

# Kubernetes 集群使用 Jenkins 持续发布

## 运行环境准备与相关软件要求

1. 生产级 kubernetes 集群 推荐 1.8 以上版本
2. Gitlab docker 镜像 slpcat/gitlab-ce
3. Jenkins docker 镜像  
Jenkins master 镜像 slpcat/kube-jenkins-service  
已安装需要的各种插件，已预先配置使用 kubernetes 管理 jenkins slave  
Jenkins slave 镜像 slpcat/kube-jenkins-dind-agent  
支持 jnlp 协议，git 源码管理，以及 Docker in Docker
4. Docker 私有镜像仓库，用户名密码，用于存放生成的 docker 镜像

## 实现功能与目标

1. gitlab 与 jenkins 使用 openid 统一认证，认证源为 gitlab 账户
2. Jenkins 使用 kubernetes 进行分布式构建任务
3. Jenkins 在 kubernetes 集群里面自动发布业务 pod, svc
4. 钉钉机器人通知 jenkins 任务构建进度与结果
5. 最终目标是开发人员运行 git push 提交代码，业务自动上线运行，无需人工干预

## 安装过程

搭建 kubernetes 集群

略

安装 Gitlab 到 kubernetes

使用持久化存储

镜像 slpcat/gitlab-ce

集群内域名 gitlab.default.svc.cluster.local

外部域名 git.example.com

配置样本：<https://github.com/slpcat/docker-images/tree/master/gitlab-ce>

安装 Jenkins 到 kubernetes

使用持久化存储

需要创建 serviceaccount 授权 jenkins 完全访问 kubernetes 集群

镜像 slpcat/kube-jenkins-service

集群内域名 jenkins.default.svc.cluster.local

外部域名 jenkins.example.com

配置样本：<https://github.com/slpcat/docker-images/tree/master/kube-jenkins-service>

## 相关功能配置

jenkins 使用 gitlab 认证用户

Gitlab 建立应用

Admin Area → Applications → New application

The screenshot shows the 'Edit application' page in the GitLab Admin Area. The left sidebar contains the 'Admin Area' menu with options like Overview, Monitoring, Messages, System Hooks, Applications (selected), Abuse Reports, Deploy Keys, Service Templates, Labels, Appearance, and Settings. The main content area shows the 'jenkins' application details. The 'Name' field is 'jenkins'. The 'Redirect URI' is 'http://jenkins.123456789.cn/securityRealm/finishLogin'. The 'Trusted' checkbox is unchecked. The 'Scopes' section is expanded, showing a list of permissions: 'api' (checked), 'read\_user' (checked), 'sudo' (unchecked), and 'read\_registry' (unchecked). Below the scopes, there is a 'Submit' button.

The screenshot shows the 'Applications' page in the GitLab Admin Area. A blue banner at the top states 'The application was created successfully.' Below this, the 'Application: jenkins' details are shown. The 'Application Id' is '0e7a72a7477ff77a56e701cc6d66d297fc32c6fb0e01cae91234e49c'. The 'Secret' is 'fed2cad550f23ad4220726fdb8c8d48152cac37c51e1a9a0f123456789d5'. The 'Callback url' is 'http://jenkins.123456789.cn/securityRealm/finishLogin'. The 'Trusted' checkbox is unchecked. The 'Scopes' section is expanded, showing a list of permissions: 'api' (checked), 'read\_user' (checked), and 'openid' (checked). Below the scopes, there are 'Edit' and 'Destroy' buttons.

Application name : jenkins

CallbackURL: http://jenkins.example.com/securityRealm/finishLogin

Scops: **read\_user, openid**

记下 Application Id 和 Secret

Jenkins 安全设置

系统管理 → 全局安全管理



**Kubernetes**

Name:

Kubernetes URL:

Kubernetes server certificate key:

Disable https certificate check: ☐

Kubernetes Namespace:

Credentials:  [Add](#)

Jenkins URL:  [Test Connection](#)

Jenkins tunnel:

Connection Timeout:

Read Timeout:

Container Cap:

Max connections to Kubernetes API:

Defaults Provider Template Name:  [View](#)

Images

**Kubernetes Pod Template**

Name:

[Save](#) [Apply](#)

Name 填 kubernetes

Kubernetes URL 填 <https://kubernetes.default.svc.cluster.local>

Kubernetes Namespace 填 default

Credentials 填 jenkins 这个 service account

然后测试连接

Jenkins URL 填 <http://jenkins.default.svc.cluster.local:8080>

创建 Jenkins slave 的 kubernetes pod 模板

Defaults Provider Template Name:

Images

**Kubernetes Pod Template**

Name:

Namespace:

Labels:

Usage:

The name of the pod template to inherit from:

Containers

**Container Template**

Name:

Docker image:

Always pull image: ☐

Working directory:

Command to run slave agent:

Arguments to pass to the command:

Allocate pseudo-TTY: ☐

EnvVars: [Add Environment Variable](#)

Run in privileged mode: ☒

Request CPU:

Request Memory:

Limit CPU:

[Save](#) [Apply](#)

Defaults Provider Template Name 填 jenkins-slave

## Kubernetes Pod Template

Name 填 jnlp-slave

Namespace 填 default

Labels 填 jnlp-slave

## Container Template

Name 填 jnlp

Docker image 填 slpcat/kube-jenkins-dind-agent

Working directory 填 /home/jenkins

Command to run slave agent 清空

Arguments to pass to the command 填 `${computer.jnlpmac}`  
`${computer.name}`

高级里面

Run in privileged mode 勾选 特权模式运行 pod

## 项目发布流程

1. 开发人员运行 Git push 提交代码到 gitlab 代码库
2. Gitlab 的 web hook 触发 jenkins 构建任务
3. Jenkins master 使用 kubernetes 集群创建 jenkins slave
4. jenkins slave 根据构建任务定义执行动作

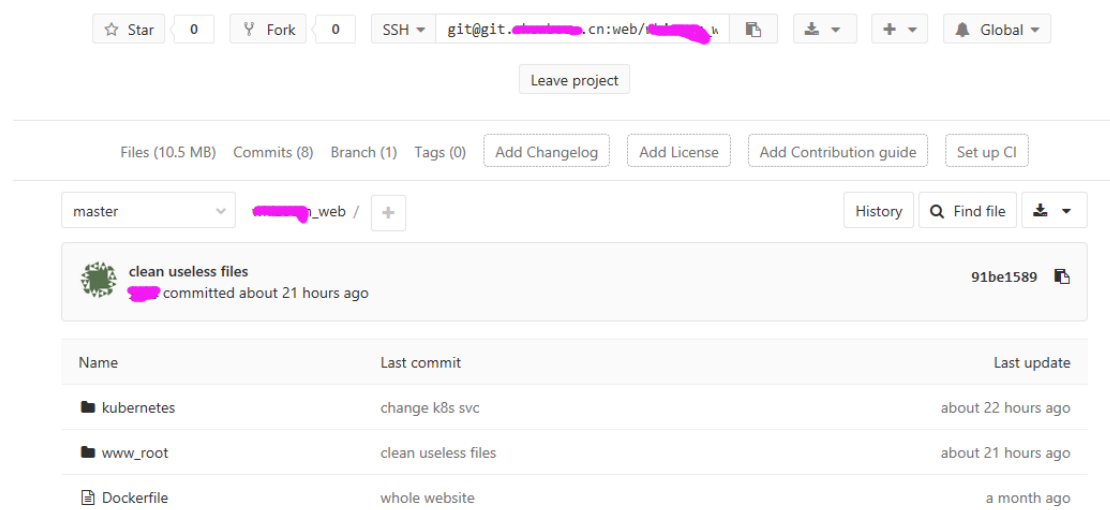
从 gitlab 代码库下载项目代码，根据 Dockerfile 定义制作 docker 镜像，上传 docker 镜像到私有仓库 registry，根据模板生成 kubernetes 配置文件，最后部署镜像到 kubernetes 运行

5. 任务完成，使用钉钉机器人发送通知

## DevOPS 操作规范与要求

### 项目代码与服务模板

每个 git 项目内容如下



The screenshot shows a GitLab repository interface. At the top, there are buttons for Star (0), Fork (0), SSH, and a dropdown menu. Below these is a 'Leave project' button. The main section shows the repository name 'clean useless files' and a commit message 'clean useless files' committed about 21 hours ago. Below this is a table with columns 'Name', 'Last commit', and 'Last update'.

Name	Last commit	Last update
kubernetes	change k8s svc	about 22 hours ago
www_root	clean useless files	about 21 hours ago
Dockerfile	whole website	a month ago

Dockerfile：镜像构建文件

kubernetes：kubernetes 相关配置文件模板目录 deploy svc pvc 等

www\_root：业务代码目录

以及其他配置文件与脚本（根据需要添加）

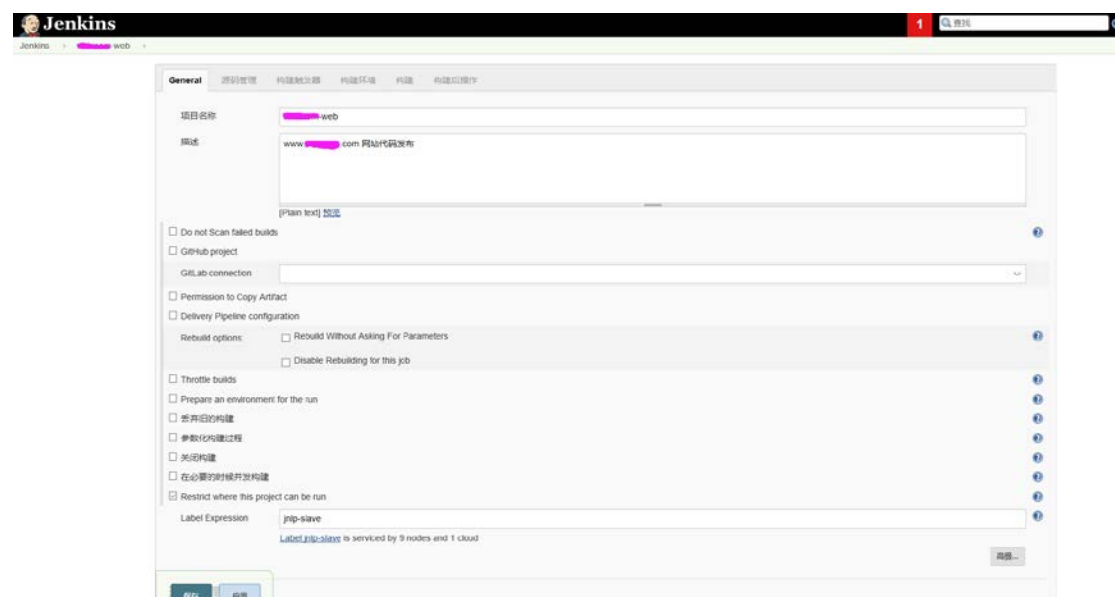
## Jenkins 新建构建任务



The image shows the 'New Item' wizard in Jenkins. At the top, there is a section titled '输入一个任务名称' (Enter a task name) with an empty text box and a '必填项' (Required) label. Below this, there are two main options: '构建一个自由风格的软件项目' (Build a free-style software project) and '构建一个maven项目' (Build a maven project). The first option is selected and has a description: '这是Jenkins的主要功能. Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.' The second option has a description: '构建一个maven项目. Jenkins利用你的POM文件. 这样可以大大减轻构建配置.'

选择构建一个自由风格的软件项目

### General 配置



The image shows the 'General' configuration page for a Jenkins job. The job name is 'web' and the description is 'www.\*\*\*.com 网站代码发布'. The 'Restrict where this project can be run' checkbox is checked, and the 'Label Expression' is set to 'jnlp-slave'. The 'Save' button is highlighted in green.

填写名字和描述

勾选 Restrict where this project can be run

Label Expression 填 jnlp-slave

限制运行 设置标签表达式 jnlp-slave

### 源码管理

**源码管理**

☐ None  
☐ CVS  
☐ CVS Projectset  
☒ Git

Repositories

Repository URL:

Credentials:

Name:

Rspec:

Branches to build

Branch Specifier (blank for 'any'):

源码库浏览器:

Additional Behaviours:

☐ Mercurial  
☐ Subversion

Repository URL

填写 git 地址 git@git.example.com/example\_web.git

Credentials 添加 git 账户名和 ssh 私钥

构建触发器 勾选 Build when a change is pushed to GitLab

General 源码管理 **构建触发器** 构建环境 构建 构建后操作

**构建触发器**

☐ 触发远程构建 (例如, 使用脚本)  
☐ Build after other projects are built  
☐ Build periodically  
☒ Build when a change is pushed to GitLab. GitLab CI Service URL: http://jenkins.example.com.cn/project/example\_web

Enabled GitLab triggers

Push Events	<input checked="" type="checkbox"/>
Opened Merge Request Events	<input checked="" type="checkbox"/>
Accepted Merge Request Events	<input checked="" type="checkbox"/>
Closed Merge Request Events	<input type="checkbox"/>
Rebuild open Merge Requests	<input type="text" value="Never"/>
Comments	<input type="checkbox"/>
Comment (regex) for triggering a build	<input type="text" value="Jenkins please retry a build"/>

☒ Enable [ci-skip]  
☒ Ignore WIP Merge Requests  
☒ Set build description to build cause (eg. Merge request or Git Push)  
☐ Build on successful pipeline events

Allowed branches

☒ Allow all branches to trigger this job  
☐ Filter branches by name  
☐ Filter branches by regex  
☐ Filter merge request by label

Secret token:

记下触发器地址和生成的 Secret token

Gitlab 项目内设置 web hook

项目名 → Settings → Integrations

填写 Jenkins 生成的触发器地址和 Secret token

构建 增加构建步骤

Docker build and publish

Repository Name 填写镜像名称

Tag 填写镜像标签 本例使用\${BUILD\_NUMBER} 构建号码作为 tag

Docker registry URL 填写私有仓库地址 https://registry.cn-beijing.aliyuncs.com/v2/

Registry credentials 设置私有仓库用户名密码

Deploy to kubernetes

填写 k8s 凭证 包括地址 证书，复制 kubectl 控制用户 ~/.kube/config 相关字段

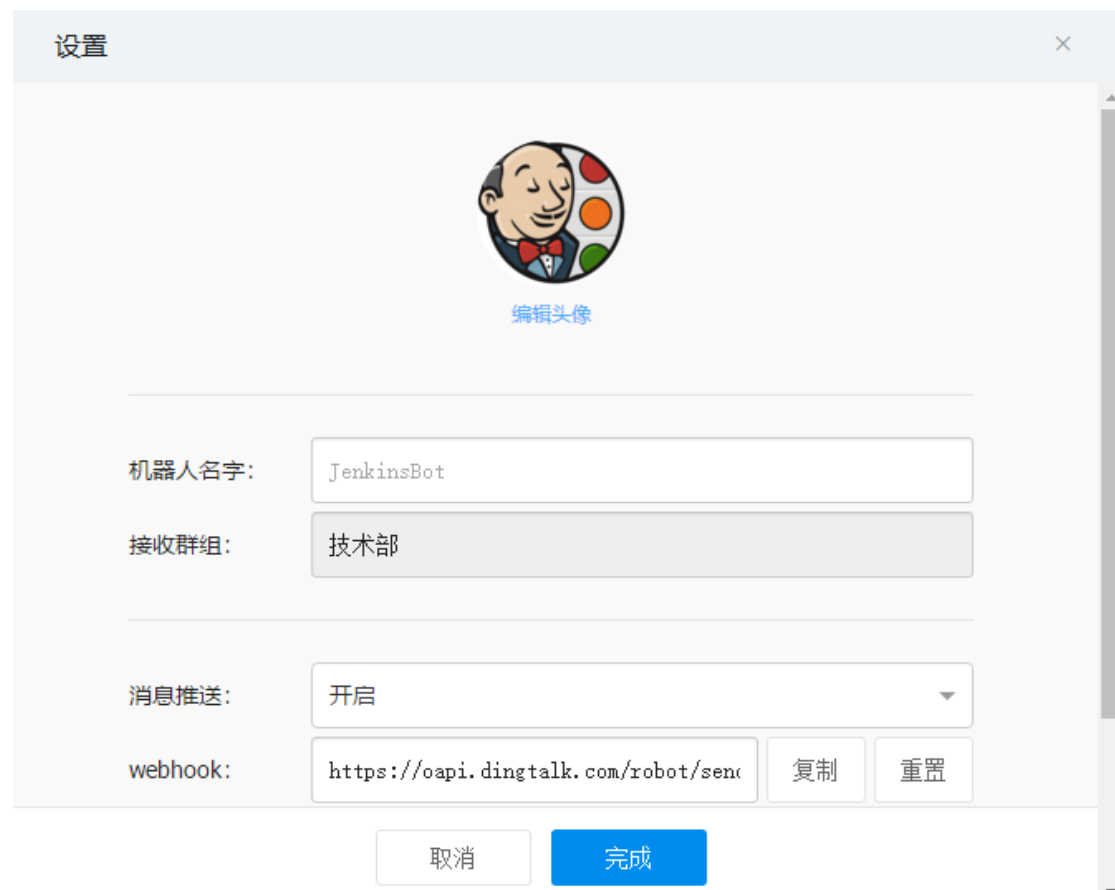
Config Files 填写项目内 kubernetes 配置文件路径 逗号分隔

Enable Variable Substitution in Config 勾选 允许变量替换




构建后操作

钉钉电脑客户端—钉钉群—群设置—群机器人—添加群机器人—自定义



设置



[编辑头像](#)

机器人名字:

接收群组:

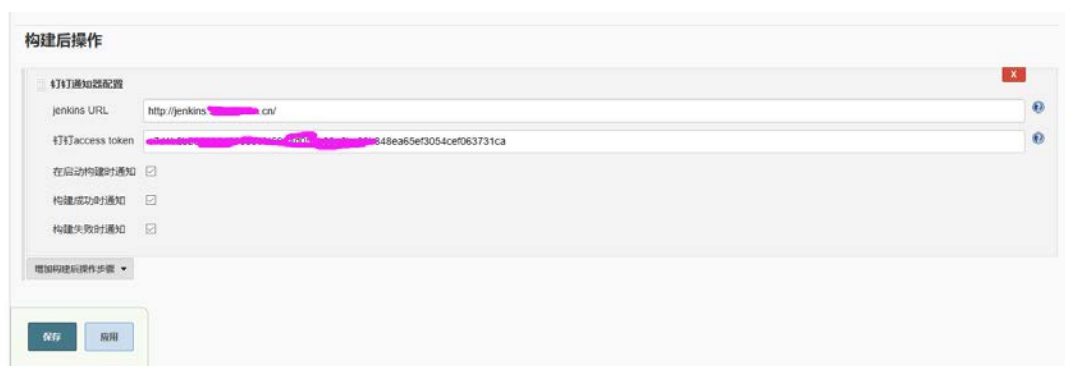
消息推送:

webhook:  [复制](#) [重置](#)

[取消](#) [完成](#)

记下 access\_token

jenkins 增加构建后操作步骤→钉钉通知器配置



构建后操作

钉钉通知器配置

jenkins URL:

钉钉access token:

☒ 在启动构建时通知

☒ 构建成功时通知

☒ 构建失败时通知

[增加构建后操作](#)

[保存](#) [应用](#)

jenkins URL 填 `http://jenkins.example.com/`

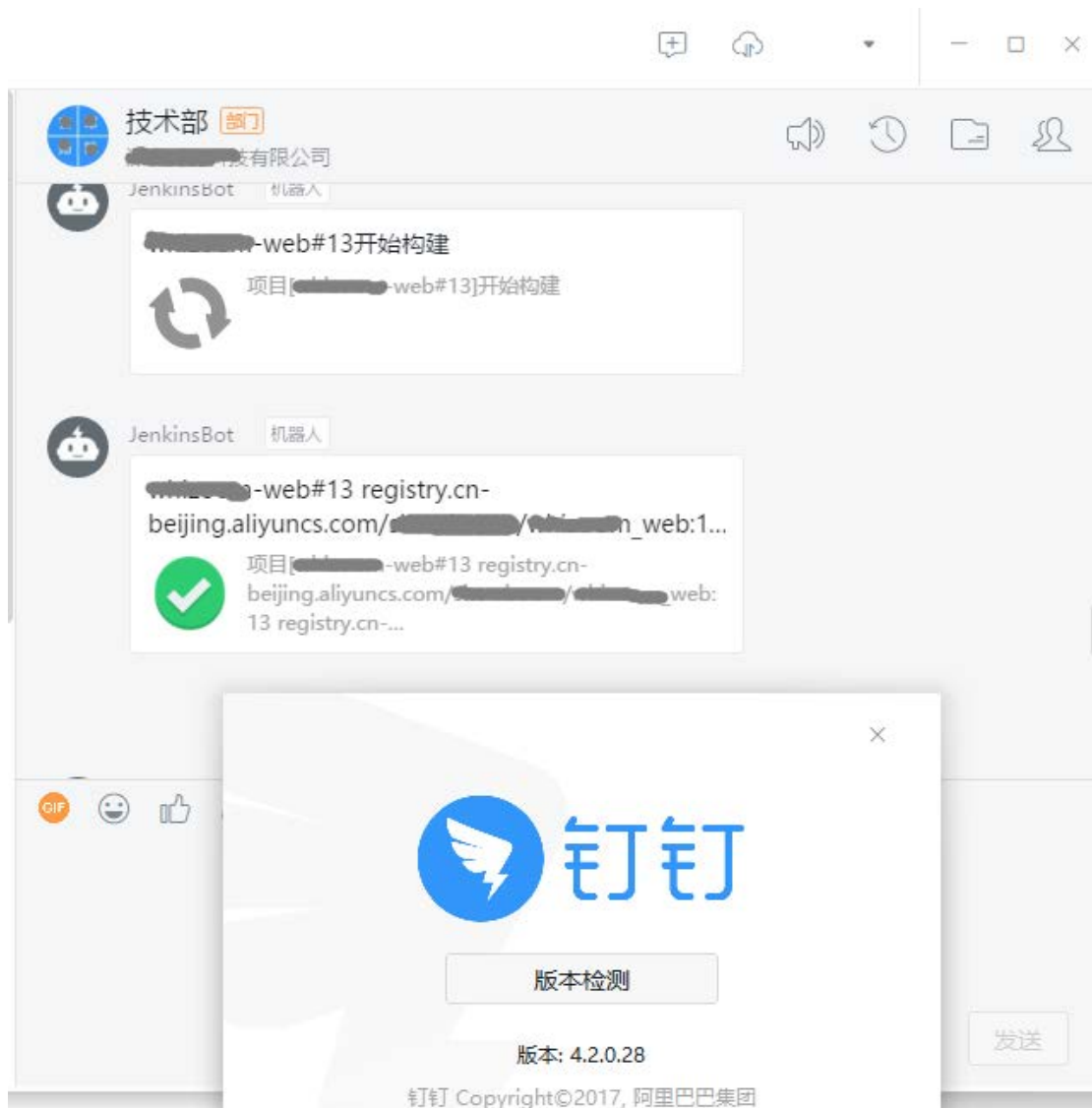
钉钉 access token 填写生成的 access\_token 字符串

勾选相应的通知选项

演示 任务构建控制台输出

```
16:11:37 85d83f4edb18: Layer already exists
16:11:37 73b4683e66e8: Layer already exists
16:11:37 ee60293db08f: Layer already exists
16:11:37 3efd1f7c01f6: Layer already exists
16:11:37 58bcc73dcf40: Layer already exists
16:11:37 9dc188d975fd: Layer already exists
16:11:37 latest: digest: sha256:062465ba061f95d9989b0898251623788e9d99990a60074aad88f252b62f7721d size: 4097
16:11:37 Starting Kubernetes deployment
16:11:38 Loading configuration: /home/jenkins/workspace/钉钉群-web/kubernetes/钉钉群-web-deploy.yml
16:11:40 Applied Deployment: Deployment(apiVersion=extensions/v1beta1, Kind=Deployment, metadata=ObjectMeta(annotations=null, clusterName=null, createTimestamp=2018-01-08T06:16:47Z, deletionGracePeriodSeconds=null, deletionTimestamp=null, finalizers=[], generateName=null, generation=4, initializers=null, namespace=default, ownerReferences=[], resourceVersion=3670247, selfLink=/apis/extensions/v1beta1/namespaces/default/deployments/钉钉群-web, uid=8a061f95d9989b0898251623788e9d99990a60074aad88f252b62f7721d), spec=DeploymentSpec(minReadySeconds=null, paused=null, progressDeadlineSeconds=null, replicas=1, revisionHistoryLimit=null, strategy=DeploymentStrategy(rollingUpdate=RollingUpdateDeployment(maxSurge=1, maxUnavailable=1, type=RollingUpdate), type=RollingUpdate), template=PodTemplateSpec(metadata=ObjectMeta(annotations=null, clusterName=null, creationTimestamp=2018-01-08T06:16:47Z, deletionGracePeriodSeconds=null, deletionTimestamp=null, finalizers=[], generateName=null, generation=1, initializers=null, labels={app=钉钉群-web}, name=null, namespace=default, ownerReferences=[], resourceVersion=3670267, selfLink=/apis/extensions/v1beta1/namespaces/default/pods/钉钉群-web-13, uid=fe11da9c-dc9c-11e7-9bd7-00163e03c313), spec=PodSpec(activeDeadlineSeconds=null, affinity=null, automountServiceAccountToken=null, containers=[Container(args=[], command=[test], env=[], envFrom=[], image=registry.cn-beijing.aliyuncs.com/钉钉群-web:13, imagePullPolicy=Always, lifecycle={exec=[httpGet-HTTPGetAction(host=null, httpHeaders=[], path=/, port=IntOrString(IntVal=80, Kind=null, StrVal=null, additionalProperties={})], failureThreshold=5, httpGet=HTTPGetAction(host=null, httpHeaders=[], path=/, port=IntOrString(IntVal=80, Kind=null, StrVal=null, additionalProperties={})], initialDelaySeconds=60, periodSeconds=10, successThreshold=1, tcpSocket=null, timeoutSeconds=5, additionalProperties={})], readinessProbe={exec=[], failureThreshold=3, periodSeconds=10, tcpSocket=null, timeoutSeconds=5, additionalProperties={})}], dnsPolicy=ClusterFirst, hostAliases=[], hostIPC=null, hostNetwork=null, hostPID=null, hostname=null, imagePullSecrets=[], initContainers=[], nodeName=null, nodeSelector=null, restartPolicy=Always, schedulerName=default-scheduler, securityContext=SecurityContext(runAsNonRoot=null, runAsUser=null, selinuxOptions=null, supplementalGroups=[], additionalProperties={}), serviceAccount=null, serviceAccountName=null, terminationGracePeriodSeconds=30, tolerations=[], volumes=[]), additionalProperties={}), additionalProperties={}), status=DeploymentStatus(availableReplicas=1, replicas=1, unavailableReplicas=0, updatedReplicas=1, additionalProperties={}), additionalProperties={})
16:11:40 Loading configuration: /home/jenkins/workspace/钉钉群-web/kubernetes/钉钉群-web-svc.yml
16:11:40 Applied Service: Service(apiVersion=v1, Kind=Service, metadata=ObjectMeta(annotations=null, clusterName=null, creationTimestamp=2018-01-08T06:16:47Z, deletionGracePeriodSeconds=null, deletionTimestamp=null, finalizers=[], generateName=null, generation=1, initializers=null, labels=null, name=钉钉群-web, namespace=default, ownerReferences=[], resourceVersion=3670267, selfLink=/apis/v1/namespaces/default/services/钉钉群-web, uid=fe11da9c-dc9c-11e7-9bd7-00163e03c313), spec=ServiceSpec(externalIPs=[], externalName=null, externalTrafficPolicy=Cluster, healthCheckNodePort=null, loadBalancerIP=null, loadBalancerSourceRanges=[], ports=[Port(name=tcp, port=80, protocol=TCP, targetPort=IntOrString(IntVal=80, Kind=null, StrVal=null, additionalProperties={}), additionalProperties={})], selector={app=钉钉群-web}, additionalProperties={}), status=ServiceStatus(loadBalancer=LoadBalancerStatus(ingress=[], additionalProperties={}), additionalProperties={}), additionalProperties={})
16:11:40 Finished Kubernetes deployment
16:11:40 Finished: SUCCESS
```

演示 钉钉群构建通知截图



参考

<https://jenkins.io/>