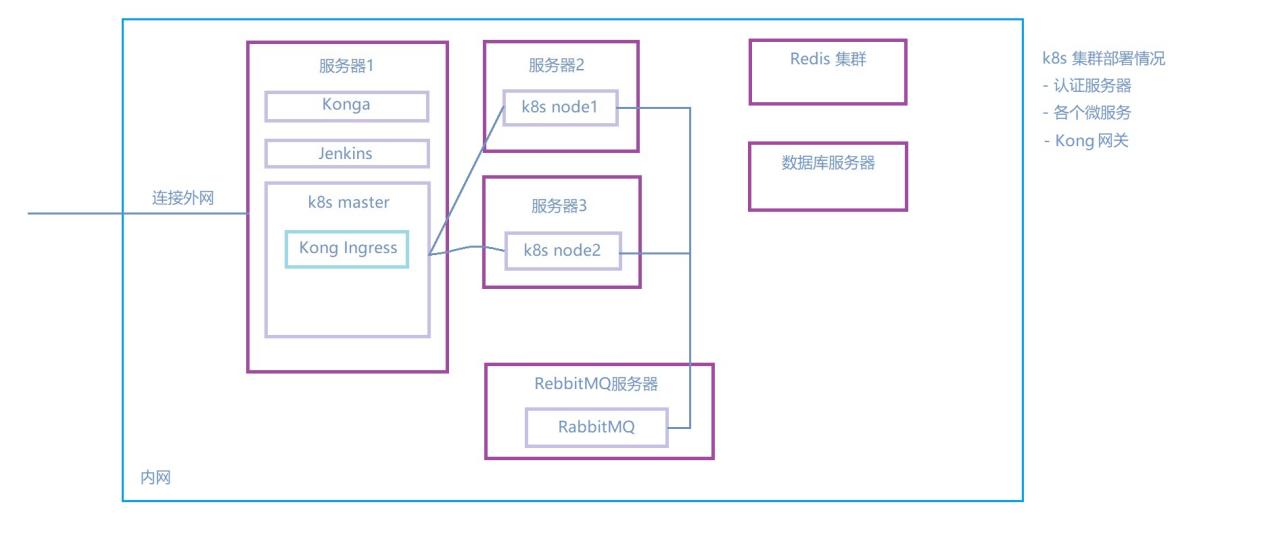
现在我们要搭建一个简单的微服务物理框架，其架构如下



**前置知识**

1. Kong

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/%E5%B9%B3%E5%8F%B0%E6%97%A0%E5%85%B3/Kong%20%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0>

1. K8s

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/%E5%B9%B3%E5%8F%B0%E6%97%A0%E5%85%B3/kubenetes>

1. Docker

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/%E5%B9%B3%E5%8F%B0%E6%97%A0%E5%85%B3/Docker>

1. RabbitMQ

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/%E5%B9%B3%E5%8F%B0%E6%97%A0%E5%85%B3/RabbitMQ>

1. Jenkins

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/%E5%B9%B3%E5%8F%B0%E6%97%A0%E5%85%B3/Jenkins>

1. Linux

<https://github.com/IceEmblem/LearningDocuments/tree/master/%E5%AD%A6%E4%B9%A0%E8%B5%84%E6%96%99/Linux%20%E5%B9%B3%E5%8F%B0/Linux%20%E6%95%99%E7%A8%8B>

这些知识在Github项目 -> 平台无关 目前下可以找到

**环境**

我们的示例不需要用到redis和数据库，所以这2个不在当前示例中

我们需要4台服务器

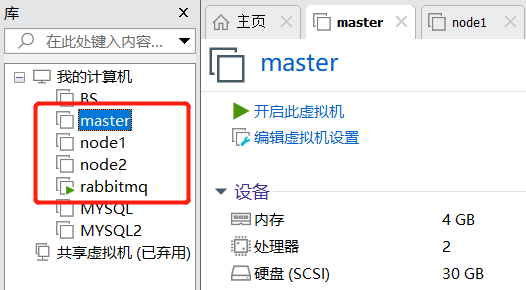
- master 服务器

- node1 服务器

- node2 服务器

- RabbitMQ服务器

其中master、node1、node2 构成k8s集群，RabbitMQ服务器作为独立的服务器



服务器系统：Ubuntu Server 20.04

注：以下环境搭建请使用root用户

**K8s集群环境搭建**

下面我们针对master、node1、node2搭建k8s的集群环境，详细的文档请转到 kubenetes目前

**安装docker**

安装请查看docker目录教程，安装完成后做如下操作

1. 编辑vim /etc/docker/daemon.json，没有就新建，添加exec-opts

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

1. 重启docker，systemctl restart docker

注：master、node1、node2都要执行

**安装kubeadm、kubelet、kubectl**

1. 安装前置

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

1. 使用阿里库（google库国内访问不了）

- 添加key

curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -

- 添加镜像源库，在vim /etc/apt/sources.list.d/kubernetes.list中添加如下行

deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main

1. 安装

apt-get update

apt-get install -y kubelet kubeadm kubectl

笔者使用的版本是1.23.5，即如下

apt-get install -y kubelet=1.23.5-00 kubeadm=1.23.5-00 kubectl=1.23.5-00

1. 设置kubelet开机自启

systemctl enable kubelet

注：master、node1、node2都要执行

**在master部署kubernetes**

1. 部署

kubeadm init \

--apiserver-advertise-address=192.168.102.130 \

--image-repository registry.aliyuncs.com/google\_containers \

--kubernetes-version v1.23.5 \

--service-cidr=10.1.0.0/16 \

--pod-network-cidr=10.244.0.0/16

1. 部署成功信息

注意，部署成功后会有如下的信息反馈

...

...

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.102.130:6443 --token qc093z.xr3ltsui1loy87cs \

  --discovery-token-ca-cert-hash sha256:902969f599d324185126f53f0b07c02126063fea73f14e05297b65f102e842a0

其中最后一行是在node节点加入集群的命令

kubeadm join 192.168.102.130:6443 --token qc093z.xr3ltsui1loy87cs \

--discovery-token-ca-cert-hash sha256:902969f599d324185126f53f0b07c02126063fea73f14e05297b65f102e842a0

**在master安装pod网络插件**

在master上执行如下命令

# kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

查看插件是否运行

root@k8s-master:~# kubectl get pod --all-namespaces -o wide

NAMESPACE NAME READY STATUS RESTARTS AGE

kube-system kube-flannel-ds-p7pf7 1/1 Running 0 61m

**在node1和node2上执行加入集群命令**

1. Node节点加入集群

如下，我们在node节点执行kubeadm init执行成功后的反馈命令

# kubeadm join 192.168.102.130:6443 --token qc093z.xr3ltsui1loy87cs \

--discovery-token-ca-cert-hash sha256:902969f599d324185126f53f0b07c02126063fea73f14e05297b65f102e842a0

执行该命令后node节点会加入到集群中

1. 查看节点是否加入

在master上运行如下名

root@k8s-master:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master Ready control-plane,master 138m v1.23.5

k8s-node1 Ready <none> 106s v1.23.5

可以看到k8s-node1已加入集群

**安装Kong**

现在我们安装kong与konga，详细文档在Github的kong教程目录下

**在k8s集群中安装kong**

执行如下命令安装kong-ingress

# kubectl apply -f https://bit.ly/k4k8s

使用如下命令查看其创建的service

root@k8s-master:~# kubectl get services -n kong

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

kong-proxy LoadBalancer 10.1.85.159 <pending> 80:32268/TCP,443:32148/TCP 10m

kong-validation-webhook ClusterIP 10.1.93.55 <none> 443/TCP 10m

可以看到我们的kong已经启动，其对应的master服务器的端口为32268，32148

如果想把端口映射到80端口可以执行如下命令，我这里就不执行了

# nohup kubectl port-forward -n kong --address 0.0.0.0 service/kong-proxy 80:80 > nohup.out 2>&1 &

**在master服务器上安装konga**

这不是必须的，但图像界面管理起来比较方便

1. 更改kong-ingress暴露的端口

kong-ingress没有暴露kong admin api，我们需要在集群内部暴露api

所以我们需要更改其Deployment：

# kubectl edit deployments/ingress-kong -n kong

将

- name: KONG\_ADMIN\_LISTEN

value: 127.0.0.1:8444 ssl

修改为

- name: KONG\_ADMIN\_LISTEN

value: 0.0.0.0:8001, 0.0.0.0:8444 ssl

然后我们新建一个konga.yaml文件

kind: Service # 资源类型

apiVersion: v1 # 资源版本

metadata: # 元数据

name: konga # 资源名称

namespace: kong # 命名空间

spec: # 描述

selector: # 标签选择器，用于确定当前service代理哪些pod

app: ingress-kong

type: ClusterIP # 类型，值NodePort允许外部访问，值 ClusterIP 只能在集群内访问

ports: # 端口信息

- name: proxy-admin

port: 8001 # service端口

targetPort: 8001 # pod端口

- name: proxy-admin-ssl

port: 8444 # service端口

targetPort: 8444 # pod端口

执行如下命令应用service

# kubectl apply -f konga.yaml

查看我们konga service的地址

root@k8s-master:~# kubectl get services -A

NAMESPACE NAME TYPE CLUSTER-IP PORT(S)

kong konga ClusterIP 10.1.184.146 8001/TCP,8444/TCP

可以看到地址为10.1.184.146

1. 创建docker网络

# docker network create -d bridge kongnet

1. 安装postgres数据库

# sudo mkdir -p /opt/postgres/data

# sudo docker run -d --name postgres \

--network kongnet \

--restart=always \

-e POSTGRES\_PASSWORD=kong123 \

-p 5432:5432 \

-v /home/software/postgres/data:/var/lib/postgresql/data \

postgres:9.6

1. 进入数据库创建用户和数据库

// 进入容器

# sudo docker exec -it postgres /bin/bash

// 进入数据库

# psql -U postgres

// 创建用户，创建数据库

# CREATE USER kong WITH PASSWORD 'kong123';

# CREATE DATABASE konga OWNER kong;

// 退出数据库，退出容器

# \q

# exit

1. 安装konga

# docker run -d --name konga \

--network kongnet \

--restart=always \

-p 1337:1337 \

-e "DB\_ADAPTER=postgres" \

-e "DB\_HOST=postgres" \

-e "DB\_PORT=5432" \

-e "DB\_USER=kong" \

-e "DB\_PASSWORD=kong123" \

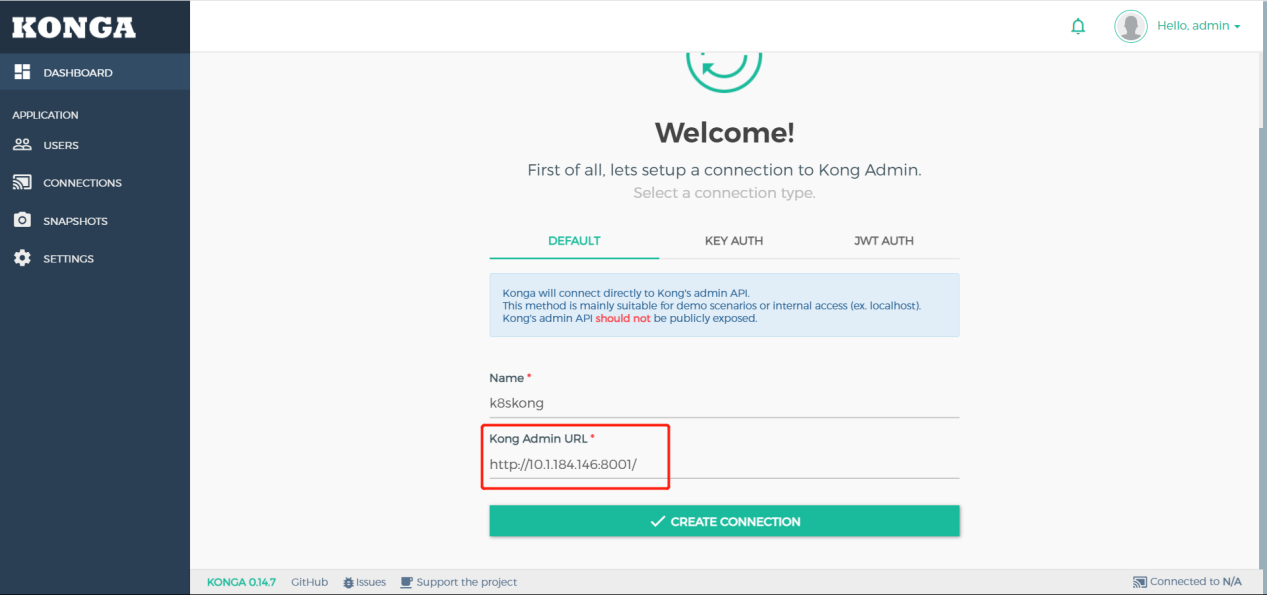
-e "DB\_DATABASE=konga" \

-e "NODE\_ENV=development" \

docker.io/pantsel/konga:0.14.7

1. 访问http://192.168.102.130:1337，（192.168.102.130是我k8s master的地址）

登录进去后连接kong，这里的连接地址是我们前面创建的k8s服务的地址10.1.184.146



**安装jenkins**

接下来我们在master服务器上安装jenkins

**安装**

1. 安装jdk

# sudo apt-get install openjdk-8-jdk

1. 设置密匙

# wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

1. 添加镜像源

# sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \

/etc/apt/sources.list.d/jenkins.list'

1. 安装Jenkins

# sudo apt-get update

# sudo apt-get install jenkins

**配置权限**

有时我们需要使用sudo，但jenkins由于没有权限无法执行某些命令，所以需要改变以下jenkins的权限

1. 添加sudoers编辑权限

# sudo chmod u+w /etc/sudoers

1. 编辑sudoers

# sudo vim /etc/sudoers

1. 在最后加一行

jenkins ALL=(ALL) NOPASSWD: ALL

1. 重启jenkins

# service jenkins restart

**测试**

我们访问http://masterip:8080就可以看到jenkins了

**安装RabbitMQ**

接下来我们在RabbitMQ服务器上安装RabbitMQ，我们使用docker进行安装

**安装docker**

详情情况docker教程目录

**安装rabbitmq**

执行如下命令安装

# docker run -itd --restart=always --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.9-management

安装完成后，访问http://IP:15672/ 进入rabbitmq，默认用户名/密码为 guest/guest

至此，物理框架已搭建完成，后面有时间在示例微服务的集成和部署

**自动部署**

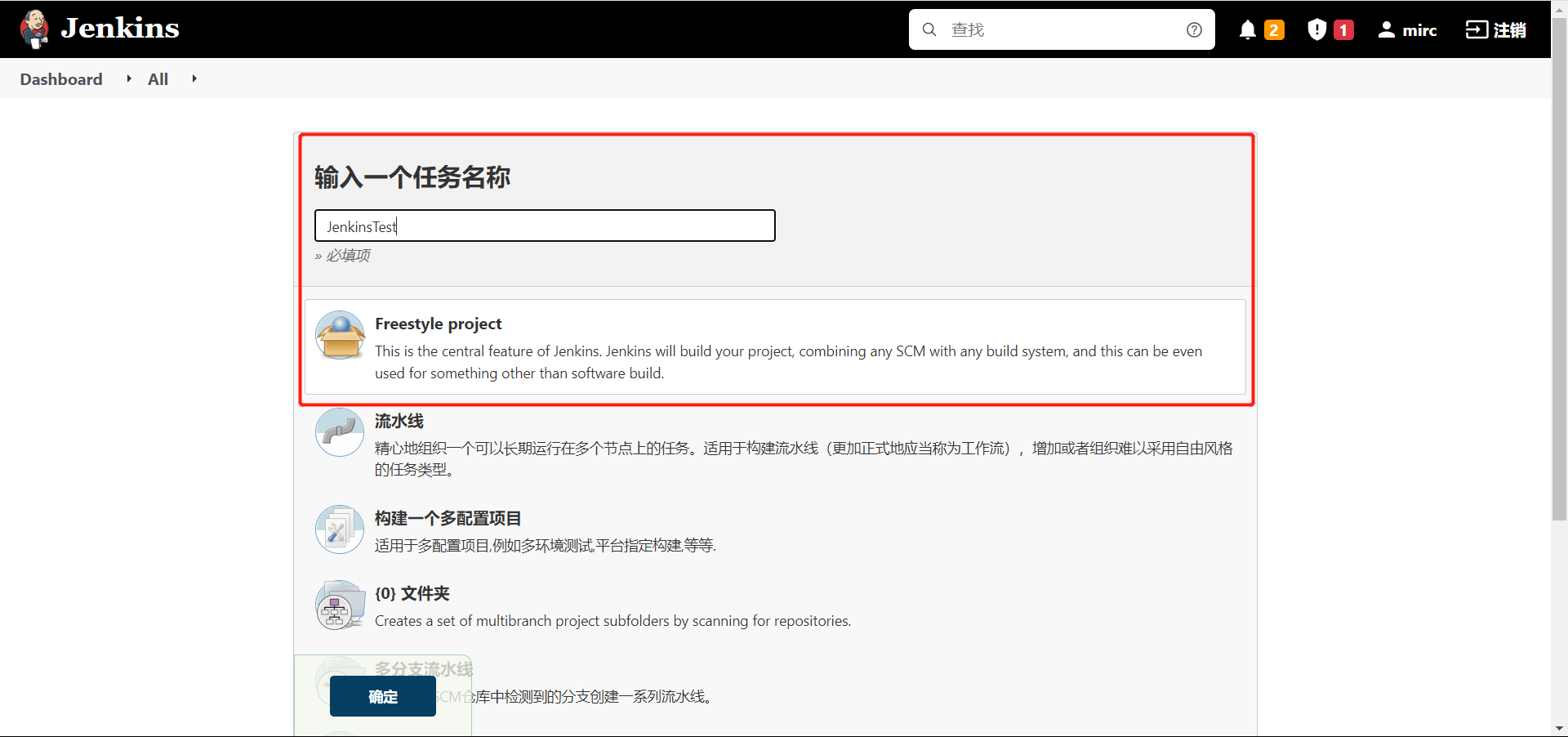
如下我们做一个自动部署的示例

**登录docker**

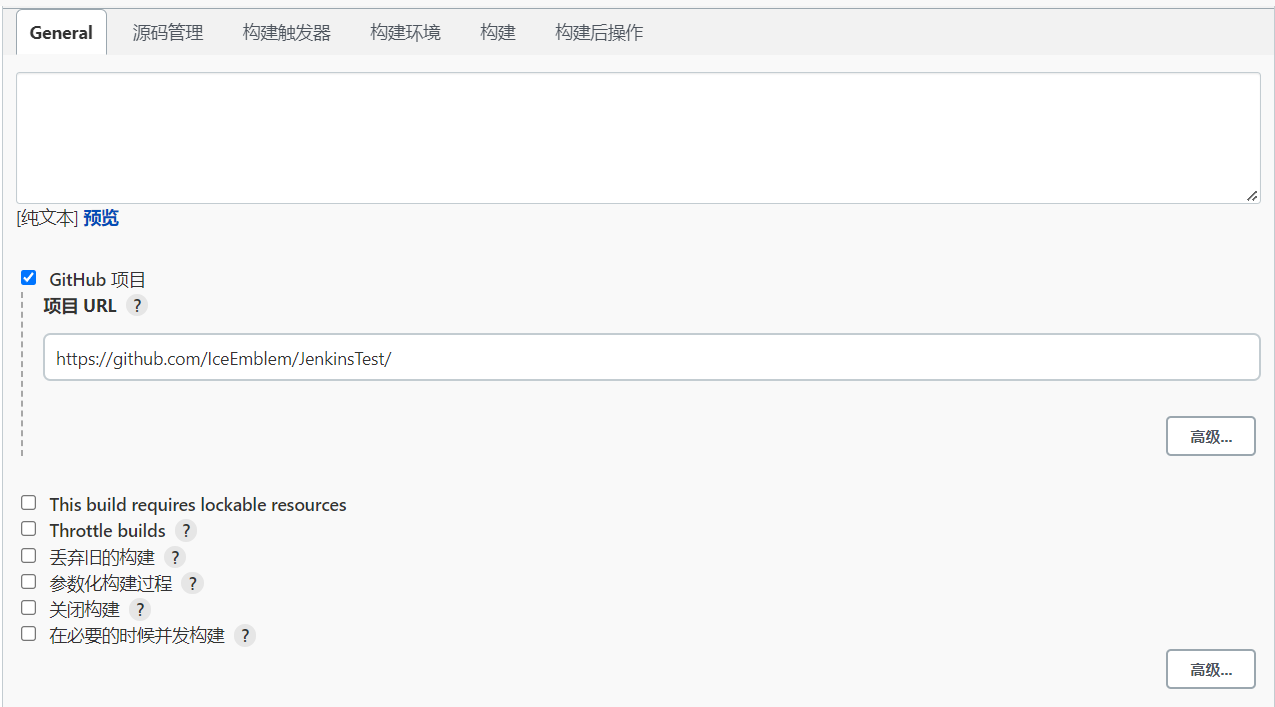
我们使用docker作为镜像库，需要在master、node1、node2上都执行 docker login 登录命令

**创建jenkins任务**

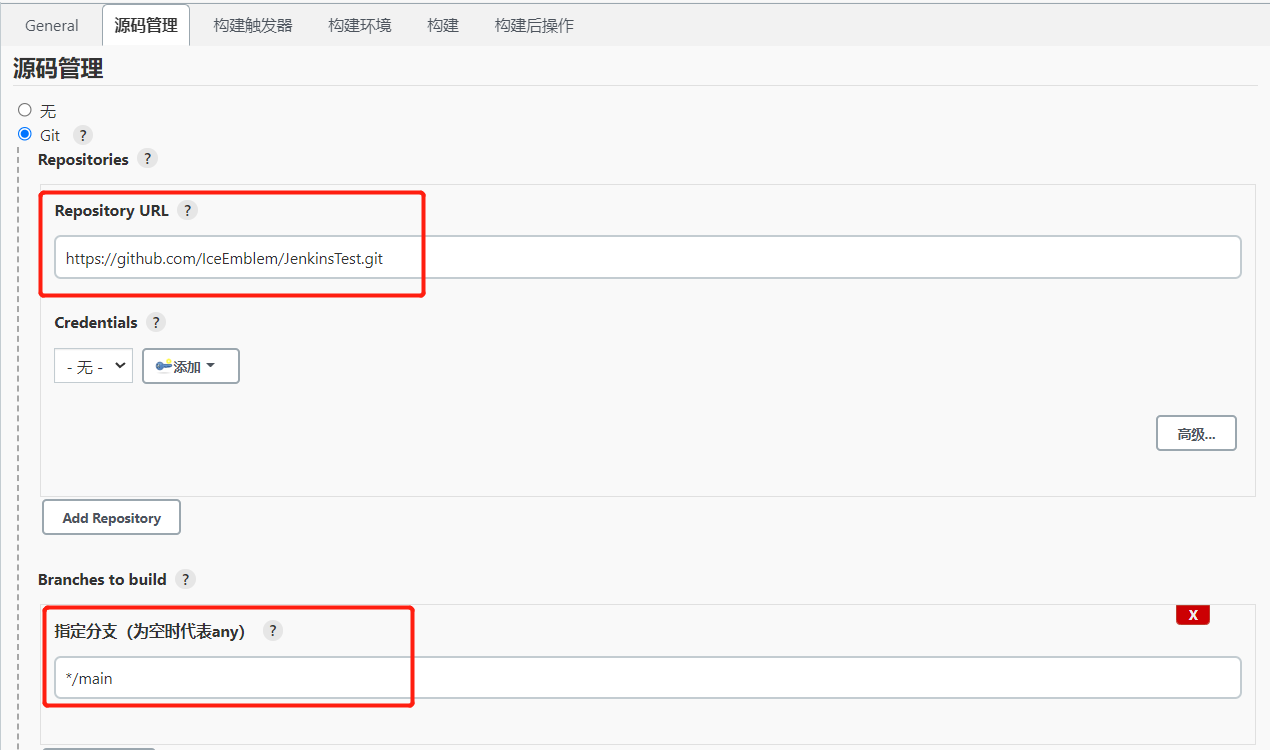
1. 进入jenkins创建一个Freestyle project风格的项目



1. 任务描述



1. 指定仓库地址和分支



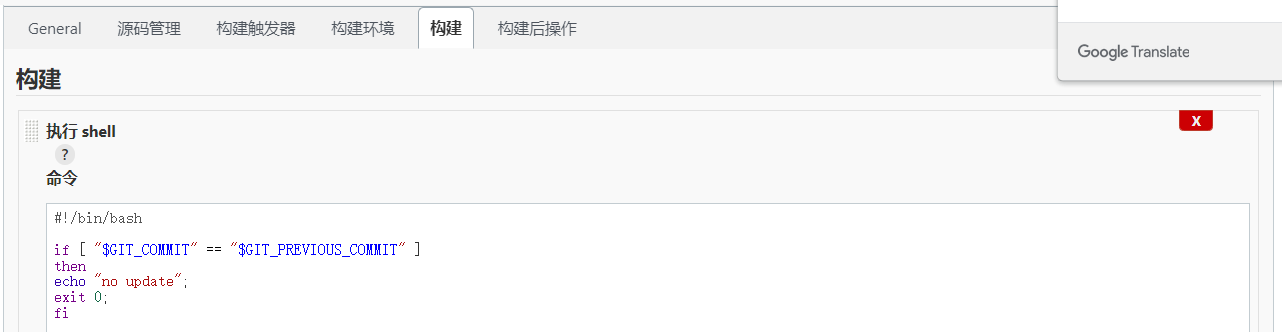
1. 指定触发条件，这里是没60分支执行一次（正常情况我们是2周发布一次）



1. 指定构建环境，根据个人喜好勾选



1. 指定构建动作，这里我使用的是Shell



其命令如下，步骤大概是

#!/bin/bash

# 如果没有更新则不发布

if [ "$GIT\_COMMIT" == "$GIT\_PREVIOUS\_COMMIT" ]

then

echo "no update";

exit 0;

fi

# 生成镜像名

echo "has update";

imagename="iceemblem/mircservice:$BUILD\_NUMBER";

echo $imagename;

# 生成镜像

sudo docker build -t $imagename .

# 发布镜像到 dockerhub

sudo docker image push $imagename

# 执行 k8s 部署命令

echo "

apiVersion: apps/v1

kind: Deployment

metadata:

name: mircservice

spec:

# 副本数量

replicas: 2

selector:

matchLabels:

app: mircservice

template:

# pods 的标签

metadata:

labels:

app: mircservice

# 使用的镜像

spec:

containers:

- name: mircservice

image: $imagename

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: mircservice-svc

spec:

ports:

- port: 80

name: 80port

protocol: TCP

targetPort: 80

selector:

app: mircservice

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: mircservice-ingress

spec:

ingressClassName: kong

rules:

- http:

paths:

- path: /api/mircservice

pathType: ImplementationSpecific

backend:

service:

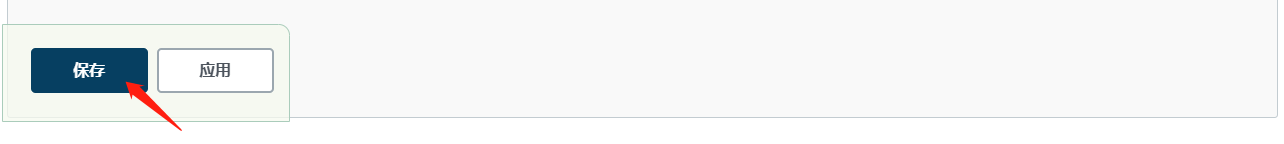
name: mircservice-svc

port:

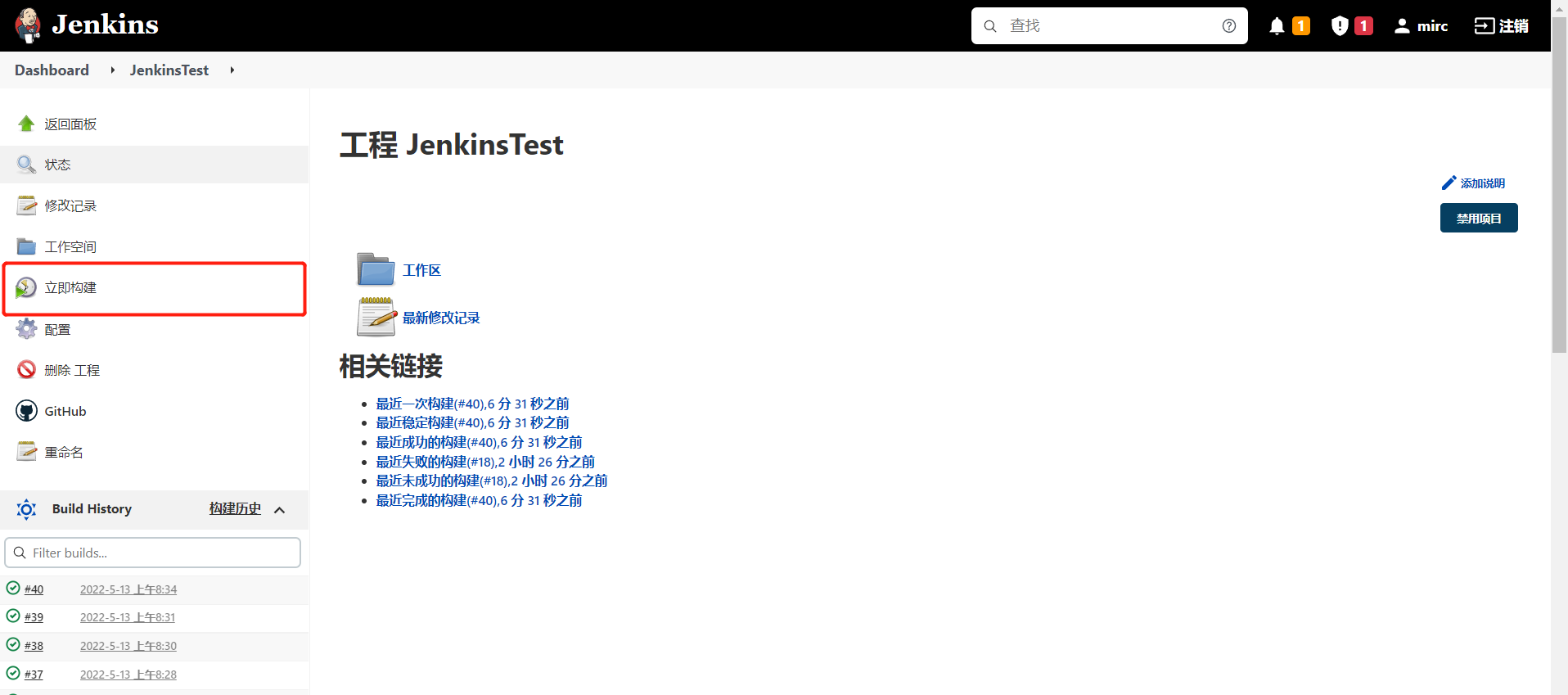
number: 80

" | sudo kubectl apply -f -

1. 最后点击保存



回到项目页点击立即发布手动发布我们的应用



**查看部署效果**

我们访问http://masterip:32268/api/mircservice/test，可以看到请求被重定向到我们的服务了，其中/api/mircservice/test返回的结构为3

