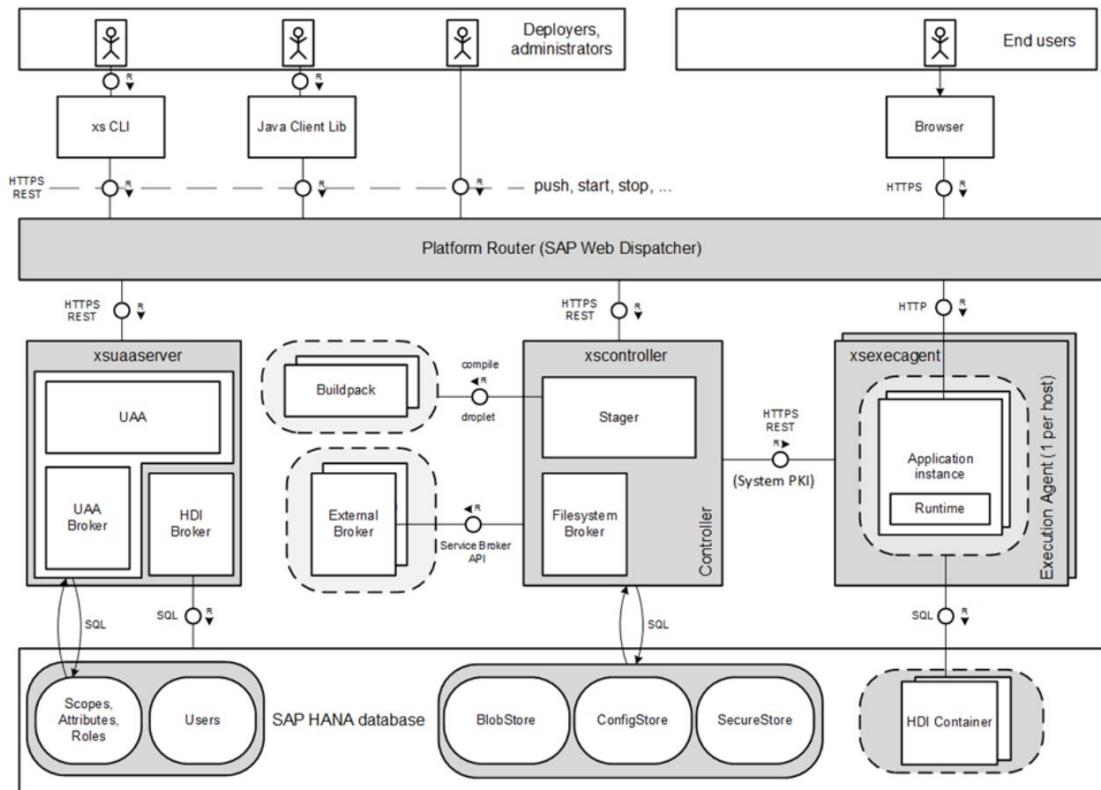


Figure 4: Technical System Landscape of XS Advanced Application Server  
d22.html

## Tools

Typical Development landscape in XS Advanced :

**Front end** - using webide – UI5/HTML5/SAP FIORI

**Application server**- using Fullstack webide – node.js based [python, java can also be used]

**Database layer**- using Hana studio - SQL

### Evolution of Development tools [All 3 ways are available]

1. Hana Studio - stored in Hana package
2. Hana workbench - sap webide - gets stored in hd container
3. Hana XS Advanced based webide - hana development infrastructure [Recommended]

### Can we run on personal laptop?

1. SAP provides SCP platform -cloud foundry(30 days trial) [Link](#)
2. To install own environment on personal laptop (16 GB Ram configuration required).  
Download -> [support.spa.com/swdc](http://support.spa.com/swdc)

You can use **Hana Express edition** if ram > 16 gb on personal laptop – [version supported sp04]

**Hana classic** can be run for less than 16gb as well [version sp01 only]

## CONCEPTS

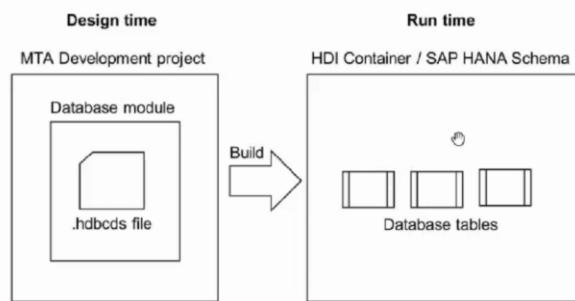
Following concepts are dealt in this topic.

- CDS
- XS Classic vs XS Advanced
- XSODATA
- Node.JS
- HANA Database

CDS:

Two types of CDS are used in SAP Application development.

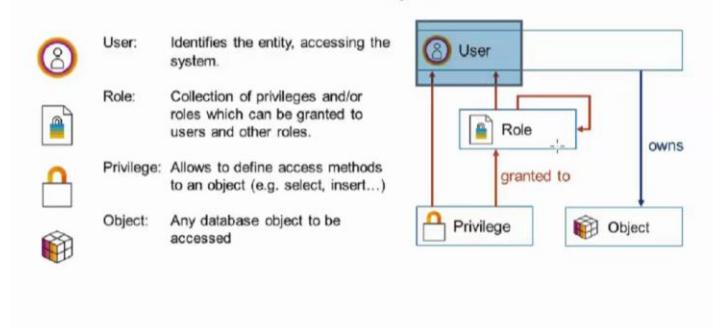
1. ABAP CDS - (ABAP & S/4 HANA- VDM ) ,DDL Views.
2. Native CDS – HANA Development . Here we deal with Native CDS, which is used for XS Application development.



### XS Classic and XS Advanced key difference-

XSA primarily has **HDI Container** which provides efficient persistence in handling database authorization based conflicts.

How HDI makes difference? Below illustration provides a usecase to explain the type of conflicts addressed.



### Be careful when dropping DB users

If user owns no objects (except its own schema)

- Simple drop sufficient

### If user owns further objects

- Need to drop with "cascade" option  
`drop user <name> cascade`
- Drops all objects owned by this account

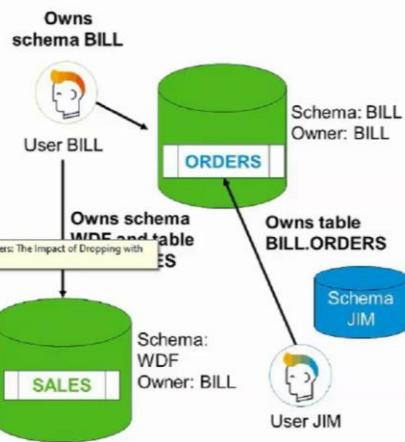
### Examples:

`drop user JIM cascade`

- Removes user JIM
- Removes schema JIM and table BILL.ORDERS

`drop user BILL cascade`

- Removes user BILL



Since HANA released:

HANA Studio:

Attribute View	- Join Engine	(Master Data) 1-D	
Analytical View	- OLAP Engine	(Star join / Star schema)	Analytics
Calculation View	- CV Engine	2-D view	

1.0 SPS9 onwards,

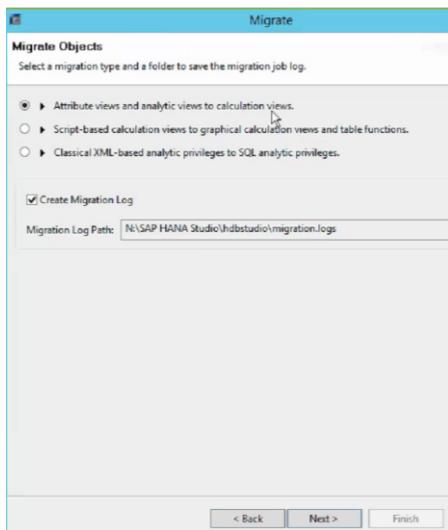
SAP optimized the CV engine to do everything using CV

Calculation View - Sub types

Dimension	(= Attribute_
Cube with Start Join	(=Analytical View

With THE SAP HANA 2.0 SPS## versions,  
we create calculation views in Web IDE

- All Hana studio based developments needs to be migrated to Calculation views.  
In Hand studio option to migrate extension is available to address this.



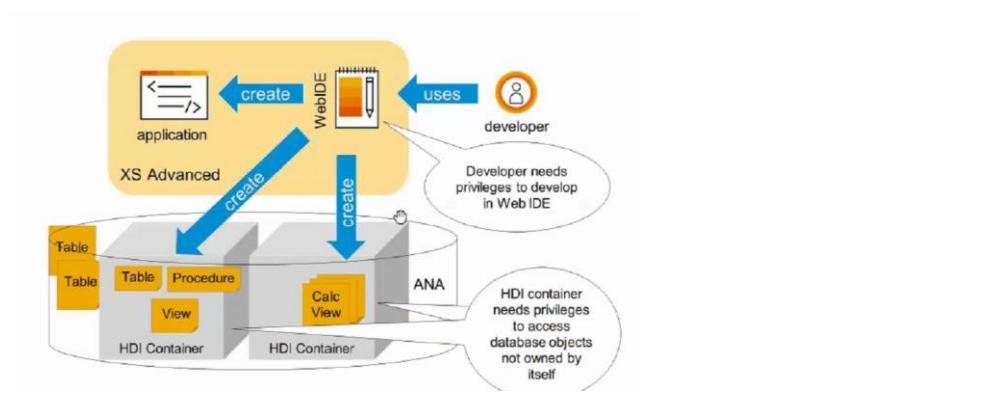
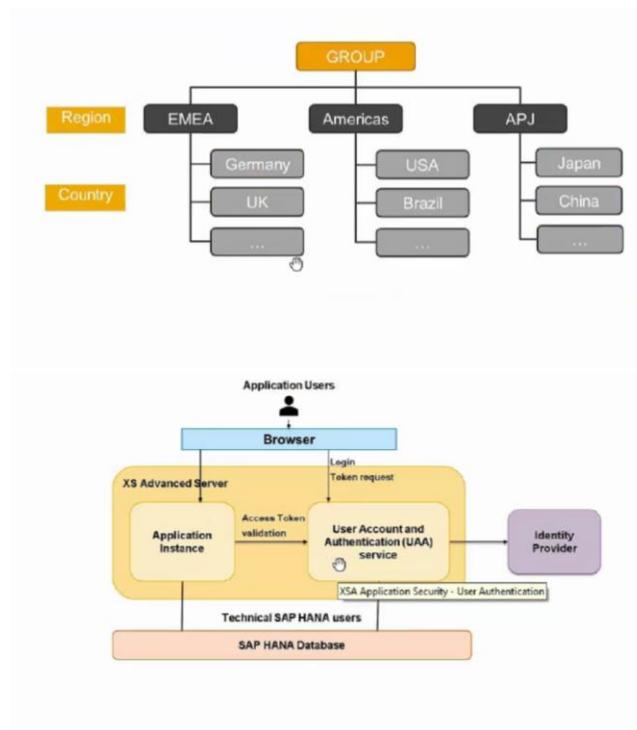
Essentials SP04 onwards all developments are Calculation views based and are available in HDI Container.

### Benefits of Calculation Views

- All calculations are performed on the fly within the database engines.  
No aggregates need to be pre-calculated, and the front-end reporting tools delegate most of the data processing workload (filtering, aggregation, calculations) to the SAP HANA in-memory engines in order to achieve very high performance.
- Reusability  
Each calculation view can be used (referenced) by other calculation views.
- Flexibility  
Calculation views provide a number of features that make them very flexible. For example, defining hierarchies, filtering data, generating prompts for variables and input parameters, and performing currency conversion.
- Adaptability  
An SAP HANA calculation view can adapt its behavior to the list of columns that are selected or projected on top of it. For example, the granularity of a Rank node can

## Authorization

Handling Authorization landscape scenarios pictorial representation .



Authorization handling in multi tenant based data handling- Illustration

User	Schema	Project	User Provisioning
STUDENT09	STUDENT09	Project_1	
SLTUSER	SLT_SCHEMA	Created	Wants to access data from SLT_SCHEMA
SAPABA01	SAPABA01		SLT_SCHEMA_CROSS_ACCESS S_4_Access
Resources			
CROSS SCHEMA		Maintain value. 1. SLT ; 2. S4	
Module		Map the correct pointers	
Create a new hdbgrants			SELECT, CREATE ANY, etc
Create a new synonym (connections- alias) -> table from CROSS Schema			
Create a new CDS Views / Calculation Views based on Synonym			

# Tutorials Initial Setup

## Using the XS Advanced Command Line Interface

1. Use power shell.

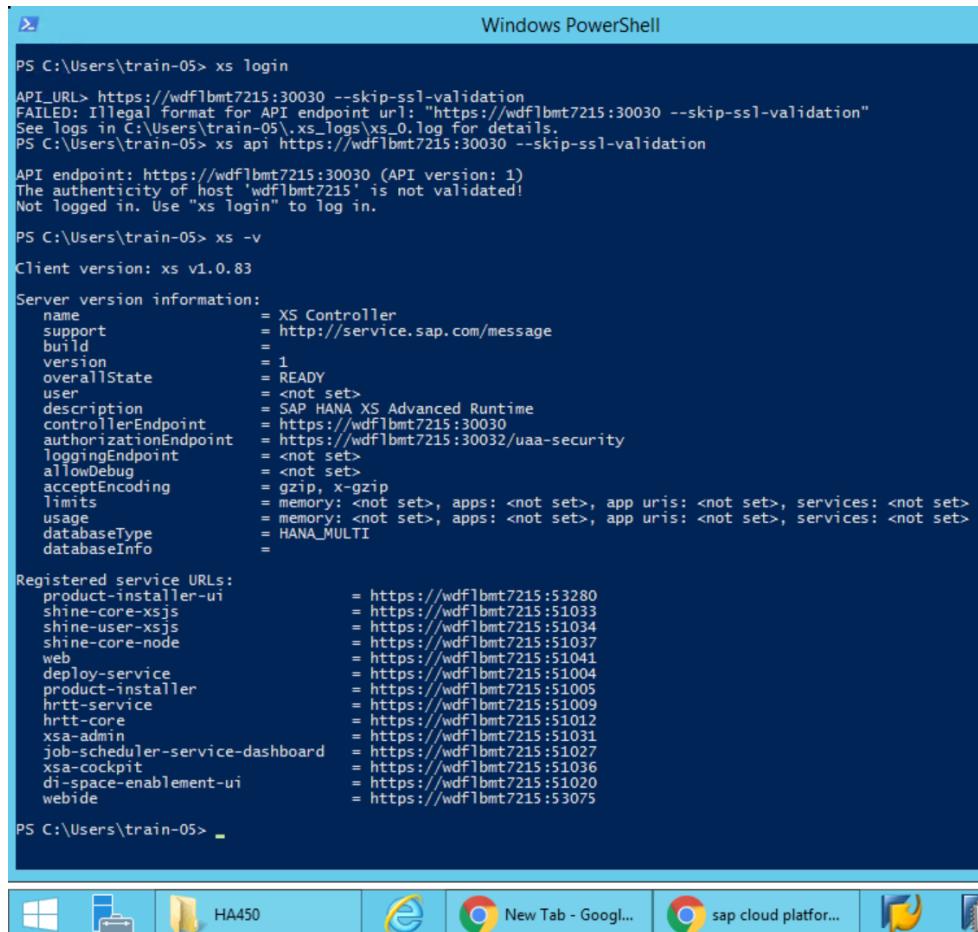
**xs login**

if it asks for api use below command

**xs api**

**xs -v**

**xs v1.0.83**



```
Windows PowerShell
PS C:\Users\train-05> xs login
API URL> https://wdflbmt7215:30030 --skip-ssl-validation
FAILED: Illegal format for API endpoint url: "https://wdflbmt7215:30030 --skip-ssl-validation"
See logs in C:\Users\train-05\.xs_logs\xs_0.log for details.
PS C:\Users\train-05> xs api https://wdflbmt7215:30030 --skip-ssl-validation
API endpoint: https://wdflbmt7215:30030 (API version: 1)
The authenticity of host 'wdflbmt7215' is not validated!
Not Logged in. Use 'xs login' to log in.

PS C:\Users\train-05> xs -v
Client version: xs v1.0.83

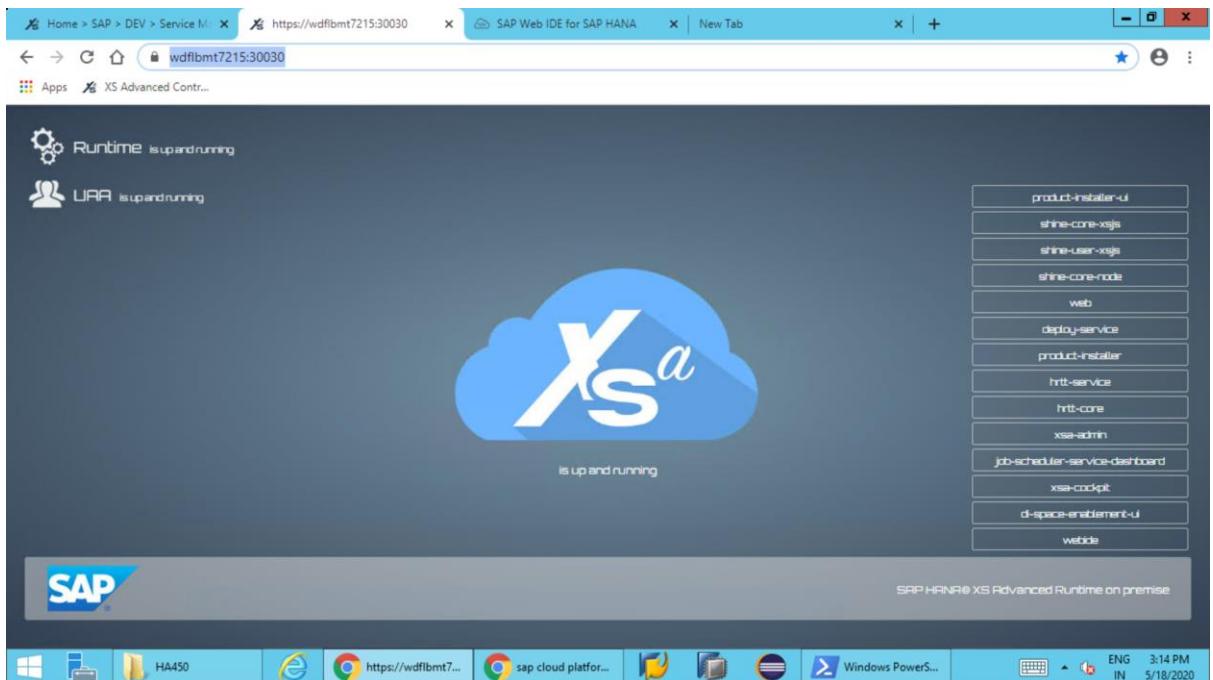
Server version information:
  name          = XS Controller
  support       = http://service.sap.com/message
  build         =
  version       = 1
  overallState  = READY
  user          = <not set>
  description   = SAP HANA XS Advanced Runtime
  controllerEndpoint = https://wdflbmt7215:30030
  authorizationEndpoint = https://wdflbmt7215:30032/uaa-security
  loggingEndpoint = <not set>
  allowDebug    = <not set>
  acceptEncoding = gzip, x-gzip
  limits        = memory: <not set>, apps: <not set>, app uris: <not set>, services: <not set>
  usage         = memory: <not set>, apps: <not set>, app uris: <not set>, services: <not set>
  databaseType  = HANA_MULTI
  databaseInfo   =

Registered service URLs:
  product-installer-ui      = https://wdflbmt7215:53280
  shine-core-xsjs            = https://wdflbmt7215:51033
  shine-user-xsjs            = https://wdflbmt7215:51034
  shine-core-node             = https://wdflbmt7215:51037
  web                         = https://wdflbmt7215:51041
  deploy-service              = https://wdflbmt7215:51004
  product-installer           = https://wdflbmt7215:51005
  hrtt-service                = https://wdflbmt7215:51009
  hrtt-core                   = https://wdflbmt7215:51012
  xsa-admin                   = https://wdflbmt7215:51031
  job-scheduler-service-dashboard = https://wdflbmt7215:51027
  xsa-cockpit                 = https://wdflbmt7215:51036
  di-space-enablement-ui     = https://wdflbmt7215:51020
  webide                      = https://wdflbmt7215:53075

PS C:\Users\train-05>
```

2. Open Chrome browser. Copy the controller end point url as shown below. It allows to access all related tools related to XS Advanced.

<https://wdflbmt7215:30030/>

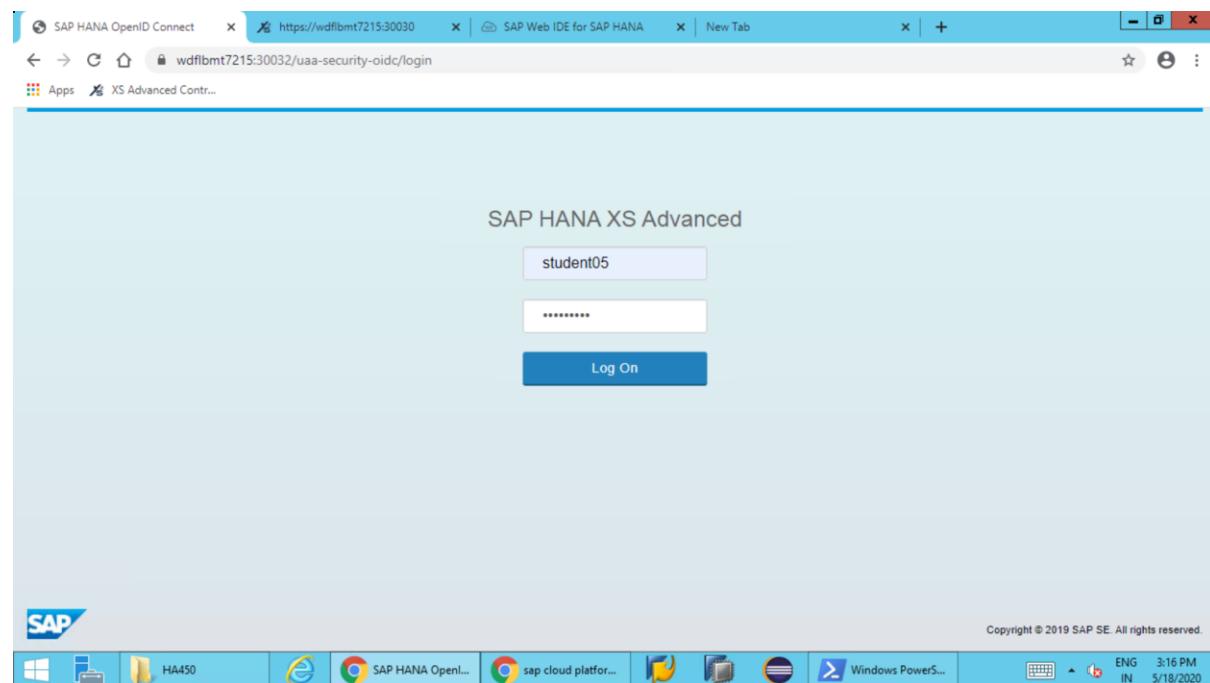


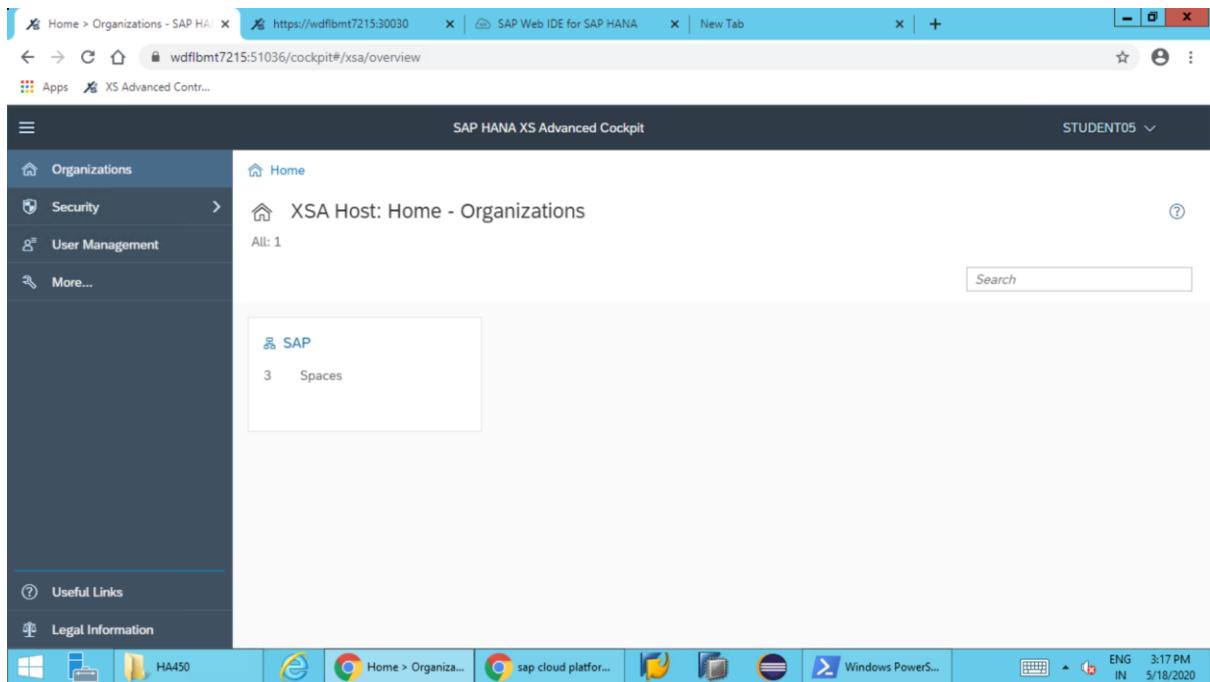
3. Select webide and xsa-cockpit – these are the XS Advanced on premise tools for development and monitoring.

XS Advanced Cockpit <https://wdflbmt7215:30032/uaa-security-oidc/login>

Useid: student05

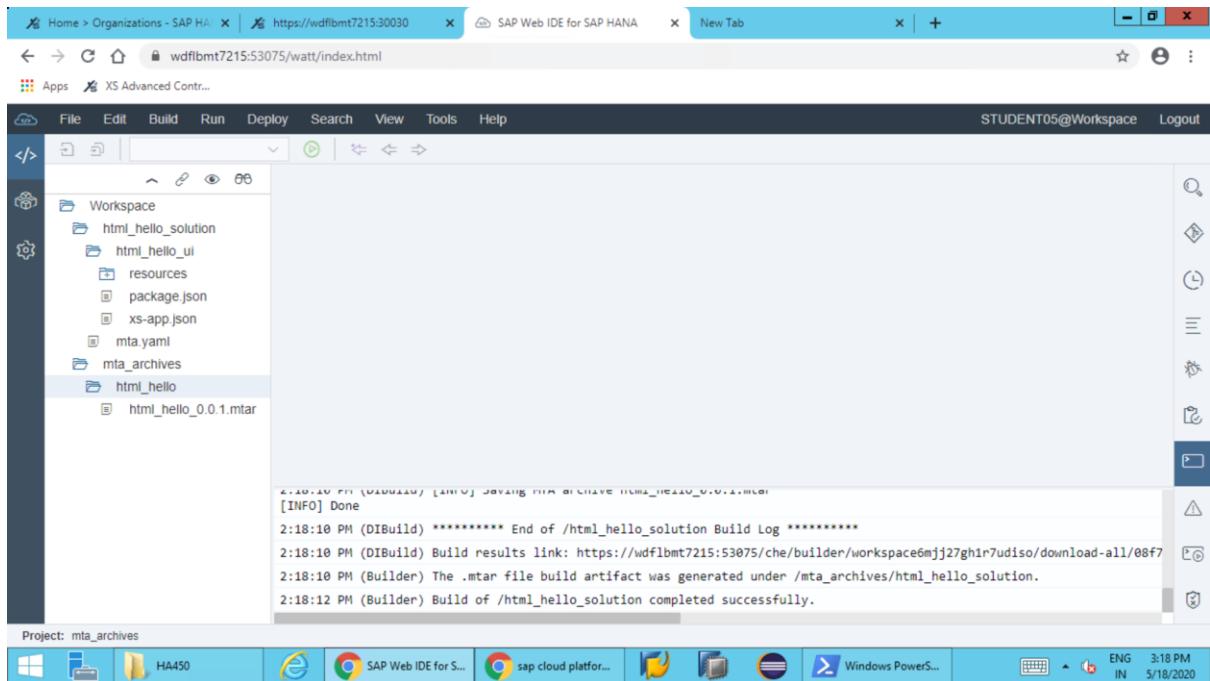
Pwd: Training1





Webide:

<https://wdflbmt7215:53075/watt/index.html>



- Now create a trial account to access cloud foundry trial account. Valid for 30 days. It takes a while to create new account. If already have account and if its expired use new phone number.

<https://cockpit.hanatrial.ondemand.com/cockpit#/home/trial>

Screenshot of SAP Cloud Platform Cockpit showing Subaccounts and Subaccount details.

**Subaccounts Overview:**

- Global Account: 314d5179trial - Subaccounts
- Subdomain: 314d5179trial-ga
- All: 1
- Buttons: New Subaccount, Switch Global Account, Delete Trial Account
- Search Bar: Search Subaccounts

**Subaccount Details (trial):**

- Provider: Amazon Web Services (AWS)
- Region: cf-eu10
- Description: -none-
- Environment: Cloud Foundry

**Subaccount Overview (trial):**

- Buttons: Delete Subaccount
- Subaccount Details:
  - Subdomain: 314d5179trial
  - ID: 43571e45-166f-4411-8d71-6ca8967f017f
- Cloud Foundry:
  - Org Name: 314d5179trial
  - Org ID: 8143d9fa-c495-44b7-a874-8cc7bfa34913
  - Members: 1
  - API Endpoint: https://api.cf.eu10.hana.ondemand.com
- Spaces (1):
 

Name	Applications	Service Instances
dev	0	0

Goto Home screen

<https://cockpit.hanatrial.ondemand.com/cockpit#/home/trial>

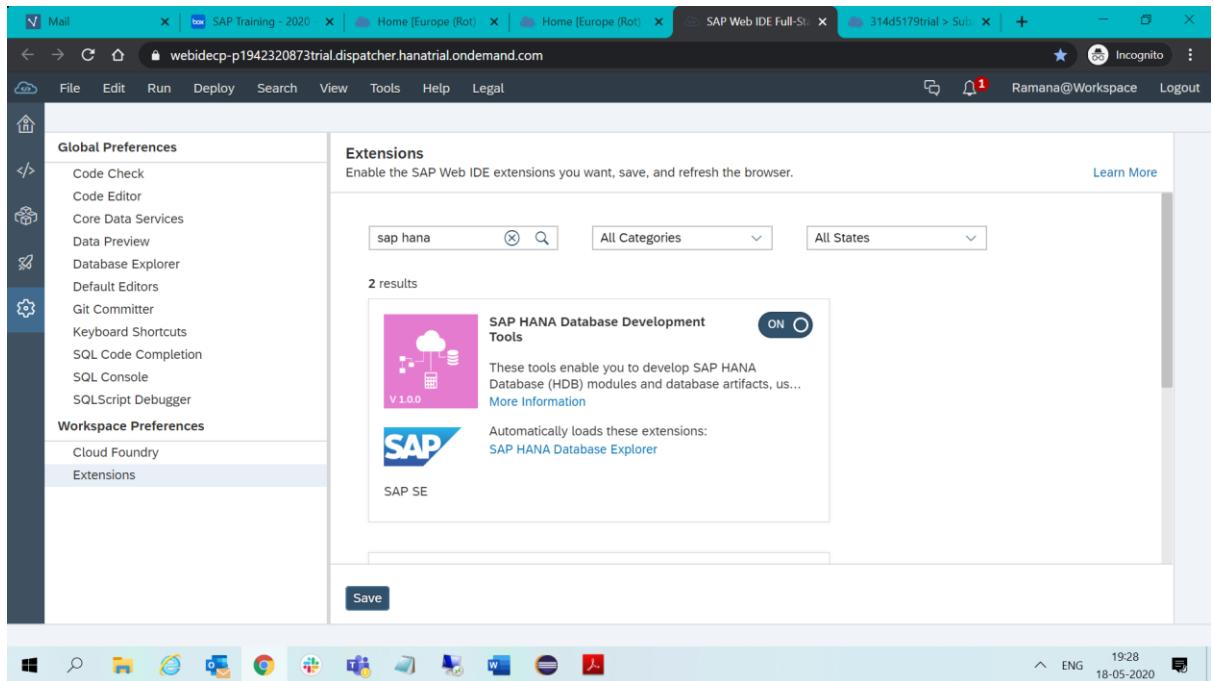
And select webide url and bookmark it.

<https://webidecp-p1942320873trial.dispatcher.hanatrial.ondemand.com/>

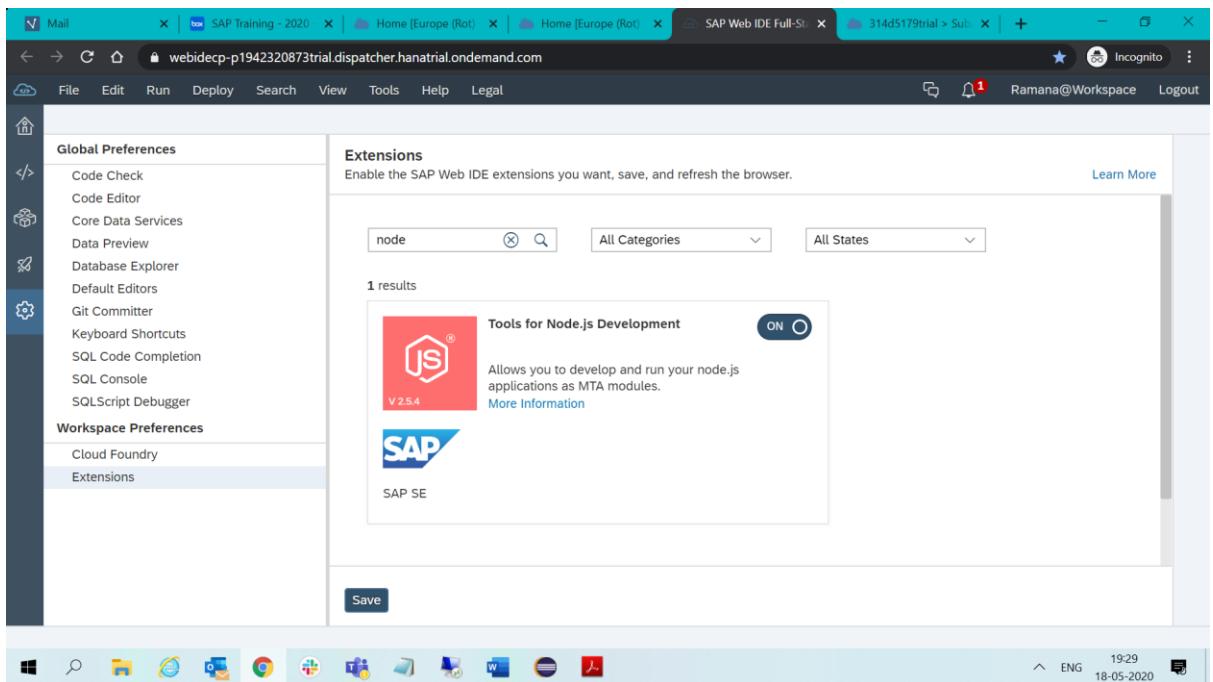
Now goto Settings and Configure below extensions and select turn on

Go to preferences → Extensions.

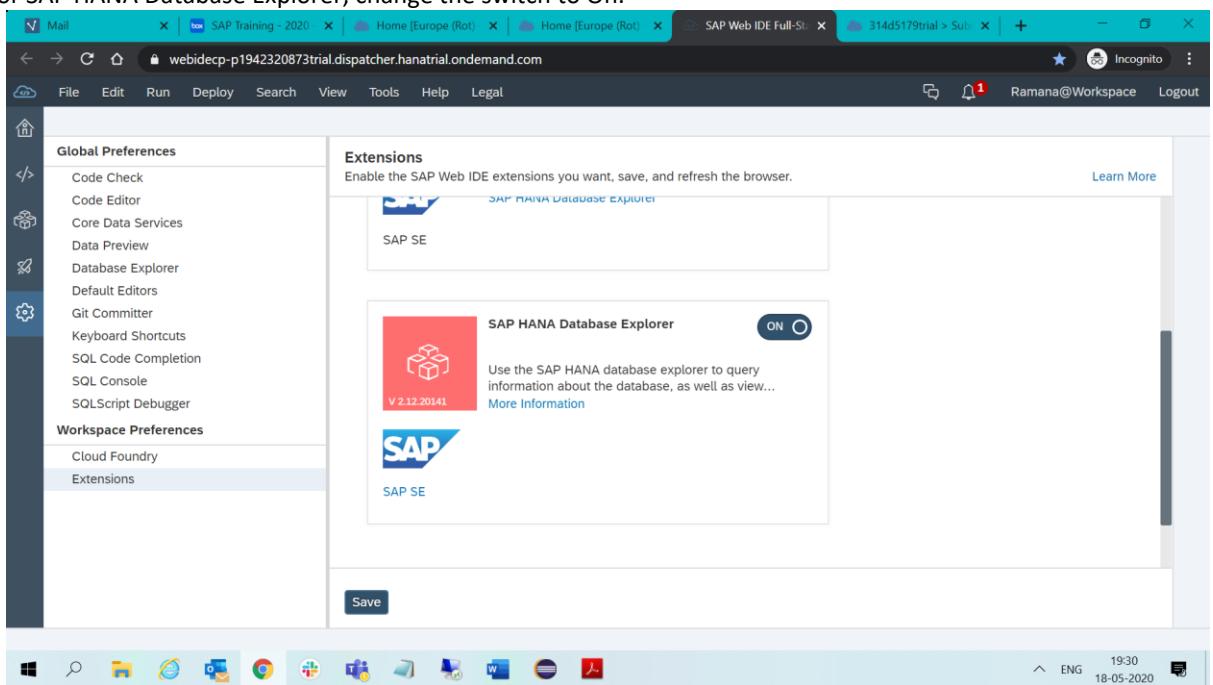
Search for Tools for SAP HANA Database Development, change the switch to On.



Search for Tools for Node.js Development, change the switch to On.



Search for SAP HANA Database Explorer, change the switch to On.



Don't forget to select Save . After opting for addons.

5. Now you can Use Exercise pdf to follow tutorial instructions to do it yourself. If you wish to have complete overview refer next steps in the tutorials



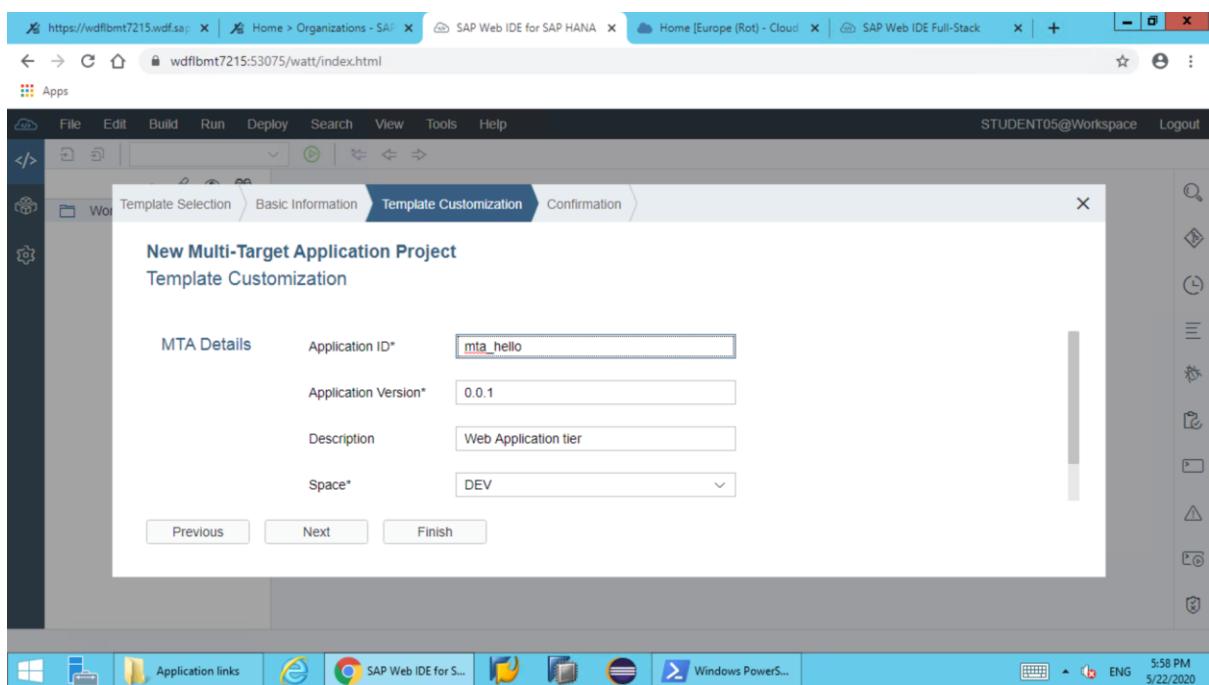
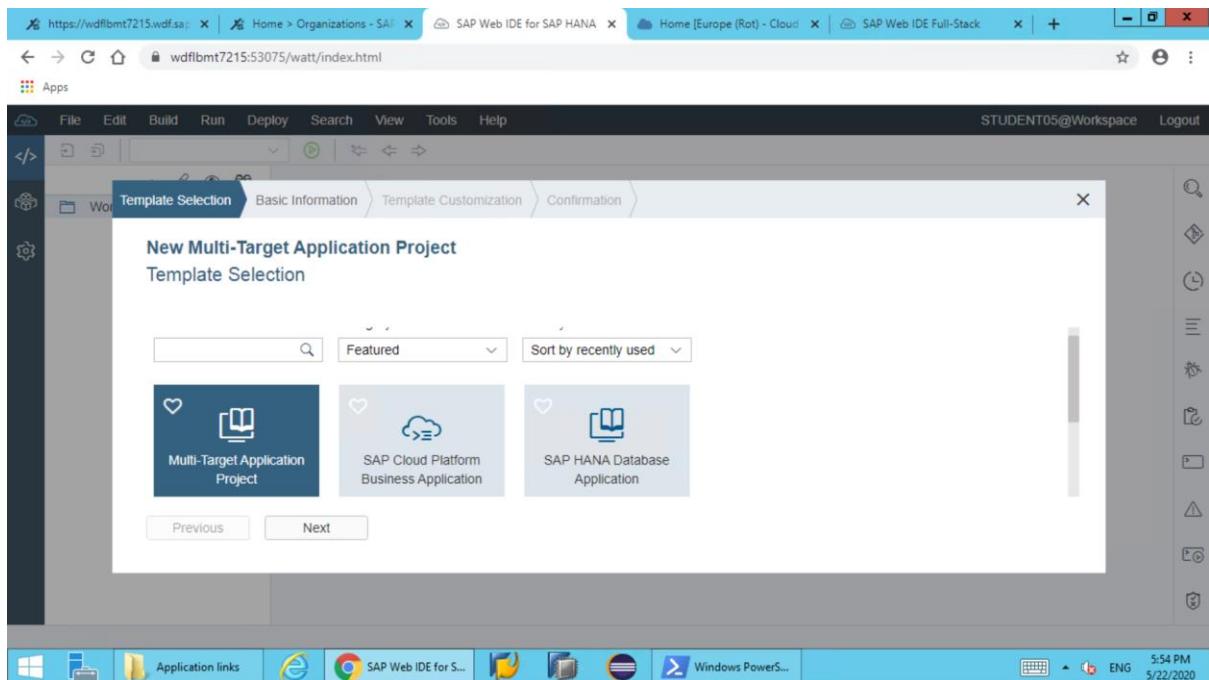
02.HA450  
Exercises.pdf

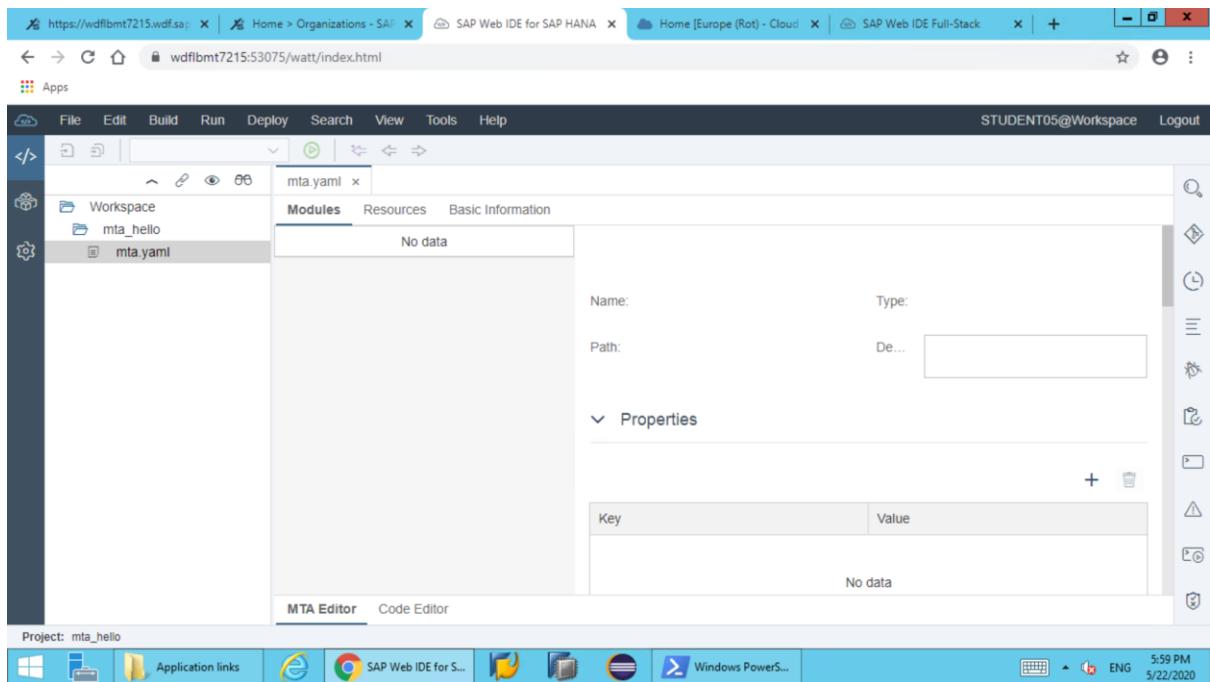
## WHAT IS MTA ?

MTA - MULTI TARGET APPLICATION are comprised of multiple software modules representing the data, business logic and UI tiers . These modules are created with different technologies, yet share the same development lifecycle.

## Getting Started with MTA

### 1. Create new MTA project

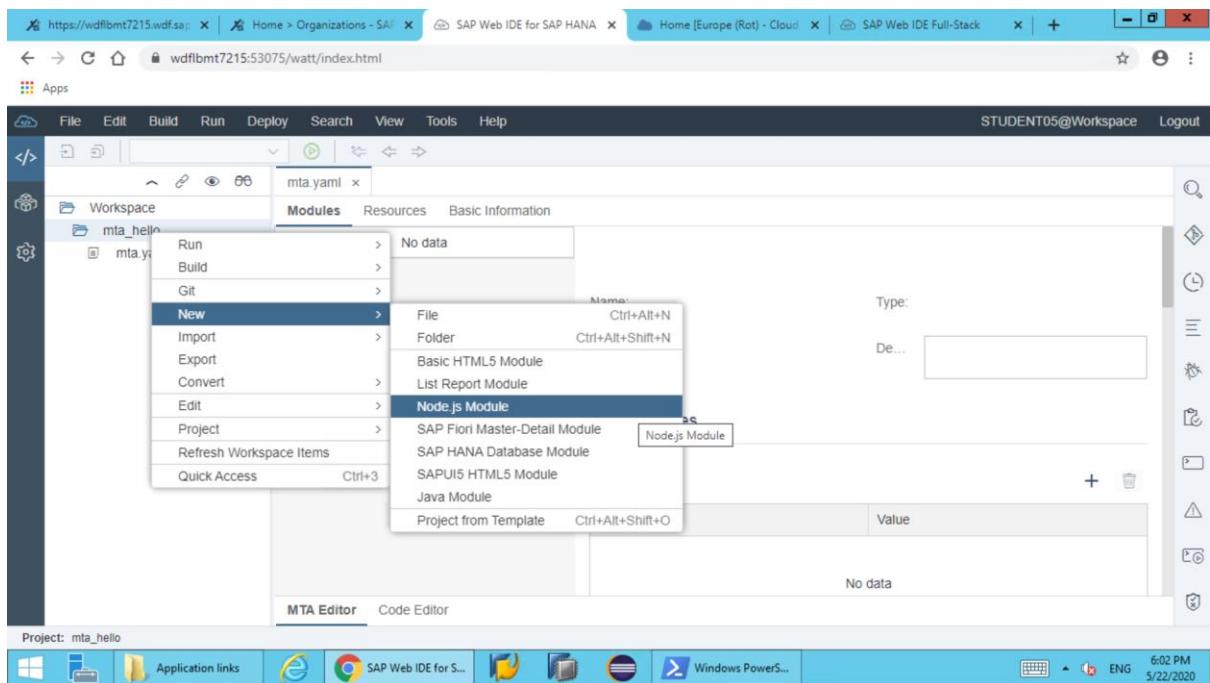




Now we can have multiple modules in MTA based projects.

1. SAP UI5/SAP Fiori/List Report /HTML5/Java module for UI Tier
2. Node.js modules for Application server tier
3. Hana database modules for Database tier

In this section , we are using Node.js module.



The screenshot shows the SAP Web IDE interface. A modal dialog titled "New Node.js Module" is open under the "Template Customization" tab. The "Module settings:" section contains the following fields:

- Version: \* 1.0.0
- Description: Node module
- Main JS file: \* server.js
- Enable XSJS support

Below the dialog are buttons for "Previous", "Next", and "Finish". The SAP Web IDE interface shows a project structure on the left with a file tree for "mta\_hello" containing "node\_hello" and other files like "lib", "test", "package.json", "server.js", and "testrun.js". On the right, a "node\_hello" module is selected in the "Modules" tab of the "mta.yaml" editor. The properties for this module are displayed, including Name: node\_hello, Type: nodejs, and Path: node\_hello. A "Properties" table is also shown.

Delete all the files under node\_hello.

And add new package.json and server.js files.

### Pakage.json

```
{
  "name": "node_hello",
  "version": "1.0.0",
  "dependencies": {
```

```

    "express": "^4.0.0"
  },
  "scripts": {
    "start": "node server.js"
  },
  "engines": {
    "node": "10.x"
  }
}

```

### Server.js

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

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

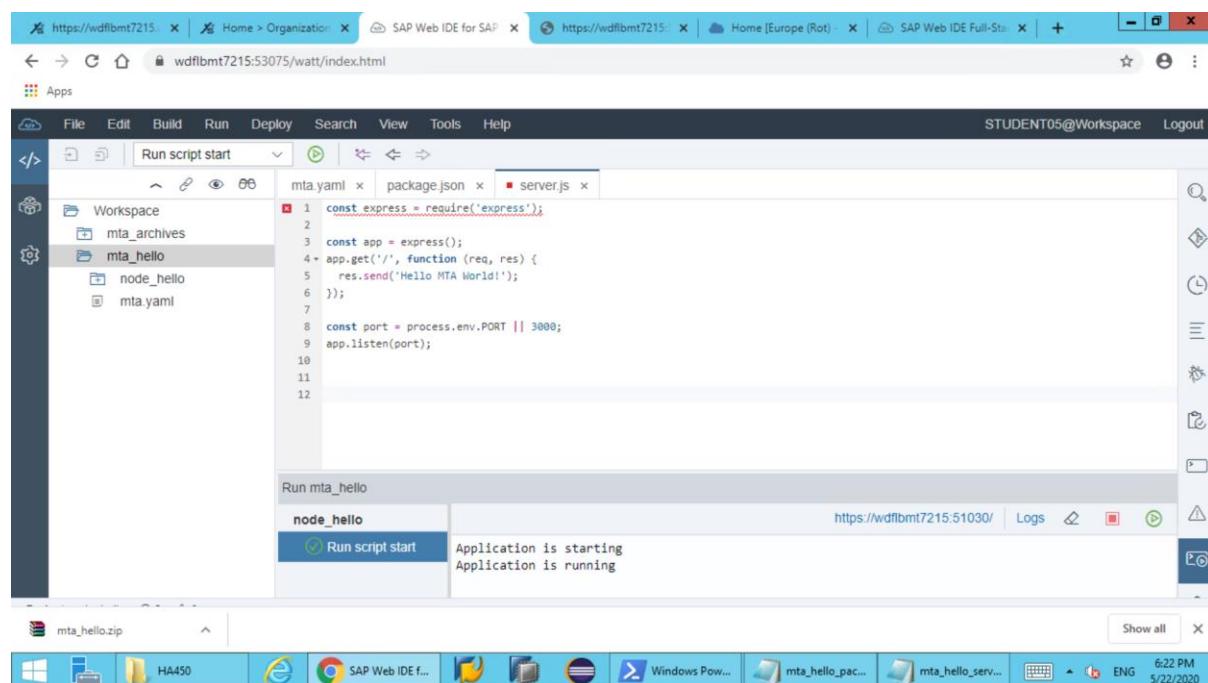
```
app.get('/', function (req, res) {
```

```
  res.send('Hello MTA World!');
```

```
});
```

```
const port = process.env.PORT || 3000;
```

```
app.listen(port);
```



Now build the node\_hello folder and mta\_hello folders.

And select node\_hello and run as node.js application.

The screenshot shows the SAP Web IDE interface. The left sidebar displays the project structure under 'Workspace' with 'mta\_hello' selected. The main editor area shows the 'server.js' file content:

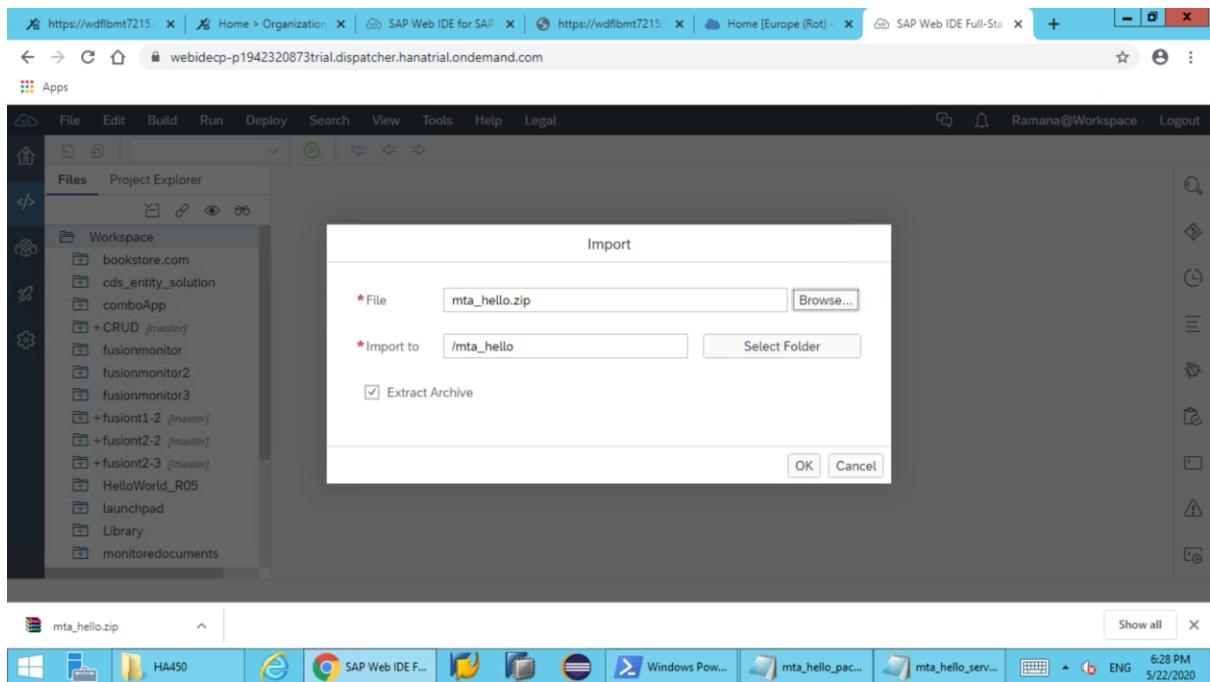
```
1 const express = require('express');
2
3 const app = express();
4 app.get('/', function (req, res) {
5   res.send('Hello MTA World!');
6 });
7
8 const port = process.env.PORT || 3000;
9 app.listen(port);
```

Below the editor, a 'Run mta\_hello' panel shows the 'node\_hello' configuration with a green 'Run script start' button. The status message indicates 'Application is starting' and 'Application is running'. The browser tab at the bottom shows the URL <https://wdflbmt7215:51030/>.

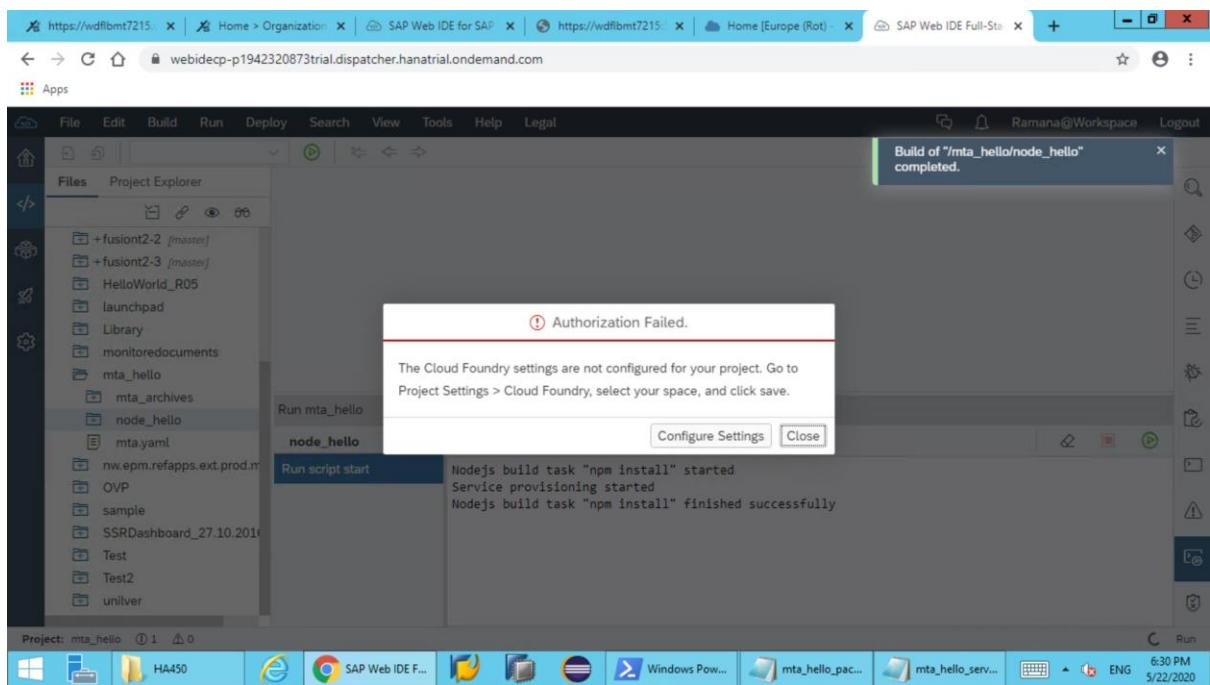
The screenshot shows the SAP Web IDE interface with the browser tab open at [https://wdflbmt7215:51030](https://wdflbmt7215:51030/). The page content displays 'Hello MTA World!'.

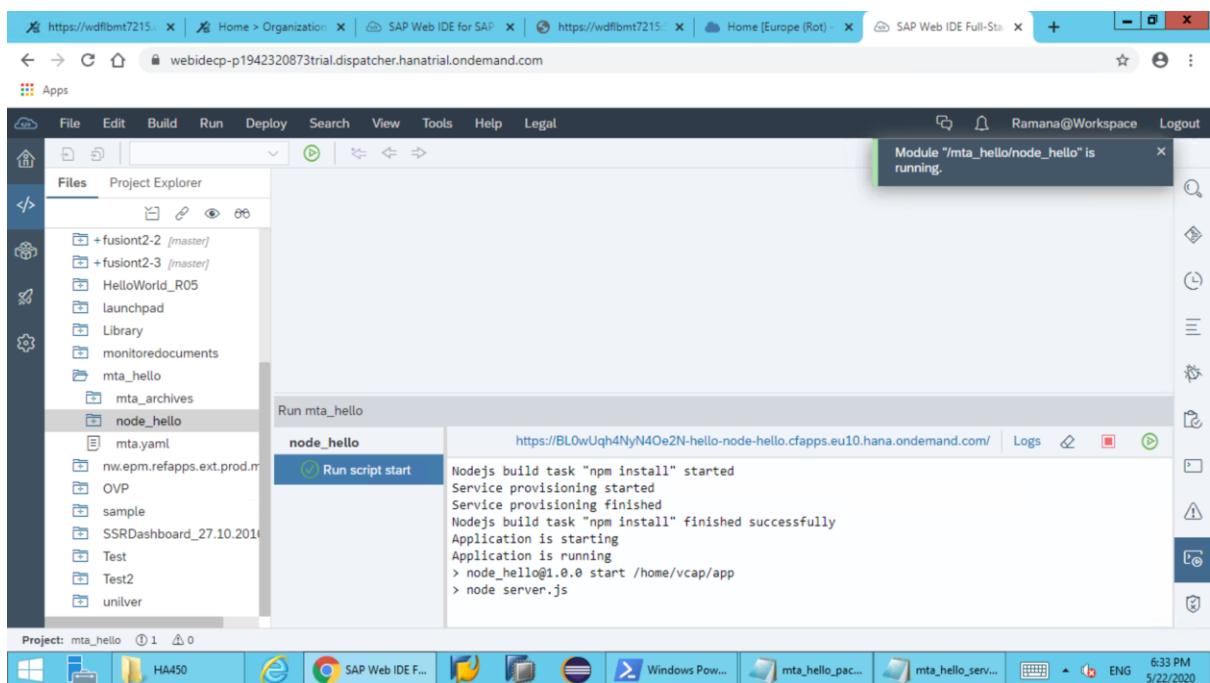
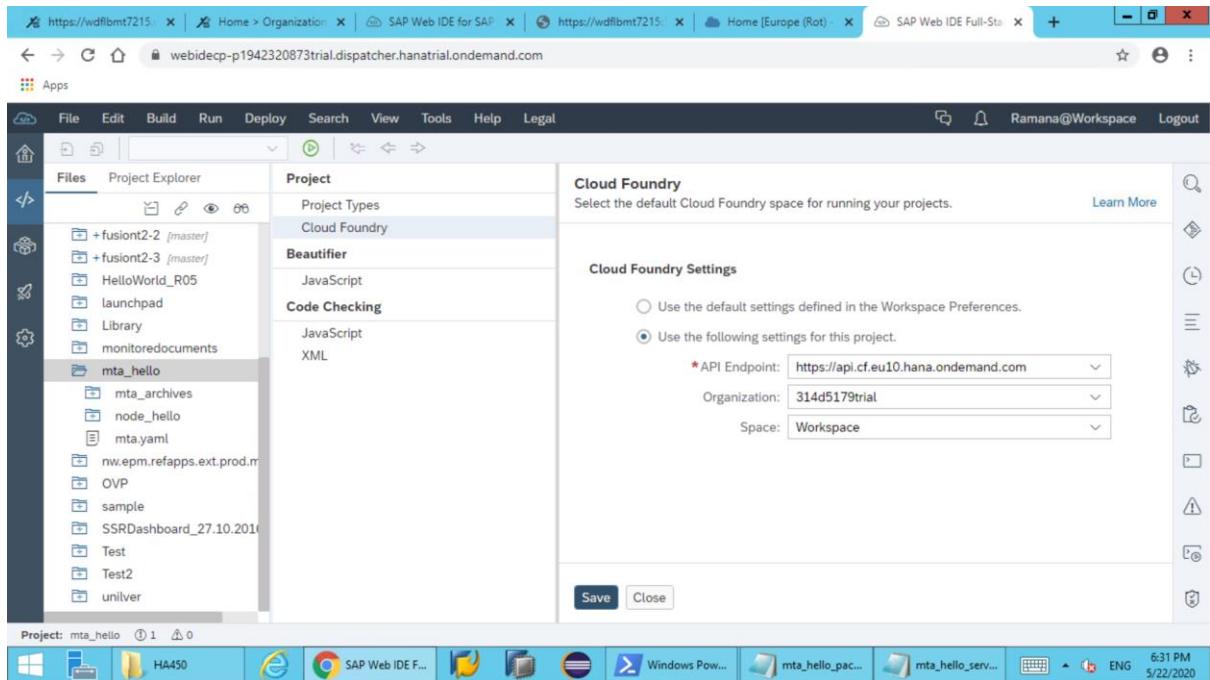


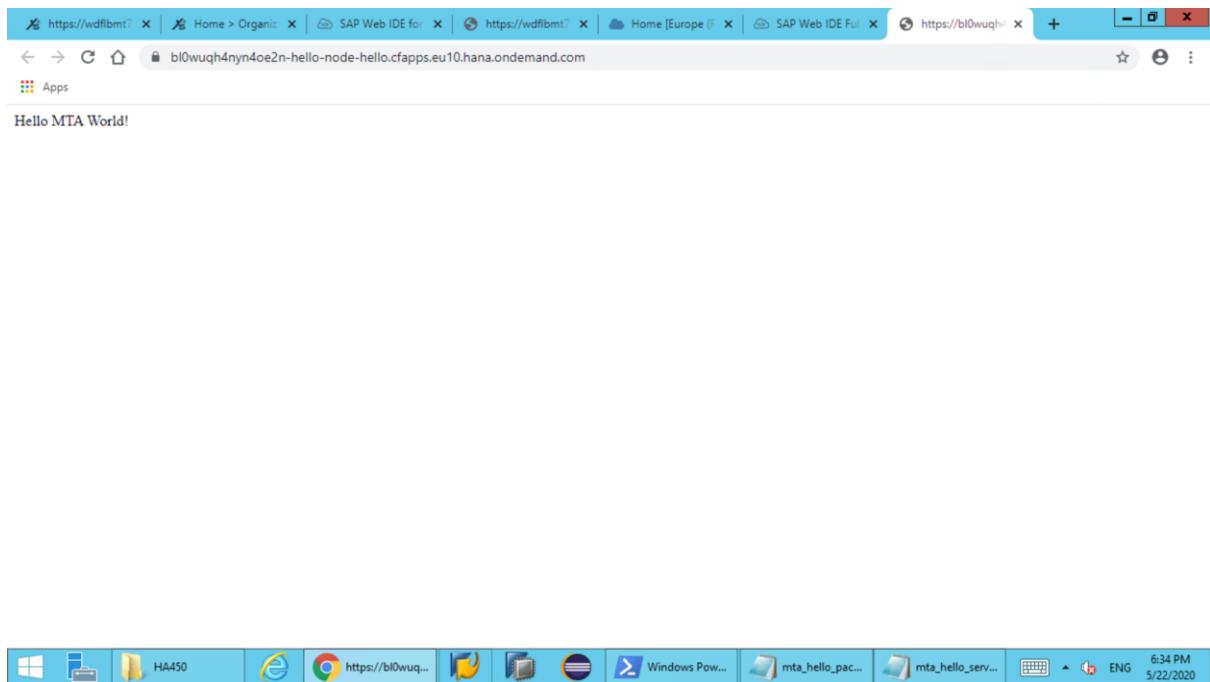
Now importing this project and lets run in SAP Cloud platform trial account. Open webide and import created node\_hello from onpremise system



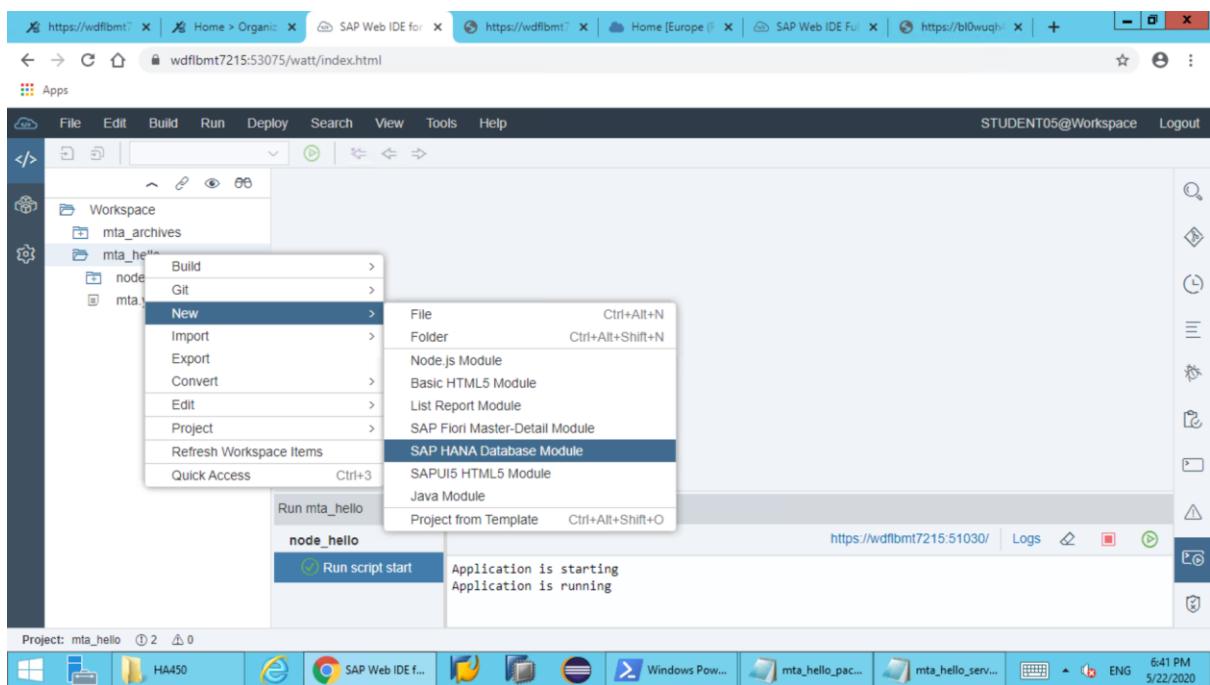
Build the mta\_hello folder and then node\_hello folder.

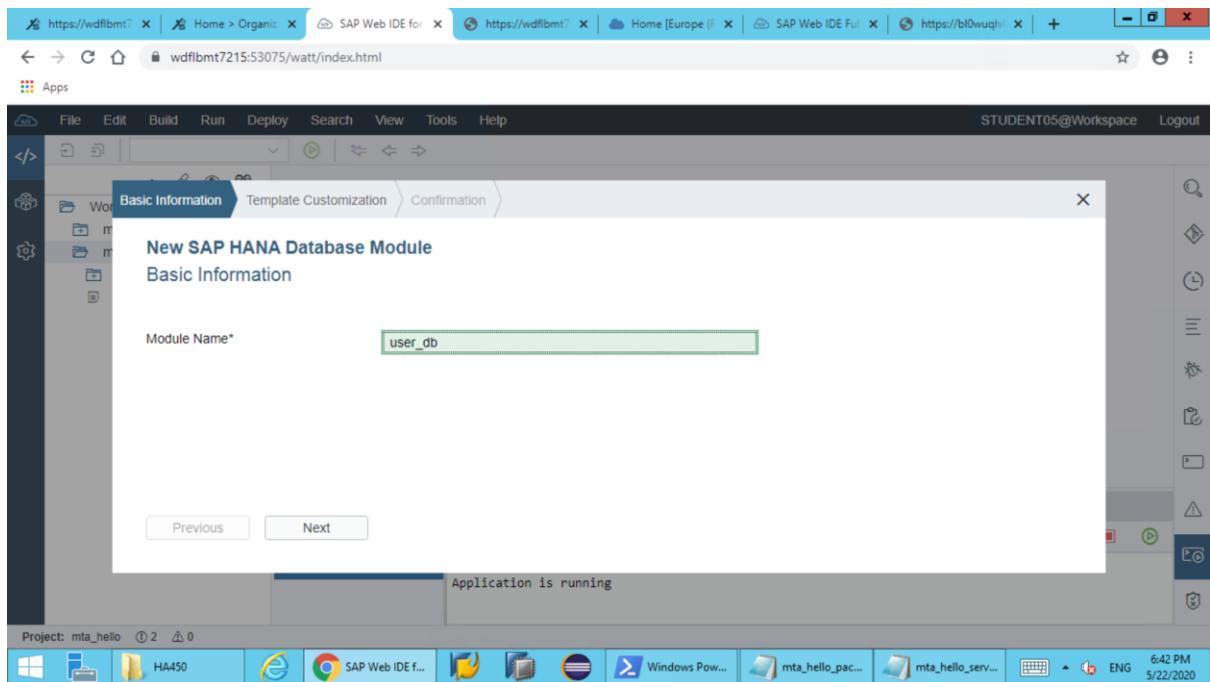




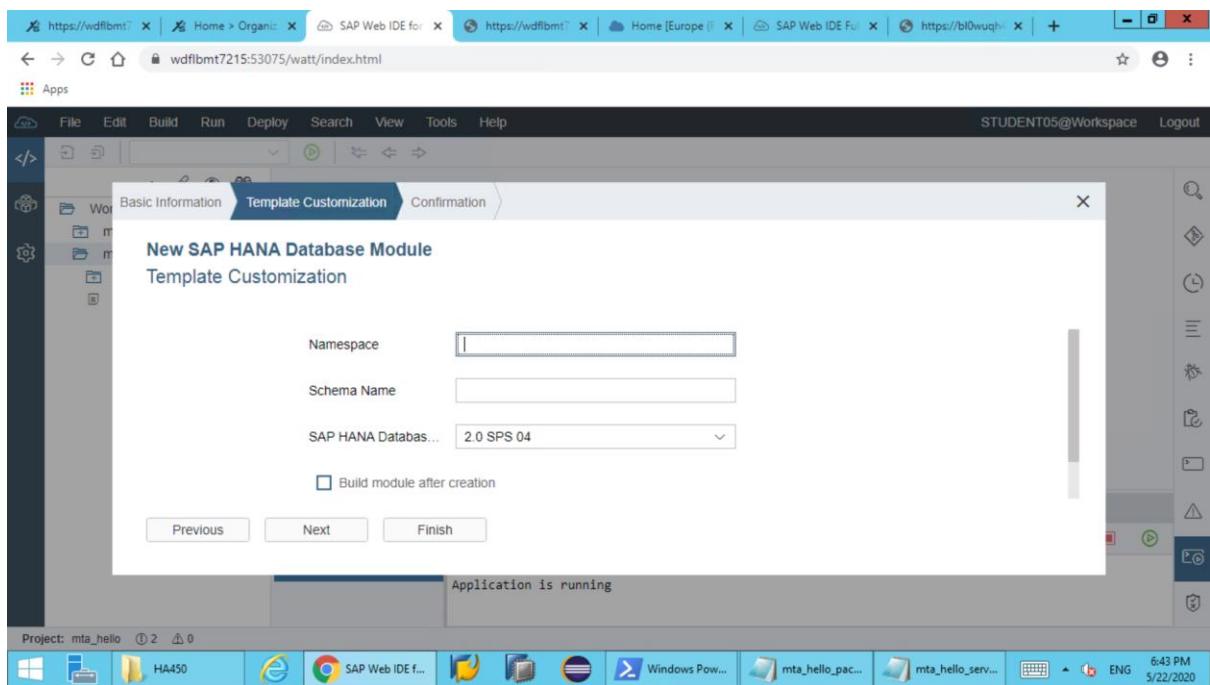


Now lets add Hana Database module using CDS .





Let namespace be blank and proceed



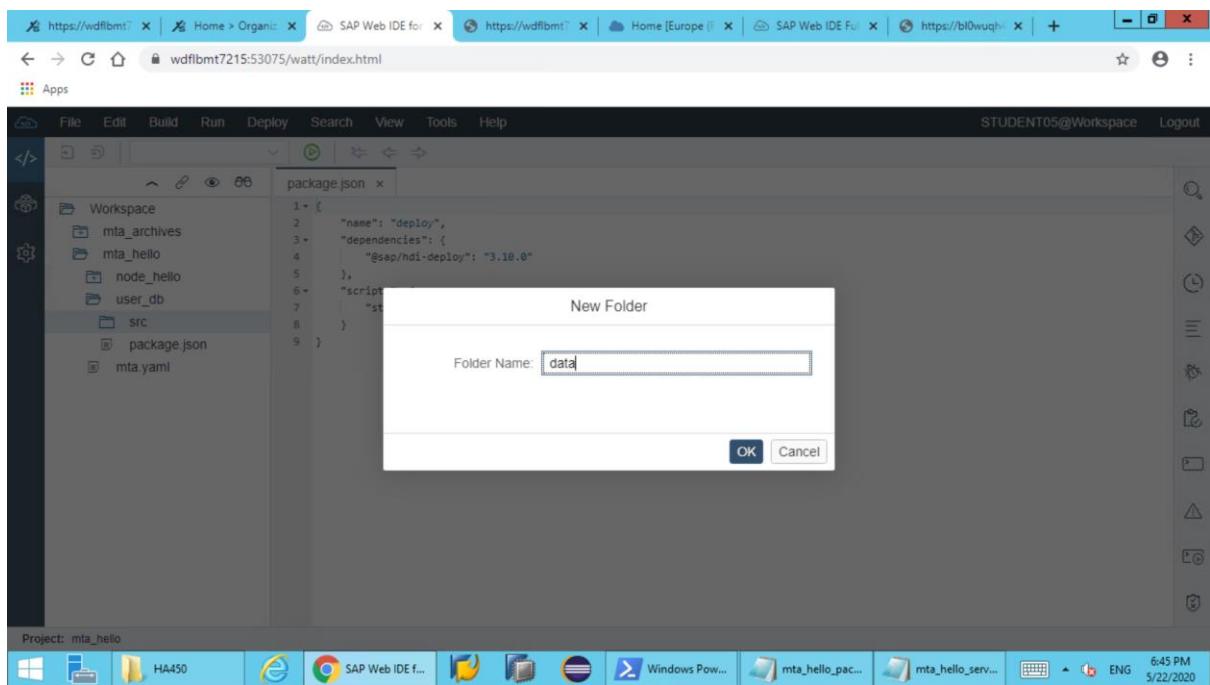
The screenshot shows the SAP Web IDE interface with the URL <https://wdflbmt7215:53075/watt/index.html>. The workspace contains several projects: mta\_archives, mta\_hello, node\_hello, user\_db, and a selected project src containing package.json and mta.yaml. The package.json file is open in the editor, displaying the following content:

```

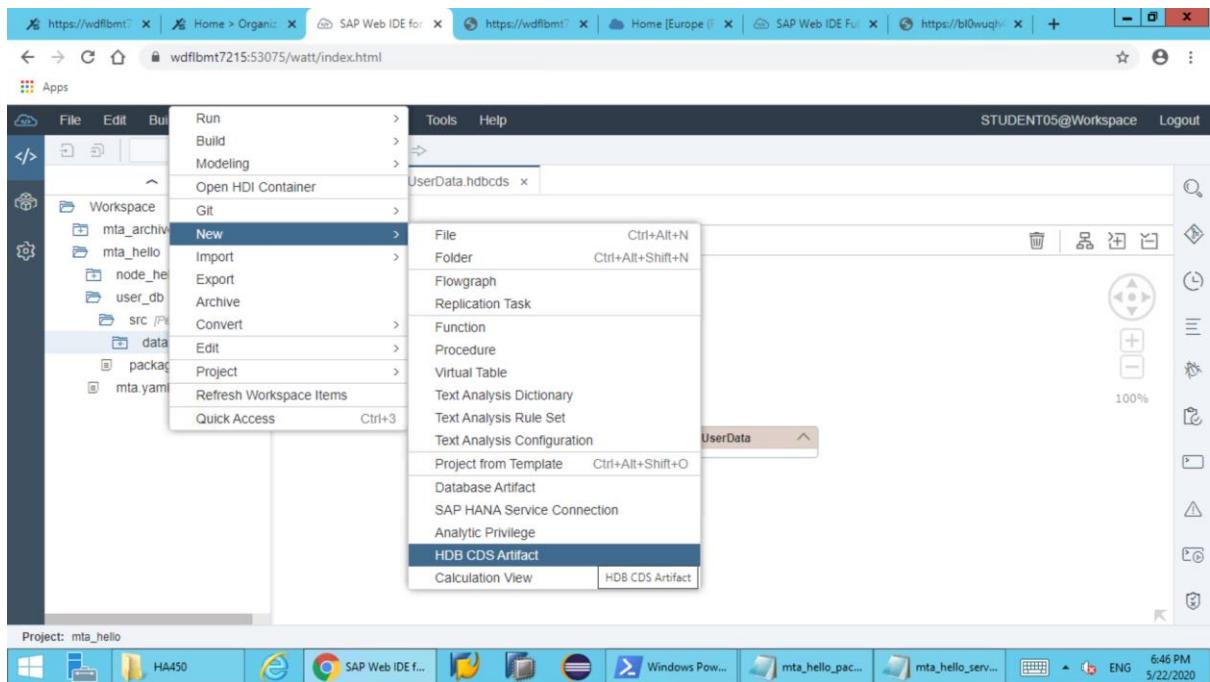
1+ {
2   "name": "deploy",
3   "dependencies": {
4     "@sap/hdi-deploy": "3.10.0"
5   },
6   "scripts": {
7     "start": "node node_modules/@sap/hdi-deploy/deploy.js"
8   }
9 }

```

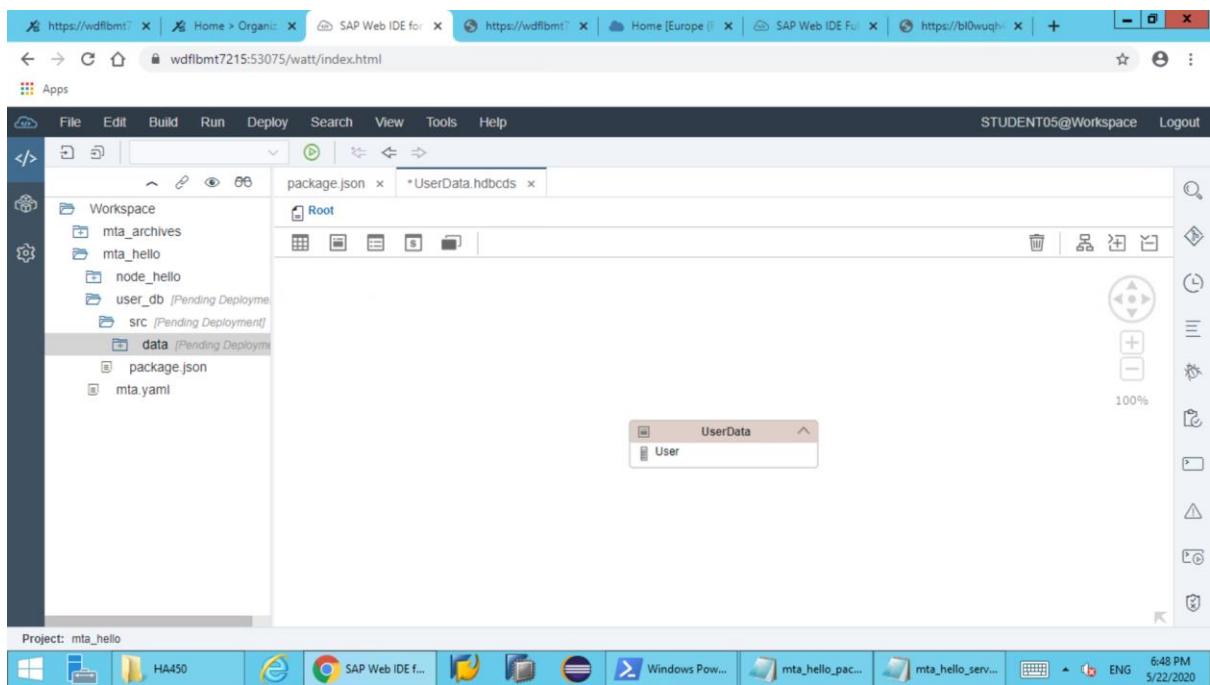
Create a new subfolder data ->under src



Create new file UserData.hdbclds



Now double click on UserData



Now create a entity named User and double click and add properties.

The screenshot shows the SAP Web IDE interface. On the left, the project structure for 'mta\_hello' is visible, including 'mta\_archives', 'mta\_hello', 'node\_hello', and 'user\_db'. The 'user\_db' folder contains 'src' (Pending Deployment) and 'package.json'. The main workspace displays the 'User' table definition in 'UserData.hdbcds'. The table has four columns: 'UserId' (Primitive, Integer), 'FirstName' (Primitive, String), 'LastName' (Primitive, String), and 'Email' (Primitive, String). Below the table, a log window shows build logs:

```

6:18:51 PM (DIBuild) up to date in 0.893s
6:18:51 PM (DIBuild) ***** End of /mta_hello/node_hello Build Log *****
6:18:51 PM (DIBuild) Build results link: https://wdflbmt7215:53075/che/builder/workspace6mj27gh1r7udiso/download-all/8653
6:18:54 PM (Builder) Build of /mta_hello/node_hello completed successfully.
6:52:43 PM (Builder) Build of "/mta_hello/user_db" started.

```

Now build the user\_db module.

In the data folder, create a “loads” subfolder and new csv file users.csv.

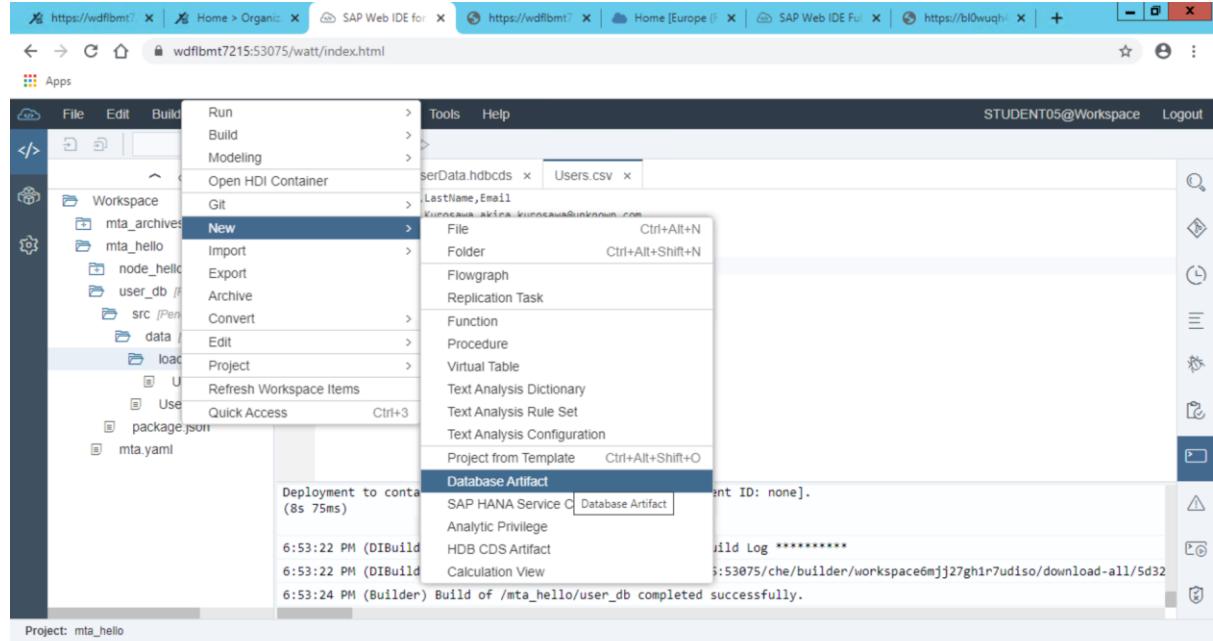
The screenshot shows the SAP Web IDE interface. The project structure for 'mta\_hello' is visible. In the 'data' folder under 'user\_db', a 'New Folder' dialog box is open, prompting for a 'Folder Name' which is set to 'loads'. The main workspace shows the 'User' table definition in 'UserData.hdbcds'. A deployment log at the bottom indicates a successful deployment to container MTA\_HELLO\_HDI\_USER\_DB.

### Users.csv

UserId,FirstName,LastName,Email  
 1000000000,Akira,Kurosawa,akira.kurosawa@unknown.com  
 1000000001,Federico,Fellini,federico.fellini@unknown.com  
 1000000002,Igmar,Bergman,igmar.bergman@unknown.com

1000000003,John,Ford,john.ford@unknown.com

In the loads folder, create a table data file named User.hdbtabledata to load data of the users.csv file into the UserData.User table.



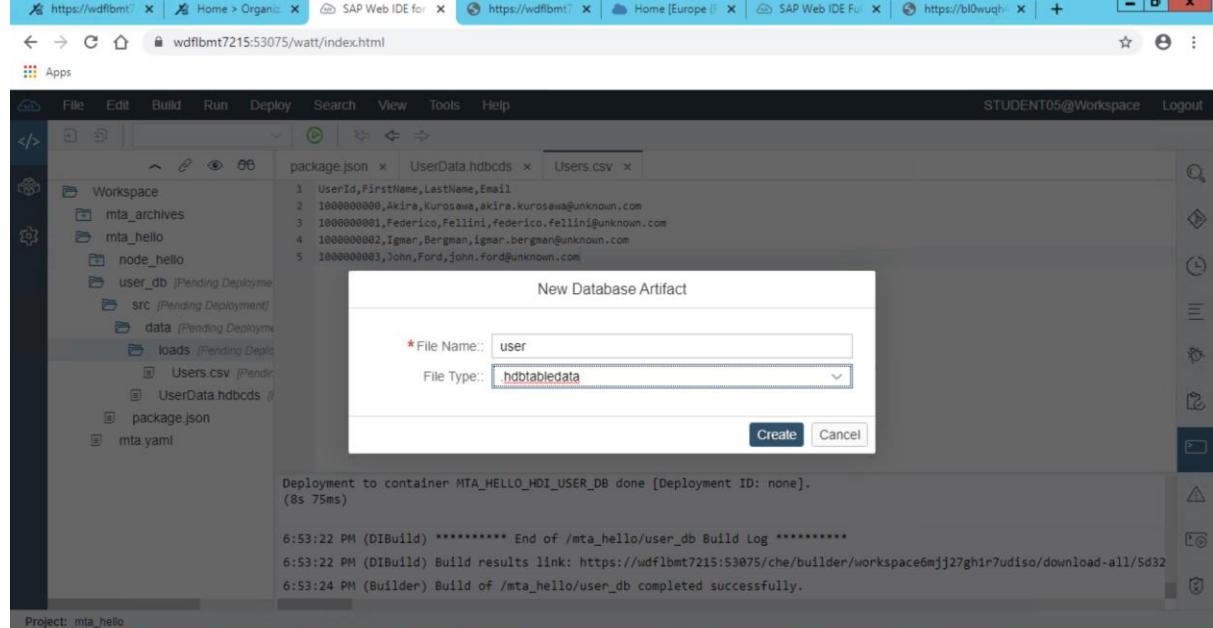
The screenshot shows the SAP Web IDE interface. In the top navigation bar, there are several tabs including "https://wdflbmt7215:53075/watt/index.html". The main workspace shows a project structure for "mta\_hello" with a "loads" folder containing "Users.csv". A context menu is open over the "User.hdbcds" file, with "New" selected. Under "New", "Database Artifact" is chosen. A modal dialog titled "New Database Artifact" is displayed, showing fields for "File Name:" set to "user" and "File Type:" set to "hdbtabledata". Below the dialog, deployment logs show a successful build of the database artifact.

Deployment to container MTA\_HELLO\_HDI\_USER\_DB done [Deployment ID: none].  
(8s 75ms)

6:53:22 PM (DIBuild) \*\*\*\*\* End of /mta\_hello/user\_db Build Log \*\*\*\*\*

6:53:22 PM (DIBuild) Build results link: https://wdflbmt7215:53075/che/builder/workspace6mj27gh1r7udiso/download-all/5d32

6:53:24 PM (Builder) Build of /mta\_hello/user\_db completed successfully.



The screenshot shows the SAP Web IDE interface again. The "User.hdbtabledata" file is open, displaying the following JSON content:

```
User.hdbtabledata
{
  "format_version": 1,
  "imports": [
    {
      "target_table": "UserData.User",
      "source_data": {
        "User": [
          {"UserId": 1, "FirstName": "Akira", "LastName": "Kurosewa", "Email": "akira.kurosewa@unknown.com"}, ...
        ]
      }
    }
  ]
}
```

```

        "data_type": "CSV",
        "file_name": "users.csv",
        "has_header": true
    }
}
]
}

```

Now build the user\_db folder.

Once build is successful. Right click folder and select open HDI Container.

Display the created database table and its content.

Right-click on the SAP HANA database module (user\_db) folder and choose Open HDI Container.

Select the Tables subfolder.

Right-click the UserData.User table and choose Open Data.

You should be able to see the data from the users.csv file uploaded to the table.

The screenshot shows the SAP Web IDE interface with the following details:

- Header:** https://wdflibmt7215:53075/watt/index.html
- User:** STUDENT05@Workspace
- Toolbar:** File, Edit, Run, Deploy, Search, View, Tools, Help
- Left Sidebar:** Public Synonyms, Remote Subscriptions, Sequences, Synonyms, Table Types, Tables (selected), Tasks, Triggers, Views.
- Current View:** package.json, UserData.User (selected), UserData.User, Users.csv, user.hdbtabledata
- Table Definition:**
  - Table Name: UserData User
  - Schema: MTA\_HELLO\_HDI\_USER\_DB
  - Type: COLUMN
  - Open Data button
- Columns:**

	Name	SQL Data Type	Column Store Dat...	Key	Not Null	Default	Comment
1	UserId	INTEGER	INT	1	X		
2	FirstName	NVARCHAR(40)	STRING				
3	LastName	NVARCHAR(40)	STRING				
4	Email	NVARCHAR(10)	STRING				
- Bottom:** Taskbar with various icons and system status (7:09 PM, ENG, 5/22/2020).

The screenshot shows the SAP Web IDE interface with multiple tabs open. The main area displays a table titled 'Raw Data Analysis' with four rows of user data. The columns are labeled 'UserId', 'FirstName', 'LastName', and 'Email'. The data is as follows:

	UserId	FirstName	LastName	Email
1	1000000000	Akira	Kurosawa	akira.kurosawa@unknown.com
2	1000000001	Federico	Fellini	federico.fellini@unknown.com
3	1000000002	Igmar	Bergman	igmar.bergman@unknown.com
4	1000000003	John	Ford	john.ford@unknown.com

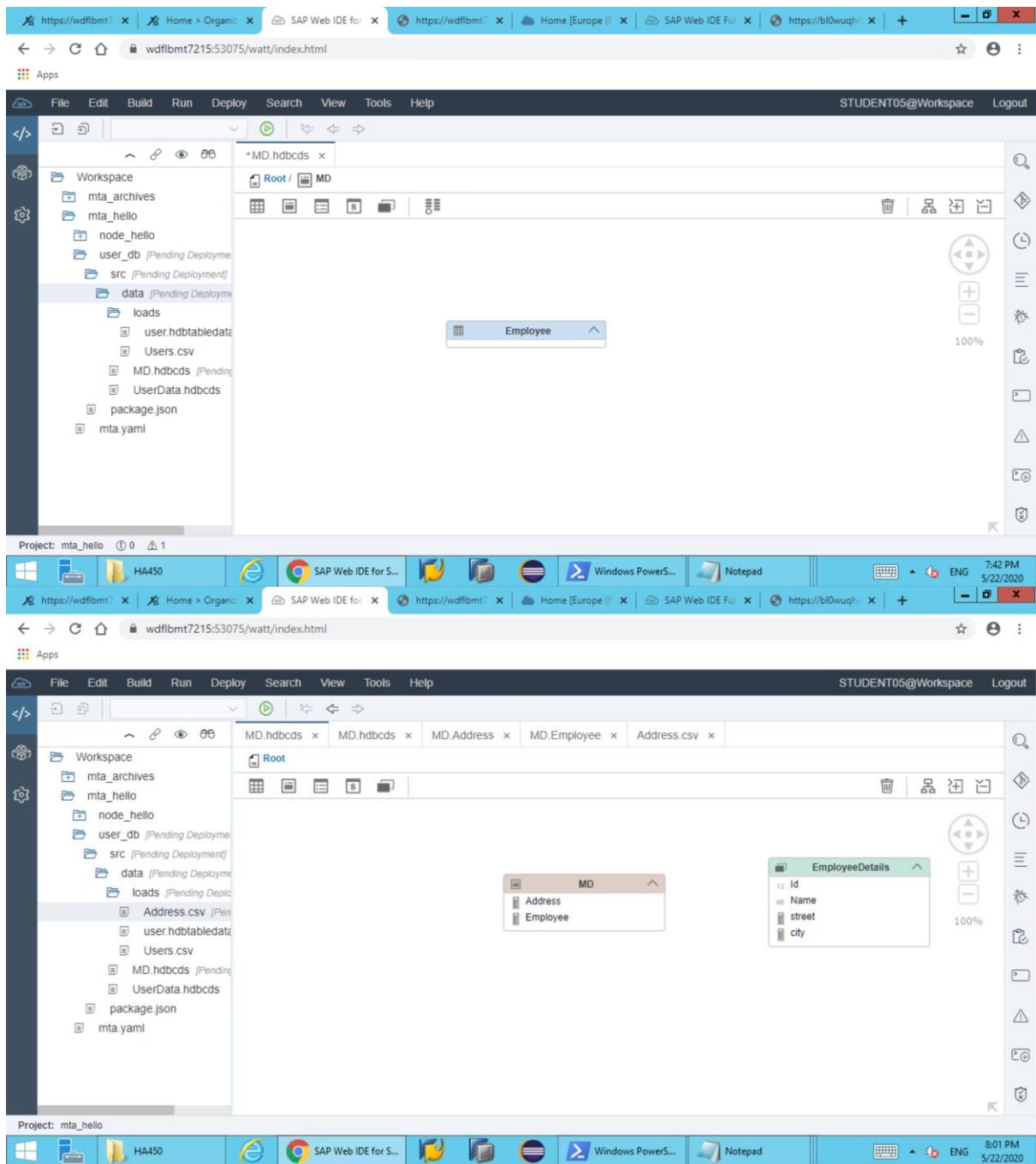
The left sidebar shows navigation links for 'Tables' (selected), 'Public Synonyms', 'Remote Subscriptions', 'Sequences', 'Synonyms', 'Table Types', 'Tasks', and 'Triggers'. A search bar at the bottom left is set to 'UserData User'.

Note: all files name are case sensitive. If any incorrect info, build fails. Check the file names used in the hdbtabledata file .

Now we could successfully, using cds created a database at runtime using hdbcards and used csv mock data and loaded data into the database table using hdbtabledata file.

Lets create a Associations and a corresponding views using CDS

The screenshot shows the SAP Web IDE interface with multiple browser tabs open. The main view displays a workspace named 'mta\_hello'. A context menu is open over the 'data' folder, specifically under the 'New' submenu. The 'HDB CDS Artifact' option is highlighted. Other options visible in the submenu include File, Folder, Flowgraph, Replication Task, Function, Procedure, Virtual Table, Text Analysis Dictionary, Text Analysis Rule Set, Text Analysis Configuration, Project from Template, Database Artifact, SAP HANA Service Connection, Analytic Privilege, Calculation View, and HDB CDS Artifact.



Screenshot of SAP Web IDE for SAP Cloud Platform showing the creation of an Employee entity.

The interface shows the SAP Web IDE for SAP Cloud Platform with multiple tabs open. The main workspace displays the structure of the 'Employee' entity:

- Elements (2)**: Contains two primitive type fields: **Name** (String, 10 characters) and **Id** (Integer).
- Associations (1)**: A pending association named **Address** is listed.

The second screenshot shows the creation of the **Address** entity:

- Elements (3)**: Contains three primitive type fields: **Id** (Integer), **street** (String, 80 characters), and **city** (String, 80 characters).
- Associations (0)**: No associations are present.

A modal dialog titled **Create Association** is visible at the bottom of the screen.

The screenshot shows the SAP Web IDE interface. In the top navigation bar, there are several tabs including "https://wdflibmt7215:53075/watt/index.html". The main workspace shows a project structure on the left with a tree view of files like "MD.hdbcds", "MD.hdbcds", "MD.Address", "MD.Employee", and "Address.csv". On the right, under the "Employee" entity, the "Associations (1)" tab is selected. A table lists the association details:

Name	*Target Entity	Source Cardinality	Target Cardinality	Assoc.
Address	MD.Address		[0..1]	Man

Below this, there is a diagram view showing the "Employee" and "Address" entities. The "Employee" entity has attributes "Id", "Name", and "Address". The "Address" entity has attributes "Id", "street", and "city". A line connects the "Address" attribute in the "Employee" entity to the "Address" entity itself, indicating a self-referencing association.

Now Build the user\_db folder. For the created structure.

Lets add csv files to load into employee and Address.

#### Address.csv

id,street,city

1,Blade Avenue,Los Angeles

3,Termin Street,New Orleans

2,Alien Square,Huston

**Employee.csv**

id,name,address.id

1,Rick Deckard,1

2,Sarah Connor,3

3,Ellen Ripley,2

**MD.hdbtledata**

{

  "format\_version": 1,

  "imports": [

    {

      "target\_table": "MD.Employee",

      "source\_data": {

        "data\_type": "CSV",

        "file\_name": "employees.csv",

        "has\_header": true

      }

    },

    {

      "target\_table": "MD.Address",

      "source\_data": {

        "data\_type": "CSV",

        "file\_name": "addresses.csv",

        "has\_header": true

      }

    }

}

Note: In the SAP Cloud Platform you won't find the graphical CDS editor for .hdbcards files. The code editor must be used instead.

The screenshot shows the SAP Web IDE interface with two tables displayed:

**MD.Address**

	id	street	city
1	1	Blade Avenue	Los Angeles
2	3	Termin Street	New Orleans
3	2	Alien Square	Huston

**MD.Employee**

	id	name	address.id
1	1	Rick Deckard	1
2	2	Sarah Connor	3
3	3	Ellen Ripley	2

The screenshot shows the SAP Web IDE interface. The top navigation bar includes tabs for 'SAP Web IDE for SAP HANA' and 'SAP Web IDE Full-Stack'. The main workspace displays a 'Raw Data' view for 'MD.EmployeeDetails' with three rows of data:

	id	name	street	city
1	1	Rick Deckard	Blade Avenue	Los Angeles
2	2	Sarah Connor	Termin Street	New Orleans
3	3	Ellen Ripley	Alien Square	Huston

The left sidebar lists database objects: Remote Subscriptions, Sequences, Synonyms, Table Types, Tables, Tasks, Triggers, and Views. A search bar for 'Views' is also present.

Create a simple procedure that will return a mailing list that can be used to send out news or notifications to users.

The screenshot shows the SAP Web IDE interface with a code editor open for an 'HDBprocedure' named 'get\_mailing\_list'. The code is as follows:

```
1 PROCEDURE "get_mailing_list"()
2 LANGUAGE SQLSCRIPT
3 SQL SECURITY INVOKER
4 --DEFAULT SCHEMA <default_schema_name>
5 READS SQL DATA AS
6 BEGIN
7   select "E_MAIL" from "User.Details";
8 END
```

The left sidebar shows a project structure under 'sql\_procedure\_solution' with files like 'calcview\_create\_cube\_solution', 'cds\_context\_association\_view\_solution', 'mta\_archives', 'sql\_procedure\_solution', 'core\_db', 'src', 'data', and 'procedures'. The 'get\_mailing\_list.hdbprocedure' file is selected.

The screenshot shows the SAP Web IDE interface with multiple tabs open. The main editor tab displays the content of the `User.hdbtabledata` file. The code defines a table structure with columns `PERS_NO`, `FIRSTNAME`, `LASTNAME`, and `E_MAIL`. The `format_version` is set to 1, and the `imports` section includes a CSV import configuration for `user_details.csv`.

```
1 "format_version": 1,
2 "imports": [
3     {
4         "target_table": "User.Details",
5         "source_data": {
6             "data_type": "CSV",
7             "file_name": "user_details.csv",
8             "has_header": false,
9             "dialect": "HANA",
10            "type_config": {
11                "delimiter": ","
12            }
13        },
14        "import_settings": {
15            "import_columns": [
16                "PERS_NO",
17                "FIRSTNAME",
18                "LASTNAME",
19                "E_MAIL"
20            ],
21            "column_mappings": {
22                "PERS_NO": 1,
23                "FIRSTNAME": 2,
24                "LASTNAME": 3,
25                "E_MAIL": 4
26            }
27        }
28    }
29 ],
30 "context": {
31     "User": {
32         "type UserKey : String(10);
33         "type FirstName : String(40);
34         "type LastName : String(40);
35         "type EMail : String(255);
36         "type tt_errors {
37             HTTP_STATUS_CODE : Integer;
38             ERROR_MESSAGE : String(100);
39             DETAIL : String(100);
40         };
41         entity Details {
42             key PERS_NO : UserKey;
43             FIRSTNAME : FirstName;
44             LASTNAME : LastName;
45             E_MAIL : EMail;
46         }
47         technical configuration {
48             column store;
49         };
50     }
51 }
```

The screenshot shows the SAP Web IDE interface with multiple tabs open. The main editor tab displays the content of the `User.hdbcds` file. The code defines a context object `User` containing a `tt_errors` type and an `entity Details` type. The `Details` entity has attributes `PERS_NO`, `FIRSTNAME`, `LASTNAME`, and `E_MAIL`.

```
1 "format_version": 1,
2 "imports": [
3     {
4         "target_table": "User.Details",
5         "source_data": {
6             "data_type": "CSV",
7             "file_name": "user_details.csv",
8             "has_header": false,
9             "dialect": "HANA",
10            "type_config": {
11                "delimiter": ","
12            }
13        },
14        "import_settings": {
15            "import_columns": [
16                "PERS_NO",
17                "FIRSTNAME",
18                "LASTNAME",
19                "E_MAIL"
20            ],
21            "column_mappings": {
22                "PERS_NO": 1,
23                "FIRSTNAME": 2,
24                "LASTNAME": 3,
25                "E_MAIL": 4
26            }
27        }
28    }
29 ],
30 "context": {
31     "User": {
32         "type UserKey : String(10);
33         "type FirstName : String(40);
34         "type LastName : String(40);
35         "type EMail : String(255);
36         "type tt_errors {
37             HTTP_STATUS_CODE : Integer;
38             ERROR_MESSAGE : String(100);
39             DETAIL : String(100);
40         };
41         entity Details {
42             key PERS_NO : UserKey;
43             FIRSTNAME : FirstName;
44             LASTNAME : LastName;
45             E_MAIL : EMail;
46         }
47         technical configuration {
48             column store;
49         };
50     }
51 }
```

```

1 PROCEDURE "get_mailing_list"( )
2   LANGUAGE SQLSCRIPT
3   SQL SECURITY INVOKER
4   --DEFAULT SCHEMA <default_schema_name>
5   READS SQL DATA AS
6   BEGIN
7     select "E_MAIL" from "User.Details";
8   END

```

Name	Type	SQL Data Type	Schema Type	Type Name

The screenshot shows the SAP Web IDE interface. In the top navigation bar, there are tabs for 'SAP Web IDE for SAP HANA' and 'SAP Web IDE Full-Stack'. The main area features a database browser on the left with categories like 'JSON Collections', 'Libraries', 'Procedures', etc. A search bar below it finds 'get\_mailing\_list'. The central workspace contains a 'SQL Console 2.sql' tab with the following content:

```
Connected to: STUDENT05-6mjj27gh1r7udiso-sql_procedure_solution-hdi_core_db (DEV)
1 CALL "SQL_PROCEDURE SOLUTION_HDI_CORE_DB"."get_mailing_list"();
```

Below the SQL console, a message says 'No Results'.

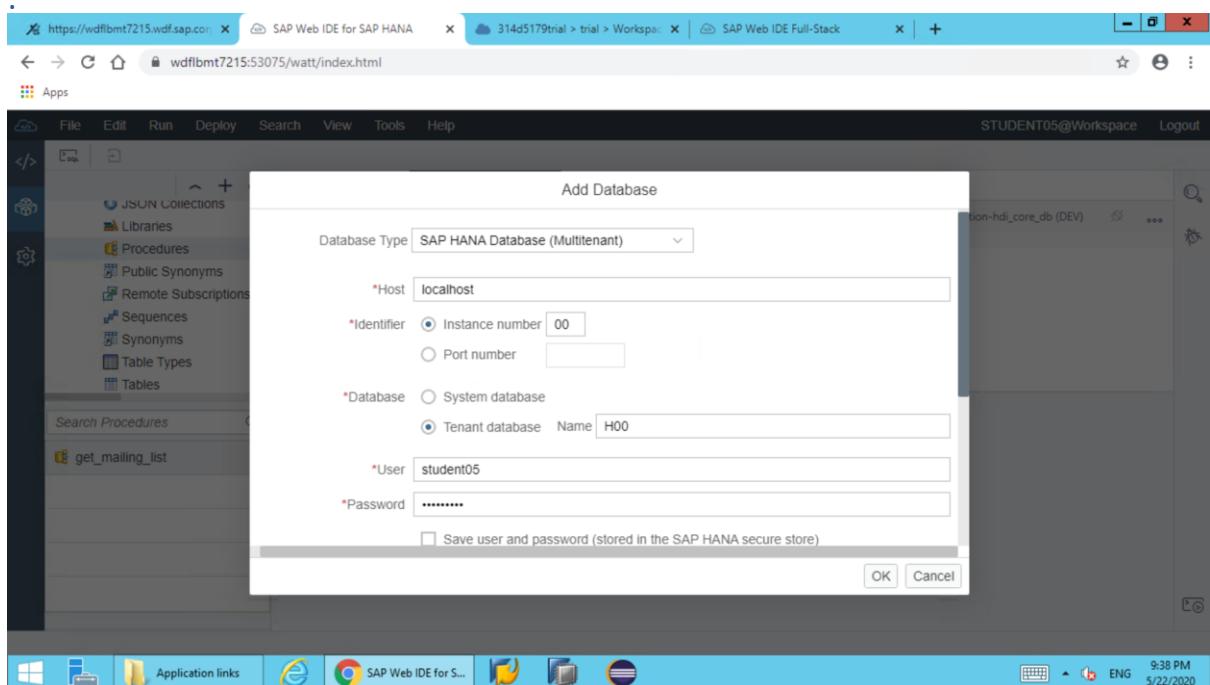
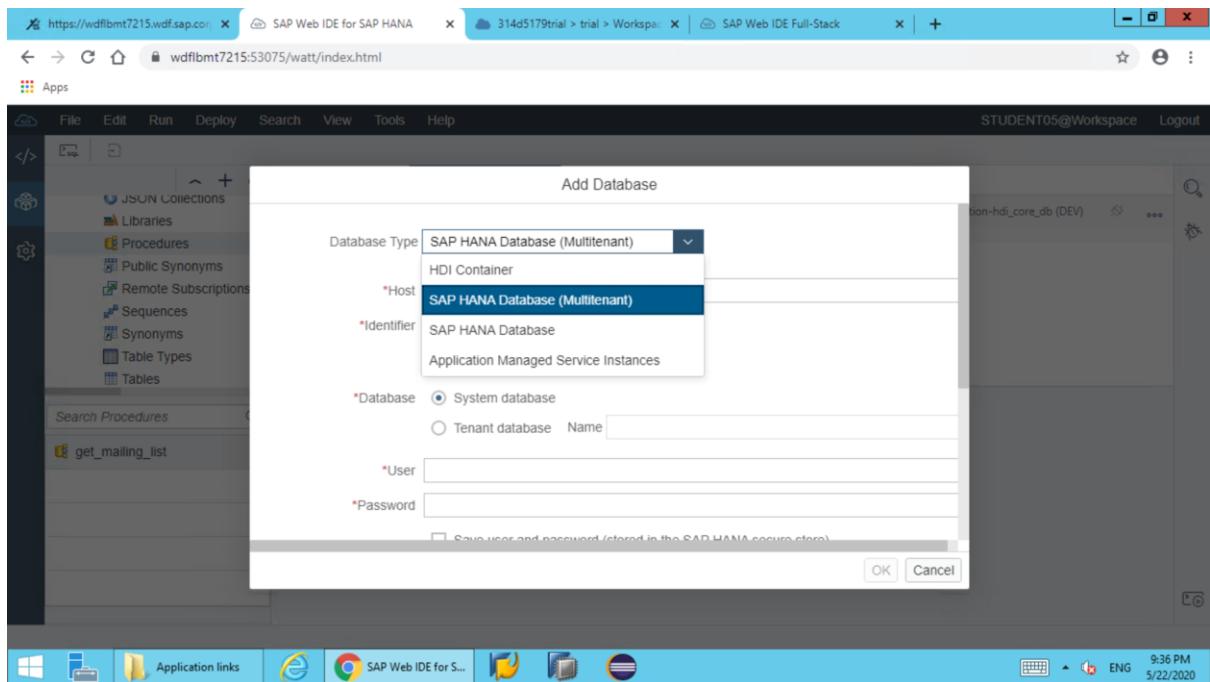
This screenshot shows the SAP Web IDE interface again. The 'Result' tab is active, showing the output of the previous SQL query. The results are displayed in a table with one column labeled 'E\_MAIL' containing the following data:

	E_MAIL
1	akira.kurosawa@unknown.com
2	federico.fellini@unknown.com
3	igmar.bergman@unknown.com
4	john.ford@unknown.com

Now successfully the mailing list is obtained. This can be used to send alerts to the subscribers of news and notifications.

Accessing multiple remote databases / non sap databases data and displaying in Application

In Database explorer, add the remote database by selecting multitenant



Create a schema

The screenshot shows the SAP Web IDE interface. At the top, there are three tabs: 'https://wdflbmt7215.wdf.sap.cor...', 'SAP Web IDE for SAP HANA', and '314d5179trial > trial > Workspace'. The main window has a toolbar with 'File', 'Edit', 'Run', 'Deploy', 'Search', 'View', 'Tools', and 'Help'. On the right, it shows 'STUDENT05@Workspace' and 'Logout'. The left sidebar shows a tree view for 'H00@localhost (student05)' under 'Catalog', listing 'Adapters', 'Agent Groups', 'Agents', 'Column Views', 'Cubes', 'DataStores', and 'Functions'. Below this is a search bar for 'student05' and a 'Search Folders' section with 'Catalog', 'Database Diagnostic Files', and 'HDI Containers'. The central area is titled 'SQL Console 4.sql' and contains the SQL command 'CREATE SCHEMA BOX05;'. The status bar at the bottom shows '9:40 PM 5/22/2020'.

Now create a table named people

```
CREATE TABLE "BOX05"."PEOPLE" (id INT PRIMARY KEY, firstname VARCHAR(50), lastname VARCHAR(50));
```

The screenshot shows the SAP Web IDE interface. In the top navigation bar, there are tabs for 'SQL Console 4.sql', 'SQL Console 5.sql', and 'STUDENT05'. The main workspace displays a SQL query window with the following code:

```
CREATE TABLE "BOX05"."PEOPLE" (id INT PRIMARY KEY, firstname VARCHAR(50), lastname VARCHAR(50));
```

The left sidebar shows a tree view of the database catalog under 'H00@localhost (student05)'. The 'Catalog' node is expanded, showing 'Adapters', 'Agent Groups', 'Agents', 'Column Views', 'Cubes', 'DataStores', and 'Functions'. Below this, the 'student05' schema is selected, and a search bar for 'Search Catalog' is present. The results pane shows 'No Results'.

The screenshot shows the SAP Web IDE interface. In the top navigation bar, there are tabs for 'SQL Console 4.sql', 'SQL Console 5.sql', 'PEOPLE', and 'STUDENT05'. The main workspace displays a table definition for the 'PEOPLE' table:

	Name	SQL Data Type	Column Store Dat...	Key	Not Null	Default	Comment
1	ID	INTEGER	INT	1	X		
2	FIRSTNAME	VARCHAR(50)	STRING				
3	LASTNAME	VARCHAR(50)	STRING				

The left sidebar shows a tree view of the database catalog under 'H00@localhost (student05)'. The 'Catalog' node is expanded, showing 'Adapters', 'Agent Groups', 'Agents', 'Column Views', 'Cubes', 'DataStores', and 'Functions'. Below this, the 'box05' schema is selected, and a search bar for 'Search Tables' is present. The results pane shows 'PEOPLE'.

The screenshot shows the SAP Web IDE interface. In the center, there's a 'Raw Data' view for a 'PEOPLE' table. The table has columns: ID, FIRSTNAME, and LASTNAME. A single row is visible with values: ID=10001, FIRSTNAME=Victor, and LASTNAME=Daniel. On the left, a sidebar shows a tree structure for 'H00@localhost (student05)' under 'Catalog', including Adapters, Agent Groups, Agents, Column Views, Cubes, DataStores, and Functions. Below the sidebar is a search bar for 'Search Tables'. At the top, there are tabs for 'SQL Console 4.sql', 'SQL Console 5.sql', 'PEOPLE', 'PEOPLE', and 'STUDENT05'. The top right shows 'STUDENT05@Workspace Logout'. The bottom navigation bar includes icons for Windows, Application links, and SAP Web IDE.

### Now open XS Advanced Cockpit

The screenshot shows the SAP HANA XS Advanced Cockpit. The left sidebar has sections for Applications, Monitoring, Services, Routes, and Members. The main area is titled 'Space: DEV - Applications' and shows a list of 19 applications. Each application entry includes its name, requested state (all are Started), number of instances (1/1), disk quota (Unlimited), memory (512 MB or 1024 MB), and actions (Edit, Stop, Delete). The top navigation bar shows tabs for Home, SAP, and DEV. The top right shows 'STUDENT05'. The bottom navigation bar includes icons for Windows, Application links, and SAP Web IDE.

Requested State	Name	Instances	Disk Quota	Memory	Actions
Started	7P5RNuwSwmhW91xfsolution2-user-js	1/1	Unlimited	512 MB	
Started	A3KtXBnzz7n4QHnasql-solution-node	1/1	Unlimited	512 MB	
Started	bQDnfRbhrcNz2HWG-HANA-APP-hana-js	1/1	Unlimited	512 MB	
Started	bQEaAcYnYcsV150h-solution-user-js	1/1	Unlimited	512 MB	
Started	di-builder	1/1	Unlimited	1024 MB	
Started	gYyGZTCXJfVDXSbVAN	1/1	Unlimited	512 MB	

The screenshot shows the SAP HANA XS Advanced Cockpit interface. On the left, a sidebar lists various service categories: Applications, Monitoring, Services, Service Marketplace, Service Instances, User-Provided Services (which is selected), Routes, and Members. Below these are Useful Links and Legal Information. The main content area is titled "Space: DEV - User-Provided Services". It displays a table with one row for "CROSS\_SCHEMA\_SERVICE", which has an "Instance Name" of "CROSS\_SCHEMA\_SERVICE" and "Referencing Applications" set to "None". A "New Instance" button is visible. A search bar is at the top right. The bottom part of the screen shows the Windows taskbar with the SAP HANA XS Advanced Cockpit icon.

```
{
  "host": "localhost", "port": "30015", "user": "STUDENT##", "password": "Training1", "driver": "com.sap.db.jdbc.Driv-er", "tags": ["hana"], "schema": "BOX##"}
```

Now go back to webide and in database explorer. Add cross schema db

The screenshot shows two instances of the SAP Web IDE interface. The top instance displays the 'Add Database' dialog, which is open to the 'HDI Container' type. A search bar at the top of the dialog contains the text 'box'. Below it, a table lists an existing HDI container named 'CROSS\_SCHEMA\_BOX05', categorized under 'SAP' in the 'Org' column and 'DEV' in the 'Space' column. The bottom right of the dialog has 'OK' and 'Cancel' buttons. The background of the SAP IDE shows a sidebar with 'Catalog' items like Adapters, Agent Groups, Agents, Column Views, Cubes, DataStores, and Functions. The main workspace shows a table named 'PEOPLE' with columns FIRSTNAME and LASTNAME. The bottom instance of the SAP IDE shows the SQL console with a single row of data:

ID	FIRSTNAME	LASTNAME
10001	Victor	NULL

Now this remote db schema information is available. In order to run the sql commands we can do the heavy lifting using node.js . It has express.js extension which reads sql and transmits to the db layer.

## Running SQL in the Database with Node.js

So remote database queries can be written and transmitted using node.js application server layer.

## Server.js

```
/*eslint no-console: 0*/
"use strict";

// Create the Express application framework
var express = require("express");
var app = express();

// Set the port the application listens to
var port = process.env.PORT || 3000;
app.listen(port, function() {
    console.log("listening on port " + port);
```

```

});

// Get XS services
var xsenv = require("@sap/xsenv");
var services = xsenv.getServices({
    hana: {tag: "hana"}
});

// Get HANA middleware and assign it to the application
var hdbext = require("@sap/hdbext");
app.use("/", hdbext.middleware(services.hana));

// Set the function to be executed when you access the application URL
app.get(
    "/",
    function(req, res, next) {
        req.db.exec(
            "SELECT COUNT(EMPLOYEEID) AS COUNT FROM \"MD.Employees\"",
            function(err, rows) {
                if (err) {
                    return next(err);
                }
                res.send("Current employees: " + rows[0].COUNT);
            }
        );
    }
);

```

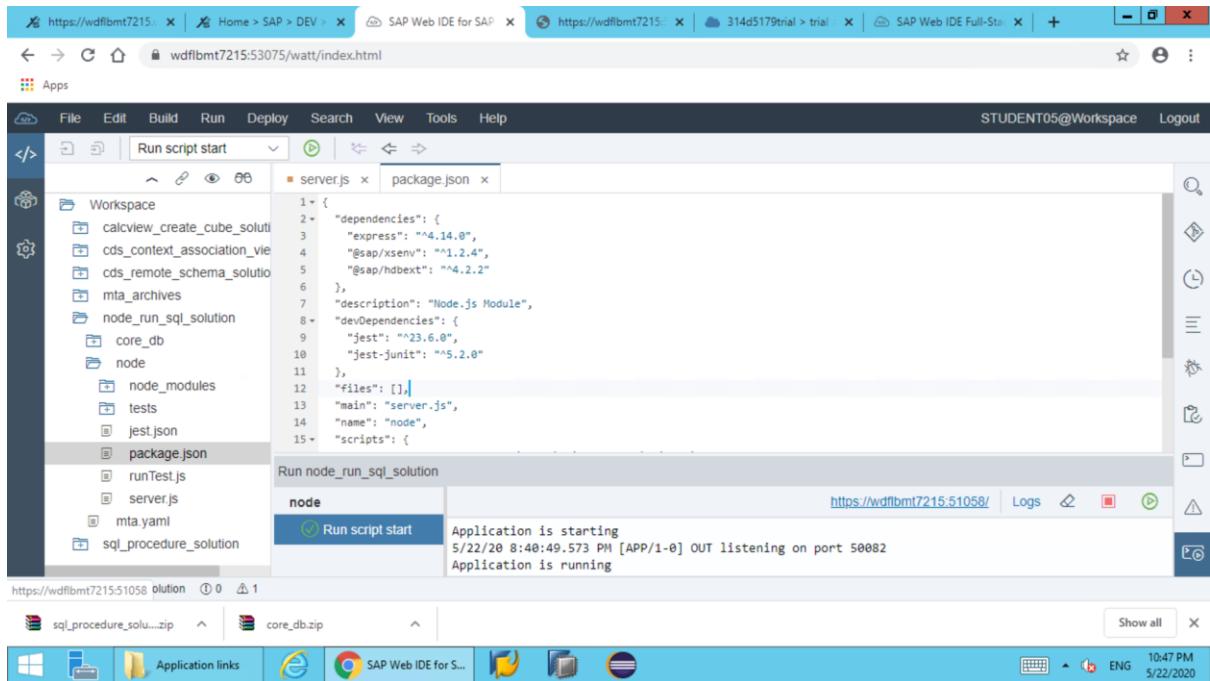
### Package.json

```
{
  "dependencies": {
    "express": "^4.14.0",
    "@sap/xsenv": "^1.2.4",
    "@sap/hdbext": "^4.2.2"
  },
  "description": "Node.js Module",
  "devDependencies": {
    "jest": "^23.6.0",
    "jest-junit": "^5.2.0"
  },
  "files": [],
  "main": "server.js",
}
```

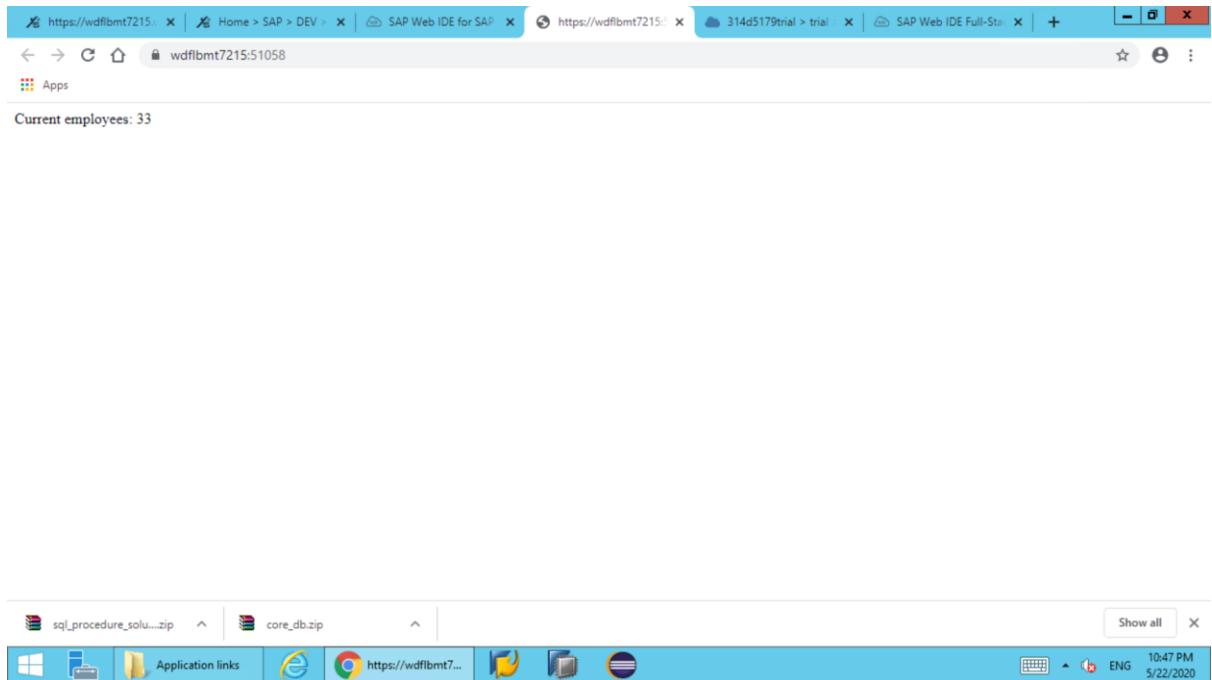
```

  "name": "node",
  "scripts": {
    "test": "node node_modules/jest/bin/jest --config jest.json",
    "test-coverage": "node node_modules/jest/bin/jest --coverage --config jest.json",
    "start": "node server.js"
  },
  "engines": {
    "node": "10.x"
  },
  "version": "1.0.0"
}

```



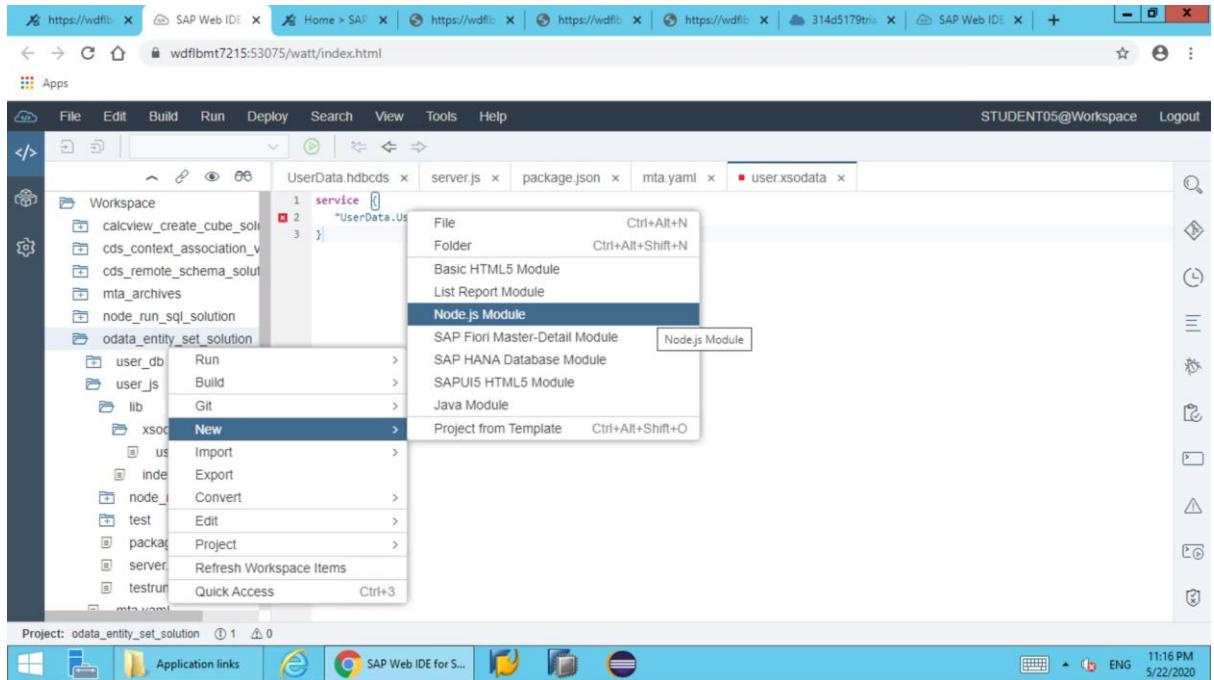
For node.js layer, only server.js and package.json are enough rest all files are not required. And they can be removed for sql commands.



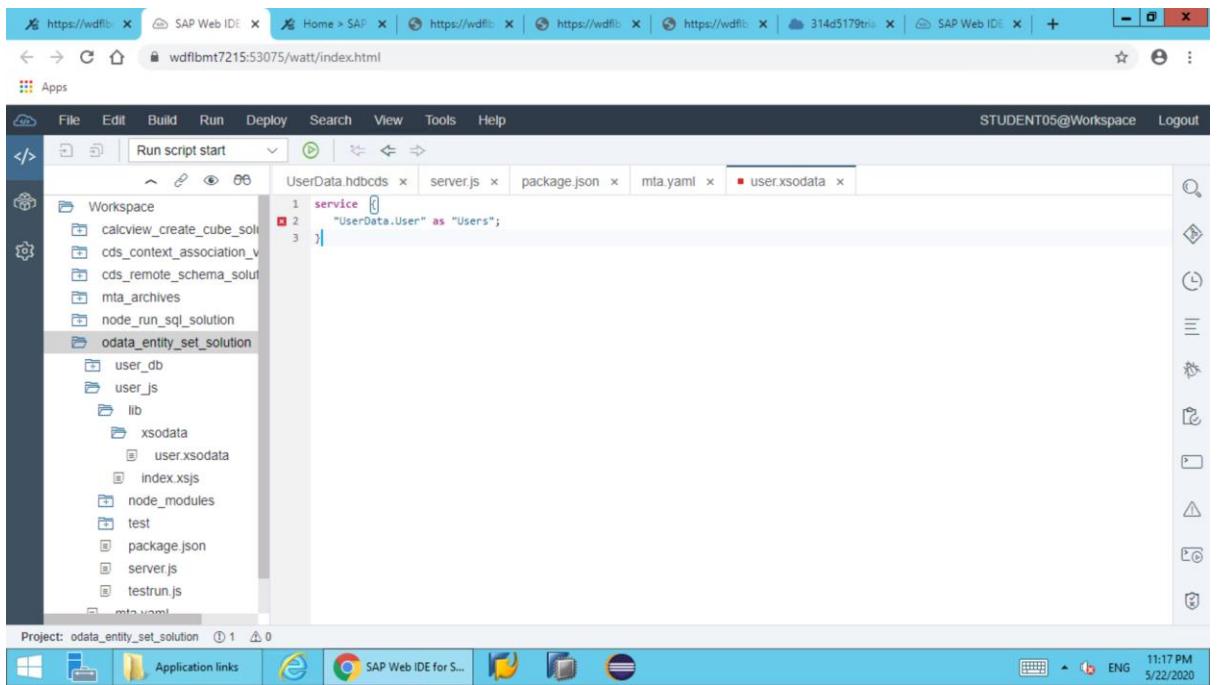
We could fetch no of employees count using sql command from node.js layer

## Generating XSodata

### 1. Create a node.js module with name user\_js



Create a folder xsodata and add new file user.xsodata



Add below snippet

```
service {
  "UserData.User" as "Users";
}
```

Now go to mta.yaml file.

Make sure these properties are defined properly using mta editor

The screenshot shows the SAP Web IDE interface. The top navigation bar includes tabs for SAP Web IDE, Home > SAP, https://wdflbmt7215:53075/watt/index.html, https://wdflbmt7215:53075/watt/index.html, https://wdflbmt7215:53075/watt/index.html, https://wdflbmt7215:53075/watt/index.html, https://wdflbmt7215:53075/watt/index.html, and SAP Web IDE. The main workspace shows a project structure under 'Workspace' with various files like calcview\_create\_cube\_solution, cds\_context\_association\_view, cds\_remote\_schema\_solution, mta\_archives, node\_run\_sql\_solution, odata\_entity\_set\_solution, user\_db, user\_js, lib, node\_modules, test, package.json, server.js, testrun.js, mta.yaml, and sql\_procedure\_solution. The 'user\_db' module is selected. The right-hand panel displays the 'Requires' section, which lists 'user\_db' (hdb) and 'user\_js' (nodejs). Below this, there are tables for 'Properties of user\_db' and 'Parameters of user\_db'.

This screenshot shows the SAP Web IDE interface again. The 'user\_js' module is selected in the workspace. The right-hand panel displays the 'Provides' section, which lists a provider named 'user\_js\_api' with a key 'url' and value '\${default-url}'. Below this, the 'Parameters' section is visible. The bottom status bar shows the time as 11:15 PM on 5/22/2020.

Now verify the same in code editor as below

## Mta.yaml

ID: odata\_entity\_set

\_schema-version: '2.0'

version: 0.0.1

modules:

- name: user\_db

```

type: hdb
path: user_db
requires:
- name: hdi_user_db
properties:
  TARGET_CONTAINER: '~{hdi-container-name}'
- name: user_js
type: nodejs
path: user_js
provides:
- name: user_js_api
properties:
  url: '${default-url}'
requires:
- name: user_db
- name: hdi_user_db
resources:
- name: hdi_user_db
properties:
  hdi-container-name: '${service-name}'
type: com.sap.xs.hdi-container

```

Build and run as node.js application.

Replace <https://wdflbmt7215:51059/index.xsjs> with  
<https://wdflbmt7215:51059/xsodata/user.xsodata/Users>

```

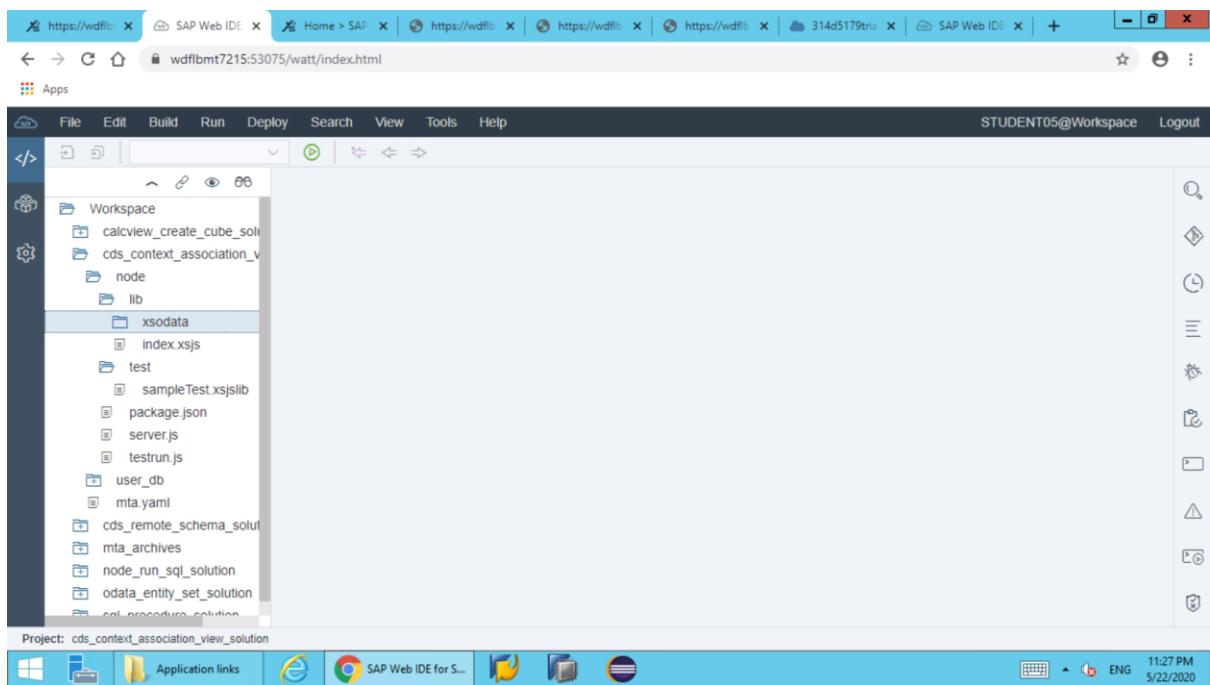
{
  "d": {
    "results": [
      {
        "_metadata": {
          "uri": "https://wdflbmt7215:51059/xsodata/user.xsodata/Users(1000000000)", "type": "default.UsersType"
        }, "User_Id": 1000000000, "First_Name": "Akira", "Last_Name": "Kurosawa", "Email": "aki.ra.kurosawa@unknown.com"
      },
      {
        "_metadata": {
          "uri": "https://wdflbmt7215:51059/xsodata/user.xsodata/Users(1000000001)", "type": "default.UsersType"
        }, "User_Id": 1000000001, "First_Name": "Federico", "Last_Name": "Fellini", "Email": "federico.fellini@unknown.com"
      },
      {
        "_metadata": {
          "uri": "https://wdflbmt7215:51059/xsodata/user.xsodata/Users(1000000002)", "type": "default.UsersType"
        }, "User_Id": 1000000002, "First_Name": "Igmar", "Last_Name": "Bergman", "Email": "igmar.r.bergman@unknown.com"
      },
      {
        "_metadata": {
          "uri": "https://wdflbmt7215:51059/xsodata/user.xsodata/Users(1000000003)", "type": "default.UsersType"
        }, "User_Id": 1000000003, "First_Name": "John", "Last_Name": "Ford", "Email": "john.ford@unknown.com"
      }
    ]
  }
}

```

Windows taskbar at the bottom:

## Creating XSodata with Associations

This is continuation to initial cds views with association exercise. Create node.js module with name node and add a subfolder xsodata in lib path.



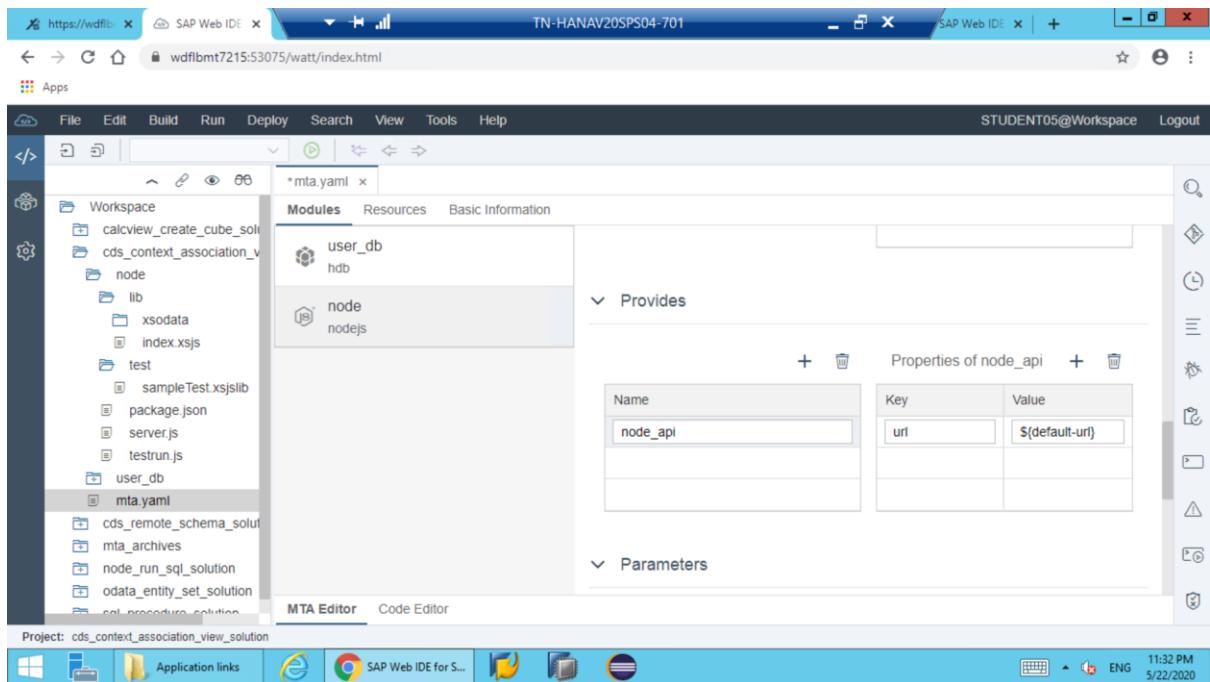
Go to mta.yaml file and select node module

The screenshot shows the SAP Web IDE interface with the MTA Editor open. The project is named 'cds\_context\_association\_view\_solution'. The 'mta.yaml' file is selected. In the 'Modules' tab, 'user\_db' and 'node' are listed. The 'Requires' section is currently empty, with two tables: 'Properties of hdi\_user...' and 'Properties of user\_db...' both showing 'No data'.

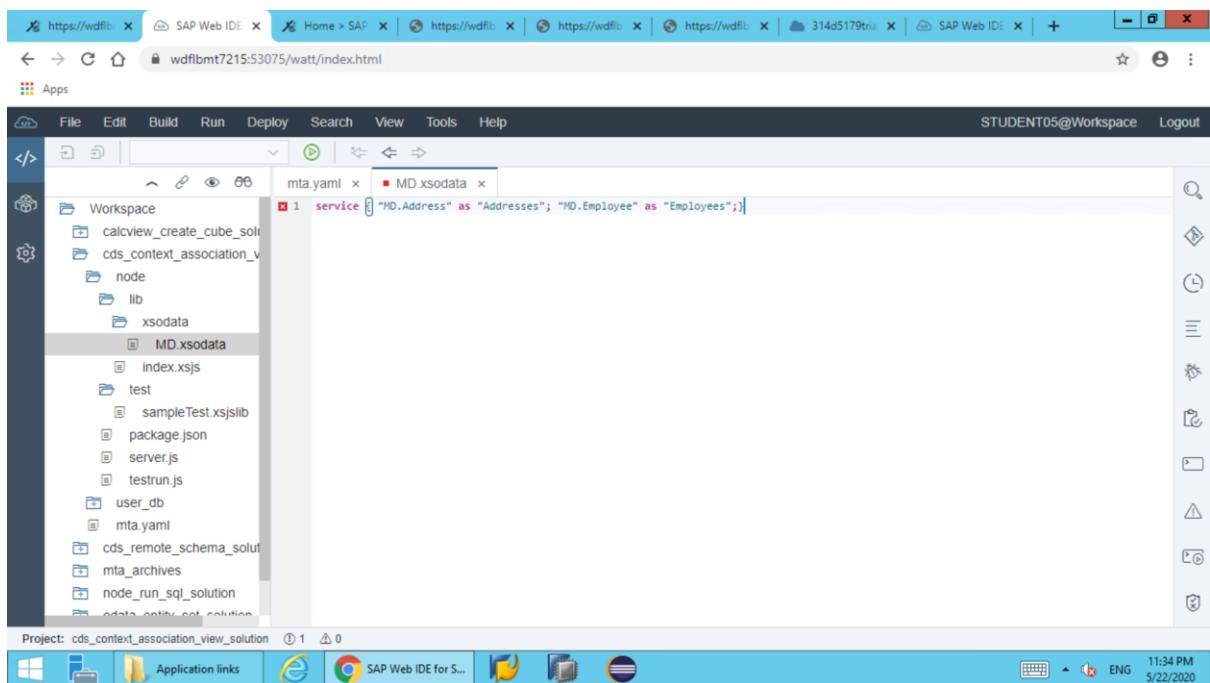
Add below values in Requires section

The screenshot shows the SAP Web IDE interface with the MTA Editor open. The project is named 'cds\_context\_association\_view\_solution'. The 'mta.yaml' file is selected. The 'Requires' section now includes entries for 'hdi\_user\_db (resource)' and 'user\_db (module)'. The 'Properties of user\_db' and 'Parameters of user\_db' sections also show 'No data'.

And in provide section , see below entries are available

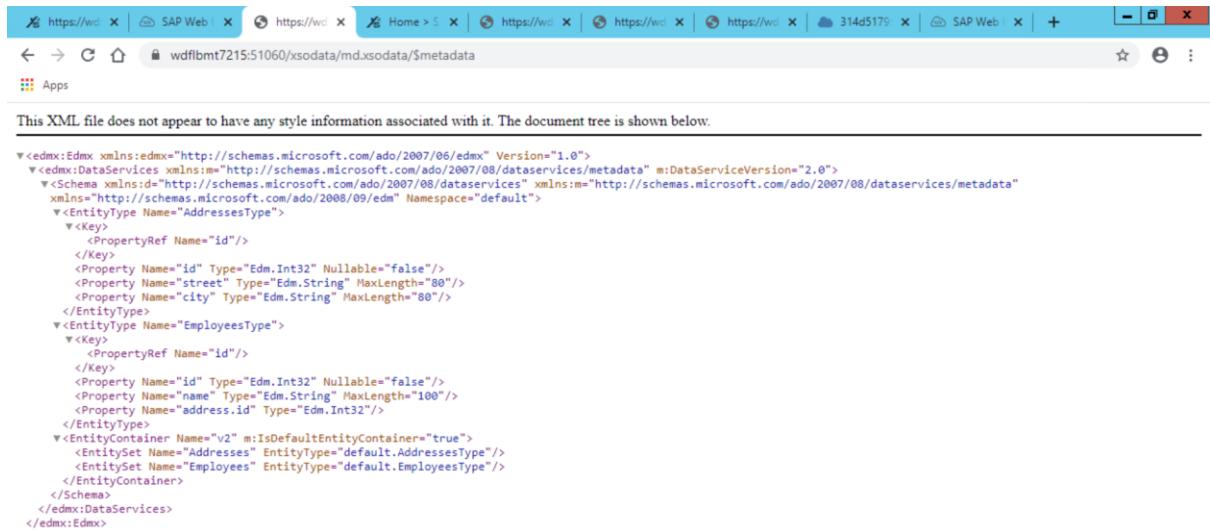


Now go to xsodata folder and add file md.xsodata



```
service {
  "MD.Address" as "Addresses";
  "MD.Employee" as "Employees";
}
```

Now build and run the app as node.js application.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">
  <edmx:DataServices xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" m:DataServiceVersion="2.0">
    <Schema xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://schemas.microsoft.com/ado/2008/09/edm" Namespace="default">
      <EntityType Name="AddressesType">
        <Key>
          <PropertyRef Name="id"/>
        </Key>
        <Property Name="id" Type="Edm.Int32" Nullable="false"/>
        <Property Name="street" Type="Edm.String" MaxLength="80"/>
        <Property Name="city" Type="Edm.String" MaxLength="80"/>
      </EntityType>
      <EntityType Name="EmployeesType">
        <Key>
          <PropertyRef Name="id"/>
        </Key>
        <Property Name="id" Type="Edm.Int32" Nullable="false"/>
        <Property Name="name" Type="Edm.String" MaxLength="100"/>
        <Property Name="address.id" Type="Edm.Int32"/>
      </EntityType>
      <EntityContainer Name="v2" m:IsDefaultEntityContainer="true">
        <EntitySet Name="Addresses" EntityType="default.AddressesType"/>
        <EntitySet Name="Employees" EntityType="default.EmployeesType"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>

```



[https://wdflbmt7215:51060/xsodata/md.xsodata/\\$metadata](https://wdflbmt7215:51060/xsodata/md.xsodata/$metadata)

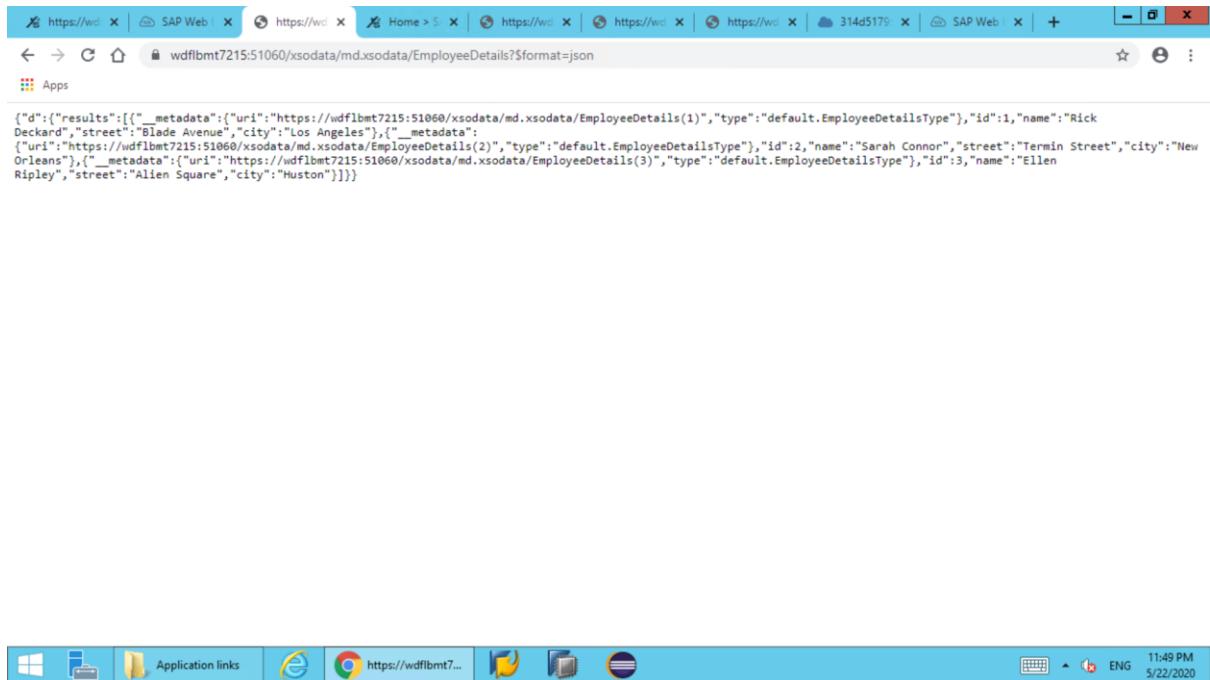
Now lets add the Employee details view to look at the association.

For this we need to add md.xsodata file with viewname along with a key

```

service {
  "MD.Address" as "Addresses";
  "MD.Employee" as "Employees";
  "MD.EmployeeDetails" as "EmployeeDetails" key ("id");
}

```



Now we could display the views.

Lets add association . Add below snippet in MD.xsodata file

```

service {

  "MD.Address" as "Addresses";
  "MD.Employee" as "Employees";
  "MD.EmployeeDetails" as "EmployeeDetails" key ("id");

  association "EmployeeAddress" with referential constraint principal "Employees"("address.id") multiplicity "1"
  dependent "Addresses"("id") multiplicity "0..1";

}

```

Lets add navigation. Overwirte MD.xsodata with below snippet

```

service {

  "MD.Address" as "Addresses";
  "MD.Employee" as "Employees";
  "MD.EmployeeDetails" as "EmployeeDetails" key ("id");

  association "EmployeeAddress" with referential constraint principal
  "Employees"("address.id") multiplicity "1" dependent "Addresses"("id") multiplicity "0..1";
}

```

```
"MD.Address" as "Addresses" navigates ("EmployeeAddress" as "MyEmployee");
"MD.Employee" as "Employees" navigates ("EmployeeAddress" as "MyAddress");
```

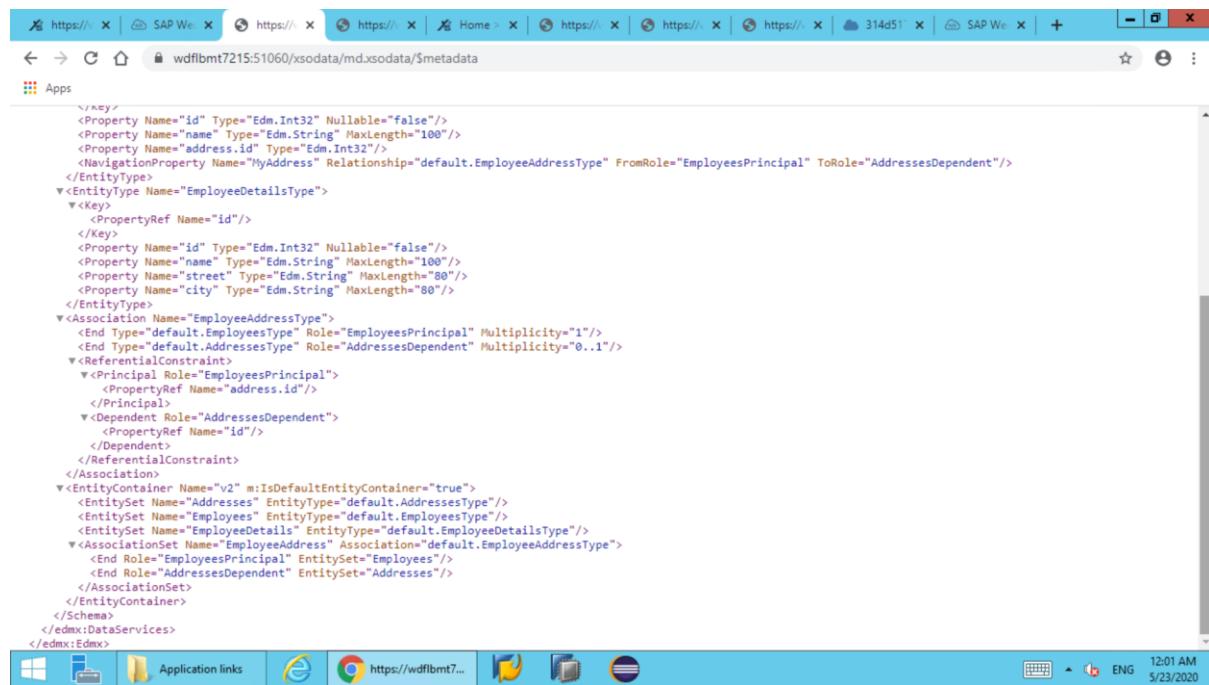
```
}
```

In Simple form, snippet could be written as below

### Final Snippet

```
service {
    "MD.Address" as "Addresses" navigates ("EmployeeAddress" as "MyEmployee");
    "MD.Employee" as "Employees" navigates ("EmployeeAddress" as "MyAddress");
    "MD.EmployeeDetails" as "EmployeeDetails" key ("id");

    association "EmployeeAddress" with referential constraint principal "Employees"("address.id") multiplicity "1"
    dependent "Addresses"("id") multiplicity "0..1";
}
```



```

{
  "d": {
    "__metadata": {
      "uri": "https://wdflbmt7215:51060/xsodata/md.xsodata/Addresses(3)",
      "type": "default.AddressesType"
    },
    "id": 3,
    "street": "Termin Street",
    "city": "New Orleans",
    "MyEmployee": {
      "__deferred": {
        "uri": "https://wdflbmt7215:51060/xsodata/md.xsodata/Addresses(3)/MyEmployee"
      }
    }
  }
}

```

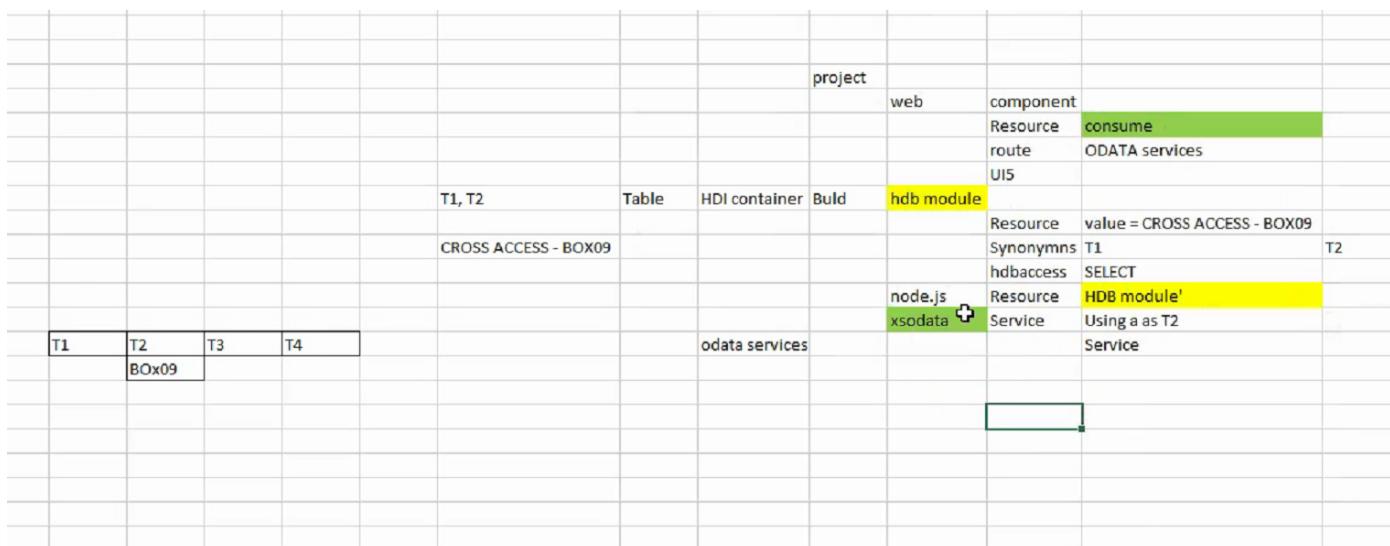


Finally , we could navigate to association and display records. [Add green highlighted code and run directly if concept is clear]



XSOADATA Mapping Use Case Snapshot:

xodata is consumed at Web Application end while it's a service at node -Application server layer.



Thank you.