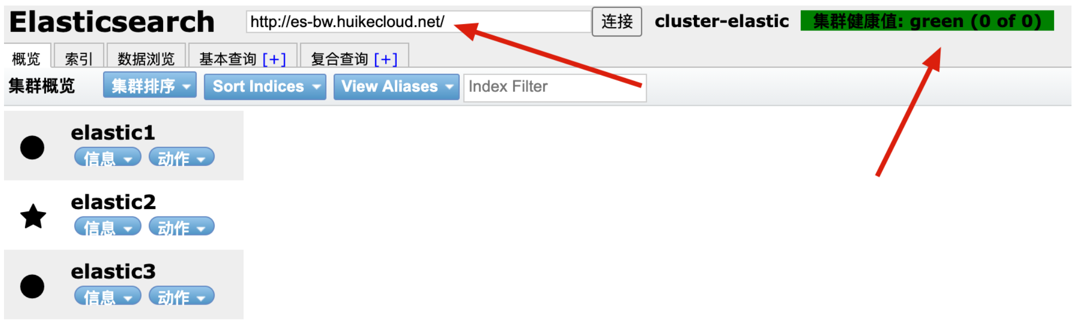
k8s配置

- ElasticSearch
    - http://es-bw.huikecloud.net/



Harbor安装

- harbor是用于存储docker镜像使用
- 配置k8s和docker源
    - 下载阿里云的docker-ce
        - cd /etc/yum.repos.d
        - wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
- 缺包问题，没有略过
    - container-selinux >= 2:2.74
    - wget http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm
    - rpm -ivh container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm
- docker 安装
    - yum install docker-ce
- 启动加入启动项
    - systemctl start docker
    - systemctl enable docker
- 解压
    - tar zxvf harbor-offline-installer-v1.7.5.tgz
- 修改配置文件
    - vim harbor.cfg

```
hostname = 10.0.54.8
harbor_admin_password = kuick123456
```

- 准备配置
    - ./prepare
- 安装docker-compoas

  - yum install -y epel-release
  - yum install docker-compose
- 导入镜像并启动
  - ./install.sh
  - docker-compose ps
- 进行访问
  - http://192.168.1.24/harbor/sign-in
- 镜像删除
  - 使用python脚本连接harbor对镜像进行删除，在停止harbor对进行镜像，使用gc进行删除

NFS

- Pod存储，使用简单的网络的存储，就算你pod挂掉也不受影响
- 新增一台主机，安装nfs-service
  - 使用yum方式部署
    - yum install nfs-utils -y
  - 创建共享目录
    - mkdir /data/volumes -p
  - 编辑共享配置文件
    - vim /etc/exports

      ```
      /data/volumes 10.0.54.0/16(rw,no_root_squash)
      ```

    - systemctl start nfs
  - node节点如果没有nfs命令进行安装
    - yum install nfs-utils -y
  - vim /etc/hosts

    ```
    10.0.54.22 nfs
    ```

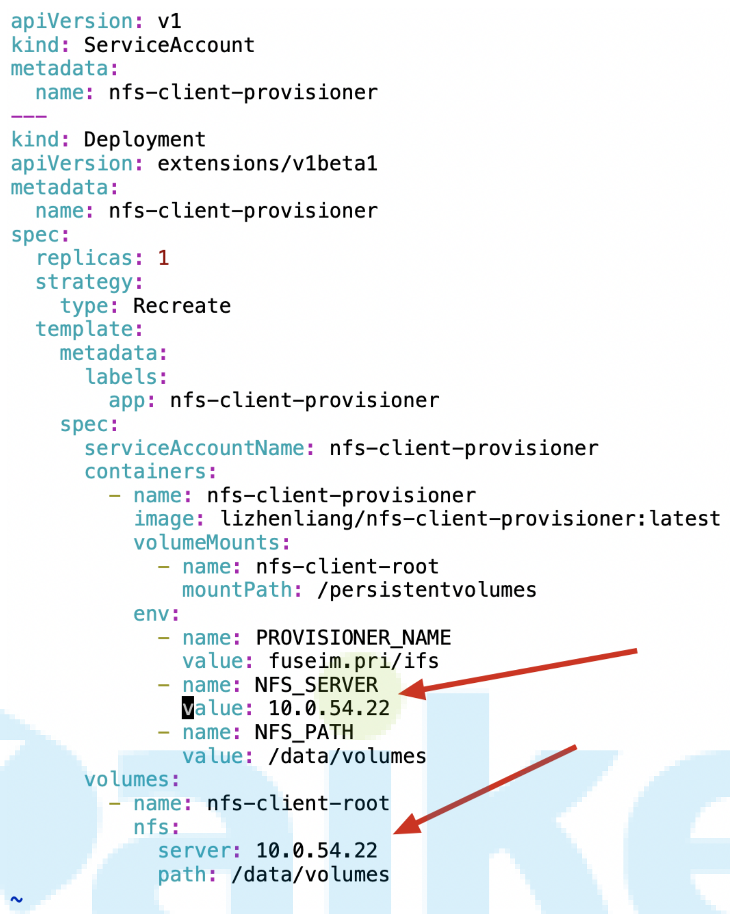
  - node上测试是否挂载成功
    - mount -t nfs nfs:/data/volumes /mnt
    - umount /mnt 卸载

Jenkins安装

- 发布你的项目
- **Jenkins-部署**
  - 代码地址
  - https://github.com/jenkinsci/kubernetes-plugin/tree/master/src/main/kubernetes
  - jenkins文件下载

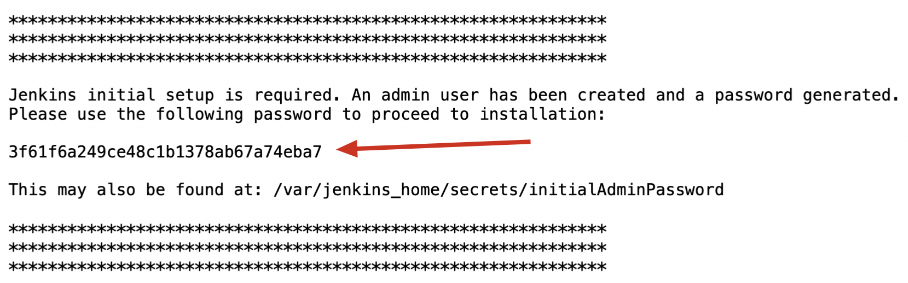
- 需要安装文件的找助教

- 修改nfs位置
  - vim nfs-client/deployment.yaml



```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-client-provisioner
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: nfs-client-provisioner
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image: lizhenliang/nfs-client-provisioner:latest
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: fuseim.pri/ifs
            - name: NFS_SERVER
              value: 10.0.54.22
            - name: NFS_PATH
              value: /data/volumes
      volumes:
        - name: nfs-client-root
          nfs:
            server: 10.0.54.22
            path: /data/volumes
~
```

- 配置jenkins的pv供给
  - /root/jenkins/nfs-client
  - kubectl create -f .
  - kubectl get pod
- jenkins安装
  - /root/jenkins/jenkins
  - kubectl create -f rbac.yml
  - kubectl create -f service.yml
  - kubectl create -f statefulset.yml   安装前先配置好jenkins磁盘大小
  - kubectl get pod
- 查看jenkins密码
  - kubectl logs jenkins-0

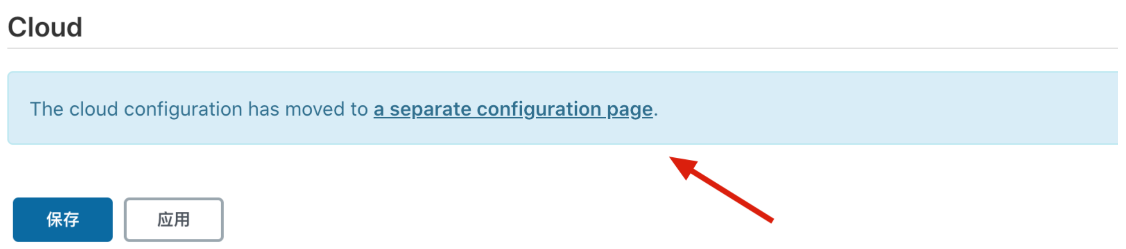


```
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

3f61f6a249ce48c1b1378ab67a74eba7

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

  - 访问，端口30006映射
    - http://jenkins-bw.huikecloud.net/

Jenkins连接k8s配置

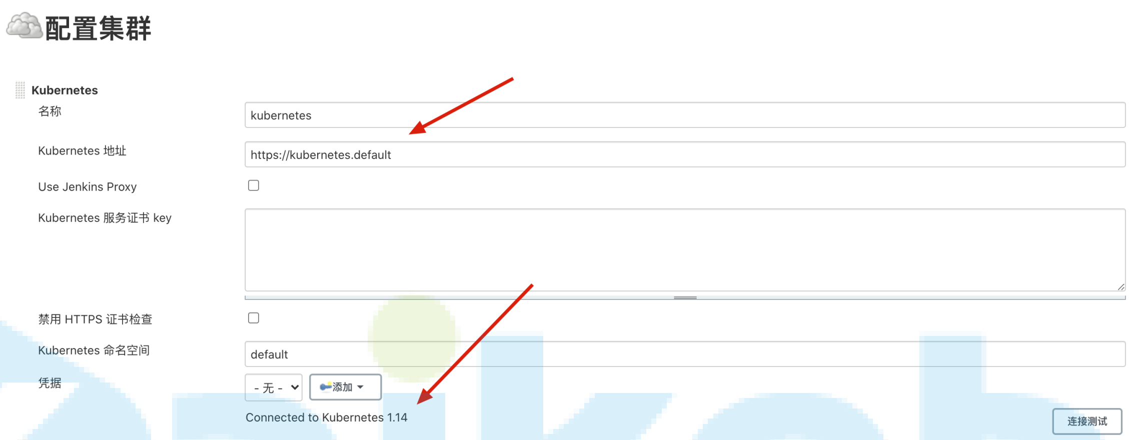
- 安装kubernetes插件
- 配置拉到最底下



Cloud

The cloud configuration has moved to a separate configuration page.

- 连接k8s配置



配置集群

Kubernetes

名称: kubernetes

Kubernetes 地址: https://kubernetes.default

Use Jenkins Proxy: ☐

Kubernetes 服务证书 key:

禁用 HTTPS 证书检查: ☐

Kubernetes 命名空间: default

凭据: - 无 -

Connected to Kubernetes 1.14

Harbor连接

- harbor端配置，ip设置为harbor地址
  - vim /etc/docker/daemon.json

```
{
  "registry-mirrors": ["https://fvn7v6mj.mirror.aliyuncs.com"],
  "insecure-registries": ["10.0.54.8"]
}
```

- 访问端，ip设置harbor本机地址
  - find / -name docker.service -type f
  - vim /usr/lib/systemd/system/docker.service

```
#ExecStart=/usr/bin/dockerd -H unix://
ExecStart=/usr/bin/dockerd --insecure-registry=10.0.54.8
```

- 服务重启
  - systemctl daemon-reload
- 访问端测试
  - docker login 10.0.54.8

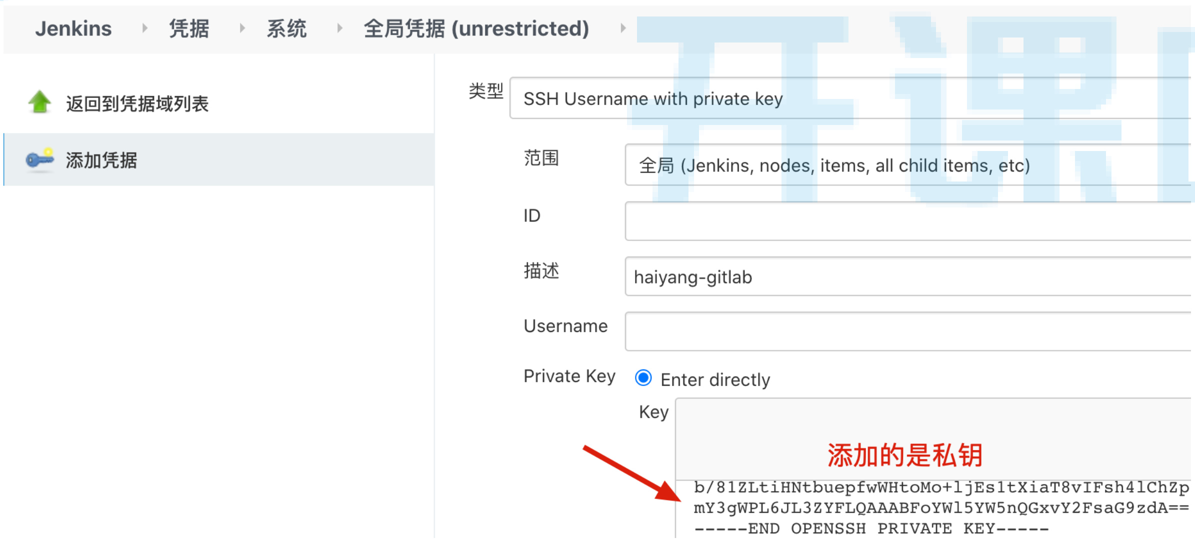
制作Jenkins镜像

- 进入工作目录
  - cd /root/mainfests/jenkins/jenkins-slave
  - vim Dockerfile

```
FROM centos:7

RUN yum install -y java-1.8.0-openjdk maven curl git libtool-ltdl-devel
&& \
    yum clean all && \
    rm -rf /var/cache/yum/* && \
    mkdir -p /usr/share/jenkins

COPY slave.jar /usr/share/jenkins/slave.jar
COPY jenkins-slave /usr/bin/jenkins-slave
COPY settings.xml /etc/maven/settings.xml
RUN chmod +x /usr/bin/jenkins-slave

ENTRYPOINT ["jenkins-slave"]
```

  - docker build -t 10.0.54.8/library/jenkins-slave-jdk:1.8 .
- 上传jenkins-salve到镜像仓库
  - docker push 10.0.54.8/library/jenkins-slave-jdk:1.8

jenkin私钥凭证



类型 SSH Username with private key
范围 全局 (Jenkins, nodes, items, all child items, etc)
ID
描述 haiyang-gitlab
Username
Private Key ● Enter directly
Key

```
b/81ZLtiHNtbuepfwWHtoMo+ljEs1tXiaT8vIFsh4lChZp
mY3gWPL6JL3ZYFLQAAABFoYWl5YW5nQGxvY2FsaG9zdA==
-----END OPENSSH PRIVATE KEY-----
```

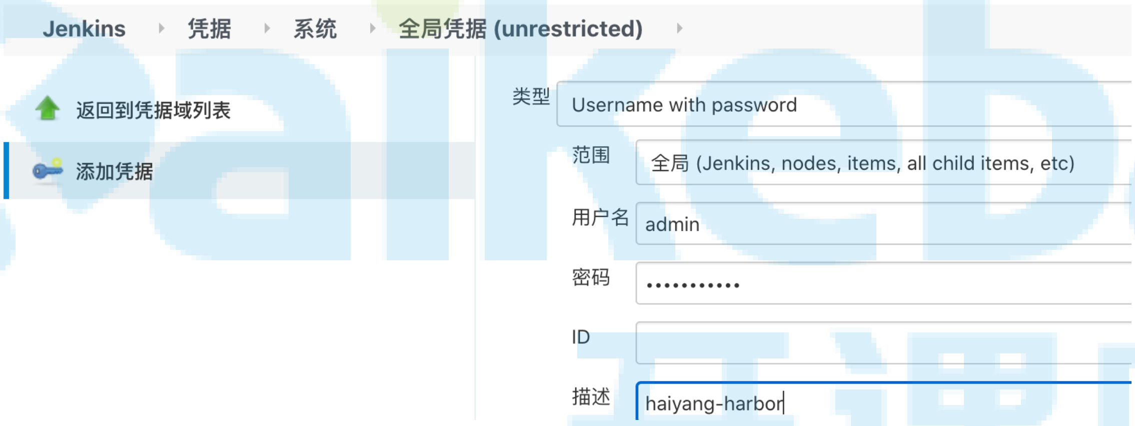
- 测试jenkins代理是否生效

```


podTemplate(label: 'jenkins-slave', cloud: 'kubernetes', containers: [
    containerTemplate(
        name: 'jnlp',
        image: "10.0.54.8/library/jenkins-slave-jdk:1.8"
    ),
  ],
)
{

node ("jenkins-slave") {
    stage('拉取代码') {
    git credentialsId: 'f3774951-6115-43d1-84da-066629855a5c', url:
'git@10.0.54.4:root/haiyang_test.git'
    }
}
}
```

- 在k8s中查看jenkins-slave是否出现
    - kubectl get pod

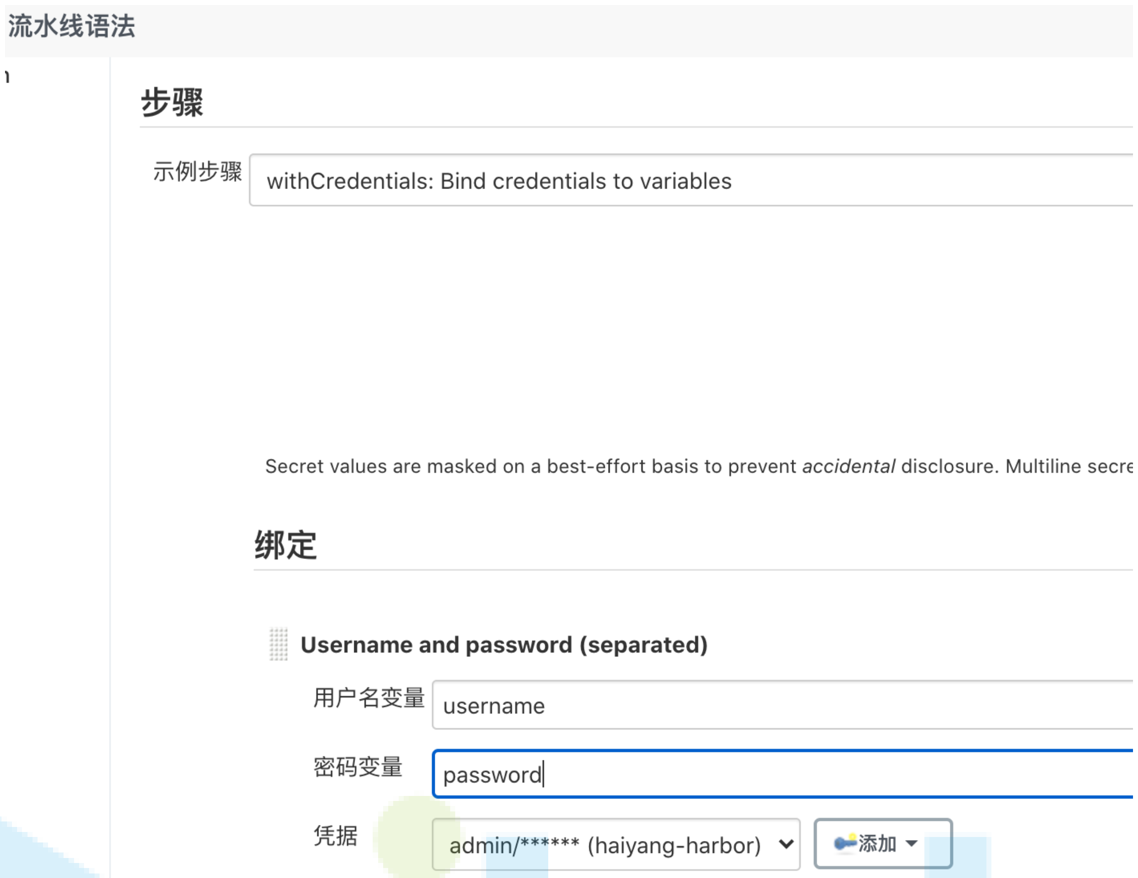
harbor凭证

- harbor账号密码转为凭证



类型 Username with password
范围 全局 (Jenkins, nodes, items, all child items, etc)
用户名 admin
密码 •••••••••••
ID
描述 haiyang-harbor

- 账号密码凭证转为变量，用于pipeline


流水线语法

步骤

示例步骤 withCredentials: Bind credentials to variables

Secret values are masked on a best-effort basis to prevent *accidental* disclosure. Multiline secre

绑定

Username and password (separated)

用户名变量 username

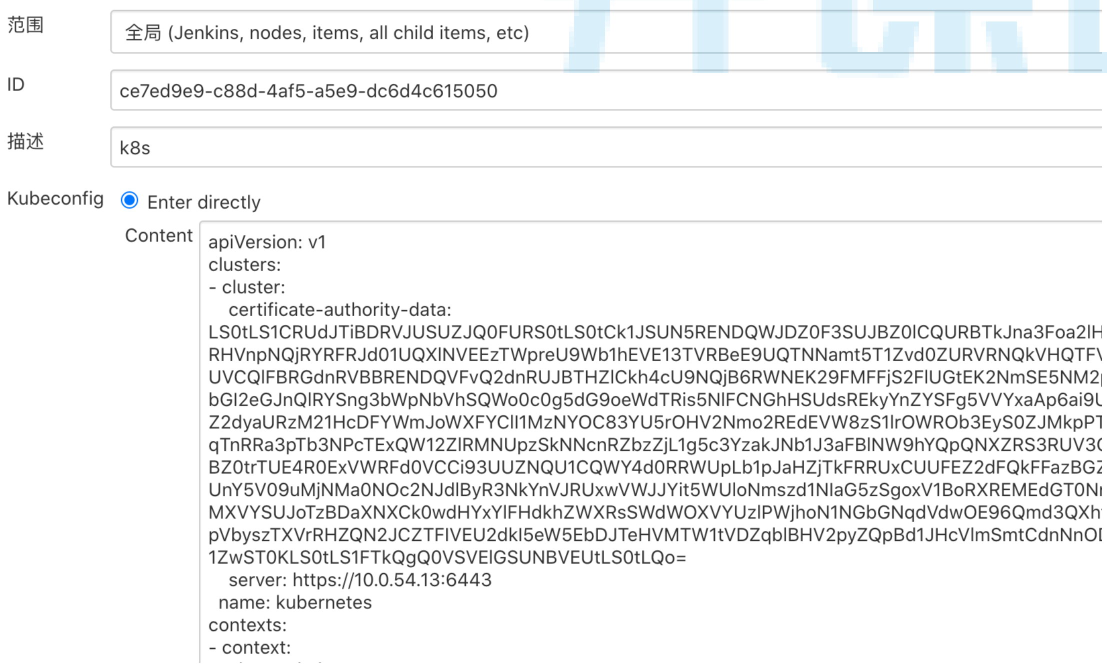
密码变量 password

凭据 admin/****** (haiyang-harbor)

- 创建 secret_name 认证，用于k8s部署pod时拉取镜像使用
  - kubectl create secret docker-registry registry-pull-secret --docker-username=admin --docker-password=kuick123456 --docker-email=ithuhaiyang@163.com --docker-server=10.0.54.8

K8S凭证

- cat .kube/config，将整个kube文件全部复制


范围 全局 (Jenkins, nodes, items, all child items, etc)

ID ce7ed9e9-c88d-4af5-a5e9-dc6d4c615050

描述 k8s

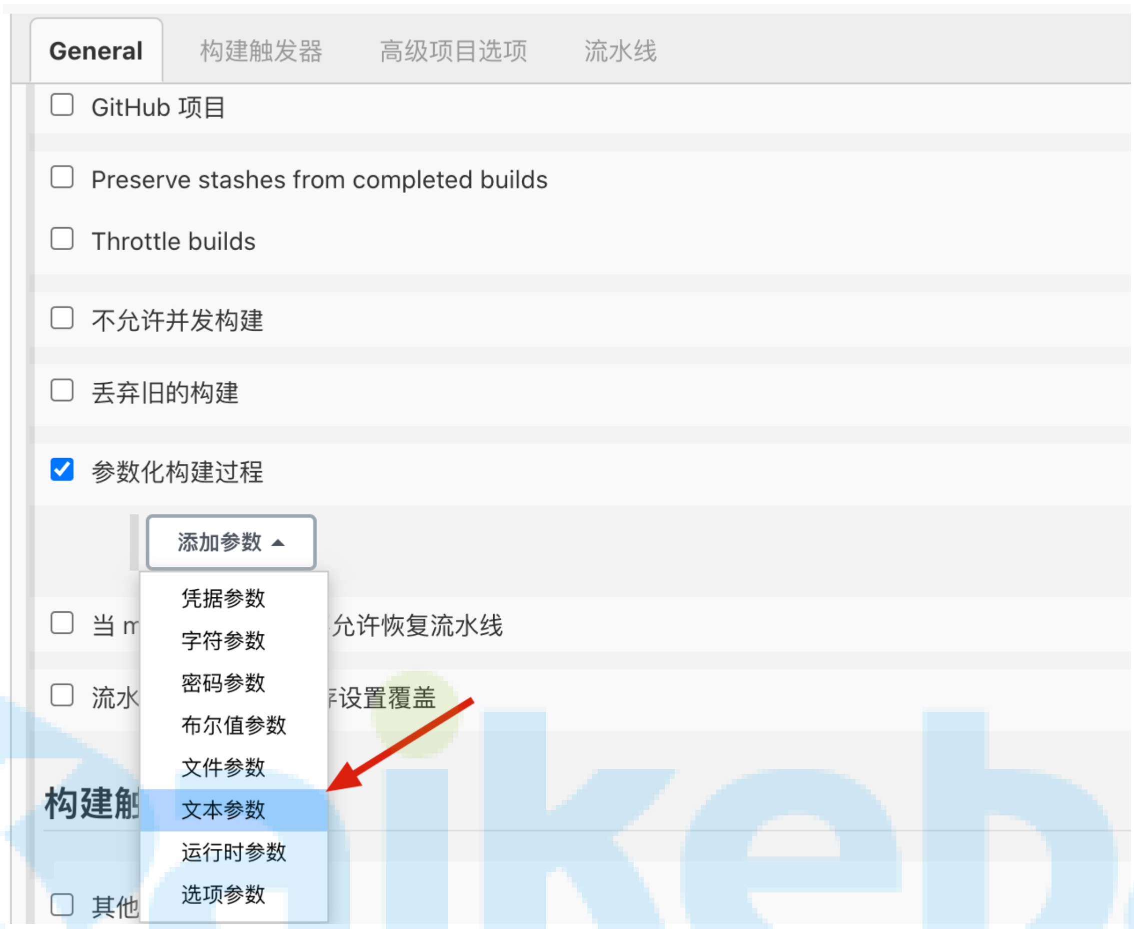
Kubeconfig (●) Enter directly

Content

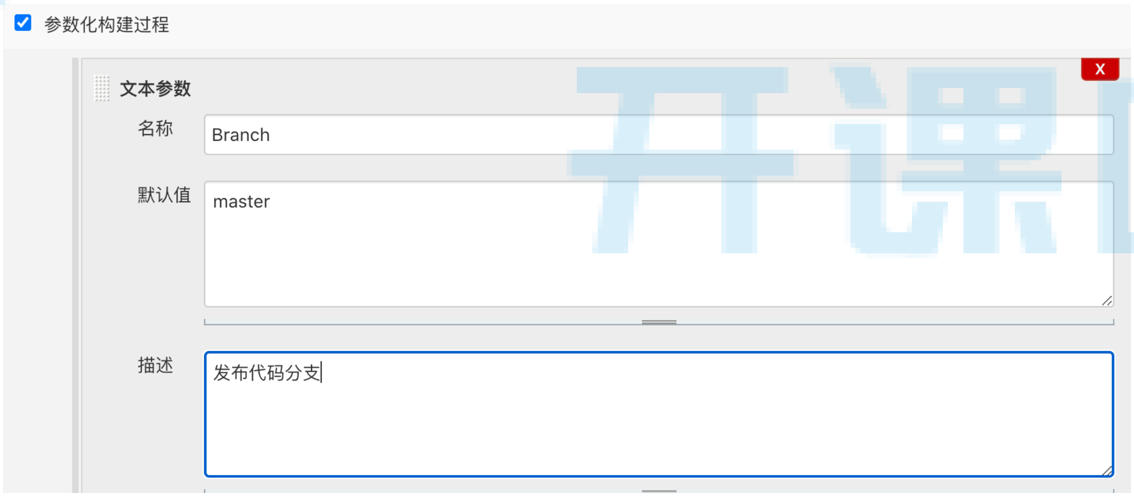
```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTkJna3Foa2lH
RHVnpNQjRYRFRJd01UQXlNVEEzTWpreU9Wb1hEVE13TVRBeE9UQTNNamt5T1Zvd0ZURVRNQkVHQTF
UVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTHZlCkh4cU9NQjB6RWNEK29FMFFjS2FlUGtEK2NmSE5NM2
bGI2eGJnQlRYSng3bWpNbVhSQWo0c0g5dG9oeWdTRis5NlFCNGhHSUdsREkyYnZYSFg5VVYxaAp6ai9U
Z2dyaURzM21HcDFYWmJoWXFYCll1MzNYOC83YU5rOHV2Nmo2REdEVW8zS1lrOWROb3EyS0ZJMkpPT
qTnRRa3pTb3NPcTExQW12ZlRMNUpzSkNNcnRZbzZjL1g5c3YzakJNb1J3aFBlNW9hYQpQNXZRS3RUV3C
BZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0RRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFFazBGZ
UnY5V09uMjNMa0NOc2NJdlByR3NkYnVJRUxwVWJJYit5WUloNmszd1NlaG5zSgoxV1BoRXREMEdGT0N
MXVYSUJoTzBDaXNXCk0wdHYxYlFHdkhZWXRsSWdWOXVYUzlPWjhoN1NGbGNqdVdwOE96Qmd3QXh
pVbyszTXVrRHZQN2JCZTFIVEU2dkI5eW5EbDJTeHVMTW1tVDZqblBHV2pyZQpBd1JHcVlmSmtCdnNnOD
1ZwST0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
    server: https://10.0.54.13:6443
  name: kubernetes
contexts:
- context:
```

分支参数化构建

- 文本参数



- 构建参数

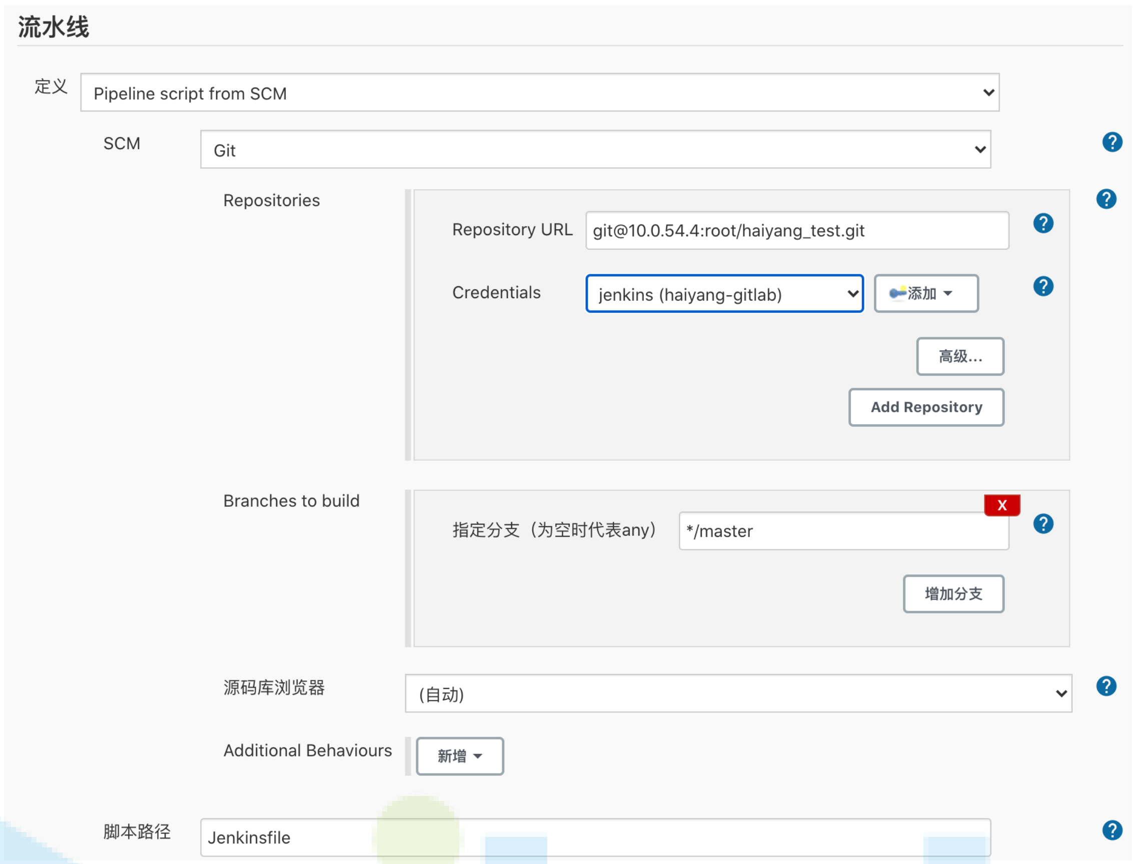


名称 Branch

默认值 master

描述 发布代码分支

- 在jenkins'的pipeline代码放到项目代码中，这样方便管理


流水线

定义: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: git@10.0.54.4:root/haiyang_test.git

Credentials: jenkins (haiyang-gitlab)

Branches to build

指定分支（为空时代表any）: */master

源码库浏览器: (自动)

Additional Behaviours

脚本路径: Jenkinsfile

Java-Gradle-发布到k8s

Jenkins-slave

- 当jenkins发布任务时，jenkins-slave会以pod的方式去运行任务，任务结束pod终止

```
FROM centos:7
#LABEL haiyang

RUN yum install -y java-1.8.0-openjdk-devel.x86_64  maven curl git libtool-ltdl-devel && \
    yum clean all && \
    rm -rf /var/cache/yum/* && \
    mkdir -p /usr/share/jenkins

ENV JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.262.b10-0.el7_8.x86_64/
ENV JRE_HOME $JAVA_HOME/jre
ENV PATH $PATH:$JAVA_HOME/bin
ENV CLASSPATH .:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar


COPY slave.jar /usr/share/jenkins/slave.jar
COPY jenkins-slave /usr/bin/jenkins-slave
COPY settings.xml /etc/maven/settings.xml
RUN chmod +x /usr/bin/jenkins-slave
```

```
ENTRYPOINT ["jenkins-slave"]
```

- docker build -t 10.0.54.8/library/jenkins-slave-jdk:2.4 .

如下这几个文件，全部都放到要发布的代码仓库中

Jenkins-pipeline脚本

```
// harbor镜像仓库地址
def registry = "10.0.54.8"

// 上传到harbor项目名称，jenkins每次构建版本名称
def project = "baiwan"
def app_name = "haiyang_gradle_test"
def image_name = "${registry}/${project}/${app_name}:${BUILD_NUMBER}"

//git地址换成自己的仓库地址
def git_address = "git@10.0.54.4:root/spring-cloud-config-server.git"
def k8s_auth = "ce7ed9e9-c88d-4af5-a5e9-dc6d4c615050"

// 认证-账号脱敏
def secret_name = "registry-pull-secret"
def docker_registry_auth = "b51ba954-a17b-40a5-8c2b-df297d7dc60f"
def git_auth = "f3774951-6115-43d1-84da-066629855a5c"

//pipeline中jenkins-slave配置
podTemplate(label: 'jenkins-slave', cloud: 'kubernetes', containers: [
    containerTemplate(
        name: 'jnlp',
        image: "${registry}/library/jenkins-slave-jdk:2.4"
    ),
  ],
  volumes: [
    hostPathVolume(mountPath: '/var/run/docker.sock', hostPath:
'/var/run/docker.sock'),
    hostPathVolume(mountPath: '/usr/bin/docker', hostPath: '/usr/bin/docker')
  ],
)

{
  node("jenkins-slave"){
      // 第一步，拉取你的项目代码到本地
      stage('拉取代码'){
         checkout([$class: 'GitSCM', branches: [[name: '${Branch}']],
userRemoteConfigs: [[credentialsId: "${git_auth}", url: "${git_address}"]]])
      }
      // 第二步进行编译，编译完成copy到你的镜像中
      stage('代码编译'){
          sh "java -version"
          sh "./gradlew clean build"
          sh "pwd"
          sh "ls build/libs/"
      }
      // 第三步，构建你的docker镜像，dockerfile是在你的代码仓库中，以拉取到本地，直接docker build即可
      stage('构建镜像'){
          withCredentials([usernamePassword(credentialsId: "${docker_registry_auth}", passwordVariable: 'password', usernameVariable: 'username')]) {
            sh """
              ls
              docker build -t ${image_name} .
              docker login -u ${username} -p '${password}' ${registry}
              docker push ${image_name}
               """
            }
      }
      // 第四步，将你的打包好的镜像发布到k8s中，Deploy.yml也是在你的代码仓库，yml文件需要根据你的需求自己去定义，不是通配的
      stage('部署到K8S平台完成'){
          sh """
          sed -i 's#\$IMAGE_NAME#${image_name}#' Deploy.yml
          sed -i 's#\$SECRET_NAME#${secret_name}#' Deploy.yml
          """
          kubernetesDeploy configs: 'Deploy.yml', kubeconfigId: "${k8s_auth}"
      }
    }
}
```

Dockerfile

- 根据你的项目语言，去编写你的dockerfile文件

```
FROM 10.0.54.8/library/alpine-oraclejdk8:1.0

# Update apk mirror
RUN cp /etc/apk/repositories /etc/apk/repositories.bak
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.aliyun.com/g' /etc/apk/repositories

# Install tools  #RUN apk update && apk add curl

# Project path
RUN mkdir -p /kuick/servers
ENV PROJECT_PATH /kuick/servers/
VOLUME /tmp

#将编译好的jar包传到docker镜像的项目工作目录
COPY ./build/libs/*.jar $PROJECT_PATH
COPY run.sh   $PROJECT_PATH
RUN ls  $PROJECT_PATH
WORKDIR $PROJECT_PATH


EXPOSE 8080
ENV JAVA_OPTS="-Duser.timezone=GMT+8 -Xms512m -Xmx2048m"
# CMD ["sleep","3600"]  调试docker镜像
CMD ["./run.sh"]
```

run.sh

- 启动命令

```
#!/usr/bin/env sh
# -*- encoding UTF-8 -*-
# Author: Johny

# Main bootRun
java $JAVA_OPTS -jar $PROJECT_PATH*.jar
```

Deploy.yml

- 发布到k8s中的资源清单，根据自己需求定义

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "11"
  generation: 13
  labels:
    app: spring-cloud-config-server
    chart: spring-cloud-config-server-2.0.0-0b06bf
    env: pro
    heritage: Helm
    language: java
    namespace: default
    release: springcloudconfigserver
    serviceKind: backend
  name: springcloudconfigserver
  namespace: default
  #resourceVersion: "220661382"
  selfLink: /apis/apps/v1/namespaces/default/deployments/springcloudconfigserver
  #uid: 59594331-8f67-11ea-92bd-00163e2e8090
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: spring-cloud-config-server
      release: springcloudconfigserver
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: spring-cloud-config-server
        release: springcloudconfigserver
    spec:
      containers:
      - env:
        - name: SENTRY_SERVERNAME
          value: spring-cloud-config-server
        - name: aliyun_logs_springcloudconfigserver-stdout
          value: stdout
        - name: aliyun_logs_springcloudconfigserver_ttl
          value: "15"
        image: $IMAGE_NAME
        imagePullPolicy: Always
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /manage/health
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 30
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: spring-cloud-config-server
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /manage/health
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 40
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        resources:
          limits:
            cpu: "1"
            memory: 4Gi
          requests:
            cpu: 500m
            memory: 1Gi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      imagePullSecrets:
      - name: $SECRET_NAME
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30

---

apiVersion: v1
kind: Service
metadata:
  labels:
    app: spring-cloud-config-server
    chart: spring-cloud-config-server-2.0.0-0b06bf
    heritage: Helm
    release: springcloudconfigserver
  name: springcloudconfigserver
  selfLink: /api/v1/namespaces/kuick-prod/services/springcloudconfigserver
spec:
  clusterIP: 10.106.61.115
  ports:
  - name: spring-cloud-config-server
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: spring-cloud-config-server
    release: springcloudconfigserver
  sessionAffinity: None
  type: ClusterIP
```

Node-发布到K8S

- jenkins

```
// harbor镜像仓库地址
def registry = "10.0.54.8"

// harbor定义项目名称，BUILD_NUMBER jenkins每次构建版本
def project = "baiwan"
def app_name = "deal-behaviour-server"
def image_name = "${registry}/${project}/${app_name}:${BUILD_NUMBER}"
def git_address = "git@10.0.54.4:root/deal-behaviour-server.git"
def k8s_auth = "ce7ed9e9-c88d-4af5-a5e9-dc6d4c615050"

// 认证-账号脱敏
def secret_name = "registry-pull-secret"
def docker_registry_auth = "b51ba954-a17b-40a5-8c2b-df297d7dc60f"
def git_auth = "f3774951-6115-43d1-84da-066629855a5c"

podTemplate(label: 'jenkins-slave', cloud: 'kubernetes', containers: [
    containerTemplate(
        name: 'jnlp',
        image: "${registry}/library/jenkins-slave-jdk:2.4"
    ),
  ],
  volumes: [
    hostPathVolume(mountPath: '/var/run/docker.sock', hostPath:
'/var/run/docker.sock'),
    hostPathVolume(mountPath: '/usr/bin/docker', hostPath:
'/usr/bin/docker')
  ],
)

{
  node("jenkins-slave"){
      // 第一步
      stage('拉取代码'){
        checkout([$class: 'GitSCM', branches: [[name: '${Branch}']],
userRemoteConfigs: [[credentialsId: "${git_auth}", url: "${git_address}"]]])
      }
      // 第二步
      stage('代码编译'){
          sh "pwd"
          sh "ls"
      }
      // 第三步
      stage('构建镜像'){
          withCredentials([usernamePassword(credentialsId:
"${docker_registry_auth}", passwordVariable: 'password', usernameVariable:
'username')]) {
            sh """
              ls
              docker build -t ${image_name} .
              docker login -u ${username} -p '${password}' ${registry}
              docker push ${image_name}
               """
            }
      }
      // 第四步
      stage('部署到K8S平台完成'){
          sh """
          sed -i 's#\$IMAGE_NAME#${image_name}#' Deploy.yml
          sed -i 's#\$SECRET_NAME#${secret_name}#' Deploy.yml
          """
          kubernetesDeploy configs: 'Deploy.yml', kubeconfigId:
"${k8s_auth}"
      }
  }
}
```

- dockerfile

```
# Latest Alpine
FROM 10.0.54.8/library/behaviour-server:base
MAINTAINER Johny.Zheng <shun.johny@gmail.com>>

# Update apk mirror
RUN cp /etc/apk/repositories /etc/apk/repositories.bak
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.aliyun.com/g'
/etc/apk/repositories

# ENV
ENV KUICK_HOME /servers/kuick-server

# Copy project to docker
COPY . $KUICK_HOME

RUN pwd
RUN ls

# Install deal-behaviour server
RUN cd $KUICK_HOME/ && cnpm install -d
```

```
# Install grunt
#RUN cnpm install grunt-cli -g

# Expose port
EXPOSE 1888

# Workdir
WORKDIR $KUICK_HOME/

# Enterport
ENTRYPOINT ["node"]

# defatul cmd
CMD ["server.js"]
```

- deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "19"
  generation: 115
  labels:
    app: deal-behaviour-server
    chart: deal-behaviour-server-2.0.0-978b3f
    env: prod
    heritage: Helm
    language: nodejs
    release: dealbehaviourserver
    serviceKind: backend
  name: dealbehaviourserver
  selfLink: /apis/apps/v1/namespaces/kuick-
prod/deployments/dealbehaviourserver
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: deal-behaviour-server
      release: dealbehaviourserver
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: deal-behaviour-server
        release: dealbehaviourserver
    spec:
      containers:
      - env:
        - name: CLOUD_CONFIG_APPLICATION
```

```yaml
          value: deal-behaviour-server
        - name: CLOUD_CONFIG_ENDPOINT
          value: http://kuickconfigserver
        - name: CLOUD_CONFIG_LABEL
          value: k8s-master
        - name: CLOUD_CONFIG_PROFILE
          value: pro
        - name: CONFIG_SOURCE
          value: cloud
        - name: NODE_ENV
          value: production
        - name: SENTRY_SERVERNAME
          value: deal-behaviour-server
        - name: TINGYUN_APP_NAME
          value: deal-behaviour-server
        - name: TINGYUN_ENABLED
          value: "true"
        - name: TINGYUN_LICENSE_KEY
          value: 660743e058f2b474d65cfd6114417bd7
        - name: aliyun_logs_dealbehaviourserver-stdout
          value: stdout
        - name: aliyun_logs_dealbehaviourserver_ttl
          value: "15"
        image: $IMAGE_NAME
        imagePullPolicy: Always
        name: deal-behaviour-server
        resources:
          limits:
            cpu: "1"
            memory: 2Gi
          requests:
            cpu: "1"
            memory: 2Gi
        securityContext:
          capabilities: {}
      dnsPolicy: ClusterFirst
      imagePullSecrets:
      - name: $SECRET_NAME
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      tolerations:
      - key: virtual-kubelet.io/provider
        operator: Exists

---

apiVersion: v1
kind: Service
metadata:
  labels:
    app: deal-behaviour-server
    chart: deal-behaviour-server-2.0.0-978b3f
    heritage: Helm
    release: dealbehaviourserver
  name: dealbehaviourserver
  selfLink: /api/v1/namespaces/kuick-prod/services/dealbehaviourserver
```

```
spec:
  clusterIP: 10.98.204.159
  ports:
  - name: deal-behaviour-server
    port: 80
    protocol: TCP
    targetPort: 1222
  selector:
    app: deal-behaviour-server
    release: dealbehaviourserver
  sessionAffinity: None
  type: ClusterIP
```