Lesson 13 Demo 01

Working with Workspaces on Terraform Cloud

Objective: To use Terraform Cloud to manage infrastructure by creating and managing workspaces for deploying resources

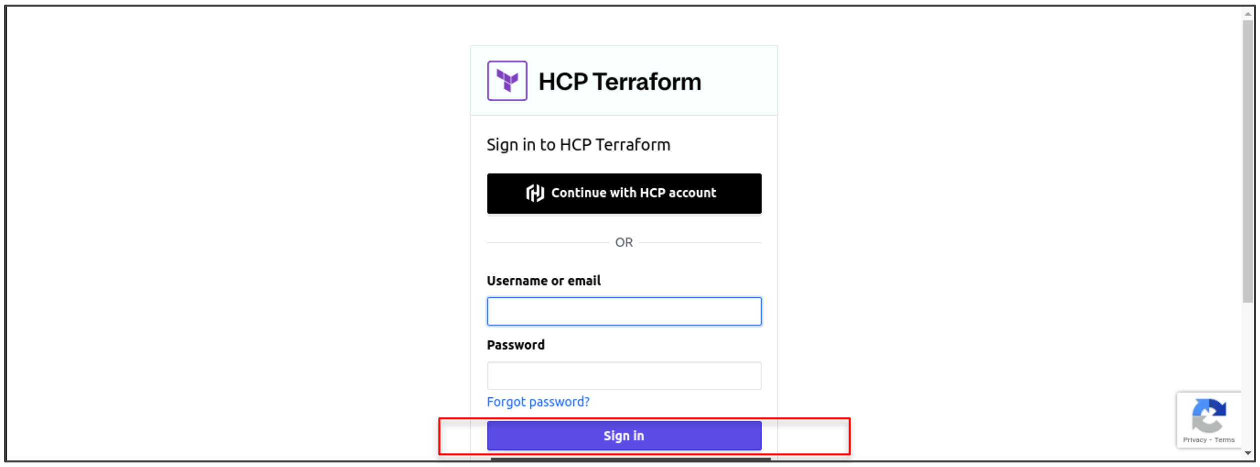
Tools required: None

Prerequisites: Terraform Cloud account
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.

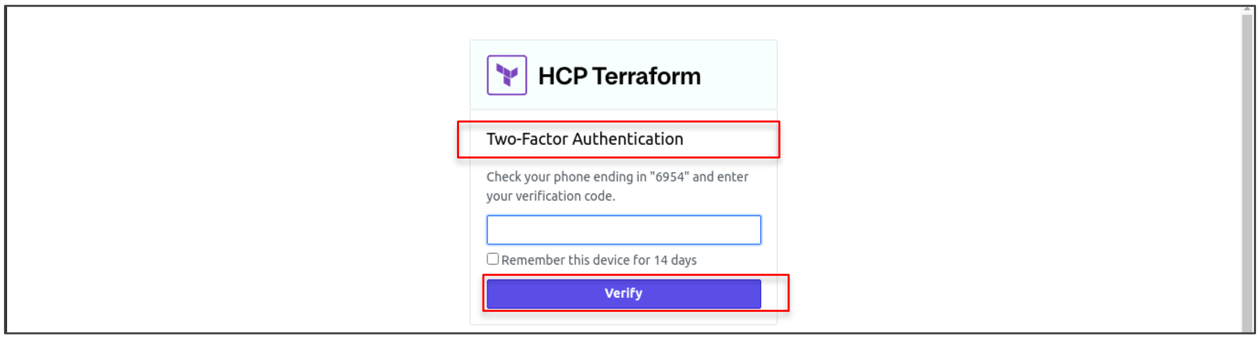
Steps to be followed:
1. Sign in to the Terraform Cloud platform
2. Create an organization and workspace
3. Initialize Terraform
4. Plan and apply the configurations
5. Select and create workspaces

Step 1: Sign in to the Terraform Cloud platform

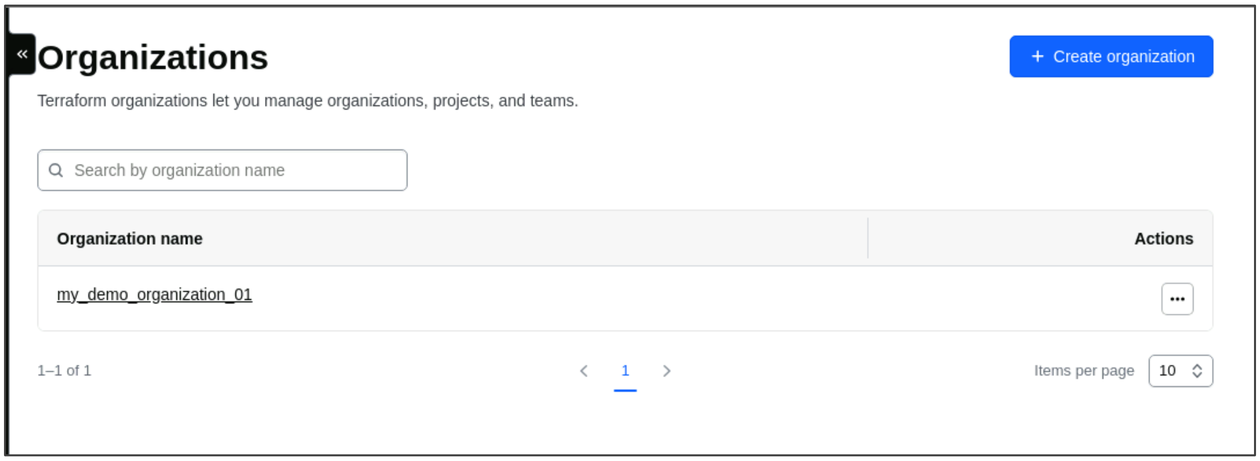
1.1 Enter the required details and click on **Sign In** by using the following URL:
https://app.terraform.io/session



1.2 Click **Verify** after entering your verification code



Check your phone ending in "6954" and enter your verification code.



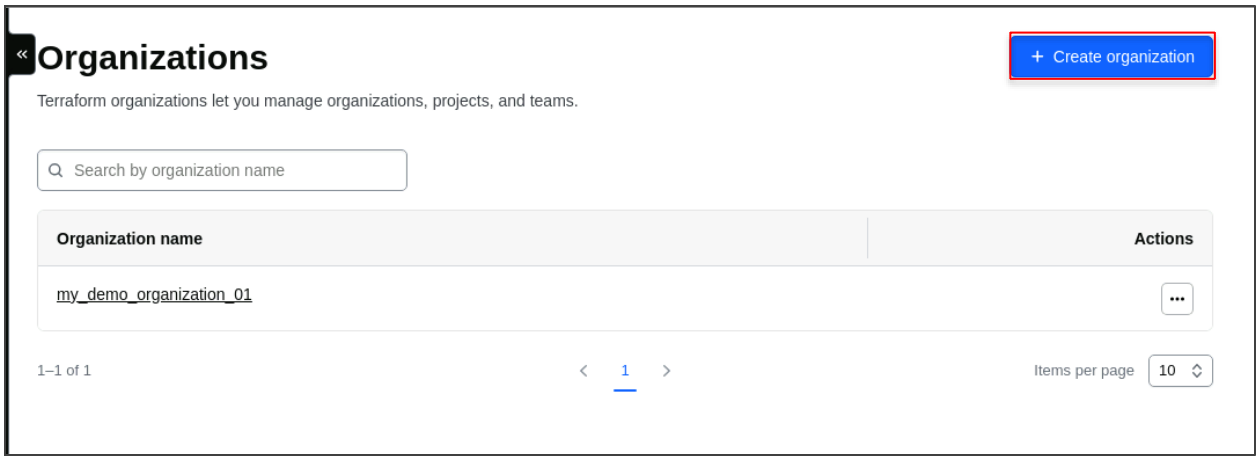
Terraform organizations let you manage organizations, projects, and teams.

Organization name	Actions
my_demo_organization_01	•••

1–1 of 1   1   Items per page 10

Step 2: Create an organization and workspace

2.1 Click on **Create organization**

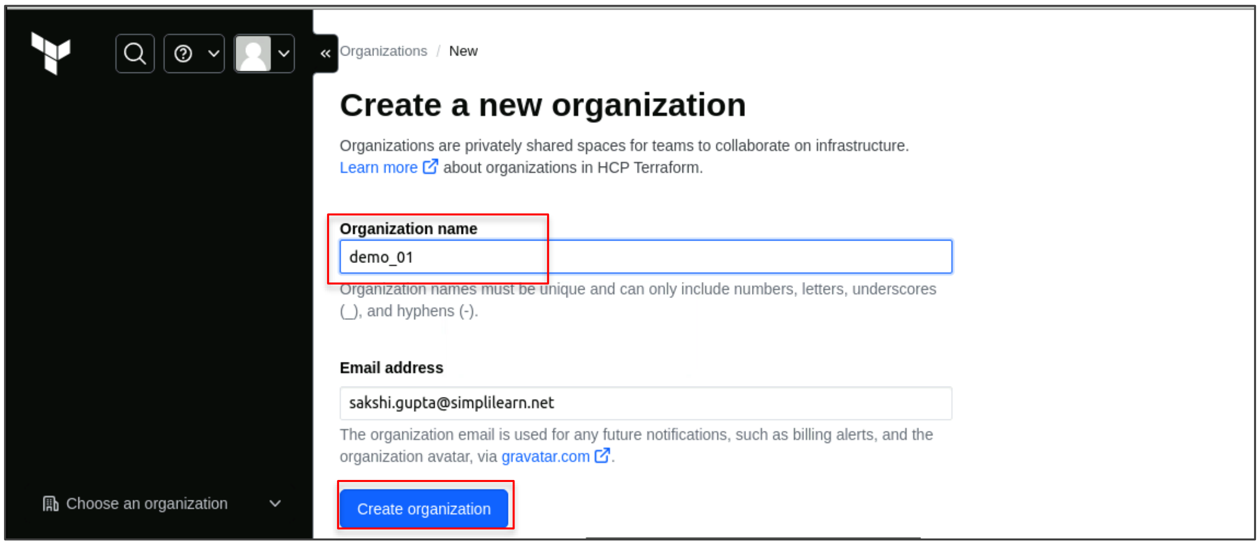


Terraform organizations let you manage organizations, projects, and teams.

Organization name	Actions
my_demo_organization_01	•••

1–1 of 1   1   Items per page 10

2.2 Enter the **Organization name** as **demo_01** and 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

demo_01

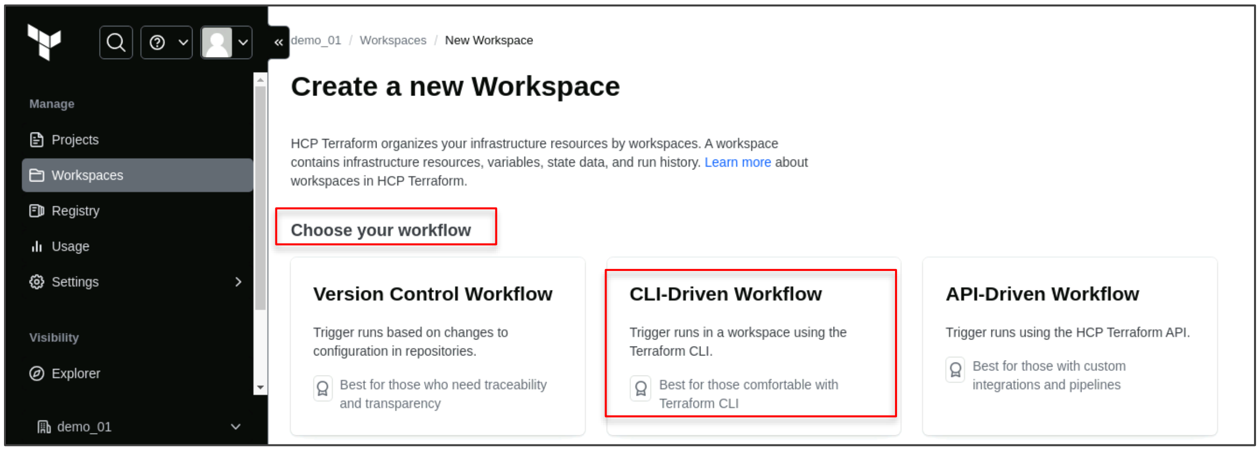
Organization names must be unique and can only include numbers, letters, underscores (_), and hyphens (-).

Email address

sakshi.gupta@simplilearn.net

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

2.3 **Choose your workflow** as **CLI-Driven Workflow**



demo_01 / Workspaces / New Workspace

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.

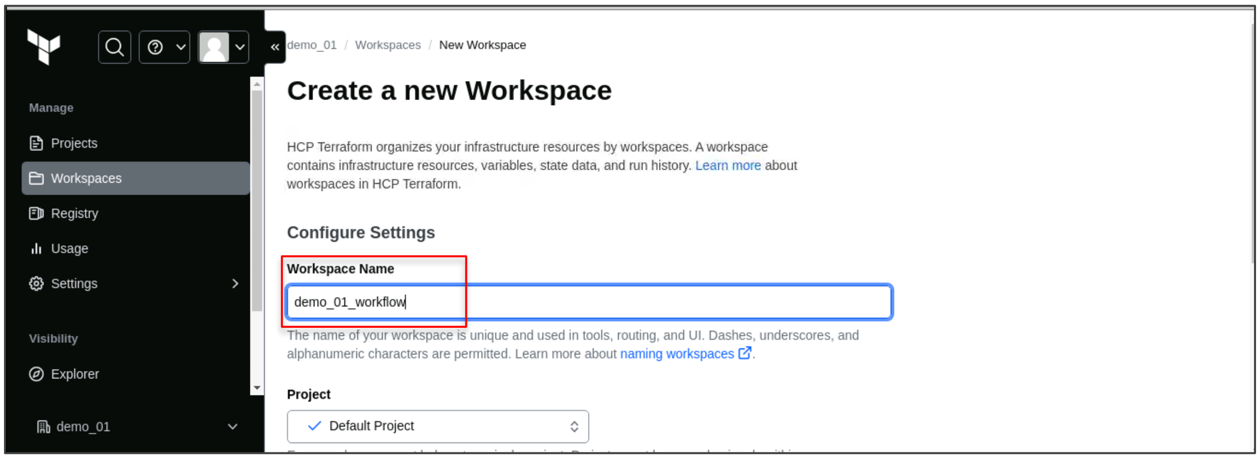
Choose your workflow

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

2.4 Enter the **Workspace Name** as **demo_01_workflow**


demo_01 / Workspaces / New Workspace

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.

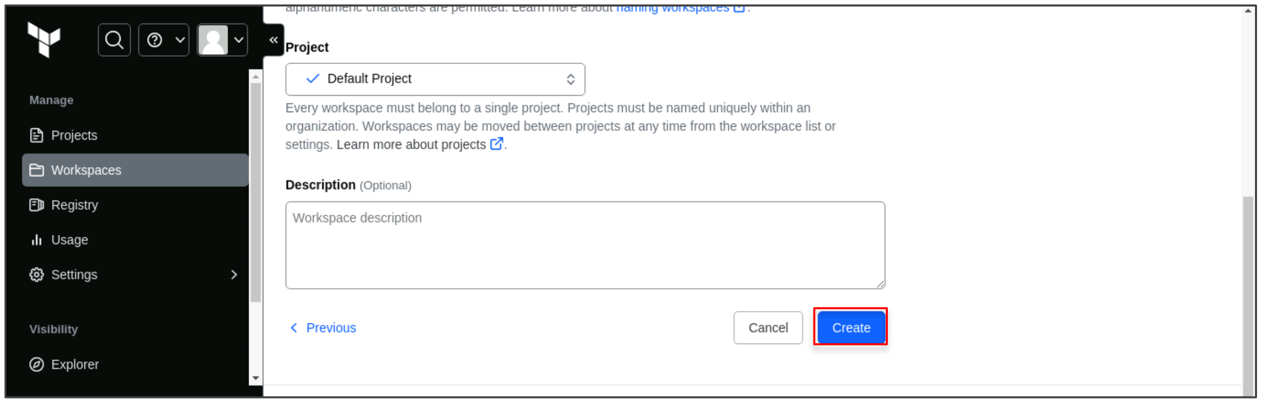
Configure Settings

Workspace Name: demo_01_workflow

The name of your workspace is unique and used in tools, routing, and UI. Dashes, underscores, and alphanumeric characters are permitted. Learn more about naming workspaces.

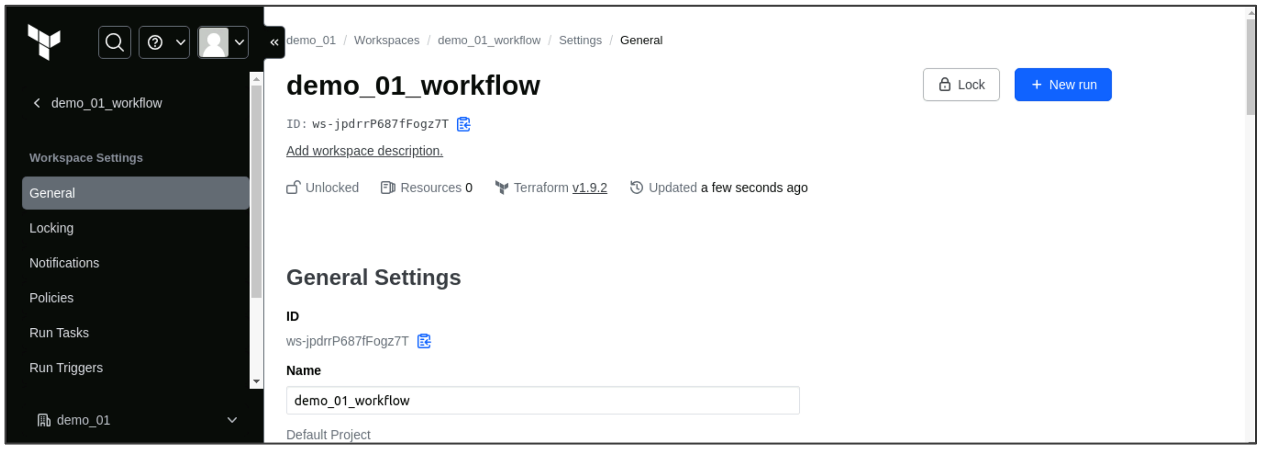
Project: Default Project

2.5 Scroll down and click on **Create**


Project: Default Project

Every workspace must belong to a single project. Projects must be named uniquely within an organization. Workspaces may be moved between projects at any time from the workspace list or settings. Learn more about projects.

Description (Optional): Workspace description


demo_01 / Workspaces / demo_01_workflow / Settings / General

demo_01_workflow

ID: ws-jpdrrP687fFogz7T

Add workspace description.

Unlocked · Resources 0 · Terraform v1.9.2 · Updated a few seconds ago

General Settings

ID: ws-jpdrrP687fFogz7T

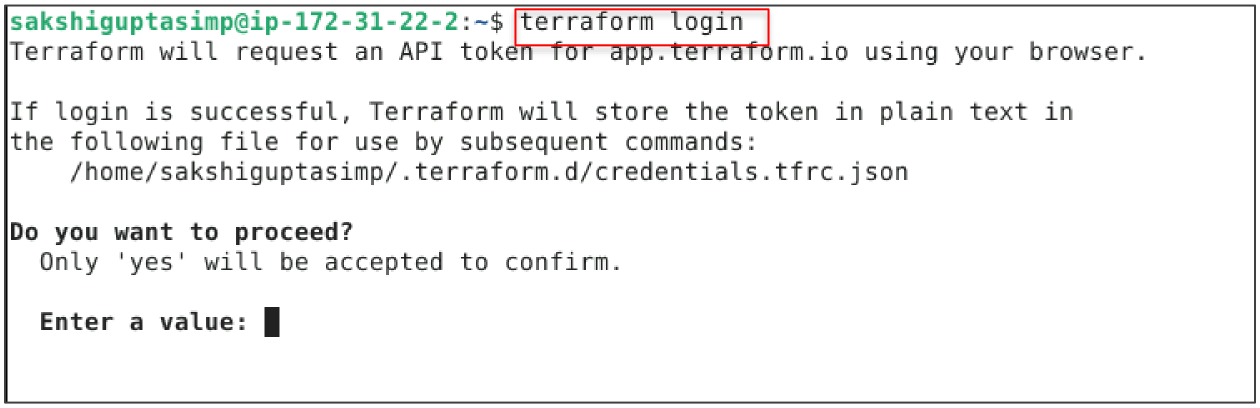
Name: demo_01_workflow

Default Project

The workspace will be created as shown.

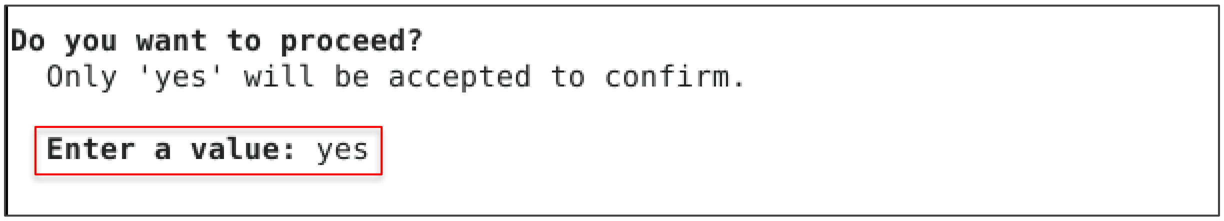
Step 3: Initialize Terraform

3.1 Go to the terminal and run the following command to log in to Terraform Cloud:
terraform login



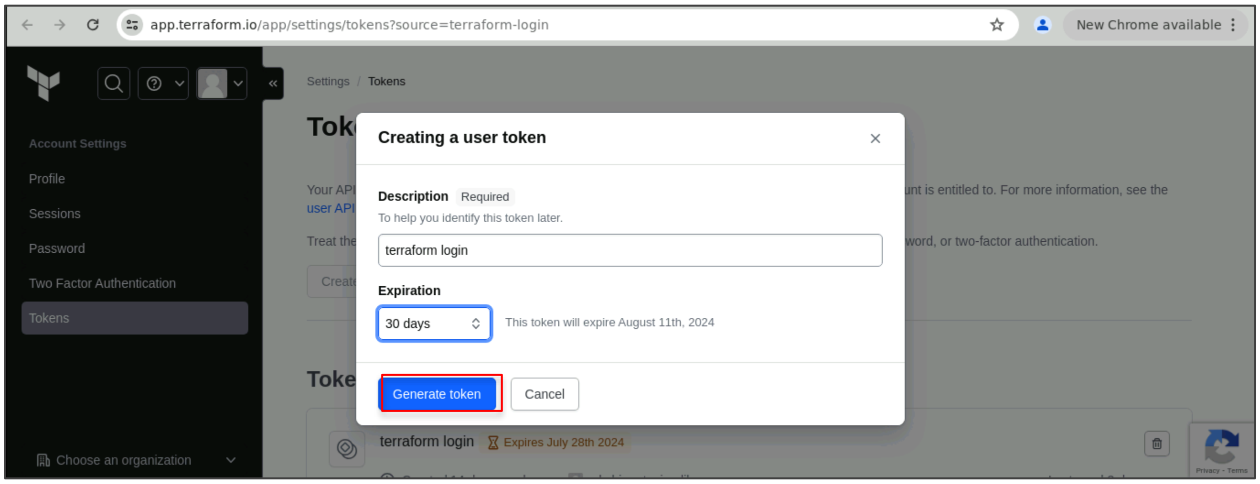
```
sakshiguptasimp@ip-172-31-22-2:~$ terraform login
Terraform will request an API token for app.terraform.io using your browser.

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: 
```

3.2 When prompted, proceed by typing **yes**



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

  Enter a value: yes
```

3.3 The Terraform Cloud interface will automatically open, and you can create a user token by clicking on **Generate token**.



app.terraform.io/app/settings/tokens?source=terraform-login

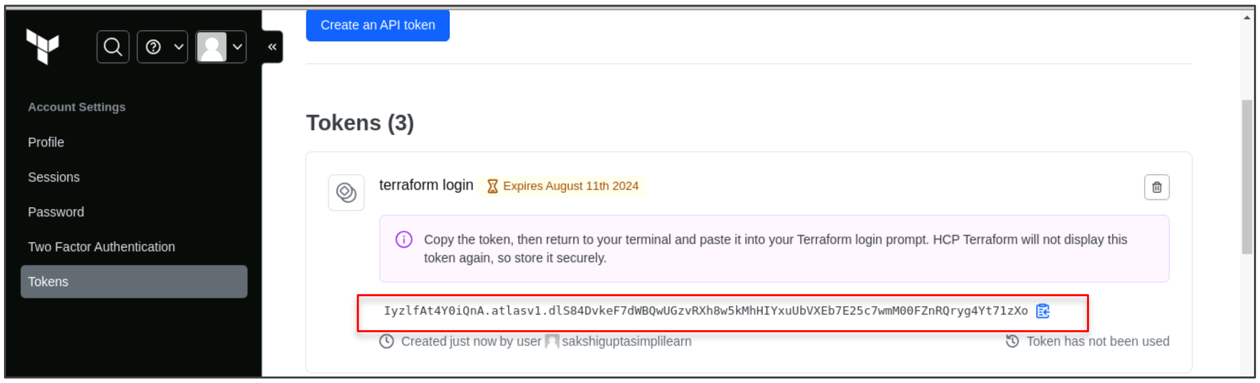
Creating a user token

Description Required
To help you identify this token later.
terraform login

Expiration
30 days
This token will expire August 11th, 2024

terraform login Expires July 28th 2024

3.4 Scroll down and copy the generated token



Tokens (3)

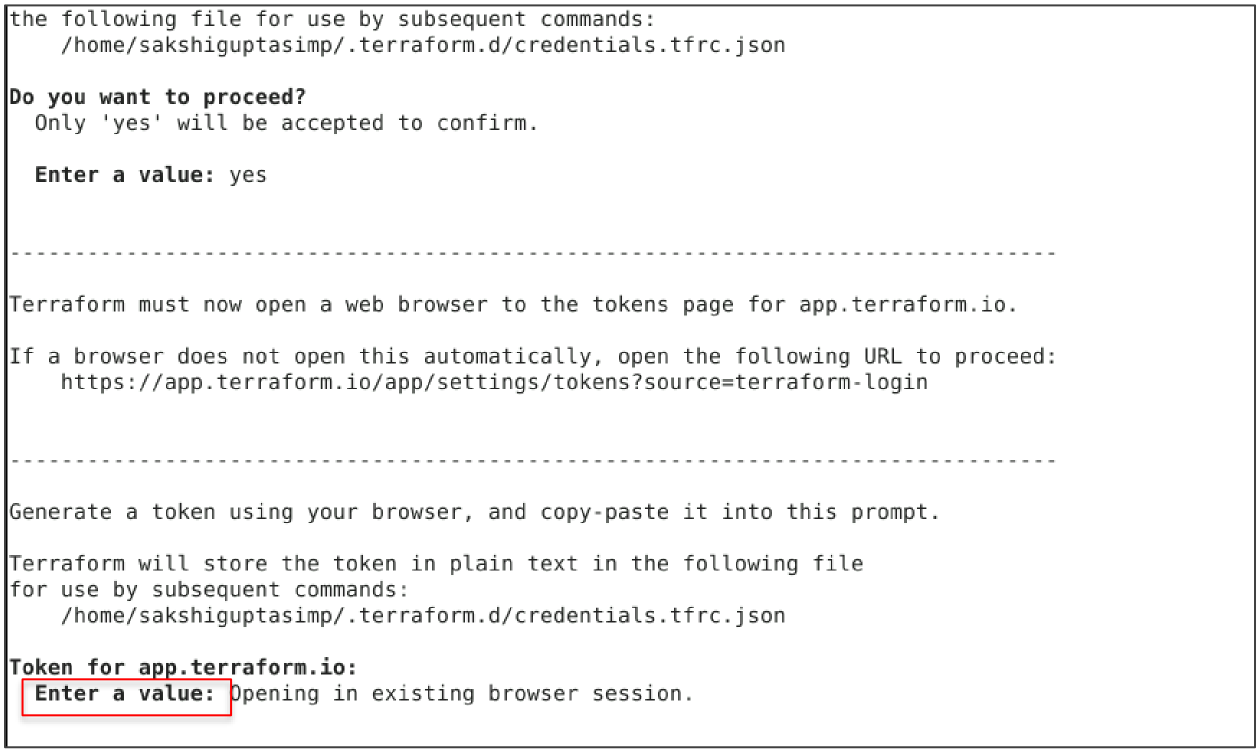
terraform login   Expires August 11th 2024

Copy the token, then return to your terminal and paste it into your Terraform login prompt. HCP Terraform will not display this token again, so store it securely.

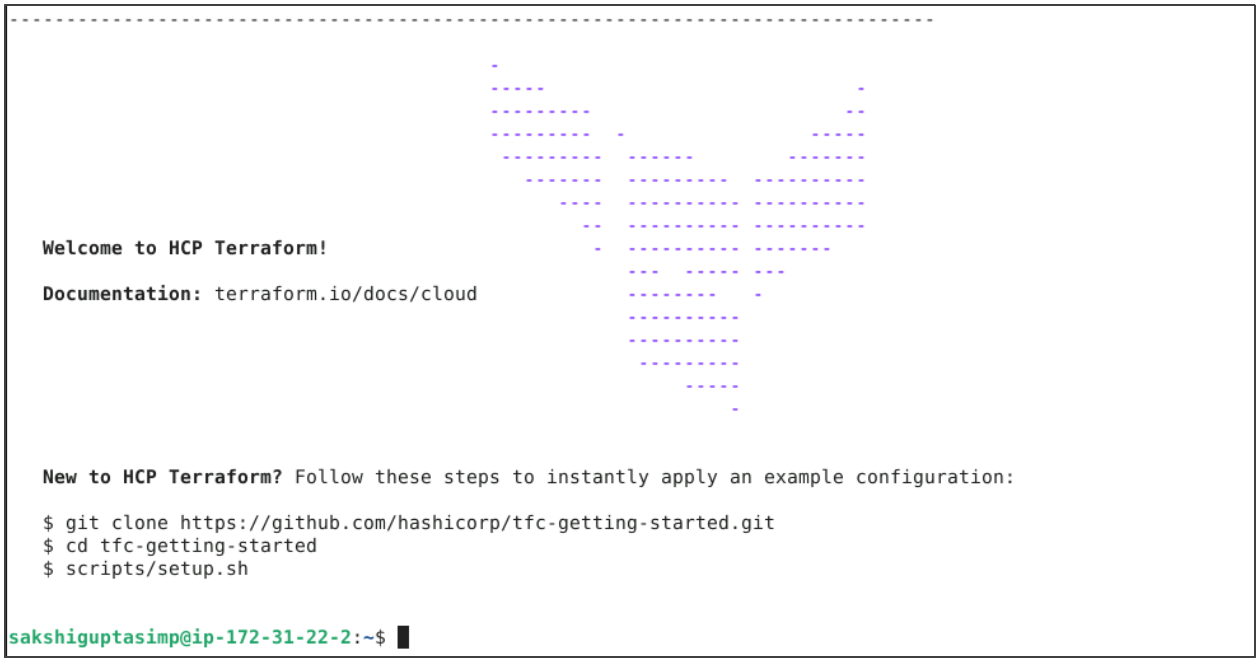
```
IyzlfAt4Y0iQnA.atlasv1.dlS84DvkeF7dWBQwUGzvRXh8w5kMhHIYxuUbVXEb7E25c7wmM00FZnRQryg4Yt71zXo
```

Created just now by user sakshiguptasimplilearn   Token has not been used

3.5 Go to the terminal and paste the copied token



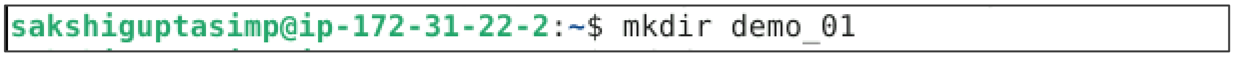
```
the following file for use by subsequent commands:
    /home/sakshiguptasimp/.terraform.d/credentials.tfrc.json

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

  Enter a value: yes


---------------------------------------------------------------------------------

Terraform must now open a web browser to the tokens page for app.terraform.io.

If a browser does not open this automatically, open the following URL to proceed:
    https://app.terraform.io/app/settings/tokens?source=terraform-login


---------------------------------------------------------------------------------

Generate a token using your browser, and copy-paste it into this prompt.

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

Token for app.terraform.io:
  Enter a value: Opening in existing browser session.
```



```
-------------------------------------------------------------------------------

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:~$
```

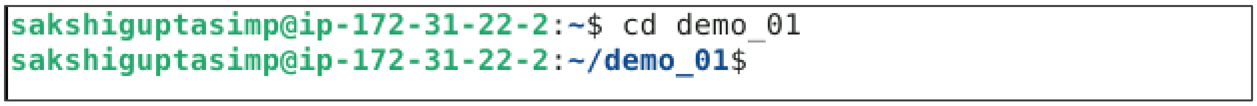
A welcome message from Terraform will appear, as shown.

3.6 Create a folder to proceed with terraform initialization using the following command:
mkdir demo_01



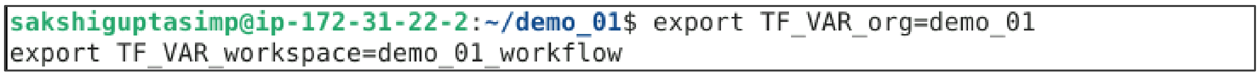
```
sakshiguptasimp@ip-172-31-22-2:~$ mkdir demo_01
```

3.7 Go to the created folder by using the following command:
cd demo_01



```
sakshiguptasimp@ip-172-31-22-2:~$ cd demo_01
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

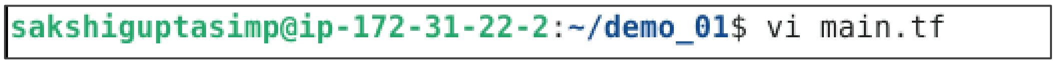
3.8 Set the Terraform Cloud organization and workspace by using the following command:
export TF_VAR_org=demo_01
export TF_VAR_workspace=demo_01_workflow



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ export TF_VAR_org=demo_01
export TF_VAR_workspace=demo_01_workflow
```

3.9 Create a **main.tf** file by using the following command:

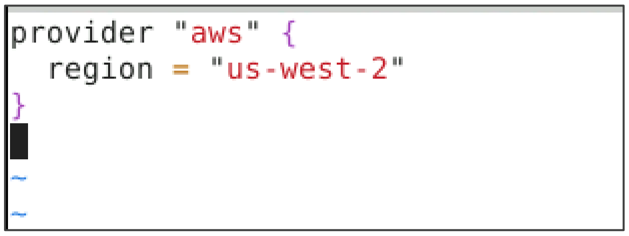
vi main.tf



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ vi main.tf
```

3.10 Enter the following code in the **main.tf** file to configure AWS:

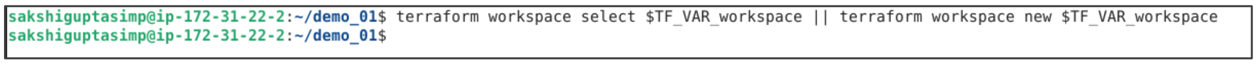
provider "aws" {
region = "us-west-2"
}



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

3.11 Set the workspace by using the following command:

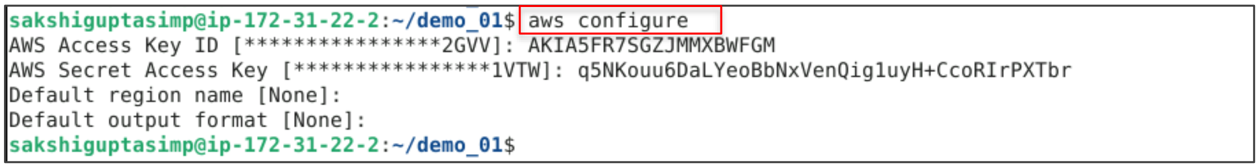
terraform workspace select $TF_VAR_workspace || terraform workspace new $TF_VAR_workspace



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform workspace select $TF_VAR_workspace || terraform workspace new $TF_VAR_workspace
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

3.12 Configure AWS credentials by using the following command:

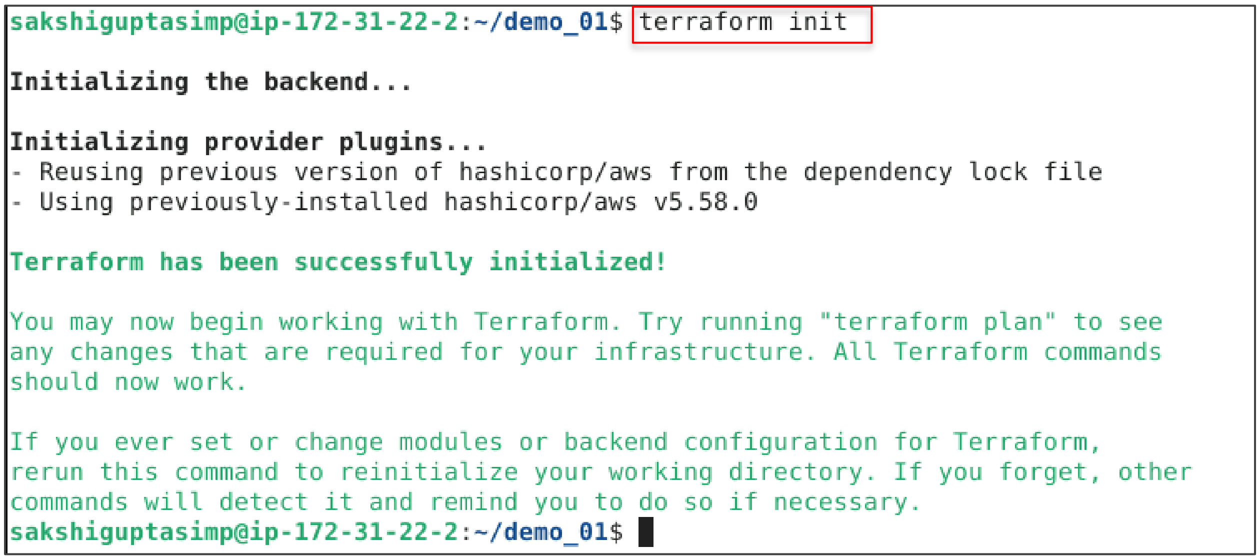
aws configure



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ aws configure
AWS Access Key ID [****************2GVV]: AKIA5FR7SGZJMMXBWFGM
AWS Secret Access Key [****************1VTW]: q5NKouu6DaLYeoBbNxVenQig1uyH+CcoRIrPXTbr
Default region name [None]:
Default output format [None]:
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

Note: Provide your AWS access key and secret key as shown

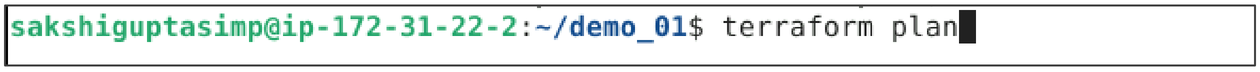
3.13 Initialize Terraform by using the following command:
terraform init



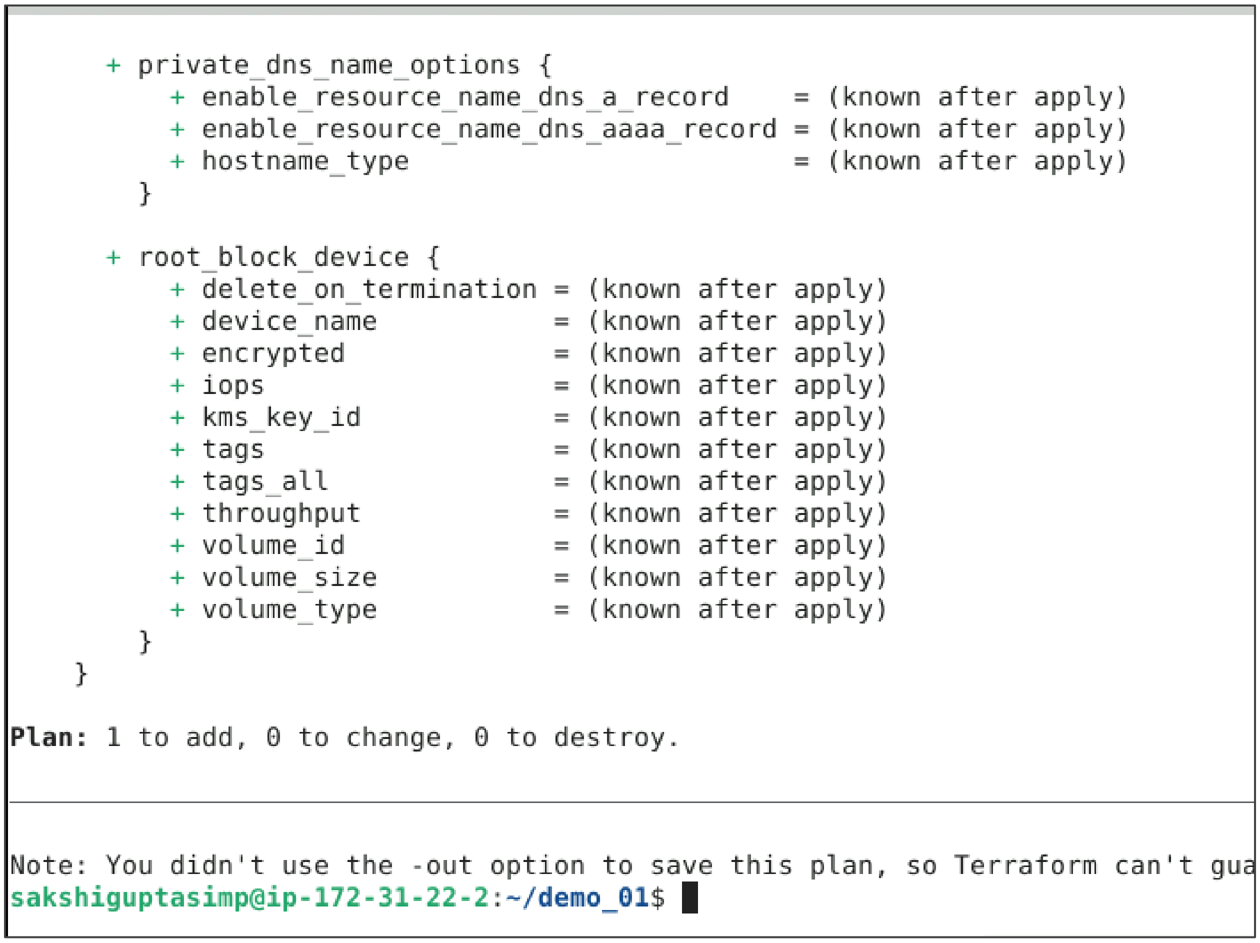
```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.58.0

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:~/demo_01$
```

Step 4: Plan and apply the configurations

4.1 Plan the configuration by using the following command:
terraform plan

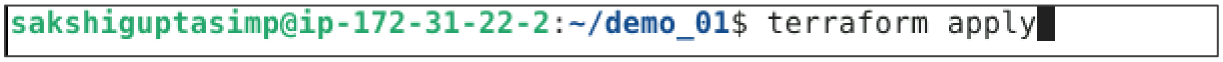


```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform plan
```



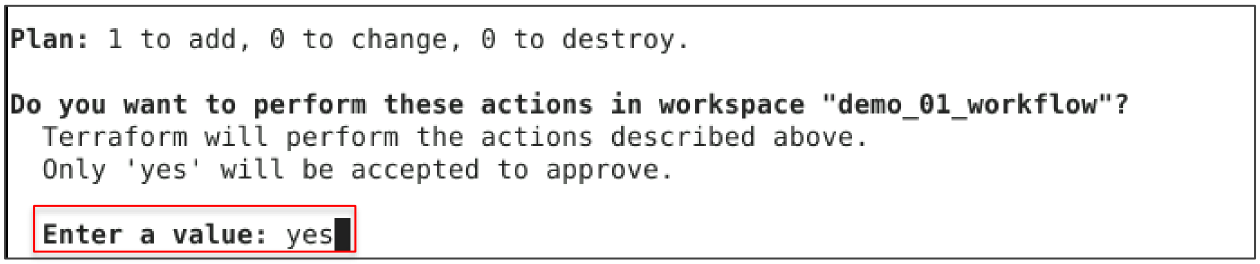
```
      + private_dns_name_options {
          + enable_resource_name_dns_a_record    = (known after apply)
          + enable_resource_name_dns_aaaa_record = (known after apply)
          + hostname_type                        = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + tags_all              = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't gua
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

4.2 Apply the configurations by using the following command:
terraform apply

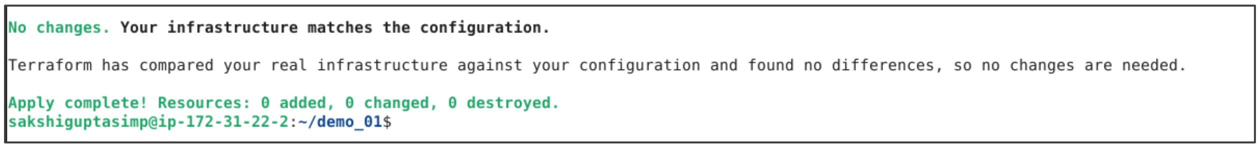


```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform apply
```

4.3 When prompted, approve the actions by typing **yes**



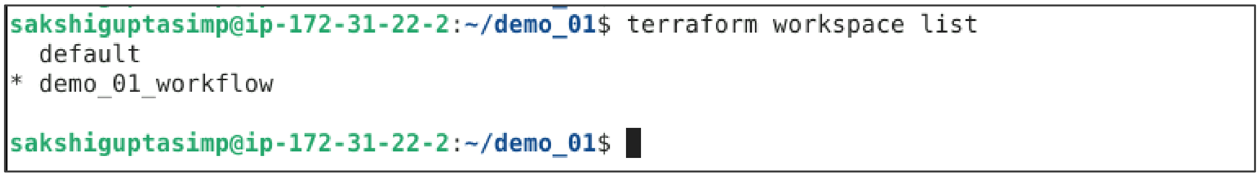
```
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions in workspace "demo_01_workflow"?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```



```
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

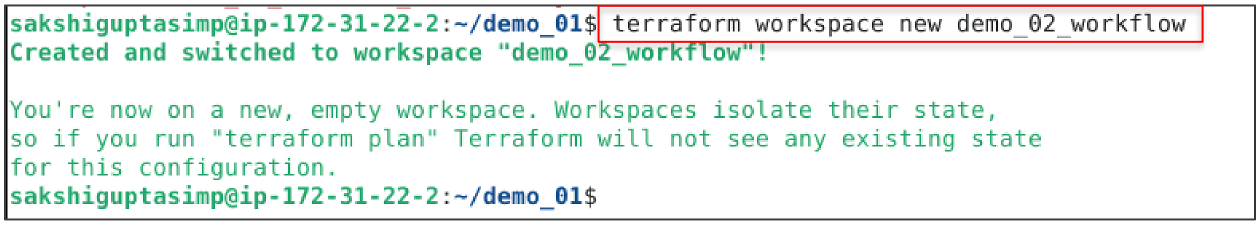
Step 5: Select and create workspaces

5.1 List all the workspaces using the following command:
terraform workspace list



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform workspace list
  default
* demo_01_workflow

sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

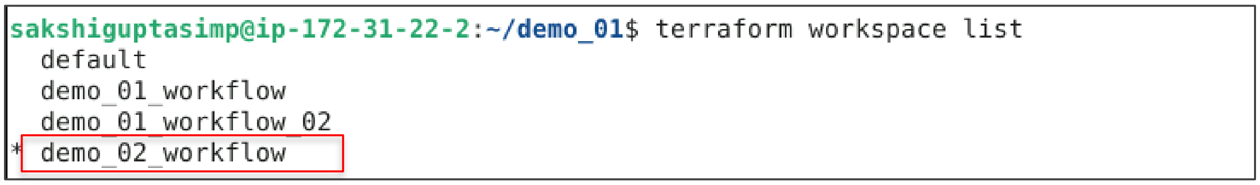
5.2 Create a new workspace, named **demo_02_workflow**, using the following command:
terraform workspace new demo_02_workflow



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform workspace new demo_02_workflow
Created and switched to workspace "demo_02_workflow"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

5.3 Verify the creation of the workspace by using the following command:

terraform workspace list

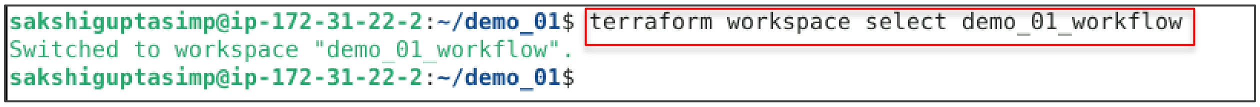


```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform workspace list
  default
  demo_01_workflow
  demo_01_workflow_02
* demo_02_workflow
```

The workspace is successfully created, as shown.

5.4 Switch to the workspace named **demo_01_workflow** using the following command:

terraform workspace select demo_01_workflow



```
sakshiguptasimp@ip-172-31-22-2:~/demo_01$ terraform workspace select demo_01_workflow
Switched to workspace "demo_01_workflow".
sakshiguptasimp@ip-172-31-22-2:~/demo_01$
```

By following the above steps, you have successfully used Terraform Cloud to manage infrastructure by creating and managing workspaces for deploying resources.