

Multi-Region Disaster Recovery

Objective

To design and implement a multi-region disaster recovery (DR) architecture using AWS services, where infrastructure is deployed in two AWS regions: one as Primary (active) and the other as Disaster Recovery (standby). The infrastructure will ensure:

High availability and scalability,

Cross-region data replication,

Automatic failover and failback,

Infrastructure as Code using Terraform for consistency and repeatability.

AWS Services Used

VPC : Network isolation in each region (public/private subnets)

EC2: Host static web content from S3 via user_data

Application Load Balancer: Load traffic across EC2 instances

Auto Scaling Group: Ensure high availability and scaling

S3: Static content and cross-region replication

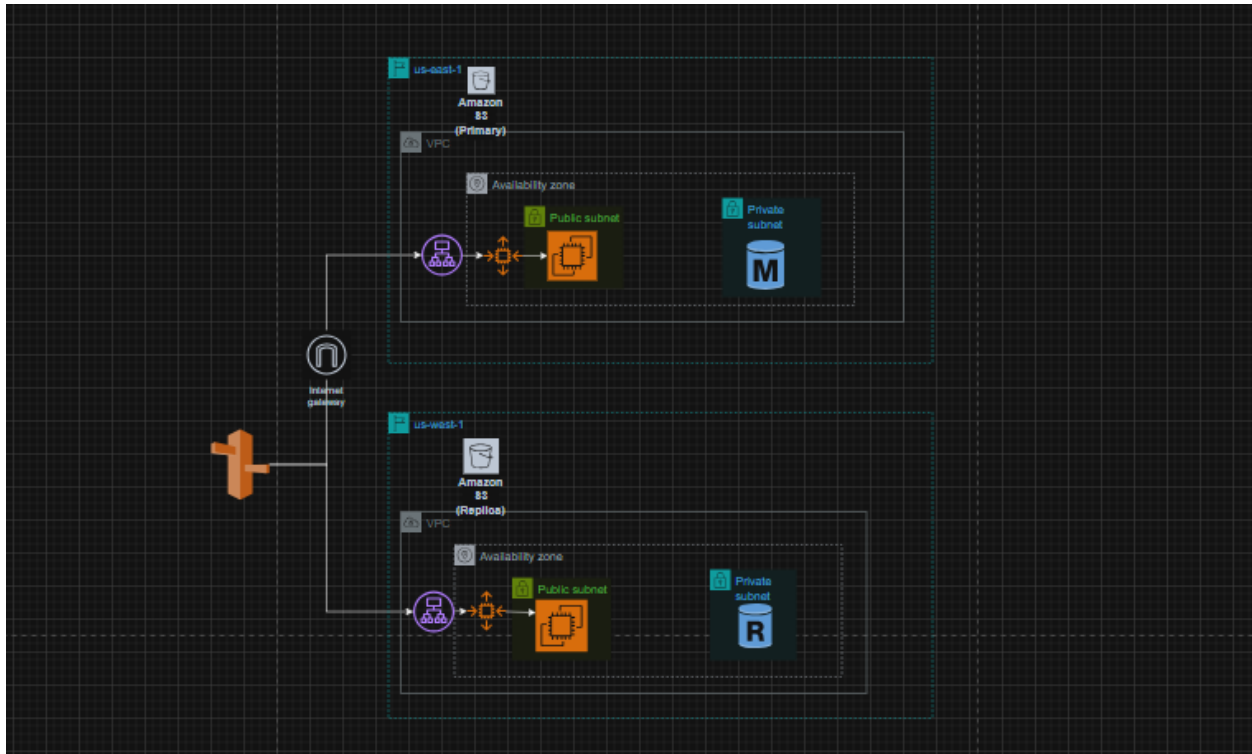
RDS (MySQL): Primary DB in Region 1, replica in Region 2

Route 53: DNS-based failover to redirect traffic to DR region

IAM: Permissions for EC2, S3 replication, etc.

Terraform: Infrastructure as Code (IaC) for automation

Architecture diagram



It is designed to provide high availability and disaster recovery for a web application by distributing its components across two separate AWS region .It ensures that if the primary region becomes unavailable, traffic can be automatically failed over to a secondary region with minimal data loss and downtime .

Terraform files over view

1. providers.tf

Defines the AWS provider and region

```
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = ">= 6.0"  
    }  
  }  
}
```

Provider for the Primary Region

```
provider "aws" {  
  alias = "primary"  
  region = var.primary_region  
}  
#Provider for the secondary region  
provider "aws" {  
  alias = "secondary"  
  region = var.secondary_region  
}
```

2. variables.tf

All inputs to make the infrastructure reusable and modular

```
variable "primary_region" {  
  default = "us-east-1"  
}  
  
variable "secondary_region" {  
  default = "us-west-1"  
}  
  
variable "vpc_cidr" {  
  default = "192.168.4.0/24"  
}  
  
variable "public_subnet_cidr_1" {  
  default = "192.168.4.0/26"  
}  
  
variable "public_subnet_cidr_2" {  
  default = "192.168.4.64/26"  
}  
  
variable "private_subnet_cidr_1" {  
  default = "192.168.4.128/26"  
}  
  
variable "private_subnet_cidr_2" {  
  default = "192.168.4.192/26"  
}  
  
variable "availability_zone_1" {
```

```
    default = "us-east-1a"
}

variable "availability_zone_2" {
    default = "us-east-1b"
}

#cidr of secondary region
variable "vpc_cidr_secondary" {
    default = "192.168.5.0/24"
}

variable "public_subnet_cidr_1_secondary" {
    default = "192.168.5.0/26"
}

variable "public_subnet_cidr_2_secondary" {
    default = "192.168.5.64/26"
}

variable "private_subnet_cidr_1_secondary" {
    default = "192.168.5.128/26"
}

variable "private_subnet_cidr_2_secondary" {
    default = "192.168.5.192/26"
}

variable "availability_zone_1_secondary" {
    default = "us-west-1a"
}

variable "availability_zone_2_secondary" {
    default = "us-west-1c"
}

#RDS
variable "db_username" {
    default = "admin"
}

variable "db_password" {
    default = "MySecurePassword123"
}
```

3. main.tf

VPC and Networking (in Both Regions)

S3 Buckets

Compute Layer – EC2, ALB, ASG

Database – RDS

```
# ----- PRIMARY REGION -----

# VPC
resource "aws_vpc" "primary" {
  provider      = aws.primary
  cidr_block    = var.vpc_cidr
  tags          = { Name = "Primary-VPC" }
}

resource "aws_internet_gateway" "primary" {
  provider = aws.primary
  vpc_id   = aws_vpc.primary.id
}

# Public Subnets
resource "aws_subnet" "primary_public_1" {
  provider              = aws.primary
  vpc_id                = aws_vpc.primary.id
  cidr_block            = var.public_subnet_cidr_1
  availability_zone     = var.availability_zone_1
  map_public_ip_on_launch = true
  tags                  = { Name = "Primary-Public-Subnet-1" }
}

resource "aws_subnet" "primary_public_2" {
  provider              = aws.primary
  vpc_id                = aws_vpc.primary.id
  cidr_block            = var.public_subnet_cidr_2
  availability_zone     = var.availability_zone_2
  map_public_ip_on_launch = true
  tags                  = { Name = "Primary-Public-Subnet-2" }
}
```

```

# Private Subnets
resource "aws_subnet" "primary_private_1" {
  provider      = aws.primary
  vpc_id        = aws_vpc.primary.id
  cidr_block    = var.private_subnet_cidr_1
  availability_zone = var.availability_zone_1
  tags          = { Name = "Primary-Private-Subnet-1" }
}

resource "aws_subnet" "primary_private_2" {
  provider      = aws.primary
  vpc_id        = aws_vpc.primary.id
  cidr_block    = var.private_subnet_cidr_2
  availability_zone = var.availability_zone_2
  tags          = { Name = "Primary-Private-Subnet-2" }
}

# Route Table
resource "aws_route_table" "primary_public_rt" {
  provider = aws.primary
  vpc_id   = aws_vpc.primary.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.primary.id
  }
}

resource "aws_route_table_association" "primary_public_assoc_1" {
  provider      = aws.primary
  subnet_id     = aws_subnet.primary_public_1.id
  route_table_id = aws_route_table.primary_public_rt.id
}

resource "aws_route_table_association" "primary_public_assoc_2" {
  provider      = aws.primary
  subnet_id     = aws_subnet.primary_public_2.id
  route_table_id = aws_route_table.primary_public_rt.id
}

# Security Group
resource "aws_security_group" "primary_web_sg" {
  provider = aws.primary
  name     = "primary-web-sg"
}

```

```

description = "Allow HTTP"
vpc_id      = aws_vpc.primary.id

ingress {
  from_port = 80
  to_port   = 80
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
}

# IAM Role for EC2 (with S3 readonly access)
resource "aws_iam_role" "ec2_role" {
  name = "ec2-s3-read-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Effect      = "Allow",
      Principal   = { Service = "ec2.amazonaws.com" },
      Action      = "sts:AssumeRole"
    }]
  })
}

resource "aws_iam_role_policy_attachment" "ec2_s3_readonly" {
  role          = aws_iam_role.ec2_role.name
  policy_arn    = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}

resource "aws_iam_instance_profile" "ec2_profile" {
  name = "ec2-instance-profile"
  role = aws_iam_role.ec2_role.name
}

# Latest Amazon Linux 2 AMI
data "aws_ami" "amazon_linux_primary" {
  provider = aws.primary

```

```

most_recent = true
owners      = ["amazon"]

filter {
  name      = "name"
  values    = ["amzn2-ami-hvm-*-x86_64-gp2"]
}
}

# S3 Bucket for static files
resource "aws_s3_bucket" "primary_bucket" {
  provider = aws.primary
  bucket   = "oneill123"
}

resource "aws_s3_bucket_versioning" "primary_bucket_versioning" {
  provider = aws.primary
  bucket   = aws_s3_bucket.primary_bucket.id

  versioning_configuration {
    status = "Enabled"
  }
}

resource "aws_s3_bucket_policy" "primary_public_read" {
  provider = aws.primary
  bucket   = aws_s3_bucket.primary_bucket.id

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Sid      = "PublicRead",
      Effect   = "Allow",
      Principal = "*",
      Action    = "s3:GetObject",
      Resource  = "arn:aws:s3:::${aws_s3_bucket.primary_bucket.id}/*"
    }]
  })
}

resource "aws_s3_object" "html" {
  provider      = aws.primary
  bucket        = aws_s3_bucket.primary_bucket.id

```



```

    key          = "index.html"
    source       = "${path.module}/files/index.html.tpl"
    content_type = "text/html"
}

resource "aws_s3_object" "image" {
    bucket      = aws_s3_bucket.primary_bucket.id
    key         = "image.jpg"
    source      = "${path.module}/files/image.jpg"
    content_type = "image/jpeg"
}

# Launch Template with user_data
resource "aws_launch_template" "primary_web_template" {
    provider      = aws.primary
    name_prefix   = "primary-web-sg"
    image_id      = data.aws_ami.amazon_linux_primary.id
    instance_type = "t2.micro"

    iam_instance_profile {
        name = aws_iam_instance_profile.ec2_profile.name
    }

    user_data = base64encode(templatefile("${path.module}/files/user_data.sh.tpl",
    {
        s3_bucket = aws_s3_bucket.primary_bucket.bucket
    }))

    vpc_security_group_ids = [aws_security_group.primary_web_sg.id]
}

# Auto Scaling Group
resource "aws_autoscaling_group" "primary_web_asg" {
    provider          = aws.primary
    desired_capacity  = 2
    max_size          = 3
    min_size          = 1
    vpc_zone_identifier = [aws_subnet.primary_public_1.id,
aws_subnet.primary_public_2.id]

    launch_template {
        id = aws_launch_template.primary_web_template.id
    }
}

```

```

    version = "$Latest"
  }

  tag {
    key          = "Name"
    value        = "Primary-Web-ASG"
    propagate_at_launch = true
  }

  health_check_type      = "EC2"
  health_check_grace_period = 300
}

# ALB + Target Group + Listener
resource "aws_lb" "primary_alb" {
  provider          = aws.primary
  name              = "primary-web-alb"
  internal          = false
  load_balancer_type = "application"
  subnets          = [aws_subnet.primary_public_1.id,
aws_subnet.primary_public_2.id]
  security_groups   = [aws_security_group.primary_web_sg.id]
}

resource "aws_lb_target_group" "primary_tg" {
  provider = aws.primary
  name     = "primary-web-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.primary.id
  health_check {
    path          = "/"
    healthy_threshold = 2
    unhealthy_threshold = 2
    timeout       = 5
    interval      = 30
    matcher       = "200"
  }
}

resource "aws_lb_listener" "primary_listener" {
  provider          = aws.primary
  load_balancer_arn = aws_lb.primary_alb.arn
  port              = 80
  protocol           = "HTTP"

```

```

    default_action {
      type           = "forward"
      target_group_arn = aws_lb_target_group.primary_tg.arn
    }
  }

resource "aws_autoscaling_attachment" "primary_asg_alb_attachment" {
  provider                = aws.primary
  autoscaling_group_name = aws_autoscaling_group.primary_web_asg.name
  lb_target_group_arn     = aws_lb_target_group.primary_tg.arn
}

# RDS Subnet Group
#resource "aws_db_subnet_group" "primary_db_subnet_group" {
#  provider = aws.primary
#  name      = "primary-db-subnet-group"
#  subnet_ids = [
#    aws_subnet.primary_private_1.id,
#    aws_subnet.primary_private_2.id,
#  ]

#  tags = { Name = "Primary DB Subnet Group" }
#}

# Primary RDS
#resource "aws_db_instance" "primary_rds" {
#  provider                = aws.primary
#  allocated_storage       = 20
#  engine                  = "mysql"
#  engine_version          = "8.0"
#  instance_class          = "db.t3.micro"
#  username                = var.db_username
#  password                = var.db_password
#  db_subnet_group_name    = aws_db_subnet_group.primary_db_subnet_group.name
#  vpc_security_group_ids = [aws_security_group.primary_web_sg.id]
#  multi_az                = false
#  publicly_accessible     = false
#  skip_final_snapshot     = true
#  backup_retention_period = 7
#  tags                    = { Name = "Primary RDS" }
#}

# ----- SECONDARY REGION -----

```

```
# Secondary VPC
resource "aws_vpc" "secondary" {
  provider      = aws.secondary
  cidr_block    = var.vpc_cidr_secondary
  tags          = { Name = "Secondary-VPC" }
}

resource "aws_internet_gateway" "secondary" {
  provider = aws.secondary
  vpc_id   = aws_vpc.secondary.id
}

# Secondary Public Subnets
resource "aws_subnet" "secondary_public_1" {
  provider              = aws.secondary
  vpc_id                = aws_vpc.secondary.id
  cidr_block            = var.public_subnet_cidr_1_secondary
  availability_zone      = var.availability_zone_1_secondary
  map_public_ip_on_launch = true
  tags                  = { Name = "Secondary-Public-Subnet-1" }
}

resource "aws_subnet" "secondary_public_2" {
  provider              = aws.secondary
  vpc_id                = aws_vpc.secondary.id
  cidr_block            = var.public_subnet_cidr_2_secondary
  availability_zone      = var.availability_zone_2_secondary
  map_public_ip_on_launch = true
  tags                  = { Name = "Secondary-Public-Subnet-2" }
}

# Secondary Private Subnets
resource "aws_subnet" "secondary_private_1" {
  provider              = aws.secondary
  vpc_id                = aws_vpc.secondary.id
  cidr_block            = var.private_subnet_cidr_1_secondary
  availability_zone      = var.availability_zone_1_secondary
  tags                  = { Name = "Secondary-Private-Subnet-1" }
}

resource "aws_subnet" "secondary_private_2" {
  provider      = aws.secondary
  vpc_id        = aws_vpc.secondary.id
  cidr_block    = var.private_subnet_cidr_2_secondary
}
```

```

availability_zone = var.availability_zone_2_secondary
tags              = { Name = "Secondary-Private-Subnet-2" }
}

# Route Table
resource "aws_route_table" "secondary_public_rt" {
  provider = aws.secondary
  vpc_id   = aws_vpc.secondary.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.secondary.id
  }
}

resource "aws_route_table_association" "secondary_public_assoc_1" {
  provider      = aws.secondary
  subnet_id     = aws_subnet.secondary_public_1.id
  route_table_id = aws_route_table.secondary_public_rt.id
}

resource "aws_route_table_association" "secondary_public_assoc_2" {
  provider      = aws.secondary
  subnet_id     = aws_subnet.secondary_public_2.id
  route_table_id = aws_route_table.secondary_public_rt.id
}

# Security Group
resource "aws_security_group" "secondary_web_sg" {
  provider      = aws.secondary
  name          = "secondary-web-sg"
  description   = "Allow HTTP"
  vpc_id        = aws_vpc.secondary.id

  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
  }
}

```

```

    cidr_blocks = ["0.0.0.0/0"]
  }
}

# IAM Role for EC2 (with S3 readonly access) - Secondary
resource "aws_iam_role" "ec2_role_secondary" {
  name = "ec2-s3-read-role-secondary"

  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Effect    = "Allow",
      Principal = { Service = "ec2.amazonaws.com" },
      Action    = "sts:AssumeRole"
    }]
  })
}

resource "aws_iam_role_policy_attachment" "ec2_s3_readonly_secondary" {
  role      = aws_iam_role.ec2_role_secondary.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}

resource "aws_iam_instance_profile" "ec2_profile_secondary" {
  name = "ec2-instance-profile-secondary"
  role = aws_iam_role.ec2_role_secondary.name
}

# Latest Amazon Linux 2 AMI
data "aws_ami" "amazon_linux_secondary" {
  provider      = aws.secondary
  most_recent   = true
  owners        = ["amazon"]

  filter {
    name     = "name"
    values   = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}

# S3 Bucket for static files - Secondary Region
resource "aws_s3_bucket" "secondary_bucket" {
  provider = aws.secondary
  bucket   = "oneill123-secondary"
}

```

```

resource "aws_s3_bucket_versioning" "secondary_bucket_versioning" {
  provider = aws.secondary
  bucket   = aws_s3_bucket.secondary_bucket.id

  versioning_configuration {
    status = "Enabled"
  }
}

resource "aws_s3_bucket_policy" "secondary_public_read" {
  provider = aws.secondary
  bucket   = aws_s3_bucket.secondary_bucket.id

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Sid      = "PublicRead",
      Effect   = "Allow",
      Principal = "*",
      Action    = "s3:GetObject",
      Resource  = "arn:aws:s3:::${aws_s3_bucket.secondary_bucket.id}/*"
    }]
  })
}

resource "aws_s3_object" "secondary_html" {
  provider      = aws.secondary
  bucket        = aws_s3_bucket.secondary_bucket.id
  key           = "index.html"
  source        = "${path.module}/files/index.html.tpl"
  content_type  = "text/html"
}

resource "aws_s3_object" "secondary_image" {
  provider      = aws.secondary
  bucket        = aws_s3_bucket.secondary_bucket.id
  key           = "image.jpg"
  source        = "${path.module}/files/image.jpg"
  content_type  = "image/jpeg"
}

# Launch Template with user_data
resource "aws_launch_template" "secondary_web_template" {
  provider = aws.secondary

```

```

name_prefix    = "secondary-web-sg"
image_id       = data.aws_ami.amazon_linux_secondary.id
instance_type  = "t2.micro"

iam_instance_profile {
  name = aws_iam_instance_profile.ec2_profile_secondary.name
}

user_data = base64encode(templatefile("${path.module}/files/user_data.sh.tpl",
{
  s3_bucket = aws_s3_bucket.secondary_bucket.bucket
}))

vpc_security_group_ids = [aws_security_group.secondary_web_sg.id]
}

# Auto Scaling Group
resource "aws_autoscaling_group" "secondary_web_asg" {
  provider           = aws.secondary
  desired_capacity   = 2
  max_size           = 3
  min_size           = 1
  vpc_zone_identifier = [aws_subnet.secondary_public_1.id,
aws_subnet.secondary_public_2.id]

  launch_template {
    id       = aws_launch_template.secondary_web_template.id
    version = "$Latest"
  }

  tag {
    key          = "Name"
    value        = "Secondary-Web-ASG"
    propagate_at_launch = true
  }

  health_check_type      = "EC2"
  health_check_grace_period = 300
}

# ALB + Target Group + Listener
resource "aws_lb" "secondary_alb" {
  provider = aws.secondary
  name     = "secondary-web-alb"

```



```

    internal          = false
    load_balancer_type = "application"
    subnets          = [aws_subnet.secondary_public_1.id,
aws_subnet.secondary_public_2.id]
    security_groups    = [aws_security_group.secondary_web_sg.id]
}

resource "aws_lb_target_group" "secondary_tg" {
  provider = aws.secondary
  name      = "secondary-web-target-group"
  port      = 80
  protocol  = "HTTP"
  vpc_id    = aws_vpc.secondary.id
  health_check {
    path            = "/"
    healthy_threshold = 2
    unhealthy_threshold = 2
    timeout         = 5
    interval        = 30
    matcher          = "200"
  }
}

resource "aws_lb_listener" "secondary_listener" {
  provider          = aws.secondary
  load_balancer_arn = aws_lb.secondary_alb.arn
  port              = 80
  protocol           = "HTTP"

  default_action {
    type = "forward"
    target_group_arn = aws_lb_target_group.secondary_tg.arn
  }
}

resource "aws_autoscaling_attachment" "secondary_asg_alb_attachment" {
  provider          = aws.secondary
  autoscaling_group_name = aws_autoscaling_group.secondary_web_asg.name
  lb_target_group_arn    = aws_lb_target_group.secondary_tg.arn
}

# Secondary DB Subnet Group
#resource "aws_db_subnet_group" "secondary_db_subnet_group" {
#  provider = aws.secondary
#  name      = "secondary-db-subnet-group"

```

```

#subnet_ids = [
  # aws_subnet.secondary_private_1.id,
  #aws_subnet.secondary_private_2.id,
#]

#tags = { Name = "Secondary DB Subnet Group" }
#}

# Secondary RDS Read Replica
#resource "aws_db_instance" "secondary_rds" {
  # provider                = aws.secondary
  #allocated_storage        = 20
  #engine                   = "mysql"
  #engine_version           = "8.0"
  #instance_class           = "db.t3.micro"
  # username                = var.db_username
  #password                 = var.db_password
  #db_subnet_group_name     = aws_db_subnet_group.secondary_db_subnet_group.name
  #vpc_security_group_ids   = [aws_security_group.secondary_web_sg.id]
  #multi_az                 = false
  #publicly_accessible      = false
  #skip_final_snapshot      = true
  #backup_retention_period  = 7

  #replicate_source_db = aws_db_instance.primary_rds.id

  #tags = { Name = "Secondary RDS Replica" }
#}

```

4. output.tf

```

output "primary_alb_dns" {
  value = aws_lb.primary_alb.dns_name
}

output "secondary_alb_dns" {
  value = aws_lb.secondary_alb.dns_name
}

```

Save these 4 files in the same directory:

- main.tf

- variables.tf
- providers.tf
- output.tf

Run Terraform commands:

- terraform init
- terraform plan
- terraform apply

Terraform init

```
PS C:\Users\Vivek\Desktop\New folder\terraform_practice\multi_region_disaster> terraform init
• Initializing the backend...
• Initializing provider plugins...
  - Retrieving previous version of hashicorp/aws from the dependency lock file
  - Using previously-installed hashicorp/aws v6.2.0

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\Vivek\Desktop\New folder\terraform_practice\multi_region_disaster>
```

Terraform plan

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
+ "Name" = "Secondary-VPC"
}
+ tags_all = {
  + "Name" = "Secondary-VPC"
}
}

Plan: 48 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + primary_alb_dns = (known after apply)
  + secondary_alb_dns = (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\Vivek\Desktop\New folder\terraform_practice\multi_region_disaster>
```

Terraform apply

```
    + "Name" = "Secondary-VPC"
  }
  tags_all = {
    + "Name" = "Secondary-VPC"
  }
}

Plan: 48 to add, 8 to change, 8 to destroy.

Changes to Outputs:
  + primary_alb_dns = (known after apply)
  + secondary_alb_dns = (known after apply)

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_s3_bucket_policy.primary_public_read: Creating...
aws_s3_bucket_policy.primary_public_read: Creation complete after 1s [id=oneill123]
aws_s3_bucket_policy.secondary_public_read: Creating...
aws_s3_bucket_policy.secondary_public_read: Creation complete after 1s [id=oneill123-secondary]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
primary_alb_dns = "primary-web-alb-401762522.us-east-1.elb.amazonaws.com"
secondary_alb_dns = "secondary-web-alb-818489321.us-west-1.elb.amazonaws.com"
PS C:\Users\Wivok\Desktop\Wiv folder\terraform_practice\multi_region_disaster>
```

Check in aws console

Primary VPC created

The screenshot shows the AWS Management Console interface for the 'Your VPCs' page. The page displays a table of VPCs with columns for Name, VPC ID, State, Block Public Access, IPV4 CIDR, and IPV6 CIDR. The 'Primary-VPC' is highlighted, showing its VPC ID as vpc-018f3a91ef71ae966 and its state as 'Available'. Below the table, the details for the 'Primary-VPC' are shown, including its VPC ID, state, DNS resolution, main network ACL, block public access, DHCP option set, and DNS hostnames.

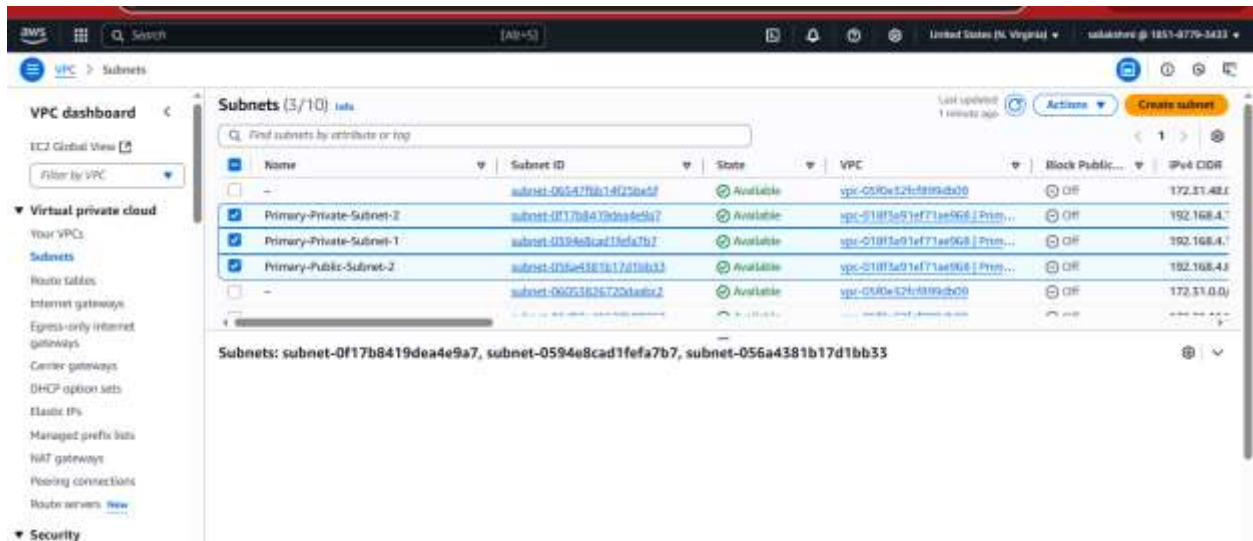
Name	VPC ID	State	Block Public...	IPV4 CIDR	IPV6 CIDR
Primary-VPC	vpc-018f3a91ef71ae966	Available	Off	192.168.4.0/24	-
-	vpc-03f0e32b1829d509	Available	Off	172.17.0.0/16	-

vpc-018f3a91ef71ae966 / Primary-VPC

Details

- VPC ID: vpc-018f3a91ef71ae966
- State: Available
- DNS resolution: Enabled
- Main network ACL: Default VPC
- Block Public Access: Off
- DHCP option set: opt-fd9804c9205d5137
- DNS hostnames: Disabled
- Main route table: rtb-004580b9f590ff90
- IPV4 CIDR: 192.168.4.0/24
- IPV6 pool: -

Subnets

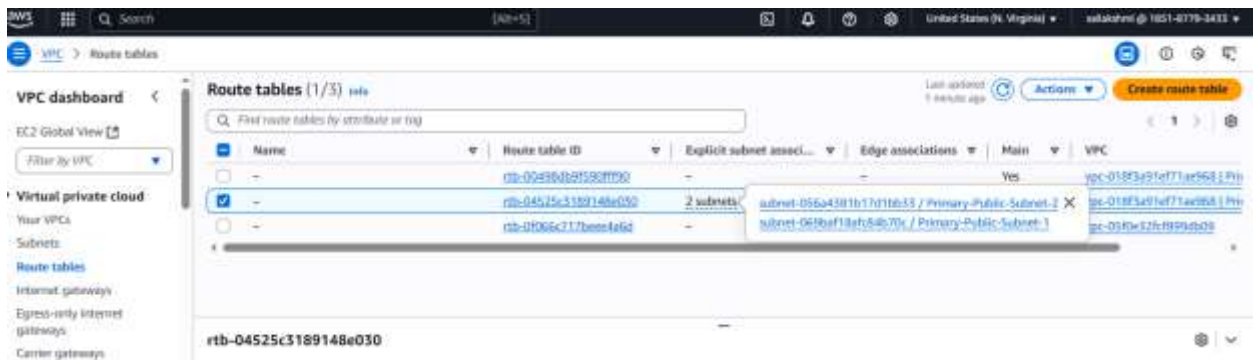


Subnets (3/10)

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
-	subnet-061a7f8a14f75ba5f	Available	vpc-018f3a91ef71ae968	Off	172.31.0.0/24
Primary-Private-Subnet-2	subnet-0f17b8419dea4e9a7	Available	vpc-018f3a91ef71ae968 Prim...	Off	192.168.4.0/24
Primary-Private-Subnet-1	subnet-0394e8cad1fefa7b7	Available	vpc-018f3a91ef71ae968 Prim...	Off	192.168.4.0/24
Primary-Public-Subnet-2	subnet-056a4381b17d1bb33	Available	vpc-018f3a91ef71ae968 Prim...	Off	192.168.4.0/24
-	subnet-0a0c362672ca4a0f2	Available	vpc-018f3a91ef71ae968	Off	172.31.0.0/24

Subnets: subnet-0f17b8419dea4e9a7, subnet-0594e8cad1fefa7b7, subnet-056a4381b17d1bb33

Route table



Route tables (1/3)

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
-	rtb-004980d8f580ff000	-	-	Yes	vpc-018f3a91ef71ae968 Pri...
-	rtb-04525c3189148e030	2 subnets	subnet-056a4381b17d1bb33 / Primary-Public-Subnet-2	X	vpc-018f3a91ef71ae968 Pri...
-	rtb-0f066c717bbee4f6d	-	subnet-061baf18af54b70c / Primary-Public-Subnet-1		vpc-018f3a91ef71ae968 Pri...

rtb-04525c3189148e030

Internet gateway

Internet gateways (1/2)

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-02235fe5c9ff5ba94	Attached	vpc-018f3a91ef71ee956 (Primary VPC)	185187795433
-	igw-0b8de18652ac794ad	Attached	vpc-090ae52b7803db00	185187795433

igw-02235fe5c9ff5ba94

Details

Internet gateway ID	State	VPC ID	Owner
igw-02235fe5c9ff5ba94	Attached	vpc-018f3a91ef71ee956 (Primary VPC)	185187795433

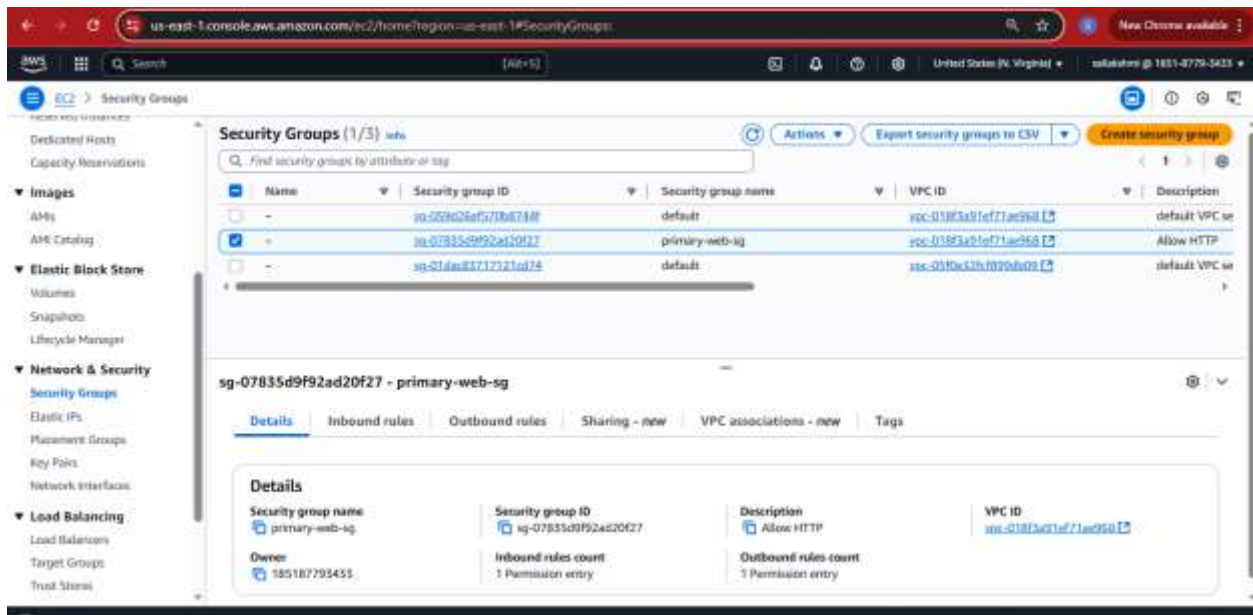
EC2 instances got created

Instances (2)

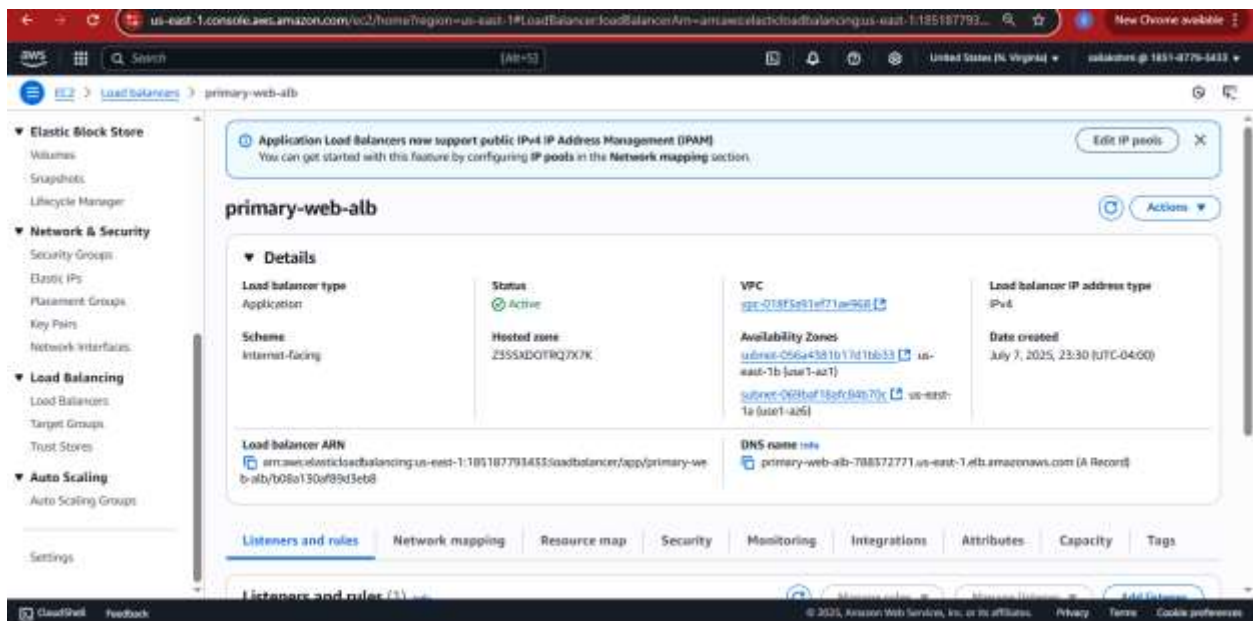
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Primary-Web-...	i-0609f07f2a5a0b6cf	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
Primary-Web-...	i-094ae0932bdc57cd	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-

Select an instance

Security group



Load balancer



Target group

The image displays two screenshots of the AWS Management Console, specifically the 'primary-web-target-group' page.

Top Screenshot: Details Tab

- Target type:** Instance
- Protocol : Port:** HTTP: 80
- Protocol version:** HTTP1
- VPC:** vpc-018f2a1ef77ae968
- IP address type:** IPv4
- Load balancer:** primary-web-alb
- Summary:** 2 Total targets, 2 Healthy, 0 Unhealthy, 0 Unused, 0 Initial, 0 Draining.
- Distribution of targets by Availability Zone (AZ):** Select values in this table to see corresponding filters applied to the Registered targets table below.
- Registered targets (2):** Anomaly mitigation: Not applicable. Buttons: Deregister, Register targets.

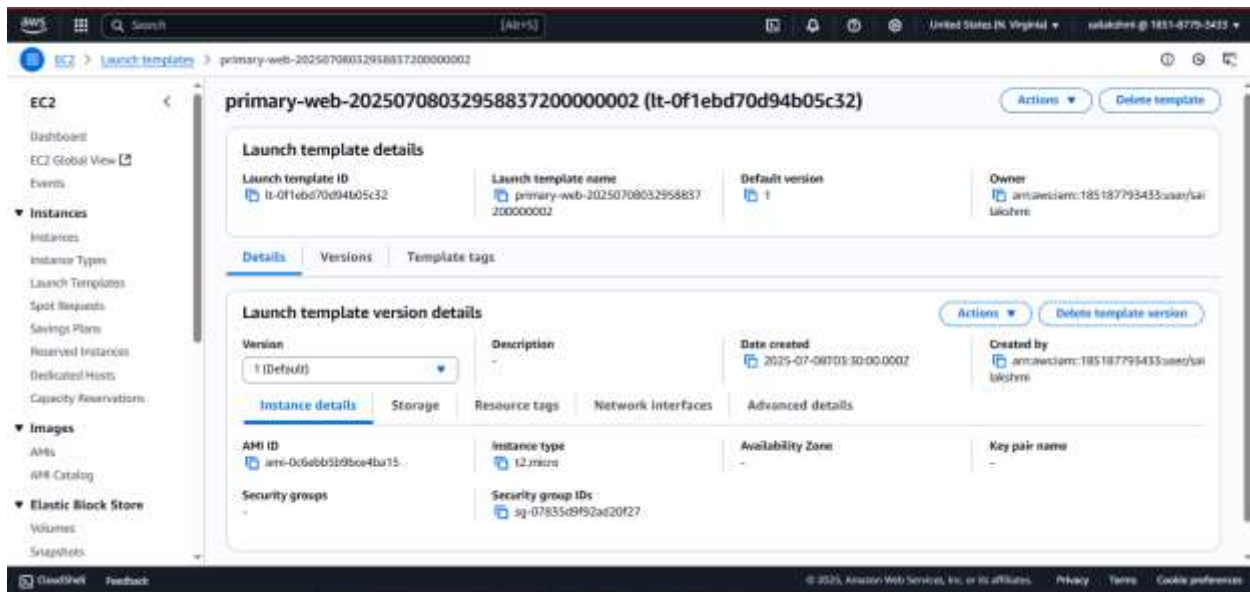
Bottom Screenshot: Targets Tab

Registered targets (2)

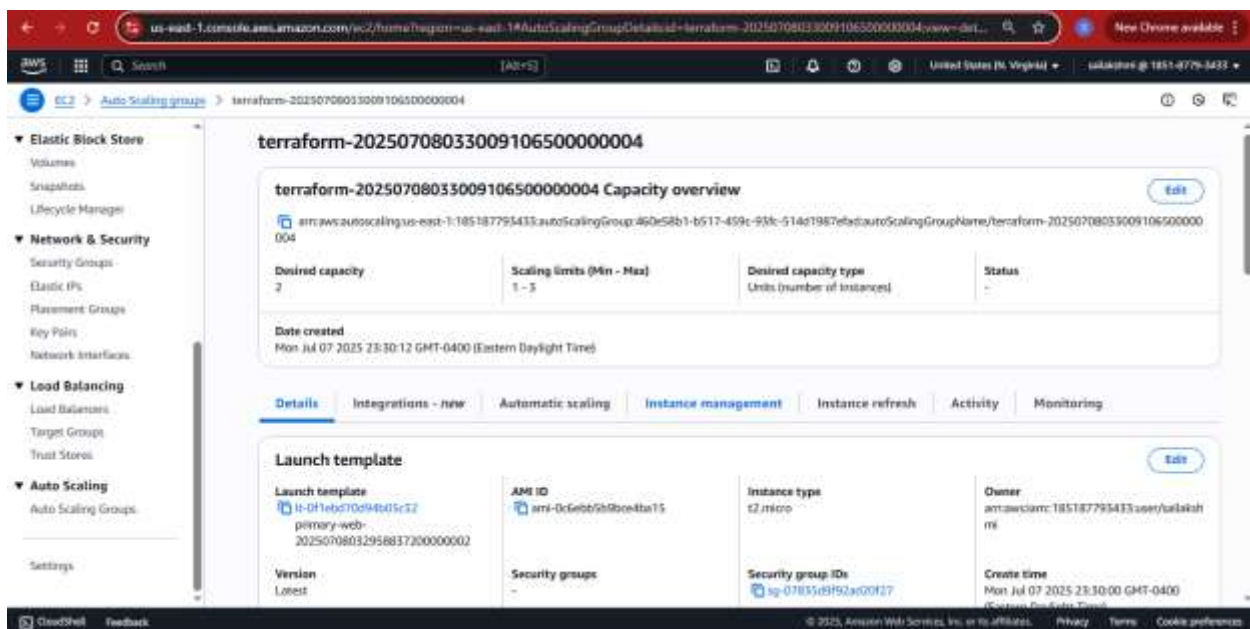
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least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

<input type="checkbox"/>	Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Over...
<input type="checkbox"/>	i-08f9f071a5a0c6cf	Primary-Web...	80	us-east-1a (us...)	Healthy	-	<input type="checkbox"/> No override	No o
<input type="checkbox"/>	i-094a4a0c912d4d57cb	Primary-Web...	80	us-east-1b (us...)	Healthy	-	<input type="checkbox"/> No override	No o

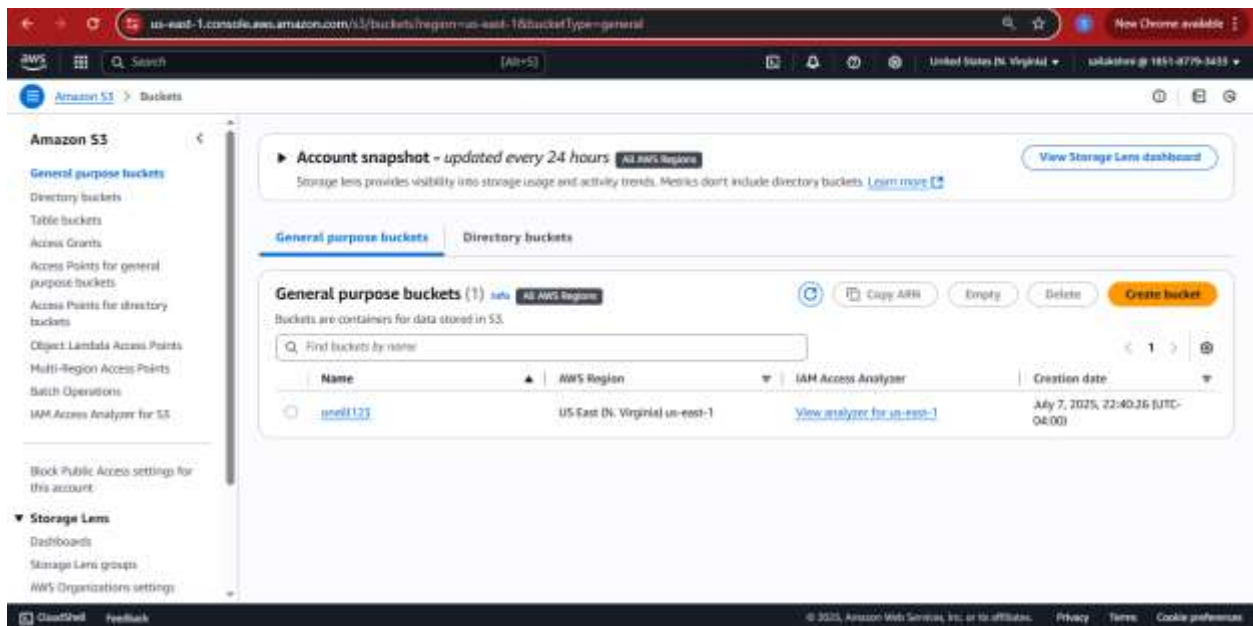
Launch template



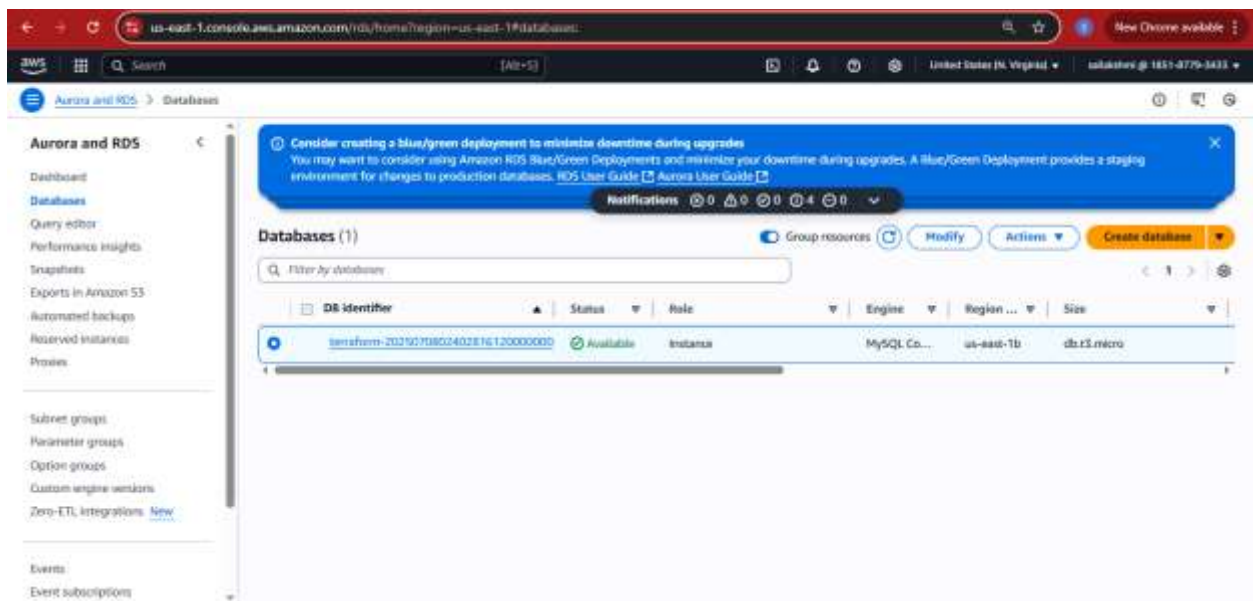
Auto scaling group



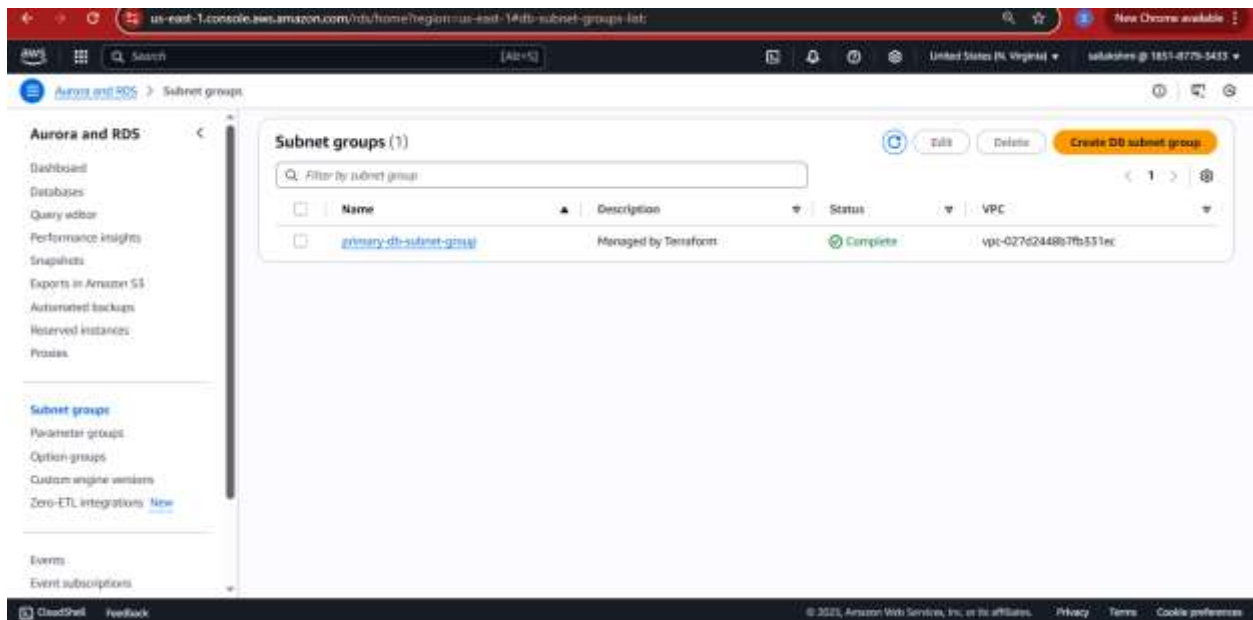
S3 bucket created



Primary database

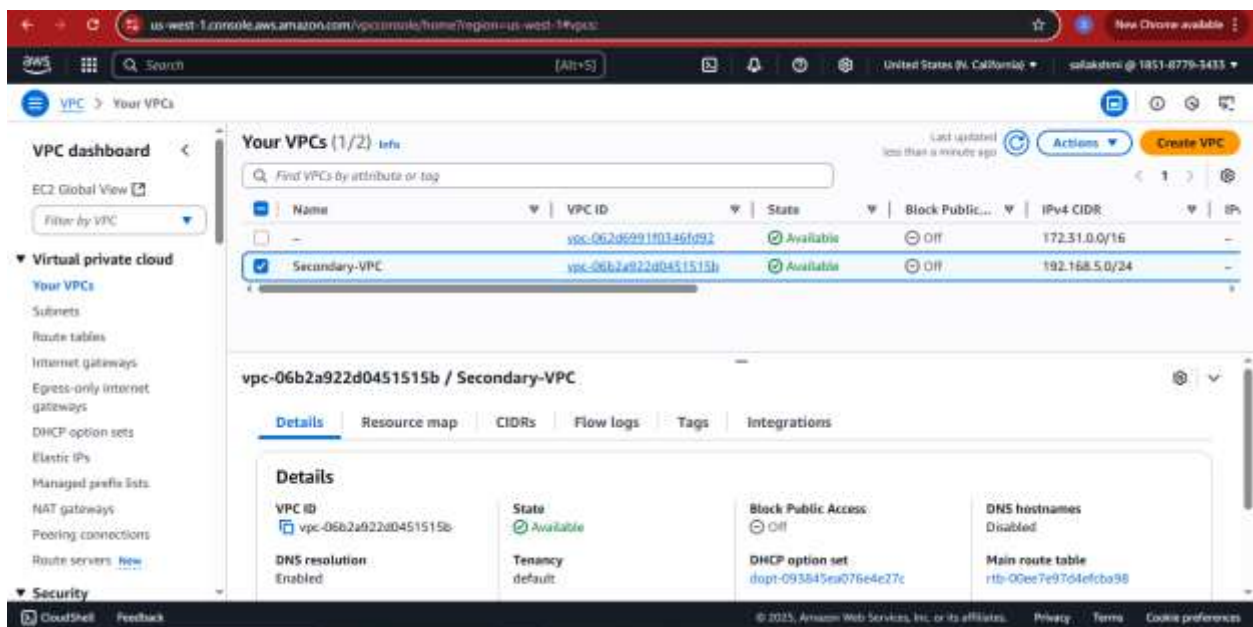


Db subnet group

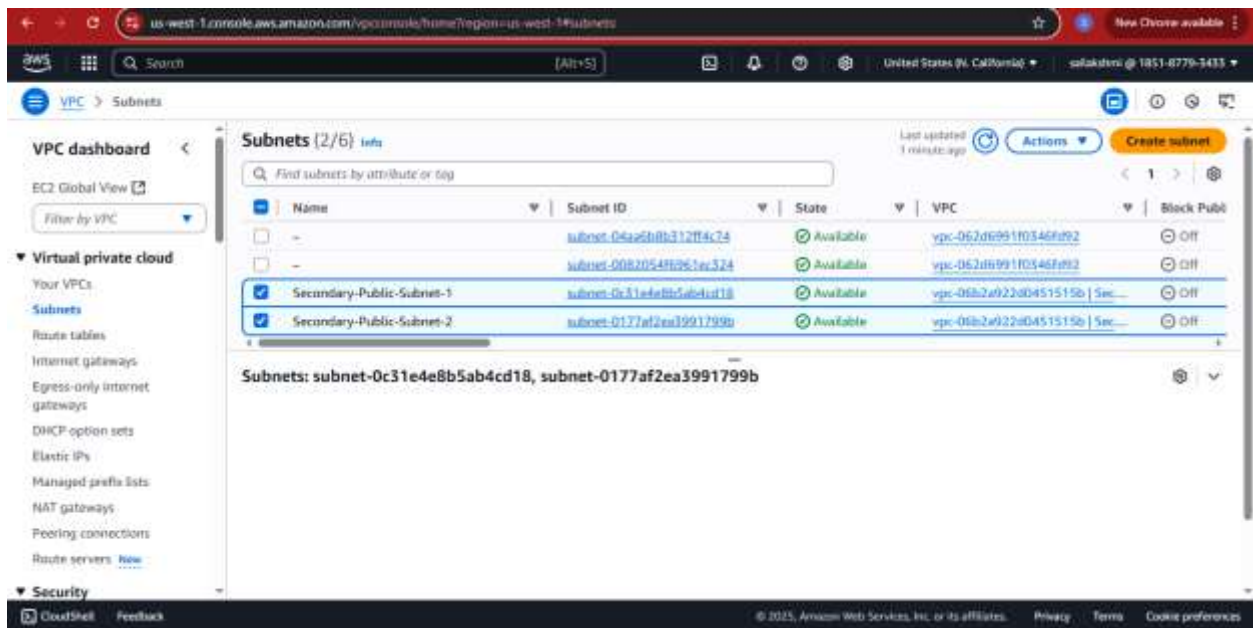


Secondary region

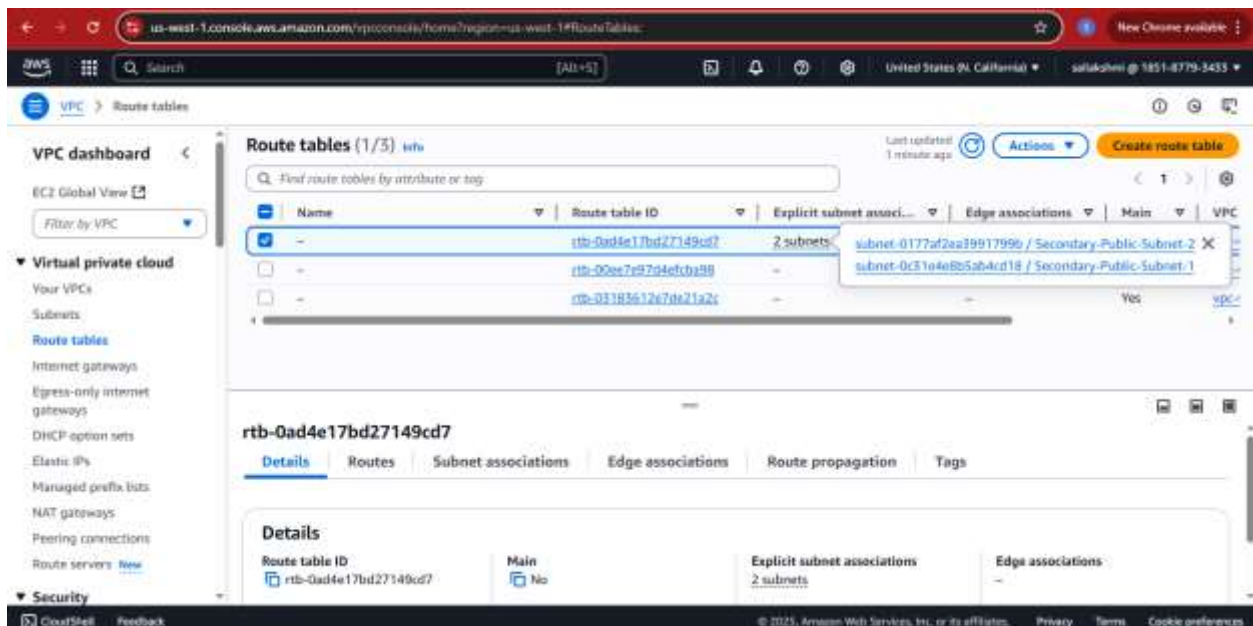
VPC created



Subnets created



Route table created



Internet gateway

us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#vgw

VPC > Internet gateways

Internet gateways (1/2) info

Find internet gateways by attribute or tag

Name	Internet gateway ID	State	VPC ID
-	igw-0c71af0e121c3c4a3	Attached	vpc-06b2a922d0451515b Secondary...
-	igw-0ab52af8b8bd1e0997	Attached	vpc-062d8291f01346h92

igw-0d71e80e121c3c4a3

Details Tags

Details

Internet gateway ID igw-0d71e80e121c3c4a3	State Attached	VPC ID vpc-06b2a922d0451515b Secondary VPC	Owner 185187793453
--	-------------------	---	-----------------------

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 instance

us-west-1.console.aws.amazon.com/ec2/home?region=us-west-1#instances:

EC2 > Instances

Instances (2) info

Last updated less than a minute ago

Connect Instance state Actions Launch instances

Find instance by attribute or tag (case-sensitive)

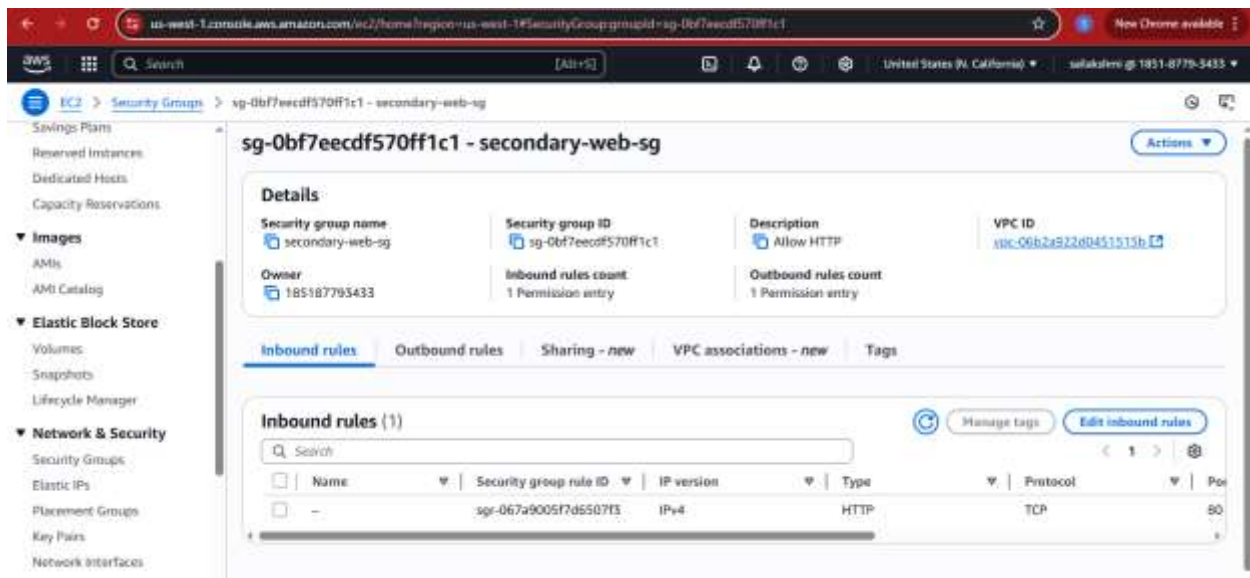
Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Secondary-We...	i-096dd09e3af0ed76f5	Running	t2.micro	2/2 checks passed	View alarms +	us-west-1c
Secondary-We...	i-007b0d02c656cbb95	Running	t2.micro	2/2 checks passed	View alarms +	us-west-1a

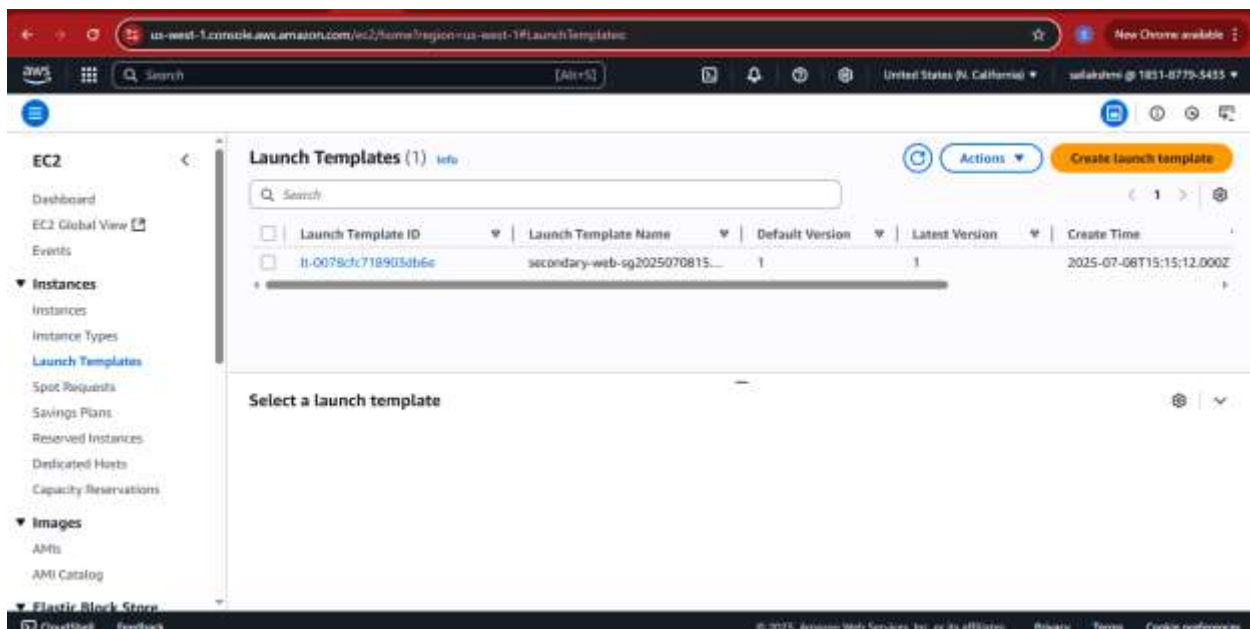
Select an instance

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

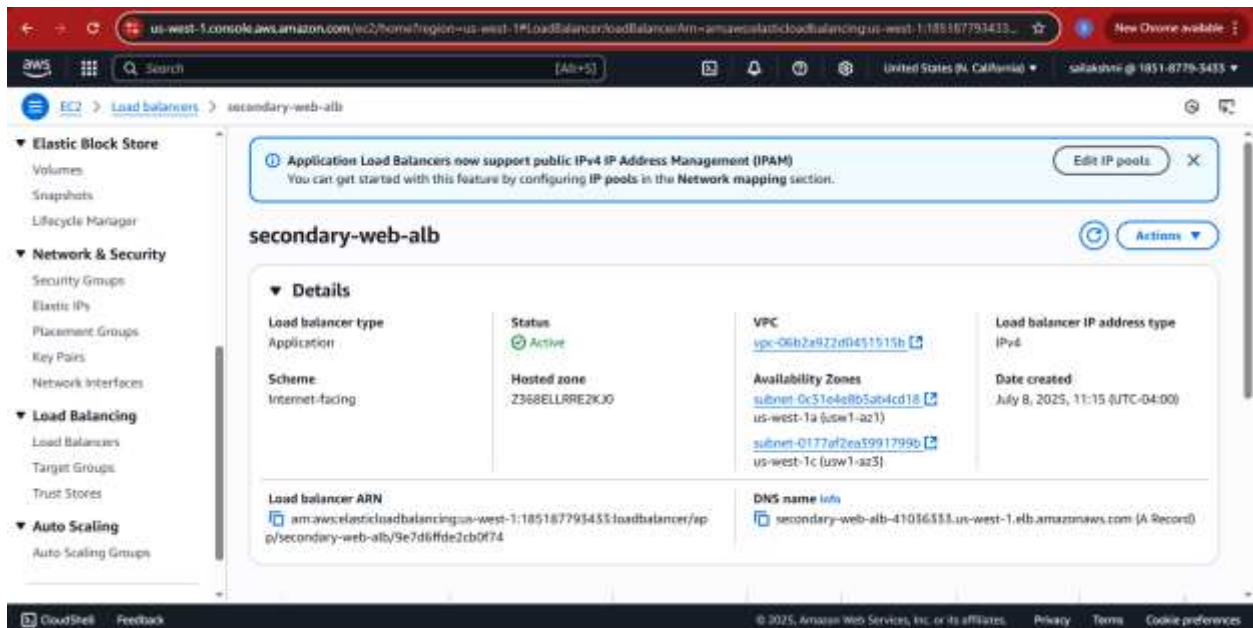
Security group



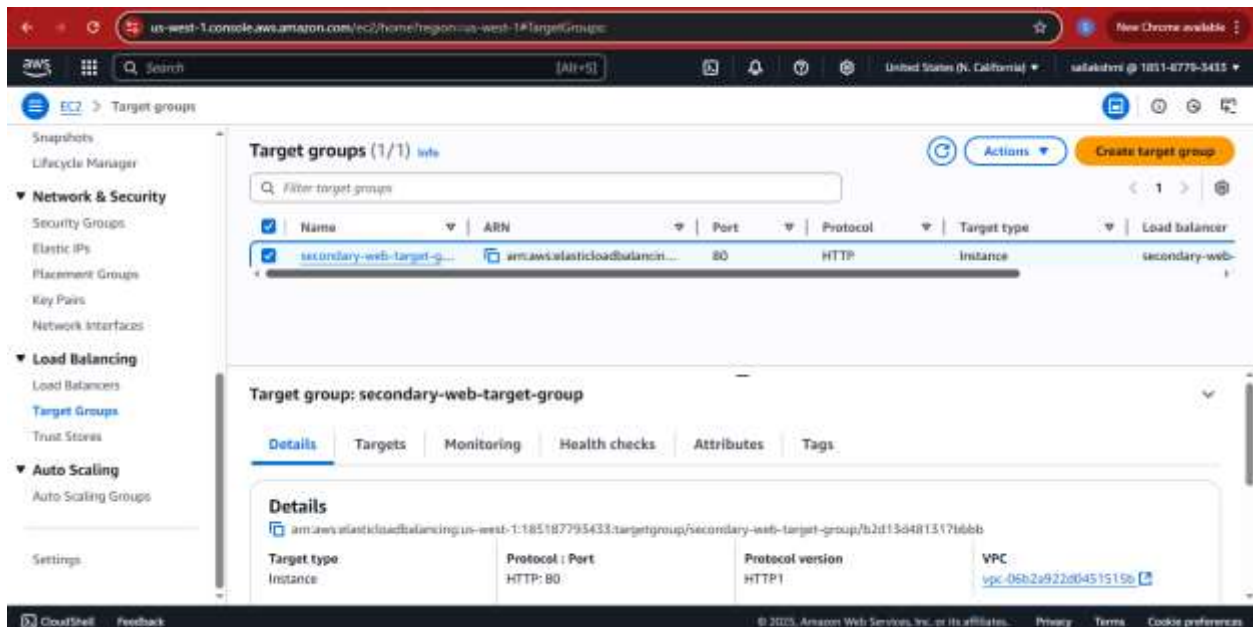
Launch template



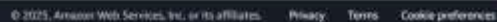
Load balancer



Target group



Autoscaling group created



Output check the dns name of primary load balancer in browser

<http://primary-web-alb-401762522.us-east-1.elb.amazonaws.com/>



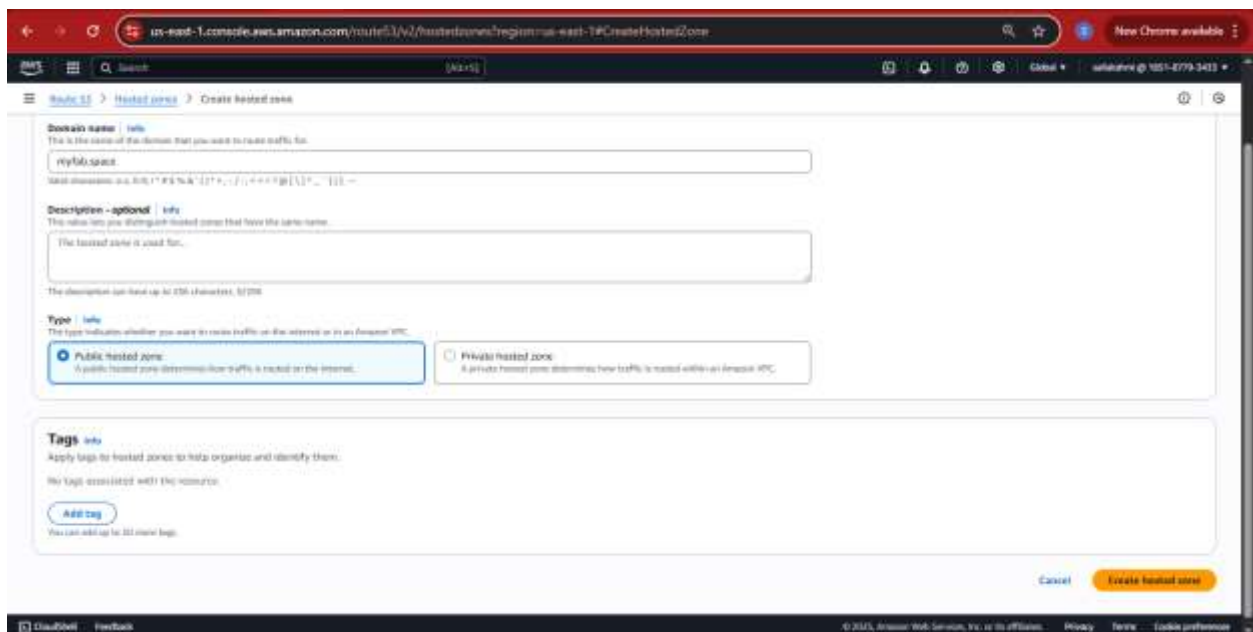
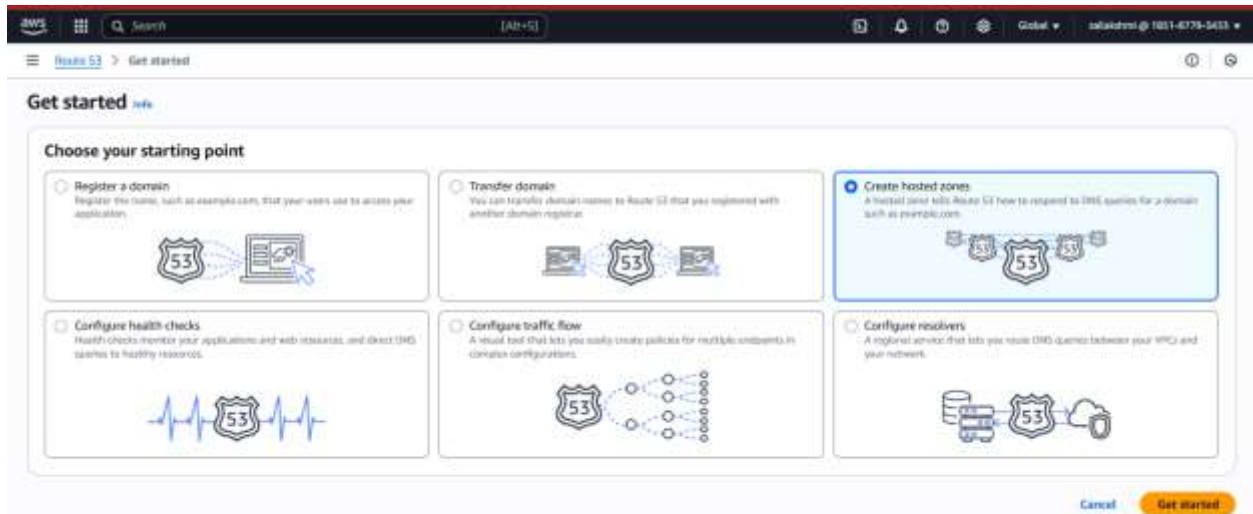
Secondary region

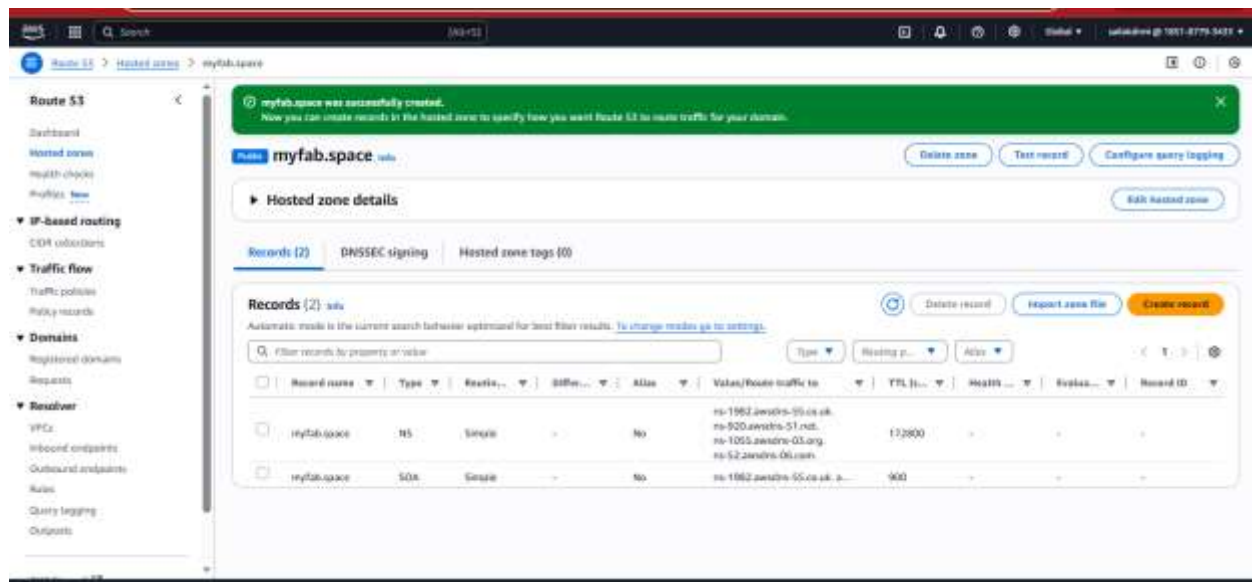
<http://secondary-web-alb-810409321.us-west-1.elb.amazonaws.com/>



Route 53

- Create a Hosted Zone in Route 53
- Go to AWS Console → Route 53 → Hosted Zones
- Click “Create hosted zone”
- Domain name: myfab.space
- Type: Public hosted zone
- Click Create
- You'll now see 4 NS records (like ns-123.awsdns-45.com, etc.)





- Update Nameservers in where you bought
- Go to Domain List → Click “Manage” on myfab.space
- Scroll to Nameservers
- Choose: Custom DNS
- Paste the 4 NS values from Route 53 here
- Save changes



Get ALB DNS Names from Terraform Outputs

primary_alb_dns = "primary-web-alb-401762522.us-east-1.elb.amazonaws.com"

secondary_alb_dns = "secondary-web-alb-810409321.us-west-1.elb.amazonaws.com"

Create Route 53 Health Check (for Primary ALB)

Go to Route 53 → Health Checks → Create Health Check

Name: primary-alb-health

Endpoint type: Public endpoint

Protocol: HTTP

Domain name: paste primary ALB DNS (e.g., primary-web-alb-xyz.us-east-1.elb.amazonaws.com)

Port: 80

Path: /

Advanced settings (leave defaults), click Create

The screenshot shows the 'Create health check' page in the AWS Management Console. The page is titled 'Create health check' and has a breadcrumb trail: 'Route 53 > Health checks > Create health check'. Below the title, it says 'Create a new health check to monitor the health of a specified resource.' The page is divided into several sections: 'Configuration', 'Specify endpoint by', 'Domain name', and 'Advanced configuration'. In the 'Configuration' section, the 'Name' field is set to 'primary-alb-health'. The 'Resource' section has three radio buttons: 'Endpoint' (selected), 'Calculated health check', and 'CloudWatch alarm'. The 'Specify endpoint by' section has two radio buttons: 'IP address' and 'Domain name' (selected). The 'Domain name' section has a dropdown menu set to 'HTTP' and a text field containing 'primary-web-alb-401762632.us-east-1.elb.amazonaws.com'. The 'Advanced configuration' section is expanded, showing 'Request interval' with 'Standard (30 seconds)' selected and 'Failure threshold' set to 3.

Create Route 53 Record Set – Failover (PRIMARY)

- Go to Route 53 → Hosted Zones → myfab.space
- Click “Create Record”
- Record name: leave blank (for root domain) or enter www if you want www.myfab.space
- Record type: A – IPv4 address
- Alias: Yes
- Alias target: Select Primary ALB
- Routing policy: Failover
- Failover type: Primary
- Attach Health Check: select the one you created
- Click Create records

us-east-1.console.aws.amazon.com/route53/v3/hostedzones#CreateRecordSetV2/011559WKSJNBYB518GQ5

Search [Name] [Go]

Route 53 Hosted zones myblu.state Create record

Quick create record [Switch to wizard](#)

Record 1 [Delete](#)

Record name [Info](#)
 myblu.state
Keep blank to create a record for the new domain.

Record type [Info](#)

Alias [Info](#)
☒ Yes

Route traffic to [Info](#)

Routing policy [Info](#)

Failover record type [Info](#)

Disable Primary to route traffic to the specified resource by default or Secondary to route traffic to the specified resource when the primary resource is unavailable. You can create only one failover record of each type.

Health check ID [Info](#)

Evaluate target health
☒ Yes

Record ID [Info](#)

[Add another record](#)

CloudShell Feedback

© 2025 Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create Route 53 Record Set – Failover (SECONDARY)

- Click “Create Record” again
- Same name: blank or www (must match above)
- Type: A – IPv4 address
- Alias: Yes
- Alias target: Select Secondary ALB
- Routing policy: Failover
- Failover type: Secondary
- No health check needed
- Click Create records

us-east-1.console.aws.amazon.com/route53/home#/CreateRecordSet/Z0115593K5NB78S18GQ5

Record 1

Record name: /myfab.space

Record type:

Alias

Route traffic to:

US West (N. California)

Routing policy:

Health check ID - optional:

Record ID:

Failover record type:

Evaluate target health: ☒ Yes

Test it

- Visit www.myfab.space
- It should load from Primary ALB
- Simulate failure by stopping EC2s in Primary ASG
- Wait 1–2 minutes → Route 53 routes to Secondary ALB



Successfully initiated stopping of i-059211aa3a4e11eaa

Instances (7/5) [help](#)

Find instances by attribute or tag (name=webserver) [All status](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Primary-Web-	i-077041a6880c5d915a	Terminated	t2.micro	-	View alarms	us-east-1a	-	-
Primary-Web-	i-0a8f9259390a8112f	Stopped	t2.micro	-	View alarms	us-east-1a	-	-
Primary-Web-	i-0fe348ac129c40366	Pending	t2.micro	-	View alarms	us-east-1a	-	\$4.102
Primary-Web-	i-032ca18000f5b187a	Terminated	t2.micro	-	View alarms	us-east-1b	-	-

i-059211aa3a4e11eaa (Primary-Web-A5G)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

Instances (5/5) [help](#)

Find instances by attribute or tag (name=webserver) [All status](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Primary-Web-	i-0a8f9259390a8112f	Stopped	t2.micro	-	View alarms	us-east-1a	-	-
Primary-Web-	i-059211aa3a4e11eaa	Stopped	t2.micro	-	View alarms	us-east-1b	-	-

From secondary ALB it loaded

Not secure myfab.space

Welcome to My Web App

This image is served from the same web server

