Use Refresh-Only Mode to Sync Terraform State

7 MIN | PRODUCTS USED Terraform

This tutorial also appears in: HashiConf Europe and Associate Tutorials.

Terraform relies on the contents of your workspace's state file to generate an execution plan to make changes to your resources. To ensure the accuracy of the proposed changes, your state file must be up to date.

In Terraform, refreshing your state file updates Terraform's knowledge of your infrastructure, as represented in your state file, with the actual state of your infrastructure. Terraform `plan` and `apply` operations run an implicit in-memory refresh as part of their functionality, reconciling any drift from your state file before suggesting infrastructure changes. You can also update your state file without making modifications to your infrastructure using the `-refresh-only` flag for `plan` and `apply` operations.

In this tutorial, you will safely refresh your Terraform state file using the `-refresh-only` flag. You will also review Terraform's implicit refresh behavior and the advantages of the `-refresh-only` flag over the deprecated `terraform refresh` subcommand.

Prerequisites

This tutorial assumes that you are familiar with the standard Terraform workflow. If you are new to Terraform, complete the Get Started tutorials first.

For this tutorial, you will need:

- Terraform v0.15.4+ installed locally
- an AWS account with credentials configured for Terraform

> **Note**: Some of the infrastructure in this tutorial may not qualify for the AWS free tier. Destroy the infrastructure at the end of the guide to avoid unnecessary charges. We are not responsible for any charges that you incur.

Clone example repository

Clone the sample repository for this tutorial.

```
$ git clone https://github.com/hashicorp/learn-terraform
```

Copy

Change into the repository directory.

```
$ cd learn-terraform-refresh
```

Copy

Deploy EC2 instance

Open `main.tf` to review the sample configuration. It defines an EC2 instance and a data source to identify the latest Amazon Linux AMI. The provider block references the `region` input variable, which defaults to `us-east-2`.

Now initialize your directory.

```
$ terraform init
```

Copy

```
Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.27"...
- Installing hashicorp/aws v3.42.0...
- Installed hashicorp/aws v3.42.0 (signed by HashiCorp)

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform pl
any changes that are required for your infrastructure. All Terrafor
should now work.

If you ever set or change modules or backend configuration for Terr
rerun this command to reinitialize your working directory. If you f
commands will detect it and remind you to do so if necessary.
```

Apply your configuration. Respond `yes` to the prompt to confirm the operation.

```
$ terraform apply
```

Copy

```
Terraform used the selected providers to generate the following exe
  + create

Terraform will perform the following actions:

  # aws_instance.server will be created
  + resource "aws_instance" "server" {
##...
```

```
    }
Plan: 1 to add, 0 to change, 0 to destroy.
Changes to Outputs:
  + instance_id = (known after apply)

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

aws_instance.server: Creating...
aws_instance.server: Still creating... [10s elapsed]
aws_instance.server: Still creating... [20s elapsed]
aws_instance.server: Still creating... [30s elapsed]
aws_instance.server: Still creating... [40s elapsed]
aws_instance.server: Still creating... [50s elapsed]
aws_instance.server: Creation complete after 56s [id=i-01bc0d010f5a

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-01bc0d010f5aa6007"
```

Run a refresh-only plan

A common error scenario that can prompt Terraform to refresh the contents of your state file is mistakenly modifying your credentials or provider configuration. Simulate this situation by updating your AWS provider's region. You will then review the proposed changes to your state file from a Terraform refresh.

Create a `terraform.tfvars` file in your `learn-terraform-refresh` directory. Open the file, and paste in the following configuration to override the default region variable.

```
region = "us-west-2"
```

Copy

Since you pass the `region` variable to your AWS provider configuration in `main.tf`, this will reconfigure your provider for the `us-west-2` region. The resources you created earlier are still in `us-east-2`.

Run `terraform plan -refresh-only` to review how Terraform would update your state file.

```
$ terraform plan -refresh-only
aws_instance.server: Refreshing state... [id=i-01bc0d010f5aa6007]

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform

  # aws_instance.server has been deleted
  - resource "aws_instance" "server" {
   ##...
    }

This is a refresh-only plan, so Terraform will not take any actions
this plan to record the updated values in the Terraform state withc

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

Changes to Outputs:
  ~ instance_id = "i-01bc0d010f5aa6007" -> (known after apply)

You can apply this plan to save these new output values to the Terr

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

Note: You didn't use the -out option to save this plan, so Terrafor
apply" now.
```

Because you updated your provider for the `us-west-2` region, Terraform tries to locate the EC2 instance with the instance ID tracked in your state file but fails to locate it since it's in a different region. Terraform assumes that you destroyed the instance and wants to remove it from your state file.

If the modifications to your state file proposed by a `-refresh-only` plan were acceptable, you could run a `terraform apply -refresh-only` and approve the operation to overwrite your state file without modifying your infrastructure. However, in this tutorial, refreshing your state file would drop your resources, so do **not** run the apply operation.

Review Terraform's refresh functionality

In previous versions of Terraform, the only way to refresh your state file was by using the `terraform refresh` subcommand. However, this was less safe than the `-refresh-only` plan and apply mode since it would automatically overwrite your state file without giving you the option to review the modifications first. In this case, that would mean automatically dropping all of your resources from your state file.

The `-refresh-only` mode for `terraform plan` and `terraform apply` operations makes it safer to check Terraform state against real infrastructure by letting you review proposed changes to the state file. It lets you avoid mistakenly removing an existing resource from state and gives you a chance to correct your configuration.

A refresh-only `apply` operation also updates outputs, if necessary. If you have any other workspaces that use the `terraform_remote_state` data source to access the outputs of the current workspace, the `-refresh-only` mode allows you to anticipate the downstream effects.

In order to propose accurate changes to your infrastructure, Terraform first attempts to reconcile the resources tracked in your state file with your actual infrastructure. Terraform `plan` and `apply` operations first run an in-memory refresh to determine which changes to propose to your

infrastructure. Once you confirm a `terraform apply`, Terraform will update your infrastructure and state file.

Though Terraform will continue to support the `refresh` subcommand in future versions, it is deprecated, and we encourage you to use the `-refresh-only` flag instead. This allows you to review any updates to your state file. Unlike the `refresh` subcommand, `-refresh-only` mode is supported in workspaces using Terraform Cloud as a remote backend, allowing your team to collaboratively review any modifications.

Clean up resources

Now that you have reviewed the behavior of the `-refresh-only` flag, you can destroy the EC2 instance you provisioned.

First, remove your `terraform.tfvars` file to use default value for the `region` variable.

```
$ rm terraform.tfvars
```

Copy

Now run `terraform destroy` to destroy your infrastructure. Respond `yes` to the prompt to confirm the operation.

```
$ terraform destroy
aws_instance.server: Refreshing state... [id=i-01bc0d010f5aa6007]

Terraform used the selected providers to generate the following exe
  - destroy

Terraform will perform the following actions:

  # aws_instance.server will be destroyed
  - resource "aws_instance" "server" {
##...
    }
```

Copy

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

Changes to Outputs:
  - instance_id = "i-01bc0d010f5aa6007" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.server: Destroying... [id=i-01bc0d010f5aa6007]
aws_instance.server: Still destroying... [id=i-01bc0d010f5aa6007, 1
aws_instance.server: Still destroying... [id=i-01bc0d010f5aa6007, 2
aws_instance.server: Still destroying... [id=i-01bc0d010f5aa6007, 3
aws_instance.server: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```

Next steps

You used Terraform's `-refresh-only` mode to safely compare your infrastructure and state file. You also reviewed the implicit refresh behavior in standard Terraform operations.

To learn more about managing state and drift, complete the following tutorials on HashiCorp Learn:

- Manage Resource Drift
- Manage Resources in State
- Troubleshoot common issues with Terraform

Was this tutorial helpful?

Yes No

BACK TO COLLECTION

HashiCorp

System Status Cookie Manager Terms of Use Security Privacy

stdin: is not a tty