Design a 3 Tier AWS VPC with NAT Gateways using Terraform

Step-01: Introduction

- Understand about Terraform Modules
- Create VPC using `Terraform Modules`
- Define `Input Variables` for VPC module and reference them in VPC Terraform Module
- Define `local values` and reference them in VPC Terraform Module
- Create `terraform.tfvars` to load variable values by default from this file
- Create `vpc.auto.tfvars` to load variable values by default from this file related to a VPC
- Define `Output Values` for VPC

Step-02: v1-vpc-module - Hardcoded Model

Step-02-01: How to make a decision of using the public Registry module?

1. Understand about Terraform Registry and Modules
2. We are going to use a VPC Module from Terraform Public Registry
3. Understand about Authenticity of a module hosted on Public Terraform Registry with HashiCorp Verified Tag
4. Review the download rate for that module
5. Review the latest versions and release history of that module
6. Review our feature needs when using that module and ensure if our need is satisfied use the module else use the standard terraform resource definition appraoch.
7. Review module inputs, outputs and dependencies too.

Step-02-02: Create a VPC Module Terraform Configuration

- c1-versions.tf
- c2-generic-variables.tf
- c3-vpc.tf
- Terraform AWS VPC Module

```
# Create VPC Terraform Module
module "vpc" {
```

```
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.78.0"

  # VPC Basic Details
  name = "vpc-dev"
  cidr = "10.0.0.0/16"
  azs                 = ["us-east-1a", "us-east-1b"]
  private_subnets     = ["10.0.1.0/24", "10.0.2.0/24"]
  public_subnets      = ["10.0.101.0/24", "10.0.102.0/24"]

  # Database Subnets
  create_database_subnet_group = true
  create_database_subnet_route_table= true
  database_subnets    = ["10.0.151.0/24", "10.0.152.0/24"]

  #create_database_nat_gateway_route = true
  #create_database_internet_gateway_route = true

  # NAT Gateways - Outbound Communication
  enable_nat_gateway = true
  single_nat_gateway = true

  # VPC DNS Parameters
  enable_dns_hostnames = true
  enable_dns_support = true

  public_subnet_tags = {
    Type = "public-subnets"
  }

  private_subnet_tags = {
    Type = "private-subnets"
  }

  database_subnet_tags = {
    Type = "database-subnets"
  }

  tags = {
    Owner = "kalyan"
    Environment = "dev"
  }

  vpc_tags = {
    Name = "vpc-dev"
  }
}
```

Step-03: Execute Terraform Commands

```
# Working Folder
terraform-manifests/v1-vpc-module
```

```
# Terraform Initialize
terraform init
Observation:
1. Verify if modules got downloaded to .terraform folder

# Terraform Validate
terraform validate

# Terraform plan
terraform plan

# Terraform Apply
terraform apply -auto-approve
Observation:
1) Verify VPC
2) Verify Subnets
3) Verify IGW
4) Verify Public Route for Public Subnets
5) Verify no public route for private subnets
6) Verify NAT Gateway and Elastic IP for NAT Gateway
7) Verify NAT Gateway route for Private Subnets
8) Verify no public route or no NAT Gateway route to Database Subnets
9) Verify Tags

# Terraform Destroy
terraform destroy -auto-approve

# Delete Files
rm -rf .terraform*
rm -rf terraform.tfstate*
```

Step-04: Version Constraints in Terraform with Modules

- Terraform Version Constraints
- For modules locking to the exact version is recommended to ensure there will not be any major breakages in production
- When depending on third-party modules, require specific versions to ensure that updates only happen when convenient to you
- For modules maintained within your organization, specifying version ranges may be appropriate if semantic versioning is used consistently or if there is a well-defined release process that avoids unwanted updates.
- Review and understand this carefully

Step-05: v2-vpc-module-standardized - Standardized and Generalized

- In the next series of steps we are going to standardize the VPC configuration

- c2-generic-variables.tf

```
# Input Variables
# AWS Region
variable "aws_region" {
  description = "Region in which AWS Resources to be created"
  type = string
  default = "us-east-1"
}
# Environment Variable
variable "environment" {
  description = "Environment Variable used as a prefix"
  type = string
  default = "dev"
}
# Business Division
variable "business_divsion" {
  description = "Business Division in the large organization this
Infrastructure belongs"
  type = string
  default = "HR"
}
```

Step-06: c3-local-values.tf

- Understand about Local Values

```
# Define Local Values in Terraform
locals {
  owners = var.business_divsion
  environment = var.environment
  name = "${var.business_divsion}-${var.environment}"
  common_tags = {
    owners = local.owners
    environment = local.environment
  }
}
```

Step-07: c4-01-vpc-variables.tf

```
# VPC Input Variables

# VPC Name
variable "vpc_name" {
  description = "VPC Name"
  type = string
  default = "myvpc"
}

# VPC CIDR Block
```

```
variable "vpc_cidr_block" {
  description = "VPC CIDR Block"
  type = string
  default = "10.0.0.0/16"
}

# VPC Availability Zones
variable "vpc_availability_zones" {
  description = "VPC Availability Zones"
  type = list(string)
  default = ["us-east-1a", "us-east-1b"]
}

# VPC Public Subnets
variable "vpc_public_subnets" {
  description = "VPC Public Subnets"
  type = list(string)
  default = ["10.0.101.0/24", "10.0.102.0/24"]
}

# VPC Private Subnets
variable "vpc_private_subnets" {
  description = "VPC Private Subnets"
  type = list(string)
  default = ["10.0.1.0/24", "10.0.2.0/24"]
}

# VPC Database Subnets
variable "vpc_database_subnets" {
  description = "VPC Database Subnets"
  type = list(string)
  default = ["10.0.151.0/24", "10.0.152.0/24"]
}

# VPC Create Database Subnet Group (True / False)
variable "vpc_create_database_subnet_group" {
  description = "VPC Create Database Subnet Group"
  type = bool
  default = true
}

# VPC Create Database Subnet Route Table (True or False)
variable "vpc_create_database_subnet_route_table" {
  description = "VPC Create Database Subnet Route Table"
  type = bool
  default = true
}


# VPC Enable NAT Gateway (True or False)
variable "vpc_enable_nat_gateway" {
  description = "Enable NAT Gateways for Private Subnets Outbound
Communication"
```

```
  type = bool
  default = true
}

# VPC Single NAT Gateway (True or False)
variable "vpc_single_nat_gateway" {
  description = "Enable only single NAT Gateway in one Availability Zone to
save costs during our demos"
  type = bool
  default = true
}
```

Step-08: c4-02-vpc-module.tf

```
# Create VPC Terraform Module
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.78.0"
  #version = "~> 2.0"

  # VPC Basic Details
  name = "${local.name}-${var.vpc_name}"
  cidr = var.vpc_cidr_block
  azs             = var.vpc_availability_zones
  public_subnets  = var.vpc_public_subnets
  private_subnets = var.vpc_private_subnets

  # Database Subnets
  database_subnets = var.vpc_database_subnets
  create_database_subnet_group = var.vpc_create_database_subnet_group
  create_database_subnet_route_table =
var.vpc_create_database_subnet_route_table
  # create_database_internet_gateway_route = true
  # create_database_nat_gateway_route = true

  # NAT Gateways - Outbound Communication
  enable_nat_gateway = var.vpc_enable_nat_gateway
  single_nat_gateway = var.vpc_single_nat_gateway

  # VPC DNS Parameters
  enable_dns_hostnames = true
  enable_dns_support   = true


  tags = local.common_tags
  vpc_tags = local.common_tags

  # Additional Tags to Subnets
  public_subnet_tags = {
    Type = "Public Subnets"
  }
  private_subnet_tags = {
```

```
    Type = "Private Subnets"
  }
  database_subnet_tags = {
    Type = "Private Database Subnets"
  }
}
```

Step-09: c4-03-vpc-outputs.tf

```
# VPC Output Values

# VPC ID
output "vpc_id" {
  description = "The ID of the VPC"
  value       = module.vpc.vpc_id
}

# VPC CIDR blocks
output "vpc_cidr_block" {
  description = "The CIDR block of the VPC"
  value       = module.vpc.vpc_cidr_block
}

# VPC Private Subnets
output "private_subnets" {
  description = "List of IDs of private subnets"
  value       = module.vpc.private_subnets
}

# VPC Public Subnets
output "public_subnets" {
  description = "List of IDs of public subnets"
  value       = module.vpc.public_subnets
}

# VPC NAT gateway Public IP
output "nat_public_ips" {
  description = "List of public Elastic IPs created for AWS NAT Gateway"
  value       = module.vpc.nat_public_ips
}

# VPC AZs
output "azs" {
  description = "A list of availability zones spefified as argument to this
module"
  value       = module.vpc.azs
}
```

Step-10: terraform.tfvars

```
# Generic Variables
aws_region = "us-east-1"
environment = "dev"
business_divsion = "HR"
```

Step-11: vpc.auto.tfvars

```
# VPC Variables
vpc_name = "myvpc"
vpc_cidr_block = "10.0.0.0/16"
vpc_availability_zones = ["us-east-1a", "us-east-1b"]
vpc_public_subnets = ["10.0.101.0/24", "10.0.102.0/24"]
vpc_private_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
vpc_database_subnets= ["10.0.151.0/24", "10.0.152.0/24"]
vpc_create_database_subnet_group = true
vpc_create_database_subnet_route_table = true
vpc_enable_nat_gateway = true
vpc_single_nat_gateway = true
```

Step-12: Execute Terraform Commands

```
# Working Folder
terraform-manifests/v2-vpc-module-standardized

# Terraform Initialize
terraform init

# Terraform Validate
terraform validate

# Terraform plan
terraform plan

# Terraform Apply
terraform apply -auto-approve
Observation:
1) Verify VPC
2) Verify Subnets
3) Verify IGW
4) Verify Public Route for Public Subnets
5) Verify no public route for private subnets
6) Verify NAT Gateway and Elastic IP for NAT Gateway
7) Verify NAT Gateway route for Private Subnets
8) Verify no public route or no NAT Gateway route to Database Subnets
9) Verify Tags
```

Step-13: Clean-Up

```
# Terraform Destroy
```

```
terraform destroy -auto-approve

# Delete Files
rm -rf .terraform*
rm -rf terraform.tfstate*
```