

Native Docker clustering with Swarm

libros, tech, docker, swarm

chap1

- boot2docker:
 - distribución Linux ligera desarrollada para Docker
 - corre totalmente en RAM
 - tarda 5 segundos en arrancar
 - arranca con Docker TLS en :2376
 - para desarrollo y test
 - soporte para <https://docs.docker.com/storage/storagedriver/aufs-driver/>
- docker-machine
 - `docker-machine create --driver virtualbox node0`
 - permite otros proveedores (AWS,Azure,DigitalOcean)
 - los autores del libro tienen un wrapper para hablar con DigitalOcean: <https://github.com/chanwit/belt>
 - `eval $(docker-machine env node0)`: te configura el entorno para trabajar con el demonio docker de node0 (en este caso)
 - `docker-machine ls`
 - `docker-machine active`
 - `docker-machine ssh node0 ps aux | grep docker`
- swarm:
 - **nodes://**: conectar nodos directamente por sus IPs (aprendizaje)

chap2: discovery services

- necesidad de disponer de un servicio de descubrimiento para localizar aquello que buscas
 - con pocos nodos y configuraciones simples no sería necesario, tu sabes donde está todo aquello que necesitas
 - con muchos nodos, cambiantes, y cientos de contenedores, es imposible de gestionar (cambian dinámicamente de IP, por ejemplo)
- existen muchos, pero todos ellos requieren:
 - sistemas distribuidos en todos los nodos
 - escalables
 - tolerancia a fallos
 - registro
 - anunciar
 - almacenaje key-value
- swarm v1
 - no integra uno propio
 - integrar el tuyo propio a través de *libkv*
 - token
 - Consul
 - Etcd (se ha acabado integrando)
 - ZooKeeper

token

- `docker run ... token://$TOKEN`
- requiere conexión a internet de los nodos y acceso a Docker Hub

- se ha de generar un UUID de swarm (swarm create)
- se utiliza para unir nodos y hablar con el manager
- se acabará deprecando

raft

- algoritmo para consensuar en sistemas distribuidos la elección del lider y la consistencia de los valores
- otro: paxos (más complejo y difícil de comprender)
- raft: Consul, Etcd
- paxos: ZooKeeper
- <https://ramcloud.stanford.edu/raft.pdf>

teoría de funcionamiento

- simplicidad
- mensajes y logs son solo enviados del lider del cluster a sus miembros
- un cluster basado en este algoritmo debería mantenerse replicado de una manera consistente, indistintamente de lo que pase: nevos nodos, caída de otros
- número impar de nodos (para evitar split-brains)
- en condiciones normales, hay un lider que mantiene informados a los seguidores de su estado (heartbeat). Si e lider, falla, los seguidores entienden que ha caído y buscan un nuevo lider (de manera consensuada)
- los mensajes, antes de ser guardados en el registro principal del lider, son enviados a los seguidores y sólo cuando una mayoría ha confirmado su recepción, este es guardado.

Etcd

- sistema de descubrimiento y compartición de configuración, en alta disponibilidad, distribuido y consistente key-value
- soporta la caída de nodos (incluso el master), tiene un sistemna de elección de master.
- los contenedores pueden leer y escribir en el almacen de Etcd
- :2379 (comunicaciones cliente)
- :2380 (comunicaciones master)
- :4001
- `etcdctl cluster-health`
- swarm v1 (lanzando servicio swarm como comentenedor y enlazando con un cluster etcd previo):

```
docker run -d -p 3376:3376 swarm manage -H tcp://0.0.0.0:3376 etcd://$(docker-machine ip etcd-m)/swarm
```

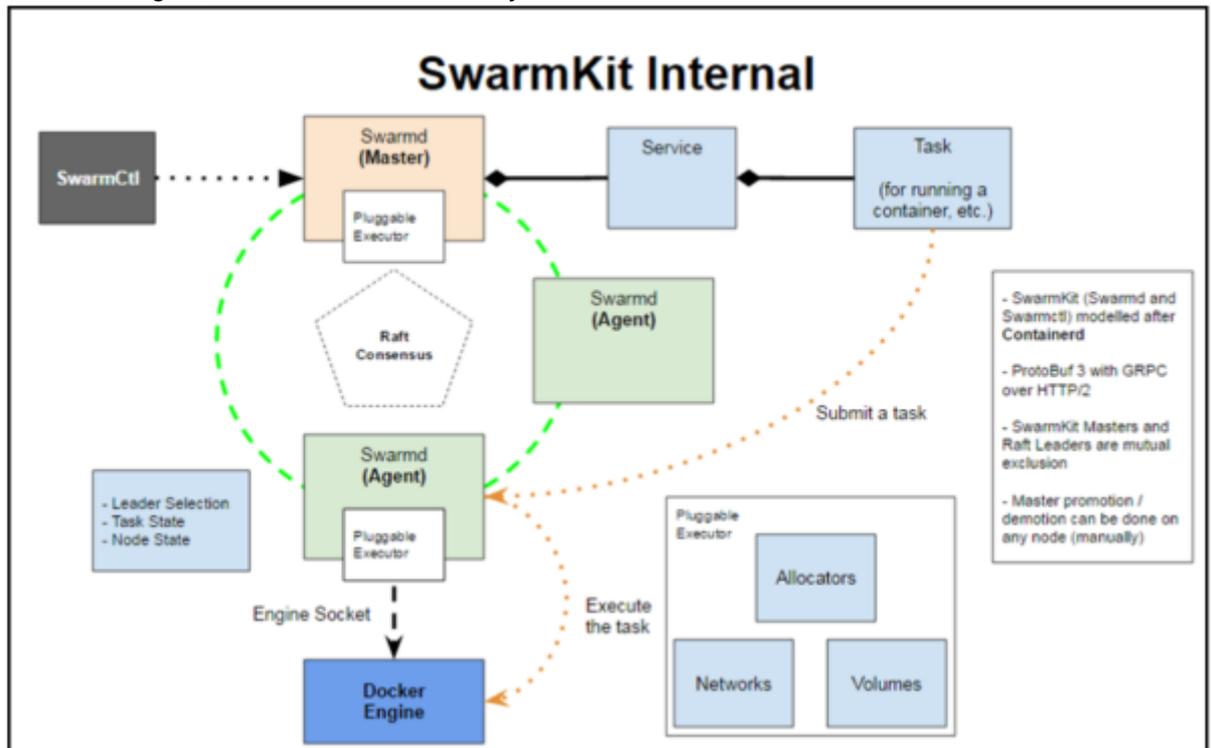
- `docker run swarm list etcd://$(docker-machine ip etcdm):2379`

chap3: docker swarm mode

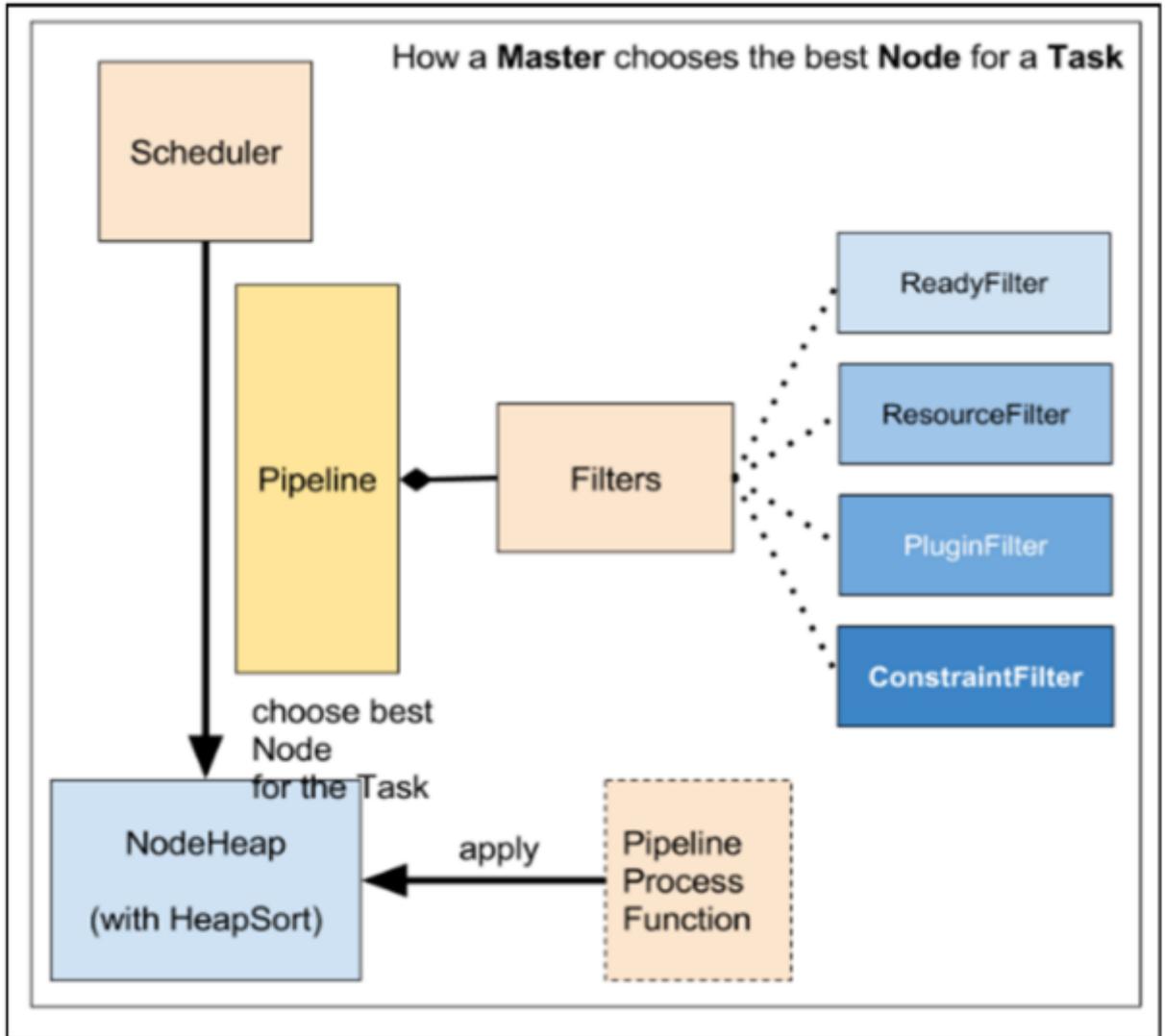
swarmkit

- «toolkit for orchestrating distributed systems at any scale. It includes primitives for node discovery, raft-based consensus, task scheduling, and more» - Docker team at DockerCon16
- Los cluster swarm está compuesto de nodos activos, que pueden actuar como managers o workers.
 - los managers, coordinados via Etcd (raft), elegidos entre todos, son responsables de reservar recursos, orquestrar servicios y repartir tareas a lo largo del cluster
 - los workers ejecutan las tareas.

- Los servicios que se lanzan al cluster se convierten en tareas cuando llegan al worker
- Los servicios no tienen que ser contenedores. La intención de swarmkit es la de orquestrar cualquier objeto.
- arquitectura:
 - número impar de nodos manager (evitar split-brain en las elecciones)
 - soporta cualquier tamaño de cluster de servicios
 - managers y workers
 - cualquier número de workers.
 - en los managers, los servicios se definen y se balancean.



- elección del mejor nodo para una tarea (**scheduling**)



-
- swarmd
 - usado para masters y slaves
 - swarmctl
 - docker run -it fsoppelsa/swarmkit swarmd -help
 - docker run -it fsoppelsa/swarmkit swarmctl -help
 - cliente para operar con el cluster swarmkit
- swarm mode (pag 88)

From: <https://miguelangel.torresegea.es/wiki/> - miguel angel torres egea

Permanent link: <https://miguelangel.torresegea.es/wiki/info:libros:swarm?rev=1638877162>

Last update: 07/12/2021 03:39

