How to use Workspaces in Terraform?

Terraform workspaces are a feature in Terraform that allows you to manage multiple versions of the same configuration in a single Terraform state and configuration directory. The workspaces in Terraform allow you to separate the state and configuration data for different environments, such as production, staging, and development, or for different customers or teams.

Each Terraform workspace is associated with a unique state and a unique set of variables. When you switch between workspaces, Terraform loads the corresponding state and variables, allowing you to manage different environments independently.

You can create a new workspace using the **terraform workspace new**command, and switch between workspaces using the **terraform workspace select**command. When you run Terraform commands such as **terraform apply**, Terraform uses the currently selected workspace.

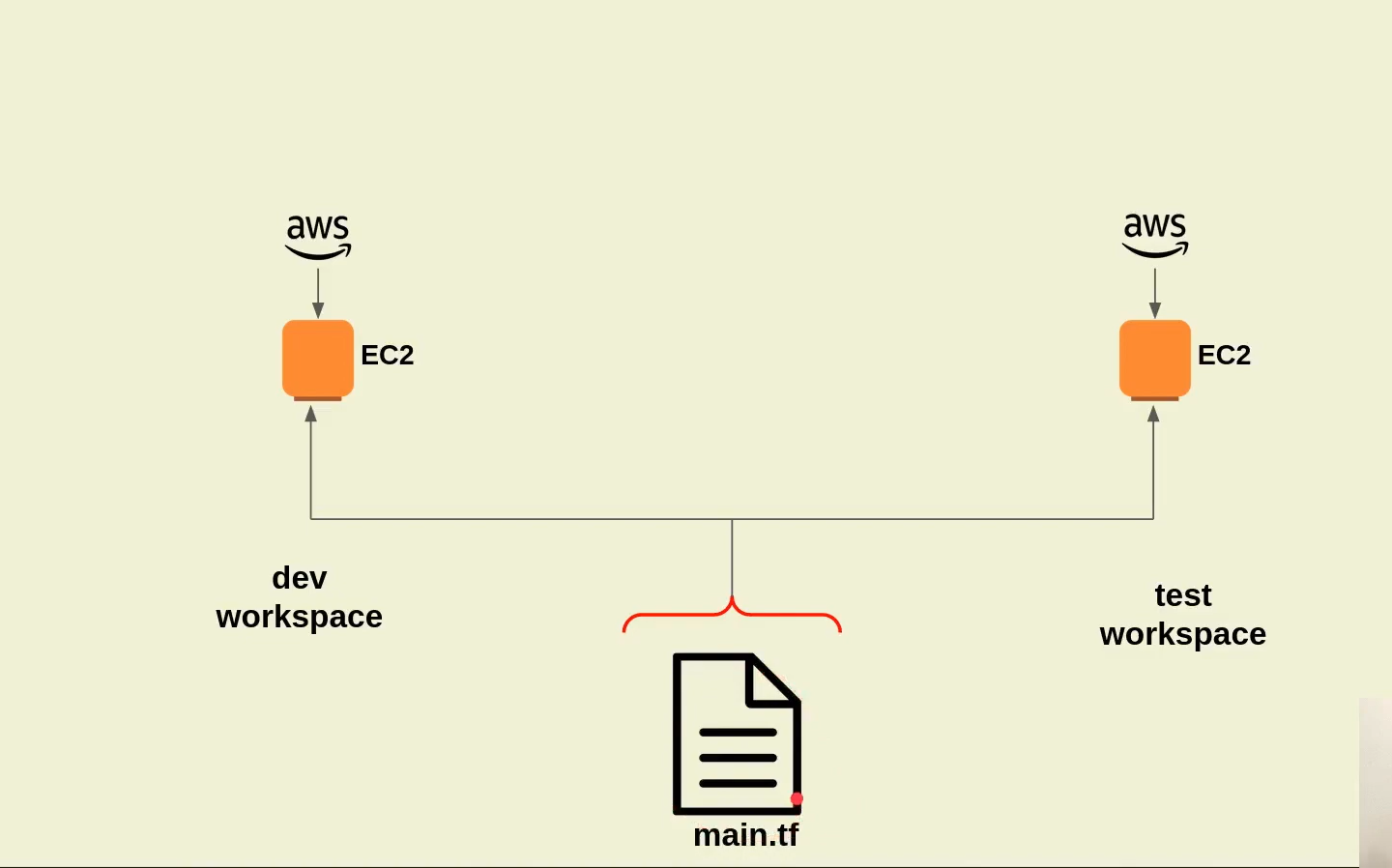
In addition to separating state and variables, workspaces can also be used to manage different sets of providers and modules. When you specify a provider in your Terraform configuration, you can specify which workspace it should be associated with. This allows you to use different providers for different workspaces, or to use different configurations for the same provider.

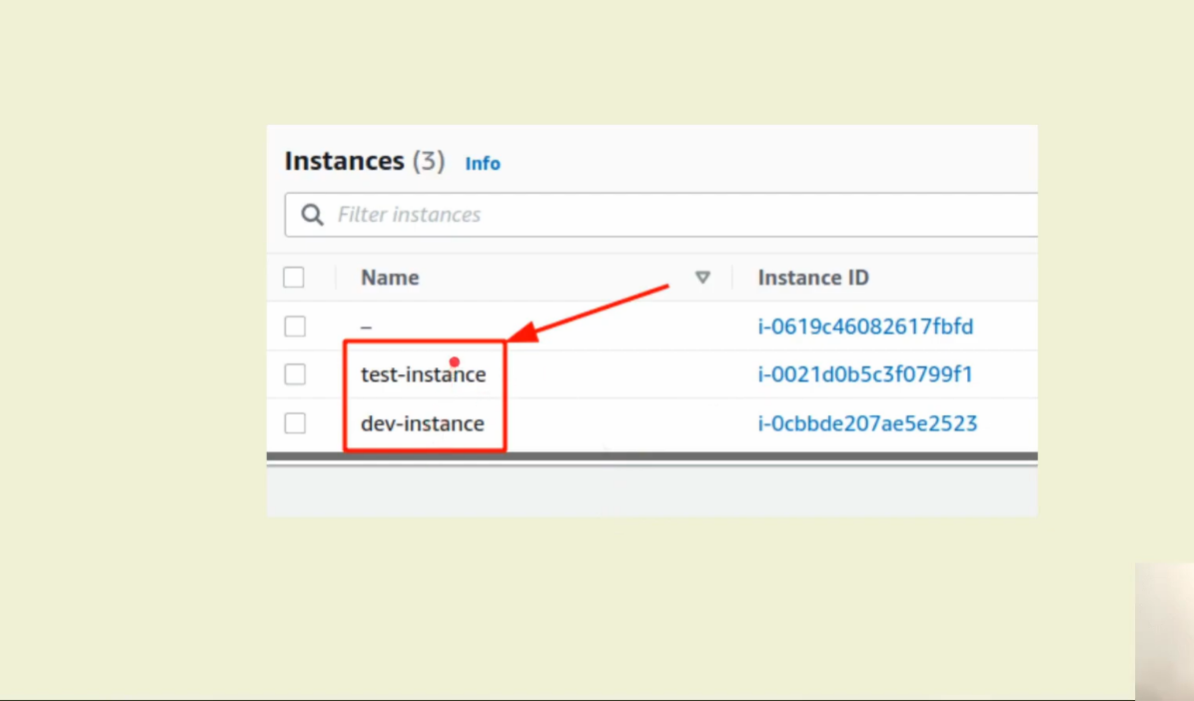
In summary, Terraform workspaces are a powerful tool for managing multiple environments, customers, or teams, within a single Terraform configuration and state. By separating state and variables, providers, and modules, Terraform workspaces enable you to manage each environment independently and to make changes to one environment without affecting others.

[Terraform worksapces](https://www.terraform.io/docs/language/state/workspaces.html) is a very logical concept where you can have multiple states of your infrastructure configuration. To put this in simple words if you are running an infrastructure configuration in *development* environment then the same infrastructure can be run in the *production* environment.

The main benefit of terraforming workspaces we get is we can have more than one state associated with a single terraform configuration.

If you have not defined any workspace then there is always a default workspace created by terraform, so you always work in a default workspace of terraform. You can list the number of terraform workspaces by running the command terraform workspace show. Also, you can not delete the default workspace





Here is our Table of Content on How to work with Terraform workspaces

1. [How to create a new Terraform workspace](https://jhooq.com/terraform-workspaces/#1-how-to-create-a-new-terraform-workspace)
2. [How to List all the Terraform workspaces](https://jhooq.com/terraform-workspaces/#2-how-to-list-all-the-terraform-workspaces)
3. [How to Show Terraform workspaces](https://jhooq.com/terraform-workspaces/#3-how-to-show-terraform-workspaces)
4. [How to Switch Terraform workspaces](https://jhooq.com/terraform-workspaces/#4-how-to-switch-terraform-workspaces)
5. [How to use the name of Current Workspace using ${terraform.workspace} Interpolation](https://jhooq.com/terraform-workspaces/#5-how-to-use-the-name-of-current-workspace-using-terraformworkspace-interpolation)
6. [Terraform workspace and terraform.tfstate, terraform.tfstate.d](https://jhooq.com/terraform-workspaces/#6-terraform-workspace-and-terraformtfstate-terraformtfstated)
7. [When to use multiple terraform workspaces?](https://jhooq.com/terraform-workspaces/#7-when-to-use-multiple-terraform-workspaces)

1. How to create a new Terraform workspace

To create a terraform workspace first you make sure that you have [installed terraform](https://jhooq.com/install-terrafrom/). After that, you simply need to run the following terraform workspace command -

terraform workspace new dev

*BASH*

Here are few points to keep in mind -

1. To create a workspace you must type in the keyword workspace after terraform
2. You must also type in new for creating a new workspace
3. The third and last thing is your workspace name, in the above command I have taken the workspace name as dev

1.1 Create one more workspace with name test

To get a better understanding of workspace and how it isolates then I would highly recommend creating one more workspace where we run the terraform commands respectively on each workspace without affecting their states

Here is the command for creating another workspace with the name test

terraform workspace new test

*BASH*

2. How to List all the Terraform workspaces?

Now in [step 1](https://jhooq.com/terraform-workspaces/#1-how-to-create-a-new-terraform-workspace) we have seen how to create a new workspace in terraform. But then how to list all the workspaces which we have created previously.

So to list all the available workspaces use the following terraform command -

terraform workspace list

*BASH*

The above command will list down all the available workspaces including the default workspace also

terraform workspace list

The active workspace is always prefixed with a star.

3. How to Show Terraform workspaces?

Now we know [how to create a terraform workspace](https://jhooq.com/terraform-workspaces/#1-how-to-create-a-new-terraform-workspace) and [how to list all the available workspace](https://jhooq.com/terraform-workspaces/#2-how-to-list-all-the-terraform-workspaces) in terraform.

*But is there a way to show active workspace*?

Terraform provides a command terraform workspace show which can help you to show the active running workspace in which you are working.

terraform workspace show

Here is one example screenshot below which will tell you that test is active running state

terraform workspace list

4. How to switch Terraform workspaces?

It might be possible that you have created multiple workspaces to isolate your infrastructure code and you want to switch between the workspaces to apply your infrastructure changes.

So to switch the workspace you should following terraform command -

terraform workspace select test

The above terraform command will make a switch to test workspace. In your case, you simply need to put your workspace name to make the switch between the workspaces

5. How to use the name of Current Workspace using ${terraform.workspace} Interpolation?

While working with longer terraform infrastructure code it often needed to name or tag your resources based on the environment profile such as *dev, test, stage, prod, etc.*

With the help of ${terraform.workspace} interpolation sequence you can use the current workspace name to [define terraform locals](https://jhooq.com/how-to-use-terraform-locals/).

Here is a simple example of provisioning a simple EC2 instance -

1. I have defined a terraform local instance\_name
2. The terraform local have a name prefixed with the current workspace name .i.e. - "${terraform.workspace}-instance"
3. And the terraform local is used to tag the aws\_instance

provider "aws" {

region = "ap-south-2"

access\_key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

secret\_key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

}

locals {

instance\_name = "${terraform.workspace}-instance"

}

resource "aws\_instance" "ec2\_example" {

ami = "ami-09d8b83b58eabf58b"

instance\_type = var.instance\_type

tags = {

Name = local.instance\_name

}

}

*BASH*

6. Terraform workspace and terraform.tfstate, terraform.tfstate.d

Whenever you work with terraform workspace and when you create multiple workspaces then you will get one directory created for each workspace inside your terraform project.

Here is an example where I have created two workspace - *dev, test*

terraform workspace terraform.tfstate and terraform.tfstate.d

So with each workspace, you end up with its own terraform.tfstate and terraform.tfstate.d file which will help you to separate and isolate the infrastructure behavior based on your configuration settings.

7. When to use multiple terraform workspaces?

Here are some key points where I felt you should use terraform workspaces -

1. When you feel that you need a parallel, distinct copy of your infrastructure which you can test and verify in the development, test, and staging environment before putting it into the production environment.
2. Secondly terraform workspaces acts as feature branch. If you have ever worked with a versioning tool then you might be familiar with main or master where every developer merges their code from the feature branch. So assume the default workspace as your main or master and the workspaces` as your featured branch where you will test and verify your changes.
3. Do not presume that with terraform workspaces you can separate or decompose your infrastructure component. Terraform workspaces are not meant to decompose your infrastructure. If you really want to separate or decompose your infrastructure then I would recommend using a separate set of configuration and backend to achieve that.
4. If you have your infrastructure where each deployment requires you to put different setup credentials then you should not use Terraform workspaces for that.

Variable.tf

Variable “instance\_type”{}

dev.tfvars

instance\_type = “t2.micro”

test.tfvars

instance\_type = “t2.medium”

terraform pan –var-file=”dev.tfvars”