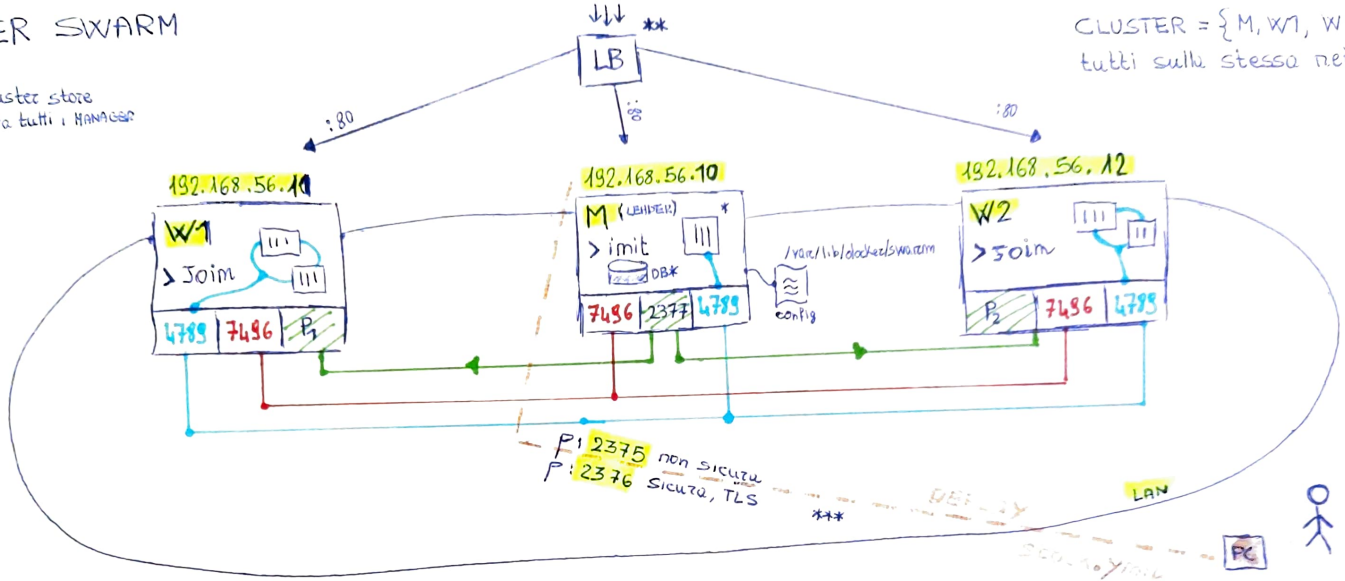
DOCKER SWARM

DB*: The cluster store condivise tra tutti i MANAGER

CLUSTER = {M, W1, W2}
tutti sulla stessa rete LAN



PORTE

MANAGER (2377) ⟷ WORKERS (dinamiche)
tcp
P_1, P_2 scelte dal Kernel in base alla disponibilità.

Comunicazione di controllo.

Ogni Manager ha un ETCD db con la configurazione e lo stato del cluster.

MANAGER ⟷ (7946, tcp/udp) ⟷ WORKERS
Comunicazione di controllo/gossip. Configurazione/Gestione del cluster

MANAGER ⟷ (4789, udp) ⟷ WORKERS
Gestione del traffico di rete overlay. Consente ai container di comunicare su nodi diversi, come se fossero sulla stessa rete locale.

*
I manager possono eseguire container. Sm PRO è preferibile che i manager non eseguano container, di modo che possano dedicarsi esclusivamente alle attività del cluster.
Per evitare che eseguano container (--availability drain)

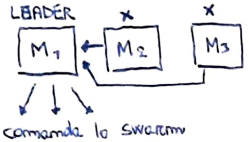
** INGRESS ROUTING MESH
Distribuzione del traffico in ingresso tra tutti i nodi.
Quando arriva una richiesta a un nodo, il nodo può gestire la richieste anche se il container di quel servizio non è in esecuzione su quel nodo.
Viene redirectto attraverso la rete di overlay al nodo che sta eseguendo quel servizio.
Tutti i nodi sono in grado di accettare richieste in ingresso per qualsiosi servizio pubblicato grazie a questo sistema.
Per esempio se pubblico un servizio sulla porta 80 Docker SWARM fa in modo che ogni nodo del cluster sia in ascolto su quella porta.

*** DEPLOY
Solo sul nodo manager sulla porta 2375 o 2376
Es:
> docker -H tcp://192.168.56.10 per connettersi a un host remoto.

****
Anche se ho più manager, solo uno alla volta è attivo, il cosiddetto LEADER. Se un manager non-attivo riceve un comando, lo inoltra al LEADER.

SWARM HA **HIGH AVAILABILITY**

- Se un nodo fallisce lo swarm non cade.



- solo un leader alla volta.
- Se i MANAGER non attivi riceveremo comandi li inoltreremo al LEADER.

Per evitare la split-brain il numero giusto di Manager da usare è 3 o 5.
Può succedere se metto i Manager su reti diverse e per qualche motivo non comunicano più.

- Restart di un vecchio manager può mettere in crisi il cluster.
  Si può implementare un meccanismo di "autoLock" di sicurezza (--autoLock = true)

- Quando parte non può entrare subito.
- Faccio tutti i controlli
- Poi per farlo accedere faccio un "unlock"

- Il compose.yml del SINGLE-ENGINE MODE ragiona a CONTAINER.
  Il stack.yml del SWARM MODE ragiona a SERVICE.

NB: > docker service inspect --pretty [NOME]
Per vedere tutto il config del servizio

compose.yml ⇝ Keyword: SERVICE ⟶ 1 CONTAINER

stack.yml ⇝ Keyword: SERVICE ⟶ n CONTAINER (capacità di replica)
e bilancia le repliche sui vari nodi del cluster

- Tutti i servizi sono continuamente monitorati da un Job in background che si chiama Reconciliation Loop che compara il

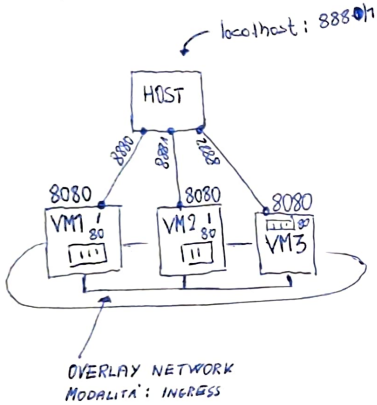
DESIRED STATE **vs** OBSERVED STATE

e agisce di conseguenza per cercare di mantenere questi due STATE allineati.

- Non è necessario creare una rete di OVERLAY per ogni servizio, posso usare la stessa per più servizi, che permette ai vari servizi di comunicare tra loro.
  La faccio solo se voglio isolare il traffico (network create -d overlay)
- Il traffico in ingresso viene bilanciato in automatico dallo SWARM sulle varie repliche di un servizio

MY LOCAL SWARM

Esempio con porta 8080:80 (nginx)



Tutte le VM devono essere in ascolto sulla 8080 mappata all'host.
Quando chiedo Host 8880/8881/8882 di fatto viene effettuata una richiesta a una qualsiasi macchina del cluster sulla porta 8080, che dunque deve essere aperta.
Dato che ho replica=1 il container può trovarsi su una qualsiasi delle 3, ci penserà poi il servizio di discovering a farlo rispondere. cercando sulle varie 8080 del cluster che sono aperte.

- NB

- Quando ho creato una rete di overlay, su quella rete circolano due tipologie di informazioni:
  "DATA-PLANE": non criptato. (application traffic)
  "CONTROL-PLANE": criptata. (management traffic)

Se voglio criptare anche i dati posso specificarlo alla creazione della rete: (driver: overlay, driver-opts: encrypted: true)
considerando potenziali overhead di performance.

Dato che la comunicazione è criptata posso vedere sui nodi la validità dei certificati

"> sudo openssl x509 -in /var/lib/docker/swarm/certificates/swarm-node.crt -text "

Anche con "> docker system info" posso vedere la CA configuration.