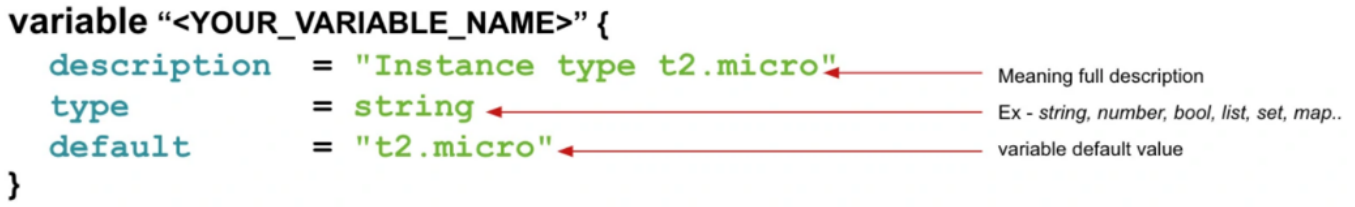
TERRAFORM CLASS 2

TERRAFORM VARIABLE TYPES:



Input Variables serve as parameters for a Terraform module, so users can customize behavior without editing the source.

Output Values are like return values for a Terraform module. Local Values are a convenience feature for assigning a short name to an expression.

TERRAFORM STRING:

It seems like your question might be incomplete or unclear. If you are looking for information about working with strings in Terraform, I can provide some guidance.

In Terraform, strings are used to represent text data and can be manipulated using various functions and operators

```
provider "aws" {
   region     = "ap-south-1"
   access_key = "AKIAWW7WL2JMJKCCMORC"
   secret_key = "DraPAxLZinm+ONtvchniWNG91MpqkwMvyrJVZo/B"
}

resource "aws_instance" "ec2_example" {

   ami           = "ami-0767046d1677be5a0"
   instance_type =  var.instance_type

   tags = {
           Name = "Terraform EC2"
   }
}

variable "instance_type" {
   description = "Instance type t2.micro"
   type        = string
   default     = "t2.micro"
}
```

TERRAFORM NUMBER: The number type can represent both whole numbers and fractional values.

```
provider "aws" {
   region     = "ap-south-1"
   access_key = "AKIAWW7WL2JMJKCCMORC"
   secret_key = "DraPAxLZinm+ONtvchniWNG91MpqkwMvyrJVZo/B"
}

resource "aws_instance" "ec2_example" {

   ami           = "ami-0af25d0df86db00c1"
   instance_type = "t2.micro"
   count  =  var.instance_count

   tags = {
           Name = "Terraform EC2"
   }
}

variable "instance_count" {
   description = "Instance type count"
   type        = number
   default     = 2
}
```

TERRAFORM BOOLEAN: a boolean represents a binary value indicating either true or false. Booleans are used to express logical conditions, make decisions, and control the flow of Terraform configurations. In HashiCorp Configuration Language (HCL), which is used for writing Terraform configurations, boolean values are written as true or false.

variable "enable_feature" {

type = bool

default = true

}

resource "aws_instance" "example" {

ami = "ami-abc123"

instance_type = var.enable_feature ? "t2.micro" : "t2.nano"

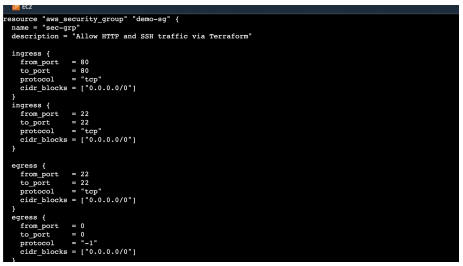
}

write a terraform code to launch 2 instances with different names

resource "aws_instance" "example_instances" {

```
  count         = 2

  ami           = "ami-0c55b159cbfafe1f0"  #Replace with your preferred AMI

  instance_type = "t2.micro"


  tags = {

    Name = var.instance_names[count.index]

  }

}


variable "instance_names" {

  type    = list(string)

  default = ["web-server-1", "web-server-2"]

}
```

LAUNCH EC2 INSTANCE WITH SG:



TERRAFORM REFRESH:

Lets assume that we created EC2 and S3 bucket using terraform, but i deleted that s3 bucket manually. now what will happen?

you can see, i have created 2 resources by using the below code and both services are running.

```
resource "aws_instance" "one" {
ami = "ami-02d7fd1c2af6eead0"
instance_type = "t2.micro"
tags = {
Name = "terraform-instance"
}
}

resource "aws_s3_bucket" "two" {
bucket = "mustafa.terraform.bucket.refresh"
}
```

here is my state list

```
[root@ip-172-31-19-166 mustafa]# terraform state list
aws_instance.one
aws_s3_bucket.two
```

now i will delete s3 bucket manually, now if i perform the same command **terraform state list**

it is giving both s3 and ec2 details

```
[root@ip-172-31-19-166 mustafa]# terraform state list
aws_instance.one
aws_s3_bucket.two
```

Now if you want to update the state list perform this command **terraform apply -refresh-only**

This will update the running resources only.

Now check the state list again

```
[root@ip-172-31-19-166 mustafa]#
[root@ip-172-31-19-166 mustafa]# terraform state list
aws_instance.one
```

now we can see it is giving only EC2 details, So if you want to create s3 bucket agian

terraform apply --auto-approve

TERRAFORM TAINT:

This command will recreate the infra in same workspace

- terraform taint aws_instance.one
- terraform apply --auto-approve

To remove the resource from taint : terraform untaint aws_instance.one

TERRAFORM REPLACE:

To recreate any service we will use terraform taint, but as per the new update instead of taint we will directly use terraform replace command.

terraform apply --auto-approve -replace="aws_instance.three”