Fatima Jinnah Women University

*Department of Software Engineering*



**SUBJECT: CC LAB**

**SUBMITTED TO: SIR SHOAIB**

**SUBMITTED BY: ZUNAIRA NOOR**

**REGISTRATION NO: 2023-BSE-075**

**SEMESTER: V-B**

**Task 0 Lab Setup (Codespace & GH CLI)**

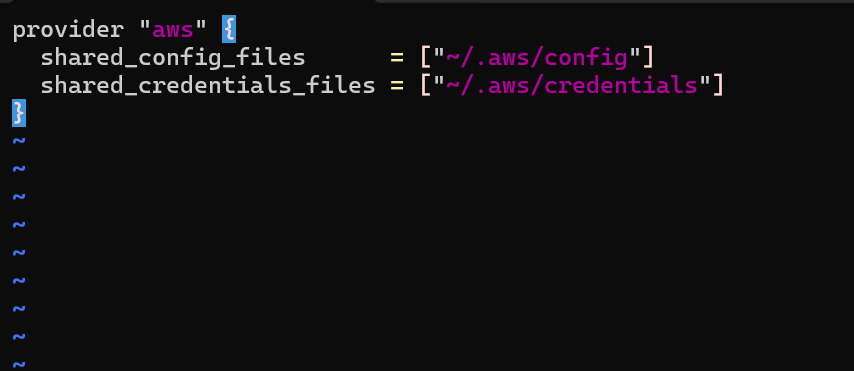


**Task 1 — Provider & Basic variable (variable precedence)**

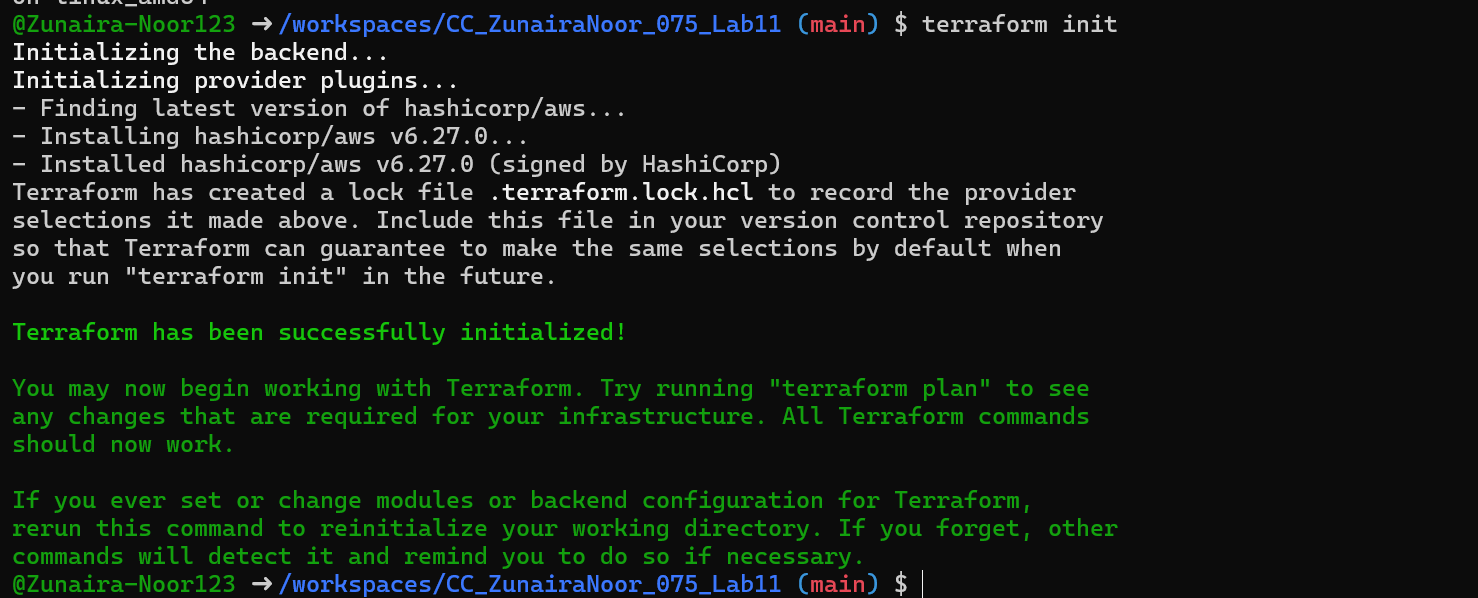
1. **In Codespace create main.tf:**

****

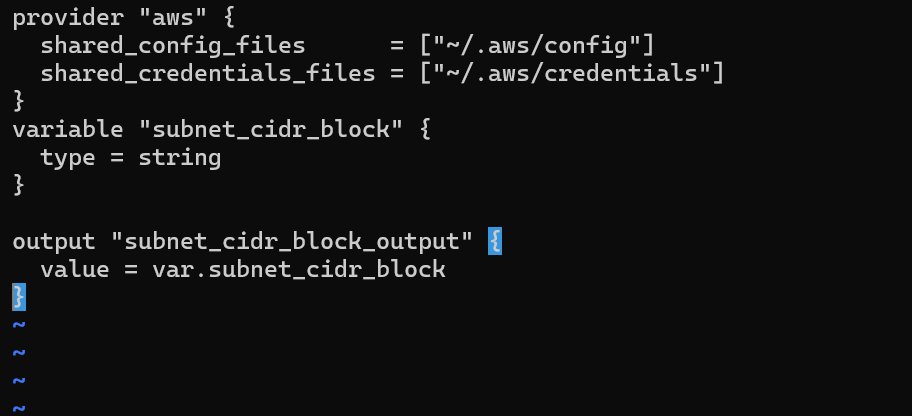
1. **Edit main.tf and add provider:**



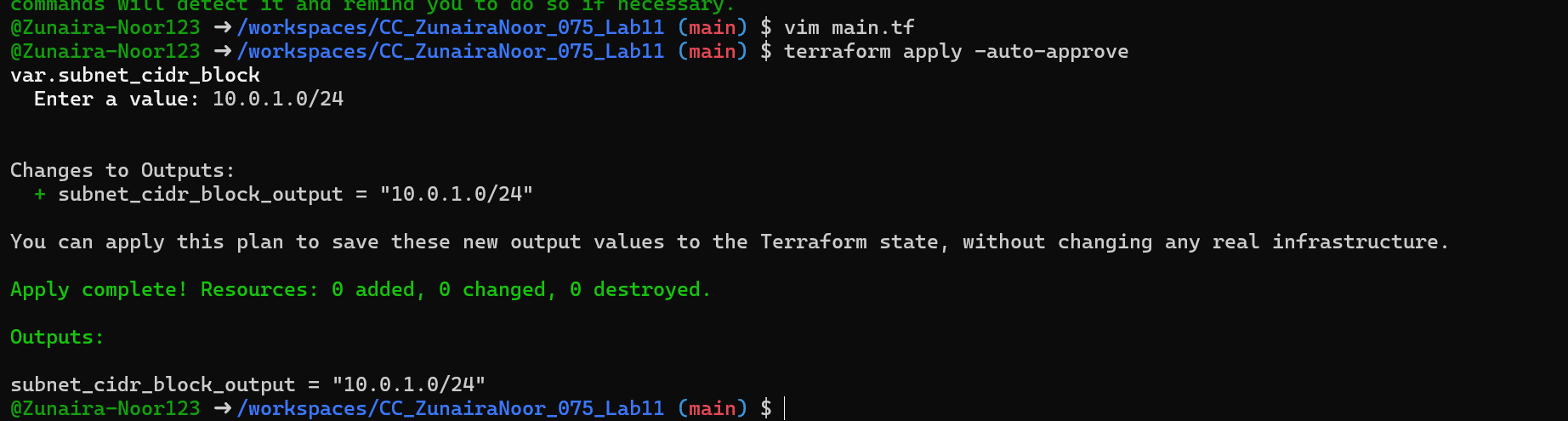
1. Initialize:



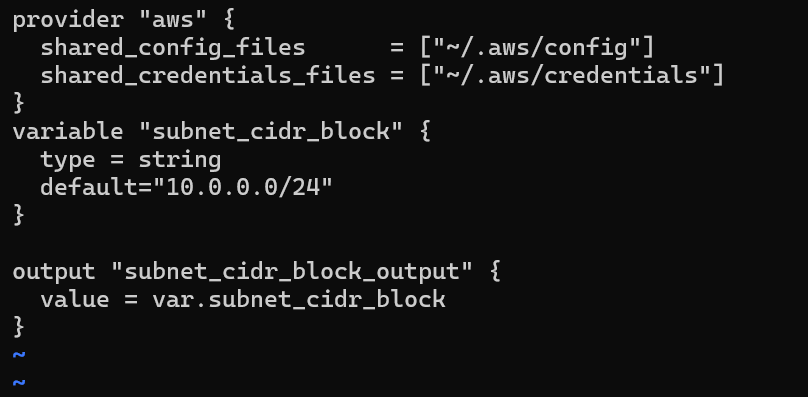
1. Define a variable and output:



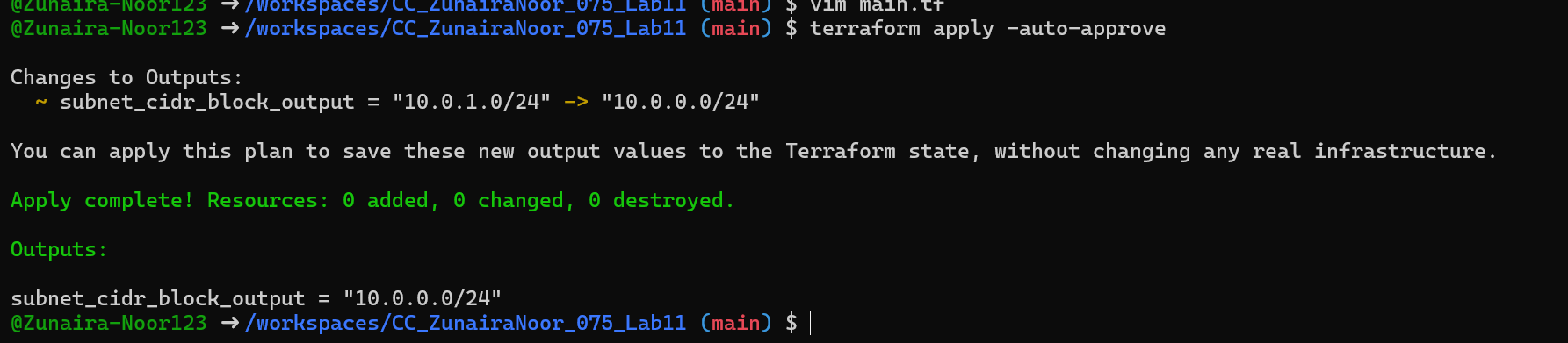
1. Run apply (first time without defaults):



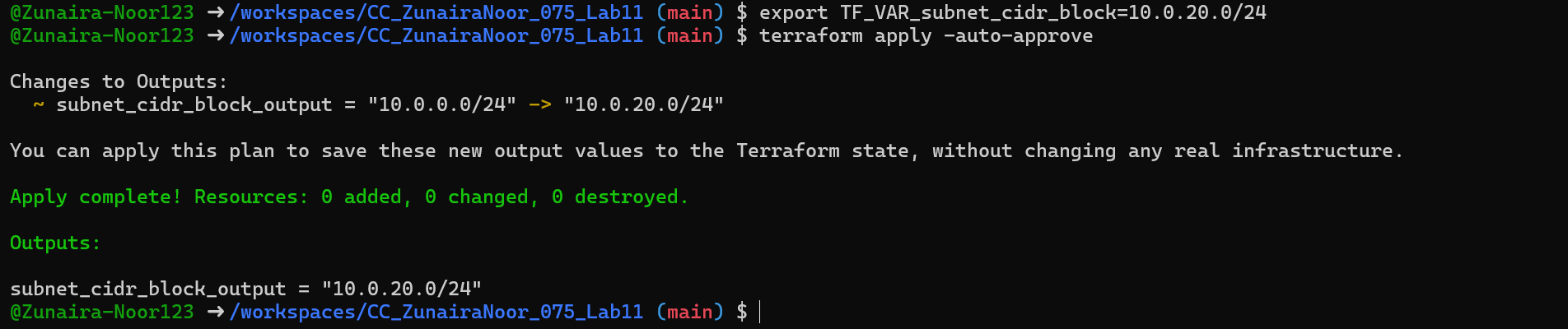
1. Add a default to the variable:



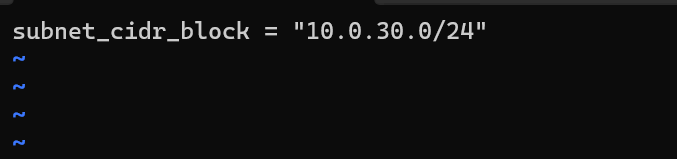
Run:

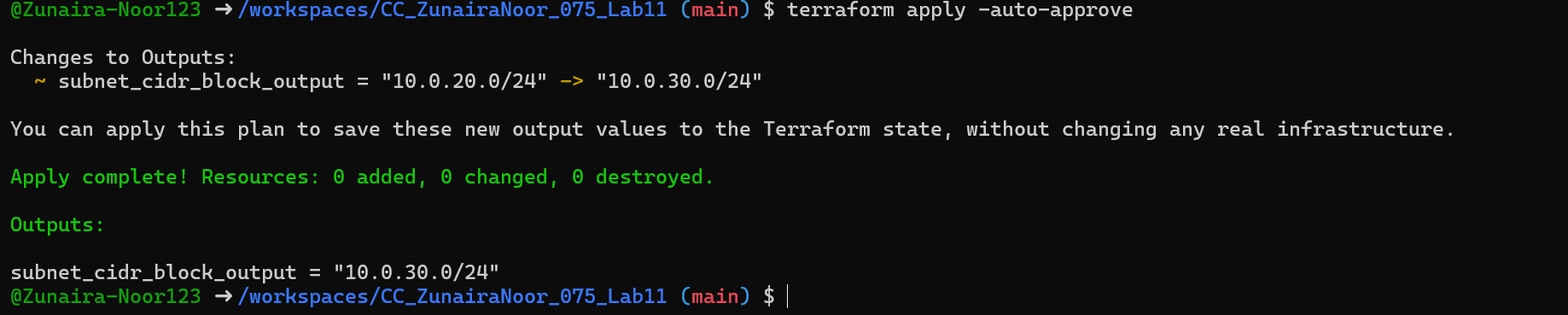


1. Export environment variable in Codespace shell:

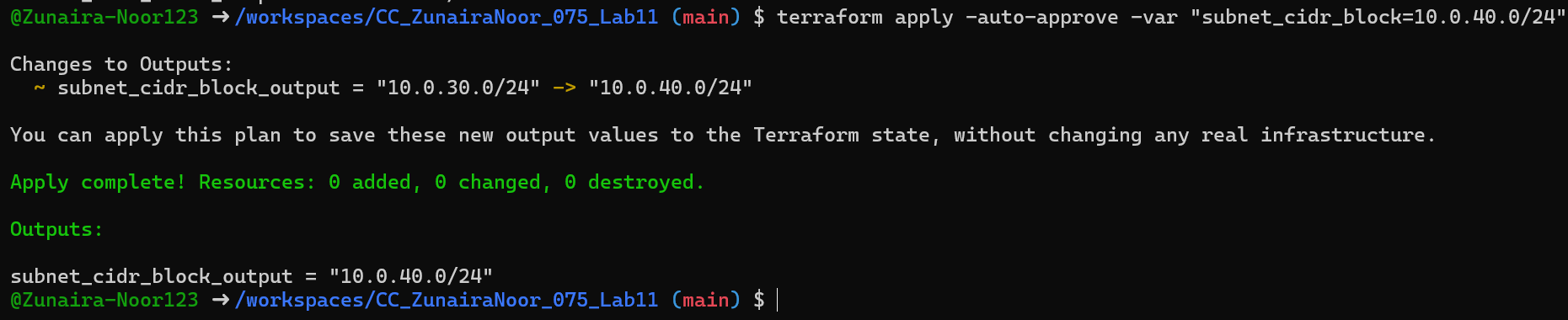


1. Create terraform.tfvars overriding values:

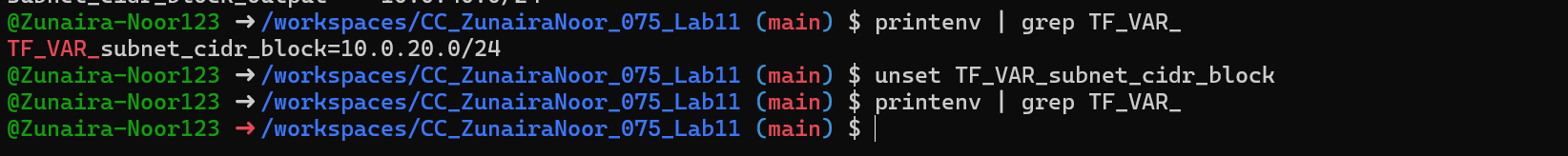




1. Override with -var:

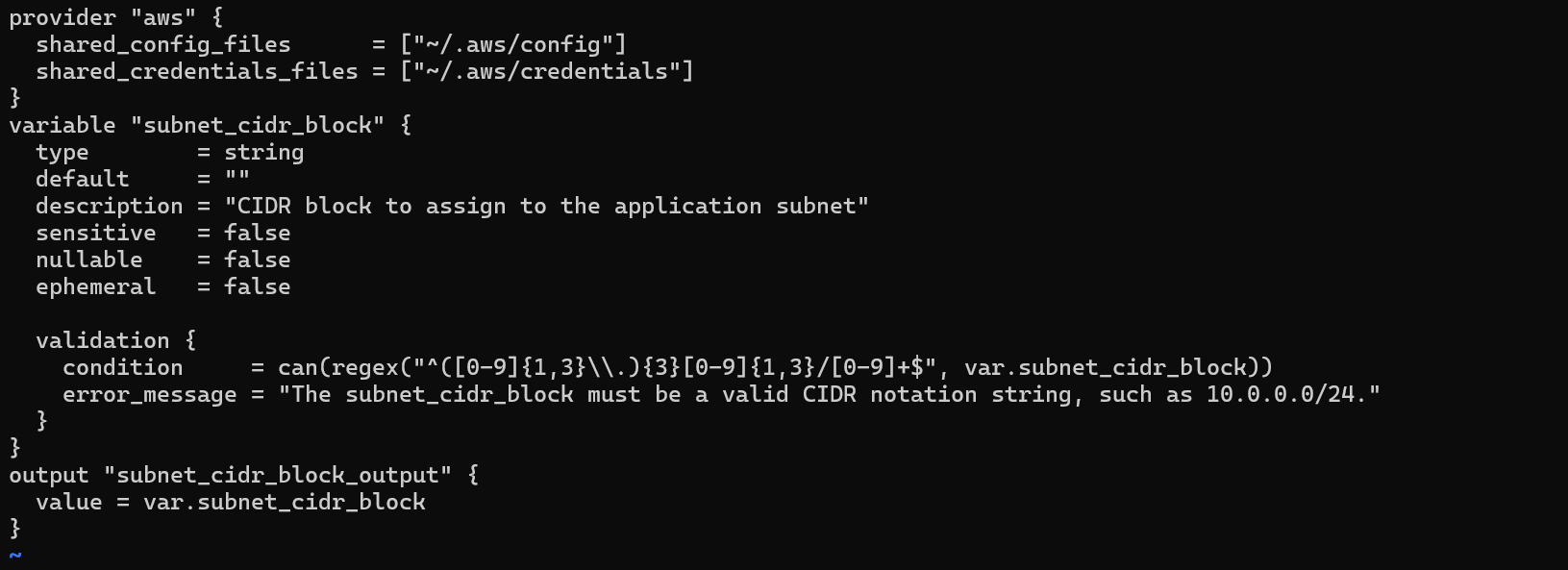


1. Show and unset env var:

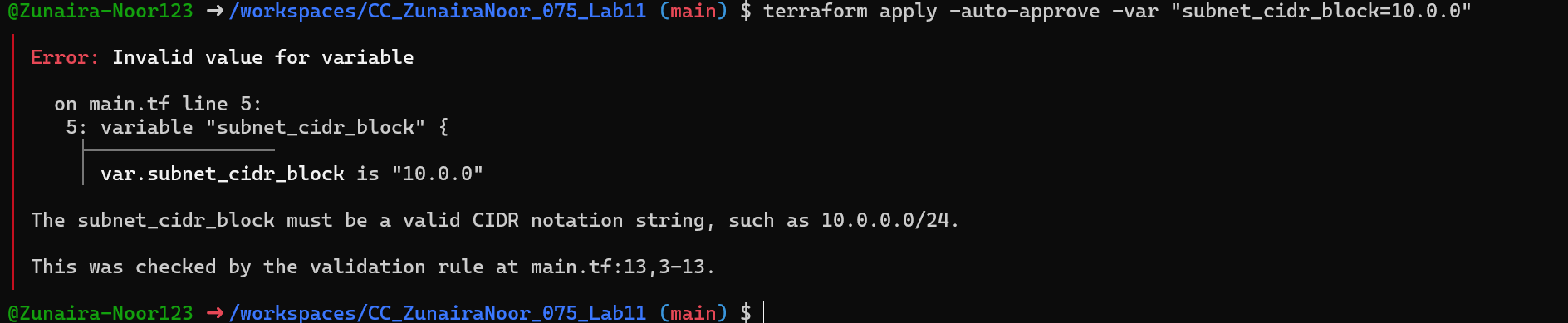


**Task 2 — Variable validation & sensitive / ephemeral variables**

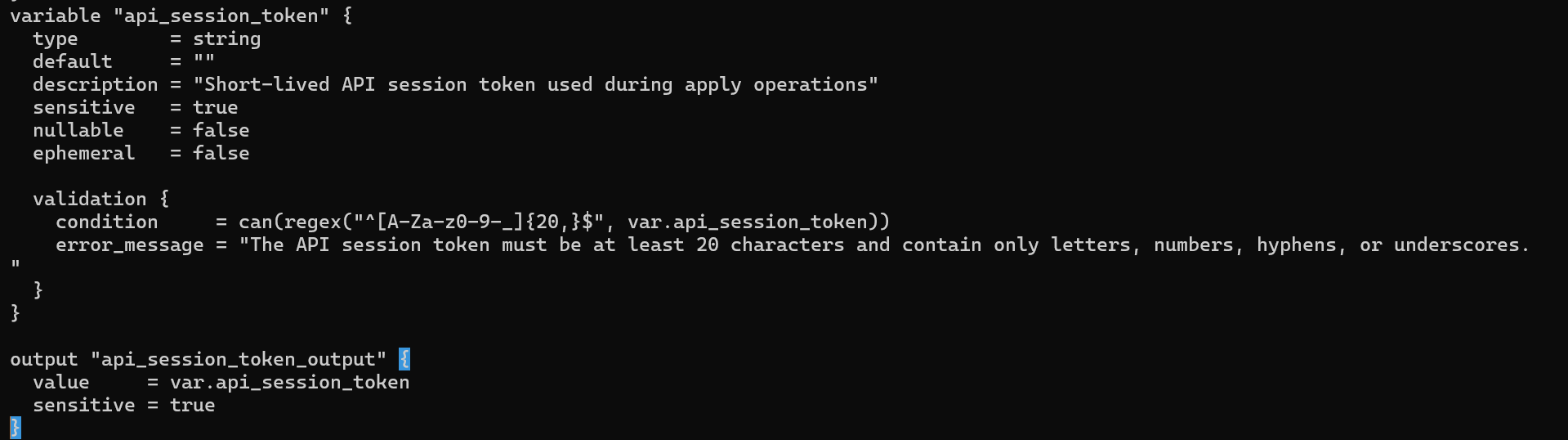
1. Replace subnet\_cidr\_block variable with this (validation included):



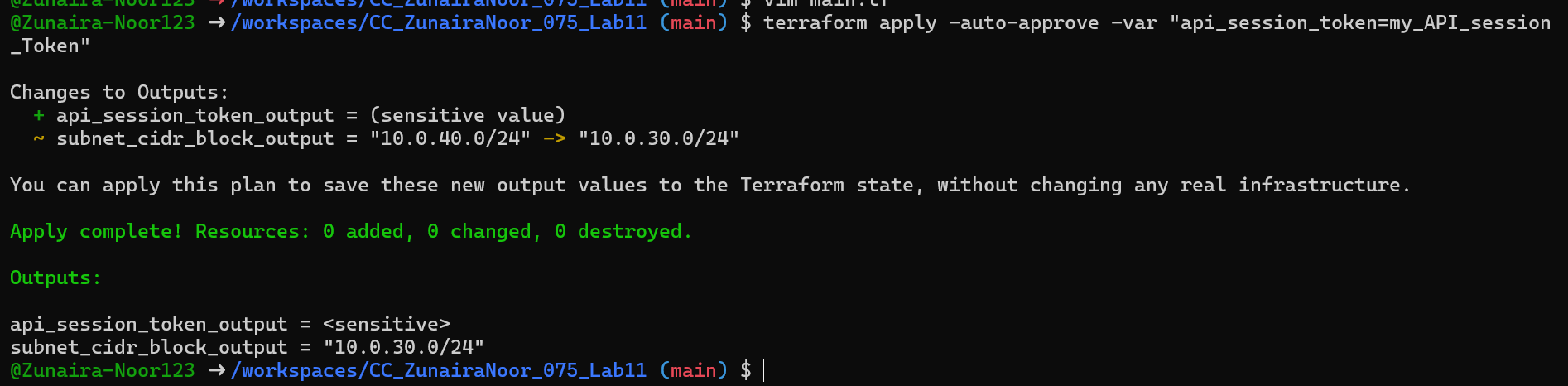
1. Test validation failure:



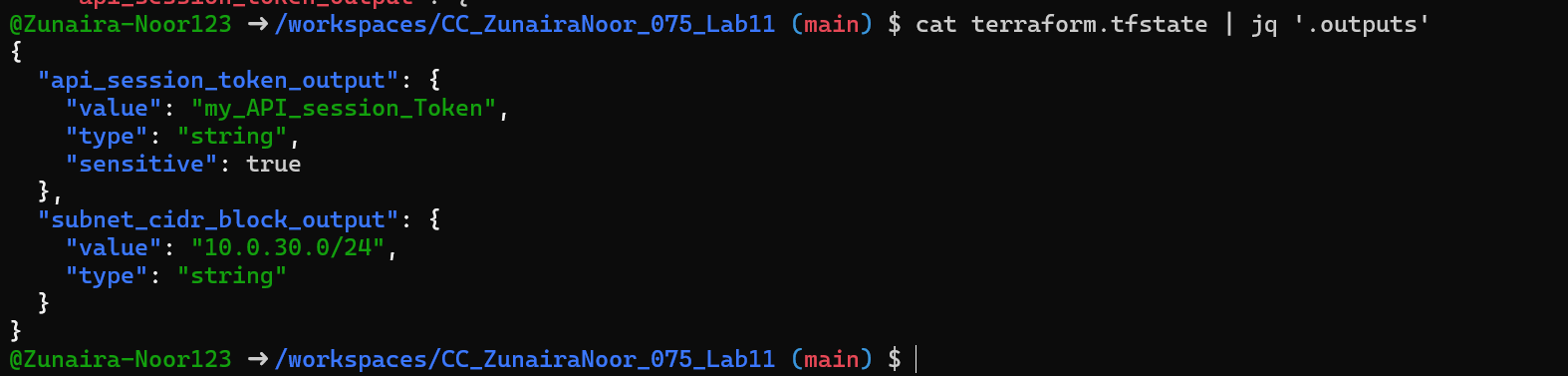
1. Create a sensitive variable api\_session\_token and output (sensitive):



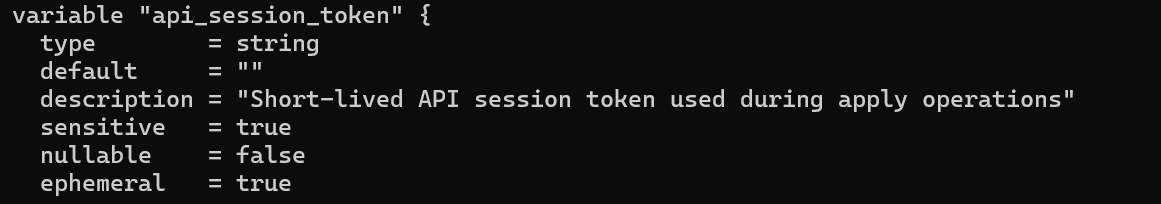
1. Run with -var to observe sensitive output behavior:

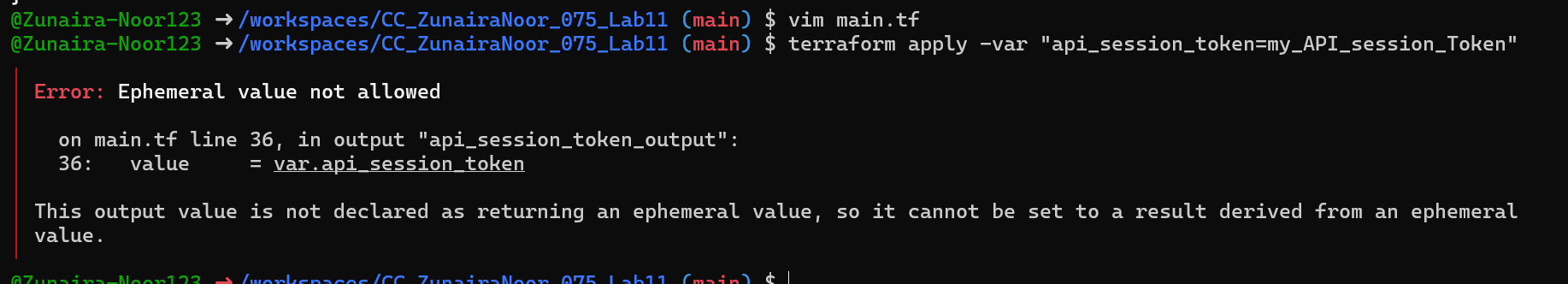


1. Check terraform.state for the sensitive output:

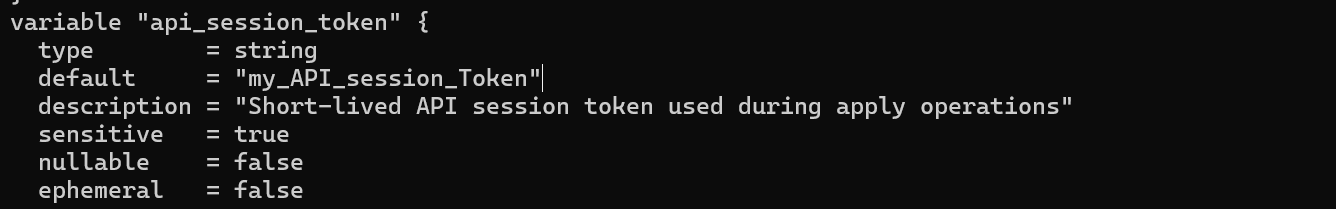


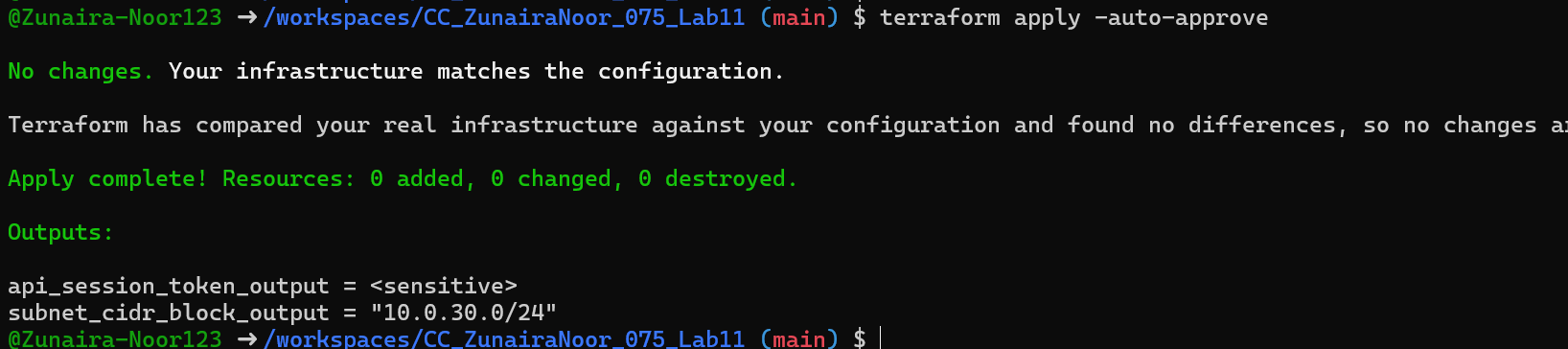
1. Make variable ephemeral to hide from state:





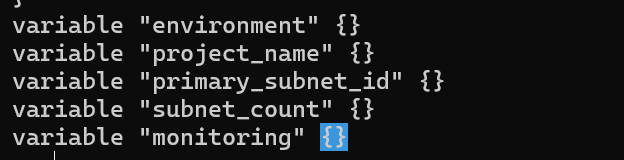
1. Set default to test local default:



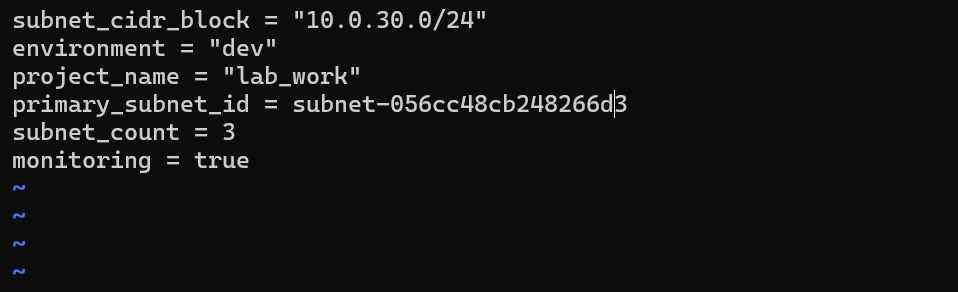


**Task 3 — Project-level variables, locals, and outputs**

1. **Add variables to main.tf:**

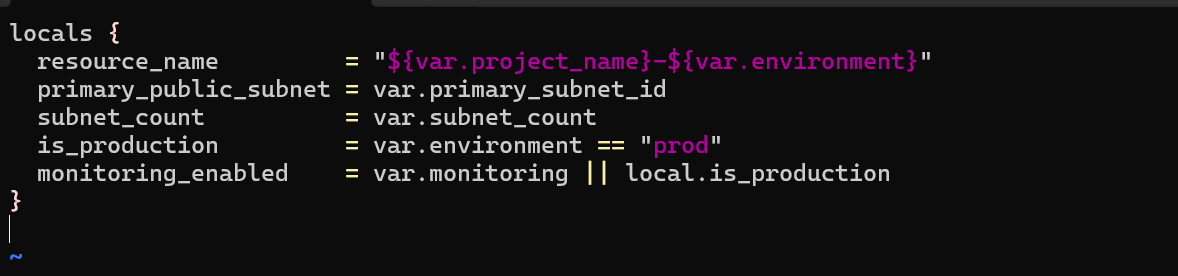


1. Populate terraform.tfvars *after* discovering actual subnet id for availability zone me-central-1a:

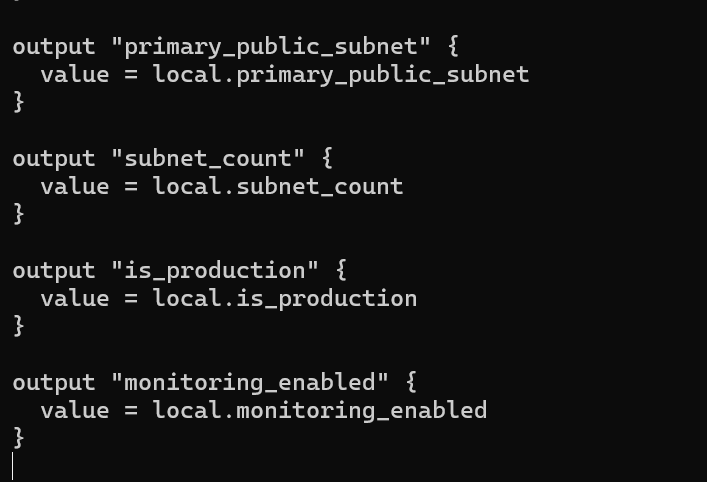




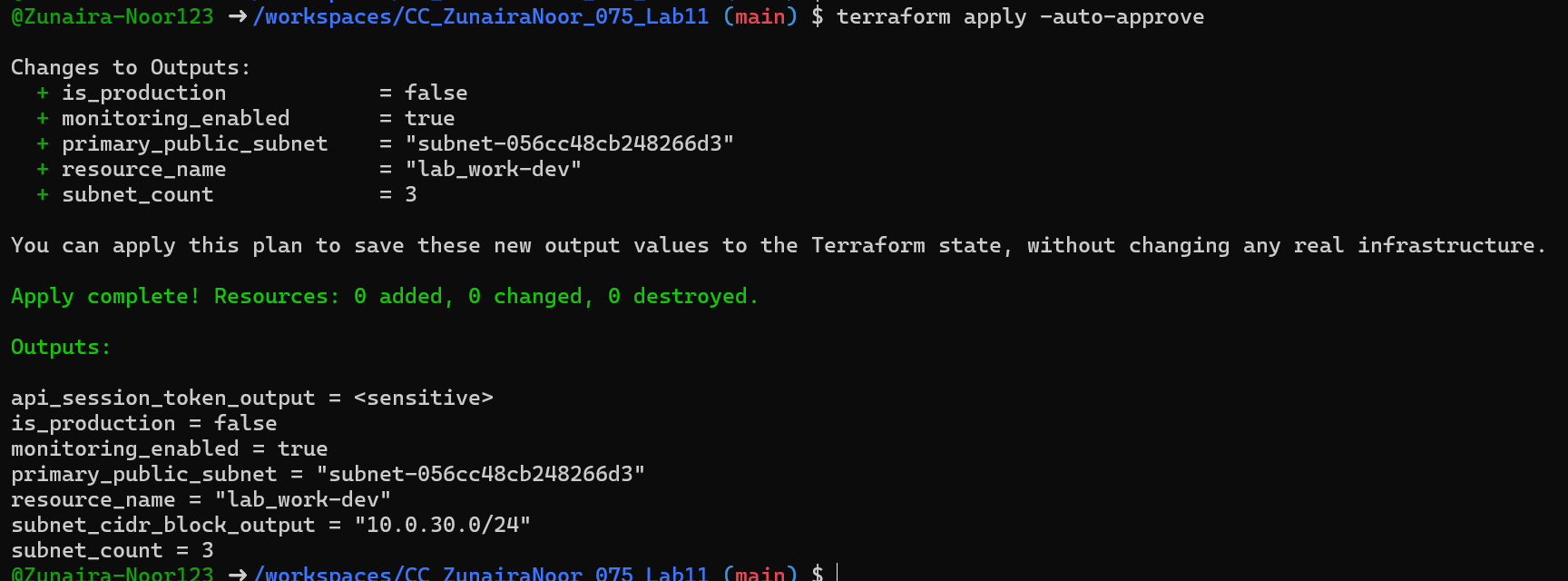
1. Create locals.tf with:



1. Add outputs to main.tf:

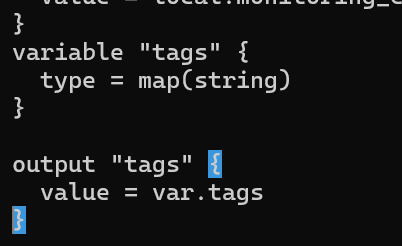


* Run:



**Task 4 — Maps and Objects**

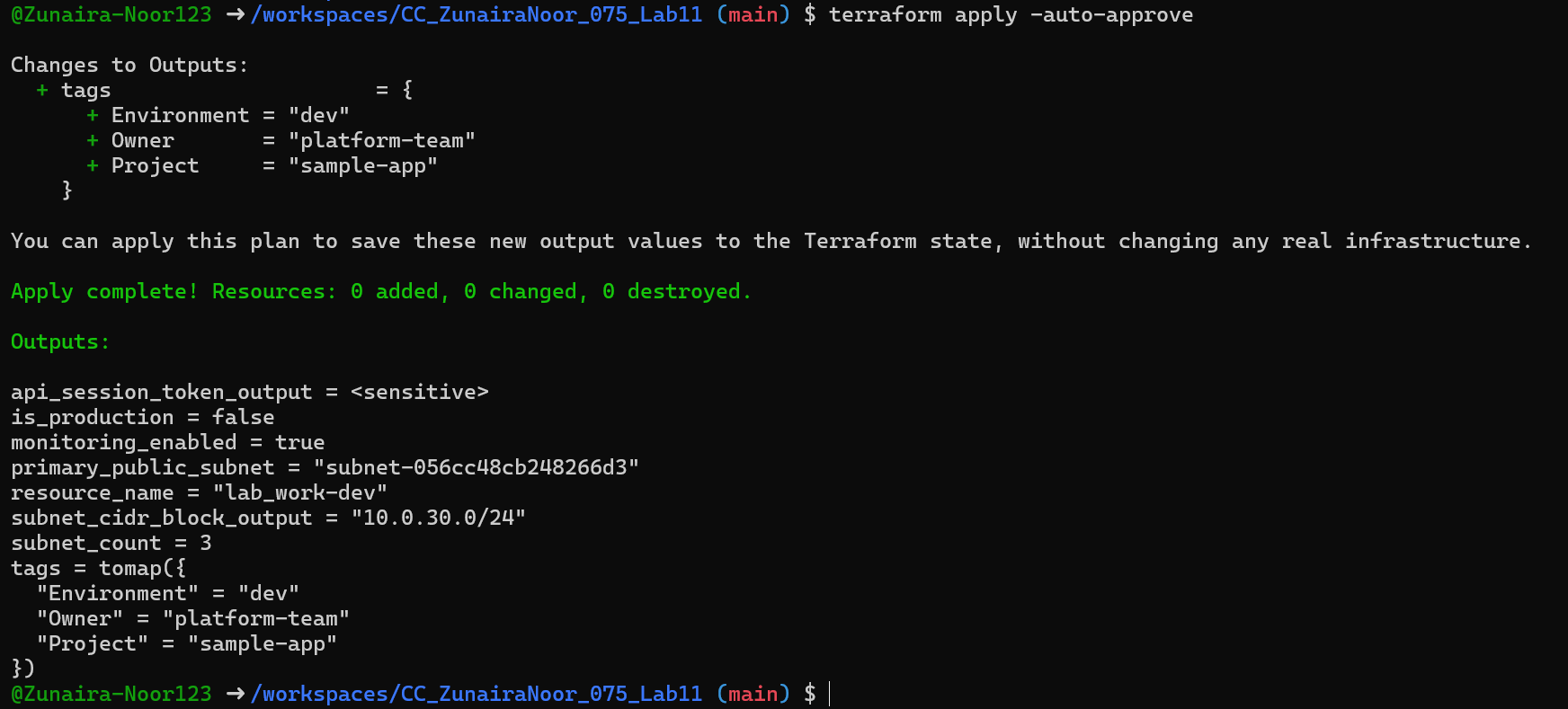
1. Map variable in main.tf:



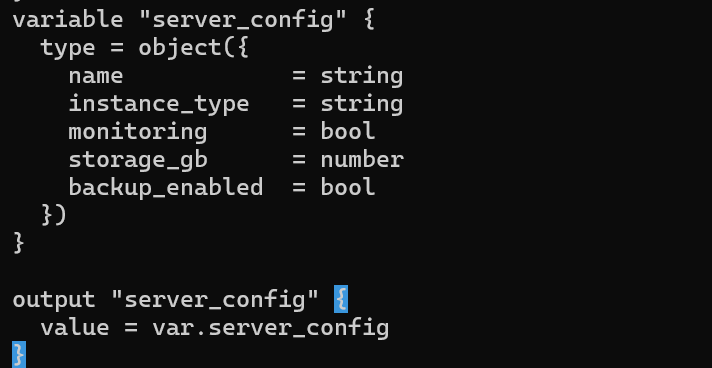
1. In terraform.tfvars:

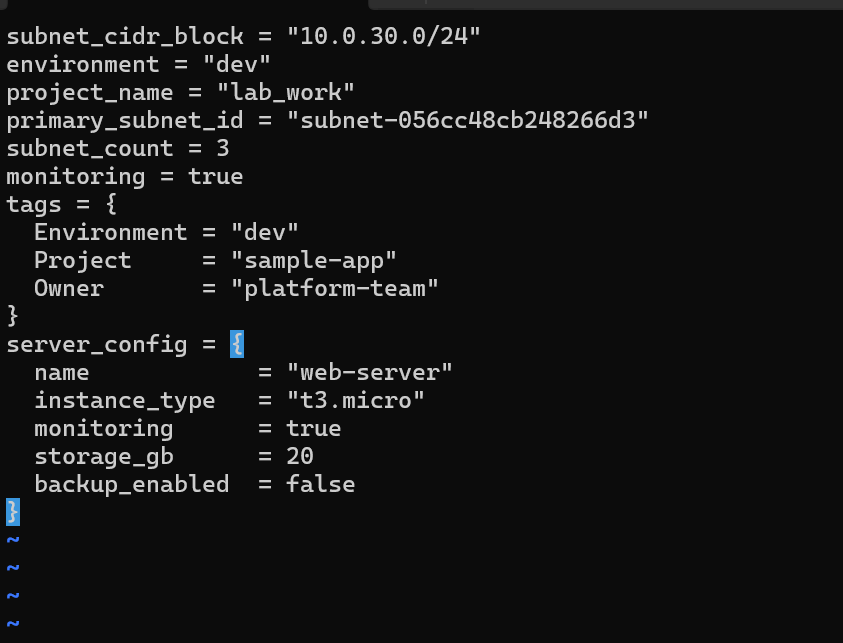


terraform apply -auto-approve

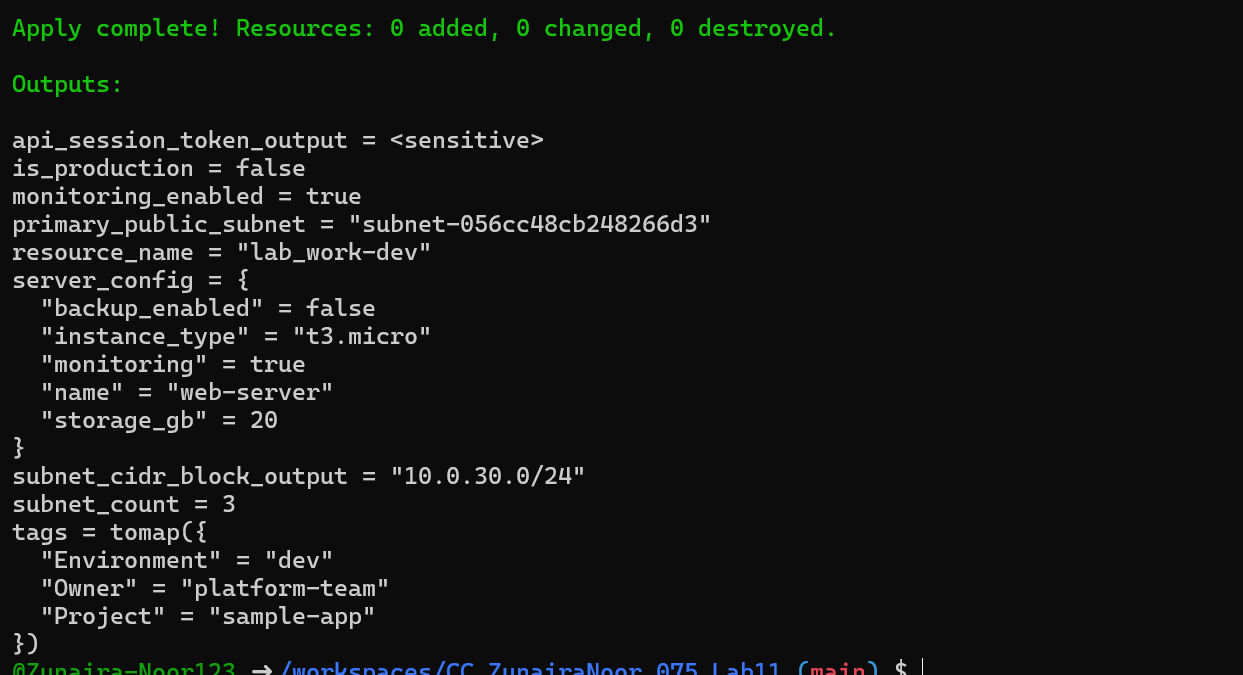


1. Define object variable:

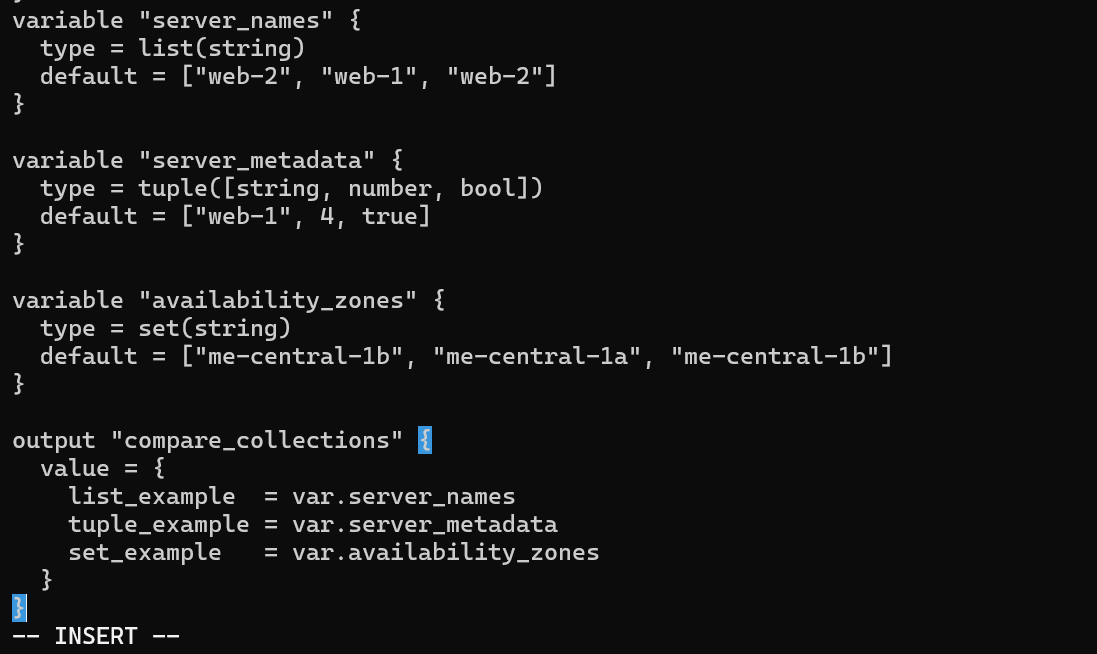




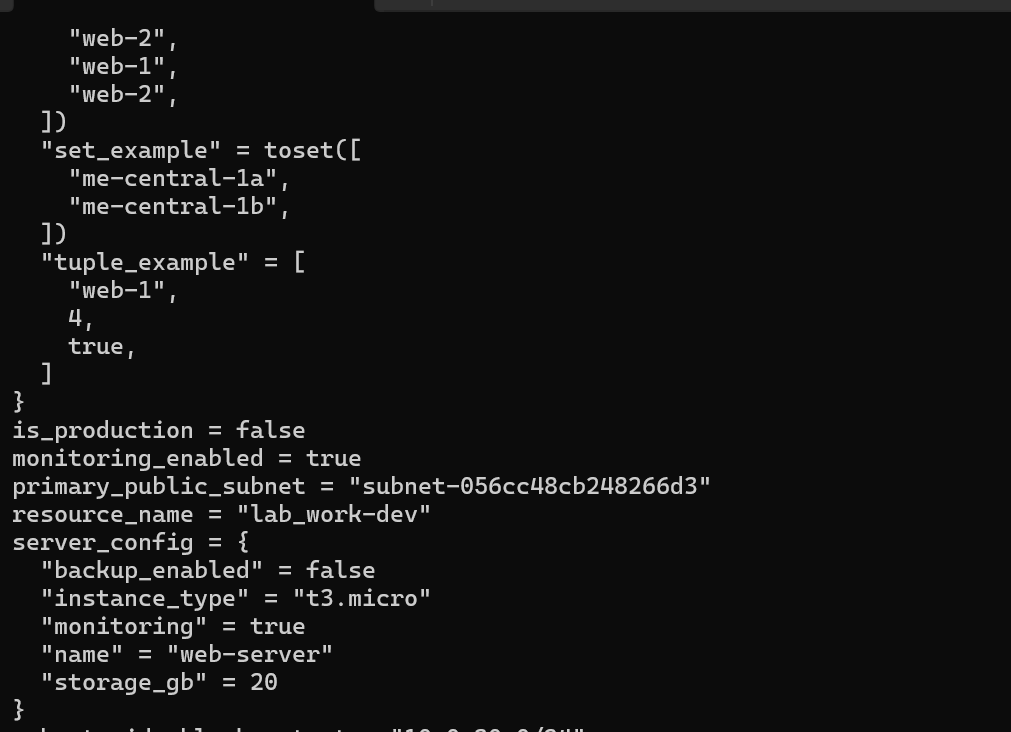
Output showing server\_config object.

****

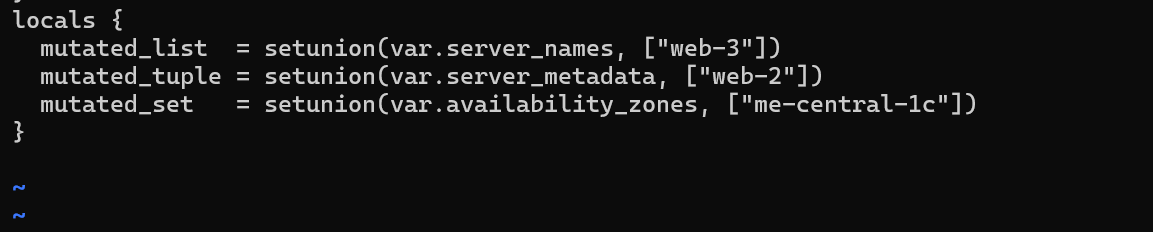
**Task 5 — Collections: list, tuple, set & mutation via locals**

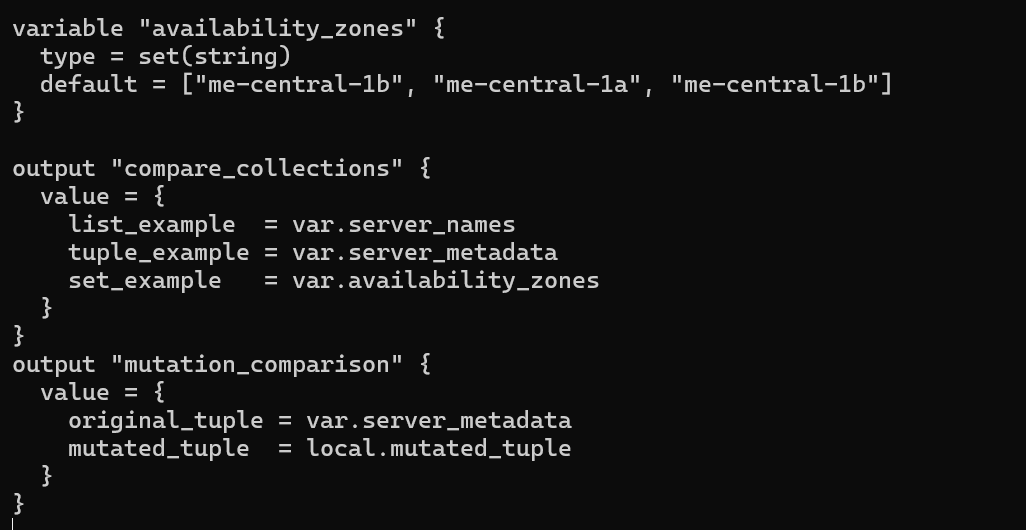
****

1. **Run:**

****

**In locals.tf add mutations: Create or edit locals.tf and add the following locals to demonstrate mutation behavior**

****

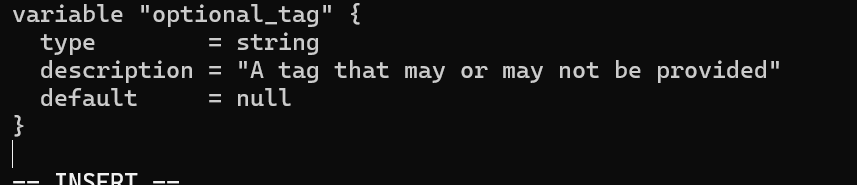
****

**Apply Terraform**

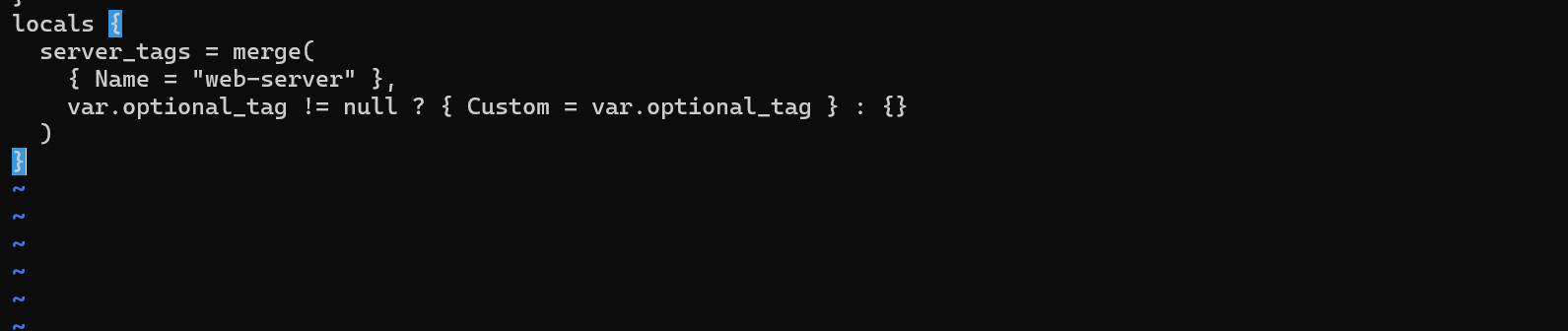
****

**Task 6 — Null, any type & dynamic values**

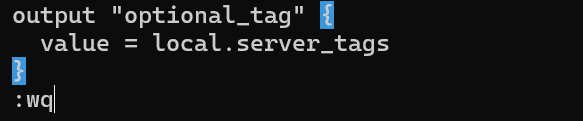
1. **Null variable:**

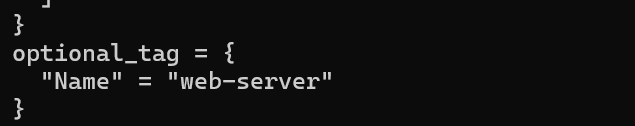
****

1. **Merge tags in locals.tf:**

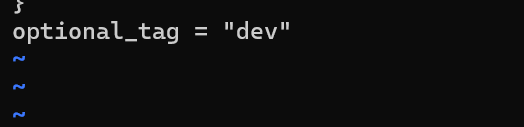
****

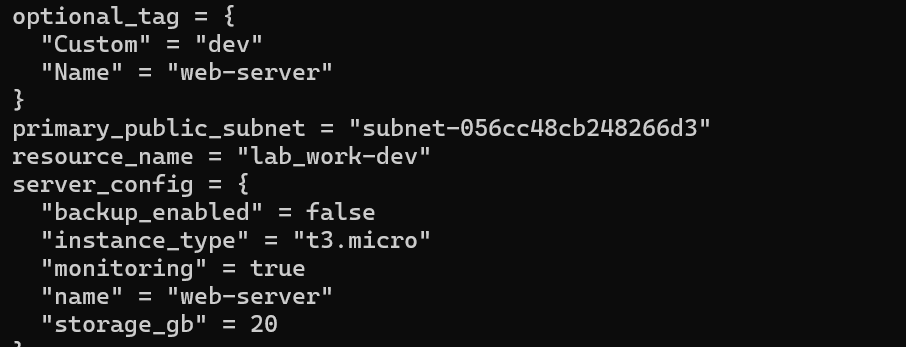
1. **Output:**

****

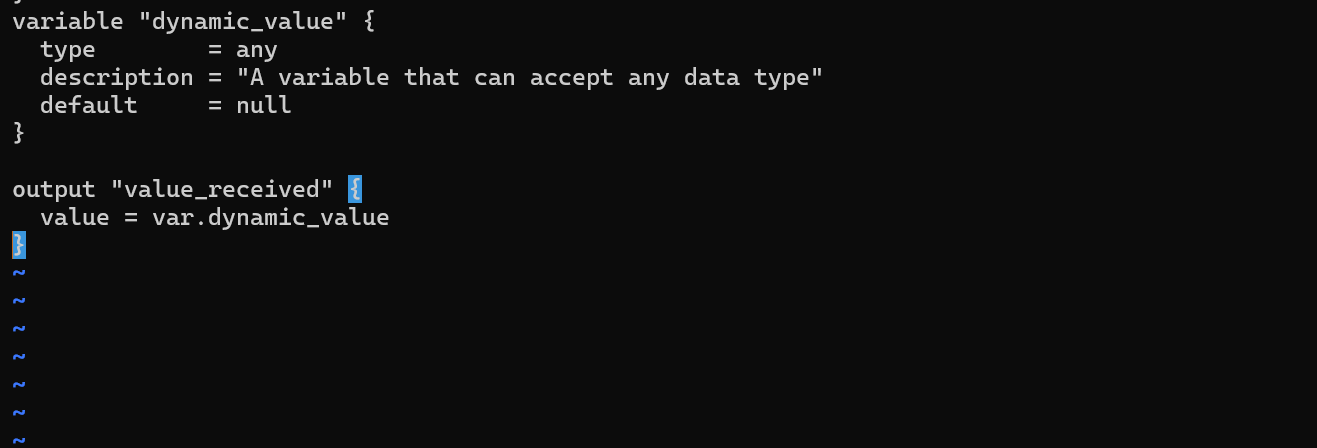
****

1. **Add in terraform.tfvars:**

****

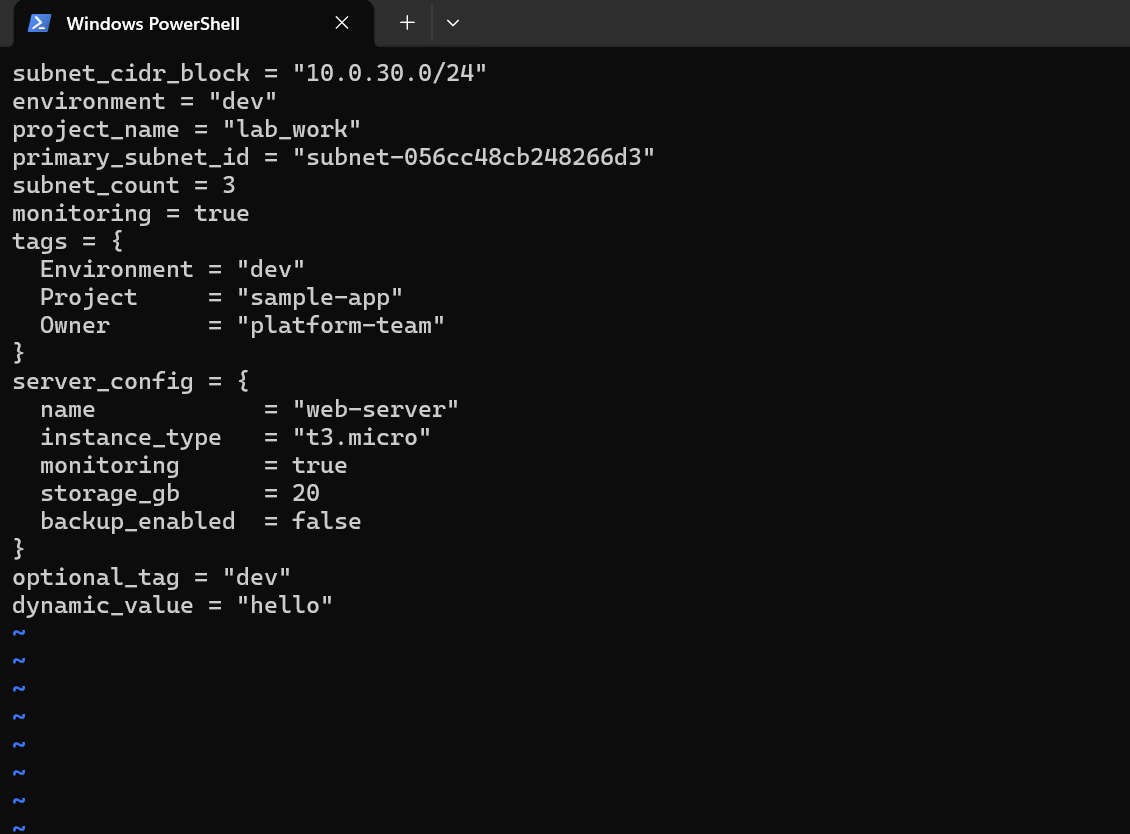
****

1. **Any type variable:**

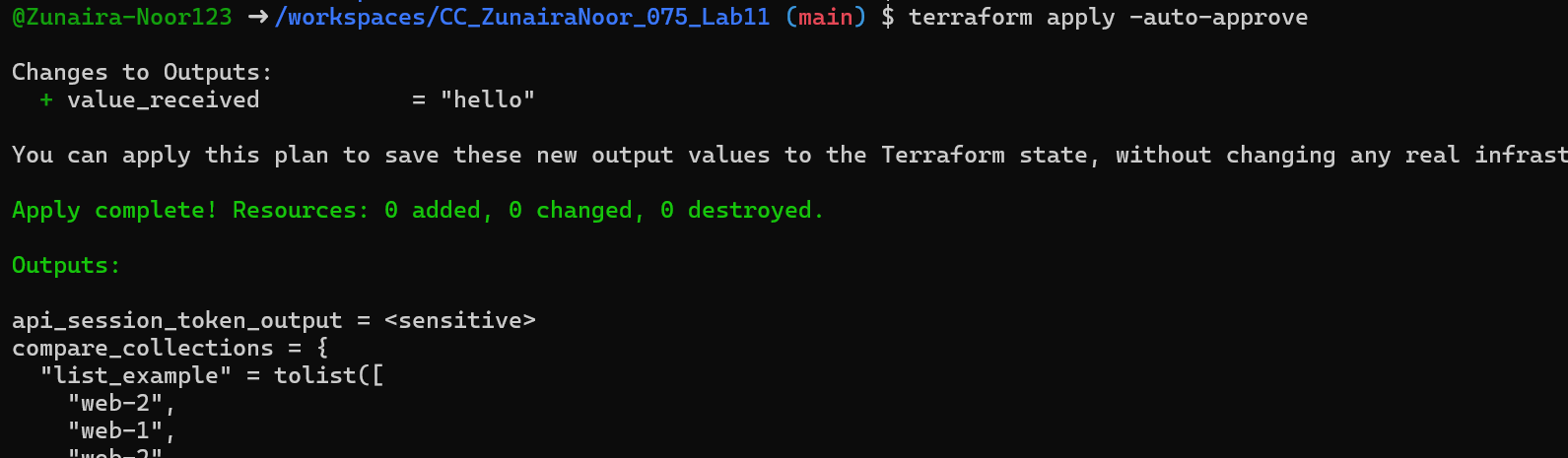
****

**a) String**

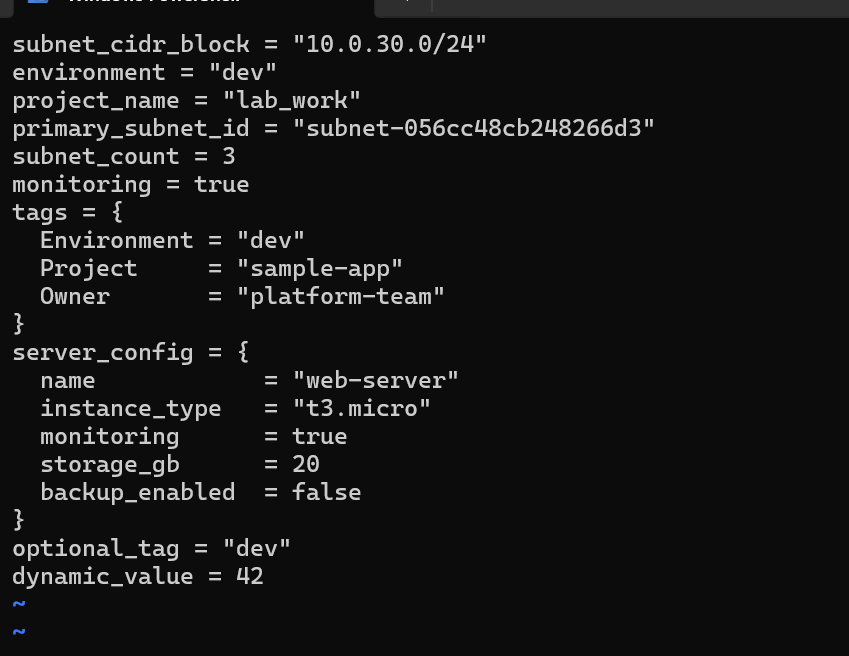
* **In terraform.tfvars:**

****

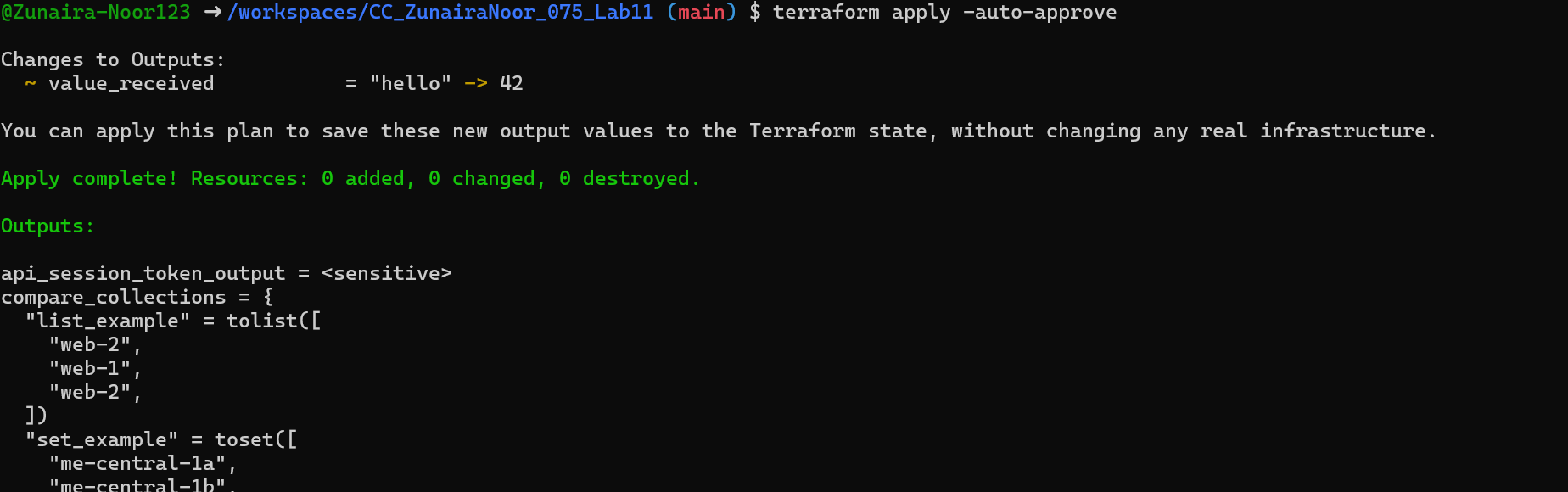
* **Run:**

****

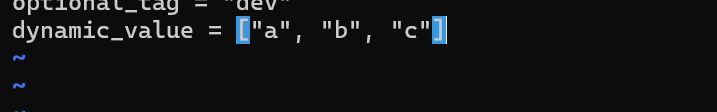
**b) Number**

****

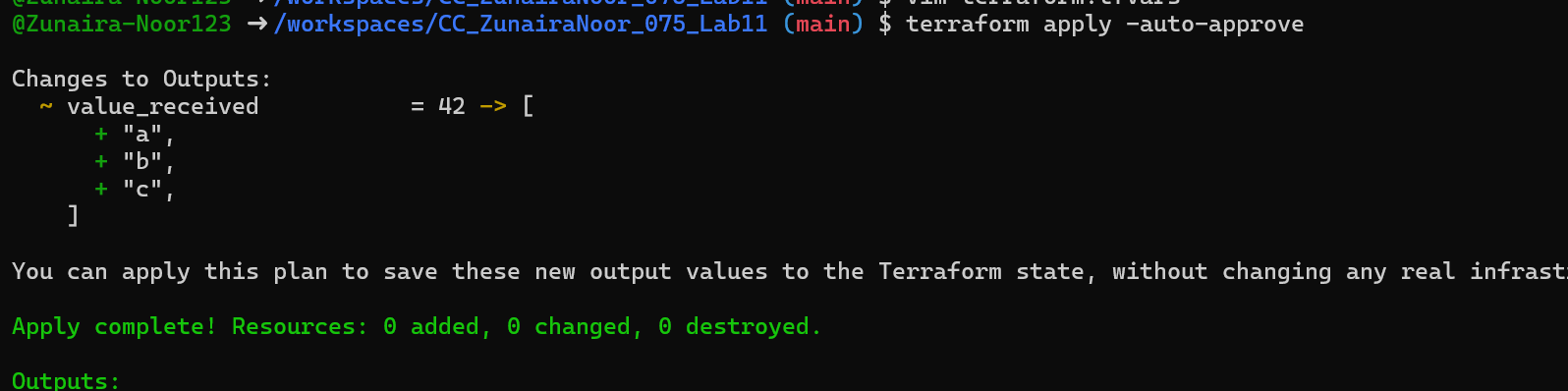
* **Run:**

****

**c) List**

****

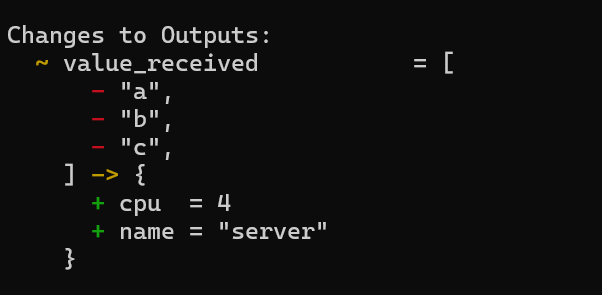
* **Run:**

****

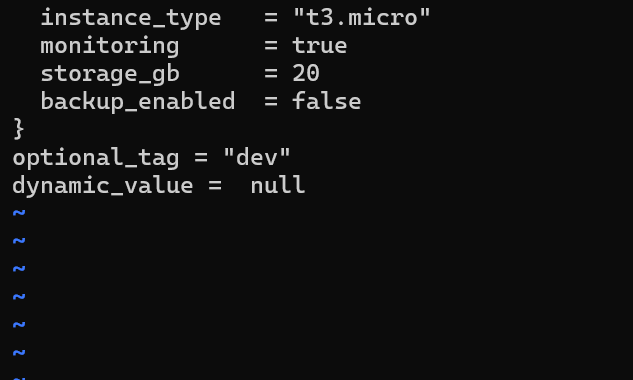
**d) Map / Object**

****

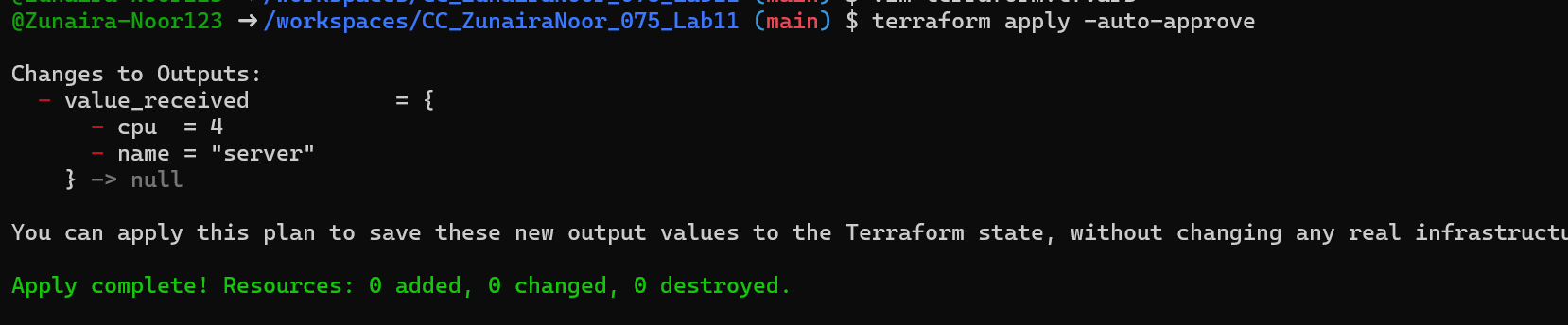
* **Run:**

****

**e) Null**

****

* **Run:**

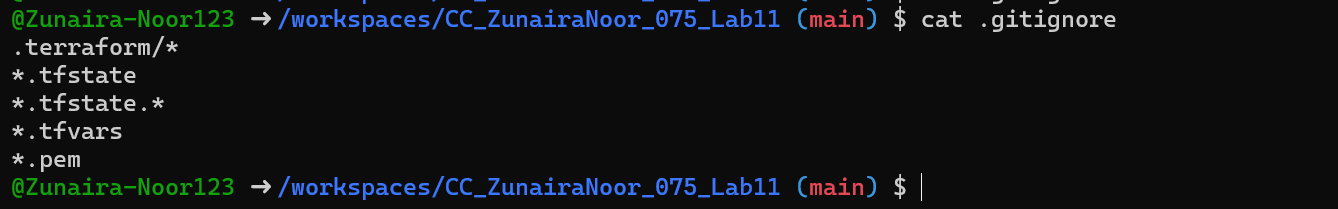
****

**Task 7 — Git ignore**

**Create .gitignore:**

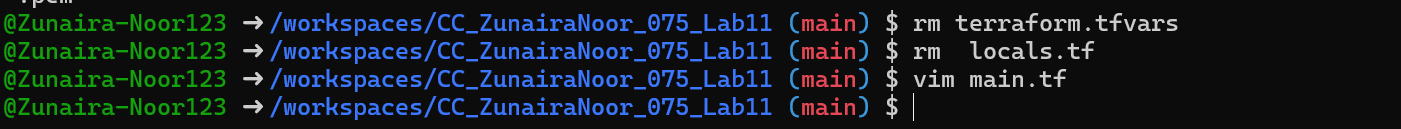
****

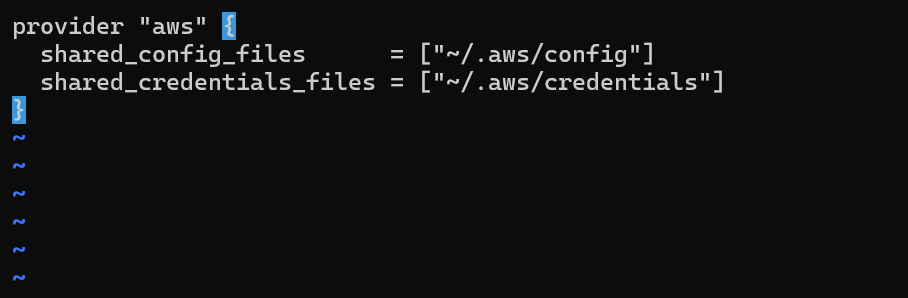
**.gitignore file content showing entries.**

****

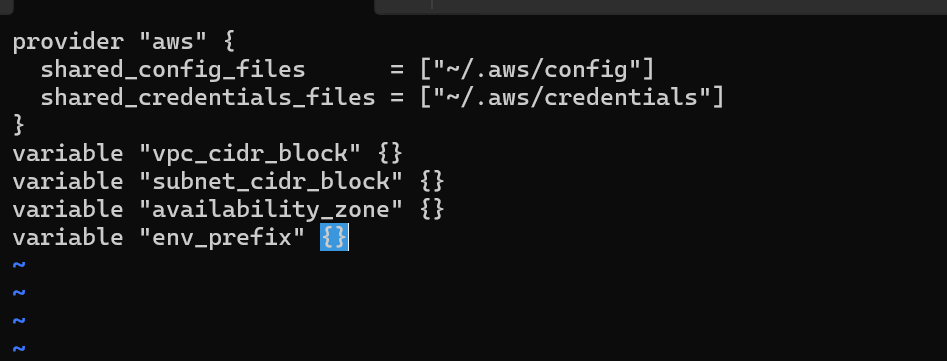
**Task 8 — Clean-up then build real infra**

1. **Clean previous files**

****

****

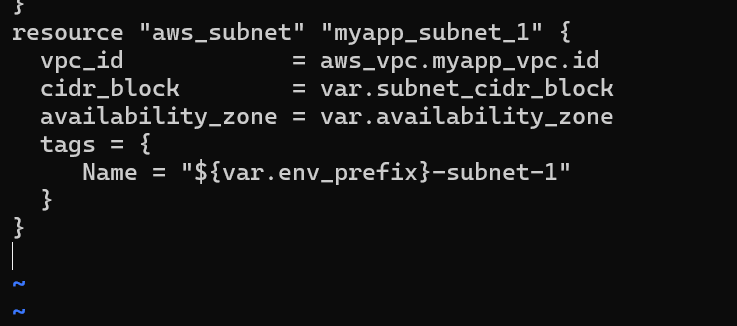
1. **Define variables in main.tf Add these variable declarations to main.tf (below the provider block):**

****

1. **Create VPC in main.tf Add the VPC resource to main.tf:**

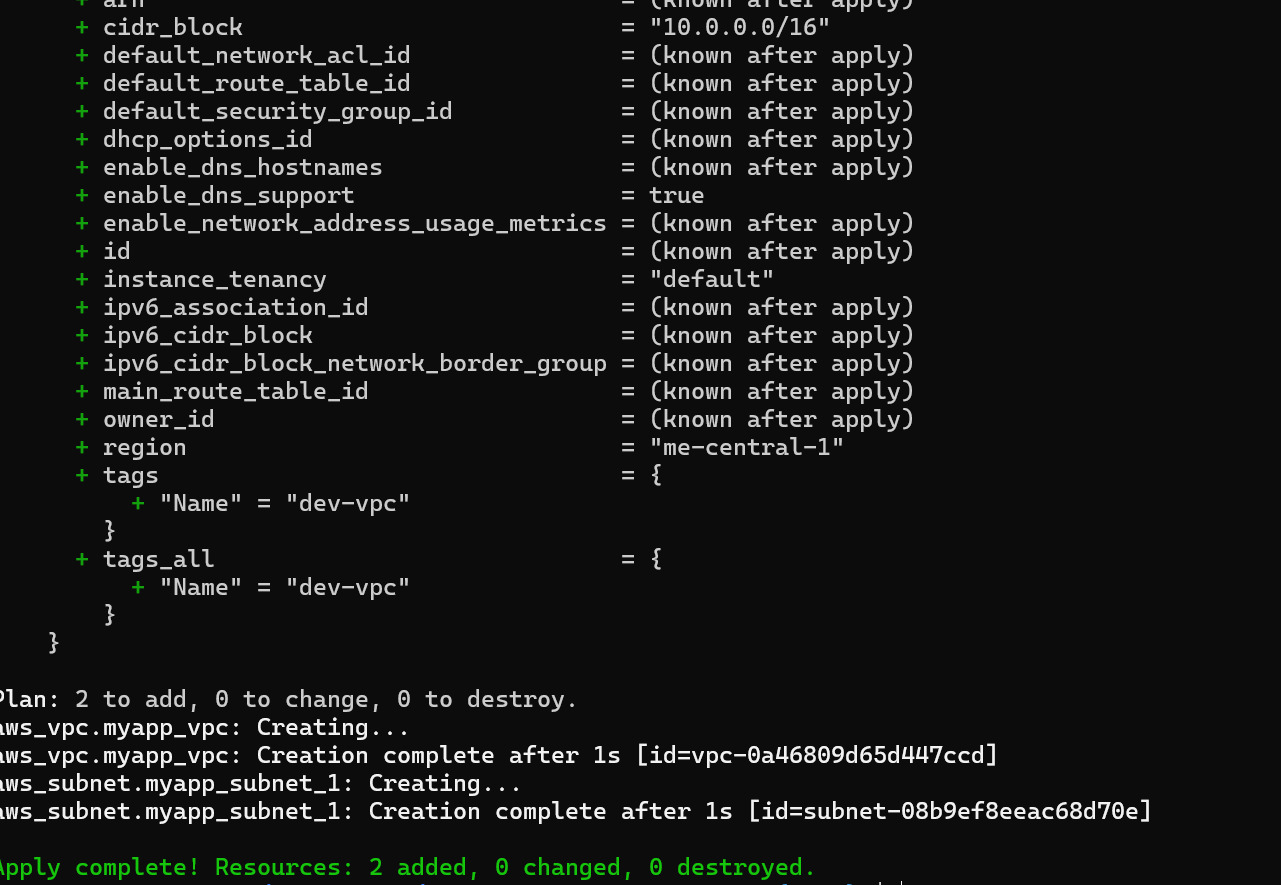
****

1. **Create Subnet in the VPC Add the subnet resource to main.tf:**

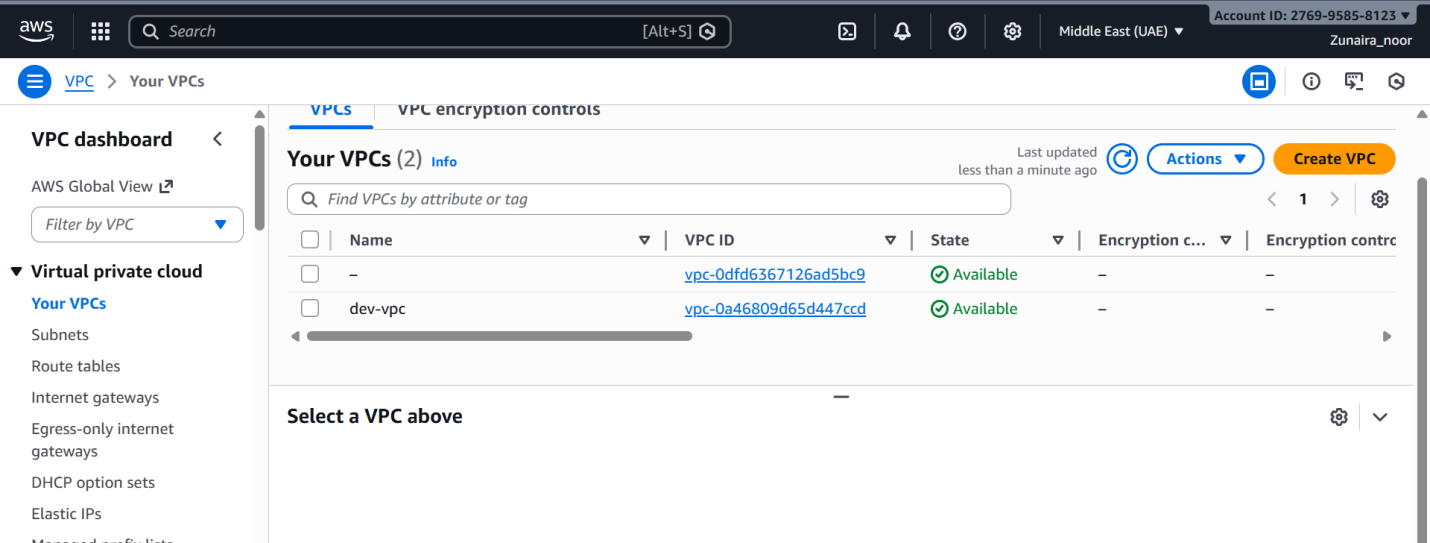
****

1. Populate terraform.tfvars In terraform.tfvars add:

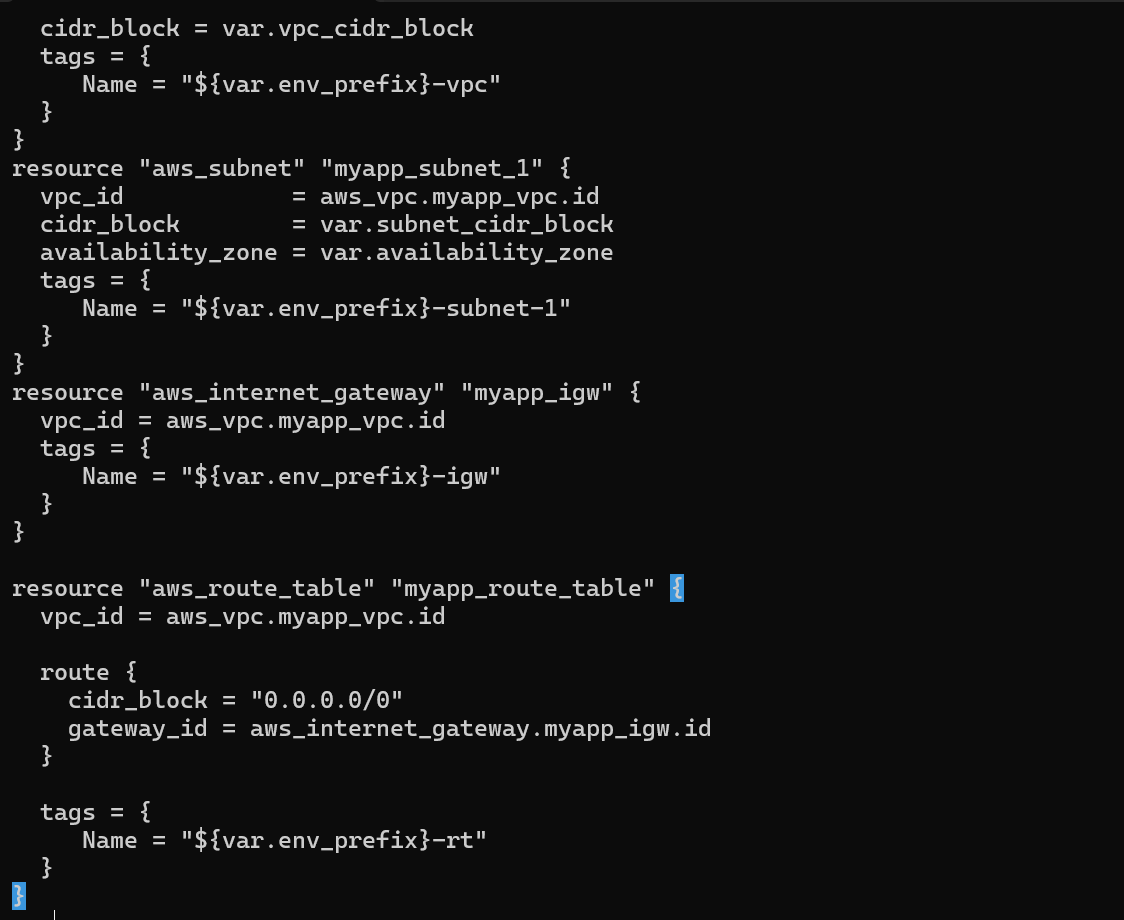


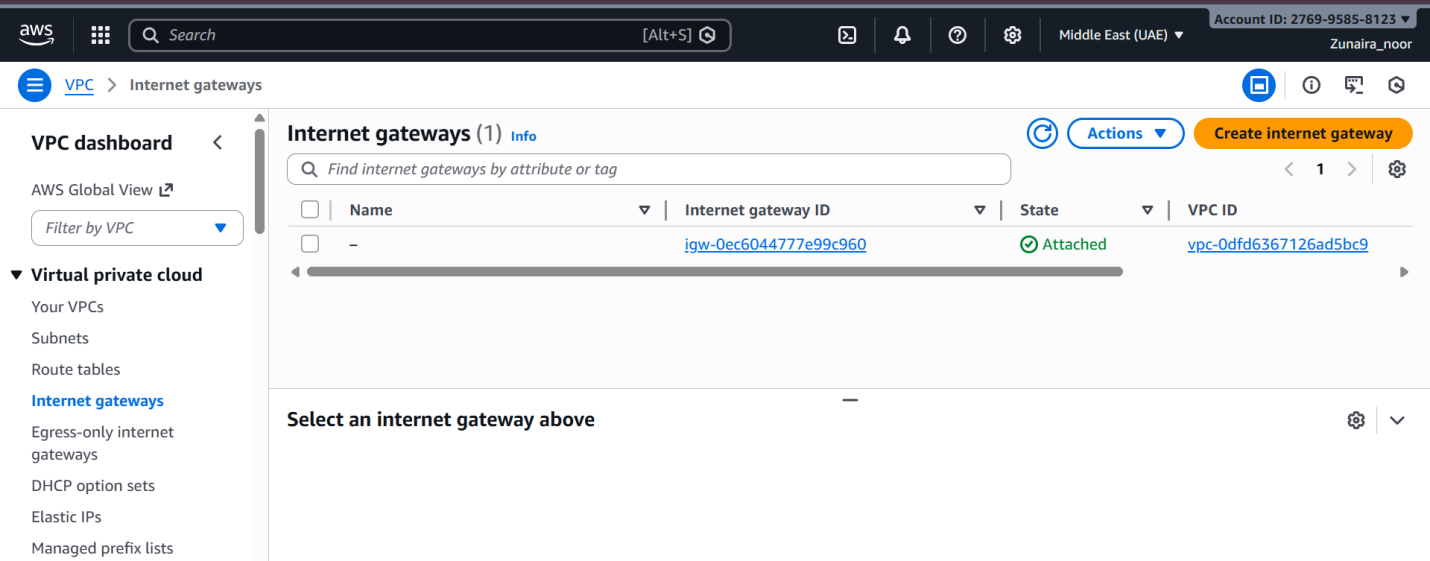


Verify in AWS Console that the VPC and Subnet were created.

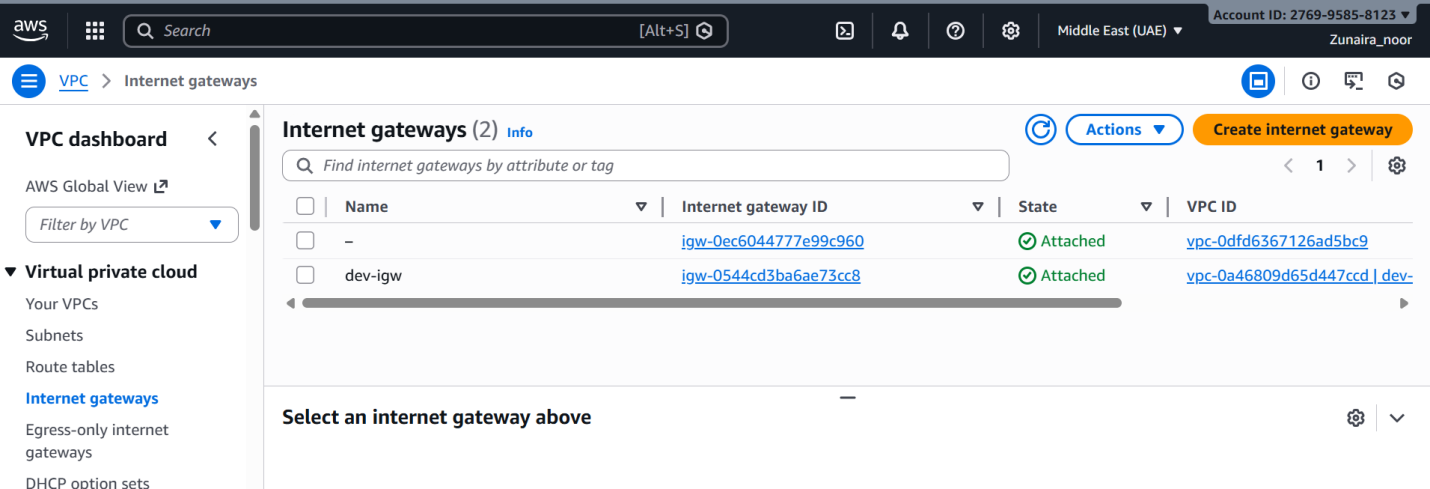
****

1. **Create Internet Gateway and Route Table (custom) Add the Internet Gateway and a custom Route Table to main.tf:**

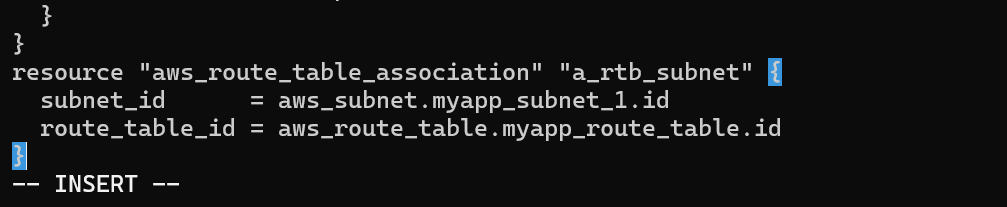
****

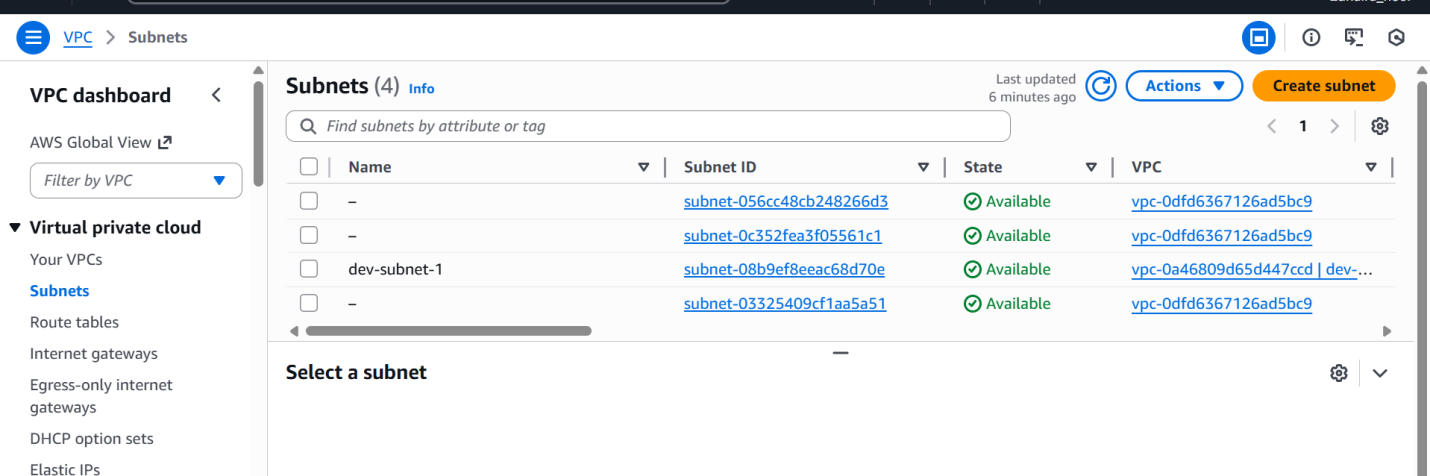
**AWS console showing IGW and route table resources (before apply).** ****

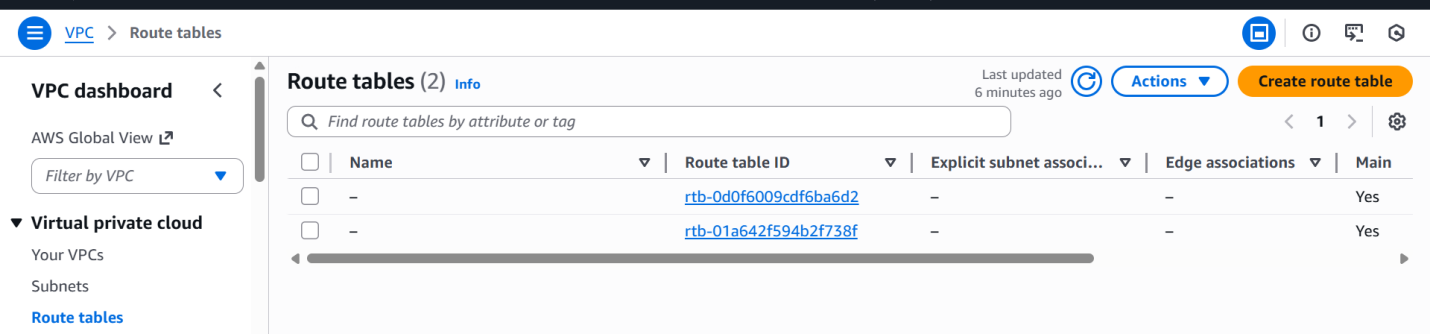
**Verify IGW and route table in AWS Console.**

****

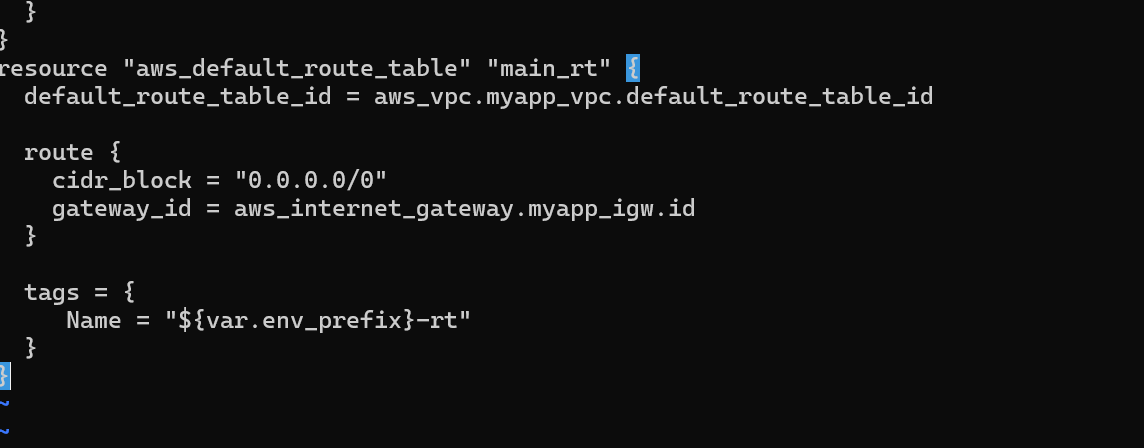
1. **Associate the Route Table with the Subnet Add the association resource to main.tf:**

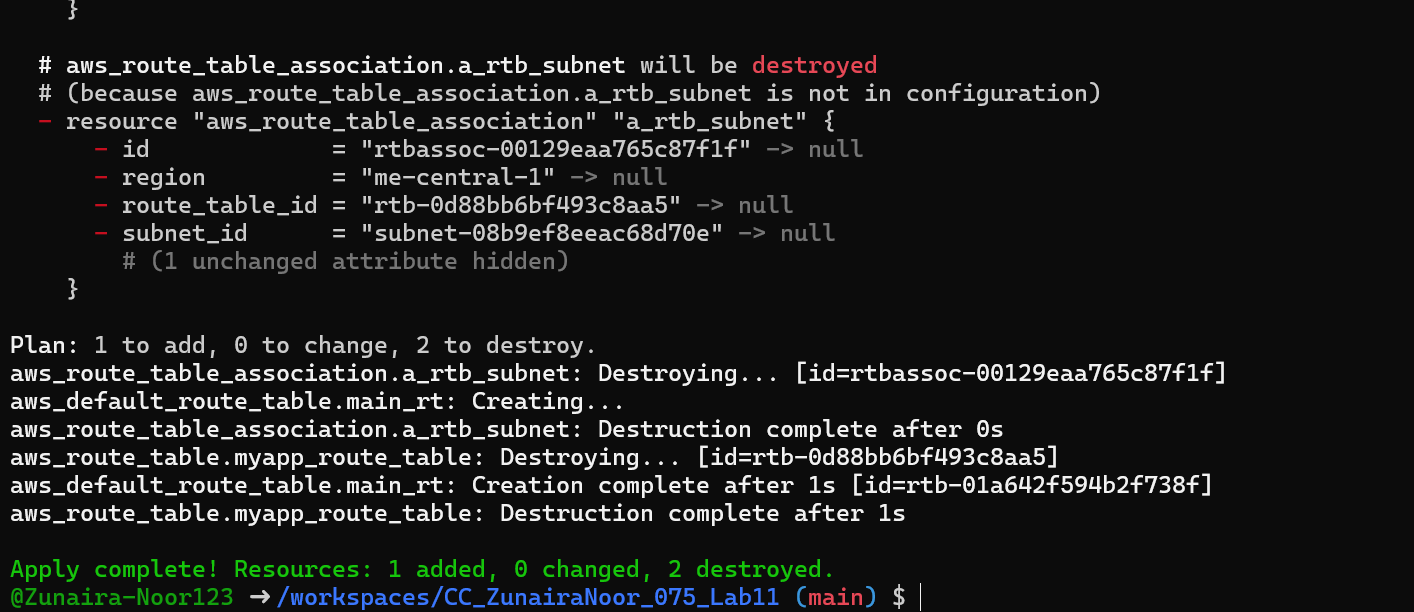
****

****

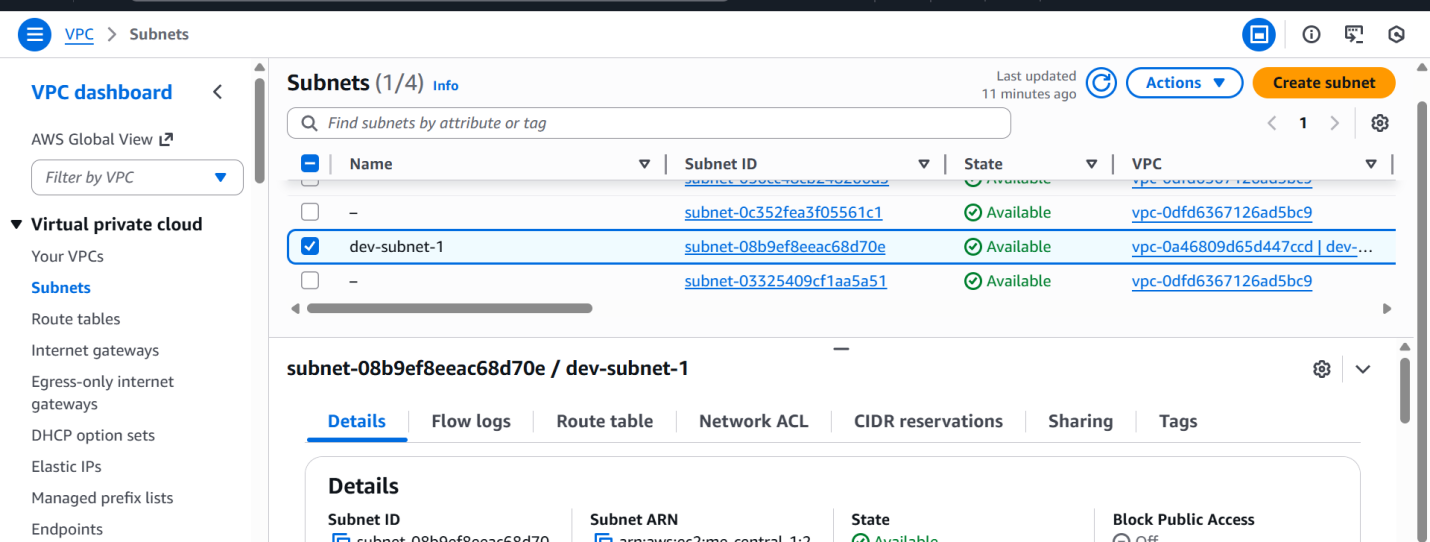
****

1. **Switch to default route table (use VPC default route table) Now remove (or comment out) the custom route table and association resources from main.tf:**

****

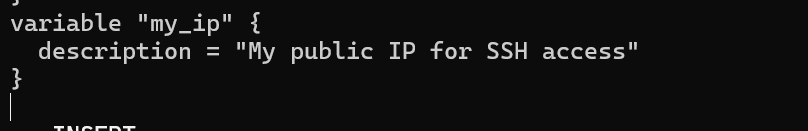
****

**Verify in AWS Console (MANDATORY)**

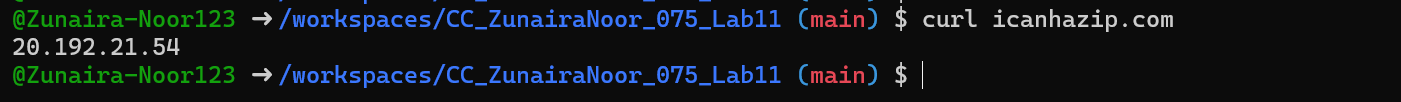
****

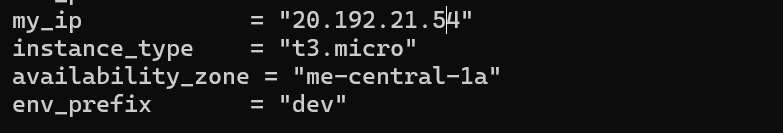
**Task 9 — Security Group, Key Pair, EC2 Instance, user\_data & nginx**

1. **Add variables to main.tf Add these variables to your main.tf:**

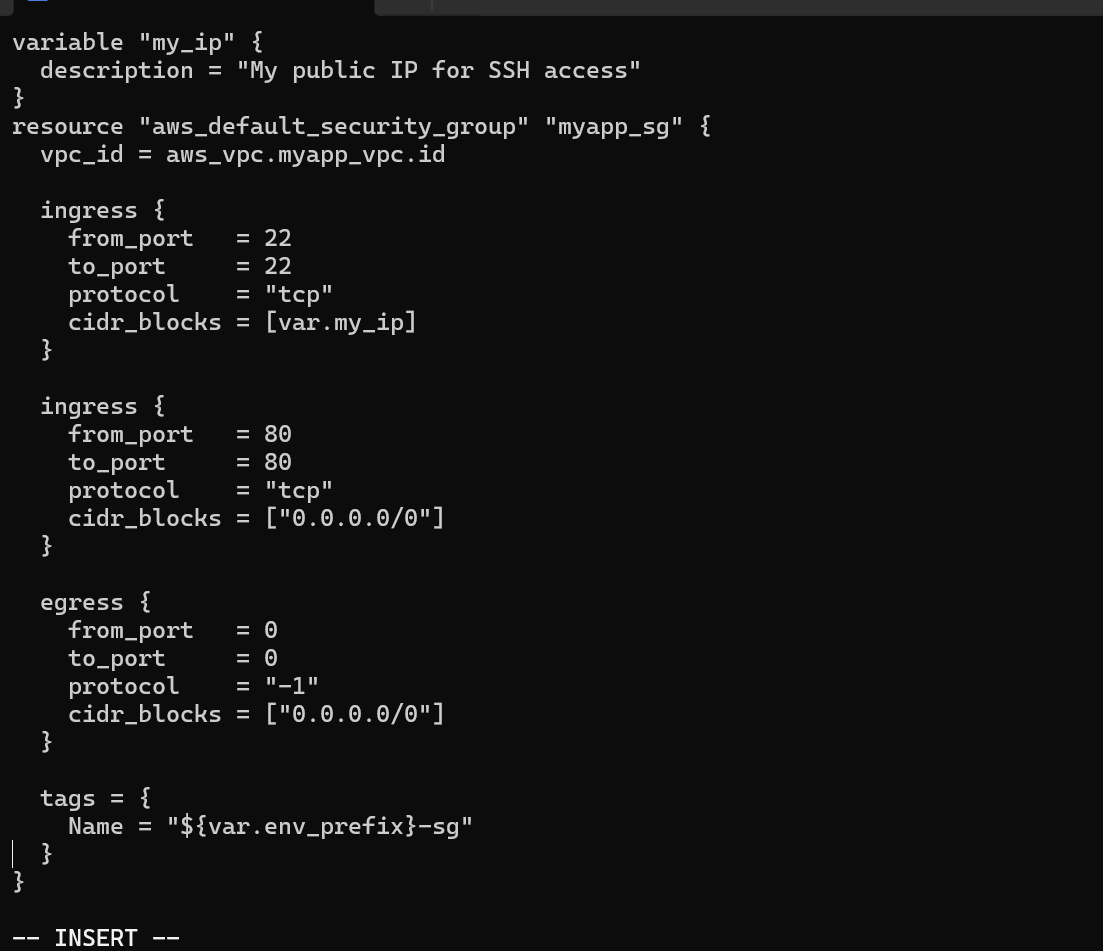
****

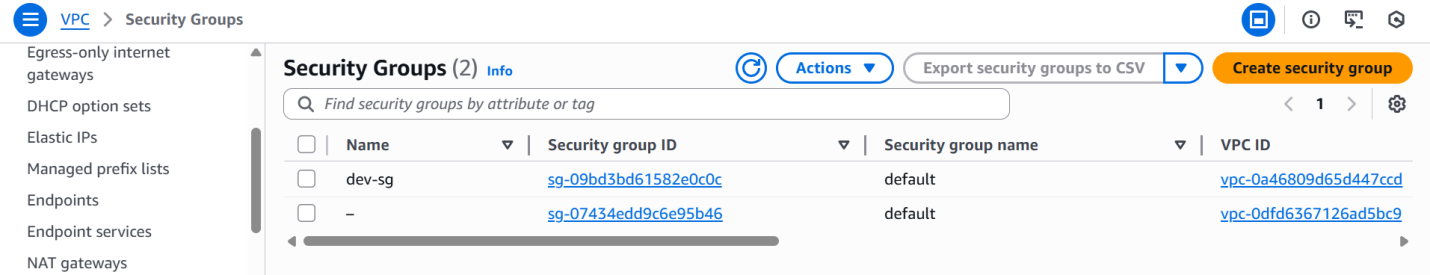
1. Get your public IP and set terraform.tfvars From the Codespace shell:

****

****

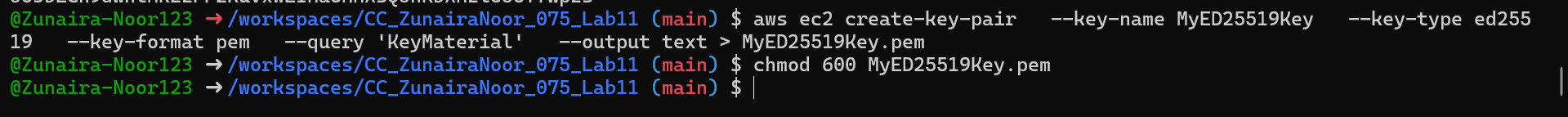
**Create the Security Group (main.tf) Add this resource to main.tf**

****

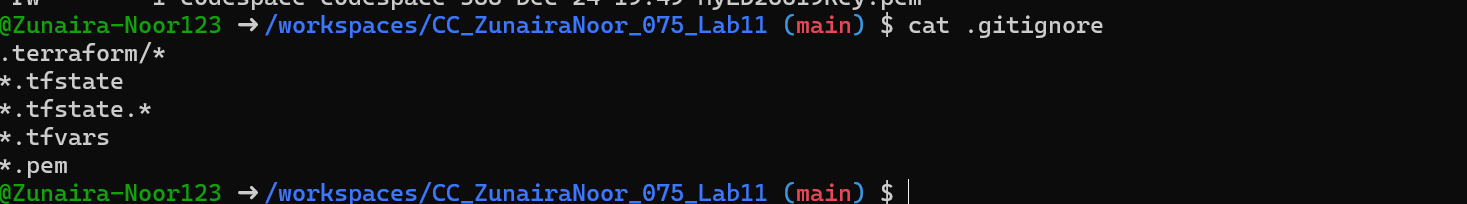
****

**Run apply:**

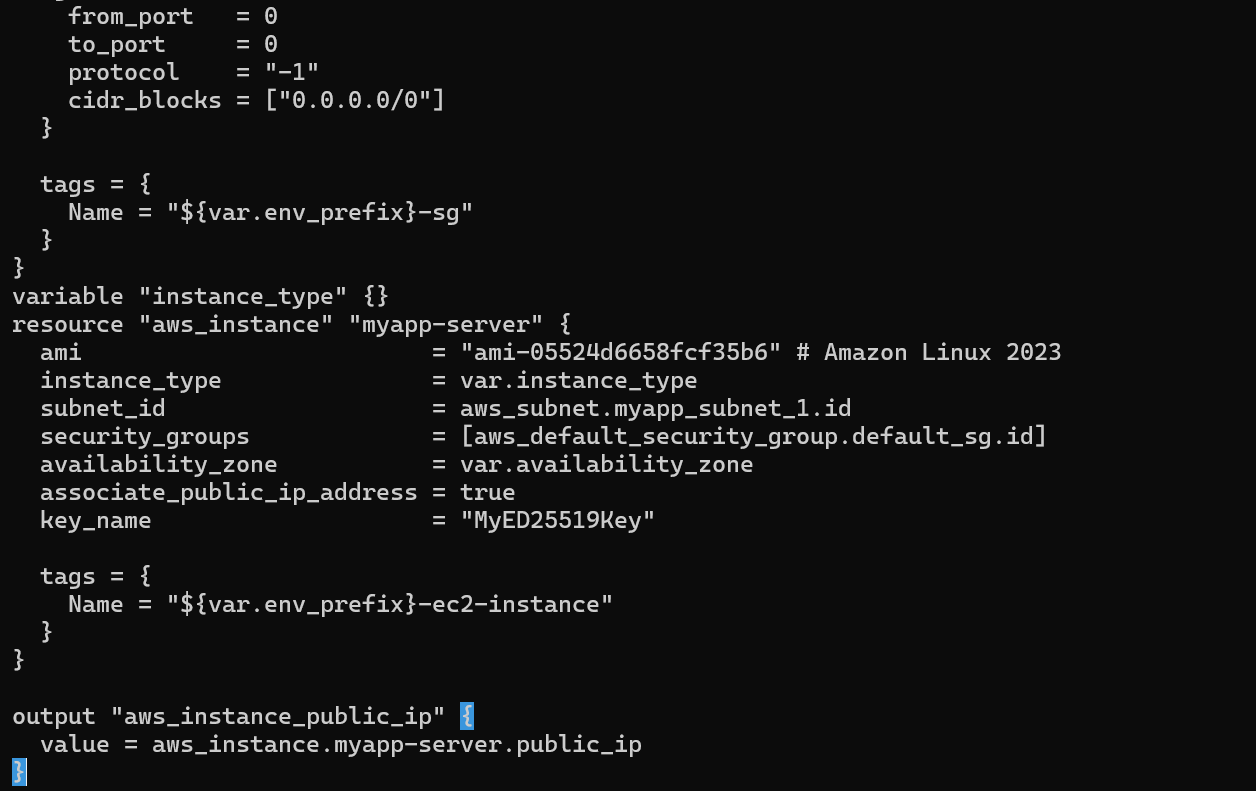
1. **Create an AWS key pair and save locally Create a key pair and store the private key in your Codespace. Do NOT commit the .pem file.**

****

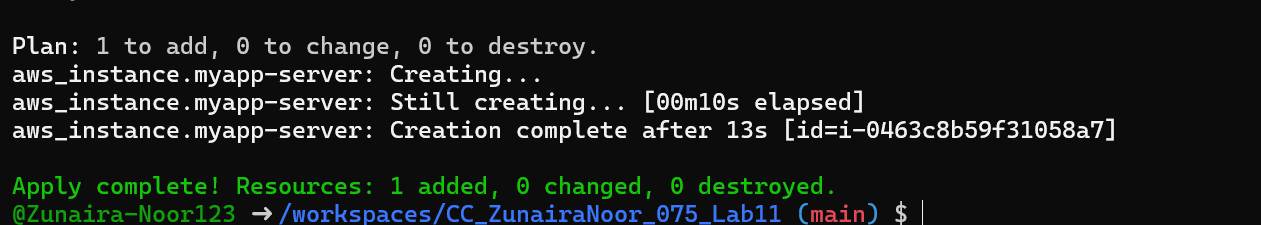
**Ensure .gitignore contains:**

****

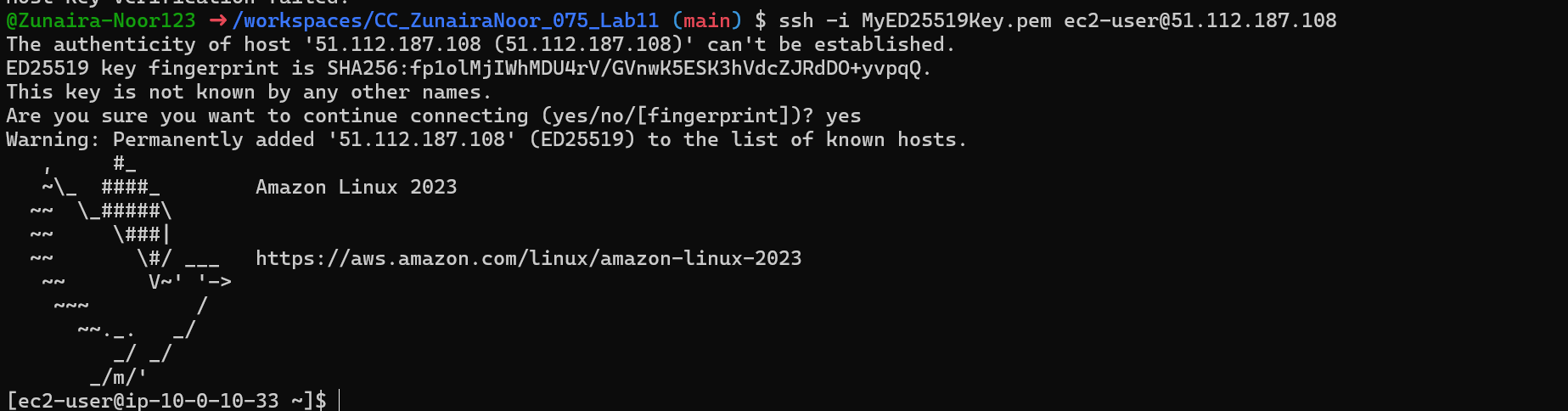
1. **Add EC2 instance resource (initial) Add the instance resource to main.tf (initially using the created key name):**

****

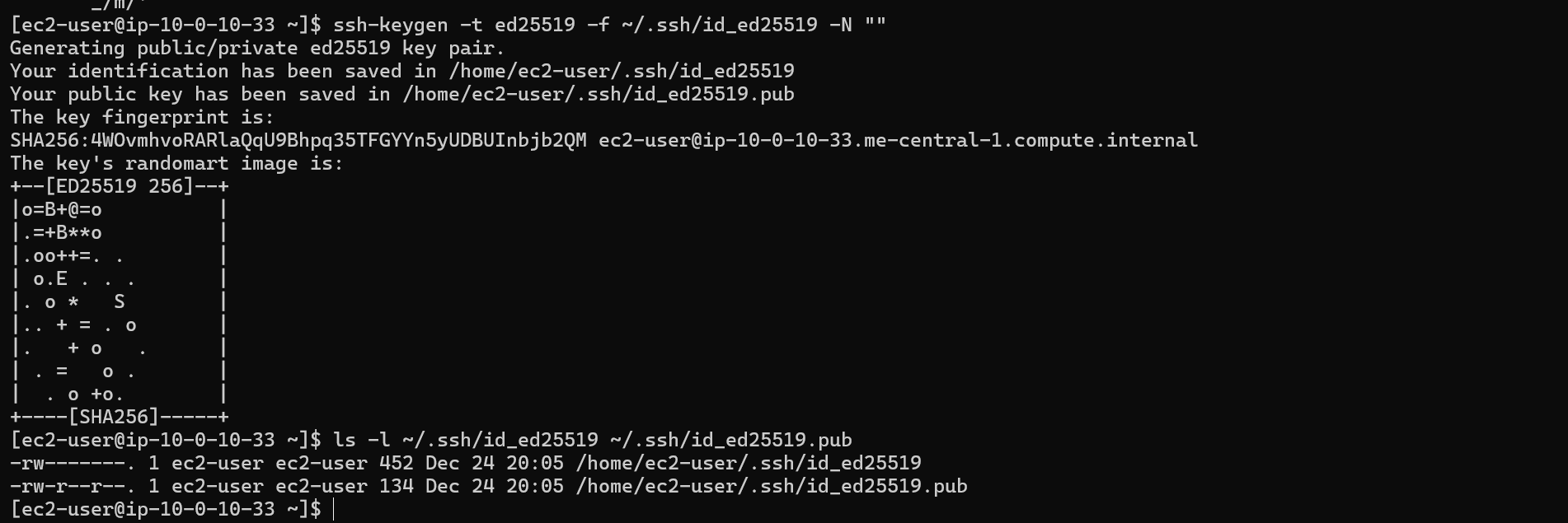
**Run:**

****

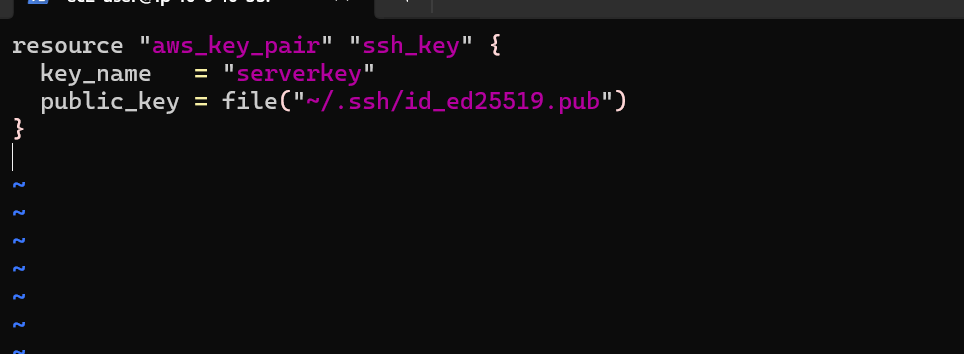
1. **SSH into the instance (using MyED25519Key) From the Codespace:**

****

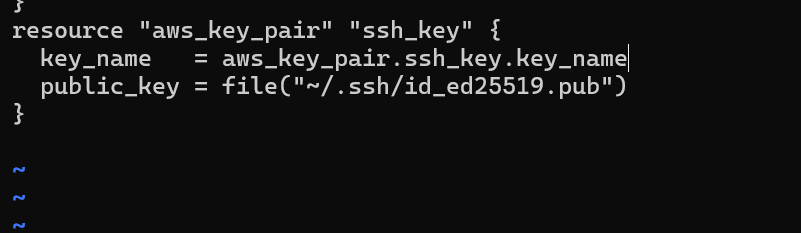
1. **Generate a local SSH keypair and register it in AWS via Terraform On your Codespace, generate an SSH key pair (accept defaults or specify path):**

****

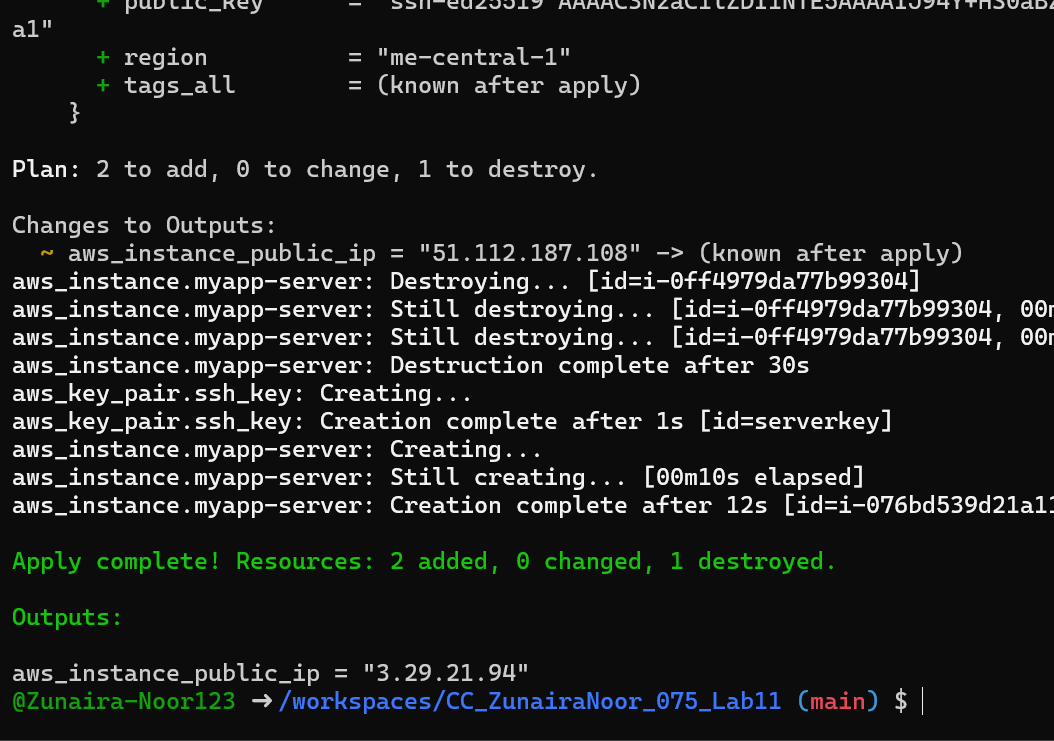
**Add a Terraform resource in main.tf to register the public key:**



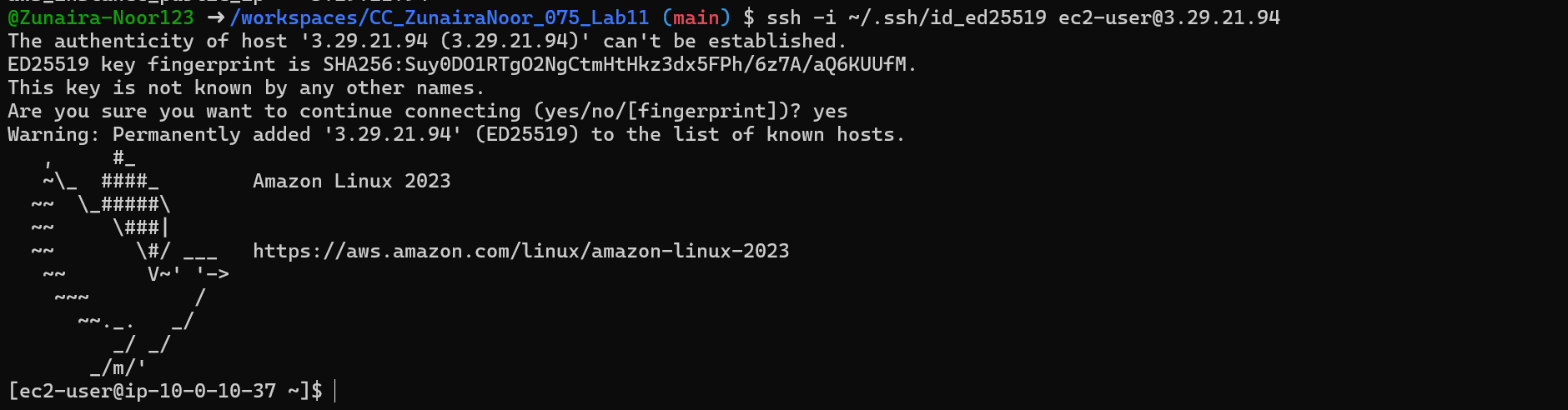
Update the EC2 resource to use the Terraform-managed key:



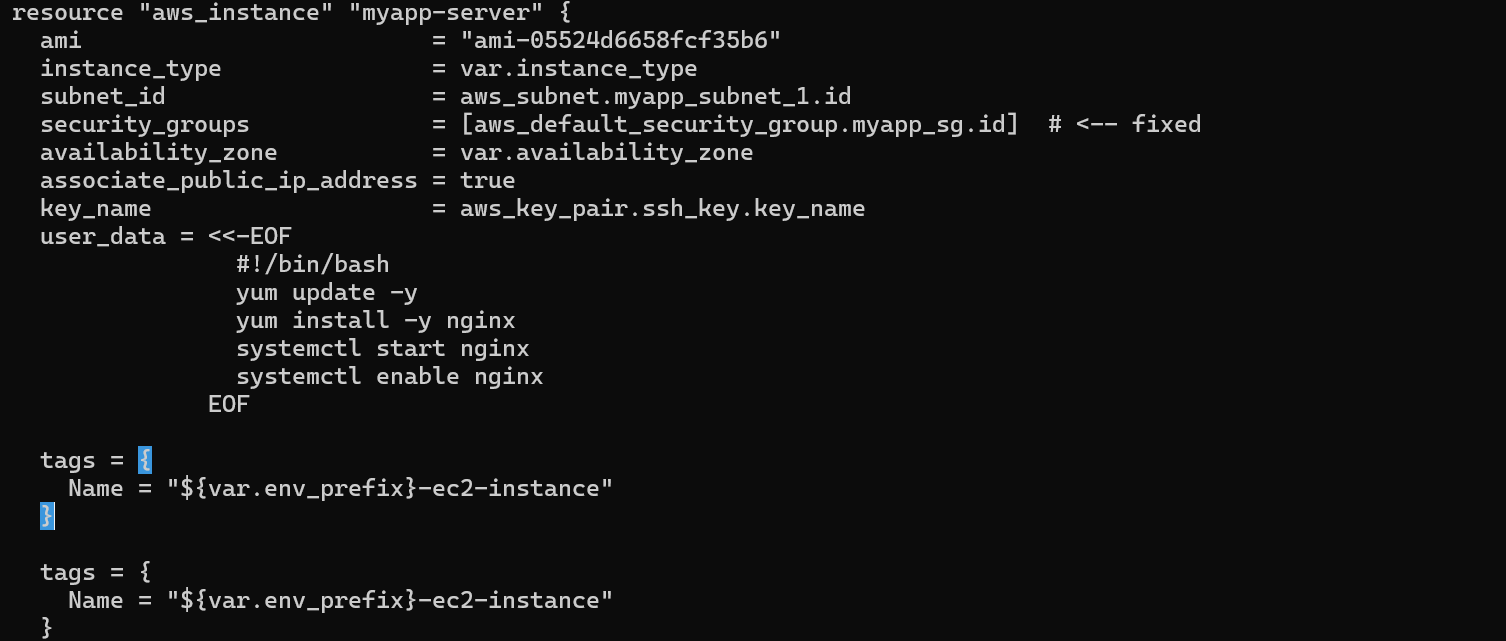
Run:



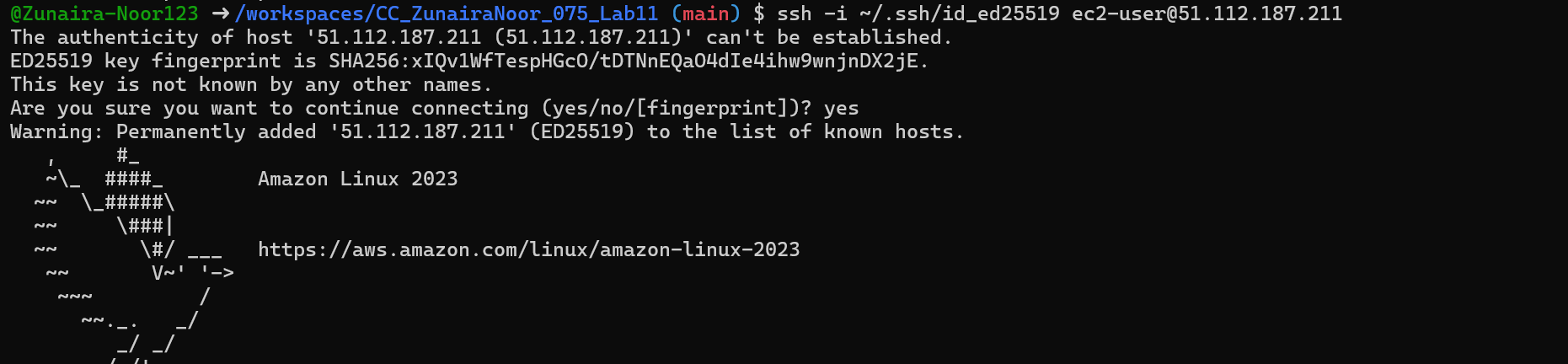
1. SSH using the newly registered key Now SSH with your generated private key (the default ssh client will pick up ~/.ssh/id\_ed25519):



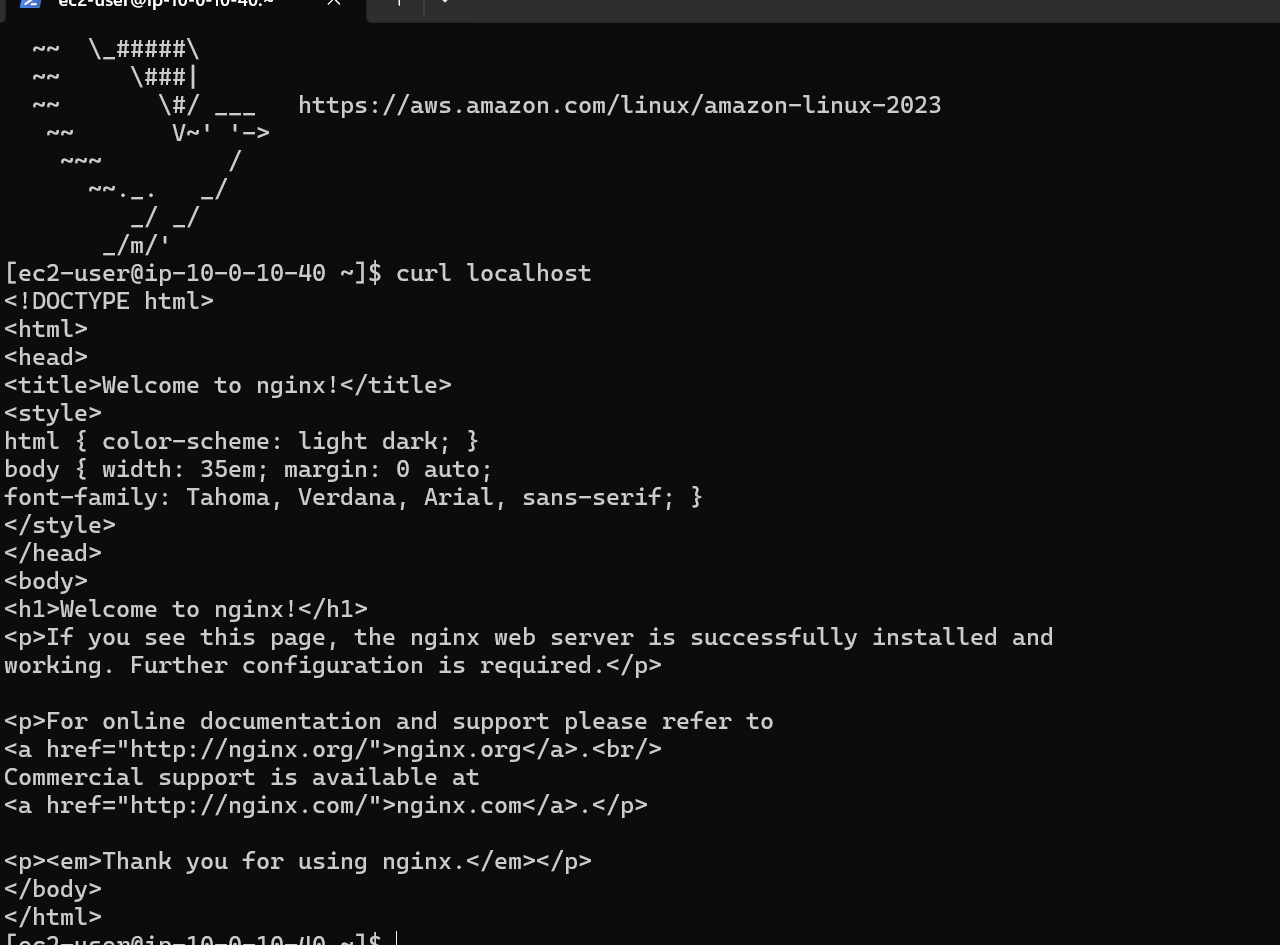
1. Install nginx via inline user\_data Modify the aws\_instance resource to include inline user\_data:



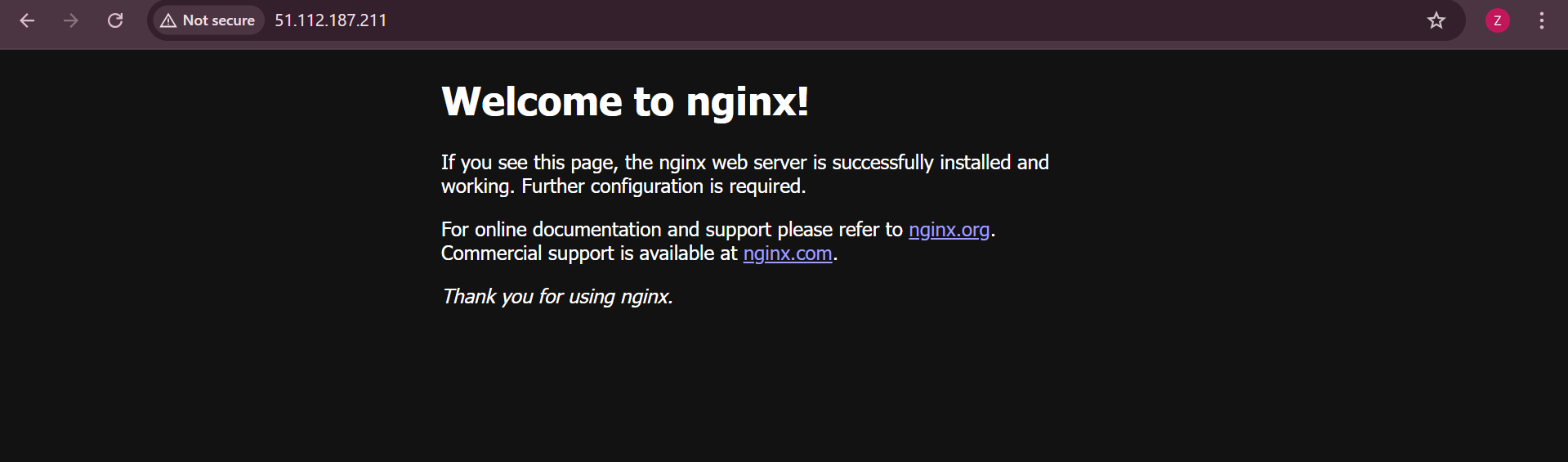
SSH in and run:



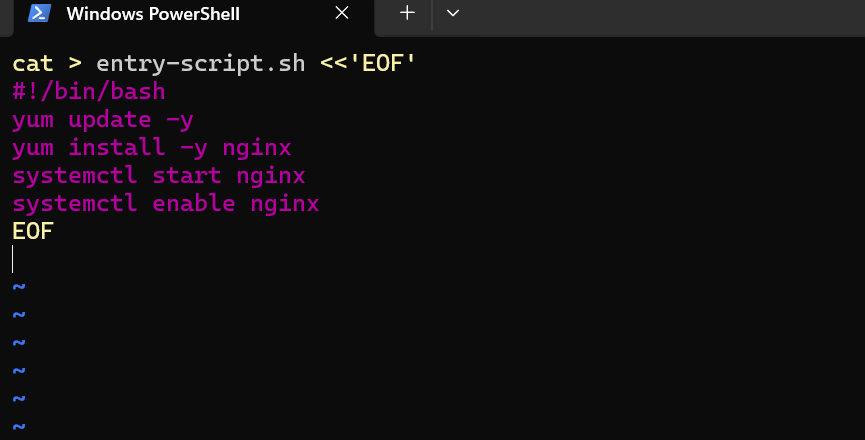
Verify nginx installation (inline user\_data or script)



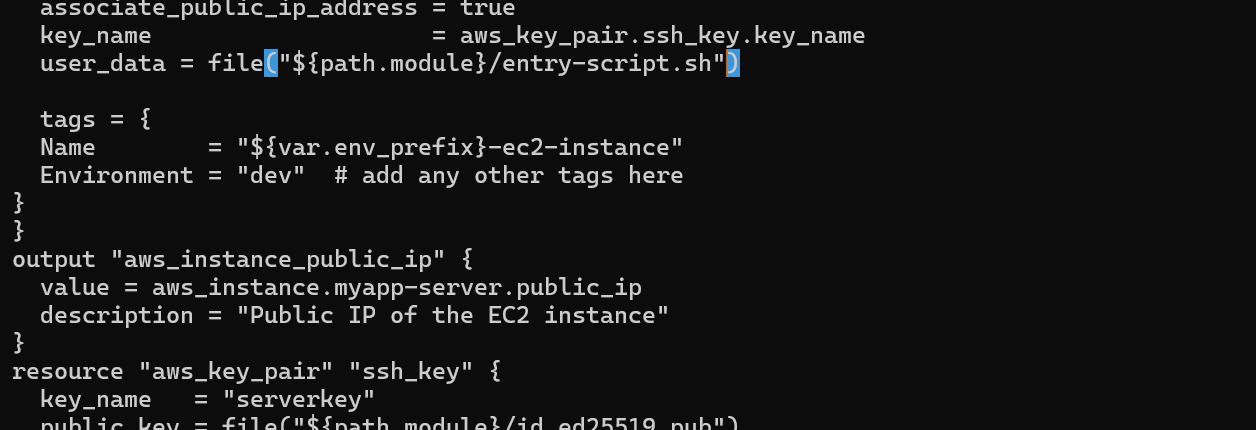
Open in browser

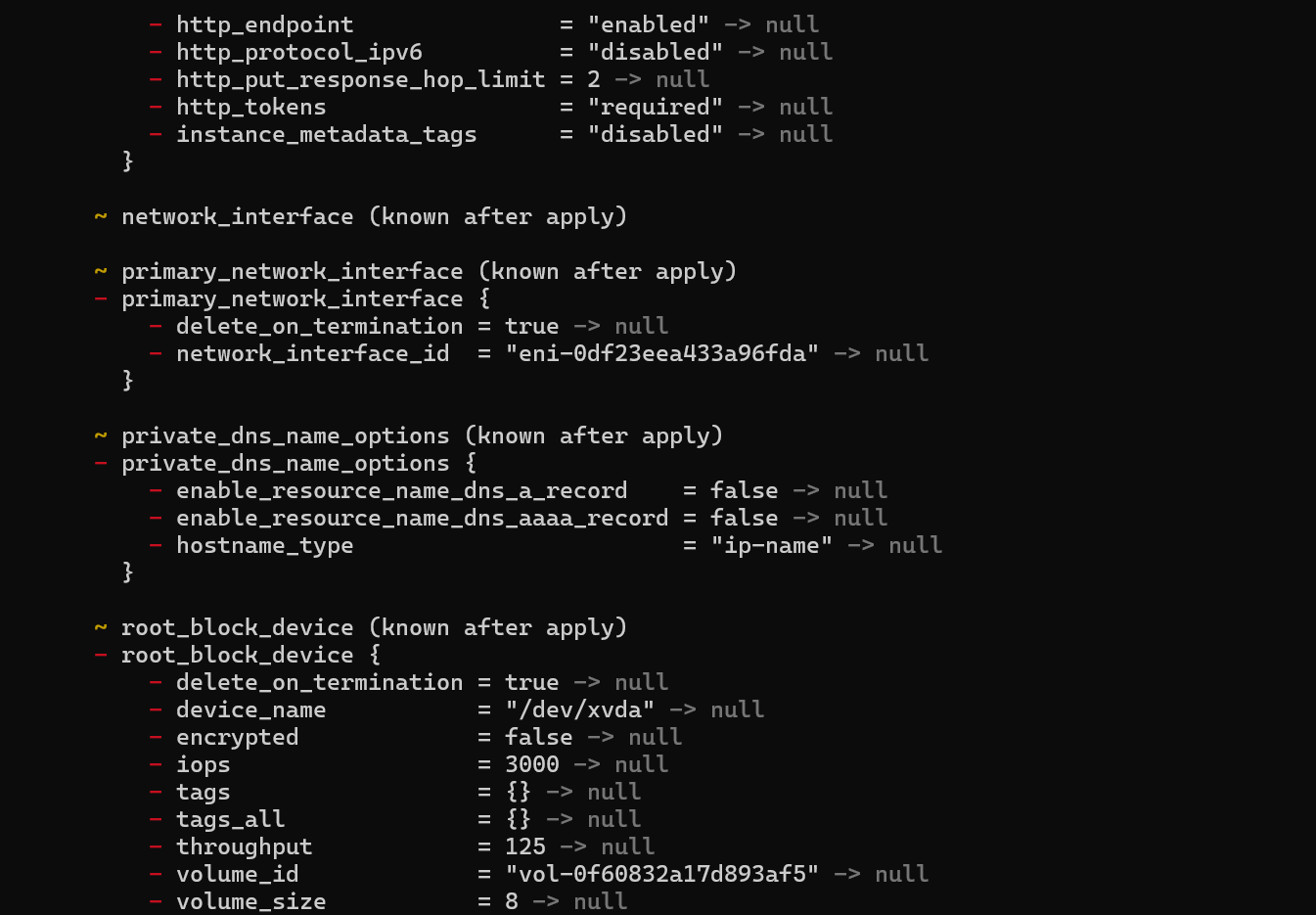


1. Use an external script for user\_data Create a script file in the Codespace:



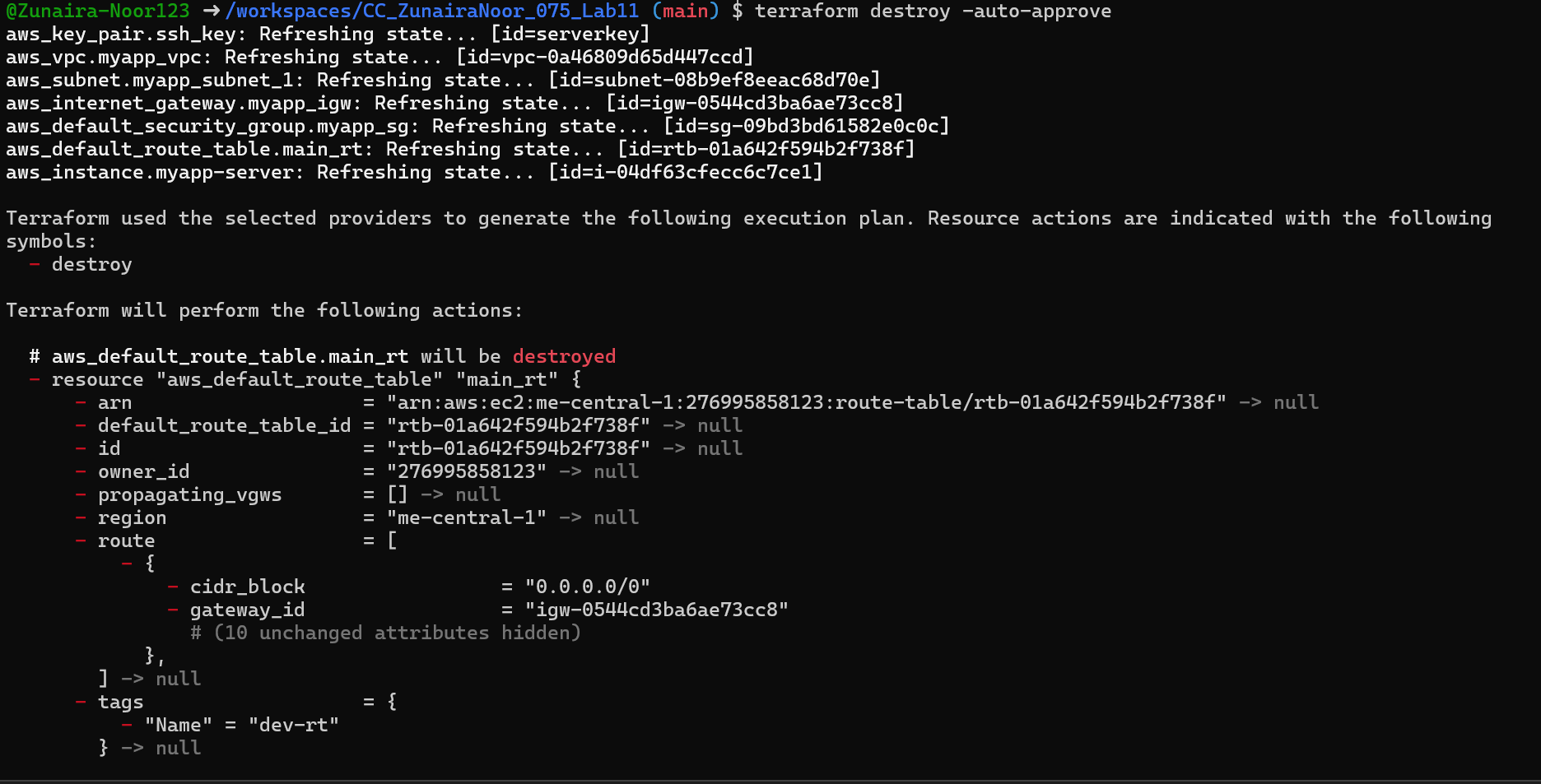
Update aws\_instance to use the file:



Apply: 

**Cleanup**

1. Destroy all resources:



1. Verify state files:

