# Lab 17 – Production-Ready VPC Using Terraform (Multi-AZ, Public & Private Subnets)

**Creator:** Sandeep Kumar Sharma

---

## 🔍Scenario – Real Production Requirement

Your company is building a **microservices-based web application** that will run on EC2/containers in AWS. The architecture team has given you these requirements:

- All application servers **must run in a custom VPC**, not in the default VPC.
- The VPC must be **highly available across multiple AZs**.
- There should be:
- **Public subnets** for load balancers and bastion hosts.
- **Private application subnets** for EC2/ECS.
- **Private database subnets** for RDS in the future.
- Outbound internet access from private subnets must go via **NAT Gateway**.
- The VPC should be created in a **standard, reusable, production-ready way** using Terraform.

Your task: **Design and deploy this VPC using Terraform** in a clean, modular, and production-friendly style.

---

## 🕑Learning Objectives

- Understand how a production VPC is designed for real workloads.
- Use the official Terraform AWS VPC module in a more advanced way.
- Create multi-AZ public, private-app, and private-db subnets.
- Configure NAT Gateway, Internet Gateway, and route tables via module.
- Organize Terraform code as if used in a real project.

---

## 🔗Learning Outcomes

By the end of this lab, you will be able to: - Explain a basic production VPC design (public + private + db subnets). - Use Terraform modules to quickly create a complex VPC. - Control subnets, AZs, and NAT behavior using module inputs. - Tag your VPC and subnets properly for production visibility.

---

### Concept Explanation (Natural Style)

In small demos, we often create a single subnet or we directly use the **default VPC**. But in real companies, this is **not acceptable**.

Production environments usually follow a pattern like this: - A **custom VPC** with its own CIDR range, e.g., `10.0.0.0/16`. - Multiple **Availability Zones** for high availability. - **Public subnets** → internet-facing components (ALB, NAT, bastion). - **Private application subnets** → EC2/ECS/EKS services. - **Private database subnets** → RDS, Redis, etc. (no direct internet). - **NAT Gateway** → outbound access for private workloads only.

If we write all of this manually with pure Terraform resources, the configuration becomes large and repetitive.

Instead, we use the official **terraform-aws-modules/vpc/aws** module in a slightly more advanced configuration than what we used earlier. This gives us **production-ready networking** with minimal code.

---

## 💣Architecture Overview (High Level)

We will create: - 1 VPC: `10.0.0.0/16` - 3 Availability Zones: `ap-south-1a`, `ap-south-1b`, `ap-south-1c` - 3 public subnets (one per AZ) - 3 private application subnets (one per AZ) - 3 private database subnets (one per AZ) - 1 Internet Gateway - 1 NAT Gateway (shared) in a public subnet - Route tables automatically configured by the module

This will be your **network foundation** for future labs (ALB, ASG, RDS, etc.).

---

## 🛠️Step-by-Step Hands-On Lab

### Step 1: Create Project Folder

Open your terminal and run:

```
mkdir terraform-lab17-prod-vpc
cd terraform-lab17-prod-vpc
```

Create the main Terraform file:

```
touch main.tf
```

---

### Step 2: Write the Terraform Configuration

Open **main.tf** and paste the following code.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }

  required_version = ">= 1.4.0"
}

provider "aws" {
  region = "ap-south-1"
}

# ---------------------------------------------
# Production-Style VPC Using Official Module
# ---------------------------------------------
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "5.1.2"  # any recent stable version is fine

  name = "prod-lab17-vpc"
  cidr = "10.0.0.0/16"

  azs = [
    "ap-south-1a",
    "ap-south-1b",
    "ap-south-1c"
  ]

  # Public subnets (for ALB, bastion, NAT)
  public_subnets = [
    "10.0.1.0/24",
    "10.0.2.0/24",
    "10.0.3.0/24"
  ]

  # Private application subnets
  private_subnets = [
    "10.0.11.0/24",
    "10.0.12.0/24",
    "10.0.13.0/24"
  ]

  # Private database subnets
  database_subnets = [
```

```
    "10.0.21.0/24",
    "10.0.22.0/24",
    "10.0.23.0/24"
  ]

  # NAT & Internet Gateway configuration
  enable_nat_gateway = true
  single_nat_gateway = true

  enable_dns_hostnames = true
  enable_dns_support   = true

  public_subnet_tags = {
    Tier = "public"
  }

  private_subnet_tags = {
    Tier = "app"
  }

  database_subnet_tags = {
    Tier = "db"
  }

  tags = {
    Project     = "Terraform-Prod-Lab17"
    Environment = "prod-demo"
    Owner       = "Sandeep-Training"
  }
}

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

output "public_subnets" {
  description = "Public subnet IDs"
  value       = module.vpc.public_subnets
}

output "private_app_subnets" {
  description = "Private app subnet IDs"
  value       = module.vpc.private_subnets
}
```

```
output "private_db_subnets" {
  description = "Private db subnet IDs"
  value       = module.vpc.database_subnets
}
```

> 💡 Note: All CIDR ranges are inside `10.0.0.0/16` and separated logically for public, app, and db tiers.

---

### Step 3: Initialize the Project

Run:

```
terraform init
```

This will: - Download the AWS provider - Download the VPC module from the Terraform Registry - Prepare the working directory

---

### Step 4: Review the Plan

Run:

```
terraform plan
```

Carefully observe that Terraform is going to create: - 1 VPC - 3 public subnets - 3 private app subnets - 3 private db subnets - 1 Internet Gateway - 1 NAT Gateway - Route tables and associations

This is **exactly what you want** in a typical production-like network.

---

### Step 5: Apply the Configuration

Run:

```
terraform apply
```

Type `yes` when prompted.

Terraform will now build the complete VPC structure. This may take a couple of minutes because NAT Gateway allocation takes some time.

After success, you will see outputs like:

```
vpc_id = "vpc-0abc123xyz"
public_subnets = [
  "subnet-111...",
  "subnet-222...",
  "subnet-333...",
]
private_app_subnets = [
  "subnet-444...",
  "subnet-555...",
  "subnet-666...",
]
private_db_subnets = [
  "subnet-777...",
  "subnet-888...",
  "subnet-999...",
]
```

---

### Step 6: Validate in AWS Console

Go to AWS Console:

1. **VPC → Your VPCs**
2. Find VPC named `prod-lab17-vpc`.
3. Check the CIDR: `10.0.0.0/16`.
4. **Subnets**
5. Filter by VPC ID.
6. You should see 9 subnets.
7. Confirm tags for tiers: `public`, `app`, `db`.
8. **Internet Gateway**
9. Confirm an IGW is attached to the VPC.
10. **NAT Gateway**
11. Check that one NAT gateway is in a public subnet.
12. **Route Tables**

13. Public route table: route `0.0.0.0/0` → Internet Gateway.
14. Private route table(s): route `0.0.0.0/0` → NAT Gateway.

You have now a proper **production-style VPC**.

---

### Step 7: (Optional) Use This VPC in Future Labs

You can reuse this VPC in upcoming labs by: - Referencing `module.vpc.vpc_id` - Using `module.vpc.public_subnets[0]` for ALB - Using `module.vpc.private_subnets[0]` for EC2/ASG - Using `module.vpc.database_subnets[0]` for RDS

We will do this chaining in later advanced labs.

---

### Step 8: Destroy (If You Don't Want to Keep It)

If this was only for practice and you don't want ongoing costs:

```
terraform destroy
```

Type `yes` to confirm.

This will delete the entire VPC and all related resources created by this module.

> 🔍In real production, you **do not** destroy the VPC casually. This is only for lab purposes.

---

## 📝Summary

In this advanced lab, you: - Took a **realistic production scenario** for a microservices app. - Designed a **multi-AZ VPC** with public, app, and db subnets. - Used the **official Terraform VPC module** in an advanced way. - Enabled NAT gateway, DNS support, and tier tagging. - Prepared a strong foundation for future components like ALB, ASG, and RDS.

This is exactly the kind of VPC design you'll see in many real-world Terraform projects.

---

**End of Lab 17 – Production-Ready VPC**