



COLLEGE OF COMPUTING TECHNOLOGY - DUBLIN
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

CLOUD COMPUTING FUNDAMENTALS & PLATFORMS

Assessment 2 – Interacting with a cloud based working environment

Adelo Vieira

Student Number: 2017279

Lecturer: Mr. Michael Weiss

May 8, 2018

Contents

1	Creating a storage bucket. Uploading items from a host computer using the command line and the GUI and place them into the bucket	2
1.1	Using the command line	3
1.2	Using the GUI	3
2	Upload items from the bucket to a Linux virtual machine using the Google CLI	4
3	Creation of a Linux VM, installing Apache and uploading the web page to the web site	5
4	Create a Linux VM, install NGINX and upload the web page to the web site	7
5	Create a Windows VM, install IIS and upload the web page to the web site	9
6	Live Migration of a VirtualBox VM to the GCP	10
7	Explain what Live Migration is and identify situations where DigiTech could benefit from it	12
8	Research topic: Other services available from Google’s Cloud Launcher	13
8.1	Python tutorial:	14
9	Challenging research topic - GCSFUSE: It allows you to mount a bucket to a Debian Linux virtual machine	17
9.1	Installing Cloud Storage FUSE and its dependencies	18
9.2	Mounting a Google Cloud Storage Bucket as a local disk	18
9.3	Change the root directory of an apache server	18
	Declaration	19
	Bibliography	20

List of Figures

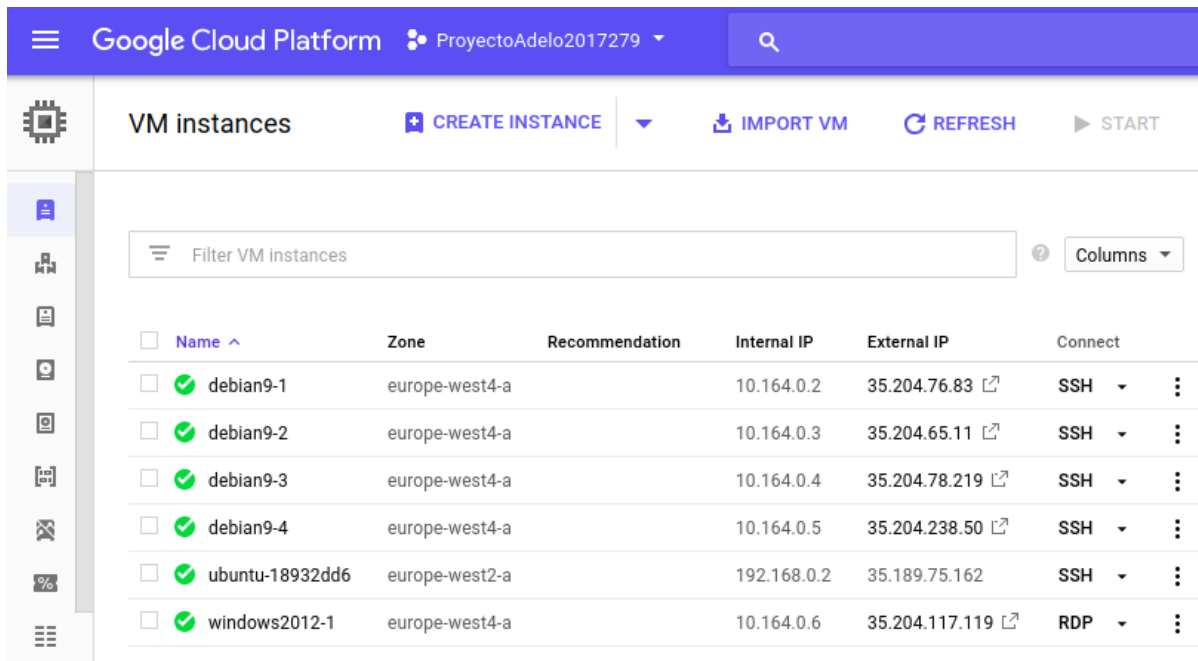
0.1	All the VM Instances we installed in our GCP project with their corresponding IP addresses.	1
1.1	Create a storage bucket	2
1.2	Upload items from a host computer to a bucket using the command line	3
1.3	Upload items from a host computer to a bucket using the GUI	3
2.1	Using the <i>gcloud init</i> command to specify the project (Created on the GCP) to which we want to access (ProyectoAdelo2017279)	4
2.2	Upload items from a bucket to a Linux VM using the Google CLI	5
3.1	Creation of a Linux VM	5
3.2	Installing Apache HTML Server.	6
3.3	Installing PHP.	6

3.4	Coping my Web Page (created on my host computer) to my Linux VM Instance.	6
3.5	Moving the Web Page to the /var/www/html/ directory. This is the DocumentRoot by default. DocumentRoot is the directory where we place our folders and files we want Apache web server to serve up to us on request.	6
3.6	DighTech Web site on Apache (using the port 80)	7
4.1	Installing NGINX HTML Server.	7
4.2	Configuring NGINX to listen to the port 8080	8
4.3	Configuring the firewall rules of our Google Cloud project in order to open the port 8080	8
4.4	DighTech Web site on NGINX (using the port 8080)	9
5.1	Installation of IIS in our Windows VM Instance	9
5.2	DighTech Web site on IIS	10
6.1	Data replication progress on the Cloud Endure Platform (VM Migration Service)	11
6.2	DigiTech on Live Migration VM	12
8.1	Starting the python tutorial in the App Engine page of the GCP	14
8.2	In the GCP we can open a command line console (Google Cloud Shell) where we can develop the application	15
8.3	Exploring the application in the Google Cloud Shell Console	16
8.4	Python App created using the Google Cloud App Engine: proyectoadelo2017279.appspot.com	17
9.1	Change the root directory of an apache server	19
9.2	DigiTech on Apache Loaded from the Attached Bucket	19

Scenario

You are the assistant to the Network Administrator for a networking consultancy company called CompuTech. Your company has recently been providing network consultancy services for DigiTech, a small product services company which is located in a small village on the southern coast of Ireland. The Chief Information Officer has decided that the time is right to migrate DigiTech's on-premise network operation to the Google Cloud Platform. The management at DigiTech would like to get a sample of some of the online business utility services that are available on the Google Cloud.

Here we show all the VM Instances we installed in our GCP project. As they were running, you can see the IP addresses for each instance.



<input type="checkbox"/>	Name ^	Zone	Recommendation	Internal IP	External IP	Connect
<input type="checkbox"/>	✓ debian9-1	europa-west4-a		10.164.0.2	35.204.76.83 ↗	SSH ▾ ⋮
<input type="checkbox"/>	✓ debian9-2	europa-west4-a		10.164.0.3	35.204.65.11 ↗	SSH ▾ ⋮
<input type="checkbox"/>	✓ debian9-3	europa-west4-a		10.164.0.4	35.204.78.219 ↗	SSH ▾ ⋮
<input type="checkbox"/>	✓ debian9-4	europa-west4-a		10.164.0.5	35.204.238.50 ↗	SSH ▾ ⋮
<input type="checkbox"/>	✓ ubuntu-18932dd6	europa-west2-a		192.168.0.2	35.189.75.162	SSH ▾ ⋮
<input type="checkbox"/>	✓ windows2012-1	europa-west4-a		10.164.0.6	35.204.117.119 ↗	RDP ▾ ⋮

Figure 0.1: All the VM Instances we installed in our GCP project with their corresponding IP addresses.

1 Creating a storage bucket. Uploading items from a host computer using the command line and the GUI and place them into the bucket

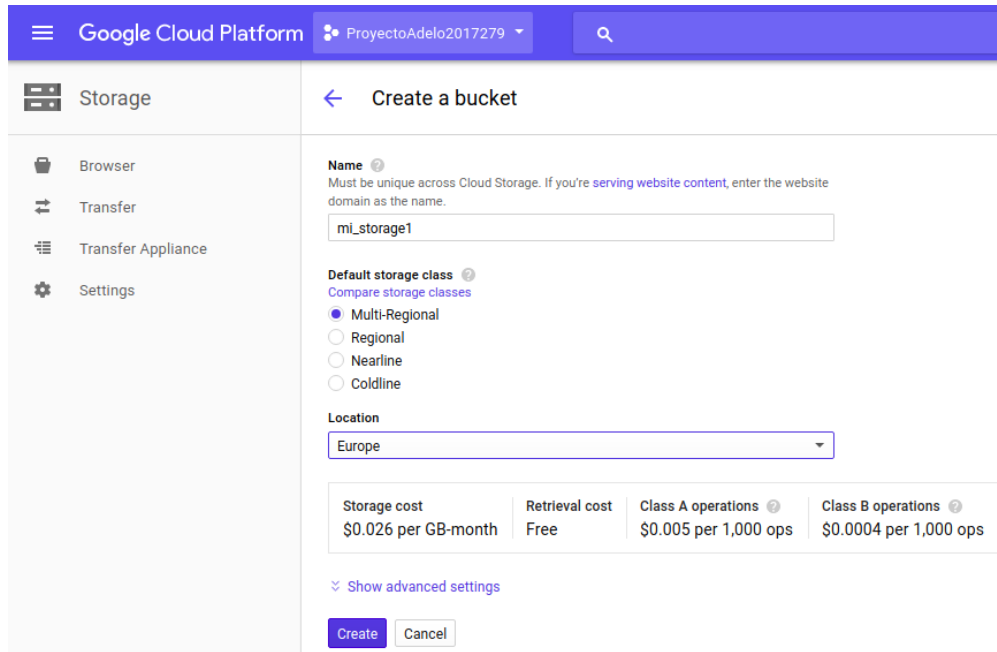


Figure 1.1: Create a storage bucket

1.1 Using the command line

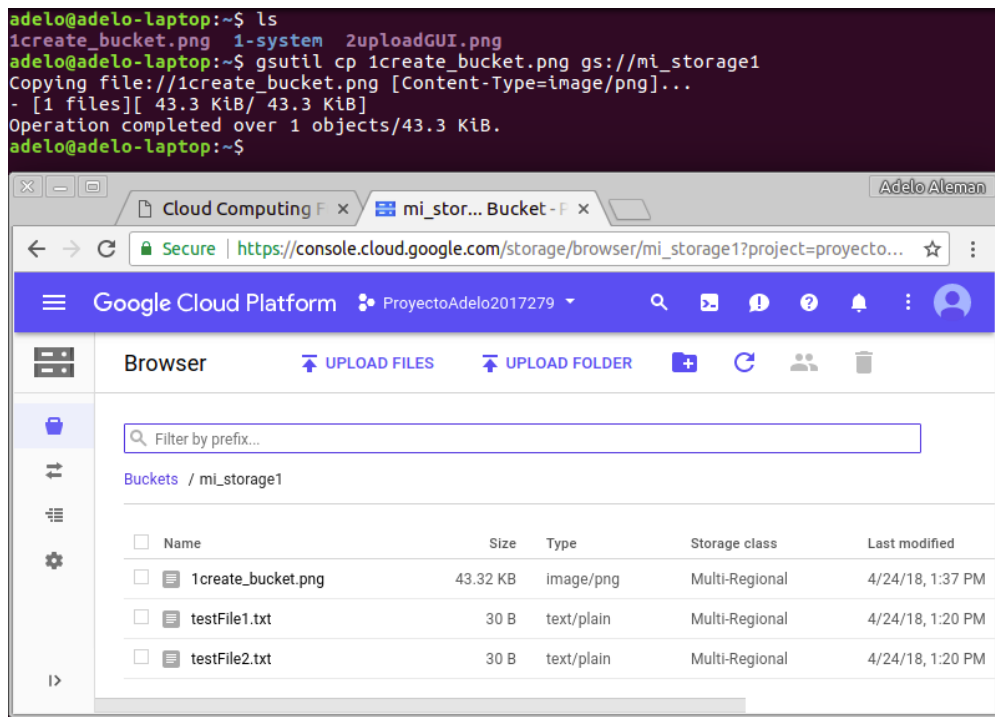


Figure 1.2: Upload items from a host computer to a bucket using the command line

1.2 Using the GUI

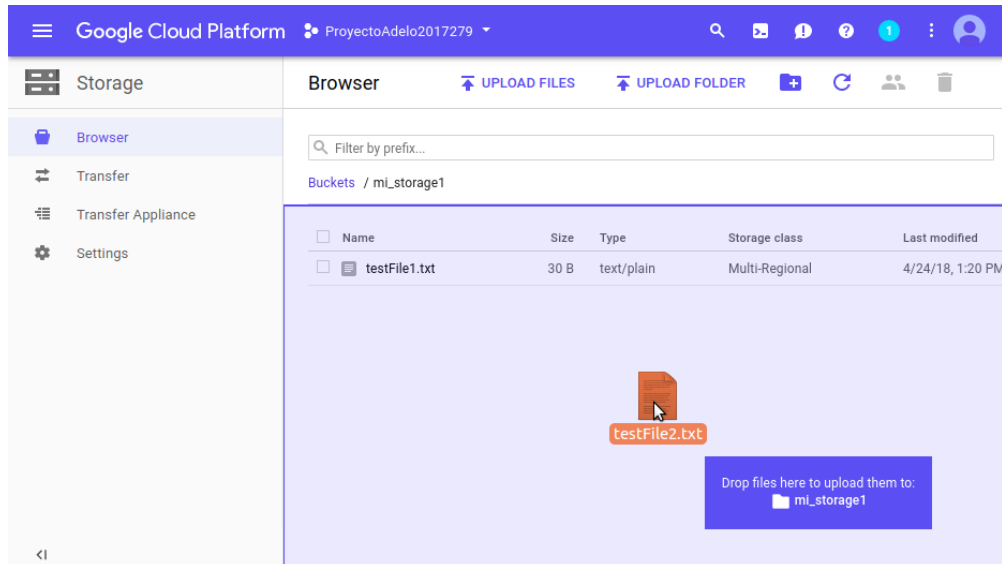


Figure 1.3: Upload items from a host computer to a bucket using the GUI

2 Upload items from the bucket to a Linux virtual machine using the Google CLI

Before you can access your VM Instance from your host computer, you need to initialize the SDK. The *gcloud init* command allows to perform several common SDK setup tasks. Now, we need to use this command to specify the project (created on the GCP) to which we want to access (ProyectoAdelo2017279). Only after that we can access the VM Instances of such project. In Figure 2.1 we show the configuration performed through *gcloud*. Then, in Figure 2.2 is shown how to connect to our Linux VM Instance and upload items from our bucket (*mi_storage1*) to our Linux VM Instance (*debian9-1*).

```
adelo@adelo-laptop:~$ gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [default] are:
core:
  account: adeloaleman@gmail.com
  disable_usage_reporting: 'True'
  project: proyectopiloto-201211

Pick configuration to use:
  [1] Re-initialize this configuration [default] with new settings
  [2] Create a new configuration
Please enter your numeric choice:
Please enter a value between 1 and 2: 1

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic (1/1 checks) passed.

Choose the account you would like to use to perform operations for
this configuration:
  [1] adeloaleman@gmail.com
  [2] Log in with a new account
Please enter your numeric choice: 1

You are logged in as: [adeloaleman@gmail.com].

Pick cloud project to use:
  [1] proyectoadelo2017279
  [2] proyectopiloto-201211
  [3] united-park-196513
  [4] Create a new project
Please enter numeric choice or text value (must exactly match list
item): 1
```

Figure 2.1: Using the *gcloud init* command to specify the project (Created on the GCP) to which we want to access (ProyectoAdelo2017279)

```
adelo@adelo-laptop:~$ gcloud compute ssh debian9-1
Linux debian9-1 4.9.0-6-amd64 #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Apr 24 13:29:36 2018 from 109.255.169.118
adelo@debian9-1:~$ ls
adelo@debian9-1:~$ gsutil cp gs://mi_storage1/testFile1.txt .
Copying gs://mi_storage1/testFile1.txt...
/ [1 files][ 30.0 B/ 30.0 B]
Operation completed over 1 objects/30.0 B.
adelo@debian9-1:~$ ls
testFile1.txt
adelo@debian9-1:~$
```

Figure 2.2: Upload items from a bucket to a Linux VM using the Google CLI

3 Creation of a Linux VM, installing Apache and uploading the web page to the web site

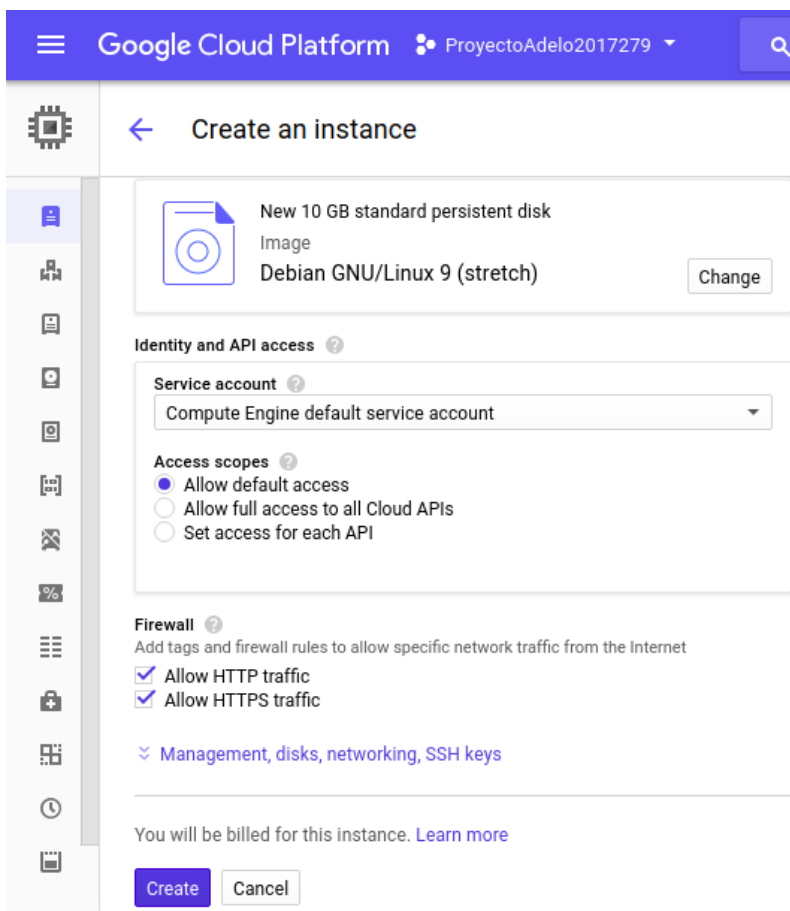


Figure 3.1: Creation of a Linux VM


```

adelo@debian9-1:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libicu57 liblua5.2-0 libperl5.24
  libxml2 perl perl-modules-5.24 rename sgml-base ssl-cert xml-core
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl
  make sgml-base-doc openssl-blacklist debhelper
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libicu57 liblua5.2-0
  libperl5.24 libxml2 perl perl-modules-5.24 rename sgml-base ssl-cert xml-core
0 upgraded, 18 newly installed, 0 to remove and 1 not upgraded.
Need to get 17.3 MB of archives.
After this operation, 80.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Figure 3.2: Installing Apache HTML Server.

```

adelo@debian9-1:~$ sudo apt install php
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.0 php-common php7.0 php7.0-cli php7.0-common php7.0-json php7.0-opcache php7.0-readline psmisc
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php7.0 php-common php7.0 php7.0-cli php7.0-common php7.0-json php7.0-opcache php7.0-readline psmisc
0 upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 3688 kB of archives.
After this operation, 14.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

Figure 3.3: Installing PHP.

```

adelo@adelo-laptop:/var/www/html/webdevelopment$ gcloud compute scp --recurse DigiTech debian9-1:
footer.php          100% 296      0.3KB/s  00:00
style.css           100% 1417     1.4KB/s  00:00
style2.css          100% 128      0.1KB/s  00:00
menu.css            100% 1160     1.1KB/s  00:00
menu2.css           100% 1287     1.3KB/s  00:00
index.php           100% 636      0.6KB/s  00:00
header.php          100% 338      0.3KB/s  00:00
adelo@adelo-laptop:/var/www/html/webdevelopment$

```

Figure 3.4: Copying my Web Page (created on my host computer) to my Linux VM Instance.

```

adelo@debian9-1:~$ ls
testFile1.txt
adelo@debian9-1:~$ ls
DigiTech testFile1.txt
adelo@debian9-1:~$ sudo cp -r DigiTech/ /var/www/html/
adelo@debian9-1:~$ cd /var/www/html/
adelo@debian9-1:/var/www/html$ ls
DigiTech index.html
adelo@debian9-1:/var/www/html$

```

Figure 3.5: Moving the Web Page to the /var/www/html/ directory. This is the DocumentRoot by default. DocumentRoot is the directory where we place our folders and files we want Apache web server to serve up to us on request.

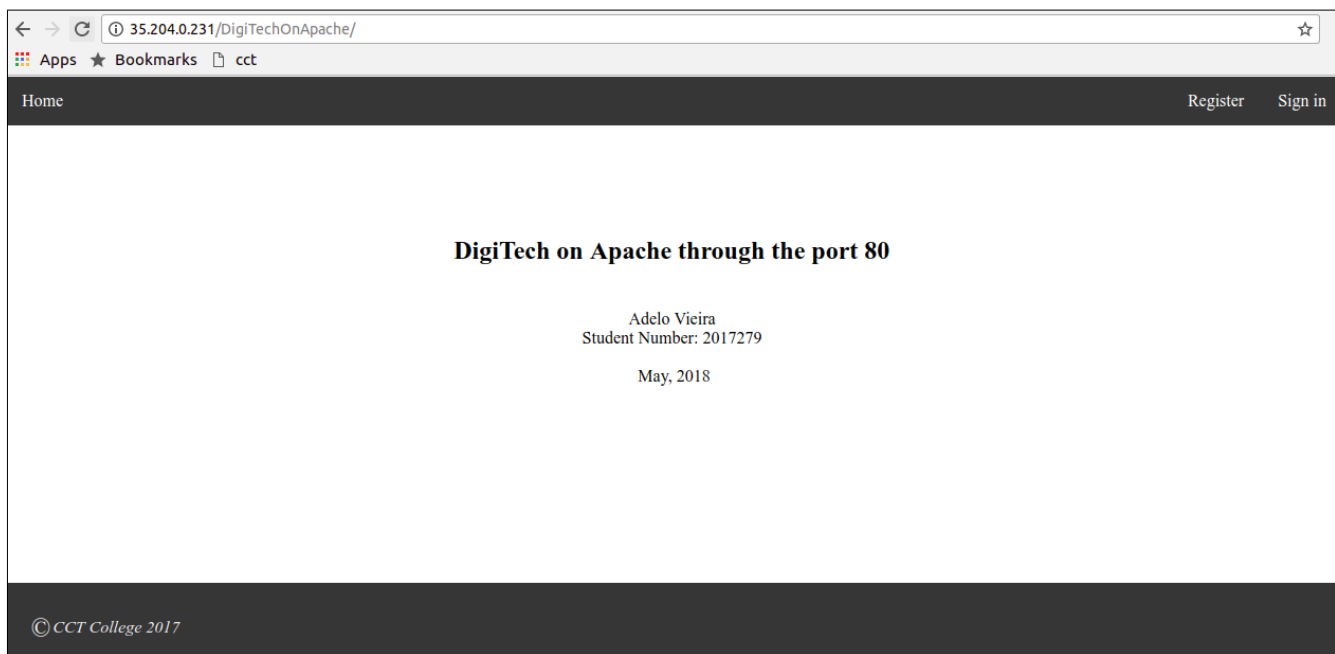


Figure 3.6: DighTech Web site on Apache (using the port 80)

4 Create a Linux VM, install NGINX and upload the web page to the web site

The web page should say “DigiTech on NGINX” and have your name on it.

We considered a better approach to this exercise was to install both servers (Apache and NGINX) in the same Linux VM Instance. In order to do so, after installing NGINX Web Server (Figure 4.1)) we had to configure NGINX to listen to a port other than the default port 80. This because the port 80 was already being using by Apache. In Figure 4.2 we show how we configured NGINX to listen to the port 8080. We just had to edit the configuration file as shown in the such Figure.

We also had to configure the firewall rules of our Google Cloud project in order to open this specific port (8080). In Figure 4.3 is shown the configuration made on the GCP.

Finally, in Figure 4.4 we show the DighTech Web site on NGINX (using the port 8080).

```
adelo@debian9-1:~$ sudo apt-get install nginx
```

Figure 4.1: Installing NGINX HTML Server.

```

adelo@debian9-1:~$ sudo vi /etc/nginx/sites-available/default
###
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
###

# Default server configuration
#
server {
    listen 8080 default_server;
    listen [::]:8080 default_server;
}

```

Figure 4.2: Configuring NGINX to listen to the port 8080

The screenshot shows the Google Cloud Platform console interface for creating a firewall rule. The left sidebar contains navigation options: VPC networks, External IP addresses, Firewall rules (selected), Routes, VPC network peering, and Shared VPC. The main content area is titled 'Create a firewall rule' and includes the following configuration fields:

- Action on match:** Radio buttons for 'Allow' (selected) and 'Deny'.
- Targets:** A dropdown menu set to 'All instances in the network'.
- Source filter:** A dropdown menu set to 'IP ranges'.
- Source IP ranges:** A text input field containing '0.0.0.0/0'.
- Second source filter:** A dropdown menu set to 'None'.
- Protocols and ports:** Radio buttons for 'Allow all' and 'Specified protocols and ports' (selected). Below it, a text input field contains 'tcp:8080'.

At the bottom of the configuration area, there is a 'Disable rule' link, 'Create' and 'Cancel' buttons, and a link for 'Equivalent REST or command line'.

Figure 4.3: Configuring the firewall rules of our Google Cloud project in order to open the port 8080

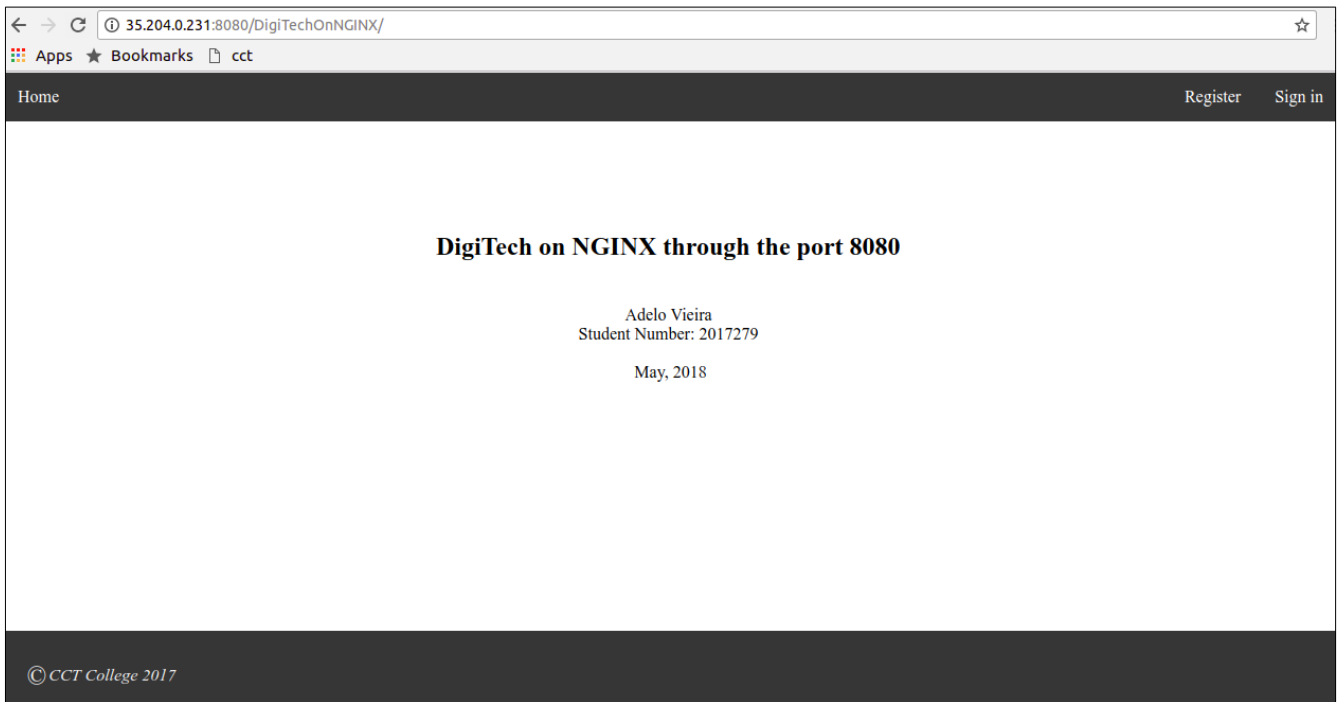


Figure 4.4: DighTech Web site on NGINX (using the port 8080)

- 5 Create a Windows VM, install IIS and upload the web page to the web site

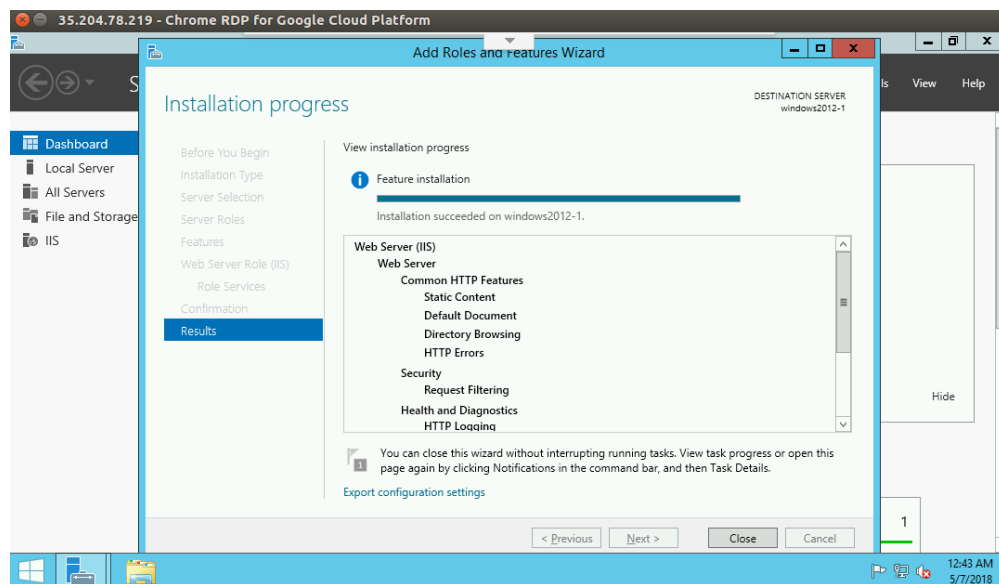


Figure 5.1: Installation of IIS in our Windows VM Instance

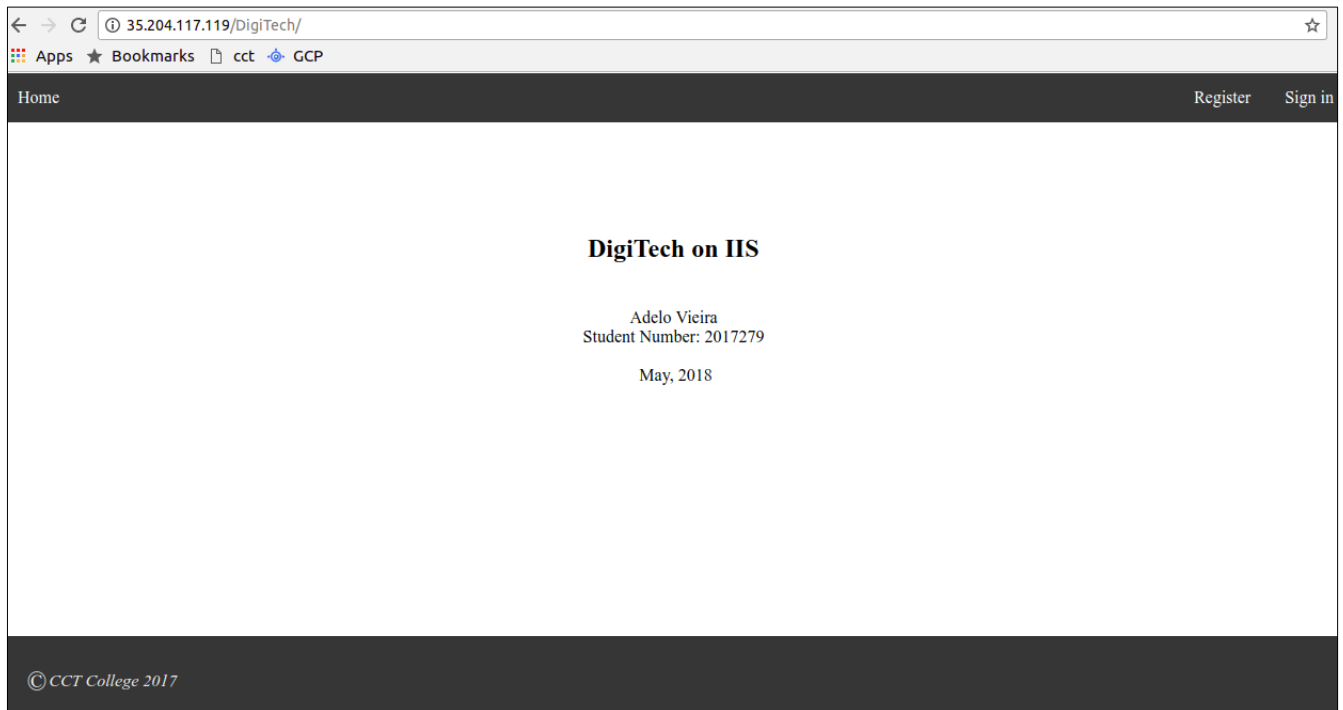


Figure 5.2: DighTech Web site on IIS

6 Live Migration of a VirtualBox VM to the GCP

1. Installing a VirtualBox VM

- We created an Ubuntu Server 17.04 Virtual Machine on VirtualBox.
- We installed Apache, PHP, python2.7 and gcc¹,
- Then, we set up the DigiTech web site.

2. In the GCP Main Menu > Compute engine > VM Instances: Import VM

There you will be redirected to the Cloud Endure Platform (VM Migration Service)

3. In the Cloud Endure Platform (VM Migration Service)

- GCP credentials
We generated a Google Cloud Platform JSON private key (JSON file)
- Replication settings
 - Live migration target: Google EU West 2 (London)
- Install the CloudEndure Agent on your Source machine:

¹Python and gcc was required to install the CloudEndure Agent on our VirtualBox VM

- In the Cloud Endure Platform, was generated our Agent installation token: 0E65-51C9-50EE-E20E-A583-9709 ...
- In our local VirtualBox VM (Ubuntu Server):

```
wget -O ./installer_linux.py https://gcp.cloudendure.com/installer_linux.py

sudo python ./installer_linux.py -t 0E65-51C9-50EE-E20E-A583-9709-1653-BA57-457D-92BF-57B9-0DE2-22DC --no-prompt

installer_win.exe -t 0E65-51C9-50EE-E20E-A583-9709-1653-BA57-457D-92BF-57B9-0DE2-22DC-EA11-3B80-81E6 --no-prompt
```

4. Data replication

- Data replication begins automatically once the installation of the CloudEndure Agent is completed. We were able to see the progress on the Cloud Endure Platform (VM Migration Service) (Figure 6.1)

5. Launch target Machine

It creates (or launches) your final target machines.

After completing all the steps listed above, our website was working as expected (Figure 6.2)

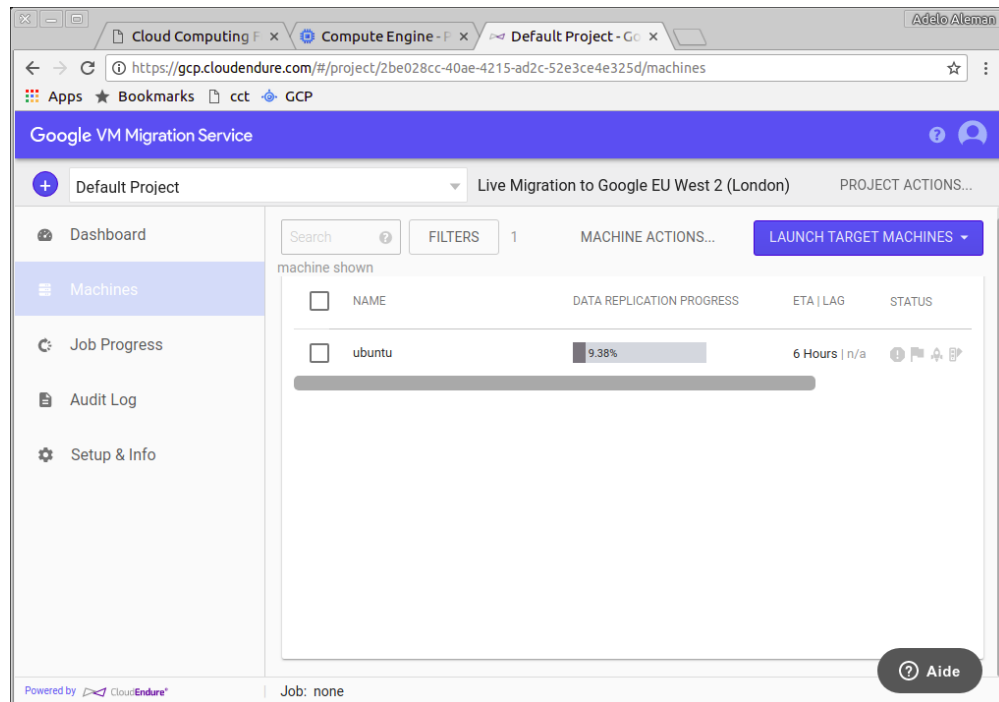


Figure 6.1: Data replication progress on the Cloud Endure Platform (VM Migration Service)

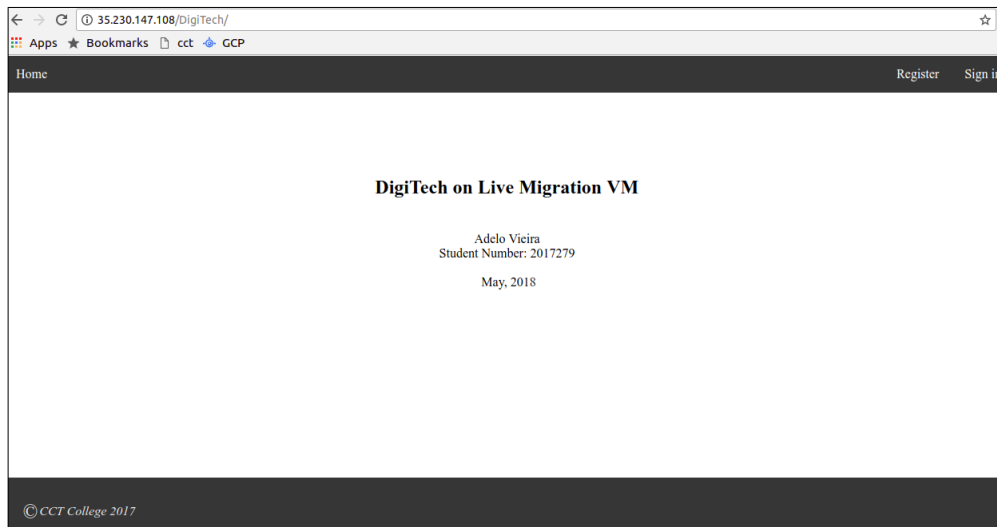


Figure 6.2: DigiTech on Live Migration VM

7 Explain what Live Migration is and identify situations where DigiTech could benefit from it

Explain situations where DigiTech could benefit from being able to perform live migrations of one or more of their production virtual machines.

Live migration offers the possibility to have a backup of your VM's that keeps running even when your host system fails. [cloud.google.com]

Also, live migration is a very convenient solution not only when the system fails. If you need to perform some maintenance task (a software or hardware update for example), you can keep your system running.

A better explanation of situations where you can benefit of live migration is available at cloud.google.com. Here we make a quote from this source:

- «Regular infrastructure maintenance and upgrades.
- Network and power grid maintenance in the data centers.
- Failed hardware such as memory, CPU, network interface cards, disks, power, and so on. This is done on a best-effort basis; if a hardware fails completely or otherwise prevents live migration, the VM crashes and restarts automatically and a hostError is logged.
- Host OS and BIOS upgrades.

- Security-related updates, with the need to respond quickly.
- System configuration changes, including changing the size of the host root partition, for storage of the host image and packages»cloud.google.com

8 Research topic: Other services available from Google's Cloud Launcher

As I'm interested in Web development, I have chose the *App Engine* service.

Google App Engine is a web framework for developing and hosting web applications in Google-managed data centers. [wikipedia.org]

The Google App Engine is a technology that provides you everything tout need to develop from the cloud.

When using App Engine, you can build your applications to run on top of Google's world-class infrastructure you don't have to worry about: [youtube.com]

- Database administration
- Server configuration
- sharing load balancing
- Cloud Storage
- Big data

One of the most important features of Google App Engine, is that it offers automatic scaling for web applications. That is, if the number of request increases, App Engine automatically allocates more resources for the web application. [wikipedia.org] [youtube.com]

When developing for App Engine you can use popular languages such as *Python*, *Java*, *PHP* and *GO*, as well as existing frameworks like *Django* and *flask*.

To get starting using the App Engine:

- Main Menu > App Engine > Dashboard:
- Your first app

- Select a language
- To test the App Engine we selected *Python*: Start new tutorial (Figure 8.1)

8.1 Python tutorial:

This tutorial shows how to deploy a sample Python application to Google App Engine using the *gcloud* command. [cloud.google.com (b)]

In this tutorial:

- We will build and run your “Hello, world!” app: You will learn how to run your app using Google Cloud Shell, right in your browser. At the end you’ll deploy your app to the web using the *gcloud* command.

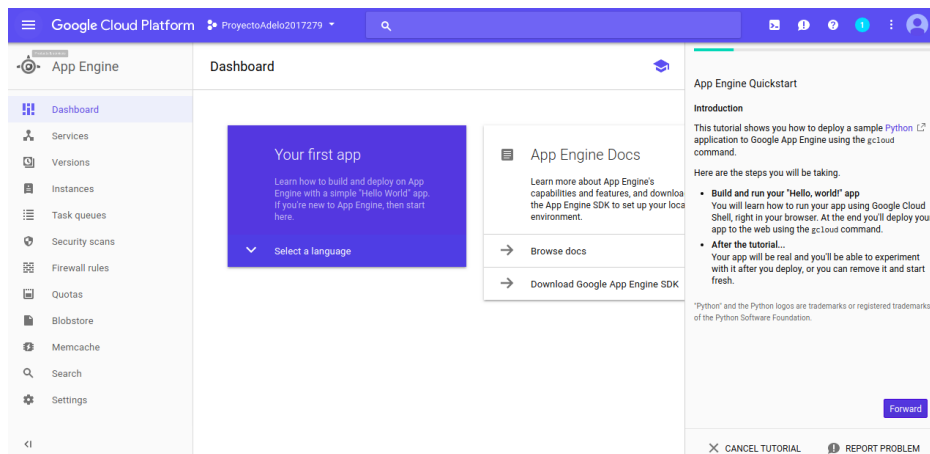


Figure 8.1: Starting the python tutorial in the App Engine page of the GCP

1. Google Cloud Shell:

- In the GCP we can open a command line console (Google Cloud Shell) where we can develop the application. «The Cloud Shell is a built-in command line tool for the console. We’re going to use Cloud Shell to deploy our app» (Figure 8.2) [cloud.google.com (b)]
- Open Cloud Shell by clicking `>_` from the navigation bar at the top.

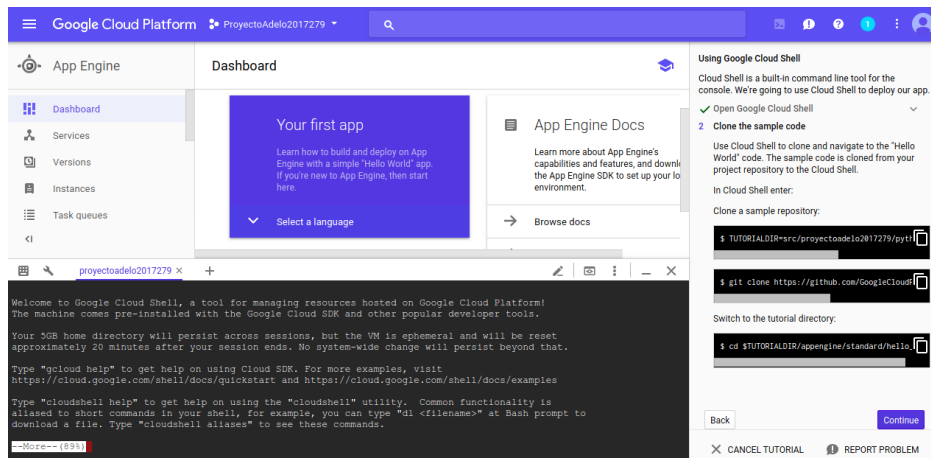


Figure 8.2: In the GCP we can open a command line console (Google Cloud Shell) where we can develop the application

2. Clone the sample code:

- Use Cloud Shell to clone and navigate to the "Hello World" code. The sample code is cloned from your project repository to the Cloud Shell.
- Clone a sample repository:

```
TUTORIALDIR=src/proyectoadelo2017279/python_gae_quickstart-2018-05-06-00-26
git clone https://github.com/GoogleCloudPlatform/python-docs-samples $TUTORIALDIR
```

Switch to the tutorial directory:

```
cd $TUTORIALDIR/appengine/standard/hello_world
```

3. Configuring your deployment

You are now in the main directory for the sample code. We'll look at the files that configure your application.

- Exploring the application:

Enter the following command to view your application code (Figure 8.3):

```
cat main.py
```

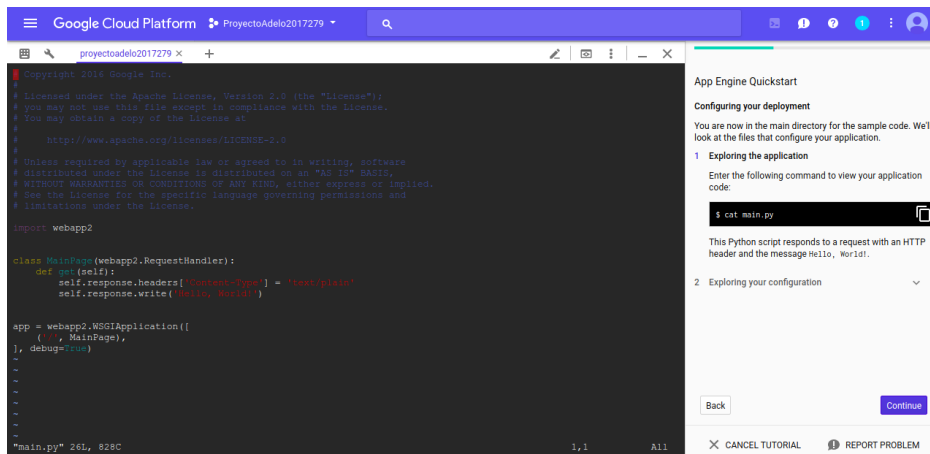


Figure 8.3: Exploring the application in the Google Cloud Shell Console

- Exploring your configuration:

- Google App Engine uses YAML files to specify a deployment’s configuration. app.yaml files contain information about your application, like the runtime environment, URL handlers, and more.
- Enter the following command to view your configuration file:

```
cat app.yaml
```

4. **Testing your app:** The application is a simple Python application that uses the webapp2 (<https://webapp2.readthedocs.io/>) web framework.

(a) Test your app on Cloud Shell

- Cloud Shell lets you test your app before deploying to make sure it’s running as intended, just like debugging on your local machine.
- To test your app enter:

```
dev_appserver.py $PWD
```

(b) Preview your app with "Web preview" Your app is now running on Cloud Shell. You can access the app by using "Web preview" (at the top right corner of the console) > Preview on port 8080

(c) Terminating the preview instance: Terminate the instance of the application by pressing Ctrl+C in the Cloud Shell.

5. **Create the application** In order to deploy our app, we need to create an App Engine application. This sets up the app and selects a region.

- To create your app enter:

```
gcloud app create
```

6. Last steps

- Deploying with Cloud Shell You can use Cloud Shell to deploy your app. To deploy your app enter:

```
gcloud app deploy app.yaml --project proyectoadelo2017279
```

- Visit your app Congratulations! Your app has been deployed. Click this URL to visit it (Figure 8.4):
`proyectoadelo2017279.appspot.com`
- View your app's status You can check in on your app by monitoring its status on the App Engine dashboard.

Open the menu on the left side of the console.

Then, select the App Engine section.



Figure 8.4: Our first Python App created using the Google Cloud App Engine
`proyectoadelo2017279.appspot.com`

9 Challenging research topic - GCSFUSE: It allows you to mount a bucket to a Debian Linux virtual machine

Google introduced GCSFUSE. This allows you to mount a bucket to a Debian Linux virtual machine.

We have to mount the bucket to the Linux VM, install Apache and load the web page from the bucket to the VM and set up a web site. The web page should say "DigiTech on Apache Loaded from the Attached Bucket" and have your name on it.

Cloud Storage FUSE is an open source FUSE adapter that allows you to mount Cloud Storage buckets as file systems on Linux or OS X systems. cloud.google.com (a).

9.1 Installing Cloud Storage FUSE and its dependencies

- Add the gcsfuse distribution URL as a package source and import its public key:

```
export GCSFUSE_REPO=gcsfuse-`lsb_release -c -s`  
echo "deb http://packages.cloud.google.com/apt $GCSFUSE_REPO main" | sudo tee /etc/apt/  
sources.list.d/gcsfuse.list  
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

- Update the list of packages available and install gcsfuse:

```
sudo apt-get update  
sudo apt-get install gcsfuse
```

9.2 Mounting a Google Cloud Storage Bucket as a local disk

- Create a directory:cloud.google.com (a)

```
sudo mkdir /var/www/mi_storage1
```

- Use Cloud Storage FUSE to mount the bucket:thedotproduct.org

```
sudo gcsfuse -o noatime -o noexec --gid 33 --implicit-dirs -o ro -o nosuid -o nodev --  
uid 33 -o allow_other mi_storage1 /var/www/mi_storage1
```

9.3 Change the root directory of an apache server

To do so, we had to edit the file:

```
/etc/apache2/sites-available/000-default.conf
```

In figure 9.1 we shown the configuration we made in `/etc/apache2/sites-available/000-default.conf`

```
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
# DocumentRoot /var/www/html
DocumentRoot /var/www/mi_storage1/DigiTech
```

Figure 9.1: Change the root directory of an apache server

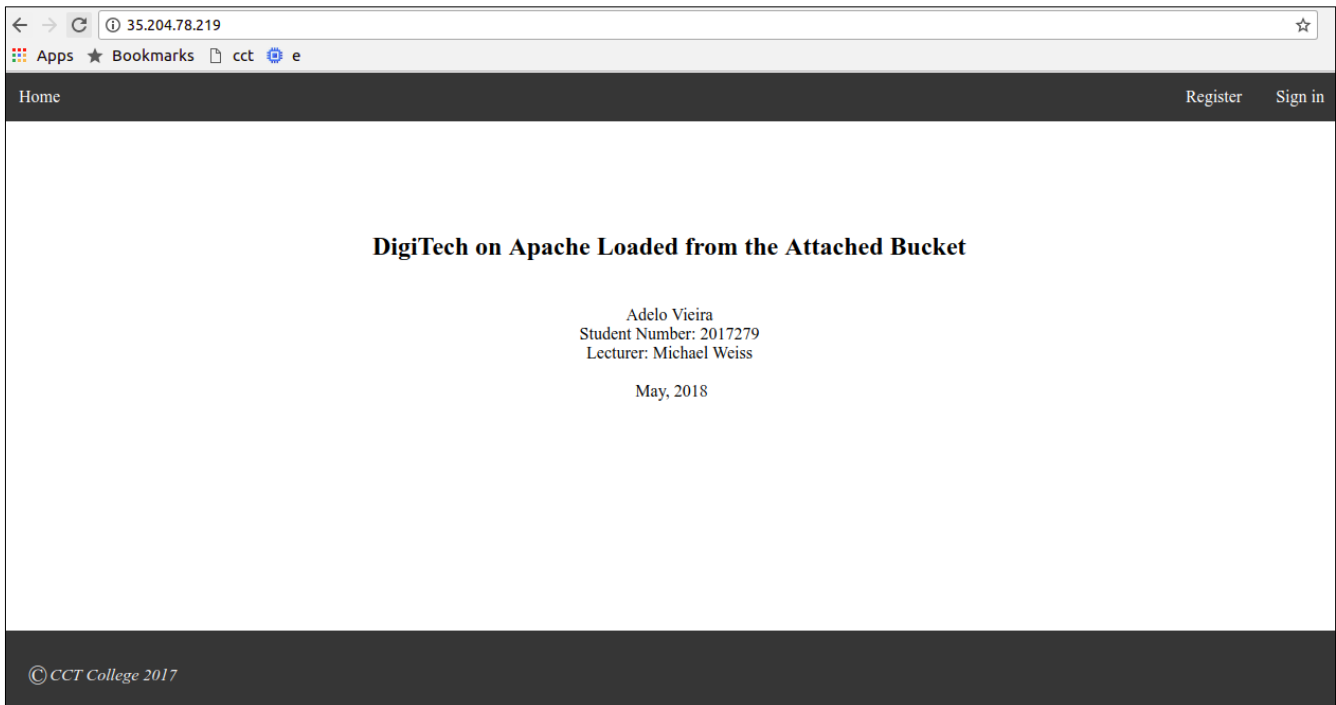


Figure 9.2: DigiTech on Apache Loaded from the Attached Bucket

Declaration

I hereby declare that all of the work shown here is my own work.

Student's Name: Adelo Vieira

Student Number: 2017279

Date: May 8, 2018

Bibliography

- Carol Britton and Jill Doake. *A student guide to object-oriented development*. Butterworth-Heinemann, 1 edition, November 22 2004.
- cloud.google.com. Cloud storage fuse, a. URL <https://cloud.google.com/storage/docs/gcs-fuse>. 17, 18
- cloud.google.com. Quickstart for python app engine standard environment, b. URL <https://cloud.google.com/appengine/docs/standard/python/quickstart>. 14
- cloud.google.com. Live migration documentation. URL <https://cloud.google.com/compute/docs/instances/live-migration>. 12, 13
- github.com. Installing cloud storage fuse and its dependencies. URL <https://github.com/GoogleCloudPlatform/gcsfuse/blob/master/docs/installing.md>.
- stackoverflow.com. Forum: do i change the root directory of an apache server? URL <https://stackoverflow.com/questions/5891802/how-do-i-change-the-root-directory-of-an-apache-server>.
- thedotproduct.org. Mounting a google cloud storage bucket as a local disk. URL <https://www.thedotproduct.org/posts/mounting-a-google-cloud-storage-bucket-as-a-local-disk.html>. 18
- wikipedia.org. Google app engine. URL https://en.wikipedia.org/wiki/Google_App_Engine. 13
- youtube.com. What is app engine? URL <https://www.youtube.com/watch?v=s1g4H4-MSJg>. 13