### **Approach for Automating Infrastructure using Terraform**

#### **Project Overview**

The objective of the project is to automate the provisioning of infrastructure using Terraform to launch an EC2 instance on AWS and then use Ansible to install Jenkins, Java, and Python on the instance. This project will utilize a lab instance running Ubuntu, provided by the training, which has AWS credentials including access key, secret key, and STS token. Additionally, the infrastructure as code (IaC) will be maintained in GitHub, and a Jenkins pipeline will be implemented to run Terraform jobs with approval stages.

#### **Prerequisites**

1. AWS Credentials (provided by the Lab):
   * Access Key ID: Required for programmatic access to AWS.
   * Secret Access Key: Paired with the access key ID to sign programmatic requests to AWS.
   * STS Token: Temporary security token required for making API requests.
2. These credentials will be provided by the training lab instance.
3. Lab Instance (already provisioned):
   * Operating System: The lab instance should be running Ubuntu.
   * Pre-installed Software: Ensure the lab instance has necessary software pre-installed as per the training's GitHub repository guidelines. This includes:
     + Terraform: Follow the installation steps provided in the training GitHub repository to install Terraform on the lab instance.
     + Jenkins: Jenkins setup will be part of the Ansible playbook, but ensure the lab instance has the capability to run the Jenkins pipeline.
     + Ansible: Ensure Ansible is installed on the lab instance for configuration management.
4. GitHub Repository (training repository that have reference code):
   * Repository Access: Access to the training GitHub repository which contains code samples and installation scripts for Terraform, Jenkins pipeline, and other necessary configurations.

#### 

#### **Solution/Implementation Approach**

1. Define Project Requirements and Prerequisites:
   * Terraform: Tool for infrastructure provisioning.
   * AWS Account: Active AWS account with access credentials provided by the lab.
   * Key Pair: An existing AWS key pair to access the EC2 instance.
   * Ansible: Tool for configuration management and application deployment.
   * GitHub Repository: Repository to store the Terraform and Ansible code.
   * Jenkins: Continuous integration and delivery tool for automating the pipeline.
2. Set Up Terraform Configuration:
   * Provider Configuration: Configure the AWS provider in Terraform using the access key, secret key, and STS token.
   * Resource Definition: Define the EC2 instance resource, specifying details such as AMI, instance type, key pair, and security group settings.
3. Initialize and Apply Terraform Configuration:
   * Initialize Terraform: Run terraform init to initialize the configuration.
   * Apply Configuration: Use terraform apply to create the specified infrastructure on AWS.
4. Retrieve EC2 Instance Details:
   * Output Configuration: Configure Terraform to output the public IP of the EC2 instance for easy access.
5. Establish Secure Connection to EC2 Instance:
   * Use the provided key pair to connect to the EC2 instance via SSH.
6. Prepare Ansible Playbook:
   * Playbook Creation: Write an Ansible playbook to install Jenkins, Java, and Python.
   * Tasks Definition: Define tasks within the playbook to add the Jenkins repository, install Java and Python, and start the Jenkins service.
7. Configure Ansible Inventory:
   * Inventory File: Create an inventory file to specify the EC2 instance details for Ansible to manage.
8. Execute Ansible Playbook:
   * Run the Ansible playbook to configure the EC2 instance with Jenkins, Java, and Python.
9. Maintain Code in GitHub:
   * Repository Setup: Create a GitHub repository to store the Terraform and Ansible code.
   * Version Control: Use Git for version control to track changes and collaborate.
10. Implement Jenkins Pipeline:
    * Pipeline Creation: Create a Jenkins pipeline to automate the Terraform job.
    * Stages Definition: Define stages in the pipeline for code checkout, Terraform plan, approval, and Terraform apply.
    * Approval Stages: Implement manual approval stages in the Jenkins pipeline to ensure changes are reviewed before application.
11. Verification and Testing:
    * Verify the installation of Jenkins, Java, and Python on the EC2 instance.
    * Test the Jenkins setup by accessing the Jenkins web interface.
    * Ensure the Jenkins pipeline works as expected, with appropriate approvals.

#### **Summary**

This approach outlines the high-level steps required to automate the provisioning and configuration of infrastructure using Terraform and Ansible, with the addition of maintaining the code in GitHub and implementing a Jenkins pipeline for automation with approval stages. By following this structured approach, you can efficiently deploy and manage a Jenkins server on an EC2 instance, ensuring that the necessary software is installed and configured correctly, with proper version control and automation practices in place.