**Case Study: Case Study t - Creating an Architecture using Terraform on AWS**

**Objectives**

XYZ Corporation had the following technical requirements for their infrastructure:

1. **Cloud Provider**: AWS
2. **Software**: Apache2 (for hosting web content)
3. **Operating System**: Ubuntu
4. **Architecture**:
   * A VPC to isolate resources.
   * Two public subnets for high availability.
   * One EC2 instance in each subnet with Apache2 installed.
   * A security group to handle inbound and outbound traffic.
   * Internet gateway and network interfaces to ensure external access.

**Challenges**

The company was focused on creating a highly available and resilient architecture for their web services while maintaining cost-efficiency. They required:

* Easy access for users to their web applications hosted on EC2 instances.
* Automatic provisioning of infrastructure to avoid manual configuration errors.
* Secure network configuration, allowing only necessary traffic (HTTP and SSH).

**Solution: AWS and Terraform Integration**

To meet the company’s requirements, a solution was designed using AWS and Terraform to automate the entire infrastructure creation. This involved:

1. **Creating a Virtual Private Cloud (VPC)**: A VPC was created to isolate the instances and other resources within a private, controlled network.
2. **Two Public Subnets**: Two subnets were created within the VPC in separate availability zones, providing a highly available architecture.
3. **EC2 Instances in Each Subnet**: Instances were provisioned in each subnet, with an Apache2 web server installed to host the company’s applications.
4. **Security Group**: A security group was configured to allow inbound HTTP (port 80) and SSH (port 22) traffic, while allowing all outbound traffic.
5. **Internet Gateway and Routing**: An Internet Gateway was attached to the VPC, enabling the instances to communicate with the internet. Routing was configured to allow internet traffic through the gateway.

**Detailed Implementation**

The Terraform configuration was used to define and deploy the architecture:

**Terraform Configuration:**

provider "aws" {

  region = "us-east-1"  # AWS region: N. Virginia

}

# VPC Creation

resource "aws\_vpc" "main\_vpc" {

  cidr\_block = "10.0.0.0/16"

  enable\_dns\_support = true

  enable\_dns\_hostnames = true

  tags = { Name = "company-vpc" }

}

# Internet Gateway

resource "aws\_internet\_gateway" "main\_igw" {

  vpc\_id = aws\_vpc.main\_vpc.id

  tags = { Name = "company-igw" }

}

# Subnets Creation (2 Subnets)

resource "aws\_subnet" "subnet\_1" {

  vpc\_id = aws\_vpc.main\_vpc.id

  cidr\_block = "10.0.1.0/24"

  map\_public\_ip\_on\_launch = true

  tags = { Name = "company-subnet-1" }

}

resource "aws\_subnet" "subnet\_2" {

  vpc\_id = aws\_vpc.main\_vpc.id

  cidr\_block = "10.0.2.0/24"

  map\_public\_ip\_on\_launch = true

  tags = { Name = "company-subnet-2" }

}

# Route Table & Association

resource "aws\_route\_table" "main\_route\_table" {

  vpc\_id = aws\_vpc.main\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.main\_igw.id

  }

  tags = { Name = "company-route-table" }

}

resource "aws\_route\_table\_association" "subnet\_1\_rta" {

  subnet\_id = aws\_subnet.subnet\_1.id

  route\_table\_id = aws\_route\_table.main\_route\_table.id

}

resource "aws\_route\_table\_association" "subnet\_2\_rta" {

  subnet\_id = aws\_subnet.subnet\_2.id

  route\_table\_id = aws\_route\_table.main\_route\_table.id

}

# Security Group

resource "aws\_security\_group" "instance\_sg" {

  vpc\_id = aws\_vpc.main\_vpc.id

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 22

    to\_port     = 22

    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 = "company-instance-sg" }

}

# EC2 Instances with Apache2 Installation

resource "aws\_instance" "instance\_1" {

  ami                    = "ami-0e86e20dae9224db8"

  instance\_type          = "t2.micro"

  subnet\_id              = aws\_subnet.subnet\_1.id

  vpc\_security\_group\_ids = [aws\_security\_group.instance\_sg.id]

  associate\_public\_ip\_address = true

  user\_data = <<-EOF

              #!/bin/bash

              apt update -y

              apt install apache2 -y

              systemctl start apache2

              EOF

  tags = { Name = "instance-1-subnet-1" }

}

resource "aws\_instance" "instance\_2" {

  ami                    = "ami-0e86e20dae9224db8"

  instance\_type          = "t2.micro"

  subnet\_id              = aws\_subnet.subnet\_2.id

  vpc\_security\_group\_ids = [aws\_security\_group.instance\_sg.id]

  associate\_public\_ip\_address = true

  user\_data = <<-EOF

              #!/bin/bash

              apt update -y

              apt install apache2 -y

              systemctl start apache2

              EOF

  tags = { Name = "instance-2-subnet-2" }

}

# Output Public IP Addresses

output "instance\_1\_public\_ip" {

  value = aws\_instance.instance\_1.public\_ip

}

output "instance\_2\_public\_ip" {

  value = aws\_instance.instance\_2.public\_ip

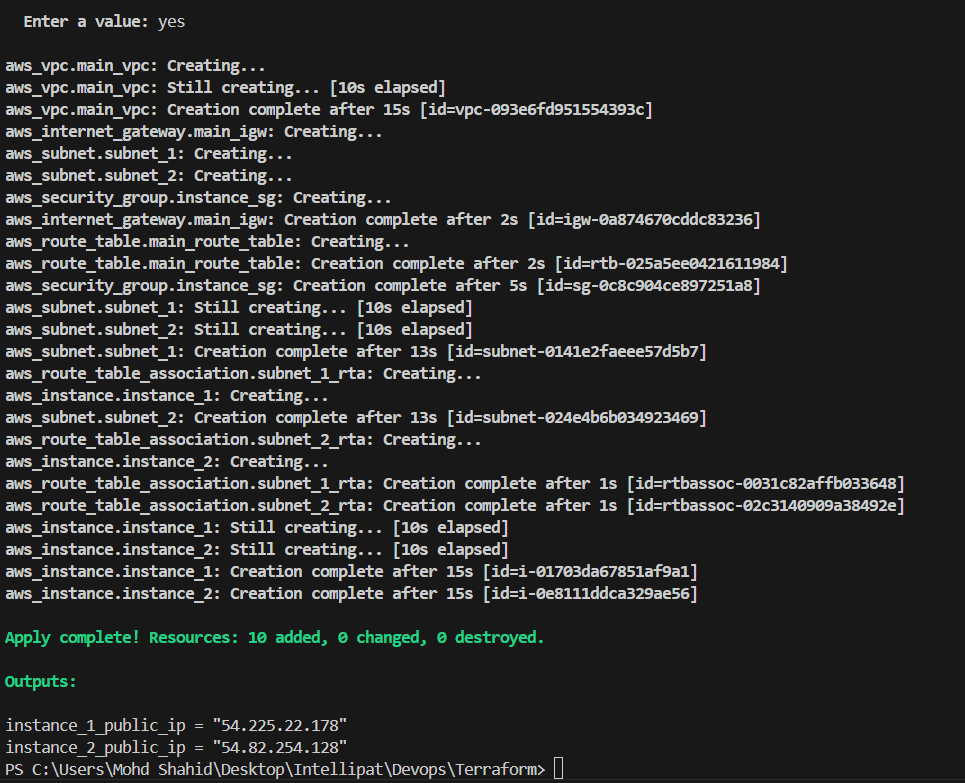
}

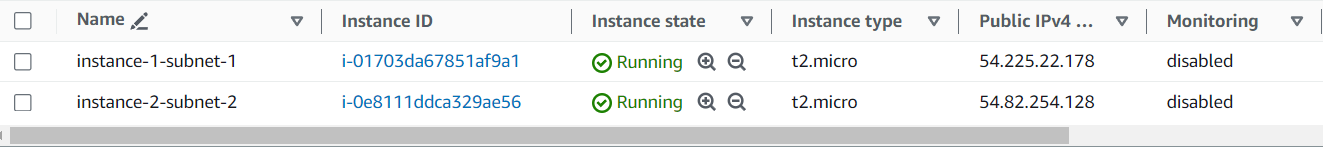
**Benefits and Results**

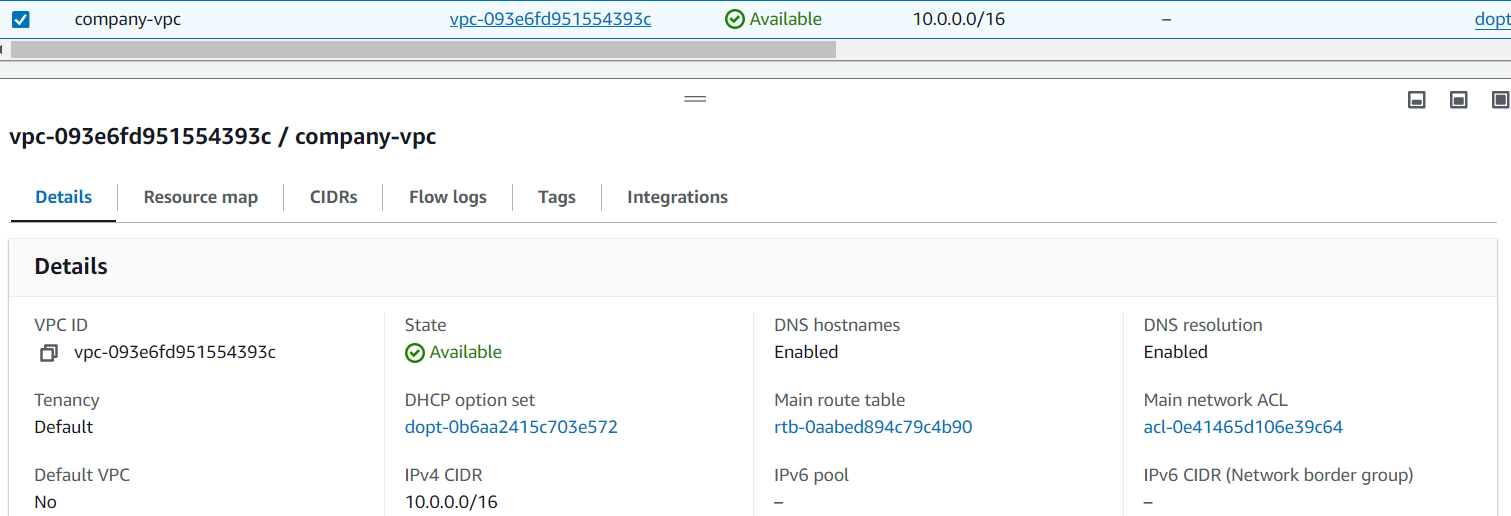
The solution delivered the following benefits:

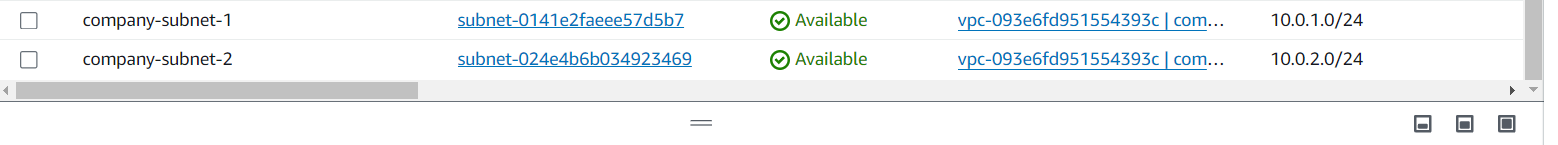
1. **High Availability**: With two EC2 instances across separate subnets, the web service was made highly available.
2. **Automated Infrastructure Deployment**: By using Terraform, XYZ Corporation was able to avoid manual configuration and ensure repeatable, error-free deployments.
3. **Security**: The security group limited access to HTTP and SSH traffic, reducing exposure to attacks.
4. **Scalability**: The architecture allowed for easy scaling by adding more subnets and instances in different availability zones.

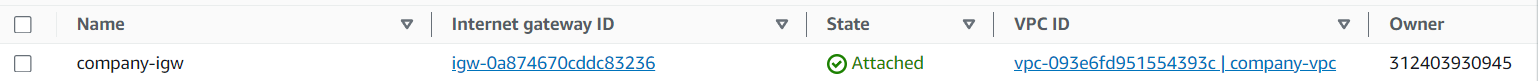
**OUTPUT:**

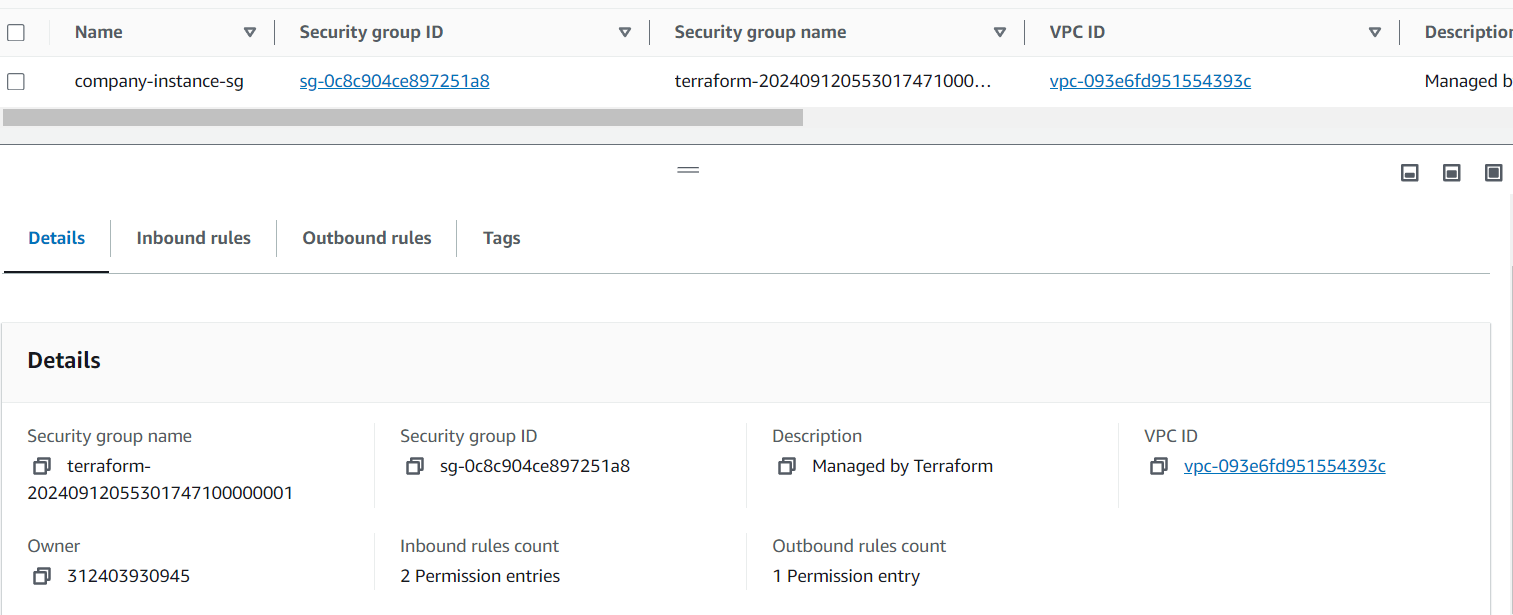
****

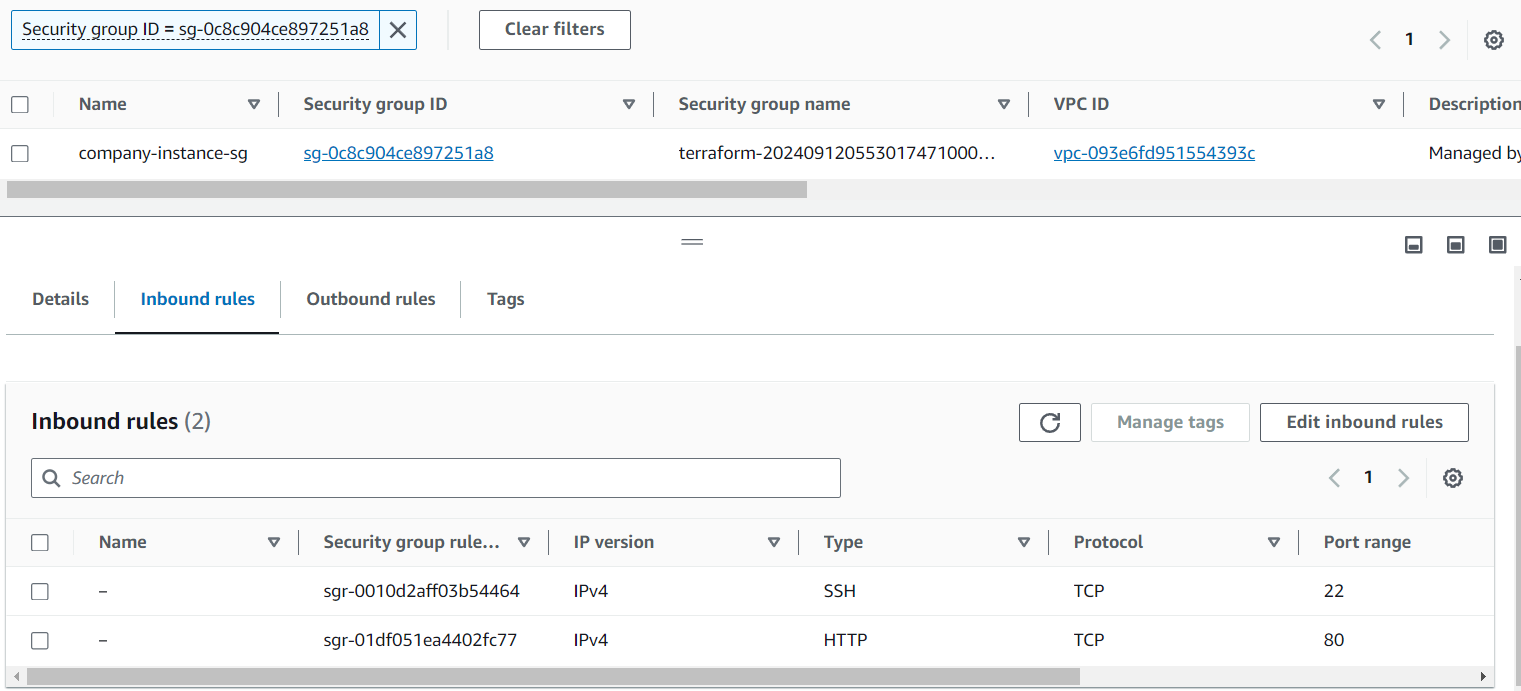
****

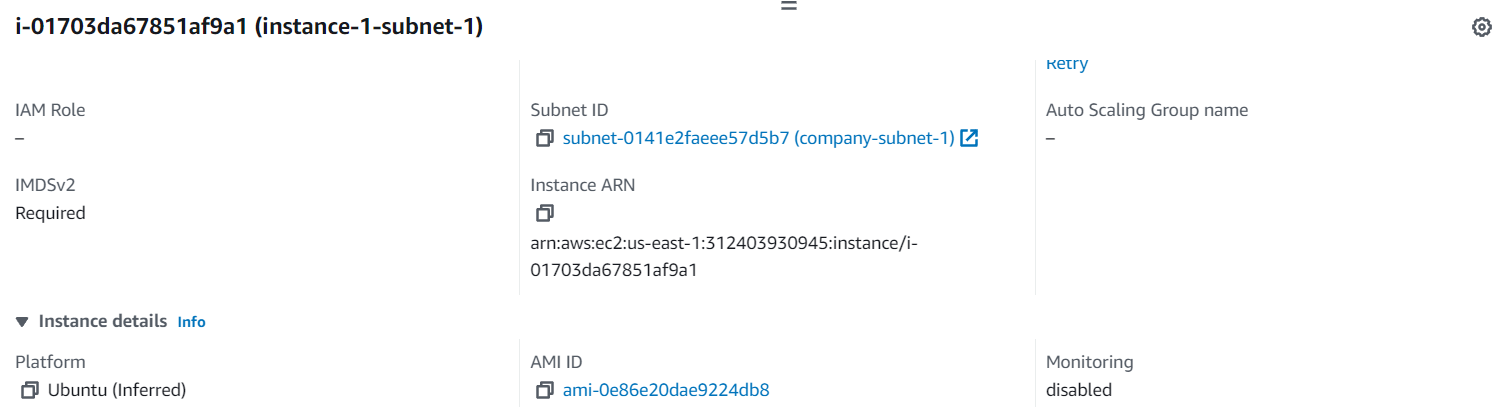
****

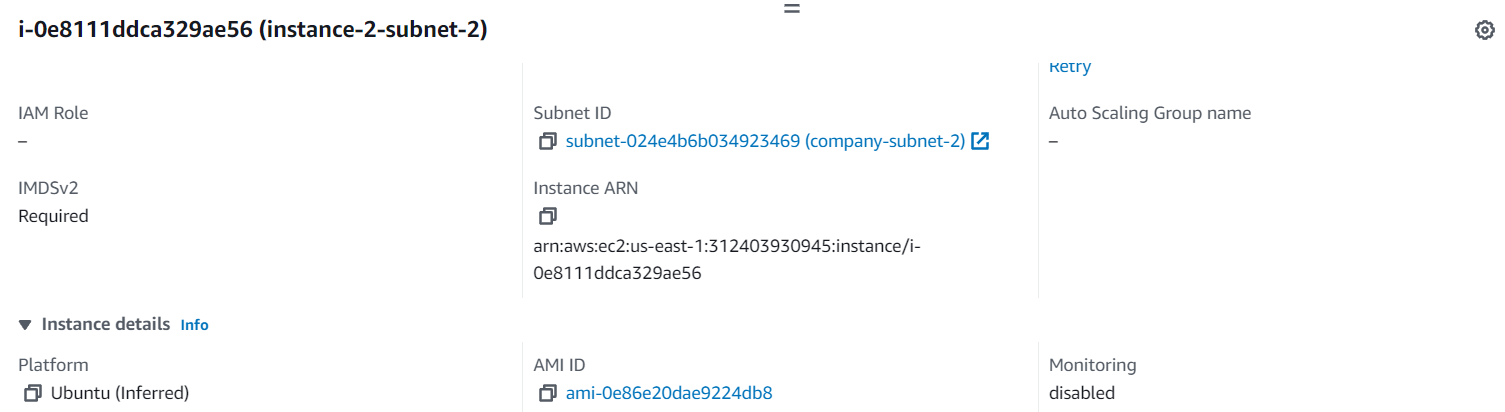
****

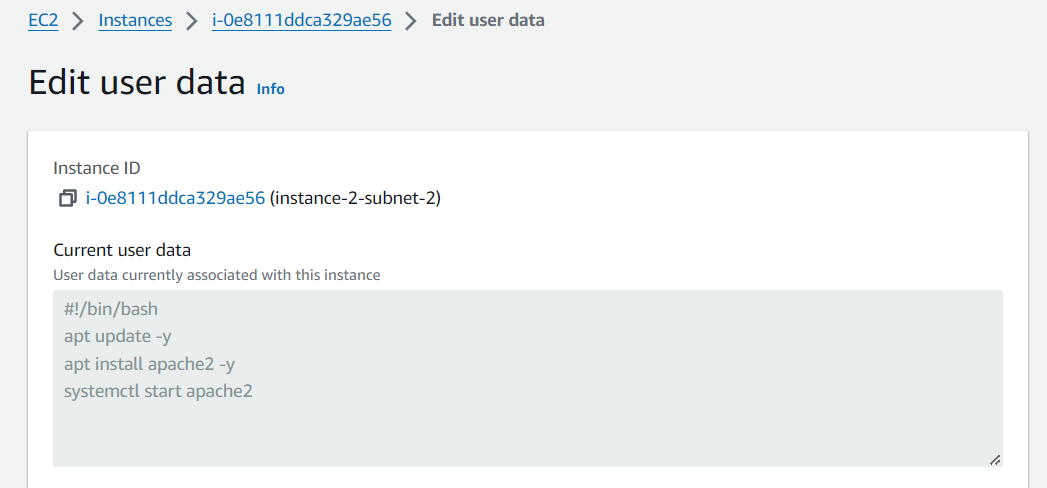
****

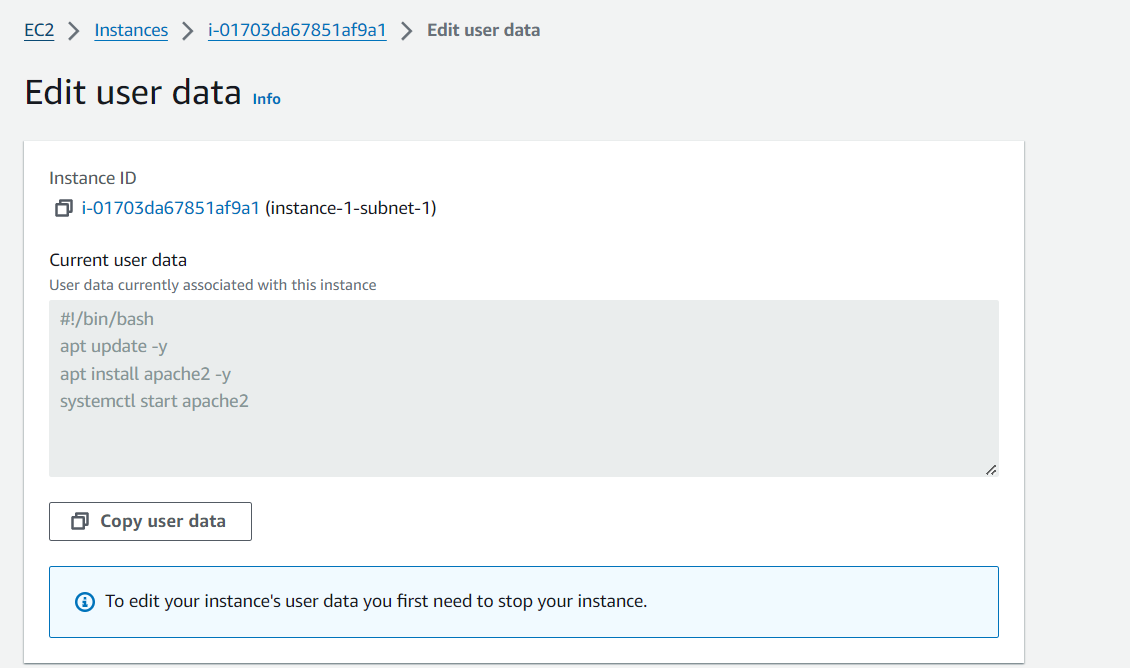
****

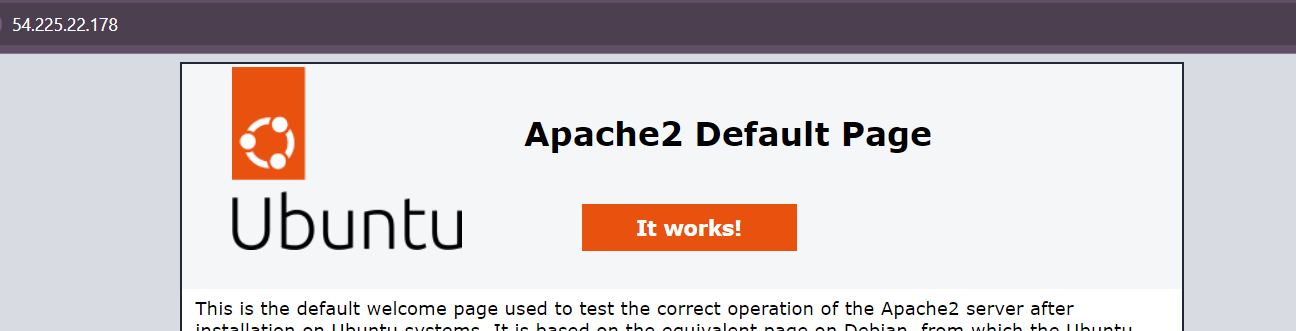
****

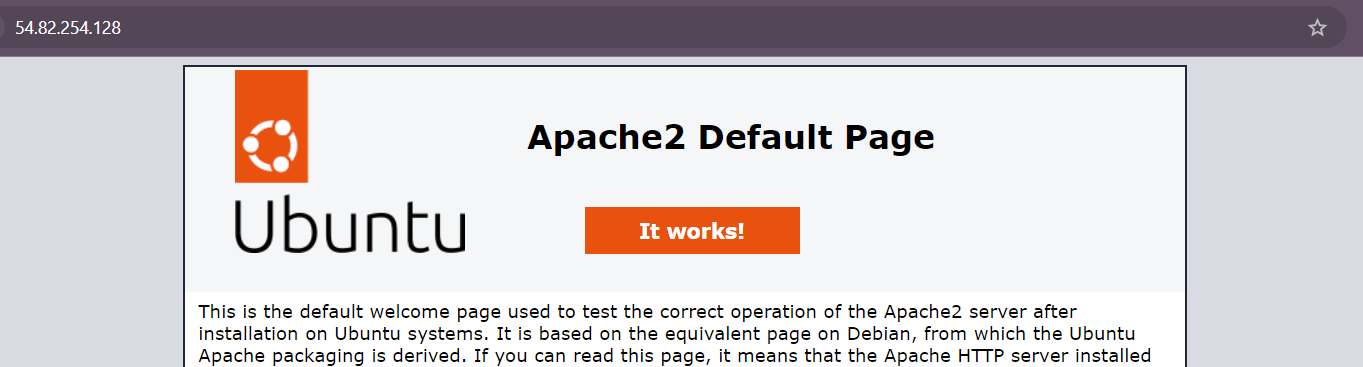
****

****

****

****

****

****