Building Cloud Native Applications with Ballerina and Choreo

Training Objective:

In this training, you will:

1. Build an HTTP service(API) using Ballerina.
2. Use Choreo to deploy the cloud-native application.
    a. Deploy a Ballerina service.
    b. Deploy a web application.

High-Level Steps:

1. Develop the HTTP service using the Ballerina programming language.
2. Push the code to your GitHub account.
3. Deploy the cloud native application on Choreo.
    a. Deploy the Ballerina HTTP service
    b. Deploy the ReactJS web application

Prerequisites:

1. A GitHub Account.
2. Git installed in your workstation.
3. A recent version of Google Chrome, Mozilla Firefox.
4. Ballerina v 2201.8.4 installed in your workstation.
5. Microsoft Visual Studio (VSCode) and WSO2 Ballerina plugin
6. PostMan and curl (or any HTTP client) installed in your workstation.
7. Choreo Account

Business Scenario:

Build a reservation system for a Luxury Hotel.

Solution

Build a web application that enables users to book rooms. This application will offer 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.
- The form validates and activates the "Reserve" button once all fields are properly filled. Upon reservation, a unique reference number is displayed for the user to copy.

List My 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:

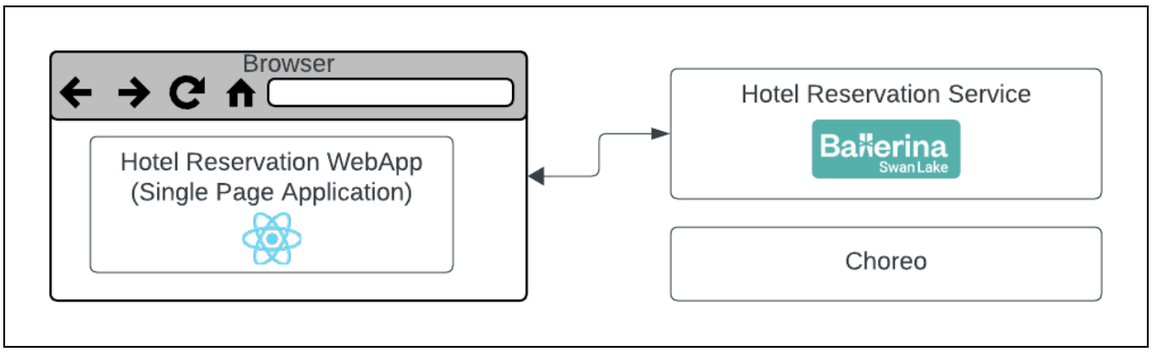
- Users have the flexibility to modify any part of their reservation.

Cancel Reservations:

- Users can cancel their reservations at any time, with a straightforward option to cancel their booking.

Design

The Hotel Reservation System will include a Hotel Reservation Service developed using Ballerina, and a Hotel Reservation Web Application developed using ReactJS.



Detailed Steps:

Developing the Hotel Reservation Service using Ballerina

The Hotel Reservation Service consists of a Ballerina HTTP service with 5 resources. A dedicated Ballerina project for the hotel reservation service has been created, complete with the required types and utility functions. During this session, we will focus on creating a Ballerina service and complete the implementation of the resources.

Base URL

http://localhost:9090/reservations

Resources

Resource	Path	Action	Query Param	Path Param	Request	Response

Get All available room types	/roomTypes/	GET	string checkinDate
			
			string checkoutDate
			
			int guestCapacity
						[
						{
						"id": 0,
						"name": "Single",
						"guestCapacity": 1,
						"price": 80
						
						},
						
						{
						"id": 0,
						"name": "Double",
						"guestCapacity": 2,
						"price": 100
						}

						]

Create a new reservation		POST			{
					"checkinDate": "2024-02-19T14:00:00Z",
					"checkoutDate": "2024-02-20T10:00:00Z",
					"rate": 100,
					"user": {
					"id": "123",
					"name": "waruna",
					"email": "waruna@someemail.com",
					"mobileNumber": "987"
					},
					"roomType": "Family"
					}
						{
						"id": "1",
						"checkinDate": "2024-02-19T14:00:00Z",
						"checkoutDate": "2024-02-20T10:00:00Z",
						"user": {
						"id": "123",
						"name": "waruna",
						"email": "waruna@someemail.com",
						"mobileNumber": "987"
						},
						"room": {
						"number" : 201,
						"type": {
						"id": 0,
						"name": "Double",
						"guestCapacity": 2,
						"price": 100
						}
						}
						}

Update an existing reservation		PUT		reservation_id	{
					"checkinDate": "2024-02-20T14:00:00Z",
					"checkoutDate": "2024-02-21T10:00:00Z"
					}
						{
						"id": "1",
						"checkinDate": "2024-02-19T14:00:00Z",
						"checkoutDate": "2024-02-21T10:00:00Z",
						"user": {
						"id": "123",
						"name": "waruna",
						"email": "waruna@someemail.com",
						"mobileNumber": "987"
						},
						"room": {
						"number" : 201,
						"type": {
						"id": 0,
						"name": "Double",
						"guestCapacity": 2,
						"price": 100
						}
						}
						}

Remove a reservation		DELTE		reservation_id		

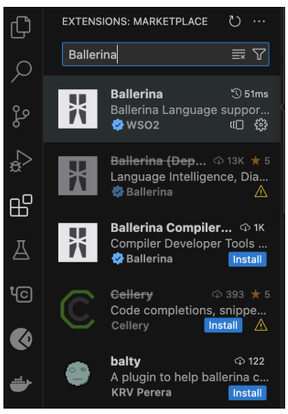
Get all reservations for a given user id	/users/	GET		userID		[
						{
						"checkinDate": "2024-02-19T14:00:00Z",
						"checkoutDate": "2024-02-20T10:00:00Z",
						"rate": 120,
						"user": {
						"id": "123",
						"name": "waruna",
						"email": "waruna@someemail.com",
						"mobileNumber": "987"
						},
						"roomType": "Family"
						},
						{
						"checkinDate": "2024-02-23T14:00:00Z",
						"checkoutDate": "2024-02-24T10:00:00Z",
						"rate": 100,
						"user": {
						"id": "123",
						"name": "waruna",
						"email": "waruna@someemail.com",
						"mobileNumber": "987"
						},
						"roomType": "Double"
						}
						]

Table 1: Reservation Service Table

Step 1: Create data types

1. Download and install Ballerina from the download page.

2. Install Ballerina Extension from Visual Studio Code



3. Fork the GitHub Repo - https://github.com/ballerina-guides/hotel-reservation-demo. Make sure you **untick** the option “Copy the main branch only”.

4. Clone the GitHub Repo https://github.com/<your_username>/ballerina-guides/hotel-reservation-demo using the following command.

```
Unset
git clone
https://github.com/<your_username>/ballerina-guides/hotel-reservation-demo
```

5. Open the project using Visual Studio Code.

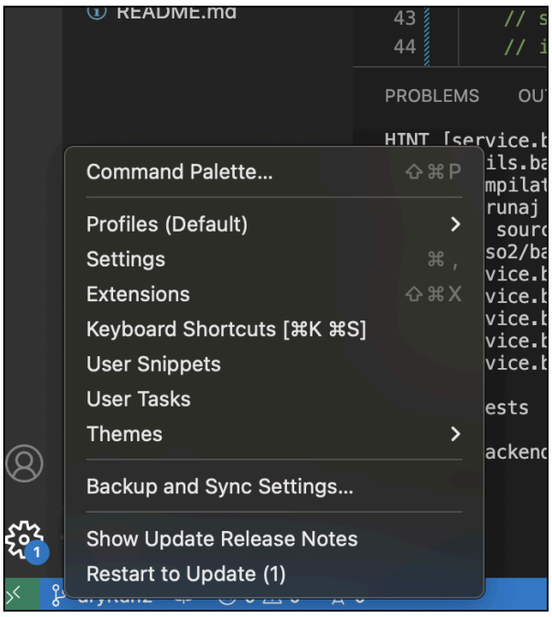
6. Go to the service directory and navigate to the types.bal file.

7. We need to generate the following two record types:
   ReservationRequest
   UpdateReservationRequest

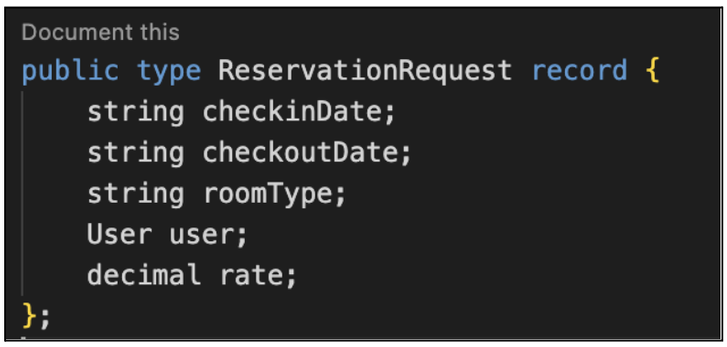
8. To generate the **ReservationRequest** type, copy the following JSON.

```
Unset
{
  "checkinDate": "2024-02-19T14:00:00Z",
  "checkoutDate": "2024-02-20T10:00:00Z",
  "rate": 100,
  "user": {
    "id": "123",
    "name": "waruna",
    "email": "waruna@someemail.com",
    "mobileNumber": "987"
  },
  "roomType": "Family"
}
```

Use the **Ballerina: Paste JSON as Record** option from the Command Palette.



9. Rename the record as Reservation Request.



Note: ***User*** *record is already added to types.bal since it is required for the* ***Reservation*** *record type. You can remove the duplicate* ***User*** *record.*

10. Copy the following JSON and generate the **UpdateReservationRequest.**

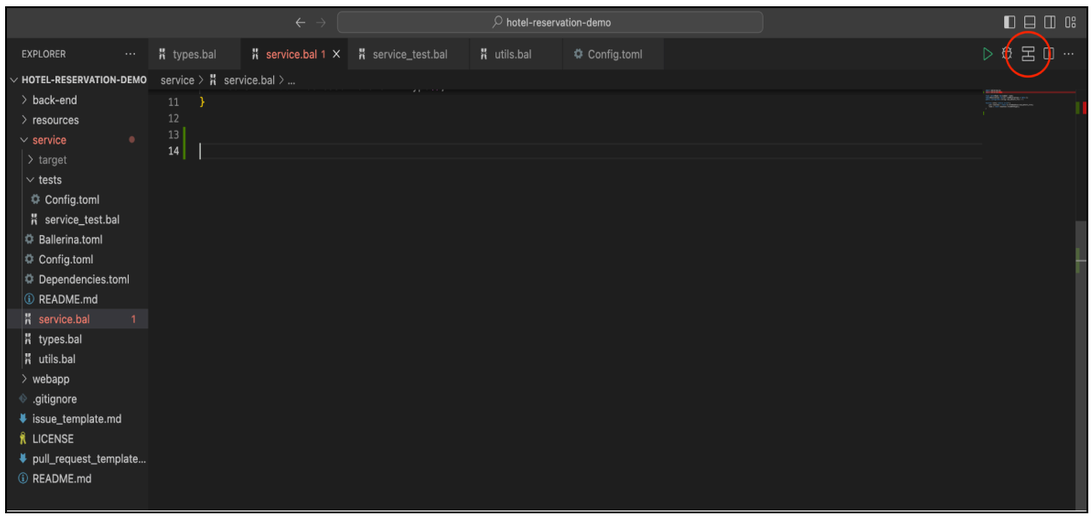
```
Unset
{
    "checkinDate": "2024-02-20T14:00:00Z",
    "checkoutDate": "2024-02-21T10:00:00Z"
}
```

Step 2: Create the HTTP service component

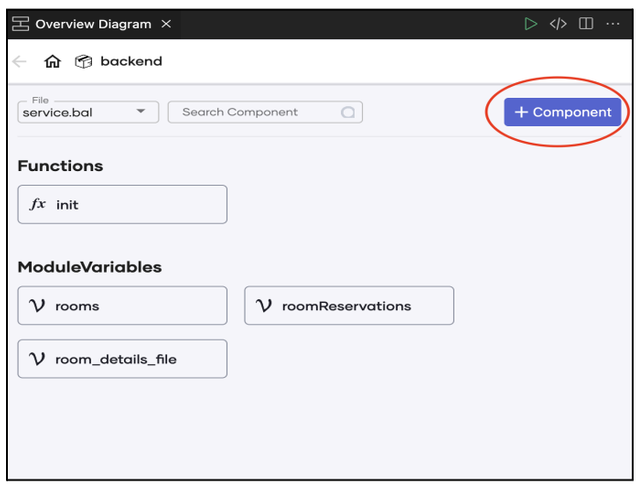
1. Add an HTTP service component to implement the hotel reservation API. Refer Table 1 for the required values for each resource.
    a. Get available room types
    b. Create a reservation
    c. Update the reservation
    d. Get user reservations
    e. Delete the reservation

You can use the VScode UI to add the HTTP services and resources.

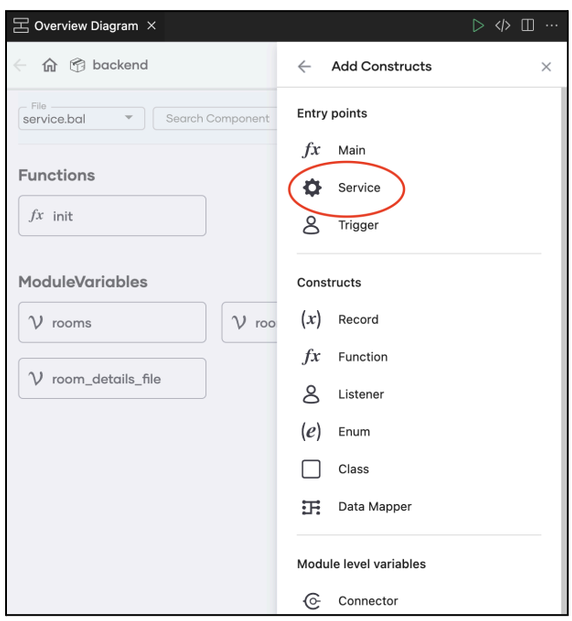
2. Click on the **Show Diagram** icon.



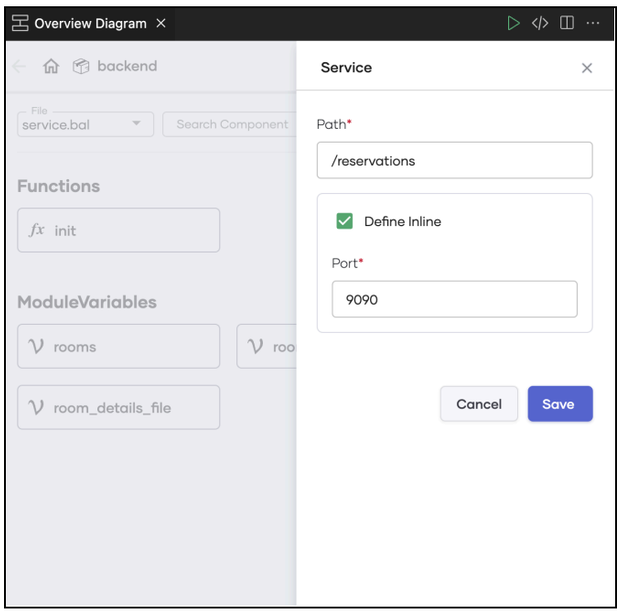
3. Click on **+Component**.



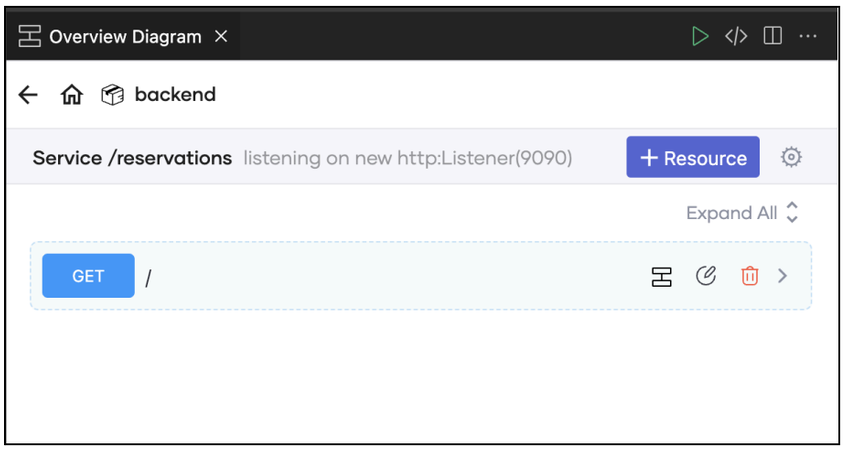
4. Click on **Service**.



5. Add the service path as **/reservations**.

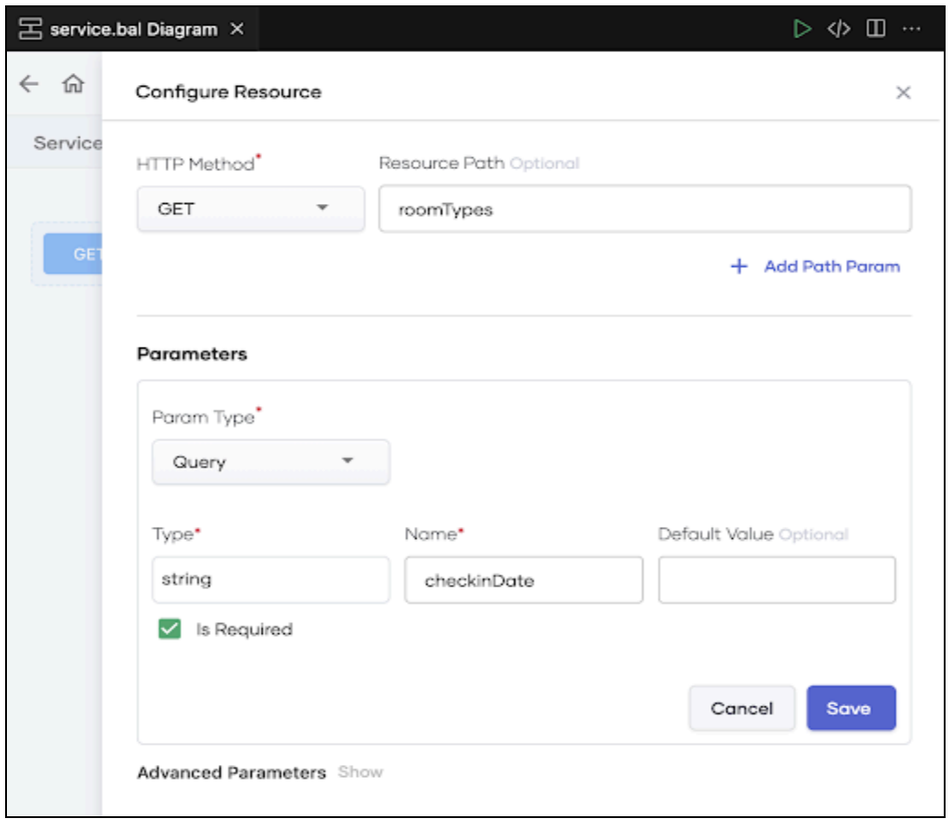


6. Edit the default GET resource.



7. Add the relevant query parameters and request and response types as shown below. Tick the “is required” check box for each parameter as all parameters are mandatory.

Field Name	Field Value
HTTP Method	GET
ResourcePath	roomTypes
Param Type	Query
Type	string
Name	checkinDate



8. Click on **+Add Param** and add the following additional parameters.

Param Type	Query
Type	string
Name	checkoutDate

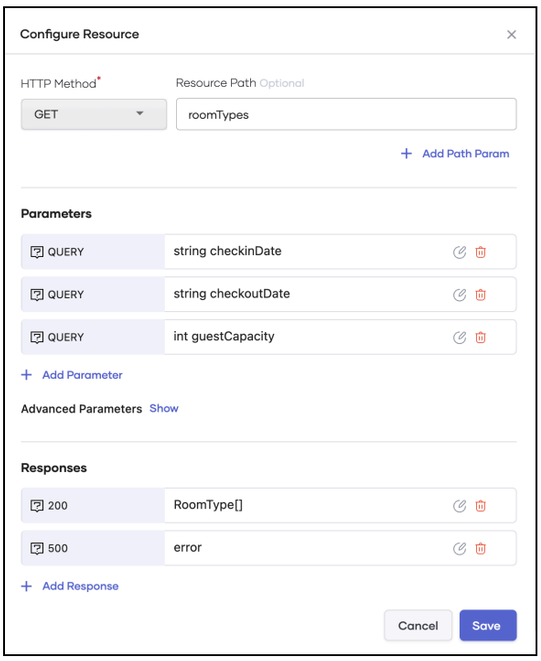
Param Type	Query
Type	int
Name	guestCapacity

9. Click on **+Add Response** and add the following responses.

Select Code	200
Type	RoomType[]

Select Code	500
Type	error

The completed resource for Get Room types can be viewed below.



10. Add the other resources similar to the steps above.

    Refer to the following table for the types required for each resource.

Resource	Path	Action	Path Param	Request Payload Type	Response Type
Create a new reservation		POST		ReservationRequest	Reservation|NewReservationError|error
Update an existing reservation		PUT	reservationId	UpdateReservationRequest	Reservation|UpdateReservationError|error
Remove a reservation		DELETE	reservationId		http:Ok|http:NOT_FOUND
Get all reservations for a given user id	/users/	GET	userID		Reservation[]

The following code will be generated once all the resources have been added.

```
service /reservations on new http:Listener(9090) {
    Visualize
    resource function get roomTypes(string checkinDate, string checkoutDate, int guestCapacity) returns error|RoomType[] {
    }

    Visualize
    resource function post .(NewReservationRequest payload) returns Reservation|error|NewReservationError {
    }

    Visualize
    resource function put [int reservationId](UpdateReservationRequest payload) returns Reservation|error|UpdateReservationError {
    }

    Visualize
    resource function delete [int reservationId](string? param) returns http:Ok|http:NotFound {
    }

    Visualize
    resource function get users/[string userId]() returns Reservation[] {
    }
}
```

The completed types will be as follows:

```
type NewReservationRequest record {
    string checkinDate;
    string checkoutDate;
    int rate;
    User user;
    string roomType;
};

type UpdateReservationRequest record {
    string checkinDate;
    string checkoutDate;
};

type NewReservationError record {|
    *http:NotFound;
    string body;
|};

type UpdateReservationError record {|
    *http:NotFound;
    string body;
|};
```

11. Complete the body of the resource with the application logic. You can use functions in utils.bal to complete the steps.

    eg: Here is the completed code for the getRoomTypes resource.

```
Unset
resource function get roomTypes(string checkinDate, string checkoutDate, int guestCapacity) returns RoomType[]|error {
        return getAvailableRoomTypes(checkinDate, checkoutDate, guestCapacity);
    }
```

12. Run the Tests and verify that all resources are working fine. You can find the ballerina test code in tests/service_test.bal

Note: For Windows, we need to use “..\resources\rooms.json” in Config.toml file

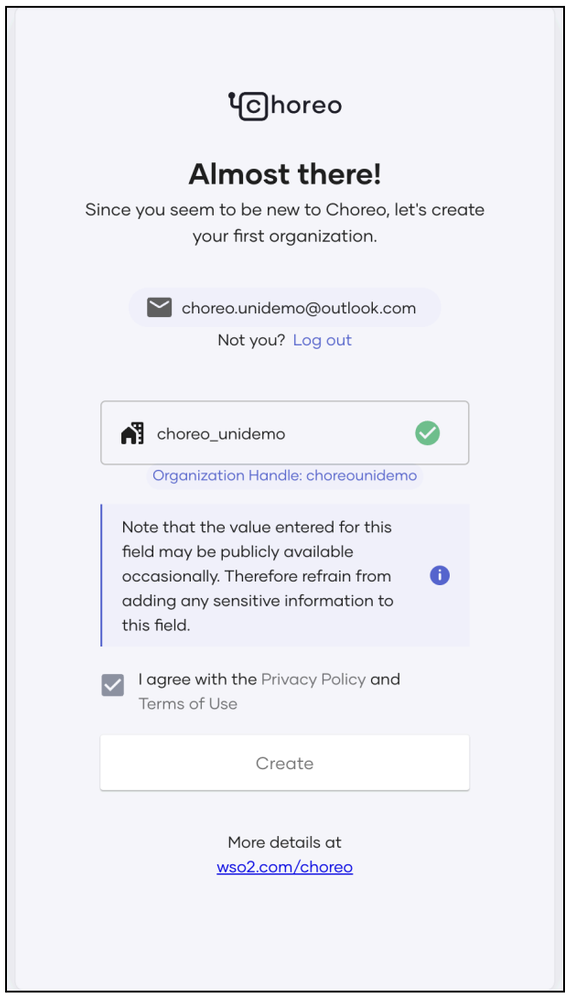
```
Unset
bal test
```

Deploying the Hotel Reservation App using Choreo

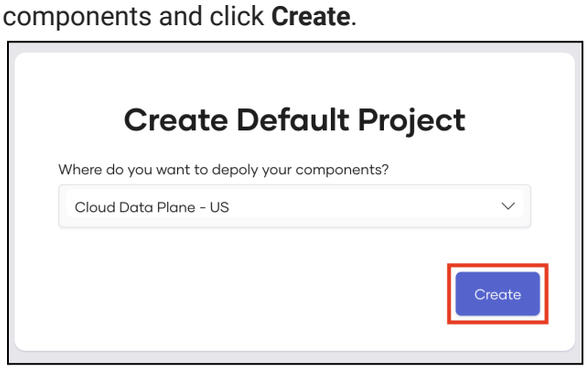
Step 1: Sign Up and Login to Choreo

1. Sign Up to Choreo by visiting the https://choreo.dev/ URL.

2. Once you log in to Choreo for the first time, you will be asked to provide an organization handle name. Provide a handle name and click **Create**.



3. To create the Default Project, choose the dataplane where you wish to deploy your components and click **Create**.



4. You will now be directed to the overview of the **Default Project**.

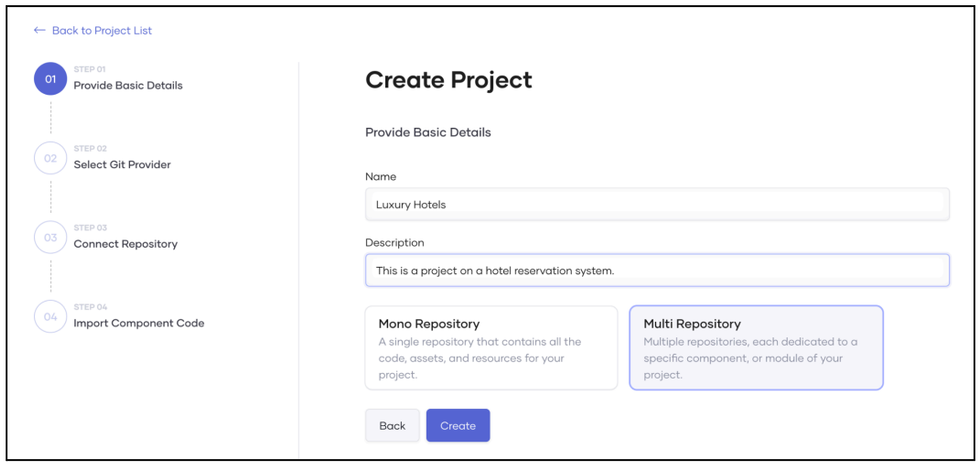
Step 2: Create a new Project

1. Click on the **Organization** card in the top menu.

2. Click the **Create Project** card.

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

Field Name	Field Value
Name	Luxury Hotels
Description	Luxury Hotels reservation system.
Project Type	Multi Repository



4. You will be directed to the overview of the project.

Step 3: Create and Deploy a Service for the Hotel Reservation Ballerina App

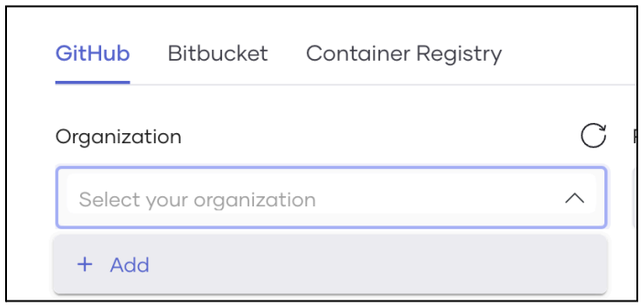
1. Select the **Service** card.

2. Provide a name and a description for the Service.

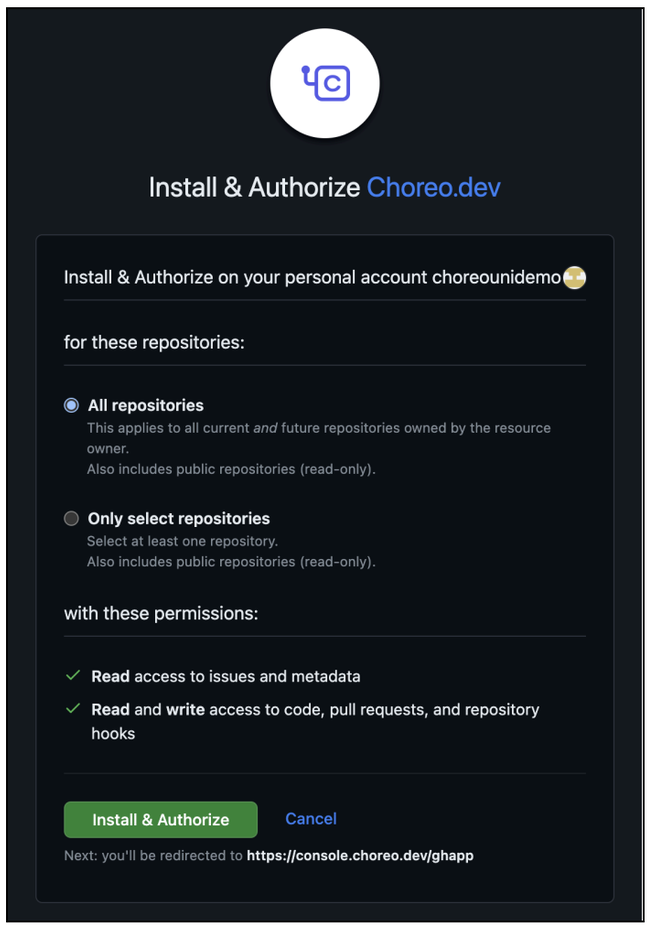
Field Name	Field Value
Name	Hotel Reservation Service
Description	Service for the hotel reservation application.

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

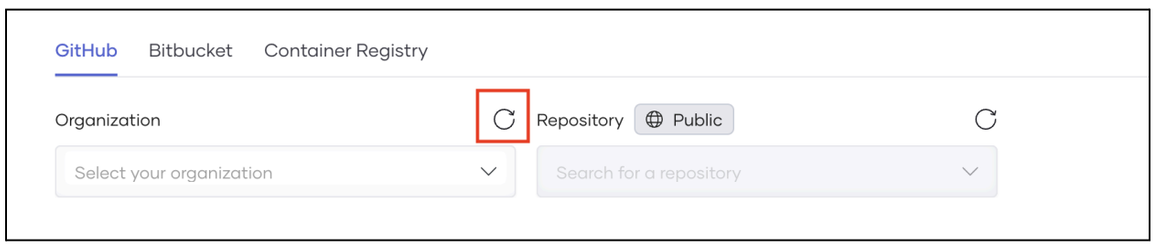
4. Expand the Select your organization section and click **Add**.



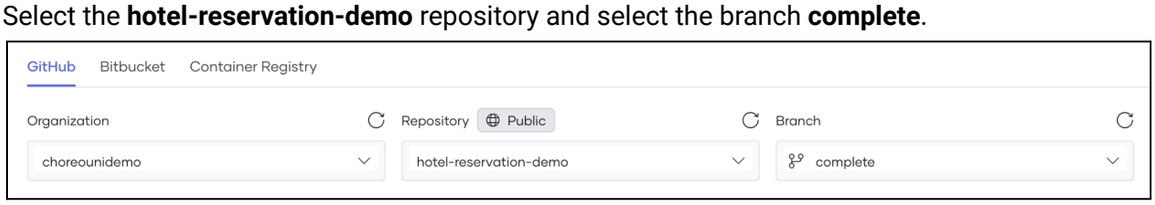
5. Keep the default selected option of **All repositories**. Then click **Install & Authorize**.



6. Click the refresh icon next to Organization.



7. Select the **hotel-reservation-demo** repository and select the branch **complete**.

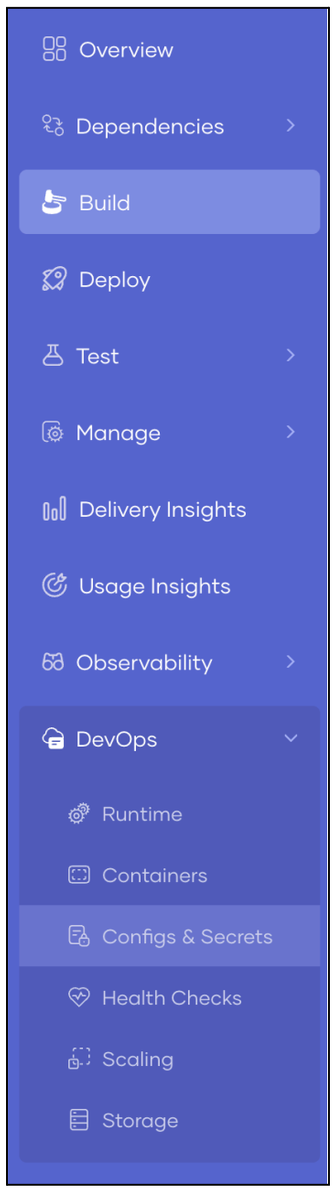


8. Select **Ballerina** as the **Buildpack** and set the Ballerina Project Directory as below. Then select **Create**.

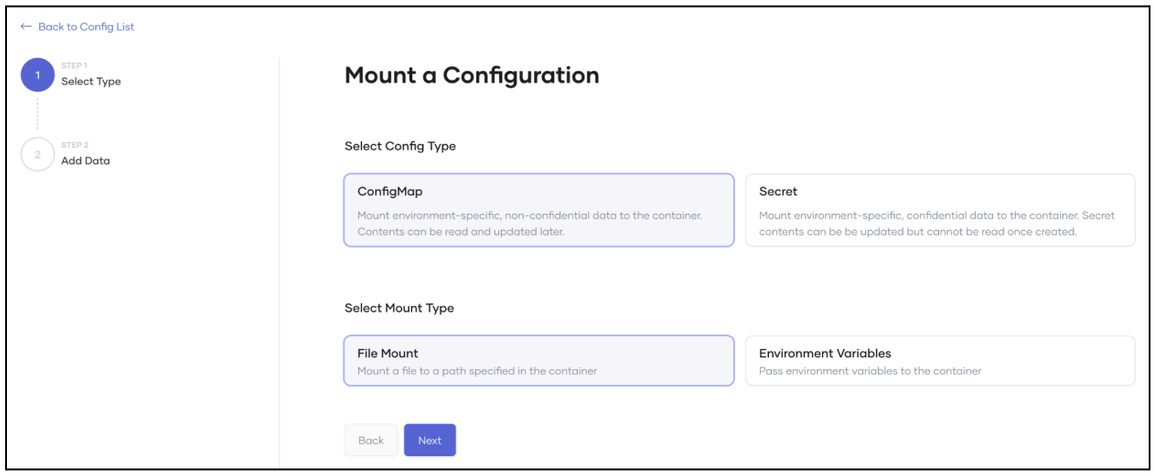
Field Name	Field Value
BuildPack	Ballerina
Ballerina Project Directory	/service

9. Click **Build** on the left navigation menu.Click the **Build** button in the Build page. This will open the **Commits** panel on the right. Select the most recent commit and click the **Build** button in the commits panel.

10. Once the build is complete, expand **DevOps** in the left navigation menu and click **Configs and Secrets** tab.



11. Click the **Create** button and select **ConfigMap** as the **Config Type** and **File Mount** as the **Mount Type** and Click **Next**.



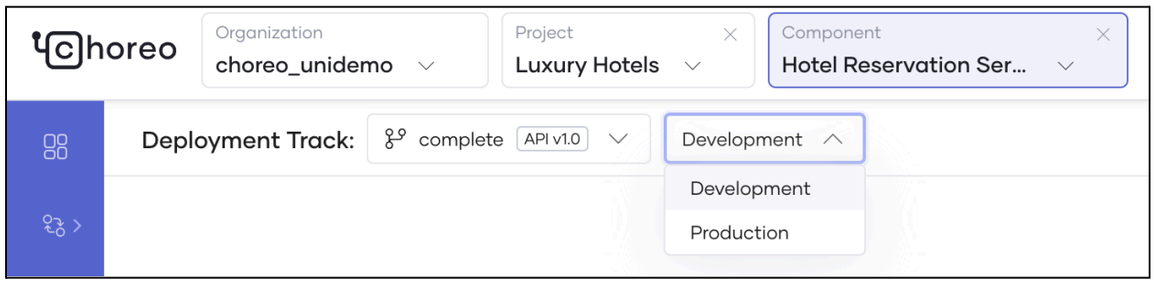
12. Provide the following details under the **Mount a Configuration** section.

Field Name	Field Value
Config Name	room-details
Mount path	/resources/rooms.json
File Content	Copy and paste all the content from https://raw.githubusercontent.com/ballerina-guides/hotel-reservation-demo/complete/resources/rooms.json location.

Alternatively, you can upload the content in the **<Path to your locally cloned repo>/resources/rooms.json** location.

13. Click the **Create** button.

14. Select **Production** from the secondary menu bar on top and repeat the steps 12, 13 and 14.

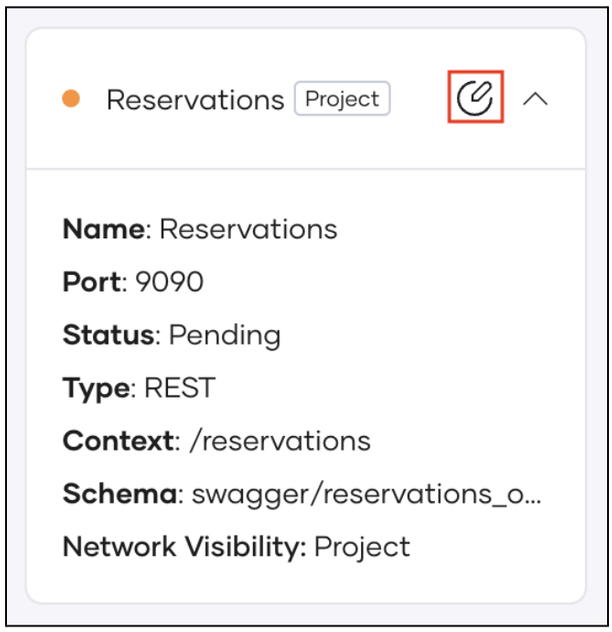


15. Click **Deploy** on the left navigation menu. In the Deploy page, click the **Configure & Deploy**.

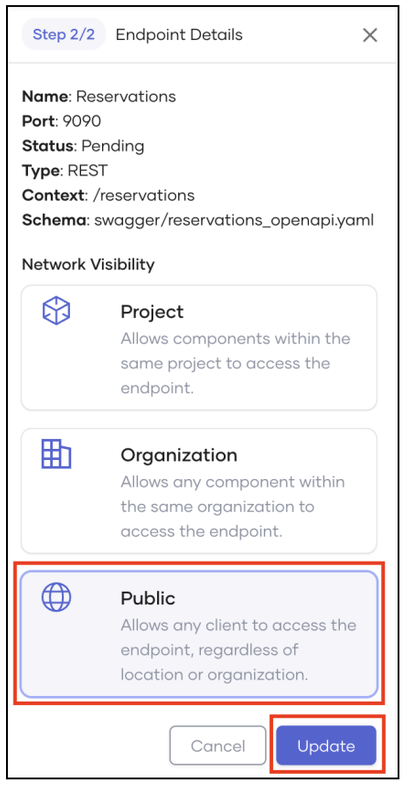
16. In the right side panel, provide the following details as inputs and click **Next**.

Field Name	Field Value
room_details_file	/resources/rooms.json

17. Click the **Edit** button.



18. Set the Network Visibility as **Public** and click **Update**.

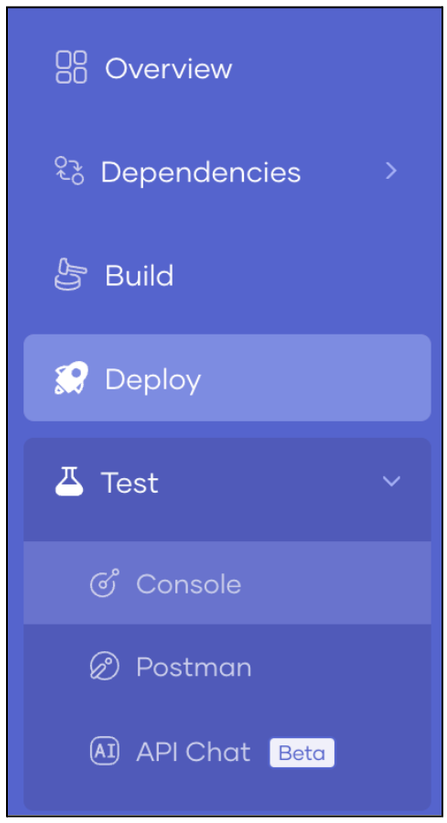


19. Click **Deploy** to deploy the service.

20. Once the service deployed successfully, click **Promote** to promote the build to the Production environment.

21. Select **Use Development Configurations** and click **Next**. This will load the provided development configs in the room_details_file. Click **Next** and click the **Promote** button in the right side panel.

22. Click and expand **Test** on the left navigation menu and click **Console**.



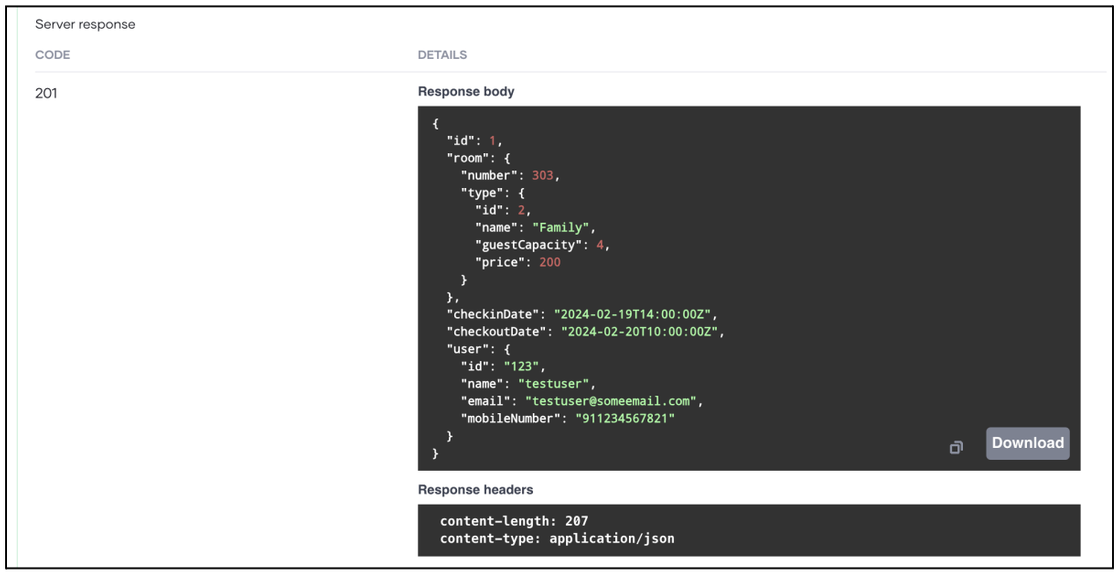
23. Expand the **POST/** section and click **Try it out**. In the **Request body** section, add the following details:

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"
	}

24. Click **Execute**.

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

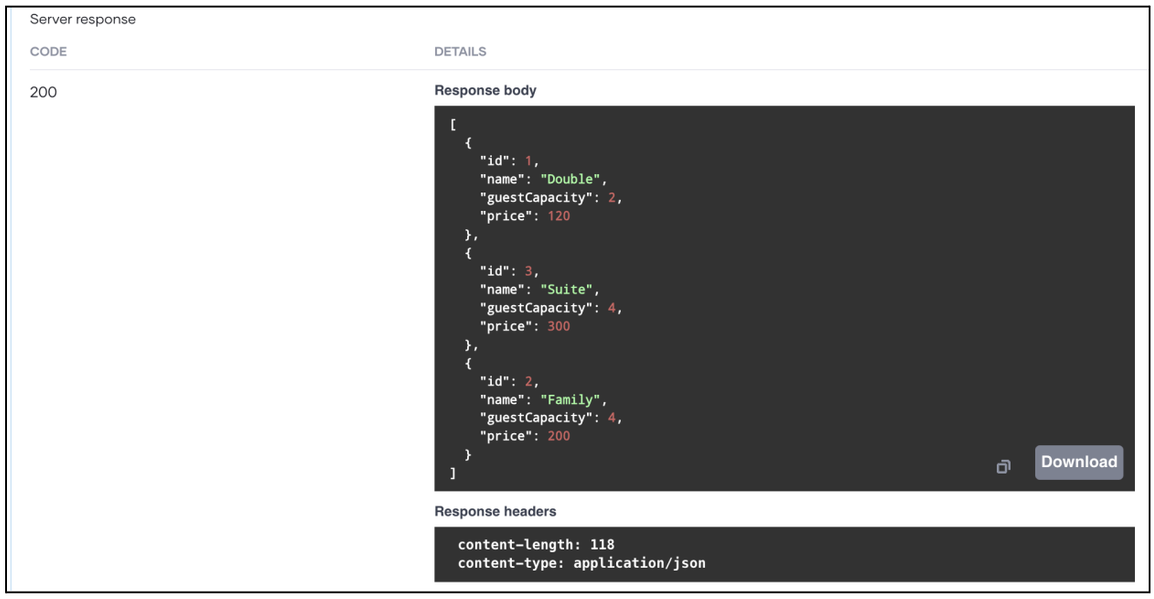


26. Expand the **GET /roomTypes** section and click **Try it out**.

27. Add the following details in each parameter and click **Execute**.

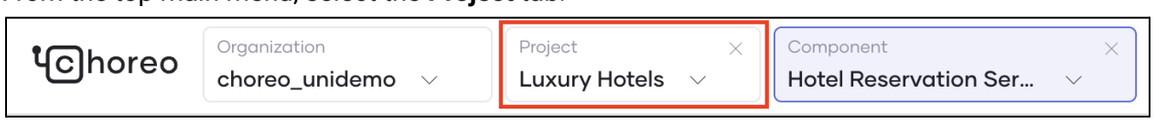
Field Name	Field Value
checkinDate	2024-02-19T14:00:00Z
checkoutDate	2024-02-20T10:00:00Z
guestCapacity	2

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



Step 4 - Deploy the Hotel Reservation Web Application

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



2. Click **Create**.

3. Select the **Web Application** card.

4. Provide a name and a description for the Web Application.

Field Name	Field Value
Name	Hotel Reservation Web Application
Description	Web Application for the hotel reservation application.

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

6. Select the **hotel-reservation-demo** repository and select the branch **complete**.

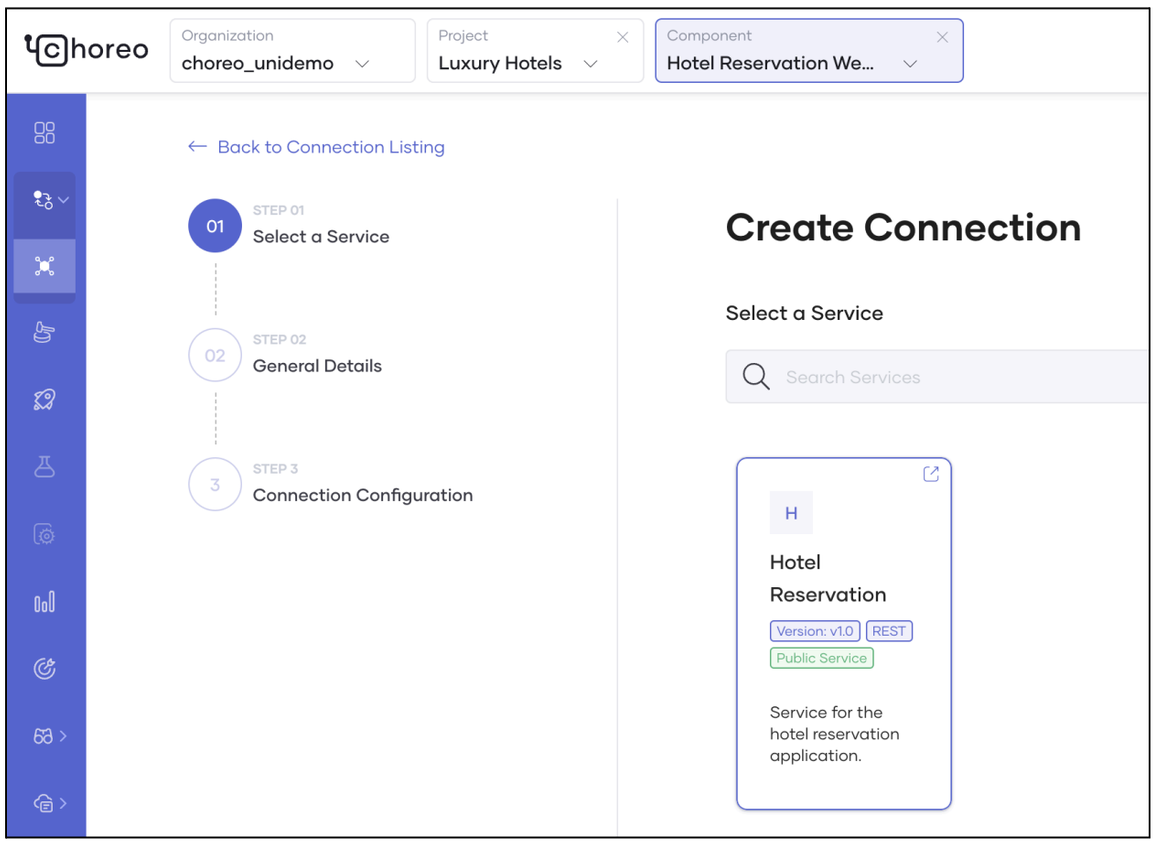
7. Select **React** as the **Buildpack** and set the following details. Then click **Create**.

Field Name	Field Value
BuildPack	React
Project Directory	/webapp
Build Command	npm run build
Build Path	/build
Node Version	20.11.0

8. Click and expand **Dependencies** on the left navigation menu and click the **Connections** tab.

9. Click **Create**.

10. Select the **Hotel Reservation** Service that was created in step 3..

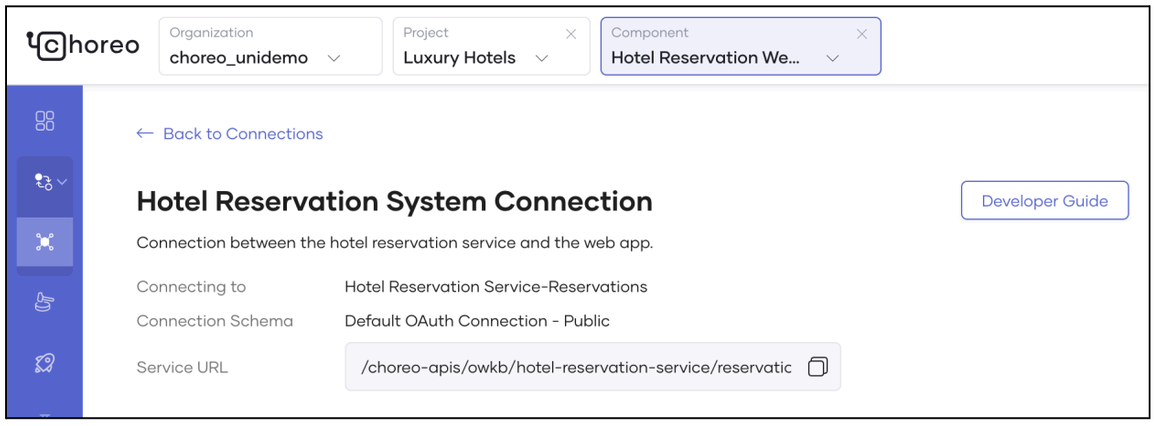


11. Provide a name and a description for the connection.

Field Name	Field Value
Name	Hotel Reservation System Connection
Description	Connection between the hotel reservation service and the web app.

12. Click **Next**. Then click **Finish**.

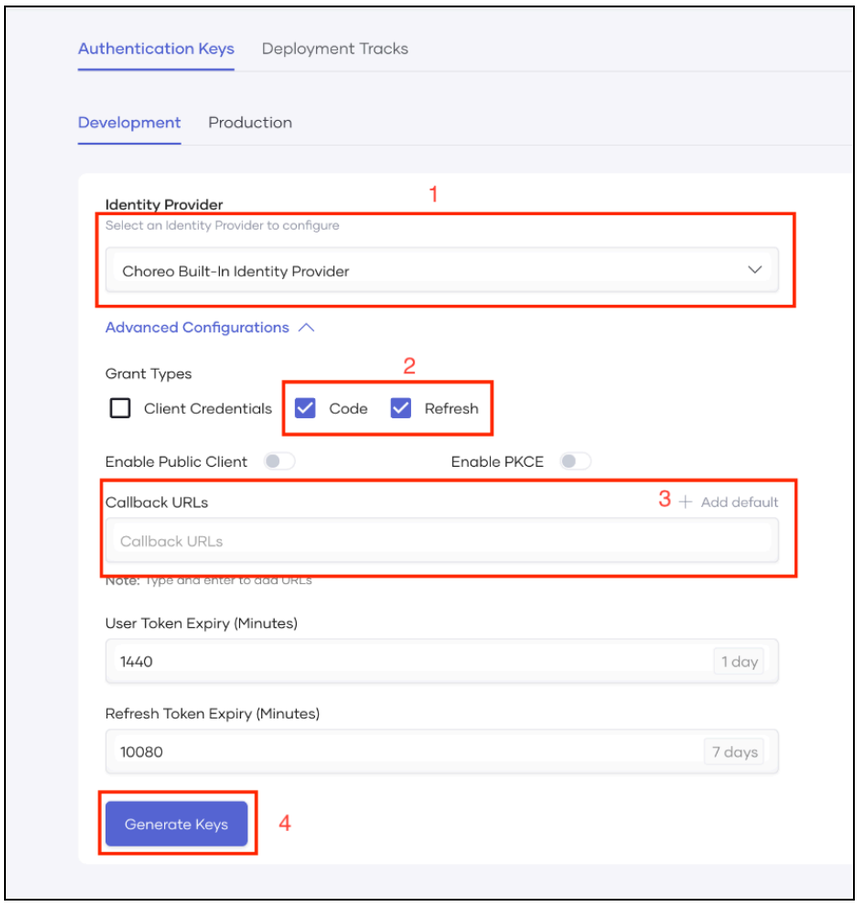
13. Copy and save the **Service URL** for later use.



14. Click **Build** on the left navigation menu. Click the **Build** button in the Build page. This will open the **Commits** panel on the right. Select the top most commit and click the **Build** button on the commits panel.

15. Once the build is completed, Click **Settings** on the left navigation menu.

16. Under the **Development** tab, select Choreo Built-In Identity Provider as the Identity Provider. Expand the **Advanced Configuration** section. Select the **Code** and **Refresh** grant types. Click **Add Default** to add a default callback url. Then click **Generate Keys**.



17. Click **Deploy** on the left navigation menu. Click the **Configure & Deploy** in the Deploy page.

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

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

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

Ex:

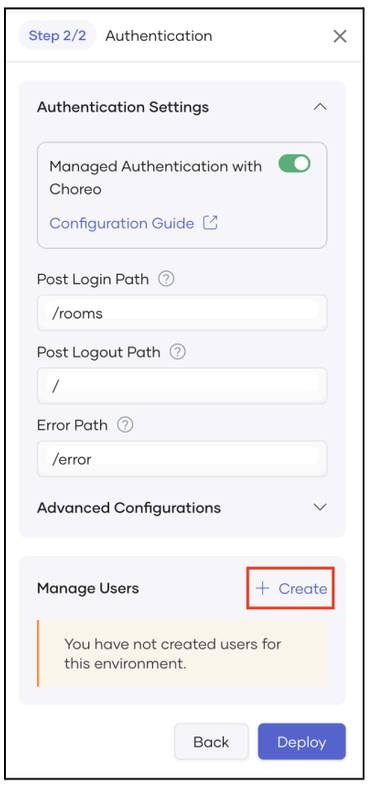
```
window.configs = {
     apiUrl:
'/choreo-apis/owkb/hotel-reservation-service/reservations-365/v1.
0',
};
```

18. Click **Next.**

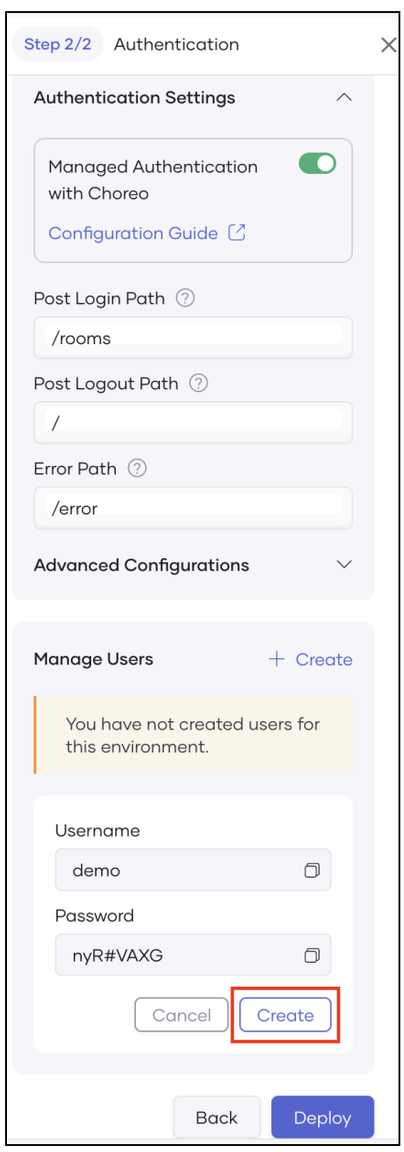
19. Provide following values for each parameter in the Authentication Settings window.

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

20. Click **Create** to expand the demo user creation window.



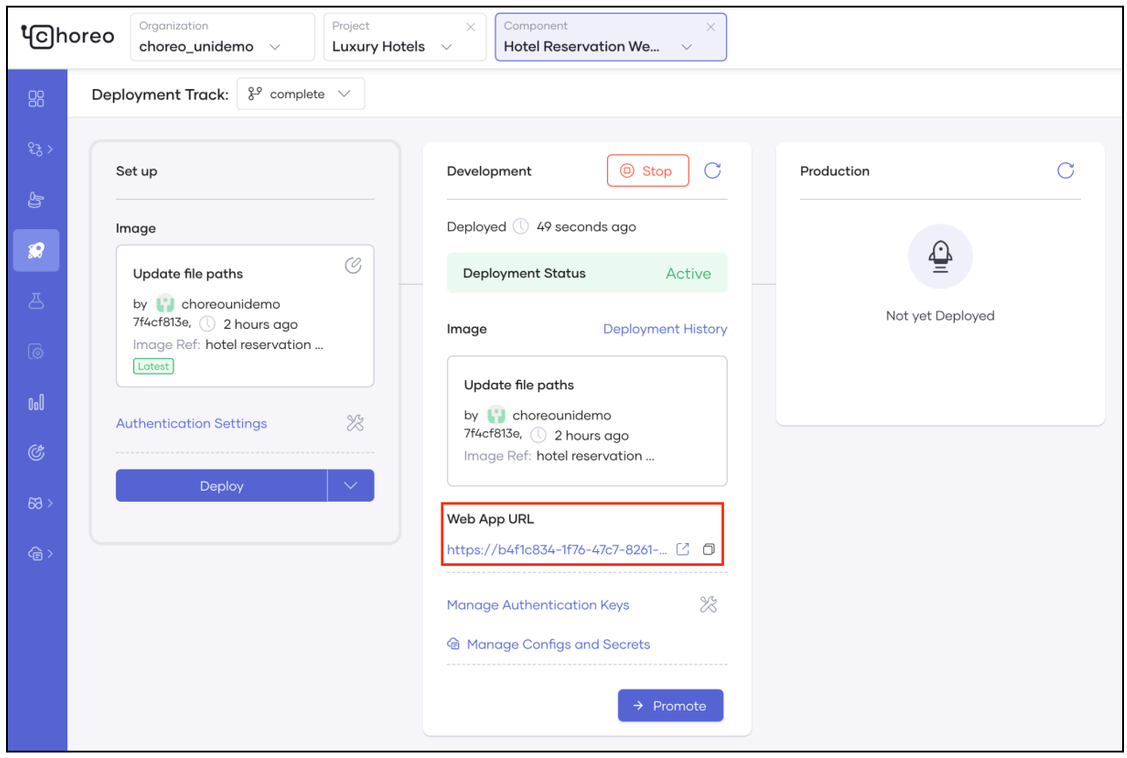
21. Click **Create** to create the user.



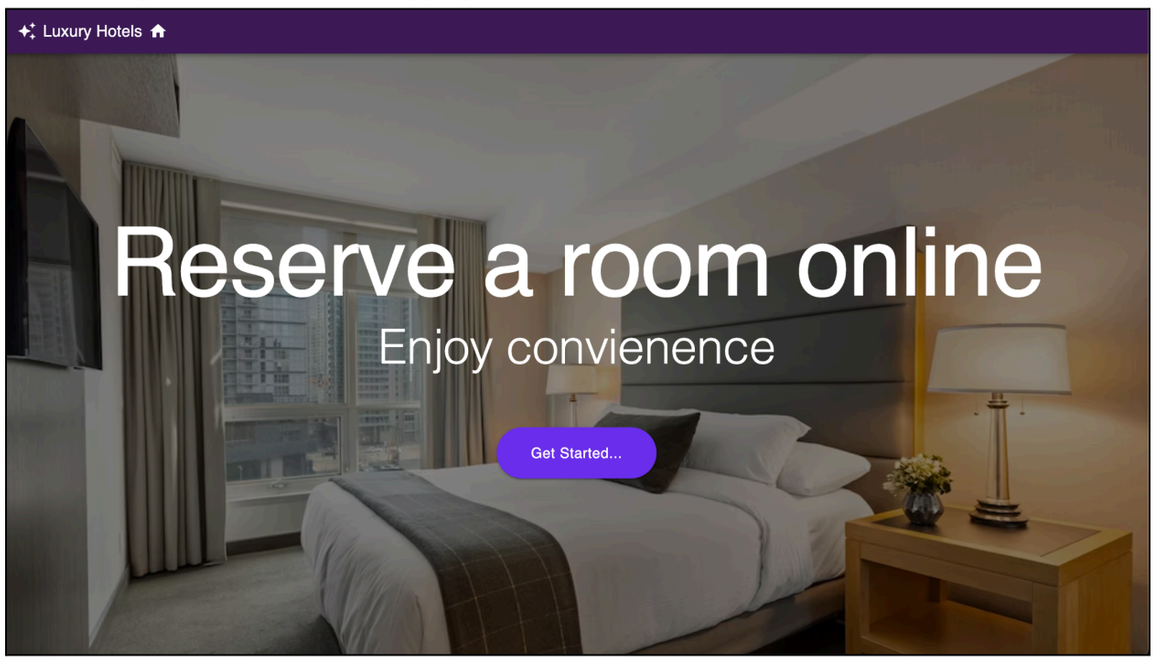
22. Once the user is created, save the username and the password for later use.

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

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

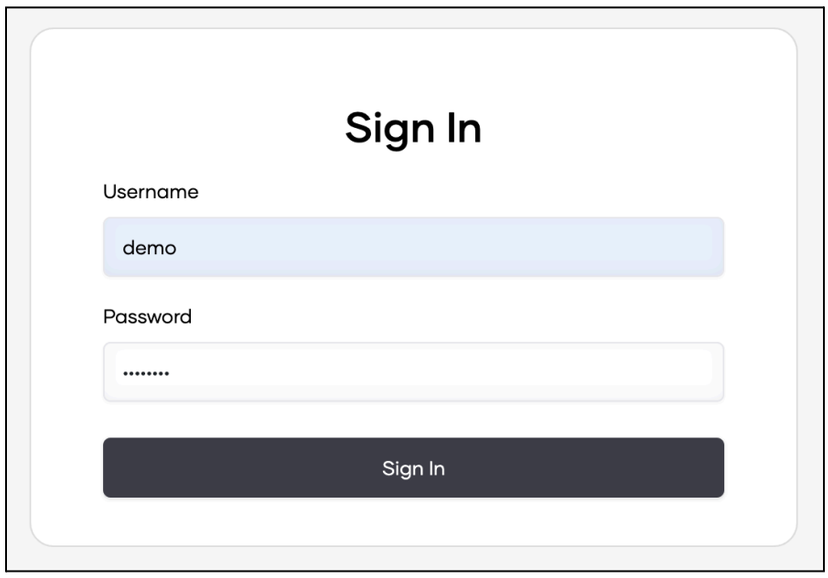


25. You will be greeted with the landing page shown below:

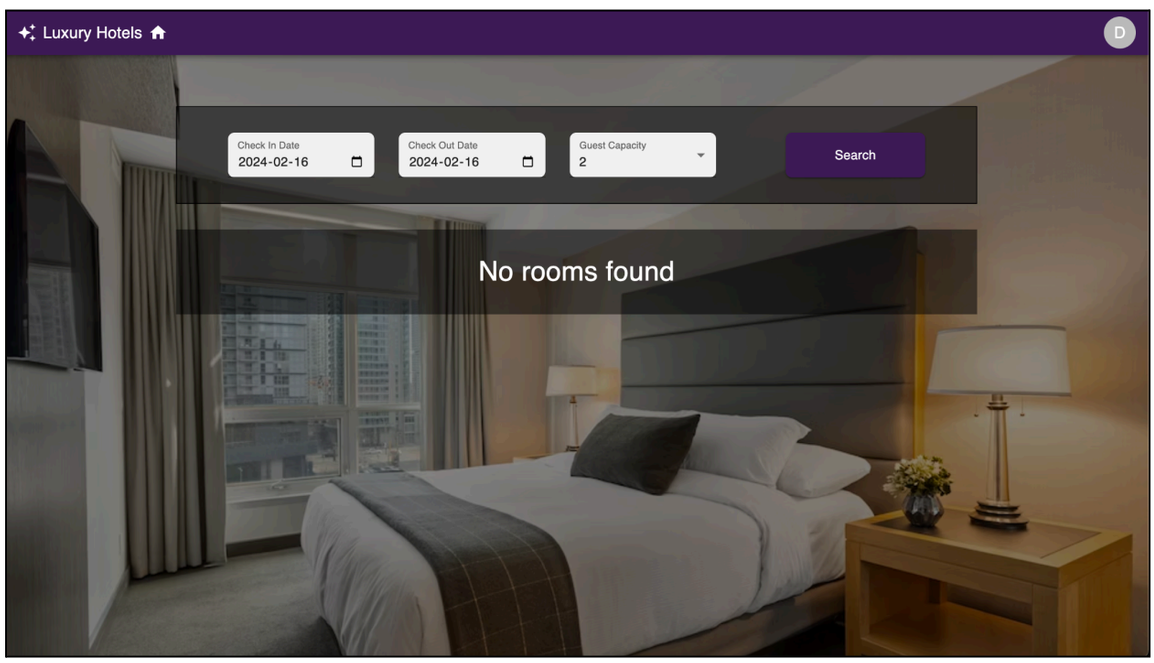


26. Click the **Get Started...** button.

27. This will direct you to the login page. Provide the demo username and password (created in the Step 23) and click Sign In.

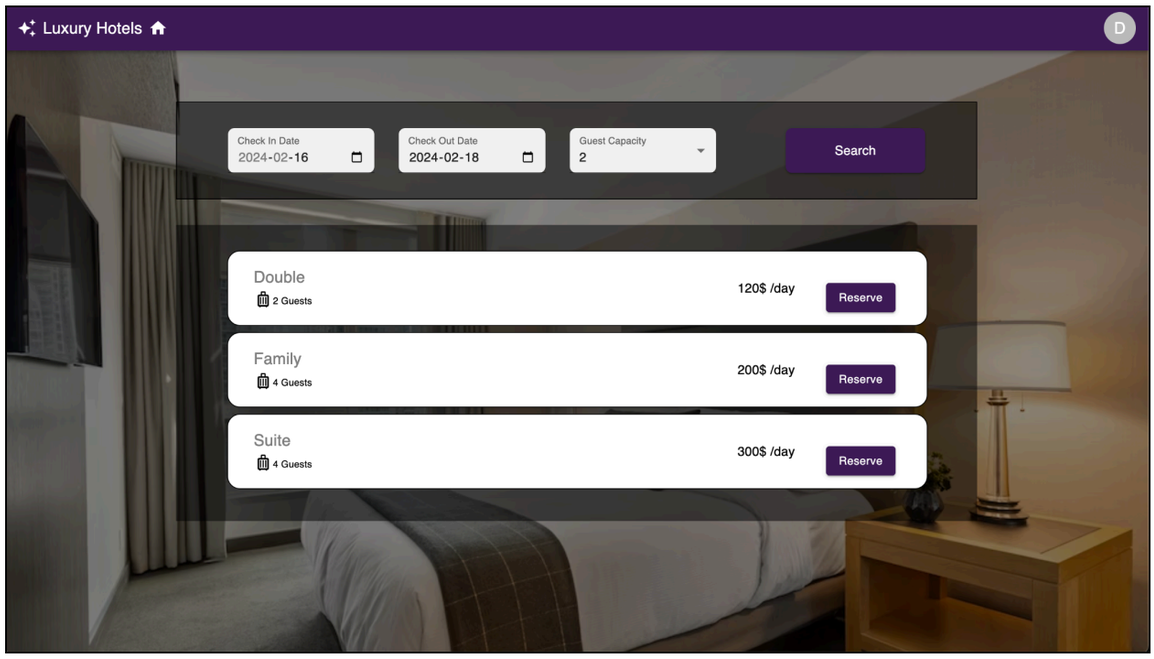


28. You will be logged into the Hotel Reservation web application.



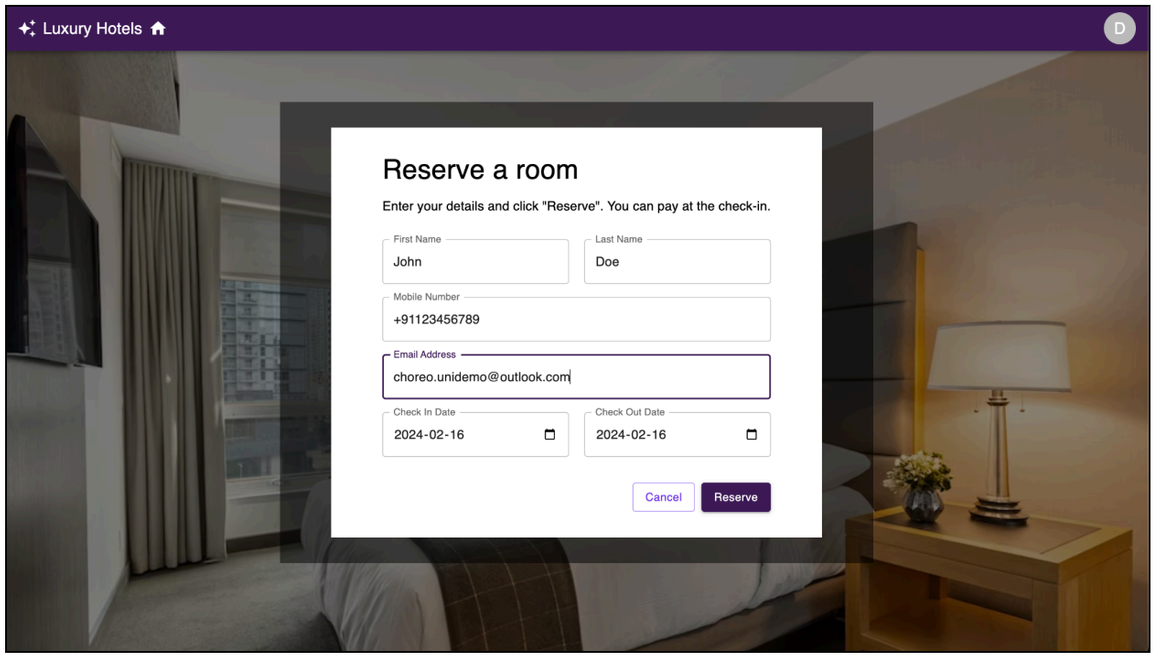
29. Set a check in date, check out date and select the no of guests. Then click **Search**.

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



31. Click **Reserve**.

32. Provide required details as below and click Reserve.



33. Your reservation will be created and you will receive an email confirmation.