AWS Project

Made by: Demond Wael, Mariam Tarek, Mariam Gamal

Overview

This documentation guides the process of creating a web application environment on AWS with the following features:

1- Infrastructure as Code (IaC) using AWS CloudFormation.

2- Hosting the web app on EC2 or via microservices (like ECS, EKS, or Lambda).

3- External storage for static content using Amazon S3.

4- Monitoring and notifications for metrics exceeding specific limits.

5- Automatic remediation in case of web service failure.

6- The environment must be secure, highly available, scalable, and have disaster recovery capabilities.

7- HTTP to HTTPS redirection via Load Balancer.

8- Mounting S3 on EC2.

9- Using API Gateway to fetch an image from S3.

Procedure

1. Create Network Environment with Infrastructure as a Code.

We used AWS CloudFormation to define our infrastructure as code.

- we created a VPC with 2 public and 2 private subnets across multiple availability zones for high availability.

- Set up Internet Gateway (IGW) and NAT Gateways for internet connectivity.

- For IAM Roles and Policies, we used LabRole for access control.

So we made a YAML template to be used on CloudFormation which is as follows:

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport:  true
      EnableDnsHostnames:  true
      Tags:
        - Key: Name
          Value: VPC1

  IGW:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: VPC1 IG

  VPCtoIGWConnection:
    Type: AWS::EC2::VPCGatewayAttachment
    DependsOn:
      - IGW
      - VPC
    Properties:
      InternetGatewayId:  !Ref IGW
      VpcId: !Ref VPC

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    DependsOn: VPC
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
```

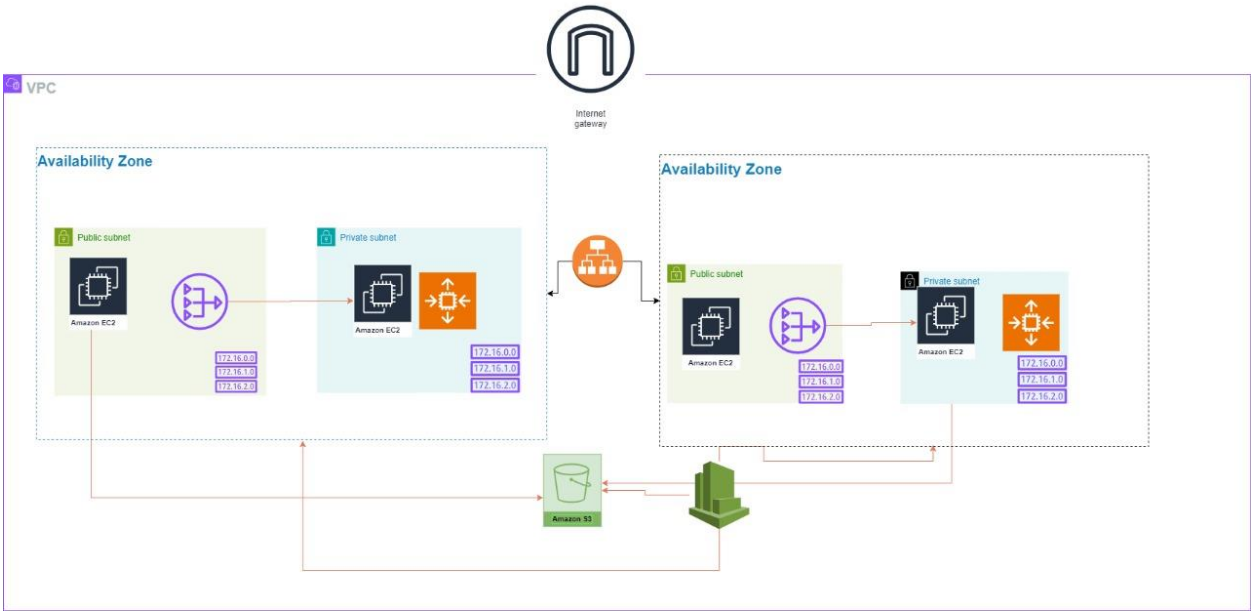
```
      Value: Public Route Table

PublicRoute:
  Type: AWS::EC2::Route
  DependsOn:
    - PublicRouteTable
    - VPCtoIGWConnection
  Properties:
    DestinationCidrBlock:  0.0.0.0/0
    GatewayId: !Ref IGW
    RouteTableId: !Ref PublicRouteTable

PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  DependsOn: VPC
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: Private Route Table 1

PublicSubnet1:
  Type: AWS::EC2::Subnet
  DependsOn: VPC
  Properties:
    VpcId: !Ref VPC
    MapPublicIpOnLaunch:  true
    CidrBlock: 10.0.0.0/24
    AvailabilityZone: !Select
      - 0
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
```

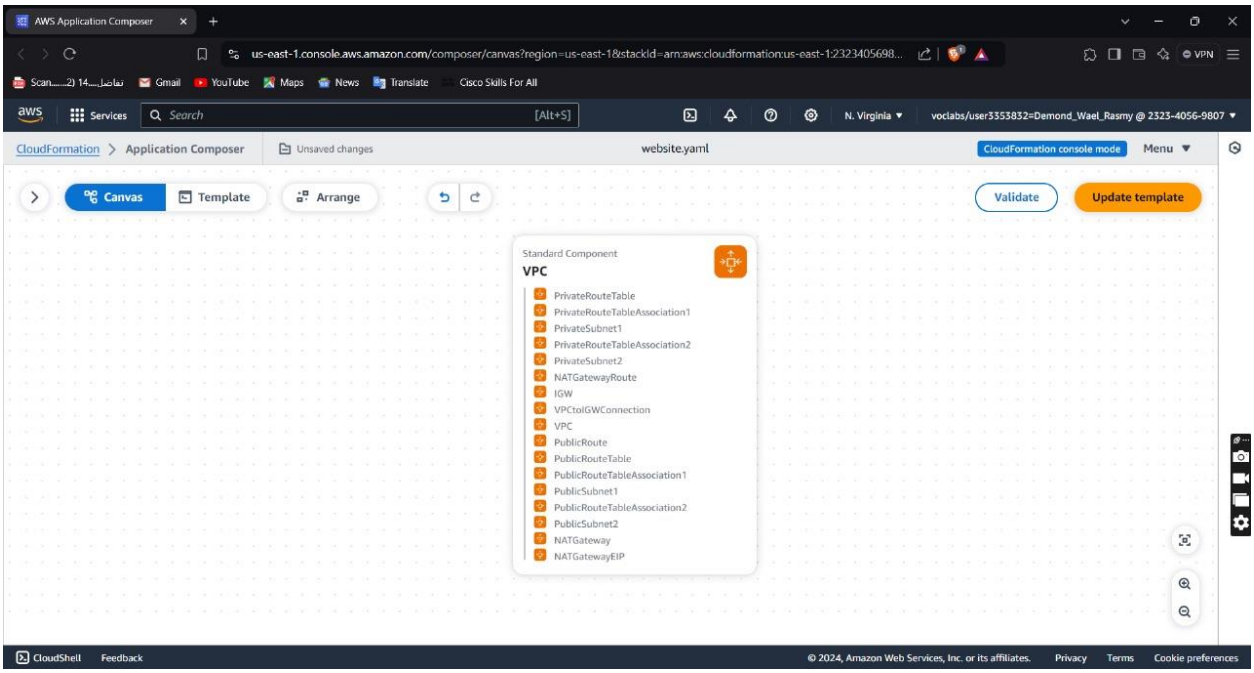
```
      Value: Public Subnet 1

PublicSubnet2:
  Type: AWS::EC2::Subnet
  DependsOn: VPC
  Properties:
    VpcId: !Ref VPC
    MapPublicIpOnLaunch:  true
    CidrBlock: 10.0.1.0/24
    AvailabilityZone: !Select
      - 1
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
        Value: Public Subnet 2

PublicRouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  DependsOn:
    - PublicRouteTable
    - PublicSubnet1
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicRouteTableAssociation2:
  Type: AWS::EC2::SubnetRouteTableAssociation
  DependsOn:
    - PublicRouteTable
    - PublicSubnet2
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2
```

```
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  DependsOn: VPC
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.2.0/23
    AvailabilityZone: !Select
      - 0
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
        Value: Private Subnet 1

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  DependsOn: VPC
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.4.0/23
    AvailabilityZone: !Select
      - 1
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
        Value: Private Subnet 2

PrivateRouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  DependsOn:
    - PrivateRouteTable
    - PrivateSubnet1
  Properties:
```

```
    RouteTableId: !Ref PrivateRouteTable
    SubnetId: !Ref PrivateSubnet1

  PrivateRouteTableAssociation2:
    Type: AWS::EC2::SubnetRouteTableAssociation
    DependsOn:
      - PrivateRouteTable
      - PrivateSubnet2
    Properties:
      RouteTableId: !Ref PrivateRouteTable
      SubnetId: !Ref PrivateSubnet2

  NATGateway:
    DependsOn: PrivateSubnet1
    Type: AWS::EC2::NatGateway
    Properties:
      SubnetId: !Ref PublicSubnet1
      AllocationId: !GetAtt
        - NATGatewayEIP
        - AllocationId

  NATGatewayEIP:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc

  NATGatewayRoute:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NATGateway
```

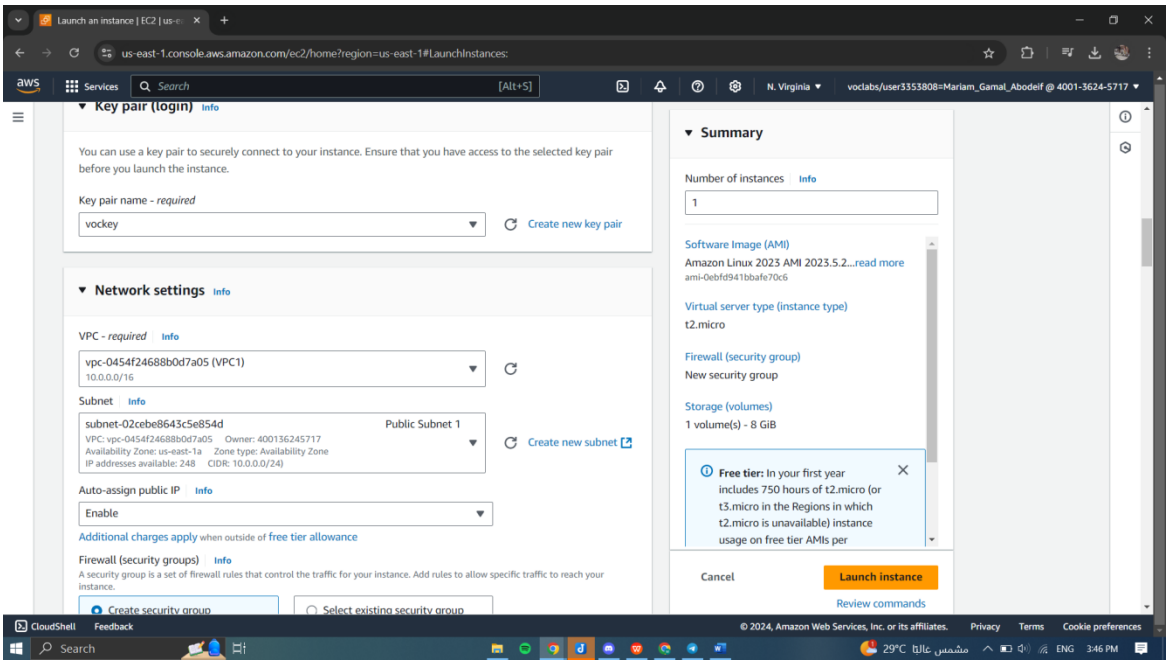
So our Environment Architecture will be like this:



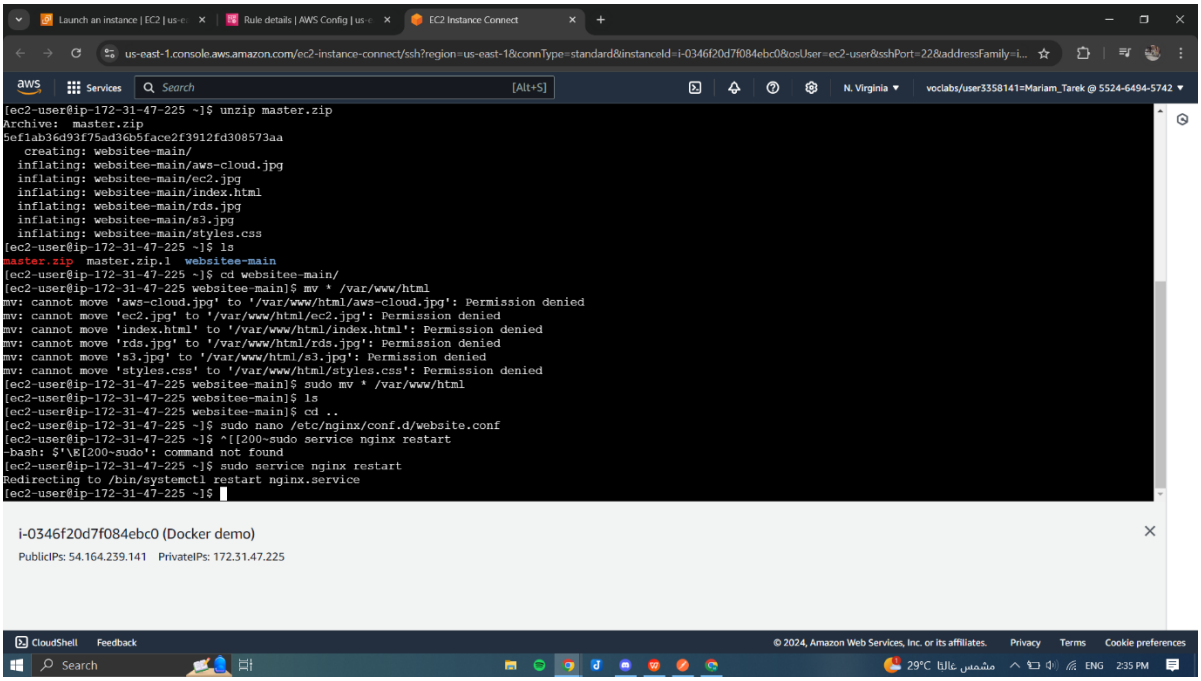


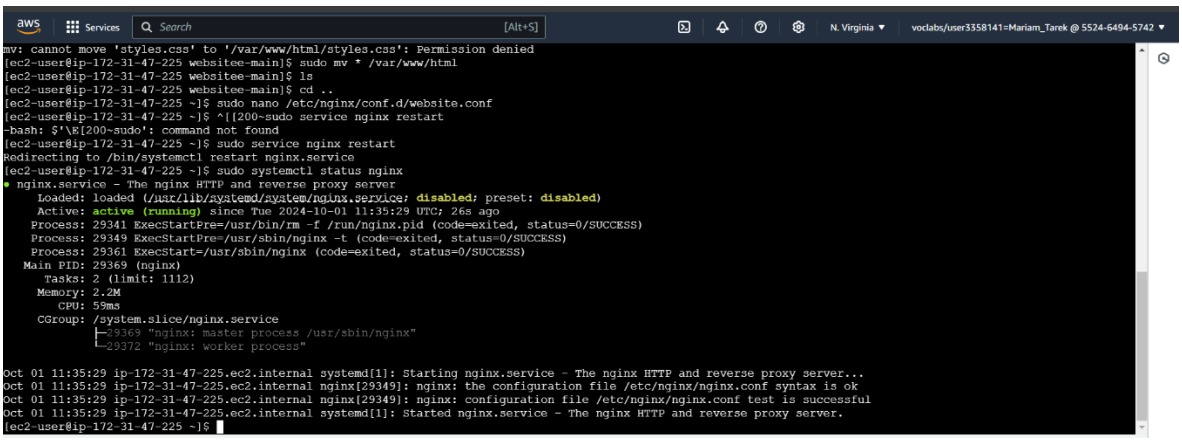
2. Web App Hosting on EC2

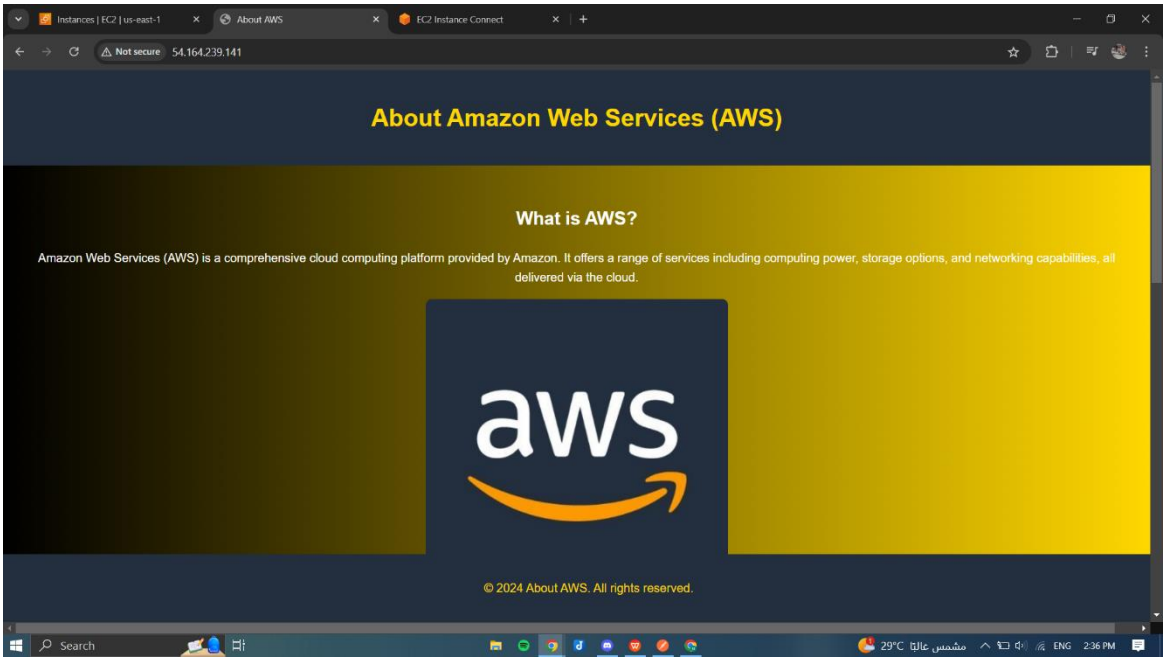
- Firstly, we have launched an EC2 instance to host the web server.



- Then, we connected to it with EC2 instance connect, and hosted the web app through nginx-service.



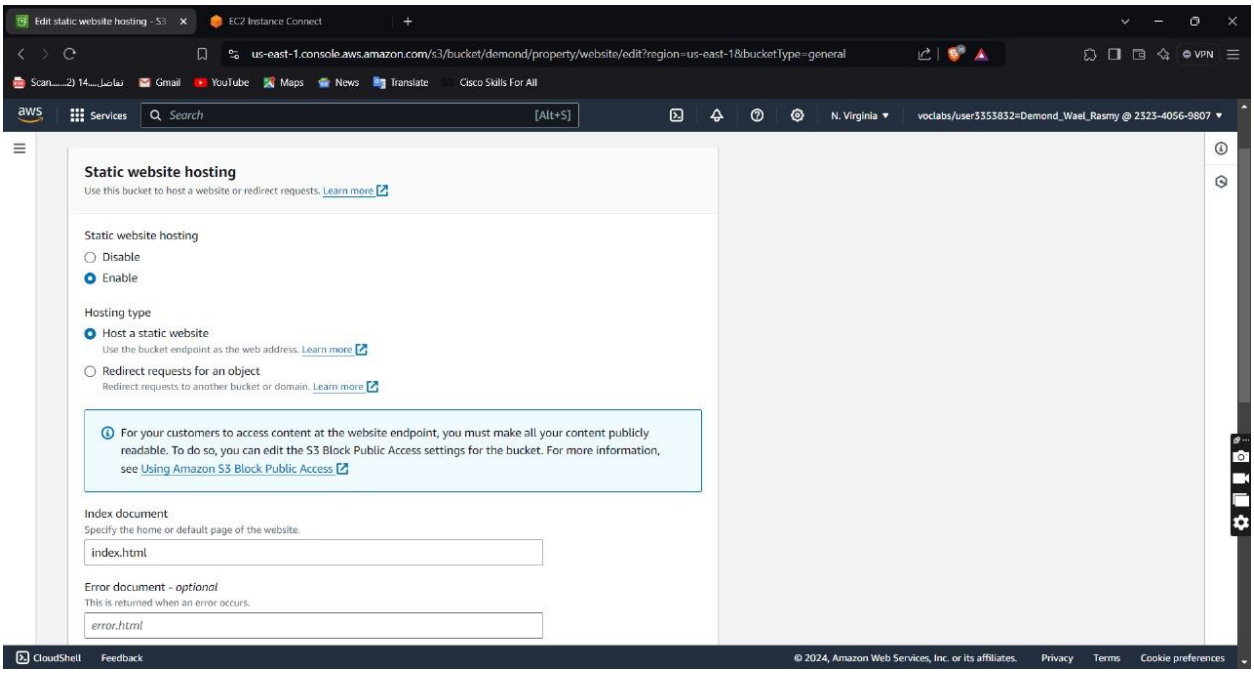


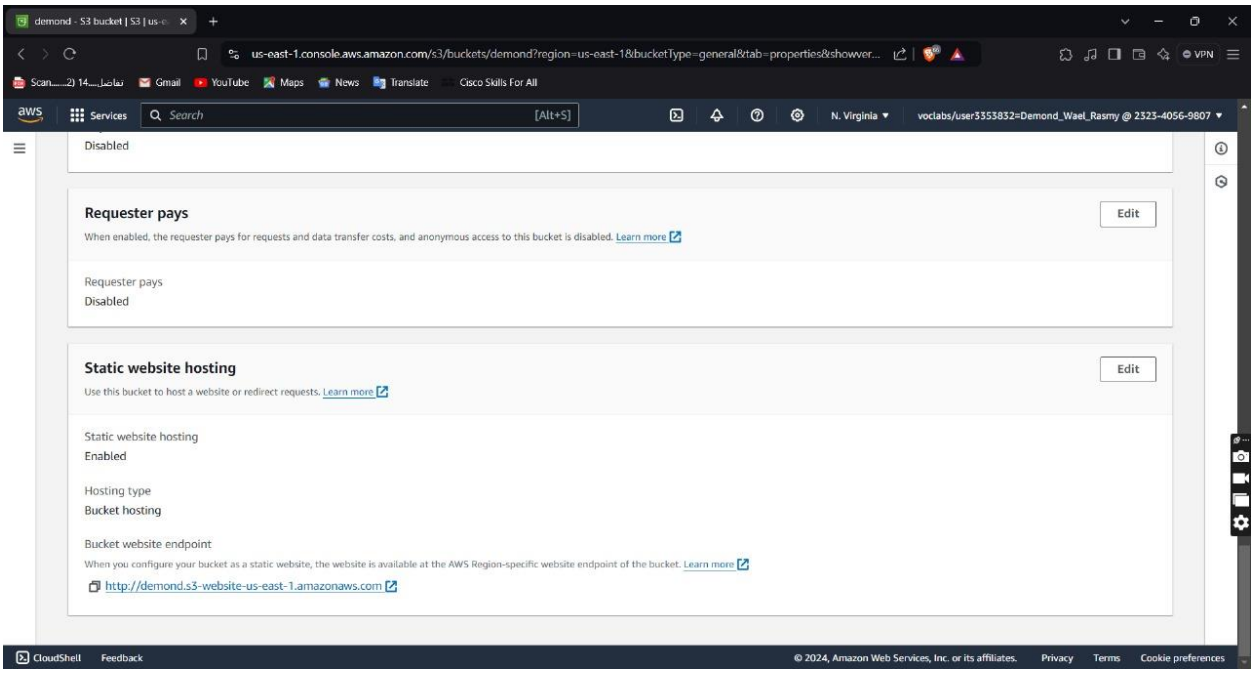


Finally, we managed to host the web app on EC2.

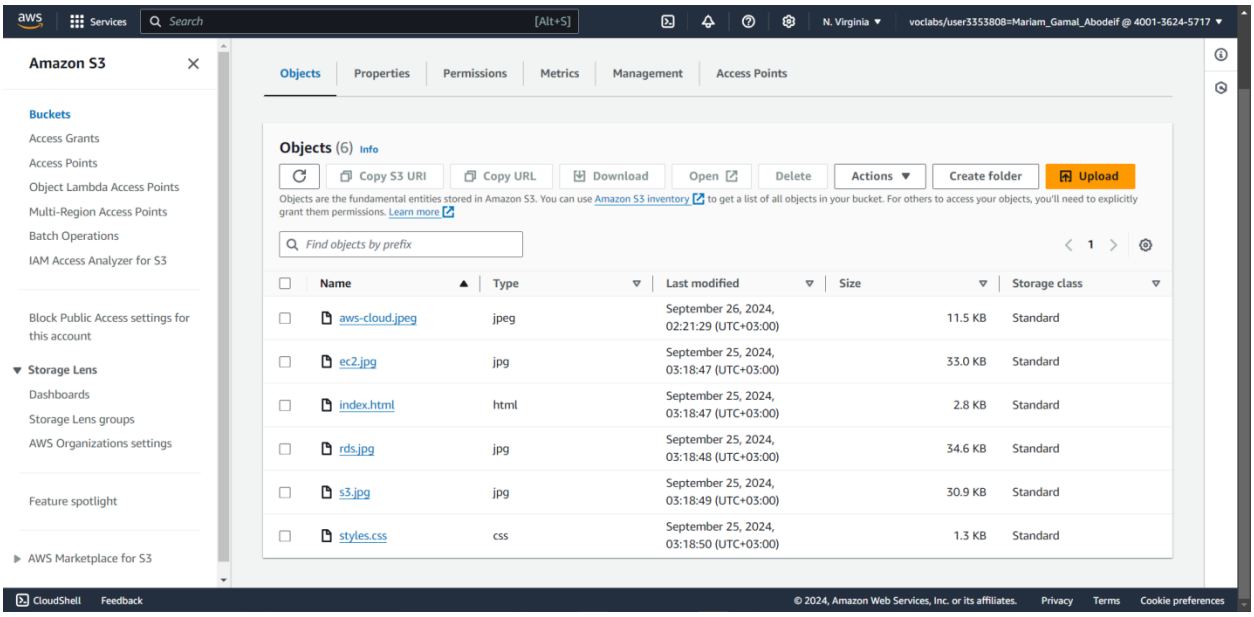
3. External Storage for Static Content through S3 bucket

- We created an S3 bucket to store the website content on it





- Then, we uploaded the content of the website on it



- Finally, we made a bucket policy to be public for all.

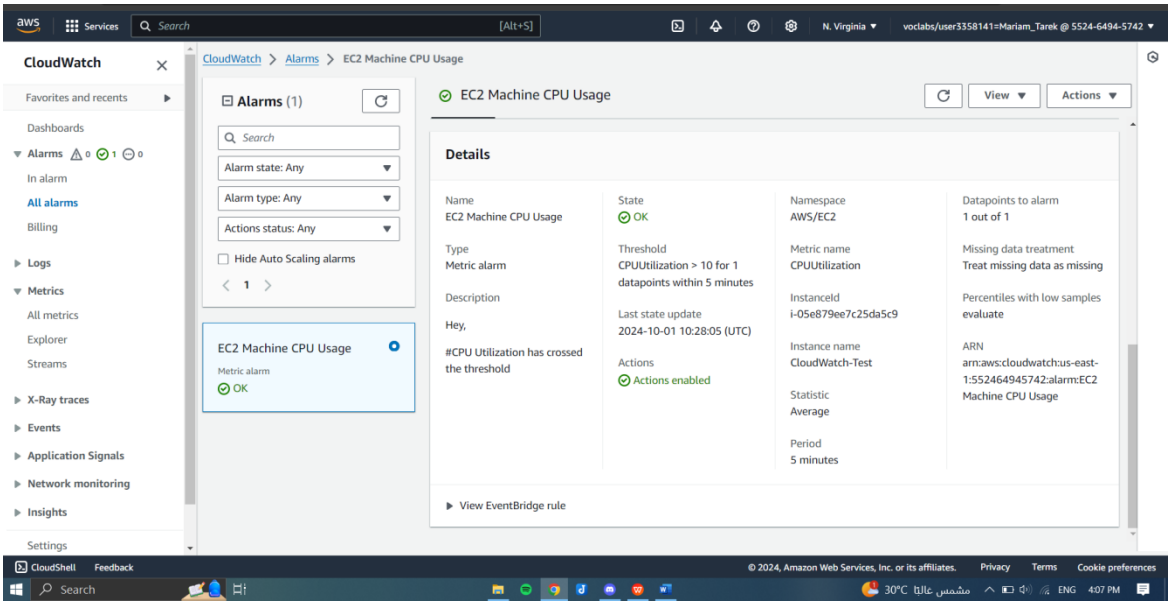
```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::EC2-INSTANCE-ROLE-ARN"
   },
   "Action": [
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": "arn:aws:s3:::your-bucket-name"
  },
  {
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::EC2-INSTANCE-ROLE-ARN"
   },
```

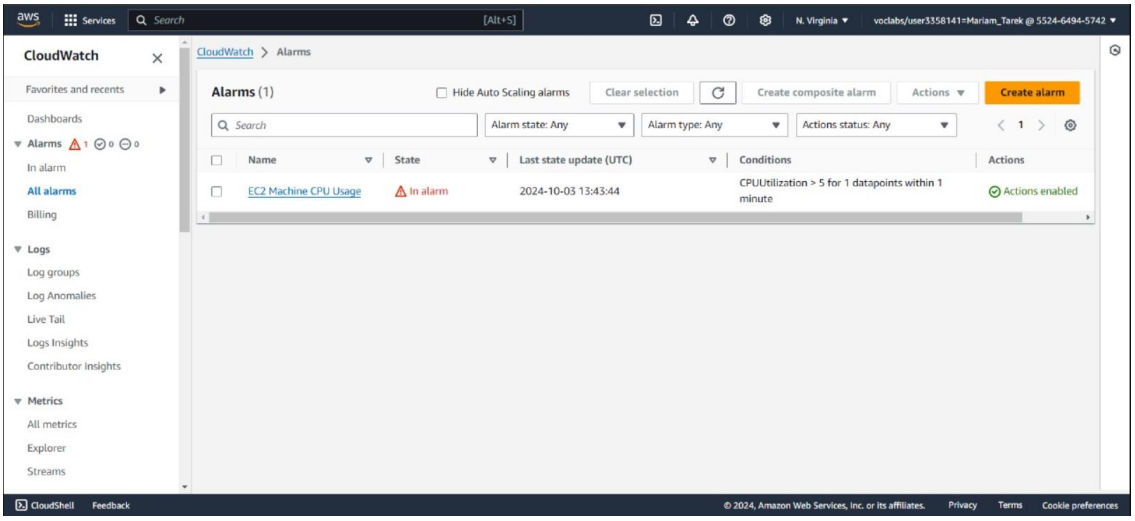
```
    "Action": [
      "s3:GetObject",
      "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::your-bucket-name/*"
  }
 ]
}
```

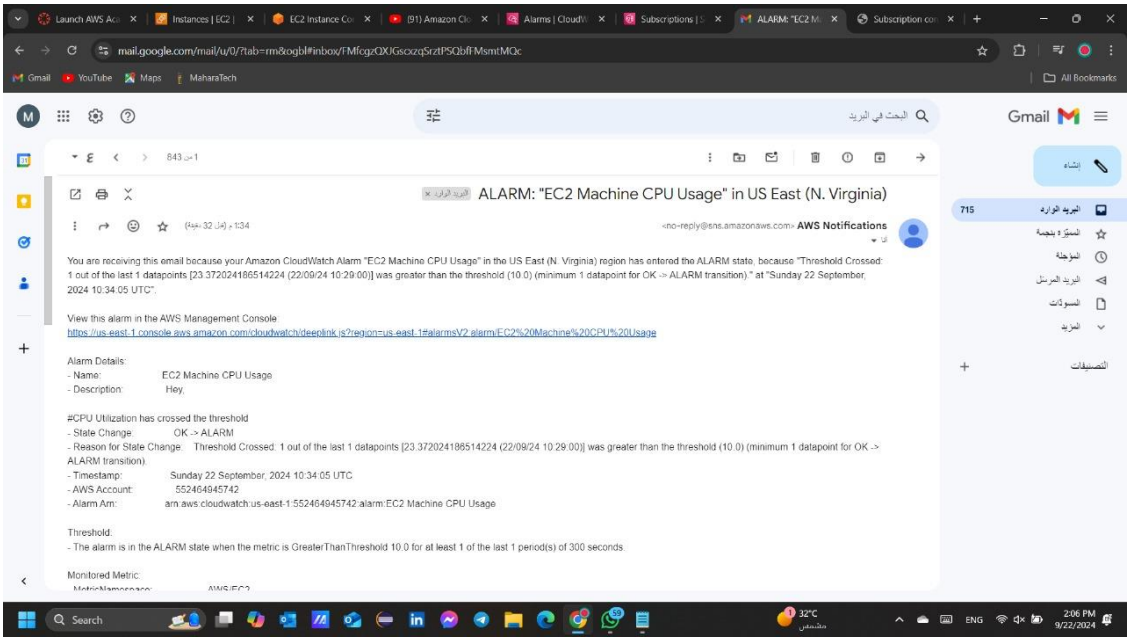
4. Monitoring your servers with integrated notifications while metrics exceed specific limit.

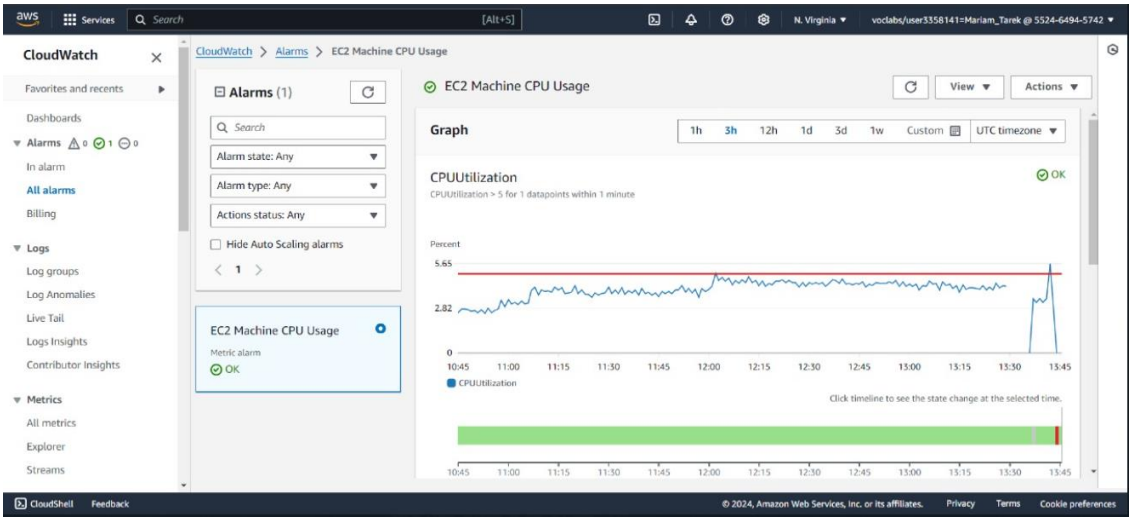
- We set up Amazon CloudWatch to monitor CPU usage for our EC2 instances.

- Then, we set CloudWatch Alarms to notify via SNS (Simple Notification Service) if a metric exceeds a defined threshold (CPUUtilization > 10 for 1 datapoints within 5 minutes).




CloudWatch
Alarms (1)
EC2 Machine CPU Usage
In alarm
2024-10-03 13:43:44
CPUUtilization > 5 for 1 datapoints within 1 minute
Actions enabled

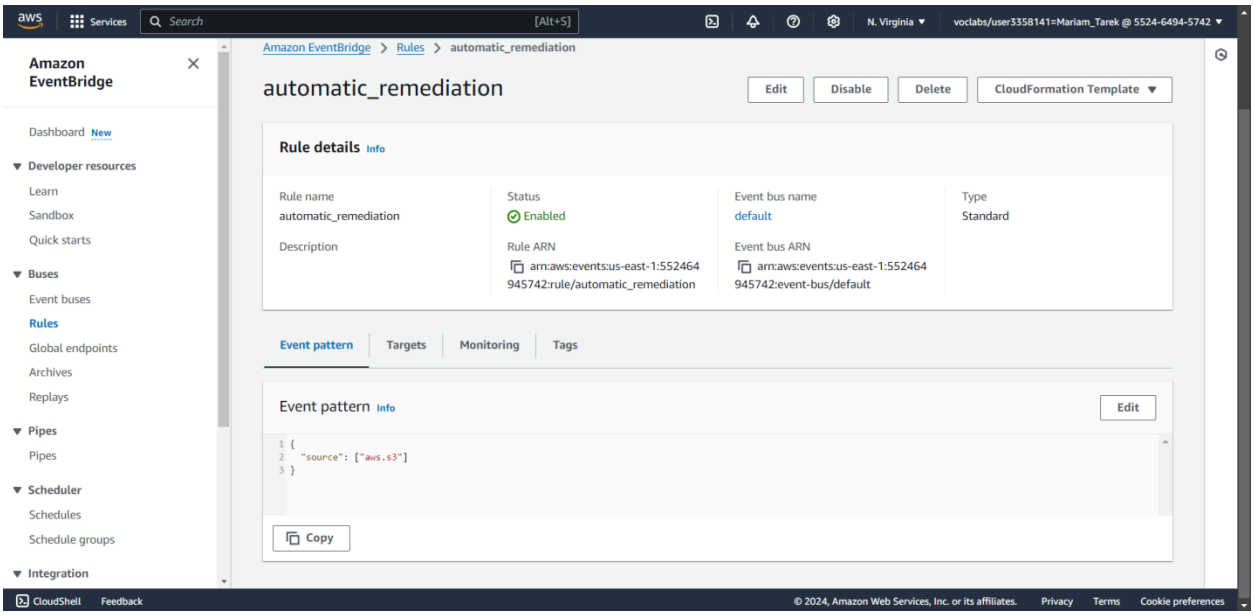

ALARM: "EC2 Machine CPU Usage" in US East (N. Virginia)
AWS Notifications
You are receiving this email because your Amazon CloudWatch Alarm "EC2 Machine CPU Usage" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [23.372024186514224 (22/09/24 10:29:00)] was greater than the threshold (10.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Sunday 22 September, 2024 10:34:05 UTC".
View this alarm in the AWS Management Console:
Alarm Details:
- Name: EC2 Machine CPU Usage
- Description: Hey,
#CPU Utilization has crossed the threshold
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [23.372024186514224 (22/09/24 10:29:00)] was greater than the threshold (10.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Sunday 22 September, 2024 10:34:05 UTC
- AWS Account: 552464945742
- Alarm Arn: arn:aws:cloudwatch:us-east-1:552464945742:alarm:EC2 Machine CPU Usage
Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 10.0 for at least 1 of the last 1 period(s) of 300 seconds.
Monitored Metric:

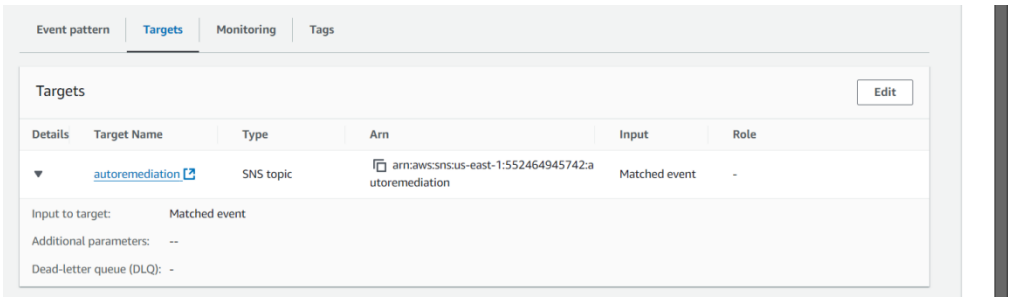

CloudWatch
Alarms (1)
EC2 Machine CPU Usage
Graph
CPUUtilization
CPUUtilization > 5 for 1 datapoints within 1 minute
OK
Percent
5.65
2.82
0
CPUUtilization
Click timeline to see the state change at the selected time.

5. Automatic Remediation

- We used Amazon EventBridge for automatic remediation.

- The rule checks if any S3 bucket has been updated, and if a change is detected, an SNS topic notification will be sent.

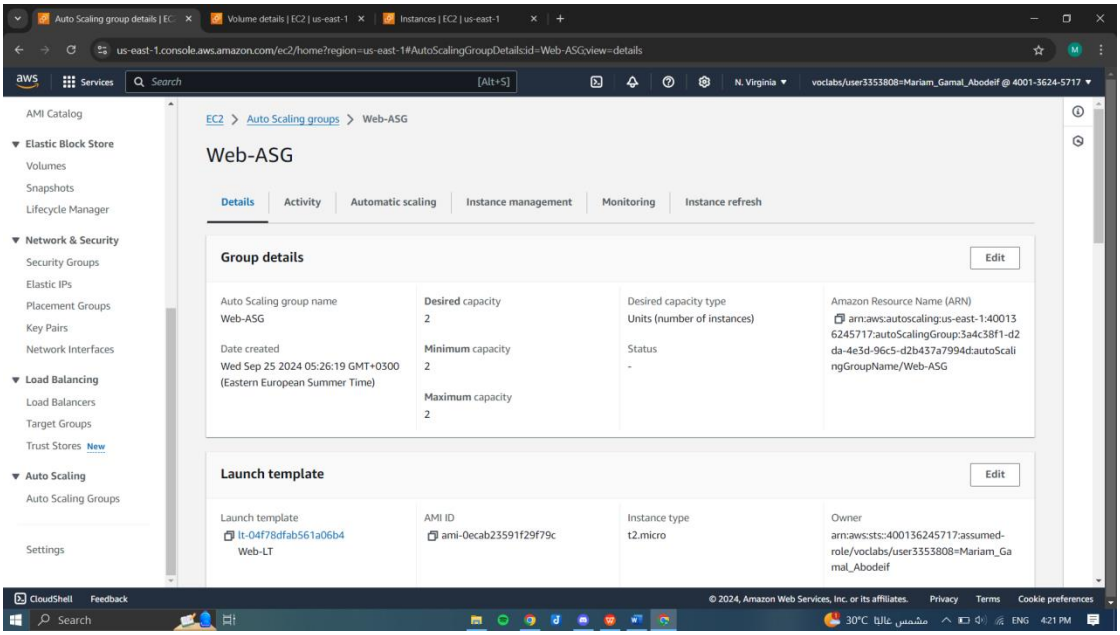


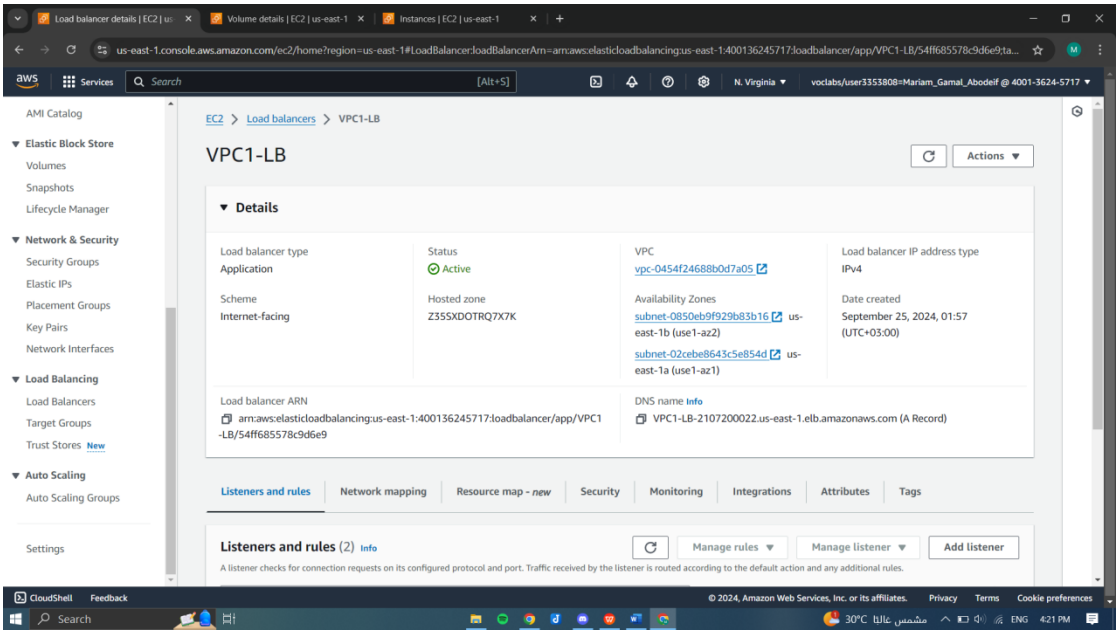


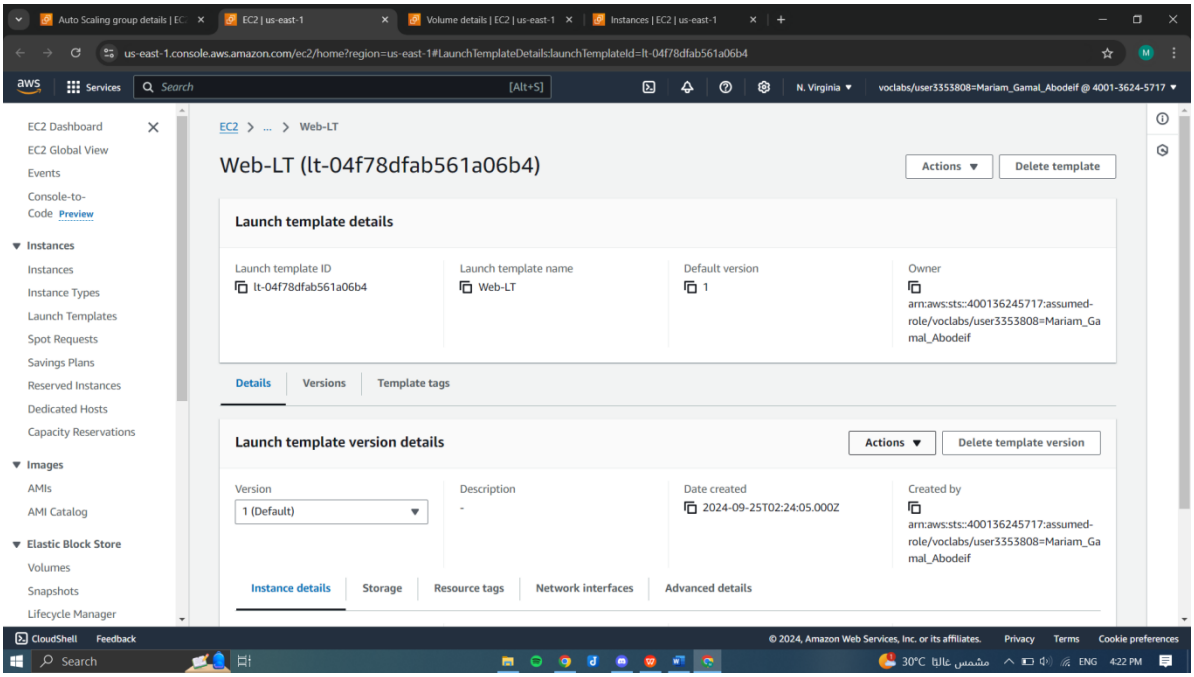
6. Check the environment

Secure: We attached the Lab IAM Role for our environment that Securely grant permissions to services and users.

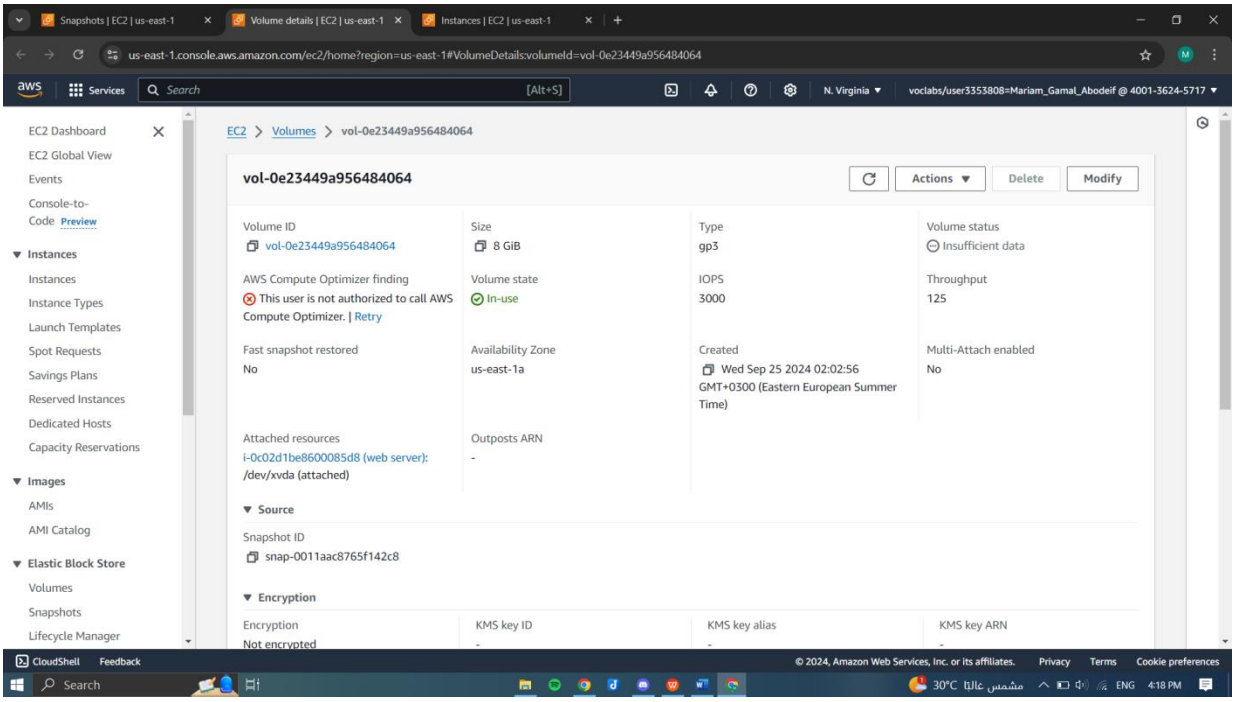
Highly Available – Scalable: We made our environment highly available and scalabe through making an application load balancer and auto scaling group that’s attached with a launch template.

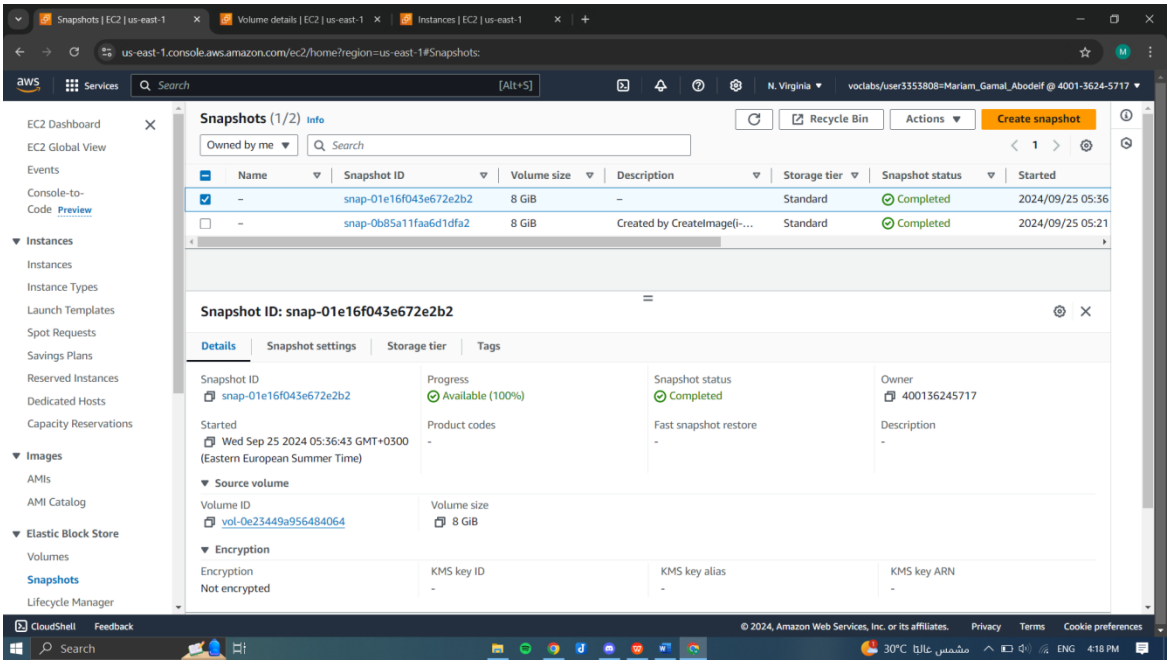






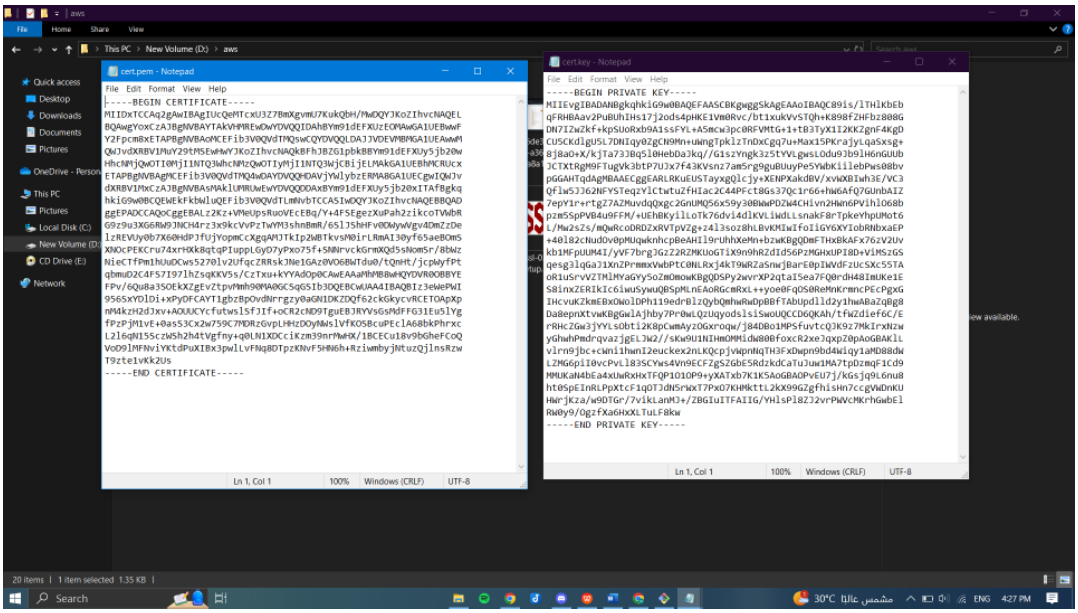
Disaster Recovery: We created a volume attached to our EC2 instance and made a snapshot for it.



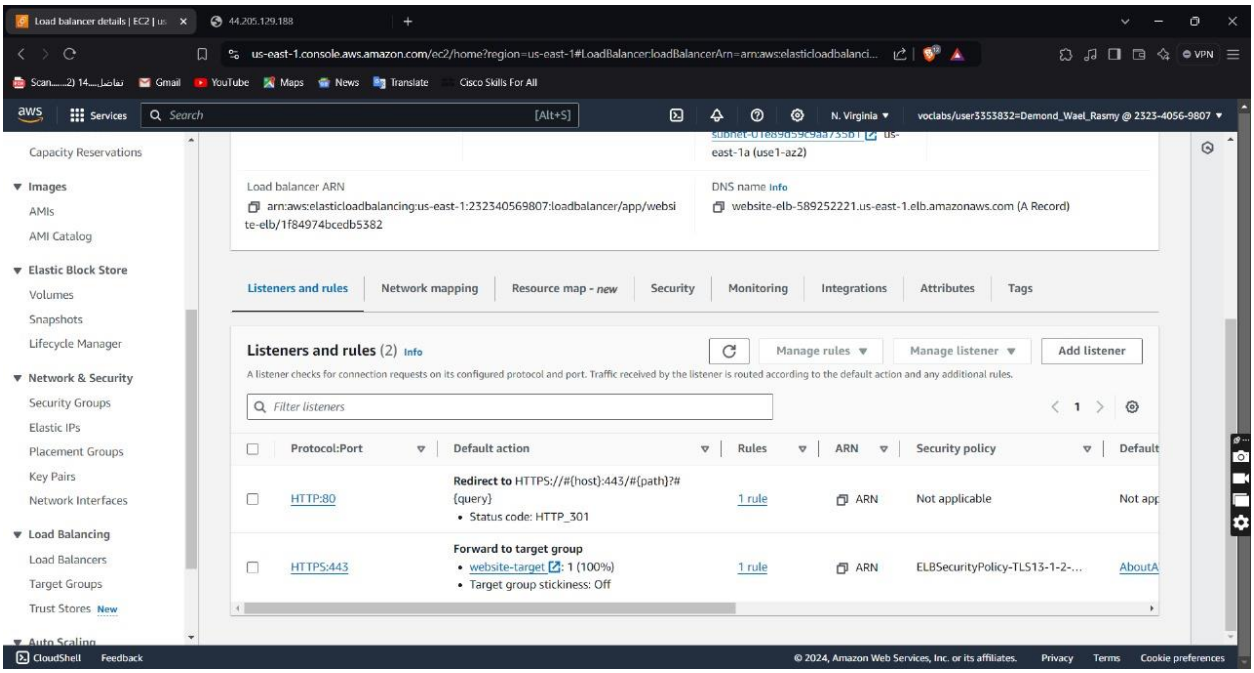


7. Enable HTTP to HTTPS Redirection on Load Balancer

1. First, we created an SSL/TLS certificate through OpenSSL.



Then, we configured the Load balancer listeners to redirect all HTTP traffic (port 80) to HTTPS (port 443).



8. Mount S3 on EC2

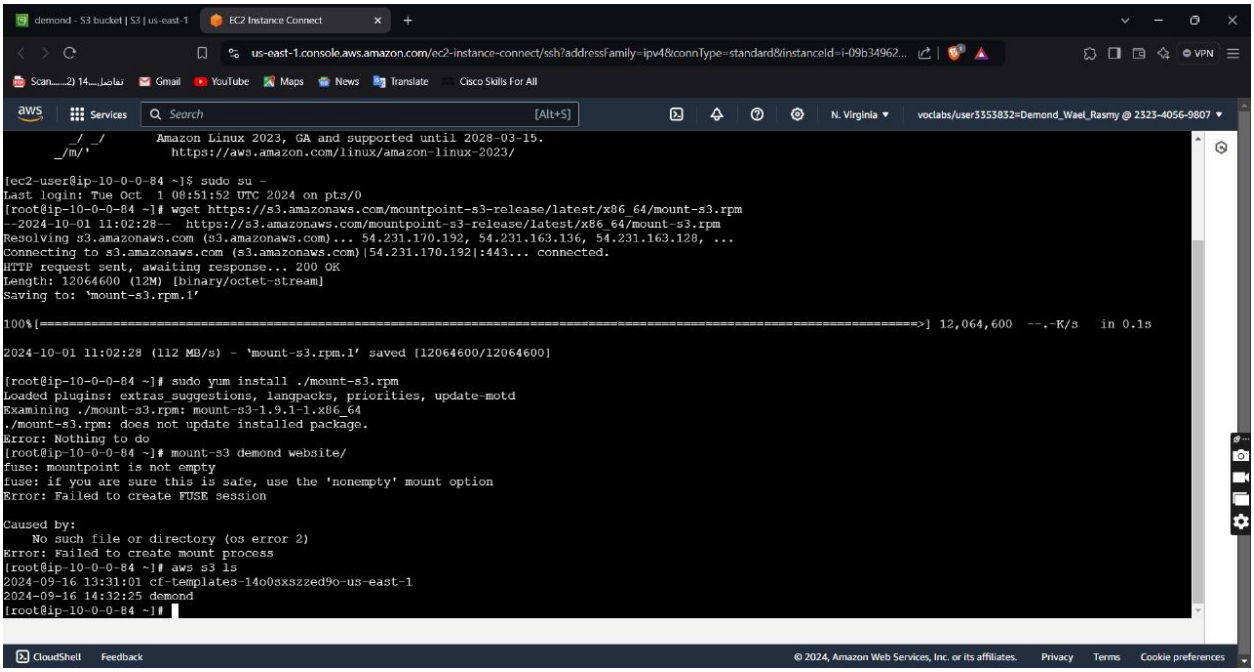
- First, we connected our EC2 instance.

- Then, wed entered the following commands:

$ wget https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm

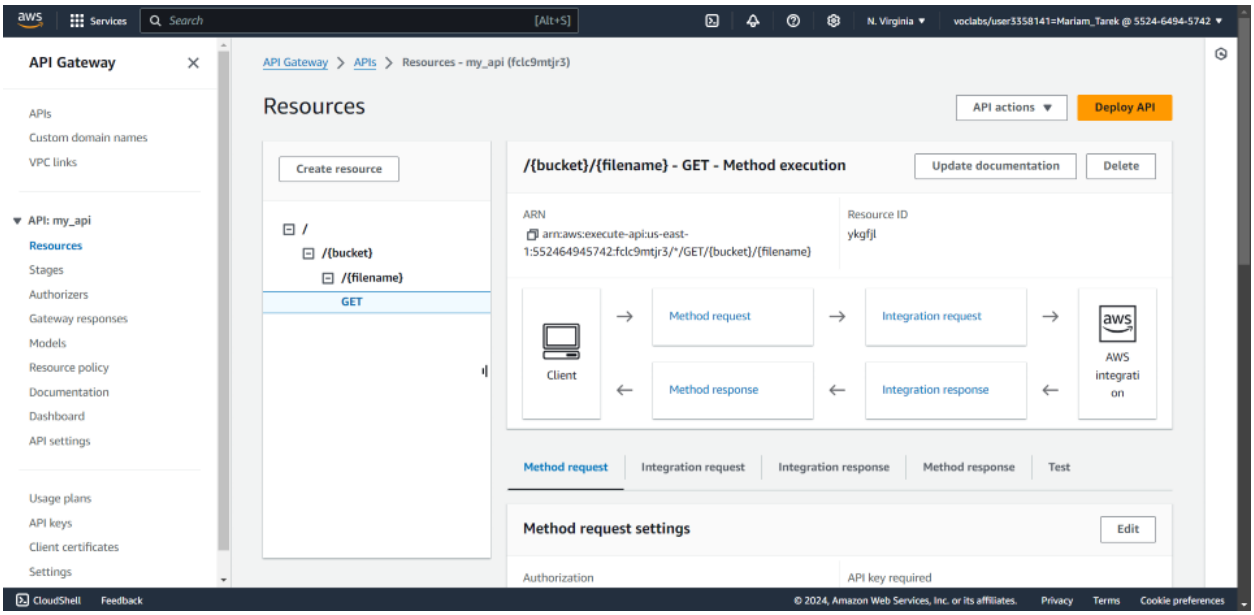
$ sudo yum install ./mount-s3.rpm

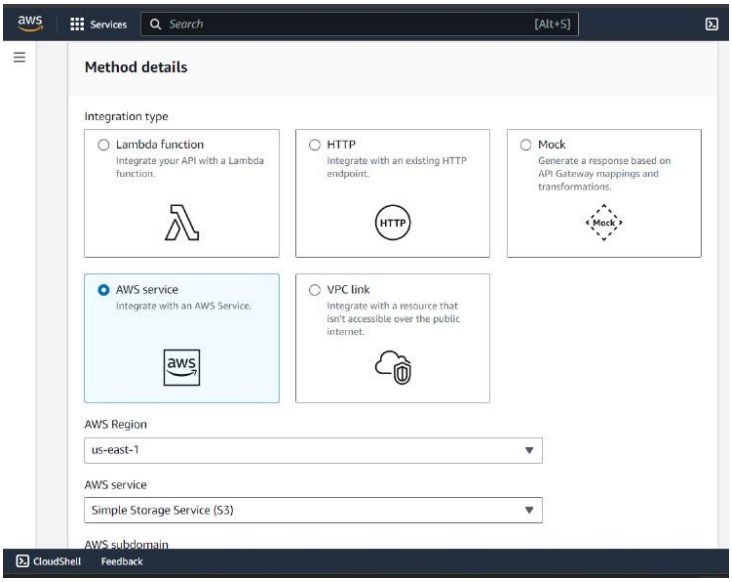
$ sudo mount-s3 your-s3-bucket s3mount/

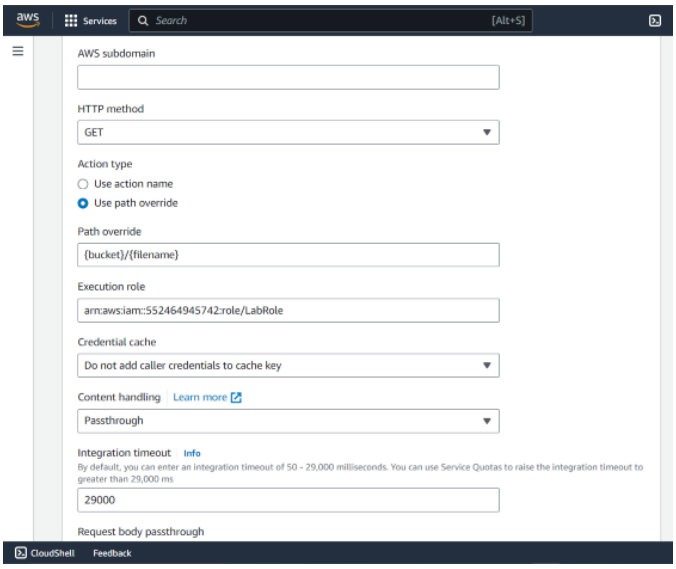


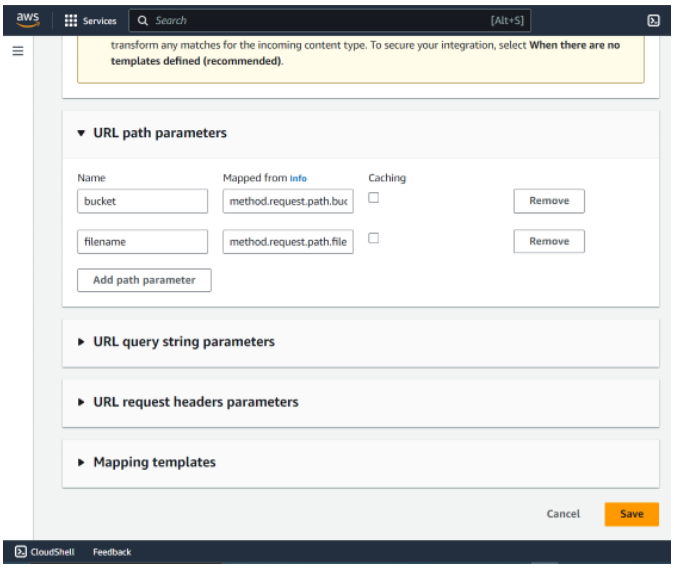
9. Using API Gateway to Fetch Images from S3

1. First, we created an API Gateway with an endpoint that fetches images stored in S3 bucket.

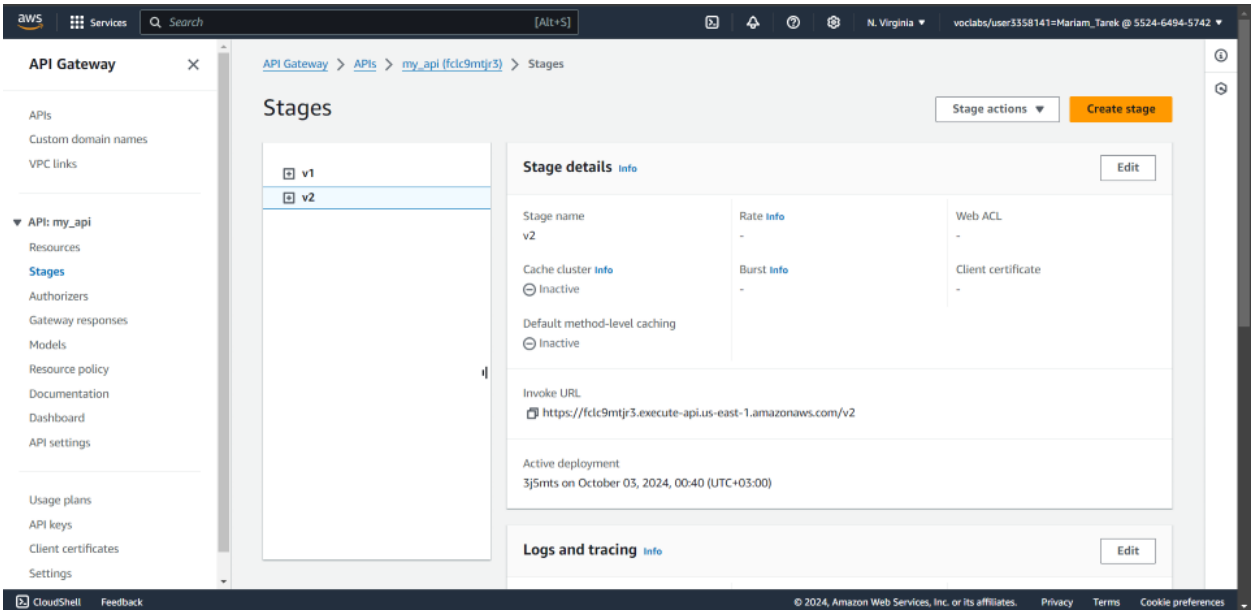



Services
Search
[Alt+S]
Method details
Integration type
Lambda function
Integrate your API with a Lambda function.
HTTP
Integrate with an existing HTTP endpoint.
Mock
Generate a response based on API Gateway mappings and transformations.
AWS service
Integrate with an AWS Service.
VPC link
Integrate with a resource that isn't accessible over the public internet.
AWS Region
us-east-1
AWS service
Simple Storage Service (S3)
AWS subdomain
CloudShell
Feedback


Services
Search
[Alt+S]
AWS subdomain
HTTP method
GET
Action type
Use action name
Use path override
Path override
{bucket}/{filename}
Execution role
arn:aws:iam::552464945742:role/LabRole
Credential cache
Do not add caller credentials to cache key
Content handling
Learn more
Passthrough
Integration timeout
Info
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms
29000
Request body passthrough
CloudShell
Feedback


Services
Search
[Alt+S]
transform any matches for the incoming content type. To secure your integration, select When there are no templates defined (recommended).
URL path parameters
Name
Mapped from
Info
Caching
bucket
method.request.path.buc
Remove
filename
method.request.path.file
Remove
Add path parameter
URL query string parameters
URL request headers parameters
Mapping templates
Cancel
Save
CloudShell
Feedback

- Then, we created a stage to deploy the API



- Finally, we tested it through Postman to view any object from S3 bucket