ECS on EC2 Nginx Server Deployment with ALB And EFS using Terraform (Task 6)

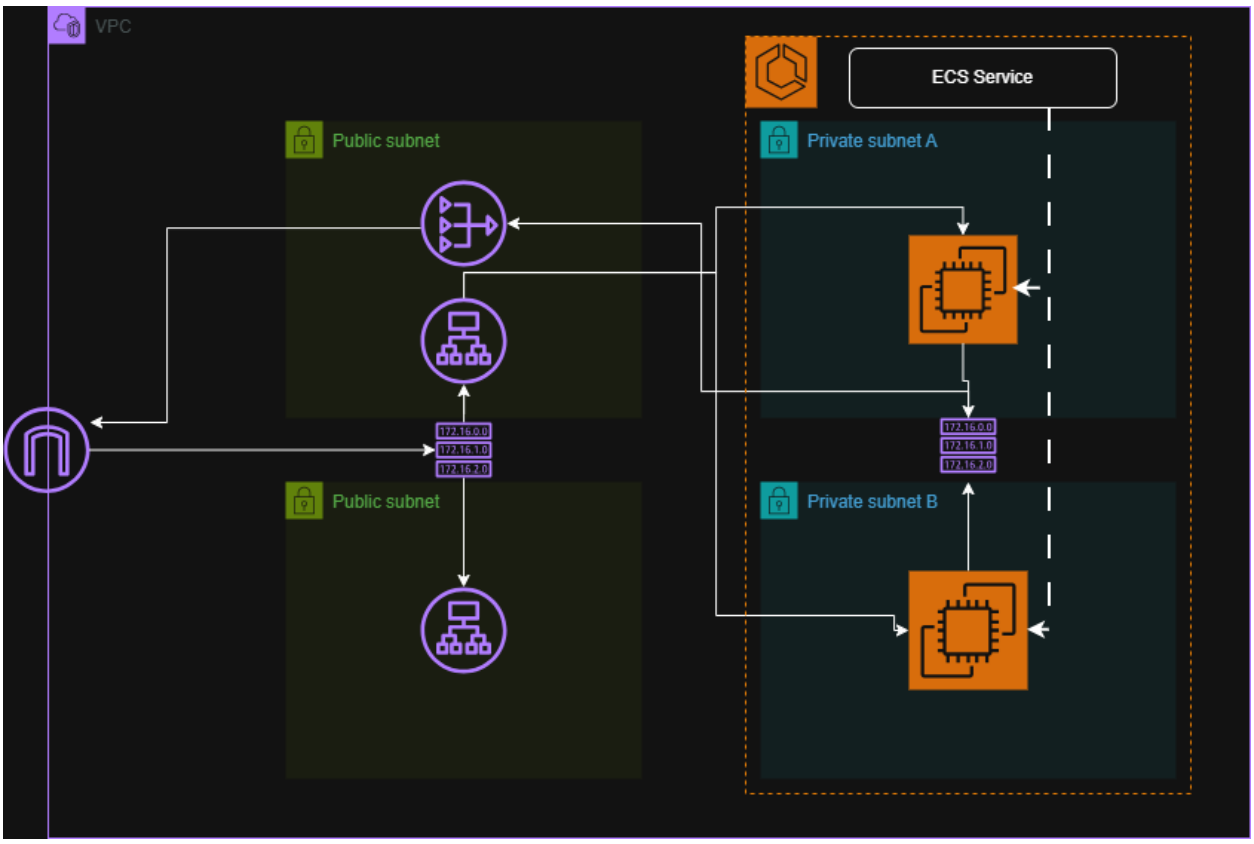
Zaeem Attique Ashar

Cloud Intern

Task Description:

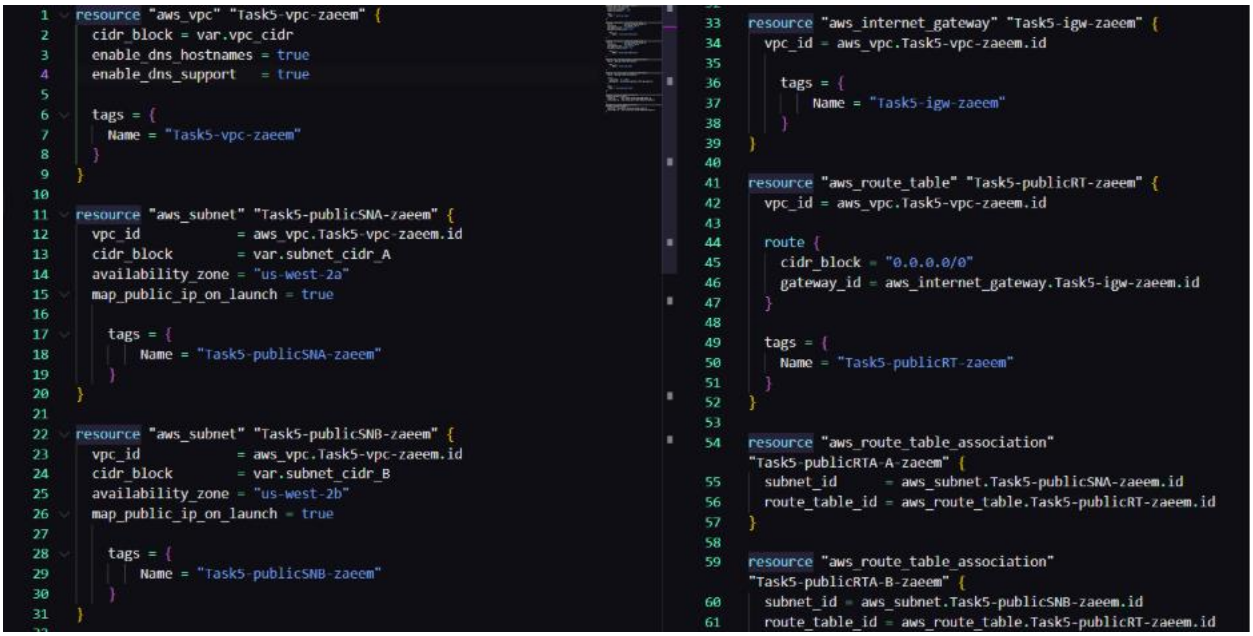
This task will be a guide on setting up a Nginx server with a highly available architecture on AWS ECS Cluster Spread over multiple availability zones. An internet facing Load Balancer will be set up in front of the ECS cluster to listen to traffic and balance it upon tasks. AWS EFS will be used for persistence in shared storage to prevent data loss.

Architecture Diagram:



Task6.1: Design and create a highly available VPC with multiple public and private subnets

- VPC Configuration:
    - CIDR: 10.0.0.0/16.
- Subnets Configuration:
    - Availability Zone A:
    - CIDR Block for subnet: 10.0.1.0/24.
    - Availability Zone: us-west-2a.
    - Availability Zone B:
    - CIDR Block for subnet: 10.0.2.0/24.
    - Availability Zone: us-west-2a.
- Internet Gateway Configuration:
    - VPC ID.
- Route Table (Public):
    - VPC ID.
    - Route: Destination 0.0.0.0/0, Target Internet Gateway.
- Security Group (Public):
    - Inbound SSH Traffic: Port 22, source 0.0.0.0/0.
    - Inbound HTTP Traffic: Port 80, source 0.0.0.0/0.



```
resource "aws_vpc" "Task5-vpc-zaeem" {
  cidr_block = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Name = "Task5-vpc-zaeem"
  }
}

resource "aws_subnet" "Task5-publicSNA-zaeem" {
  vpc_id            = aws_vpc.Task5-vpc-zaeem.id
  cidr_block        = var.subnet_cidr_A
  availability_zone = "us-west-2a"
  map_public_ip_on_launch = true

    tags = {
        Name = "Task5-publicSNA-zaeem"
    }
}

resource "aws_subnet" "Task5-publicSNB-zaeem" {
  vpc_id            = aws_vpc.Task5-vpc-zaeem.id
  cidr_block        = var.subnet_cidr_B
  availability_zone = "us-west-2b"
  map_public_ip_on_launch = true

    tags = {
        Name = "Task5-publicSNB-zaeem"
    }
}
```

```
resource "aws_internet_gateway" "Task5-igw-zaeem" {
  vpc_id = aws_vpc.Task5-vpc-zaeem.id

    tags = {
        Name = "Task5-igw-zaeem"
    }
}

resource "aws_route_table" "Task5-publicRT-zaeem" {
  vpc_id = aws_vpc.Task5-vpc-zaeem.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.Task5-igw-zaeem.id
  }

  tags = {
    Name = "Task5-publicRT-zaeem"
  }
}

resource "aws_route_table_association"
"Task5-publicRTA-A-zaeem" {
  subnet_id      = aws_subnet.Task5-publicSNA-zaeem.id
  route_table_id = aws_route_table.Task5-publicRT-zaeem.id
}

resource "aws_route_table_association"
"Task5-publicRTA-B-zaeem" {
  subnet_id = aws_subnet.Task5-publicSNB-zaeem.id
  route_table_id = aws_route_table.Task5-publicRT-zaeem.id
```

Task6.2: Create Launch Template and ASG

- Create a launch template that will be used by the ASG to deploy instances
    - Name prefix: Task6-ECS-Launch-Template-Zaeem
    - Image ID: AMI image ID from the EC2 catalogue
    - Instance Type: t3.micro
    - Instance Profile: ECS Instance Profile ARN
    - Network Interfaces: ec2 security group
    - User data script to register the instance with the cluster.
- Create an auto scaling group that will deploy the instances with launch template
    - Pass the launch template ID
    - Maximum size: 2
    - Minimum size: 1
    - Desired capacity: 1
    - Availability zones us-east-1a and 1b

```
resource "aws_launch_template" "Task6-ECS-Launch-Template-Zaeem" {
  name_prefix   = "Task6-ECS-Launch-Template-Zaeem"
  image_id      = "ami-0fa3fe0fa7920f68e"
  instance_type = "t3.micro"
  iam_instance_profile {
    arn = var.ecs_instance_profile_arn
  }
  network_interfaces {
    associate_public_ip_address = false
    security_groups             = [ var.ec2_sg_id ]
  }
  user_data = base64encode(<<-EOF
    #!/bin/bash
    sleep 30
    yum update -y
    yum install -y ecs-init docker
    systemctl start docker
    systemctl enable docker
    systemctl start ecs
    systemctl enable ecs

    cat <<'EOFCONFIG' >> /etc/ecs/ecs.config
    ECS_CLUSTER=Task6-ECS-Cluster-Zaeem
    ECS_ENABLE_TASK_IAM_ROLE=true
    ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
    ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
    EOFCONFIG

    yum install -y amazon-efs-utils
    mkdir -p /mnt/efs
  EOF
  )
}
```

```
resource "aws_autoscaling_group" "Task6-ECS-Auto-Scaling-Group-Zaeem" {
  depends_on = [ aws_launch_template.Task6-ECS-Launch-Template-Zaeem ]
  launch_template {
    id      = aws_launch_template.Task6-ECS-Launch-Template-Zaeem.id
    version = "$Latest"
  }
  min_size                  = 2
  max_size                  = 1
  desired_capacity          = 2
  availability_zones = [ "us-east-1a", "us-east-1b" ]
  health_check_type         = "EC2"
  health_check_grace_period = 300

  tag {
    key                 = "Name"
    value               = "Task6-ECS-Instance-Zaeem"
    propagate_at_launch = true
  }
}
```

Task6.3: Configure security groups for ALB, ECS tasks, and EFS access

- Security Groups Configuration:
  - For Application Load Balancer:
    - Ingress port 80, source 0.0.0.0/0
    - Egress port any, destination 0.0.0.0/0
  - For Elastic Container Service task:
    - Ingress port 80, source ALB Security Group
    - Egress port any, destination 0.0.0.0/0
  - For Elastic File System
    - Ingress port 2049, source ECS Security Group
    - Egress port any, destination 0.0.0.0/0
- IAM Roles for Task Execution and Task:
  - Task Role Policies:
    - elasticfilesystem:ClientRootAccess
    - elasticfilesystem:ClientWrite
    - elasticfilesystem:ClientMount
  - Task Execution Role Policies:
    - AmazonECSTaskExecutionRolePolicy

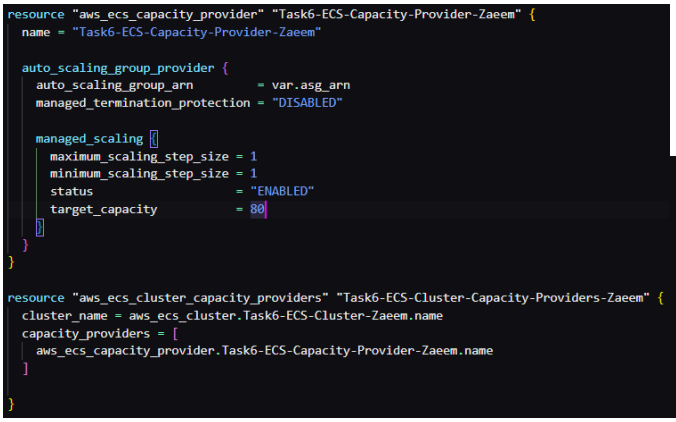
```
resource "aws_security_group" "Task5-ALB-SG-Zaeem" {
  name        = "Task5-ALB-SG-Zaeem"
  vpc_id      = var.vpc_id

  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" ]
  }

  tags = {
    Name = "Task5-ALB-SG-Zaeem"
  }
}
```

```
resource "aws_security_group" "Task5-EFS-SG-Zaeem" {
  name        = "Task5-EFS-SG-Zaeem"
  vpc_id      = var.vpc_id

  ingress {
    from_port = 2049
    to_port = 2049
    protocol = "tcp"
    security_groups = [ aws_security_group.Task5-ECS-SG-Zaeem.id ]
  }

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

  tags = {
    Name = "Task5-EFS-SG-Zaeem"
  }
}
```

Task6.4: Create an ECS cluster with Terraform

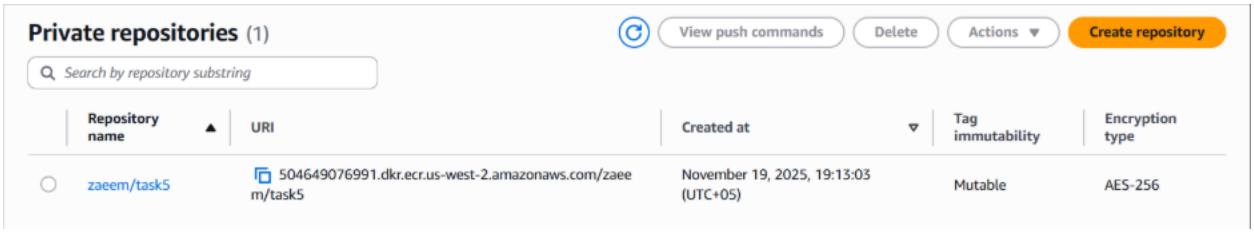
- ECS Cluster Definition
  - Name: Task6-ECS-Cluster-Zaeem

- ECS Service Configuration:
    - Name: Task6-ECS-Service-Zaeem
    - Cluster ID passed
    - Task definition ARN passed
    - Desired Count: 2
    - Launch Type: Fargate
    - Load Balancer Configuration:
        - Target Group ARN passed
        - Container Name given.
        - Container port: Port that the container will listen on 80
    - Network Configuration:
        - Public Subnet A ID and Public Subnet B ID passed.
        - ECS Security Group ID passed.
        - Assign Public IP option turned on.
- ECS Cluster Capacity Provider:
    - Pass the ASG ARN
    - Managed Scaling Step size: 1
    - Minimum Scaling Step Size: 1
    - Status: Enabled
    - Target Capacity: 80



Task6.5: Build and push a custom Docker image to a container registry

1. Create a repository in the Elastic Container Registry (ECR).



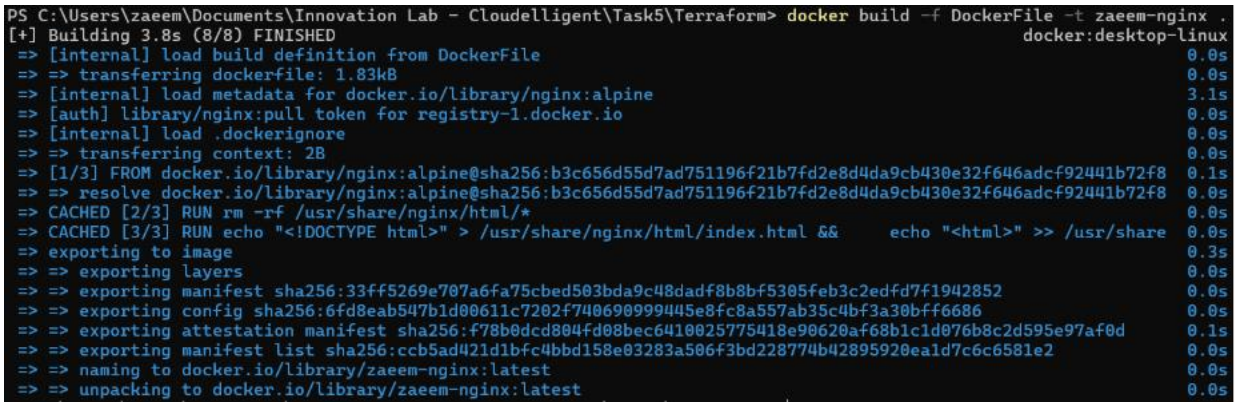
2. Write the nginx server installation and configuration in a DockerFile in the root directory.

```
FROM nginx:alpine

# Remove default content
RUN rm -rf /usr/share/nginx/html/*

# Copy custom HTML files (if you have a directory with your website)
# COPY ./website /usr/share/nginx/html

# Create a simple custom page
RUN echo "<!DOCTYPE html>" > /usr/share/nginx/html/index.html && \
    echo "<html>" >> /usr/share/nginx/html/index.html && \
    echo "<head>" >> /usr/share/nginx/html/index.html && \
    echo "    <title>Task5 - Zaeem</title>" >> /usr/share/nginx/html/index.html && \
    echo "    <style>" >> /usr/share/nginx/html/index.html && \
    echo "        body { font-family: Arial, sans-serif; text-align: center; padding: 50px; }" >> /usr/share/nginx/html/index.html && \
    echo "        h1 { color: &#333; }" >> /usr/share/nginx/html/index.html && \
    echo "    </style>" >> /usr/share/nginx/html/index.html && \
    echo "</head>" >> /usr/share/nginx/html/index.html && \
    echo "<body>" >> /usr/share/nginx/html/index.html && \
    echo "    <h1>🚀 Task5 - ECS with Nginx</h1>" >> /usr/share/nginx/html/index.html && \
    echo "    <p>Successfully deployed on ECS Fargate</p>" >> /usr/share/nginx/html/index.html && \
    echo "    <p>Created by: Zaeem</p>" >> /usr/share/nginx/html/index.html && \
    echo "    <p>Container ID: ${HOSTNAME}</p>" >> /usr/share/nginx/html/index.html && \
    echo "</body>" >> /usr/share/nginx/html/index.html && \
    echo "</html>" >> /usr/share/nginx/html/index.html

# Copy custom nginx config (if needed)
# COPY nginx-custom.conf /etc/nginx/nginx.conf

# Expose port 80
EXPOSE 80

# Health check (optional)
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
    CMD curl -f http://localhost/ || exit 1

# Start Nginx in foreground
CMD ["nginx", "-g", "daemon off;"]
```

Explanation: Using the base image nginx:alpine which is a slim image of Nginx. Write the custom webpage in HTML and save it to /usr/share/nginx/html/index.html. Expose port 80 to listen to HTTP traffic. Use CMD directive to run the command ‘nginx -g daemon off;” inside the container CLI to run the Nginx service and keep the task running.

3. Build the DockerFile into an Image

Command: *docker build –f <file name> -t <tag name> .*



4. Push the image to the ECR:

Command: *docker push <ECR Repository URI>*

Task6.6: Define ECS task definition to use the custom Docker image and mount EFS volumes

- ECS Task Definition
  - Family: Task6-Zaeem
  - Task Role ARN: ARN of the IAM Role created for the task.
  - Execution Role ARN: ARN of the IAM Role created for execution of task.
  - Cluster Mode: FARGATE
  - Network Mode: bridge
  - CPU: 256
  - Memory: 512
  - Container definition:
    - Image: 504649076991.dkr.ecr.us-west-2.amazonaws.com/zaeem/Task6
    - Port Mappings: Container port 80 -> Host port 80
    - Mount points: Source volume Task6-EFS-Zaeem, container path /mnt/data
    - Volume Configuration: EFS ID passed, Root dir “/”, encryption enabled, port 2999, Access Point ID passed

```
resource "aws_ecs_task_definition" "Task5-TaskDefinition" {
  depends_on = [ var.efs_id ]
  family = "Task5-Zaeem"
  task_role_arn = var.Task5-ecs-task-role-arn
  execution_role_arn = var.Task5-ecs-task-execution-role-arn
  requires_compatibilities = ["FARGATE"]
  network_mode = "awsvpc"
  cpu = "256"
  memory = "512"
  container_definitions = jsonencode([
    {
      name       = "NginxServer"
      image      = "504649076991.dkr.ecr.us-west-2.amazonaws.com/zaeem/task5:latest"
      # cpu        = 256
      # memory     = 512
      essential = true
      portMappings = [
        {
          containerPort = 80
          hostPort      = 80

        }
      ]
      mountPoints = [
        {
          sourceVolume = "Task5-EFS-Zaeem"
          containerPath = "/mnt/data"

        }
      ]
    }
  ])

  volume {
    name = "Task5-EFS-Zaeem"

    efs_volume_configuration {
      file_system_id          = var.efs_id
      root_directory          = "/"
      transit_encryption      = "ENABLED"
      transit_encryption_port = 2999
      authorization_config {
        access_point_id = var.efs_ap_id
        iam             = "ENABLED"
      }
    }
  }

}
```

Task6.2: Create ECS and Cluster Capacity Providers

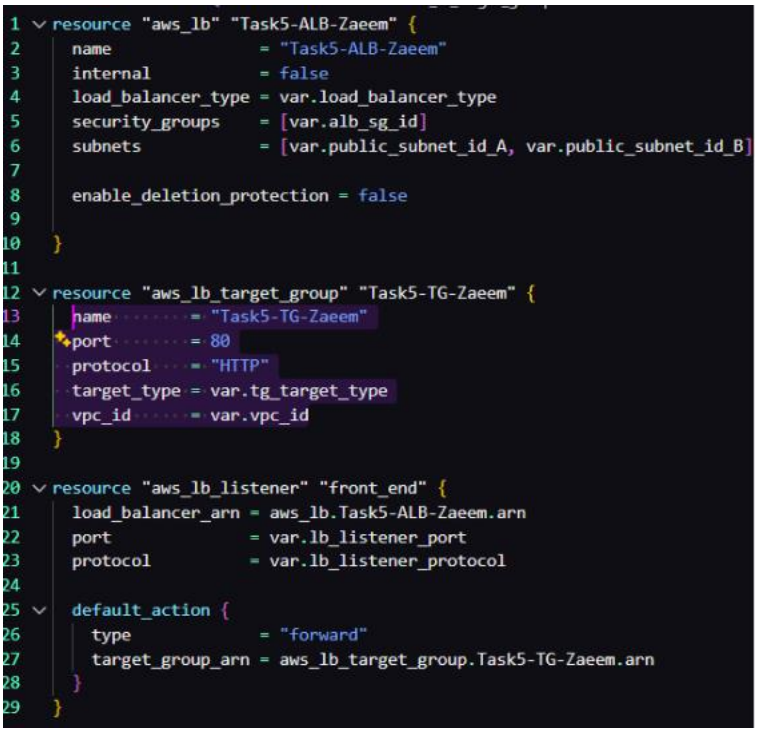
- Provide ASG ARN
- Disable termination protection for terraform destroy
- Maximum Scaling Step Size: 1
- Minimum Scaling Step Size: 1
- Target Capacity: 80% CPU Usage
- For Cluster Capacity provider:
    - Pass the Cluster Name
    - Pass the Capacity Provider Name

```
resource "aws_ecs_capacity_provider" "Task6-ECS-Capacity-Provider-Zaeem" {
  name = "Task6-ECS-Capacity-Provider-Zaeem"

  auto_scaling_group_provider {
    auto_scaling_group_arn         = var.asg_arn
    managed_termination_protection = "DISABLED"

    managed_scaling {
      maximum_scaling_step_size = 1
      minimum_scaling_step_size = 1
      status                    = "ENABLED"
      target_capacity           = 80
    }
  }
}

resource "aws_ecs_cluster_capacity_providers" "Task6-ECS-Cluster-Capacity-Providers-Zaeem" {
  cluster_name = aws_ecs_cluster.Task6-ECS-Cluster-Zaeem.name
  capacity_providers = [
    aws_ecs_capacity_provider.Task6-ECS-Capacity-Provider-Zaeem.name
  ]

}
```

Task6.7: Create an Application Load Balancer (ALB) with proper listeners and target groups

- Load Balancer Configuration:
    - Name: Task6-ALB-Zaeem
    - Internal: False (The ALB is internet facing)
    - Load Balancer Type: Application
    - Security Groups: ALB Security Group ID attached
    - Subnets: Public Subnet A ID and Public Subnet B ID passed
    - Enable Deletion Protection: false (To allow terraform to manage the ELB)
- Target Group Configuration:
    - Name: Task6-TG-Zaeem
    - Port: 80

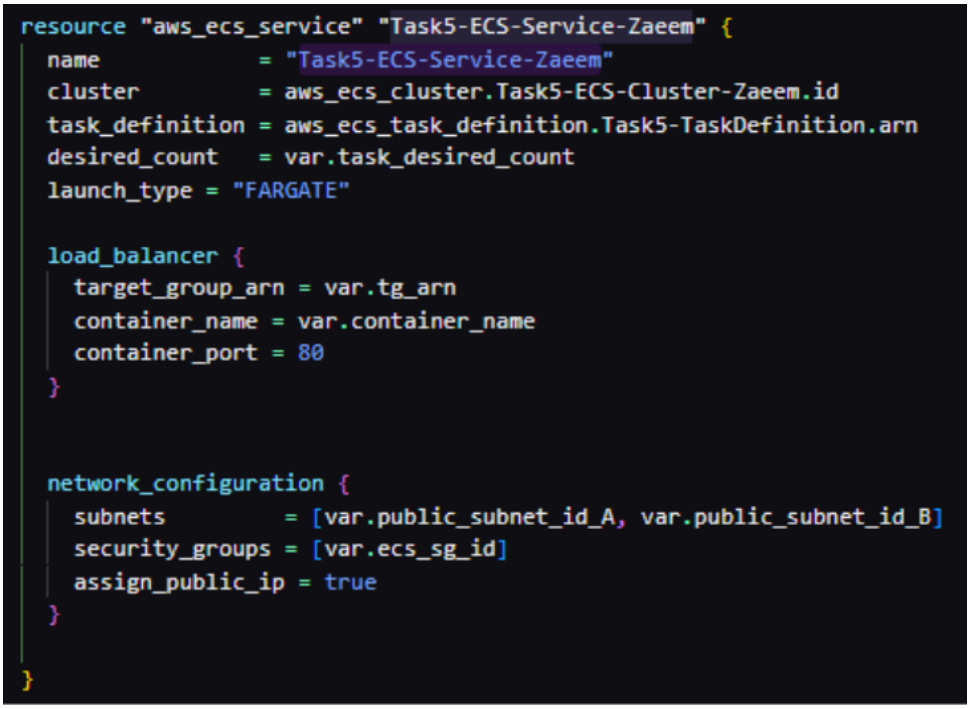
    - Protocol: HTTP
    - Target Type: IP (ECS will attach the Task IP)
    - VPC ID: Task 6 VPC ID passed
- Load Balancer Listener:
    - Load Balancer ARN passed
    - Port to listen on: 80
    - Protocol to listen for: HTTP
    - Default Action: Forward traffic to Target Group ARN



```
resource "aws_lb" "Task5-ALB-Zaeem" {
  name               = "Task5-ALB-Zaeem"
  internal           = false
  load_balancer_type = var.load_balancer_type
  security_groups    = [var.alb_sg_id]
  subnets            = [var.public_subnet_id_A, var.public_subnet_id_B]

  enable_deletion_protection = false

}

resource "aws_lb_target_group" "Task5-TG-Zaeem" {
  name        = "Task5-TG-Zaeem"
  port        = 80
  protocol    = "HTTP"
  target_type = var.tg_target_type
  vpc_id      = var.vpc_id
}

resource "aws_lb_listener" "front_end" {
  load_balancer_arn = aws_lb.Task5-ALB-Zaeem.arn
  port              = var.lb_listener_port
  protocol          = var.lb_listener_protocol

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.Task5-TG-Zaeem.arn
  }
}
```

Task6.8: Configure ECS service to use EC2 with ALB integration for load balancing

- ECS Service Configuration:
  - Name: Task6-ECS-Service-Zaeem
  - Cluster ID passed
  - Task definition ARN passed
  - Desired Count: 2
  - Launch Type: EC2
  - Load Balancer Configuration:
    - Target Group ARN passed
    - Container Name given.
    - Container port: Port that the container will listen on 80



```
resource "aws_ecs_service" "Task5-ECS-Service-Zaeem" {
  name            = "Task5-ECS-Service-Zaeem"
  cluster         = aws_ecs_cluster.Task5-ECS-Cluster-Zaeem.id
  task_definition = aws_ecs_task_definition.Task5-TaskDefinition.arn
  desired_count   = var.task_desired_count
  launch_type = "FARGATE"

  load_balancer {
    target_group_arn = var.tg_arn
    container_name = var.container_name
    container_port = 80
  }


  network_configuration {
    subnets         = [var.public_subnet_id_A, var.public_subnet_id_B]
    security_groups = [var.ecs_sg_id]
    assign_public_ip = true
  }

}
```

Task6.9: Set up and mount EFS for persistent storage in ECS tasks

- Add the local mount point for the EFS volume in the container configuration under the mountPoints block:
    - Source Volume: Name of the EFS Volume
    - Container Path: /mnt/data
- Configure the Volume block for the task:
    - Name: Name the volume
    - Under the efs_volume_configuration:
        - ♣ File system ID: EFS ID passed
        - ♣ Root Directory: "/" Default
        - ♣ Encryption: Enabled
        - ♣ Encryption Port: 2999
        - ♣ Access Point ID passed.

Task6.10: Attach IAM roles and policies for ECS tasks and EFS access

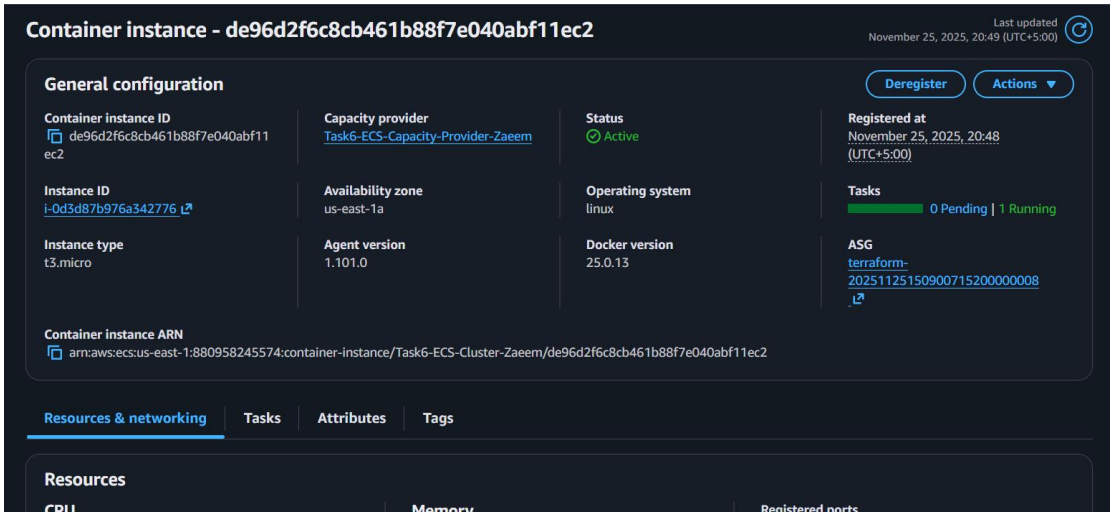
Under ECS Task Definition, we need to attach the roles for Task Execution and the Task running.

- Task Role ARN passed
- Execution Role ARN passed

```
resource "aws_ecs_task_definition" "Task5-TaskDefinition" {
  depends_on = [ var.efs_id ]
  family = "Task5-Zaeem"
  task_role_arn = var.Task5-ecs-task-role-arn
  execution_role_arn = var.Task5-ecs-task-execution-role-arn
  requires_compatibilities = ["FARGATE"]
  network_mode = "awsvpc"
  cpu = "256"
  memory = "512"
```

Task6.11: Verify scalability and availability

To verify availability, delete one of the running tasks. In response, the service running should detect that the cluster is not at the desired state of 2 running tasks and should automatically provision another task form the task definition.



To access the home page of the Nginx Server, enter the DNS name into the browser.

Task6.12: Difficulties Faced