Terraform for multiple resource

Provider.tf

```
provider "aws" {
 region = "us-east-1"  # Specify your AWS region
}
```

Main.tf

VPC

```
#vpc
resource "aws_vpc" "main" {
   cidr_block = "10.0.0.0/16"
   tags = {
   Name = "MainVPC"
 }
}


#subnet
resource "aws_subnet" "main" {
   vpc_id = aws_vpc.main.id
   cidr_block              = "10.0.1.0/24"
   availability_zone       = "us-east-1a"
   tags = {
   Name = "MainSubnet"
 }
}
# Subnet in us-east-1b
resource "aws_subnet" "secondary" {
 vpc_id                  = aws_vpc.main.id
 cidr_block              = "10.0.2.0/24"
 availability_zone       = "us-east-1b"
 map_public_ip_on_launch = true
 tags = {
   Name = "SecondarySubnet"
 }
}
```

Ec2

```
#ec2 instance

resource "aws_instance" "main" {
   ami = "ami-0453ec754f44f9a4a"
   instance_type = "t2.micro"
   subnet_id = aws_subnet.main.id
   tags = {
   Name = "myec2"
 }
 }
```

S3 with folder

```
#s3

# S3 Bucket with Folder
resource "aws_s3_bucket" "example" {
 bucket = "example-bucket-terraform-qqqqqqqqqqq"
 tags = {
   Name = "ExampleBucket"
 }
}

resource "aws_s3_object" "folder" {
   bucket = "awsaws_s3_bucket.example.id"
   key    = "folder/"
   depends_on = [aws_s3_bucket.example]

}
```

Rds

```
#rds
# DB Subnet Group
resource "aws_db_subnet_group" "main" {
 name       = "main-db-subnet-group"
 subnet_ids = [aws_subnet.main.id, aws_subnet.secondary.id]  # Include both subnets
 tags = {
   Name = "MainDBSubnetGroup"
 }
}
resource "aws_db_instance" "default" {
 allocated_storage    = 10
 db_name              = "mydb"
 engine               = "mysql"
 engine_version       = "8.0"
 instance_class       = "db.t3.micro"
 username             = "admin"
 password             = "admin123"
 parameter_group_name = "default.mysql8.0"
 db_subnet_group_name = aws_db_subnet_group.main.name
 skip_final_snapshot  = true
 tags = {
   Name = "ExampleRDS"
 }
}
```

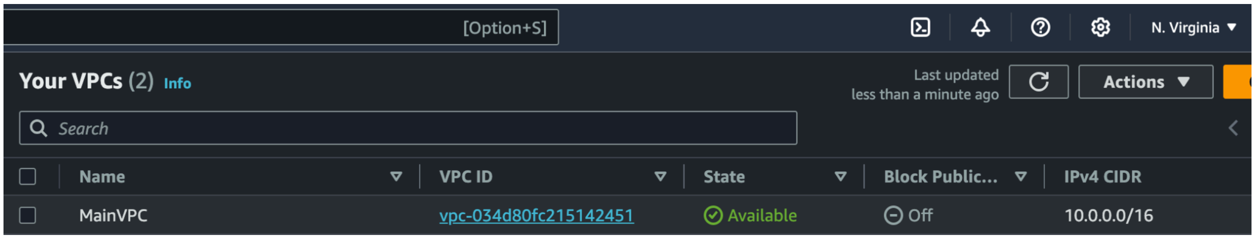
Iam role

```
#iam

resource "aws_iam_role" "test_role" {
 name = "test_role"

 assume_role_policy = jsonencode({
   Version = "2012-10-17"
   Statement = [
     {
       Action = "sts:AssumeRole"
       Effect = "Allow"
       Sid    = ""
       Principal = {
         Service = "ec2.amazonaws.com"
       }
     },
   ]
 })

 tags = {
   tag-key = "tag-value"
 }
}
```

Commands
1.terraform init
2. terraform plan
3.terraform apply
4.terraform destroy

OUTPUT

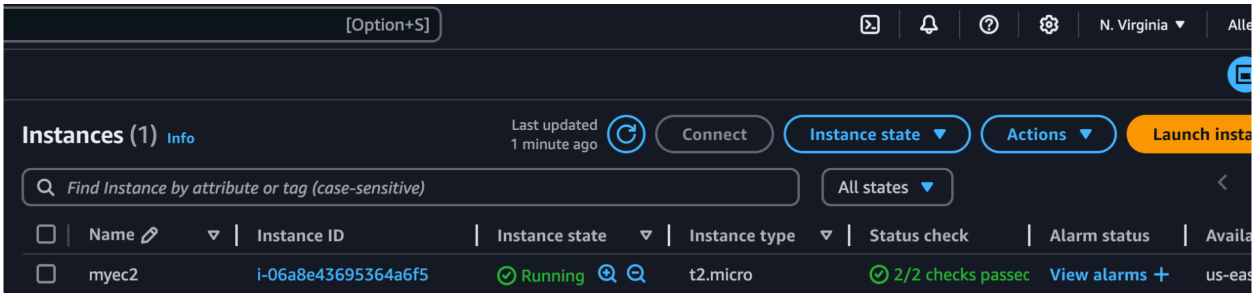
VPC



Your VPCs (2)

□	Name	VPC ID	State	Block Public...	IPv4 CIDR
□	MainVPC	vpc-034d80fc215142451	✓ Available	⊖ Off	10.0.0.0/16

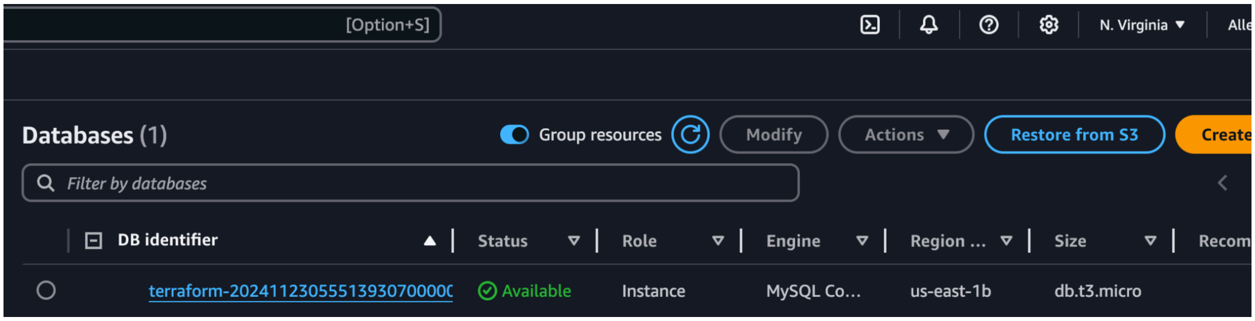
Ec2



Instances (1)

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availa
□	myec2	i-06a8e43695364a6f5	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-eas

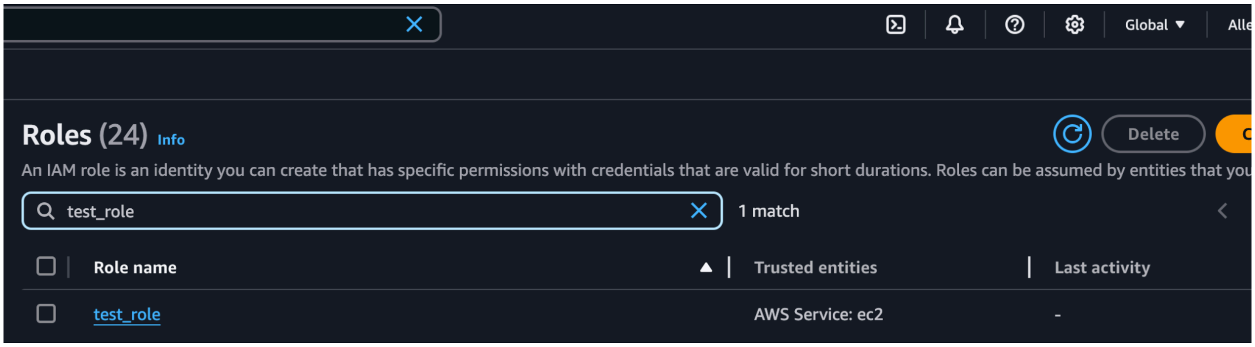
Rds



Databases (1)

	DB identifier	Status	Role	Engine	Region ...	Size	Recom
○	terraform-2024112305551393070000C	✓ Available	Instance	MySQL Co...	us-east-1b	db.t3.micro	

Iam role



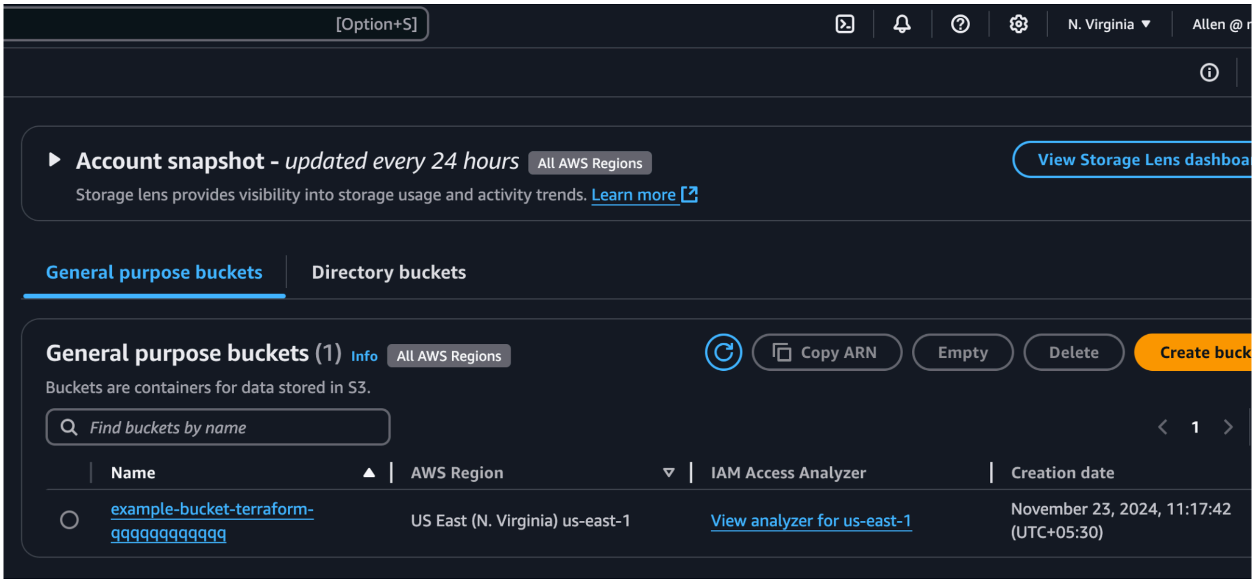
Roles (24) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you

test_role ✕ 1 match

□	Role name ▲	Trusted entities	Last activity
□	test_role	AWS Service: ec2	-

S3



▶ **Account snapshot -** *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

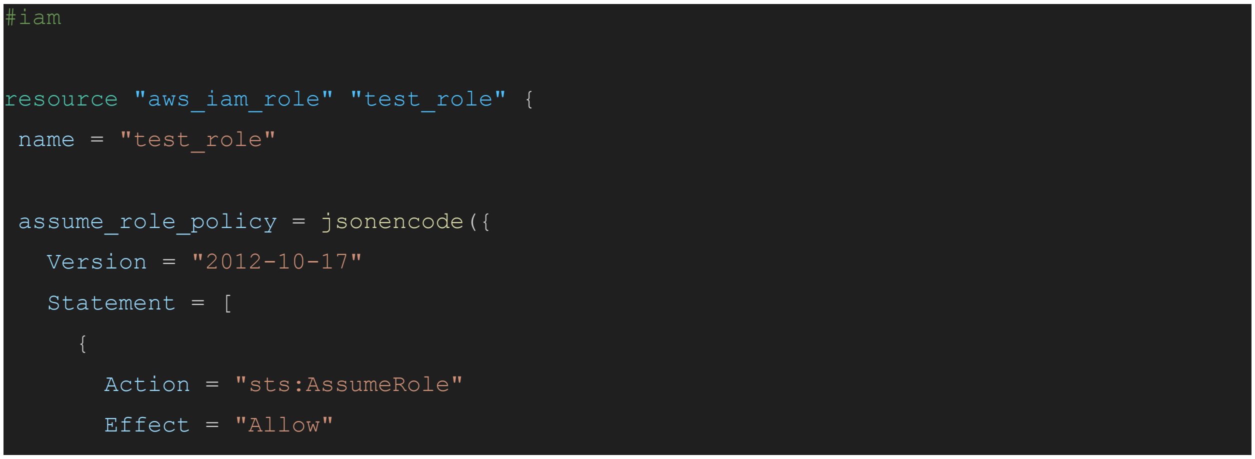
General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

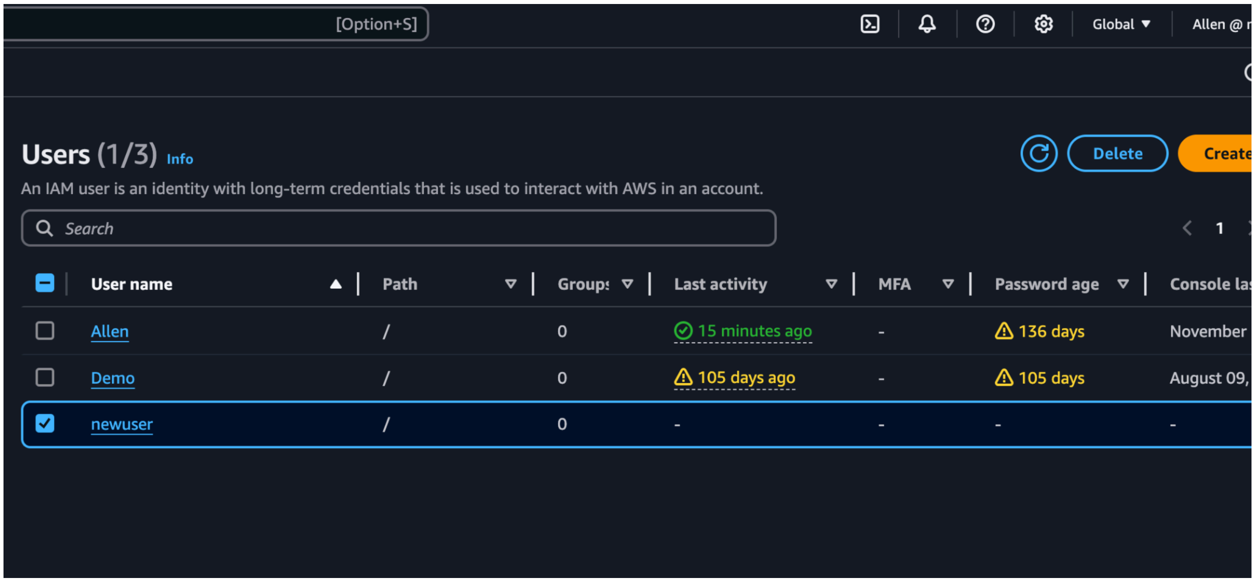
Find buckets by name

	Name ▲	AWS Region ▽	IAM Access Analyzer	Creation date
○	example-bucket-terraform-qqqqqqqqqqqq	US East (N. Virginia) us-east-1	View analyzer for us-east-1	November 23, 2024, 11:17:42 (UTC+05:30)

Iam user



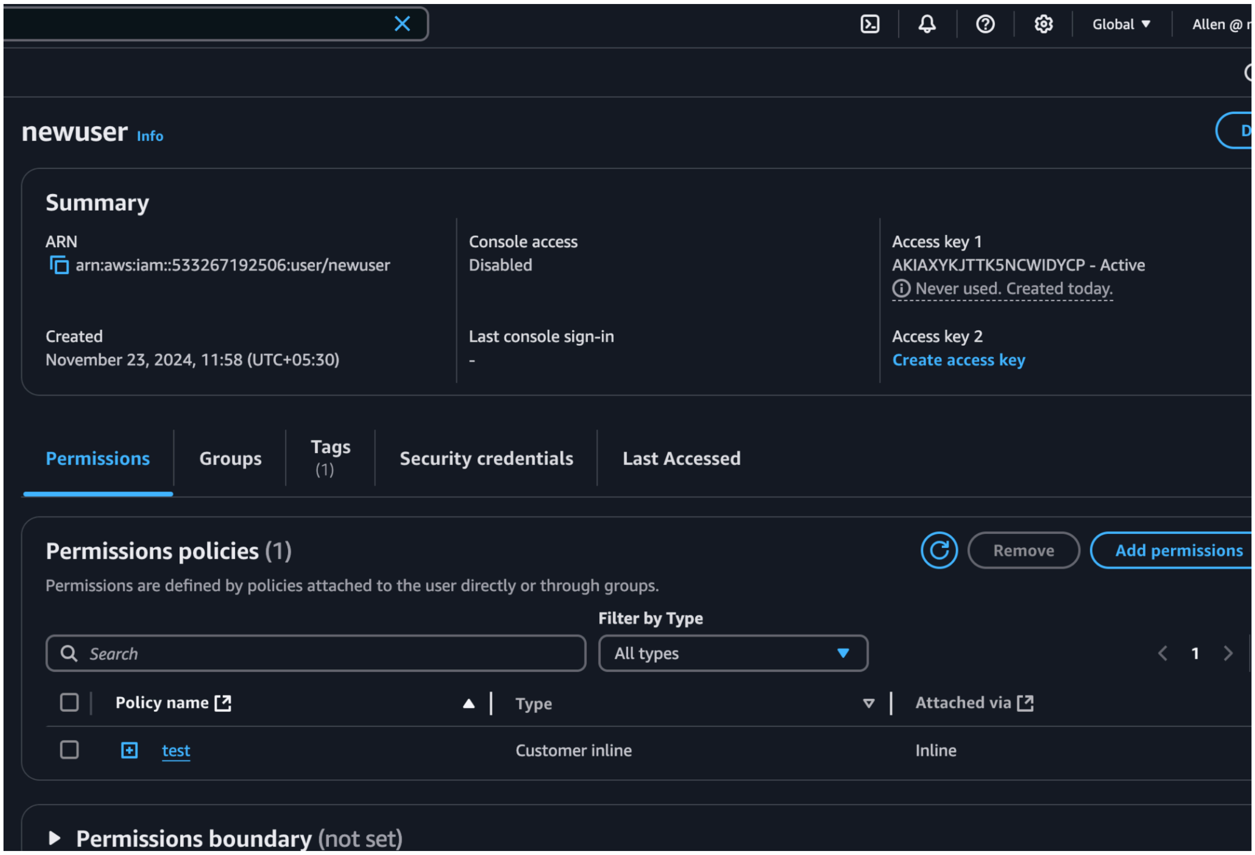
```
#iam

resource "aws_iam_role" "test_role" {
 name = "test_role"

 assume_role_policy = jsonencode({
   Version = "2012-10-17"
   Statement = [
     {
       Action = "sts:AssumeRole"
       Effect = "Allow"
       Sid    = ""
       Principal = {
         Service = "ec2.amazonaws.com"
       }
     },
   ]
 })

 tags = {
   tag-key = "tag-value"
 }
}

#iam user
resource "aws_iam_user" "lb" {
 name = "newuser"

 tags = {
   tag-key = "tag-value"
 }
}

resource "aws_iam_access_key" "lb" {
 user = aws_iam_user.lb.name
}

data "aws_iam_policy_document" "lb_ro" {
 statement {
   effect    = "Allow"
   actions   = ["ec2:*", "s3:*", "rds:*", "vpc:*"]
   resources = ["*"]
 }
}

resource "aws_iam_user_policy" "lb_ro" {
 name   = "test"
 user   = aws_iam_user.lb.name
 policy = data.aws_iam_policy_document.lb_ro.json
}
```



Users (1/3) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

▣	User name ▲	Path ▽	Groups ▽	Last activity ▽	MFA ▽	Password age ▽	Console las
□	Allen	/	0	✓ 15 minutes ago	-	⚠ 136 days	November
□	Demo	/	0	⚠ 105 days ago	-	⚠ 105 days	August 09,
☑	newuser	/	0	-	-	-	-



newuser Info

Summary

ARN
arn:aws:iam::533267192506:user/newuser

Created
November 23, 2024, 11:58 (UTC+05:30)

Console access
Disabled

Last console sign-in
-

Access key 1
AKIAXYKJTTK5NCWIDYCP - Active
ⓘ Never used. Created today.

Access key 2
Create access key

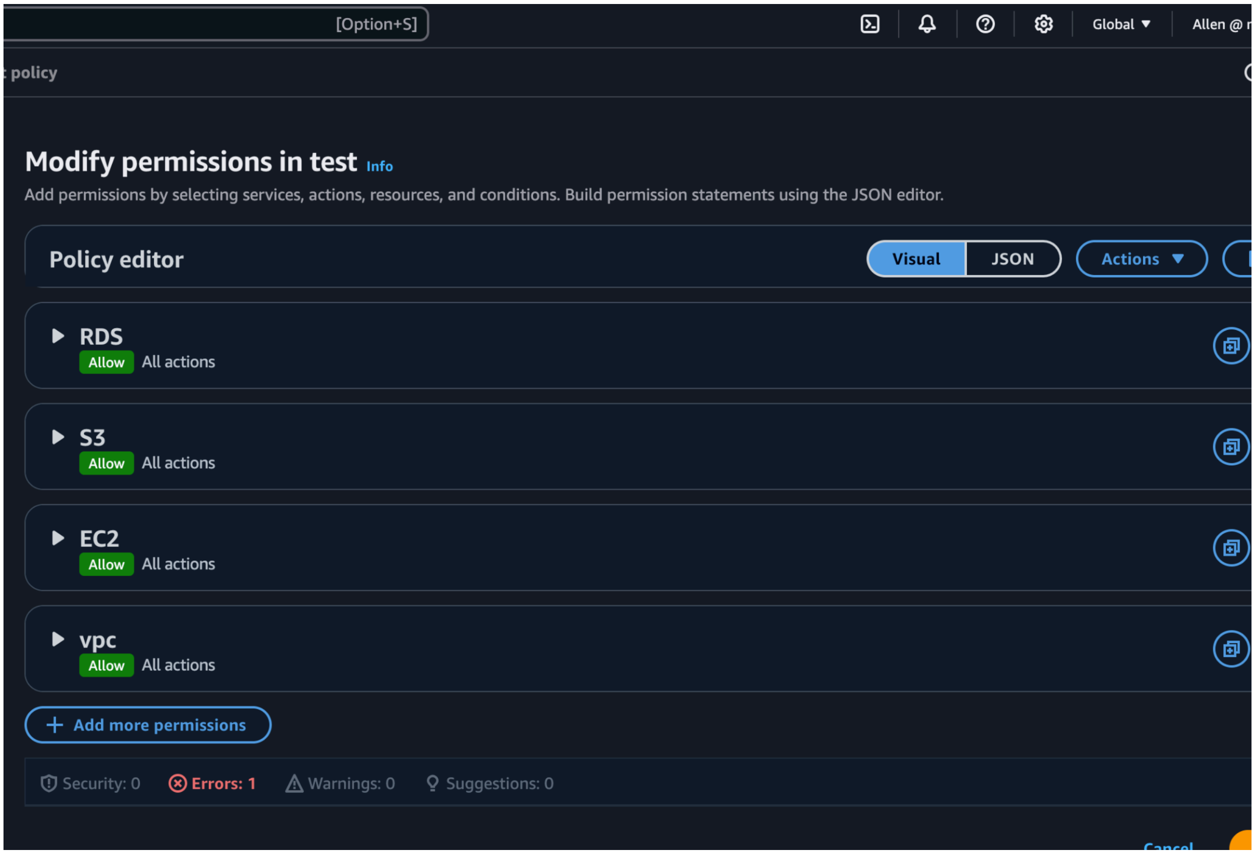
Permissions policies (1)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type
All types ▼

□	Policy name ▲	Type ▽	Attached via
□	⊞ test	Customer inline	Inline

▶ Permissions boundary (not set)



Modify permissions in test Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

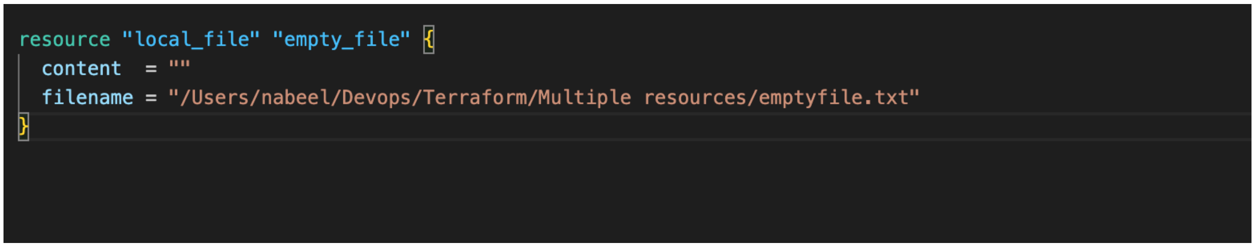
RDS
Allow All actions

S3
Allow All actions

EC2
Allow All actions

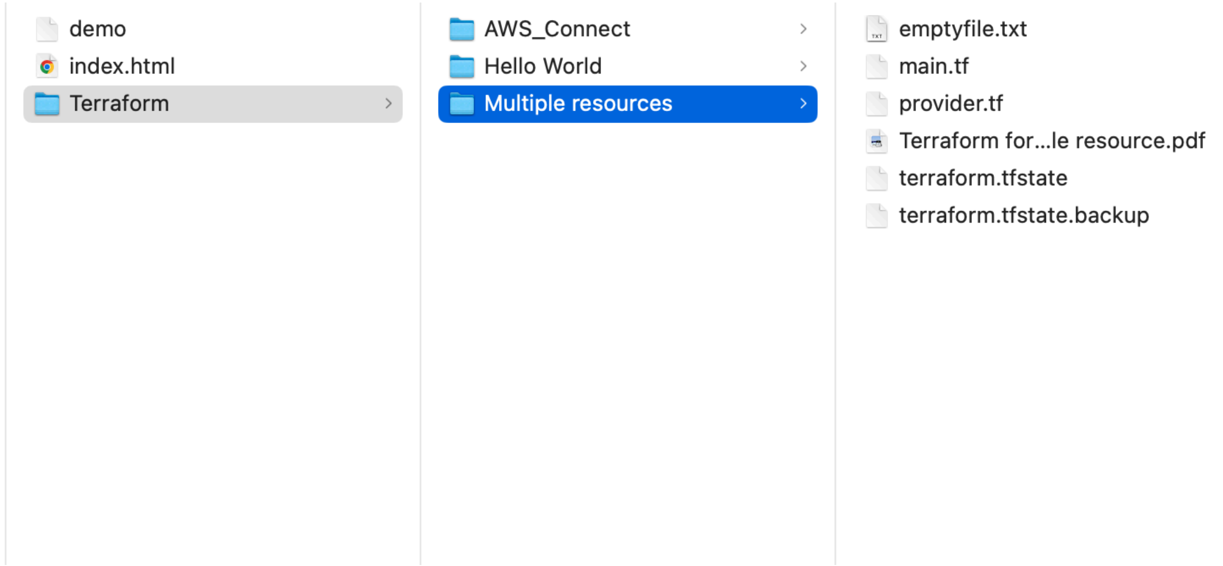
vpc
Allow All actions

Make empty file on local



```
resource "local_file" "empty_file" {
  content  = ""
  filename = "/Users/nabeel/Devops/Terraform/Multiple resources/emptyfile.txt"
}
```

```
terraform init -upgrade
terraform plan -target=local_file.empty_file
terraform apply -target=local_file.empty_file
```



demo
index.html
Terraform

AWS_Connect
Hello World
Multiple resources

emptyfile.txt
main.tf
provider.tf
Terraform for...le resource.pdf
terraform.tfstate
terraform.tfstate.backup