TASK-2

Problem Statement

-->Perform the task-1 using EFS instead of EBS service on the AWS as,
Create/launch Application using Terraform

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

Step - 1 **First of all, configure your AWS profile in your local system using cmd. Fill your details & press Enter.**

Here's a snap to help you out.

```
aws configure --profile Chaitanya
            AWS Access Key ID [****************NOP3]:
            AWS Secret Access Key [****************CKLD]:
            Default region name [ap-south-1]:
            Default output format [None]:
```

Step - 2 **Next, we create a VPC**

```
            resource "aws_vpc" "lw_vpc"
              { cidr_block =
              "192.168.0.0/16"
              instance_tenancy = "default"
              enable_dns_hostnames = true
              tags = {
                Name = "lw_vpc"
              }
```

Step - 3 **subnet needs to be created**

```
            resource "aws_subnet" "lw_subnet"
              { vpc_id = "${aws_vpc.lw_vpc.id}"
              cidr_block = "192.168.0.0/24"
              availability_zone = "ap-south-1a"
              map_public_ip_on_launch = "true"
```

```
  tags = {
    Name = "lw_subnet"
  }
}
```

Step - 4 **Now, create an EFS Account.**

```
resource "aws_efs_file_system" "lw_efs"
  { creation_token = "lw_efs"
  tags = {
    Name = "lw_efs"
  }
}



resource "aws_efs_mount_target" "lw_efs_mount"
  { file_system_id = "${aws_efs_file_system.lw_efs.id}"
  subnet_id = "${aws_subnet.lw_subnet.id}"
  security_groups = [aws_security_group.default_sg.id]
}
```

Step - 5 **Next,create a Gateway & a Routing Table.**

```
resource "aws_internet_gateway" "lw_gw" {
  vpc_id = "${aws_vpc.lw_vpc.id}"
  tags = {
    Name = "lw_gw"
  }
}



resource "aws_route_table" "lw_rt"
  { vpc_id = "${aws_vpc.lw_vpc.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.lw_gw.id}"
  }

  tags = {
    Name = "lw_rt"
  }
}



resource "aws_route_table_association" "lw_rta" {
```

```
  subnet_id = "${aws_subnet.lw_subnet.id}"
  route_table_id = "${aws_route_table.lw_rt.id}"
 }
```

Step - 7 **Now launch instance**

```
resource "aws_instance" "test_ins" {
                    ami             =  "ami-063c98e690leg0bc76"
                    instance_type   =  "t2.micro"
                    key_name        =  "lw_key"
                    subnet_id     = "${aws_subnet.lw_subnet.id}"
                    security_groups = ["${aws_security_group.lw_sg.id}"]



                   connection {
                     type     = "ssh"
                     user     = "ec2-user"
                     private_key = file("C:/Users/Chait/Downloads/lw_key.pem")
                     host     = aws_instance.test_ins.public_ip
                    }


                    provisioner "remote-exec" {
                     inline = [
                       "sudo yum install amazon-efs-utils -y",
                       "sudo yum install httpd  php git -y",
                       "sudo systemctl restart httpd",
                       "sudo systemctl enable httpd",
                       "sudo setenforce 0",
                       "sudo yum -y install nfs-utils"
                     ]
                    }


                    tags = {
                     Name = "task1_os"
                    }
                  }
```

Step - 8: **Now that our instance is launched, we mount the EFS Volume to /var/www/html folder where all the code is stored.**

```
resource "null_resource" "mount"  {
  depends_on = [aws_efs_mount_target.lw_efs_mount]
  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = file("C:/Users/Chait/Downloads/lw_key.pem")
```

```
                host     = aws_instance.test_ins.public_ip

              }

            provisioner "remote-exec"

               { inline = [

                 "sudo mount -t nfs -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport
${aws_efs_file_system.lw_efs.id}.efs.ap-south-1.amazonaws.com:/ /var/www/html",

                 "sudo rm -rf /var/www/html/*",

                 "sudo git clone https://github.com/ChaizMaster/Hybrid_cloud_task1.git /var/www/html/",

                 "sudo sed -i 's/url/${aws_cloudfront_distribution.my_front.domain_name}/g' /var/www/html/index.html"

               ]

              }

            }
```

Step - 9 **We create an S3 bucket on AWS.**

```
resource "aws_s3_bucket" "lw_bucket" {

       bucket = "lw_os_bucket"

       acl    = "private"


       tags = {

         Name        = "lw_bucket"

       }

     }

      locals {

        s3_origin_id = "S3Origin"

       }
```

Step - 10 **Now that the S3 bucket has been created, we will upload the images that we had downloaded from Github**

```
resource "aws_s3_bucket_object" "object" {

         bucket = "${aws_s3_bucket.lw_bucket.id}"

         key    = "my_pic"

         source = "C:/Users/Chait/Pictures/my_pic.jpg"

         acl    = "public-read"

       }
```

Step - 11 **Now, we create a CloudFront & connect it to our S3 bucket.**

```
resource "aws_cloudfront_distribution" "my_front"

        { origin {

              domain_name = "${aws_s3_bucket.lw_bucket.bucket_regional_domain_name}"

              origin_id   = "${local.s3_origin_id}"


       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 = "${local.s3_origin_id}"

    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 =
  }
```

Step - 12 **Now, we write a terraform code**

```
resource "null_resource" "local_exec"  {



     depends_on =
         [ null_resource.moun
         t,
```

```
      provisioner "local-exec" {
          command = "start chrome  ${aws_instance.test_ins.public_ip}"
            }
    }
```

```
terraform init
```

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_cloudfront_distribution.my_front will be created
  + resource "aws_cloudfront_distribution" "my_front" {
      + active_trusted_signers         = (known after apply)
      + arn                            = (known after apply)
      + caller_reference               = (known after apply)
      + domain_name                    = (known after apply)
      + enabled                        = true
      + etag                           = (known after apply)
      + hosted_zone_id                 = (known after apply)
      + http_version                   = "http2"
      + id                             = (known after apply)
      + in_progress_validation_batches = (known after apply)
      + is_ipv6_enabled                = false
      + last_modified_time             = (known after apply)
      + price_class                    = "PriceClass_All"
      + retain_on_delete               = false
      + status                         = (known after apply)
      + wait_for_deployment            = true
```

When prompted Choose YES.

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

null_resource.git_copy: Creating...
null_resource.git_copy: Provisioning with 'local-exec'...
null_resource.git_copy (local-exec): Executing: ["cmd" "/C" "git clone https://github
null_resource.git_copy (local-exec): 'git' is not recognized as an internal or exter
null_resource.git_copy (local-exec): operable program or batch file.
```

HENCE TASK SUCCESSFULLY COMPLETED!!