Google Cloud Platform

Monday 29th April, 2019 - 18:03

Kevin L. Biewesch
University of Luxembourg
Email: kevin.biewesch.001@student.uni.lu

Alfredo Capozucca
University of Luxembourg
Email: alfredo.capozucca@uni.lu

Abstract

This document is a template for the scientific and technical (S&T for short) report that is to be delivered by any BiCS student at the end of each Bachelor Semester Project (BSP). The Latex source files are available at: ***https://github.com/nicolasguelfi/lu.uni.course.bics.global***

This template is to be used using the Latex document preparation system or using any document preparation system. The whole document should be in between 6000 to 8000 words (excluding the annexes) and the proportions must be preserved. The other documents to be delivered (summaries, ...) should have their format adapted from this template.

1. Introduction (± 5% total words)

This paper presents the bachelor semester project made by Motivated Student together with Motivated Tutor as his motivated tutor. It presents the scientific and technical dimensions of the work done. All the words written here have been newly created by the authors and if some sequence of words or any graphic information created by others are included then it is explicitly indicated the original reference to the work reused.

This report separates explicitly the scientific work from the technical one. In deed each BSP must cover those two dimensions with a constrained balance (cf. [**?**]). Thus it is up to the Motivated Tutor and Motivated Student to ensure that the deliverables belonging to each dimension are clearly stated. As an example, a project whose title would be "A multi-user game for multi-touch devices" could define as scientific [**?**] deliverables the following ones:

- Study of concurrency models and their implementation
- Study of ergonomics in human-computer interaction

The length of the report should be from 6000 to 8000 words excluding images and annexes.

2. Project description (± 10% total words)

2.1. Domains

There are two big domains that are being tackled in this BSP. On the one hand we have virtualization and on the other hand we have cloud computing along side its service models.

2.1.1. Scientific. The different service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The three service models are briefly explained in the following.

IaaS We are provided with the necessary infrastructure to run a virtual machine and the rest is up to us.

PaaS We get the necessary resource as well as an OS installed with some basic software. We can use the provided platform and modify it as we please. However we are not able to touch the underlying virtual hardware.

SaaS We are provided with everything needed to run a specific software. We are only able to access the software and none of its underlying components, like the OS or the hardware.

2.1.2. Technical. Virtualization allows us to move from physical mediums to virtual ones without really losing anything. The only major difference is that now we do not manipulate the hardware directly but instead manipulate the virtual hardware through specific software. This allows for great flexibility in a lot of ways.

Cloud computing brings the domain of virtualization to yet another level. We are still manipulating a virtual medium, but the virtual machine does not run locally on our physical machines. Instead these machines run on some servers and we can controll them through a remote connection. This leads us to the various service models employed by the cloud.

2.2. Targeted Deliverables

2.2.1. Scientific deliverables. We want to create a tutorial that accompanies our technical solution. The tutorial is meant to guide the DevOps engineer and help him understand and use our solution.

2.2.2. Technical deliverables. We have to consider two types of users: the DevOps engineer and the Excalibur user.

As for the DevOps engineer, the objective of this BSP is to have a fully automated solution that handles the virtual machine creation on the Google Cloud Platform. Added to this, all the necessary tools should be downloaded such that we can put and Excalibur Environment on it. He also needs to ensure access to the new virtual machine for the Excalibur user.

As for the Excalibur user, we want him to simply be able to connect to the virtual machine created in the cloud and use the Excalibur Environment. The end user should not worry about anything related to the virtual machine's creation and setup.

3. Background (± 10% total words)

Describe in this section the main knowledge supposed to be formerly known by you and that is useful to remind in order to understand the remaining parts of your report. Do not include presentation of technologies or scientific concepts that belong to an objective of your BSP since it must be described in the section **??**. Thus all the content of this section is not considered as a deliverable

3.1. Scientific background

Virtual machines constitute the big scientific aspect to be familiar with for this BSP. They were already briefly explained before in section 2.1.

Also the concept of provisioning should not be foreign. In essence, provisioning allows us to put files and packages as well as execute scripts onto a virtual machine without accessing it directly. We can write scripts, or in our case configuration files, that will then access the machine and accomplish all the specified tasks.

3.2. Technical background

The technical aspect to be familiar with for this BSP were how provisioning works with Ansible. This is important because we are trying to reuse the work done in the previous BSP and get it to work on our virtual machines in the cloud.

Ansible is a tool that allows to write configuration files that basically specify what a machine should be provisioned with. It allows for great flexibility and keeps the focus on easy to read and write code.

4. A Scientific Deliverable 1

For each scientific deliverable targeted in section 2.2 provide a full section with all the subsections described below.

4.1. Requirements (± 15% of section's words)

Describe here all the properties that characterize the deliverables you produced. It should describe, for each main deliverable, what are the expected functional and non functional properties of the deliverables, who are the actors exploiting the deliverables. It is expected that you have at least one scientific deliverable (e.g. "Scientific presentation of the Python programming language", "State of the art on quality models for human computer interaction", ...) and one technical deliverable (e.g. "BSProSoft - A python/django website for IT job offers retrieval and analysis", ...).

4.2. Design (± 30% of section's words)

Provide the necessary and most useful explanations on how those deliverables have been produced.

4.3. Production (± 40% of section's words)

Provide descriptions of the deliverables concrete production. It must present part of the deliverable (e.g. source code extracts, scientific work extracts, ...) to illustrate and explain its actual production.

4.4. Assessment (± 15% of section's words)

Provide any objective elements to assess that your deliverables do or do not satisfy the requirements described above.

5. A Technical Deliverable 1

For each technical deliverable targeted in section 2.2 provide a full section with all the subsections described below. The cumulative volume of all deliverable sections represents 75% of the paper's volume in words. Volumes below are indicated relative the the section.

5.1. Requirements (± 15% of section's words)

The main techical deliverable for this BSP, is a solution that automates the process of creating and provisioning virtual machines on the Google Cloud Platform in order to host an Excalibur Environment on it. Part of this solution relies on work that has been done in the previous semester.

The functional requirement of this solution is to set up a new Excalibur Environment as fast as possible while needing the least amount of effort. We want as much of the set up as possible to be done by simply executing the provided scripts.

The non-functional requirements are more numerous. We want to tackle:

1) Operability, which determines how accessible and easy to use the solution is.
2) Efficiency, which considers the amount of effort necessary to bring the solution to use.
3) Satisfaction, informs us on the degree to which user needs are satisfied when a product or system is used in a specified context of use.
4) Maintainability, considers the ease at which the developer can change aspects of the Virtual Machine creation and the provisioning.
5) Reliability, tells us whether or not our solution is reliable. In other words: Does it give the same result in successive trials?

This technical deliverable is targeted towards a DevOps engineer, who is in charge of setting up these machines. The Excalibur user should not have to deal with anything regarding the production of the final product, namely the Excalibur Environment.

5.2. Design (± 30% of section's words)

As already mentioned, part of this deliverable relies on work done in the previous semester. In the last semester, we also tried to produce an automated solution, however we were working locally on our machines. In this semester we tried to adapt the knowledge aquired and bring it to the next level, namely cloud computing.

While the basic concepts remain the same, we now have to look at the subject from a different perspective which is due to the different nature of working locally versus working in the cloud. Before, all the DevOps could do was to provide scripts the user had to run himself. We considered there was no way for a DevOps to interact with the user's machine which means that actually creating the machine with the provided solution was up to the user.

Now we want a DevOps to create and set up the machines in the cloud and the Excalibur user will merely have to connect to it and is ready to go. There is close to nothing that the user will have to do in order to get access to an Excalibur Environment. Essentially, what we are trying to achieve is Software as a Service.

A first difficulty for this BSP was to create virtual machines in the cloud. We had to get accustomed to the Google Cloud Platform and learn how to use it. This was a relatively easy and straight forward process because Google provided interactive tutorials to guide its users through the basics of the platform.

After getting the hang on the basics, we immediately started wondering if we could reuse the scripts that handled the provisioning from the previous semester. We opted for a provisioning solution that was not specific to the previous semester's solution but would allow for a broader range of application. A quick research revealed that indeed, we were able to reuse the provisioning scripts. Only one minor modification had to be done, namely *how* do we execute it[1].

Lastly, we inspected the `gcloud` command line tool which basically allows us to interact with GCP through the terminal. We wrote scripts around this tool to have new machines be created automatically. Automating the provising also relies on some output generated by this tool.

5.3. Production (± 40% of section's words)

The first thing we had to get comfortable with is the Google Cloud Platform. There were a few things we needed to know and had to set up before we could even get started with creating virtual machines.

A first requirement was to have a gmail account to be able to operate on GCP. Next, we need to create a new project in which we manage and manipulate our virtual machines. You can read about it in the tutorial on github[2]. Please Refer to section **Setting up GCP** and section **Creating Virtual Machines**, subsection **Manual Creation**. After having created a first virtual machine, with very few resources attributed to it because it was merely a test and we did not want to have high billing for this, we inspected ways to connect to it. Please refer to section **Connecting to Virtual Machines** and its subsections in the tutorial. The first obvious choice was to connect directly from GCP. However, this did not allow us to provision the machine using our Ansible scripts. So we were forced to inspect manual SSH connection. We had to ensure SSH connection to the virtual machine in order to run our Ansible provisioning.

This is due to the fact, that Ansible relies on a host file or inventory file as it is called. This file contains a list of hosts to provision and Ansible will connect to these host via SSH. There seem to be methods for connecting to a host on GCP from Ansible without requiring manual SSH connection, but these methods looked rather complex and we did not get them to work. That said, we stuck to manually ensuring SSH connection.

```
[GCP]
35.227.50.0

[GCP:vars]
ansible_user = student002_bics_lu
```

Here is an example of a host file that was written manually in order to check if provisioning works. With the `[...]` we can specify a group of hosts. That way we can have Ansible provision only a specific collection of machines if we so desire. However, in the Ansible scripts, we have set it such that all hosts will be provisioned.

Furthermore, we have the `[...:vars]` with which we can specify variables that apply to a certain group of hosts. Here we say what user to connect as on the remote machine. With the scripts that automate the solution, our host file will be generated and look a little different.

```
35.227.50.0  ansible_user = student002_bics_lu
35.653.43.0  ansible_user = some_other_user
```

This way we are specifying the *main* user for each machine. Since each machine will have multiple users, we will get to this later, we need to specify a user as which to connect as to do the provisioning.

After being able to handle the provisioning, which is a big step in the right direction, we looked into how to automate the creation of the virtual machine. Our first attempt, was to integrate this into our Ansible workflow. There are methods provided by Ansible to create virtual machines on GCP but here again, we did not manage to get them to work. So we went for the gcloud tool for doing this. The gcloud command we used to create virtual machines is a follows. Please refer to the tutorial section **Creating Virtual Machines** subsection **Creation using gcloud** for more details on this command.

```
gcloud compute instances create "$1" \
    --machine-type="$MACHINE_TYPE" \
    --boot-disk-size="$BOOT_DISK_SIZE" \
```

1. In the previous semester, the executing of these scripts was very specific to the solution
2. https://github.com/niveK77pur/ExcaliburEnvironmentGCP

```
    --image-project="$IMAGE_PROJECT" \
    --image-family="$IMAGE_FAMILY" \
    --metadata="$METADATA"
```

After having all of this figured out, we wrote one script around the above command to handle the virtual machine creation and another script is used to handle the provisioning. This second script relies on gcloud to get the external IP of the remote machine because the IP changes each time you start it anew. So we write the IP along with the *main user*, as explained above, to a new host file and have ansible provision the machines specified in that host file.

To wrap things up, we wrote a third script that calls these two scripts. That way we only need to execute one script and everything is handled for us. The last script has the following content.

```
#!/bin/bash

# $1 : instance name
# $2 : main user name at the instance

bash ./new_vm.sh "$1"
bash ./provision.sh "$1" "$2"
```

5.4. Assessment (± 15% of section's words)

Due to time constraints, we are not going to use the Excalibur Tutorial as our test case. Instead, we will do a few basic tasks to check if the machine is usable and see how well our solution meets non-functional requirements.

We are going to download and install Remarkable for this purpose. Since downloading and installing software will be part of the tasks required to set up Excalibur, this seems like a good trade off.

Further, to make this test case more adapted to a real lift situation, we will be running one or two Youtube videos in the background as well as having a word editor opened while doing the installation.

Acknowledgment

The authors would like to thank the BiCS management and education team for the amazing work done.

6. Conclusion

The conclusion goes here.

References

[BiCS(2018a)] BiCS Bachelor Semester Project Report Template. https://github.com/nicolasguelfi/lu.uni.course.bics.global University of Luxembourg, BiCS - Bachelor in Computer Science (2017).

7. Appendix

The final solution along with the tutorial as well as this report are on github: https://github.com/niveK77pur/ExcaliburEnvironmentGCP