**SESSION - 81**

**Catalogue/Dockerfile**

FROM node:20-alpine3.20 AS builder

WORKDIR /opt/server

COPY package.json .

COPY \*.js .

RUN npm install

FROM node:20-alpine3.20

RUN addgroup -S roboshop && adduser -S roboshop -G roboshop

RUN apk update && apk add --no-cache --upgrade musl openssl

ENV MONGO="true" \

MONGO\_URL="mongodb://mongodb:27017/catalogue"

WORKDIR /opt/server

USER roboshop

COPY --from=builder /opt/server /opt/server

CMD ["node","server.js"]

# FROM node:20-alpine3.21

# RUN addgroup -S roboshop && adduser -S roboshop -G roboshop

# WORKDIR /opt/server

# COPY package.json .

# COPY \*.js .

# RUN npm install

# RUN chown -R roboshop:roboshop /opt/server

# ENV MONGO="true" \

# MONGO\_URL="mongodb://mongodb:27017/catalogue"

# USER roboshop

# CMD ["node","server.js"]

--> Connect the jenkins

--> build the code

--> aws ecr describe-image-scan-findings \

--repository-name roboshop/frontend \

--image-id imageTag=1.0.0 \

--region us-east-1 \

--output json

Static source code analysis

Open source library scan

image scanning

Static application security testing

**Dynamic application security testing**

**======================================**

Category 1 and Category 2

DEV or SIT or PRE-PROD

--> cd catalogue

--> push the code

--> cd /terraform-aws-eks/

--> for i in 60-acm/ 70-frontend-alb; do cd $i; terraform apply -auto-approve; cd ..;done

--> you can do from the EKS

1. Allow ingress to create load balancer

2. We create loadbalancer, target group, listener, rule. We will add the pods IP to target group

**Terraform-aws-eks/70-frontend-alb/main.tf**

module "ingress\_alb" {

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

version = "9.16.0"

internal = false

name = "${var.project}-${var.environment}-ingress-alb" #roboshop-dev-backend-alb

vpc\_id = local.vpc\_id

subnets = local.public\_subnet\_ids

create\_security\_group = false

security\_groups = [local.ingress\_alb\_sg\_id]

enable\_deletion\_protection = false

tags = merge(

local.common\_tags,

{

Name = "${var.project}-${var.environment}-ingress-alb"

}

)

}

resource "aws\_lb\_listener" "ingress\_alb" {

load\_balancer\_arn = module.ingress\_alb.arn

port = "443"

protocol = "HTTPS"

ssl\_policy = "ELBSecurityPolicy-2016-08"

certificate\_arn = local.acm\_certificate\_arn

default\_action {

type = "fixed-response"

fixed\_response {

content\_type = "text/html"

message\_body = "<h1>Hello, I am from ingress ALB using HTTPS</h1>"

status\_code = "200"

}

}

}

resource "aws\_route53\_record" "ingress\_alb" {

zone\_id = var.zone\_id

name = "${var.environment}.${var.zone\_name}" #dev.daws84s.site

type = "A"

alias {

name = module.ingress\_alb.dns\_name

zone\_id = module.ingress\_alb.zone\_id # This is the ZONE ID of ALB

evaluate\_target\_health = true

}

}

resource "aws\_lb\_target\_group" "frontend" {

name = "${var.project}-${var.environment}-frontend" #roboshop-dev-catalogue

port = 8080

protocol = "HTTP"

vpc\_id = local.vpc\_id

deregistration\_delay = 120

target\_type = "ip"

health\_check {

healthy\_threshold = 2

interval = 5

matcher = "200-299"

path = "/"

port = 8080

timeout = 2

unhealthy\_threshold = 3

}

}

resource "aws\_lb\_listener\_rule" "frontend" {

listener\_arn = aws\_lb\_listener.ingress\_alb.arn

priority = 10

action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.frontend.arn

}

condition {

host\_header {

values = ["${var.environment}.${var.zone\_name}"] # https://dev.daws84s.site

}

}

}

**--> cd 70-frontend-alb**

**--> terraform plan**

**--> terraform apply -auto-approve**

--> we have to add targets

--> not adding vms. Directly ip address

--> target type should be ip.

--> check target groups

--> https:443 listner

--> cd k8-roboshop

--> aws eks update-kubeconfig --region us-east-1 --name roboshop-dev

--> kubectl get nodes

--> run our application

--> git clone https://github.com/daws-84s/k8-roboshop.git

--> cd k8-roboshop/

--> kubectl create ns roboshop

--> kubens roboshop

--> ls

--> cd mongodb/

--> kubectl apply -f mainfest.yaml

--> cd mysql/

--> kubectl apply -f mainfest.yaml

--> cd rabbitmq

--> kubectl apply -f mainfest.yaml

--> cd catalogue

--> kubectl apply -f mainfest.yaml

--> cd user

--> kubectl apply -f mainfest.yaml

--> cd cart

--> kubectl apply -f mainfest.yaml

--> cd shipping

--> kubectl apply -f mainfest.yaml

--> cd payment

--> kubectl apply -f mainfest.yaml

--> kubectl get pods

--> roboshop-dev

--> eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster roboshop-dev \

--approve

--> curl -o iam-policy.json https::/raw.githubusercontent.com/kubernates-signs/aws-load-balancer-controller/v2.13.4/docs/install/iam\_policy.json

--> eksctl create iamserviceaccount \

--cluster=roboshop-dev \

--namespace=kube-system \

--name=aws-load-balancer-controller \

--attach-policy-arn=arn:aws:iam::315069654700:policy/AWSLoadBalancerControllerIAMPolicy \

--override-existing-serviceaccounts \

--region us-east-1 \

--approve

--> roboshop-dev-frontend

--> kubectl get sa -n kube-system

--> cloud formation --> stacks in AWS account

--> kubectl get sa -n kube-system | grep aws-lod\*

--> helm repo add eks https://aws.github.io/eks-charts

--> helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=<cluster-name> --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller

--> kubectl get pods -n kube-system

--> kubectl logs aws-load-balancer-controller-5cd57969dc-6qv2m -n kube-system

--> helm uninstall aws-load-balancer-controller -n kube-system

--> helm repo add eks https://aws.github.io/eks-charts

--> helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=roboshop-dev --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set hostNetwork=true

--> helm repo add eks https://aws.github.io/eks-charts

--> helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=roboshop-dev --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller --set hostNetwork=true

--> kunectl get pods -n kube-system

--> curl <http://169.254.169.254/latest/meta-data/>

--> kubectl get pods -n kube-system

--> Target group binding is a custom resource (CR) that can expose your pods using an existing ALB TargetGroup or NLB TargetGroup.

**K8-roboshop/frontend/manifest-tgb.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: nginx-conf

namespace: roboshop

labels:

component: frontend

project: roboshop

tier: web

data:

nginx.conf: |

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log notice;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

keepalive\_timeout 65;

types\_hash\_max\_size 4096;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

include /etc/nginx/conf.d/\*.conf;

server {

listen 8080;

listen [::]:8080;

server\_name \_;

root /usr/share/nginx/html;

include /etc/nginx/default.d/\*.conf;

error\_page 404 /404.html;

location = /404.html {

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

location /images/ {

expires 5s;

root /usr/share/nginx/html;

try\_files $uri /images/placeholder.jpg;

}

location /api/catalogue/ { proxy\_pass http://catalogue:8080/; }

location /api/user/ { proxy\_pass http://user:8080/; }

location /api/cart/ { proxy\_pass http://cart:8080/; }

location /api/shipping/ { proxy\_pass http://shipping:8080/; }

location /api/payment/ { proxy\_pass http://payment:8080/; }

location /health {

stub\_status on;

access\_log off;

}

}

}

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

namespace: roboshop

# deployment labels

labels:

component: frontend

project: roboshop

tier: web

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: frontend

project: roboshop

tier: web

# This is pod definition

template:

metadata:

labels:

component: frontend

project: roboshop

tier: web

spec:

containers:

- name: frontend

image: joindevops/frontend:v1

volumeMounts:

- name: nginx-conf

mountPath: /etc/nginx/nginx.conf

subPath: nginx.conf

readOnly: true

volumes:

- name: nginx-conf

configMap:

name: nginx-conf

items:

- key: nginx.conf

path: nginx.conf

---

apiVersion: v1

kind: Service

metadata:

name: frontend

namespace: roboshop

labels:

component: frontend

project: roboshop

tier: web

spec:

selector:

component: frontend

project: roboshop

tier: web

ports:

- protocol: TCP

port: 80 # service port

targetPort: 8080 # container port

---

apiVersion: elbv2.k8s.aws/v1beta1

kind: TargetGroupBinding

metadata:

name: frontend

namespace: roboshop

spec:

serviceRef:

name: frontend # route traffic to the awesome-service

port: 80

targetGroupARN: arn:aws:elasticloadbalancing:us-east-1:315069654700:targetgroup/roboshop-dev-frontend/adeb3f8866bb6d81

--> Push the code

--> everything success and you can see pods here

--> target groups-> roboshop-dev-frontend

--> kubectl get pods

--> kubectl get svc

--> kubectl logs front-end

--> kubectl delete pod frontend-7c77c9f77b-swbzz

--> kubectl get pods

--> kubectl get svc

--> cd ../debug

--> kubectl apply -f mainfest.yaml

--> kubectl exec -it debug --bash

--> nslookup catalogue

--> telnet catalogue 8080

--> curl <http://catalogue:8080/health>

--> curl [http://user:8080/health](http://catalogue:8080/health)

--> curl [http://cart:8080/health](http://catalogue:8080/health)

--> curl [http://shipping:8080/health](http://catalogue:8080/health)

--> curl [http://payment:8080/health](http://catalogue:8080/health)

--> take duplicate session

--> k9s

-->