# ICT & Infra S3 Automation & Orchestration, week 4

|  |  |
| --- | --- |
| Class: |  |
| Student number: |  |
| Student name: |  |

## Introduction

This week you will practice provisioning an infrastructure using Terraform configuration file. Additionally, Ansible Playbook(s) must be used to automate a complex process. Before executing the assignment, ensure that you have working Ansible control node. For this assignment, you must be familiar with preparing Apache server and hosting Flask application using Ansible.

Follow the tutorial at <https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code> and complete all the steps to install and configure Terraform. Practise to build, change and destroy an infrastructure. Additionally, practise to define Input Variables and query Data with Outputs.

This is **individual** assignment.

### Assignment 1. Deploy a Static Website on an EC2 Instance in a New VPC Using Terraform and Ansible

### Difficulty: ★★★★★.

During the Week 4 lesson you saw an example how to use Terraform to provision a simple infrastructure. In the demo, we had no time to host a website.

In this assignment, you will use Terraform to create a new VPC and an EC2 instance in it. You will then use Ansible to configure the EC2 instance by installing and configuring software to host a static website. You may host a project from your previous semester or use any available static website from online resources. Beware, you must execute whole automation from Ansible playbook. This means that the terraform script must be executed from the hostWeb.yml playbook (as in the Process Diagram).

Requirements:

1. Configure the AWS CLI on your local machine (or VM).
2. Create a Terraform module to deploy the following resources:
   * A new VPC with a CIDR block of 10.10.0.0/16
   * A public subnet with a CIDR block of 10.10.1.0/24
   * A new EC2 instance in the public subnet with Ubuntu as the OS, t3.micro instance type, and with a new security group that allows SSH access and HTTP(S) traffic.
   * EC2 instance AMI must be selected dynamically.
3. Use Terraform to deploy the AWS resources.
4. Once the resources are deployed, use Ansible to configure the EC2 instance by doing the following:
   * Install Apache web server on the EC2 instance.
   * Copy a static website files to the EC2 instance.
   * Configure Apache to serve the static website files from the local file system on the EC2 instance.
   * Test the Apache installation by accessing the website via the public IP of the EC2 instance.
5. Submit your **Terraform and Ansible code**, along with a short write-up that describes your approach and any issues you encountered.

The following process diagram shows possible process for your solution:

Diagram

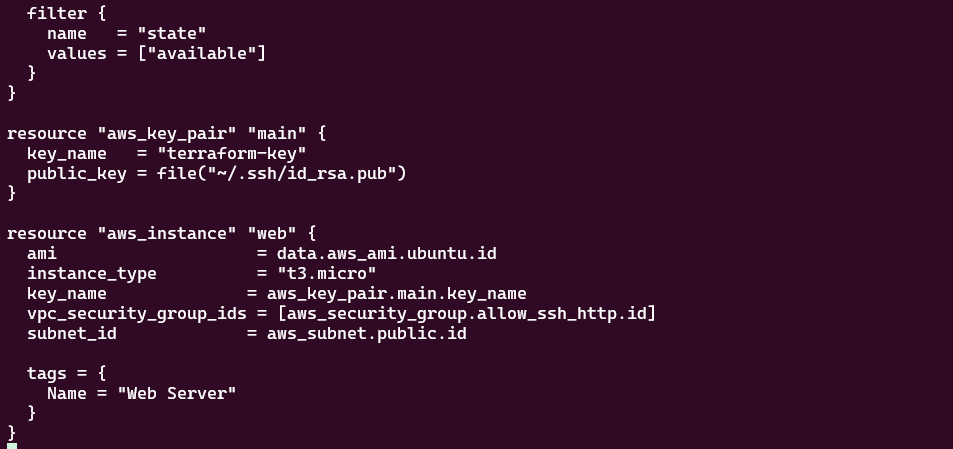
Description automatically generated

Provide screenshots (evidence) for your solution. Always explain your evidence! As a prof, we expect at least:

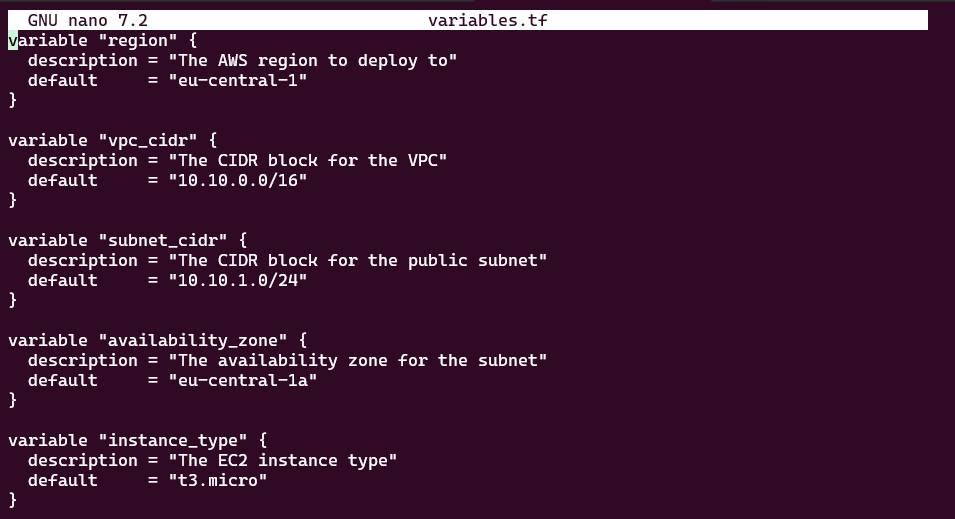
* A screenshot of running EC2 Ubuntu instance.
* Ansible Playbook file(s) that automates the required process.
* Ansible output after executing the playbook.
* Terraform code
* A prof that the website is working.

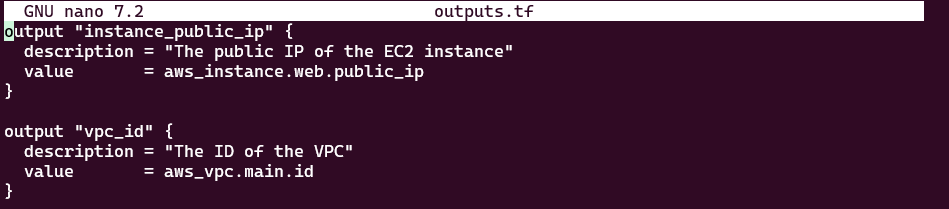
|  |
| --- |
| *Solution:* |

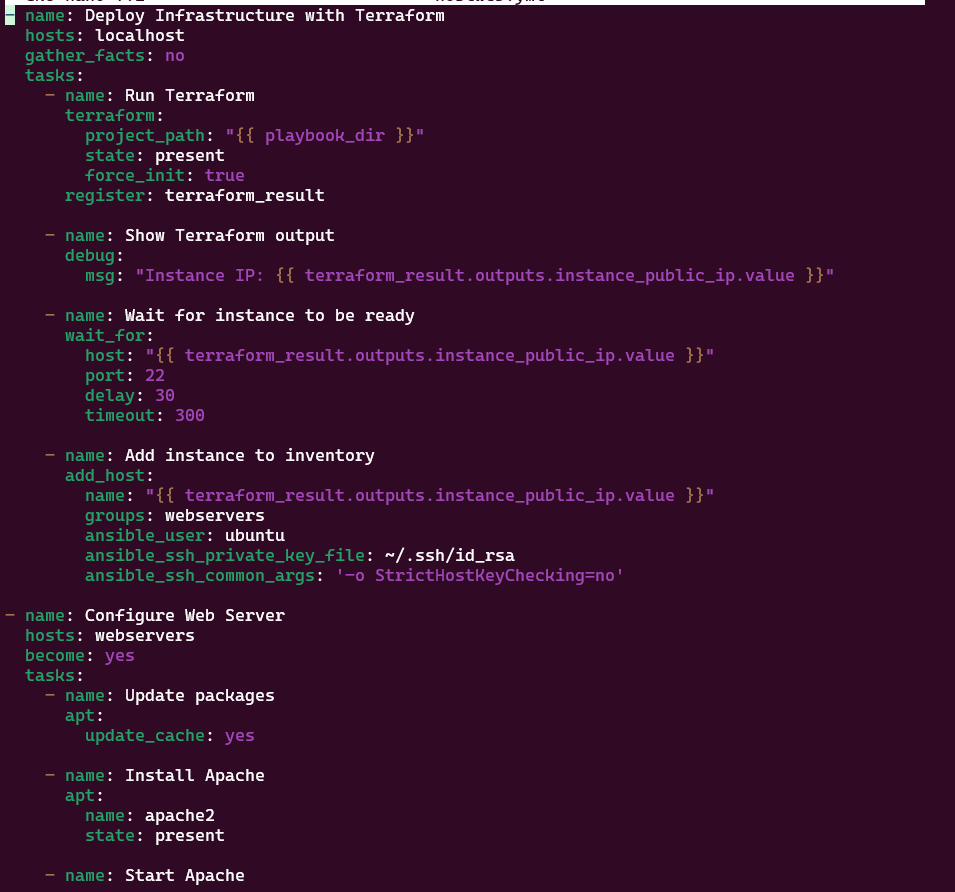
Afbeelding met tekst, schermopname, software

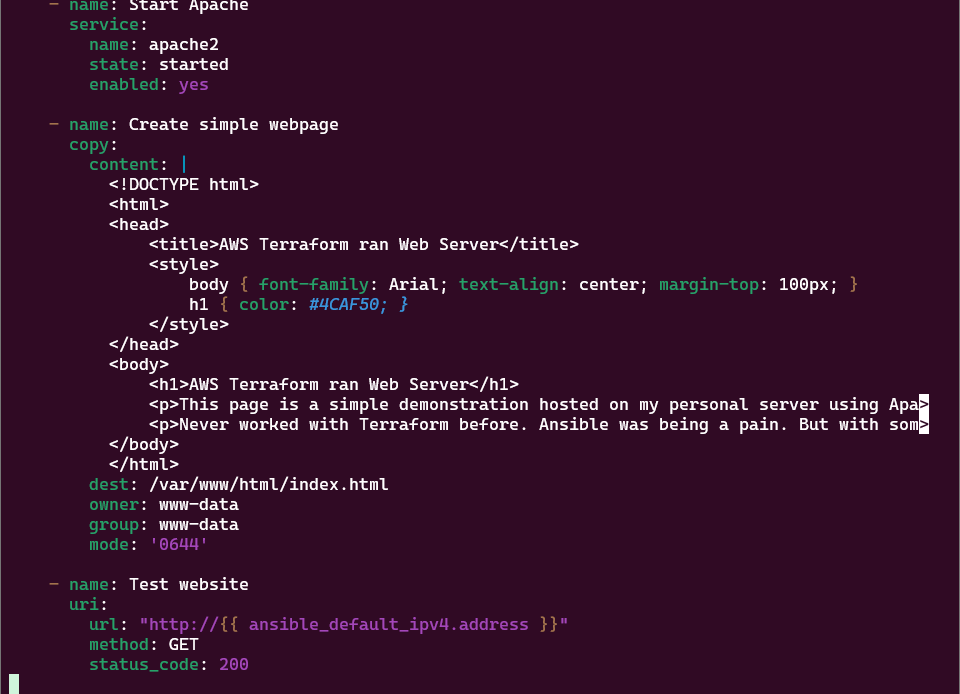
Door AI gegenereerde inhoud is mogelijk onjuist.

Main.tf set’s up vpc with ingress and egress port rules then find ami I had to add filters because I kept getting erros I couldn’t find a viable ami, next is ssh key and the ec2 instanse creation

  
variables.tf

  
outputs.tf

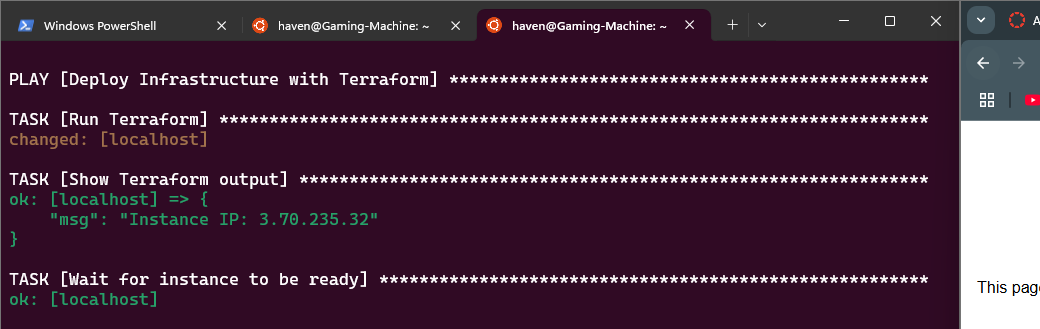


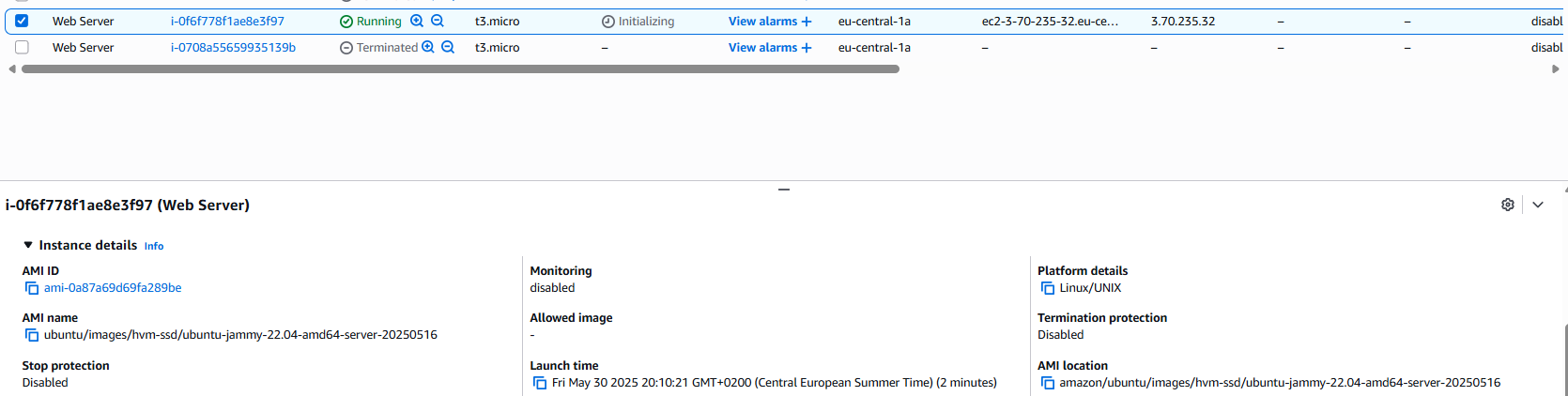


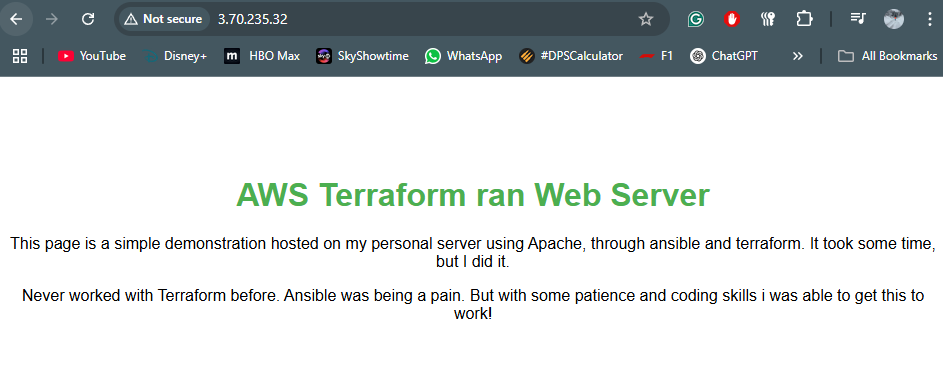
Ansible playbook

Afbeelding met tekst, schermopname, paars, violet

Door AI gegenereerde inhoud is mogelijk onjuist.





apache 2 web server running on my ubuntu server using ansible and terraform web page copied from classmate.