自动化部署方案

在过去几年的DevOps的浪潮中，自动化、持续集成这两个概念早已深入人心(互联网技术人)。比尔盖茨先生曾经都说过：“任何技术在一个业务中使用的第一条规则就是，将自动化应用到一个高效的操作上将会放大高效。第二条就是自动化应用到一个低效操作上，则放大了低效率。”

## 1、自动化部署概述

#### 1.1 什么是自动化部署

一句简单的话概括：部署的过程中所有的操作全部自动化，无需人工手工干预。

#### 1.2 自动部署的好处

传统的部署方式如下：

运维人员手工使用Scp、Xftp等方式来传输数据

手工登录服务器执行git pull 、svn update等命令进行更新代码的操作

开发人员手工编译打包，然后通过内网传输给运维人员

运维人员通过rz上传的方式上传到目标服务器，然后，执行重命名原包、拷贝新包到目标目录，再执行服务应用重启命令完成整个部署过程

看似非常简单，也不是很麻烦，但是一旦项目多，部署频繁，这种情况下就会大大降低工作效率。公司的活动类项目高达100+，很多都是需要快速上线及下线、或者更新的，手工部署真的累。

传统的部署方式有以下的缺点：

整个过程都需要人员参与，占用大量的时间，效率低下

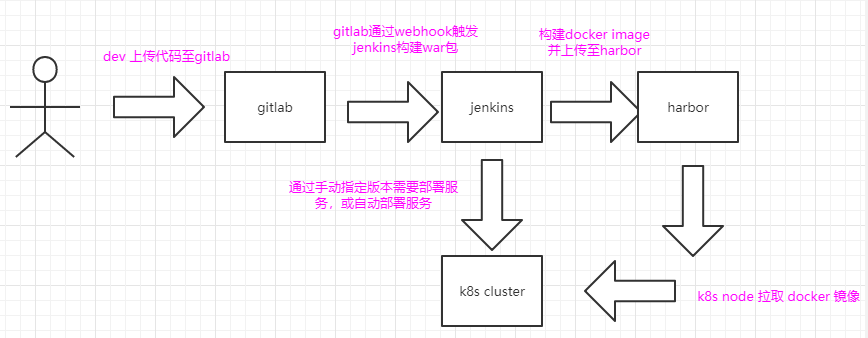
上线、更新、回滚速度慢

存在一定的管理混乱，人为误操作的机率增大

所以，自动化部署的优势就通过这种对比显现出来了！！

## 2、自动化部署的流程

gitlab + jenkins + kubernetes + harbor

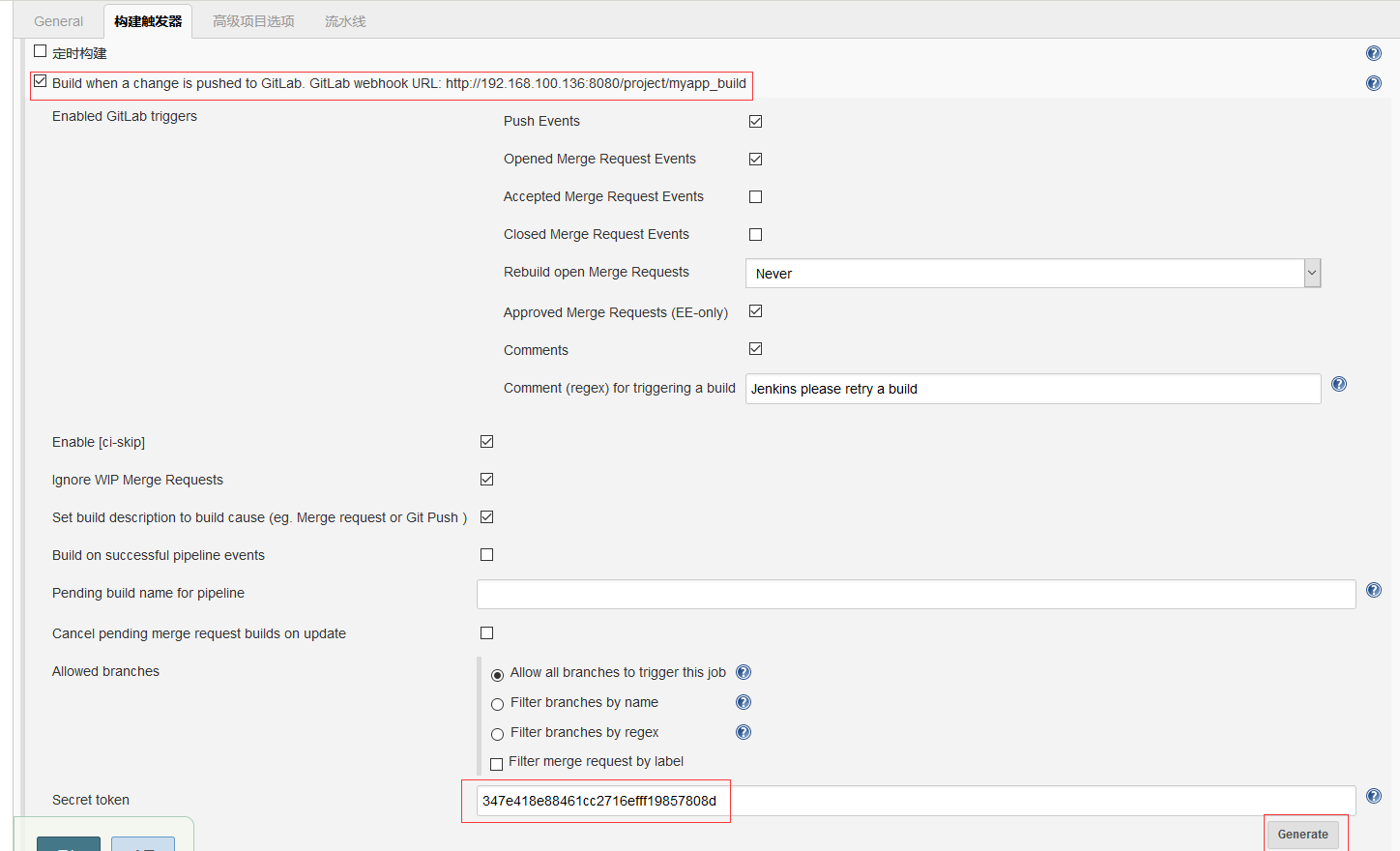


* 开发者向自己的gitlab网站提交了代码
* 通过webhook让jenkins执行自动化构建过程
* jenkins从git上拉取代码进行构建，如构建失败可配置邮件通知开发人员
* jenkins在自动化构建脚本中调用docker命令将构建好的镜像push 私有镜像服务器
* 同时，jenkins也可以直接执行指定版本的docker镜像，进行滚动更新，服务升级过程中客户无感知，业务可正常访问
* 如有问题，也可以通过jenkins进行回滚至上一个版本

#### 2.1 这里安装软件的过程就不在这里啰嗦了，直接开始业务流程

1、在gitlab中创建代码仓库，并配置好webhook，一旦代码仓库有push代码操作，webhook会自动通知jenkins进行构建镜像

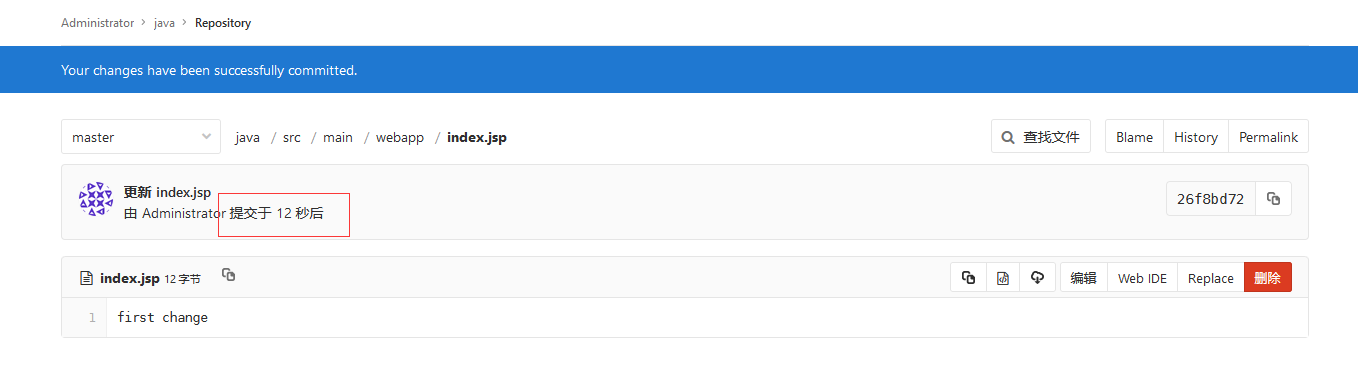
2、我们需要新建一个jenkins的pipeline项目，比如我这里新建的myapp\_build，配置触发器 gitlab webhook，需要安装gitlab webhook插件



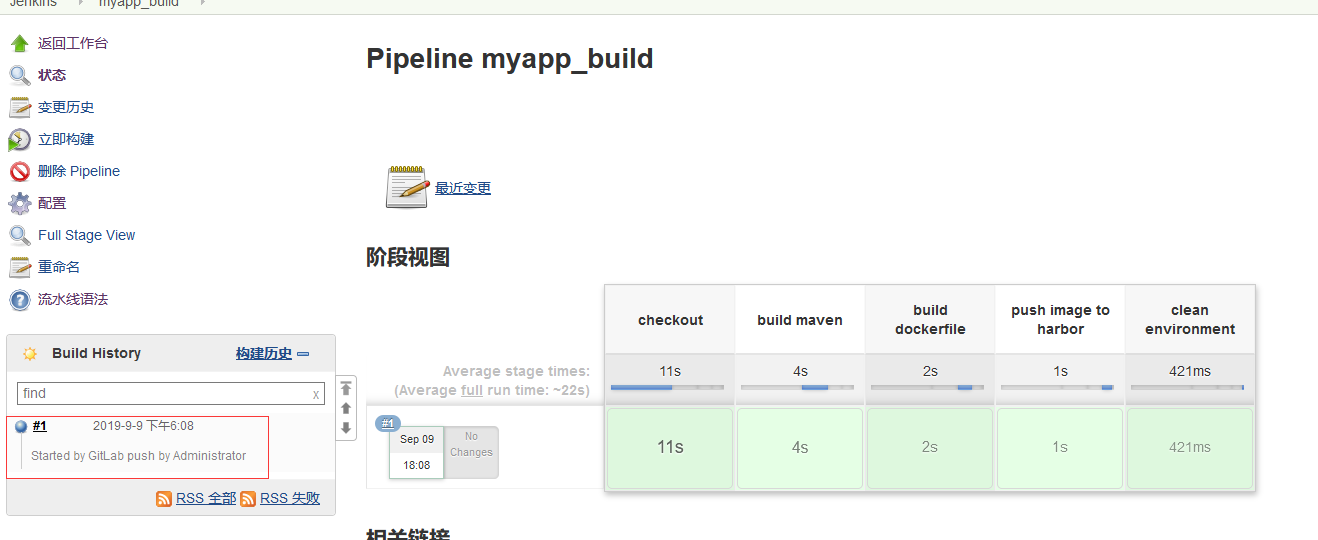
3、我们在gitlab 项目中设置中的集成选项配置jenkins webhook的触发地址和token



4、然后只要gitlab java项目中有代码上传，就会自动构建镜像，例如我将这个java项目 中的一个index.jsp的内容 修改成 first change

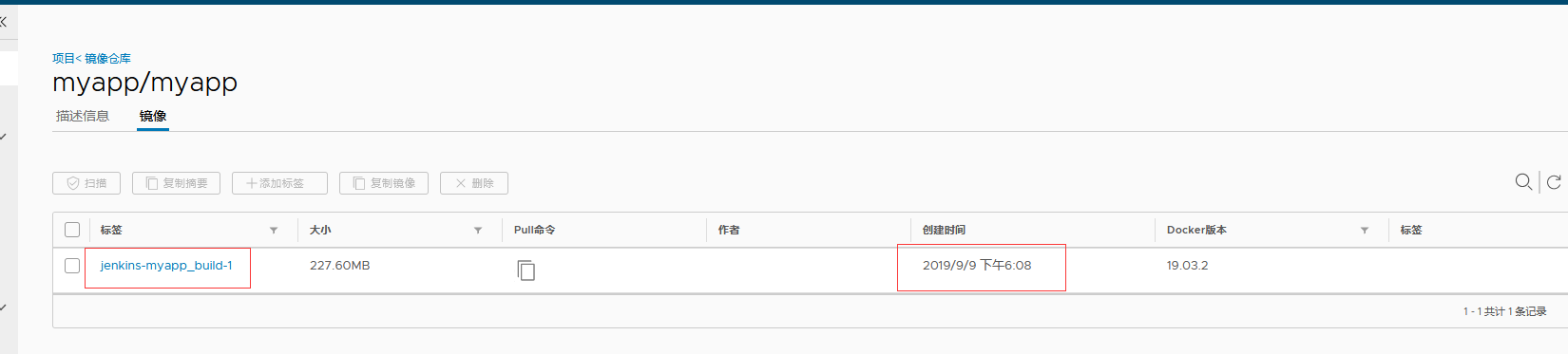


5、我们现在去查看jenkins中，看是否有任务进行构建



Jenkins上已经构建完成了

6、然后去看一下harbor上，镜像是否传入进去



目前整个CI流程中就只缺少测试部分了，我们这里目前不涉及

#### 2.2 自动化部署，同样这个部分的软件安装也不详细说明，需要提前安装好k8s-cluster

1、在jenkins上创建myapp\_deploy项目，写入pipeline脚本



2、第一次构建会显示失败，这是正常的，第一次jenkins还加载到pipeline中的参数选项



3、然后进行第二次构建，选择上图中的Build with Parameters



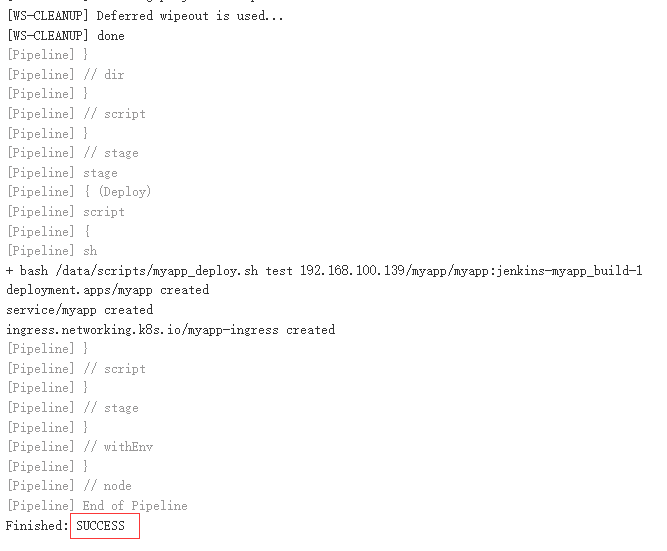
这里有四个参数需要我们填写或者选择

* Action： 选择是发版本还是回滚
* deploy\_env: 选择操作的是测试环境还是开发环境
* dockerImageName：输入harbor中应用镜像的绝对地址
* JenkinsApi：用来控制自动化部署中免交互的选项，默认是false，需要交互

我们输入相关参数，点击开始构建

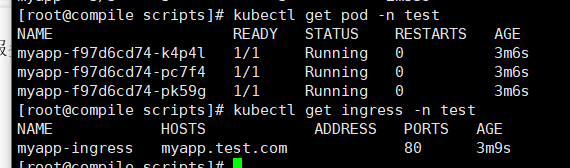


查看控制台输出

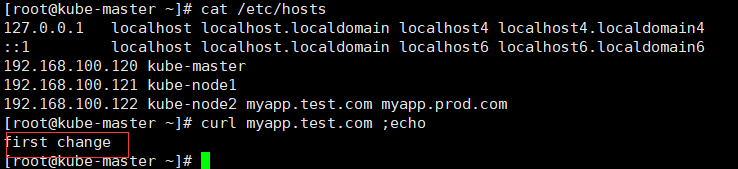




4、我们进入到k8s的shell界面，查看服务以及ingress暴露的域名



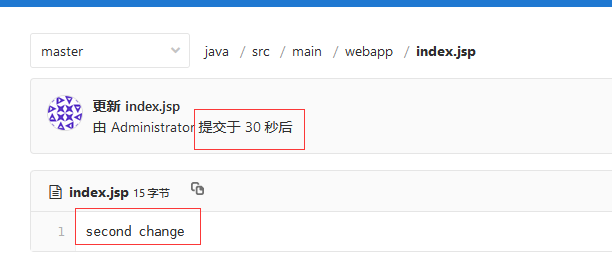
5、因为这个域名只能在本地访问，因此，我在本地机器中添加host，然后访问



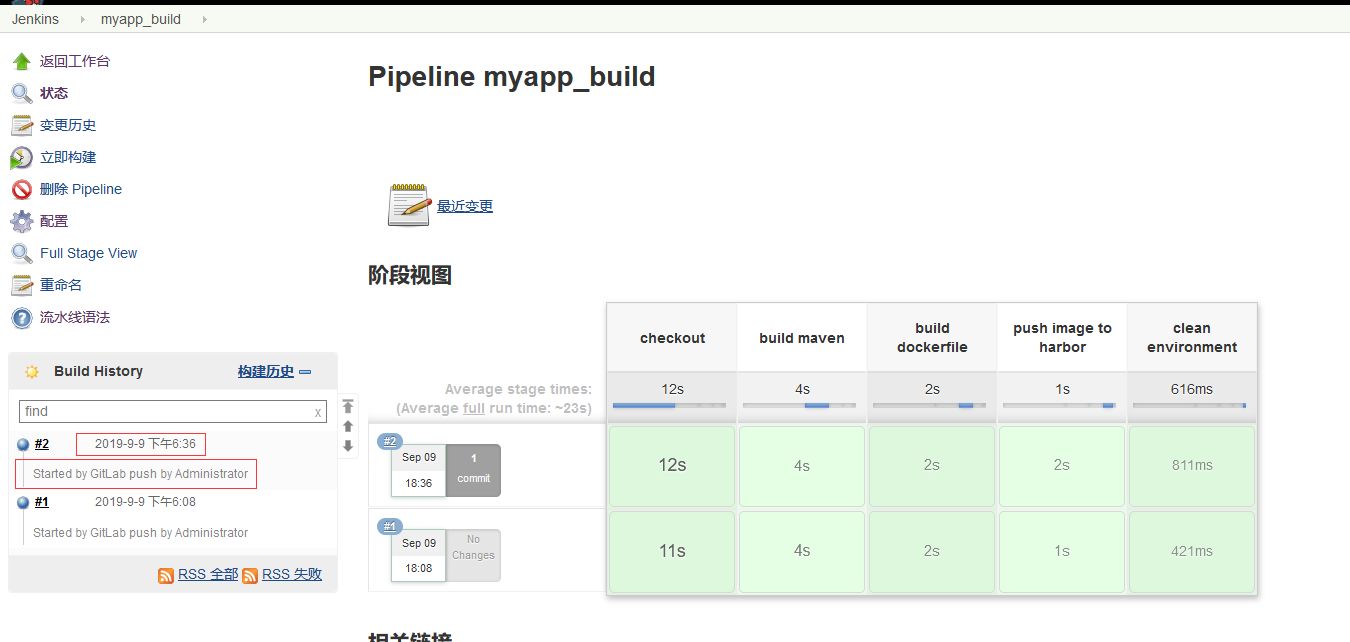
6、成功返回我们之前在gitlab中修改index.jsp中的内容

#### 2.3 这里我们测试一下滚动更新

1、先修改gitlab中index.jsp的内容为second change



2、jenkins中也成功构建中镜像



3、查看harbor中是否存在新构建的镜像



4、然后我们操作jenkins中的myapp\_deploy,进行部署。在部署的过程中，我们一直访问myapp.test.com这个域名，来测试更新中用户无感知



因为是滚动更新，它默认会启动deployment中数量的25%的新镜像容器，然后关闭25%的老镜像容器,所以截图中，有一段时间又有版本二的，也有版本一的，最后都会成为版本一的



#### 2.4、程序回滚

1、我们操作jenkins中的myapp\_deploy,进行回滚。在回滚的过程中，我们也一直访问myapp.test.com这个域名，来测试更新中用户无感知

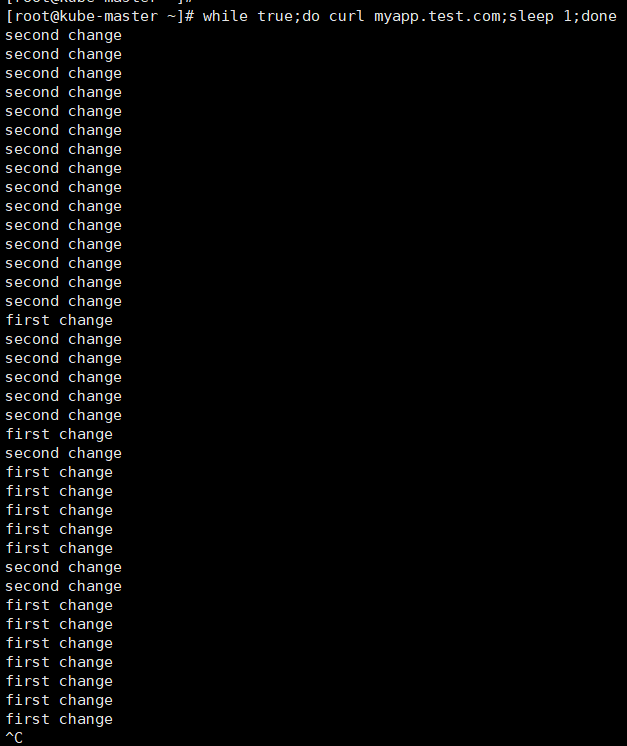
Action 选择程序回滚

dockerimageName填写回滚,点击构建



2、jenkins显示回滚成功



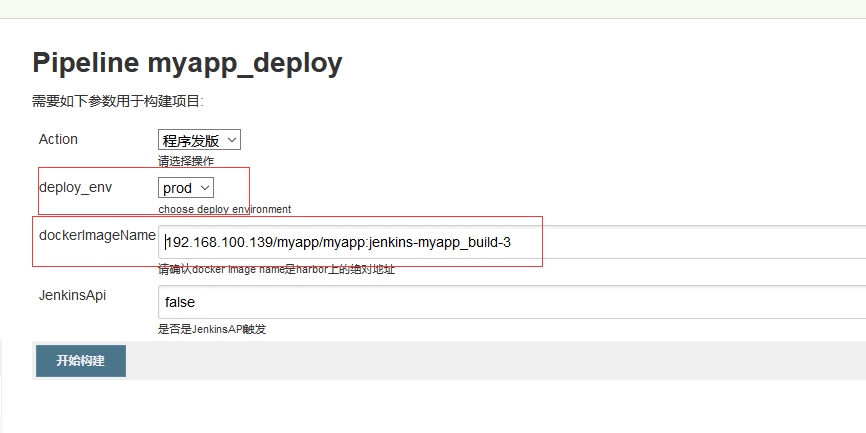


#### 3.1、生产环境中部署跟测试环境中部署差不多，就是多一个领导确认步骤

1、我们操作jenkins中的myapp\_deploy,进行部署。

Deploy\_env: 选择prod，生产环境

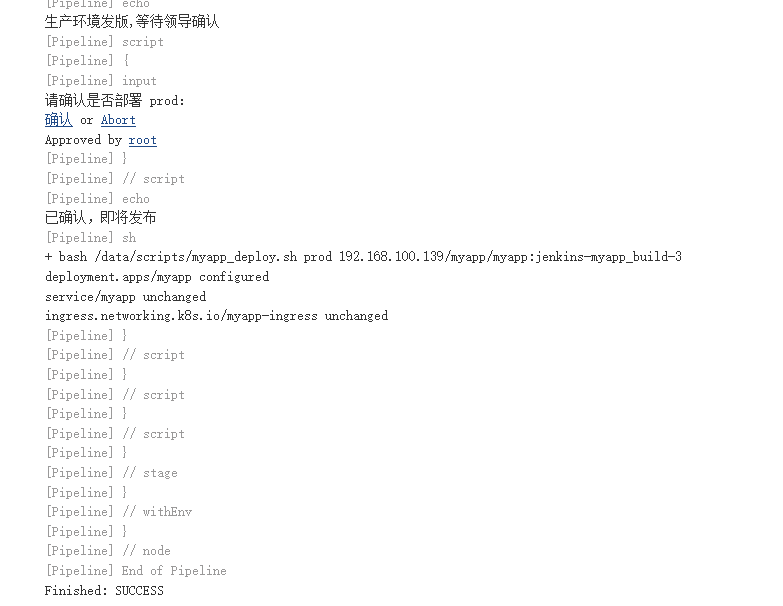
dockerImageName: 填写harbor中的镜像地址



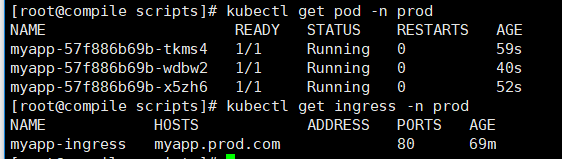
2、点击开始构建后，程序不会一直走完，需要我们进行一步确认后，才能进行部署，点击控制台，查看控制台输出

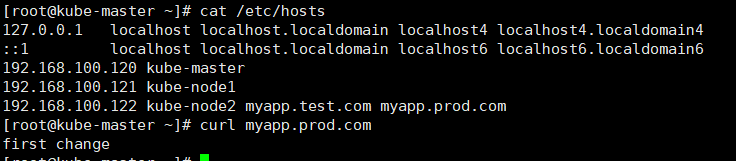


3、这里我们点击确认



4、查看prod的域名，添加hosts解析





#### 3.2 生产环境程序回滚，相比测试环境也是多一个确认环节，这里就不演示了。同上