University of Science and Technology of Hanoi

Cloud and Big Data

Project Report

Student ID: 2440057

Student name: Nguyen Nhat Anh

I. Architecture Overview

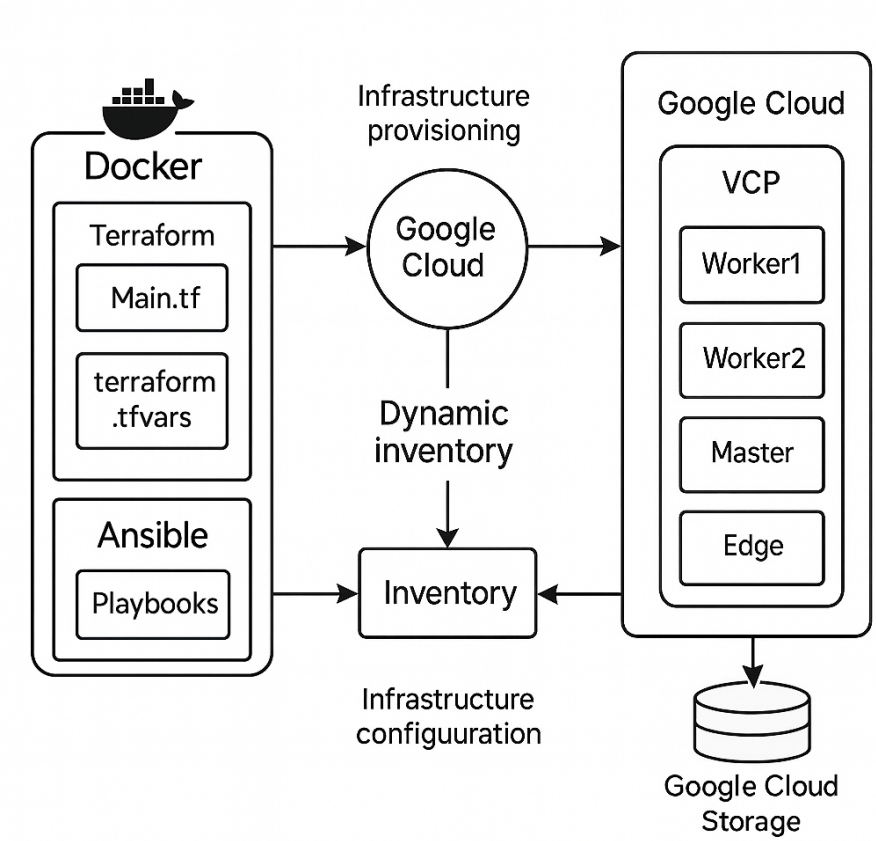

Figure 1: Overall architecture

1. Infrastructure Components

The automated deployment provisions the following cloud components:

- **VPC Network**: A dedicated virtual network for Docker-based Spark nodes.
- **Firewall Rules**: Allow internal container communication and restricted SSH for administration.
- **Compute Resources (Container-based)**:
  - Dockerized Spark Master
  - Dockerized Worker Containers
  - Dockerized Edge Container for job submission
- **Google Cloud Storage (GCS)**:
  - Stores application JAR files
  - Stores input datasets for Spark jobs
  - Provides scalable, durable blob storage for cluster data

2. Automation Workflow

The workflow consists of:

1. Terraform provisions the VPC, firewall rules, and a host VM used to run Docker containers.
2. Terraform outputs generate a dynamic host inventory for Ansible.
3. Ansible installs Docker, deploys Spark Master/Worker/Edge containers, and configures networking.
4. The WordCount JAR and input text are uploaded to Google Cloud Storage.
5. The Edge container pulls data from GCS and submits the Spark job to the cluster.

II. Methodology

1. Terraform Configuration

The Terraform design prioritizes modularity and minimal infrastructure:

- One compute instance used as a **Docker host** for all Spark containers
- GCS bucket automatically created for:
  - storing datasets,
  - storing compiled JAR applications,
  - logs or result output
- Firewall rules restricted using `source_ranges = [var.admin_ip]`
- Service accounts created to provide GCS access for containers through key-less metadata authentication

2. Ansible Roles

Three roles automate the deployment:

- **common**: Installs Docker, configures networking, prepares directories
- **master**: Starts the Spark Master container with published ports
- **worker**: Starts Spark Worker containers and links them to the Master

The dynamic inventory reads Terraform outputs to determine the IP of the Docker host.

3. Security

Security is enhanced across the pipeline through:

- Firewall allowing only necessary ingress traffic
- IAM roles restricting access to GCS buckets
- SSH key authentication to the Docker host
- No exposed Docker APIs; container control is local only

III. Benchmark

Execution times were recorded for deployments using 2–6 worker cores. Input data was stored in **Google Cloud Storage**, downloaded by the Edge container, and passed to Spark.

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

Table 1: WordCount performance using Docker-based Spark cluster with GCS input.

Results show strong scalability for distributed processing even on a containerized architecture.

IV. Conclusion

This project demonstrates a fully automated deployment of a Spark cluster based on Docker containers running on Google Cloud. Terraform manages cloud infrastructure and storage, while Ansible configures the Docker environment and launches all Spark services.

Using GCS for dataset storage simplifies workflow management and ensures scalable, reliable access for distributed workloads.