TASK-1:
HYBRID MULTI CLOUD COMPUTING

Launching Web Application on AWS using Terraform, CloudFront, and Github with Complete End-to-End Automation

Launch Web Application with AWS EC2 + EBS + S3 + CloudFront using Terraform and Git

Report By: Mridul Markandey
Mentor: Mr. Vimal Daga Sir

1. PREREQUISITES:

1.1 AWS Account:

You need to have an account on AWS. If you don’t have, follow this link and create one: https://aws.amazon.com/

1.2 Terraform:

You should have Terraform downloaded and configured in your system. If you don’t have the software, download from here:
https://www.terraform.io/downloads.html

After downloading, just add the path to the environment variable to properly configure it. You can confirm using this command:

```
#terraform -version
```

1.3 Profile:

You must create one profile (IAM User) on AWS and because in the future we will use that profile to build our infrastructure on AWS using terraform.

1.4 Key-Pair:

You should already create one key pair in advance in the AWS EC2 console. This will be used to create key-pairs or can even attach this key to Web server that we will create. After creating just download that to some location in your PC.

2. INFRASTRUCTURE AS CODE (IAC)

Let me walk you through what infrastructure we are talking about. So as an overview our main goal is to launch a web application on AWS using Terraform and Github with complete end-to-end automation.

2.1 TASKS TO BE DONE (Only using Terraform, Nothing to be done manually:

2.1.1 Create one security group which allows ingress traffic through port 80 (HTTP) and port 22 (ssh).

2.1.2 Launch an EC2 instance using the pre-created Key-pair and Security Group which we created in step 1.

2.1.3 Through remote login, install services such as HTTPD, GIT, PHP.

2.1.4 Launch one Volume (EBS). Attach it to the instance and after creating a partition, format the volume. Next mount that volume into /var/www/html/

2.1.5 As soon as the developer updates something in the code in GitHub, clone this Github repo which has our website source code also has some static images(static content).Clone the GitHub repo to this location /var/www/html

2.1.6 Create an S3 bucket. All the static content of our web server is stored over here. Copy images from your system to S3 and change the permission to public readable.

2.1.7 Create a distribution through CloudFront and using S3 bucket (which contains static images) generate one CloudFront URL and dynamically update this URL in website code in the “index.html” file.

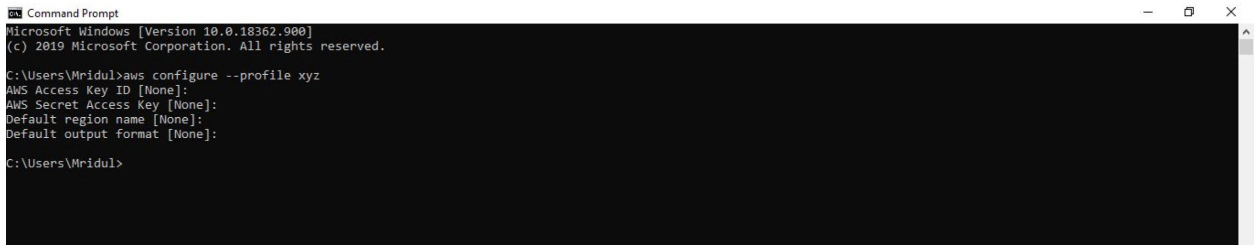
2.1.8 In the end, terraform will automatically launch our website having the latest content in the browser.

3. STEP BY STEP PROCEDURE:

3.1 CREATE PROFILE AND WORKSPACE:

First of all, you have to create a profile and create one workspace where we can initialize Terraform to build infrastructure in AWS cloud.

```
#aws configure --profile <profile-name>

//enter your details
```



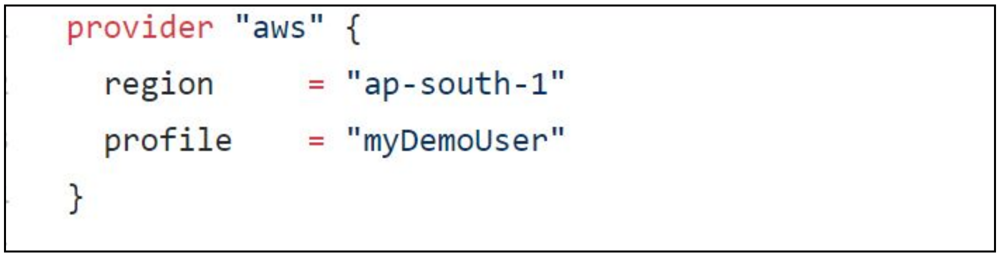
```
Microsoft Windows [Version 10.0.18362.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Mridul>aws configure --profile xyz
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:

C:\Users\Mridul>
```

Then create a workspace for terraform and create a fie in .tf format:

```
#mkdir /tera
# cd tera
#notepad  task1.tf
```

3.2 DECLARE PROVIDER:

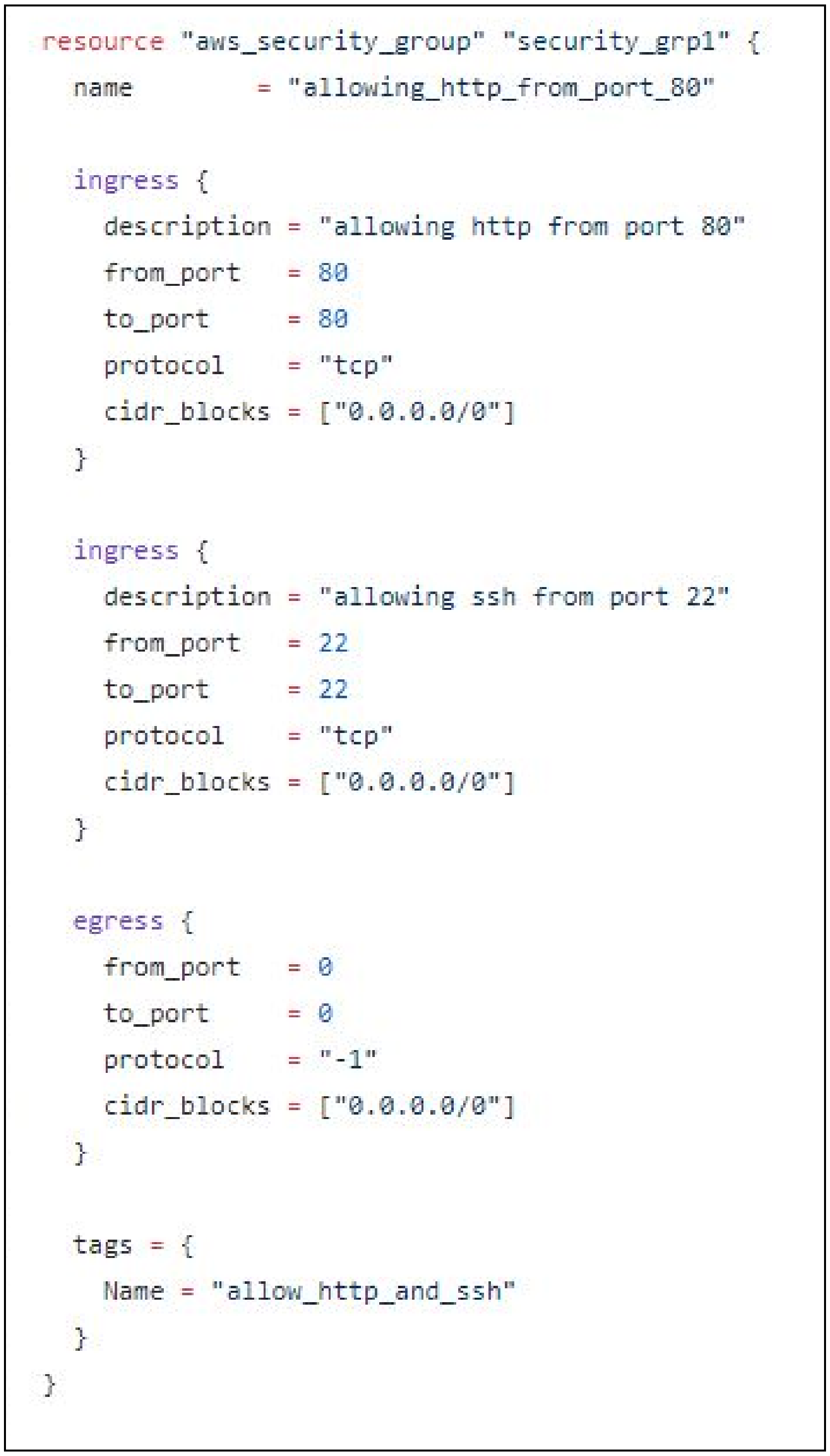
Now in “task1.tf” file add a provider.:



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

3.3 CREATE SECURITY GROUP:

Next create one security group which allows ingress traffic through port 80 (HTTP) and port 22 (ssh).

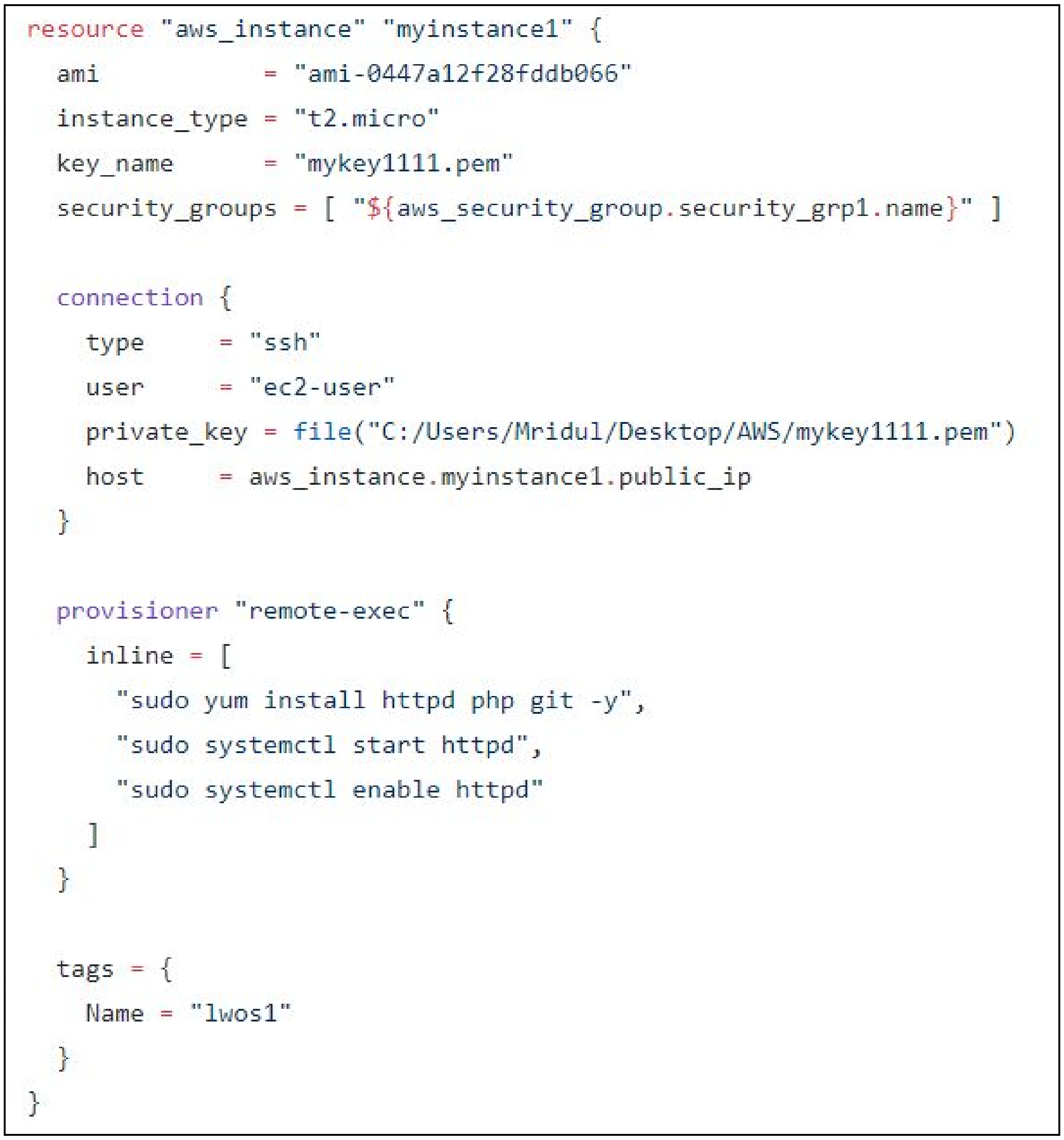


```
resource "aws_security_group" "security_grp1" {
  name        = "allowing_http_from_port_80"

  ingress {
    description = "allowing http from port 80"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "allowing ssh from port 22"
    from_port   = 22
    to_port     = 22
    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 = "allow_http_and_ssh"
  }
}
```

3.4 LAUNCH EC2 INSTANCE & INSTALL SERVICES:

Launch an EC2 instance using the pre-created Key-pair and Security Group which we created in the previous step.

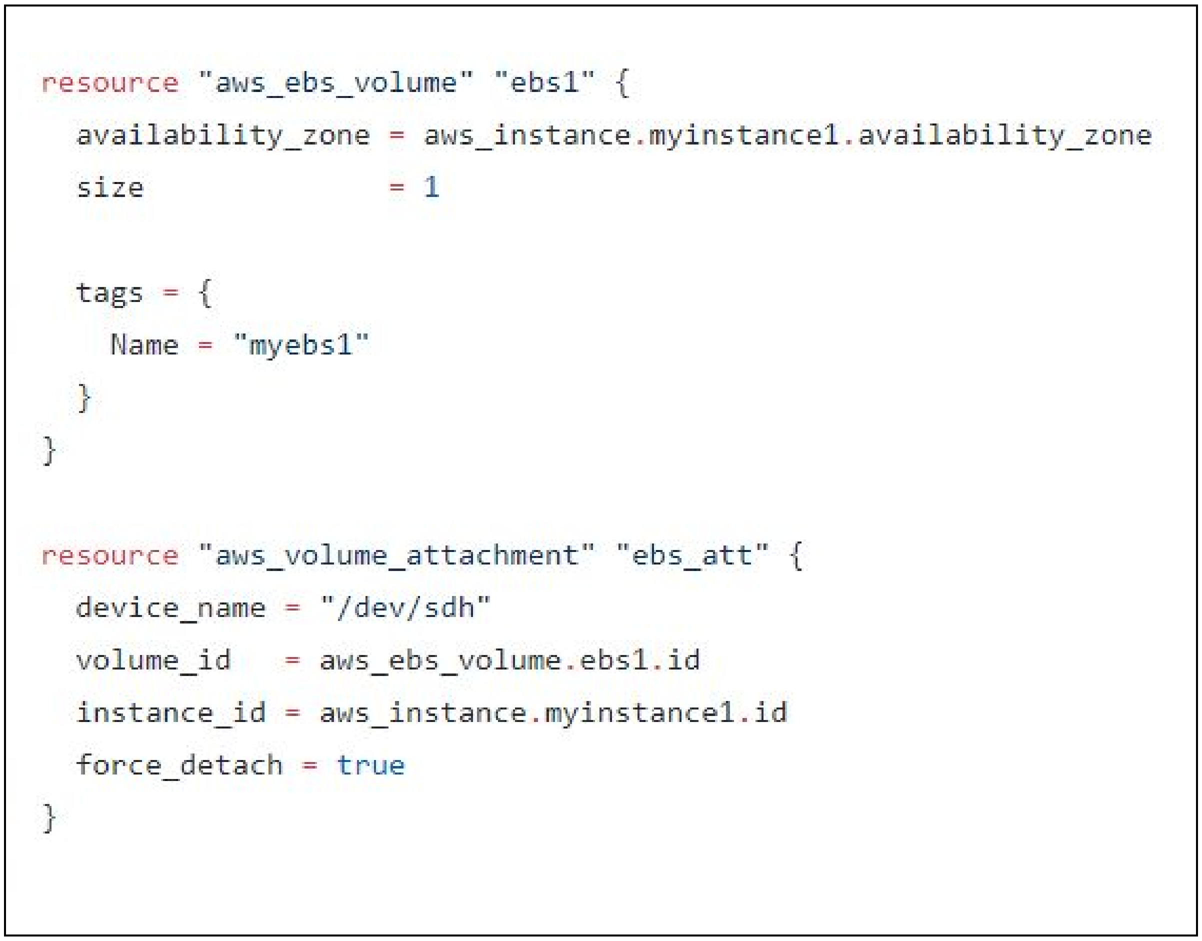
Through remote login, install services such as HTTPD, GIT, PHP.



```
resource "aws_instance" "myinstance1" {
  ami           = "ami-0447a12f28fddb066"
  instance_type = "t2.micro"
  key_name      = "mykey1111.pem"
  security_groups = [ "${aws_security_group.security_grp1.name}" ]

  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = file("C:/Users/Mridul/Desktop/AWS/mykey1111.pem")
    host     = aws_instance.myinstance1.public_ip
  }

  provisioner "remote-exec" {
    inline = [
      "sudo yum install httpd php git -y",
      "sudo systemctl start httpd",
      "sudo systemctl enable httpd"
    ]
  }

  tags = {
    Name = "lwos1"
  }
}
```

3.5 CREATE EBS VOLUME, ATTACH IT TO INSTANCE *:*

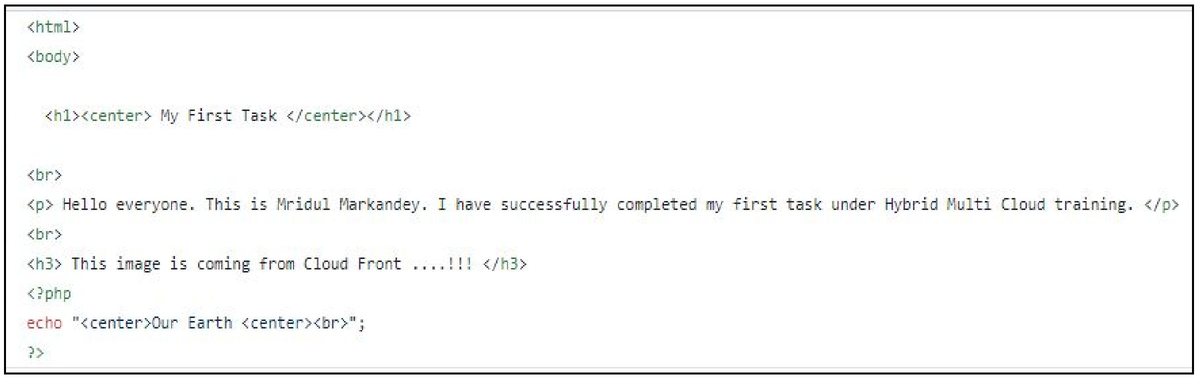
Launch one Volume (EBS). Attach it to the instance and after creating a partition, format the volume. Next mount that volume into /var/www/html/



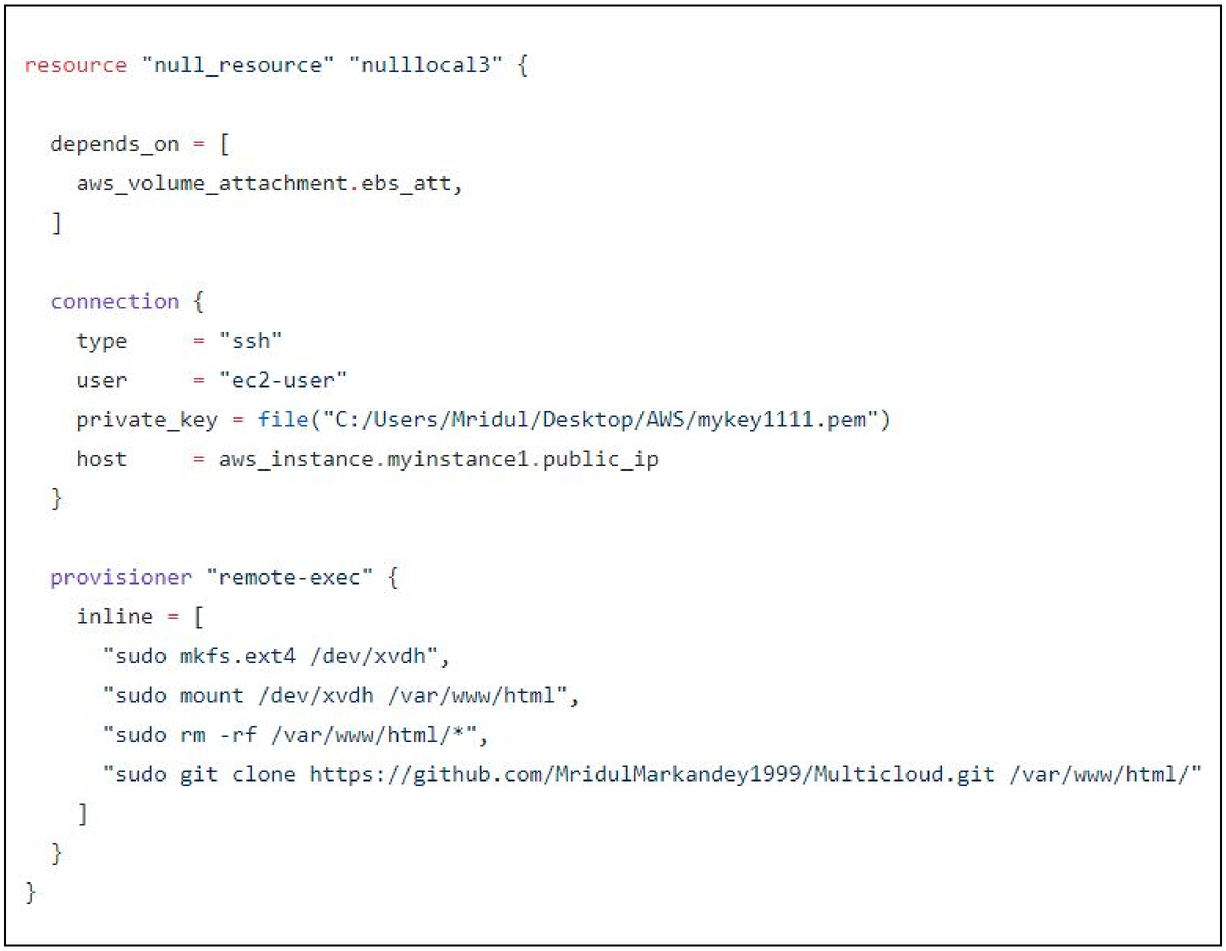
```
resource "aws_ebs_volume" "ebs1" {
  availability_zone = aws_instance.myinstance1.availability_zone
  size              = 1

  tags = {
    Name = "myebs1"
  }
}

resource "aws_volume_attachment" "ebs_att" {
  device_name = "/dev/sdh"
  volume_id   = aws_ebs_volume.ebs1.id
  instance_id = aws_instance.myinstance1.id
  force_detach = true
}
```

3.6 MOUNT PARTITION TO */var/www/html/* & DOWNLOAD SOURCE CODE FROM GITHUB:

As soon as the developer updates something in the code in GitHub, clone this Github repo which has our website source code also has some static images(static content).Clone the GitHub repo to this location /var/www/html



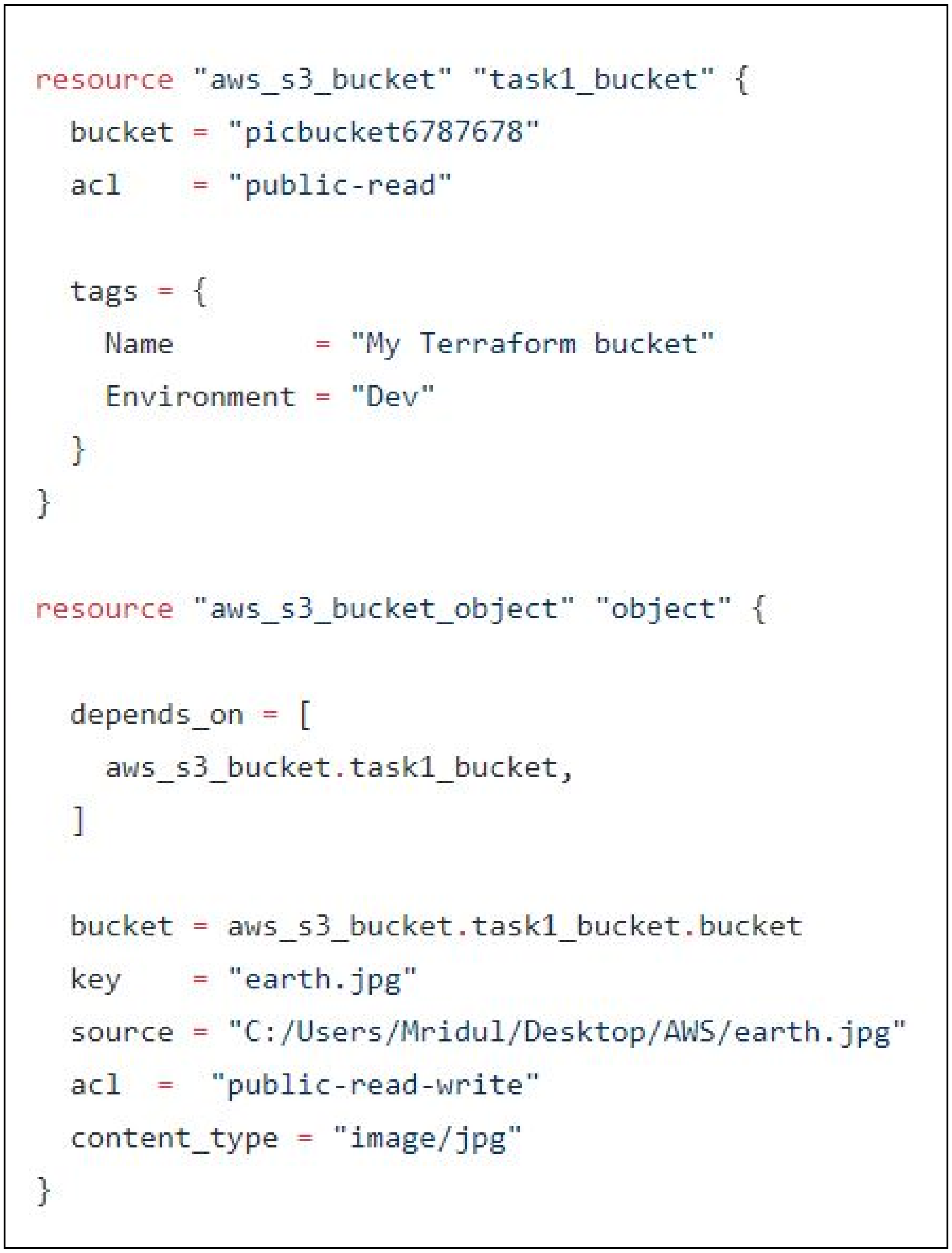
```
<html>
<body>

  <h1><center> My First Task </center></h1>

<br>
<p> Hello everyone. This is Mridul Markandey. I have successfully completed my first task under Hybrid Multi Cloud training. </p>
<br>
<h3> This image is coming from Cloud Front ....!!! </h3>
<?php
echo "<center>Our Earth <center><br>";
?>
```



```
resource "null_resource" "nulllocal3" {

  depends_on = [
    aws_volume_attachment.ebs_att,
  ]

  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = file("C:/Users/Mridul/Desktop/AWS/mykey1111.pem")
    host     = aws_instance.myinstance1.public_ip
  }

  provisioner "remote-exec" {
    inline = [
      "sudo mkfs.ext4 /dev/xvdh",
      "sudo mount /dev/xvdh /var/www/html",
      "sudo rm -rf /var/www/html/*",
      "sudo git clone https://github.com/MridulMarkandey1999/Multicloud.git /var/www/html/"
    ]
  }
}
```

3.7 CREATE S3 BUCKET AND COPY STATIC CONTENT TO S3:

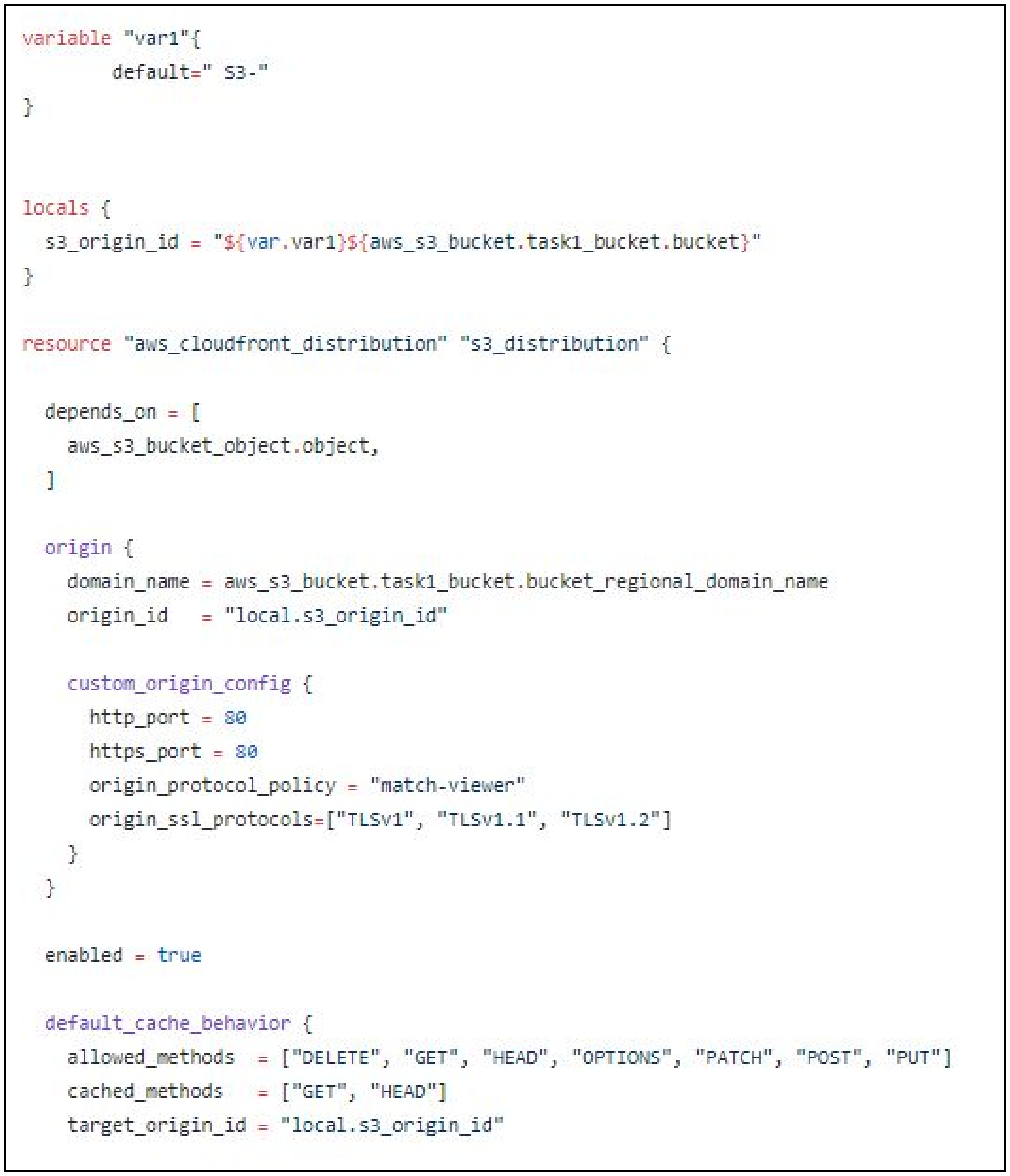
Create an S3 bucket. All the static content of our web server is stored over here. Copy images from your system to S3 and change the permission to public readable.



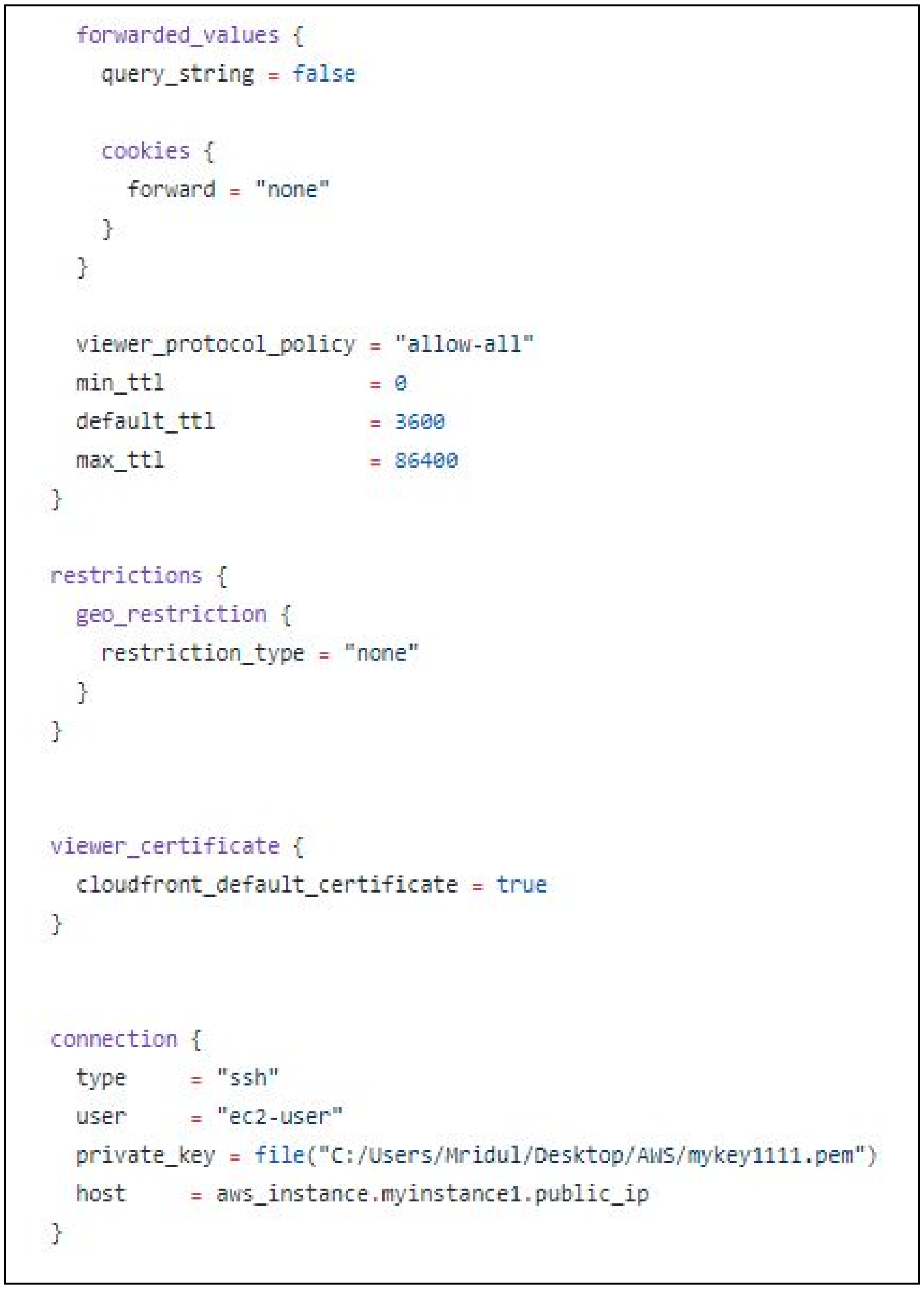
```
resource "aws_s3_bucket" "task1_bucket" {
  bucket = "picbucket6787678"
  acl    = "public-read"

  tags = {
    Name        = "My Terraform bucket"
    Environment = "Dev"
  }
}

resource "aws_s3_bucket_object" "object" {

  depends_on = [
    aws_s3_bucket.task1_bucket,
  ]

  bucket = aws_s3_bucket.task1_bucket.bucket
  key    = "earth.jpg"
  source = "C:/Users/Mridul/Desktop/AWS/earth.jpg"
  acl  =  "public-read-write"
  content_type = "image/jpg"
}
```

3.8 CREATING DISTRIBUTION USING CLOUDFRONT AND DYNAMICALLY UPDATE IMAGE URL:

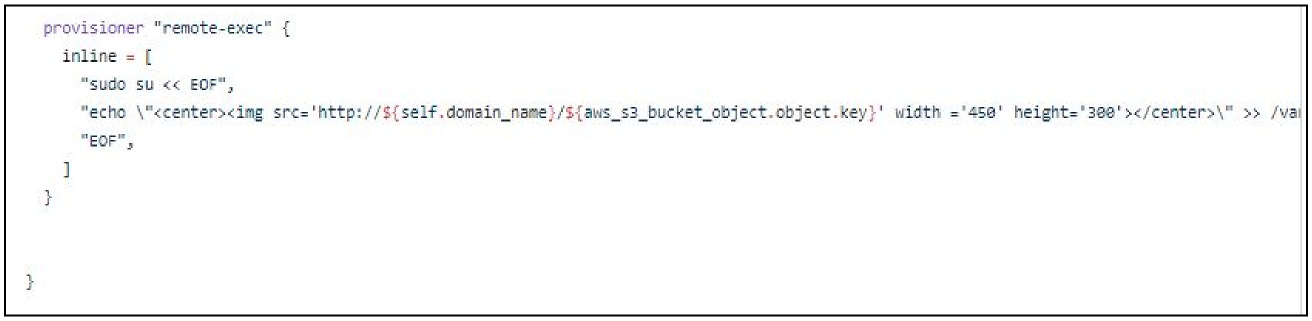
Create a distribution through CloudFront and using the S3 bucket (which contains static images) generate one CloudFront URL and dynamically update this URL in website code in the “index.html” file.



```
variable "var1"{
        default=" S3-"
}



locals {
  s3_origin_id = "${var.var1}${aws_s3_bucket.task1_bucket.bucket}"
}

resource "aws_cloudfront_distribution" "s3_distribution" {

  depends_on = [
    aws_s3_bucket_object.object,
  ]

  origin {
    domain_name = aws_s3_bucket.task1_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"
```



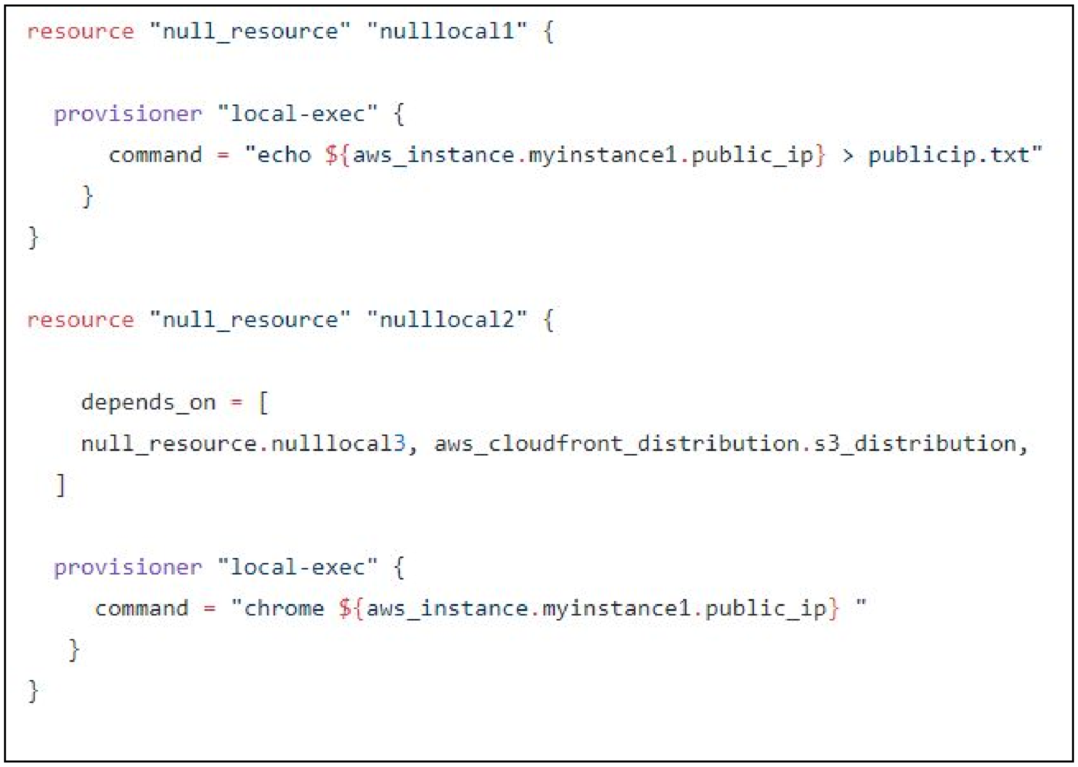
```
  forwarded_values {
    query_string = false

    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 = true
}


connection {
  type        = "ssh"
  user        = "ec2-user"
  private_key = file("C:/Users/Mridul/Desktop/AWS/mykey1111.pem")
  host        = aws_instance.myinstance1.public_ip
}
```



```
  provisioner "remote-exec" {
    inline = [
      "sudo su << EOF",
      "echo \"<center><img src='http://${self.domain_name}/${aws_s3_bucket_object.object.key}' width ='450' height='300'></center>\" >> /va
      "EOF",
    ]
  }


}
```

3.9 LAUNCHING WEB APP AS SOON AS IT IS READY:

In the end, terraform will automatically launch our website having the latest content in the browser. Also here we store our public IP into a file for future reference.



```
resource "null_resource" "nulllocal1" {


  provisioner "local-exec" {
      command = "echo ${aws_instance.myinstance1.public_ip} > publicip.txt"
    }
}

resource "null_resource" "nulllocal2" {


    depends_on = [
    null_resource.nulllocal3, aws_cloudfront_distribution.s3_distribution,
  ]

  provisioner "local-exec" {
     command = "chrome ${aws_instance.myinstance1.public_ip} "
   }
}
```

4. APPLY AND SEE THE MAGIC..!!

After writing this code, save the file.
Open cmd and type these commands:

```
#terraform init

#terraform validate

#terraform apply -auto-approve
```

And you will see the magic of terraform. You will come to know how powerful this tool is. After some time teraform will automatically launch your web server whose static content is coming from S3 bucket through Cloud Front distribution. Complete end to end automation is done and your web server will launch with the most up to date content fetched from Github.

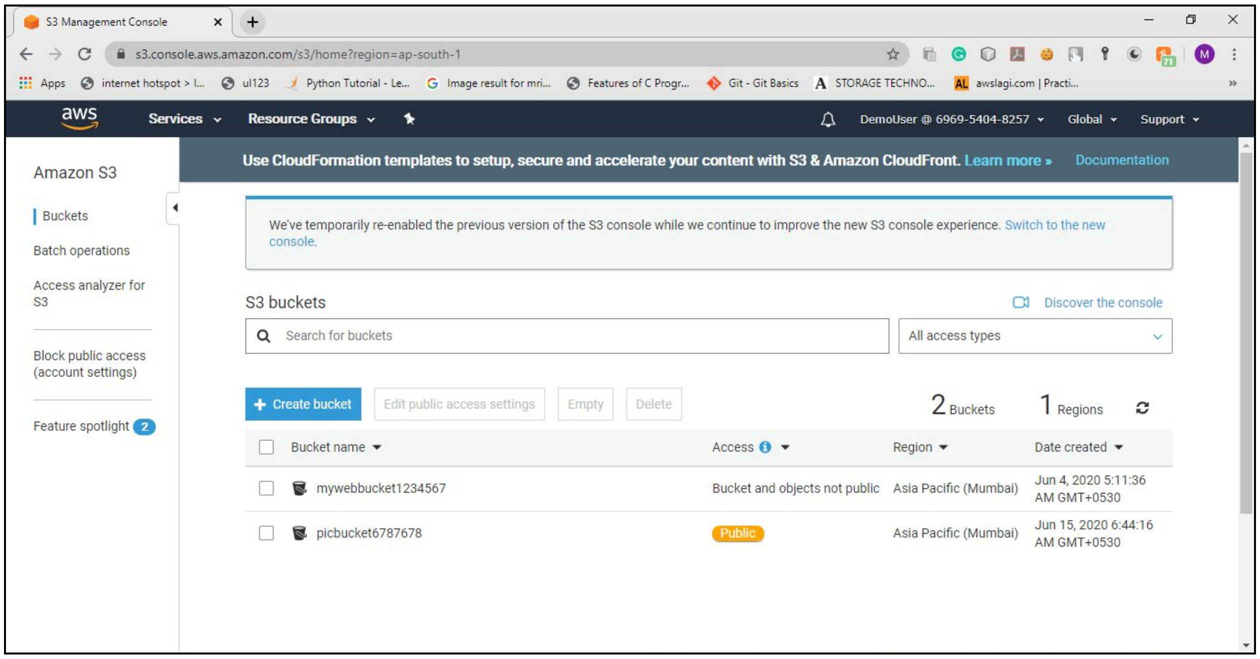
In the end, if you want to destroy the complete infrastructure you created, you can do this by just typing one command:

```
#terraform destroy -auto-approve
```

So by this, you must have got some idea about how powerful tool is Terraform. Although in real world, there is another powerful tool that is used for configuration management, i.e Ansible. In real-world, Ansible and Terraform work hand in hand and both have different use case which they solve.

5. OUTPUT SCREENS:

After the complete infrastructure is deployed we see such a setup done in AWS cloud.



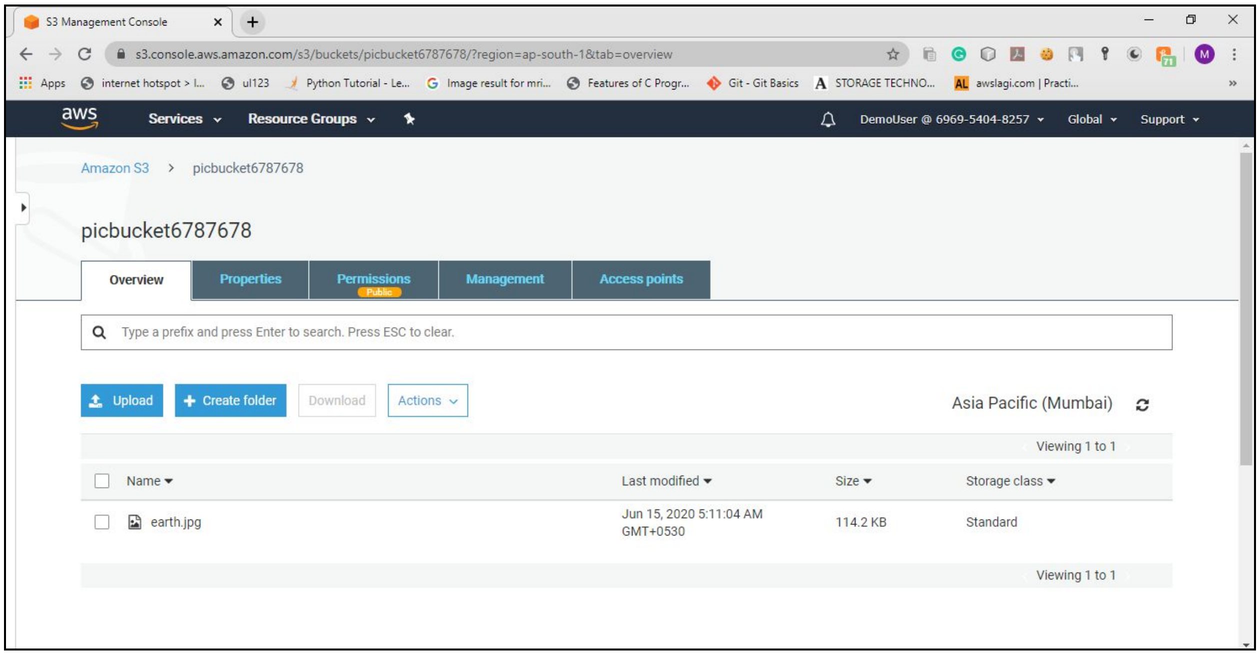
Use CloudFormation templates to setup, secure and accelerate your content with S3 & Amazon CloudFront. Learn more » Documentation

We've temporarily re-enabled the previous version of the S3 console while we continue to improve the new S3 console experience. Switch to the new console.

S3 buckets

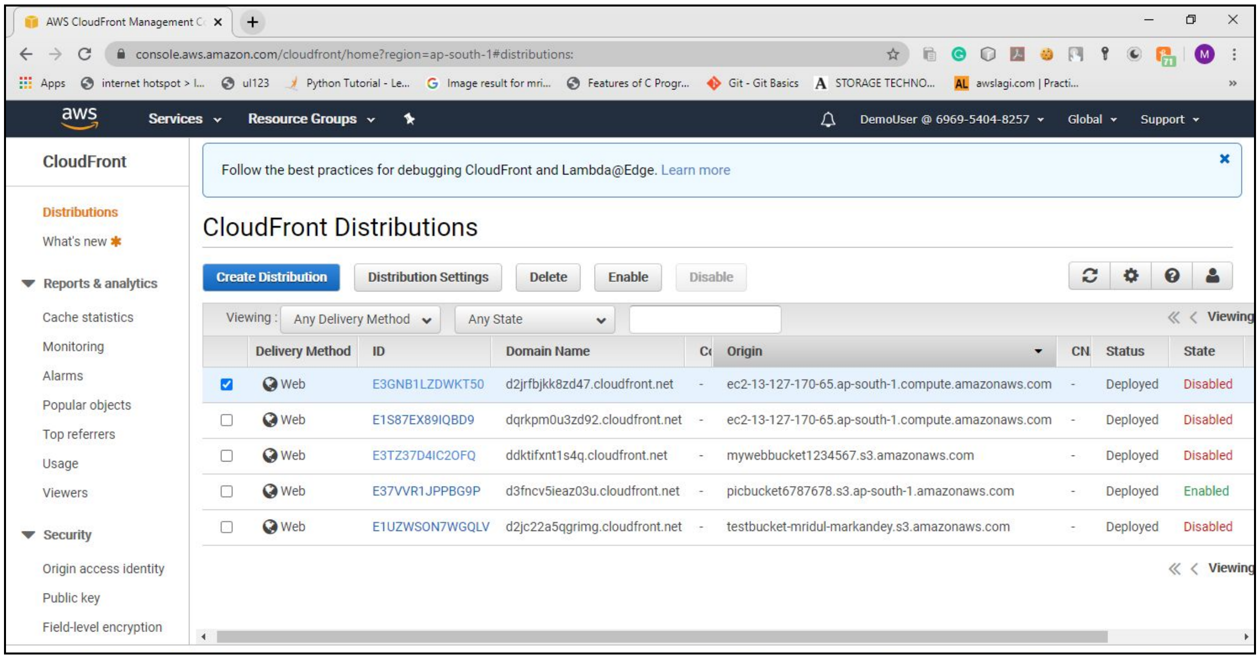
Bucket name	Access	Region	Date created
mywebbucket1234567	Bucket and objects not public	Asia Pacific (Mumbai)	Jun 4, 2020 5:11:36 AM GMT+0530
picbucket6787678	Public	Asia Pacific (Mumbai)	Jun 15, 2020 6:44:16 AM GMT+0530

New Bucket created in S3



Name	Last modified	Size	Storage class
earth.jpg	Jun 15, 2020 5:11:04 AM GMT+0530	114.2 KB	Standard

Image uploaded by Terraform in Bucket and this image is used by CloudFront

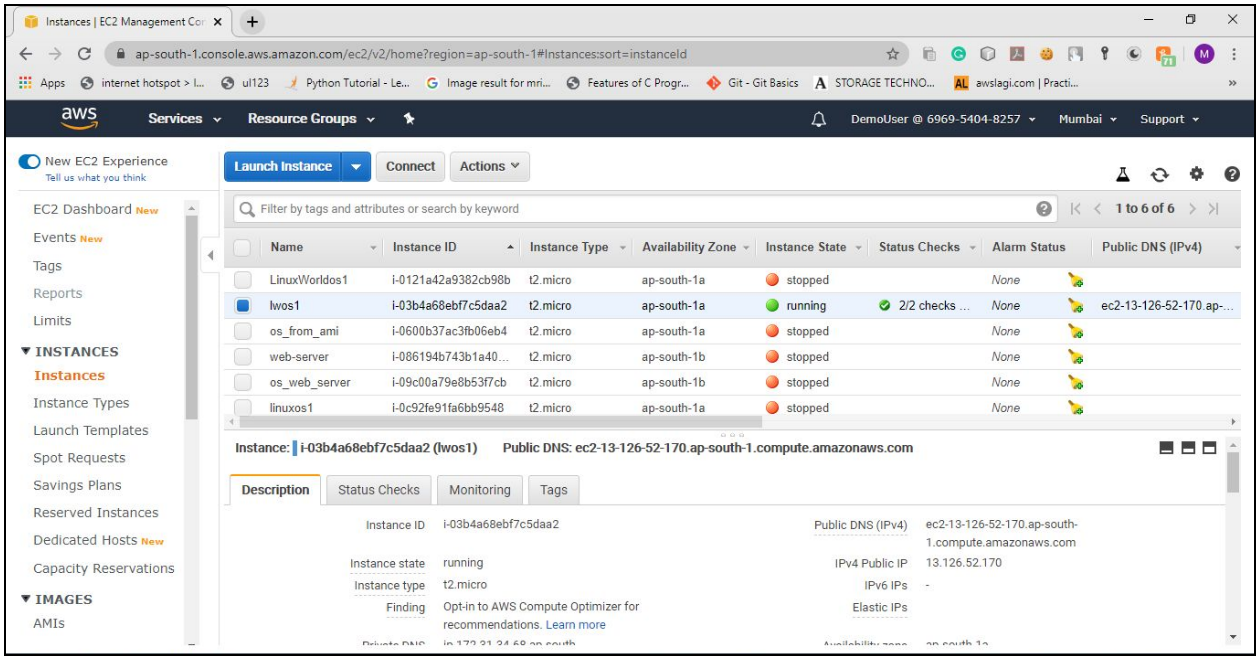


Follow the best practices for debugging CloudFront and Lambda@Edge. Learn more

CloudFront Distributions

	Delivery Method	ID	Domain Name	Cc	Origin	CN	Status	State
☑	Web	E3GNB1LZDWKT50	d2jrfbjkk8zd47.cloudfront.net	-	ec2-13-127-170-65.ap-south-1.compute.amazonaws.com	-	Deployed	Disabled
☐	Web	E1S87EX89IQBD9	dqrkpm0u3zd92.cloudfront.net	-	ec2-13-127-170-65.ap-south-1.compute.amazonaws.com	-	Deployed	Disabled
☐	Web	E3TZ37D4IC2OFQ	ddktifxnt1s4q.cloudfront.net	-	mywebbucket1234567.s3.amazonaws.com	-	Deployed	Disabled
☐	Web	E37VVR1JPPBG9P	d3fncv5ieaz03u.cloudfront.net	-	picbucket6787678.s3.ap-south-1.amazonaws.com	-	Deployed	Enabled
☐	Web	E1UZWSON7WGQLV	d2jc22a5qgrimg.cloudfront.net	-	testbucket-mridul-markandey.s3.amazonaws.com	-	Deployed	Disabled

New Distribution created by Terraform which serve static content to the web app

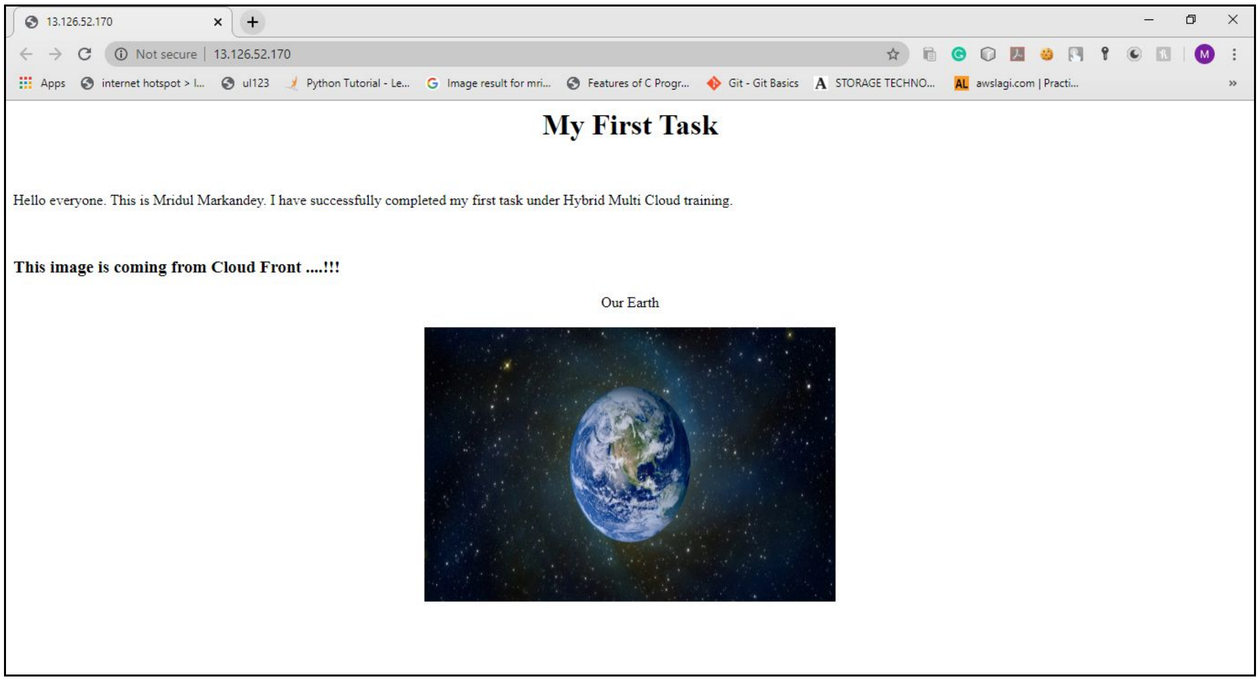


	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
☐	LinuxWorldos1	i-0121a42a9382cb98b	t2.micro	ap-south-1a	stopped		None	
☑	lwos1	i-03b4a68ebf7c5daa2	t2.micro	ap-south-1a	running	2/2 checks ...	None	ec2-13-126-52-170.ap-...
☐	os_from_ami	i-0600b37ac3fb06eb4	t2.micro	ap-south-1a	stopped		None	
☐	web-server	i-086194b743b1a40...	t2.micro	ap-south-1b	stopped		None	
☐	os_web_server	i-09c00a79e8b53f7cb	t2.micro	ap-south-1b	stopped		None	
☐	linuxos1	i-0c92fe91fa6bb9548	t2.micro	ap-south-1a	stopped		None	

Instance: i-03b4a68ebf7c5daa2 (lwos1) Public DNS: ec2-13-126-52-170.ap-south-1.compute.amazonaws.com

Instance ID	i-03b4a68ebf7c5daa2	Public DNS (IPv4)	ec2-13-126-52-170.ap-south-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	13.126.52.170
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	

New instance Launched by Terraform which is our web server



My First Task

Hello everyone. This is Mridul Markandey. I have successfully completed my first task under Hybrid Multi Cloud training.

This image is coming from Cloud Front ....!!!

Our Earth

And finally, our web app looks like this…!!

I hope you find this information useful. If you feel I have missed something, by any chance, I am open to your suggestions. Kindly contact me at:

- **Email:** **mridul.mark@gmail.com**
- **LinkedIn:** **https://www.linkedin.com/in/mridul-markandey/**
- **Source Code Link:**

https://github.com/MridulMarkandey1999/Multicloud/blob/master/task1.tf