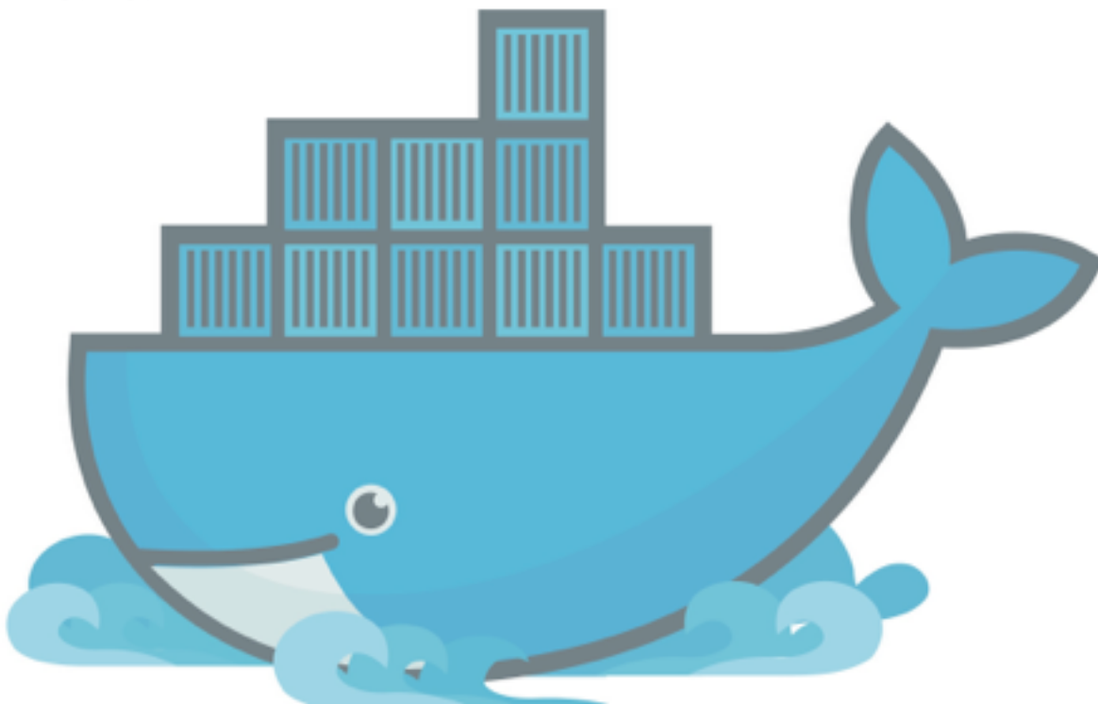
基于Docker的云处理服务架构实践

叶靖 (yejingx) @ 又拍云 杭州

2016.7.9 SF D-Day Dalian



upyun

云分发

云处理

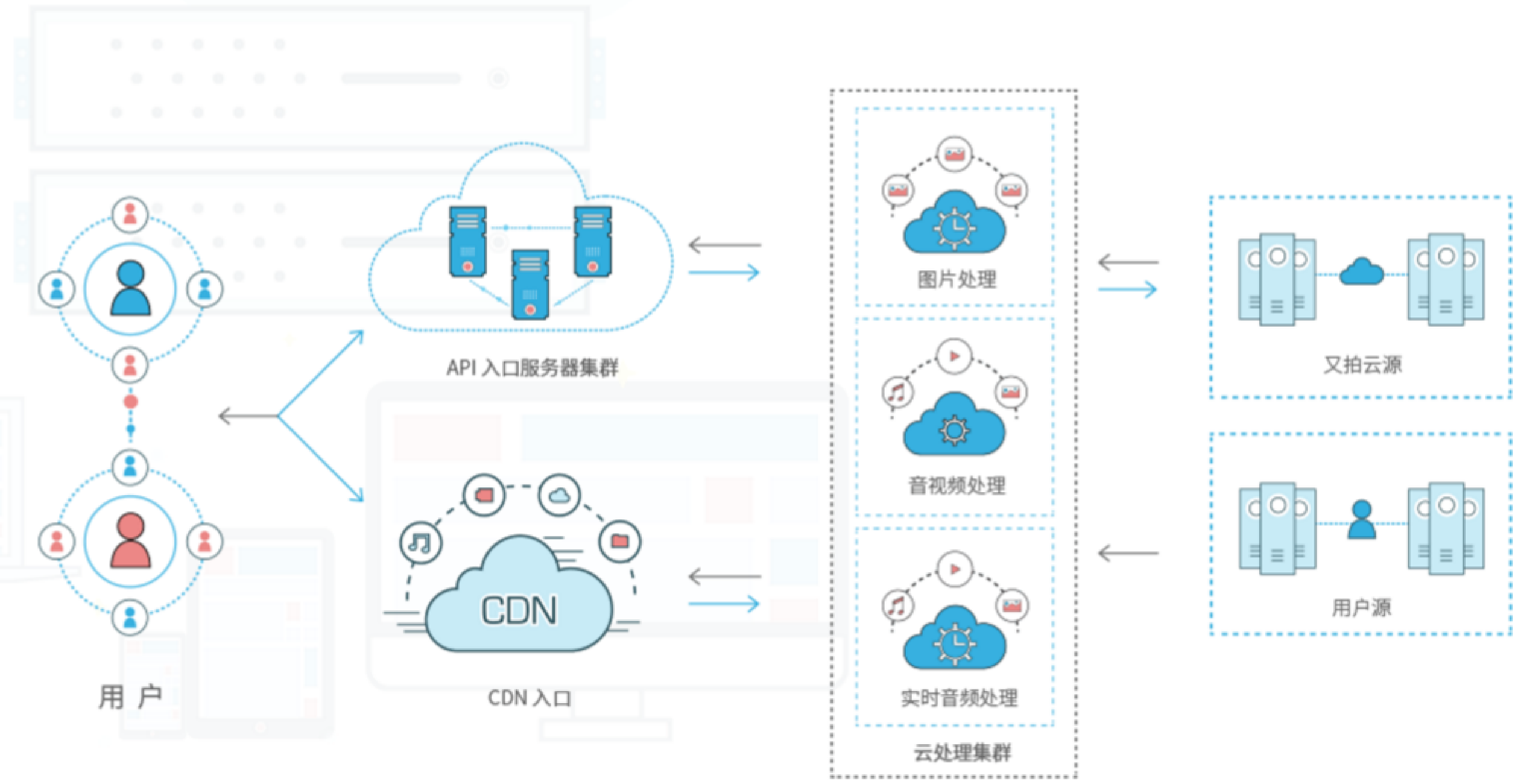
云存储

云安全

视频云

云处理在又拍

upyun



裸机部署

upyun

部署、升级复杂

环境乱

资源抢占

管理分散

资源利用率低

Docker 的优势

upyun

资源、环境隔离

部署、升级方便

扩容方便

统一管理

社区活跃

性能好

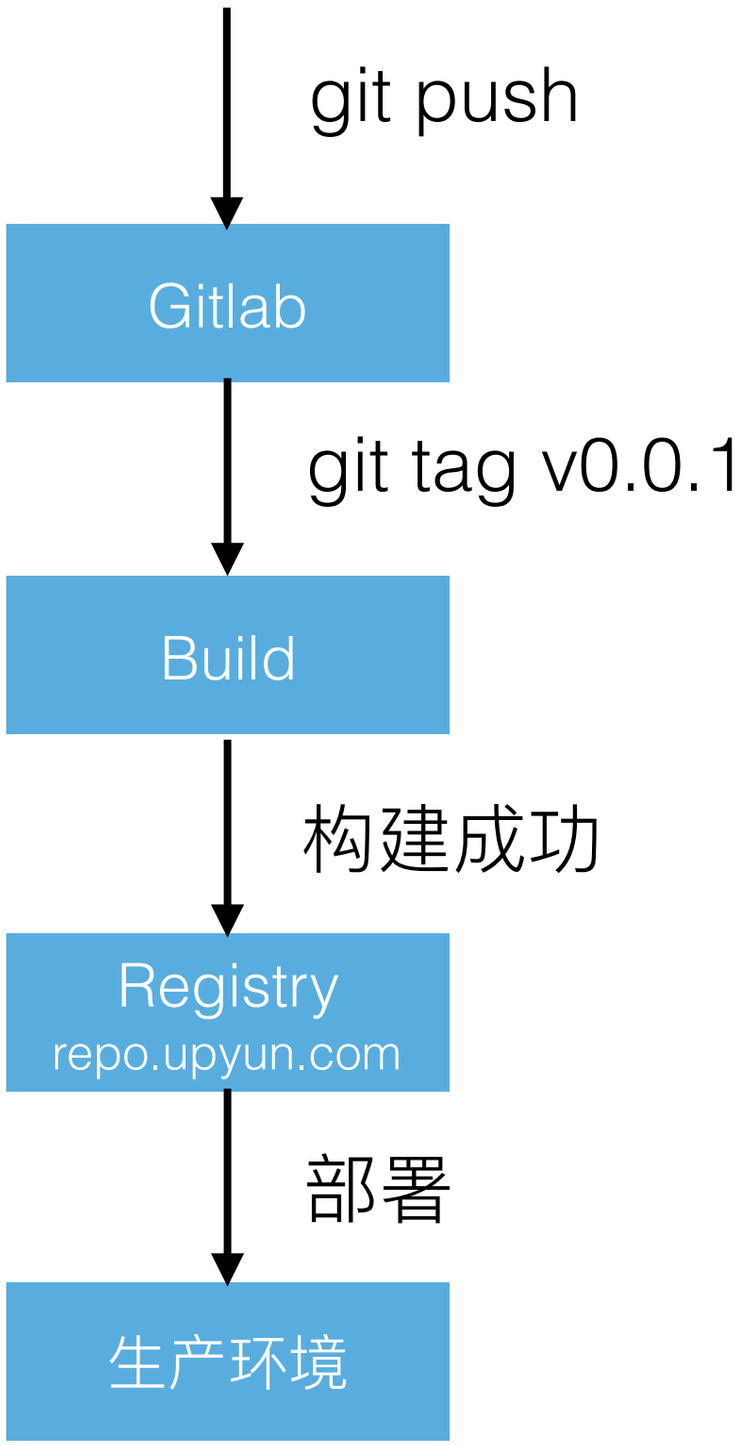
upyun

+

+

基于 Docker 的 CI

upyun



基于 Docker 的 CI

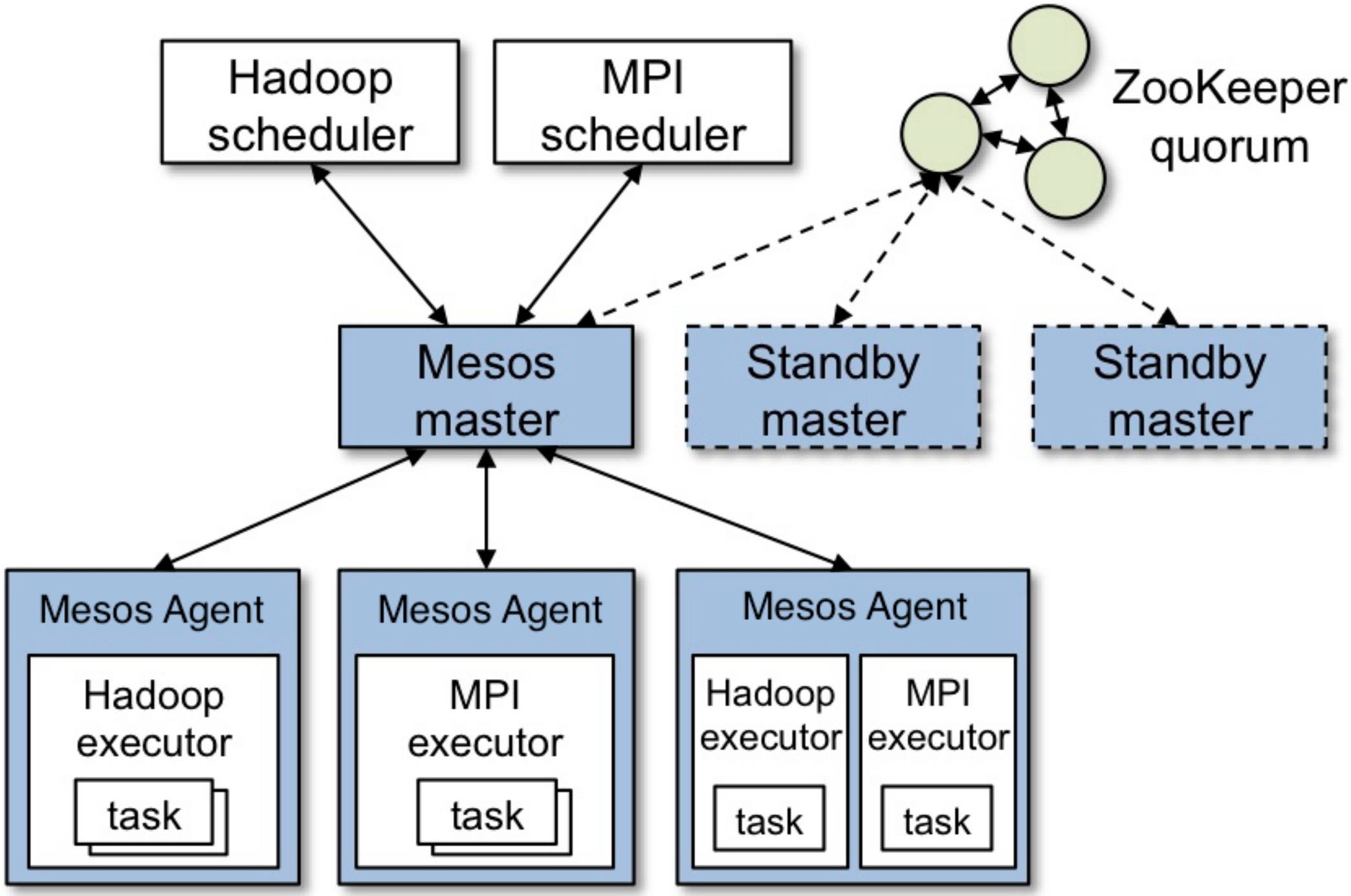
upyun

#.gitlab-ci.yaml

```
build:
    stage: build
    script:
        - img=repo.upyun.com:5043/alertman:$CI_BUILD_TAG
        - docker build -t $img .
        - docker push $img
        - docker rmi $img
    tags:
        - on-line-docker-builder
    only:
        - tags@consumers/alertman
```

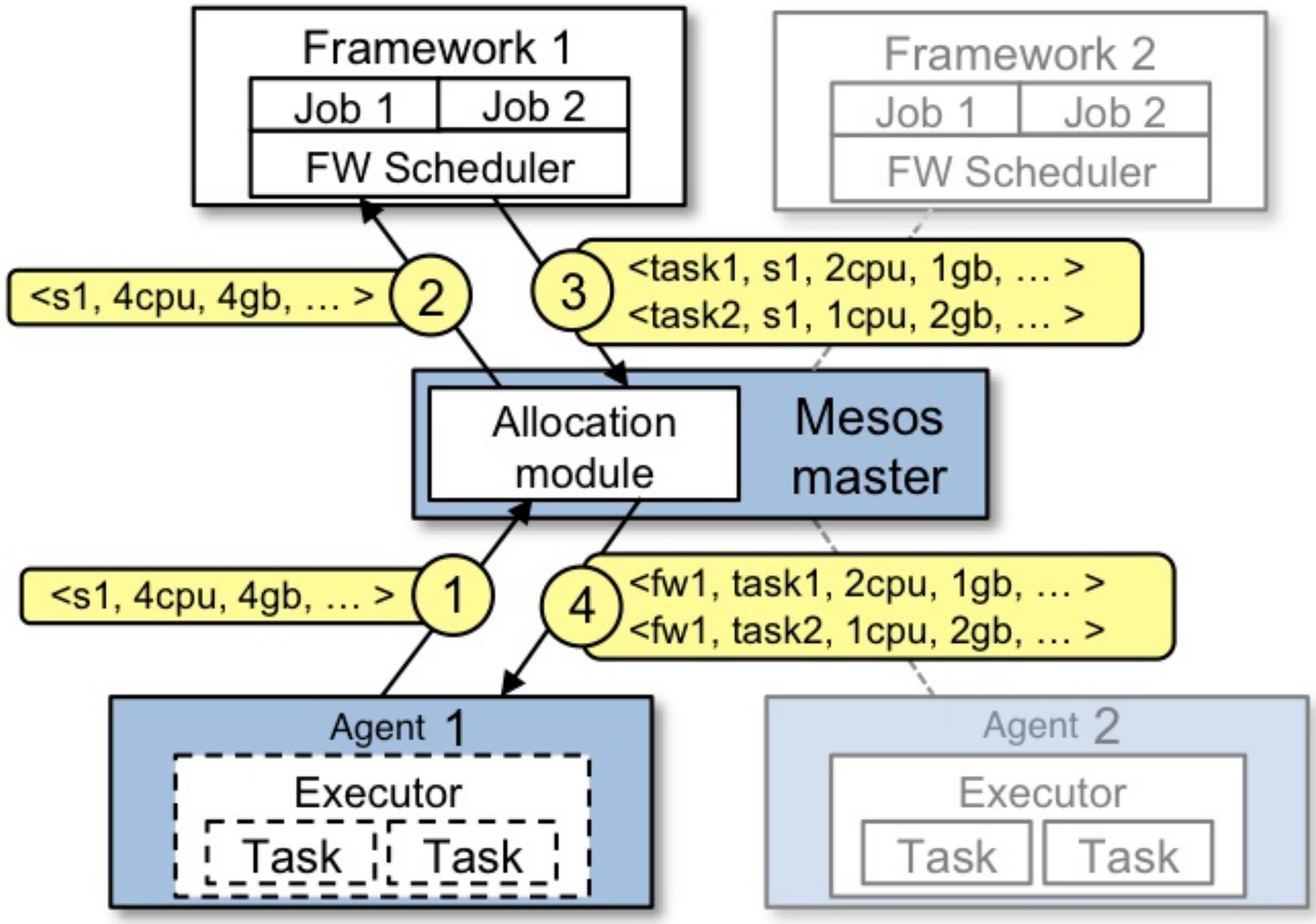
Mesos

upyun


Hadoop scheduler
MPI scheduler
ZooKeeper quorum
Mesos master
Standby master
Standby master
Mesos Agent
Hadoop executor
task
Mesos Agent
MPI executor
task
Mesos Agent
Hadoop executor
task
MPI executor
task

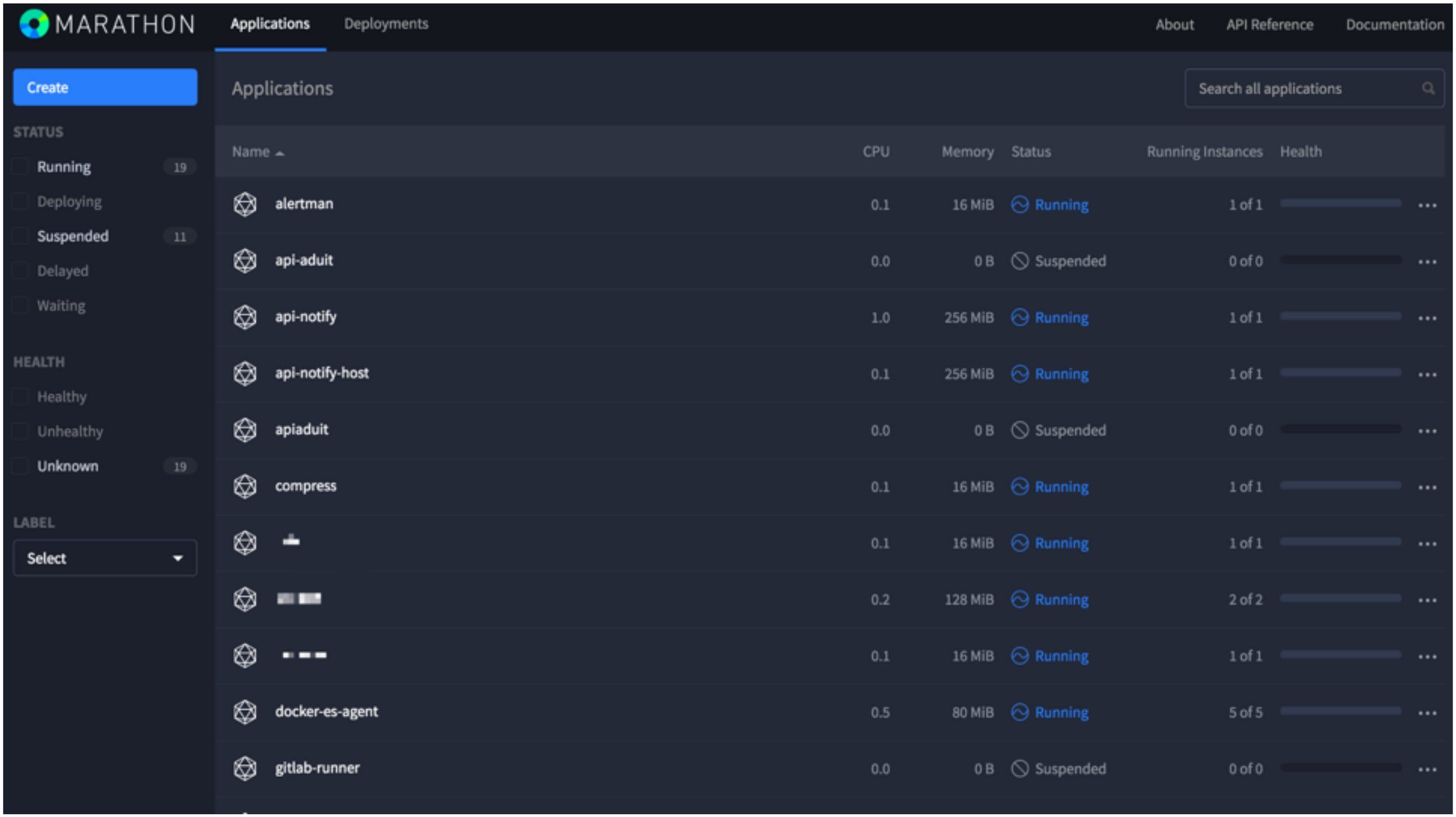
Mesos

upyun


Framework 1
Job 1
Job 2
FW Scheduler
Framework 2
Job 1
Job 2
FW Scheduler
<s1, 4cpu, 4gb, … >
2
3
<task1, s1, 2cpu, 1gb, … >
<task2, s1, 1cpu, 2gb, … >
Allocation module
Mesos master
<s1, 4cpu, 4gb, … >
1
4
<fw1, task1, 2cpu, 1gb, … >
<fw1, task2, 1cpu, 2gb, … >
Agent 1
Executor
Task
Task
Agent 2
Executor
Task
Task

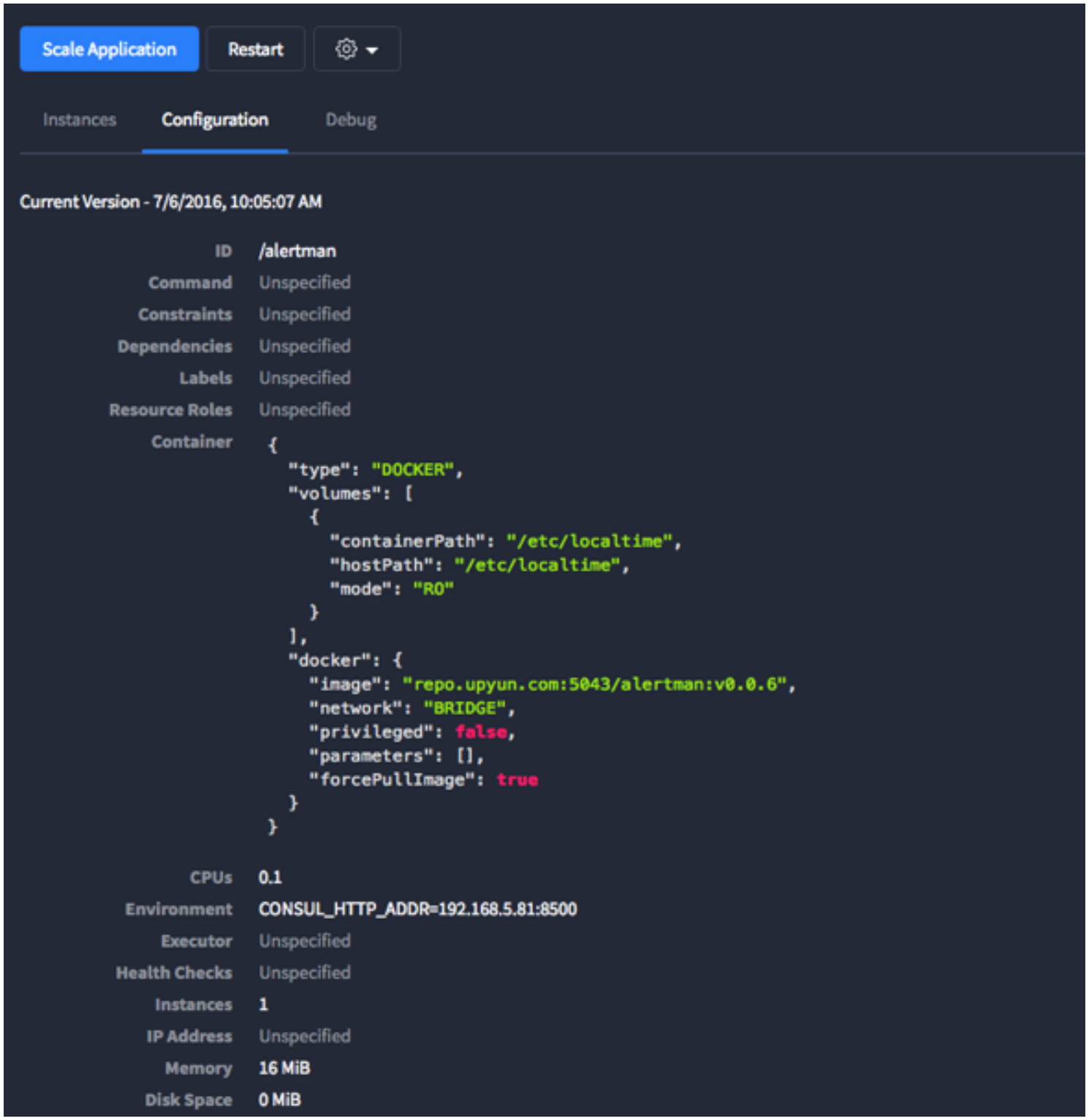
Marathon

upyun



Marathon

upyun



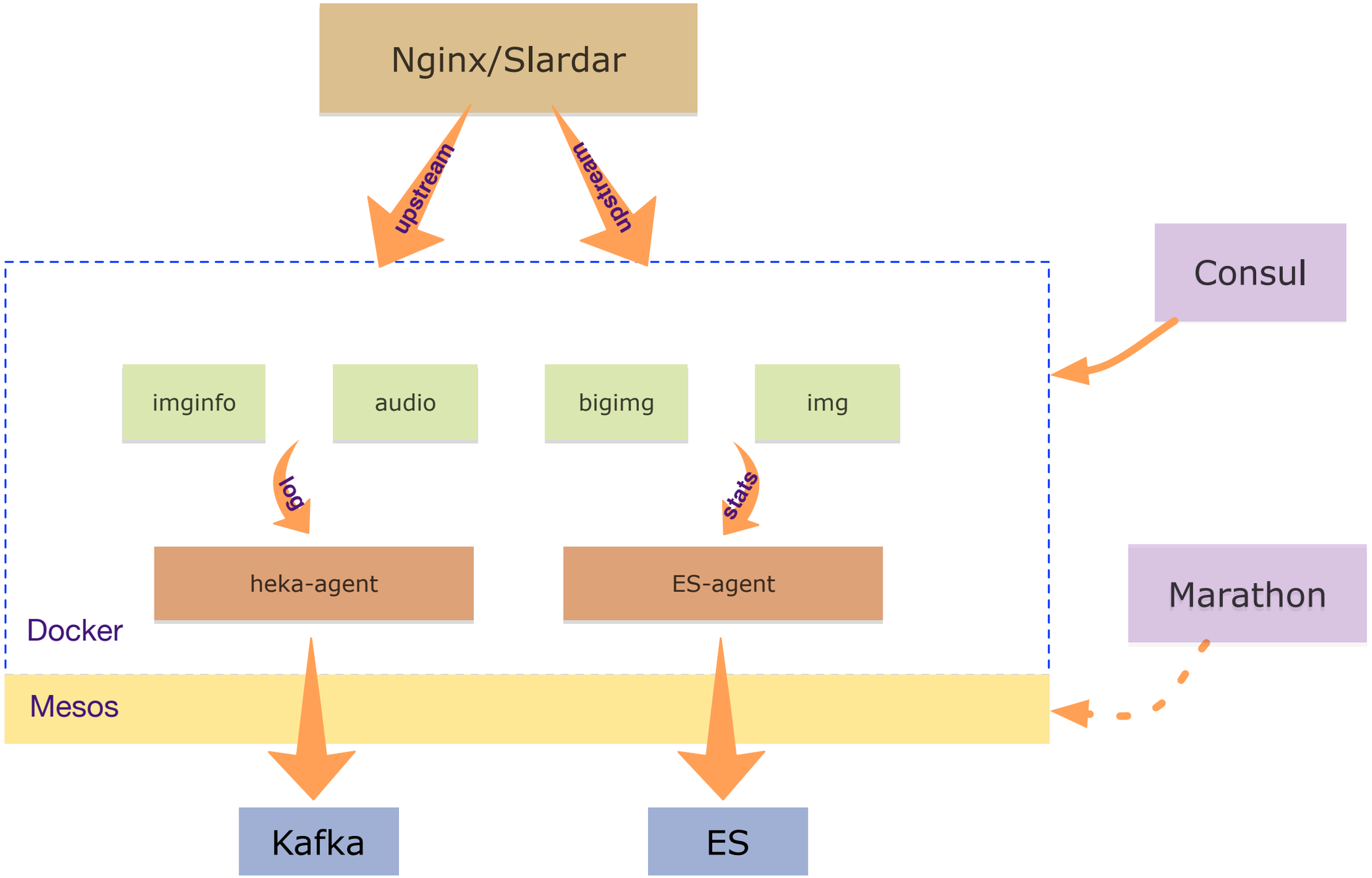
光有这些是 **不够的！**

我们还需要

upyun

- 服务路由
- 配置管理
- 日志收集
- 服务状态
- 监控报警
- ……

upyun


Nginx/Slardar
upstream
upstream
Consul
imginfo
audio
bigimg
img
log
stats
heka-agent
ES-agent
Marathon
Docker
Mesos
Kafka
ES

动态服务路由

upyun

HAProxy ?

动态服务路由

upyun

上游列表如何改变?

新服务如何添加?

动态服务路由 - Slardar

upyun

NGINX + Lua

lua-resty-logger-socket

lua-resty-httpipe

lua-resty-checkups

lua-resty-limit-traffic

lua-resty-httproxy

动态服务路由 - Slardar

以 Host 区分服务:

```
$ curl -T cat.jpg 192.168.1.155:3130 \
    -H "Host: imageinfo"
$ curl -T cat.jpg 192.168.1.155:3130 \
    -H "Host: imgprocess"
```

动态服务路由 - Slardar

通过 HTTP 接口动态更新 upstream 列表:

```
curl -d \
  '{“servers”:[ \
   {"host":"10.0.5.108", "port": 4001}, \       ——> node docker 1
   {“host":"10.0.5.109", "port": 4001}], \      ——> node docker 2
   ”keepalive": 20}’\
  127.0.0.1:1995/upstream/node-dev.upyun.com
```

动态服务路由 - Slardar

upyun


```
"cls:node-dev.upyun.com": [
    [
        {
            "server": "node-dev.upyun.com:10.0.5.108:4001",
            "msg": null,
            "status": "ok",
            "lastmodified": "2016-07-05 16:23:48",
            "fail_num": 0
        },
        {
            "server": "node-dev.upyun.com:10.0.5.109:4001",
            "msg": "connection refused",
            "status": "err",
            "lastmodified": "2016-07-06 14:50:22",
            "fail_num": 1
        }
    ]
],
```

Consul 配置管理

upyun

Service Discovery

Failure Detection

Multi Datacenter

Key/Value Storage

Consul 配置管理

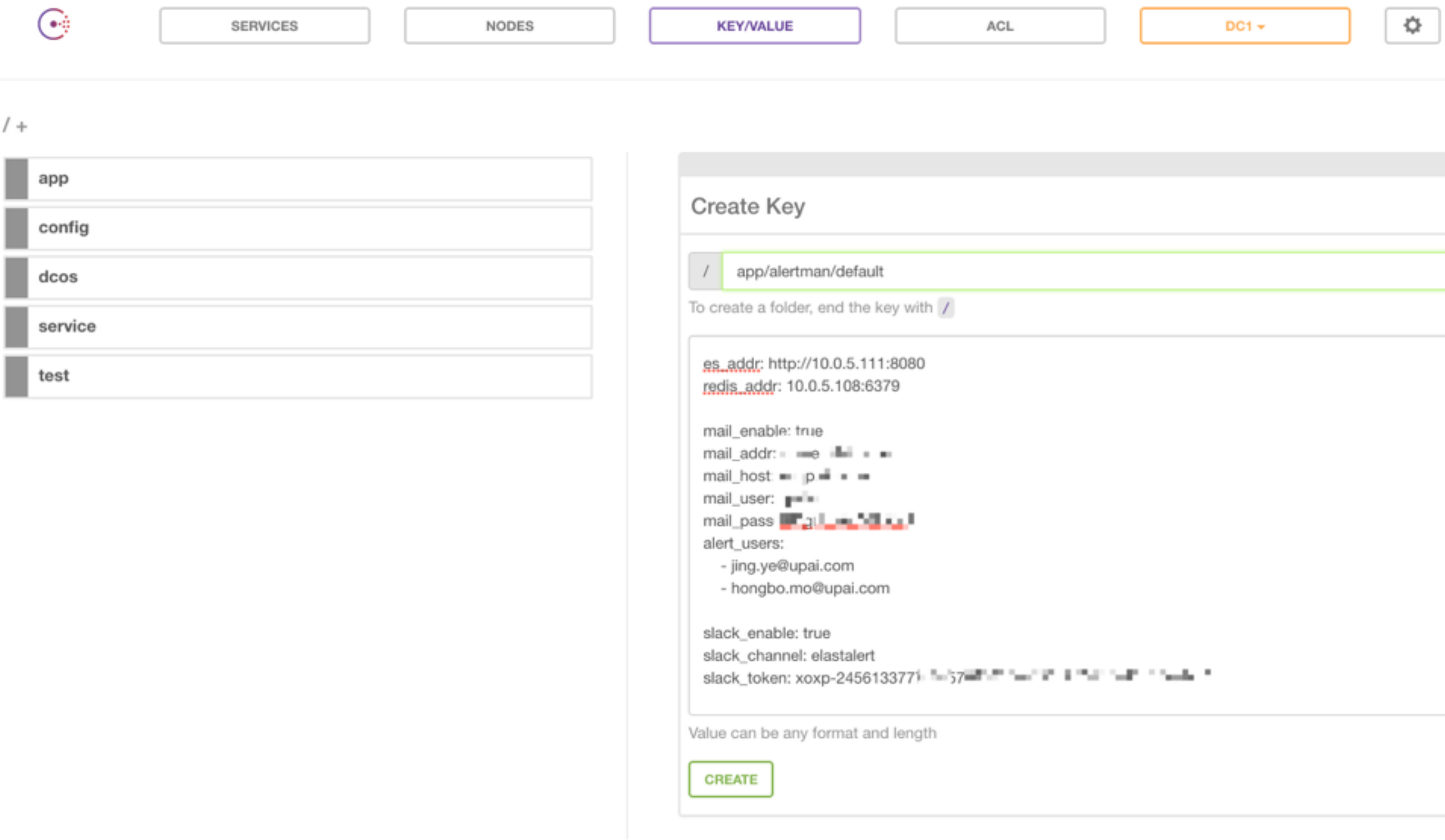
upyun

Terminal

```
admin@hashicorp: curl -X PUT -d 'bar' http://localhost:8500/v1/kv/foo
true
admin@hashicorp: curl http://localhost:8500/v1/kv/foo
[
    {
        "CreateIndex": 100,
        "ModifyIndex": 200,
        "Key": "foo",
        "Flags": 0,
        "Value": "YmFy"
    }
]
admin@hashicorp: echo "YmFy" | base64 --decode
bar
admin@hashicorp:
```

Consul 配置管理

upyun



日志收集

upyun

日志量大、频率高

分布在多台机器

经常迁移

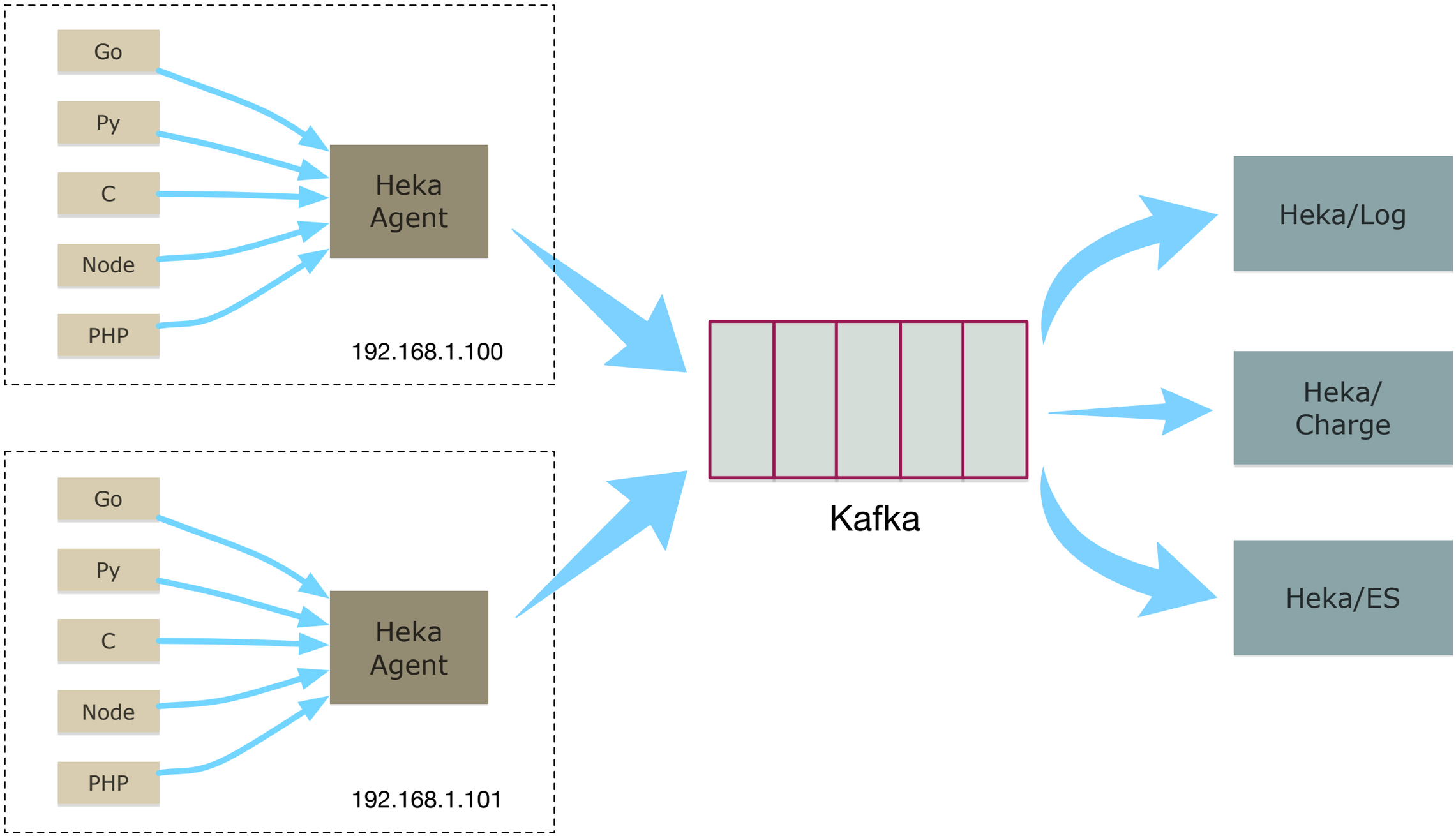
按服务分割

多种用途

日志收集

upyun

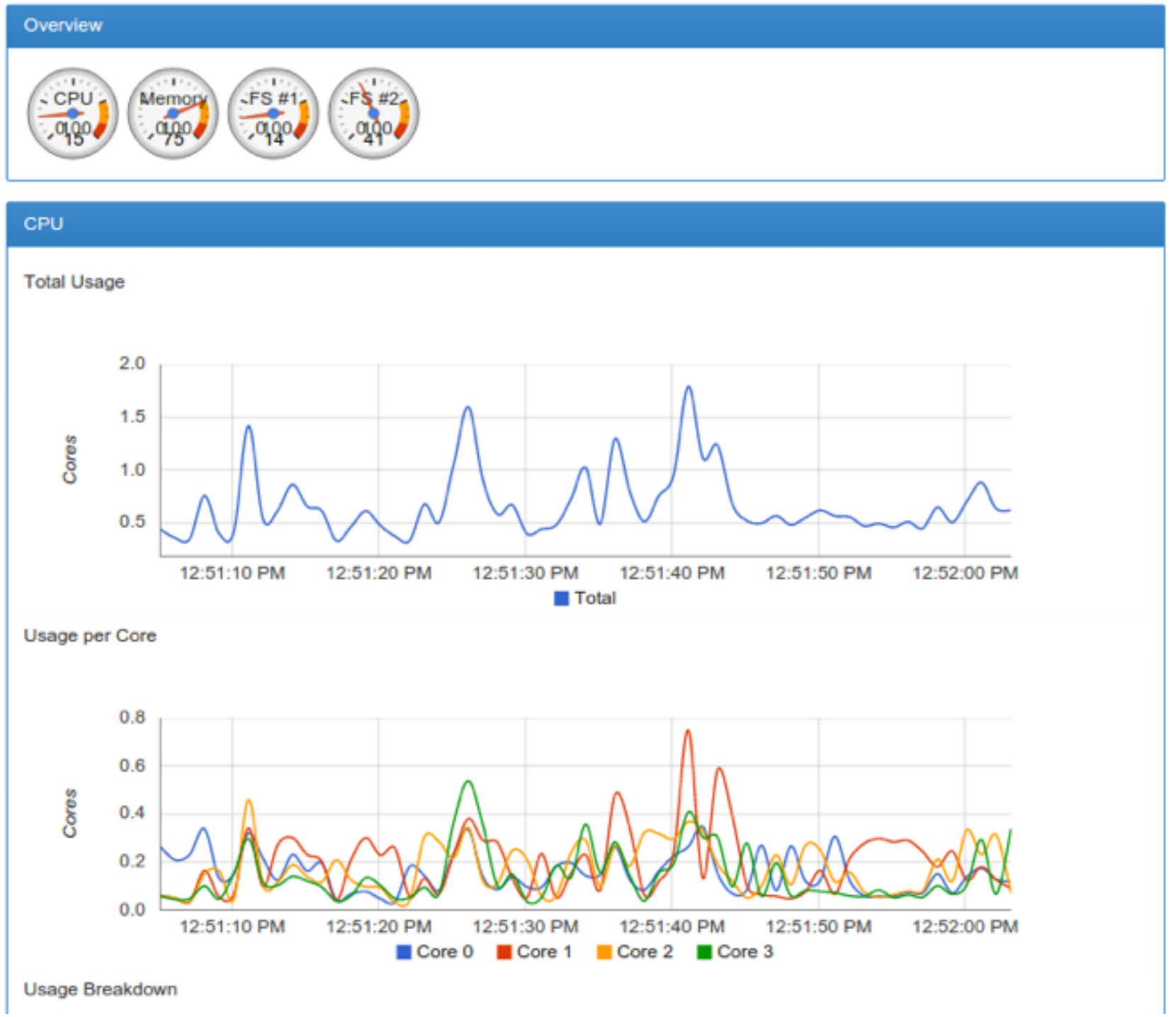
Heka-Agent + Heka Plugin



服务状态

upyun

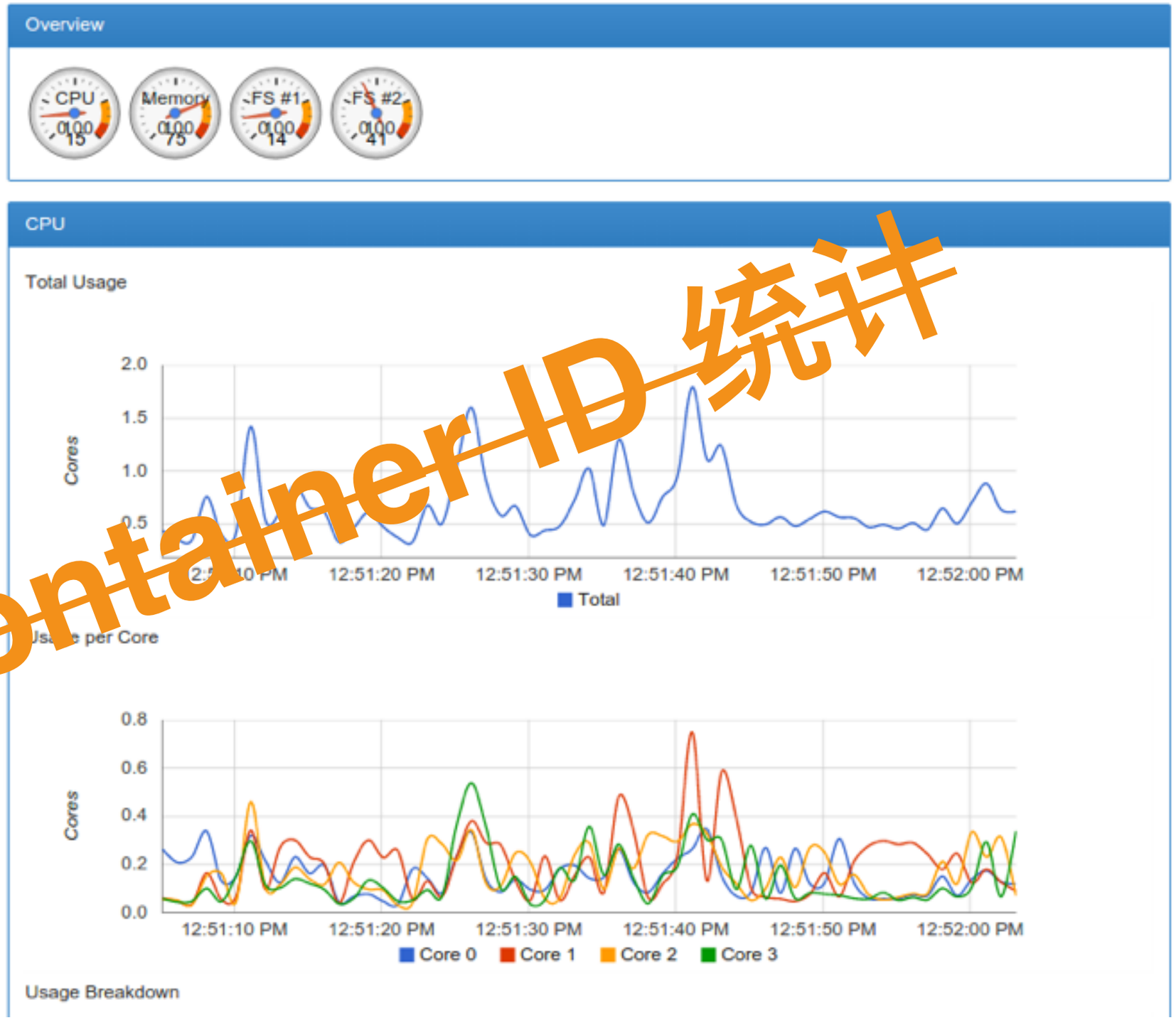
cAdvisor



服务状态

upyun

cAdvisor

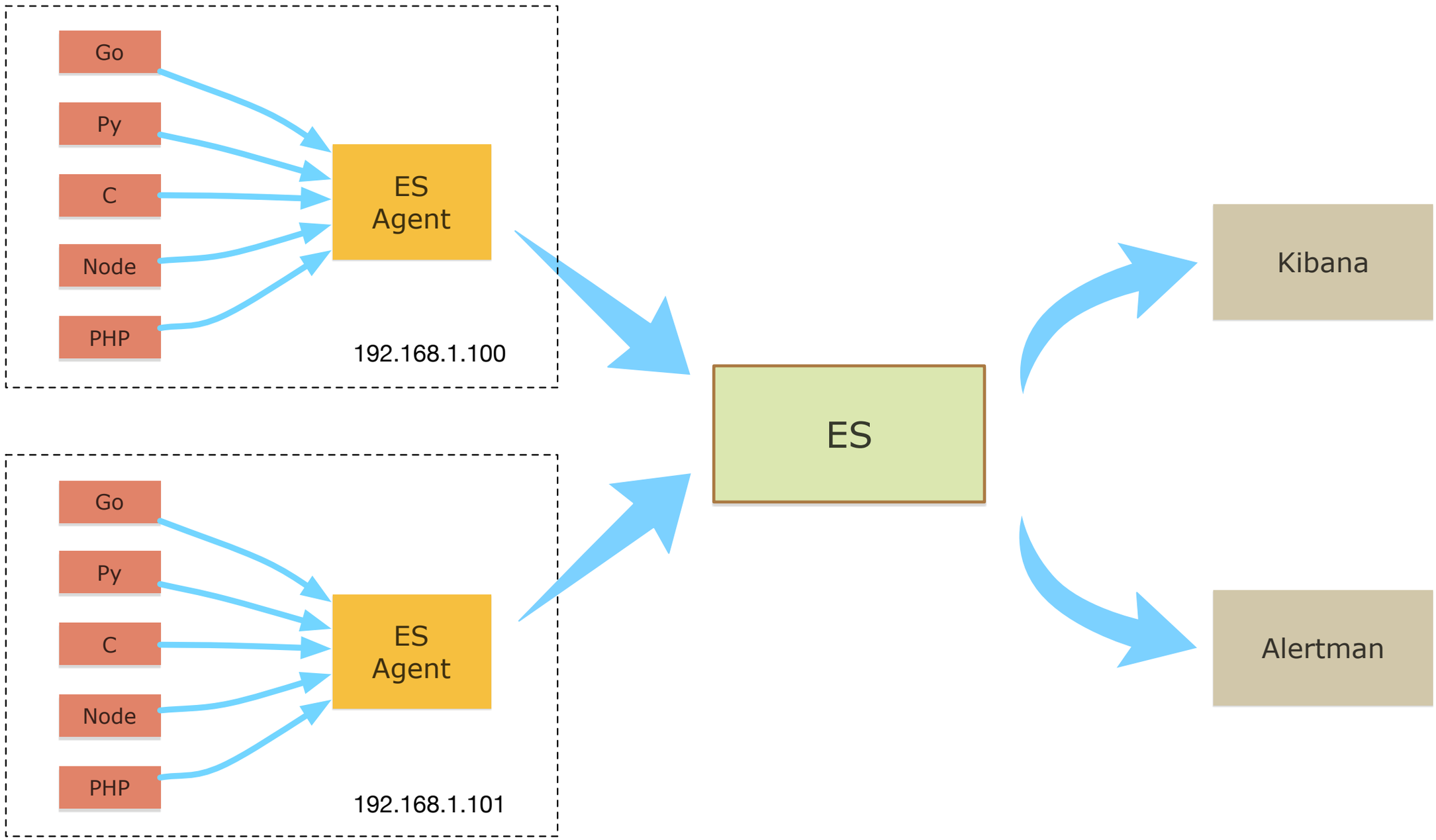


~~按container ID 统计~~

服务状态

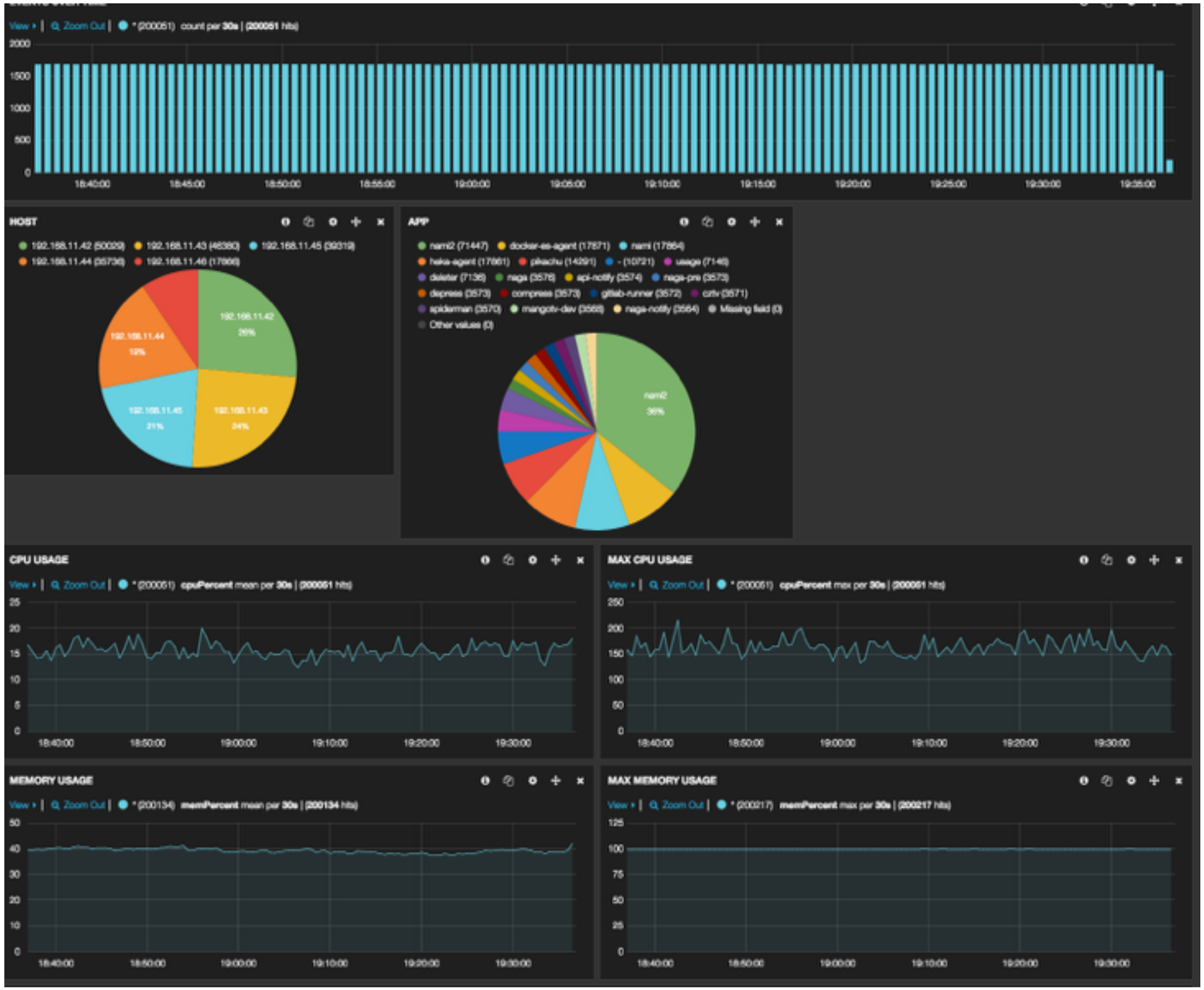
upyun

docker-es-agent



服务状态

upyun


HOST
APP
CPU USAGE
MAX CPU USAGE
MEMORY USAGE
MAX MEMORY USAGE

监控报警

upyun

Nagios®

VS

Alertman

ZABBIX

监控报警

upyun

- 不需要大而全的监控
- 异常点分析
- 自动故障恢复
- 业务相关性大

监控报警 - 自己造轮子

upyun

Nginx 日志

ES 数据

队列

……

数据分析

Alertman

监控报警 - 实例

upyun

```
2016-06-02 14:09:27,461 - status - DEBUG - got hits of status 504: [(u'192.168.13.245:6501', 438), (u'192.168.13.243:6501', 24), (u'192.168.13.34:6501', 23), (u'192.168.13
.242:6501', 15), (u'192.168.13.36:6501', 15), (u'192.168.13.105:6501', 14), (u'192.168.11.19:6501', 14), (u'192.168.11.18:6501', 14), (u'192.168.11.17:6501', 14), (u'192.1
.13.237:6501', 12), (u'192.168.11.22:6501', 12), (u'192.168.11.21:6501', 12), (u'192.168.13.107:6501', 12), (u'192.168.13.106:6501', 12), (u'192.168.11.27:6501', 11), (u'1
.168.5.205:6501', 7), (u'192.168.5.208:6501', 7), (u'192.168.5.200:6501', 6), (u'192.168.5.209:6501', 5), (u'192.168.5.201:6501', 3), (u'192.168.5.208:6501 : 192.168.13.24
2016-06-02 14:09:27,464 - status - DEBUG - stddev_from_average mean: 23.0526315789, stddev: 69.348297342
2016-06-02 14:09:27,465 - status - INFO - anomalous nodes of status 504: [(u'192.168.13.245:6501', 438)]
2016-06-02 14:09:27,467 - status - INFO - try removing [(u'192.168.13.245:6501', 438)] from slardar_test
2016-06-02 14:09:42,386 - usage - INFO - nami usage is 0.00
2016-06-02 14:09:42,433 - usage - INFO - nami2 usage is 45.41
2016-06-02 14:09:42,616 - status - DEBUG - 502 count is 95(<100), sleep 15s and try later
2016-06-02 14:09:42,749 - status - DEBUG - 504 count is 240(>80), checking hists ...
2016-06-02 14:09:48,638 - status - DEBUG - got hits of status 504: [(u'192.168.13.245:6501', 451), (u'192.168.13.243:6501', 25), (u'192.168.13.244:6501', 22), (u'192.168.1
.20:6501', 15), (u'192.168.13.105:6501', 15), (u'192.168.13.242:6501', 15), (u'192.168.11.16:6501', 15), (u'192.168.11.25:6501', 14), (u'192.168.11.19:6501', 14), (u'192.1
.13.101:6501', 13), (u'192.168.13.106:6501', 13), (u'192.168.13.237:6501', 12), (u'192.168.11.27:6501', 12), (u'192.168.13.107:6501', 12), (u'192.168.11.30:6501', 12), (u'
2.168.11.28:6501', 8), (u'192.168.5.205:6501', 7), (u'192.168.5.208:6501', 6), (u'192.168.5.200:6501', 5), (u'192.168.5.201:6501', 4), (u'192.168.5.208:6501 : 192.168.13.2
2016-06-02 14:09:48,641 - status - DEBUG - stddev_from_average mean: 23.7368421053, stddev: 71.4016746758
2016-06-02 14:09:48,642 - status - INFO - anomalous nodes of status 504: [(u'192.168.13.245:6501', 451)]
2016-06-02 14:09:48,643 - status - INFO - try removing [(u'192.168.13.245:6501', 451)] from slardar_test
2016-06-02 14:09:57,589 - usage - INFO - nami usage is 0.00
2016-06-02 14:09:57,714 - usage - INFO - nami2 usage is 46.03
2016-06-02 14:10:03,796 - status - DEBUG - 502 count is 94(<100), sleep 15s and try later
2016-06-02 14:10:03,886 - status - DEBUG - 504 count is 239(>80), checking hists ...
2016-06-02 14:10:09,323 - status - DEBUG - got hits of status 504: [(u'192.168.13.245:6501', 447), (u'192.168.13.243:6501', 25), (u'192.168.13.244:6501', 22), (u'192.168.1
.20:6501', 15), (u'192.168.13.105:6501', 15), (u'192.168.13.242:6501', 15), (u'192.168.11.25:6501', 14), (u'192.168.11.19:6501', 14), (u'192.168.11.16:6501', 14), (u'192.1
.13.101:6501', 13), (u'192.168.13.106:6501', 13), (u'192.168.13.237:6501', 12), (u'192.168.11.27:6501', 12), (u'192.168.13.107:6501', 12), (u'192.168.11.30:6501', 12), (u'
2.168.11.28:6501', 8), (u'192.168.5.205:6501', 7), (u'192.168.5.208:6501', 6), (u'192.168.5.200:6501', 5), (u'192.168.5.201:6501', 4), (u'192.168.5.208:6501 : 192.168.13.2
2016-06-02 14:10:09,327 - status - DEBUG - stddev_from_average mean: 23.5263157895, stddev: 70.7675054619
2016-06-02 14:10:09,328 - status - INFO - anomalous nodes of status 504: [(u'192.168.13.245:6501', 447)]
2016-06-02 14:10:09,329 - status - INFO - try removing [(u'192.168.13.245:6501', 447)] from slardar_test
2016-06-02 14:10:12,800 - usage - INFO - nami usage is 0.00
2016-06-02 14:10:12,905 - usage - INFO - nami2 usage is 46.88
2016-06-02 14:10:24,490 - status - DEBUG - 502 count is 96(<100), sleep 15s and try later
2016-06-02 14:10:24,550 - status - DEBUG - 504 count is 240(>80), checking hists ...
2016-06-02 14:10:28,154 - usage - INFO - nami usage is 0.00
2016-06-02 14:10:28,329 - usage - INFO - nami2 usage is 47.43
```

遇到的坑

kernel: nf_conntrack: table full, dropping packet

由于 Docker 使用了 NAT 网络，在高并发流量下，追踪连接状态的表满导致内核丢包。

解决：

```
net.netfilter.nf_conntrack_max = 655350
```

遇到的坑

upyun

Mesos 里看到容器频繁重启，但 docker ps 看旧进程却依然在

"Unrecognized input header" interrupting docker client #19950

New issue

Closed **clkao** opened this issue on Feb 3 · 58 comments

clkao commented on Feb 3

I am seeing this in 1.10.0 rc1 and rc2 about 5~10% of the time.

The `docker run` client exited with "Unrecognized input header" after a few minutes, but the container was still running and you can see new output with `docker logs -f`

The `docker run` cmmand is: (simplified just to show what switches are being used)

```
docker run --entrypoint /sbin/smell-baron --name somename -h somehost --volumes-from somevc
```

Labels

priority/P0

Milestone

1.10.3

Assignees

No one assigned

18 participants

https://github.com/docker/docker/issues/19950

遇到的坑

upyun

. . . . . .

欢迎加入又拍云

hr@upai.com

Q & A