Launch Web Server in single click using AWS-EFS and Terraform - Fully Automated

This is the Second Task of my Hybrid Multi-Cloud Internship under the guidance of Mr. Vimal Daga sir.

This is almost similar to my first task. The only change is today I'm going to use EFS service to make data permanent.

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. We can attach multiple instances to same Elastic File System.

Task Overview:

1. Create one VPC and subnets to connect your EFS to EC2.
2. Create a Security group that allows the port 80 and enable NFS port.
3. Launch EC2 instance and in this EC2 instance use the existing key or provided key and security group which we have created in step 2.
4. Launch one Volume using the EFS service and attach it in the same VPC we create in step 1, then mount that volume into /var/www/html.

5. Developer has uploaded the code into GitHub repo having 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.

Software required:

- **AWS CLI**
- **Terraform CLI**

Task Description:

1. Create one VPC and subnet having Internet Access:

```
provider "aws" {
  region = "ap-south-1"
  profile = "anuddeeph"
}
```

We need to give provider first to connect our terraform tool with AWS.

```
resource "aws_vpc" "efsvpc" {
  cidr_block = "10.7.0.0/16"
  enable_dns_hostnames = true
  tags = {
    Name = "main"
  }
}

resource "aws_subnet" "alpha-1a" {
  vpc_id            = "${aws_vpc.efsvpc.id}"
  availability_zone = "ap-south-1a"
  cidr_block        = "10.7.1.0/24"
  map_public_ip_on_launch = true
  tags = {
    Name = "main-1a"
  }
}


resource "aws_internet_gateway" "gw" {
  vpc_id = "${aws_vpc.efsvpc.id}"
  tags = {
    Name = "main-1a"
```

```
  }
}
resource "aws_route_table" "rt" {
  vpc_id = "${aws_vpc.efsvpc.id}"
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.gw.id}"
  }
  tags = {
    Name = "main-1a"
  }
}
resource "aws_route_table_association" "rta" {
  subnet_id      = aws_subnet.alpha-1a.id
  route_table_id = aws_route_table.rt.id
}
```

Now our VPC is ready. I am creating an Internet Gateway and routing table to make the internet available.

2. Create a Security group that allows the port 80 and also enable NFS port:

```
resource "aws_security_group" "allow_http" {
  name        = "allow_http"
  description = "Allow HTTP inbound traffic"
  vpc_id      = "${aws_vpc.efsvpc.id}"


  ingress {
    description = "Http from VPC"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = [ "0.0.0.0/0" ]
  }
  ingress {
    description = "SSH from VPC"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [ "0.0.0.0/0" ]
  }
  ingress {
    description = "NFS"
    from_port   = 2049
    to_port     = 2049
    protocol    = "tcp"
    cidr_blocks = [ "0.0.0.0/0" ]
  }
  egress {
    from_port   = 0
```

```
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }


  tags = {
    Name = "efssgroup"
  }
}
```

I am using same VPC here which we created in our first step.

3. Launch EC2 instance and in this EC2 instance use the existing key or provided key and security group which we have created in step 2:

For creating a key-pair...

#create key

```
resource "tls_private_key" "key_create" {
  algorithm = "RSA"
}


resource "aws_key_pair" "taskkey" {
  key_name   = "taskkey"
  public_key = "${tls_private_key.key_create.public_key_openssh}"
}
resource "local_file" "save_key" {
  content = tls_private_key.key_create.private_key_pem
  filename= ”taskkey.pem”

        }
```

We also save this for future reference.

```
resource "aws_instance" "webapp" {
  ami           = "ami-00b494a3f139ba61f"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  availability_zone = "ap-south-1a"
  subnet_id     = "${aws_subnet.alpha-1a.id}"
  security_groups = [ "${aws_security_group.allow_http.id}" ]
  tags = {
    Name = "efsweb"
  }
}
```

Now my instance is launched by attaching the same VPC and security group which we created already.

4. Launch one Volume using the EFS service and attach it in the same VPC we create in step 1, then mount that volume into /var/www/html:

```
resource "aws_efs_file_system" "efsdrive" {
  creation_token = "my-secure-efsdrive"
  tags = {
    Name = "Myefsdrive"
  }
}


resource "aws_efs_file_system_policy" "policy" {
  file_system_id = "${aws_efs_file_system.efsdrive.id}"
  policy = <<POLICY
{
    "Version": "2012-10-17",
    "Id": "efs-policy-wizard-c45881c9-af16-441d-aa48-0fbd68ffaf79",
    "Statement": [
        {
            "Sid": "efs-statement-20e4223c-ca0e-412d-8490-3c3980f60788",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Resource": "${aws_efs_file_system.efsdrive.arn}",
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
POLICY
}


resource "aws_efs_mount_target" "alpha" {
  file_system_id = "${aws_efs_file_system.efsdrive.id}"
  subnet_id      = "${aws_subnet.alpha-1a.id}"
  security_groups = [ "${aws_security_group.allow_http.id}" ]
}
```

Now our EFS file system is created and attach to same VPC and security groups which we created above.

For launching the webserver and mount EFS to /var/www/html, we use remote exec for this...

```
resource "null_resource" "null_vol_attach"  {
  depends_on = [
    aws_efs_mount_target.alpha,
  ]
  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key =  tls_private_key.key_create.private_key_pem
    host     = aws_instance.webapp.public_ip
  }
  provisioner "remote-exec" {
    inline = [
      "sleep 30",
      "sudo yum install -y httpd git php amazon-efs-utils nfs-utils",
      "sudo systemctl start httpd",
      "sudo systemctl enable httpd",
      "sudo chmod ugo+rw /etc/fstab",
      "sudo echo '${aws_efs_file_system.efsdrive.id}:/ /var/www/html efs
tls,_netdev' >> /etc/fstab",
      "sudo mount -a -t efs,nfs4 defaults",
      "sudo rm -rf /var/www/html/*",
      "sudo git clone https://github.com/Anuddeeph/HMCTask2.git
/var/www/html/"
    ]
  }

}
```

This code launches a webserver and also mount EFS. After this, we clone the git repo to /var/www/html folder.

Now my EFS is mounted to my instance.

5. Create S3 bucket, and copy/deploy the images from GitHub repo into the S3 bucket and change the permission to public readable:

For creating S3...

```
#To create S3 bucket
resource "aws_s3_bucket" "my-terra-task-bucket" {
  bucket = "my-terra-task-bucket"
  acl    = "public-read"
  force_destroy  = true
  cors_rule {
    allowed_headers = ["*"]
    allowed_methods = ["PUT", "POST"]
    allowed_origins = ["https://my-terra-task-bucket"]
```

```
    expose_headers  = ["ETag"]
    max_age_seconds = 3000
  }
depends_on = [
   null_resource.null_vol_attach,
  ]
}
```

For uploading picture...

```
resource "aws_s3_bucket_object" "obj" {
  key = "ironman.jpg"
  bucket = aws_s3_bucket.my-terra-task-bucket.id
  source = "ironman.jpg"
  acl="public-read"

}
```

Now my image is uploaded to S3 and I am now linking it to CloudFront service to get a URL.

6. Create a CloudFront using s3 bucket (which contains images):

```
# Create Cloudfront distribution
resource "aws_cloudfront_distribution" "distribution_efs" {
    origin {
        domain_name = "${aws_s3_bucket.my-terra-task-
bucket.bucket_regional_domain_name}"
        origin_id = "S3-${aws_s3_bucket.my-terra-task-bucket.bucket}"




        custom_origin_config {
            http_port = 80
            https_port = 443
            origin_protocol_policy = "match-viewer"
            origin_ssl_protocols = ["TLSv1", "TLSv1.1", "TLSv1.2"]
        }
}
    # By default, show ironman.jpg file
    default_root_object = "ironman.jpg"
    enabled = true




    # If there is a 404, return ironman.jpg with a HTTP 200 Response
    custom_error_response {
        error_caching_min_ttl = 3000
        error_code = 404
```

```
        response_code = 200
        response_page_path = "/ironman.jpg"
    }



    default_cache_behavior {
        allowed_methods = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH",
"POST", "PUT"]
        cached_methods = ["GET", "HEAD"]
        target_origin_id = "S3-${aws_s3_bucket.my-terra-task-bucket.bucket}"



        #Not Forward all query strings, cookies and headers
        forwarded_values {
            query_string = false
            cookies {
                forward = "none"
            }

        }



        viewer_protocol_policy = "redirect-to-https"
        min_ttl = 0
        default_ttl = 3600
        max_ttl = 86400

 }

    # Restricts who is able to access this content
    restrictions {
        geo_restriction {
            # type of restriction, blacklist, whitelist or none
            restriction_type = "none"
        }
    }



    # SSL certificate for the service.
    viewer_certificate {
        cloudfront_default_certificate = true
    }
}
```

7. For update the code in /var/www/html folder:

```
resource "null_resource" "update_pic_url"  {
  depends_on = [
    null_resource.null_vol_attach,
    aws_cloudfront_distribution.distribution_efs,
  ]
  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key =  tls_private_key.key_create.private_key_pem
    host     = aws_instance.webapp.public_ip
  }
  provisioner "remote-exec" {
    inline = [
        "sudo chmod ugo+rw /var/www/html/index.php",
        "sudo echo '<img
src=http://${aws_cloudfront_distribution.distribution_efs.domain_name}/ironma
n.jpg alt='ANUDDEEPH NALLA' width='500' height='600'</a>' >>
/var/www/html/index.php"
    ]
  }

}
```

8.Output (print on screen):

```
output "cloudfront_ip_addr" {
  value = aws_cloudfront_distribution.distribution_efs.domain_name
}
output "myoutaz" {
               value = aws_instance.webapp.availability_zone
}
output "myoutip" {
               value = aws_instance.webapp.public_ip
}
 resource "null_resource" "nullloc"  {
 depends_on = [
 null_resource.null_vol_attach,aws_cloudfront_distribution.distribution_efs,a
ws_s3_bucket.my-terra-task-bucket,
 ]
       provisioner "local-exec" {
           command = "chrome  ${aws_instance.webapp.public_ip}"
       }
}
```

Now my code is completed...😀

You can also login (SSH) in your instance using the same key which we saved.

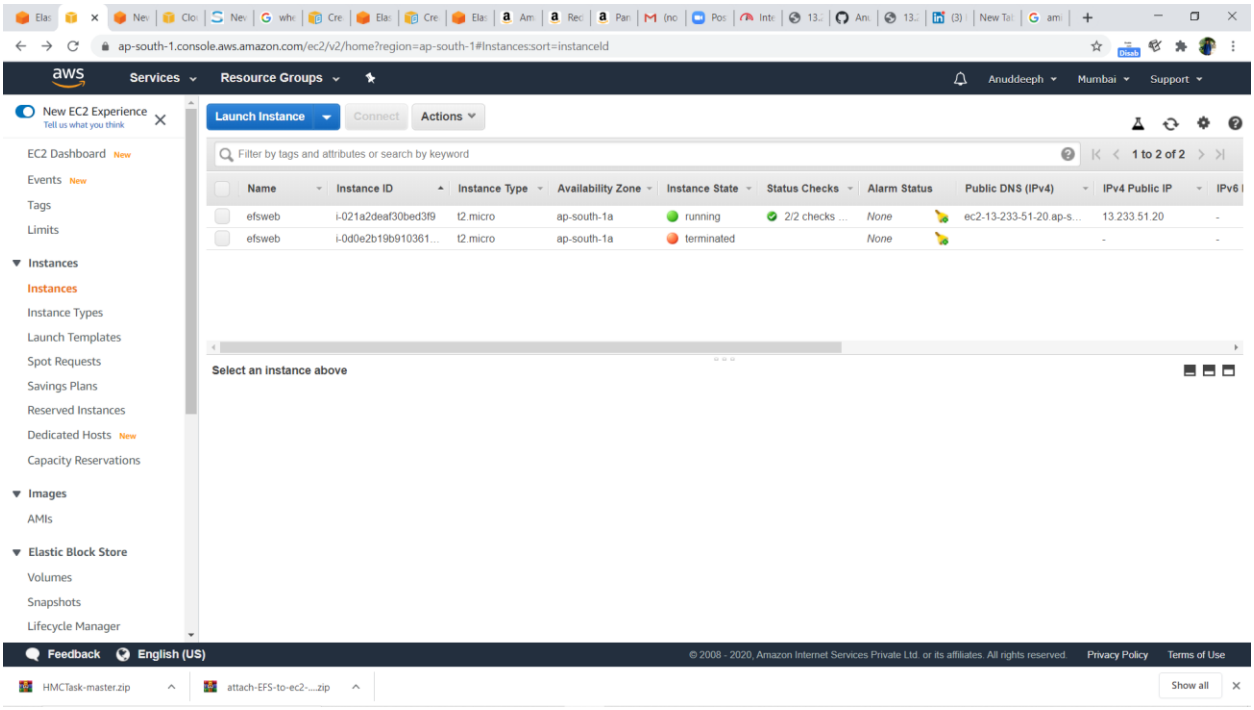
Save this code in a file having .tf extension.

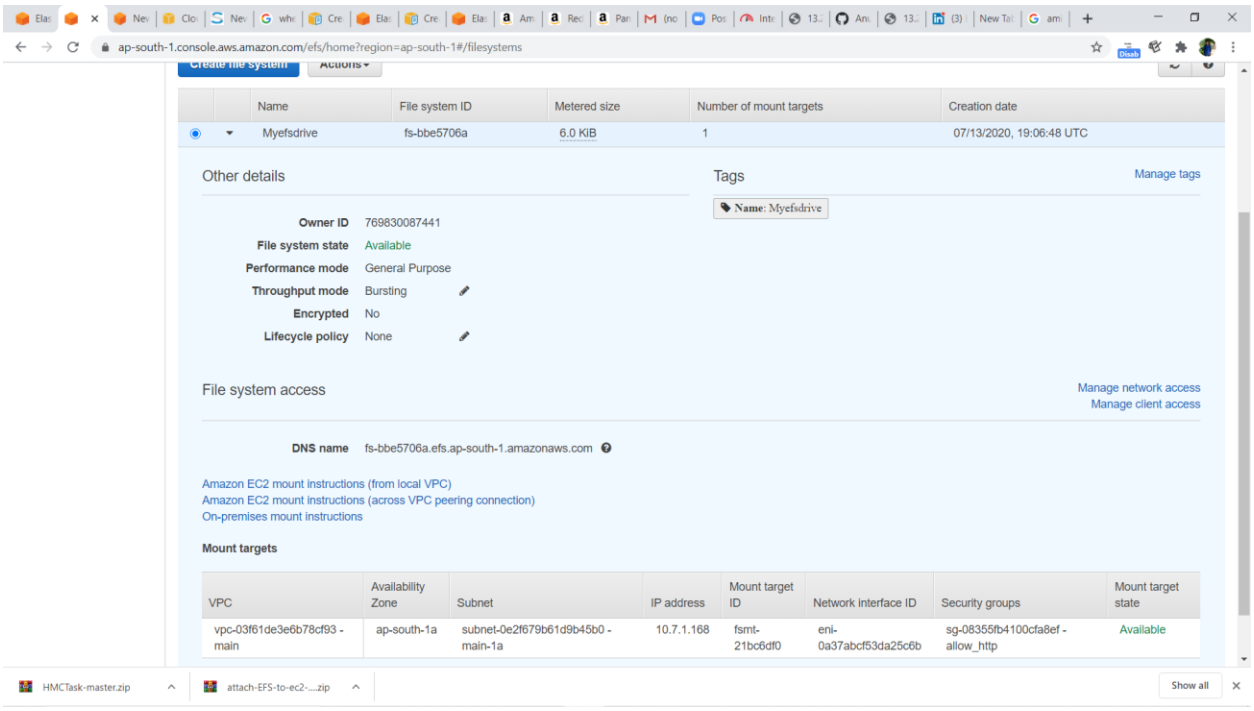
For run, command is...

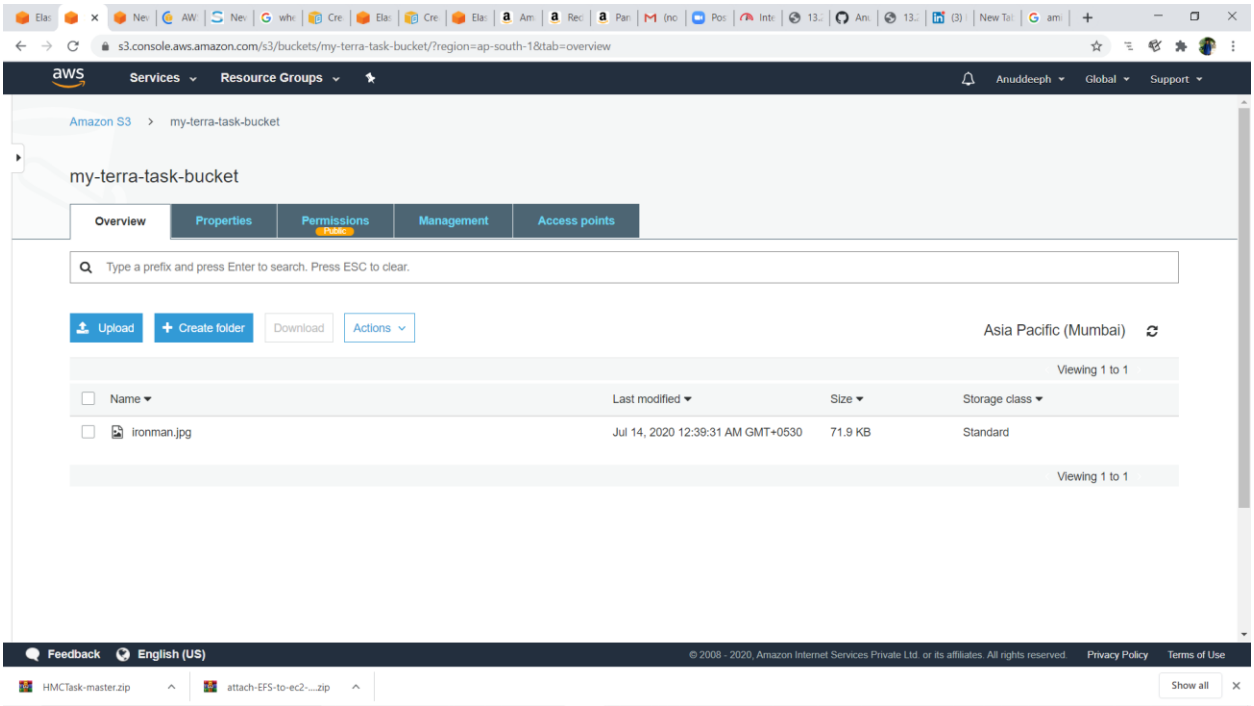
- **terraform init**
- **terraform apply**

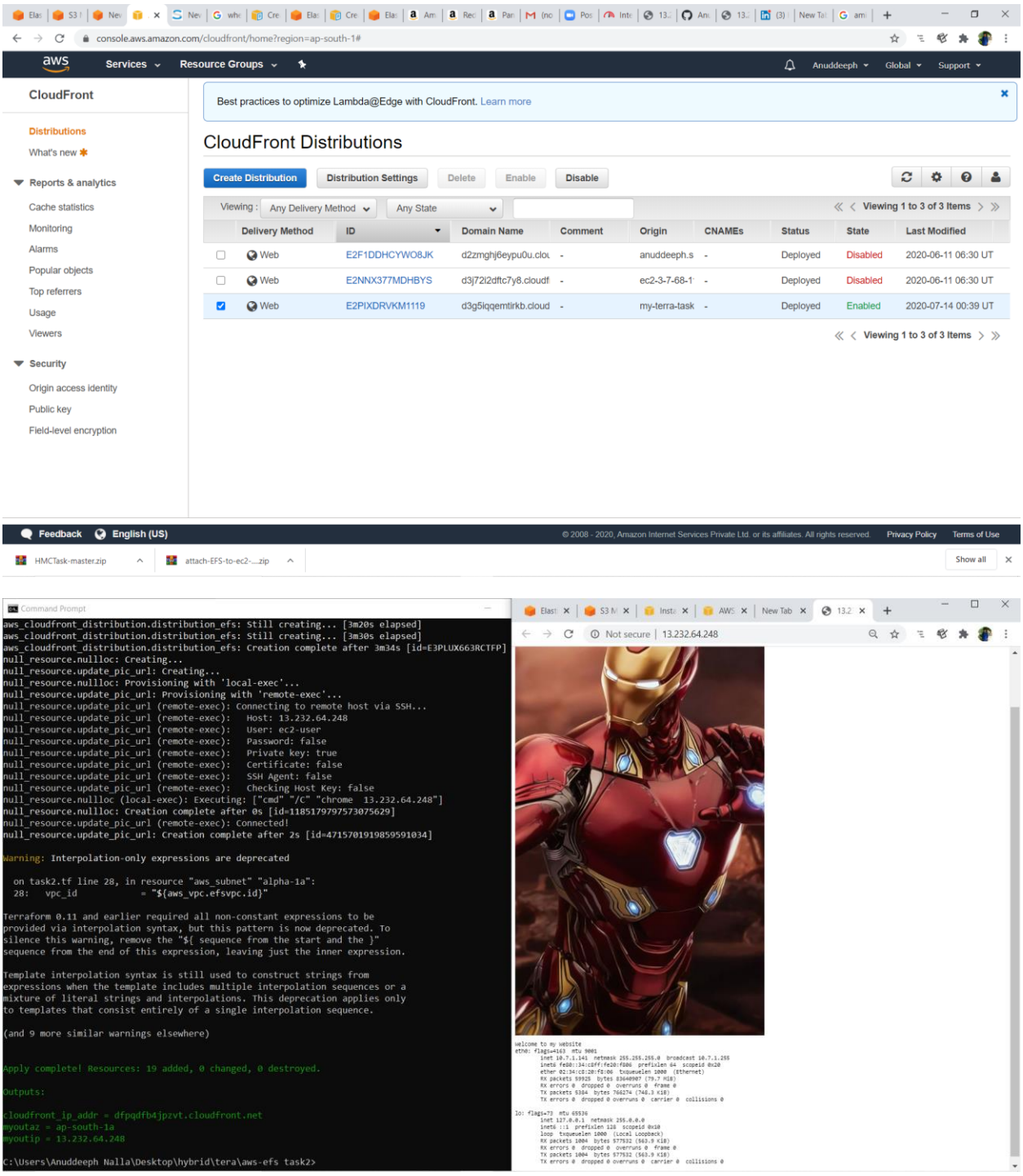
And the best part is, this setup is fully automated so we can destroy too the whole setup in one single command...

- **terraform destroy**
- **GitHub Link:** https://github.com/Anuddeeph/HMCTask2.git
- Now my infrastructure is created...




ap-south-1.console.aws.amazon.com/efs/home?region=ap-south-1#/filesystems
Create file system
Actions
Name
File system ID
Metered size
Number of mount targets
Creation date
Myefsdrive
fs-bbe5706a
6.0 KiB
1
07/13/2020, 19:06:48 UTC
Other details
Owner ID 769830087441
File system state Available
Performance mode General Purpose
Throughput mode Bursting
Encrypted No
Lifecycle policy None
Tags
Manage tags
Name: Myefsdrive
File system access
Manage network access
Manage client access
DNS name fs-bbe5706a.efs.ap-south-1.amazonaws.com
Amazon EC2 mount instructions (from local VPC)
Amazon EC2 mount instructions (across VPC peering connection)
On-premises mount instructions
Mount targets
VPC
Availability Zone
Subnet
IP address
Mount target ID
Network interface ID
Security groups
Mount target state
vpc-03f61de3e6b78cf93 - main
ap-south-1a
subnet-0e2f679b61d9b45b0 - main-1a
10.7.1.168
fsmt-21bc6df0
eni-0a37abcf53da25c6b
sg-08355fb4100cfa8ef - allow_http
Available
HMCTask-master.zip
attach-EFS-to-ec2-....zip
Show all


s3.console.aws.amazon.com/s3/buckets/my-terra-task-bucket/?region=ap-south-1&tab=overview
aws
Services
Resource Groups
Anuddeeph
Global
Support
Amazon S3 > my-terra-task-bucket
my-terra-task-bucket
Overview
Properties
Permissions
Public
Management
Access points
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
Asia Pacific (Mumbai)
Viewing 1 to 1
Name
Last modified
Size
Storage class
ironman.jpg
Jul 14, 2020 12:39:31 AM GMT+0530
71.9 KB
Standard
Viewing 1 to 1
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
HMCTask-master.zip
attach-EFS-to-ec2-....zip
Show all



Please give a thumbs up if you gain some knowledge...

Feel free to ask me if you have any query regarding the task.

Thanks for reading :)