VPC configuration using terraform:

Configuring a Virtual Private Cloud (VPC) using Terraform allows us to define and manage our AWS networking resources in a structured and repeatable way. Here's a step-by-step guide to set up a basic VPC configuration using Terraform:

Terraform config file can be done as

```
mkdir vpc
touch main.tf  // this is the main config file to execute
Vim main.tf
```

```
# Define the provider
provider "aws" {
  region = "us-east-2"  # Change to your desired region
}

# Define the VPC
resource "aws_vpc" "main" {
  cidr_block           = "10.0.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = {
    Name = "amrit-vpc"
  }
}

# Define an Internet Gateway
resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "amrit-igw"
  }
}

# Define a Public Subnet
resource "aws_subnet" "public" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-2a"
  tags = {
    Name = "amrit-public-subnet"
  }
}

# Define a Route Table
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }
  tags = {
    Name = "amrit-public-rt"
  }
}

# Associate the Route Table with the Public Subnet
resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.public.id
  route_table_id = aws_route_table.public.id
}

# Define a Security Group
resource "aws_security_group" "web_sg" {
```

We can initialize as

```
amrit@amrit-Inspiron-5567:~/vpc$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.55.0

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.
amrit@amrit-Inspiron-5567:~/vpc$ terraform plan
aws_route_table_association.public_subnet_association: Refreshing state... [id=rtbassoc-0ffb4d3cda6b8e4c7
aws_security_group.allow_all: Refreshing state... [id=sg-038302c2c7e2d1c5b]
aws_vpc.main: Refreshing state... [id=vpc-0352ec7e24b90419f]
aws_subnet.private: Refreshing state... [id=subnet-0b521be794f13b27e]
aws_instance.example: Refreshing state... [id=i-0d4a6c164fff369c2]
aws_internet_gateway.gw: Refreshing state... [id=igw-0e3da40f260291fcf]
```

We can run the terraform plan as

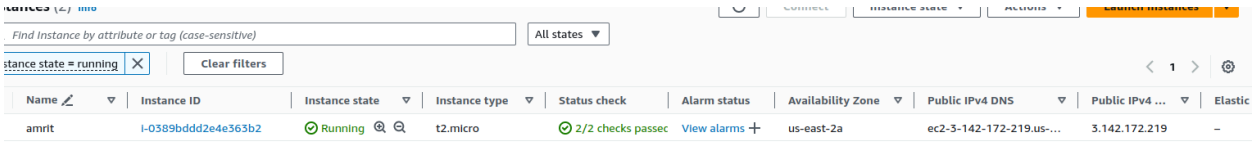
```
amrit@amrit-Inspiron-5567:~/vpc$ terraform plan
aws_route_table_association.public_subnet_association: Refreshing state... [id=rtbassoc-0ffb4d3cda6b8e4c7]
aws_security_group.allow_all: Refreshing state... [id=sg-038302c2c7e2d1c5b]
aws_vpc.main: Refreshing state... [id=vpc-0352ec7e24b90419f]
aws_subnet.private: Refreshing state... [id=subnet-0b521be794f13b27e]
aws_instance.example: Refreshing state... [id=i-0d4a6c164fff369c2]
aws_internet_gateway.gw: Refreshing state... [id=igw-0e3da40f260291fcf]
aws_subnet.public: Refreshing state... [id=subnet-04523af7e8859d306]
aws_route_table.public: Refreshing state... [id=rtb-053b860ee3498031f]

Terraform used the selected providers to generate the following execution plan. Resource actions are indic
  + create
  ~ update in-place
  - destroy

Terraform will perform the following actions:

  # aws_instance.example will be destroyed
  # (because aws_instance.example is not in configuration)
  - resource "aws_instance" "example" {
      - ami                                  = "ami-0c55b159cbfafe1f0" -> null
      - arn                                  = "arn:aws:ec2:us-east-2:730335211981:instance/i-0d4a6c164fff
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-2a" -> 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-0d4a6c164fff369c2" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - key_name                             = "punkeypair" -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0c6fab7bb7215f157" -> null
```

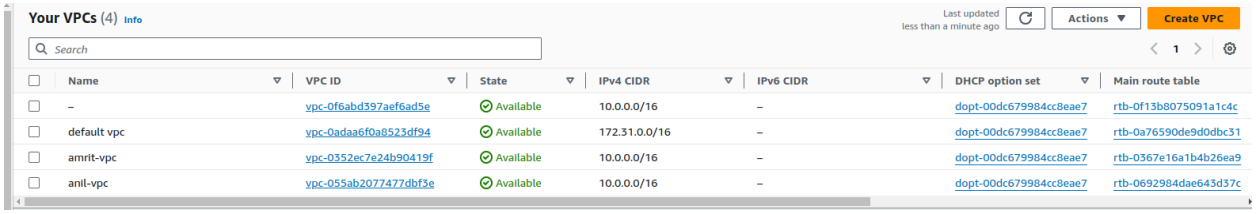
Finally we can apply the .tf file as

```
      ~ tags_all                              = {
          ~ "Name" = "amritvpc" -> "amrit-vpc"
        }
        # (18 unchanged attributes hidden)
    }

Plan: 3 to add, 4 to change, 4 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_route_table_association.public_subnet_association: Destroying... [id=rtbassoc-0ffb4d3cda6b8e4c7]
aws_security_group.allow_all: Destroying... [id=sg-038302c2c7e2d1c5b]
aws_subnet.private: Destroying... [id=subnet-0b521be794f13b27e]
aws_instance.example: Destroying... [id=i-0d4a6c164fff369c2]
aws_route_table_association.public_subnet_association: Destruction complete after 2s
aws_subnet.private: Destruction complete after 3s
aws_security_group.allow_all: Destruction complete after 4s
aws_instance.example: Still destroying... [id=i-0d4a6c164fff369c2, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0d4a6c164fff369c2, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0d4a6c164fff369c2, 30s elapsed]
aws_instance.example: Destruction complete after 33s
aws_vpc.main: Modifying... [id=vpc-0352ec7e24b90419f]
aws_vpc.main: Modifications complete after 3s [id=vpc-0352ec7e24b90419f]
aws_internet_gateway.gw: Modifying... [id=igw-0e3da40f260291fcf]
aws_security_group.web_sg: Creating...
aws_subnet.public: Modifying... [id=subnet-04523af7e8859d306]
aws_internet_gateway.gw: Modifications complete after 1s [id=igw-0e3da40f260291fcf]
aws_route_table.public: Modifying... [id=rtb-053b860ee3498031f]
aws_subnet.public: Modifications complete after 2s [id=subnet-04523af7e8859d306]
aws_route_table.public: Modifications complete after 1s [id=rtb-053b860ee3498031f]
aws_route_table_association.public: Creating...
aws_route_table_association.public: Creation complete after 1s [id=rtbassoc-070039ec7d2ddbe58]
aws_security_group.web_sg: Creation complete after 5s [id=sg-0e84b7503e3f4b9cd]
aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Still creating... [30s elapsed]
aws_instance.web: Creation complete after 36s [id=i-0389bddd2e4e363b2]

Apply complete! Resources: 3 added, 4 changed, 4 destroyed.
amrit@amrit-Inspiron-5567:~/vpc$
```

We can destroy the vpc using command in the cli as well so that minimum resources are utilized.

The created vpc and ec2 can be shown as





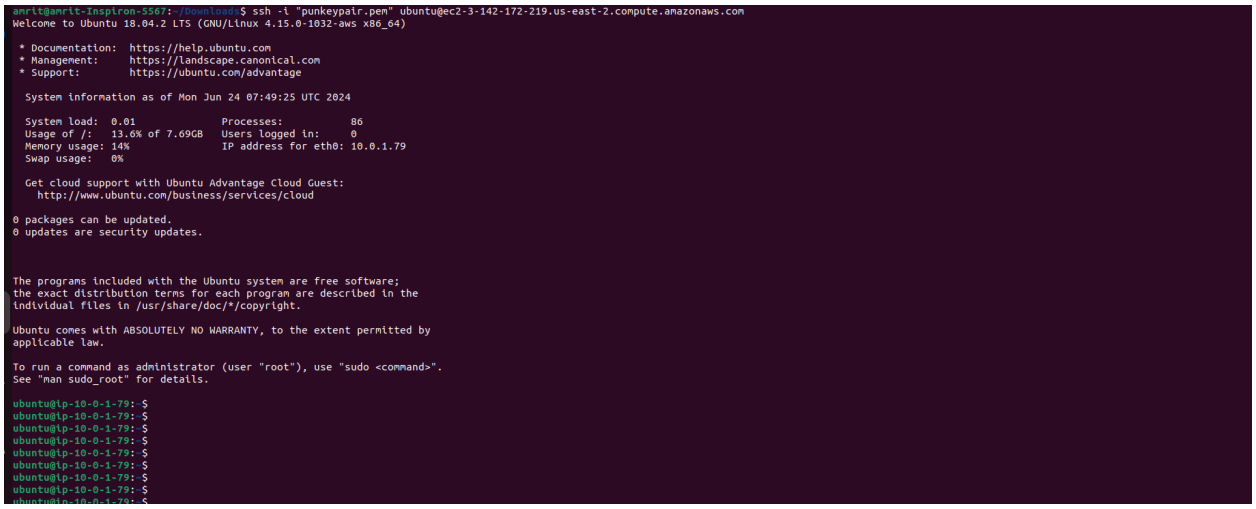
Your VPCs (4) Info

Last updated less than a minute ago | Actions | Create VPC

Search

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
–	vpc-0f6abd397aef6ad5e	Available	10.0.0.0/16	–	dopt-00dc679984cc8eae7	rtb-0f13b8075091a1c4c
default vpc	vpc-0adaa6f0a8523df94	Available	172.31.0.0/16	–	dopt-00dc679984cc8eae7	rtb-0a76590de9d0dbc31
amrit-vpc	vpc-0352ec7e24b90419f	Available	10.0.0.0/16	–	dopt-00dc679984cc8eae7	rtb-0367e16a1b4b26ea9
anil-vpc	vpc-055ab2077477dbf3e	Available	10.0.0.0/16	–	dopt-00dc679984cc8eae7	rtb-0692984dae643d37c

We can access the ec2 instance through ssh like this



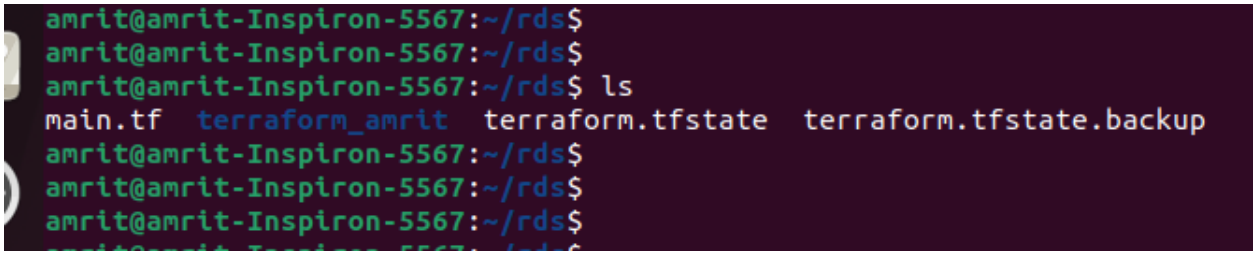
```
amrit@amrit-Inspiron-5567:~/Downloads$ ssh -i "punkeypair.pem" ubuntu@ec2-3-142-172-219.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1032-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Jun 24 07:49:25 UTC 2024

  System load:  0.01              Processes:           86
  Usage of /:   13.6% of 7.69GB   Users logged in:     0
  Memory usage: 14%               IP address for eth0: 10.0.1.79
  Swap usage:   0%

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
ubuntu@ip-10-0-1-79:~$
```

RDS configuration using terraform:

To configure an RDS (Relational Database Service) instance using Terraform, we need to define several resources and configurations in our Terraform script. Below is an example configuration that sets up a basic MySQL RDS instance in AWS. This example assumes we have already set up our AWS provider configuration in Terraform.

Here's how we can configure an RDS instance using Terraform:

Here is main tf file for rds



```
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$ ls
main.tf  terraform_amrit  terraform.tfstate  terraform.tfstate.backup
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
```

The code can be configured as below

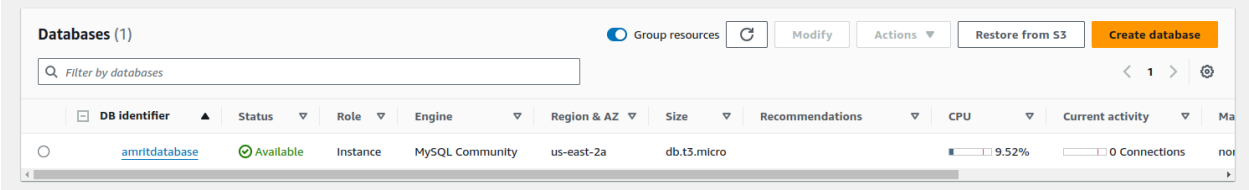
```
# configured aws provider with proper credentials
provider "aws" {
  region = "us-east-2"

}

# create default vpc if one does not exist
resource "aws_default_vpc" "default_vpc" {
  tags = {
    Name = "default vpc"
  }
}

# use data source to get all availability zones in the region
data "aws_availability_zones" "available_zones" {}

# create a default subnet in the first available zone
resource "aws_default_subnet" "subnet_az" {
  availability_zone = data.aws_availability_zones.available_zones.names[0]  # First
}

# create a default subnet in the second available zone
resource "aws_default_subnet" "subnet_az2" {
availability_zone = data.aws_availability_zones.available_zones.names[1]  # Second A
}

# create security group for the web server
resource "aws_security_group" "amrit" {
  name        = "amrit"
  description = "enable http access on port 80"
  vpc_id      = aws_default_vpc.default_vpc.id

  ingress {
    description = "http access"
    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 = "webserver security group"
  }
}
```

The terraform file can be executed as follows

```
resource "aws_db_subnet_group" "database4" {
  name        = "database4"
  subnet_ids  = [aws_default_subnet.subnet_az.id, aws_default_subnet.subnet_az2.id]

  description = "Subnet group for RDS"

  tags = {
    Name = "database subnet group"
  }
}


# create the rds instance
resource "aws_db_instance" "db_instance" {
  engine                = "mysql"
  engine_version        = "5.7"
  multi_az              = false
  identifier            = "amritdatabase"
  username              = "amrit"
  password              = "amrit123"
  instance_class        = "db.t3.micro"
  allocated_storage     = 20
  db_subnet_group_name  = aws_db_subnet_group.database4.name
  vpc_security_group_ids = [aws_security_group.punamrit.id]
  availability_zone     = data.aws_availability_zones.available_zones.names[0]
  publicly_accessible = true
  db_name               = "amritdb"
  skip_final_snapshot   = true
}

amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$
amrit@amrit-Inspiron-5567:~/rds$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.55.0

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.
amrit@amrit-Inspiron-5567:~/rds$
```

```
      ~ ingress                = [
          - {
              - cidr_blocks      = [
                  - "0.0.0.0/0",
                ]
              - from_port        = 3306
              - ipv6_cidr_blocks = []
              - prefix_list_ids  = []
              - protocol         = "tcp"
              - security_groups  = []
              - self             = false
              - to_port          = 3306
                # (1 unchanged attribute hidden)
            },
            # (1 unchanged element hidden)
        ]
        name                     = "amrit"
        tags                     = {
            "Name" = "webserver security group"
        }
        # (8 unchanged attributes hidden)
    }

  # aws_security_group.punamrit will be updated in-place
  ~ resource "aws_security_group" "punamrit" {
        id                       = "sg-036928f16a4ab4204"
      ~ ingress                = [
          - {
              - cidr_blocks      = [
                  - "0.0.0.0/0",
                ]
              - from_port        = 3306
              - ipv6_cidr_blocks = []
              - prefix_list_ids  = []
              - protocol         = "tcp"
              - security_groups  = []
              - self             = false
              - to_port          = 3306
                # (1 unchanged attribute hidden)
            },
            # (1 unchanged element hidden)
        ]
        name                     = "punamrit"
        tags                     = {
            "Name" = "database security group"
        }
        # (8 unchanged attributes hidden)
    }

Plan: 1 to add, 2 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee
amrit@amrit-Inspiron-5567:~/rds$
```

The created database is shown in aws mgmt console and accessed via ssh.



Launching ec2 instance using terraform:

Configuring an EC2 instance using Terraform involves defining various resources and configurations to provision and manage the instance on AWS. Here's a step-by-step guide to set up a basic EC2 instance using Terraform:

Prerequisites:

- Ensure we have Terraform installed on our local machine.
- Have AWS credentials configured either through environment variables, AWS CLI configuration, or directly in our Terraform provider configuration.

The sample code format is

```
amrit@amrit-Inspiron-5567: ~/ec2

  name        = "rdstest"
  description = "enable mysql/aurora access on port 3306"
  vpc_id      = aws_default_vpc.default_vpc.id

  ingress {
    description      = "mysql/aurora access"
    from_port        = 3306
    to_port          = 3306
    protocol         = "tcp"
    security_groups  = [aws_security_group.samundra.id]
  }

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

  tags = {
    Name = "database security group"
  }
}

# create the subnet group for the rds instance
resource "aws_db_subnet_group" "database1" {
  name        = "database1"
  subnet_ids  = [aws_default_subnet.subnet_az.id, aws_default_subnet.subnet_az2.id]  # Corrected subnet resource names

  description = "Subnet group for RDS"

  tags = {
    Name = "database subnet group"
  }
}


# create the rds instance
```

The three terraform related commands are

Terraform init

Terraform plan

Terraform apply

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

root@samundra:~/terraformdemo/rds-terraform# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.54.1

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.
root@samundra:~/terraformdemo/rds-terraform#
```

```
root@samundra:~/terraformdemo/rds-terraform# terraform plan

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_db_instance.db_instance will be created
  + resource "aws_db_instance" "db_instance" {
      + address                               = (known after apply)
      + allocated_storage                     = 20
      + apply_immediately                     = false
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = "us-east-2a"
      + backup_retention_period               = (known after apply)
      + backup_target                         = (known after apply)
```

The created ec2 instance is shown as