Connecting Terraform with AWS Cloud Platform: A Step-by-Step Guide

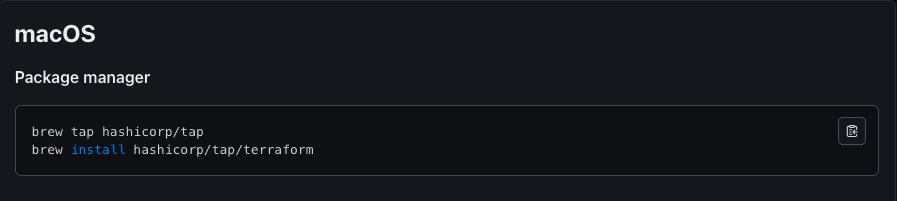
Introduction

Terraform is a powerful Infrastructure as Code (IaC) tool that allows you to manage and provision your cloud resources with a simple configuration file. In this article, we will walk you through the steps to connect Terraform with AWS Cloud Platform, from installing Terraform to deploying your first AWS resource.

---

Step 1: Install Terraform

First, you need to install Terraform on your local machine. You can download the latest version of Terraform from the [official website](https://www.terraform.io/downloads.html).



Command for installation (Linux/macOS):

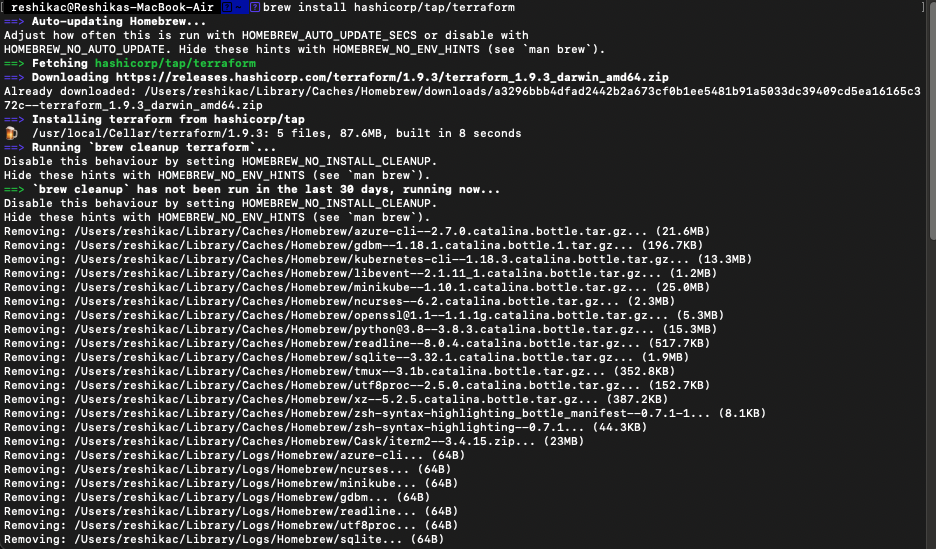
```sh

$ wget https://releases.hashicorp.com/terraform/1.4.0/terraform\_1.4.0\_linux\_amd64.zip

$ unzip terraform\_1.4.0\_linux\_amd64.zip

$ sudo mv terraform /usr/local/bin/

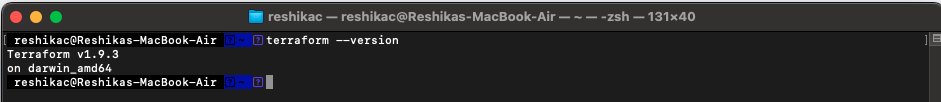
```



\*\*Image:\*\*

- Screenshot of the Terraform download page.

- Screenshot of the terminal with the installation commands being executed.



---

Step 2: Configure AWS CLI

Terraform uses AWS CLI to interact with AWS services. Make sure you have AWS CLI installed and configured with your AWS credentials.

\*\*Install AWS CLI:\*\*

```sh

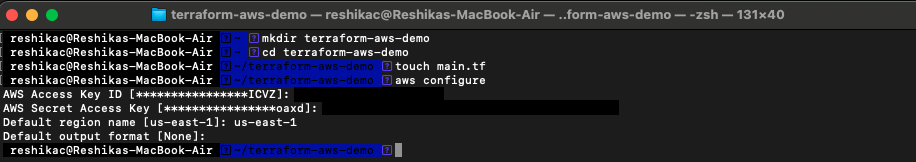
$ pip install awscli

```

\*\*Configure AWS CLI:\*\*

```sh

$ aws configure

```

You will be prompted to enter your AWS Access Key ID, Secret Access Key, region, and output format.

\*\*Image:\*\*

- Screenshot of the terminal showing the AWS CLI installation command.

- Screenshot of the terminal showing the AWS configure command with inputs.

---

Step 3: Create a New Terraform Configuration

Create a new directory for your Terraform configuration files. Inside this directory, create a file named `main.tf`.

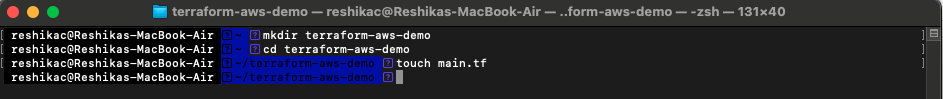
```sh

$ mkdir terraform-aws-demo

$ cd terraform-aws-demo

$ touch main.tf

```



\*\*Image:\*\*

- Screenshot of the terminal showing the directory creation and file creation commands.

---

Step 4: Write Terraform Configuration

In the `main.tf` file, write the Terraform configuration to create an AWS EC2 instance. Below is a sample configuration:

```hcl

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 4.16"

}

}

required\_version = ">= 1.2.0"

}

provider "aws" {

region = "us-west-2"

}

resource "aws\_instance" "app\_server" {

ami = "ami-830c94e3"

instance\_type = "t2.micro"

tags = {

Name = "ExampleAppServerInstance"

}

}

```



\*\*Image:\*\*

- Screenshot of a text editor with the sample `main.tf` content.

---

Step 5: Initialize Terraform

Run the `terraform init` command to initialize your Terraform configuration. This command downloads the necessary provider plugins.

```sh

$ terraform init

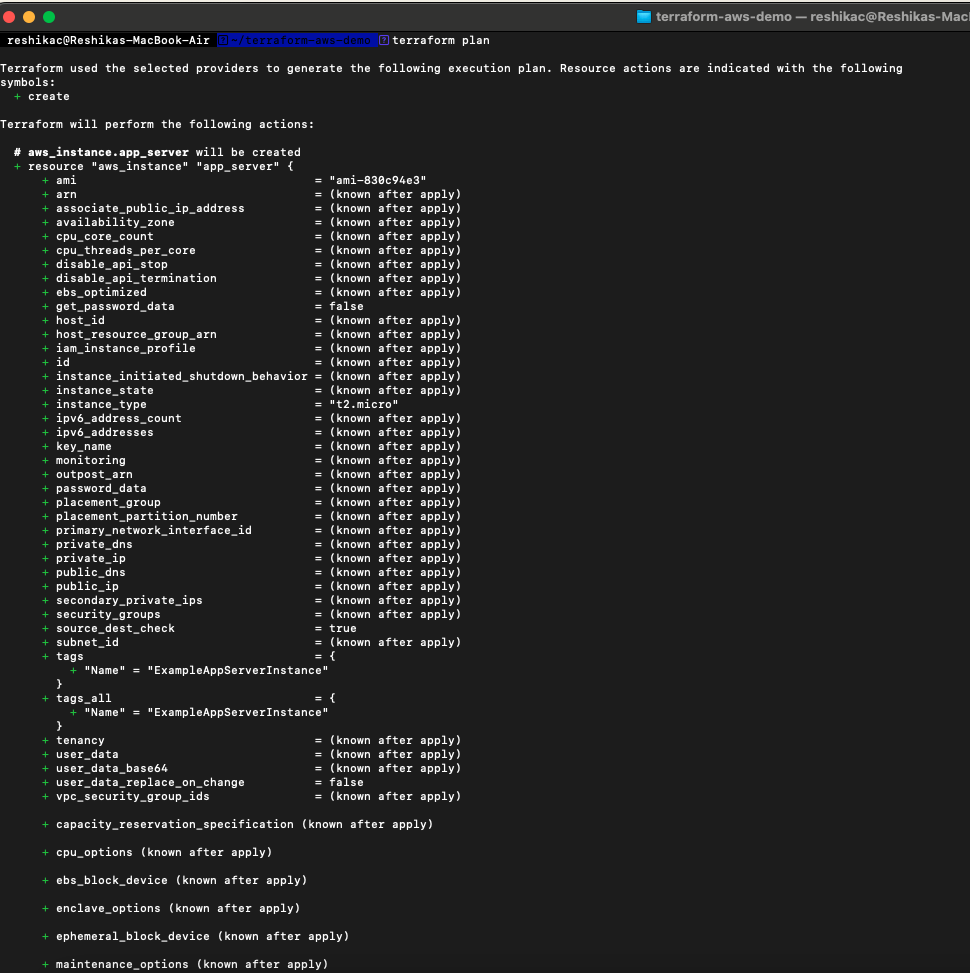
```

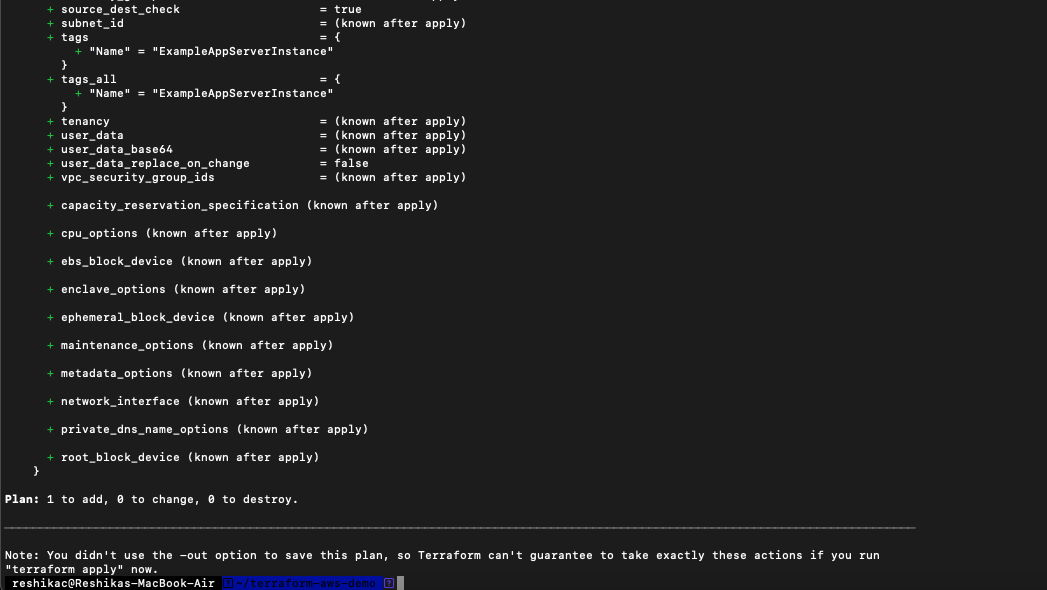


\*\*Image:\*\*

- Screenshot of the terminal showing the `terraform init` command and its output.

---





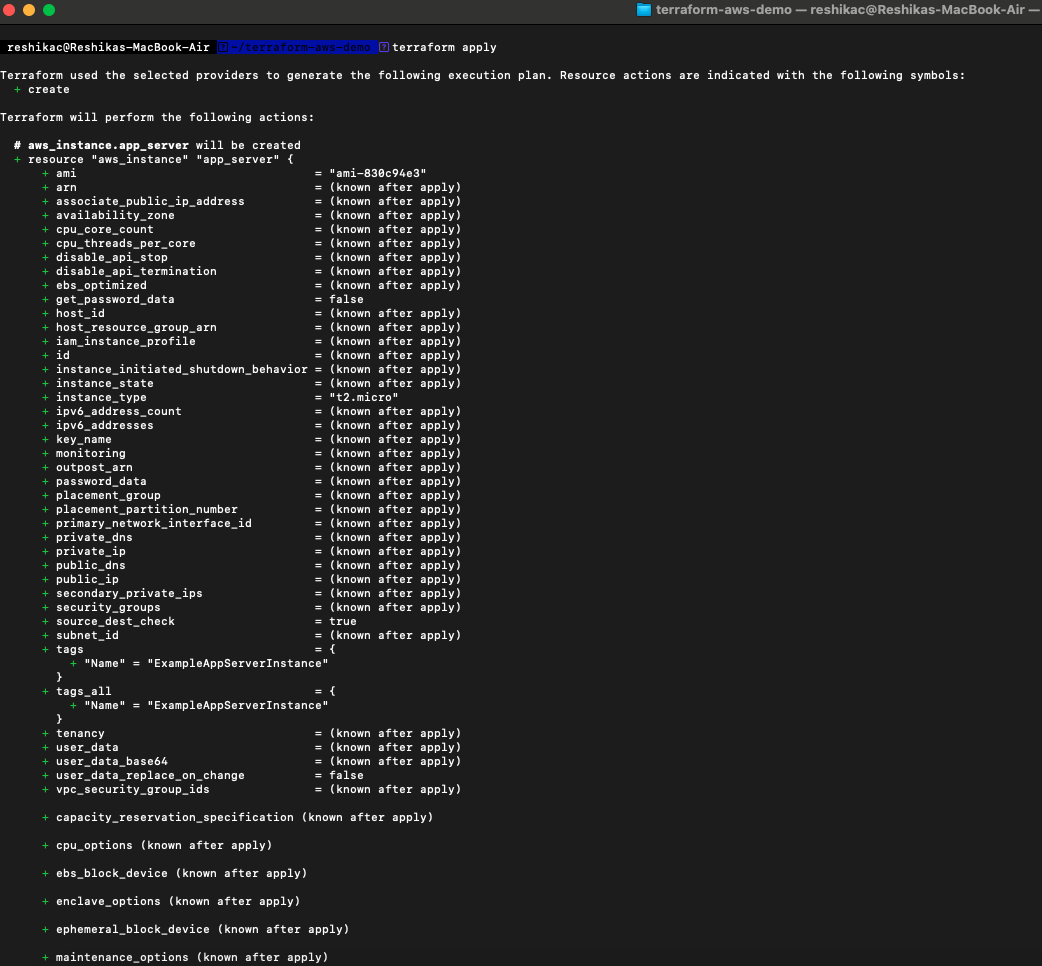
Step 6: Apply Terraform Configuration

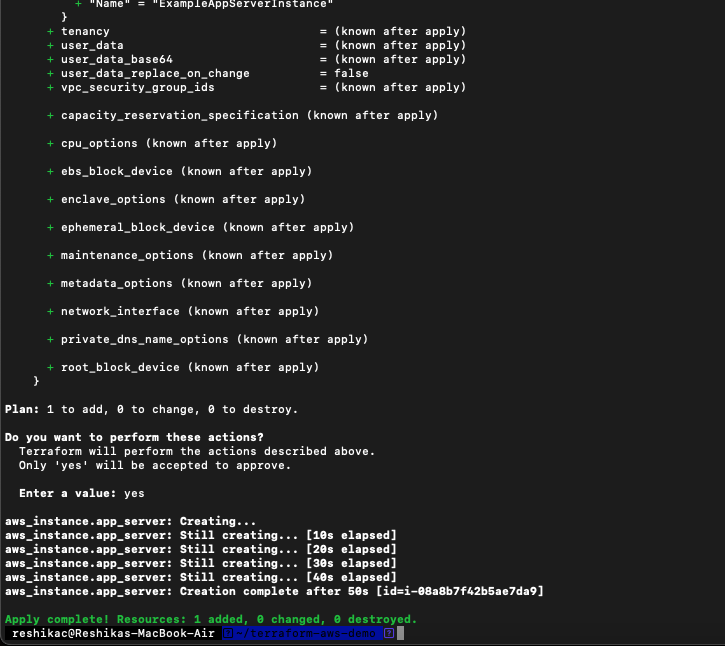
Run the `terraform apply` command to apply your configuration. Terraform will show you a plan of the actions it will take and ask for your confirmation to proceed.

```sh

$ terraform apply

```





Type `yes` when prompted to confirm the action.

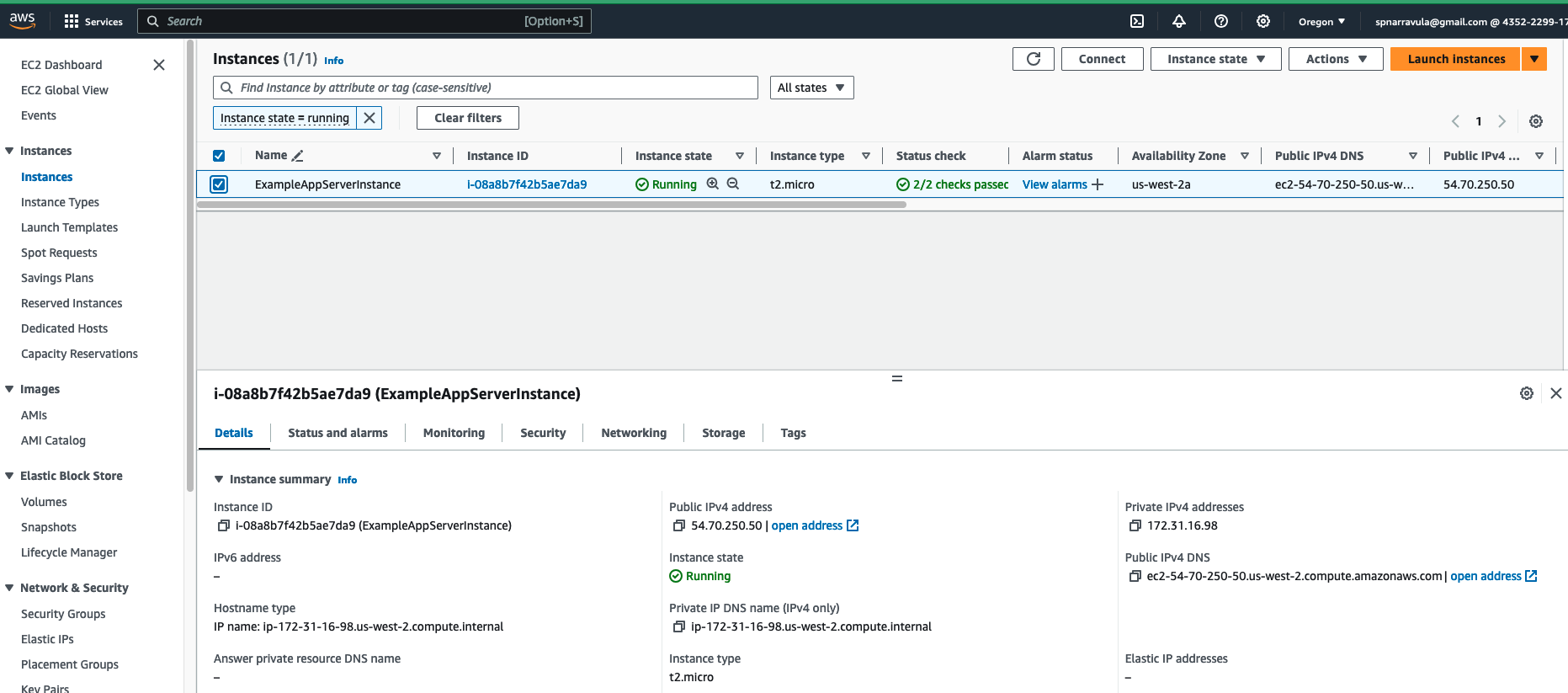
\*\*Image:\*\*

- Screenshot of the terminal showing the `terraform apply` command, the plan output, and the confirmation prompt.

---

Step 7: Verify the Deployment

After the `terraform apply` command completes, you can verify that the AWS EC2 instance has been created by logging into the AWS Management Console and navigating to the EC2 Dashboard.



\*\*Image:\*\*

- Screenshot of the AWS Management Console showing the newly created EC2 instance.

---

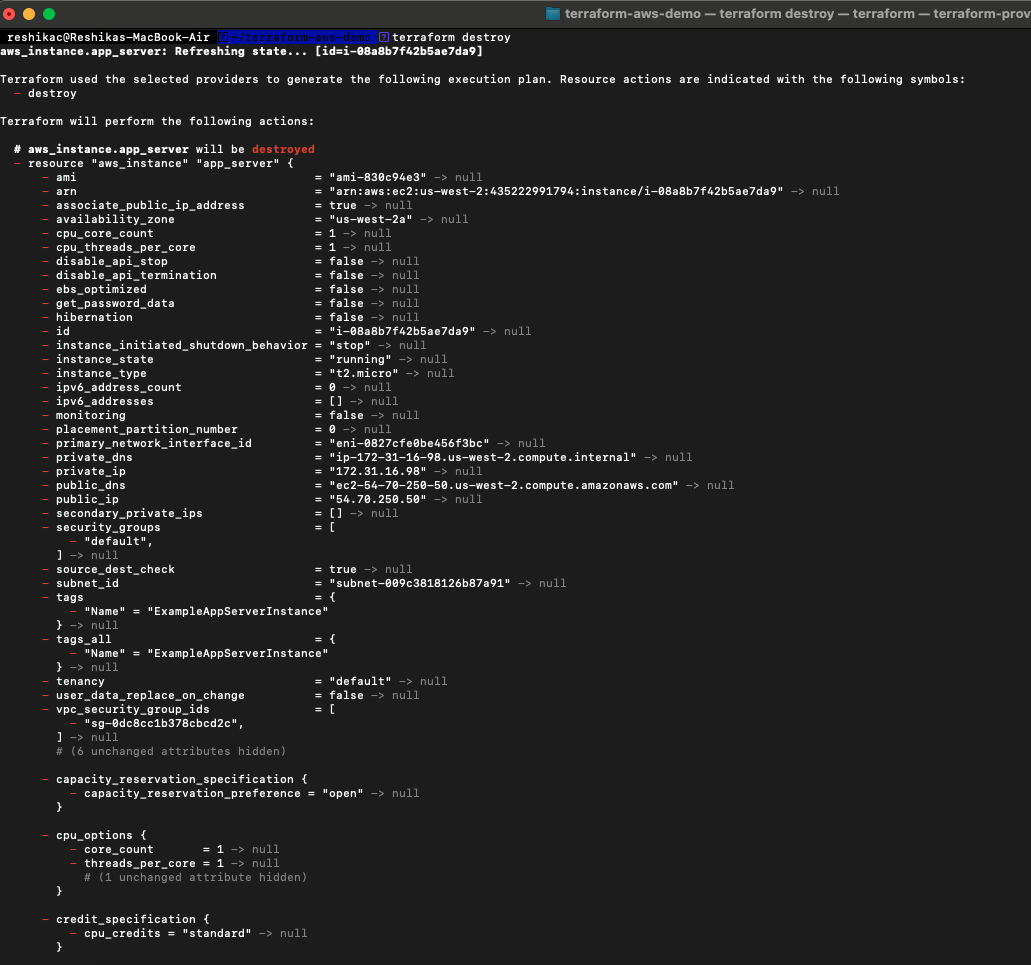
Step 8: Clean Up Resources

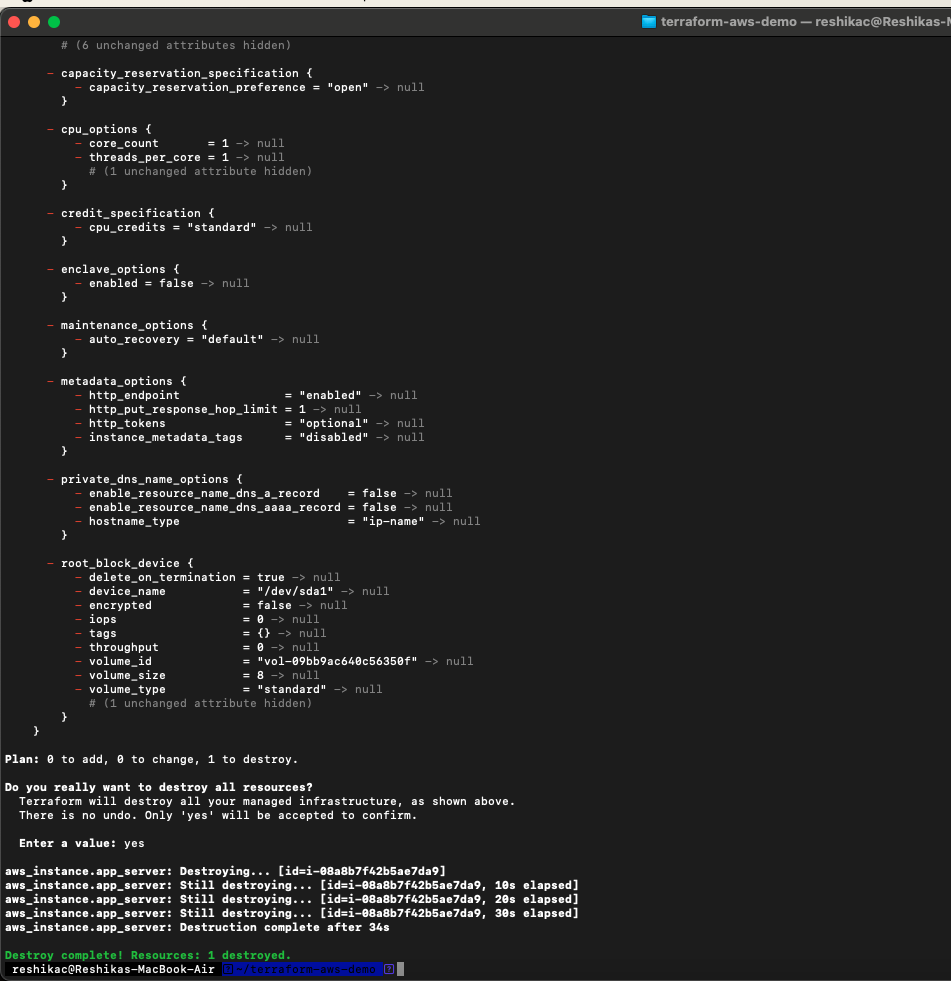
To avoid incurring charges, you can clean up the resources you created by running the `terraform destroy` command.

```sh

$ terraform destroy

```

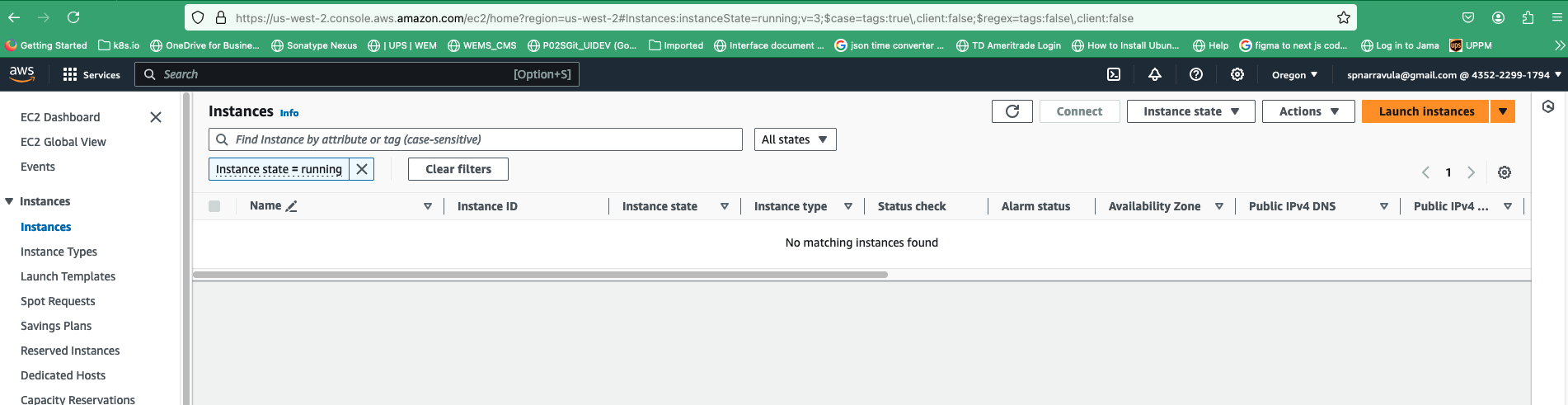




Type `yes` when prompted to confirm the destruction.

\*\*Image:\*\*

- Screenshot of the terminal showing the `terraform destroy` command and its output.



---

\*\*Conclusion\*\*

Congratulations! You have successfully connected Terraform with AWS Cloud Platform and deployed your first AWS resource using Infrastructure as Code. Terraform's declarative syntax makes it easy to manage your infrastructure efficiently and repeatably.

---