**3 Tier Architecture Application Using Terraform, Ansible and AWS Instance**

The three-tier architecture means our project developed and maintained into three separate layers or parts.

1. Presentation Layer or User Interface Layer (UI)
2. Business Logic Layer (For write a Logic Code)
3. Data Access Layer (DAL for Connectivity with the database)

**Presentation Layers or User Interface Layers:**

* Presentation layer is a user interface where we can design our webpage or windows page. This is basically Html or aspx page where we can make design and controls. User Interface layer where users can input and get some result as output.

**Business Logic Layers:**

* Business Logic Layer is intermediate or middle layer that communicate with Presentation Layer and Data Access Layer. This layer used to validate input condition and correct the data before calling method from database layer.

**Data Access Layer:**

* Data Access Layer used to make connection with database server. In this layer we can write database query, store procedure for insert, update, delete, create, select operation on database. This layer only communicates with business logic layer.

Here in this document, I am creating new instance with Terraform as first layer presentation layer and then as second layer configuring the instance with Ansible in the third step connect instance with database RDS as third layer data access layer.

**Creating Provision AWS Infrastructure with Terraform:**

1. Here I am doing provision an Amazon Machine Image (AMI) on Amazon Web Services (AWS).

**What it does:** Terraform is known as a Terraform configuration. i am configuration to launch a single AWS EC2 instance

Prerequisites: we need

* I am using An AWS account
* The AWS CLI installed
* With your account created and the CLI installed to configure the AWS CLI

For configuration process create a fil at ~/.aws/credentials in Linux.

configuration below into tier.tf and save it. An example terraform config:

# Configure AWS credentials:

provider "aws" {

access\_key = “ACCSESS\_KEY\_HERE”

Secret\_key = “ACCSESS\_KEY\_HERE”   
 profile = "default"  
 region = "us-east-1"  
}

# Deploye a virtual server (EC2):

resource "aws\_instance" "tier" {  
 ami = "ami-12345678"  
 instance\_type = "t2.micro"

key\_name = "${var.key\_name}"

subnet\_id = "${module.network.public\_subnets[0]}"

vpc\_security\_group\_ids = ["${aws\_security\_group.bastion.id}"]

associate\_public\_ip\_address = "true"

}

# Deploy a database (RDS)

resource "aws\_db\_instance" "source-db" {

engine = "mysql"

instance\_clas = "db.t2.micro"

name = "db\_name"

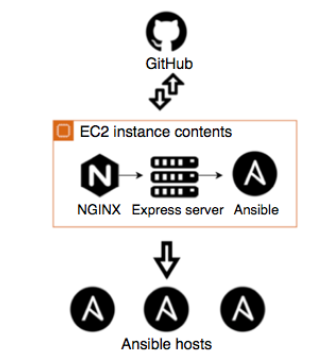
username = "DB\_USERNAME"

password = "DB\_PASSWORD"

}

with the above created infrastructure using terraform apply for check Visit the EC2 console to see the created EC2 instance.

1. **Here I am configuring EC2 instance with the help of Ansible configuration management:**



For Ansible and AWS ec2 instance environments configuration and setup we need

* AWS EC2 – Setup Boto, boto3, python version for Ansible
* Setup AWS Authentication Before Running the Playbook with Ansible EC2
* Ansible Playbook to connect EC2 instance
* Getting ready to execute the playbook – Ansible AWS EC2
* Execution Part – Run the playbook with Ansible EC2
* A Playbook with Ansible EC2 & Ansible Vault – Secure Approach
* Saving the AWS Secrets
* Execution of Ansible AWS Playbook tier

# Launch instances, runs some tasks

amazon.aws.ec2:

key\_name: mykey

instance\_type: t2.micro

image: ami-123456

network\_interfaces:

assign\_public\_ip: yes

Task:

name: Configure instance(s)

hosts: launched

become: True

gather\_facts: True

roles:

- my\_awesome\_role

- my\_awesome\_test

1. **Once the EC2 instance has been created and deploy a database Connect to the RDS DB instance:** MySQL Workbench client to connect to the host:

# Start a new connection and select Standard TCP/IP over SSH for the Connection Method.

# Enter the following details about the EC2 instance for the SSH settings:

SSH Hostname = Enter the public DNS name of the EC2 instance.

SSH Username = Enter the username for your EC2 instance.

SSH Key File = Select the private key that was used when the EC2 instance was created.

# MySQL instance settings:

resource "aws\_instance" "source-db" {  
 MySQL Hostname: RDS DB instance endpoint  
 MySQL Server port: Enter 3306 or the custom port   
 name = "db\_name"

username = "DB\_USERNAME"

password = "DB\_PASSWORD"

Diagram

Description automatically generated

Image:2 Workflow using Terraform and Ansible in AWS