

Terraform documentation for spinning Nginx Server

Kiran Reddy Bokkala:

1. Created the EC2 virtual machine and installed the Terraform in it as shown below.

```
[root@ip-172-31-89-216 Terraformfortest]# history
 1  yum update -y
 2  curl -O https://releases.hashicorp.com/terraform/1.5.0/terraform_1.5.0_linux_amd64.zip
 3  ls
 4  sudo unzip terraform_1.5.0_linux_amd64.zip
 5  ls
 6  mv terraform /usr/local/bin/
 7  ls
 8  terraform --version
```

2. Created a directory VPC and EC2

In the VPC directory created a main.tf file where it has all the code related to creating VPC, subnets, internet gateways, routes, security groups

```
[root@ip-172-31-89-216 ~]# cat vpc/main.tf
provider "aws" {
  region = "us-east-1"
  access_key = "AKIA6ODU3AVWRAICEGGM"
  secret_key = "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"
}

resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true
  tags = {
    Name = "MyVPCforTerraform"
  }
}

resource "aws_subnet" "public_subnet_1" {
  vpc_id = aws_vpc.main.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet1"
  }
}

resource "aws_subnet" "public_subnet_2" {
  vpc_id = aws_vpc.main.id
  cidr_block = "10.0.2.0/24"
  availability_zone = "us-east-1b"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet2"
  }
}

resource "aws_subnet" "public_subnet_3" {
  vpc_id = aws_vpc.main.id
  cidr_block = "10.0.3.0/24"
  availability_zone = "us-east-1c"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet3"
  }
}

resource "aws_subnet" "private_subnet_1" {
  vpc_id = aws_vpc.main.id
  cidr_block = "10.0.4.0/24"
  availability_zone = "us-east-1a"
  tags = {
```

```
resource "aws_subnet" "public_subnet_1" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.1.0/24"
  availability_zone = "us-east-1a"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet1"
  }
}

resource "aws_subnet" "public_subnet_2" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.2.0/24"
  availability_zone = "us-east-1b"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet2"
  }
}

resource "aws_subnet" "public_subnet_3" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.3.0/24"
  availability_zone = "us-east-1c"
  map_public_ip_on_launch = true
  tags = {
    Name = "public_subnet3"
  }
}

resource "aws_subnet" "private_subnet_1" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.4.0/24"
  availability_zone = "us-east-1a"
  tags = {
    Name = "private_subnet1"
  }
}

resource "aws_subnet" "private_subnet_2" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.5.0/24"
  availability_zone = "us-east-1b"
  tags = {
    Name = "private_subnet2"
  }
}

resource "aws_subnet" "private_subnet_3" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.6.0/24"
  availability_zone = "us-east-1c"
  tags = {
```

```

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "Terraform_IG"
  }
}

resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "RouteTable_terraform"
  }
}

resource "aws_route" "internet_access" {
  route_table_id = aws_route_table.public_route_table.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.main.id
}

resource "aws_route_table_association" "public_association_1" {
  subnet_id = aws_subnet.public_subnet_1.id
  route_table_id = aws_route_table.public_route_table.id
}

resource "aws_route_table_association" "public_association_2" {
  subnet_id = aws_subnet.public_subnet_2.id
  route_table_id = aws_route_table.public_route_table.id
}

resource "aws_route_table_association" "public_association_3" {
  subnet_id = aws_subnet.public_subnet_3.id
  route_table_id = aws_route_table.public_route_table.id
}

resource "aws_security_group" "http_sg" {
  vpc_id = aws_vpc.main.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 = "SG Terraform"
  }
}

```

3. Then executed the main.tf code as shown below for provisioning the Network.

```

9  mkdir terraform
10 mkdir vpc
11 cd vpc/
12 vi main.tf
13 terraform init
14 terraform plan
15 terraform validate
16 terraform apply
17 cd ..

```

4. Now created the terraform code for provisioning Nginx EC2 server in ec2 folder.

```
[root@ip-172-31-89-216 ~]# cat ec2/ec2_main.tf
provider "aws" {
  region = "us-east-1"
  access_key = "AKIA6ODU3AVWRAICEGGM"
  secret_key = "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"
}

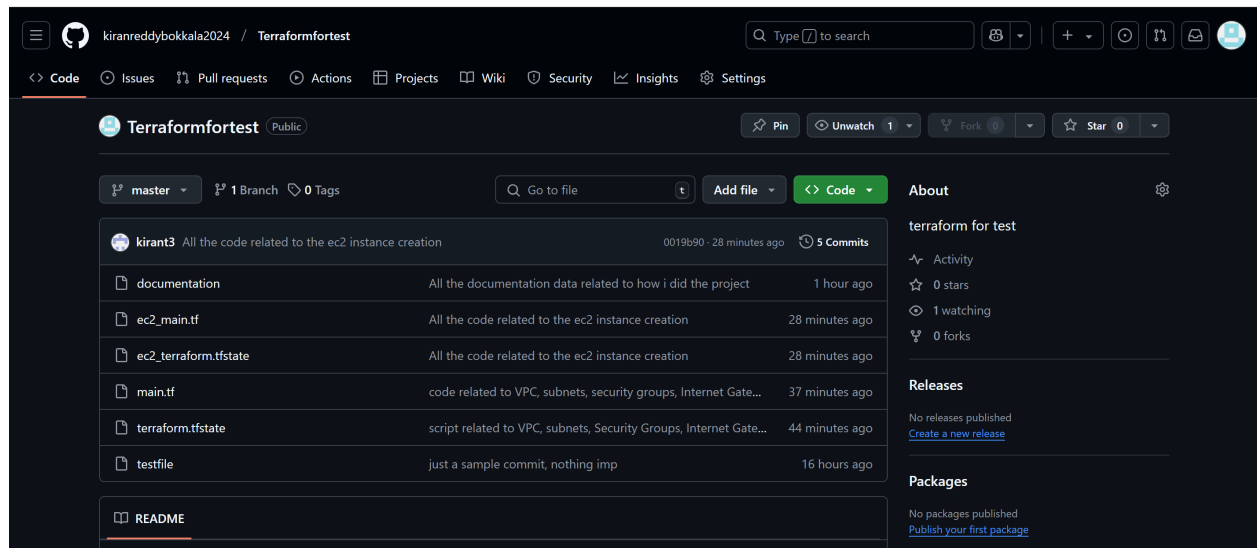
resource "aws_instance" "nginx_instance" {
  ami           = "ami-0ac4dfa1c5c0cce9"
  instance_type = "t2.micro"
  subnet_id     = "subnet-0275bf6852fb6996f"
  vpc_security_group_ids = ["sg-06219e258f3aeec4b"]
  key_name      = "terraform_key"
  tags = {
    Name = "NginxInstance"
  }
}

[root@ip-172-31-89-216 ~]#
```

And executed the code as shown below.

```
20 cd ec2/
21 vi main.tf
22 cat ../vpc/main.tf
23 vi main.tf
24 terraform init
25 terraform plan
26 vi main.tf
27 terraform plan
28 vi main.tf
29 terraform plan
30 terraform validate
31 terraform apply
```

5. Now created a GitHub account with a public repository named Terraformfortest as shown below.



6. Installed Git in the terraform server and cloned the data first.

```
42 yum install git
43 git config --global user.name "kiranReddyBokkala";
44 git config --global user.email "kiran@gmail.com";
45 git config -l
46 git remote add origin https://github.com/kiranreddybokkala2024/Terraformfortest.git
47 cd vpc/
```

7. Now just created a documentation file for steps and pushed it to the GitHub account.

```
93 git clone https://github.com/kiranreddybokkala2024/Terraformfortest.git
94 ls
95 touch documentation
96 ls
97 git add documentation
98 cd Terraformfortest/
99 ls
100 touch documentation
101 git add documentation
102 git commit -m "All the documentation data related to how i did the project"
103 git branch
104 git push
```

8. Now pushed the data related to VPC into GitHub.

```
126 git add main.tf terraform.tfstate
127 git commit -m "script related to VPC, subnets, Security Groups, Internet Gateways, Routes"
128 git status
129 git push
```

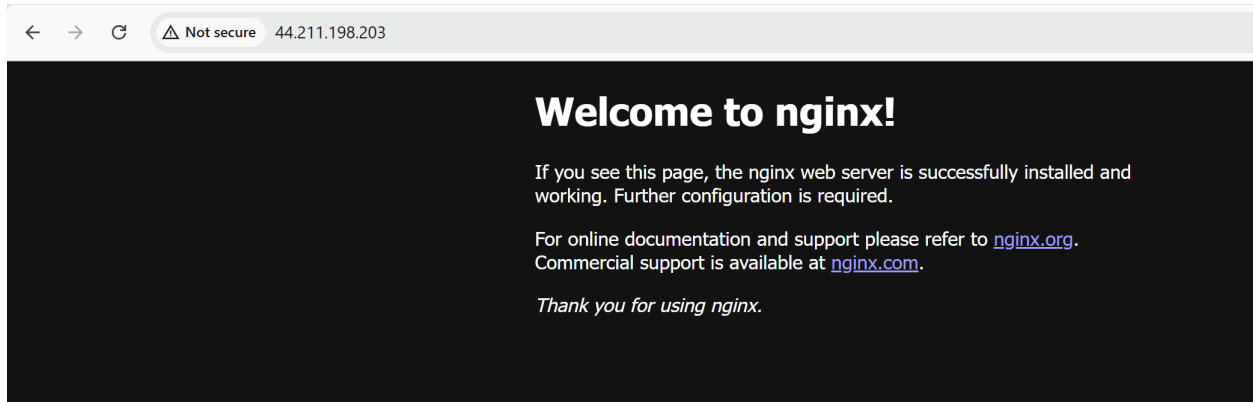
9. Now pushed the data related to EC2 Nginx instance into GitHub account.

```
186 git add ec2_terraform.tfstate ec2_main.tf
187 cat ec2_main.tf
188 git commit -m "All the code related to the ec2 instance creation"
189 git push
```

10. As mentioned, Later after deploying the Nginx server using Terraform ,Logged into the Nginx server and executed below commands to install Nginx server.

```
[root@ip-10-0-1-14 html]# history
 1  yum update -y
 2  yum install nginx
 3  yum install nginx -y
 4  service httpd start
 5  systemctl httpd start
 6  service nginx start
 7  service nginx status
```

Default page:



Custom page

```
[root@ip-10-0-1-14 html]# pwd
/usr/share/nginx/html
[root@ip-10-0-1-14 html]# cat hello.html
This is all about provisioning a Nginx server using terraform
Used terraform to spin VPC, subnets, security groups, Internet gateways, Route tables and the nginx server
[root@ip-10-0-1-14 html]#
```

i-01bafe1e32f6acbf1 (NginxInstance)

PublicIPs: 44.211.198.203 PrivateIPs: 10.0.1.14

AWS Platform screenshots:

1. Created Terraform EC2 server.

The screenshot shows the AWS Management Console interface for an EC2 instance. The left sidebar contains navigation links for Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, and Security Groups. The main content area displays the 'Instance summary for i-03ca68c39fddaf5b6 (Terraform)'. The instance is in a 'Running' state. Key details include: Instance ID: i-03ca68c39fddaf5b6, Public IPv4 address: 52.55.159.138, Private IPv4 address: 172.31.89.216, Hostname type: IP name: ip-172-31-89-216.ec2.internal, Answer private resource DNS name: IPv4 (A), Auto-assigned IP address: 52.55.159.138 [Public IP], IAM Role: -, IMDSv2: Required, Subnet ID: subnet-0b0d8e6bb36f12026, and Instance ARN: arn:aws:ec2:us-east-1:992382485869:instance/i-03ca68c39fddaf5b6. A warning message indicates that the user is not authorized to perform the compute-optimizer:GetEnrollmentStatus action on the resource.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-03ca68c39fddaf5b6	52.55.159.138 open address	172.31.89.216

Instance state	Private IP DNS name (IPv4 only)	Instance type
Running	ip-172-31-89-216.ec2.internal	t2.micro

Subnet ID	Instance ARN
subnet-0b0d8e6bb36f12026	arn:aws:ec2:us-east-1:992382485869:instance/i-03ca68c39fddaf5b6

2. VPC got deployed.

The screenshot shows the AWS Management Console interface for a VPC. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, and Carrier gateways. The main content area displays the 'vpc-080bd8655fdf1f706 / MyVPCforTerraform' details. The VPC is in an 'Available' state. Key details include: VPC ID: vpc-080bd8655fdf1f706, DNS resolution: Enabled, Main network ACL: acl-036ff7e06cb07a824, IPv6 CIDR (Network border group): -, Network Address Usage metrics: Disabled, Block Public Access: Off, DHCP option set: dopt-08f7b68c1b06515f4, IPv4 CIDR: 10.0.0.0/16, Route 53 Resolver DNS Firewall rule groups: -, DNS hostnames: Enabled, Main route table: rtb-082f8cfe5ed6fadc9, IPv6 pool: -, and Owner ID: 992382485869.

VPC ID	State	Block Public Access	DNS hostnames
vpc-080bd8655fdf1f706	Available	Off	Enabled

DNS resolution	Tenancy	Default VPC	Network Address Usage metrics
Enabled	default	No	Disabled

Block Public Access	DHCP option set	IPv4 CIDR	Route 53 Resolver DNS Firewall rule groups
Off	dopt-08f7b68c1b06515f4	10.0.0.0/16	-

DNS hostnames	Main route table	IPv6 pool	Owner ID
Enabled	rtb-082f8cfe5ed6fadc9	-	992382485869

3. Subnets got deployed.

The screenshot shows the AWS Management Console interface for a list of subnets. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet, and Carrier gateways. The main content area displays the 'Subnets (1/12) Info' page. The subnets are listed in a table with columns: Name, Subnet ID, State, VPC, and Block Public... The subnets are all in an 'Available' state and are associated with the VPC vpc-080bd8655fdf1f706.

Name	Subnet ID	State	VPC	Block Public...
public_subnet1	subnet-0275bf6852fb6996f	Available	vpc-080bd8655fdf1f706 MyV...	Off
private_subnet2	subnet-0c9f57b8957feb91d	Available	vpc-080bd8655fdf1f706 MyV...	Off
private_subnet2	subnet-02ce8ebcad565dk92	Available	vpc-080bd8655fdf1f706 MyV...	Off
private_subnet1	subnet-0973f56cb2bb547b6	Available	vpc-080bd8655fdf1f706 MyV...	Off
public_subnet3	subnet-04c8e68afe487eb38	Available	vpc-080bd8655fdf1f706 MyV...	Off
private_subnet3	subnet-0c03393a6a2345d3b	Available	vpc-080bd8655fdf1f706 MyV...	Off

4. Internet gateway deployed and attached to VPC

The screenshot shows the AWS Management Console interface for an Internet Gateway. The breadcrumb navigation indicates the path: VPC > Internet gateways > igw-031bbebbecb4a2c2. The left sidebar shows the VPC dashboard with a filter by VPC dropdown and a list of VPC resources including Virtual private cloud, Your VPCs, Subnets, Route tables, Internet gateways (selected), Egress-only internet gateways, and Security.

igw-031bbebbecb4a2c2 / Terraform_IG

Details [Info](#)

Internet gateway ID igw-031bbebbecb4a2c2	State Attached	VPC ID vpc-080bd8655fdf1f706 MyVPCforTerraform	Owner 992382485869
--	--------------------------	--	------------------------------

Tags [Manage tags](#)

Key	Value
Name	Terraform_IG

5. Route table created and associated with public subnets.

The screenshot shows the AWS Management Console interface for a Route Table. The breadcrumb navigation indicates the path: VPC > Route tables > rtb-020a8d6f619fa7a7b. The left sidebar shows the VPC dashboard with a filter by VPC dropdown and a list of VPC resources including Virtual private cloud, Your VPCs, Subnets, Route tables (selected), Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, and Security.

rtb-020a8d6f619fa7a7b / RouteTable_terraform

Details [Info](#)

Route table ID rtb-020a8d6f619fa7a7b	Main No	Explicit subnet associations 3 subnets	Edge associations -
VPC vpc-080bd8655fdf1f706 MyVPCforTerraform	Owner ID 992382485869		

Routes **Subnet associations** **Edge associations** **Route propagation** **Tags**

Explicit subnet associations (3) [Edit subnet associations](#)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
public_subnet1	subnet-0275bf6852fb6996f	10.0.1.0/24	-
public_subnet2	subnet-02ce8ebcad565dc92	10.0.2.0/24	-
public_subnet3	subnet-04c8e68afe487eb38	10.0.3.0/24	-

6. Security group got created with 22 and 80 ports open.

The screenshot shows the AWS Management Console interface for a security group. The left sidebar contains navigation links for VPC, Security Groups, and PrivateLink and Lattice. The main content area displays the details for the security group **sg-06219e258f3aee4b - terraform-20250127170828469700000002**. The details section includes the security group name, ID, description, VPC ID, owner, inbound rules count, and outbound rules count. Below the details, there are tabs for Inbound rules, Outbound rules, Sharing, VPC associations, and Tags. The Inbound rules tab is active, showing a table with two rules: one for SSH (port 22) and one for HTTP (port 80).

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0bfabfc4a7adf3989	IPv4	SSH	TCP	22
-	sgr-0a6fa2f88a2e51718	IPv4	HTTP	TCP	80

7. Nginx server got deployed with terraform code.

The screenshot shows the AWS Management Console interface for an EC2 instance. The left sidebar contains navigation links for EC2, Instances, and Network & Security. The main content area displays the details for the instance **i-01baf1e32f6acbf1 (NginxInstance)**. The instance is in the **Running** state. The details section includes the instance ID, public IPv4 address, private IPv4 addresses, public IPv4 DNS, private IP DNS name, instance type, VPC ID, subnet ID, instance ARN, IAM Role, and IMDSv2 status. There is also an AWS Compute Optimizer finding displayed.

Field	Value
Instance ID	i-01baf1e32f6acbf1
Public IPv4 address	44.211.198.203
Private IPv4 addresses	10.0.1.14
Public IPv4 DNS	ec2-44-211-198-203.compute-1.amazonaws.com
Private IP DNS name (IPv4 only)	ip-10-0-1-14.ec2.internal
Instance type	t2.micro
VPC ID	vpc-080bd8655fdf1f706
Subnet ID	subnet-0275bf6852fb6996f
Instance ARN	arn:aws:ec2:us-east-1:992382485869:instance/i-01baf1e32f6acbf1
IAM Role	-
IMDSv2	Required