Towards AWS
ALL THINGS AWS

Following

733 Followers

Creating EFS File System And Mount Targets Using Terraform

Vinayak Pandey 1 day ago · 3 min read

In this post, we'll see how we can create EFS File System and Mount Targets using Terraform.

Note: We'll run our setup in the us-east-1 region. Terraform and AWS credentials must be configured before proceeding with the execution.

Step 1: Create the following terraform files:

efs.tf

```
resource "aws_efs_file_system" "efs" {
   creation_token = "efs"
   performance_mode = "generalPurpose"
   throughput_mode = "bursting"
   encrypted = "true"
 tags = {
     Name = "EFS"
   }
 }


resource "aws_efs_mount_target" "efs-mt" {
   count = length(data.aws_availability_zones.available.names)
   file_system_id  = aws_efs_file_system.efs.id
   subnet_id = aws_subnet.subnet[count.index].id
   security_groups = [aws_security_group.efs.id]
 }
```

network.tf

```
data "aws_availability_zones" "available" {}

resource "aws_vpc" "vpc" {
   cidr_block = "10.0.0.0/16"
   enable_dns_hostnames = true
   enable_dns_support = true
   tags= {
     Name = "test-env"
   }
 }

resource "aws_internet_gateway" "internet_gateway" {
  vpc_id = aws_vpc.vpc.id
  tags = {
    Name = "Internet_Gateway"
  }
}

 resource "aws_subnet" "subnet" {
   count=length(data.aws_availability_zones.available.names)
   cidr_block = cidrsubnet(aws_vpc.vpc.cidr_block, 8, count.index)
   vpc_id = aws_vpc.vpc.id
   availability_zone =
data.aws_availability_zones.available.names[count.index]
 }


resource "aws_route_table" "public" {
  vpc_id = aws_vpc.vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.internet_gateway.id
  }

  tags = {
    Name = "Public Route Table"
  }
  depends_on = [aws_internet_gateway.internet_gateway]
}

resource "aws_route_table_association" "vpc_public_assoc" {
  count          =
length(data.aws_availability_zones.available.names)
  subnet_id      = aws_subnet.subnet.*.id[count.index]
  route_table_id = aws_route_table.public.id
}
```

security.tf

```
resource "aws_security_group" "ec2" {
  name        = "allow_efs"
  description = "Allow efs outbound traffic"
  vpc_id      = aws_vpc.vpc.id
  ingress {
     cidr_blocks = ["0.0.0.0/0"]
     from_port = 22
     to_port = 22
     protocol = "tcp"
   }
  egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
  }
  tags = {
    Name = "allow_efs"
  }
}

resource "aws_security_group" "efs" {
   name = "efs-sg"
   description= "Allos inbound efs traffic from ec2"
   vpc_id = aws_vpc.vpc.id

   ingress {
     security_groups = [aws_security_group.ec2.id]
     from_port = 2049
     to_port = 2049
     protocol = "tcp"
   }

   egress {
     security_groups = [aws_security_group.ec2.id]
     from_port = 0
     to_port = 0
     protocol = "-1"
   }
 }
```

ec2.tf

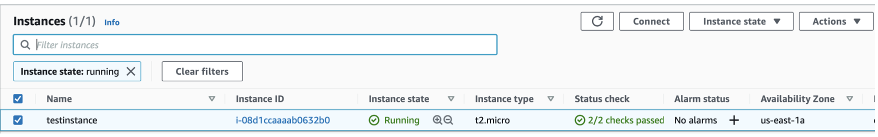
```
resource "aws_instance" "testinstance" {
    ami = "ami-087c17d1fe0178315"
    instance_type = "t2.micro"
    subnet_id = aws_subnet.subnet[0].id
    associate_public_ip_address= true
    vpc_security_group_ids = [ aws_security_group.ec2.id ]
    key_name="efs"
    tags= {
        Name = "testinstance"
    }
}
```

This will create a new VPC and launch our EFS and EC2 resources there. You need to change key_name in ec2.tf and it should already be created.

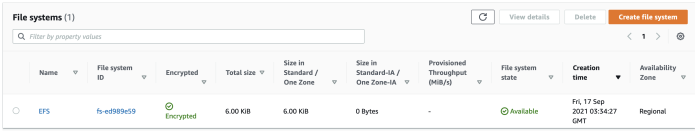
Step 2: Now we have all our terraform resources and it’s time to run following Terraform commands

```
terraform init
terraform plan
terraform apply --auto-approve
```

Step 3: Once all the resources are created successfully, you'll see our EFS file system and EC2 instance.



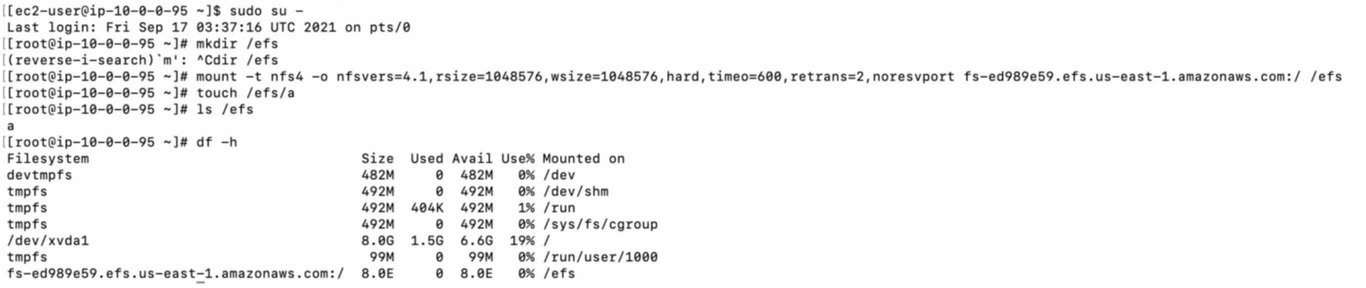
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	testinstance	i-08d1ccaaaab0632b0	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	us-east-1a



	Name	File system ID	Encrypted	Total size	Size in Standard / One Zone	Size in Standard-IA / One Zone-IA	Provisioned Throughput (MiB/s)	File system state	Creation time	Availability Zone
○	EFS	fs-ed989e59	✓ Encrypted	6.00 KiB	6.00 KiB	0 Bytes	-	✓ Available	Fri, 17 Sep 2021 03:34:27 GMT	Regional

Step 4: Now SSH in to the instance and execute following commands to mount our EFS files:

```
ssh -I <key> ec2-user@<instance_ip>
sudo su -
mkdir /efs
mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,nore
svport fs-ed989e59.efs.us-east-1.amazonaws.com:/ /efs
df -h
touch /efs/a
ls /efs
```



```
[[ec2-user@ip-10-0-0-95 ~]$ sudo su -
Last login: Fri Sep 17 03:37:16 UTC 2021 on pts/0
[[root@ip-10-0-0-95 ~]# mkdir /efs
[(reverse-i-search)`m': ^Cdir /efs
[[root@ip-10-0-0-95 ~]# mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-ed989e59.efs.us-east-1.amazonaws.com:/ /efs
[[root@ip-10-0-0-95 ~]# touch /efs/a
[[root@ip-10-0-0-95 ~]# ls /efs
a
[[root@ip-10-0-0-95 ~]# df -h
Filesystem                                Size  Used Avail Use% Mounted on
devtmpfs                                  482M     0  482M   0% /dev
tmpfs                                     492M     0  492M   0% /dev/shm
tmpfs                                     492M  404K  492M   1% /run
tmpfs                                     492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1                                8.0G  1.5G  6.6G  19% /
tmpfs                                      99M     0   99M   0% /run/user/1000
fs-ed989e59.efs.us-east-1.amazonaws.com:/  8.0E     0  8.0E   0% /efs
```

As you can see, our EFS file system was successfully mounted and is ready to use. We can also add following entry in /etc/fstab to allow auto mount after reboot.

```
<file_system_id>.us-east-1.amazonaws.com:/ /efs nfs4 defaults,_netdev
0 0
```

Get this newsletter

Emails will be sent to yegnasivasai@gmail.com.
Not you?

AWS