

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

## Netflix: Video Streaming Service Design Document

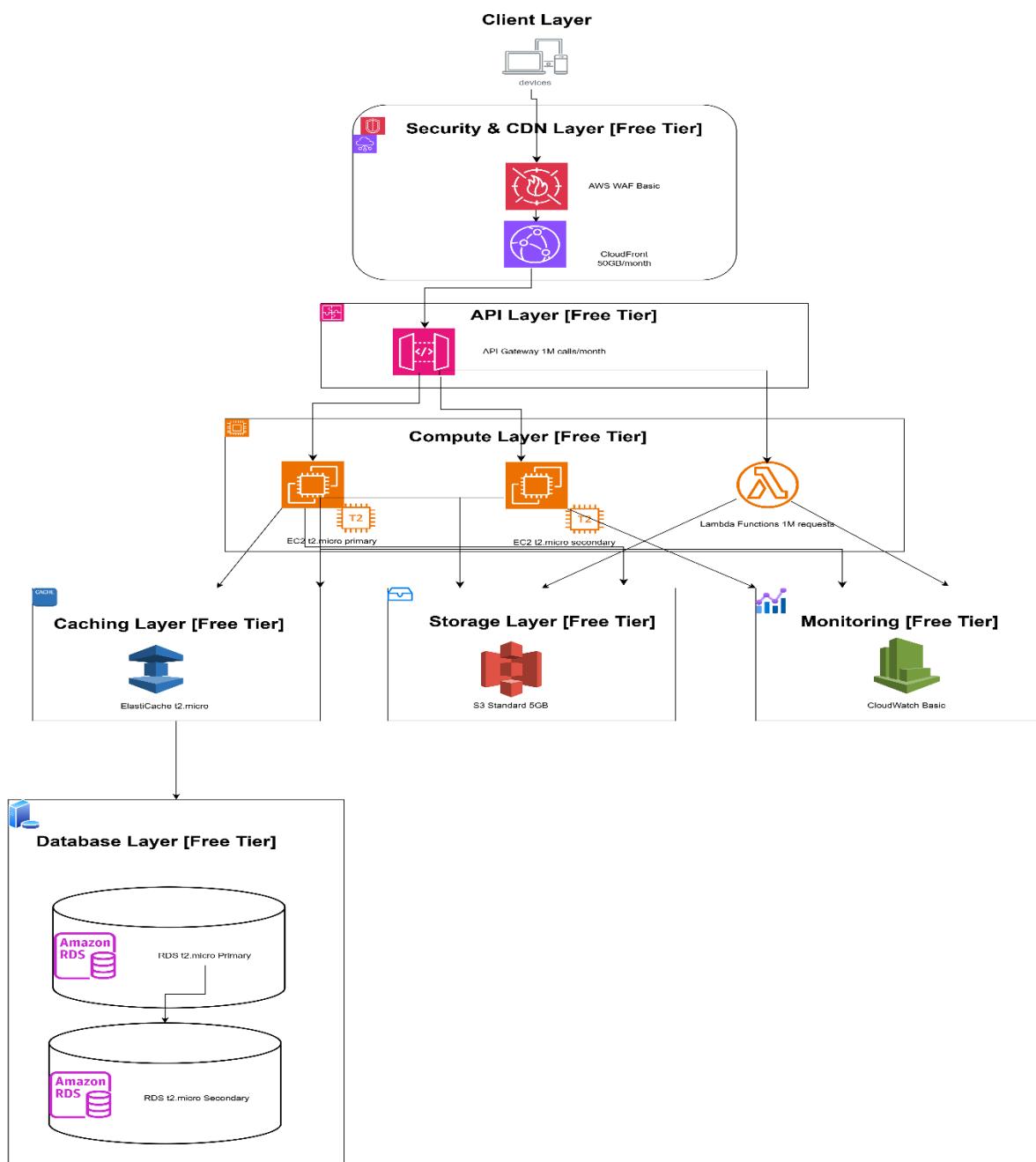
### 1. Service Overview :

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%22UvJI\\_OaGWNZtmt75ACKXv%22%7D](https://app.diagrams.net/#G11lrgf8yCwjZKjnztC0zPCS8N2kh7fT1P%7B%22pageId%22%3A%22UvJI_OaGWNZtmt75ACKXv%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:  $1,000 \text{ users} \times 555 \text{ requests} = 555,000$
  - Writes:  $1,000 \text{ users} \times 3 \text{ actions} = 3,000$
- Request sizes (optimized):
  - Read: 2MB (video), 5KB (metadata)
  - Write: 1KB (user actions)
- Storage requirements (within 5GB S3 free tier):
  - Initial:  $500 \text{ movies} \times 8\text{MB} = 4\text{GB}$
  - User data:  $1,000 \text{ users} \times 500\text{KB} = 500\text{MB}$
- 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:

The screenshot shows the Netflix mobile application interface. At the top, there is a navigation bar with links for Home, TV Shows, Movies, Originals, Recently Added, and Portfolio. To the right of the navigation bar are a search bar, a magnifying glass icon, a bell icon, and links for Register and Login.

The main content area displays two sections of recommended content:

- Action**: Includes thumbnails for "The Road trick", "Wynonna", "The Ballad of Hugo Sanchez", "Grey's anatomy", "Step Up 2", and "Liquid Science".
- Adventure**: Includes thumbnails for "LOST IN SPACE", "Queen of the South", "Undercover Boss", "Penny Dreadful", "Money heist", and "The Week of".

Below the content sections, there is a registration form titled "Home". The form fields are:

- First name:
- Last name:
- Email Address:
- Password:
- Password confirmation:

At the bottom left of the form is a "Register" button, and at the top right is a "Login" link. The URL "127.0.0.1:8000/register" is visible in the browser's address bar.



Home

Email Address:

amrutha.junnuri@sjsu.edu

Password:

.....

Login



Home TV Shows Movies Originals Recently Added Portfolio

Search



amrutha.junnuri@sjsu.edu



### Action



The Road trick



Wynonna



The Ballad of Hugo  
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



Home TV Shows Movies Originals Recently Added Portfolio

Q amrutha.junnuri@sjtu.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
1 provider "aws" {
2   region = var.aws_region
3 }
4
5 # VPC Configuration
6 resource "aws_vpc" "netflix_vpc" {
7   cidr_block        = "10.0.0.0/16"
8   enable_dns_hostnames = true
9   enable_dns_support    = true
10
11  tags = {
12    Name = "netflix-vpc-${var.environment}"
13  }
14}
15
16 # Create a public subnet
17 resource "aws_subnet" "public_a" {
18   vpc_id          = aws_vpc.netflix_vpc.id
19   cidr_block      = "10.0.101.0/24"
20   availability_zone = "${var.aws_region}a"
21
22   tags = {
23     Name = "netflix-public-subnet-a"
24   }
25 }
26
27 resource "aws_subnet" "public_b" {
28   vpc_id          = aws_vpc.netflix_vpc.id
29   cidr_block      = "10.0.102.0/24"
30   availability_zone = "${var.aws_region}b"
31
32   tags = {
33     Name = "netflix-public-subnet-b"
```

```
infrastructure > 🐾 main.tf > 📁 resource "aws_vpc" "netflix_vpc" > ⚒ cidr_block
37 # Create a DB subnet group
38 resource "aws_db_subnet_group" "netflix_db_subnet_group" {
39   name        = "netflix-db-subnet-group"
40   subnet_ids = [aws_subnet.public_a.id, aws_subnet.public_b.id]
41
42   tags = {
43     Name = "Netflix DB Subnet Group"
44   }
45 }
46
47 # Create an ElastiCache subnet group
48 resource "aws_elasticache_subnet_group" "netflix_cache_subnet_group" {
49   name        = "netflix-cache-subnet-group"
50   subnet_ids = [aws_subnet.public_a.id, aws_subnet.public_b.id]
51 }
52
53 # Update the RDS instances to use the subnet group
54 resource "aws_db_instance" "primary" {
55   identifier      = "netflix-db-primary-${var.environment}"
56   engine          = "mysql"
57   engine_version  = "8.4.3" # Use a supported version
58   instance_class  = "db.t3.micro" # Compatible instance class
59   allocated_storage = 20
60   username        = "admin"
61   password        = "temppassword123" # Change this in production!
62   skip_final_snapshot = true
63   db_subnet_group_name = aws_db_subnet_group.netflix_db_subnet_group.name
64 }
65
66 resource "aws_db_instance" "secondary" {
67   identifier      = "netflix-db-secondary-${var.environment}"
68   engine          = "mysql"
69   engine_version  = "8.4.3" # Use a supported version
70 }
```

```
infrastructure > 🌐 main.tf > 🏫 resource "aws_vpc" "netflix_vpc" > 📁 cidr_block
66 resource "aws_db_instance" "secondary" {
67   identifier      = "netflix-db-secondary-${var.environment}"
68   engine          = "mysql"
69   engine_version  = "8.4.3" # Use a supported version
70   instance_class  = "db.t3.micro" # Compatible instance class
71   allocated_storage = 20
72   username        = "admin"
73   password        = "temppassword123" # Change this in production!
74   skip_final_snapshot = true
75   db_subnet_group_name = aws_db_subnet_group.netflix_db_subnet_group.name
76 }
77
78 # Update the ElastiCache cluster to use the subnet group
79 resource "aws_elasticache_cluster" "netflix_cache" {
80   cluster_id      = "netflix-cache-${var.environment}"
81   engine          = "redis"
82   node_type       = "cache.t2.micro"
83   num_cache_nodes = 1
84   port            = 6379
85   subnet_group_name = aws_elasticache_subnet_group.netflix_cache_subnet_group.name
86 }
87
88 # Security Layer
89 resource "aws_waf_ipset" "ip_blacklist" {
90   name = "netflix-ip-blacklist-${var.environment}"
91 }
92
93 resource "aws_waf_rule" "ip_rate_limit" {
94   name      = "netflix-ip-rate-limit-${var.environment}"
95   metric_name = "netflixIpRateLimit${var.environment}"
96
97   predicates {
98     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 {
  98       data_id = aws_waf_ipset.ip_blacklist.id
  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   }
```

```
-->
130   origin {
131     domain_name = aws_s3_bucket.content.bucketRegionalDomainName
132     origin_id   = "S3Origin"
133
134     s3_origin_config {
135       origin_access_identity = aws_cloudfront_origin_access_identity.oai.cloudfrontAccessIdentityPath
136     }
137   }
138
139   default_cache_behavior {
140     allowed_methods      = ["GET", "HEAD"]
141     cached_methods        = ["GET", "HEAD"]
142     target_origin_id      = "S3Origin"
143     viewer_protocol_policy = "redirect-to-https"
144
145     forwarded_values {
146       query_string = false
147       cookies {
148         forward = "none"
149       }
150     }
151   }
152
153   restrictions {
154     geo_restriction {
155       restriction_type = "none"
156     }
157   }
158
159   viewer_certificate {
160     cloudfrontDefaultCertificate = true
161 }
```

```
163
164 # API Layer
165 resource "aws_api_gateway_rest_api" "netflix_api" {
166   name      = "netflix-api-${var.environment}"
167   description = "Netflix API Gateway"
168 }
169
170 # Compute Layer
171 resource "aws_instance" "ec2_primary" {
172   ami          = "ami-0c7217cdde317cfec"  # Amazon Linux 2023
173   instance_type = "t2.micro"
174   subnet_id    = aws_subnet.public_a.id
175   associate_public_ip_address = true
176
177   user_data = <<-EOF
178     | #!/bin/bash
179     | yum update -y
180     | yum install -y python3 python3-pip git
181     | pip3 install django==4.0.2 Pillow==9.0.1
182     | EOF
183
184   tags = {
185     Name = "netflix-primary-${var.environment}"
186   }
187 }
188
189 resource "aws_instance" "ec2_secondary" {
190   ami          = "ami-0c7217cdde317cfec"
191   instance_type = "t2.micro"
192   subnet_id    = aws_subnet.public_b.id
193   associate_public_ip_address = true
194
195   tags = {
196     Name = "netflix-secondary-${var.environment}"
```

```
200 # Lambda Functions
201 resource "aws_iam_role" "lambda_role" {
202     name = "netflix_lambda_role_${var.environment}"
203
204     assume_role_policy = jsonencode({
205         Version = "2012-10-17"
206         Statement = [
207             Action = "sts:AssumeRole"
208             Effect = "Allow"
209             Principal = {
210                 Service = "lambda.amazonaws.com"
211             }
212         ]
213     })
214 }
215
216 resource "aws_lambda_function" "content_processor" {
217     filename      = data.archive_file.content_processor.output_path
218     function_name = "netflix-content-processor-${var.environment}"
219     role          = aws_iam_role.lambda_role.arn
220     handler       = "index.handler"
221     runtime        = "python3.9"
222
223     environment {
224         variables = {
225             ENVIRONMENT = var.environment
226         }
227     }
228 }
229
230 # Storage Layer
231 resource "aws_s3_bucket" "content" {
232     bucket_prefix = "netflix-content-${var.environment}"
233 }
```

```
infrastructure > 🐾 main.tf > 📁 resource "aws_vpc" "netflix_vpc" > ✎ cidr_block
228 }
229
230 # Storage Layer
231 resource "aws_s3_bucket" "content" {
232   bucket_prefix = "netflix-content-${var.environment}"
233 }
234
235 # Monitoring
236 resource "aws_cloudwatch_metric_alarm" "cpu_alarm" {
237   alarm_name          = "netflix-cpu-alarm-${var.environment}"
238   comparison_operator = "GreaterThanThreshold"
239   evaluation_periods  = "2"
240   metric_name         = "CPUUtilization"
241   namespace           = "AWS/EC2"
242   period               = "120"
243   statistic            = "Average"
244   threshold            = "80"
245   alarm_description    = "This metric monitors EC2 CPU utilization"
246   alarm_actions        = []
247 }
248
249 # Data source for Lambda ZIP files
250 data "archive_file" "content_processor" {
251   type      = "zip"
252   source_dir = "${path.module}/lambda/content-processor"
253   output_path = "${path.module}/content-processor.zip"
254 }
```

### Outputs.tf:

```
infrastructure > 🏛 outputs.tf > 📄 output "api_gateway_url" > ⏷ value
  3  # API Gateway URL
  4  output "api_gateway_url" {
  5    value      = aws_api_gateway_rest_api.netflix_api.execution_arn
  6    description = "URL of the API Gateway endpoint"
  7  }
  8
  9  # CloudFront Domain
 10 output "cloudfront_domain" {
 11   value      = aws_cloudfront_distribution.netflix_cdn.domain_name
 12   description = "Domain name of the CloudFront distribution"
 13 }
 14
 15 # Primary EC2 Instance Public IP
 16 output "primary_ec2_public_ip" {
 17   value      = aws_instance.ec2_primary.public_ip
 18   description = "Public IP of the primary EC2 instance"
 19 }
 20
 21 # Secondary EC2 Instance Public IP
 22 output "secondary_ec2_public_ip" {
 23   value      = aws_instance.ec2_secondary.public_ip
 24   description = "Public IP of the secondary EC2 instance"
 25 }
 26
 27 # RDS Primary Endpoint
 28 output "rds_primary_endpoint" {
 29   value      = aws_db_instance.primary.endpoint
 30   description = "Endpoint of the primary RDS instance"
 31 }
 32
 33 # RDS Secondary Endpoint
 34 output "rds_secondary_endpoint" {
 35   value      = aws_db_instance.secondary.endpoint
 36   description = "Endpoint of the secondary RDS instance"
```

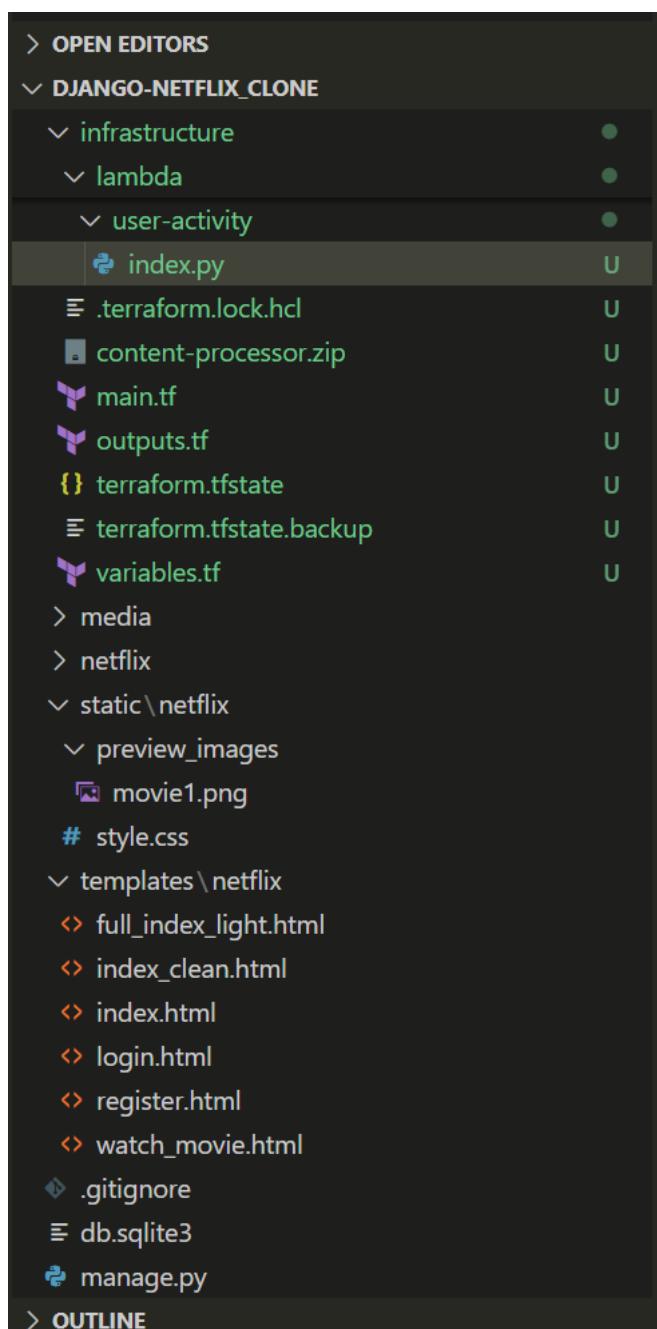
### Variables.tf:

```
/ ...\\user-activity U | 🏛 main.tf U | {} terraform.tfstate U | ⚡ index.h
infrastructure > 🏛 variables.tf > 📄 variable "environment"
  1  # variables.tf
  2  variable "aws_region" {
  3    description = "AWS region"
  4    default     = "us-east-1"
  5  }
  6
  7  variable "environment" {
  8    description = "Environment name"
  9    default     = "dev"
 10 }
```

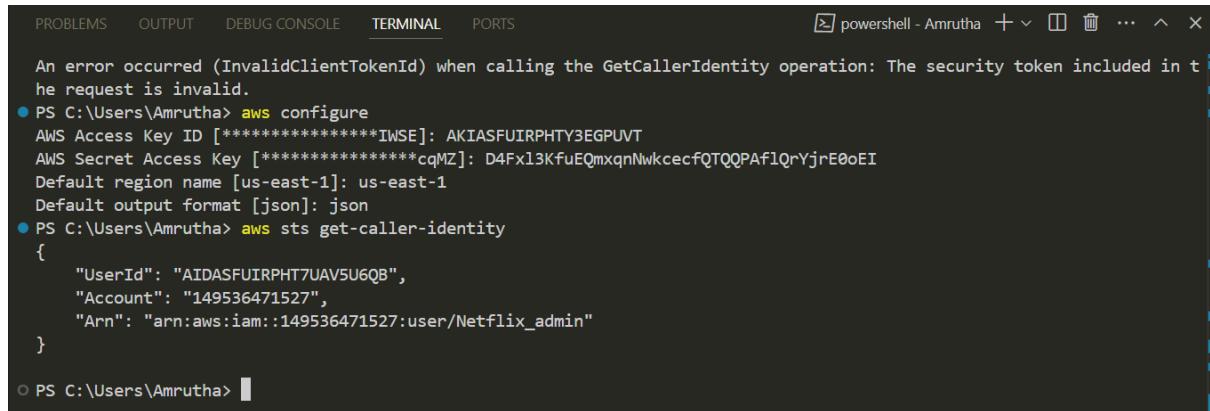
## Netflix project:

```
manage.py > ...
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks."""
3  import os
4  import sys
5
6
7  def main():
8      """Run administrative tasks."""
9      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'django_netflix_clone.settings')
10     try:
11         from django.core.management import execute_from_command_line
12     except ImportError as exc:
13         raise ImportError(
14             "Couldn't import Django. Are you sure it's installed and "
15             "available on your PYTHONPATH environment variable? Did you "
16             "forget to activate a virtual environment?"
17         ) from exc
18     execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
```

Overall project folders:



## Configure your AWS IAM Account:



A screenshot of a terminal window titled "powershell - Amrutha". The window shows the following command history:

- PS C:\Users\Amrutha> **aws configure**  
AWS Access Key ID [\*\*\*\*\*IWSE]: AKIASFUIRPHTY3EGPUVT  
AWS Secret Access Key [\*\*\*\*\*cqMZ]: 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  
Default region name [us-east-1]: us-east-1  
Default output format [json]: json
- PS C:\Users\Amrutha> **aws sts get-caller-identity**  
{  
    "UserId": "AIDASFUIRPH7UAV5U6QB",  
    "Account": "149536471527",  
    "Arn": "arn:aws:iam::149536471527:user/Netflix\_admin"  
}
- PS C:\Users\Amrutha>

## Terraform:

### Terraform init:

The screenshot shows the Visual Studio Code interface with the Terraform extension installed. The left sidebar displays the project structure under 'OPEN EDITORS' and 'EXPLORER'. The main editor area shows the 'main.tf' file with Terraform code. The terminal at the bottom shows the output of the 'terraform init' command, which includes initializing the backend, provider plugins, and creating a lock file. It concludes with a message: 'Terraform has been successfully initialized!'. The status bar indicates the current terminal is 'powershell - infrastructure'.

```
infrastructure > main.tf > resource "aws_cloudwatch_metric_alarm" "cpu_alarm"
186 resource "aws_cloudwatch_metric_alarm" "cpu_alarm" {
194   threshold      = "80"
195   alarm_description = "This metric monitors EC2 CPU utilization"
196   alarm_actions    = []
197 }

198 # Data source for Lambda ZIP files
199 data "archive_file" "content_processor" {
200   type        = "zip"
201   source_dir  = "${path.module}/lambda/content-processor"
202   output_path = "${path.module}/content-processor.zip"
203 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Initialization the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/archive...
- Installing hashicorp/aws v5.79.0...
- Installed hashicorp/aws v5.79.0 (signed by HashiCorp)
- Installing hashicorp/archive v2.6.0...
- Installed hashicorp/archive v2.6.0 (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.
PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

### Terraform validate:

The screenshot shows a terminal window with the output of the 'terraform validate' command. It starts with a success message: 'Terraform has been successfully initialized!'. It then provides instructions for working with Terraform and reinitializing the directory. The command concludes with a success message: 'Success! The configuration is valid.' The prompt shows the command was run from the 'infrastructure' directory.

```
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:

```
4 }
5
6 # VPC Configuration
7 resource "aws_vpc" "netflix_vpc" {
8   cidr_block = "10.0.0.0/16"
9
10  tags_all  = (known after apply)
11
12  default_action {
13    + type = "ALLOW"
14  }
15
16  rules {
17    + priority = 1
18    + rule_id  = (known after apply)
19    + type     = "REGULAR"
20
21    + action {
22      + type = "BLOCK"
23    }
24  }
25}
26
27
28 Plan: 17 to add, 0 to change, 0 to destroy.
29
30 Changes to Outputs:
31  + api_gateway_url  = (known after apply)
32  + cloudfront_domain = (known after apply)
33
34
35 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.
36 PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

## Terraform apply:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - infrastructure + v x
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.c140k8qc22gh.us-east-1.rds.amazonaws.com:3306"
rds_secondary_endpoint = "netflix-db-secondary-dev.c140k8qc22gh.us-east-1.rds.amazonaws.com:3306"
secondary_ec2_public_ip = ""
PS C:\Users\Amrutha\Netflix clone proj\django-netflix_clone\infrastructure>
```

## Outputs:

### Terraform:

```

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

resource "aws_cloudfront_distribution" "netflix_cdn" {
  origin {
    id: "netflix-db-primary-dev"
    domain_name: "dfkjaot82p4ek.cloudfront.net"
    port: 443
    protocol: "SSL"
    ssl_certificate: "arn:aws:acm:us-east-1:149536471527:certificate:fdtkb9eqoi"
  }
  viewer_protocol_policy: "RedirectToHttps"
  price_class: "PriceClass_100"
  comment: "Netflix API Distribution"
}

```

```

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:

The screenshot displays two views of the AWS VPC dashboard.

**Subnets View:**

- Subnets (2) Info:** A table showing two subnets: **subnet-092a5cc09075d70d5** and **subnet-0682b28c605c72e84**. Both are in the **Available** state and belong to the **vpc-0d233da915a27022c** VPC. Their IPv4 CIDRs are **10.0.102.0/24** and **10.0.101.0/24** respectively. The **Block Public Access** setting is off for both.
- Select a subnet:** A dropdown menu where **subnet-092a5cc09075d70d5** is selected.

**Your VPCs View:**

- Your VPCs (1/1) Info:** A table showing one VPC named **netflix-vpc-dev** with VPC ID **vpc-0d233da915a27022c**. It is in the **Available** state and has an IPv4 CIDR of **10.0.0.0/16**. The **Block Public Access** setting is off. The **DHCP option set** is **dopt-0f131508445150afb**.
- vpc-0d233da915a27022c / netflix-vpc-dev:** A detailed view of the selected VPC.
- Details:** A table showing various configuration details:

VPC ID	Name	State	Block Public Access	DNS hostnames
vpc-0d233da915a27022c	netflix-vpc-dev	Available	Off	Enabled
				Main route table rtb-030e73a7b367d2927
				IPv6 pool -
				Owner ID 149536471527

## EC2:

**Instances (2/2) Info**

Last updated less than a minute ago

Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

Instance state = running

Clear filters

Instances (2/2)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Pub
netflix-second...	i-091b41886321c9698	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-	-
netflix-primer...	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)

Percent Bytes Bytes Count

Network packets out (count) Metadata no token (count) CPU credit usage (count) CPU credit balance (count)

Count No unit Count Count

CloudShell EC2 > Instances > i-0f75af65cd7dd6b90

**Instance summary for i-0f75af65cd7dd6b90 (netflix-primary-dev)**

Updated less than a minute ago

Instance ID	i-0f75af65cd7dd6b90	Public IPv4 address	Private IPv4 addresses
IPv6 address	-	Instance state	Public IPv4 DNS
Hostname type	IP name: ip-10-0-101-212.ec2.internal	Private IP DNS name (IPv4 only)	Elastic IP addresses
Answer private resource DNS name	-	ip-10-0-101-212.ec2.internal	AWS Compute Optimizer finding
Auto-assigned IP address	-	Instance type	Opt-in to AWS Compute Optimizer for recommendations.
IAM Role	-	t2.micro	Learn more
IMDsv2	Optional	VPC ID	Auto Scaling Group name
Optional	⚠ EC2 recommends setting IMDsv2 to required	vpc-0d233da915a27022c (netflix-vpc-dev)	-
Learn more		Subnet ID	Managed
Operator	-	subnet-0682b28c605c72e84 (netflix-public-subnet-a)	false
		Instance ARN	
		arn:aws:ec2:us-east-1:149536471527:instance/i-0f75af65cd7dd6b90	

Details Status and alarms Monitoring Security Networking Storage Tags

Instance details

AMI ID Monitoring Platform details

CloudShell EC2 > Instances > i-091b41886321c9698

**Instance summary for i-091b41886321c9698 (netflix-secondary-dev)**

Updated less than a minute ago

Instance ID	i-091b41886321c9698	Public IPv4 address	Private IPv4 addresses
IPv6 address	-	Instance state	Public IPv4 DNS
Hostname type	IP name: ip-10-0-102-248.ec2.internal	Private IP DNS name (IPv4 only)	Elastic IP addresses
Answer private resource DNS name	-	ip-10-0-102-248.ec2.internal	AWS Compute Optimizer finding
Auto-assigned IP address	-	Instance type	Opt-in to AWS Compute Optimizer for recommendations.
IAM Role	-	t2.micro	Learn more
IMDsv2	Optional	VPC ID	Auto Scaling Group name
Optional	⚠ EC2 recommends setting IMDsv2 to required	vpc-0d233da915a27022c (netflix-vpc-dev)	-
Learn more		Subnet ID	Managed
Operator	-	subnet-092a5cc09075d70d5 (netflix-public-subnet-b)	false
		Instance ARN	
		arn:aws:ec2:us-east-1:149536471527:instance/i-091b41886321c9698	

Details Status and alarms Monitoring Security Networking Storage Tags

Instance details

AMI ID Monitoring Platform details

## Volume:

Saved filter sets   [Alt+S]

N. Virginia | Netflix\_admin @ 1495-5647-1527

CloudShell

Actions

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability Zone
vol-0d37c482f6df19911	gp2	8 GiB	100	-	snap-091ad9e...	2024/12/03 14:58 GMT-8	us-east-1a	
vol-085357a7c854c28d4	gp2	8 GiB	100	-	snap-091ad9e...	2024/12/03 14:58 GMT-8	us-east-1b	

Volume IDs: vol-0d37c482f6df19911, vol-085357a7c854c28d4

Read throughput (KiB/s)

No unit

427

213

0

22:40 23:40

Write throughput (KiB/s)

No unit

94.6

47.3

0

22:40 23:40

Read operations (Ops/s)

No unit

10.4

5.18

0

22:40 23:40

Write operations (Ops/s)

No unit

4.81

2.4

0

22:40 23:40

Average queue length (Oper...)

Count

0.017

9e-3

0

22:40 23:40

Time spent idle (%)

No unit

100

50

0

22:40 23:40

Average read size (KiB/op)

No unit

41.2

20.6

0

22:40 23:40

Average write size (KiB/op)

No unit

19.7

9.84

0

22:40 23:40

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
- Network Interfaces

## Network Interface:

aws CloudShell Search [Alt+5] N. Virginia Netflix\_admin @ 1495-3647-1527

**Network interfaces (1/5) Info**

Name	Network interface ID	Subnet ID	VPC ID	Availability Zone	Security group n...	Security group ID:
eni-0083dd9af608f2dea	subnet-092a5cc09075d70d5	vpc-0d233da915a27022c	us-east-1b	default	sg-05f109ce6c468	
eni-044ce6668938b5053	subnet-092a5cc09075d70d5	vpc-0d233da915a27022c	us-east-1b	default	sg-05f109ce6c468	
eni-05b45bdd5161592cf	subnet-0682b28c605c72e84	vpc-0d233da915a27022c	us-east-1a	default	sg-05f109ce6c468	
eni-0079191b569f3679d	subnet-0682b28c605c72e84	vpc-0d233da915a27022c	us-east-1a	default	sg-05f109ce6c468	

**Network interface: eni-044ce6668938b5053**

**Details** Flow logs Tags

**Network interface details**

Network interface ID eni-044ce6668938b5053	Name -	Description -
Network interface status In-use	Interface type Elastic network interface	Security groups sg-05f109ce6c468c547 (default)
VPC ID vpc-0d233da915a27022c	Subnet ID subnet-092a5cc09075d70d5	Availability Zone us-east-1b
Owner 149536471527	Requester ID -	Requester-managed False
Source/dest. check True	Managed False	Operator -
<b>IP addresses</b>	Private IPv4 DNS ip-10-0-102-248.ec2.internal	
Private IPv4 address 10.0.102.248	Elastic Fabric Adapter False	

**Network interfaces (1/5) Info**

Name	Network interface ID	Subnet ID	VPC ID	Availability Zone	Security group n...	Security group ID:
eni-0083dd9af608f2dea	subnet-092a5cc09075d70d5	vpc-0d233da915a27022c	us-east-1b	default	sg-05f109ce6c468	
eni-044ce6668938b5053	subnet-092a5cc09075d70d5	vpc-0d233da915a27022c	us-east-1b	default	sg-05f109ce6c468	
eni-05b45bdd5161592cf	subnet-0682b28c605c72e84	vpc-0d233da915a27022c	us-east-1a	default	sg-05f109ce6c468	
eni-0079191b569f3679d	subnet-0682b28c605c72e84	vpc-0d233da915a27022c	us-east-1a	default	sg-05f109ce6c468	

**Network interface: eni-05b45bdd5161592cf**

**Details** Flow logs Tags

**Network interface details**

Network interface ID eni-05b45bdd5161592cf	Name -	Description -
Network interface status In-use	Interface type Elastic network interface	Security groups sg-05f109ce6c468c547 (default)
VPC ID vpc-0d233da915a27022c	Subnet ID subnet-0682b28c605c72e84	Availability Zone us-east-1a
Owner 149536471527	Requester ID -	Requester-managed False
Source/dest. check True	Managed False	Operator -
<b>IP addresses</b>	Private IPv4 DNS ip-10-0-101-212.ec2.internal	
Private IPv4 address 10.0.101.212	Public IPv4 DNS -	
Public IPv4 address -	Elastic Fabric Adapter False	
IPv6 addresses -	IPv6 addresses -	

## DB Instance:

Amazon RDS < netflix-db-primary-dev

**Summary**

<b>DB identifier</b> netflix-db-primary-dev	<b>Status</b> <span style="color: green;">Available</span>	<b>Role</b> Instance	<b>Engine</b> MySQL Community
<b>CPU</b> 3.11%	<b>Class</b> db.t3.micro	<b>Current activity</b> 0 Connections	<b>Region &amp; AZ</b> us-east-1b

**Connectivity & security**

<b>Endpoint</b> netflix-db-primary-dev.c40k8qc22gh.us-east-1.rds.amazonaws.com	<b>Networking</b>	<b>Security</b>
<b>Port</b> 3306	<b>Availability Zone</b> us-east-1b	<b>VPC security groups</b> default (sg-05f109ce6c468c547)
	<b>VPC</b> netflix-vpc-dev (vpc-0d233da915a27022c)	<b>Active</b>
	<b>Subnet group</b> netflix-db-subnet-group	<b>Publicly accessible</b> No
	<b>Subnets</b> subnet-092a5cc09075d70d5 subnet-0682b28c605c72e84	<b>Certificate authority</b> rds-ca-sa2048-q1
	<b>Network type</b> IPv4	<b>Certificate authority date</b> May 25, 2025, 16:34 (UTC-07:00)
		<b>DB instance certificate expiration date</b> December 03, 2025, 14:40 (UTC-08:00)

aws CloudShell Search [Alt+S]

RDS > Databases > netflix-db-primary-dev

**Proxies (0)**

Proxy identifier	Status	Engine family
------------------	--------	---------------

No proxies.

**Create proxy**

**Security group rules (2)**

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)**

DB identifier	Role	Region & AZ	Replication source	Replication state	Lag
netflix-db-primary-dev	Instance	us-east-1b	-	-	-

**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**

**Recommendations** 0

**Certificate update**

**CloudShell**

RDS > Databases > netflix-db-secondary-dev

## netflix-db-secondary-dev

**Summary**

<b>DB identifier</b>	netflix-db-secondary-dev	<b>Status</b>	Available
<b>CPU</b>	3.02%	<b>Role</b>	Instance
		<b>Current activity</b>	0 Connections
		<b>Engine</b>	MySQL Community
		<b>Region &amp; AZ</b>	us-east-1a

**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)

**Proxies (0)**

No proxies. You don't have any proxies.

**Create proxy**

**Security group rules (2)**

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)**

DB identifier	Role	Region & AZ	Replication source	Replication state	Lag
netflix-db-secondary-dev	Instance	us-east-1a	-	-	-

Testing it on MySQL:

Query 1 Administration - Server Status

Connection Name  
**netflix-db-primary-dev.cl40k8qc22gh.us-east-1.rds.amazonaws.com**

MySQL Server

Host: n/a  
Socket: n/a  
Port: n/a  
Version: n/a ()  
Compiled For: n/a (n/a)  
Configuration File: unknown  
Running Since: n/a

Refresh

---

### Available Server Features

Performance Schema:	n/a	Windows Authentication:	Off
Thread Pool:	n/a	Password Validation:	n/a
Memcached Plugin:	n/a	Audit Log:	n/a
Semisync Replication Plugin:	n/a	Firewall:	n/a
SSL Availability:	Off	Firewall Trace:	n/a

Subnet group:

RDS > Subnet groups > netflix-db-subnet-group

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-subnet-group**

**Subnet group details**

VPC ID: vpc-0d233da915a27022c [Edit](#)

ARN: arn:aws:rds:us-east-1:149536471527:subgrp:netflix-db-subnet-group

Supported network types: IPv4

Description: Managed by Terraform

**Subnets (2)**

Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1b	netflix-public-subnet-b	<a href="#">subnet-092a5cc09075d70d5</a>	10.0.102.0/24
us-east-1a	netflix-public-subnet-a	<a href="#">subnet-0682b28c605c72e84</a>	10.0.101.0/24

**Tags (1)**

Manage tags

Tags	Value
Name	Netflix DB Subnet Group

## Parameter groups:

Screenshot of the AWS RDS Parameter Groups page showing the 'default.mysql8.4' group.

**Details**

Parameter Group Type	Resource Type	Parameter group family	Description
Default	DB instance	mysql8.4	Default parameter group for mysql8.4

**Parameters (538) Info**

Name	Value	Apply type	Data type	Value type	Source
activate_all_roles_on_login	0	Dynamic	Boolean	Modifiable	Engine default
allow-suspicious-udfs	-	Static	Boolean	Non Modifiable	Engine default
authentication_policy	*:caching_sha2_password	Dynamic	String	Modifiable	System default
auto_generate_certs	-	Static	Boolean	Non Modifiable	Engine default
auto_increment_increment	-	Dynamic	Integer	Modifiable	Engine default
auto_increment_offset	-	Dynamic	Integer	Modifiable	Engine default
autocommit	-	Dynamic	Boolean	Modifiable	Engine default
automatic_sp_privileges	-	Dynamic	Boolean	Modifiable	Engine default
back_log	-	Static	Integer	Modifiable	Engine default
basedir	/rdsdbbin/mysql	Static	String	Non Modifiable	System default
big_tables	0	Dynamic	Boolean	Non Modifiable	Engine default

**Assigned resources (2)**

Name	Version	Region	Size
netflix-db-primary-dev	8.4.3	us-east-1b	db.t3.micro
netflix-db-secondary-dev	8.4.3	us-east-1a	db.t3.micro

**Recent events (0)**

No Recent events found

**Tags (0)**

Manage tags

Filter by Tag key

Tags	Value
------	-------

You don't have any tags associated with this resource.

Manage tags

## Option Groups:

The screenshot shows the AWS RDS console with the path: RDS > Option groups > default:mysql-8-4. The left sidebar includes links for Amazon RDS, Subnet groups, Parameter groups, Option groups (which is selected), Custom engine versions, Zero-ETL integrations, Events, Event subscriptions, Recommendations (0), and Certificate update.

**default:mysql-8-4**

**Option group properties**

- Amazon Resource Name (ARN): arn:aws:rds:us-east-1:149536471527:og:default:mysql-8-4
- Option group name: default:mysql-8-4
- Option group description: Default option group for mysql 8.4
- Name of database engine: mysql
- Major engine version: 8.4

**Associated DB instances and snapshots**

Resource	Type
<a href="#">netflix-db-primary-dev</a>	Instance
<a href="#">netflix-db-secondary-dev</a>	Instance

**Options**

Name	Persistent	Permanent	Port	Security groups	Version	Settings
No Options found						

## ElastiCache:

The screenshot shows the AWS ElastiCache console with the path: ElastiCache > Dashboard. The left sidebar includes links for Amazon ElastiCache, Dashboard, Resources (Valley caches, Memcached caches, Redis OSS caches, Global datastores, Reserved nodes, Backups), Configurations (Subnet groups, Parameter groups, User management, User group management), Events, Service updates, ElastiCache cluster client, Documentation, and Amazon MemoryDB.

**Create a cache**

Create a serverless cache, or choose your node size and type in Valley, Memcached or Redis OSS. You can also migrate your data from a Valley or Redis OSS cache. [Learn more](#)

**Create cache ▾**

**Resource overview**

Viewing data from US East (N. Virginia) region.

Valley caches	Memcached caches	Redis OSS caches	Global datastores
0	0	1	0

**ElastiCache health**

Region: US East (N. Virginia)  
Status: Service is operating normally

**Resources**

Resource type	Count
Backups	0
Reserved nodes	0
Subnet groups	1
Parameter groups	19
Users	1
User groups	0

## Subnet group:

The screenshot shows the AWS ElastiCache console with the 'Subnet groups' section selected. A specific subnet group, 'netflix-cache-subnet-group', is being viewed. The page displays the following details:

- Name:** netflix-cache-subnet-group
- Description:** Managed by Terraform
- VPC ID:** vpc-0d233da915a27022c
- Supported network types:** IPv4
- ARN:** arn:aws:elasticache:us-east-1:149536471527:subnetgroup:netflix-cache-subnet-group

The 'Subnets' tab is active, showing two subnets assigned to the group:

Availability Zone	Subnet ID	CIDR block (IPv4)
us-east-1a	subnet-0682b28c605c72e84	10.0.101.0/24
us-east-1b	subnet-092a5cc09075d70d5	10.0.102.0/24

The screenshot shows the AWS ElastiCache console with the 'Parameter groups' section selected. The page displays a list of 19 parameter groups:

Name	Family	Global	Description	ARN
default.memcached1.4	memcached1.4	No	Default parameter group for ...	arn:aws:elasticache:us-east-1:1495364715...
default.memcached1.5	memcached1.5	No	Default parameter group for ...	arn:aws:elasticache:us-east-1:1495364715...
default.memcached1.6	memcached1.6	No	Default parameter group for ...	arn:aws:elasticache:us-east-1:1495364715...
default.redis2.6	redis2.6	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis2.8	redis2.8	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis3.2	redis3.2	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis3.2.cluster.on	redis3.2	No	Customized default parameter...	arn:aws:elasticache:us-east-1:1495364715...
default.redis4.0	redis4.0	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis4.0.cluster.on	redis4.0	No	Customized default parameter...	arn:aws:elasticache:us-east-1:1495364715...
default.redis5.0	redis5.0	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis5.0.cluster.on	redis5.0	No	Customized default parameter...	arn:aws:elasticache:us-east-1:1495364715...
default.redis6.x	redis6.x	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis6.x.cluster.on	redis6.x	No	Customized default parameter...	arn:aws:elasticache:us-east-1:1495364715...
default.redis7	redis7	No	Default parameter group for r...	arn:aws:elasticache:us-east-1:1495364715...
default.redis7.cluster.on	redis7	No	Customized default parameter...	arn:aws:elasticache:us-east-1:1495364715...

## Events:

The screenshot shows the AWS ElastiCache console under the 'Events' section. The left sidebar includes links for Dashboard, Resources (Valley caches, Memcached caches, Redis OSS caches, Global datastores, Reserved nodes, Backups), Configurations (Subnet groups, Parameter groups, User management, User group management), Events (Service updates), and ElastiCache cluster client/Documentation/Amazon MemoryDB.

The main content area displays a table titled 'Events (13)'. The table has columns: Source ID, Type, Date, Event, Cause, and Status. All events listed are of type 'cache-cluster' and occurred on December 3, 2024, at various times between 13:01 and 14:45. Most events are marked as '-' for Cause and Status.

Source ID	Type	Date	Event	Cause	Status
netflix-cache-dev	cache-cluster	December 3, 2024, 14:45...	Added cache node 0001 ...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:45...	This cache cluster does n...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:45...	Cache cluster created	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:35...	Cache cluster deleted	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:35...	Removed cache nodes 0...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:15...	Added cache node 0001 ...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:15...	Cache cluster created	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:15...	This cache cluster does n...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:05...	Cache cluster deleted	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 14:05...	Removed cache nodes 0...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 13:15...	Added cache node 0001 ...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 13:15...	This cache cluster does n...	-	-
netflix-cache-dev	cache-cluster	December 3, 2024, 13:15...	Cache cluster created	-	-

## CloudFront:

The screenshot shows the AWS CloudFront console under the 'Distributions' section. The left sidebar includes links for Policies, Functions, Static IPs, VPC origins, What's new, Telemetry (Monitoring, Alarms, Logs), Reports & analytics (Cache statistics, Popular objects, Top referrers, Usage, Viewers), Security (Origin access, Field-level encryption), and Key management (Public keys, Key groups).

The main content area shows the 'E3QKJZ312NVGHT' distribution details. It includes tabs for General, Security, Origins, Behaviors, Error pages, Invalidations, Tags, and Logging. The General tab is selected. It displays the distribution domain name (dfkjaot8p4ek.cloudfront.net), ARN (arn:aws:cloudfront::149536471527:distribution/E3QKJZ312NVGHT), and last modified time (December 3, 2024 at 10:38:38 UTC). The Settings section includes fields for Description, Alternate domain names, Standard logging (Off), Cookie logging (Off), and Default root object. The Continuous deployment section shows a 'Create staging distribution' button.

## API Gateway:

The screenshot shows the AWS API Gateway console under the 'APIs' section. The left sidebar includes links for APIs (Custom domain names, Domain name access associations, VPC links), Usage plans, API keys, Client certificates, and Settings.

The main content area displays a table titled 'APIs (1/1)'. The table has columns: Name, Description, ID, Protocol, API endpoint type, and Created. There is one entry: 'netflix-api-dev' (Description: Netflix API Gateway, ID: fdtkb9eqoi, Protocol: REST, API endpoint type: Edge, Created: 2024-12-03).

Name	Description	ID	Protocol	API endpoint type	Created
netflix-api-dev	Netflix API Gateway	fdtkb9eqoi	REST	Edge	2024-12-03

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections' (empty), 'Environments' (empty), and 'History'. The main area is titled 'Netflix\_App / API\_Gateway' with a 'GET' request to 'https://fdtkb9eqoi.execute-api.us-east-1.amazonaws.com/dev/netflix'. The 'Params' tab is selected, showing a single parameter 'Key' with 'Value'. Below this, the 'Body' tab shows a JSON response:

```

1  {
2   "statusCode": 200,
3   "body": "{\"message\": \"Content processed successfully\", \"event\": {}}"
4 }

```

The status bar at the bottom indicates a 200 OK response with 508 ms and 628 B.

## S3 Bucket:

The screenshot shows the AWS S3 console. The left sidebar has sections for 'Amazon S3', 'General purpose buckets' (selected), 'Table buckets', 'Access Grants', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. It also includes links for 'Block Public Access settings for this account', 'Storage Lens', 'Dashboards', 'Storage Lens groups', 'AWS Organizations settings', and 'Feature spotlight'.

The main content area shows an 'Account snapshot - updated every 24 hours' with a link to 'View Storage Lens dashboard'. Below it, under 'General purpose buckets', there is one bucket listed:

Name	AWS Region	IAM Access Analyzer	Creation date
netflix-content-dev20241203223830927100000002	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	December 3, 2024, 14:38:35 (UTC-08:00)

Screenshot of the AWS CloudShell interface showing the Amazon S3 service. The user is viewing the bucket 'netflix-content-dev20241203223830927100000002'. The left sidebar shows various AWS services like General purpose buckets, Storage Lens, and Feature spotlight. The main content area displays a list of objects, including 'Netflix Intro (1080p).mp4'.

Name	Type	Last modified	Size	Storage class
Netflix Intro (1080p).mp4	mp4	December 3, 2024, 16:20:01 (UTC-08:00)	326.6 KB	Standard

Screenshot of the AWS CloudShell interface showing the Lambda service. The user is viewing the function 'netflix-content-processor-dev'. The left sidebar shows the Lambda environment 'NETFLIX-CONTENT-PROCESSOR-DEV'. The main content area shows the code editor for 'index.py' and a 'Create new test event' dialog. The right sidebar features a 'Create a simple web app' tutorial.

```
index.py
1 import json
2
3 def handler(event, context):
4     """
5         Process content-related operations
6     """
7     try:
8         return {
9             'statusCode': 200,
10            'body': json.dumps({
11                'message': 'Content processed successfully',
12                'event': event
13            })
14        }
15    except Exception as e:
16        return {
17            'statusCode': 500,
18            'body': json.dumps({'error': str(e)})
19        }
```

Create new test event

Event Name: TestAPI

Event sharing settings:

- This event is only available in the Lambda Console and to the event creator. You can configure a total of ten. [Learn more](#)
- 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:

```
{ "statusCode": 200, "body": "{\"message\": \"Content processed successfully\", \"event\": {\"key1\": \"value1\", \"key2\": \"value2\", \"key3\": \"value3\"}}"}
```