Spark WordCount Experiment on GCP

1 Setup

I deployed a Spark cluster on Google Cloud Platform with one master node, one edge node, and two worker nodes. All nodes run Debian, and I used Ansible to install dependencies and configure Spark. Hadoop installation was skipped to simplify the WordCount job.

1.1 Playbook 1: Install and Configure Spark

The first playbook handles:

- Installing Java.
- Downloading Spark 3.5.0 and extracting it to `/opt/spark`.
- Configuring the Spark master and workers, writing the workers file, and adding the master's internal IP to `/etc/hosts`.
- Starting the master and workers, ensuring the Spark UI is reachable before moving on.

1.2 Playbook 2: Compile and Run WordCount

The second playbook runs on the edge node and handles:

- Installing Java and Maven. The second playbook installs Java and Maven again on the edge node. It's idempotent so it doesn't add any overhead. I kept it because I was playing a lot with VMs.
- Compiling `WordCount.java` using the Spark classpath.
- Creating a JAR and a sample text file.
- Submitting the job to the Spark master with `spark-submit`.

2 Terraform and Infrastructure

The GCP infrastructure was provisioned using Terraform. The setup includes:

- Defining VM instances for master, edge, and worker nodes with Debian images.
- Assigning internal and external IPs to each VM.

- Setting up firewall rules for SSH and Spark ports (7077 for master-worker communication and 8080 for the Spark UI).
- Using Ansible to automatically configure Spark on all VMs after provisioning.

Terraform made it easy to quickly spin up and tear down the environment. Changing the number of workers or their instance types only requires updating the Terraform configuration and reapplying it.

3 Issues

- HDFS was very difficult to set up and maintain, so I don't use it in the demo but it was used to check performance to have "big-data" (e,g: increasing number of workers)
- Increasing the number of workers did not significantly improve performance.
- Spark-submit requires exact paths and permissions. Jobs fail silently with non-zero exit codes if resources are missing or paths are misconfigured.
- Using Ansible to manage internal IPs worked, but race conditions can occur if the master is not ready before workers start.

4 Conclusion

The experiment demonstrates a basic Spark setup on GCP using Terraform for infrastructure and Ansible for configuration. Proper care must be taken with paths, process cleanup, and worker-master coordination. Adding more workers does not automatically translate to faster jobs for small workloads.