
Line Arguments

Objective:

Learn how to pass values to Terraform variables using command line arguments.

Prerequisites:

- Terraform installed on your machine.
- Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-cli-variables  
cd terraform-cli-variables
```

2. Create Terraform Configuration Files:

- Create a file named main.tf:

main.tf

```
provider "aws" {  
  region = var.region  
}  
  
resource "aws_instance" "example" {  
  ami      = var.ami  
  instance_type = var.instance_type  
}
```

Create a file named variables.tf: #

variables.tf

```
variable "region" {  
  description = "AWS region"  
  default    = "us-west-2"  
}  
  
variable "ami" {  
  description = "AMI ID"  
  default    = "ami-0c55b159cbfafa1fo"  
}  
  
variable "instance_type" {  
  description = "EC2 Instance Type"  
  default    = "t2.small"  
}
```

3. Use Command Line Arguments:

- Open a terminal and navigate to your Terraform project directory.
- Run the terraform init command:

```
terraform init
```

- Run the terraform apply command with command line arguments to set variable values:

```
terraform apply -var 'region=us-east-1' -var 'ami=ami-12345678' -var  
'instance_type=t2.micro'
```

- Adjust the values based on your preferences.

4. Test and Verify:

- Observe how the command line arguments dynamically set the variable values during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.

5. Clean Up:

After testing, you can clean up resources:

```
terraform destroy
```

Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to use command line arguments to set variable values dynamically during the terraform apply process. It allows you to customize your Terraform deployments without modifying the configuration files directly. Experiment with different variable values and observe how command line arguments impact the infrastructure provisioning process.

```
main.tf x instance.tf var.tf
main.tf
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.31.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "ap-south-1"
12   access_key = "AKIAWCHJE0QYJV3PQHM2"
13   secret_key = "3mbL8074QL9vqQXa2V2701/r9z11UeHfnAk7KfaE"
14 }
```

```
main.tf instance.tf x var.tf
instance.tf
1 resource "aws_instance" "exp-4" {
2   instance_type = var.instance_type
3   ami = var.ami_id
4   count = 1
5   tags = {
6     Name = "exp4-B3"
7   }
8 }
```

```
main.tf instance.tf var.tf x
var.tf
1 variable "instance_type" {
2   type = string
3 }
4 variable "ami_id" {
5   type = string
6   default = "ami-00952f27cf14db9cd"
7 }
```

```
C:\Users\anu39\Terraform-Script1>terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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.

```
C:\Users\anu39\Terraform-Script1>terraform validate
Success! The configuration is valid.
```

```
C:\Users\anu39\Terraform-Script1>terraform plan
var.instance_type
  Enter a value: t2.micro
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# aws_instance.exp-4[0] will be created
+ resource "aws_instance" "exp-4" {
  + ami              = "ami-00952f27cf14db9cd"
  + arn              = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone = (known after apply)
  + cpu_core_count   = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop  = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized     = (known after apply)
  + get_password_data = false
  + host_id           = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id                = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)
  + instance_state     = (known after apply)
  + instance_type      = "t2.micro"
  + ipv6_address_count = (known after apply)
  + ipv6_addresses     = (known after apply)
  + key_name           = (known after apply)
  + monitoring         = (known after apply)
  + outpost_arn        = (known after apply)
```

```

+ outpost_arn                = (known after apply)
+ password_data              = (known after apply)
+ placement_group            = (known after apply)
+ placement_partition_number = (known after apply)
+ primary_network_interface_id = (known after apply)
+ private_dns                = (known after apply)
+ private_ip                 = (known after apply)
+ public_dns                 = (known after apply)
+ public_ip                  = (known after apply)
+ secondary_private_ips      = (known after apply)
+ security_groups            = (known after apply)
+ source_dest_check          = true
+ spot_instance_request_id   = (known after apply)
+ subnet_id                  = (known after apply)
+ tags                       = {
+   "Name" = "exp4-B3"
+ }
+ tags_all                   = {
+   "Name" = "exp4-B3"
+ }
+ tenancy                    = (known after apply)
+ user_data                  = (known after apply)
+ user_data_base64          = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids     = (known after apply)
}

```

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

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.

```

C:\Users\anu39\Terraform-Script1>terraform apply
var.instance_type
  Enter a value: t2.micro

```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```

# aws_instance.exp-4[0] will be created
+ resource "aws_instance" "exp-4" {
+   ami                        = "ami-00952f27cf14db9cd"
+   arn                       = (known after apply)
+   associate_public_ip_address = (known after apply)
+   availability_zone          = (known after apply)
+   cpu_core_count             = (known after apply)
+   cpu_threads_per_core       = (known after apply)
+   disable_api_stop           = (known after apply)
+   disable_api_termination    = (known after apply)
+   ebs_optimized              = (known after apply)
+   get_password_data          = false
+   host_id                    = (known after apply)
+   host_resource_group_arn     = (known after apply)
+   iam_instance_profile        = (known after apply)
+   id                         = (known after apply)
+   instance_initiated_shutdown_behavior = (known after apply)
+   instance_lifecycle         = (known after apply)
+   instance_state             = (known after apply)
+   instance_type              = "t2.micro"
+   ipv6_address_count          = (known after apply)
+   ipv6_addresses             = (known after apply)
+   key_name                    = (known after apply)
+   monitoring                  = (known after apply)
+   outpost_arn                = (known after apply)
+   password_data              = (known after apply)
+   placement_group            = (known after apply)
+   placement_partition_number = (known after apply)
+   primary_network_interface_id = (known after apply)
+   private_dns                = (known after apply)
+   private_ip                 = (known after apply)
+   public_dns                 = (known after apply)
+   public_ip                  = (known after apply)
+   secondary_private_ips      = (known after apply)
+   security_groups            = (known after apply)
+   source_dest_check          = true
+   spot_instance_request_id   = (known after apply)
+   subnet_id                  = (known after apply)

```

```

+ secondary_private_ips = (known after apply)
+ security_groups        = (known after apply)
+ source_dest_check      = true
+ spot_instance_request_id = (known after apply)
+ subnet_id              = (known after apply)
+ tags                    = {
  + "Name" = "exp4-B3"
}
+ tags_all                = {
  + "Name" = "exp4-B3"
}
+ tenancy                  = (known after apply)
+ user_data                = (known after apply)
+ user_data_base64        = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids   = (known after apply)
}

```

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

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.exp-4[0]: Creating...
aws_instance.exp-4[0]: Still creating... [10s elapsed]
aws_instance.exp-4[0]: Still creating... [20s elapsed]
aws_instance.exp-4[0]: Creation complete after 22s [id=i-0ae4852b180b41165]

```

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

```

C:\Users\anu39\Terraform-Script1>terraform destroy
var.instance_type
Enter a value: t2.micro

```

```
aws_instance.exp-4[0]: Refreshing state... [id=i-0ae4852b180b41165]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

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

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

Enter a value: yes

```

aws_instance.exp-4[0]: Destroying... [id=i-0ae4852b180b41165]
aws_instance.exp-4[0]: Still destroying... [id=i-0ae4852b180b41165, 10s elapsed]
aws_instance.exp-4[0]: Still destroying... [id=i-0ae4852b180b41165, 20s elapsed]
aws_instance.exp-4[0]: Still destroying... [id=i-0ae4852b180b41165, 30s elapsed]
aws_instance.exp-4[0]: Destruction complete after 31s

```

Destroy complete! Resources: 1 destroyed.

```
C:\Users\anu39\Terraform-Script1>
```

Instances (1) <small>Info</small>									
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				Any state		< 1 > ⚙			
<input type="checkbox"/>	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	
<input type="checkbox"/>	exp4-B3	i-0ae4852b180b41165	Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-13-235-134	

EC2 > Instances > i-0ae4852b180b41165

Instance summary for i-0ae4852b180b41165 (exp4-B3) Info

Updated less than a minute ago

Instance ID
i-0ae4852b180b41165 (exp4-B3)

IPv6 address
-

Hostname type
IP name: ip-172-31-47-202.ap-south-1.compute.internal

Answer private resource DNS name
-

Auto-assigned IP address
13.235.134.88 [Public IP]

IAM Role
-

IMDSv2
Required

Public IPv4 address
13.235.134.88 [open address](#)

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-47-202.ap-south-1.compute.internal

Instance type
t2.micro

VPC ID
vpc-0e4fe532fd5496e97

Subnet ID
subnet-0bf96d4af079e8d53

Private IPv4 addresses
172.31.47.202

Public IPv4 DNS
ec2-13-235-134-88.ap-south-1.compute.amazonaws.com [open address](#)

Elastic IP addresses
-

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
[Learn more](#)

Auto Scaling Group name
-

Instances (1) Info

Any state Launch instances

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
<input type="checkbox"/>	exp4-B3	i-0ae4852b180b41165	Shutting-d...	t2.micro	-	View alarms	ap-south-1a	-

Second Method: -

```
C:\Users\anu39\Terraform-Script1>terraform plan -var "instance_type=t2.micro"
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# aws_instance.exp-4[0] will be created
+ resource "aws_instance" "exp-4" {
  + ami              = "ami-08952f27cf14db9cd"
  + arn              = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone = (known after apply)
  + cpu_core_count   = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop  = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized     = (known after apply)
  + get_password_data = false
  + host_id           = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id                = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)
  + instance_state     = (known after apply)
  + instance_type      = "t2.micro"
  + ipv6_address_count = (known after apply)
  + ipv6_addresses     = (known after apply)
  + key_name           = (known after apply)
  + monitoring         = (known after apply)
  + outpost_arn        = (known after apply)
  + password_data      = (known after apply)
  + placement_group     = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns         = (known after apply)
  + private_ip          = (known after apply)
  + public_dns          = (known after apply)
}
```