基于kubernetes构建动态Jenkins-slave

安装配置 Master

1. 创建pvc- 基于NFS的存储类

```yaml
---

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jenkins-rbd-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 10Gi
  storageClassName: managed-nfs-storage
```

2. 创建RBAC

   需要先创建namespace,这里不写在yaml里，怕误操作。 `kubectl create ns devops`

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: devops

---

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: jenkins
rules:
  - apiGroups: ["extensions", "apps"]
    resources: ["deployments"]
    verbs: ["create", "delete", "get", "list", "watch", "patch", "update"]
  - apiGroups: [""]
    resources: ["services"]
    verbs: ["create", "delete", "get", "list", "watch", "patch", "update"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create","delete","get","list","patch","update","watch"]
  - apiGroups: [""]
    resources: ["pods/exec"]
    verbs: ["create","delete","get","list","patch","update","watch"]
  - apiGroups: [""]
    resources: ["pods/log"]
    verbs: ["get","list","watch"]
  - apiGroups: [""]
```

```
    resources: ["secrets"]
    verbs: ["get","list","watch"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins
  namespace: devops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: jenkins
subjects:
  - kind: ServiceAccount
    name: jenkins
    namespace: devops
```

3. 创建Jenkins master的 `jenkins-statefulset.yaml`

   这里我采用的是deployment，因为我本地没有存储集群，所以我这里使用的是hostpath，也添加nodeSelector，只能调度到此节点，避免数据丢失。

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  namespace: devops
  labels:
    app: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      name: jenkins
      labels:
        app: jenkins
    spec:
      terminationGracePeriodSeconds: 10
      serviceAccountName: jenkins
      nodeSelector:
        jenkins: home
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          name: web
          protocol: TCP
        - containerPort: 50000
          name: agent
          protocol: TCP
        resources:
          limits:
            cpu: 2000m
            memory: 4Gi
          requests:
            cpu: 500m
            memory: 512Mi
        volumeMounts:
        - name: jenkinshome
          subPath: jenkins
          mountPath: /var/jenkins_home
        env:
        - name: LIMITS_MEMORY
          valueFrom:
            resourceFieldRef:
              resource: limits.memory
              divisor: 1Mi
        - name: JAVA_OPTS
          value: -Xmx$(LIMITS_MEMORY)m -XshowSettings:vm -
Dhudson.slaves.NodeProvisioner.initialDelay=0 -
Dhudson.slaves.NodeProvisioner.MARGIN=50 -
Dhudson.slaves.NodeProvisioner.MARGIN0=0.85 -Duser.timezone=Asia/Shanghai
      securityContext:
        fsGroup: 1000
      volumes:
      - name: jenkinshome
        hostPath:
          path: /jenkins_home

---

kind: Service
apiVersion: v1
metadata:
  labels:
    app: jenkins
  name: jenkins
  namespace: devops
spec:
  type: NodePort
  ports:
    - name: web
      port: 8080
      targetPort: 8080
      nodePort: 30086
    - name: agent
      port: 50000
      targetPort: 50000
      nodePort: 30087
  selector:
    app: jenkins
```

如果使用了存储类，可以参考如下配置文件。

```
---
apiVersion: apps/v1
kind: Deployment
```

```yaml
metadata:
  name: jenkins
  namespace: devops
  labels:
    app: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      name: jenkins
      labels:
        app: jenkins
    spec:
      terminationGracePeriodSeconds: 10
      serviceAccountName: jenkins
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          name: web
          protocol: TCP
        - containerPort: 50000
          name: agent
          protocol: TCP
        resources:
          limits:
            cpu: 2000m
            memory: 4Gi
          requests:
            cpu: 500m
            memory: 512Mi
        volumeMounts:
        - name: jenkinshome
          subPath: jenkins
          mountPath: /var/jenkins_home
        env:
        - name: LIMITS_MEMORY
          valueFrom:
            resourceFieldRef:
              resource: limits.memory
              divisor: 1Mi
        - name: JAVA_OPTS
          value: -Xmx$(LIMITS_MEMORY)m -XshowSettings:vm -
Dhudson.slaves.NodeProvisioner.initialDelay=0 -
Dhudson.slaves.NodeProvisioner.MARGIN=50 -
Dhudson.slaves.NodeProvisioner.MARGIN0=0.85 -Duser.timezone=Asia/Shanghai
      securityContext:
        fsGroup: 1000
      volumes:
      - name: jenkinshome
        persistentVolumeClaim:
          claimName: jenkins-rbd-pvc
```

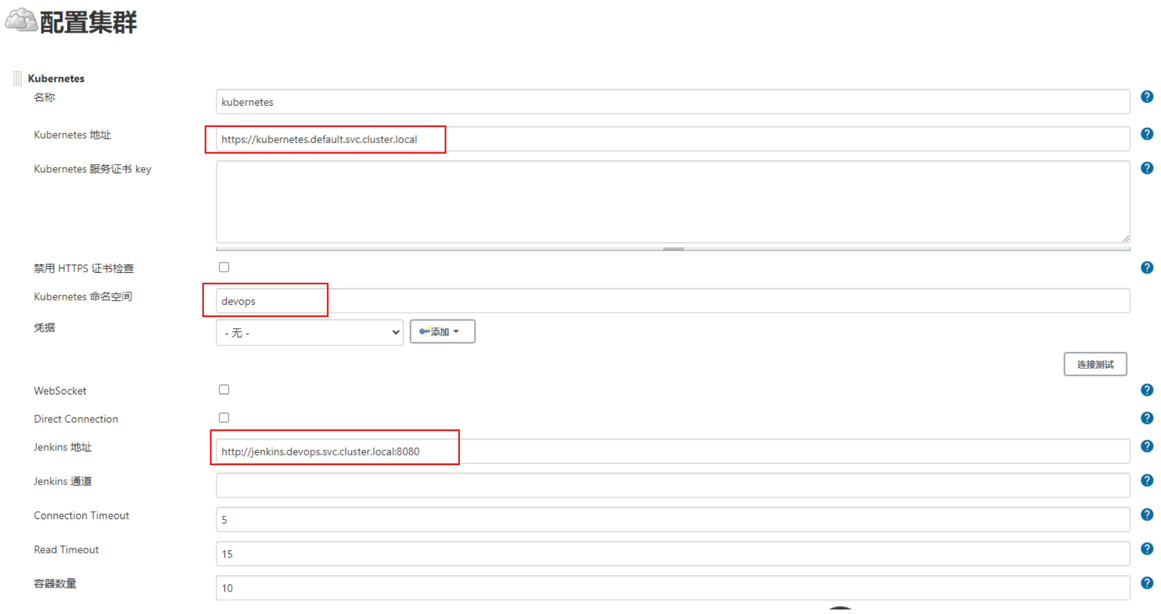
```yaml
---

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jenkins-rbd-pvc
  namespace: devops
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 10Gi
  storageClassName: managed-nfs-storage
---

kind: Service
apiVersion: v1
metadata:
  labels:
    app: jenkins
  name: jenkins
  namespace: devops
spec:
  type: NodePort
  ports:
    - name: web
      port: 8080
      targetPort: 8080
      nodePort: 30086
    - name: agent
      port: 50000
      targetPort: 50000
      nodePort: 30087
  selector:
    app: jenkins
```

4. 此时可以访问k8s节点的nodePort端口，进行配置和验证。

配置Jenkins cloud

需要安装kubernetes相关插件。以及docker和pipeline相关的插件，可自行搜索。

在新版本的Jenkins当中，增加了Manage Nodes and Clouds，在此处配置我们的k8s集群。配置如下图所示：


配置集群

Kubernetes

名称: kubernetes

Kubernetes 地址: https://kubernetes.default.svc.cluster.local

Kubernetes 服务证书 key:

禁用 HTTPS 证书检查: □

Kubernetes 命名空间: devops

凭据: - 无 -

WebSocket: □

Direct Connection: □

Jenkins 地址: http://jenkins.devops.svc.cluster.local:8080

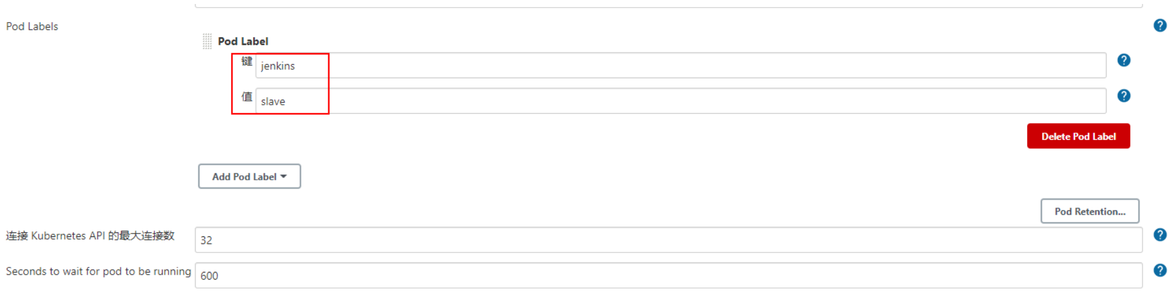
Jenkins 通道:

Connection Timeout: 5

Read Timeout: 15

容器数量: 10

添加Pod Labels


Pod Labels

Pod Label

键: jenkins

值: slave

连接 Kubernetes API 的最大连接数: 32

Seconds to wait for pod to be running: 600

Jenkins在kubernetes集群内部的话，是不需要进行证书配置的。

编写测试的流水线

```
def label = "slave-${UUID.randomUUID().toString()}"

podTemplate(label: label, serviceAccount: 'jenkins', containers: [
  containerTemplate(name: 'maven', image: 'maven:3.6-alpine', command: 'cat',
ttyEnabled: true),
  containerTemplate(name: 'docker', image: 'docker', command: 'cat', ttyEnabled:
true),
  containerTemplate(name: 'kubectl', image: 'cnych/kubectl', command: 'cat',
ttyEnabled: true),
  containerTemplate(name: 'helm', image: 'cnych/helm', command: 'cat',
ttyEnabled: true)
], volumes: [
  hostPathVolume(mountPath: '/root/.m2', hostPath: '/var/run/m2'),
  hostPathVolume(mountPath: '/home/jenkins/.kube', hostPath: '/root/.kube'),
  hostPathVolume(mountPath: '/var/run/docker.sock', hostPath:
'/var/run/docker.sock')
]) {
  node(label) {


    stage('单元测试') {
      echo "测试阶段"
    }
    stage('代码编译打包') {
      container('maven') {
        echo "打码编译打包阶段"
      }
    }
    stage('构建 Docker 镜像') {
      container('docker') {
        echo "构建 Docker 镜像阶段"
      }
    }
    stage('运行 Kubectl') {
      container('kubectl') {
        echo "查看 K8S 集群 Pod 列表"
        sh "whoami"
        sh "echo $HOME"
        sh "ls -l $HOME/.kube/config"
        sh "sed -i 's/apiserver.k8s.local:8443/192.168.50.101:6443/g'
$HOME/.kube/config"
        sh "cat $HOME/.kube/config"
        sh "ls -l $HOME/.kube/"
        sh "kubectl get pods"
      }
    }
    stage('运行 Helm') {
      container('helm') {
        echo "查看 Helm Release 列表"
        sh "helm list"
      }
    }
  }
}
```

PS: 这里挂载宿主机的kubeconfig配置文件，可能需要所有的node节点 进行相关配置，因为slave会动态的创建在随机节点中。如果自行命令报错。也可检查Jenkins RBAC授权的权限是否足够。