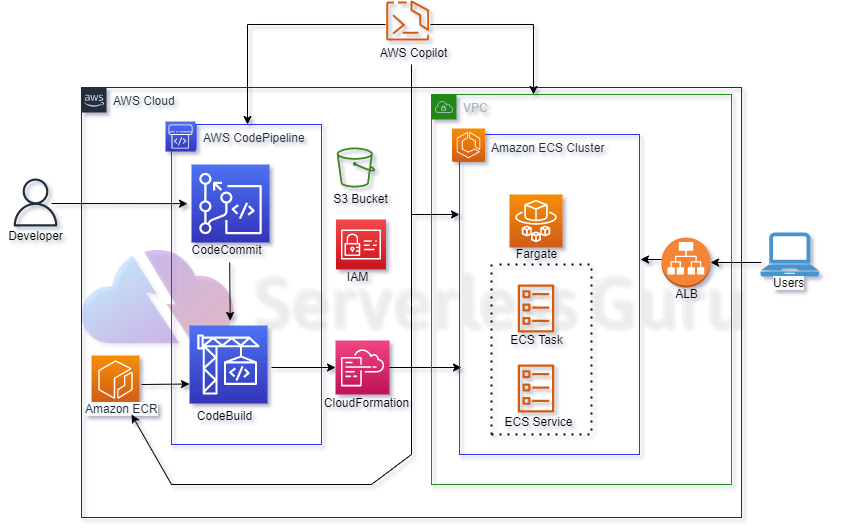
**Assignment** -Creating Ecs-Farget service using Code pipeline

**Basic understanding of assignment –** In this assignment we are going to deploy Two Fargate service inside Amazon ECS cluster through code pipeline and code deploy using terraform so the basic requirement we need for the assignment is virtual private cloud(VPC) ,amazon ECR, Amazon ECS cluster, IAM role and policies , code build ,code pipeline and S3 bucket

* **Terraform:**
* 01-virtual \_private\_cloud .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. **02 var-**vpc.tf – is the file where we define the veriables we have allocated to the VPC

* 03-IAM.tf – contains the basic IAM Roles and Policies that are required the purpose is to allow access to user or services to have compliance ,have basic polices and Roles for

**Codebuild -**  policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action   = ["codecommit:GitPull"]

Effect   = "Allow"

Resource = "\*"

},

{

Action = [

"ecr:BatchCheckLayerAvailability",

"ecr:GetDownloadUrlForLayer",

"ecr:BatchGetImage",

"ecr:CompleteLayerUpload",

"ecr:GetAuthorizationToken",

"ecr:InitiateLayerUpload",

"ecr:PutImage",

"ecr:UploadLayerPart"]

Effect   = "Allow"

Resource = "\*"

},

{

Action = [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"]

Effect   = "Allow"

Resource = "\*"

},

{

Action = [

"s3:PutObject",

"s3:GetObject",

"s3:GetObjectVersion",

"s3:GetBucketAcl",

"s3:GetBucketLocation"]

Effect   = "Allow"

Resource = "\*"

},

{

"Effect": "Allow",

"Action": [

"ecr-public:\*",

"sts:GetServiceBearerToken"

],

"Resource": "\*"

}

]

Allow code build to use the objects of S3 , ECR service

* **Route\_table.tf-** 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 services

apart form that it consist of the target groups which target the two subnet created at different availability zone and a listener group

**ECS-ECR.tf-** Consist of the ECR repository where the docker image will be stored once it is build through code-deploy

ECS service consist of task definition or the basic requirement of the cluster such as family, network used ,CPU and the memory required

family                   = "HTTPserver"

network\_mode             = "awsvpc"

requires\_compatibilities = ["FARGATE"]

cpu                      = 256

memory                   = 512

execution\_role\_arn       = "${data.aws\_iam\_role.ecs-task.arn}"

container\_definitions = jsonencode([

{

name   = "golang-container"

image  = aws\_ecrpublic\_repository.foo.repository\_uri #URI

cpu    = 256

memory = 512

portMappings = [

{

containerPort = 8000

}

And the service we configuring the network and load balancer

network\_configuration {

subnets          = ["${aws\_subnet.pub-subnets[0].id}", "${aws\_subnet.pub-subnets[1].id}"]

security\_groups  = ["${aws\_security\_group.sg1.id}"]

assign\_public\_ip = true

}

load\_balancer {

target\_group\_arn = "${aws\_lb\_target\_group.tg-group.arn}"

container\_name   = "golang-container"

container\_port   = "8000"

}

}

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

1. Code build – the basic purpose of the code build is to build the image for the cluster using the buildspec.yml file

Which have the three stages

A)- pre\_build use to login the repository,

B)- build use to build the docker image ,

3)-final stage is to push the image

Pipeline – to automate the code as an update is made to code