Automated Testing in the pipeline

The DevOps team wants to improve the quality & resiliency of deployments, by preventing unforeseen errors and defects that creep into the production environment. The best way to accomplish this is by detecting & preventing such scenarios in the pipeline itself. Basically, there are 2 kinds of defects that need to be controlled.

a. Defects in the chart

b. Defects in the applications comprising the deployment architecture

Learning Outcomes

After completing the lab, you will be able to understand

- Checking for Chart errors & Validation
- Helm Chart Testing
- API/Contract Test
- Testing on multiple environments

Create the Test-Suite

1. Create the yaml manifest files for controller contract tests and database test

```
touch ~/workspace/helm-charts/pages/charts/api/templates/test-api-contracts.yaml
touch ~/workspace/helm-charts/pages/charts/api/templates/test-dbconnect-positiv
e.yaml
touch ~/workspace/helm-charts/pages/charts/api/templates/test-message-get.yaml
```

2. Write the test cases

`helm-charts/charts/api/templates/test-api-contracts.yaml`

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: contracts-pages-api
    helm: test
  name: contract-test
  annotations:
    "helm.sh/hook": test-success
spec:
  containers:
    - name: postman-test
      image: dellcloud/newman:latest #derived from postman/newman
      command: ["newman","run", "https://cloud-native-labs.s3.ap-south-1.amazona
ws.com/J21/labguide/pages-testsuite.json" ]
      args:
        - --env-var
        - BASE_URL={{ .Release.Name }}-{{ .Chart.Name }}.{{ .Release.Namespace
 }}.svc.cluster.local:8080
  restartPolicy: Never
```

helm-charts/charts/api/templates/test-dbconnect-positive.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: dbconnect-pages-api
    helm: test
  name: pages-test-dbconnect-passtest
  annotations:
    "helm.sh/hook": test
spec:
  containers:
```

```
    - image: mysql:8.0
      name: pages-test-pass
      imagePullPolicy: Always
      command: ["/bin/bash"]
      args: ["-c", "while true; do (mysql -u root -h pages-mysql pages -ppasswor
d  -e 'show tables;' > logs.txt);count=$(cat logs.txt | grep '[p]ages' | wc -l);
if [[ $count -gt 0 ]]; then echo 'Found'; exit 0;else echo 'Not Found';exit 1;f
i;done"]
  restartPolicy: Never
```

helm-charts/charts/api/templates/test-message-get.yaml

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: message-pages-api
    helm: test
  name: {{ .Chart.Name}}-test-getmessage
  annotations:
    "helm.sh/hook": test-success
spec:
  containers:
    - image: curlimages/curl
      name: {{ .Chart.Name }}-test
      imagePullPolicy: {{ .Values.imagePullPolicy }}
      command: ["/bin/sh","-c"]
      args:
        - curl  http://{{ .Release.Name }}-{{ .Chart.Name }}.{{ .Release.Namespa
ce }}.svc.cluster.local:8080
  restartPolicy: Never
```

Testing in the pipeline

1. Create the script file for testing helm charts in the pipeline.

```
touch ~/workspace/helm-charts/scripts/helm-test.sh
```

2. Write a bash script for testing using helm

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

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

echo Namespace = "$1"
NAMESPACE=$1
RELEASE_NAME="$2"

kubectl config get-contexts
kubectl create ns "$NAMESPACE"
kubectl config set-context --current --namespace "$NAMESPACE"
helm lint pages
helm template pages


echo "------------------------Start time is--------  $(date +%Y-%m-%dT%H%M%S%z)"

helm upgrade --install "$RELEASE_NAME" pages --debug

echo "------------------------End time is--------  $(date +%Y-%m-%dT%H%M%S%z)"

echo '---------------------Started testing--------------'
sleep 60s
kubectl get po -n "$NAMESPACE" --show-labels
kubectl get svc -n "$NAMESPACE" -o wide
helm test "$RELEASE_NAME" --logs
echo '---------------------Completed testing------------'
```

```
helm uninstall "$RELEASE_NAME"
kubectl delete ns "$NAMESPACE"
```

3. Create a new job to deploy to a kind cluster (test cluster) where the test cases will be run prior to the staging deployment.

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

```yaml
name: Pages Pipeline

on:
  push:
    branches: [master]

jobs:
  deploy-to-kind:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Chart-testing (lint)
        id: lint
        uses: HexF/chart-testing-action@v3.0.0
        with:
          command: lint
      - name: Create kind cluster
        uses: helm/kind-action@v1.2.0
        with:
          install_local_path_provisioner: true
      - name: Install Kubectl
        run: |
          ls
          bash ./scripts/install-kubectl.sh

      - name: Helm Testing
        run: |
```

```
          bash ./scripts/helm-test.sh ${{ secrets.STAGING_NAMESPACE }} ${{ secre
ts.RELEASE_NAME }}

  deploy-to-staging:
    runs-on: ubuntu-latest
    needs:  deploy-to-kind
    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 "Adding Test Suite 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.