aws ec2 run-instances --image-id ami-08ebab39574d1e681 --key-name LaptopKey --security-group-ids sg-08944ea9f10aa4121 --instance-type t2.micro --placement AvailabilityZone=us-east-1b --subnet-id subnet-0ac37305c1dc6efc5 --tag-specifications ResourceType=instance,Tags=[{Key=Name,Value=DockerSwarm-}] --count 3

docker node ls

docker node promote node

docker node demote node

docker swarm leave

docker swarm leave --force

docker service rm nginxtest

docker service ps

docker service ps --format 'table'

docker service create --name nginx --replicas 3 nginx

1. Create Swarm Cluster & Swarm basic commands. RAFT DB.

2. Understand overlay driver & Create overlay network.

<https://docs.docker.com/v17.09/engine/swarm/networking/#firewall-considerations>

3. Understand services and Create a service.

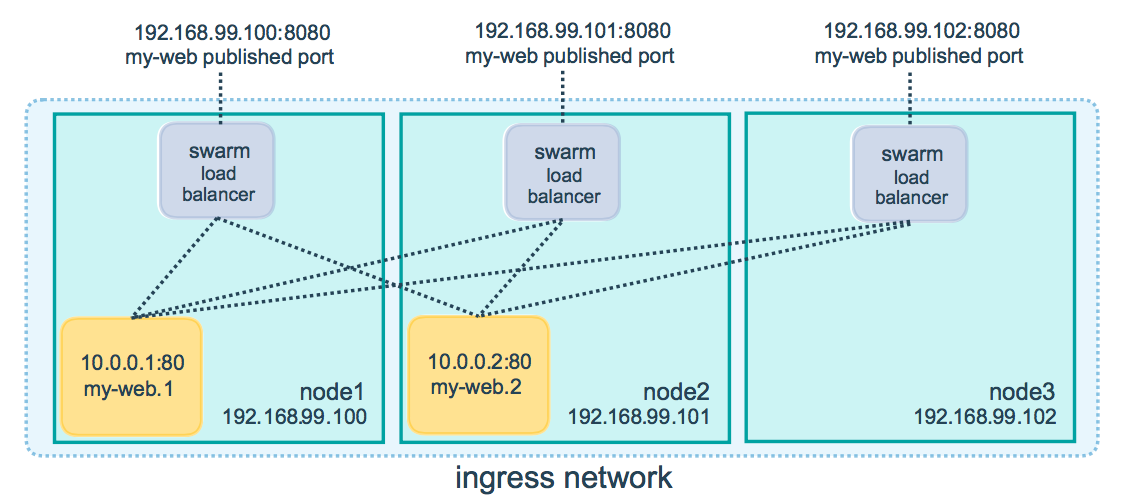
4. Updating & Scaling Swarm service.

5. Routing Mesh Swarm visualizer.

<https://github.com/dockersamples/docker-swarm-visualizer>

docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock

dockersamples/visualizer



6. Container Placement.

* Service Constraint:
* docker service create --constraint=node.role==manager nginx
* docker service create --constraint=node.role!=worker nginx
* node.<label>=custom label
* node.role=inbuult label
* docker node update --label-add=dmz=true node2
* docker service create --constraint=node.labels.dmz==true nginx

7. Swarm Rolling Updates.

docker service create \

--replicas 3 \

--public 8000:80 \

--name nginx \

--update-parallelism 1 \

--update-delay 10s \

sreeharshav/rollingupdate:v1

docker service inspect --pretty nginx

docker service update --image sreeharshav/rollingupdate:v3 nginx

docker service inspect --pretty nginx

docker service update nginx

docker service ps nginx

docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer

Default Labels:

docker node update --label-add mgmt=yes node1

docker node update --label-rm mgmt node1

docker node update --label-add PRODUCTION node5

docker node update --label-rm PRODUCTION node5

docker node update --label-add dev=yes node4

docker node update --label-add dev=yes node5

docker service create --name prodtest --constraint node.labels.prod==yes --replicas 2 sreeharshav/rollingupdate:v3

docker service create --name ptest --constraint node.labels.dev==yes --replicas 4 sreeharshav/rollingupdate:v3

OR

docker node update --label-add ssd=yes node2

docker node update --label-add ssd=yes node3

docker node update --label-add hdd=yes node4

docker node update --label-add hdd=yes node5

docker service create --name SSD-APP --constraint node.labels.ssd==yes --publish 9000:80 --replicas 6 sreeharshav/rollingupdate:v1

docker service create --name HDD-APP --constraint node.labels.hdd==yes --publish 6000:80 --replicas 6 sreeharshav/rollingupdate:v1

docker service create --name HDD-APP --constraint node.labels.hdd==yes --publish 6000:80 --replicas 6 sreeharshav/rollingupdate:v1

docker service create --name TESTING1 --constraint=node.role==manager -p 5000:80 --replicas 3 nginx

docker service create --name TESTING2 -p 5000:80 --constraint=node.role!=manager --replicas 4 sreeharshav/rollingupdate:v3

docker node update --label-add dev=true node2

docker node update --label-add dev=true node3

docker node update --label-add prod=true node4

docker node update --label-add prod=true node5

docker service create --name DEVNGINX --publish 7000:80 --constraint=node.labels.dev==true --replicas 4 nginx

docker service create --name PRODNGINX --publish 5000:80 --constraint=node.labels.prod==true --replicas 4 nginx

## **Swarm Scheduling Strategies**

Docker swarm uses the different scheduling strategies where to distribute the load

a.Spread

b.binpack

c.random

## Spread :

a.This is the default Strategies in docker swarm cluster. b.In this strategy docker swarm distributes the load evenly in all available Worker Nodes. c.If we have three nodes swarm cluster ,Docker swarm distribute the one containers in each node.

## binpack :

a.In this strategy docker swarm distribute the load on the node which is most packed with many container until that node can not run any containers.

## random :

a.In this strategy docker swarm distribute the load randomly on the different nodes.

We can choose the strategy by specifying the --strategy flag while swarm creation.

**Docker Stack Deploy:**

<https://github.com/dockersamples/example-voting-app>

**Docker Traefik Steps:**

<https://blog.programster.org/using-traefik-with-docker-swarm-for-deploying-web-applications>

**DNS Records alias to NLB:**

web1.sreetrainings.xyz

web2.sreetrainings.xyz

web3.sreetrainings.xyz

**Traefik Service Creation:**

docker service create \

--name traefik \

--constraint=node.role==manager \

--publish 80:80 \

--publish 8080:8080 \

--mount type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock \

--network traefik-net \

traefik:v1.6 \

--docker \

--docker.swarmmode \

--docker.domain=traefik \

--docker.watch \

--web

docker service create \

--name prodnginx \

--label traefik.port=80 \

--network traefik-net \

--label traefik.frontend.rule=Host:www.sreetrainings.xyz\

sreeharshav/rollingupdate:v3

docker service create \

--name devnginx \

--label traefik.port=80 \

--network traefik-net \

--label traefik.frontend.rule=Host:dev.sreetrainings.xyz\

sreeharshav/rollingupdate:v1

docker service create \

--name vishwaprod \

--label traefik.port=80 \

--network traefik-net \

--label traefik.frontend.rule=Host:www.vishwadrona.com\

sreeharshav/rollingupdate:v3

docker service create \

--name vishwadev \

--label traefik.port=80 \

--network traefik-net \

--label traefik.frontend.rule=Host:dev.vishwadrona.com\

sreeharshav/rollingupdate:v3

NODE AVAILABILITY:

================

docker node update --availability pause node5 - Dont accept new tasks , runs existing.

docker node update --availability active node5

docker node update --availability drain node5 - Reschedule talsks

Resource Requirements:

====================

Limits: max

Reservations: min

docker service create --reserve-memory 800M --reserve-cpu 1 --name MEMCPUTEST1 --replicas 3 --publish 4000:80 sreeharshav/rollingupdate:v3

docker service create --name LIMITTEST --limit-cpu .25 --limit-memory 100M --replicas 3 --publish 3000:80 sreeharshav/rollingupdate:v3

docker service update LIMITTEST --limit-memory 10M --limit-cpu .10

docker service update LIMITTEST --limit-memory 0 --limit-cpu 0

docker service rm DEVNGINX LIMITTEST MEMCPUTEST1 PRODNGINX

docker service create --name CPULARGE2 --reserve-cpu 80 sreeharshav/rollingupdate:v3

STRESS Dockerfile:

================

FROM debian:latest

RUN apt-get update && apt-get install -y stress \

--no-install-recommends && rm -r /var/lib/apt/lists/\*

CMD ["stress", "--verbose", "--vm", "1", "--vm-bytes", "256M"]

Container Healthchecks:

====================

Docker container healthchecks

<https://blog.sixeyed.com/docker-healthchecks-why-not-to-use-curl-or-iwr/>

HEALTHCHECK CMD curl --fail http://localhost:3000/ || exit 1

docker inspect --format='{{json .State.Health}}' your-container-name

docker service logs NGINX

docker service logs wza88dx6v4pr

docker service logs wza88dx6v4pr --no-task-ids

docker service logs --raw --no-trunc wza88dx6v4pr

docker service logs --raw --no-task-ids --no-trunc wza88dx6v4pr

docker service logs --tail 10 --follow --raw --no-trunc NGINX

==========================================================================

https://docs.docker.com/engine/reference/commandline/events/

=====================================================================

**DOCKER CONFIG:**

docker config create nginxindex1 index.html

docker service create --name nginx1 --config src=nginxindex1 ,target=/usr/share/nginx/html/index.html --publish 8000:80 sreeharshav/rollingupdate:v3

docker service update --config-rm config1 --config-add src=config2,target=/usr/share/nginx/html/index.html nginx1

Docker Swarm Secrets:

===================

<https://blog.ruanbekker.com/blog/2017/11/23/use-docker-secrets-with-mysql-on-docker-swarm/>

Jenkins Docker Swarm Deploy:

=========================

nano /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H unix:// -H tcp://0.0.0.0:2375

systemctl daemon-reload

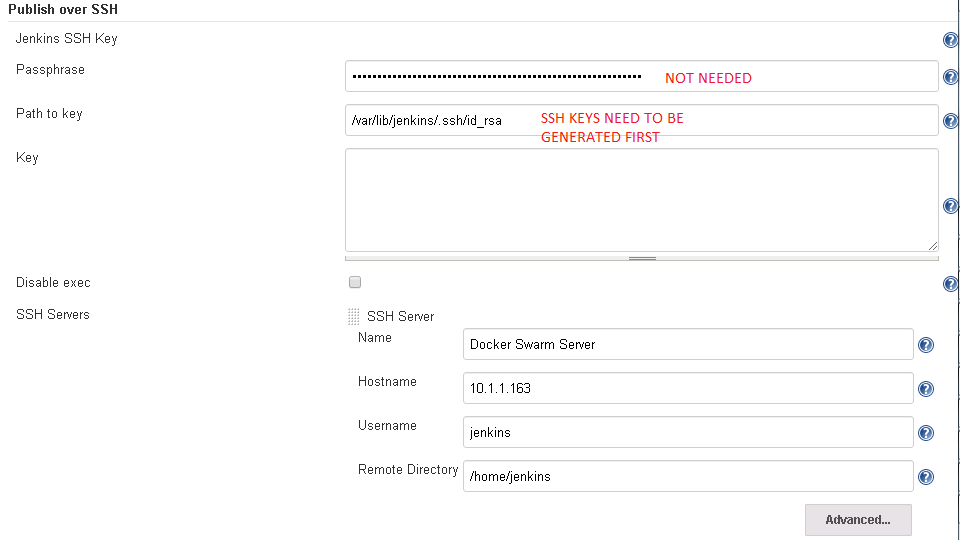
systemctl restart docker

sudo nohup docker daemon -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock &

sudo usermod -a -G root jenkins

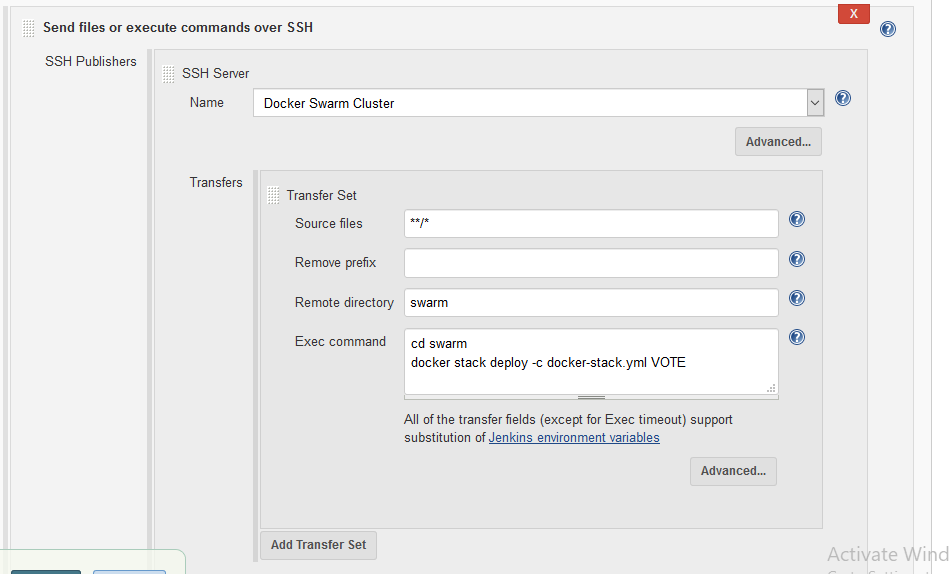
usermod -a -G docker jenkins

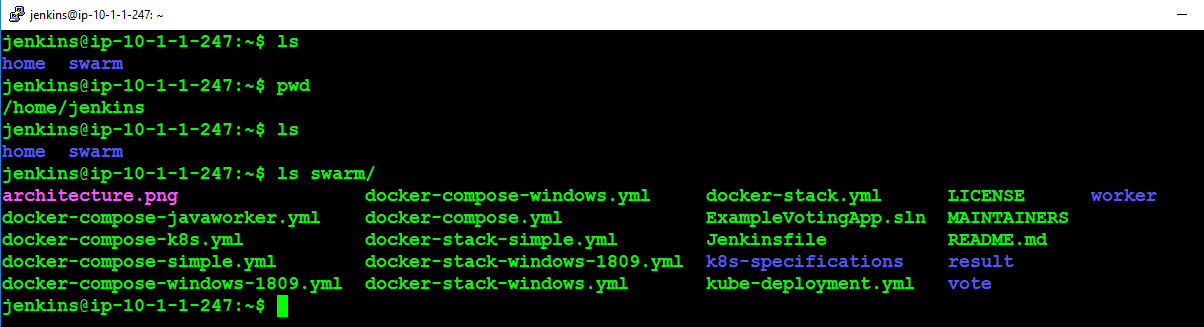
<http://www.littlebigextra.com/automate-service-deployment-docker-swarm-using-jenkins/>



Give /swarm/ in the remote directory.

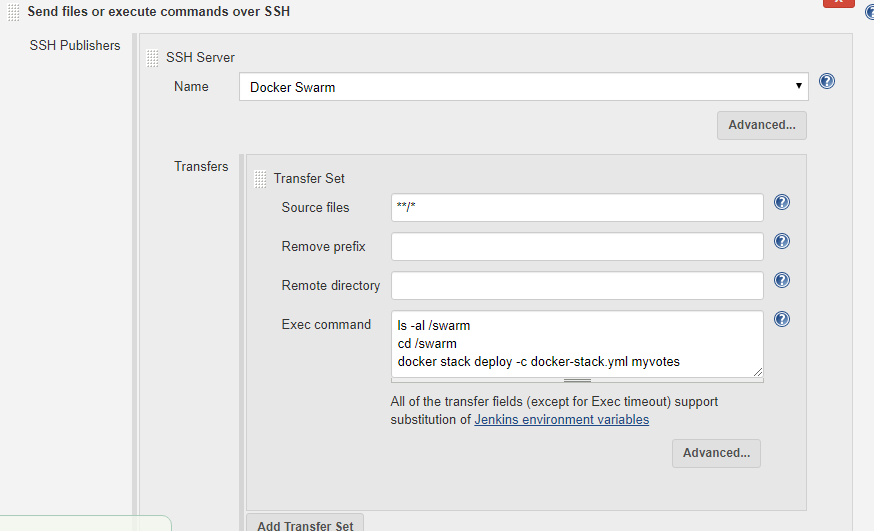
OLD-PIC:





NEW-PIC:

Now Create a new Jenkins Job with the following settings in the PIC:



<https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/>

nano /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H unix:// -H tcp://0.0.0.0:2375

systemctl daemon-reload

systemctl restart docker

sudo nohup docker daemon -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock &

sudo usermod -a -G root jenkins

usermod -a -G docker jenkins