

从Docker到Kubernetes 第7周

DATAGURU专业数据分析社区

【声明】 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散布，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

<http://edu.dataguru.cn>



- Docker管理工具
- Shipyard入门
- cAdvisor入门

Docker Machine是什么鬼

从前

你需要登录主机，按照主机及操作系统特有的安装以及配置步骤安装Docker，使其能运行Docker容器。

现在

Docker Machine的产生简化了这一过程，让你可以使用一条命令在你的计算机，公有云平台以及私有数据中心创建及管理Docker主机。

Create Docker Machine主要包括三个Create过程。

- 首先是Provider Create（libmachine/provider.go），此函数主要是在当前运行docker-machine命令主机上创建以machine name命名的文件夹，并将根证书，服务器证书以及用户证书拷贝到此文件夹。
- 其次是Driver create（例如drivers/virtualbox/virtualbox.go）用来创建主机，
- 最后是运行Host create（libmachine/host.go）通过SSH安装并配置Docker。目前在本地环境中使用的是boot2docker镜像，云端环境使用的是Ubuntu镜像。

其实真相是这样的：自动创建一个虚机并且安装好设置好Docker Engine

Docker Machine是什么鬼

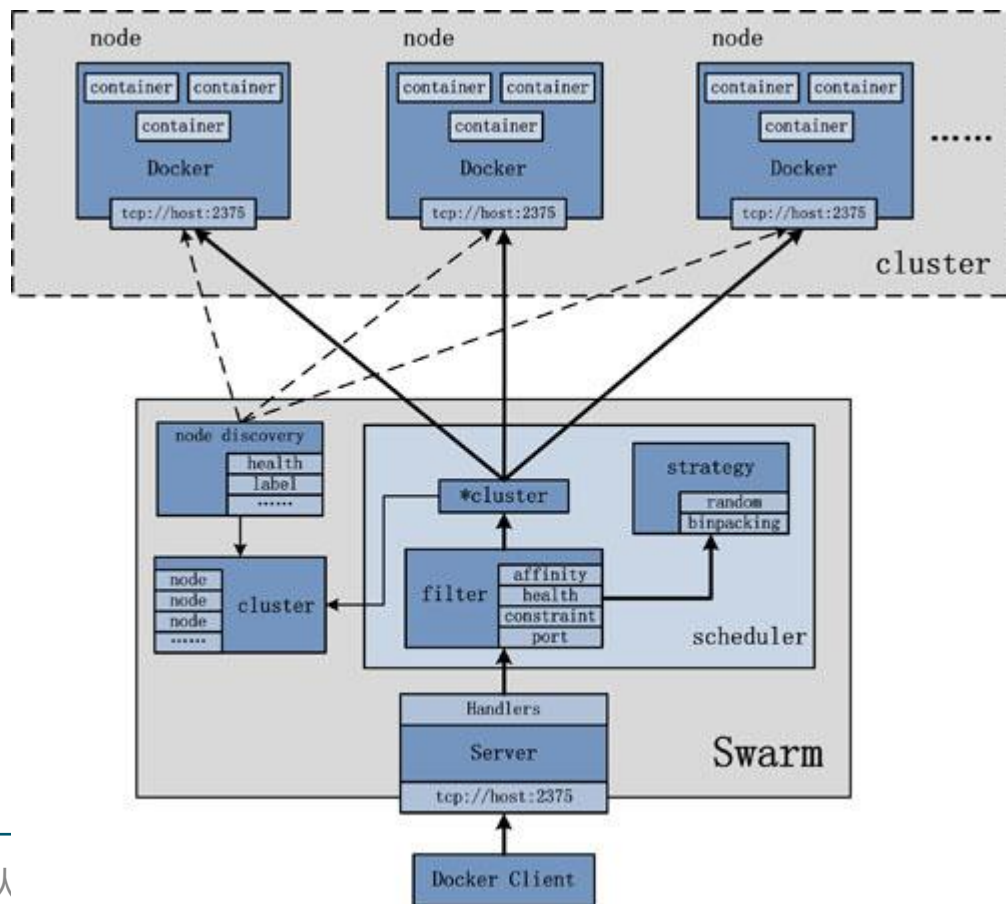
- Docker Machine简化了部署的复杂度，无论是在本机的虚拟机上还是在公有云平台，只需要一条命令便可搭建好Docker主机
- Docker Machine提供了多平台多Docker主机的集中管理
- Docker Machine 使应用由本地迁移到云端变得简单，只需要修改一下环境变量即可和任意Docker主机通信部署应用。

为什么会有Docker Compose



与容器技术同样受到关注的微服务架构也在潜移默化的改变着应用的部署方式，其提倡将应用分割成一系列细小的服务，每个服务专注于单一业务功能，服务之间采用轻量级通信机制相互沟通

在很长的一段时间内，Docker只能在单host上运行，其跨host的部署、运行与管理能力颇受外界诟病。跨host能力的薄弱，直接导致Docker容器与host的紧耦合，这种情况下，Docker容器的灵活性很难令人满意，容器的迁移、分组等都成为很难实现的功能点。Swarm发布于2014年12月，以管理Docker集群，并将其抽象为一个虚拟整体暴露给用户，其架构以及命令比较简单。



Swarm作为一个管理Docker集群的工具，可以单独部署于一个节点。

Swarm的具体工作流程：Docker Client发送请求给Swarm；Swarm处理请求并发送至相应的Docker Node；Docker Node执行相应的操作并返回响应。

1. 运行一个命令去创建一个集群.
2. 运行另一个命令去启动Swarm.
3. 在运行有Docker Engine的每个主机上，运行一个命令与上面的集群相连

在某些点, Swarm将可以在主机故障时重调度容器.

Swarm可以很好地与第三方容器编配产品和运供应商提供的编配服务整合，如Mesos

Docker Swarm

swarm则将一组docker engine作为一个集群进行管理，并提供过了label, schedule, filter的能力。其中调度部分，允许用户定制自己的调度策略。

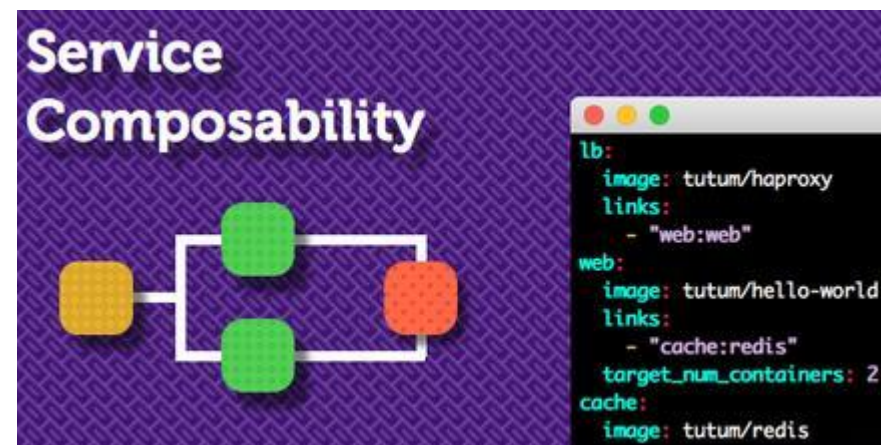
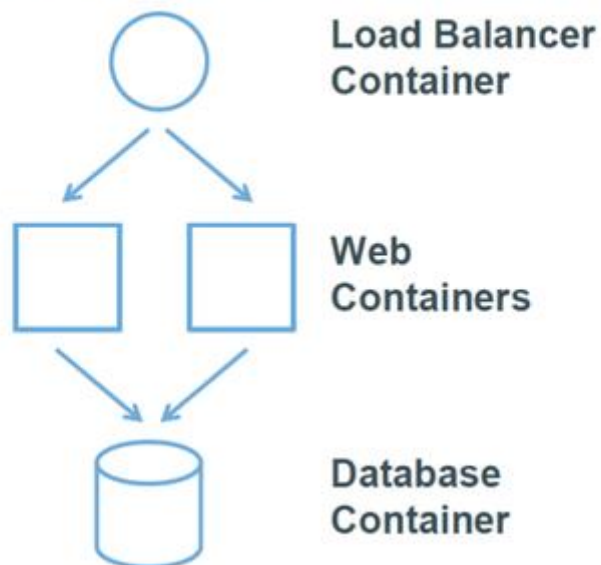
1. `docker run -e "constraint: operationsystem=fedora"`
2. `docker run -e "constraint: storagedriver=aufs"`

Docker Compose是什么鬼

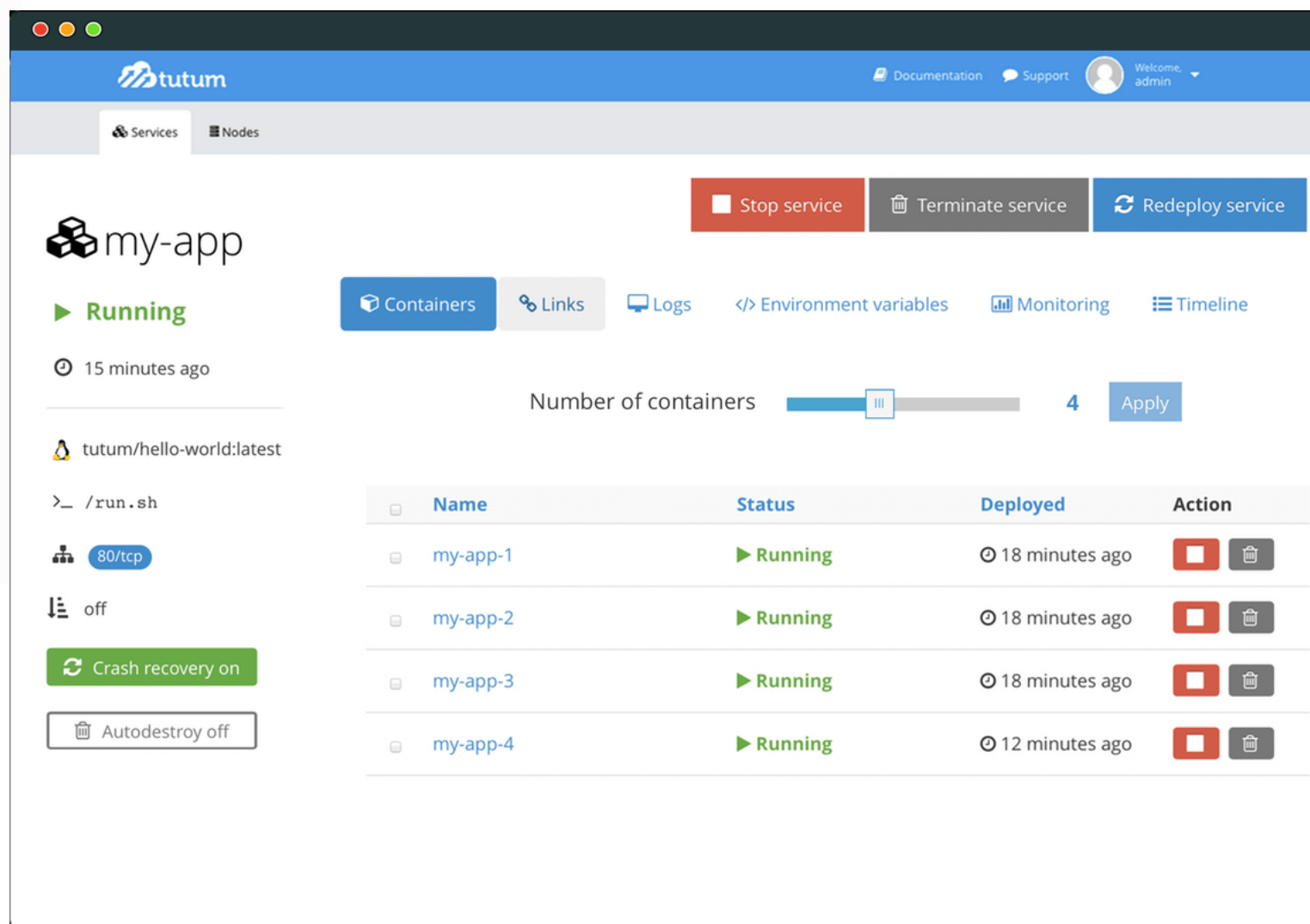
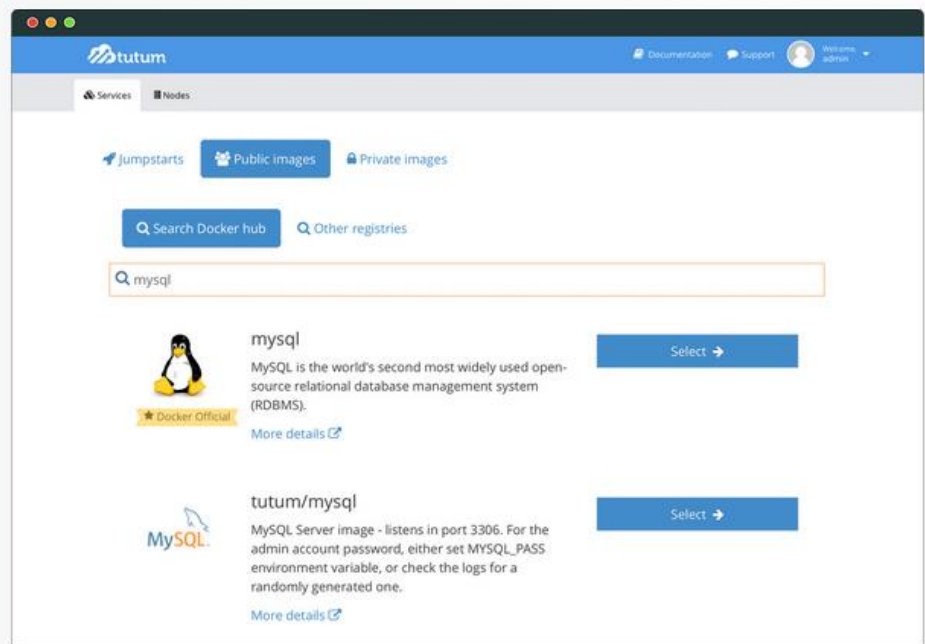
Docker Compose将所管理的容器分为三层，工程（project），服务（service）以及容器（container）。一个工程当中可包含多个服务，每个服务中定义了容器运行的镜像，参数，依赖。一个服务当中可包括多个容器实例，**Docker Compose**并没有解决负载均衡的问题，因此需要借助其他工具实现服务发现及负载均衡。

Docker Compose中定义构建的镜像只存在在一台Docker Swarm主机上，无法做到多主机共享

Docker Compose



Docker管理工具——Tutum



Docker管理工具——shipyard



Shipyard 是一个基于 Web 的 Docker 管理工具，支持多 host，可以把多个 Docker host 上的 containers 统一管理；可以查看 images，甚至 build images；并提供 RESTful API 等等。Shipyard 要管理和控制 Docker host 的话需要先修改 Docker host 上的默认配置使其支持远程管理。

[Explore](#) [Features](#) [Enterprise](#) [Pricing](#) [Sign up](#) [Sign in](#)

[shipyard / shipyard](#) [Watch](#) 235 [★ Star](#) 3,883 [Fork](#) 388

Composable Docker Management <http://shipyard-project.com>

1,063 commits 5 branches 7 releases 32 contributors

Branch: master [shipyard / +](#)

Merge pull request #633 from gerco/master ...

ehazlett authored 4 days ago latest commit 5a6b9b9cf8

[Code](#)
[Issues](#) 40
[Pull requests](#) 3
[Wiki](#)

Docker管理工具——shipyard

所有需要纳管的Docker主机，需要让Docker在TCP上监听，以便被纳管

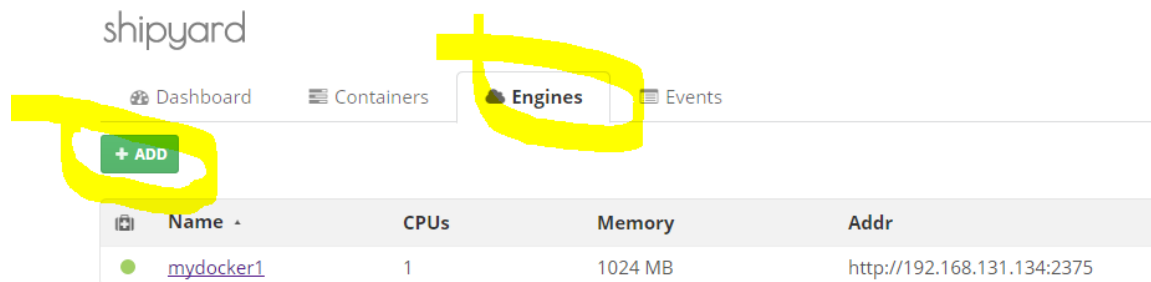
OPTIONS=-H=unix:///var/run/docker.sock -H=tcp://0.0.0.0:2375

安装shipyard

docker run --rm -v /var/run/docker.sock:/var/run/docker.sock \ shipyard/deploy start

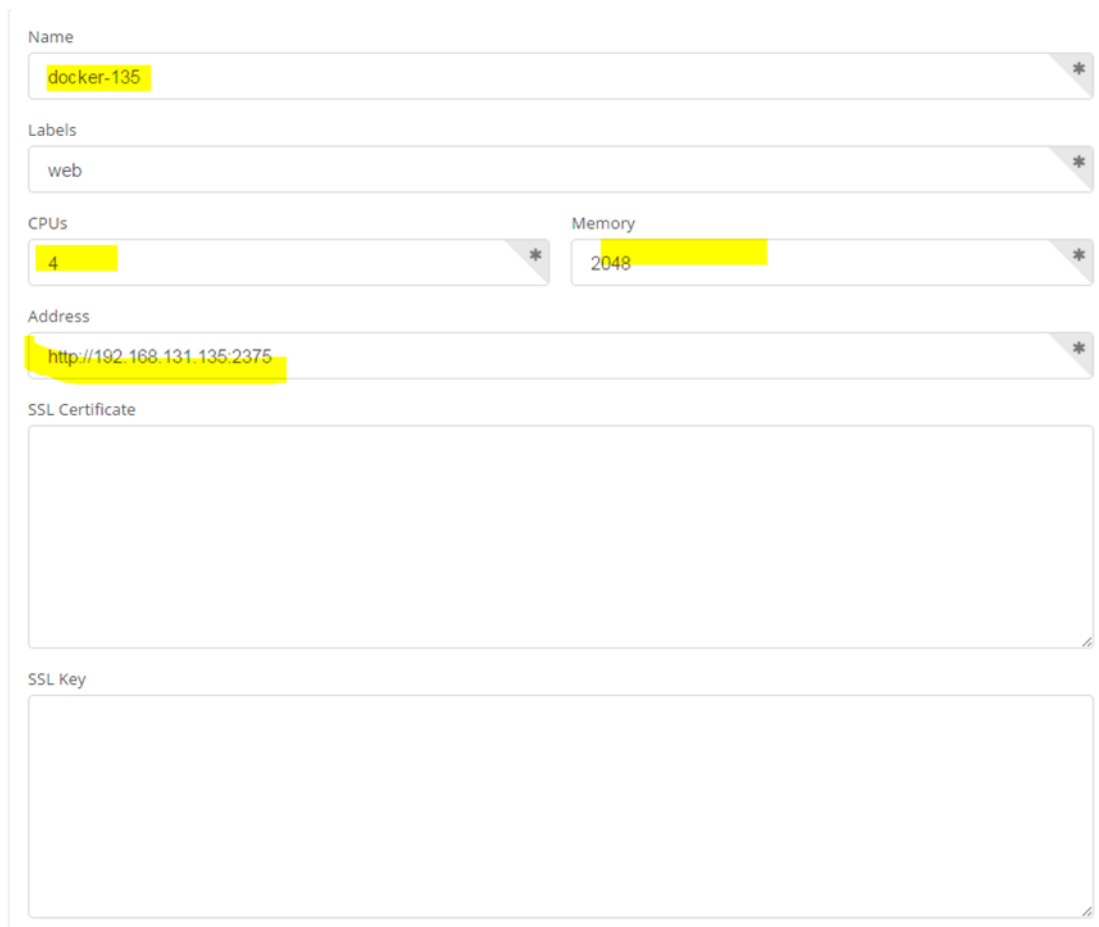
运行：

浏览器访问物理机的8080端口：<http://192.168.131.134:8080/> 默认用户名密码:admin/shipyard，， 点击Engines标签页，添加一个Docker主机（Engine）：



Docker管理工具——shipyard

填写Docker主机的名字、CPU核心数、内存数量（单位MB）、以及Labels，比如部署Web的标签为web，以后调度容器的时候，会优先调度到相应的标签。



The image shows a web form for configuring a Docker container. The fields are as follows:

- Name:** A text input field containing "docker-135".
- Labels:** A text input field containing "web".
- CPUs:** A text input field containing "4".
- Memory:** A text input field containing "2048".
- Address:** A text input field containing "http://192.168.131.135:2375".
- SSL Certificate:** A large, empty text area.
- SSL Key:** A large, empty text area.

Docker管理工具——shipyard

创建成功后，列表页面显示Docker主机的版本信息：

Dashboard


Containers

Engines

Events

+ ADD

如果某个Docker主机停止（或者无法连接了），页面上可以看到状态变化：

 Name ^	CPU's	Memory	Addr	Labels	Response Time (ms)	Docker Version
 docker-135	4	2048 MB	http://192.168.131.135:2375	web	0.8	1.5.0-dev
 mydocker1	1	1024 MB	http://192.168.131.134:2375	service	0.6	1.5.0-dev

如果添加失败，则需要排除是否端口不可访问，可以用wget/telnet/curl等方式来排查问题，下面截图是在另外主机上Telnet Docker 2375端口看是否能连接：

```
HTTP/1.1 400 Bad Request

遗失对主机的连接。

C:\Users\wuzhih>telnet 192.168.131.135 2375
```

Docker管理工具——shipyard

Containers标签页显示了所有的Docker容器，可以按照Engine（主机）排序，目前还没有分页和查询功能。

shipyard

Status	ID	Name	Image	Engine	CPUs	Memory
●	65e5c37b41d7	thirsty_wilson	ubuntu:latest	docker-135	∞	∞
●	fcf76913dbda	high_jones	centos:latest	docker-135	∞	∞
●	a1a21b1d1294	myapp3	java:latest	docker-135	∞	∞
●	b0195b6a9784	myapp2	java:latest	docker-135	∞	∞
●	5961f1fbc5e	mysql3	mysql:latest	docker-135	∞	∞
●	ca5a141a3700	mysql2	mysql:latest	docker-135	∞	∞
●	1f34e2a067ac	mysql1	mysql:latest	docker-135	∞	∞
●	24cb02c1585b	fervent_lovelace	centos:latest	docker-135	∞	∞
●	6d93faeee289	loving_goodall	ubuntu:latest	docker-135	∞	∞
●	15a00cdd9ad	jolly_perlman	ubuntu:latest	docker-135	∞	∞
●	97b0c9e852fc	fervent_sinoussi	ubuntu:latest	docker-135	∞	∞
●	2-766b70-8-0	-----1	-----latest	docker-135	∞	∞

Docker管理工具——shipyard

点击Container标签页的Deploy按钮，可以启动新的容器

shipyard

Dashboard

Containers

Engines

Events

Image

mysql

*

Hostname

foo

Domain

example.com

Container Name

my-mysql-server

Environment

MYSQL_ROOT_PASSWORD=123456

Arguments

(space separated)

Volumes

(space separated: /host/path:/container/path or /container/path)

Links

(space separated, container:name format)

CPU

0.1

*

Memory (MB)

256

*

Count

1

Ports

+

NO PORT BINDINGS SPECIFIED

☐ Publish all Exposed Ports

☐ Privileged

Type

service

Restart Policy

no

Network

host

Labels

☒ service ☐ web

DEPLOY

Docker管理工具——shipyard



容器的镜像，名称，环境变量，启动参数，是否有Volume存储、是否有端口映射等都可以在界面上定义。启动成功以后，可以看到列表中新容器的状态为RUNNING：

+ DEPLOY

Status	ID	Name	Image	Engine ^	CPUs
●	96233bda95a3	serene_feynman	ubuntu:latest	docker-135	∞
●	fcf76913dbda	high_jones	centos:latest	docker-135	∞
●	92255d7771aa	gloomy_ritchie	mysql:latest	docker-135	0.08
●	0a408ea58971	app1-mysql	mysql:latest	docker-135	0.08
●	6591e8043473	mytest_mysql	mysql:latest	docker-135	0.08
●	f9692c142487	nervcoreserver	java:latest	docker-135	∞
●	a1a21b1d1294	mvapp3	java:latest	docker-135	∞

Docker管理工具——shipyard

点击容器的链接，可以看到容器的细节信息，如端口、CPU占用、内存占用、环境变量、重启策略等，还可以重启、停止、销毁容器或者查看容器日志。

shipyard

Dashboard Containers Engines Events

Containers 0a408ea58971

RESTART STOP DESTROY SCALE LOGS

Name

app1-mysql

Image

mysql:latest

Type

service

Environment

MYSQL_MAJOR
5.6

HIDE

MYSQL_ROOT_PASSWORD
123456

HIDE

MYSQL_VERSION
5.6.24

HIDE

Hostname

mycomputer

Engine

docker-135

Network Mode

host

Restart Policy

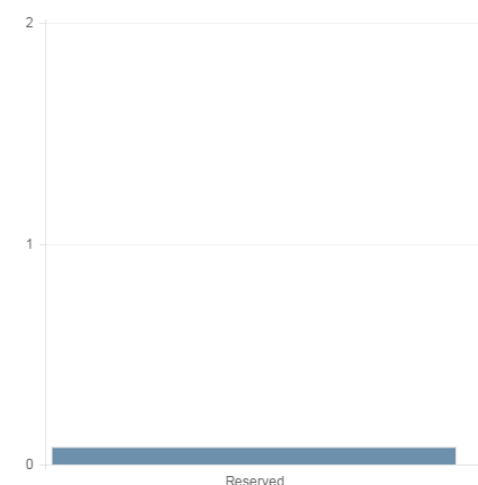
no

Privileged

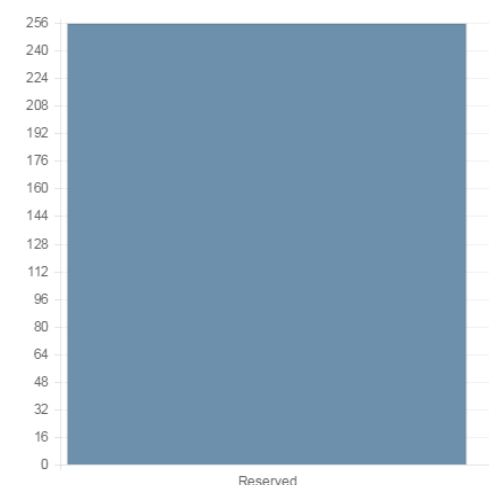
Ports

tcp/ -> 192.168.131.135:3306

CPU



Memory



如果容器启动失败，或者排查问题，则可以查看容器的日志信息：

```
Finished mysql_install_db
2015-06-01 08:55:02 0 [Note] mysqld (mysqld 5.6.24) starting as process 1 ...
2015-06-01 08:55:02 1 [Note] Plugin 'FEDERATED' is disabled.
2015-06-01 08:55:02 1 [Note] InnoDB: Using atomics to ref count buffer pool pages
2015-06-01 08:55:02 1 [Note] InnoDB: The InnoDB memory heap is disabled
2015-06-01 08:55:02 1 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
2015-06-01 08:55:02 1 [Note] InnoDB: Memory barrier is not used
2015-06-01 08:55:02 1 [Note] InnoDB: Compressed tables use zlib 1.2.7
2015-06-01 08:55:02 1 [Note] InnoDB: Using Linux native AIO
2015-06-01 08:55:02 1 [Note] InnoDB: Using CPU crc32 instructions
2015-06-01 08:55:02 1 [Note] InnoDB: Initializing buffer pool, size = 128.0M
2015-06-01 08:55:02 1 [Note] InnoDB: Completed initialization of buffer pool
2015-06-01 08:55:02 1 [Note] InnoDB: Highest supported file format is Barracuda.
2015-06-01 08:55:02 1 [Note] InnoDB: 128 rollback segment(s) are active.
2015-06-01 08:55:02 1 [Note] InnoDB: Waiting for purge to start
2015-06-01 08:55:02 1 [Note] InnoDB: 5.6.24 started; log sequence number 1625987
2015-06-01 08:55:02 1 [Warning] No existing UUID has been found, so we assume that this is the first time that thi
2015-06-01 08:55:02 1 [Note] Server hostname (bind-address): '*' ; port: 3306
2015-06-01 08:55:02 1 [Note] IPv6 is available.
2015-06-01 08:55:02 1 [Note] - '::' resolves to '::';
2015-06-01 08:55:02 1 [Note] Server socket created on IP: '::'.
2015-06-01 08:55:02 1 [Note] Event Scheduler: Loaded 0 events
2015-06-01 08:55:02 1 [Note] Execution of init_file '/tmp/mysql-first-time.sql' started.
2015-06-01 08:55:02 1 [Note] Execution of init_file '/tmp/mysql-first-time.sql' ended.
2015-06-01 08:55:02 1 [Note] mysqld: ready for connections.
Version: '5.6.24' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)
```

Docker管理工具——shipyard

[Dashboard](#) [Containers](#) [Engines](#) [Events](#)

containers / f49b36fa9184

RESTARTSTOPDESTROYSCALELOGS

Name

auto-mysql

Image

mysql:latest

Type

service

Environment

MYSQL_MAJORSHOW

MYSQL_ROOT_PASSWORDSHOW

MYSQL_VERSIONSHOW

Hostname

mycomputer

Engine

docker-135

Network Mode

host

Restart Policy

always

Privileged

Ports

[tcp/-> 192.168.131.135:3306](#)

该容器会有如下的自我修复能力

- 该容器的进程被意外Kill而停止时，shipyard能监测到这个变化，并且自动重启此容器。
- 如果该容器是正常stop的，如登录到主机上执行docker stop命令，则此容器不会自动重启。

在Web的页面上，我们能观察到这些事件：

Events

Container	Engine	Type
f49b36fa9184	docker-135	start
f49b36fa9184	docker-135	die
f49b36fa9184	docker-135	start
f49b36fa9184	docker-135	die
f49b36fa9184	docker-135	stop
f49b36fa9184	docker-135	die
f49b36fa9184	docker-135	start
f49b36fa9184	docker-135	start
f49b36fa9184	docker-135	die
f49b36fa9184	docker-135	restart
f49b36fa9184	docker-135	start
f49b36fa9184	docker-135	die
f49b36fa9184	docker-135	create
f49b36fa9184	docker-135	start
5581cfee6b95	docker-135	kill

cAdvisor的监控图默认1秒刷新一次，显示最近一分钟的实时数据，不显示汇聚的和历史数据，也没有阈值告警功能，此外它也无法同时监控多个Docker主机，不过由于其简单方便，并且具备很好的实时性能监控能力，所以适合特殊情况下的性能监控和问题排查。

google的cAdvisor，免费开源，实施简单，每个Docker主机上启动一个容器即可通过Web端口监控，

```
docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --  
volume=/var/lib/docker:/var/lib/docker:ro --publish=8082:8082 --detach=true --name=cadvisor  
google/cadvisor:latest --port=8082
```

上述部分参数可能与主机操作系统有关，需要修改，可参照官方文档：<https://github.com/google/cadvisor>

由于shipyard是在本机8080端口运行，因此上面把cAdvisor改为了8082端口，运行起来后，访问本机8082端口，可看到监控界面：

Docker管理工具——cAdvisor

Isolation

CPU

Shares 1024 *shares*

Allowed Cores 0 1

Memory

Limit unlimited

Swap Limit unlimited

Usage

Overview



Docker管理工具——cAdvisor

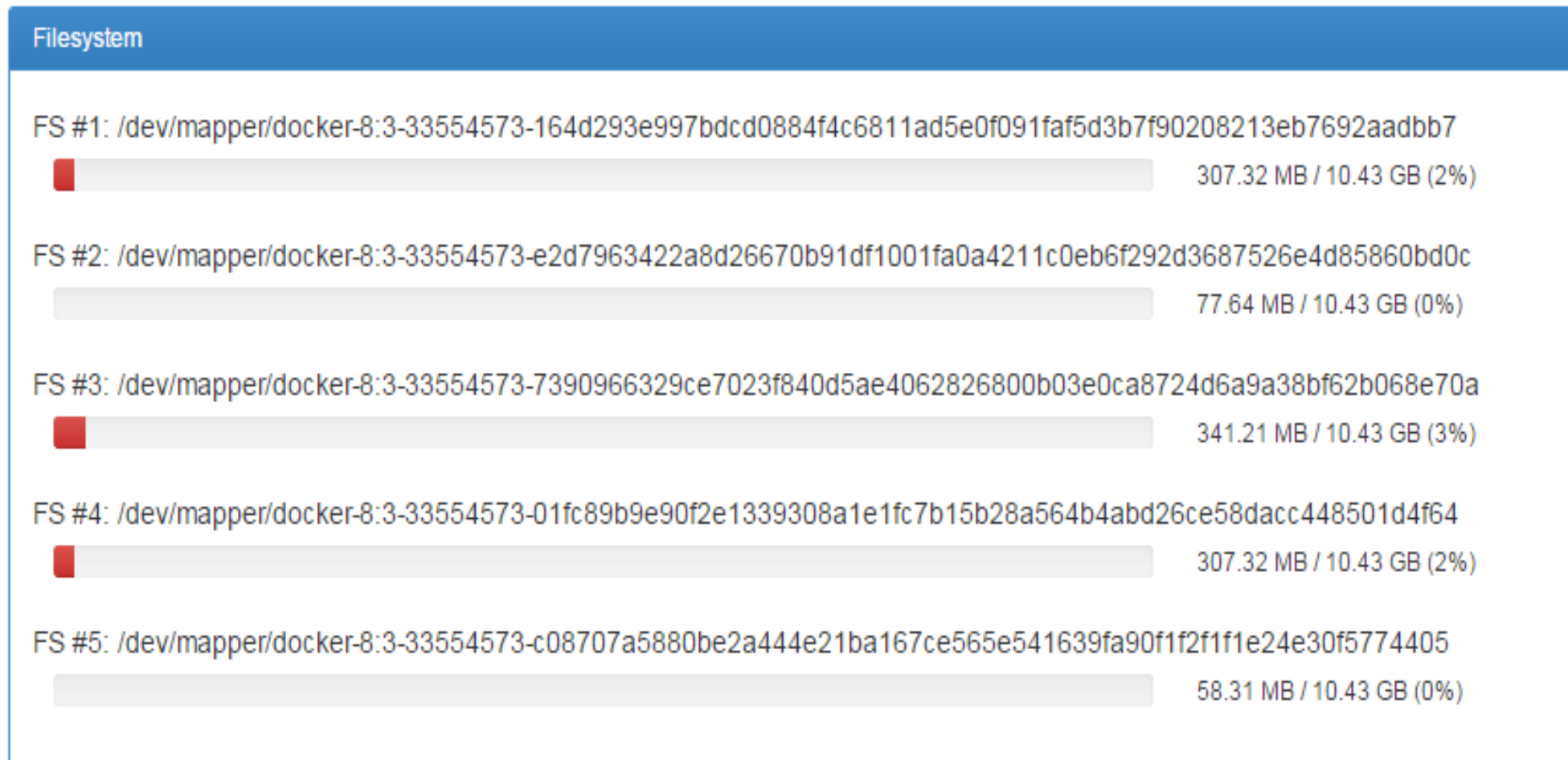


Docker管理工具——cAdvisor



Docker管理工具——cAdvisor

显示了当前活动的容器的磁盘占用情况



Thanks

FAQ时间