Azure + Terraform + Ansible

how to create easily your own GitLab CE server and implement a CI/CD for Python application

What are we going to see?

- Part 1:
  - How to create the infrastructure on Azure with Terraform
  - How to install and pre-configure Gitlab CE with Ansible
- Part 2:
  - How to activate CI/CD on Gitlab
  - How to implement a sample pipeline with a simple dockerized Flask application

Azure A very simple architecture…

Part.1

Virtual Machine, hosting the GitLab CE Server

+ Public IP Address, so we can access our server

+ Storage account, cause we need to store stuff!

+ Network Interface

+ Security Group (basically, security rules)

+ Virtual Network where we belong to

= GitLab

HashiCorp Terraform for helping us!

Part.1

- Allow to *code* your infrastructure
- A provider for driving Azure infrastructure exists
- Using modules allow to better organize the code
- One module for every kind of object we have to manipulate under Azure:
  - resource group
  - storage account
  - virtual network
  - subnet
  - virtual machines

HashiCorp Terraform will do some magic!

- Start to activate you Azure account:

```
$ az login
```

- Then go to your directory:

```
$ cd /Workspaces/azure-ansible-gitlab/terraform
```

- Initialize terraform:

```
$ terraform init
```

- And launch terraform to ***apply*** infrastructure as code:

```
$ terraform apply -var-file ..\config\azure-gitlab-server.tfvars
```

ANSIBLE to configure everything

Part.1

- Being sure to *keep the same configuration* whatever happens
- Start to create a first project, and to *preconfigure* what we can
- Using the GitLab REST API to manage the configuration
- Use a docker image to wrapped ansible, the ssh keys and the playbooks in order to be able to run even under windows 10!

ANSIBLE : let’s install and configure

- First, change from terraform to ansible directory:

```
$ cd ../ansible
```

- Second, build the docker image:

```
$ docker build -t keyteo/gitlab/ansible .
```

- Then launch the image. By default, it’s the ***install*** playbook that is runned, so it’s going to install your server:

```
$ docker run --rm -it keyteo/gitlab/ansible
```

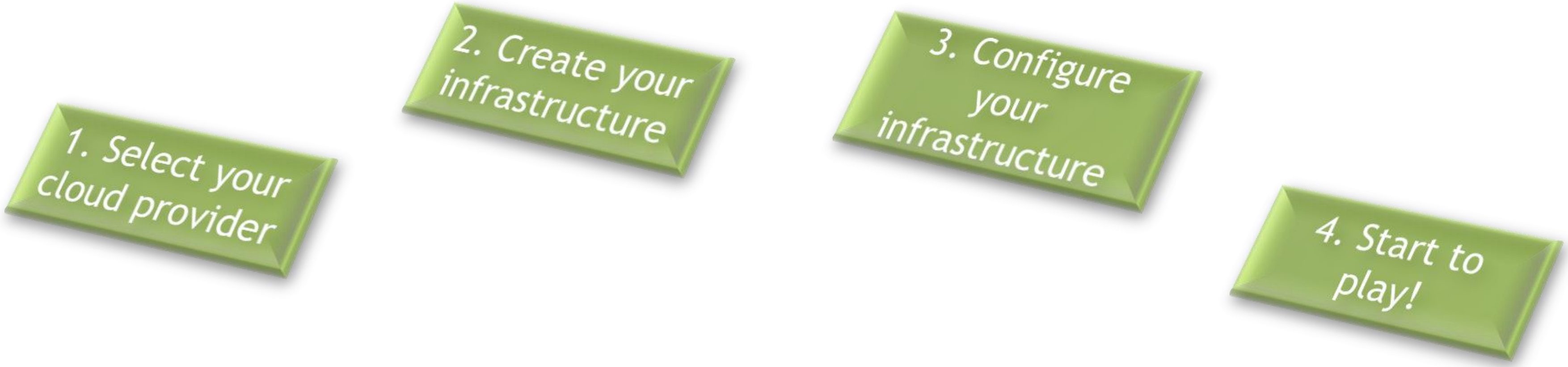
- Now, you need to open the URL of your gitlab server on a browser, to enter the password for the root user. This can’t be automatized, sorry!
- Once done, you can run the ***config*** playbook, with this command:

```
$ docker run -e GITLAB_USER_PASSWORD=<password>
  -e GITLAB_ROOT_PASSWORD=<password> --rm -it keyteo/gitlab/ansible
  /ansible/playbooks/gitlab-config-playbook.yml
```

- A lot happens there, but at the end you have a project ready for next steps!

So, what happens there?

- The workflow is quite easy to understand:



1. Select your cloud provider

2. Create your infrastructure

3. Configure your infrastructure

4. Start to play!

- At this point you already seen the first 3 points! Or at least you should have a pretty good idea of what it is.

- So now, it’s time to play with our all brand new GitLab

How to activate CI/CD on GitLab

Part.2

- First, create a proper SSH Config in order to access easily your gitlab instance.
- Second, add an entry on your /etc/hosts file to add your gitlab instance name.
- Third, generate SSH Keypair that is going to be associated with the root admin user, and optionally another one for you.
- Now, open a browser and go to the URL of you gitlab instance.

The sample application

Part.2

- Since we have now the prerequisites for implementing a nice pipeline, the missing part now is a nice application to build!
- So let’s introduce ***simpleblog***:
  - It’s a very simple Flask application in two parts:
    - Part1 > API to manage users and blog articles
    - Part2 > html templates to display the articles
- So that means that *we are going to build a flask application, run the tests and deploy it somewhere, but only if the tests are validated*. this is our pipeline.
- Now let’s connect to our gitlab server and…

Additional reading…

- Terraform Azure Provider: https://www.terraform.io/docs/providers/azurerm/index.html
- Ansible: extensive list of all the available modules: https://docs.ansible.com/ansible/2.5/modules/list_of_all_modules.html
- Gitlab: REST API documentation: https://docs.gitlab.com/ee/api/
- Gitlab: CI/CD documentation: https://docs.gitlab.com/ee/ci/README.html
- And what's next?
  - Maybe using https instead of http
  - Better deployment script
  - Using docker container instead of physical deployment
  - *And any improvement that you can think about!!!*