Crear 3 o más vm, para que haya managers y workers, entonces en el manager lanzas el init con la ip del que será manager:

**docker swarm init --advertise-addr 192.168.1.110**

y si quieres sumar más manager o worker entonces, en este manager donde lanzaste el init, lanzas:

**docker swarm join-token manager**

lo que mandará una línea con un token, lo tomas y lo pegas en el otro manager que ocuparás.

para un worker, igual lanzas este comando:

**docker swarm join-token worker**

e igual que en el manager, tomas toda la línea y lo pegas en la máquina que funcionará como worker.

Para tener servicios, ocupas un docker-compose, en este ejemplo, se creó algo como esto:

version: '3.7'

services:

# Spring boot app service.

jenkins:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8000:8080'

volumes:

# Bind volumne host to container

- 'jenkins:/var/jenkins\_home'

networks:

- voteapp

web:

image: nginx

networks:

- voteapp

ports:

- 8080:80

deploy:

#replicas: 2

placement:

constraints: [node.role == worker]

# Open source docker-swarm-visualizer service

visualizersvc:

# Image of docker-swarm-visualizer app

image: dockersamples/visualizer

# This service will run only when appsvc has no failure

depends\_on:

- jenkins

- nginx

deploy:

placement:

# We want to deploy this service only in manager node (suppose we have only one)

constraints:

- 'node.role==manager'

ports:

# Host port : container port

- '5000:8080'

volumes:

# Bind volumne host to container

- '/var/run/docker.sock:/var/run/docker.sock'

networks:

voteapp:

volumes:

jenkins:

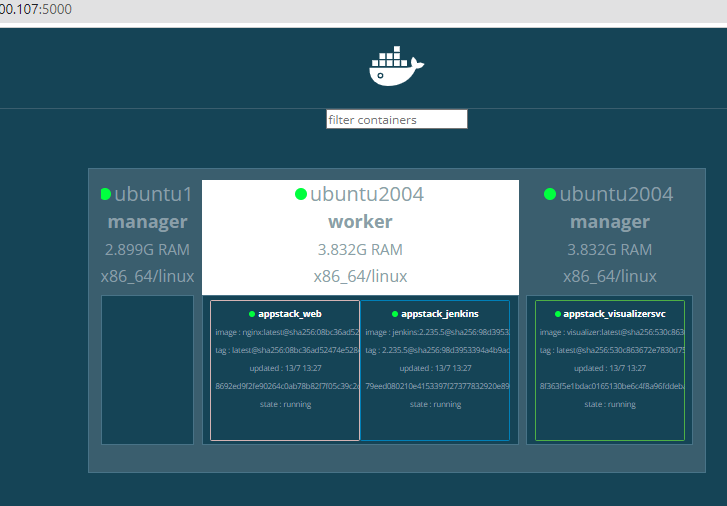
para hacer el deploy, es desde el manager, donde colocas el archivo yml y lanzas:

docker stack deploy --compose-file docker-compose.yml appstack

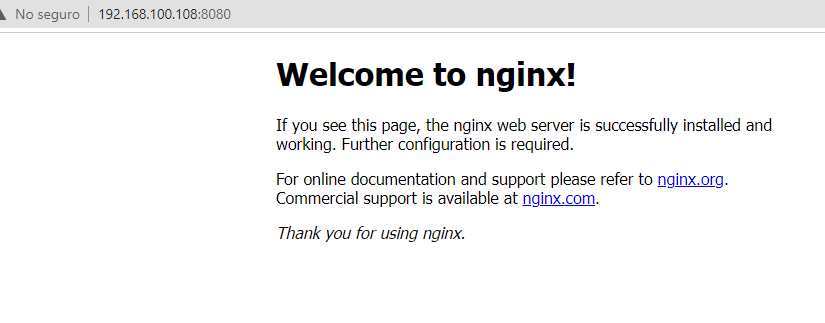
si quieres eliminar tooodo, entonces lanzas:

docker stack rm appstack

En el archivo se lanza un jenkins, un nginx y un visualizer







Con este yml, dimos de alta otro jenkins:

version: '3.7'

services:

# Spring boot app service.

jenkins1:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8001:8080'

volumes:

# Bind volumne host to container

- 'jenkins1:/var/jenkins\_home'

networks:

- voteapp

jenkins2:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8002:8080'

volumes:

# Bind volumne host to container

- 'jenkins2:/var/jenkins\_home'

networks:

- voteapp

web:

image: nginx

networks:

- voteapp

ports:

- 8080:80

deploy:

#replicas: 2

placement:

constraints: [node.role == worker]

# Open source docker-swarm-visualizer service

visualizersvc:

# Image of docker-swarm-visualizer app

image: dockersamples/visualizer

# This service will run only when appsvc has no failure

depends\_on:

- jenkins1

- jenkins2

- nginx

deploy:

placement:

# We want to deploy this service only in manager node (suppose we have only one)

constraints:

- 'node.role==manager'

ports:

# Host port : container port

- '5000:8080'

volumes:

# Bind volumne host to container

- '/var/run/docker.sock:/var/run/docker.sock'

networks:

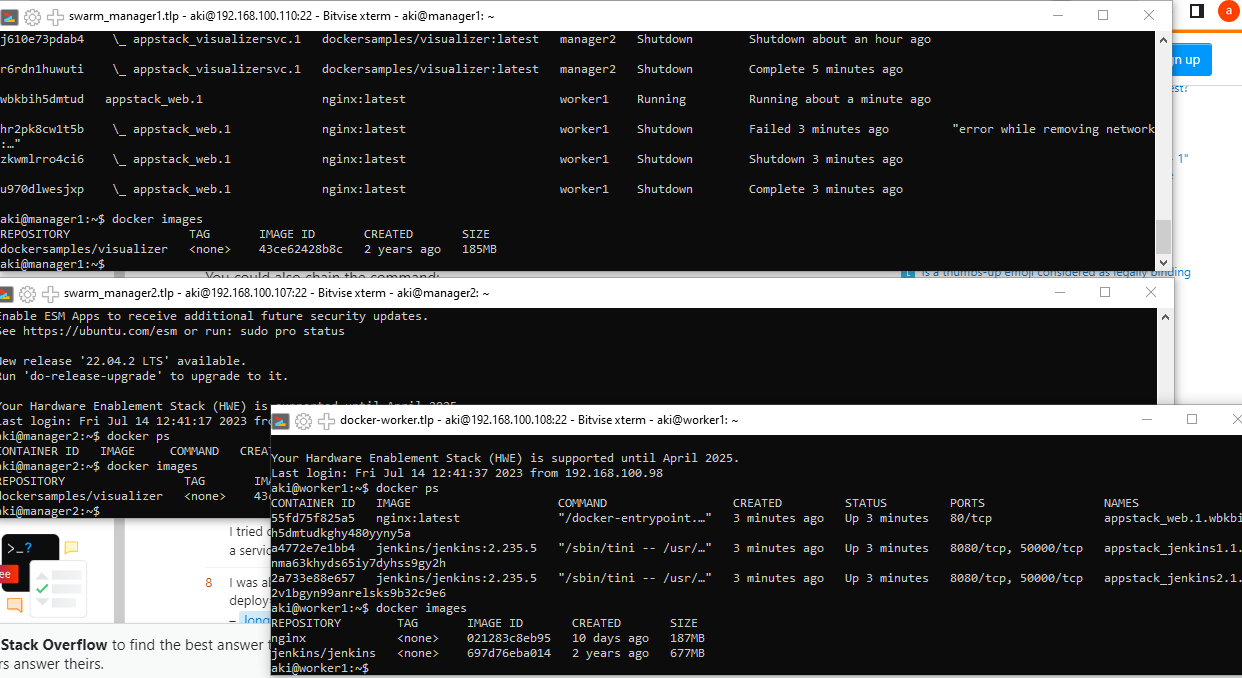
voteapp:

volumes:

jenkins1:

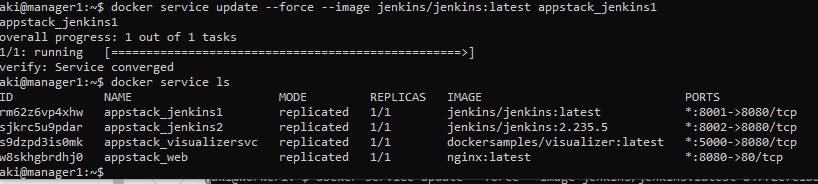
jenkins2:

En este caso, las imágenes de jenkins y nginx viven en el worker, el visualizer en los manager



Para cambiar la imagen:

docker service update --force --image jenkins/jenkins:2.235.5 appstack\_jenkins1

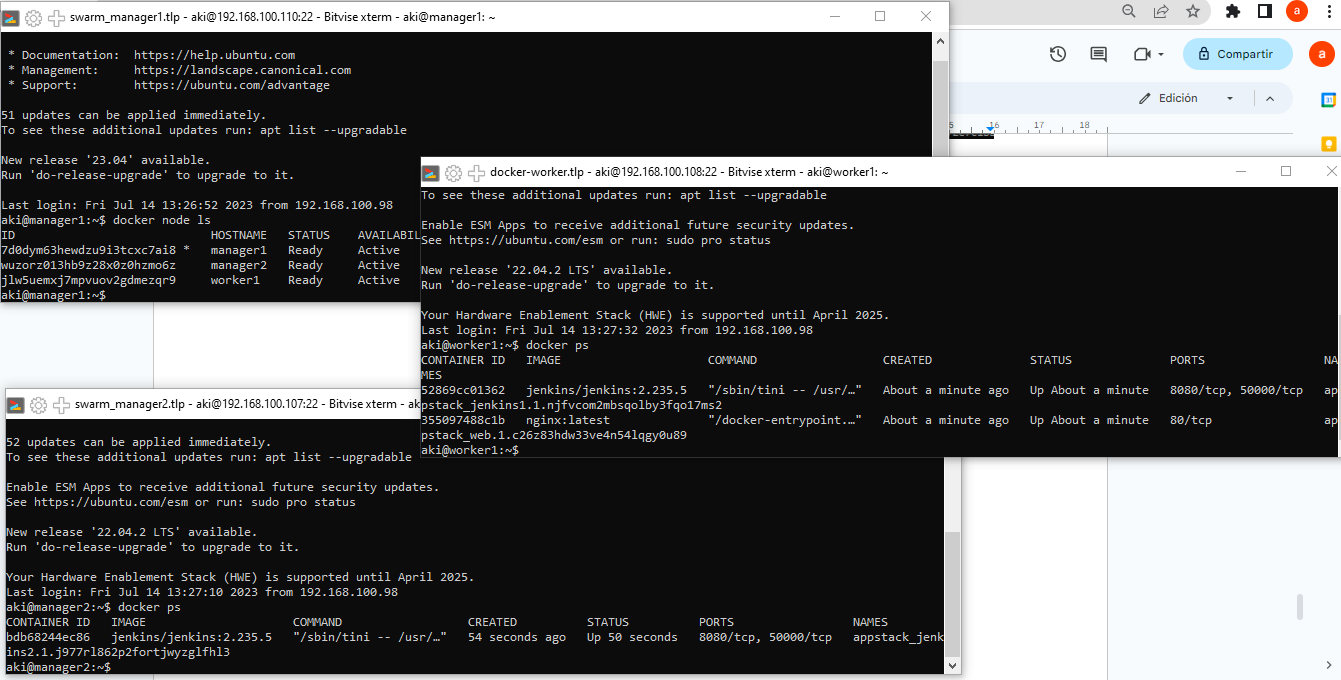


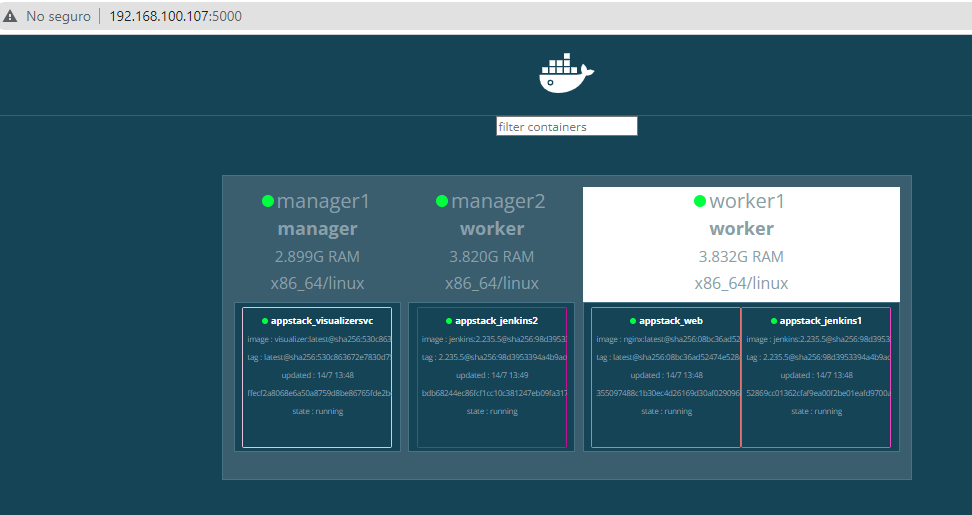
Para regresar el servicio de manera original:

docker service rollback appstack\_jenkins1

Para cambiar de manager a worker:

docker node demote <nodo>





Para cambiar de manager a worker:

docker node promote <nodo>

docker pull jenkins/jenkins

docker save -o archivo.tar <image\_name>

...e importarla con load desde un servidor sin salida:

docker load -i archivo.tar

en este caso la dejé en worker y manager

cambiarle el tag

docker image tag ce05b59420f9 jenkins/jenkins:latest

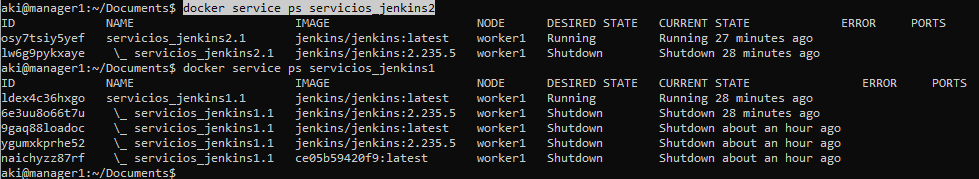
y para cambiar ya la imagen:

docker service update --force --image jenkins/jenkins servicios\_jenkins1

docker service rollback appstack\_jenkins1

Verificar servicios:

docker service ps servicios\_jenkins2



Referencias:

<https://www.w3cschool.cn/doc_docker_1_12/docker_1_12-compose-reference-envvars-index.html?lang=en>

<https://dit.gonzalonazareno.org/gestiona/proyectos/2016-17/Evaristo%20GZ%20-%20Docker%20y%20Docker%20Swarm.pdf>

<https://dccn-docker-swarm.readthedocs.io/en/latest/tutorial/swarm.html>

<https://devm.io/docker/services-and-stacks-in-the-cluster-141399>

<https://blog.container-solutions.com/rolling-updates-with-docker-swarm>

<https://dev.to/greenteabiscuit/tutorial-with-docker-swarm-and-stack-2b0n>

**Traefik:**

Este es el docker compose que contiene 3 jenkins, 1 nginx y 1 visualizer, todos dentro de traefik.

Los servicios corren en el worker, mientras que visualizer y traefik en managers.

Al inicio se crean los jenkins con la versión 2.235.5, pero una vez arriba, se hace el update.

Se tiene que tener la imagen con el tag y en manager y worker.

Para el update se hace desde el manager.

version: "3.8"

services:

traefik:

image: traefik:v2.2

networks:

- ejemplo

command:

- --api=true # enable the management api

- --api.dashboard=true # enable the monitoring dashboard

- --api.insecure=true # allow insecure access to the dashboard

- --providers.docker=true # use docker

- --providers.docker.swarmMode=true # in swarm mode

- --providers.docker.exposedbydefault=false # but don't pick up services automatically

- --entrypoints.web.address=:80 # define `web` entry point listening at port 80

ports:

- 80:80

- 8087:8080 # dashboard

volumes:

# must mount the docker socket so that traefik can listen to changes

- /var/run/docker.sock:/var/run/docker.sock

deploy:

# this basically says that only deploy 1 per node in every manager node

# and reserve 128MB of ram to it, also limit the memory to 256MB

#mode: global

replicas: 1

placement:

constraints:

- "node.role == manager"

resources:

reservations:

memory: 128M

limits:

memory: 256M

web:

image: nginx

networks:

- ejemplo

ports:

- 8080:80

deploy:

#replicas: 2

labels:

- "traefik.enable=true"

# by default, traefik picks up the first exposed port, we can explicitly set it

# to something else here

- "traefik.http.services.web.loadbalancer.server.port=8000"

# tell traefik to send all requests to `docker-swarm.local` to this service

- "traefik.http.routers.web.rule=Host(`docker-swarm.local`)"

# the default entry point is `web` which is HTTP

- "traefik.http.routers.web.entrypoints=web"

placement:

constraints: [node.role == worker]

jenkins1:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

labels:

- "traefik.enable=true"

# by default, traefik picks up the first exposed port, we can explicitly set it

# to something else here

- "traefik.http.services.jenkins1.loadbalancer.server.port=8000"

# tell traefik to send all requests to `docker-swarm.local` to this service

- "traefik.http.routers.jenkins1.rule=Host(`docker-swarm.local`)"

# the default entry point is `web` which is HTTP

- "traefik.http.routers.jenkins1.entrypoints=web"

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8001:8080'

volumes:

# Bind volumne host to container

- 'jenkins1:/var/jenkins\_home'

networks:

- ejemplo

jenkins2:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

labels:

- "traefik.enable=true"

# by default, traefik picks up the first exposed port, we can explicitly set it

# to something else here

- "traefik.http.services.jenkins2.loadbalancer.server.port=8000"

# tell traefik to send all requests to `docker-swarm.local` to this service

- "traefik.http.routers.jenkins2.rule=Host(`docker-swarm.local`)"

# the default entry point is `web` which is HTTP

- "traefik.http.routers.jenkins2.entrypoints=web"

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8002:8080'

volumes:

# Bind volumne host to container

- 'jenkins2:/var/jenkins\_home'

networks:

- ejemplo

jenkins3:

# Spring boot app image from DockerHub

image: jenkins/jenkins:2.235.5

deploy:

# Number of replicas

#replicas: 2

placement:

constraints: [node.role == worker]

labels:

- "traefik.enable=true"

# by default, traefik picks up the first exposed port, we can explicitly set it

# to something else here

- "traefik.http.services.jenkins3.loadbalancer.server.port=8000"

# tell traefik to send all requests to `docker-swarm.local` to this service

- "traefik.http.routers.jenkins3.rule=Host(`docker-swarm.local`)"

# the default entry point is `web` which is HTTP

- "traefik.http.routers.jenkins3.entrypoints=web"

ports:

# Expose container port 8080 to host on port 8080 (HOST:CONTAINER)

- '8003:8080'

volumes:

# Bind volumne host to container

- 'jenkins3:/var/jenkins\_home'

networks:

- ejemplo

visualizersvc:

# Image of docker-swarm-visualizer app

image: dockersamples/visualizer

# This service will run only when appsvc has no failure

depends\_on:

- jenkins1

- jenkins2

- jenkins3

- nginx

deploy:

placement:

# We want to deploy this service only in manager node (suppose we have only one)

constraints:

- 'node.role==manager'

labels:

- "traefik.enable=true"

# by default, traefik picks up the first exposed port, we can explicitly set it

# to something else here

- "traefik.http.services.visualizersvc.loadbalancer.server.port=8000"

# tell traefik to send all requests to `docker-swarm.local` to this service

- "traefik.http.routers.visualizersvc.rule=Host(`docker-swarm.local`)"

# the default entry point is `web` which is HTTP

- "traefik.http.routers.visualizersvc.entrypoints=web"

ports:

# Host port : container port

- '5000:8080'

volumes:

# Bind volumne host to container

- '/var/run/docker.sock:/var/run/docker.sock'

networks:

ejemplo:

external: true

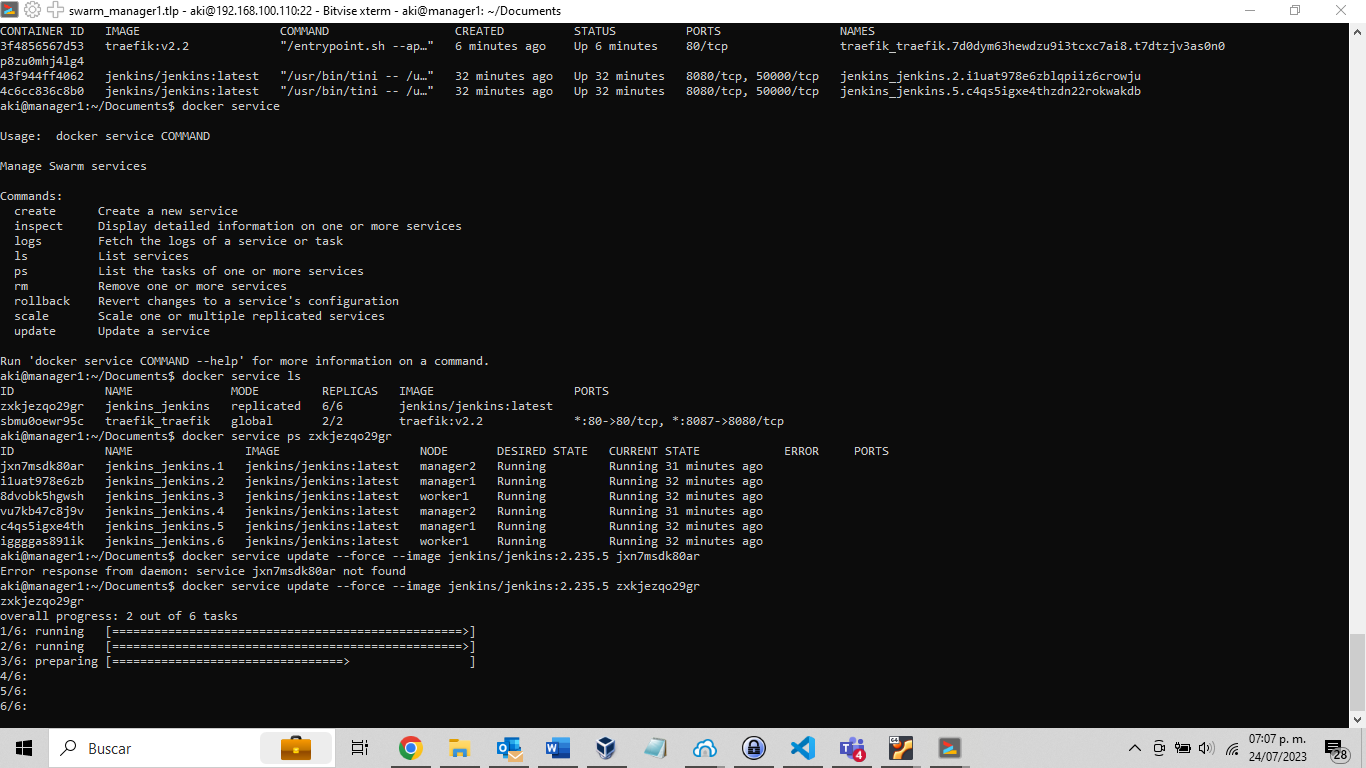
volumes:

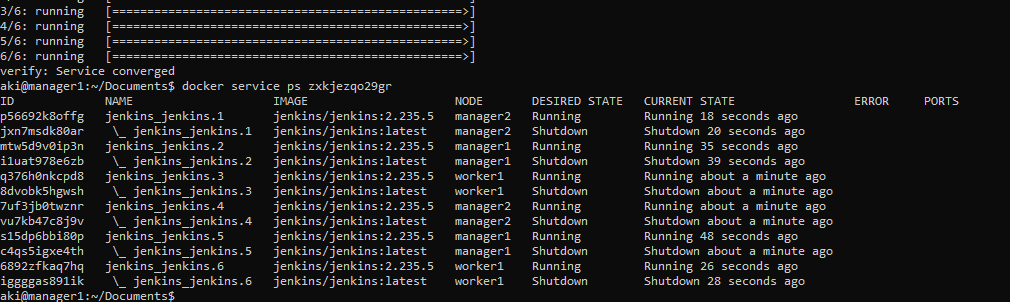
jenkins1:

jenkins2:

jenkins3:

En esta imagen, los servicios eran solamente 1 jenkins pero replicado 6 veces, por lo que al actualizarlo, se tomó el id que sale del comando docker service ls, de esta manera se hizo el update para todos los jenkins al mismo tiempo.





Una vez que se logró tener los servicios separados, o sea, crear jenkins1, jenkins2, jenkins3, entonces se puede cambiar la imagen de manera individual. Pero las imagenes deben de estar tanto en el manager como en el worker.

