

# Outline

A small online news company, Generic Reporting, is using OpenStack to house their current cloud infrastructure. Their webserver is a basic Ubuntu 18.4 install, with Apache2 manually installed and configured onto it. Generic Reporting's new CTO thinks that their current cloud infrastructure is in need of an update. She wants a production server with an identical development server that can be modified/rebuilt at a moments notice.

## Hand-in instructions

All submissions must be made via Learning Central. Upload the following files in a **single zip file named "CMXXXX\_[student\_number].zip"**. Make sure that the files are organised in the folder structure specified.

### Hand in files

| Description       | Type                        | Name   |
|-------------------|-----------------------------|--|
| Cover sheet       | PDF (.pdf) file             | [student_number].pdf   |
| Terraform section | Terraform (.tf) files       | provider.tf<br>compute.tf<br>networking_secgroup.tf<br>networking.tf<br>output.tf<br>vars.tf |
|                   | SSH key file (no extension) | dragon<br>pheonix  |
| Ansible section   | Configuration (.cfg) file   | ansible.cfg  |
|                   | Plain file (no extension)   | hosts  |
|                   | HTML (.html) file           | file.html  |
|                   | YAML (.yaml) file           | main.yml   |

### Folder structure

```
zip file
├── coversheet
├── terraform/
│   ├── ...all 8 Terraform section files...
├── ansible/
│   ├── ...all 4 Ansible section files...
```

# Tasks

1. Using Terraform, construct the proposed cloud infrastructure as code from the description below.

The new infrastructure must contain two server instances that are identical in all respects other than names and SSH keys. These two instances should be both connected to a network along with a router that is also connected to the internet. There should only be one subnet with all three of these devices on it. The subnet DNS nameservers should be set to the University DNS servers. Using exactly one security group allow SSH, HTTP, and HTTPS connections to the subnet. All connections can be allowed from anywhere.

The server instances should be named “Production” and “Development”. “Development” should be allocated a dynamically associated IP address pulled from the University private floating IP pool. “Production” should have a statically associated IP address that is pre-reserved on OpenStack and will therefore result in the instance having the same IP address each time you create it with Terraform. This IP should **not** be written in any of the Terraform source files but should be stored in the “vars.tf” file.

There should be two different SSH keypairs for accessing the instances. One for “Production” and one for “Development”, the public keys for these keypairs should be stored in the “vars.tf” file. The “Production” keypair should be called “pheonix” and the “Development” keypair should be called “dragon”, both the private keys also need to be submitted.

There should be six outputs from the Terraform deployment, the IPs of each instance, the security groups of each instance, and the SSH key name of each instance.

Your Terraform code should be split into multiple files, each file should contain the following:

| File name              | What it contains   |
|------------------------|--|
| compute.tf             | All blocks with resource type starting “openstack_compute” |
| networking_secgroup.tf | Resource types starting “openstack_networking_secgroup”    |
| networking.tf          | All other “openstack_networking” blocks                    |
| provider.tf            | The provider block   |
| vars.tf                | All variable blocks  |
| output.tf              | All output blocks  |

2. Write an Ansible playbook that can provision the webserver from task 1 to the following specifications:
- Apache2 installed and running
  - A file named file.html has been uploaded to the server in such a way that going to the address [IP]/file.html will return that file. (The content of the file is unimportant, just some text to prove that it works is sufficient)
  - All packages in the system are up to date
  - The dynamically assigned IP for the development server from task 1 is entered **at runtime** (entered when Ansible is run, not hard-coded in a file)

# Marking criteria

## Task 1

- Production server instance is associated with a static IP that is declared in the vars.tf file
- Development server instance is associated with a dynamic IP
- All servers should be on the same subnet and network
- The two webservers should not be accessible by anything other than HTTP, HTTPS and SSH
- Only one security group has been used
- Two SSH keys used. One for each webserver.
  - Production key named pheonix
  - Development key named dragon
  - Public keys stored in vars.tf file
  - Private keys authenticate correctly and allow access to server setup with public key
- The Terraform module should have six outputs.
  - The IPs of each instance
  - The security groups of each instance
  - The name of the SSH key used by each instance
- Terraform code split into multiple files as specified in the hand-in instructions.
- Effort has been made to variablise values that are not static

## Task 2

- Only one IP is defined in the hosts file (production IP)
- A working method to input a IP to Ansible at runtime is used (command line args, prompt, ect)
- There are at least three tasks in the Ansible playbook that do the following:
  1. Update the system packages - Only this task is allowed to be accomplished with the raw module
  2. Install Apache2
  3. Upload a html file from the local machine to the servers and puts it into the /html/ directory of the Apache webserver
- All tasks successfully run on both server instances
- The uploaded file "file.html" is accessible on both instances from the "web". (Cardiff University internal network)