Canary Deployments with Kubernetes and Jenkins

Introduction

In this lab, we'll be setting up a train-schedule application. However, some recent deployments introduced bugs that did not become visible until after the changes were in the hands of real users. To mitigate these kinds of issues, we'll implement a canary deployment as part of the larger deployment pipeline.

The application runs on Kubernetes cluster, and we'll use a Jenkins pipeline to perform CI and deployments. We will need to add logic to this existing pipeline in order to implement the canary deployment.

We will need to create a fork of the sample course code here: **https://github.com/ACloudGuru-Resources/cicd-pipeline-train-schedule-canary**

Solution

1. Create a free **GitHub** account, or log in to your GitHub account.
2. Create a free **DockerHub** account, or log in to your account.
3. We will also need to log in to all three of our servers using `ssh` and the provided credentials:

   ```
   ssh cloud_user@<PUBLIC_IP>
   ```

Set up the project and Jenkins

To do this, you will need to perform the following steps:

1. First, we need to go to this code repo: **https://github.com/ACloudGuru-Resources/cicd-pipeline-train-schedule-canary**.
2. Click the **Fork** button on this page.
3. After creating your personal fork, select the `Jenkinsfile`.
4. Click the **Edit** button for our file, and change the user name `willbla` to your personal Docker Hub username.

Get Into our Jenkins Server

1. In your web browser, enter the public IP provided for our Jenkins server, followed by `:8080` .

Set up our Jenkins Credentials

Perform the following in our web browser:

1. Select **Manage Jenkins** > **Manage Credentials** from the menu.
2. From the single credential present, click on **global**, then **Add Credentials**.
3. Enter your personal GitHub user name in the *Username* section.
4. Go to our *GitHub* browser and select your account, then from the dropdown, choose **Settings**.
5. Next, choose **Developer settings** followed by **Personal access tokens**.
6. Select **Tokens (classic)** > **Generate new token (classic)** and do the following:
    - Set the *Token description* of *jenkins*.
    - Select the checkbox for *admin:repo_hook*.
7. Click **Generate token**.
8. Copy the token, and back in Jenkins, paste the token into the *Password* field.
9. Set the *ID* to `github_key` and the *Description* to `GitHub Key` .
10. Click **OK**.

Add Docker Hub Credentials

1. Click **Add Credentials**.
2. Provide your Docker Hub username and password in the corresponding fields.
3. Set the *ID* to `docker_hub_login` and the *Description* to `Docker Hub` .
4. Click **OK**.

Add Kubernetes Credentials

1. Click **Add Credentials**.
2. Set the *Kind* to `Kubernetes configuration (kubeconfig)`
3. Set the *ID* to `kubeconfig` and the description to `Kubeconfig` .
4. Click into our Kubernetes command line.

5. Run the following:
   `cat ~/.kube/config`
6. Copy the contents of the file that appears.
7. Back in our Jenkins browser, paste what was copied into the *Enter directly* > *Content* section.
8. Click **OK**.

Set Up the project

To set up the project, follow these steps:

1. Go back to the main page by selecting the **Jenkins** icon at the top left of the website.
2. Click **New Item**.
3. Give the item the name `train-schedule`.
4. Select **Multibranch Pipeline** and click **OK**.
5. Under *Branch Sources*, change the *Add source* to `GitHub`.
6. Set up the page as follows:

- *Credentials*: Option that ends in `(GitHub Key)`
- *Repository HTTPS URL*: ***https://github.com/Your_GitHub_Username/cicd-pipeline-train-schedule-canary***, then click *Validate*.

1. Select **Save**.

Review the Builds

Complete the following to review what we've done up to this point:

1. Should an item called *Build Queue* appear above your *Build Executor Status*, select the **cancel** button for it (the red box with an X in it).
2. Click on our Master build and wait for it to process.
3. Once the master build gets to the *DeployToProduction* stage, hover over the *DeployToProduction*, and select **Proceed**.
4. Click on our Kubernetes server, and enter the following to review our deployment:
   `kubectl get pods -w`
5. To review our application, from our lab's credentials page, copy the Kubernetes' Public IP and paste it into a new browser window followed by `:8080`.

Add a Canary Stage to the Pipeline

To complete this, you will need to do the following:

1. Go back to GitHub and go to the fork that we made.
2. Copy the **train-schedule-kube-canary.yml code**.
3. Click **Add file** > **Create new file**.
4. Name the file `train-schedule-kube-canary.yml` .
5. Paste in the **train-schedule-kube-canary.yml code**.
6. Click **Commit new file**.
7. Select the `Jenkinsfile` , which can be found **here**.
8. Add a new stage after the `Push Docker Image` stage and before `DeployToProduction` :

```
stage('CanaryDeploy') {
  when {
    branch 'master'
  }
  environment {
    CANARY_REPLICAS = 1
  }
  steps {
    kubernetesDeploy(
      kubeconfigId: 'kubeconfig',
      configs: 'train-schedule-kube-canary.yml',
      enableConfigSubstitution: true
      )
  }
}
```

9. Modify the `DeployToProduction` section adding the following between the `when { ... }` and original `kubernetesDeploy( ... )` sections:

```
environment {
  CANARY_REPLICAS = 0
}
steps {
  input 'Deploy to Production?'
  milestone(1)
  kubernetesDeploy(
    kubeconfigId: 'kubeconfig',
    configs: 'train-schedule-kube-canary.yml',
    enableConfigSubstitution: true
```

```
)
```

10. Select **Commit changes**

Run a Successful Deployment

Run our set up to make sure everything is running correctly:

1. Go to our Kubernetes server and review the following:

   ```
   kubectl get pods -w
   ```

2. Back on Jenkins, click **Build Now** to build our updated information.
3. Back on our Kubernetes node, we see our new pod information appears with our `canary` section.
4. Check on our website that everything is working by putting in the Public IP address for our Kubernetes with port `:8081` at the end.
5. Once we know the site is working, go back to Jenkins, hover over *DeployToProduction*, and click **Proceed**.
6. On our Kubernetes server, we see the new node running.
7. Enter **Ctrl** + **C** to exit the node.
8. Run `kubectl get pods` and note that everything is running.

Conclusion

Congratulations! You've completed the lab!