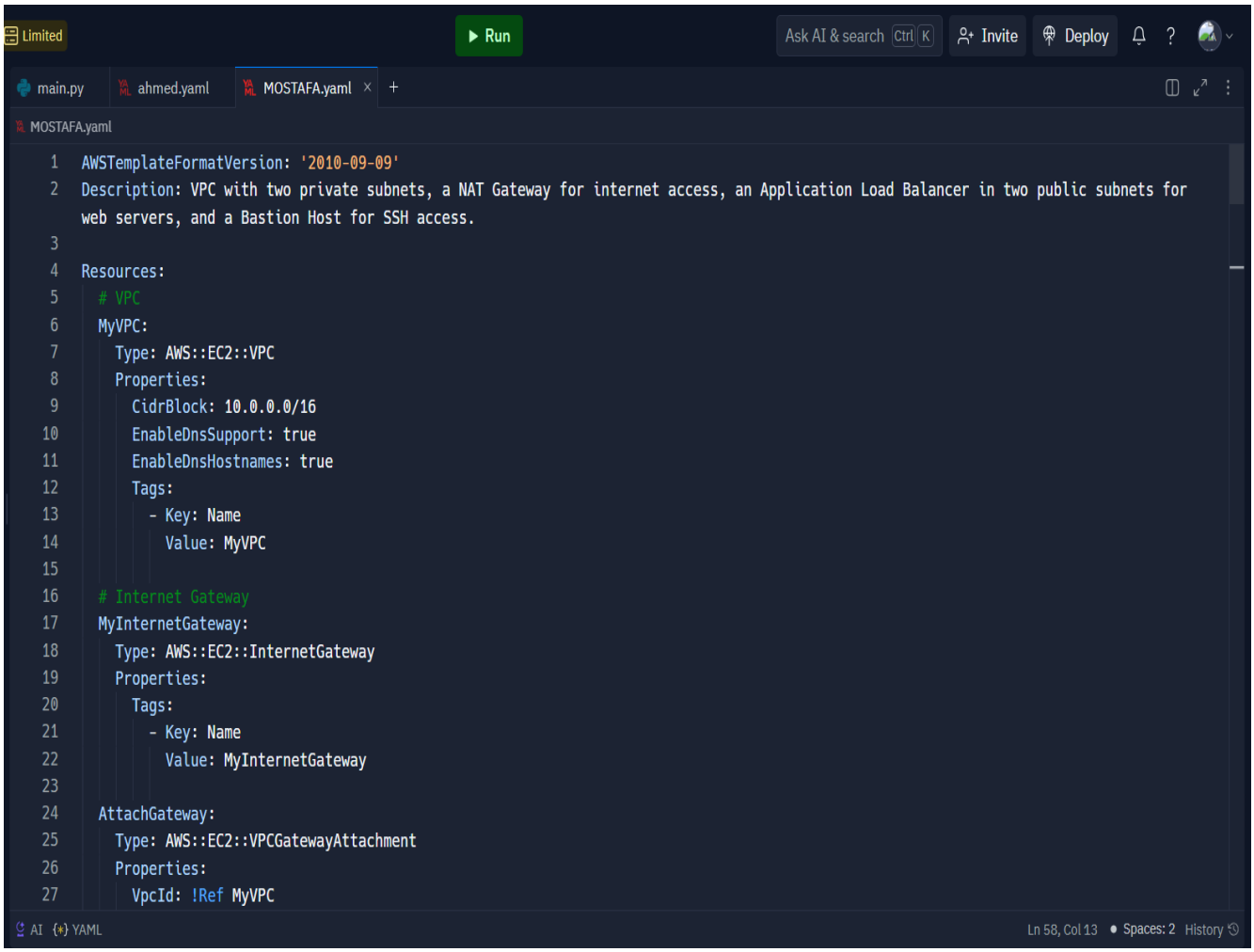
Final Project Documentation:

1. Create Network Environment with Infrastructure as Code (IaC)

Use AWS Cloud Formation or Terraform to define and manage our network infrastructure programmatically.

Use version control for IaC templates, segment our network using Subnets, and secure access with Security Groups and Network ACLs. Deploy a NAT Gateway to allow private instances internet access without exposing them to the public.




```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: VPC with two private subnets, a NAT Gateway for internet access, an Application Load Balancer in two public subnets for web servers, and a Bastion Host for SSH access.

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

  # Internet Gateway
  MyInternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: MyInternetGateway

  AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref MyVPC
```

main.py | ahmed.yaml | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
      VpcId: !Ref MyVPC
      InternetGatewayId: !Ref MyInternetGateway

  # NAT Gateway
  NatEIP:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc

  MyNatGateway:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatEIP.AllocationId
      SubnetId: !Ref PublicSubnet1
      Tags:
        - Key: Name
          Value: MyNatGateway

  # Route Tables
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref MyVPC
      Tags:
        - Key: Name
          Value: PublicRouteTable

  PrivateRouteTable:
```

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
  PrivateRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref MyVPC
      Tags:         - Key: Name
          Value: PrivateRouteTable

  # Public Route
  PublicRoute:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref MyInternetGateway

  # Private Route
  PrivateRoute:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref MyNatGateway

  # Subnets
  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
```

AI {*} YAML Ln 54, Col 21 • Spaces: 2 History

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone: us-east-1a
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: PublicSubnet1

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.2.0/24
      AvailabilityZone: us-east-1b
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: PublicSubnet2

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.3.0/24
      AvailabilityZone: us-east-1a
      MapPublicIpOnLaunch: false
      Tags:
```

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
      Tags:
        - Key: Name
          Value: PrivateSubnet1

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.4.0/24
      AvailabilityZone: us-east-1b
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: PrivateSubnet2

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnet1
      RouteTableId: !Ref PublicRouteTable

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable

  PrivateSubnet1RouteTableAssociation:
```

AI {•} YAML Ln 134, Col 39 • Spaces: 2 History

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet1
      RouteTableId: !Ref PrivateRouteTable

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable

  # Security Group for Bastion Host
  BastionSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow SSH from anywhere
      VpcId: !Ref MyVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0  # Limit this to a specific IP for security

  # Bastion EC2 Instance (in Public Subnet)
  BastionHost:
    Type: AWS::EC2::Instance
    Properties:
```

AI {*} YAML Ln 161, Col 16 • Spaces: 2 History

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
  BastionHost:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      KeyName: vockey  # Replace with your KeyPair name
      ImageId: ami-04d40dba56f6e1303  # Replace with the latest Amazon Linux AMI for your region
      NetworkInterfaces:
        - AssociatePublicIpAddress: true
          DeviceIndex: 0
          SubnetId: !Ref PublicSubnet1
          GroupSet:
            - !Ref BastionSecurityGroup
      Tags:
        - Key: Name
          Value: BastionHost

  # Security Group for Web Servers (allow SSH from Bastion Host)
  WebServerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP, HTTPS, and SSH from Bastion
      VpcId: !Ref MyVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          SourceSecurityGroupId: !Ref ALBSecurityGroup
        - IpProtocol: tcp
```

AI {*} YAML Ln 186, Col 26 • Spaces: 2 History

main.py MOSTAFA.yaml

MOSTAFA.yaml

```yaml
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          SourceSecurityGroupId: !Ref ALBSecurityGroup
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          SourceSecurityGroupId: !Ref BastionSecurityGroup

  # ALB Security Group
  ALBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP and HTTPS access
      VpcId: !Ref MyVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0

  # Web Server Launch Template
  WebServerLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
```

AI {•} YAML Ln 213, Col 26 • Spaces: 2 History

main.py MOSTAFA.yaml

MOSTAFA.yaml

```yaml

  # Web Server Launch Template
  WebServerLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: WebServerLaunchTemplate
      LaunchTemplateData:
        KeyName: vockey
        ImageId: ami-04d40dba56f6e1303
        InstanceType: t2.micro
        SecurityGroupIds:
          - !Ref WebServerSG
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            yum update -y
            yum install -y httpd
            systemctl start httpd
            systemctl enable httpd
            echo "<h1>Web Server launched by Auto Scaling Group</h1>" > /var/www/html/index.html

  # Auto Scaling Group for Web Servers
  WebServerAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
```

AI {•} YAML Ln 213, Col 26 • Spaces: 2 History

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
        - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
      LaunchTemplate:
        LaunchTemplateId: !Ref WebServerLaunchTemplate
        Version: !GetAtt WebServerLaunchTemplate.LatestVersionNumber
      MinSize: '2'
      MaxSize: '4'
      DesiredCapacity: '2'
      TargetGroupARNs:
        - !Ref ALBTargetGroup
      HealthCheckType: ELB
      HealthCheckGracePeriod: 300
      Tags:
        - Key: Name
          Value: WebServer
          PropagateAtLaunch: true

  # ALB
  ALB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: MyALB
      Subnets:
        - !Ref PublicSubnet1
        - !Ref PublicSubnet2
      Scheme: internet-facing
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '60'
```

AI {*} YAML Ln 213, Col 26 Spaces: 2 History

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '60'
      SecurityGroups:
        - !Ref ALBSecurityGroup
      Tags:
        - Key: Name
          Value: MyALB

  ALBTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      VpcId: !Ref MyVPC
      Port: 80
      Protocol: HTTP
      TargetType: instance
      HealthCheckProtocol: HTTP
      HealthCheckPort: 80
      HealthCheckPath: /
      HealthCheckIntervalSeconds: 30
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 3
      UnhealthyThresholdCount: 2
      Tags:
        - Key: Name
          Value: ALBTargetGroup

  ALBListener:
```

main.py | MOSTAFA.yaml

MOSTAFA.yaml

```yaml
  ALBListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn: !Ref ALB
      Port: 80
      Protocol: HTTP
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref ALBTargetGroup

Outputs:
  VPCId:
    Description: The ID of the VPC
    Value: !Ref MyVPC

  PublicSubnet1Id:
    Description: The ID of Public Subnet 1
    Value: !Ref PublicSubnet1

  PublicSubnet2Id:
    Description: The ID of Public Subnet 2
    Value: !Ref PublicSubnet2

  PrivateSubnet1Id:
    Description: The ID of Private Subnet 1
    Value: !Ref PrivateSubnet1
```

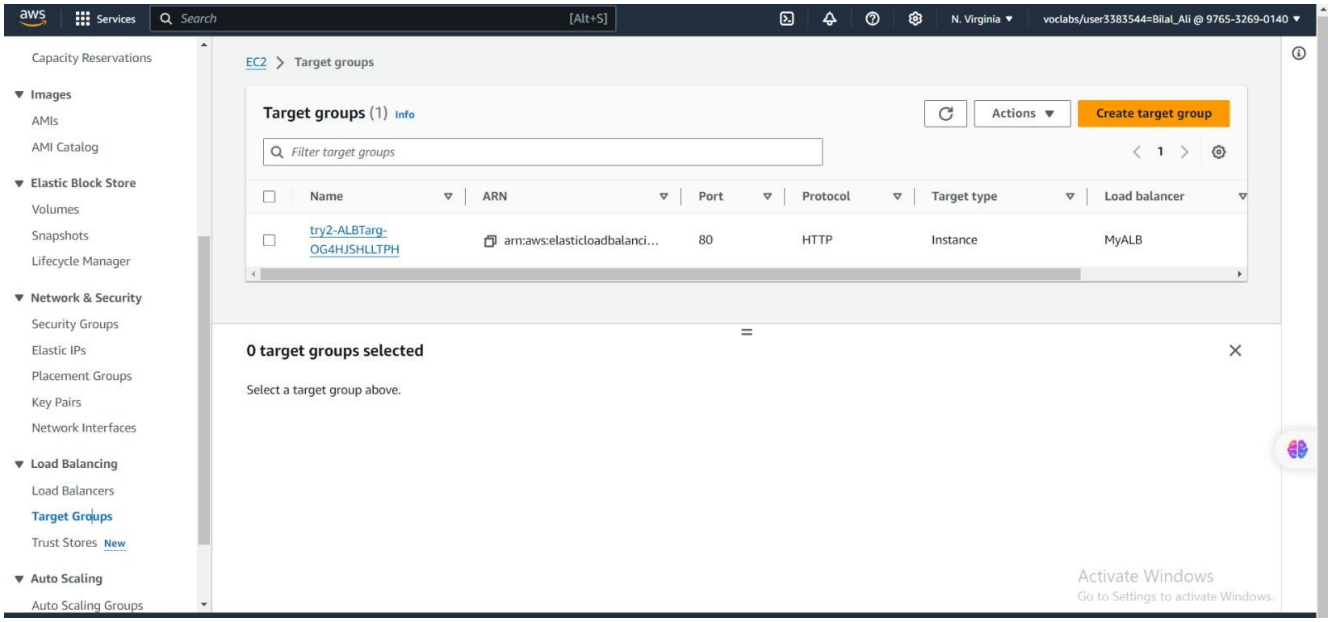
AI {*} YAML Ln 315, Col 1 • Spaces: 2 History

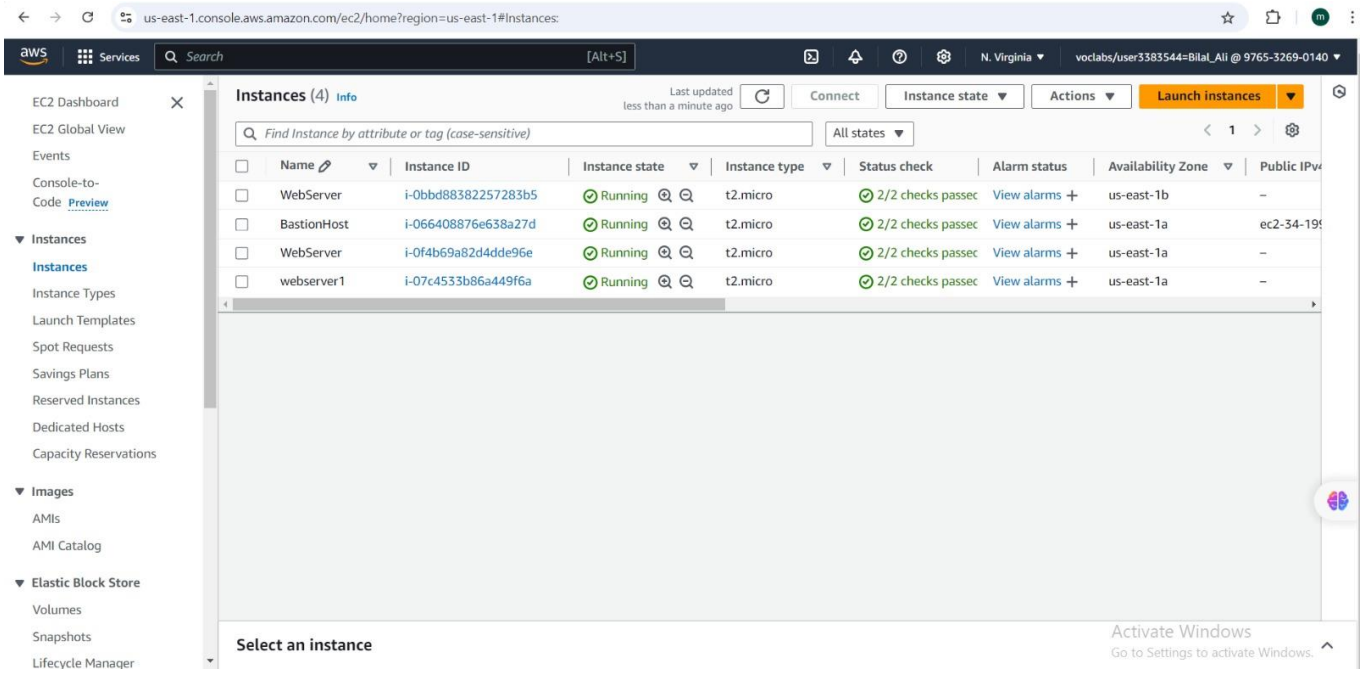
main.py | MOSTAFA.yaml

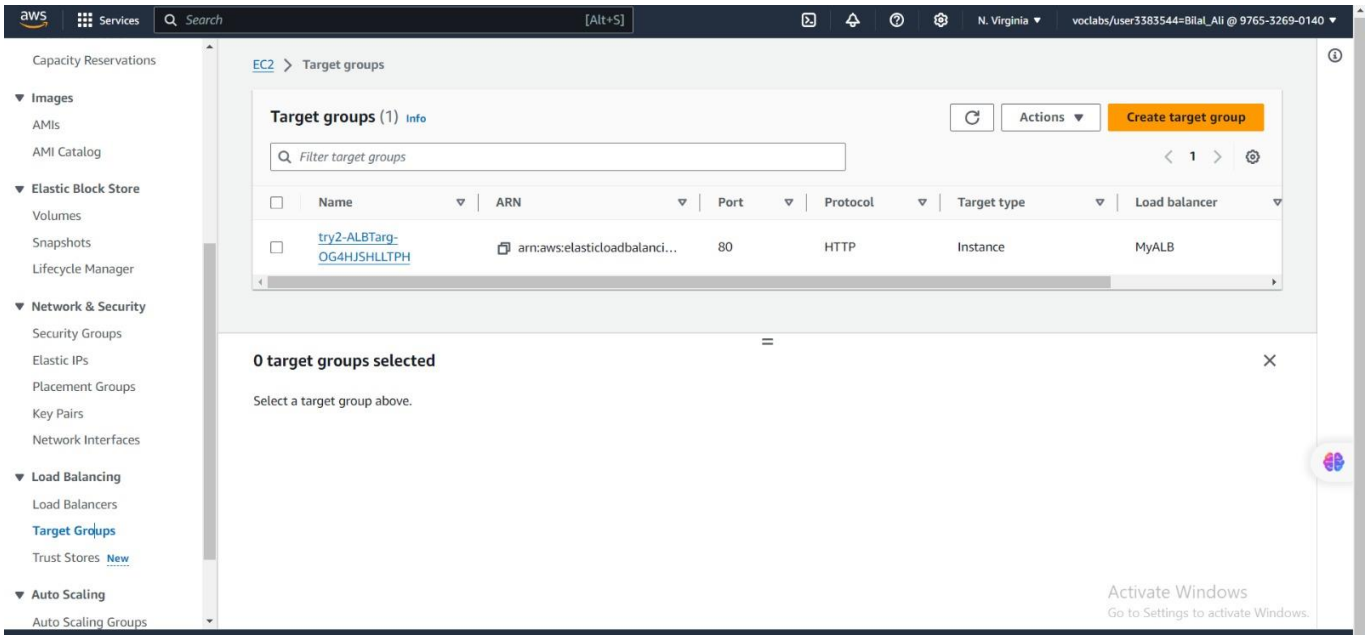
MOSTAFA.yaml

```yaml
  PrivateSubnet1Id:
    Description: The ID of Private Subnet 1
    Value: !Ref PrivateSubnet1

  PrivateSubnet2Id:
    Description: The ID of Private Subnet 2
    Value: !Ref PrivateSubnet2

  LoadBalancerDNSName:
    Description: The DNS name of the ALB
    Value: !GetAtt ALB.DNSName

  BastionHostPublicIp:
    Description: Public IP address of the Bastion Host
    Value: !GetAtt BastionHost.PublicIp
```

AI {*} YAML Ln 294, Col 21 • Spaces: 2 History


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3383544=Bilal_Ali @ 9765-3269-0140
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
EC2 > Target groups
Target groups (1) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
try2-ALBTarg-OG4HJSHLLTPH
arn:aws:elasticloadbalanci...
80
HTTP
Instance
MyALB
0 target groups selected
Select a target group above.
Activate Windows
Go to Settings to activate Windows.

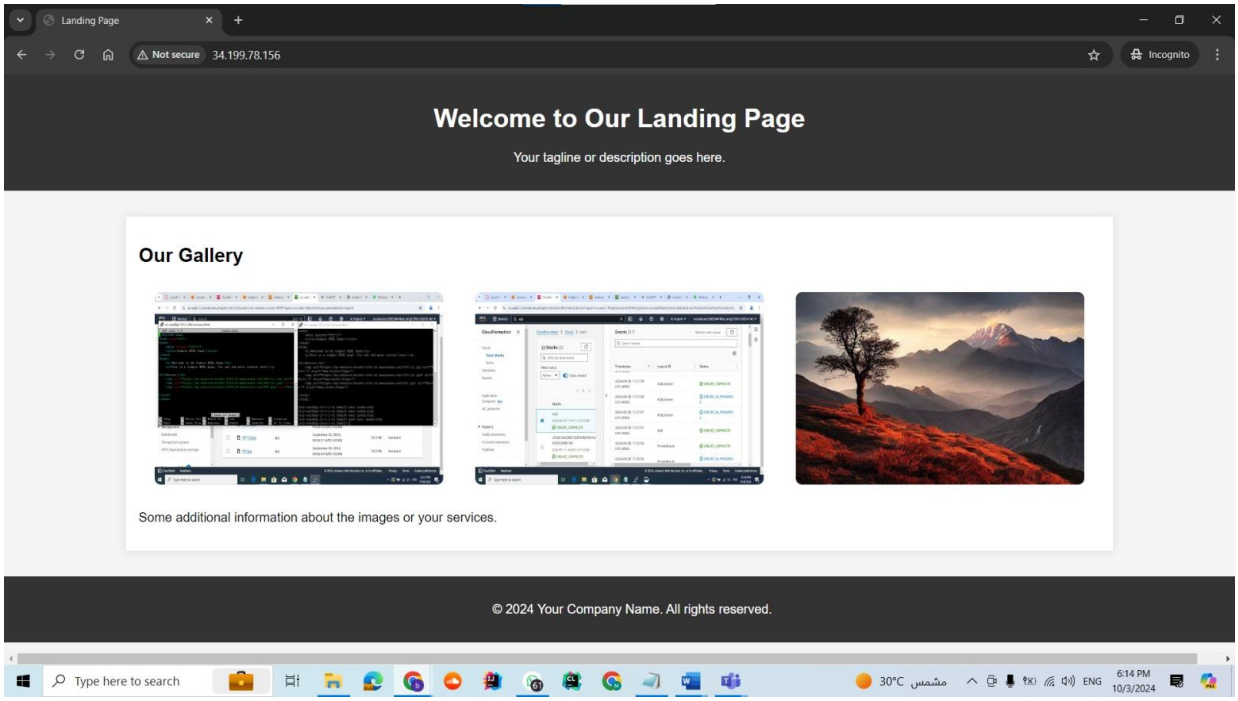

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3383544=Bilal_Ali @ 9765-3269-0140
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (4) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4
WebServer
i-0bbd88382257283b5
Running
t2.micro
2/2 checks passed
View alarms
us-east-1b
BastionHost
i-066408876e638a27d
Running
t2.micro
2/2 checks passed
View alarms
us-east-1a
ec2-34-19
WebServer
i-0f4b69a82d4dde96e
Running
t2.micro
2/2 checks passed
View alarms
us-east-1a
webserver1
i-07c4533b86a449f6a
Running
t2.micro
2/2 checks passed
View alarms
us-east-1a
Select an instance
Activate Windows
Go to Settings to activate Windows.

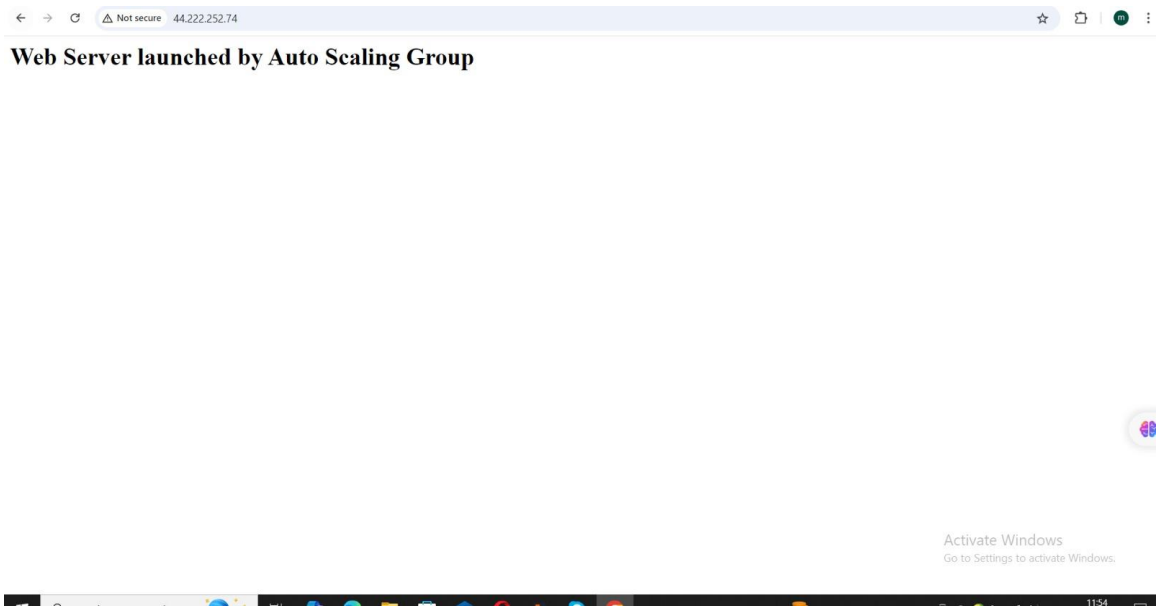


2. Host Web App on EC2 or Using Microservices

Use Amazon EC2 or Amazon ECS/EKS for deploying web apps or containerized microservices.

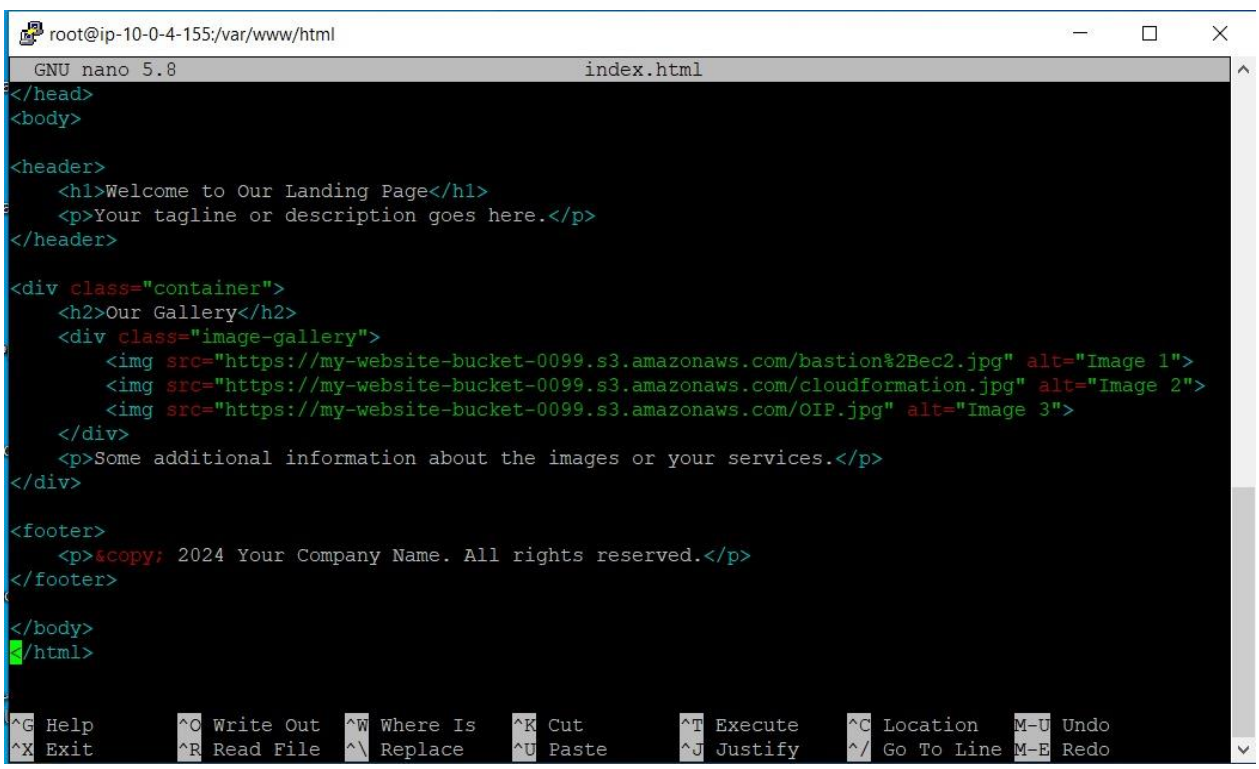
Use Auto Scaling to ensure high availability and scalability. Deploy Elastic Load Balancers across multiple Availability Zones for redundancy and traffic distribution.

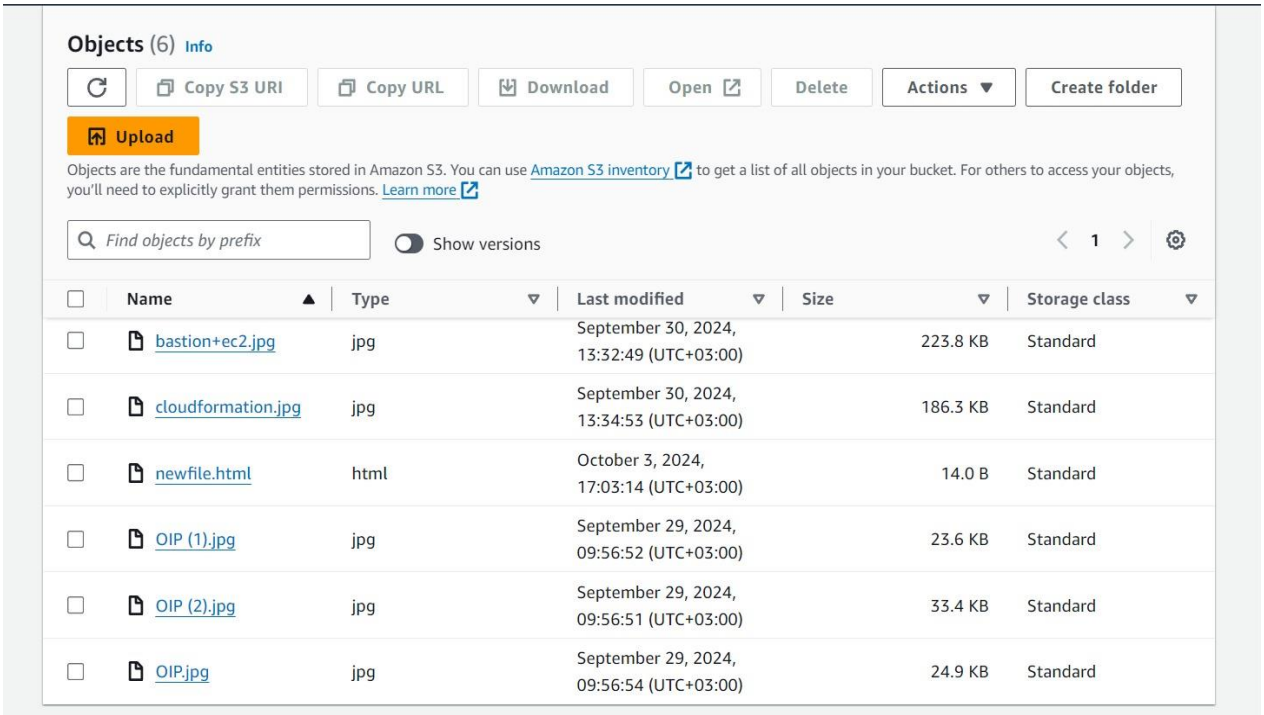




3. Store App Static Content on External Storage (S3)

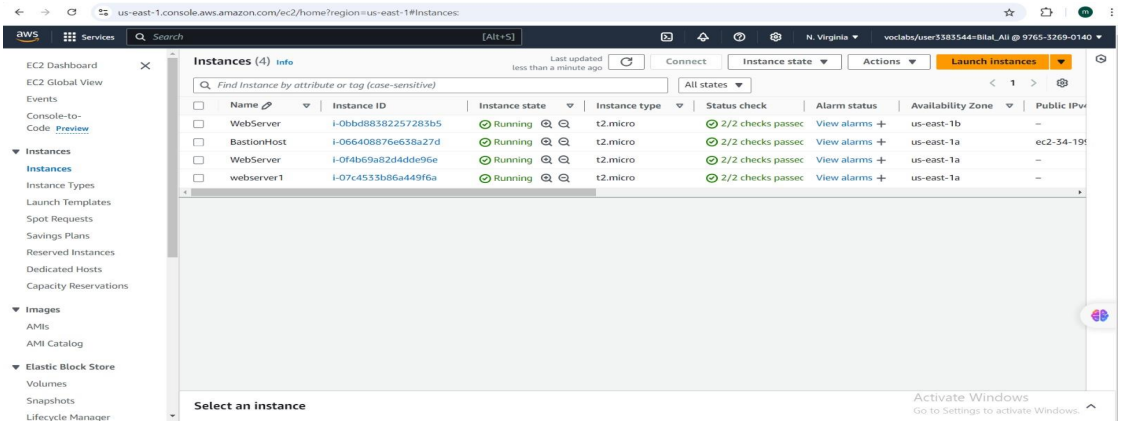
Enable S3 Versioning for backup and recovery and use CloudFront as a CDN to improve latency and user experience globally.



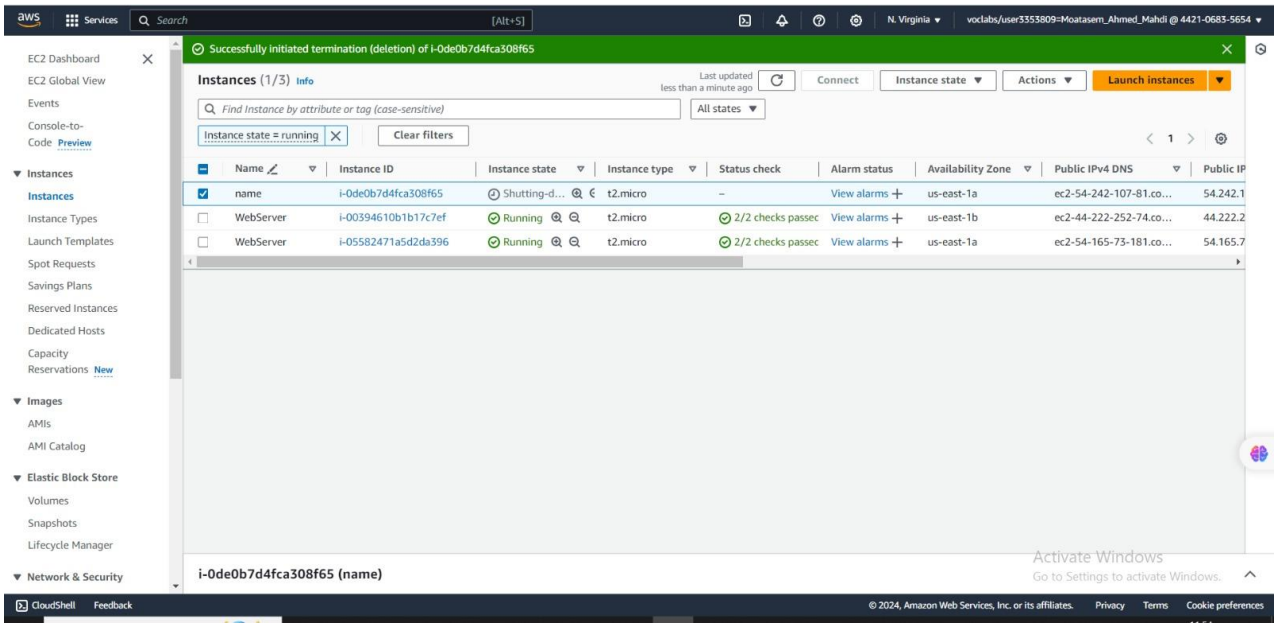


4. Secure, Scalable, High Availability, and Disaster Recovery

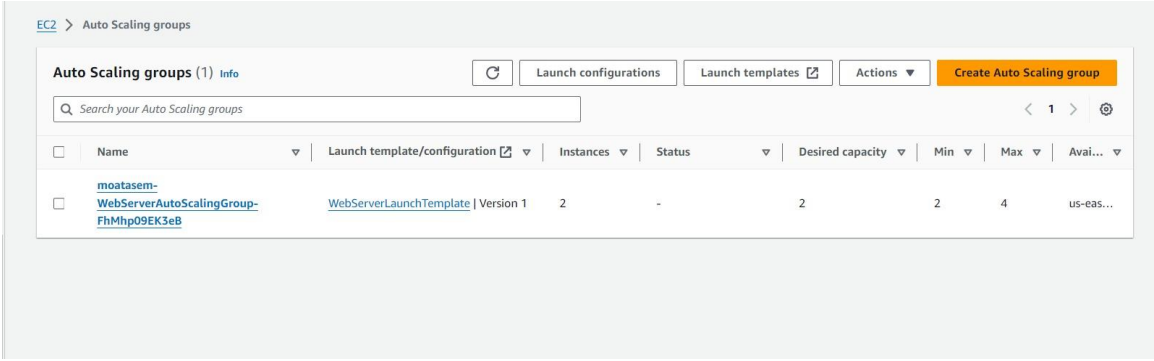
Use IAM for security, Auto Scaling for scalability, and Multi-AZ deployment for high availability.



Highly available with two availability zones



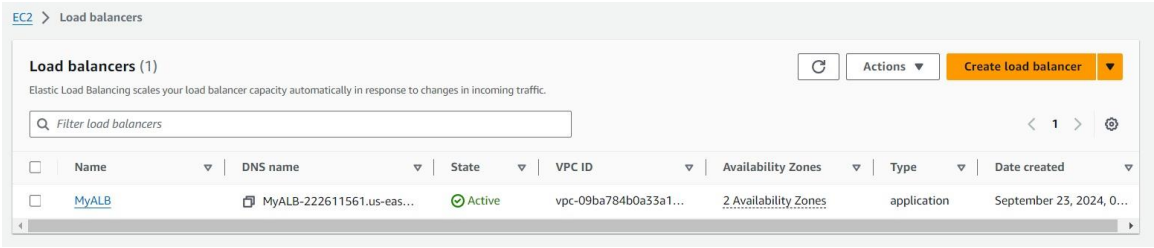
Web server multi-AZ



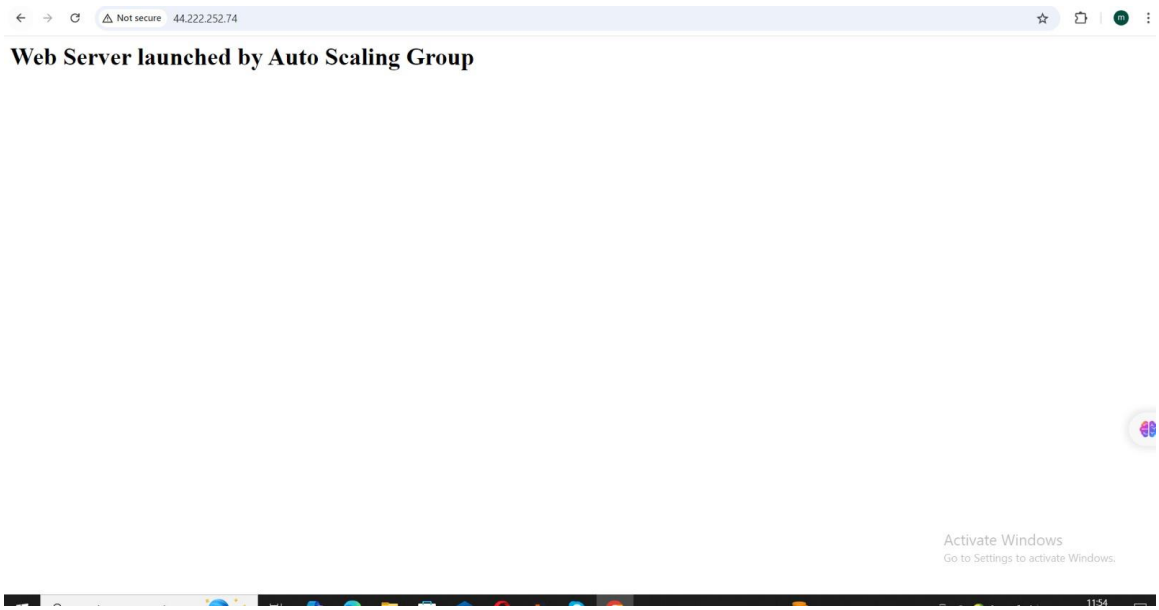
Auto scaling

5. Enable Redirection on Load Balancer (HTTP to HTTPS)

: Use an Application Load Balancer to redirect HTTP requests to HTTPS.

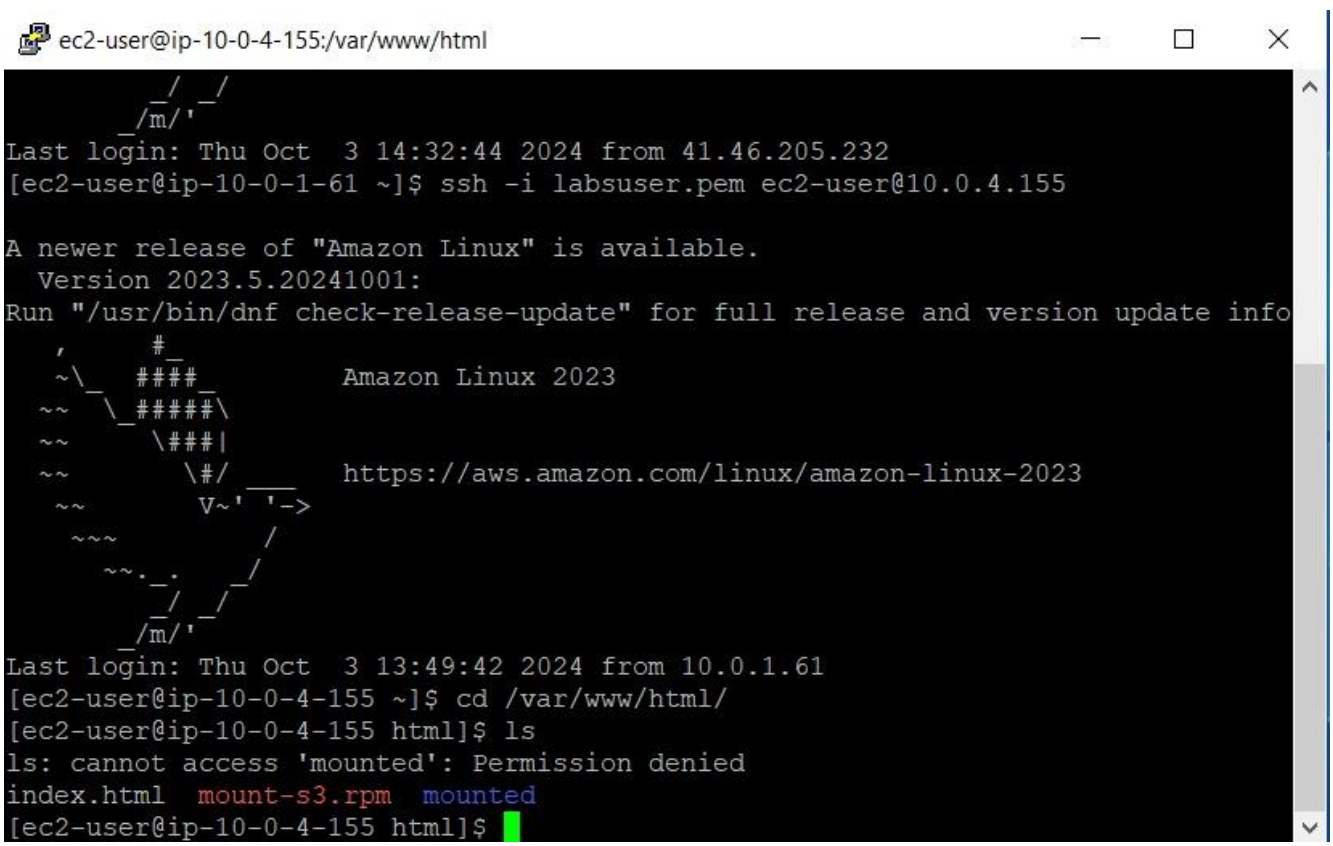


Use SSL/TLS certificates self-signed certificate to ensure encrypted communications.



6. Mount S3 on EC2

: Use s3fs or AWS CLI to mount an S3 bucket on your EC2 instance.



```
root@ip-10-0-4-155:/var/www/html/mounted
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Oct  3 13:49:42 2024 from 10.0.1.61
[ec2-user@ip-10-0-4-155 ~]$ cd /var/www/html/
[ec2-user@ip-10-0-4-155 html]$ ls
ls: cannot access 'mounted': Permission denied
index.html  mount-s3.rpm  mounted
[ec2-user@ip-10-0-4-155 html]$ cd mounted
-bash: cd: mounted: Permission denied
[ec2-user@ip-10-0-4-155 html]$ sudo su
[root@ip-10-0-4-155 html]# cd mounted/
[root@ip-10-0-4-155 mounted]# ls
'OIP (1).jpg'   OIP.jpg           cloudformation.jpg
'OIP (2).jpg'   bastion+ec2.jpg   newfile.html
[root@ip-10-0-4-155 mounted]#
```

user@34.199.78.156

sudo nano /etc/httpd/conf.d/reverse-proxy.conf

sudo apachectl configtest

```
<VirtualHost *:80>

   ServerName 34.199.78.156  # Replace with your Bastion host's public IP


   ProxyPreserveHost On

   ProxyPass / http://10.0.3.84:80/  # Replace with your EC2 private IP

   ProxyPassReverse / http://10.0.3.84:80/


</VirtualHost>
```

---

s3fs my-website-bucket-0099 ~/s3-bucket -o passwd_file=~/.passwd-s3fs -o allow_other

sudo s3fs my-website-bucket-0099 /mnt/s3bucket1 -o iam_role=EMR_DefaultRole -o use_cache=/tmp -o allow_other -o uid=1001 -o mp_umask=002 -o multireq_max=5 -o use_path_request_style -o url=https://s3-{{us-east-1}}.amazonaws.com