Cloud and Big Data Final Report: Automation of Apache Spark Deployment using Terraform and Ansible on Google Cloud Platform

Nguyen Thanh Binh - 2440052
Information and Communication Technology
University of Science and Technology of Hanoi
Hanoi, Vietnam
binhnt2440052@usth.edu.vn

***Abstract*—In this report, we will present how we implement Terraform to create infrastructure for Apache Spark, specifically, deploy Spark nodes as virtual machine (VM) instances, and Ansible to automate the configuration of Spark and related software. The result currently is still not obtainable, as connection to a cloud VM instance is not successful.**

***Index Terms*—Automation, Apache Spark Deployment, Ansible, Terraform, Cloud Platform.**

I. Introduction

A. What is Terraform?

Terraform is an open-source tool that lets you define infrastructure components and their relationships using a high-level configuration language.

In Terraform's human-readable configuration files, you can specify the desired state of your infrastructure and Terraform automatically works out how to get to that state. These files can be versioned, shared, and reused to provide a consistent way to manage your infrastructure, from compute and storage resources, to DNS and SaaS features.

Terraform can be used with various cloud providers, in multi-cloud infrastructures, and on-premises environments.

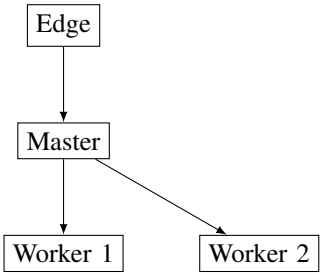
B. What is Ansible?

While Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy.

II. Resources architecture:

For this project, we will use Linux VM operating system and Apache Spark to construct Java automated application.

Our architecture model for this small project will have two Spark worker nodes, one Spark master node and one Spark edge node for submitting application.



III. Methodology

A. Deploy Infrastructure with Terraform:

For Terraform, we will create 6 terraform files, each with specified task for the project:

- `provider.tf`: Terraform file for the provider.
- `sparkinstance.tf`: This file will provide the VM instances and its IP adresses, both public and private. For the ssh keys, we will generate it from our local machine and add it to the metadata block.
- `network.tf`: This file will provide the VPC network with firewall rules. This is where the main problem that connecting to a VM instance becomes daunting, and the result will says it all.
- `output.tf`: Prints out the output once the terraform code successfully applies, and successfully initialized VM instances, network and firewall rules. This gives the owner to get the information of each VM instance.
- `templates/ansible_template.tpl`: The template file where terraform will automatically fills in to create inventories for executing playbooks.
- `output_render.tf`: Creates inventory at the destination directory, in this case the Ansible inventory, for playbooks, of course.

For Ansible playbook, we will assign to Master and Worker node instructions:

- `Master`: Install necessary packages - Java, Hadoop, Spark. Get public and private keys to connect to the worker nodes for assigning task.
- `Worker`: Executes the tasks given by the Master node.

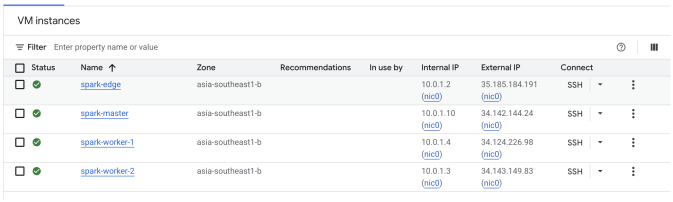


Fig. 1. VM Instances created from Terraform

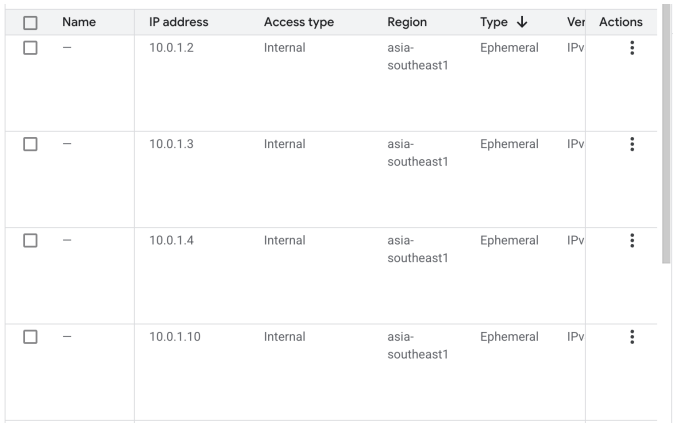


Fig. 2. Internal IP addresses for each VM instances

IV. Result

A. Create VPC and VM instances on Google Cloud Platform

You can see that there are four instances have been added to the compute engine, each has the External IP and Internal IP addresses.

B. VPC network and Firewall rules

A VPC network is also established on GCP and it comes with two firewall rules.

C. Ansible playbook execution

However, because the connection to each node in GCP fails, Ansible was not able to perform instructions as planned. We received the unreachable error, due to the undefined connection

Fig. 3. External IP addresses for each VM instances

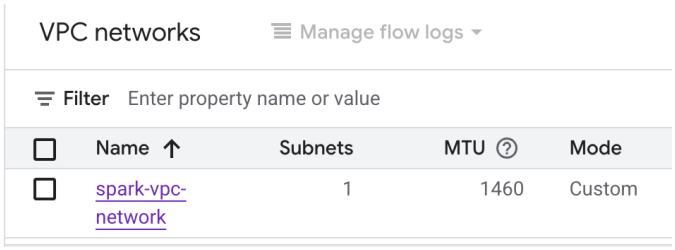


Fig. 4. VPC network created in GCP

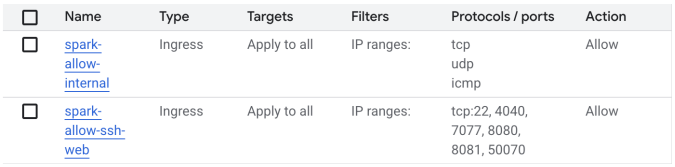


Fig. 5. Firewall rules implemented from the `network.tf` Terraform file

from the local machine, that is blocking the transmission from the local machine, causing the packet loss to be 100%. We also tried different ways to check the connection.

V. Conclusion

Even though we did not managed to fully complete the project as intended. But the terraform code works successfully is a small stepping stones for us to continue to fix the problem. The only issue we remain is the connection to the instances, as we are looking to resolve this issue.

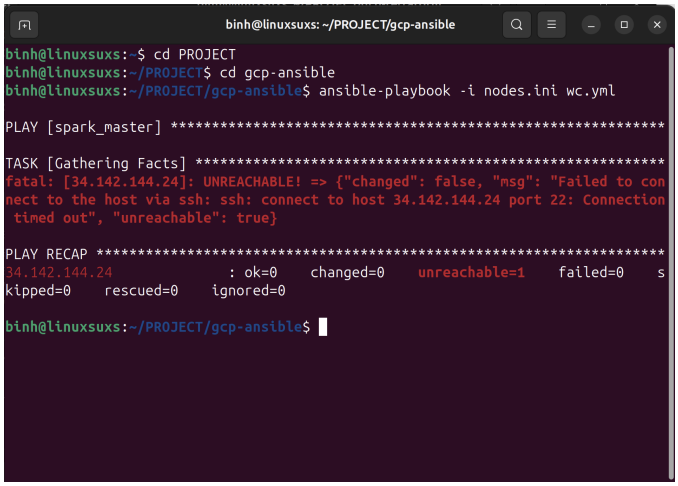


Fig. 6. Failed attempt to execute playbook due to UNREACHABLE error

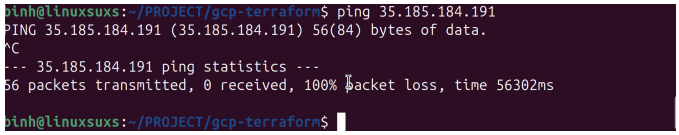


Fig. 7. Testing VM instance transmission, the result was 100% packet loss