

Terraform backend local to terraform cloud migration

A backend defines where Terraform stores its state data files.

Available Backends

By default, Terraform uses a backend called local, which stores state as a local file on disk.

To store your terraform backend in a remote location, you can use Terraform Cloud.

Steps:

1. Go to <https://www.terraform.io/>
2. Click on Try Terraform Cloud button.
3. Create an terraform cloud account

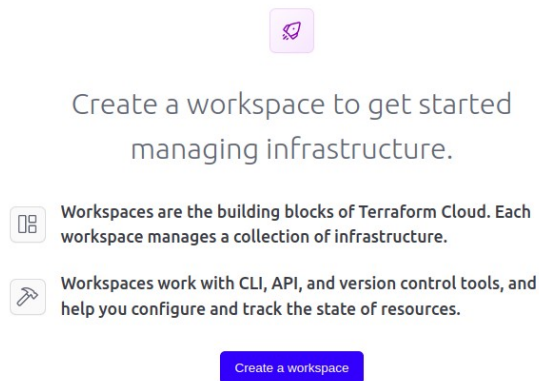
There are 3 ways to do it:

1. CLI-driven Run Workflow
2. API-driven Run Workflow
3. UI and VCS-driven Run Workflow

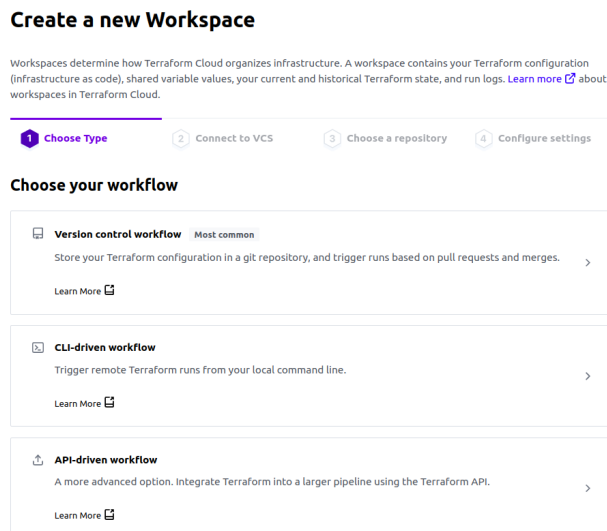
CLI-driven Run Workflow

STEPS:-

1. Create a workspace

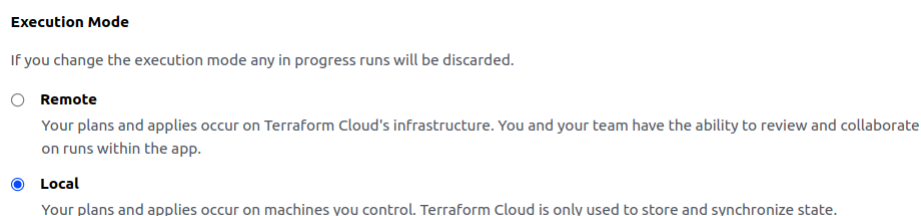


2. Choose workspace



3. Choose CLI-driven Workflow, give “Workspace Name” and “Description” and Create workspace.

4. Go to Workspace Settings -> General -> Execution Mode, select Local option so that terraform plan, apply and destroy can run from CLI only.



5. In your CLI, run “terraform login” command to login into your terraform cloud account, enter “yes”. It will redirect to Terraform cloud site in your default browser to create an API token.

6. Now enter the created token in your CLI, now you will be successfully logged-in into your terraform cloud account, and your token would be stored locally in “~/.terraform.d/credentials.tfrc.json” file

7. After creating the workspace, you will get a code of block to integrate your local terraform code with terraform cloud workspace, copy-paste this block in your main.tf file.

```
terraform {
  cloud {
    organization = "nikhil-org-001"

    workspaces {
      name = "networking-test-us-east"
    }
  }
}
```

8. Run “terraform init” command to initialize modules, initialize remote backend and initialize provider plugins.

```
admin1@system1:~/terraform-cloud/cli-driven/playing-with-terraform$ terraform init
Initializing modules...
- ec2 in module/ec2
- sg in module/sg
- vpc in module/vpc

Initializing Terraform Cloud...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.46.0...
- Installed hashicorp/aws v4.46.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform Cloud has been successfully initialized!
```

9. Now, run “terraform plan” to dry run your code, “terraform apply” command to apply your terraform code and enter “yes” to confirm.

After completion of your apply, you can go to your workspace in terraform cloud site and check your terraform backend there.

The screenshot shows the Terraform Cloud interface for a workspace named "networking-test-us-east". The workspace ID is "ws-FCrJVfnkVBHQycqz". It shows 7 resources and is using Terraform version 1.3.6. A recent state file is listed with ID "#sv-f211kzEspS53Cm5Y", triggered by "nikhildobriyal" 31 minutes ago. The workspace is currently "Unlocked".

Resources	Terraform version	Updated
7	1.3.6	a few seconds ago

networking-test-us-east
ID: ws-FCrJVfnkVBHQycqz

No workspace description available. [Add workspace description.](#)

Unlocked [Actions](#)

New state #sv-f211kzEspS53Cm5Y
nikhildobriyal triggered from Terraform 31 minutes ago

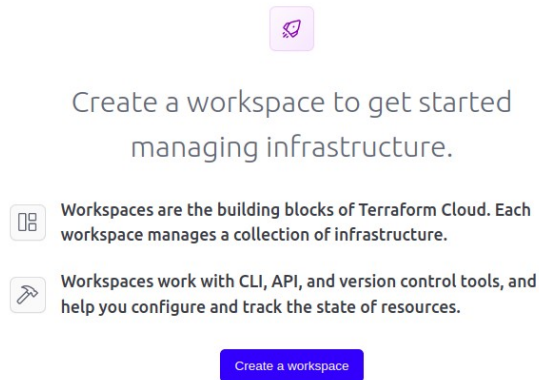
Now, you can see your state is stored there in your workspace, number of resources are 7 currently.

10. Just like you have applied your code from CLI, you can destroy your terraform infra by running “terraform destroy” command and enter “yes” for confirmation.

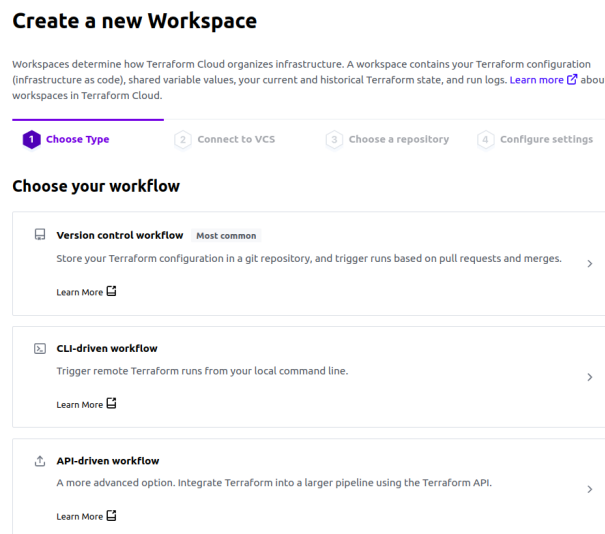
API-driven Run Workflow

STEPS:-

1. Create a workspace



2. Choose your workflow.



3. Choose API-driven Workflow, give “Workspace Name” and “Description” and Create workspace.

4. Go to Workspace Settings -> General -> Execution Mode, select Local option so that terraform plan, apply and destroy can run from your CI/CD pipeline.

Execution Mode

If you change the execution mode any in progress runs will be discarded.

☐ Remote

Your plans and applies occur on Terraform Cloud's infrastructure. You and your team have the ability to review and collaborate on runs within the app.

☒ Local

Your plans and applies occur on machines you control. Terraform Cloud is only used to store and synchronize state.

5. Write your CI/CD pipeline for terraform init, apply and destroy your AWS infra.

Here, we are using GitHub Actions for our CI/CD pipeline, create a directory `.github/workflows` in your Terraform code's root and create a `<file-name>.yaml` file where you will have to write CI/CD pipeline.

First, give a name for workflow

```
name: Create AWS Infra
```

Now, give the event when this pipeline will trigger.

```
on:
  push:
    branches: [master]
```

Now, define the pipeline's job name, runner(where the pipeline will runs on) and environment variables for your pipeline(which user has to define in your GitHub repo's secrets).

```
jobs:
  job1:
    runs-on: ubuntu-latest
    env:
      AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
      AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
      TF_API_TOKEN: ${ secrets.TF_API_TOKEN }
```

Now, define steps of your job in pipeline

```
steps:
  - name: Checkout Codebase
    uses: actions/checkout@v2

  - name: Setup Terraform
    uses: hashicorp/setup-terraform@v1
    with:
      # terraform_version: 0.13.0
      cli_config_credentials_token: ${ secrets.TF_API_TOKEN }
```

The first step is for checking out the codebase(which is terraform code of our repo) and the second step would sets up Terraform CLI in your GitHub Actions workflow

```
- name: Terraform Initialize
  run: terraform init

- name: Terraform apply
  run: terraform apply -auto-approve
```

Third step is for running “terraform init” command in runner(which will initialize modules, initialize remote backend and initialize provider plugins) and the fourth step is for terraform apply command(which will deploy infra through CI/CD).

```
.github > workflows > ! pipeline.yaml
1  name: Create AWS Infra
2  on:
3    push:
4      branches: [master]
5  jobs:
6    job1:
7      runs-on: ubuntu-latest
8      env:
9        AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
10       AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
11       TF_API_TOKEN: ${ secrets.TF_API_TOKEN }
12     steps:
13       - name: Checkout Codebase
14         uses: actions/checkout@v2
15       - name: Setup Terraform
16         uses: hashicorp/setup-terraform@v1
17         with:
18           # terraform_version: 0.13.0
19           cli_config_credentials_token: ${ secrets.TF_API_TOKEN }
20       - name: Terraform Initialize
21         run: terraform init
22       - name: Terraform apply
23         run: terraform apply -auto-approve
```

6. After creating the workspace, you will get a code of block to integrate your local terraform code with terraform cloud workspace, copy-paste this block in your main.tf file.

```
terraform {
  cloud {
    organization = "nikhil-org-001"

    workspaces {
      name = "networking-test-us-east"
    }
  }
}
```

7. Now, go to Workspace Settings -> Version Control

Version Control

Not Connected

Connecting the workspace to Version Control will enable automatic runs on git commits, automatic plan-only runs on pull requests and status updates. Read more about [connecting VCS providers to Terraform Cloud](#).

[Connect to version control](#)

And click on Connect to version control, choose Version Control -> click on GitHub, select your repository and then select “Auto apply” in apply method and save the settings.

Apply Method

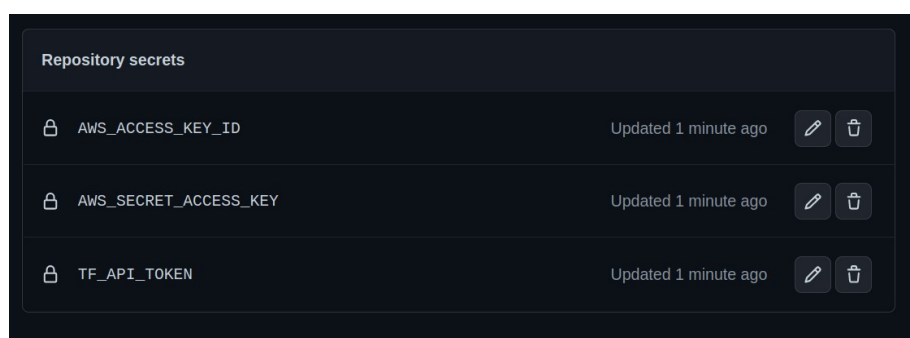
☒ **Auto apply**

Automatically apply changes when a Terraform plan is successful. Plans that have no changes will not be applied. If this workspace is linked to version control, a push to the default branch of the linked repository will trigger a plan and apply.

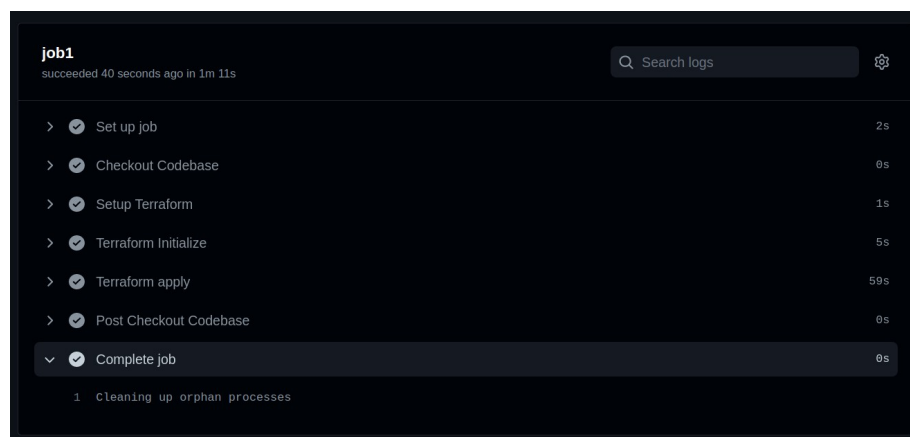
☐ **Manual apply**

Require an operator to confirm the result of the Terraform plan before applying. If this workspace is linked to version control, a push to the default branch of the linked repository will only trigger a plan and then wait for confirmation.

8. Define your AWS account credentials and terraform account token in your repo’s secrets.



9. Push your code into your master branch, which will trigger your pipeline.



As soon as you will push your code, your pipeline will trigger and jobs will be scheduled. Thus, your infra will be deployed and state will be stored in terraform cloud workspace.



Now, you can see your state is stored there in your workspace, number of resources are 7 currently.

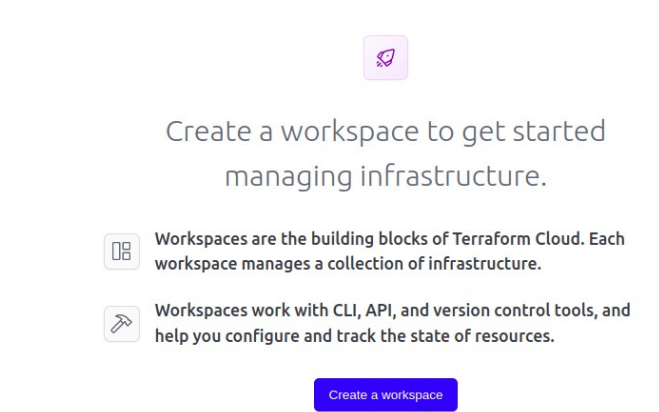
9. Just like you have applied your code from CI/CD, you can destroy your terraform infra by creating another pipeline for destroying the infra which will trigger on manually dispatch the workflow.

```
.github > workflows > ! destroy.yaml
1  name: Destroy AWS Infra
2  on:
3    workflow_dispatch:
4  jobs:
5    job1:
6      runs-on: ubuntu-latest
7      env:
8        AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
9        AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
10       TF_API_TOKEN: ${ secrets.TF_API_TOKEN }
11     steps:
12       - name: Checkout Codebase
13         uses: actions/checkout@v2
14       - name: Setup Terraform
15         uses: hashicorp/setup-terraform@v1
16         with:
17           # terraform_version: 0.13.0
18           cli_config_credentials_token: ${ secrets.TF_API_TOKEN }
19       - name: Terraform Initialize
20         run: terraform init
21       - name: Terraform destroy
22         run: terraform destroy -auto-approve
```

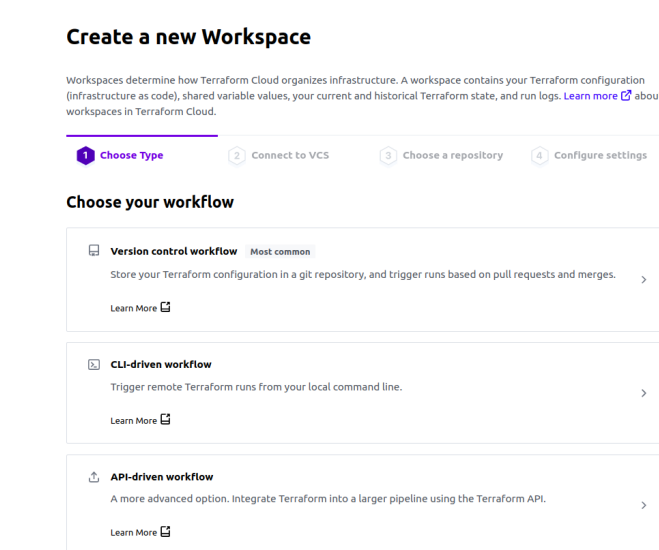

UI and VCS-driven Run Workflow

STEPS:-

1. Create a workspace



2. Choose your workflow.



3. Choose Version control workflow, give “Workspace Name” and “Description” and Create workspace.

4. Go to Workspace Settings -> General -> Execution Mode, select Remote option so that terraform plan, apply and destroy can run from your terraform cloud account only.

Execution Mode

If you change the execution mode any in progress runs will be discarded.

☒ **Remote**

Your plans and applies occur on Terraform Cloud's infrastructure. You and your team have the ability to review and collaborate on runs within the app.

☐ **Local**

Your plans and applies occur on machines you control. Terraform Cloud is only used to store and synchronize state.

5. After creating the workspace, you will get a code of block to integrate your local terraform code with terraform cloud workspace, copy-paste this block in your main.tf file.

```
terraform {
  cloud {
    organization = "nikhil-org-001"

    workspaces {
      name = "networking-test-us-east"
    }
  }
}
```

6. Now, push your code to a Github repository.

7. Configure your terraform workspace, go to Workspace Settings -> Version Control

Version Control

Not Connected

Connecting the workspace to Version Control will enable automatic runs on git commits, automatic plan-only runs on pull requests and status updates. Read more about [connecting VCS providers to Terraform Cloud](#).

[Connect to version control](#)

And click on Connect to version control, choose Version Control -> click on GitHub, select your repository and then select “Auto apply” in apply method and save the settings.

Apply Method

☒ **Auto apply**

Automatically apply changes when a Terraform plan is successful. Plans that have no changes will not be applied. If this workspace is linked to version control, a push to the default branch of the linked repository will trigger a plan and apply.

☐ **Manual apply**

Require an operator to confirm the result of the Terraform plan before applying. If this workspace is linked to version control, a push to the default branch of the linked repository will only trigger a plan and then wait for confirmation.

8. Define your AWS account credentials in your workspace’s Variables section.

Workspace variables (2)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set [precedence](#).

Key	Value	Category
AWS_SECRET_ACCESS_KEY SENSITIVE	Sensitive - write only	env
AWS_ACCESS_KEY_ID SENSITIVE	Sensitive - write only	env

9. Go to home page of your workspace, click on “Actions” button and click on “Start new run”

networking-test-us-east

ID: ws-FCrJVFnkVbHQycq

No workspace description available. [Add workspace description.](#)

Unlocked

Resources
0

Terraform version
1.3.6

Updated
a minute ago

Latest Run [View all runs](#)

[nikhil1828](#)
Execution

Actions

Start new run

Lock workspace

10. Give a reason to start your run and choose run type “Plan and apply (standard)” and click on “Start run” button.

Start a new run

This plan will be automatically applied if checks pass, because "Auto apply" is enabled in this workspace.

Reason for starting run

apply through UI

Choose run type

Plan and apply (standard)

Start run

Cancel

apply through UI

CURRENT

Planning

Plan duration

Resources to be changed

Less than a minute

nikhildobriyal triggered a run from UI a few seconds ago

Run Details

Plan running

a few seconds ago

Started a few seconds ago

Terraform 1.3.6

Download raw log

Waiting for resources...

Apply pending

Plan is running

Plan finished

a few seconds ago

Resources: 7 to add, 0 to change, 0 to destroy

Started a minute ago

Finished a few seconds ago

+ 7 to create

Filter resources by address...

Terraform 1.3.6

Download raw log

> + aws module.ec2.aws_instance.web["ec2-001"]

> + aws module.sg.aws_security_group.allow_tls["ec2-sg"]

> + aws module.vpc.aws_internet_gateway.igw

> + aws module.vpc.aws_route_table_association.pbsnet_assoc["snet-pb-1"]

> + aws module.vpc.aws_route_table.pub_rt

> + aws module.vpc.aws_subnet.pub-snet["snet-pb-1"]

> + aws module.vpc.aws_vpc.vpc

> Outputs 6 planned to change

Download Sentinel mocks

Sentinel mocks can be used for testing your Sentinel policies

Plan is finished

Apply running
a few seconds ago

Started a few seconds ago

1 applying...

+ 6 created

Terraform 1.3.6
Download raw log

> + module.ec2.aws_instance.web["ec2-001"]	<div>Creating</div>
> + module.sg.aws_security_group.allow_tls["ec2-sg"]	<div>Created id=sg-0ec436a7e106d3ee4</div>
> + module.vpc.aws_internet_gateway.igw	<div>Created id=igw-06890e1453cdc4fa7</div>
> + module.vpc.aws_route_table_association.pbsnet_assoc["snet-pb-1"]	<div>Created id=rtbassoc-0a34e026102227601</div>
> + module.vpc.aws_route_table.pub_rt	<div>Created id=rtb-0a23474e0f5e89a2c</div>
> + module.vpc.aws_subnet.pub-snet["snet-pb-1"]	<div>Created id=subnet-0d204d82d481d3e05</div>
> + module.vpc.aws_vpc.vpc	<div>Created id=vpc-0d4be82ac86122a43</div>

Apply is running

Apply finished
a few seconds ago
Resources: 7 added, 0 changed, 0 destroyed

Started a minute ago > Finished a few seconds ago

+ 7 created

Terraform 1.3.6
Download raw log

> + module.ec2.aws_instance.web["ec2-001"]	<div>Created id=i-032909cc53304fc87</div>
> + module.sg.aws_security_group.allow_tls["ec2-sg"]	<div>Created id=sg-0ec436a7e106d3ee4</div>
> + module.vpc.aws_internet_gateway.igw	<div>Created id=igw-06890e1453cdc4fa7</div>
> + module.vpc.aws_route_table_association.pbsnet_assoc["snet-pb-1"]	<div>Created id=rtbassoc-0a34e026102227601</div>
> + module.vpc.aws_route_table.pub_rt	<div>Created id=rtb-0a23474e0f5e89a2c</div>
> + module.vpc.aws_subnet.pub-snet["snet-pb-1"]	<div>Created id=subnet-0d204d82d481d3e05</div>
> + module.vpc.aws_vpc.vpc	<div>Created id=vpc-0d4be82ac86122a43</div>

Apply is finished

Now, you can see your state is stored there in your workspace, number of resources are 7 currently.

networking-test-us-east
 ID: ws-FCrJVFnkVbHQyczq

Resources
7

Terraform version
1.3.6

Updated
5 minutes ago

No workspace description available. [Add workspace description.](#)

Unlocked

Actions

apply through UI

#sv-gXDecwUjNbk0EmJ4 | nikhildobriyal triggered from Terraform 5 minutes ago | #run-aM9gUXC9km2eAPiq | 3ec6e84

11. Through UI, you can also delete your infra, Go to workspace settings -> Destruction and Deletion, click on “Queue destroy plan”.

Destruction and Deletion

There are two independent steps for destroying this workspace and any infrastructure associated with it. First, any Terraform infrastructure should be destroyed. Second, the workspace in Terraform Cloud, including any variables, settings, and alert history can be deleted.

Destroy infrastructure

☒ Allow destroy plans

When enabled, this setting allows a destroy plan to be created and applied. This also applies when using the CLI.


Manually destroy

Queuing a destroy plan will redirect to a new plan that will destroy all of the infrastructure managed by Terraform. It is equivalent to running `terraform plan -destroy -out=destroy.tfplan` followed by `terraform apply destroy.tfplan` locally.

[Queue destroy plan](#)

12. Give workspace name for confirmation and destroy the infra.

Queue destroy plan for networking-test-us-east ✕

 **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 **networking-test-us-east**. If you're certain you wish to proceed, please enter the workspace name below to confirm.

Enter the workspace name to confirm:

[Queue destroy plan](#) [Cancel](#)

 **Apply finished** a minute ago

Resources: 0 added, 0 changed, **7 destroyed** ⌵

Started 2 minutes ago > Finished a minute ago

7 destroyed

Filter resources by address...

Terraform 1.3.6 [Download raw log](#)

>  module.ec2.aws_instance.web["ec2-001"]	✓ Deleted <code>id=i-032909cc53304fc87</code>
>  module.sg.aws_security_group.allow_tls["ec2-sg"]	✓ Deleted <code>id=sg-0ec436a7e106d3ee4</code>
>  module.vpc.aws_internet_gateway.lgw	✓ Deleted <code>id=igw-06890e1453cdc4fa7</code>
>  module.vpc.aws_route_table_association.pbsnet_assoc["snet-pb-1"]	✓ Deleted <code>id=rtbassoc-0a34e026102227601</code>
>  module.vpc.aws_route_table.pub_rt	✓ Deleted <code>id=rtb-0a23474e0f5e89a2c</code>
>  module.vpc.aws_subnet.pub-snet["snet-pb-1"]	✓ Deleted <code>id=subnet-0d204d82d481d3e05</code>
>  module.vpc.aws_vpc.vpc	✓ Deleted <code>id=vpc-0d4be82ac86122a43</code>

State versions created:
[nikhil-org-001/networking-test-us-east#sv-b26aQUsVoJaWtD6H](#) (Dec 11, 2022 20:13:38 pm)

Destroy done