**ATC Assignment**

Name: Antony Jebinraj Y

Role: DevOps Engineer

Email: jebin1848@gmail.com

Github: https://github.com/antonyjebin/atc\_project.git

**Steps:**

1. Install Git, Terraform, Kubectl installation.

2. Write tf code(terraform) for the EKS cluster and node creation.

3. Web application and Docker image creation..

4. Design Kubernetes deployment and service.

5. Webapplication deployment and service using kubectl cmd.

6. Setup Prometheus and Grafana.

# **Step :1 Install Git, Terraform, Kubectl installation.**

# (Note: I use lunix machine from start to end because debugging is easy and its light weight)

* Installs Git, Terraform, and kubectl, and sets up Jenkins to run on system startup on an Amazon Linux system.

|  |  |
| --- | --- |
|  | **#finally install kubectl**  **sudo curl -LO**  **https://storage.googleapis.com/kubernetes-release/release/v1.23.6/bin/linu x/amd64/kubectl**  **sudo chmod +x ./kubectl**  **sudo mkdir -p $HOME/bin && sudo cp ./kubectl $HOME/bin/kubectl &&**  **export PATH=$PATH:$HOME/bin** |

**Step 2: Write tf code(terraform) for the EKS cluster and node creation.**

The code block sets the AWS provider for Terraform to use the "us-west-2" region, using the key.

#provider.tf

provider "aws" {

region = "us-west-2"

}

This Terraform configuration sets up remote state storage in an S3 bucket named "terraform-webapp-eks-01" within the "us-west-2" region, using the key

#backend.tf

terraform {

backend "s3" {

bucket = "terraform-webapp-eks-01"

key = "eks/terraform.tfstate"

region = "us-west-2"

}

}

This Terraform variable definition file specifies the CIDR blocks for a VPC, along with its associated private and public subnets

#terraform.tfvars

vpc\_cidr\_block = "10.0.0.0/16"

private\_subnet\_cidr\_blocks=["10.0.1.0/24","10.0.2.0/24","10.0.3.0/24"]

public\_subnet\_cidr\_blocks=["10.0.4.0/24","10.0.5.0/24","10.0.6.0/24"]

This Terraform configuration declares variables for VPC CIDR block, lists of private subnet CIDR blocks, and lists of public subnet CIDR blocks.

#variable.tf

variable "vpc\_cidr\_block" {

type = string

}

variable "private\_subnet\_cidr\_blocks" {

type = list(string)

}

variable "public\_subnet\_cidr\_blocks" {

type = list(string)

}

This Terraform module creates a VPC with public and private subnets, enables NAT gateway, DNS hostnames, and assigns Kubernetes tags for Webapp deployment on AWS.

#vpc.tf

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

module "mywebapp-server-vpc" {

source = "terraform-aws-modules/vpc/aws"

name = "mywebapp-server-vpc"

cidr = var.vpc\_cidr\_block

private\_subnets = var.private\_subnet\_cidr\_blocks

public\_subnets = var.public\_subnet\_cidr\_blocks

azs = data.aws\_availability\_zones.azs.names

enable\_nat\_gateway = true

single\_nat\_gateway = true

enable\_dns\_hostnames = true

tags = {

"kubernetes.io/cluster/mywebapp-server-eks-cluster" = "shared"

}

public\_subnet\_tags = {

"kubernetes.io/cluster/mywebapp-server-eks-cluster" = "shared"

"kubernetes.io/role/elb" = 1

}

private\_subnet\_tags = {

"kubernetes.io/cluster/mywebapp-server-eks-cluster" = "shared"

"kubernetes.io/role/internal-elb" = 1

}

}

This Terraform module configures an Amazon EKS cluster named "mywebapp-server-eks-cluster" with version "1.24", accessible publicly, linked to a VPC and subnets, and managed node group with specified instance types for a development environment.

#eks-cluster.tf

module "eks" {

source = "terraform-aws-modules/eks/aws"

version = "~> 19.0"

cluster\_name = "mywebapp-server-eks-cluster"

cluster\_version = "1.24"

cluster\_endpoint\_public\_access = true

vpc\_id = module.mywebapp-server-vpc.vpc\_id

subnet\_ids = module.mywebapp-server-vpc.private\_subnets

tags = {

environment = "development"

application = "mywebapp-server"

}

eks\_managed\_node\_groups = {

dev = {

min\_size = 1

max\_size = 3

desired\_size = 2

instance\_types = ["t2.small"]

}

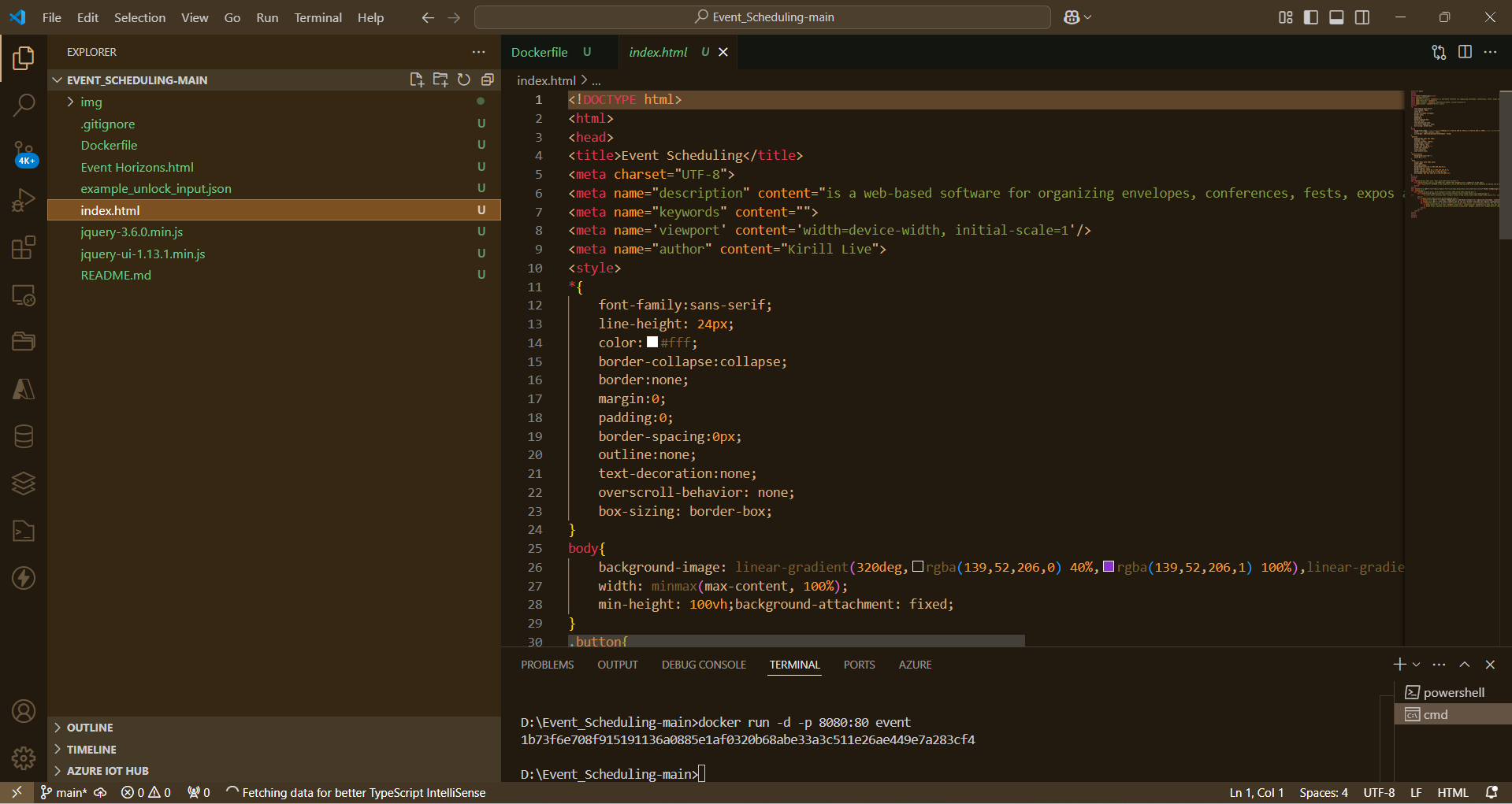
}

}

**Step 3: Web application and Docker image creation.**

Github: https://github.com/antonyjebin/atc\_project.git

(Note: In your mail you mentioned: web application shall have a simple static page of content. So I taken one simple even scheduling web based application)



Dockerfile for docker image creation

# Step 1: Use the official Nginx image to serve static content

FROM nginx:alpine

# Step 2: Copy all the HTML, CSS, and JS files into the nginx server's root directory

COPY . /usr/share/nginx/html

# Step 3: Expose port 80 so that the app is accessible on this port

EXPOSE 80

# Step 4: Start the Nginx server (default command for nginx)

CMD ["nginx", "-g", "daemon off;"]

Execute cmd for docker image creation:

docker build -t event-scheduling-app .

Log in to Docker Hub:

docker login

Tag the Docker Image:

docker tag event-scheduling-app antonyjebinraj/event-scheduling-app:latest

Push the Docker Image to Docker Hub

docker push antonyjebinraj/event-scheduling-app:latest

# **Step 4: Design Kubernetes deployment and service.**

This file defines a Kubernetes Deployment and Service configuration for a Node.js application, specifying resource limits, ports, and replicas.

#k8s-deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp-deployment

spec:

replicas: 2

selector:

matchLabels:

app: webapp

template:

metadata:

labels:

app: webapp

spec:

containers:

- name: webapp

image: antonyjebinraj/event-scheduling-app:latest

resources:

limits:

memory: "356Mi"

cpu: "500m"

ports:

- containerPort: 4000

#k8s-svc.yml

apiVersion: v1

kind: Service

metadata:

name: webapp-service

spec:

type: NodePort

selector:

app: webapp

ports:

- port: 8080

targetPort: 80

protocol: TCP

# **Step 5: Webapplication deployment and service using kubectl cmd.**

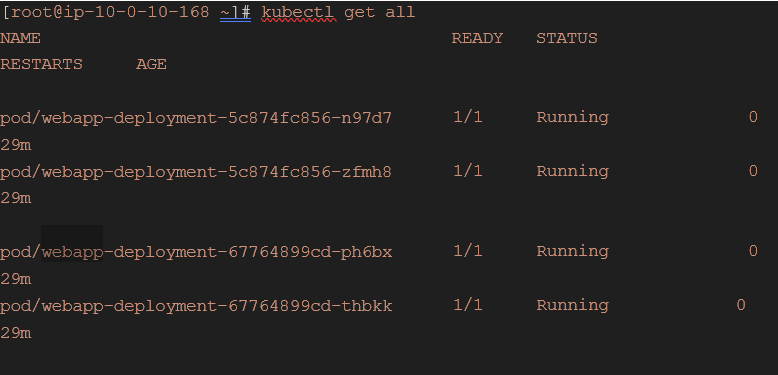
This cmd use for deploy our web app on kubernetes cluster:

kubectl apply -f k8s-deployment.yml

This cmd use for create service to expose application on console on kubernetes cluster:

kubectl apply -f k8s-svc.yml

Here, You can all deployment pods on kubernetes cluster:



**Step 6: Setup Prometheus and Grafana.**

Install Prometheus:

Create a Prometheus YAML file for deployment.

Create a Prometheus Configuration:

apiVersion: apps/v1

kind: Deployment

metadata:

name: prometheus

labels:

app: prometheus

spec:

replicas: 1

selector:

matchLabels:

app: prometheus

template:

metadata:

labels:

app: prometheus

spec:

containers:

- name: prometheus

image: prom/prometheus:latest

ports:

- containerPort: 9090

volumeMounts:

- name: prometheus-config

mountPath: /etc/prometheus/prometheus.yml

subPath: prometheus.yml

volumes:

- name: prometheus-config

configMap:

name: prometheus-config

Create a ConfigMap for Prometheus configuration.

apiVersion: v1

kind: ConfigMap

metadata:

name: prometheus-config

data:

prometheus.yml: |

global:

scrape\_interval: 15s

scrape\_configs:

- job\_name: 'kubernetes-apiservers'

kubernetes\_sd\_configs:

- role: endpoints

scheme: https

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

relabel\_configs:

- source\_labels: [\_\_meta\_kubernetes\_namespace, \_\_meta\_kubernetes\_service\_name, \_\_meta\_kubernetes\_endpoint\_port\_name]

action: keep

regex: default;kubernetes;https

- job\_name: 'kubernetes-nodes'

scheme: https

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt

bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token

kubernetes\_sd\_configs:

- role: node

- job\_name: 'kubernetes-pods'

kubernetes\_sd\_configs:

- role: pod

Install Grafana:

Create a Grafana YAML file for deployment.

apiVersion: apps/v1

kind: Deployment

metadata:

name: grafana

labels:

app: grafana

spec:

replicas: 1

selector:

matchLabels:

app: grafana

template:

metadata:

labels:

app: grafana

spec:

containers:

- name: grafana

image: grafana/grafana:latest

ports:

- containerPort: 3000

volumeMounts:

- name: grafana-storage

mountPath: /var/lib/grafana

volumes:

- name: grafana-storage

emptyDir: {}

**-----**

apiVersion: v1

kind: Service

metadata:

name: grafana

labels:

app: grafana

spec:

type: NodePort

selector:

app: grafana

ports:

- port: 3000

targetPort: 3000

nodePort: 32000

Successfully set up all these things!