

# 

# Task-2 Create/launch an application using EFS as storage instead of EBS service (writing the terraform code)

**Amazon Elastic File System(EFS):-**

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth.The service manages all the file storage infrastructure for you, meaning that you can avoid the complexity of deploying, patching, and maintaining complex file system configurations.

# 

# 

**TASK:-**

1. Create Security group which allow the port 80.
2. Launch EC2 instance.

3. In this Ec2 instance use the existing key or provided key and security group which we have created in step 1.

4. Launch one Volume using the EFS service and attach it in your vpc, then mount that volume into /var/www/html

5. Developer have uploded the code into github repo also the repo has some images.

6. Copy the github repo code into /var/www/html

7. Create S3 bucket, and copy/deploy the images from github repo into the s3 bucket and change the permission to public readable.

8 Create a Cloudfront using s3 bucket(which contains images) and use the Cloudfront URL to update in code in /var/www/html.

**##### Creating Key-pair ######**

resource "tls\_private\_key" "my-task2key" {

algorithm = "RSA"

}

resource "aws\_key\_pair" "my-key" {

key\_name = "my-task2key"

public\_key = tls\_private\_key.my-task2key.public\_key\_openssh

depends\_on = [ tls\_private\_key.my-task2key,]

}

**###### Security group creation ######**

resource "aws\_security\_group" "MYSG" {

vpc\_id = "vpc-55b9a43d"

name = "MYSG"

description = "allow ssh and http traffic"

ingress {

cidr\_blocks = ["0.0.0.0/0"]

from\_port = 80

to\_port = 80

protocol = "tcp"

}

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"]

}

}

**######### Create instance ##########**

resource "aws\_instance" "my-instance" {

ami = "ami-0732b62d310b80e97"

instance\_type = "t2.micro"

key\_name = aws\_key\_pair.my-key.key\_name

security\_groups = [ "MYSG" ]

connection {

agent = "false"

type = "ssh"

user = "ec2-user"

private\_key = tls\_private\_key.my-task2key.private\_key\_pem

host = aws\_instance.my-instance.public\_ip

}

provisioner "remote-exec" {

inline = [

"sudo yum install httpd php git -y",

"sudo systemctl restart httpd",

"sudo systemctl enable httpd",

]

}

tags = {

Name = "my-instance"

}

}

**######## Create the EFS ##########**

resource "aws\_efs\_file\_system" "my-efs" {

creation\_token = "my-efs"

performance\_mode = "generalPurpose"

tags = {

Name = "my-efs"

}

}

########### Mounting EFS ############

resource "aws\_efs\_mount\_target" "my\_efs\_mount" {

file\_system\_id = aws\_efs\_file\_system.my-efs.id

subnet\_id = aws\_instance.my-instance.subnet\_id

security\_groups = ["${aws\_security\_group.MYSG.id}"]

}

**#######Mounting the EFS volume into /var/www/html#######**

resource "null\_resource" "mounting" {

depends\_on = [

aws\_efs\_mount\_target.my\_efs\_mount,

]

connection {

type = "ssh"

user = "ec2-user"

private\_key = tls\_private\_key.my-task2key.private\_key\_pem

host = aws\_instance.my-instance.public\_ip

}

provisioner "remote-exec" {

inline = [

"sudo echo ${aws\_efs\_file\_system.my-efs.dns\_name}:/var/www/html efs defaults, \_netdev 0 0 >> sudo /etc/fstab",

"sudo mount ${aws\_efs\_file\_system.my-efs.dns\_name}:/ /var/www/html",

"sudo git clone https://github.com/isha-sharma96/task1.git /var/www/html"

]

} }

**####### Creating S3 Bucket and upload the document# ######**

resource "aws\_s3\_bucket" "my\_task2\_bucket" {

bucket = "my-tk2-bucket"

acl = "private"

region = "ap-south-1"

versioning {

enabled = true }

tags = {

Name = "my-tk2-bucket"

} }

resource "aws\_s3\_bucket\_object" "mytask2bucket\_object" {

depends\_on = [aws\_s3\_bucket.my\_task2\_bucket , ]

bucket = aws\_s3\_bucket.my\_task2\_bucket.id

key = "myimg.jpg"

source = "D:/myimg.jpg"

acl = "public-read }

**######CloudFront Distribution of S3Creating bucket#######**

resource "aws\_cloudfront\_distribution" "my\_task2\_cloudfront" {

//depends\_on = [aws\_s3\_bucket.my\_task2\_bucket , null\_resource.local-1 ]

origin {

domain\_name = aws\_s3\_bucket.my\_task2\_bucket.bucket\_regional\_domain\_name

origin\_id = "S3-my-tk2-bucket"

custom\_origin\_config { http\_port = 80

https\_port = 80

origin\_protocol\_policy = "match-viewer"

origin\_ssl\_protocols = ["TLSv1", "TLSv1.1", "TLSv1.2"]

}

}

enabled = true

default\_cache\_behavior {

allowed\_methods = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]

cached\_methods = ["GET", "HEAD"]

target\_origin\_id = "S3-my-tk2-bucket"

forwarded\_values {

query\_string = false

cookies {

forward = "none"

} }

viewer\_protocol\_policy = "allow-all"

min\_ttl = 0

default\_ttl = 3600

max\_ttl = 86400 }

restrictions {

geo\_restriction {

restriction\_type = "none"

} }

viewer\_certificate {

cloudfront\_default\_certificate = true

}

}

output "domain-name" {

value = aws\_cloudfront\_distribution.my\_task2\_cloudfront.domain\_name

# terraform init























