Building Cloud Native Applications with Choreo

Training Objective:

In this training, you will learn to deploy a full stack app in Choreo.

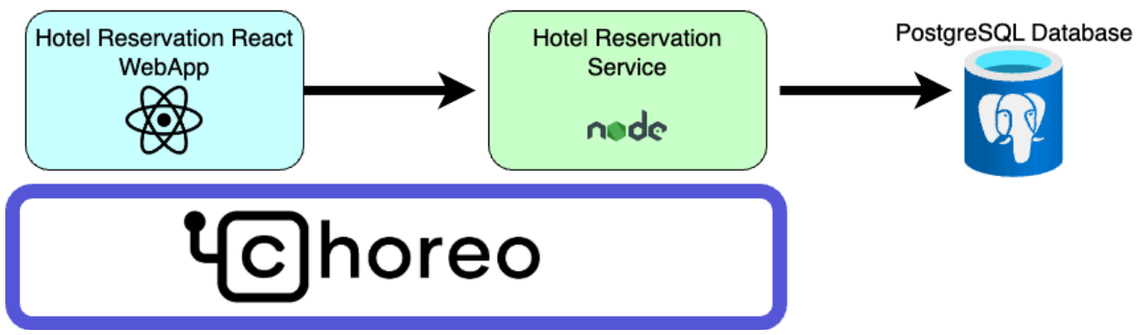
1. Deploy a backend service with a database
2. Deploy a frontend web app
3. Securing the frontend with Choreo Managed Authentication
4. Connecting the backend and frontend
5. Monitoring and troubleshooting

Prerequisites:

1. A GitHub Account
2. Git installed in your workstation
3. A recent version of Google Chrome, Mozilla Firefox
4. Microsoft Visual Studio (VSCode)
5. Choreo Account
6. NodeJS installed (above v20.11.0)
7. DBeaver or any other postgresql client

Business Scenario:

A luxury hotel has a reservation system that needs to be deployed to the cloud. The hotel reservation system will include a Hotel Reservation Service developed using NodeJS, and a Hotel Reservation Web Application developed using ReactJS. The data will be persisted in a postgreSQL database.



The web application offers the following features:

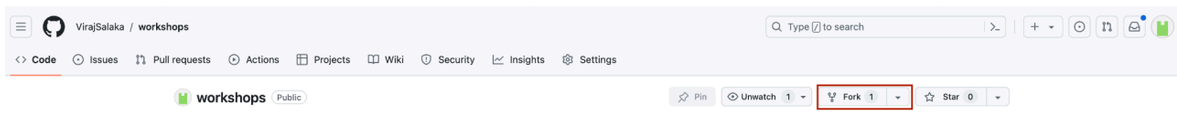
- **Room Search:** Users can search for rooms by specifying check-in and check-out dates, with an option to filter by number of guests. Search results will showcase a list of room types (eg: single, double etc). Each room listing will feature a "Reserve" button for easy booking.
- **Room Reservation:** To reserve a room, users need to input personal information: full name, contact number, and email.mUpon reservation, a unique reference number is displayed for the user to copy.
- **List Reservations:** Users can look up their reservations once they are logged in. Each entry in the list will provide options to either update the reservation details or cancel the reservation.
- **Update Reservations:** Users have the flexibility to modify any part of their reservation.
- **Cancel Reservations:** Users can cancel their reservations at any time, with a straightforward option to cancel their booking.

Detailed Steps:

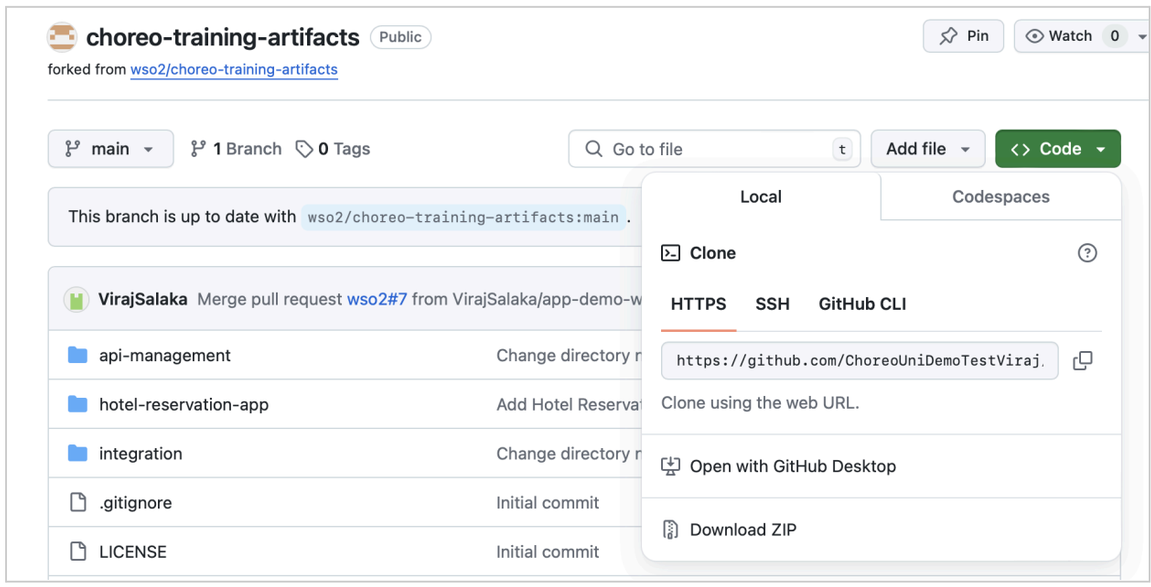
Follow the steps below to deploy the Hotel Reservation System.

Step 1: Fork and Clone the repository

1. Go to https://github.com/wso2/choreo-training-artifacts and click **Fork** repository.



2. To clone the repository, click the **Code** dropdown and copy the HTTPs URL that appears in the text box.



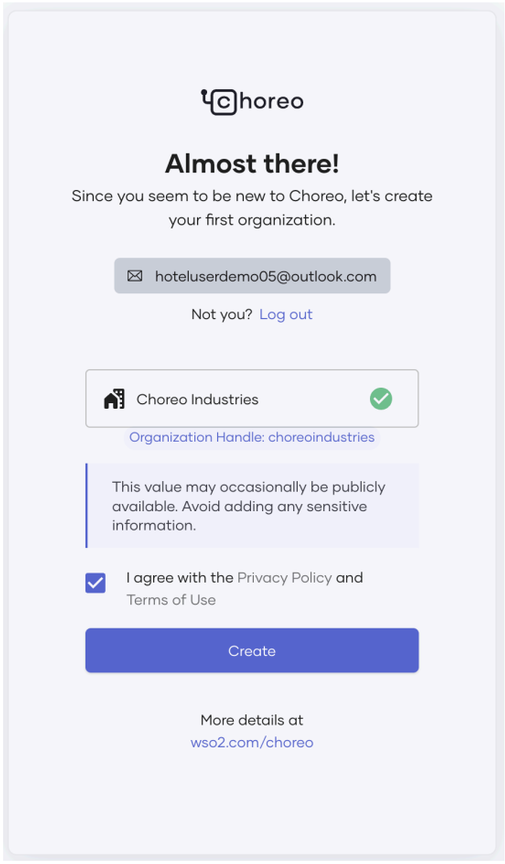
3. Run the following command in your terminal by adding the URL you copied in the previous step.
For Windows users: Open git bash and run the command below.

```
Unset
git clone <your_repository_url>
```

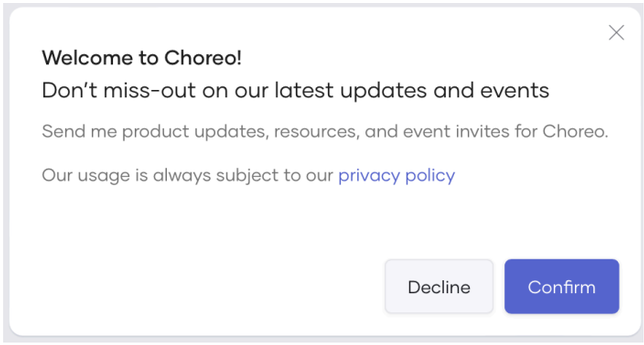
Step 2: Sign Up to Choreo

1. Sign Up to Choreo.

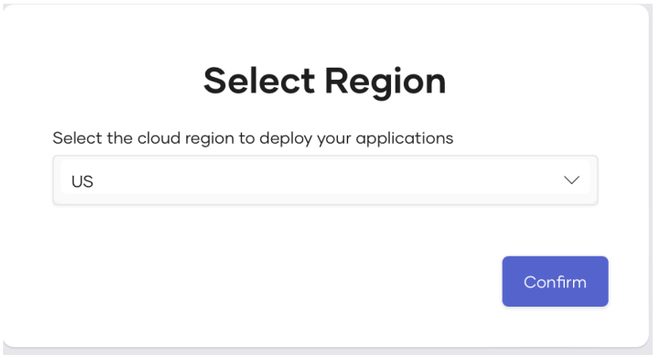
2. Once you log in to Choreo for the first time, you will be asked to create an organization. Provide a name for your organization and click **Create**. *(The organization name **must** be unique)*



3. If you are willing to subscribe for new updates regarding choreo, click **Confirm**.

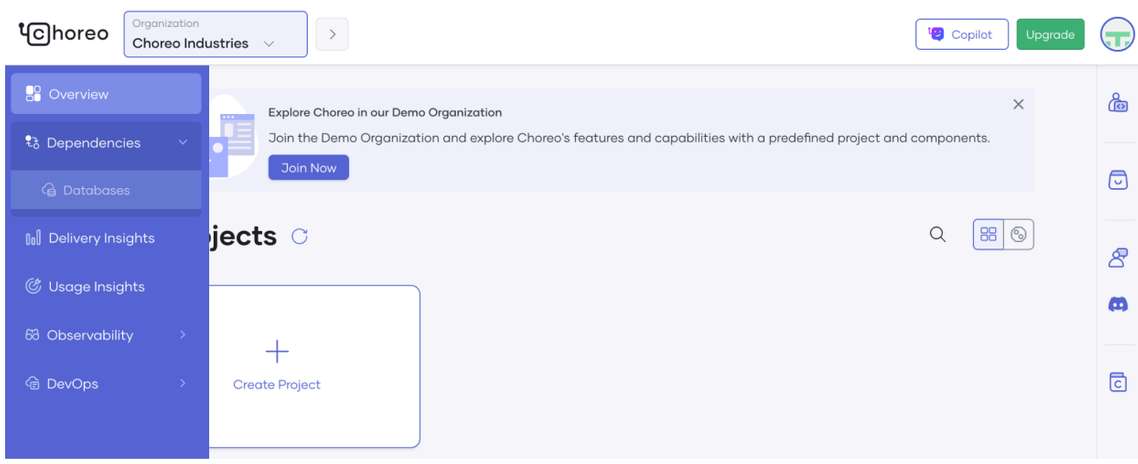


4. In the region selector window, keep the default selected option as US. Then click **Confirm**.

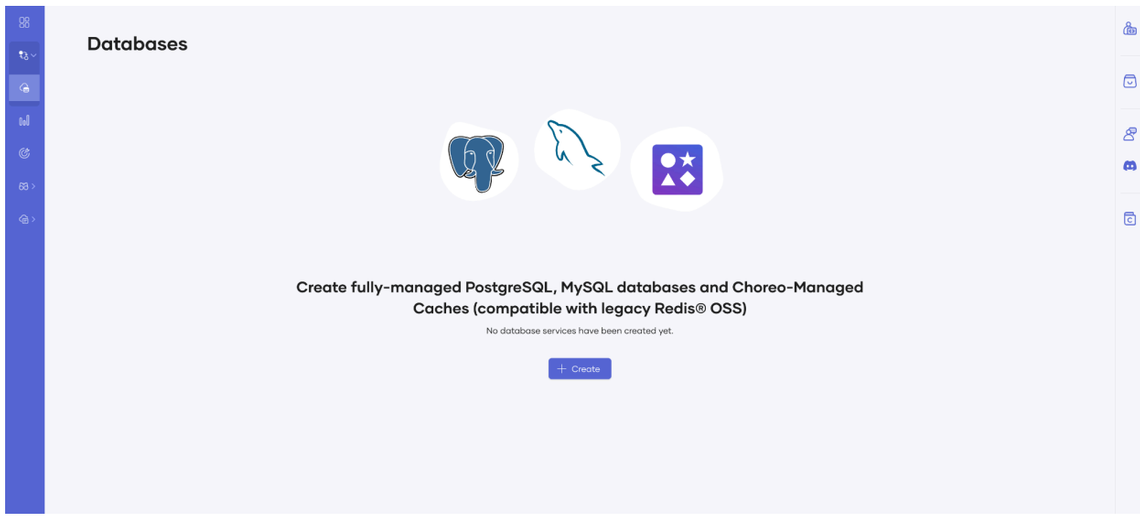


Step 3: Create a database

1. You are currently in the Organization view. In the left navigation menu, click **Dependencies -> Databases**.

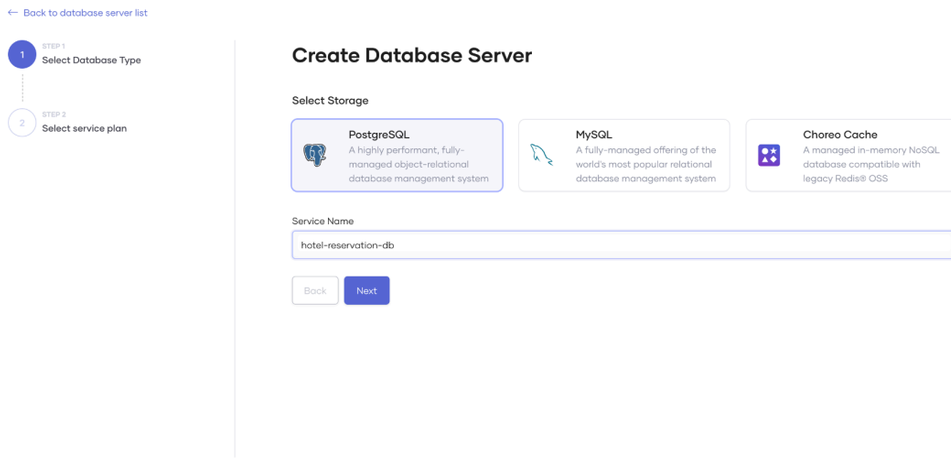


2. In the Databases page, click the **Create** button.

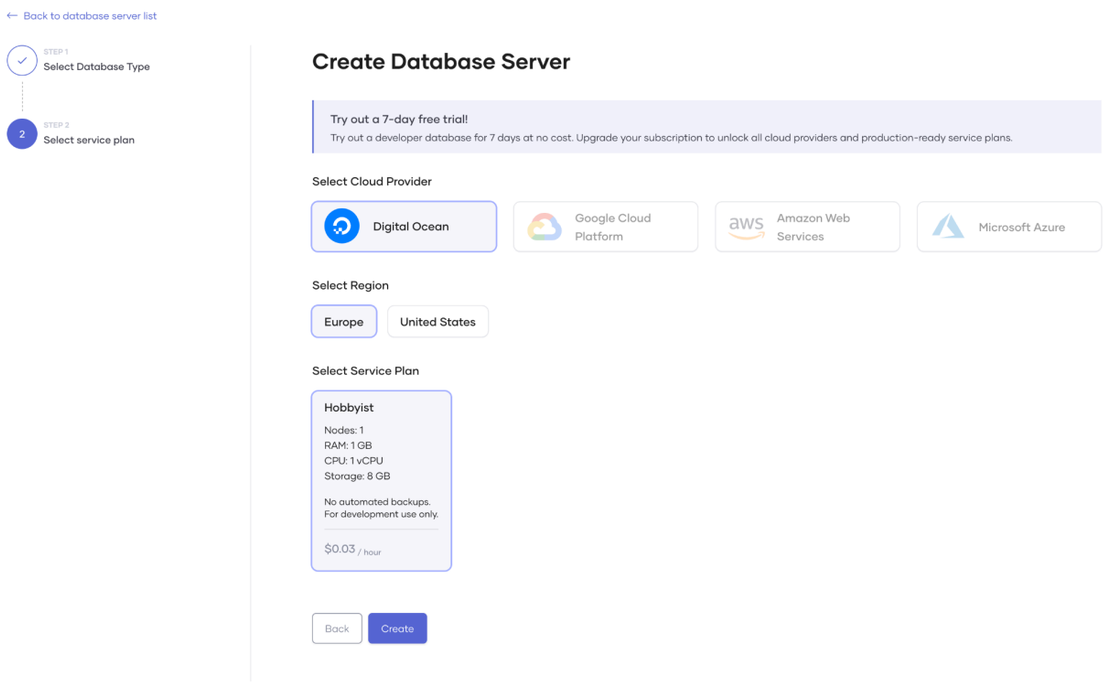


3. Add the details shown below and click **Next**. *(This is a reference for the database and not the database name)*

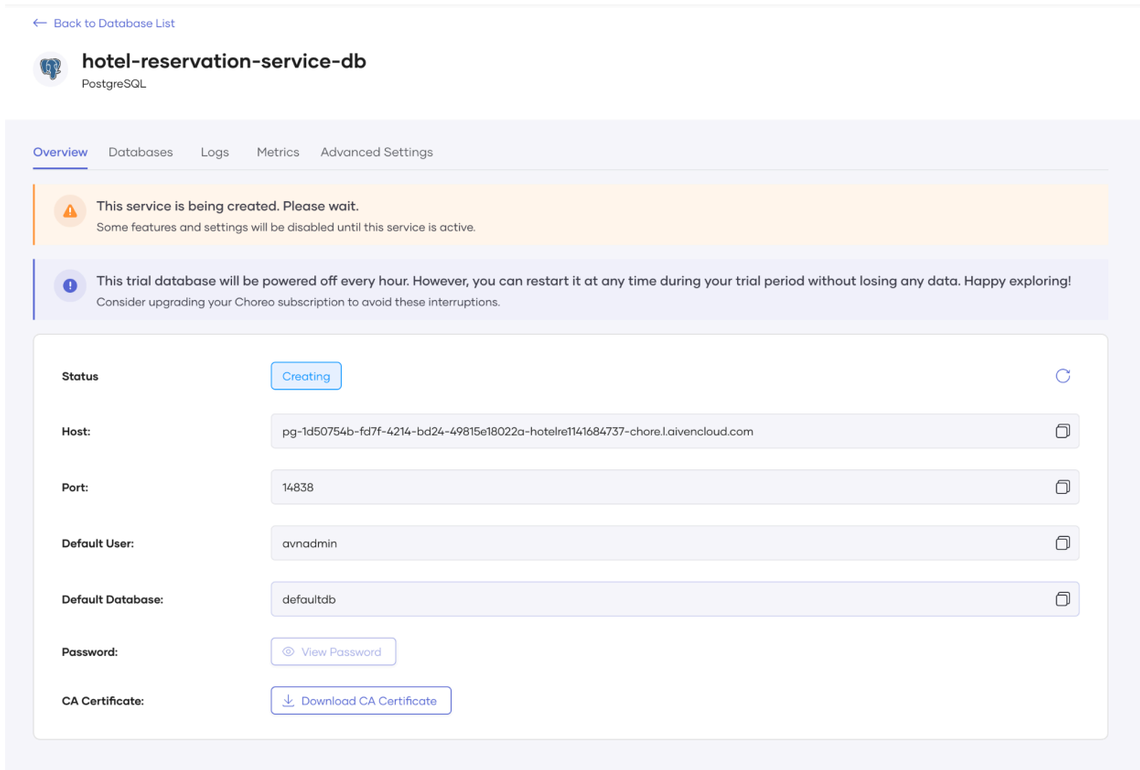
Field Name	Field Value
Name	hotel-reservation-service-db



4. The free tier only provides Digital Ocean as the available cloud provider. Leave the default values and click on the Hobbyist card under the **Select Service Plan**. Then, click **Create**.



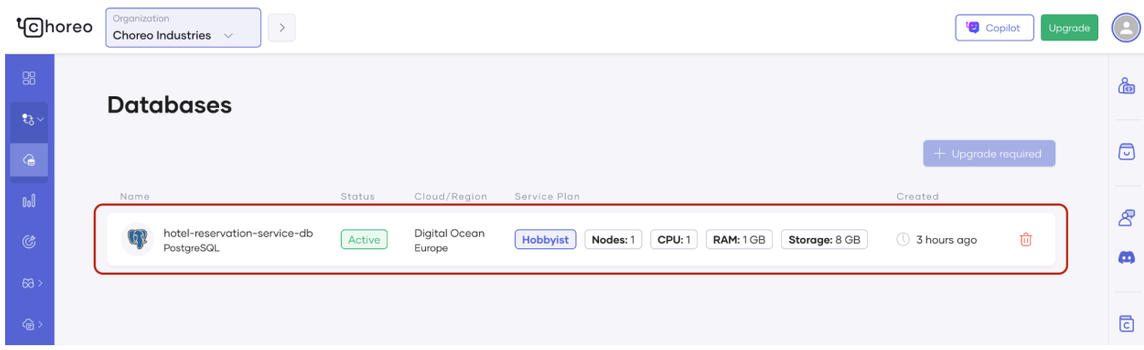
5. The created database will be displayed (The database creation process may take a few minutes).



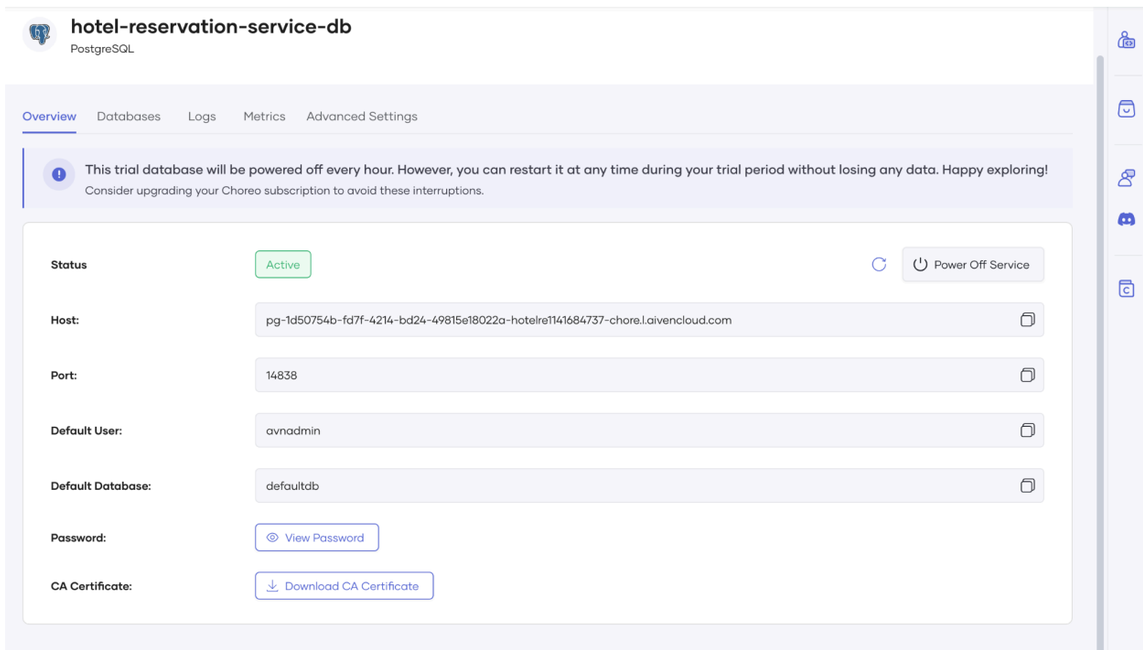
Step 4: Populate the database

1. Open the Choreo Console in a new tab.

2. Click the **Organization** card in the top menu. In the left navigation menu, click **Dependencies -> Databases**. You can see your database and view database related information. Make sure that the database is in **active** state to perform the following operations. Click on the row to see database information.

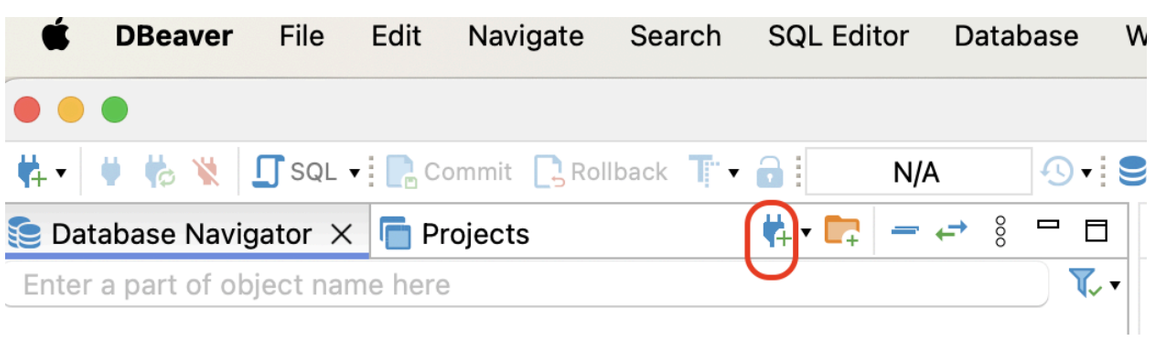


3. We will use the database information to make a connection from our postgreSQL database client.

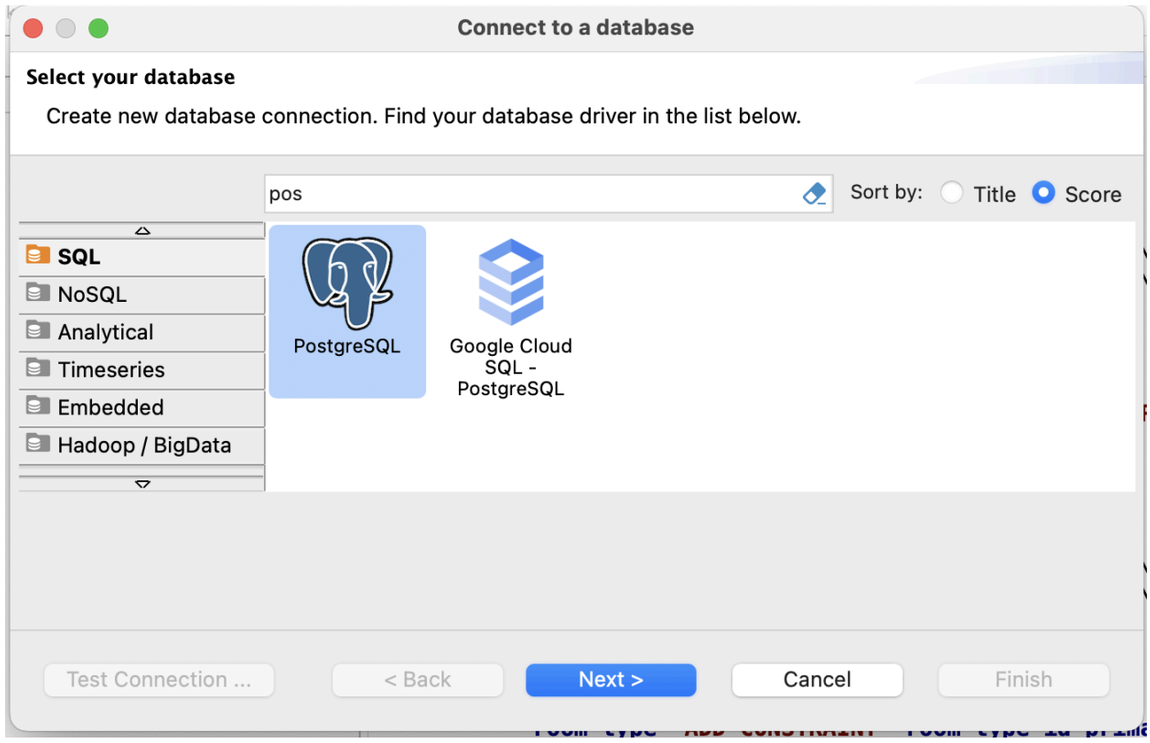


4. Open DBeaver (or any other postgreSQL client you use. The following steps are for DBeaver).

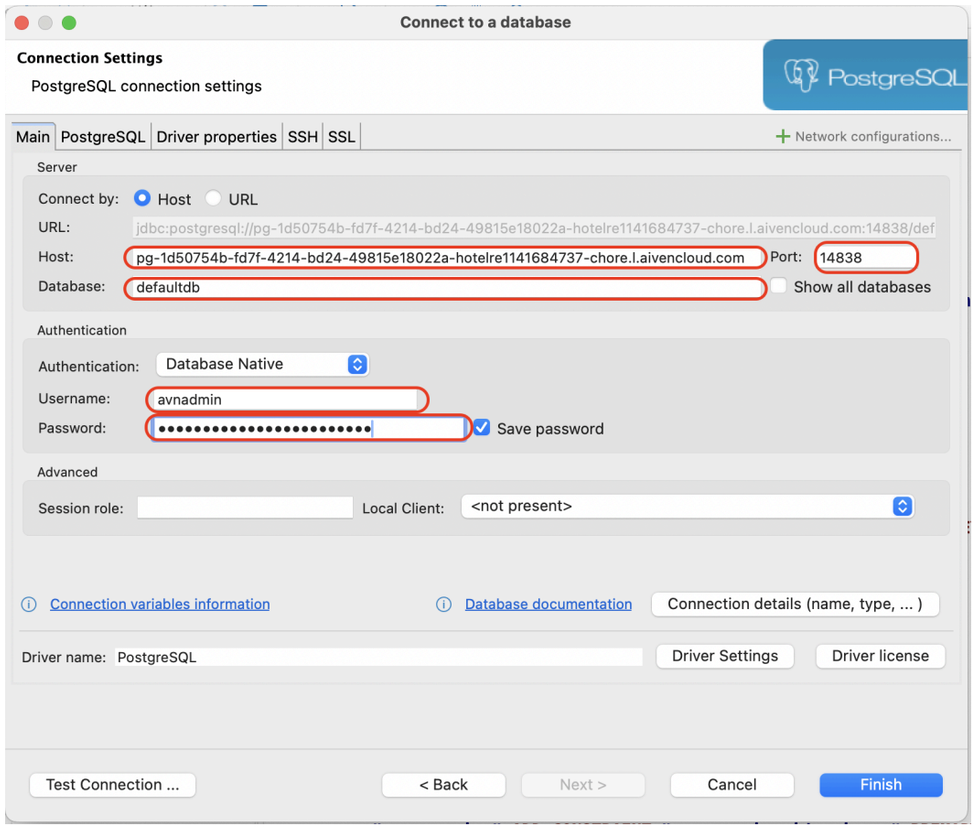
5. From DBeaver, create a database connection.



6. Select the driver as **PostgreSQL** and click **Next**.

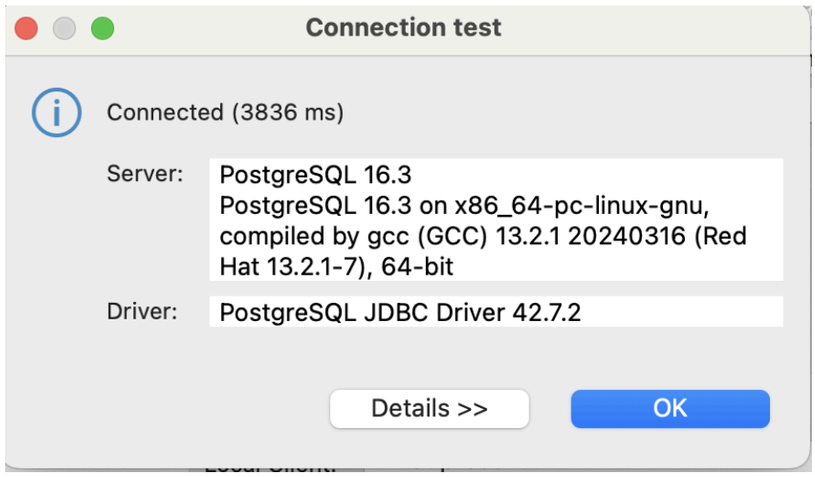


7. Update the Host, Port, Database, Username and Password with the database details obtained in step 3.



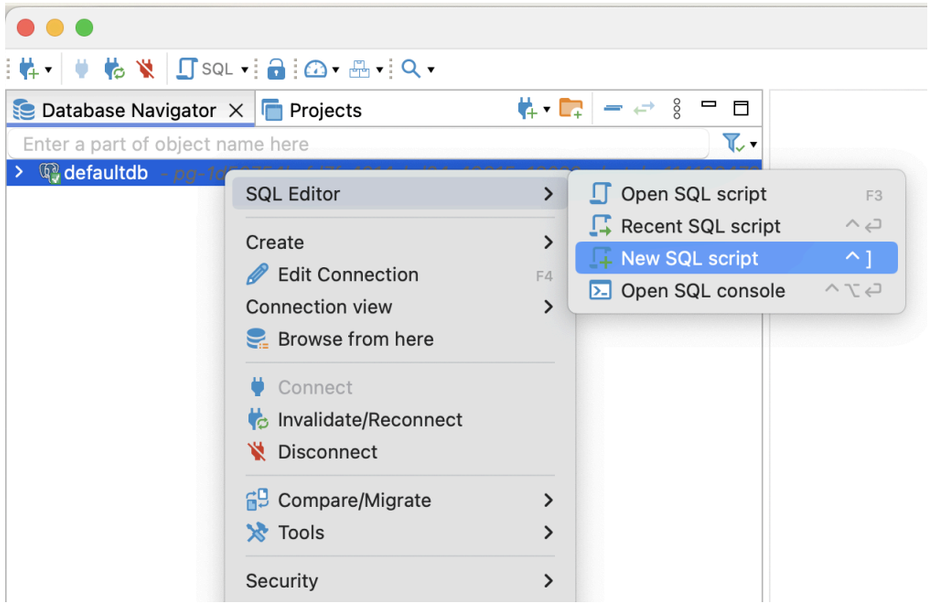
8. Click **Test Connection** to verify the configurations. If the connection is successful you will see the following message.

Note: If your connection test fails, verify your configurations and whether the database is in active state.

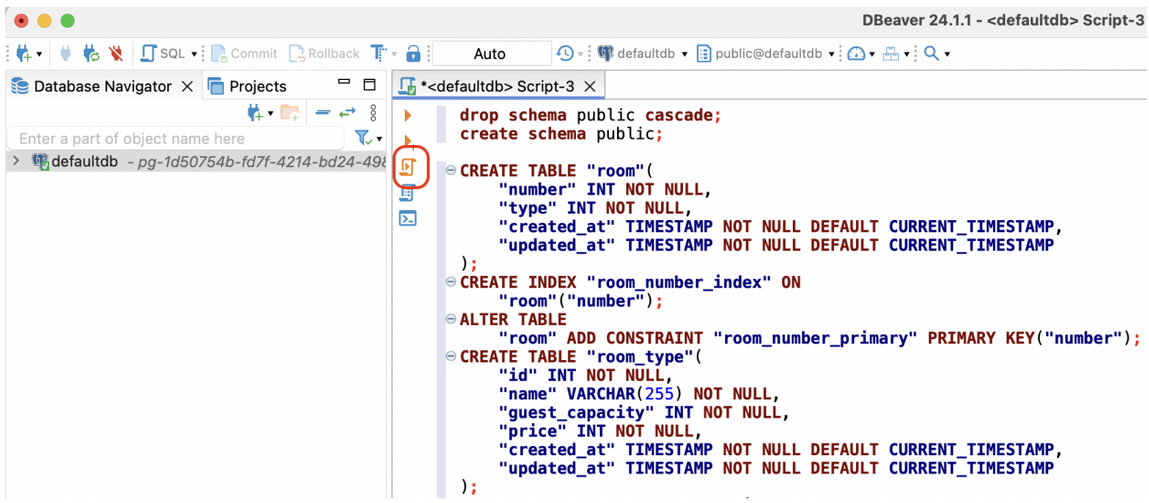


9. Finally click **Finish** to confirm the connection.

10. Right click on the database and select **SQL Editor -> New SQL** script and click.



11. Copy the content you find in this schema.sql file to the create script. And then click on the Execute Script icon. Wait till all the lines are executed.

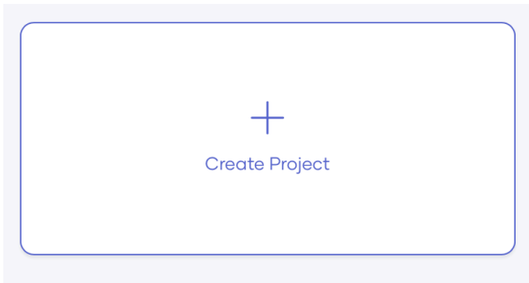


12. Open another script as in step 10. And Copy the content from the seed.sql file and execute the complete script. The database is now populated.

Step 5: Create a new Project

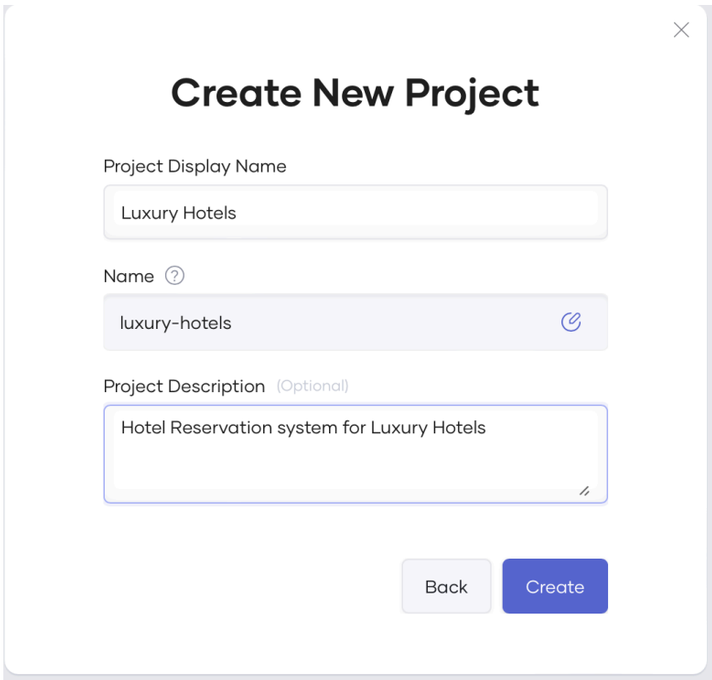
1. Click the Organization card in the top menu.

2. Click **Create Project** to create a new project.



3. Add the details shown below and click **Create**.

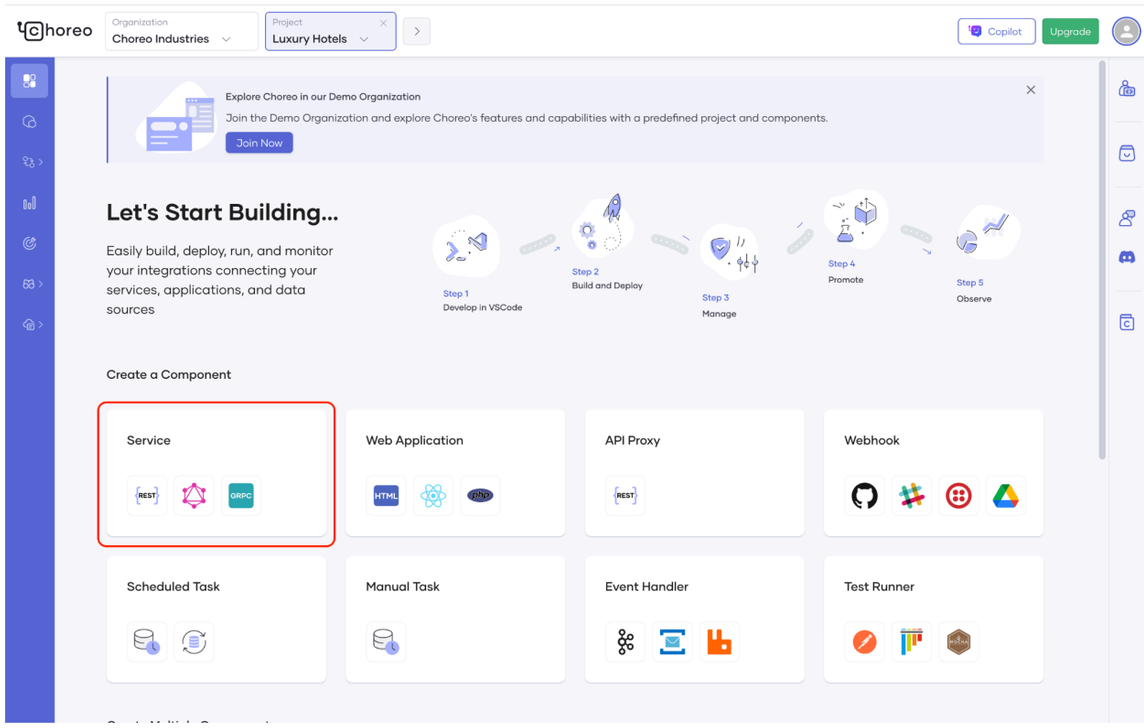
Field Name	Field Value
Name	Luxury Hotels



4. Once the project is successfully created, you will be directed to the overview of the project.

Step 6: Create and Deploy the Hotel Reservation Service

1. To deploy the backend NodeJS service, we will create a service component. On the component page, select **Service**.



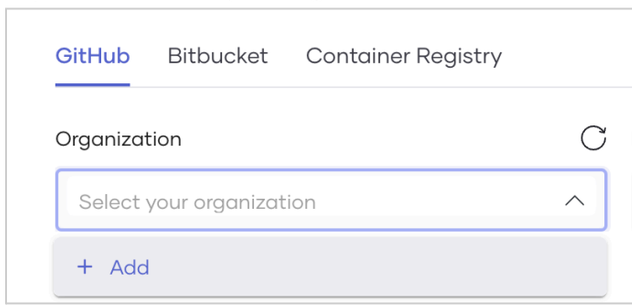
2. Provide a name for the service component.

Field Name	Field Value
Component Display Name	Hotel Reservation Service

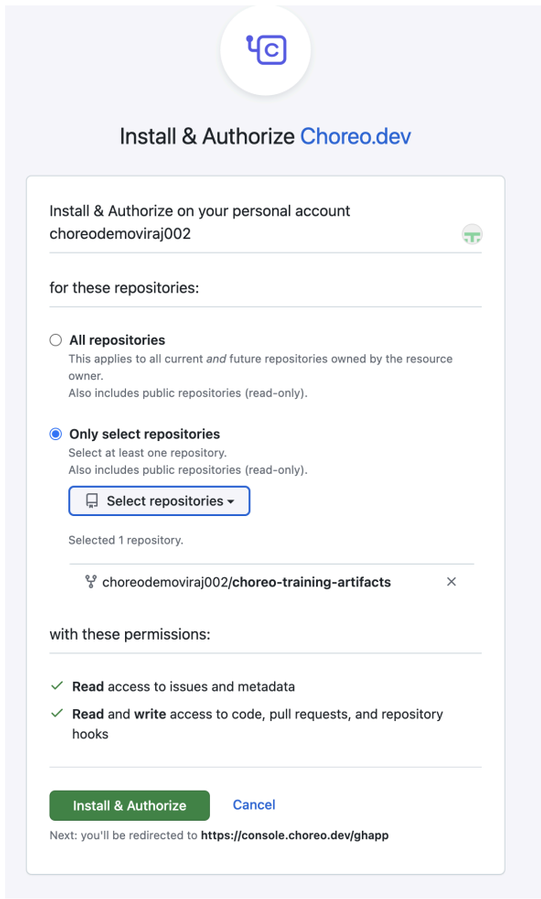
3. Click on the **Authorize With GitHub** button and click **Authorize Choreo.dev**.

Note: If you prefer not to authorize choreo.dev for your GitHub account at the moment, you can provide the GitHub repository URL in the input field provided and proceed.

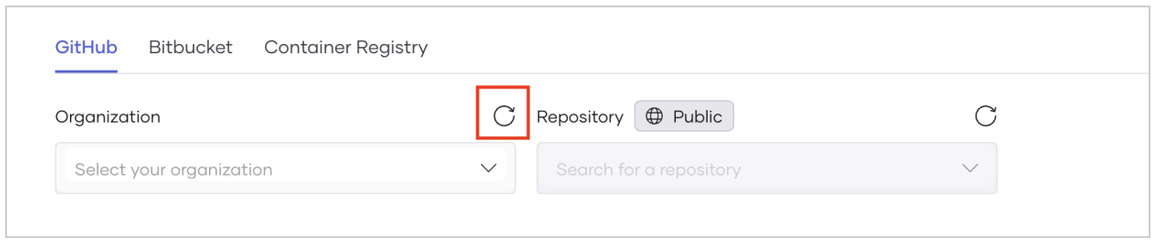
4. Expand the Select your organization section and click **+Add**.



5. Choose the **Only select repositories** radio button (You can select the **all repositories** option is preferred). Then select the **choreo-training-artifacts** repository from the repository list and finally click **Install & Authorize**.



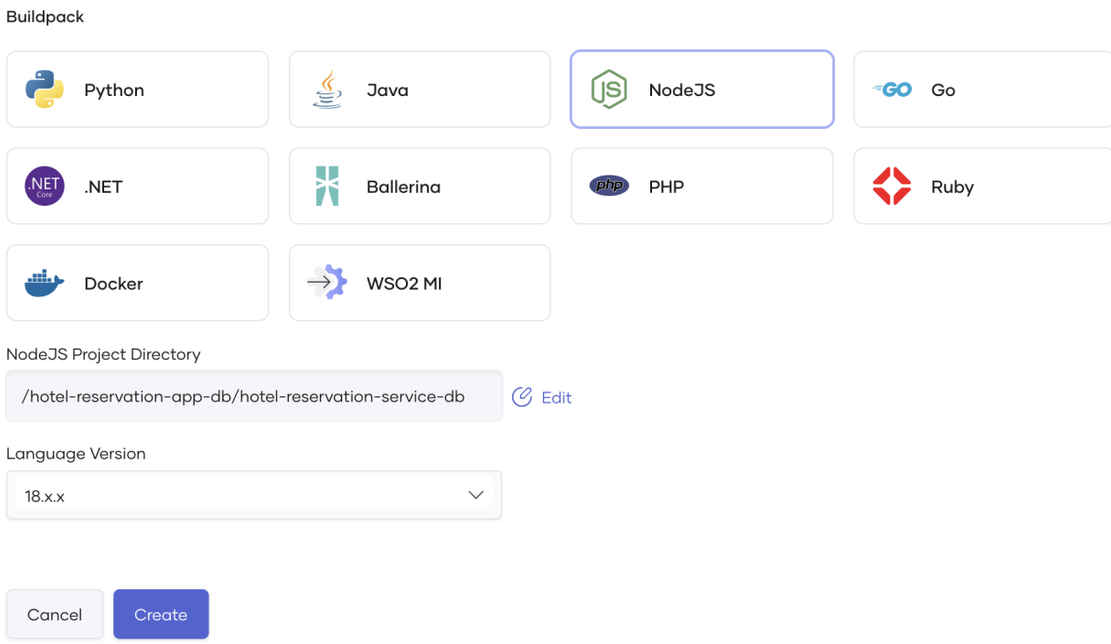
6. Click the refresh icon next to Organization.



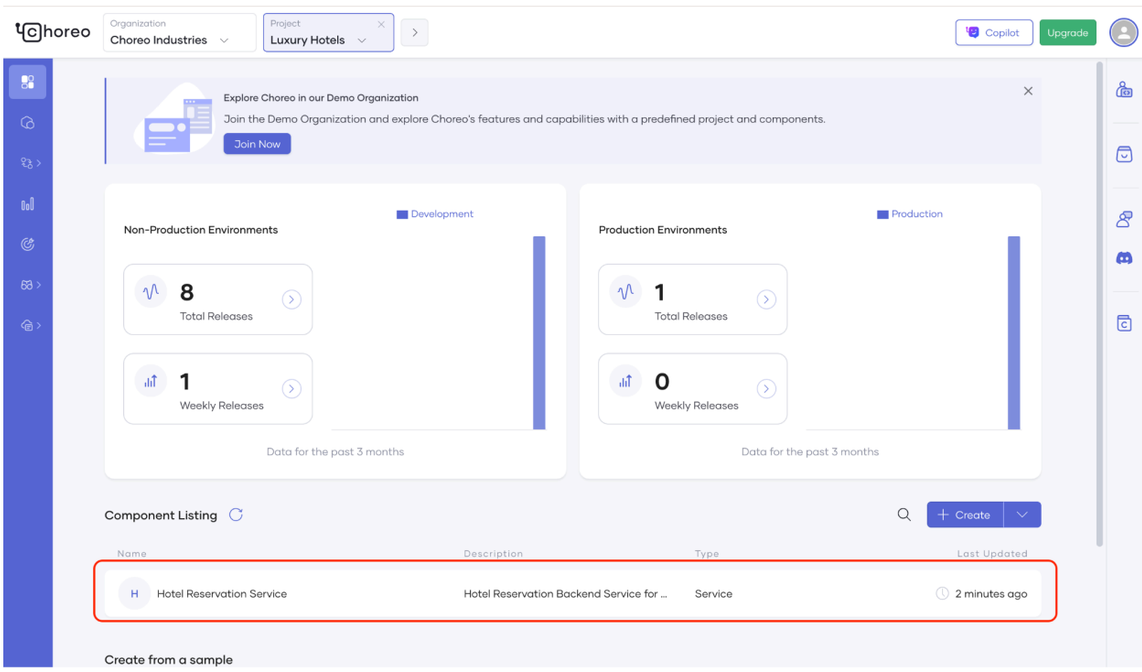
7. Select the relevant organization and select the **choreo-training-artifacts** repository. The branch will be automatically selected as main.

8. Select NodeJS as the Buildpack and set the Project Directory as below. Then click **Create**.

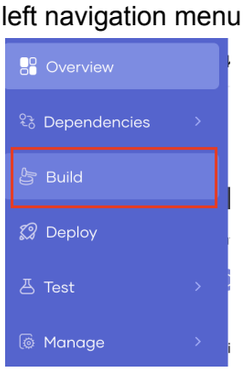
Field Name	Field Value
NodeJS Project Directory	hotel-reservation-app-db/hotel-reservation-service-db
Language Version	18.x.x



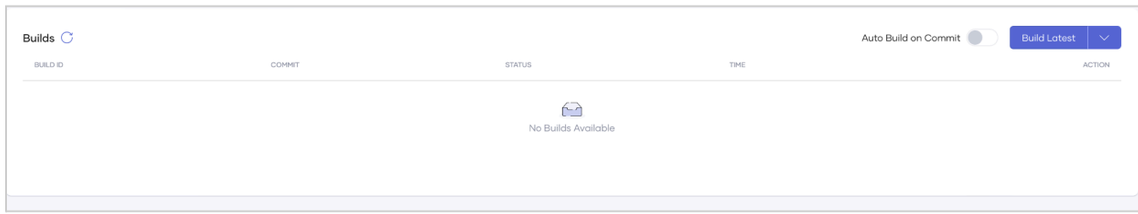
9. Your component will now appear in the component listing. Click on the component to proceed.



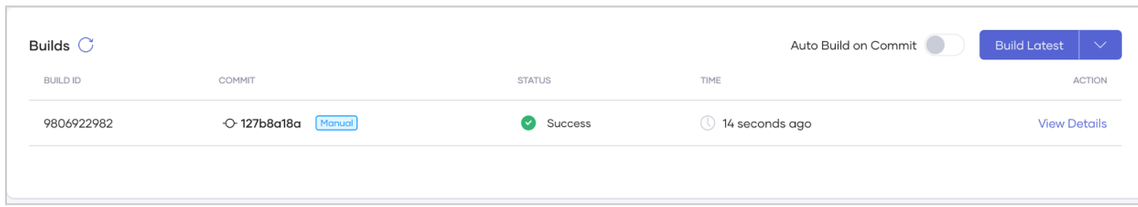
10. Once you click the component from the listing as mentioned above, you are on the component's page. Next step is to build the code. Navigate to the **Build** page from the left navigation menu.



11. Click **Build Latest** to build the latest commit of the code. This process will take a few minutes.

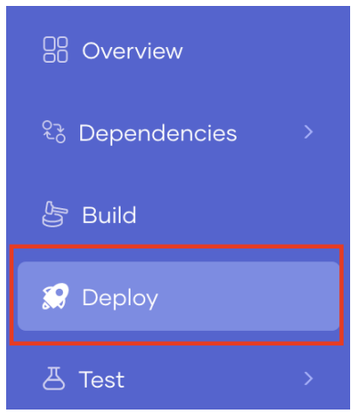


Once the build is completed, the status will show as **Success**.

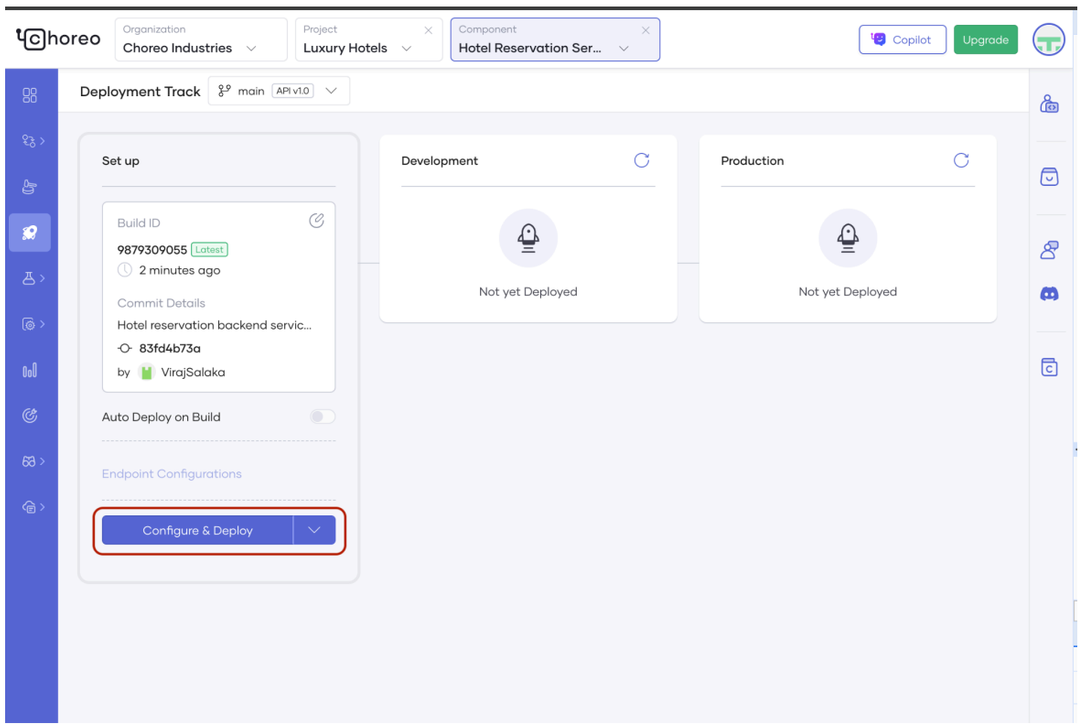


12. Until the build is complete, we can move to populating the database with necessary tables and data in **Step 3**.

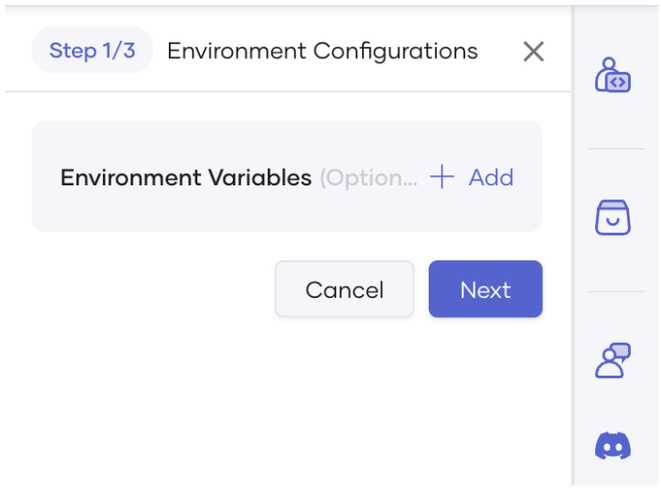
13. After a successful build the next step is to deploy the service. Click **Deploy** on the left navigation menu.



14. In the Deploy page, click **Configure & Deploy**.



15. In the right side panel, you can add environment variables by clicking **+Add**.

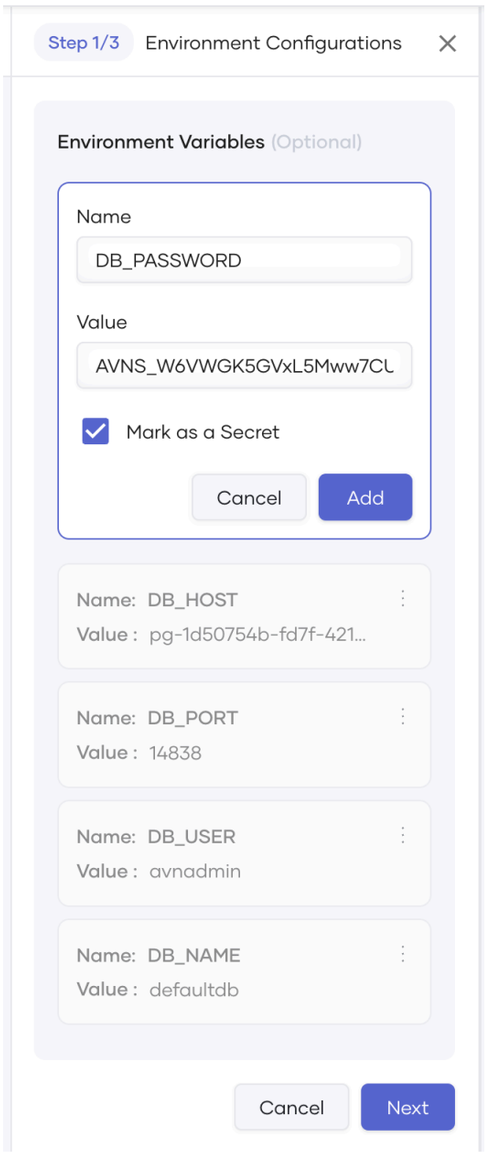


16. Add the following key value pairs as environment variables from the database connection details from Step 3. (You can refer to step 4.3 on how to obtain these values.

Name	Value
DB_HOST	*Your database host*
DB_PORT	*Your database port*
DB_USER	*Your database username*
DB_NAME	*Your Database name*

17. Let's add the database password as a secret. Get the password from database connection details as in above. Click **+Add** and provide the value, **tick the check box** to make it a secret. After providing the key value pair click **Add.**

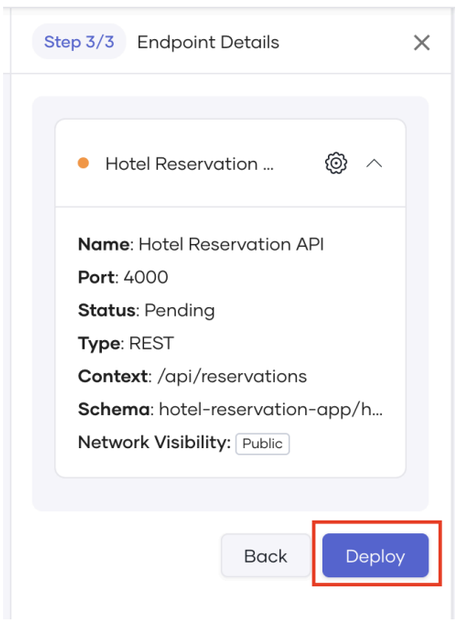
Name	Value
DB_PASSWORD	*Your database password*



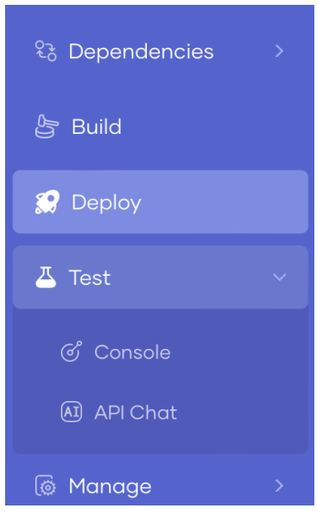
18. After all the environment variables are added, click **Next**.

19. As for file mounts, click the **Next** button in the right side panel, as we don't have any configurations to provide.

20. Click **Deploy** in the right side panel to deploy the service.

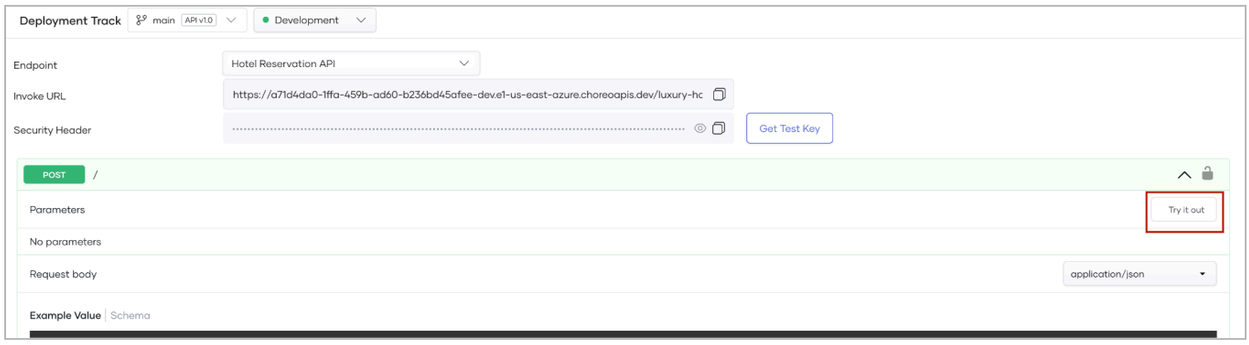


21. Navigate to **Test** in the left navigation menu and expand it. Click **Console** to access the Open API Console to test the service.



22. To test the service deployed in the development environment, select **Development** from the drop down.

23. Expand the **POST/** resource and click **Try it out.**



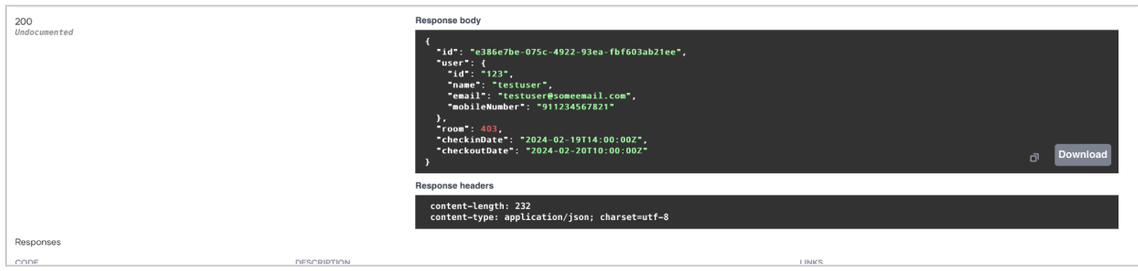
24. Provide the following details to the Request body section:

Field Name	Field Value

Request body	{
	"checkinDate": "2024-02-19T14:00:00Z",
	"checkoutDate": "2024-02-20T10:00:00Z",
	"rate": 120,
	"user": {
	"id": "123",
	"name": "testuser",
	"email": "testuser@someemail.com",
	"mobileNumber": "911234567821"
	},
	"roomType": "Family"
	}

25. Click **Execute**.

26. Upon a successful POST request, you will receive a response as shown below.



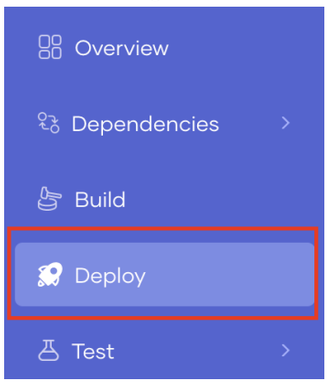
27. Expand the **GET /roomTypes** resource and click **Try it out**.

28. Add the following details for each parameter and click **Execute**.

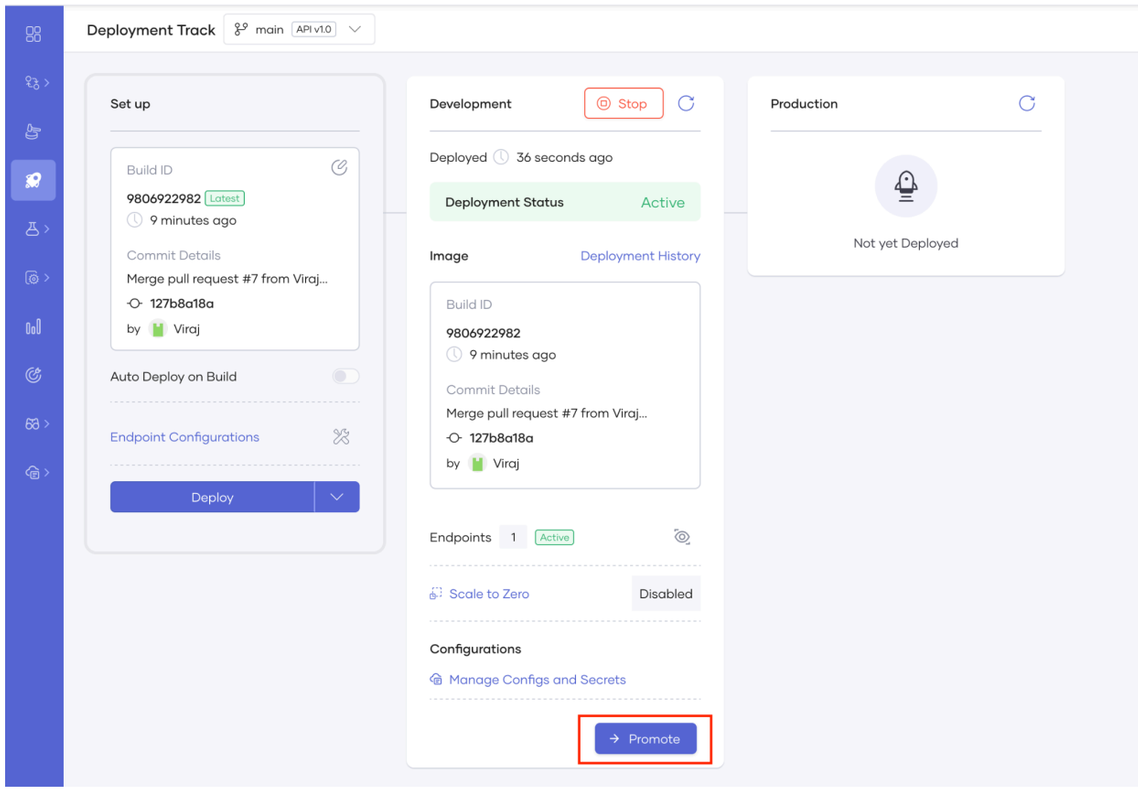
Field Name	Field Value
checkinDate	2024-02-19T14:00:00Z
checkoutDate	2024-02-20T14:00:00Z
guestCapacity	2

29. Upon a successful GET request, you will receive a response as shown below.

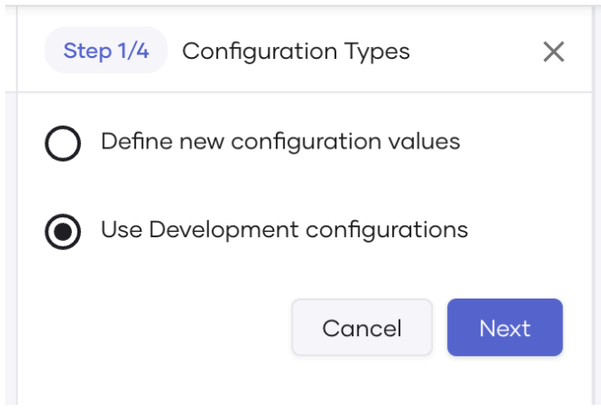
30. The next step is to deploy the service to the production environment. Click **Deploy** on the left navigation menu.



31. Click **Promote** to promote the build to the Production environment.

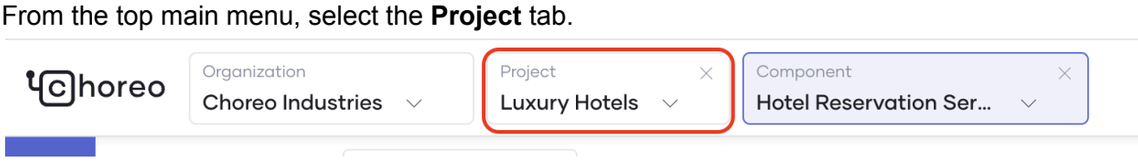


32. In the right side panel, choose **Use Development Configurations** and click **Next.** In all the next configurations in the right side panel click **Next** and finally **Promote**.

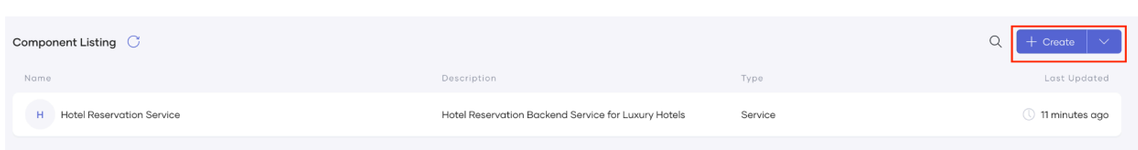


Step 7: Create and Deploy the Hotel Reservation Web Application

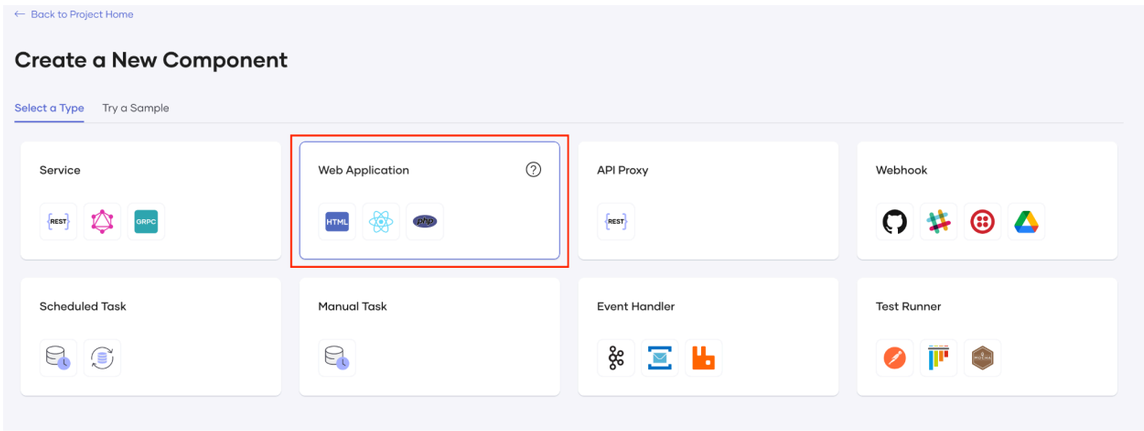
1. From the top main menu, select the **Project** tab.



2. Click **Create** to create a new component.



3. To deploy the React web application, we will create a Web Application component. On the component page, select **Web Application**.

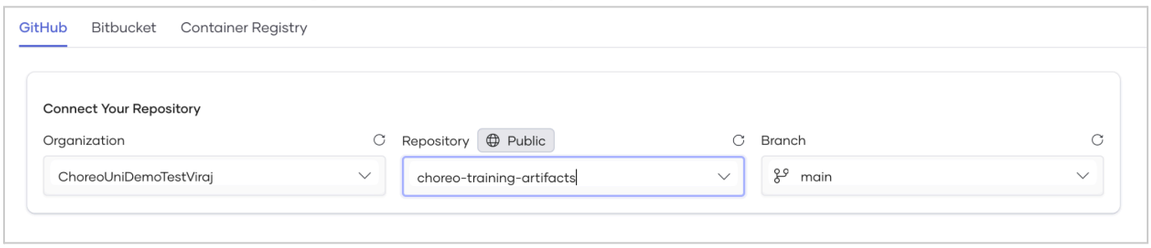


4. Provide a name for the Web Application.

Field Name	Field Value
Name	Hotel Reservation Frontend

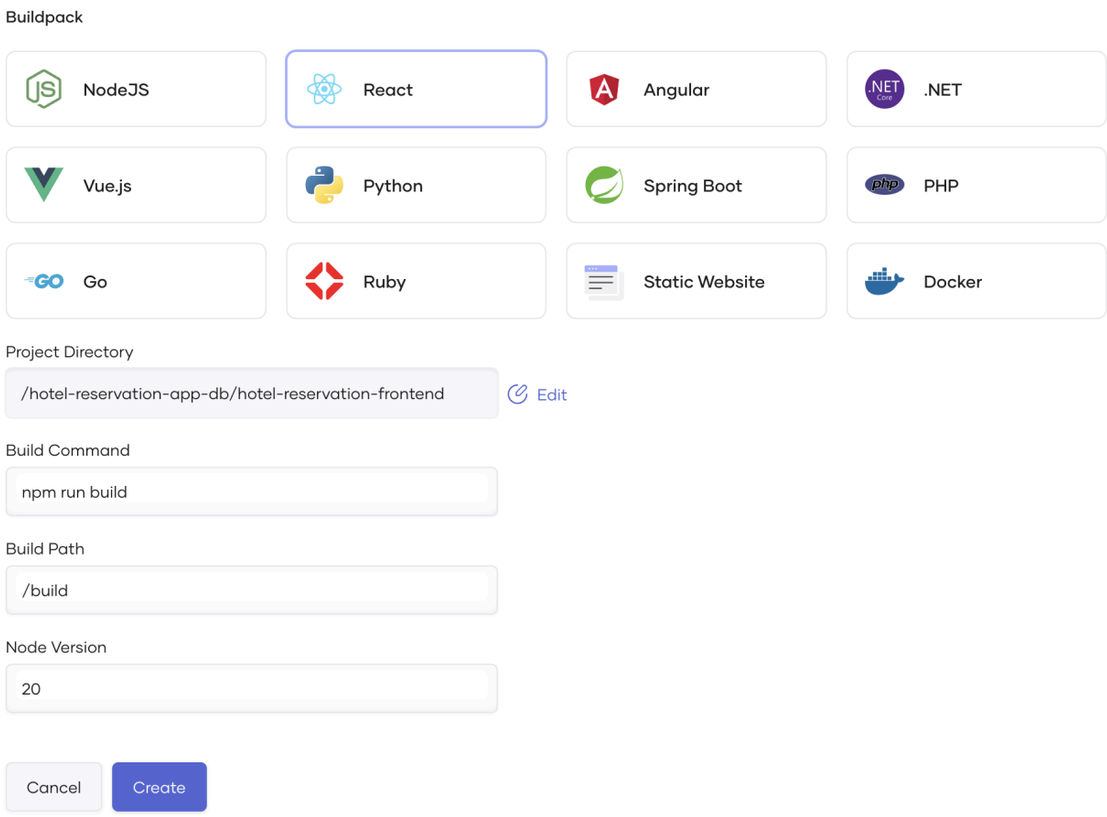
5. Click on **Authorize With GitHub**.

6. Select the **choreo-training-artifacts** repository and the **main** branch.

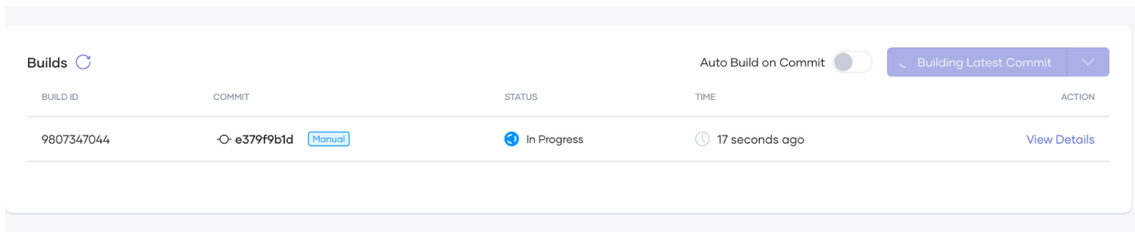


7. Select the **React** Buildpack and set the following details. Then click **Create**.

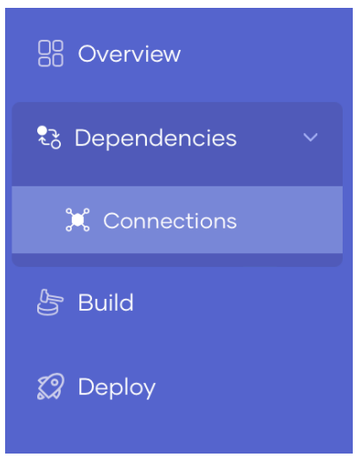
Field Name	Field Value
BuildPack	React
Project Directory	/hotel-reservation-app-db/hotel-reservation-frontend
Build Command	npm run build
Build Path	/build
Node Version	20



8. Click **Build** on the left navigation menu. Click **Build Latest**.

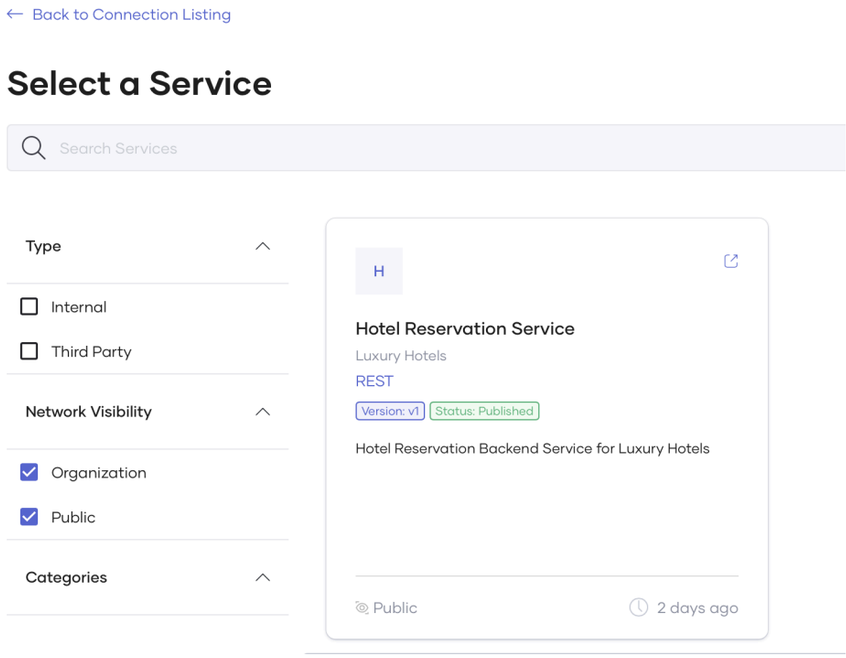


9. In order to connect the frontend and the backend we need to create a connection in Choreo. Click and expand **Dependencies** on the left navigation menu and click the **Connections** tab.



10. Click **Create**.

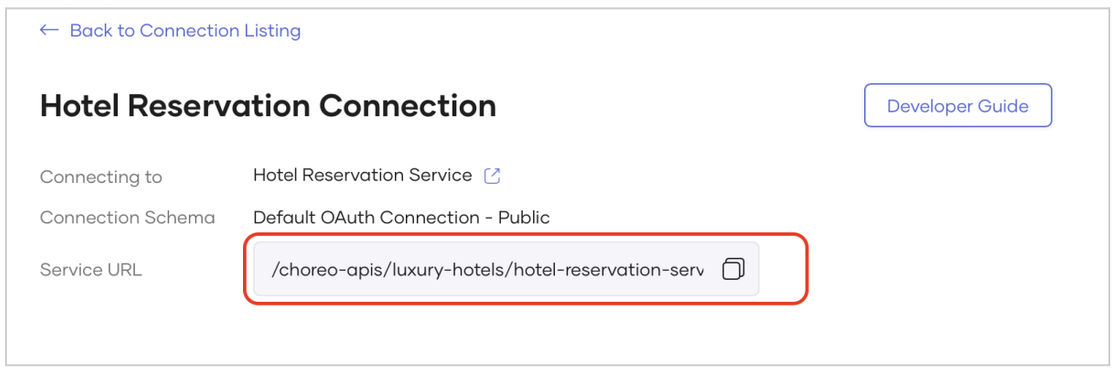
11. Select the **Hotel Reservation Service** that was created in step 4.



12. Provide a name for the connection and click **Create**.

Field Name	Field Value
Name	Hotel Reservation Connection

13. Copy and save the Service URL as we need to provide it as a configuration when deploying the web application.



14. Click **Deploy** on the left navigation menu and click **Configure & Deploy** .

In the right side panel, provide config.json as the Configuration File Name and provide the following content:

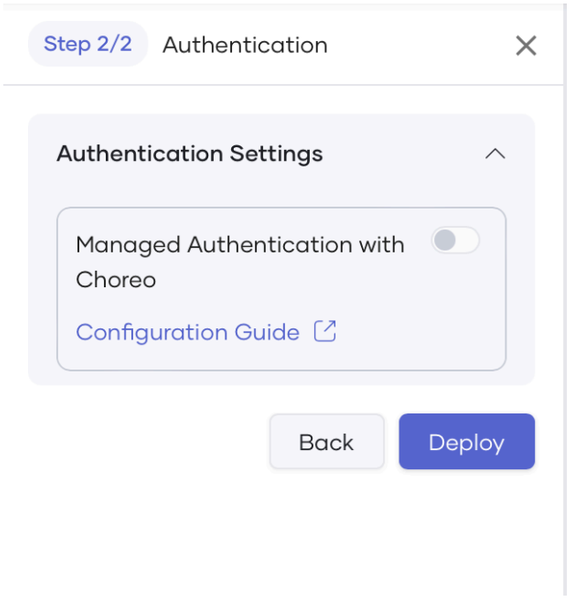
```
Unset
window.configs = {
        apiUrl: '<Service URL>',
};
```

Replace <Service URL> with the value that you copied after creating a connection to the Service in Step 13.
Ex:

```
Unset
window.configs = {
       apiUrl: '/choreo-apis/luxury-hotels/hotel-reservation-service/v1',
};
```

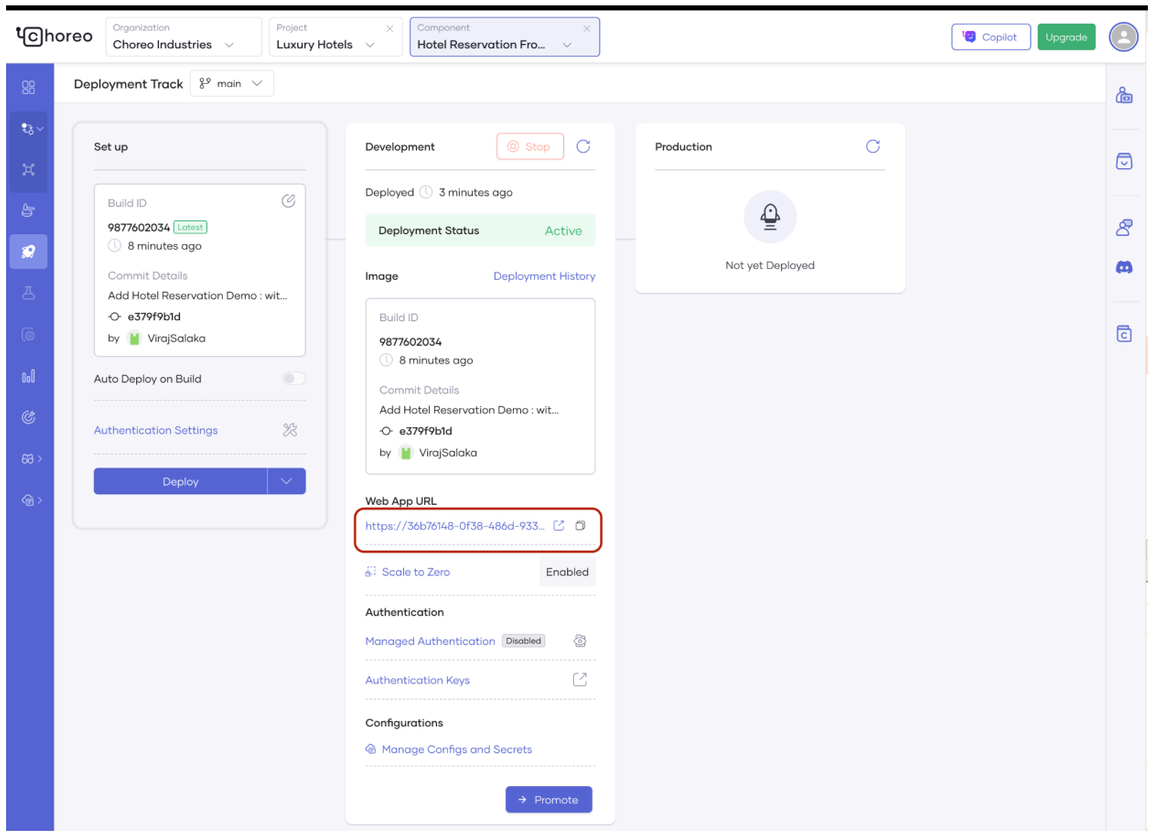
15. Click **Next**.

16. **Disable** Managed Authentication with Choreo (Toggle the button). In our source code, we do not have the necessary implementation to make it work as of now. We will revisit this later.

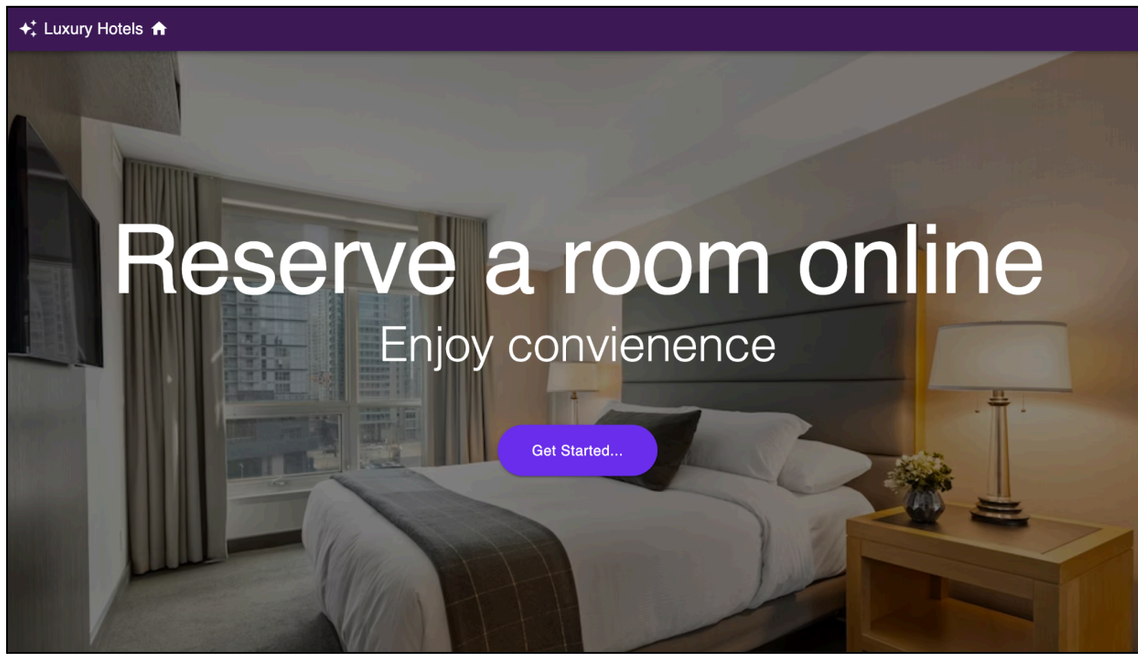


17. Click **Deploy** in the right side panel to deploy the web application.

18. Once the web application is deployed successfully, click on the Web App URL to access the web app.

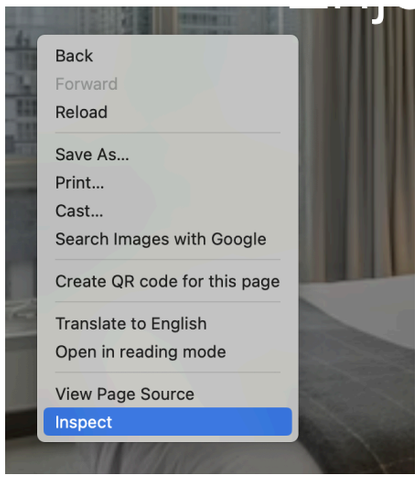


19. You will be greeted with the landing page shown below:

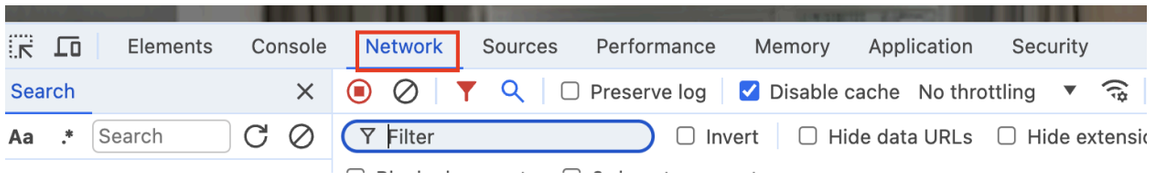


20. Click the **Get Started…** button.

21. Right click and click **inspect**.

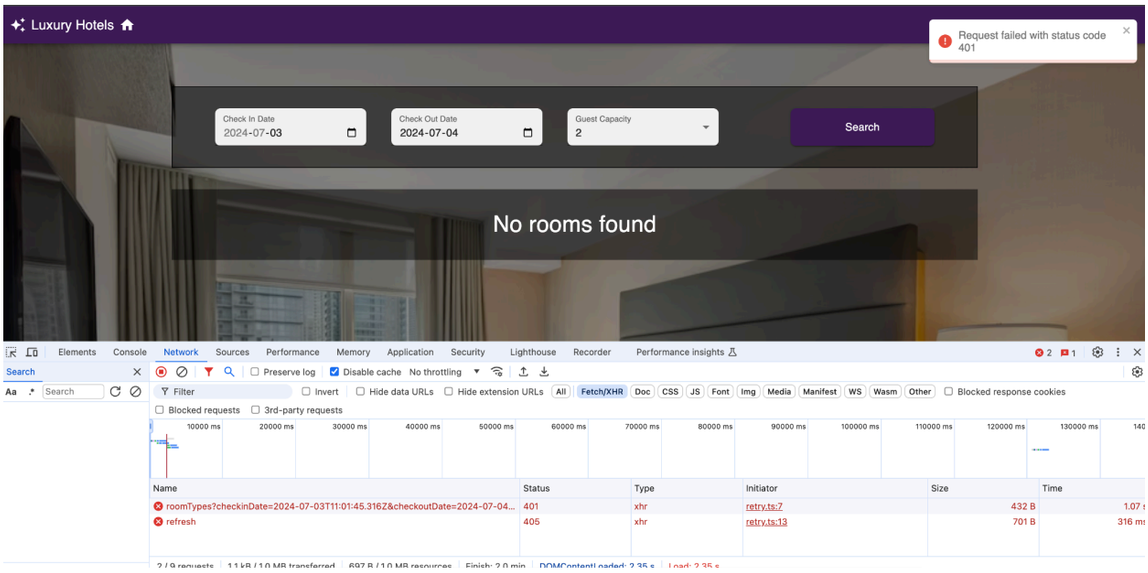


22. Click on the **network** tab in the inspect view. Here we will monitor the network behavior.



23. Click **Get Started** first.

24. Set a check in date, check out date and select the no of guests. Then click **Search**.



The reservation failed because our backend is secured. Our web application requires a valid token for successful invocation.

Step 8 - Add Managed Authentication for the Hotel Reservation Web App

1. Open **Visual Studio Code** and open the cloned repository.

2. Open the Terminal and run the following commands to navigate inside the web application source.

```
Unset
cd hotel-reservation-app-db
```

```
Unset
cd hotel-reservation-frontend
```

3. Run the following command to install node modules.

```
Unset
npm install
```

4. We need to secure our web application. Hence we need to perform login functionality right after the user clicks on **Get Started**. Therefore, we need to do the following change in "**hotel-reservation-app-db/hotel-reservation-frontend/src/pages/landing_page/index.tsx**" file so that it directs to the login page rather than the *rooms* page.

Replace the following code segment:

```
Unset
<Button
        onClick={() => {
          window.location.href = "/rooms";
        }}
        variant="contained"
        color="secondary"
        style={{
          borderRadius: 32,
          textTransform: "none",
          height: 64,
          width: 200,
          fontSize: 18,
        }}
      >
        Get Started...
      </Button>
```

With the following:

```
Unset
<Button
        onClick={() => {
          window.location.href = "/auth/login";
        }}
        variant="contained"
        color="secondary"
        style={{
          borderRadius: 32,
          textTransform: "none",
          height: 64,
          width: 200,
          fontSize: 18,
        }}
      >
        Get Started...
      </Button>
```

5. Add the import statement and the code segment highlighted in green to the “**hotel-reservation-app-db/hotel-reservation-frontend/src/App.tsx”** file in the exact order.

```
Unset
import Cookies from "js-cookie";
```

JavaScript

```
export default function App() {
 const [signedIn, setSignedIn] = useState(false);
 const [user, setUser] = useState<User>({
   email: "",
   id: "",
   name: "",
   mobileNumber: "",
 });
 const [isAuthLoading, setIsAuthLoading] = useState(false);

 function getMappedUser(userInfo: any): User {
   return {
     email: userInfo?.email || "",
     id: userInfo?.sub || "",
     name: userInfo?.first_name + " " + userInfo?.last_name,
     mobileNumber: userInfo?.mobile_number || "",
   };
 }

 useEffect(() => {
   setIsAuthLoading(true);
   if (Cookies.get("userinfo")) {
     // We are here after a login
     const userInfoCookie = Cookies.get("userinfo");
     sessionStorage.setItem("userInfo", userInfoCookie || "");
     Cookies.remove("userinfo");
     var userInfo = userInfoCookie ? JSON.parse(atob(userInfoCookie)) : {};
     setSignedIn(true);
     setUser(getMappedUser(userInfo));
   } else if (sessionStorage.getItem("userInfo")) {
     // We have already logged in
     var userInfo = JSON.parse(atob(sessionStorage.getItem("userInfo")!));
     setSignedIn(true);
     setUser(getMappedUser(userInfo));
   } else {
     console.log("User is not signed in");
     if (
       window.location.pathname !== "/auth/login" &&
       window.location.pathname !== "/"
     ) {
       window.location.pathname = "/auth/login";
     }
   }
   setIsAuthLoading(false);
 }, []);
```

```
if (isAuthLoading) {
  return <div>User authenticating...</div>;
}
```

6. Let's now implement the Logout functionality. Go to "**hotel-reservation-app-db/hotel-reservation-frontend/src/layout/AppBar.tsx**" and add the following import statement and the code segment highlighted in green to add the **Logout** functionality.

```
Unset
import Cookies from "js-cookie";
```

```
Unset
<MenuItem
          onClick={() => {
            sessionStorage.removeItem("userInfo");
            window.location.href =
`/auth/logout?session_hint=${Cookies.get('session_hint')}`;
          }}
        >
          <Button style={{ textTransform: "none" }}>
            <Typography textAlign="center">Logout</Typography>
          </Button>
        </MenuItem>
```

7. We now have all the required code level changes. Next we need to push these changes to the github repository. Run the following set of commands.

```
Unset
git add -A
```

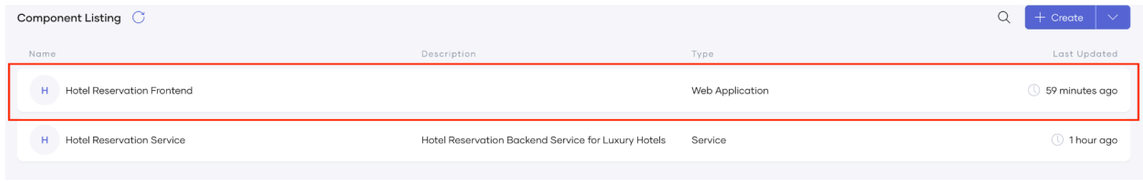
```
Unset
git commit -m "Implement authentication functionality for the web
application"
```

```
Unset
git push origin main
```

Optional:
In case you run into issues with pushing the changes to the remote repository, Follow the github docs to use a personal access token. And when the terminal prompts you asking the password, provide the github personal access token.

8. Visit the github repository and verify that the changes are there.

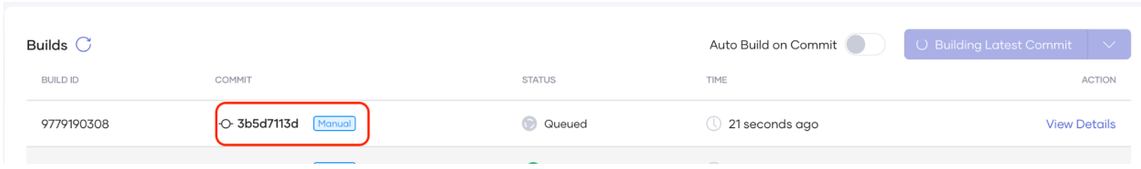
9. Visit the Choreo console. Click **Hotel Reservation Frontend** from the component listing.



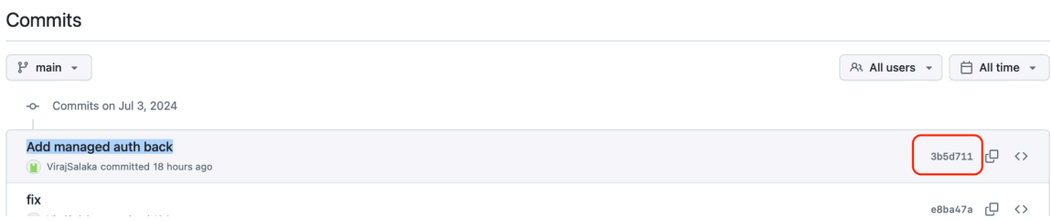
10. In the left navigation menu, click **Build Latest**.

11. While the build is queued, make sure that the latest **commit ID** you have in your github repository's main branch matches with the **commit** listed in the processing record .

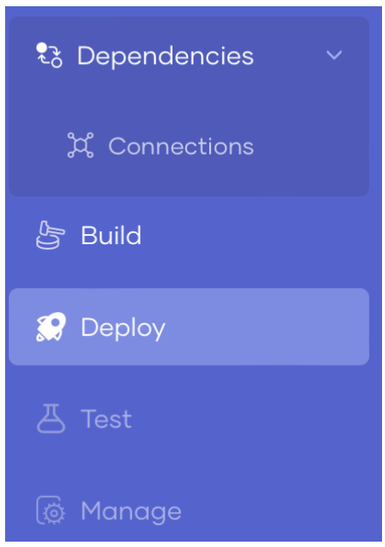
    In choreo:



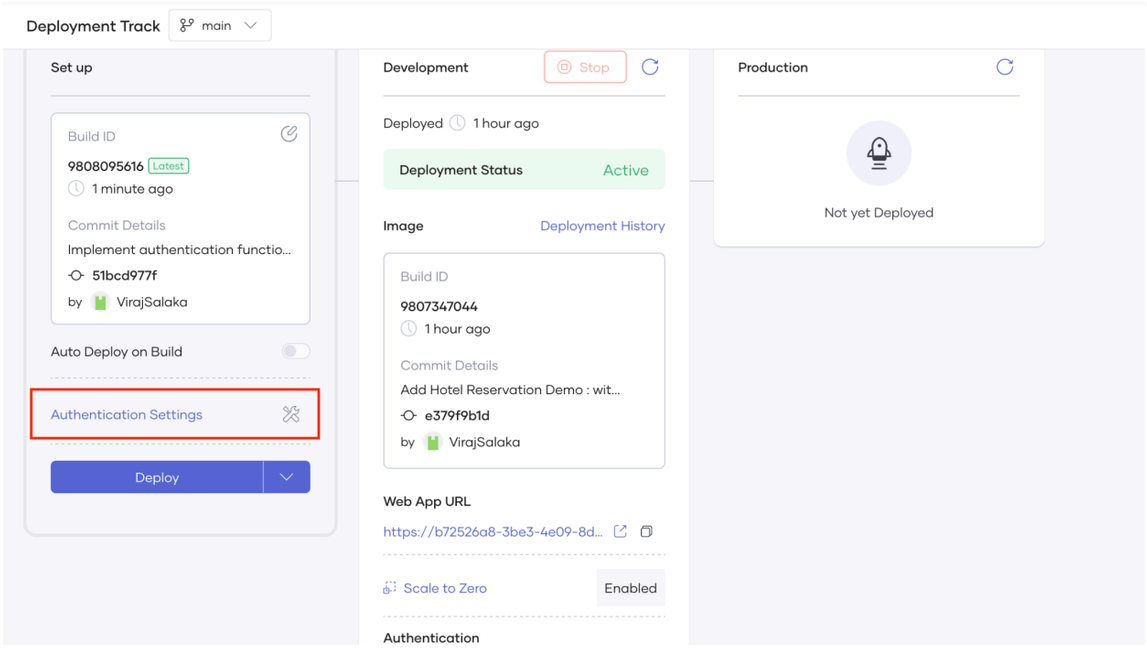
In github commits,



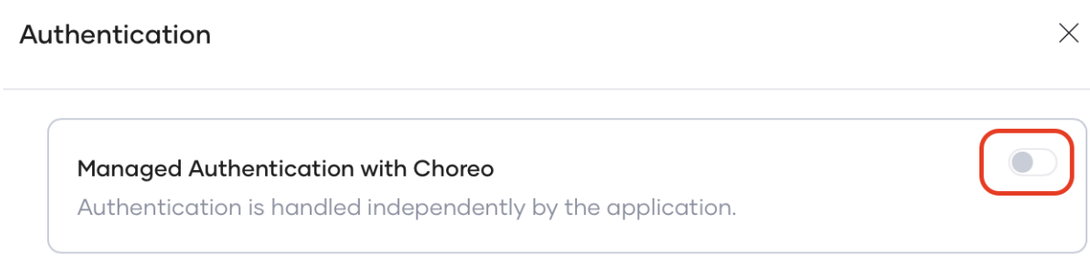
12. Once the build is successful, click **Deploy** on the left navigation menu.



13. In the **Set up** card in the Deploy page , click **Authentication Settings**.



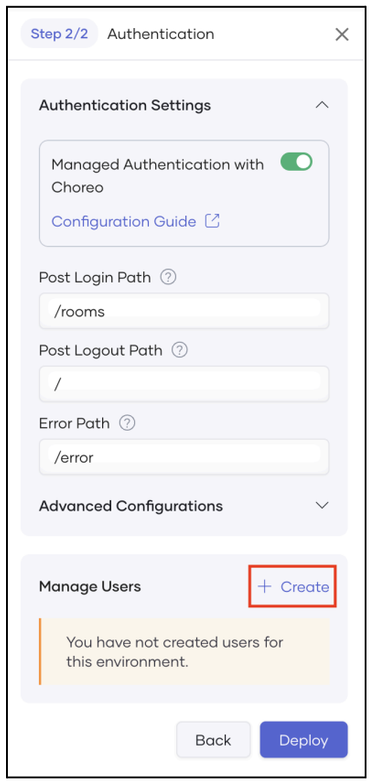
14. Enable Managed Authentication by clicking on the **Manage Authentication with Choreo** toggle button in the right side menu.



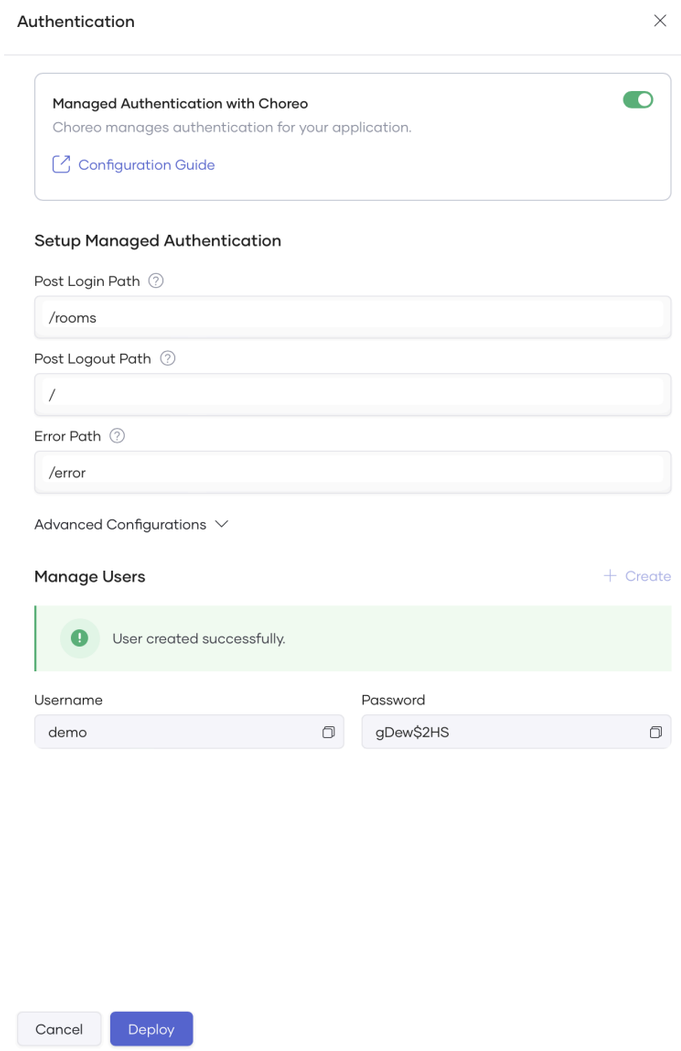
15. Provide following values for each parameter in the Authentication Settings window.

Field Name	Field Value
Post Login Path	rooms
Post Logout Path	/
Error Path	/error

16. Click **Create** to create a demo user.

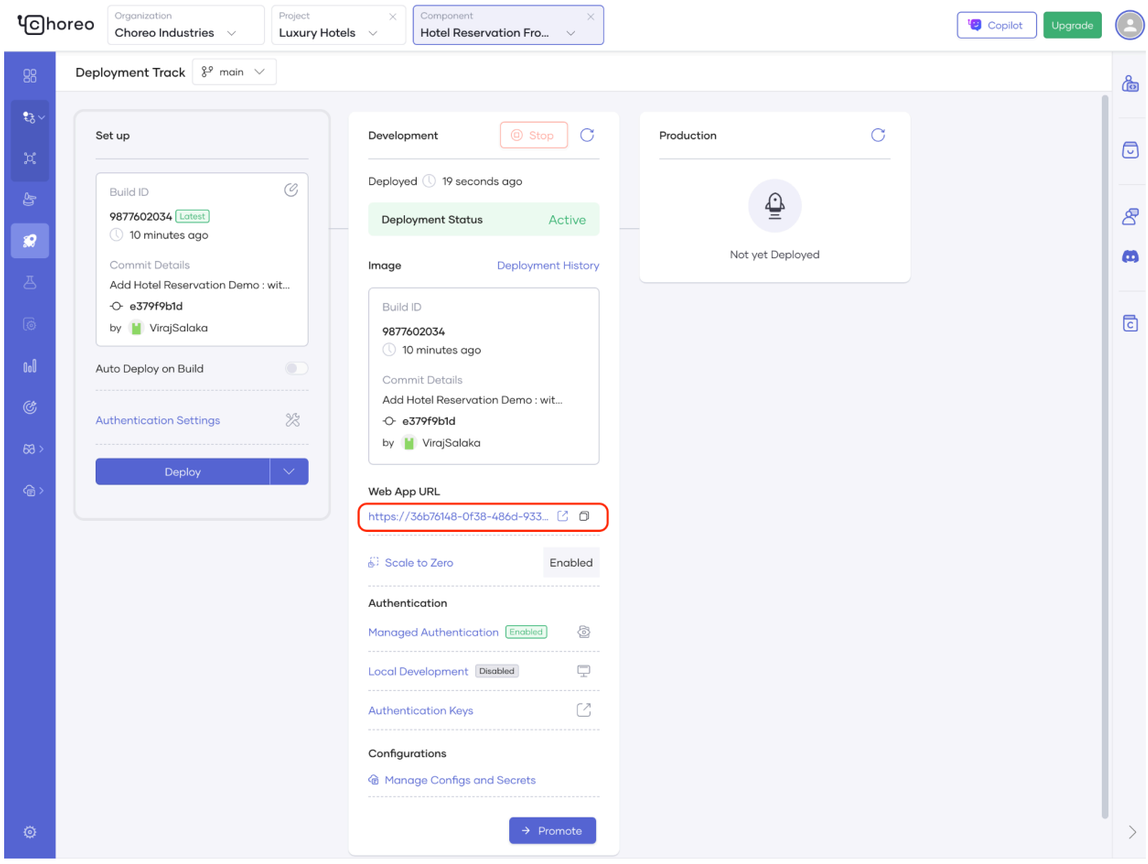


17. Once the user is created, save the username and the password for later use.

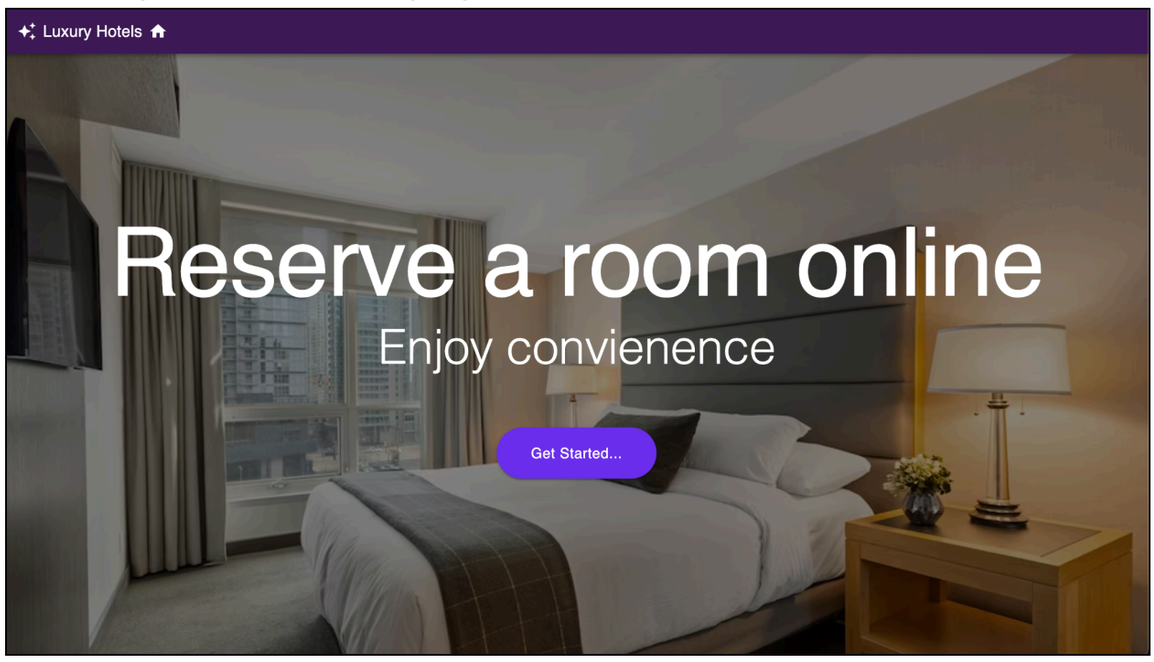


18. Click **Deploy** in the right side panel to deploy the web application.

19. Once the web application is deployed successfully, click on the Web App URL to access the web app.



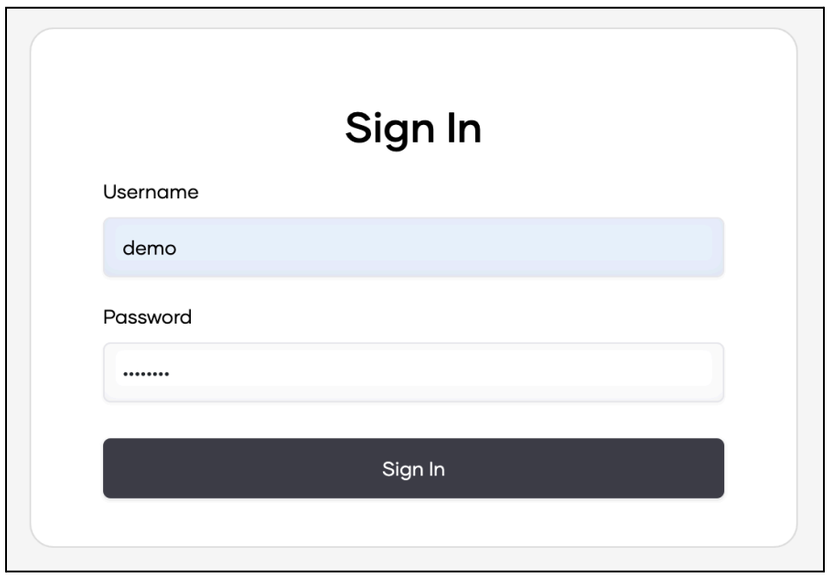
20. You will be greeted with the landing page shown below:



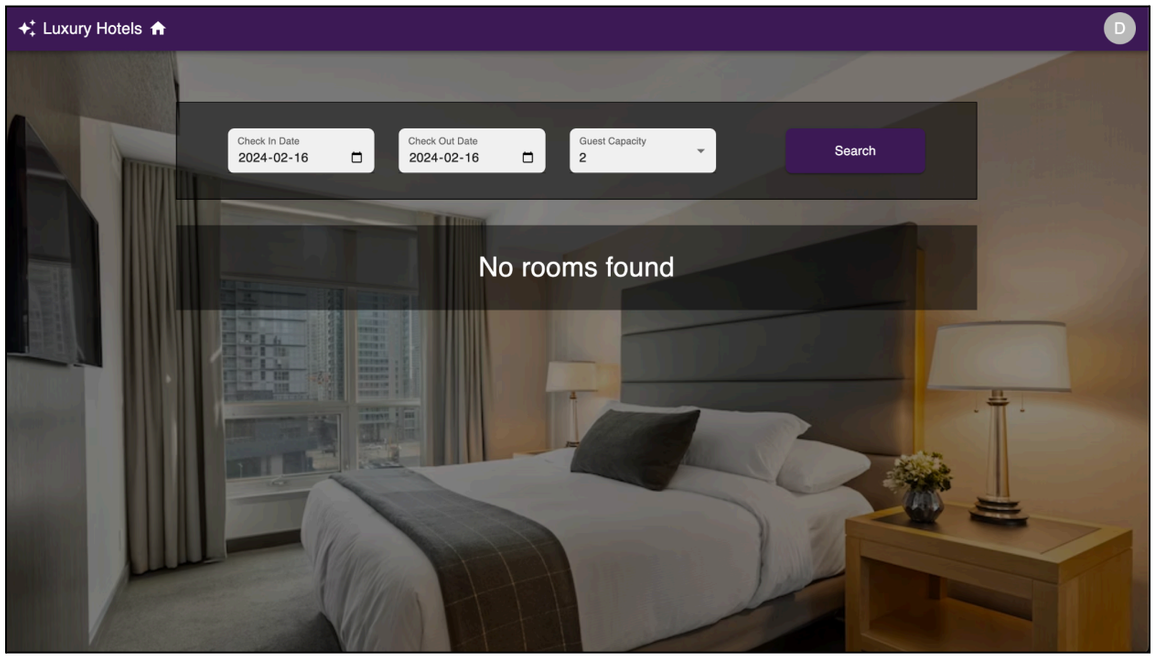
21. Click the **Get Started**… button.

22. This will direct you to the login page. Provide the demo username and password (created in the Step 17) and click **Sign In**.

    Note
    If you forgot to copy the password, you can still retrieve it. Follow the instructions in this document to navigate to the location where you can view the configured user and copy the username and password.

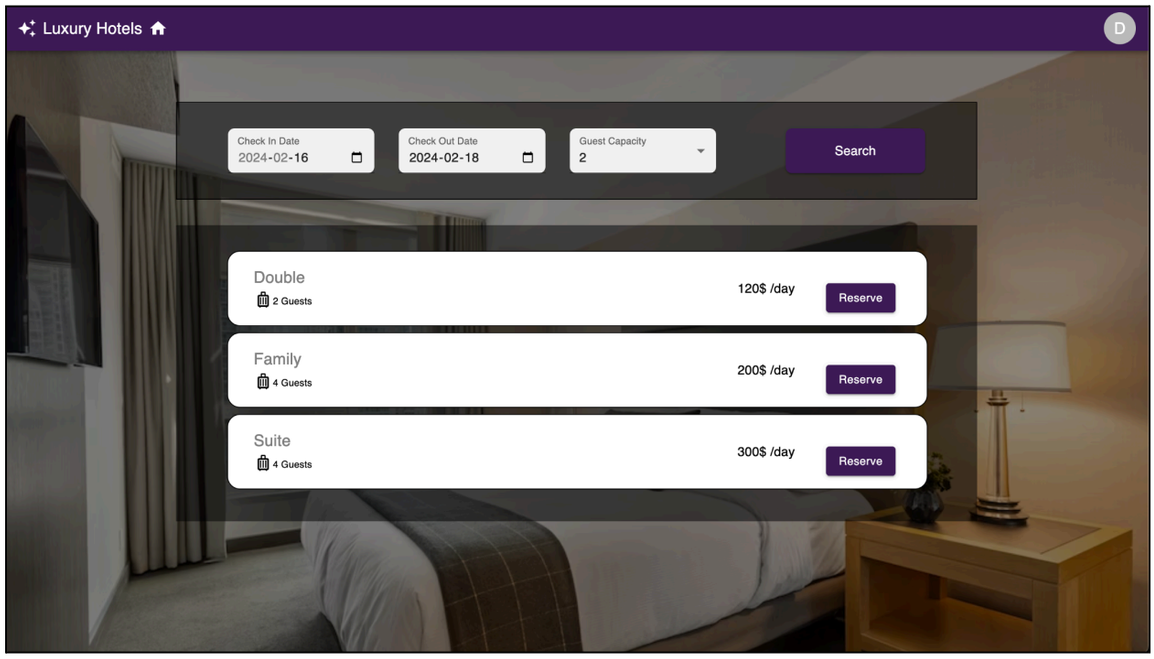


23. You will be logged into the Hotel Reservation web application.



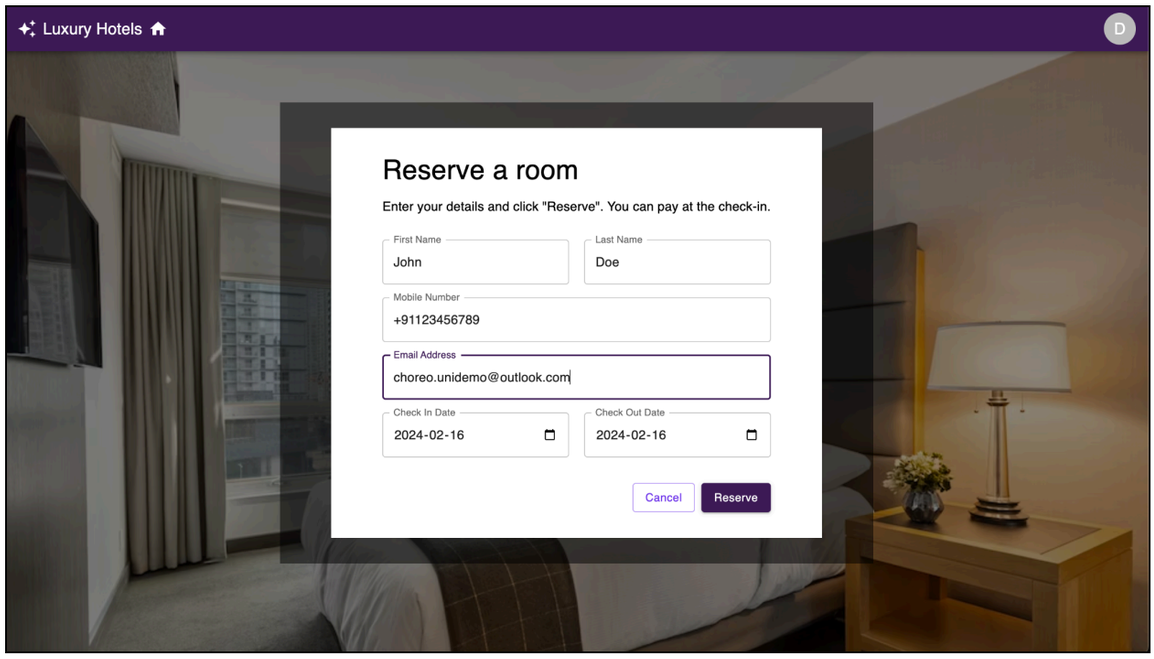
24. Set a check in date, check out date and select the no of guests. Then click **Search**.

25. You will be able to get the available rooms for the selected date range and guest capacity.



26. Click **Reserve**.

27. Provide required details as below and click **Reserve**.



28. Your reservation will be created.