

Lesson 10 Demo 02

Authenticating Terraform State Backend

Objective: To authenticate and manage Terraform state using two different backend types, S3 standard backend and remote enhanced backend

Tools required: Visual Studio Code

Prerequisites: Ensure you have created and implemented the AWS access key and secret key before starting this demo. Refer to Lesson 08 Assisted Practice 02 for detailed steps

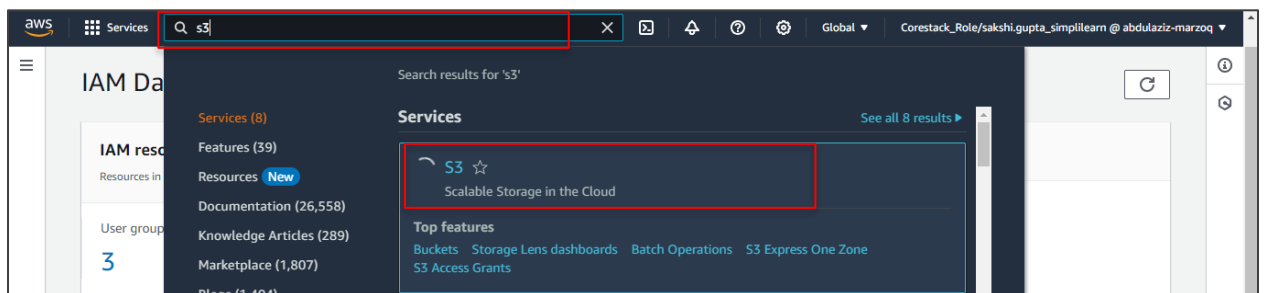
Note: The folder structure created in the previous demos is used here. It is also included in the resources section of LMS. Please refer Lesson_10_demo_01

Steps to be followed:

1. Authenticate S3 standard backend
2. Authenticate remote enhanced backend

Step 1: Authenticate S3 standard backend

1.1 Log in to the AWS Management Console and navigate to the **S3** service using the search bar



1.2 Click on **Create bucket**

Amazon S3

► **Account snapshot - updated every 24 hours** All AWS Regions [View Storage Lens dashboard](#)

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets | Directory buckets

General purpose buckets (1) Info All AWS Regions Copy ARN Empty Delete **Create bucket**

Buckets are containers for data stored in S3.

< 1 >

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	implicitdenydemo	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 30, 2022, 11:31:45 (UTC+05:30)

1.3 Name it appropriately for storing Terraform state files

[Amazon S3](#) > [Buckets](#) > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type Info

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory - New**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region
US East (N. Virginia) us-east-1

Bucket type [Info](#)

☒ **General purpose**
 Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory - New**
 Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

myterraformstatedemo

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
 Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

1.4 Retain all the other configurations at their default settings and click on **Create bucket**

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ **Block all public access**
 Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☒ **Block public access to buckets and objects granted through new access control lists (ACLs)**
 S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**
 S3 will ignore all ACLs that grant public access to buckets and objects.
- ☒ **Block public access to buckets and objects granted through new public bucket or access point policies**
 S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☒ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
 S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)

☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the **Storage** tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ Disable

☒ Enable

► Advanced settings

ⓘ

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

Create bucket

☐ myterraformstatedemo

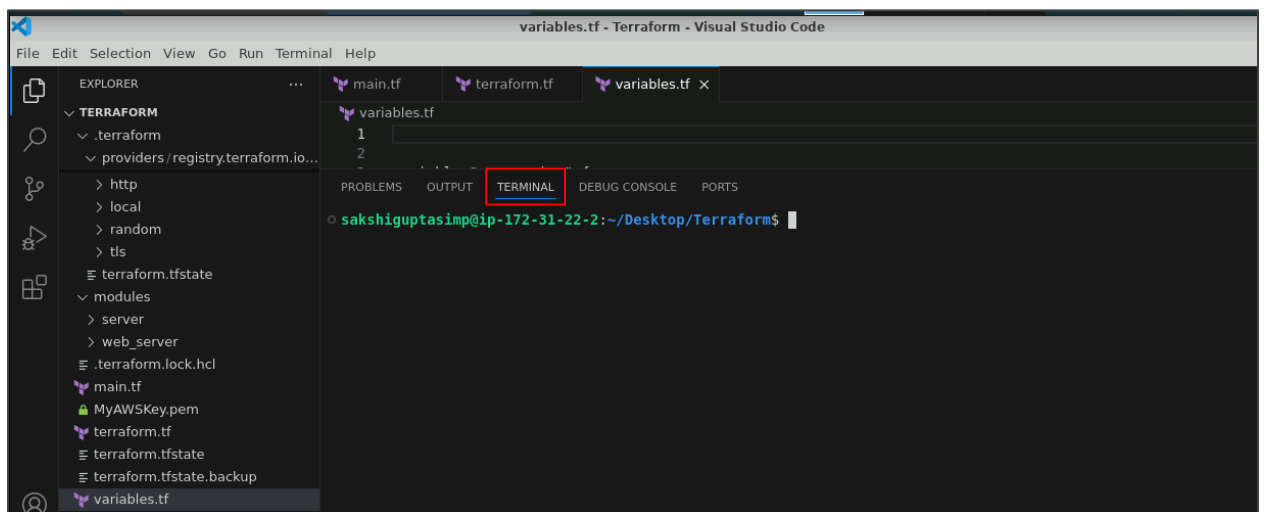
US East (N. Virginia) us-east-1

[View analyzer for us-east-1](#)

June 27, 2024, 18:30:11 (UTC+05:30)

The bucket will be created as shown above.

1.5 Open Visual Studio Code and navigate to the **Terminal**



1.6 Clean up any existing infrastructure to prepare for backend configuration by using the following command:

terraform destroy

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS
○ sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform destroy
```

1.7 When prompted, confirm the destruction of resources by typing **yes**

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

- public_ip                = "18.207.104.133" -> null
- public_ip_server_subnet_1 = "54.163.63.185" -> null
- size                     = "t2.micro" -> null

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
```

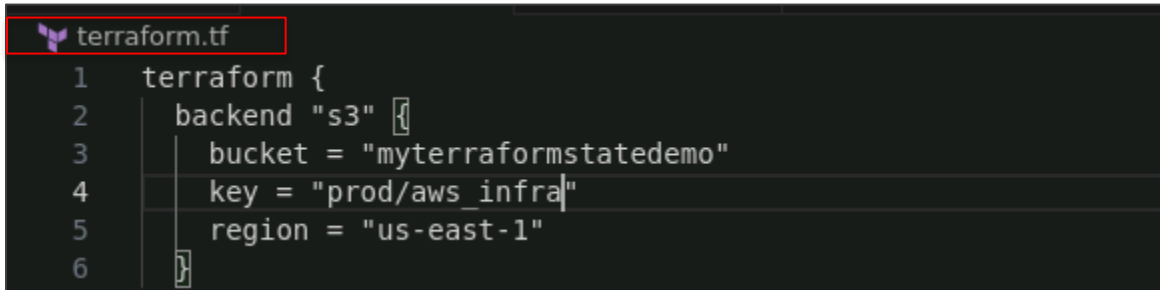
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

aws_nat_gateway.nat_gateway: Still destroying... [id=nat-05b819ebdb65964b, 40s elapsed]
aws_nat_gateway.nat_gateway: Destruction complete after 40s
aws_subnet.public_subnets["public_subnet_1"]: Destroying... [id=subnet-0d049c1ebbd0a0872]
aws_subnet.public_subnets["public_subnet_3"]: Destroying... [id=subnet-019ef181be686d113]
aws_eip.nat_gateway_eip: Destroying... [id=eipalloc-077e4d1caf6a7c46d]
aws_subnet.public_subnets["public_subnet_2"]: Destroying... [id=subnet-00149eda4bed0de7b]
aws_subnet.public_subnets["public_subnet_1"]: Destruction complete after 0s
aws_subnet.public_subnets["public_subnet_3"]: Destruction complete after 0s
aws_subnet.public_subnets["public_subnet_2"]: Destruction complete after 0s
aws_eip.nat_gateway_eip: Destruction complete after 1s
aws_internet_gateway.internet_gateway: Destroying... [id=igw-01ffeafe8937357ef]
aws_internet_gateway.internet_gateway: Destruction complete after 0s
aws_vpc.vpc: Destroying... [id=vpc-077c667402e032623]
aws_vpc.vpc: Destruction complete after 1s

Destroy complete! Resources: 30 destroyed.
○ sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

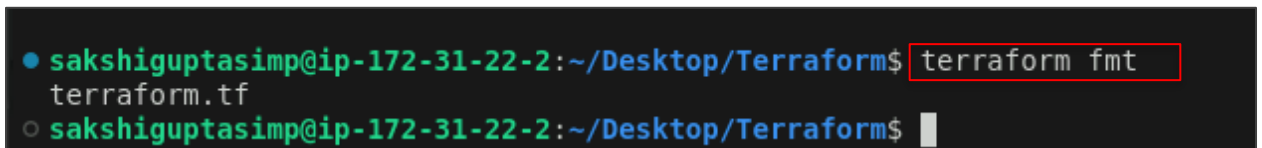
- 1.8 Go to **terraform.tf** file and set up Terraform to use the created S3 bucket by using the following code:

```
terraform {  
  backend "s3" {  
    bucket = "myterraformstatedemo"  
    key    = "prod/aws_infra"  
    region = "us-east-1"  
  }  
}
```

A screenshot of a code editor with a dark background. The file name 'terraform.tf' is highlighted in a red box at the top left. The code is as follows:

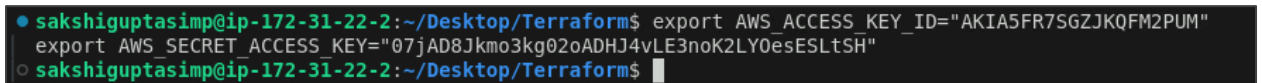
```
1 terraform {  
2   backend "s3" {  
3     bucket = "myterraformstatedemo"  
4     key    = "prod/aws_infra"  
5     region = "us-east-1"  
6   }  
}
```

- 1.9 Format the terraform configuration file by using the following command:
terraform fmt

A screenshot of a terminal window with a dark background. The prompt is 'sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform\$'. The command 'terraform fmt' is entered and highlighted in a red box. The output shows the file 'terraform.tf' being formatted.

```
● sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform fmt  
terraform.tf  
○ sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

- 1.10 Set the environment variables for AWS credentials by using the following command:
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"

A screenshot of a terminal window with a dark background. The prompt is 'sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform\$'. The commands to set AWS credentials are entered.

```
● sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ export AWS_ACCESS_KEY_ID="AKIA5FR7SGZJKQFM2PUM"  
export AWS_SECRET_ACCESS_KEY="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"  
○ sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

- 1.11 Initialize the Terraform working directory by using the following commands:
terraform init

```
● sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform init
Initializing modules...

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Using previously-installed hashicorp/tls v3.1.0
- Using previously-installed hashicorp/aws v3.76.1
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

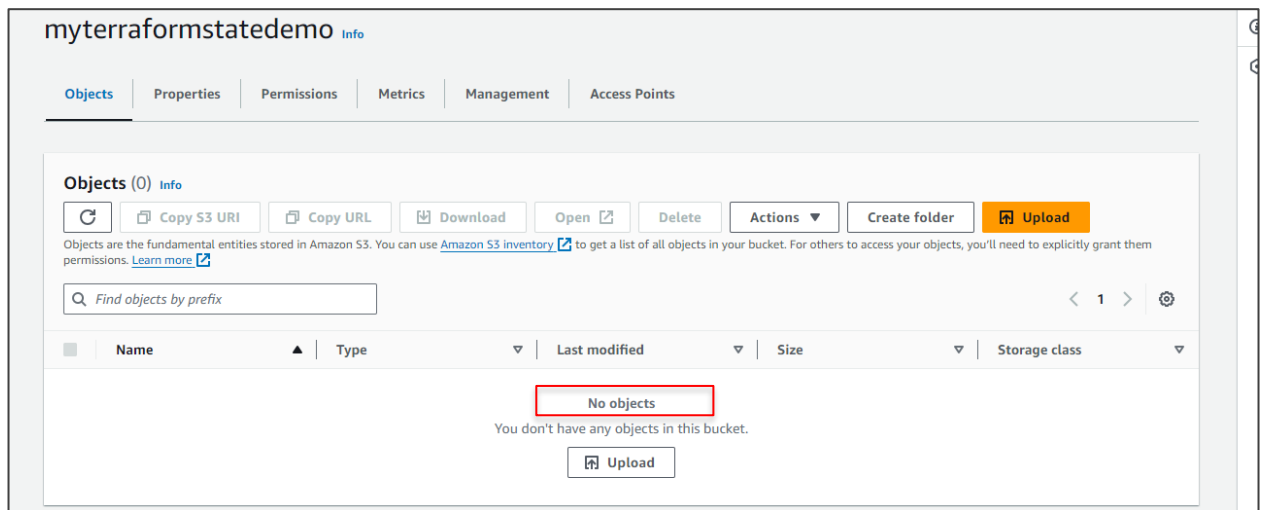
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Using previously-installed hashicorp/tls v3.1.0
- Using previously-installed hashicorp/aws v3.76.1
- Using previously-installed hashicorp/http v2.1.0
- Using previously-installed hashicorp/random v3.1.0
- Using previously-installed hashicorp/local v2.1.0

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.
● sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

- 1.12 Navigate to the newly created bucket in the AWS Console and observe that there are no **objects** yet



- 1.13 Navigate to the terminal in Visual Studio Code and update infrastructure resources by using the following command:
terraform apply

```
o sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

+ volume_id      = (known after apply)
+ volume_size    = (known after apply)
+ volume_type     = (known after apply)
}
}

Plan: 30 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ public_dns      = (known after apply)
+ public_dns_server_subnet_1 = (known after apply)
+ public_ip       = (known after apply)
+ public_ip_server_subnet_1 = (known after apply)
+ size            = "t2.micro"

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
```



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS

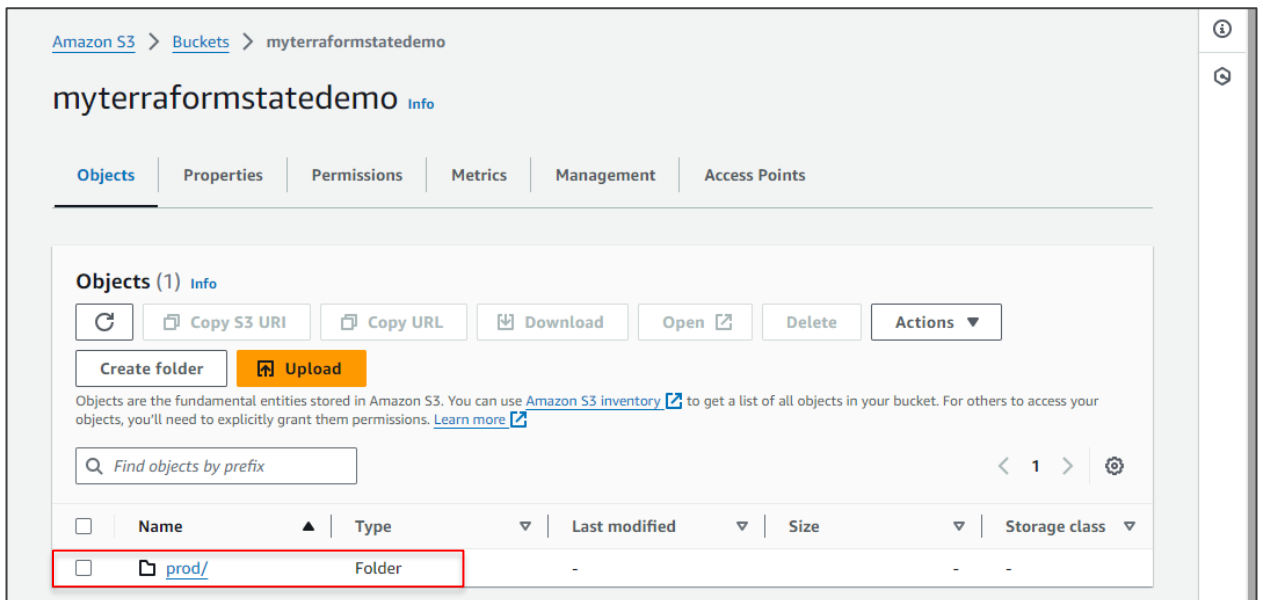
aws_route_table_association.private["private_subnet_3"]: Creating...
aws_route_table_association.private["private_subnet_1"]: Creating...
aws_route_table_association.private["private_subnet_2"]: Creating...
aws_route_table_association.private["private_subnet_3"]: Creation complete after 1s [id=rtbassoc-0dfa9f92545019a79]
aws_route_table_association.private["private_subnet_2"]: Creation complete after 1s [id=rtbassoc-0b1f4308fa4c8eae]
aws_route_table_association.private["private_subnet_1"]: Creation complete after 1s [id=rtbassoc-0012cbbd8402d4717]

Apply complete! Resources: 30 added, 0 changed, 0 destroyed.

Outputs:

public_dns = "ec2-34-201-56-208.compute-1.amazonaws.com"
public_dns_server_subnet_1 = "ec2-54-236-6-73.compute-1.amazonaws.com"
public_ip = "34.201.56.208"
public_ip_server_subnet_1 = "54.236.6.73"
size = "t2.micro"
o sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

- 1.14 Go to the created bucket in the AWS Console and observe that object has been added. Click on it to view it in detail.



prod/

Copy S3 URI

Objects

Properties

Objects (1) Info

Refresh

Copy S3 URI

Copy URL

Download

Open

Delete

Actions


Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

< 1 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 aws_infra	-	June 27, 2024, 18:53:58 (UTC+05:30)	63.0 KB	Standard

aws_infra Info

Copy S3 URI

Download

Open

Object actions

Properties

Permissions

Versions

Object overview

Owner

claaslabs+5eef7ae62d11de440f147083

AWS Region

US East (N. Virginia) us-east-1

Last modified


June 27, 2024, 18:53:58 (UTC+05:30)

Size


63.0 KB

Type


S3 URI

 s3://myterraformstatedemo/prod/aws_infra


Amazon Resource Name (ARN)

 arn:aws:s3:::myterraformstatedemo/prod/aws_infra

Entity tag (Etag)

 5d141a40576f12d428cda2567d06642d

Object URL

 https://myterraformstatedemo.s3.amazonaws.com/prod/aws_i

```
{
  "version": 4,
  "terraform_version": "1.1.6",
  "serial": 0,
  "lineage": "a592cde6-75be-4072-a088-766d409b08c7",
  "outputs": {
    "public_dns": {
      "value": "ec2-34-201-56-208.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_dns_server_subnet_1": {
      "value": "ec2-54-236-6-73.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_ip": {
      "value": "34.201.56.208",
      "type": "string"
    },
    "public_ip_server_subnet_1": {
      "value": "54.236.6.73",
      "type": "string"
    },
    "size": {
      "value": "t2.micro",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "data",
      "type": "aws_ami",
      "name": "ubuntu",
      "provider": "provider[\\\"registry.terraform.io/hashicorp/aws\\\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "architecture": "x86_64"
```

Step 2: Authenticate remote enhanced backend authentication


2.1 Create a Terraform Cloud account by visiting the following URL:

<https://app.terraform.io/public/signup/account>

← → ↻ app.terraform.io/public/signup/account < ☆ □ 👤 Relaunch

Create an account


Have an account? [Sign in](#)


 Continue with HCP account

OR

Username

Email

Password
 



HCP Terraform

You're minutes away from collaborating on infrastructure with your team.


- ✓ Single workflow across multiple providers to save time
- ✓ Write infrastructure as code to increase productivity
- ✓ Re-use configurations to reduce mistakes

2.2 Enter the details and click on **Create account**

OR

Username

Email


Password
 

☐ I agree to the [Terms of Use](#).

☐ I acknowledge the [Privacy Policy](#).

Please review the [Terms of Use](#) and [Privacy Policy](#).

Create account



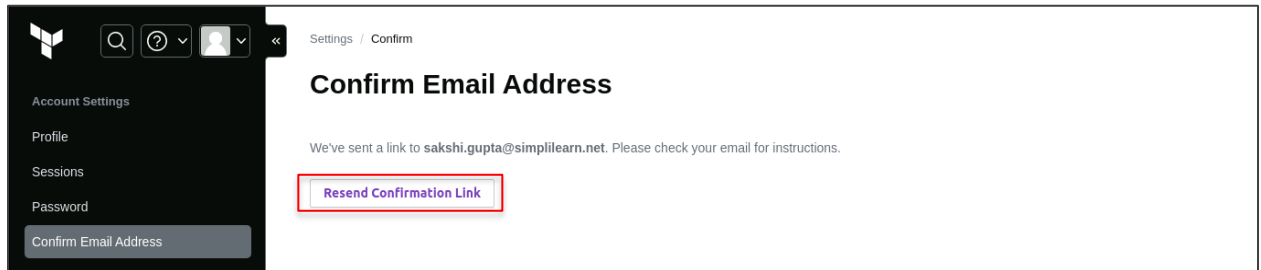
HCP Terraform

You're minutes away from collaborating on infrastructure with your team.

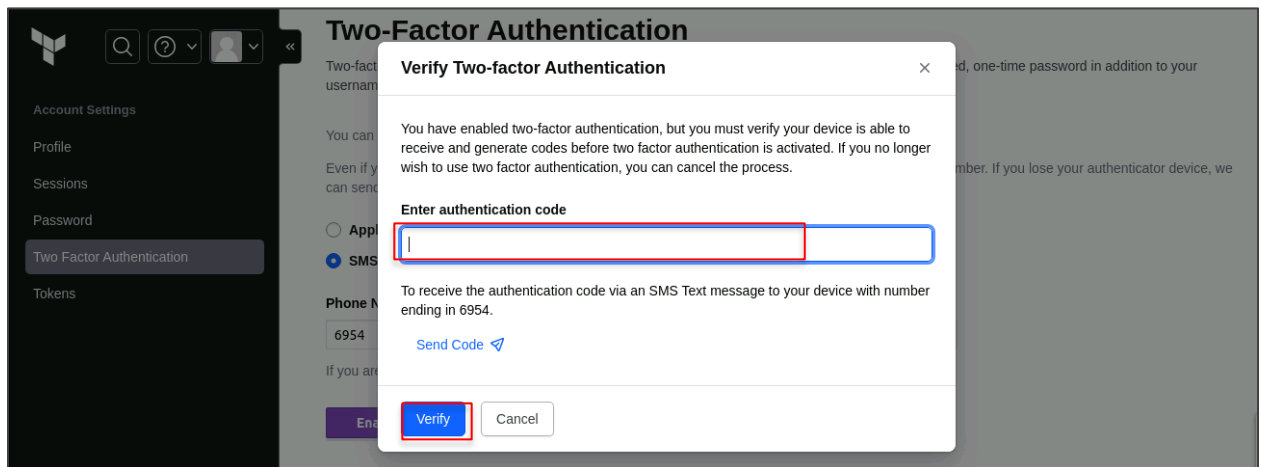
- ✓ Single workflow across multiple providers to save time
- ✓ Write infrastructure as code to increase productivity
- ✓ Re-use configurations to reduce mistakes

[Learn more about Terraform](#)

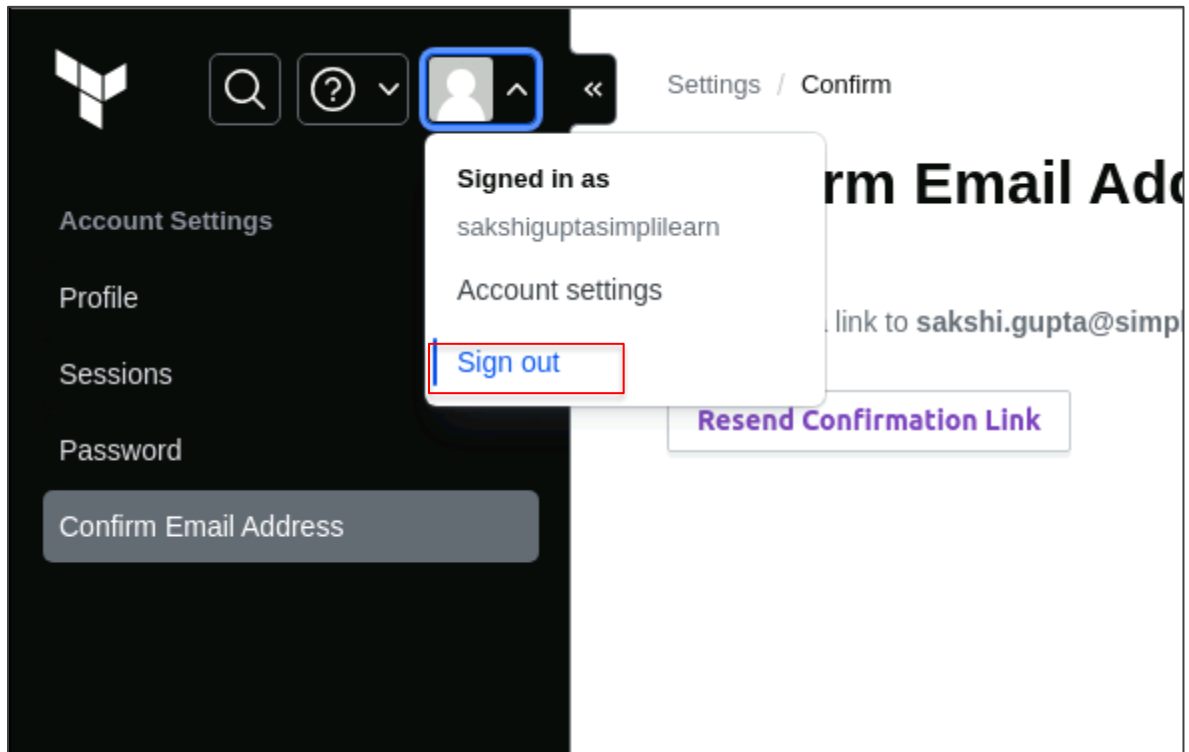
2.3 Click on **Resend Confirmation Link** and follow the procedures through your registered email



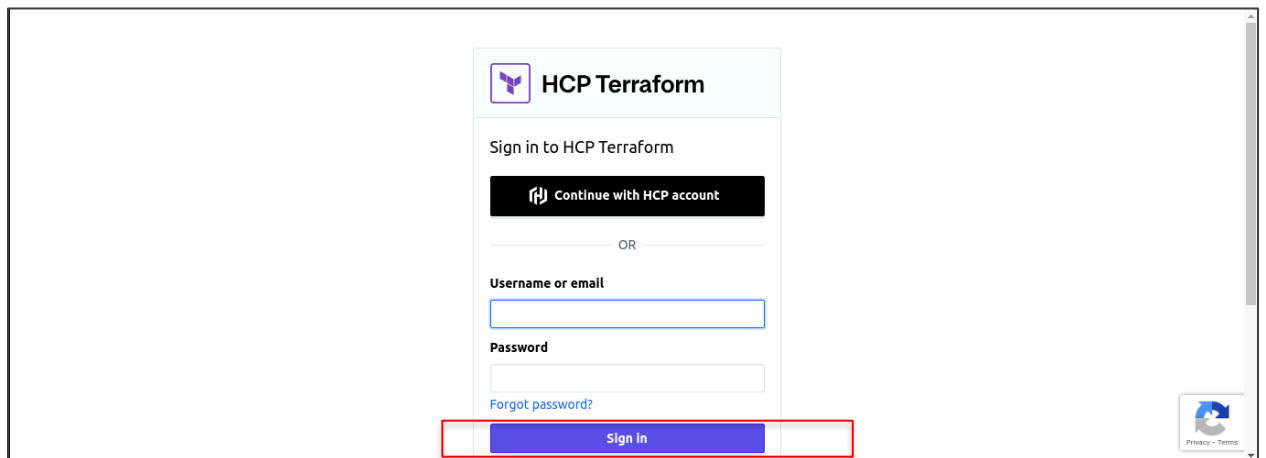
2.4 When prompted, enter the authentication code and click **Verify**.



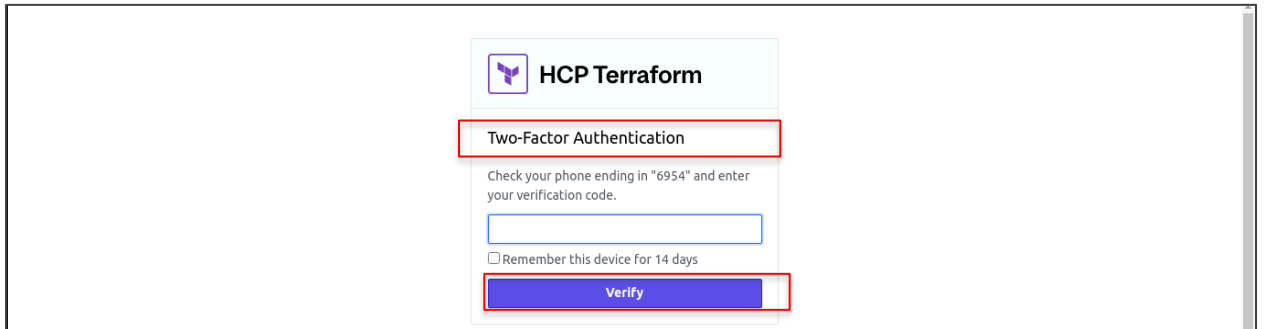
2.5 Sign out and re-login to the Terraform Cloud



2.6 Enter the required details and click on Sign In

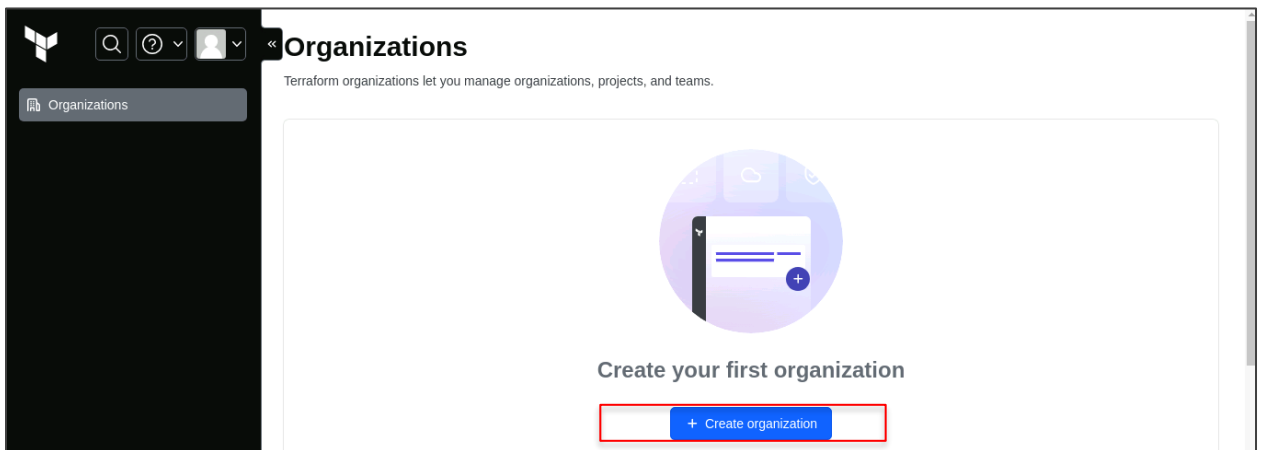


2.7 Click **Verify** after entering the verification code



The screenshot shows a modal dialog titled "HCP Terraform" with a purple icon. Below the title is a section labeled "Two-Factor Authentication" enclosed in a red box. The text inside says "Check your phone ending in '6954' and enter your verification code." Below this is a text input field. Under the input field is a checkbox labeled "Remember this device for 14 days". At the bottom of the dialog is a blue button labeled "Verify", which is also enclosed in a red box.

2.8 Click on **Create organization**



2.9 Enter a unique organization name and email address, then click on **Create organization**

[<<](#) Organizations / New

Create a new organization

Organizations are privately shared spaces for teams to collaborate on infrastructure. [Learn more](#) about organizations in HCP Terraform.


Organization name

Organization names must be unique and can only include numbers, letters, underscores (`_`), and hyphens (`-`).

Email address

The organization email is used for any future notifications, such as billing alerts, and the organization avatar, via [gravatar.com](#).

Create organization



🔍

?

👤

my_demo_organization_01 / Workspaces / New Workspace

Manage

Projects

Workspaces

Registry

Usage

Settings

Visibility

Explorer

Create a new Workspace

HCP Terraform organizes your infrastructure resources by workspaces. A workspace contains infrastructure resources, variables, state data, and run history. [Learn more](#) about workspaces in HCP Terraform.

Choose your workflow

Version Control Workflow

Trigger runs based on changes to configuration in repositories.

Best for those who need traceability and transparency

CLI-Driven Workflow

Trigger runs in a workspace using the Terraform CLI.

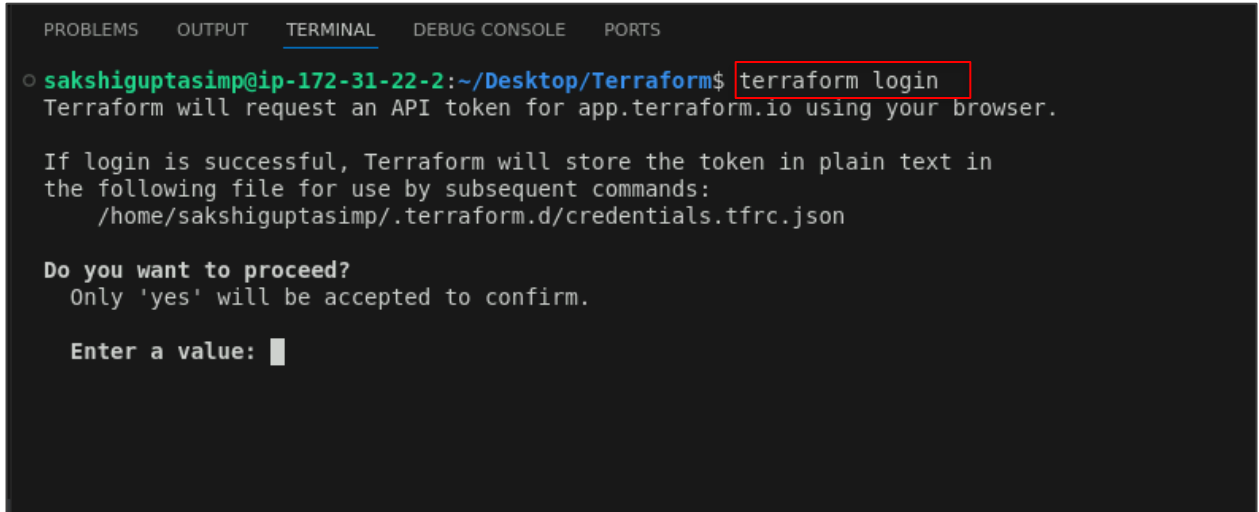
Best for those comfortable with Terraform CLI

API-Driven Workflow

Trigger runs using the HCP Terraform API.

Best for those with custom integrations and pipelines

- 2.10 Authenticate the Terraform CLI to interact with the Terraform Cloud by using the following command:
terraform login



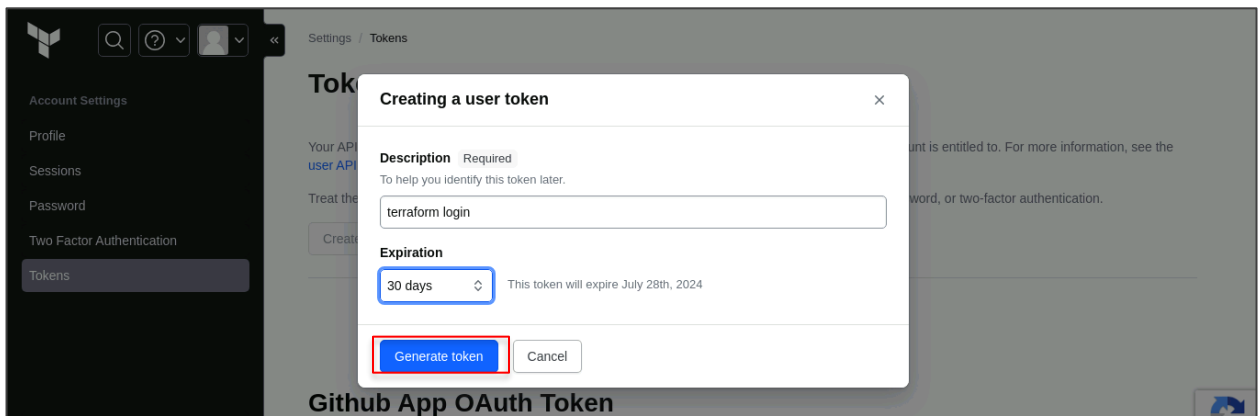
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform login
Terraform will request an API token for app.terraform.io using your browser.

If login is successful, Terraform will store the token in plain text in
the following file for use by subsequent commands:
/home/sakshiguptasimp/.terraform.d/credentials.tfrc.json

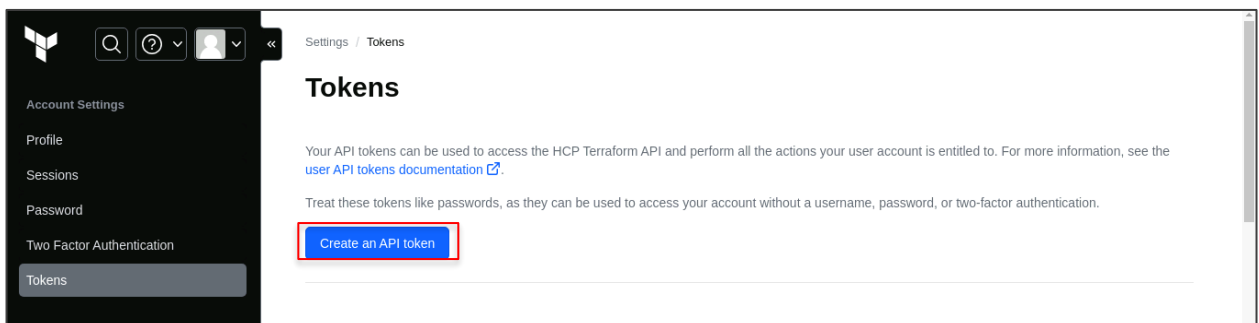
Do you want to proceed?
Only 'yes' will be accepted to confirm.

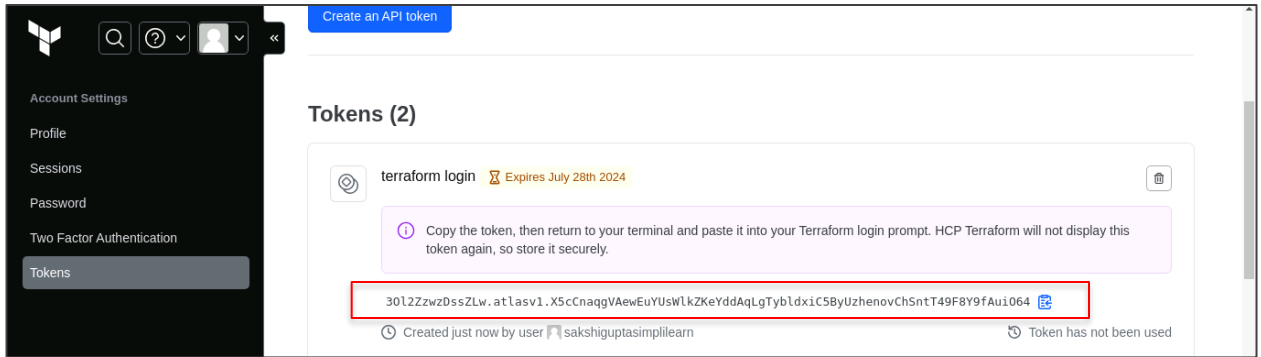
Enter a value: 
```

- 2.11 The Terraform Cloud will automatically open in the browser. Click on **Generate token**.

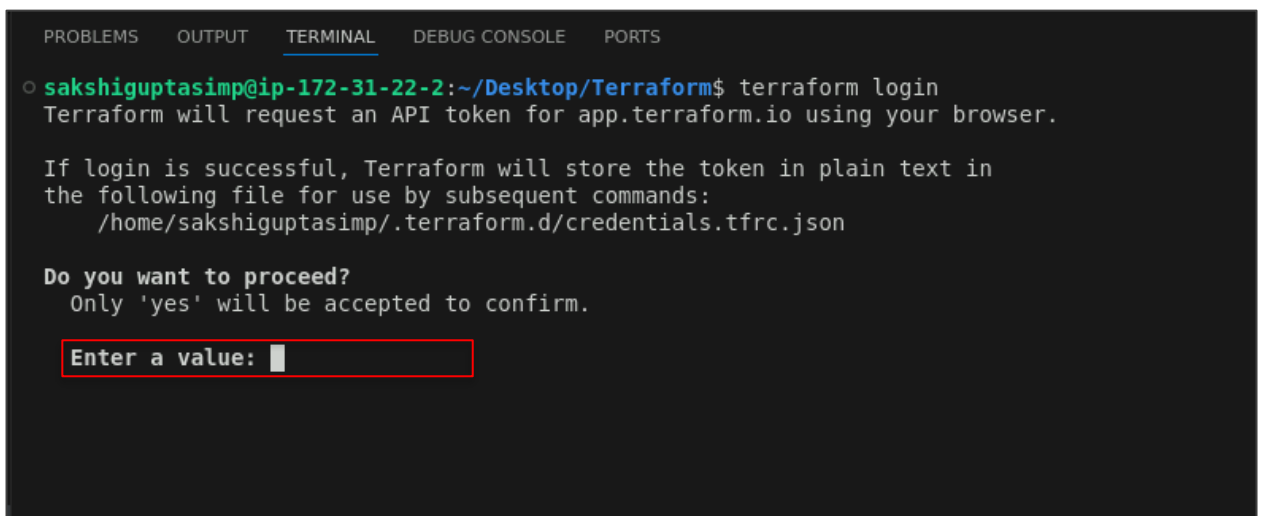


- 2.12 Click on **Create an API token** and copy the token when generated

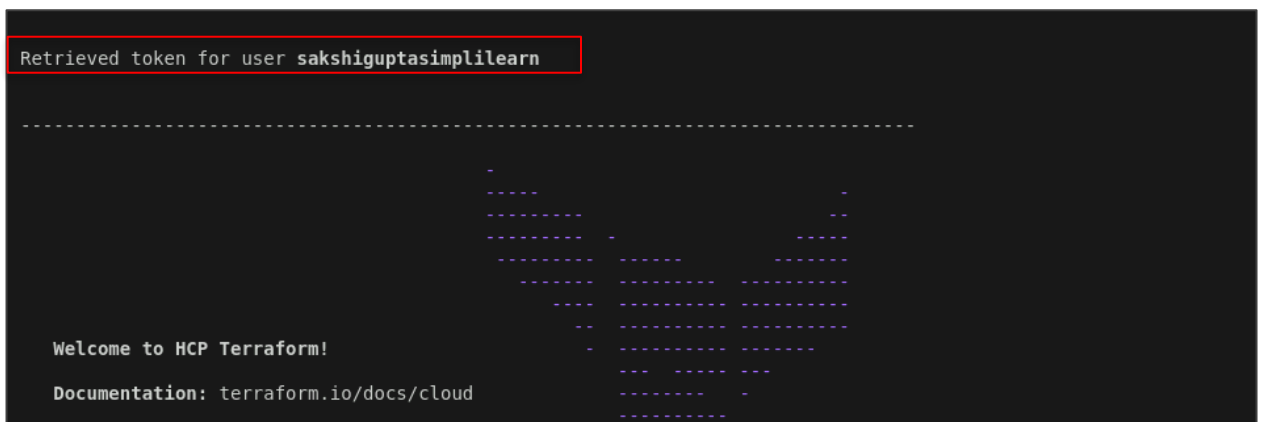




2.13 Paste the generated token in the Visual Studio Code terminal:



2.14 A welcome message from Terraform will be displayed.



```

Welcome to HCP Terraform!

Documentation: terraform.io/docs/cloud

New to HCP Terraform? Follow these steps to instantly apply an example configuration:

$ git clone https://github.com/hashicorp/tfc-getting-started.git
$ cd tfc-getting-started
$ scripts/setup.sh

sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

By following the above steps, you have successfully authenticated and managed Terraform state using two different backend types, S3 standard backend and remote enhanced backend.