Overview

The cluster management and orchestration features embedded in the Docker Engine are built using SwarmKit. Engines participating in a cluster are running in swarm mode. You enable swarm mode for the Engine by either initializing a swarm or joining an existing swarm.
A swarm is a cluster of Docker Engines where you deploy services. The Docker Engine CLI includes the commands for swarm management, such as adding and removing nodes. The CLI also includes the commands you need to deploy services to the swarm and manage service orchestration.
Worker nodes receive and execute tasks dispatched from manager nodes. By default manager nodes are also worker nodes, but you can configure managers to be manager-only nodes. The agent notifies the manager node of the current state of its assigned tasks so the manager can maintain the desired state.

Services and tasks

- A service is the definition of the tasks to execute on the worker nodes. It is the central structure of the swarm system and the primary root of user interaction with the swarm.
- When you create a service, you specify which container image to use and which commands to execute inside running containers.
- In the replicated services model, the swarm manager distributes a specific number of replica tasks among the nodes based upon the scale you set in the desired state.
- For **global services**, the swarm runs one task for the service on every available node in the cluster.
- A task carries a Docker container and the commands to run inside the container. It is the atomic scheduling unit of swarm. Manager nodes assign tasks to worker nodes according to the number of replicas set in the service scale. Once a task is assigned to a node, it cannot move to another node. It can only run on the assigned node or fail.

Swarm Init

- switches the current node into swarm mode.
- creates a swarm named default.
- designates the current node as a leader manager node for the swarm.
- names the node with the machine hostname.
- configures the manager to listen on an active network interface on port 2377.
- sets the current node to Active availability, meaning it can receive tasks from the scheduler.
- starts an internal distributed data store for Engines participating in the swarm to maintain a consistent view of the swarm and all services running on it.
- by default, generates a self-signed root CA for the swarm.
- by default, generates tokens for worker and manager nodes to join the swarm.
- creates an overlay network named ingress for publishing service ports external to the swarm.

Configure the advertise address

- Manager nodes use an advertise address to allow other nodes in the swarm access to the Swarmkit API and overlay networking. The other nodes on the swarm must be able to access the manager node on its advertise address IP address.
- If you don't specify an advertise address, Docker checks if the system has a single IP address. If so, Docker uses the IP address with with the listening port 2377 by default. If the system has multiple IP addresses, you must specify the correct --advertise-addr to enable inter-manager communication and overlay networking:

  $ docker swarm init --advertise-addr <MANAGER-IP>

- You must also specify the --advertise-addr if the address where other nodes reach the first manager node is not the same address the manager sees as its own. For instance, in a cloud setup that spans different regions, hosts have both internal addresses for access within the region and external addresses that you use for access from outside that region. In this case, specify the external address with --advertise-addr so that the node can propogate that information to other nodes that subsequently connect to it.

Join Token Rotation
We recommend that you rotate the join tokens in the following circumstances:

- If a token was checked-in by accident into a version control system, group chat or accidentally printed to your logs.
- If you suspect a node has been compromised.
- If you wish to guarantee that no new nodes can join the swarm

$ docker swarm join-token --rotate worker

Nodes

The **AVAILABILITY** column shows whether or not the scheduler can assign tasks to the node:

Active means that the scheduler can assign tasks to a node.
Pause means the scheduler doesn't assign new tasks to the node, but existing tasks remain running.
Drain means the scheduler doesn't assign new tasks to the node. The scheduler shuts down any existing tasks and schedules them on an available node.

The **MANAGER** STATUS column shows node participation in the Raft consensus:
No value indicates a worker node that does not participate in swarm management.
Leader means the node is the primary manager node that makes all swarm management and orchestration decisions for the swarm.
Reachable means the node is a manager node is participating in the Raft consensus. If the leader node becomes unavailable, the node is eligible for election as the new leader.
Unavailable means the node is a manager that is not able to communicate with other managers. If a manager node becomes unavailable, you should either join a new manager node to the swarm or promote a worker node to be a manager.

I**nspect a node**
You can run docker node inspect <NODE-ID> on a manager node to view the details for an individual node. The output defaults to JSON format, but you can pass the --pretty flag to print the results in human-readable format

List nodes	docker node ls
List tasks running on one or more nodes, defaults to current node	docker node ps docker node ps <NODE ID>
To inspect a service	docker inspect server <service name>
To see list the tasks of one or more services	docker service ps <service name> e.g. docker service ps raghu-nginx
Leave swarm	docker swarm leave docker swarm leave --force
From manager, Inspect an individual node	docker node inspect <NODE-ID> docker node inspect <Node-ID> --pretty # to print node info. In human readable format docker node inspect self --pretty
To promote a node to manager	docker node promote node-1 docker node promote node-3 node-2
To demote a manager to worker level	docker node demote node-3
Inspect nodes	docker node inspect <node-name> docker node inspect self docker node inspect <node-name> --pretty
To make a node leave the swarm. This command needs to be run on a node	docker swarm leave When a node leaves the swarm, the Docker Engine stops running in swarm mode. The orchestrator no longer schedules tasks to the node
Label a node	docker node update --label-add ws-mgr zoxaxafzmjrpqsbhmngl4sbik
To see the tasks running on a service	docker service tasks website
Change the availability of node to **DRAINED**	docker node update --availability drain node-1 #drain a manager node so that only performs swarm management tasks and is unavailable for task assignment
to return the drained node to an active state:	docker node update --availability active <NODE-ID>
To remove a node from swarm	docker node rm node-2 # you can only remove once the node leaves the swarm

Commands

To create a swarm	docker swarm init
To see the nodes in the swarm	docker node ls
To generate a worker token	docker swarm join-token worker #Use the --rotate flag to generate a new join token for the specified role: docker swarm join-token --rotate worker #After using --rotate, only the new token will be valid for joining with the specified role.
To generate a manager token	docker swarm join-token manager
To print the join token	docker swarm join-token -q worker docker swarm join-token -q manager
Restore a swarm that has lost consensus - to reinitialize	docker swarm init --force-new-cluster
Updating a running service	
Update replicas	docker service update --replicas=5 raghu-nginx #docker service scale raghu-nginx=8
Rolling back to previous state	docker service rollback raghu-nginx

First Swarm

Docker swarm init	Initializes a swarm
docker swarm join-token manager	To add a manager to the swarm
docker swarm join --token	To add a worker

Swarm Mode: Docker's native docker clustering mode.
Farm mode, nodes have two roles..
Manager node: Used to orchestrate swarm cluster
Worker node: Only runs the containers
Recommendation is to use odd number of managers as cluster can tolerate (n-1)/2 failures.
Managers among themselves elect a leader
A worker can be promoted to a manager role
A manager can be demoted to a worker role

Create 3 machines	Ubuntu 16.04 Make sure Docker-engine is installed in all the three machines.
Make one a manager	docker swarm init
	docker node ls #to check the swarm status
Run the swarm join command in the other two machines	docker swarm join --token SWMTKN-1-2q6ksz9l0if9uohijuqzq3b4bto7w7w7bp3xmiooe2m22dg7m4-dcw9cekp8olfgzgdeqbpqclh3 192.168.65.2:2377
	docker node ls
Create a service in the manager	docker service create --name raghu-nginx --replicas 3 --publish 8080:80 nginx
Check the service	docker service ls
Install docker visualizer in the manager https://github.com/manomarks/docker-swarm-visualizer	docker service create \ --name=vmanager \ --publish=8082:8080/tcp \ --constraint=node.role==manager \ --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \ dockersamples/visualizer
Open the visualizer in browser	e.g. http://localhost:8082 # as per the command given above
	docker service ls # try it on manager and worker(s) also
Scaling the service From manager	docker service scale raghu-nginx=7
	docker service ls # to see the status
Scale down	docker service scale raghu-nginx=5
Which IP address to use?	You can use the IP address of any of these 3 hosts (manager or works')
Inspect the docker network in manager	docker inspect network docker network ls
To leave the current swarm #has to be given from worker	docker swarm leave

Swarm Cluster E.g.

Pre-requisite	Create 4 instances….. Mgr-1 Mgr-2 Worker-1 Worker-2
Install docker-engine in all the 4 VMs	
Initialize swarm in mgr-1	docker swarm init
Create a token for other managers	docker swarm join-token manager # you will get a token something like the below one SWMTKN-1-1dv6b0rqgydz1xet8wxjh6p4n0xcowt08b7g3h599qyhsu2i8s-60qyabbp4v3eu9jxluzwdat6o 10.140.0.3:2377
Log into mgr-2 and make him join as manager to the existing swarm Pick the command generated in the above step and run here	docker swarm join --token SWMTKN-1-1dv6b0rqgydz1xet8wxjh6p4n0xcowt08b7g3h599qyhsu2i8s-60qyabbp4v3eu9jxluzwdat6o 10.140.0.3:2377
Check the status from both the managers	docker node ls #you will see one as leader and other as just a manager
SSH mgr-1 add worker -1 to the Mgr-1	#give the below command from Mgr-1 docker swarm join-token -q worker # gives you the token for worker # ssh into worker-1 machine and then run the command generated in the above step. This will add workder-1 to Mgr-1
SSH mgr-1 add worker -2 to the Mgr-2	#give the below command from Mgr-2 docker swarm join-token -q worker # gives you the token for worker # ssh into worker-2 machine and then run the command generated in the above step. This will add workder-2 to Mgr-2
Create a service from Mgr-1	docker service create --name raghu-nginx --replicas 3 --publish 8080:80 nginx

Create a swarm visual manager	docker service create \
	  --name=vmanager \
	  --publish=8082:8080/tcp \
	  --constraint=node.role==manager \
	  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
	  dockersamples/visualizer

Scale-up the service	docker service scale raghu-nginx=8