网易云 163yun.com | IT大咖说 知识分享平台

17K基于kubernetes容器云平台实践

李志伟 2017年6月

为什么选择容器云平台

面临的问题：

- 基础设施故障率高，运维自动化水平低
- 系统架构不一致，维护复杂度高
- 计算资源利用率低
- 业务迁移或水平扩展困难。
- 各类服务总量繁多，管理复杂度高。

我们的目标

- 一个高效的服务编排平台
- 一个高可用的运维平台
- 一个高度的自动化运维平台

为什么选择Kubernetes

- 高效的进程级资源调度模型，可以提高计算资源的利用率和弹性扩展能力。
- 业务容器镜像一次构建，可运行在多种运行环境，配合CI/CD,可大幅提升开发效率。
- 消除本地依赖，迁移成本低，解决服务商锁定问题。
- Kubernetes集群遵循Borg架构思想，可规范网站系统架构设计。
- 实现运维自动化

kubernetes
by Google

向容器云平台迁移的准备工作

- 搭建kubernetes集群
- 构建私有Docker镜像仓库（registry）
- 集群监控Heapster
- CI/CD基础设施（Hudson/Jenkins）
- 共享存储解决方案（ Glusterfs ）

组件选型

- Kubernetes 1.5.2　主程序
- Docker 1.10.3容器
- Flannel 0.7.0　网络组件
- Etcd　　3.1.3 数据库
- Kubernetes-Dashboard　UI
- Kubedns DNS组件
- Registry 容器私有镜像库
- Heapster　　监控
- GlusterFS　3.7.9　共享存储

容器镜像封装的基本原则

- 尽可能地设计成无状态服务
- 尽可能消除不必要的运行环境依赖
- 需要持久化的数据写入到共享存储
- 尽可能保持业务的单一性
- 控制输出到stdout和stderr的日志写入量。
- 容器中使用k8s内网dns代替ip地址配置形式
- 日志采用集中化处理方案（ELK）
- 采用独立的容器处理定时任务

关于NameSpace的使用

- 软件组件在不同的运行环境之间不会产生命名冲突。
- 通过namespace实现资源隔离（Resource Quota）。
- 开发、测试、staging、生产环境可同时运行在一个集群内部

关于service的命名规范

- 最多不能超过24个字符。
- 需要满足正则regex [a-z]([-a-z0-9]*[a-z0-9])?):的要求， 意味着首字母必须是a-z的字母，末字符不能是-，其他部分可以是字母数字和-符号。
- 命名方式：业务名-应用服务器类型-主机名 ，例如 book-mysql-m1 、book-redis-n1、book-tomcat-n1 、 book-zk-n1

关于应用健康检查

健康检查是系统故障自动发现和自我恢复最重要的手段

进程级健康检查

- 检验容器进程是否存活，进程级的健康检查都是默认启用的

业务级健康检查：

1.检查自身核心业务是否正常

2.健康检查程序执行时间要小于健康检查周期

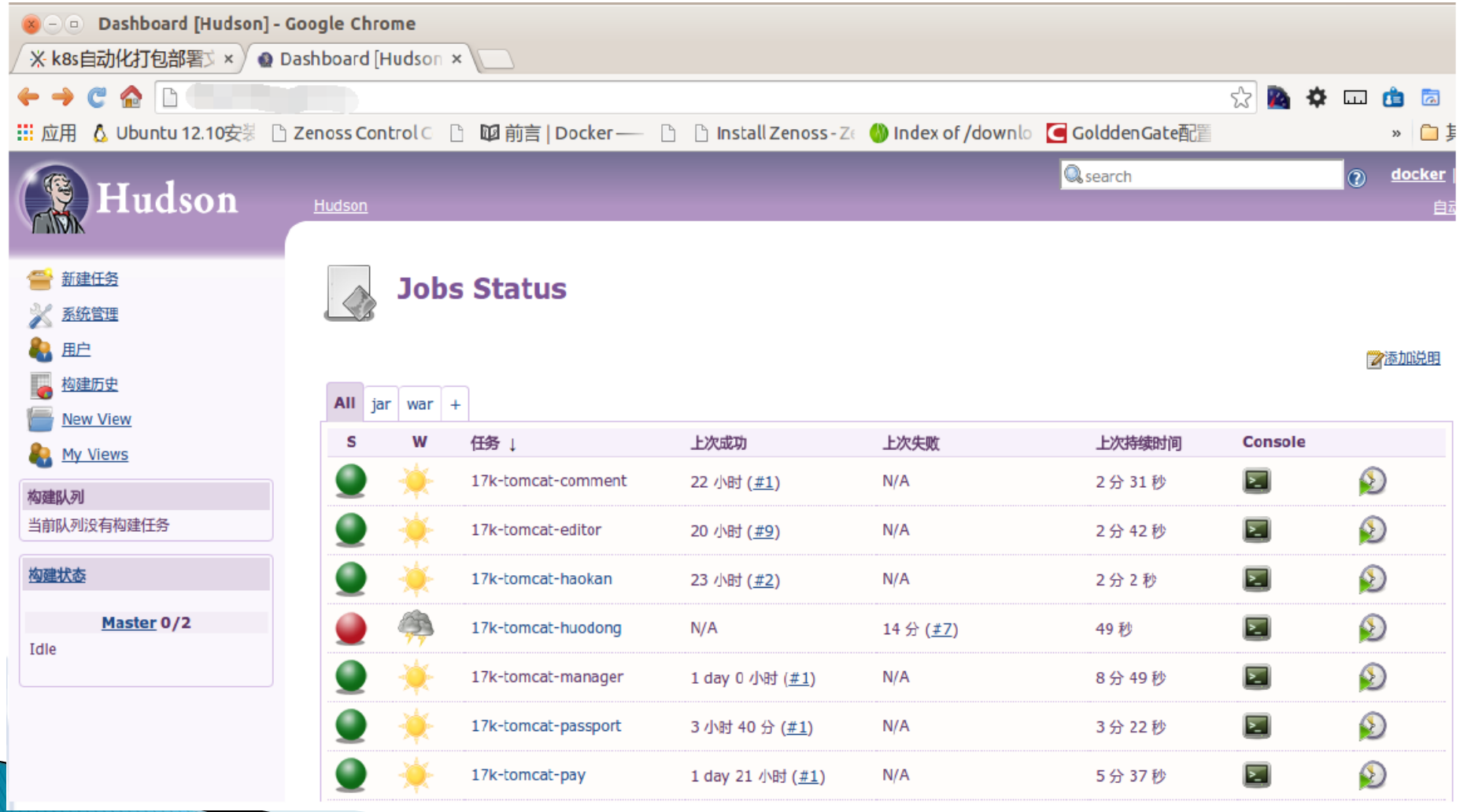
3.健康检查程序消耗资源要合理控制，避免出现服务抖动

关于健康检查程序的实现

- **WEB服务**：采用HTTPGET方式进行健康检查，需要实现一个“/healthz” URI，这个URI程序需要检查所有核心业务服务是否正常，健康检查程序还应该在异常情况下输出每一个检查项的状态明细。
- **其他网络服务**：可以采用探查容器的指定端口状态来判断容器健康状态。
- **非网络服务**：在容器内部执行特定命令，根据退出码判断容器健康状态。

HttpGetProb	TCP Socket Action	Exec Action
livenessProbe: httpGet: path: /healthz port: 8080 initialDelaySeconds: 15 timeoutSeconds: 1	livenessProbe: initialDelaySeconds: 15 timeoutSeconds: 1 tcpSocket: port: 80	livenessProbe: exec: command: - cat - /tmp/health initialDelaySeconds: 15 timeoutSeconds: 1
范例1	范例2	范例3

关于CI/CD


Dashboard [Hudson] - Google Chrome
k8s自动化打包部署
Dashboard [Hudson
应用
Ubuntu 12.10安装
Zenoss Control C
前言 | Docker——
Install Zenoss - Ze
Index of /downlo
GolddenGate配置
Hudson
search
docker
Hudson
新建任务
系统管理
用户
构建历史
New View
My Views
构建队列
当前队列没有构建任务
构建状态
Master 0/2
Idle
Jobs Status
添加说明
All
jar
war
+
S
W
任务 ↓
上次成功
上次失败
上次持续时间
Console
17k-tomcat-comment
22 小时 (#1)
N/A
2 分 31 秒
17k-tomcat-editor
20 小时 (#9)
N/A
2 分 42 秒
17k-tomcat-haokan
23 小时 (#2)
N/A
2 分 2 秒
17k-tomcat-huodong
N/A
14 分 (#7)
49 秒
17k-tomcat-manager
1 day 0 小时 (#1)
N/A
8 分 49 秒
17k-tomcat-passport
3 小时 40 分 (#1)
N/A
3 分 22 秒
17k-tomcat-pay
1 day 21 小时 (#1)
N/A
5 分 37 秒

时区问题：关于容器镜像中的时区配置的问题

- 所有镜像的/etc/localtime根据需要设置为对应的时区 如：/usr/share/zoneinfo/Asia/Shanghai。
- 实现方式采用 yaml配置文件中的volume挂载宿主机对应的localtime文件。采用yaml配置文件来控制时区信息的方式目的是为了让容器镜像的配置方式更灵活。
- 配置代码：

```
volumes:
  - name: local-time
    hostPath:
      path: /etc/localtime
      readOnly: true
```

```
volumeMounts:
    - name: local-time
          mountPath: /etc/localtime
          readOnly: true
```

网络问题：外部网络如何访问到Service

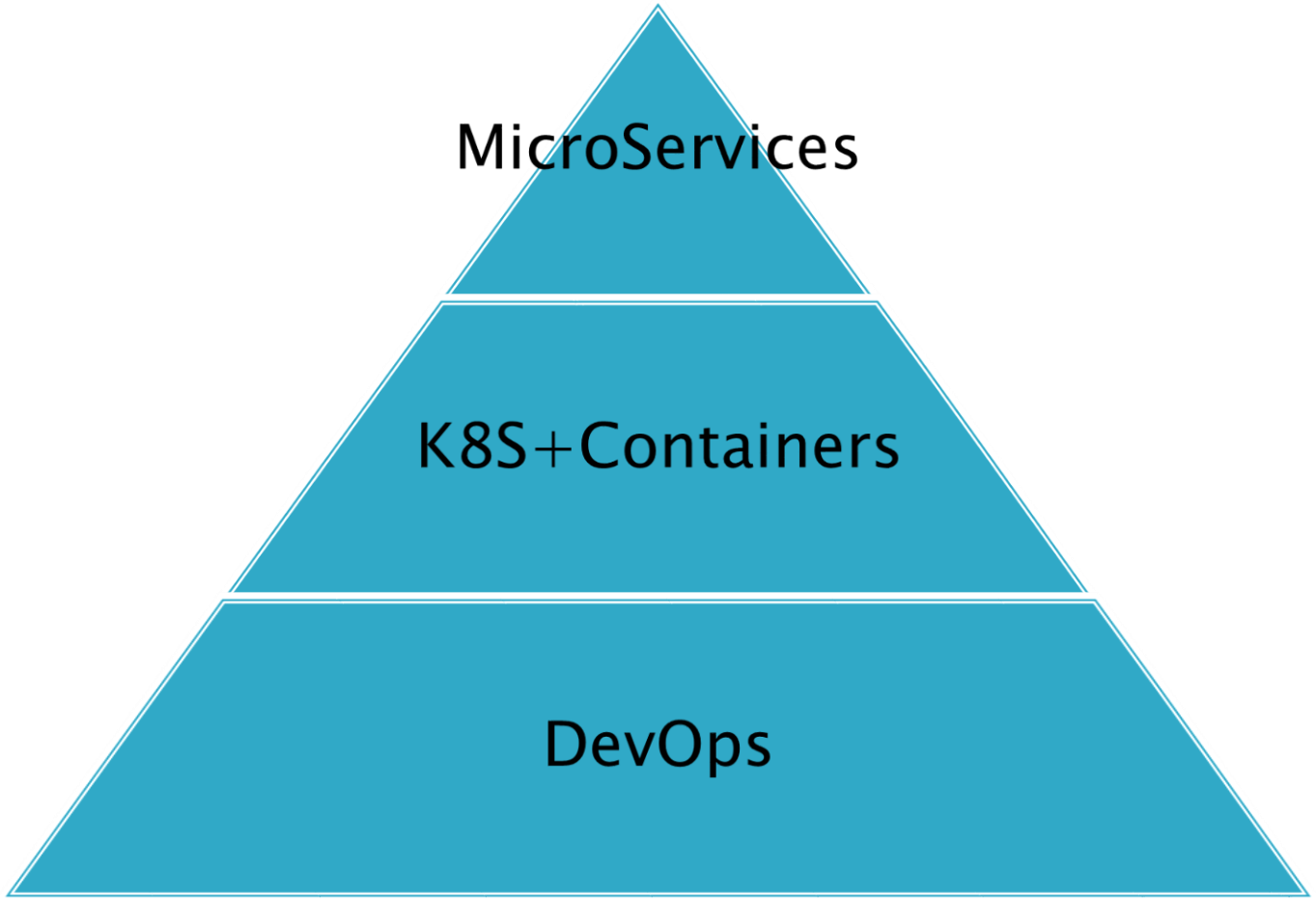
- 集群内部采用Nginx负责集群内部service的对外转发
- 通过NodePort方式暴露给外部网络

网络问题：关于nf_conntrack: table full, dropping packet 的问题

- ip_conntrack 是 Linux 系统内 NAT 的一个跟踪连接条目的模块。ip_conntrack 模块会使用一个哈希表记录 tcp 通讯协议的 established connection 记录，当这个哈希表满了的时候，便会导致 nf_conntrack: table full, dropping packet 错误。

```
# vi /etc/sysctl.conf
#哈希表项最大值
net.netfilter.nf_conntrack_max = 655350
#超时时间，默认情况下 timeout 是5天（432000秒）
net.netfilter.nf_conntrack_tcp_timeout_established = 1200
```

最佳组合



谢谢
Q&A