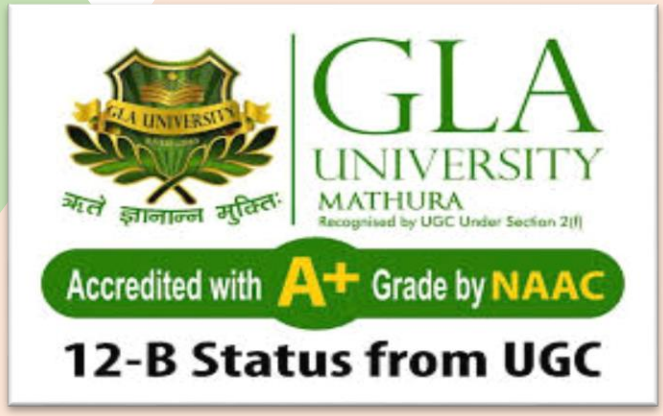
GLA UNIVERSITY



Department of Computer Science and Engineering

JOVAC – Project Report

Project Title

Multi-website hosting on Amazon Linux Server in Mumbai region

Mentor: Raushan Kumar Singh

Submitted By:

Prakhar Shukla (Leader)	2215001261
Sakshi Tangar	2215001558
Ankur Sinsinwar	2215000273
Anuj Chaturvedi	2215000302
Tanisha Singh	2215001832

TABLE OF CONTENTS

3. ## Setting Up AWS EC2 Instance

- Launching an EC2 Instance
- Connecting to Your EC2 Instance

4. ## Configuring the Linux Server

- Updating and Securing Your Server
- Installing Necessary Software (Apache, Nginx, etc.)

5. ## Setting Up Virtual Hosts

- Configuring Apache Virtual Hosts
- Configuring Nginx Server Blocks

6. ## Domain Management with Route 53

- Setting Up a Hosted Zone
- Configuring DNS Records for Multiple Domains

7. ## Deploying Websites

- Uploading Website Files
- Configuring Website Settings

8. ## Security Best Practices

- Setting Up Firewalls
- Implementing SSL Certificates

9. ## Monitoring and Maintenance

- Setting Up Monitoring Tools
- Regular Maintenance Tasks

10. ## Troubleshooting

- Common Issues and Solutions
- Useful Commands and Logs

Introduction

Overview of Multi-Website Hosting: -

Multi-website hosting refers to the practice of hosting more than one website on a single server. This approach can be cost-effective and efficient, especially for small to medium-sized websites. AWS EC2, combined with Route 53, provides a powerful solution for hosting multiple websites with high availability and scalability.

Benefits of Using AWS EC2 and Route 53: -

- **Scalability:** Easily scale your resources up or down based on demand.
- **Cost-Effective:** Pay only for the resources you use.
- **Flexibility:** Full control over your server environment.
- **Reliability:** High availability and redundancy with AWS infrastructure.

Required Software and Tools:

AWS Services:

- **EC2 (Elastic Compute Cloud):**
  - Create and manage Amazon Linux instances.
  - Choose an appropriate instance type based on your workload (e.g., t2.micro for light loads, m5.large for more demanding applications).
  - Set up security groups to control inbound and outbound traffic.
- **Elastic Load Balancing (ELB):**
  - Distribute traffic across multiple EC2 instances to ensure high availability and fault tolerance.
- **Auto Scaling:**
  - Automatically scale your instances up or down based on demand to optimize costs and performance.
- **Amazon RDS (Relational Database Service):**
  - Managed database service that supports MySQL, PostgreSQL, MariaDB, and more.
  - Simplifies database management tasks like backups, patching, and scaling.

- **Amazon S3 (Simple Storage Service):**
  - Store static assets, backups, or other data that need to be accessed by your applications.

- **Amazon CloudFront:**
  - Content Delivery Network (CDN) to deliver your content with low latency.

- **Amazon Route 53:**
  - Domain Name System (DNS) web service to route end users to your applications.

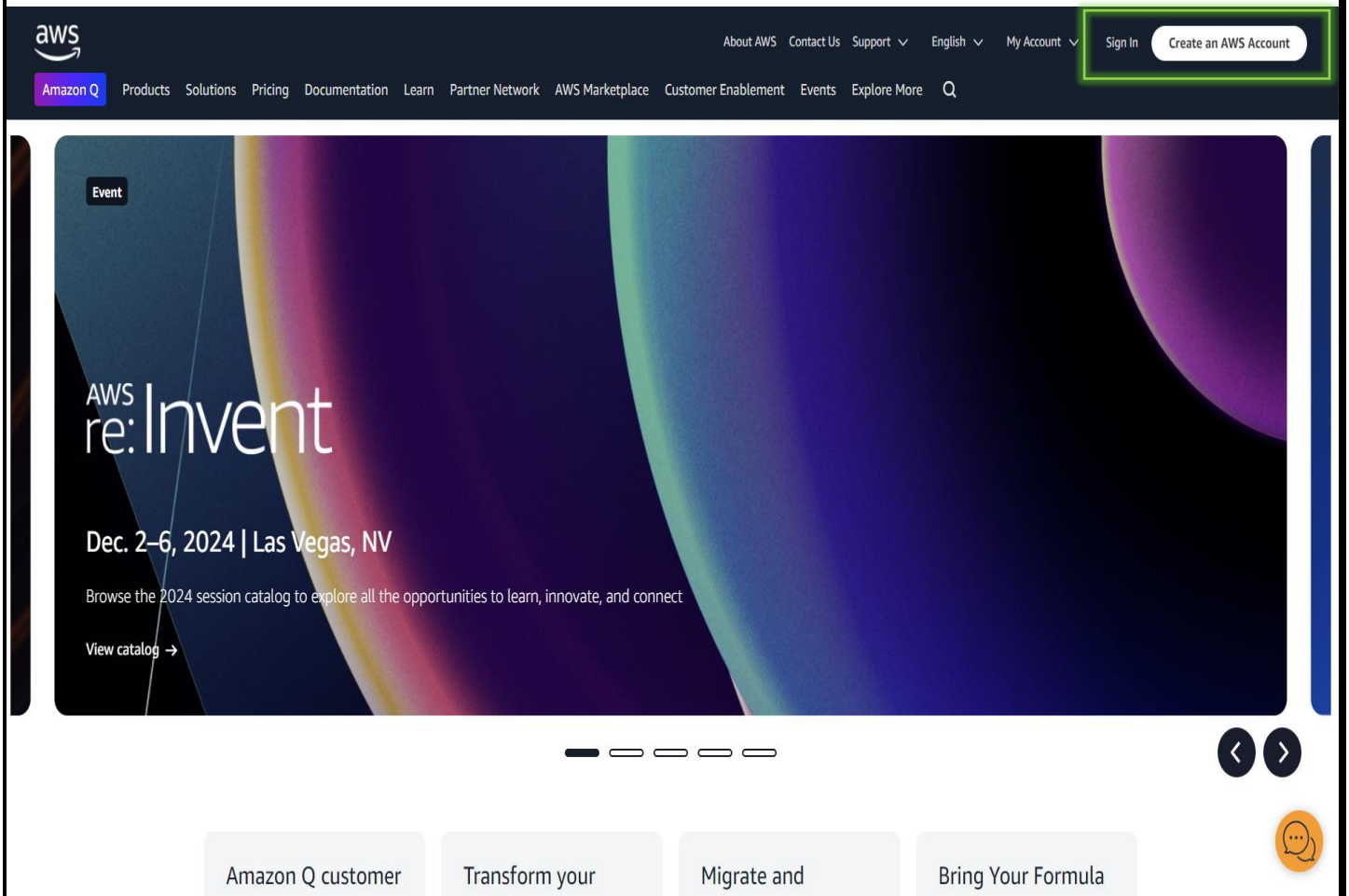
- **Amazon CloudWatch:**
  - Monitor your AWS resources and applications in real-time.
  - Set up alarms and logs to track performance and errors.

- **AWS IAM (Identity and Access Management):**
  - Manage access to your AWS resources securely.
  - Create users, groups, and roles with the necessary permissions.

Prerequisites

AWS Account Setup: -

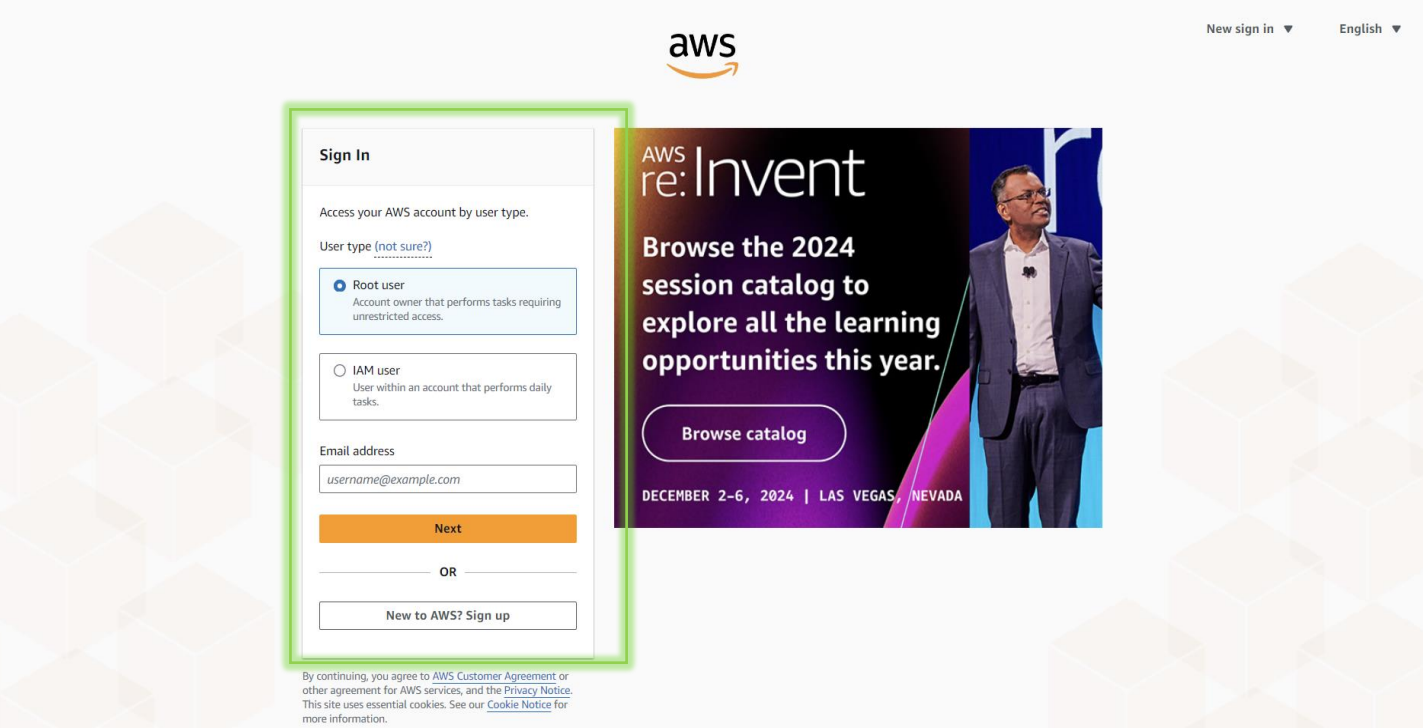
Before you can start hosting websites on AWS, you need to set up an AWS account. Visit the AWS website and follow the sign-up process. Ensure you have a valid payment method and complete the necessary identity verification steps.



Setting Up AWS EC2 Instance

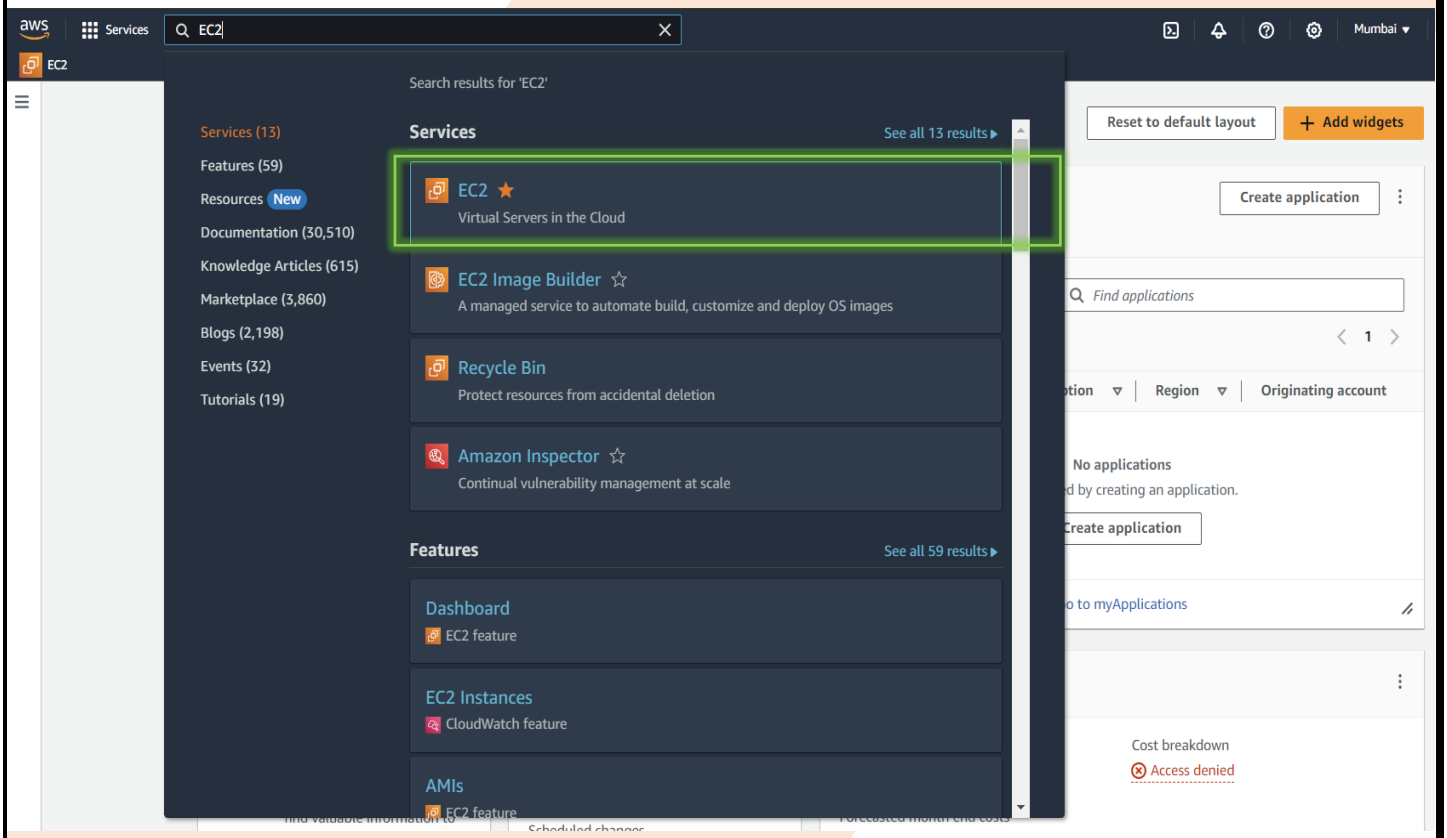
Launching an EC2 Instance:

1. Log in to the AWS Management Console.



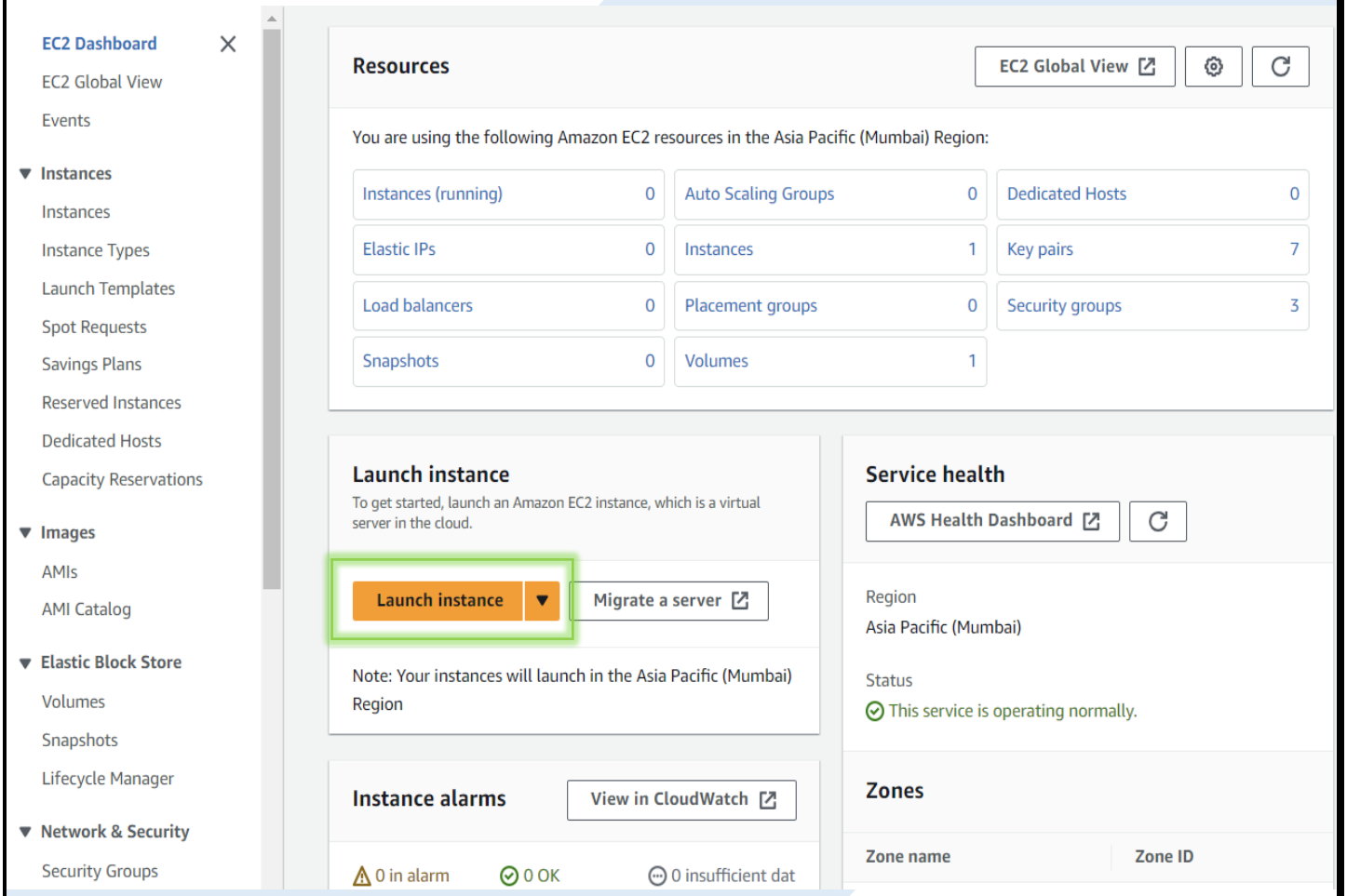
Sign in as root user or with IAM to work in a Group

2. Navigate to the EC2 Dashboard.



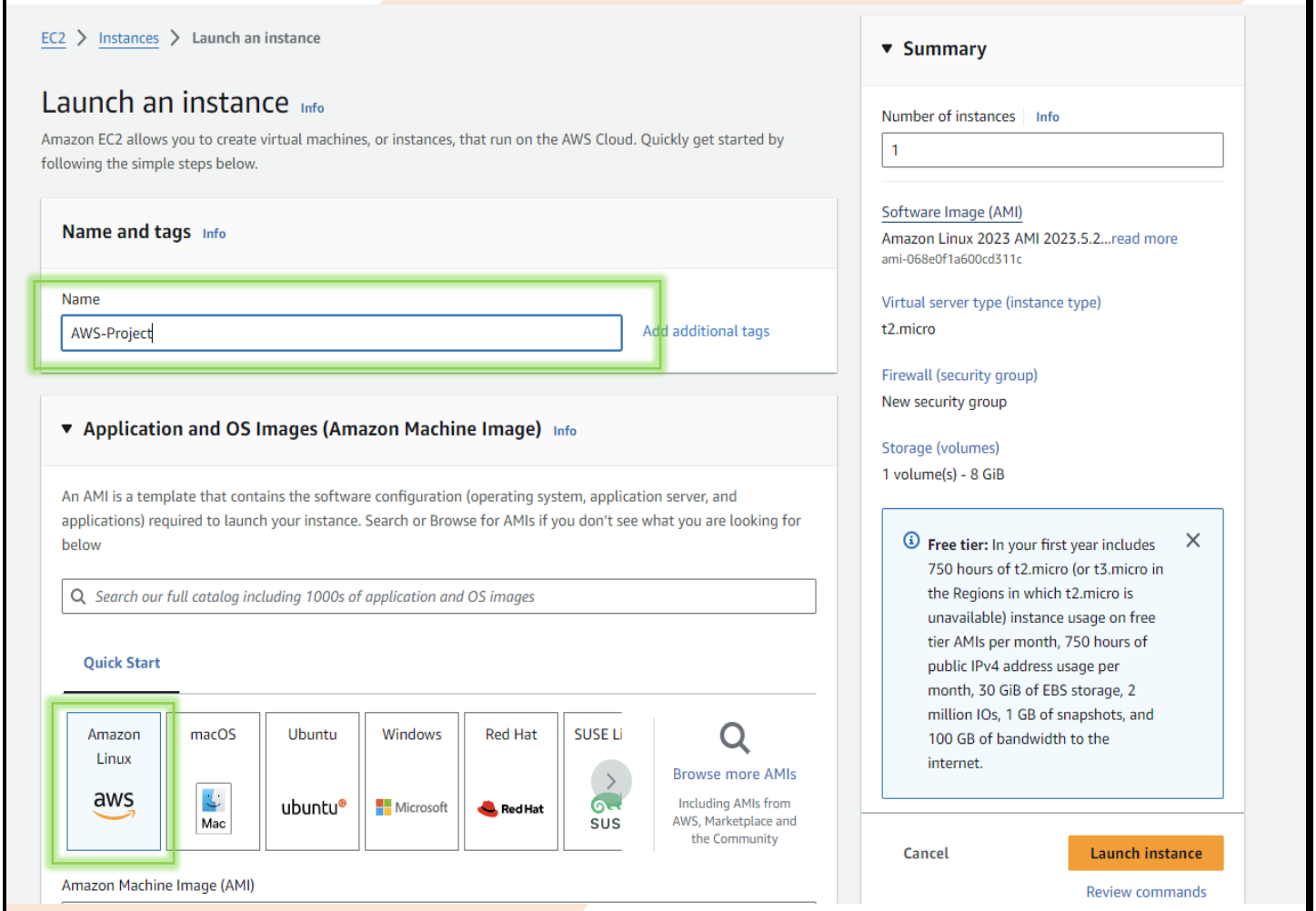
Go ahead and click the first link to open the EC2 dashboard

3. Click on "Launch Instance."



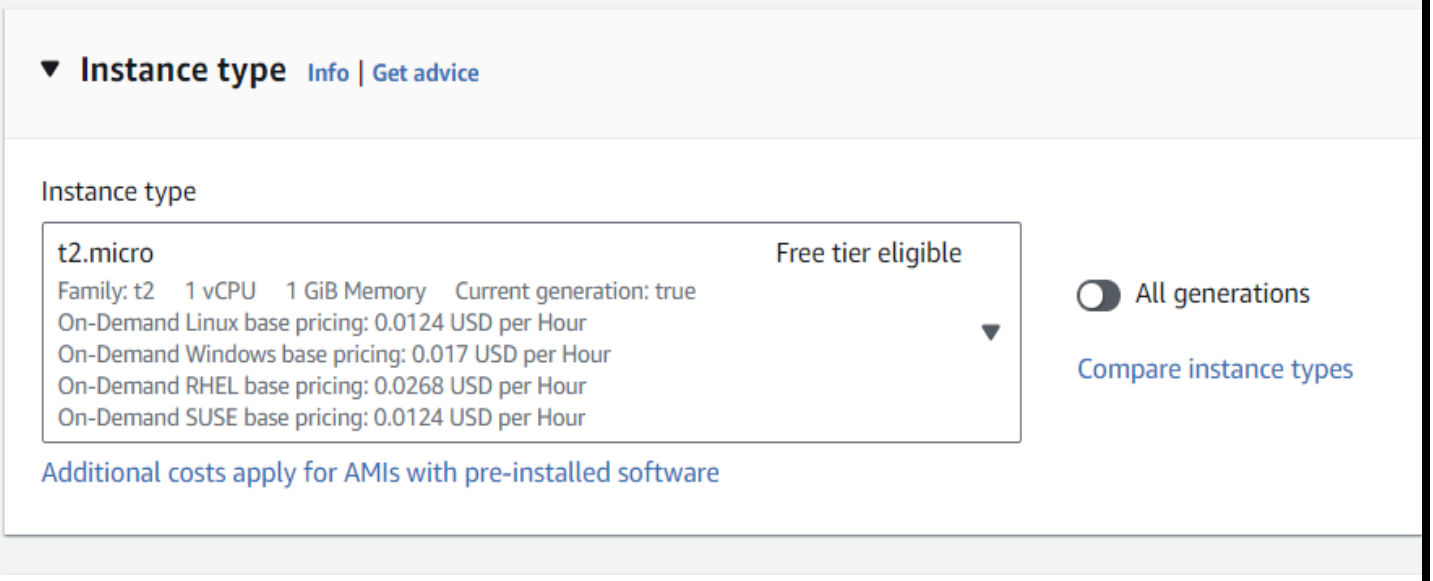
Click the Launch instance button to launch the instance

4. Choose an Amazon Machine Image (AMI). For this guide, we will use Amazon Linux.

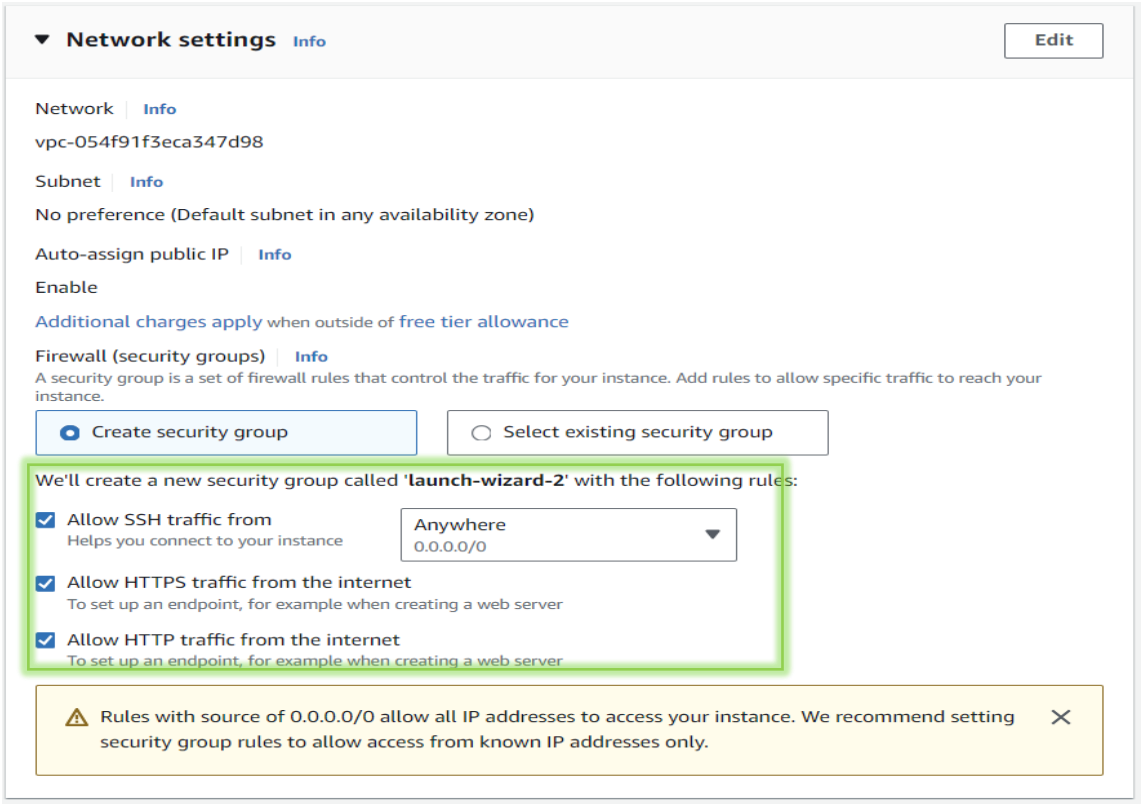


Giving the 'AWS-Project' name to our instance and chousing Amazon Linux 2023 AMI

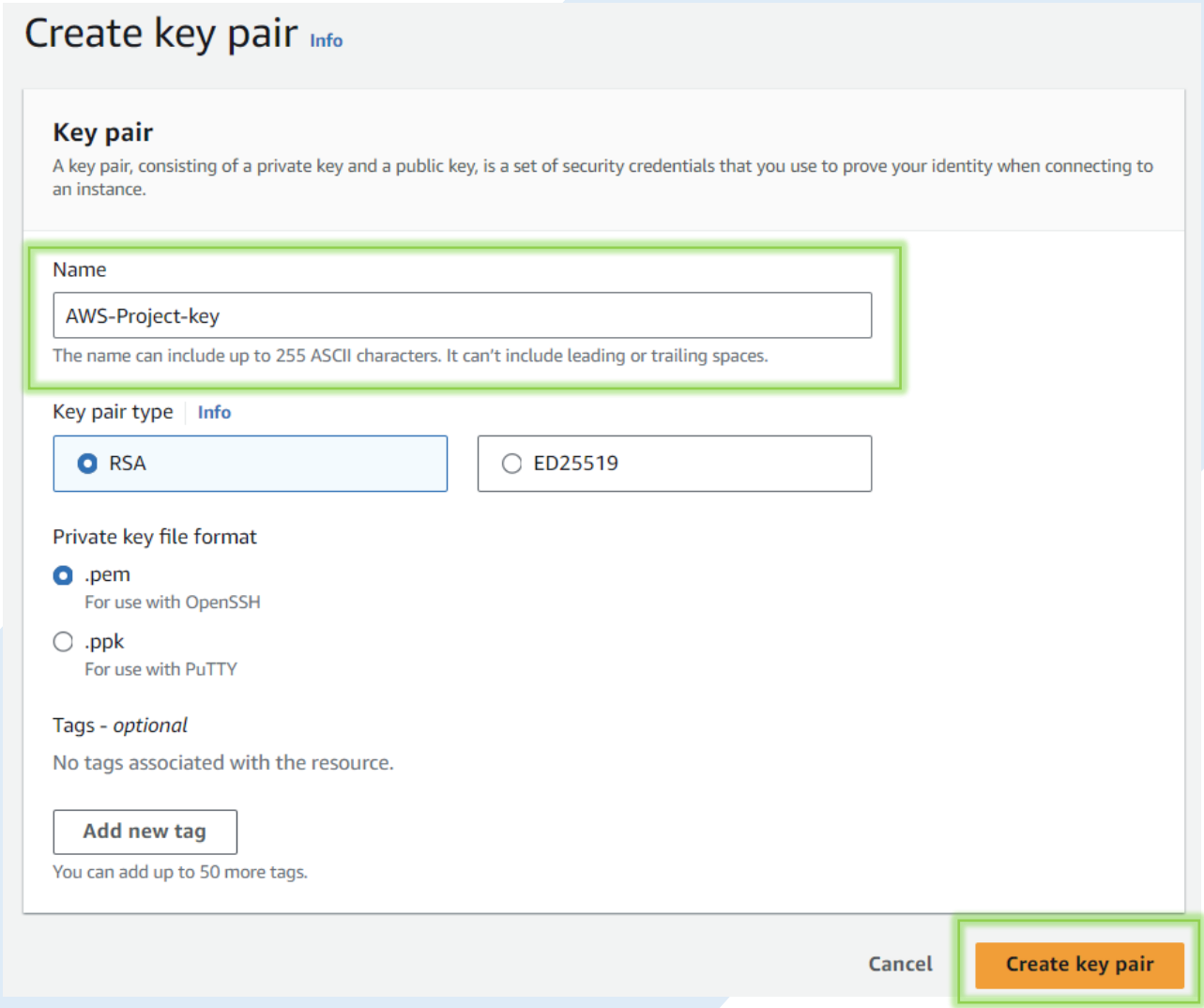
5. Select an instance type (e.g., t2. micro for the free tier).



6. Configure instance details, including network settings and storage.

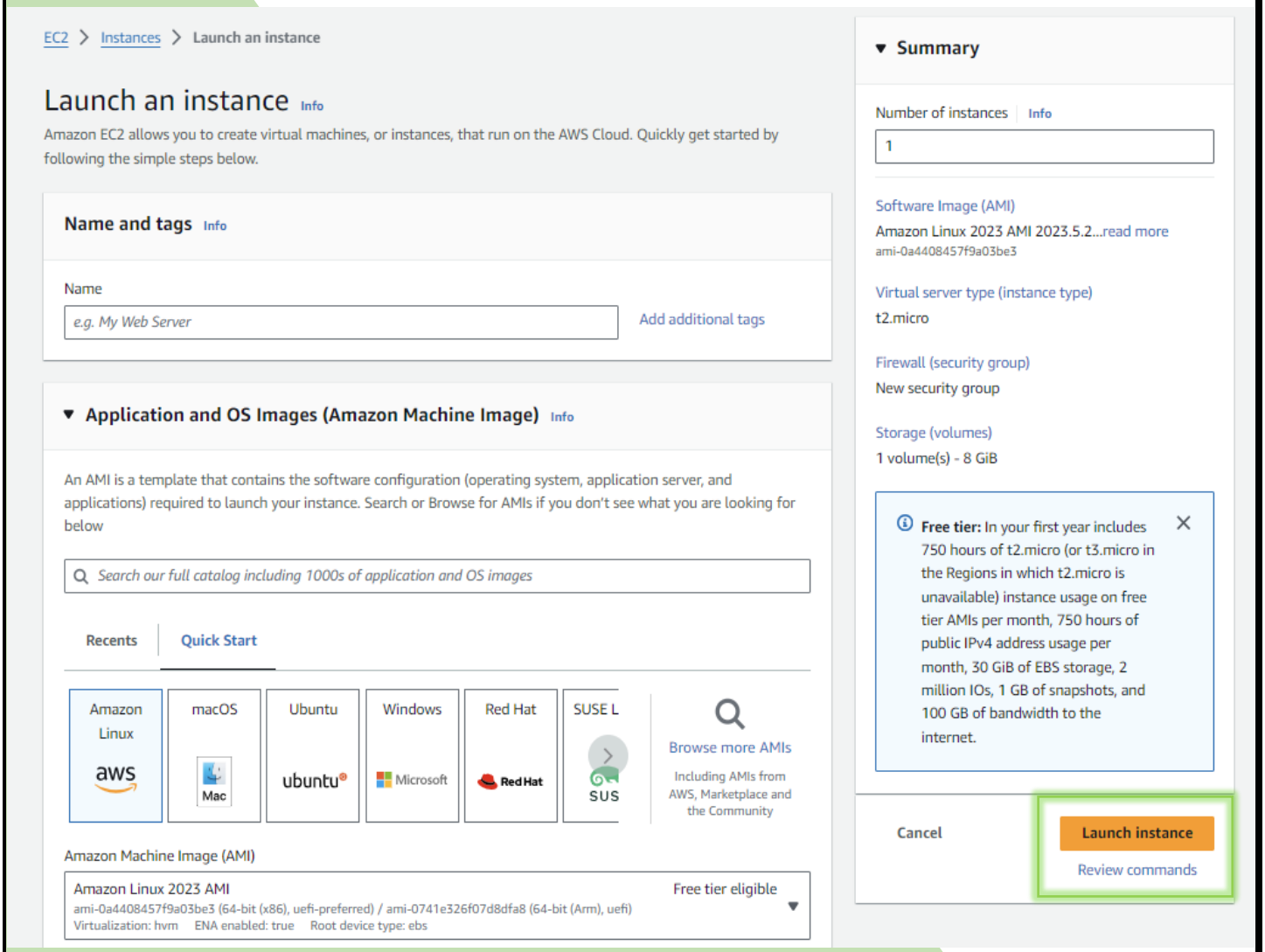


7. Add a key pair for SSH access.



Name your Key pair and chouse key format (.ppm for Linux)

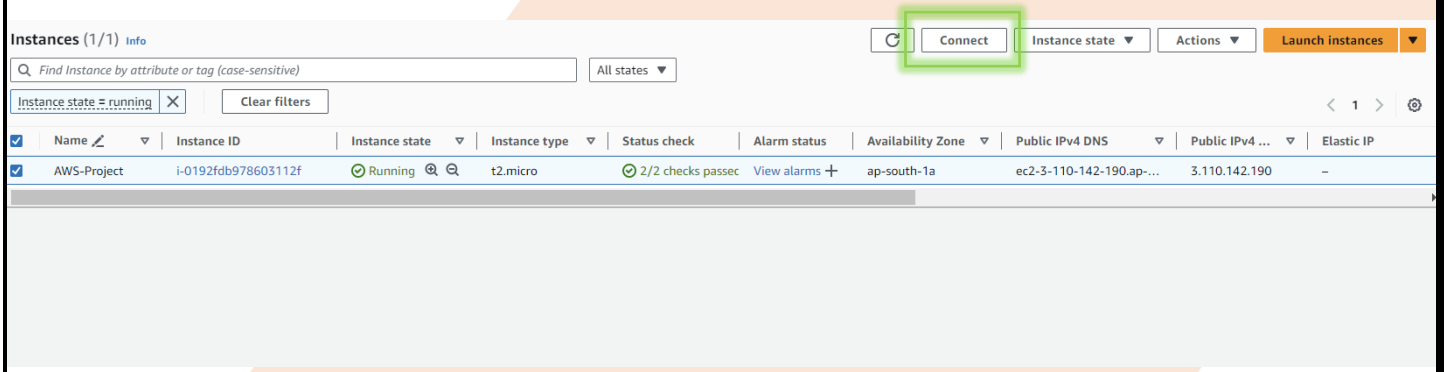
8. Review and launch the instance.

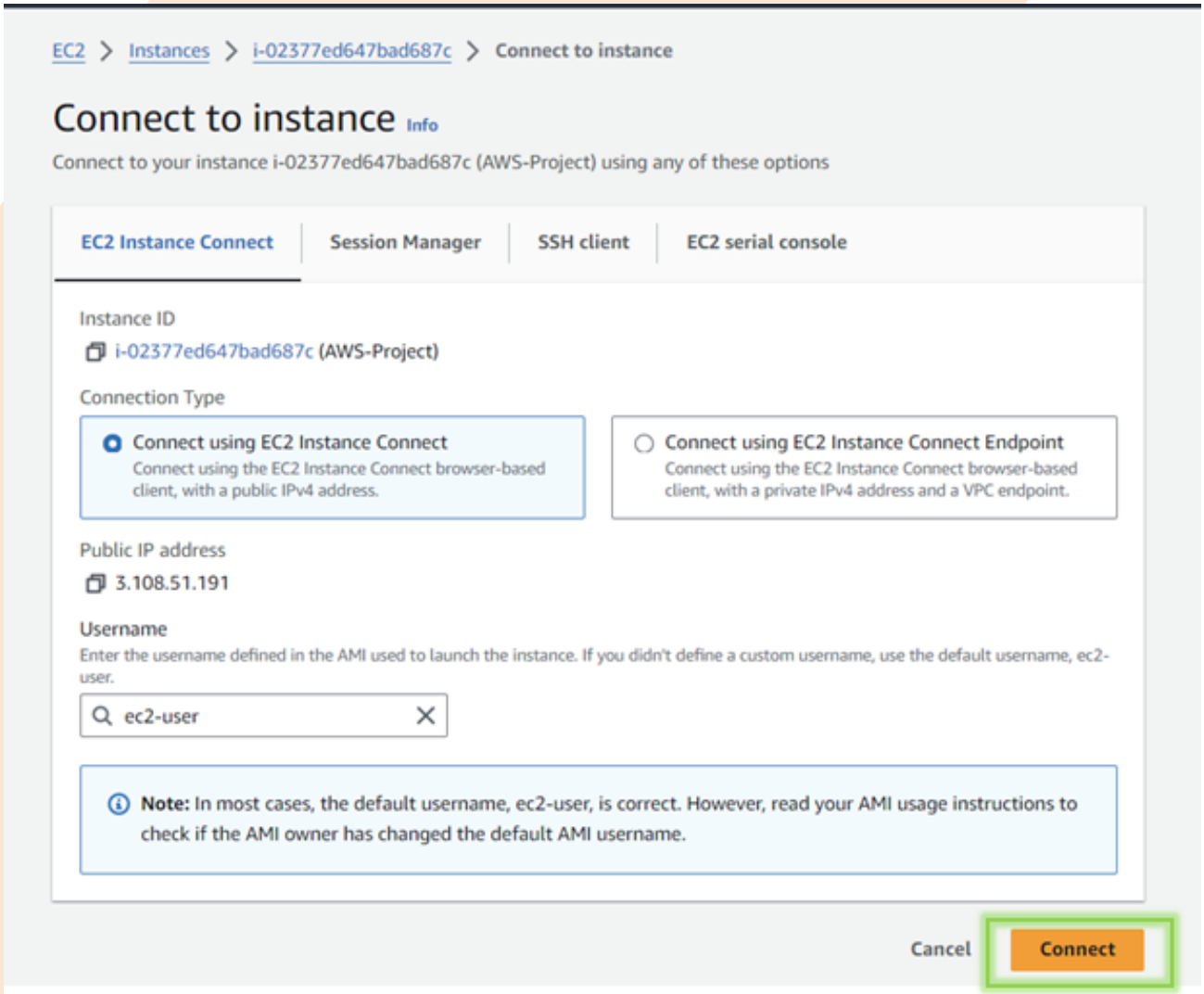


Click the Launch instance and wait until the instance is ready

Connecting To EC-2 Instance:

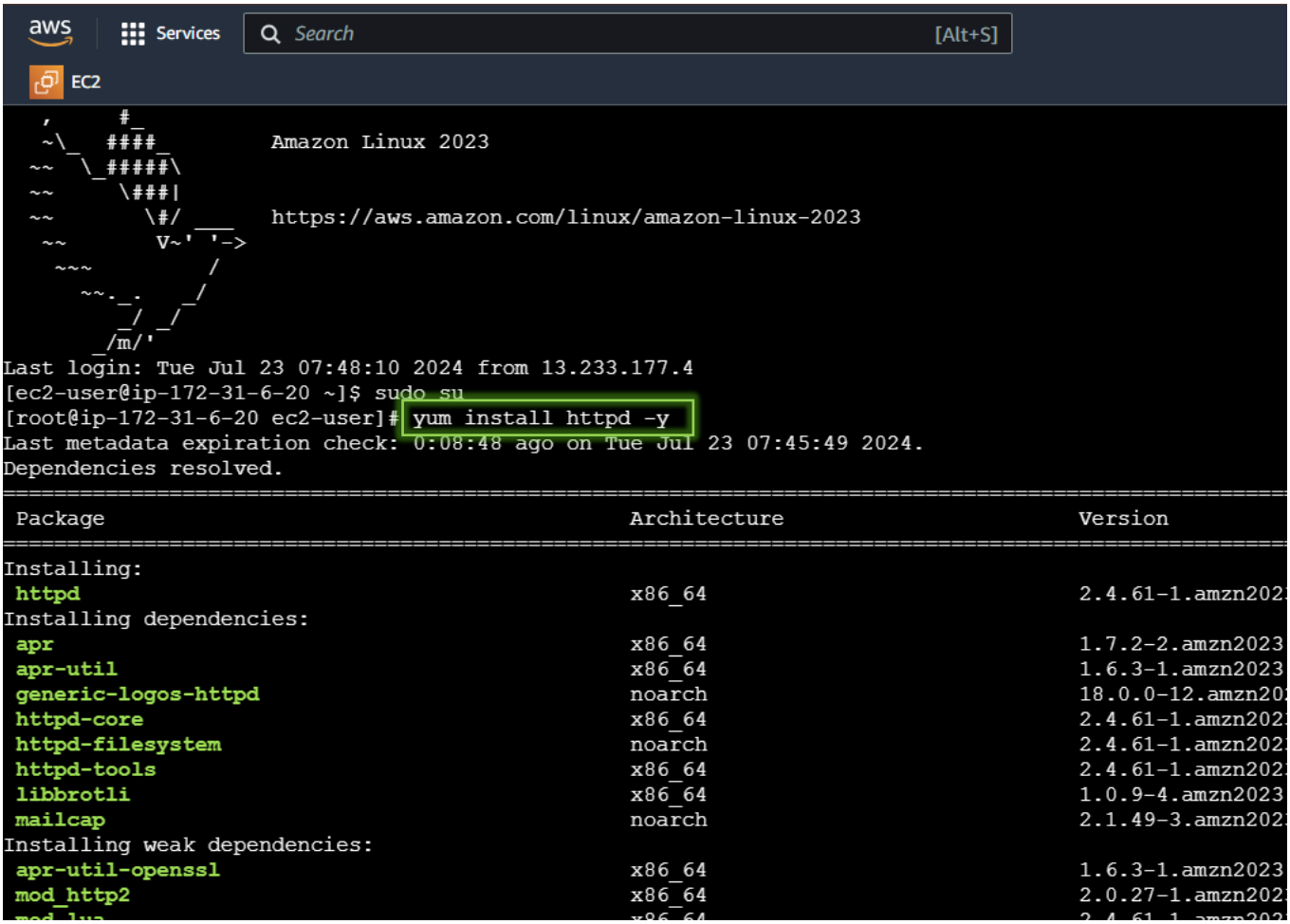
Select your instance and connect.





Configuring a Linux Server

Step 1: Install Apache



Starting Service

Step 4 : Deploy Website Content

- Making directories [inside 'var/www/html' directory]

```
[root@ip-172-31-6-20 html]# mkdir vidixer.com
```

```
[root@ip-172-31-6-20 html]# mkdir cashify.com
```

- Create Index.html as base webpage [for testing].

```
[root@ip-172-31-6-20 html]# vi index.html
```

```
<h1> hii </h1>
<hr>

  <a href="/vidixer.com"> Vidixer.com </a>
  <hr>

  <a href="/cashify.com"> Cashify.com </a>
~
~
~
~
~
~
~
```

- Creating index.html for both cashify.com and vidixer.com

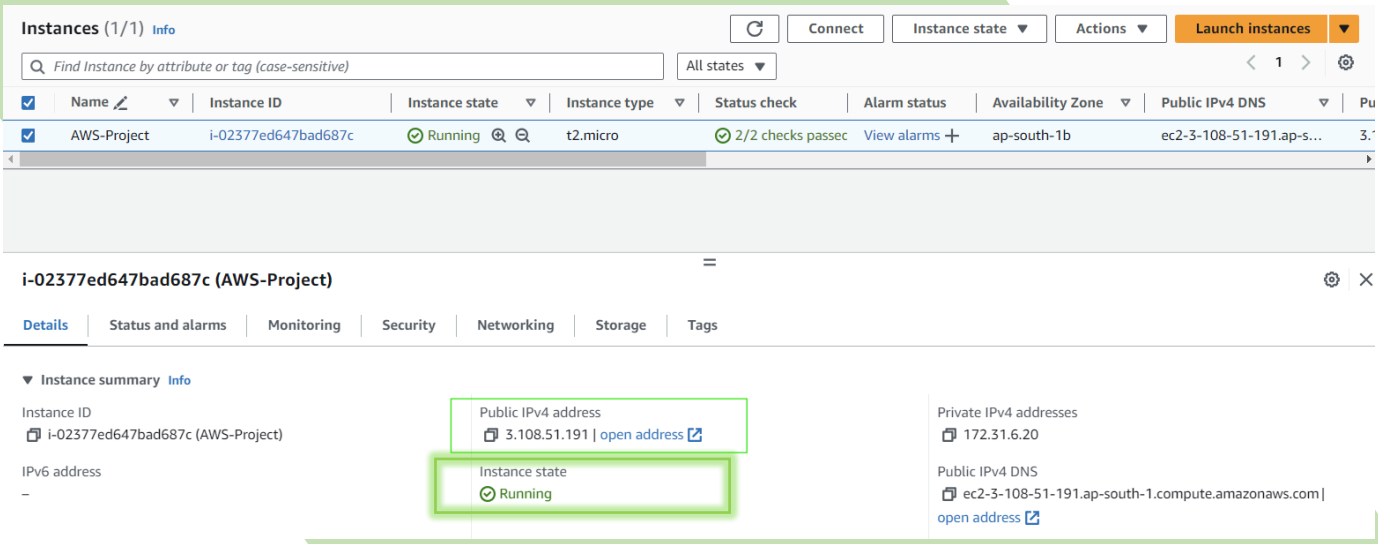
```
[root@ip-172-31-6-20 html]# ls -l
total 4
drwxr-xr-x. 3 root root  35 Jul 23 09:35 cashify.com
-rw-r--r--. 1 root root 115 Jul 23 09:47 index.html
drwxr-xr-x. 2 root root  58 Jul 23 08:34 vidixer.com
[root@ip-172-31-6-20 html]#
```

```
[root@ip-172-31-6-20 html]# cd vidixer.com
[root@ip-172-31-6-20 vidixer.com]# vi index.html
```

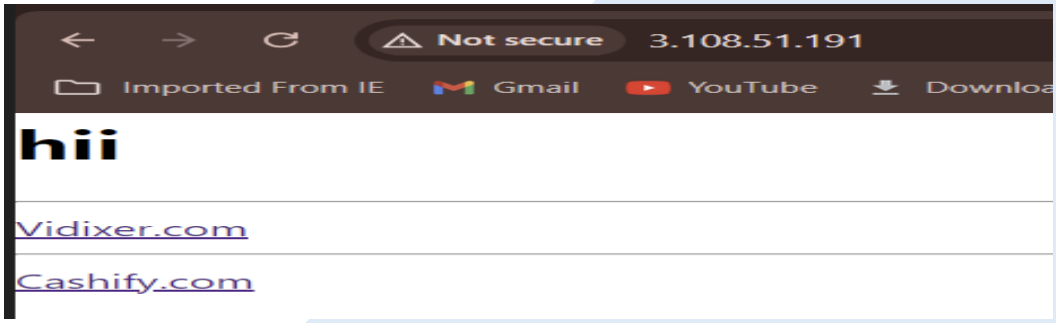
```
[root@ip-172-31-6-20 html]# cd cashify.com
[root@ip-172-31-6-20 cashify.com]# vi index.html
```

Step 5. Test your website.

- Copy instance's public IPv4 address and past in your browser.



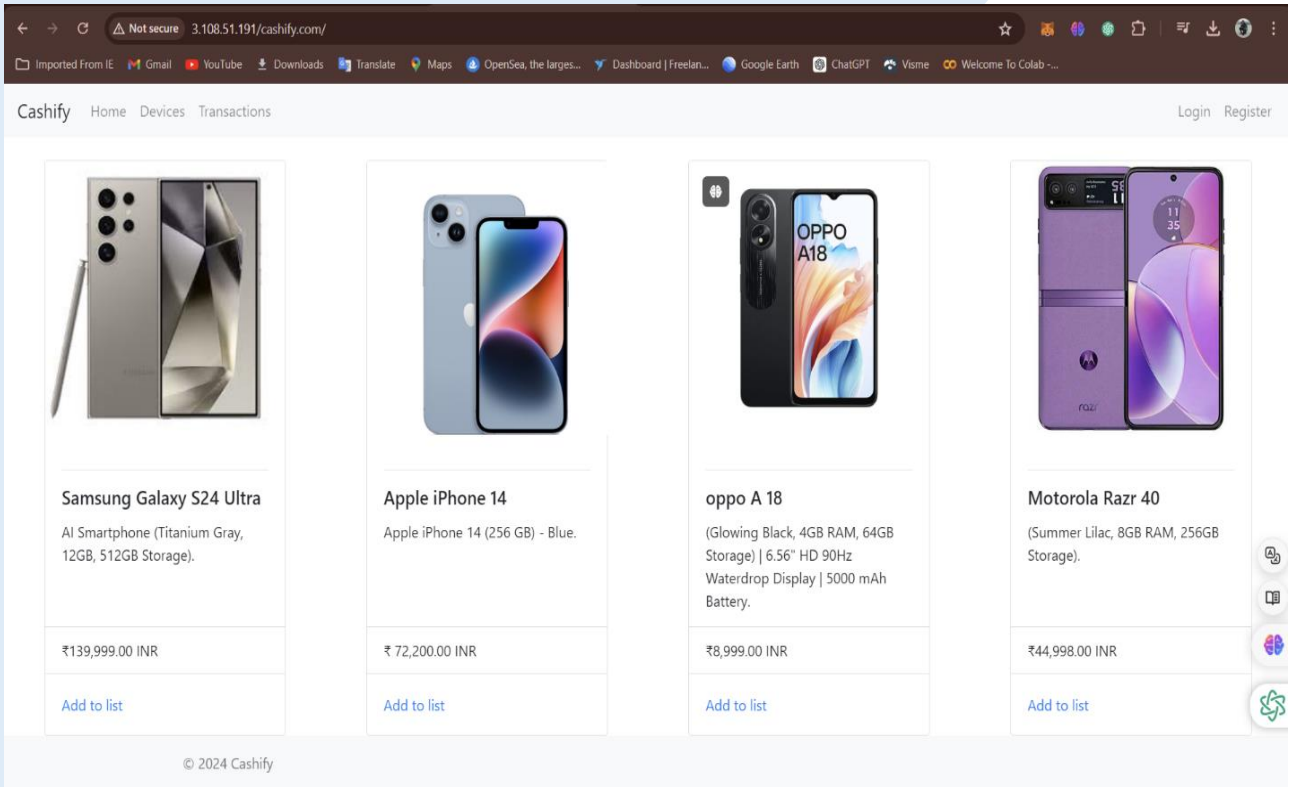
- Now you will See the base index.html [for testing]



Go ahead and visit each website

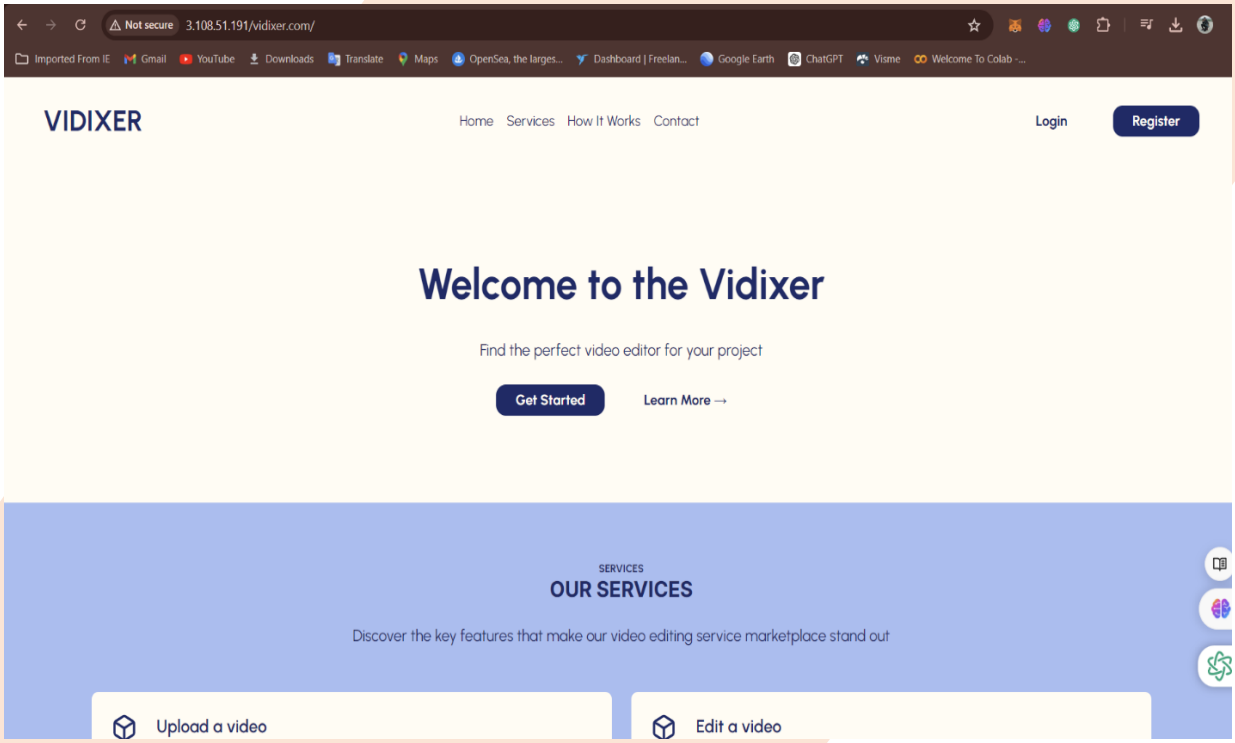
Our First Website :

cashify.com



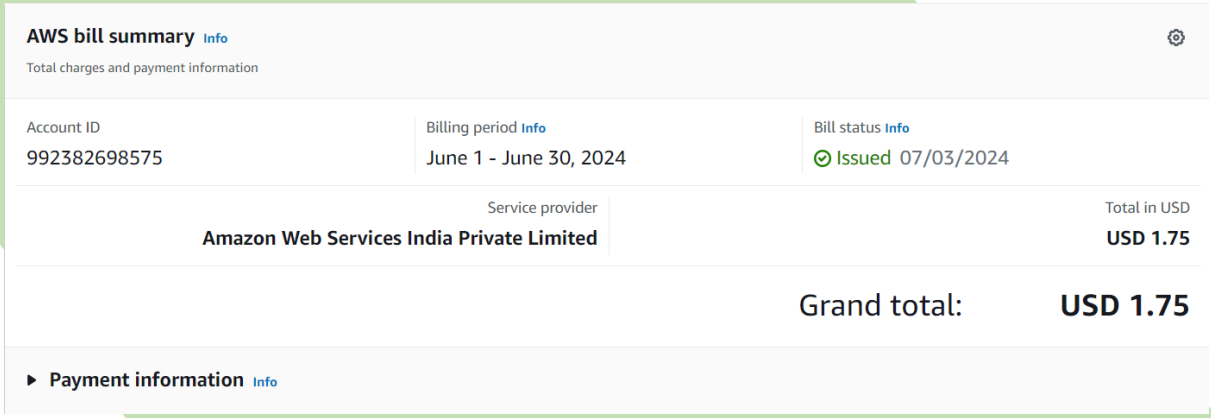
Second Website:

vidixer.com

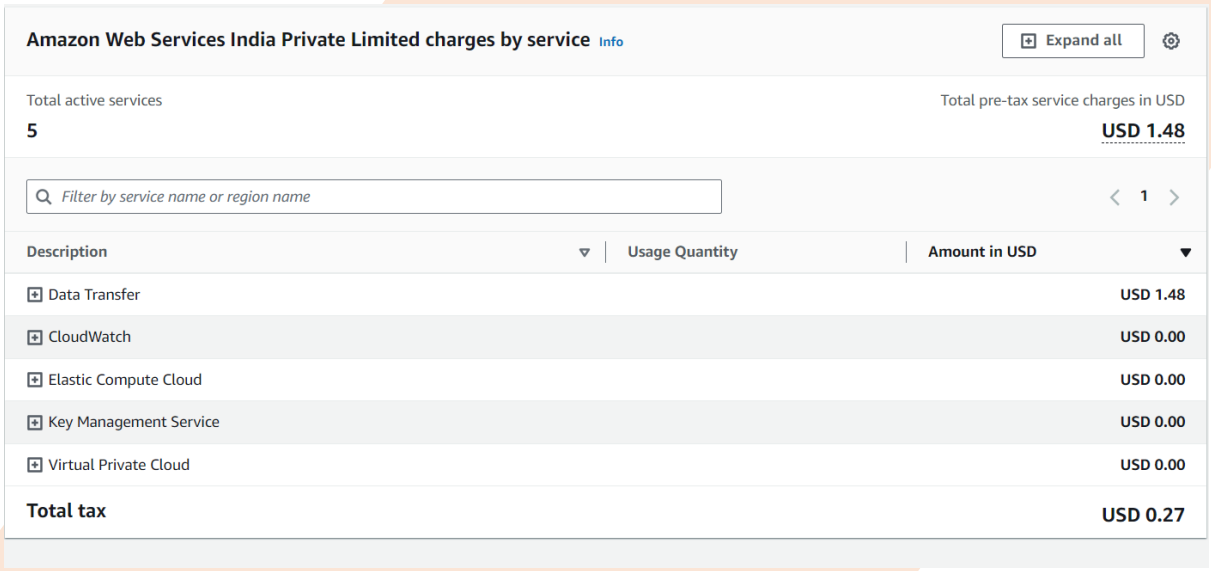


Cost Analyses

AWS Provide service as billing and cost management to see how much amount of bill is generated in the services we have used.



This is all about the cost generated in the month of June as working with AWS Account



Amazon Web Services India Private Limited charges by service Info

Expand all

Total active services
5

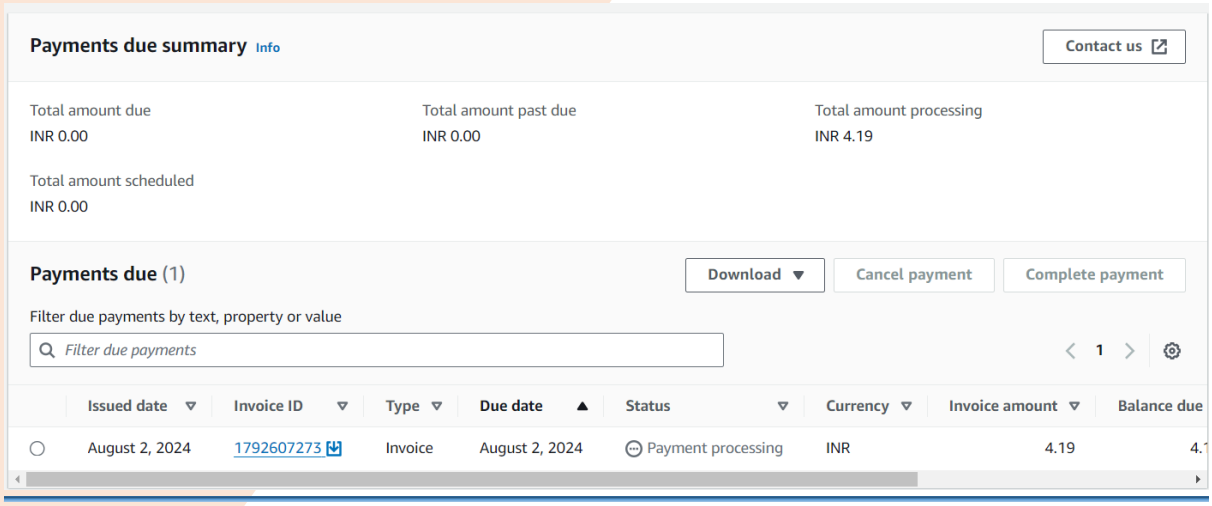
Total pre-tax service charges in USD
USD 1.48

Filter by service name or region name

1

Description	Usage Quantity	Amount in USD
Data Transfer		USD 1.48
CloudWatch		USD 0.00
Elastic Compute Cloud		USD 0.00
Key Management Service		USD 0.00
Virtual Private Cloud		USD 0.00
Total tax		USD 0.27

It not only shows cost but also shows which service has used how much amount of cost each service has costed in there use.



Payments due summary Info

Contact us

Total amount due
INR 0.00

Total amount past due
INR 0.00

Total amount processing
INR 4.19

Total amount scheduled
INR 0.00

Payments due (1)

Download | Cancel payment | Complete payment

Filter due payments by text, property or value

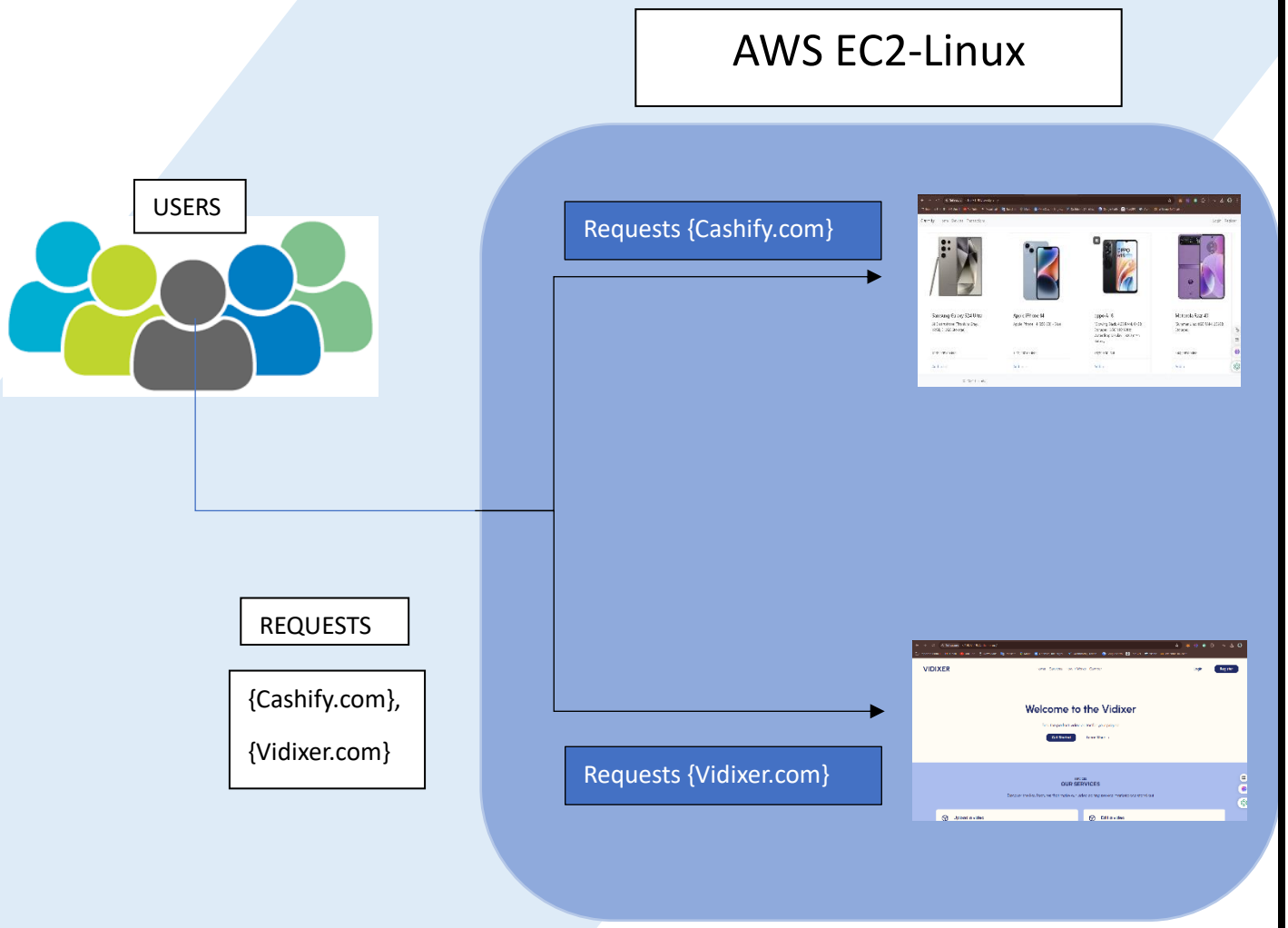
Filter due payments

1

Issued date	Invoice ID	Type	Due date	Status	Currency	Invoice amount	Balance due
August 2, 2024	1792607273	Invoice	August 2, 2024	Payment processing	INR	4.19	4.1

You get your payment information also in AWS

System Design And Architecture :



Conclusion

The successful implementation of a multi-website hosting environment on AWS EC2 has demonstrated the platform's flexibility, scalability, and cost-effectiveness. By leveraging EC2 instances, we were able to efficiently host multiple websites with varying traffic patterns, ensuring optimal performance and resource utilization.

Key achievements include:

- Effective configuration of EC2 instances for optimal resource allocation.
- Successful deployment and management of multiple websites.
- Implementation of robust security measures to protect sensitive data
- and prevent unauthorized access.
- Optimization of infrastructure costs through [Cost optimization
- strategies, e.g., reserved instances, spot instances].

Lessons learned during the project highlight the importance of:

- Careful planning and resource estimation for optimal performance.
- Regular security audits and updates to mitigate potential threats.
- Continuous monitoring and optimization of infrastructure to ensure cost-efficiency.