Deploying DeveloperIQ App to Kubernetes(Minikube) on AWS EC2

DeveloperIQ App Overview

DeveloperIQ App is designed using 2 microservices such as GitHub_Data_Collector and Data_Storage and developed using Python Flask.

Pre-requisites:

- GitHub Repository Setup:
  - Ensure your microservices code is hosted on a GitHub repository.
  - Set up GitHub Actions workflows for CI/CD (eg: github_data_collector.yml and data_store.yml).
- Docker Images:
  - Build Docker images for your microservices and push them to a container registry (e.g., Docker Hub).
  - Need to create separate Dockerfiles for every microservice. Since this app contains relatively small microservices, use one repo and create separate folders for each microservice.
- Create an AWS Account:
  - Create a separate IAM user in addition to the root user which had DynamoDB access, and EC2 access.
  - Store AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY of the IAM role on GitHub Secrets

Below is the file structure:

```
DeveloperIQ/
|--.github/
|   |--workflows/
|   |   |--python-app.yml
|   |   |--github_data_collector.yml
|   |   |--data_storage.yml
|-- GitHub_Data_Collector/
|   |-- github_data_collector.py
|   |-- requirements.txt
|   |-- Dockerfile
|-- Data_Storage/
|   |-- data_storage.py
```

```
|   |-- requirements.txt
|   |-- Dockerfile
|--K8s/
|   |--github_data_collector_deployment.yml
|   |--data_storage_deployment.yml
|-- docker-compose.yml
```

Deployment Steps:

- AWS EC2:
  - Launch AWS EC2 instances while having at least minimum requirements
    - 2 CPUs or more
    - 2GB of free memory (t3 small)
    - 20GB of free disk space
  - Store EC2 information such as `EC2_HOST, EC2_SSH_KEY, EC2_USERNAME` on GitHub Secrets
  - Connect with the EC2 instance

```
ssh -i <PEM file> ec2-user@<Public_IP>
```

  - Install minikube

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-a
md64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

  - Install Docker

```
sudo yum install docker
sudo usermod -aG docker ec2-user
sudo service docker start
```

```
minikube start --driver=docker
```

- Install kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo mv kubectl /usr/local/bin/

sudo chmod 755 /usr/local/bin/kubectl
```

- Test the kubectl installation

```
kubectl get namespaces
kubectl get pods -n kube-system
```

- Configure security groups to allow necessary traffic.
- Create a DynamoDB table
  - Store all the credentials such as `DYNAMODB_REGION` on GitHub secrets.
- Update Code on GitHub:
  - Make changes to your microservices code and push it to the GitHub repository.
- GitHub Actions CI/CD:
  - GitHub Actions will trigger automatically upon code push.
  - CI workflow should include steps for building and testing microservices. Refer to the GitHub repository here. The steps for each deployment are under **.github/workflows**
  - CD workflow should include steps for building Docker images and pushing them to the container registry.

Note: The below docker image building pushing it to Docker Hub and Deploy microservices to conduct the CI/CD pipeline. The below commands need to be run inside the workflow files separately for each microservice.

- Build and Push Docker Images to Minikube:
  - Build Docker images locally using Minikube's Docker daemon:

```
docker build -t <your-image-name>:<tag> .
```

➢ Tag the image for minikube

```
docker tag <your-image-name>:<tag
$(minikubeip):5000/<your-image-name>:<tag>
```

➢ Push the image to Minikube's registry:

```
docker push $(minikube ip):5000/<your-image-name>:<tag>
```

❖ Deploy microservices to Minikube

```
kubectl apply -f path/to/deployment.yaml
kubectl apply -f path/to/service.yaml
```

❖ Prepare Kubernetes Manifests:

➢ Create Kubernetes manifest files for your microservices (e.g., github_data_collector_deployment.yml, data_collector_deployment.yml).
➢ Note:- These _deployments.yml files can be created separately as deployment.yml and services.yml
➢ Customize these files with appropriate configurations.

To verify the deployments need to connect to the EC2 instance locally. Type the below command in cmd/PowerShell.

```
ssh -i <EC2 secret key file>.pem ec2-user@<EC2 host ID>
```

❖ Verify Deployments:

```
kubectl get pods
kubectl get svc
```

❖ Tunneling to generate External IP for each service

```
ssh -i node-js.pem -L <local random port>:<External IP>:<port given
to service> ec2-user@13.234.239.76
```

Example: ssh -i node-js.pem -L 6560:10.102.190.88:8080 ec2-user@13.234.239.76

- Generating Observability cluster
  - For this case use Minikube Dashboard as the observer.

```
minikube dashboard
```

  - Create a tunnel using ssh as above

  - Open the URL in a browser that uses Minikube dashboard.