**Exercise 4b**

*Using Docker Compose with Terraform on AWS*

**Prior Knowledge**

Unix command-line  
Apt package manager  
Amazon AWS Access key and SSH key

**Learning Objectives**

Understand how Terraform manages cloud infra

**Software Requirements**

* AWS
* Docker-Compose
* Terraform

**Overview**

Terraform is a very useful tool from Hashicorp that enables declaratively defining and managing infrastructure in various cloud environments, including AWS, Azure, Google, Kubernetes, Alibaba and many others.

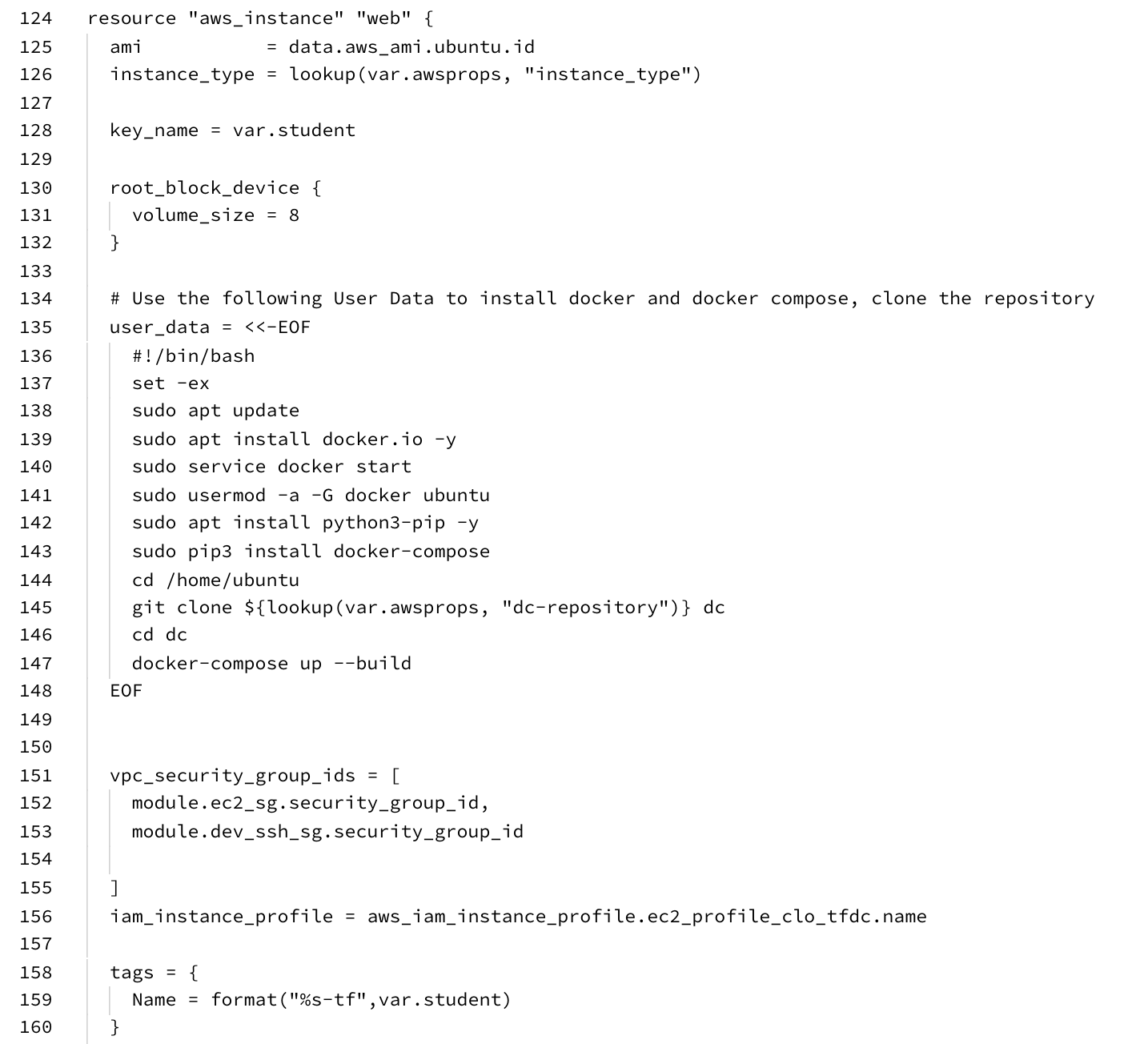
<https://terraform.io>   
  
We are going to use Terraform to instantiate a virtual machine in EC2 and run a Docker Compose workload in it.

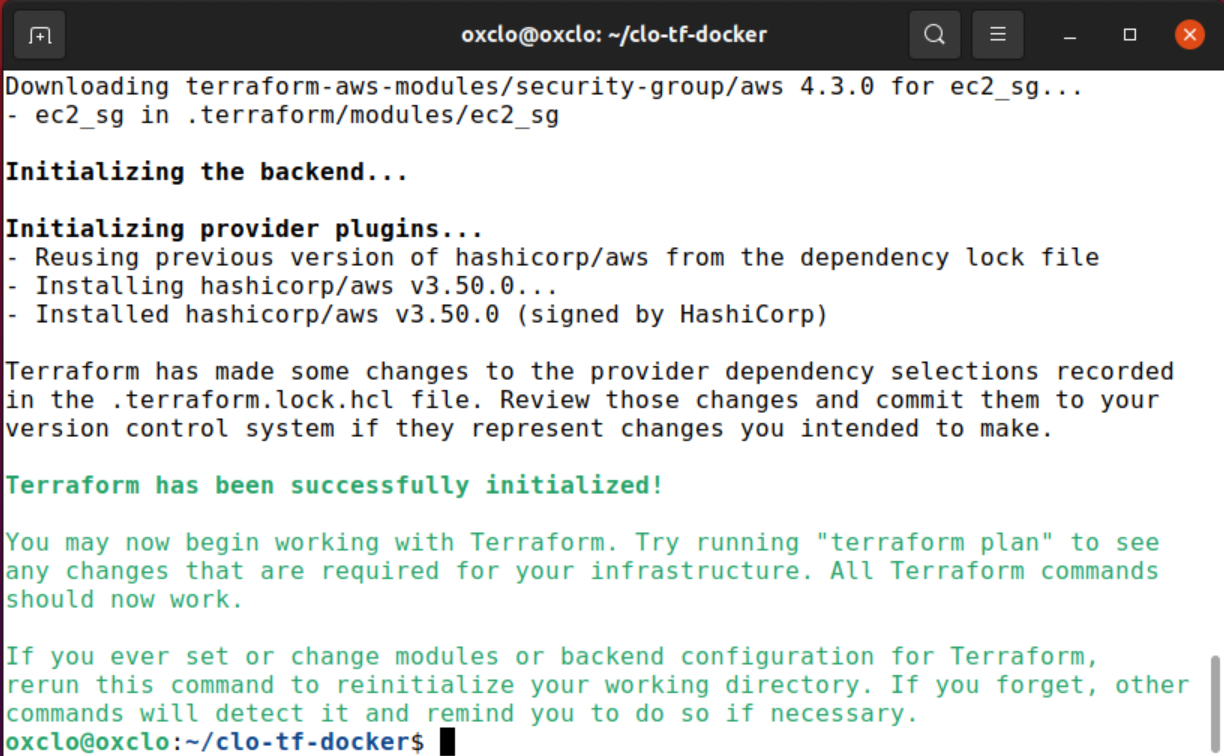
**Steps**

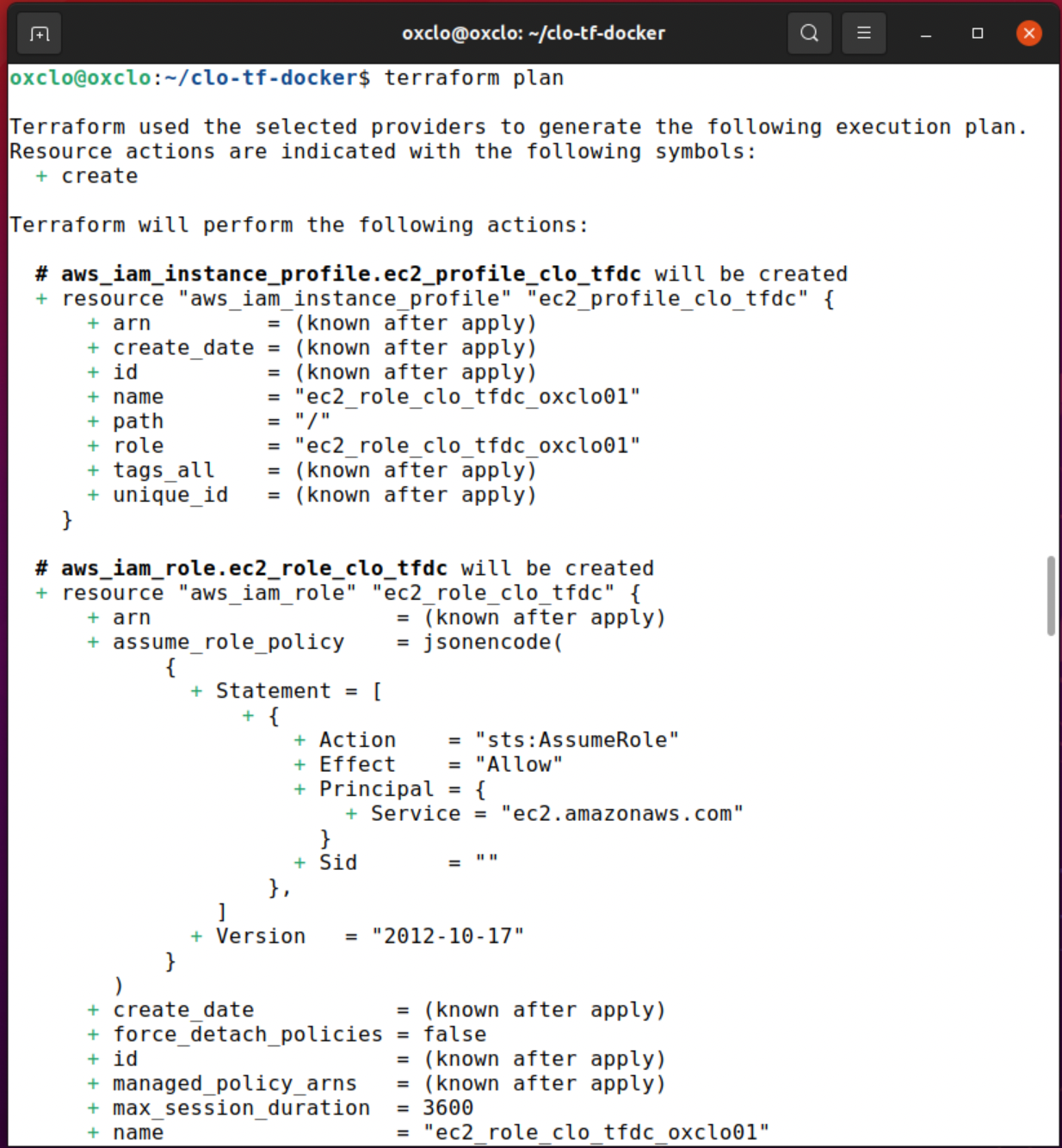
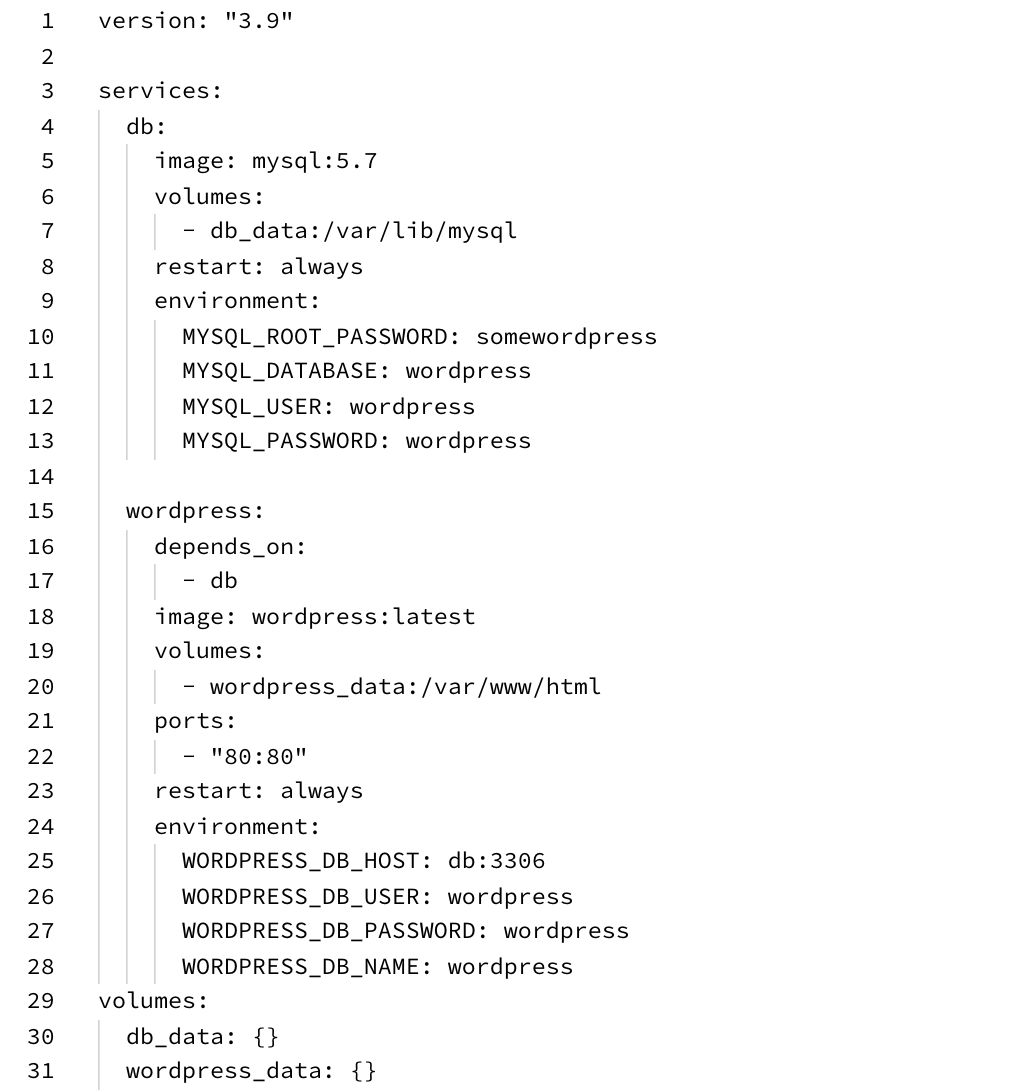
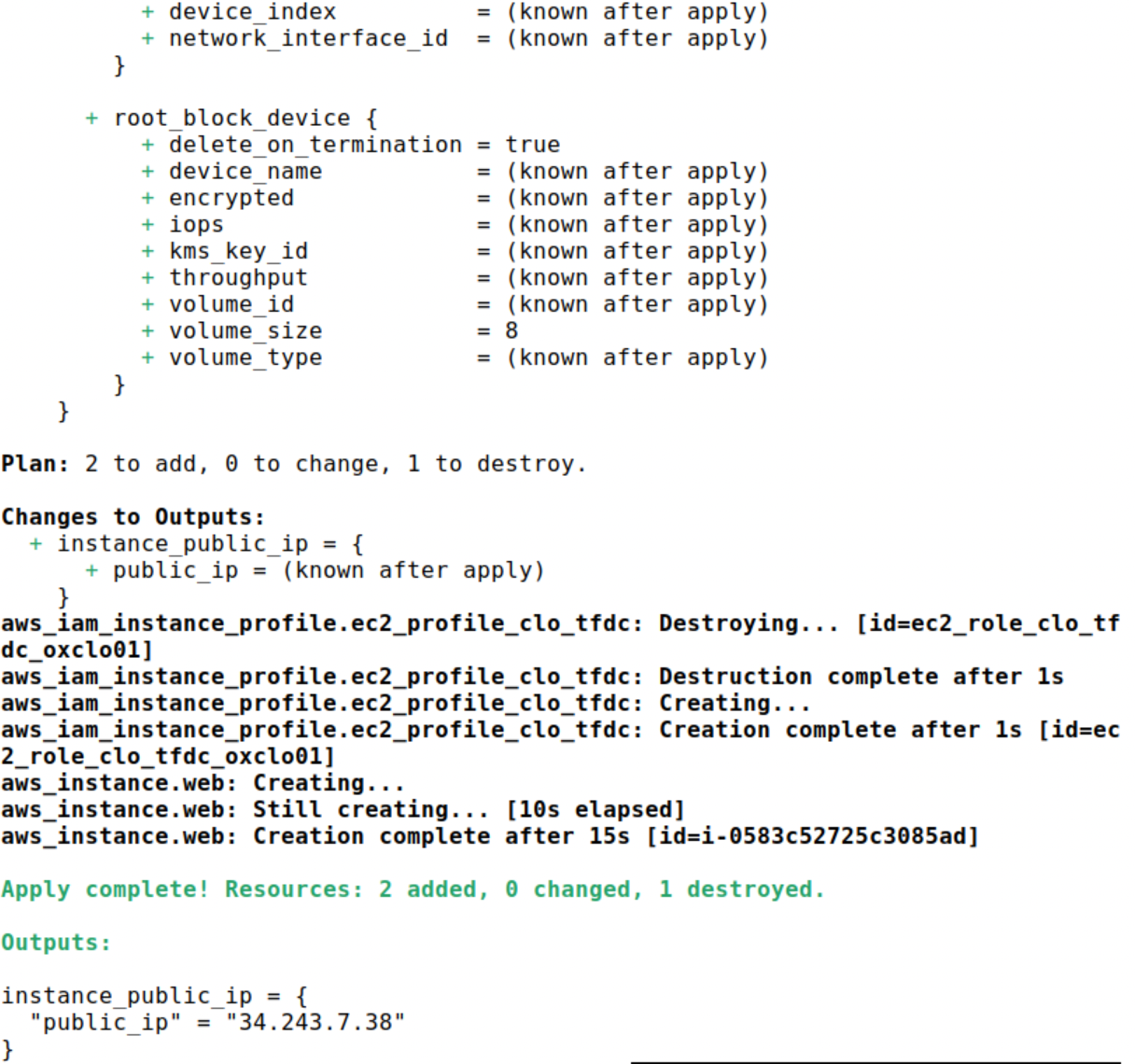
1. Install terraform into the Ubuntu VM:  
     
   wget -O - -q https://freo.me/install-tf | bash
2. Test it works:  
     
   $ terraform -version

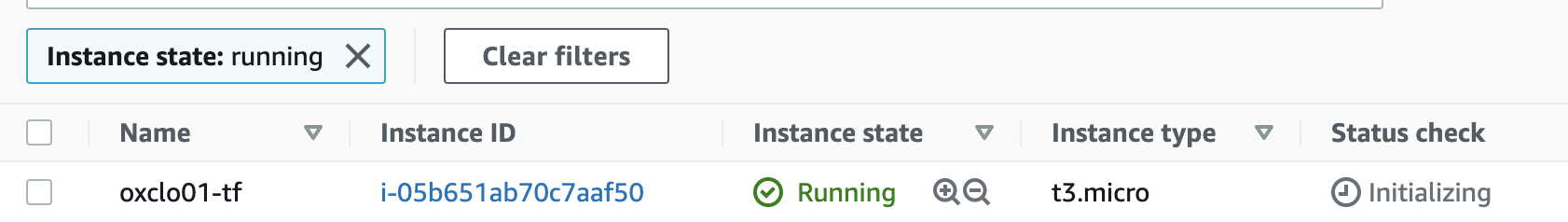
Terraform v1.0.2

on linux\_amd64

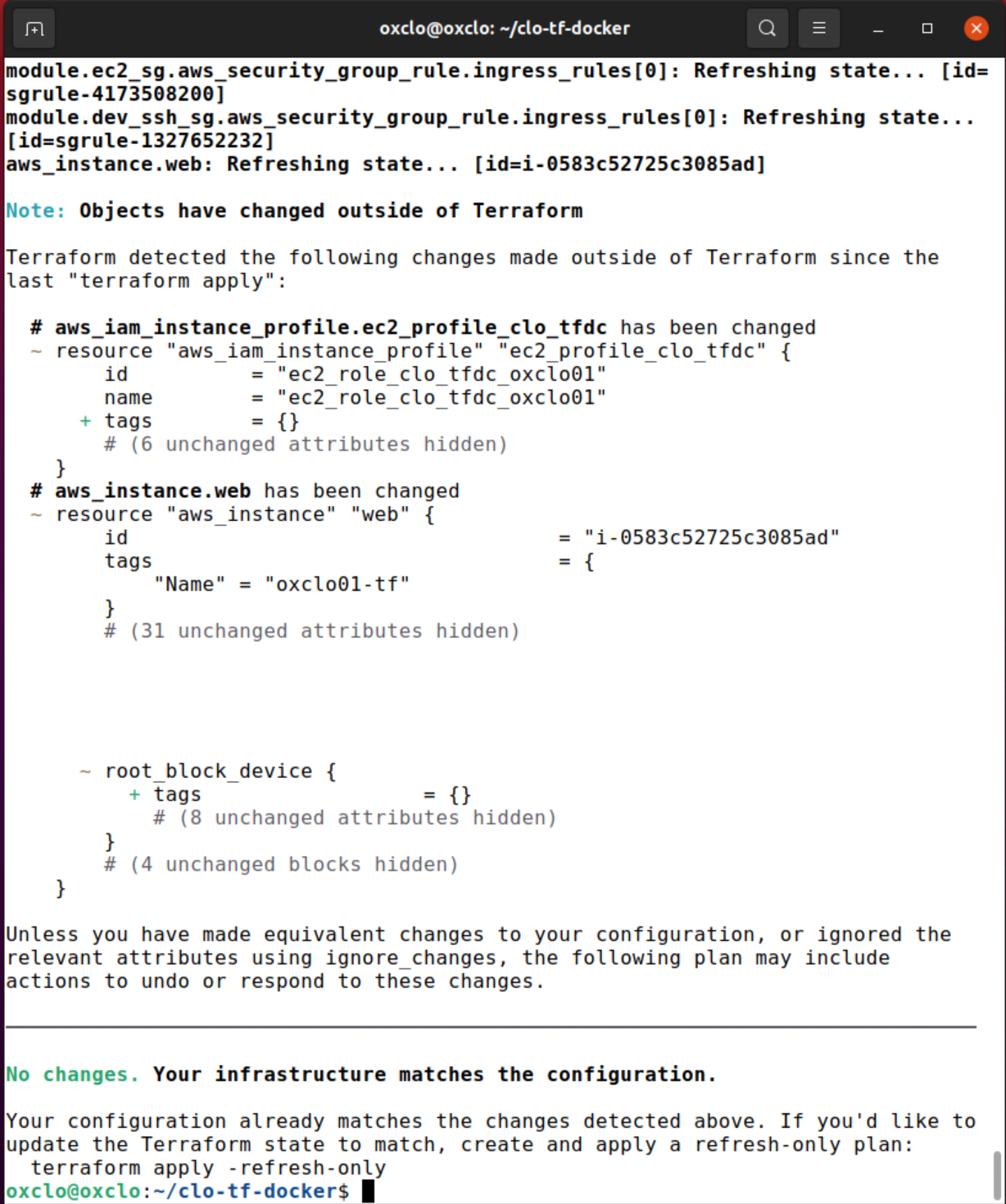
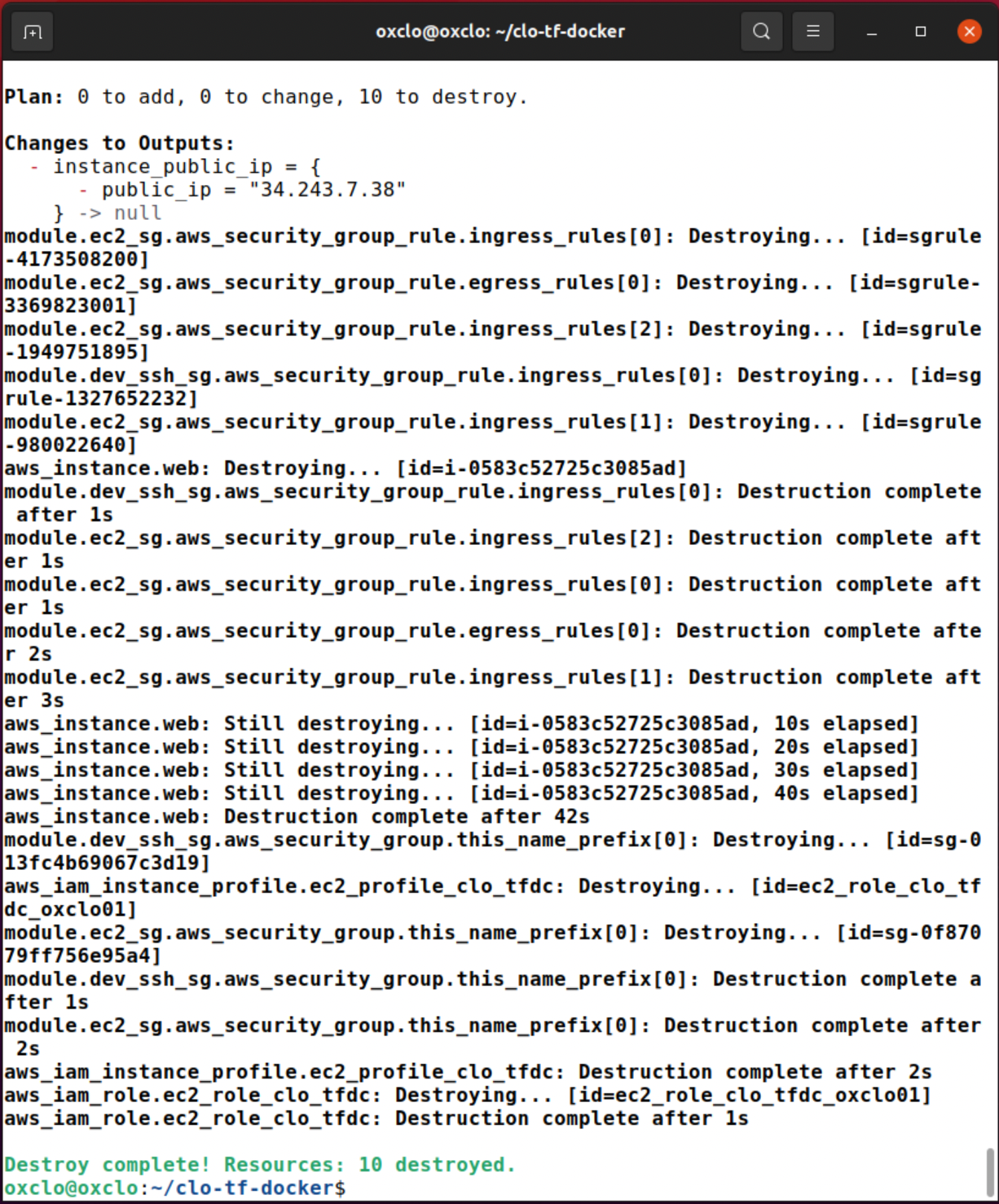
1. Clone the sample repository:  
     
   git clone <https://github.com/pzfreo/clo-tf-docker.git>  
   cd clo-tf-docker
2. Take a look at the file main.tf. It is reasonably well commented and should make sense.  
     
   code main.tf  
     
   The main part is the bit that creates the AWS EC2 instance (everything else is creating subsidiary parts):  
     
   
3. Notice how much control we have over subnets, VPCs, IAM roles, security groups, etc!
4. It is also really useful that you can use variables in this language (which is known as Hashicorp Configuration Language (HCL)).
5. Change the default student name to your student name. (You can also override this when you use this plan)
6. Before we can use terraform, we need to initialise the system, which downloads any providers that are defined (in our case the AWS EC2 provider):  
     
   terraform init

You should see a bunch of stuff scroll by:  
  


1. You can see what terraform plans to do by:  
     
   terraform plan  
     
   
2. Take a look at the docker-compose.yaml:  
     
   This is the standard “wordpress” docker-compose:  
     
   
3. Now we can apply our terraform project:  
     
   terraform apply -auto-approve
4. Terraform will use your access key and secret to call EC2 to create everything needed:  
   

1. You should see your instance starting in the EC2 console:  
   
2. It will take a bit of time for the userdata / cloud-init script to run.

If you want, you can SSH into the instance and  
  
tail -f /var/log/cloud-init-output.log  
  
to see what is happening

1. Once it does, you should be able to access your service on port 80 on the server. The IP address was printed out as part of the startup.
2. You can also check if the current infrastructure matches the plan:  
     
   terraform plan  
     
   
3. You can delete everything with:  
     
   terraform destroy -auto-approve  
   
4. That’s all!

**Extension**

Fork the clo-tf-docker repository in Github and modify it to use the docker-compose file from Ex 4.  
  
Hints:  
  
1. You will need to change which repository is pulled from Git in main.tf

2. You will need to use the docker-compose file from the node-docker repository

3. Notice that because I have previously done “docker-compose push”, it is possible to “docker-compose pull / docker-compose up” without the source   
4. If we were being more rigorous, we could version the docker images and create a git tag / release to specify the exact version of the docker-compose file to use.