Demo Project: CD - Deploy to EKS cluster from Jenkins Pipeline

This guide covers installing required tools on the Jenkins server, creating a kubeconfig file, adding AWS credentials to Jenkins, and updating the Jenkinsfile to deploy to an EKS cluster.

Note: This project depends on the previously created EKS cluster (e.g., using eksctl).

Step 1: Install kubectl on the Jenkins Server

1. **SSH into the Jenkins Server:** `ssh root@<jenkins ip>`
2. **Identify the Jenkins Docker Container:** `docker ps`
3. **Enter the Jenkins Container as Root:** `docker exec -u 0 -it <container id> bash`
4. **Install kubectl:**
    - `curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl;`
    - `chmod +x ./kubectl;`
    - `mv ./kubectl /usr/local/bin/kubectl`

5. **Verify kubectl Installation:** `kubectl version`

Step 2: Install AWS IAM Authenticator on the Jenkins Server

1. **Download and Install aws-iam-authenticator:**
    - `curl -Lo aws-iam-authenticator https://github.com/kubernetes-sigs/aws-iam-authenticator/releases/download/v0.6.11/aws-iam-authenticator_0.6.11_linux_amd64`
    - `chmod +x ./aws-iam-authenticator`
    - `mv ./aws-iam-authenticator /usr/local/bin`
2. **Verify Installation:** `aws-iam-authenticator help`

Step 3: Create kubeconfig file for EKS

Since the Jenkins container is lightweight and lacks an editor, create the kubeconfig file on the host machine.

1. **Exit the Jenkins Container** (if inside).
    - Ensure you are on the host machine.
2. **Obtain Cluster Information:**
    - Gather the following details from your EKS cluster:
        - **K8s Cluster Name**
        - **Server Endpoint**
        - **Certificate-authority-data** (use `cat ~/.kube/config` on your local machine to retrieve this from an existing kubeconfig file)
3. **Create a Kubeconfig File:**
    - Open an editor on your host machine (e.g., using `vim config`) and insert the following content, modifying the placeholders accordingly:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    certificate-authority-data: <certificate-data>
    server: <endpoint-url>
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
users:
- name: aws
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      command: /usr/local/bin/aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - <cluster-name>
```

4. **Save the File**

Step 4: Copy the kubeconfig File to the Jenkins Container

1. **Enter the Jenkins Docker Container:** `docker exec -it <container-id> bash`
2. **Create a .kube Directory in the Jenkins Home:**
    - `cd ~`

- `mkdir .kube`
- `exit`

3. **Copy the kubeconfig File** from the host to the Jenkins container:

   `docker cp config <container-id>:/var/jenkins_home/.kube/`

   (Replace `<container-id>` with your actual Jenkins container ID.)

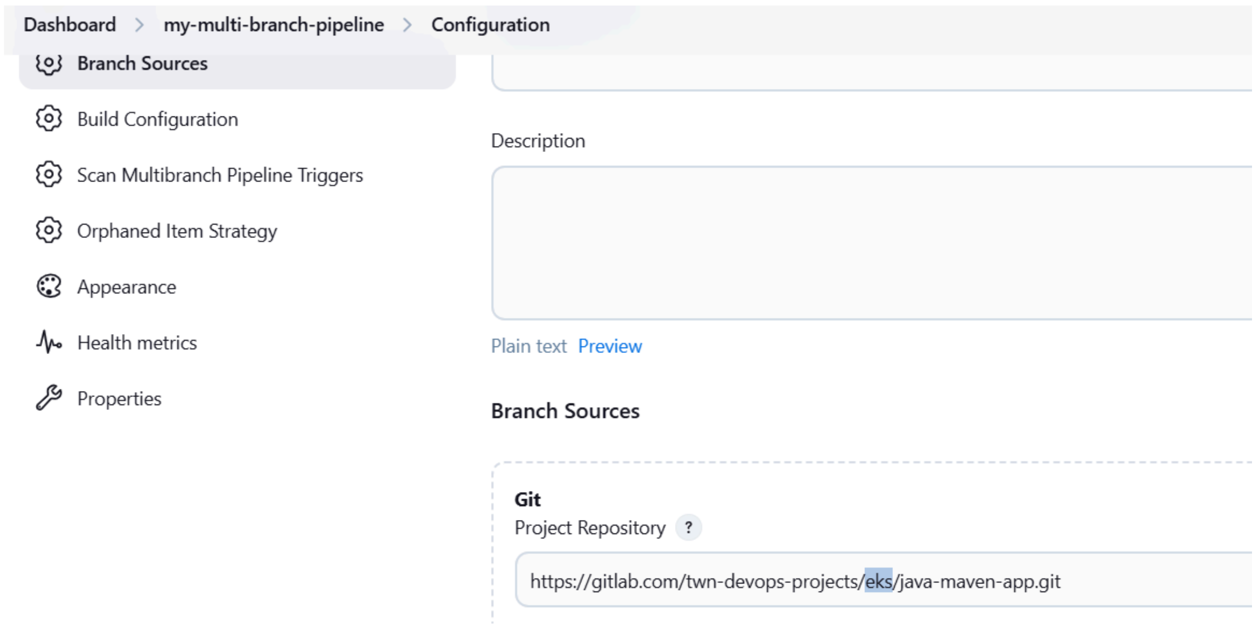
4. **Verify the File Exists in the Container**:

   - `docker exec -it <container-id> bash`
   - `ls .kube`

5. **Exit the container:** `exit`

Step 5: Add AWS Credentials in Jenkins

1. **Fork the Jenkins Project:**
   - Fork the "java-maven-app" repository into an "eks" repository.
2. **Create a Branch for Deployment:**
   - From the main branch, create a branch named `deploy-on-k8s`.
3. **Update Jenkins Multibranch Pipeline Configuration:**
   - In the Jenkins UI, update the source project in your multibranch pipeline configuration and click **Save**.


Dashboard > my-multi-branch-pipeline > Configuration

Branch Sources

Git

Project Repository

https://gitlab.com/twn-devops-projects/eks/java-maven-app.git

4. **Add AWS Credentials to Jenkins:**
   - Navigate to Manage Jenkins → Global Credentials
   - Add **Credential 1**:
     - Kind: Secret text
     - ID: `jenkins_aws_access_key_id`
     - Secret: Paste your AWS Access Key ID (from `cat ~/.aws/credentials`).
   - Add **Credential 2**:
     - Kind: Secret text
     - ID: `jenkins_aws_secret_access_key`
     - Secret: Paste your AWS Secret Access Key (from `cat ~/.aws/credentials`).

Step 6: Configure Jenkinsfile to deploy to EKS

1. **Switch to the Branch** `deploy-on-k8s` .

2. **Update Your Jenkinsfile** to include a deployment stage that uses `kubectl`. For example:

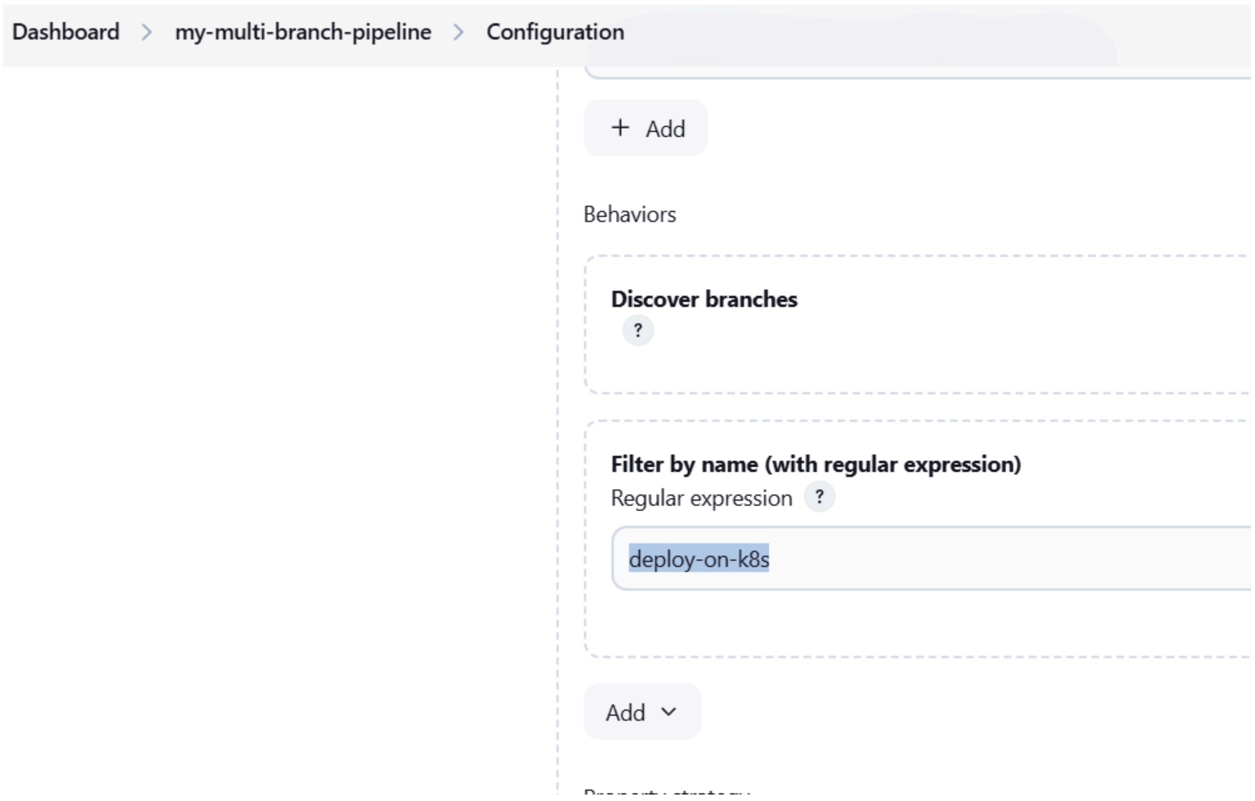
```
#!/usr/bin/env groovy

pipeline {
    agent any
    stages {
        stage('build app') {
            steps {
               script {
                   echo "building the application..."
               }
            }
        }
        stage('build image') {
            steps {
                script {
                    echo "building the docker image..."
                }
            }
        }
        stage('deploy') {
            environment {
                AWS_ACCESS_KEY_ID = credentials('jenkins_aws_access_key_id')
                AWS_SECRET_ACCESS_KEY = credentials('jenkins-aws_secret_acce
ss_key')
            }
            steps {
                script {
                   echo 'deploying docker image...'
                   sh 'kubectl create deployment nginx-deployment --image=nginx'
                }
            }
        }
      }
    }
```

3. **Commit and Push Your Changes:**
   - Commit the updated Jenkinsfile to the `deploy-on-k8s` branch and push it to the repository.

Step 7: Execute the Jenkins Pipeline

1. **Go to Jenkins UI**
2. **Update Pipeline Configuration:**
   - Ensure that only the `deploy-on-k8s` branch is triggered.



Filter by name (with regular expression)
Regular expression: deploy-on-k8s

3. **Trigger the Multi Branch Pipeline**.

4. **Monitor the Pipeline Console Output** to ensure the build and deployment stages complete successfully.
5. **Verify Deployment in EKS**: `kubectl get pod`

Step 8: Clean up

When finished, you can clean up your resources:

1. **Delete the Deployment** (if needed): `kubectl delete deployment <deployment-name>`
2. **Optionally, Delete the EKS Cluster**: `eksctl delete cluster --name <your-cluster-name> --region <your-region>`
