**AWS Deployment**

**Access to EC2 via Putty**

To access an AWS EC2 instance using PuTTY on Windows, you need to convert the private key file from the **.pem** format to PuTTY's **.ppk** format and then use PuTTY to establish an SSH connection. Here are the steps:

**Step 1: Convert .pem Key to .ppk Format**

1. Download and install PuTTY if you haven't already. You can get it from the [PuTTY website](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html).
2. Launch PuTTYgen, which comes with PuTTY, and click the "Load" button.
3. In the file dialog, select your **.pem** private key file.
4. Click "Open."
5. PuTTYgen will prompt you to convert the key into PuTTY's own format. Click the "Save private key" button.
6. Save the key in **.ppk** format.

**Step 2: Configure PuTTY**

1. Launch PuTTY.
2. In the "Session" category, enter your EC2 instance's public IP address or DNS hostname in the "Host Name (or IP address)" field.
3. In the "Connection" > "Data" category, enter the username used for your EC2 instance. For Ubuntu 20.4, the username is typically **ubuntu**. For other distributions, check the official documentation for the correct username.
4. In the "Connection" > "SSH" > "Auth" category, click the "Browse" button and select the **.ppk** private key file you created earlier.
5. Optionally, you can save this configuration for future use by entering a name in the "Saved Sessions" field and clicking the "Save" button.
6. Click "Open" to initiate the SSH connection.

**Step 3: Connect to EC2 Instance**

PuTTY will use the private key to authenticate, and if everything is configured correctly, it should establish a connection to your EC2 instance.

Once connected, you can use the terminal provided by PuTTY to interact with your EC2 instance as if you were using SSH from a Linux terminal.

**Configure Environment**

1. **Update the system packages**: Ensure that your instance is up to date by running the following commands:

sudo apt update

1. **Install necessary software**: Install Java, MySQL and Apache Httpd Server which is required for backend.

**Java 8**

**Installing OpenJDK 8**

You can install it by typing the following commands:

sudo apt update

sudo apt install openjdk-8-jdk

Verify the installation by checking the Java version:

java -version

**Apache Httpd Server**

**Step 1 — Installing Apache**

Apache is available within Ubuntu’s default software repositories, making it possible to install it using conventional package management tools.

Let’s begin by updating the local package index to reflect the latest upstream changes:

sudo apt update

Then, install the apache2 package:

sudo apt install apache2

**Step 3 — Checking your Web Server**

At the end of the installation process, Ubuntu 20.04 starts Apache. The web server should already be up and running.

Check with the systemd init system to make sure the service is running by typing:

sudo systemctl status apache2

**MSQL**

**Step 1 — Installing MySQL**

On Ubuntu 20.04, you can install MySQL using the APT package repository.

To install it, update the package index on your server if you’ve not done so recently:

sudo apt update

Then install the mysql-server package:

sudo apt install mysql-server

Ensure that the server is running using the systemctl start command:

sudo systemctl start mysql.service

**Step 2 — Configuring MySQL**

For fresh installations of MySQL, you’ll want to run the DBMS’s included security script. This script changes some of the less secure default options for things like remote root logins and sample users.

To authenticate as the **root** MySQL user using a password, run this command:

mysql -u root -p

Run the following query:

SELECT user, host FROM mysql.user;

UPDATE mysql.user SET Host = '%' WHERE User = localhost;

GRANT ALL PRIVILEGES ON \*.\* TO ‘root’@’%’ GRANT OPTION;

FLUSH PRIVILEGES;

Then run the following ALTER USER command to change the root user’s authentication method to one that uses a password.

ALTER USER 'root'@'%' IDENTIFIED WITH caching\_sha2\_password BY 'demoadmin';

There is a known issue with some versions of PHP that causes problems with caching\_sha2\_password. Then run the following ALTER USER command to change the root user’s authentication method to one that uses a password. The following changes the authentication method to mysql\_native\_password:

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'demoadmin';

Step 1 – Download the git repository.

It will create following directory

/home/ubuntu/AIOPs\_Hackathon\_Lab\_Application

Inside it you will find following sub directories:

* aiops-hackathon-client
* aiops-hackathon-services
* architecture-documents
* database

**Backend**

Step 1 – Let’s create directories as follows,

/home/ubuntu/Lab\_App/Services/discovery-service

/home/ubuntu/Lab\_App/Services/api-gateway

/home/ubuntu/Lab\_App/Services/location-service

/home/ubuntu/Lab\_App/Services/car-service

/home/ubuntu/Lab\_App/Services/reservation-service

Step 2 – Run the following commands to copy the jar files to the relevant location

**Jar files**

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/aiops-hackathon/discovery-service/target/discovery-service-0.0.1-SNAPSHOT.jar /home/ubuntu/Lab\_App/Services/discovery-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/api-gateway/target/api-gateway-0.0.1-SNAPSHOT.jar /home/ubuntu/Lab\_App/Services/api-gateway

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/location-service/target/location-service-0.0.1-SNAPSHOT.jar /home/ubuntu/Lab\_App/Services/location-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/car-service/target/car-service-0.0.1-SNAPSHOT.jar /home/ubuntu/Lab\_App/Services/car-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/reservation-service/target/reservation-service-0.0.1-SNAPSHOT.jar /home/ubuntu/Lab\_App/Services/reservation-service

**application.properties files**

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/discovery-service/src/main/resources/application.properties /home/ubuntu/Lab\_App/Services/discovery-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/api-gateway/src/main/resources/application.properties /home/ubuntu/Lab\_App/Services/api-gateway

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/location-service/src/main/resources/application.properties /home/ubuntu/Lab\_App/Services/location-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/car-service/src/main/resources/application.properties /home/ubuntu/Lab\_App/Services/car-service

cp /home/ubuntu/AIOPs\_Hackathon\_Lab\_Application/aiops-hackathon-services/reservation-service/src/main/resources/application.properties /home/ubuntu/Lab\_App/Services/reservation-service

Step 4 – Once you copied following changes you have to do in application.properties file by navigating to each directory

1. /home/ubuntu/Lab\_App/Services/discovery-service/application.properties

eureka.instance.hostname=<your-ec2-instance-ip>

1. /home/ubuntu/Lab\_App/Services/api-gateway/application.properties

eureka.instance.client.serviceUrl.defaultZone=http:// <your-ec2-instance-ip>:8761/eureka/

spring.cloud.gateway.routes[0].uri=http:// <your-ec2-instance-ip>:9001

spring.cloud.gateway.routes[1].uri=http:// <your-ec2-instance-ip>:9003

spring.cloud.gateway.routes[2].uri=http:// <your-ec2-instance-ip>:9002

spring.cloud.gateway.routes[3].uri=http:// <your-ec2-instance-ip>:8761

spring.cloud.gateway.routes[4].uri=http:// <your-ec2-instance-ip>:8761

1. /home/ubuntu/Lab\_App/Services/location-service/application.properties

spring.datasource.url=jdbc:mysql://<your-db-ec2-instance-ip>:3306/car\_rental\_db

spring.datasource.username=root

spring.datasource.password=demoadmin

eureka.instance.client.serviceUrl.defaultZone=http:// <your-ec2-instance-ip>:8761/eureka/

1. /home/ubuntu/Lab\_App/Services/car-service/application.properties

spring.datasource.url=jdbc:mysql://<your-db-ec2-instance-ip>:3306/car\_rental\_db

spring.datasource.username=root

spring.datasource.password=demoadmin

eureka.instance.client.serviceUrl.defaultZone=http://<your-ec2-instance-ip>:8761/eureka/

1. /home/ubuntu/Lab\_App/Services/reservation-service/application.properties

spring.datasource.url=jdbc:mysql://<your-db-ec2-instance-ip>:3306/car\_rental\_db

spring.datasource.username=root

spring.datasource.password=demoadmin

eureka.instance.client.serviceUrl.defaultZone=http:// <your-ec2-instance-ip>:8761/eureka/

Step 5 – Now run the following commands to run your jar files.

nohup java -Xms256m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./java\_pid.hprof -jar /home/ubuntu/Lab\_App/Services/discovery-service/discovery-service-0.0.1-SNAPSHOT.jar > discovery-service.log 2>&1 &

nohup java -Xms256m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./java\_pid.hprof -jar /home/ubuntu/Lab\_App/Services/api-gateway/api-gateway-0.0.1-SNAPSHOT.jar > api-gateway.log 2>&1 &

nohup java -Xms256m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./java\_pid.hprof -jar /home/ubuntu/Lab\_App/Services/location-service/location-service-0.0.1-SNAPSHOT.jar > location-service.log 2>&1 &

nohup java -Xms256m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./java\_pid.hprof -jar /home/ubuntu/Lab\_App/Services/car-service/car-service-0.0.1-SNAPSHOT.jar > car-service.log 2>&1 &

nohup java -Xms256m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./java\_pid.hprof -jar /home/ubuntu/Lab\_App/Services/reservation-service/reservation-service-0.0.1-SNAPSHOT.jar > reservation-service.log 2>&1 &

Note: If you want to build backend services by own you will follow the instructions listed below

Step 1 – Navigate to the directory where your Backend services are located on the Local environment.

To build the backend, follow these steps:

1. Go to `aiops-hackathon-services`.

2. Navigate to `discovery-service` and run ` mvn clean package -DskipTests `.

3. Navigate to `api-gateway` and run ` mvn clean package -DskipTests `.

4. Similarly, after navigating to `location-service`, run ` mvn clean package -DskipTests`.

5. Go to `car-service` and run ` mvn clean package -DskipTests `.

6. Finally, go to `reservation-service` and run ` mvn clean package -DskipTests `.

And you can use generated .jar files to use for your deployment

**Frontend**

Here I have put the build files you want. You just need to follow the steps to deploy angular application.

Step 1 – Do cd into /var/www/html and create directory named “demo.”

Step 2 – Run the following command to copy the build files of the angular application.

Sudo cp <your-directory-name>/aiops-hackathon-client/dist/car-booking-app /var/www/html/demo

If you’d like to build the application from scratch, you will follow the instructions before deploying it.

**Build Angular application.**

Step 1 – Navigate to the directory where your Angular application is located on the Local environment.

Step 2 – Before deploying the frontend application there are few things to change in following files:

1. aiops-hackathon-client\src\index.html

<base href="/" /> change to <base href="/demo/" />

1. aiops-hackathon-client\src\environments\environment.prod.ts

baseUrl: 'http://<your-ec2-public-ip>:9004'

1. aiops-hackathon-client\src\proxy.conf.json

"target": "http:// <your-ec2-public-ip>:9004"

Step 3 – Build with the following command.

ng build --prod --aot --output-hashing=all

Note: It will generate dist file for you and inside dist/car-booking-app contains all build files which is related to angular application.