

Terraform AWS Mono-Repo Walkthrough

Repository: `notoriousjay-terraform-aws-infra`

Jordan Suber – Software Engineer

June 21, 2025

Agenda

- 1 Project Structure
- 2 Terraform Workflow
- 3 Module Layout
- 4 Network & Security
- 5 SSH Key & Access
- 6 Conclusion

1. Repository Layout

High-level tree

```
notoriousjayy-terraform-aws-infra/  
|-- README.md  
|-- terraform.tfvars.example  
|-- environments/  
|   |-- dev/  
|       |-- backend.tf  
|       |-- main.tf  
|       |-- ...  
|-- modules/  
|   |-- ec2/  
|   |-- vpc/
```

- ****modules/**** – reusable building blocks (VPC, EC2, ...)
- ****environments/**** – per-environment compositions (dev, stg, prod)
- ****terraform.tfvars.example**** – team template for variable input
- ****backend.tf**** – S3 remote state + DynamoDB locking

2. Day-to-Day Workflow

❶ **cd environments/dev**

❷ Configure values:

```
1 cp terraform.tfvars.example terraform.tfvars
2 # edit: aws_region, key_name, ...
```

❸ Initialise plugins & back-end \$ terraform init

❹ Plan changes \$ terraform plan -out=tfplan

❺ Apply infrastructure \$ terraform apply tfplan

Why this matters

Remote state prevents drift and the saved plan supports CI/CD gates.

3. Module Highlights

VPC Module

```
1 resource "aws_vpc" "this" { ... }
2 resource "aws_subnet" "public" { ... }
3 resource "aws_internet_gateway" "this" { ... }
```

- Parametrised by CIDR, subnet list, AZ list.
- Outputs: vpc_id, public_subnet_ids.

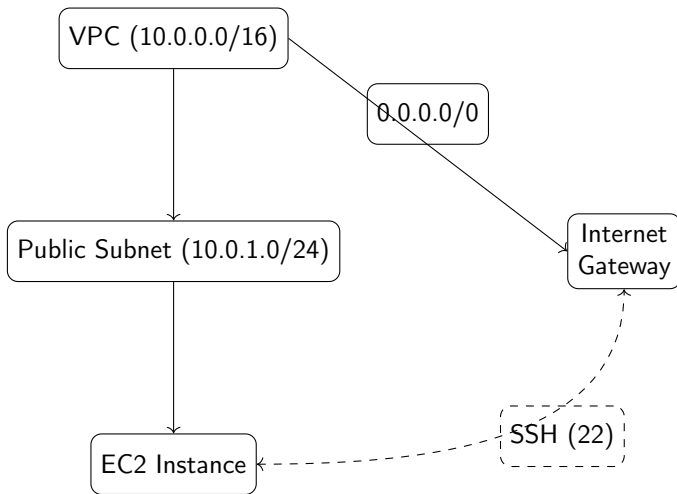
EC2 Module

```
1 data "aws_ami" "debian" { ... }      # latest Debian 11
2 resource "aws_security_group" "ssh" { ... }
3 resource "aws_instance" "this" { ... }
```

Inter-Module Wiring

```
1  module "vpc" {                                # (1) Network foundation
2      source      = "../..modules/vpc"
3      cidr_block  = var.vpc_cidr
4      ...
5  }
6
7  module "ec2" {                                # (2) Compute layer
8      source      = "../..modules/ec2"
9      vpc_id      = module.vpc.vpc_id
10     public_subnet_ids = module.vpc.public_subnet_ids
11     key_name     = aws_key_pair.debian.key_name
12 }
```

4. VPC Topology



Security-Group Definition

```
1  resource "aws_security_group" "ssh" {
2      name     = "${var.instance_name}-ssh"
3      vpc_id = var.vpc_id
4
5      ingress {
6          from_port = 22
7          to_port   = 22
8          protocol  = "tcp"
9          cidr_blocks = [var.ssh_ingress_cidr] # lock to VPN in prod
10     }
11
12     egress {
13         from_port = 0
14         to_port   = 0
15         protocol  = "-1"
16         cidr_blocks = ["0.0.0.0/0"]
17     }
```


5. Key-Pair Automation

```
1 resource "tls_private_key" "ssh" {
2     algorithm = "RSA"
3     rsa_bits  = 4096
4 }
5
6 resource "aws_key_pair" "debian" {
7     key_name   = var.key_name
8     public_key = tls_private_key.ssh.public_key_openssh
9 }
```

SSH into the Instance — Retrieve & Static Connect

After terraform apply, you'll see something like:

```
1 instance_public_ip = "3.145.128.42"  
2 private_key_pem    = "<redacted>"
```

1 Save & secure your private key:

```
1 $ terraform output -raw private_key_pem > dev.pem  
2 $ chmod 600 dev.pem
```

2 SSH in (static example):

```
1 $ ssh -i dev.pem admin@3.145.128.42
```

SSH into the Instance — Dynamic Connect & Tip

- ③ Or SSH dynamically via Terraform outputs:

```
1 $ ssh -i dev.pem \  
2   admin@$(terraform output -raw instance_public_ip)
```

Tip: Add to ~/.ssh/config

```
1 Host dev-ec2  
2   HostName $(terraform output -raw instance_public_ip)  
3   User      admin  
4   IdentityFile ~/dev.pem
```

Key Takeaways

- Mono-repo split into `**modules**` and `**environments**`.
- Remote S3 back-end + locking keeps state safe.
- Clean outputs wire VPC → EC2 without duplication.
- Key-pair generated at apply-time; never committed.
- One-command onboarding: `terraform init && apply`.

Welcome aboard – Happy provisioning!

Questions?