CASE STUDY -CREATING AN ARCHITECTURE USING TERRAFORM ON AWS

You work as a DevOps Engineer in leading Software Company. You have been asked to build an infrastructure safely and efficiently. **The company Requirements:**

1. Use AWS cloud Provider and the software to be installed is Apache2
2. Use Ubuntu AMI

The company wants the Architecture to have the following services:

1. Create a template with a VPC, 2 subnets and 1 instance in each subnet
2. Attach Security groups, internet gateway and network interface to the instance

Solution:

1) Create a Directory:
   mkdir terraform-aws-architecture
   cd terraform-aws-architecture
2) Create a main.tf File:

```
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true

  tags = {
    Name = "my-vpc"
  }
}

# Create Subnets
resource "aws_subnet" "subnet1" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    Name = "subnet-1"
  }
}

resource "aws_subnet" "subnet2" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.2.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    Name = "subnet-2"
  }
}

# Create an Internet Gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.my_vpc.id
  tags = {
    Name = "my-igw"
  }
}
```

```
# Create a Route Table
resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.my_vpc.id

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

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

# Associate Route Table with Subnets
resource "aws_route_table_association" "subnet1_association" {
  subnet_id      = aws_subnet.subnet1.id
  route_table_id = aws_route_table.public_route_table.id
}

resource "aws_route_table_association" "subnet2_association" {
  subnet_id      = aws_subnet.subnet2.id
  route_table_id = aws_route_table.public_route_table.id
}

# Create a Security Group
resource "aws_security_group" "apache_sg" {
  vpc_id = aws_vpc.my_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 = "apache-sg"
  }
}

# Create EC2 Instances
resource "aws_instance" "instance1" {
  ami           = "ami-04b4f1a9cf54c11d0"
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.subnet1.id
  security_groups = [aws_security_group.apache_sg.name]

  # Install Apache2
  user_data = <<-EOF
    #!/bin/bash
    sudo apt update -y
    sudo apt install apache2 -y
    sudo systemctl start apache2
    sudo systemctl enable apache2
  EOF

  tags = {
    Name = "Instance-1"
  }
}

resource "aws_instance" "instance2" {
  ami           = "ami-04b4f1a9cf54c11d0"
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.subnet2.id
  security_groups = [aws_security_group.apache_sg.name]

  # Install Apache2
  user_data = <<-EOF
    #!/bin/bash
    sudo apt update -y
    sudo apt install apache2 -y
    sudo systemctl start apache2
    sudo systemctl enable apache2
  EOF
```

```
  tags = {
    Name = "Instance-2"
  }
}

# Outputs
output "instance1_public_ip" {
  value = aws_instance.instance1.public_ip
}

output "instance2_public_ip" {
  value = aws_instance.instance2.public_ip
}
```

3) Deploy the Architecture:
   Initialize:

```
ubuntu@ip-172-31-7-35:~/terraform-project$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.91.0

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.
ubuntu@ip-172-31-7-35:~/terraform-project$
```

i-0c2ef8ee473810c4e (my-terraform)

PublicIPs: 3.110.54.55 PrivateIPs: 172.31.7.35

Plan:

```
Plan: 10 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance1_public_ip = (known after apply)
  + instance2_public_ip = (known after apply)


Note: You didn't use the -out option to save this plan
ubuntu@ip-172-31-7-35:~/terraform-project$
```

i-0c2ef8ee473810c4e (my-terraform)

PublicIPs: 3.110.54.55 PrivateIPs: 172.31.7.35

Apply:

```
aws_instance.instance1: Creating...
aws_instance.instance2: Creating...
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance2: Still creating... [10s elapsed]
aws_instance.instance1: Creation complete after 15s [id=i-0424657458bb2b651]
aws_instance.instance2: Creation complete after 15s [id=i-0bc43e3d5694e5371]

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

Outputs:

instance1_public_ip = "98.83.208.252"
instance2_public_ip = "3.237.183.240"
ubuntu@ip-172-31-7-35:~/terraform-project$
```

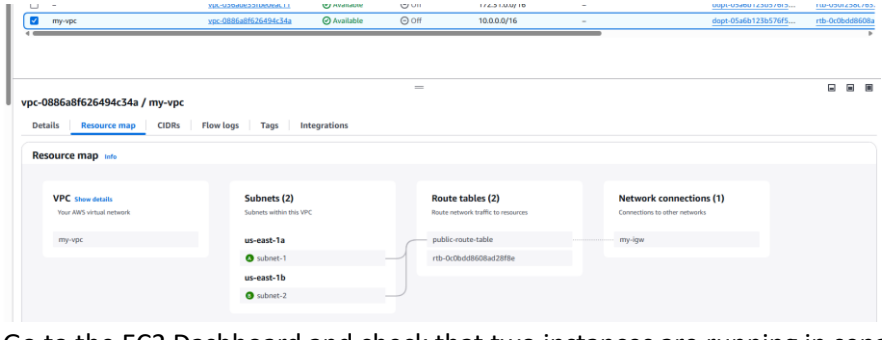
i-0c2ef8ee473810c4e (my-terraform)

PublicIPs: 3.110.54.55 PrivateIPs: 172.31.7.35

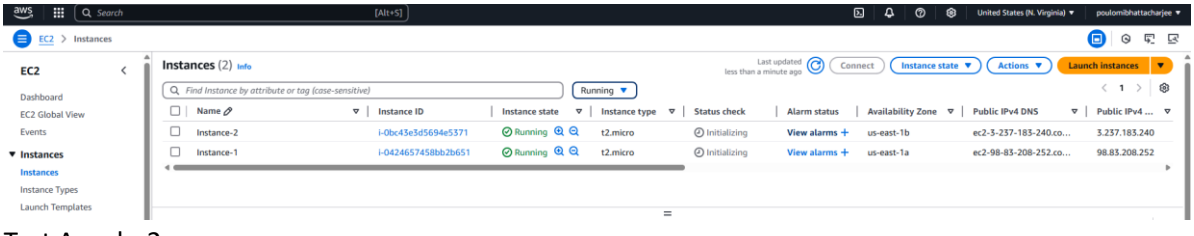
4) Verify the Deployment

Access the AWS Console:

Go to the VPC Dashboard and confirm the VPC, subnets, and Internet Gateway are created.



Go to the EC2 Dashboard and check that two instances are running in separate subnets.

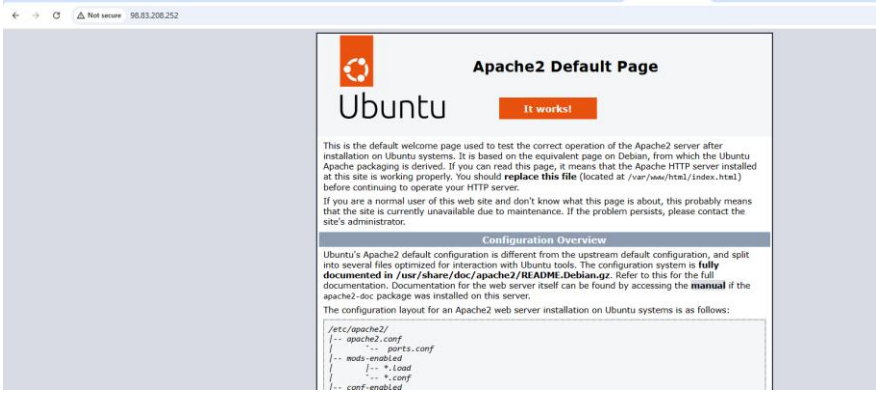


5) Test Apache2:

Copy the public IPs of both instances (outputted by Terraform or visible in the EC2 Dashboard).

Open a browser and navigate to http://<public-ip> to verify that the Apache2 web server is running.


Not secure 3.237.183.240
Apache2 Default Page
Ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
| `-- ports.conf
|-- mods-enabled


Not secure 98.83.208.252
Apache2 Default Page
Ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
| `-- ports.conf
|-- mods-enabled
| |-- *.load
| `-- *.conf
|-- conf-enabled