TÉCNICO LISBOA

Traffic engineering

2024/2025

Introduction to network virtualization

and softwarization

Part #B

Infrastructure as Code with Terraform

Fernando Mira da Silva

May 2025

1. Introduction

This is the second part of the third TE project, which deals with Terraform, Terraform is a widely used Infrastructure as Code (IaC) tool for provisioning and managing cloud and virtualized IT infrastructures. It plays a key role in VM-based network softwarization scenarios by enabling automated, repeatable deployment of compute and network resources, although it is usually complemented by other tools for full orchestration and lifecycle management.

While Terraform was originally developed for provision of VM and containerized applications in large scale infrastructures, as AWS, Azure, Google Cloud or OpenStack, it can also be used to instantiate simple container in local machines. We will follow this approach to implement and test Terraform to deploy and network a set of containers.

The report must include Parts #A and #B, and it must be delivered until June 15th, Sunday. This first part must be completed in two sessions (9 June, Monday, and 12 June, Thursday).

2. Required components

To complete this project, you need Docker Desktop and Terraform installed, available from https://www.docker.com/ and https://developer.hashicorp.com/terraform, respectively.

3. Setup

The setup involves a simple configuration consisting of a client, a load balancer, and two web servers (web server 1 and web server 2). The client must issue an HTTP request to a default address on the load balancer, which redirects it to one of the web servers in a round-robin manner. The test page provided by each web server must differ for each request, clearly indicating whether it is served by web server 1 or web server 2, and must demonstrate that the load balancer operates in a round-robin fashion.

4. Components

To implement the configuration, use `HAProxy` for the load balancer. The client must support curl for testing HTTP requests (e.g., alpine/curl). Web servers web1 and web2 can be implemented using any HTTP server of your choice.

5. Testing

Configure the **main.tf** file to deploy the overall configuration. Investigate how to integrate in the deployment the web pages.

1. Execute and verify the Terraform commands **`init, plan, apply`** and **`destroy.`**
2. Record and report the IP addresses assigned to each component.
3. Demonstrate that web servers 1 and 2 are running and accessible from the client.

4. Verify that the load balancer correctly distributes requests when accessed from the client.
5. Using port mapping, confirm that the load balancer functions correctly when accessed via a browser from the host.
6. Using port mapping, show that the load balancer is working correctly using a browser from the host.
7. Test the load balancer behavior with a single web server down: stop one of the web servers and check resilience
8. Modify the `main.tf` file to deploy a third web server (web server 3). Re-run `terraform plan` and `apply` without destroying the existing configuration. Review the plan report and verify the final configuration.