Project-Report-Checkpoint_I_Group-20A.md

Project - CheckPoint I README

AGISIT 20201-2022

Authors

Team 20A

Number	Name	Username	Email
ist189399	Afonso Goncalves	https://git.rnl.tecnico.ulisboa.pt/ist189399	mailto:afonso.corte-real.goncalves@tecnico.ulisboa.pt
ist190621	Maria Filipe	https://git.rnl.tecnico.ulisboa.pt/ist190621	mailto:maria.j.d.c.filipe@tecnico.ulisboa.pt
ist189498	Maria Martins	https://git.rnl.tecnico.ulisboa.pt/ist189498	mailto:maria.d.martins@tecnico.ulisboa.pt

Module Leaders

The group unanimously decided to work together in every aspect of this project, since this way, everyone would learn something from every part of it, and at the same pace. The group worked always together, thus everyone worked the same number of hours.

Project Files

Our project contains the following file structure:

```
.
├── ansible
│   └── bootstrap-app.yaml
├── ansible.cfg
├── inventory.tmpl
├── terraform-networks.tf
├── terraform-outputs.tf
├── terraform-provider.tf
├── terraform-servers.tf
└── terraform-variables.tf
```

The `terraform` configuration was split across different files, according to the configuration purpose:

- `terraform-variables.tf` sets up variables that will be used in the rest of the configuration, to simplify future alterations that we may want to do; It also defines the list of services that our project will run;
- `terraform-provider.tf` configures the google provider, by setting the project it will act on, the credentials to do so and the region it will act on;
- `terraform-servers.tf` declares the instances that will be launched and their configuration. It launches one VM per service declared in the `terraform-variables.tf` file.
- `terraform-networks.tf` declares the firewall rules to apply in the generated VPC: By default it blocks every connection, except 1: `icmp` messages, 2: `ssh` from the outside of the VPC to the bastion host, 3: `ssh` from the bastion to any machine insice the VPC. 4:

`HTTP/S` connections to the web machine. This rule is disabled for now since we don't have a web server ready yet.

- `terraform-outputs.tf` automatically generates an ansible inventory file according to `inventory.tmpl` template. Terraform uses the IP addresses created by the provider to populate the inventory file.

The `ansible` directory will contain the playbooks used in this project. Currently, we only have the `bootstrap.yml` playbook, which sets up the launched instances.

Pre-Requesites and Deployment

Pre-Requesites

In order to deploy our infrastructure, there are some steps that need to be completed:

- Install **vagrant**
- Create a **Google Cloud Provider account** and follow these steps:
  i. Enable APIs & Services in the google cloud platform (with the type of API being Compute Engine API)
  ii. Access APIs & Services after they are enable and click on credentials
  iii. Select the default service account that shows up, using the checkbox and then select on the right side Manage Service Account
  iv. After, select Manage Keys in actions and press Create new Key
  v. Choose the type JSON and save your file to the project folder

Deployment

First of all, we need to launch and connect to the `mgmt` VM, from which we will manage the whole infrastructure. From the root directory of the repository, run:

```
user@mypc:~/team20A$ vagrant up
user@mypc:~/team20A$ vagrant ssh mgmt
```

Now that we are connected to the VM, let's go to the project directory:

```
vagrant@mgmt:~$ cd labs/project
```

If it is the first time we are deploying this project, we need to install the provider plugins, so that we can interact with them:

```
vagrant@mgmt:~/labs/project$ terraform init
```

Now that the required plugins are installed, we can build the infrastructure with the command:

```
vagrant@mgmt:~/labs/project$ terraform apply
```

This command will prompt you to confirm that you want to apply these changes, type `yes` and then hit Enter. This command will take some time to run, so go grab a cup of coffee and enjoy the process!

After the infrastructure is created, we need to configure the deployed servers. We can use Ansible to do so, by running:

```
vagrant@mgmt:~/labs/project$ ansible-playbook ansible/bootstrap.yml
```

In the first time we connect to these machines, Ansible will prompt us if we want to add their fingerprint to the `known_hosts` file. We need to type `y` for the operation to succeed

```
 -----------
< TEAM 20 A >
 -----------
        \   ^__^
         \  (oo)\______
            (__)\       )\/
                ||----w
                ||     |
```