International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 06, June-2021

Deploying Deep Learning using AWS and Devops

M Pavani[#1], Raaga Sindhu M[#2], K Shrutha Keerthi[#3]
Department Of Electronics And Computer Engineering
J.N.T.U, Hyderabad ,
Sreenidhi Institute Of Science And Technology,
Ghatkesar, Yamnampet ,Hyderabad,India

***Abstract*— Now a days the world is moving towards automation, any small or big task to be done, automation is preferred. With the power of Docker Engine, we create customized docker image from CentOS official docker image. This image consists the software and python libraries. The generated Deep Learning model is loaded into a backend python code. In the front end, a form is displayed to the user. As the user enters the required values, they will be carried over to the backend server. The backend code will get the data entered by user using GET request and, these values act as input to the pre-trained model, gives the predicted output. This result is sent to the user. This backend process is run by the Flask framework of python which provides an inbuilt server.**

Keywords*— *Docker, Flask framework, Machine Learning, AWS(Amazon Web Services),docker image

1.INTRODUCTION

In the realm of customary programming advancement, a bunch of practices known as DevOps have made it conceivable to transport programming to creation in minutes and to keep it running dependably. This methodology has been fruitful to such an extent that numerous organizations are now proficient at it, we can apply same thing for Machine Learning (ML). The underlying driver is that there's a key contrast among ML and conventional programming: L isn't simply code, it's code in addition to data. A ML model, the antique that user can wind up placing underway, is made by applying a calculation to a mass of preparing information, which will influence the conduct of the model underway.

MLOps is a bunch of practices that consolidates Machine Learning, DevOps, which intends to send and keep up ML frameworks underway dependably and proficiently. MLOps is the way toward taking a trial Machine Learning model into a creation framework. The word is a compound of "ML" and the ceaseless advancement practice of DevOps in the product field. ML models are tried and created in confined trial frameworks. At the point when a calculation is fit to be dispatched, MLOps is drilled between Data Scientists, DevOps, and Machine Learning architects to progress the calculation to creation of frameworks. Like DevOps or DataOps approaches, MLOps looks to build computerization and improve the nature of creation models, while additionally zeroing in on business and administrative prerequisites. While MLOps began as a bunch of best practices, it is gradually advancing into a free way to deal with ML lifecycle. MLOps applies to the whole lifecycle - from integrating with model generation, orchestration, and deployment, to health, diagnostics, governance, and business metrics.

2. EXISTING SYSTEMS

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains tothe field of machine learning. In its application across business problems, machine learning is also referred to as predictive analytics. In this every step involves a manual approach from developing code, building model, testing, deploying, etc.

3.PROPOSED SYSTEMS

MLOps (machine learning operations) is a practice that aims to make developing and maintaining production machine learning seamless and efficient. While MLOps is relatively nascent, the data science community generally agrees that it’s an umbrella term for best practices and guiding principles around machine learning – not a single technical solution.

MLOps is not dependent on a single technology or platform. However, technologies play a significant role in practical implementations of MLOps, similarly to how adopting Scrum often culminates in setting up and onboarding the whole team. Therefore, the project to rethink machine learning from an operational perspective is often about adopting the guiding principles and making decisions on infrastructure that will support the organization going forward.

4.MODEL BUILDING AND TRAINING

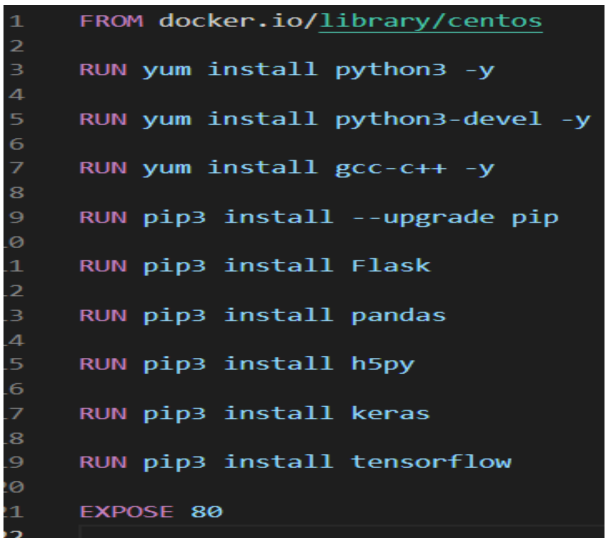
A model is being built for the specified dataset, using the Keras, Tensorflow libraries of Python and Jupyter notebook as editor. This model is further is copied into the Docker image for prediction. Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. TensorFlow is a free and open-source software library for machine learning. It can be usedacross a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. The generated model is saved in the HDF (Hierarchical Data Format) which the extension ‘h5’. For this Keras uses the h5py python library. Pandas Python library is used to import the dataset.

5. IMAGE BUILDING

5.1 Creating a Docker Image

A Dockerfile is a simple text document that contains a series of commands which Docker uses to build an image. FROM, RUN, EXPOSE are several commands supported in Dockerfile. The docker build command builds animage from a Dockerfile to build the image from our above Dockerfile.

The image built using following Dockerfile, this image consists the required software Python3, Python3-devel, C++, python libraries Flask, Pandas, h5py, Keras and the port number 80 is exposed so that it can be connected to outside world.



```
FROM docker.io/library/centos

RUN yum install python3 -y

RUN yum install python3-devel -y

RUN yum install gcc-c++ -y

RUN pip3 install --upgrade pip

RUN pip3 install Flask

RUN pip3 install pandas

RUN pip3 install h5py

RUN pip3 install keras

RUN pip3 install tensorflow

EXPOSE 80
```

Figure 1 File containing docker image built from base image CentOS

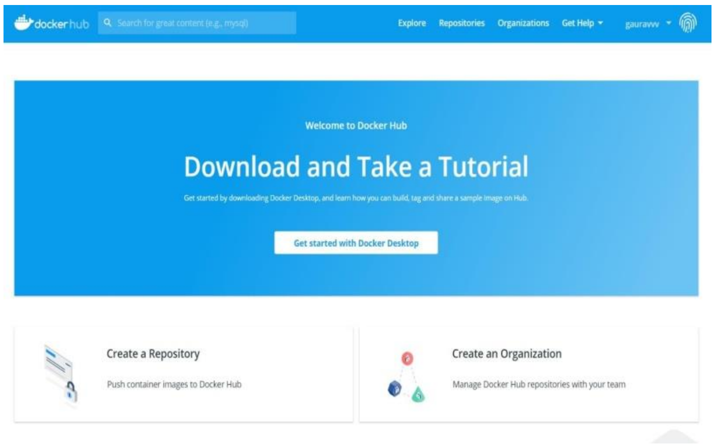
The docker build command builds an image from a Dockerfile. To build the image from our above Dockerfile.Publish Image to Docker Hub

To publish our Docker images to Docker Hub, these are the steps

Step 1: *Sign Up for Docker Hub*

Step 2*: Create a Repository on Docker Hub*

Step 3*: Push Image to Docker Hub*



Welcome to Docker Hub

Download and Take a Tutorial

Get started by downloading Docker Desktop, and learn how you can build, tag and share a sample image on Hub.

Create a Repository

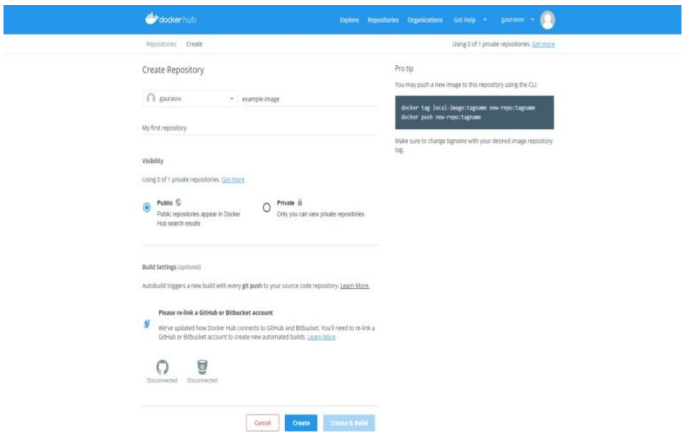
Push container images to Docker Hub

Create an Organization

Manage Docker Hub repositories with your team

docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
gauravvv
Get started with Docker Desktop

Figure 2 Docker Hub Window to Create Repository



Create Repository

gauravvv / example-image

My first repository

Visibility

Using 0 of 1 private repositories. Get more

Public: Public repositories appear in Docker Hub search results

Private: Only you can view private repositories

Build Settings (optional)

Autobuild triggers a new build with every git push to your source code repository. Learn More.

Please re-link a GitHub or Bitbucket account

We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. Learn More

Pro tip

You may push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

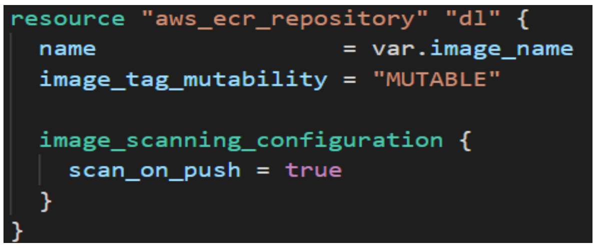
Make sure to change tagname with your desired image repository tag.

Figure 3 Creating Repository in Docker Hub

6.DEPLOYING

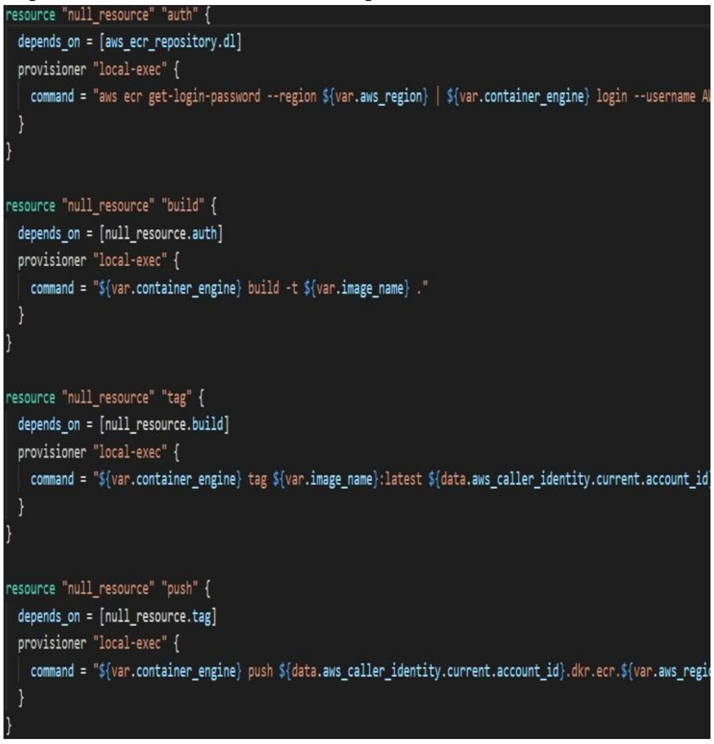
6.1 Deploying Infrastructure using Terraform

The private ECR registry is created



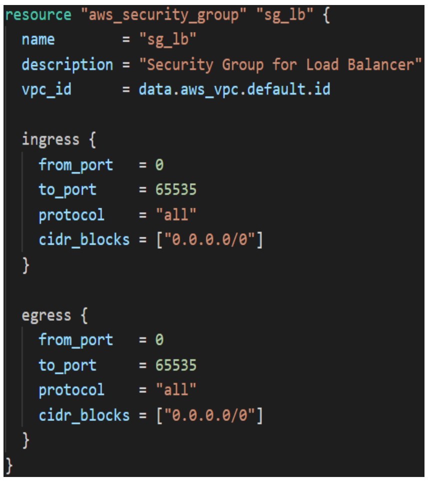
```
resource "aws_ecr_repository" "dl" {
  name                 = var.image_name
  image_tag_mutability = "MUTABLE"

  image_scanning_configuration {
    scan_on_push = true
  }
}
```

The authentication to the AWS private registry is done, the image is built in which the model and python files are added to the image at last the image is pushed to the ECR, these commands are run on the local host with the provisioner with "local-exec" option

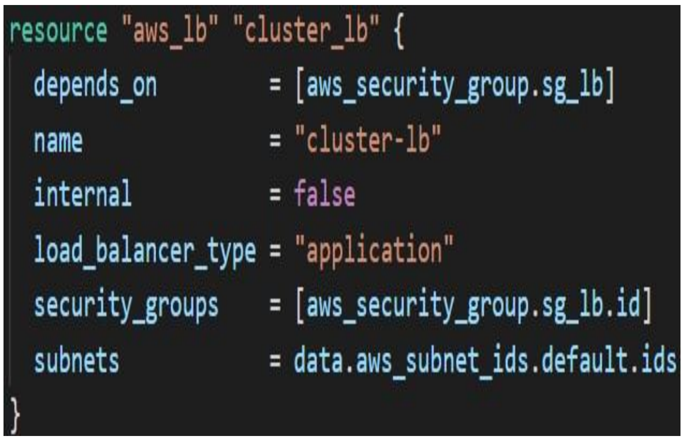


```
resource "null_resource" "auth" {
  depends_on = [aws_ecr_repository.dl]
  provisioner "local-exec" {
    command = "aws ecr get-login-password --region ${var.aws_region} | ${var.container_engine} login --username A
  }
}

resource "null_resource" "build" {
  depends_on = [null_resource.auth]
  provisioner "local-exec" {
    command = "${var.container_engine} build -t ${var.image_name} ."
  }
}

resource "null_resource" "tag" {
  depends_on = [null_resource.build]
  provisioner "local-exec" {
    command = "${var.container_engine} tag ${var.image_name}:latest ${data.aws_caller_identity.current.account_id
  }
}

resource "null_resource" "push" {
  depends_on = [null_resource.tag]
  provisioner "local-exec" {
    command = "${var.container_engine} push ${data.aws_caller_identity.current.account_id}.dkr.ecr.${var.aws_regi
  }
}
```

This code snippet indicates the resource Security Group in AWS allowing all types of connections on all ports



```
resource "aws_security_group" "sg_lb" {
  name        = "sg_lb"
  description = "Security Group for Load Balancer"
  vpc_id      = data.aws_vpc.default.id

  ingress {
    from_port   = 0
    to_port     = 65535
    protocol    = "all"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "all"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```



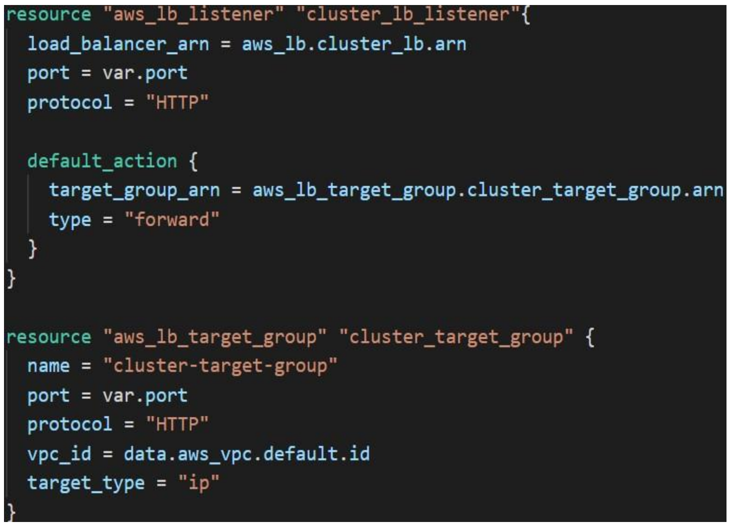
```
resource "aws_lb" "cluster_lb" {
  depends_on         = [aws_security_group.sg_lb]
  name               = "cluster-lb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.sg_lb.id]
  subnets            = data.aws_subnet_ids.default.ids
}
```

The load balancer is created with the previously created security group attached to this.

The AMI role created will be attached to the ECS

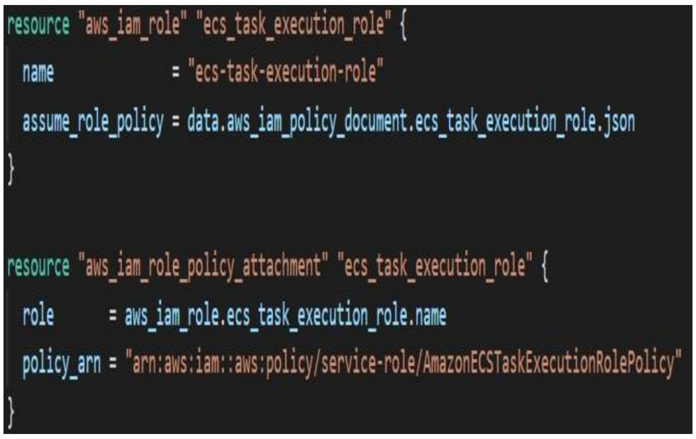
The load balancer is allowed to listen on port number 80 and HTTP protocol, the target group of the load balancer is the containers in the ECS cluster

task



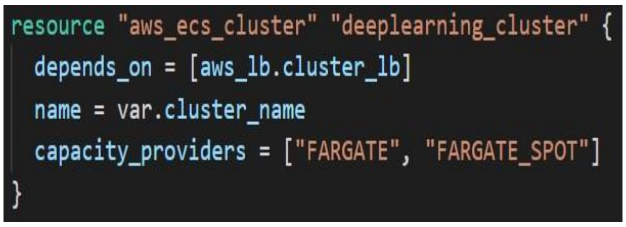
```
resource "aws_lb_listener" "cluster_lb_listener"{
  load_balancer_arn = aws_lb.cluster_lb.arn
  port = var.port
  protocol = "HTTP"

  default_action {
    target_group_arn = aws_lb_target_group.cluster_target_group.arn
    type = "forward"
  }
}

resource "aws_lb_target_group" "cluster_target_group" {
  name = "cluster-target-group"
  port = var.port
  protocol = "HTTP"
  vpc_id = data.aws_vpc.default.id
  target_type = "ip"
}
```

execution Role policy



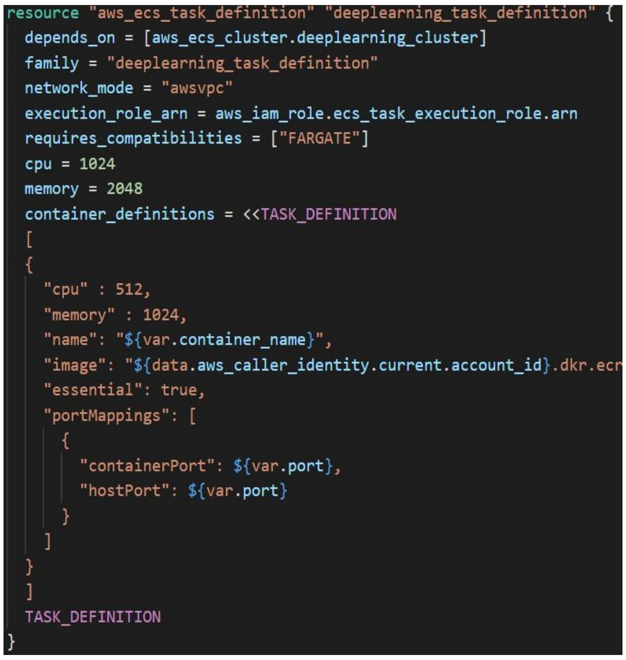
```
resource "aws_iam_role" "ecs_task_execution_role" {
  name               = "ecs-task-execution-role"
  assume_role_policy = data.aws_iam_policy_document.ecs_task_execution_role.json
}

resource "aws_iam_role_policy_attachment" "ecs_task_execution_role" {
  role       = aws_iam_role.ecs_task_execution_role.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
}
```

The cluster is created using the FARGATE type capacity provider



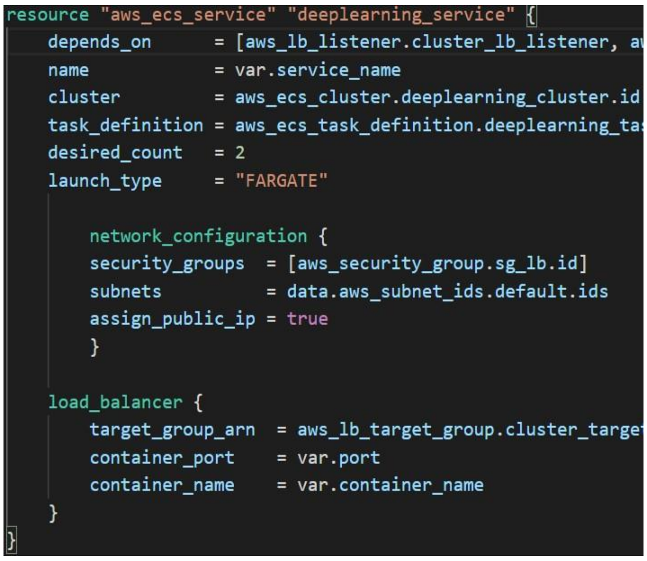
```
resource "aws_ecs_cluster" "deeplearning_cluster" {
  depends_on = [aws_lb.cluster_lb]
  name = var.cluster_name
  capacity_providers = ["FARGATE", "FARGATE_SPOT"]
}
```

The ECS task definition is the details of the container containing image name, computing resources, workingport number, etc.



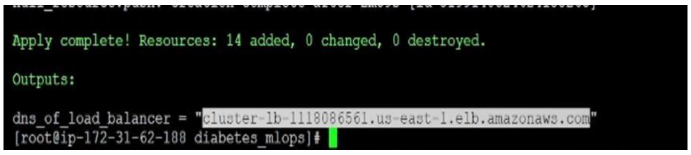
```
resource "aws_ecs_task_definition" "deeplearning_task_definition" {
  depends_on = [aws_ecs_cluster.deeplearning_cluster]
  family = "deeplearning_task_definition"
  network_mode = "awsvpc"
  execution_role_arn = aws_iam_role.ecs_task_execution_role.arn
  requires_compatibilities = ["FARGATE"]
  cpu = 1024
  memory = 2048
  container_definitions = <<TASK_DEFINITION
  [
  {
    "cpu" : 512,
    "memory" : 1024,
    "name": "${var.container_name}",
    "image": "${data.aws_caller_identity.current.account_id}.dkr.ecr
    "essential": true,
    "portMappings": [
      {
        "containerPort": ${var.port},
        "hostPort": ${var.port}
      }
    ]
  }
  ]
  TASK_DEFINITION
}
```

The ECS service will use the use the task definition to deploy the containers in ECS cluster



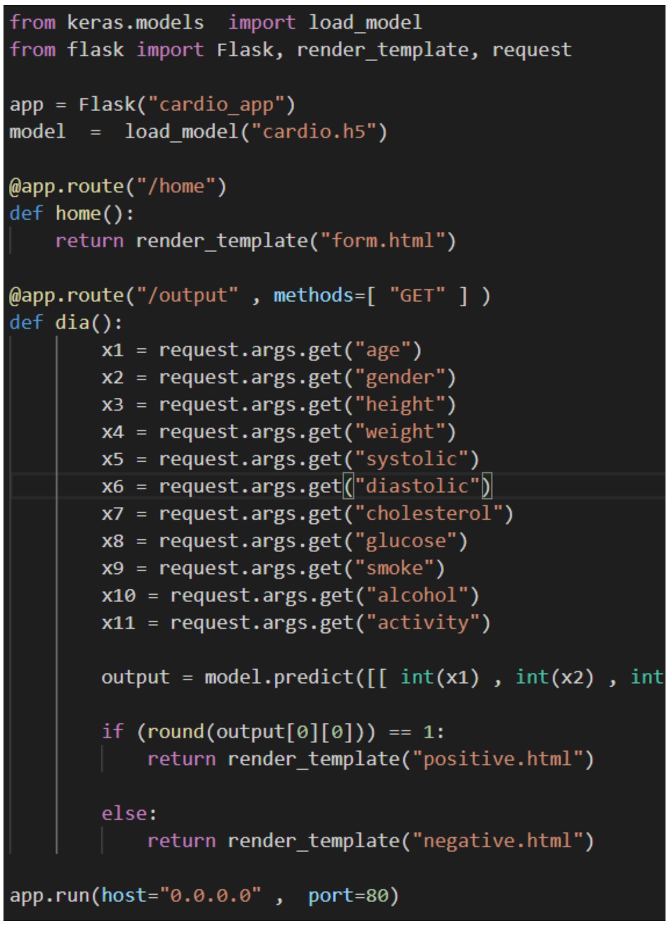
```
resource "aws_ecs_service" "deeplearning_service" {
    depends_on      = [aws_lb_listener.cluster_lb_listener, a
    name            = var.service_name
    cluster         = aws_ecs_cluster.deeplearning_cluster.id
    task_definition = aws_ecs_task_definition.deeplearning_ta
    desired_count   = 2
    launch_type     = "FARGATE"

        network_configuration {
        security_groups  = [aws_security_group.sg_lb.id]
        subnets          = data.aws_subnet_ids.default.ids
        assign_public_ip = true
        }

    load_balancer {
        target_group_arn  = aws_lb_target_group.cluster_targe
        container_port    = var.port
        container_name    = var.container_name
    }
}
```

After all the resources creation is completed, terraform prints the DNS of load balancer, using this the users can connect to the server



```
Apply complete! Resources: 14 added, 0 changed, 0 destroyed.

Outputs:

dns_of_load_balancer = "cluster-lb-1118086561.us-east-1.elb.amazonaws.com"
[root@ip-172-31-62-188 diabetes_mlops]#
```

This python file loads the model in the that is copied into the image, the inbuilt server of Flask framework starts listening on port 80, when a user hits the server with “/home” path-based routing the user is displayed with a form where the user enters the details related to his/her cardiac disease



```
from keras.models  import load_model
from flask import Flask, render_template, request

app = Flask("cardio_app")
model   =  load_model("cardio.h5")

@app.route("/home")
def home():
    return render_template("form.html")

@app.route("/output" , methods=[ "GET" ] )
def dia():
        x1 = request.args.get("age")
        x2 = request.args.get("gender")
        x3 = request.args.get("height")
        x4 = request.args.get("weight")
        x5 = request.args.get("systolic")
        x6 = request.args.get("diastolic")
        x7 = request.args.get("cholesterol")
        x8 = request.args.get("glucose")
        x9 = request.args.get("smoke")
        x10 = request.args.get("alcohol")
        x11 = request.args.get("activity")

        output = model.predict([[ int(x1) , int(x2) , int

        if (round(output[0][0])) == 1:
            return render_template("positive.html")

        else:
            return render_template("negative.html")

app.run(host="0.0.0.0" ,  port=80)
```

7. CONCLUSION

We have built a model on diabetes in the same way we can perform our tasks on other models too. For example, we have built another model which detects the risk of cardiac disease. Here we have changed the image which consists the model. Here we used a load balancer that will distribute the traffic among all the container. As FARGATE is used which is completely serverless makes the work easier

8. FUTURE SCOPE

Here we have used ECS resource of AWS. This can be further done using the resource EKS (Elastic Kubernetes Service).Kubernetes is a container orchestration system.
There are many features provided by Kubernetes like Auto Scaling, Load Balancing, etc. Kubernetes is also provided by GCP (Google Cloud Platform) in the name of GKE (Google Compute Engine), as Google is author of Kubernetes it provides Kubernetes in a way far better than other cloud providers.
The need of creating pipeline can be solved using AWS Code Build, GCP Cloud Build

9. REFERENCES

[1] Damain. A. Tamburri, Sustainable MLOps: Trends and Challenges Eindhoven University Of Technology JADS, Hertogenbosch, The Netherlands.
[2] Gaurav Singhal, Create Docker Images for Docker Hub, July 15, 2019, https://www.pluralsight.com/guides/create-docker-images-docker-hub
[3] https://www.terraform.io/guides/core-workflow.html
[4] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html
[5] https://keras.io/api/layers/
[6] https://www.tensorflow.org/api_docs/python/tf