**1.1实战案例——的安装**

**1.1.1 案例目标**

（1）了解Jenkins的离线安装步骤。

（2）掌握Gitlab的使用和管理。

（3）了解CICD的配置步骤和方法。

**1.1.2 案例分析**

**1.规划节点**

ZooKeeper集群系统的节点规划，见表1-1-1。

表1-1-1节点规划

|  |  |  |
| --- | --- | --- |
| **IP** | **主机名** | **节点** |
| 10.24.2.10 | master | master节点 |
| 10.24.2.8 | node | node节点 |
| 10.24.2.10 | master | harbor节点 |
| 10.24.2.10 | master | cicd节点 |

**2.基础准备**

登录OpenStack平台，使用提供的CentOS\_7.5\_x86\_64\_XD.qcow2镜像创建两台云主机，并使用提供的软件包XianDian-PaaS-V2.4.iso部署好双节点Kubernetes集群。

**1.1.3 案例实施**

**1.安装Jenkins环境**

（1）基础环境准备

查看Kubernetes集群状态和节点信息：

[root@master ~]# kubectl get cs

NAME STATUS MESSAGE ERROR

controller-manager Healthy ok

scheduler Healthy ok

etcd-0 Healthy {"health":"true"}

[root@master ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

master Ready master 37m v1.18.1

node Ready <none> 3m59s v1.18.1

将提供的离线包jenkins\_offline.tar上传至master节点/root目录下，解压文件：

# tar -zxvf jenkins\_offline.tar -C /opt/

导入镜像：

# cd /opt/

# docker load -i jenkins.tar

（2）安装Jenkins

安装Jenkins：

[root@master ~]# docker run -d --name jenkins -p 8080:8080 -u root \

-v /home/jenkins\_home:/var/jenkins\_home \

-v /var/run/docker.sock:/var/run/docker.sock \

-v $(which docker):/usr/bin/docker \

-v /usr/bin/kubectl:/usr/local/bin/kubectl \

-v /root/.kube:/root/.kube \

jenkins/jenkins:2.262-centos

344d4fa5b8eac072b93d5e041fc0ba89401ec535d1f881b9713286820bf30c15

安装插件：

[root@master ~]# cp -rfv /opt/plugins/\* /home/jenkins\_home/plugins/

[root@master ~]# docker restart jenkins

jenkins

在web端通过http://IP:8080访问Jenkins，如图所示：



查看密码：

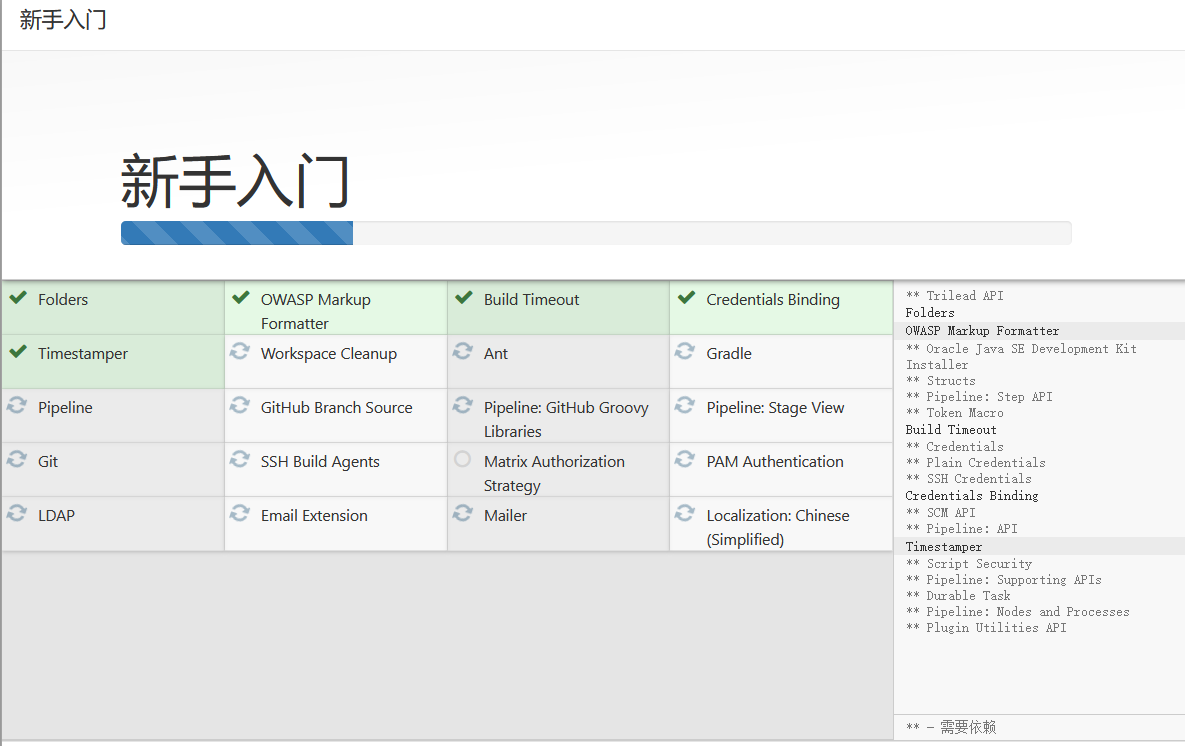
[root@master ~]# docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword

54670064056e42a2a47485a5e3356e57

输入密码并点击“继续”，如图所示：



选择“安装推荐的插件”，如图所示：



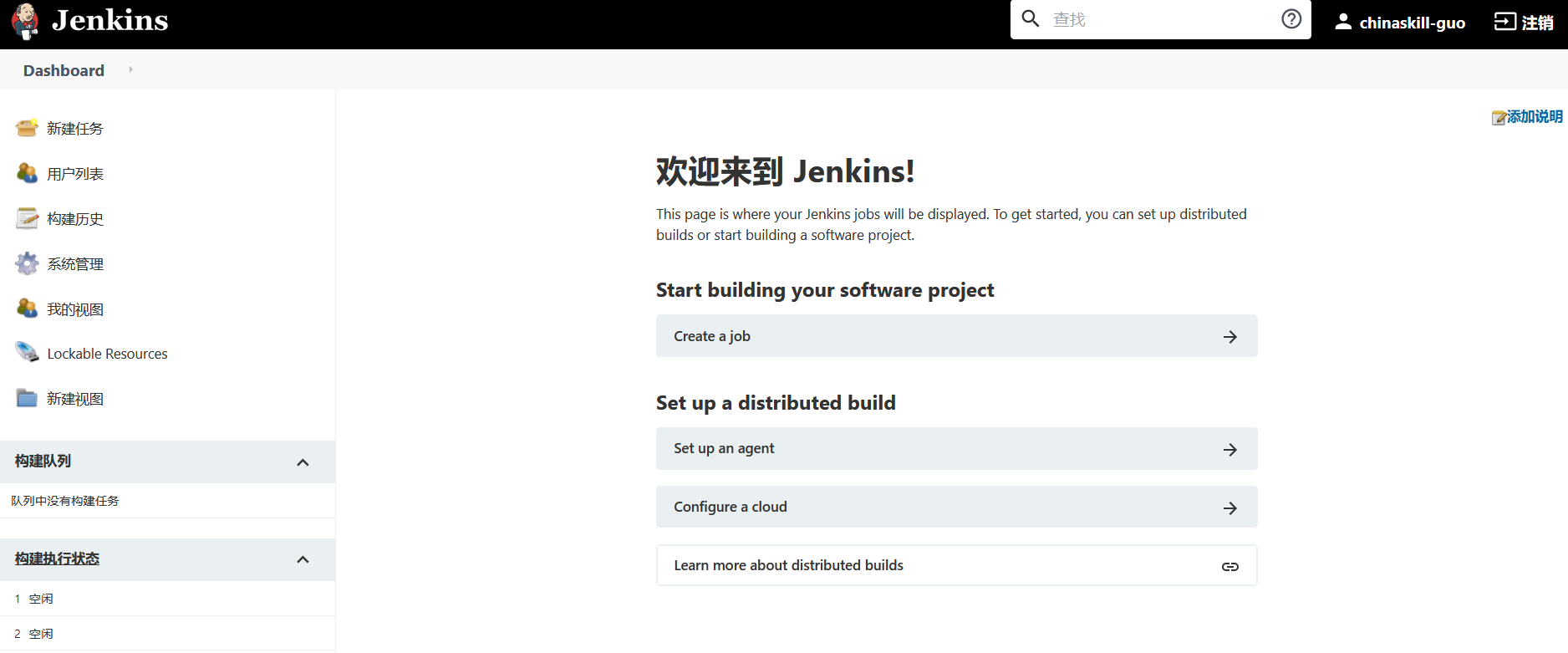
插件安装完成后自动转入管理员用户创建界面，如图所示：



输入用户信息，点击“保存并完成”，如图所示：



配置完URL后使用新建的用户登录Jenkins，如图所示：



**2.部署Gitlab**

GitLab是利用Ruby on Rails一个开源的版本管理系统，实现一个自托管的Git项目仓库，可通过Web界面进行访问公开的或者私人项目。与Github类似，GitLab能够浏览源代码，管理缺陷和注释，可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库，团队成员可以利用内置的简单聊天程序(Wall)进行交流。Gitlab还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

本项目Gitlab与Harbor共用一台服务器。

（1）启动Gitlab

启动Gitlab：

[root@master ~]# docker run -d -h gitlab -p 1022:22 -p 81:80 -p 443:443 \

--volume /srv/gitlab/config:/etc/gitlab \

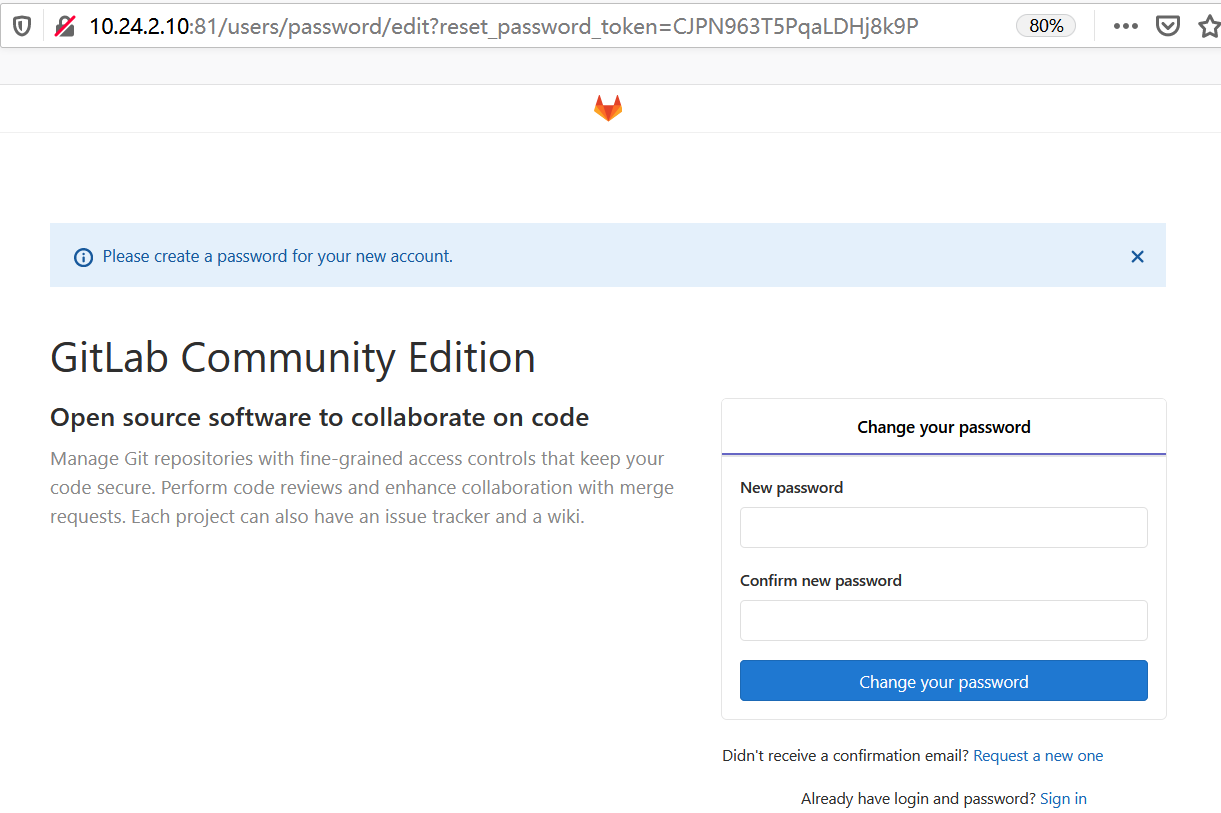
--volume /srv/gitlab/gitlab/logs:/var/log/gitlab \

--volume /srv/gitlab/gitlab/data:/var/opt/gitlab \

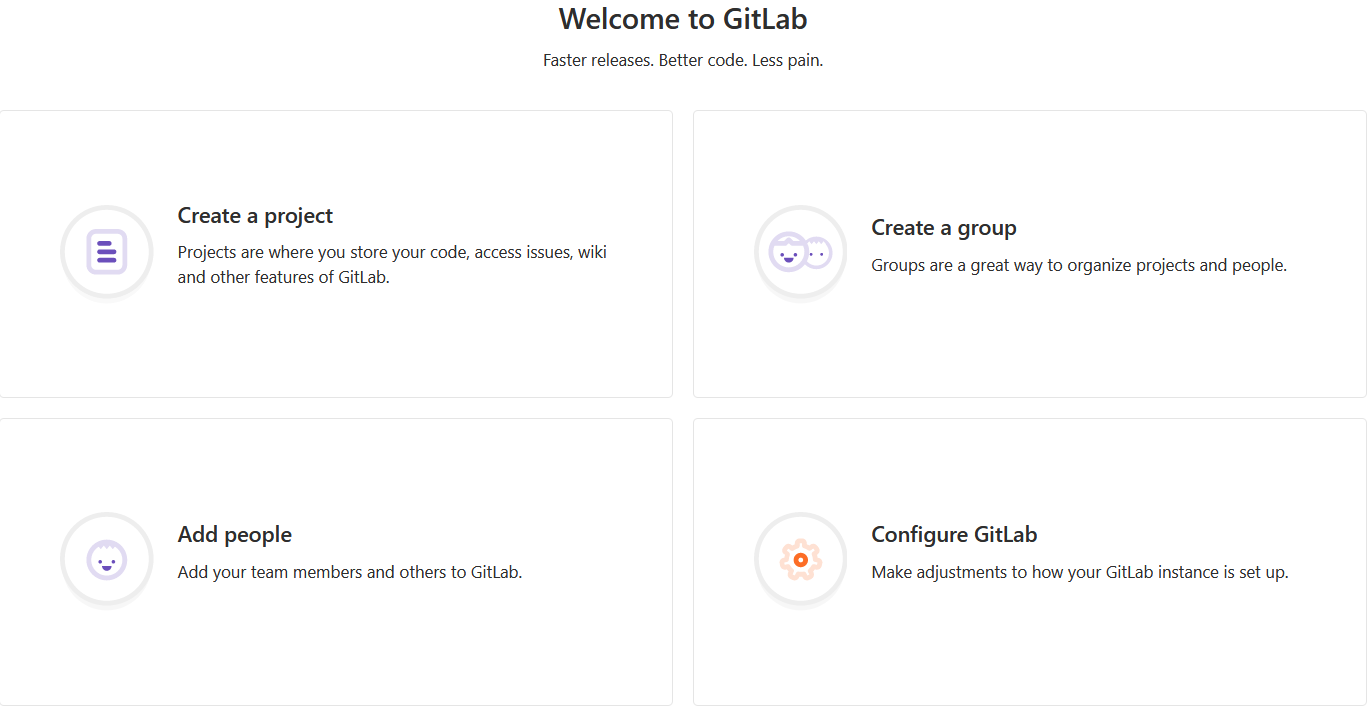
--restart always --name mygitlab gitlab/gitlab-ce:12.9.2-ce.0

37a3ff5d932fbcd6d471c52329faeb1defd1cd3ca0ad14f7cf12f31a721d502f

Gitlab启动较慢，可以通过docker logs查看启动状态。启动完成后，在web端访问Gitlab（http://IP:81），如图所示：

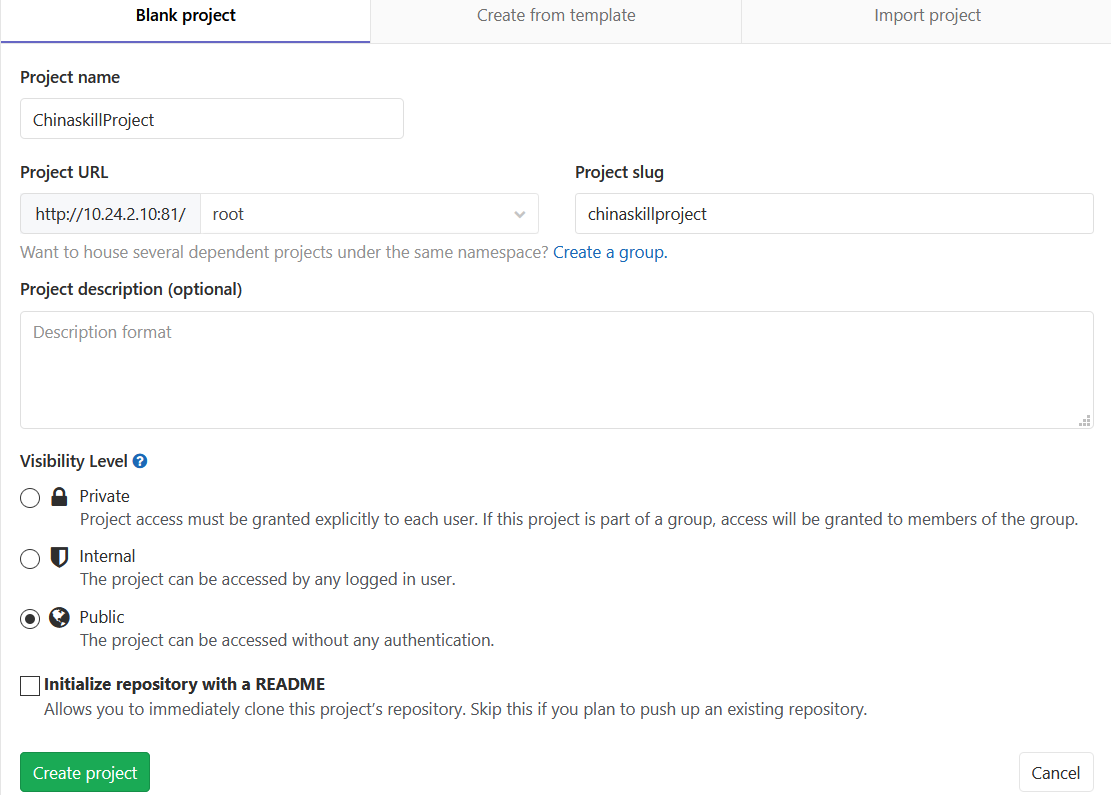


设置root用户信息并使用root用户登录Gitlab，如图所示：

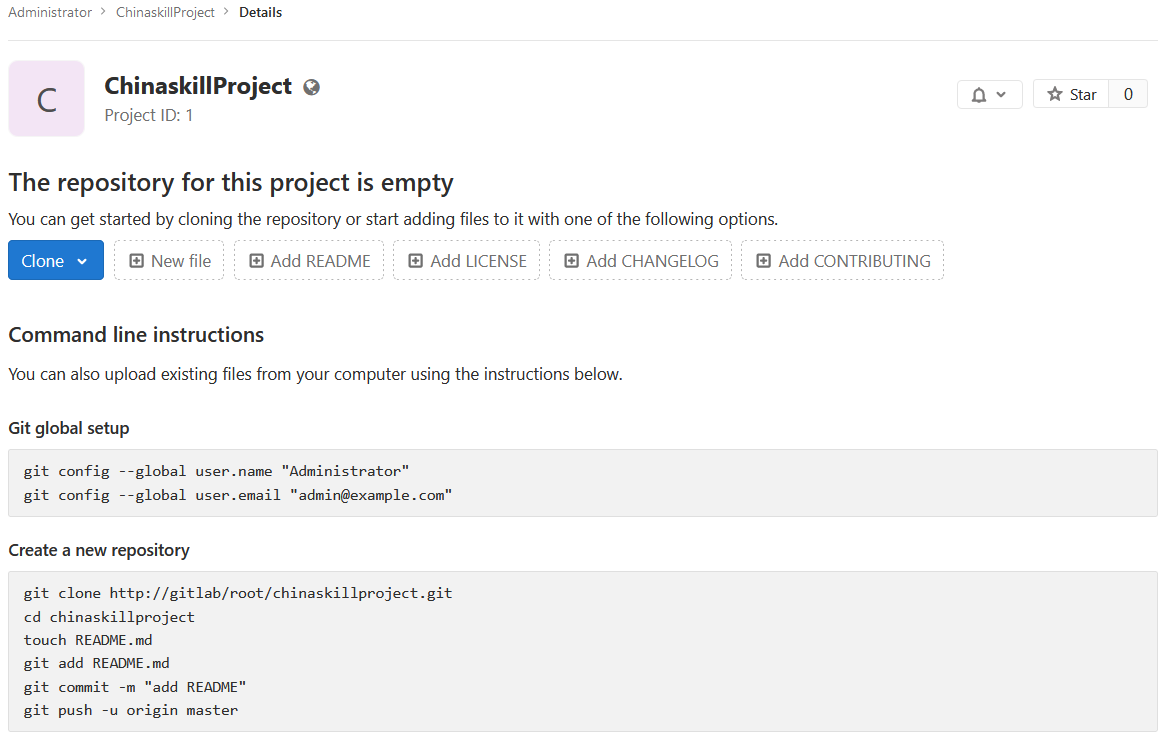


（2）创建项目

点击“Create a project”，创建项目ChinaskillProject，可见等级选择“Public”，如图所示：



点击“创建项目”，进入项目，如图所示：



push源代码到gitlab的ChinaskillProject项目：

[root@master ~]# yum install -y git

[root@master ~]# cd /opt/ChinaskillProject/

[root@master ChinaskillProject]# git config --global user.name "Administrator"

[root@master ChinaskillProject]# git config --global user.email "admin@example.com"

[root@master ChinaskillProject]# git remote remove origin

[root@master ChinaskillProject]# git remote add origin http://10.24.2.10:81/root/chinaskillproject.git

[root@master ChinaskillProject]# git add .

[root@master ChinaskillProject]# git commit -m "Initial commit"

[root@master ChinaskillProject]# git push -u origin master

Username for 'http://10.24.2.10:81': root

Password for 'http://root@10.24.2.10:81':

Counting objects: 3192, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (1428/1428), done.

Writing objects: 100% (3192/3192), 1.40 MiB | 0 bytes/s, done.

Total 3192 (delta 1233), reused 3010 (delta 1207)

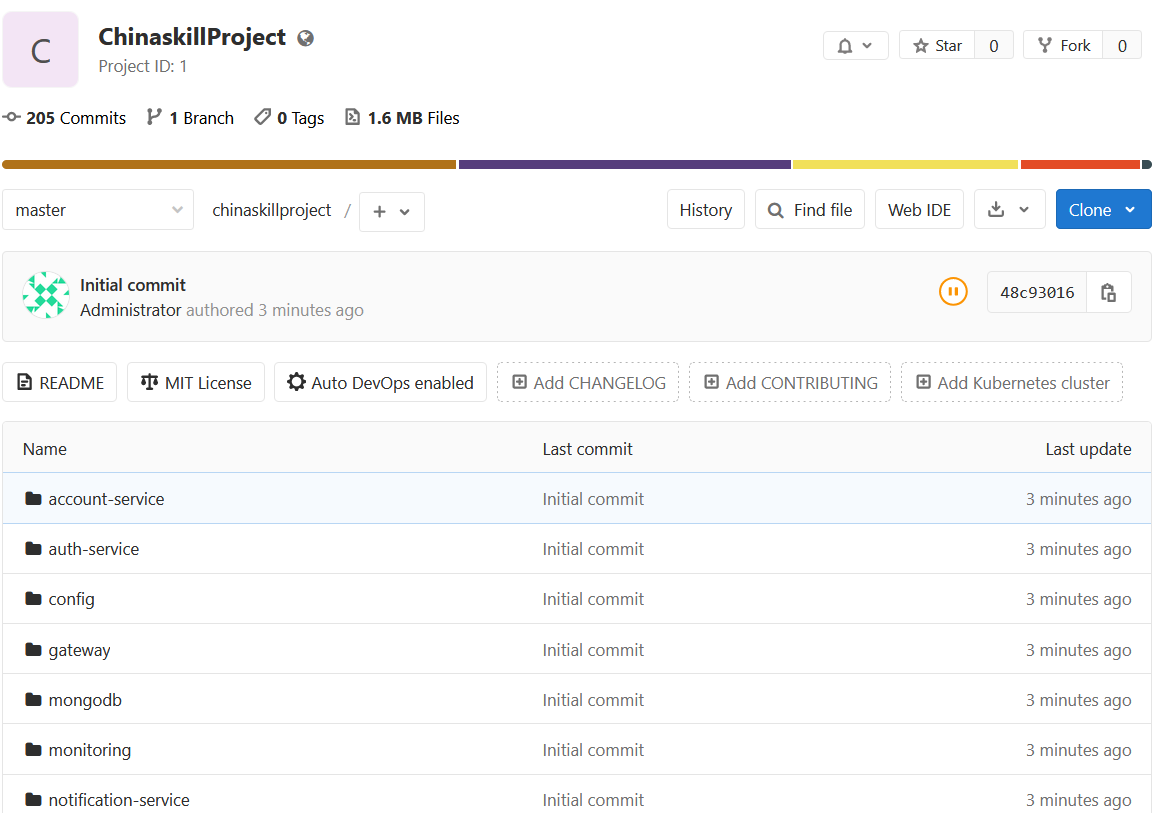
remote: Resolving deltas: 100% (1233/1233), done.

To http://10.24.2.10:81/root/chinaskillproject.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

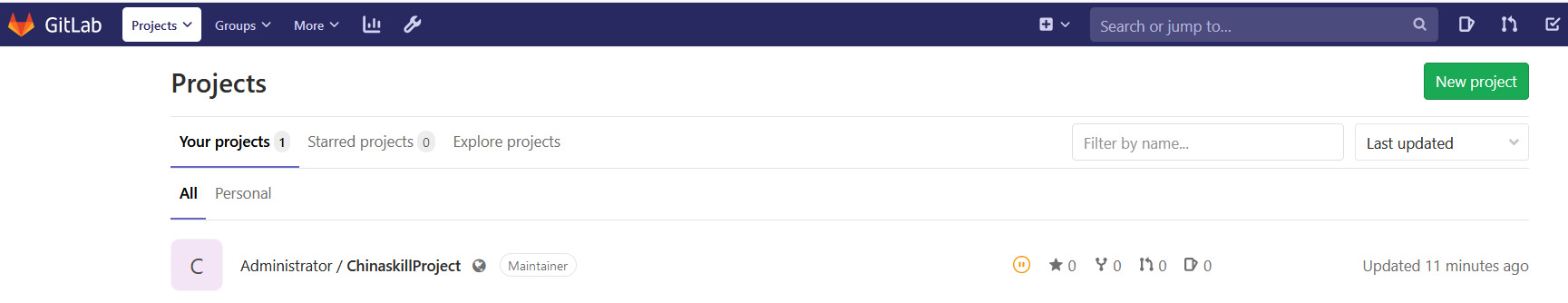
刷新网页，ChinaskillProject项目中文件已经更新了，如图所示：



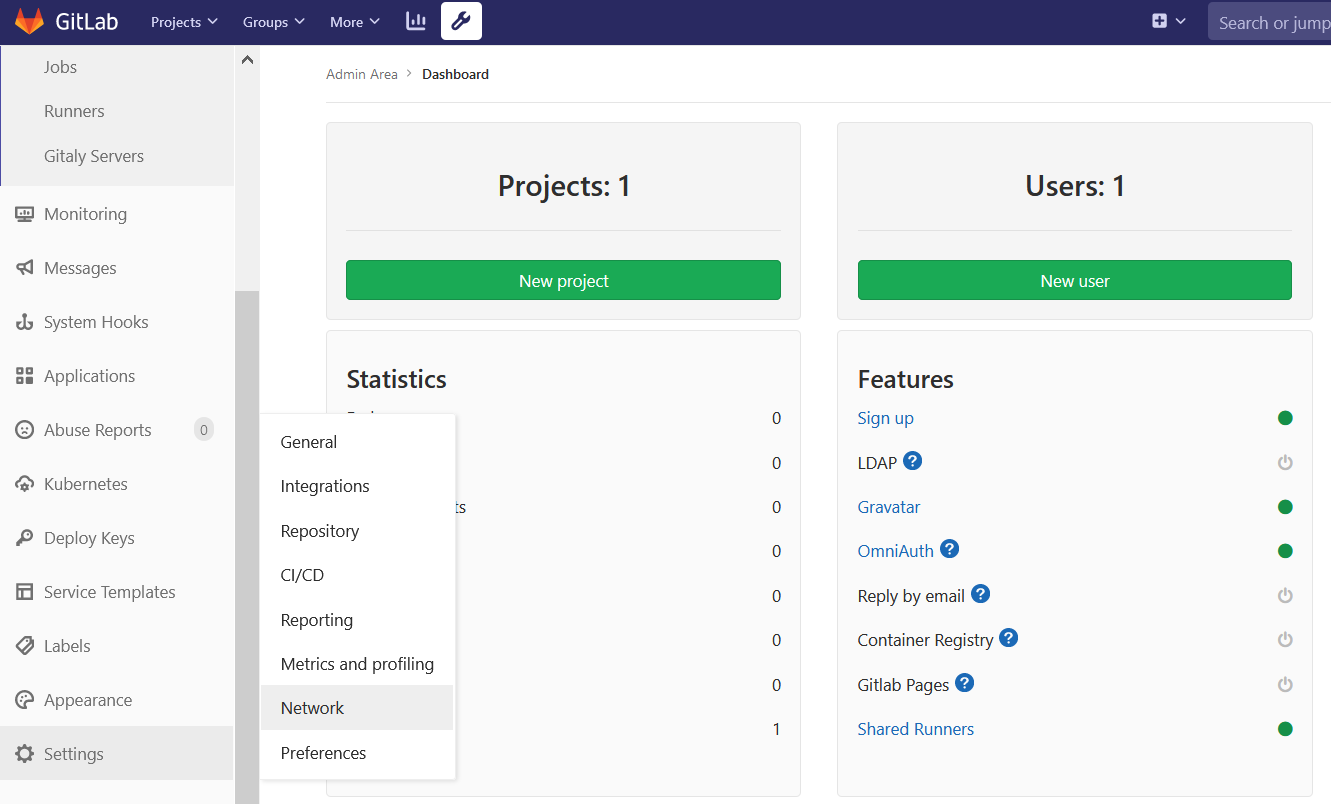
**3.配置Jenkins连接Gitlab**

（1）设置Outbound requests

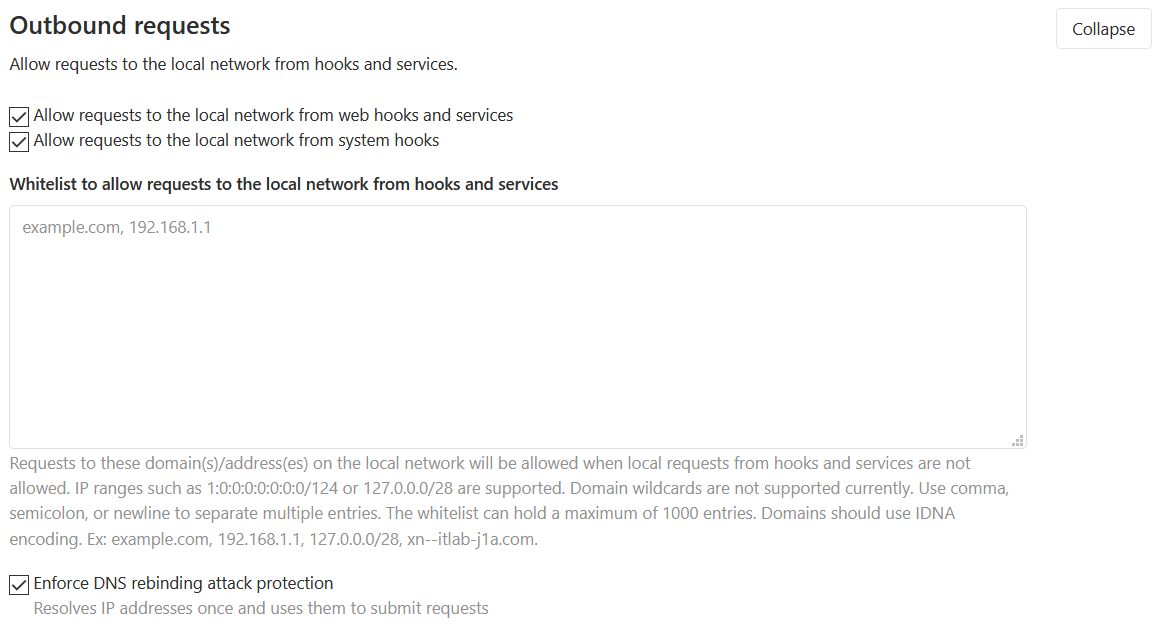
登录Gitlab首页，如图所示：



点击管理区域的扳手图标，如图所示：



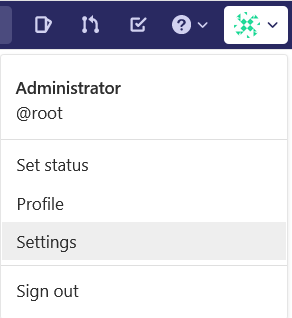
点击左侧导航栏的“Settings”→“Network”，设置“Outbound requests”，勾选“Allow requests to the local network from web hooks and services”，如图所示：



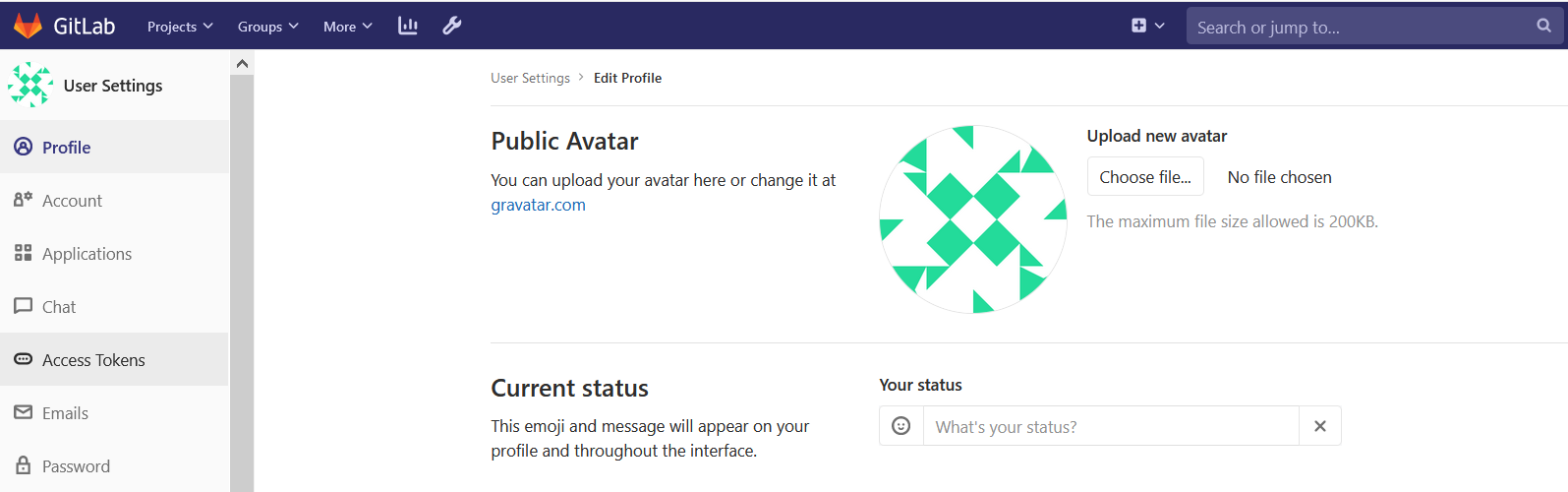
配置完成后保存。

（2）创建Gitlab API Token

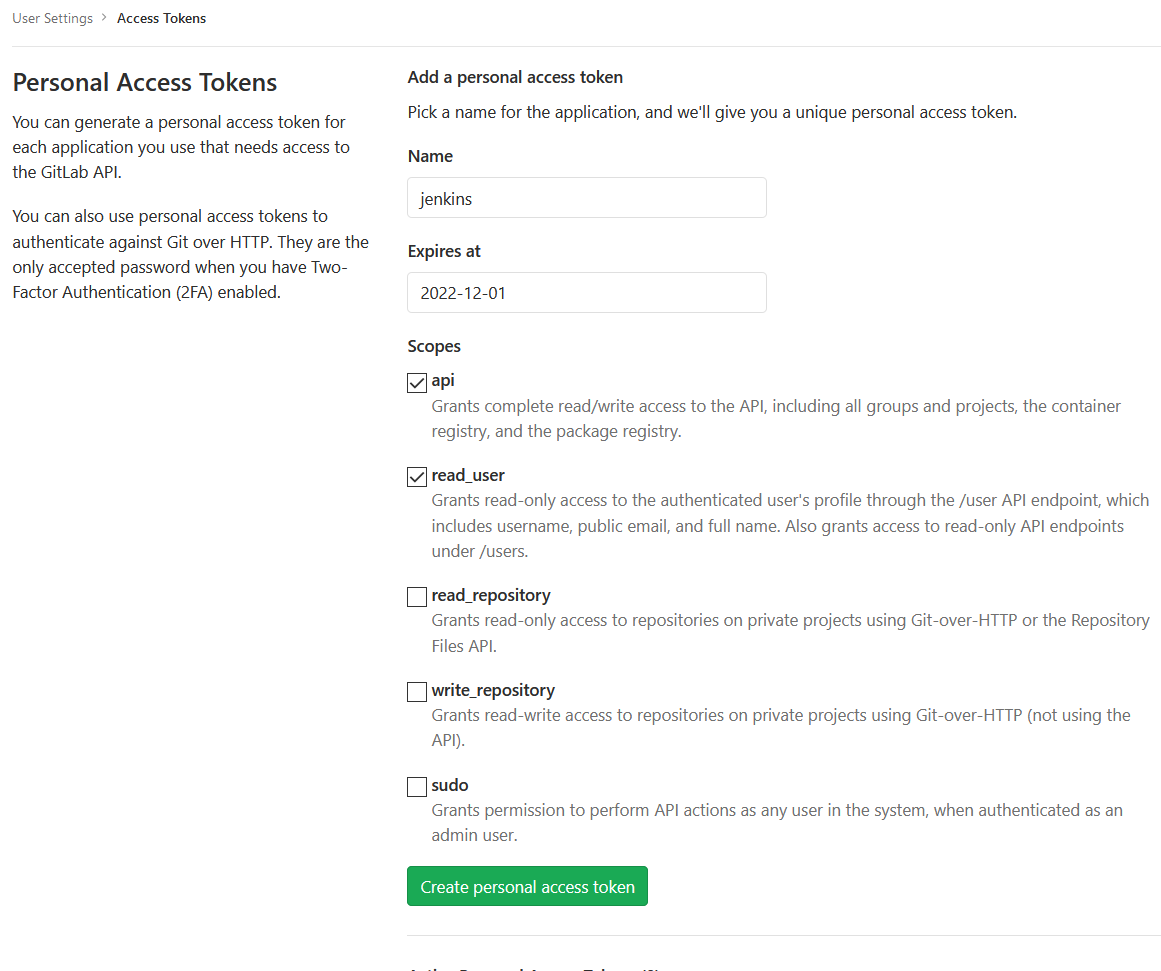
点击Gitlab用户头像图标，如图所示：



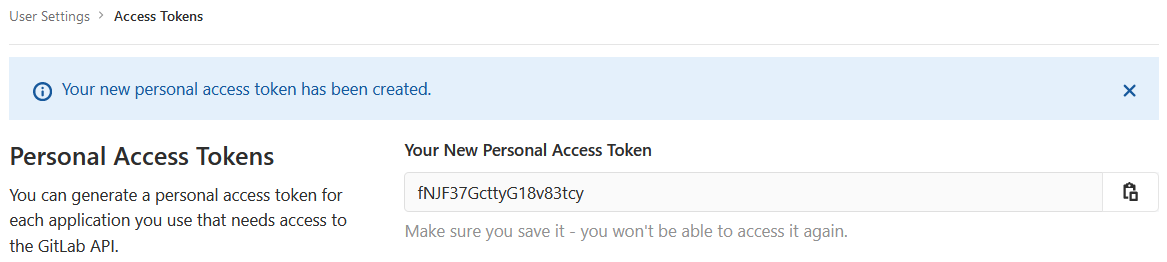
点击“Settings”，如图所示：



点击左侧导航栏的“Access Tokens”添加token，如图所示：



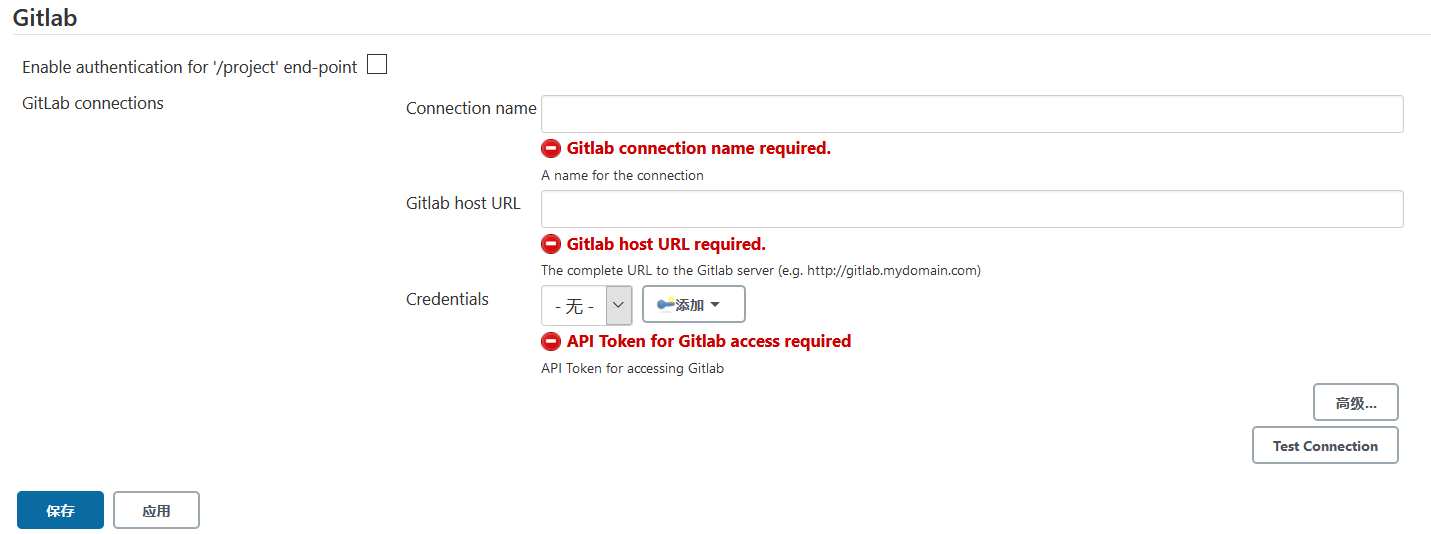
点击“Create personal access token”生成Token，如图所示：



复制Token（fNJF37GcttyG18v83tcy），后面配置Jenkins时会用到。

（3）设置Jenkins

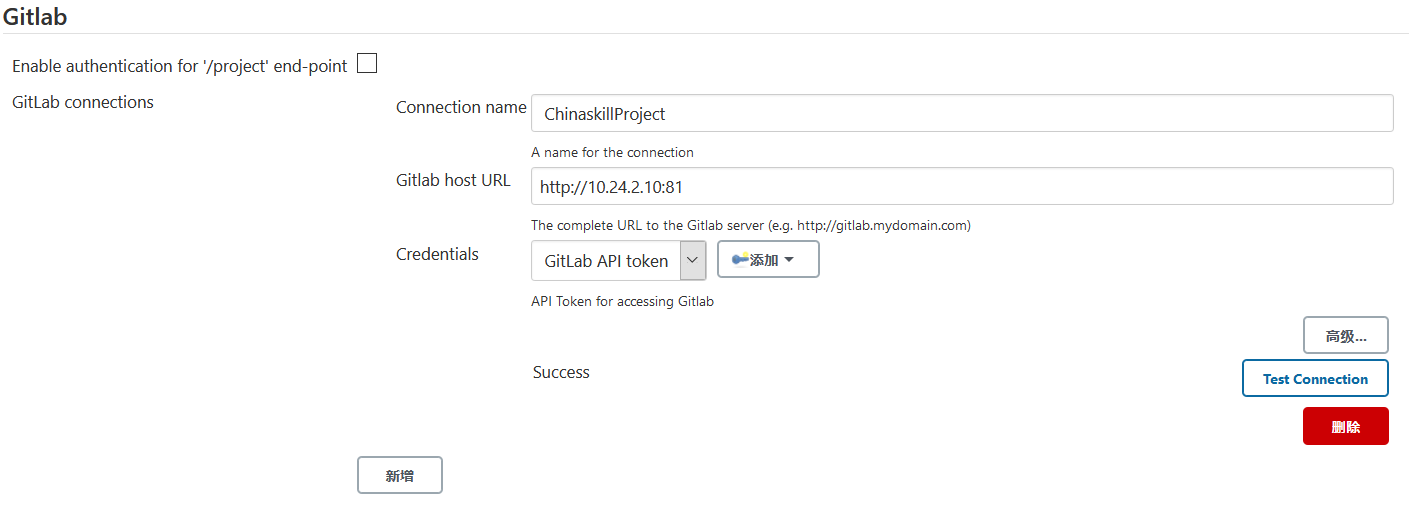
登录Jenkins首页，点击“系统管理”→“系统配置”，配置Gitlab信息，取消勾选“Enable authentication for '/project' end-point”，如图所示：



点击“添加”→“Jenkins”添加认证信息，将Gitlab API Token填入，如图所示：



点击“Test Connection”，如图所示：



**4.配置Jenkins连接maven**

（1）安装maven

由于Jenkins是采用docker in docker的方式启动的，所以需要在jenkins容器内安装maven：

[root@master ~]# cp -rf /opt/apache-maven-3.6.3-bin.tar.gz /home/jenkins\_home/

[root@master ~]# docker exec -it jenkins bash

[root@344d4fa5b8ea:/]# tar -zxvf /var/jenkins\_home/apache-maven-3.6.3-bin.tar.gz -C .

[root@344d4fa5b8ea:/]# mv apache-maven-3.6.3/ /usr/local/maven

[root@344d4fa5b8ea:/]# vi /etc/profile

export M2\_HOME=/usr/local/maven # 行末添加两行

export PATH=$PATH:$M2\_HOME/bin

[root@344d4fa5b8ea /]# vi /root/.bashrc

# .bashrc

# User specific aliases and functions

alias rm='rm -i'

alias cp='cp -i'

alias mv='mv -i'

# Source global definitions

if [ -f /etc/bashrc ]; then

. /etc/bashrc

source /etc/profile # 添加本行

fi

退出容器重新进入：

[root@344d4fa5b8ea /]# mvn -v

Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

Maven home: /usr/local/maven

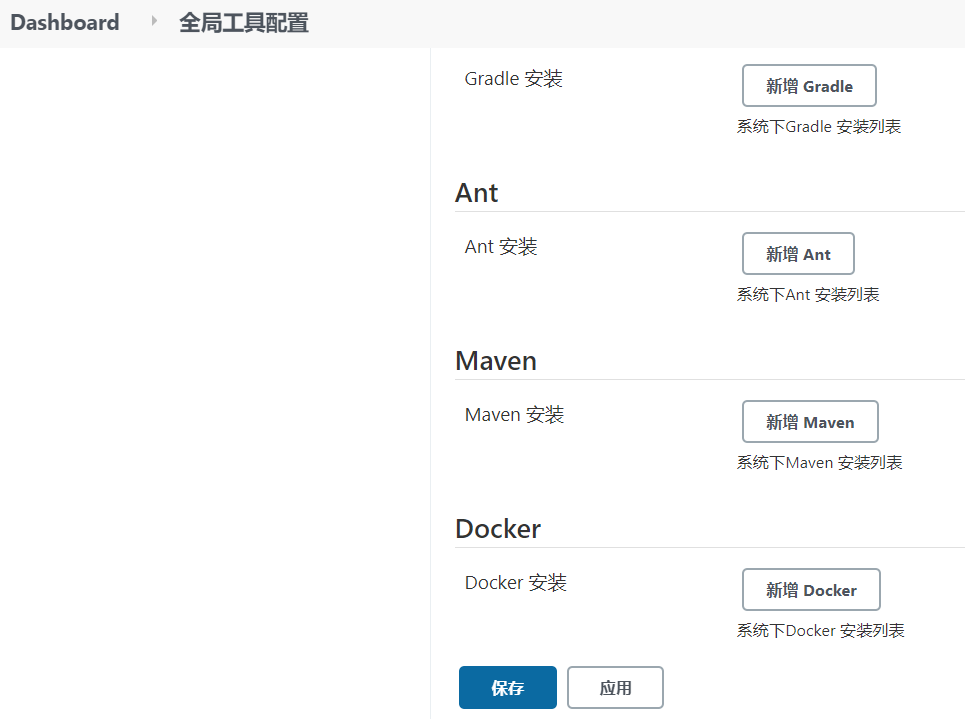
Java version: 1.8.0\_265, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-0.el8\_2.x86\_64/jre

Default locale: en\_US, platform encoding: ANSI\_X3.4-1968

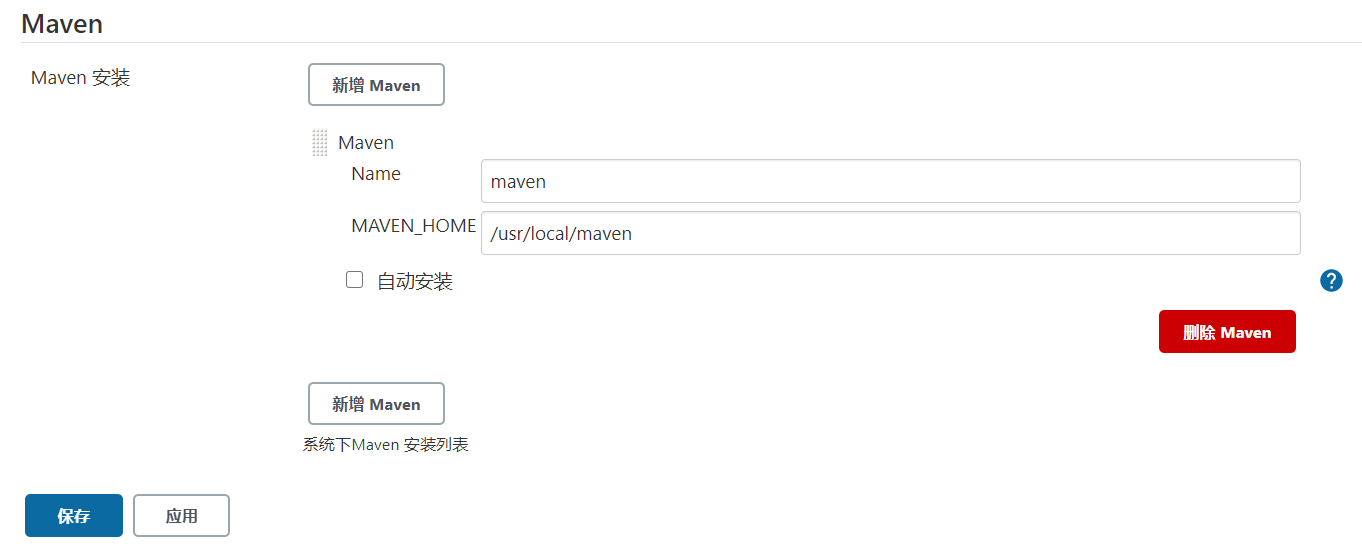
OS name: "linux", version: "3.10.0-862.2.3.el7.x86\_64", arch: "amd64", family: "unix"

（2）连接maven

登录Jenkins首页，点击“系统管理”→“全局工具配置”，如图所示：



点击“新增Maven”，如图所示。取消勾选“自动安装”，填入maven名称和安装路径，配置完成后点击“应用”。



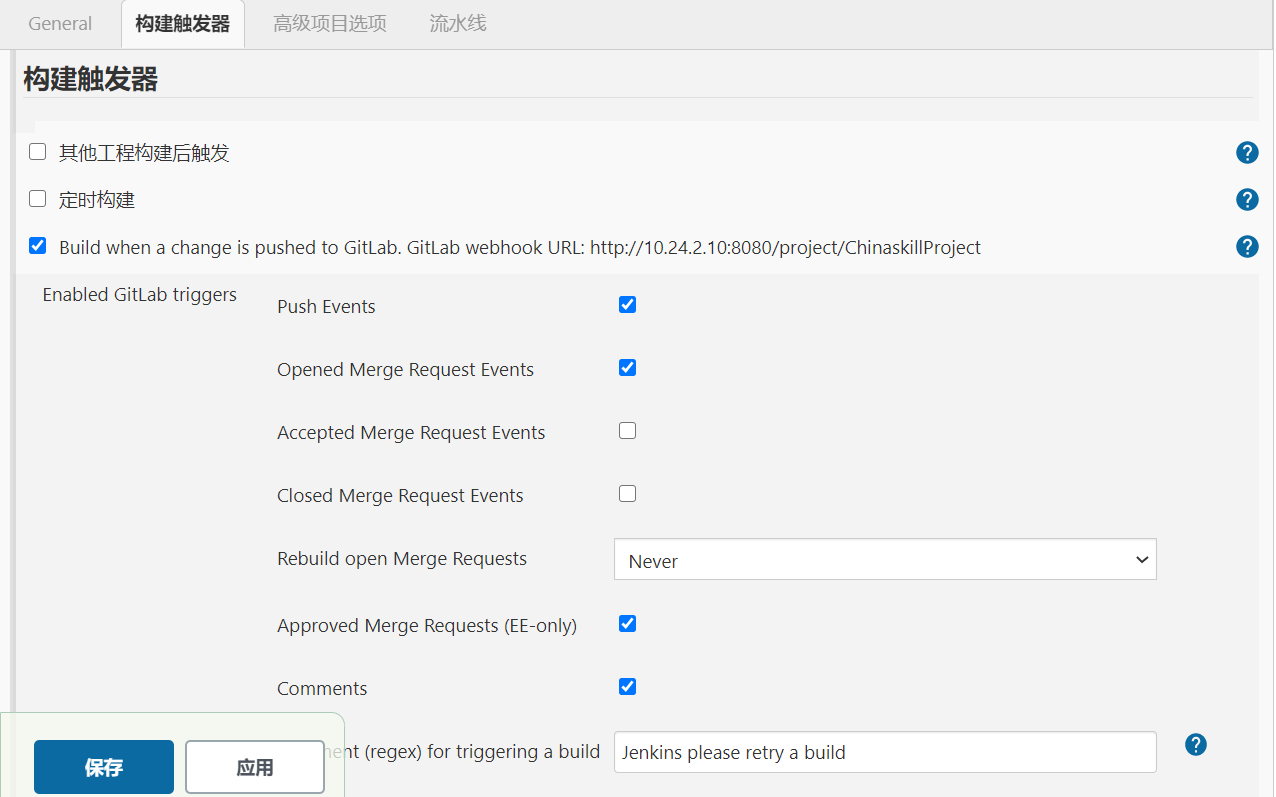
**5. 配置CI/CD**

（1）新建任务

登录Jenkins首页，点击左侧导航栏“新建任务”，如图所示，选择构建一个流水线。

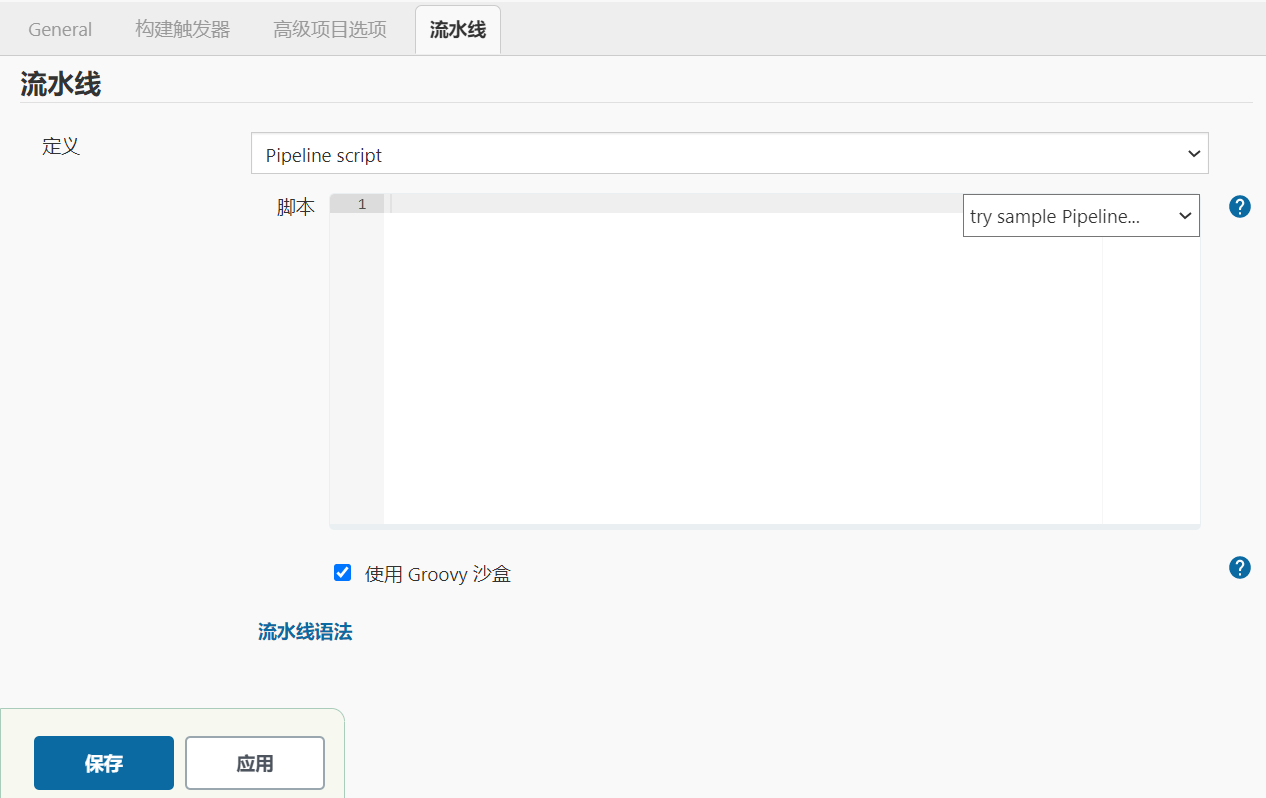


点击“确定”，配置构建触发器，如图所示：

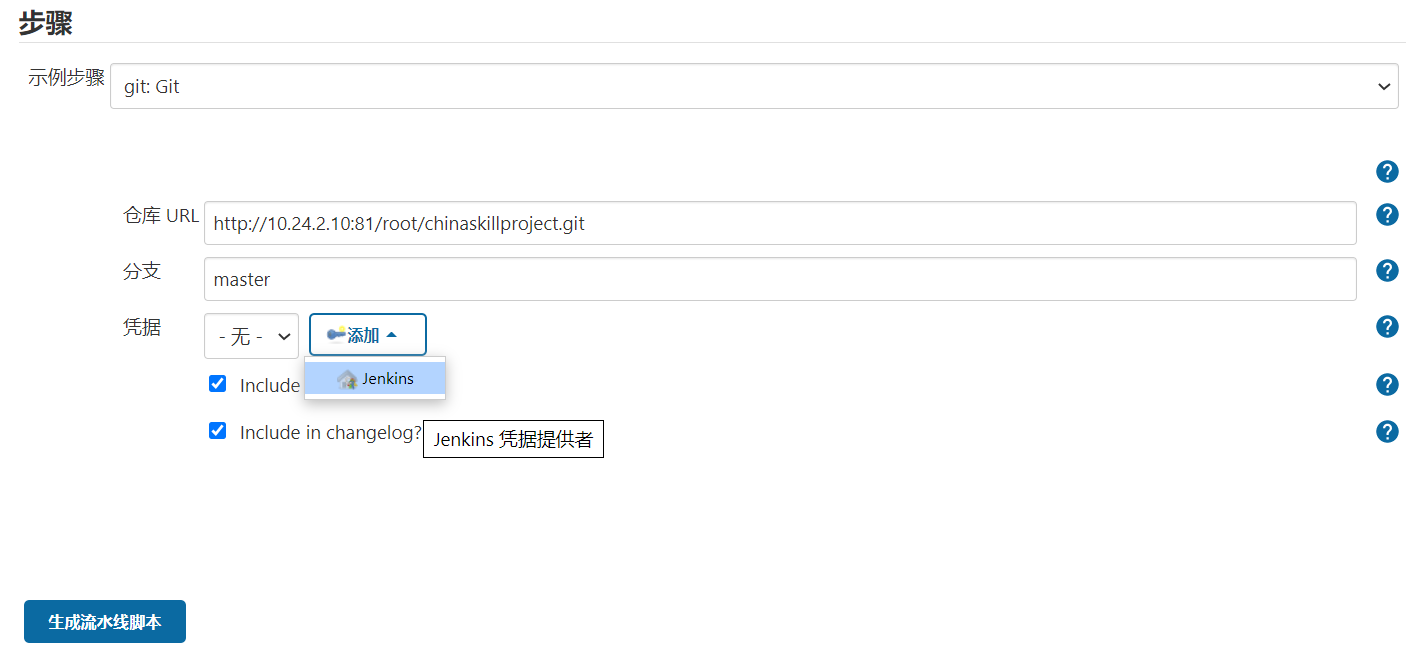


记录下GitLab webhook URL的地址（http://10.24.2.10:8080/project/ChinaskillProject），后期配置webhook需要使用。

配置流水线，如图所示：



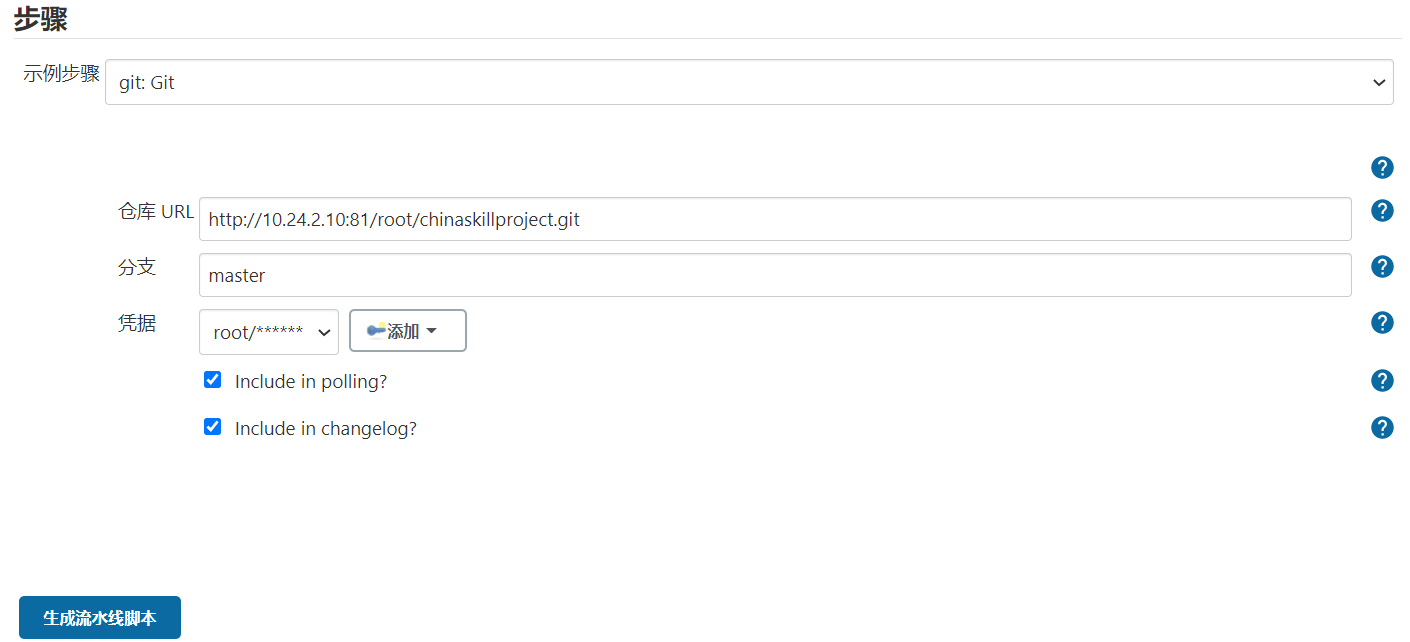
点击“流水线语法”，如图所示，示例步骤选择“git：Git”，将springcloud项目地址填入仓库URL。



点击“添加”→“jenkins”添加凭据，如图所示。类型选择“Username with password”，用户名和密码为Gitlab仓库的用户名和密码。



添加凭据后选择凭据，如图所示：



点击“生成流水线脚本”，如图所示：



记录生成的值，并将其写入流水线脚本中，完整的流水线脚本如下：

node{

stage('git clone'){

//check CODE

git credentialsId: 'ad561b3b-5252-43d5-b9c4-ec9bece0134c', url: 'http://10.24.2.10:81/root/ChinaskillProject.git'

}

stage('maven build'){

sh '''/usr/local/maven/bin/mvn package -DskipTests -f /var/jenkins\_home/workspace/ChinaskillProject'''

}

stage('image build'){

sh '''

echo $BUILD\_ID

docker build -t 10.24.2.10/chinaskillproject/gateway:$BUILD\_ID -f /var/jenkins\_home/workspace/ChinaskillProject/gateway/Dockerfile /var/jenkins\_home/workspace/ChinaskillProject/gateway

docker build -t 10.24.2.10/chinaskillproject/config:$BUILD\_ID -f /var/jenkins\_home/workspace/ChinaskillProject/config/Dockerfile /var/jenkins\_home/workspace/ChinaskillProject/config'''

}

stage('test'){

sh '''docker run -itd --name gateway 10.24.2.10/chinaskillproject/gateway:$BUILD\_ID

docker ps -a|grep chinaskillproject|grep Up

if [ $? -eq 0 ];then

echo "Success!"

docker rm -f gateway

else

docker rm -f gateway

exit 1

fi

'''

}

stage('upload registry'){

sh '''docker login 10.24.2.10 -u=admin -p=Harbor12345

docker push 10.24.2.10/chinaskillproject/gateway:$BUILD\_ID

docker push 10.24.2.10/chinaskillproject/config:$BUILD\_ID'''

}

stage('deploy Rancher'){

//执行部署脚本

sh 'sed -i "s/sqshq\\/piggymetrics-gateway/10.24.2.10\\/chinaskillproject\\/gateway:$BUILD\_ID/g" /var/jenkins\_home/workspace/ChinaskillProject/yaml/deployment/gateway-deployment.yaml'

sh 'sed -i "s/sqshq\\/piggymetrics-config/10.24.2.10\\/chinaskillproject\\/config:$BUILD\_ID/g" /var/jenkins\_home/workspace/ChinaskillProject/yaml/deployment/config-deployment.yaml'

sh 'kubectl create ns springcloud'

sh 'kubectl apply -f /var/jenkins\_home/workspace/ChinaskillProject/yaml/deployment/gateway-deployment.yaml --kubeconfig=/root/.kube/config'

sh 'kubectl apply -f /var/jenkins\_home/workspace/ChinaskillProject/yaml/deployment/config-deployment.yaml --kubeconfig=/root/.kube/config'

sh 'kubectl apply -f /var/jenkins\_home/workspace/ChinaskillProject/yaml/svc/gateway-svc.yaml --kubeconfig=/root/.kube/config'

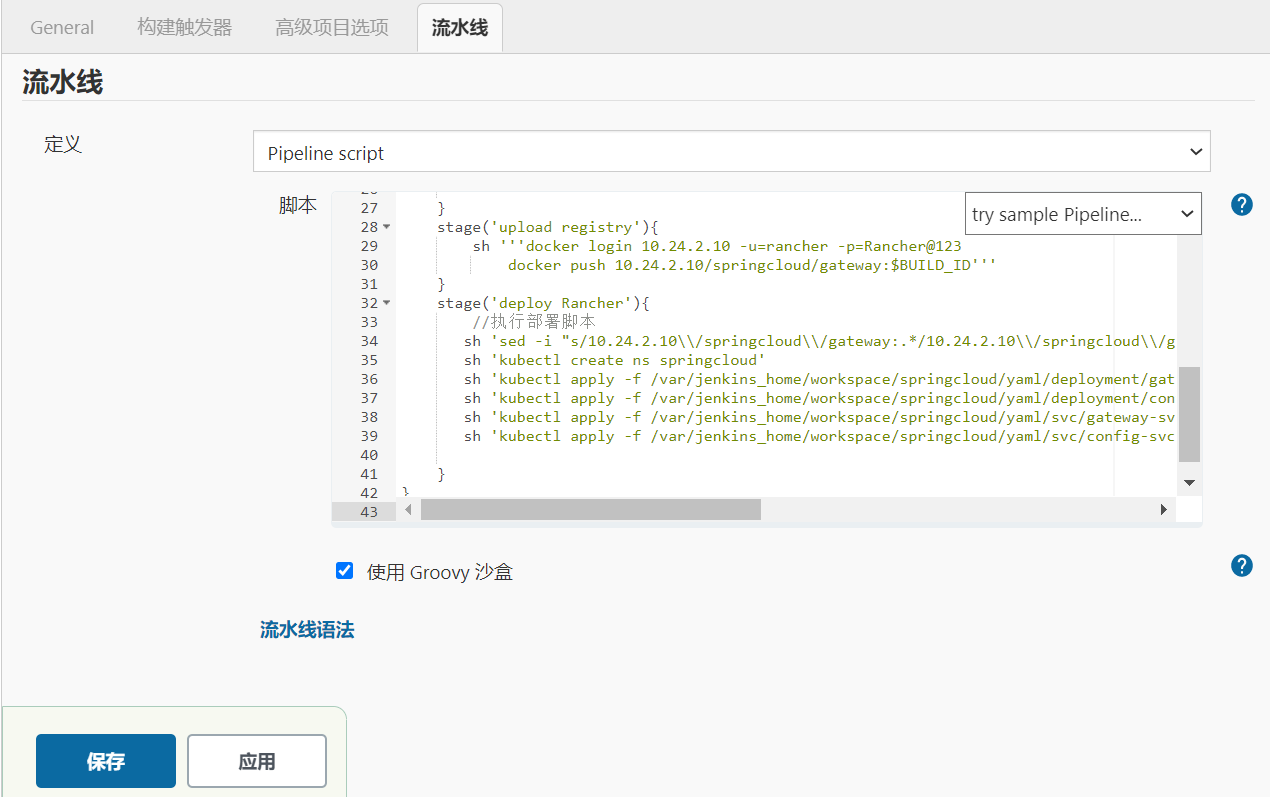
sh 'kubectl apply -f /var/jenkins\_home/workspace/ChinaskillProject/yaml/svc/config-svc.yaml --kubeconfig=/root/.kube/config'

}

}

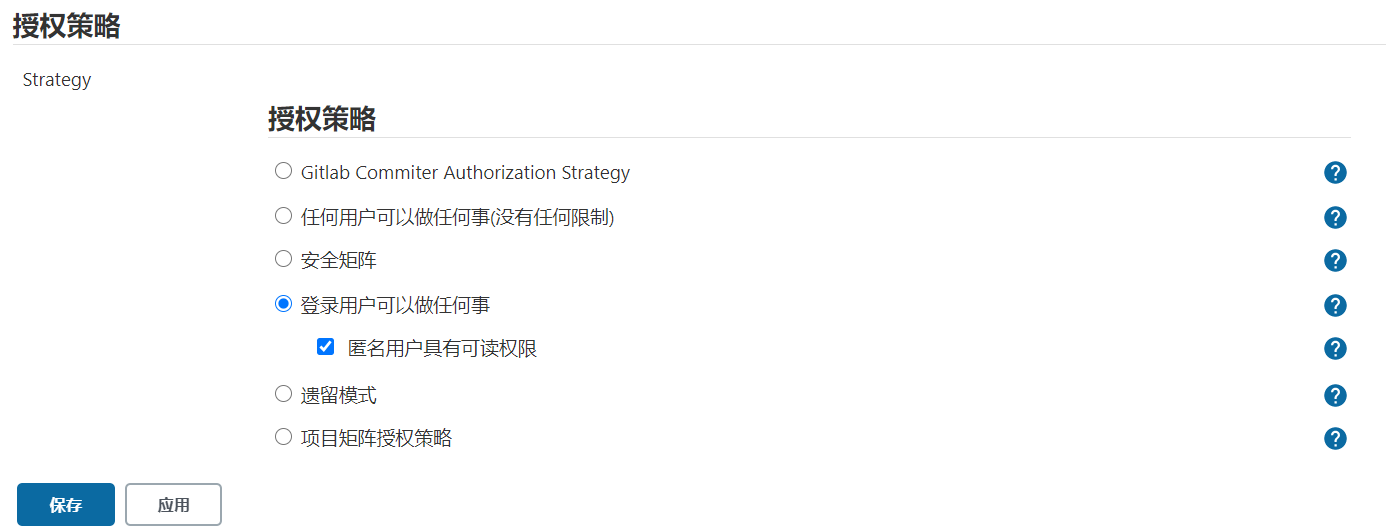
脚本中所有IP均为Harbor仓库的地址。

在网页写入完整的流水线脚本，如图所示，完成后点击“应用”。



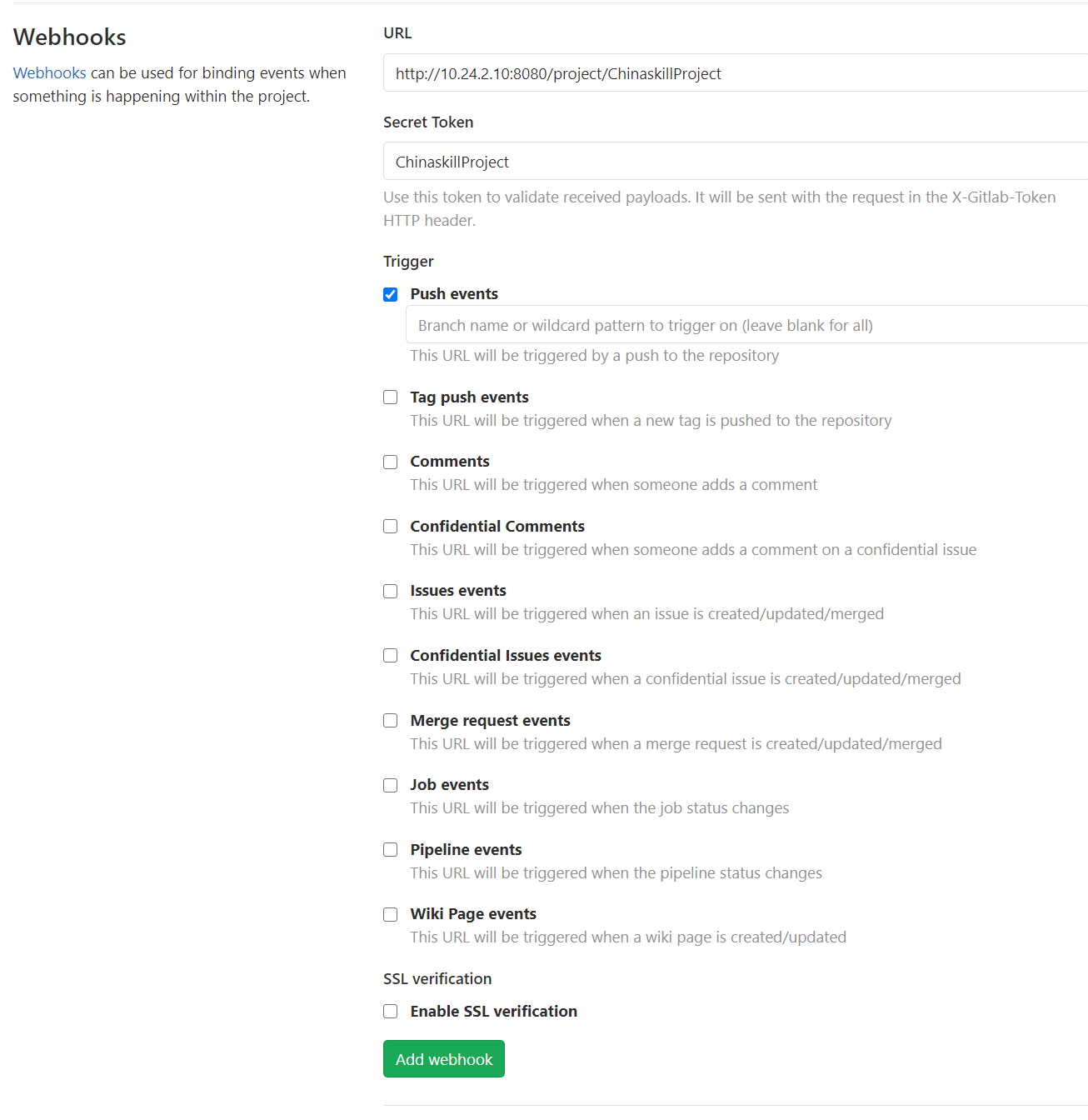
（2）开启Jenkins匿名访问

登录Jenkins首页，点击“系统管理”→“全局安全配置”，配置授权策略允许匿名用户访问，如图所示。

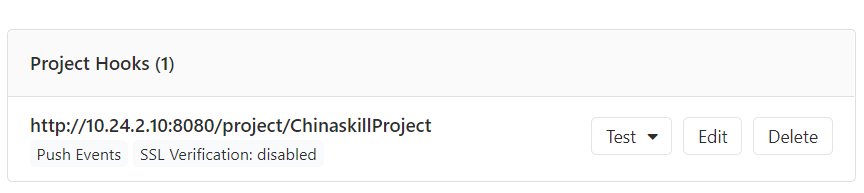


（3）配置Webhook

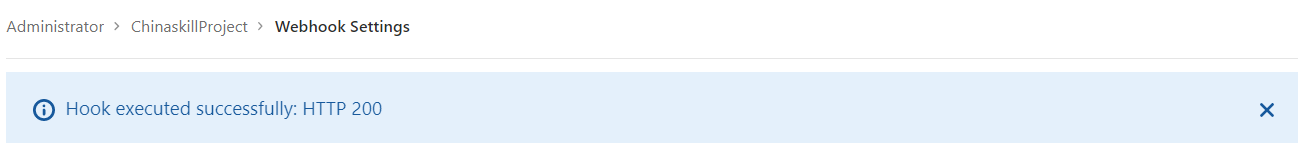
登录Gitlab，进入springcloud项目，点击左侧导航栏“Settings”→“Webhooks”，将前面记录的GitLab webhook URL地址填入URL处，禁用SSL认证，如图所示。



点击“Add webhook”添加webhook，完成后如图所示：



点击“Test”→“Push events”进行测试，如图所示：



结果返回HTTP 200则表明Webhook配置成功。

（4）创建仓库项目

登录Harbor，新建项目chinaskillproject，访问级别设置为公开，创建完成后如图所示：



进入项目查看镜像列表，如图所示，此时为空，无任何镜像：



**6. 触发CI/CD**

（1）触发构建

上传代码触发自动构建：

[root@master ~]# docker cp /opt/repository/ jenkins:/root/.m2/

[root@master ~]# cd /opt/ChinaskillProject/

[root@master ChinaskillProject]# git add .

[root@master ChinaskillProject]# git commit -m "Initial commit"

# On branch master

nothing to commit, working directory clean

[root@master ChinaskillProject]# git push -u origin master

Username for 'http://10.24.2.10:81': root

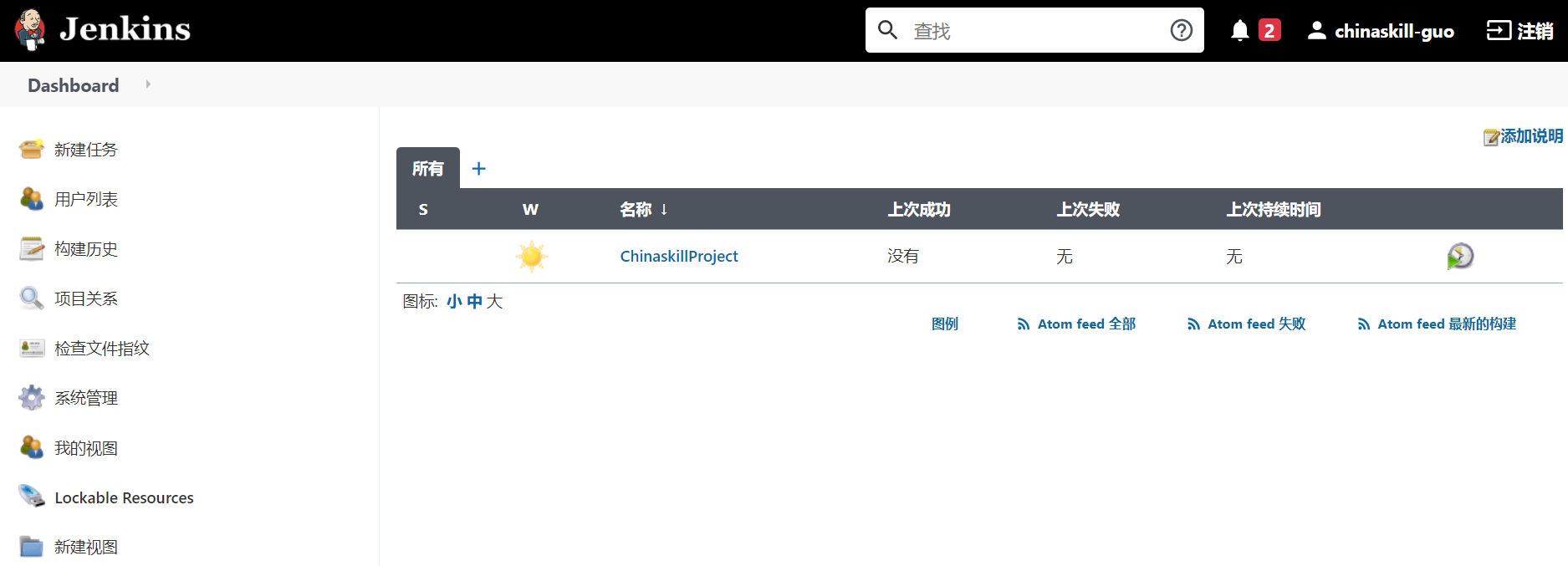
Password for 'http://root@10.24.2.10:81':

Branch master set up to track remote branch master from origin.

Everything up-to-date

（2）Jenkins查看

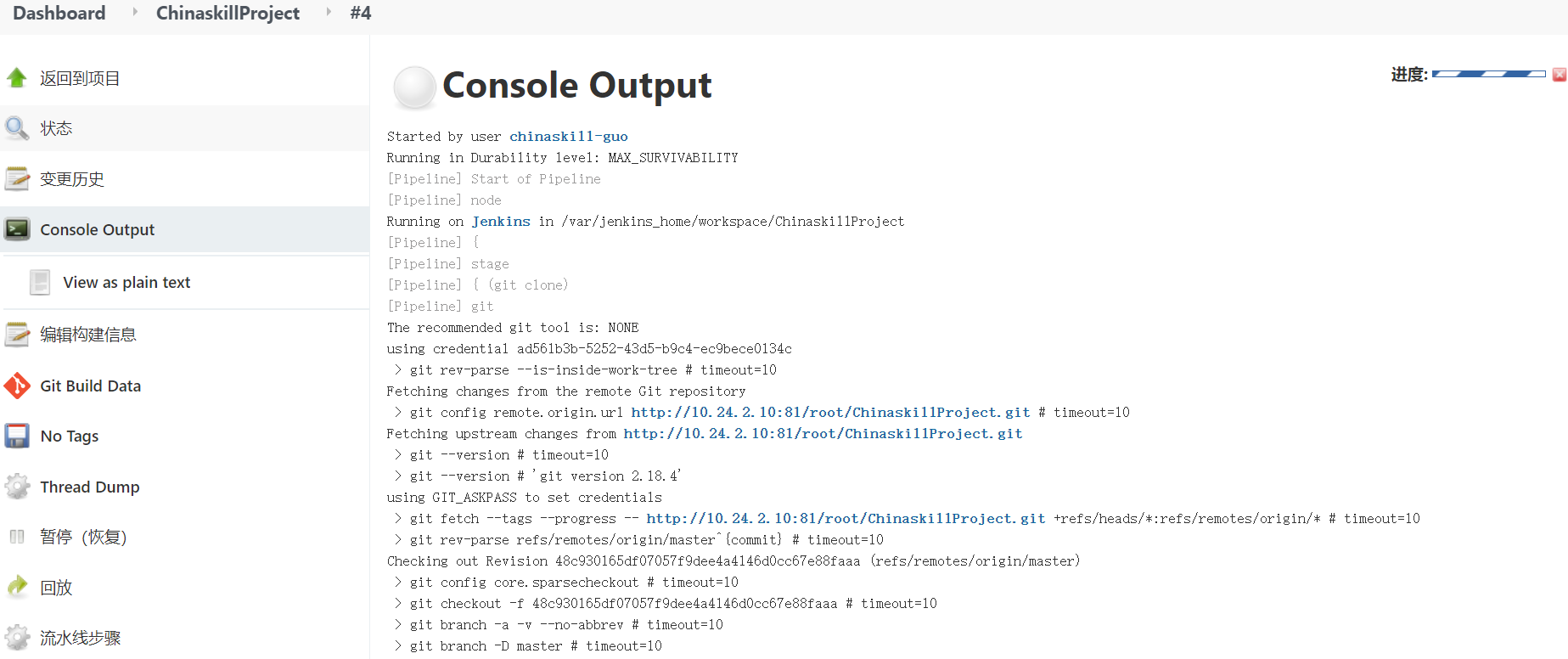
登录Jenkins，可以看到springcloud项目已经开始构建，如图所示：



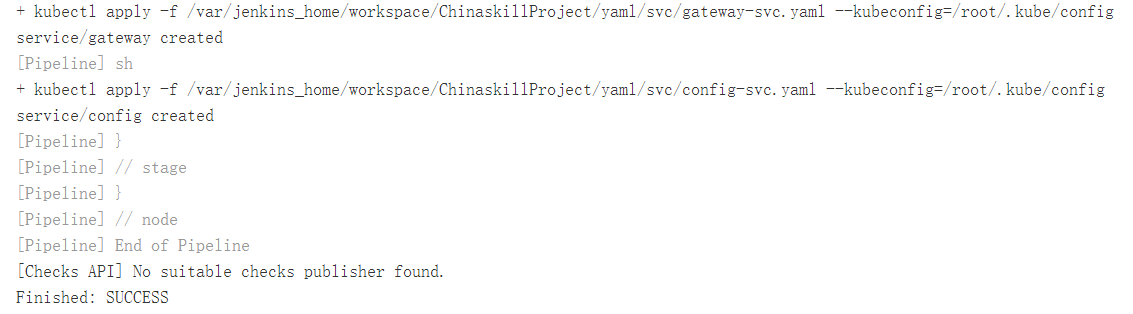
点击项目名称查看流水线阶段视图，如图所示：



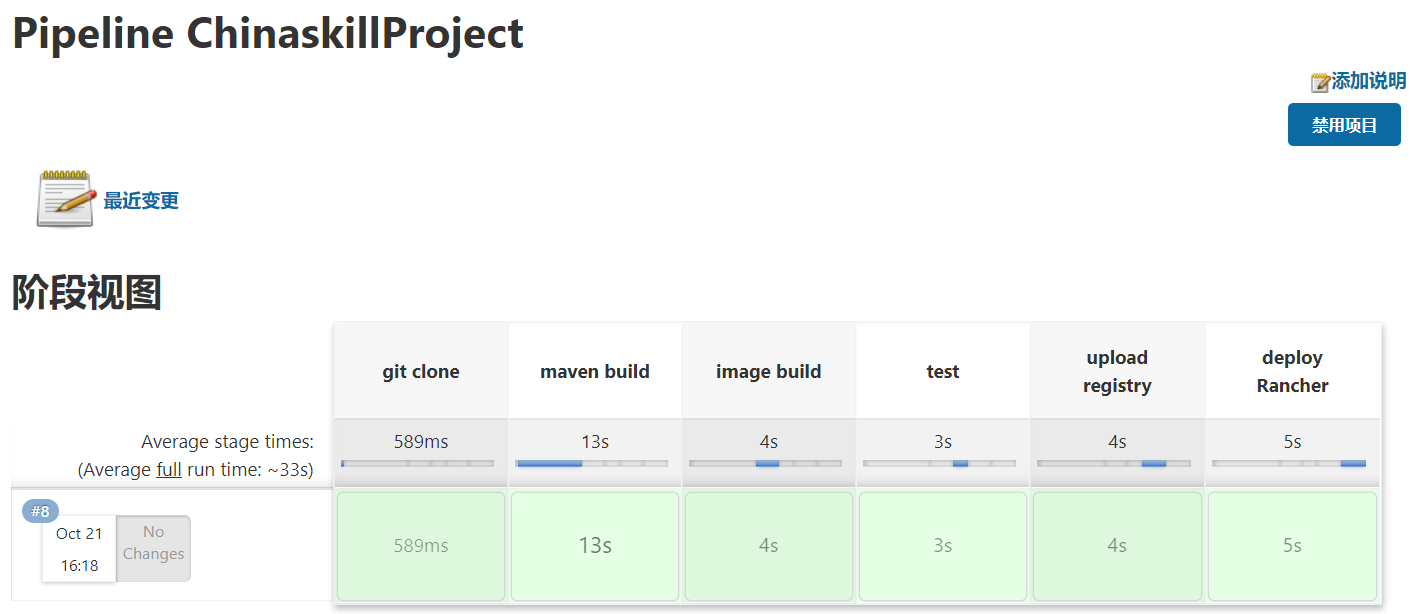
点击右侧“#1”可查看控制台输出，此处会显示构建的详细进程，如图所示：



构建完成后控制台输出如图所示：



返回项目查看流水线阶段视图，如图所示：



（3）Harbor查看

进入Harbor仓库springcloud项目查看镜像列表，可以看到已自动上传了一个gateway镜像，如图所示：



（4）Kubernetes查看

Pod的启动较慢，需等待3--5分钟。在命令行查看Pod：

[root@master ~]# kubectl -n springcloud get pods

NAME READY STATUS RESTARTS AGE

config-6c988c4dc5-2522c 1/1 Running 0 21m

gateway-6545fc58c5-d6rgn 1/1 Running 0 21m

查看service：

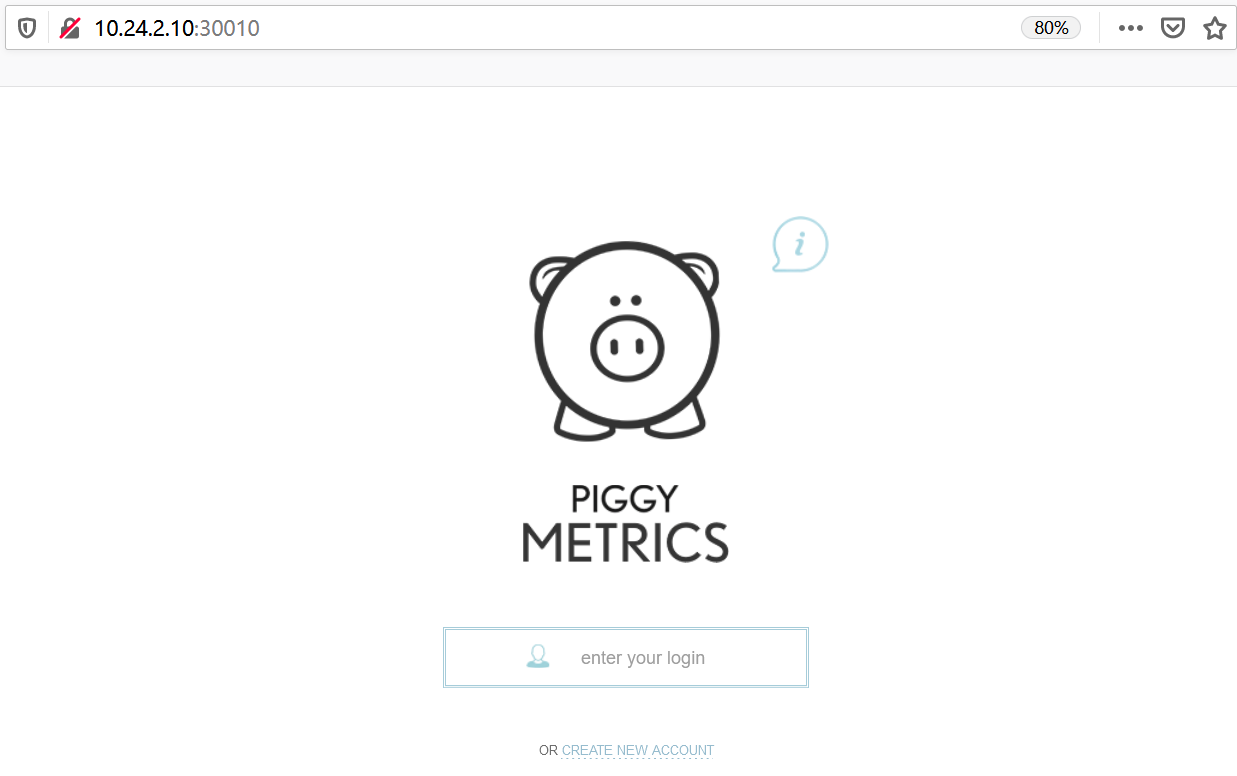
[root@master ~]# kubectl -n springcloud get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

config NodePort 10.101.42.47 <none> 8888:30015/TCP 22m

gateway NodePort 10.100.62.39 <none> 4000:30010/TCP 22m

通过端口30010访问服务，如图所示：



至此，完整的CI/CD流程就完成了。