Hybrid Multi Cloud Task1
Launch Web-Server by Single Command using AWS and Terraform

Task 1:

Have to create/launch Application using Terraform

1. Create the key and security group which allow the port 80.

2. Launch EC2 instance.

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

4. Launch one Volume (EBS) and mount that volume into /var/www/html

5. Developer have uploaded 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

Project description:

Prerequisite:

- Account on aws, if you do not have aws account go to the URL and create aws account. https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/
- Terraform download from https://www.terraform.io/downloads.html
- aws cli-v2 download from https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html

- use aws configure (setup the profile for aws from CMD using command aws configure --profile profilename)

Step-1: Create Security Group and Key

<u>Code:</u>

#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"

}
```

#create security_groups

```
resource "aws_security_group" "allow_http" {

  name        = "allow_http"

  description = "Allow TLS inbound traffic"

  vpc_id      = "vpc-09bfa361"

  ingress {
```

```
  description = "SSH"
  from_port   = 22
  to_port     = 22
  protocol    = "tcp"
  cidr_blocks = [ "0.0.0.0/0" ]
}
ingress {
  description = "HTTP"
  from_port   = 80
  to_port     = 80
  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 = "tasksg" } }
```

Explanation:

- First, we need to add the resource i.e., aws_security_group which allow to create a new security group.
- Port number 80 for http(website), 22 for ssh connection.
- Let's create key , we need to add resource ,i.e., tls_private_key which allow to generating the key for ssh in PEM format , for saving a file we need to add the resource, i.e., local_file and finally we saved the key locally.

Step-2,3: Launch EC2 instance, use the key and security group which we have created in step1

<u>Code:</u>

#Launch EC2 Instance

```
variable "enter_ur_key_name" {
                    type = string
          default = "taskkey"
}


#create instance
resource "aws_instance" "webapp" {
  ami           = "ami-0447a12f28fddb066"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  security_groups = [ "allow_http" ]
   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 yum update -y",
      "sudo yum install -y httpd git php",
      "sudo systemctl start httpd",
      "sudo systemctl enable httpd",
    ]
  }
tags = {
    Name = "webserver"
  }
}
```

By above code, my instance is launched and install some necessary programs to run my webserver. For this, I make an ssh-connection with my ec2 instance using provisioner resource.

Step-4,5,6: Launch one Volume (EBS) and mount that volume into /var/www/html:

```
#create EBS
resource "aws_ebs_volume" "ebs" {
  availability_zone = aws_instance.webapp.availability_zone
  size              = 1
  tags = {
    Name = "task_ebs"
  }}
```

```
#create attachment

resource "aws_volume_attachment" "ebs-attach" {

  device_name = "/dev/sdh"

  volume_id    = "${aws_ebs_volume.ebs.id}"

  instance_id  = "${aws_instance.webapp.id}"

  force_detach = true

}
```

This code makes an EBS volume and attach it to the instance. I use force_detach here because after mount when you try to destroy the environment, it failed because your volume is mounted.

For mount this to /var/www/html, see the below code:

```
#mount

resource "null_resource" "null_vol_attach"  {

  depends_on = [

    aws_volume_attachment.ebs-attach,

  ]

 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 mkfs.ext4  /dev/xvdh",

      "sudo mount  /dev/xvdh  /var/www/html",
      "sudo rm -rf /var/www/html/*",
      "sudo git clone https://github.com/Anuddeeph/HMCTask.git /var/www/html/"
    ]
  }
}
```

Now my EBS is mounted to the webserver instance. I also copied my GitHub code in /var/www/html where too which is uploaded by the developer.

Now come to next step...

Step-7: 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 = [
   aws_volume_attachment.ebs-attach,
  ]
}
```

<u>Now I need to upload my image in bucket:</u>

```
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.

```
# Create Cloudfront distribution
resource "aws_cloudfront_distribution" "distribution_s3" {
    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 can 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
    }
}
output "cloudfront_ip_addr" {
  value = aws_cloudfront_distribution.distribution_s3.domain_name
}
```

Finally, code is completed…

Complete Final Code:

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


#create key
resource "tls_private_key" "key_create"  {
  algorithm = "RSA"
```

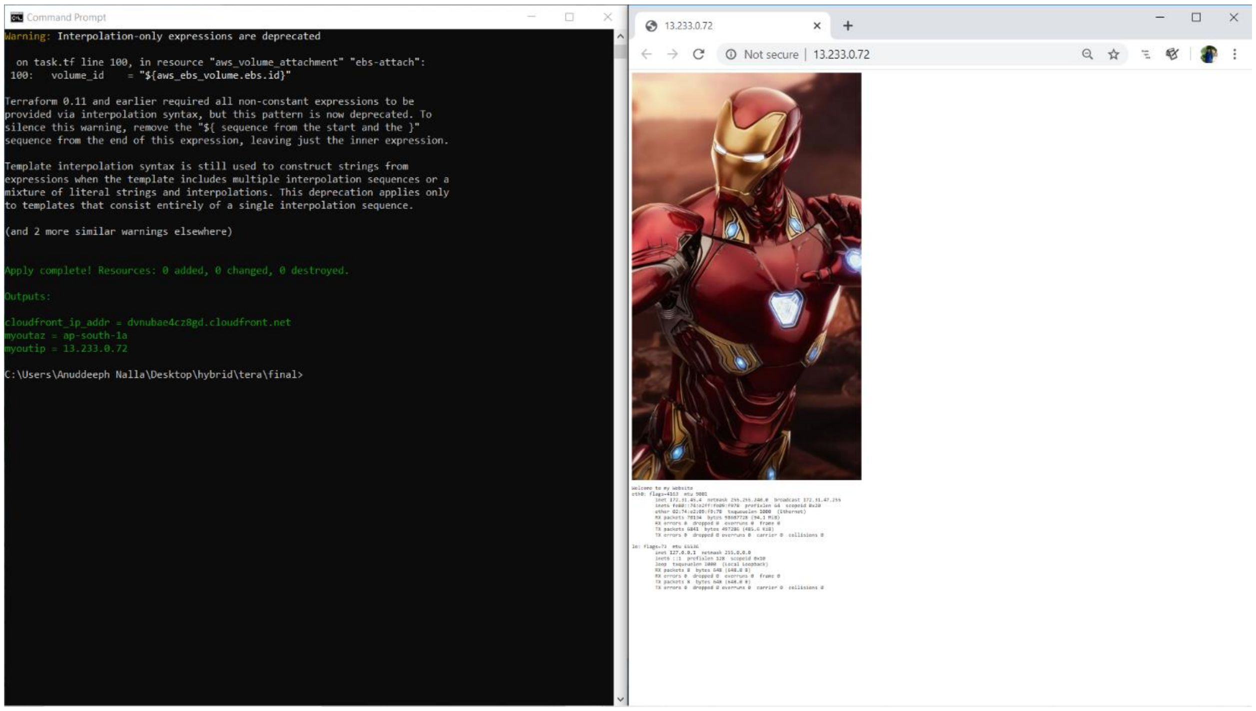
```
}
resource "aws_key_pair" "taskkey" {
  key_name    = "taskkey"
  public_key = tls_private_key.key_create.public_key_openssh
  }
resource "local_file" "save_key" {
    content     = tls_private_key.key_create.private_key_pem
    filename = "taskkey.pem"
}

#create security_groups
resource "aws_security_group" "allow_http" {
  name        = "allow_http"
  description = "Allow TLS inbound traffic"
  vpc_id      = "vpc-09bfa361"
  ingress {
    description = "SSH"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [ "0.0.0.0/0" ]
```

```
}



ingress {
  description = "HTTP"
  from_port   = 80
  to_port     = 80
  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 = "tasksg"
  }

}
variable "enter_ur_key_name" {
                    type = string
          default = "taskkey"
}

#create instance
resource "aws_instance" "webapp" {
  ami           = "ami-0447a12f28fddb066"
  instance_type = "t2.micro"
  key_name      = var.enter_ur_key_name
  security_groups = [ "allow_http" ]

   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 yum update -y",
      "sudo yum install -y httpd git php",
      "sudo systemctl start httpd",
      "sudo systemctl enable httpd",
    ]
  }
tags = {
    Name = "webserver"
  }
}

#create EBS
resource "aws_ebs_volume" "ebs" {
  availability_zone = aws_instance.webapp.availability_zone
  size              = 1
  tags = {
    Name = "task_ebs"
  }
}

#create attachment
resource "aws_volume_attachment" "ebs-attach" {
  device_name = "/dev/sdh"
  volume_id    = "${aws_ebs_volume.ebs.id}"
  instance_id  = "${aws_instance.webapp.id}"
  force_detach = true
}
output "myoutaz" {
                    value = aws_instance.webapp.availability_zone
}
output "myoutip" {
                    value = aws_instance.webapp.public_ip
}

resource "null_resource" "save_ip" {
  provisioner "local-exec" {
    command = "echo ${aws_instance.webapp.public_ip} >> public_ip.txt"
  }
}

#mount
resource "null_resource" "null_vol_attach"  {
  depends_on = [
    aws_volume_attachment.ebs-attach,
  ]
 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 mkfs.ext4  /dev/xvdh",
      "sudo mount  /dev/xvdh  /var/www/html",
      "sudo rm -rf /var/www/html/*",
      "sudo git clone https://github.com/Anuddeeph/HMCTask.git
/var/www/html/"

    ]

  }

}

resource "null_resource" "null_vol_depend"  {




depends_on = [

    null_resource.null_vol_attach,

  ]

}


#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 = [

   aws_volume_attachment.ebs-attach,

  ]

}

resource "aws_s3_bucket_object" "obj" {

  key = "ironman.jpg"

  bucket = aws_s3_bucket.my-terra-task-bucket.id

  source = "ironman.jpg"

  acl="public-read"

}


# Create Cloudfront distribution

resource "aws_cloudfront_distribution" "distribution_s3" {

    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
    }
}
output "cloudfront_ip_addr" {
  value = aws_cloudfront_distribution.distribution_s3.domain_name
}
 resource "null_resource" "nullloc"  {
```

```
 depends_on = [

null_resource.null_vol_attach,aws_cloudfront_distribution.distribution_s3,aws_s3_bucket.my-terra-task-bucket

 ]

        provisioner "local-exec" {

            command = "chrome  ${aws_instance.webapp.public_ip}"

        }

}
```



```
Warning: Interpolation-only expressions are deprecated

  on task.tf line 100, in resource "aws_volume_attachment" "ebs-attach":
 100:   volume_id    = "${aws_ebs_volume.ebs.id}"

Terraform 0.11 and earlier required all non-constant expressions to be
provided via interpolation syntax, but this pattern is now deprecated. To
silence this warning, remove the "${ sequence from the start and the }"
sequence from the end of this expression, leaving just the inner expression.

Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixture of literal strings and interpolations. This deprecation applies only
to templates that consist entirely of a single interpolation sequence.

(and 2 more similar warnings elsewhere)


Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

cloudfront_ip_addr = dvnubae4cz8gd.cloudfront.net
myoutaz = ap-south-1a
myoutip = 13.233.0.72

C:\Users\Anuddeeph Nalla\Desktop\hybrid\tera\final>
```

```
Welcome to my Website
eth0: flags=4163  mtu 9001
        inet 172.31.45.4  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::74:c2ff:fe09:f97a  prefixlen 64  scopeid 0x20
        ether 02:74:c2:09:f9:7a  txqueuelen 1000  (Ethernet)
        RX packets 70134  bytes 98687728 (94.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6841  bytes 497286 (485.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 648 (648.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 648 (648.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Github repo: https://github.com/Anuddeeph/HMCTask.git

Linkedin article: https://www.linkedin.com/pulse/hybrid-multi-cloud-task1-launch-web-server-single-command-nalla