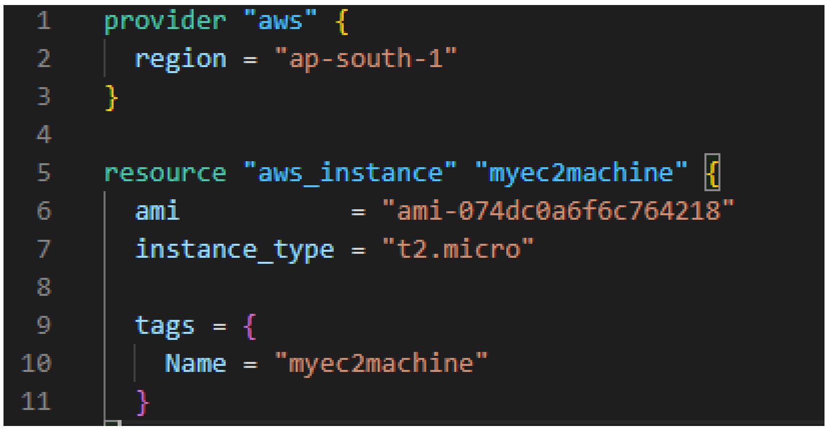
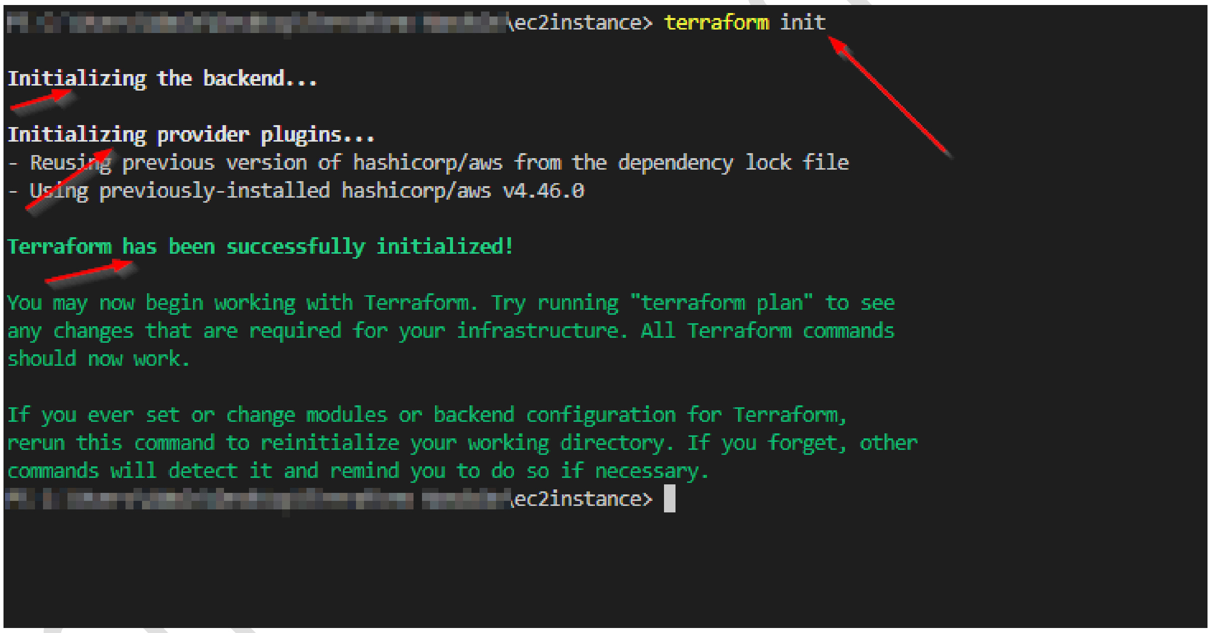
Basic Terraform Operations

1) **terraform init**: The terraform init command is used to initialize a working directory containing Terraform configuration files. Open any text editor like vscode for example and type the below code



```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "myec2machine" {
  ami           = "ami-074dc0a6f6c764218"
  instance_type = "t2.micro"

  tags = {
    Name = "myec2machine"
  }
```

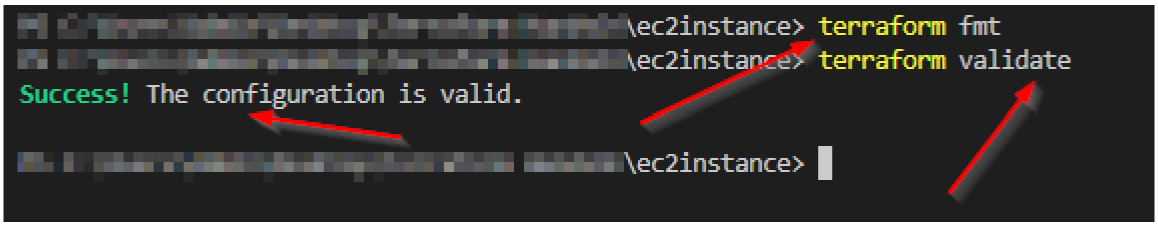
Save this file inside the same directory of the terraform. Now open the Command prompt, navigate to your project folder and type terraform init. It will initialize all the provider's plugins.



```
[redacted]\ec2instance> terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.46.0

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.
[redacted]\ec2instance>
```

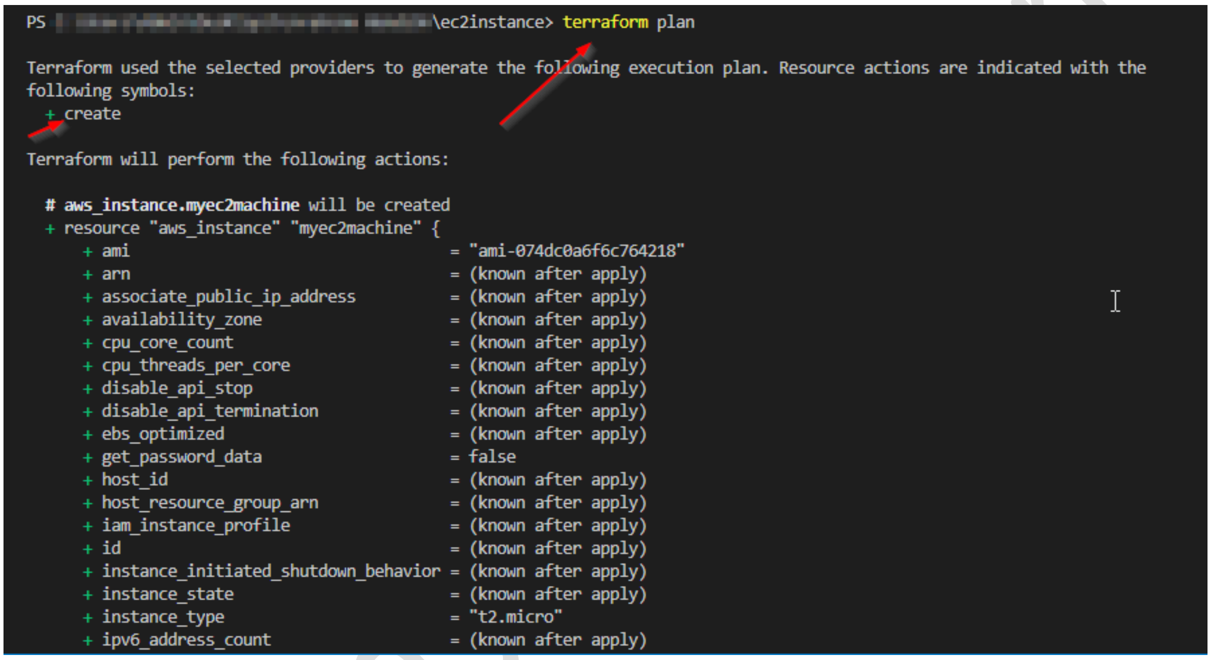
2. terraform fmt: The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability.

3. terraform validate: The terraform validate command is used to validate the configuration in a directory and checks whether a configuration is syntactically valid. It is used for general verification of correctness of attribute names and value types.

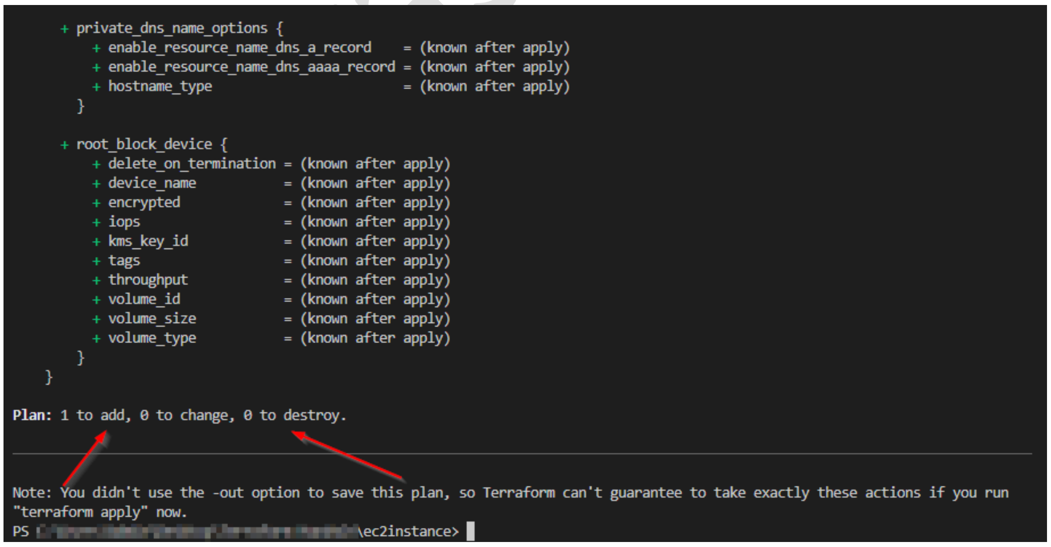


```
[redacted]\ec2instance> terraform fmt
[redacted]\ec2instance> terraform validate
Success! The configuration is valid.

[redacted]\ec2instance>
```

4) terraform plan: The terraform plan command is used to create an execution plan. Now we have the code written in text editor and we will tell the terraform to plan it. This will create an AWS instance with the specified AMI and the type.



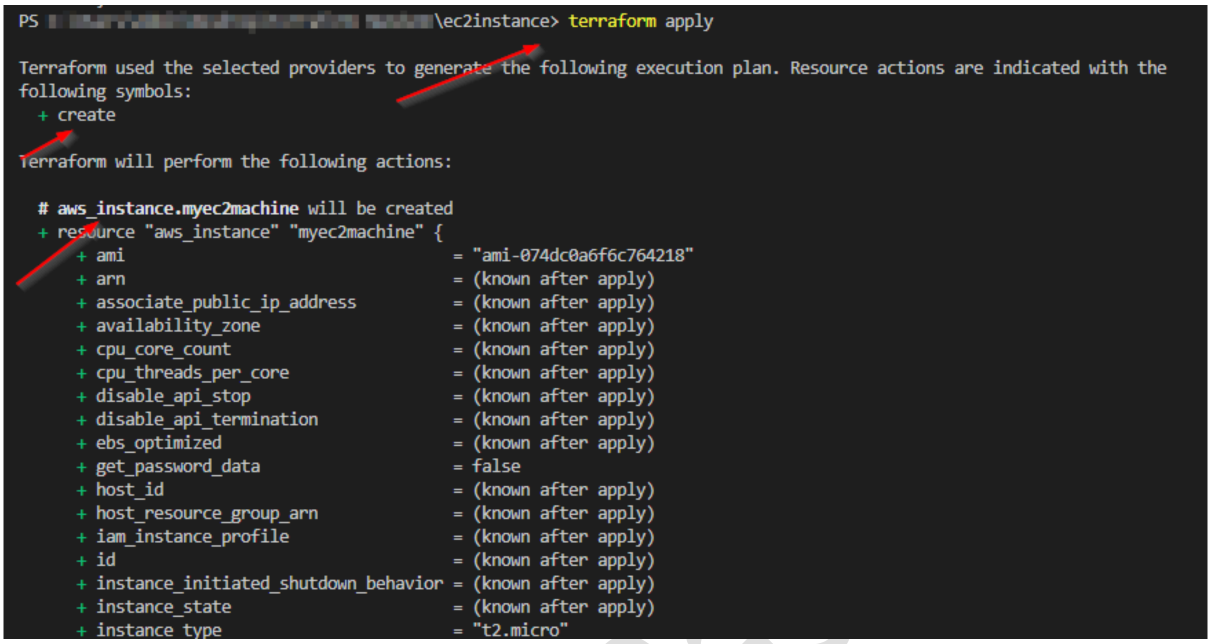
```
PS [redacted]\ec2instance> terraform plan

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_instance.myec2machine will be created
  + resource "aws_instance" "myec2machine" {
      + ami                                  = "ami-074dc0a6f6c764218"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
```



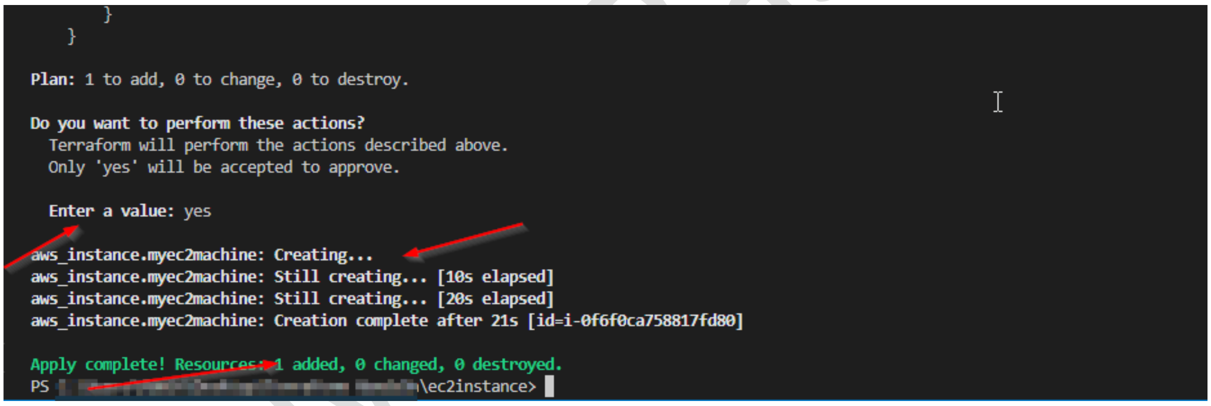
```
      + private_dns_name_options {
          + enable_resource_name_dns_a_record    = (known after apply)
          + enable_resource_name_dns_aaaa_record = (known after apply)
          + hostname_type                        = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 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 exactly these actions if you run
"terraform apply" now.
PS [redacted]\ec2instance>
```

This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state.

5) terraform apply: The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan. After this terraform apply an AWS instance will be created on the EC2.



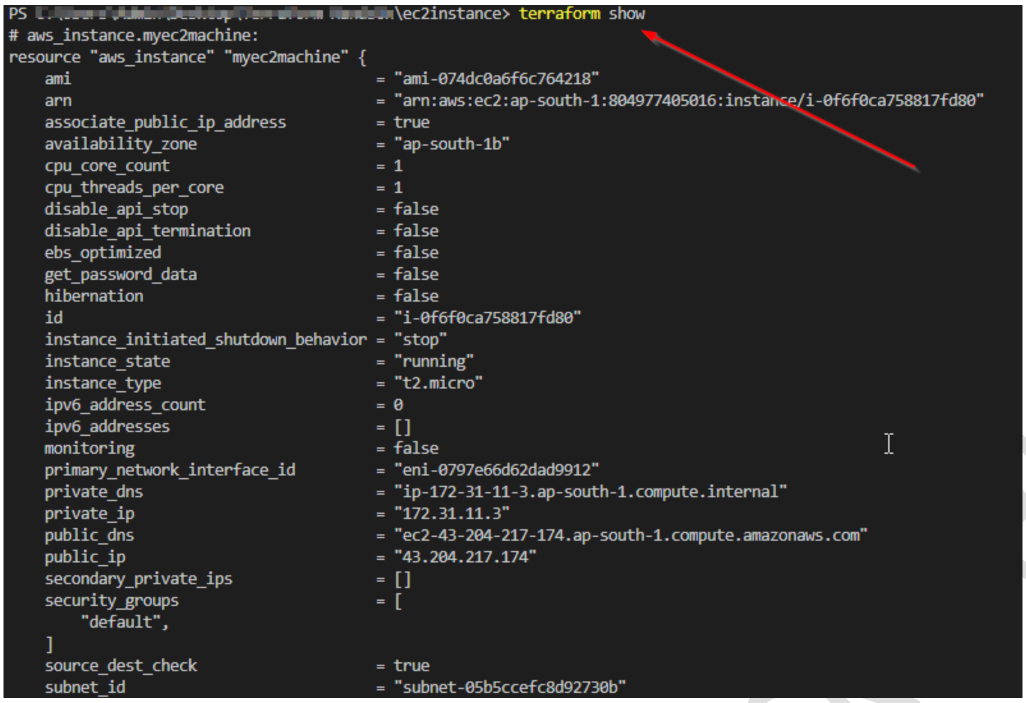
```
PS [redacted]\ec2instance> terraform apply

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_instance.myec2machine will be created
  + resource "aws_instance" "myec2machine" {
      + ami                                  = "ami-074dc0a6f6c764218"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```



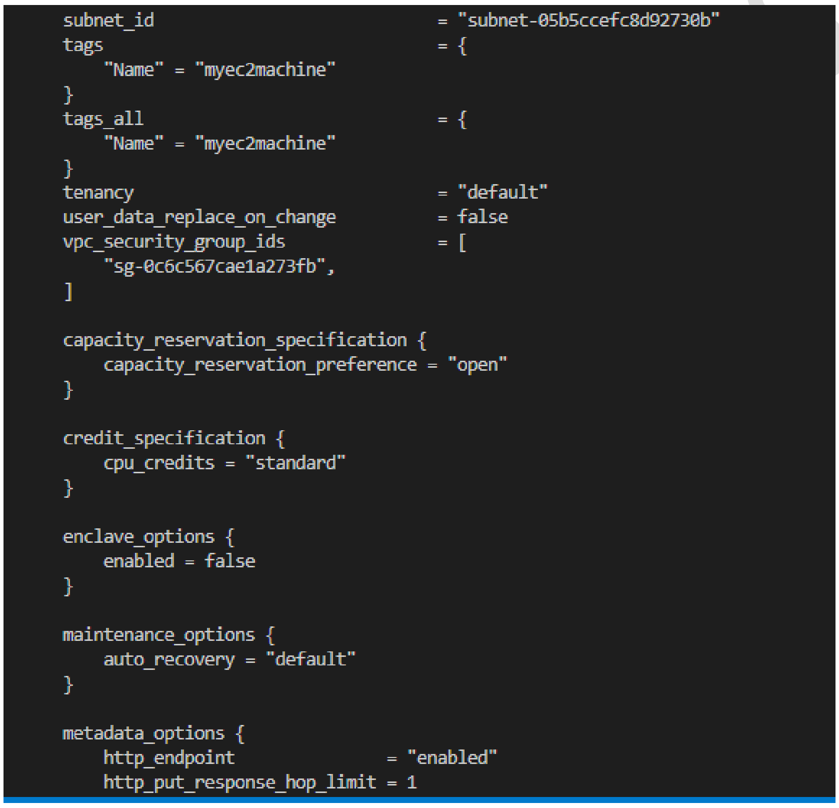
```
        }
    }

  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_instance.myec2machine: Creating...
  aws_instance.myec2machine: Still creating... [10s elapsed]
  aws_instance.myec2machine: Still creating... [20s elapsed]
  aws_instance.myec2machine: Creation complete after 21s [id=i-0f6f0ca758817fd80]

  Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
  PS [redacted]\ec2instance>
```

6) terraform show: The terraform show command is used to provide human-readable output from a state or plan file. This can be used to inspect a plan to ensure that the planned operations are expected, or to inspect the current state as Terraform sees it.

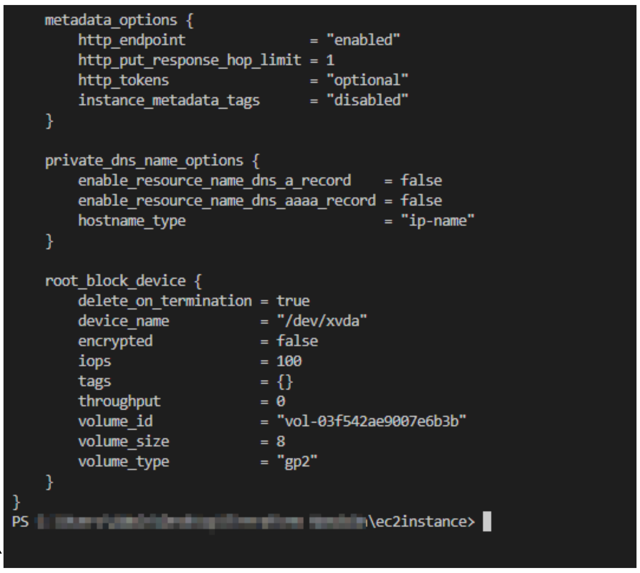
Machine-readable output is generated by adding the `-json` command-line flag.



```
PS [redacted]\ec2instance> terraform show
# aws_instance.myec2machine:
resource "aws_instance" "myec2machine" {
    ami                                  = "ami-074dc0a6f6c764218"
    arn                                  = "arn:aws:ec2:ap-south-1:804977405016:instance/i-0f6f0ca758817fd80"
    associate_public_ip_address          = true
    availability_zone                    = "ap-south-1b"
    cpu_core_count                       = 1
    cpu_threads_per_core                 = 1
    disable_api_stop                     = false
    disable_api_termination              = false
    ebs_optimized                        = false
    get_password_data                    = false
    hibernation                          = false
    id                                   = "i-0f6f0ca758817fd80"
    instance_initiated_shutdown_behavior = "stop"
    instance_state                       = "running"
    instance_type                        = "t2.micro"
    ipv6_address_count                   = 0
    ipv6_addresses                       = []
    monitoring                           = false
    primary_network_interface_id         = "eni-0797e66d62dad9912"
    private_dns                          = "ip-172-31-11-3.ap-south-1.compute.internal"
    private_ip                           = "172.31.11.3"
    public_dns                           = "ec2-43-204-217-174.ap-south-1.compute.amazonaws.com"
    public_ip                            = "43.204.217.174"
    secondary_private_ips                = []
    security_groups                      = [
        "default",
    ]
    source_dest_check                    = true
    subnet_id                            = "subnet-05b5ccefc8d92730b"
```



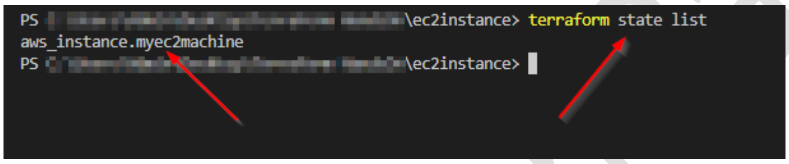
```
    subnet_id                            = "subnet-05b5ccefc8d92730b"
    tags                                 = {
        "Name" = "myec2machine"
    }
    tags_all                             = {
        "Name" = "myec2machine"
    }
    tenancy                              = "default"
    user_data_replace_on_change          = false
    vpc_security_group_ids               = [
        "sg-0c6c567cae1a273fb",
    ]

    capacity_reservation_specification {
        capacity_reservation_preference = "open"
    }

    credit_specification {
        cpu_credits = "standard"
    }

    enclave_options {
        enabled = false
    }

    maintenance_options {
        auto_recovery = "default"
    }

    metadata_options {
        http_endpoint               = "enabled"
        http_put_response_hop_limit = 1
```



```
    metadata_options {
        http_endpoint               = "enabled"
        http_put_response_hop_limit = 1
        http_tokens                 = "optional"
        instance_metadata_tags      = "disabled"
    }

    private_dns_name_options {
        enable_resource_name_dns_a_record    = false
        enable_resource_name_dns_aaaa_record = false
        hostname_type                        = "ip-name"
    }

    root_block_device {
        delete_on_termination = true
        device_name           = "/dev/xvda"
        encrypted             = false
        iops                  = 100
        tags                  = {}
        throughput            = 0
        volume_id             = "vol-03f542ae9007e6b3b"
        volume_size           = 8
        volume_type           = "gp2"
    }
}
PS [redacted]\ec2instance>
```

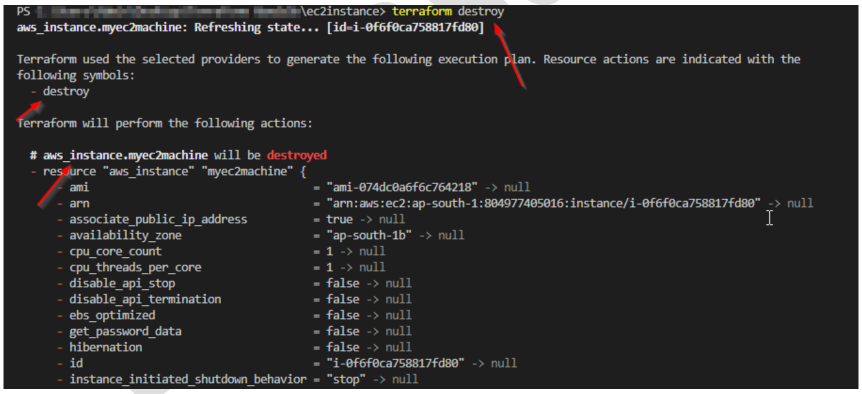
`

7) terraform state list: The terraform state list command will list all resources in the state file. The resources listed are sorted according to module depth order followed by alphabetical. This means that resources that are in your immediate configuration are listed first, and resources that are more deeply nested within modules are listed last.

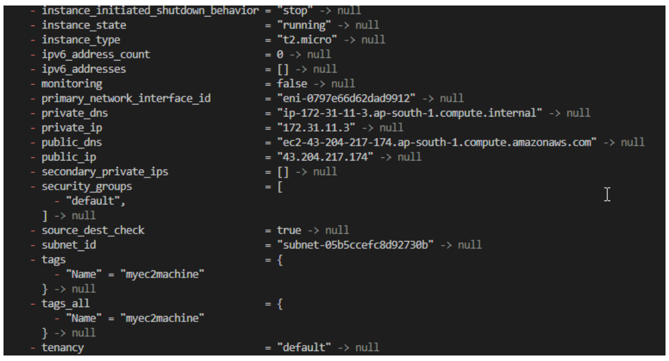


```
PS [redacted]\ec2instance> terraform state list
aws_instance.myec2machine
PS [redacted]\ec2instance>
```

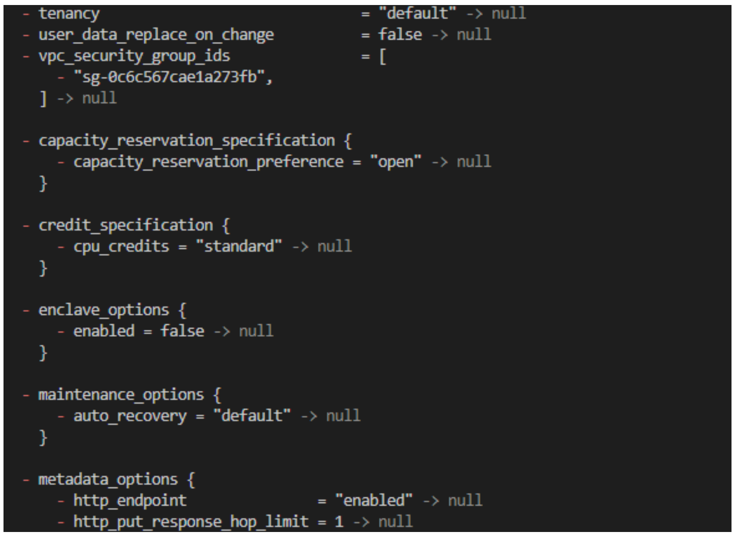
8) terraform destroy: The terraform destroy command is a convenient way to destroy all remote objects managed by a particular Terraform configuration.



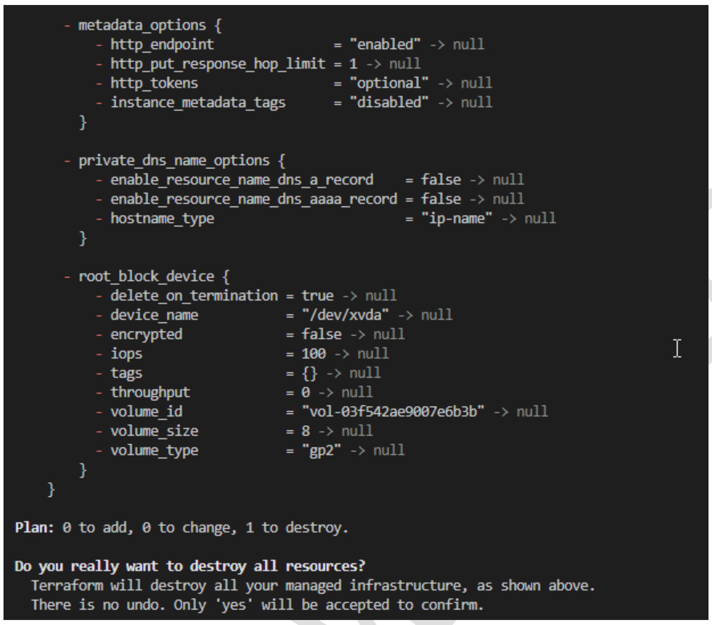
```
PS [redacted]\ec2instance> terraform destroy
aws_instance.myec2machine: Refreshing state... [id=i-0f6f0ca758817fd80]

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

Terraform will perform the following actions:

  # aws_instance.myec2machine will be destroyed
  - resource "aws_instance" "myec2machine" {
      - ami                                  = "ami-074dc0a6f6c764218" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:804977405016:instance/i-0f6f0ca758817fd80" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-0f6f0ca758817fd80" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
```



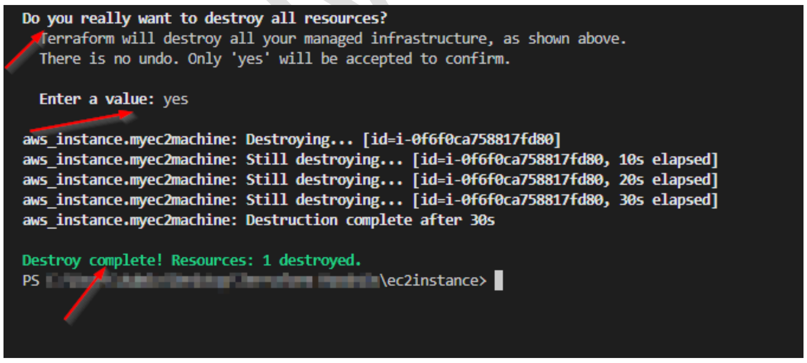
```
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - primary_network_interface_id         = "eni-0797e66d62dad9912" -> null
      - private_dns                          = "ip-172-31-11-3.ap-south-1.compute.internal" -> null
      - private_ip                           = "172.31.11.3" -> null
      - public_dns                           = "ec2-43-204-217-174.ap-south-1.compute.amazonaws.com" -> null
      - public_ip                            = "43.204.217.174" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
          - "default",
        ] -> null
      - source_dest_check                    = true -> null
      - subnet_id                            = "subnet-05b5ccefc8d92730b" -> null
      - tags                                 = {
          - "Name" = "myec2machine"
        } -> null
      - tags_all                             = {
          - "Name" = "myec2machine"
        } -> null
      - tenancy                              = "default" -> null
```



```
  - tenancy                              = "default" -> null
  - user_data_replace_on_change          = false -> null
  - vpc_security_group_ids               = [
      - "sg-0c6c567cae1a273fb",
    ] -> null

  - capacity_reservation_specification {
      - capacity_reservation_preference = "open" -> null
    }

  - credit_specification {
      - cpu_credits = "standard" -> null
    }

  - enclave_options {
      - enabled = false -> null
    }

  - maintenance_options {
      - auto_recovery = "default" -> null
    }

  - metadata_options {
      - http_endpoint               = "enabled" -> null
      - http_put_response_hop_limit = 1 -> null
```



```
      - metadata_options {
          - http_endpoint               = "enabled" -> null
          - http_put_response_hop_limit = 1 -> null
          - http_tokens                 = "optional" -> null
          - instance_metadata_tags      = "disabled" -> null
        }

      - private_dns_name_options {
          - enable_resource_name_dns_a_record    = false -> null
          - enable_resource_name_dns_aaaa_record = false -> null
          - hostname_type                        = "ip-name" -> null
        }

      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-03f542ae9007e6b3b" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

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



```
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_instance.myec2machine: Destroying... [id=i-0f6f0ca758817fd80]
aws_instance.myec2machine: Still destroying... [id=i-0f6f0ca758817fd80, 10s elapsed]
aws_instance.myec2machine: Still destroying... [id=i-0f6f0ca758817fd80, 20s elapsed]
aws_instance.myec2machine: Still destroying... [id=i-0f6f0ca758817fd80, 30s elapsed]
aws_instance.myec2machine: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
PS [redacted]\ec2instance>
```