QCon[上海站]
全球软件开发大会2016

Yarn on Docker

容器技术在大数据场景下的应用

SPEAKER

TalkingData 宋净超

About me

宋净超 Jimmy Song

At TalkingData 北京

大数据工程师

专注于大数据Hadoop架构与微服务Docker实践

jingchao.song@tendcloud.com

2016年10月

内容

1. 为何使用Yarn on Docker
2. 架构
3. 镜像制作与发布
4. 容器配置与管理
5. 优化
6. 自定义网络
7. Future

Data Center存在的问题

- 主机资源利用率低
- 部署和扩展复杂
- 资源隔离无法动态调整
- 无法快速响应业务

为何使用Yarn on Docker

- 彻底隔离队列
  - 为了合理利用Hadoop yarn的资源，队列间会互相抢占计算资源，造成重要任务阻塞
  - 根据部门申请的机器数量划分Yarn集群方便财务管理
- 更细粒度的资源分配
  - 统一的资源分配
  - 每个NodeManager和容器都可以限定CPU、内存资源
  - Yarn资源划分精确到CPU核数和内存大小
- 弹性伸缩性服务
  - 每个容器中运行一个NodeManager，增减yarn资源只需增减容器个数
  - 可以指定每个NodeManager拥有的计算资源多少，按需申请资源

带来的好处

- Swarm统一集群资源调度

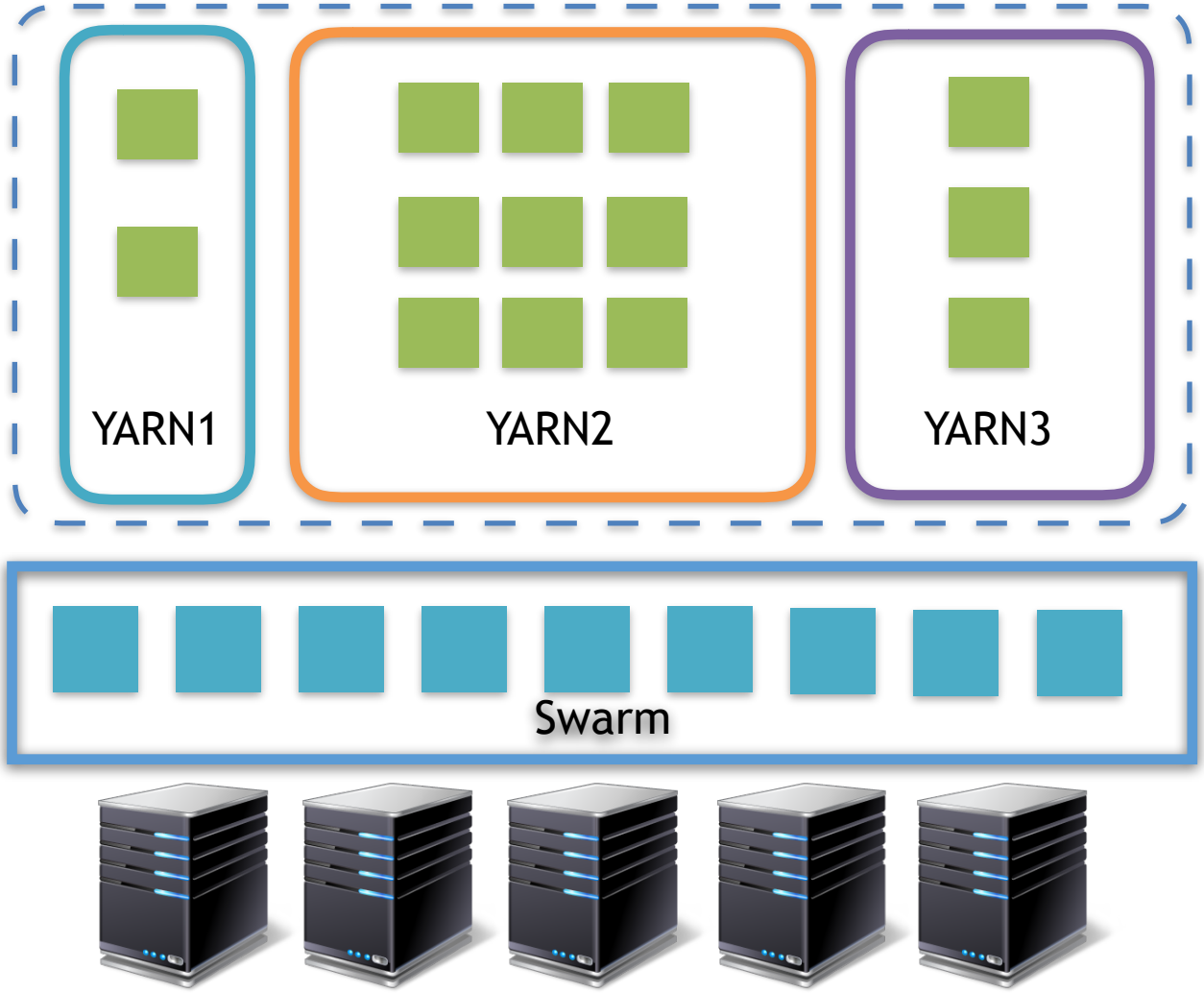
- 统一资源
- 增加Docker虚拟化层，降低运维成本

- 增加Hadoop集群资源利用率

- For datacenter：避免了静态资源隔离
- For cluster：加强集群内部资源隔离

架构

Yarn on Docker



二层调度

Swarm调度给主机的物理资源给不同的Yarn

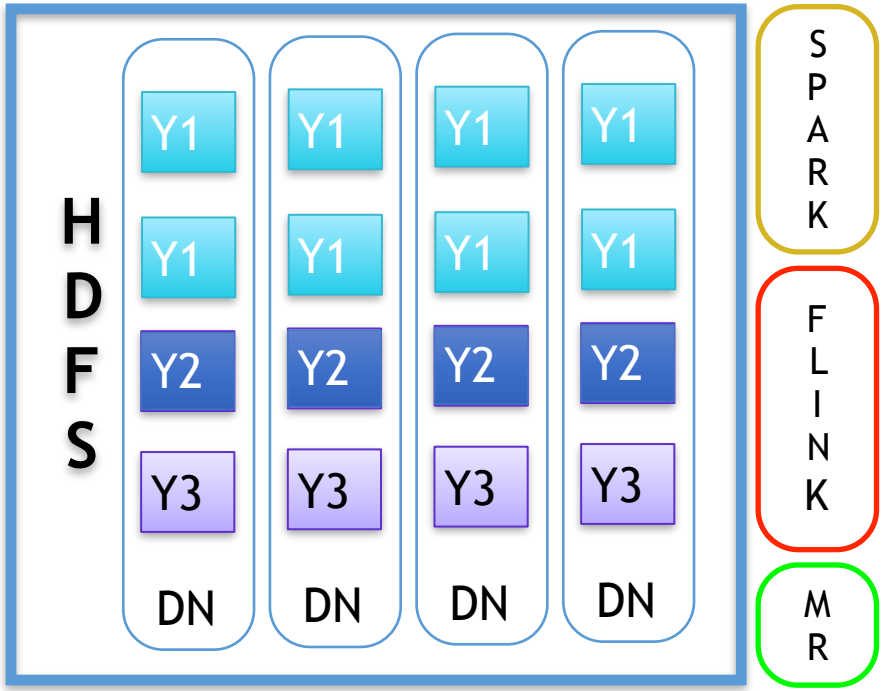
Yarn再调度App到ContainerExecutor上

Docker container

ContainerExecutor

架构

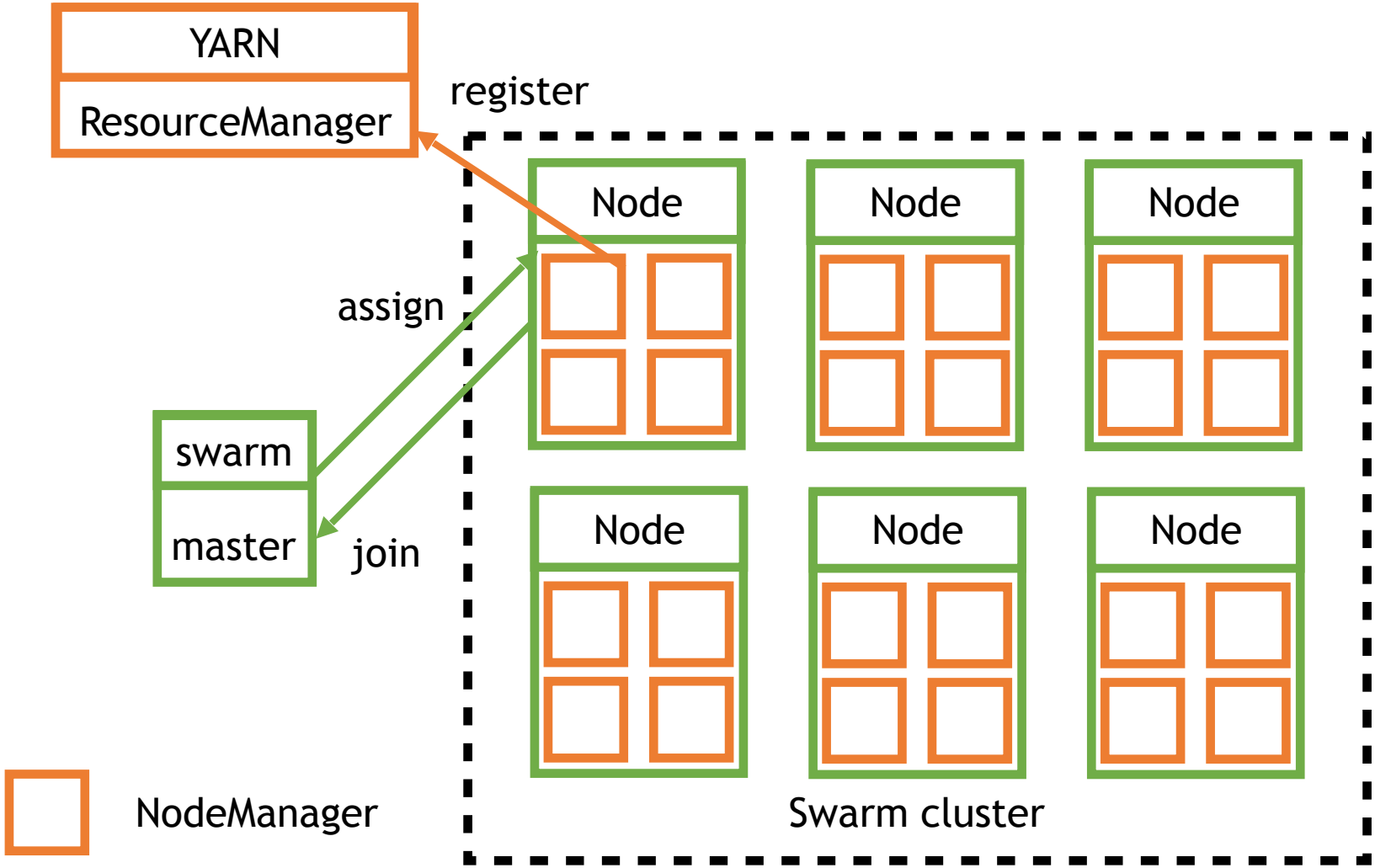
Yarn on Docker



主机规格：24核 64G
容器规格：5核 12G
NM规格：4核 10G
OS、DN预留4核 16G

Y1 Yarn1容器　Y2 Yarn2 容器　Y3 Yarn3容器

Yarn on Docker



带来的好处

1.swarm node向swarm master注册主机资源并加入到swarm cluster中

2.swarm master向cluster申请资源请求启动容器

3.swarm根据调度策略选择在某个node上

启动docker container

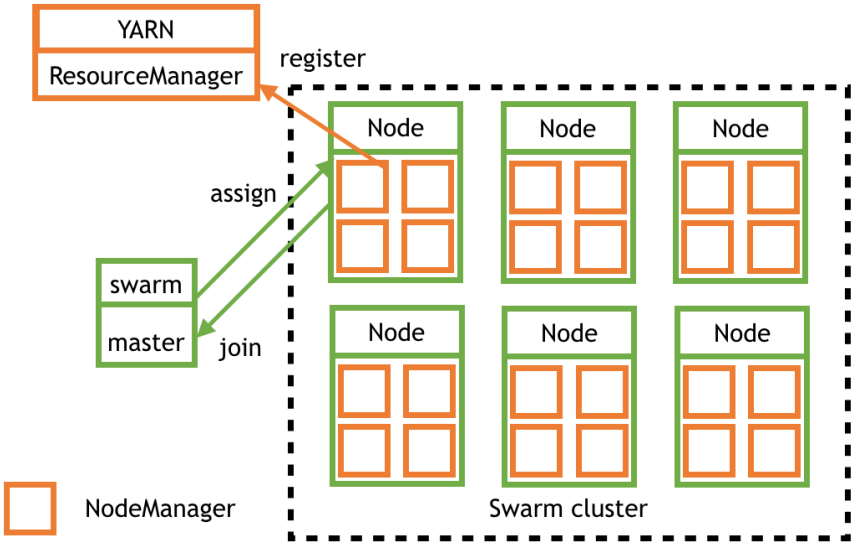
4.swarm node的docker deamon

根据容器启动参数启动相应

资源大小的NodeManager

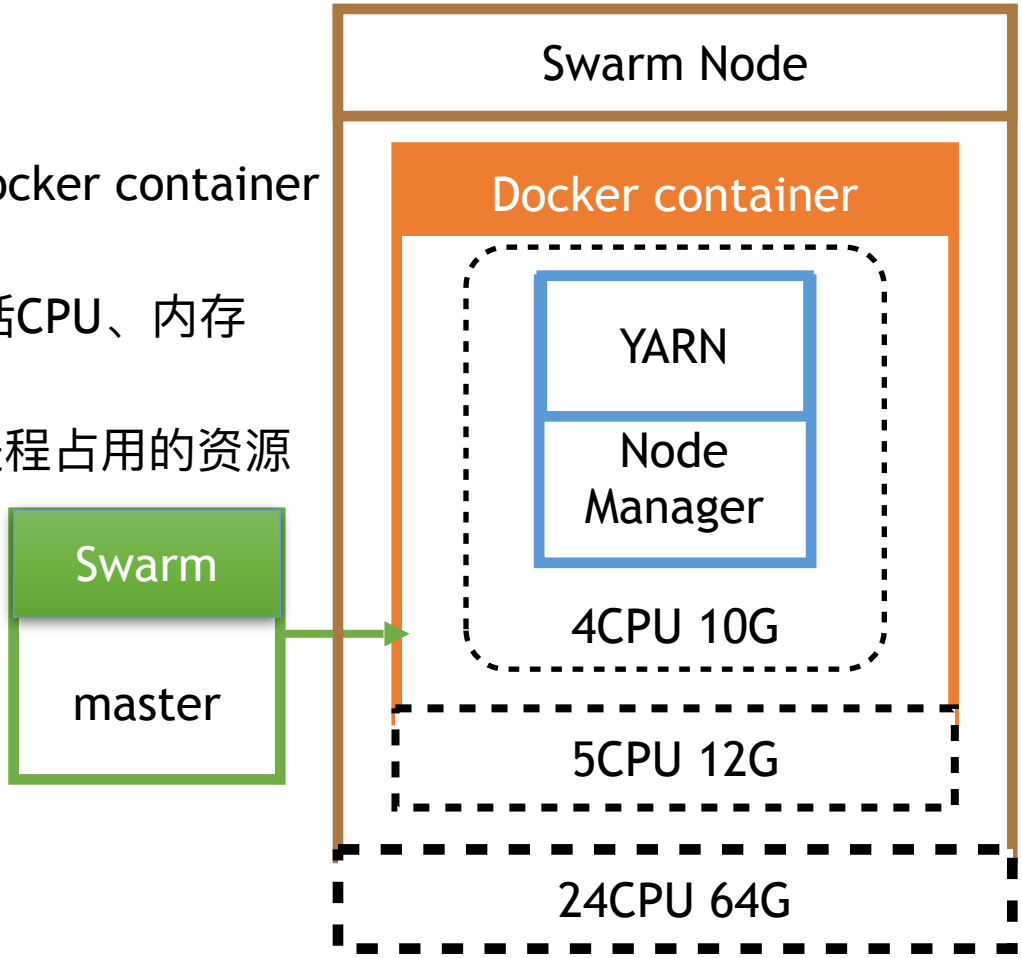
5.NodeManager自动向YARN的ResourceManager注册资源

一个NodeManager资源添加完成



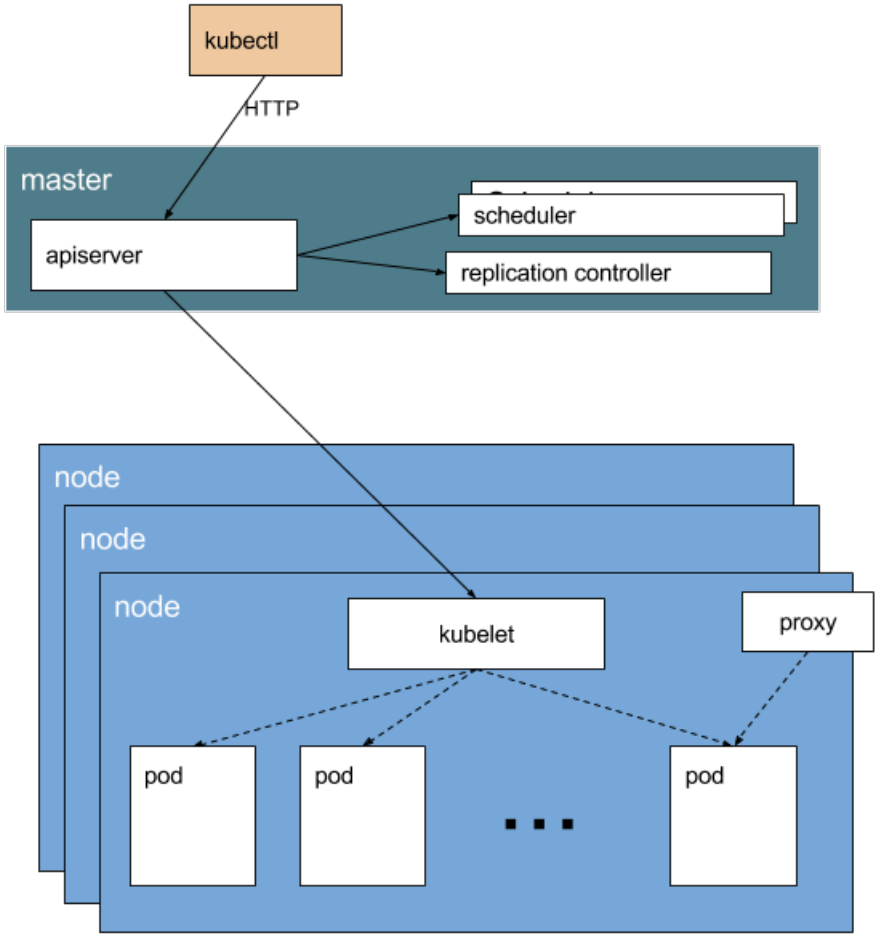
Swarm Node结构

- 一个Swarm node就是一台物理机
- 一台主机上可以起多个同类型的docker container
- 每个container的资源都有限制包括CPU、内存
- NodeManager容器需要考虑本身进程占用的资源
- 需要给主机预留资源



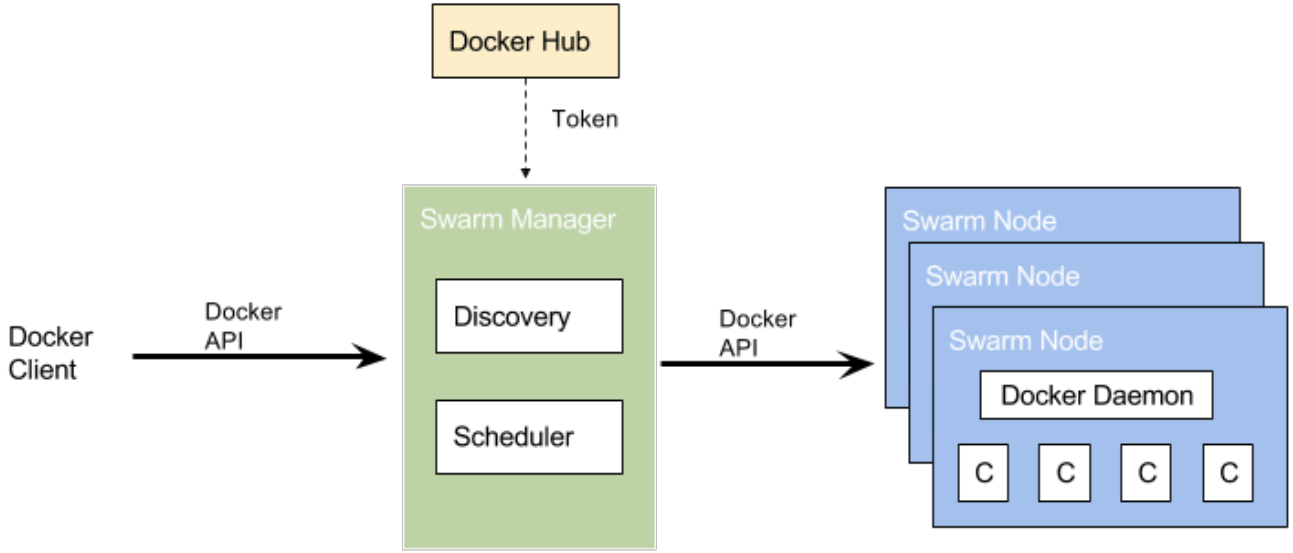
Kubernetes

- 结构复杂，概念众多
- 文档较少
- 功能强大
- 开发成本高

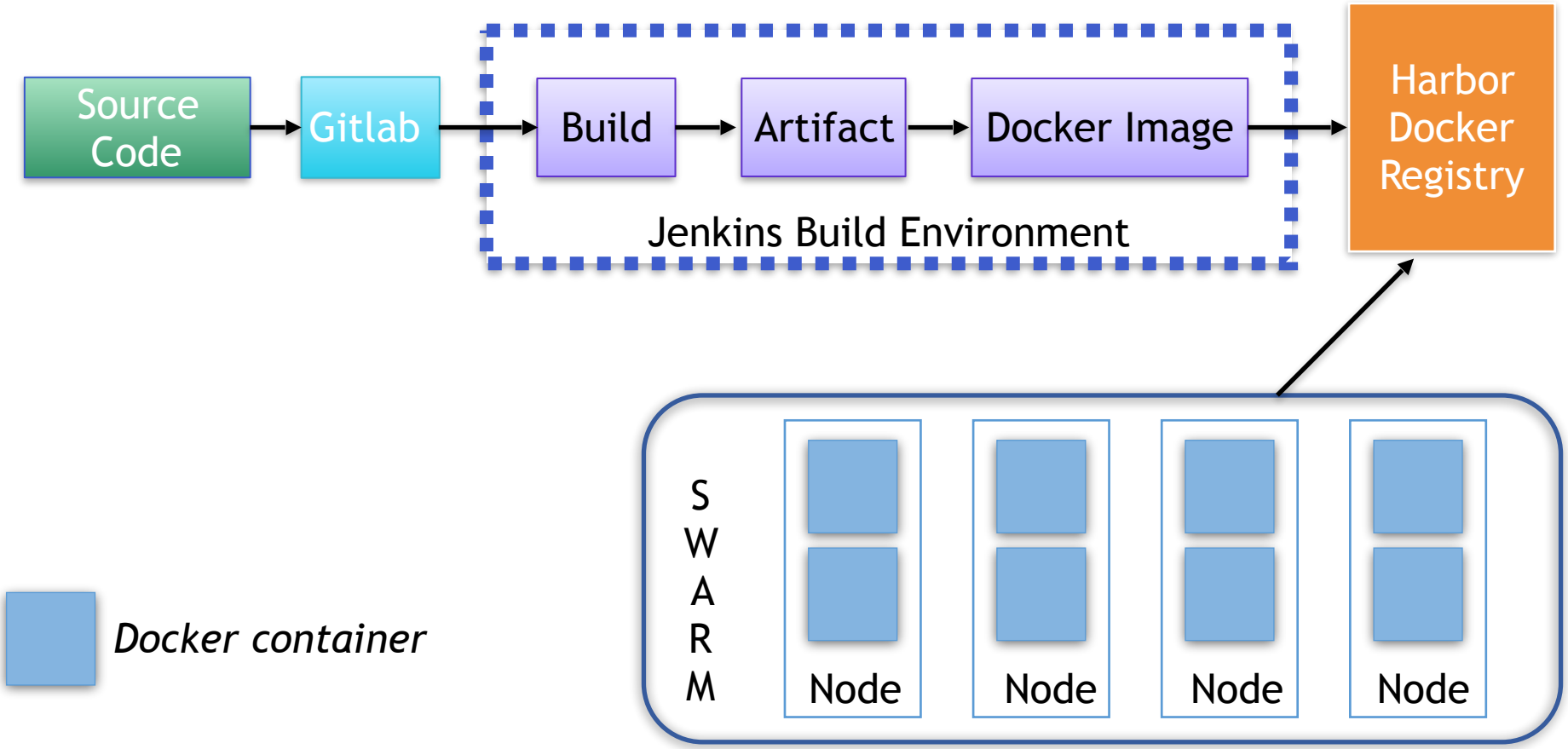


Swarm

- 兼容docker API
- 文档丰富
- 功能够用
- 插件机制



镜像制作与发布


Source Code
Gitlab
Build
Artifact
Docker Image
Jenkins Build Environment
Harbor Docker Registry
S W A R M
Node
Node
Node
Node
Docker container

Dockerfile


```
startup.sh
sed -i -E "s/NAMESERVICE/$NAMESERVICE/g"
$HADOOP_HOME/etc/hadoop/core-site.xml
```

```
ENTRYPOINT
[“startup.sh"]
CMD ["nodemanager"]
```

```
docker run -e
NAMESERVICE=ns1
```

Dockerfile

```
……
ENV HA
ENV NAMESERVICE
ENV ACTIVE_NAMENODE_IP
ENV …
……
ADD files to image
ENTRYPOINT [“startup.sh"]
CMD ["nodemanager"]
```

容器启动


```
docker run -d --net=mynet
-e NAMESERVICE=nameservice
-e ACTIVE_NAMENODE_ID=namenode29 \
-e STANDBY_NAMENODE_ID=namenode63 \
-e HA_ZOOKEEPER_QUORUM=zk1:2181,zk2:2181,zk3:2181 \
-e YARN_ZK_DIR=rmstore \
-e YARN_CLUSTER_ID=yarnRM \
-e YARN_RM1_IP=rm1 \
-e YARN_RM2_IP=rm2 \
-e CPU_CORE_NUM=5
-e NODEMANAGER_MEMORY_MB=12288 \
-e YARN_JOBHISTORY_IP=jobhistory \
-e ACTIVE_NAMENODE_IP=active-namenode \
-e STANDBY_NAMENODE_IP=standby-namenode \
-e HA=yes \
docker-registry/library/hadoop-yarn:v0.1 resourcemanager
```

集群管理

Yarn on Docker

```
usage: magpie.py [-h] [-o] [-d] [-s] [-v] [-c CLUSTERNAME] [-r CONTAINER]
                 [-n HOSTNAME] [-p PREFIX] [-i] [-w] [-m] [-u NUMBER]

Magpie is a Yarn-on-Docker operating tool.You can use this tool to inspect the
Docker and Yarn cluster, decommisioning the nodemanagers of a host, delete the
containers of a host.

optional arguments:
  -h, --help            show this help message and exit
  -o, --offline         Decommisioned nodemanagers on a host.
  -d, --delete          Delete all containers on the host no matter it is
                        running or not.
  -s, --scale           Scale the container number in the swarm cluster.
  -v, --view            Show the container distribution or not, default NOT
  -c CLUSTERNAME, --cluster CLUSTERNAME
                        Sepecify the yarn cluster.
  -r CONTAINER, --remove CONTAINER
                        Sepecify the container name or ID.
  -n HOSTNAME, --hostname HOSTNAME
                        Sepecify the hostname.
  -p PREFIX, --prefix PREFIX
                        Sepecify the new container name prefix.
  -i, --inspect         View of containers distribution on each yarn cluster.
  -w, --swarm           View of containers distribution on swarm cluster.
  -m, --compare         Compare the docker contianers and active nodemanagers.
  -u NUMBER, --number NUMBER
                        Specify the scaling container number.
```

Shipyard管理集群

Yarn on Docker

shipyard | CONTAINERS | IMAGES | NODES | REGISTRIES | ACCOUNTS | EVENTS | ADMIN

Refresh | Deploy Container | yarn1

Id	Node	Name	Image	Status	Created	Actions
374c48312654		yarn1add-nm145		Up 37 hours	2016-08-12 11:39:56 +0800	
7533fcb3c39d		yarn1add-nm146		Up 37 hours	2016-08-12 11:39:55 +0800	
c51af21fe7da		yarn1add-nm143		Up 2 days	2016-08-11 16:14:30 +0800	
015ea594c60f		yarn1add-nm144		Up 2 days	2016-08-11 16:14:30 +0800	
ba312670b622		yarn1add-nm141		Up 2 days	2016-08-11 14:09:42 +0800	
2790d5494bb6		yarn1add-nm142		Up 2 days	2016-08-11 14:09:41 +0800	
1b1e2bdecd38		yarn1add-nm137		Up 3 days	2016-08-10 17:08:10 +0800	

性能瓶颈与优化

iPerf带宽测试

a)不同主机容器间

Window Size	4K	16K	64K	256K	1M
Bandwidth	137 Mbits/sec	770 Mbits/sec	2.32 Gbits/sec	4.03 Gbits/sec	6.30 Gbits/sec

b)相同主机容器间

Window Size	4K	16K	64K	256K	1M
Bandwidth	1.06 Gbits/sec	5.13 Gbits/sec	13.3 Gbits/sec	32.2 Gbits/sec	37.8 Gbits/sec

c)不同主机间

Window Size	4K	16K	64K	256K	1M
Bandwidth	157 Mbits/sec	690 Mbits/sec	2.13 Gbits/sec	5.63 Gbits/sec	7.37 Gbits/sec

优化

docker默认使用devicemapper存储方式
默认容器存储空间10G
更改为overlayfs
Linux内核版本为3.10.0-327.el7.x86_64
解决容器存储空间不足问题

```
:/root]# docker info
Containers: 5
 Running: 5
 Paused: 0
 Stopped: 0
Images: 2
Server Version: 1.11.1
Storage Driver: overlay
 Backing Filesystem: xfs
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge null host
Kernel Version: 3.10.0-327.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
CPUs: 24
Total Memory: 62.64 GiB
```

Hadoop优化

yarn.nodemanager.localizer.fetch.thread-count默认4，改为8，随着容器数量增加，需要相应调整该参数

yarn.resourcemanager.amliveliness-monitor.interval-ms默认1秒，改为10秒，否则时间太短可能导致有些节点无法注册

yarn.resourcemanager.resource-tracker.client.thread-count默认50，改为100，随着容器数量增加，需要相应调整该参数

yarn.nodemanager.pmem-check-enabled默认true，改为false，不检查任务正在使用的物理内存量

容器中hadoop ulimit值修改，默认4096，改成655350

yarn.nodemanager.resource.cpu-vcores设置为4

yarn.nodemanager.resource.memory-mb设置为10240

RM页面

Yarn on Docker

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
64624	0	4	64620	64	372 GB	372 GB	228 GB	64	202	38	60	0	0	0	0

Application Queues

Legend: Capacity | Used | Used (over capacity) | Max Capacity

root — 100.0% used

default — 100.0% used

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
application_1469715169876_64626	mcloud	bitmap-topN-sample-count	SPARK	default	Mon Aug 1 15:56:06 +0800 2016	N/A	RUNNING	UNDEFINED	1	1	3072		ApplicationMaster
application_1469715169876_64625	mcloud	bitmap-count	SPARK	default	Mon Aug 1 15:51:04 +0800 2016	N/A	RUNNING	UNDEFINED	14	14	82944		ApplicationMaster
application_1469715169876_64624	mcloud	bitmap-count	SPARK	default	Mon Aug 1 15:51:04 +0800 2016	N/A	RUNNING	UNDEFINED	18	18	107520		ApplicationMaster
application_1469715169876_64623	mcloud	bitmap-count	SPARK	default	Mon Aug 1 15:51:04 +0800 2016	N/A	RUNNING	UNDEFINED	31	31	187392		ApplicationMaster

RM页面

Yarn on Docker

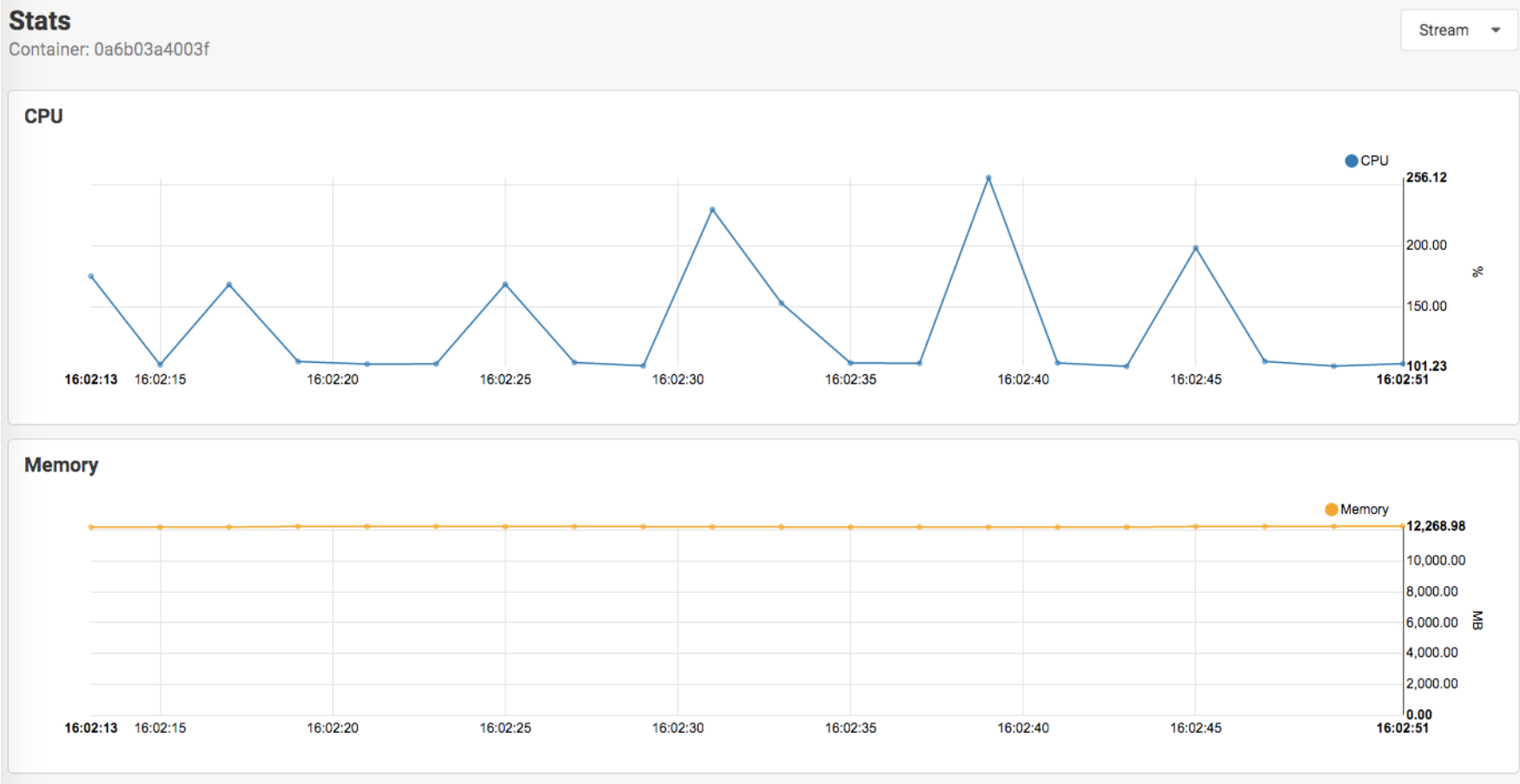
Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
64624	0	4	64620	64	372 GB	372 GB	228 GB	64	202	38	60	0	0	0	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	417b8e258a3a:56372	417b8e258a3a:8042	Mon Aug 01 16:10:15 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	eefc0d8fd9d2:54676	eefc0d8fd9d2:8042	Mon Aug 01 16:08:54 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	8aa78f108831:52572	8aa78f108831:8042	Mon Aug 01 16:08:57 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	587bd8f653e2:49407	587bd8f653e2:8042	Mon Aug 01 16:08:52 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	1f6fc7b90409:48667	1f6fc7b90409:8042	Mon Aug 01 16:08:55 +0800 2016		2	9 GB	1 GB	2	2	2.6.0-cdh5.5.2
	/default-rack	RUNNING	5a33a3aa756e:47703	5a33a3aa756e:8042	Mon Aug 01 16:08:51 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	f5c13e15fd97:58673	f5c13e15fd97:8042	Mon Aug 01 16:08:58 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	d73409c6237c:35820	d73409c6237c:8042	Mon Aug 01 16:08:56 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	2bee28dc361e:44188	2bee28dc361e:8042	Mon Aug 01 16:08:59 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2
	/default-rack	RUNNING	02f9e1282ccb:60780	02f9e1282ccb:8042	Mon Aug 01 16:10:22 +0800 2016		1	6 GB	4 GB	1	3	2.6.0-cdh5.5.2

监控

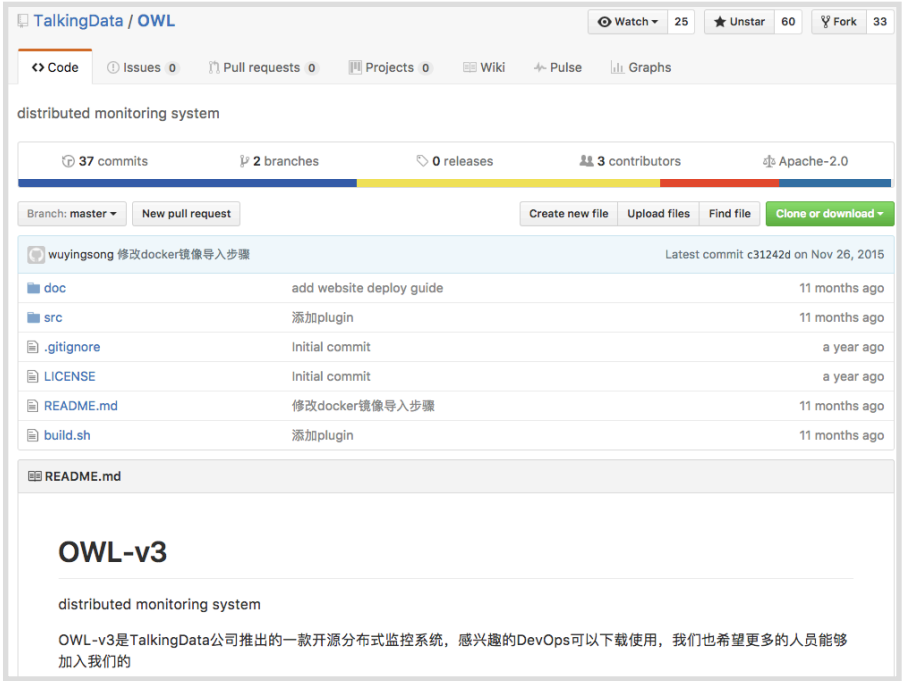

Stats
Container: 0a6b03a4003f
Stream
CPU
CPU
256.12
200.00
150.00
101.23
%
16:02:13
16:02:15
16:02:20
16:02:25
16:02:30
16:02:35
16:02:40
16:02:45
16:02:51
Memory
Memory
12,268.98
10,000.00
8,000.00
6,000.00
4,000.00
2,000.00
0.00
MB
16:02:13
16:02:15
16:02:20
16:02:25
16:02:30
16:02:35
16:02:40
16:02:45
16:02:51

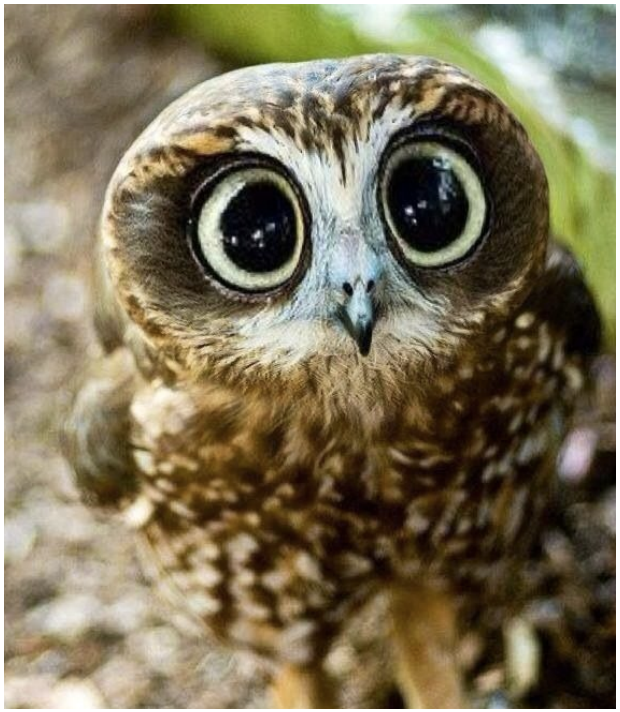
监控

基于大数据的**开源**监控平台OWL

https://github.com/TalkingData/OWL-v3

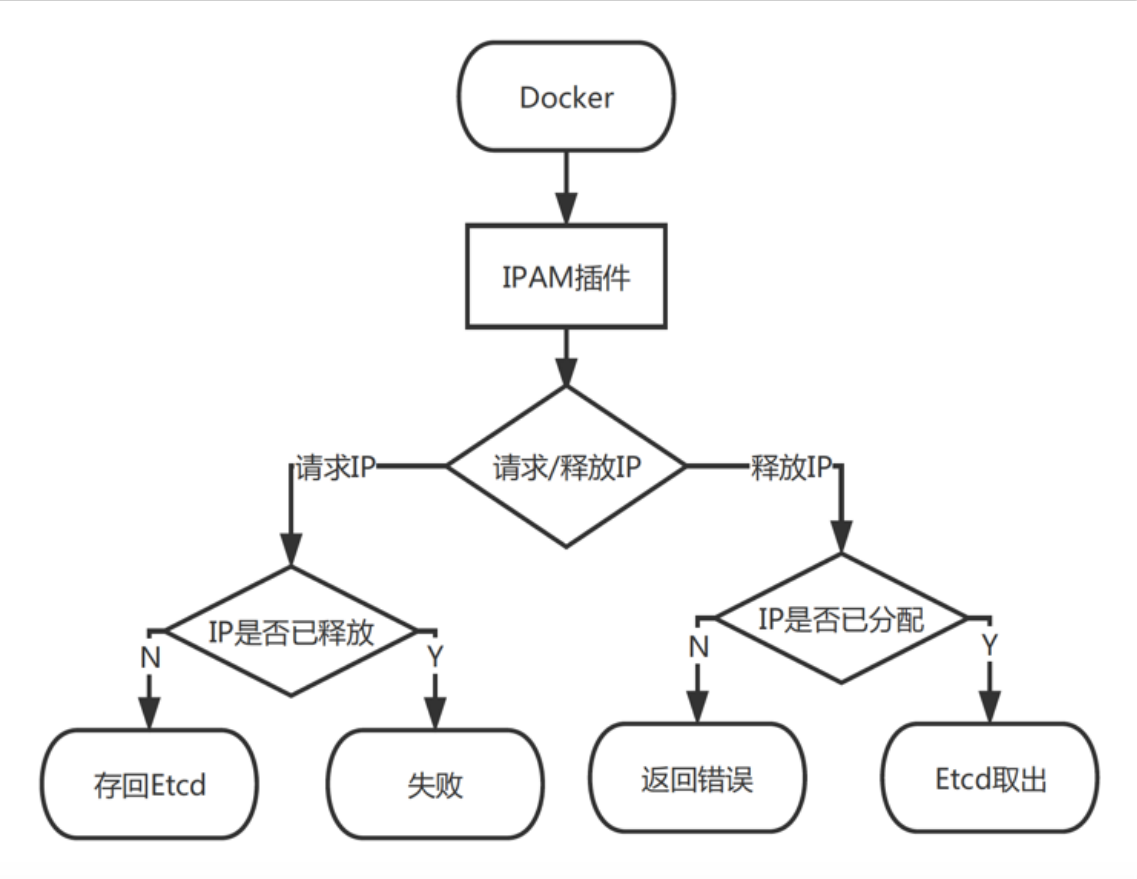
即将支持Docker监控





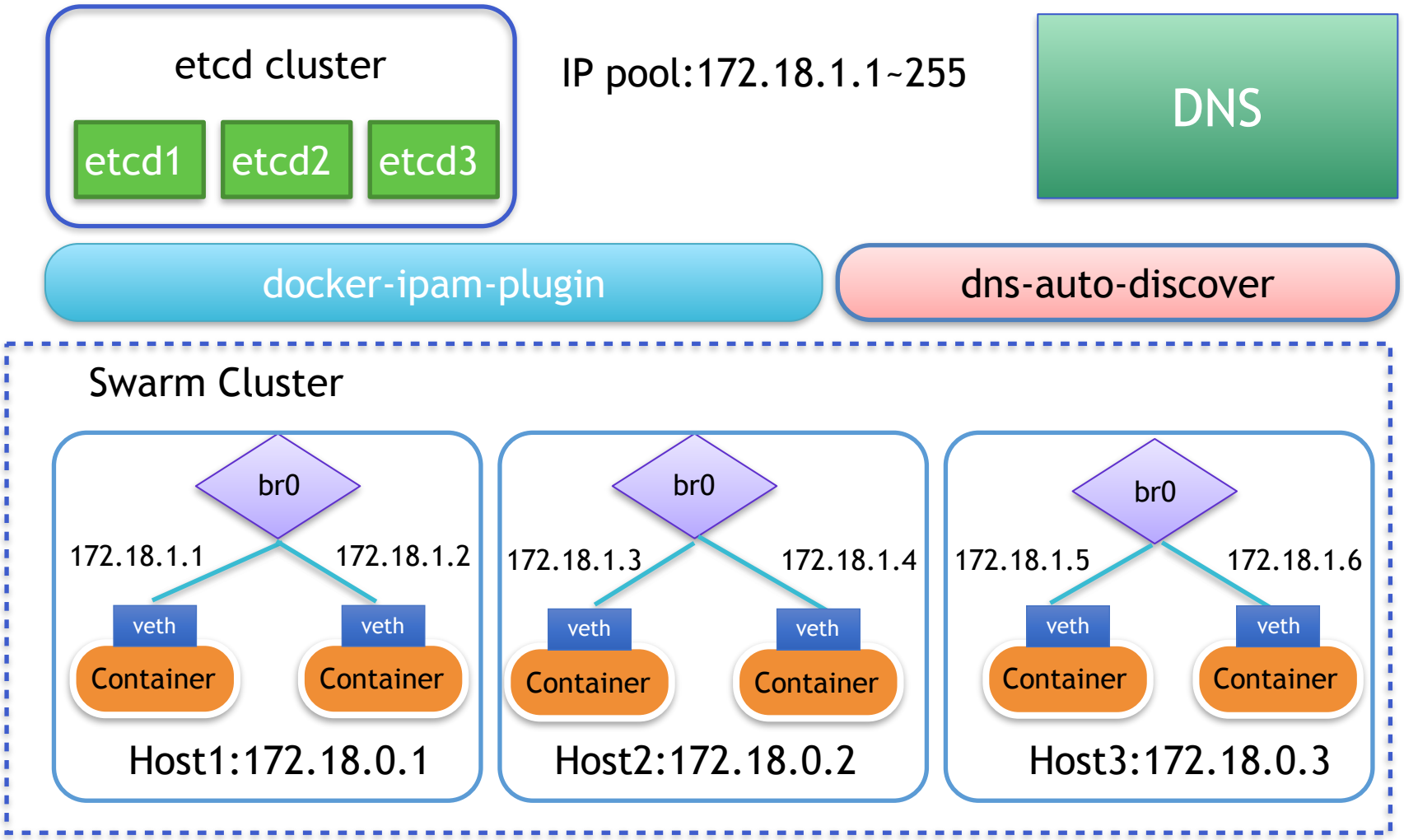
自定义网络

- 静态IP
- 二层网络
- 结构简单
- 性能优越



自定义网络

Yarn on Docker



开源

Docker自定义网络插件：https://github.com/TalkingData/Shrike

TalkingData / Shrike

Unwatch 9 | Unstar 4 | Fork 2

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

Docker扁平二层网络工具与Swarm集群管理工具

10 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

chao ma added some comments — Latest commit 11dbbe6 2 days ago

oam-docker-ipam	added some comments	2 days ago
rpms	init project files	10 days ago
README.md	modify doc	10 days ago

README.md

Shrike

开发背景

众所周知，Docker容器跨主机互访一直是一个问题，Docker官方为了避免网络上带来的诸多麻烦，故将跨主机网络开了比较大的口子，而由用户自己去实现。

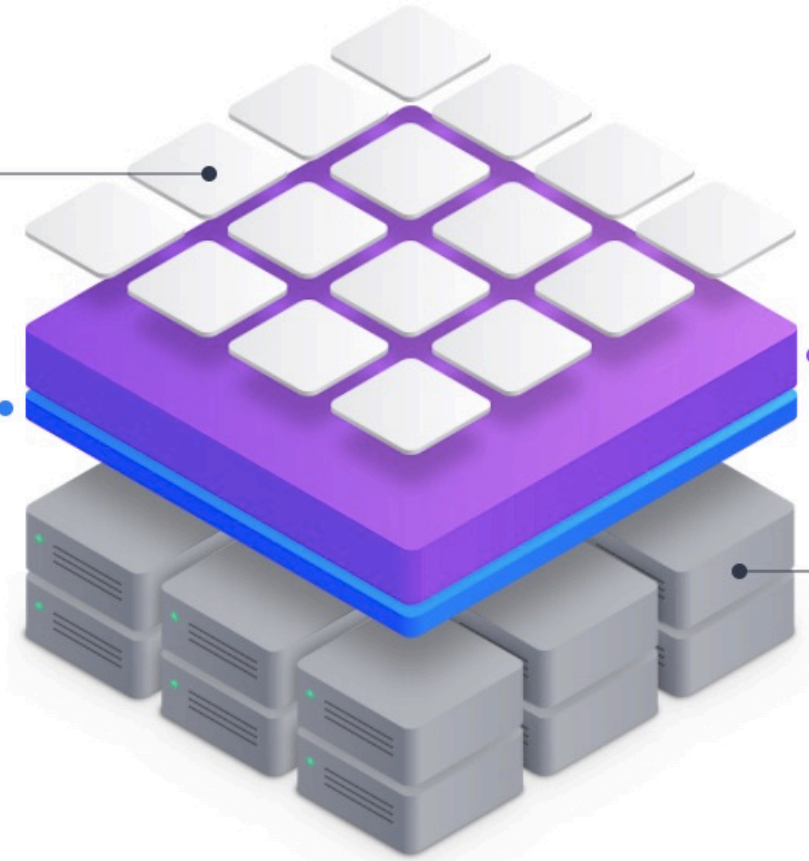
目前Docker跨主机的网络实现方案也有很多种，主要包括端口映射，ovs, fannel等。但是这些方案都无法满足我们的需求，端口映射服务内的内网IP会映射成外网的IP，这样会给开发带来困惑，因为他们往往在跨网络交互时是不需要内网IP的，而ovs与fannel则是在基础网络协议上又包装了一层自定义协议，这样当网络流量大时，却又无端的增加了网络负载，最后我

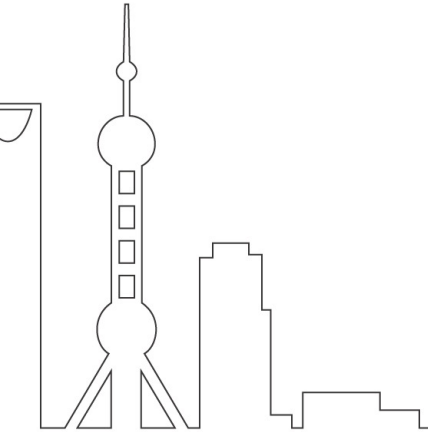
Future

- Service Control Panel：统一的根据服务来管理的web页面
- Load balance：容器根据机器负载情况自动迁移
- Scheduler：swarm调度策略优化
- 服务配置文件：提供镜像启动参数的配置文件，所有启动参数可通过文件配置
- 监控：服务级别的监控

DC/OS

Yarn on Docker





Thanks!

International Software Development Conference

宋净超

主办方 Geekbang 极客邦科技 InfoQ