Github repo: https://github.com/Amrutha-J822/Netflix-Design-Implementation-with-Terraform

Netflix: Video Streaming Service Design Document

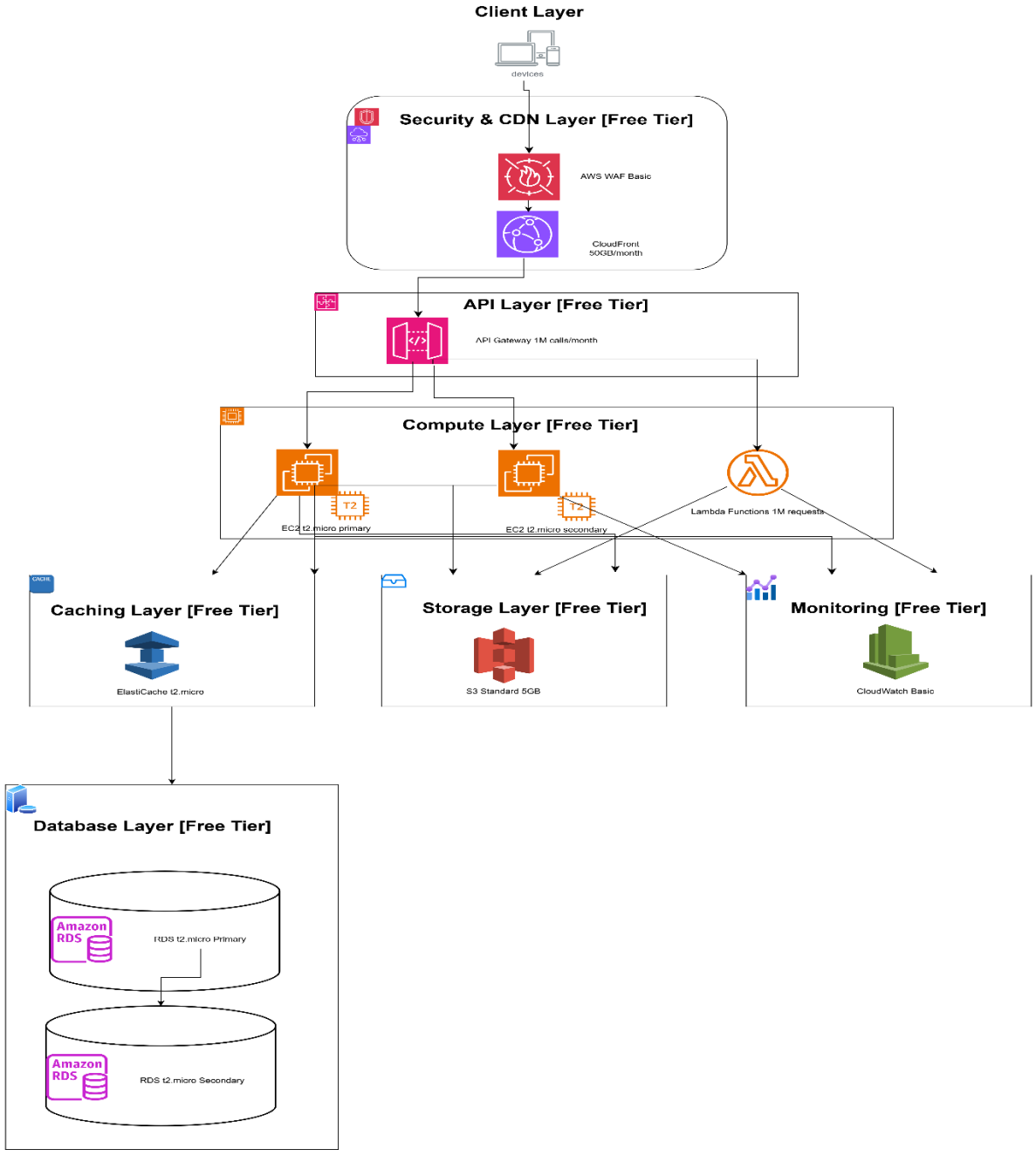
1. Service Overview :

Netflix is a video streaming platform that allows users to:

1. Watch movies and TV shows on-demand
2. Browse content catalog
3. Manage personal watchlists
4. Support multiple device streaming

2. Architecture Overview:

Architecture design:



Link to Architecture Diagram:
https://app.diagrams.net/#G11lrgf8yCwjZKjnztC0zPCS8N2kh7fT1P#%7B%22pageId%22%3A%22UvJIOaGWNZtmt75ACKXv%22%7D

Key Components:

1. Security Layer:
    - AWS WAF Basic: Provides web application firewall protection
    - CloudFront: Content delivery with 50GB free tier limit
    - API Gateway: Manages API requests with 1M free requests/month
2. Compute Layer:
    - EC2 t2.micro instances: Hosts web and application services
    - Lambda Functions: Handles event-driven processing
    - Auto Scaling: Manages compute resources within free tier limits
3. Backend Services:
    - Authentication Service: User management and session control
    - Catalog Service: Content metadata management
    - Streaming Service: Video delivery optimization
4. Storage Layer:
    - S3 Standard (5GB): Video content and static assets
    - RDS t2.micro: User and catalog data with primary-secondary setup
    - ElastiCache t2.micro: Caching frequently accessed data

3. Resiliency Implementation:

3.1 Elasticity:

- Auto Scaling Groups:
    - Scale up: CPU utilization > 70%
    - Scale down: CPU utilization < 30%
    - Maximum instances limited to free tier eligible resources
    - Lambda functions for sporadic workloads

3.2 Auto Recovery:

- CloudWatch Basic monitoring
    - 5-minute intervals for metrics (free tier)
    - Basic health checks for critical services
    - Resource utilization alerts

3.3 Failure Isolation:

- Multi-AZ deployment within free tier limits
- Service independence maintained through microservices
- Regional redundancy based on free tier eligibility

4. Performance Requirements:

4.1 User Data (S3 for video content - Free Tier: 5GB)

- Initial user base: 1,000 daily active users
- Growth projections:
  - Month 3: 1,728 users (20% monthly growth)
  - Month 6: 2,986 users
  - Month 12: 8,916 users
- Average request size optimization:
  - Metadata: 5KB (reduced from 10KB for free tier)
  - Video chunks: 2MB (optimized from 5MB)
- Peak hours: 7 PM - 11 PM (40% of daily traffic)
- Requests per user per day (optimized):
  - Browse: 15 requests (reduced from 20)
  - Stream: 1.5 hours average (540 chunks)
- Seasonal variance: +20% during holidays/weekends

4.2 Storage Data (Free Tier Optimized)

- Daily operations within free tier limits:
  - Reads: 1,000 users × 555 requests = 555,000
  - Writes: 1,000 users × 3 actions = 3,000
- Request sizes (optimized):
  - Read: 2MB (video), 5KB (metadata)
  - Write: 1KB (user actions)
- Storage requirements (within 5GB S3 free tier):
  - Initial: 500 movies × 8MB = 4GB
  - User data: 1,000 users × 500KB = 500MB
- Retention:
  - User history: 6 months (reduced from 12)
  - Viewing statistics: 12 months (reduced from 24)
  - Content: Rolling basis (most popular content)

4.3 Database Requirements (t2.micro RDS & ElastiCache)

- Transactions per second:
  - Peak: 50 TPS (optimized for free tier)
  - Average: 20 TPS
- Read/Write ratio: 90:10
- Query patterns:
  - Simple: User profile, basic content metadata
  - Complex: Basic recommendations, simple search
- Caching (ElastiCache t2.micro):

    - Content metadata: 30 minutes TTL
    - User preferences: 10 minutes TTL
    - Popular content: 12 hours TTL

4.4 Bandwidth Requirements (CloudFront 50GB Free Tier)

- Data ingress:
    - User actions: 2MB/day/user
    - New content: 50GB/day
- Data egress:
    - Video streaming: 1.5GB/hour/user (optimized)
    - Metadata: 20MB/day/user
- Latency targets (adjusted for free tier):
    - Page load: < 3 seconds
    - Video start: < 4 seconds
    - Buffering: < 1%
- Traffic patterns:
    - Daily peak: Evening hours (2x normal)
    - Weekend peak: 1.3x weekday
    - Regional: Single region focus

5. Secure Architecture Considerations:

1. CNAS-1: Cloud Configuration Security Affected Components: S3, RDS t2.micro, EC2 t2.micro

- S3 bucket policies limited to 5GB free tier storage
- RDS t2.micro with basic encryption enabled
- EC2 t2.micro instances with mandatory security groups
- Resource limits aligned with free tier quotas

2. CNAS-2: Input Validation Affected Components: API Gateway, Lambda Functions

- API Gateway (1M free requests) with basic request validation
- Lambda functions with input sanitization
- Rate limiting within free tier thresholds

3. CNAS-3: Authentication & Authorization Affected Components: Authentication Service, CloudFront

- Basic JWT implementation without MFA
- CloudFront signed URLs within 50GB limit
- IAM roles with free tier service limits

4. CNAS-4: Development Pipeline Affected Components: EC2 t2.micro instances

- Basic CI/CD pipeline using AWS free tier tools
- Development and production environments

- Manual security checks for deployments

5. CNAS-5: Secrets Management Affected Components: RDS, Lambda Functions

- AWS Secrets Manager (free tier eligible)
- Basic encryption for sensitive data
- Environment variables for configuration

6. CNAS-6: Network Security Affected Components: VPC, Security Groups

- VPC with basic network segmentation
- Security group rules for t2.micro instances
- Basic network ACLs configuration

7. CNAS-7: Vulnerability Management Affected Components: EC2, S3

- Regular OS updates for t2.micro instances
- Basic security patches
- Manual vulnerability assessments

8. CNAS-8: Resource Management Affected Components: AWS Free Tier Resources

- Resource tagging for cost tracking
- Automated cleanup of unused resources
- Basic asset inventory management

9. CNAS-9: Resource Quotas Affected Components: Free Tier Services

- t2.micro instance limits
- API Gateway throttling (1M requests)
- S3 storage quotas (5GB)

10. CNAS-10: Basic Monitoring Affected Components: CloudWatch

- Basic CloudWatch metrics
- Simple automated alerts
- Essential service health checks

6. Cost-Optimized Architecture Implementation:

6.1 Free Tier Service Utilization

1. Compute Layer
    - EC2 t2.micro instances (750 hours/month free)
        - Run essential services during peak hours
        - Implement automated start/stop schedules for dev environments

- Lambda Functions (1M free requests/month)
    - Use for periodic tasks and background processing
    - Implement request batching to optimize invocations

2. Storage Optimization
    - S3 (5GB free)
        - Implement lifecycle policies for infrequently accessed content
        - Use S3 Intelligent-Tiering for automatic cost optimization
    - RDS t2.micro (750 hours/month free)
        - Optimize database queries to reduce compute usage
        - Implement efficient indexing strategies
3. Network Optimization
    - CloudFront (50GB/month free)
        - Cache static content at edge locations
        - Optimize video chunk sizes for efficient delivery
    - API Gateway (1M calls/month free)
        - Implement request throttling
        - Use caching to reduce backend calls

6.2 Cost Control Measures

1. Auto Scaling Policies
    - Implement scale-in during off-peak hours
    - Set appropriate cooldown periods
    - Use target tracking scaling policies
2. Resource Lifecycle Management
    - Automated cleanup of unused resources
    - Regular audit of active services
    - Implementation of resource tagging for cost allocation

6.3 Monitoring and Optimization

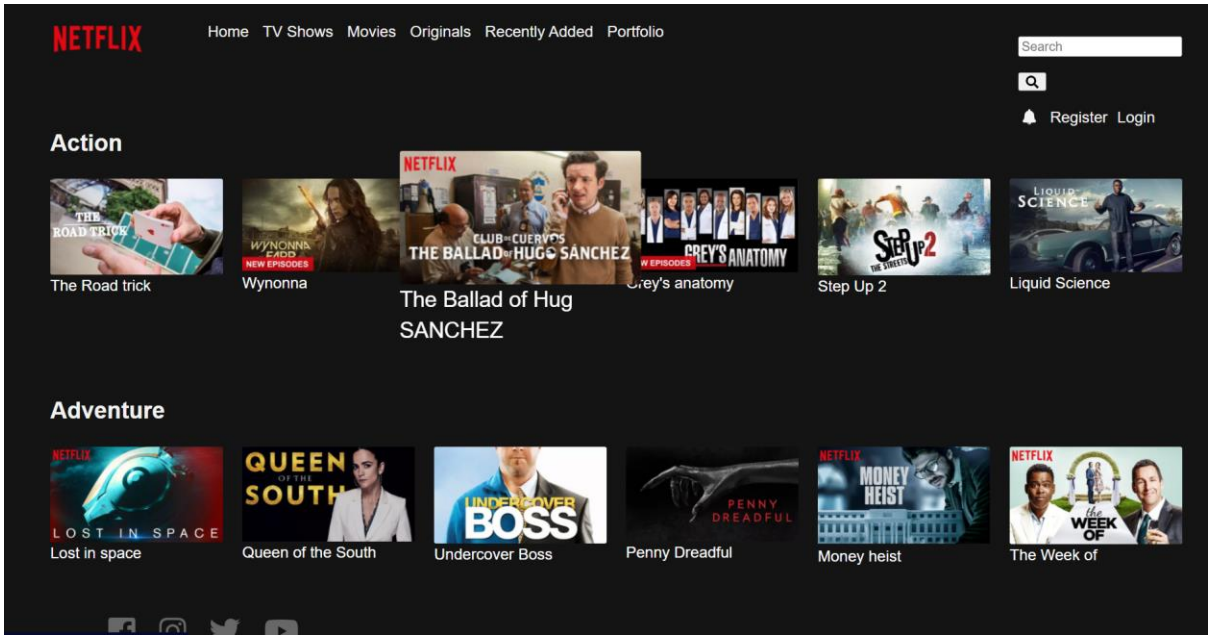
1. CloudWatch Basic Monitoring
    - Track resource utilization metrics
    - Set up alerts for cost thresholds
    - Monitor service usage patterns

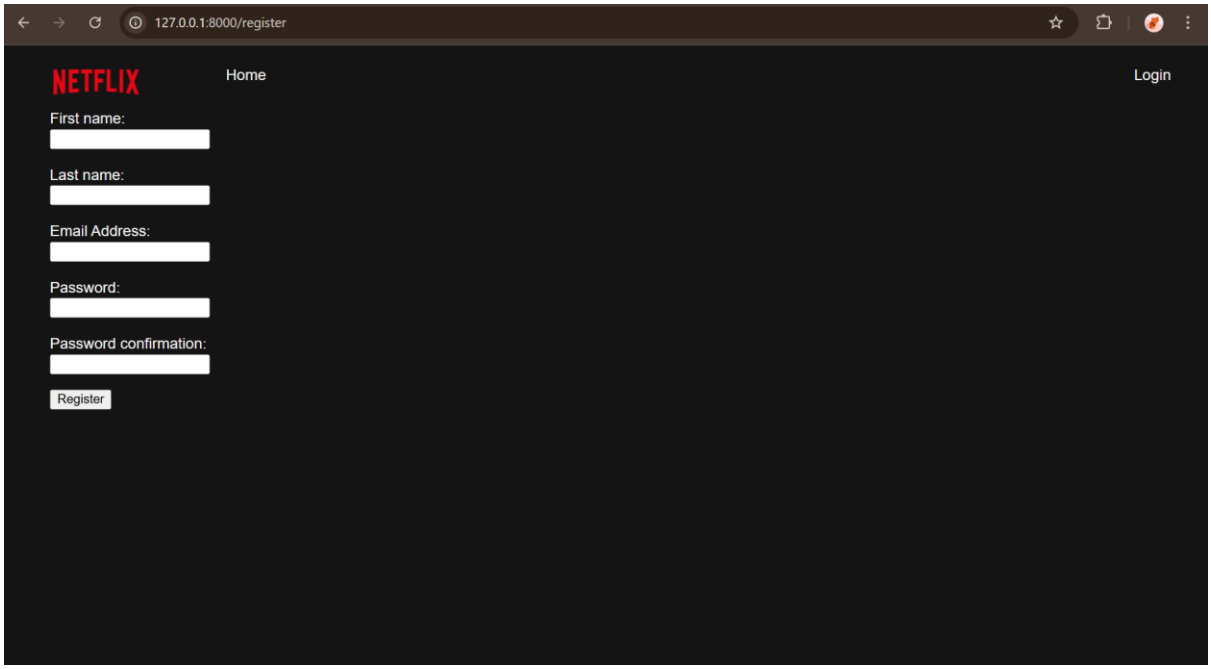
2. Cost Analysis
    - Regular review of AWS Cost Explorer data
    - Implementation of cost allocation tags
    - Monthly budget reviews and adjustments

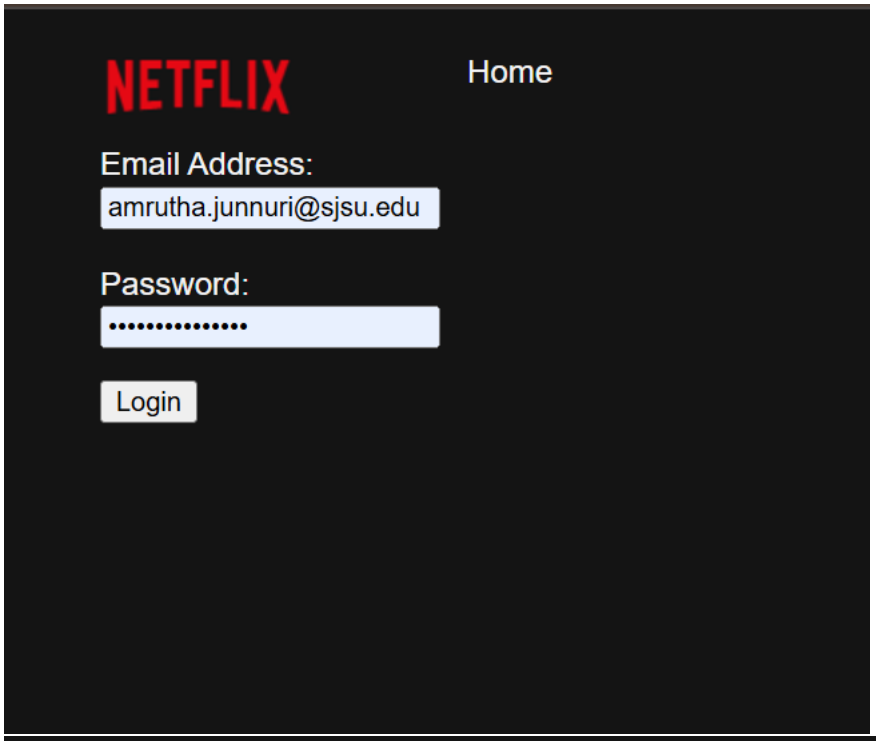
6.4 Development Environment Optimization

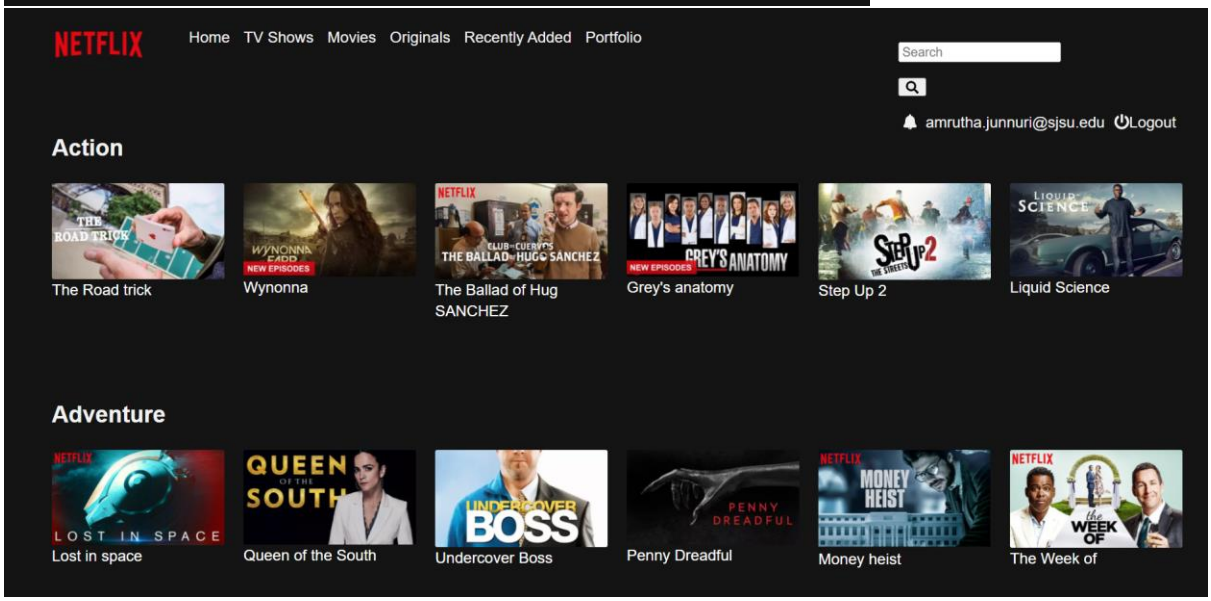
1. Resource Scheduling
    - Automatic shutdown of non-production environments
    - Reduced capacity for testing environments
    - Use of spot instances for development workloads
2. Storage Management
    - Regular cleanup of development data
    - Implementation of storage lifecycle policies
    - Use of compressed storage formats

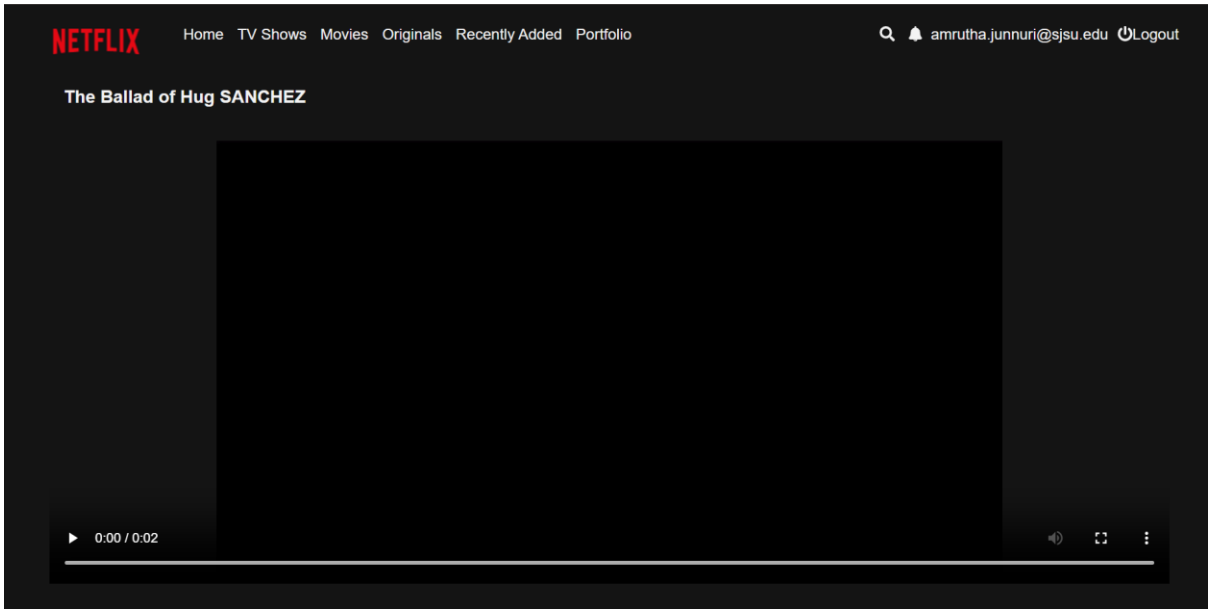
Netflix App working:





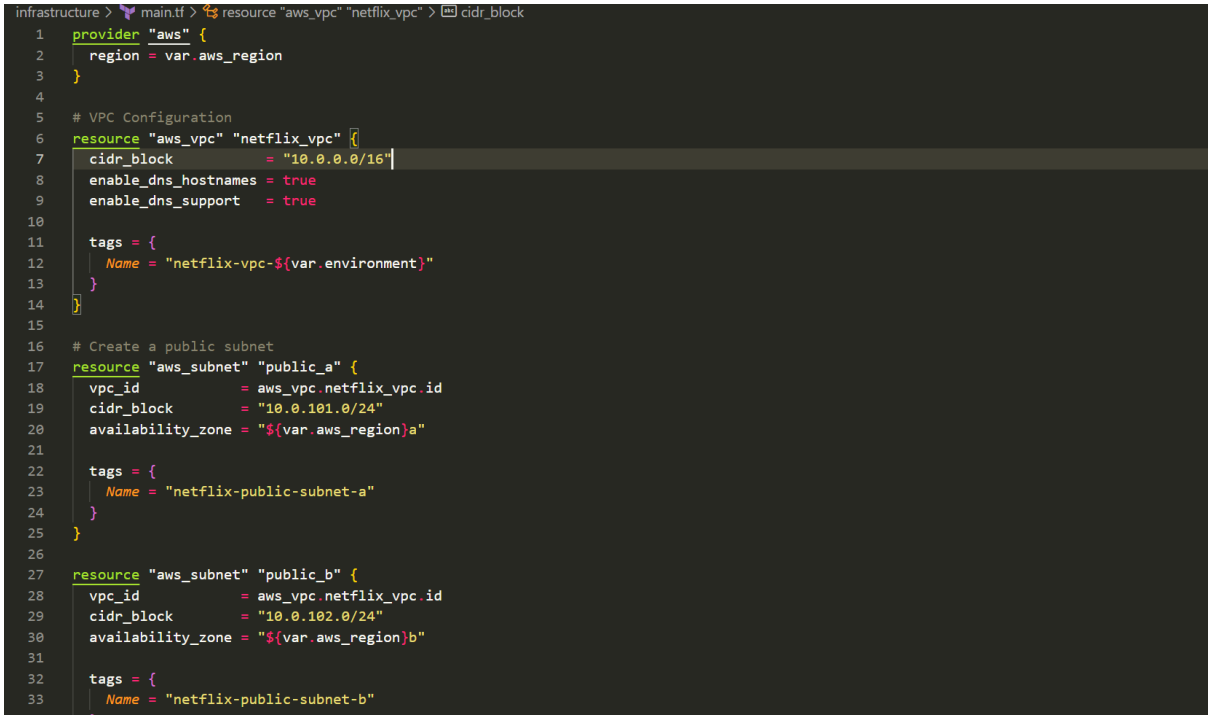

NETFLIX
Home
Email Address:
amrutha.junnuri@sjsu.edu
Password:
Login

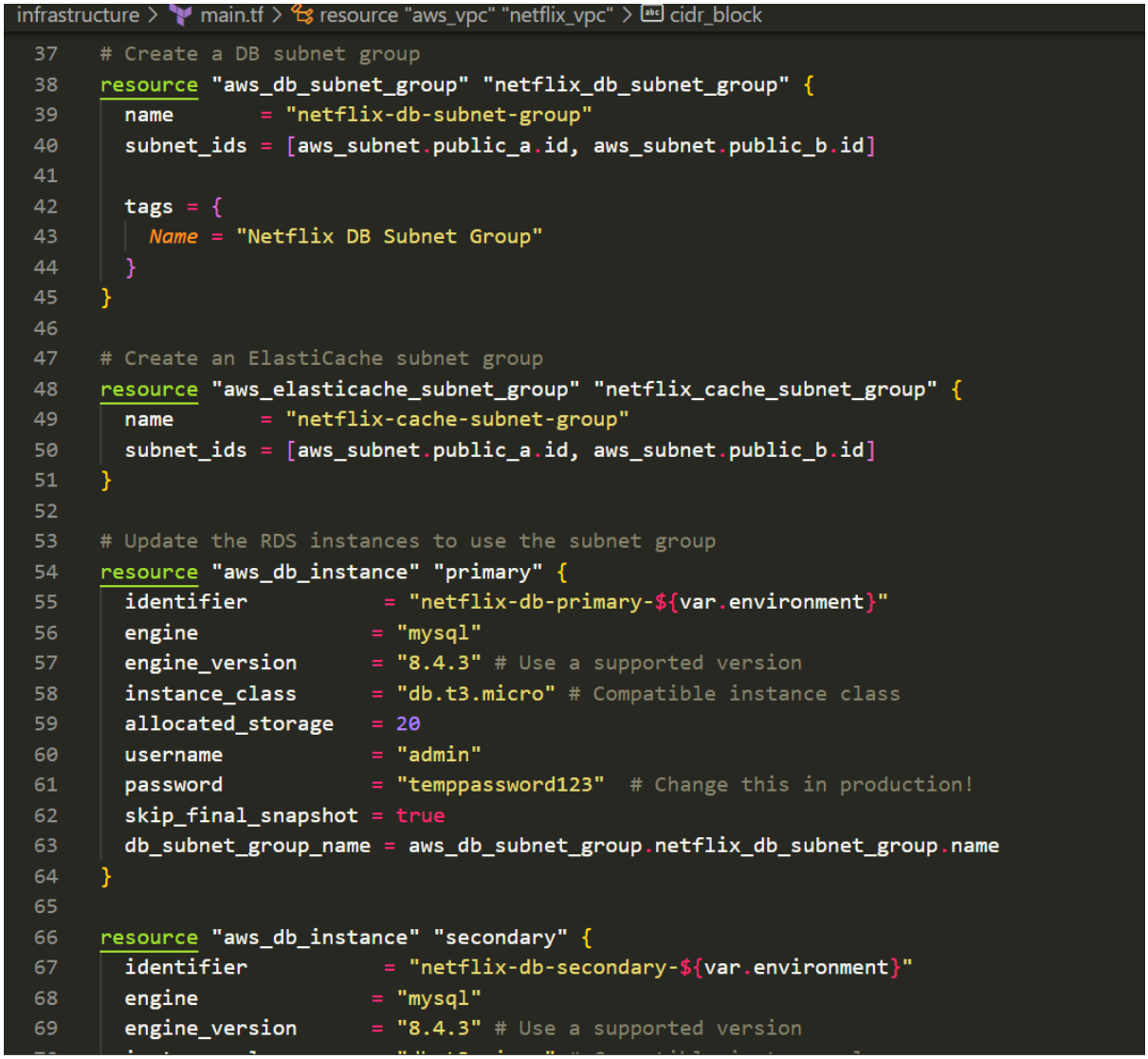

NETFLIX
Home
TV Shows
Movies
Originals
Recently Added
Portfolio
Search
amrutha.junnuri@sjsu.edu
Logout
Action
The Road trick
Wynonna
The Ballad of Hug SANCHEZ
Grey's anatomy
Step Up 2
Liquid Science
Adventure
Lost in space
Queen of the South
Undercover Boss
Penny Dreadful
Money heist
The Week of

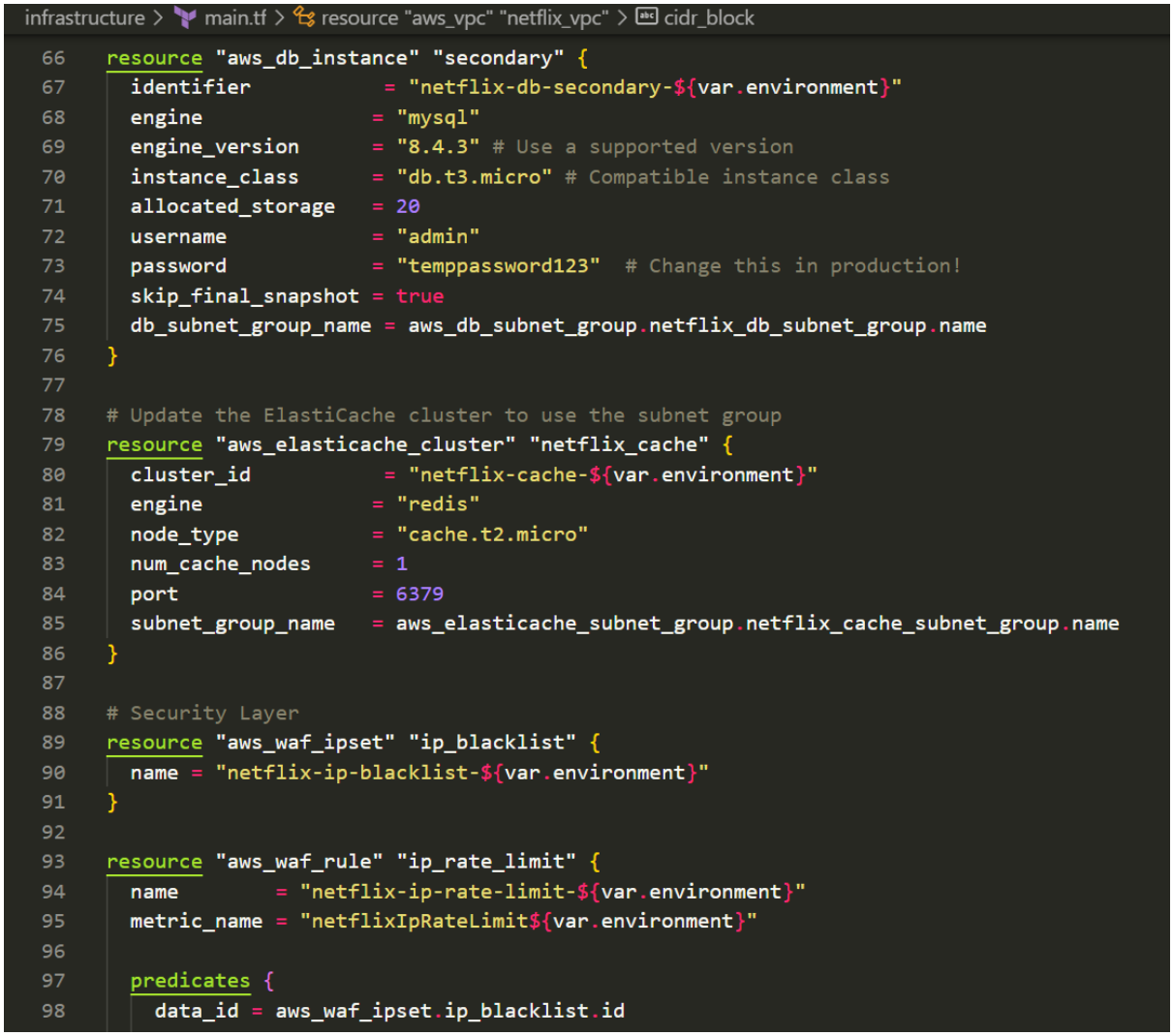

NETFLIX
Home TV Shows Movies Originals Recently Added Portfolio
amrutha.junnuri@sjsu.edu Logout
The Ballad of Hug SANCHEZ
0:00 / 0:02

Terraform code:

Main.tf:




infrastructure > main.tf > resource "aws_vpc" "netflix_vpc" > cidr_block
# Create a DB subnet group
resource "aws_db_subnet_group" "netflix_db_subnet_group" {
name = "netflix-db-subnet-group"
subnet_ids = [aws_subnet.public_a.id, aws_subnet.public_b.id]
tags = {
Name = "Netflix DB Subnet Group"
}
}
# Create an ElastiCache subnet group
resource "aws_elasticache_subnet_group" "netflix_cache_subnet_group" {
name = "netflix-cache-subnet-group"
subnet_ids = [aws_subnet.public_a.id, aws_subnet.public_b.id]
}
# Update the RDS instances to use the subnet group
resource "aws_db_instance" "primary" {
identifier = "netflix-db-primary-${var.environment}"
engine = "mysql"
engine_version = "8.4.3" # Use a supported version
instance_class = "db.t3.micro" # Compatible instance class
allocated_storage = 20
username = "admin"
password = "temppassword123" # Change this in production!
skip_final_snapshot = true
db_subnet_group_name = aws_db_subnet_group.netflix_db_subnet_group.name
}
resource "aws_db_instance" "secondary" {
identifier = "netflix-db-secondary-${var.environment}"
engine = "mysql"
engine_version = "8.4.3" # Use a supported version

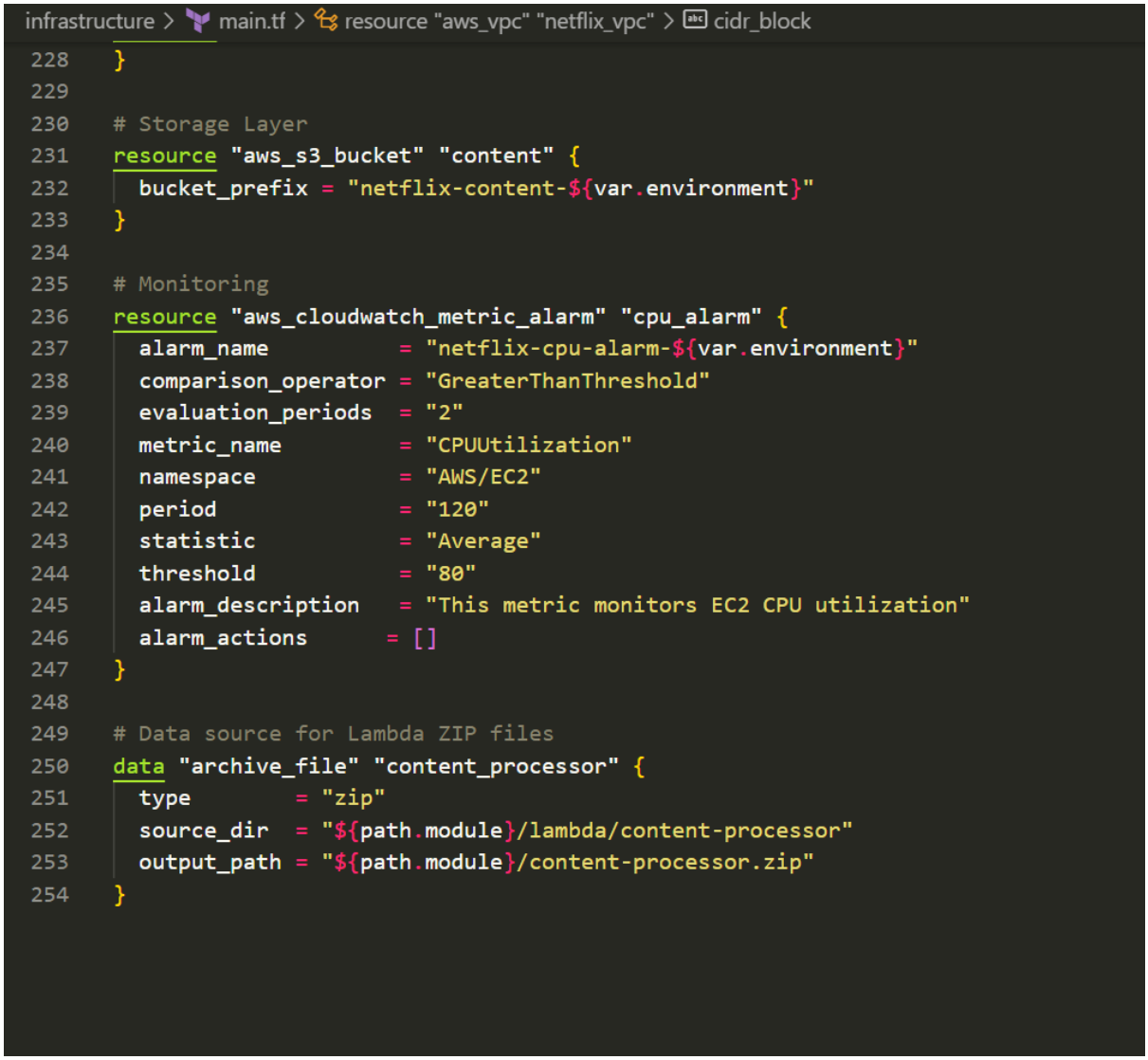

infrastructure > main.tf > resource "aws_vpc" "netflix_vpc" > cidr_block
resource "aws_db_instance" "secondary" {
identifier = "netflix-db-secondary-${var.environment}"
engine = "mysql"
engine_version = "8.4.3" # Use a supported version
instance_class = "db.t3.micro" # Compatible instance class
allocated_storage = 20
username = "admin"
password = "temppassword123" # Change this in production!
skip_final_snapshot = true
db_subnet_group_name = aws_db_subnet_group.netflix_db_subnet_group.name
}
# Update the ElastiCache cluster to use the subnet group
resource "aws_elasticache_cluster" "netflix_cache" {
cluster_id = "netflix-cache-${var.environment}"
engine = "redis"
node_type = "cache.t2.micro"
num_cache_nodes = 1
port = 6379
subnet_group_name = aws_elasticache_subnet_group.netflix_cache_subnet_group.name
}
# Security Layer
resource "aws_waf_ipset" "ip_blacklist" {
name = "netflix-ip-blacklist-${var.environment}"
}
resource "aws_waf_rule" "ip_rate_limit" {
name = "netflix-ip-rate-limit-${var.environment}"
metric_name = "netflixIpRateLimit${var.environment}"
predicates {
data_id = aws_waf_ipset.ip_blacklist.id


infrastructure > main.tf > resource "aws_vpc" "netflix_vpc" > cidr_block
93 resource "aws_waf_rule" "ip_rate_limit" {
97 predicates {
99 negated = false
100 type = "IPMatch"
101 }
102 }
103
104 resource "aws_waf_web_acl" "netflix_waf" {
105 name = "netflix-waf-${var.environment}"
106 metric_name = "netflixWaf${var.environment}"
107
108 default_action {
109 type = "ALLOW"
110 }
111
112 rules {
113 action {
114 type = "BLOCK"
115 }
116 priority = 1
117 rule_id = aws_waf_rule.ip_rate_limit.id
118 type = "REGULAR"
119 }
120 }
121
122 # Add the missing CloudFront OAI
123 resource "aws_cloudfront_origin_access_identity" "oai" {
124 comment = "OAI for Netflix content"
125 }
126
127 resource "aws_cloudfront_distribution" "netflix_cdn" {
128 enabled = true
129

```
  origin {
    domain_name = aws_s3_bucket.content.bucket_regional_domain_name
    origin_id   = "S3Origin"

    s3_origin_config {
      origin_access_identity = aws_cloudfront_origin_access_identity.oai.cloudfront_access_identity_path
    }
  }

  default_cache_behavior {
    allowed_methods        = ["GET", "HEAD"]
    cached_methods         = ["GET", "HEAD"]
    target_origin_id       = "S3Origin"
    viewer_protocol_policy = "redirect-to-https"

    forwarded_values {
      query_string = false
      cookies {
        forward = "none"
      }
    }
  }

  restrictions {
    geo_restriction {
      restriction_type = "none"
    }
  }

  viewer_certificate {
    cloudfront_default_certificate = true
  }
```

```
# API Layer
resource "aws_api_gateway_rest_api" "netflix_api" {
  name        = "netflix-api-${var.environment}"
  description = "Netflix API Gateway"
}

# Compute Layer
resource "aws_instance" "ec2_primary" {
  ami           = "ami-0c7217cdde317cfec"  # Amazon Linux 2023
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_a.id
  associate_public_ip_address = true

  user_data = <<-EOF
              #!/bin/bash
              yum update -y
              yum install -y python3 python3-pip git
              pip3 install django==4.0.2 Pillow==9.0.1
              EOF

  tags = {
    Name = "netflix-primary-${var.environment}"
  }
}

resource "aws_instance" "ec2_secondary" {
  ami           = "ami-0c7217cdde317cfec"
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_b.id
  associate_public_ip_address = true

  tags = {
    Name = "netflix-secondary-${var.environment}"
```

```
# Lambda Functions
resource "aws_iam_role" "lambda_role" {
  name = "netflix_lambda_role_${var.environment}"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action = "sts:AssumeRole"
      Effect = "Allow"
      Principal = {
        Service = "lambda.amazonaws.com"
      }
    }]
  })
}

resource "aws_lambda_function" "content_processor" {
  filename      = data.archive_file.content_processor.output_path
  function_name = "netflix-content-processor-${var.environment}"
  role          = aws_iam_role.lambda_role.arn
  handler       = "index.handler"
  runtime       = "python3.9"

  environment {
    variables = {
      ENVIRONMENT = var.environment
    }
  }
}

# Storage Layer
resource "aws_s3_bucket" "content" {
  bucket_prefix = "netflix-content-${var.environment}"
```


infrastructure > main.tf > resource "aws_vpc" "netflix_vpc" > cidr_block
}
# Storage Layer
resource "aws_s3_bucket" "content" {
bucket_prefix = "netflix-content-${var.environment}"
}
# Monitoring
resource "aws_cloudwatch_metric_alarm" "cpu_alarm" {
alarm_name = "netflix-cpu-alarm-${var.environment}"
comparison_operator = "GreaterThanThreshold"
evaluation_periods = "2"
metric_name = "CPUUtilization"
namespace = "AWS/EC2"
period = "120"
statistic = "Average"
threshold = "80"
alarm_description = "This metric monitors EC2 CPU utilization"
alarm_actions = []
}
# Data source for Lambda ZIP files
data "archive_file" "content_processor" {
type = "zip"
source_dir = "${path.module}/lambda/content-processor"
output_path = "${path.module}/content-processor.zip"
}

Outputs.tf:

infrastructure > outputs.tf > output "api_gateway_url" > value

```
# API Gateway URL
output "api_gateway_url" {
  value       = aws_api_gateway_rest_api.netflix_api.execution_arn
  description = "URL of the API Gateway endpoint"
}

# CloudFront Domain
output "cloudfront_domain" {
  value       = aws_cloudfront_distribution.netflix_cdn.domain_name
  description = "Domain name of the CloudFront distribution"
}

# Primary EC2 Instance Public IP
output "primary_ec2_public_ip" {
  value       = aws_instance.ec2_primary.public_ip
  description = "Public IP of the primary EC2 instance"
}

# Secondary EC2 Instance Public IP
output "secondary_ec2_public_ip" {
  value       = aws_instance.ec2_secondary.public_ip
  description = "Public IP of the secondary EC2 instance"
}

# RDS Primary Endpoint
output "rds_primary_endpoint" {
  value       = aws_db_instance.primary.endpoint
  description = "Endpoint of the primary RDS instance"
}

# RDS Secondary Endpoint
output "rds_secondary_endpoint" {
  value       = aws_db_instance.secondary.endpoint
```

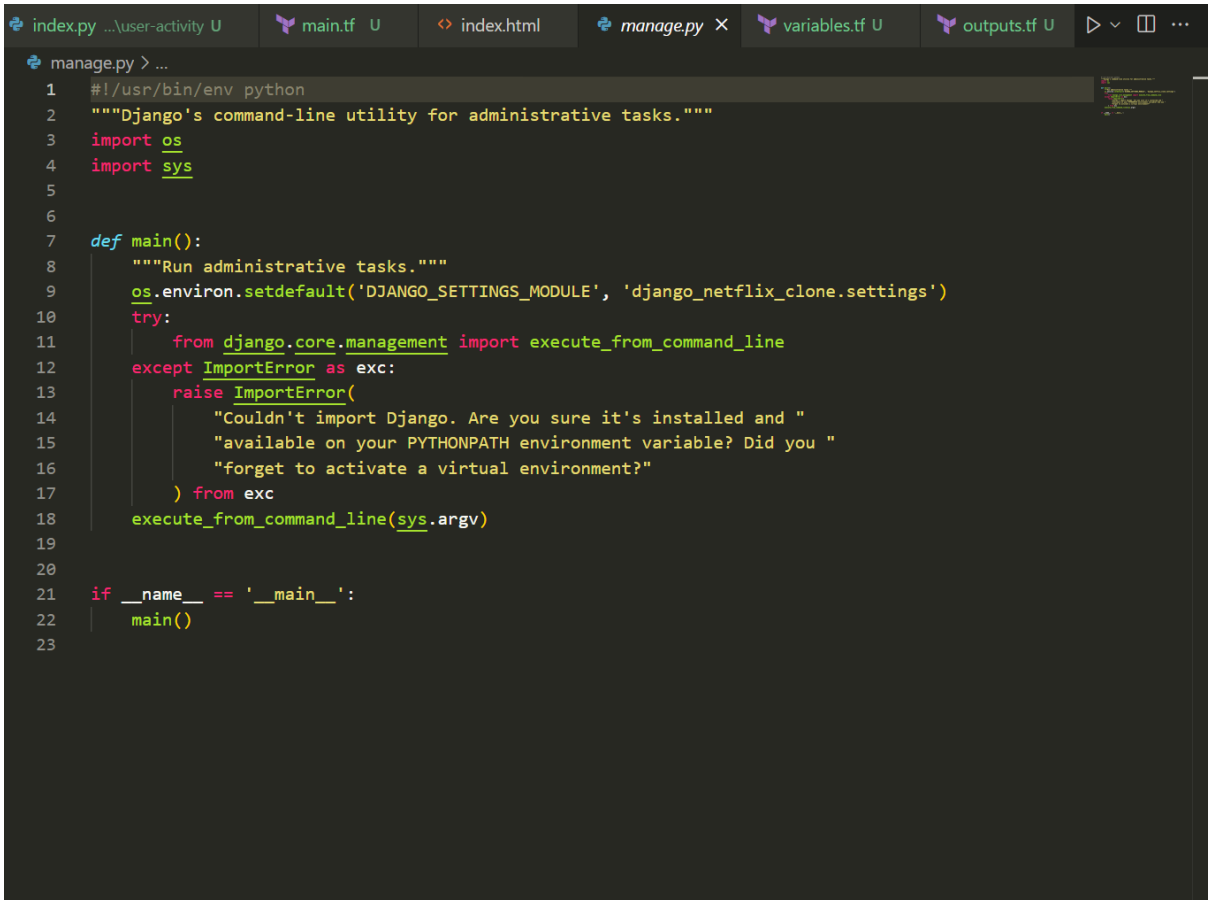
Variables.tf:

...\user-activity U | main.tf U | terraform.tfstate U | index.ht

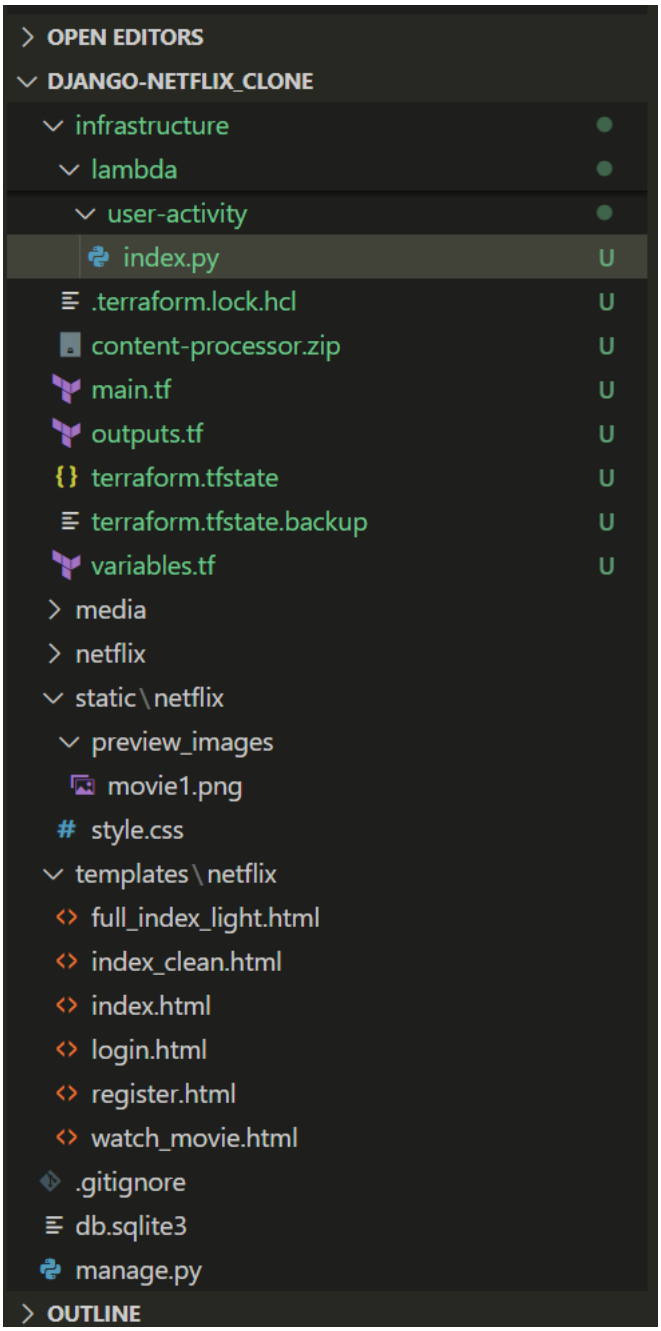
infrastructure > variables.tf > variable "environment"

```
# variables.tf
variable "aws_region" {
  description = "AWS region"
  default     = "us-east-1"
}

variable "environment" {
  description = "Environment name"
  default     = "dev"
}
```

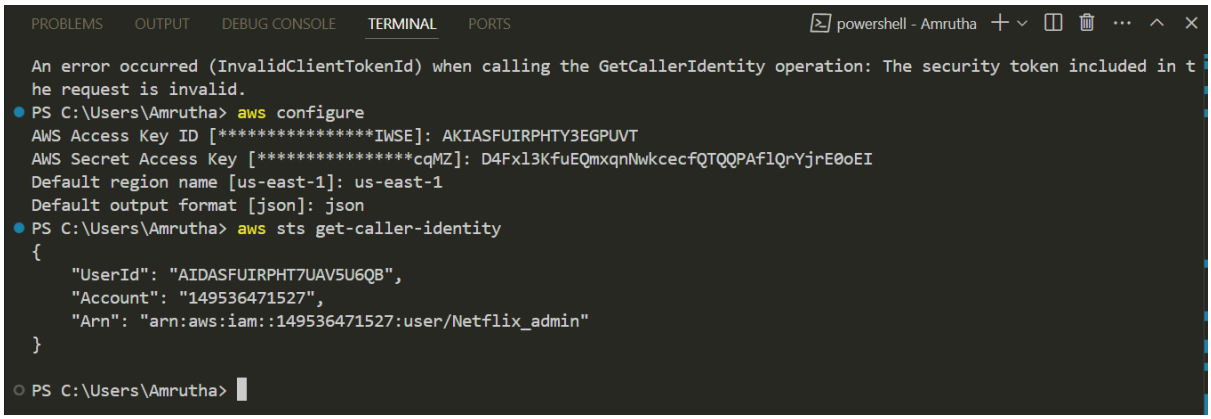
Netflix project:



Overal project folders:

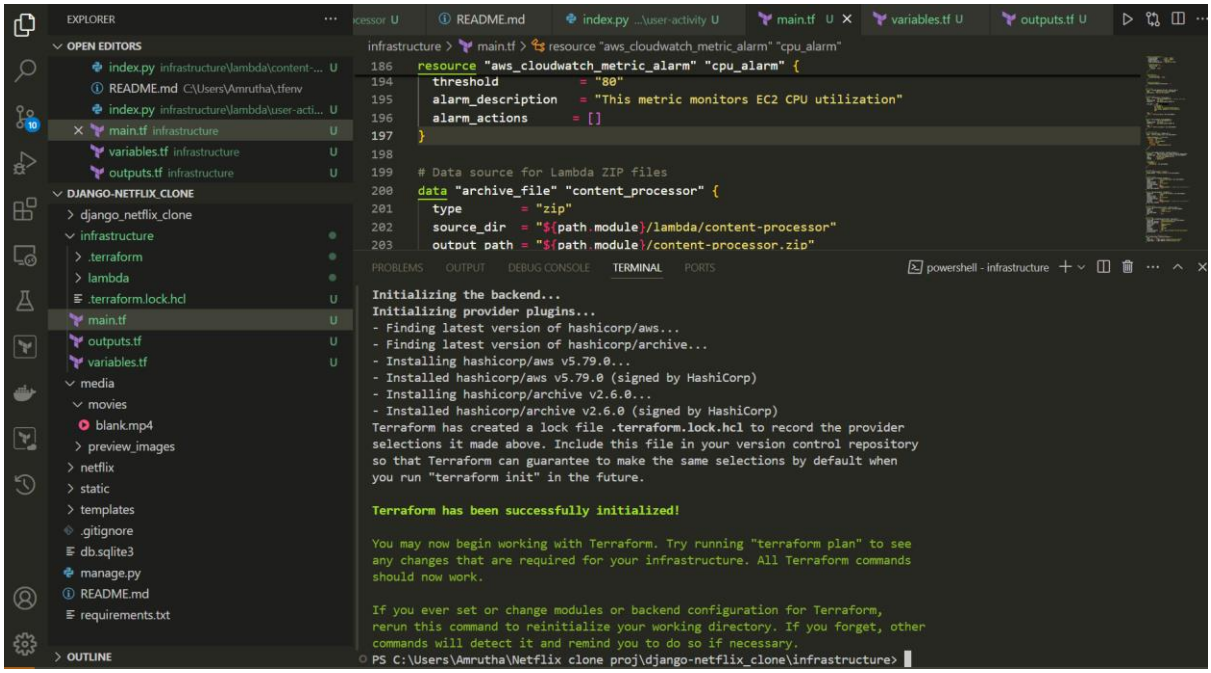


Configure your AWS IAM Account:



Terraform:

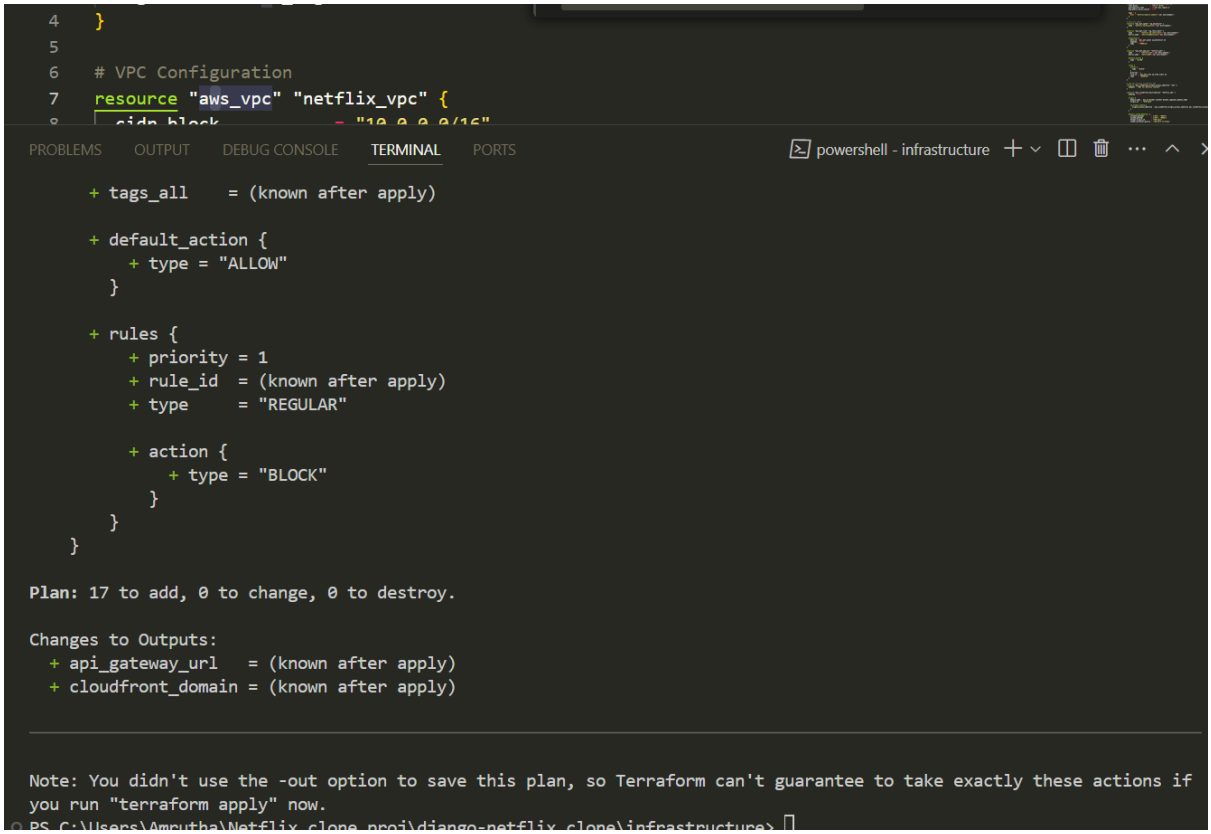
Terraform init:



Terraform validate:

```
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.
PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure> terraform validate
Success! The configuration is valid.

PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

Terraform plan:

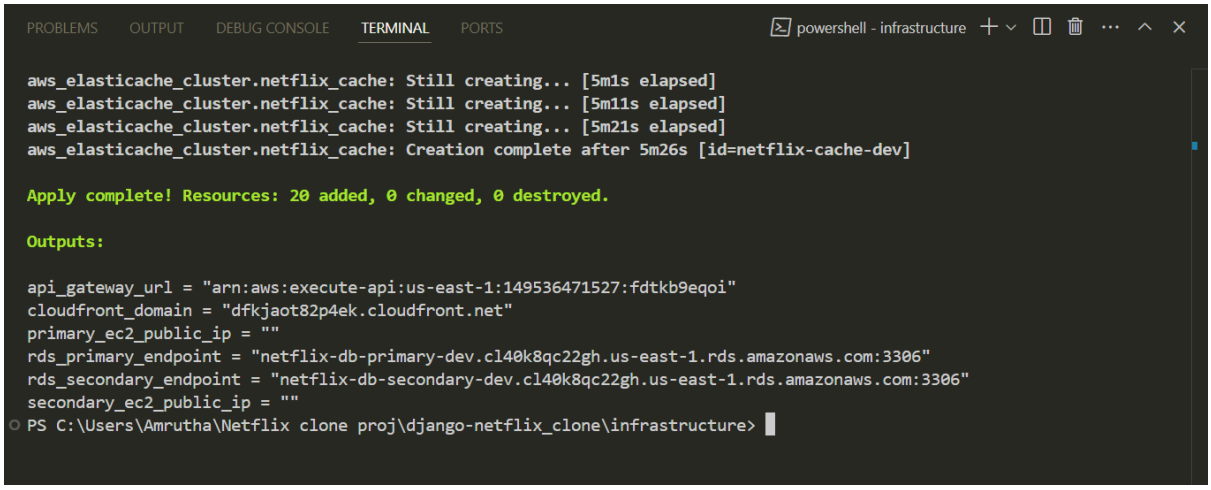


```
  }

  # VPC Configuration
  resource "aws_vpc" "netflix_vpc" {
    cidr_block           = "10.0.0.0/16"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS — powershell - infrastructure

```
      + tags_all    = (known after apply)

      + default_action {
          + type = "ALLOW"
        }

      + rules {
          + priority = 1
          + rule_id  = (known after apply)
          + type     = "REGULAR"

          + action {
              + type = "BLOCK"
            }
        }
    }

Plan: 17 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + api_gateway_url   = (known after apply)
  + cloudfront_domain = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

Terraform apply:

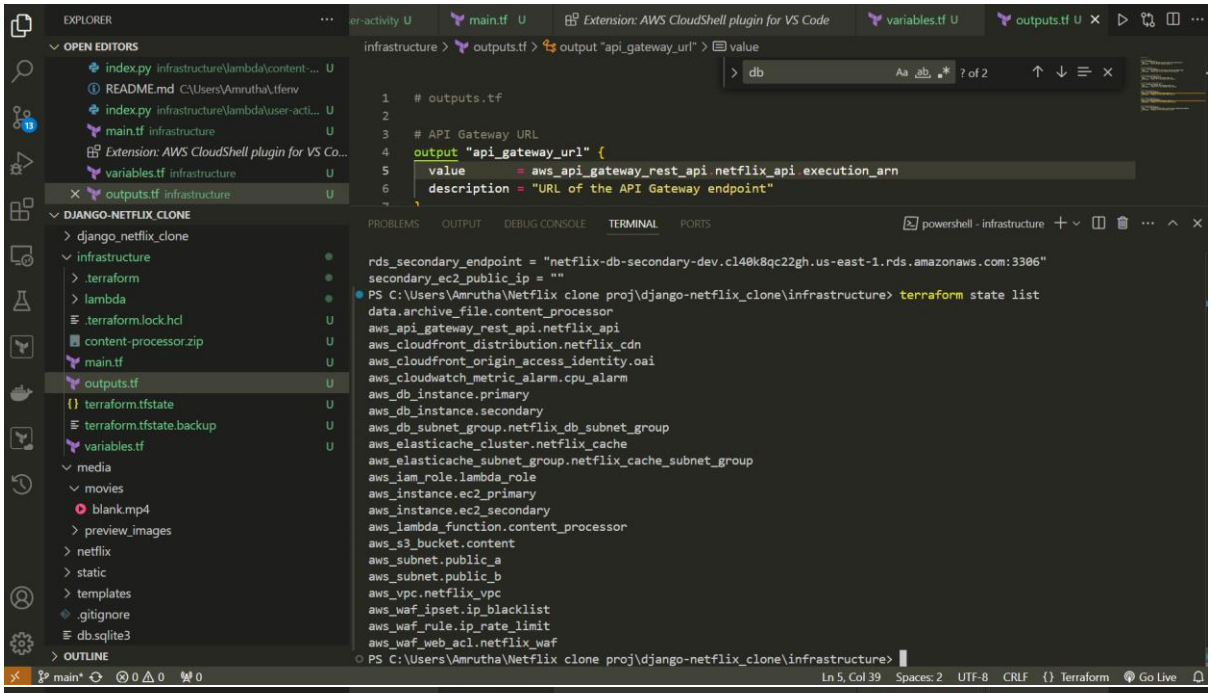


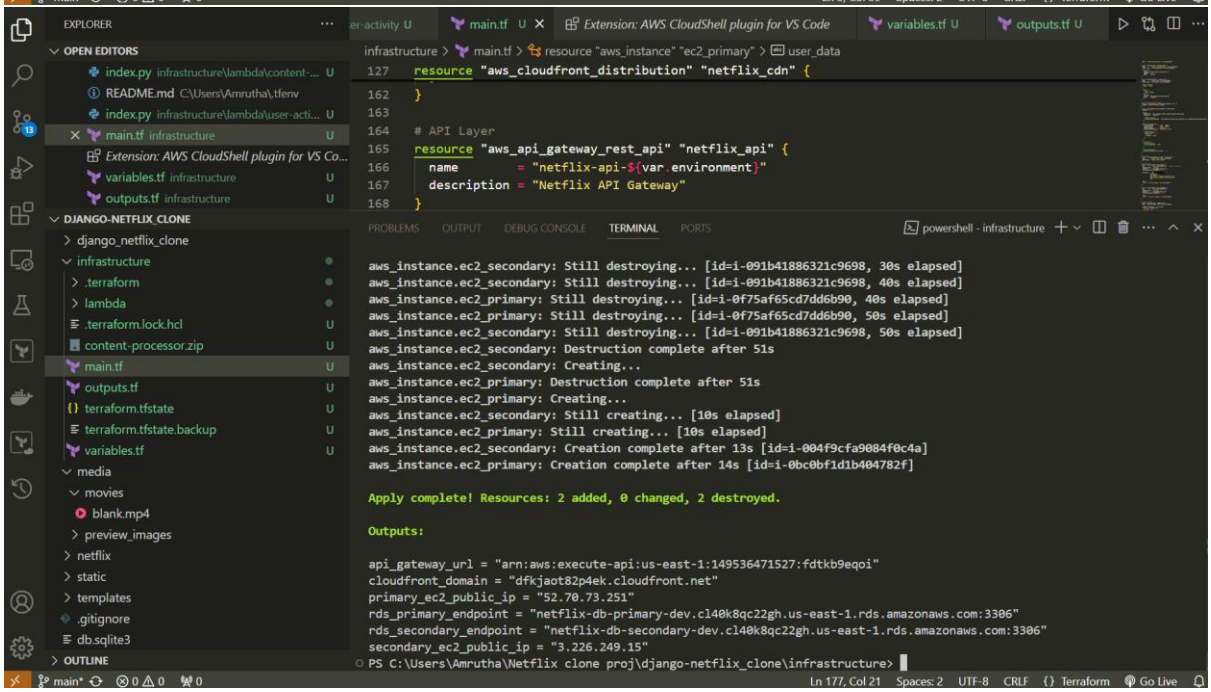
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS — powershell - infrastructure

```
aws_elasticache_cluster.netflix_cache: Still creating... [5m1s elapsed]
aws_elasticache_cluster.netflix_cache: Still creating... [5m11s elapsed]
aws_elasticache_cluster.netflix_cache: Still creating... [5m21s elapsed]
aws_elasticache_cluster.netflix_cache: Creation complete after 5m26s [id=netflix-cache-dev]

Apply complete! Resources: 20 added, 0 changed, 0 destroyed.

Outputs:

api_gateway_url = "arn:aws:execute-api:us-east-1:149536471527:fdtkb9eqoi"
cloudfront_domain = "dfkjaot82p4ek.cloudfront.net"
primary_ec2_public_ip = ""
rds_primary_endpoint = "netflix-db-primary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com:3306"
rds_secondary_endpoint = "netflix-db-secondary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com:3306"
secondary_ec2_public_ip = ""
PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

Outputs:

Terraform:





api_gateway_url = "arn:aws:execute-api:us-east-1:149536471527:fdtkb9eqoi"

cloudfront_domain = "dfkjaot82p4ek.cloudfront.net"

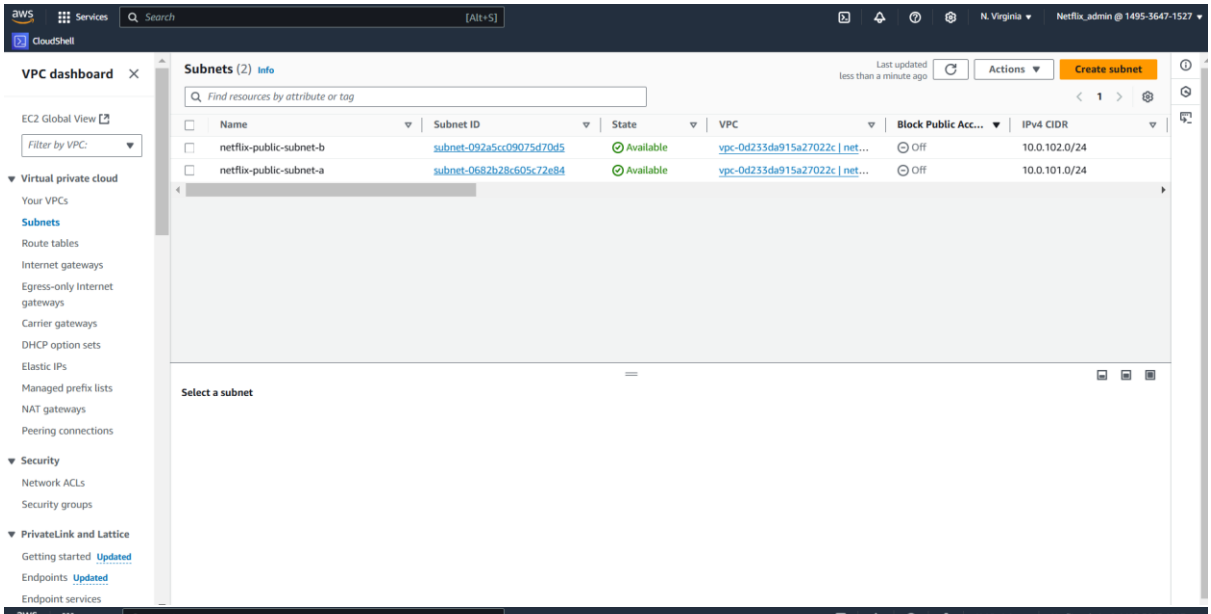
primary_ec2_public_ip = "52.70.73.251"

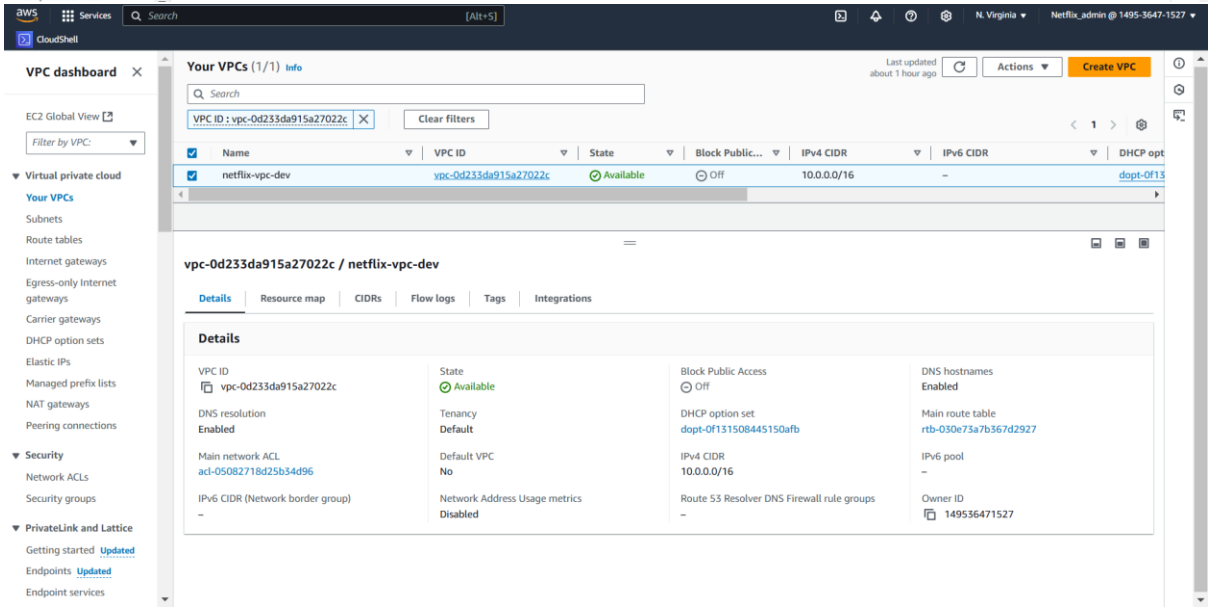
rds_primary_endpoint = "netflix-db-primary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com:3306"

rds_secondary_endpoint = "netflix-db-secondary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com:3306"

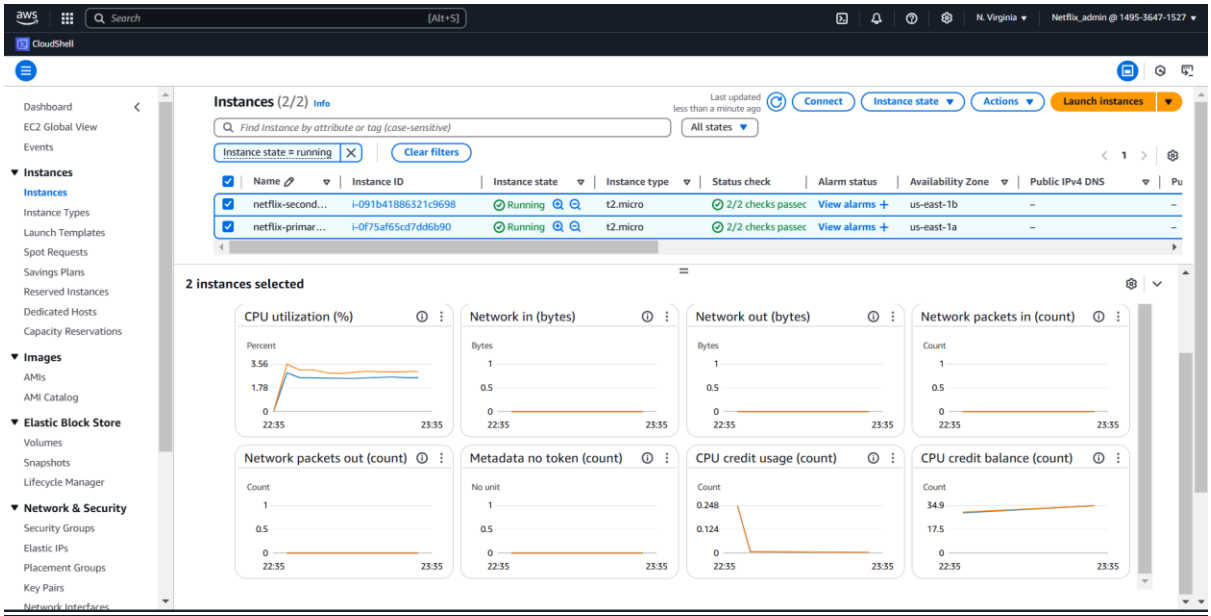
secondary_ec2_public_ip = "3.226.249.15"

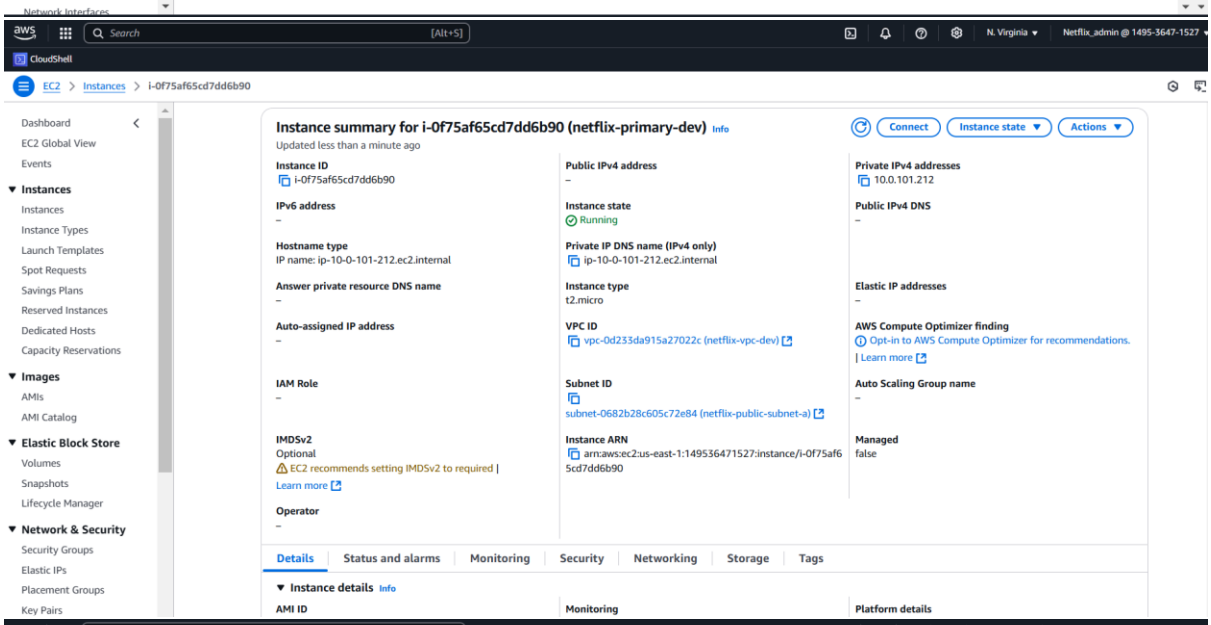
VPC:

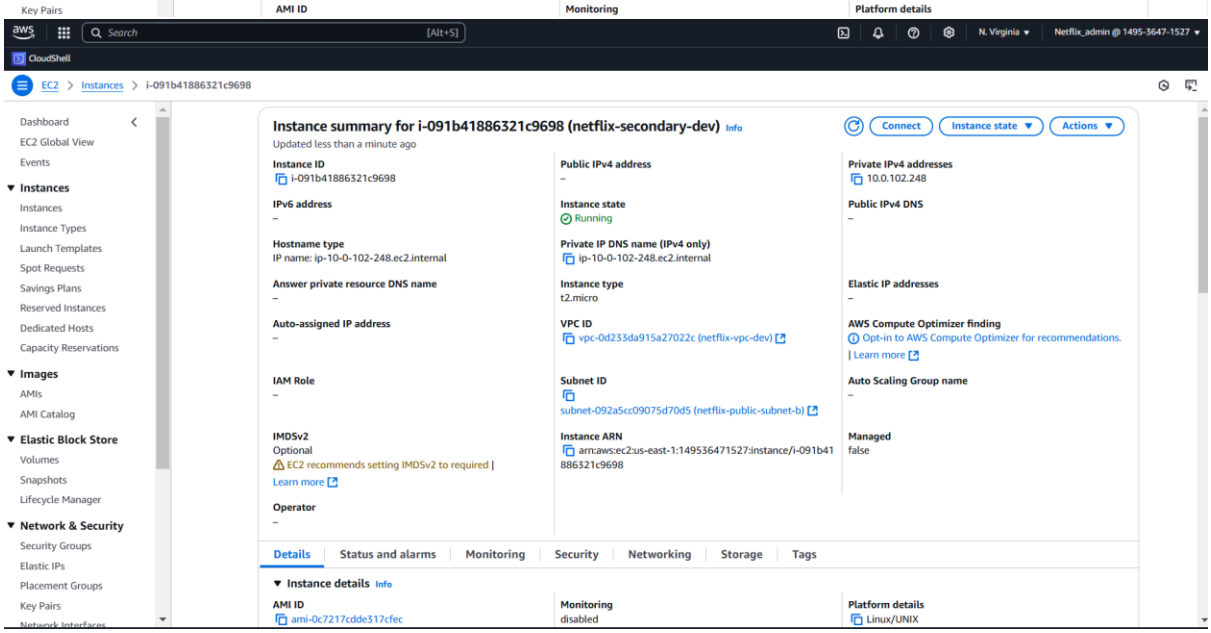

aws
Services
Search
[Alt+S]
CloudShell
N. Virginia
Netflix_admin @ 1495-3647-1527
VPC dashboard
EC2 Global View
Filter by VPC:
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only Internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started Updated
Endpoints Updated
Endpoint services
Subnets (2) Info
Last updated less than a minute ago
Actions
Create subnet
Find resources by attribute or tag
Name
Subnet ID
State
VPC
Block Public Acc...
IPv4 CIDR
netflix-public-subnet-b
subnet-092a5cc09075d70d5
Available
vpc-0d233da915a27022c | net...
Off
10.0.102.0/24
netflix-public-subnet-a
subnet-0682b28c605c72e84
Available
vpc-0d233da915a27022c | net...
Off
10.0.101.0/24
Select a subnet


aws
Services
Search
[Alt+S]
CloudShell
N. Virginia
Netflix_admin @ 1495-3647-1527
VPC dashboard
EC2 Global View
Filter by VPC:
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only Internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started Updated
Endpoints Updated
Endpoint services
Your VPCs (1/1) Info
Last updated about 1 hour ago
Actions
Create VPC
Search
VPC ID : vpc-0d233da915a27022c
Clear filters
Name
VPC ID
State
Block Public...
IPv4 CIDR
IPv6 CIDR
DHCP opt
netflix-vpc-dev
vpc-0d233da915a27022c
Available
Off
10.0.0.0/16
-
vpc-0d233da915a27022c / netflix-vpc-dev
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
Details
VPC ID
vpc-0d233da915a27022c
State
Available
Block Public Access
Off
DNS hostnames
Enabled
DNS resolution
Enabled
Tenancy
Default
DHCP option set
dopt-0f131508445150afb
Main route table
rtb-030e73a7b367d2927
Main network ACL
acl-05082718d25b34d96
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
-
IPv6 CIDR (Network border group)
-
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
-
Owner ID
149536471527

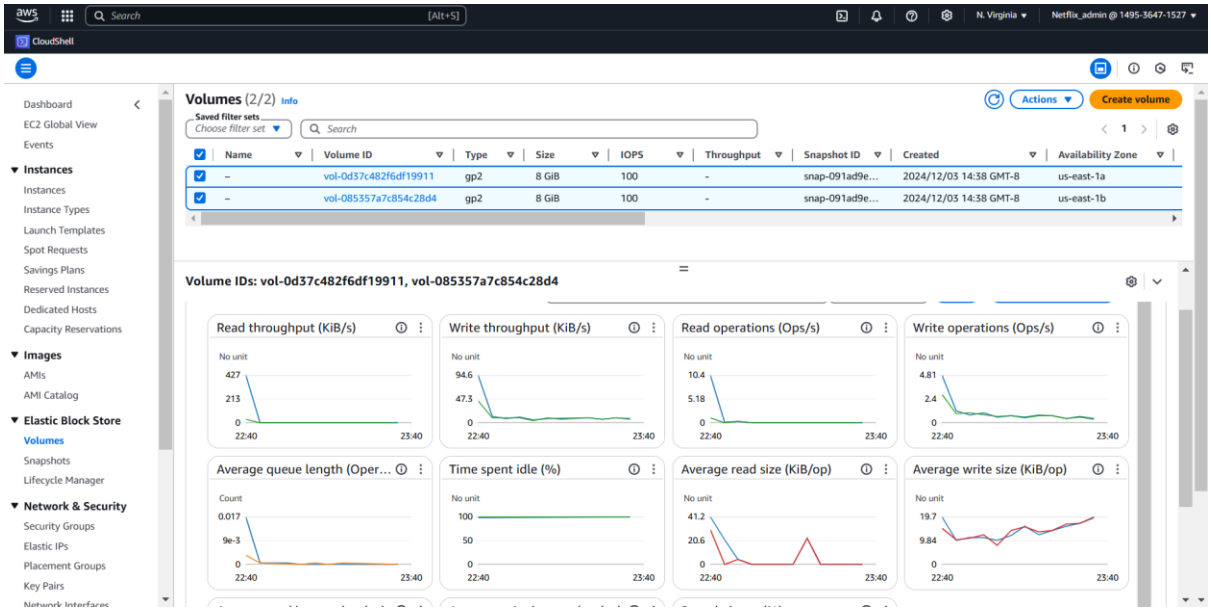
EC2:


Instances (2/2) Info
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
netflix-second...
i-091b41886321c9698
Running
t2.micro
2/2 checks passed
View alarms
us-east-1b
netflix-primar...
i-0f75af65cd7dd6b90
Running
t2.micro
2/2 checks passed
View alarms
us-east-1a
2 instances selected
CPU utilization (%)
Network in (bytes)
Network out (bytes)
Network packets in (count)
Network packets out (count)
Metadata no token (count)
CPU credit usage (count)
CPU credit balance (count)

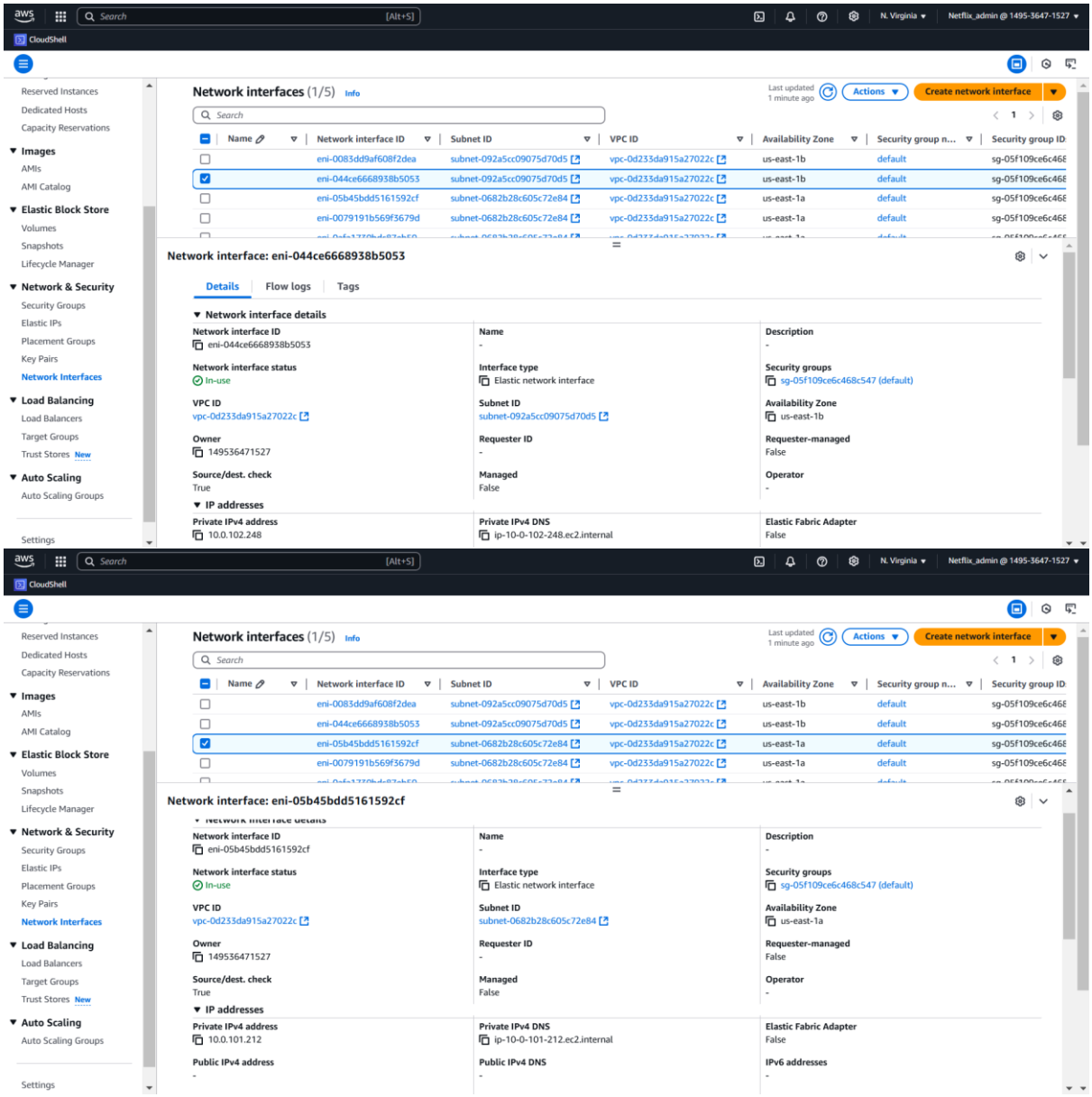

Instance summary for i-0f75af65cd7dd6b90 (netflix-primary-dev) Info
Instance ID
i-0f75af65cd7dd6b90
Public IPv4 address
Private IPv4 addresses
10.0.101.212
Instance state
Running
Public IPv4 DNS
Private IP DNS name (IPv4 only)
ip-10-0-101-212.ec2.internal
Hostname type
IP name: ip-10-0-101-212.ec2.internal
Instance type
t2.micro
Elastic IP addresses
VPC ID
vpc-0d233da915a27022c (netflix-vpc-dev)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Subnet ID
subnet-0682b28c605c72e84 (netflix-public-subnet-a)
Auto Scaling Group name
Instance ARN
arn:aws:ec2:us-east-1:149536471527:instance/i-0f75af65cd7dd6b90
Managed
false
IMDSv2
Optional
EC2 recommends setting IMDSv2 to required
Operator
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance details


Instance summary for i-091b41886321c9698 (netflix-secondary-dev) Info
Instance ID
i-091b41886321c9698
Public IPv4 address
Private IPv4 addresses
10.0.102.248
Instance state
Running
Public IPv4 DNS
Private IP DNS name (IPv4 only)
ip-10-0-102-248.ec2.internal
Hostname type
IP name: ip-10-0-102-248.ec2.internal
Instance type
t2.micro
Elastic IP addresses
VPC ID
vpc-0d233da915a27022c (netflix-vpc-dev)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Subnet ID
subnet-092a5cc09075d70d5 (netflix-public-subnet-b)
Auto Scaling Group name
Instance ARN
arn:aws:ec2:us-east-1:149536471527:instance/i-091b41886321c9698
Managed
false
IMDSv2
Optional
EC2 recommends setting IMDSv2 to required
Operator
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance details
AMI ID
ami-0c7217cdde317cfec
Monitoring
disabled
Platform details
Linux/UNIX

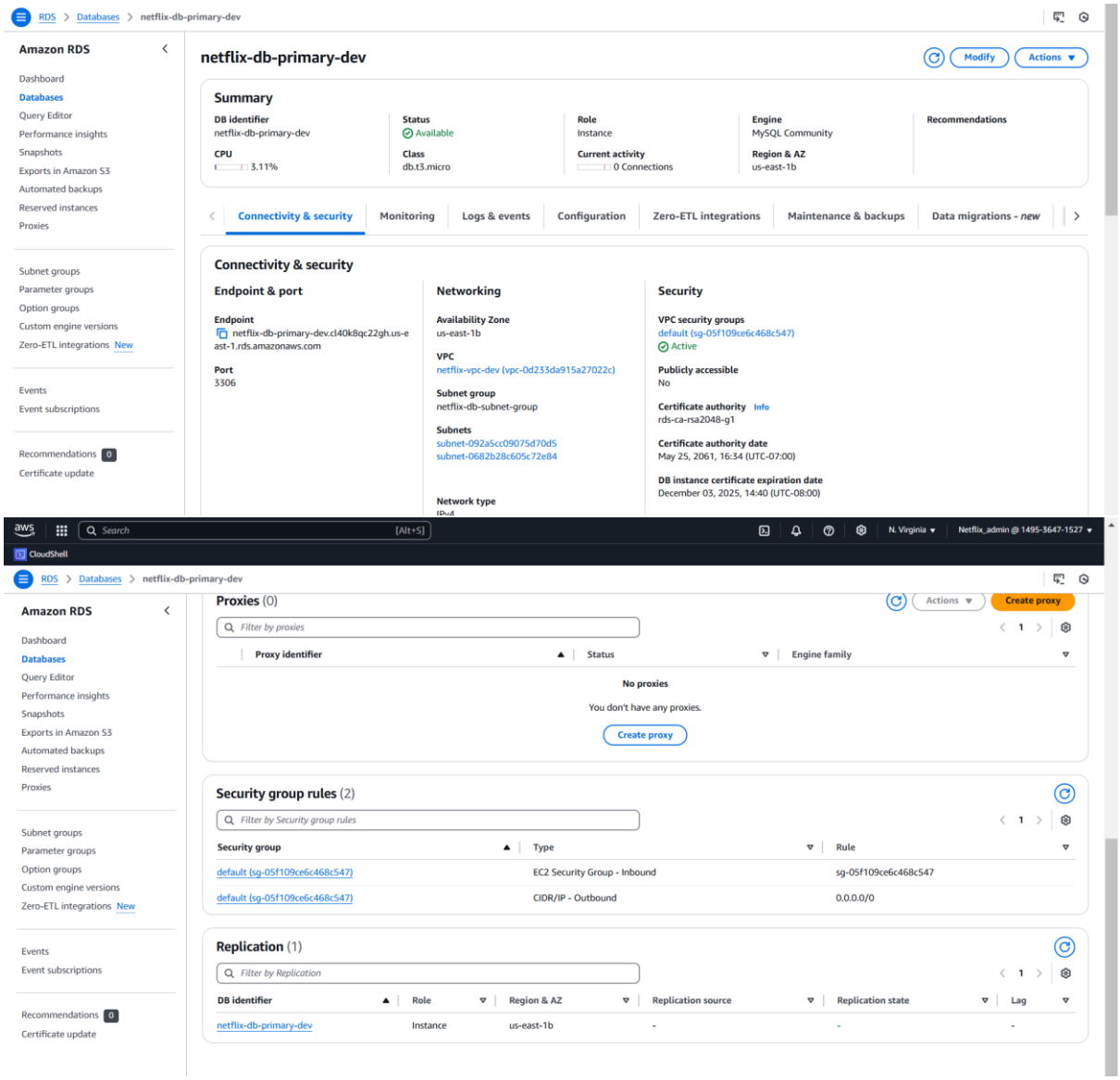
Volume:


Search
[Alt+S]
CloudShell
N. Virginia
Netflix_admin @ 1495-3647-1527
Dashboard
EC2 Global View
Events
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
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Volumes (2/2) Info
Actions
Create volume
Saved filter sets
Choose filter set
Search
Name
Volume ID
Type
Size
IOPS
Throughput
Snapshot ID
Created
Availability Zone
vol-0d37c482f6df19911
gp2
8 GiB
100
snap-091ad9e...
2024/12/03 14:38 GMT-8
us-east-1a
vol-085357a7c854c28d4
us-east-1b
Volume IDs: vol-0d37c482f6df19911, vol-085357a7c854c28d4
Read throughput (KiB/s)
Write throughput (KiB/s)
Read operations (Ops/s)
Write operations (Ops/s)
Average queue length (Oper...
Time spent idle (%)
Average read size (KiB/op)
Average write size (KiB/op)
No unit
Count
22:40
23:40

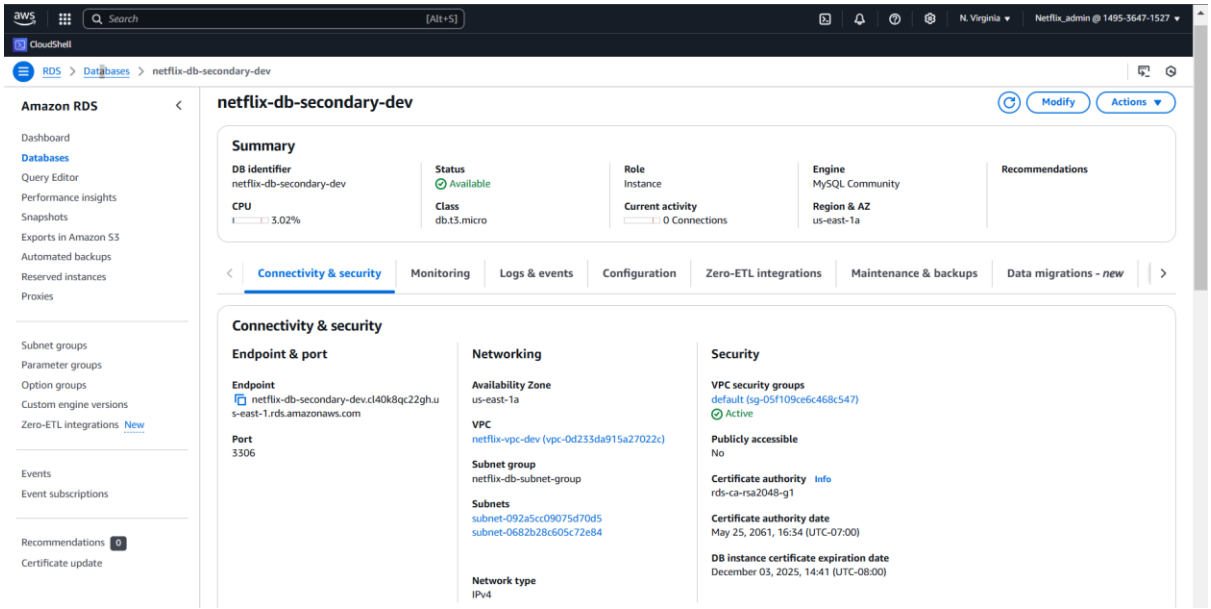
Network Interface:

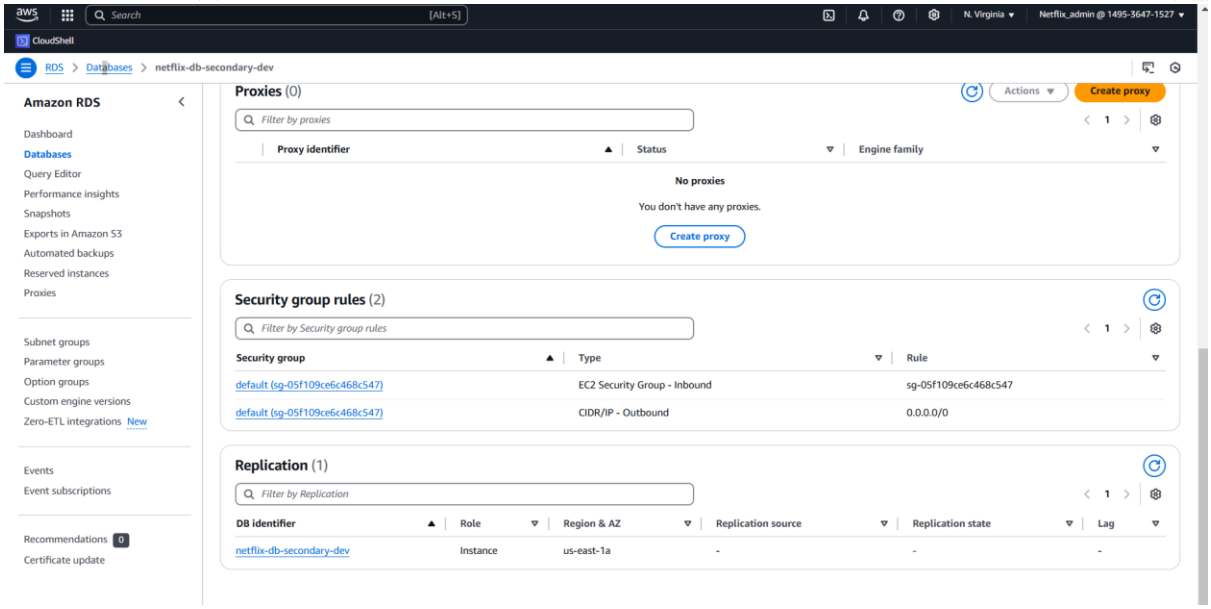


DB Instance:

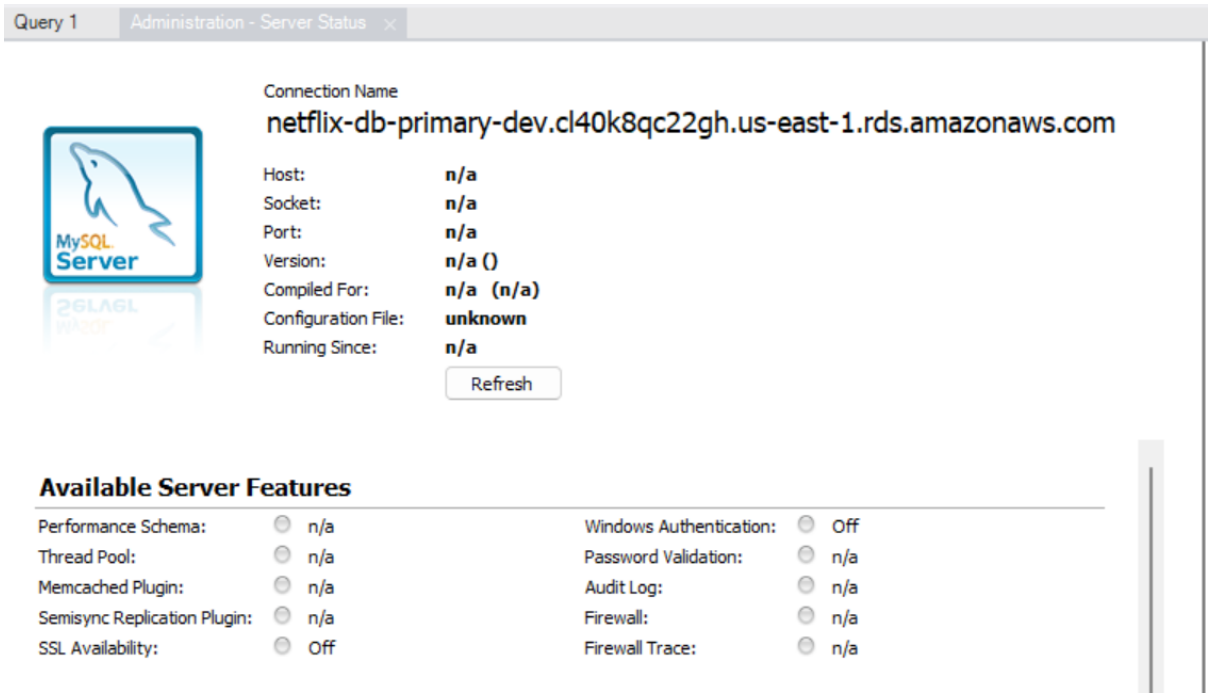

RDS > Databases > netflix-db-primary-dev
Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
netflix-db-primary-dev
Modify
Actions
Summary
DB identifier
netflix-db-primary-dev
Status
Available
Role
Instance
Engine
MySQL Community
Recommendations
CPU
3.11%
Class
db.t3.micro
Current activity
0 Connections
Region & AZ
us-east-1b
Connectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Data migrations - new
Connectivity & security
Endpoint & port
Endpoint
netflix-db-primary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com
Port
3306
Networking
Availability Zone
us-east-1b
VPC
netflix-vpc-dev (vpc-0d233da915a27022c)
Subnet group
netflix-db-subnet-group
Subnets
subnet-092a5cc09075d70d5
subnet-0682b28c605c72e84
Network type
Security
VPC security groups
default (sg-05f109ce6c468c547)
Active
Publicly accessible
No
Certificate authority Info
rds-ca-rsa2048-g1
Certificate authority date
May 25, 2061, 16:34 (UTC-07:00)
DB instance certificate expiration date
December 03, 2025, 14:40 (UTC-08:00)

aws
Search
[Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
RDS > Databases > netflix-db-primary-dev
Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
Proxies (0)
Actions
Create proxy
Filter by proxies
Proxy identifier
Status
Engine family
No proxies
You don't have any proxies.
Create proxy
Security group rules (2)
Filter by Security group rules
Security group
Type
Rule
default (sg-05f109ce6c468c547)
EC2 Security Group - Inbound
sg-05f109ce6c468c547
default (sg-05f109ce6c468c547)
CIDR/IP - Outbound
0.0.0.0/0
Replication (1)
Filter by Replication
DB identifier
Role
Region & AZ
Replication source
Replication state
Lag
netflix-db-primary-dev
Instance
us-east-1b

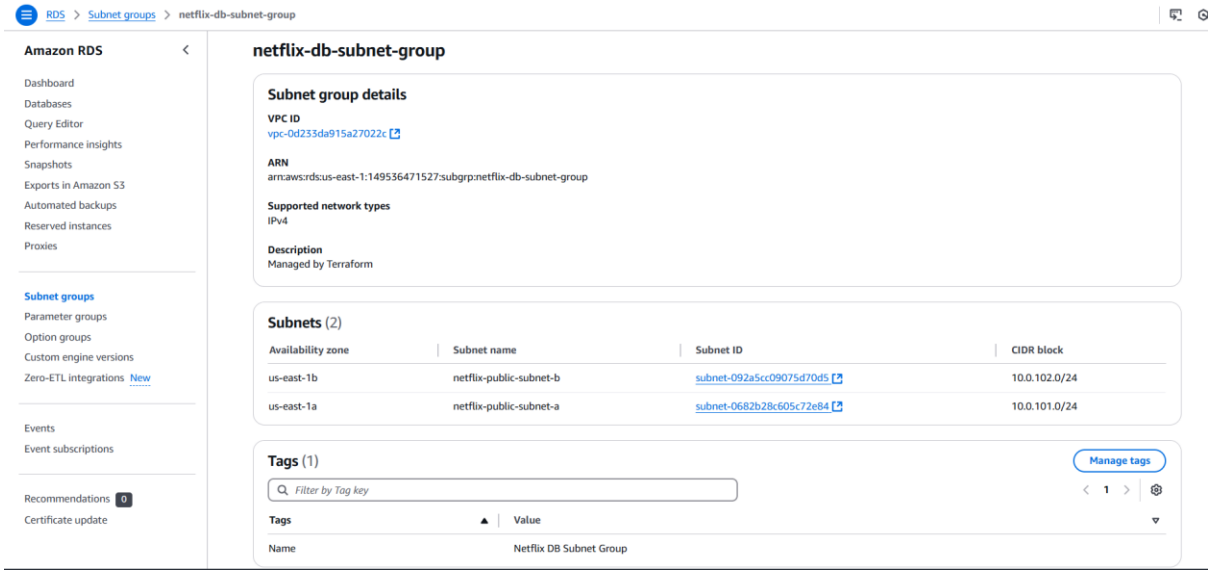

Search [Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
RDS > Databases > netflix-db-secondary-dev
Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
netflix-db-secondary-dev
Modify
Actions
Summary
DB identifier
netflix-db-secondary-dev
Status
Available
Role
Instance
Engine
MySQL Community
Recommendations
CPU
3.02%
Class
db.t3.micro
Current activity
0 Connections
Region & AZ
us-east-1a
Connectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Data migrations - new
Connectivity & security
Endpoint & port
Endpoint
netflix-db-secondary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com
Port
3306
Networking
Availability Zone
us-east-1a
VPC
netflix-vpc-dev (vpc-0d233da915a27022c)
Subnet group
netflix-db-subnet-group
Subnets
subnet-092a5cc09075d70d5
subnet-0682b28c605c72e84
Network type
IPv4
Security
VPC security groups
default (sg-05f109ce6c468c547)
Active
Publicly accessible
No
Certificate authority Info
rds-ca-rsa2048-g1
Certificate authority date
May 25, 2061, 16:34 (UTC-07:00)
DB instance certificate expiration date
December 03, 2025, 14:41 (UTC-08:00)


Search [Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
RDS > Databases > netflix-db-secondary-dev
Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
Proxies (0)
Actions
Create proxy
Filter by proxies
Proxy identifier
Status
Engine family
No proxies
You don't have any proxies.
Create proxy
Security group rules (2)
Filter by Security group rules
Security group
Type
Rule
default (sg-05f109ce6c468c547)
EC2 Security Group - Inbound
sg-05f109ce6c468c547
default (sg-05f109ce6c468c547)
CIDR/IP - Outbound
0.0.0.0/0
Replication (1)
Filter by Replication
DB identifier
Role
Region & AZ
Replication source
Replication state
Lag
netflix-db-secondary-dev
Instance
us-east-1a

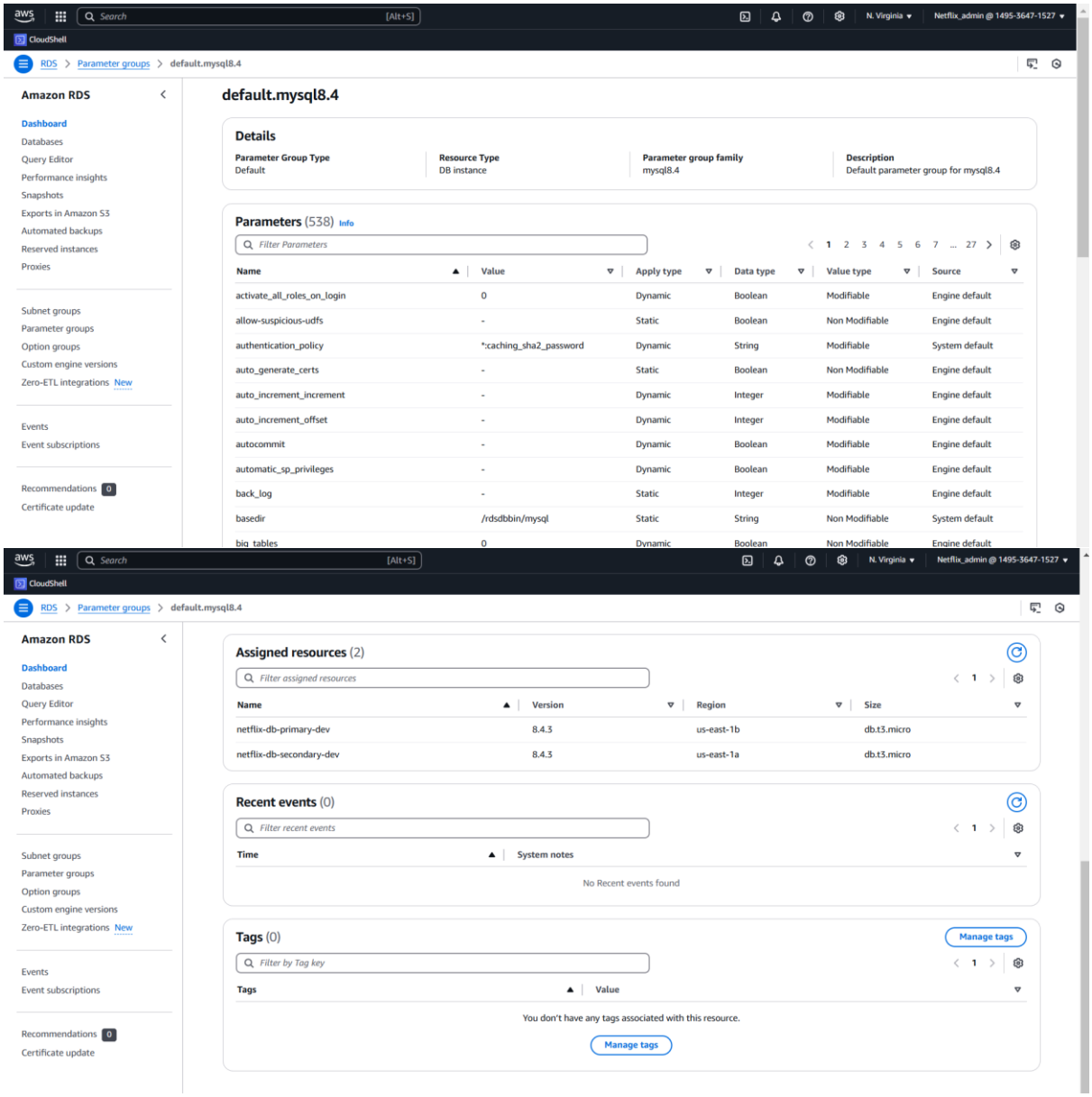
Testing it on MySQL:



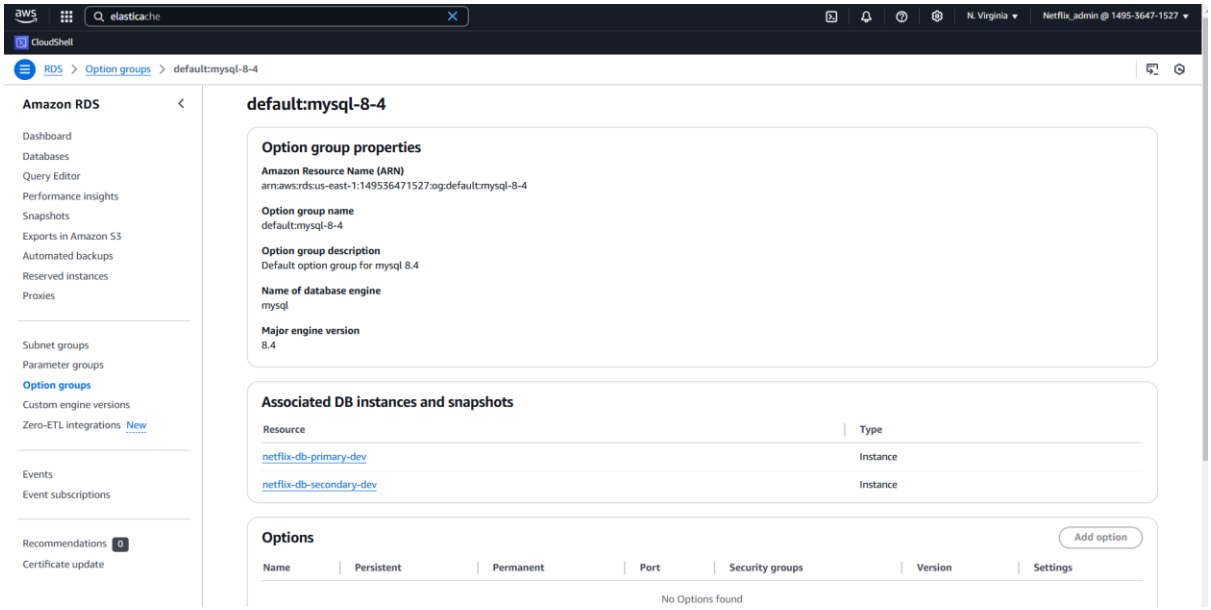
Subnet group:



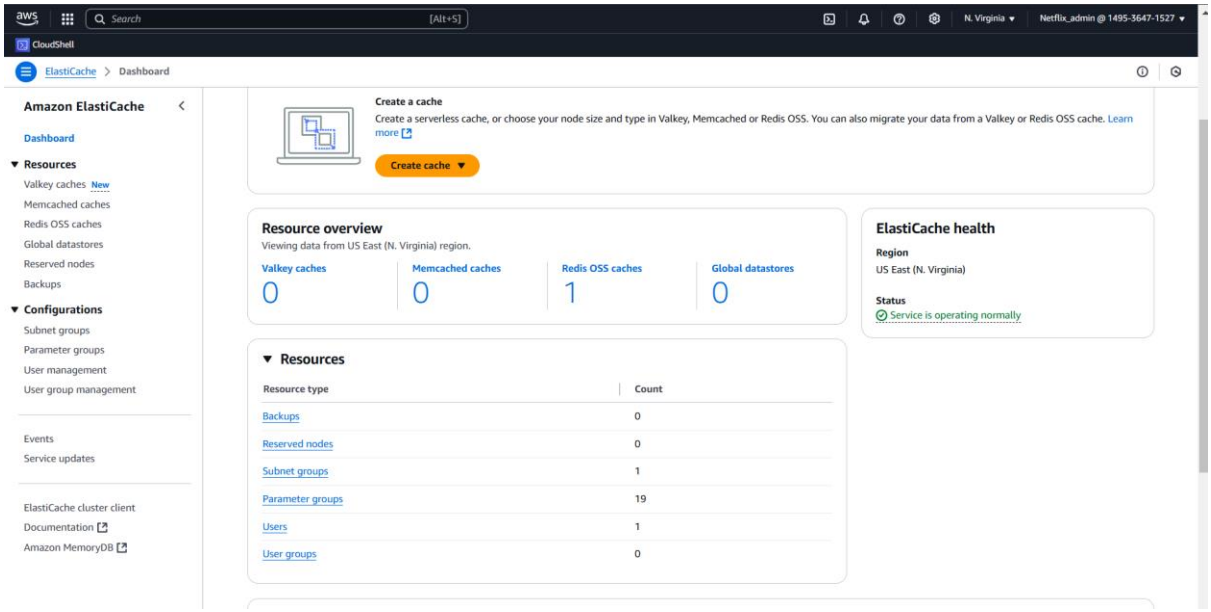
Parameter groups:



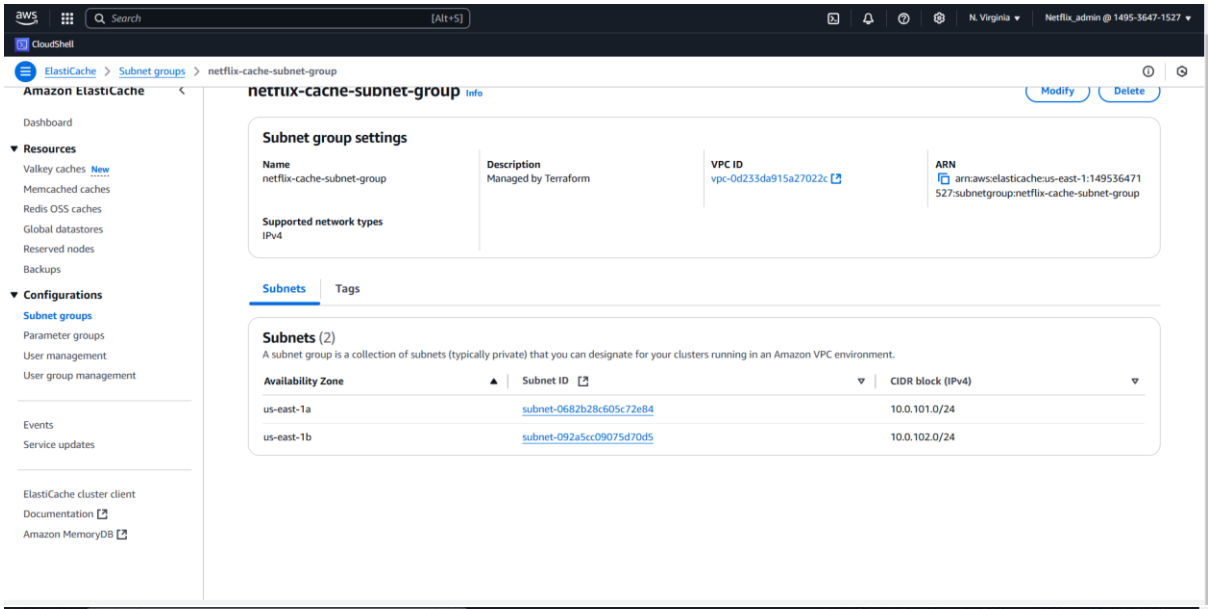
Option Groups:

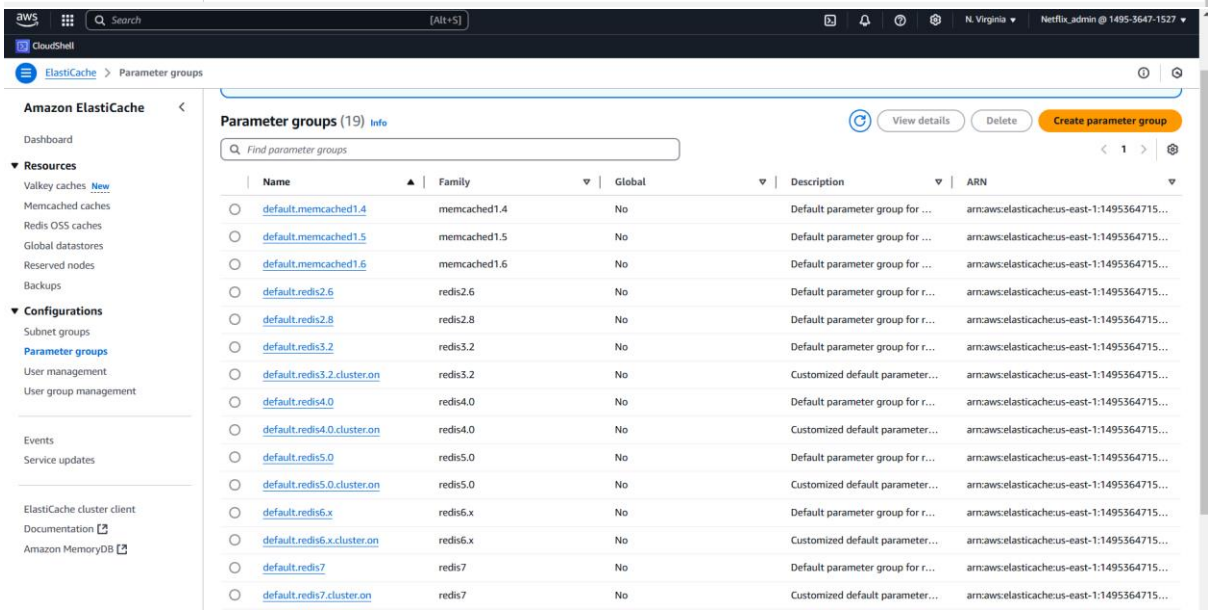


Elasticache:

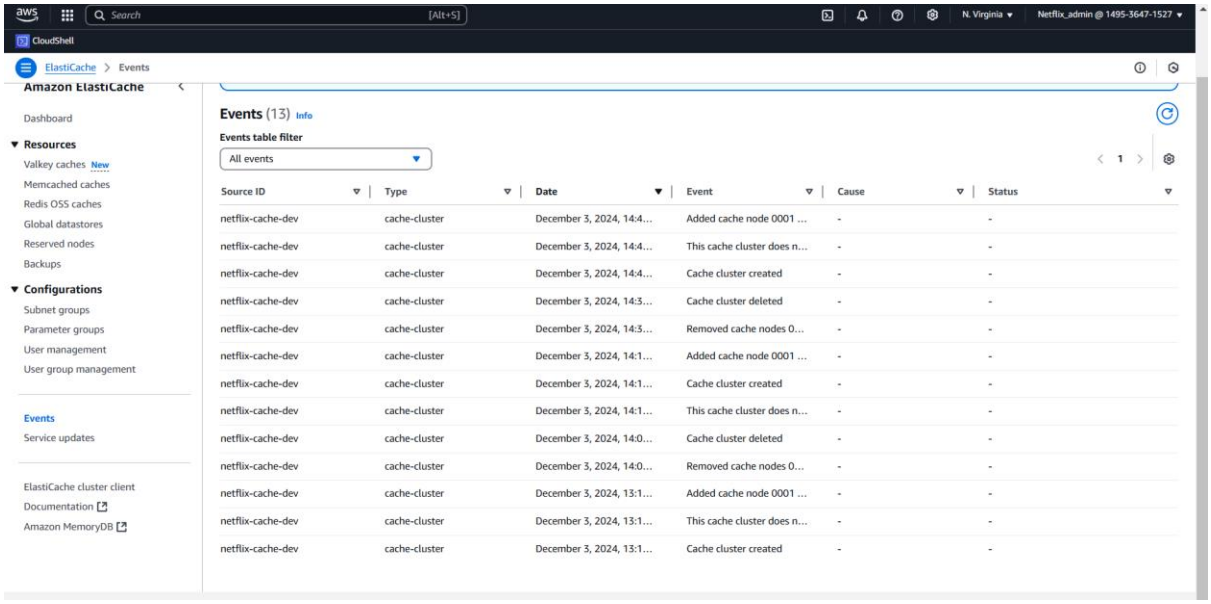


Subnet group:

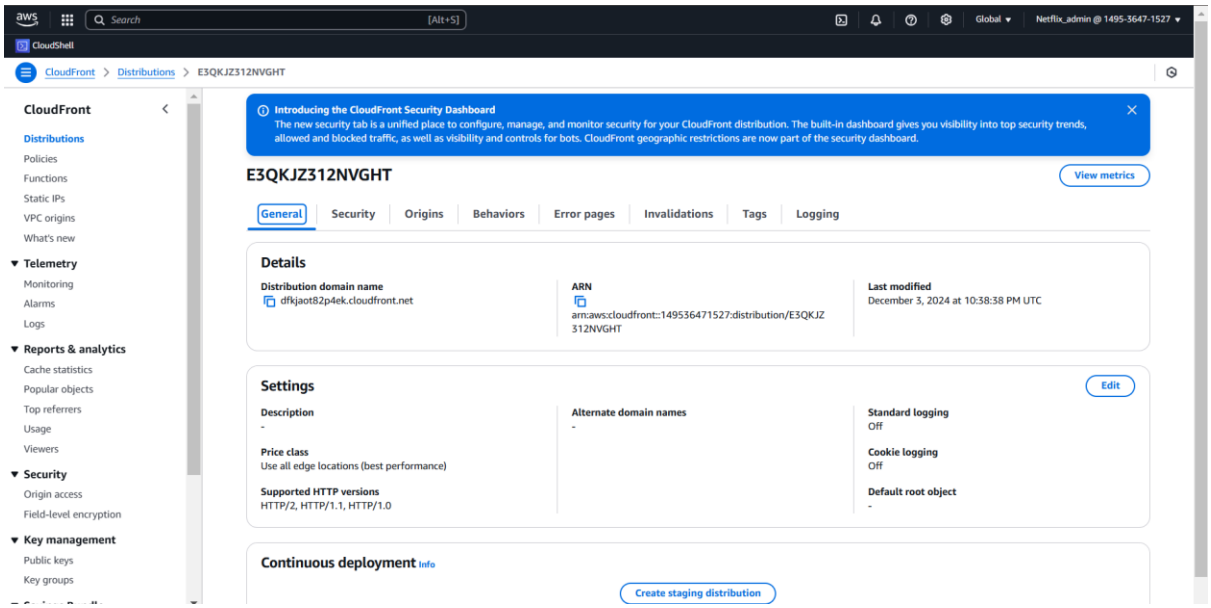




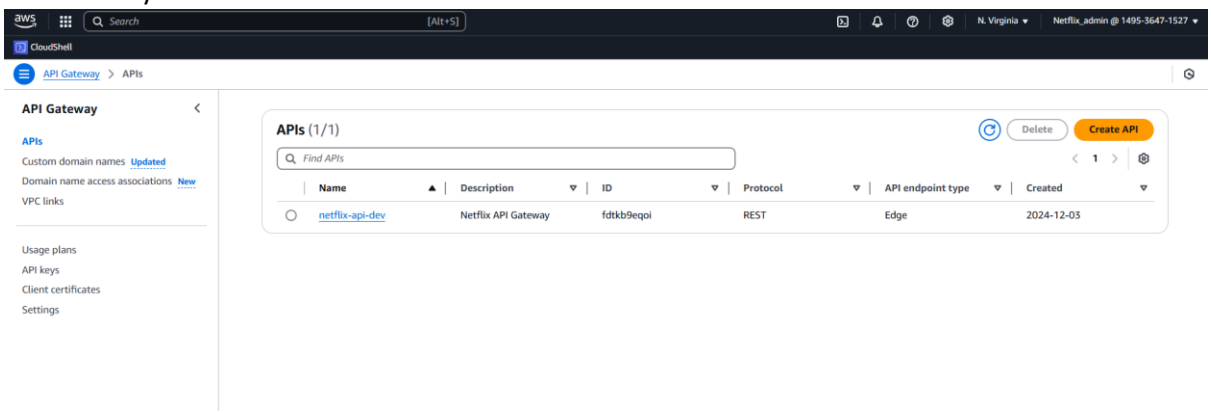
Events:

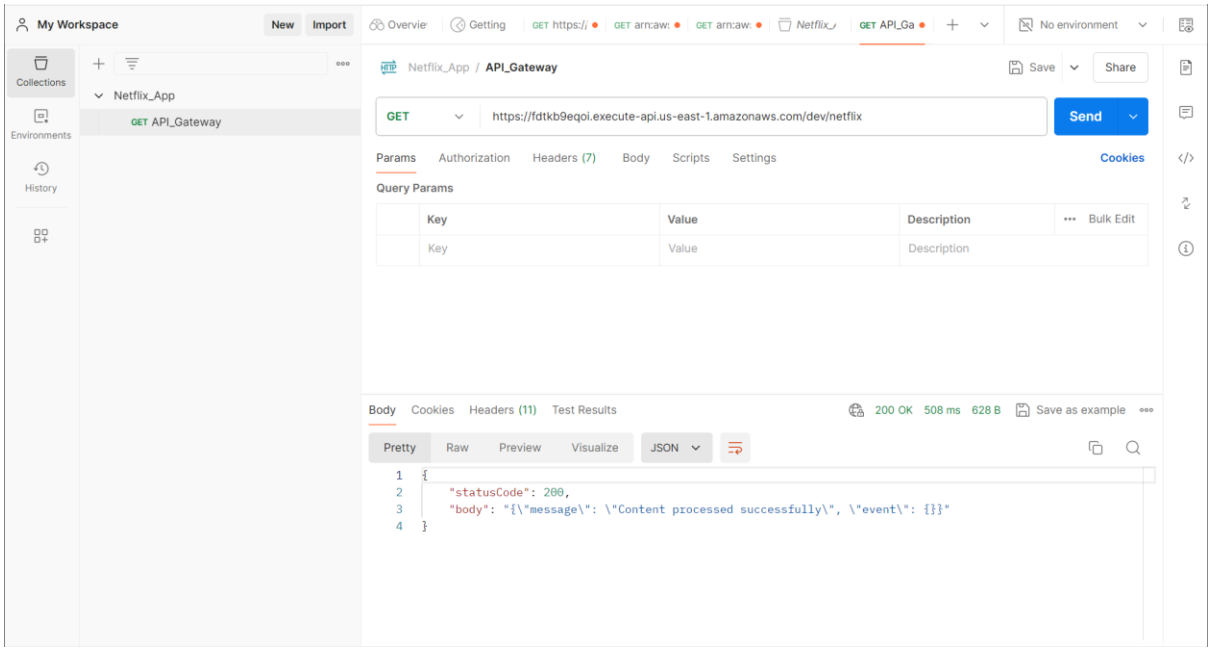


CloudFront:

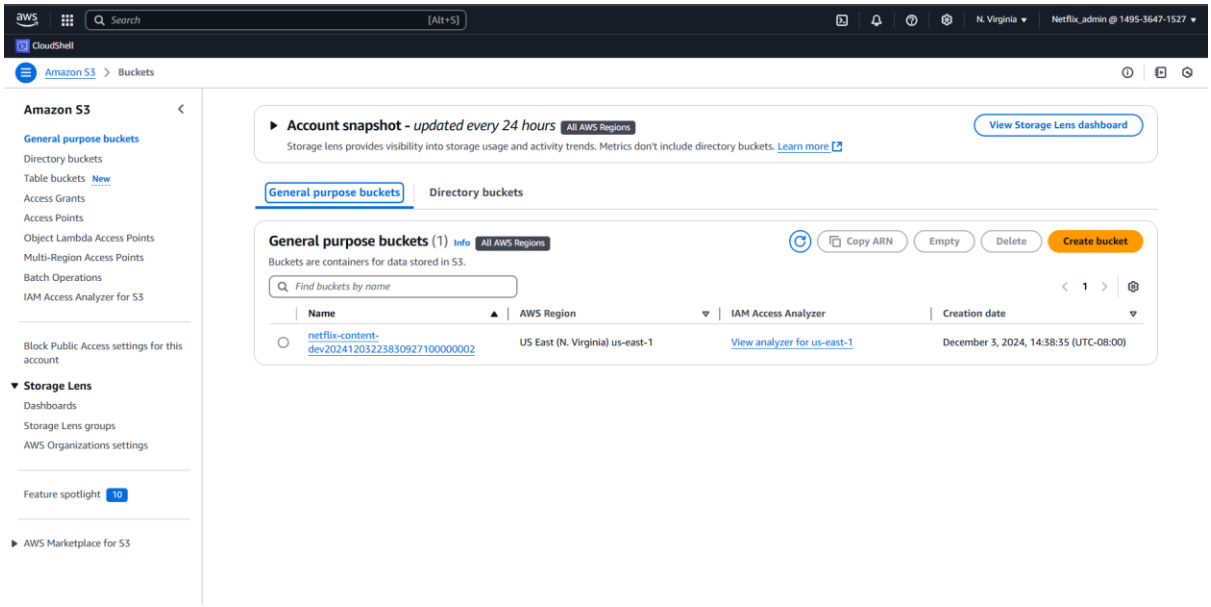


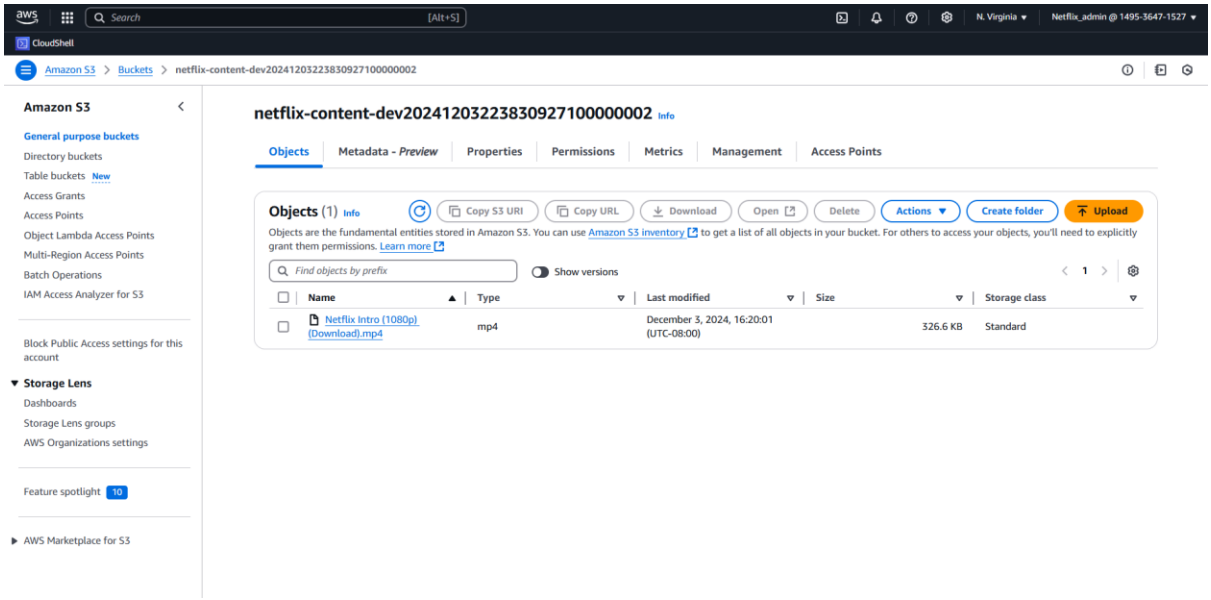
API Gateway:

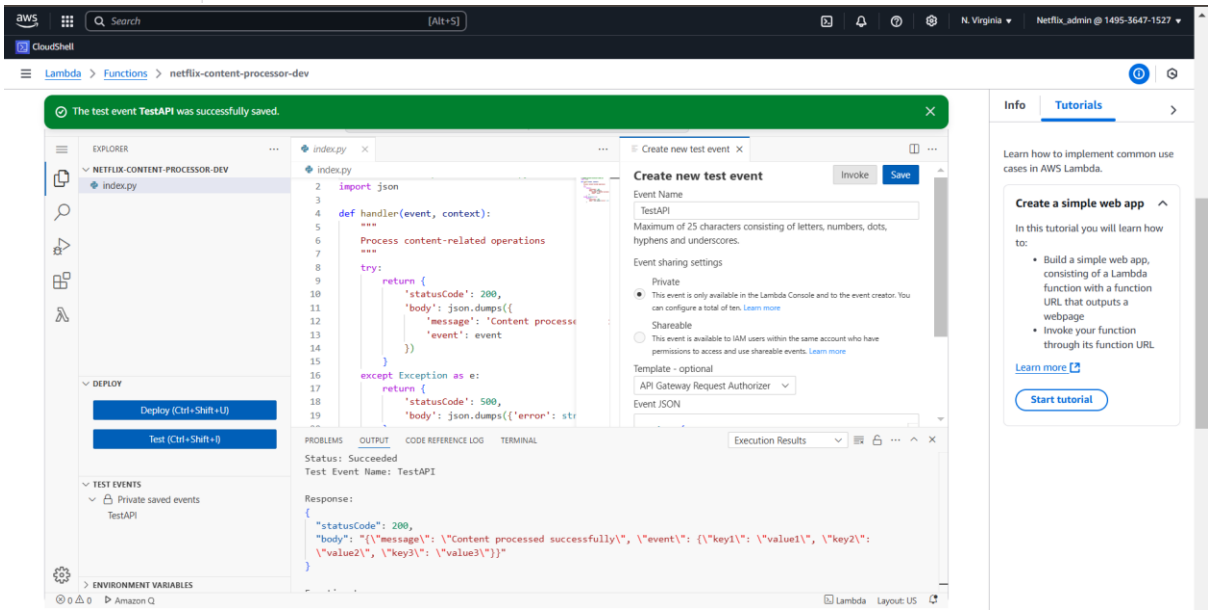



My Workspace
New
Import
Collections
Environments
History
Netflix_App
GET API_Gateway
Netflix_App / API_Gateway
Save
Share
GET
https://fdtkb9eqoi.execute-api.us-east-1.amazonaws.com/dev/netflix
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (11)
Test Results
200 OK
508 ms
628 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"statusCode": 200,
"body": "{\"message\": \"Content processed successfully\", \"event\": {}}"
}

S3 Bucket:


Search
[Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
Amazon S3
Buckets
Amazon S3
General purpose buckets
Directory buckets
Table buckets New
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 10
AWS Marketplace for S3
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (1) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
netflix-content-dev20241203223830927100000002
US East (N. Virginia) us-east-1
View analyzer for us-east-1
December 3, 2024, 14:38:35 (UTC-08:00)


Search [Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
Amazon S3 > Buckets > netflix-content-dev20241203223830927100000002
Amazon S3
General purpose buckets
Directory buckets
Table buckets New
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 10
AWS Marketplace for S3
netflix-content-dev20241203223830927100000002 Info
Objects
Metadata - Preview
Properties
Permissions
Metrics
Management
Access Points
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
Name
Type
Last modified
Size
Storage class
Netflix intro (1080p) (Download).mp4
mp4
December 3, 2024, 16:20:01 (UTC-08:00)
326.6 KB
Standard


Search [Alt+S]
N. Virginia
Netflix_admin @ 1495-3647-1527
CloudShell
Lambda > Functions > netflix-content-processor-dev
The test event TestAPI was successfully saved.
EXPLORER
NETFLIX-CONTENT-PROCESSOR-DEV
index.py
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS
Private saved events
TestAPI
ENVIRONMENT VARIABLES
import json
def handler(event, context):
"""
Process content-related operations
"""
try:
return {
'statusCode': 200,
'body': json.dumps({
'message': 'Content processe
'event': event
})
}
except Exception as e:
return {
'statusCode': 500,
'body': json.dumps({'error': str
Create new test event
Invoke
Save
Event Name
TestAPI
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
Event sharing settings
Private
This event is only available in the Lambda Console and to the event creator. You can configure a total of ten. Learn more
Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more
Template - optional
API Gateway Request Authorizer
Event JSON
PROBLEMS
OUTPUT
CODE REFERENCE LOG
TERMINAL
Execution Results
Status: Succeeded
Test Event Name: TestAPI
Response:
{
"statusCode": 200,
"body": "{\"message\": \"Content processed successfully\", \"event\": {\"key1\": \"value1\", \"key2\": \"value2\", \"key3\": \"value3\"}}"
}
Info
Tutorials
Learn how to implement common use cases in AWS Lambda.
Create a simple web app
In this tutorial you will learn how to:
Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
Invoke your function through its function URL
Learn more
Start tutorial