

Building Cloud Native Applications with Choreo

Training Objective:

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

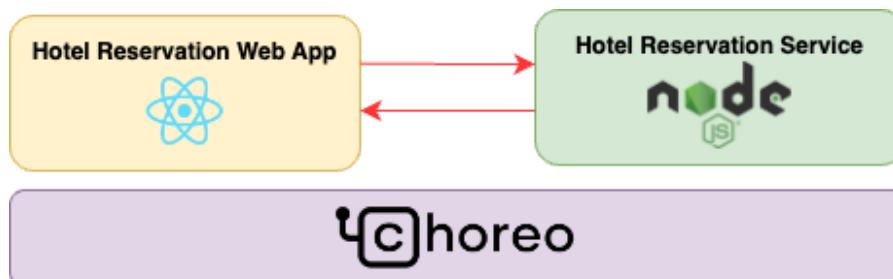
1. Deploy a backend service
2. Deploy a frontend web app
3. Securing the frontend with Choreo Managed Authentication
4. Connecting the backend and frontend
5. Monitoring and troubleshooting

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)

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 web application offers the following features:

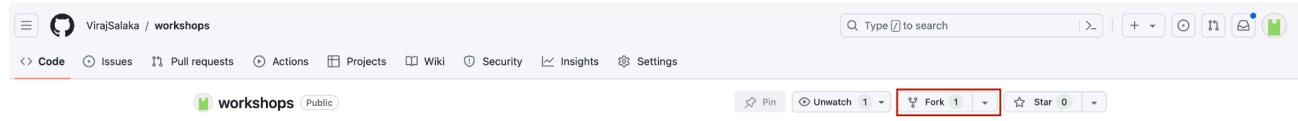
- **Room Search:** Users can search for rooms by specifying check-in and check-out dates, with an option to filter by number of guests. Search results will showcase a list of room types (eg: single, double etc). Each room listing will feature a "Reserve" button for easy booking.
- **Room Reservation:** To reserve a room, users need to input personal information: full name, contact number, and email. Upon 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:

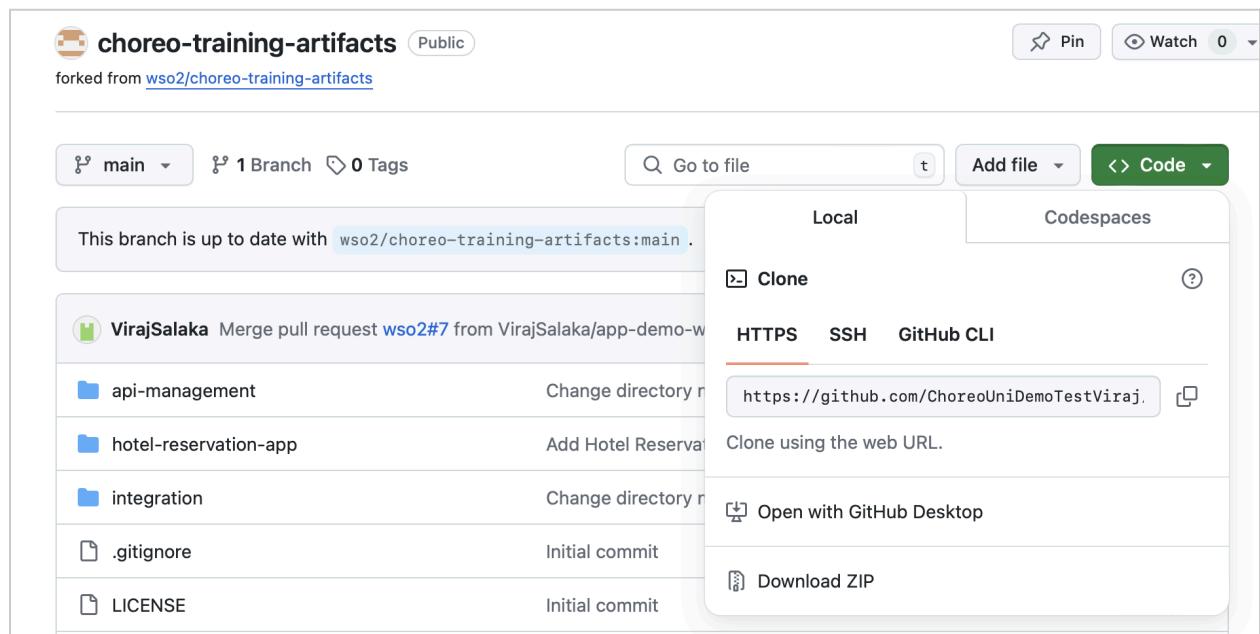
Follow the steps below to deploy the Hotel Reservation System.

Step 1: Fork and Clone the repository

1. Go to <https://github.com/wso2/choreo-training-artifacts> and click on **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 that you copied in the previous step.

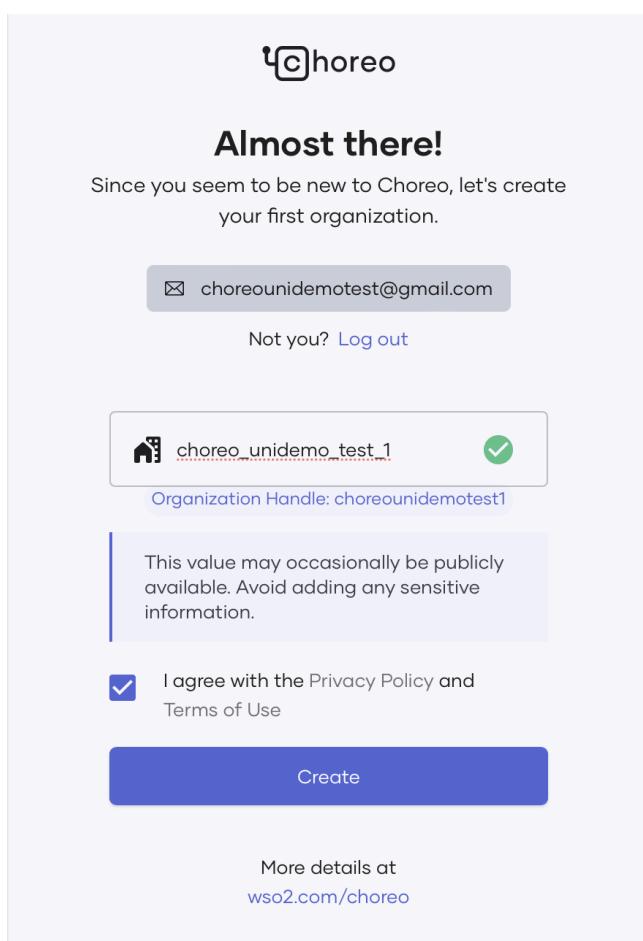
For Windows users: Open git bash and run the command below.

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

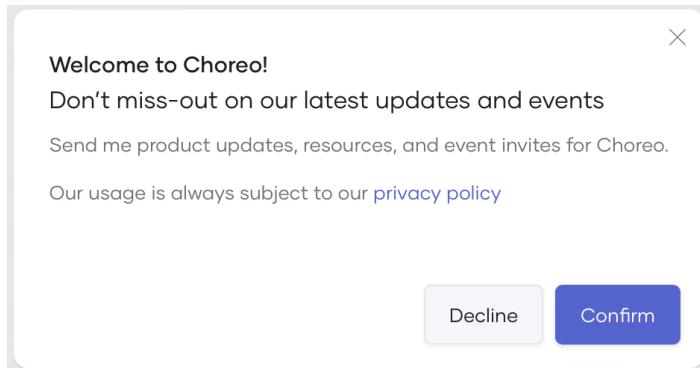
Step 2: Sign Up to Choreo

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**.

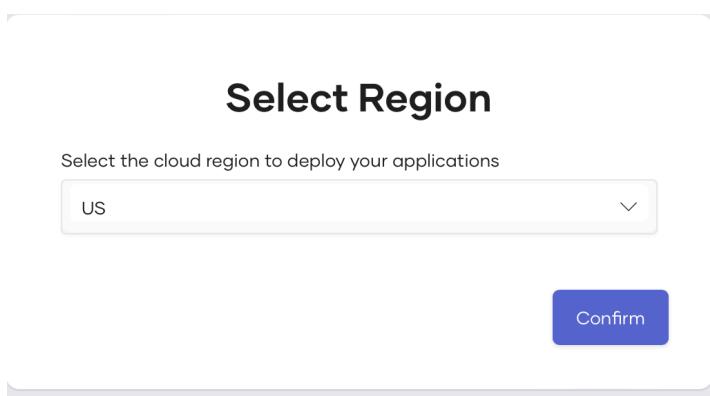
It should be a unique name across choreo. Hence if the provided name is invalidated by platform itself please change the name and retry it.



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

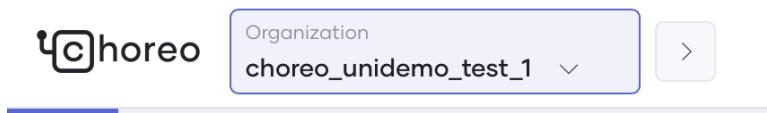


4. In the region selector window, keep the default selected option as US. Then click **Confirm**.

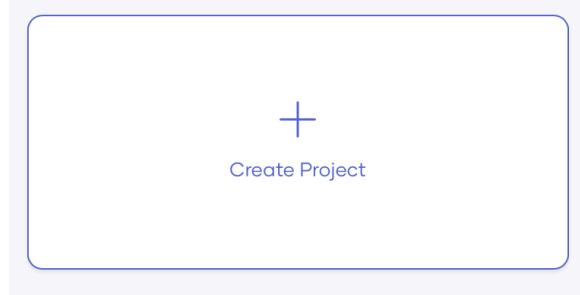


Step 3: Create a new Project

1. Click the Organization card in the top menu.

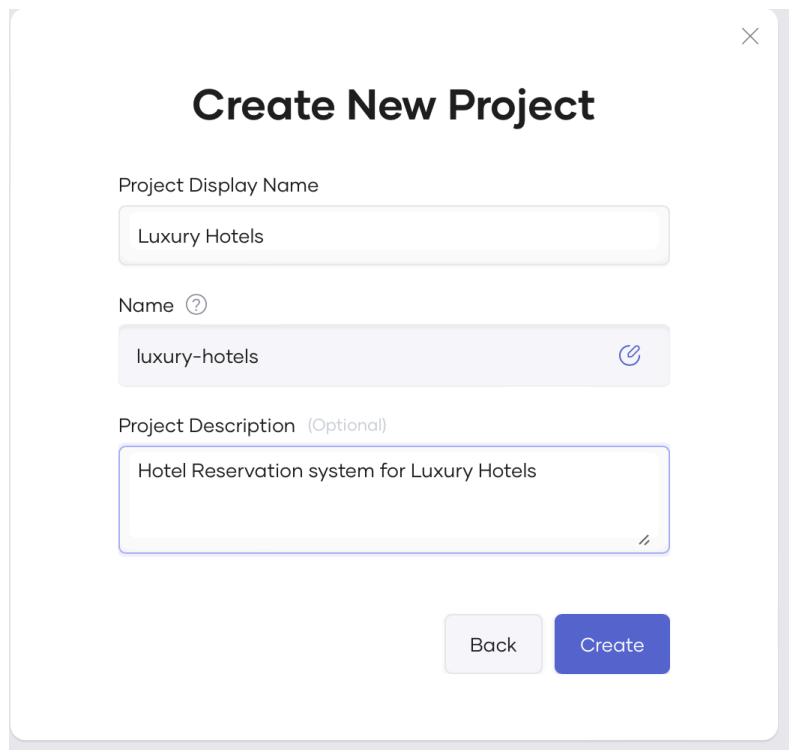


2. Click **Create Project** to create a new project.



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

Field Name	Field Value
Name	Luxury Hotels

A screenshot of a modal dialog titled 'Create New Project'. It contains fields for 'Project Display Name' (Luxury Hotels), 'Name' (luxury-hotels), and 'Project Description (Optional)' (Hotel Reservation system for Luxury Hotels). At the bottom are 'Back' and 'Create' buttons.

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

Step 4: 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**.

The screenshot shows the WSO2 Choreo interface with the title 'Create a New Component'. A red box highlights the 'Service' category, which contains icons for REST, GraphQL, and gRPC. Other categories shown include 'Web Application', 'API Proxy', 'Webhook', 'Scheduled Task', 'Manual Task', 'Event Handler', and 'Test Runner'.

- Provide a name for the service component.

Field Name	Field Value
Component Display Name	Hotel Reservation Service

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

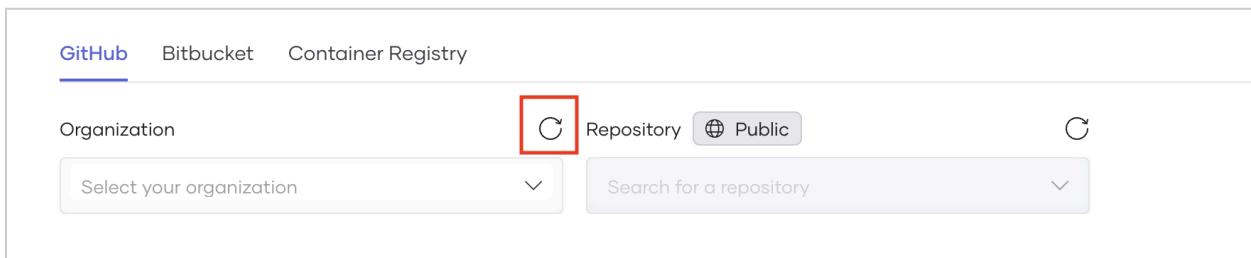
- Expand the Select your organization section and click **+Add**.

The screenshot shows a modal window titled 'Select your organization'. It has tabs for 'GitHub', 'Bitbucket', and 'Container Registry', with 'GitHub' selected. Below the tabs is a dropdown menu with the placeholder 'Select your organization'. At the bottom is a button labeled '+ Add'.

- Select All repositories, then click **Install & Authorize**.

The screenshot shows a dialog box titled 'Install & Authorize Choreo.dev'. It includes a heading 'Install & Authorize on your personal account' with a user icon, a section for selecting repositories ('for these repositories:'), and a radio button for 'All repositories' with a note about applying to all future repositories owned by the resource owner. It also lists permissions ('with these permissions:'), including 'Read access to issues and metadata' and 'Read and write access to code, pull requests, and repository hooks'. At the bottom are 'Install & Authorize' and 'Cancel' buttons, with a note that the user will be redirected to <https://console.choreo.dev/ghapp>.

- Click the refresh icon next to Organization.

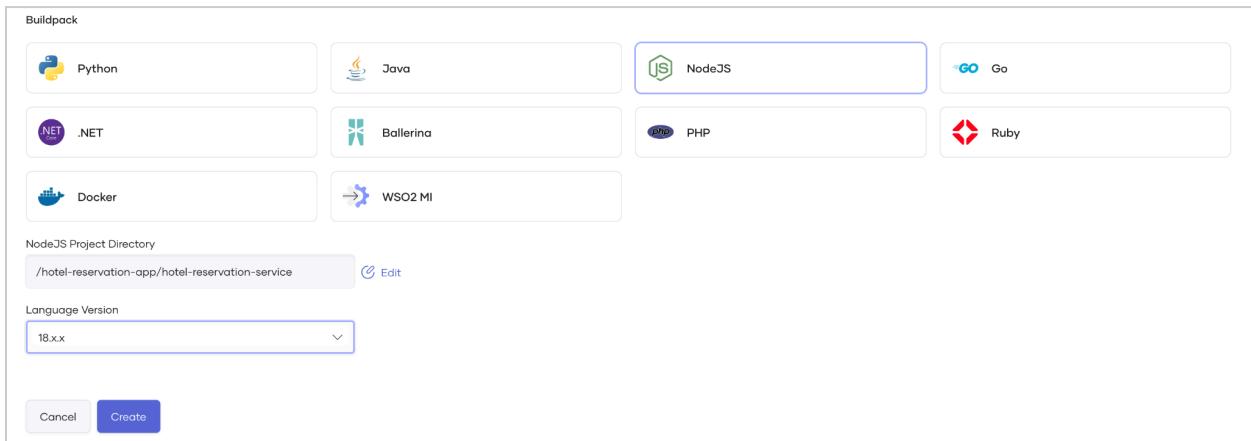


- Select the relevant organization and select the **choreo-training-artifacts** repository. The branch will be automatically selected as main.



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

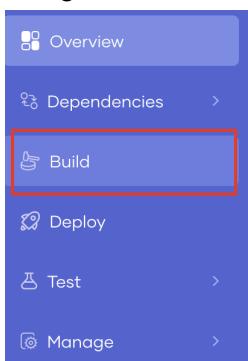
Field Name	Field Value
NodeJS Project Directory	/hotel-reservation-app/hotel-reservation-service
Language Version	18.x.x



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

Name	Description	Type	Last Updated
Hotel Reservation Service	Hotel Reservation Backend Service for Luxury Hotels	Service	39 seconds ago

- Now that the component is successfully created the 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.

The screenshot shows the 'Builds' section of the WSO2 UI. At the top right, there are two buttons: 'Auto Build on Commit' (unchecked) and 'Build Latest' (highlighted in blue). Below the table, a message says 'No Builds Available'.

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

The screenshot shows the 'Builds' section after a build has been completed. The last entry in the table has a green checkmark icon next to 'Status' and the word 'Success'. The 'Time' column shows '14 seconds ago' and the 'View Details' link is visible.

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

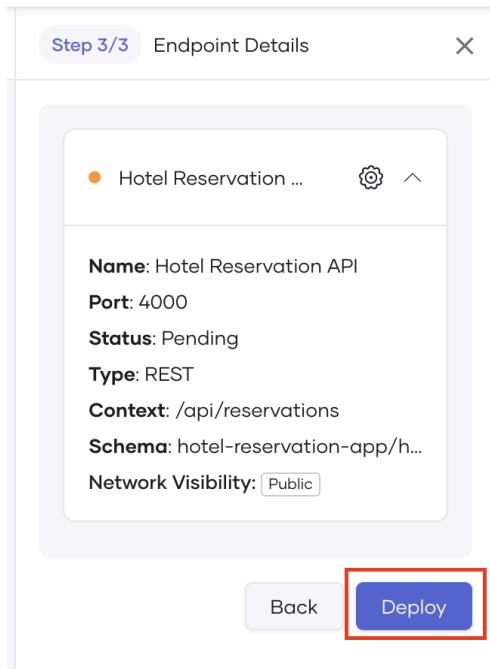
The screenshot shows the left navigation menu with five options: Overview, Dependencies, Build, Deploy, and Test. The 'Deploy' option is highlighted with a red box.

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

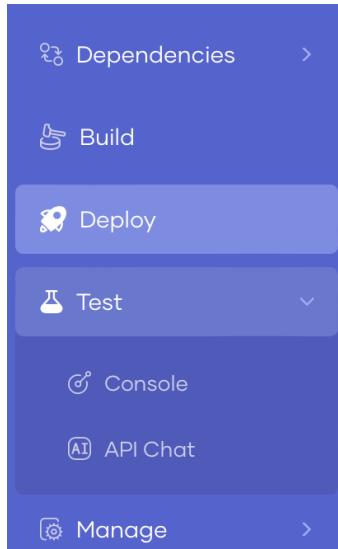
The screenshot shows the 'Deploy' page. At the top, there are three dropdown menus: 'Organization' (choreo), 'Project' (Luxury Hotels), and 'Component' (Hotel Reservation Ser...). The main area is divided into three tabs: 'Setup' (selected), 'Development', and 'Production'. The 'Setup' tab contains sections for 'Build ID' (9806922982 [Latest]), 'Commit Details' (Merge pull request #7 from Viraj...), and 'Auto Deploy on Build' (unchecked). The 'Configure & Deploy' button is highlighted with a red box at the bottom of the setup section.

14. Click the **Next** button in the right side panel for environment configurations and file mounts since we don't have any configurations to provide for this service.

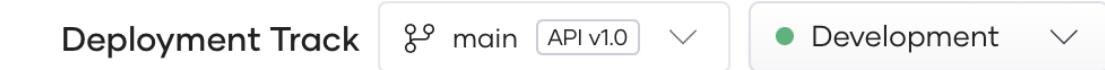
15. Click **Deploy** in the right side panel to deploy the service.



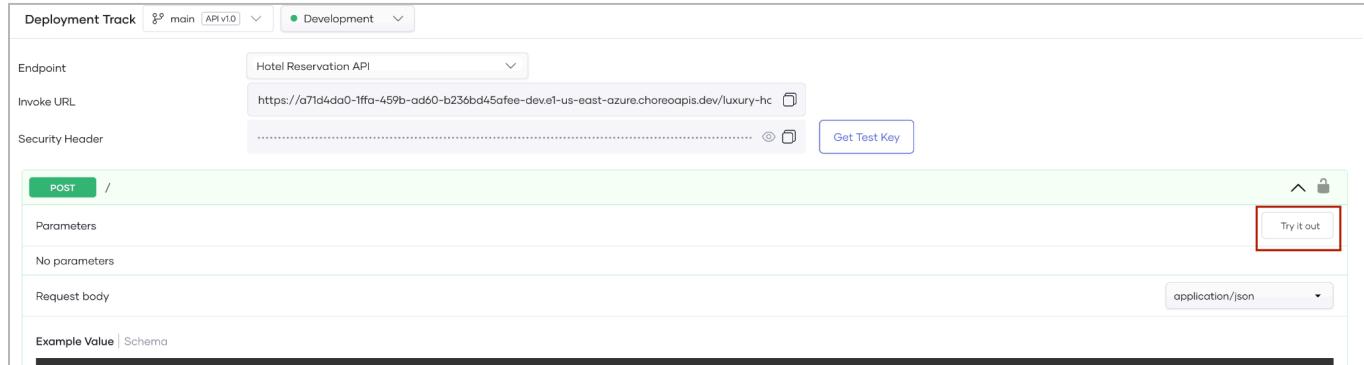
16. Navigate to **Test** in the left navigation menu and expand it. Click **Console** to access the Open API Console to test the service.



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



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



19. 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": 100, "user": { "id": "123", "name": "testuser", "email": "testuser@someemail.com", "mobileNumber": "911234567821" }, "roomType": "Family" }

20. Click **Execute**.

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

```

200
Undocumented

Response body
{
  "id": "e386e7be-075c-4922-93ea-fbf603ab21ee",
  "user": {
    "id": "123",
    "name": "testuser",
    "email": "testuser@someemail.com",
    "mobileNumber": "911234567821"
  },
  "room": 403,
  "checkinDate": "2024-02-19T14:00:00Z",
  "checkoutDate": "2024-02-20T10:00:00Z"
}

Response headers
content-length: 232
content-type: application/json; charset=utf-8

```

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

23. 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

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

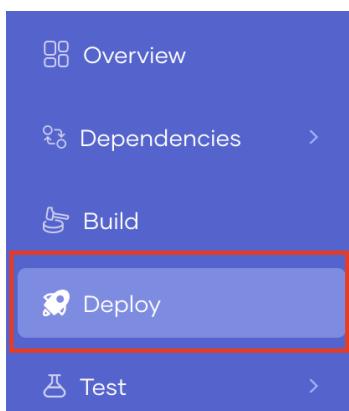
```

Server response
CODE DETAILS
200 Response body
[
  {
    "id": 1,
    "name": "Double",
    "guestCapacity": 2,
    "price": 120
  },
  {
    "id": 3,
    "name": "Suite",
    "guestCapacity": 4,
    "price": 300
  }
]

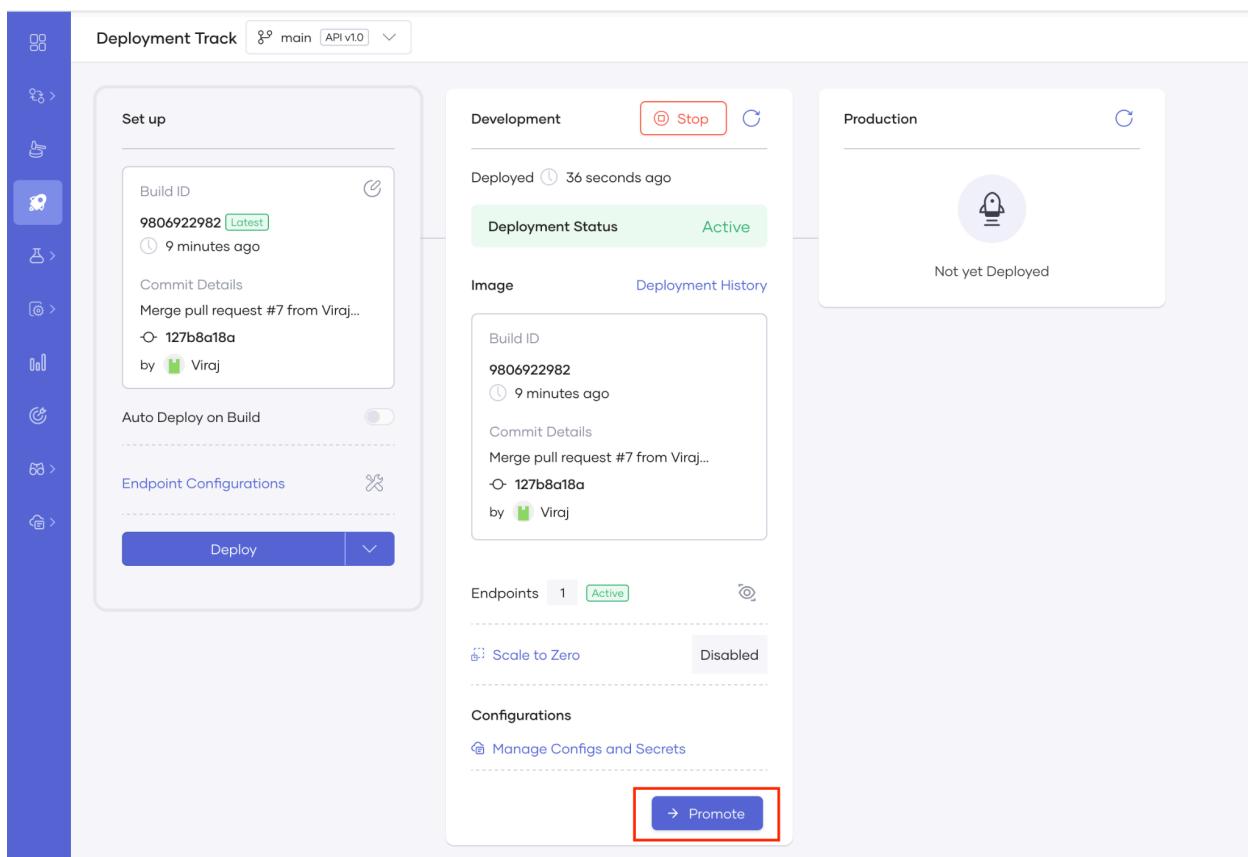
Response headers
content-length: 110
content-type: application/json; charset=utf-8

```

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

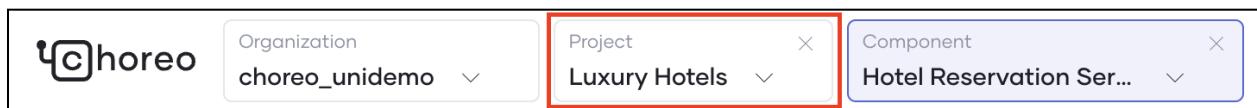


26. Click **Promote** to promote the build to the Production environment. Pick **Next** in all the pop ups and finally click on **promote** on the right side panel.



Step 5: Create and Deploy the Hotel Reservation Web Application

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



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



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

[← Back to Project Home](#)

Create a New Component

Select a Type [Try a Sample](#)

Service	Web Application	API Proxy	Webhook
[REST] [RPC] [GRPC]	HTML [Node.js] [PHP]	[REST]	[GitHub] [Android] [iOS] [Google Sheets]
Scheduled Task	Manual Task	Event Handler	Test Runner
[Docker] [cron]	[Docker]	[Apache Beam] [Apache Flink] [Apache Nifi]	[JUnit] [Cucumber] [Apache Qpid Proton]

- Provide a name for the Web Application.

Field Name	Field Value
Name	Hotel Reservation Frontend

- Click on **Authorize With GitHub**.

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

GitHub Bitbucket Container Registry

Connect Your Repository

Organization: ChereoUniDemoTestViraj Repository: choreo-training-artifacts Branch: main

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

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

Buildpack

NodeJS	React	Angular	.NET
Vue.js	Python	Spring Boot	PHP
Go	Ruby	Static Website	Docker

Project Directory: /hotel-reservation-app/hotel-reservation-frontend [Edit](#)

Build Command: npm run build

Build Path: /build

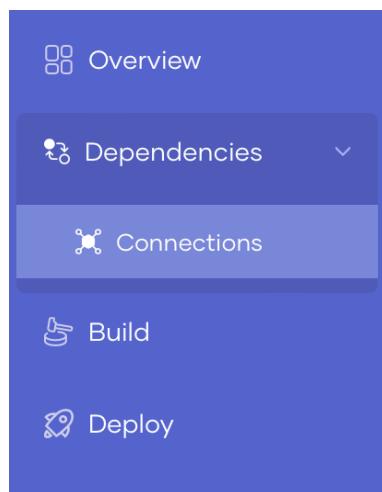
Node Version: 20

[Cancel](#) [Create](#)

8. Click **Build** on the left navigation menu. Click the **Build Latest**. Then the following view will appear.

BUILD ID	COMMIT	STATUS	TIME	ACTION
9807347044	e379f9b1d (Manual)	In Progress	17 seconds ago	View Details

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.

[← Back to Connection Listing](#)

Select a Service

Type	Network Visibility
<input type="checkbox"/> Internal	<input checked="" type="checkbox"/> Organization
<input type="checkbox"/> Third Party	<input checked="" type="checkbox"/> Public
Categories	

H

Hotel Reservation Service
Luxury Hotels
REST
Version: v1 Status: Published

Hotel Reservation Backend Service for Luxury Hotels

Public 2 days ago

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.

The screenshot shows the 'Hotel Reservation Connection' configuration page. It includes fields for 'Connecting to' (Hotel Reservation Service), 'Connection Schema' (Default OAuth Connection - Public), and 'Service URL' (/choreo-apis/luxury-hotels/hotel-reservation-serv). The 'Service URL' field is highlighted with a red border.

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',
};
```

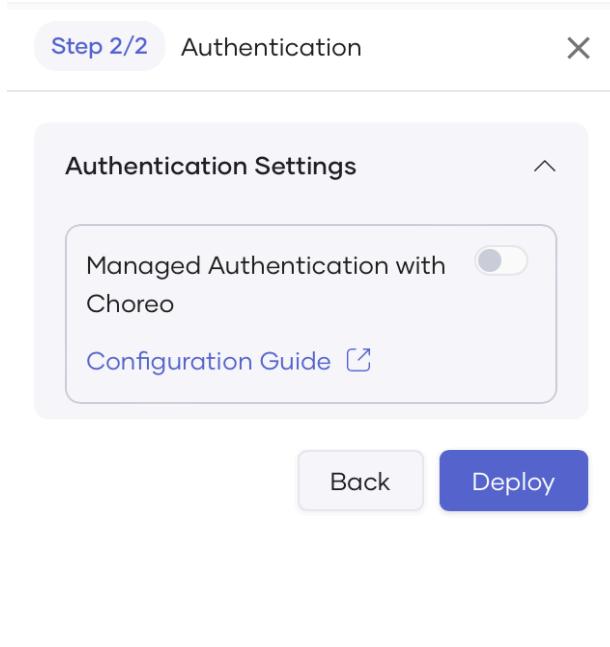
The screenshot shows the 'Step 1/2 Configure & Deploy' dialog. It has a 'File Mount' section where 'config.js' is selected. The file content is displayed as:

```
1 window.configs = {
2     apiUrl: '/choreo-apis/luxury-hotels/hotel-reservation-service/v1',
3 };
```

At the bottom, there is a note: 'Refer this configuration file in your Web App as: ./public/<configuration filename>'.

15. Click **Next**.

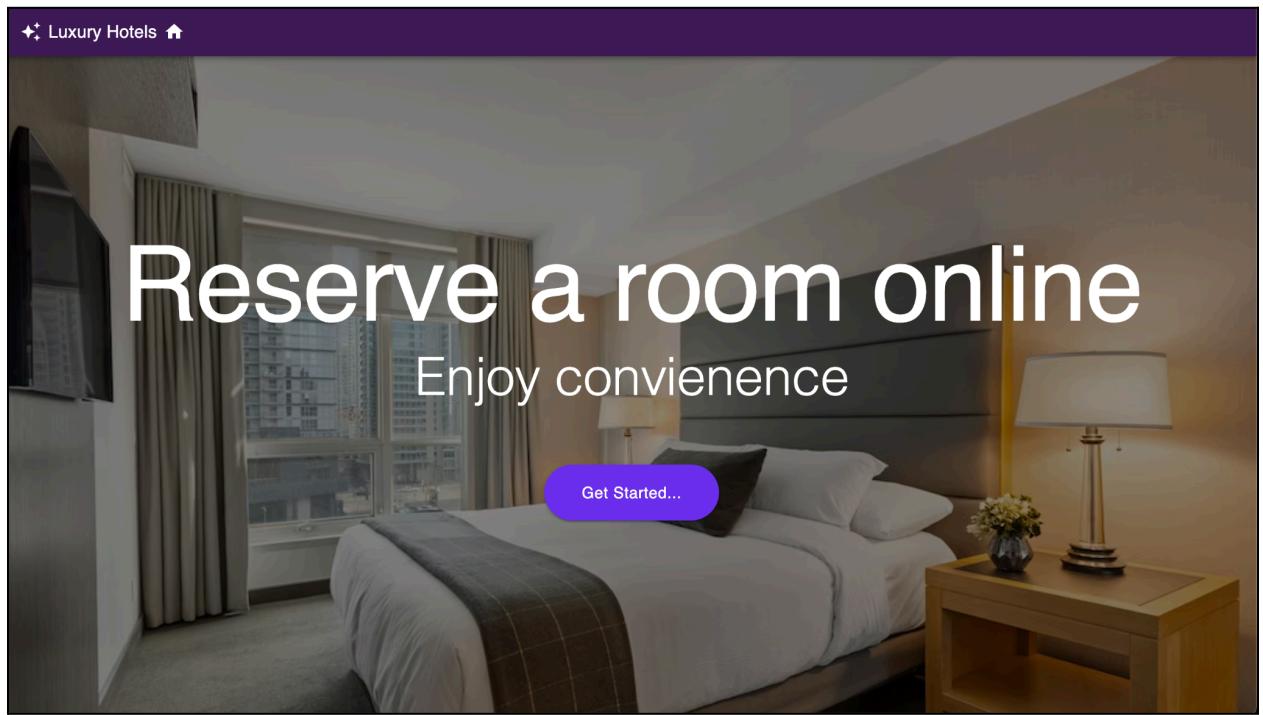
16. **Disable Managed Authentication with Choreo** (Toggle the button). In our source code, we do not have the necessary implementation to make it work as of now. We will revisit this later.



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

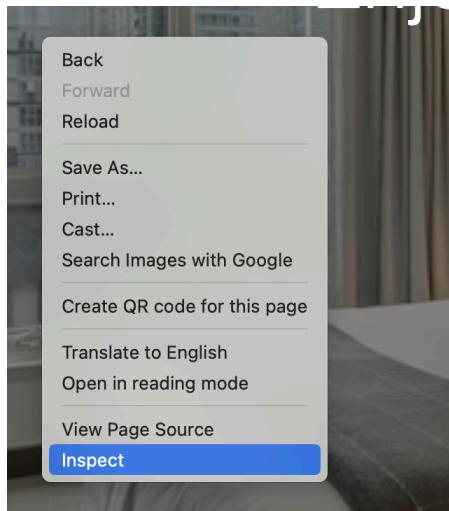
18. Once the web application is deployed successfully, click on the Web App URL to access the web app.

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

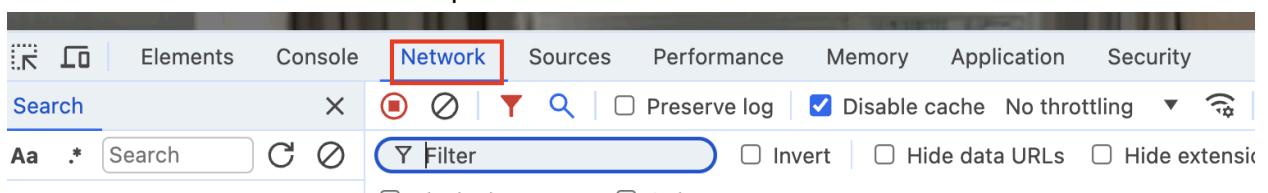


20. Click the **Get Started...** button.

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

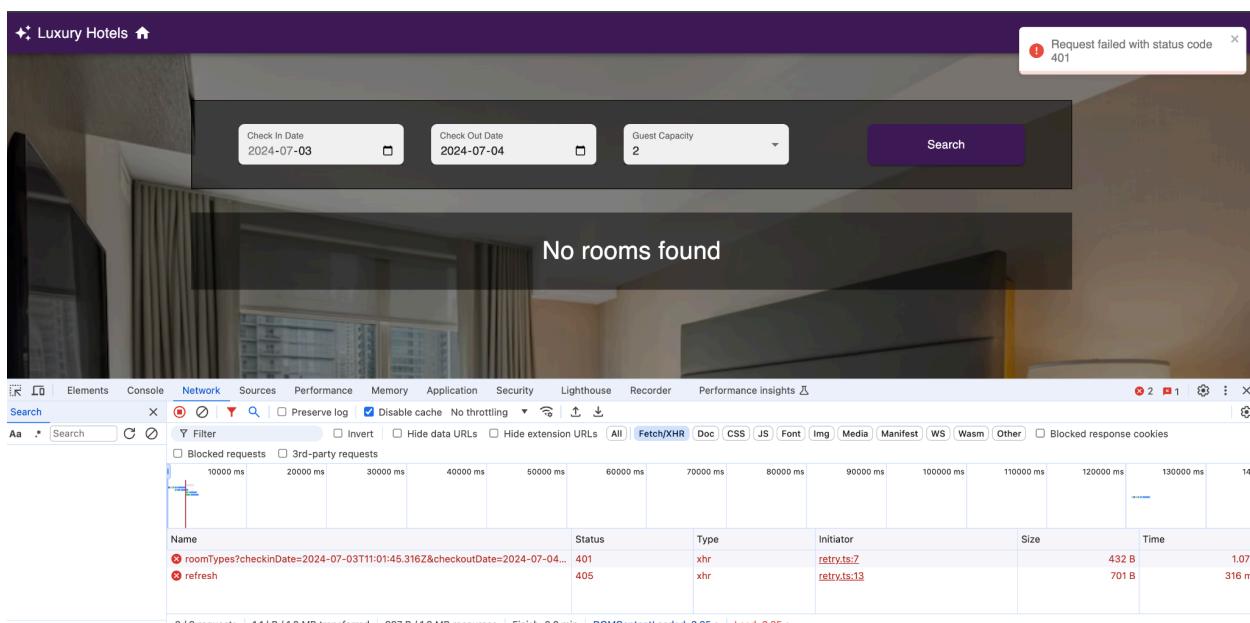


22. Click on the **network** tab in the inspect view. Here we will monitor the network behavior.



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 6 - 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
```

```
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/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. Let's now implement the Logout functionality. Go to **"hotel-reservation-app/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={{ transform: "none" }}>
        <Typography textAlign="center">Logout</Typography>
    </Button>
</MenuItem>
```

6. Add the import statement and the code segment highlighted in green to the “**hotel-reservation-app/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>;
}
```

- 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.

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

- Visit the [Choreo console](#). Click **Hotel Reservation Frontend** from the component listing.

Name	Description	Type	Last Updated
H Hotel Reservation Frontend		Web Application	59 minutes ago
H Hotel Reservation Service	Hotel Reservation Backend Service for Luxury Hotels	Service	1 hour ago

- In the Left Navigation Menu, Click **Build Latest**.

Builds	Auto Build on Commit	Build Latest
	<input type="checkbox"/>	<button>Build Latest</button>

- 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:

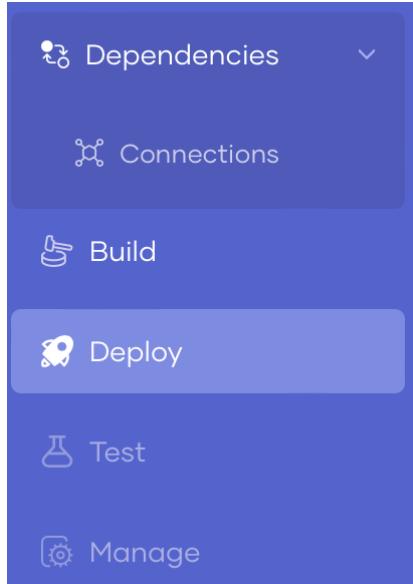
Builds	Commit	Status	Time	Action
9779190308	3b5d7113d Manual	Queued	21 seconds ago	View Details

In github commits,

Commits

The screenshot shows the 'Commits' section of the WSO2 Dev Studio interface. It displays a list of commits for the 'main' branch. At the top, there are filters for 'All users' and 'All time'. A commit from 'VirajSalaka' is highlighted with a red box around the commit ID '3b5d711'. The commit message is 'Add managed auth back' and it was committed 18 hours ago. Below this, another commit 'fix' is shown with the ID 'e8ba47a'.

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**.

The screenshot shows the 'Deployment Track' page. On the left, the 'Set up' card is visible, which includes a 'Build ID' section with details like '9808095616 (Latest)' and '1 minute ago', and an 'Authentication Settings' button highlighted with a red box. Below this is a 'Deploy' button. The 'Development' and 'Production' sections show deployment status: 'Development' is 'Deployed 1 hour ago' with 'Active' status, and 'Production' is 'Not yet Deployed'. The 'Web App URL' is listed as <https://b72526a8-3be3-4e09-8d...>. A 'Scale to Zero' toggle is set to 'Enabled'.

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

Authentication

X

Managed Authentication with Choreo

Authentication is handled independently by the application.



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.

Step 2/2 Authentication

Authentication Settings

Managed Authentication with Choreo

Configuration Guide ↗

Post Login Path ⓘ /rooms

Post Logout Path ⓘ /

Error Path ⓘ /error

Advanced Configurations ▾

Manage Users + Create

You have not created users for this environment.

Back Deploy

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

Authentication

Managed Authentication with Choreo
Choreo manages authentication for your application.

[Configuration Guide](#)

Setup Managed Authentication

Post Login Path [?](#)
/rooms

Post Logout Path [?](#)
/

Error Path [?](#)
/error

Advanced Configurations [▼](#)

Manage Users [+ Create](#)

User created successfully.

Username: demo Password: gDew\$2HS

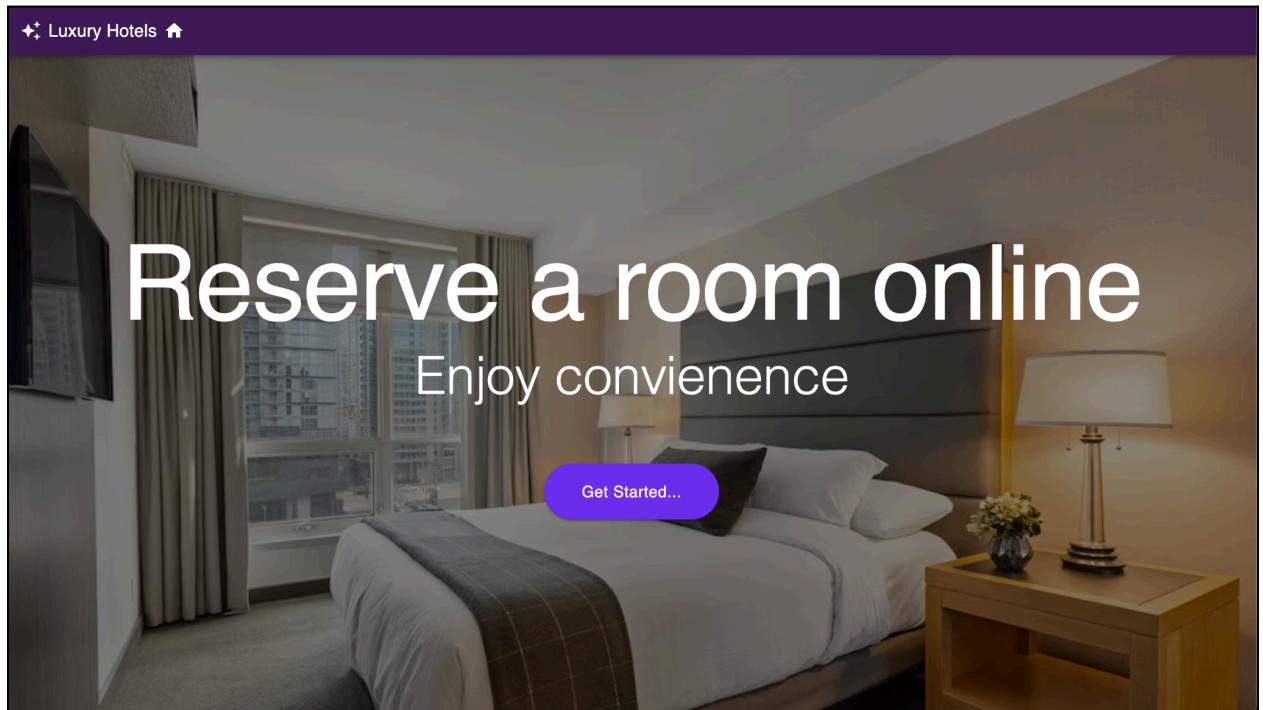
[Cancel](#) [Deploy](#)

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.

The screenshot shows the WSO2 Choreo deployment interface. At the top, there are three dropdown menus: Organization (choreo_unidemo_test_1), Project (Luxury Hotels), and Component (Hotel Reservation Fro...). On the left, a sidebar contains icons for deployment, monitoring, logs, and metrics. The main area is titled "Deployment Track" and shows two environments: "Development" and "Production". In the Development section, a deployment for Build ID 9808095616 (Latest) was deployed 36 seconds ago. In the Production section, the status is "Not yet Deployed". Below the environments, the "Web App URL" field is highlighted with a red box, containing the URL <https://b72526a8-3be3-4e09-8d...>.

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

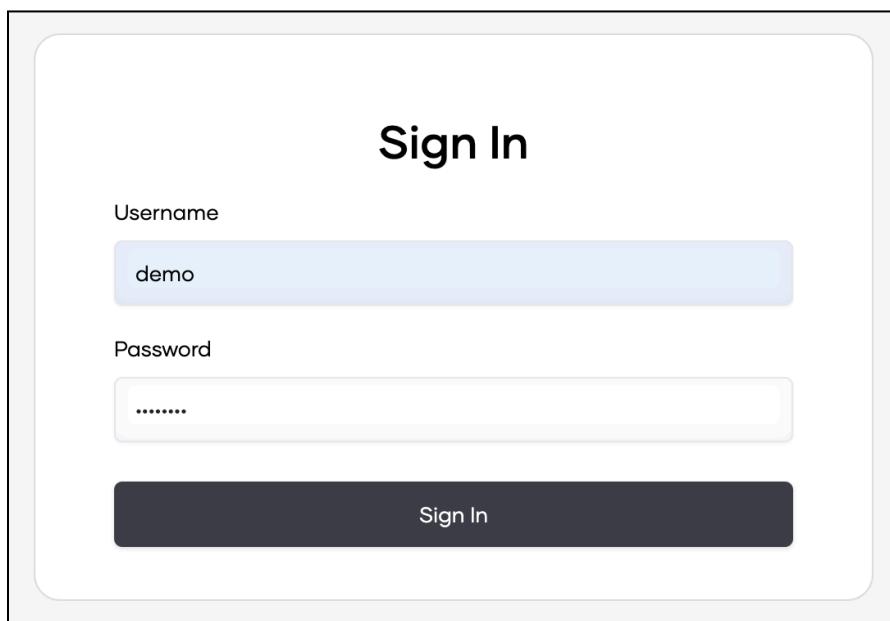


21. Click the **Get Started...** button.

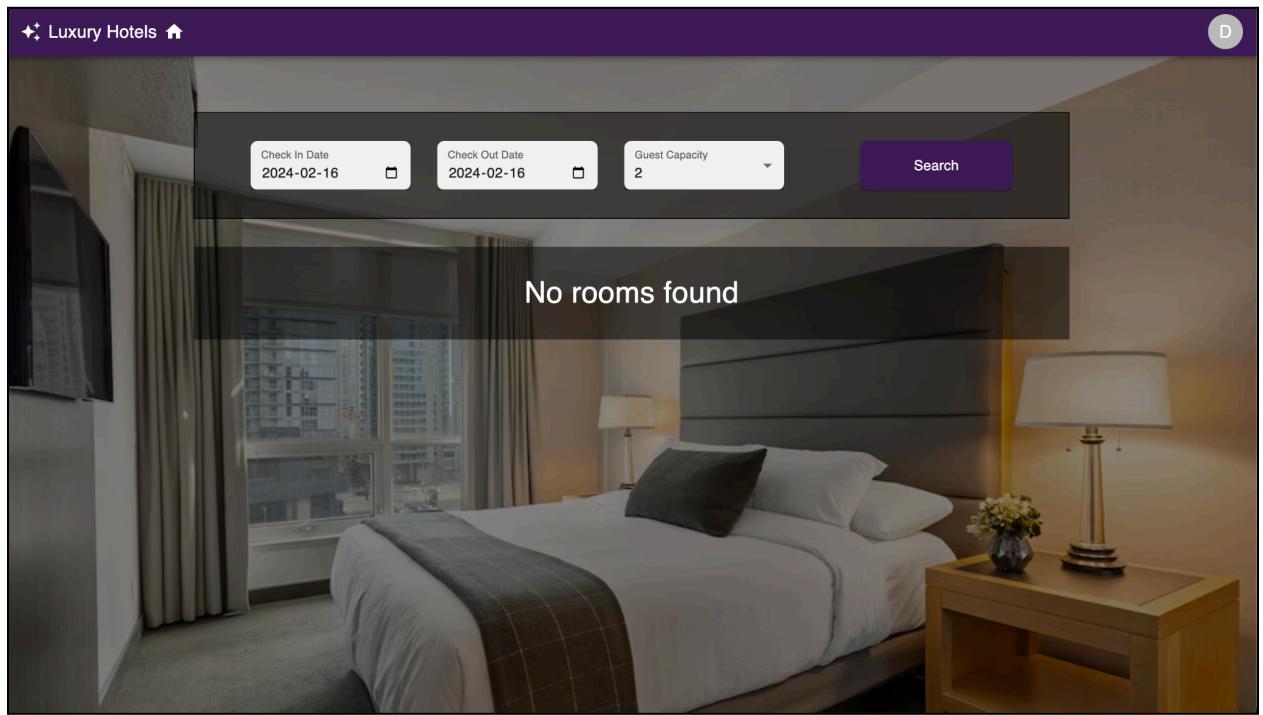
22. This will direct you to the login page. Provide the demo username and password (created in the Step 17) and click **Sign In**.

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.

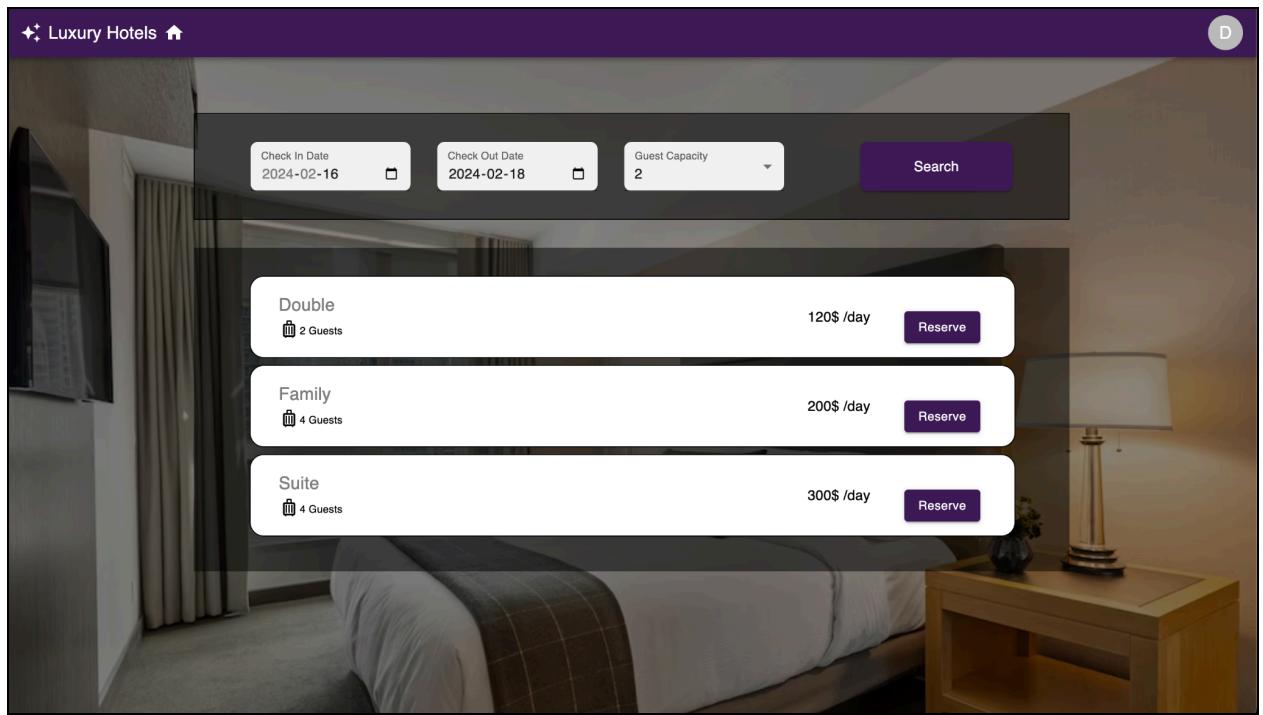
A screenshot of a sign-in form. The title "Sign In" is centered at the top. Below it is a "Username" field containing "demo". Below that is a "Password" field with four dots visible. At the bottom is a dark grey "Sign In" button.

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



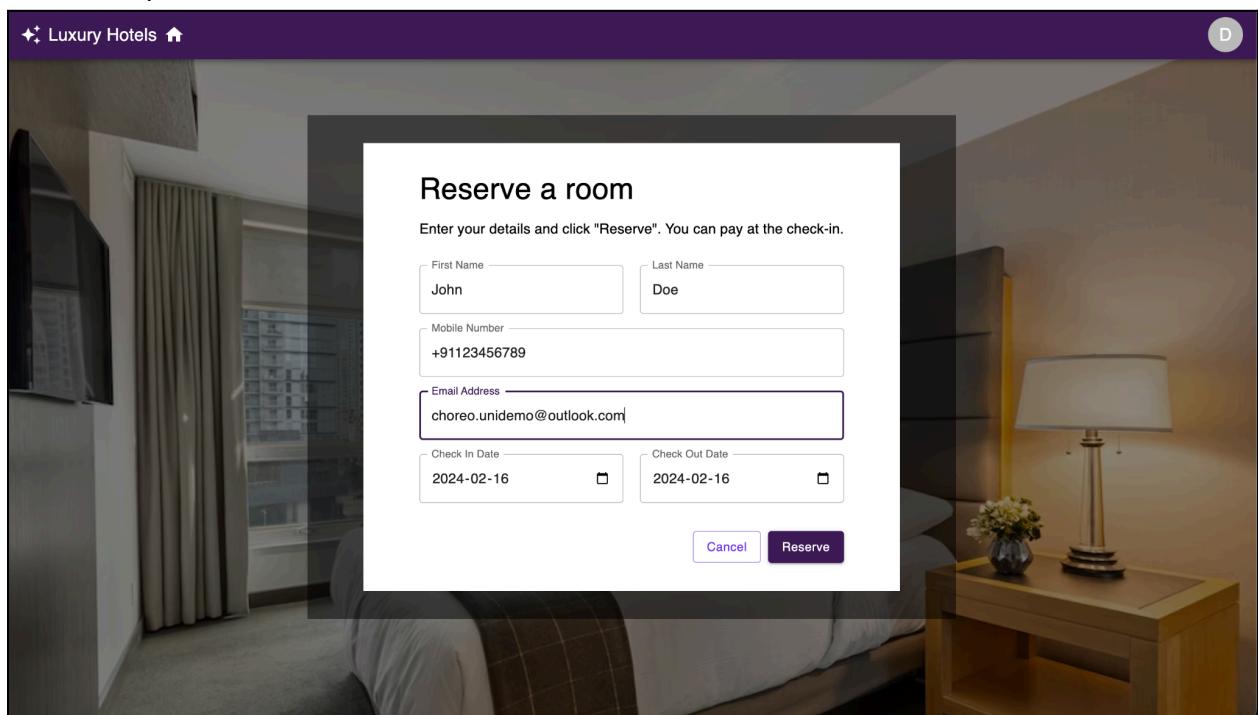
24. Set a check in date, check out date and select the no of guests. Then click **Search**.

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.