高效运维社区

GreatOPS Community

—陪伴您的运维生涯

基于容器的持续集成平台建设

郭宏泽@高效运维社区

用户故事

- 创业公司，运维需求变化快，资源相对有限。以速度为中心，快速服务，快速响应，运维可控，降低成本，稳定安全的运维要求。

- 为满足公司要求，简化流程，提高效率，建设以k8s docker，jenkins等开源软件为基础的一套可以自动配置、注册、发布、服务、测试的持续集成容器平台。

业务架构


智能DNS调度
运维管理
CDN
四层负载
F5/LVS/HAPROXY
七层负载/Ningx
监控
cacti/zabbix
日志ELK
webserverNi
ngx
Java/Service
Memcached
NTP/DNS
LDAP/Jumserv
er
缓存/VANISH
Redis/codis
Kafka
CMDB
Saltstack
MogileFS
FastdFS
MySQL
BACKUP
Jenkins
Yum/Repo
CPU
内存
硬盘
网络

版本选型

组件	版本	说明
Kubernetes	1.5.2	主程序
Docker	1.12.6	容器
Flannel	0.7.0	网络组件
Etcd	3.1.0	数据库
Kubernetes-Dashboard	1.6.0	界面
Kubedns	1.9	DNS组件
Harbor	0.5.0	私有镜像库
Heapster	1.2.0	监控

使用心得

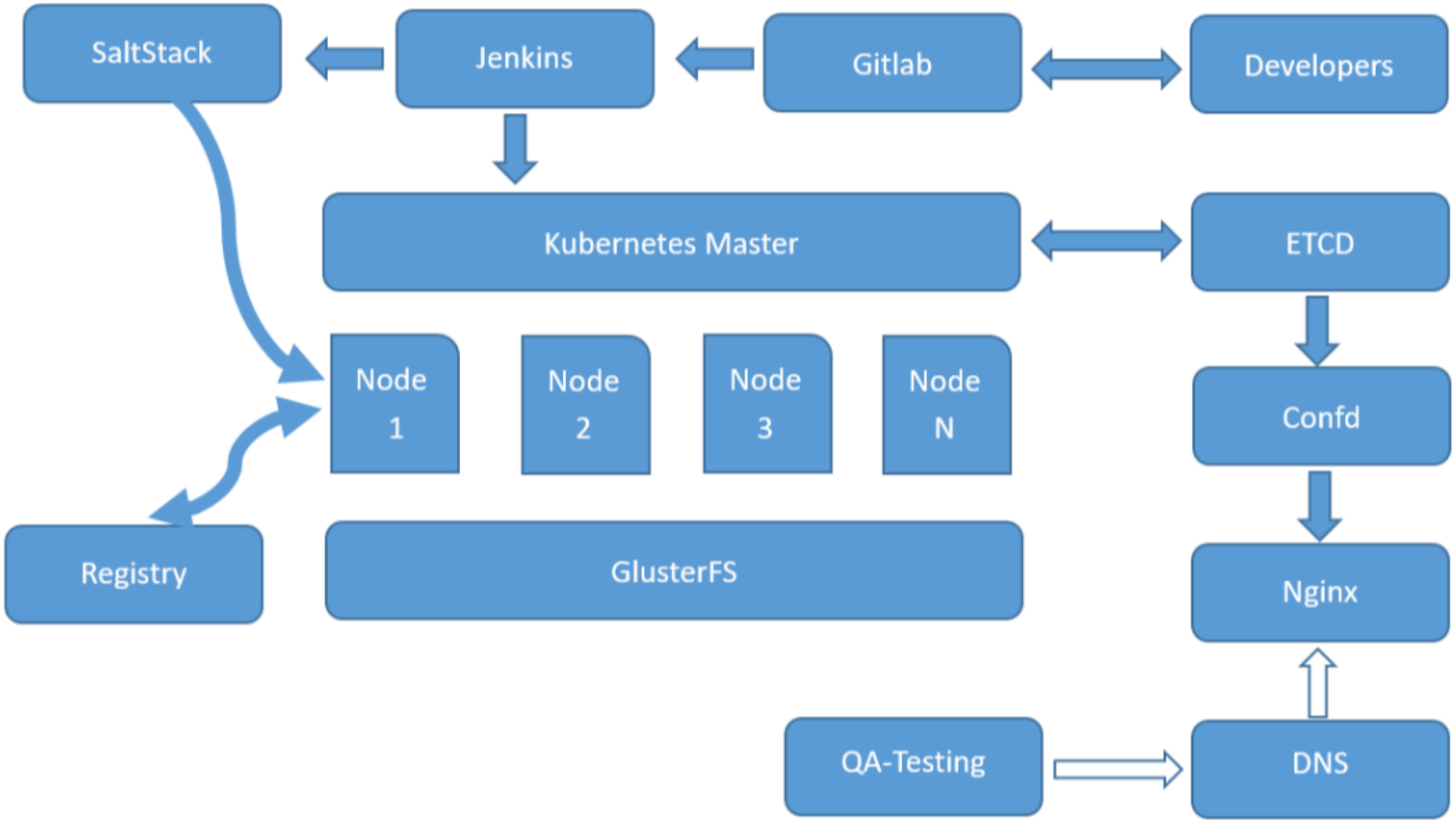
- K8s每个版本将会发布三种功能：
  1. Aplpha功能，预览功能，不稳定。
  2. Beta功能，测试中的功能，不保证稳定。
  3. 正式功能,已经经过测试的稳定功能，不保证无BUG。

- 尽可能减少对功能的使用，保持简单。

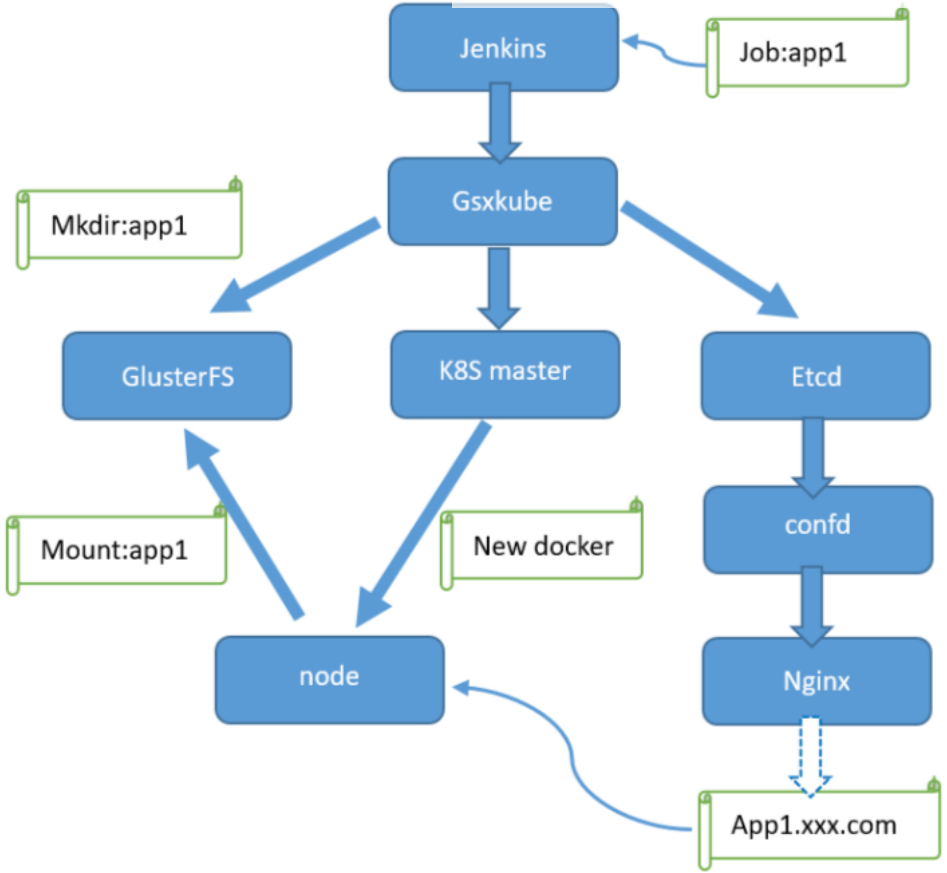
- 不要轻易升级你的生产系统。

- K8S的设计是面向基础平台，而非用户应用。

平台架构


SaltStack
Jenkins
Gitlab
Developers
Kubernetes Master
ETCD
Node 1
Node 2
Node 3
Node N
Confd
Registry
GlusterFS
Nginx
QA-Testing
DNS

控制程序


Jenkins
Job:app1
Gsxkube
Mkdir:app1
GlusterFS
K8S master
Etcd
Mount:app1
New docker
confd
node
Nginx
App1.xxx.com

RC管理

```
class ReplicationController(object):
    def __init__(self):
        self.client = HTTPClient(kubeconfig)
    def create_replication_controller(self, name, replicas, images):
        volume_gfs_name = "vol-{}".format(name)
        host_path_mount = "/mnt/{}".format(name)
        volumemount = {'name': volume_gfs_name, 'mountPath': '/apps'}
        volumemounts = [volumemount]
        container_atom = {'name': name,
                          'image': images,
                          'ports': [{'containerPort': 22}],
                          'imagePullPolicy': 'Always',
                          'volumeMounts': volumemounts,
                          'dnsPolicy': 'ClusterFirst',
                          'resources': {'limits': {'cpu': '500m',
                                                   'memory': '2000Mi'},
                                        'requests': {'cpu': '10m',
                                                     'memory':  ‘100Mi'}}}
```

ETCD


```
class Etcd(object):
    def __init__(self):
        self.namespace = "default"
        self.etcd_client = etcd.Client(host='172.21.133.1', port=4001)

    def add_web_app(self, name, namespace, http_port, https_port, host_ip):
        domain = "{}.ctest.baijiahulian.com {}.ctest.genshuixue.com".format(
            name, name)
        app = "devnginx/{}/{}".format(namespace, name)
        app_info = {"name": name,
                    "http_port": http_port,
                    "https_port": https_port,
                    "domain": domain,
                    "host_ip": host_ip}
        self.etcd_client.set(app, json.dumps(app_info))
```

Gluster


```
class GFS(object):
    def __init__(self):
        self.volume = gfapi.Volume("172.21.133.1", "gv0")
        self.volume.mount()

    def create_dir(self, name):
        self.volume.mkdir(name, 0755)
        self.volume.umount()
```

解决DNS问题

- K8s的service IP将被kubedns解析为内部负载均衡地址
- baijiahulian.com 的解析将由自定义域名服务器172.16.133.100解析
- 其它域名将由公网DNS解析（最多设置三个）

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {"baijiahulian.com": ["172.16.133.100"]}
  upstreamNameservers: |
    ["8.8.8.8", "219.141.133.10"]
```

域名发现

在etcd中注册了一个新的键devnginx，在控制程序第一次创建容器时会向etcd发送容器的相关信息，如：

主机的物理IP地址(NodePort)，SSH端口，HTTP端口，所用域名等。这些信息将提供给confd使用。

```
[root@of-bj-op-k8s01 ~]# curl -l http://172.21.133.1:4001/v2/keys/devnginx/default/test-txcc
{"action":"get","node":{"key":"/devnginx/default/test-txcc","value":"{\"domain\": \"test-txcc.ctest
.[redacted] test-txcc.ctest.[redacted]\", \"http_port\": 31153, \"name\": \"test-txcc\",
\"host_ip\": \"172.21.133.12\", \"https_port\": 30269}","modifiedIndex":4448481,"createdIndex":4448
481}}
[root@of-bj-op-k8s01 ~]#
```

Confd

从etcd的devnginx/default/ 中读取所有信息，然后通过nginx模板生成 nginx配置文件，生成新的配置文件后重启nginx。

```
[root@of-bj-op-nginx01 conf.d]# cat myconfig.toml
[template]
src = "service.conf.tmpl"
dest = "/apps/srv/nginx/conf/conf.d/k8s.conf"
keys = [
  "/devnginx/default"
]
#check_cmd = "/apps/srv/nginx/sbin/nginx -t"
reload_cmd = "/apps/srv/nginx/sbin/nginx -t && /apps/srv/nginx/sbin/nginx -s reload"
[root@of-bj-op-nginx01 conf.d]#
```

Nginx

内部DNS将泛域名解析到nginx，然后nginx通过由confd生成的配置文件将域名解析到对应的docker，当jenkins创建一个docker后，docker的域名信息就被注册到etcd中，confd读取后生成新的nginx配置，访问者便可在发布一个job后直接访问自己新建或更新的代码结果了。

```
[root@of-bj-op-nginx01 conf.d]# head -n 20 k8s.conf

upstream beta-tianxiao100-m {

    server 172.21.133.12:30509;
    #server 172.21.133.19:30509;
}
server {
    listen       80;
    server_name  beta-tianxiao100.hu

    access_log   /apps/log/nginx/k8s.access.log main;
    error_log    /apps/log/nginx/k8s.error.log;
    location / {
        proxy_cache off;
        proxy_pass http://beta-tianxiao100-m;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size 2000m;
        client_body_buffer_size 128k;
        proxy_connect_timeout 120;
```

发布模式

➢**镜像发布模式：**

将需要在由的程序由Dockerfile直接打包到镜像内，统一镜像版本与代码版本。

➢**代码更新模式：**

容器只包括固定的libraries和runtime，配置文件、代码等放在持久化存储之中。

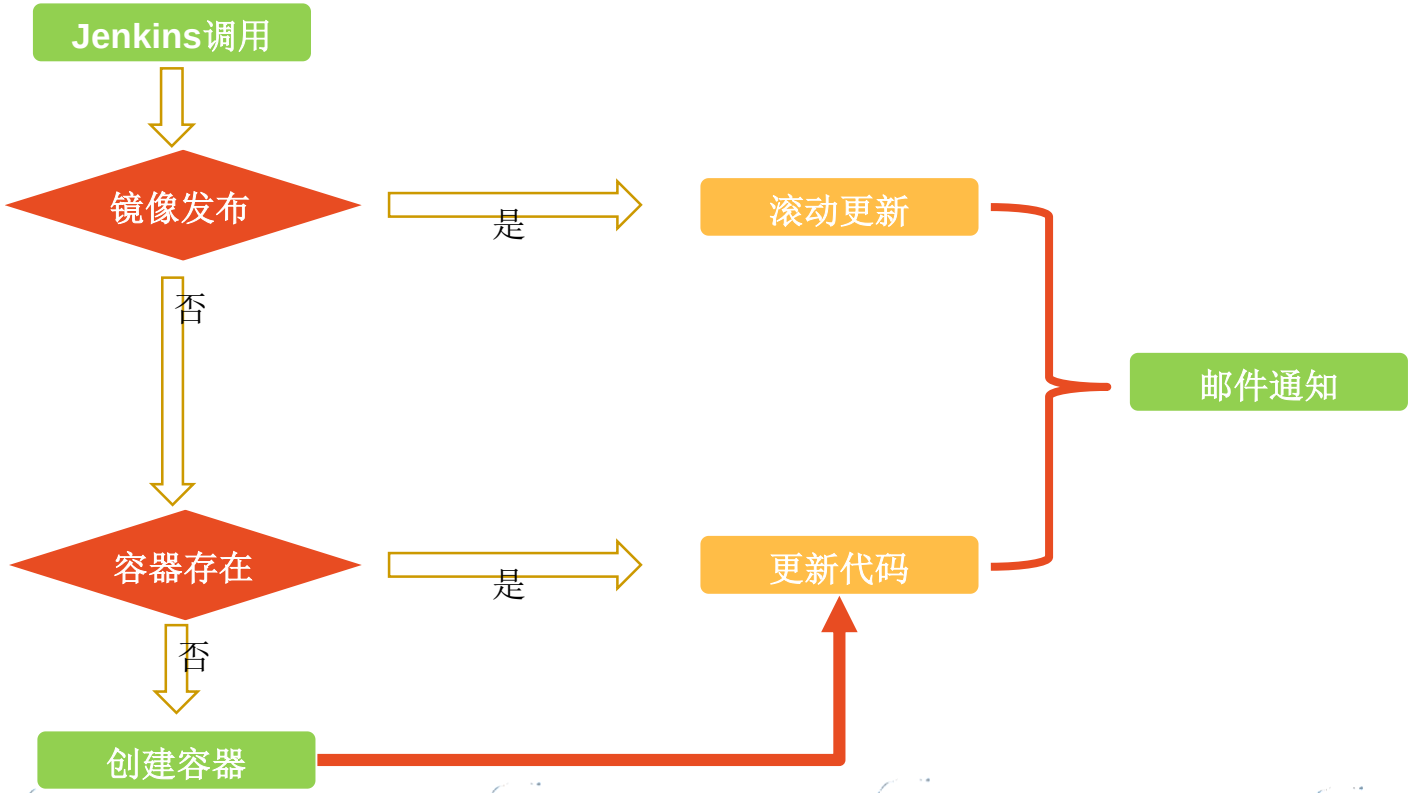
管理功能

- 查看容器
- 创建容器
- 销毁容器

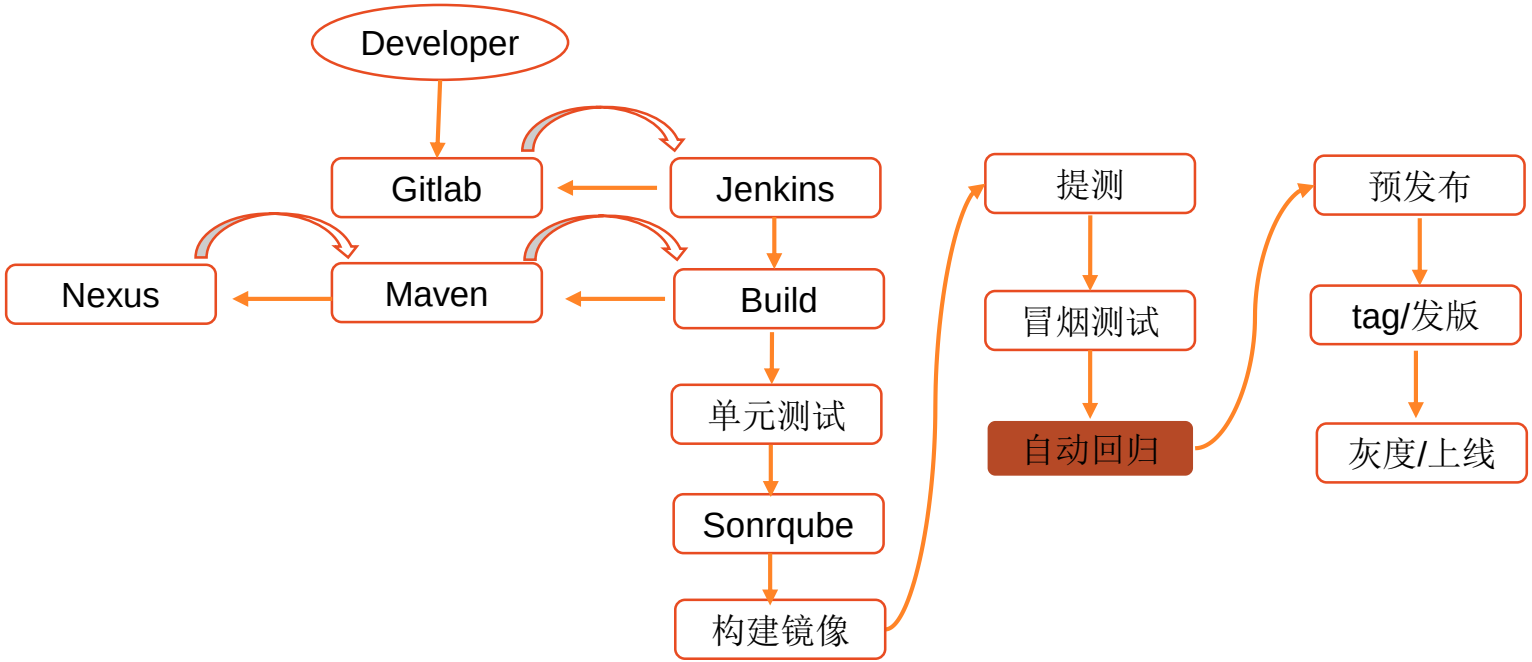
```
[root@jenkins kube]#./borg
Usage: borg [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  list_pod
  re_create   :param name: runtime env name :param...
  re_delete
  re_rebuild
  re_view
```

```
[root@jenkins kube]#./borg re_create --name=test-java
host_IP=192.168.47.100
phase=Running
ssh_port=31498,http_port=30879,https_port=30508
[root@jenkins kube]#
[root@jenkins kube]#
[root@jenkins kube]#./borg re_view --name=test-java
{"status": "sucess", "http_port": 30879, "https_port": 30508, "ssh_port": 31498, "host_ip": "192.168.47.100",
"phase": "Running"}
```

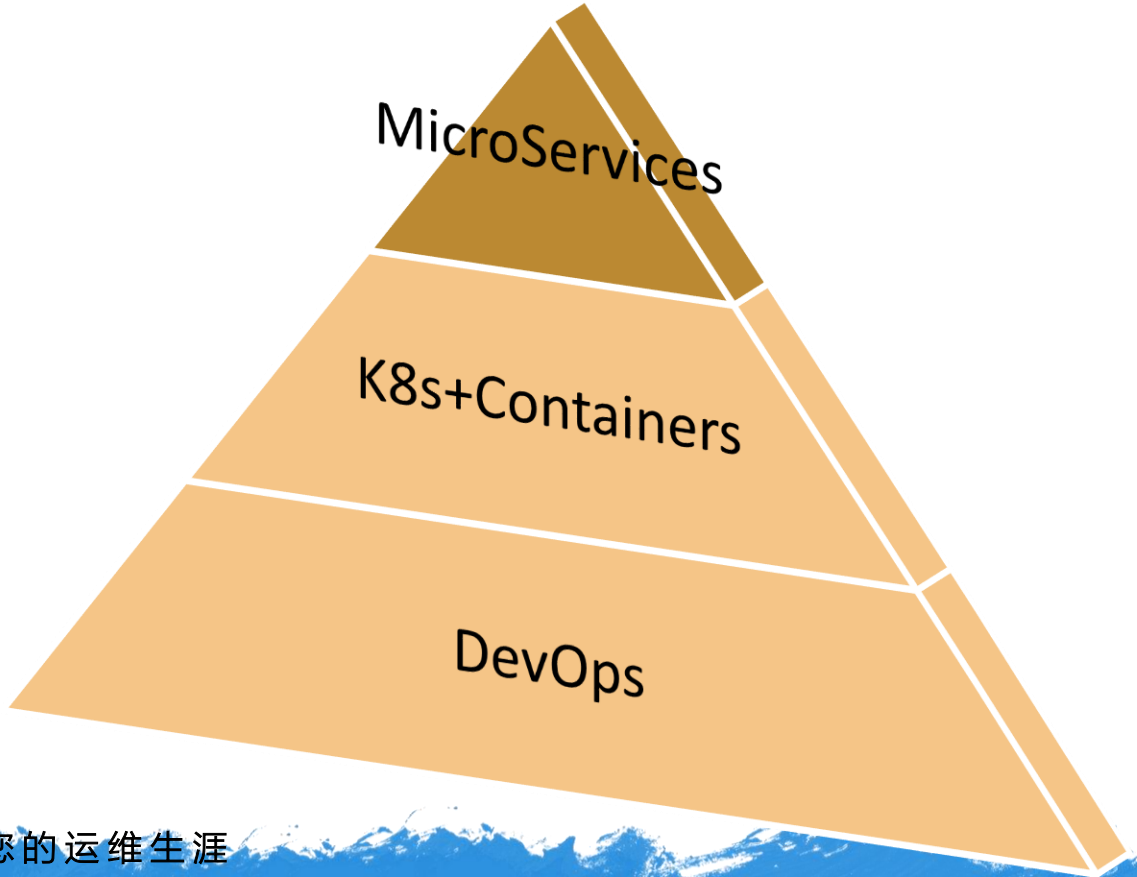
Jenkins部署脚本



部署流水线


Developer
Gitlab
Jenkins
Nexus
Maven
Build
单元测试
Sonrqube
构建镜像
提测
冒烟测试
自动回归
预发布
tag/发版
灰度/上线

生产力组合



Thanks

高效运维社区

荣誉出品