TÉCNICO LISBOA

AGISIT 20/21	LAB ASSIGNMENT	Number:	9
Cloud-based Infrastructures		Issue Date:	30 Nov 2020
IaaS (Infrastructure as a Service) - GCPCloud - Part 2		Report Due:	07 Dec 2020
Author: Prof. Rui Cruz		Due Date:	09 Dec 2020

1 Introduction

The objective of the experiments in this Lab assignment series is to continue learning about the deployment of Cloud-based Infrastructure on the **Google Cloud Platform (GCP)**, using as Automation Engine **Terraform**, an *orchestration* tool for building, changing, and versioning infrastructures. In this Lab you will explore deployment of containerized applications on orchestration platforms such as **Kubernetes**. You will use **Google Kubernetes Engine (GKE)**, a fully managed Kubernetes service for deploying, managing, and scaling containerized applications on Google Cloud. You will deploy a separately managed node pool GKE cluster using Terraform, and learn how to interact with Kubernetes by scheduling and exposing an Web Application in the cluster.

2 Google Cloud Platform

The Google Cloud Platform consists of a set of resources, such as Compute products (scalable, high-performance virtual machines, Kubernetes clusters for Container Orchestration, event-driven serverless functions, etc.), Storage and database products (secure object storage, fully-managed MySQL and PostgreSQL database service, etc.), and other Services (cloud load balancing, cloud VPN, Virtual Private Clouds (VPC), managed Spark and Hadoop data processing, etc.), as shown in Figures 1 and 2, that are provided from Google's Data Centers around the globe.

Each Data Center location is in a global region. Regions include Central US, Western Europe, and East Asia. Each region is a collection of zones, which are isolated from each other within the region.

Figure 1: Google Cloud Platform services - 1

AGISIT 20/21	LAB ASSIGNMENT	Number:	9
Cloud-based Infrastructures		Issue Date:	30 Nov 2020
IaaS (Infrastructure as a Service) - GCPCloud - Part 2		Report Due:	07 Dec 2020
Author: Prof. Rui Cruz		Due Date:	09 Dec 2020

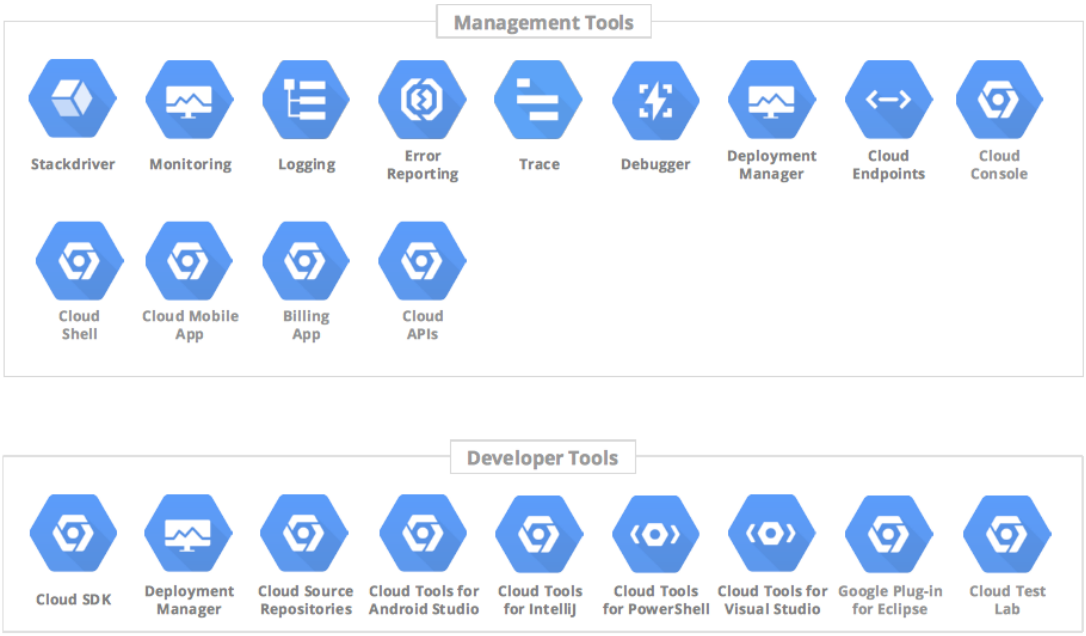

Figure 2: Google Cloud Platform services - 2

2.1 Creating a Project and Exploring the GCP Console

Your first task is to create a Project. For that purpose select on the top Menu Bar the Organization/Projects drop down button, that will open a window for selecting and/or creating a Project, as illustrated in Figure 3.

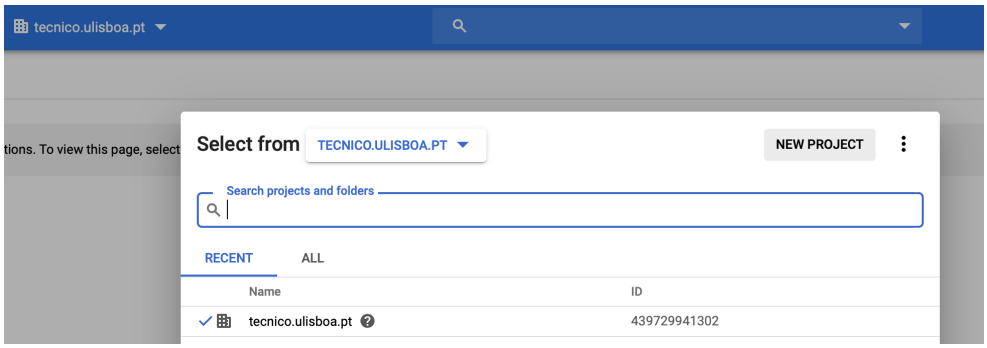

Figure 3: Google Cloud Platform - Create a Project

Select **NEW PROJECT** and give it a name, following the pattern **AGISIT-2021-AAAA-XXXXX**, where **AAAA** can be for example **kube** and **XXXXX** is your IST ID. The system will then take some time (a minute or more) to create the Project. When complete you will enter the Project Console (Dashboard).

AGISIT 20/21	LAB ASSIGNMENT	Number:	9
Cloud-based Infrastructures		Issue Date:	30 Nov 2020
IaaS (Infrastructure as a Service) - GCPCloud - Part 2		Report Due:	07 Dec 2020
Author: Prof. Rui Cruz		Due Date:	09 Dec 2020

Preliminary Notes

One nice feature of the software stack we are going to use is that it is portable to many platforms including **YOUR OWN** personal computers, running the following Operating Systems:

- Microsoft Windows from version 10 up
- Apple macOS from versions 10.13 'High Sierra' up
- Debian-based Linux, such as Ubuntu (recommended) from versions 12.04 'Precise' up.

It is **not recommended** to apply this setup to a virtual machine (**nested virtualization**), although possible, as the configuration requires access to the Hypervisor environment (mandatory Virtualbox) in the host system.

Before proceeding you should verify if you have a "clean" environment, i.e., no Virtual Machine "instances" running (using precious resources in your system), or inconsistent instances in Vagrant and Virtualbox.

For that purpose run the `vagrant global-status` command and observe the results (as in the following example):

```
:~$ vagrant global-status
id       name      provider   state    directory
-------------------------------------------------------------------
28fb48a  mininet   virtualbox poweroff /Users/x/Projects/mininet
f0ccec2  web1      virtualbox running  /Users/x/Projects/multinode
f09c279  web2      virtualbox running  /Users/x/Projects/multinode
```

In the above example, you can observe that there are three Virtual Machines, being the first "mininet", which is **powered off**, but two "web" servers **still running**. It is **advisable to halt those VMs** if running, and then eventually **clean and destroy the VMs** if not needed anymore.

Note: Avoid copying text strings from the examples or configurations illustrated in this document, as pasting them into your system or files may introduce/modify some characters, leading to errors or inadequate results.

AGISIT 20/21	LAB ASSIGNMENT	Number:	9
Cloud-based Infrastructures		Issue Date:	30 Nov 2020
IaaS (Infrastructure as a Service) - GCPCloud - Part 2		Report Due:	07 Dec 2020
Author: Prof. Rui Cruz		Due Date:	09 Dec 2020

3 Submitting the Results of the Experiments

The experiments that you will execute in this LAB will either produce results that you need to report or from which you will be asked questions about the execution of some procedures. In order to report the results you will achieve, proceed as follows:

3.1 General Procedure

On your system, be it Microsoft Windows, Linux or macOS, it is advisable to be working in a **Projects folder not located under the User Profile folder** (i.e., /Users/username/). That folder will contain the **Infrastructure folders** that will be managed by **Git** for versioning (i.e, will contain Repositories). You should also have a **work folder** to contain the necessary files for reporting the results of the experiments. These files may be screenshots, code snippets, text documents, system logs, etc.

It is advisable to give specific and good identifying names to those files, following what will be asked to report. For example, you may be asked to take several screenshots during the procedures, and so, rename those screenshots to the specific items being requested in the assignment. It is also advisable to collect results in a text file or document file while doing the experiment, so that you will have those results when you will need to submit them in the online assignment form.

The procedure for submission is quite simple:

1. This Lab assignment MUST be reported up to the **Report Due** date;
2. In the Moodle (`https://moodle.dei.tecnico.ulisboa.pt/login/index.php`) for the AGISIT course, you will find an **Assignment** for reporting the results from this specific Lab experiment.
3. The Assignment Link opens a **TSUGI CLOUD Web Form** containing the Questions to be answered;
4. The Assignment **Due Date** corresponds to the date it MUST be considered completed.
5. When you are prepared with the requested materials (screen-shots, command line outputs, developed code, etc.) you will submit the items into the respective Exercise Form question boxes of the Assignment;
6. At the end of the Exercise Form you may comment your submission, or respond to feedbacks received (quoting them);
7. When finished answering the Exercise Form, click the button **Done** in the top left of the Form; You will be returned to the Moodle Assignment Link;

8. Please note that this process involves a **Peer Review** mechanism, in which you will be asked to provide **feedback** (anonymously) to the Reports of your classmates. So, after submitting your own assignment, get back to it to see the Reviews that the system have distributed automatically to you;

3.2 Specific Procedure

For this Lab Experiment you will need to submit the following items:

1. Interpret the structure of the folder `gcpkube-tenant` and the role of the different Terraform Configuration files, and submit in the TSUGI Form where asked;
2. Report your Interpretation of the Pods configuration file, in relation to the size of the Cluster (pods, replicas and nodes), and submit in the TSUGI Form where asked:
3. Capture a Screenshot of the Guestbook Web Applicatio, after having inserted some entries (showing the URL of your Cluster), and submit in the TSUGI Form where asked;
4. Capture a Screenshot of your Kubernetes Engine Clusters, and submit in the TSUGI Form where asked;
5. Posts the output of the command “kubectl get pods” in the TSUGI Form where asked;
6. Posts the output of the command “kubectl get service” in the TSUGI Form where asked;
7. Post the output of of the command “gcloud container clusters describe guestbook ...” as indicated in Section 4.4, item 7, in the TSUGI Form where asked;
8. Capture a Screenshot of the **Kubernetes Engine** Workloads page, and submit in the TSUGI Form where asked;
9. Capture a Screenshot of the Load Balancer Statistics for a cluster of 5 webservers, and submit in the TSUGI Form where asked;
10. Capture a Screenshot of the GCP Dashboard ACTIVITY after having completed the assignment (after terraform destroy), and submit in the TSUGI Form where asked;

WARNING. Submissions MUST BE MADE in the TSUGI CLOUD Web Form you can find in Moodle for this course. No other type of submission will be considered.

4 Create a Google Cloud Scalable Kubernetes Cluster

The end result of of this lab experiment will be a fully functioning Kubernetes Cluster, composed by several nodes deployed in different Zones of a Region. We will use **Terraform** to deploy the infrastructure in Google Cloud.

4.1 Step 1: Preparing the Deployment

In a temporary area of your system (Downloads, for example), clone the following repository:

```
https://git.rnl.tecnico.ulisboa.pt/AGISIT-20-21/gcpcloudkube.git
```

Go to the local Repository for your Projects, `projects-team-XX` and create there a **gcpcloudkube** folder, into which you will copy the files of the repository you just cloned (except specific git hidden files/folders), and verify that you have the following file/folder structure in **gcpcloudkube** folder:

```
.
|__ gcpcloudkube
    |__ gcp-gke-provider.tf
    |__ gcp-gke-variables.tf
    |__ gcp_gke
    |   |__ gcp-gke-cluster.tf
    |__ gcp_k8s
    |   |__ k8s-pods.tf
    |   |__ k8s-pods.tf
    |   |__ k8s-services.tf
    |   |__ k8s-variables.tf
    |__ main.tf
```

The **gcploudkube** folder is where all the definition files for the Google Cloud infrastructure reside, allowing you to run the tasks that successively deploy the infrastructure and configure Kubernetes Cluster.

Before continuing, you need to modify the `Vagrantfile` file to add a new shared folder for the Kubernetes Cluster:

```
    if Vagrant::Util::Platform.windows? then
      # Configuration SPECIFIC for Windows 10 hosts
      ....
      osmgmt_config.vm.synced_folder "gcpcloudkube/", "/home/vagrant/
         gcpkube-tenant",
      id: "vagrant-k8s", owner: "vagrant", group: "vagrant",
      mount_options: ["dmode=0755","fmode=0755"]
    else
      ...
      osmgmt_config.vm.synced_folder "gcpcloudkube/", "/home/vagrant/
         gcpkube-tenant",
```

```
      mount_options: ["dmode=0755","fmode=0755"]
    end # of shared folders
```

You can now start your Management node with `vagrant up` or `vagrant reload` if already running, in order to refresh the shares. When ready, confirm that the **osmgmt** VM is reachable by pinging its IP address (as defined in the Vagrantfile). After booting up, open the console to the Management node with `vagrant ssh`.

Due to eventual updating of more recent package versions, after entering the Console, you may find a notice of *** **System restart required** ***, due to some kernel module update. Just exit the Console and do `vagrant reload`.

4.2 Step 2: Generating the Google Cloud credentials

Start by creating the security keys using GCP Console. For that purpose you need to **ENABLE APIS AND SERVICES** for **this new Project**, by choosing **API & services** and next selecting the **Dashboard**, where you can see a button on the top menu for enabling, as illustrated in Figure 4.

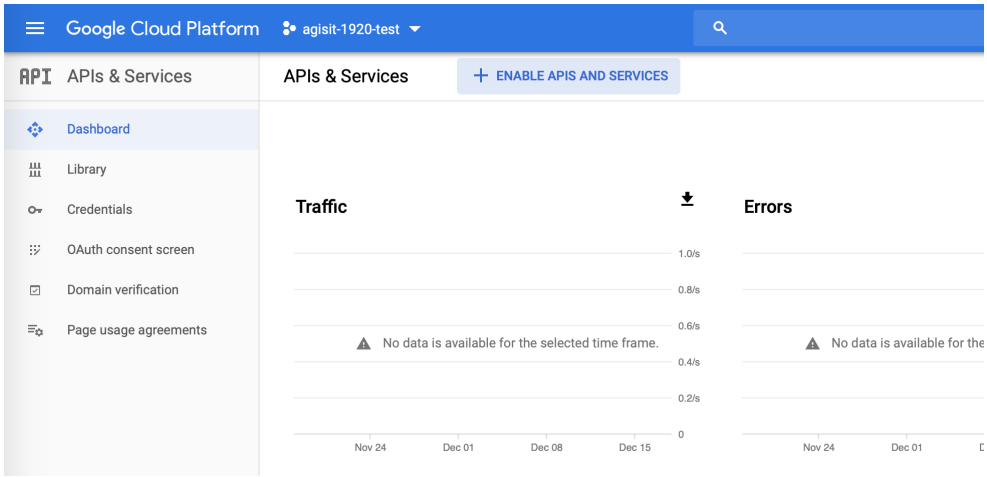

Figure 4: API & Services Dashboard

Selecting that button opens a new window for selecting the type of API and search for Compute Engine API, select it and then click ENABLE.

When the API is enabled (it may take some time. . .), Open the console **Navigation menu** (on the top left of the Menubar) and select the Kubernetes Engine, as illustrated in Figure 5, and wait for the API and related services to be enabled for the selected Project. This can take several minutes. 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**

AGISIT 20/21	LAB ASSIGNMENT	Number:	9
Cloud-based Infrastructures		Issue Date:	30 Nov 2020
IaaS (Infrastructure as a Service) - GCPCloud - Part 2		Report Due:	07 Dec 2020
Author: Prof. Rui Cruz		Due Date:	09 Dec 2020

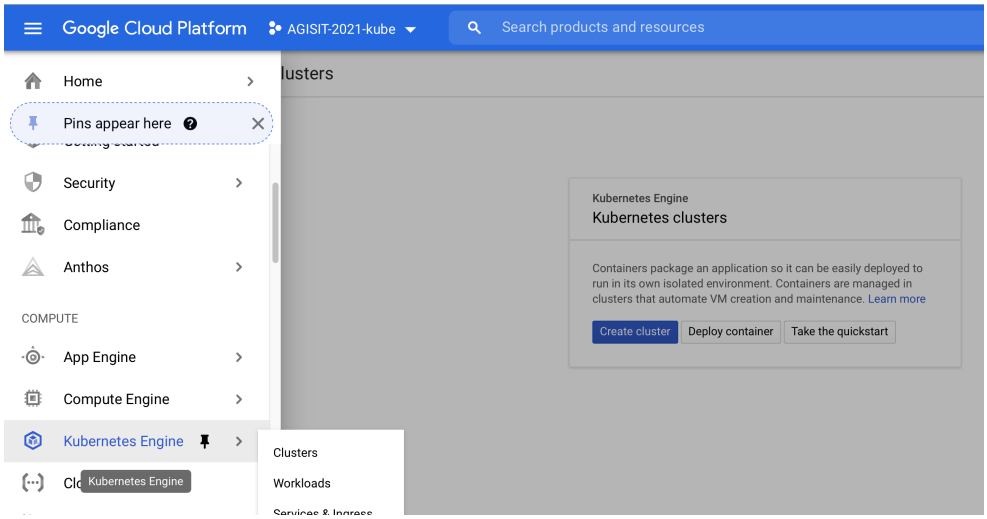

Figure 5: Selecting and Enabling the Kubernetes Engine API

and then select on the right side Manage Service Account, which will open a new window, as illustrated in Figure 6.

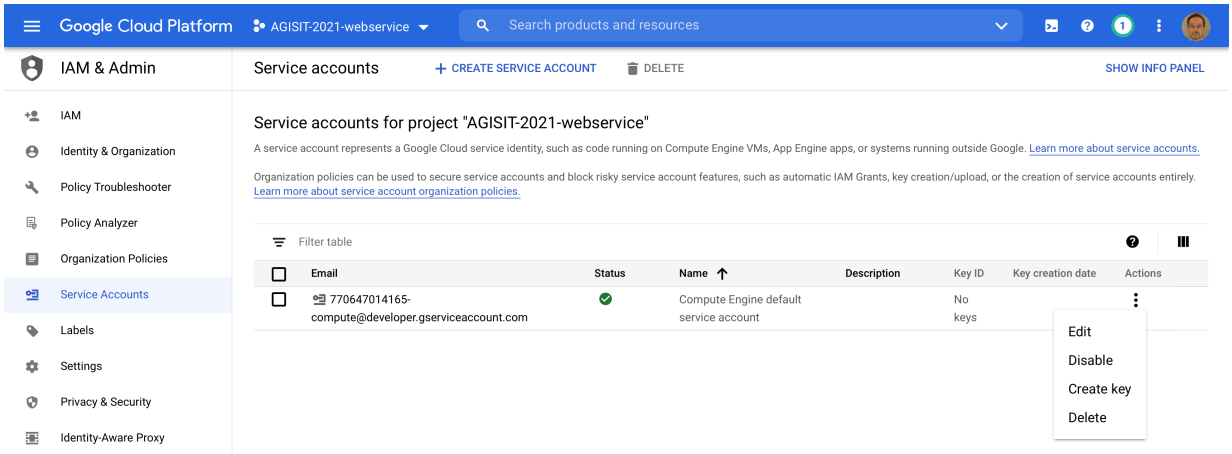

Figure 6: Manage Service Account screen

There, in **Actions**, You select **Create key** and then the **JSON** checkbox, as illustrated in Figure 7, which will download to your computer a Credentials file.

Save the file to the **gcpcloudkube** folder.

The Credentials file has your keys to access the GCP service and should be kept safe, and not shared in a git repository.

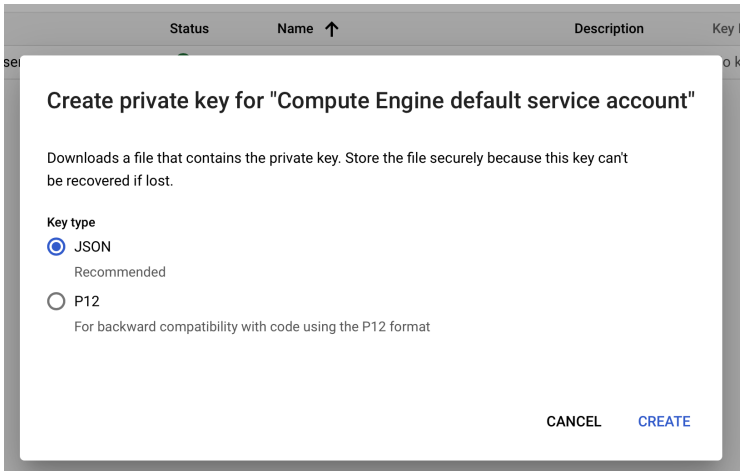

Figure 7: Creating the Service Account Key credentials

4.3 Step 3: Provisioning the infrastructure

In this step-by-step approach you will be able to create the infrastructure in the GCP Public Cloud. Go to the `gcpckube-tenant` folder, and there you have several **.tf** files organized in folders.

You will need to replace in some of the TAGs the field values corresponding to your project, as well as the name of the Credential files from Google. Check `https://www.terraform.io/docs/providers/google/index.html` to find more information about how to configure these files.

Edit the files and replace **XXXXX** with the value you gave for the ID of your project (the exact name of your project in the GCP Console), and also password for the Kubernetes Cluster.

Study all the files in order to understand what is their purpose.

You need first to find the project that Google Cloud SDK is currently pointing to in order to change it. For that do the following:

```
vagrant@osmgmt:~/gcpckube-tenant$ gcloud config get-value project
agisit-2021-xxxxxxx
```

The output may correspond to the previous project you were working on. As such you need to change to the new project, with the following (replace the project-name with your project ID.

```
vagrant@osmgmt:~/gcpckube-tenant$ gcloud config set project new-agisit-
    project-name
```

With that step done, you can now initialize Terraform, in order to eventually satisfy some plugin requirements, and then validate the configuration files:

```
vagrant@osmgmt:~/gcpckube-tenant$ terraform init
```

```
Initializing provider plugins...
- Checking for available provider plugins ...
- Downloading plugin for provider "google"...
...
Terraform has been successfully initialized!

vagrant@osmgmt:~/gcpckube-tenant$ terraform validate
```

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 code in 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@osmgmt:~/gcpcloud-tenant$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
    persisted to local or remote state storage.
------------------------------------------------------------------------
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
....
....

Plan: 6 to add, 0 to change, 0 to destroy.
```

To execute the **Plan** and create the infrastructure, run **terraform apply**, and a result similar to the following will be output:

```
vagrant@osmgmt:~/gcpckube-tenant$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
.....
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

Once the command finishes, **Terraform** will create a new file **terraform.tfstate**. 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, it 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.

Observe the **terraform.tfstate** in order to understand what was created. Search also for the `load_balancer_ingress` IP address, to use in the next step.

4.4 Step 4: Test the created Infrastructure

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

You can now use a web browser in whatever system to observe the Web Application, as the Project Infrastructure is available publicly from the Internet, and do the following:

1. Open a web browser, write the URL `http://ip-of-the-balancer-ingress` and hit return. You should be connected to a simple Guestbook Web Application, a load balanced web service with an external IP.

2. The Guestbook Web Application lets visitors enter text in a log and see the last few logged entries in a database. Tthe Kubernetes cluster is a Redis cluster (`https://redis.io`) deployed in several zones (for resiliency) with a single master (leader) and multiple replicas (followers). Make some entries in the Guestbook, and observe that they are persisted.

3. Open the **Kubernetes Engine** *Clusters* in the Console and take note of the Cluster size.

4. In that page select the *Connect* button, and in the pop-up window, copy the command to configure the **kubectl** command line access.

5. Paste that command in the Management node shell, and verify if the result confirms that the **kubeconfig entry** was generated.

6. In the Management node shell use the command `kubectl get pods` to get information about their status, and the command `kubectl get service` to get information about the Service elements and addresses.

7. In the Management node shell use the following command to get information about the locations of the cluster:

```
$ gcloud container clusters describe guestbook --region europe-west3
    --format='default(locations)'
```

8. In the Console, go to the Kubernetes Engine Clusters and select the Name of the cluster. In the windows that opens, select View GKE Dashboard.

5 Finnishing 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 GCP Dashboard for your Project. Do not forget that **instances deployed and running are billed**, i.e., their usage is accounted **and discounted from you budget!**

When finished the experiment, Stop the Virtual Machine and verify the global state of all active Vagrant environments on the system, issuing the following commands:

```
:~$ vagrant halt
:~$ vagrant global-status
```

Confirm that the status of the VM is “powered off”.