

School of Computer Science

University of Petroleum and Energy Studies



System Provisioning & Configuration Management

Lab File (6th Sem)

Submitted By:-

Akshat Pandey

500101788

R2142220306

DevOps B1

Submitted To:-

Dr Hitesh Kumar Sharma

EXPERIMENT 11

Creating a VPC in Terraform

Objective:

Learn how to use Terraform to create a basic Virtual Private Cloud (VPC) in AWS.

Prerequisites:

- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-vpc
cd terraform-vpc
```

```
C:\Users\aksha\Documents\SPCM_LAB>mkdir terraform-vpc
C:\Users\aksha\Documents\SPCM_LAB>cd terraform-vpc
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>|
```

- Create Terraform Configuration Files:
- Create a file named main.tf

vpc.tf

```
resource "aws_vpc" "gfg-vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "gfg-subnet" {
  vpc_id    = aws_vpc.gfg-vpc.id
  cidr_block = "10.0.1.0/24"
```

```
tags = {
  Name = "gfg-subnet"
}

resource "aws_internet_gateway" "gfg-gw" {
  vpc_id = aws_vpc.gfg-vpc.id

  tags = {
    Name = "gfg-IG"
  }
}

resource "aws_route_table" "gfg-rt" {
  vpc_id = aws_vpc.gfg-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gfg-gw.id
  }

  tags = {
    Name = "GFG-Route-Table"
  }
}

resource "aws_route_table_association" "gfg-rta" {
  subnet_id    = aws_subnet.gfg-subnet.id
  route_table_id = aws_route_table.gfg-rt.id
}

resource "aws_security_group" "gfg-sg" {
  name      = "my-gfg-sg"
  vpc_id    = aws_vpc.gfg-vpc.id

  ingress {
```

```

description    = "TLS from VPC"
from_port      = 20
to_port        = 20
protocol       = "tcp"
cidr_blocks    = ["0.0.0.0/0"]
ipv6_cidr_blocks = [ ":::/0" ]
}

egress {
  from_port      = 0
  to_port        = 0
  protocol       = "-1"
  cidr_blocks    = ["0.0.0.0/0"]
  ipv6_cidr_blocks = [ ":::/0" ]
}

tags = {
  Name = "my-gfg-sg"
}
}

```

In this configuration, we define an AWS provider, a VPC with a specified CIDR block, and two subnets within the VPC.

```

vpctf  x  provider.tf
vpctf > resource "aws_internet_gateway" "gfg-gw"
1 resource "aws_vpc" "gfg-vpc" {
2   cidr_block = "10.0.0.0/16"
3   tags = {
4     Name = "gfg-vpc"
5   }
6 }
7
8 resource "aws_subnet" "gfg-subnet" {
9   vpc_id = aws_vpc.gfg-vpc.id
10  cidr_block = "10.0.1.0/24"
11
12  tags = {
13    Name = "gfg-subnet"
14  }
15 }
16
17 resource "aws_internet_gateway" "gfg-gw" {
18   vpc_id = aws_vpc.gfg-vpc.id
19
20   tags = {
21     Name = "gfg-IG"
22   }
23 }
24
25 resource "aws_route_table" "gfg-rt" {
26   vpc_id = aws_vpc.gfg-vpc.id
27
28   route {
29     cidr_block = "0.0.0.0/0"
30     gateway_id = aws_internet_gateway.gfg-gw.id
31   }
32
33   tags = {
34     Name = "GFG-Route-Table"
35   }
36 }
37
38 resource "aws_route_table_association" "gfg-rta" {
39   subnet_id = aws_subnet.gfg-subnet.id
40 }
41
42
43 resource "aws_security_group" "gfg-sg" {
44   name = "my-gfg-sg"
45   vpc_id = aws_vpc.gfg-vpc.id
46
47   ingress {
48     description = "TLS from VPC"
49     from_port = 20
50     to_port = 20
51     protocol = "tcp"
52     cidr_blocks = ["0.0.0.0/0"]
53     ipv6_cidr_blocks = [ ":::/0" ]
54   }
55
56   egress {
57     from_port = 0
58     to_port = 0
59     protocol = "-1"
60     cidr_blocks = ["0.0.0.0/0"]
61     ipv6_cidr_blocks = [ ":::/0" ]
62   }
63
64   tags = {
65     Name = "my-gfg-sg"
66   }
67 }

```

2. Initialize and Apply:

- Run the following Terraform commands to initialize and apply the configuration:

terraform init

terraform apply

```
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.96.0...
- Installed hashicorp/aws v5.96.0 (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.

```
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>terraform apply
```

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_internet_gateway.gfg-gw will be created
+ resource "aws_internet_gateway" "gfg-gw" {
+   arn           = (known after apply)
+   id            = (known after apply)
+   owner_id      = (known after apply)
+   tags          = {
+     "Name" = "gfg-IG"
+   }
+   tags_all      = {
+     "Name" = "gfg-IG"
+   }
+   vpc_id        = (known after apply)
+ }

# aws_route_table.gfg-rt will be created
+ resource "aws_route_table" "gfg-rt" {
+   arn           = (known after apply)
+   id            = (known after apply)
+   owner_id      = (known after apply)
+   propagating_vgws = (known after apply)
+   route         = [
+     {
+       cidr_block      = "0.0.0.0/0"
+       gateway_id      = (known after apply)
+       # (11 unchanged attributes hidden)
+     },
+   ]
+   tags           = {
+     "Name" = "GFG-Route-Table"
+   }
+   tags_all       = {
```

```
+   enable_dns_hostnames      = (known after apply)
+   enable_dns_support        = true
+   enable_network_address_usage_metrics = (known after apply)
+   id                        = (known after apply)
+   instance_tenancy          = "default"
+   ipv6_association_id       = (known after apply)
+   ipv6_cidr_block           = (known after apply)
+   ipv6_cidr_block_network_border_group = (known after apply)
+   main_route_table_id       = (known after apply)
+   owner_id                  = (known after apply)
+   tags                      = {
+     "Name" = "gfg-vpc"
+   }
+   tags_all                  = {
+     "Name" = "gfg-vpc"
+   }
+ }
```

Plan: 6 to add, 0 to change, 0 to destroy.

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.gfg-vpc: Creating...
aws_vpc.gfg-vpc: Creation complete after 2s [id=vpc-00fcd3a6f8745301d]
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete after 0s [id=igw-0f24598595076ab70]
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 0s [id=subnet-04b07c71ee05e7987]
aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0a03a855b1f33f06e]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 0s [id=rtbassoc-0c0148fef7a957db5]
aws_security_group.gfg-sg: Creation complete after 2s [id=sg-09e2eb305d8aa2a9a]
```

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

- Terraform will prompt you to confirm the creation of the VPC and subnets. Type yes and press Enter.

3. Verify Resources in AWS Console:

- Log in to the AWS Management Console and navigate to the VPC service.
- Verify that the VPC and subnets with the specified names and settings have been created.

Your VPCs (2)
[Info](#)

Q

Find VPCs by attribute or tag

Last updated

4 minutes ago

↺

1

↻

⚙️

<input type="checkbox"/>	Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	-	vpc-06eb75fb5a123f903	✔️ Available	⊖ Off	172.31.0.0/16	-
<input type="checkbox"/>	gfg-vpc	vpc-00fcd3a6f8745301d	✔️ Available	⊖ Off	10.0.0.0/16	-

Subnets (4)
[Info](#)

Q

Find subnets by attribute or tag

Last updated

less than a minute ago

↺

1

↻

⚙️

<input type="checkbox"/>	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
<input type="checkbox"/>	-	subnet-012932dab17c51ca2	✔️ Available	vpc-06eb75fb5a123f903	⊖ Off	172.31.0.0/20
<input type="checkbox"/>	gfg-subnet	subnet-04b07c71ee05e7987	✔️ Available	vpc-00fcd3a6f8745301d gfg-v...	⊖ Off	10.0.1.0/24
<input type="checkbox"/>	-	subnet-0b69bc54540894400	✔️ Available	vpc-06eb75fb5a123f903	⊖ Off	172.31.16.0/20
<input type="checkbox"/>	-	subnet-0571e32fd1505e87c	✔️ Available	vpc-06eb75fb5a123f903	⊖ Off	172.31.32.0/20

4. Update VPC Configuration:

- If you want to modify the VPC configuration, update the main.tf file with the desired changes.
- Rerun the terraform apply command to apply the changes:

terraform apply

```
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>terraform apply
```

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_internet_gateway.gfg-gw will be created
+ resource "aws_internet_gateway" "gfg-gw" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + tags     = {
    + "Name" = "gfg-IG"
  }
  + tags_all = {
    + "Name" = "gfg-IG"
  }
  + vpc_id   = (known after apply)
}

# aws_route_table.gfg-rt will be created
+ resource "aws_route_table" "gfg-rt" {
  + arn              = (known after apply)
  + id               = (known after apply)
  + owner_id         = (known after apply)
  + propagating_vgws = (known after apply)
  + route            = [
    + {
      + cidr_block      = "0.0.0.0/0"
      + gateway_id      = (known after apply)
      # (11 unchanged attributes hidden)
    },
  ]
  + tags             = {
    + "Name" = "GFG-Route-Table"
  }
  + tags_all         = {

    + enable_dns_hostnames      = (known after apply)
    + enable_dns_support       = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                       = (known after apply)
    + instance_tenancy         = "default"
    + ipv6_association_id      = (known after apply)
    + ipv6_cidr_block          = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id      = (known after apply)
    + owner_id                 = (known after apply)
    + tags                     = {
      + "Name" = "gfg-vpc"
    }
    + tags_all                 = {
      + "Name" = "gfg-vpc"
    }
  }
}
```

Plan: 6 to add, 0 to change, 0 to destroy.

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.gfg-vpc: Creating...

aws_vpc.gfg-vpc: Creation complete after 2s [id=vpc-00fcd3a6f8745301d]

aws_internet_gateway.gfg-gw: Creating...

aws_subnet.gfg-subnet: Creating...

aws_security_group.gfg-sg: Creating...

aws_internet_gateway.gfg-gw: Creation complete after 0s [id=igw-0f24590595076ab70]

aws_route_table.gfg-rt: Creating...

aws_subnet.gfg-subnet: Creation complete after 0s [id=subnet-04b07c71ee05e7987]

aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0a03a855b1f33f06e]

aws_route_table_association.gfg-rta: Creating...

aws_route_table_association.gfg-rta: Creation complete after 0s [id=rtbassoc-0c0148fef7a957db5]

aws_security_group.gfg-sg: Creation complete after 2s [id=sg-09e2eb305d8aa2a9a]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

5. Clean Up:

After testing, you can clean up the VPC and subnets:

terraform destroy

```
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>terraform destroy
aws_vpc.gfg-vpc: Refreshing state... [id=vpc-00fcd3a6f8745301d]
aws_internet_gateway.gfg-gw: Refreshing state... [id=igw-0f24590595076ab70]
aws_subnet.gfg-subnet: Refreshing state... [id=subnet-04b07c71ee05e7987]
aws_security_group.gfg-sg: Refreshing state... [id=sg-09e2eb305d8aa2a9a]
aws_route_table.gfg-rt: Refreshing state... [id=rtb-0a03a855b1f33f06e]
aws_route_table_association.gfg-rta: Refreshing state... [id=rtbassoc-0c0148fef7a957db5]

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_internet_gateway.gfg-gw will be destroyed
- resource "aws_internet_gateway" "gfg-gw" {
  - arn      = "arn:aws:ec2:ap-south-1:158878148841:internet-gateway/igw-0f24590595076ab70" -> null
  - id      = "igw-0f24590595076ab70" -> null
  - owner_id = "158878148841" -> null
  - tags    = {
    - "Name" = "gfg-IG"
  } -> null
  - tags_all = {
    - "Name" = "gfg-IG"
  } -> null
  - vpc_id   = "vpc-00fcd3a6f8745301d" -> null
}

# aws_route_table.gfg-rt will be destroyed
- resource "aws_route_table" "gfg-rt" {
  - arn      = "arn:aws:ec2:ap-south-1:158878148841:route-table/rtb-0a03a855b1f33f06e" -> null
  - id      = "rtb-0a03a855b1f33f06e" -> null
  - owner_id = "158878148841" -> null
  - propagating_vgws = [] -> null
  - route      = [
    - {
      - cidr_block      = "0.0.0.0/0"
      - gateway_id      = "igw-0f24590595076ab70"
    } # (11 unchanged attributes hidden)
  ],
  - default_route_table_id      = "rtb-0f7faa5ad07b17de8" -> null
  - default_security_group_id   = "sg-07d25005ee3cef497" -> null
  - dhcp_options_id            = "dopt-0f5aa075b3f4a8b77" -> null
  - enable_dns_hostnames       = false -> null
  - enable_dns_support          = true -> null
  - enable_network_address_usage_metrics = false -> null
  - id                          = "vpc-00fcd3a6f8745301d" -> null
  - instance_tenancy           = "default" -> null
  - ipv6_netmask_length         = 0 -> null
  - main_route_table_id        = "rtb-0f7faa5ad07b17de8" -> null
  - owner_id                    = "158878148841" -> null
  - tags                        = {
    - "Name" = "gfg-vpc"
  } -> null
  - tags_all                    = {
    - "Name" = "gfg-vpc"
  } -> null
  # (4 unchanged attributes hidden)
}

Plan: 0 to add, 0 to change, 6 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_route_table_association.gfg-rta: Destroying... [id=rtbassoc-0c0148fef7a957db5]
aws_security_group.gfg-sg: Destroying... [id=sg-09e2eb305d8aa2a9a]
aws_route_table_association.gfg-rta: Destruction complete after 0s
aws_subnet.gfg-subnet: Destroying... [id=subnet-04b07c71ee05e7987]
aws_route_table.gfg-rt: Destroying... [id=rtb-0a03a855b1f33f06e]
aws_subnet.gfg-subnet: Destruction complete after 1s
aws_security_group.gfg-sg: Destruction complete after 1s
aws_route_table.gfg-rt: Destruction complete after 1s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-0f24590595076ab70]
aws_internet_gateway.gfg-gw: Destruction complete after 0s
aws_vpc.gfg-vpc: Destroying... [id=vpc-00fcd3a6f8745301d]
aws_vpc.gfg-vpc: Destruction complete after 1s

Destroy complete! Resources: 6 destroyed.

C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>
```

6. Conclusion:

This lab exercise demonstrates how to create a basic Virtual Private Cloud (VPC) with subnets in AWS using Terraform. The example includes a simple VPC configuration with two subnets. Experiment with different CIDR blocks, settings, and additional AWS resources to customize your VPC.