IS 698 – Special Topics in Information Systems

CLOUD COMPUTING

HOMEWORK 2

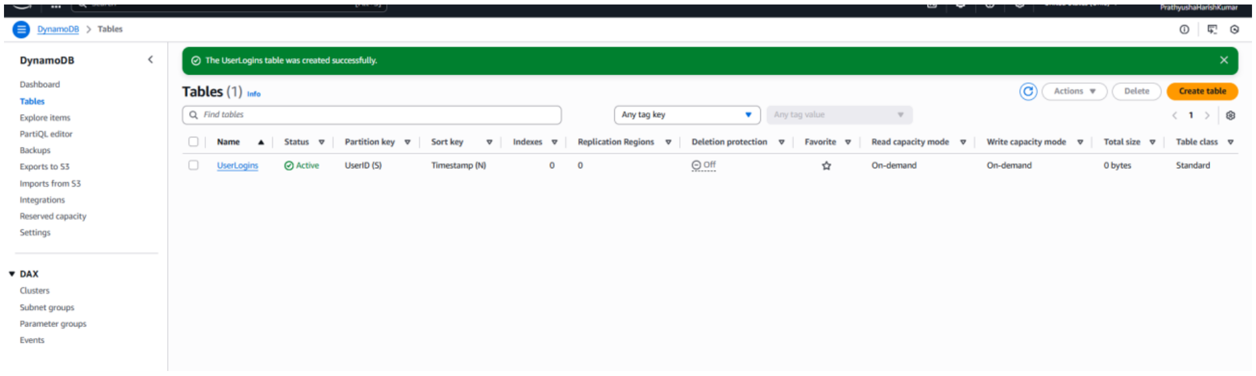
Submitted By

Prathyusha Harish Kumar
JB24771

Part 1: AWS Networking and Database

Task 1: DynamoDB Table Creation and Interaction (10 points)

- *Create a **DynamoDB Table** with the following schema:*
  - *Primary Key: `UserID` (String)*
  - *Sort Key: `Timestamp` (Number)*
  - *Attributes: `Name` (String), `Email` (String), `LastLogin` (String)*

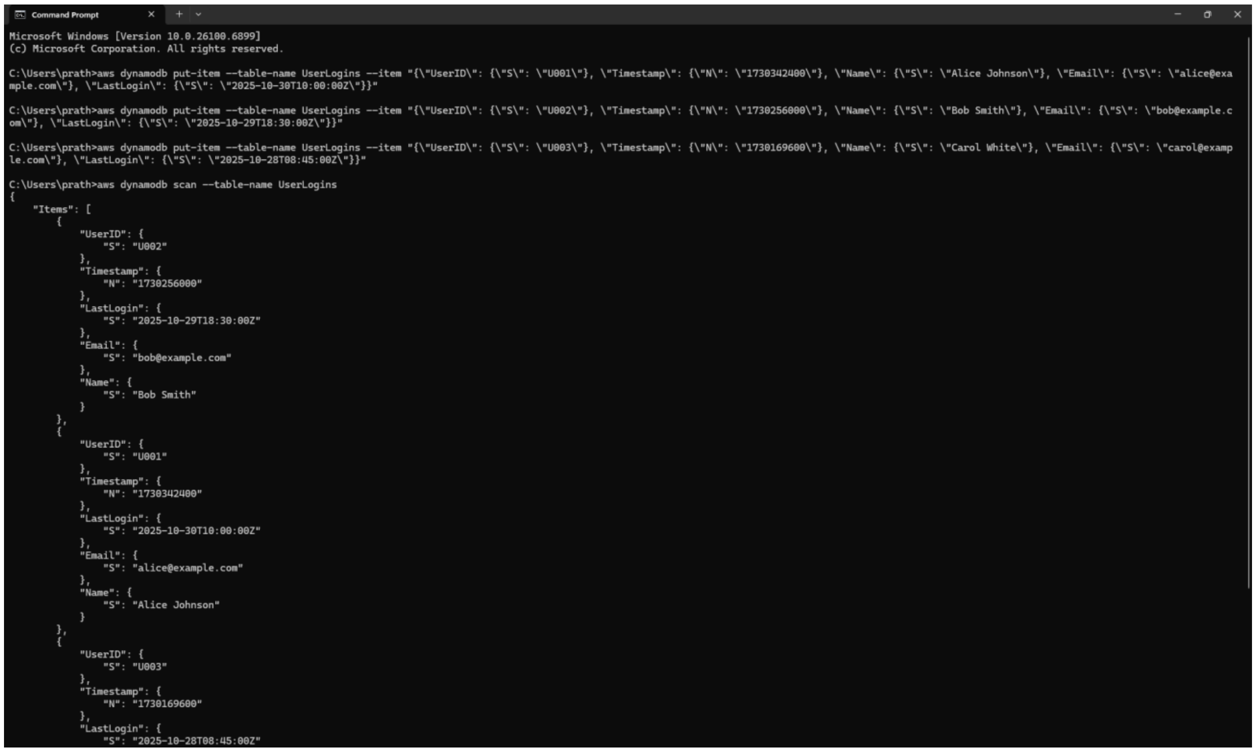


The UserLogins table was created successfully.

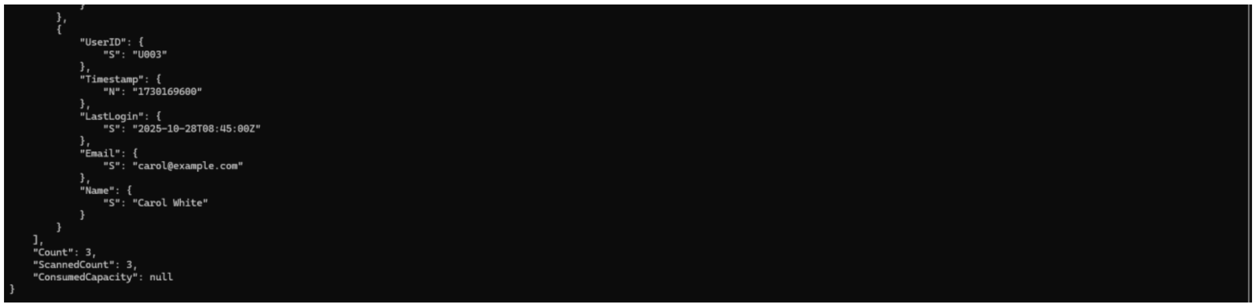
Tables (1) Info

	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode	Total size	Table class
	UserLogins	Active	UserID (S)	Timestamp (N)	0	0	Off	☆	On-demand	On-demand	0 bytes	Standard

- *Insert sample data using the AWS CLI or Python **Boto3**.*

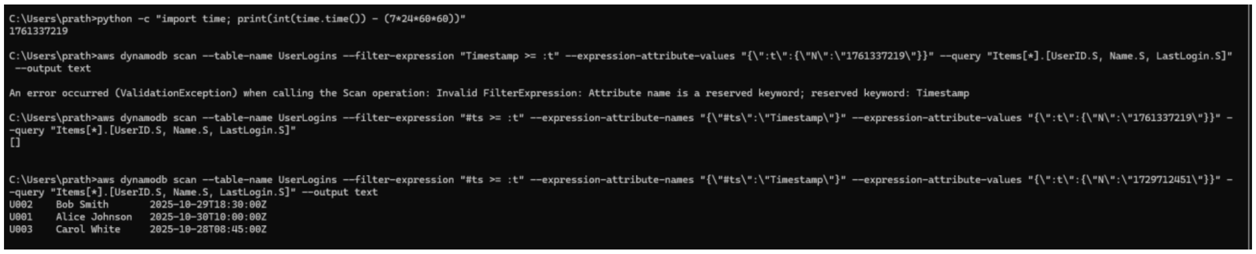


```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\prath>aws dynamodb put-item --table-name UserLogins --item "{\"UserID\": {\"S\": \"U001\"}, \"Timestamp\": {\"N\": \"1730342400\"}, \"Name\": {\"S\": \"Alice Johnson\"}, \"Email\": {\"S\": \"alice@exa
mple.com\"}, \"LastLogin\": {\"S\": \"2025-10-30T10:00:00Z\"}}"

C:\Users\prath>aws dynamodb put-item --table-name UserLogins --item "{\"UserID\": {\"S\": \"U002\"}, \"Timestamp\": {\"N\": \"1730256000\"}, \"Name\": {\"S\": \"Bob Smith\"}, \"Email\": {\"S\": \"bob@example.c
om\"}, \"LastLogin\": {\"S\": \"2025-10-29T18:30:00Z\"}}"

C:\Users\prath>aws dynamodb put-item --table-name UserLogins --item "{\"UserID\": {\"S\": \"U003\"}, \"Timestamp\": {\"N\": \"1730169600\"}, \"Name\": {\"S\": \"Carol White\"}, \"Email\": {\"S\": \"carol@examp
le.com\"}, \"LastLogin\": {\"S\": \"2025-10-28T08:45:00Z\"}}"

C:\Users\prath>aws dynamodb scan --table-name UserLogins
{
    "Items": [
        {
            "UserID": {
                "S": "U002"
            },
            "Timestamp": {
                "N": "1730256000"
            },
            "LastLogin": {
                "S": "2025-10-29T18:30:00Z"
            },
            "Email": {
                "S": "bob@example.com"
            },
            "Name": {
                "S": "Bob Smith"
            }
        },
        {
            "UserID": {
                "S": "U001"
            },
            "Timestamp": {
                "N": "1730342400"
            },
            "LastLogin": {
                "S": "2025-10-30T10:00:00Z"
            },
            "Email": {
                "S": "alice@example.com"
            },
            "Name": {
                "S": "Alice Johnson"
            }
        },
        {
            "UserID": {
                "S": "U003"
            },
            "Timestamp": {
                "N": "1730169600"
            },
            "LastLogin": {
                "S": "2025-10-28T08:45:00Z"
```



```
        },
        {
            "UserID": {
                "S": "U003"
            },
            "Timestamp": {
                "N": "1730169600"
            },
            "LastLogin": {
                "S": "2025-10-28T08:45:00Z"
            },
            "Email": {
                "S": "carol@example.com"
            },
            "Name": {
                "S": "Carol White"
            }
        }
    ],
    "Count": 3,
    "ScannedCount": 3,
    "ConsumedCapacity": null
}
```

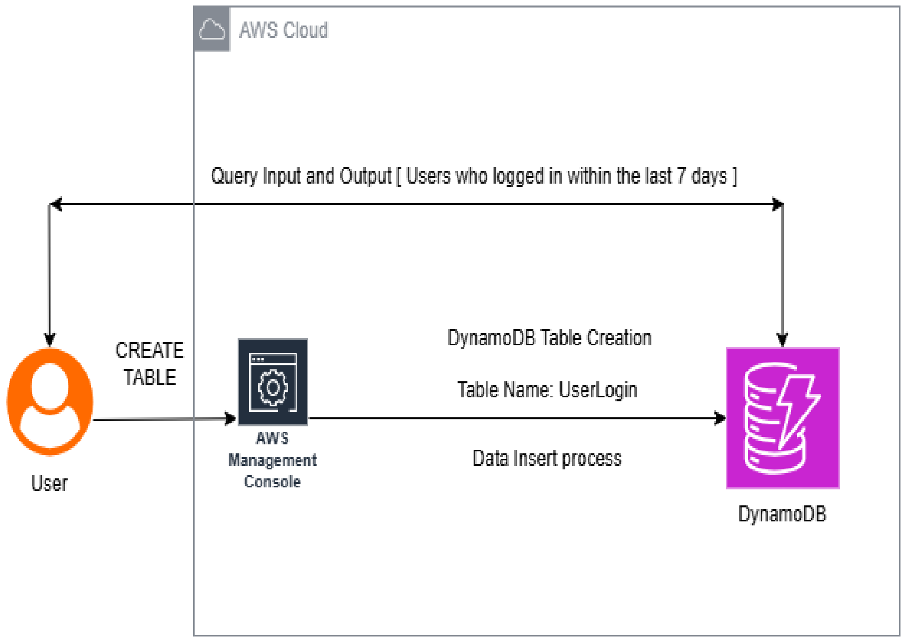
- *Query the table for users who logged in within the last 7 days.*



```
C:\Users\prath>python -c "import time; print(int(time.time()) - (7*24*60*60))"
1761337219

C:\Users\prath>aws dynamodb scan --table-name UserLogins --filter-expression "Timestamp >= :t" --expression-attribute-values "{\":t\":{\"N\":\"1761337219\"}}" --query "Items[*].[UserID.S, Name.S, LastLogin.S]"
 --output text

An error occurred (ValidationException) when calling the Scan operation: Invalid FilterExpression: Attribute name is a reserved keyword; reserved keyword: Timestamp

C:\Users\prath>aws dynamodb scan --table-name UserLogins --filter-expression "#ts >= :t" --expression-attribute-names "{\"#ts\":\"Timestamp\"}" --expression-attribute-values "{\":t\":{\"N\":\"1761337219\"}}" -
-query "Items[*].[UserID.S, Name.S, LastLogin.S]"
[]


C:\Users\prath>aws dynamodb scan --table-name UserLogins --filter-expression "#ts >= :t" --expression-attribute-names "{\"#ts\":\"Timestamp\"}" --expression-attribute-values "{\":t\":{\"N\":\"1729712451\"}}" -
-query "Items[*].[UserID.S, Name.S, LastLogin.S]" --output text
U002    Bob Smith       2025-10-29T18:30:00Z
U001    Alice Johnson   2025-10-30T10:00:00Z
U003    Carol White     2025-10-28T08:45:00Z
```

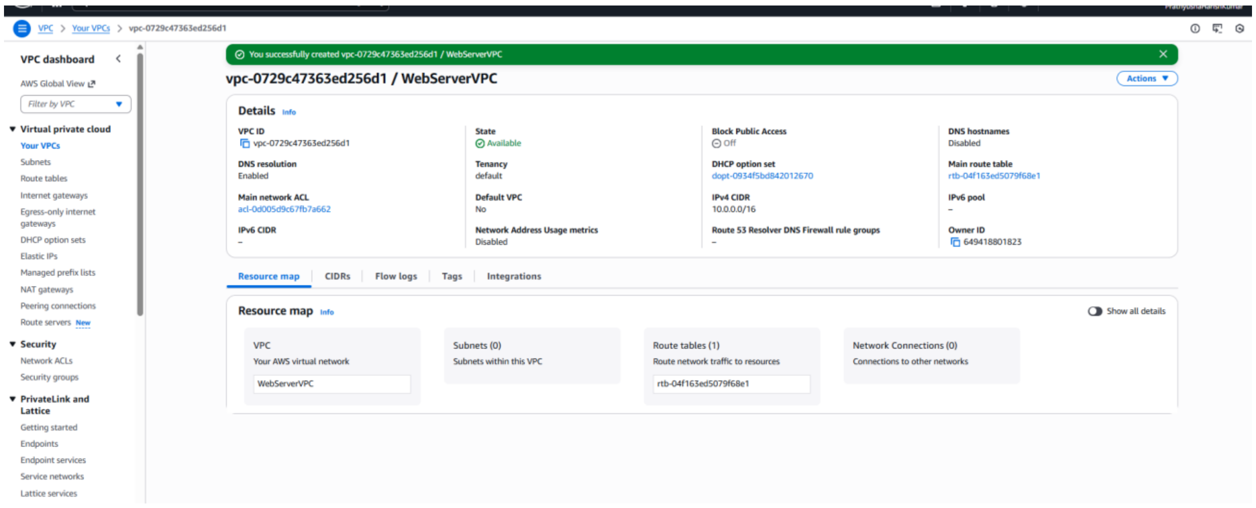
- *Deliverable: Submit screenshots of your table creation, sample data insertions, and query results.*

- ***Architectural Diagram:*** *Create an AWS architecture diagram using* ***Draw.io*** *that represents the DynamoDB interaction.*
    - *Deliverable: Submit a Draw.io, PNG, or PDF version of your architecture diagram.*



Part 2: Web Server Deployment and Automation

Task 1: Web Server Deployment on EC2 (10 points)



VPC > Your VPCs > vpc-0729c47363ed256d1

You successfully created vpc-0729c47363ed256d1 / WebServerVPC

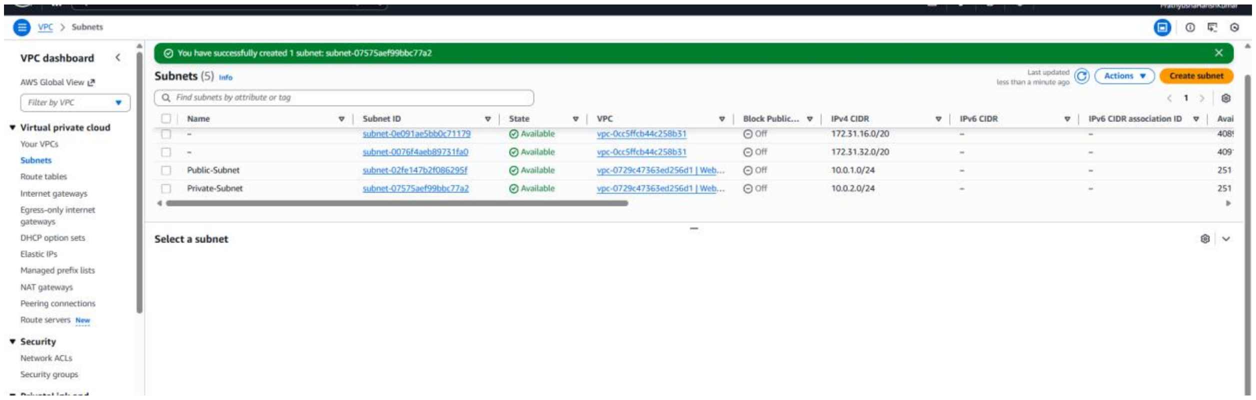
vpc-0729c47363ed256d1 / WebServerVPC

Details

VPC ID: vpc-0729c47363ed256d1	State: Available	Block Public Access: Off	DNS hostnames: Disabled
DNS resolution: Enabled	Tenancy: default	DHCP option set: dopt-0934f5bd842012670	Main route table: rtb-04f163ed5079f68e1
Main network ACL: acl-0d005d9c67fb7a662	Default VPC: No	IPv4 CIDR: 10.0.0.0/16	IPv6 pool: –
IPv6 CIDR: –	Network Address Usage metrics: Disabled	Route 53 Resolver DNS Firewall rule groups: –	Owner ID: 649418801823

Resource map

VPC	Subnets (0)	Route tables (1)	Network Connections (0)
Your AWS virtual network	Subnets within this VPC	Route network traffic to resources	Connections to other networks
WebServerVPC		rtb-04f163ed5079f68e1	



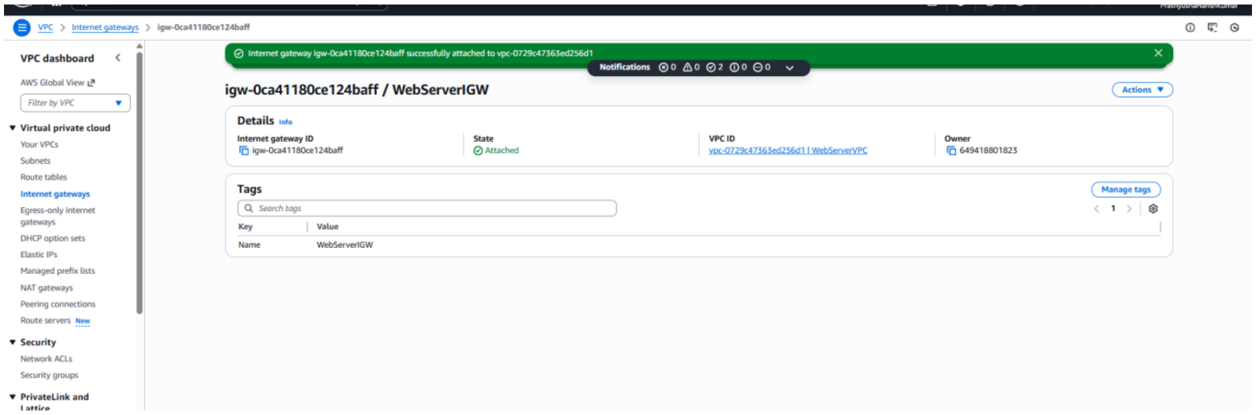
VPC > Subnets

You have successfully created 1 subnet: subnet-07575aef99bbc77a2

Subnets (5)

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Avai
–	subnet-0e091ae5bb0c71179	Available	vpc-0cc5ffcb44c258b31	Off	172.31.16.0/20	–	–	408
–	subnet-0076f4aeb89731fa0	Available	vpc-0cc5ffcb44c258b31	Off	172.31.32.0/20	–	–	409
Public-Subnet	subnet-02fe147b2f086295f	Available	vpc-0729c47363ed256d1 | Web...	Off	10.0.1.0/24	–	–	251
Private-Subnet	subnet-07575aef99bbc77a2	Available	vpc-0729c47363ed256d1 | Web...	Off	10.0.2.0/24	–	–	251

Select a subnet



VPC > Internet gateways > igw-0ca41180ce124baff

Internet gateway igw-0ca41180ce124baff successfully attached to vpc-0729c47363ed256d1

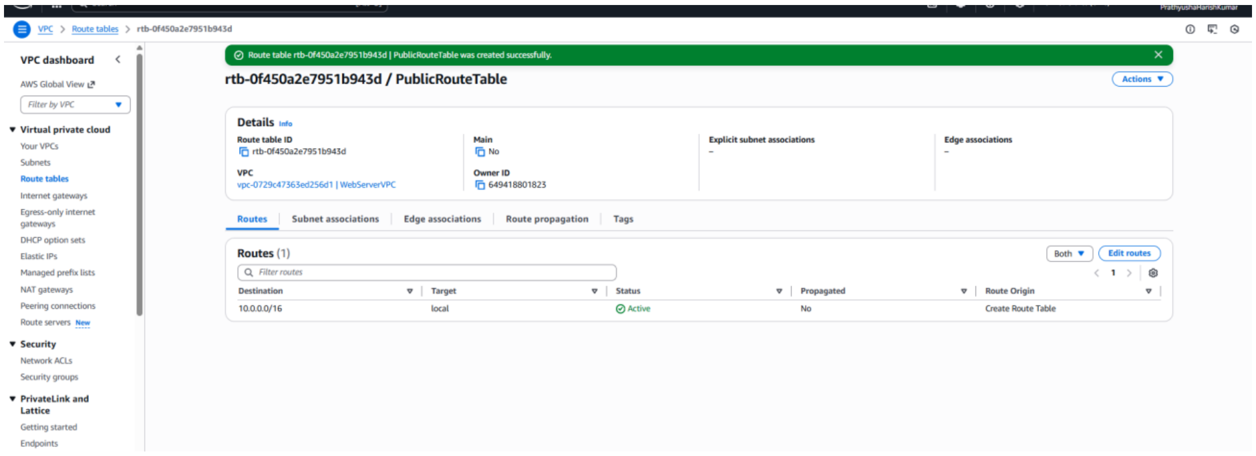
igw-0ca41180ce124baff / WebServerIGW

Details

Internet gateway ID	State	VPC ID	Owner
igw-0ca41180ce124baff	Attached	vpc-0729c47363ed256d1 | WebServerVPC	649418801823

Tags

Key	Value
Name	WebServerIGW



VPC > Route tables > rtb-0f450a2e7951b943d

Route table rtb-0f450a2e7951b943d | PublicRouteTable was created successfully.

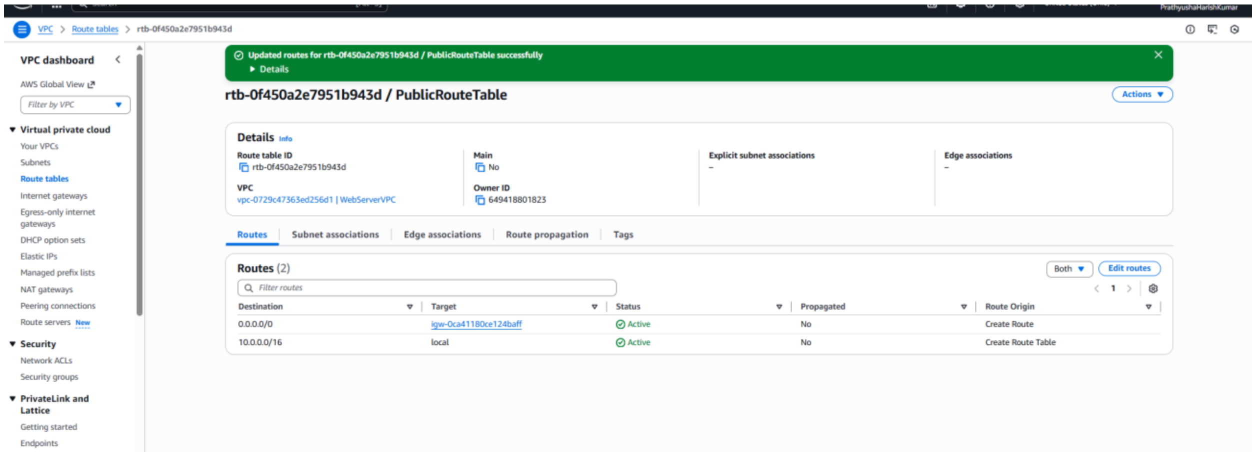
rtb-0f450a2e7951b943d / PublicRouteTable

Details

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0f450a2e7951b943d	No	–	–
VPC	Owner ID		
vpc-0729c47363ed256d1 | WebServerVPC	649418801823		

Routes (1)

Destination	Target	Status	Propagated	Route Origin
10.0.0.0/16	local	Active	No	Create Route Table



VPC > Route tables > rtb-0f450a2e7951b943d

Updated routes for rtb-0f450a2e7951b943d / PublicRouteTable successfully

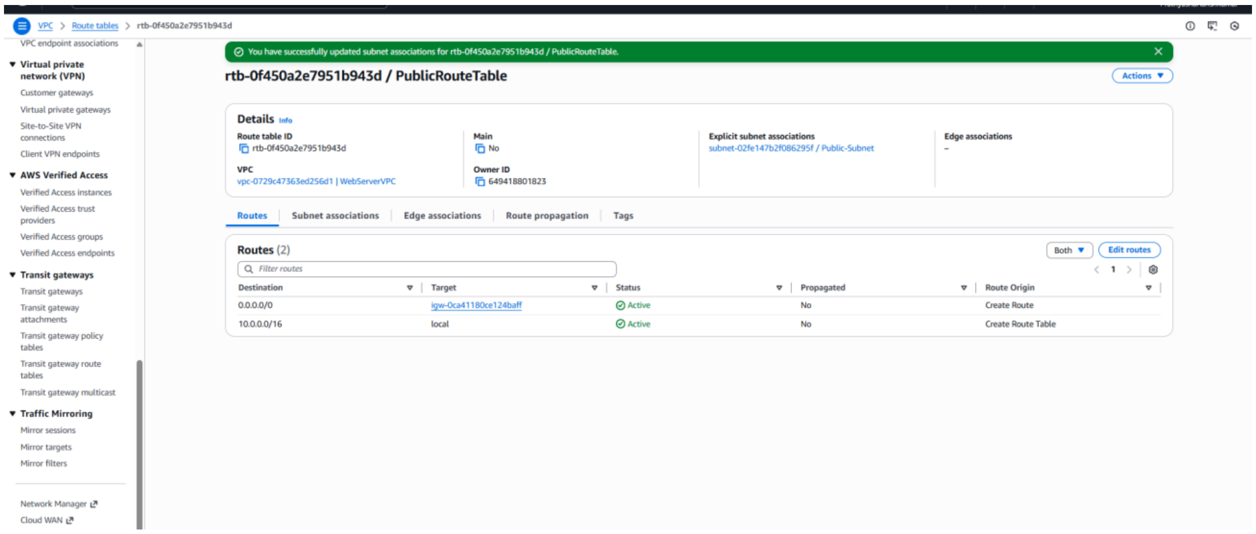
rtb-0f450a2e7951b943d / PublicRouteTable

Details

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0f450a2e7951b943d	No	–	–
VPC	Owner ID		
vpc-0729c47363ed256d1 | WebServerVPC	649418801823		

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0ca41180ce124baff	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table



VPC > Route tables > rtb-0f450a2e7951b943d

You have successfully updated subnet associations for rtb-0f450a2e7951b943d / PublicRouteTable.

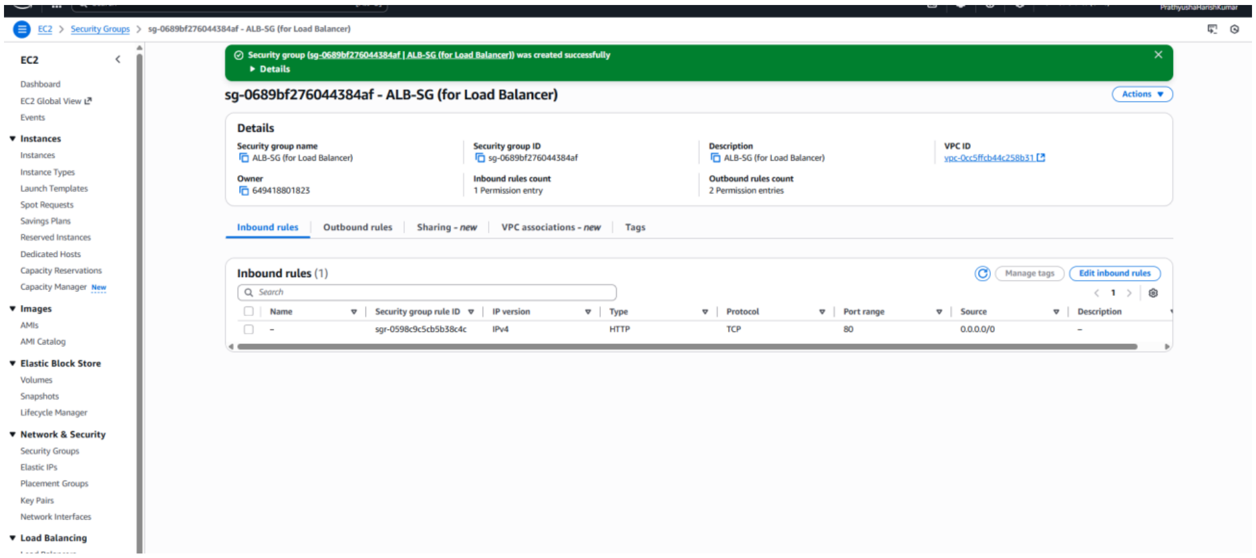
rtb-0f450a2e7951b943d / PublicRouteTable

Details

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0f450a2e7951b943d	No	subnet-02fe147b2f086295f / Public-Subnet	–
VPC	Owner ID		
vpc-0729c47363ed256d1 | WebServerVPC	649418801823		

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0ca41180ce124baff	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table


EC2 > Security Groups > sg-0689bf276044384af - ALB-SG (for Load Balancer)

EC2

Dashboard
EC2 Global View
Events

Instances
- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations
- Capacity Manager New

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

Security group (sg-0689bf276044384af | ALB-SG (for Load Balancer)) was created successfully

Details

sg-0689bf276044384af - ALB-SG (for Load Balancer)

Actions

Details

Security group name	Security group ID	Description	VPC ID
ALB-SG (for Load Balancer)	sg-0689bf276044384af	ALB-SG (for Load Balancer)	vpc-0cc5ffcb44c258b31
Owner	**Inbound rules count**	**Outbound rules count**	
649418801823	1 Permission entry	2 Permission entries	

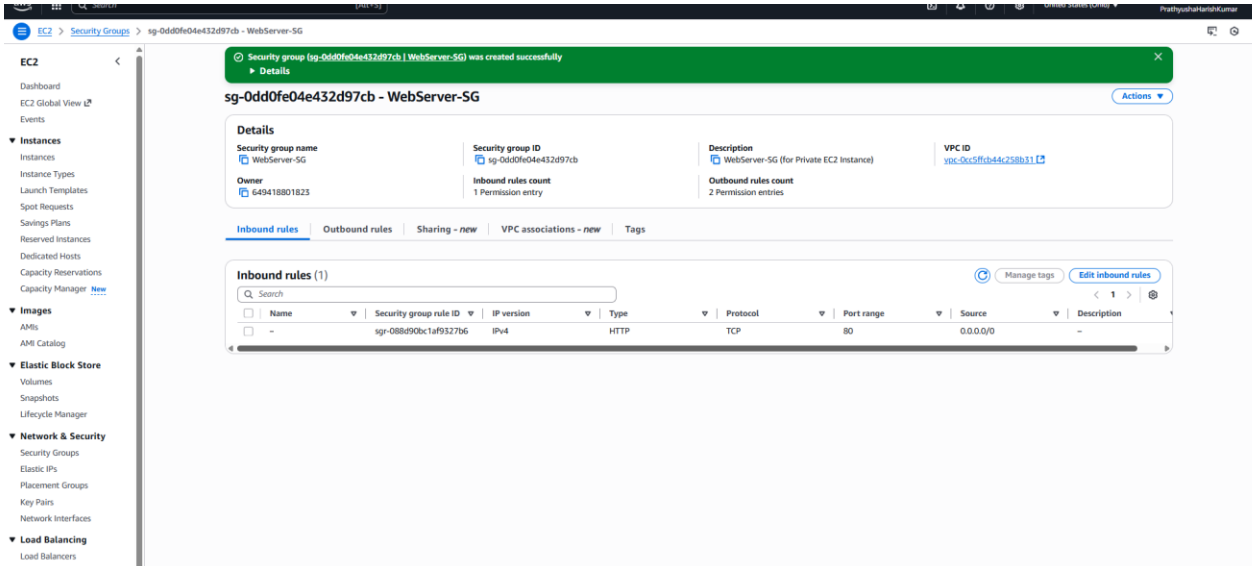
Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (1)

Manage tags | Edit inbound rules

Search

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0598c9c5cb5b38c4c	IPv4	HTTP	TCP	80	0.0.0.0/0	–


EC2 > Security Groups > sg-0dd0fe04e432d97cb - WebServer-SG

EC2

Dashboard
EC2 Global View
Events

Instances
- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations
- Capacity Manager New

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

Security group (sg-0dd0fe04e432d97cb | WebServer-SG) was created successfully

Details

sg-0dd0fe04e432d97cb - WebServer-SG

Actions

Details

Security group name	Security group ID	Description	VPC ID
WebServer-SG	sg-0dd0fe04e432d97cb	WebServer-SG (for Private EC2 Instance)	vpc-0cc5ffcb44c258b31
Owner	**Inbound rules count**	**Outbound rules count**	
649418801823	1 Permission entry	2 Permission entries	

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

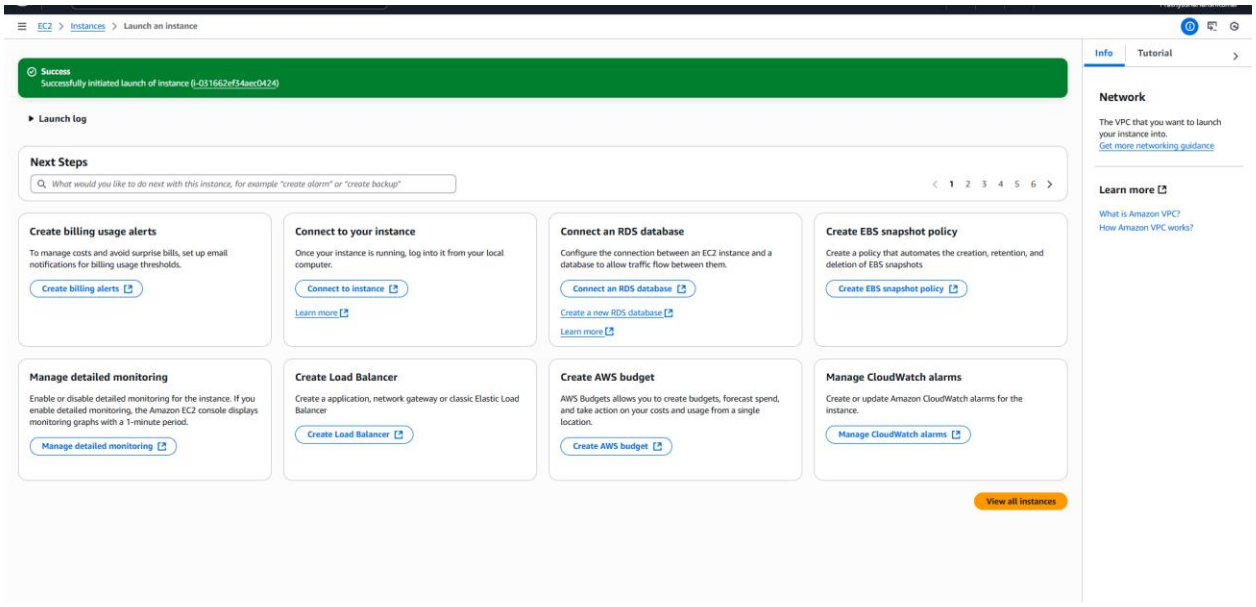
Inbound rules (1)

Manage tags | Edit inbound rules

Search

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-088d90bc1af9327b6	IPv4	HTTP	TCP	80	0.0.0.0/0	–

- *Launch an EC2 instance in a **private subnet**.*



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-031662ef34aec0424)

▸ Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create backup"

Create billing usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds.

Connect to your instance
Once your instance is running, log into it from your local computer.

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots

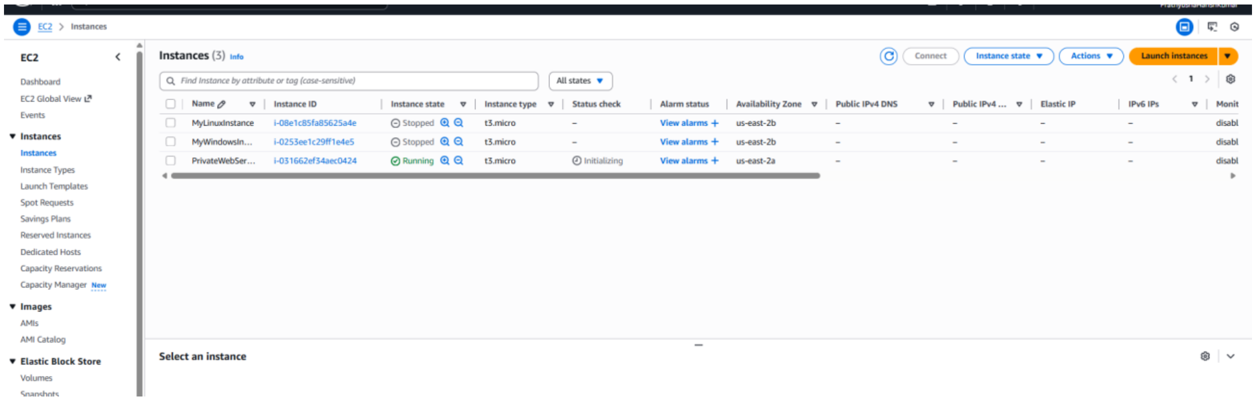
Manage detailed monitoring
Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.

Create Load Balancer
Create a application, network gateway or classic Elastic Load Balancer

Create AWS budget
AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.

Manage CloudWatch alarms
Create or update Amazon CloudWatch alarms for the instance.

Network
The VPC that you want to launch your instance into.



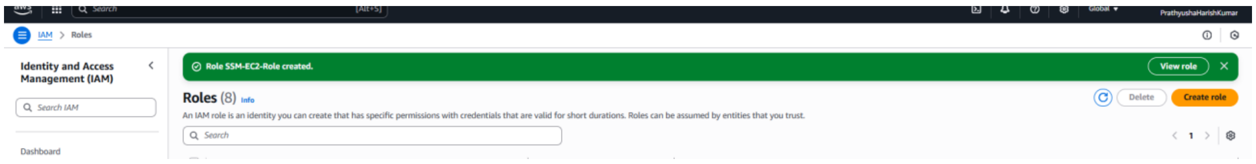
EC2 > Instances

Instances (3)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monit
	MyLinuxInstance	i-08e1c85fa85625a4e	Stopped	t3.micro	–	View alarms +	us-east-2b	–	–	–	–	disabl
	MyWindowsIn...	i-0253ee1c29ff1e4e5	Stopped	t3.micro	–	View alarms +	us-east-2b	–	–	–	–	disabl
	PrivateWebSer...	i-031662ef34aec0424	Running	t3.micro	Initializing	View alarms +	us-east-2a	–	–	–	–	disabl

Select an instance

- *Install and configure an* ***Apache or Nginx web server****.*

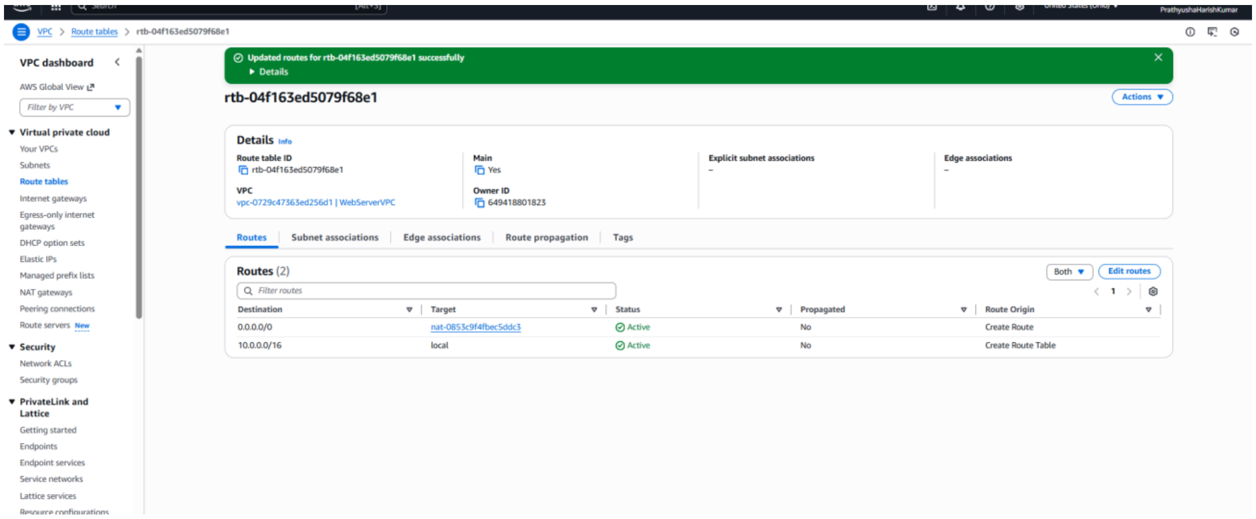


IAM > Roles

Role SSM-EC2-Role created.

Roles (8)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.



VPC > Route tables > rtb-04f163ed5079f68e1

Updated routes for rtb-04f163ed5079f68e1 successfully

rtb-04f163ed5079f68e1

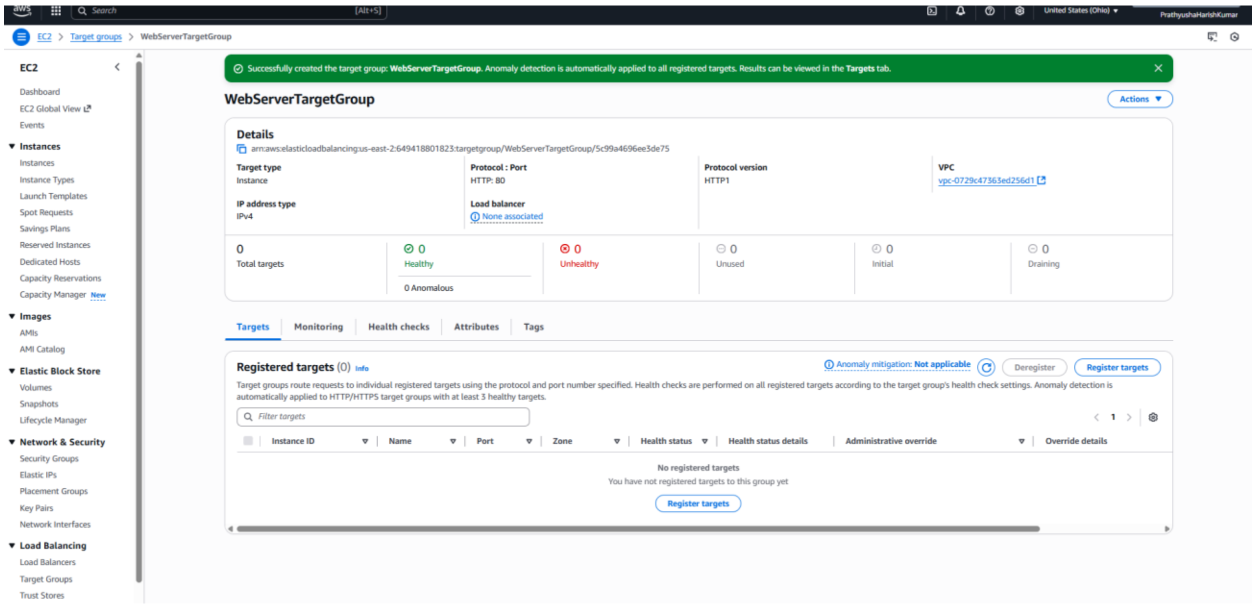
Details

Route table ID rtb-04f163ed5079f68e1	**Main** Yes	**Explicit subnet associations** –	**Edge associations** –
VPC vpc-0729c47363ed256d1 | WebServerVPC	**Owner ID** 649418801823		

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	nat-0853c9f4fbec5ddc3	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

- *Ensure the web server is accessible only via an Application Load Balancer (ALB) from the public subnet.*



EC2 > Target groups > WebServerTargetGroup

Successfully created the target group: **WebServerTargetGroup**. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the **Targets** tab.

WebServerTargetGroup

Details

arn:aws:elasticloadbalancing:us-east-2:649418801823:targetgroup/WebServerTargetGroup/5c99a4696ee3de75

Target type Instance	**Protocol : Port** HTTP: 80	**Protocol version** HTTP1	**VPC** vpc-0729c47363ed256d1
IP address type IPv4	**Load balancer** None associated		

0 Total targets	0 Healthy (0 Anomalous)	0 Unhealthy	0 Unused	0 Initial	0 Draining

Registered targets (0)

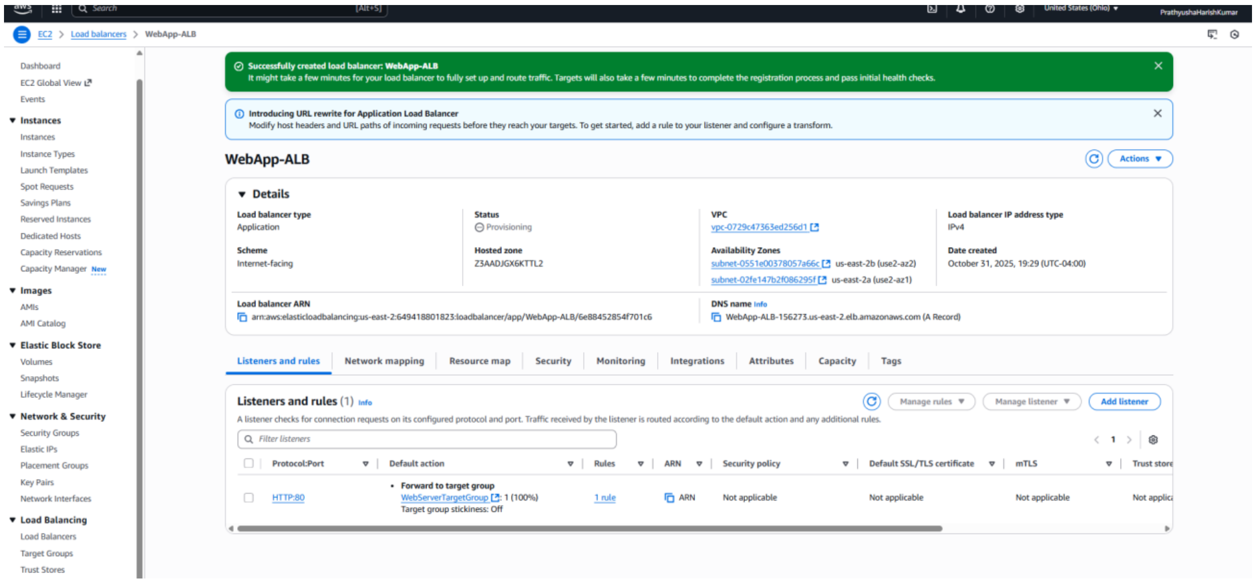
Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Administrative override	Override details

No registered targets

You have not registered targets to this group yet



Successfully created load balancer: WebApp-ALB
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

Introducing URL rewrite for Application Load Balancer
Modify host headers and URL paths of incoming requests before they reach your targets. To get started, add a rule to your listener and configure a transform.

WebApp-ALB

Details

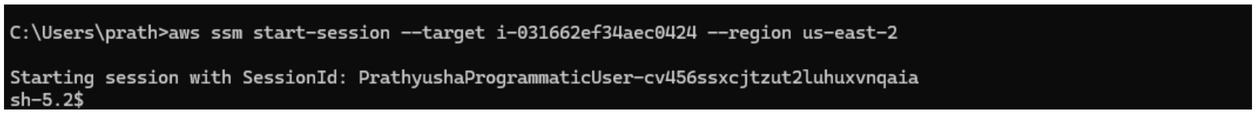
Load balancer type Application	**Status** Provisioning	**VPC** vpc-0729c47363ed256d1	**Load balancer IP address type** IPv4
Scheme Internet-facing	**Hosted zone** Z3AADJGX6KTTL2	**Availability Zones** subnet-0551e00378057a66c us-east-2b (use2-az2); subnet-02fe147b2f086295f us-east-2a (use2-az1)	**Date created** October 31, 2025, 19:29 (UTC-04:00)

Load balancer ARN arn:aws:elasticloadbalancing:us-east-2:649418801823:loadbalancer/app/WebApp-ALB/6e88452854f701c6

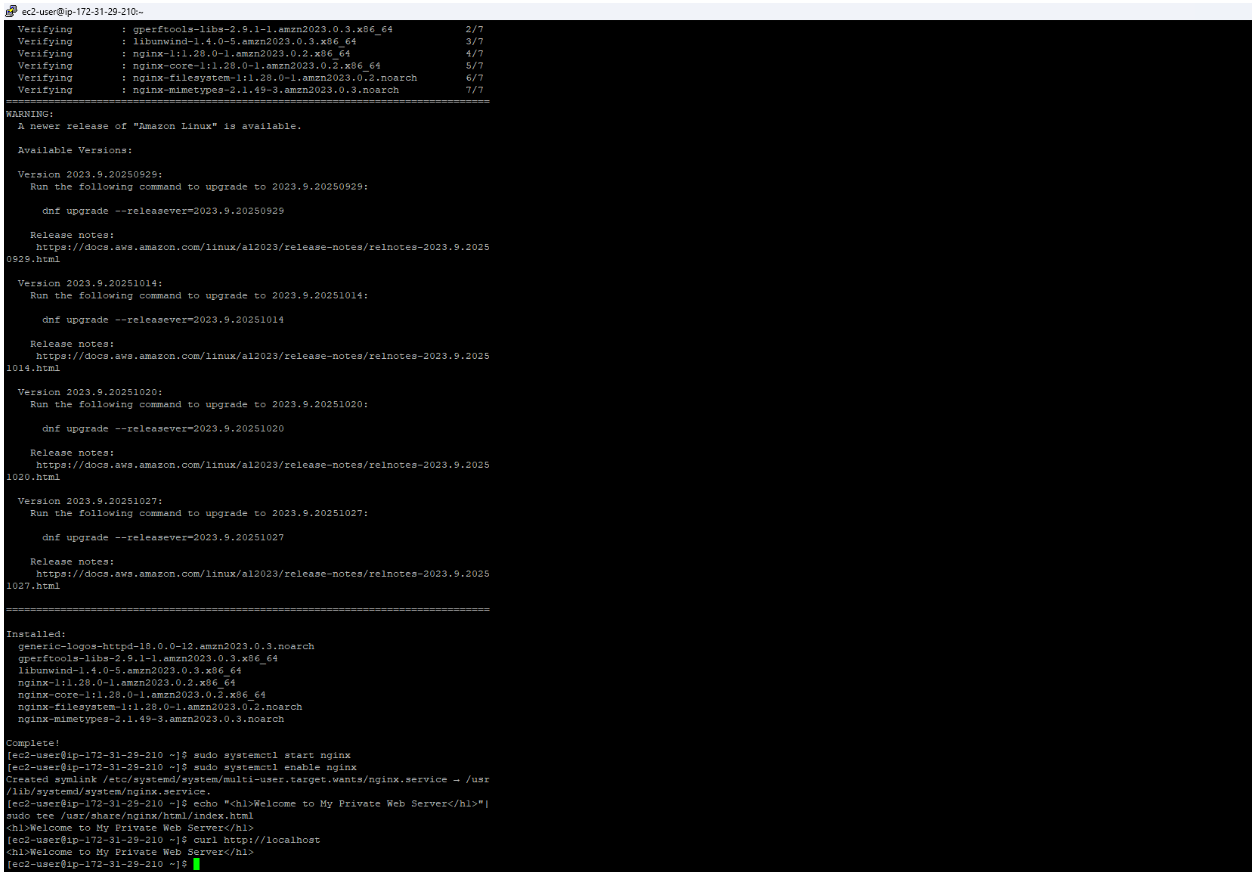
DNS name WebApp-ALB-156273.us-east-2.elb.amazonaws.com (A Record)

Listeners and rules (1)
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:80	• Forward to target group WebServerTargetGroup: 1 (100%) Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applica



```
C:\Users\prath>aws ssm start-session --target i-031662ef34aec0424 --region us-east-2

Starting session with SessionId: PrathyushaProgrammaticUser-cv456ssxcjtzut2luhuxvnqaia
sh-5.2$
```



ec2-user@ip-172-31-29-210:~

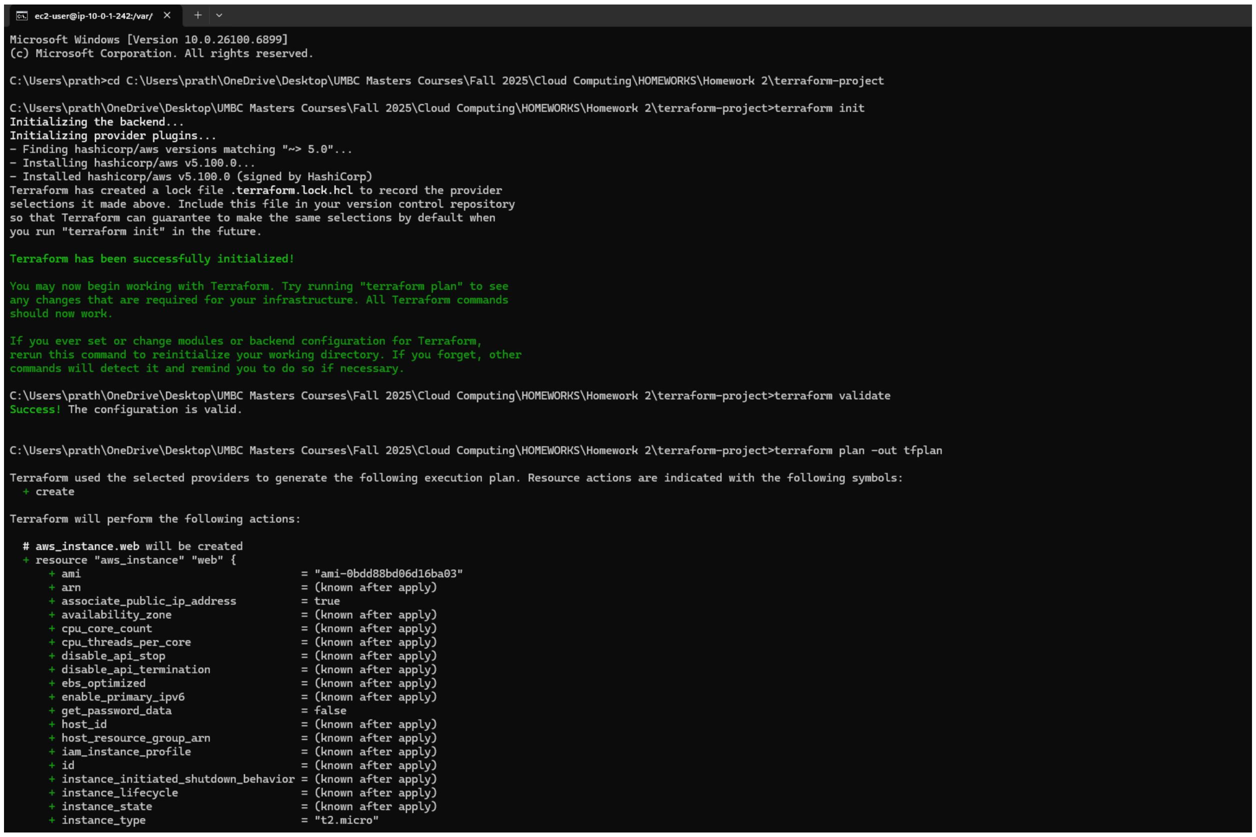
```
  Verifying        : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64            2/7
  Verifying        : libunwind-1.4.0-5.amzn2023.0.3.x86_64                  3/7
  Verifying        : nginx-1:1.28.0-1.amzn2023.0.2.x86_64                   4/7
  Verifying        : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64              5/7
  Verifying        : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch        6/7
  Verifying        : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch           7/7
================================================================================
WARNING:
  A newer release of "Amazon Linux" is available.

  Available Versions:

  Version 2023.9.20250929:
    Run the following command to upgrade to 2023.9.20250929:

      dnf upgrade --releasever=2023.9.20250929

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.9.2025
0929.html

  Version 2023.9.20251014:
    Run the following command to upgrade to 2023.9.20251014:

      dnf upgrade --releasever=2023.9.20251014

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.9.2025
1014.html

  Version 2023.9.20251020:
    Run the following command to upgrade to 2023.9.20251020:

      dnf upgrade --releasever=2023.9.20251020

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.9.2025
1020.html

  Version 2023.9.20251027:
    Run the following command to upgrade to 2023.9.20251027:

      dnf upgrade --releasever=2023.9.20251027

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.9.2025
1027.html

================================================================================

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  libunwind-1.4.0-5.amzn2023.0.3.x86_64
  nginx-1:1.28.0-1.amzn2023.0.2.x86_64
  nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64
  nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[ec2-user@ip-172-31-29-210 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-29-210 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr
/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-29-210 ~]$ echo "<h1>Welcome to My Private Web Server</h1>"|
sudo tee /usr/share/nginx/html/index.html
<h1>Welcome to My Private Web Server</h1>
[ec2-user@ip-172-31-29-210 ~]$ curl http://localhost
<h1>Welcome to My Private Web Server</h1>
[ec2-user@ip-172-31-29-210 ~]$
```

- *Document your process and include a screenshot of the working web server.*

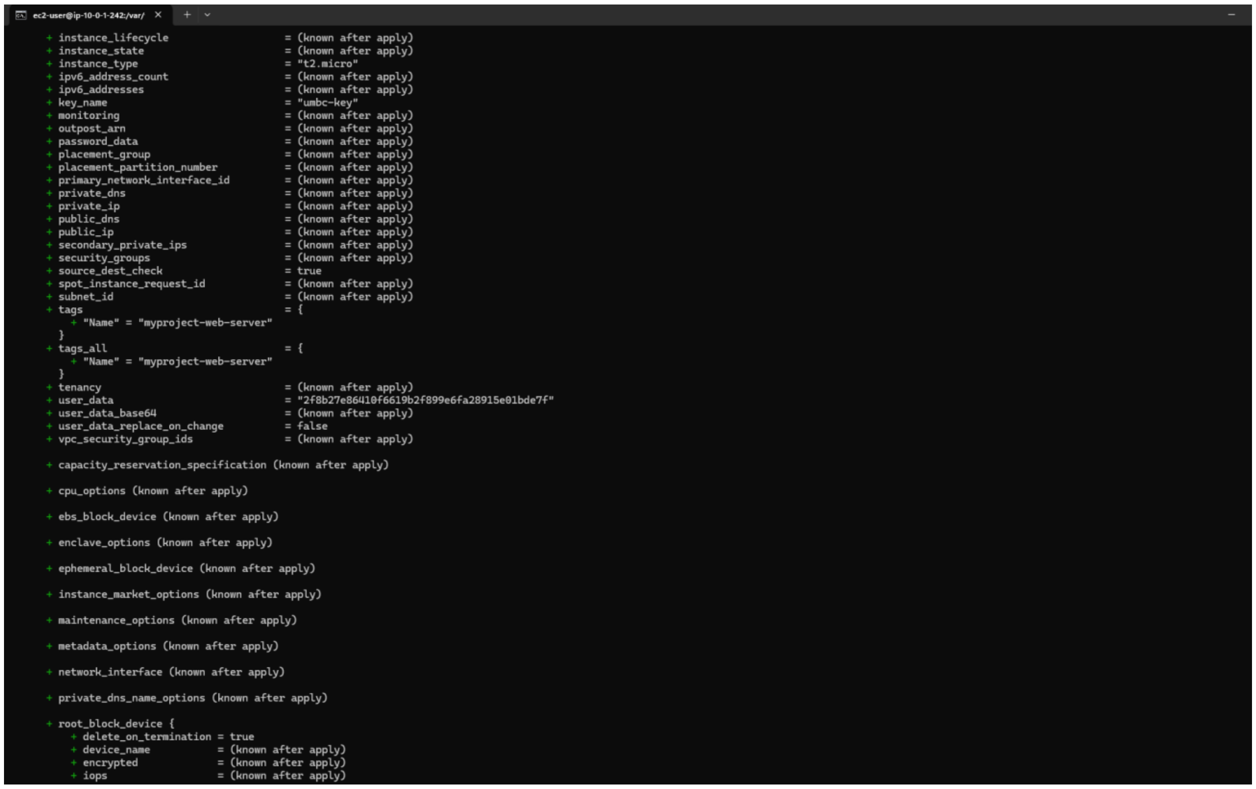
- *Deliverable: A brief document describing your setup and evidence of a successful deployment.*

Task 2: Automating Deployment with Terraform (10 points)

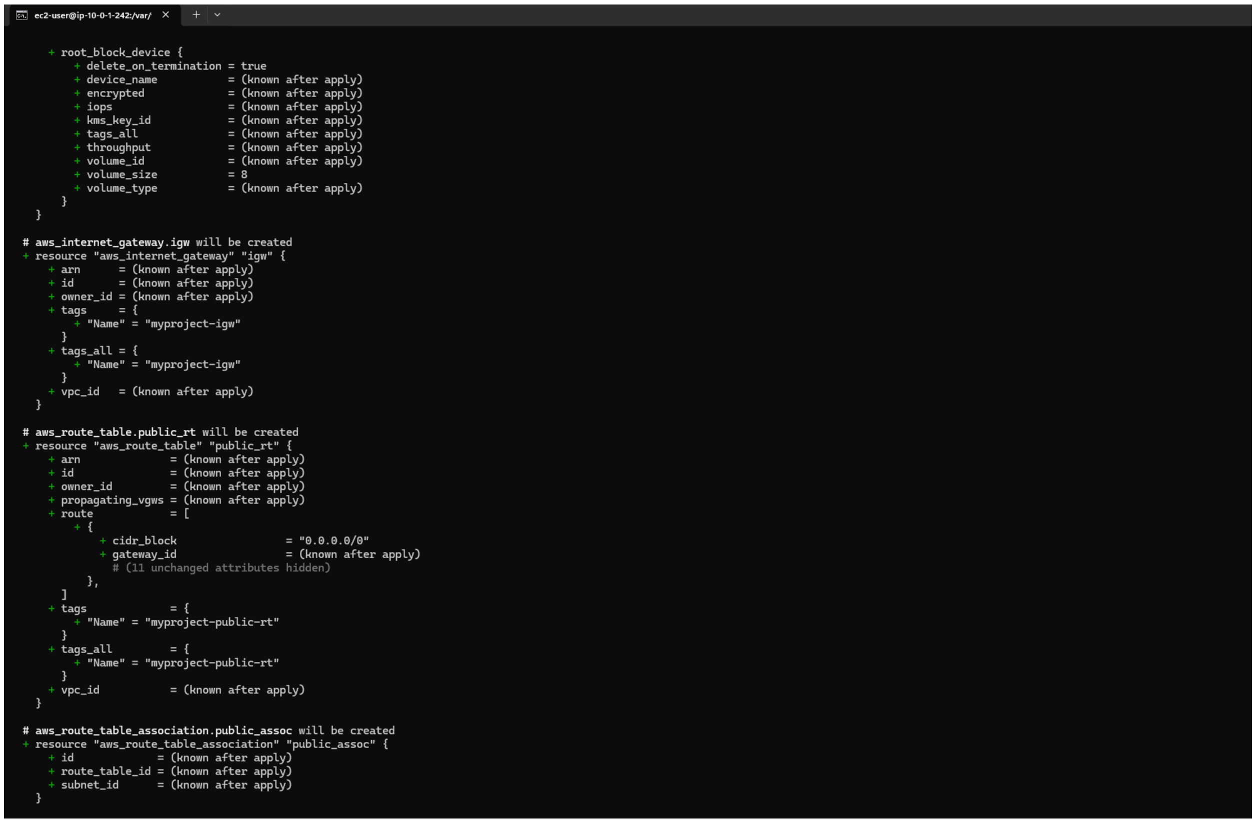
- *Use **Terraform** to create the following AWS resources:*
    - *A **VPC** with public and private subnets.*
    - *An EC2 instance configured as a **web server** in the private subnet.*
    - *An **S3 bucket** for storing static web files.*
    - *A **DynamoDB table** for storing user login details.*



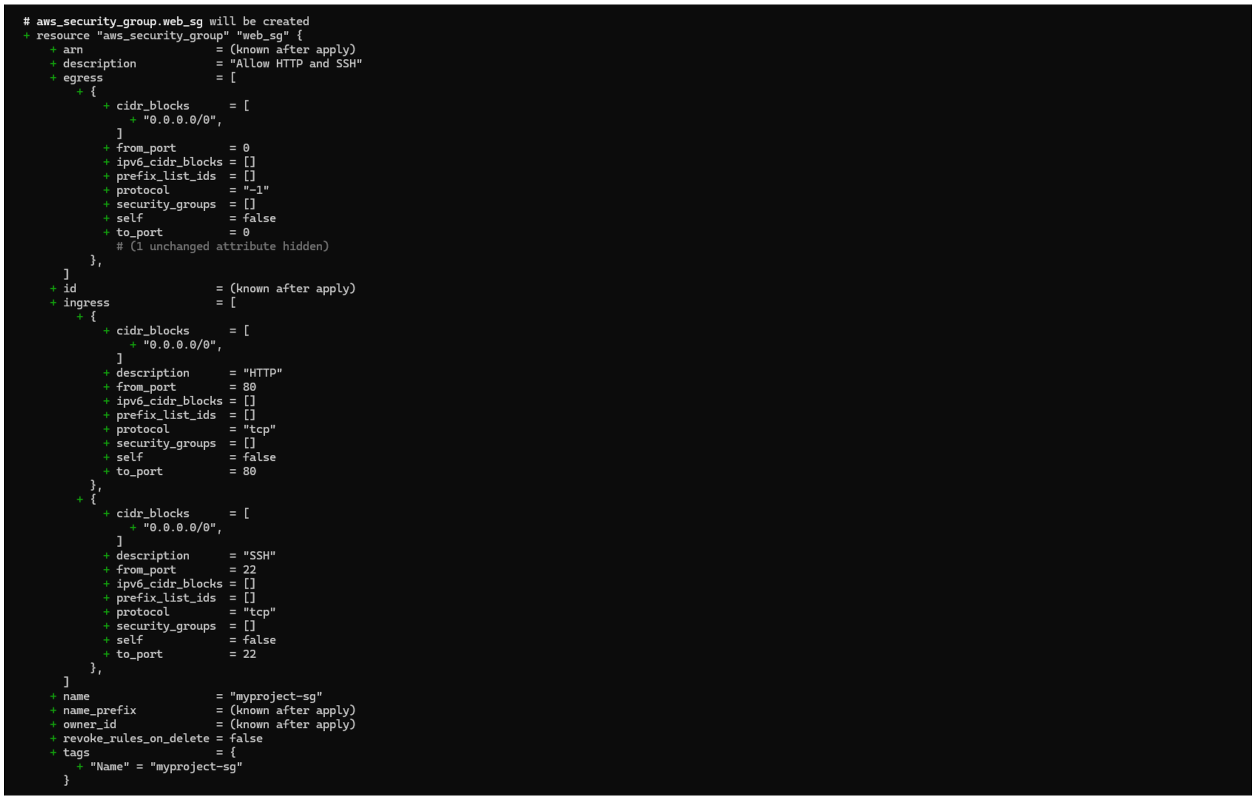
```
Microsoft Windows [Version 10.0.26100.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\prath>cd C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project

C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>terraform validate
Success! The configuration is valid.


C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>terraform plan -out tfplan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.web will be created
  + resource "aws_instance" "web" {
      + ami                                  = "ami-0bdd88bd06d16ba03"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```



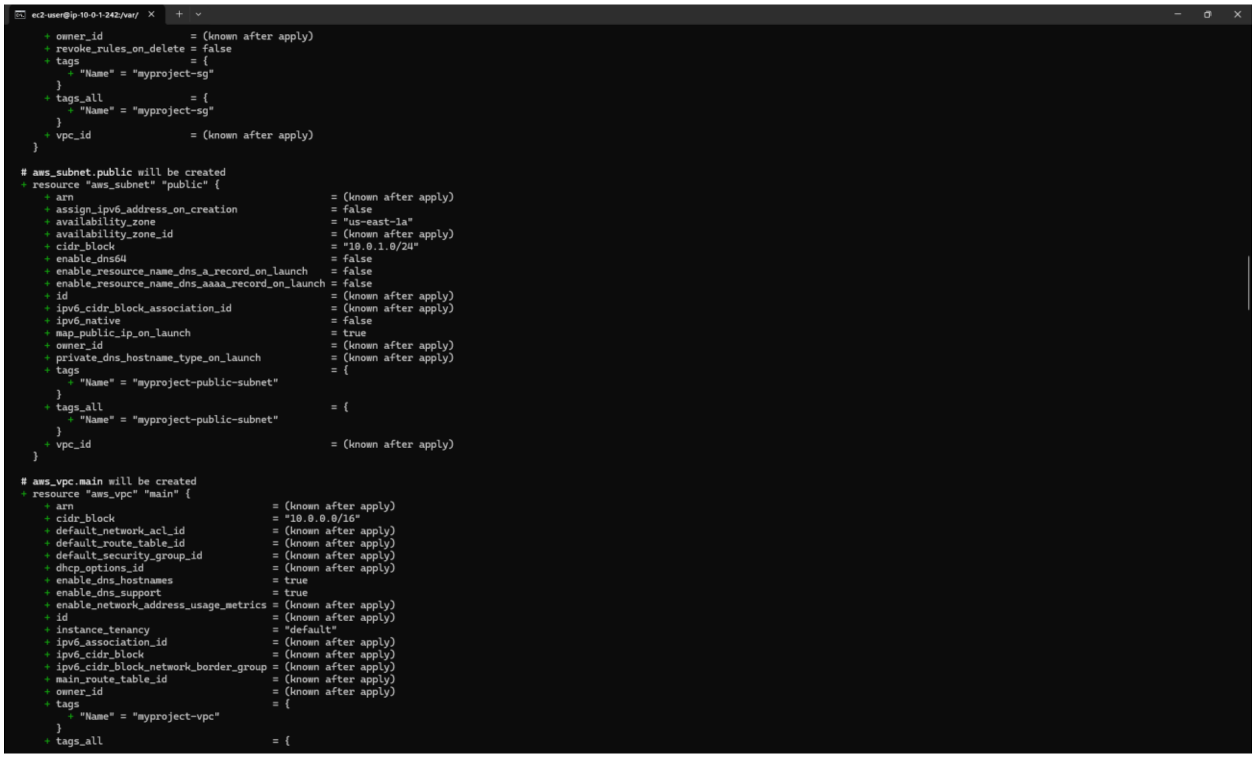
```
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "umbc-key"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "myproject-web-server"
        }
      + tags_all                             = {
          + "Name" = "myproject-web-server"
        }
      + tenancy                              = (known after apply)
      + user_data                            = "2f8b27e86410f6619b2f899e6fa28915e01bde7f"
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device {
          + delete_on_termination = true
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
```



```
      + root_block_device {
          + delete_on_termination = true
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags_all              = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = 8
          + volume_type           = (known after apply)
        }
    }

  # aws_internet_gateway.igw will be created
  + resource "aws_internet_gateway" "igw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "myproject-igw"
        }
      + tags_all = {
          + "Name" = "myproject-igw"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route_table.public_rt will be created
  + resource "aws_route_table" "public_rt" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + gateway_id                 = (known after apply)
                # (11 unchanged attributes hidden)
            },
        ]
      + tags             = {
          + "Name" = "myproject-public-rt"
        }
      + tags_all         = {
          + "Name" = "myproject-public-rt"
        }
      + vpc_id           = (known after apply)
    }

  # aws_route_table_association.public_assoc will be created
  + resource "aws_route_table_association" "public_assoc" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }
```

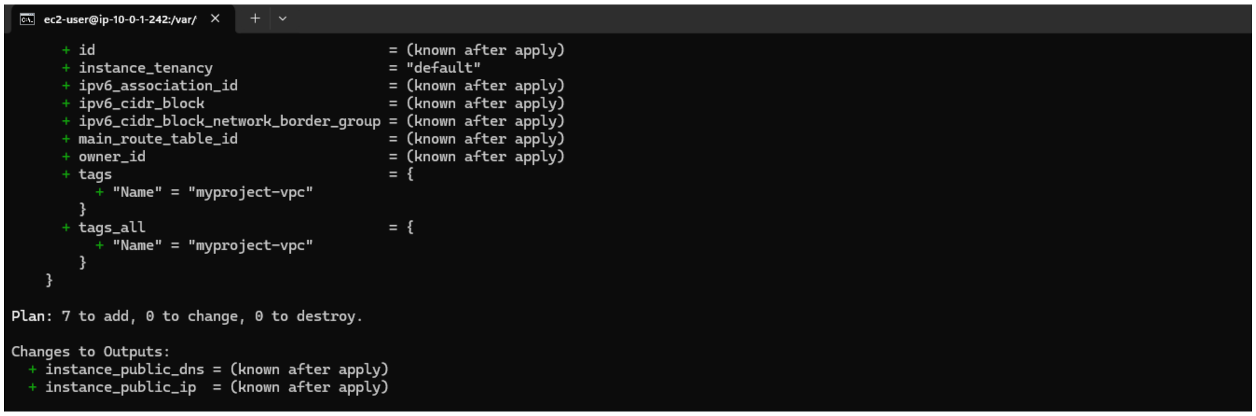


```
# aws_security_group.web_sg will be created
+ resource "aws_security_group" "web_sg" {
    + arn                    = (known after apply)
    + description            = "Allow HTTP and SSH"
    + egress                 = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
            + to_port          = 0
              # (1 unchanged attribute hidden)
          },
      ]
    + id                     = (known after apply)
    + ingress                = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "HTTP"
            + from_port        = 80
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = "SSH"
            + from_port        = 22
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
          },
      ]
    + name                   = "myproject-sg"
    + name_prefix            = (known after apply)
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "myproject-sg"
      }
```

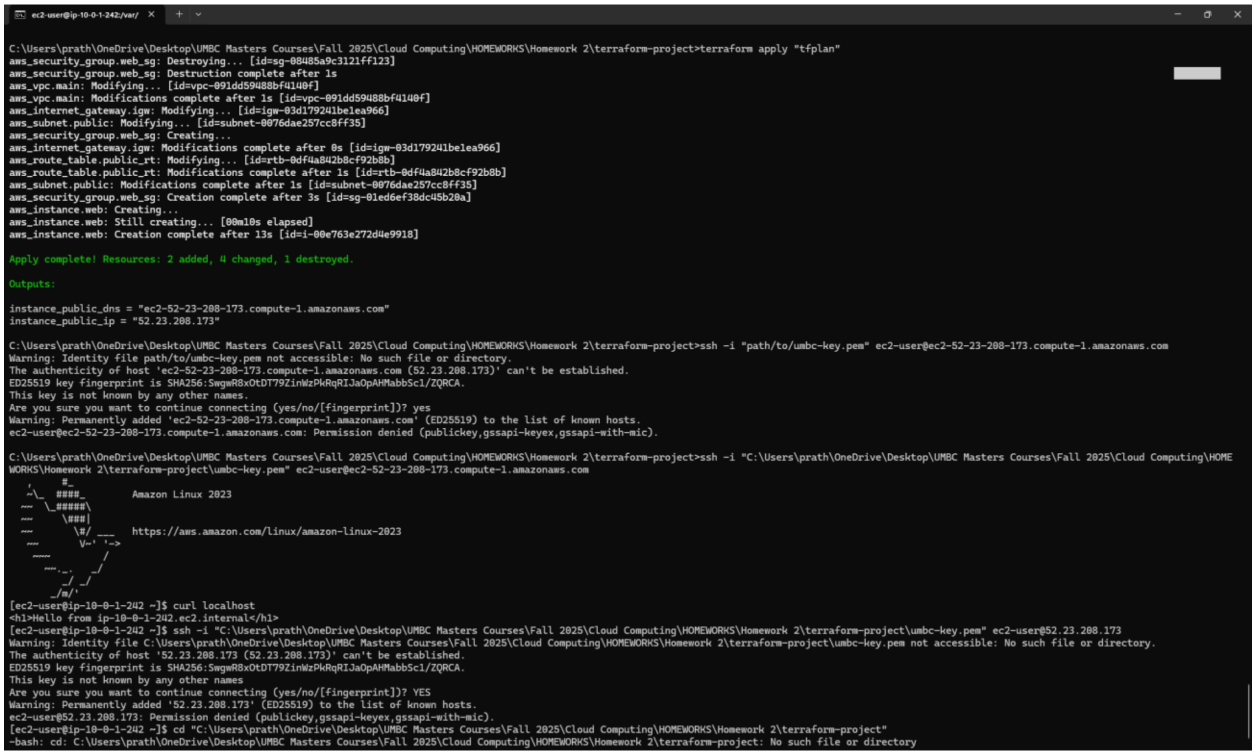


ec2-user@ip-10-0-1-242:/var/

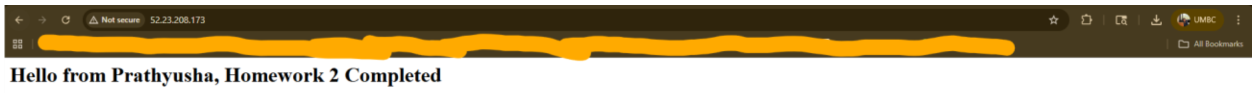
```
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "myproject-sg"
      }
    + tags_all               = {
        + "Name" = "myproject-sg"
      }
    + vpc_id                 = (known after apply)
  }

# aws_subnet.public will be created
+ resource "aws_subnet" "public" {
    + arn                                            = (known after apply)
    + assign_ipv6_address_on_creation                = false
    + availability_zone                              = "us-east-1a"
    + availability_zone_id                           = (known after apply)
    + cidr_block                                     = "10.0.1.0/24"
    + enable_dns64                                   = false
    + enable_resource_name_dns_a_record_on_launch    = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                                             = (known after apply)
    + ipv6_cidr_block_association_id                 = (known after apply)
    + ipv6_native                                    = false
    + map_public_ip_on_launch                        = true
    + owner_id                                       = (known after apply)
    + private_dns_hostname_type_on_launch            = (known after apply)
    + tags                                           = {
        + "Name" = "myproject-public-subnet"
      }
    + tags_all                                       = {
        + "Name" = "myproject-public-subnet"
      }
    + vpc_id                                         = (known after apply)
  }

# aws_vpc.main will be created
+ resource "aws_vpc" "main" {
    + arn                                  = (known after apply)
    + cidr_block                           = "10.0.0.0/16"
    + default_network_acl_id               = (known after apply)
    + default_route_table_id               = (known after apply)
    + default_security_group_id            = (known after apply)
    + dhcp_options_id                      = (known after apply)
    + enable_dns_hostnames                 = true
    + enable_dns_support                   = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                                   = (known after apply)
    + instance_tenancy                     = "default"
    + ipv6_association_id                  = (known after apply)
    + ipv6_cidr_block                      = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id                  = (known after apply)
    + owner_id                             = (known after apply)
    + tags                                 = {
        + "Name" = "myproject-vpc"
      }
    + tags_all                             = {
```



```
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "myproject-vpc"
        }
      + tags_all                             = {
          + "Name" = "myproject-vpc"
        }
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance_public_dns = (known after apply)
  + instance_public_ip  = (known after apply)
```



```
C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>terraform apply "tfplan"
aws_security_group.web_sg: Destroying... [id=sg-08485a9c3121ff123]
aws_security_group.web_sg: Destruction complete after 1s
aws_vpc.main: Modifying... [id=vpc-091dd59488bf4140f]
aws_vpc.main: Modifications complete after 1s [id=vpc-091dd59488bf4140f]
aws_internet_gateway.igw: Modifying... [id=igw-03d179241be1ea966]
aws_subnet.public: Modifying... [id=subnet-0076dae257cc8ff35]
aws_security_group.web_sg: Creating...
aws_internet_gateway.igw: Modifications complete after 0s [id=igw-03d179241be1ea966]
aws_route_table.public_rt: Modifying... [id=rtb-0df4a842b8cf92b8b]
aws_route_table.public_rt: Modifications complete after 1s [id=rtb-0df4a842b8cf92b8b]
aws_subnet.public: Modifications complete after 1s [id=subnet-0076dae257cc8ff35]
aws_security_group.web_sg: Creation complete after 3s [id=sg-01ed6ef38dc45b20a]
aws_instance.web: Creating...
aws_instance.web: Still creating... [00m10s elapsed]
aws_instance.web: Creation complete after 13s [id=i-00e763e272d4e9918]

Apply complete! Resources: 2 added, 4 changed, 1 destroyed.

Outputs:

instance_public_dns = "ec2-52-23-208-173.compute-1.amazonaws.com"
instance_public_ip = "52.23.208.173"

C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>ssh -i "path/to/umbc-key.pem" ec2-user@ec2-52-23-208-173.compute-1.amazonaws.com
Warning: Identity file path/to/umbc-key.pem not accessible: No such file or directory.
The authenticity of host 'ec2-52-23-208-173.compute-1.amazonaws.com (52.23.208.173)' can't be established.
ED25519 key fingerprint is SHA256:SwgwR8xOtDT79ZinWzPkRqRIJaOpAHMabbSc1/ZQRCA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-23-208-173.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-52-23-208-173.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project>ssh -i "C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOME
WORKS\Homework 2\terraform-project\umbc-key.pem" ec2-user@ec2-52-23-208-173.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-242 ~]$ curl localhost
<h1>Hello from ip-10-0-1-242.ec2.internal</h1>
[ec2-user@ip-10-0-1-242 ~]$ ssh -i "C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project\umbc-key.pem" ec2-user@52.23.208.173
Warning: Identity file C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project\umbc-key.pem not accessible: No such file or directory.
The authenticity of host '52.23.208.173 (52.23.208.173)' can't be established.
ED25519 key fingerprint is SHA256:SwgwR8xOtDT79ZinWzPkRqRIJaOpAHMabbSc1/ZQRCA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? YES
Warning: Permanently added '52.23.208.173' (ED25519) to the list of known hosts.
ec2-user@52.23.208.173: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-1-242 ~]$ cd "C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project"
-bash: cd: C:\Users\prath\OneDrive\Desktop\UMBC Masters Courses\Fall 2025\Cloud Computing\HOMEWORKS\Homework 2\terraform-project: No such file or directory
```



[redacted]

Hello from Prathyusha, Homework 2 Completed

- *Deliverable: Submit your **Terraform configuration files** and a short report explaining your code. Write your terraform script in modules and not everything in 1 tf fine.*

Code summary:

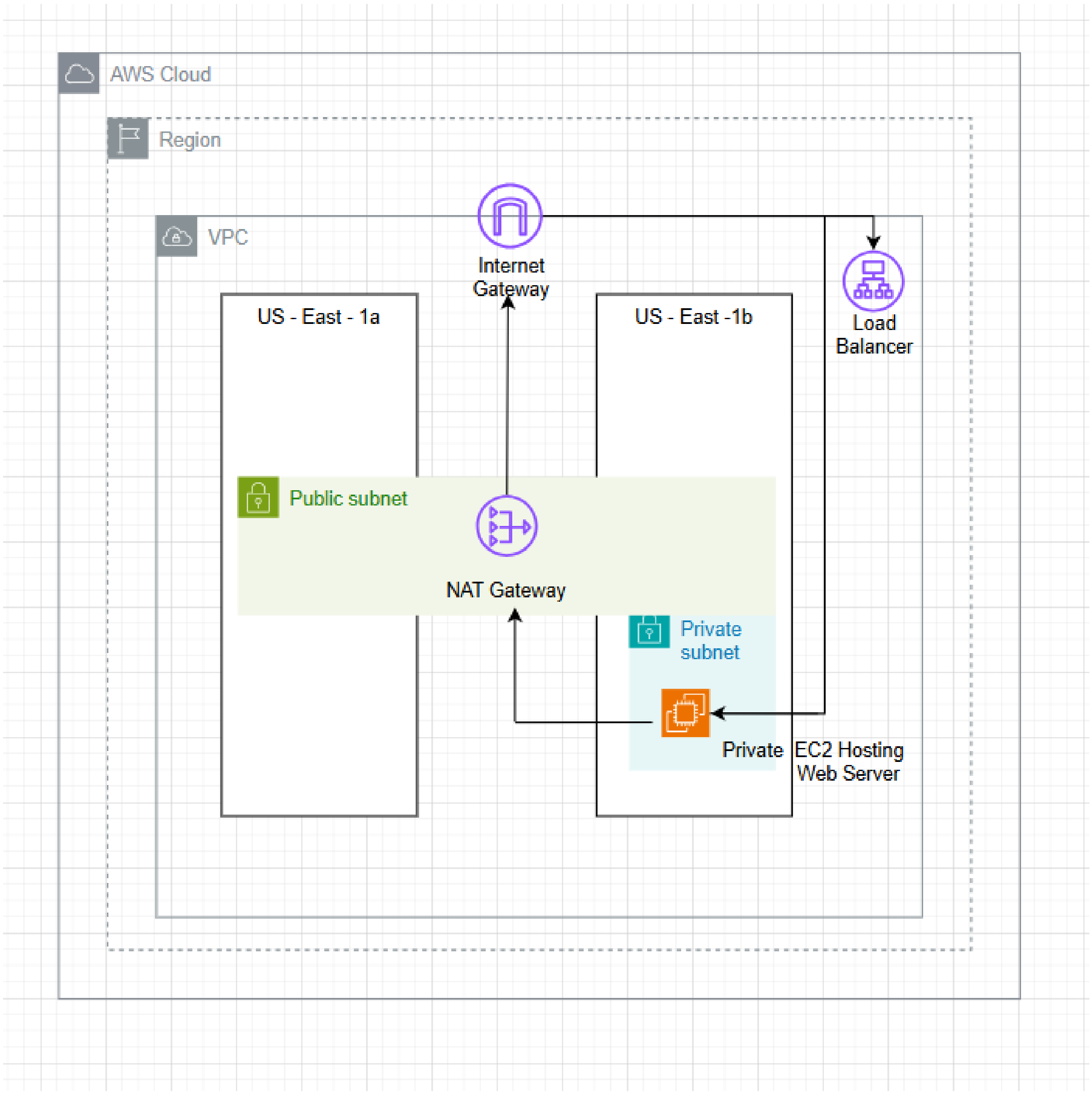
VPC: The given Terraform file automatically provisions a VPC on AWS with its subnet structure divided into a public subnet and a private subnet. The public subnet traffic is routed through the Internet Gateway which allows direct access to the web, whereas the private subnet traffic goes through the NAT Gateway which permits only secure outbound connections. The Route tables have been configured to handle the traffic in such a way that while the resources in the private subnet are isolated from the Internet, they can still access the public subnet. Therefore, the public subnet becomes the gateway for all the Internet traffic that is managed.

EC2:
This Terraform code provisioned an AWS VPC consisting of public/private subnets and, in addition, it deployed an EC2 web server in the private subnet to serve the VPC. An Internet Gateway provides access to the public subnet while the private subnet blocks traffic at its ingress. The EC2 instance is running Apache which is only accessible through SSH from your IP address (restricted) or through a load balancer (ALB) in the future. The private IP is outputted for troubleshooting purposes.

S3: In this Terraform script, an S3 bucket is created with static website hosting configuration and a bucket policy allowing public read access. A sample index.html file is included, proper permission is set, and the website URL is outputted. The bucket is prepared to serve static files (HTML, CSS, JS) with automatic index and error page support.

DynamoDB: A DynamoDB table called User Logins is created by the Terraform code with UserID as the partition (or primary) key and LoginTimestamp as the sort key. A global secondary index is there for querying by email, server-side encryption is enabled, and pay-per-request billing is the mode for auto-scaling. Outputs going to be easy reference.

- ***Architectural Diagram:*** *Create an AWS architecture diagram using* ***Draw.io*** *that represents the web server deployment and Terraform-managed resources.*
    - *Deliverable: Submit a Draw.io, PNG, or PDF version of your architecture diagram.*



-

Part 3: Written Questions

Networking & Security (6 points):

1. *What are the benefits of placing a web server in a private subnet instead of a public one?*

   Reference: https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html?utm_

   The use of a web server in a private subnet results in better security, isolation, and control of network access in your AWS environment. The AWS VPC User Guide states that a private subnet is not directly connected to an Internet Gateway. Thus, resources like EC2 instances in a private subnet are not accessible from the internet. Rather, they use secure intermediaries like a NAT Gateway or a bastion host to connect to the internet if required.

   The configuration has some advantages including:

   - Security is Enhanced: The web server is denied direct external access and so its exposure to attacks is lessened.
   - Access is Controlled: Private subnet can be reached only by the traffic from the trusted components (e.g., an Application Load Balancer in a public subnet) that has been allowed.
   - Compliance and Best Practices: AWS recommends the use of private subnets to protect resources that are sensitive and to maintain a secure network topology.
   - And Attack Surface is Reduced: No public IP is assigned and thus, attackers cannot target the instance directly.
   - Protection of Network in Layers: Security groups and network ACLs perform additional filtering at the instance and subnet levels, respectively.

   To put it simply, putting a web server in a private subnet actually follows the security measures recommended by AWS because it completely cuts off direct internet access to the server but at the same time allows controlled communication through secure routing components like NAT gateways or load balancers.

2. *Explain the role of an Application Load Balancer in AWS.*

   Reference: https://aws.amazon.com/elasticloadbalancing/application-load-balancer/?utm_

   In AWS, an Application Load Balancer (ALB) is tasked with distributing HTTP and HTTPS traffic amongst several targets like EC2 instances, containers, IP addresses, and AWS Lambda functions in an efficient manner. It is a Layer 7 (the application layer) device of the OSI model, which provides it with the capability to make smart routing decisions depending on the content of the request — for example, the URL path, hostname, query parameters, or HTTP headers.

The ALB is a perfect fit for contemporary application designs, including microservices and container-based applications, because it offers content-based routing and smooth integration with AWS ECS (Elastic Container Service) and EKS (Elastic Kubernetes Service).

Main characteristics and functions are:

- Advanced Request Routing: Diverts traffic to various targets depending on rules such as path or host-based routing, allowing for microservice-based designs.
- Security and Encryption: Reduces complexity in security by taking care of SSL/TLS termination (TLS offloading) and by guaranteeing the application of the latest encryption protocols and ciphers.
- High Availability: The system automatically shares the incoming traffic among different Availability Zones, thus increasing its fault tolerance and resilience.
- Protocol Support: The balancer accepts HTTP/2, gRPC, and WebSockets enabling new, adjustable applications.
- Integration with AWS WAF and IAM: Through the use of the AWS Web Application Firewall (WAF), it offers protection to applications against typical web threats, besides, it supports other authentication methods like OIDC or Cognito-based user authentication.
- Monitoring and Scaling: It operates together with AWS CloudWatch and Auto Scaling to always provide the best performance, even when traffic levels change dramatically.

To sum up, the Application Load Balancer is an intermediary for web traffic, it arranges and secures backend services requests smartly, and thus enhances availability, scalability, and performance.

DynamoDB & Data Management (6 points):

3. How does DynamoDB differ from traditional relational databases?

Reference:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.WhyDynamoDB.html?utm_source

DynamoDB, a NoSQL database offered by AWS, differs from traditional relational databases (RDBMS) in several key ways:

Characteristic	**Relational Database (SQL)**	**Amazon DynamoDB (NoSQL)**
Data Model	Uses structured tables with rows and columns. Requires a predefined schema. Relationships between tables are defined using foreign keys and joins.	Schemaless design. Each item in a table can have different attributes, except the primary key. Ideal for flexible and evolving data models.

Characteristic	Relational Database (SQL)	Amazon DynamoDB (NoSQL)
Query Language	Uses SQL for querying, including complex joins, aggregations, and transactions.	Provides PartiQL (SQL-compatible) for querying, but is best used with key-value access patterns. Supports SDK-based operations rather than complex joins.
Scaling	Generally vertically scaled (adding more CPU/RAM to a single machine). Horizontal scaling (sharding) is possible but complex.	Designed for horizontal scaling across distributed clusters with no upper limits on storage or throughput. You specify capacity and DynamoDB scales automatically.
Performance	Performance can degrade with large datasets or improper indexing; often requires tuning by DBAs.	Performance is predictable and optimized for high-traffic workloads. Managed service abstracts hardware and performance management.
Ideal Use Cases	Suited for structured, complex transactional systems like financial applications, where strong consistency is required.	Ideal for web-scale applications such as gaming, IoT, mobile apps, and social media, where high write/read volumes and low latency are needed.

In summary, **DynamoDB offers schema less flexibility, automatic scaling, and high performance for large-scale applications**, whereas relational databases rely on structured data models and are best suited for traditional, transactional workloads.

4. What are the benefits of using AWS IAM policies to control access to DynamoDB?

Reference: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/security-iam.html?utm_source

AWS Identity and Access Management (IAM) is an extremely competent service that lets companies to manage access to AWS resources, one of which being Amazon DynamoDB, in a centralized way. It is very important to apply IAM policies for access to DynamoDB, as it will help to create a secure, scalable, and manageable environment.

- One of the major advantages of applying IAM policies for **custom access** is the control over permissions in detail. IAM policies give an opportunity to the administrators to indicate very particularly which actions users or services are allowed to do, e.g. GetItem, PutItem, or Query, and which exact tables, indexes, or even items these actions can be performed on. This is the way of least privilege in operation, meaning that the users and applications only have access which is needed for them and thus the risk of unauthorized data access or inadvertent changes is reduced.

- Moreover, **centralized management** of permissions is one more advantage. Instead of embedding credentials in the applications or setting up temporary access patterns, IAM allows permission management and updates from one place. This would lead to continuously maintaining the security standards throughout the organization and being able to respond quickly (and easily) to changes in staff, application behavior and even security threats.
- IAM also provides improved **auditing and governance**. Every operation performed with IAM-based credentials can be traced back via AWS CloudTrail which records who did what, when and on which resources. This ability to trace back supports both internal governance and external compliance that might be required in the case of SOC, HIPAA or GDPR. It guarantees that organizations can promptly respond to security queries and thus audits or investigations take less time.
- IAM policies also support **flexible access models**. Using IAM roles, federated users (such as employees authenticated via corporate directories), or applications running on EC2 can be assigned temporary credentials to interact securely with DynamoDB. This is a best practice for securing both human and programmatic access without the need for long-term credentials.
- Finally, IAM supports resource-based policies, which enable secure cross-account access to DynamoDB resources. This is particularly useful in multi-account architectures, where a central data account might expose a DynamoDB table to multiple application accounts, all while maintaining strict control over allowed actions.

In summary, IAM policies provide secure, scalable, auditable, and flexible access control for DynamoDB, helping organizations protect their data while enabling efficient, secure development and operations.

5. *How does DynamoDB ensure high availability and scalability?*

Reference: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

Amazon DynamoDB is a very highly available and scalable NoSQL database service which is the result of the combination of distributed architecture, automatic scaling mechanisms, and regional replication features. DynamoDB is totally managed as well as serverless database at its core which means that the AWS takes care of all the infrastructure provisioning, maintenance, patching, and scaling automatically. Thus, users are not required to manage servers or clusters and can thus concentrate on application development.

- High Availability: DynamoDB uses multi-AZ replication to achieve high availability. The data copied into a DynamoDB table is replicated over several Availability Zones (AZs) within an AWS Region. This means that data is safe and can be retrieved even when the problem is at one of the data centers. AWS has set up a monitoring system that always watches over the infrastructure and does the automatically traffic redirecting if one of the replicas goes offline, hence keeping still the disruption to a minimum.
  If even more redundancy and availability are needed across the regions, then DynamoDB has a Global Tables option. This function will bring your table data to multiple AWS Regions and thus,

it will let users in different areas of the world access it quickly and easily. The user's application will also be operational even if there happens to be an outage in the region, thanks to this feature.

- **Scalability:** DynamoDB is capable of handling huge amounts of data using a partitioned architecture which has a massive scalability feature. The primary key determines the number of partitions the table is divided into, allowing DynamoDB to share the read and write load across many servers. There are two capacity modes options that are available to the users:

  1. Provisioned capacity is the mode in which you indicate the read and write capacity units (RCUs and WCUs) and is resource-efficient, allocating the necessary resources.
  2. On-demand capacity is another mode in which DynamoDB, through its power of auto-scaling, decides the capacity based on the traffic, thus making it the best option for workloads that are hard to predict even through the "scale" feature up to the point of automatic adjustment..
- DynamoDB also features auto-scaling, which auto-manages provisioned throughput from the usage pattern, resulting in a smooth transition during traffic spikes while at the same time cutting down on costs through the prevention of over-provisioning.

Moreover, through very clever management of the data and optimizing the access paths, the design of DynamoDB has very low latency even when a lot of data is being handled. In conclusion, the high availability of DynamoDB comes from the multi-AZ and multi-region data replication, while it has the capability of scaling elastically through the automated partitioning and capacities adjustments techniques, thus, it can be considered as a perfect fit for mission-critical, large-scale applications.

Infrastructure as Code (4 points):

Reference: https://docs.aws.amazon.com/prescriptive-guidance/latest/getting-started-terraform/resources.html

6. Why is Terraform preferred over manually creating AWS resources?

Terraform is preferred over manually creating AWS resources for several key reasons:

- **Infrastructure as Code (IaC):** Terraform allows you to define cloud infrastructure in code, which can be version-controlled, reviewed, and reused. This brings software development best practices to infrastructure management.

- **Automation and Repeatability:** Using Terraform scripts, you can repeatedly create the same infrastructure environment (e.g., for development, testing, and production) with minimal effort and without human error.

- **Resource Dependency Management:** Terraform understands the dependencies between resources and automatically manages the order of provisioning. For example, it ensures that a VPC is created before its subnets and routing tables.

- **Cross-Platform Support:** Terraform is cloud-agnostic. It can manage infrastructure across AWS, Azure, Google Cloud, and many other services from a single tool using the same configuration language (HCL).

- **Efficiency in Updates:** With Terraform, changes to infrastructure are as simple as updating code and applying it. Terraform can detect what has changed and only perform necessary actions.

In contrast, manually creating resources through the AWS Console is time-consuming, error-prone, and difficult to scale or track consistently especially in large, collaborative environments.

7. What is the benefit of using a state file in Terraform?

The Terraform state file (terraform.tfstate) is very important when it comes to Terraform's way of handling the infrastructure:

- **Stores Resource Mapping:** The mapping of the real-world resources (e.g., VPCs, EC2 instances) to the Terraform configuration keeps the state file updated on the current condition of your infrastructure. This enables Terraform to know what already exists and what needs to be created, updated, or removed.
- **Enables Dependency Awareness:** The state file helps Terraform to manage and categorize resource dependencies properly and thus, take care of the operations in the right order.
- **Improves Performance:** With the metadata about resources and their attributes being cached, Terraform is able to limit the querying of cloud provider APIs, which results in faster operations.
- **Enables Collaboration:** When kept in a remote location (e.g., in S3 with state locking via DynamoDB), the state file makes it possible for teams to collaborate safely as it prohibits more than one person from making changes to the Terraform configuration that are in conflict with each other at the same time.

Deliverables (4 points):

- *Screenshots of DynamoDB operations.*
- *Web server setup documentation.*
- *Terraform configuration files and deployment report.*
- ***AWS Architecture Diagrams (Draw.io, PNG, or PDF).***
- *Answers to written questions in **PDF or DOCX format**.*

Total: 50 points

Note*: don't forget to destroy all created resources to avoid getting charged.*