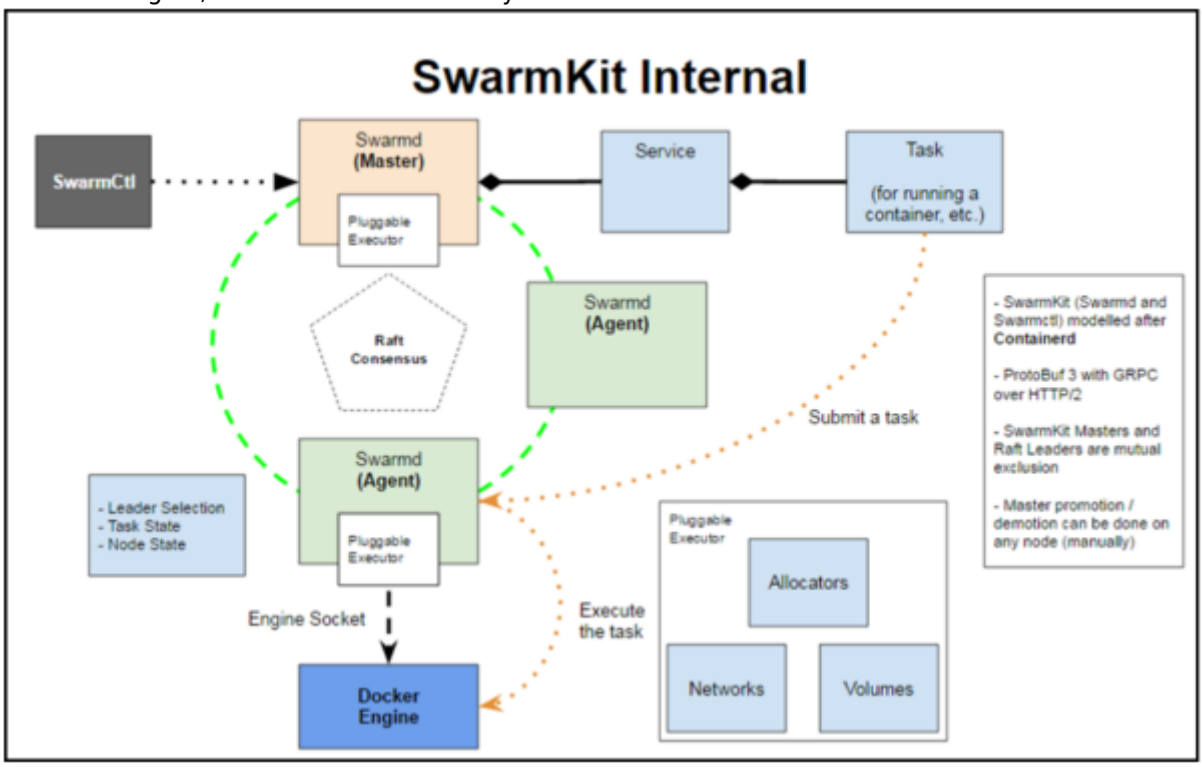
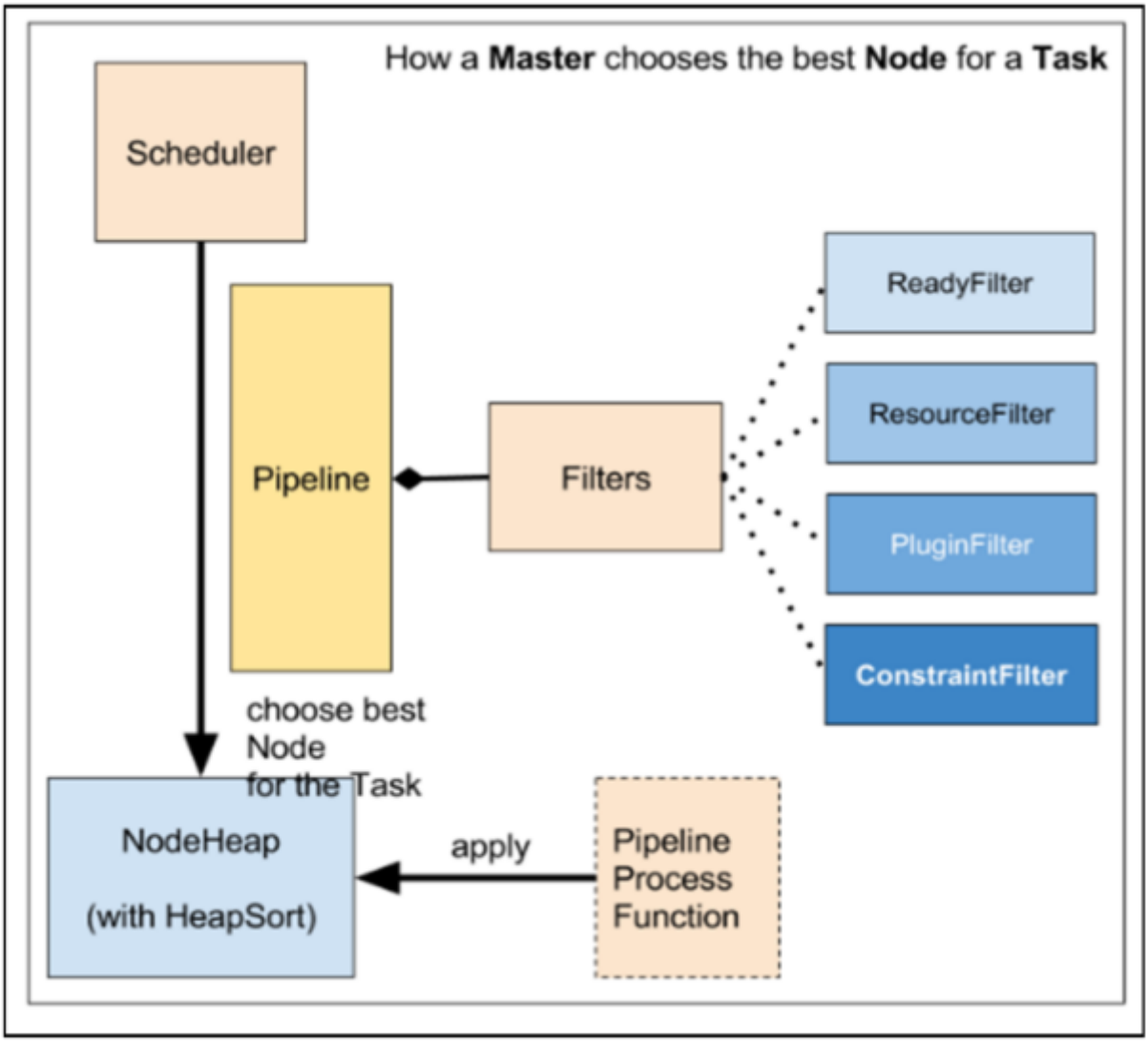
[native docker clustering with swarm] Meeting docker swarm mode

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 porque ser contenedores. La intención del 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.
  - 

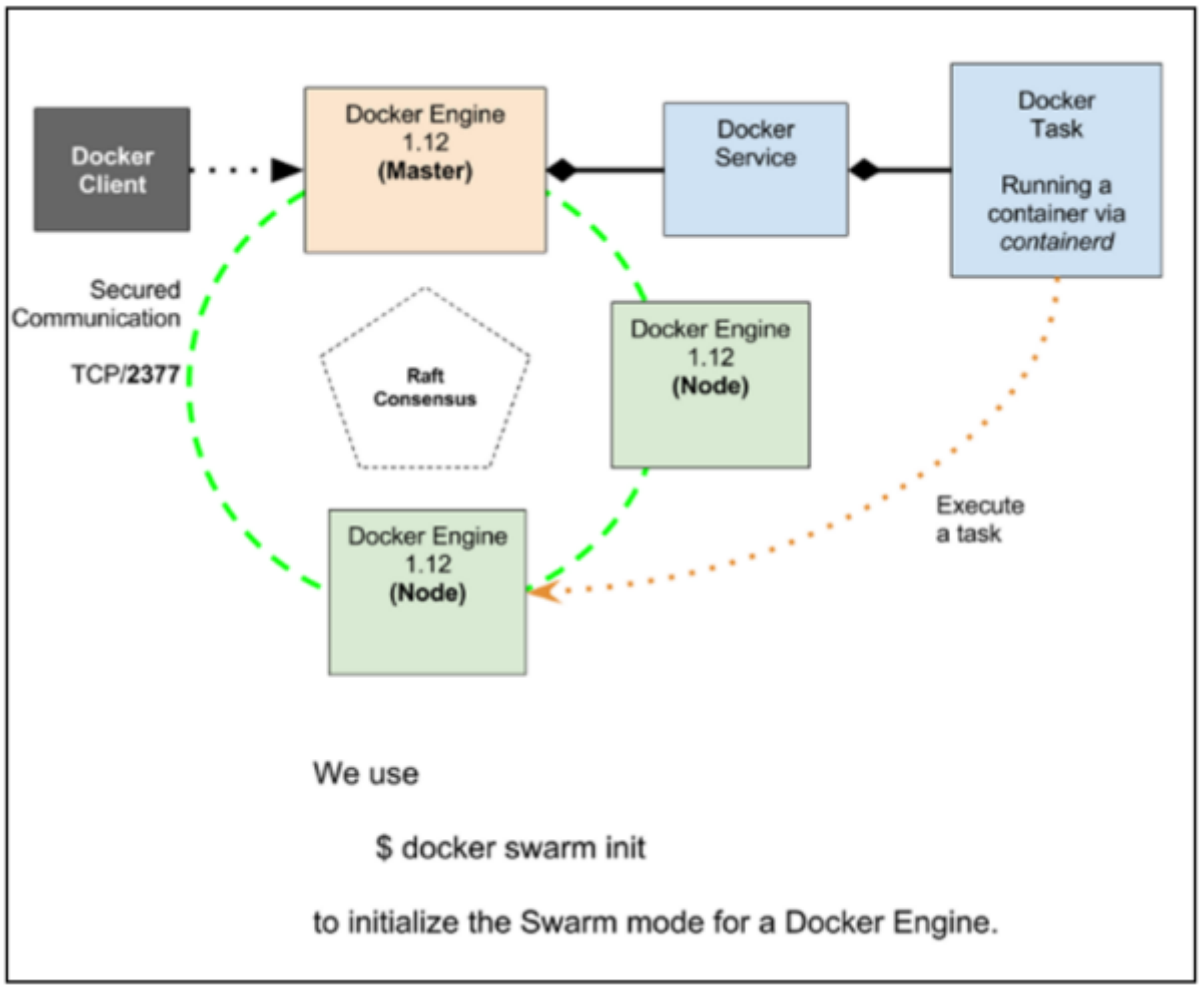
- elección del mejor nodo para una tarea (**scheduling**)



- 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

- > 1.12
- integrado en docker (al contrario que swarmkit). Comandos:
  - **swarm**: `docker swarm init`
  - **nodes**: `docker node ls`
  - **service**: `docker service tasks`
  - **task**



- 3 maneras de orquestrar (evolución)
  - swarm v1
  - swarmkit
  - swarm mode

comparativa

swarm v1	**swarm mode**
docker 1.8	docker 1.12
container	docker engine
servicio discovery externo (Consul, Etcd, Zookeeper)	Etcd integrado
no securizado por defecto	securizado por defecto
no replica, no scaling	si replica, si scaling
no existen los conceptos de servicio y tareas	servicios, tareas y balanceo de carga de serie
no existen servicios de networking adicionales	lleva integrado VxLAN (mesh networking)

swarmkit	**swarm mode**
binarios (swarmd, swarmctl)	integrados en docker
son tareas genéricas	son tareas en contenedores
incluye servicios y tareas	incluye servicios y tareas
no balanceo de carga, no VxLAN	incluye balanceo de carga y VxLAN

zoom in

- integración de los comandos swarm en `docker`

docker swarm

- **init**:
  - inicializa el swarm
  - crea manager en el host actual
  - genera secreto (token) para los workers
- **join**:
  - usado por los workers para añadirse al cluster.
  - debe especificar el token y lista de IP:PORT de los managers
- **join-token**:
  - maneja los join-tokens (el de los managers y el de los workers)
  - imprime el comando necesario para el **join**
    - `docker swarm join-token { worker | manager }`
- **update**:
  - actualiza valores del cluster (por ejemplo, una nueva URL de certificado)
- **leave**:
  - el nodo actual deja el cluster
  - `–force` si hay algún incoveniente
  - manager → worker → leave

docker node

- para gestionar los nodos del cluster swarm.
- se ha de lanzar desde un manager
- **demote/promote**:
  - demote: pasa un nodo de manager a worker
  - promote: pasa un nodo de worker a manager
- **inspect**:
  - equivalente a `docker info`
  - muestra información del nodo
- **ls**:
  - lista de nodos conectados al cluster
- **rm**:
  - intenta desconectar un worker (hacer **demote** si es un **manager**)
- **ps**:
  - lista las tareas corriendo en un nodo especificado
- **update**:
  - permite cambiar valores del nodo

docker service

- manejar los servicios en el cluster
- create: crea un nuevo servicio
- inspect: muestra información detallada del servicio(s)
- ps: lista las tareas de un servicio
- ls: lista los servicios
- rm: elimina un servicio
- scale: scale servicios
- update: actualiza datos del servicio

docker stack

- introducido en 1.12

- conjuntos de servicios (aka docker-compose)
- JSON
- DABΞDistributed Application Bundle

Etcd's raft integrador

- integrado, no necesita de terceros
- CoreOS Etcd Raft library
- DNS y balanceo de carga

Balanceo de carga y DNS

- cada servicio tiene un único nombre DNS
- los balanceadores corriendo contenedores usan el DNS interno
- por cada servicio con la etiqueta **--name <SERVICE>**, cada contenedor sera capaz de resolver la IP del servicio a través de ese nombre (intercomunicación)
- posibilidad de publicar puertos de servicio al balanceador de carga externo → **--publish-add**
- del 30000 al 32767 (externamente)
- internamente, se usa **iptables** y **IPVS** para realizar filtrado de paquetes, reenvio y balanceo de carga.
- `docker service update --port-add 80 nginx-service`
  - crea un mapeo entre el puerto 30000 en cada uno de los nodos del cluster contra los contenedores *nginx* en el puerto 80. Si conectas <IP_NODO>:30000 aparecerá la página de presentación de nginx
  - esto se consigue gracias a la red creada por **ingress overlay** swarm. Cualquier nodo del cluster resuelve a través de la VIP (VirtualIP) y el DNS interno. Cada nodo implementa el balanceo de carga a nivel de kernel, especificamente en el *namespaces*, añadiendo una regla MARK en la cadena OUTPUT en la red:

```
Chain OUTPUT (policy ACCEPT 21 packets, 1634 bytes)
 pkts bytes target     prot opt in     out     source               destination
    4   336 MARK       all  --  *      *       0.0.0.0/0            10.255.0.7           MARK set 0x101
```