HashiCorp
Terraform

Importing Resources and State into Terraform Enterprise

May 2022

Terraform Enterprise Path to Production

Week 1-3

Terraform Instance Deployed

- Kickoff, Architectural Deep-dive
- Migration of State files to TFE
- Office Hours

Week 4

Terraform Workflow

- Terraform Workspaces Variables, Modules, Git-Repo

Week 5-7

Monitoring & Lifecycle Management

- Lifecycle Management

Week 8

Terraform Governance

- Terraform Sentinel, Run Tasks, RBAC

Week 9

Ready for Production!

- Exit ramp and operational readiness

Agenda

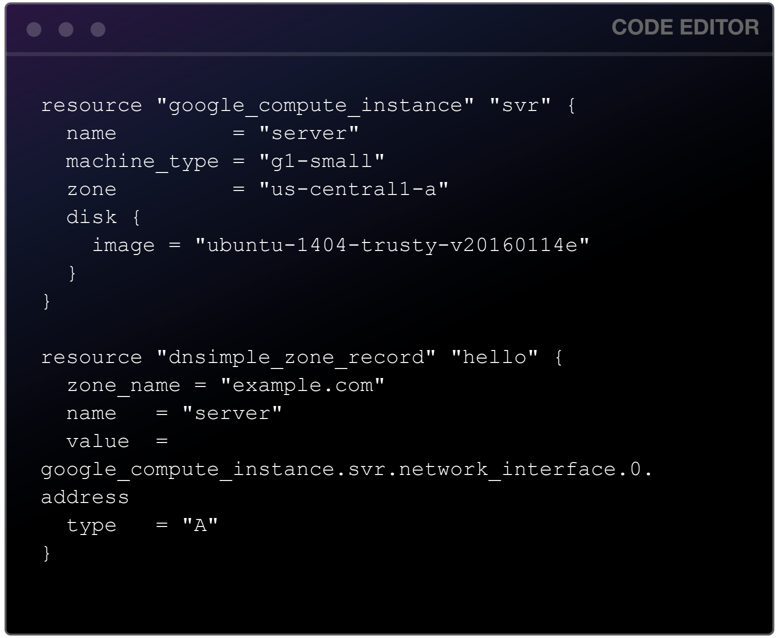
- Terraform Basics
- Importing existing infrastructure
- Migrating from OSS to Enterprise
- Next steps
- Q & A

The Basics
How Terraform Works

Infrastructure as Code

With HashiCorp Configuration Language (HCL), infrastructure and services from any provider can be provisioned in a codified, secure, and automated fashion.

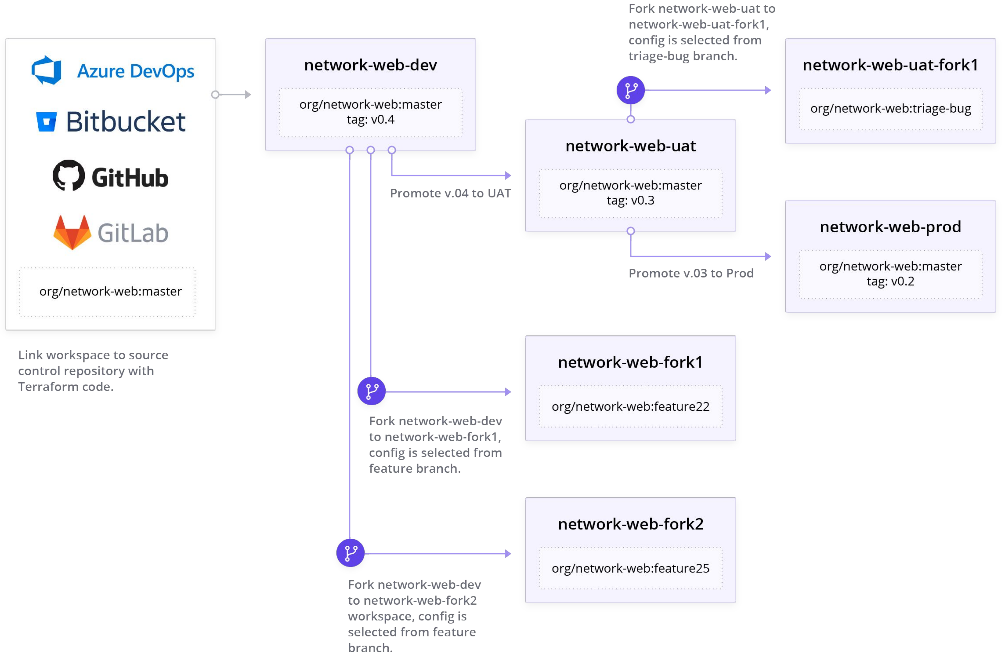
- HashiCorp Configuration Language (HCL) is human readable
  and machine executable
- Declarative, Turing-Complete language
- Used to automate, version, and collaborate on infrastructure



```
resource "google_compute_instance" "svr" {
  name         = "server"
  machine_type = "g1-small"
  zone         = "us-central1-a"
  disk {
    image = "ubuntu-1404-trusty-v20160114e"
  }
}

resource "dnsimple_zone_record" "hello" {
  zone_name = "example.com"
  name   = "server"
  value  =
google_compute_instance.svr.network_interface.0.
address
  type   = "A"
}
```

Benefits of Infrastructure as Code

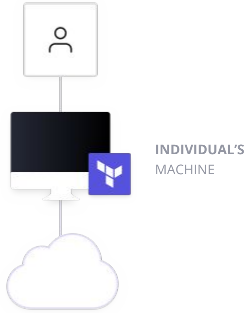
- Versioning
- Collaboration
- Promotion
- Forking


Azure DevOps
Bitbucket
GitHub
GitLab
org/network-web:master
Link workspace to source control repository with Terraform code.
network-web-dev
org/network-web:master tag: v0.4
Promote v.04 to UAT
network-web-uat
org/network-web:master tag: v0.3
Fork network-web-uat to network-web-uat-fork1, config is selected from triage-bug branch.
network-web-uat-fork1
org/network-web:triage-bug
Promote v.03 to Prod
network-web-prod
org/network-web:master tag: v0.2
network-web-fork1
org/network-web:feature22
Fork network-web-dev to network-web-fork1, config is selected from feature branch.
network-web-fork2
org/network-web:feature25
Fork network-web-dev to network-web-fork2 workspace, config is selected from feature branch.

Ways to interact with Terraform

CLI

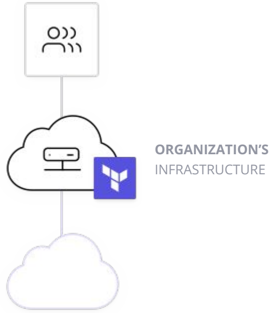
Terraform CLI



- No requirements for collaboration
- No requirements for centralized reusable configs
- No policy or governance requirements

Enterprise/Cloud

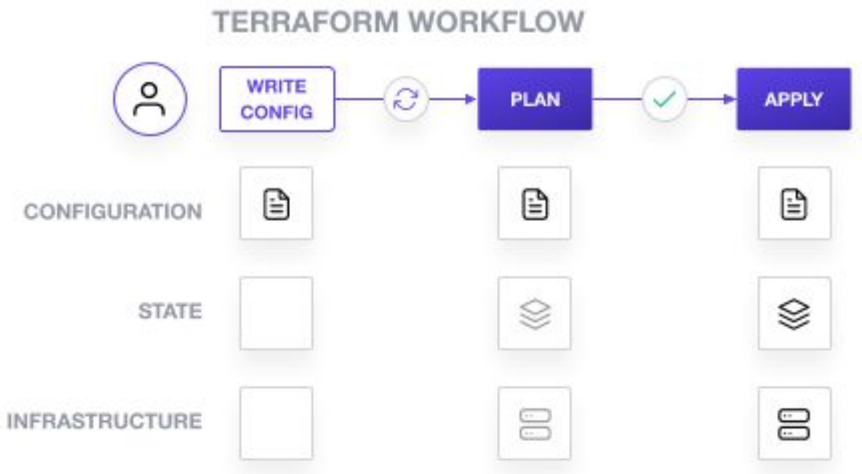
Self-Hosted/Managed



- Air gapped infrastructure and applications
- Data sovereignty requirements
- Regulatory compliance requirements
- Stringent reliability and availability requirements

Foundational Concept

Terraform Workflow

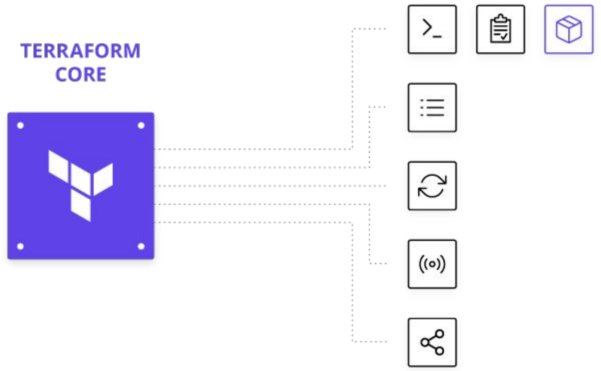


Terraform Core Engine

- OSS hosted at github.com/hashicorp/terraform
- The engine Terraform runs on
- Loads providers as needed

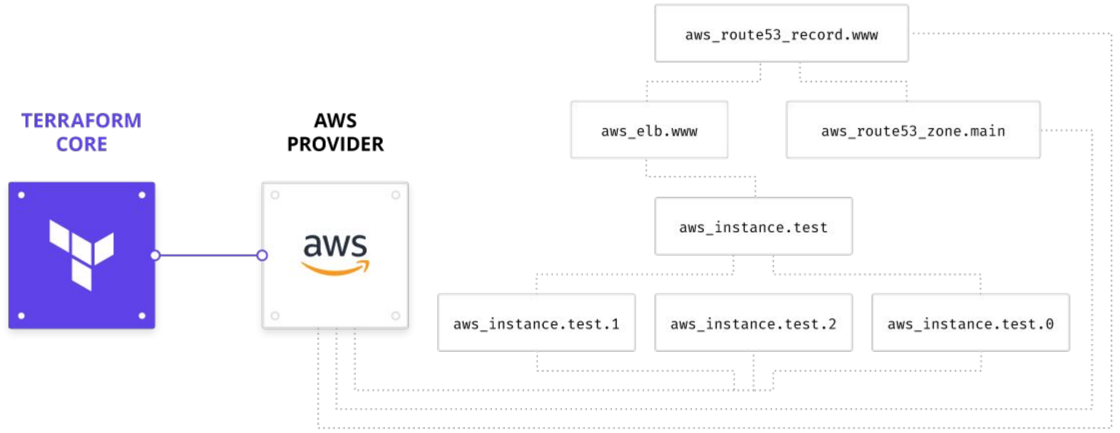
Responsible for:

- Reading and Interpolating configuration files and modules
- State Management
- Executing plan
- Communicating with providers
- Constructing resource graph



Resource Graph

- Safely provision and change infrastructure
- See planned infrastructure changes before execution
- No need to manually coordinate dependent resources

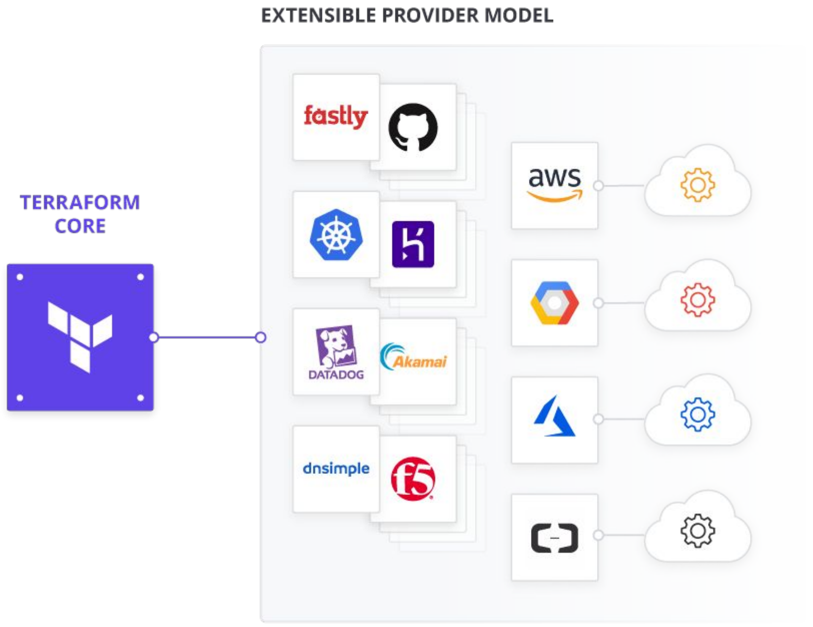


Provider Plugins

- Provider plugins expose implementation for specific services
- Offer extensible layer for 'Core' to learn how to talk to anything with an API without any upgrades

Responsible for:

- Initializing libraries for API calls
- Authenticating with the provider
- Defining resources that map to services
- Executing commands or scripts for designated resources

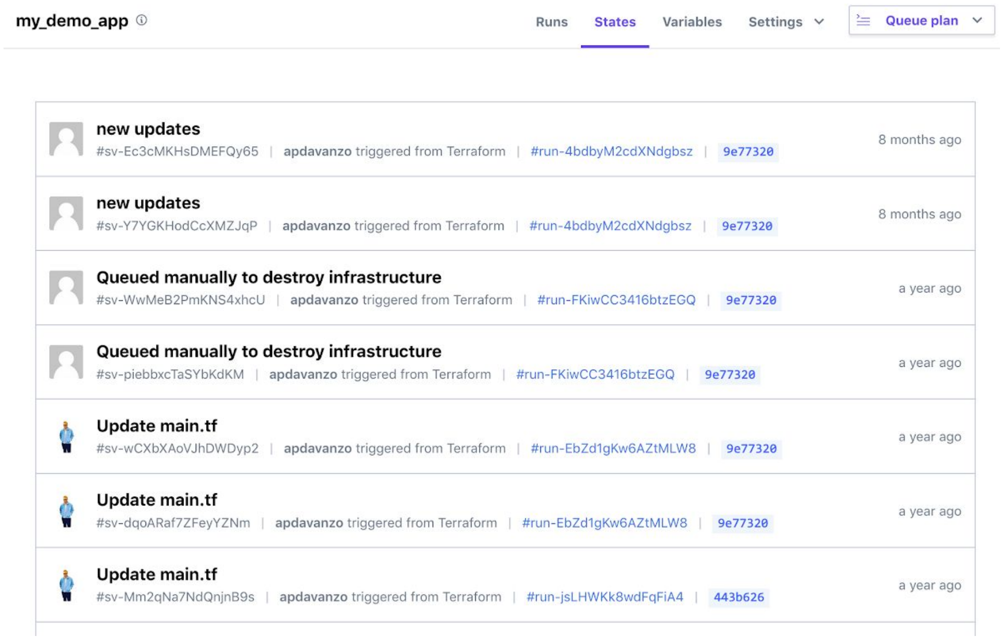


Infrastructure State

State is Terraform's understanding of an infrastructure

Terraform uses state to provide an understanding of resources under its control. Terraform Enterprise can also show previous runs where infrastructure has been changed.

Terraform Enterprise provides remote state management which encrypts the state file.



new updates
#sv-Ec3cMKHsDMEFQy65 | **apdavanzo** triggered from Terraform | #run-4bdbyM2cdXNdgbsz | 9e77320 — 8 months ago

new updates
#sv-Y7YGKHodCcXMZJqP | **apdavanzo** triggered from Terraform | #run-4bdbyM2cdXNdgbsz | 9e77320 — 8 months ago

Queued manually to destroy infrastructure
#sv-WwMeB2PmKNS4xhcU | **apdavanzo** triggered from Terraform | #run-FKiwCC3416btzEGQ | 9e77320 — a year ago

Queued manually to destroy infrastructure
#sv-piebbxcTaSYbKdKM | **apdavanzo** triggered from Terraform | #run-FKiwCC3416btzEGQ | 9e77320 — a year ago

Update main.tf
#sv-wCXbXAoVJhDWDyp2 | **apdavanzo** triggered from Terraform | #run-EbZd1gKw6AZtMLW8 | 9e77320 — a year ago

Update main.tf
#sv-dqoARaf7ZFeyYZNm | **apdavanzo** triggered from Terraform | #run-EbZd1gKw6AZtMLW8 | 9e77320 — a year ago

Update main.tf
#sv-Mm2qNa7NdQnjnB9s | **apdavanzo** triggered from Terraform | #run-jsLHWKk8wdFqFiA4 | 443b626 — a year ago

Options for moving to Terraform Enterprise

Objective

Start managing your existing infrastructure within Terraform Enterprise.

Options

Once you understand how you manage infrastructure today, you can determine how to migrate your state into Terraform Enterprise.

- Infrastructure **not managed** by Terraform:

  Option: Import unmanaged cloud resources into Terraform Enterprise

- Infrastructure **managed** by Terraform:

  Option: Migrate existing state from Terraform OSS to Terraform Enterprise

Import Unmanaged Cloud Resources

Terraform Import

Terraform is able to import existing infrastructure.

This allows you take resources you've created by some other means and bring it under Terraform management.

This is a great way to slowly transition infrastructure to Terraform.

https://www.terraform.io/docs/cli/import/index.html

Foundational Concept

Terraform Import Workflow

	CODE	IMPORT	PLAN	APPLY
CODE	●	●	●	●
STATE	○	●	●	●
INFRA	○	○	○	●

Importing Prerequisites

Before you can import existing infrastructure into Terraform Enterprise you must have completed the following:

- Installed Terraform CLI locally
- Deployed Terraform Enterprise

Check out the HashiCorp Learn tutorial:

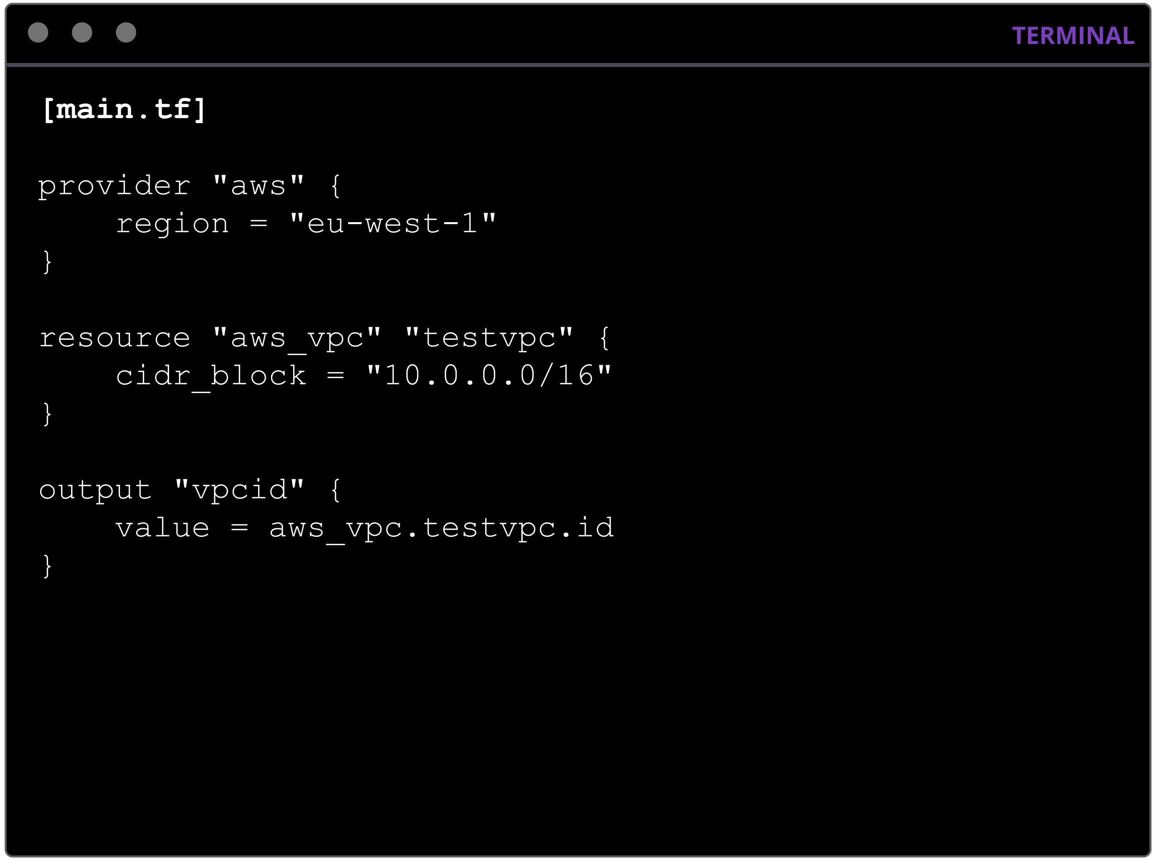
https://learn.hashicorp.com/tutorials/terraform/state-import

Steps

1. Write Terraform code that matches your infrastructure
2. Import infrastructure into your Terraform state file using `terraform import`
3. Review plan output from `terraform plan` to ensure the configuration matches expected state
4. Apply the configuration to update your Terraform state by running `terraform apply`

Step 1

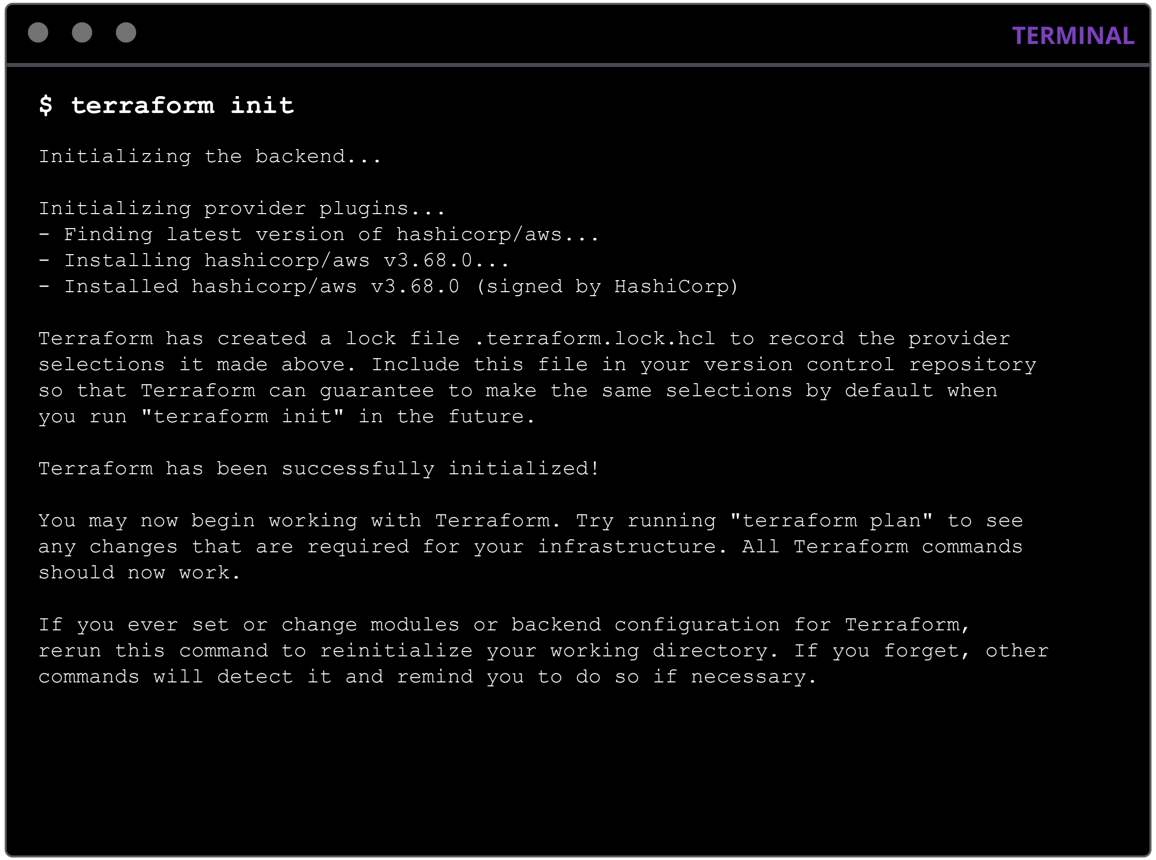
Write Terraform code that matches your infrastructure.



```
[main.tf]

provider "aws" {
    region = "eu-west-1"
}

resource "aws_vpc" "testvpc" {
    cidr_block = "10.0.0.0/16"
}

output "vpcid" {
    value = aws_vpc.testvpc.id
}
```

Step 2

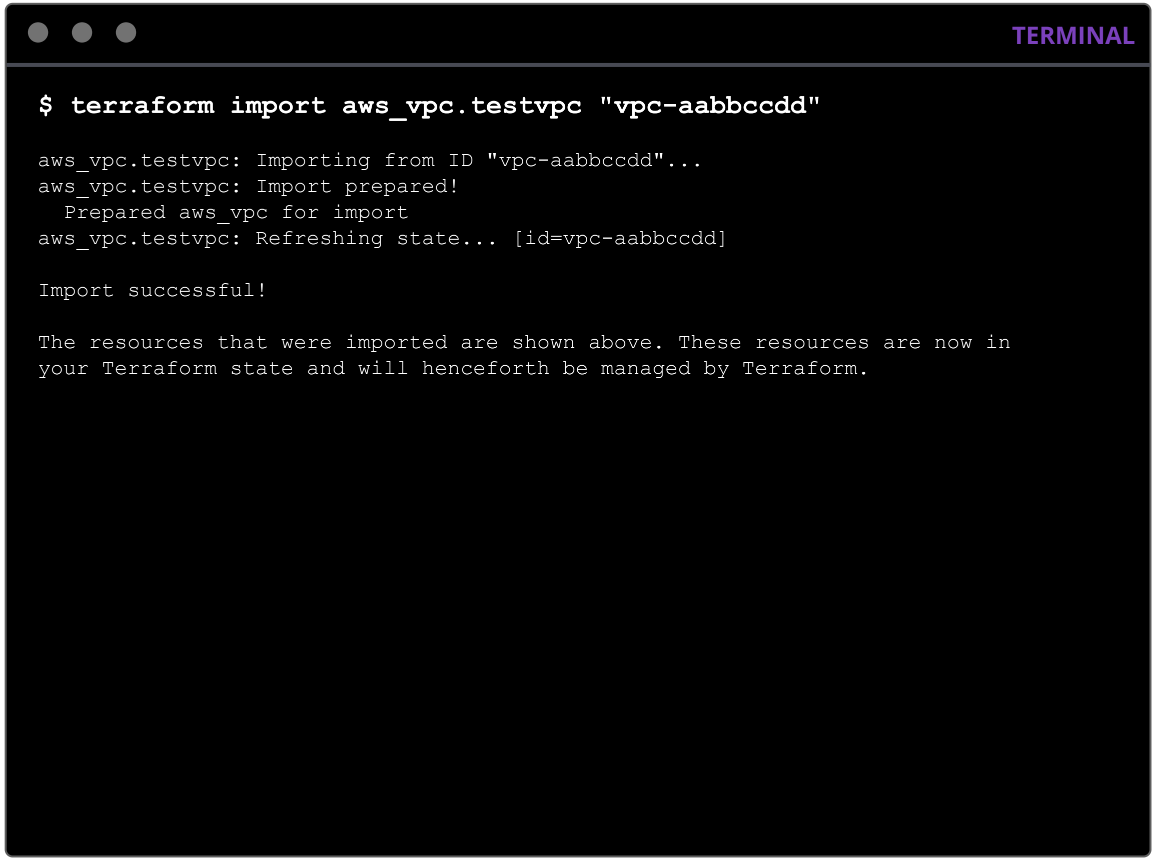
Initialize Terraform with `terraform init`



```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.68.0...
- Installed hashicorp/aws v3.68.0 (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 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.
```

Step 3

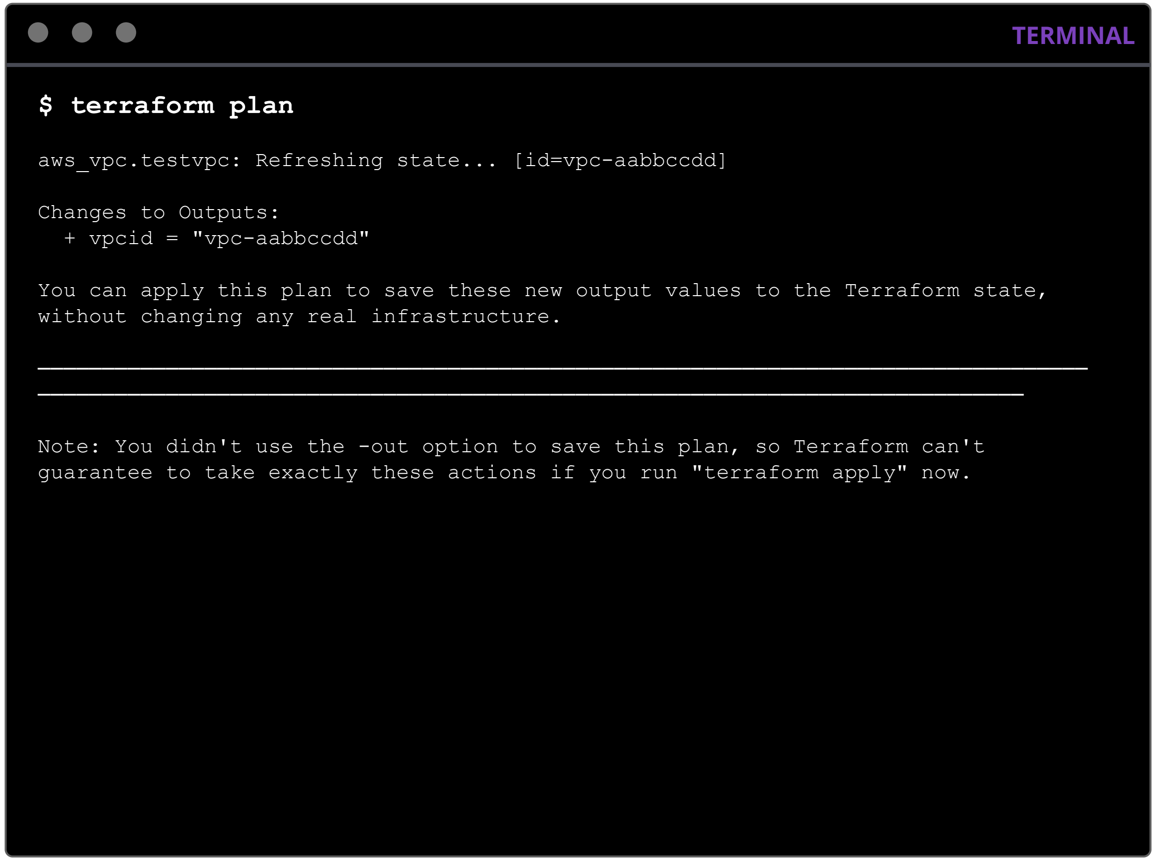
Import infrastructure into your Terraform state file using `terraform import`



```
$ terraform import aws_vpc.testvpc "vpc-aabbccdd"

aws_vpc.testvpc: Importing from ID "vpc-aabbccdd"...
aws_vpc.testvpc: Import prepared!
  Prepared aws_vpc for import
aws_vpc.testvpc: Refreshing state... [id=vpc-aabbccdd]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

Step 4

Review the plan output from `terraform plan` to ensure the configuration matches expected state

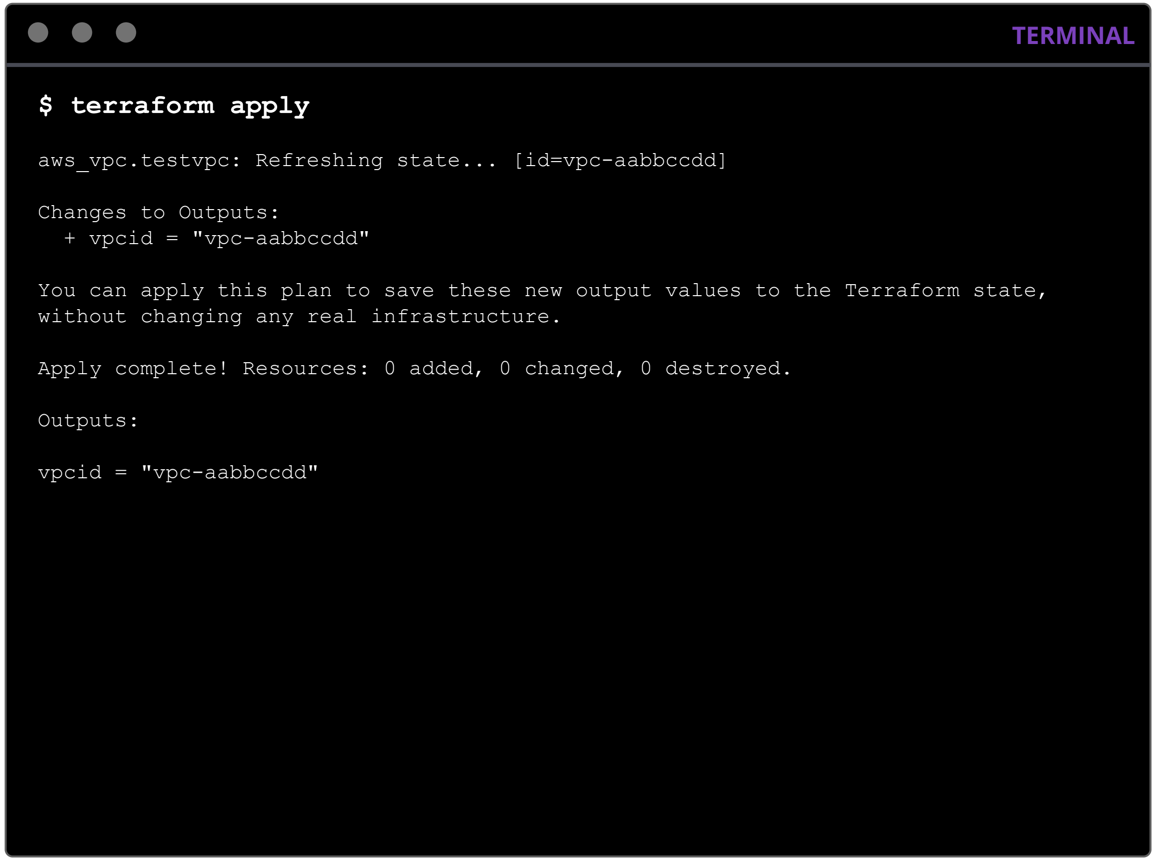

TERMINAL

```
$ terraform plan

aws_vpc.testvpc: Refreshing state... [id=vpc-aabbccdd]

Changes to Outputs:
  + vpcid = "vpc-aabbccdd"

You can apply this plan to save these new output values to the Terraform state,
without changing any real infrastructure.

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

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
```

Step 5

Apply the configuration to update your Terraform state by running

`terraform apply`



TERMINAL

```
$ terraform apply

aws_vpc.testvpc: Refreshing state... [id=vpc-aabbccdd]

Changes to Outputs:
  + vpcid = "vpc-aabbccdd"

You can apply this plan to save these new output values to the Terraform state,
without changing any real infrastructure.

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

Outputs:

vpcid = "vpc-aabbccdd"
```

Migrating Existing State

Prerequisites

Before you can migrate Terraform OSS state into Terraform Enterprise you must have completed the following:

- Installed Terraform CLI locally
- Deployed Terraform Enterprise

Optionally: Acquired an API token for Terraform Enterprise

Steps

… Before

… During

… After

Before migration:

- Take a backup!
- Ensure that you have initialized your existing state
- Create and configure the Workspace in Terraform Enterprise

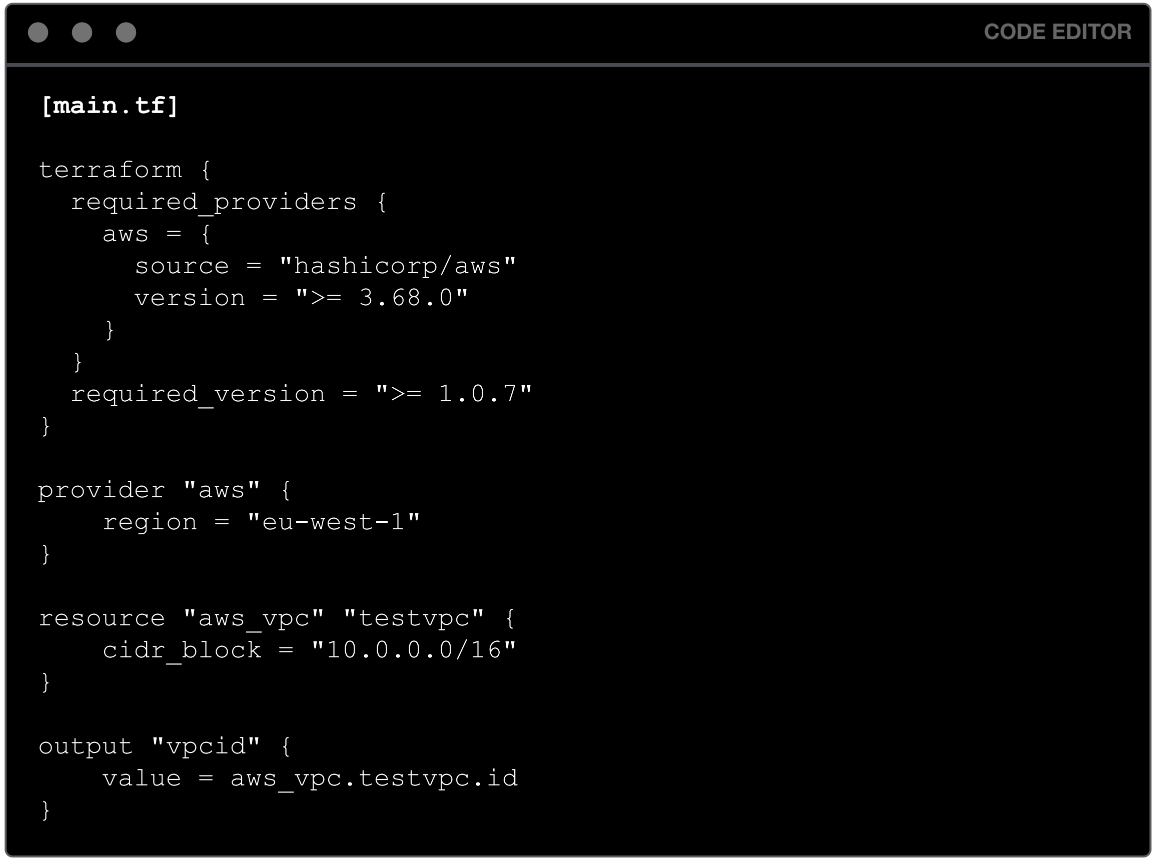
1. Login to Terraform Enterprise and generate an API Token
2. Add the Terraform remote backend
3. Reinitialize Terraform and confirm state migration

After migration:

- Verify that the state has been migrated to the workspace
- Move old state (to another backup)
- Trigger a remote run within Terraform Enterprise
- Check everything worked as expected

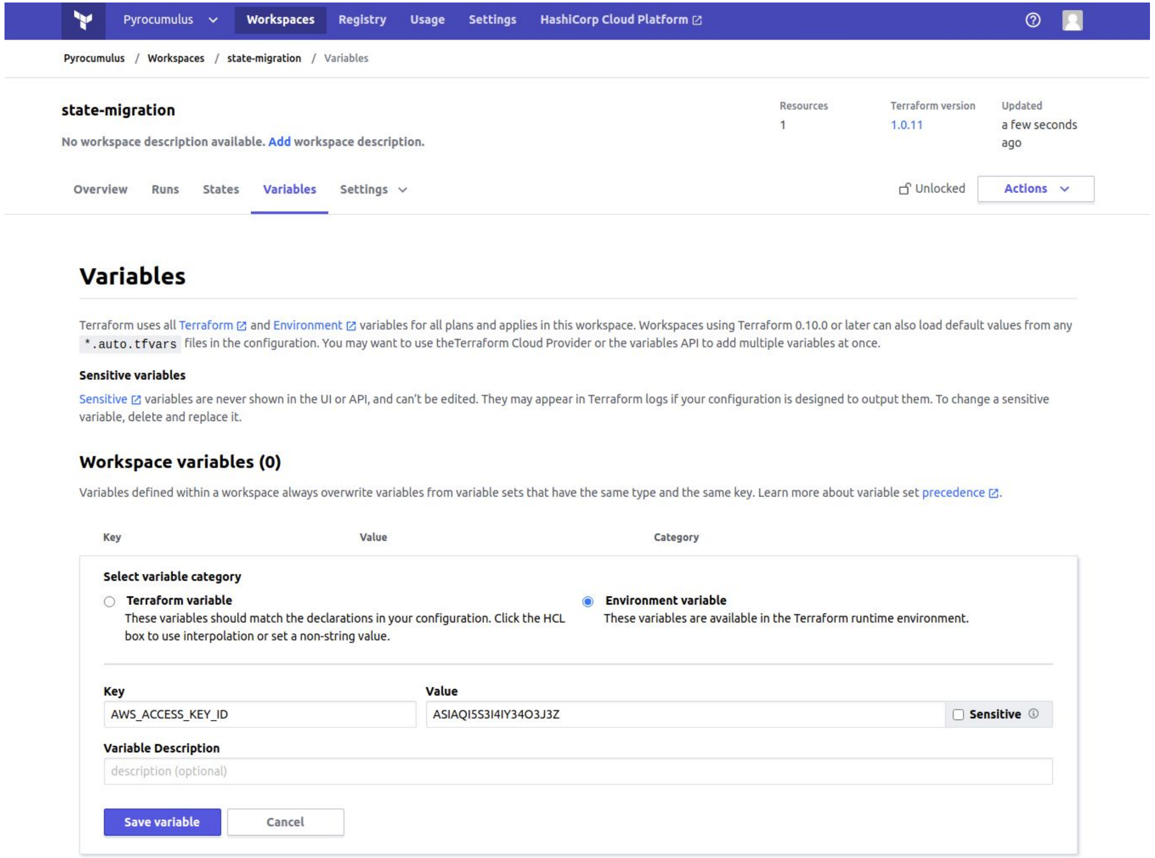
Before

Review the existing Terraform code


CODE EDITOR

```
[main.tf]

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = ">= 3.68.0"
    }
  }
  required_version = ">= 1.0.7"
}

provider "aws" {
    region = "eu-west-1"
}

resource "aws_vpc" "testvpc" {
    cidr_block = "10.0.0.0/16"
}

output "vpcid" {
    value = aws_vpc.testvpc.id
}
```

Create and configure the workspace to include AWS credentials



Pyrocumulus / Workspaces / state-migration / Variables

state-migration

No workspace description available. Add workspace description.

Resources	Terraform version	Updated
1	1.0.11	a few seconds ago

Variables

Terraform uses all Terraform and Environment variables for all plans and applies in this workspace. Workspaces using Terraform 0.10.0 or later can also load default values from any `*.auto.tfvars` files in the configuration. You may want to use theTerraform Cloud Provider or the variables API to add multiple variables at once.

Sensitive variables

Sensitive variables are never shown in the UI or API, and can't be edited. They may appear in Terraform logs if your configuration is designed to output them. To change a sensitive variable, delete and replace it.

Workspace variables (0)

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

Select variable category

Terraform variable [✓ Environment variable]

Terraform variable: These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.

Environment variable: These variables are available in the Terraform runtime environment.

Key: AWS_ACCESS_KEY_ID

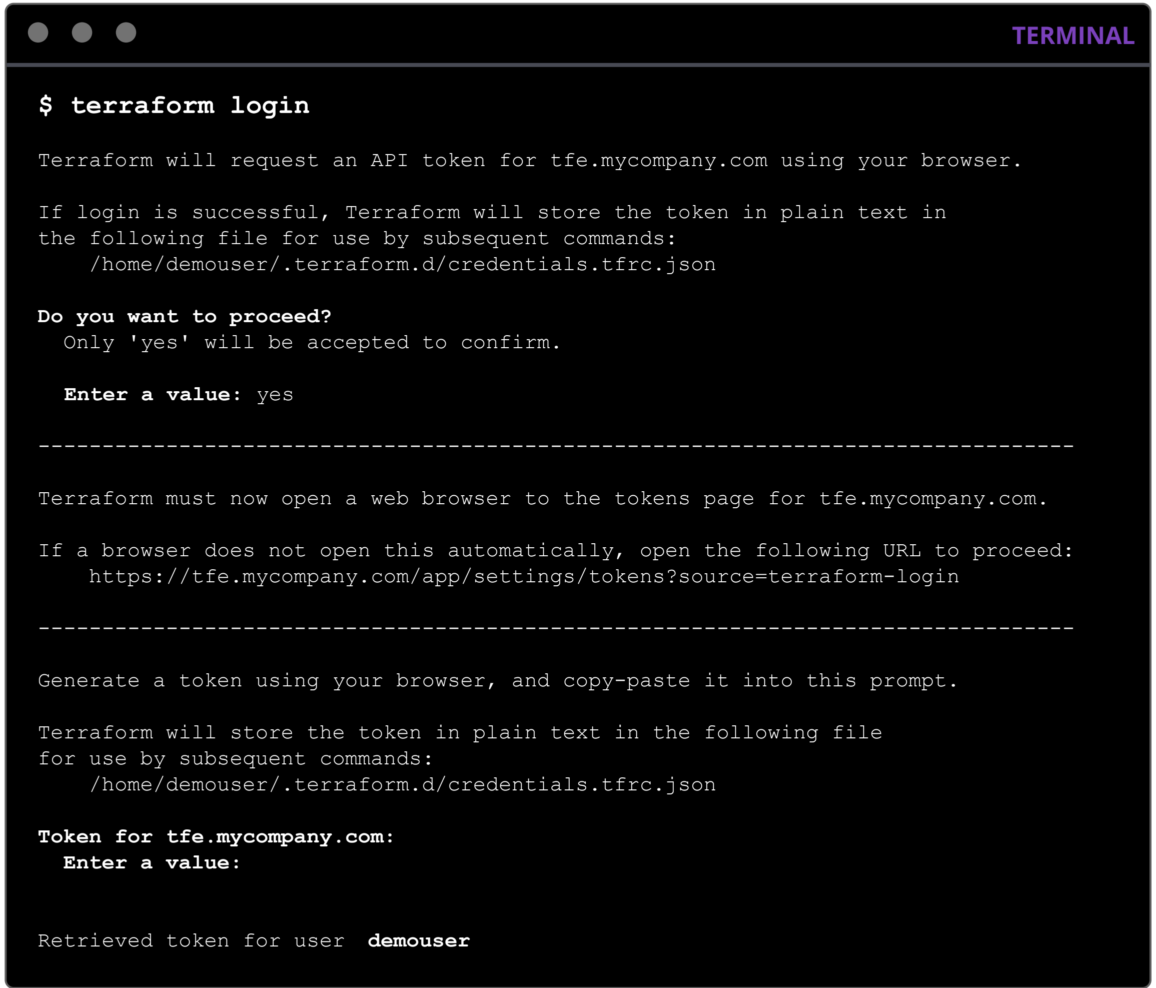
Value: ASIAQI5S3I4IY34O3J3Z

Sensitive: ☐

Variable Description: description (optional)

Step 1

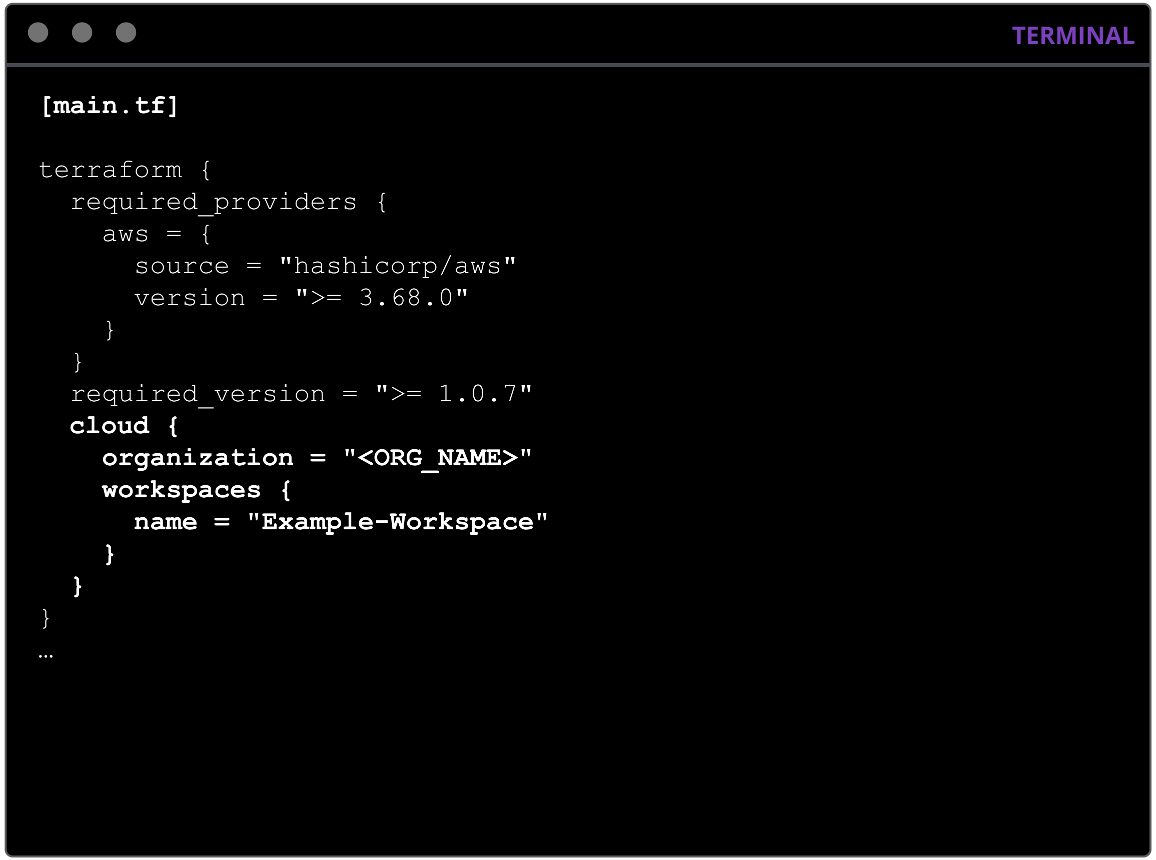
Login to Terraform Enterprise and generate an API Token



```
$ terraform login

Terraform will request an API token for tfe.mycompany.com 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/demouser/.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 tfe.mycompany.com.

If a browser does not open this automatically, open the following URL to proceed:
    https://tfe.mycompany.com/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/demouser/.terraform.d/credentials.tfrc.json

Token for tfe.mycompany.com:
  Enter a value:


Retrieved token for user  demouser
```

Step 2

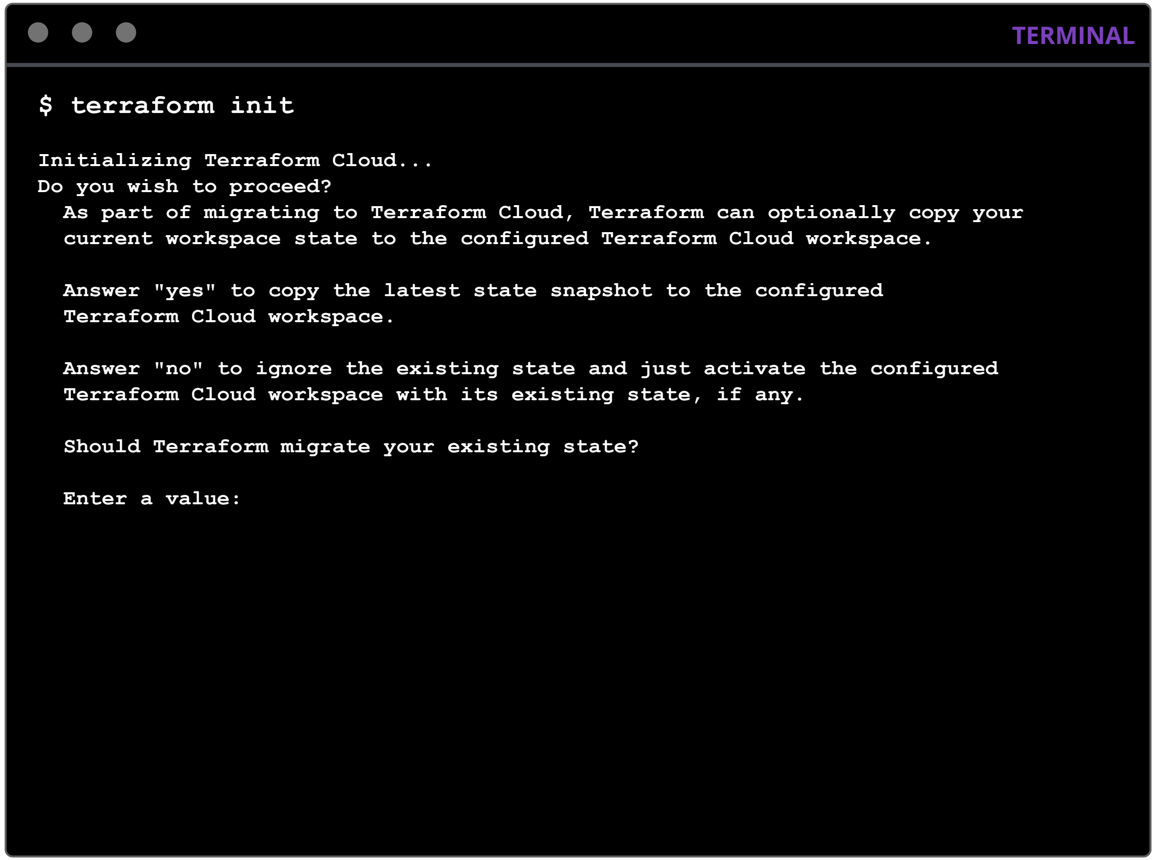
Add the Terraform Remote Backend



```
[main.tf]

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = ">= 3.68.0"
    }
  }
  required_version = ">= 1.0.7"
  cloud {
    organization = "<ORG_NAME>"
    workspaces {
      name = "Example-Workspace"
    }
  }
}
…
```

Step 3

Reinitialize Terraform and confirm state migration

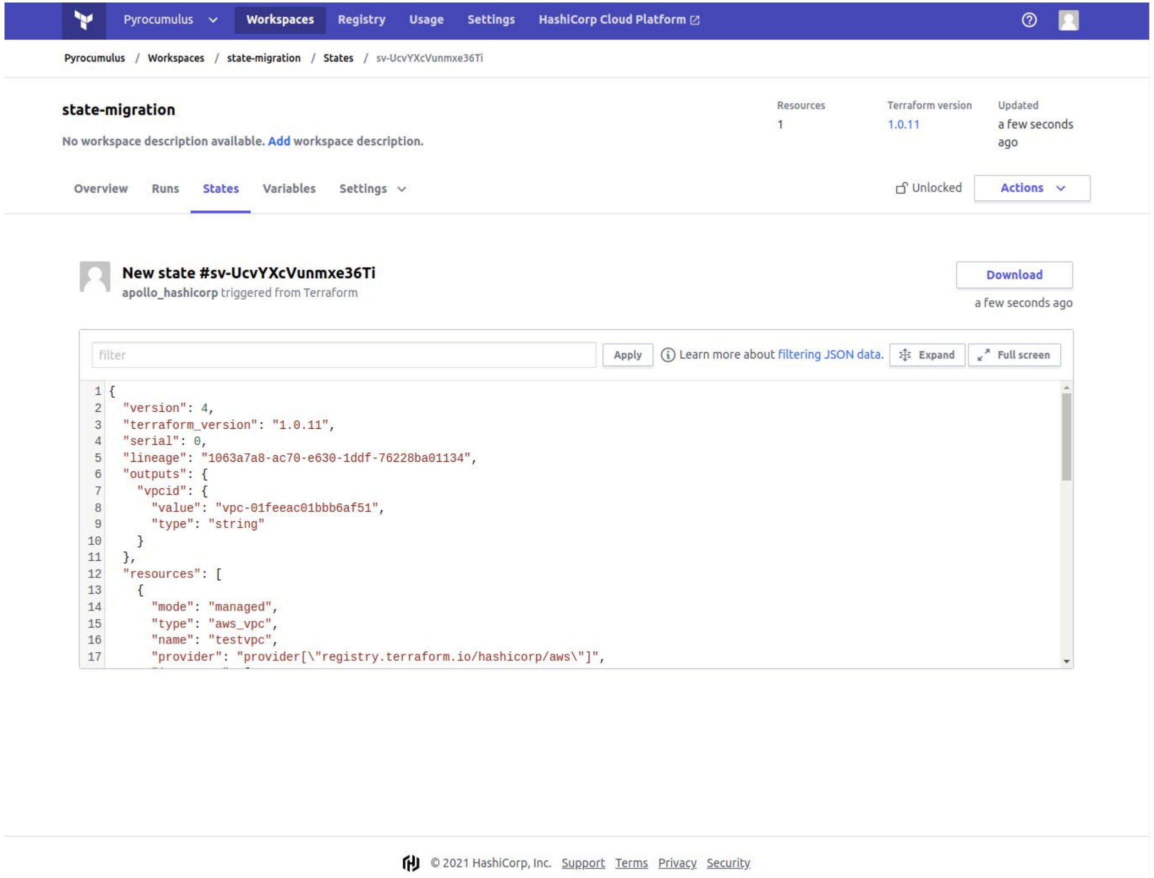


TERMINAL

```
$ terraform init

Initializing Terraform Cloud...
Do you wish to proceed?
  As part of migrating to Terraform Cloud, Terraform can optionally copy your
  current workspace state to the configured Terraform Cloud workspace.

  Answer "yes" to copy the latest state snapshot to the configured
  Terraform Cloud workspace.

  Answer "no" to ignore the existing state and just activate the configured
  Terraform Cloud workspace with its existing state, if any.

  Should Terraform migrate your existing state?

  Enter a value:
```

After

Verify the workspace exists and that the state file has been uploaded



Pyrocumulus / Workspaces / state-migration / States / sv-UcvYXcVunmxe36Ti

state-migration

No workspace description available. Add workspace description.

Resources: 1 | Terraform version: 1.0.11 | Updated: a few seconds ago

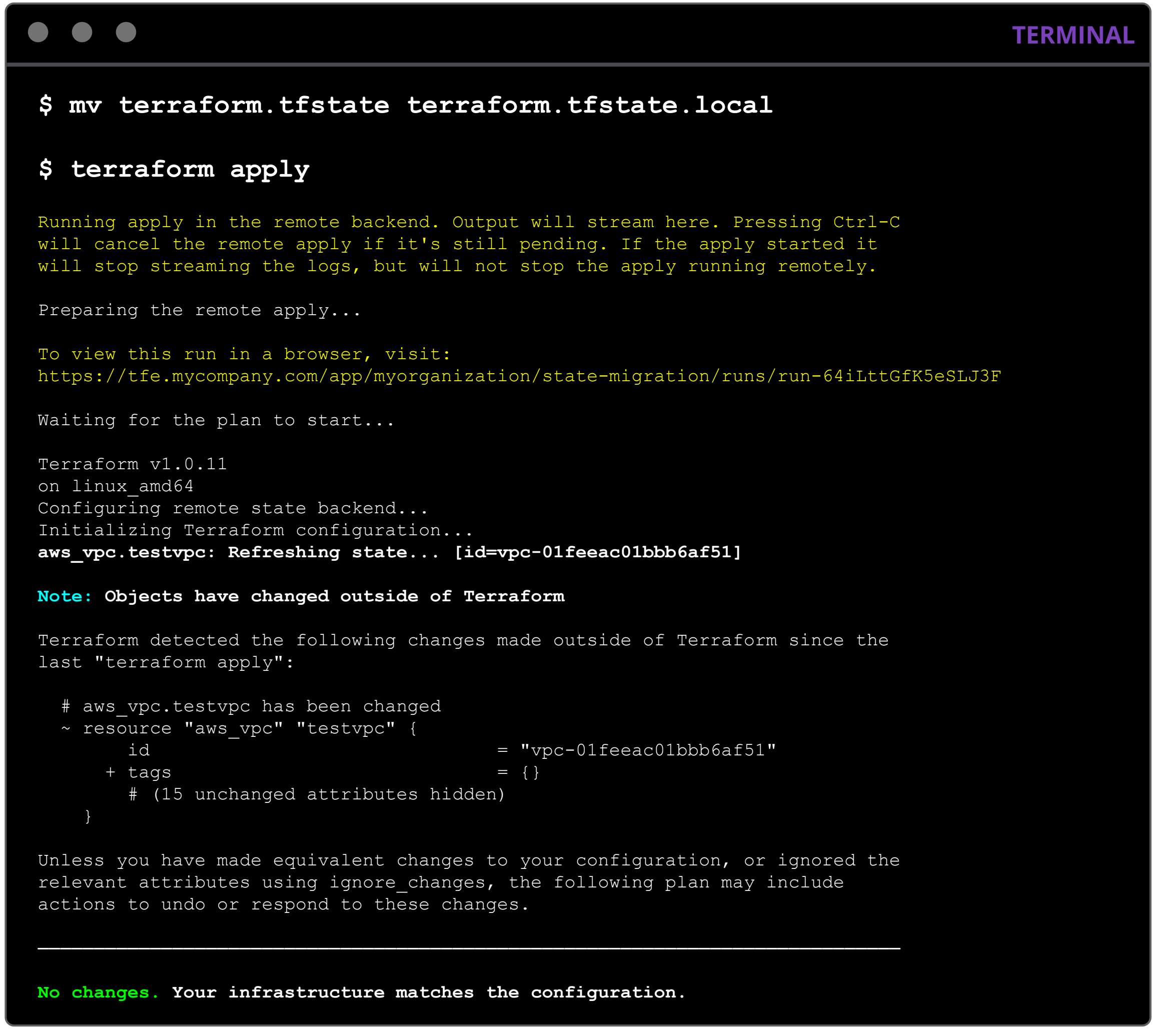
New state #sv-UcvYXcVunmxe36Ti
apollo_hashicorp triggered from Terraform

a few seconds ago

Learn more about filtering JSON data.

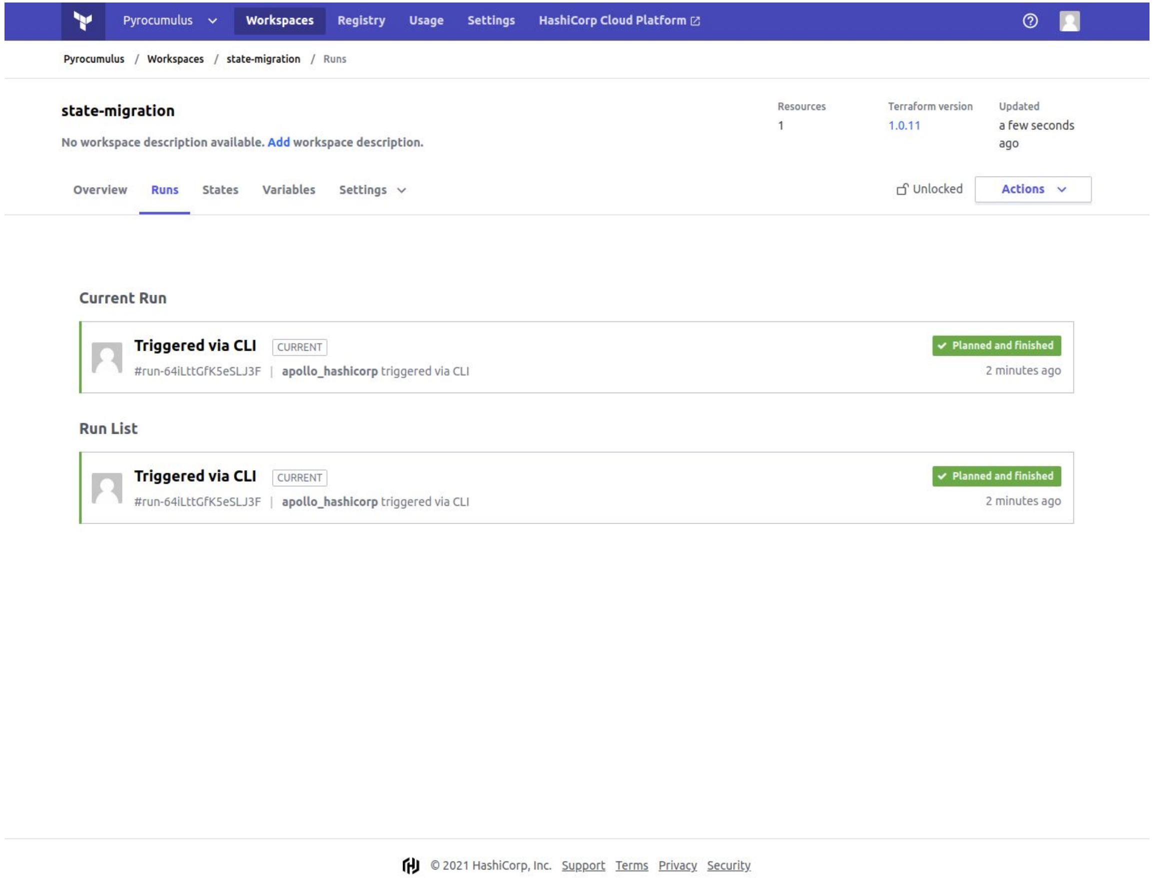
```
{
  "version": 4,
  "terraform_version": "1.0.11",
  "serial": 0,
  "lineage": "1063a7a8-ac70-e630-1ddf-76228ba01134",
  "outputs": {
    "vpcid": {
      "value": "vpc-01feeac01bbb6af51",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "aws_vpc",
      "name": "testvpc",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
```

Trigger a remote run within Terraform Enterprise.



```
$ mv terraform.tfstate terraform.tfstate.local

$ terraform apply

Running apply in the remote backend. Output will stream here. Pressing Ctrl-C
will cancel the remote apply if it's still pending. If the apply started it
will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply...

To view this run in a browser, visit:
https://tfe.mycompany.com/app/myorganization/state-migration/runs/run-64iLttGfK5eSLJ3F

Waiting for the plan to start...

Terraform v1.0.11
on linux_amd64
Configuring remote state backend...
Initializing Terraform configuration...
aws_vpc.testvpc: Refreshing state... [id=vpc-01feeac01bbb6af51]

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the
last "terraform apply":

  # aws_vpc.testvpc has been changed
  ~ resource "aws_vpc" "testvpc" {
        id                                = "vpc-01feeac01bbb6af51"
      + tags                              = {}
        # (15 unchanged attributes hidden)
    }

Unless you have made equivalent changes to your configuration, or ignored the
relevant attributes using ignore_changes, the following plan may include
actions to undo or respond to these changes.

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

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

Verify a remote run has been triggered by the CLI



Pyrocumulus / Workspaces / state-migration / Runs

state-migration

No workspace description available. Add workspace description.

Resources	Terraform version	Updated
1	1.0.11	a few seconds ago

Current Run

Triggered via CLI CURRENT
#run-64iLttGfK5eSLJ3F | **apollo_hashicorp** triggered via CLI
✓ Planned and finished
2 minutes ago

Run List

Triggered via CLI CURRENT
#run-64iLttGfK5eSLJ3F | **apollo_hashicorp** triggered via CLI
✓ Planned and finished
2 minutes ago

Resources

Resources

- Migrating Terraform OSS to Terraform Enterprise
- Importing Existing Infrastructure into Terraform Enterprise
- Terraform Import
- Community Tools for importing resources*
    - aws2tf
    - Terraformer

* These community projects are **not maintained, supported or endorsed** by HashiCorp.

Next Steps

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered.

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a new ticket for your issue at support.hashicorp.com.

Q & A

Thank You

hello@hashicorp.com
www.hashicorp.com