Python与Devops

Law

目录 CONTENTS

Devops技术栈

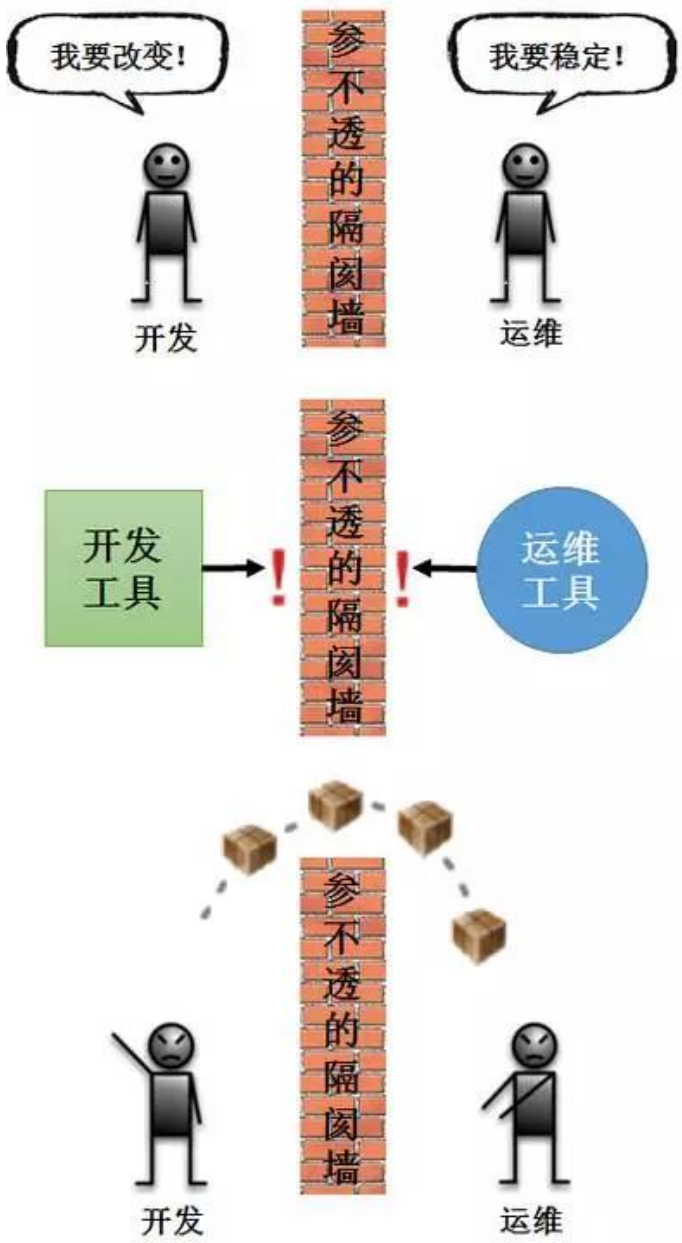
Devops构建示例

Python工具二次开发

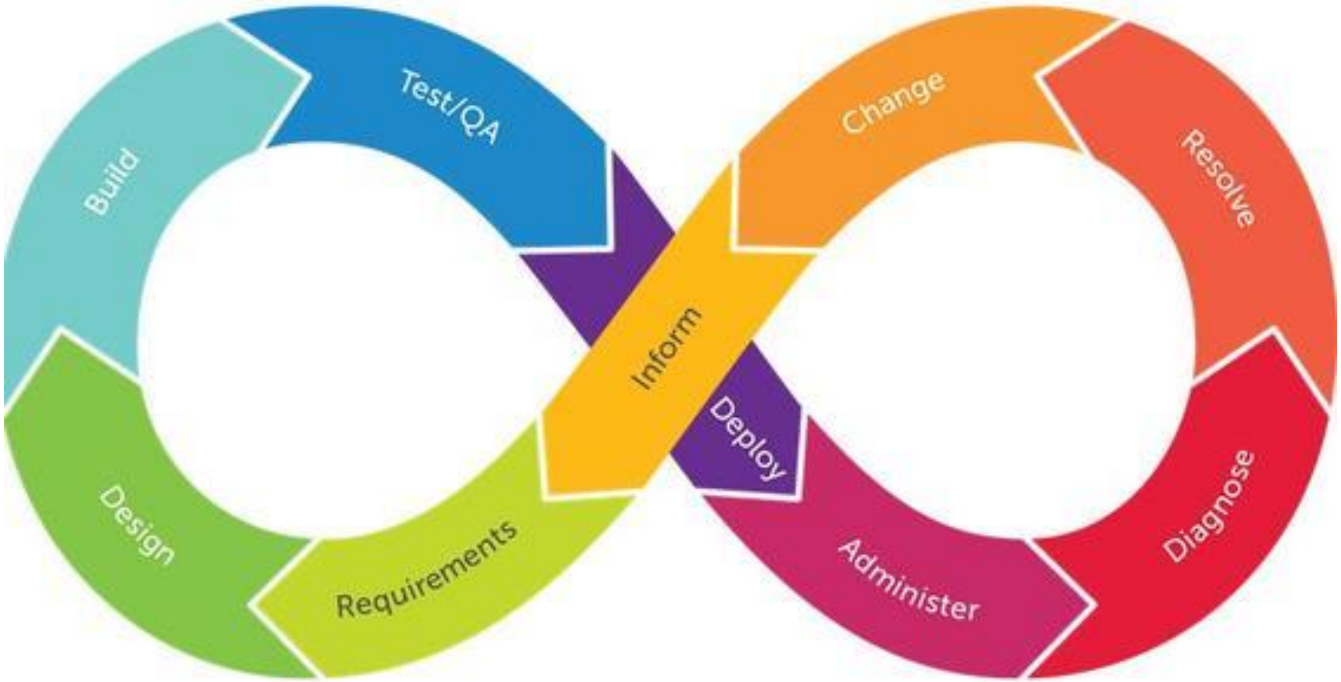
Devops

DevOps 是一个完整的面向IT运维的工作流，以 IT 自动化以及持续集成（CI）、持续部署（CD）为基础，来优化程式开发、测试、系统运维等所有环节

构建Devops之前面临的问题



Devops

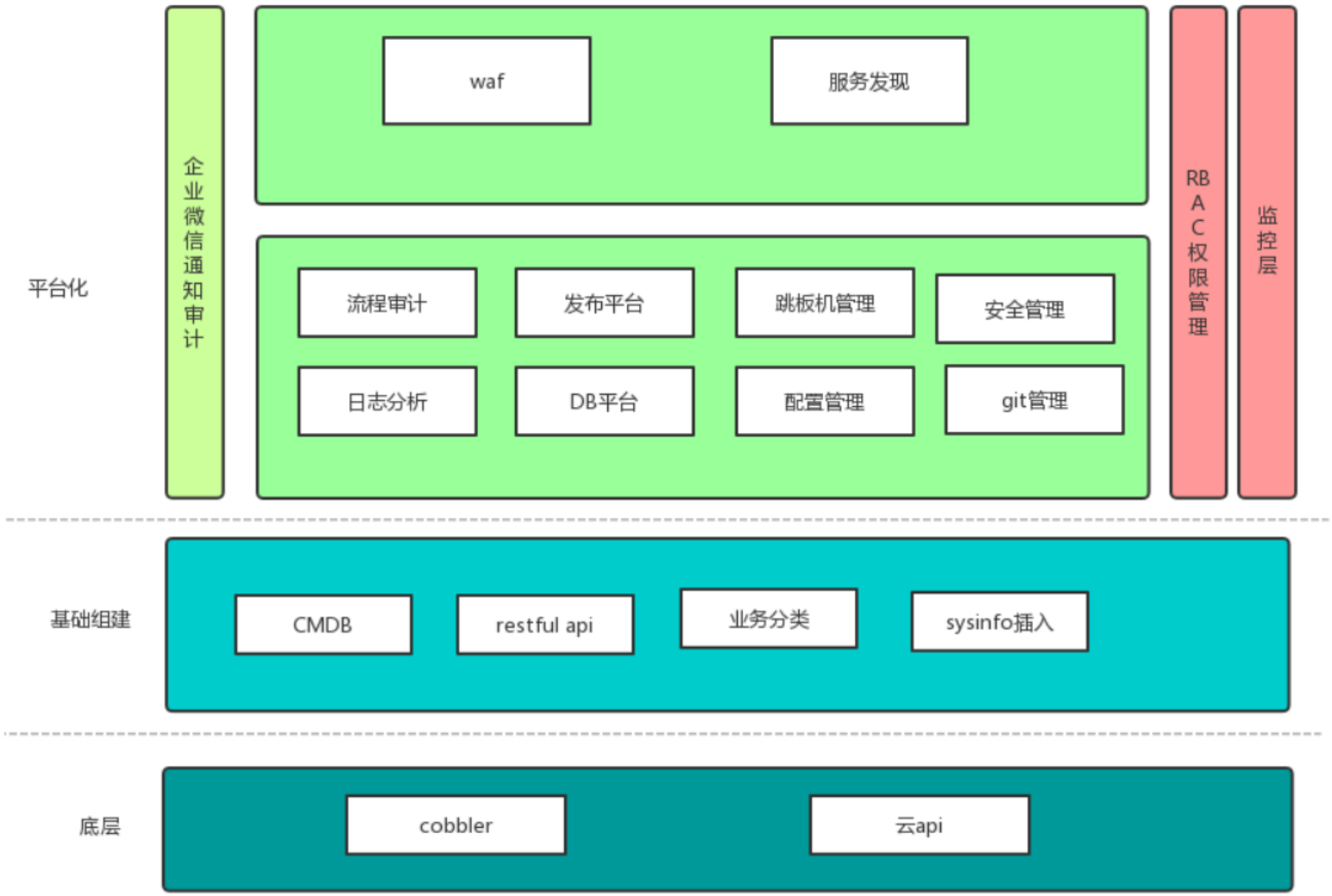


过程包含，代码构建打包、测试、部署、发布、监控、回滚等等一个项目的闭环、快速构建以上过程。

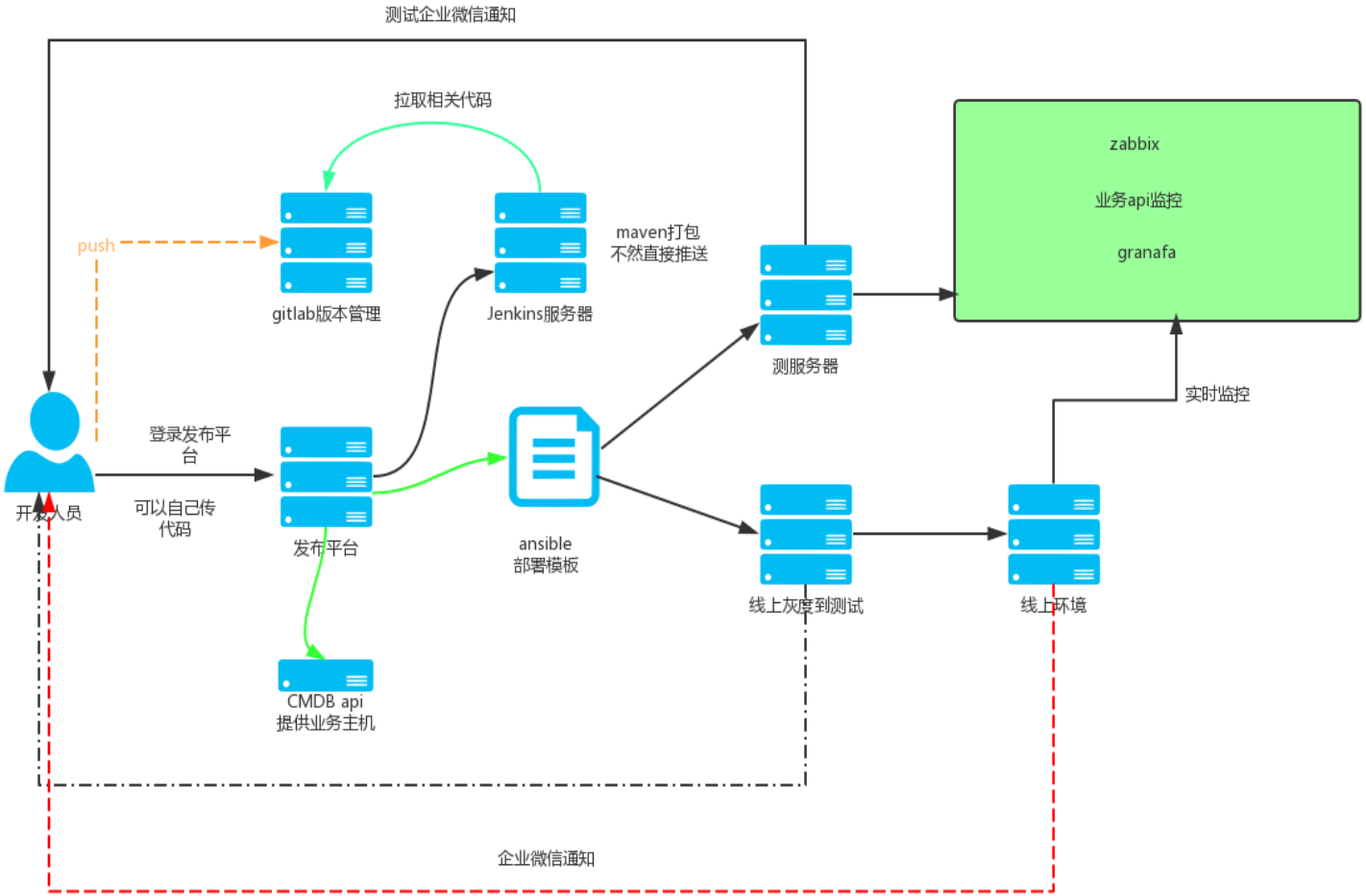
构建Devops常用软件技术栈

- 代码管理（SCM）：GitHub、GitLab、SubVersion
- 构建工具：Ant、maven
- 自动部署：ansible、saltstack、puppet
- 持续集成（CI）：Jenkins
- 配置管理：Ansible、Chef、Puppet、SaltStack
- 容器：Docker、kvm
- 编排：Kubernetes、openstack、Apache Mesos、swarm
- 服务注册与发现：Zookeeper、etcd、Consul
- 脚本语言：python、ruby、shell
- 日志管理：ELK、Logentries
- 系统监控：zabbix、prometheus

前东家关于Devops的一些实践



Devops传统方式下构建示例



Devops发布流程需要考虑几个问题

1、发布什么应用，发布到哪里（CMDB）

2、发布人员是否有权限（RBAC）

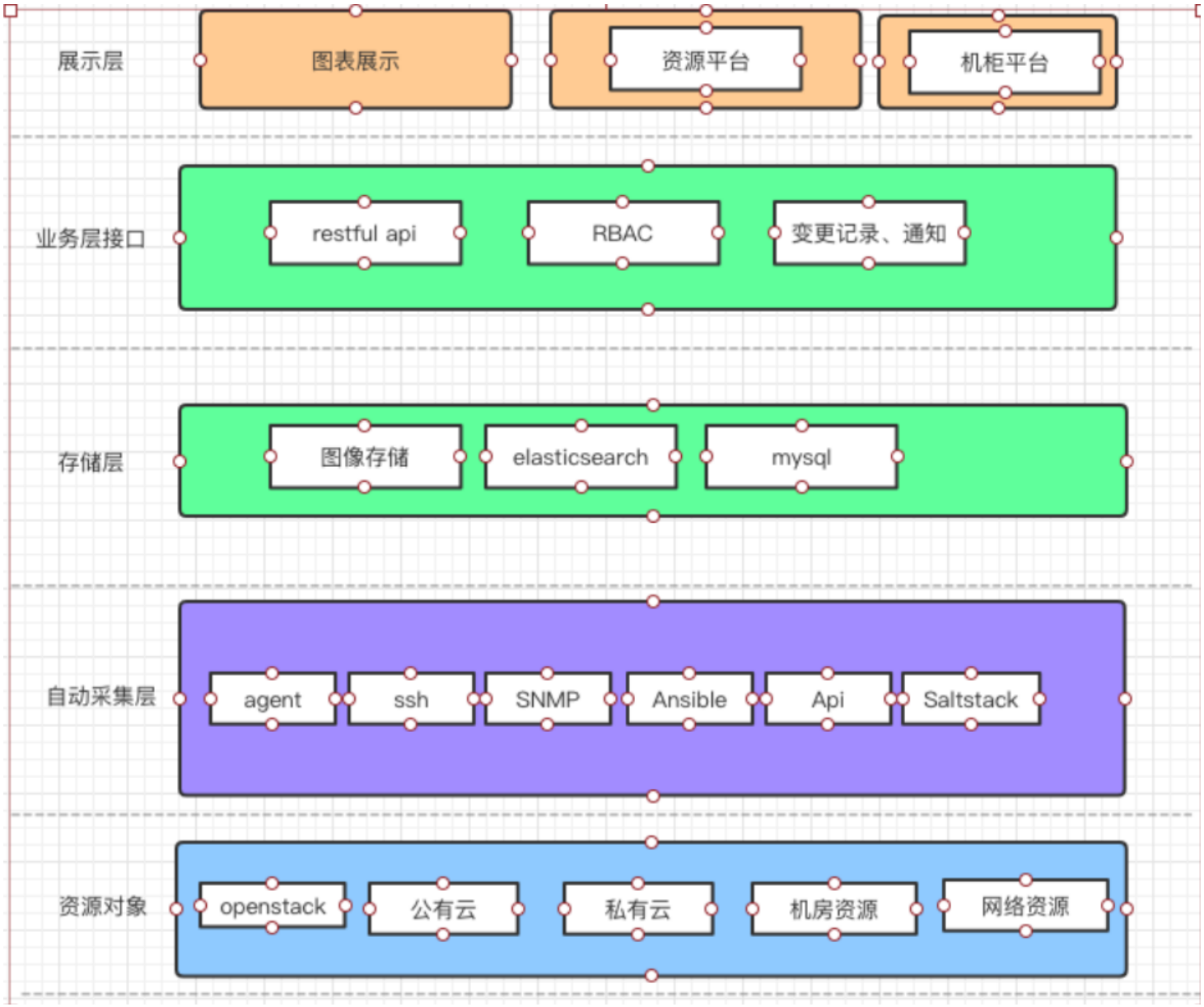
3、发布过程是否有认证，授权发布（流程审计）

4、发布的时候需要做什么操作，批量时候如何处理，如何知道执行结果返回

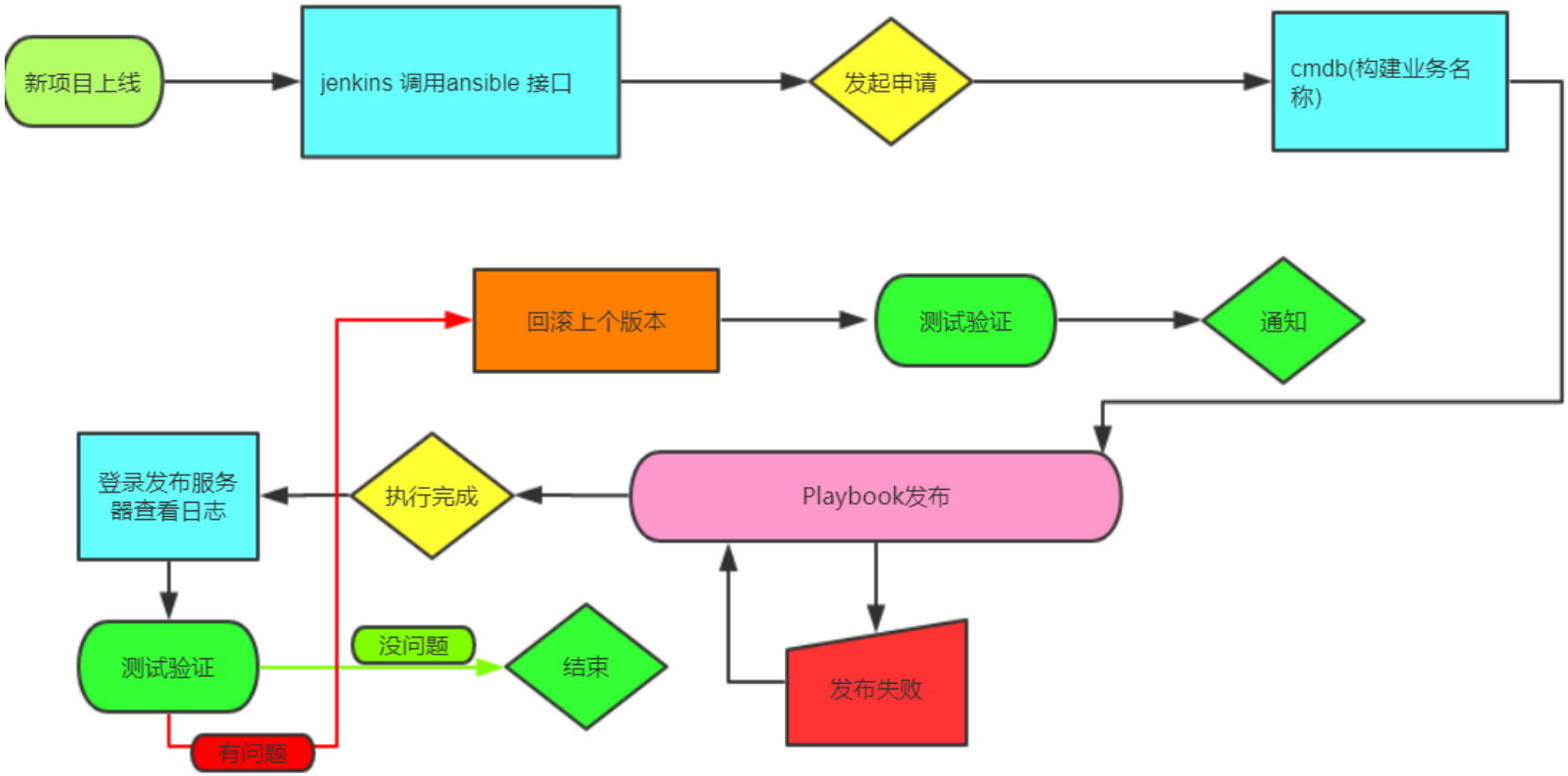
5、消息通知机制

6、发布故障如何快速回滚

CMDB为基础架构由下至上



Ansible发布


新项目上线
jenkins 调用ansible 接口
发起申请
cmdb(构建业务名称)
回滚上个版本
测试验证
通知
登录发布服务器查看日志
执行完成
Playbook发布
测试验证
没问题
结束
发布失败
有问题

为什么使用python

1、入门快速简单

2、丰富的系统管理模块，可以快速管理linux 操作系统

3、丰富的模块、类似gitlab、jenkins、ansible 等自动化软件可以快速开发

4、自动化批量管理软件，ansible、saltstack由python开发、可以自定义开发相关组件

5、web框架简单上述，笔者使用flask一个文件就可以完成后端简单开发。

对于ansible api重新封装

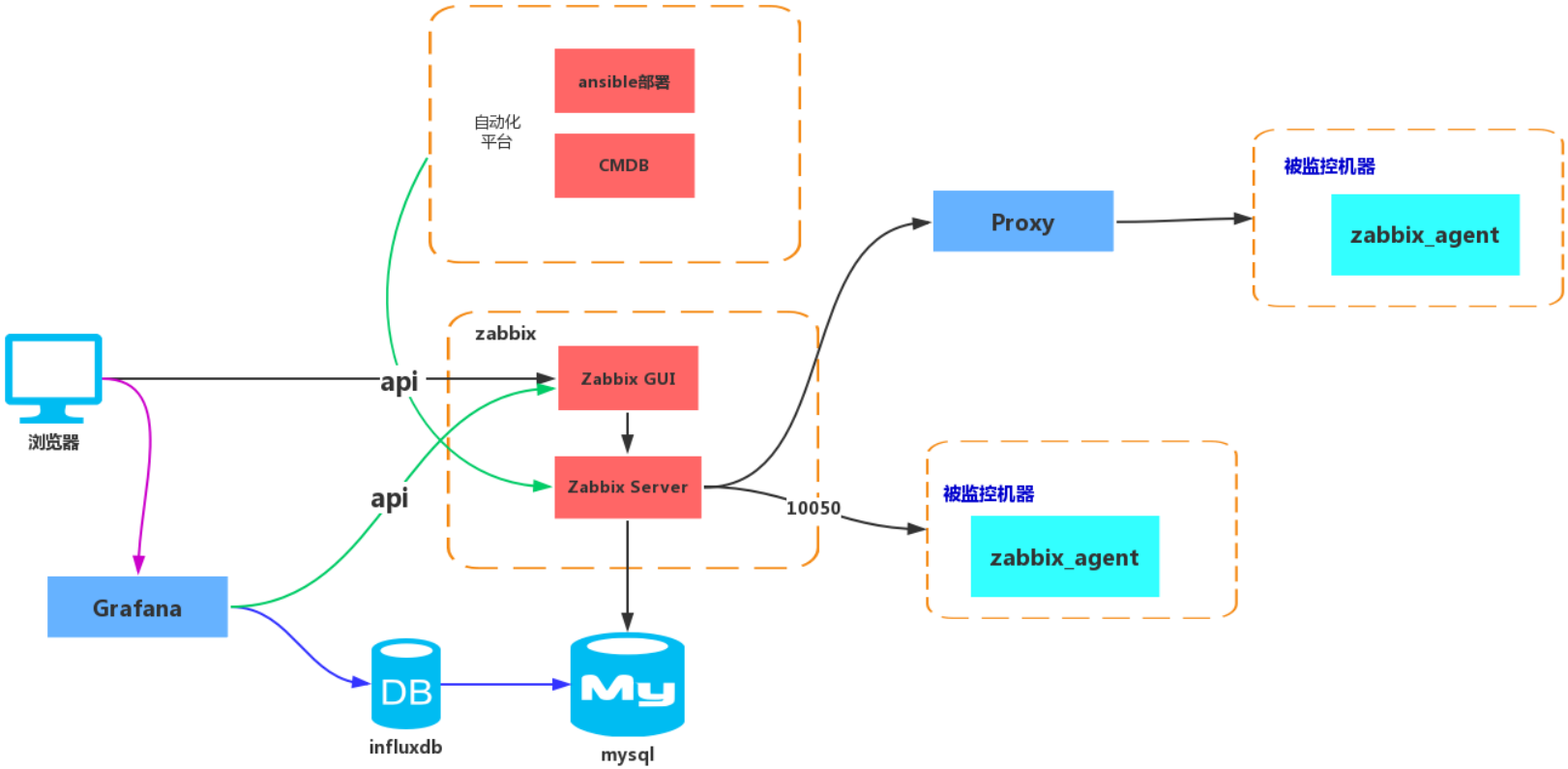

```
from pprint import pprint
from Ansible2_myAPI.playbook_runner import PlaybookRunner

runner = PlaybookRunner(
                playbook_path=“deploy.yml",
   hosts="192.168.1.100, 192.168.1.200",
)

result = runner.run()
pprint(result)
```

##单独封装api
定义好deploy的playbook,执行代码流程发布
Hosts:主机可以从业务线来获取执行
封装的api 返回接口执行远端主机信息

对于ansible 简单返回

{'contacted': {
'192.168.1.100': {
u'changed': True,
u'cmd': u'uptime',
u'delta': u'0:00:00.006604',
u'end': u'2017-01-16 16:47:44.051826',
'invocation': {
u'module_args': {
u'_raw_params': u'uptime',
u'_uses_shell': True,
u'warn': True},
'module_name': u'command'},
u'rc': 0,
u'start': u'2017-01-16 16:47:44.045222',
u'stderr': u'',
u'stdout': u' 16:47:44 up 570 days, 1:40, 1 user, load average: 0.01, 0.02, 0.05',
'stdout_lines': [u' 16:47:44 up 570 days, 1:40, 1 user, load average: 0.01, 0.02, 0.05'],
u'warnings': []}},
'dark': {‘192.168.1.200': {
'changed': False,
'msg': u'Failed to connect to the host via ssh: ssh: connect to host 1.1.1.1 port 22: Connection timed out\r\n',
'unreachable': True}}}

监控系统二次开发

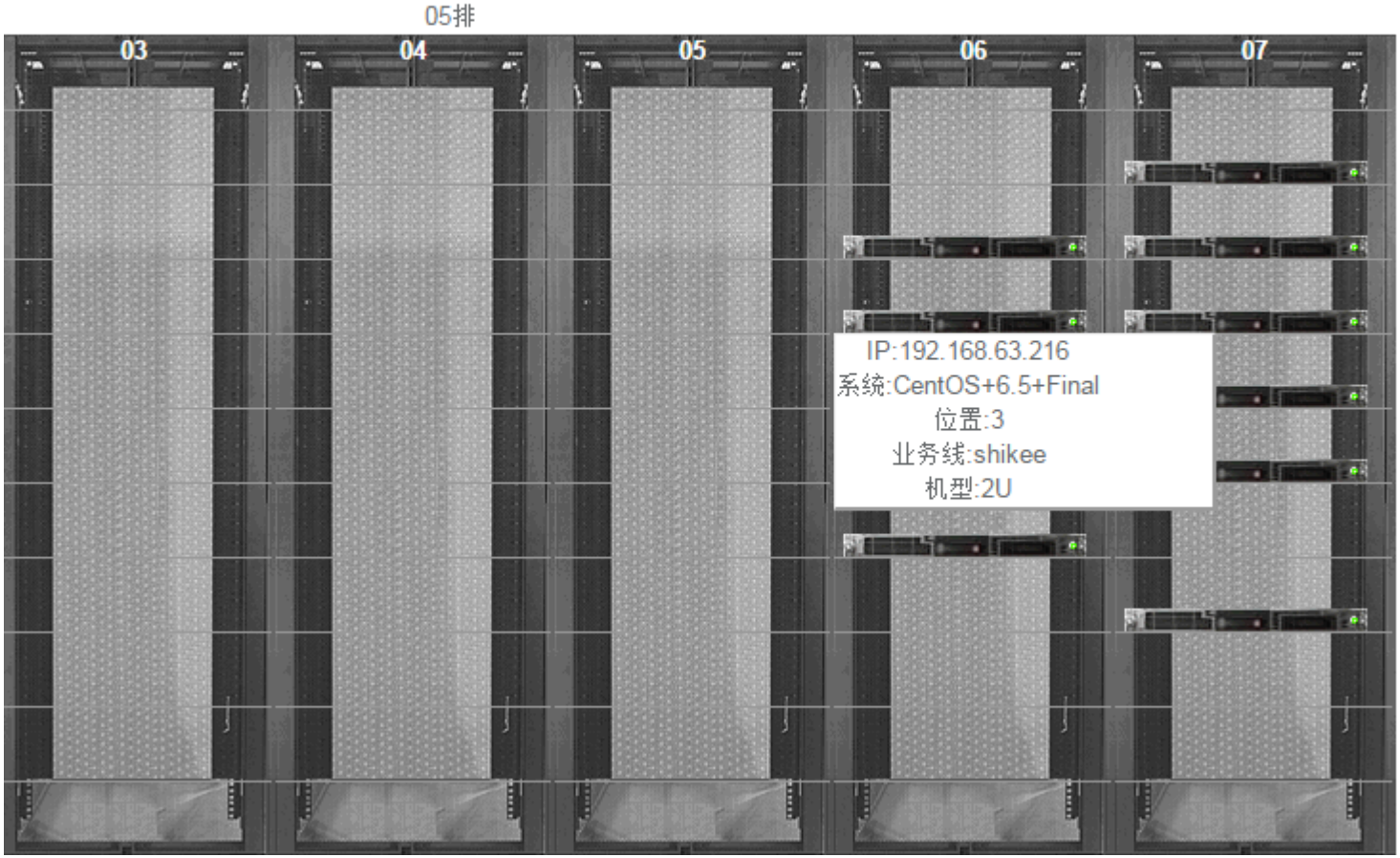

自动化
平台
ansible部署
CMDB
被监控机器
Proxy
zabbix_agent
zabbix
Zabbix GUI
Zabbix Server
浏览器
api
api
10050
被监控机器
zabbix_agent
Grafana
DB
influxdb
mysql

Zabbix api 二次开发

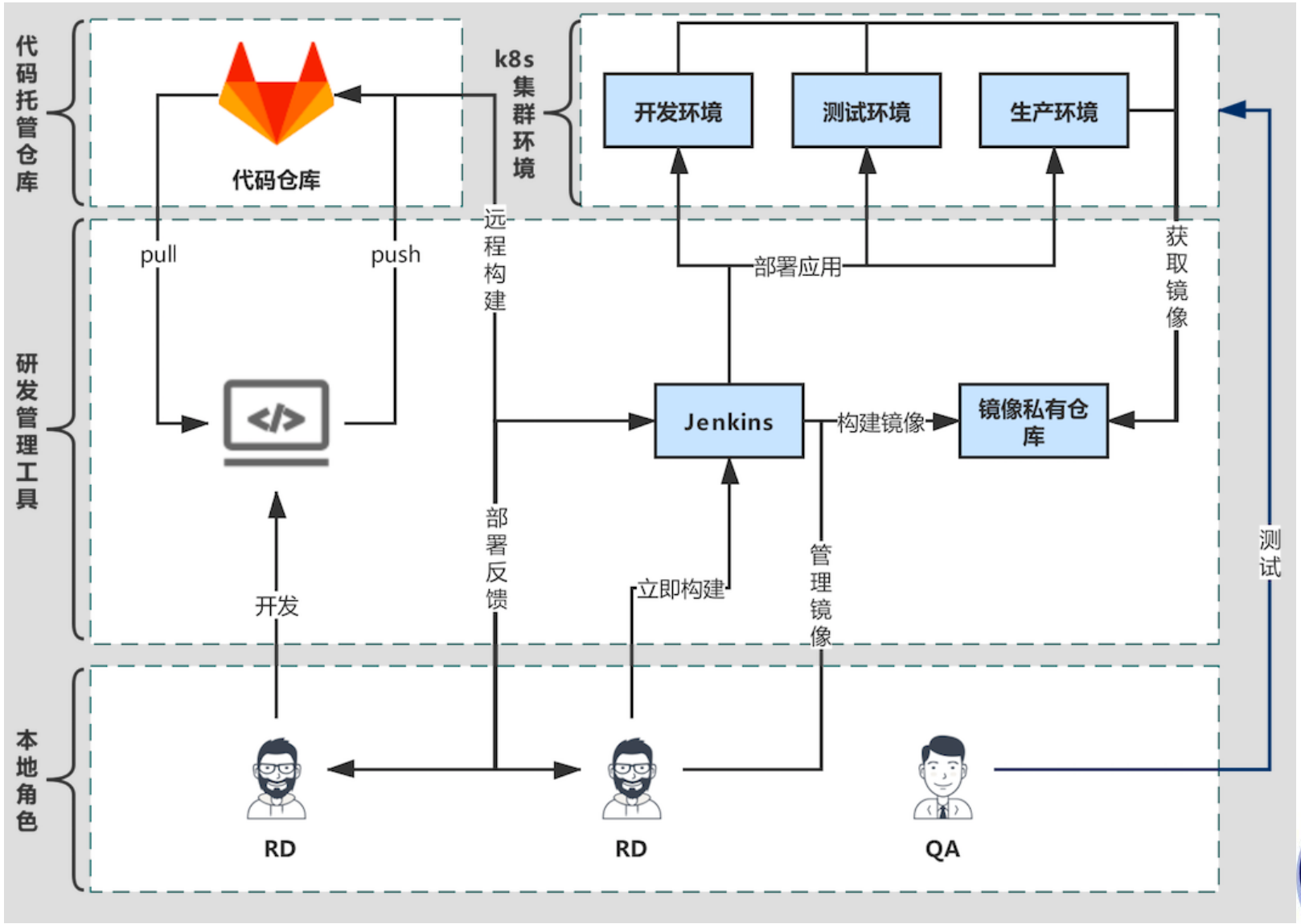
https://blog.51cto.com/xiaoluoge/1827151

```
#coding:utf-8
import time
from zabbix_client import ZabbixServerProxy
class Zabbix():
    def __init__(self,url):
        self.zb = ZabbixServerProxy(url)
        self.zb.user.login(user="Admin", password="zabbix")
     ############### 查询组所有组获取组id ###############
    def get_hostgroup(self):
        data = {
          "output":['groupid','name']
         }
        ret = self.zb.hostgroup.get(**data)
        return ret
```

Zabbix api 开发的机柜告警平台

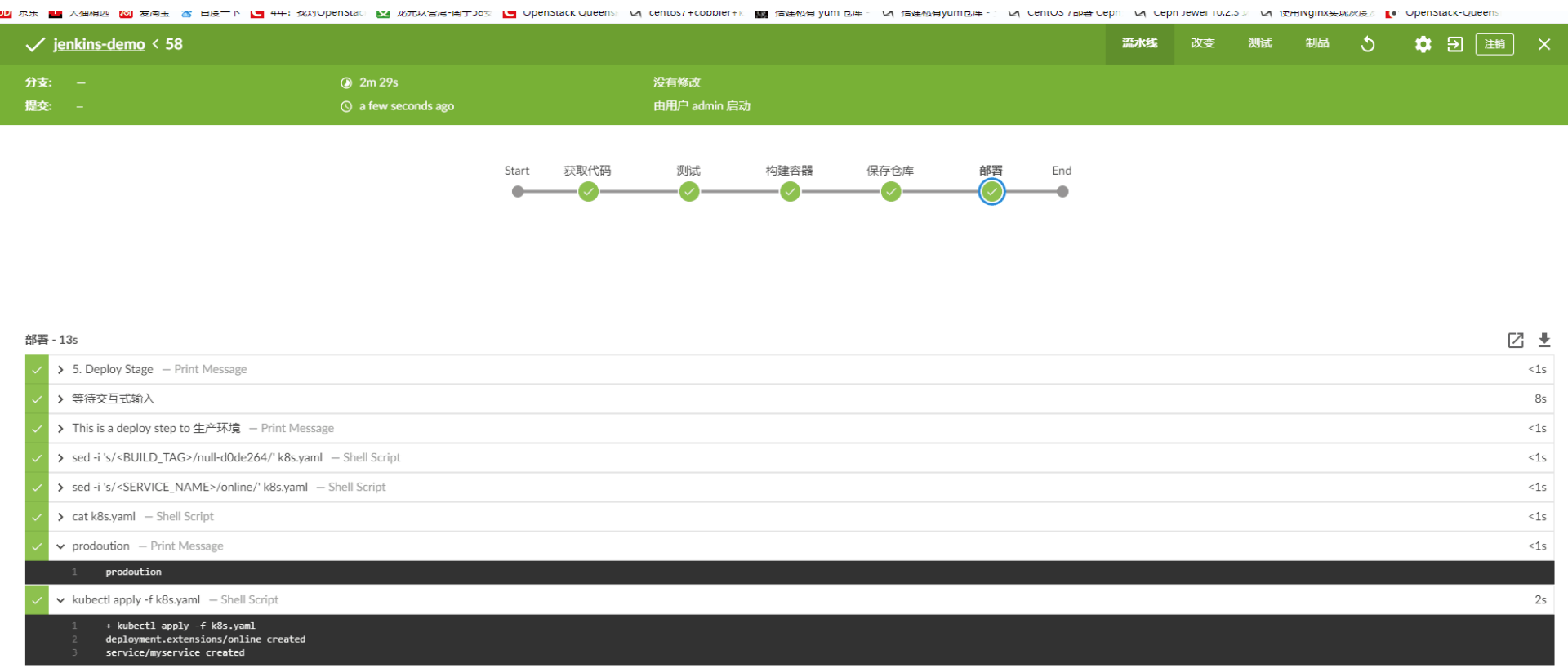


在Docker环境下的发布系统



在Docker环境下的发布系统

PyConChina 2019



https://github.com/xiaoluoge11/jenkins_demo

由Jenkinsfile确定操作流程

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master ▾ jenkins_demo / Jenkinsfile Find file Copy path

Fetching contributors...

Cannot retrieve contributors at this time.

58 lines (57 sloc) | 1.86 KB Raw Blame History

```
node('haimaxy-jnlp') {
    //服务名称
    def service_name = "online"
    stage('获取代码') {
       echo "1.Prepare Stage"
       checkout scm
       script {
           build_tag = sh(returnStdout: true, script: 'git rev-parse --short HEAD').trim()
           if (env.BRANCH_NAME != 'master') {
               build_tag = "${env.BRANCH_NAME}-${build_tag}"
           }
       }
    }
    stage('测试') {
      echo "2.Test Stage"
    }
    stage('构建容器') {
      echo "3.Build Docker Image Stage"
      sh "docker build -t 192.168.100.27/devops/jenkins-demo:${build_tag} ."
    }
    stage('保存仓库') {
      echo "4.Push Docker Image Stage"
      withCredentials([usernamePassword(credentialsId: 'Harbor', passwordVariable: 'HarborPassword', usernameVariable: 'HarborUser')]) {
            sh "docker login -u ${HarborUser} -p ${HarborPassword} 192.168.100.27"
            sh "docker push 192.168.100.27/devops/jenkins-demo:${build_tag}"
```

K8s Ci/CD快速发布流程

1. 开发人员提交代码到 Gitlab 代码仓库
2. 编写Dockerfile 到代码目录
3. 通过 Gitlab 配置的 Jenkins Webhook 触发 Pipeline 自动构建
4. Jenkins 触发构建构建任务，编写jenkinsfile.
5. 先进行代码静态分析，单元测试
6. 然后进行 Maven 构建（Java 项目）
7. 根据构建结果构建 Docker 镜像
8. 推送 Docker 镜像到 Harbor 仓库
9. 触发更新服务阶段，使用kubectl 从pod构建，当然kubectl版本打包到jenkin-slave镜像里
10. 发布更新、检测状态，更新失败停止，返回上一个版本

THANK YOU

law

无

保密

扫一扫上面的二维码图案，加我微信