

SAP HANA Cloud

Collaborative Database Development in SAP HANA Cloud, SAP HANA Database

**Session 2: Create a CAP Application and
Automate Project Deployment using CI/CD**

**SAP Data and Analytics
Virtual Forum:
Accelerating Outcomes**

GET STARTED

WELCOME	4
ABOUT THIS SESSION	4
PREREQUISITES	5
INTRODUCTION	6
CREATE A CAP DEV SPACE AND SET UP A CAP CDS PROJECT	7
CREATE A NEW DEV SPACE	7
CONNECT TO CLOUD FOUNDRY	10
ADD A LOCAL GIT REPOSITORY	12
ADJUST THE MTA.YAML FILE	13
ADJUST THE PACKAGE.JSON FILE	15
INSTALL NODE MODULES AND HANA CLIENT TOOL	17
CREATE A PACKAGE.JSON FILE IN THE SRV FOLDER	18
DEFINE AND CHANGE TABLES USING CAP CDS	19
CREATE THE INVENTORY.CDS FILE TO GENERATE YOUR TABLES	19
GENERATE YOUR TABLE WITH CDS BUILD	21
ADD DATA TO THE INVENTORY TABLE	23
GENERATE AN HDBMIGRATIONTABLE	24
MIGRATION TO VERSION 2	25
MIGRATION TO VERSION 3	28
SET UP CI/CD CHAIN	30
CREATE A NEW GITHUB REPOSITORY	30
CONNECT THE GITHUB REPOSITORY TO YOUR CAP PROJECT	31
CREATE A GITHUB TOKEN	32

SUBSCRIBE TO CI/CD AND MANAGE CI/CD ADMINISTRATOR RIGHTS	33
CI/CD: ADD CREDENTIALS	36
CI/CD: DEFINE THE GITHUB REPOSITORY	37
CI/CD: DEFINE AND TRIGGER A JOB	38
AUTOMATE THE CI/CD CHAIN WITH A WEBHOOK	41
ADD WEBHOOK CREDENTIALS TO GITHUB	41
TEST THE AUTOMATIC DEPLOYMENT USING THE WEBHOOK	43
ADDITIONAL RESOURCES	44

<https://community.sap.com/topics/hana-cloud>

© 2021 SAP SE or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.
The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.
These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions. Service names mentioned are the trademarks of their respective companies. See www.sap.com/copyright for additional trademark information and notices.

WELCOME

Join the SAP HANA Cloud Workshop series to learn from experts and follow along live using your own SAP HANA Cloud (trial) account.

Learn in this workshop how database development in SAP HANA Cloud and SAP Business Application Studio can be done collaboratively. You will get to know basic HDI concepts, ways to develop applications and to automate their deployment as well as creating template projects that others in your organization can easily use and modify.

About this session

Are you struggling with time-intensive manual deployment of your development projects? In the second part of this hands-on workshop, follow along as SAP Product Manager, Volker Saggau, will walk you through the process of creating a CAP application and automating its deployment, using Continuous Integration & Delivery. In this session, you'll learn how to:

- Use Cloud Application Programming and CDS for schema evolution
 - Create a CAP development project with SAP HANA tools
 - Set up CDS and CAP (libraries, packages, folder structure)
 - CDS code and graphical view editor
- Automating project deployment with SAP Continuous Integration and Delivery
 - Set up CI/CD pipeline
 - Prepare the project deployment flow
 - Deploy your project automatically

Your presenters:



Volker Saggau
Product Manager, SAP HANA Cloud | SAP



Jan Zwickel
Product Manager, SAP HANA Cloud | SAP

Prerequisites

Here is what you need to prepare before the workshop starts to be able to follow along:

1. [Sign up](#) for the free SAP HANA Cloud trial.
2. Provision an instance of SAP HANA Cloud, SAP HANA database and make sure it is running.
3. Subscribe to the free SAP Business Application Studio trial.
4. Subscribe to the CI/CD service in BTP cockpit.
5. Create a free account on www.github.com to use the required sample data.

The steps needed to prepare are all covered in our [preparation reader](#).

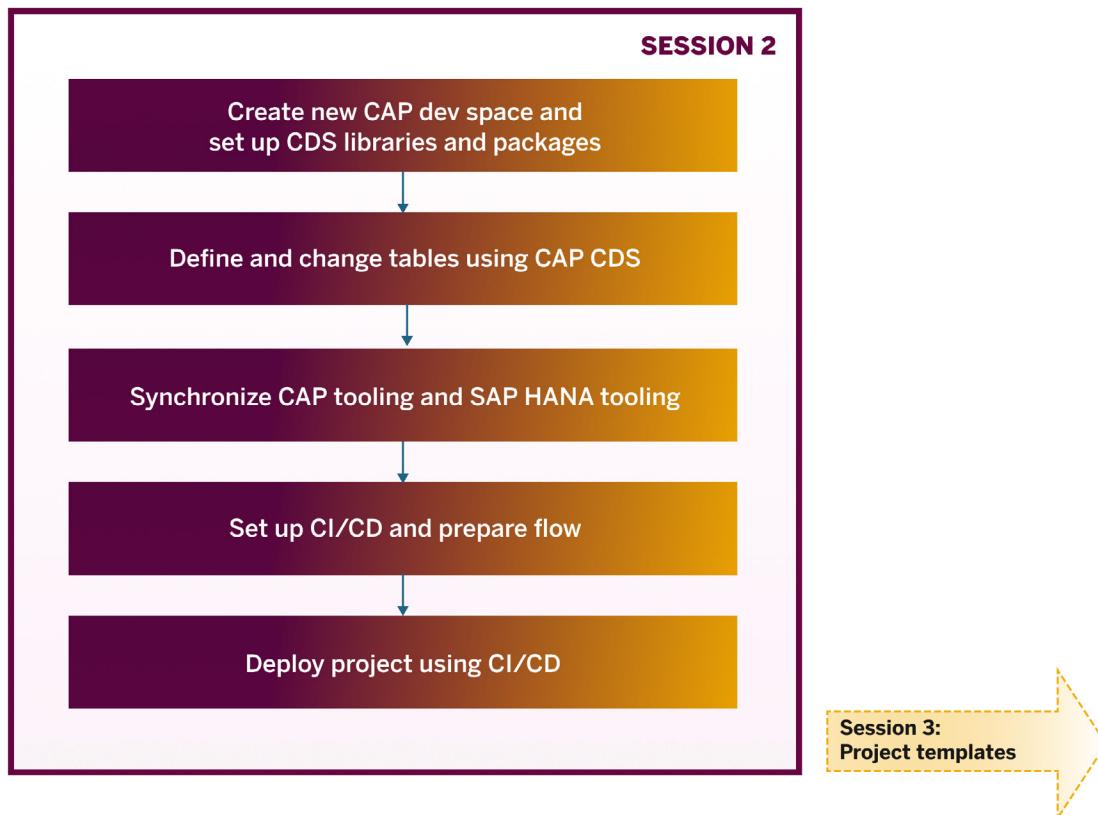
If you do, however, have any issues, please ask a question in the [SAP Community](#) using the tag SAP HANA Cloud or refer to the [SAP HANA Cloud Technical Documentation](#).

Introduction

In the previous session, you created an SAP HANA native application development space and project in SAP Business Application Studio. In this project, you created an hdbtable object and imported sample data. You then changed the data structure of this table, first using the existing hdbtable object and secondly using an hdmigrationtable object, that allows for smoother changes that consume less resources. You manually deployed all changes to the design-time objects to your SAP HANA Cloud, SAP HANA database.

In this next session, you will learn how you can create a CAP CDS development space and project, which allows you to automate the generation of your design-time objects. We will basically perform the same steps, creating hdbtable and hdmigrationtable objects and then adding columns to these tables, but using a CDS file that helps you generate the design-time objects in your project.

In the second part of this session, you will learn how you can automate the deployment of your project using the Continuous Integration & Delivery (CI/CD) service, one of the many services available in SAP BTP. CI/CD will allow you to fully automate the deployment chain using a webhook and github repository. At the end of this session, you will have a setup in which whenever changes are made to your project and pushed to your git, these changes will be automatically deployed to your database.



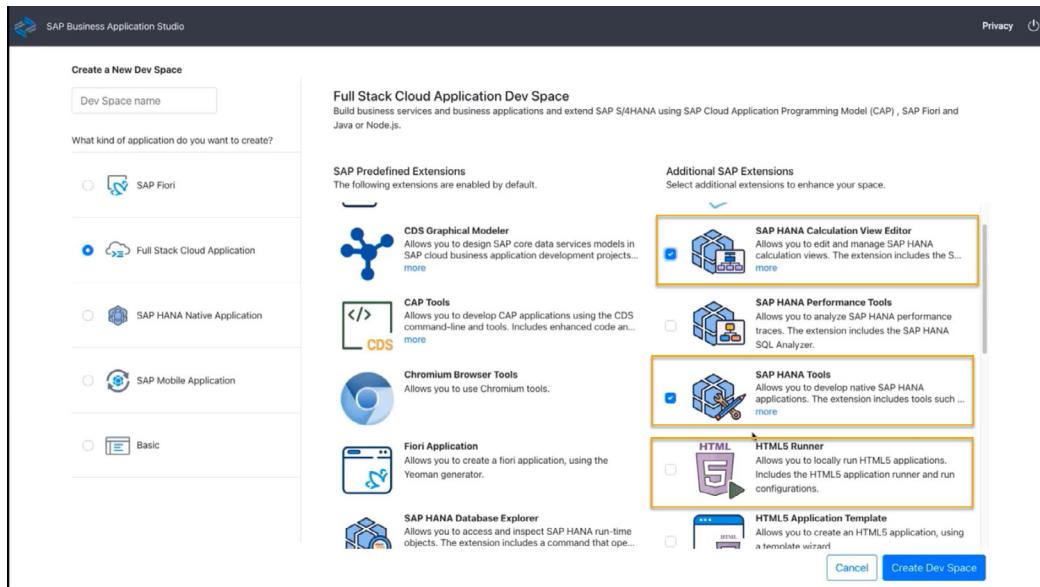
CREATE A CAP DEV SPACE AND SET UP A CAP CDS PROJECT

Create a new Dev Space

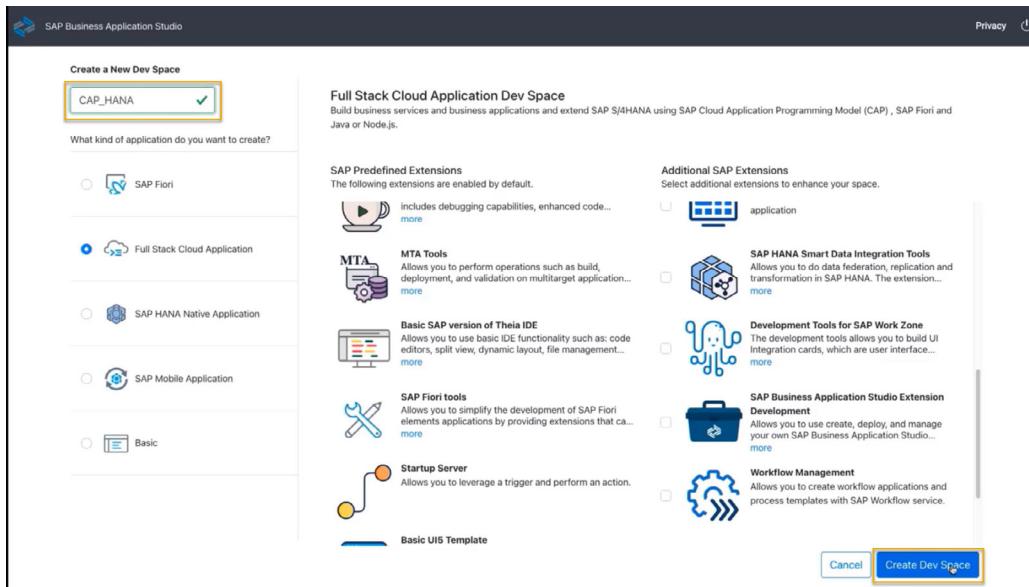
1. Open **SAP Business Application Studio**.
2. On the page with your Dev Spaces, click on **Create Dev Space**.

Important: Ensure that you have no running Dev Space, since you can only have a single running Dev Space at a time as a trial user. Also, you cannot have more than two existing Dev Spaces as a trial user. Therefore, delete any other Dev Spaces you have other than the one (Dev Space WS3_1) you created in the first session of this workshop series.

3. In the Create a New Dev Space window, select **Full Stack Cloud Application** as the option for the kind of application you want to create.
4. Next, select **SAP HANA Calculation View Editor**, **SAP HANA Tools** and **HTML5 Runner** from the list of Additional SAP Extensions to add to your development space.



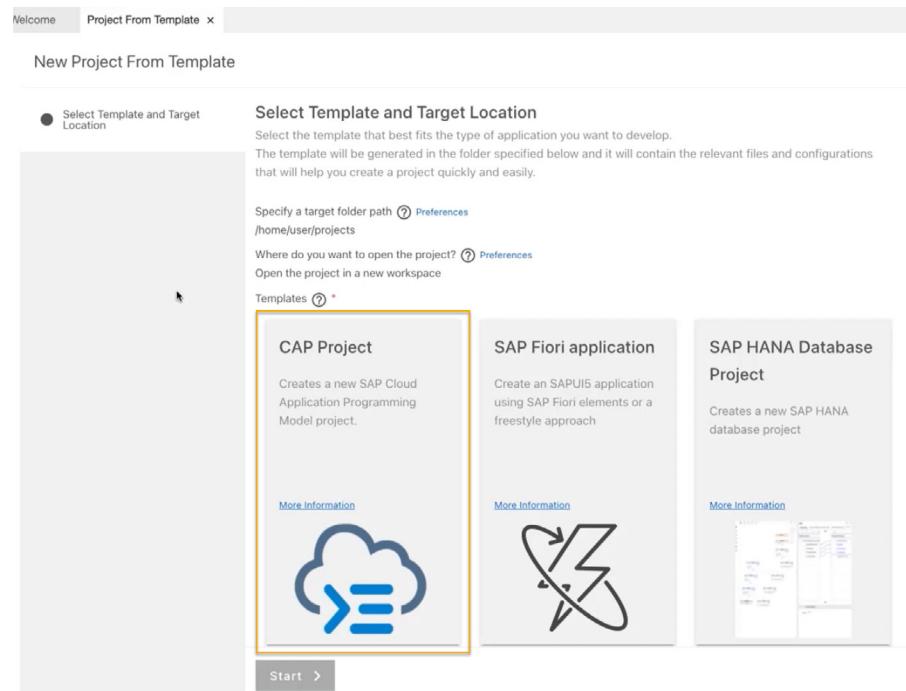
5. Name your Dev Space as **CAP_HANA** in the box provided at the top. Click on **Create Dev Space**.



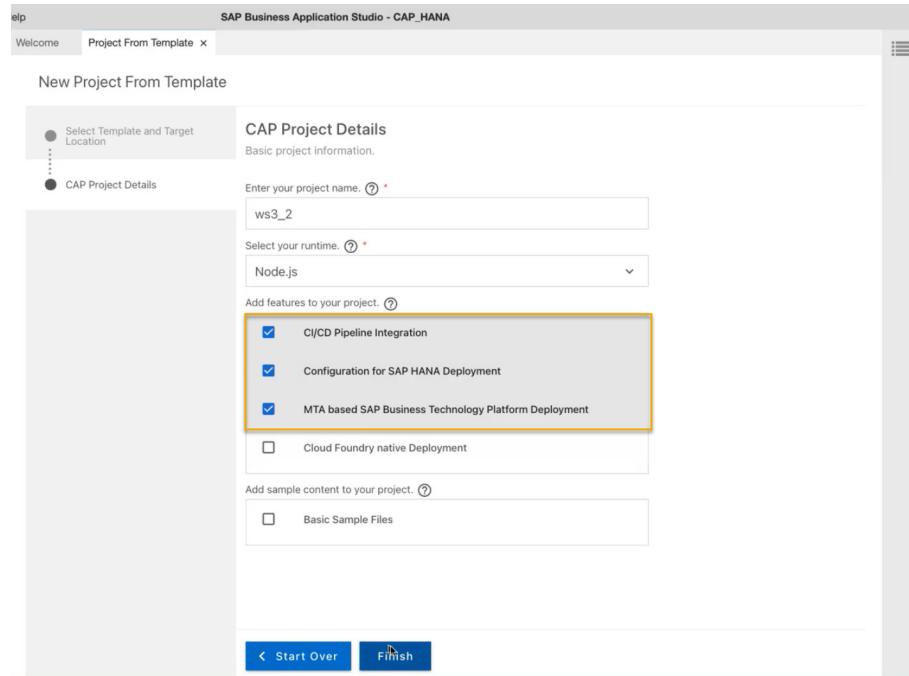
Note: The name of your Dev Space is valid for use only if a green tick mark becomes visible alongside the name. This name must start with a letter or number and only allows alphanumeric characters and underscores, no special characters.

6. You will return to the page with the list of Dev Spaces in SAP Business Application Studio. Creating the dev space usually takes a few minutes. Once the status changes to **RUNNING**, you can open it.
7. After you have opened your dev space in SAP Business Application Studio, select **Start from template** from the welcome screen to create a new project.

8. Choose **CAP Project** and click on **Start**.



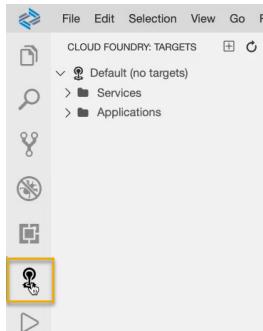
9. In the field 'Add Basic Information', add the name of your new Project as **ws3_2**. It's important to have the name of the project in lower case characters.
10. Click on **Next**.
11. In the field 'Select your runtime', select **Node.js**.
12. In the field
13. 'Add features to your project' option, check the boxes for **CI/CD Pipeline Integration, Configuration for SAP HANA Deployment** and **MTA based SAP Business Technology Platform Deployment**.



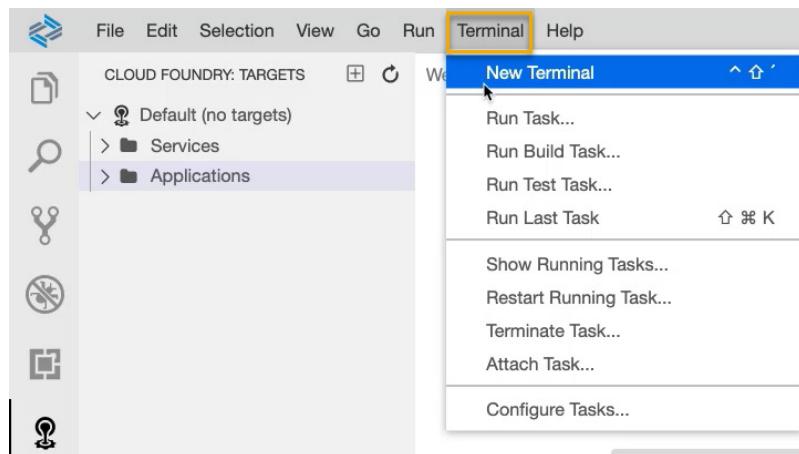
14. Click on **Finish** to complete the project generation. It might take a few minutes to complete the process.

Connect to Cloud Foundry

- Click on the Cloud Foundry icon on the left-side of the screen. In the adjacent panel, you can see the connections to Cloud Foundry.



- From the horizontal menu bar at the top, click on **Terminal**. Select **New Terminal**.



- In the new terminal, run the following command.

```
cf login
```

The output will ask you to enter the API endpoint. Follow the next step to access this information.

- Open a new window on your browser. Go to your **subaccount** in SAP BTP cockpit and copy the API endpoint information given under **Cloud Foundry Environment** section.

A screenshot of the SAP BTP cockpit interface, specifically the 'Subaccount: [redacted] - Overview' page. The left sidebar shows 'Overview', 'Services', 'Cloud Foundry' (selected), 'HTML5 Applications', 'Connectivity', 'Security', 'Entitlements', and 'Usage Analytics'. The main panel shows 'Subaccount: [redacted] - Overview' with tabs for 'General', 'Cloud Foundry Environment', 'Kyma Environment', and 'Entitlements'. Under 'Cloud Foundry Environment', it shows '88 Entitlements' and '14 Instances & Subscriptions'. It lists 'Subdomain: [redacted]', 'Tenant ID: [redacted]', 'Region: [redacted]', and 'Subaccount ID: [redacted]'. At the bottom, there's a table for 'Cloud Foundry Environment' with columns 'Name', 'Applications', and 'Service Instances'. One row is shown: 'dev' with '3' applications and '7' service instances. An 'API Endpoint: https://api.cf.eu10.hana.ondemand.com' field is highlighted with a yellow box. Below it, 'Org Name: [redacted]' and 'Org ID: [redacted]' are listed. Buttons for 'Create Space' and 'Manage environment instance' are visible.

5. Return to the open terminal in SAP Business Application Studio. Paste your API endpoint in the space provided and press **Enter**.
6. As the next step, you will be asked to enter your email. Enter your **email id** used to login into the trial account. Press **Enter**.
7. Enter your **password** for the login id. Press **Enter**.

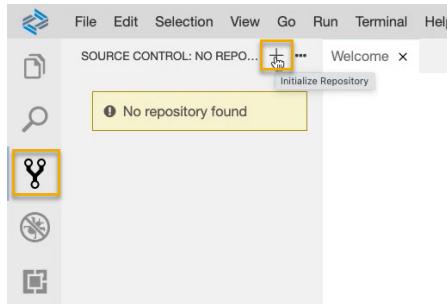
Note: If you have multiple Organizations, you will have to select the one having your trial account next. Once you have completed the above steps, you can see the connection details of your target organization as the result.

8. In the **CLOUD FOUNDRY: TARGETS** panel you can check the connection has been made by expanding **Services** and **Applications**.

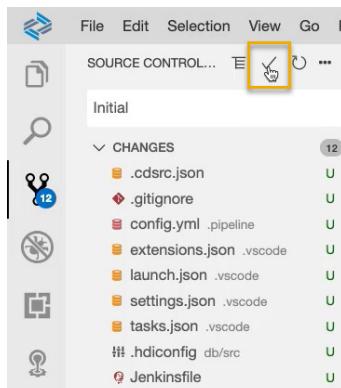
Add a Local Git repository

To be able to track the changes you make to your project in subsequent steps, we recommend adding a local git repository to your project. We will later sync this local repository with a remote GitHub repository.

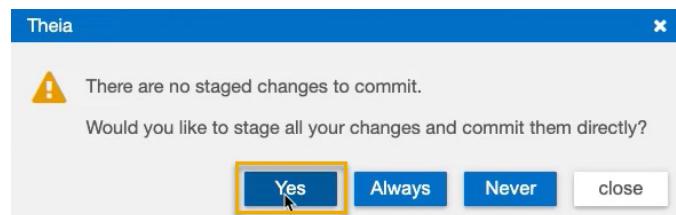
1. In SAP Business Application Studio, click on the  icon to go to the **Source Control** panel.
2. Click on the **plus icon** to initialize the Git repository.



3. Type **Initial** in the Message box under the Source Control panel. Click on the **tick mark icon** at the top of the panel to stage all your changes and commit them.



4. If you encounter a message box about staging all changes and committing them directly, select **Yes**. You may also select the option **Always**.

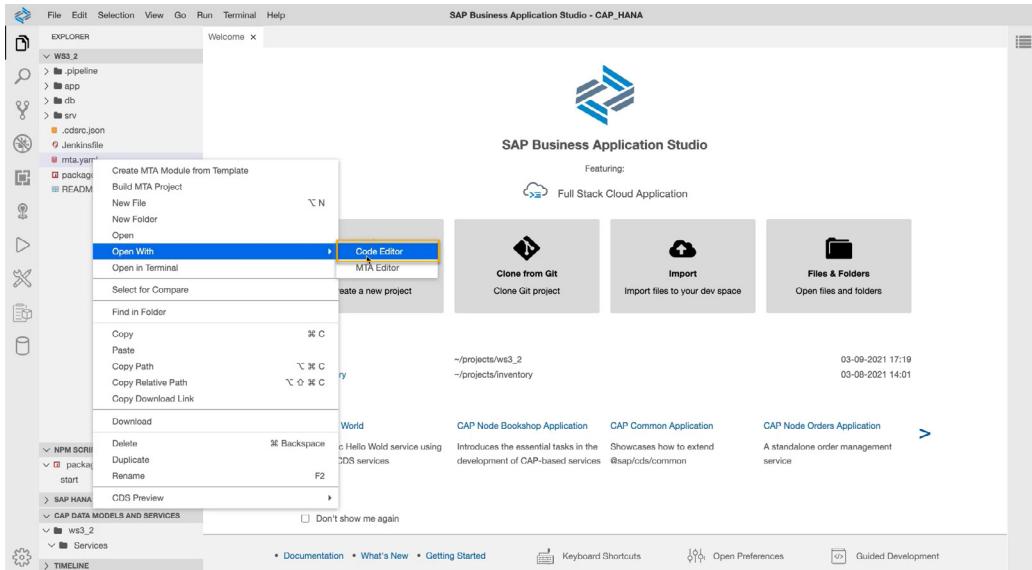


Now the basic setup of your new project is done. In the next steps, we will show you how to make necessary adjustment to important files for your project to work properly with your SAP HANA Cloud, SAP HANA database.

Adjust the mta.yaml file

The first file that needs to be adjusted is the **mta.yaml** file.

1. Go back to the **file explorer** and expand the folders under your project **ws3_2**.
2. From the list of files in the project, right-click on the **mta.yaml** file. Select **Open With** and choose **Code Editor**.



3. Within the code editor, you need to make two changes: First, change the path of Server Module from **gen/srv** to **srv**. Second, change the path of Sidecar Module from **gen/db** to **db**. Refer to the image below to make these changes.

```

parameters:
  enable-parallel-deployments: true
build-parameters:
  before-all:
    - builder: custom
      commands:
        - npm install --production
        - npx -p @sap/cds-dk cds build --production
modules:
  # ----- SERVER MODULE -----
  - name: ws3_2-srv
  # ----- SIDECAR MODULE -----
  - type: nodejs
    path: gen/srv
    parameters:
      buildpack: nodejs_buildpack
    requires:
      # Resources extracted from CAP configuration
      - name: ws3_2-db
    provides:
      - name: srv-api      # required by consumers of CAP services (e.g. approuter)
        properties:
          srv-url: ${default-url}
  # ----- SIDECAR MODULE -----
  - name: ws3_2-db-deployer
  # ----- SIDECAR MODULE -----
  - type: hdb
    path: gen/db
    parameters:
      buildpack: nodejs_buildpack
    requires:
      # 'hana' and 'xsuaa' resources extracted from CAP configuration
      - name: ws3_2-db
resources:
  # services extracted from CAP configuration
  # 'service-plan' can be configured via 'cds.requires.<name>.vcap.plan'
  # -----
  - name: ws3_2-db
  # -----

```

4. To check the changes made to the mta.yaml file that will be committed to your local git repository, click on the icon to go to the **Source Control** panel. Select the mta.yaml file under the list of changes.

The screenshot shows the SAP Business Application Studio interface with the 'mta.yaml' file open in the Source Control panel. The left pane displays the original code, and the right pane shows the updated code with changes highlighted in yellow boxes. The changes involve modifications to the 'path' and 'parameters' sections of the 'gen/srv' and 'gen/db' modules.

```

3 ## language=nodejs; multitenant=false
4 ## approuter=
5 # version: '3.1'
6 ID: ws3_2
7 version: 1.0.0
8 description: "A simple CAP project."
9 parameters:
10 enable-parallel-deployments: true
11
12 build-parameters:
13 before-all:
14   - builder: custom
15   commands:
16     - npm install --production
17     - npx -p @sap/cds-dk cds build --production
18
19 modules:
20 # ----- SERVER MODULE -----
21 # name: ws3_2-srv
22 # type: nodejs
23
24- path: gen/srv
25+ path: srv
26 parameters:
27   buildpack: nodejs_buildpack
28 requires:
29 # Resources extracted from CAP configuration
30   - name: ws3_2-db
31 provides:
32   - name: srv-api    # required by consumers of CAP services (e.g., i
33   properties:
34     - srv-url: ${default-url}
35
36 # ----- SIDECAR MODULE -----
37 # name: ws3_2-db-deployer
38 # type: hdb
39- path: gen/db
40+ path: db
41 parameters:
42   buildpack: nodejs_buildpack
43 requires:
44 # 'hana' and 'xsuaa' resources extracted from CAP configuration
45   - name: ws3_2-db

```

You can compare the changes made to the file by viewing both versions of the file side-by-side.

The code file appearing on the right side is the updated version while the one on the left is the older version.

Adjust the package.json file

Next, the package.json file of your project needs to be adjusted.

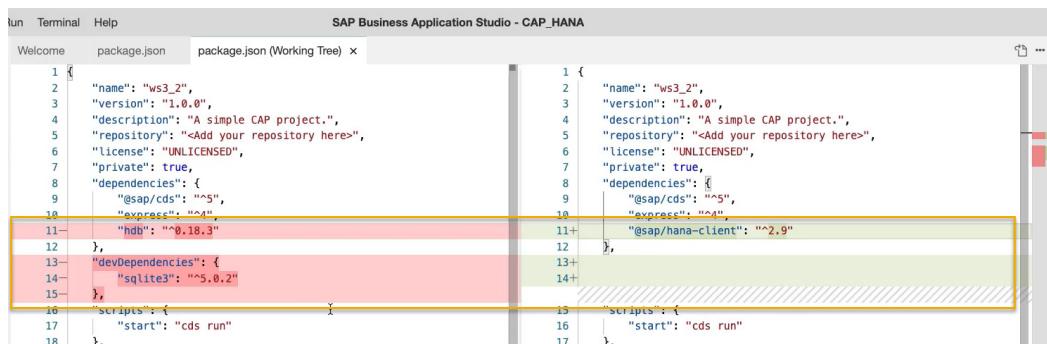
1. Click on the  **file explorer** icon to go back to the project folders.
2. From the list of files in the project, click on the **package.json** file.
3. Next, you need to edit the code present in the package.json file. Remove the following code from the file:

```
"hdb": "^0.18.3"
},
"devDependencies": {
  "sqlite3": "^5.0.2"
},
```

After removing the above code snippet, paste the following code in its place:

```
@sap/hana-client": "^2.9"
},
```

You can compare the changes made to the file by going to the Source Control panel. Select the **package.json** file under the section CHANGES. Here is how it should like:



```
1 [
2   "name": "ws3_2",
3   "version": "1.0.0",
4   "description": "A simple CAP project.",
5   "repository": "<Add your repository here>",
6   "license": "UNLICENSED",
7   "private": true,
8   "dependencies": {
9     "@sap/cds": "^5",
10    "express": "^4",
11-   "hdb": "^0.18.3"
12 },
13-  "devDependencies": {
14-    "sqlite3": "^5.0.2"
15 },
16  "scripts": {
17    "start": "cds run"
18 },
```

```
1 {
2   "name": "ws3_2",
3   "version": "1.0.0",
4   "description": "A simple CAP project.",
5   "repository": "<Add your repository here>",
6   "license": "UNLICENSED",
7   "private": true,
8   "dependencies": {
9     "@sap/cds": "^5",
10    "express": "^4",
11+   "@sap/hana-client": "^2.9"
12 },
13+  "scripts": {
14+    "start": "cds run"
15 },
```

4. Next, remove the following code from the file:

```
"cds": {
  "requires": {
    "db": {
      "kind": "sql"
    }
}
```

5. After removing the above code snippet, paste the following code in its place:

```
"cds": {  
  "build": {  
    "target": "."  
  },  
  "requires": {  
    "db": {  
      "kind": "hana"  
    }  
  }  
},  
"rules": {  
  "no-console": "off",  
  "require-atomic-updates": "off"  
},  
"cds": {  
  "requires": {  
    "db": {  
      "kind": "sql"  
    }  
  },  
  "hana": {  
    "deploy-format": "hdbtable"  
  }  
}
```

6. You can again compare the changes made to the file by going to the **Source Control** panel. Here is how it should like:

```
34  "cds": {  
35    "build": {  
36      "target": "."  
37    },  
38    "rules": {  
39      "no-console": "off",  
40      "require-atomic-updates": "off"  
41    },  
42    "cds": {  
43      "build": {  
44        "target": "."  
45      },  
46      "requires": {  
47        "db": {  
48          "kind": "hana"  
49        }  
50      },  
51      "hana": {  
52        "deploy-format": "hdbtable"  
53      }  
54    }  
55  }  
56 }  
57 }
```

7. In the **Source Control** panel, you can now commit these changes.
 8. Type **HANA prep** in the Message box under the Source Control panel.
 9. Click on the **tick mark** icon at the top of the panel to stage all your changes and commit them.
 10. If you encounter a message box about staging all changes and committing them directly, select **Yes**.

Now that the mta.yaml and package.json files are adjusted, we can continue by installing node modules that will help us generate the HDI objects we will need in our project.

Install Node Modules and HANA Client tool

1. Go back to the  **file explorer** and open a new terminal.
2. In the new terminal window, enter the following command and press **Enter**.

```
npm install
```

As a result of running the above command, you can see some new files created in your project folders. One of them will be a node_modules folder. You can view these files in the EXPLORER panel.

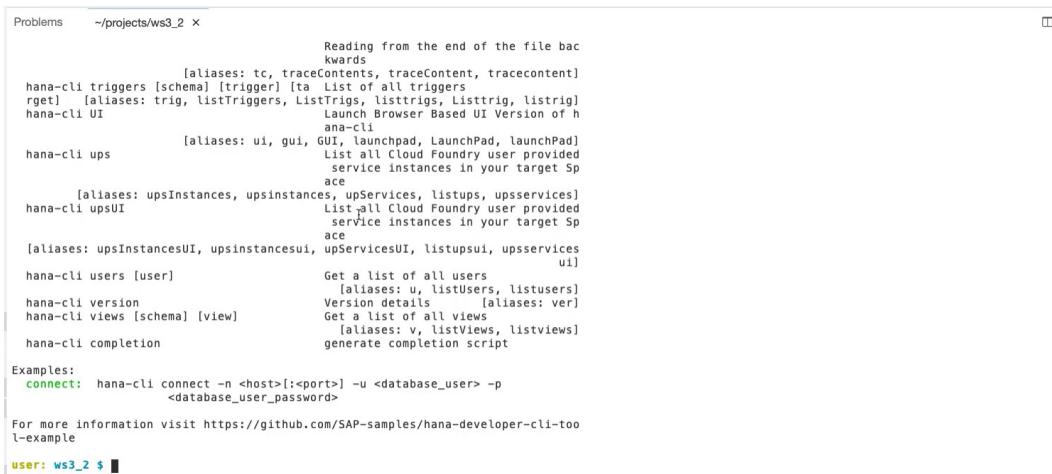
3. Go back to your terminal and enter the following command. Press Enter. This will install the hana-cli tool from the NPM library.

```
npm i -g hana-cli
```

4. Next, enter the following command and press **Enter**.

```
hana-cli
```

Running this command will show a help guide to what can be done using this hana-cli tool.



```
Problems ~/projects/ws3_2 ×
          Reading from the end of the file bac
          kwards
          [aliases: tc, traceContents, traceContent, tracecontent]
hana-cli triggers [schema] [trigger] [ta List of all triggers
rget] [aliases: trig, listTriggers, ListTrigs, listtrigs, Listtrig, listtrig]
hana-cli UI [aliases: ui, gui, launchpad, LaunchPad, launchPad]
hana-cli ups [aliases: upsInstances, upsinstances, upServices, listups, upsservices]
hana-cli upsUI [aliases: upsInstancesUI, upsinstancesui, upServicesUI, listupsui, upsservices
ui]
hana-cli users [user] [aliases: u, listUsers, listusers]
hana-cli version [aliases: ver]
hana-cli views [schema] [view] [aliases: v, listViews, listviews]
hana-cli completion [aliases: completionScript]
Examples:
  connect: hana-cli connect -n <host>[:<port>] -u <database_user> -p
<database_user_password>
For more information visit https://github.com/SAP-samples/hana-developer-cli-too
l-example
user: ws3_2 $
```

5. Next, enter the following command and press **Enter**

```
hana-cli createModule
```

As a result of running the above command, you can see some changes to the files present in the db folder of your project.

Create a package.json file in the srv folder

Next, you need to create a second package.json file in addition to the one on the root level of your project folder. This new package.json file needs to be created in the **srv** folder.

1. Right-click on the **srv** folder under your project in the  **file explorer** panel.
2. Select **New File**.
3. Name the file as **package.json**. Click **OK**.
4. Open this new package.json with the Code editor and copy and paste the following code:

```
{  
  "name": "ws3_2-srv",  
  "version": "1.0.0",  
  "dependencies": {  
    "@sap/cds": "^5",  
    "express": "^4",  
    "@sap/hana-client": "^2.9"  
  },  
  "scripts": {  
    "start": "cds serve gen/csn.json"  
  },  
  "cds": {  
    "requires": {  
      "db": {  
        "kind": "hana"  
      }  
    }  
  },  
  "engines": {  
    "node": "^14"  
  }  
}
```

And with that, all preparations in your CAP CDS project are done and you can start using the project to generate the table objects you need.

DEFINE AND CHANGE TABLES USING CAP CDS

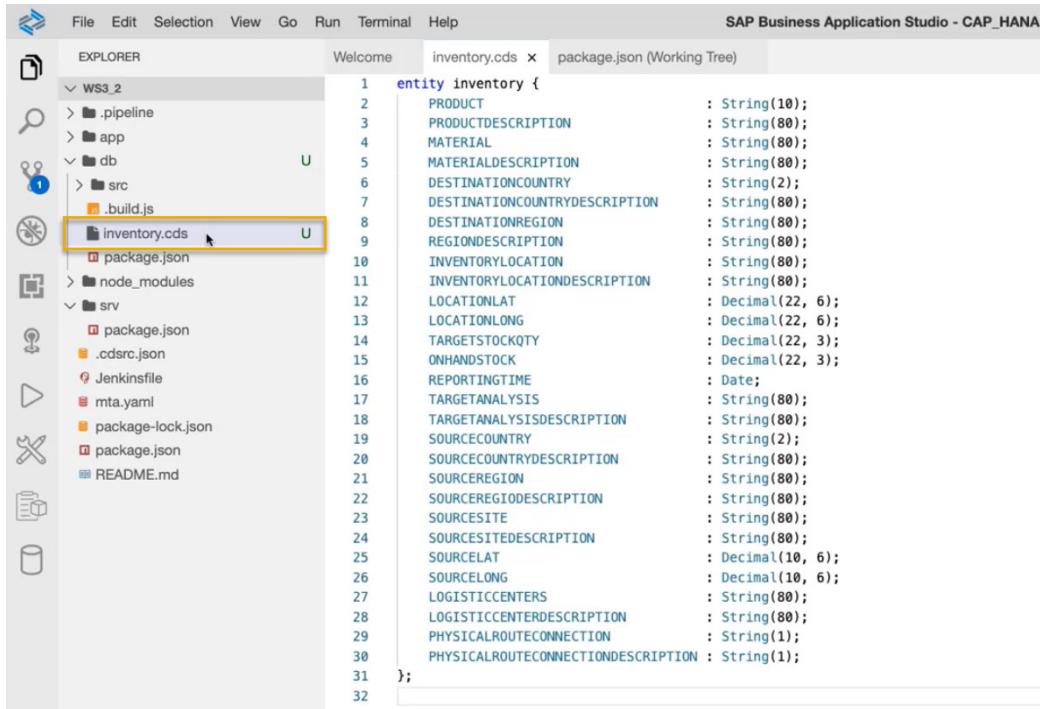
Create the **inventory.cds** file to generate your tables

We will create two files next that are needed for our hdbtable and hdbmigrationtable objects to be generated.

1. Right-click on the **db** folder from the  **file explorer** panel.
2. Select **New File**.
3. Name the file as **inventory.cds**. Click **OK**.
4. Once the file is created, select the **inventory.cds** file from the **src** folder.
5. Copy and paste the following code into the file:

```
entity inventory {
    PRODUCT : String(10);
    PRODUCTDESCRIPTION : String(80);
    MATERIAL : String(80);
    MATERIALDESCRIPTION : String(80);
    DESTINATIONCOUNTRY : String(2);
    DESTINATIONCOUNTRYDESCRIPTION : String(80);
    DESTINATIONREGION : String(80);
    REGIONDESCRIPTION : String(80);
    INVENTORYLOCATION : String(80);
    INVENTORYLOCATIONDESCRIPTION : String(80);
    LOCATIONLAT : Decimal(22, 6);
    LOCATIONLONG : Decimal(22, 6);
    TARGETSTOCKQTY : Decimal(22, 3);
    ONHANDSTOCK : Decimal(22, 3);
    REPORTINGTIME : Date;
    TARGETANALYSIS : String(80);
    TARGETANALYSISDESCRIPTION : String(80);
    SOURCECOUNTRY : String(2);
    SOURCECOUNTRYDESCRIPTION : String(80);
    SOURCEREGION : String(80);
    SOURCEREGIODESCRIPTION : String(80);
    SOURCESITE : String(80);
    SOURCESITEDESCRIPTION : String(80);
    SOURCELAT : Decimal(10, 6);
    SOURCELONG : Decimal(10, 6);
    LOGISTICCENTERS : String(80);
    LOGISTICCENTERDESCRIPTION : String(80);
    PHYSICALROUTECONNECTION : String(1);
    PHYSICALROUTECONNECTIONDESCRIPTION : String(1);
};
```

The file should look like this:



The screenshot shows the SAP Business Application Studio interface. The title bar reads "SAP Business Application Studio - CAP_HANA". The left sidebar is the "EXPLORER" panel, which lists various project files and folders. In the center, the "Welcome" tab is selected, showing the content of the "inventory.cds" file. The file contains CDS code defining an entity named "inventory" with numerous attributes and their data types. A yellow box highlights the "inventory.cds" file in the Explorer panel.

```

entity inventory {
  PRODUCT : String(10);
  PRODUCTDESCRIPTION : String(80);
  MATERIAL : String(80);
  MATERIALDESCRIPTION : String(80);
  DESTINATIONCOUNTRY : String(2);
  DESTINATIONCOUNTRYDESCRIPTION : String(80);
  DESTINATIONREGION : String(80);
  REGIONDESCRIPTION : String(80);
  INVENTORYLOCATION : String(80);
  INVENTORYLOCATIONDESCRIPTION : String(80);
  LOCATIONLAT : Decimal(22, 6);
  LOCATIONLONG : Decimal(22, 6);
  TARGETSTOCKQTY : Decimal(22, 3);
  ONHANDSTOCK : Decimal(22, 3);
  REPORTINGTIME : Date;
  TARGETANALYSIS : String(80);
  TARGETANALYSISDESCRIPTION : String(80);
  SOURCECOUNTRY : String(2);
  SOURCECOUNTRYDESCRIPTION : String(80);
  SOURCEREGION : String(80);
  SOURCEREGIODESCRIPTION : String(80);
  SOURCESITE : String(80);
  SOURCESITEDESCRIPTION : String(80);
  SOURCELAT : Decimal(10, 6);
  SOURCELONG : Decimal(10, 6);
  LOGISTICCENTERS : String(80);
  LOGISTICCENTERDESCRIPTION : String(80);
  PHYSICALROUTECONNECTION : String(1);
  PHYSICALROUTECONNECTIONDESCRIPTION : String(1);
};


```

Note: This file will be used to generate an hdbtable object with the same column definitions as in session 1. The cds compiler will later generate the hdbtable with columns defined as NVARCHAR data type although these columns are defined as STRING in the cds file.

6. Next, right-click on the **srv** folder from the  **file explorer** panel.
7. Select **New File**.
8. Name the file as **inventory_srv.cds** and click on **OK**.
9. Once the file is created, select the file to open the code editor.
10. Copy and paste the following code into the file

```

using inventory from '../db/inventory';

service CatalogService {

  entity inventory_srv as projection on inventory;
}

```

The file should look like this:

```

File Edit Selection View Go Run Terminal Help
SAP Business Application Studio - CAP_HANA

EXPLORER
WELCOME inventory.cds inventory_svr.cds x package.json (Working Tree)
1 using inventory from './db/inventory';
2
3 service CatalogService {
4     entity inventory_svr as projection on inventory;
5 }
6

1. pipeline
2. app
3. db
4. src
    .build.js
    inventory.cds
    package.json
5. node_modules
6. srv
    inventory_svr.cds
    package.json
    .cdsrc.json
    Jenkinsfile
    mta.yaml
    package-lock.json
    package.json
    README.md

```

Generate your table with cds build

Now that all necessary files are created for the cds generator to work, we can start building and deploying the first object. Like in session 1, we will first create our inventory table as an hdbtable object.

1. Go to the open terminal or open a new one. Enter the following command and press **Enter**.

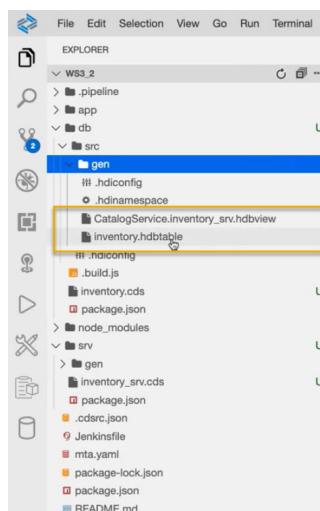
```
cds
```

The result will show you the available commands under cds option.

2. Next, enter the following command and press **Enter**.

```
cds build
```

As a result of running the above command, you can see some new files were generated in the src folder of your project. These include the **inventory.hdbtable** file and the **CatalogService.inventory_svr.hdbview** file in the gen folder.

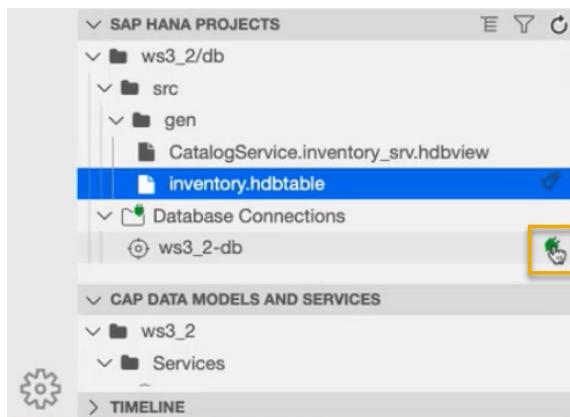


3. Next, enter the following command and press **Enter**.

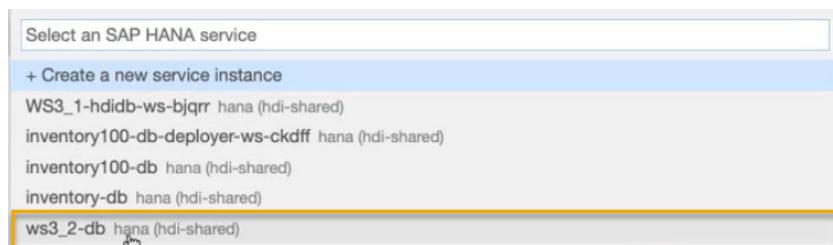
```
cds deploy
```

After completing the deployment, the next step is to proceed to your database.

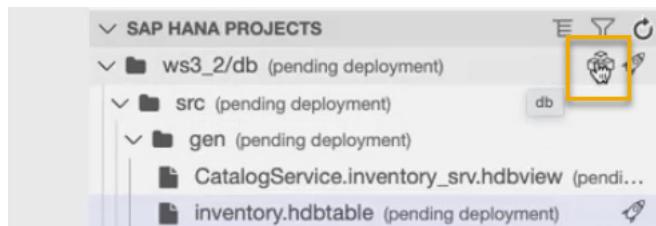
4. Before going to your database, you must establish the connection to Cloud Foundry. To do that, click on the  **bind icon** (green) for your project **ws3_2-db** under Database Connections in the SAP HANA PROJECTS panel.



5. You can now select the Cloud Foundry service instances available from the search prompt along the menu bar. Select your service instance **ws3_2-db**.



6. If you encounter a message box asking if you would like to add 'node_modules' to .gitignore, click **Yes**. This will make sure that all the node modules you have installed in the previous step are not pushed to your git repository.
7. Now, you can open your HDI container from SAP HANA PROJECTS panel. Click on the  icon next to your project **ws3_2/db** to open the HDI container in the SAP HANA Database Explorer.



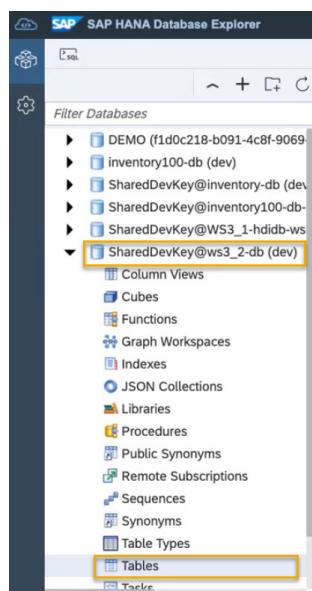
Note: This first deployment of the project to your database will synchronize the CAP tooling with the SAP HANA tooling in your project.

Add data to the Inventory table

Now that the cds compiler has generated the hdbtable object and you deployed it to the database, we can go ahead and add data to this table.

Important: You will use the same sample data that was used in the first session. If you did not attend the first session, you can download the sample data from [here](#). Select the **Download** option for the **Sample Data.zip** file under the folder location SAP-samples/hana-cloud-learning/Workshop: Collaborative Development. You need to **unzip** the Sample Data.zip file before it can be used to upload into the table.

1. Select your HDI container from the list of database connections on the left.
2. In the catalog of the HDI container, click on **Tables**.



3. Click on the table **INVENTORY** on the bottom left. You can view the columns present in the table on a new window.
4. Next, right-click on the **INVENTORY** table and select **Import Data**.
5. A wizard opens where you can select **Import Data** as IMPORT TYPE. Click on **Step 2**.
6. Under IMPORT SOURCE, select **Local** to specify where the data is imported from. Click on **Browse** to the find the unzipped **.csv** file on your local machine. Click on the .csv file and select **open**. Click on **Step 3**.
7. Under IMPORTTARGET, name the Table as **INVENTORY** unless it's not named so by default. Click on **Step 4**.
8. Under TABLE MAPPING, verify that the column properties match with the column definition that you had used to create the inventory.hdbtable. If you have not made any changes yourself, all should be good to go as is. Click on **Step 5**.
9. Under ERROR HANDLING, select the option **Save all successful rows and list the errors (if any)**. Click on **Review**.
10. After reviewing the Import Summary, click **Import Into Database**. Wait until the Import Status shows the **Import complete** notification.
11. To verify the data import, click on the **INVENTORY** table from the left side panel. Select **Open Data** to view the data added into the table.

Sidenote: Creating graphs in the SAP HANA Database Explorer

- Now that your data is imported successfully, you can preview it using the Analysis functions of the SAP HANA Database Explorer.
- In the Open Data window, click on Analysis to view the data in the form of graphs.
- For example, you can drag PRODUCT from the list of Available Columns to the Value Axis to see how many product values are present.
- You can select different graphical options provided in a main toolbar at the top or access the corresponding SQL code to these graphical views.

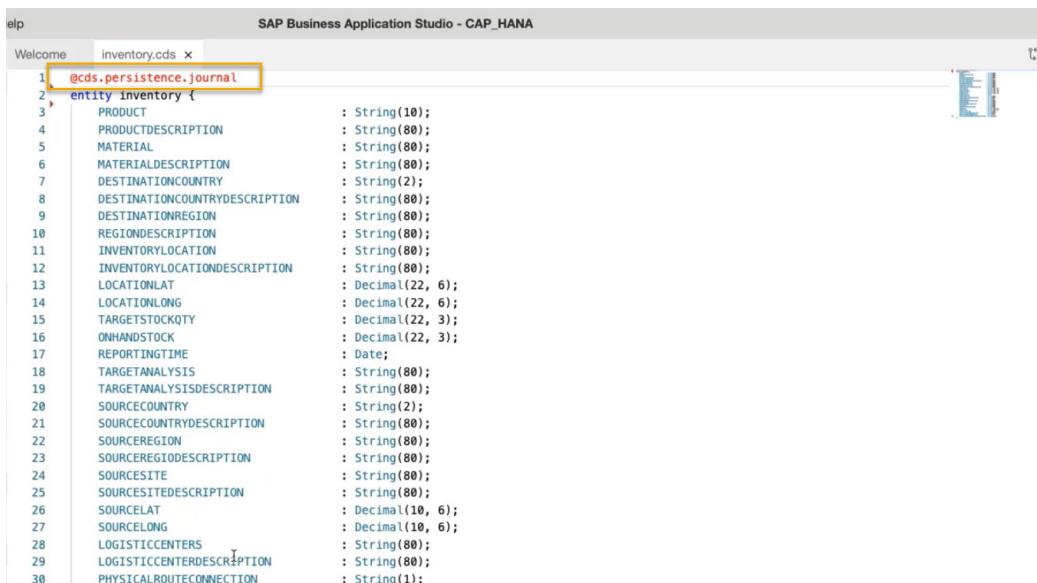
Generate an hdbmigrationtable

Now that your table is created and the data is imported, we will change the structure of this table by adding two columns and having it generated as an hdbmigrationtable object instead of an hdbtable object. You can then examine in the terminal how this type of object handles data structure changes, which we will compare to other table types later.

- Go back to SAP Business Application Studio.
- Select the **inventory.cds** file from the src folder. Add the following statement to the beginning of the file:

```
@cds.persistence.journal
```

This will influence the compiler to change the inventory.hdbtable table into inventory.hdbmigrationtable when you build the cds file. The cds file should like this before you deploy it:



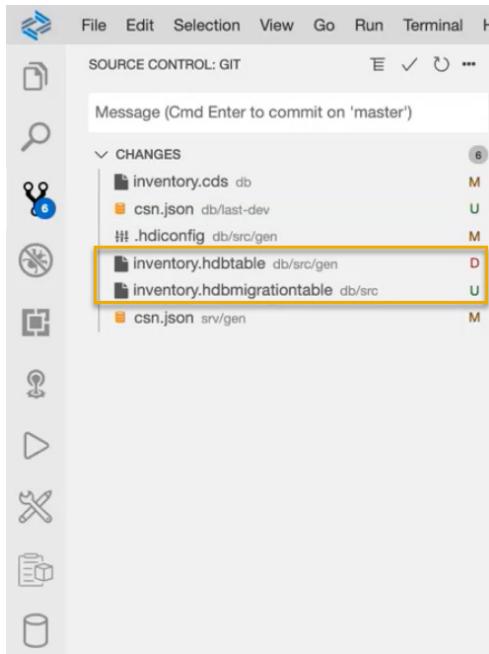
```
elp SAP Business Application Studio - CAP_HANA
Welcome inventory.cds x
1 @cds.persistence.journal
2 entity inventory {
3   PRODUCT : String(10);
4   PRODUCTDESCRIPTION : String(80);
5   MATERIAL : String(80);
6   MATERIALDESCRIPTION : String(80);
7   DESTINATIONCOUNTRY : String(2);
8   DESTINATIONCOUNTRYDESCRIPTION : String(80);
9   DESTINATIONREGION : String(80);
10  REGIONDESCRIPTION : String(80);
11  INVENTORYLOCATION : String(80);
12  INVENTORYLOCATIONDESCRIPTION : String(80);
13  LOCATIONLAT : Decimal(22, 6);
14  LOCATIONLONG : Decimal(22, 6);
15  TARGETSTOCKQTY : Decimal(22, 3);
16  ONHANDSTOCK : Decimal(22, 3);
17  REPORTINGTIME : Date;
18  TARGETANALYSIS : String(80);
19  TARGETANALYSISDESCRIPTION : String(80);
20  SOURCECOUNTRY : String(2);
21  SOURCECOUNTRYDESCRIPTION : String(80);
22  SOURCEREGION : String(80);
23  SOURCEREGIODESCRIPTION : String(80);
24  SOURCESITE : String(80);
25  SOURCESITEDESCRIPTION : String(80);
26  SOURCELAT : Decimal(10, 6);
27  SOURCELONG : Decimal(10, 6);
28  LOGISTICCENTERS : String(80);
29  LOGISTICCENTERDESCRIPTION : String(80);
30  PHYSICALROUTECONNECTION : String(1);
```

- Go to the open terminal or open a new one. Enter the following command and press **Enter** to execute it.

```
cds build
```

In the EXPLORER panel, you will notice that the inventory.hdbtable is replaced by an inventory.hdbmigrationtable file.

4. You can also review this change in the  **Source Control** panel.



5. Go back to the EXPLORER panel. Click on the  deploy icon next to the project **WS3_2/db** folder under SAP HANA PROJECTS panel.
6. Now, you can open your HDI container from SAP HANA PROJECTS panel. Click on  icon for project **WS3_2/db** to open the HDI container in SAP HANA Database Explorer.
7. Click on the **Refresh** icon in the SAP HANA Database Explorer.
8. Select the **INVENTORY** table. When you now view the table and open the data, the definitions and the number of rows should remain the same as before. (You can repeat the steps given in the [previous section on creating graphs](#) to double check)

Migration to Version 2

1. Go back to SAP Business Application Studio.
2. Select the **inventory.cds** file from the src folder. Add two columns to the table definition by copying the following code and pasting it at the end of the line 31:

```
ADDITIONALATTRIBUTESTRING      : String(10);
ADDITIONALATTRIBUTESTRING_NN   : String(10) NOT NULL DEFAULT 'TEST';
```

Make sure to paste these lines in between the last column definitions and the closing parenthesis '};' in line 32.

The new table definition should look like in this image:

```

entity inventory {
    PRODUCT : String(10);
    PRODUCTDESCRIPTION : String(80);
    MATERIAL : String(80);
    MATERIALDESCRIPTION : String(80);
    DESTINATIONCOUNTRY : String(2);
    DESTINATIONCOUNTRYDESCRIPTION : String(80);
    DESTINATIONREGION : String(80);
    REGIONDESCRIPTION : String(80);
    INVENTORYLOCATION : String(80);
    INVENTORYLOCATIONDESCRIPTION : String(80);
    LOCATIONLAT : Decimal(22, 6);
    LOCATIONLONG : Decimal(22, 6);
    TARGETSTOCKQTY : Decimal(22, 3);
    ONHANDSTOCK : Decimal(22, 3);
    REPORTINGTIME : Date;
    TARGETANALYSIS : String(80);
    TARGETANALYSISDESCRIPTION : String(80);
    SOURCECOUNTRY : String(2);
    SOURCECOUNTRYDESCRIPTION : String(80);
    SOURCEREGION : String(80);
    SOURCEREGIONDESCRIPTION : String(80);
    SOURCESITE : String(80);
    SOURCESITEDESCRIPTION : String(80);
    SOURCELAT : Decimal(10, 6);
    SOURCELONG : Decimal(10, 6);
    LOGISTICCENTERS : String(80);
    LOGISTICCENTERDESCRIPTION : String(80);
    PHYSICALROUTECONNECTION : String(1);
    PHYSICALROUTECONNECTIONDESCRIPTION : String(1);
    ADDITIONALATTRIBUTESTRING : String(10);
    ADDITIONALATTRIBUTESTRING_NN : String(10) not null default 'TEST';
};

```

3. Go to the open terminal or open a new one. Enter the following command and press **Enter**.

`cds build`

4. Click on the **inventory.hdbmigrationtable** file from the EXPLORER panel. You will notice that the file version has been updated to version 2.

```

== version=2
-- generated by cds-compiler version 2.5.2
COLUMN TABLE inventory (
    "PRODUCT" NVARCHAR(10),
    PRODUCTDESCRIPTION NVARCHAR(80),
    MATERIAL NVARCHAR(80),
    MATERIALDESCRIPTION NVARCHAR(80),
    DESTINATIONCOUNTRY NVARCHAR(2),
    DESTINATIONCOUNTRYDESCRIPTION NVARCHAR(80),
    DESTINATIONREGION NVARCHAR(80),
    REGIONDESCRIPTION NVARCHAR(80),
    INVENTORYLOCATION NVARCHAR(80),
    INVENTORYLOCATIONDESCRIPTION NVARCHAR(80),
    LOCATIONLAT DECIMAL(22, 6),
    LOCATIONLONG DECIMAL(22, 6),
    TARGETSTOCKQTY DECIMAL(22, 3),
    ONHANDSTOCK DECIMAL(22, 3),
    REPORTINGTIME DATE,
)

```

You can also see the codes for the migration steps were added by the cds-compiler at the end of the file as shown below:

```

29 LOGISTICCENTERS NVARCHAR(80),
30 LOGISTICCENTERDESCRIPTION NVARCHAR(80),
31 PHYSICALROUTECONNECTION NVARCHAR(1),
32 PHYSICALROUTECONNECTIONDESCRIPTION NVARCHAR(1),
33 ADDITIONALATTRIBUTESTRING NVARCHAR(10),
34 ADDITIONALATTRIBUTESTRING_NN NVARCHAR(10) NOT NULL DEFAULT 'TEST'
35 )
36
37 == migration=2
38 -- generated by cds-compiler version 2.5.2
39 ALTER TABLE inventory ADD (ADDITIONALATTRIBUTESTRING NVARCHAR(10), ADDITIONALATTRIBUTESTRING_NN NVARCHAR(10) NOT NULL DEFAULT '

```

Note: The ALTER TABLE commands are used to migrate the table from version 1 to version 2. If the version 1 has not yet been deployed, then version 2 is run directly without running the migration steps.

5. Click on the deploy icon next to the project **WS3_2/db** folder under SAP HANA PROJECTS panel.
6. Now, you can open your HDI container from SAP HANA PROJECTS panel. Click on icon for project **WS3_2/db** to open the HDI container in SAP HANA Database Explorer.
7. Click on the **Refresh** icon in the SAP HANA Database Explorer.
8. Select the **INVENTORY** table. In the table definitions, you should now see that two additional columns have been added to the table and opening the data should not result in any changes to the number of rows in the previous graph.

INVENTORY						
Table Name			Schema			
INVENTORY			163E7A0663DA437792D5DB05B8817CF3			
Columns	Indexes	Properties	Runtime Information			
15 REPORTINGTIME			SQL Data Type	Key	Not Null	Default
16 TARGETANALYSIS			NVARCHAR(80)			NULL
17 TARGETANALYSISDESCRIPTION			NVARCHAR(80)			NULL
18 SOURCECOUNTRY			NVARCHAR(2)			NULL
19 SOURCECOUNTRYDESCRIPTION			NVARCHAR(80)			NULL
20 SOURCEREGION			NVARCHAR(80)			NULL
21 SOURCEREGIODESCRIPTION			NVARCHAR(80)			NULL
22 SOURCESITE			NVARCHAR(80)			NULL
23 SOURCESITEDESCRIPTION			NVARCHAR(80)			NULL
24 SOURCELAT			DECIMAL(10,6)			NULL
25 SOURCELONG			DECIMAL(10,6)			NULL
26 LOGISTICCENTERS			NVARCHAR(80)			NULL
27 LOGISTICCENTERDESCRIPTION			NVARCHAR(80)			NULL
28 PHYSICALROUTECONNECTION			NVARCHAR(1)			NULL
29 PHYSICALROUTECONNECTIONDESCR			NVARCHAR(1)			NULL
30 ADDITIONALATTRIBUTESTRING			NVARCHAR(10)			NULL
31 ADDITIONALATTRIBUTESTRING_NN			NVARCHAR(10)	X		TEST

9. You can verify that no changes happened in the data by repeating the steps in the [section on creating graphs](#).

Migration to Version 3

1. Go back to SAP Business Application Studio.
2. Select the **inventory.cds** file from the src folder. Add two columns to the table definition by copying the following code and pasting it at the end of the line 33:

```
ADDINTEGER : Integer;
ADDINTEGER_NN : Integer NOT NULL DEFAULT 0;
```

Make sure to paste these lines in between the last column definitions and the closing parenthesis '};'.

```
elip SAP Business Application Studio - CAP_HANA
Welcome inventory.cds
29 LOGISTICCENTERDESCRIPTION : String(80);
30 PHYSICALROUTECONNECTION : String(1);
31 PHYSICALROUTECONNECTIONDESCRIPTION : String(1);
32 ADDITIONALATTRIBUTESTRING : String(10);
33 ADDITIONALATTRIBUTESTRING_NN : String(10) not null default 'TEST';
34 ADDINTEGER : Integer;
35 ADDINTEGER_NN : Integer NOT NULL DEFAULT 0;
36 };
37
```

3. Go to the open terminal or open a new one in SAP BAS environment. Enter the following command and press Enter.
- ```
cds build
```
4. Click on the **inventory.hdbmigrationtable** file from the EXPLORER panel. You will notice that the file version has been updated to version 3.

```
elip SAP Business Application Studio - CAP_HANA
Welcome inventory.cds inventory.hdbmigrationtable
1 == version=3
2 -- generated by cds-compiler version 2.5.2
3 COLUMN TABLE inventory (
4 "PRODUCT" NVARCHAR(10),
5 PRODUCTDESCRIPTION NVARCHAR(80),
6 MATERIAL NVARCHAR(80),
7 MATERIALDESCRIPTION NVARCHAR(80),
8 DESTINATIONCOUNTRY NVARCHAR(2),
9 DESTINATIONCOUNTRYDESCRIPTION NVARCHAR(80),
10 DESTINATIONREGION NVARCHAR(80),
11 REGIONDESCRIPTION NVARCHAR(80),
12 INVENTORYLOCATION NVARCHAR(80),
13 INVENTORYLOCATIONDESCRIPTION NVARCHAR(80),
14 LOCATIONLAT DECIMAL(22, 6),
15 LOCATIONLONG DECIMAL(22, 6),
16 TARGETSTOCKQTY DECIMAL(22, 3),
17 ONHANDSTOCK DECIMAL(22, 3),
18 REPORTINGTIME DATE,
```

You can also see the codes for migration added at the end of the file by the cds-compiler as shown below:

```
elip SAP Business Application Studio - CAP_HANA
Welcome inventory.cds inventory.hdbmigrationtable
34 AUDITUNALIATTRIBUTESKIN_NN NVARCHAR(10) NOT NULL DEFAULT 'TEST',
35 ADDINTEGER INTEGER,
36 ADDINTEGER_NN INTEGER NOT NULL DEFAULT 0
37)
38
39 == migration=3
40 -- generated by cds-compiler version 2.5.2
41 ALTER TABLE inventory ADD (ADDINTEGER INTEGER, ADDINTEGER_NN INTEGER NOT NULL DEFAULT 0);
42
43 == migration=2
44 -- generated by cds-compiler version 2.5.2
45 ALTER TABLE inventory ADD (ADDITIONALATTRIBUTESTRING NVARCHAR(10), ADDITIONALATTRIBUTESTRING_NN NVARCHAR(10) NOT NULL DEFAULT '
```

**Note:** Deploying the project will ensure that all other versions of the table migrate to the version 3.

5. Click on the  deploy icon next to the project **WS3\_2/db** folder under SAP HANA PROJECTS panel.
6. You can again go to the SAP HANA Database Explorer to verify the columns have been successfully added to INVENTORY table definition and no changes in the data have happened.

And with that, our data structure changes are complete.

You have seen how the cds-compiler in a CAP project can help you easily manage the migration of different table versions.

After the initial set up, all you need to do is make the necessary changes to your cds file and the compiler will handle the creation of objects automatically.

# SET UP CI/CD CHAIN

Now that we have seen how to automate the generation of database objects, we want to also automate the deployment process of our project. So far, whenever we made changes to the cds-file and had new or updated objects generated, we had to manually  deploy these changes to our database. In the second part of this session, we will create an automated deployment chain using the SAP Continuous Integration and Delivery (CI/CD) service, which will automatically deploy to the database whenever changes are made and pushed to a connected git repository.

In the first step, we will show you how to set up CI/CD to create a job that you can trigger manually. After that, we want our deployment with CI/CD to be triggered automatically via a webhook from a remote git repository. Whenever changes are pushed to the remote repository, this webhook will trigger the CI/CD chain to start a deployment to the database.

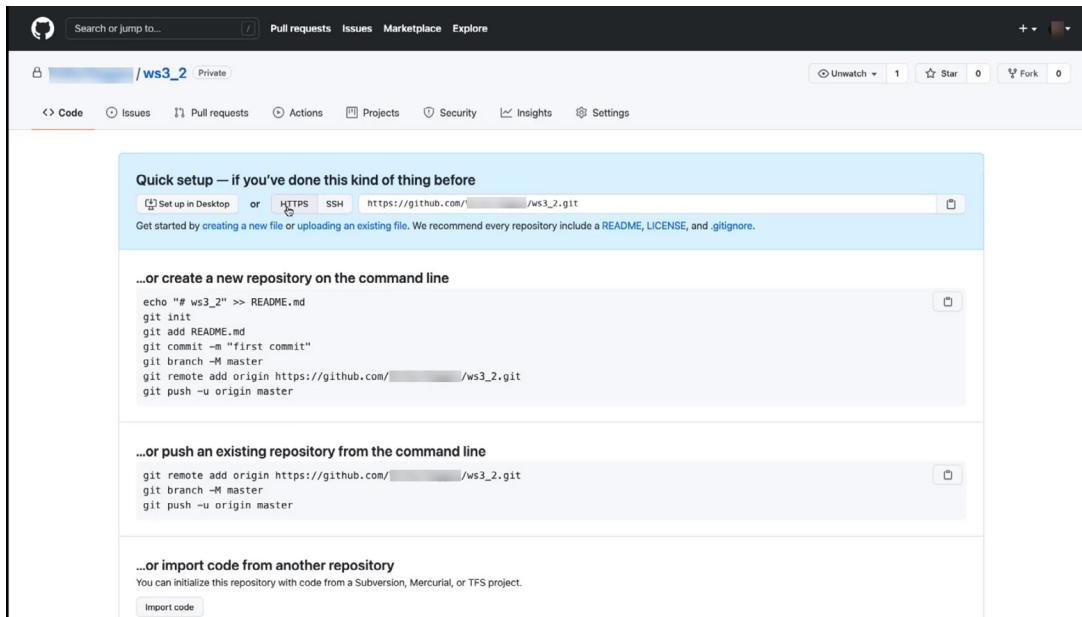
## Create a new GitHub repository

The first thing you need to do is establish a connection to a remote git repository.

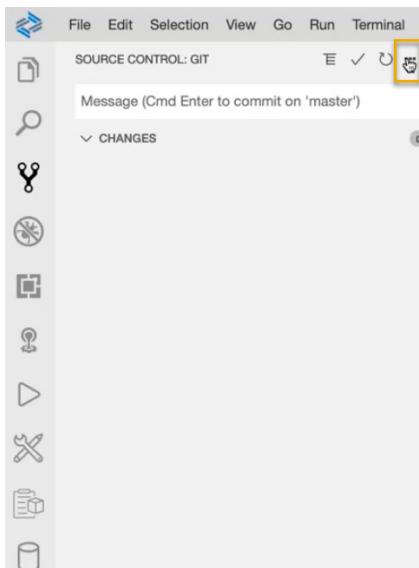
1. Open [github.com](https://github.com) on a new browser tab. Enter your GitHub account using your credentials.
2. On the top right corner of the screen, click on the profile icon in the top right corner and select **Your repositories**.
3. In the new window that opens, click on **New** the top right corner to create a new repository.
4. Under Create a new repository, enter **WS3\_2** as the Repository name.
5. Optionally, you can add a short description, for example: Workshop with CICD.
6. You can choose to make your repository visible publicly or privately. We suggest you select the option **Private**.
7. Click on **Create repository** to finish.

## Connect the GitHub repository to your CAP project

1. Once your repository is created, you will see the **Code** view of this repository, which contains the **Quick setup** section. At the top, make sure the field **HTTPS** is selected.
2. Copy the HTTPS connection link for the new repository and save it in a safe location locally.



3. Go back to SAP Business Application Studio.
4. On the left menu, click on the  icon to go to the **Source Control** panel.
5. If there are any changes that have not been pushed, you need to commit, stage and push them before you continue.
6. Click on the **three dots** icon on the top right corner of the Source Control panel on the left.



7. Select **Remote** and then **Add Remote**.
8. A prompt appears at the top of the screen requesting the repository URL. Paste the HTTPS connection link for your WS3\_3 GitHub repository in the prompt. Press **Enter**.
9. Name the remote **WS3\_2** and press **Enter**.

**Note:** If SAP Business Application Studio does not have your GitHub credentials cached from previous activities, you might need to enter your credentials at this point in the prompt before continuing.

10. Click on the **three dots** icon next to the Source Control panel.
11. Select the **Push/Pull** option. Choose **Sync**.
12. Click **OK**, if you encounter a message box asking whether you want to publish.
13. Click on the **three dots** icon next to the Source Control panel.
14. Select the **Push/Pull** option. Choose **Push**.
15. You can go back to your GitHub repository and verify that all the folders and files of your development project are now present in it.

## Create a GitHub Token

For the CI/CD service to authenticate your access rights to the repository and the webhook, you need to use a token, which we will show you how to create next. The token needs to have repo, workflow, notifications, and - most importantly - admin:repo\_hook access rights.

1. Go to your GitHub account. On the top right corner of the screen, click on your **GitHub profile icon** and choose **Settings**.
2. Scroll down the Settings page to find **Developer settings** on the left side of the screen and select it.
3. In Developer settings, select **Personal access tokens**.
4. Click on **Generate new token** at the top right corner of the screen. You will be asked to sign in using your GitHub account password.
5. After signing in, **enter a name for the token** (for example: HC\_WS3) under the caption Note.
6. Next you can select an expiration date.
7. Under **Select scopes**, select **repo, workflow, admin:repo\_hook** and **notifications**.
8. Click on **Generate token** to complete generation.
9. Copy the new token you generated and paste it in a safe location that could be accessed later.

**Note:** After the creation of your token has finished, you will not see it again. So, make sure to save it somewhere safe before continuing. Otherwise, you need to create a new token.

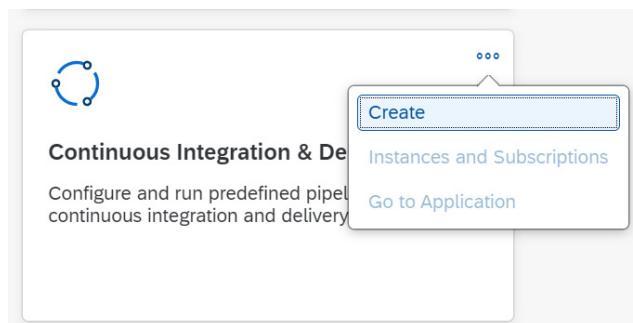
## Subscribe to CI/CD and manage CI/CD Administrator rights

Now that all preparations with GitHub are complete, the next step is to prepare to access the CI/CD service from the SAP BTP cockpit.

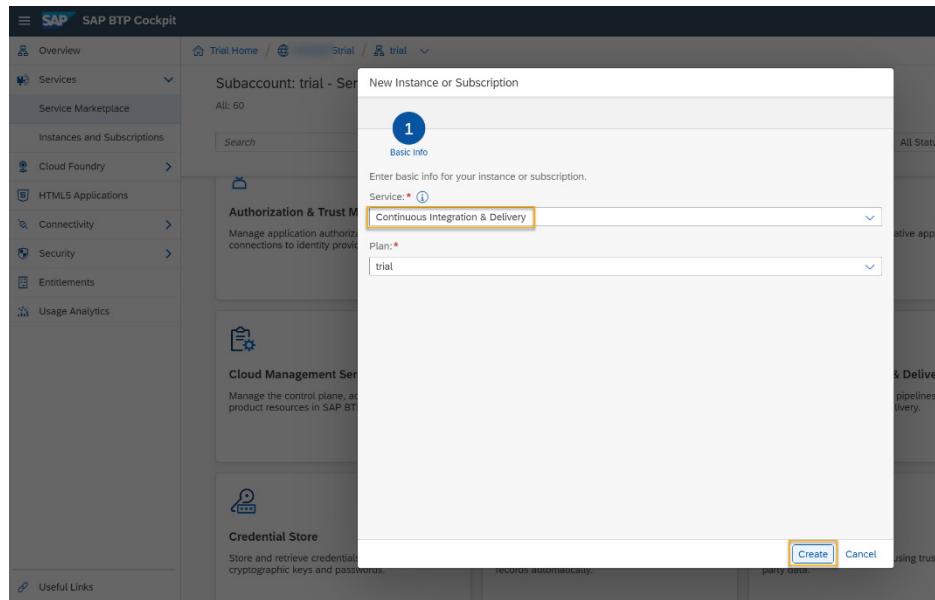
1. Open your trial account in SAP BTP cockpit.
2. In SAP BTP Cockpit, navigate to your **Subaccount** and within the subaccount, select the option **Instances and Subscriptions** from the left menu bar.
3. If you have not yet subscribed to the **Continuous Integration and Delivery** service, you need to select the option **Service Marketplace** from the left menu bar.
4. Enter **Continuous Integration and Delivery** in the search bar or scroll down to find it.

The screenshot shows the SAP BTP Cockpit interface. On the left, the navigation bar includes 'Overview', 'Services' (selected), 'Service Marketplace' (highlighted in blue), 'Instances and Subscriptions', 'Cloud Foundry', 'HTML5 Applications', 'Connectivity', 'Security', 'Entitlements', and 'Usage Analytics'. The main area is titled 'Subaccount: - Service Marketplace' and shows a search bar with filters for 'All Types', 'All Environments', 'All Capabilities', and 'All Statuses'. Below the search bar, there are two sections: 'Extension Suite - Development Efficiency' and 'Integration Suite'. Under 'Extension Suite', the 'Continuous Integration & Delivery' service is highlighted with a yellow box. It has a circular icon with a gear and a checkmark, and the text 'Configure and run predefined pipelines for continuous integration and delivery.' Other services listed include 'Cloud Management Service' and 'SAP HANA Schemas & HDI Containers'. Under 'Integration Suite', there is a service for 'SAP HANA Schemas & HDI Containers T...' and 'Content Agent Service'. At the bottom left, there are links for 'Useful Links' and 'Legal Information'.

5. On the Continuous Integration & Delivery tile, click on the **three dots** and select **Create** to add a subscription.



- Click on **Create** in the wizard that opens.



**Note:** You will now find CI/CD in the list of Instances and Subscriptions in your subaccount. Before you can access it, however, you still need to assign yourself a role collection that grants you the rights to do so.

- Select **Role Collections** from the left menu bar.
- Click on **CICD Service Administrator** from the list of Role Collections.

The screenshot shows the SAP BTP Cockpit interface with the 'Role Collections' section selected in the sidebar (highlighted by a yellow box). The main area displays a table of role collections. One row, 'CICD Service Administrator', is specifically highlighted with a yellow box. The table columns include Name, Description, Roles, User Groups, and Actions. Other visible rows include 'Business\_Application\_Studio\_Administrator', 'Business\_Application\_Studio\_Developer', 'Business\_Application\_Studio\_Extension\_Deployer', 'CICD Service Developer', and 'Cloud Connector Administrator'.

| Name                                           | Description                                                                               | Roles                    | User Groups | Actions                                       |
|------------------------------------------------|-------------------------------------------------------------------------------------------|--------------------------|-------------|-----------------------------------------------|
| Business_Application_Studio_Administrator      | Allows administrators to manage (export and delete) user data.                            | Administrator            |             | <button>Copy</button> <button>Delete</button> |
| Business_Application_Studio_Developer          | Allows developers to load and develop applications using SAP Business Application Studio. | Developer                |             | <button>Copy</button> <button>Delete</button> |
| Business_Application_Studio_Extension_Deployer | Allows extension developers to deploy simple extensions.                                  | Extension deployer       |             | <button>Copy</button> <button>Delete</button> |
| <b>CICD Service Administrator</b>              | Administrator Role Collection for the Continuous Integration & Delivery Service           | Administrator            |             | <button>Copy</button> <button>Delete</button> |
| CICD Service Developer                         | Developer Role Collection for the Continuous Integration & Delivery Service               | Developer                |             | <button>Copy</button> <button>Delete</button> |
| Cloud Connector Administrator                  | Operate the data transmission tunnels used by the Cloud Connector.                        | Cloud Connector Admin... |             | <button>Copy</button> <button>Delete</button> |

- In the CICD Service Administrator - Overview, you can see the role collection Administrator at the top section and all users that are assigned this role collection. If you use CI/CD for the first time, your user will not be listed here yet.

- Click on **Edit** at the top right corner.

The screenshot shows the SAP BTP Cockpit interface. The title bar says "SAP BTP Cockpit". The main content area is titled "Role Collection: CICD Service Administrator - Overview". Below it, a description states "Administrator Role Collection for the Continuous Integration & Delivery Service". There are two tabs: "Roles (1)" and "Users (2)". The "Roles (1)" tab shows one role named "Administrator" with a role template of "Administrator" and an application identifier of "cicdservice1b38579". The "Users (2)" tab shows two users with IDs "00000000000000000000000000000000" and "00000000000000000000000000000000", both associated with the "Default identity provider".

- Enter the **user ID for your trial account** and click on **Save**. (SAP employees: for internal reasons, you need to use the credentials of a technical p-user in this step and subsequent authentications in CI/CD. If you do not have a p-user, you may proceed with your trial account, but some functionalities of the CI/CD job will be limited.)
- With that, the preparations for accessing CI/CD are complete and you can open the service.
- From the Role Collections view, navigate back to your **subaccount**.
- Go to **Instances and Subscriptions** on the left menu bar.
- Click on the application **Continuous Integration and Delivery**.

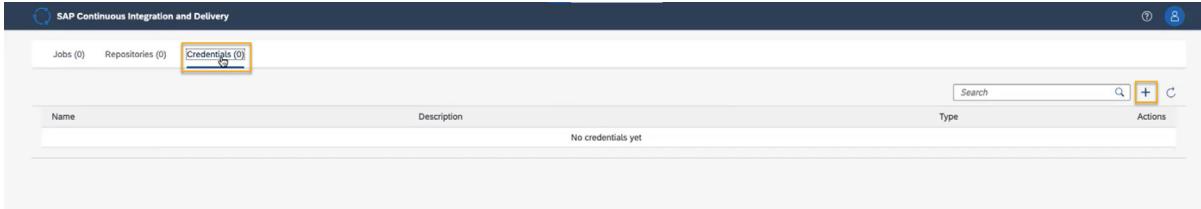
The screenshot shows the SAP BTP Cockpit interface with the left sidebar expanded. The "Instances and Subscriptions" section is selected. The main content area is titled "Subaccount - Instances and Subscriptions". It includes a search bar and filters for "All Services", "All Plans", and "All Statuses". Below this, there are three tabs: "Subscriptions (4)", "Instances (9)", and "Environments (2)". The "Subscriptions" tab is active, showing a table with columns "Application", "Plan", "Created On", "Changed On", and "Status". One row in the table is highlighted with a green border, showing the application "Continuous Integration & Delivery" and a status of "Subscribed".

- The SAP Continuous Integration and Delivery service will open in a new browser tab.

## CI/CD: Add credentials

For CI/CD to interact with your trial environment and GitHub repository, you can have the necessary credentials stored and defined directly in this service. First, you first need to add two credentials manually: Your cloud foundry credentials and your GitHub credentials.

1. Click on the **Credentials** tab.
2. Click on the **plus icon** to add a new credential.



3. In the dialogue box, enter **cfdeploy** as the Name of the credential.
4. In the field **Username** enter the email address associated to your trial account and the corresponding password in the next field. (SAP employees: enter p-user credentials)

 A screenshot of the 'Create Credentials' dialog box. It has fields for 'Name' (containing 'cfdeploy'), 'Description', 'Type' (set to 'Basic Authentication'), 'Username', and 'Password'. There are 'Create' and 'Discard' buttons at the bottom.

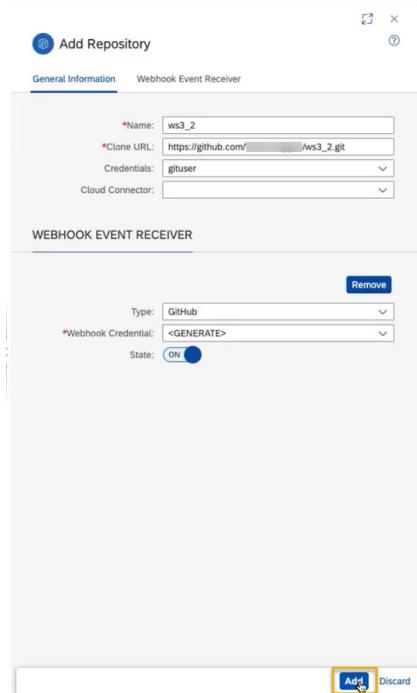
5. Click on **Create**.
6. Next, click on the plus icon again to create a second credential.
7. In the dialogue box, type **gituser** as the Name of the credential.
8. In the field Username enter the email address associated to your GitHub account and the corresponding password in the next field.
9. Click on **Create**.

Now your cloud foundry and GitHub credentials are stored in CI/CD and can be used when defining repositories and creating jobs.

## CI/CD: Define the GitHub repository

Before we can create the CI/CD job that will deploy changes to your database, we need to define the GitHub repository where the changes are found that should be deployed to the database.

1. Click on **Repositories** from the options given at the top. Click on **plus icon** to add a new repository.
2. Enter **WS3\_2** as the Name under General information.
3. In the field **Clone URL**, paste the **HTTPS connection link** for your GitHub repository.
4. In the field **Credentials** select the credential “gituser” you have previously created.
5. The field Cloud Connector can remain empty.
6. Keep the default settings in the section **Webhook Event Receiver**. Make sure the field... is set to <GENERATE>. This will automatically generate a webhook token in your credentials section once the repository is added.
7. Click on **Add** at the bottom of the Add Repository panel and the repository will be added.



8. If you now check the Credentials tab, you will see a third entry there, which is the webhook token that was automatically generated when the repository was added.

| Name     | Description                          | Type                 | Actions |
|----------|--------------------------------------|----------------------|---------|
| cfdeploy |                                      | Basic Authentication |         |
| gituser  |                                      | Basic Authentication |         |
| ws3_2    | Webhook token for repository 'ws3_2' | Webhook Secret       |         |

## CI/CD: Define and trigger a job

Now that the credentials and repository are defined in CI/CD, you can create your first job that can be triggered to deploy to your database.

1. Click on the **Job** tab at the top.
2. Click on the **plus icon** to add a new job.
3. Enter **WS3\_2\_deployment** as the Job Name.
4. In the field **Repository**, select the repository **ws3\_2**, that you just added from the drop-down menu.
5. In the next field, enter the name of your GitHub branch.
6. In the following fields, keep the default settings as they are:
  - a. **Build Retention:**
    - Keep logs for: 7 days
    - Keep maximum: 50 build items
  - b. **Stages:**
    - Configuration Mode: Job Editor
    - Build Tool: mta

The screenshot shows the 'Create Job' dialog box with the following configuration:

- General Information:**
  - Job Name:** ws3\_2\_deployment
  - Repository:** ws3\_2
  - Branch:** master
  - Pipeline:** SAP Cloud Application Programming M...
  - Version:** 1.0
  - State:** ON
- BUILD RETENTION:**
  - Keep logs for:** 7 days
  - Keep maximum:** 50 build items
- STAGES:**
  - Configuration Mode:** Job Editor
  - Build:** ON (Build Tool: mta)
  - Maven Static Code Checks:** OFF
  - Lint Check:** OFF
  - Additional Unit Tests:** OFF (npm Script: test)

At the bottom right of the dialog box are the **Create** and **Discard** buttons.

7. In the field **Release** select **ON**. (SAP employees: unless you used p-user credentials in the previous steps, select **OFF** in this field)
8. In the field **Deploy to Cloud Foundry Production Space** select **ON**.

The screenshot shows the SAP BTP Cockpit interface. On the left, there's a navigation sidebar with options like Overview, Services, Cloud Foundry, HTML5 Applications, Connectivity, Security, Users, Role Collections, Roles, Trust Configuration, Useful Links, and Legal Information. The main area is titled "Subaccount: trial - Overview". It has tabs for General, Cloud Foundry Environment, Kyma Environment, and Entitlements. Under General, it shows "Modified On: Apr 16, 2021, 15:20". The Cloud Foundry Environment section contains fields for "Org Name" (highlighted), "API Endpoint" (highlighted), "Org ID", and buttons for "Manage environment instance" and "Disable Cloud Foundry". Below that is a "Spaces (1)" section with a table showing one space named "trial" with 0 applications and 4 service instances. A "Create Space" button is also present. The Kyma Environment section is partially visible at the bottom.

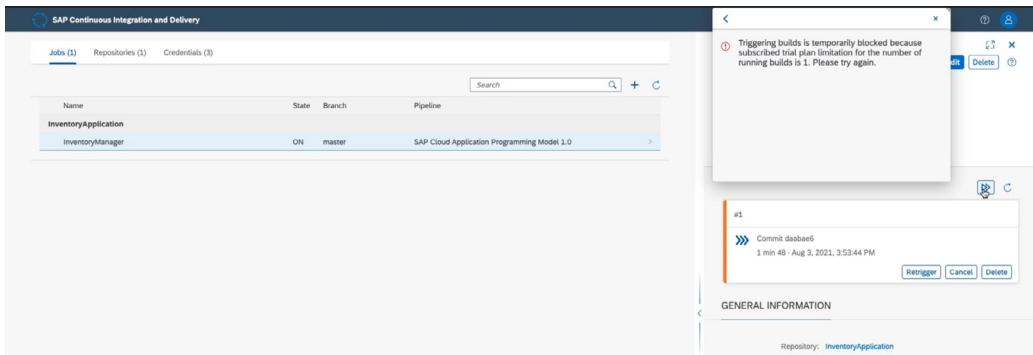
9. To enter the information in the corresponding fields for this section, you need to go to your **subaccount** in **SAP BTP cockpit** to obtain some information. (Do not close the CI/CD browser tab.)
10. Copy and paste the **API Endpoint**, **Org Name** and **Space** name from your trial account.
11. In the field **Credentials**, select **cfdeploy** from the drop-down menu.

The screenshot shows the "Create Job" dialog in SAP CI/CD. It has tabs for General Information, Build Retention, and Stages. The Stages tab is selected. Under Stages, there's a "Configuration Mode" dropdown set to "Job Editor". The "Build" section includes "Build Tool" (set to "mta"), "Maven Static Code Checks" (OFF), "Lint Check" (OFF), and a "Fail on Error" checkbox. The "Additional Unit Tests" section has an "npm Script" field set to "test". The "Release" section is highlighted with a yellow box and contains "Deploy to Cloud Foundry Production Space" (ON) and fields for "API Endpoint", "Org Name", "Space", and "Credentials" (set to "cfdeploy"). The "Upload to Cloud Transport Management" section is shown below but is OFF. At the bottom are "Create" and "Discard" buttons.

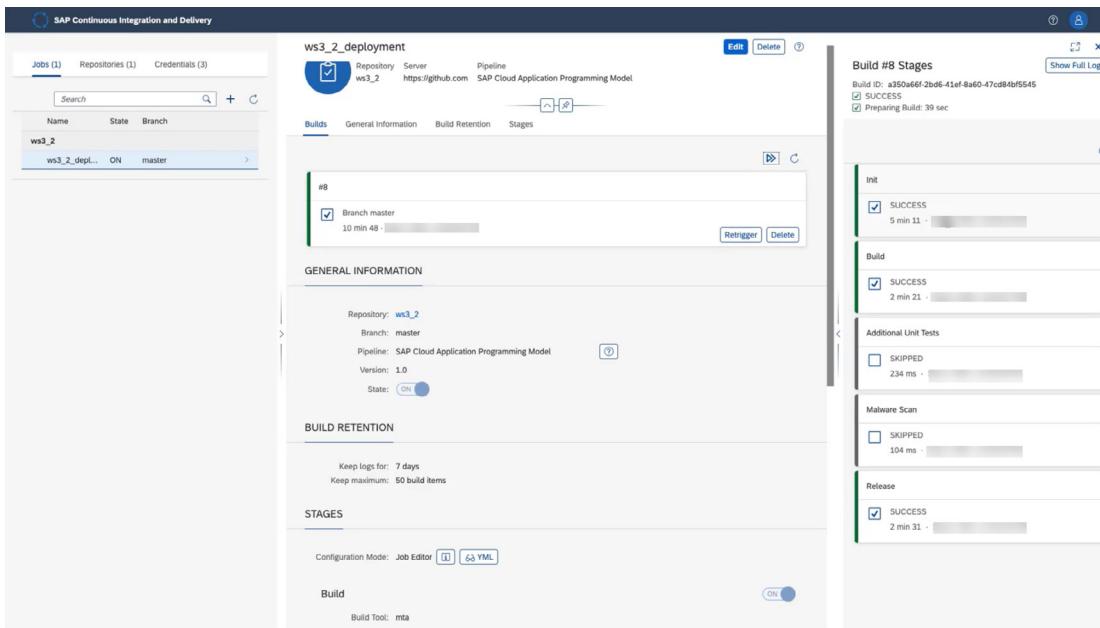
12. Finally, click on **Create**. Your job is now created, and you can see the entry in the **Jobs** tab.
13. Click on this new job **WS3\_2\_deployment**.
14. Click on the to start the deployment.

The screenshot shows the SAP Continuous Integration and Delivery interface. On the left, there's a navigation bar with "Jobs (1)", "Repositories (1)", and "Credentials (3)". The "Jobs" tab is selected, showing a table with one entry: "ws3\_2" (State: ON, Branch: master, Pipeline: SAP Cloud Application Programming Model). The "ws3\_2\_deployment" job is selected, and its details are shown on the right. It has tabs for General Information, Build Retention, and Stages. The General Information section shows "Repository: ws3\_2", "Branch: master", and "Pipeline: SAP Cloud Application Programming Model". The Stages section shows a "No builds yet" message with a "Trigger" button highlighted with a yellow box.

15. The process of deployment will take up to 10 minutes, when it is triggered for the first time.



16. Once the build is successful, your screen should look like this:

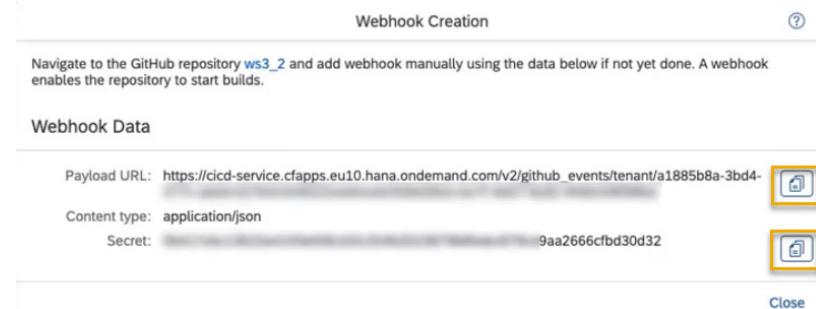


And with that, you have successfully deployed your application using the CI/CD job. In the final step, we will now show you how to use a webhook to automate this deployment chain so it will be triggered each time changes are pushed to the GitHub repository.

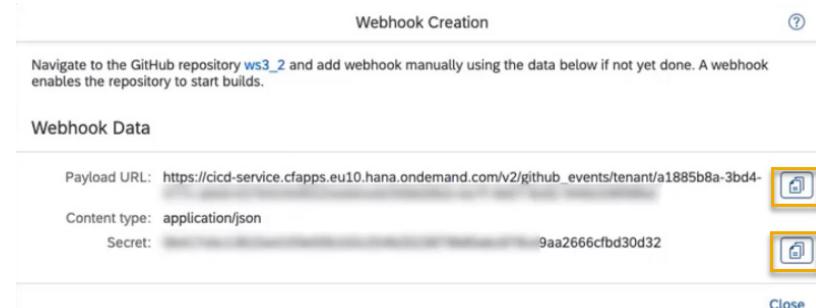
# AUTOMATE THE CI/CD CHAIN WITH A WEBHOOK

## Add webhook credentials to GitHub

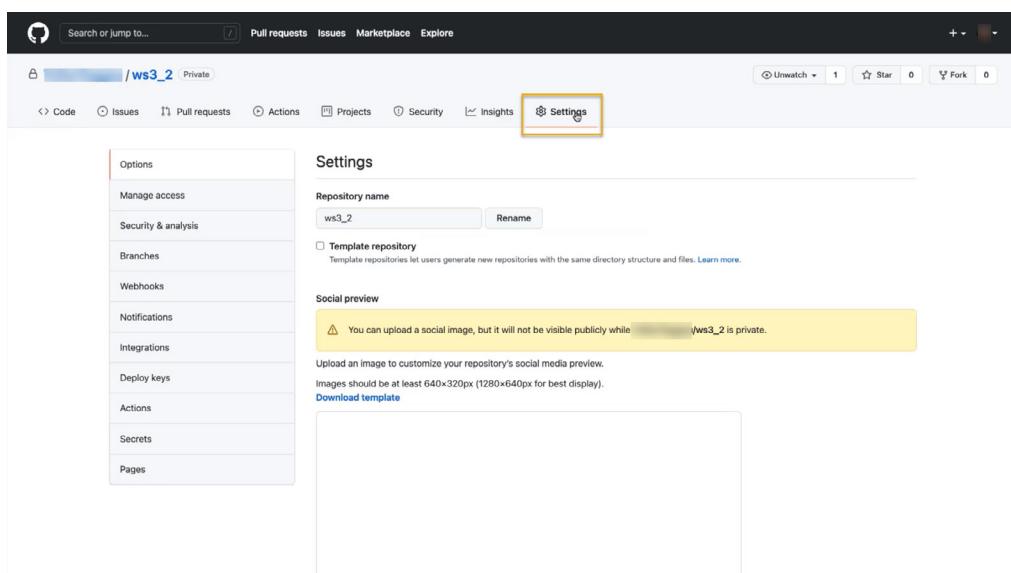
1. In CI/CD, go to the **Repositories** tab and select the repository **ws3\_2**.
2. Click on **Webhook Data** at the top right corner. This will reveal the webhook information.



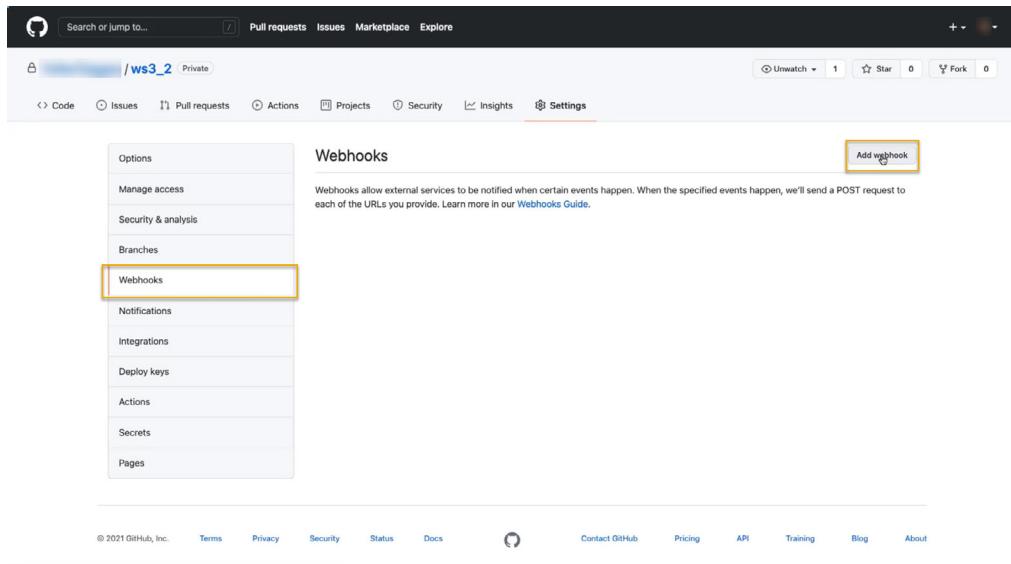
3. Copy the **Payload URL** and **Secret** from the Webhook Data dialogue box. Save them separately in a safe location.



4. Now switch to your GitHub tab and navigate to your **ws3\_2** repository.
5. In the repository, click on **Settings**.



- From the bar on the left, select Webhooks and then click on Add Webhook at the top right corner.



- Paste the Payload URL you have copied from CI/CD into the field **Payload URL**.
- In the field **Content type** select **application/json** from the drop-down menu.
- Paste the Secret you had copied earlier into the field **Secret**.

The screenshot shows the 'Add webhook' configuration form. On the left, there's a sidebar with the same options as the previous screenshot: Options, Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Actions, Secrets, and Pages. The 'Webhooks' option is selected. The main form area has several fields: 'Payload URL' (highlighted with a blue box), 'Content type' dropdown set to 'application/json' (highlighted with a blue box), 'Secret' (highlighted with a blue box), 'SSL verification' (with 'Enable SSL verification' checked), 'Which events would you like to trigger this webhook?' (with 'Just the push event' checked), and 'Active' (with the checkbox checked). At the bottom right is a green 'Add webhook' button, which is highlighted with a yellow box.

- Now, the webhook in CI/CD is connected to your GitHub repository.

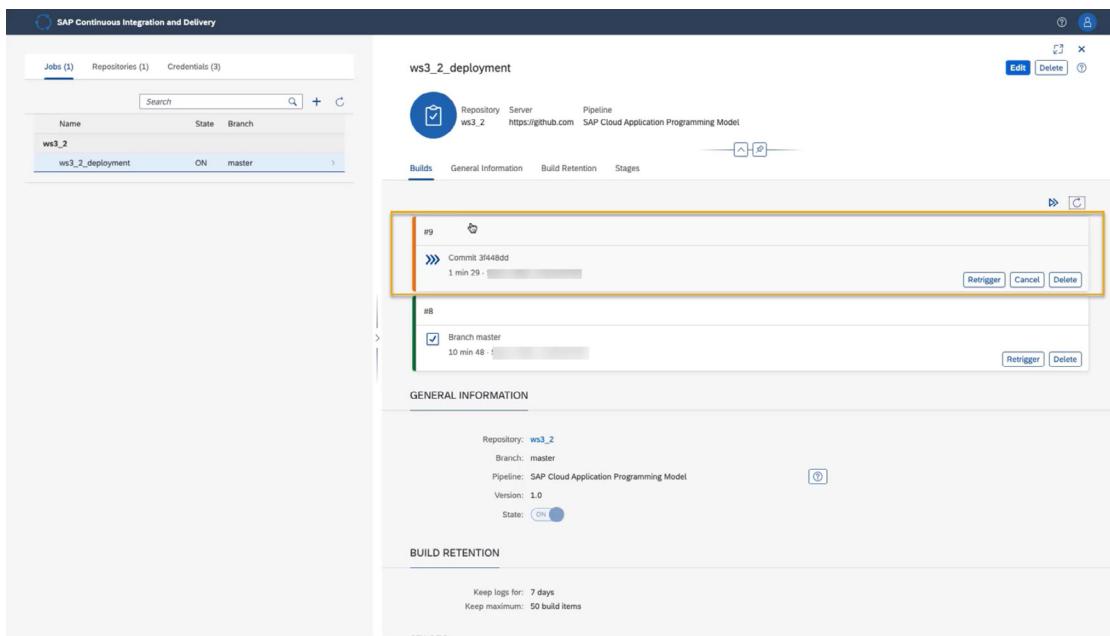
## Test the automatic deployment using the webhook

Now that the webhook is set up, you can test it by making any change to your development project in SAP Business Application Studio and pushing it to the remote repository. As soon as you push the changes, the next build of your job in CI/CD will be triggered automatically.

1. Go back to SAP Business Application Studio and to your project **ws3\_2**.
2. Select the **inventory.hdbmigrationtable** in your src folder.
3. For example, **add a Space** within the file without affecting the formatting or the contents of the file. Now, you will see a notification on the Source Control  icon informing you that the new change is yet to be pushed to GitHub.

**Note:** If you are unsure about changing any of the existing files without affecting their functionality, you can also create a new file named test.txt in your project as this new file will not affect any runtime objects or the deployment to the database.

4. Click on the  source control icon.
5. Type **test webhook** in the Message box and click on the **tick icon** to commit the changes.
6. Select **Yes** if you are asked whether to stage all changes and commit directly.
7. Click on the **three dots** icon next to the Source Control panel.
8. Select the **Push** option.
9. Go back to the SAP Continuous Integration and Delivery tab.
10. Click on **Jobs**. Select the **WS3\_2\_deployment**.



The screenshot shows the SAP Continuous Integration and Delivery interface. On the left, there is a navigation bar with 'Jobs (1)', 'Repositories (1)', and 'Credentials (3)'. Below this is a table with one row: 'ws3\_2\_deployment' (Name), 'ON' (State), and 'master' (Branch). On the right, the details for 'ws3\_2\_deployment' are shown. At the top, there is a summary card for 'ws3\_2\_deployment' with a repository icon, 'ws3\_2' (Repository), 'https://github.com' (Server), and 'SAP Cloud Application Programming Model' (Pipeline). Below this is a 'Builds' section. It shows two builds: 'a9' (Commit 3f4ebdd, 1 min 29 ·) and 'a8' (Branch master, 10 min 48 ·). Both builds have 'Retrigger', 'Cancel', and 'Delete' buttons. Below the builds is a 'GENERAL INFORMATION' section with fields: Repository: ws3\_2, Branch: master, Pipeline: SAP Cloud Application Programming Model, Version: 1.0, and State: ON. At the bottom is a 'BUILD RETENTION' section with options: Keep logs for: 7 days and Keep maximum: 50 build items.

You will see that a new build has been triggered automatically to start the deployment chain.

Now, you have automated the way to deploy your project using CI/CD. Now, whenever the GitHub repository updates, a build of the pipeline is triggered, which automatically builds and deploys your application to your space in SAP BTP. This way, you just need to build using SAP Continuous Integration and Delivery and perform a Git push to automatically deploy your project.

**Excellent! You've completed Session 2 of the Digital Hands-on Workshop!**

**Make sure to come back for session 3!**

If you have any issues with the steps described in this workbook, please ask your questions in the [SAP Community](#) using the tag [SAP HANA Cloud](#).

## Additional Resources

- [SAP HANA Cloud product page](#)
- [Upcoming SAP HANA Cloud events](#)
- [Explore more SAP HANA Cloud tutorials](#)
  - [Mission: Jump start your SAP HANA Cloud, SAP HANA Database Trial](#)
  - [Mission: Get Started with a Standalone SAP HANA Cloud, Data Lake](#)
  - [Mission: Extend Your SAP HANA On-Premise to SAP HANA Cloud, SAP HANA Database](#)
- Technical Documentation
  - [SAP HANA Cloud](#)
  - [SAP Business Application Studio](#)
  - [SAP Continuous Integration and Delivery](#)