Prepare all nodes

NOTE: *For this process we will use Centos7*

First, install required dependencies:

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

Second, add Docker repository:

```
$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Install Docker CE:

```
$ sudo yum install -y docker-ce
```

Enable Docker start on boot and start daemon:

```
$ sudo systemctl enable docker
$ sudo systemctl start docker
```

Initialize first cluster manager

NOTE: *This should be done only on one of manager nodes*

```
$ sudo docker swarm init --advertise-addr=eth1 --data-path-addr=eth1
```

`--advertise-addr` and `--data-path-addr` both set to eth1. This is to make sure that all communication goes through private network only.

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-a9ujgk4iy3f86bs7xecysta2n 10.136.166.159:2377

To add a manager to this swarm, run  'docker swarm join-token manager' and follow the instructions.
```

You can also verify status by listing all current nodes and their status:

```
$ sudo docker node ls
```

The output at this stage will similar to below:

```
ID                            HOSTNAME            STATUS              AVAILABILITY
MANAGER STATUS      ENGINE VERSION
ppp9aa6itx4r3e4u4rsbb6u7g *   manager1            Ready               Active
Leader              18.09.0
```

Add more manager nodes. First, display shell command for joining new managers to cluster:

```
manager1$ sudo docker swarm join-token manager
```

The output will be:

```
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-304rmelcpj5k46baa59einuv8
10.136.166.159:2377
```

On remaining manager nodes use modified version of this command. Just add same arguments as we did on first node –advertise-addr=eth1 –data-path-addr=eth1:

```
$ sudo docker swarm join --advertise-addr=eth1 --data-path-addr=eth1 --token SWMTKN-1-
05r99dbfwrvg
```

The output will be rather simple:

```
This node joined a swarm as a manager.
```

Now, verify current cluster status:

```
$ sudo docker node ls
```

The output should be similar to this:

```
ID                            HOSTNAME            STATUS              AVAILABILITY
MANAGER STATUS      ENGINE VERSION
ppp9aa6itx4r3e4u4rsbb6u7g     manager1            Ready               Active
Leader              18.09.0
xskfh4of12jogw29jklawcy2b     manager2            Ready               Active
Reachable           18.09.0
f81xxoyhwbh745nurw2nur570 *   manager3            Ready               Active
Reachable           18.09.0
```

We have our Docker Swarm HA cluster up and running!

Add worker nodes

NOTE: This part should be done only on designated worker nodes. In this example it is worker1 and worker2.

First, on any of managers run this command to show token for joining cluster as worker:

```
$ sudo docker swarm join-token worker
```

The output will be similar to below:

```
    docker swarm join --token SWMTKN-1-
05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-a9ujgk4iy3f86bs7xecysta2n
10.136.166.220:2377
```

On each worker we will use this command adding same arguments as before to isolate internal comms to private network only:

```
$ sudo docker swarm join --advertise-addr=eth1 --data-path-addr=eth1 --token SWMTKN-1-
05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-a9ujgk4iy3f86bs7xecysta2n
10.136.166.220:2377
```

Veirfy again list of nodes and their status:

```
$ sudo docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY
MANAGER STATUS      ENGINE VERSION
ppp9aa6itx4r3e4u4rsbb6u7g     manager1            Ready               Active
Leader              18.09.0
xskfh4of12jogw29jklawcy2b     manager2            Ready               Active
Reachable           18.09.0
f81xxoyhwbh745nurw2nur570 *   manager3            Ready               Active
Reachable           18.09.0
s6lwqd5nir2u4pva58uy5ryhy     worker1             Ready               Active
18.09.0
n3efneuhnwa57869tox6sdlhv     worker2             Ready               Active
18.09.0
```

Deploy a web interface

There are multiple web interfaces that are available for Docker Swarm HA clusters.

We will use Portainer official documentation to deploy it. First, on any of managers get stack definition file:

```
$ curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o portainer-agent-stack.yml
```

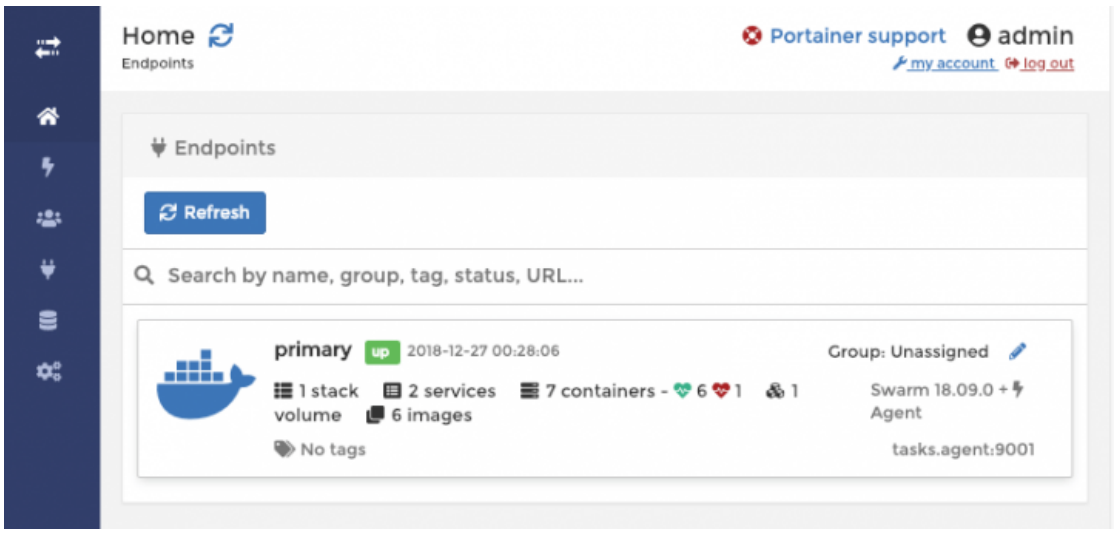
By default Portainer will publish on port 9000, if you want to use different port, simply edit first value in file we just downloaded:

```
      - "9000:9000"
```

Now, deploy the stack to Docker Swarm HA cluster:

```
$ sudo docker stack deploy --compose-file=portainer-agent-stack.yml portainer
```

Give it a few seconds to start everything up and deploy all resources. Once all resources provisioned, you will be able to access web interface in browser on http://manager1_ip_address:9000 (replace IP address and port with your corresponding values). You will be first offered to set password for admin user. After this you will see a new and shiny dashboard:



Congrats! Now you have fully functional Docker Swarm HA cluster with a nice management Web UI.

Final notes

The process to deploy Docker Swarm HA cluster is very simple and straightforward. It can be done in a few simple steps. Anyone going through this process should pay attention to multiple points:

- Calculate your manager node resiliency according to number of nodes you can possibly lose. You should have minimum 3 managers. In this case you can lose at most 1 manager node
- If you lost more than half of manager hosts, your cluster will be not functional anymore. In case of 3 managers this will happen when you lost any 2 of them
- Design High Availability before starting building the system. Think about multiple zones. Label your nodes according to zones
- Isolate managers from workers. Do not run your applications on managers, you have workers exactly for that. Only acceptable application to run on managers is a Web UI
- Minimize attack surface on manager nodes by restricting access to them from Internet. For example, setup VPN to access them. Or use firewall rules to lock down access to them only to trusted source IP addresses

- Make sure control and data plane communication of your cluster happens over private network. If you are building distributed Docker Swarm HA cluster, use SSL encryption for data plane using overlay networks with encryption options
- Learn how to properly add/remove managers to ensure cluster normal operations despite doing changes. Also important to learn how to recover from failures such as loss of manager quorum. All of this described in great details in Swarm guide.