

Create and Deploy an AWS Infrastructure with Terraform

- Sakshi Rahane

Project Documentation

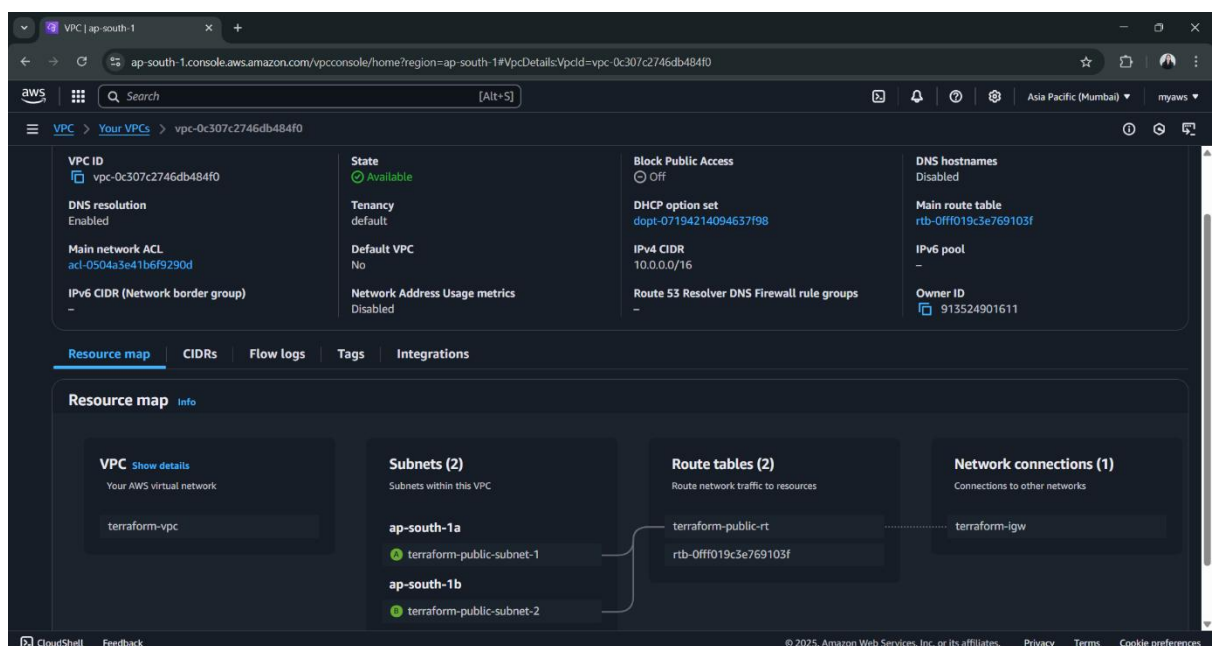
Objective:

Use Terraform to create and deploy a fully functional Virtual Private Cloud (VPC) in AWS, along with other components such as subnets, EC2 instance, Security group, Internet gateway. Goal is to Implements the infrastructure customize the configurations, and deploy a static application on an EC2 instance.

Task implementations:

1. VPC Creation:

- Create a VPC.
 - Created VPC with CIDR 10.0.0.0/16
 - VPC named as **terraform-vpc**
- Include at least two public subnets in the VPC.
 - **terraform-public-subnet-1** in availability zone **ap-south-1a** with CIDR 10.0.1.0/24
 - **terraform-public-subnet-2** in availability zone **ap-south-1b** with CIDR 10.0.2.0/24



2. Security Group:

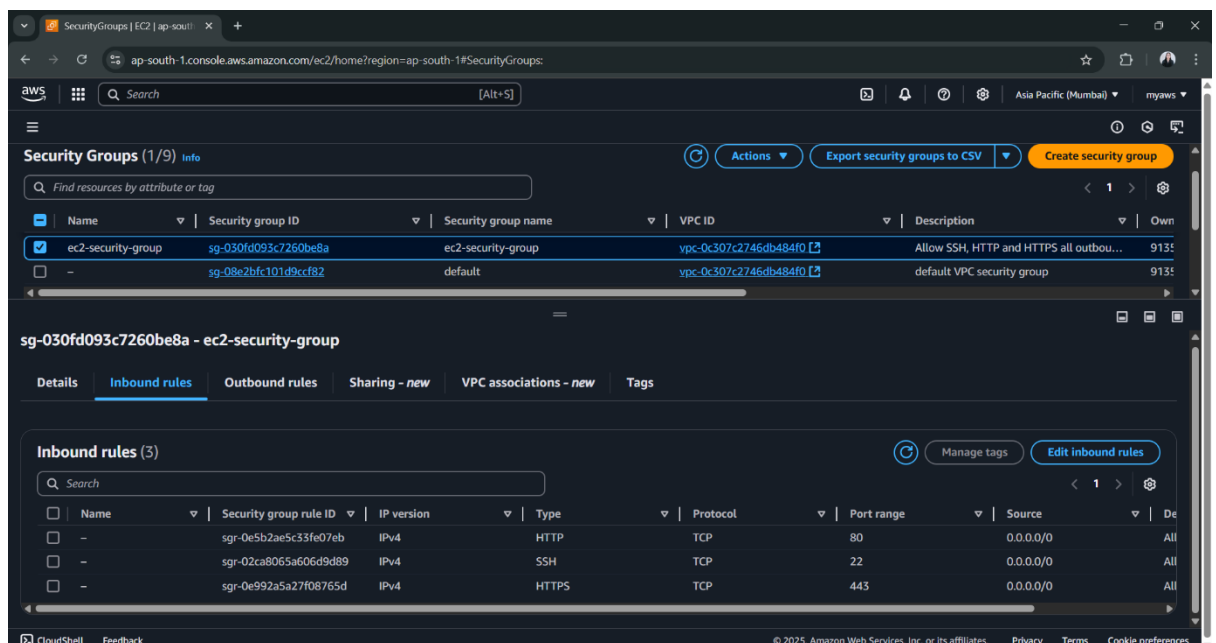
Configure a Security Group for the EC2 instance.

- Security group named as **ec2-security-group**
- Exposed necessary ports:

Port 80 (HTTP)

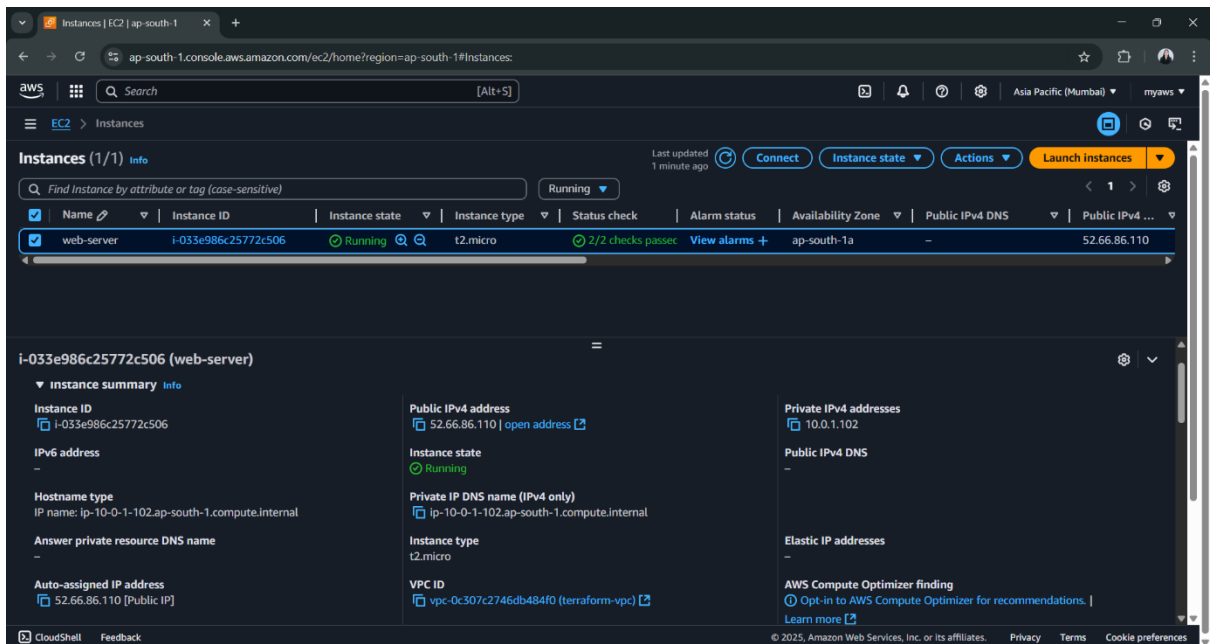
Port 22 (SSH)

Port 443 (HTTPS)



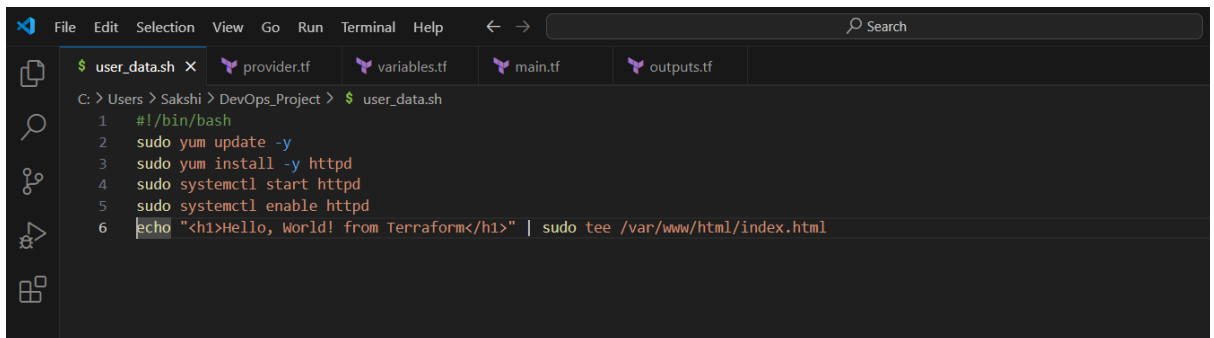
3. EC2 Instance:

- Deployed an EC2 instance within **terraform-public-subnet-1** one of the public subnet of **terraform-vpc** VPC.
- EC2 instance named as **web-server**
- AMI used "**ami-05c179eced2eb9b5b**"
- Instance type used **t2.micro**
- Used previously created Security Group for EC2 instance launching



- Used an User Data Script to:
 - Installed the httpd service.
 - Started the httpd service.
 - Hosted a static web application.

Deploys “Hello, World! From Terraform” HTML Page



Terraform files:

1. provider.tf

Terraform AWS Provider setup

This file specifies the required AWS provider (version 5.75.0 from HashiCorp) and Terraform version (≥ 1.2), and configures the AWS provider to use the region defined in the `aws_region` variable.

2. main.tf

This file creates resources VPC, Subnets, Internet Gateway, Route table, Subnet Association with Route table, Security Group, Access User Data Script, EC2 Instance.

3. variables.tf

This file contains variables for `aws_region`, `vpc_cidr`, `public_subnet_1_cidr`, `public_subnet_2_cidr`, `availability_zone_1`, `availability_zone_2`, `ami_id`, `instance_type`, `ssh_cidr`.

4. outputs.tf

This file returns the public ip of ec2 web-server instance

5. user_data.sh

This script runs on ec2 web-server instance. Updates the yum package manager, installs the Apache web server (httpd), starts the service, enables it to start on boot, and creates a simple "Hello, World! from Terraform" webpage in the default Apache document root.

Terraform Instructions:

1. Terraform Initialization

terraform init

```
PS C:\Users\Sakshi\DevOps_Project> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.75.0"...
- Finding latest version of hashicorp/template...
- Installing hashicorp/aws v5.75.0...
- Installed hashicorp/aws v5.75.0 (signed by HashiCorp)
- Installing hashicorp/template v2.2.0...
- Installed hashicorp/template v2.2.0 (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.
```

2. Terraform validate checks

terraform validate

```
PS C:\Users\Sakshi\DevOps_Project> terraform validate
Success! The configuration is valid.
```

3. Terraform plan execution

terraform plan

```
PS C:\Users\Sakshi\DevOps_Project> terraform plan
data.template_file.user_data: Reading...
data.template_file.user_data: Read complete after 0s [id=767ff8e7c8a77fc6ed8335448ca68e15f6df4a36f075f7b782b6364e3918f01a]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.web_server will be created
+ resource "aws_instance" "web_server" {
  ami           = "ami-05c179eced2eb9b5b"
  arn           = (known after apply)
  associate_public_ip_address = true
```

4. Terraform apply the planed execution

terraform apply

```
Windows PowerShell
Plan: 9 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ public_ip = (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

aws_vpc.main: Creating...
aws_vpc.main: Creation complete after 2s [id=vpc-0c307c2746db484f0]
aws_internet_gateway.gw: Creating...
aws_subnet.public_subnet_2: Creating...
aws_subnet.public_subnet_1: Creating...
aws_security_group.ec2_sg: Creating...
aws_internet_gateway.gw: Creation complete after 0s [id=igw-09607e61de9526232]
aws_route_table.public_route_table: Creating...
aws_route_table.public_route_table: Creation complete after 1s [id=rtb-0c512a86924de86a1]
aws_security_group.ec2_sg: Creation complete after 2s [id=sg-030fd093c7260be8a]
aws_subnet.public_subnet_2: Still creating... [10s elapsed]
aws_subnet.public_subnet_1: Still creating... [10s elapsed]
aws_subnet.public_subnet_1: Creation complete after 11s [id=subnet-06650f738cd21a0e8]
aws_subnet.public_subnet_2: Creation complete after 11s [id=subnet-056026457a24f183c]
aws_route_table_association.public_subnet_1_assoc: Creating...
aws_route_table_association.public_subnet_2_assoc: Creating...
aws_instance.web_server: Creating...
aws_route_table_association.public_subnet_2_assoc: Creation complete after 0s [id=rtbassoc-073bf9b62895f0e69]
aws_route_table_association.public_subnet_1_assoc: Creation complete after 0s [id=rtbassoc-072630bfff03caf982]
aws_instance.web_server: Still creating... [10s elapsed]
aws_instance.web_server: Still creating... [20s elapsed]
aws_instance.web_server: Still creating... [30s elapsed]
aws_instance.web_server: Creation complete after 32s [id=i-033e986c25772c506]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.

Outputs:
public_ip = "52.66.86.110"
PS C:\Users\Sakshi\DevOps_Project> |
```

5. Terraform destroy created resources

terraform destroy

```
Windows PowerShell
PS C:\Users\Sakshi\DevOps_Project> terraform destroy
data.template_file.user_data: Read complete after 0s [id=767ff8e7c0a77fc6ed8335448ca68e15f6df4a36f075f7b782b6364e3918f01a]
aws_vpc.main: Refreshing state... [id=vpc-0c307c2746db484f0]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-06650f738cd21a0e8]
aws_internet_gateway.gw: Refreshing state... [id=igw-09607e61de9526232]
aws_subnet.public_subnet_2: Refreshing state... [id=subnet-056026457a24f183c]
aws_security_group.ec2_sg: Refreshing state... [id=sg-030fd093c7260be8a]
aws_route_table.public_route_table: Refreshing state... [id=rtb-0c512a86924de86a1]
aws_instance.web_server: Refreshing state... [id=i-033e986c25772c506]
aws_route_table_association.public_subnet_1_assoc: Refreshing state... [id=rtbassoc-072630bff03caf982]
aws_route_table_association.public_subnet_2_assoc: Refreshing state... [id=rtbassoc-073bf9b62895f0e69]

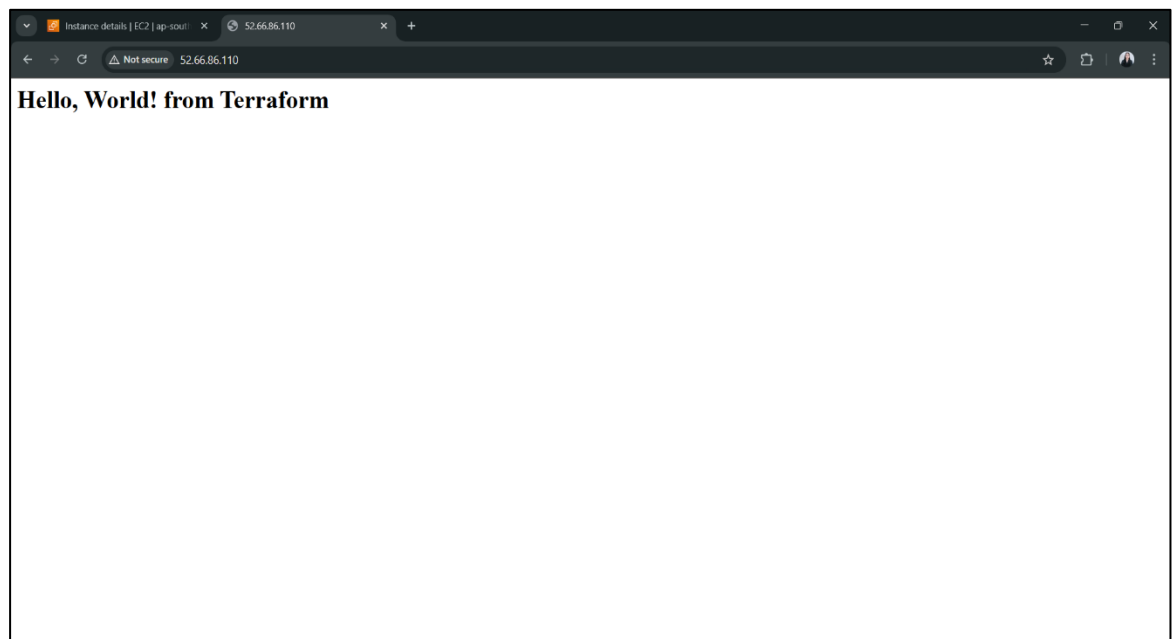
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.web_server will be destroyed
- resource "aws_instance" "web_server" {
  - ami                    = "ami-05c179eced2eb9b5b" -> null
  - arn                   = "arn:aws:ec2:ap-south-1:913524901611:instance/i-033e986c25772c506" -> null
  - associate_public_ip_address = true -> null
  - availability_zone      = "ap-south-1a" -> null
  - cpu_core_count         = 1 -> null
  - cpu_threads_per_core   = 1 -> null
  - disable_api_stop       = false -> null
  - disable_api_termination = false -> null
  - ebs_optimized          = false -> null
  - get_password_data      = false -> null
  - hibernation            = false -> null
  - id                    = "i-033e986c25772c506" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state         = "running" -> null
  - instance_type          = "t2.micro" -> null
  - ipv6_address_count     = 0 -> null
  - ipv6_addresses        = [] -> null
  - monitoring             = false -> null
  - placement_partition_number = 0 -> null
  - primary_network_interface_id = "eni-09f4edc267ab56397" -> null
  - private_dns            = "ip-10-0-1-102.ap-south-1.compute.internal" -> null
}
```

6. Output

Access the public IP address of a web-server in web browser.



Error and Resolution:

Error: Got an error in provider.tf file when I run terraform apply command. The error is that the system where you're running Terraform cannot find the IP address associated with the AWS STS endpoint, preventing Terraform from authenticating with AWS and proceeding with the infrastructure deployment.

```
PS C:\Users\Sakshi\DevOps_Project> terraform apply
data.template_file.user_data: Reading...
data.template_file.user_data: Read complete after 0s [id=767ff8e7c0a77fc6ed8335448ca68e15f6df4a36f075f7b782b6364e3918f01a]
Error: Retrieving AWS account details: validating provider credentials: retrieving caller identity from STS: operation error STS: GetCallerIdentity, https
response error StatusCode: 0, RequestID: , request send failed, Post "https://sts.ap-south-1.amazonaws.com/": dial tcp: lookup sts.ap-south-1.amazonaws.com
: no such host

with provider["registry.terraform.io/hashicorp/aws"],
on provider.tf line 13, in provider "aws":
13: provider "aws" {
```

Resolution: So added “required_version = ">= 1.2"” in provider.tf file

It’s safeguards and ensures that your Terraform configuration will run correctly with the intended Terraform version.

Conclusion:

This project provisions a fully functional AWS VPC with public subnets and deploys an EC2 instance running a static website. The infrastructure is defined and managed as code, enabling version control and repeatable deployments. SSH access is configured while demonstrating secure network practices. Successful execution results in an accessible "**Hello, World! from Terraform**" application via the instance's public IP.