**CD/CI NESS POLL APPLICATION WITH DOCKER SWARM USAGE**

Docker has two modes: “common” and “swarm”.

* **BUILD SWARM TO IMPLEMENT CD-CI**

*CREATED WITH*:

*DOCKER VERSION*

Docker version 17.12.0-ce, build c97c6d6!!!

There is an issue with Docker version 18.03 when creating docker machines. I needed downgrade Docker version to get it working.

WINDOWS VERSION

Microsoft Windows 10 Pro, version 10.0.16299 Build 16299

RUN COMMANDS FROM WINDOWS POWERSHELL AS ADMIN!

1. GET TO THE SWARM MODE: **docker swarm init** - once running this command host pc(better to say linux kit vm running on host pc) becomes swarm manager - is part of swarm cluster too.

You can see something like:

Swarm initialized: current node (vch0rb88br2196p8qpov9sq7l) is now a manager.

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

docker swarm join --token SWMTKN-1-1sx4htgt16fcjiy8q3xllcsxb4a9qcmrnz6d5bgh5h71rhbdgj-ay2jyu07phv62pxdtp259qh59 192.168.65.3:2377

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

NOTE: GET OFF THE SWARM MODE: **docker swarm leave** BE CAREFULL IT ERASES ALL DATA. There is possibility to save swarm state. How? Find out later..

1. CREATE SWARM NODES- VMs
   1. Enable Hyper-V management on your Windows
   2. Create network switch in Hyper-V Manager:

see => <https://docs.docker.com/get-started/part4/#create-a-cluster>

NOTE: For testing purposes create only two nodes to save memory

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" dev-node

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" test-node

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" prod-node

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" jenkins-master

This is successful output:

Running pre-create checks...

Creating machine...

(dev-node) Copying C:\Users\jaros\.docker\machine\cache\boot2docker.iso to C:\Users\jaros\.docker\machine\machines\dev-node\boot2docker.iso...

(dev-node) Creating SSH key...

(dev-node) Creating VM...

(dev-node) Using switch "myswitch"

(dev-node) Creating VHD

(dev-node) Starting VM...

(dev-node) Waiting for host to start...

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 this virtual machine, run: C:\Program Files\Docker\Docker\Resources\bin\docker-machine.exe env dev-node

NOTE: To see all docker machines: **docker-machine ls**

You can see:

NAME ACTIVE DRIVER STATE URL SWARM DOCKER ERRORS

dev-node - hyperv Running tcp://192.168.1.178:2376 v18.03.0-ce

jenkins-master - hyperv Running tcp://192.168.1.182:2376 v18.03.0-ce

NOTE: If they are not running start them via docker command,e.g: **docker-machine start dev-node**

1. INSTUCT VMs

NOTE: You can send commands to your VMs from host using: **docker-machine ssh,** like:

**docker-machine ssh jenkins-master "docker swarm init --advertise-addr <jenkins-master ip>"**

or you can directly ssh to your machine:

**docker-machine ssh jenkins-master**

and run:

**docker swarm init --advertise-addr <jenkins-master ip>**

* 1. *make a jenkins-master = swarm master*, run on jenkins-master:

**docker swarm init --advertise-addr 192.168.1.182:2377**

or

**docker swarm init --advertise-addr 192.168.1.182**

NOTE: 2377 is default port for nodes comunication

You see, something like:

Swarm initialized: current node (6vbkc6hx4h9m5cxl7dpi4i6s1) is now a manager.

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

docker swarm join --token SWMTKN-1-0z1n4htm2dqmgb9eiry8ij15rp10h99oh5gzwfb7ndcl5zgerm-7joqs8vvwjpsvsmronihluz4a 192.168.1.182:2377

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

* 1. *add dev-node to swarm master*:

**docker-machine ssh dev-node "docker swarm join --token SWMTKN-1-0z1n4htm2dqmgb9eiry8ij15rp10h99oh5gzwfb7ndcl5zgerm-7joqs8vvwjpsvsmronihluz4a 192.168.1.182:2377"**

To see all nodes in swarm, you need to run command on swarm master(now it is jenkins-master, not your local pc anymore..):

**docker-machine ssh jenkins-master "docker node ls"**

You will see, something like:

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

1gofveyk7h4ejn6i6guywzzck dev-node Ready Active 18.03.0-ce

nqmvvdiw3hxkise8o52nnnaio jenkins-master Ready Active Leader 18.03.0-cePS

NOTE: To promote node to be swarm manager(run from swarm master):

**docker promote node dev-node**

NOTE: To see info about dev-node, ssh to it and run:

**docker-machine ssh dev-node "docker info"**

or from swarm master(Leader= jenkins-master) run:

**docker inspect dev node**

1. CREATE JENKINS SERVICE ON JENKINS-MASTER NODE:
   1. CREATE YOUR OWN REGISTRY

<https://blog.docker.com/2013/07/how-to-use-your-own-registry/>

<https://docs.docker.com/registry/insecure/>

To share your own registry with all swarm nodes, create insecure registry (suitable for testing purposes) as a container:

docker run -d -p 5000:5000 --name registry registry:2

you can see repositories on http://localhost:5000/v2/\_catalog-> it's empty now

tag the image you want to push to local registry:

docker tag myimage myimage

push it to local registry:

docker push localhost:5000/myimage

you can see under http://localhost:5000/v2/\_catalog:

{"repositories":["myimage"]}

Try pull it:

docker pull myimage

if it’s not working try set up Docker Daemon in Docker Settings to 127.0.0.1/8 to provide insecured! registry

* 1. CREATE YOUR OWN REGISTRY IN SWARM AS A SERVICE

If you want to make your registry secure or be part of swarm cluster, read more on:

<https://docs.docker.com/registry/deploying/#run-the-registry-as-a-service>

<https://docs.docker.com/engine/swarm/secrets/>

<https://docs.docker.com/engine/swarm/secrets/#defining-and-using-secrets-in-compose-files>

1. start jenkins service on JENKINS-MASTER:

create volume: **docker volume create jenkins\_volume**

<https://docs.docker.com/engine/reference/commandline/volume_create/#driver-specific-options>

<https://docs.docker.com/engine/swarm/services/#data-volumes>

* 1. run command with mounted volume to persist jenkins-master data:

**docker service create --name=jenkins-master -p 50000:50000 -p 80:8080 --mount "type=volume,source=jenkins\_volume,target=/var/jenkins\_home" jenkins**

basic command without volume: docker-machine ssh jenkins-master "docker service create --name=jenkins-master -p 50000:50000 -p 80:8080 jenkins"

NOTE:

- port 50000 used by slaves to join master

- created as service so docker swarm will manage the container and reschedule it if needed

- /var/jenkins\_home is bonund to volume now

- mount type=bind

Now Jenkins is running on: http://192.168.1.185:80

NOTE:

To see logs on jenkins-master node:

docker-machine ssh jenkins-master 'docker service logs -f h94phi66batezhbum80s63xx4' -> to see initial pwd for jenkins

To see services (from master): docker service ls; remove them: docker service remove {id}

1. CREATE JENKINS SLAVE AGENT ON DEV-NODE

NOTE: run on dev-node(make it swarm manager: docker promote node dev-node, if you didn't do so yet)

* 1. To implement CD-CI with Docker Swarm you need to first create "secret" that slaves will use to connect to master, run on jenkins-master (it should be global in swarm)

**echo "-master http://192.168.1.185 -password password -username admin"|docker secret create jenkins-v1 –**

(we need to use same user we are logged in jenkins-master service!)

**docker service create --mode=global --name jenkins-swarm-agent -e LABELS=dev-node --mount "type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock" --mount "type=bind,source=/tmp/,target=/tmp/" --secret source=jenkins-v1,target=jenkins-master vipconsult/jenkins-swarm-agent**

* **CREATE PIPELINE PROJECT ON JENKINS-MASTER**