Lesson 09 Demo 04

Implementing Local-Exec Provisioners

Objective: To implement local-exec provisioners for executing commands on the machine running Terraform during resource provisioning

Tools required: Ubuntu OS and AWS management console

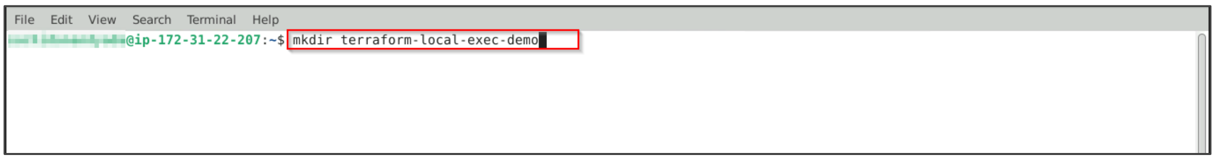
Prerequisites: None

Steps to be followed:

1. Implement local-exec provisioners
2. Create the local script
3. Initialize and verify the configuration

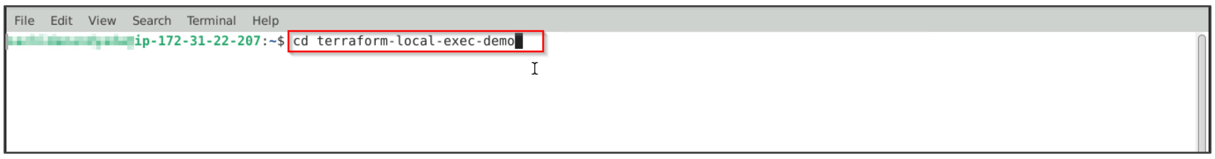
Step 1: Implement local-exec provisioners

1.1 Run the following command to create a directory:
mkdir terraform-local-exec-demo



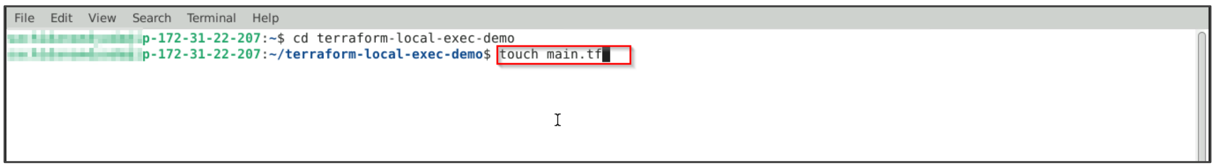
```
[redacted]@ip-172-31-22-207:~$ mkdir terraform-local-exec-demo
```

1.2 Navigate inside the directory using the following command:
cd terraform-local-exec-demo



```
[redacted]ip-172-31-22-207:~$ cd terraform-local-exec-demo
```

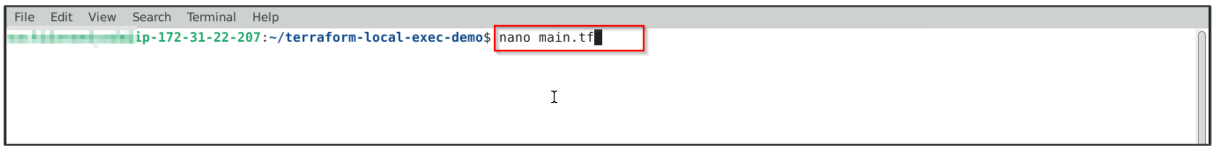
1.3 Create a terraform file using the following command:
touch main.tf



```
[redacted]p-172-31-22-207:~$ cd terraform-local-exec-demo
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ touch main.tf
```

1.4 Execute the following command to edit the **main.tf** file:

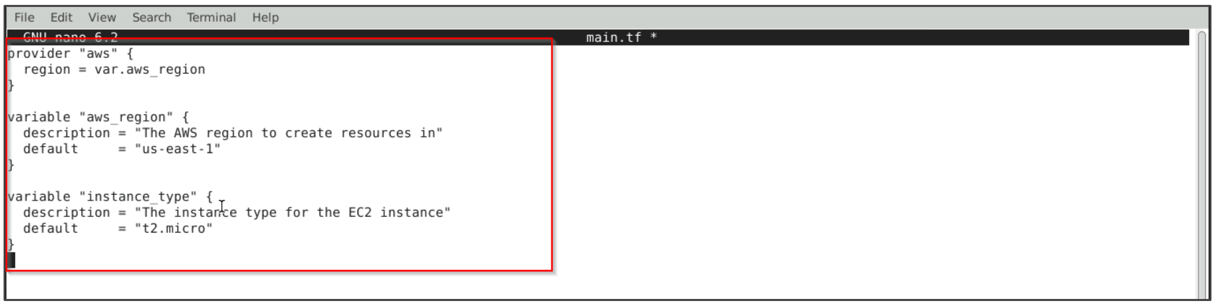
nano main.tf



```
[redacted]ip-172-31-22-207:~/terraform-local-exec-demo$ nano main.tf
```

1.5 Add the script inside the **main.tf** file to set the AWS provider region using a variable and define default values for the AWS region and EC2 instance type:

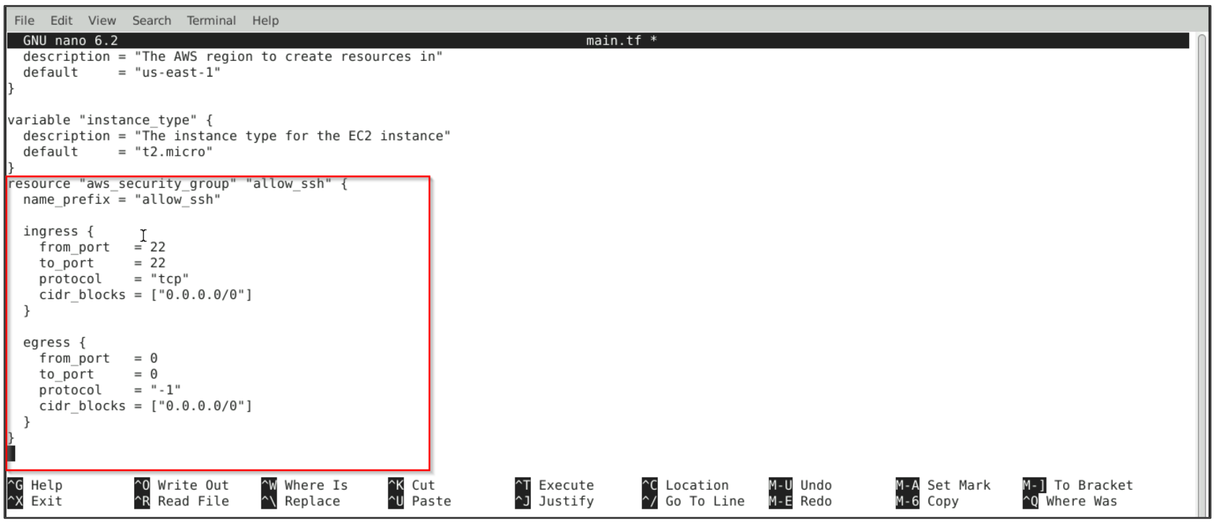
```
provider "aws" {
  region = var.aws_region
}

variable "aws_region" {
  description = "The AWS region to create resources in"
  default     = "us-east-1"
}

variable "instance_type" {
  description = "The instance type for the EC2 instance"
  default     = "t2.micro"
}
```



```
  GNU nano 6.2                                           main.tf *
provider "aws" {
  region = var.aws_region
}

variable "aws_region" {
  description = "The AWS region to create resources in"
  default     = "us-east-1"
}

variable "instance_type" {
  description = "The instance type for the EC2 instance"
  default     = "t2.micro"
}
```

1.6 Further, add the following code to the **tf** file to define an AWS security group that allows incoming SSH access on port 22 from any IP address and permits all outbound traffic:

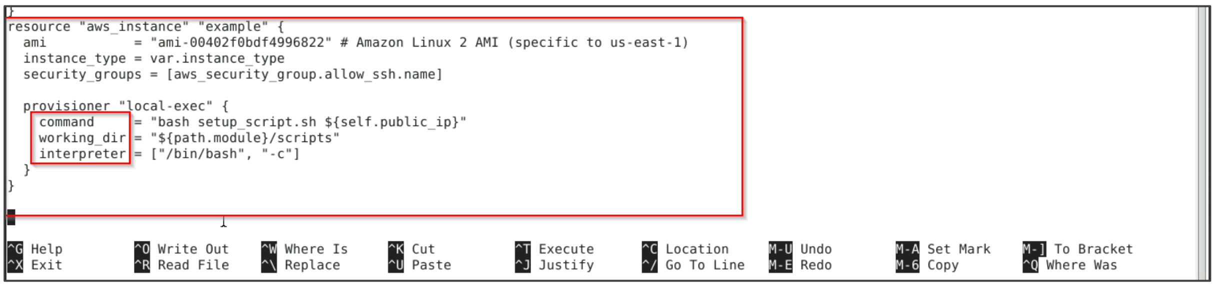
```
resource "aws_security_group" "allow_ssh" {
  name_prefix = "allow_ssh"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```



```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                                                     main.tf *
  description = "The AWS region to create resources in"
  default     = "us-east-1"
}

variable "instance_type" {
  description = "The instance type for the EC2 instance"
  default     = "t2.micro"
}
resource "aws_security_group" "allow_ssh" {
  name_prefix = "allow_ssh"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

1.7 Append the given script to the **main.tf** file:

```
resource "aws_instance" "example" {
  ami           = "ami-00402f0bfd4996822" # Amazon Linux 2 AMI (specific to us-east-
1)
  instance_type = var.instance_type
  security_groups = [aws_security_group.allow_ssh.name]

  provisioner "local-exec" {
    command     = "bash setup_script.sh ${self.public_ip}"
    working_dir = "${path.module}/scripts"
    interpreter = ["/bin/bash", "-c"]
  }
}
```



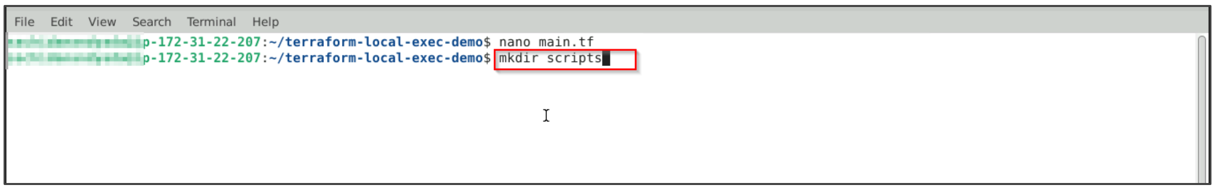
```
}
resource "aws_instance" "example" {
  ami           = "ami-00402f0bdf4996822" # Amazon Linux 2 AMI (specific to us-east-1)
  instance_type = var.instance_type
  security_groups = [aws_security_group.allow_ssh.name]

  provisioner "local-exec" {
    command     = "bash setup_script.sh ${self.public_ip}"
    working_dir = "${path.module}/scripts"
    interpreter = ["/bin/bash", "-c"]
  }
}

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

By adding this script, the Terraform code tells your local computer to run a script called **setup_script.sh**. This script is stored in a folder named scripts on your local computer.

Note: Click on **ctrl + x** and **y** to save the file and **enter** to exit back to the terminal

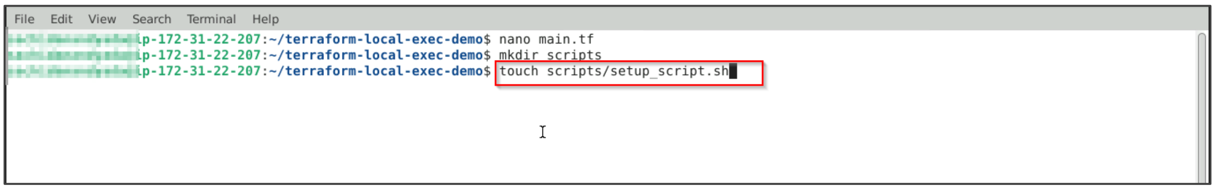
Step 2: Create the local script

2.1 Run the command given to create the scripts directory:
mkdir scripts



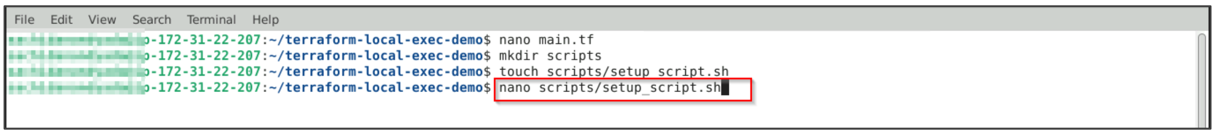
```
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ nano main.tf
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ mkdir scripts
```

2.2 Run the command given to create the script file inside the **scripts** directory:
touch scripts/setup_script.sh



```
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ nano main.tf
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ mkdir scripts
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ touch scripts/setup_script.sh
```

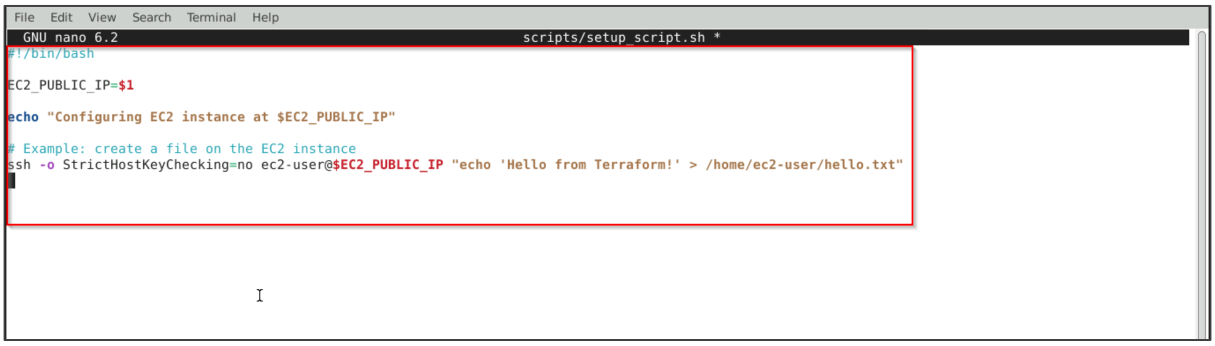
2.3 Run the command given to open the **setup_script.sh** file in a text editor:
nano scripts/setup_script.sh



```
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ nano main.tf
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ mkdir scripts
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ touch scripts/setup_script.sh
[redacted]p-172-31-22-207:~/terraform-local-exec-demo$ nano scripts/setup_script.sh
```

2.4 Enter the following bash script into the **setup_script.sh** file to configure an EC2 instance using the provided public IP address as an argument:

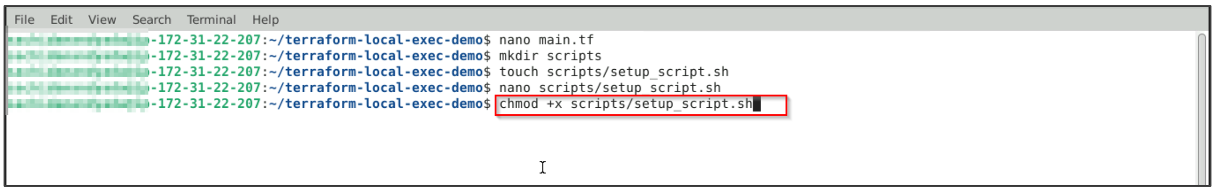
```
#!/bin/bash

EC2_PUBLIC_IP=$1

echo "Configuring EC2 instance at $EC2_PUBLIC_IP"

# Example: create a file on the EC2 instance
ssh -o StrictHostKeyChecking=no -i ~/.ssh/your-private-key.pem ec2-
user@$EC2_PUBLIC_IP "echo 'Hello from Terraform!' > /home/ec2-user/hello.txt"
```


File Edit View Search Terminal Help
GNU nano 6.2
scripts/setup_script.sh *

```
#!/bin/bash

EC2_PUBLIC_IP=$1

echo "Configuring EC2 instance at $EC2_PUBLIC_IP"

# Example: create a file on the EC2 instance
ssh -o StrictHostKeyChecking=no ec2-user@$EC2_PUBLIC_IP "echo 'Hello from Terraform!' > /home/ec2-user/hello.txt"
```

Note: Click on **ctrl + x** and **y** to save the file, and enter to exit back to the terminal

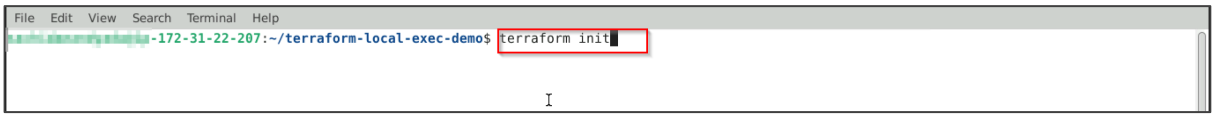
2.5 Make the script executable by running the following command:
chmod +x scripts/setup_script.sh



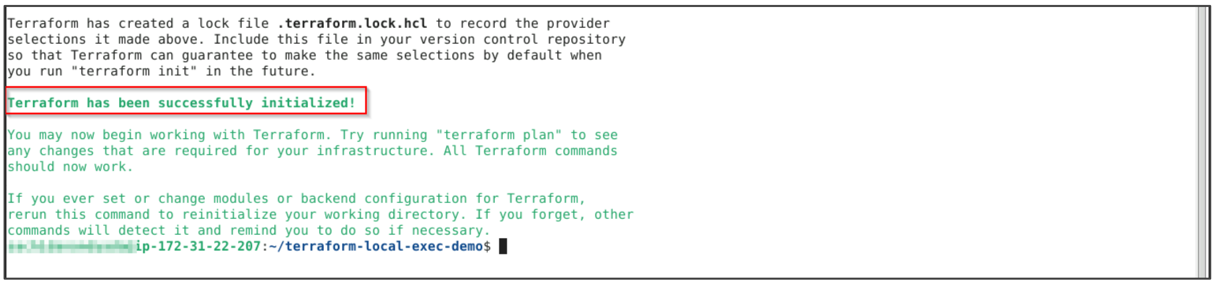
```
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ nano main.tf
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ mkdir scripts
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ touch scripts/setup_script.sh
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ nano scripts/setup_script.sh
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ chmod +x scripts/setup_script.sh
```

Step 3: Initialize and verify the configuration

3.1 Open the terminal and enter the command given below to initialize the Terraform configuration file:
terraform init



```
[redacted]-172-31-22-207:~/terraform-local-exec-demo$ terraform init
```

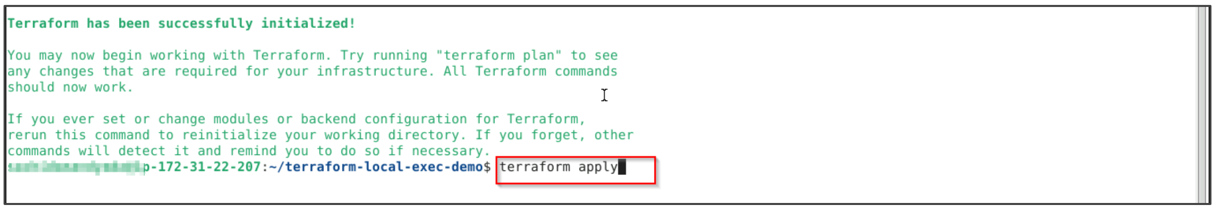


```
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
[redacted]ip-172-31-22-207:~/terraform-local-exec-demo$
```

The Terraform file is successfully initialized.

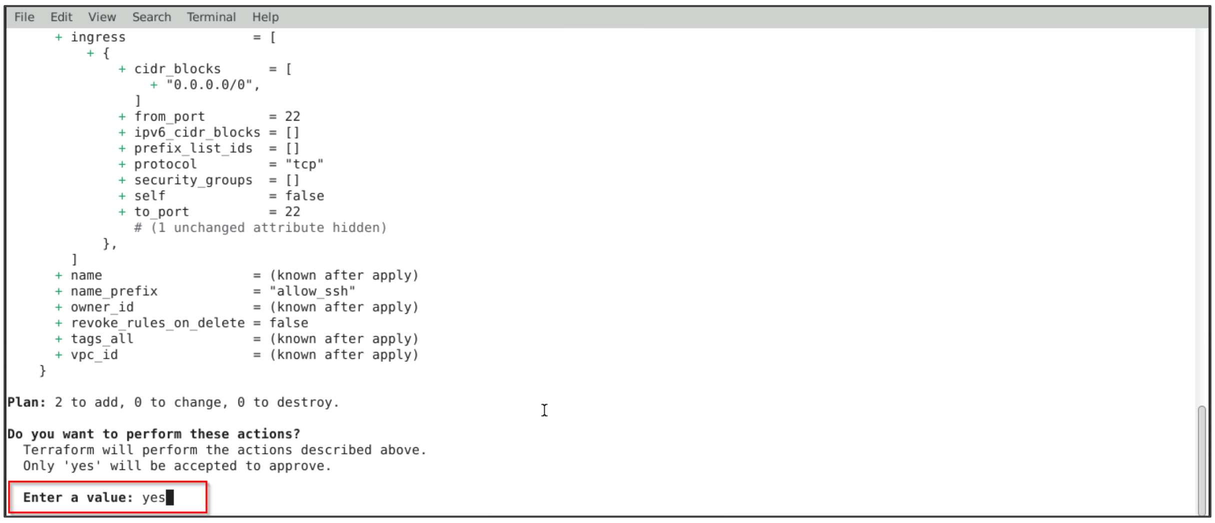
3.2 Execute the given command to apply the changes:

terraform apply

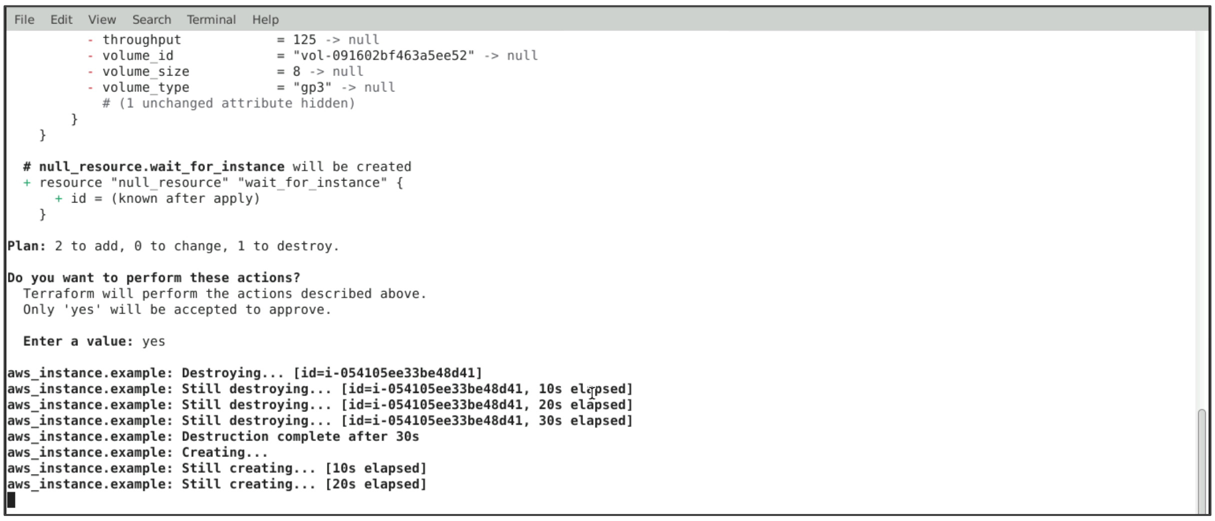


```
Terraform has been successfully initialized!

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]p-172-31-22-207:~/terraform-local-exec-demo$ terraform apply
```

3.3 Enter **yes** to confirm the apply command:



```
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
                # (1 unchanged attribute hidden)
            },
        ]
      + name                   = (known after apply)
      + name_prefix            = "allow_ssh"
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags_all               = (known after apply)
      + vpc_id                 = (known after apply)
    }

Plan: 2 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
```

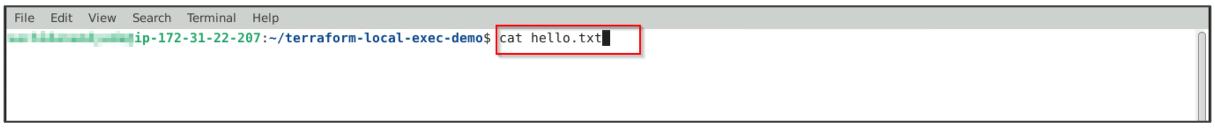


```
            - throughput           = 125 -> null
            - volume_id            = "vol-091602bf463a5ee52" -> null
            - volume_size          = 8 -> null
            - volume_type          = "gp3" -> null
              # (1 unchanged attribute hidden)
        }
    }

  # null_resource.wait_for_instance will be created
  + resource "null_resource" "wait_for_instance" {
      + id = (known after apply)
    }

Plan: 2 to add, 0 to change, 1 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.example: Destroying... [id=i-054105ee33be48d41]
aws_instance.example: Still destroying... [id=i-054105ee33be48d41, 10s elapsed]
aws_instance.example: Still destroying... [id=i-054105ee33be48d41, 20s elapsed]
aws_instance.example: Still destroying... [id=i-054105ee33be48d41, 30s elapsed]
aws_instance.example: Destruction complete after 30s
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
```

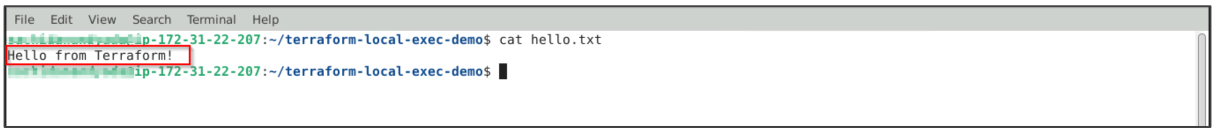
The apply command is completed successfully.

3.4 Run the given command to verify that the file was created:

cat hello.txt



```
[redacted]ip-172-31-22-207:~/terraform-local-exec-demo$ cat hello.txt
```



```
[redacted]ip-172-31-22-207:~/terraform-local-exec-demo$ cat hello.txt
Hello from Terraform!
[redacted]ip-172-31-22-207:~/terraform-local-exec-demo$
```

The creation of the **hello.txt** file is confirmed with the valid contents present inside it per the **command** and **working_dir** argument of the terraform local-exec provisioner.

By following these steps, you have successfully implemented local-exec provisioners for executing commands on the machine running Terraform during resource provisioning.