**Project : 3-Tier Web Application Infrastructure Deployment on AWS using Terraform**

**Obejctive :**

The objective of this project is to design, implement, and automate the deployment of a secure and scalable 3-tier web application infrastructure on AWS using **Terraform modules** and **automation tools** such as **Ansible** or **Terraform provisioners**. The architecture includes:

* A **Web Tier** (NGINX on EC2) to serve a registration form,
* An **Application Tier** (PHP on EC2) to process form submissions and interact with the database,
* A **Database Tier** (Amazon RDS) to store user data securely.

The project follows Infrastructure as Code (IaC) principles to ensure consistency, repeatability, and automation of infrastructure provisioning and software configuration.

**Introduction :**

In today’s cloud-first world, deploying scalable and secure infrastructure is critical for modern web applications. This project demonstrates the use of **Infrastructure as Code (IaC)** with **Terraform** to provision a complete **3-tier architecture** on Amazon Web Services (AWS). The infrastructure is organized into modular Terraform code to improve reusability, scalability, and manageability.

The architecture consists of:

* A **Web Tier** to handle incoming traffic (NGINX)
* An **Application Tier** on a private EC2 for backend logic (PHP).
* A **Database Tier** hosted on **Amazon RDS (MySQL)** within a private subnet, securely storing user registration data.

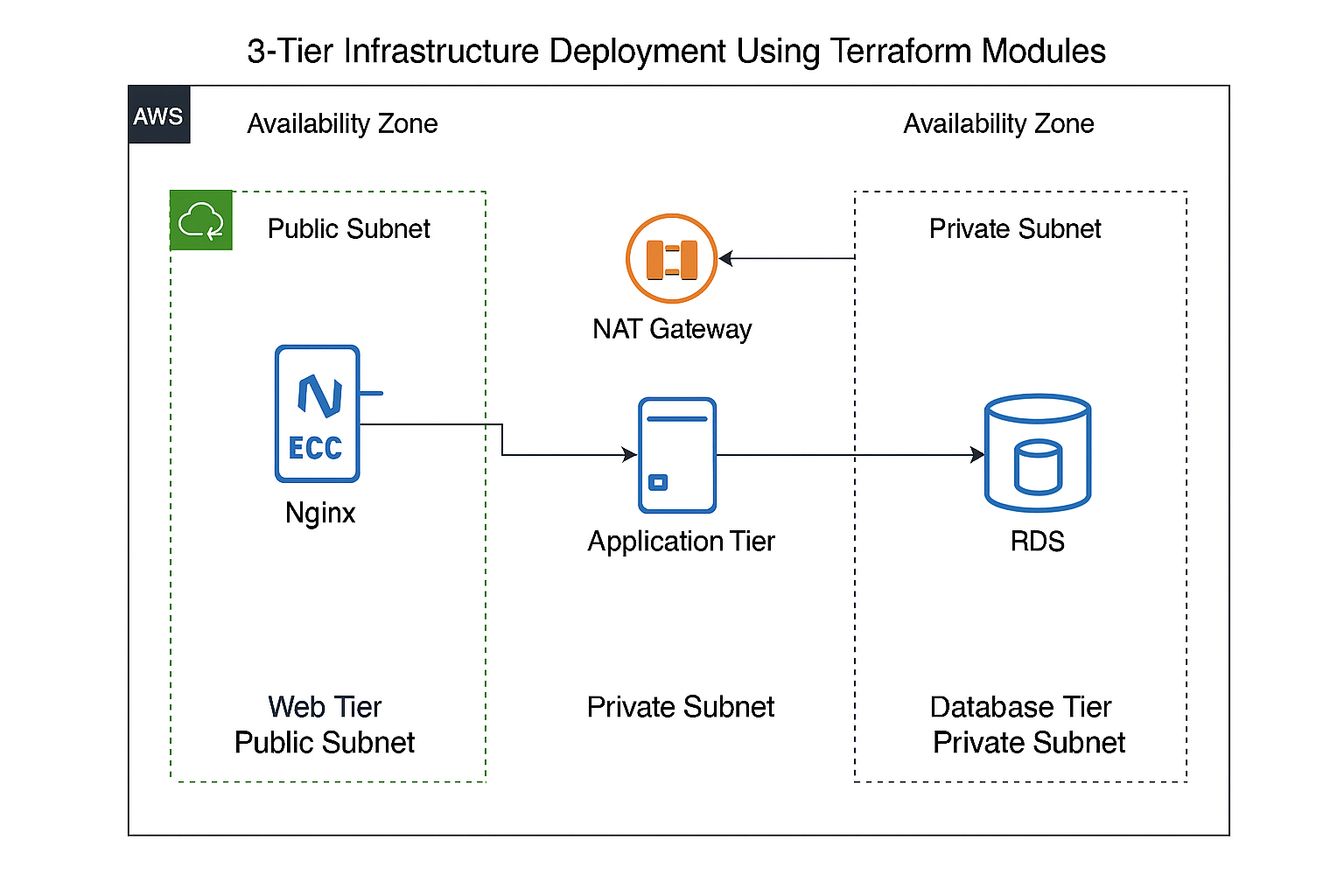
To further enhance automation and consistency, **Terraform provisioners** are used to install software, deploy code, and configure each server.

This project serves as a hands-on implementation of cloud infrastructure deployment using best practices for **security**, **network segmentation**, and **automation**.

**Techonology Stack :**

|  |  |
| --- | --- |
| IaC | Terraform |
| Cloud Provider | AWS |
| Software | NGINX , PHP , MySQL |
| OS | Amazon linux |
| Configuration management | Terraform Provisioners |

**System architecture diagram :**



**Project Structure :**

3-Tier\_Architecture/

- main.tf

- variables.tf

- outputs.tf

- terraform.tfvars

- modules/

- vpc/

- web/

- app/

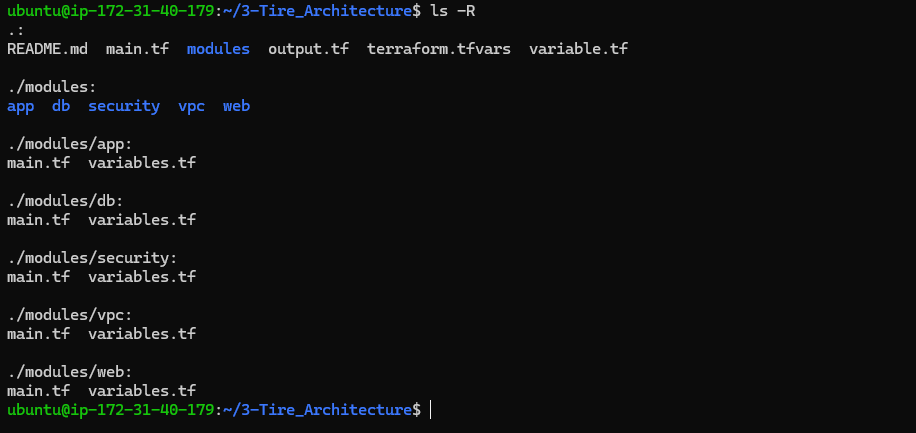
- db/

- security/

**Implementation Steps :**

**🔹 Step 1: Create Folder Structure**

* Create the main project folder named 3-Tier\_Architecture.
* Inside, create Terraform root files:
  + main.tf, variables.tf, outputs.tf, terraform.tfvars
* Create a modules/ directory with subfolders for:
  + vpc, web, app, db, security
* Each module contains:
  + main.tf – main logic
  + variable.tf – input variables

****

### 🔹 Step 2: VPC and Networking Configuration (modules/vpc)

* Create a **custom VPC** with CIDR block (e.g., 10.0.0.0/16)
* Create **2 public** and **2 private** subnets across **2 Availability Zones**
* Create and attach:
  + **Internet Gateway** to VPC (for public access)
  + **NAT Gateway** in public subnet (for private instances to reach the internet)
* Define **Route Tables**:
  + Public subnets route to Internet Gateway
  + Private subnets route through NAT
* Associate the route tables with the correct subnets

**🔹 Step 3: Security Configuration (modules/security)**

* Create **Security Groups**:
  + **Web Tier**: Allow SSH (22) and HTTP (80) from the internet
  + **App Tier**: Allow access from Web Tier on port 80
  + **DB Tier**: Allow access from App Tier on port 3306 (MySQL)
* Optionally create **NACLs** for additional control

**🔹 Step 4: Web Tier EC2 Setup (modules/web)**

* Launch an **EC2 instance** in the **public subnet**
* Use **user\_data script** or **remote-exec** to:
  + Install **NGINX**
  + Deploy a static HTML page (optional)
* Attach the Web Tier Security Group
* Output the EC2 instance's **public IP**

**🔹 Step 5: Application Tier EC2 Setup (modules/app)**

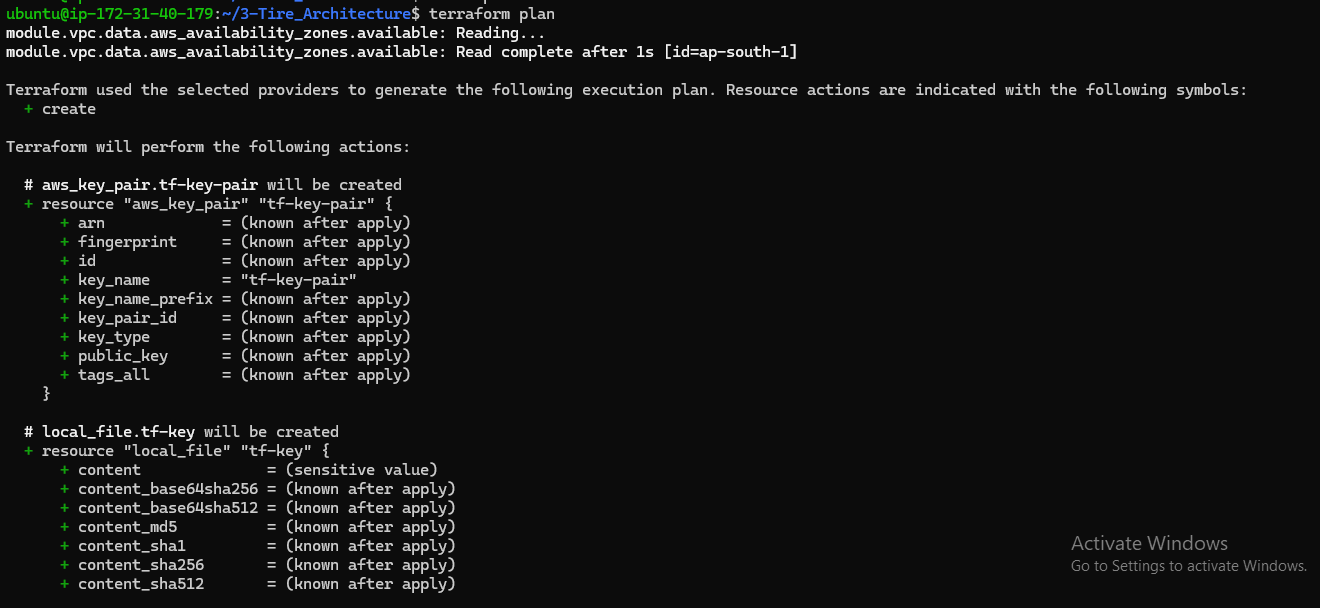
* Launch an **EC2 instance** in the **private subnet**
* Use **user\_data** or **provisioners** to:
  + Install **PHP** and **MySQL client**
  + (Optional) Configure backend logic (e.g., submit.php)
* Attach the App Tier Security Group

### 🔹 ****Step 6: Database Tier Setup (****modules/db****)****

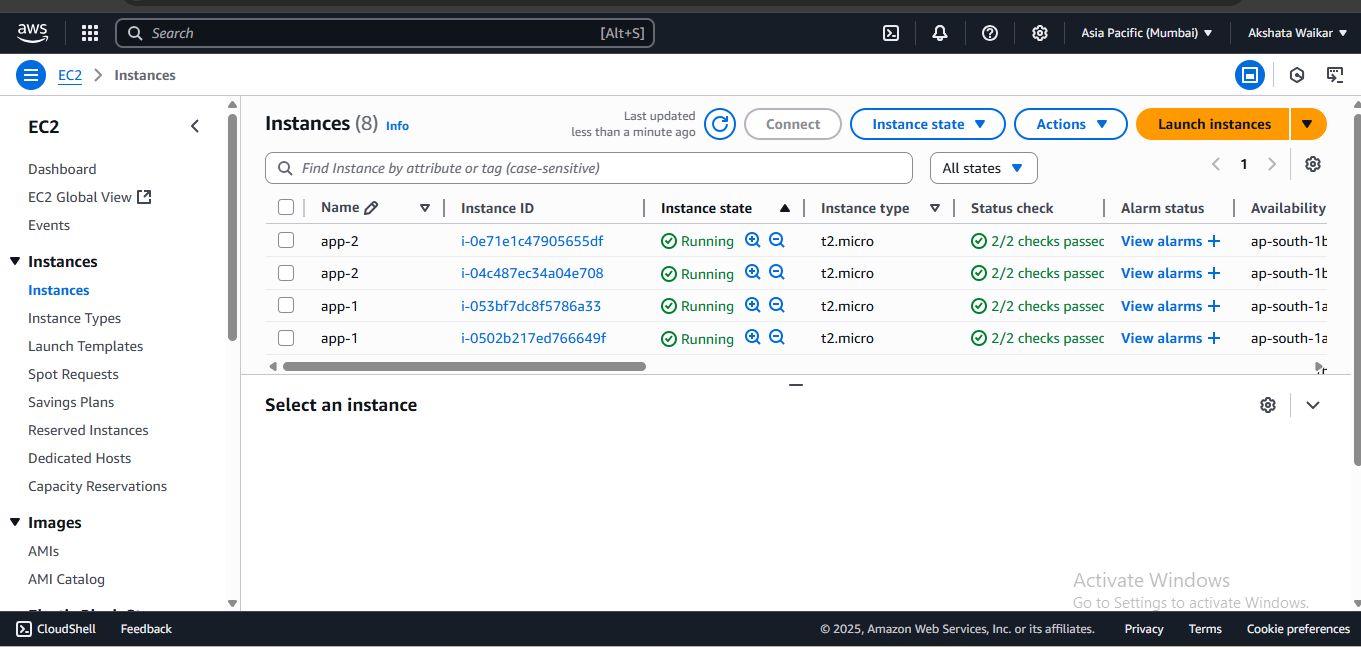
* Provision an **Amazon RDS** instance (MySQL)
* Create **DB Subnet Group** using private subnets
* Configure:
  + Engine type and version
  + DB instance class (e.g., db.t2.micro)
  + Username and password (from terraform.tfvars)
  + Attach **DB Security Group** (only allow from App Tier)
* Output the **RDS endpoint**

**🔹 Step 6: Deploy Infrastructure**

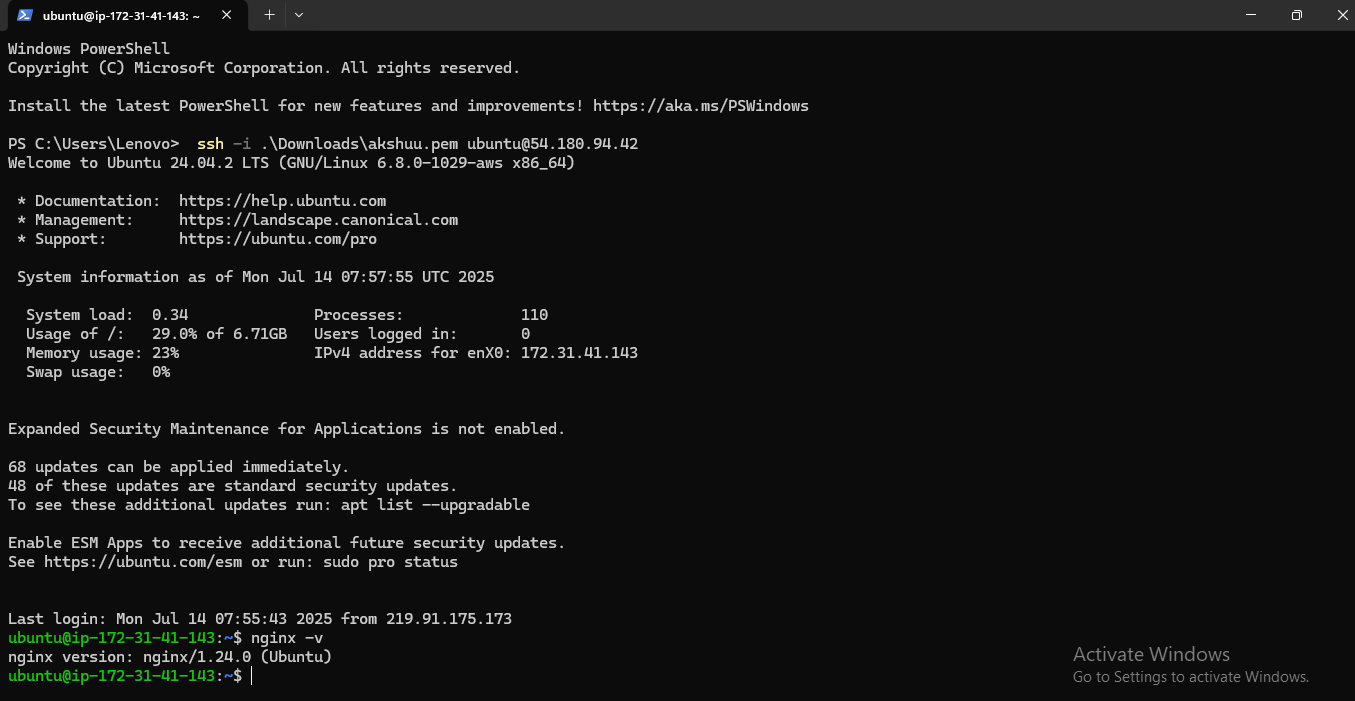
* + - terraform init # Initialize working directory
    - terraform plan # Preview changes
    - terraform apply # Deploy resources



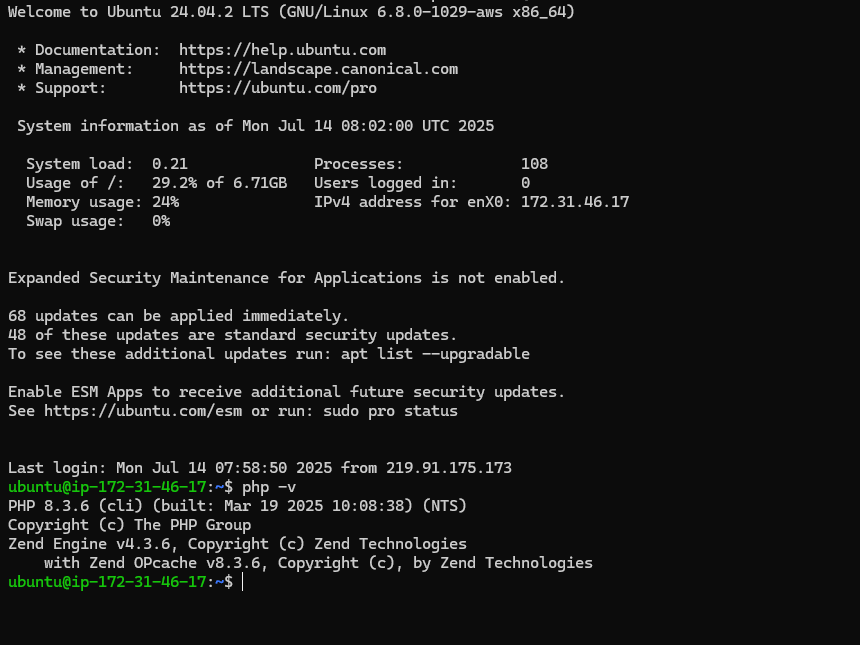
🔹 **Step 7: Verify infrastructure**

****

**Check NGINX installation :**



**Check PHP installation :**



**🔹 Step 8: Clean-up**

* + - **terraform destroy**

**Conclusion :**

In this project, we successfully designed and deployed a **3-Tier Web Application Infrastructure** on **Amazon Web Services (AWS)** using **Terraform modules** and automation tools like **Ansible** or **Terraform provisioners**. The architecture was built using best practices for **network isolation**, **security**, and **scalability**.

Each layer of the infrastructure was carefully separated:

* The **Web Tier** hosted a frontend using **NGINX** on a public EC2 instance.
* The **Application Tier** processed backend logic using **PHP** on a private EC2 instance.
* The **Database Tier** used **Amazon RDS (MySQL)** to securely store data, accessible only by the App Tier.

We also followed **Infrastructure as Code (IaC)** principles by organizing the Terraform code into reusable modules, which enhanced clarity, reusability, and scalability of the deployment process. This modular approach simplifies management and enables rapid changes or scaling in production environments.

Overall, this project provided practical experience in deploying a secure and efficient multi-tier cloud architecture, while demonstrating the power of automation in real-world cloud infrastructure management.

**GitHub Repository link** : https://github.com/Akshata-Waikar/3-Tire\_Architecture

**Prepared By:**  
**Akshata Waikar**  
Cloud & DevOps Intern  
Cravita Technologies