**Cloud & DevOps Engineer Assessment Task – Rakesh Marri**

**Overview**

Deploy a web application in a cloud-based Kubernetes solution, ensuring proper logging and monitoring are in place.

**Application Functionality**

The web application shall have a simple static page of content.

Please refer the Github link for all the configuration and deployment files: <https://github.com/Rakesh-Marri/K8-Web-Application/tree/main>

**Cloning the Git Repository in the Local**

*# git clone* [*https://github.com/Rakesh-Marri/K8-Web-Application*](https://github.com/Rakesh-Marri/K8-Web-Application/tree/main)

1. **Provision ECR and EKS cluster using Terraform**

Refer to the following link for Terraform files: <https://github.com/Rakesh-Marri/K8-Web-Application/tree/main/Terraform>

Pre-requisites:

1. Terraform and AWS CLI to be installed locally
2. AWS account and IAM user to be created with permissions to ECR and EKS

Configure the aws credentials

*# aws configure* -> provide the aws access key, secret key and the default region

Terraform files structure ->

variables.tf - contains the variables required for the provisioning of ECR and EKS

main.tf - contains the code for provisioning ECR and EKS

backend.tf - contains the details of S3 bucket for storing the state file

provider.tf - contains the aws provider details.

output.tf - contains the output variables required for displaying after apply

use the following terraform commands for Provisioning ECR, EKS and dependent services.

(change the directory to terraform files path -> cd terraform)

*# terraform fmt*  -> This command will format the configuration files.

*# terraform validate* -> This command will validate the syntax and configuration of terraform files.

*# terraform init* -> This command will initialize the working directory of terraform files by Downloading the required providers(aws), Initialize the provided backend(s3) and installing the modules required.

*# terraform plan* -> This command will show the changes to be applied to the infrastructure by referring main.tf

*# terraform apply ->* This command will apply the changes by creating the infrastructure provided in the main.tf

This process will create the required AWS services provided in the main.tf

1. **Containerization of nginx web application using Docker**

Refer to the following link for Dockerfile: <https://github.com/Rakesh-Marri/K8-Web-Application/blob/main/Dockerfile>

Pre-requisites:

Docker to be installed in the local before using the docker-cli commands.

Dockerfile is using for passing commands for containerization of the web application.

* *FROM nginx:alpine* -> fetched the official nginx alpine image
* *COPY nginx.conf /etc/nginx/nginx.conf* -> copy the nginx.conf file from local to the container in /etc/nginx path
* *COPY ./web-application/ /usr/share/nginx/html ->* copy the index.html file from local to container
* *EXPOSE 80 ->* Expose port 80 in the container (default port of nginx)
* *CMD ["nginx", "-g", "daemon off;"] ->* Command to run inside the container to start the nginx process

Next step is to build the dockerfile into image using the docker-cli commands

Use Docker build command to build the dockerfile

*# docker build -t nginx:1.0 .*

nginx:1.0 is the image name

Verify the built image using the command

*# docker images*

Login to AWS ECR for pushing the images

*# aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin <aws\_account\_id>.dkr.ecr.us-west-2.amazonaws.com*

aws\_account\_id is the AWS account ID

Tag the image according to the ECR

*# docker tag nginx:1.0* <aws\_account\_id>.dkr.ecr.us-west-2.amazonaws.com/nginx-web-application-ecr/nginx:1.0

Push the image to the ECR

*# docker push* <aws\_account\_id>.dkr.ecr.us-west-2.amazonaws.com/nginx-web-application-ecr/nginx:1.0

Verify the images in ECR

*# aws ecr list-images --repository-name* nginx-web-application-ecr *--region us-west-2*

1. **Deploying Nginx Web-application and monitoring with Prometheus using Kubernetes**

Refer the following link for Kubernetes yaml files: <https://github.com/Rakesh-Marri/K8-Web-Application/tree/main/K8>

Pre-requisites:

1. Kubectl to be installed locally for kubernetes commands

Update the kubeconfig file with connecting to EKS cluster

*# aws eks --region us-west-2 update-kubeconfig --name nginx-web-application-eks*

Verify the configuration by viewing nodes

*# kubectl get nodes*

Kubernetes file structure ->

nginx-deployment.yaml - contains the complete configuration of nginx pod with replicas, containers, image, volumemounts

nginx-service.yaml - contains the networking of nginx pod

nginx-ingress.yaml – contains the hostname and path for exposing the nginx web application to outside EKS cluster

nginx-pv.yaml – persistent volume for nginx mountpath

nginx-pvc.yaml – persistent volume request for nginx mountpath

nginx-sc.yaml – Storageclass required for AWS EBS

(change the directory to kubernetes files path -> cd K8)

**Deploying Nginx**

*# kubectl apply -f nginx-sc.yaml* - Storage class will be created

*# kubectl apply -f nginx-pv.yaml* - persistent volume will be created

*# kubectl apply -f nginx-pvc.yaml -* persistent volumeclaim required for nginx pod will be created

*# kubectl apply -f nginx-deployment.yaml* - nginx web application will be created

*# kubectl apply* -f nginx-service.yaml - nginx ClusterIP svc will be created

*# kubectl apply* -f nginx-ingress.yaml - nginx ingress will be created with provided DNS

**Deploying Prometheus for monitoring**

*# kubectl apply -f prometheus-configmap.yaml* - prometheus configmap will be created

*# kubectl apply* -f prometheus-deployment.yaml - prometheus monitoring will be created

*# kubectl apply* -f prometheus-service.yaml - prometheus ClusterIP svc will be created

*# kubectl apply* -f promethues-ingress.yaml - prometheus ingress will be created with path /prometheus

Verification:

*# kubectl get pods* - To view the status of nginx and prometheus pods

*# kubectl get svc* - To view the InternalIP status of ClusterIP of nginx and prometheus services

*# kubectl get ingress* – To view the ingress.

In nginx.conf file add the stub\_status for exposing it to prometheus for monitoring scrapping

*location /stub\_status {*

*stub\_status on;*

*access\_log off;*

*allow 127.0.0.1;*

*deny all;*

*}*

Troubleshooting:

*# kubectl describe pods <podname>* - view the info and events of pod

*# kubectl logs <podname>* - view the logs of pod

Access the nginx web application with <http://devops-assignment.dev>



Access the Prometheus UI with the following DNS: <http://devops-assignment.dev/prometheus>

In the Targets page we can see the nginx exporter list and metrics of it.