University of Science and Technology of Hanoi

Cloud and big data

Project report

Student id: ict2440050

Student name: Nguyen Vu Bach

I. Architecture Overview

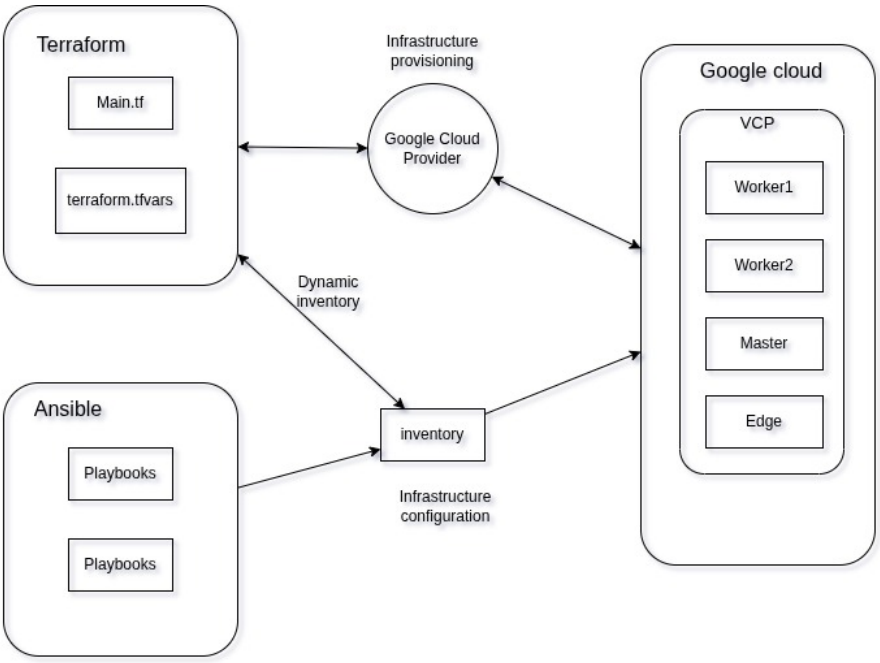

Figure 1: Overall architecture

1. Infrastructure Components

The automated deployment provisions the following components:

- VPC Network: A custom virtual network with a dedicated subnet for Spark nodes.
- Firewall Rules: Rules allowing SSH access (restricted to specific IPs) and internal communication between cluster nodes.
- Compute Engine Instances:
  - Spark Master Node
  - Worker Nodes
  - Edge Node for job submission
- IAM Roles: Used to restrict permissions and enforce secure access.

2. Automation Workflow

The workflow consists of:

1. Terraform provisions all GCP resources.
2. Dynamic inventory generates an Ansible host list using Terraform output.
3. Ansible installs Java, configures Spark, sets environment variables, and enables systemd services.
4. WordCount JAR is uploaded to the edge node and executed on the cluster.

II. Methodology

1. Terraform Configuration

The Terraform configuration focuses on modularity and security:

- Service accounts created per role (master, worker, edge)
- Firewall rules limited by `source_ranges = [var.admin_ip]`
- Instances tagged with logical roles (`spark`, `master`, `worker`)
- SSH keys passed via metadata for VM-level authentication

Compute Engine nodes use a common machine type (e.g., e2-standard-4), persistent disks, and startup scripts when required.

2. Ansible Roles

Three roles are defined:

- common: installs dependencies (Java 8, system tools), sets hostnames, configures SSH.
- master: configures Spark master services, exposes the Web UI, and enables systemd units.
- worker: configures Spark worker daemons and links them to the master.

A dynamic inventory script reads Terraform outputs to determine IP addresses of each VM.

3. Security

Security is enforced through:

- SSH key-only authentication
- Restricted firewall ingress
- IAM roles limiting resource modification
- No plaintext passwords stored in Terraform or Ansible files

III. Benchmark

Execution times were recorded for executor configurations ranging from 2 to 6 total cores using file sample.txt which is the French poem to benchmark. As expected, performance improves as additional worker nodes are added. However, diminishing returns appear beyond 6 cores due to communication overhead and limits on parallelism for the relatively small WordCount dataset.

Total Executor Cores	Execution Time (s)
2	3.419
4	1.215
6	0.368

Table 1: WordCount performance on Spark cluster.

The results indicate that the automated deployment provides a functional, scalable Spark cluster suitable for distributed workloads.

IV. Conclusion

This project successfully demonstrates a fully automated pipeline for deploying an Apache Spark cluster on GCP using Terraform and Ansible. By defining infrastructure as code and automating all configuration steps, the solution delivers a reproducible, secure, and scalable environment for big data processing.