GOPS2017
Shenzhen

GOPS

# 2017 全球运维大会

深圳站

指导单位：数据中心联盟 Data Center Alliance

主办单位：高效运维社区 GreatOPS Community　开放运维联盟 OOPSA Open OPS Alliance

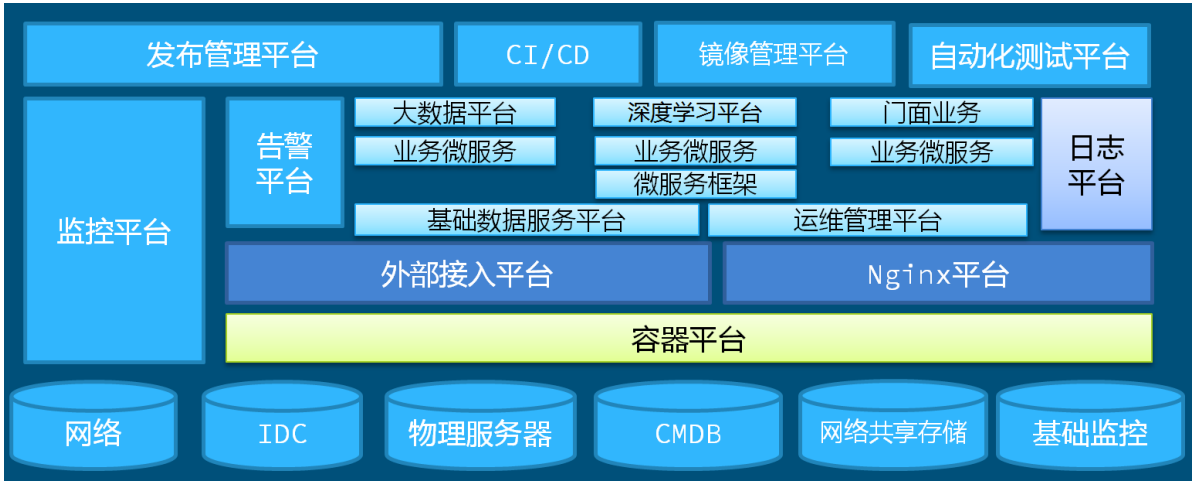
# 魅族容器云平台基于k8s的自动化运维实践

曾彬

# 目录

# 基本介绍

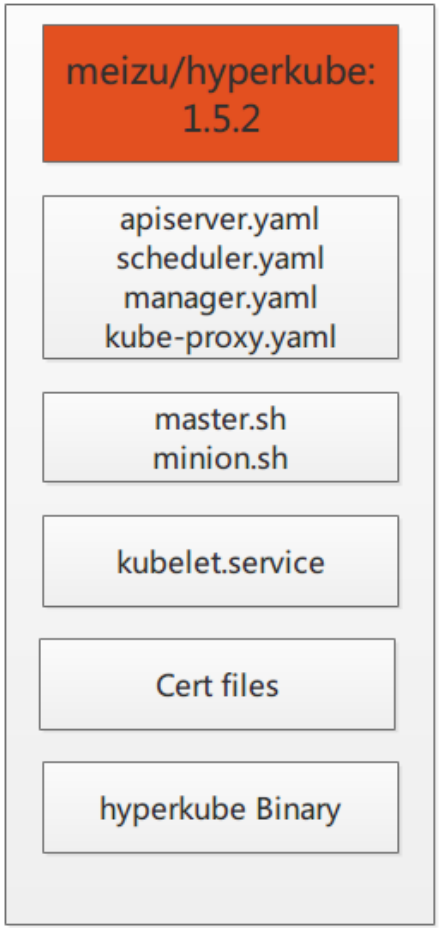
- 定位
  - 私有云平台
  - 网站业务+高级业务
  - 替换传统虚拟化
- 现状
  - 网站业务，17年完成90%迁移
  - 3个数据中心
- 运作方式
  - 低成本试错，几个人小团队
  - 紧跟k8s步伐
  - 局部和关键创新
  - 保证核心稳定，重视非功能性



# 目录

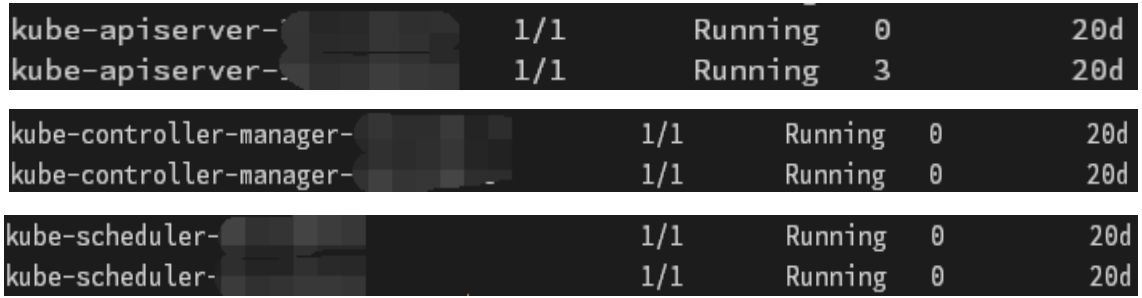
# K8s集群–单一镜像

1. 快速部署/升级，docker run一键安装
2. Hyperkube Image
   - hyperkube二进制
   - 安装/升级脚本
   - kubelet service
   - static pod yaml
   - 自签证书
3. Master安装/升级
   - apiserver/scheduler/manager
4. Minion安装/升级
   - kubelet/kube-proxy



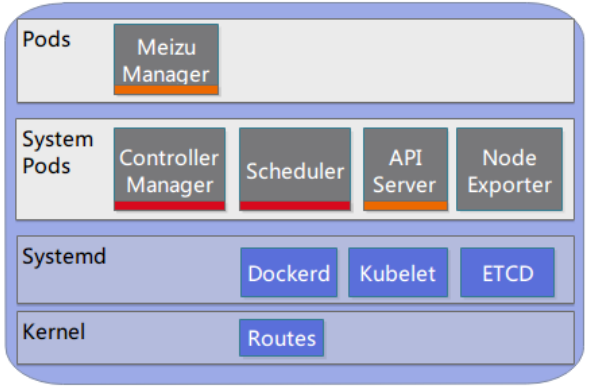
# K8s集群-Master核心组件

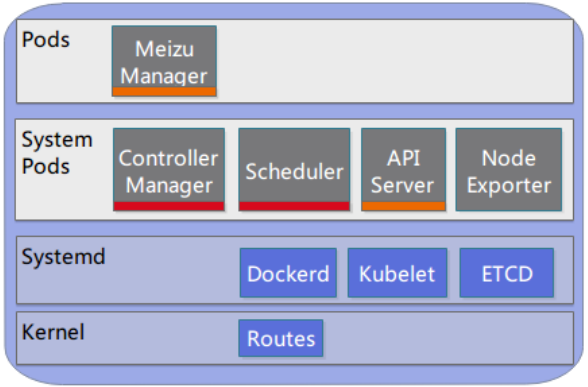
1. Static Pod保证自动加载
   - API Server
   - Controller
   - Scheduler
2. 自动修复
   - Liveness probe
   - 自动重启
3. 自动升级
   - 指定新的hyperkube version

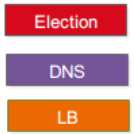


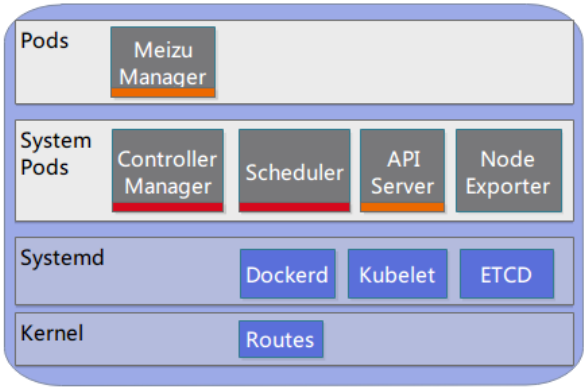
# K8s集群-Master高可用设计

1. API Server
   - LB or DNS
   - LB多活，DNS单活
2. Scheduler
   - Leader Election
3. Controller
   - Leader Election



# K8s集群-Master异常处理

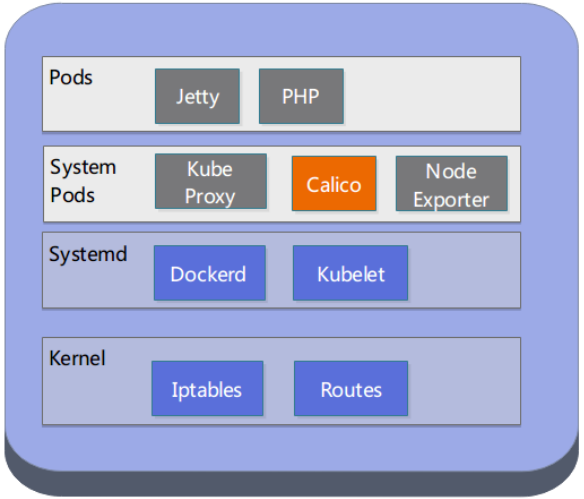
1. k8s核心Pod运行状况
   - 重启告警
   - Leader Election导致Pod重启
2. 频繁的Leader Election
   - 时钟同步?
   - 与API Server通信不稳定?
   - 重点关注Controller Manager的重启，及时介入检查有无异常状态

```
ALERT KubeControllerManagerRestart
  IF changes(kube_pod_container_status_restarts{container="kube-controller-manager"}[2m]) != 0
  LABELS {severity="warn", type="action"}
  ANNOTATIONS {description="enviroment {{ $labels.env }}s {{ $labels.pod }} is restart for {{ $
```

# K8s集群-Minion常用配置

1. 硬件
   - E5-2620 24core(HT)
   - 内存 128GB
   - Intel千兆
2. 参数调整
   - 中断相关
   - TCP backlog
   - Swapness

```
Capacity:
 alpha.kubernetes.io/nvidia-gpu:        0
 cpu:                                   32
 memory:                                131987564Ki
 pods:                                  110
Allocatable:
 alpha.kubernetes.io/nvidia-gpu:        0
 cpu:                                   32
 memory:                                131987564Ki
 pods:                                  110
System Info:
 Machine ID:                            1dd00654bd744a65a6a94879
 System UUID:                           4C4C4544-0032-4810-804D-
 Boot ID:                               fb1bbc39-020f-49c6-a9a7-
 Kernel Version:                        3.16.41
 OS Image:                              CentOS Linux 7 (Core)
 Operating System:                      linux
 Architecture:                          amd64
 Container Runtime Version:             docker://1.12.0
 Kubelet Version:                       v1.5.2
 Kube-Proxy Version:                    v1.5.2
```



# K8s集群-Minion-存储

1. Docker存储
   - DeviceMapper
   - Direct LVM
   - 实际用量并不大

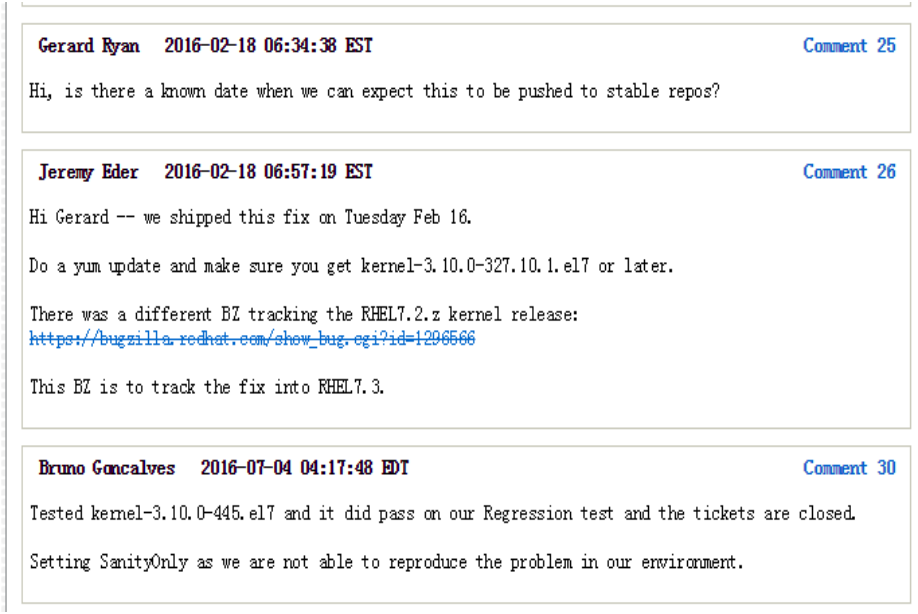
```
└─sda5                          198.2G  0 part
  ├─VG-docker--pool_tmeta         204M  0 lvm
  │ └─VG-docker--pool           188.1G  0 lvm
```

2. 本地普通存储
   - 日志/临时文件等存储于EmptyDir，EmptyDir映射到本地目录
   - 大量读写
   - 修改EmptyDir落地路径 --root-dir=/data/kubelet (默认 /var/lib/kubelet)
   - 考虑使用普通分区
   - 使用lvm参数设置不当导致meta full

```
sdb
          8:16   0   1.1T  0 disk
└─sdb1
          8:17   0   1.1T  0 part /data
```

# K8s集群-Minion-存储踩坑

- Kernel issue
  - **Bug 1292481 - device mapper hung tasks on an openshift/docker system**
  - https://bugzilla.redhat.com/show_bug.cgi?id=1292481
- So,经常更新内核



**Gerard Ryan** 2016-02-18 06:34:38 EST Comment 25

Hi, is there a known date when we can expect this to be pushed to stable repos?

**Jeremy Eder** 2016-02-18 06:57:19 EST Comment 26

Hi Gerard -- we shipped this fix on Tuesday Feb 16.

Do a yum update and make sure you get kernel-3.10.0-327.10.1.el7 or later.

There was a different BZ tracking the RHEL7.2.z kernel release:
https://bugzilla.redhat.com/show_bug.cgi?id=1296566

This BZ is to track the fix into RHEL7.3.

**Bruno Goncalves** 2016-07-04 04:17:48 EDT Comment 30

Tested kernel-3.10.0-445.el7 and it did pass on our Regression test and the tickets are closed.

Setting SanityOnly as we are not able to reproduce the problem in our environment.

# K8s集群-Minion-内核

- 版本
  - OS Centos 7.2，3.10
  - **bugfix back porting**
  - 4.1以后,**hacking tcp_v4_syn_recv_sock 会有问题**
  - 我们的选择是**3.16**

Longterm release kernels

| Version | Maintainer | Released | Projected EOL |
|---|---|---|---|
| 4.9 | Greg Kroah-Hartman | 2016-12-11 | Jan, 2019 |
| 4.4 | Greg Kroah-Hartman | 2016-01-10 | Feb, 2018 |
| 4.1 | Sasha Levin | 2015-06-21 | Sep, 2017 |
| 3.16 | Ben Hutchings | 2014-08-03 | Apr, 2020 |
| 3.12 | Jiri Slaby | 2013-11-03 | May, 2017 |
| 3.10 | Willy Tarreau | 2013-06-30 | Oct, 2017 |
| 3.4 | Li Zefan | 2012-05-20 | Apr, 2017 |
| 3.2 | Ben Hutchings | 2012-01-04 | May, 2018 |

# K8s集群-Minion-Label管理

1. 标签

```
xx.xx.52.22   Ready   80d   calico=v2,kubernetes.io/hostname=xx.xx.52.22,logging=N3,rack=0702
```

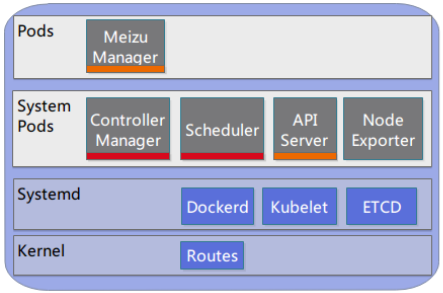
   - 结合CMDB，为Minion打上标签
   - rack/网络/机型/功能等

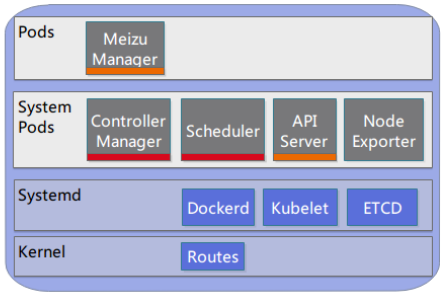
2. 调度

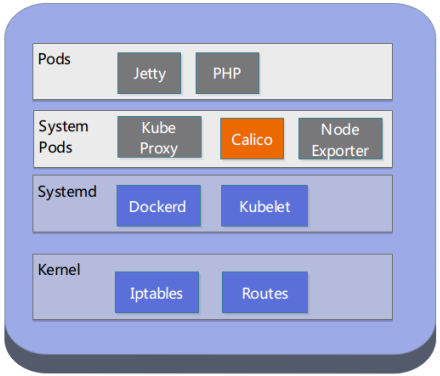
   - Node Affinity
   - Pod Anti-Affinity

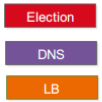
# K8s集群–总结

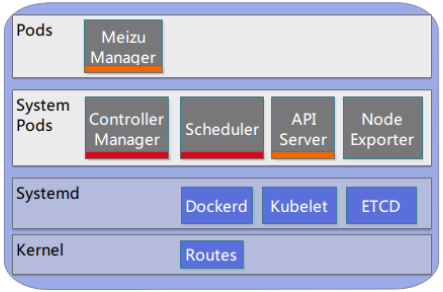
1. 部署便利度
   - 集群部署
   - 集群升级
2. 高可用
   - 核心组件进程
   - 核心组件健康检测
   - 核心组件冗余设计
3. 存储
   - 容器存储
   - 非容器存储
4. 内核
   - 版本
   - 参数
5. 标签管理



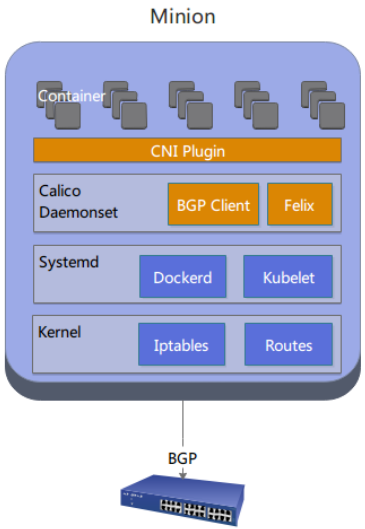



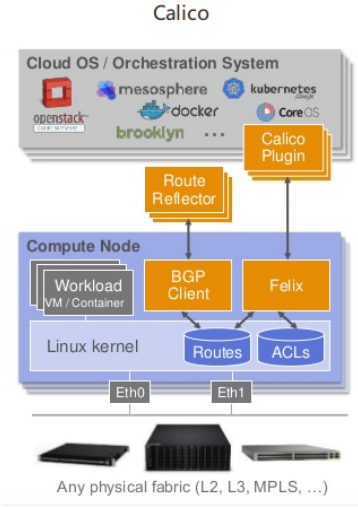



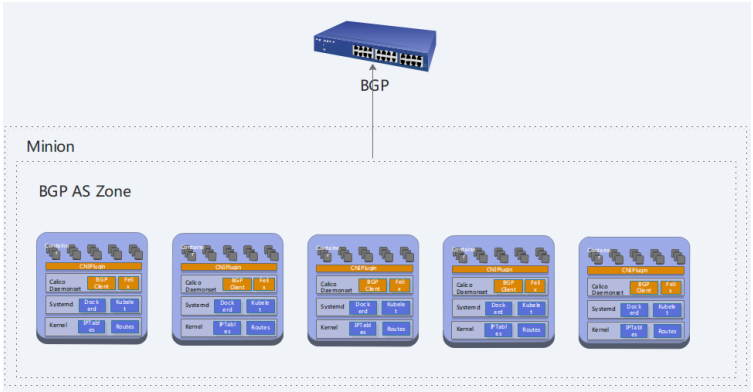
# 目录

# 网络-整体

1. 方案
   - Calico
2. 控制层面
   - BGP No Mesh
   - Bird对接核心路由设备
3. 数据层面
   - 3层路由
   - Netfilter Conntrack
   - Forward Chain
4. 部署
   - Daemonset



# 网络-优化点

1. 配置
   - 合理的Conntrack参数设置
   - 使用headless service

```
--conntrack-max-per-core int32
--conntrack-min int32

--conntrack-tcp-timeout-close-wait duration
--conntrack-tcp-timeout-established duration
```

2. 监控告警
   - 监控conntrack用量
   - Calico Prometheus Met

```
# HELP node_nf_conntrack_entries Number of currently allocated flow entries for connection tracking.
# TYPE node_nf_conntrack_entries gauge
node_nf_conntrack_entries 406
# HELP node_nf_conntrack_entries_limit Maximum size of connection tracking table.
# TYPE node_nf_conntrack_entries_limit gauge
node_nf_conntrack_entries_limit 655350
```

3. Readness ping
   - 容器Ping 核心交换机，确保网络联通性
   - Readness检查fail，endpoint下线，并触发告警

# 网络-优化conntrack

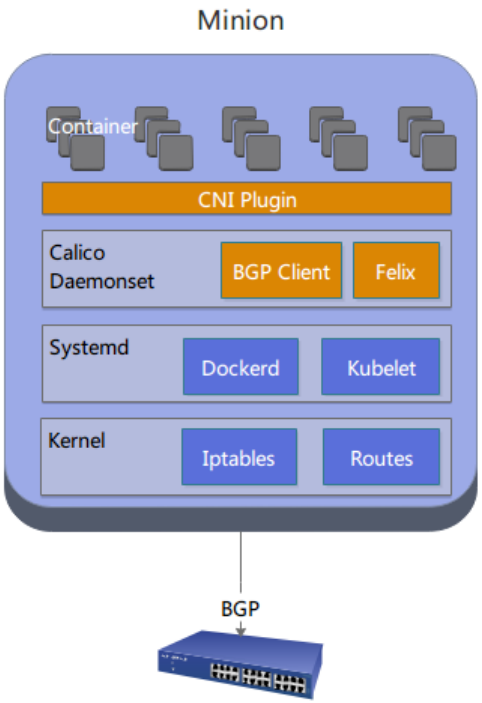
1. Bypass Tracking
   - Tracking多用于访问控制
   - LVS过来的流量大，信任度高，Tracking意义不大
   - Bypass LVS过来的流量
2. 方法
   - felix insert改为append，保证自定义规则的最高优先级
   - 在raw表PREROUTING和filter表FORWARD链添加规则

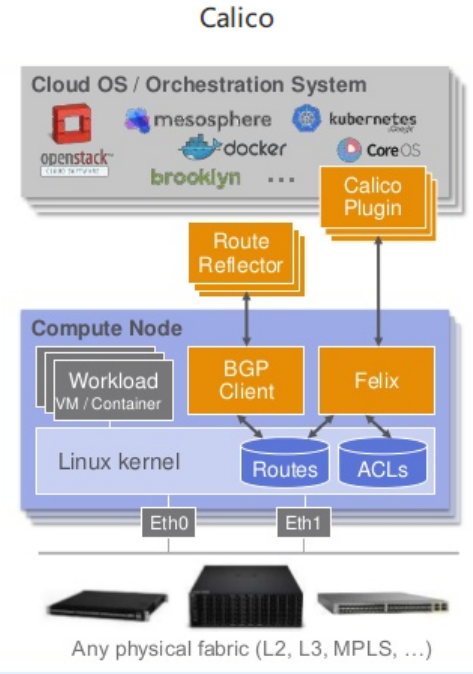
```
[root@ bin]iptables -t filter -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.  .0/24          anywhere
ACCEPT     all  --  anywhere             10.  0/24
ACCEPT     all  --  10.  .0/24           anywhere
ACCEPT     all  --  anywhere             10.  .0/24
felix-FORWARD  all  --  anywhere             anywhere
```

```
[root@10. bin]iptables -t raw -L PREROUTING
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
CT         all  --  anywhere             10.  0/24          NOTRACK
CT         all  --  10.  0/24            anywhere           NOTRACK
CT         all  --  10.  0/24            anywhere           NOTRACK
CT         all  --  anywhere             10.  .0/24         NOTRACK
```

# 网络-总结

1. 稳定性
   - mesh vs no mesh
2. 性能优化
   - Netfilter Conntrack
3. 异常处理
   - Pod主动检测网络
   - Calico整体健康告警

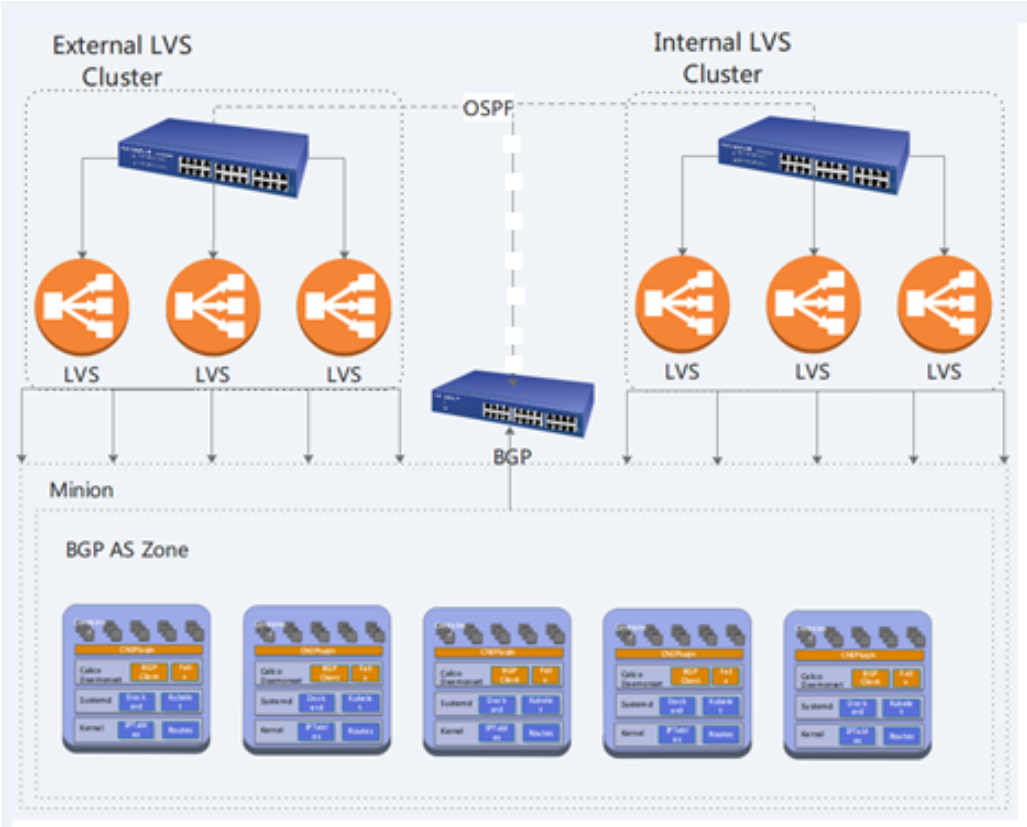


# 目录

# 外部访问-4层

1. LVS集群
   - Alibaba LVS
   - FullNat
   - 内/外集群
2. 流量进出
   - Client<->LVS<->Minion<->Pod
   - Minion上不做Nat



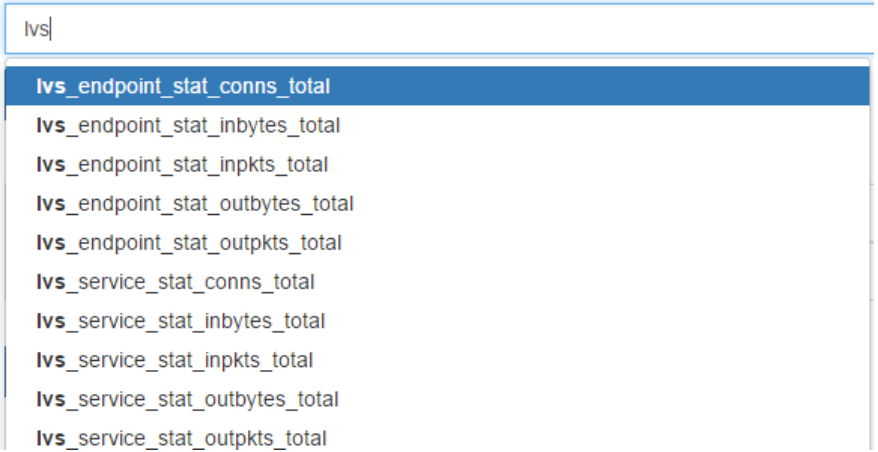
# 外部访问-自动配置

- 自动化配置
  - VIP OSPF宣告（预先宣告，减少变更）
  - LVS配置
    - Virtual Service->K8s Service->Endpoints

```
apiVersion: v1
data:
  xx.xx.x3.225-80: yy.yy.y0.130:80, yy.yy.y51.20:80
  xx.xx.x3.225-443: yy.yy.y0.130:443, yy.yy.y51.20:443
  xx.xx.x3.226: gslb/nginx-ingress-controller-new-lb:s
  xx.xx.x3.227: openapi/openapi-nginx-lb:s
kind: ConfigMap
```

# 外部访问-4层-流量采集

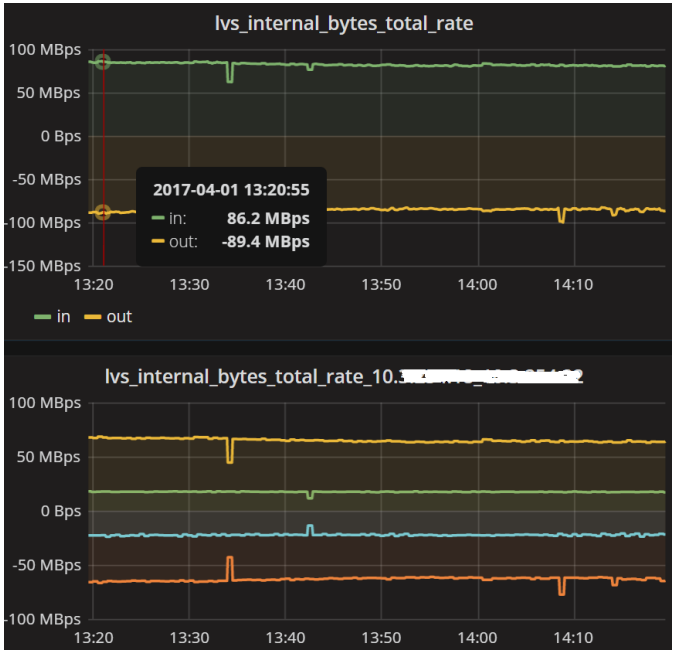
- 流量采集
  - 暴露Prometheus Metrics
  - 业务信息(命名空间，服务名)
  - Virtual Service
  - RealServer(Endpoint)

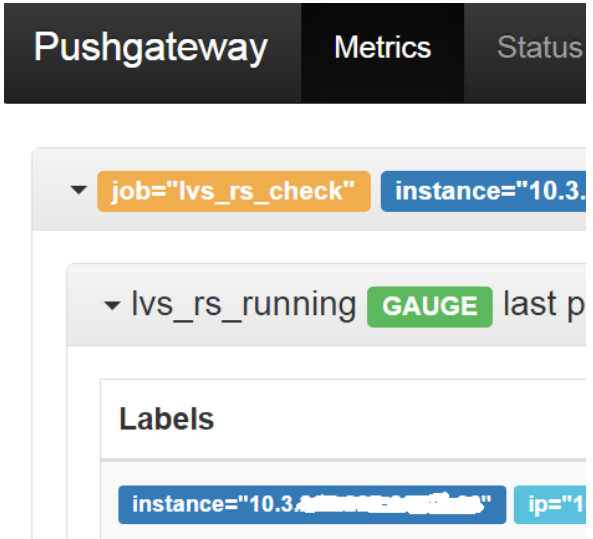



lvs_endpoint_stat_conns_total{endpoint="10.[illegible]",env="GGZ-YG",instance="10.[illegible]",job="external-lvs",linkaddr="1[illegible]",namespace="g[illegible]b",port="443",service="nginx-ingress-controller-new-lb",vport="443"} 5026006

# 外部访问-4层-监控告警

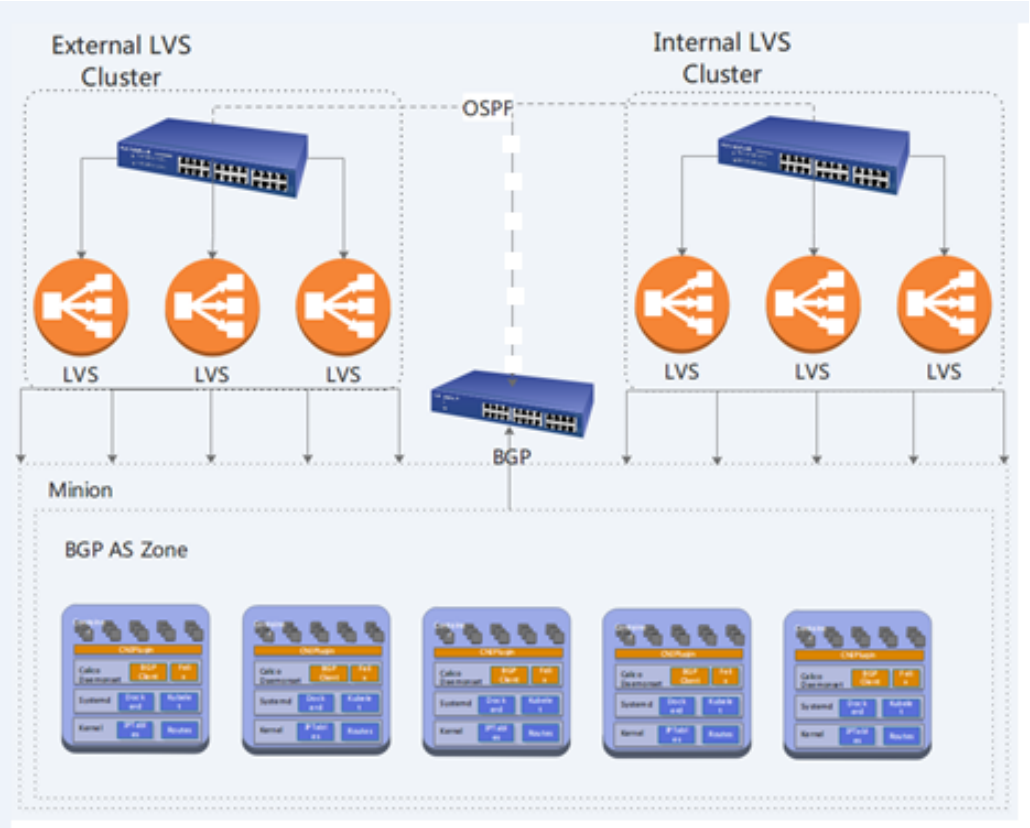
1. 监控
   - Grafana可视化
   - 聚合集群数据
2. 告警
   - 流量异常
   - RealServer MISC_CHECK 检测异常



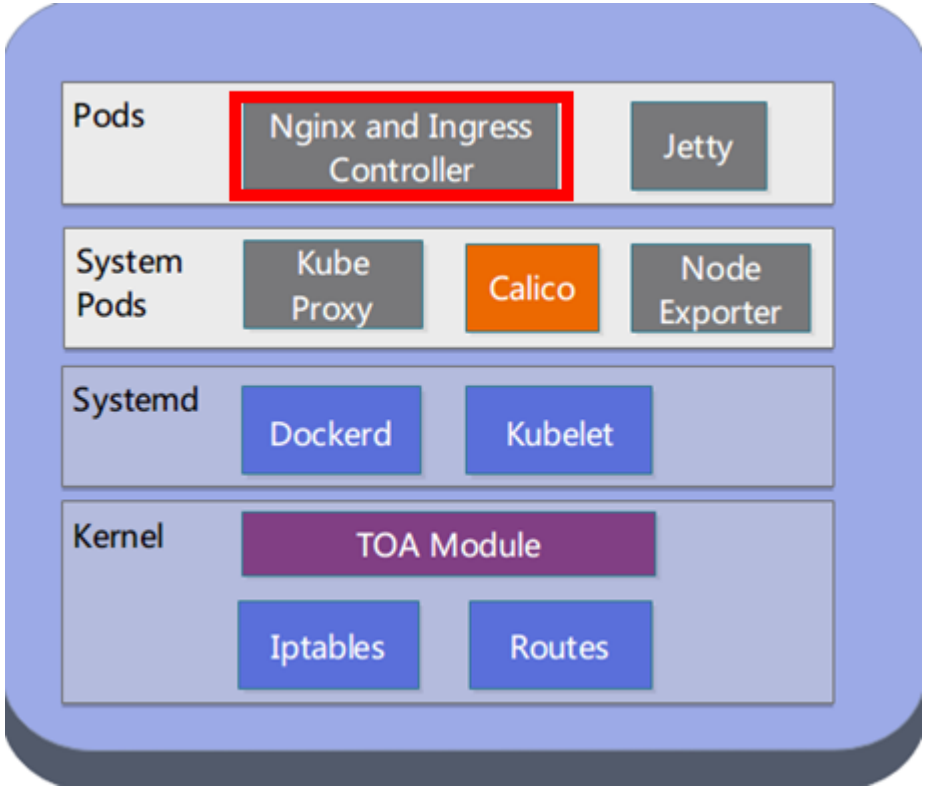
# 外部访问-4层-总结

1. 性能/扩展性
   - 方案选型
2. 与容器连通
   - 路由
3. 自动化运维
   - 尽量减少手工配置
4. 监控告警
   - 数据聚合
   - 检测异常



# 外部访问-7层-Nginx

- Nginx
  - Nginx + Ingress Controller
  - 业务专属
  - 容器化
  - 自动伸缩



# 外部访问-Nginx-获取用户地址

- 获取client地址
  - 默认获取到的是LVS的Local ip+Port，要从TCP Option里”挖出来”
  - TOA补丁只支持2.6.32
  - 移植到3.16内核

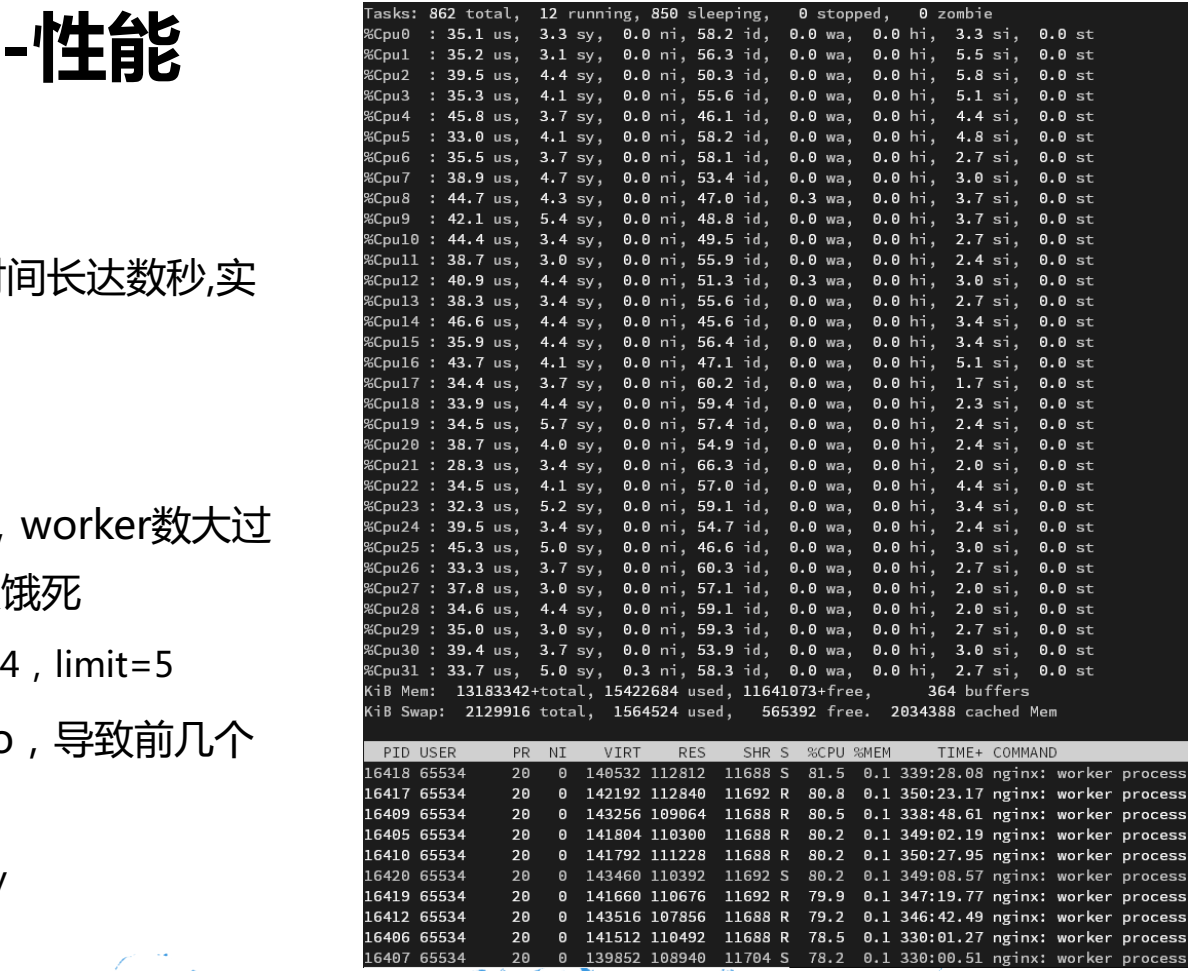
```
Transmission Control Protocol, Src Port: 5015 (5015), Dst Port: 8080 (8080), Seq: 1, Ack: 1, Len: 0
    Source Port: 5015
    Destination Port: 8080
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header Length: 28 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    [Window size scaling factor: -2 (no window scaling used)]
  ▸ Checksum: 0x2fb2 [validation disabled]
    Urgent pointer: 0
  ▾ Options: (8 bytes), Experimental
    ▾ TCP Option - Experimental: fe08b5f3ac10108a
        Kind: RFC3692-style Experiment 2 (254)
        Length: 8
        Magic Number: 0xb5f3
  ▸ [SEQ/ACK analysis]

0000  00 00 00 01 00 06 00 50  56 92 53 0f 00 00 08 00   .......P V.S.....
0010  45 00 00 30 ee 82 40 00  3c 06 c6 f7 ac 11 20 82   E..0..@. <..... .
0020  c0 a8 fc 11 13 97 1f 90  c7 69 a0 5d fd e7 5b 4e   ........ .i.]..[N
0030  70 10 72 10 2f b2 00 00  fe 08 b5 f3 ac 10 10 8a   p.r./... ........
```

```
[root@[redacted]cker-node-[redacted] toa-master]# uname -r
3.16.41
[root@[redacted]-docker-node-[redacted] toa-master]# modinfo toa.ko
filename:       /root/toa-master/toa.ko
license:        GPL
srcversion:     BFA597D0C6615463A995C7D
depends:
vermagic:       3.16.41 SMP mod_unload modversions
```

```
[redacted] ~]$ cat /proc/net/toa_stats
                    CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7       CPU8       CPU9       CPU
    CPU16      CPU17      CPU18      CPU19      CPU20      CPU21      CPU22      CPU23      CPU24      CPU25      CPU26      CPU27
syn_recv_sock_toa       : 210722769  229438508  233346715  237758960  242640661  252758640  235125087  245413511  169267640  166447870  165505915
8  171039454  173819700  171210881  174446579  173172526  193848363  189489481  158893341  162995508  159397445  160723939  168867308  173304645
syn_recv_sock_no_toa    :    146094     175650     174723     168299     177199     190807     163891     172783     130418     124577     121835
9     112938     109978     112969     112219     120118     134115     137762     106911     117561     113452     124026     161262     234801
getname_toa_ok          : 272245758  294641947  306076548  319348947  336113338  366164731  345027318  377157575  143122568  134549520  131681328
9  138929167  146699347  139310452  148320882  145908581  178219239  165723624  105366604  122985066  112452022  116746135  145473735  160868420
getname_toa_mismatch    :          0          0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0          0          0          0
getname_toa_bypass      :          0          0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0          0          0          0
getname_toa_empty       : 241241563  262153225  273675863  286299214  301920384  331824612  313484582  345615230  134632947  127203429  124736255
2  148897891  155287161  147717479  156235587  154023483  185648664  174120399  115294143  132404895  121764088  126222961  154041546  169274242
```

# 外部访问-Nginx-性能

1. 高延迟问题
   - 观察到upstream的响应时间长达数秒,实际后端服务一直正常
2. 解决方案
   - worker_processes auto，worker数大过limit较多，worker可能被饿死
     - 合理设置，例如worker=4，limit=5
   - worker_cpu_affinity auto，导致前几个核占用率高
     - 禁用worker_cpu_affinity



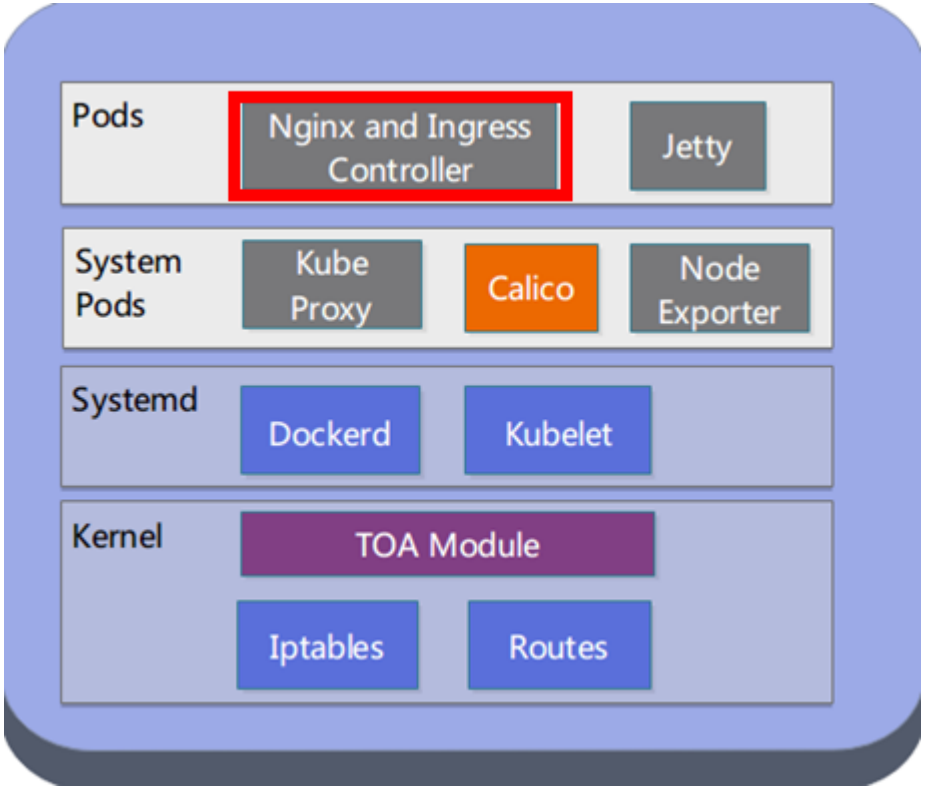
# 外部访问-nginx-动态伸缩

1. 缩容
   - 从容退出
     - 稍等一会，确保前端LB已摘除endpoint
     - 确保LB不会送来新的请求，把未完成请求处理完，再优雅退出
2. 扩容
   - 使用**readinessProbe initialDelaySeconds**留足初始化时间
3. 注意
   - 设置足够大的timeout， 若Liveness/Readiness Probe轻易失败，可能引起雪崩效应
   - 根据应用特点和实际运行情况，优化cpu request和hpa

```
lifecycle:
  preStop:
    exec:
      command:
      - /bin/sh
      - -c
      - sleep 8 && /ensure_nginx_quit.sh
```

# 外部访问-7层-总结

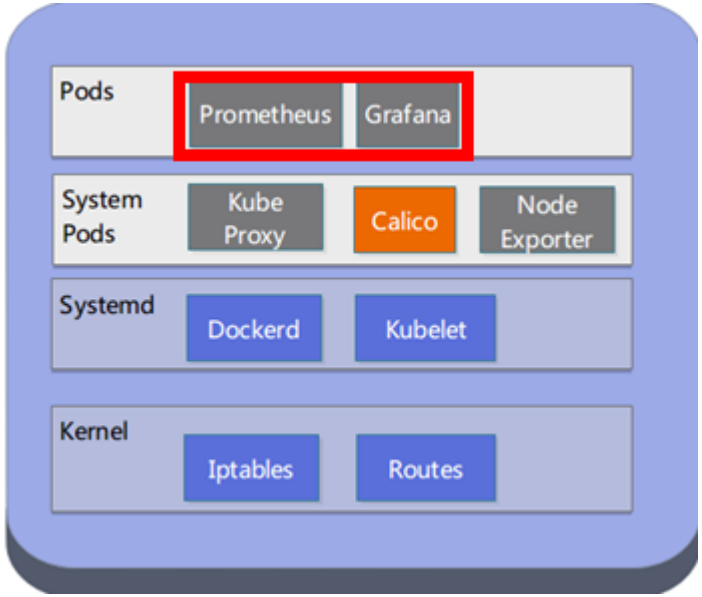
1. 降低成本
   - 提升资源利用率
   - 标准化/规范化
2. 动态化
   - Nginx动态伸缩，注意柔性
   - 后端upstream伸缩同样注意柔性
3. 满足业务需求
   - 灰度发布
   - 获取客户端IP
   - 配置定制



# 目录

# 监控-Prometheus

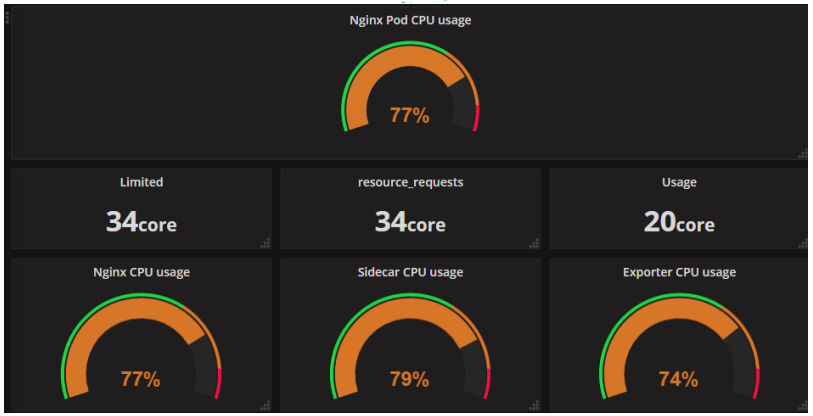
- 部署
  - Daemonset，调度到特定机型
  - 部署冗余副本
- 配置
  - ConfigMap
  - 按用途拆分

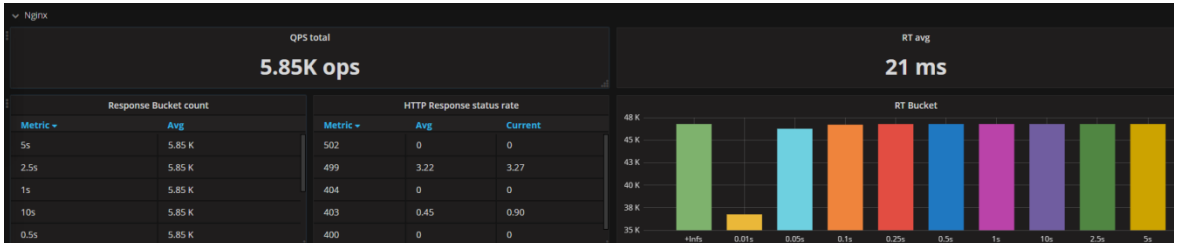


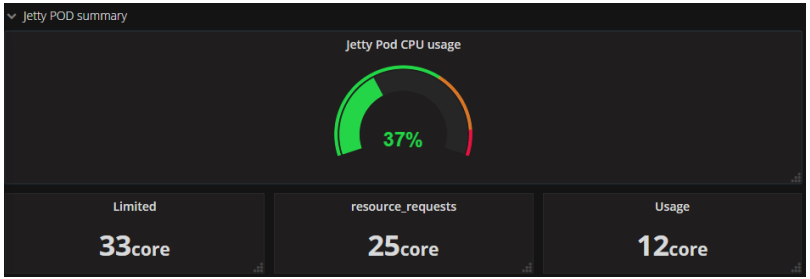

```
[devbasic@                    -master-1      ~]$ kubectl get configmap --namespace=monitor
NAME                            DATA      AGE
prometheus-alert-rules          1         28d
prometheus-alertmanager-config  1         134d
prometheus-config               2         132d
prometheus-config-alone         1         28d
prometheus-recording-rules      1         29d
```

# 监控-业务“硬”指标

1. 从业务Nginx获取
   - 当前QPS
   - HTTP Code分布
   - upstream平均rt
   - upstream rt分布
2. 计算资源消耗状况
   - Nginx
   - 后端Jetty

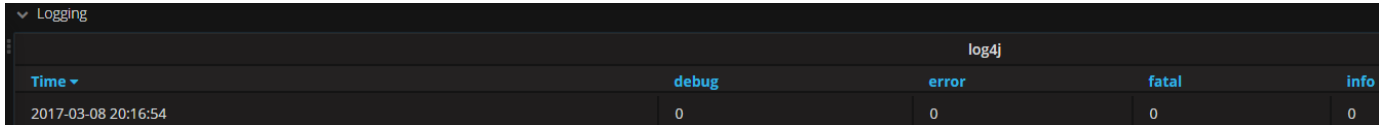


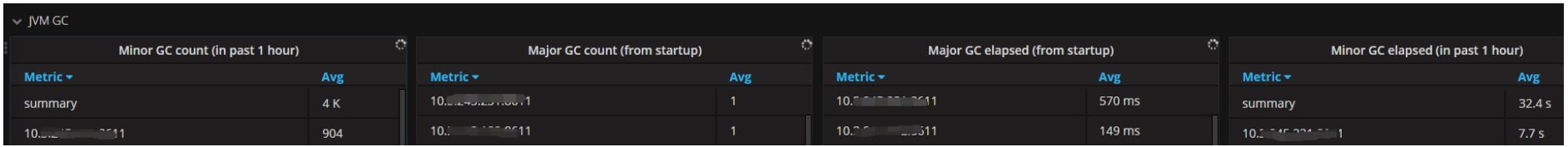



# 监控-业务-“软”指标

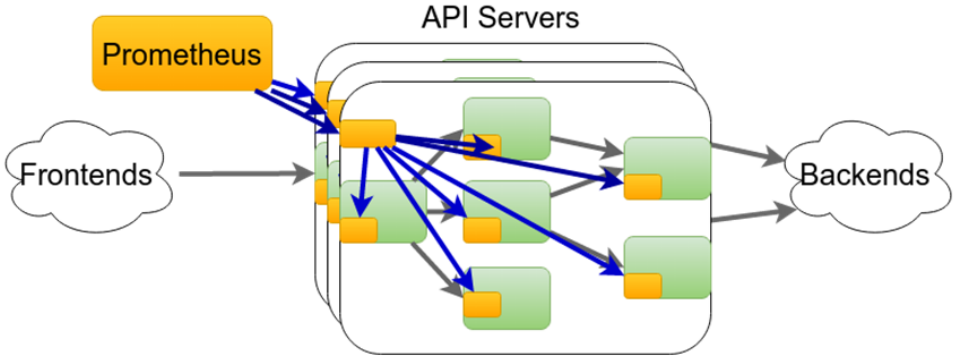
1. “内部”指标
   - JVM
   - Logging Counter
   - 业务指标
2. 暴露方式
   - 低侵入，可拔插
     - Java Agent运行http Prometheus Handler
     - -javaagent:lib/meizu-metric-agent-0.0.4.jar



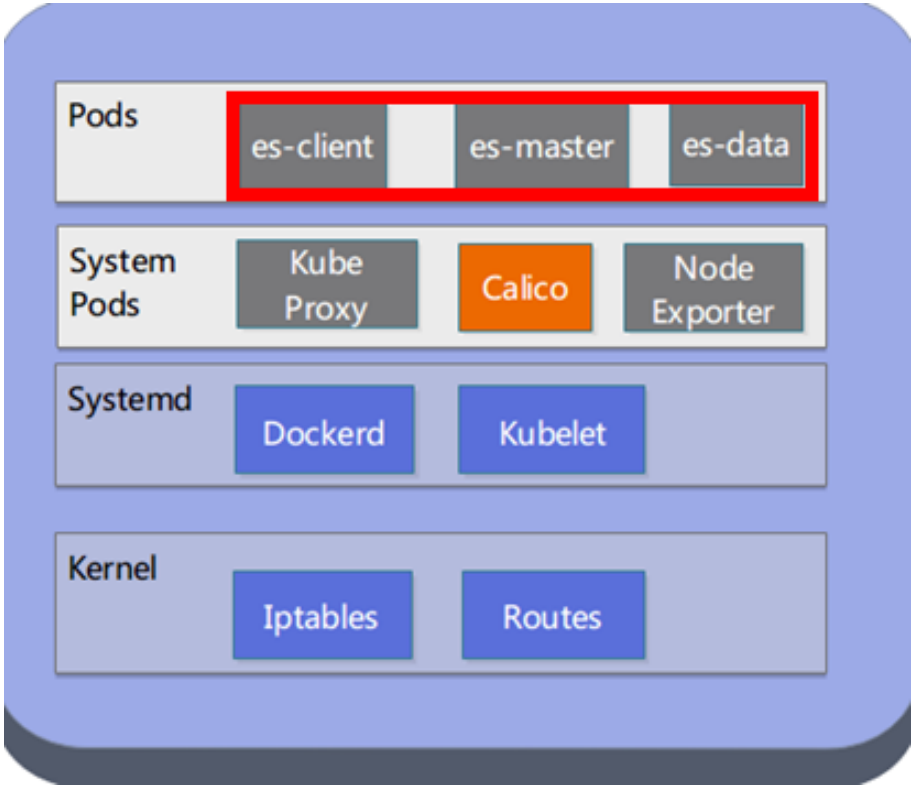
# 监控-Prometheus-总结

1. 暴露方式
   - pull
   - push
2. 部署
   - 易升级
   - 调度到特定机型
   - 冗余副本
3. 效率优化
   - 查询速度
   - 告警准确度
   - 业务监控面板



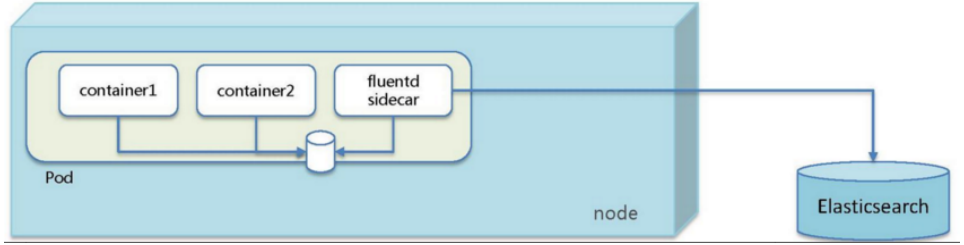

# 日志-集群部署

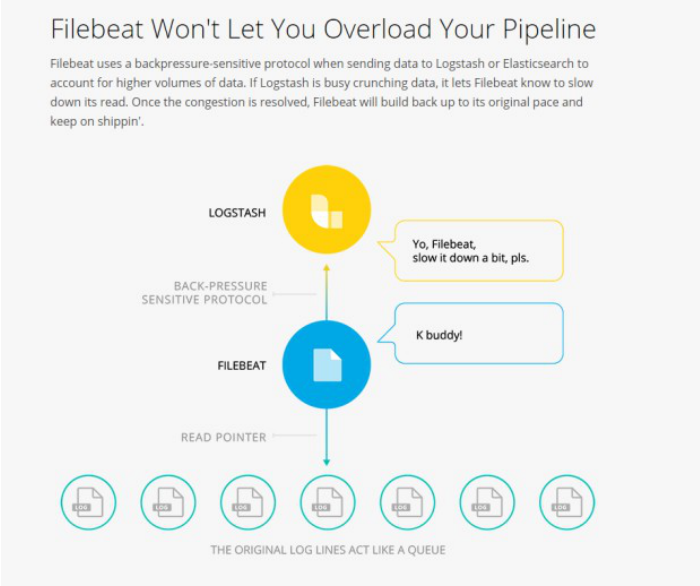
1. 部署
   - 特定机型
   - 快速扩容
   - 容器化
   - 线程数匹配CPU Limit
2. 收集效率



# 日志-收集方法

1. Logging Sidecar
   - Fluentd agent
   - 业务容器和agent使用EmptyDir非持久化存储
   - 日志文件滚动，防止占用过多空间
   - 直接发送到ElasticSearch
2. 问题
   - 日志滞后，抓包发现是agent发送落后
   - 优化bucket
   - 记录传输进度的pos文件，放在EmptyDir
   - 用**filebeat**取代

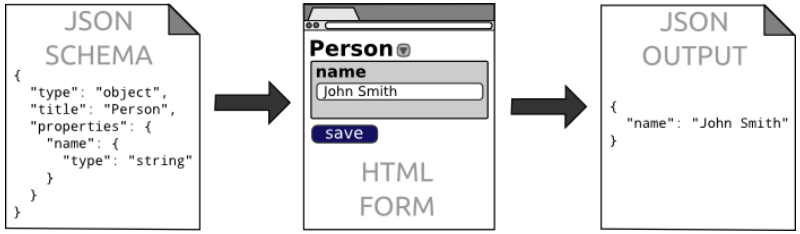


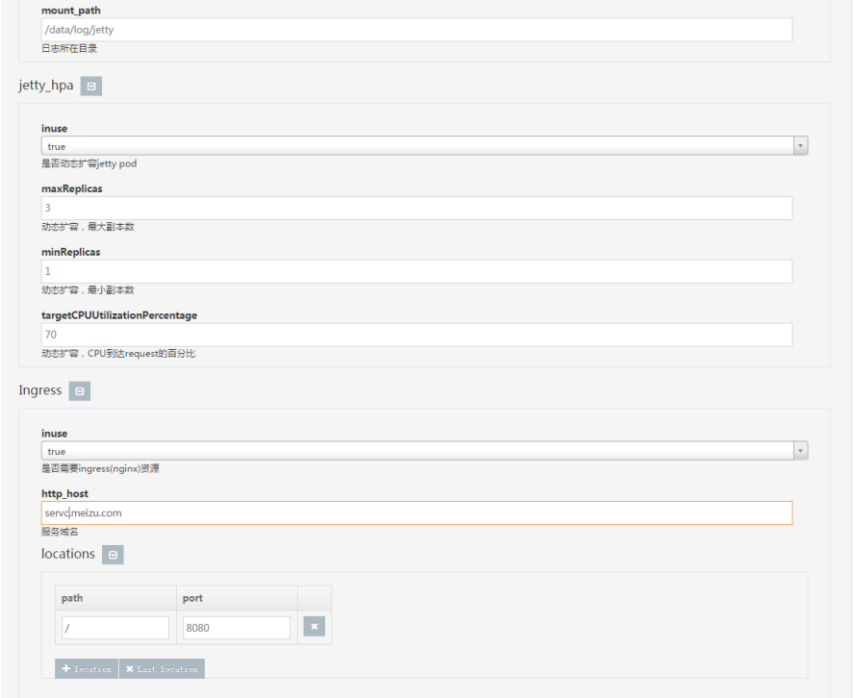



# 目录

# 业务部署-实现

1. 生成K8s Resource Yaml
    1. JSON SCHEMA生成界面
    2. 界面输入生成JSON
    3. JSON+Template= Resource Yaml
2. 执行动作
    1. Ansible调用kubectl
    2. 获取结果输出



# 业务部署-实现

- Schema
  - 分组描述所有参数
  - 类型
  - 文字说明
  - 默认值

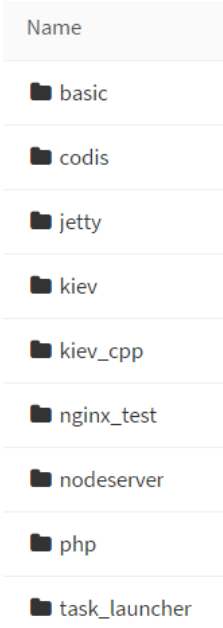
schema.json 12.9 KB

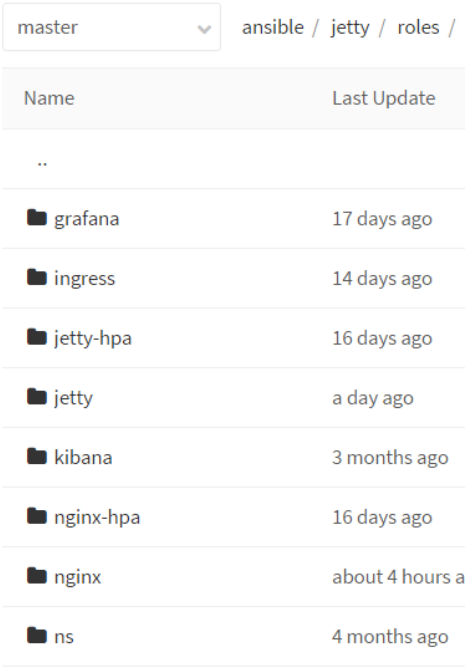
```
{
  "title": "Jetty-Dashboard",
  "type": "object",
  "properties": {
     "global": {
       "title": "Global",
          "type": "object",
          "properties": {
             "create_ns": {
                "type": "boolean",
                "description": "创建namespace",
                "enum": [
                   false,
                   true
                ],
                "default": "{{global.create_ns}}"
             },
             "org": {
                "type": "string",
                "description": "组织名",
                "default": "{{global.org}}"
             },
             "registry": {
                "type": "string",
                "description": "镜像仓库地址",
                "enum": [
                   "reg.local:5000",
                   "registry.meizu.com"
                ],
                "default": "{{global.registry}}"
             }
          }
     },
     "mfs": {
       "title": "mfs",
          "type": "object",
          "properties": {
             "inuse": {
                "type": "boolean",
                "description": "是否需要mfs",
                "default": "{{mfs.inuse}}"
             },
             "mfs_name": {
```

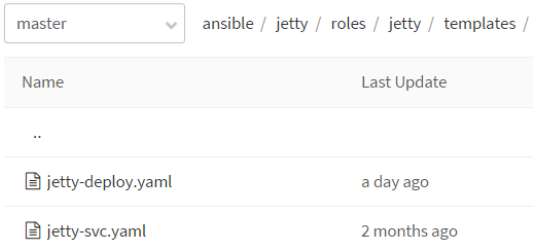
```
"jetty":{
            "title": "Jetty",
            "type": "object",
            "properties": {
                    "version": {
                            "type": "string",
                            "description": "镜像版本",
                            "minLength": 3,
                            "default": "{{jetty.version}}"
                    },
                    "image": {
                            "type": "string",
                            "description": "镜像名字",
                            "minLength": 1,
                            "default": "{{jetty.image}}"
                    },
                    "jvm_xmx": {
                            "type": "string",
                            "description": "JVM最大分配内存",
                            "minLength": 1,
                            "default": "{{jetty.jvm_xms}}"
                    },
                    "jvm_xms": {
                            "type": "string",
                            "description": "JVM最小分配内存",
                            "minLength": 1,
                            "default": "{{jetty.jvm_xmx}}"
                    },
                    "limit_memory": {
                            "type": "string",
                            "description": "容器内存限制,单位Gi，Mi",
                            "minLength": 2,
                            "default": "{{jetty.limit_memory}}"
                    },
                    "limit_cpu": {
                            "type": "string",
                            "description": "容器cpu限制,单位(核)最小0.1",
                            "minLength": 1,
                            "default": "{{jetty.limit_cpu}}"
                    },
```

# 业务部署-Ansible

- 生成资源描述文件Yaml
  - 模板库
  - 角色库
  - Resource template+json=yaml
- 创建K8s资源
  - 直接调用kubectl，比API功能强大
  - Ansible roles
  - Namespace,Deployment,Service,Ingress,Configmap，HPA
- 创建其它资源
  - Kibana Search，Dashboard
  - Grafana面板
  - 调用脚本

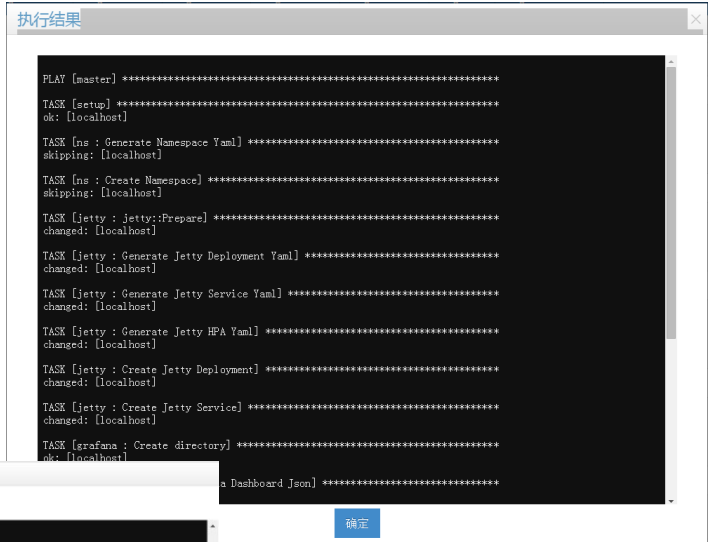


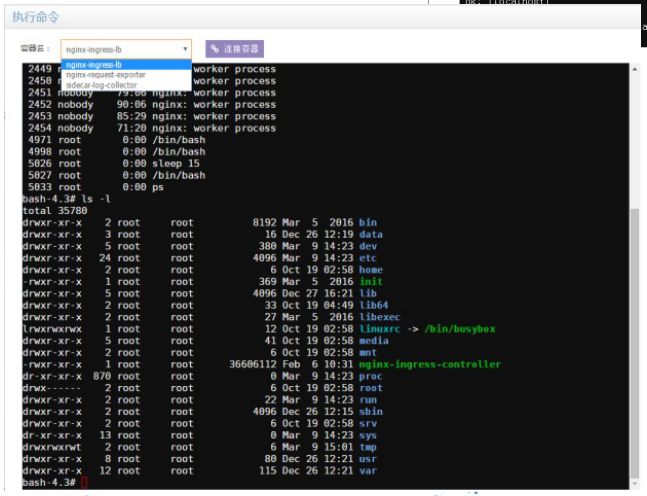



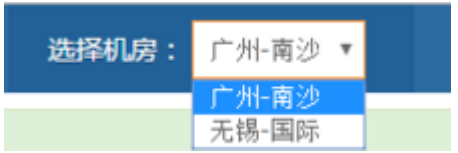
# 业务部署-易用

- 方便使用
  - 查看部署进度
  - schema的模板，保存常用配置，输入更少
  - 查看/修改历史发布
- 安全感
  - 提供熟悉的console界面



# 业务部署-多集群管理




1. 实现
   - 每套集群自签证书不同
   - 对每套Token和CA进行分组
2. 访问方式
   - 管理界面，通过https访问
   - Javascript web console,通过Nginx反向代理websocket访问
   - 模板系统，ansible通过指定不同的k8s context访问

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /etc/kubernetes/ssl/ca.pem
    server: https://kubernetes-ns.meizu.com:443
  name: ns-cluster
- cluster:
    insecure-skip-tls-verify: true
    server: https://kubernetes-bj.meizu.com:443
  name: bj-cluster
- cluster:
    insecure-skip-tls-verify: true
    server: https://kubernetes-wx.meizu.com:443
  name: wx-cluster
contexts:
- context:
    cluster: ns-cluster
    user: ns-cluster
  name: ns-cluster
- context:
    cluster: bj-cluster
    user: bj-cluster
  name: bj-cluster
- context:
    cluster: wx-cluster
    user: wx-cluster
```

# 镜像-alpine


1. 选择Alpine
   - 小且够用
   - 加入glibc支持，保证兼容性
   - 搭建本地私有仓库

```
registry.meizu.com/codis/codis-server:rdb3.4 d3077932ef66  3 months ago 50.22 MB
registry.meizu.com/common/ingress-alpine:1.11.6.18 2697f0a9a16c 13 days ago 96.67 MB
```

2. 业务需求
   - 可能需要多个进程配合
   - 允许进程一次性执行，结束后检查返回值，并执行动作，如退出重试

```
PID   USER     TIME   COMMAND
    1 root       0:00 s6-svscan -t0 /var/run/s6/services
   23 root       0:00 s6-supervise s6-fdholderd
  187 root       0:00 s6-supervise add-to-group
  188 root       0:00 s6-supervise codis
  190 root       0:00 s6-supervise create-group
```

3. 选择s6作为进程/服务管理器
   - run，启动脚本
   - finish，run退出后执行，根据返回值选择
   - 根据add_to_group脚本返回值，1->不退出，0->退出服务

```
#run
/codis/codis-start add_to_group
```

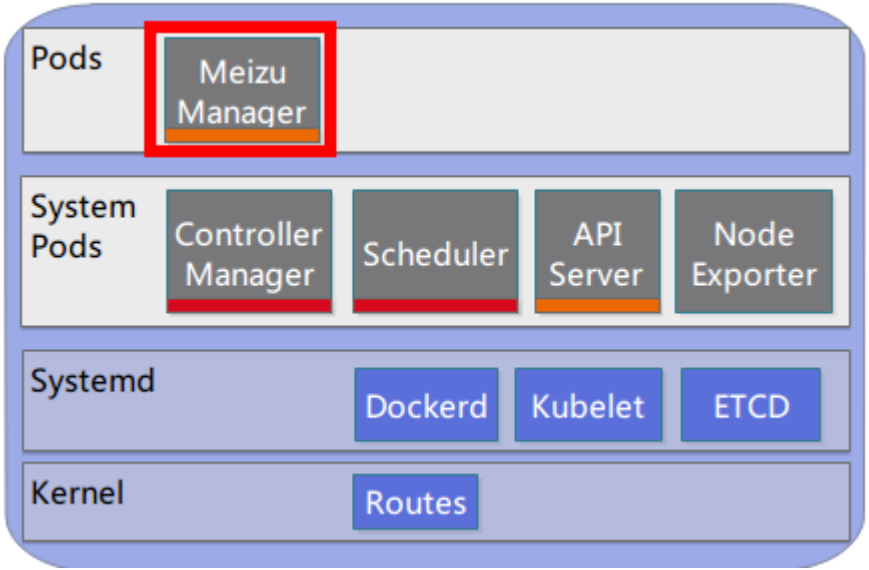
```
#finish
if [ 0 -eq $1 ]; then
   s6-svc -d /var/run/s6/services/add-to-group
fi
exit 0
```

# 镜像-多机房缓存

1. 镜像缓存
   - 镜像拉取加速
   - 机房本地缓存
   - 简单，小成本
2. 方案
   - 基于squid
   - Pod化,自动运维

# 业务部署-总结

1. 用户接受度
   - 易用
2. 维护
   - 简单，易扩展
   - 问题易排查
3. 多集群管理
   - 简单，够用
   - 基于k8s context
4. 镜像
   - 小,快,够用



# 高效运维社区

GreatOPS Community

## 会议

- 3月18日 DevOpsDays 北京
- 8月18日 DevOpsDays 上海
- 全年 DevOps China 巡回沙龙
- 4月21日 GOPS深圳
- 11月17日 DevOps金融上海

## 培训

- EXIN  DevOps Master 认证培训
- DevOps 企业内训
- DevOps 公开课
- 互联网运维培训

## 咨询

- 企业DevOps 实践咨询
- 企业运维咨询

商务经理：刘静女士
电话 / 微信：13021082989
邮箱：liujing@greatops.com

# Thanks

高效运维社区

开放运维联盟

荣誉出品

想第一时间看到
高效运维社区公众号
的好文章吗？


请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好