Automated Deployments - CI/CD pipelines

The DevOps team wants to continuously deploy the charts to the cluster whenever there is a change made to the applications/charts automatically. There are a lot of tools to achieve CI/CD. We are not focusing on IAC automation here, but can be integrated to this lab if you are installing the clusters on your own. Here we would be leveraging `GitHub Actions` .

Learning Outcomes

After completing the lab, you will be able to understand

- Introduction to CI/CD
- Writing workflows and jobs using Github Actions
- Deploying to multiple environments
- Continuously deploy to target kubernetes cluster

Setting up the codebase

1. Create a repository called `helm-charts` in your GitHub account. You will need to initialize the local codebase as a git repo & add your remote repository url prior to executing any commits on it

```
git init
git remote add origin https://github.com/[your-github-user-name]/helm-charts.git
```

Directory Structure

1. Pipeline design is the most important factor to consider before implementing CI/CD as it is highly dependent on the deployment strategy and cluster envinronment which tends to vary from one project to another.

2. DevOps team would like to have a pipeline designed with the following tasks/jobs:
    a. Test the chart syntax
    b. Test the application
    c. Install the necessary softwares such as `kubectl`, `helm`
    d. Setup `kubeconfig` to point to `staging` environment cluster
    e. Deploy to the staging environment
3. Create the directory structure required for github actions inside `helm-charts` directory

```
mkdir ~/workspace/helm-charts/.github
mkdir ~/workspace/helm-charts/.github/workflows
touch ~/workspace/helm-charts/.github/workflows/pipeline.yaml
mkdir ~/workspace/helm-charts/scripts
```

Automation Scripts

1. Create the script files for installing `kubect`l cli, installing `helm`, and deploying the chart to the cluster.

```
touch ~/workspace/helm-charts/scripts/install-kubectl.sh
touch ~/workspace/helm-charts/scripts/install-helm.sh
touch ~/workspace/helm-charts/scripts/deploy.sh
```

2. Write a script file in bash to install kubectl

`helm-charts/scripts/install-kubectl.sh`

```
#!/bin/sh
set -e

sudo apt-get update
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s ht
tps://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/am
```

```
d64/kubectl"
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
kubectl version --client
kubectl config get-contexts
```

3. Write a script file in bash to install helm

`helm-charts/scripts/install-helm.sh`

```
#!/bin/bash
set -e

curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -
sudo apt-get install apt-transport-https --yes
echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/ap
t/sources.list.d/helm-stable-debian.list
sudo apt-get update
sudo apt-get install helm
```

4. Write a script file in bash to deploy helm charts to the k8s cluster

`helm-charts/scripts/deploy.sh`

```
#!/bin/bash
set -e

echo Namespace = "$1"
echo Releasename = "$2"
echo "----------------------Installing pages--------------------------------"


NAMESPACE="$1"
RELEASE_NAME="$2"
```

```
kubectl get ns "$NAMESPACE" &> /dev/null || kubectl create ns "$NAMESPACE"

helm upgrade --install "$RELEASE_NAME" pages -n "$NAMESPACE" --debug
helm list
kubectl config current-context
sleep 30s

kubectl get pods -n "$NAMESPACE"


echo "----------------------Completed Installation of  pages--------------------
--------------"
```

Set-up Github Actions

1. Create the following secrets in github

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_REGION
STAGING_NAMESPACE
RELEASE_NAME
```

2. Configure the pipeline

`helm-charts/.github/workflows/pipeline.yaml`

```yaml
name: Pages Pipeline

on:
  push:
    branches: [master]

jobs:
  deploy-to-staging:
```

```
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ secrets.AWS_REGION }}
      - name: Configure EKS
        run: |
          aws eks --region ap-south-1 update-kubeconfig --name dees-cloud

      - name: Install Kubectl
        uses: actions/checkout@v2
      - name: Check kubectl
        run: |
          ls
          bash ./scripts/install-kubectl.sh
      - name: Install Helm3
        run: |
          bash ./scripts/install-helm.sh
      - name: Deploy to staging
        run: |
          bash ./scripts/deploy.sh ${{ secrets.STAGING_NAMESPACE }} ${{ secrets.
RELEASE_NAME }}
```

Code Commit

1. Commit the changes made to the workspace and push to github. The github webhooks should identify the changes and start running the pipeline.

```
git add .
git commit -m "Pipeline 1.0"
```

```
git push -u origin master
```

2. Test the pages application by performing CRUD operations using curl/postman. Refer Pages Curl Guide for testing.