**Terraform Configuration to Set Up AWS Infrastructure: IAM Role, S3 Bucket, and**

**EC2 Instance**

**What is terraform?**

* Terraform is an Infrastructure As code (IAC) tool that uses declarative configurational language that is Hashicorp Configurational Language (HCL) to define and provision infrastructure on a different Cloud Platform.

**What is IAM Role?**

* An IAM role (Identity and Access Management role) is a set of permissions in cloud environments, such as AWS, that defines what actions are allowed and what resources can be accessed by entities like users, applications, or services.
* Unlike a user, a role does not have long-term credentials; instead, it can be assumed temporarily by users or services. This makes roles particularly useful for granting permissions without needing to manage permanent credentials.

**What is S3?**

* Amazon S3 (Simple Storage Service) is a scalable object storage service provided by AWS (Amazon Web Services). It allows users to store and retrieve any amount of data from anywhere on the web.

**What is ec2 instance?**

* An EC2 instance (Elastic Compute Cloud instance) is a virtual server in Amazon Web Services (AWS) that allows you to run applications and services in the cloud.

**STEP 1:**

**CODE:**

provider "aws" {

region = "us-east-1"

}

resource "aws\_iam\_role" "example\_role" {

name = "examplerole"

assume\_role\_policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "ec2.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

EOF

}



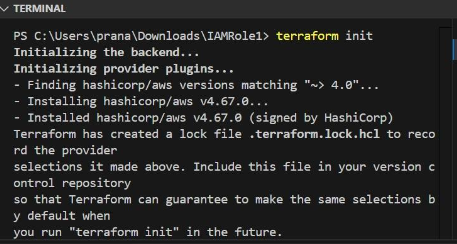
**What is Terraform Provider Block?**

* In Terraform, a “provider block” is a configuration block used to define the specific provider and its settings that Terraform will use to manage and interact with infrastructure resources.

After giving the provider and resource block

We have to check these terraform commands

* Terraform init
* Terraform plan
* Terraform validate
* Terraform apply

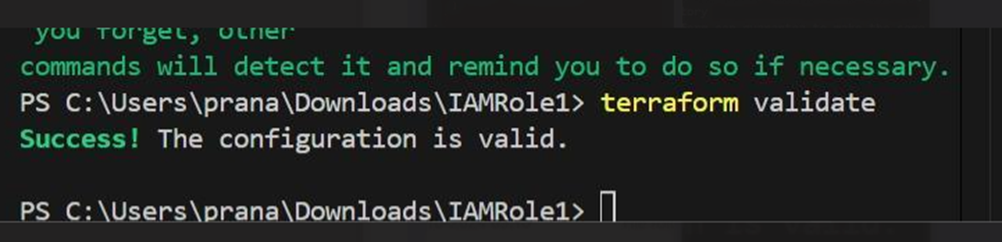


**Terraform init:** terraform init is the command used to initialize a Terraform working directory.

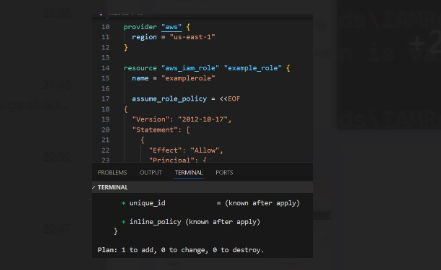


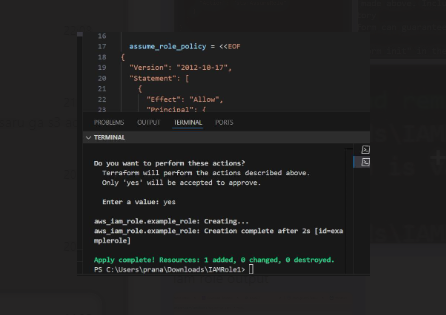
**Terraform plan**: terraform plan is a command used in Terraform to create an execution plan.

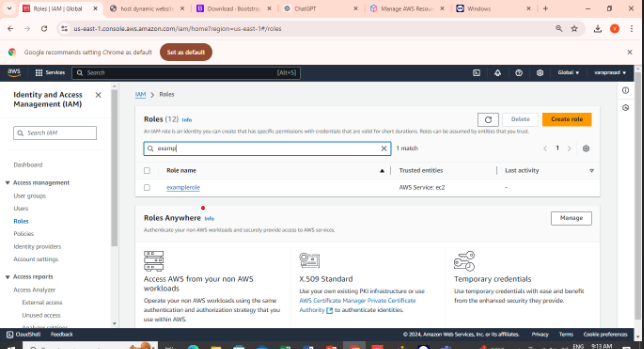
**Terraform validate**: terraform validate is a command used to check the syntax and validity of your Terraform configuration files.



**Terraform apply**: terraform apply is a command used to apply the changes required to reach the desired state of your infrastructure, as defined in your Terraform configuration files.







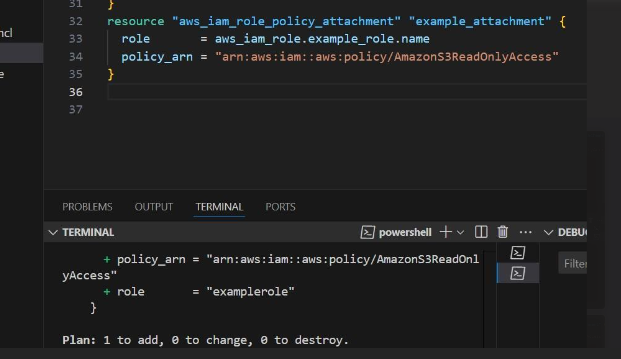
The above image shows that we are creating IAM role successfully.

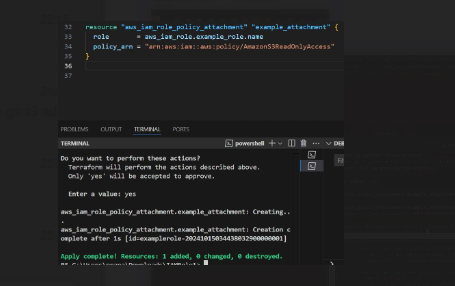
Here the below image shows we are giving IAM role policy attachment.

**CODE:**

resource "aws\_iam\_role\_policy\_attachment" "example\_attachment" {

role = aws\_iam\_role.example\_role.name

policy\_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"



The above image shows that we are giving the code iam role policy attachment then we give terraform plan and terraform apply.

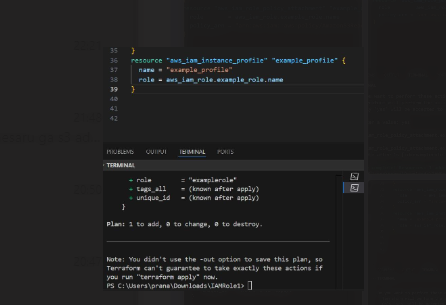
**CODE:**

resource "aws\_iam\_instance\_profile" "example\_profile" {

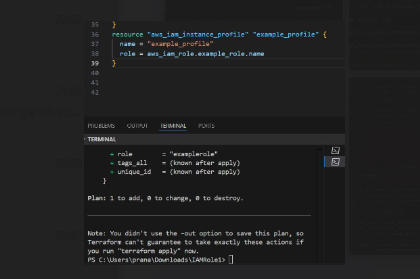
name = "example\_profile"

role = aws\_iam\_role.example\_role.name

}



Here we give the terraform plan the image is as shown below.



After the terraform plan we have to give terraform apply the image is as shown below.

**CODE:**

resource "aws\_iam\_role\_policy\_attachment" "example\_attachment" {

role = aws\_iam\_role.example\_role.name

policy\_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"

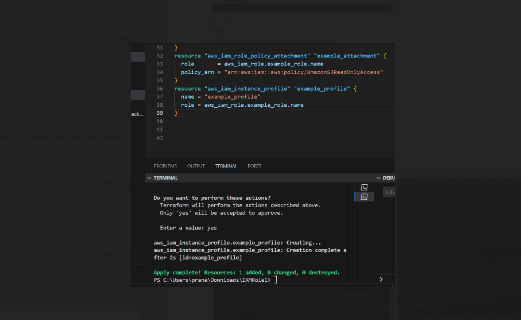
}

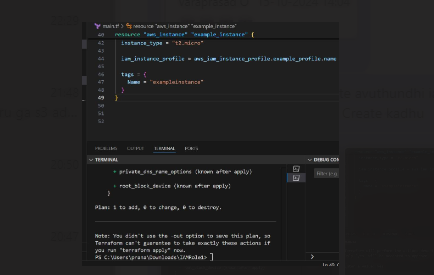
resource "aws\_iam\_instance\_profile" "example\_profile" {

name = "example\_profile"

role = aws\_iam\_role.example\_role.name

}





The above image shows that we are given the terraform plan and see the changes.

**CODE:**

resource "aws\_instance" "example\_instance" {

ami = "ami-06ca3ca175f37dd66"

instance\_type = "t2.micro"

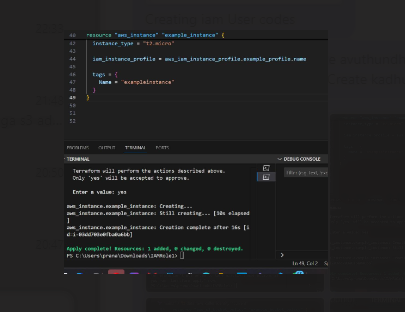
iam\_instance\_profile = aws\_iam\_instance\_profile.example\_profile.name

tags = {

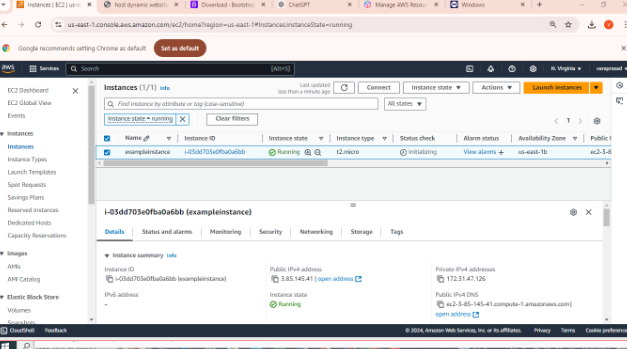
Name = "exampleinstance"

}

}



Then after we have to give the terraform apply the above image is shown.



The above image shows the instance is successfully created.

**CODE:**

resource "aws\_s3\_bucket\_policy" "example\_bucket\_policy" {

bucket = "example-bucket"

policy = jsonencode({

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::${data.aws\_caller\_identity.current.account\_id}:role/${aws\_iam\_role.example\_role.name}"

},

"Action": [

"s3:GetObject",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::example-bucket",

"arn:aws:s3:::example-bucket/\*"

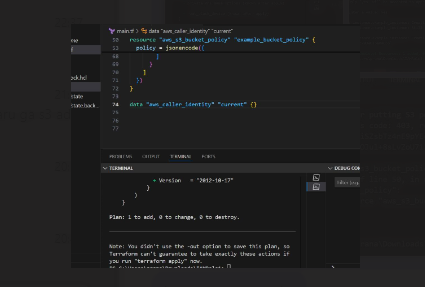
]

}

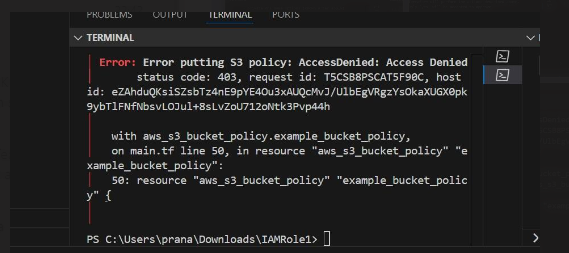
]

})

}



The above image shows the after giving the terraform plan.



Here the final output is s3 policy is access denied. Because we are giving access to read only purpose to the s3 bucket that’s why bucket is not created.