# CS 5610 Introduction to Cloud Computing

Instructor: Dr. Jun Wang

Project Title

Orgray Car Rental Application

Team Name

*13269 - Innova Cloud*

|  |  |
| --- | --- |
| Chunduri Shiva Sai Kumar | 700754515 |

Contents

[Abstract 3](#_Toc152091533)

[Definitions 4](#_Toc152091534)

[Virtual Private Clouds (VPC) 4](#_Toc152091535)

[Subnets 5](#_Toc152091536)

[Route Table 5](#_Toc152091538)

[Internet Gateway 6](#_Toc152091539)

[Security Groups 6](#_Toc152091540)

[Elastic Cloud Compute (EC2) 8](#_Toc152091542)

[Relational Database Service (RDS) 8](#_Toc152091543)

[Elastic Bean Stalk (EBS) 9](#_Toc152091545)

[Cloud Watch 10](#_Toc152091546)

[Creation of VPC, Subnets, Route Tables, Internet Gateway 10](#_Toc152091547)

[Creation of Key-Pair 13](#_Toc152091548)

[Creation of DB Subnet Groups 13](#_Toc152091549)

[Creation of Relational Database Service using Postgres 14](#_Toc152091550)

[Creation of IAM role for EC2 Instance 16](#_Toc152091551)

[Creation of Security Group for Elastic Beanstalk 16](#_Toc152091552)

[Creation of ElasticBeanStalk 17](#_Toc152091553)

[Application Deployment in Elastic Bean Stalk 20](#_Toc152091554)

[Creation of Dashboards in Cloud Watch 23](#_Toc152091555)

[Application Overview 24](#_Toc152091556)

[Login Page 24](#_Toc152091557)

[Home Page 25](#_Toc152091558)

[Check-In Page 25](#_Toc152091559)

[Current Bookings Page 26](#_Toc152091560)

[Upload Car Page 26](#_Toc152091561)

[Edit Car Details Page 27](#_Toc152091562)

[Check-out Page 28](#_Toc152091563)

[Check-Out Summary Page 28](#_Toc152091564)

[Database Verification 29](#_Toc152091565)

[Conclusion 30](#_Toc152091566)

[References 30](#_Toc152091567)

# Abstract

This project makes use of a variety of AWS services for scalable and effective architecture, leveraging AWS to develop a full car rental application. A Virtual Private Cloud (VPC) and related subnets are created as the basis, and route tables are used to regulate traffic flow. Web connectivity is ensured by an Internet gateway. Carefully established Security Groups and Key Pairs are used to enforce security.

The backbone of the application is formed by Elastic Cloud Compute (EC2) instances, providing scalable and resizable compute capacity. The data layer is managed by the Relational Database Service (RDS) using PostgreSQL, organized into DB Subnet Groups for enhanced security. Elastic Beanstalk facilitates the deployment and management of the application, streamlining the process and ensuring optimal resource utilize.

To monitor and manage the entire system, CloudWatch is employed, creating dashboards for comprehensive insights into performance and health. The application itself consists of essential pages, including a secure login, a user-friendly home page, a check-in page for reservations, a current bookings page, an upload car page, and pages for editing car details and checking out.

The seamless integration of these AWS services not only ensures a robust and scalable infrastructure but also enhances the application's performance, security, and overall user experience. This project serves as a practical demonstration of deploying a dynamic and functional car rental application on the AWS cloud platform.

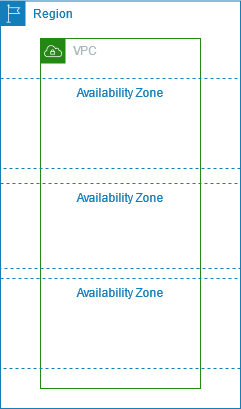
# Definitions

## Virtual Private Clouds (VPC)

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon EC2 instances, into your VPC. A VPC spans all of the Availability Zones in a Region. After you create a VPC, you can add one or more subnets in each Availability Zone

When you create a VPC, you specify its Ip addresses as follows:

* Ipv4 only – The VPC has an Ipv4 CIDR block but does not have an Ipv6 CIDR block.
* Dual stack – The VPC has both an Ipv4 CIDR block and an Ipv6 CIDR block.



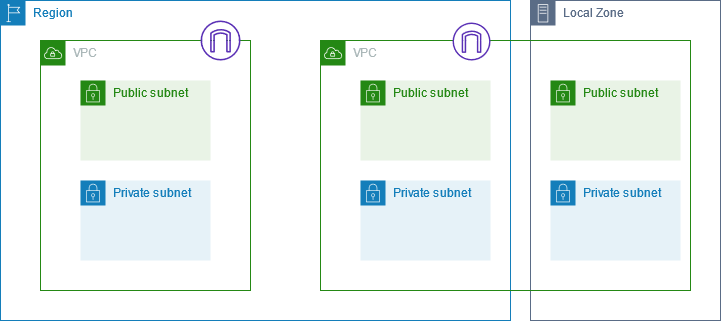
## Subnets

A subnet is a range of Ip addresses in your VPC. You can create AWS resources, such as EC2 instances, in specific subnets. Each subnet must reside entirely within one Availability Zone and cannot span zones. By launching AWS resources in separate Availability Zones, you can protect your applications from the failure of a single Availability Zone.

### Subnet Types

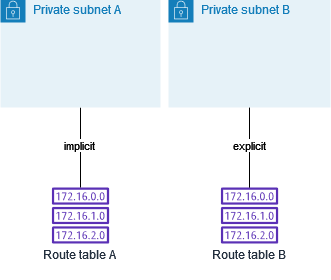
The subnet type is determined by how you configure routing for your subnets. For example:

* public subnet – The subnet has a direct route to an internet gateway. Resources in a public subnet can access the public internet.
* private subnet – The subnet does not have a direct route to an internet gateway. Resources in a private subnet require a NAT device to access the public internet.
* VPN-only subnet – The subnet has a route to a Site-to-Site VPN connection through a virtual private gateway. The subnet does not have a route to an internet gateway.
* Isolated subnet – The subnet has no routes to destinations outside its VPC. Resources in an isolated subnet can only access or be accessed by other resources in the same VPC.



## Route Table

A route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed.



## Internet Gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet. It supports Ipv4 and Ipv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.

An internet gateway enables resources in your public subnets (such as EC2 instances) to connect to the internet if the resource has a public Ipv4 address or an Ipv6 address. Similarly, resources on the internet can initiate a connection to resources in your subnet using the public Ipv4 address or Ipv6 address. For example, an internet gateway enables you to connect to an EC2 instance in AWS using your local computer.

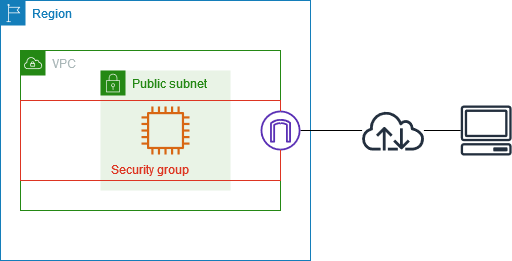
An internet gateway provides a target in your VPC route tables for internet-routable traffic. For communication using Ipv4, the internet gateway also performs network address translation (NAT). For communication using Ipv6, NAT is not needed because Ipv6 addresses are public.

## Security Groups

A security group controls the traffic that is allowed to reach and leave the resources that it is associated with. For example, after you associate a security group with an EC2 instance, it controls the inbound and outbound traffic for the instance.

When you create a VPC, it comes with a default security group. You can create additional security groups for a VPC, each with their own inbound and outbound rules. You can specify the source, port range, and protocol for each inbound rule. You can specify the destination, port range, and protocol for each outbound rule.

The following diagram shows a VPC with a subnet, an internet gateway, and a security group. The subnet contains an EC2 instance. The security group is assigned to the instance. The security group acts as a virtual firewall. The only traffic that reaches the instance is the traffic allowed by the security group rules. For example, if the security group contains a rule that allows ICMP traffic to the instance from your network, then you could ping the instance from your computer. If the security group does not contain a rule that allows SSH traffic, then you could not connect to your instance using SSH.

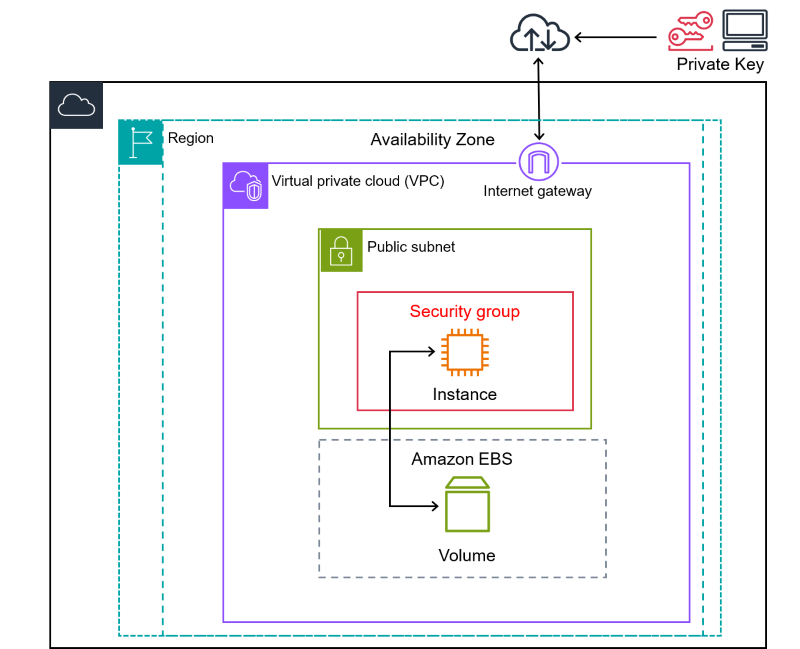


## Key Pairs

Secure login information for your instances. AWS stores the public key and you store the private key in a secure place.

## Elastic Cloud Compute (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.



## Relational Database Service (RDS)

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

### DB Subnet Groups

Subnets are segments of a VPC's Ip address range that you designate to group your resources based on security and operational needs. A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances. By using a DB subnet group, you can specify a particular VPC when creating DB instances using the AWS CLI.

Each DB subnet group should have subnets in at least two Availability Zones in a given AWS Region. When creating a DB instance in a VPC, you choose a DB subnet group for it. From the DB subnet group, Amazon RDS chooses a subnet and an Ip address within that subnet to associate with the DB instance. The DB uses the Availability Zone that contains the subnet.

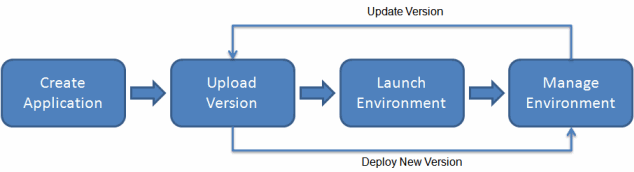
## Elastic Bean Stalk (EBS)

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, pHp, python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or ebc cli, a high-level CLI designed specifically for Elastic Beanstalk.



## Cloud Watch

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

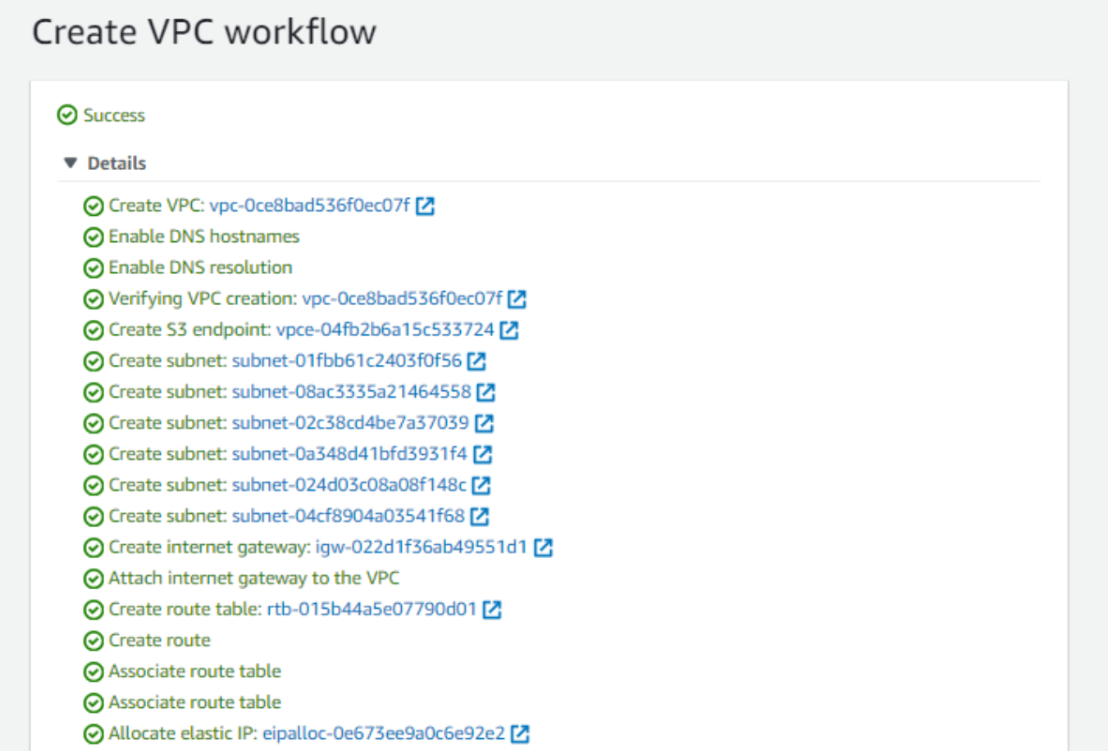
The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

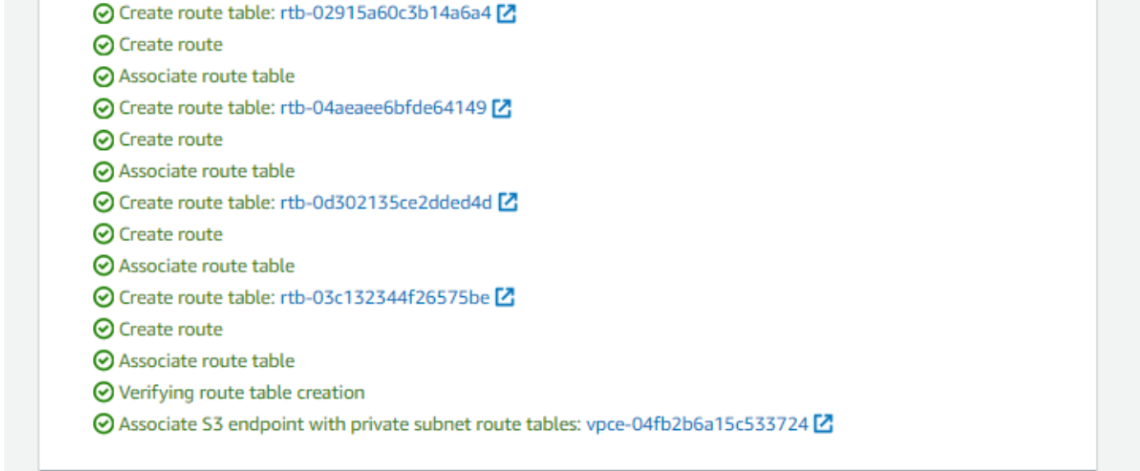
# Creation of VPC, Subnets, Route Tables, Internet Gateway

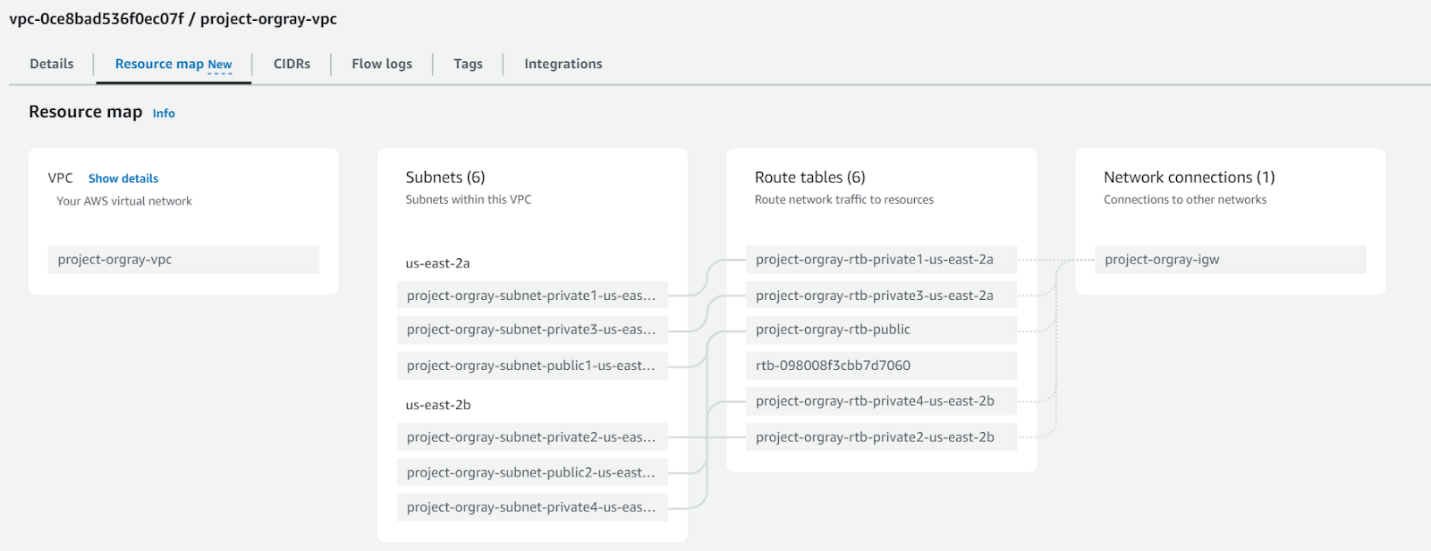
1. In the search box to the right of Services, search for and choose VPC to open the VPC console.
2. Begin creating a VPC.
   * In the top right of the screen, verify that Ohio (us-east-2) is the region.
   * Choose the VPC dashboard link towards the top left of the console.
   * Next, choose Create VPC
3. Configure the VPC details in the VPC settings panel on the left:
   * Choose VPC and more.
   * Under Name tag auto-generation, keep Auto-generate selected, however, change the value from project to project-orgray-VPC.
   * Keep the Ipv4 CIDR block set to 10.0.0.0/16
   * For the Number of Availability Zones, choose 2.
   * For the Number of public subnets, keep the 2.
   * For the Number of private subnets, keep the 4.
   * Keep both DNS hostnames and DNS resolution enabled.
4. In the preview panel on the right, confirm your configured settings

* VPC: project-orgray-VPC
* Subnets:
  + us-east-2a
    - public subnet name:
      * project-orgray-subnet-public1-us-east-2a
    - private subnet name:
      * project-orgray-subnet-private1-us-east-2a
      * project-orgray-subnet-private3-us-east-2a
  + us-east-2b
    - public subnet name:
      * project-orgray-subnet-public2-us-east-2b
    - private subnet name:
      * project-orgray-subnet-private2-us-east-2b
      * project-orgray-subnet-private4-us-east-2b
* Route tables
  + project-orgray-rtb-public
  + project-orgray-rtb-private1-us-east-2a
  + project-orgray-rtb-private2-us-east-2a
  + project-orgray-rtb-private3-us-east-2a
  + project-orgray-rtb-private4-us-east-2a
* Network connections
  + project-orgray-igw

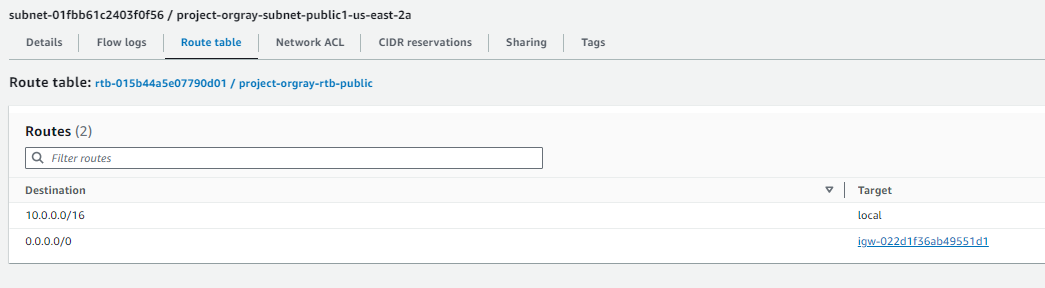
1. At the bottom of the screen, choose Create VPC



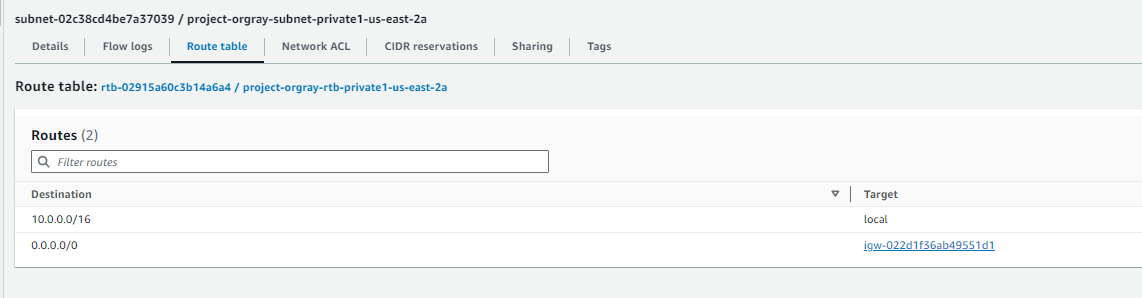




Public Route Table



Private Route Table



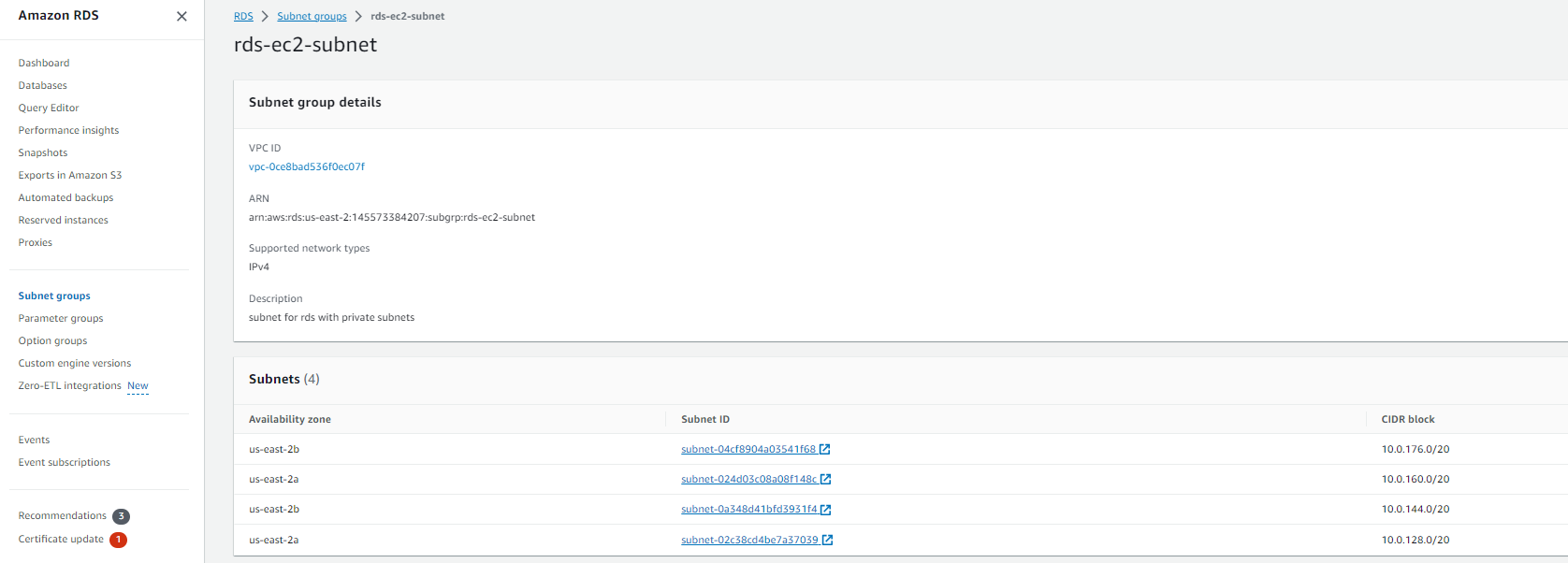
# Creation of Key-Pair

1. In the AWS Management Console, go to the "Services" dropdown and select "EC2" under the "Compute" section.
2. In the EC2 Dashboard, look for the "Network & Security" section on the left-hand side, and click on "Key Pairs."
3. Click the "Create Key Pair" button.
4. Give **oca** as key pair name.
5. Choose the key pair type. AWS supports both RSA and Ed25519 key types. We used RSA.
6. After creating the key pair, a private key (ppk) file will be downloaded automatically to your computer. This file is crucial for accessing your instances securely. Keep it in a secure location.

Note: AWS only allows you to download the private key file once. If you lose the private key, you'll need to create a new key pair.

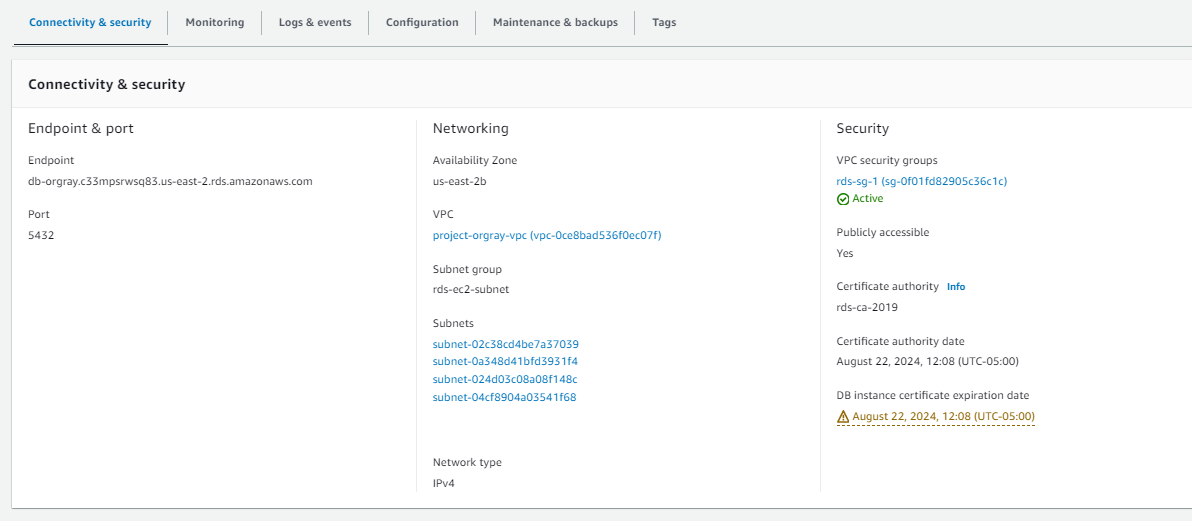
# Creation of DB Subnet Groups

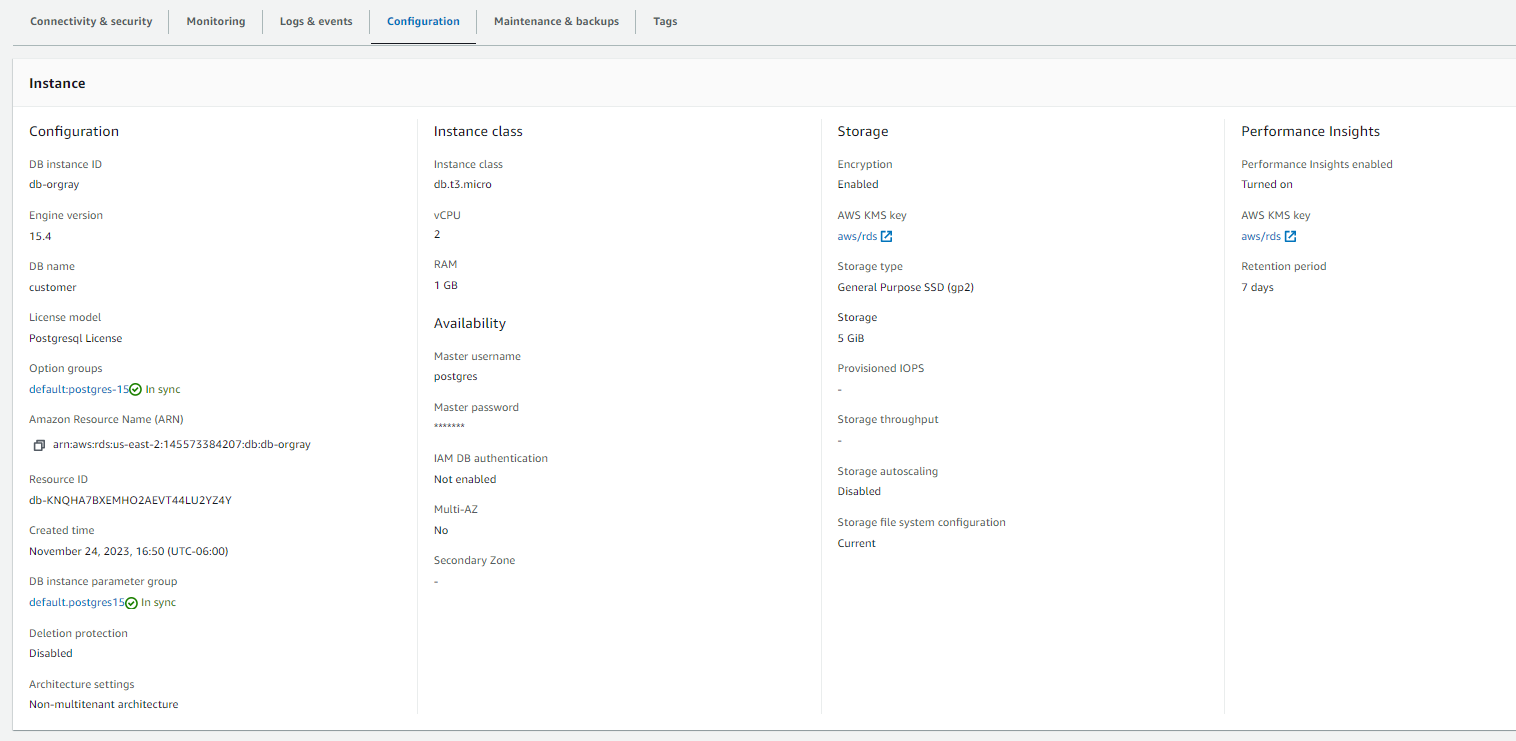
1. In the AWS Management Console, go to the "Services" and search Amazon RDS service and click on it.
2. In the left navigation pane, choose "Subnet groups."
3. click the "Create DB Subnet Group" button.
4. Provide a name for your subnet group as **rds-ec2-subnet**.
5. Select the VPC that you have already created **project-orgray-vpc**.
6. Select the Availability zones and select us-east-2a & us-east-2b.
7. In the "Add Subnets" section, add the private subnets where you want your RDS instance to be placed.
8. Select the private subnets you want to add to the subnet group.
   * project-orgray-subnet-private1-us-east-2a
   * project-orgray-subnet-private2-us-east-2b
   * project-orgray-subnet-private3-us-east-2a
   * project-orgray-subnet-private4-us-east-2b
9. Ensure that you choose subnets from different Availability Zones for high availability.
10. Review the configuration details.
11. Click the "Create" button to create the subnet group.



# Creation of Relational Database Service using Postgres

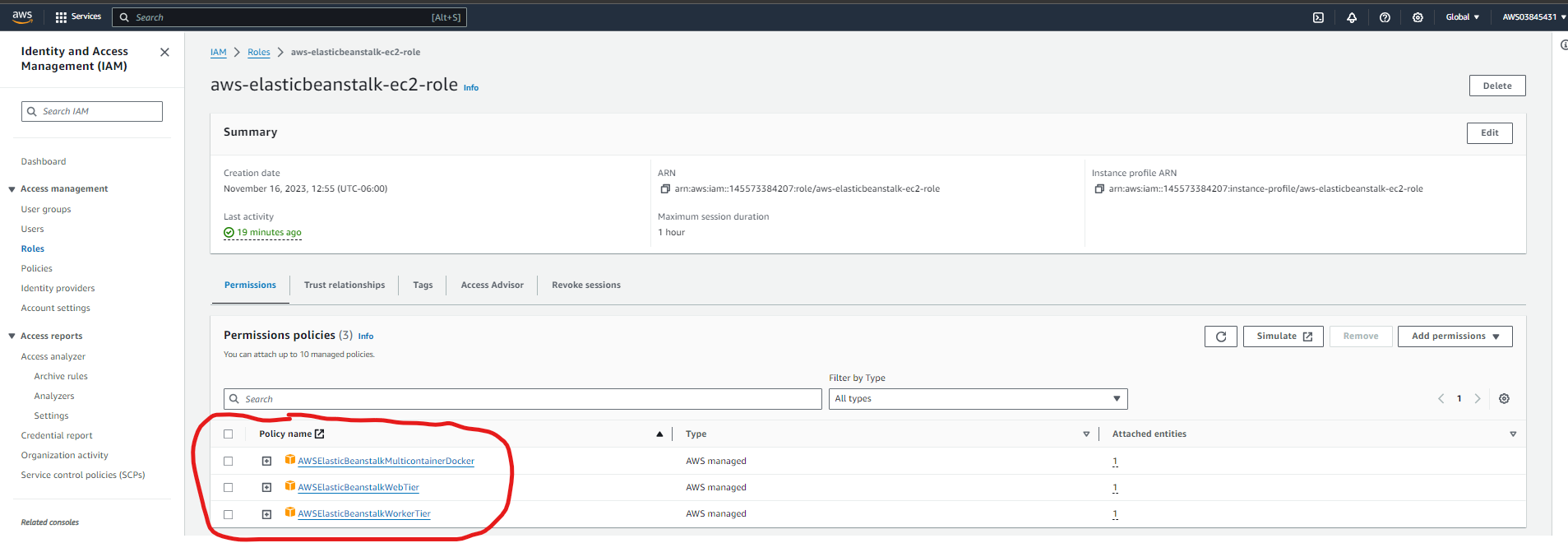
1. AWS Management Console and navigate to the Amazon RDS service.
2. Click on the "Create database" button.
3. Choose the "Standard create" option for more configuration options.
4. Select "PostgreSQL" as the database engine.
5. From Templates choose free tier.
6. DB Cluster Identifier: Provide a name as db-orgray for your RDS instance.
7. Master Username and Password: Set the master username as admin and password for your PostgreSQL database.
8. Choose the instance class as db.t3.micro.
9. Set the allocated storage for your RDS instance.
   * Storage type: General Purpose SSD (gp2)
   * Storage: 5 GiB
10. In Connectivity, under compute resource select Don’t connect to an ec2 Compute resource.
11. Select the VPC that you have already created project-orgray-vpc.
12. Create a new security group for rds with name **rds-sg-1**
13. Choose the subnet group you created in the previous steps rds-ec2-subnet.
14. In the "Public access", select "Yes" to make your RDS instance publicly accessible.
15. In connectivity, expand additional configuration and set database port as 5432.
16. In Additional Configuration set initial database name as customer.
17. Configure other settings such as backup retention as 7 days.
18. Choose enable encryption for your RDS instance. Configure the encryption settings if needed.
19. Review your configurations, and if everything looks correct, click the "Create database" button.





# Creation of IAM role for EC2 Instance

1. Go to the AWS Management Console.
2. Open the IAM console.
3. In the left navigation pane, choose "Roles."
4. Click on "Create role."
5. Choose "AWS service" as the type of trusted entity.
6. Choose Elastic Beanstalk as the use case.
7. Attach the following policies
   * AWSElasticBeanstalkWebTier
   * AWSElasticBeanstalkWorkerTier
   * AWSElasticBeanstalkMulticontainerDocker
8. Review the role details and provide a name as **aws-elasticbeanstalk-ec2-role.**
9. Click "Create role."



# Creation of Security Group for Elastic Beanstalk

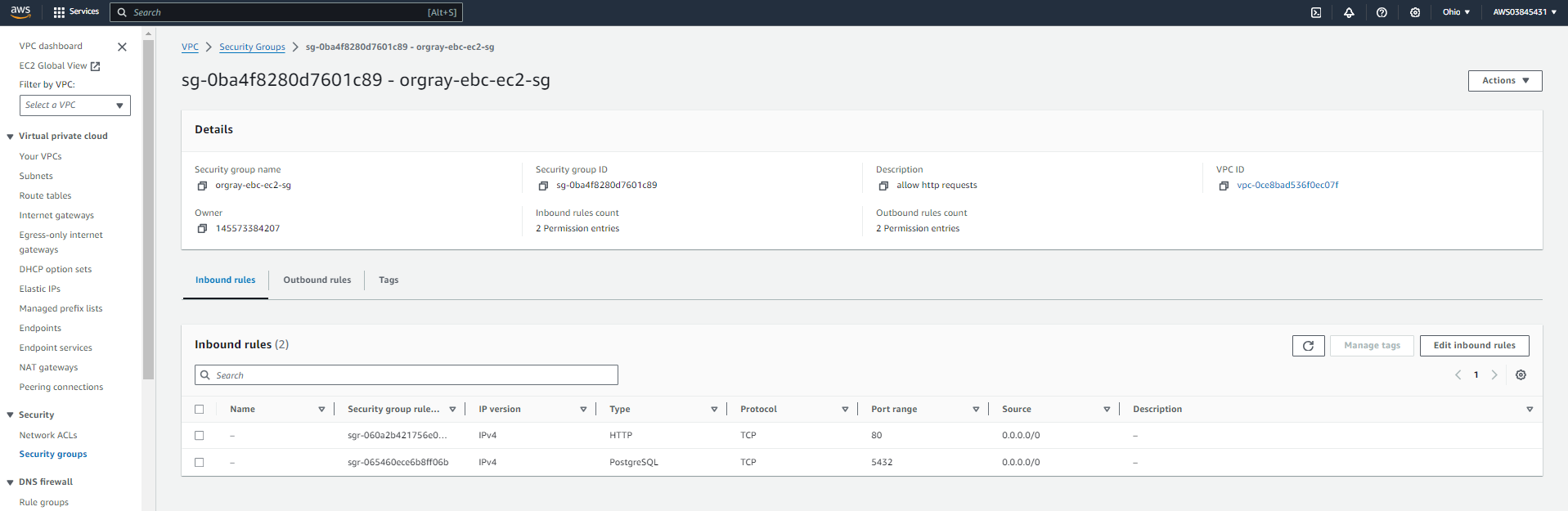
1. In the AWS Management Console, go to the "Services" dropdown and select "EC2" under the "Compute" section.
2. In the EC2 Dashboard, look for the "Network & Security" section on the left-hand side, and click on "Security Groups."
3. Click on the "Create Security Group" button.
4. Provide name as **orgray-ebc-ec2-sg** for security group.
5. Add a description to help identify the purpose of the security group.
6. Inbound Rules

* Add inbound rules to control incoming traffic. Click on the "Add Rule" button.
* Choose the type of rule HTTP and port range is 80. Set the source for the rule as 0.0.0.0/0
* Add another inbound rule with type as PostgresSQL and port as 5432 with source as 0.0.0.0/0

1. Outbound Rules

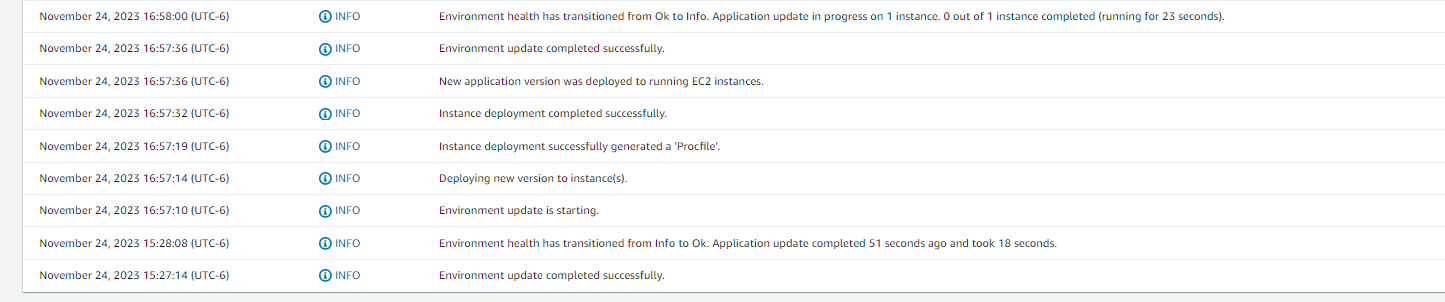
* Add outbound rules to control incoming traffic. Click on the "Add Rule" button.
* Choose the type of rule HTTP and port range is 80. Set the destination for the rule as 0.0.0.0/0
* Add another outbound rule with type as PostgresSQL and port as 5432 with destination as 0.0.0.0/0

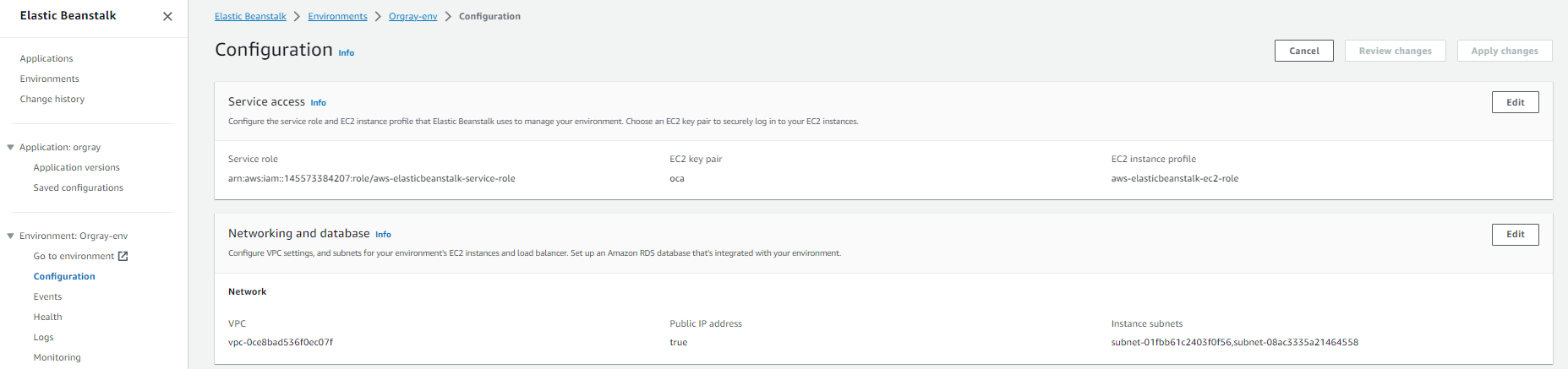
1. Click the "Create" button to create the security group.

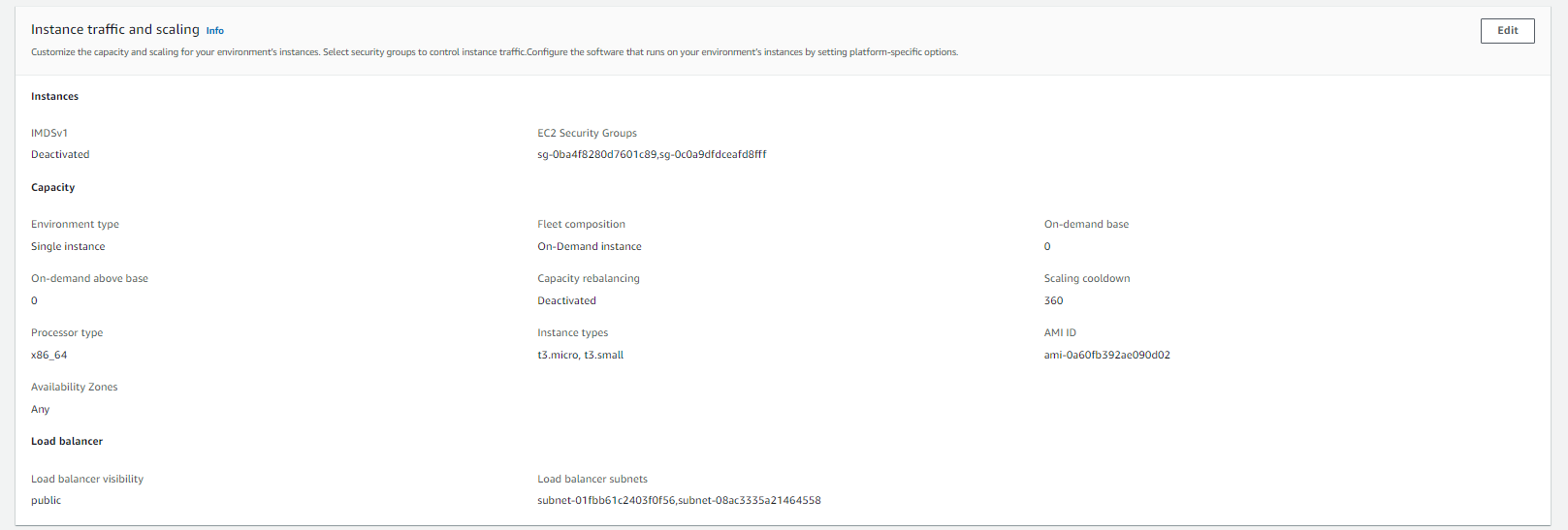


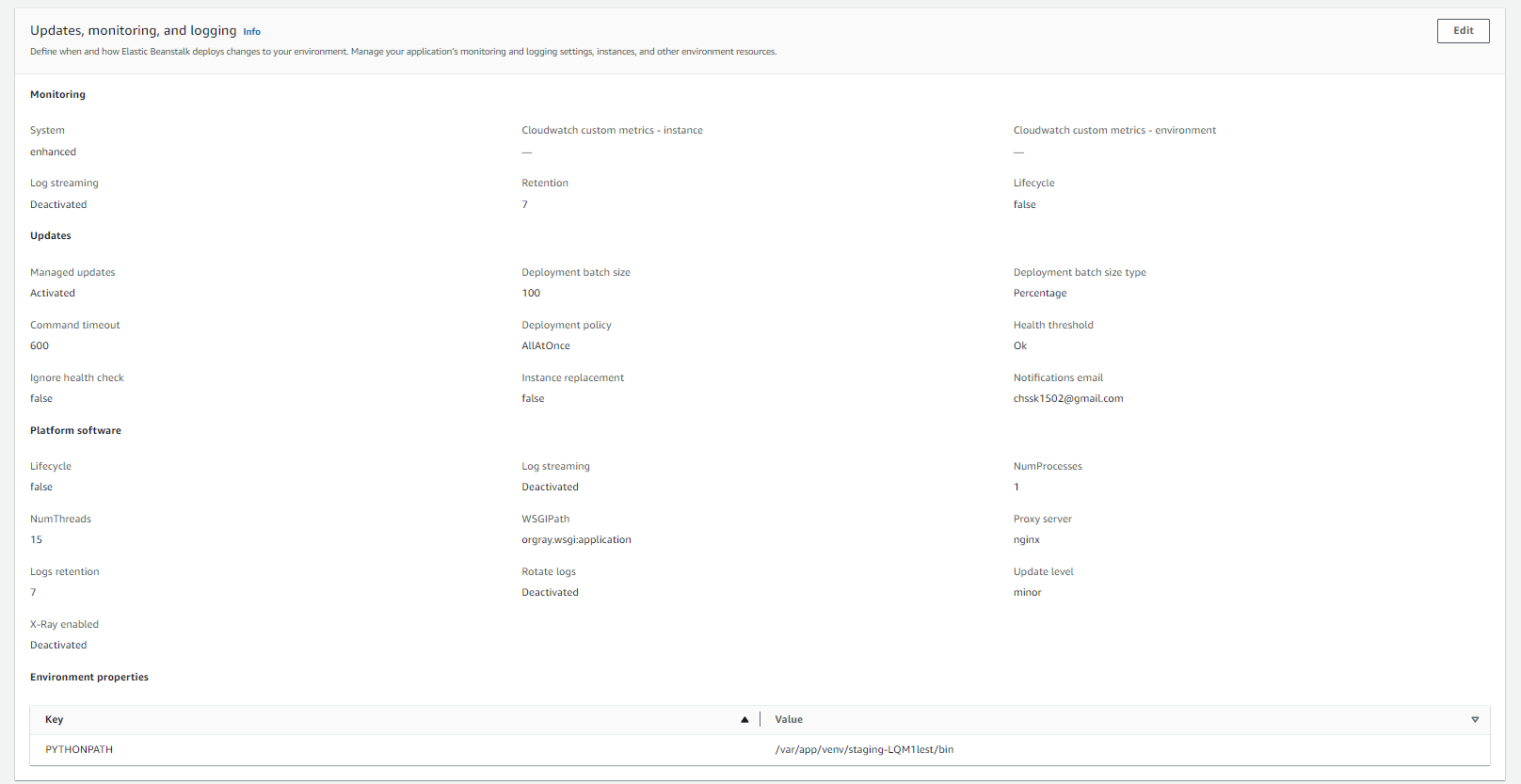
# Creation of ElasticBeanStalk

1. In the AWS Management Console, go to the "Services" dropdown and select "Elastic Beanstalk" under the "Compute" section.
2. Click on the "Create Application" button.
3. Provide a name as orgray for your application.
4. Optionally, add a description for your application.
5. Click on "Create."
6. Within your newly created application, click on the "Create Environment" button.
7. Choose "Web server environment" for a web application.
8. In Environment Information, Provide a name as Orgray-env for your environment.
9. In Platform selection, Choose "Python" as the platform.
10. Choose Python 3.11 running on 64bit Amazon Linux 2023 as platform branch.
11. Choose 4.0.6 platform version.
12. In Application Code, select sample application as option and at the bottom of page click on next.
13. In service role, choose create and use new service role. A service role will be created with name as **aws-elasaticbeanstalk-service-role** with following permissions
    * AWSElasticBeanstalkEnhancedHealth
    * AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy
14. In EC2 key pair, choose a key pair **oca**, which we created earlier.
15. In EC2 instance profile, choose IAM role as **aws-elasaticbeanstalk-ec2-role**, which we created earlier.
16. Choose VPC as project-orgray-vpc, which we created earlier.
17. In instance settings, select activated under public IP address.
18. Choose the two public subnets in instance subnets.
    * project-orgray-subnet-public-us-east-2a
    * project-orgray-subnet-public-us-east-2b
19. At the bottom of page click on next.
20. Choose the security group as **orgray-ebc-ec2-sg**
21. In capacity section, choose environment type as Single instance and choose fleet composition as On-Demand instance.
22. Select instance type as t3.micro and t3.small.
23. Choose health reporting system as Enhanced.
24. Select following metrics to monitor the instance and environment.
    * Cloud watch Custom Metrics – Instance
    * Cloud watch Custom Metrics – Environment
25. Enter an email address to receive email notifications for important events from your environment
26. In platform software, choose proxy server as Ngnix.
27. Set environment properties
    * : PYTHONPATH
    * Value: /var/app/venv/staging-LQM1lest/bin
28. At the bottom click next.
29. Review the configuration and click on the submit button.



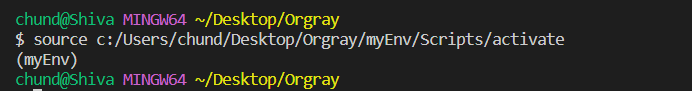




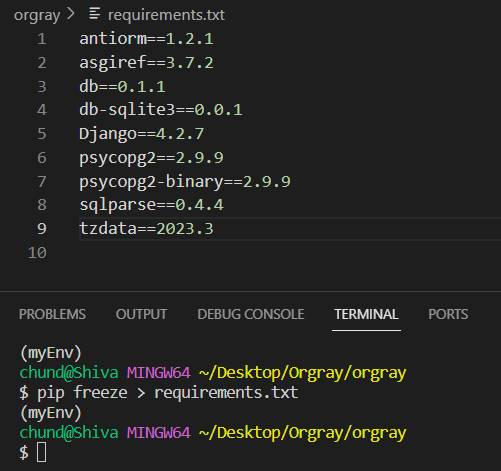


# Application Deployment in Elastic Bean Stalk

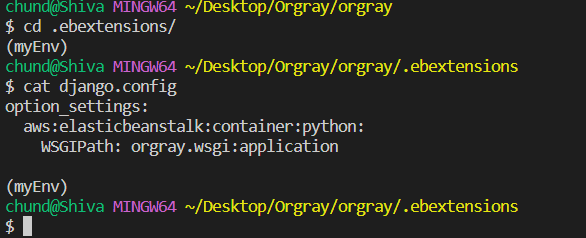
1. Activate your virtual environment.



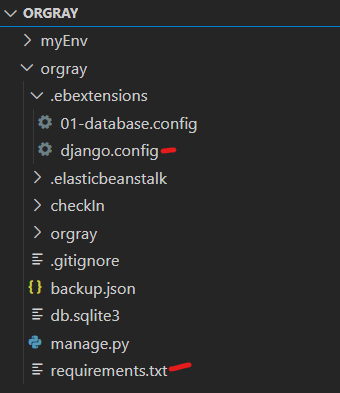
1. Run pip freeze, and then save the output to a file named requirements.txt.



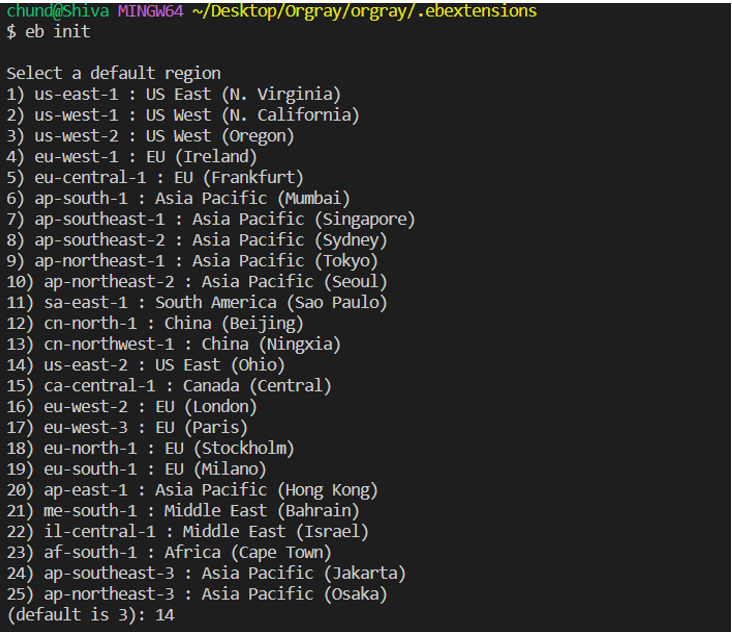
1. Create a directory named .ebextensions.
2. In the .ebextensions directory, add a configuration file named django.config with the following text.
3. This setting, WSGIPath, specifies the location of the WSGI script that Elastic Beanstalk uses to start your application.



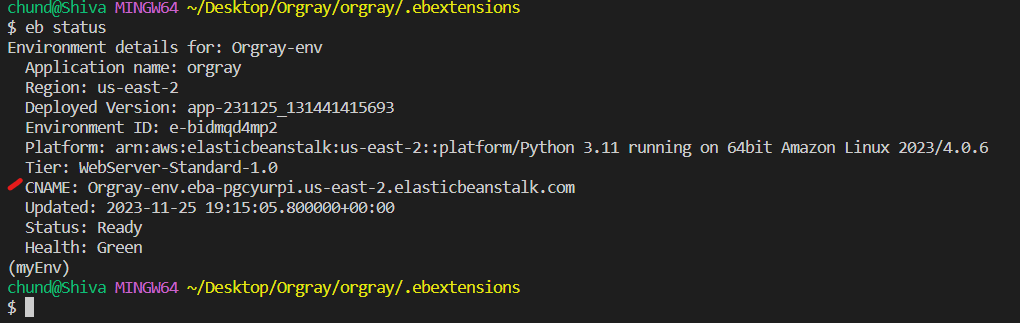
1. You've added everything you need to deploy your application on Elastic Beanstalk. Your project directory should now look like this.



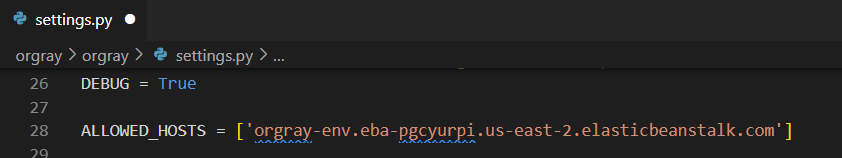
1. Initialize your EB CLI repository with the eb init command.



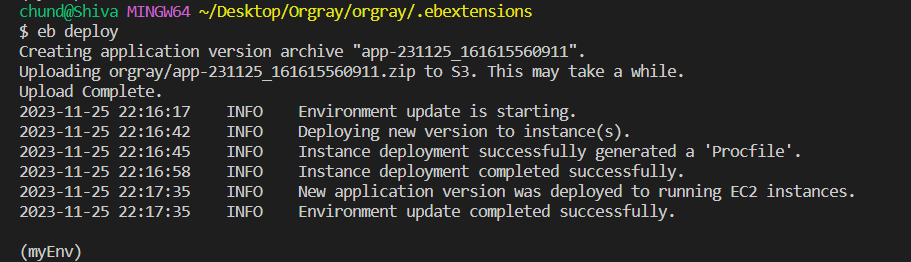
1. Find the domain name of your new environment by running eb status



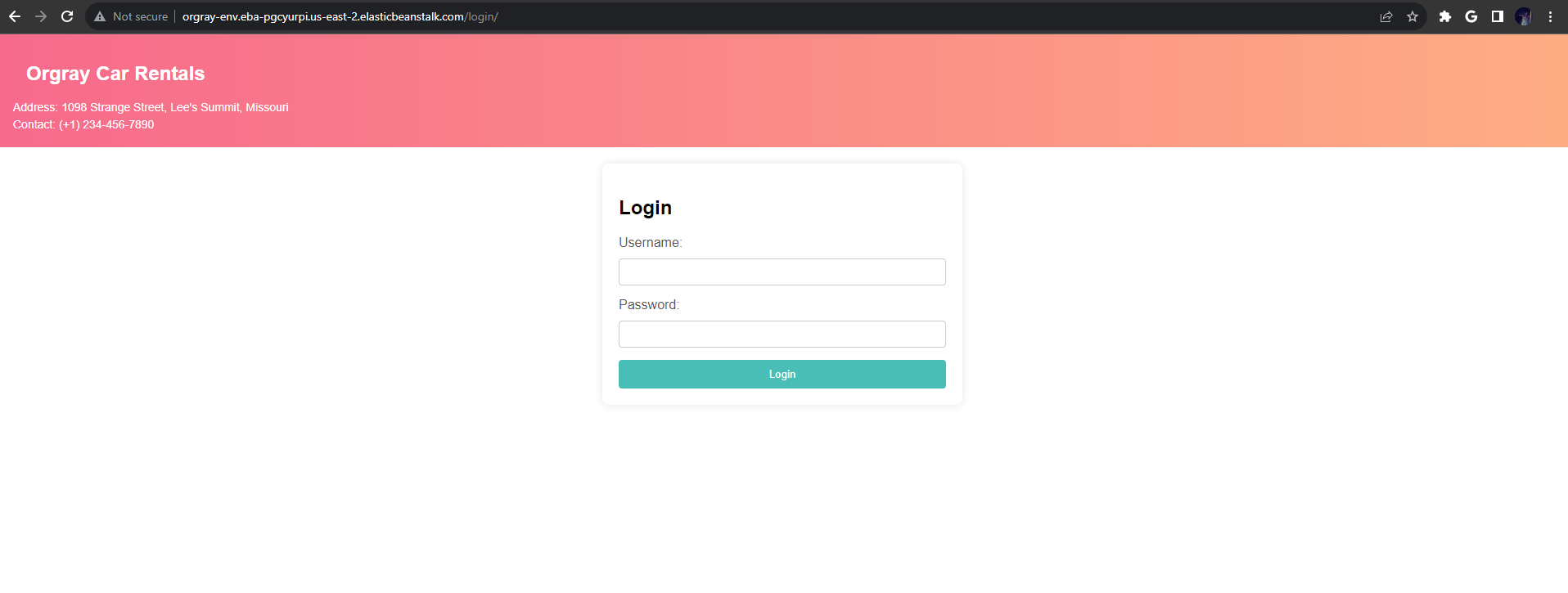
1. Open the settings.py file in the orgray directory. Locate the ALLOWED\_HOSTS setting, and then add your application's domain name that you found in the previous step to the setting's value. If you can't find this setting in the file, add it to a new line.



1. Save the file, and then deploy your application by running eb deploy. When you run eb deploy, the EB CLI bundles up the contents of your project directory and deploys it to your environment.

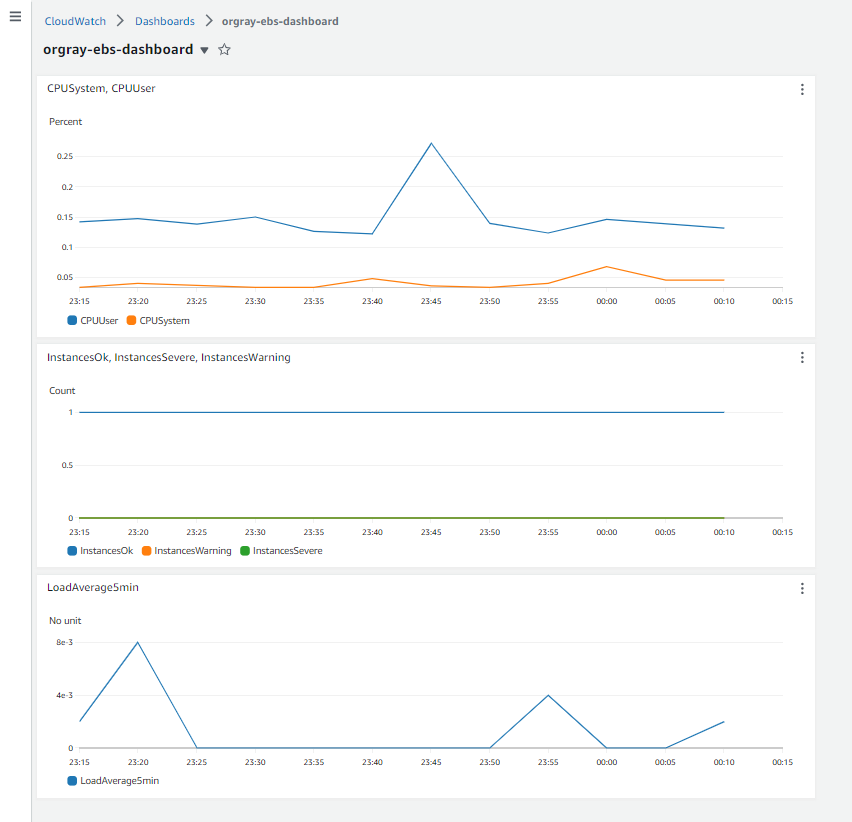


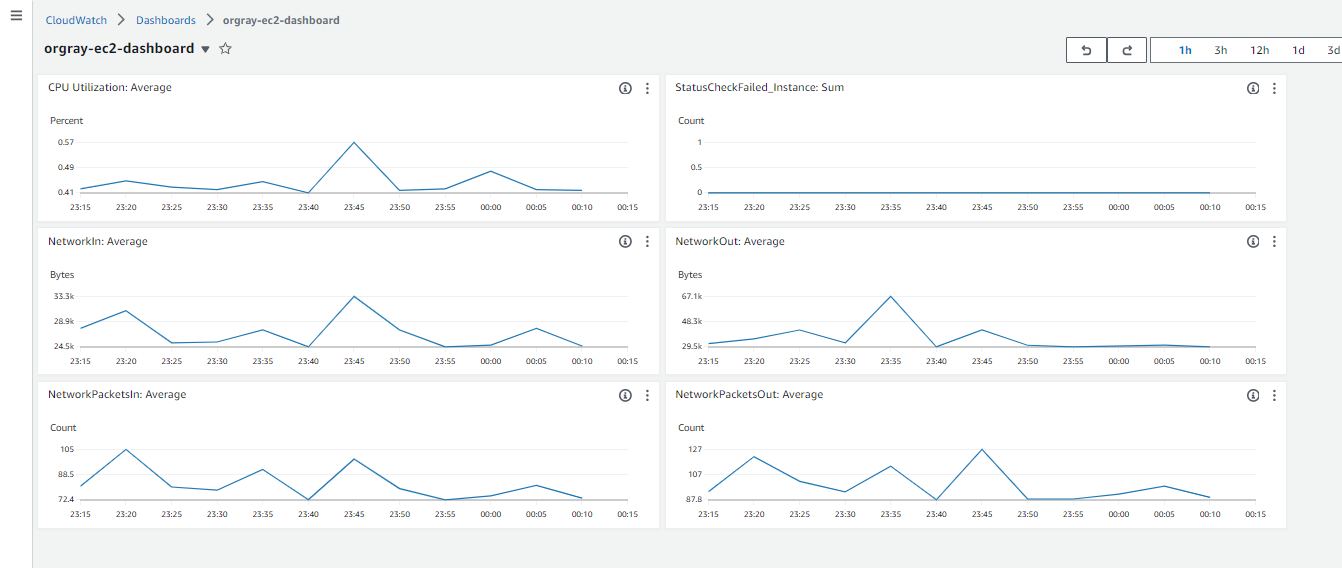
1. When the environment update process completes, open your website with eb open.



# Creation of Dashboards in Cloud Watch

1. In the AWS Management Console, go to the "Services" dropdown and select "CloudWatch" under the "Management & Governance" section.
2. In the CloudWatch console, click on "Dashboards" in the left navigation pane.
3. Click on the "Create dashboard" button.
4. Provide a name for your dashboard in the "Dashboard name" field.
5. Click on the "Add widget" button to add visualizations to your dashboard.
6. Select the type of widget you want to add (e.g., Line, Stacked Area, etc.).
7. Configure the widget settings, including the metric, period, and other relevant options.
8. Choose the metrics you want to display in the widget. You can select metrics from various AWS services, including EC2, S3, RDS, Lambda, and more.
9. Drag and drop the widgets to arrange them in the desired layout on your dashboard.
10. Click on the "Save dashboard" button to save your configuration.
11. Once saved, you can view and edit your dashboard at any time by selecting it from the list of dashboards in the CloudWatch console.





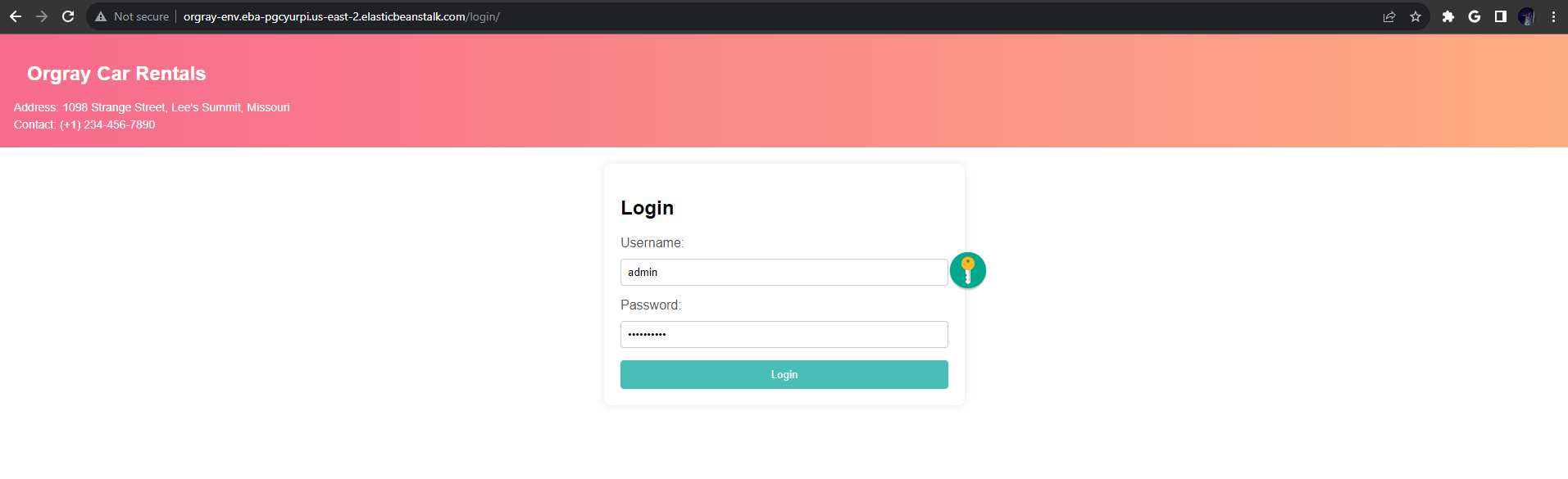
# Application Overview

Orgray Car Booking System is an online car rental application designed for rental shops. The system is deployed on AWS using Elastic Beanstalk. The application consists the following pages: Log-in, Home, Check-In, Current Bookings, Upload Car, Edit car Details Check-Out and Check-Out Summary.

## 

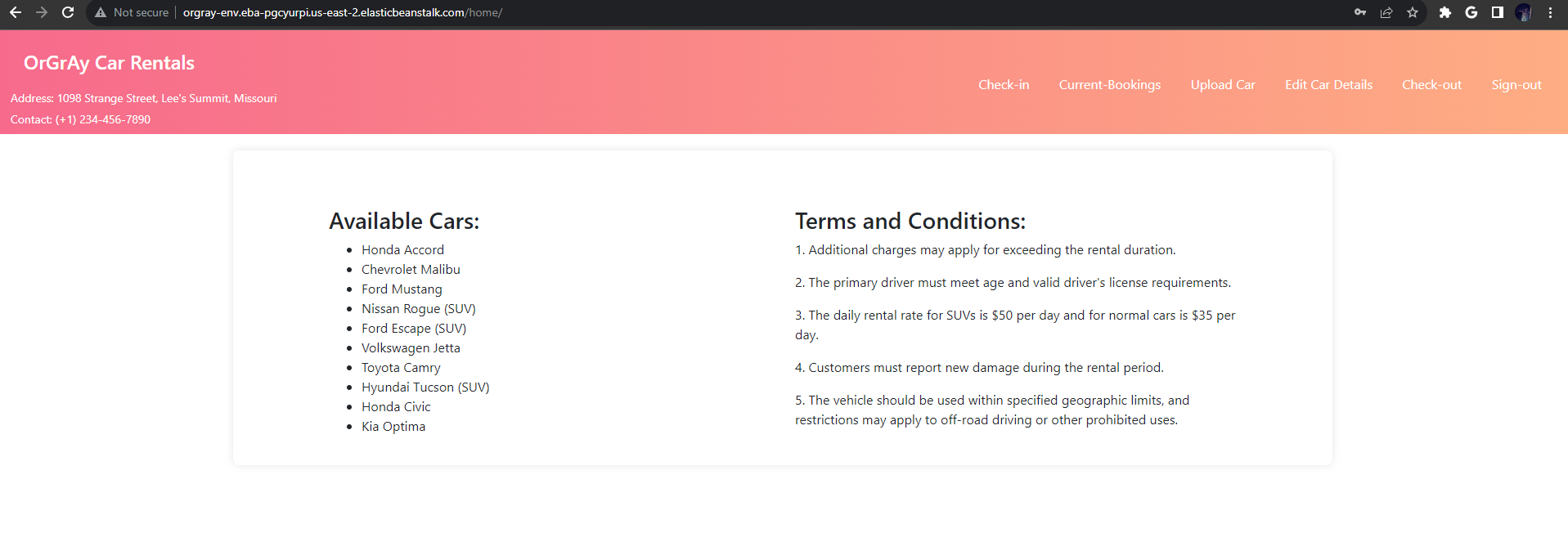
## Login Page

* Application Url: <http://orgray-env.eba-pgcyurpi.us-east-2.elasticbeanstalk.com/login>
* User Name: admin, Password: admin@1234
* Once credentials are validated then you will be navigated to home page.



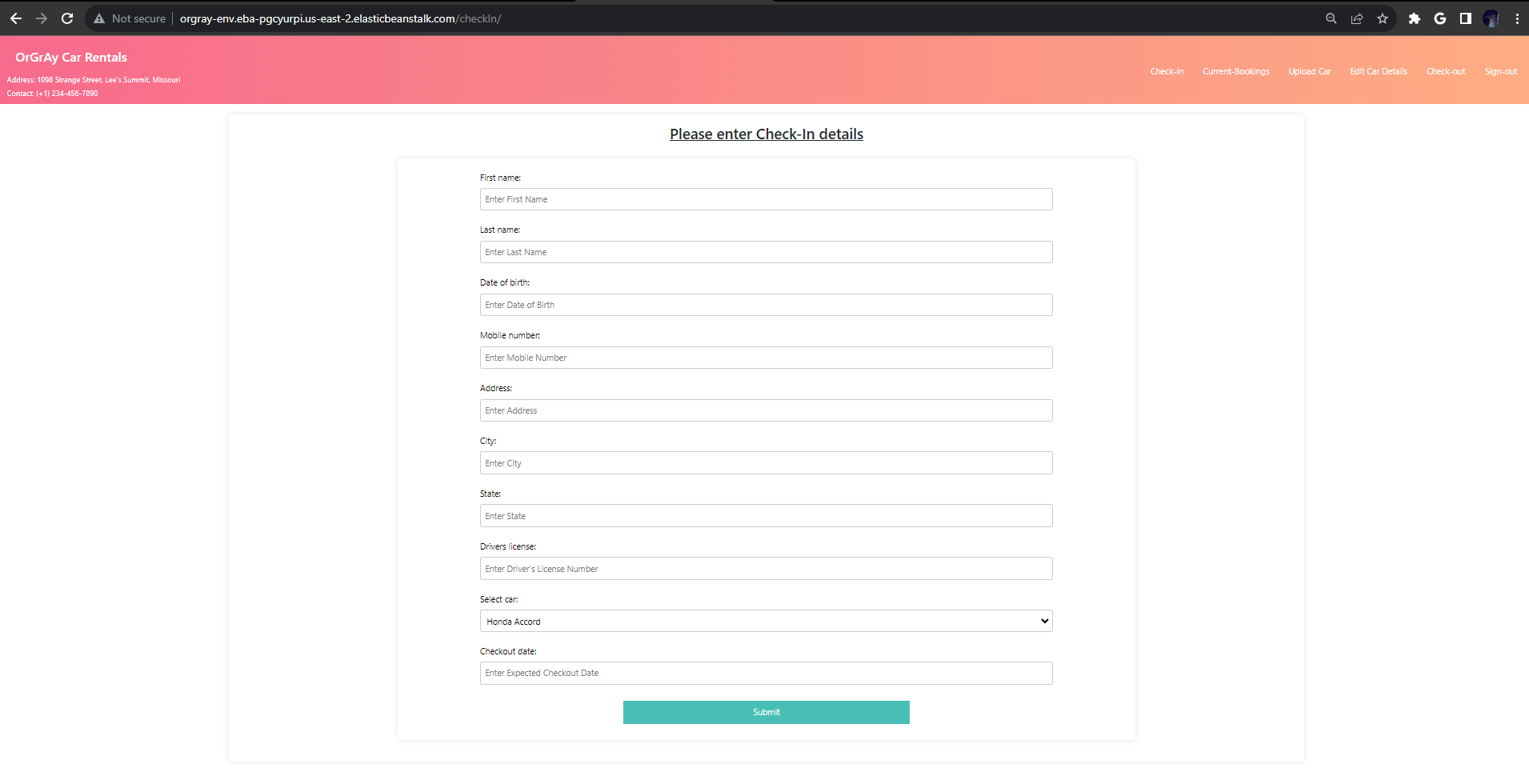
## Home Page

* This page consists of Available cars list which are ready for check-in.
* Terms and conditions are displayed for user friendly purpose to stress these rules to customers
* It also has the links that re-direct to check-In, check-out, current-bookings, upload car, edit car details, and log-out.



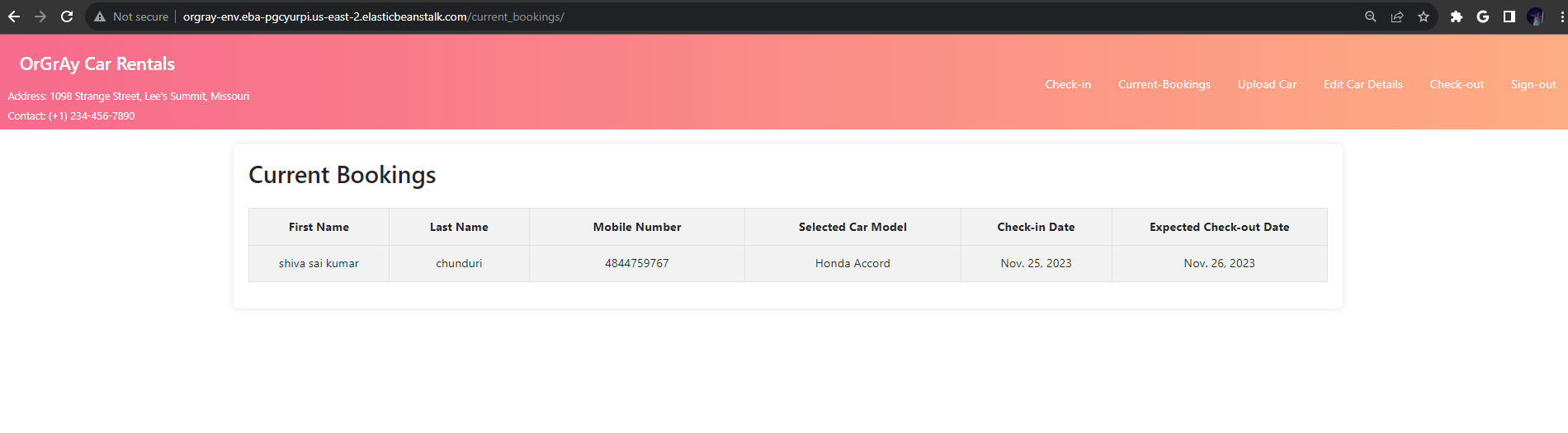
## Check-In Page

* If customer wants to rent a car, then we need to open check-in page and need to fill the required fields.
* Once you submit the form, the details will under go validation and if they are valid then check-in will be successful and data will be stored in database else validation errors will be displayed on the top of form.



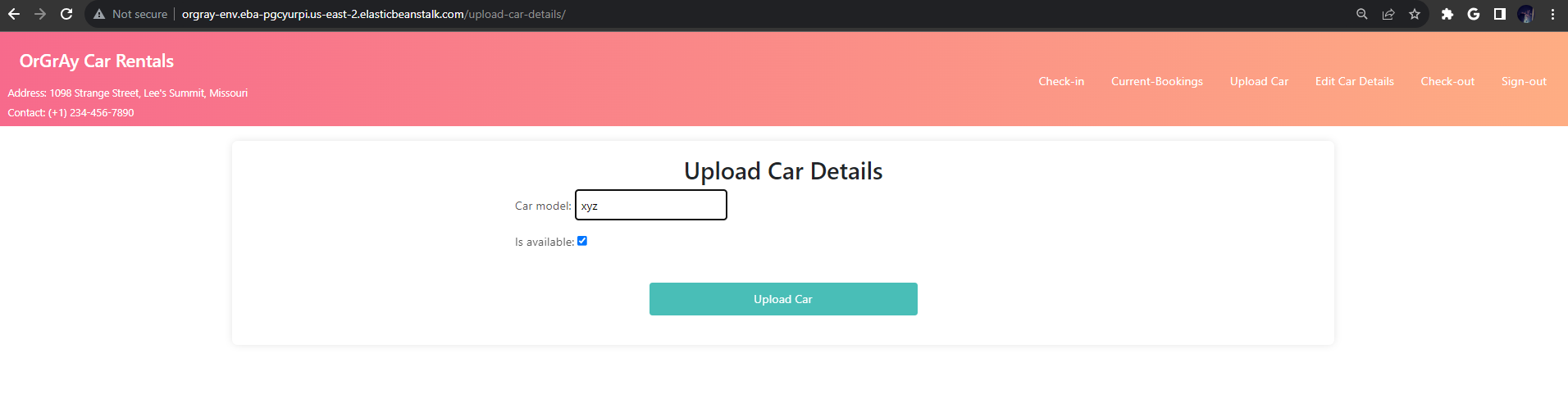
## Current Bookings Page

* Current bookings will be displayed in this page.

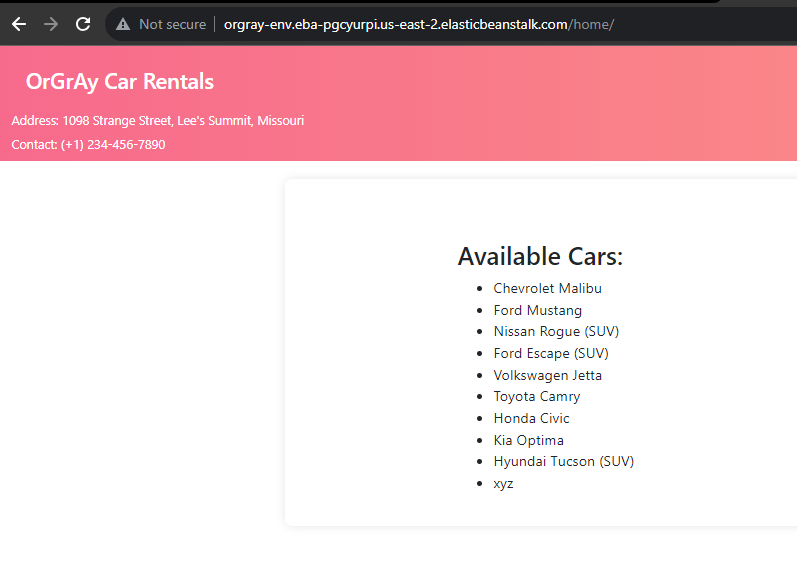


## Upload Car Page

* If owner wants to add a new car to his rental store, then he uploads that car details in this page.
* Is available check box representing the availability of car for rental purpose. If you don’t mark it then that car won’t be displayed in available cars list in home page as well as in check in page.

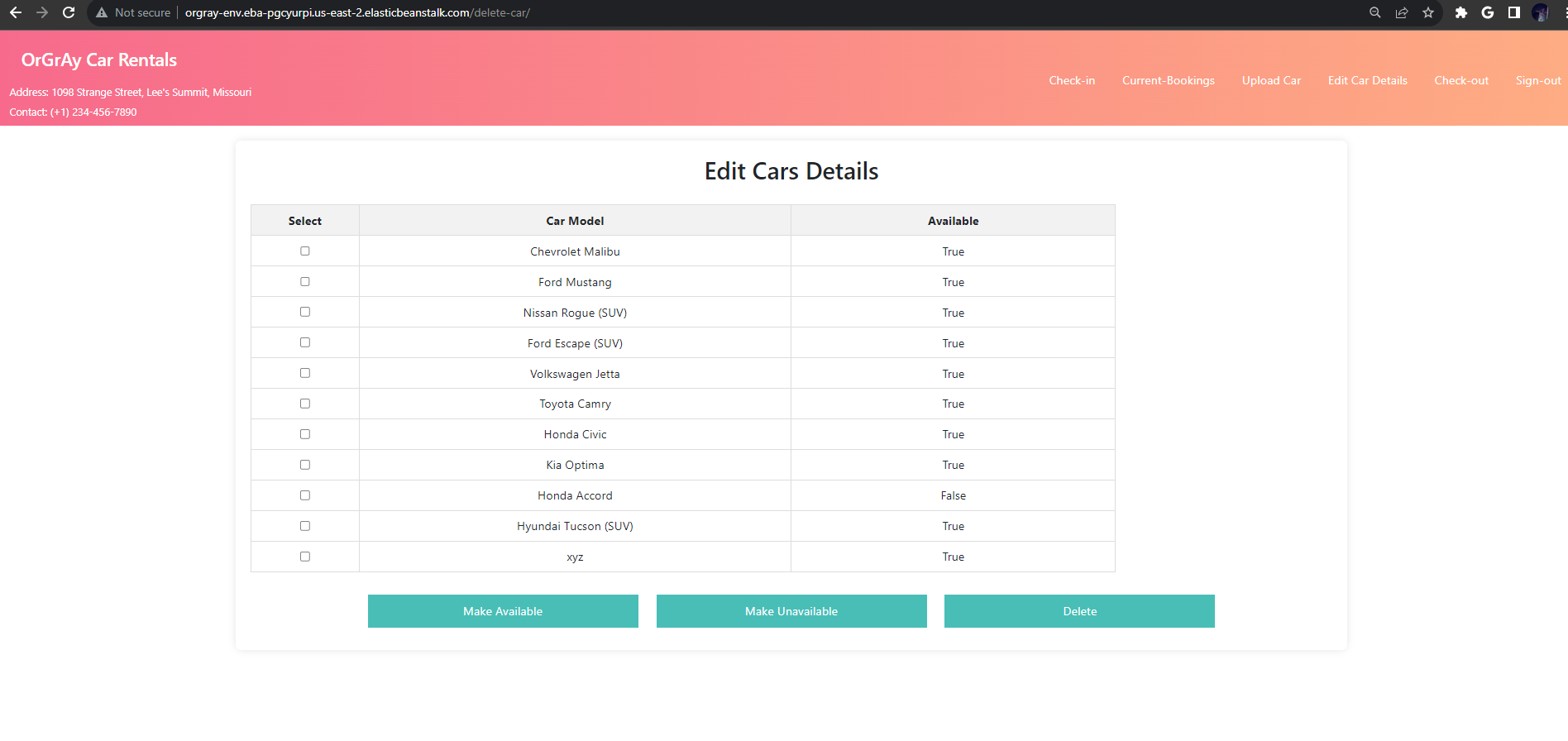


* From the above image, I added new car XYZ with marked as available.
* From below image that XYZ car is displayed in available cars list.



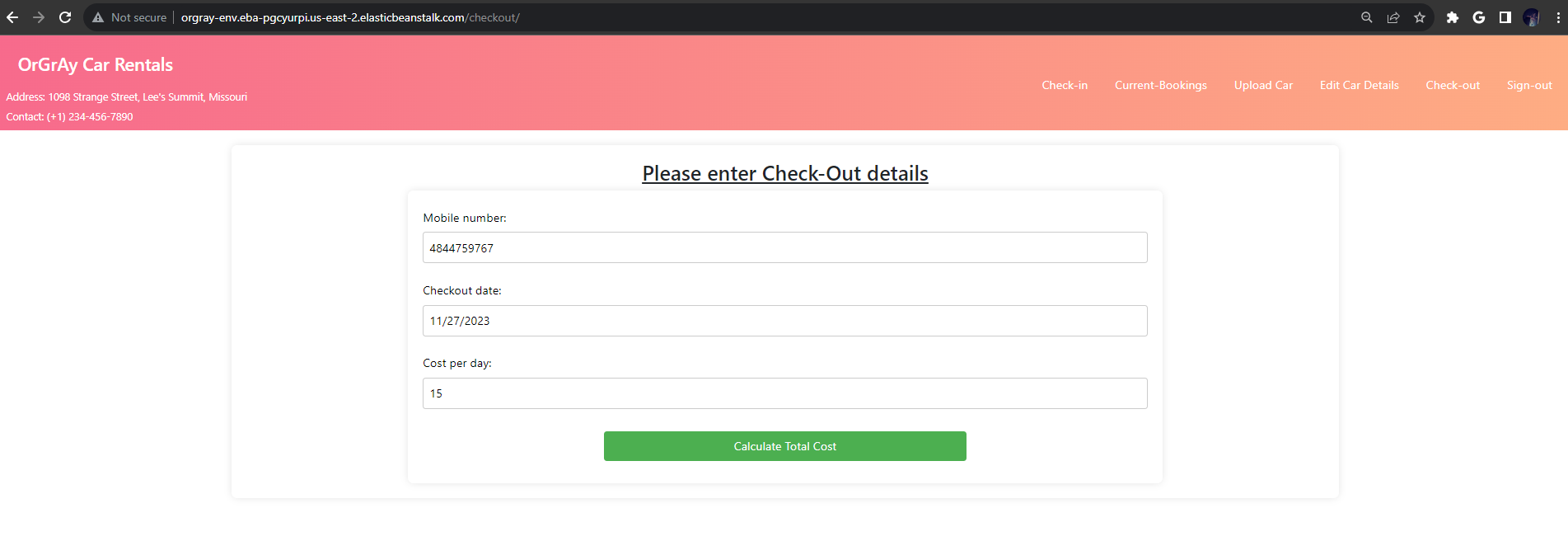
## Edit Car Details Page

* This page contains the editing of car details with two cases.
  + If owner don’t want to show few cars in available page due to x reasons. It may be car repair or else anything, then he can disable them in this page. He can also make enable them like vice versa.
  + If owner want to remove that car from rental store, he can delete it. Once it is deleted, that cannot be restored.



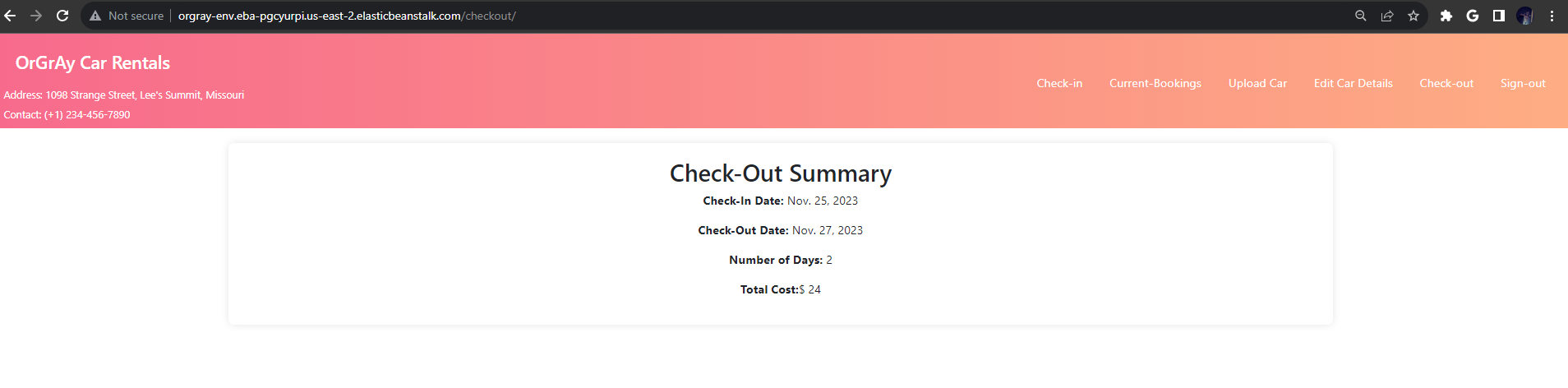
## Check-out Page

* If user want to return the car, then we need to click on check-out link.
* We need to enter the mobile number of users, checkout date and cost of rent car per day. Based on mobile number application can fetch his history and take check-in date into consideration to calculate the final amount to be paid. And it will re-direct to the check-out summary page.



## Check-Out Summary Page

* This page contains the check-out details if user. His check-in date, check-out date and total number of days the car is in rent, and total cost that user need to pay.
* Once this is shown then user details we will delete from the database. We are not storing any user’s check-out history.
* The car which user is returned will be displayed in available page and also in check in page.



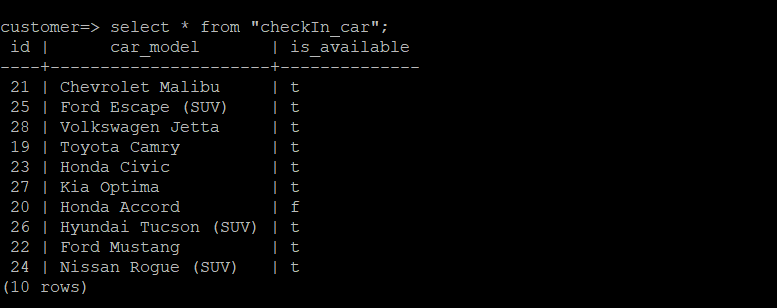
## Database Verification

We will be connecting to the PostgreSQL database and will be verifying the data

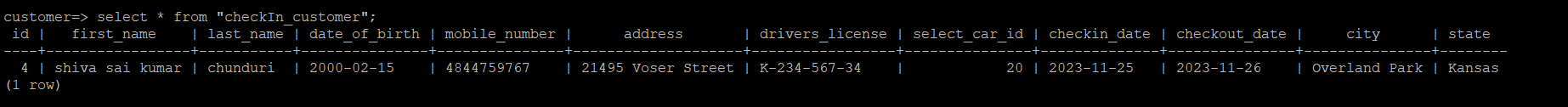
**Tables**



**Cars Data**



**Customer Data**



# Conclusion

In conclusion, the outlined AWS architecture exhibits a robust and secure design. Leveraging a single VPC with strategically placed public and private subnets across multiple availability zones enhances fault tolerance and scalability. The association of subnets with a single internet gateway streamlines network management. Elastic Beanstalk simplifies application deployment, deploying EC2 instances in public subnets for accessibility and redundancy. The use of a private RDS subnet group ensures secure data storage, with dedicated security groups for EC2 and RDS enhancing isolation. CloudWatch dashboards offer comprehensive monitoring, providing insights into the performance of both EC2 instances and RDS databases, ensuring optimal application health.

# References

1. [Amazon VPC Documentation](https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html)
2. [Amazon EC2 Concepts Documentation](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html)
3. [Amazon RDS User Guide](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html)
4. [AWS Elastic Beanstalk Documentation](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html)
5. [Install the EB CLI](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html)
6. [Amazon CloudWatch Documentation](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html)
7. [Django Documentation](https://docs.djangoproject.com/en/4.2/)
8. [Django Tutorial](https://docs.djangoproject.com/en/4.2/intro/tutorial01/)
9. [Deploying a Django application to Elastic Beanstalk](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html)