Name: Lujain Zia
Roll no: 2023-BSE-034
Date: December 4, 2025

Fatima Jinnah Women University

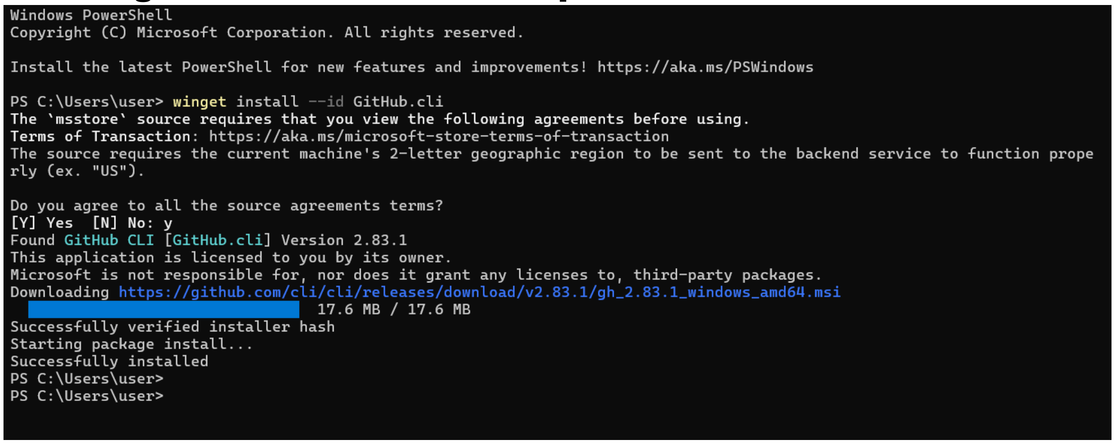
Lab 10

Task 1 — GitHub CLI, Codespace setup and authentication

1. Install GitHub CLI:

winget install --id GitHub.cli

Save screenshot as: task1_gh_install.png — terminal showing the winget install command output.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\user> winget install --id GitHub.cli
The `msstore` source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to the backend service to function prope
rly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes  [N] No: y
Found GitHub CLI [GitHub.cli] Version 2.83.1
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/cli/cli/releases/download/v2.83.1/gh_2.83.1_windows_amd64.msi
                              17.6 MB / 17.6 MB
Successfully verified installer hash
Starting package install...
Successfully installed
PS C:\Users\user>
PS C:\Users\user>
```

2. (Local) Authenticate GH CLI for Codespaces:

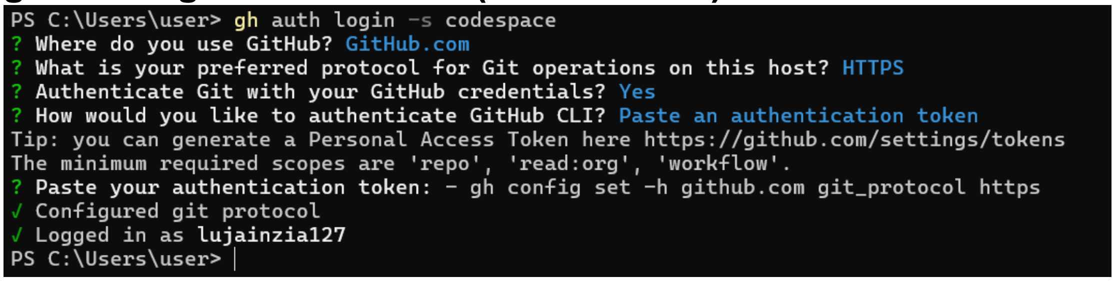
gh auth login -s codespace

When prompted, generate a GitHub Access Token (classic) in the browser with the following Token scopes:

admin:org
codespace
repo

Copy the token into the GH CLI prompt to complete login.

Save screenshot as: task1_gh_auth_login.png — screenshot of gh auth login confirmation (redact token).



```
PS C:\Users\user> gh auth login -s codespace
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: - gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as lujainzia127
PS C:\Users\user> |
```

3. List available Codespaces (optional verification):

gh codespace list

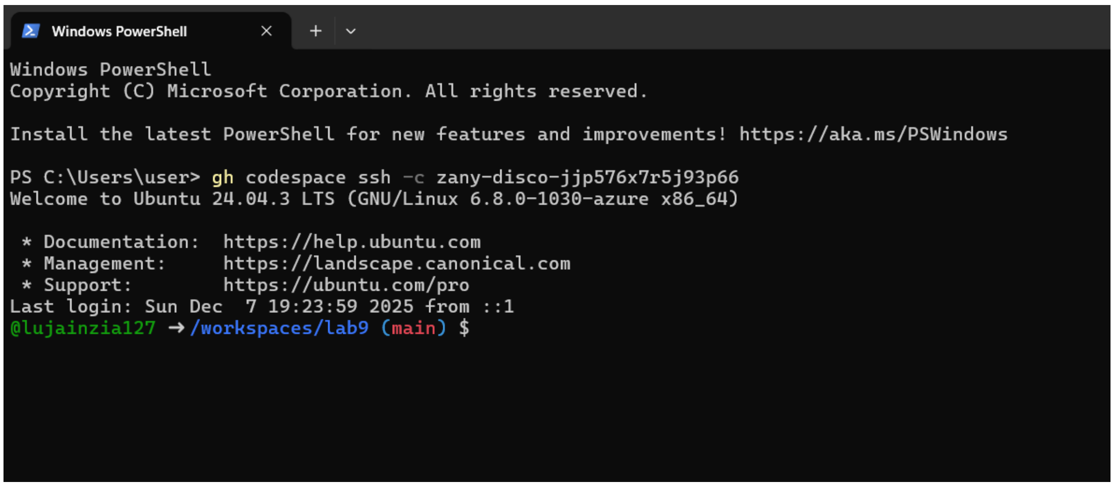
Save screenshot as: task1_codespace_list.png — gh codespace list output (show codespace name).

4. Connect to a codespace via SSH:

gh codespace ssh -c <name_of_codespace>

After connecting via gh codespace ssh, you will be inside a Linux shell that the rest of this lab assumes.

Save screenshot as: task1_codespace_ssh_connected.png — terminal inside the Codespace shell after gh codespace ssh -c <name>.



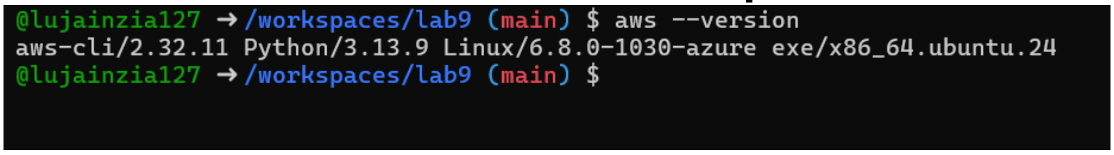
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\user> gh codespace ssh -c zany-disco-jjp576x7r5j93p66
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
Last login: Sun Dec  7 19:23:59 2025 from ::1
@lujainzia127 ➜ /workspaces/lab9 (main) $
```

Task 2 — Install AWS CLI, Terraform CLI, Provider Setup

1. In Codespace shell, install AWS CLI:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws --version

Save screenshot as: task2_aws_install_and_version.png — show AWS CLI installation and version output.



```
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws --version
aws-cli/2.32.11 Python/3.13.9 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
@lujainzia127 ➜ /workspaces/lab9 (main) $
```

2. Configure AWS CLI:

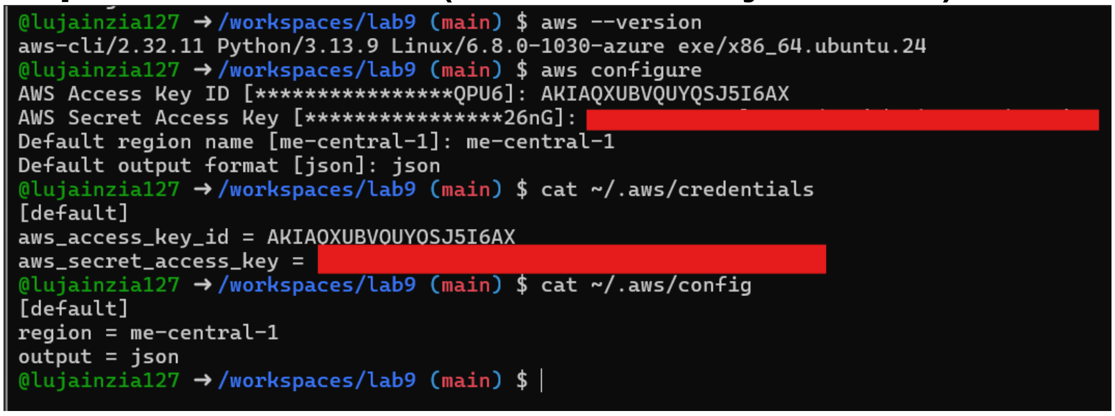
aws configure

Save screenshot as: task2_aws_configure_and_files.png — show aws configure prompt (redact secret values).

Then check config files:

cat ~/.aws/credentials
cat ~/.aws/config

Save screenshot as: task2_aws_configure_and_files.png — output of cat commands (redact secret keys if visible).

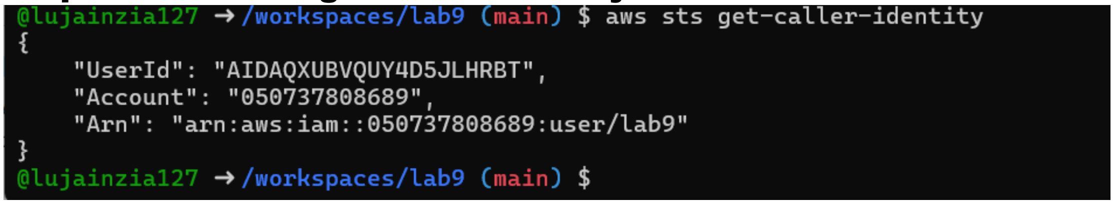


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws --version
aws-cli/2.32.11 Python/3.13.9 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws configure
AWS Access Key ID [****************QPU6]: AKIAQXUBVQUYQSJ5I6AX
AWS Secret Access Key [****************26nG]: [redacted]
Default region name [me-central-1]: me-central-1
Default output format [json]: json
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat ~/.aws/credentials
[default]
aws_access_key_id = AKIAQXUBVQUYQSJ5I6AX
aws_secret_access_key = [redacted]
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat ~/.aws/config
[default]
region = me-central-1
output = json
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

3. Verify AWS CLI connectivity:

aws sts get-caller-identity

Save screenshot as: task2_aws_get_caller_identity.png — output of aws sts get-caller-identity.



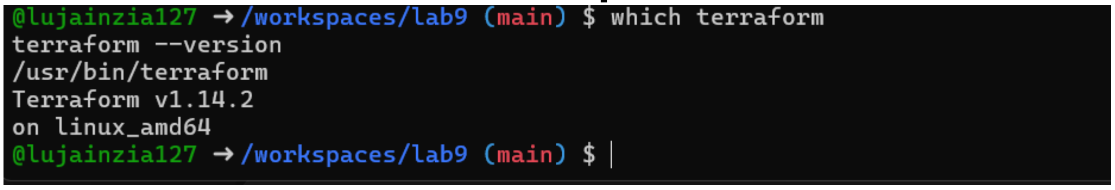
```
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws sts get-caller-identity
{
    "UserId": "AIDAQXUBVQUY4D5JLHRBT",
    "Account": "050737808689",
    "Arn": "arn:aws:iam::050737808689:user/lab9"
}
@lujainzia127 ➜ /workspaces/lab9 (main) $
```

B. Install Terraform CLI

1. Install Terraform CLI:

wget -O - https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpgecho "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(grep -oP '(?<=UBUNTU_CODENAME=).*' /etc/os-release || lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt install terraform
which terraform
terraform --version

Save screenshot as: task2_terraform_install_and_version.png — terraform installation and version check output.



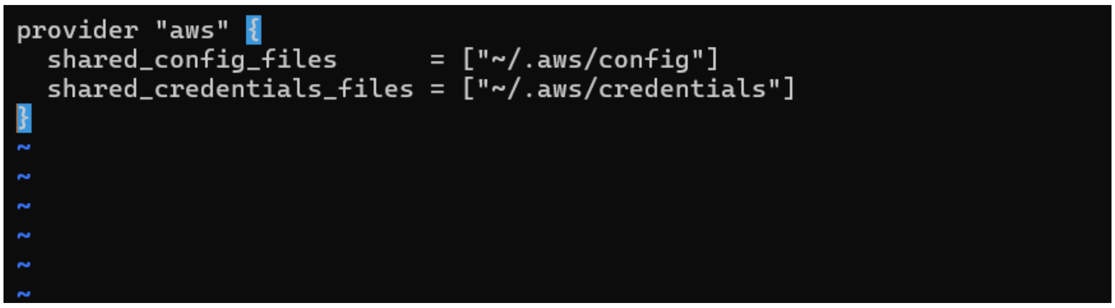
```
@lujainzia127 ➜ /workspaces/lab9 (main) $ which terraform
terraform --version
/usr/bin/terraform
Terraform v1.14.2
on linux_amd64
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

C. Provider Configuration (main.tf)

1. Create main.tf using Vim:

vim main.tf

Save screenshot as: task2_provider_file_creation.png — creation/editing of main.tf.

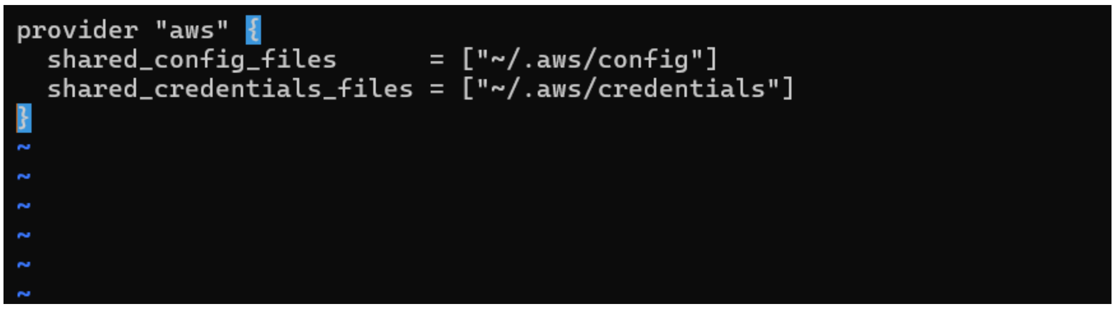


```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
~
~
~
~
~
~
```

2. Inside main.tf, write:

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

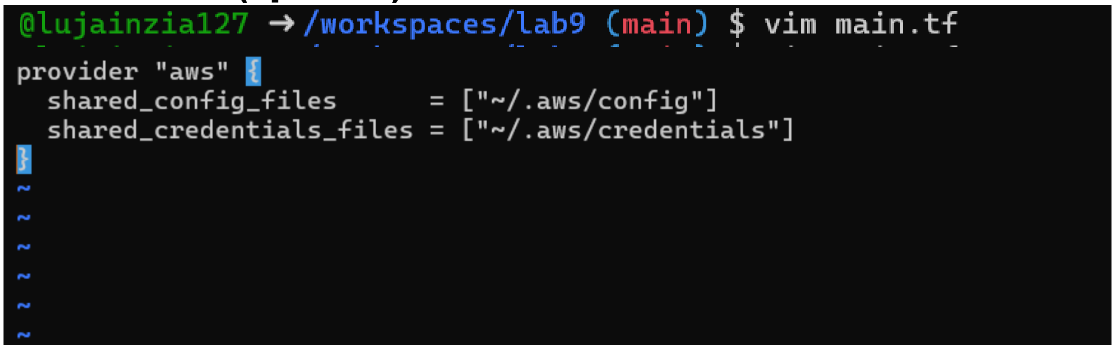
Save screenshot as: task2_provider_block.png — final content saved in main.tf.



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

3. Save and quit: Press ESC then :wq

Save screenshot as: task2_vim_save_main_tf.png — Vim save confirmation (optional).

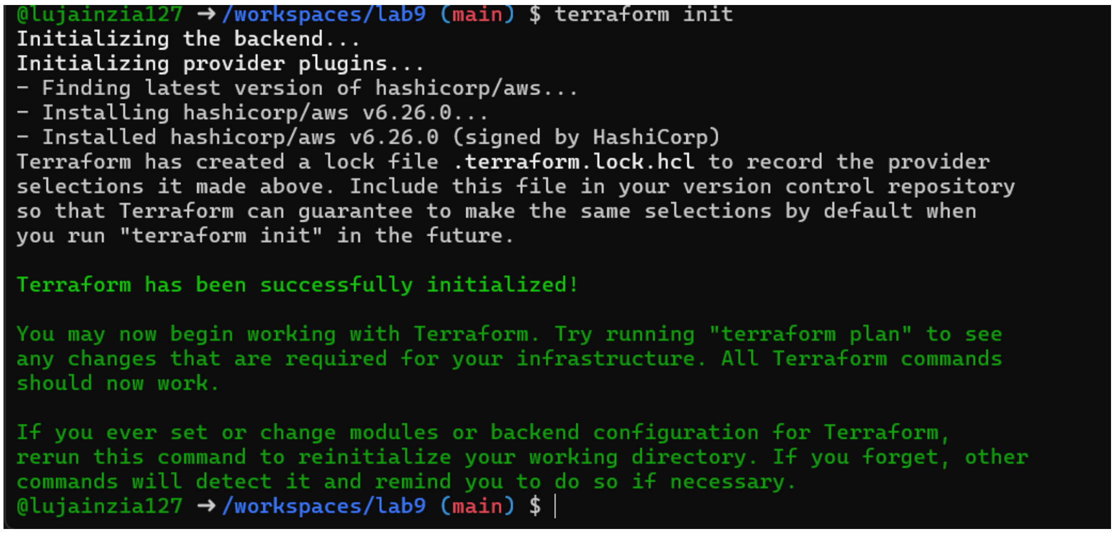


```
@lujainzia127 → /workspaces/lab9 (main) $ vim main.tf
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
~
~
~
~
~
~
```

4. Initialize Terraform:

terraform init

Save screenshot as: task2_terraform_init_output.png — terraform init output.

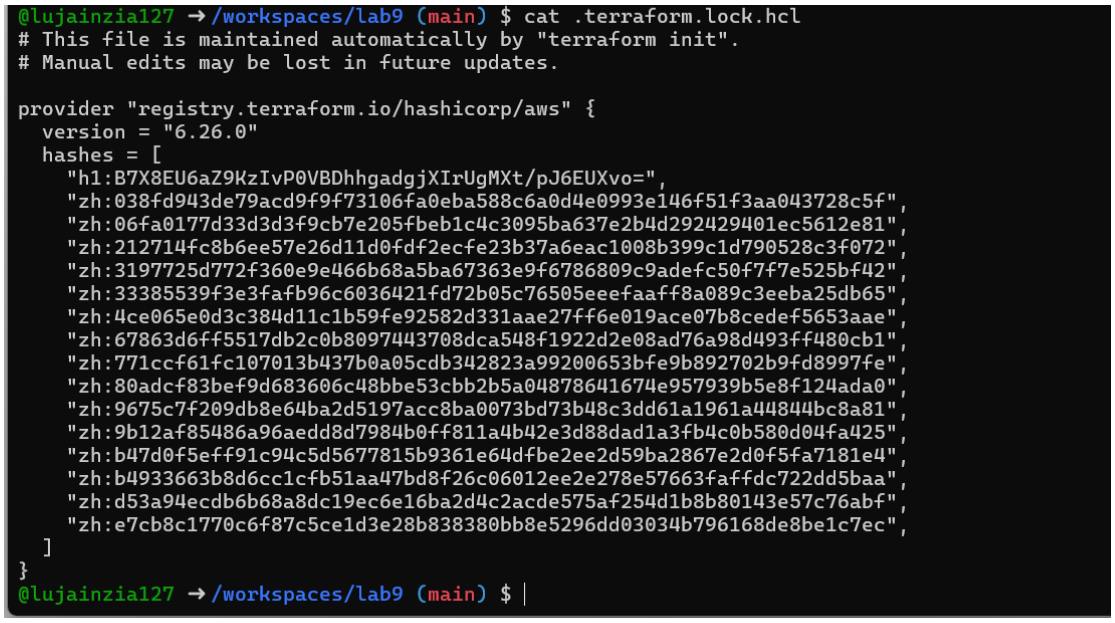


```
@lujainzia127 → /workspaces/lab9 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.26.0...
- Installed hashicorp/aws v6.26.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
@lujainzia127 → /workspaces/lab9 (main) $ |
```

5. Show contents of .terraform.lock.hcl:

cat .terraform.lock.hcl

Save screenshot as: task2_terraform_lock_hcl.png — output of .terraform.lock.hcl.

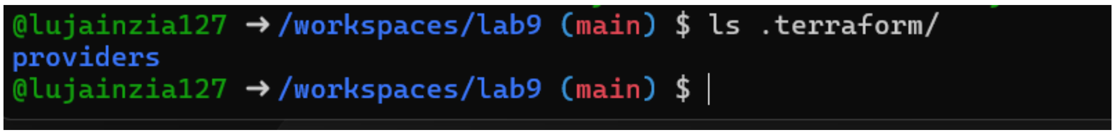


```
@lujainzia127 → /workspaces/lab9 (main) $ cat .terraform.lock.hcl
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/aws" {
  version = "6.26.0"
  hashes = [
    "h1:B7X8EU6aZ9KzIvP0VBDhhgadgjXIrUgMXt/pJ6EUXvo=",
    "zh:038fd943de79acd9f9f73106fa0eba588c6a0d4e0993e146f51f3aa043728c5f",
    "zh:06fa0177d33d3d3f9cb7e205fbeb1c4c3095ba637e2b4d292429401ec5612e81",
    "zh:212714fc8b6ee57e26d11d0fdf2ecfe23b37a6eac1008b399c1d790528c3f072",
    "zh:3197725d772f360e9e466b68a5ba67363e9f6786809c9adefc50f7f7e525bf42",
    "zh:33385539f3e3fafb96c6036421fd72b05c76505eeefaaff8a089c3eeba25db65",
    "zh:4ce065e0d3c384d11c1b59fe92582d331aae27ff6e019ace07b8cedef5653aae",
    "zh:67863d6ff5517db2c0b8097443708dca548f1922d2e08ad76a98d493ff480cb1",
    "zh:771ccf61fc107013b437b0a05cdb342823a99200653bfe9b892702b9fd8997fe",
    "zh:80adcf83bef9d683606c48bbe53cbb2b5a04878641674e957939b5e8f124ada0",
    "zh:9675c7f209db8e64ba2d5197acc8ba0073bd73b48c3dd61a1961a44844bc8a81",
    "zh:9b12af85486a96aedd8d7984b0ff811a4b42e3d88dad1a3fb4c0b580d04fa425",
    "zh:b47d0f5eff91c94c5d5677815b9361e64dfbe2ee2d59ba2867e2d0f5fa7181e4",
    "zh:b4933663b8d6cc1cfb51aa47bd8f26c06012ee2e278e57663faffdc722dd5baa",
    "zh:d53a94ecdb6b68a8dc19ec6e16ba2d4c2acde575af254d1b8b80143e57c76abf",
    "zh:e7cb8c1770c6f87c5ce1d3e28b838380bb8e5296dd03034b796168de8be1c7ec",
  ]
}
@lujainzia127 → /workspaces/lab9 (main) $ |
```

6. Show contents of .terraform/ directory:

ls .terraform/

Save screenshot as: task2_terraform_dir_ls.png — output of ls .terraform/.



```
@lujainzia127 → /workspaces/lab9 (main) $ ls .terraform/
providers
@lujainzia127 → /workspaces/lab9 (main) $ |
```

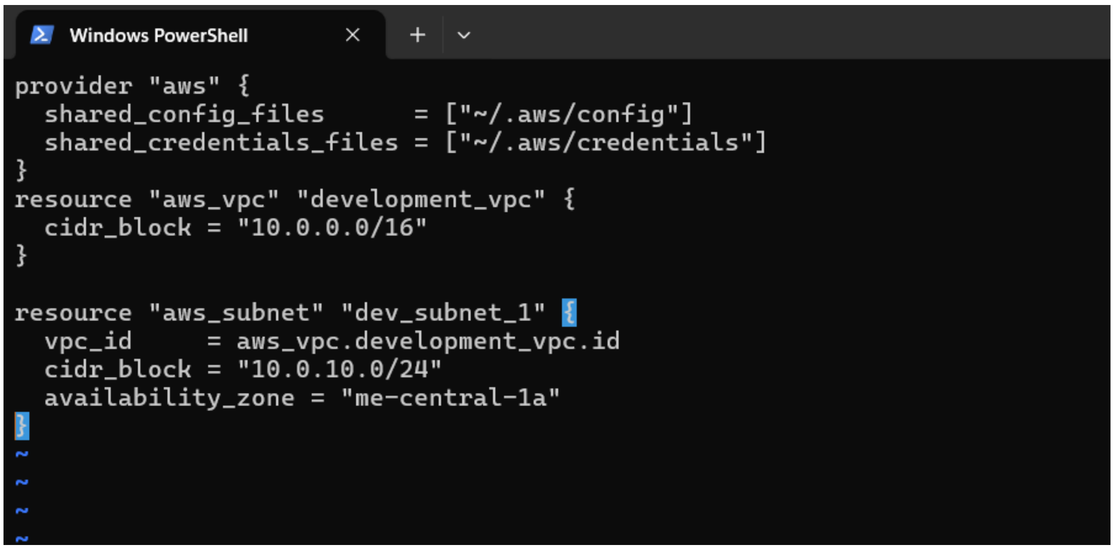
Task 3 — VPC/Subnet Creation, Initialization, Verification

1. Edit main.tf to add:

```
resource "aws_vpc" "development_vpc" {
  cidr_block = "10.0.0.0/16"
}
resource "aws_subnet" "dev_subnet_1" {
  vpc_id      = aws_vpc.development_vpc.id
  cidr_block = "10.0.10.0/24"
  availability_zone = "me-central-1a"
}
```

Use vim main.tf, edit and insert code.

Save screenshot as: task3_main_tf_resource_add.png — new resources added and visible in main.tf.



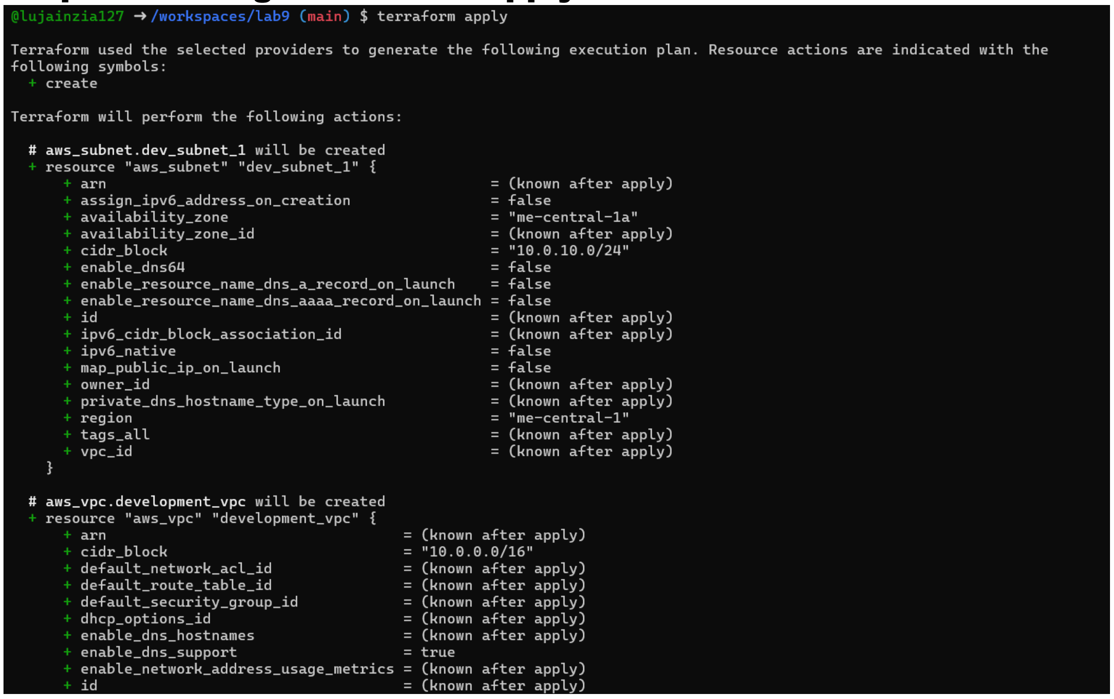
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
resource "aws_vpc" "development_vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "dev_subnet_1" {
  vpc_id      = aws_vpc.development_vpc.id
  cidr_block = "10.0.10.0/24"
  availability_zone = "me-central-1a"
}
~
~
~
~
```

2. Run:

terraform apply

Type yes when prompted.

Save screenshot as: task3_terraform_apply_vpc_subnet.png — output showing successful apply and resources creation.



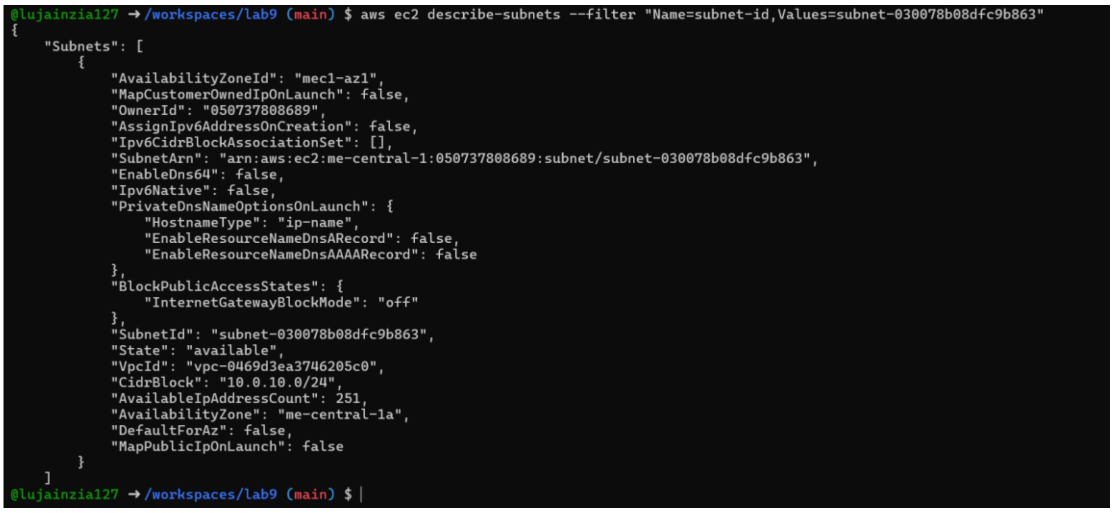
```
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_subnet.dev_subnet_1 will be created
  + resource "aws_subnet" "dev_subnet_1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "me-central-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.10.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + region                                         = "me-central-1"
      + tags_all                                       = (known after apply)
      + vpc_id                                         = (known after apply)
    }

  # aws_vpc.development_vpc will be created
  + resource "aws_vpc" "development_vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
```

3. Verify resources using AWS CLI:

Run:

aws ec2 describe-subnets --filter "Name=subnet-id,Values=<subnet-id>"

Save screenshot as: task3_aws_cli_verify_subnet.png — output of describe-subnets command.

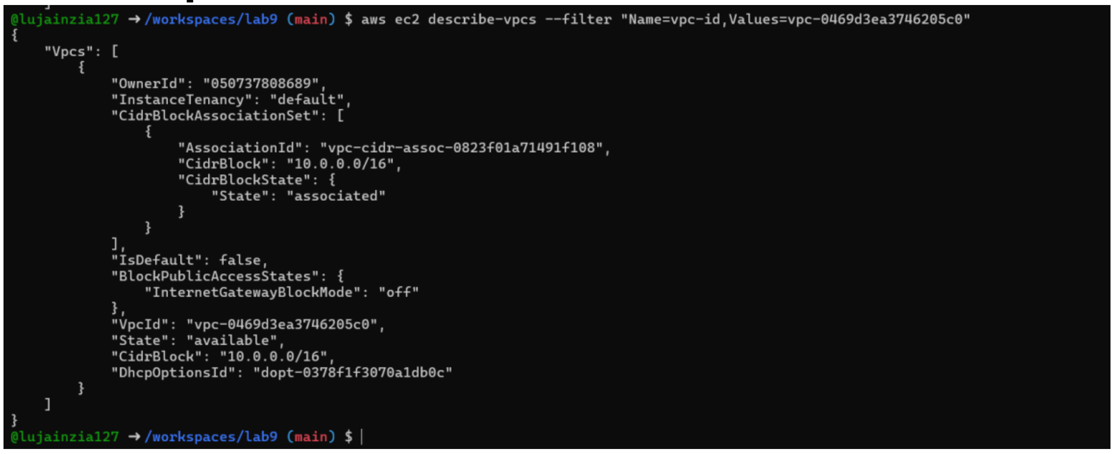


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws ec2 describe-subnets --filter "Name=subnet-id,Values=subnet-030078b08dfc9b863"
{
    "Subnets": [
        {
            "AvailabilityZoneId": "mec1-az1",
            "MapCustomerOwnedIpOnLaunch": false,
            "OwnerId": "050737808689",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:me-central-1:050737808689:subnet/subnet-030078b08dfc9b863",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "SubnetId": "subnet-030078b08dfc9b863",
            "State": "available",
            "VpcId": "vpc-0469d3ea3746205c0",
            "CidrBlock": "10.0.10.0/24",
            "AvailableIpAddressCount": 251,
            "AvailabilityZone": "me-central-1a",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false
        }
    ]
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

Run:

aws ec2 describe-vpcs --filter "Name=vpc-id,Values=<vpc-id>"

Save screenshot as: task3_aws_cli_verify_vpc.png — output of describe-vpcs command.



```
@lujainzia127 ➜ /workspaces/lab9 (main) $ aws ec2 describe-vpcs --filter "Name=vpc-id,Values=vpc-0469d3ea3746205c0"
{
    "Vpcs": [
        {
            "OwnerId": "050737808689",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0823f01a71491f108",
                    "CidrBlock": "10.0.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "VpcId": "vpc-0469d3ea3746205c0",
            "State": "available",
            "CidrBlock": "10.0.0.0/16",
            "DhcpOptionsId": "dopt-0378f1f3070a1db0c"
        }
    ]
}
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

Task 4 — Data Source, Targeted Destroy, Tags

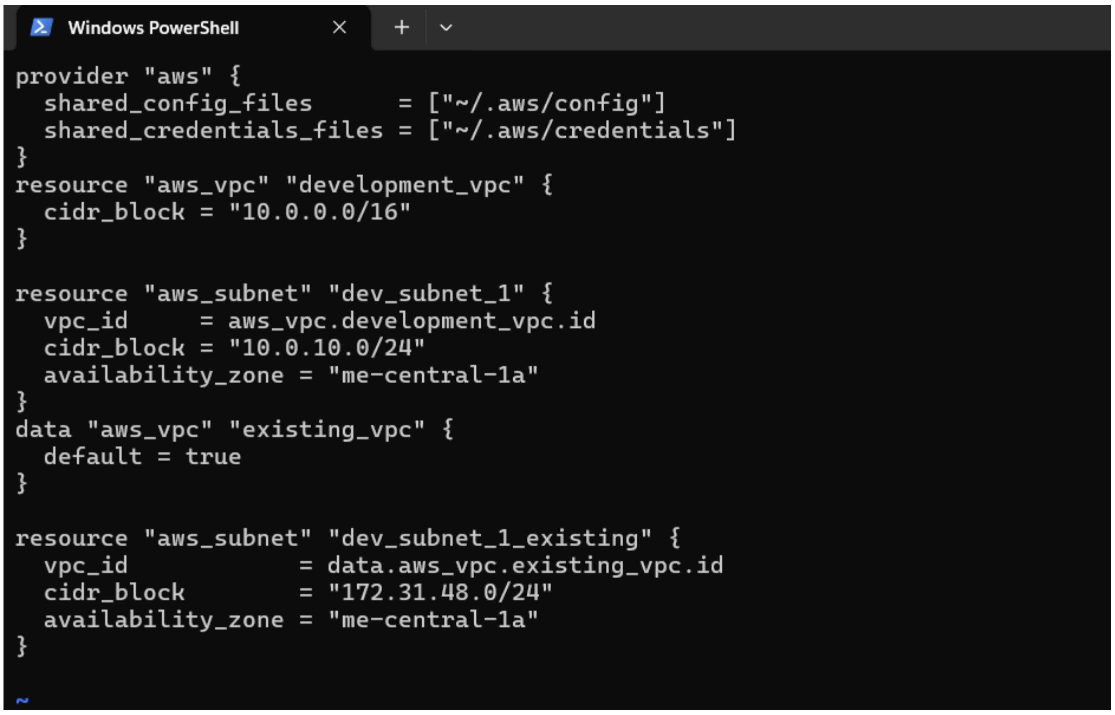
A. Data Source & Resource Creation

1. Add to main.tf:

```
data "aws_vpc" "existing_vpc" {
  default = true
}
resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id      = data.aws_vpc.existing_vpc.id
  cidr_block = "172.31.48.0/24"
  availability_zone = "me-central-1a"
}
```

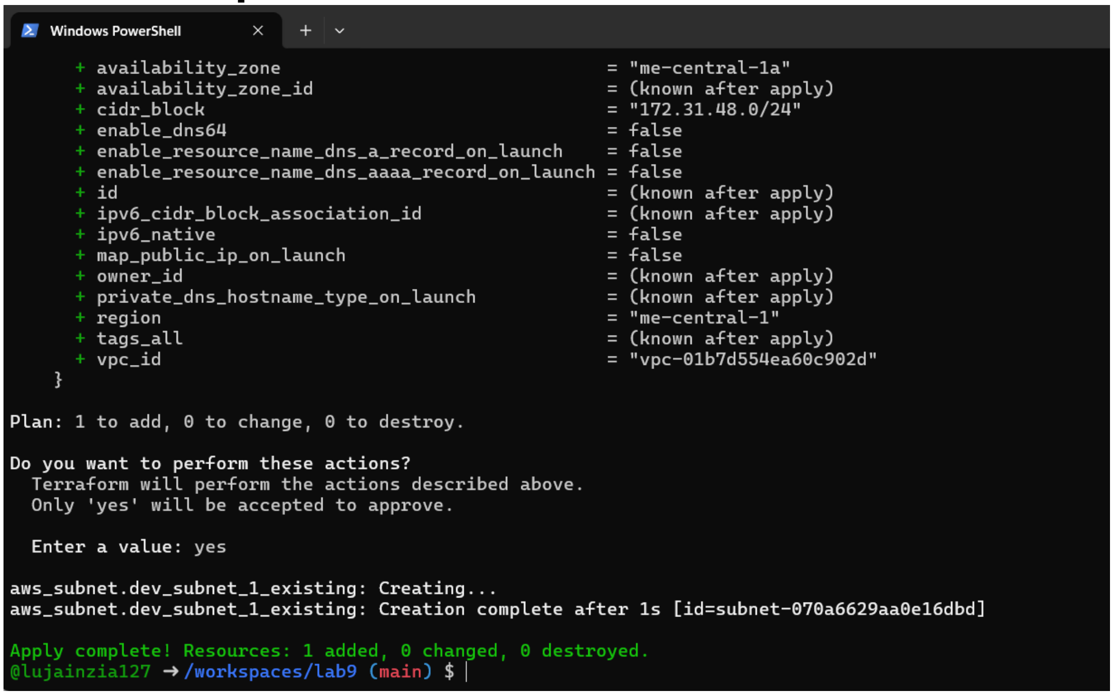
Use vim to edit and insert code.

Save screenshot
as: task4_main_tf_datasource_resource_add.png — main.tf with datasource and resource added.



```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
resource "aws_vpc" "development_vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "dev_subnet_1" {
  vpc_id     = aws_vpc.development_vpc.id
  cidr_block = "10.0.10.0/24"
  availability_zone = "me-central-1a"
}
data "aws_vpc" "existing_vpc" {
  default = true
}

resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id            = data.aws_vpc.existing_vpc.id
  cidr_block        = "172.31.48.0/24"
  availability_zone = "me-central-1a"
}

~
```

2. Apply configuration:
terraform apply
Confirm only the new subnet is created.
Save screenshot
as: task4_terraform_apply_datasource_resource.png — terraform output of resource creation.



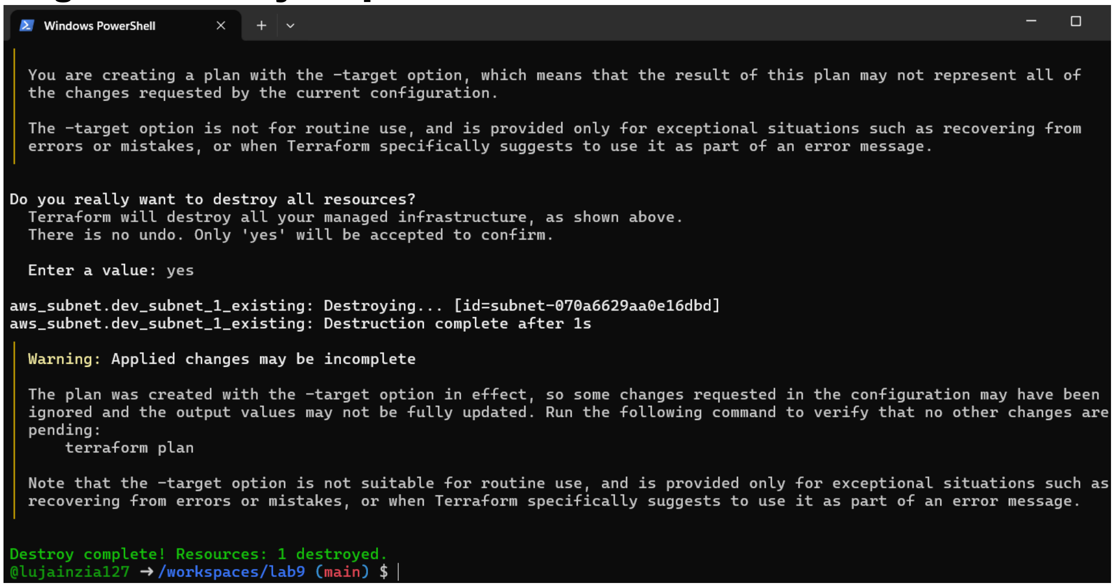
```
      + availability_zone                              = "me-central-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "172.31.48.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + region                                         = "me-central-1"
      + tags_all                                       = (known after apply)
      + vpc_id                                         = "vpc-01b7d554ea60c902d"
    }

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_subnet.dev_subnet_1_existing: Creating...
aws_subnet.dev_subnet_1_existing: Creation complete after 1s [id=subnet-070a6629aa0e16dbd]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

B. Targeted Destroy & Refresh

1. Destroy only one resource:

terraform destroy -target=aws_subnet.dev_subnet_1_existing

Save screenshot as: task4_terraform_destroy_targeted.png — targeted destroy output.

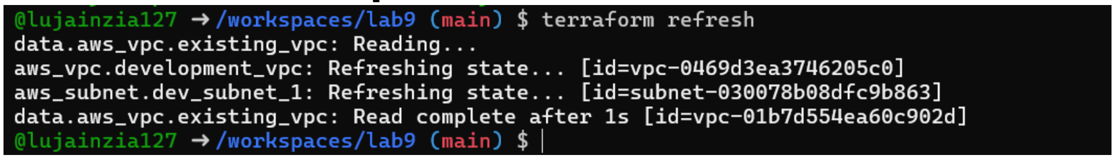


```
  You are creating a plan with the -target option, which means that the result of this plan may not represent all of
  the changes requested by the current configuration.

  The -target option is not for routine use, and is provided only for exceptional situations such as recovering from
  errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.


Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_subnet.dev_subnet_1_existing: Destroying... [id=subnet-070a6629aa0e16dbd]
aws_subnet.dev_subnet_1_existing: Destruction complete after 1s

  Warning: Applied changes may be incomplete

  The plan was created with the -target option in effect, so some changes requested in the configuration may have been
  ignored and the output values may not be fully updated. Run the following command to verify that no other changes are
  pending:
      terraform plan

  Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as
  recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.


Destroy complete! Resources: 1 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

2. Refresh state:

terraform refresh

Save screenshot as: task4_terraform_refresh_state.png — terraform refresh output.

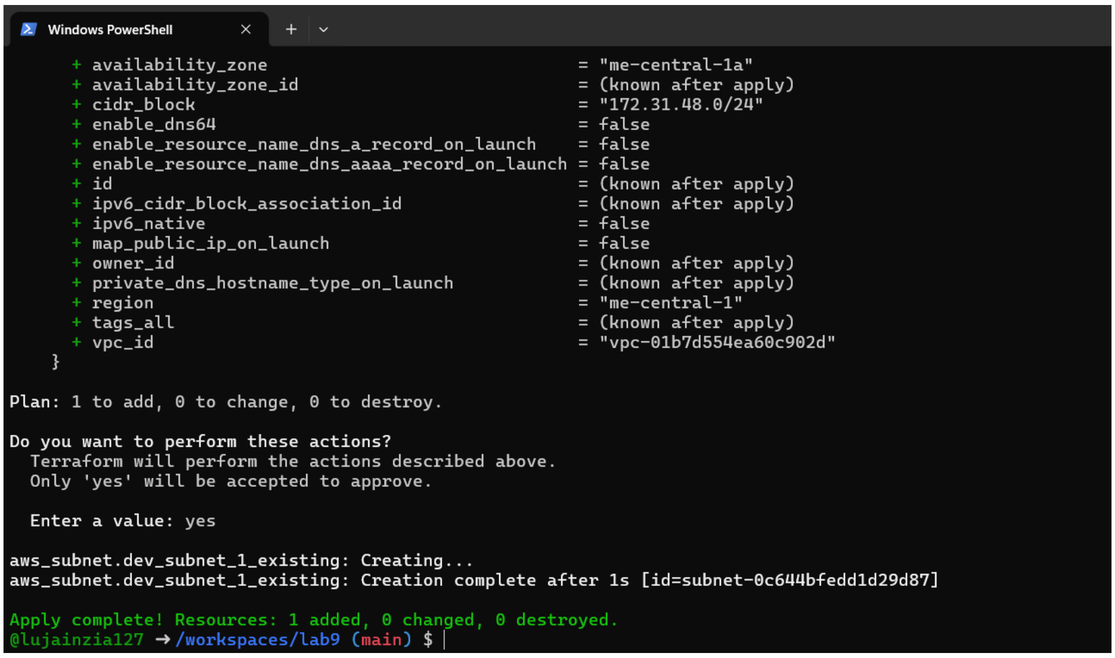


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform refresh
data.aws_vpc.existing_vpc: Reading...
aws_vpc.development_vpc: Refreshing state... [id=vpc-0469d3ea3746205c0]
aws_subnet.dev_subnet_1: Refreshing state... [id=subnet-030078b08dfc9b863]
data.aws_vpc.existing_vpc: Read complete after 1s [id=vpc-01b7d554ea60c902d]
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

3. Re-apply configuration:

terraform apply

Save screenshot as: task4_terraform_apply_after_refresh.png — terraform apply after refresh.

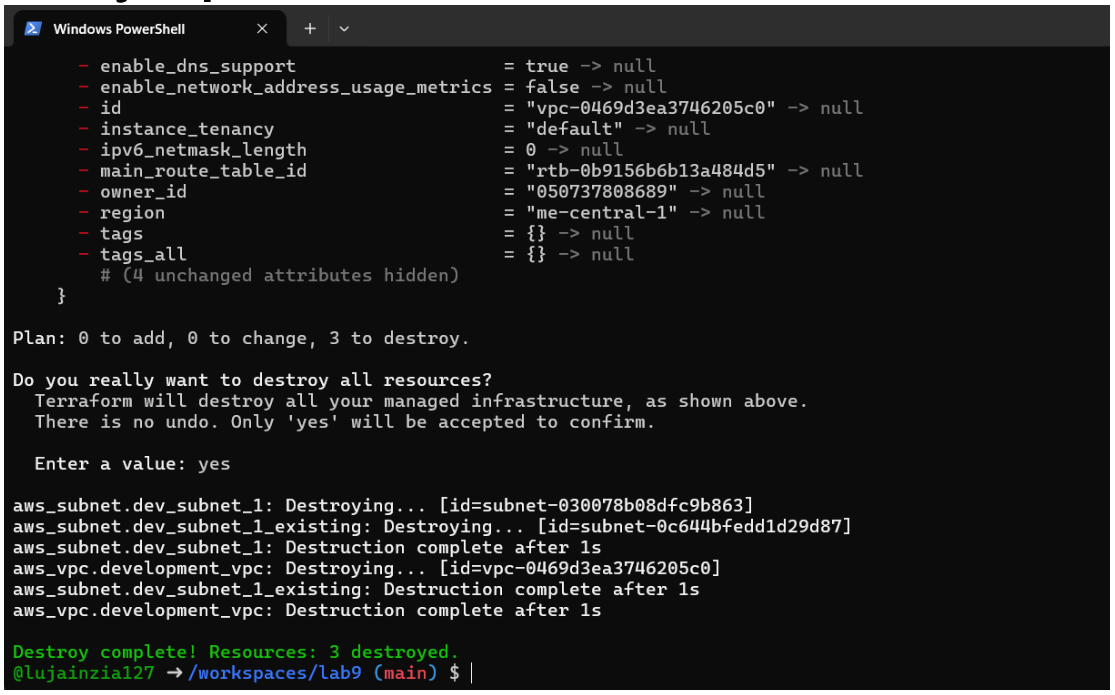


```
      + availability_zone                              = "me-central-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "172.31.48.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + region                                         = "me-central-1"
      + tags_all                                       = (known after apply)
      + vpc_id                                         = "vpc-01b7d554ea60c902d"
    }

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_subnet.dev_subnet_1_existing: Creating...
aws_subnet.dev_subnet_1_existing: Creation complete after 1s [id=subnet-0c644bfedd1d29d87]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

4. Destroy all resources:

terraform destroy

Save screenshot as: task4_terraform_destroy_all.png — full destroy output.

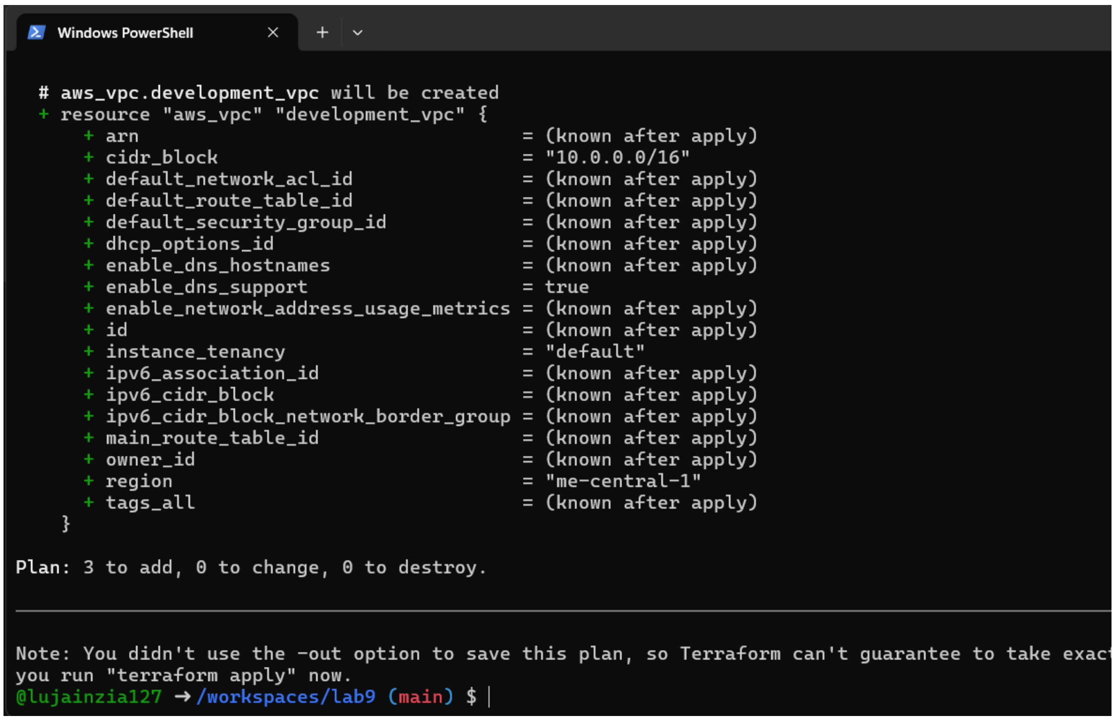


```
      - enable_dns_support                   = true -> null
      - enable_network_address_usage_metrics = false -> null
      - id                                   = "vpc-0469d3ea3746205c0" -> null
      - instance_tenancy                     = "default" -> null
      - ipv6_netmask_length                  = 0 -> null
      - main_route_table_id                  = "rtb-0b9156b6b13a484d5" -> null
      - owner_id                             = "050737808689" -> null
      - region                               = "me-central-1" -> null
      - tags                                 = {} -> null
      - tags_all                             = {} -> null
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 0 to change, 3 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_subnet.dev_subnet_1: Destroying... [id=subnet-030078b08dfc9b863]
aws_subnet.dev_subnet_1_existing: Destroying... [id=subnet-0c644bfedd1d29d87]
aws_subnet.dev_subnet_1: Destruction complete after 1s
aws_vpc.development_vpc: Destroying... [id=vpc-0469d3ea3746205c0]
aws_subnet.dev_subnet_1_existing: Destruction complete after 1s
aws_vpc.development_vpc: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

5. Plan configuration:

terraform plan

Save screenshot as: task4_terraform_plan_output.png — terraform plan showing next changes.



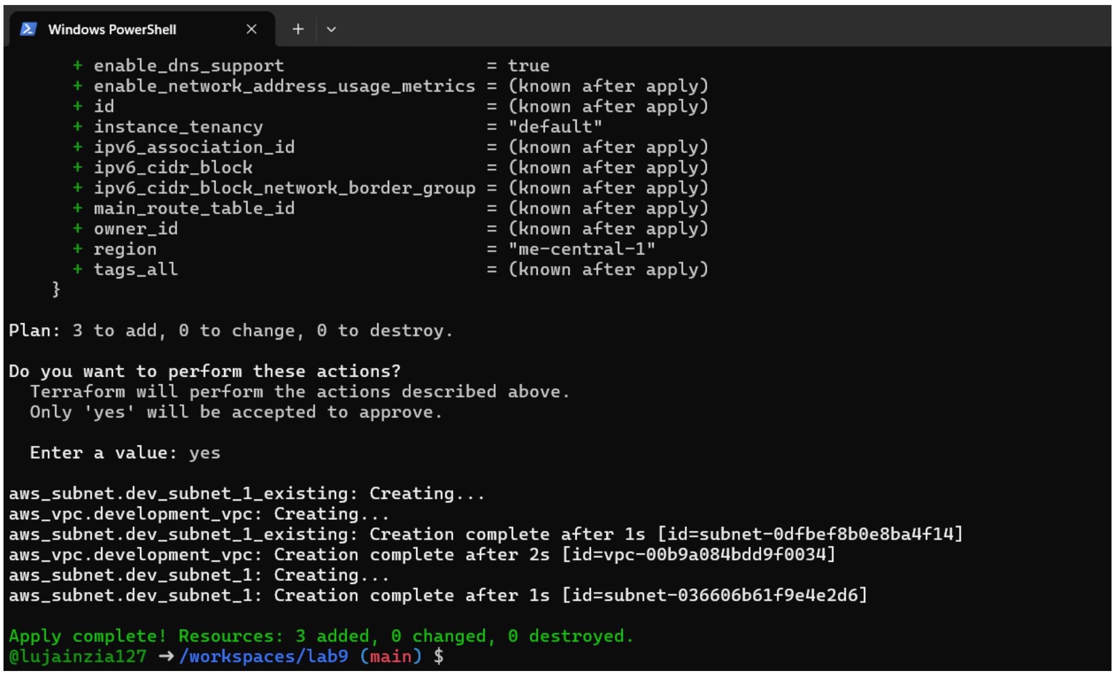
```
Windows PowerShell

  # aws_vpc.development_vpc will be created
  + resource "aws_vpc" "development_vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + region                               = "me-central-1"
      + tags_all                             = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exac
you run "terraform apply" now.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

6. Apply again:

terraform apply

Save screenshot

as: task4_terraform_apply_after_destroy.png — output showing resources recreated.



```
Windows PowerShell

      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + region                               = "me-central-1"
      + tags_all                             = (known after apply)
    }

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_subnet.dev_subnet_1_existing: Creating...
aws_vpc.development_vpc: Creating...
aws_subnet.dev_subnet_1_existing: Creation complete after 1s [id=subnet-0dfbef8b0e8ba4f14]
aws_vpc.development_vpc: Creation complete after 2s [id=vpc-00b9a084bdd9f0034]
aws_subnet.dev_subnet_1: Creating...
aws_subnet.dev_subnet_1: Creation complete after 1s [id=subnet-036606b61f9e4e2d6]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $
```

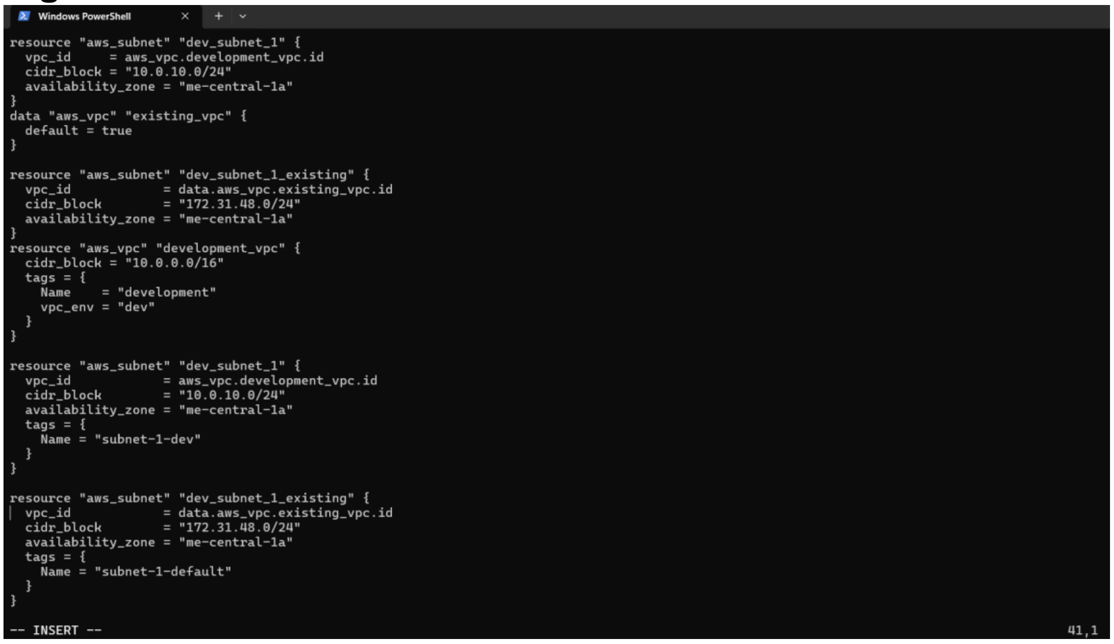
C. Tagging Resources

1. Modify main.tf to add tags:

```
resource "aws_vpc" "development_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name: "development"
    vpc_env = "dev"
  }
}
resource "aws_subnet" "dev_subnet_1" {
  vpc_id     = aws_vpc.development_vpc.id
  cidr_block = "10.0.10.0/24"
  availability_zone = "me-central-1a"
  tags = {
    Name: "subnet-1-dev"
  }
}
resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id     = data.aws_vpc.existing_vpc.id
  cidr_block = "172.31.48.0/24"
  availability_zone = "me-central-1a"
  tags = {
    Name: "subnet-1-default"
  }
}
```

Use vim to update tags.

Save screenshot as: task4_main_tf_tagging.png — main.tf with tags added.

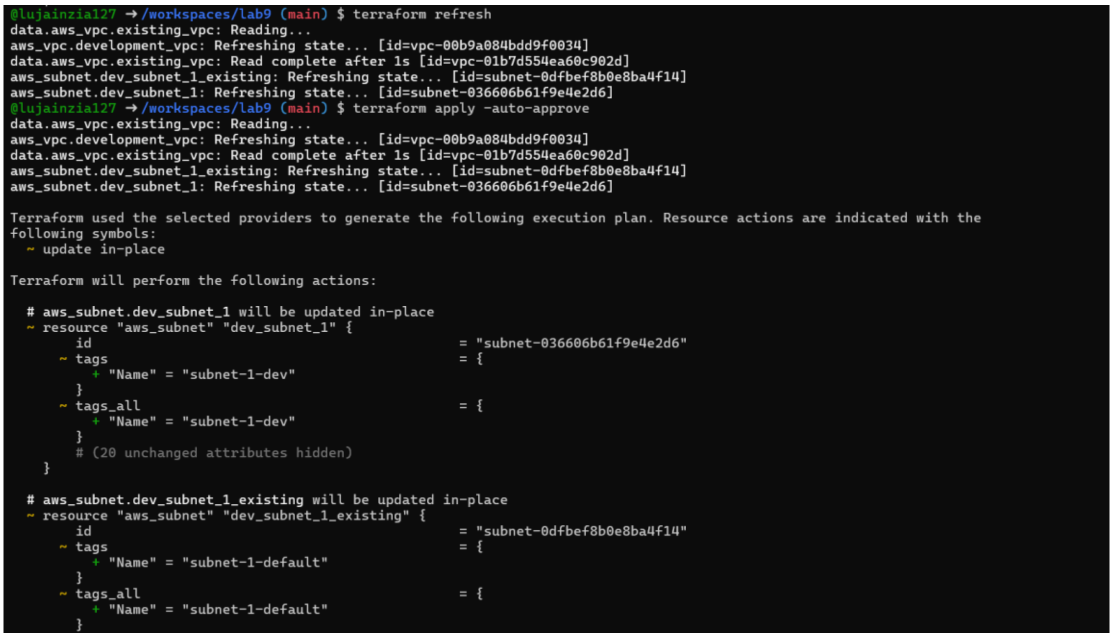


```
resource "aws_subnet" "dev_subnet_1" {
  vpc_id     = aws_vpc.development_vpc.id
  cidr_block = "10.0.10.0/24"
  availability_zone = "me-central-1a"
}
data "aws_vpc" "existing_vpc" {
  default = true
}

resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id            = data.aws_vpc.existing_vpc.id
  cidr_block        = "172.31.48.0/24"
  availability_zone = "me-central-1a"
}
resource "aws_vpc" "development_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name    = "development"
    vpc_env = "dev"
  }
}

resource "aws_subnet" "dev_subnet_1" {
  vpc_id            = aws_vpc.development_vpc.id
  cidr_block        = "10.0.10.0/24"
  availability_zone = "me-central-1a"
  tags = {
    Name = "subnet-1-dev"
  }
}

resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id            = data.aws_vpc.existing_vpc.id
  cidr_block        = "172.31.48.0/24"
  availability_zone = "me-central-1a"
  tags = {
    Name = "subnet-1-default"
  }
}

-- INSERT --                                                        41,1
```

2. Run:

terraform refresh
terraform apply -auto-approve

Save screenshot as: task4_terraform_apply_tagging.png — output showing tags applied.

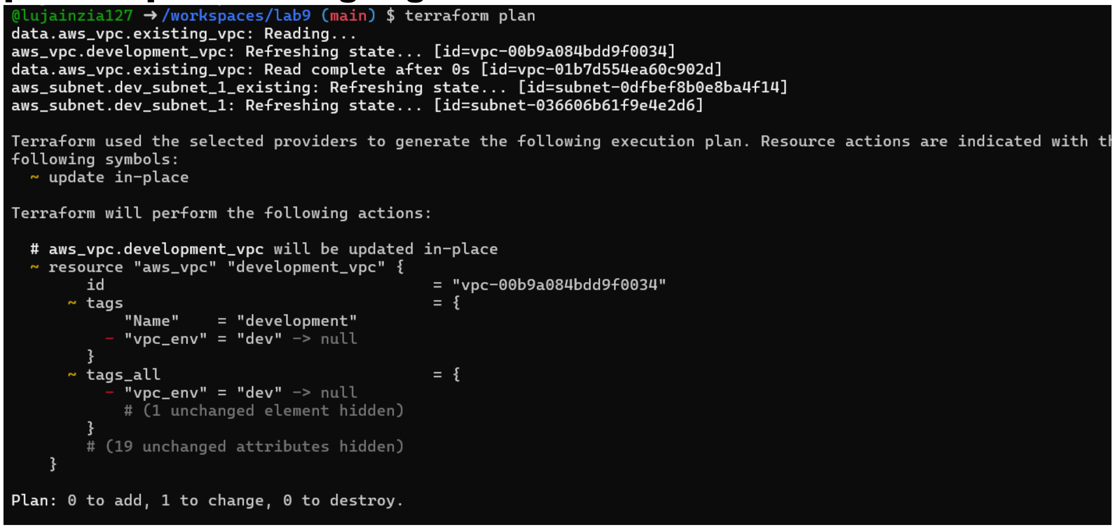


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform refresh
data.aws_vpc.existing_vpc: Reading...
aws_vpc.development_vpc: Refreshing state... [id=vpc-00b9a084bdd9f0034]
data.aws_vpc.existing_vpc: Read complete after 1s [id=vpc-01b7d554ea60c902d]
aws_subnet.dev_subnet_1_existing: Refreshing state... [id=subnet-0dfbef8b0e8ba4f14]
aws_subnet.dev_subnet_1: Refreshing state... [id=subnet-036606b61f9e4e2d6]
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform apply -auto-approve
data.aws_vpc.existing_vpc: Reading...
aws_vpc.development_vpc: Refreshing state... [id=vpc-00b9a084bdd9f0034]
data.aws_vpc.existing_vpc: Read complete after 1s [id=vpc-01b7d554ea60c902d]
aws_subnet.dev_subnet_1_existing: Refreshing state... [id=subnet-0dfbef8b0e8ba4f14]
aws_subnet.dev_subnet_1: Refreshing state... [id=subnet-036606b61f9e4e2d6]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_subnet.dev_subnet_1 will be updated in-place
  ~ resource "aws_subnet" "dev_subnet_1" {
        id                                             = "subnet-036606b61f9e4e2d6"
      ~ tags                                           = {
          + "Name" = "subnet-1-dev"
        }
      ~ tags_all                                       = {
          + "Name" = "subnet-1-dev"
        }
        # (20 unchanged attributes hidden)
    }

  # aws_subnet.dev_subnet_1_existing will be updated in-place
  ~ resource "aws_subnet" "dev_subnet_1_existing" {
        id                                             = "subnet-0dfbef8b0e8ba4f14"
      ~ tags                                           = {
          + "Name" = "subnet-1-default"
        }
      ~ tags_all                                       = {
          + "Name" = "subnet-1-default"
        }
```

3. Remove vpc_env = "dev" tag

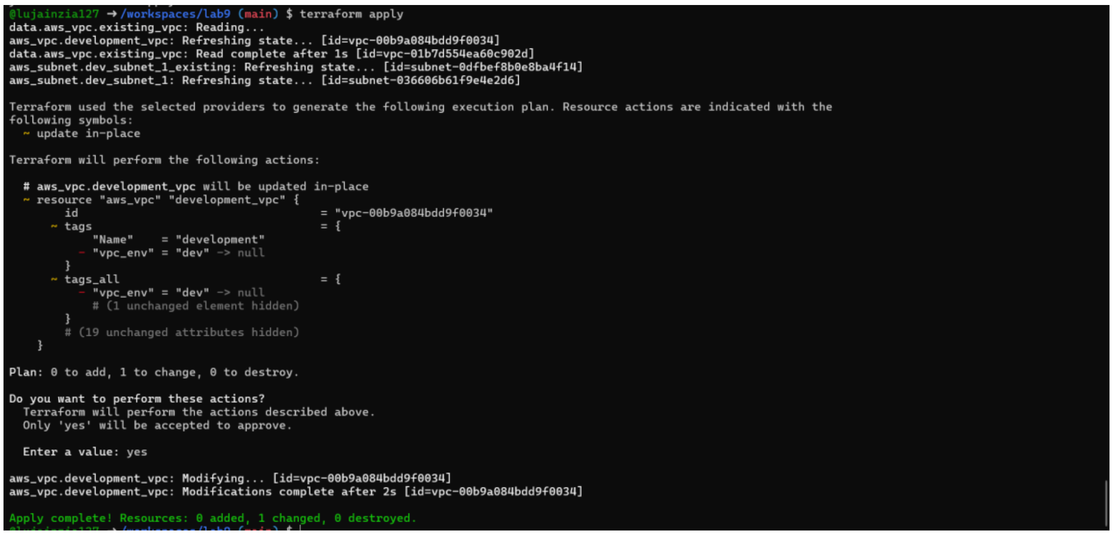
from development_vpc resource, re-plan/apply:

Save screenshot as: task4_terraform_plan_remove_tag.png — plan output showing tag removal.



```
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform plan
data.aws_vpc.existing_vpc: Reading...
aws_vpc.development_vpc: Refreshing state... [id=vpc-00b9a084bdd9f0034]
data.aws_vpc.existing_vpc: Read complete after 0s [id=vpc-01b7d554ea60c902d]
aws_subnet.dev_subnet_1_existing: Refreshing state... [id=subnet-0dfbef8b0e8ba4f14]
aws_subnet.dev_subnet_1: Refreshing state... [id=subnet-036606b61f9e4e2d6]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with t
following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_vpc.development_vpc will be updated in-place
  ~ resource "aws_vpc" "development_vpc" {
        id                                   = "vpc-00b9a084bdd9f0034"
      ~ tags                                 = {
            "Name"    = "development"
          - "vpc_env" = "dev" -> null
        }
      ~ tags_all                             = {
          - "vpc_env" = "dev" -> null
            # (1 unchanged element hidden)
        }
        # (19 unchanged attributes hidden)
    }

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

Save screenshot as: task4_terraform_apply_remove_tag.png — apply output showing tag deleted.



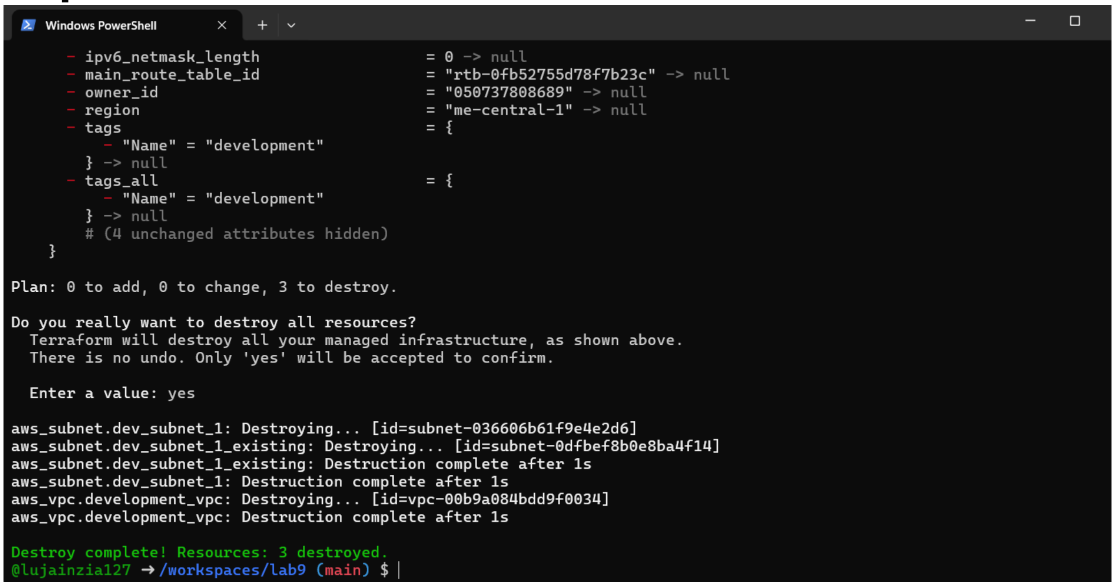
```
@lujainzia127 → /workspaces/lab9 (main) $ terraform apply
data.aws_vpc.existing_vpc: Reading...
aws_vpc.development_vpc: Refreshing state... [id=vpc-00b9a084bdd9f0034]
data.aws_vpc.existing_vpc: Read complete after 1s [id=vpc-01b7d554ea60c902d]
aws_subnet.dev_subnet_1_existing: Refreshing state... [id=subnet-0dfbef8b0e8ba4f14]
aws_subnet.dev_subnet_1: Refreshing state... [id=subnet-036606b61f9e4e2d6]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_vpc.development_vpc will be updated in-place
  ~ resource "aws_vpc" "development_vpc" {
        id                                   = "vpc-00b9a084bdd9f0034"
      ~ tags                                 = {
            "Name"    = "development"
          - "vpc_env" = "dev" -> null
        }
      ~ tags_all                             = {
          - "vpc_env" = "dev" -> null
            # (1 unchanged element hidden)
        }
        # (19 unchanged attributes hidden)
    }

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.development_vpc: Modifying... [id=vpc-00b9a084bdd9f0034]
aws_vpc.development_vpc: Modifications complete after 2s [id=vpc-00b9a084bdd9f0034]

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

Task 5 — State File Inspection & Terraform State Commands

1. Destroy all resources:

terraform destroy

Save screenshot as: task5_terraform_destroy.png — destroy output.



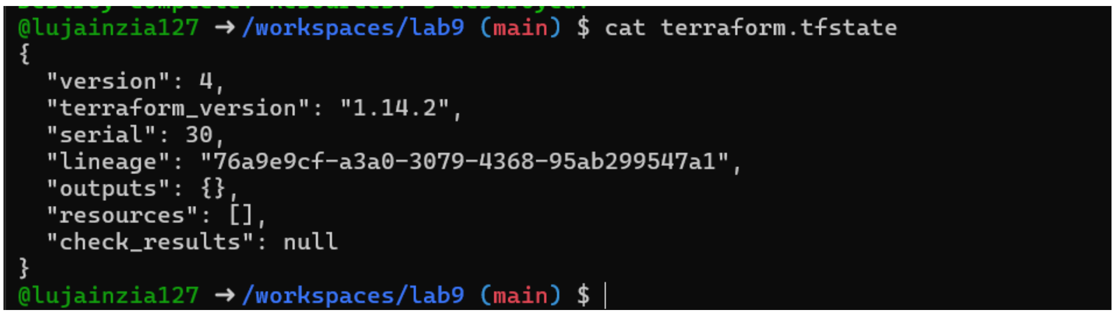
```
      - ipv6_netmask_length                  = 0 -> null
      - main_route_table_id                  = "rtb-0fb52755d78f7b23c" -> null
      - owner_id                             = "050737808689" -> null
      - region                               = "me-central-1" -> null
      - tags                                 = {
          - "Name" = "development"
        } -> null
      - tags_all                             = {
          - "Name" = "development"
        } -> null
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 0 to change, 3 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_subnet.dev_subnet_1: Destroying... [id=subnet-036606b61f9e4e2d6]
aws_subnet.dev_subnet_1_existing: Destroying... [id=subnet-0dfbef8b0e8ba4f14]
aws_subnet.dev_subnet_1_existing: Destruction complete after 1s
aws_subnet.dev_subnet_1: Destruction complete after 1s
aws_vpc.development_vpc: Destroying... [id=vpc-00b9a084bdd9f0034]
aws_vpc.development_vpc: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
@lujainzia127 → /workspaces/lab9 (main) $
```

2. Inspect state files after destroy:

Run:

cat terraform.tfstate

Save screenshot as: task5_terraform_state_file_empty.png — output showing terraform.tfstate is empty after destroy.

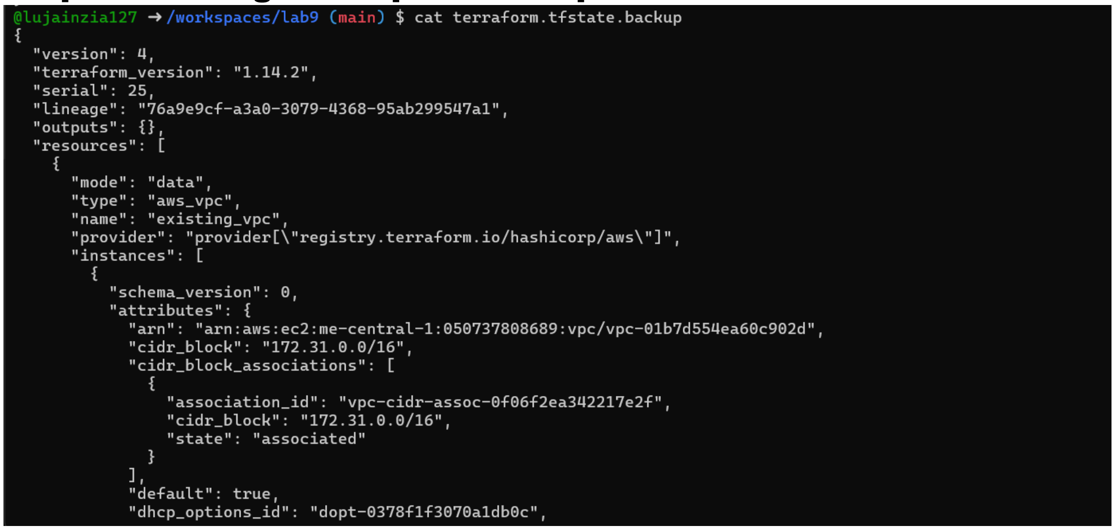


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 30,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {},
  "resources": [],
  "check_results": null
}
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

Run:

cat terraform.tfstate.backup

Save screenshot as: task5_terraform_state_backup_prev.png — output showing backup file with previous resources.

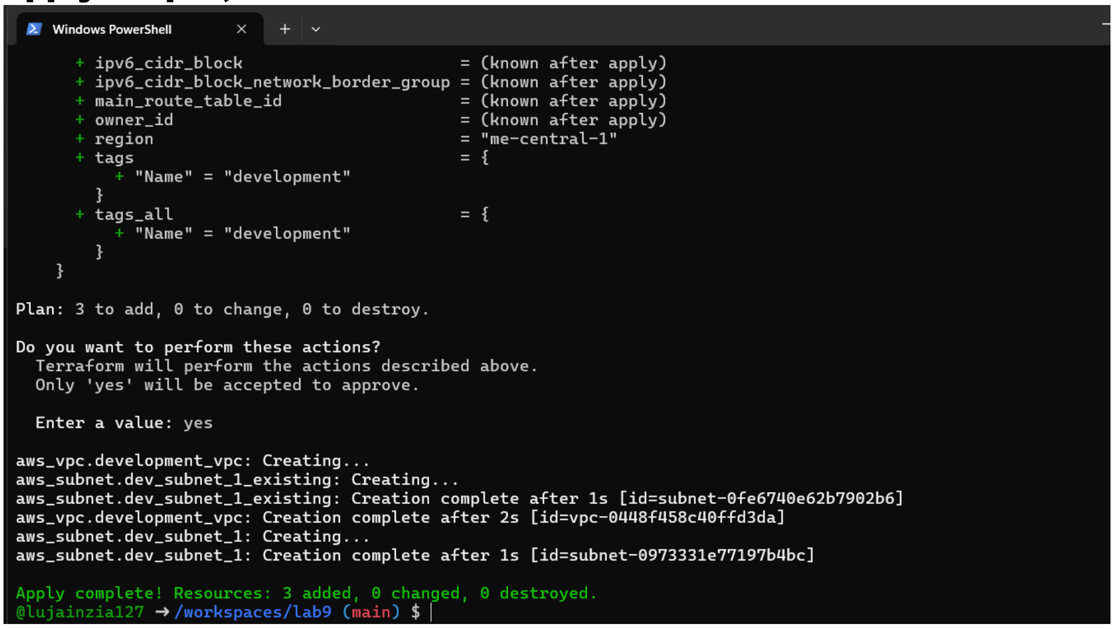


```
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat terraform.tfstate.backup
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 25,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {},
  "resources": [
    {
      "mode": "data",
      "type": "aws_vpc",
      "name": "existing_vpc",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "arn": "arn:aws:ec2:me-central-1:050737808689:vpc/vpc-01b7d554ea60c902d",
            "cidr_block": "172.31.0.0/16",
            "cidr_block_associations": [
              {
                "association_id": "vpc-cidr-assoc-0f06f2ea342217e2f",
                "cidr_block": "172.31.0.0/16",
                "state": "associated"
              }
            ],
            "default": true,
            "dhcp_options_id": "dopt-0378f1f3070a1db0c",
```

3. Recreate resources:

terraform apply

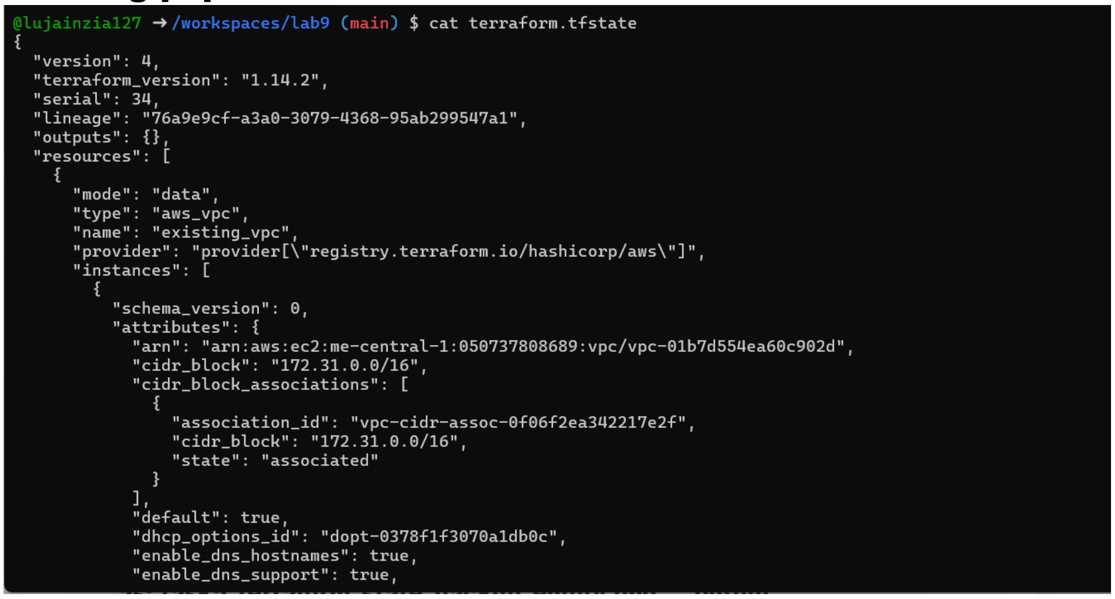
Save screenshot as: task5_terraform_apply_recreated.png — apply output, resources recreated.



```
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + region                               = "me-central-1"
      + tags                                 = {
          + "Name" = "development"
        }
      + tags_all                             = {
          + "Name" = "development"
        }
    }

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.development_vpc: Creating...
aws_subnet.dev_subnet_1_existing: Creating...
aws_subnet.dev_subnet_1_existing: Creation complete after 1s [id=subnet-0fe6740e62b7902b6]
aws_vpc.development_vpc: Creation complete after 2s [id=vpc-0448f458c40ffd3da]
aws_subnet.dev_subnet_1: Creating...
aws_subnet.dev_subnet_1: Creation complete after 1s [id=subnet-0973331e77197b4bc]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

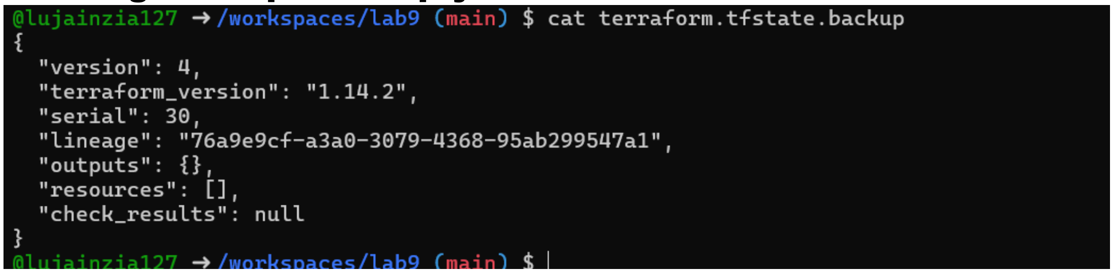
4. View state files:

Run:
cat terraform.tfstate
Save screenshot as: task5_terraform_state_file_populated.png — output showing populated terraform.tfstate after restore.



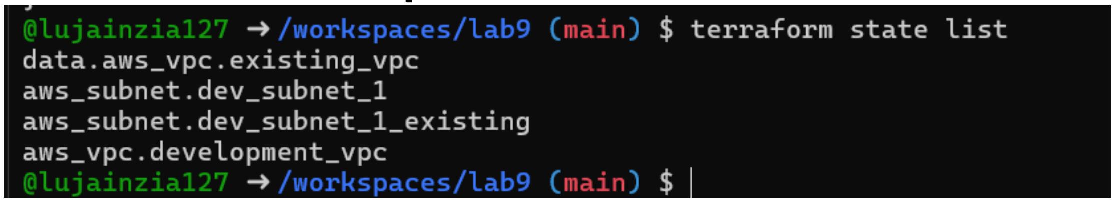
```
@lujainzia127 → /workspaces/lab9 (main) $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 34,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {},
  "resources": [
    {
      "mode": "data",
      "type": "aws_vpc",
      "name": "existing_vpc",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "arn": "arn:aws:ec2:me-central-1:050737808689:vpc/vpc-01b7d554ea60c902d",
            "cidr_block": "172.31.0.0/16",
            "cidr_block_associations": [
              {
                "association_id": "vpc-cidr-assoc-0f06f2ea342217e2f",
                "cidr_block": "172.31.0.0/16",
                "state": "associated"
              }
            ],
            "default": true,
            "dhcp_options_id": "dopt-0378f1f3070a1db0c",
            "enable_dns_hostnames": true,
            "enable_dns_support": true,
```

Run:
cat terraform.tfstate.backup
Save screenshot as: task5_terraform_state_backup_empty.png — output showing backup file empty after restore.



```
@lujainzia127 → /workspaces/lab9 (main) $ cat terraform.tfstate.backup
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 30,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {},
  "resources": [],
  "check_results": null
}
@lujainzia127 → /workspaces/lab9 (main) $
```

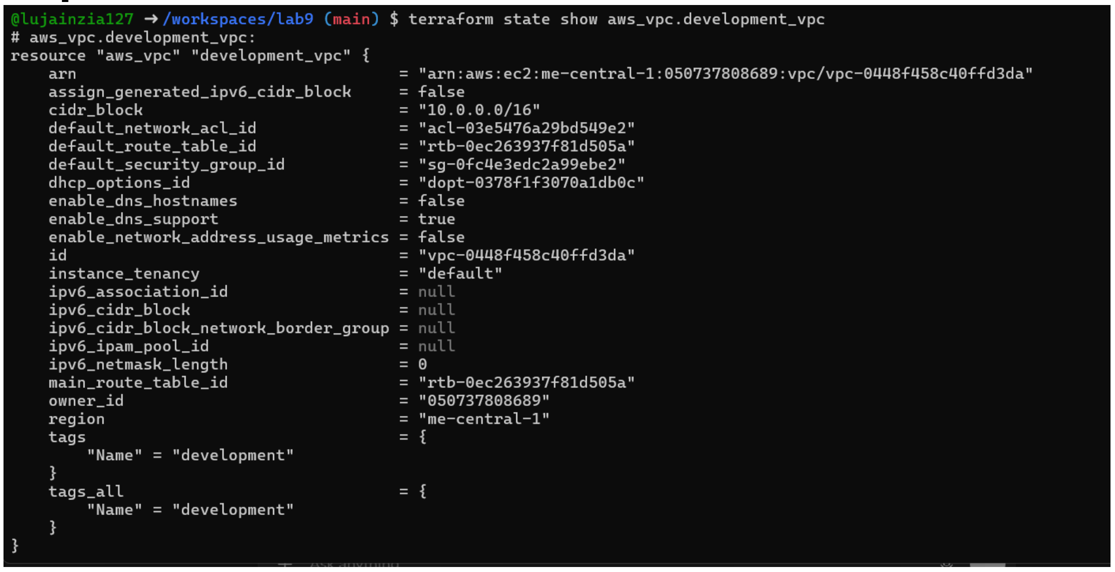
5. List resources:
terraform state list
Save screenshot as: task5_terraform_state_list.png — terraform state list output.



```
@lujainzia127 → /workspaces/lab9 (main) $ terraform state list
data.aws_vpc.existing_vpc
aws_subnet.dev_subnet_1
aws_subnet.dev_subnet_1_existing
aws_vpc.development_vpc
@lujainzia127 → /workspaces/lab9 (main) $
```

6. Show full attributes:
terraform state show <resource-name>

Save screenshot as: task5_terraform_state_show_resource.png — state show output for one resource.



```
@lujainzia127 ➜ /workspaces/lab9 (main) $ terraform state show aws_vpc.development_vpc
# aws_vpc.development_vpc:
resource "aws_vpc" "development_vpc" {
    arn                                  = "arn:aws:ec2:me-central-1:050737808689:vpc/vpc-0448f458c40ffd3da"
    assign_generated_ipv6_cidr_block     = false
    cidr_block                           = "10.0.0.0/16"
    default_network_acl_id               = "acl-03e5476a29bd549e2"
    default_route_table_id               = "rtb-0ec263937f81d505a"
    default_security_group_id            = "sg-0fc4e3edc2a99ebe2"
    dhcp_options_id                      = "dopt-0378f1f3070a1db0c"
    enable_dns_hostnames                 = false
    enable_dns_support                   = true
    enable_network_address_usage_metrics = false
    id                                   = "vpc-0448f458c40ffd3da"
    instance_tenancy                     = "default"
    ipv6_association_id                  = null
    ipv6_cidr_block                      = null
    ipv6_cidr_block_network_border_group = null
    ipv6_ipam_pool_id                    = null
    ipv6_netmask_length                  = 0
    main_route_table_id                  = "rtb-0ec263937f81d505a"
    owner_id                             = "050737808689"
    region                               = "me-central-1"
    tags                                 = {
        "Name" = "development"
    }
    tags_all                             = {
        "Name" = "development"
    }
}
```

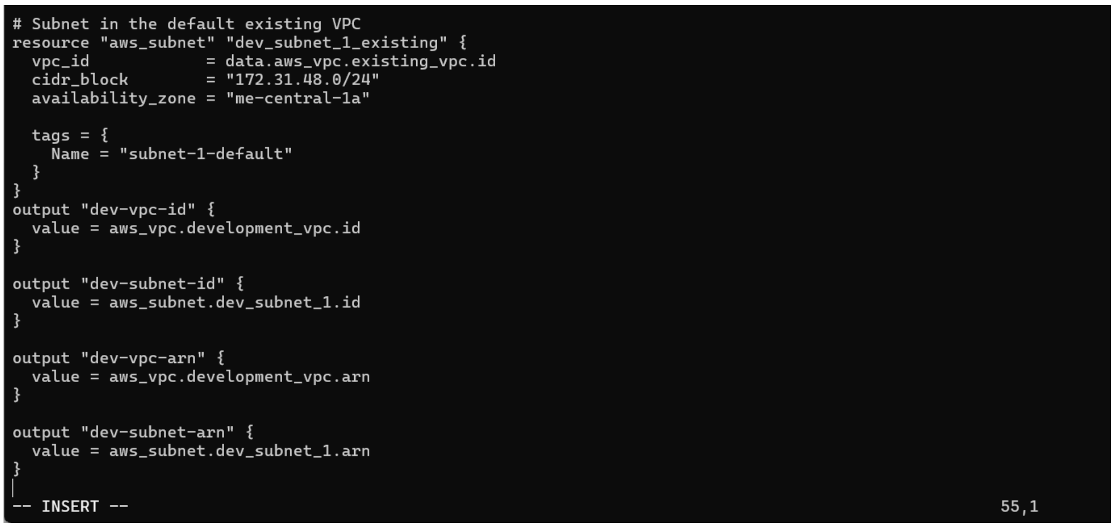
7. Note: Do NOT run the following (for information only):
terraform state rm <resource-name>
Mentioned for knowledge — do not run.

Task 6 — Terraform Outputs & Attributes Reporting

1. Add outputs in main.tf:

```
output "dev-vpc-id" {
  value = aws_vpc.development_vpc.id
}output "dev-subnet-id" {
  value = aws_subnet.dev_subnet_1.id
}output "dev-vpc-arn" {
  value = aws_vpc.development_vpc.arn
}output "dev-subnet-arn" {
  value = aws_subnet.dev_subnet_1.arn
}
```

Save screenshot as: task6_terraform_outputs_basic.png — output values for ids and arns after apply.



```
# Subnet in the default existing VPC
resource "aws_subnet" "dev_subnet_1_existing" {
  vpc_id            = data.aws_vpc.existing_vpc.id
  cidr_block        = "172.31.48.0/24"
  availability_zone = "me-central-1a"

  tags = {
    Name = "subnet-1-default"
  }
}
output "dev-vpc-id" {
  value = aws_vpc.development_vpc.id
}

output "dev-subnet-id" {
  value = aws_subnet.dev_subnet_1.id
}

output "dev-vpc-arn" {
  value = aws_vpc.development_vpc.arn
}

output "dev-subnet-arn" {
  value = aws_subnet.dev_subnet_1.arn
}
|
-- INSERT --                                                                    55,1
```

2. Return ALL of the following output attributes from your VPC and Subnet resources:

a) cidr_block

b) region

c) tags.Name

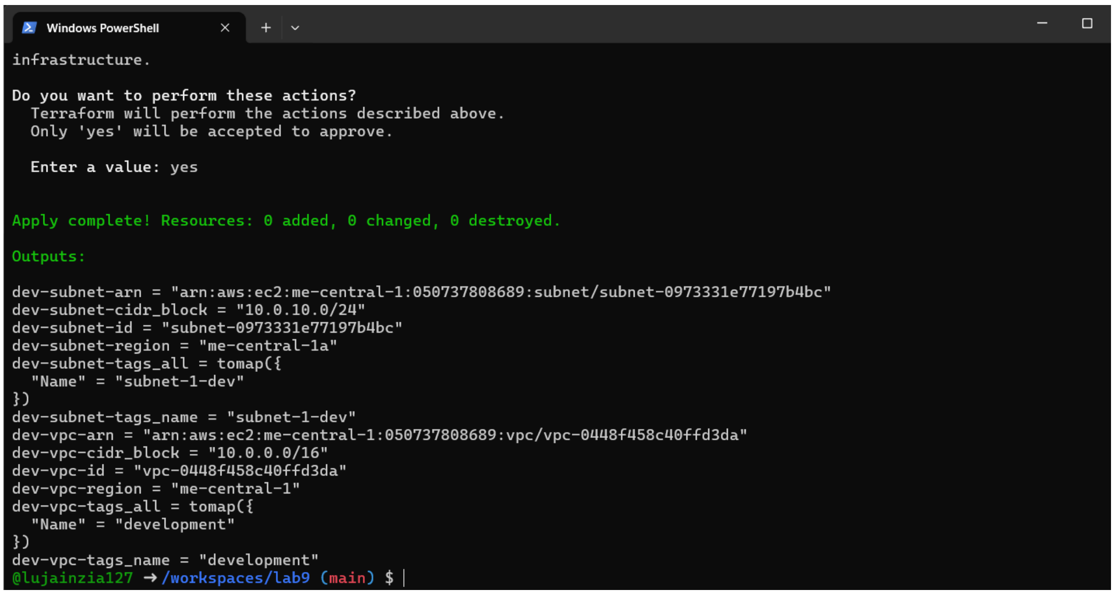
d) tags_all

Expand output section in main.tf accordingly:

```
output "dev-vpc-cidr_block" {
  value = aws_vpc.development_vpc.cidr_block
}output "dev-vpc-region" {
  value = aws_vpc.development_vpc.region
}output "dev-vpc-tags_name" {
  value = aws_vpc.development_vpc.tags["Name"]
}output "dev-vpc-tags_all" {
  value = aws_vpc.development_vpc.tags_all
}output "dev-subnet-cidr_block" {
  value = aws_subnet.dev_subnet_1.cidr_block
}output "dev-subnet-region" {
  value = aws_subnet.dev_subnet_1.availability_zone
}output "dev-subnet-tags_name" {
  value = aws_subnet.dev_subnet_1.tags["Name"]
}output "dev-subnet-tags_all" {
  value = aws_subnet.dev_subnet_1.tags_all
}
```

Run terraform apply and record outputs.

Save screenshot as: task6_expanded_outputs.png — output values for each required attribute after apply.



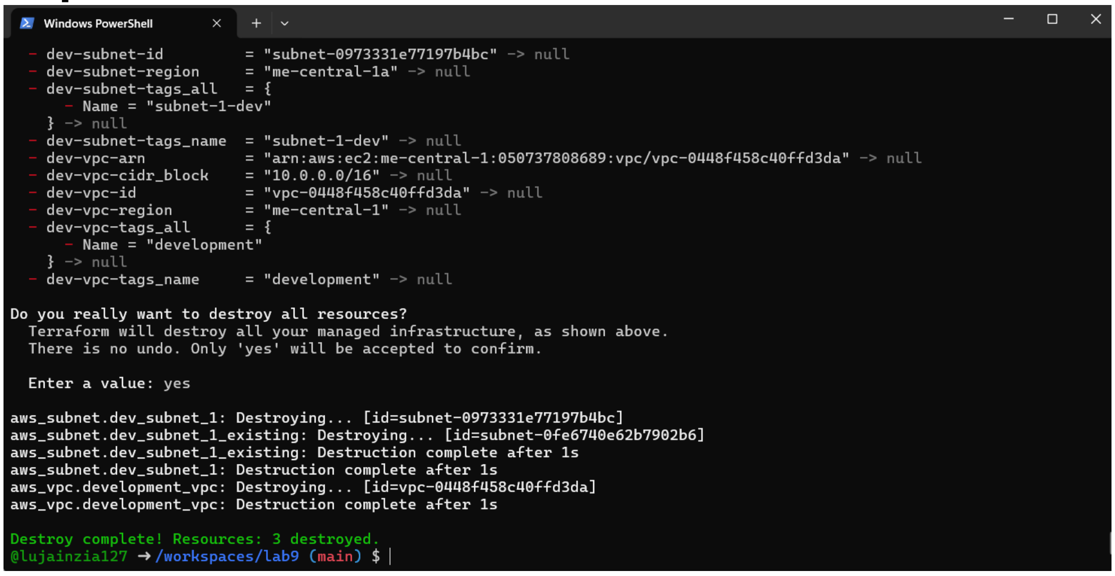
```
infrastructure.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes


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

Outputs:

dev-subnet-arn = "arn:aws:ec2:me-central-1:050737808689:subnet/subnet-0973331e77197b4bc"
dev-subnet-cidr_block = "10.0.10.0/24"
dev-subnet-id = "subnet-0973331e77197b4bc"
dev-subnet-region = "me-central-1a"
dev-subnet-tags_all = tomap({
  "Name" = "subnet-1-dev"
})
dev-subnet-tags_name = "subnet-1-dev"
dev-vpc-arn = "arn:aws:ec2:me-central-1:050737808689:vpc/vpc-0448f458c40ffd3da"
dev-vpc-cidr_block = "10.0.0.0/16"
dev-vpc-id = "vpc-0448f458c40ffd3da"
dev-vpc-region = "me-central-1"
dev-vpc-tags_all = tomap({
  "Name" = "development"
})
dev-vpc-tags_name = "development"
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

Cleanup — Delete Resources & State Verification

1. Destroy all resources:

terraform destroy

Save screenshot as: cleanup_destroy_resources.png — destroy output.

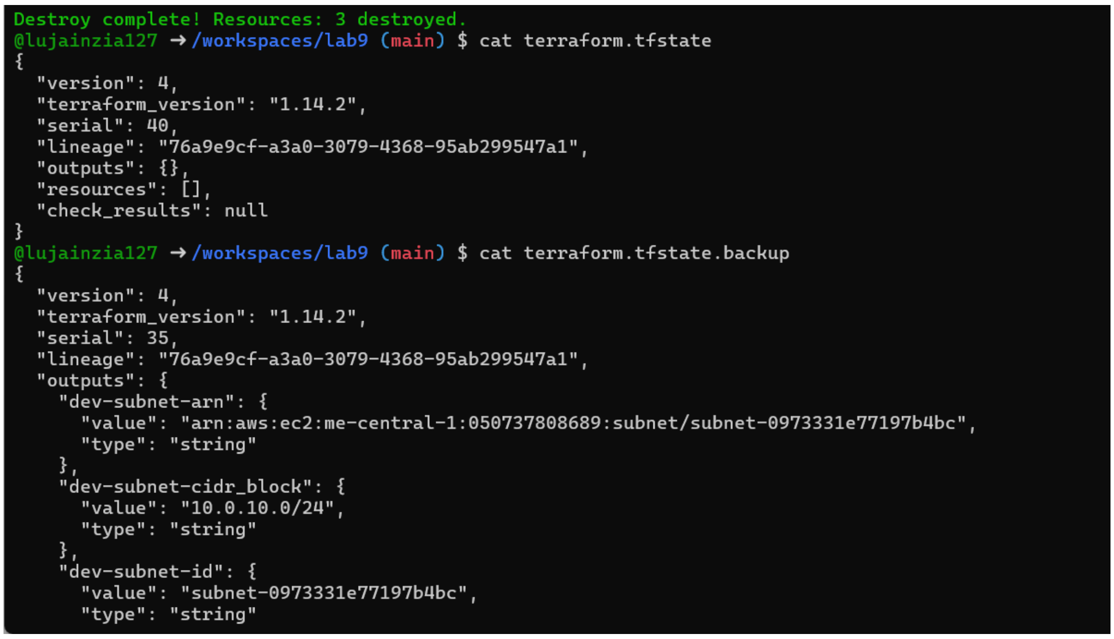


```
  - dev-subnet-id          = "subnet-0973331e77197b4bc" -> null
  - dev-subnet-region      = "me-central-1a" -> null
  - dev-subnet-tags_all    = {
      - Name = "subnet-1-dev"
    } -> null
  - dev-subnet-tags_name   = "subnet-1-dev" -> null
  - dev-vpc-arn            = "arn:aws:ec2:me-central-1:050737808689:vpc/vpc-0448f458c40ffd3da" -> null
  - dev-vpc-cidr_block     = "10.0.0.0/16" -> null
  - dev-vpc-id             = "vpc-0448f458c40ffd3da" -> null
  - dev-vpc-region         = "me-central-1" -> null
  - dev-vpc-tags_all       = {
      - Name = "development"
    } -> null
  - dev-vpc-tags_name      = "development" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_subnet.dev_subnet_1: Destroying... [id=subnet-0973331e77197b4bc]
aws_subnet.dev_subnet_1_existing: Destroying... [id=subnet-0fe6740e62b7902b6]
aws_subnet.dev_subnet_1_existing: Destruction complete after 1s
aws_subnet.dev_subnet_1: Destruction complete after 1s
aws_vpc.development_vpc: Destroying... [id=vpc-0448f458c40ffd3da]
aws_vpc.development_vpc: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ |
```

2. Inspect state files:

cat terraform.tfstate
cat terraform.tfstate.backup

Save screenshot as: cleanup_state_files.png — outputs showing file contents after clean-up.



```
Destroy complete! Resources: 3 destroyed.
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 40,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {},
  "resources": [],
  "check_results": null
}
@lujainzia127 ➜ /workspaces/lab9 (main) $ cat terraform.tfstate.backup
{
  "version": 4,
  "terraform_version": "1.14.2",
  "serial": 35,
  "lineage": "76a9e9cf-a3a0-3079-4368-95ab299547a1",
  "outputs": {
    "dev-subnet-arn": {
      "value": "arn:aws:ec2:me-central-1:050737808689:subnet/subnet-0973331e77197b4bc",
      "type": "string"
    },
    "dev-subnet-cidr_block": {
      "value": "10.0.10.0/24",
      "type": "string"
    },
    "dev-subnet-id": {
      "value": "subnet-0973331e77197b4bc",
      "type": "string"
```

Reapply then compare state and backup files to see differences.