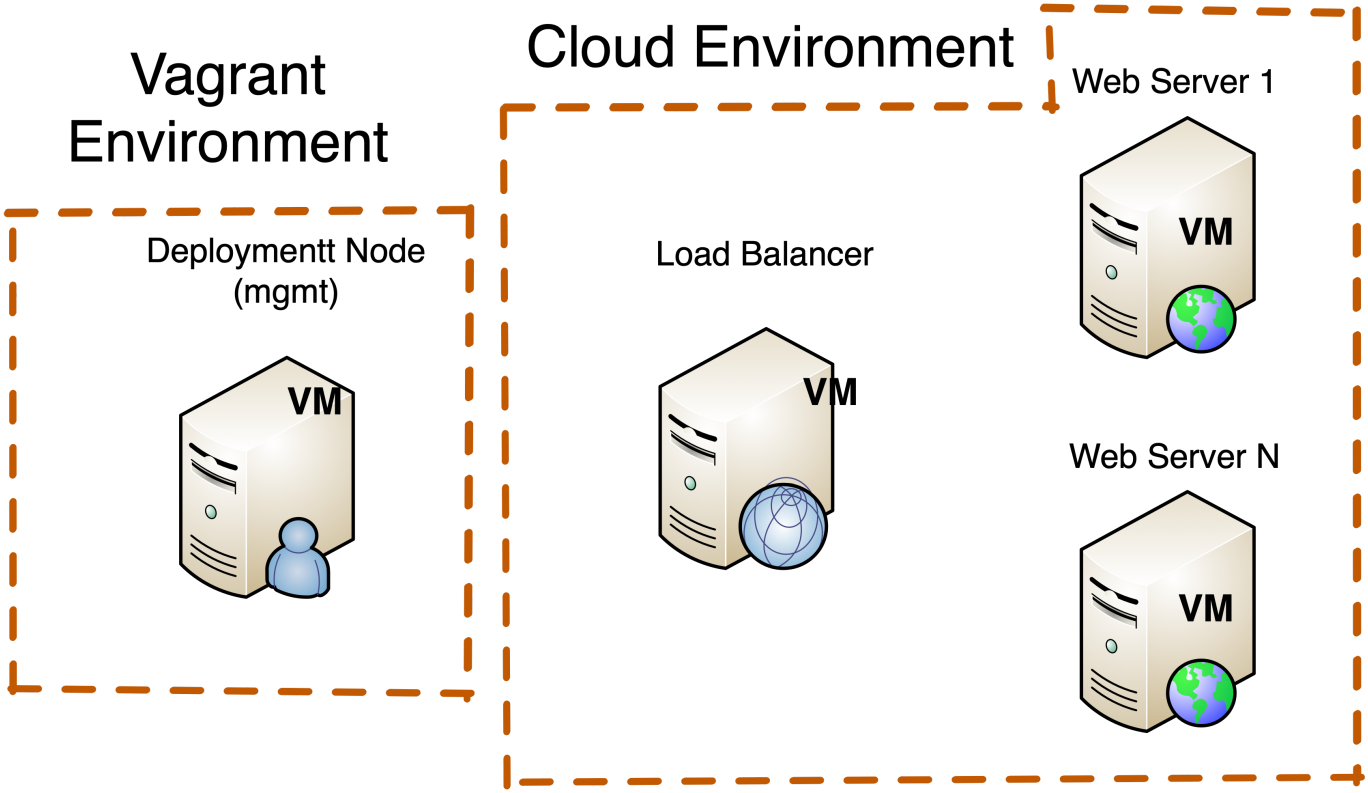
AGISIT Lab Guide 03

Deployment of a cloud-based infrastructure in a Public Cloud (Google Cloud Platform) using automation tools

Objectives

The end result of of this lab experiment will be a fully functioning web infrastructure, composed by several back-end web servers (implemented with **NGINX**) and one front-end Load Balancer (implemented with **haproxy**). We will use **Terraform** to Provision the infrastructure in **Google Cloud Platform** (GCP), together with the **Ansible** Configuration Management tool for configuring the compute instances of the infrastructure, install in those instances all of the required applications and packages, deploy the corresponding customized configuration files, and start the respective services.

In essence, it is the **same solution** used in previous Lab, but now deployed in a Public Cloud.



The Google Cloud SDK

Google Cloud SDK is a set of tools that you can use to manage resources and applications hosted on Google Cloud Platform. These include the **gcloud**, **gsutil**, and **bq** command line tools. A comprehensive guide to the **gcloud CLI** can be found in https://cloud.google.com/sdk/gcloud/.

The Google Cloud SDK is already installed in the `mgmt` node, and so you can use those tools manually for some operations.

To familiarize yourself with the Google Provider in **Terraform**, you can access the Guide at Terraform Google Provider.

Working Methodology

The working Methodology from this lab should be as **Teamwork** for each Group, based on Coordination and/or division of Tasks among members of each Group.

Role Division

It is highly recommended to establish an Agreement among Group members to split “Roles” in the execution the the Lab Works and essentially on the Project, for example:

- **Infrastructure Resources:** one member of the Group is responsible for deploying the “network related” things, which may include a Load Balancer, firewall rules, and addresses and ports in configuration files, etc;
- **Compute Resources:** another member of the Group is responsible for deploying the servers, i.e., configuring all scripts and configuration files for the servers that will be created;
- **Application Configuration:** another member of the Group is responsible for Configuring the servers of the Infrastructure in terms of the applications that will be run in them, including the configurations of those applications;

The local Git repository for this Lab

Each member of the Group in their own computer, opens a Terminal, changes to the "Projects" directory previously created, where the respective Group Repository in the GIT platform (https://git.rnl.tecnico.ulisboa.pt) was cloned (hopefully). The repository of each Group is identified with the format `team-XXC` in which `C` represents the campus (**A** for Alameda, **T** for Taguspark) and `XX` is the number of the Group in Fenix System. For example, Group 3 from Alameda has the repository team-03A.

In that Group Repository you will find a Directory named `labs/gcpcloud` that already contains the files for this Lab experiment.

```
├── README.md
├── Vagrantfile.docker
├── Vagrantfile.vbox
├── bootstrap-mgmt-docker.sh
├── bootstrap-mgmt.sh
└── labs
    ├── arch-arm64
    │   └── README.md
    ├── awscloud
    │   └── README.md
    ├── gcpcloud
    │   ├── README.md
    │   ├── ansible-gcp-servers-setup-all.yml
    │   ├── ansible-load-credentials.sh
    │   ├── ansible.cfg
    │   ├── gcphosts
    │   ├── templates
    │   │   ├── haproxy.cfg.j2
    │   │   └── index.html.j2
    │   ├── terraform-gcp-networks.tf
```

```
│   ├── terraform-gcp-outputs.tf
│   ├── terraform-gcp-provider.tf
│   ├── terraform-gcp-servers.tf
│   └── terraform-gcp-variables.tf
├── k8scloud
│   └── README.md
├── project
│   └── README.md
├── vmcloud
│   └── README.md
└── webfront
```

Create a Google Cloud fully functional Balanced Web Service

> **PLEASE NOTE:** The following instructions and steps **ARE NOT A RECIPE**, just a recommended Procedure. **Your job is to study the Process** to get insight of its purpose, **and modify whatever you will need to modify** in order to ensure that you will arrive to a correctly Provisioned Infrastructure, and a correctly Configured Web Service.

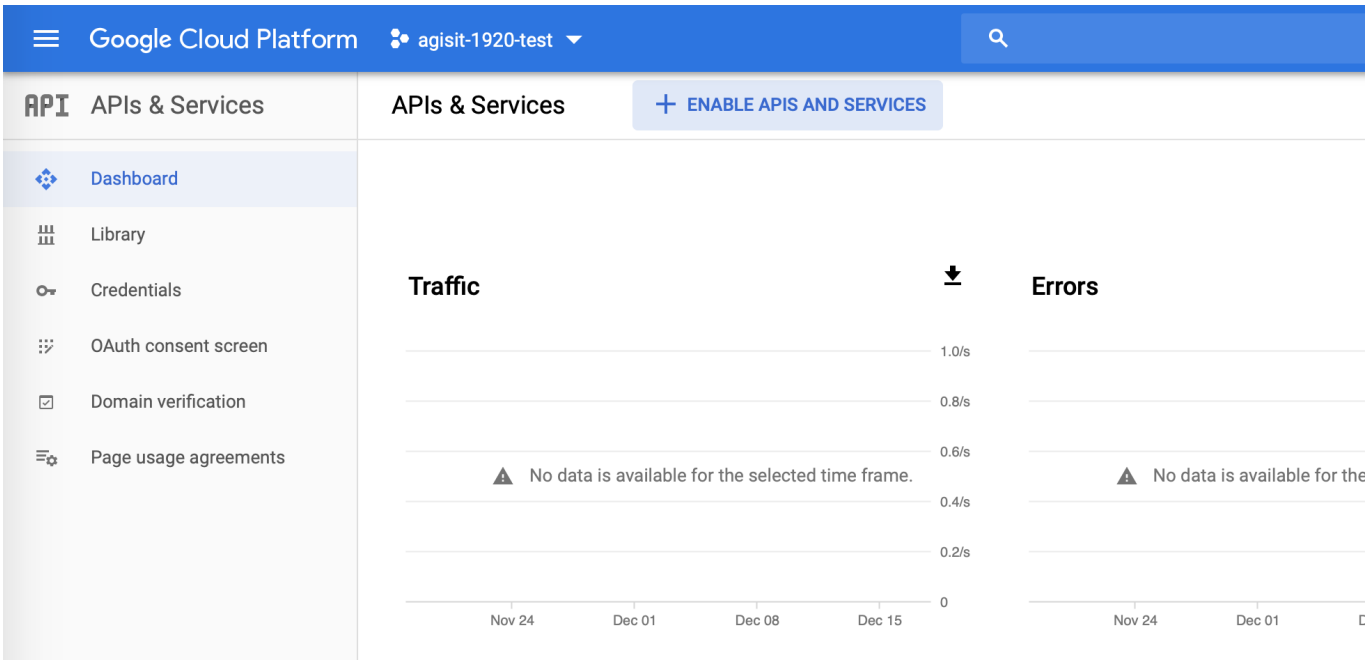
In the local Repository for your Projects, `team-XXC` you will find a `Vagrantfile.(docker/vbox)` and a `bootstrap-mgmt(-docker).sh` that will be used to create and configure the **Management Node** (`mgmt`) that will be used to manage your infrastructures. We will use a Ubuntu OS, named “ubuntu/ focal64”, which corresponds to Ubuntu 20.04 LTS. The `mgmt` local node contains the Tools needed for Provisioning and Configuring the Infrastructures.

The `gcpcloud` folder is where all the definition files for the **GCPCloud** infrastructure reside, allowing you to run, in several steps, the tasks that successively deploy the infrastructure and configure the corresponding compute instances with the software that sets up a **Load Balanced Web site** (in practice, "identical" to the one you have created locally in previous Lab 01, and also in the Private Cloud of Instituto Superior Técnico (VMCloud) in Lab 02.

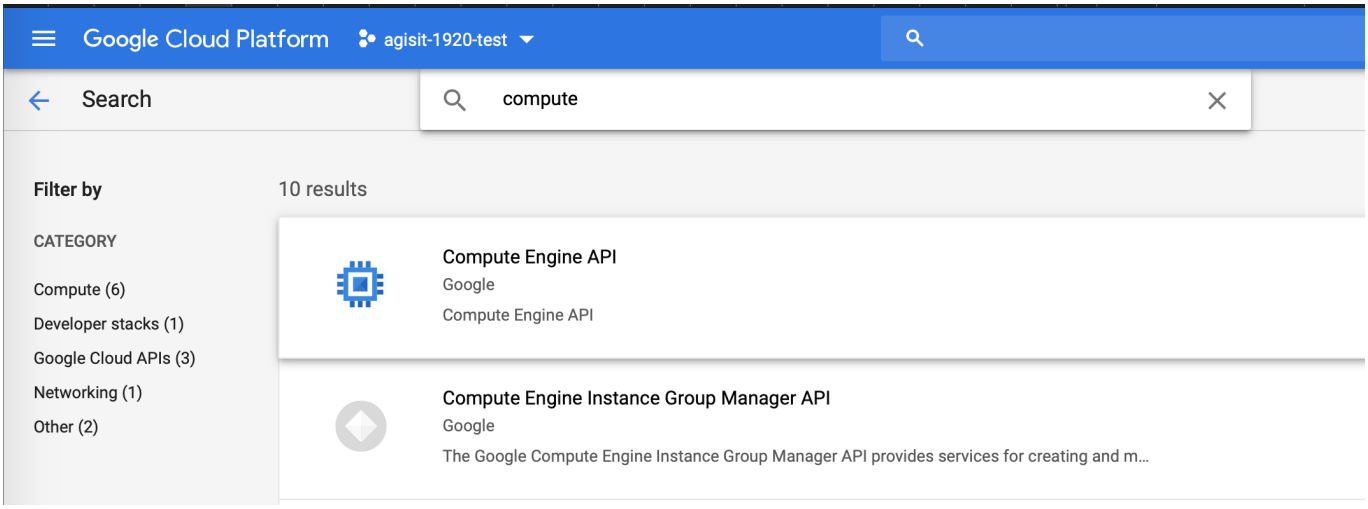
1. Generating the Google Cloud credentials

Similarly to the process you used in Lab 02 for VMCloud, you need to prepare the Security Keys to access the resources in GCP.

For that purpose you need to ENABLE APIs AND SERVICES for your Project, by choosing in the Google Cloud Console **API & services** and next selecting the Dashboard, where you can see a button on the top menu for enabling those services, as illustrated in Figure:

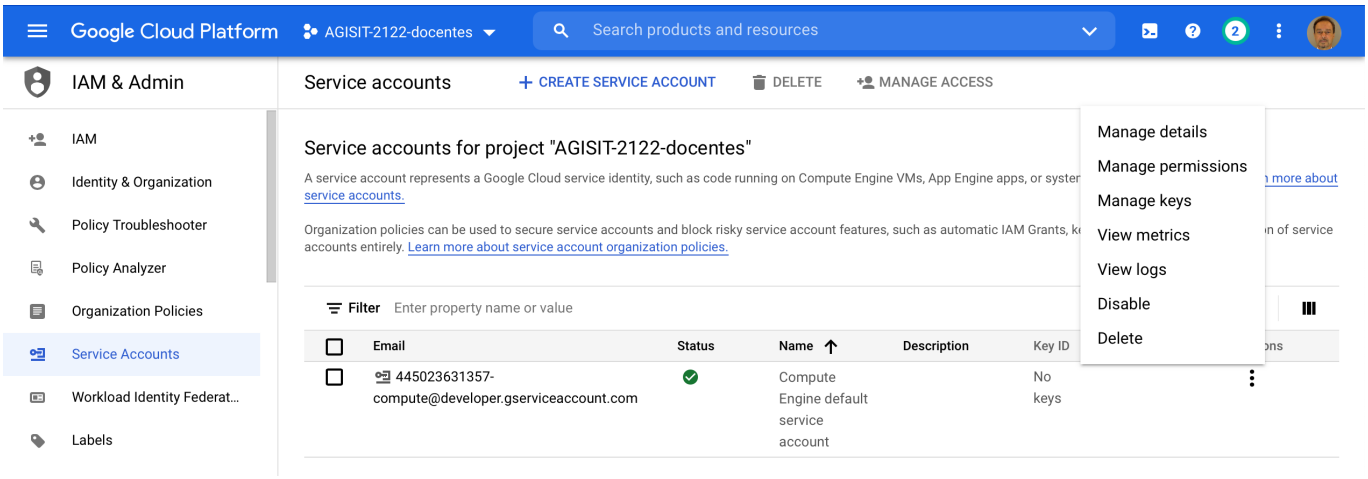


Selecting that button opens a new window for selecting the type of API (as in following Figure). In that window search for **Compute Engine API**, select it and then click ENABLE.

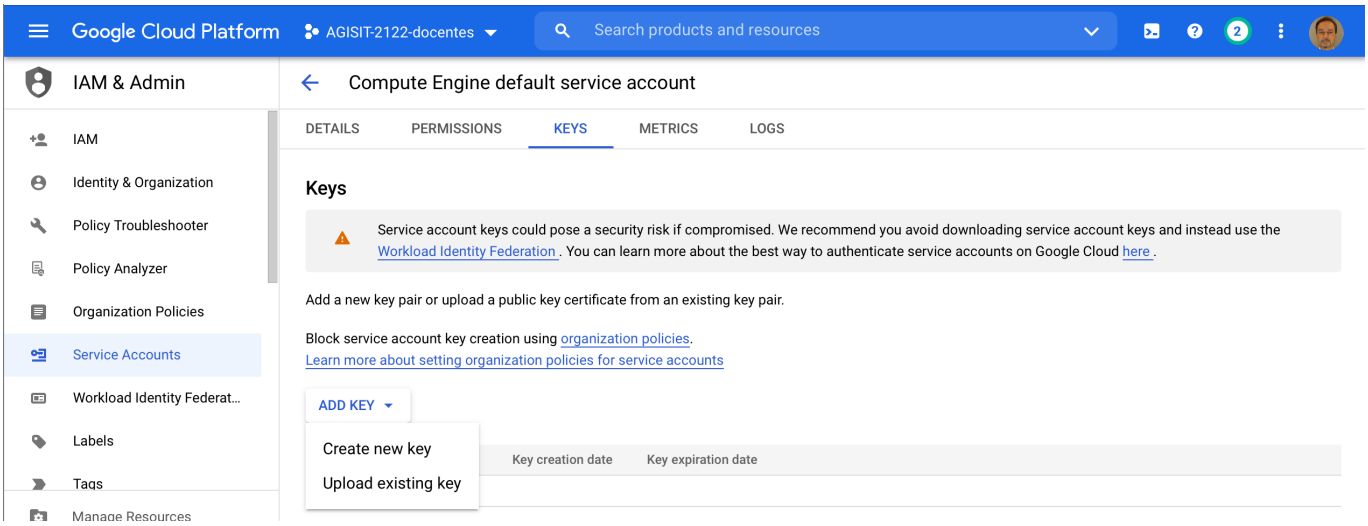


When the API is enabled (it may take some time...) you can then access **API & Services** and next **Credentials**, in order to create the necessary file.

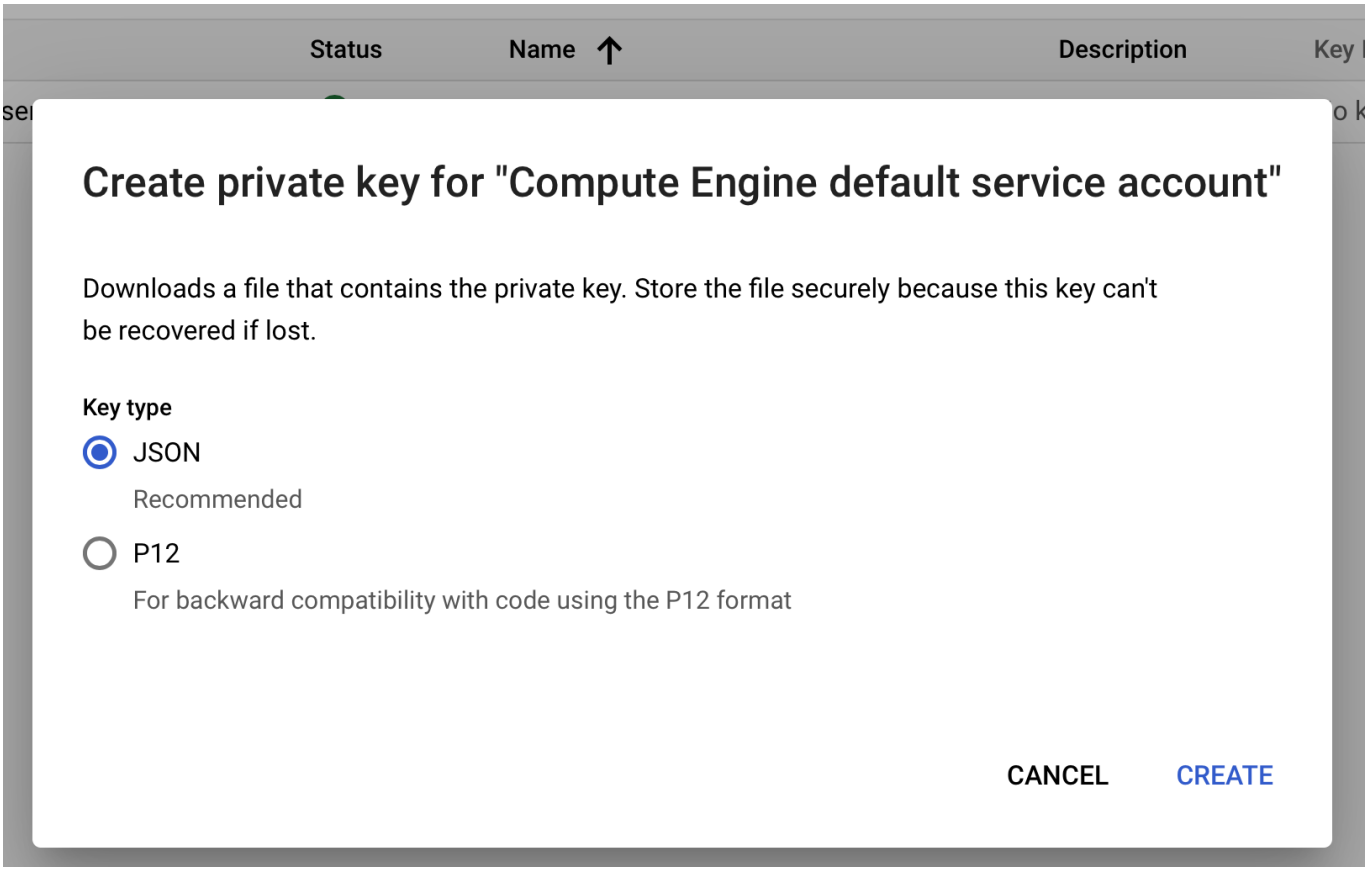
You may then see that a **Service Account** for the Compute Engine default service account is created, and so you need to select the check box of that Service Account and then select on the right side **Manage Service Account**, which will open a new window, as illustrated in Figure:



There, in **Actions**, You select **Manage keys**, that opens a new windows to select either **Create new key** or **Upload existing key**:



You will select **Create new key** which opens a pop-up window in which you select the JSON checkbox, as illustrated in the following Figure, which will download to your computer a **Credentials file**.



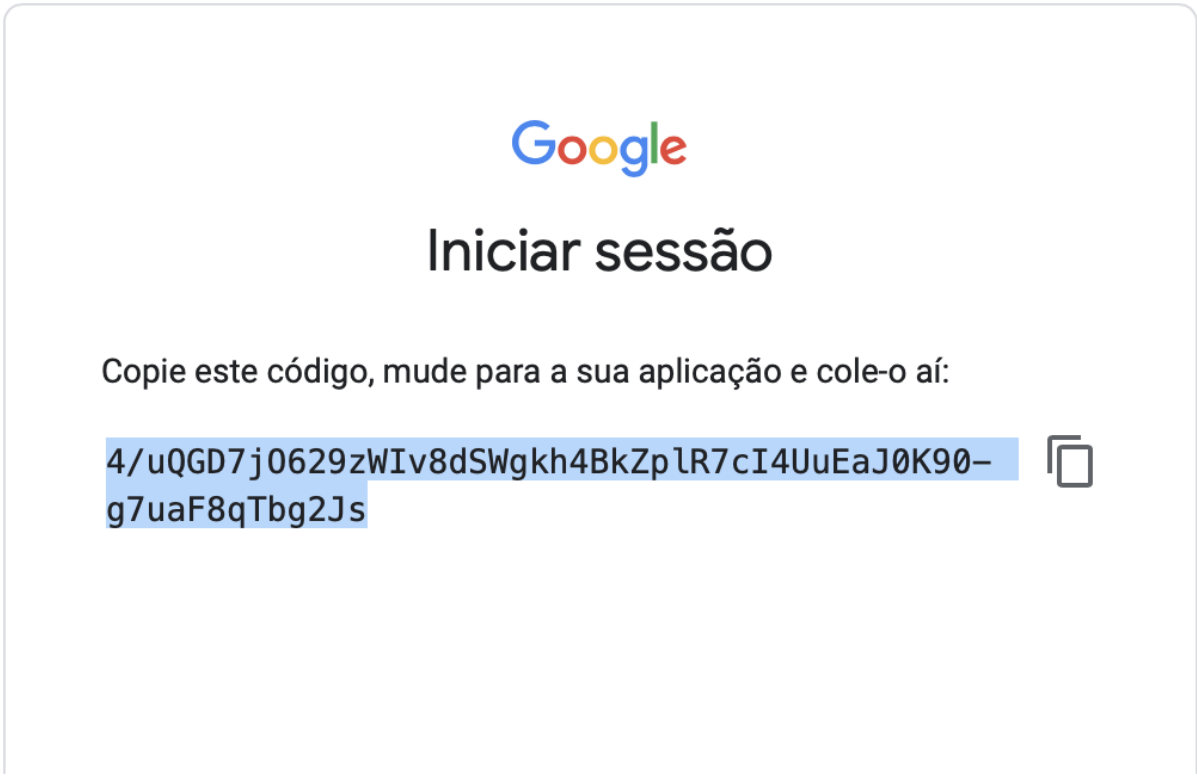
Save the file to the `gcpcloud` folder. The **Credentials file** has your keys to access the GCP service and should be kept safe, and **not shared in the git repository**.

> **Please Note:** You may need to authorize the Google Cloud SDK (client API) in order to allow controlling the GCP resources from interfaces (APIs) rather than the Dashboard. This API can be invoked in the console of your Management VM (`mgmt`) with the command **gcloud**, in the project folder `gcpcloud`. The API must be authorized before using it and so, in that `gcpcloud`, you could run the command, as illustrated hereunder, and paste its output (the whole string starting with `https://`) in a web browser in your host system. You would then be requested to specify your `@tecnico.ulisboa.pt` email address in order to ensure that you can be authenticated to the GCP account. Next, as in the following Figure, you should copy the response code, and go back to the `mgmt` console and paste that code in the field **Enter verification code**:. This authorization is required for tasks such as connecting to the instances via SSH.

```
vagrant@mgmt:~/labs/gcpcloud$ gcloud auth login
Go to the following link in your browser:

https://accounts.google.com/o/oauth2/auth?redirect_uri=urn%3Aietf%3
Awg%3Aoauth%3A2.0%3Aoob&prompt=select_account&response_type=code&
client_id=......www.googleapis.com%2Fauth%2Faccounts.reauth&
access_type=offline

Enter verification code:
```



2. Provisioning the infrastructure

In this step-by-step approach you will be able to create the infrastructure in the GCP. Go to the `gcpcloud` folder, and there you have several `.tf` files.

> **Please note:** All the project files may contain "**dummy**" values for some variables, and so you MUST very carefully edit the relevant values, so that they are adequate to the project (even *file paths* may need to be modified...).

Look into the `terraform-gcp-variables.tf` and `terraform-gcp-provider.tf` files and replace some of the **TAG**s with the field values corresponding to your project, as well as the name of the **Credentials file** from Google (note that you may have already there some dummy value that you need to replace). Check https://www.terraform.io/docs/providers/google/index.html to find more information about how to configure these files.

The `terraform-gcp-variables.tf` defines variables that are used by other files. Edit this file and replace **XXXXX** with the value you gave for the **ID of your project** (the *exact name* of your project in the GCP Console), for example:

```
variable "GCP_PROJECT_NAME" {
    default = "agisit-2021-XXXX"
}
```

You also need to obtain the image names for your instances, to replace them in the `terraform-gcp-servers.tf` file. For that purpose, go to https://cloud.google.com/compute/docs/images and search for the Ubuntu, selecting the most recent Ubuntu (Focal 20.04 LTS). You also should check "machine type", in order to know which would be the "most affordable" for your budget. For that purpose, go to Compute

Engine Pricing and look for **N1 standard machine types**, selecting the **Region**, for example **Frankfurt (europe-west3)**, where you want to make the deployment. You should see there, for this project, that a `n1-standard-1` type is enough. Do not also forget to look at the **Zones** in GCP Zones.

With that step done, you can now initialize **Terraform**, in order to eventually satisfy some plugin requirements:

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

As you can observe by studying the files `terraform-gcp-servers.tf` and `terraform-gcp-networks.tf`, your infrastructure code is written (or, to be rigorous, the infrastructure is declared), by defining the desired resources and their characteristics.

You just need to make it live in the Google Cloud. **Terraform** "talks" to the Google Cloud APIs and makes sure that the planned infrastructure is always up-to-date with what you described in the code of the configuration files.

Before making it live, you are supposed to create a **Plan**, i.e., a process that compares your assumed *current state*, with the one declared in those configuration files, using API calls that fetch the current state from the Google Cloud Platform. Once you are happy with the **Plan** output, you **Apply** it, and if any changes are scheduled, they are actually performed.

So, let's go and create the **Plan** with the following command,resulting in a long list something similar to:

```
vagrant@mgmt:~/labs/gcpcloud$ terraform plan
```

To execute the **Plan** and create the infrastructure, run **Apply**:

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

The output of **Terraform** can be viewed again with the command `output`.

Once the command finishes, **Terraform** will create a new `terraform.tfstate` file. This file keeps the *current state* of the deployed infrastructure, and **it is a 1:1 mapping** of your deployed infrastructure, meaning that if you change/remove resources form the configuration files, they will be removed from your infrastructure and if you add/change resources to the configuration files, when you run the command **Apply** those changes will be reflected to the infrastructure.

Note the last lines of the apply command. They print out the **public IP addresses** of the instances created.

> **Please note:** In case you receive errors similar to the following, then you may just need to select some other GCP Zone. Change that in the `terraform-gcp-variables.tf` and Plan/Apply again.

```

```

```
Error: Error waiting for instance to create:
The zone 'projects/agisit-2122-XXXXX/zones/europe-west3-c' does not have
enough resources available to fulfill the request.
Try a different zone, or try again later.
```

3. Configuring GCP Instances with Ansible

Now that the infrastructure is (hopefully) created and deployed in **Google Cloud Platform**, in order for **Ansible** to access the machines and configure them, there is the need to populate the **INVENTORY** file, in this case named `gcphosts`, present in the same project directory, as well as edit the `mgmt` system `hosts` file with the **IP addresses** and names of the servers (**retrieved from the output of Terraform**).

4. Test Communication with remote Instances

In order to ensure that you can reach the remote instances created and provisioned in **Google Cloud Platform**, you should execute two tests. The first one is just a simple **PING** using ICMP protocol. The other test is an **Ansible** *ad-hoc* "Ping-Pong" command.

As for the following example, execute those tests to all the Instances that you provisioned with **Terraform**.

```
vagrant@mgmt:~/labs/gcpcloud$ ping balancer
vagrant@mgmt:~/labs/gcpcloud$ ansible targets -m ping
```

5. Configuring GCP Instances with Ansible

The **Ansible** *Playbook* for this last step has three *Plays*. The first *Play* contains pre-configuration of the server instances, in order to insert the SSH key for "password less", and to allow remote operation without requiring direct user input.

The second *Play* includes the tasks to install the web server program "NGINX" in all Web nodes and adapt the corresponding configuration files, ensuring in the end that the web service is duly started.

The third *Play* includes the tasks to install the "haproxy" program in the Load Balancer server, as well as the corresponding configuration file, ensuring that the Load Balancer service is duly started.

The Playbook to use for this step is the `ansible-gcp-servers-setup-all.yml`, which is *almost identical* to the one you have already used in previous Lab.

> Due to recent changes in some of the tools used, as well as the Operating System of the Instances, you **MAY** need to modify some lines in the configuration files provided in your Group Repository. Please note also that the **Lab Guide IS NOT A RECIPE** to strictly follow. As already said, **it is of your Responsibility** to ensure that you successfully deploy the infrastructure and the Service (the Web Service). For that to happen, you need to make all the necessary adjustments in configuration files, in Playbooks, etc. **HINT 1:** Files you may need to verify having the adequate "variables" defined or adequate configurations, are:
>
> - The `haproxy.cfg.j2` template file

> - The `index.html.j2` template file
> - The `gcphosts` inventory file
> - The `ansible.cfg` configuration file
> - The `terraform-gcp-variables.tf` file
> - The `terraform-gcp-provider.tf` file
> - The `terraform-gcp-servers.tf`file, to change the server instances desired
> - The `ansible-gcp-servers-setup-all.yml` Playbook, to adapt for adequate "variables", etc.

So, having done some necessary changes, go ahead and play it:

```
vagrant@mgmt:~/labs/gcpcloud$ ansible-playbook ansible-gcp-servers-setup-
all.yml
```

The tasks in the three *Plays* will take some time, so be patient and wait a few minutes for them to complete. Analyze the returned information (quite large) to see what was performed (or not?).

In case you get errors in some tasks, try to figure out what are the causes, then correct the configuration files corresponding to the failed tasks, and play again the *Playbook*. Repeat the procedure until success.

6. Test the Deployed Infrastructure

If everything went smoothly, it means that you have now your Infrastructure created, with the servers running and providing services.

You can now use a web browser in whatever system to observe the Web site, and do the following:

- Open a web browser, write the URL http://ip-of-the-balancer and hit return. You should be connected to the web site, through the Load Balancer, and one of the web servers should have served your request (the name of the web server and its IP address are shown).
- Hit the refresh button on the web browser. **Did something change?** If so, repeat the refresh several times and observe the changes.
- Open a second browser tab and navigate to the statistics page of the Load Balancer at http://ip-of-the-balancer/haproxy?stats.
- Going back to the first tab, Hit the refresh button on the web browser several times, and then go to the second tab and verify the changes in the counters.

7. Finnish the Experiments

To clean the environment (destroy created networks, security groups and instances) run `terraform destroy`. Make sure the infrastructure has been destroyed by checking with `terraform refresh` and checking it manually on the Google Cloud Platform Dashboard.

> **DO NOT FORGET THAT STEP:** If you leave resources running, Google will continue to take `$$ CASH $$` out of your credit.

When finished the experiment, Stop the `mgmt`Virtual Machine and verify the global state of all active Vagrant environments on the system.