Three tier AWS VPC create using terraform

In this document we will show you how to build a three tier AWS VPC network architecture using Terraform. This network architecture has three subnet tiers split across two availability zones. The web subnets also have a VPC routing table that will provide it access to the internet. The application and database tiers will not have such access; their routing tables will only allow internal network communication.

Prerequisites

You should have the Terraform CLI setup on your computer. You can follow the official Terraform docs to learn how to do this.

Create Directories & Files

Next, create a baseline directory and file structure as follows.

```
3-tier-app/
├── main.tf
├── outputs.tf
└── variables.tf
└── env/
    └──dev.tfvars
```

Modify the Main Conf File

Open the `main.tf` file and add the following configuration. This will define the AWS provider, set the region to work in, and specify what credentials to use.
You must write in the name of your AWS profile to use e.g. “default”

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.26.0"
    }
  }
}
```

```
provider "aws" {
  region = "ap-southeast-1"
  profile = "name-of-your-profile"
}
```

Run Terraform Init

Now let's initialize the Terraform project. At your CLI, enter in this command.

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.26.0"...
- Installing hashicorp/aws v3.26.0...
- Installed hashicorp/aws v3.26.0 (signed by HashiCorp)

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.
```

Create a Workspace

Terraform has a concept of workspaces. Workspaces are a way for you to create different working environments, such as dev, test, staging, prod, etc. You can also use them to setup resources in different regions. They are a useful way to organize your projects.
In this tutorial, we will create a workspace for a `dev` environment. At your CLI, type in the following:

```
$ terraform workspace list
* default


$ terraform workspace new dev
Created and switched to workspace "dev"!


You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing
state
for this configuration.


$ terraform workspace list
  default
* dev
```

Define Variables

We can use variables to make it easier to work with Terraform in multiple environments (e.g. dev, test, prod). To do so, we need to define the variables and their types. Then, we can define what the values for our dev environment.

Append the following code to the `variables.tf` file:

```
variable "aws_region" {
    description = "AWS Region"
}


variable "vpc_cidr_block" {
    description = "Main VPC CIDR Block"
```

```
}

variable "az_public_subnet" {
  type = map(string)
}

variable "az_private_subnet" {
  type = map(string)
}

variable "az_database_subnet" {
  type = map(string)
}

variable "availability_zones" {
  type = list(string)
}
```

Code Review

- The above code defines the variables and their types. It does not define their values; we'll do that next
- You can learn more about this topic on Terraform's Input Variables docs

Next, we will define the variable values for the env environment. Append the following code to the `env/dev.tfvars` file:

```
aws_region = "ap-southeast-1"

vpc_cidr_block = "10.0.0.0/16"

az_public_subnet = {
  "ap-southeast-1a" : "10.0.0.0/24",
  "ap-southeast-1b" : "10.0.1.0/24"
}
```

```
az_private_subnet = {
  "ap-southeast-1a" : "10.0.101.0/24",
  "ap-southeast-1b" : "10.0.102.0/24"
}


az_database_subnet = {
  "ap-southeast-1a" : "10.0.201.0/24",
  "ap-southeast-1b" : "10.0.202.0/24"
}


availability_zones = [
  "ap-southeast-1a",
  "ap-southeast-1b"
]
```

Code Review

- If you compare the variables.tf and dev.tfvars files, you'll see that the variable names match up
- In the dev.tfvars, we define the location (Singapore) and shape (subnet structure) of our VPC network

Create the VPC Resource

Next, we'll switch to the working on the main.tf file. Here, we will start by creating a VPC resource. You can see a full list of Terraform's AWS Resources on its registry.

Append the following code to the `main.tf` file:

```
# VPC Resource
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr_block


  tags = {
    Name = "cloudy-lab-vpc"
  }
```

```
}
```

Create the Internet Gateway

By default, all traffic within a VPC is bound to the VPC itself. The only way to enable internet access is by creating an Internet Gateway resource. We can then create network routes for subnets that route internet bound traffic through the gateway.

Append the following code to the `main.tf` file:

```
# Internet Gateway

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id


  tags = {
    Name = "cloudy-lab-internet-gateway"
  }
}
```

Create the Public Subnet Resources

The web tier will contain all of the resources needed for the web server. This includes the load balancer and EC2 instances that are running the web server application. This is a public subnet because it needs to provide connectivity to the load balancer.

Append the following code to the `main.tf` file:

```
# Public Subnet

resource "aws_subnet" "public_subnet" {
  for_each = var.az_public_subnet


  vpc_id = aws_vpc.main.id


  availability_zone = each.key
  cidr_block        = each.value


  tags = {
    Name = "cloudy-lab-public-subnet-${each.key}"
  }
}
```

Code Review

- This code follows the Terraform VPC Subnet specification
- New to the code above is a value called for_each. This value takes in the public subnet variables, of which there are two values. Using for_each will create a resource for as many items are in the value. In this case, there are two public subnets, so two resources will be created. Learn more about for_each.

Private Subnet

This code is similar to the web tier code. Instead of iterating through the az_public_subnet variable, it will iterate through the private subnet values.
Append the following code to the `main.tf` file:

```
# Private Subnet

resource "aws_subnet" "private_subnet" {

  for_each = var.az_private_subnet


  vpc_id = aws_vpc.main.id


  availability_zone = each.key

  cidr_block        = each.value


  tags = {

    Name = "cloudy-lab-private-subnet-${each.key}"

  }

}
```

Database Subnet

The last major section of this component is the database tier config. These two subnets are both private, and will provide connectivity for the RDS database instance.
Append the following code to the `main.tf` file:

```
# Database Subnet

resource "aws_subnet" "database_subnet" {

  for_each = var.az_database_subnet


  vpc_id = aws_vpc.main.id
```

```
  availability_zone = each.key
  cidr_block        = each.value

  tags = {
    Name = "cloudy-lab-database-subnet-${each.key}"
  }
}
```

Create a Route Table and an Association

VPC Route Tables define what network routes exist within the VPC subnets. It does this with a set of route rules. In the below code, we will create a Route Table resource that directs all internet bound traffic to the internet gateway (created in a previous step).
Append the following code to the `main.tf` file:

```
# Route Table
resource "aws_route_table" "public_subnet_route_table" {
  vpc_id = aws_vpc.main.id

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

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

# Public subnet route table association
resource "aws_route_table_association"
"public_subnet_route_table_association" {
  for_each = var.az_public_subnet
```

```
  subnet_id      = aws_subnet.public_subnet[each.key].id
  route_table_id = aws_route_table.public_subnet_route_table.id
}
```

Plan Changes

Let's run the Terraform plan commands to see if we setup everything correctly.

```
$ terraform plan -var-file=env/dev.tfvars


Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------
---


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_internet_gateway.gw will be created
  + resource "aws_internet_gateway" "gw" {
      + arn      = (known after apply)


# ..removed extra lines
```

Apply Changes

Now, let's apply the changes to your account. Note that this you are responsible for any AWS charges incurred.

You will be prompted to proceed. Enter 'yes'.

```
$ terraform apply -var-file=env/dev.tfvars
```

```
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_internet_gateway.gw will be created
  + resource "aws_internet_gateway" "gw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "skillmix-lab-internet-gateway"
        }
      + vpc_id   = (known after apply)
    }

# ..removed extra lines

Plan: 11 to add, 0 to change, 0 to destroy.

Do you want to perform these actions in workspace "dev"?
  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 3s [id=vpc-0d4bb8f9ce6cbb5bf]
```

```
aws_internet_gateway.gw: Creating...

aws_subnet.public_subnet["ap-southeast-1a"]: Creating...

aws_subnet.database_subnet["ap-southeast-1b"]: Creating...

aws_subnet.private_subnet["ap-southeast-1b"]: Creating...

aws_subnet.private_subnet["ap-southeast-1a"]: Creating...

aws_subnet.database_subnet["ap-southeast-1a"]: Creating...

aws_subnet.public_subnet["ap-southeast-1b"]: Creating...

aws_subnet.private_subnet["ap-southeast-1b"]: Creation complete after
1s [id=subnet-0b3e9797da5ac6cf9]

aws_subnet.public_subnet["ap-southeast-1b"]: Creation complete after
1s [id=subnet-02c49822b3a975d43]

aws_subnet.database_subnet["ap-southeast-1b"]: Creation complete
after 1s [id=subnet-05c0d657a3a30d299]

aws_subnet.public_subnet["ap-southeast-1a"]: Creation complete after
1s [id=subnet-09fabf508790be076]

aws_subnet.database_subnet["ap-southeast-1a"]: Creation complete
after 1s [id=subnet-0ab6d187f919642d5]

aws_internet_gateway.gw: Creation complete after 1s
[id=igw-0040600c01a2c2436]

aws_subnet.private_subnet["ap-southeast-1a"]: Creation complete after
1s [id=subnet-028c1a2553b8bc65a]

aws_route_table.public_subnet_route_table: Creating...

aws_route_table.public_subnet_route_table: Creation complete after 1s
[id=rtb-0896ae38b27809989]

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1b"]: Creating...

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1a"]: Creating...

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1b"]: Creation complete after 0s
[id=rtbassoc-09637135f0e765f9b]

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1a"]: Creation complete after 0s
[id=rtbassoc-052d79321b6a8585d]
```

```
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.
```

Run Terraform Destroy

Last step is to destroy your infrastructure (if you want). Run this command:

```
$ terraform destroy -var-file=env/dev.tfvars


# ... this will be a list of all things to destroy


An execution plan has been generated and is shown below.

Resource actions are indicated with the following symbols:

  - destroy


# ..removed extra lines


Plan: 0 to add, 0 to change, 11 to destroy.


Do you really want to destroy all resources in workspace "dev"?

  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.public_subnet_route_table_association["ap
-southeast-1a"]: Destroying... [id=rtbassoc-052d79321b6a8585d]

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1b"]: Destroying... [id=rtbassoc-09637135f0e765f9b]

aws_subnet.private_subnet["ap-southeast-1b"]: Destroying...
[id=subnet-0b3e9797da5ac6cf9]

aws_subnet.database_subnet["ap-southeast-1b"]: Destroying...
[id=subnet-05c0d657a3a30d299]
```

```
aws_subnet.private_subnet["ap-southeast-1a"]: Destroying...
[id=subnet-028c1a2553b8bc65a]

aws_subnet.database_subnet["ap-southeast-1a"]: Destroying...
[id=subnet-0ab6d187f919642d5]

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1a"]: Destruction complete after 0s

aws_route_table_association.public_subnet_route_table_association["ap
-southeast-1b"]: Destruction complete after 0s

aws_subnet.public_subnet["ap-southeast-1b"]: Destroying...
[id=subnet-02c49822b3a975d43]

aws_subnet.public_subnet["ap-southeast-1a"]: Destroying...
[id=subnet-09fabf508790be076]

aws_route_table.public_subnet_route_table: Destroying...
[id=rtb-0896ae38b27809989]

aws_subnet.database_subnet["ap-southeast-1a"]: Destruction complete
after 1s

aws_subnet.private_subnet["ap-southeast-1a"]: Destruction complete
after 1s

aws_subnet.private_subnet["ap-southeast-1b"]: Destruction complete
after 1s

aws_subnet.database_subnet["ap-southeast-1b"]: Destruction complete
after 1s

aws_subnet.public_subnet["ap-southeast-1a"]: Destruction complete
after 1s

aws_subnet.public_subnet["ap-southeast-1b"]: Destruction complete
after 1s

aws_route_table.public_subnet_route_table: Destruction complete after
1s

aws_internet_gateway.gw: Destroying... [id=igw-0040600c01a2c2436]

aws_internet_gateway.gw: Still destroying...
[id=igw-0040600c01a2c2436, 10s elapsed]

aws_internet_gateway.gw: Destruction complete after 11s

aws_vpc.main: Destroying... [id=vpc-0d4bb8f9ce6cbb5bf]

aws_vpc.main: Destruction complete after 0s
```