Lesson 09 Demo 05

Implementing Remote-Exec Provisioners

Objective: To implement the remote-exec provisioners in Terraform to automate the setup of an AWS EC2 instance for efficient and consistent deployment across multiple environments

Tools required: Linux terminal and AWS console management

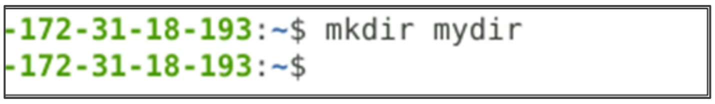
Prerequisites: None

Steps to be followed:

1. Implement remote-exec provisioners in Terraform
2. Deploy EC2 instances from AWS console

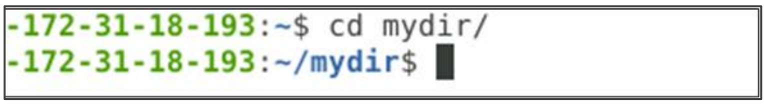
Step 1: Implement remote-exec provisioners in Terraform

1.1 Open the Linux terminal in your practice lab and then create a directory using the following command:
mkdir mydir



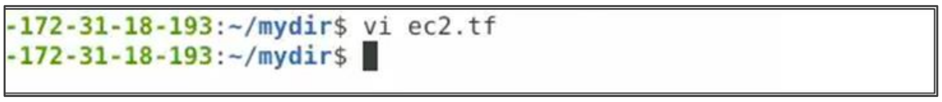
```
-172-31-18-193:~$ mkdir mydir
-172-31-18-193:~$
```

1.2 Navigate inside the directory using the following command:
cd mydir/



```
-172-31-18-193:~$ cd mydir/
-172-31-18-193:~/mydir$
```

1.3 Create a Terraform file using the following command:
vi ec2.tf



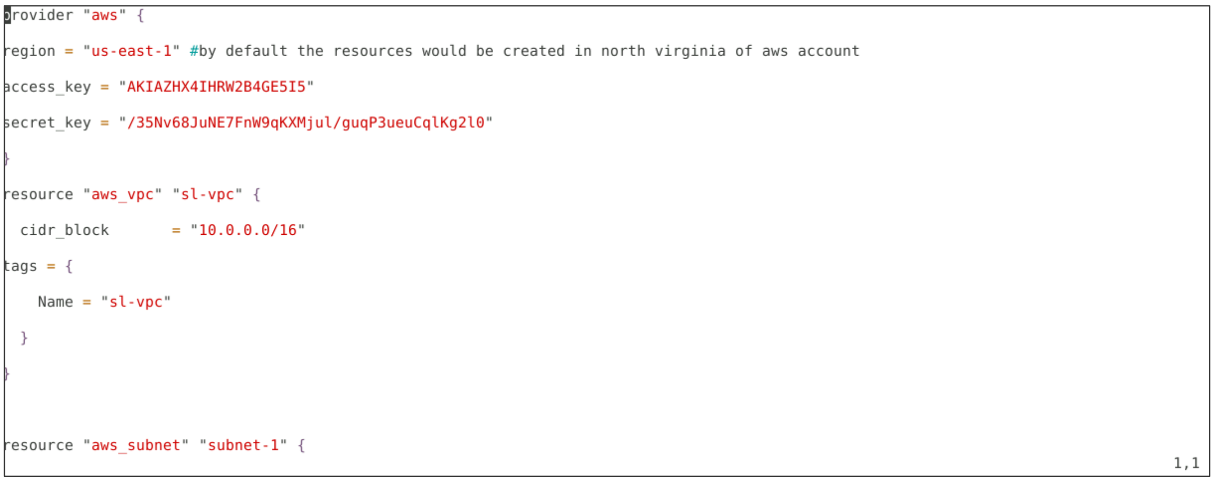
```
-172-31-18-193:~/mydir$ vi ec2.tf
-172-31-18-193:~/mydir$
```

1.4 Enter the following Terraform script inside the **ec2.tf** file to implement the remote-exec provisioners:

```
provider "aws" {

region = "us-east-1" #by default the resources would be created in north virginia of aws account

access_key = "###USE YOUR ACCESS KEYS"

secret_key = "## USE YOUR ACCESS KEYS "

}


resource "aws_vpc" "sl-vpc" {

  cidr_block      = "10.0.0.0/16"

tags = {

    Name = "sl-vpc"

  }

}


resource "aws_subnet" "subnet-1" {

  vpc_id     = aws_vpc.sl-vpc.id

  cidr_block = "10.0.1.0/24"

  map_public_ip_on_launch  = true

  depends_on = [aws_vpc.sl-vpc]

  tags = {

    Name = "sl-subnet"

  }

}


resource "aws_route_table" "sl-route-table" {

  vpc_id = aws_vpc.sl-vpc.id

  tags = {

    Name = "sl-route-table"
  }
}

resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.subnet-1.id
  route_table_id = aws_route_table.sl-route-table.id
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.sl-vpc.id
  depends_on = [aws_vpc.sl-vpc]
  tags = {
    Name = "sl-gw"
  }
}

resource "aws_route" "sl-route" {
  route_table_id = aws_route_table.sl-route-table.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id  = aws_internet_gateway.gw.id
}

resource "aws_security_group" "sl-sg" {
  name        = "allow_web_traffic"
  description = "Allow web inbound traffic"
  vpc_id      = aws_vpc.sl-vpc.id

  ingress {
    description = "TLS from VPC"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
 ingress {
    description = "HTTPS"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
 ingress {
    description = "SSH"
    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"]
  }

  tags = {
    Name = "sl-sg"
  }
}

resource "tls_private_key" "web-key" {
  algorithm   = "RSA"
```

```
}
resource "aws_key_pair" "app-key" {
  key_name   = "web-key"
  public_key = tls_private_key.web-key.public_key_openssh
}
resource "local_file" "web-key" {
  content  = tls_private_key.web-key.private_key_pem
  filename = "web-key.pem"
}
resource "aws_instance" "myec2" {
  ami = "ami-04823729c75214919"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.subnet-1.id
  key_name = "web-key"
  security_groups = [aws_security_group.sl-sg.id]
  tags = {
    Name = "Webserver"
 }
provisioner "remote-exec" {
   connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = tls_private_key.web-key.private_key_pem
    host     = self.public_ip
  }
    inline = [
      "sudo yum install httpd php -y",
      "sudo systemctl restart httpd",
      "sudo systemctl enable httpd"
    ]
  }
```

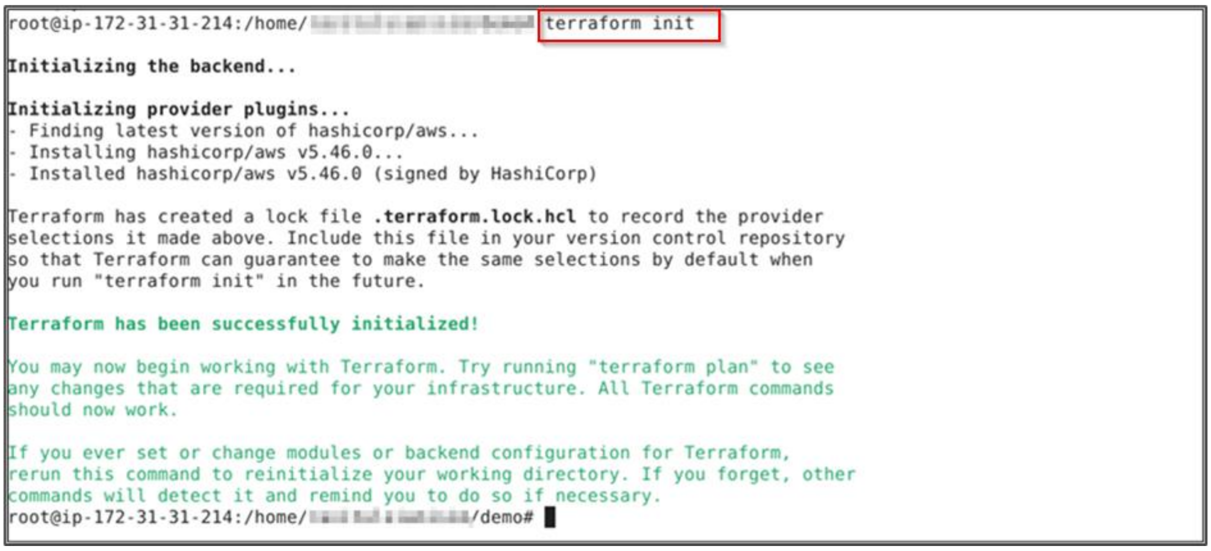
}



```
provider "aws" {

region = "us-east-1" #by default the resources would be created in north virginia of aws account

access_key = "AKIAZHX4IHRW2B4GE5I5"

secret_key = "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"

}

resource "aws_vpc" "sl-vpc" {

  cidr_block        = "10.0.0.0/16"

tags = {

    Name = "sl-vpc"

  }

}


resource "aws_subnet" "subnet-1" {
                                                                                                                  1,1
```

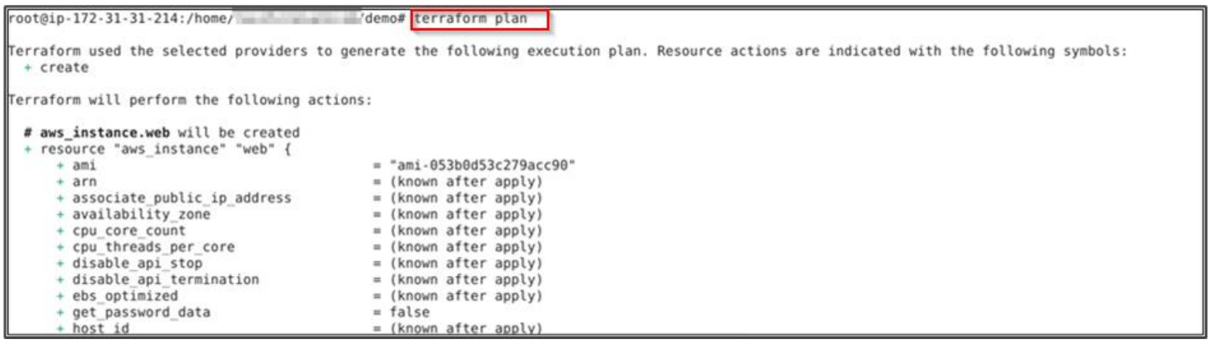
Note: Ensure you provide the **Access key** and **Secret access key** credentials of your AWS account in line 3 and line 4

1.5 Run the following command to initialize Terraform as shown in the screenshot below:
terraform init



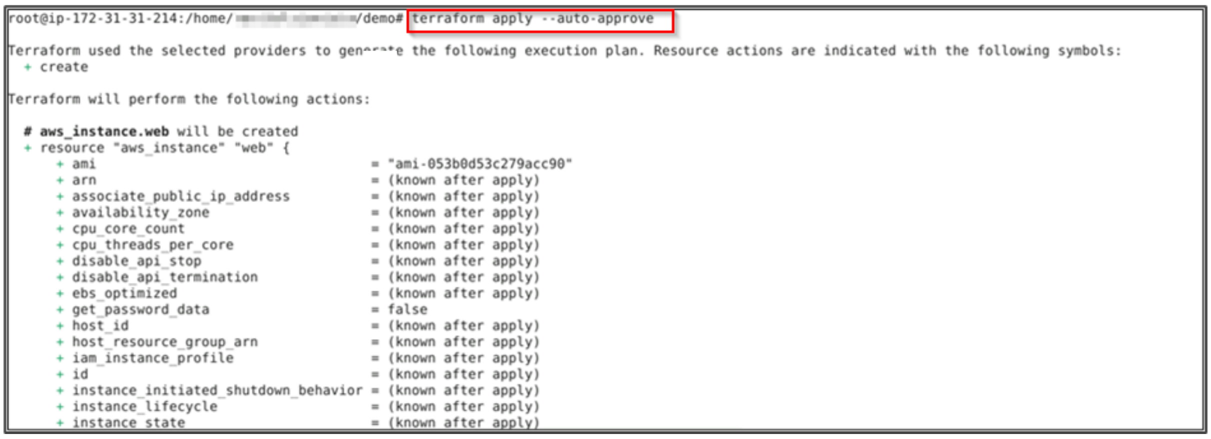
```
root@ip-172-31-31-214:/home/[redacted] terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.46.0...
- Installed hashicorp/aws v5.46.0 (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.
root@ip-172-31-31-214:/home/[redacted]/demo#
```

1.6 Run the following command to preview the infrastructure before applying the configurations:
terraform plan



```
root@ip-172-31-31-214:/home/[redacted]/demo# 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.web will be created
  + resource "aws_instance" "web" {
      + ami                                  = "ami-053b0d53c279acc90"
      + 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)
```

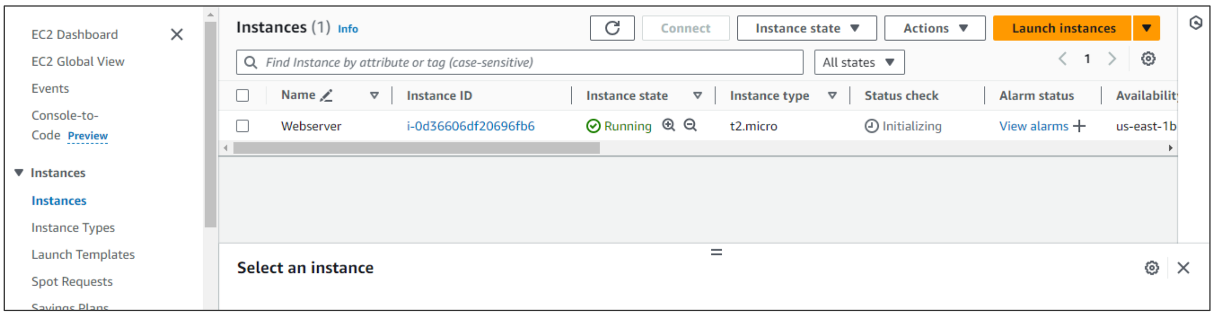
1.7 Run the following command to automatically apply the infrastructure:
terraform apply --auto-approve



```
root@ip-172-31-31-214:/home/[redacted]/demo# terraform apply --auto-approve

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.web will be created
  + resource "aws_instance" "web" {
      + ami                                  = "ami-053b0d53c279acc90"
      + 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_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
```

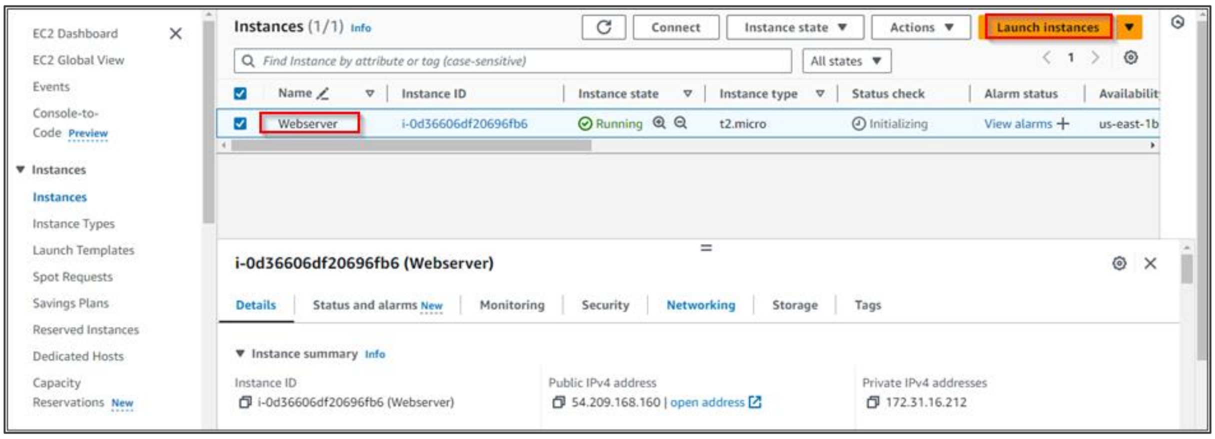
Step 2: Deploy EC2 instances from AWS console

2.1 Navigate into your AWS console to check for the deployment of an EC2 instance as shown in the screenshot below:



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabilit
Webserver	i-0d36606df20696fb6	Running	t2.micro	Initializing	View alarms	us-east-1b

2.2 Select the created instance and then click on the **Launch instances** button as shown in the screenshot below:

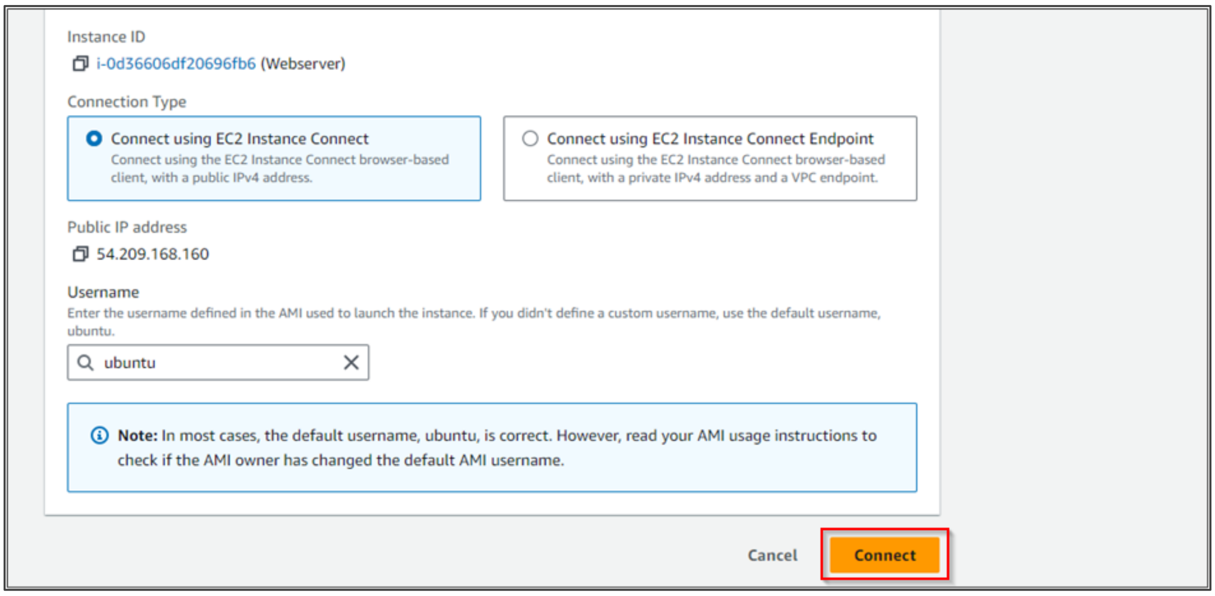


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabilit
Webserver	i-0d36606df20696fb6	Running	t2.micro	Initializing	View alarms +	us-east-1b

i-0d36606df20696fb6 (Webserver)

Instance ID: i-0d36606df20696fb6 (Webserver)
Public IPv4 address: 54.209.168.160 | open address
Private IPv4 addresses: 172.31.16.212

2.3 Scroll down and click the **Connect** button as shown in the screenshot below:


Instance ID
i-0d36606df20696fb6 (Webserver)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

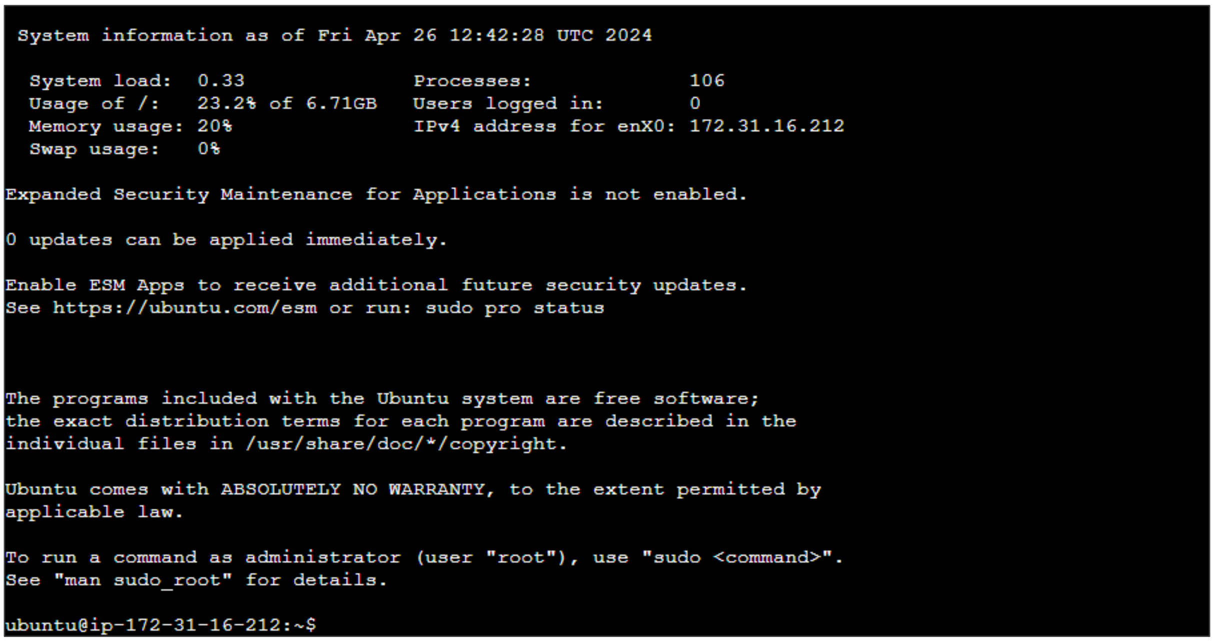
Public IP address
54.209.168.160

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

2.4 Confirm the successful connection of the instance by checking the landing page for any error messages:



```
 System information as of Fri Apr 26 12:42:28 UTC 2024

  System load:  0.33               Processes:             106
  Usage of /:   23.2% of 6.71GB   Users logged in:       0
  Memory usage: 20%                IPv4 address for enX0: 172.31.16.212
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status



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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-16-212:~$
```

By following these steps, you have successfully implemented the remote-exec provisioners in Terraform to automate the setup of an AWS EC2 instance for efficient and consistent deployment across multiple environments.