Name: Harsh Dalvi Subject: Advance DevOps , Sem: SEM V
Roll No: 13 Class / Batch: TE-IT / Batch B

Experiment No. 6

Aim: To Construct, change, and destroy AWS / GCP / Microsoft Azure / Digital Ocean infrastructure Using Terraform. **(LO1, LO3)**

Theory:

Terraform:

Terraform provides open-source infrastructure as code software for cloud service management with a consistent CLI workflow. This includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc. Terraform allows you to write, plan, and apply changes to deliver infrastructure as code. Terraform can manage both existing service providers and custom in-house solutions.

Terraform Cloud is an application that helps teams use Terraform together. It manages Terraform runs in a consistent and reliable environment, and includes easy access to shared state and secret data, access controls for approving changes to infrastructure, a private registry for sharing Terraform modules, detailed policy controls for governing the contents of Terraform configurations, and more.

Terraform Cloud is available as a hosted service at https://app.terraform.io. Small teams can sign up for free to connect Terraform to version control, share variables, run Terraform in a stable remote environment, and securely store remote state. Paid tiers allow you to add more than five users, create teams with different levels of permissions, enforce policies before creating infrastructure, and collaborate more effectively.

What is Terraform used for?

One of the main functions of Terraform is for public cloud provisioning on one of the major providers. Providing an IaC for services such as AWS and Azure has -- and will continue to be -- the main focus of Terraform. Terraform enables the use of these public clouds via a provider, a plugin that wraps existing APIs and languages like Azure Bicep, and creates Terraform syntax.

The second main use for Terraform is to facilitate multi-cloud deployments. One of the main draws of Terraform is that it performs across all cloud providers simultaneously, unlike some of its other IaC competitors. The capability to deploy resources into multiple cloud providers is critical because engineers can utilize the same syntax without familiarizing themselves with multiple tools and technologies.

The third most common use of Terraform is deploying, managing, and orchestrating resources with custom cloud providers. A provider is a way in Terraform to wrap an existing API and convert it to the Terraform declarative syntax, and this can be done even if you’re not using AWS or another one of the major cloud services. Providers can also be written for internal use cases where you may desire to convert existing tools or APIs into Terraform.

Name: Harsh Dalvi
Roll No: 13

Features of Terraform:

Here are some of the features of terraform:

1. **Infrastructure as Code:** IT professionals use Terraform's high-level configuration language (HCL) to describe the infrastructure in human-readable, declarative configuration files. Terraform lets you create a blueprint, a template that you can version, share, and re-use.

2. **Execution Plans:** Once the user describes the infrastructure, Terraform creates an execution plan. This plan describes what Terraform will do and asks for your approval before initiating any infrastructure changes. This step lets you review changes before Terraform does anything to the infrastructure, including creating, updating, or deleting it.

3. **Resource Graph:** Terraform generates a resource graph, creating or altering non-dependent resources in parallel. This graph enables Terraform to build resources as efficiently as possible while giving the users greater insight into their infrastructure.

4. **Change Automation:** Terraform can implement complex changesets to the infrastructure with virtually no human interaction. When users update the configuration files, Terraform figures out what has changed and creates an incremental execution plan that respects the dependencies.

5. **Flexible Workflows:** Run Terraform the way your team prefers. Execute runs from the CLI or a UI, your version control system, or integrate them into your existing workflows with an API.

6. **Managed Service:** It means that you do not have to manage the infrastructure, Terraform will take care of it.
   a. **Collaboration with the team**, i.e. multiple users in a team accessing the cloud which allows one of them to create the plan, the other one to review, commit, reject, or cancel it.

   b. **Remote state storage**, the state will be stored in Cloud which means the state is not going to be lost.

   c. **Version Control System**, Terraform Cloud provides easy integration with VCS such as GitHub.

7. **High Security:** It ensures that nobody can enter the Terraform cloud without proper credentials so the data is secure.

8. **Reliability:** Chances of data loss are less and the configurations are run efficiently.

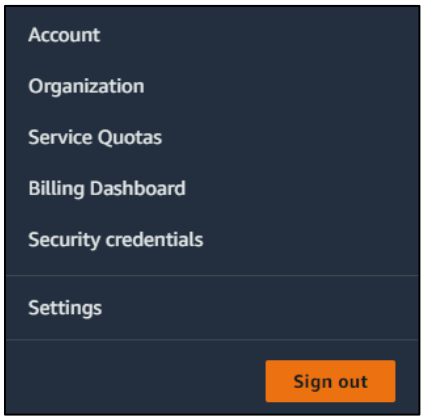
9. **Scalability:** It which means the capability of the system to handle heavy traffic when the number of users increases.

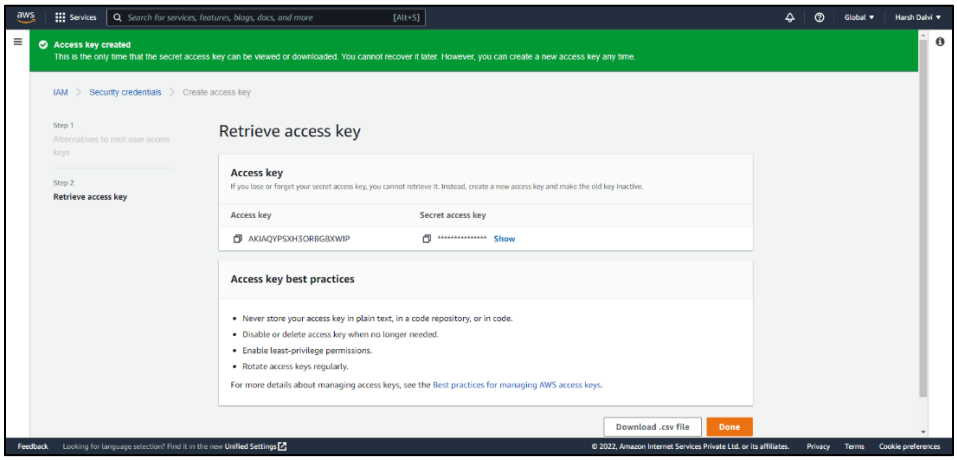
Name: Harsh Dalvi
Roll No: 13

Subject: Advance DevOps , Sem: SEM V
Class / Batch: TE-IT / Batch B

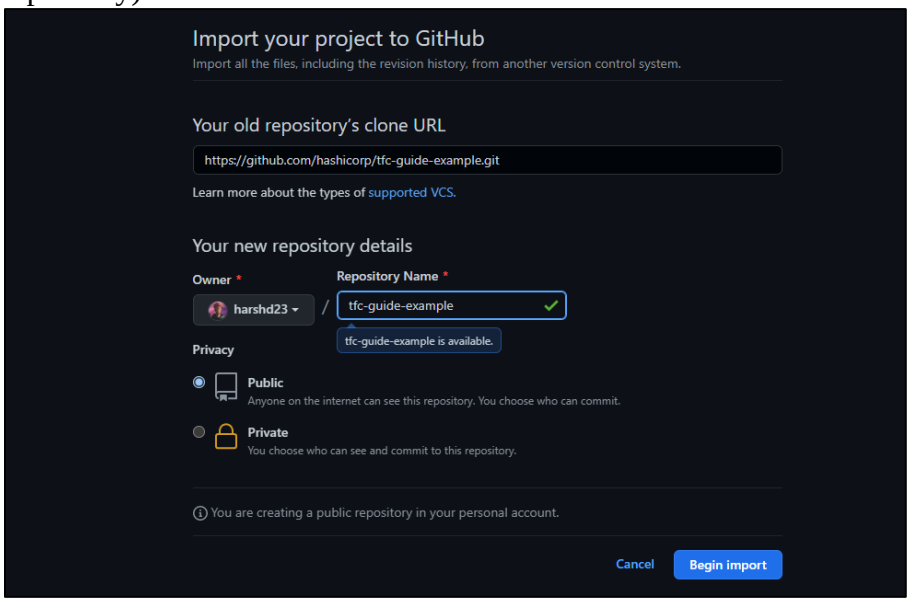
Steps to construct, change and destroy plans using terraform:

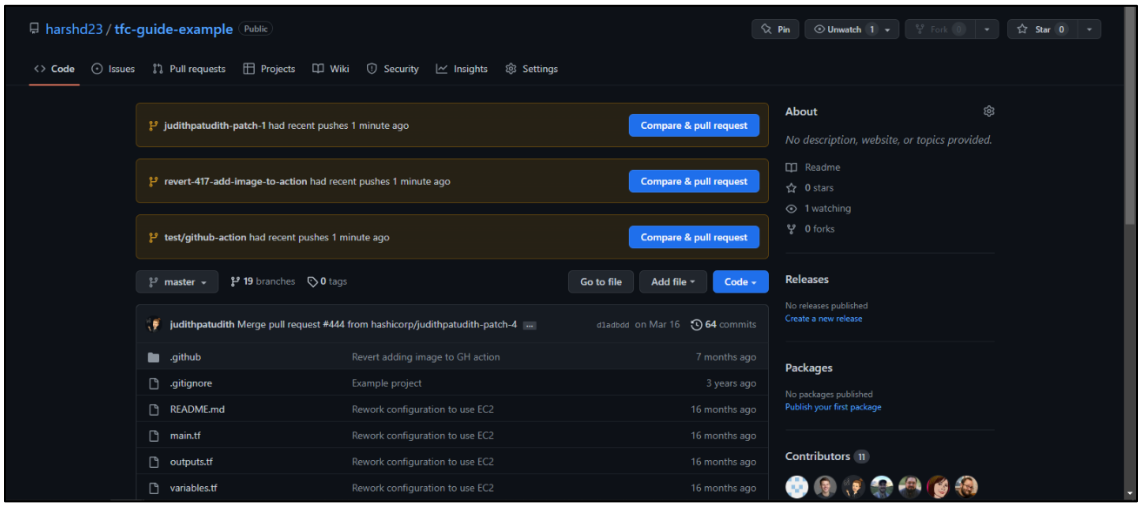
1. Download the Access Key from the AWS Account.



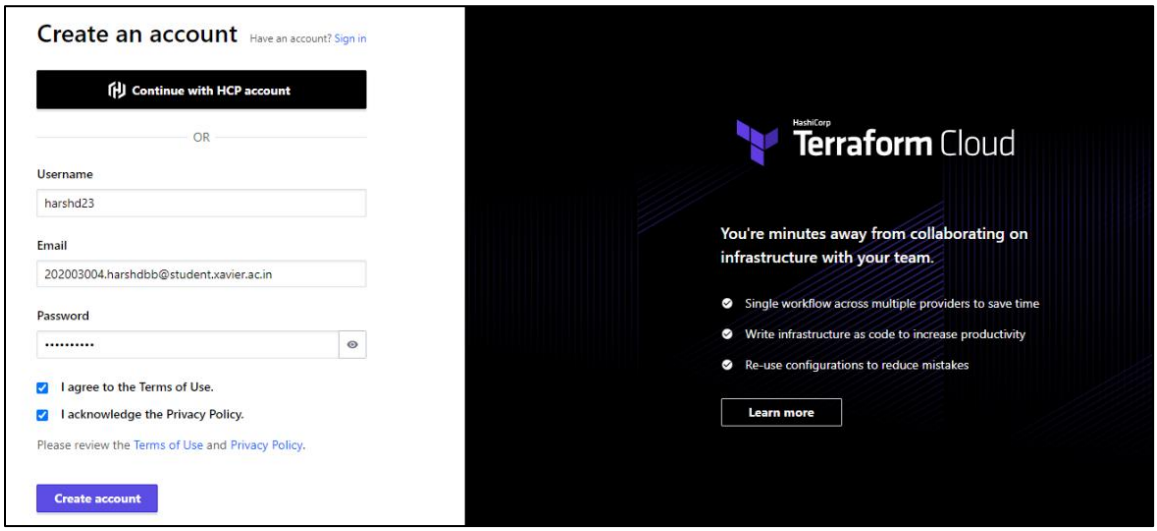


2. Import the example repository on your GitHub Account. (Eg. Hashicorp's tcf-guide-example repository)

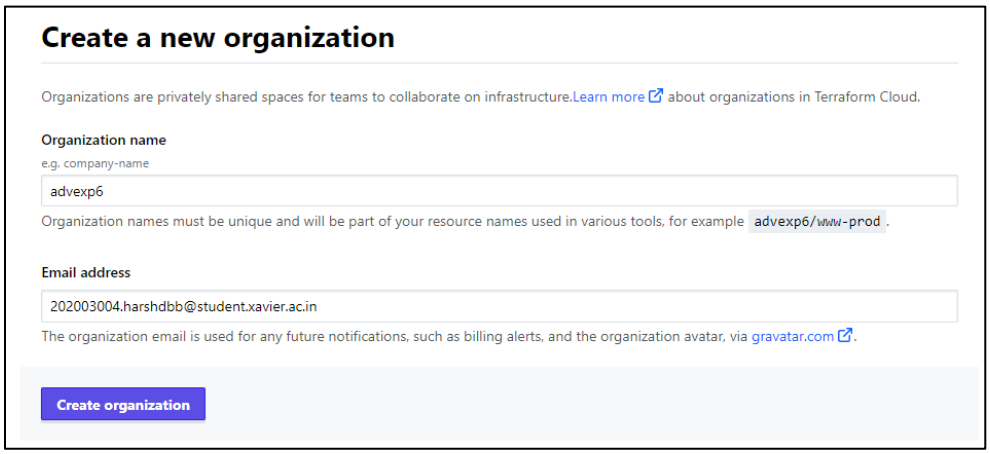




3. Create an account on Terraform Cloud.



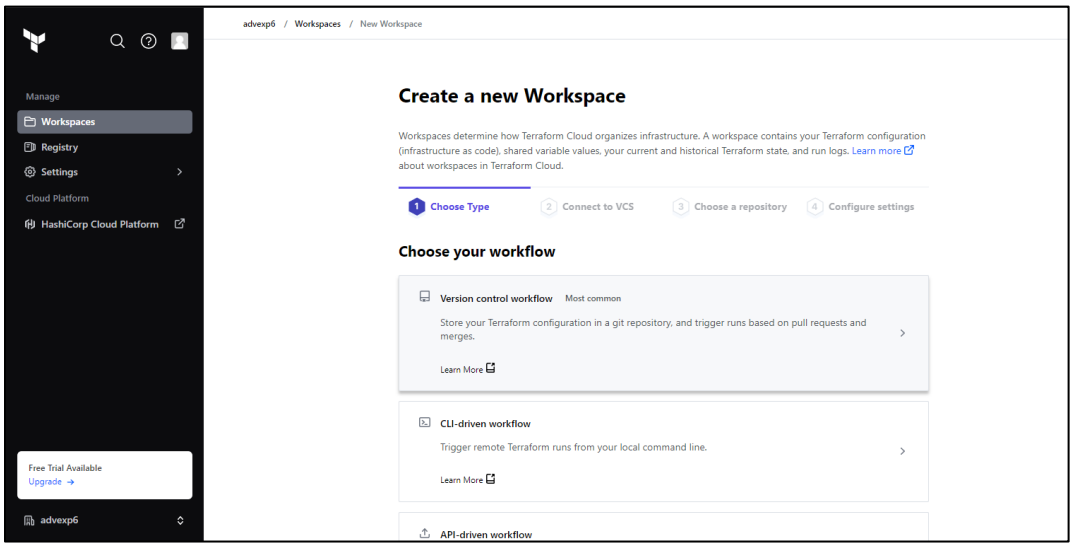
4. Create an organization.

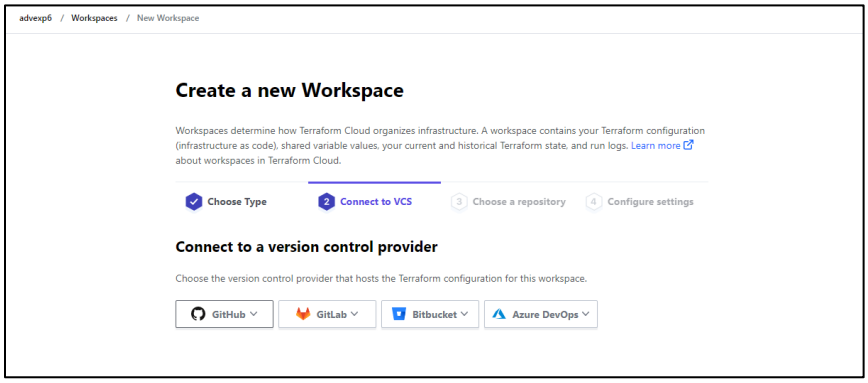


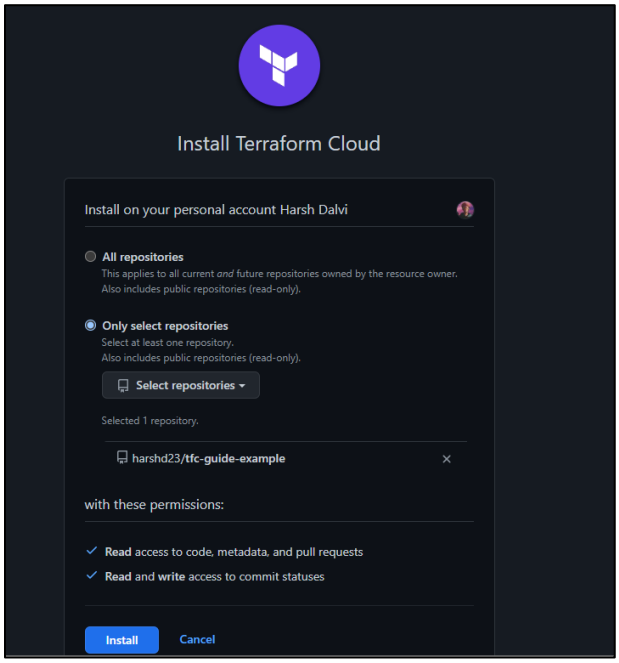
Name: Harsh Dalvi
Roll No: 13

Subject: Advance DevOps , Sem: SEM V
Class / Batch: TE-IT / Batch B

5. Create a Workspace wherein the imported repository in the GitHub account is selected.

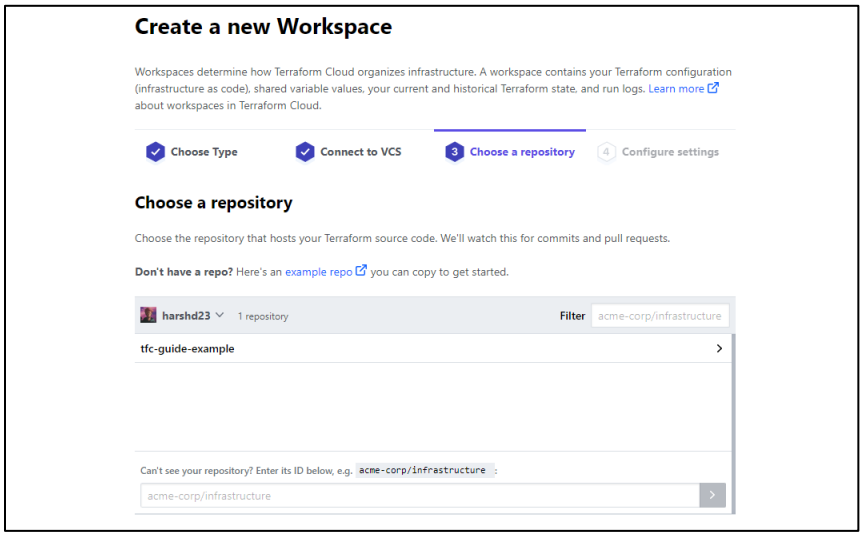


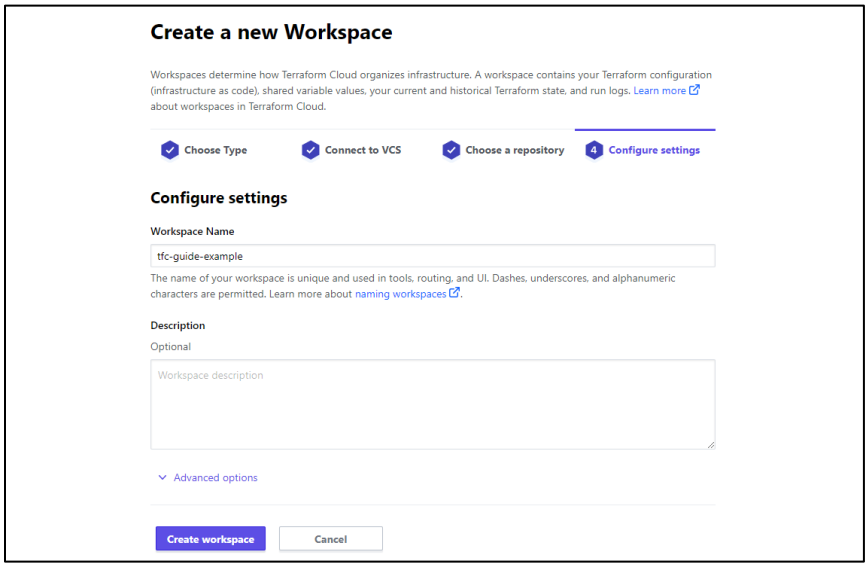


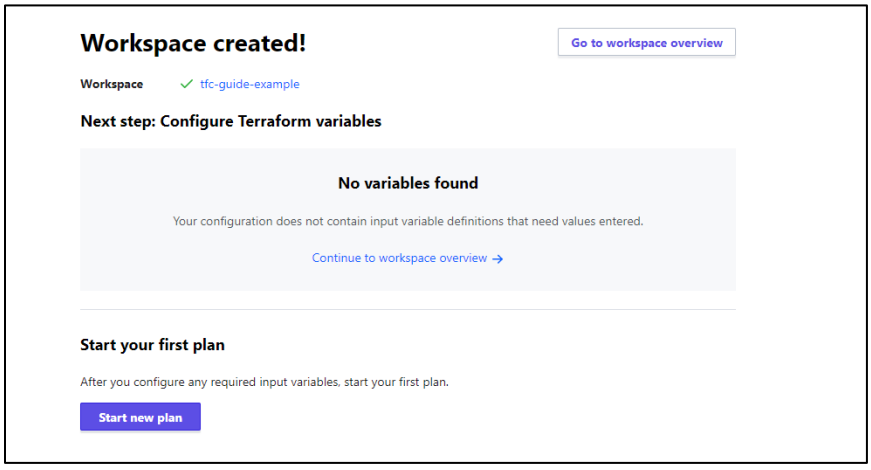


Name: Harsh Dalvi
Roll No: 13

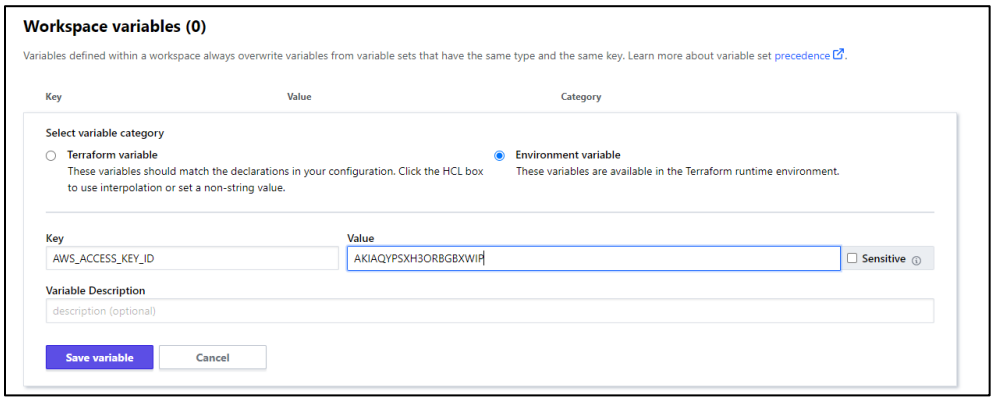
Subject: Advance DevOps , Sem: SEM V
Class / Batch: TE-IT / Batch B

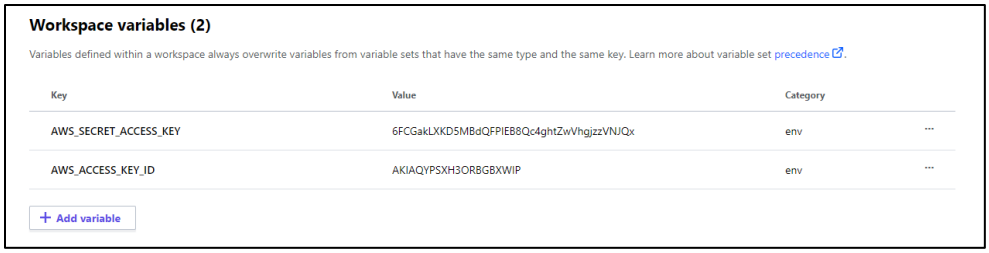




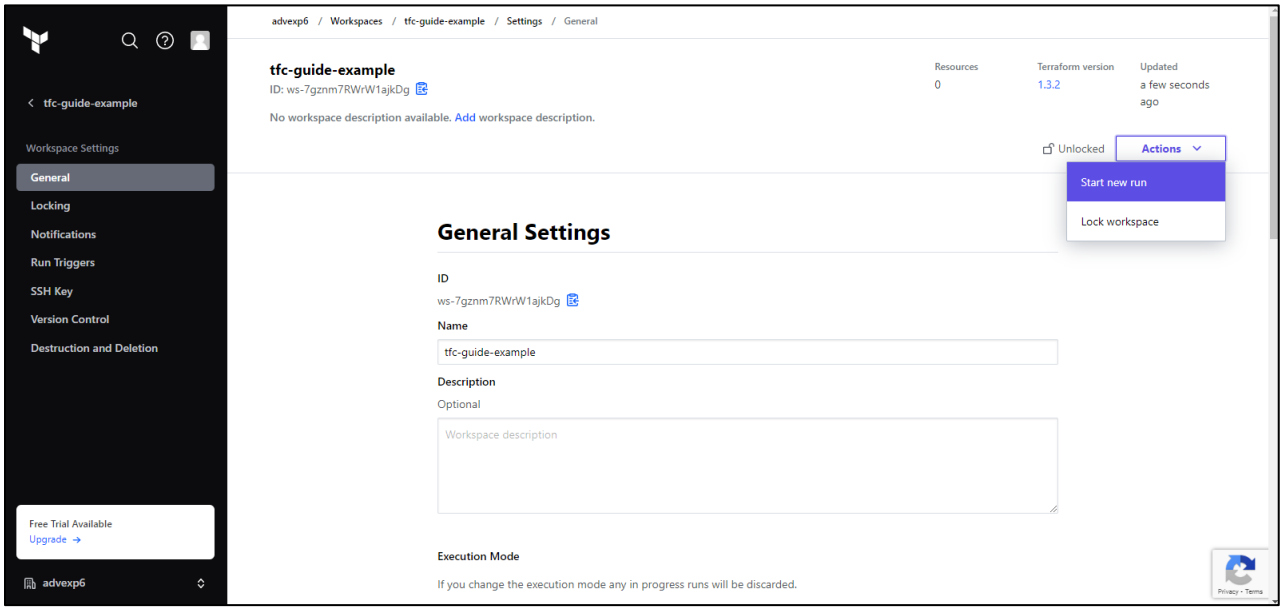


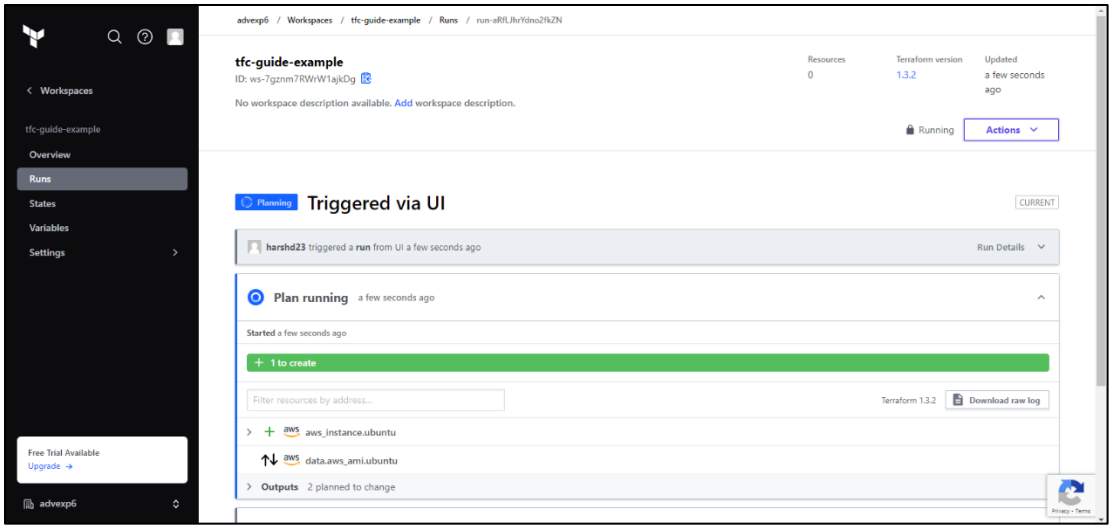
6. Once the Workspace is created go to variables and generate two variables as the access key and secret key and put in the values from the file downloaded when the access key from AWS Account was generated.

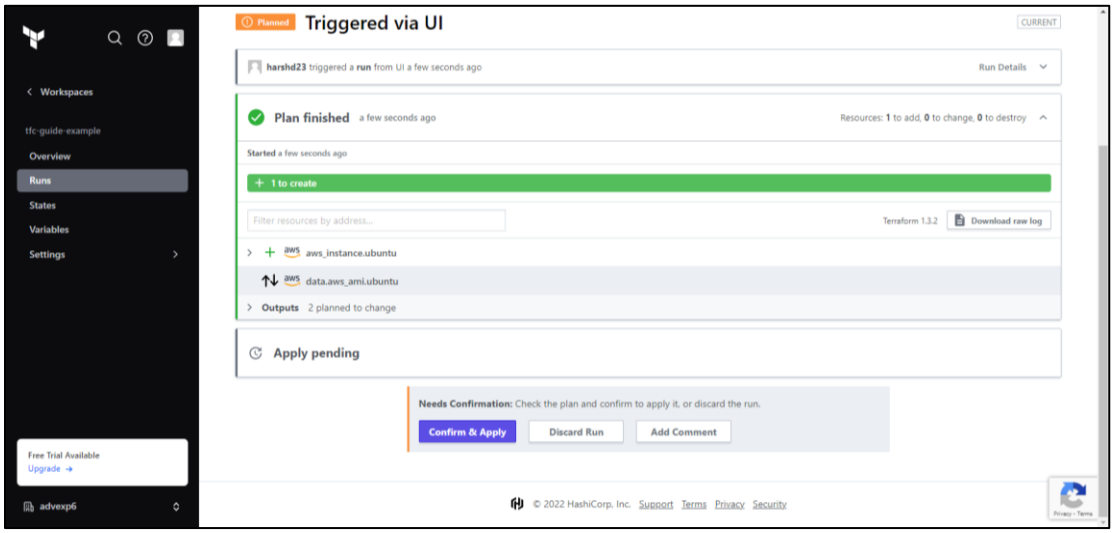




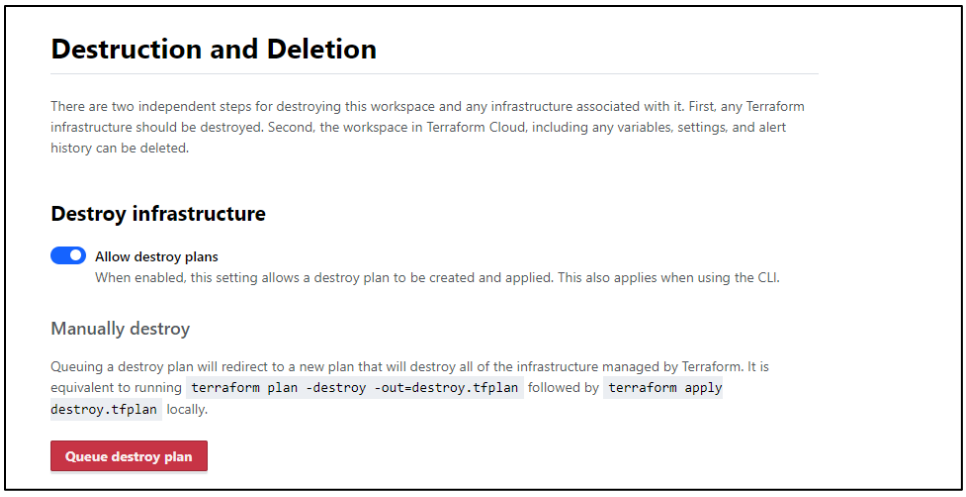
7. Now Start a new run where you will have to also apply the Plan.

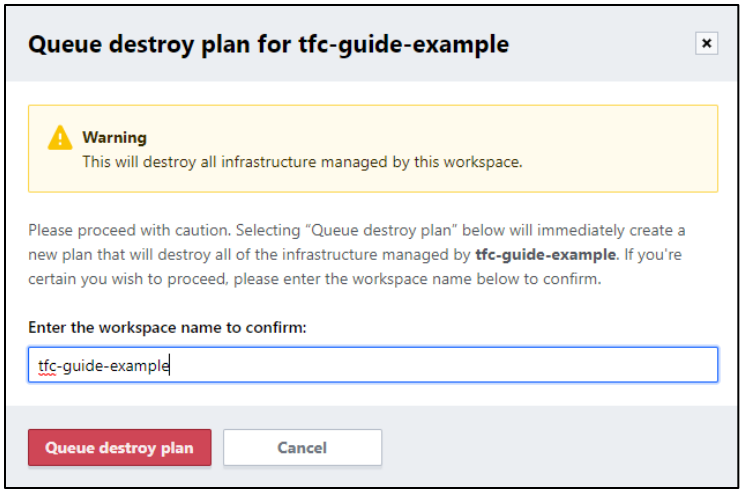


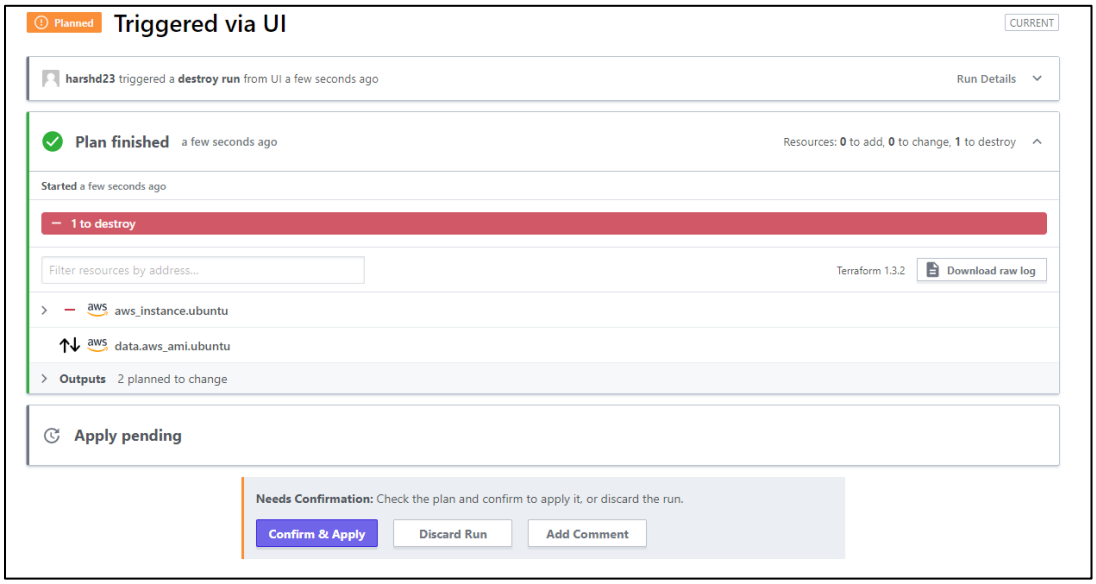


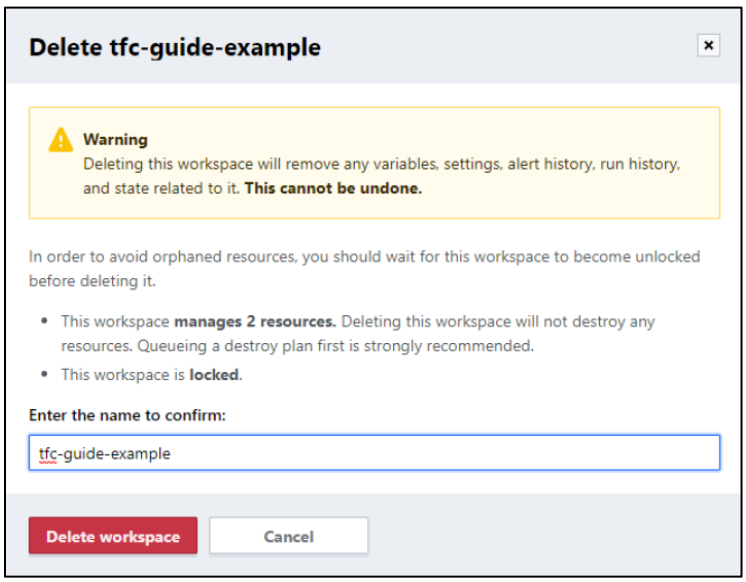


8. Now since the Plan has been created and applied it can now be destroyed along with the Workspace.



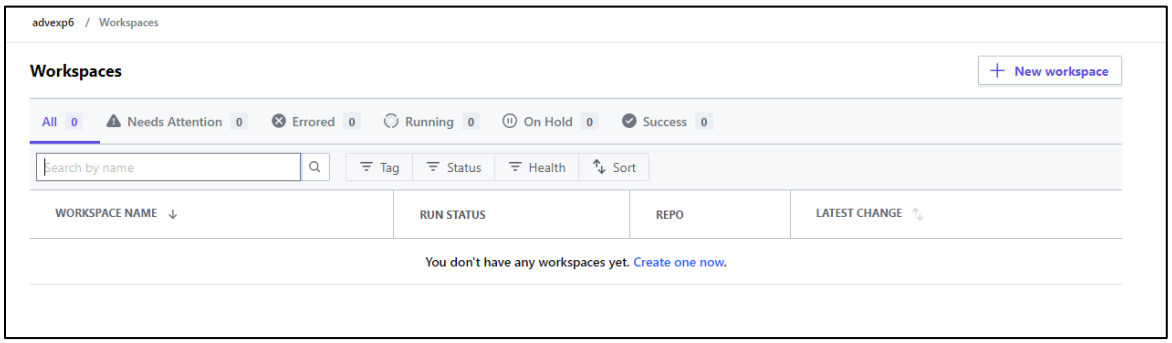

Queue destroy plan for tfc-guide-example
Warning
This will destroy all infrastructure managed by this workspace.
Please proceed with caution. Selecting "Queue destroy plan" below will immediately create a new plan that will destroy all of the infrastructure managed by tfc-guide-example. If you're certain you wish to proceed, please enter the workspace name below to confirm.
Enter the workspace name to confirm:
tfc-guide-example
Queue destroy plan
Cancel


Planned
Triggered via UI
CURRENT
harshd23 triggered a destroy run from UI a few seconds ago
Run Details
Plan finished
a few seconds ago
Resources: 0 to add, 0 to change, 1 to destroy
Started a few seconds ago
1 to destroy
Filter resources by address...
Terraform 1.3.2
Download raw log
aws_instance.ubuntu
data.aws_ami.ubuntu
Outputs
2 planned to change
Apply pending
Needs Confirmation: Check the plan and confirm to apply it, or discard the run.
Confirm & Apply
Discard Run
Add Comment


Delete tfc-guide-example
Warning
Deleting this workspace will remove any variables, settings, alert history, run history, and state related to it. This cannot be undone.
In order to avoid orphaned resources, you should wait for this workspace to become unlocked before deleting it.
This workspace manages 2 resources. Deleting this workspace will not destroy any resources. Queueing a destroy plan first is strongly recommended.
This workspace is locked.
Enter the name to confirm:
tfc-guide-example
Delete workspace
Cancel

Name: Harsh Dalvi
Roll No: 13

Subject: Advance DevOps , Sem: SEM V
Class / Batch: TE-IT / Batch B



Conclusion: From this experiment, we have studied about the concept of Terraform cloud. In this experiment, we created a repository for the terraform on Github and then forked the repository to Terraform workspace which we created. Then we run and applied plans on the workspace and destroyed it from the Terraform Cloud as we got the desired output. Hence, we have successfully achieved the Lab Outcome 1 and 3 (LO1 and LO3). Also, we have achieved PO1, PO2, PO3, PO4, PO5, PO9, PO10 and PO12 from this experiment.