Docker Swarm adalah Sekelompok mesin yang menjalankan Docker dan bergabung ke dalam sebuah cluster dan penjadwalan untuk wadah Docker. Dengan Swarm, administrator dan pengembang TI dapat membangun dan mengelola sekelompok node Docker sebagai sistem virtual tunggal

Swarm adalah sekelompok mesin yang menjalankan docker dan bergabung dalam cluster

Pre-requisites

1. dijalankan di Ubuntu 18.04
2. sudah install Docker dan Docker Machine

Instalasi

1. Install Docker
https://docs.docker.com/install/linux/docker-ce/ubuntu/

2. install docker machine
https://docs.docker.com/machine/install-machine/#installing-machine-directly
versi docker-machine

```
root@ubuntu-X456UQK:~# docker-machine --version
docker-machine version 0.16.0, build 702c267f
root@ubuntu-X456UQK:~#
```

3. setelah docker dan docker machine sudah di install, sekarang akan membuat docker swarm
4. membuat satu docker machine sebagai manager dan dua bertindak sebagai workers

```
root@ubuntu-X456UQK:~# docker-machine create --driver virtualbox manager1
Creating CA: /root/.docker/machine/certs/ca.pem
Creating client certificate: /root/.docker/machine/certs/cert.pem
Running pre-create checks...
Error with pre-create check: "VBoxManage not found. Make sure VirtualBox i
s installed and VBoxManage is in the path"
root@ubuntu-X456UQK:~#
```

5. jika ada error seperti diatas, kita harus install virtualbox menggunakan command line dibawah ini

apt install virtualbox

6. setelah itu lakukan kembali membuat docker machine

docker-machine create –driver virtualbox manager1
docker-machine create –driver virtualbox worker1
docker-machine create –driver virtualbox worker2

```
root@ubuntu-X456UQK:~# docker-machine create --driver virtualbox manager1
Running pre-create checks...
(manager1) Image cache directory does not exist, creating it at /root/.doc
ker/machine/cache...
(manager1) No default Boot2Docker ISO found locally, downloading the lates
t release...
(manager1) Latest release for github.com/boot2docker/boot2docker is v18.09
.9
(manager1) Downloading /root/.docker/machine/cache/boot2docker.iso from ht
tps://github.com/boot2docker/boot2docker/releases/download/v18.09.9/boot2d
ocker.iso...
(manager1) 0%....10%....20%....30%....40%....50%....60%....70%....80%....9
0%....100%
Creating machine...
(manager1) Copying /root/.docker/machine/cache/boot2docker.iso to /root/.d
ocker/machine/machines/manager1/boot2docker.iso...
(manager1) Creating VirtualBox VM...
(manager1) Creating SSH key...
(manager1) Starting the VM...
(manager1) Check network to re-create if needed...
(manager1) Found a new host-only adapter: "vboxnet0"
(manager1) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on t
his virtual machine, run: docker-machine env manager1
root@ubuntu-X456UQK:~#
```

`7. dan setelah itu buat docker machine untuk worker1 dan worker2
8. setelah kita create docker swarm untuk pemastian docker berjalan dengan baik, maka kita dapat melihat docker machine menggunakan perintah berikut

```
root@ubuntu-X456UQK:~# docker-machine ls
NAME       ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
manager1   -        virtualbox   Running   tcp://192.168.99.100:2376           v18.09.9
worker1    -        virtualbox   Running   tcp://192.168.99.101:2376           v18.09.9
worker2    -        virtualbox   Running   tcp://192.168.99.102:2376           v18.09.9
root@ubuntu-X456UQK:~#
```

maka kita telah memiliki 3 machine running

9. setelah itu kita akan connect ke docker-machine yang kita buat mengunakan perintah berikut ini

```
root@ubuntu-X456UQK:~# docker-machine ssh manager1
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@manager1:~$ ip r
```

maka kita telah connect ke docker-machine manager
10. setelah kita connect ke setiap docker machine, maka kita akan inisialisasi docker swarm, dengan cara perintah berikut
pada bagian ini machine manager1 sebagai manager dan
machine worker1 dan machine worker2 sebagai worker

```
docker@manager1:~$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (atfrxtovx6k7pug5hjgm57am0) is now a manager.

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

    docker swarm join --token SWMTKN-1-2oayxkpi9x003hqtm0xv05zp4ieh5l4zzlx8utoc4i6xbo8hcb-7xcsuxor4m2lw86xw5z5tzqa9 192.168.99.100:2377

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

docker@manager1:~$
```

maka manager1 sudah menjadi node
11. untuk melihat dapat menggunakan kode berikut

```
docker@manager1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
atfrxtovx6k7pug5hjgm57am0 *   manager1            Ready               Active              Leader              18.09.9
docker@manager1:~$
```

dan kode tersebut tidak dapat dijalankan diworker dan hanya dapat berjalan di manager

12. setelah itu, kita akan membuat worker1 dan worker2 join sebagai worker
menggunakan perintah berikut ini pada manager 1

```
docker@manager1:~$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2oayxkpi9x003hqtm0xv05zp4ieh5l4zzlx8utoc4i6xbo8hcb-7xcsuxor4m2lw86xw5z5tzqa9 192.168.99.100:2377

docker@manager1:~$
```

kita akan mendapatkan sebuat code atau perintah yang akan kita jalankan di worker 1 dan worker2yang tujuannya untuk menambah worker pada swarm

maka hasilnya akan seperti berikut ini
a. worker 1

```
docker@worker1:~$ docker swarm join --token SWMTKN-1-2oayxkpi9x003hqtm0xv05zp4ieh5l4zzlx8utoc4i6xbo8hcb-7xcsuxor4m2lw86xw5z5tzqa9 192.168.99.100:2377
This node joined a swarm as a worker.
docker@worker1:~$
```

b.worker2

```
docker@worker2:~$ docker swarm join --token SWMTKN-1-2oayxkpi9x003hqtm0xv05zp4ieh5l4zzlx8utoc4i6xbo8hcb-7xcsuxor4m2lw86xw5z5tzqa9 192.168.99.100:2377
This node joined a swarm as a worker.
docker@worker2:~$
```

13untuk memastikan worker sudah menjadi node swarm maka dapat kita lihat dengan menggunakan perintah berikut

```
docker@manager1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
atfrxtovx6k7pug5hjgm57am0 *   manager1            Ready               Active              Leader              18.09.9
uat895837gghy73jzs4hjmg1t     worker1             Ready               Active                                  18.09.9
l3udqct1mzuusec7m54ms1ntw     worker2             Ready               Active                                  18.09.9
docker@manager1:~$
```

14. setelah itu kita akan run container menggunakan docker swarm
dan pada praktek ini kita akan run NodeJs dan Angular pada docker swarm

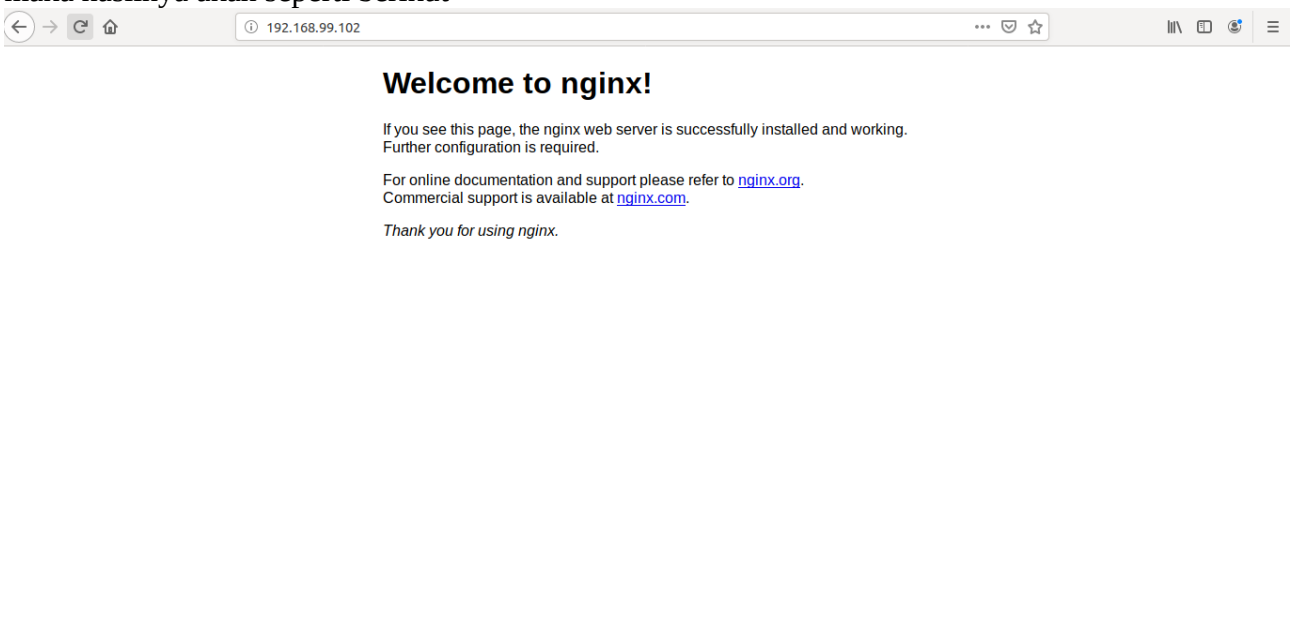
15. untuk pertama kali saya menggunakan Nginx sebagai uji coba
16. pertama kita akan run container nginx di docker swarm, menggunakan kode berikut ini

17. setelah itu kita dapat melihat nginx dapat running di semua node, menggunakan kode berikut

```
docker@manager1:~$ docker service ps web1
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE               ERROR
RTS
m732gisc3qj9        web1.1              nginx:latest        manager1            Running             Running about a minute ago
mdjkidemb7gn        web1.2              nginx:latest        worker1             Running             Running about a minute ago
oe4ymxhyp12w        web1.3              nginx:latest        worker2             Running             Running about a minute ago
docker@manager1:~$
```

digambar diatas menunjukan bahwa ketiga node berjalan container nginx

untuk membuktikan bahwa nginx running di ketiga node, maka kita dapat mengakses ip dari node tersebut
kita akan membuat contoh pada worked1 dengan ip : 192.168.99.102
maka hasilnya akan seperti berikut



akan tampil Web nginx

a. Hello World Nodejs dijalankan menggunakan Docker swarm

1. berikut file nodejs yang saya gunakan file index.js

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(8081, function () {
  console.log('app listening on port 8081!')
})
```

2. file package.json

```
{
  "name": "helloworld",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.15.2"
  }
}
```

3. dan file Dockerfile

```
FROM node:latest
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
CMD node index.js
EXPOSE 8081
```

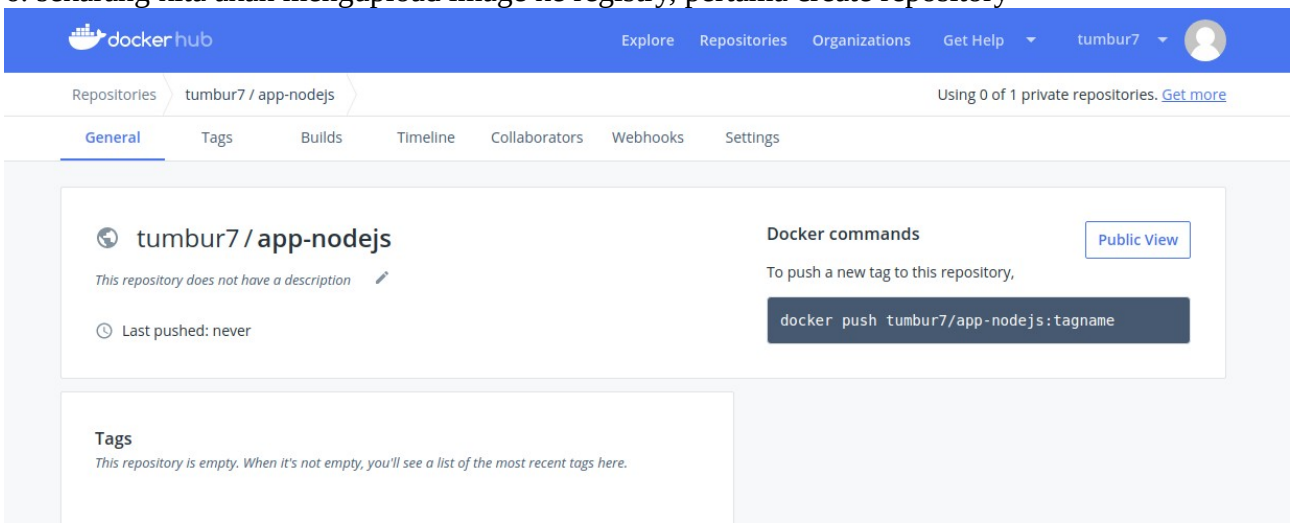
4. setelah itu, kemudian membuat docker image dari Dockerfile yang dibuat dengan cara

```
root@ubuntu-X456UQK:~/node# docker build --tag app-nodejs:1.0 .
Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM node:latest
latest: Pulling from library/node
Digest: sha256:cda22332e2dd46807a14d3268ee3fb298651386ad1f03cfd58772a7dc65f89c7
Status: Downloaded newer image for node:latest
 ---> d8c33ae35f44
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> bab737b39224
Step 3/7 : COPY package.json /app
 ---> Using cache
 ---> 147df090c7bc
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 40620fb0f825
Step 5/7 : COPY . /app
 ---> 82ea263c0a89
Step 6/7 : CMD node index.js
 ---> Running in 8479d37e295c
Removing intermediate container 8479d37e295c
 ---> 61fc3328bcc8
Step 7/7 : EXPOSE 8081
 ---> Running in 1012a6f6cec3
Removing intermediate container 1012a6f6cec3
 ---> ee9ec30afd9b
Successfully built ee9ec30afd9b
Successfully tagged app-nodejs:1.0
root@ubuntu-X456UQK:~/node#
```

5. maka image untuk nodejs telah selesai dengan repostory app-nodejs

```
root@ubuntu-X456UQK:~/node# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
app-nodejs          1.0                 ee9ec30afd9b        About a minute ago   911MB
app-node            2.0                 db3a1289df17        2 hours ago          911MB
<none>              <none>              fc8356b93fb7        2 hours ago          660MB
app-golang          1.0                 b455361fcb5c        3 hours ago          775MB
node                latest              d8c33ae35f44        13 days ago          907MB
golang              1.11.4              dd46c1256829        8 months ago         775MB
root@ubuntu-X456UQK:~/node#
```

6. sekarang kita akan mengupload image ke registry, pertama create repository



maka hasil repository seperti gambar diatas, kemudian kita upload image dengan cara menggunakan kode yang ada pada Docker Commands dengan cara push docker, maka hasilnya akan seperti berikut,

```
root@ubuntu-X456UQK:~/node# docker push tumbur7/app-nodejs:1.0
The push refers to repository [docker.io/tumbur7/app-nodejs]
An image does not exist locally with the tag: tumbur7/app-nodejs
root@ubuntu-X456UQK:~/node#
```

jika hasilnya seperti diatas maka ada error artinya tidak ada image repository tumbur7/app-nodejs:1.0

maka kita akan menambakan image pada local dengan nama tumbur7/app-nodejs:1.0
dengan cara

```
root@ubuntu-X456UQK:~/node# docker tag app-nodejs:1.0 tumbur7/app-nodejs:1.0
root@ubuntu-X456UQK:~/node# docker images
REPOSITORY           TAG                 IMAGE ID            CREATED             SIZE
app-nodejs           1.0                 ee9ec30afd9b        15 minutes ago      911MB
tumbur7/app-nodejs   1.0                 ee9ec30afd9b        15 minutes ago      911MB
app-node             2.0                 db3a1289df17        2 hours ago         911MB
<none>               <none>              fc8356b93fb7        3 hours ago         660MB
app-golang           1.0                 b455361fcb5c        3 hours ago         775MB
node                 latest              d8c33ae35f44        13 days ago         907MB
golang               1.11.4              dd46c1256829        8 months ago        775MB
root@ubuntu-X456UQK:~/node#
```

dari gambar diatas image tumbur7/app-nodejs sudah ada,

maka sekarang kita dapat push image yang ada di lokal ke docker hub atau docker registry
pertama kali kita harus login ke docker hub dengan perintah berikut ini

```
root@ubuntu-X456UQK:~/node# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: tumbur7
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

setelah itu kita push nodejs yang ada di lokal

```
root@ubuntu-X456UQK:~/node# docker push tumbur7/app-nodejs:1.0
The push refers to repository [docker.io/tumbur7/app-nodejs]
303011c8900a: Pushed
d32e63ecda87: Pushed
79eca34f68ce: Pushed
514cc6b51002: Pushed
ff88a6b0c676: Mounted from library/node
99edc82a9595: Mounted from library/node
3bc36b22af5d: Mounted from library/node
72be3b1da83c: Mounted from library/node
409170aec809: Mounted from library/node
2e669e0134f5: Mounted from library/node
8bacec4e3446: Mounted from library/node
26b1991f37bd: Mounted from library/node
55e6b89812f3: Mounted from library/node
1.0: digest: sha256:df5cc58bc6f44488d83ee13124fad8ca65151e04a344540f2a20e52ef9fcbdb9 size: 3047
root@ubuntu-X456UQK:~/node#
```

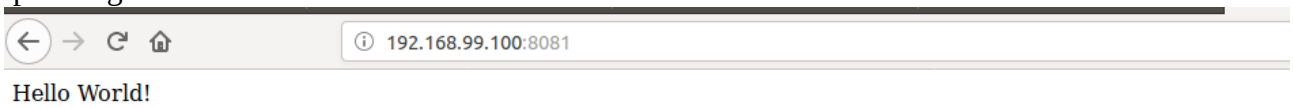
maka docker image yang ada di local sudah ada di docker hub

7. sekarang kita akan run nodejs di docker swarm cluster, menggunakan image register yang kita push ke docker hub
dengan menggunakan perintah berikut ini

```
docker@manager1:~$ docker service create --replicas 3 -p 8081:8081 --name app-nodejs tumbur7/app-nodejs:1.0
nljstu6ls49j76kt3aikh375j
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
docker@manager1:~$
```

maka dapat kita lihat bahwa app-nodejs berjalan dengan baik, untuk meastikan bahwa nodejs berjalan, dapat kita access menggunakan ip docker swarm cluster
contohnya ip worker2 192.168.99.102

ip manager1 : 192.168.99.100



Maka Hello Word pada docker swarm berjalan dengan baik

b. Hello World Angular dijalankan menggunakan Docker swarm