IaC using Terraform

- **Infrastructure as Code (IAC) using Terraform**: This means using code to manage and automate the setup and maintenance of your infrastructure (like servers, databases, networks) instead of doing it manually. With Terraform, you write configuration files that describe what your infrastructure should look like, and Terraform takes care of creating and updating those resources as needed.

- **Version Control and Reproducibility**: Terraform allows you to store your infrastructure configuration files in a version control system like Git. This way, you can track changes over time, collaborate with others, and easily reproduce your infrastructure setup whenever needed, ensuring consistency and reducing the risk of errors.

Terraform on AWS Demonstration

Prerequisites:

- An active free-tier AWS account.
- Installed `Terraform` and `AWS CLI` on Linux.

Step 1: Install Terraform

1. Open a terminal with `Ctrl + Alt + T`.
2. Google "install Terraform" and follow the instructions to reach the Linux-specific installation commands.
3. Run the following commands one by one:

```
josh@josh-VirtualBox:~$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
[sudo] password for josh:
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://apt.releases.hashicorp.com jammy InRelease
Hit:6 https://packages.microsoft.com/repos/code stable InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gnupg is already the newest version (2.2.27-3ubuntu2.1).
software-properties-common is already the newest version (0.99.22.9).
0 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
josh@josh-VirtualBox:~$
```

```
osh@josh-VirtualBox:~$ wget -O- https://apt.releases.hashicorp.com/gpg | \
pg --dearmor | \
udo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
-2024-05-19 10:38:58--  https://apt.releases.hashicorp.com/gpg
esolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.154.185.73, 18.154.185.31, 18.154.185.57, ...
onnecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.154.185.73|:443... connected.
TTP request sent, awaiting response... 200 OK
ength: 3980 (3.9K) [binary/octet-stream]
aving to: 'STDOUT'

                                    100%[===================================================================>]   3.89K  --.-KB/s    in 0s

024-05-19 10:38:59 (160 MB/s) - written to stdout [3980/3980]

osh@josh-VirtualBox:~$
```

```
josh@josh-VirtualBox:~$ gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```

```
josh@josh-VirtualBox:~$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

```
josh@josh-VirtualBox:~$ sudo apt update
```

```
josh@josh-VirtualBox:~$ sudo apt-get install terraform
```

4. Verify installation by running:

terraform -help

```
josh@josh-VirtualBox:~$ terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
```

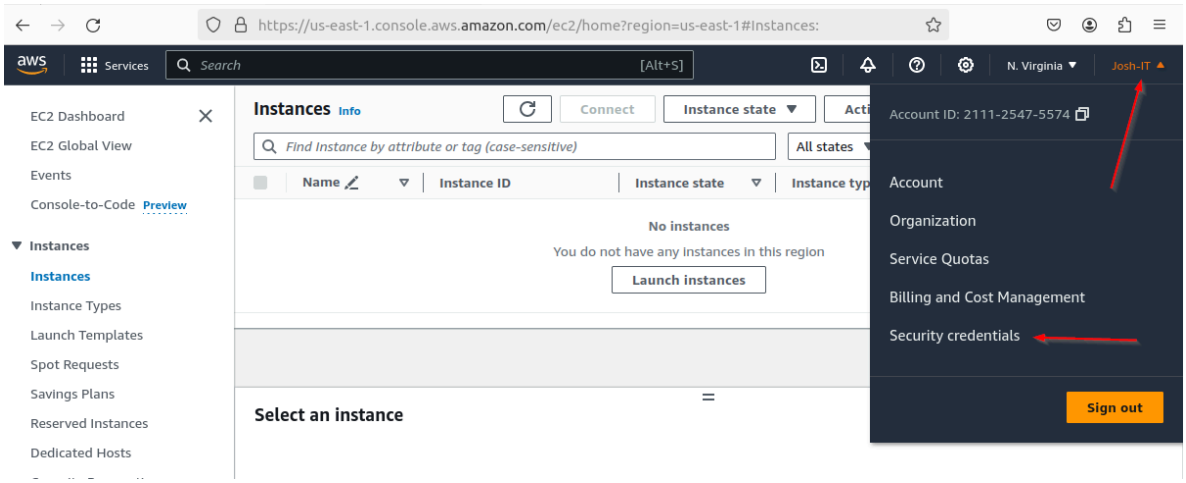
Step 2: Install AWS CLI

1. Google "install AWS CLI" and find the appropriate section for Linux x86.
2. Run the following commands one by one:

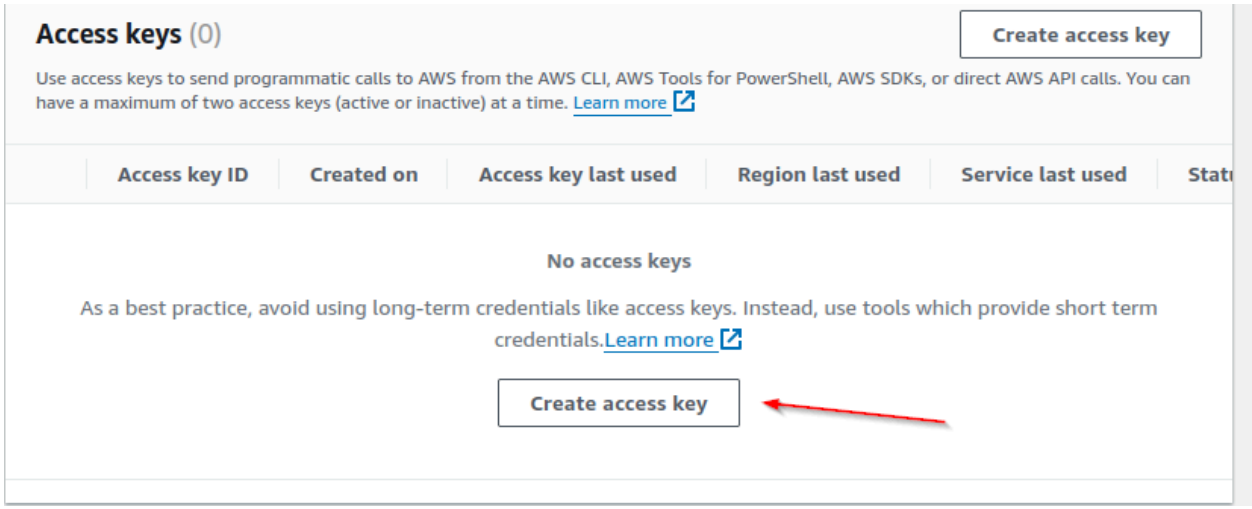
```
josh@josh-VirtualBox:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

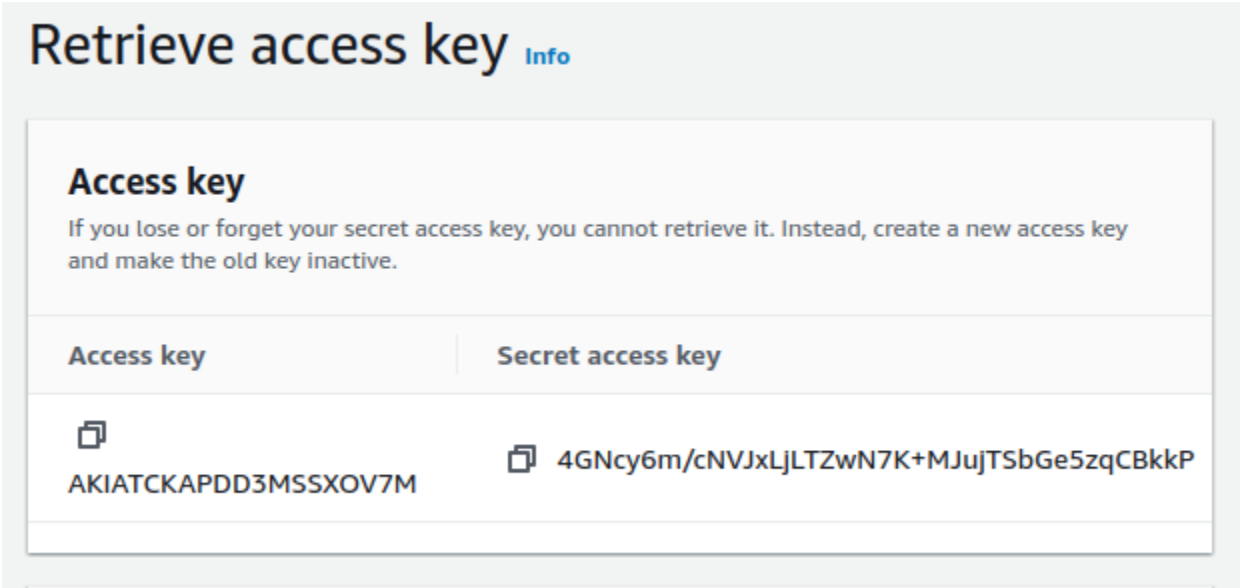
Step 3: Obtain AWS Access Keys

1. In the AWS Management Console, go to "Security Credentials" under your account name.



2. Create a new access key, download, and note both the Access Key ID and Secret Access Key.

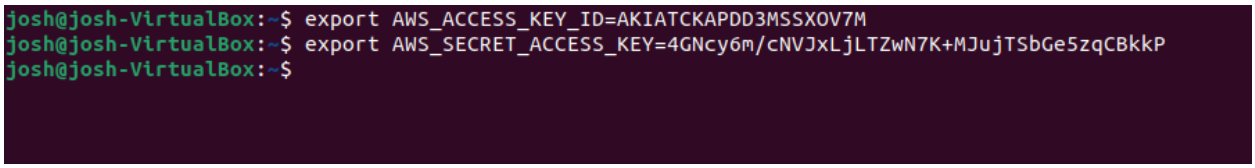




Step 4: Configure AWS CLI with Access Keys

1. Set up access keys using the following commands (replace `<ACCESS_KEY>` and `<SECRET_KEY>`):

   export AWS_ACCESS_KEY_ID="<ACCESS_KEY>"
   export AWS_SECRET_ACCESS_KEY="<SECRET_KEY>"



Step 5: Create the Terraform Configuration

1. Make a directory to hold Terraform files and navigate into it:

mkdir learn-terraform-aws
cd learn-terraform-aws

2. Create and edit a `main.tf` file with a basic Terraform configuration for an EC2 instance:

```
josh@josh-VirtualBox:~$ mkdir learn-terraform-aws-instance
josh@josh-VirtualBox:~$ cd learn-terraform-aws-instance
josh@josh-VirtualBox:~/learn-terraform-aws-instance$ touch main.tf
josh@josh-VirtualBox:~/learn-terraform-aws-instance$
```

nano main.tf

```
josh@josh-VirtualBox:~/learn-terraform-aws-instance$ nano main.tf
```

3. Paste the following code into `main.tf`:

```
terraform {
required_providers {
aws = {
source  = "hashicorp/aws"
version = "~> 4.16"
}
}
required_version = ">= 1.2"
}

provider "aws" {
region = "us-west-2"
}

resource "aws_instance" "app_server" {
ami           = "ami-"830c94e3”
instance_type = "t2.micro"
tags = {
Name = "ExampleAppServerInstance"
}
```

}

```
  GNU nano 6.2                                                    main.tf
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"
  }
}
```

4. Save and close the file (`Ctrl + X`, `Y`, `Enter`).

Step 6: Initialize and Apply Terraform

1. Initialize Terraform:

terraform init

```
josh@josh-VirtualBox:~/learn-terraform-aws-instance$ terraform init
```

2. Apply the Terraform plan:

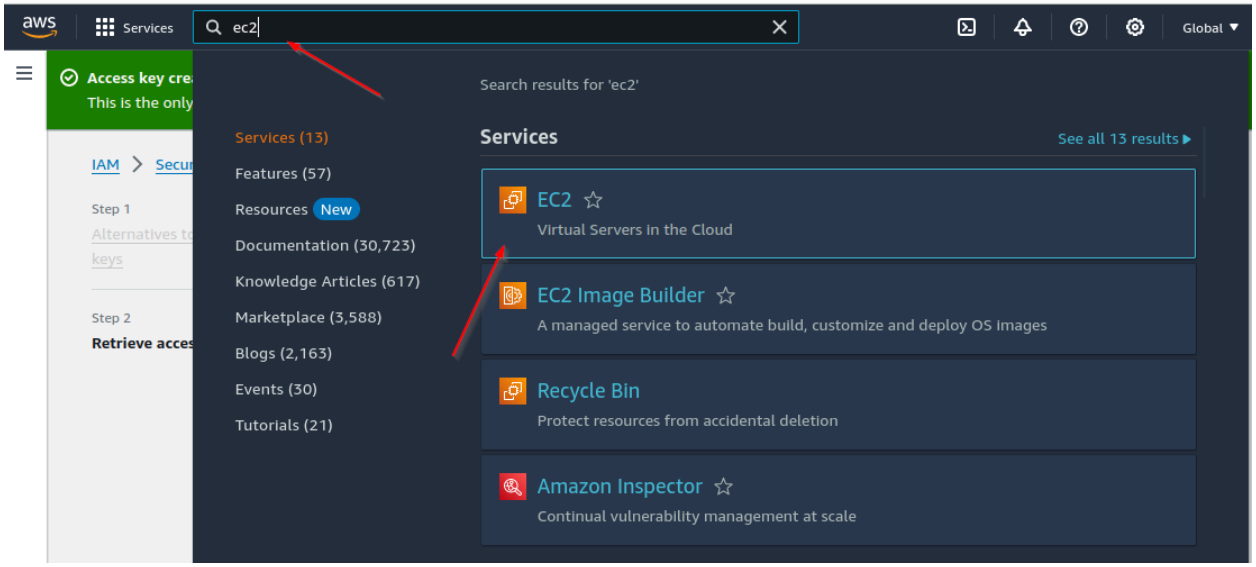
terraform apply

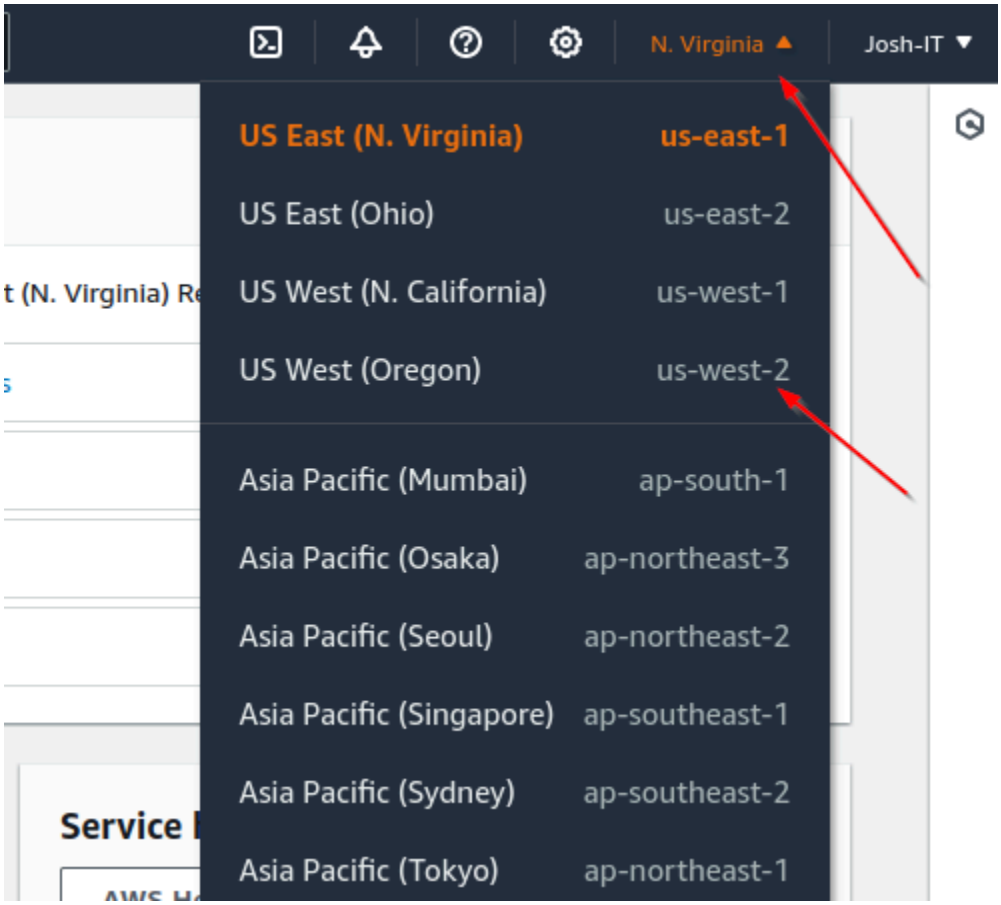
```
josh@josh-VirtualBox:~/learn-terraform-aws-instance$ terraform apply
```

3. Type "yes" when prompted.

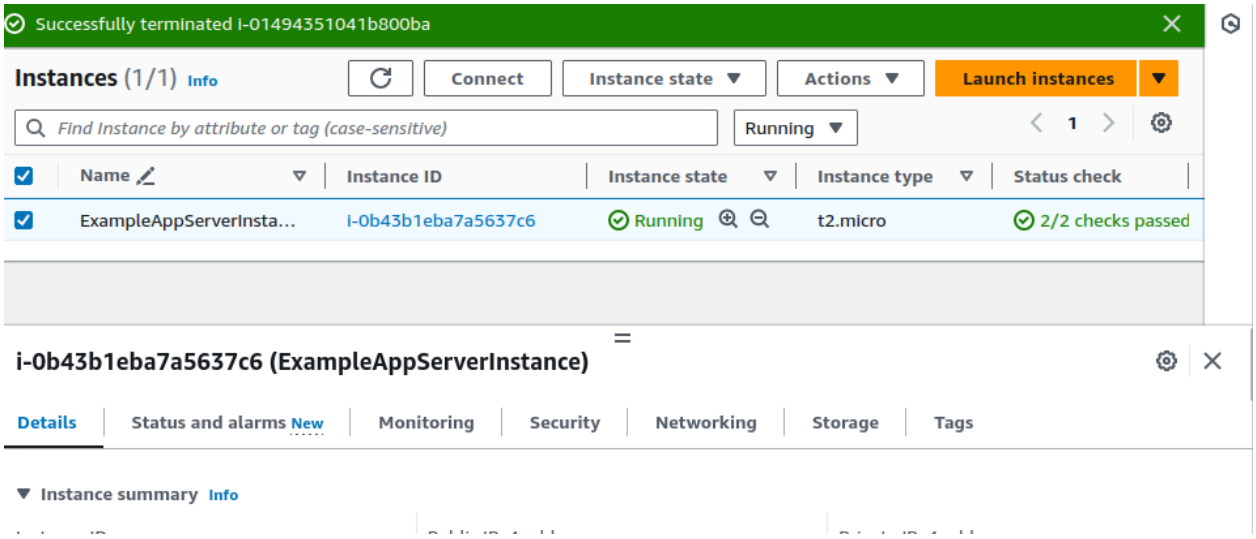
Step 7: Verify Infrastructure Creation

1. In the AWS Management Console, switch to the "us-west-2" region.





2. Navigate to "EC2 > Instances Running" and confirm that the instance is active.



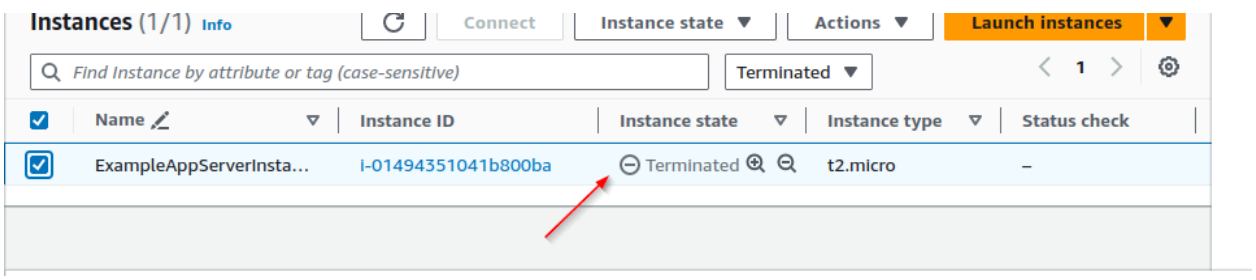
Step 8: Destroy the Infrastructure

1. To remove all resources:

terraform destroy

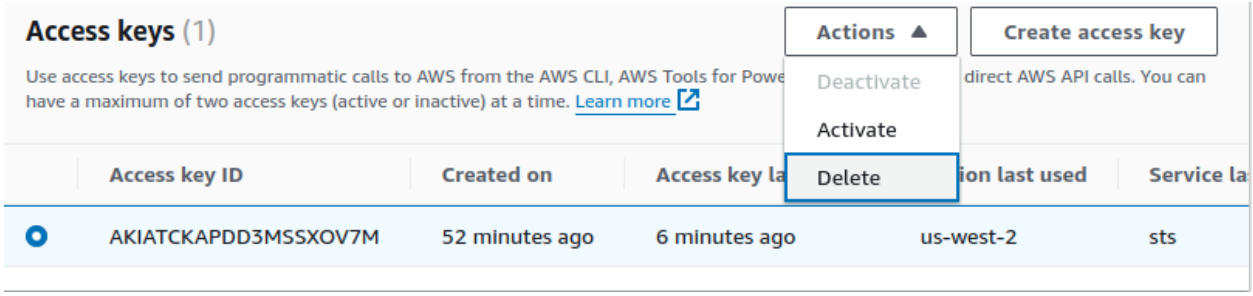
```
josh@josh-VirtualBox:~/learn-terraform-aws-instance$ terraform destroy
```

2. Type "yes" when prompted.



Step 9: Clean Up Access Keys

1. Go back to "Security Credentials" in your AWS account.
2. Deactivate and delete the access keys used for this demonstration.



Conclusion

Terraform enables quick, repeatable, and declarative infrastructure management. By simply running a few commands, we provisioned the necessary resources, verified the creation of a running instance in the AWS console, and later destroyed it to avoid lingering costs. This demonstration emphasized the efficiency and flexibility of infrastructure-as-code methodologies, helping to minimize manual intervention and reduce errors when managing complex cloud environments.