**Task - 1**  
1)Go Application

● A simple Go application that serves "Hello from Botgauge" at the /hello path.

● Dockerfile to containerize the application  
  
  
  
apt-get update  
  
mkdir hello-botgauge

cd hello-botgauge  
  
Create a new file named main.go and add the following content

package main

import (

"fmt"

"net/http"

)

func helloHandler(w http.ResponseWriter, r \*http.Request) {

fmt.Fprintln(w, "Hello from Botgauge")

}

func main() {

http.HandleFunc("/hello", helloHandler)

http.ListenAndServe(":8080", nil)

}

Create a Dockerfile  
  
FROM golang:1.18-alpine

WORKDIR /app

COPY go.mod ./

RUN go mod download

COPY . .

RUN go build -o /hello-botgauge

EXPOSE 8080

CMD ["/hello-botgauge"]

create a file go.mod  
  
module hello-botgauge

go 1.18

require (

// your dependencies here

)

Then install golang  
  
  
apt install golang-go

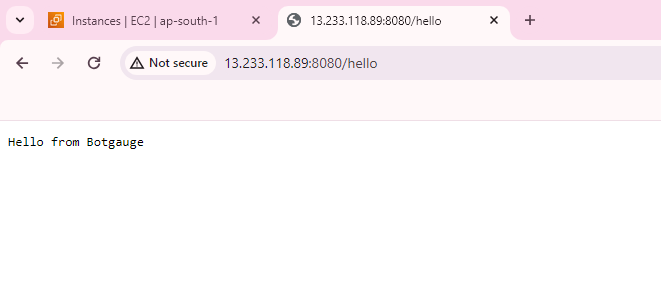
go mod init hello-botgauge

Install docker  
  
apt install docker.io  
  
  
Build the Docker image

docker build -t hello-botgauge .

docker run -d --name task -p 8080:8080 hello-botgauge

Hit this   
  
http://13.233.118.89:8080/hello



**Task 2**

Pipeline to build the Go application

1.First create a github repo  
  
2. Create a docker hub account

3. Create a workflow file in the .gihub/workflows directory of your repo

name: Build and Deploy

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout Repository

uses: actions/checkout@v2

- name: Set up Go

uses: actions/setup-go@v2

with:

go-version: 1.18

- name: Build Go Application

run: go build -o hello-botgauge

- name: Docker Build and Push

uses: docker/build-push-action@v2

with:

push: true

tags: username/hello-botgauge:latest

**Task 3**

Terraform Configuration

● Terraform scripts to create an EKS cluster.

● Any necessary add-ons (e.g., IAM roles, VPC setup, security groups, node groups,

etc.).

Install terraform   
  
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

mkdir eks-terraform

cd eks-terraform  
  
  
  
create a main.tf file  
  
  
provider "aws" {

region = var.region

}

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

}

resource "aws\_subnet" "public\_subnet" {

count = 2

vpc\_id = aws\_vpc.main.id

cidr\_block = cidrsubnet(aws\_vpc.main.cidr\_block, 8, count.index)

map\_public\_ip\_on\_launch = true

availability\_zone = element(data.aws\_availability\_zones.available.names, count.index)

}

data "aws\_availability\_zones" "available" {}

resource "aws\_iam\_role" "eks\_cluster\_role" {

name = "eks\_cluster\_role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "eks.amazonaws.com"

}

},

]

})

}

resource "aws\_iam\_role\_policy\_attachment" "eks\_cluster\_AmazonEKSClusterPolicy" {

policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"

role = aws\_iam\_role.eks\_cluster\_role.name

}

resource "aws\_eks\_cluster" "eks\_cluster" {

name = var.cluster\_name

role\_arn = aws\_iam\_role.eks\_cluster\_role.arn

vpc\_config {

subnet\_ids = aws\_subnet.public\_subnet[\*].id

}

depends\_on = [aws\_iam\_role\_policy\_attachment.eks\_cluster\_AmazonEKSClusterPolicy]

}

resource "aws\_iam\_role" "eks\_node\_group\_role" {

name = "eks\_node\_group\_role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [

{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "ec2.amazonaws.com"

}

},

]

})

}

resource "aws\_iam\_role\_policy\_attachment" "eks\_worker\_AmazonEKSWorkerNodePolicy" {

policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"

role = aws\_iam\_role.eks\_node\_group\_role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks\_worker\_AmazonEKS\_CNI\_Policy" {

policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"

role = aws\_iam\_role.eks\_node\_group\_role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks\_worker\_AmazonEC2ContainerRegistryReadOnly" {

policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"

role = aws\_iam\_role.eks\_node\_group\_role.name

}

resource "aws\_eks\_node\_group" "eks\_node\_group" {

cluster\_name = aws\_eks\_cluster.eks\_cluster.name

node\_group\_name = "my-node-group"

node\_role\_arn = aws\_iam\_role.eks\_node\_group\_role.arn

subnet\_ids = aws\_subnet.public\_subnet[\*].id

scaling\_config {

desired\_size = var.desired\_size

max\_size = var.max\_size

min\_size = var.min\_size

}

depends\_on = [

aws\_iam\_role\_policy\_attachment.eks\_worker\_AmazonEKSWorkerNodePolicy,

aws\_iam\_role\_policy\_attachment.eks\_worker\_AmazonEKS\_CNI\_Policy,

aws\_iam\_role\_policy\_attachment.eks\_worker\_AmazonEC2ContainerRegistryReadOnly

]

}

output "cluster\_endpoint" {

value = aws\_eks\_cluster.eks\_cluster.endpoint

}

output "cluster\_name" {

value = aws\_eks\_cluster.eks\_cluster.name

}  
  
  
  
create a variables.tf file  
  
  
variable "region" {

description = "The AWS region to deploy the EKS cluster"

type = string

default = "us-west-2"

}

variable "vpc\_cidr" {

description = "The CIDR block for the VPC"

type = string

default = "10.0.0.0/16"

}

variable "cluster\_name" {

description = "The name of the EKS cluster"

type = string

default = "my-eks-cluster"

}

variable "desired\_size" {

description = "The desired number of worker nodes"

type = number

default = 2

}

variable "max\_size" {

description = "The maximum number of worker nodes"

type = number

default = 3

}

variable "min\_size" {

description = "The minimum number of worker nodes"

type = number

default = 1

}  
  
  
  
  
**Initialize Terraform:**

terraform init  
  
  
Review the Terraform plan:

terraform plan  
  
  
Apply the Terraform plan:

terraform apply