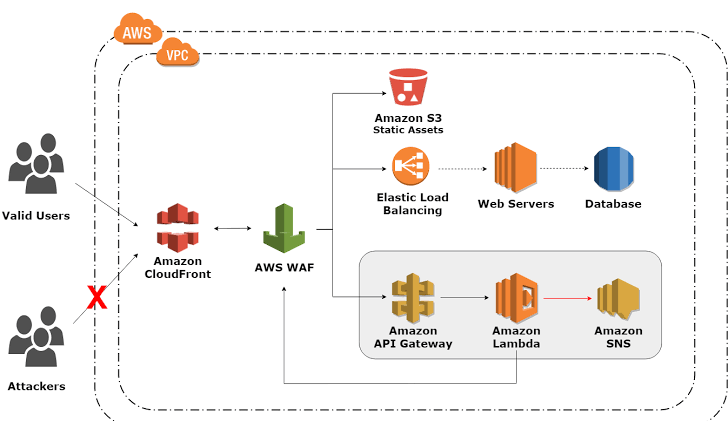
**Assignment on - Deploying Web Application on AWS Cloudfront**



Basic understanding of Assignment – in this assignment we have to create the infra consisting of services Amazon cloud front, attached to an application firewall which will host both static and dynamic website

The static part will be hosted on S3-bucket while the dynamic part will be hosted on EC2 instances which will be attached to an application load balancer, apart from that we will use the code build , code deploy And code pipeline to implement the project

Detailed explanation of project-

Terraform

S3bucket.tf -the S3 bucket is use to store the artifacts created during the code build , apart from that it is used for hosting the static website   bucket = aws\_s3\_bucket.bucket.bucket

   index\_document {

    suffix = "index.html"

  }

  error\_document {

    key = "/index.html"

  }

}

* 01-virtualprivatecloud .tf – Is **a private cloud computing environment contained within a public cloud**. Essentially, a VPC provide logically isolated environment in the cloud inside the VPC we have created different services such as,
* **security groups –** is act as a virtual firewall for controlling the incoming and outgoing traffic which allows or block the Ports

ingress {

description      = "Allow Port 8000"

from\_port        = 8000

to\_port          = 8000

protocol         = "tcp"

cidr\_blocks      = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

egress {

description = "Allow all ip and ports outboun"

from\_port   = 0

to\_port     = 0

protocol    = "-1"

cidr\_blocks = ["0.0.0.0/0"]

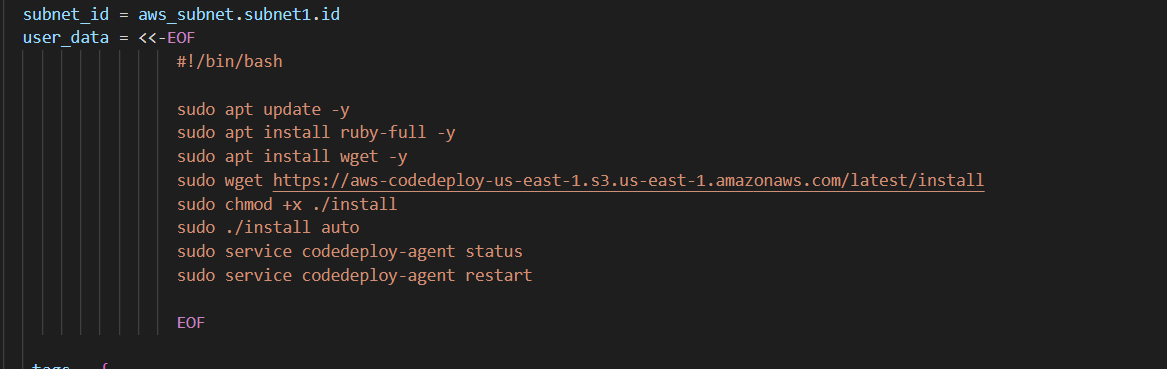
}

}

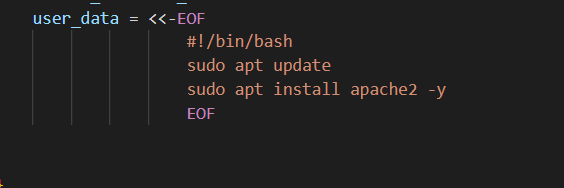
1. **Internet gateway** – the basic purpose of gateway is to allow the communication between the VPC and internet
2. **Subnet -**it is use to define the IP address of the cider block
3. **Rotatable.** the file consist of basic creation of the route table and route which uses the default configuration of the rote and an resource that associate the route table

* **ALB.tf –** the file contains the Resource of application load balancer which distribute the load among the different Instance we have create(instance1,instance 2)apart form that it consist of the target groups which target the two subnet created at different availability zone and a listener group and listen to listen to port 80
* **Instances –** we have created two instance where at one with different applications

**Instance 1 .-** will have the os and ubuntu and code deploy agent is installed and attached to subnet 1

****

**Instance 2**  - will have basic Apache server update



* CICD-pipeline.tf - consist of the basic code for the

1. Code build – the basic purpose of the code build is to build the code from source artifact created in the source stage of pipeline

stage {

    name = "Source"

    action {

      name = "name"

      category = var.category

      owner = "ThirdParty"

      provider = "GitHub"

      version = "1"

      output\_artifacts = ["SourceArtifact"]

1. Code deploy – now that source artifact will be used as the input for build stage and that give the output of build artifact

3 now in final stage these artifact will be deployed at S3 (source artifact as static) and at EC2

Code for deployment for EC2

  name = "Deploy\_to\_EC2"

    action {

      name = "Deploy"

      category = "Deploy"

      owner = "AWS"

      provider = "CodeDeploy"

      input\_artifacts = ["BuildArtifact"]

      version = "1"

for S3 (source artifact as static)

 stage {

    name = "Deploy"

    action {

      name            = "Deploy"

      category        = "Deploy"

      owner           = "AWS"

      provider        = "S3"

      input\_artifacts = ["SourceArtifact"]

      version         = "1"

* **Codebuild.tf –** the file consist of IAM role and policy for the code

Build along with the resource along with the source destination

   source {

    type            = "GITHUB"

    location        = "https://github.com/harshhhit/djangot2.git"

    git\_clone\_depth = 1

  }

  tags = {

    Environment = "Test"

  }

* Code deploy- will create the Police and role for the deploy along with application and deploy group is created
* **Cloud front .tf-** CloudFront is **a web service that speeds up distribution of your static and dynamic web content, by storing the cache at edge location. In this code we have created a cloud front and configure it with S3 and loadbalancer**

**configure it with S3**

     custom\_origin\_config {

      http\_port = 80

      https\_port = 443

      origin\_protocol\_policy = "http-only"

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

**configure it with load balancer –**

  custom\_origin\_config {

    http\_port = 80

    https\_port = 443

    origin\_protocol\_policy = "http-only"

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

    }

  }

  enabled             = true

  is\_ipv6\_enabled     = true

  # comment             = "Some comment"

  # default\_root\_object = "index.html"

  default\_cache\_behavior {

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

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

    target\_origin\_id = "lbattached"

**blocking the region**

  restrictions {

    geo\_restriction {

      restriction\_type = "blacklist"

      locations        = ["US", "CA", "GB", "DE"]

    }

  }

* **WAF.tf(Application firewall)** – the application firewall let you monitor the Htpp and htpps request that are foreword to the cloud front in this file we created the WAF