Lab 11

Lab title: GH CLI Codespaces + AWS + Terraform: Variables, Collections, Sensitivity & EC2 Provisioning

Submitted to: Engr. Mohammad Shoaib

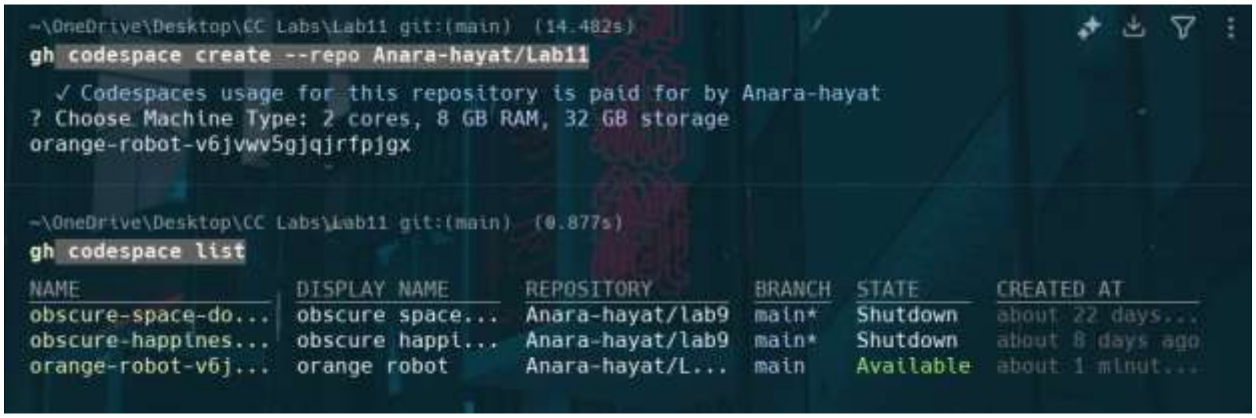
Submitted by: Anara Hayat

Reg#No: 2023-BSE-008

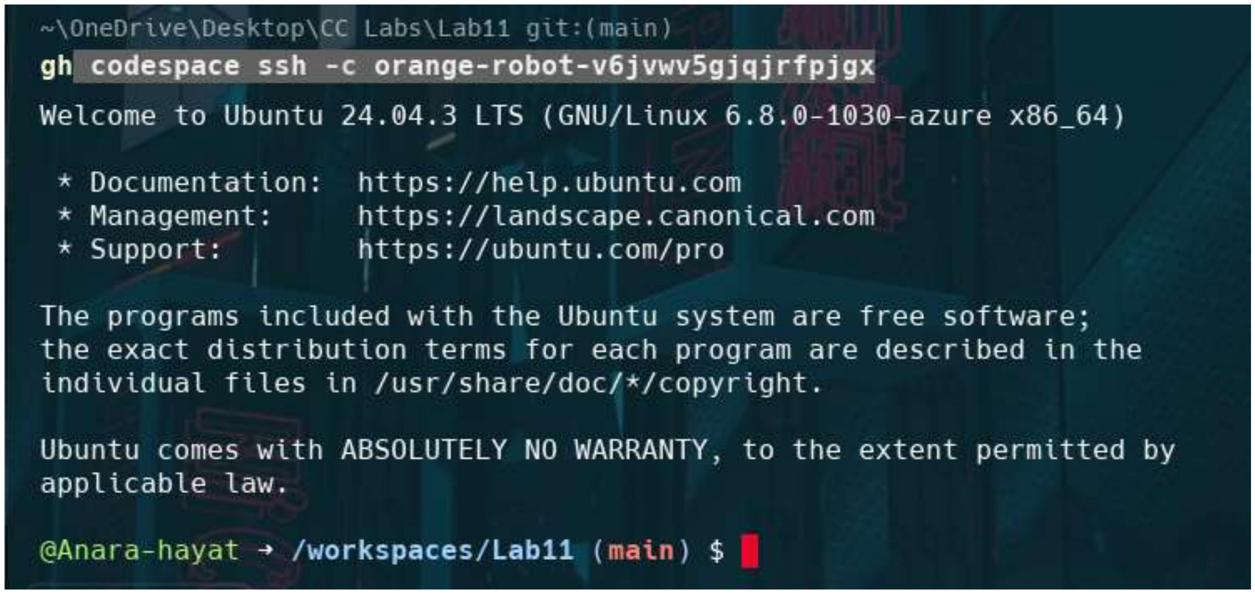
Task 0 Lab Setup (Codespace & GH CLI)

All actions below should be executed inside the Codespace shell.

Create Codespace & connect:



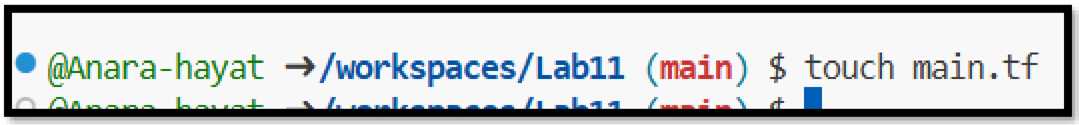
```
~\OneDrive\Desktop\CC Labs\Lab11 git:(main)  (14.482s)
gh codespace create --repo Anara-hayat/Lab11

  ✓ Codespaces usage for this repository is paid for by Anara-hayat
? Choose Machine Type: 2 cores, 8 GB RAM, 32 GB storage
orange-robot-v6jvwv5gjqjrfpjgx


~\OneDrive\Desktop\CC Labs\Lab11 git:(main)  (0.877s)
gh codespace list

NAME                 DISPLAY NAME     REPOSITORY          BRANCH  STATE      CREATED AT
obscure-space-do...  obscure space... Anara-hayat/lab9    main*   Shutdown   about 22 days...
obscure-happines...  obscure happi... Anara-hayat/lab9    main*   Shutdown   about 8 days ago
orange-robot-v6j...  orange robot     Anara-hayat/L...    main    Available  about 1 minut...
```



```
~\OneDrive\Desktop\CC Labs\Lab11 git:(main)
gh codespace ssh -c orange-robot-v6jvwv5gjqjrfpjgx
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

@Anara-hayat → /workspaces/Lab11 (main) $
```

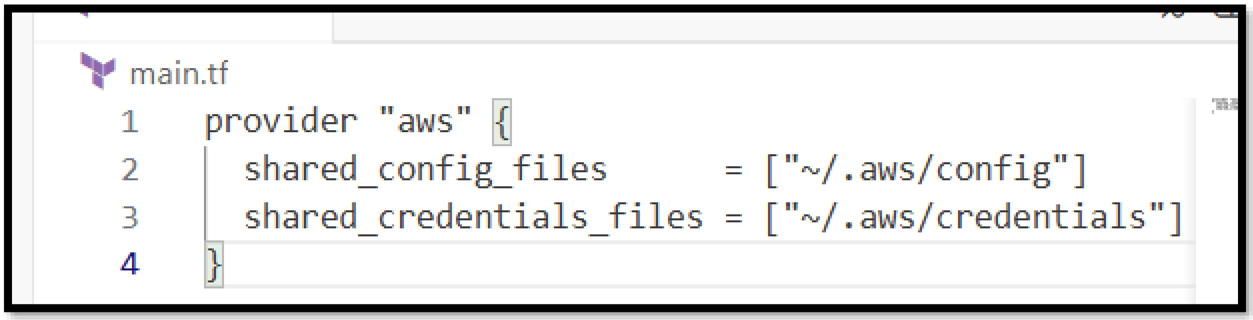
Task 1 — Provider & Basic variable (variable precedence)

1. **In Codespace create main.tf:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ touch main.tf
```

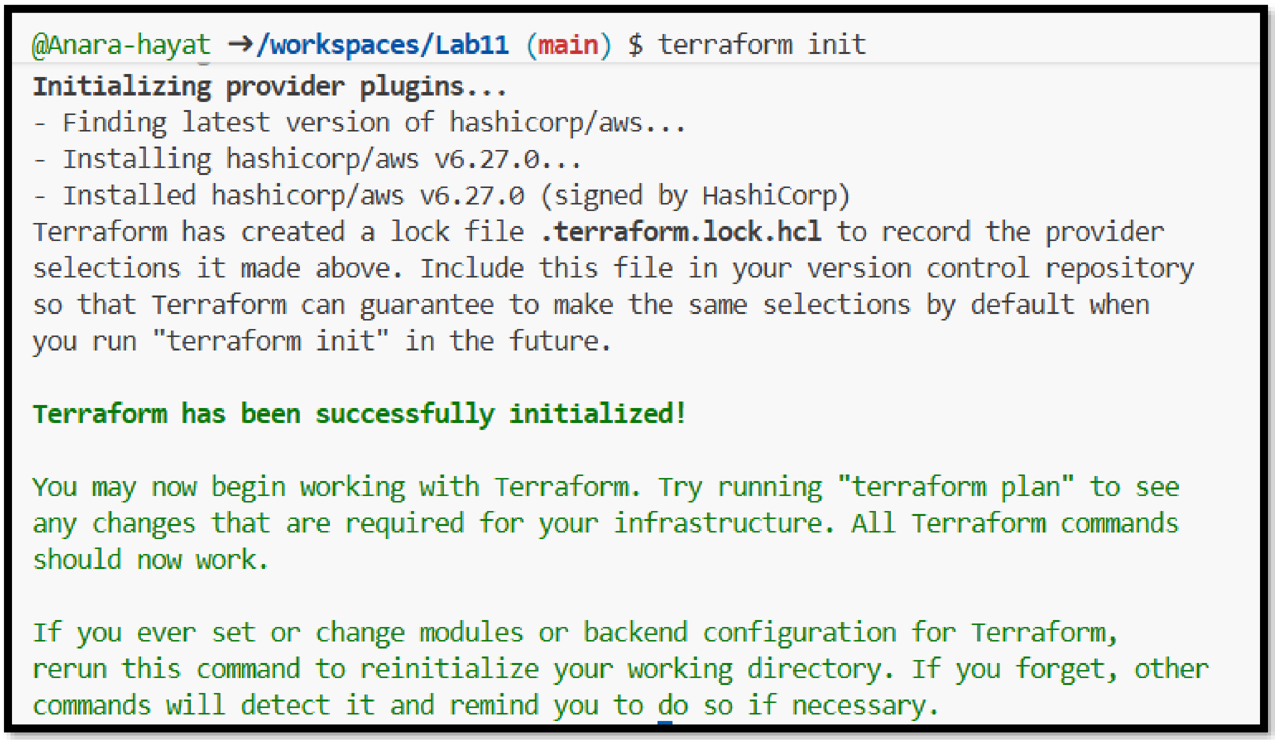
2. **Edit main.tf and add provider:**



main.tf

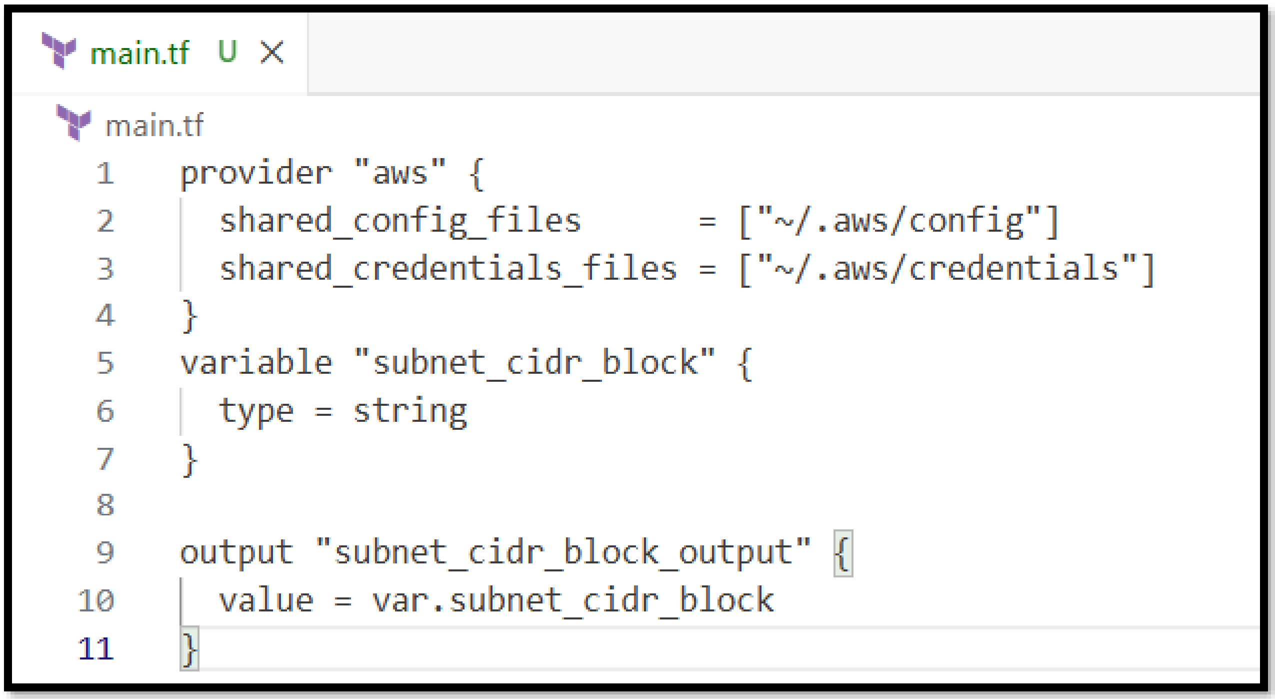
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
```

3. **Initialize:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform init
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

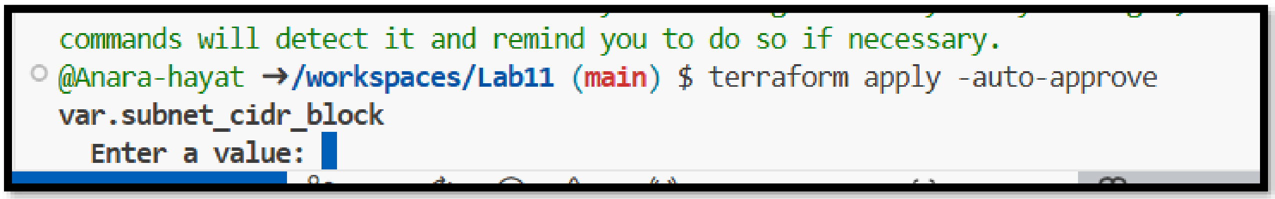
4. **Define a variable and output:**



main.tf U
main.tf

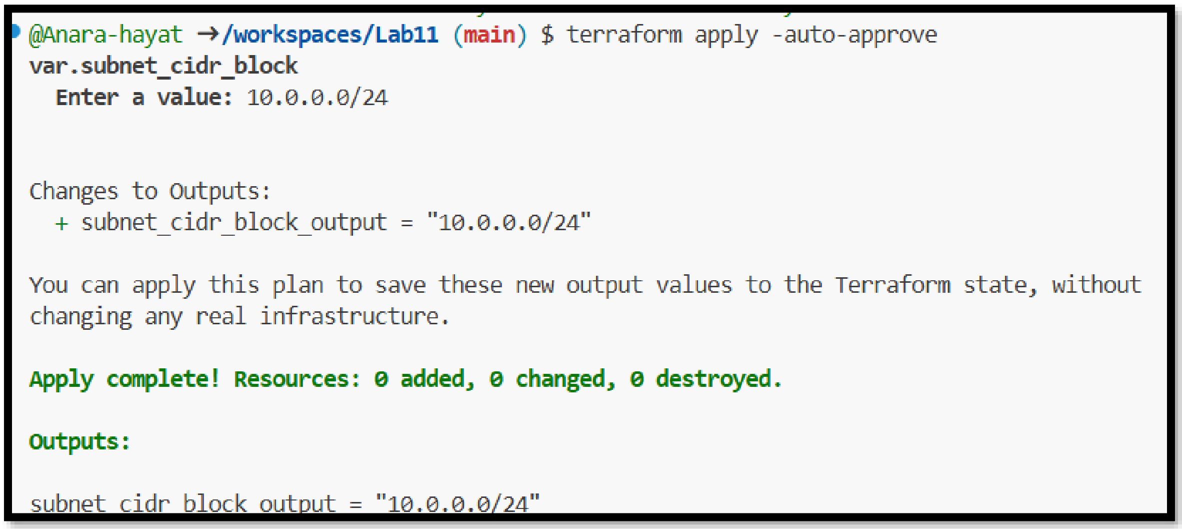
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
variable "subnet_cidr_block" {
  type = string
}

output "subnet_cidr_block_output" {
  value = var.subnet_cidr_block
}
```

5. **Run apply (first time without defaults):**



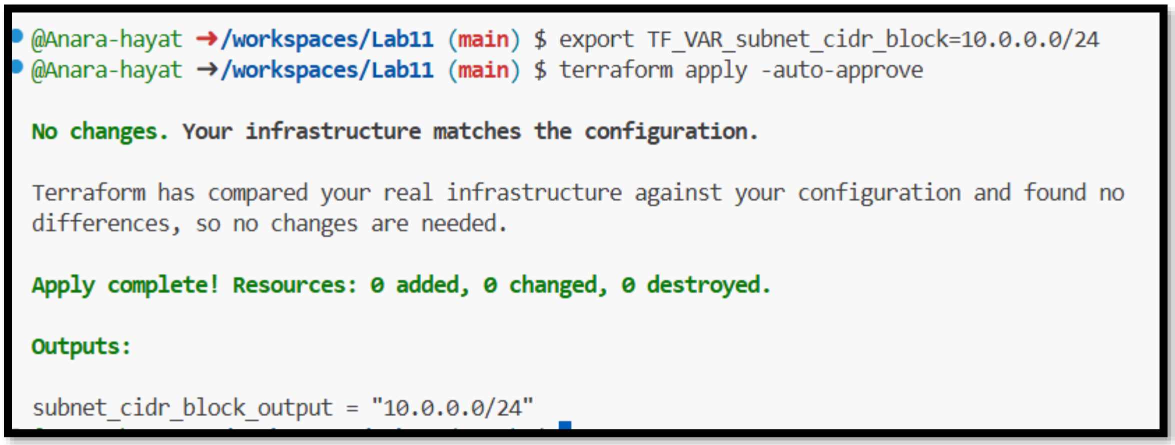
```
commands will detect it and remind you to do so if necessary.
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
var.subnet_cidr_block
  Enter a value: 
```

6. Add a default to the variable:

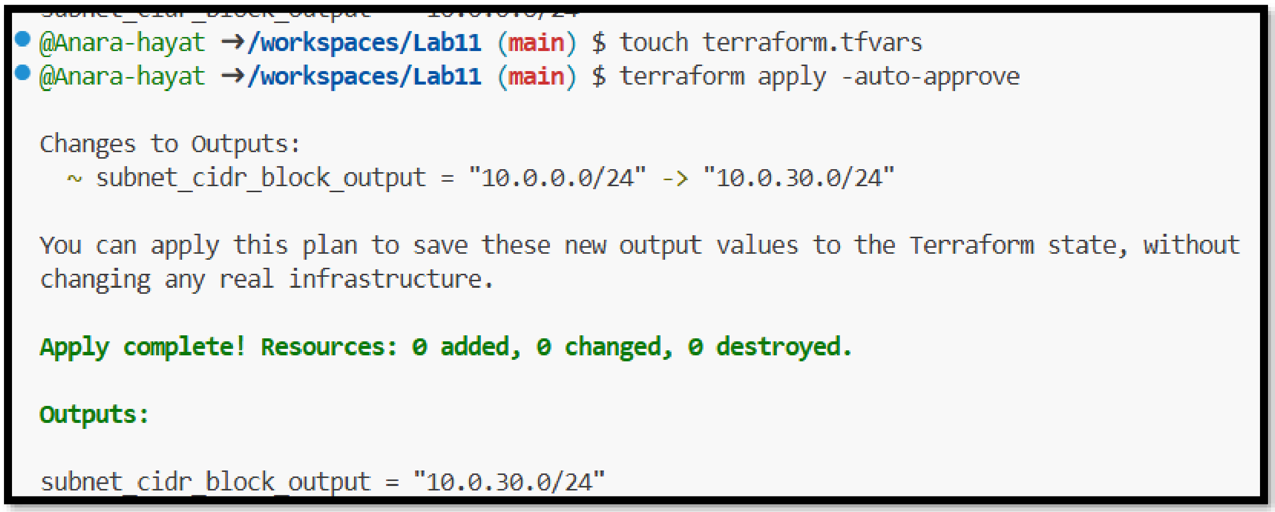


```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
var.subnet_cidr_block
  Enter a value: 10.0.0.0/24


Changes to Outputs:
  + subnet_cidr_block_output = "10.0.0.0/24"

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

subnet_cidr_block_output = "10.0.0.0/24"
```

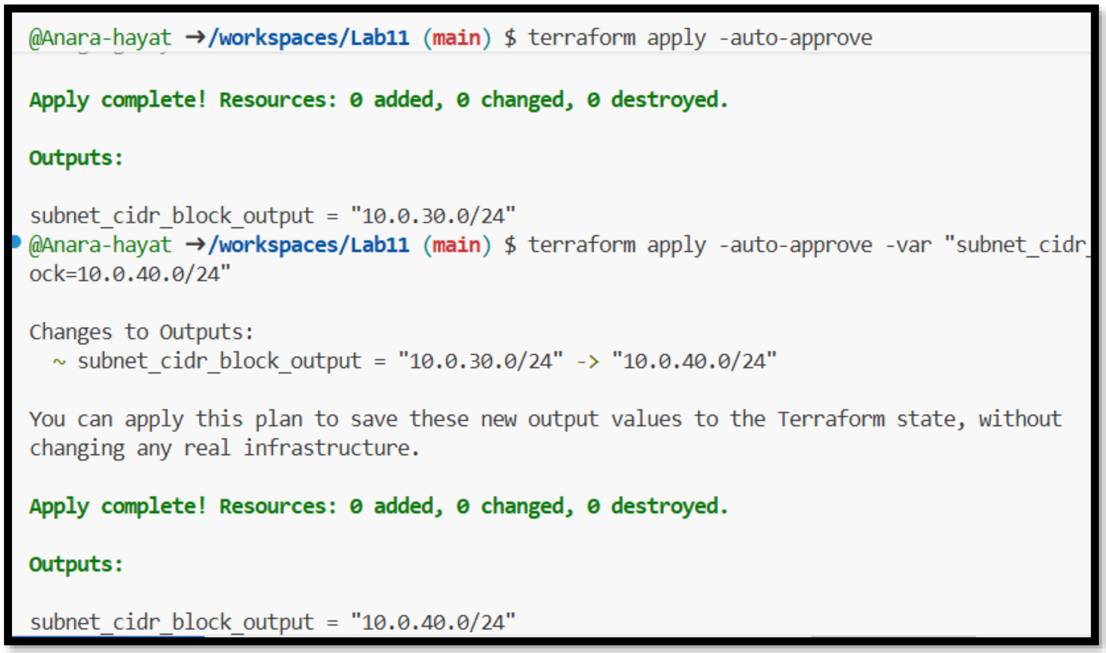
7. **Export environment variable in Codespace shell:**


```
@Anara-hayat →/workspaces/Lab11 (main) $ export TF_VAR_subnet_cidr_block=10.0.0.0/24
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no
differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

subnet_cidr_block_output = "10.0.0.0/24"
```

8. **Create terraform.tfvars overriding values:**

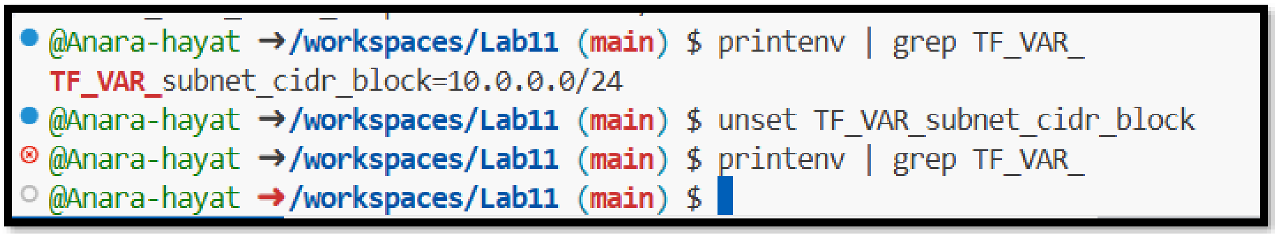

```
@Anara-hayat →/workspaces/Lab11 (main) $ touch terraform.tfvars
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

Changes to Outputs:
  ~ subnet_cidr_block_output = "10.0.0.0/24" -> "10.0.30.0/24"

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

subnet_cidr_block_output = "10.0.30.0/24"
```

9. **Override with -var:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

subnet_cidr_block_output = "10.0.30.0/24"
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve -var "subnet_cidr_
ock=10.0.40.0/24"

Changes to Outputs:
  ~ subnet_cidr_block_output = "10.0.30.0/24" -> "10.0.40.0/24"

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

subnet_cidr_block_output = "10.0.40.0/24"
```

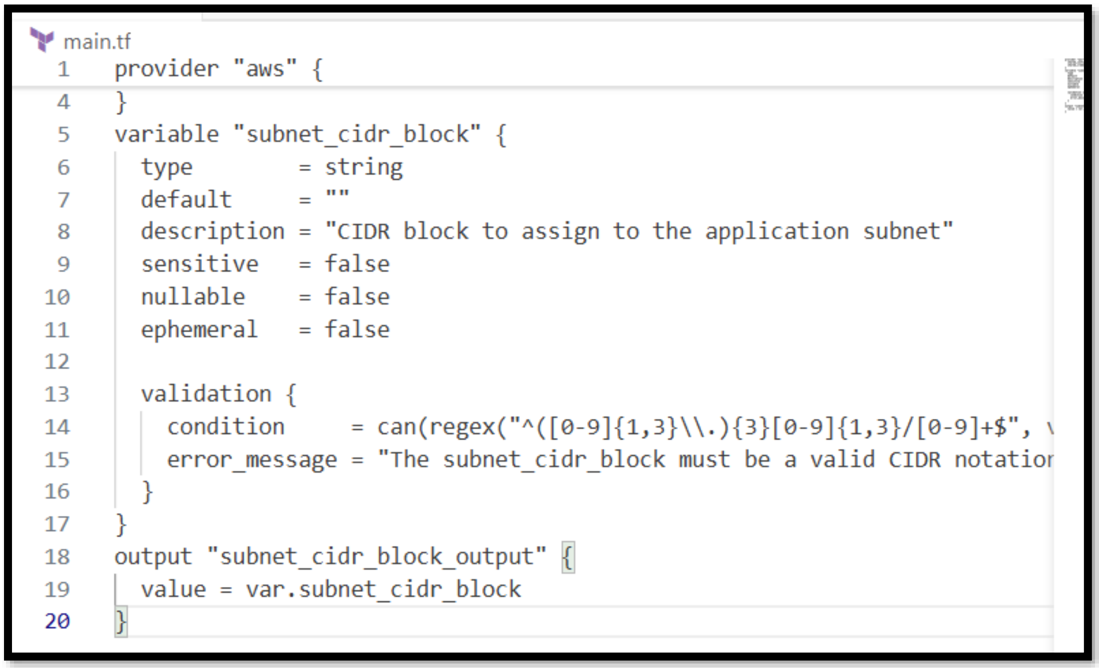
10. **Show and unset env var:**



```
● @Anara-hayat →/workspaces/Lab11 (main) $ printenv | grep TF_VAR_
  TF_VAR_subnet_cidr_block=10.0.0.0/24
● @Anara-hayat →/workspaces/Lab11 (main) $ unset TF_VAR_subnet_cidr_block
⊗ @Anara-hayat →/workspaces/Lab11 (main) $ printenv | grep TF_VAR_
○ @Anara-hayat →/workspaces/Lab11 (main) $
```

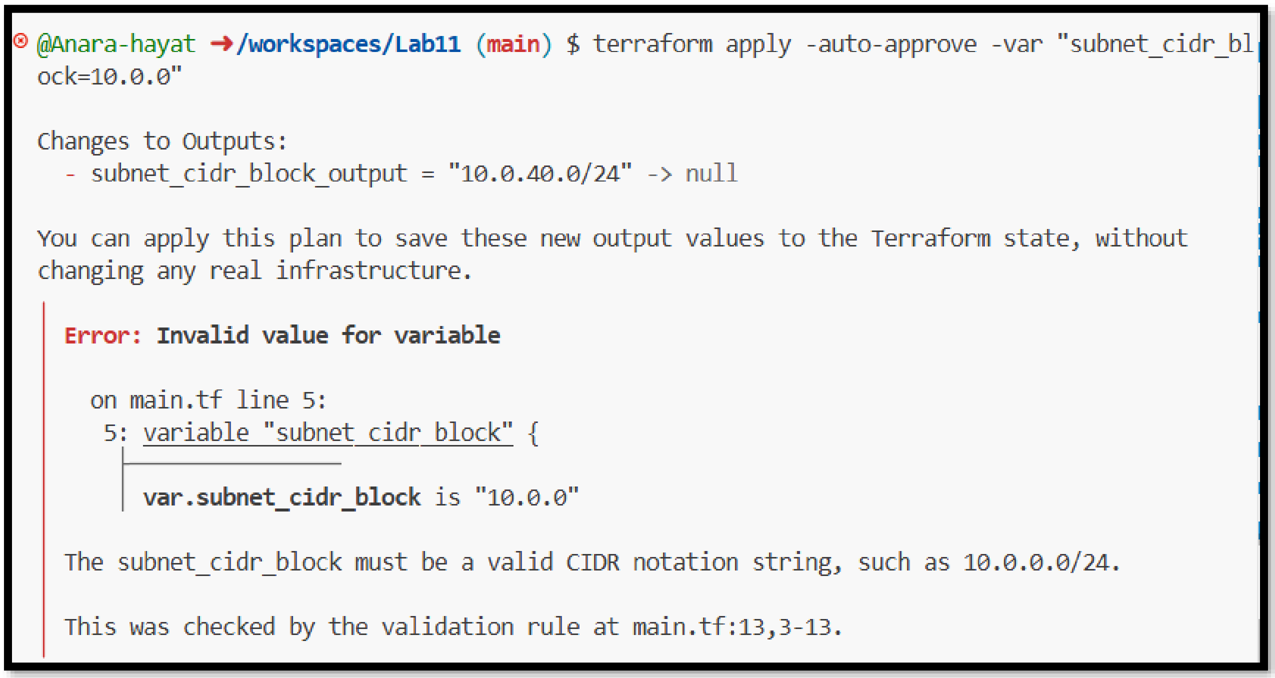
Task 2 — Variable validation & sensitive / ephemeral variables

1. **Replace subnet_cidr_block variable with this (validation included):**



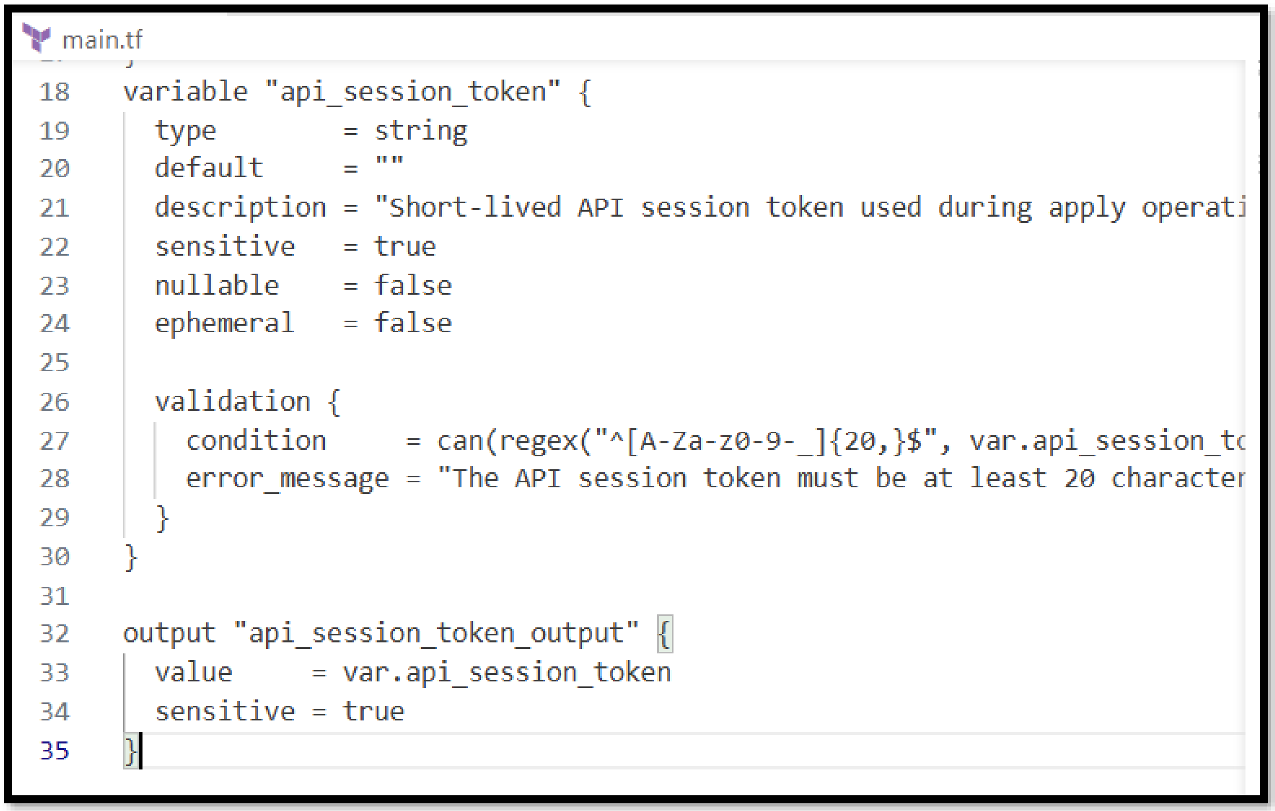
```
main.tf
provider "aws" {
}
variable "subnet_cidr_block" {
  type        = string
  default     = ""
  description = "CIDR block to assign to the application subnet"
  sensitive   = false
  nullable    = false
  ephemeral   = false

  validation {
    condition     = can(regex("^([0-9]{1,3}\\.){3}[0-9]{1,3}/[0-9]+$", v
    error_message = "The subnet_cidr_block must be a valid CIDR notatior
  }
}
output "subnet_cidr_block_output" {
  value = var.subnet_cidr_block
}
```

2. **Test validation failure:**



```
@Anara-hayat ➜ /workspaces/Lab11 (main) $ terraform apply -auto-approve -var "subnet_cidr_bl
ock=10.0.0"

Changes to Outputs:
  - subnet_cidr_block_output = "10.0.40.0/24" -> null

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

  Error: Invalid value for variable

    on main.tf line 5:
     5: variable "subnet_cidr_block" {
      ├────────────────
      │ var.subnet_cidr_block is "10.0.0"

  The subnet_cidr_block must be a valid CIDR notation string, such as 10.0.0.0/24.

  This was checked by the validation rule at main.tf:13,3-13.
```

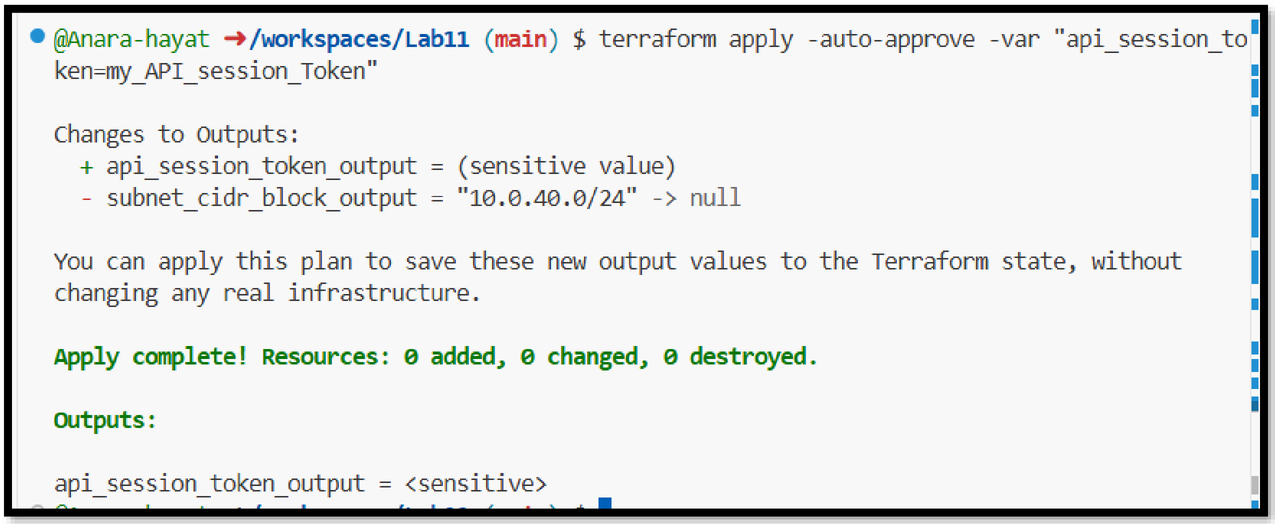
3. **Create a sensitive variable api_session_token and output (sensitive):**



main.tf

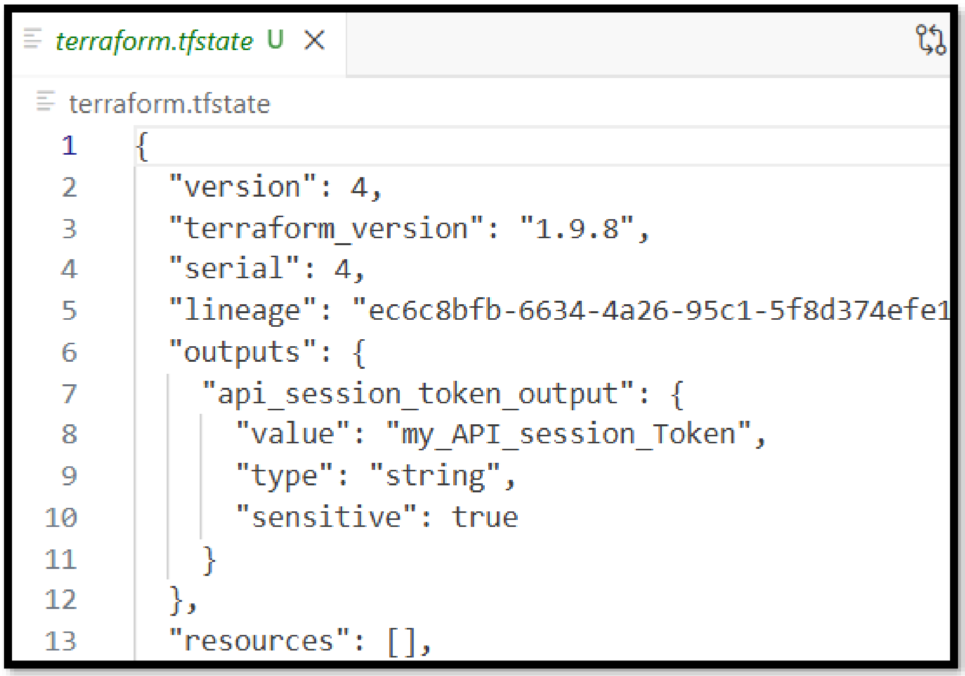
```
variable "api_session_token" {
  type        = string
  default     = ""
  description = "Short-lived API session token used during apply operati
  sensitive   = true
  nullable    = false
  ephemeral   = false

  validation {
    condition     = can(regex("^[A-Za-z0-9-_]{20,}$", var.api_session_to
    error_message = "The API session token must be at least 20 character
  }
}

output "api_session_token_output" {
  value     = var.api_session_token
  sensitive = true
}
```

4. **Run with -var to observe sensitive output behavior:**



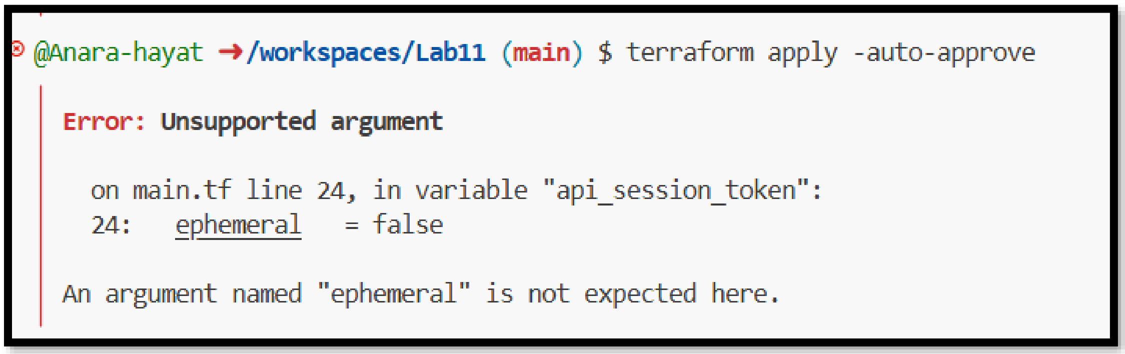
```
@Anara-hayat ➜ /workspaces/Lab11 (main) $ terraform apply -auto-approve -var "api_session_to
ken=my_API_session_Token"

Changes to Outputs:
  + api_session_token_output = (sensitive value)
  - subnet_cidr_block_output = "10.0.40.0/24" -> null

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_session_token_output = <sensitive>
```

5. **Check terraform.state for the sensitive output:**



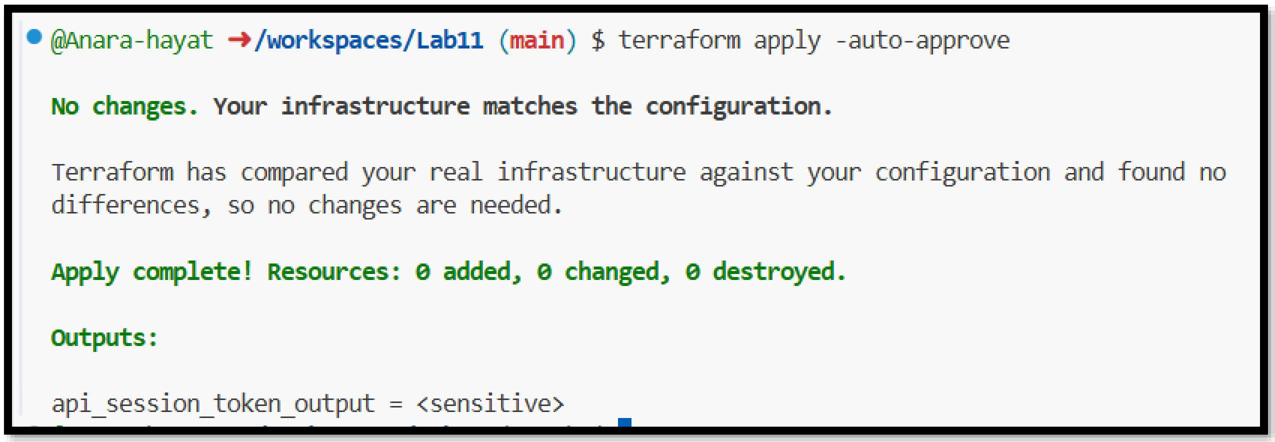
```
terraform.tfstate U X
terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.9.8",
  "serial": 4,
  "lineage": "ec6c8bfb-6634-4a26-95c1-5f8d374efe1
  "outputs": {
    "api_session_token_output": {
      "value": "my_API_session_Token",
      "type": "string",
      "sensitive": true
    }
  },
  "resources": [],
```

6. **Make variable ephemeral to hide from state:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

Error: Unsupported argument

  on main.tf line 24, in variable "api_session_token":
  24:   ephemeral    = false

An argument named "ephemeral" is not expected here.
```

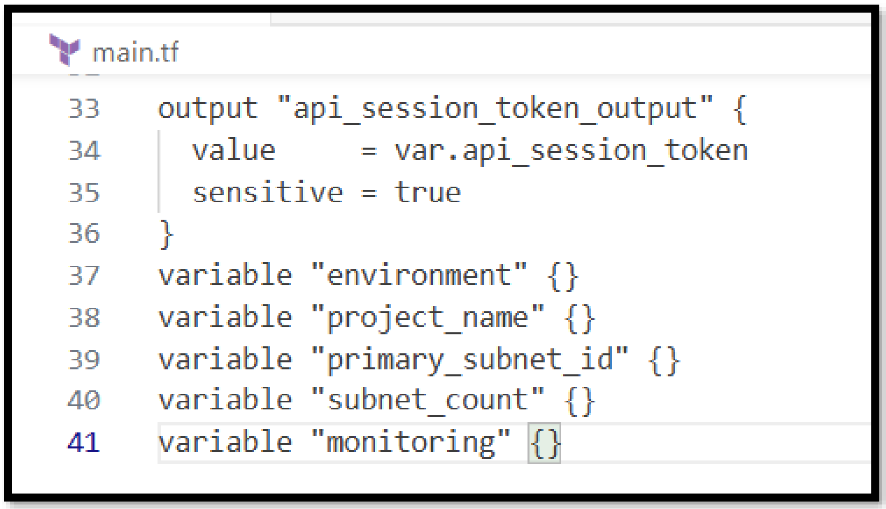
7. **Set default to test local default:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no
differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_session_token_output = <sensitive>
```

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

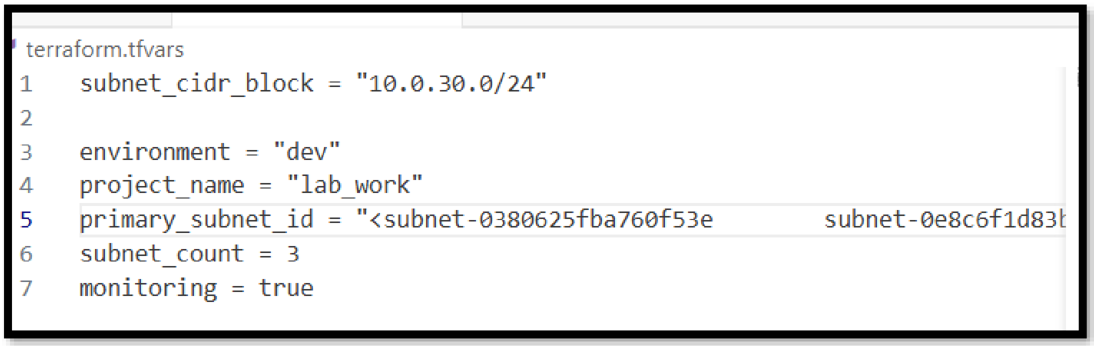
1. **Add variables to main.tf:**



main.tf

```
output "api_session_token_output" {
  value     = var.api_session_token
  sensitive = true
}
variable "environment" {}
variable "project_name" {}
variable "primary_subnet_id" {}
variable "subnet_count" {}
variable "monitoring" {}
```

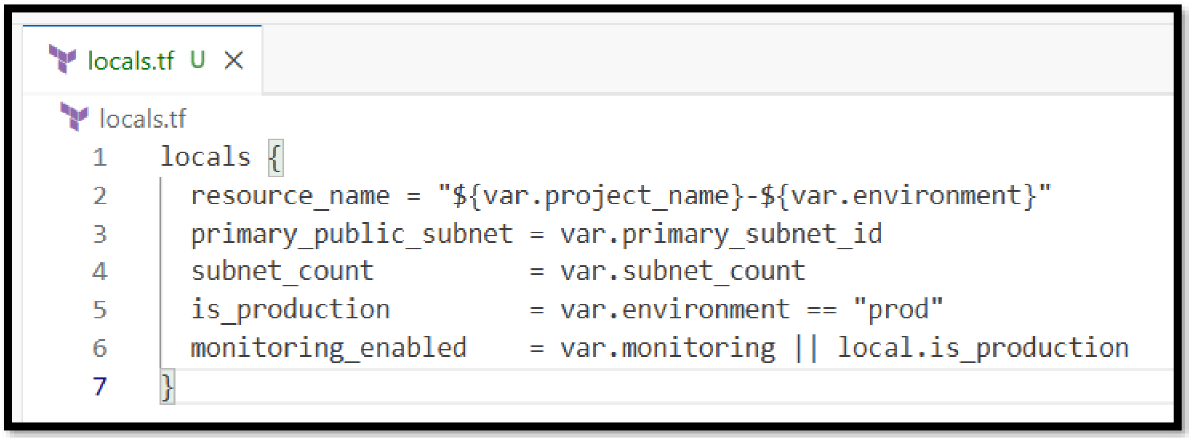
2. **Populate terraform.tfvars *after* discovering actual subnet id for availability zone me-central-1a:**



terraform.tfvars

```
subnet_cidr_block = "10.0.30.0/24"

environment = "dev"
project_name = "lab_work"
primary_subnet_id = "<subnet-0380625fba760f53e       subnet-0e8c6f1d83b
subnet_count = 3
monitoring = true
```

3. **Create locals.tf with:**

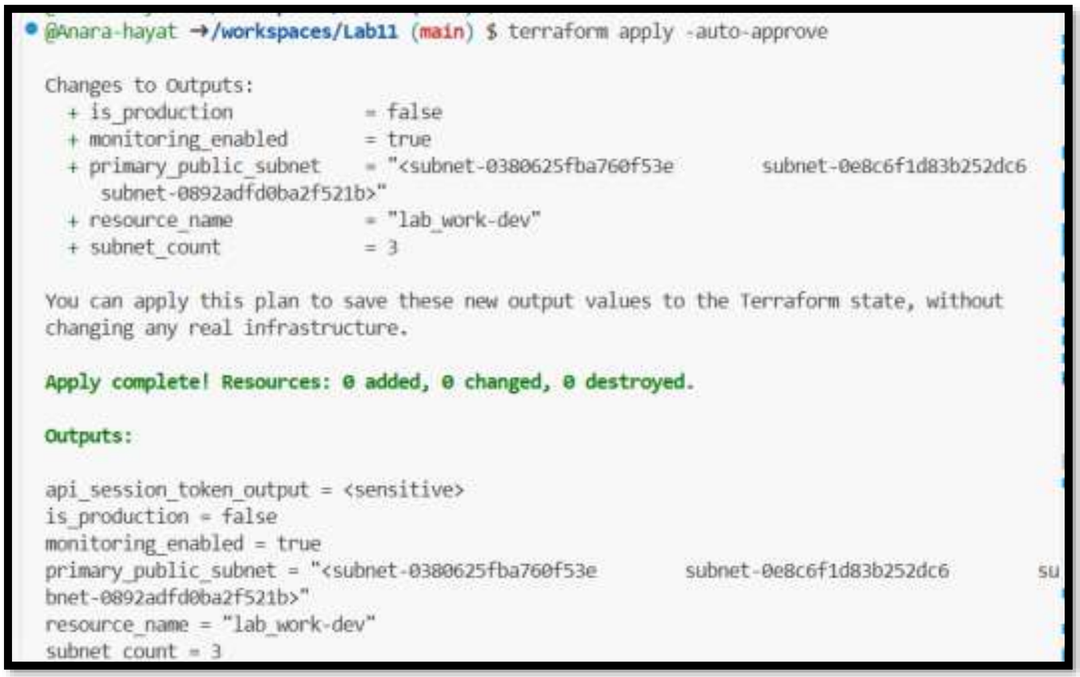


locals.tf U ×

locals.tf

```
locals {
  resource_name = "${var.project_name}-${var.environment}"
  primary_public_subnet = var.primary_subnet_id
  subnet_count          = var.subnet_count
  is_production         = var.environment == "prod"
  monitoring_enabled    = var.monitoring || local.is_production
}
```

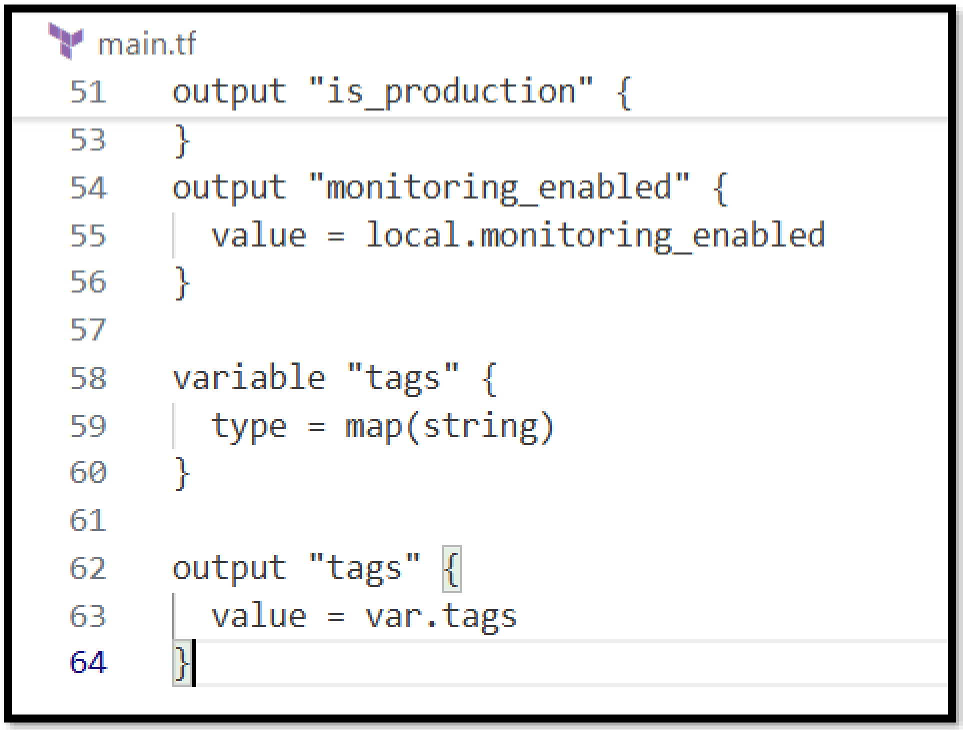
4. **Add outputs to main.tf:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

Changes to Outputs:
  + is_production            = false
  + monitoring_enabled       = true
  + primary_public_subnet    = "<subnet-0380625fba760f53e       subnet-0e8c6f1d83b252dc6
    subnet-0892adfd0ba2f521b>"
  + resource_name            = "lab_work-dev"
  + subnet_count             = 3

You can apply this plan to save these new output values to the Terraform state, without
changing any real infrastructure.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_session_token_output = <sensitive>
is_production = false
monitoring_enabled = true
primary_public_subnet = "<subnet-0380625fba760f53e       subnet-0e8c6f1d83b252dc6       su
bnet-0892adfd0ba2f521b>"
resource_name = "lab_work-dev"
subnet_count = 3
```

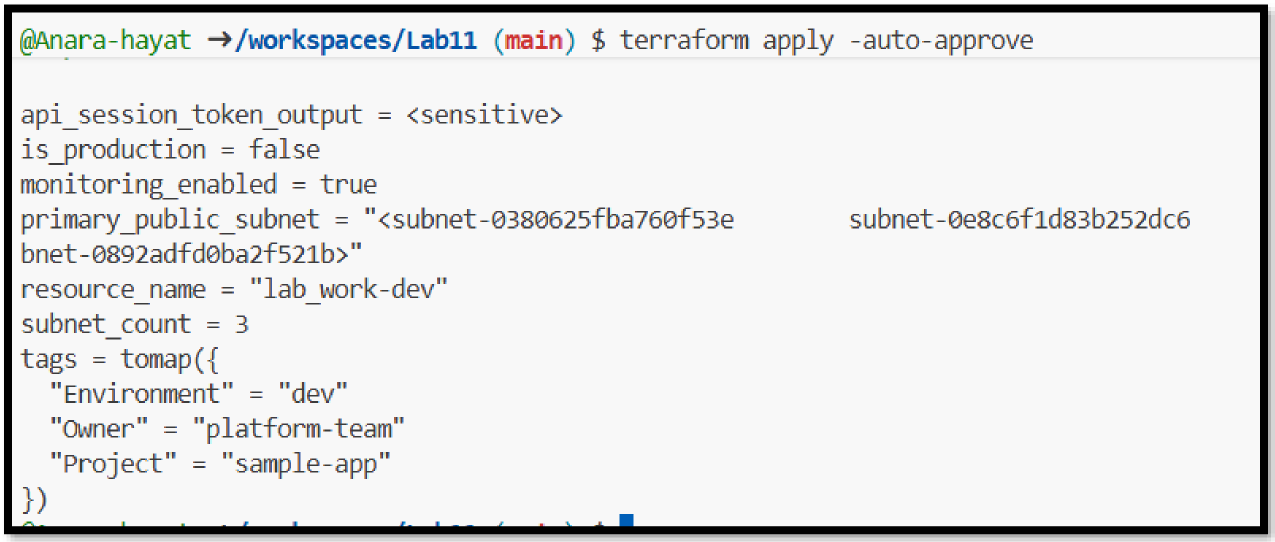
Task 4 — Maps and Objects

1. **Map variable in main.tf:**



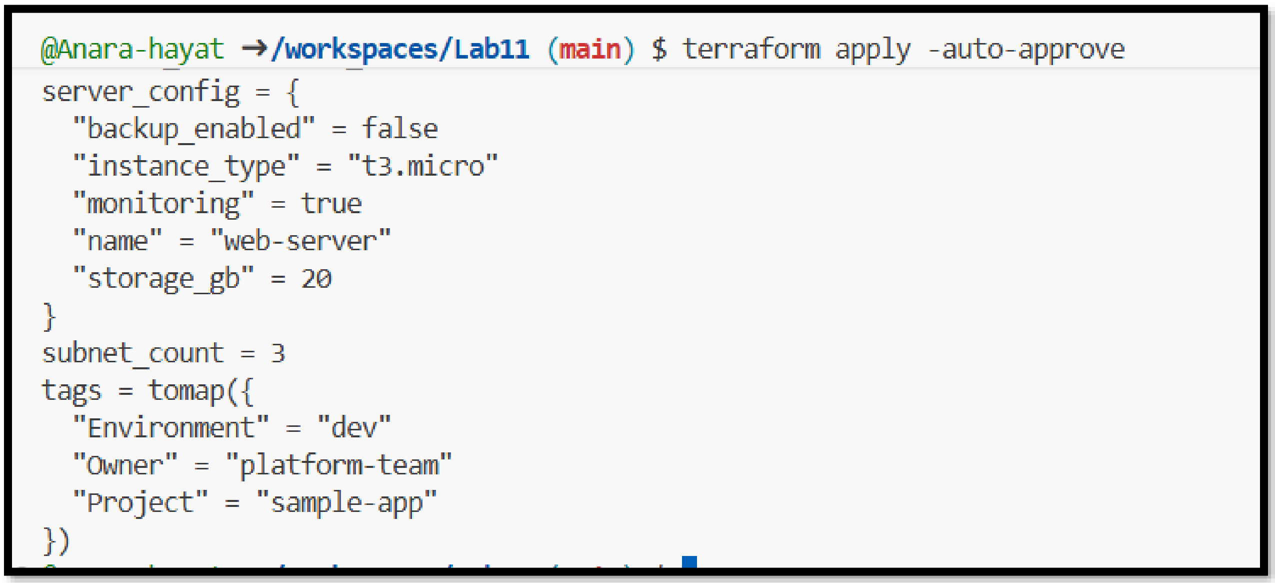
```
main.tf
output "is_production" {
}
output "monitoring_enabled" {
  value = local.monitoring_enabled
}

variable "tags" {
  type = map(string)
}

output "tags" {
  value = var.tags
}
```

2. **In terraform.tfvars:**



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve

api_session_token_output = <sensitive>
is_production = false
monitoring_enabled = true
primary_public_subnet = "<subnet-0380625fba760f53e       subnet-0e8c6f1d83b252dc6
bnet-0892adfd0ba2f521b>"
resource_name = "lab_work-dev"
subnet_count = 3
tags = tomap({
  "Environment" = "dev"
  "Owner" = "platform-team"
  "Project" = "sample-app"
})
```

3. **Define object variable:**

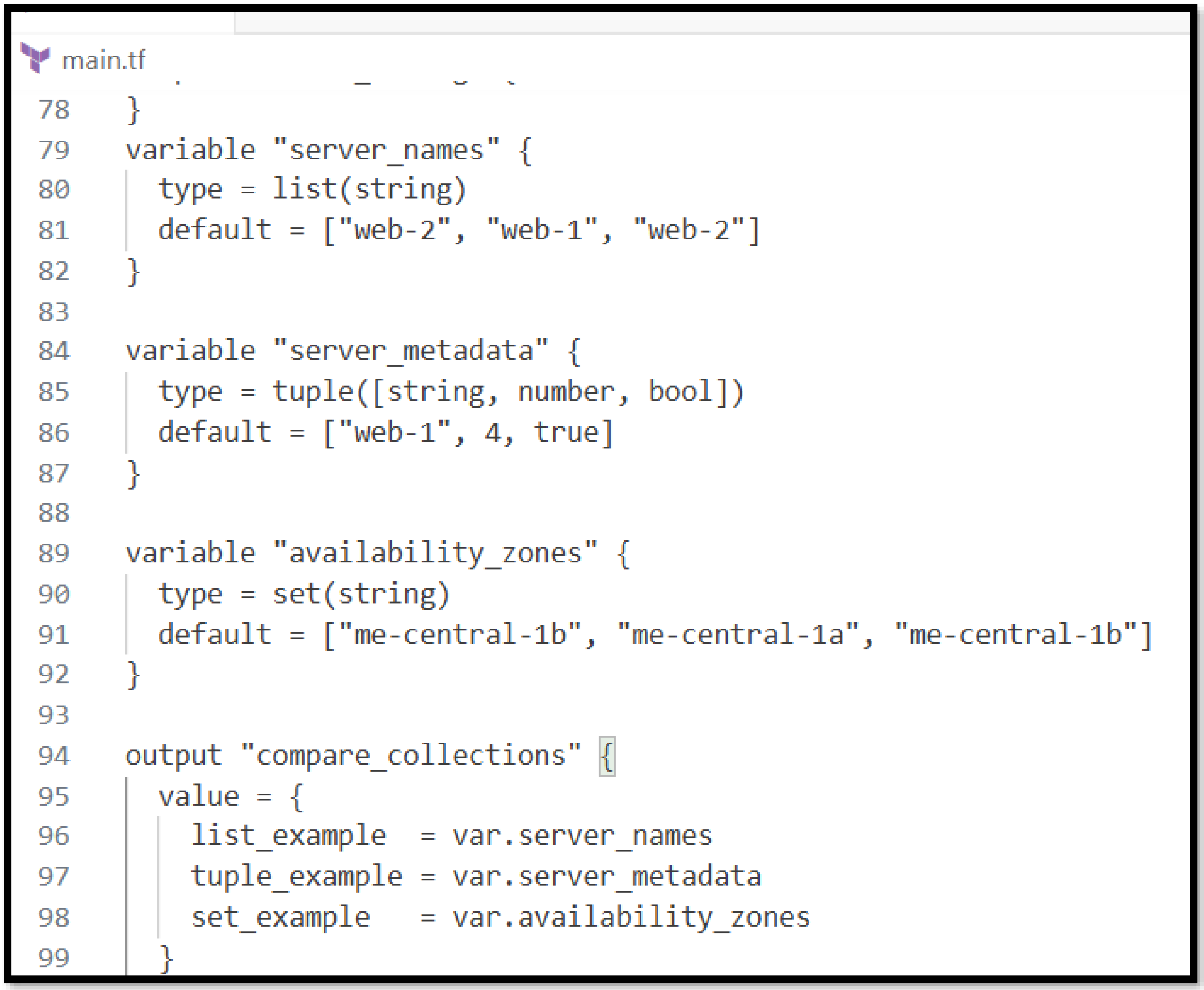


```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
subnet_count = 3
tags = tomap({
  "Environment" = "dev"
  "Owner" = "platform-team"
  "Project" = "sample-app"
})
```

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

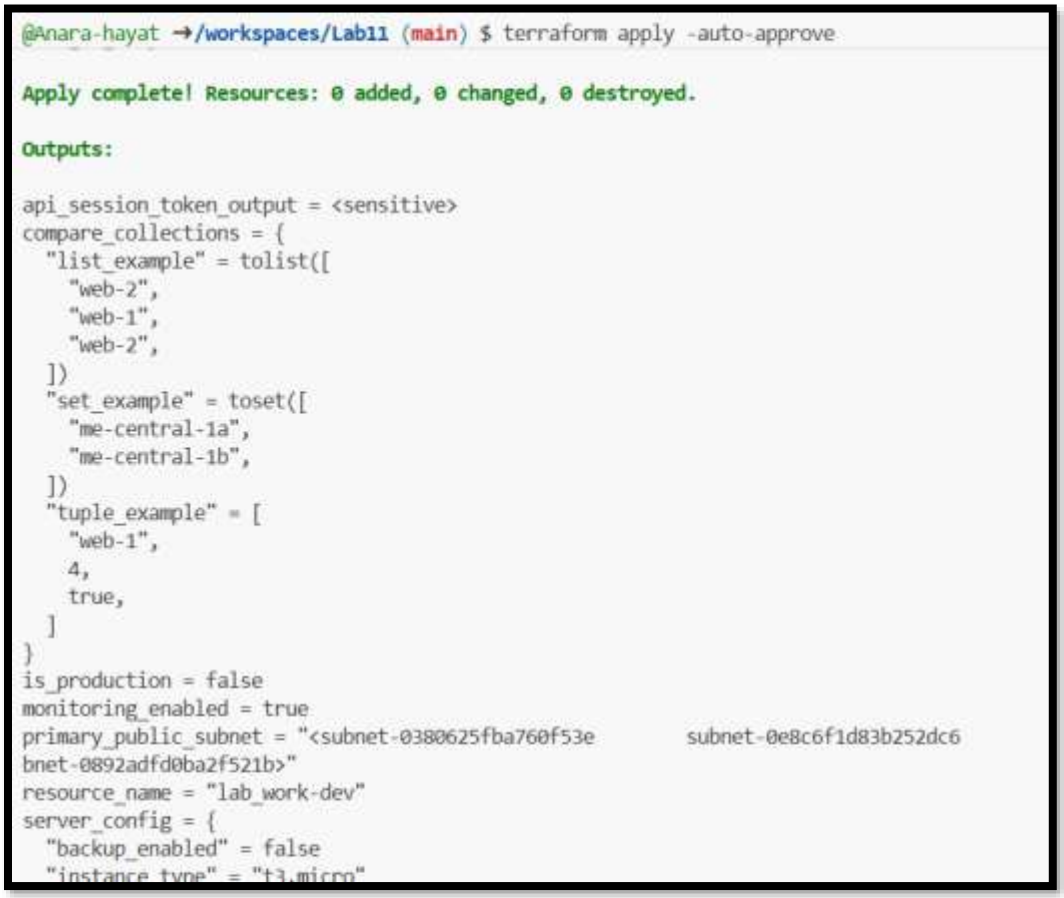
In this task you will define collection variables (list, tuple, set), observe their behavior, then perform mutations via locals and compare results.

1. **In main.tf define:**


main.tf

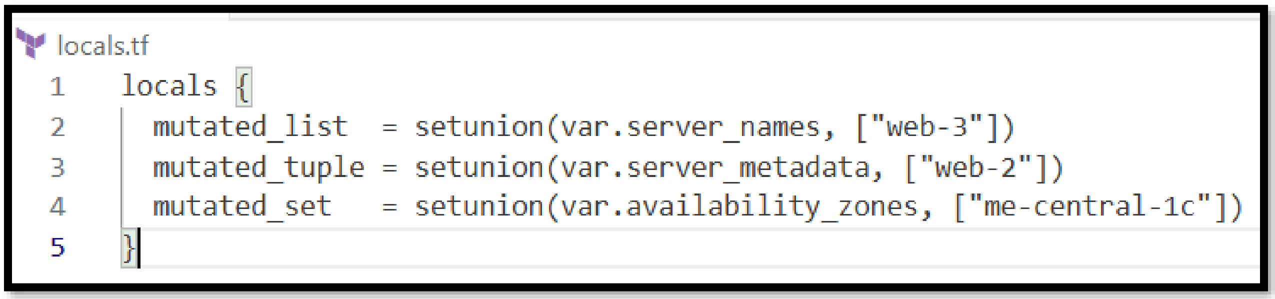
```
}
variable "server_names" {
  type = list(string)
  default = ["web-2", "web-1", "web-2"]
}

variable "server_metadata" {
  type = tuple([string, number, bool])
  default = ["web-1", 4, true]
}

variable "availability_zones" {
  type = set(string)
  default = ["me-central-1b", "me-central-1a", "me-central-1b"]
}

output "compare_collections" {
  value = {
    list_example  = var.server_names
    tuple_example = var.server_metadata
    set_example   = var.availability_zones
  }
```

2. **Run:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_session_token_output = <sensitive>
compare_collections = {
  "list_example" = tolist([
    "web-2",
    "web-1",
    "web-2",
  ])
  "set_example" = toset([
    "me-central-1a",
    "me-central-1b",
  ])
  "tuple_example" = [
    "web-1",
    4,
    true,
  ]
}
is_production = false
monitoring_enabled = true
primary_public_subnet = "<subnet-0380625fba760f53e       subnet-0e8c6f1d83b252dc6
bnet-0892adfd0ba2f521b>"
resource_name = "lab_work-dev"
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
```

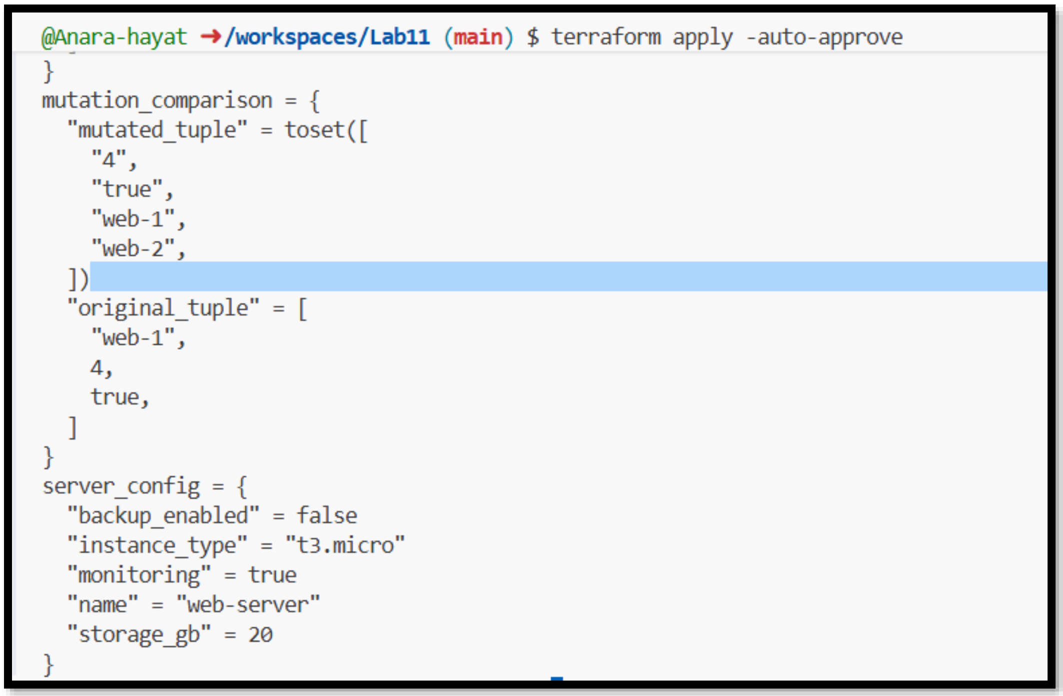
3. **In locals.tf add mutations: Create or edit locals.tf and add the following locals to demonstrate mutation behavior. Note: tuples are immutable in Terraform's type system, but many operations convert them to lists for evaluation.**



locals.tf

```
locals {
  mutated_list  = setunion(var.server_names, ["web-3"])
  mutated_tuple = setunion(var.server_metadata, ["web-2"])
  mutated_set   = setunion(var.availability_zones, ["me-central-1c"])
}
```

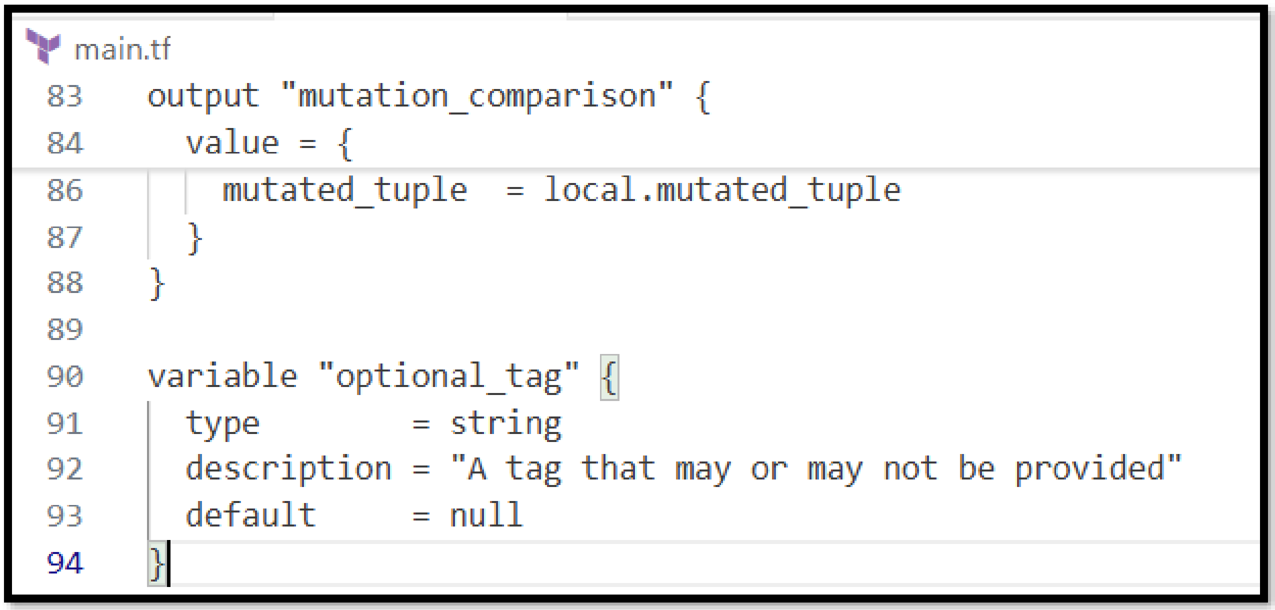
4. **Add comparison output in main.tf:**



```
@Anara-hayat ➜ /workspaces/Lab11 (main) $ terraform apply -auto-approve
}
mutation_comparison = {
  "mutated_tuple" = toset([
    "4",
    "true",
    "web-1",
    "web-2",
  ])
  "original_tuple" = [
    "web-1",
    4,
    true,
  ]
}
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
```

Task 6 — Null, any type & dynamic values

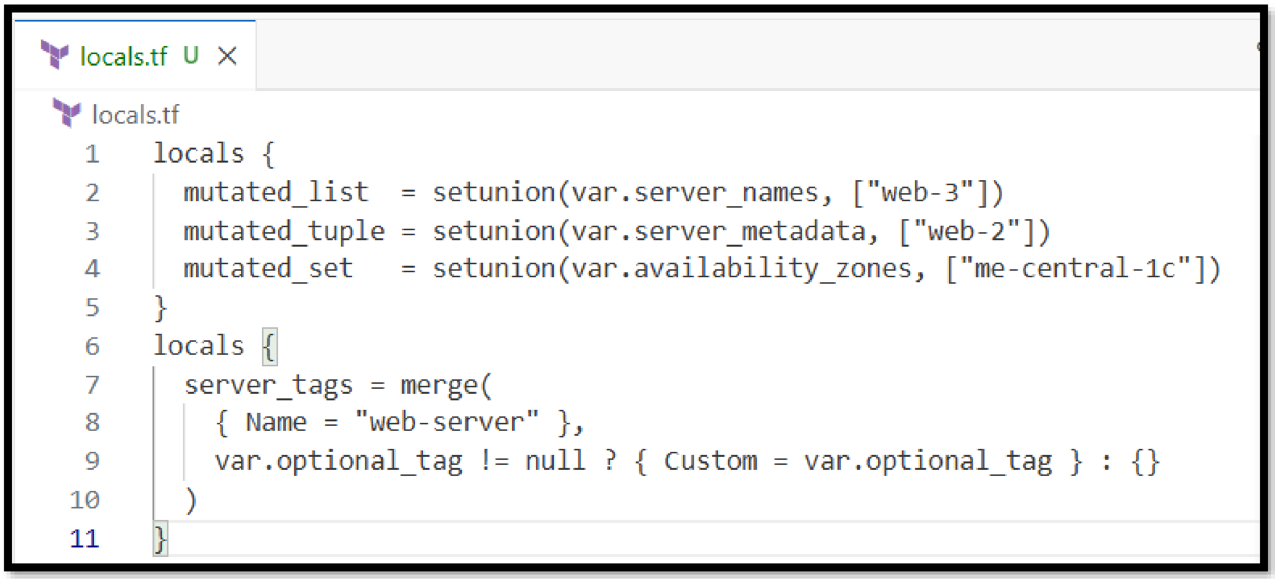
1. **Null variable:**



main.tf

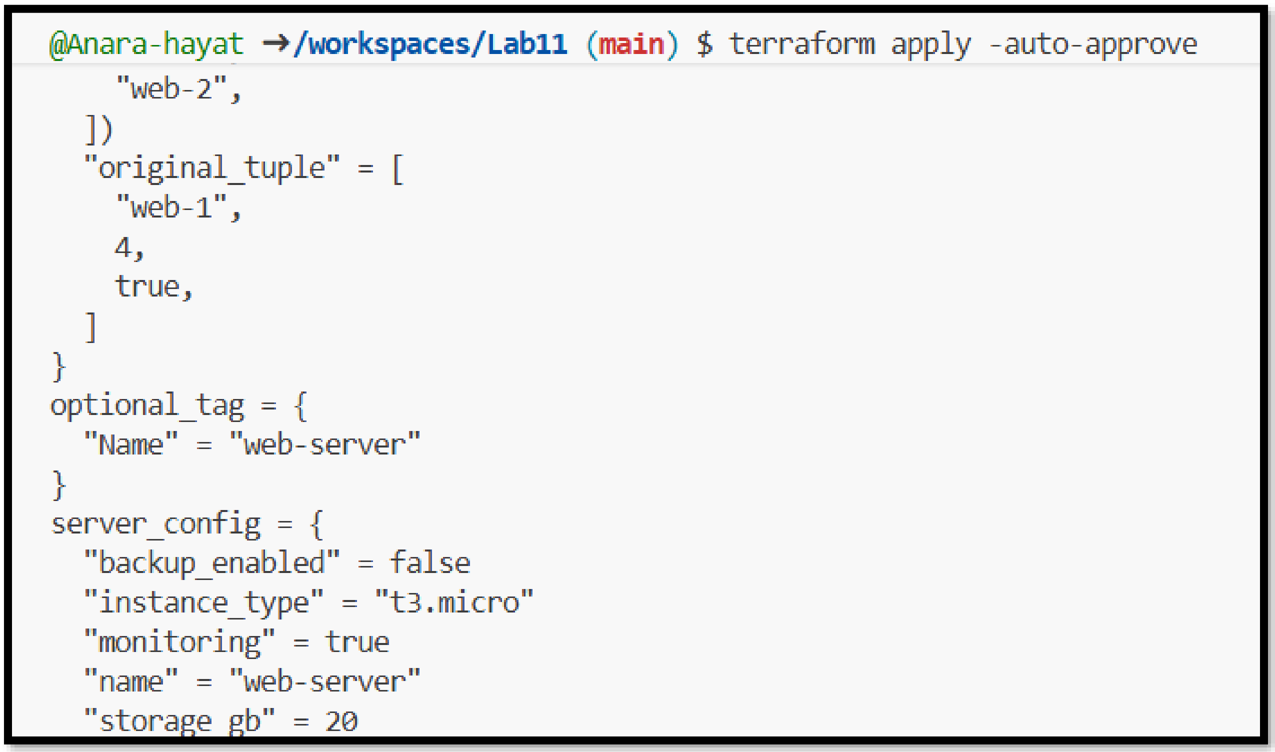
```
output "mutation_comparison" {
  value = {
    mutated_tuple  = local.mutated_tuple
  }
}

variable "optional_tag" {
  type        = string
  description = "A tag that may or may not be provided"
  default     = null
}
```

2. **Merge tags in locals.tf:**



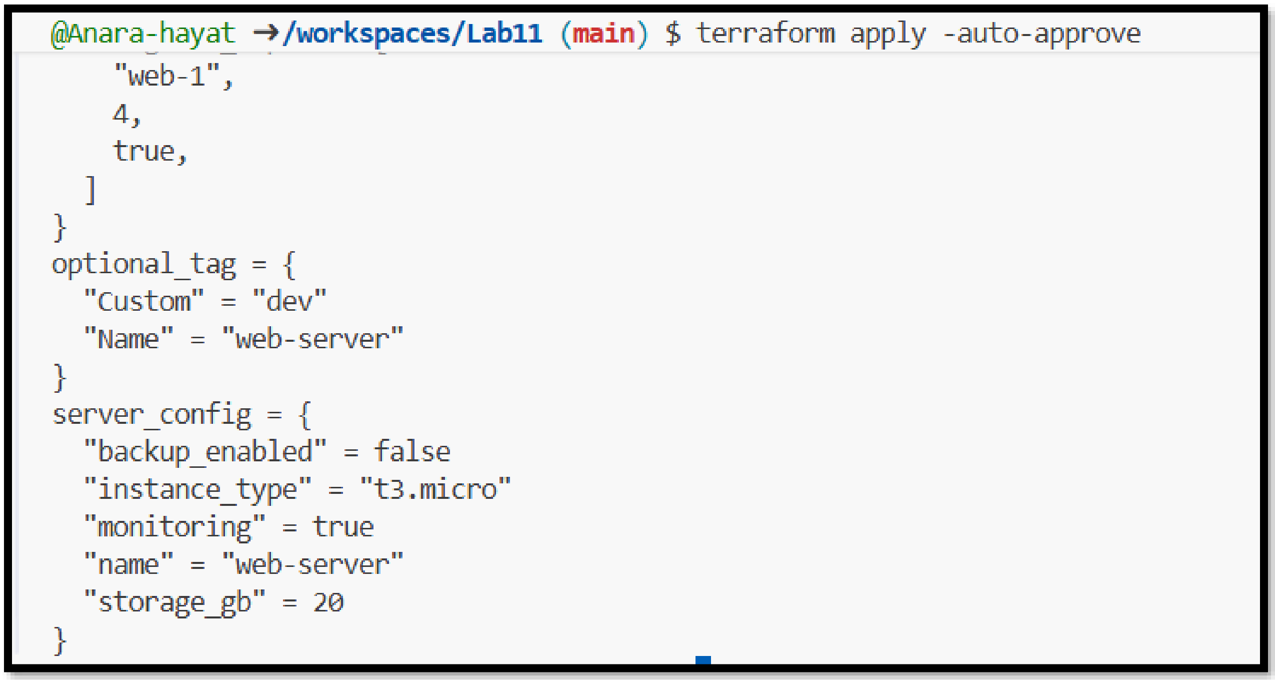
```
locals {
  mutated_list  = setunion(var.server_names, ["web-3"])
  mutated_tuple = setunion(var.server_metadata, ["web-2"])
  mutated_set   = setunion(var.availability_zones, ["me-central-1c"])
}
locals {
  server_tags = merge(
    { Name = "web-server" },
    var.optional_tag != null ? { Custom = var.optional_tag } : {}
  )
}
```

3. Output:



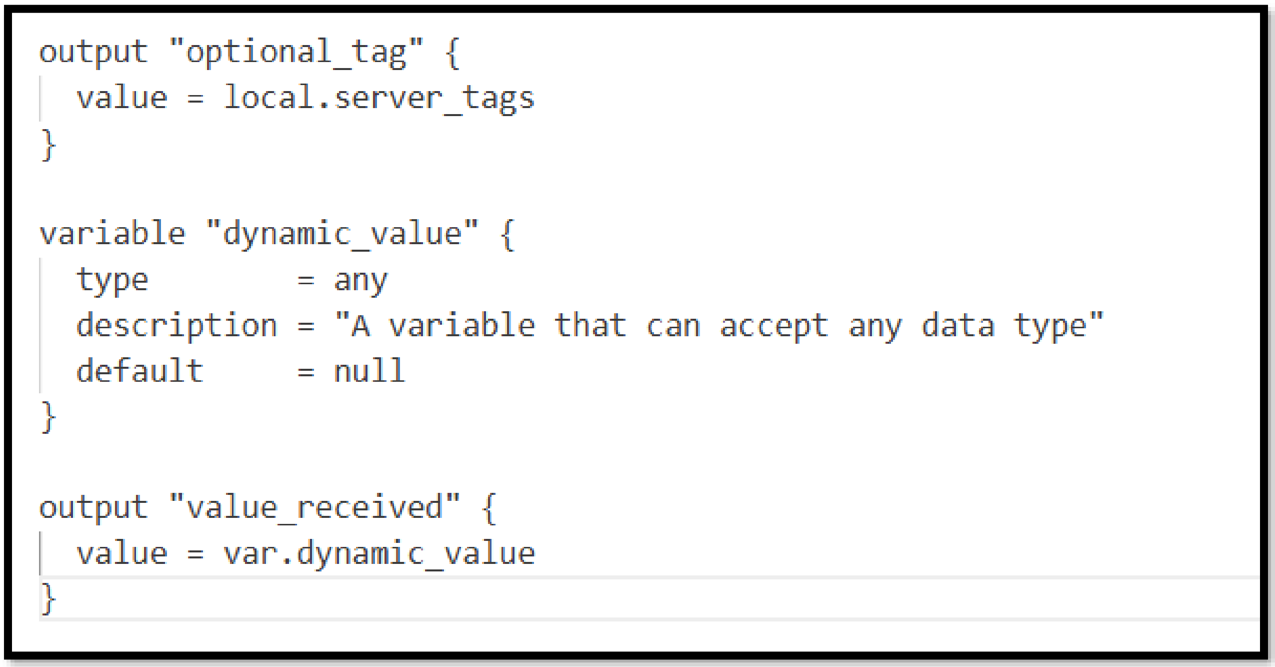
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
    "web-2",
  ])
  "original_tuple" = [
    "web-1",
    4,
    true,
  ]
}
optional_tag = {
  "Name" = "web-server"
}
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
```

4. Add in terraform.tfvars:



```
@Anara-hayat ➜ /workspaces/Lab11 (main) $ terraform apply -auto-approve
      "web-1",
      4,
      true,
    ]
  }
  optional_tag = {
    "Custom" = "dev"
    "Name" = "web-server"
  }
  server_config = {
    "backup_enabled" = false
    "instance_type" = "t3.micro"
    "monitoring" = true
    "name" = "web-server"
    "storage_gb" = 20
  }
```

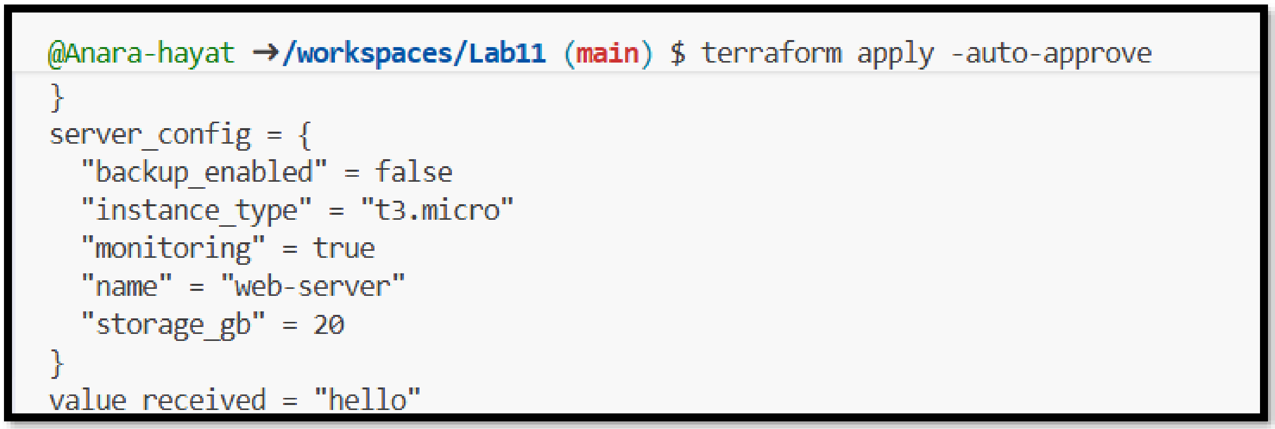
5. **Any type variable:**



```
output "optional_tag" {
  value = local.server_tags
}

variable "dynamic_value" {
  type        = any
  description = "A variable that can accept any data type"
  default     = null
}

output "value_received" {
  value = var.dynamic_value
}
```

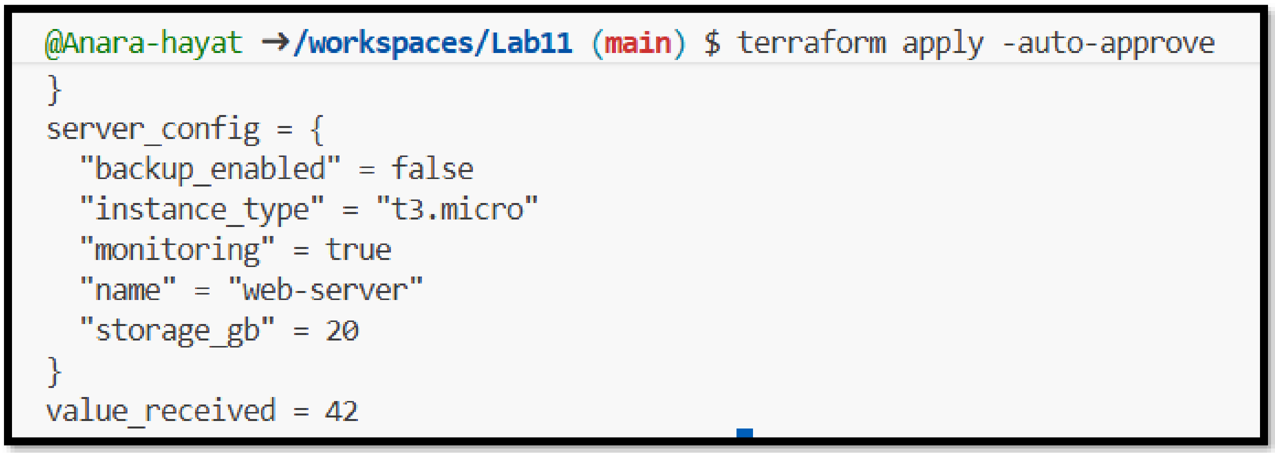
6. **Now test the dynamic (any) variable with different types. For each change, update terraform.tfvars with the specified value and run terraform apply -auto-approve, then capture the output.**

a.String



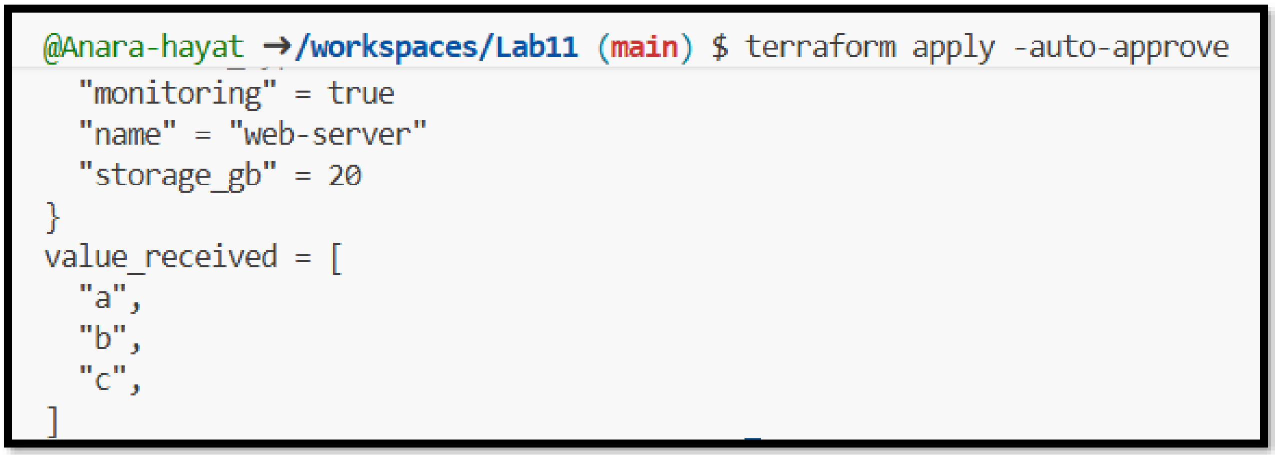
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
}
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
value_received = "hello"
```

b. Number



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
}
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
value_received = 42
```

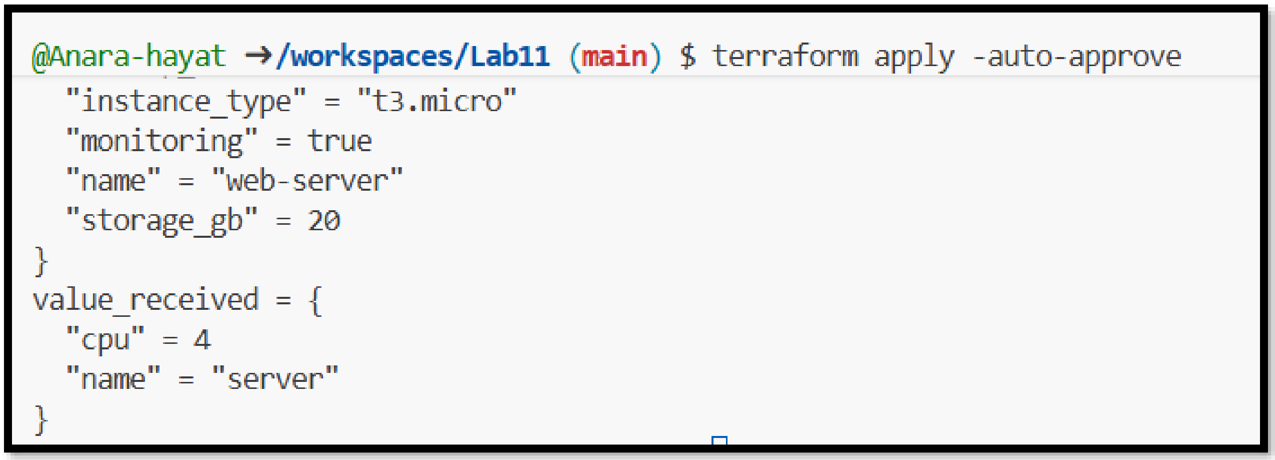
c. list



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
value_received = [
  "a",
  "b",
  "c",
]
```

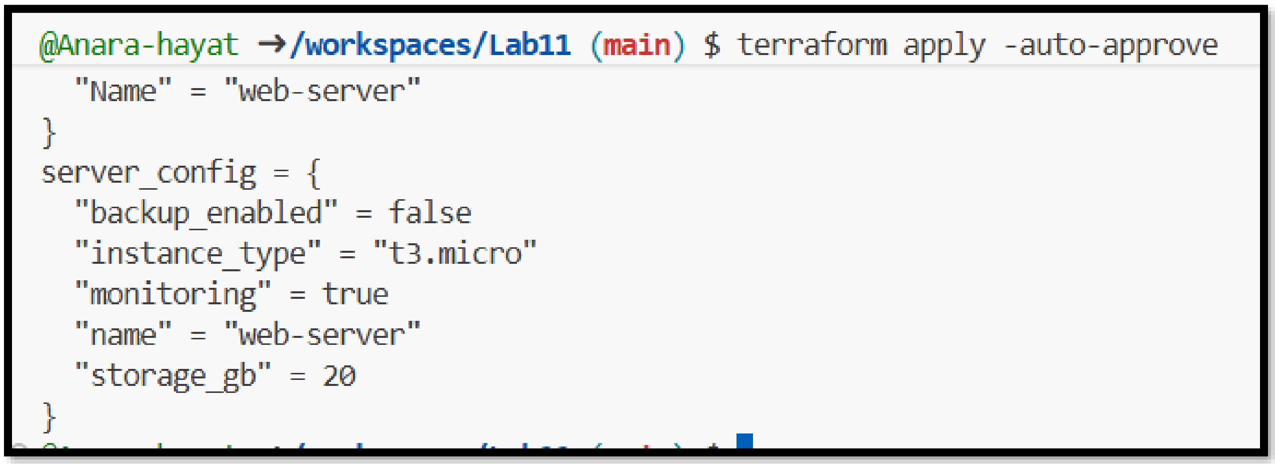
d) Map / Object

- **Change terraform.tfvars:**



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
value_received = {
  "cpu" = 4
  "name" = "server"
}
```

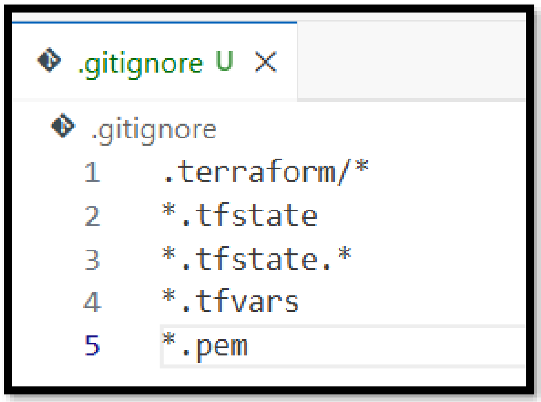
e) Null



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
  "Name" = "web-server"
}
server_config = {
  "backup_enabled" = false
  "instance_type" = "t3.micro"
  "monitoring" = true
  "name" = "web-server"
  "storage_gb" = 20
}
```

Task 7 — Git ignore

Create .gitignore:



```
.gitignore U ×
.gitignore
.terraform/*
*.tfstate
*.tfstate.*
*.tfvars
*.pem
```

Task 8 — Clean-up then build real infra (VPC, Subnet, IGW, routing, default route table)

In this task you will clean previous example values and build a simple VPC + Subnet, attach an Internet Gateway, and configure routing (first with a custom route table and association, then switch to the default route table).

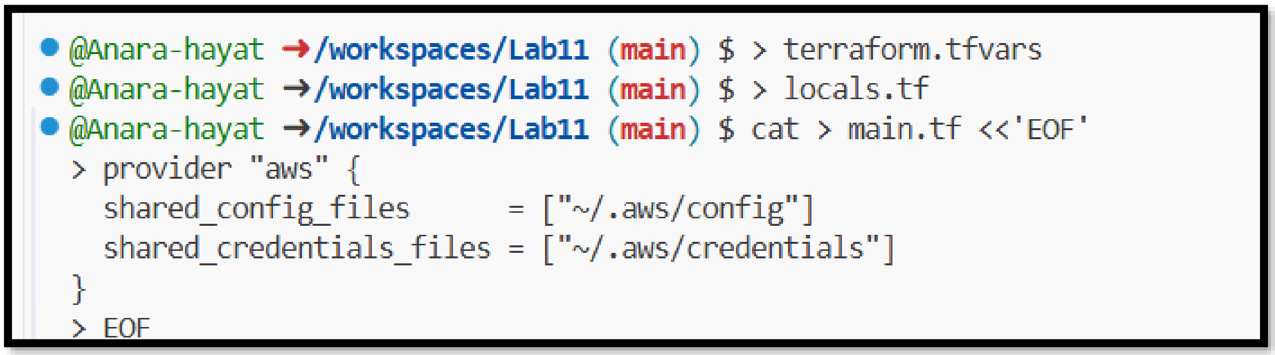
Perform all commands inside your Codespace shell.

1. **Clean previous files**

1. **Clean previous files**

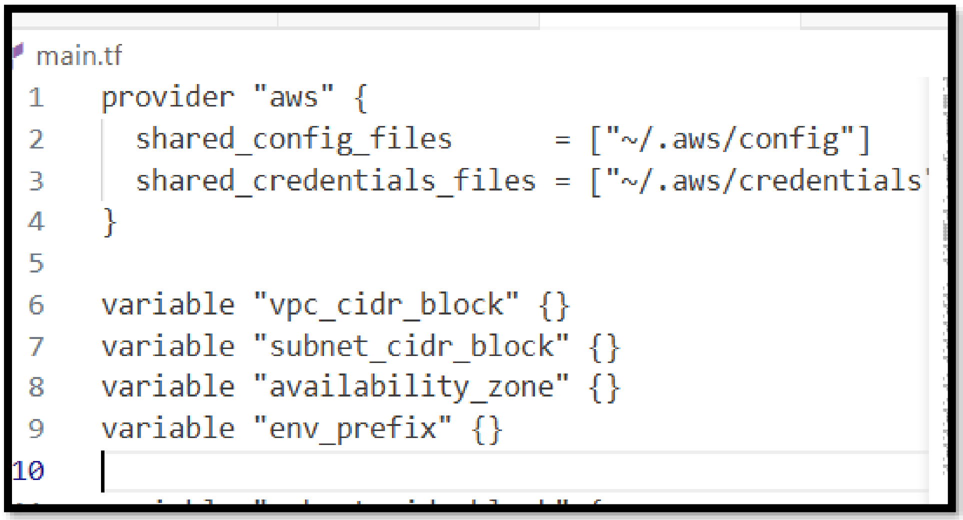
- **Remove all variable assignments from terraform.tfvars (empty the file or delete it).**
- **Remove all content from locals.tf (delete or empty the file).**
- **Replace main.tf contents with only the provider block below (start fresh).**

Provider block to put in main.tf:



```
● @Anara-hayat ➜/workspaces/Lab11 (main) $ > terraform.tfvars
● @Anara-hayat ➜/workspaces/Lab11 (main) $ > locals.tf
● @Anara-hayat ➜/workspaces/Lab11 (main) $ cat > main.tf <<'EOF'
  > provider "aws" {
    shared_config_files      = ["~/.aws/config"]
    shared_credentials_files = ["~/.aws/credentials"]
  }
  > EOF
```

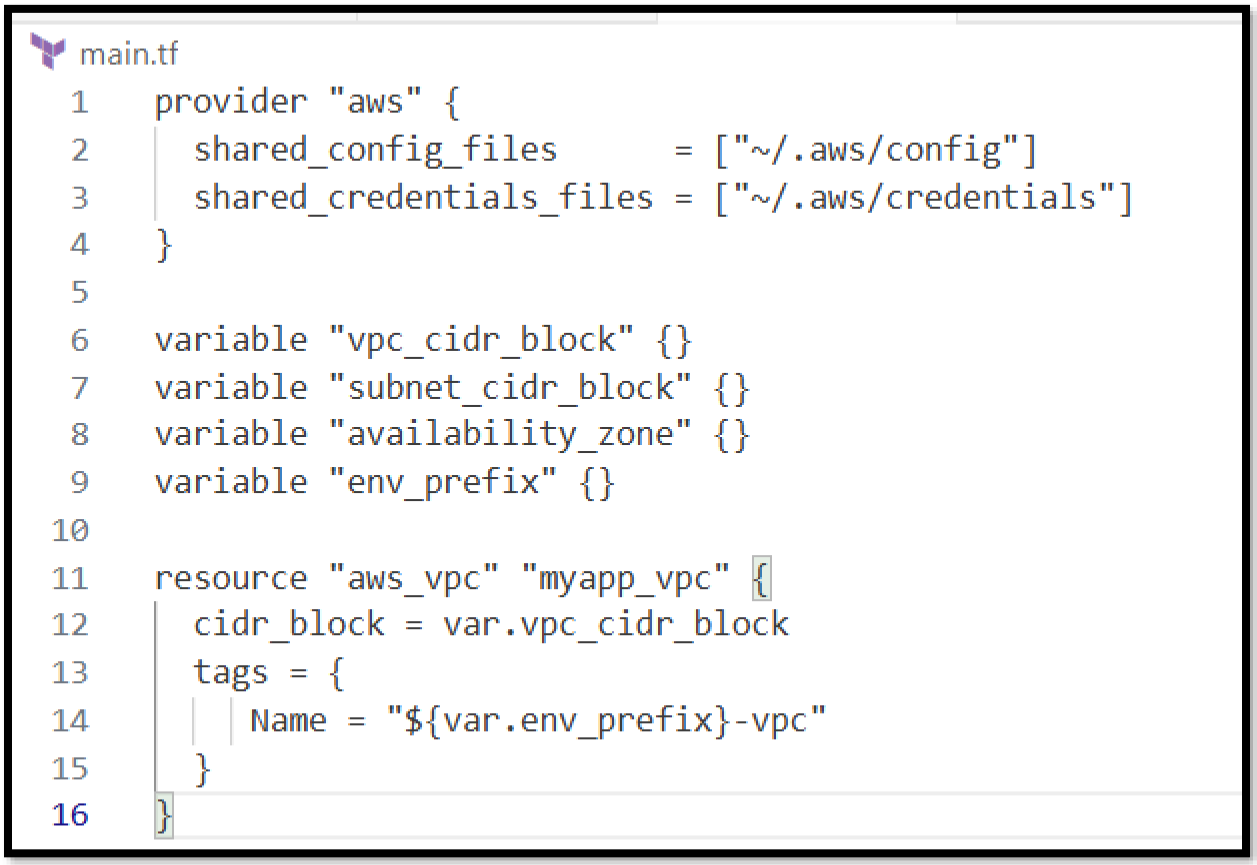
2. **Define variables in main.tf Add these variable declarations to main.tf (below the provider block):**



main.tf

```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials'
}

variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}

```

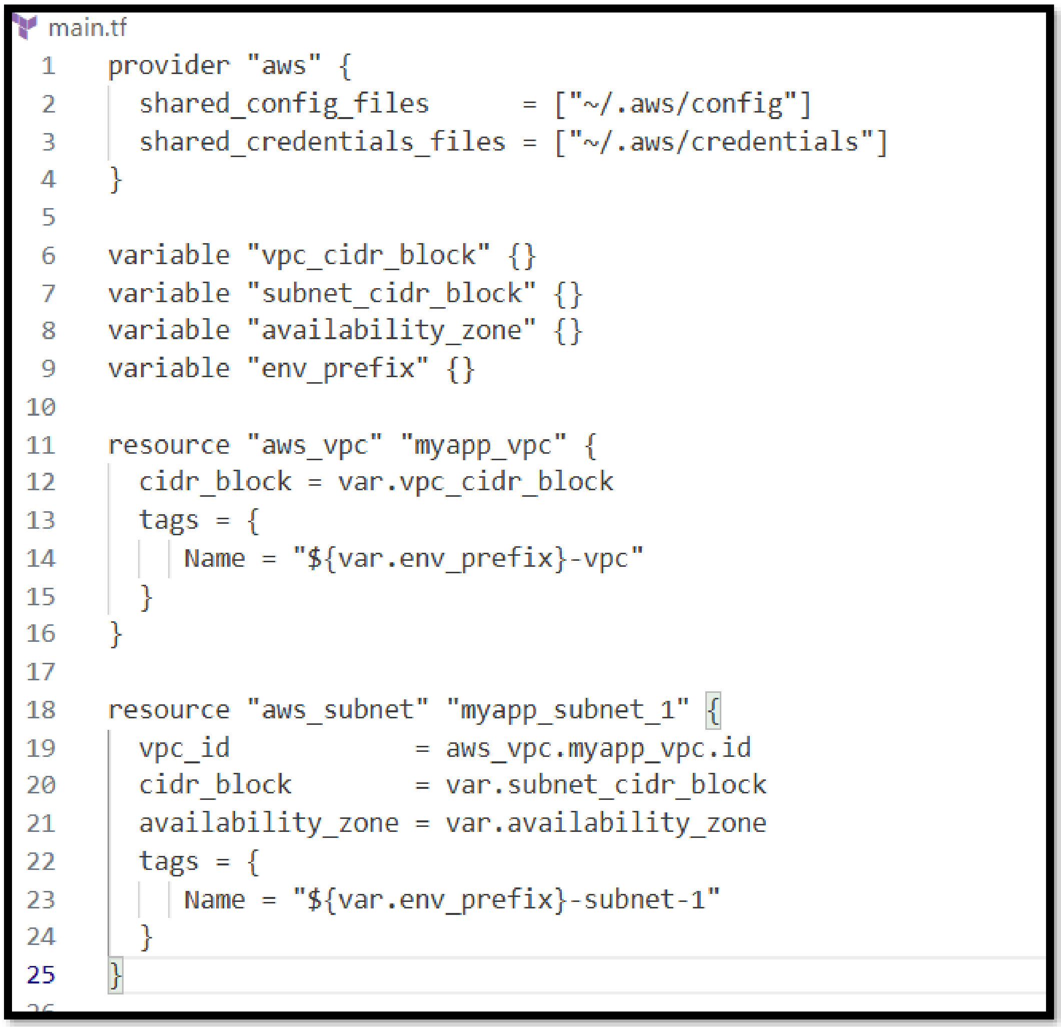
3. **Create VPC in main.tf Add the VPC resource to main.tf:**



main.tf

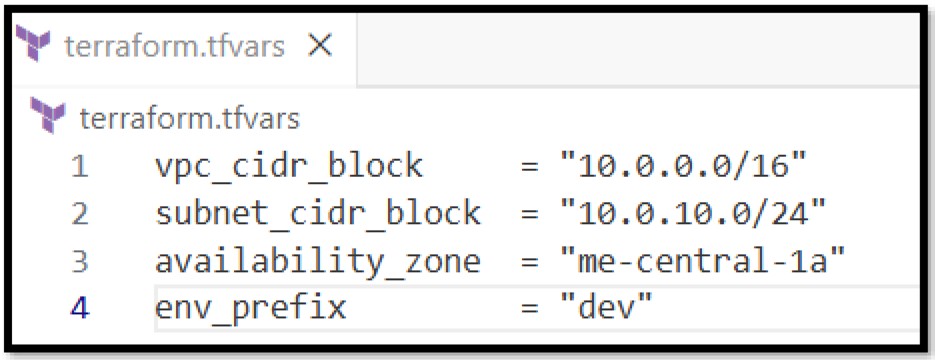
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}

resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}
```

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


main.tf

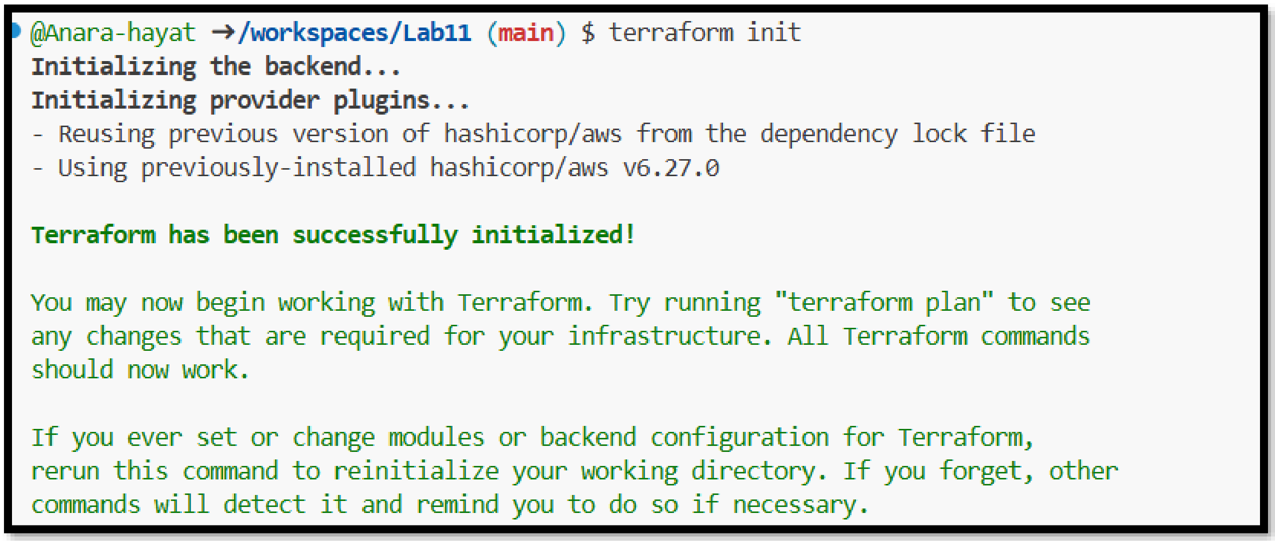
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}

resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

resource "aws_subnet" "myapp_subnet_1" {
  vpc_id            = aws_vpc.myapp_vpc.id
  cidr_block        = var.subnet_cidr_block
  availability_zone = var.availability_zone
  tags = {
    Name = "${var.env_prefix}-subnet-1"
  }
}
```

5. **Populate terraform.tfvars In terraform.tfvars add:**

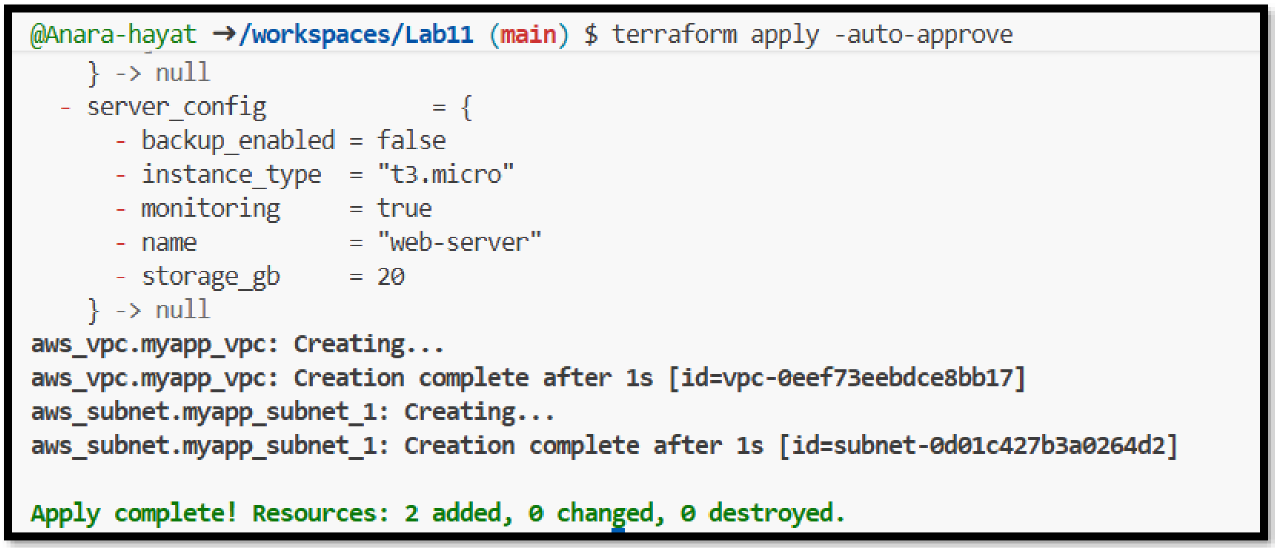

terraform.tfvars

```
vpc_cidr_block    = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix        = "dev"
```

6. **Apply to create VPC and Subnet Initialize (if needed) and apply:**

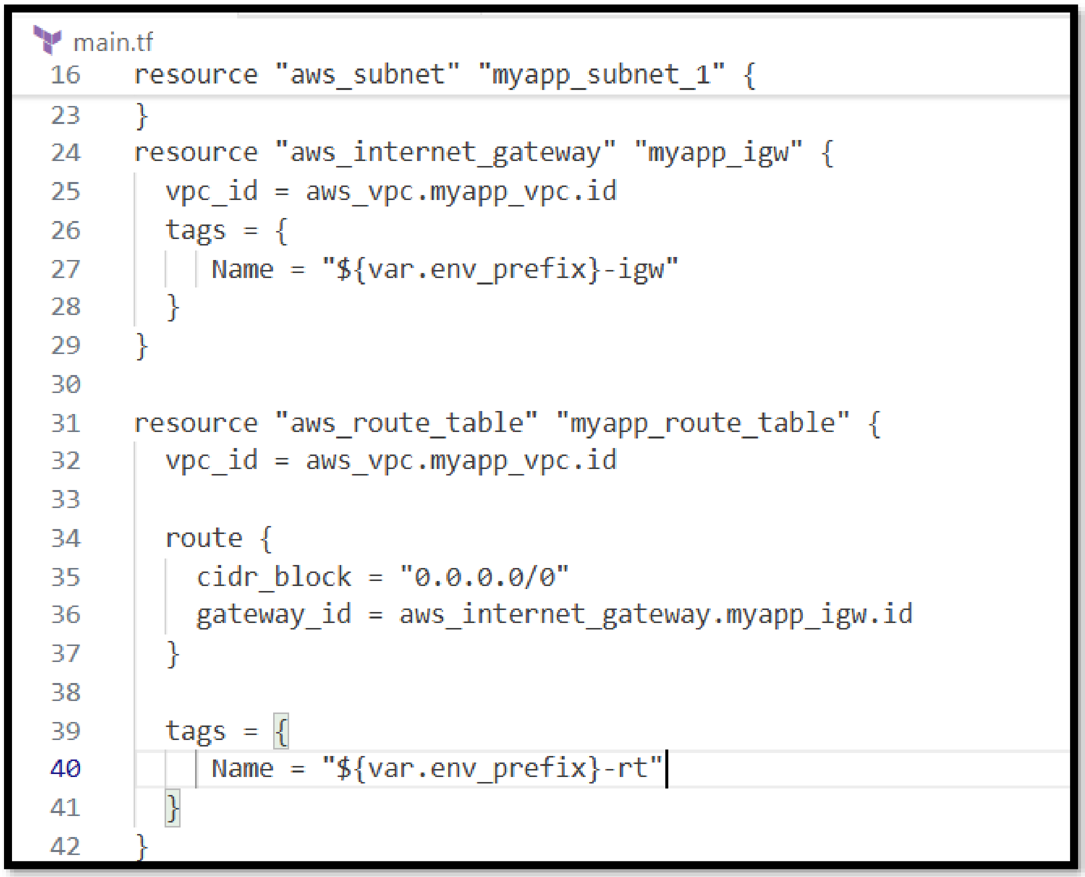


```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```



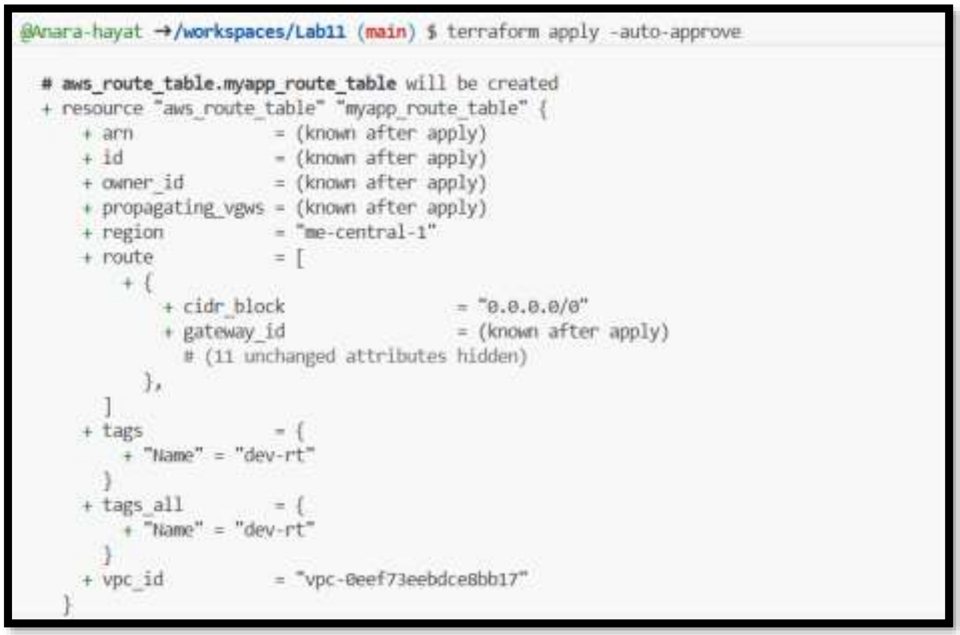
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
    } -> null
  - server_config           = {
      - backup_enabled = false
      - instance_type  = "t3.micro"
      - monitoring     = true
      - name           = "web-server"
      - storage_gb     = 20
    } -> null
aws_vpc.myapp_vpc: Creating...
aws_vpc.myapp_vpc: Creation complete after 1s [id=vpc-0eef73eebdce8bb17]
aws_subnet.myapp_subnet_1: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-0d01c427b3a0264d2]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

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



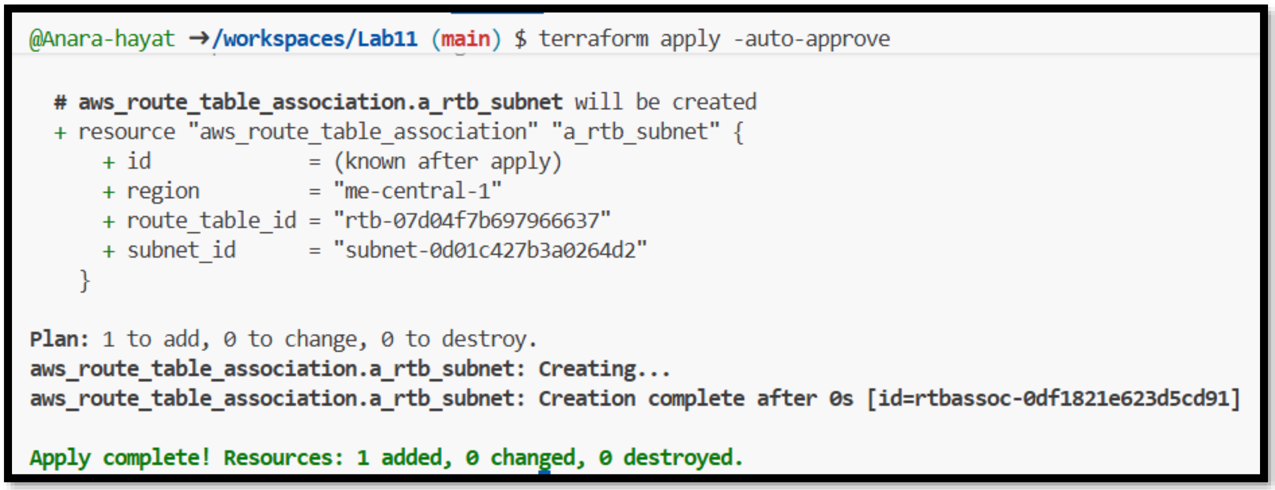
main.tf

```
resource "aws_subnet" "myapp_subnet_1" {
}
resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = aws_vpc.myapp_vpc.id
  tags = {
    Name = "${var.env_prefix}-igw"
  }
}

resource "aws_route_table" "myapp_route_table" {
  vpc_id = aws_vpc.myapp_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }

  tags = {
    Name = "${var.env_prefix}-rt"
  }
}
```



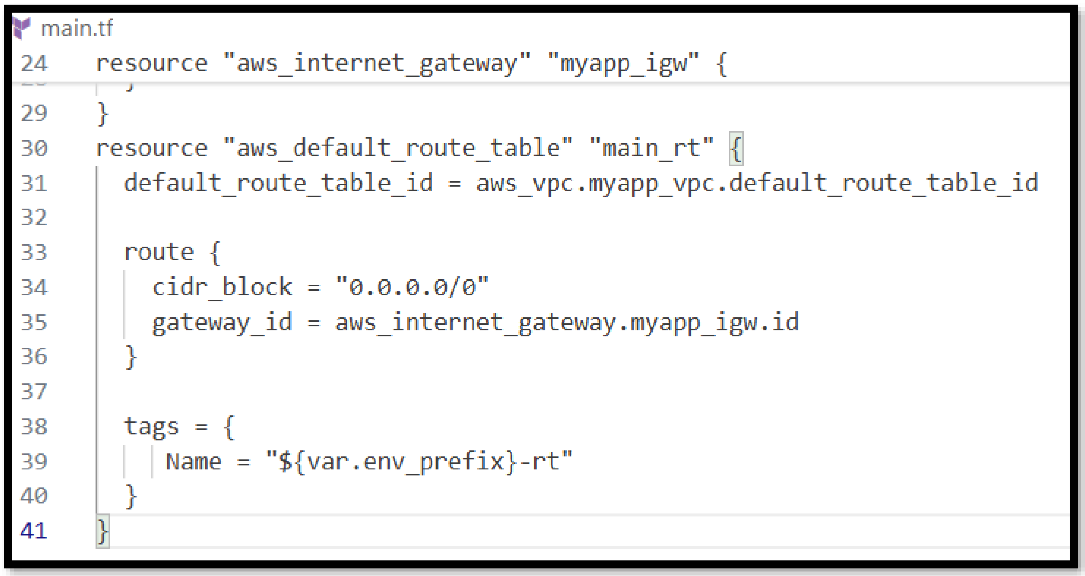
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve

  # aws_route_table.myapp_route_table will be created
  + resource "aws_route_table" "myapp_route_table" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + region           = "me-central-1"
      + route            = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + gateway_id                 = (known after apply)
                # (11 unchanged attributes hidden)
            },
        ]
      + tags             = {
          + "Name" = "dev-rt"
        }
      + tags_all         = {
          + "Name" = "dev-rt"
        }
      + vpc_id           = "vpc-0eef73eebdce8bb17"
    }
```

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



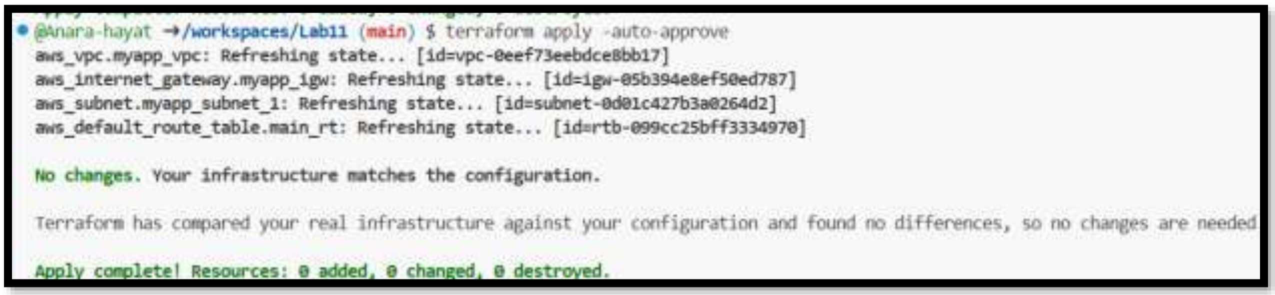
```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve

  # aws_route_table_association.a_rtb_subnet will be created
  + resource "aws_route_table_association" "a_rtb_subnet" {
      + id             = (known after apply)
      + region         = "me-central-1"
      + route_table_id = "rtb-07d04f7b697966637"
      + subnet_id      = "subnet-0d01c427b3a0264d2"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_route_table_association.a_rtb_subnet: Creating...
aws_route_table_association.a_rtb_subnet: Creation complete after 0s [id=rtbassoc-0df1821e623d5cd91]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

9. **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:**



main.tf

```
resource "aws_internet_gateway" "myapp_igw" {
}
resource "aws_default_route_table" "main_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }

  tags = {
    Name = "${var.env_prefix}-rt"
  }
}
```

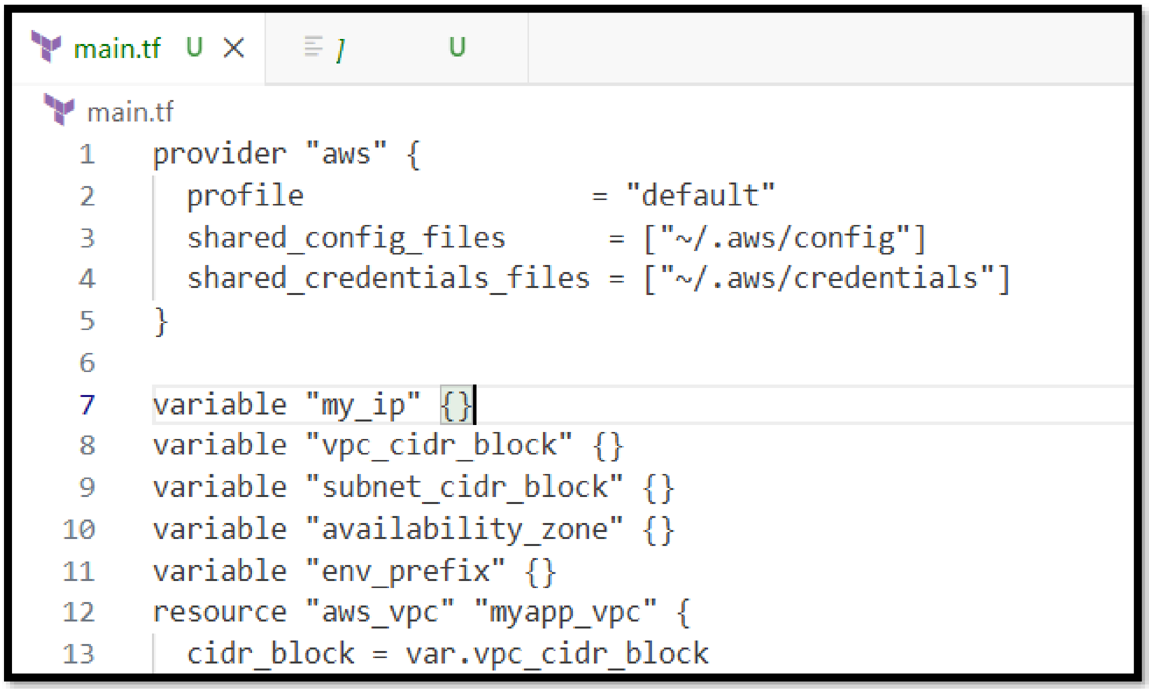


```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve
aws_vpc.myapp_vpc: Refreshing state... [id=vpc-0eef73eebdce8bb17]
aws_internet_gateway.myapp_igw: Refreshing state... [id=igw-05b394e8ef50ed787]
aws_subnet.myapp_subnet_1: Refreshing state... [id=subnet-0d01c427b3a0264d2]
aws_default_route_table.main_rt: Refreshing state... [id=rtb-099cc25bff3334970]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Task 9 — Security Group, Key Pair, EC2 Instance, user_data & nginx

This task walks you through creating a security group, creating an EC2 key pair, launching an EC2 instance, verifying SSH access, and installing nginx via user_data (inline and from a script). Perform all commands from your Codespace shell.

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

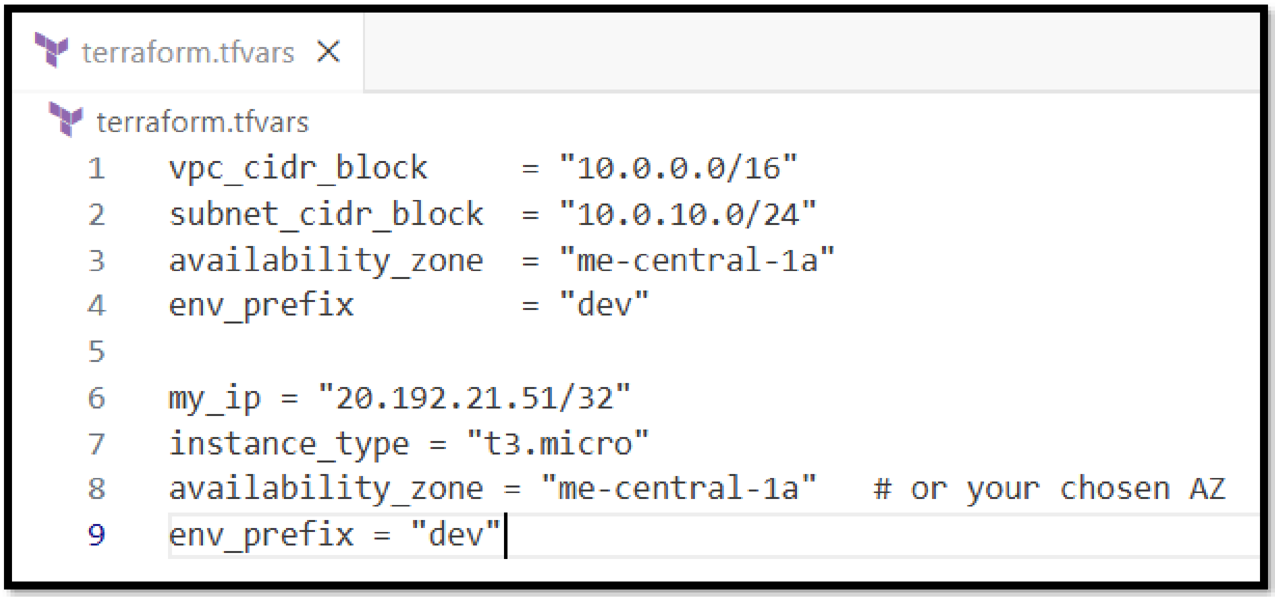


main.tf U ×] U

main.tf

```
provider "aws" {
  profile                  = "default"
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

variable "my_ip" {}
variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block
```

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

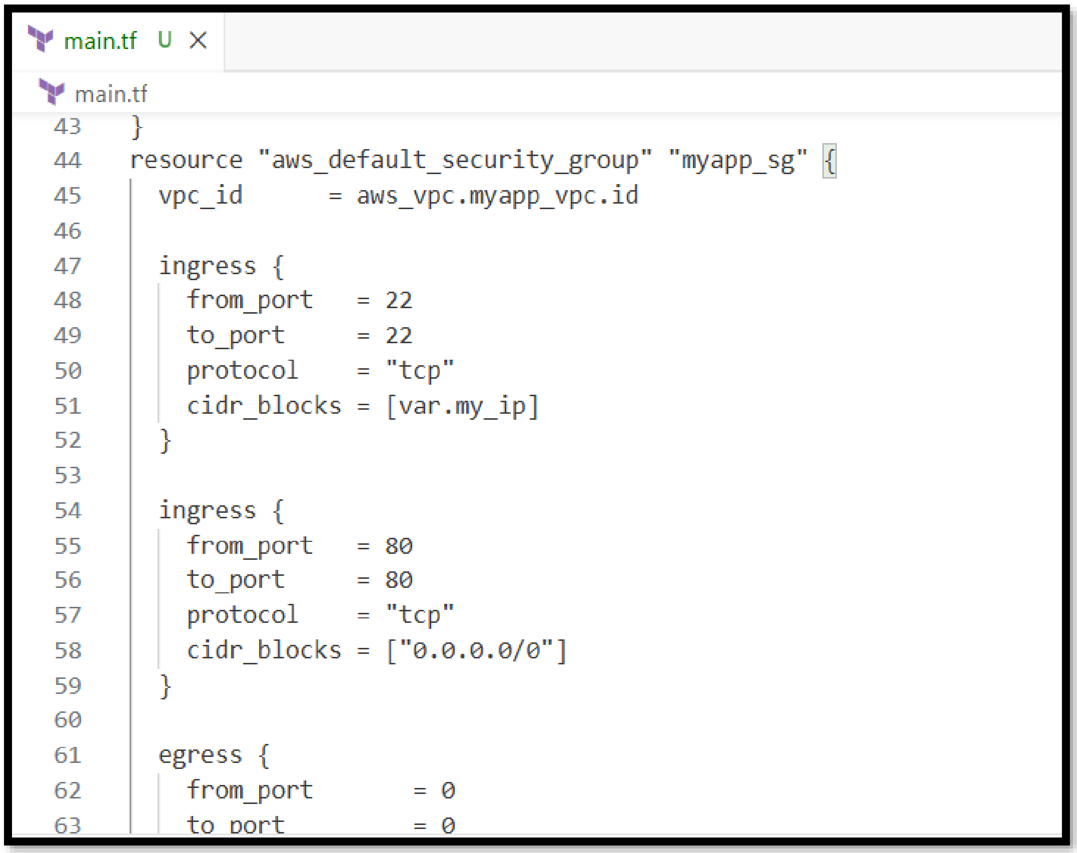


terraform.tfvars ×

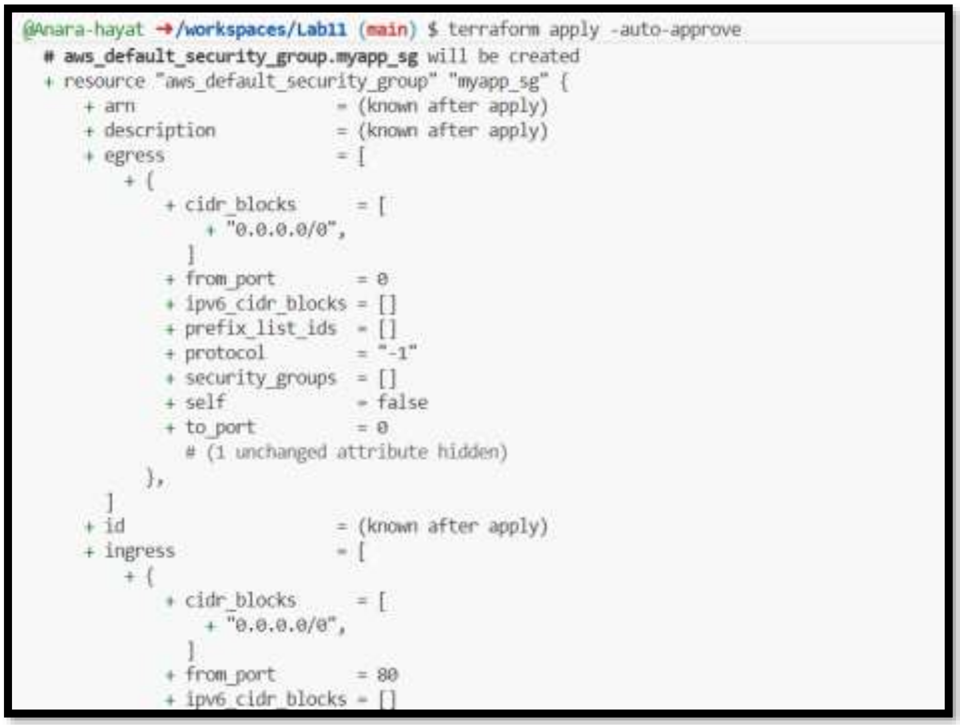
terraform.tfvars

```
vpc_cidr_block      = "10.0.0.0/16"
subnet_cidr_block   = "10.0.10.0/24"
availability_zone   = "me-central-1a"
env_prefix          = "dev"

my_ip = "20.192.21.51/32"
instance_type = "t3.micro"
availability_zone = "me-central-1a"   # or your chosen AZ
env_prefix = "dev"
```

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

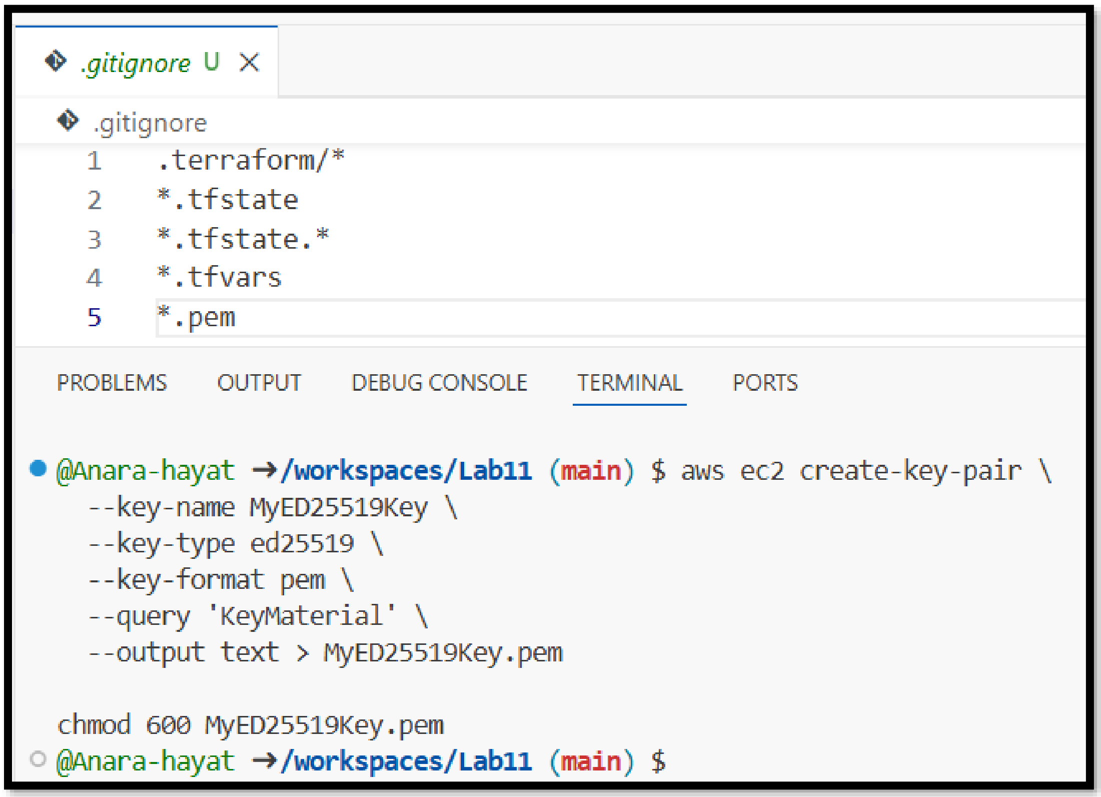


```
}
resource "aws_default_security_group" "myapp_sg" {
  vpc_id      = aws_vpc.myapp_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port       = 0
    to_port         = 0
```



```
@Anara-hayat ➜ /workspaces/Lab11 (main) $ terraform apply -auto-approve
  # aws_default_security_group.myapp_sg will be created
  + resource "aws_default_security_group" "myapp_sg" {
      + arn                    = (known after apply)
      + description            = (known after apply)
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
                # (1 unchanged attribute hidden)
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 80
              + ipv6_cidr_blocks = []
```

4. **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.**

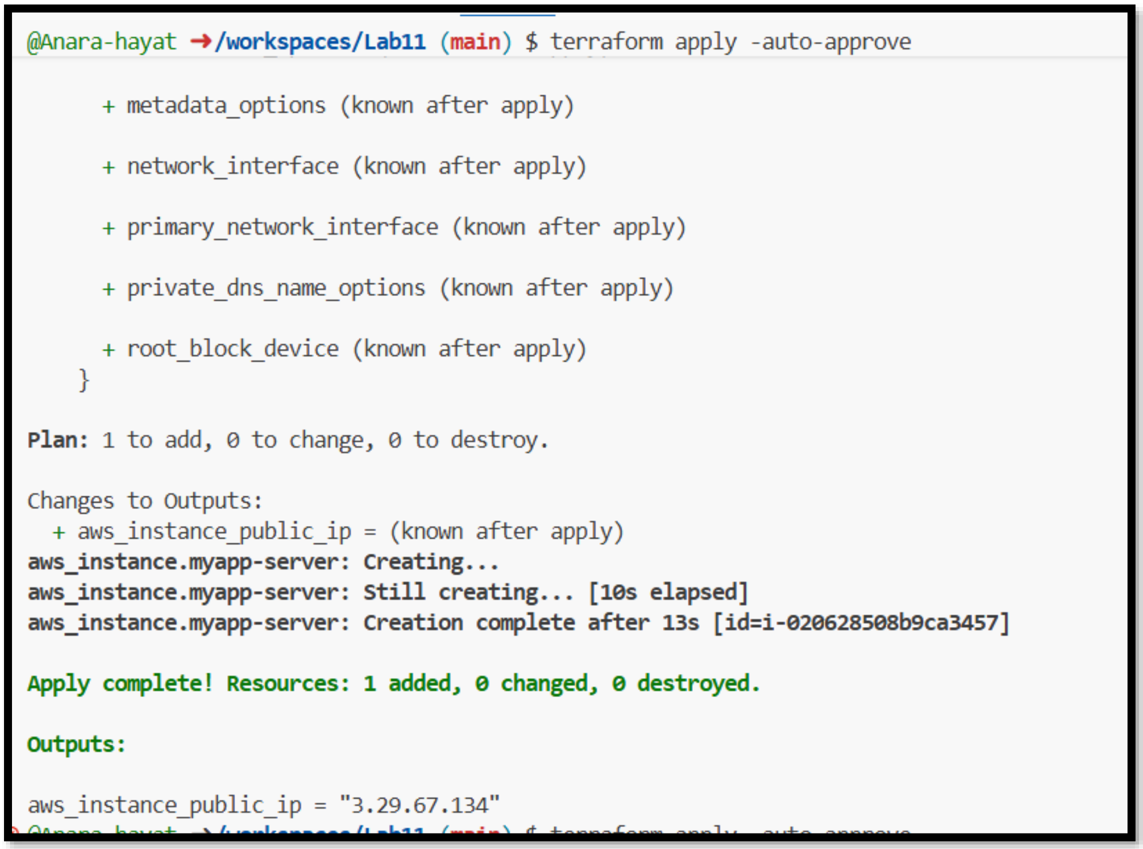


```
.gitignore
.terraform/*
*.tfstate
*.tfstate.*
*.tfvars
*.pem
```

```
@Anara-hayat →/workspaces/Lab11 (main) $ aws ec2 create-key-pair \
   --key-name MyED25519Key \
   --key-type ed25519 \
   --key-format pem \
   --query 'KeyMaterial' \
   --output text > MyED25519Key.pem

  chmod 600 MyED25519Key.pem
@Anara-hayat →/workspaces/Lab11 (main) $
```

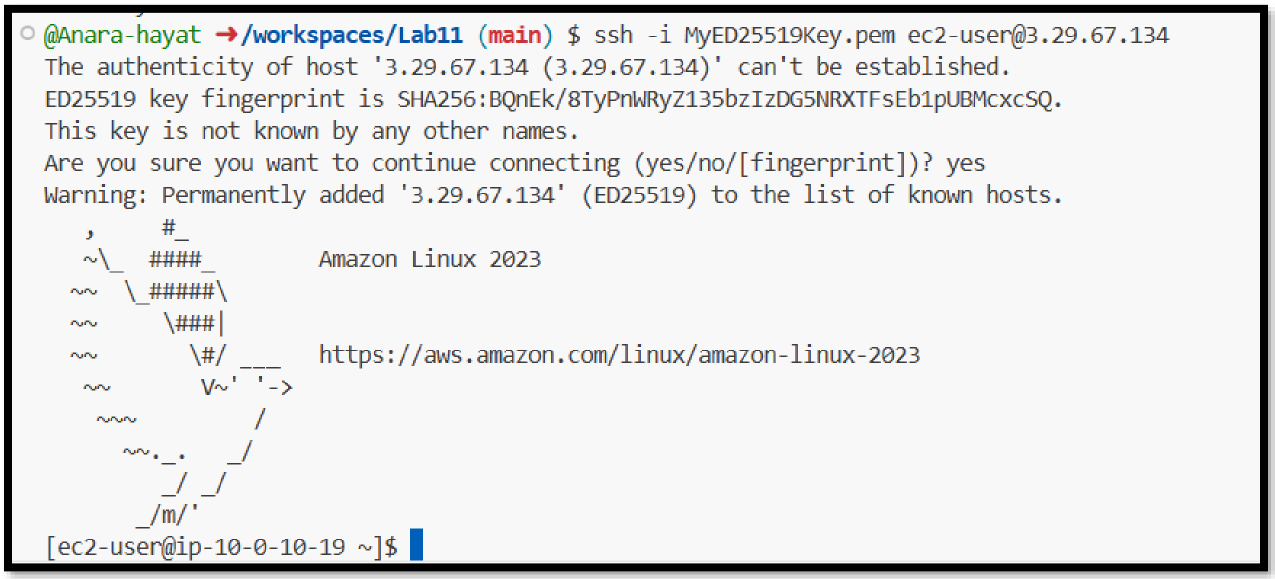
5. **Add EC2 instance resource (initial) Add the instance resource to main.tf (initially using the created key name):**

```
resource "aws_instance" "myapp-server" {
  ami                         = "ami-05524d6658fcf35b6" # Amazon Linux 2023
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet_1.id
  security_groups             = [aws_default_security_group.myapp_sg.id]
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = "MyED25519Key"

  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
}
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```



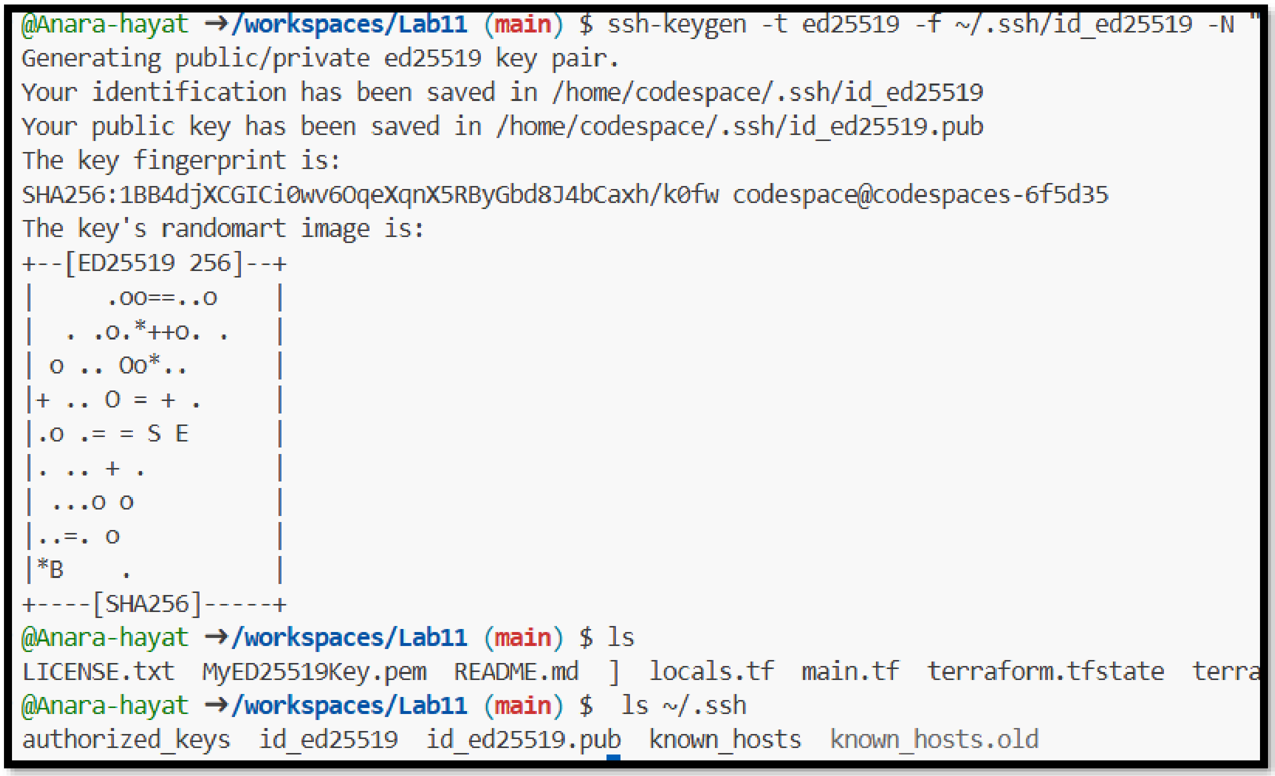
```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + primary_network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + aws_instance_public_ip = (known after apply)
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-020628508b9ca3457]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_public_ip = "3.29.67.134"
```

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



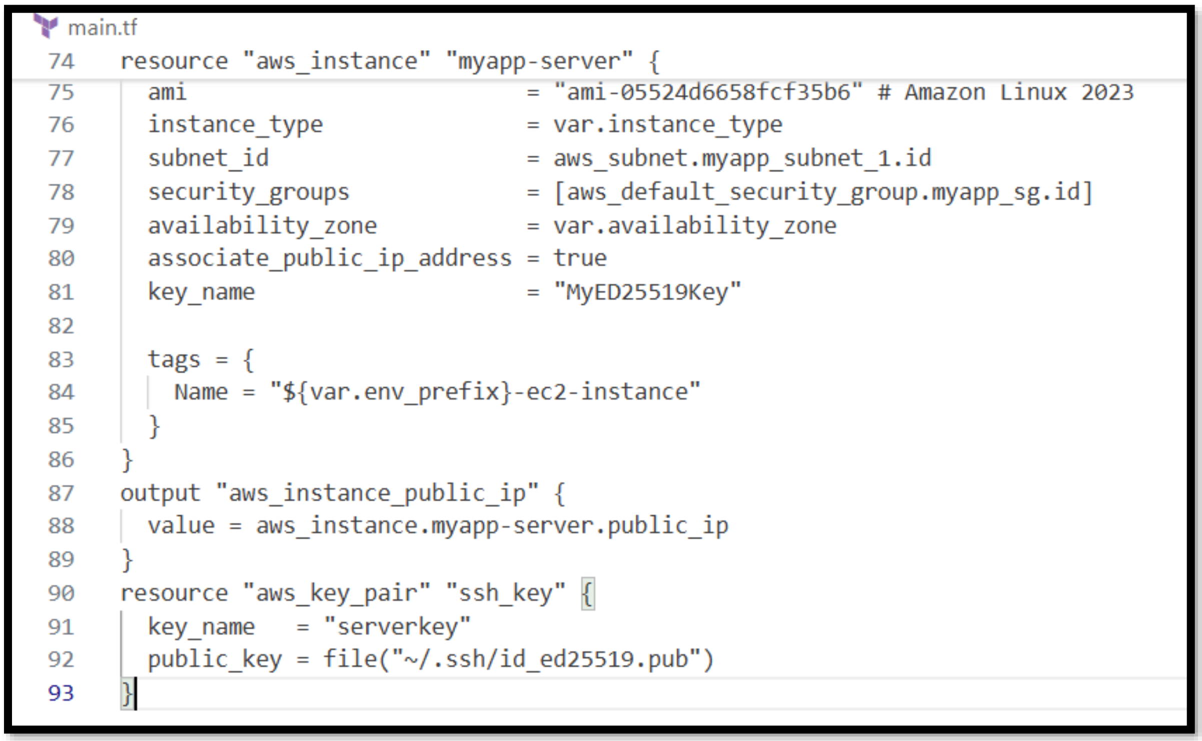
```
@Anara-hayat ➜/workspaces/Lab11 (main) $ ssh -i MyED25519Key.pem ec2-user@3.29.67.134
The authenticity of host '3.29.67.134 (3.29.67.134)' can't be established.
ED25519 key fingerprint is SHA256:BQnEk/8TyPnWRyZ135bzIzDG5NRXTFsEb1pUBMcxcSQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.29.67.134' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-19 ~]$
```

7. **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):**



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N "
Generating public/private ed25519 key pair.
Your identification has been saved in /home/codespace/.ssh/id_ed25519
Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:1BB4djXCGICi0wv6OqeXqnX5RByGbd8J4bCaxh/k0fw codespace@codespaces-6f5d35
The key's randomart image is:
+--[ED25519 256]--+
|     .oo==..o    |
|  . .o.*++o. .   |
| o .. Oo*..      |
|+ .. O = + .     |
|.o .= = S E      |
|. .. + .         |
| ...o o          |
|..=. o           |
|*B    .          |
+----[SHA256]-----+
@Anara-hayat ➜/workspaces/Lab11 (main) $ ls
LICENSE.txt  MyED25519Key.pem  README.md  ]  locals.tf  main.tf  terraform.tfstate  terra
@Anara-hayat ➜/workspaces/Lab11 (main) $  ls ~/.ssh
authorized_keys  id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old
```

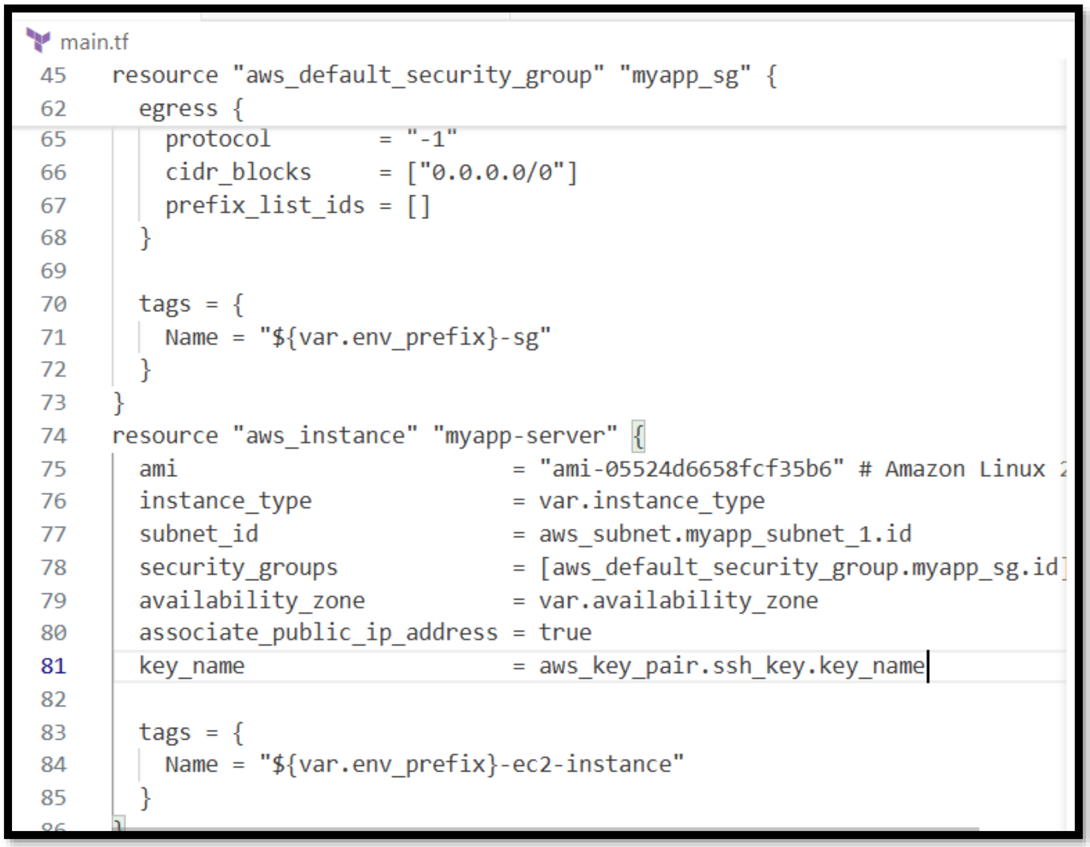
Add a Terraform resource in main.tf to register the public key:



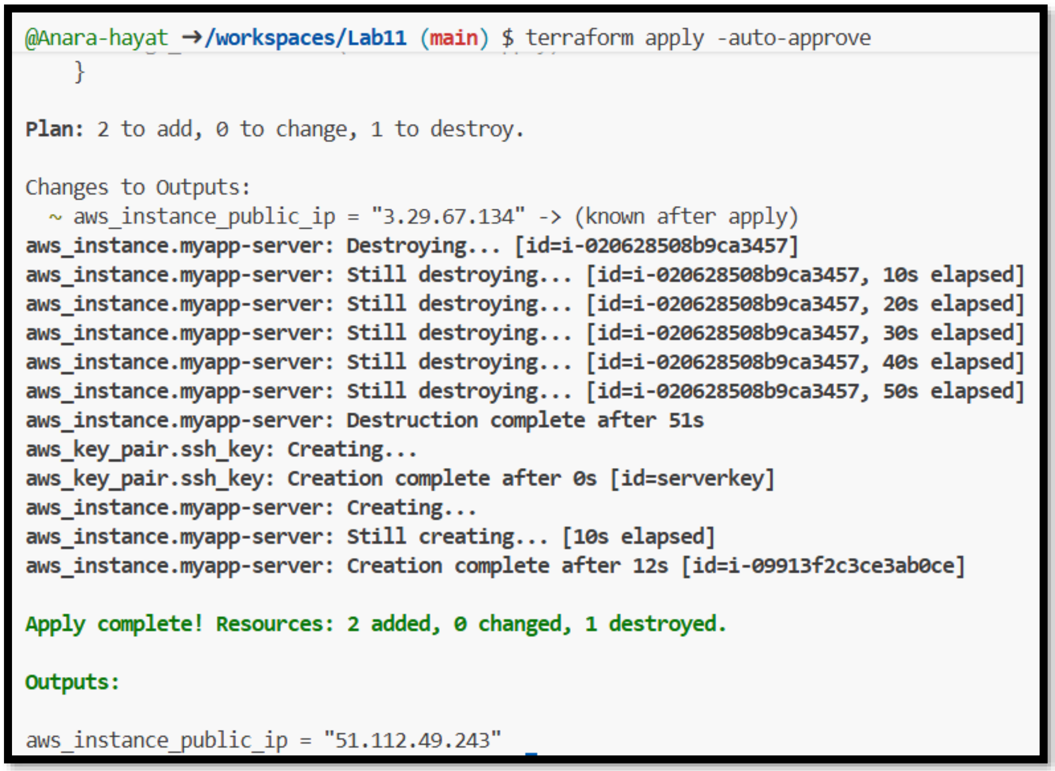
main.tf

```
resource "aws_instance" "myapp-server" {
  ami                         = "ami-05524d6658fcf35b6" # Amazon Linux 2023
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet_1.id
  security_groups             = [aws_default_security_group.myapp_sg.id]
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = "MyED25519Key"

  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
}
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
resource "aws_key_pair" "ssh_key" {
  key_name   = "serverkey"
  public_key = file("~/.ssh/id_ed25519.pub")
}
```

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

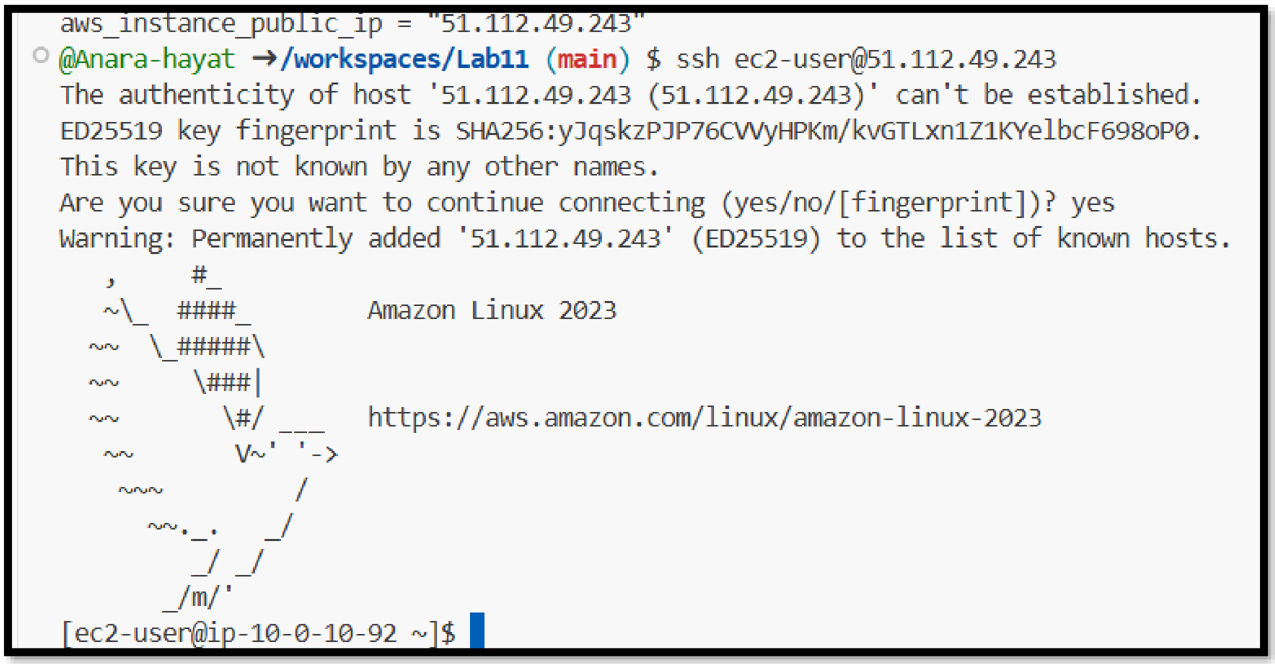

main.tf

```
resource "aws_default_security_group" "myapp_sg" {
  egress {
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
    prefix_list_ids = []
  }

  tags = {
    Name = "${var.env_prefix}-sg"
  }
}
resource "aws_instance" "myapp-server" {
  ami                         = "ami-05524d6658fcf35b6" # Amazon Linux
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet_1.id
  security_groups             = [aws_default_security_group.myapp_sg.id]
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = aws_key_pair.ssh_key.key_name

  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
```



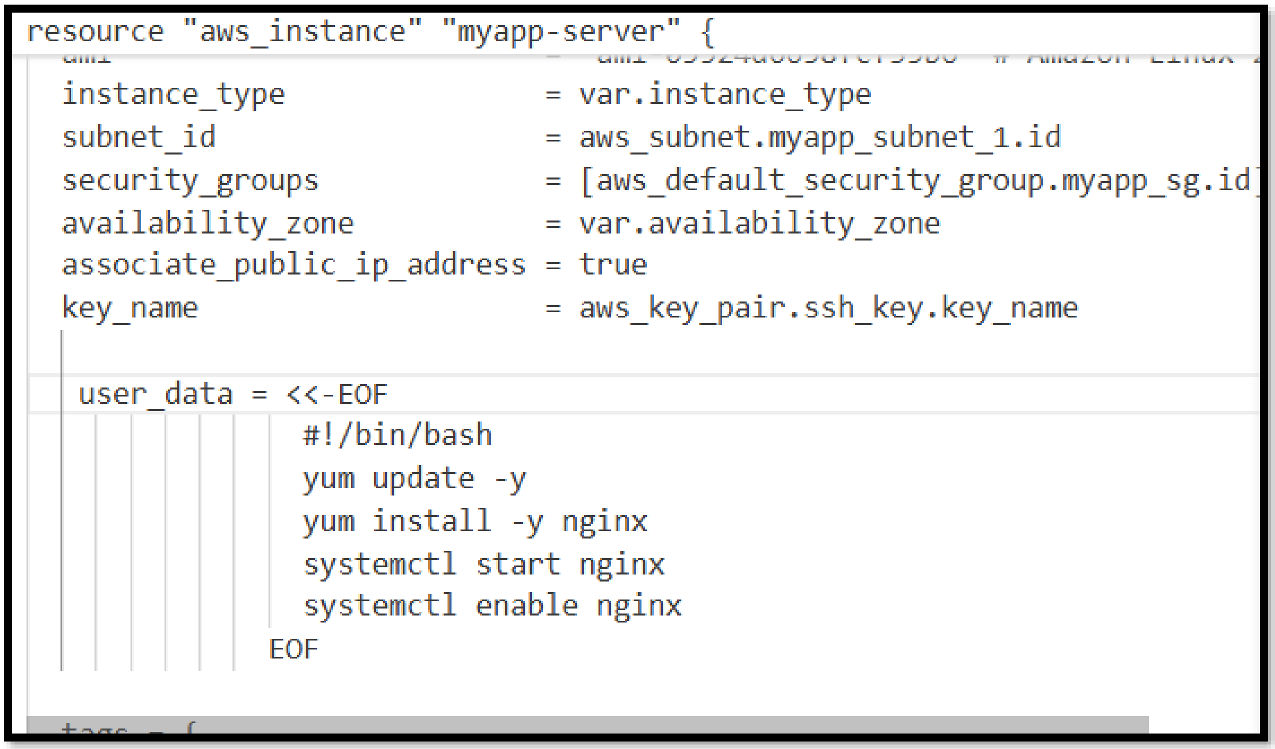
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
    }

Plan: 2 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  ~ aws_instance_public_ip = "3.29.67.134" -> (known after apply)
aws_instance.myapp-server: Destroying... [id=i-020628508b9ca3457]
aws_instance.myapp-server: Still destroying... [id=i-020628508b9ca3457, 10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-020628508b9ca3457, 20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-020628508b9ca3457, 30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-020628508b9ca3457, 40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-020628508b9ca3457, 50s elapsed]
aws_instance.myapp-server: Destruction complete after 51s
aws_key_pair.ssh_key: Creating...
aws_key_pair.ssh_key: Creation complete after 0s [id=serverkey]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 12s [id=i-09913f2c3ce3ab0ce]

Apply complete! Resources: 2 added, 0 changed, 1 destroyed.

Outputs:

aws_instance_public_ip = "51.112.49.243"
```

8. **SSH using the newly registered key Now SSH with your generated private key (the default ssh client will pick up ~/.ssh/id_ed25519):**

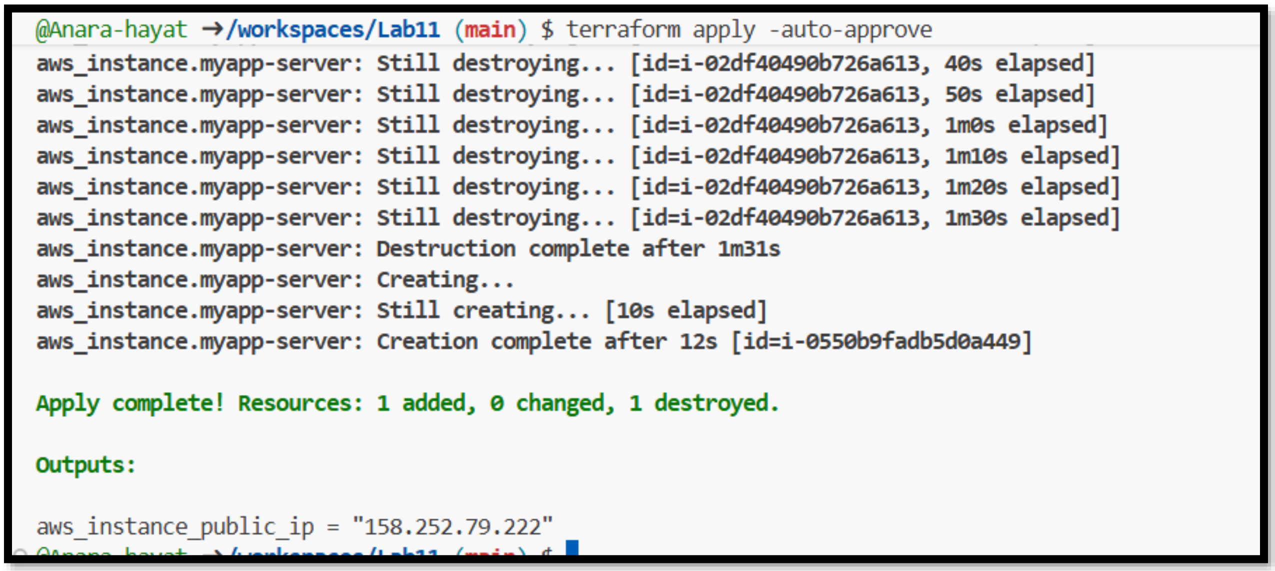


```
aws_instance_public_ip = "51.112.49.243"
@Anara-hayat →/workspaces/Lab11 (main) $ ssh ec2-user@51.112.49.243
The authenticity of host '51.112.49.243 (51.112.49.243)' can't be established.
ED25519 key fingerprint is SHA256:yJqskzPJP76CVVyHPKm/kvGTLxn1Z1KYelbcF698oP0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.49.243' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-92 ~]$
```

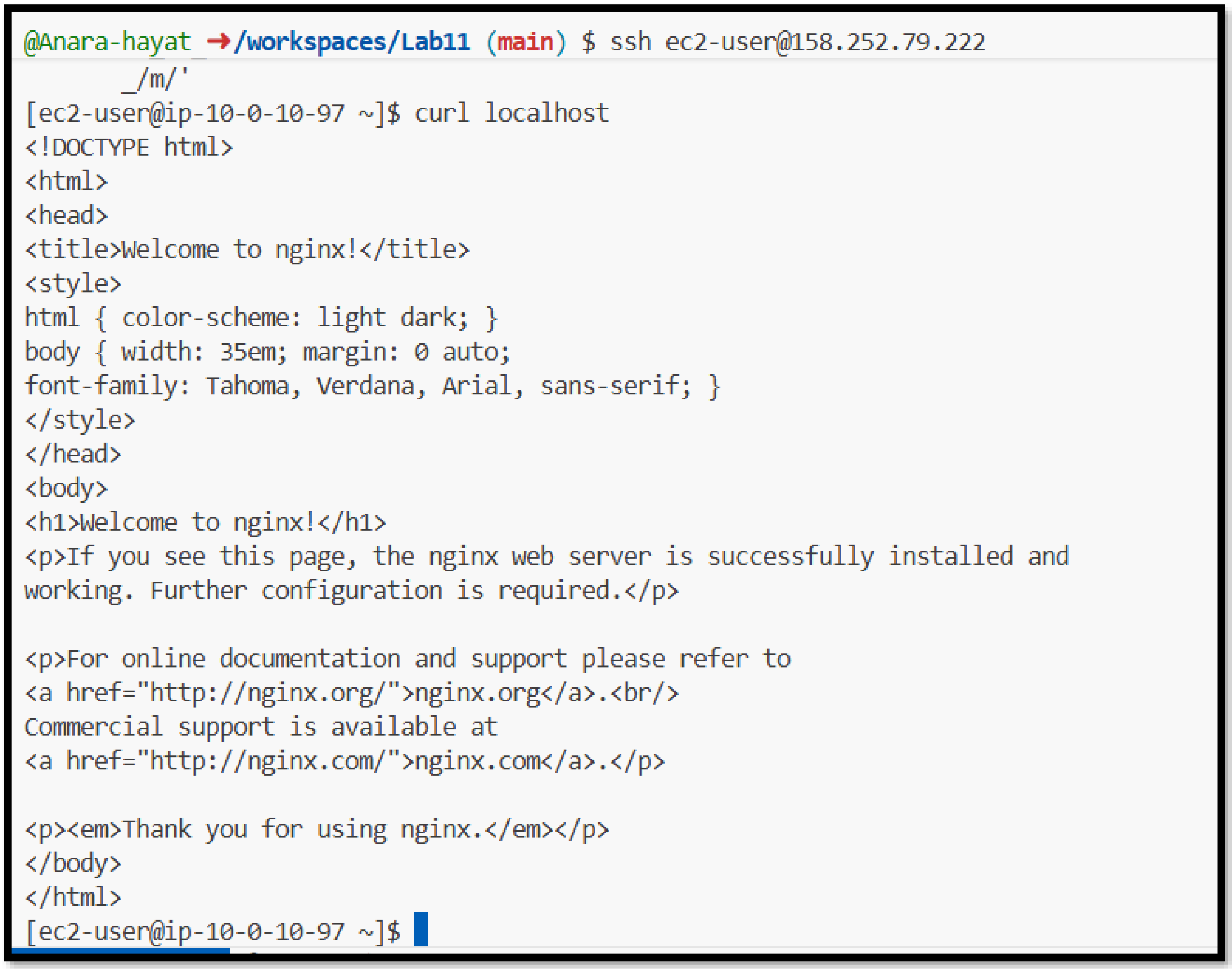
9. **Install nginx via inline user_data Modify the aws_instance resource to include inline user_data:**



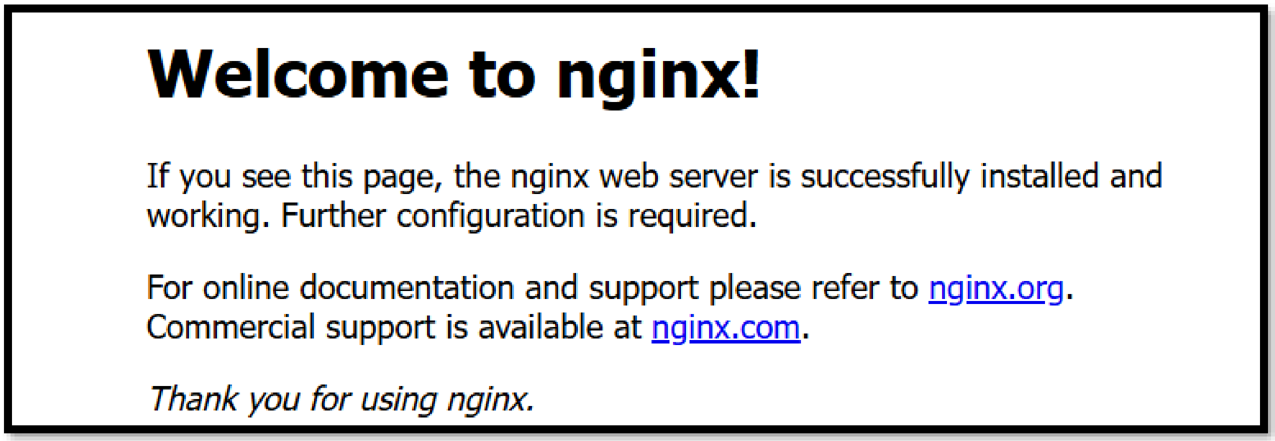
```
resource "aws_instance" "myapp-server" {
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet_1.id
  security_groups             = [aws_default_security_group.myapp_sg.id
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = aws_key_pair.ssh_key.key_name

   user_data = <<-EOF
              #!/bin/bash
              yum update -y
              yum install -y nginx
              systemctl start nginx
              systemctl enable nginx
             EOF
```



```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform apply -auto-approve
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 50s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 1m0s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 1m10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 1m20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-02df40490b726a613, 1m30s elapsed]
aws_instance.myapp-server: Destruction complete after 1m31s
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 12s [id=i-0550b9fadb5d0a449]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

aws_instance_public_ip = "158.252.79.222"
```



```
@Anara-hayat →/workspaces/Lab11 (main) $ ssh ec2-user@158.252.79.222
       _/m/'
[ec2-user@ip-10-0-10-97 ~]$ curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@ip-10-0-10-97 ~]$
```



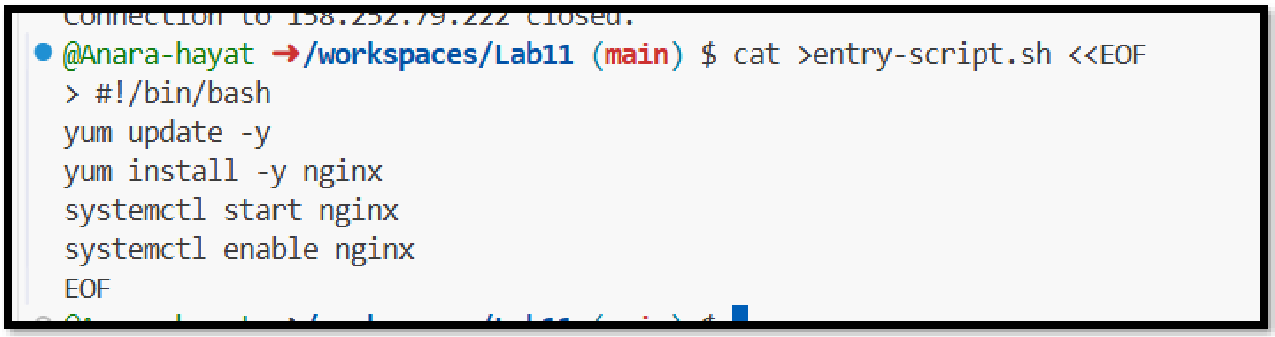
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

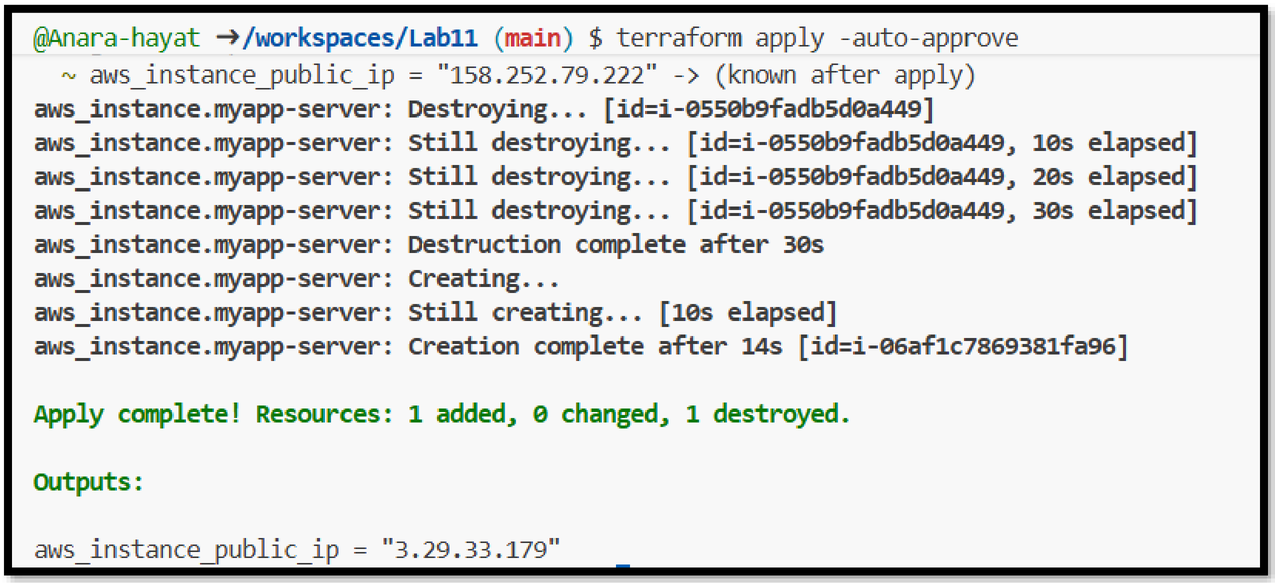
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

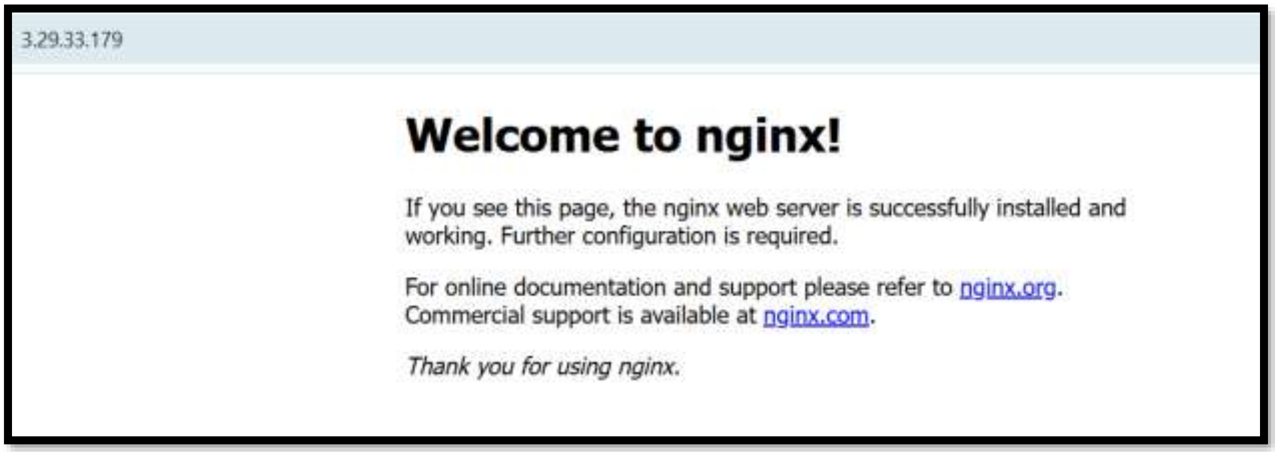
10. Use an external script for user_data Create a script file in the Codespace:



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ cat >entry-script.sh <<EOF
> #!/bin/bash
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
EOF
```



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ terraform apply -auto-approve
  ~ aws_instance_public_ip = "158.252.79.222" -> (known after apply)
aws_instance.myapp-server: Destroying... [id=i-0550b9fadb5d0a449]
aws_instance.myapp-server: Still destroying... [id=i-0550b9fadb5d0a449, 10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0550b9fadb5d0a449, 20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0550b9fadb5d0a449, 30s elapsed]
aws_instance.myapp-server: Destruction complete after 30s
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 14s [id=i-06af1c7869381fa96]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

aws_instance_public_ip = "3.29.33.179"
```


3.29.33.179

Welcome to nginx!

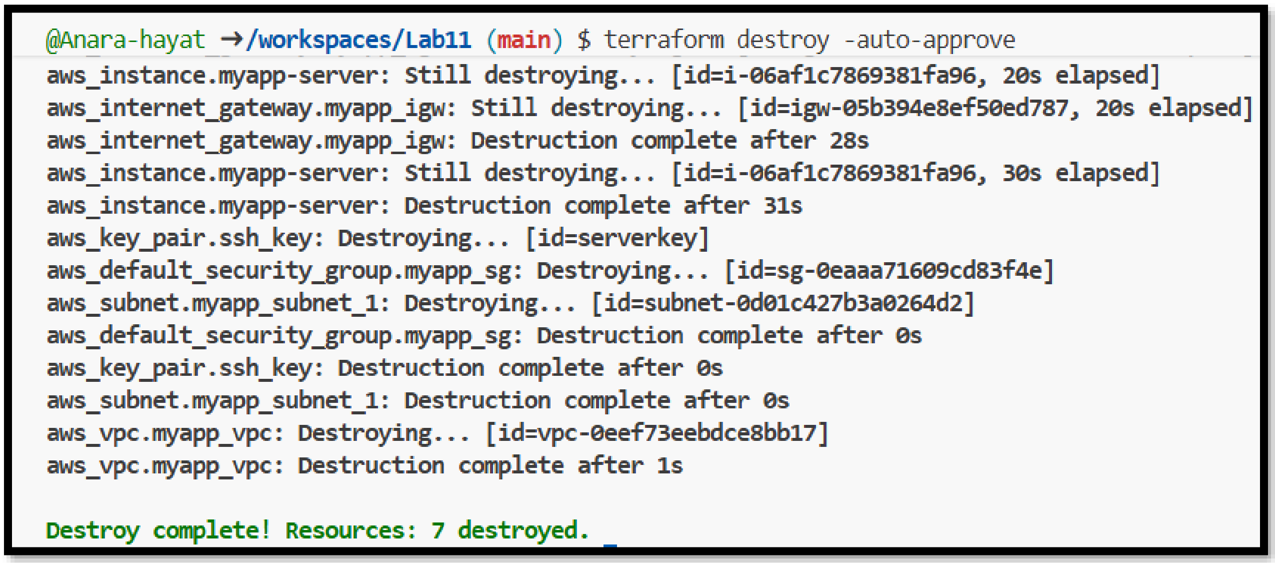
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

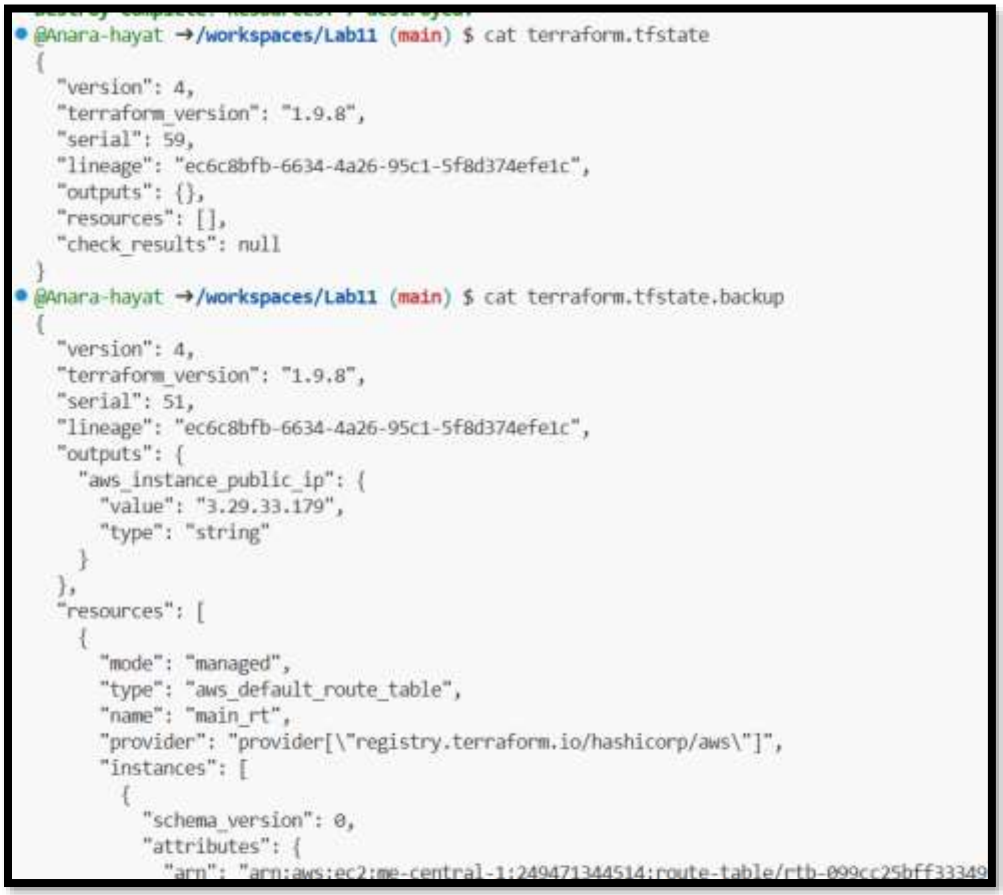
Cleanup

1. **Destroy all resources:**



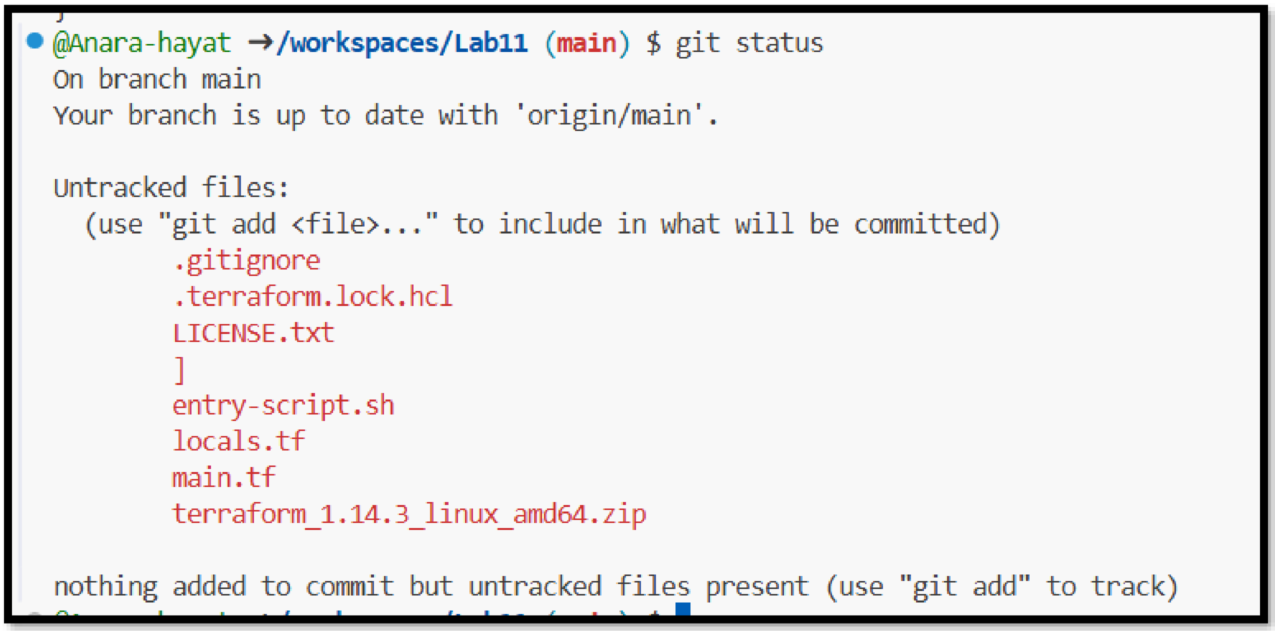
```
@Anara-hayat →/workspaces/Lab11 (main) $ terraform destroy -auto-approve
aws_instance.myapp-server: Still destroying... [id=i-06af1c7869381fa96, 20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-05b394e8ef50ed787, 20s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 28s
aws_instance.myapp-server: Still destroying... [id=i-06af1c7869381fa96, 30s elapsed]
aws_instance.myapp-server: Destruction complete after 31s
aws_key_pair.ssh_key: Destroying... [id=serverkey]
aws_default_security_group.myapp_sg: Destroying... [id=sg-0eaaa71609cd83f4e]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0d01c427b3a0264d2]
aws_default_security_group.myapp_sg: Destruction complete after 0s
aws_key_pair.ssh_key: Destruction complete after 0s
aws_subnet.myapp_subnet_1: Destruction complete after 0s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0eef73eebdce8bb17]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```

2. **Verify state files:**



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.9.8",
  "serial": 59,
  "lineage": "ec6c8bfb-6634-4a26-95c1-5f8d374efe1c",
  "outputs": {},
  "resources": [],
  "check_results": null
}
@Anara-hayat ➜/workspaces/Lab11 (main) $ cat terraform.tfstate.backup
{
  "version": 4,
  "terraform_version": "1.9.8",
  "serial": 51,
  "lineage": "ec6c8bfb-6634-4a26-95c1-5f8d374efe1c",
  "outputs": {
    "aws_instance_public_ip": {
      "value": "3.29.33.179",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "aws_default_route_table",
      "name": "main_rt",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "arn": "arn:aws:ec2:me-central-1:249471344514:route-table/rtb-099cc25bff33349
```

3. **Ensure no sensitive files are committed. If any private keys or credentials were accidentally committed — rotate them immediately.**



```
@Anara-hayat ➜/workspaces/Lab11 (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .terraform.lock.hcl
        LICENSE.txt
        ]
        entry-script.sh
        locals.tf
        main.tf
        terraform_1.14.3_linux_amd64.zip

nothing added to commit but untracked files present (use "git add" to track)
```