Version : **2023.01**

Last update : 2023/12/17 05:50

DOE605 - Docker Compose, Docker Machine and Docker Swarm

Contents

LAB #1 - Docker Compose

Docker Compose is a utility for compiling images and managing multiple containers, all integrated into a single software application. Its role is to make it easier to handle multiple interconnected elements. To do this, Docker Compose uses a file called **docker-compose.yml** in **YAML** format.

This file, called by the **docker-compose build** command, begins with an **image** or **build** keyword, depending on whether the image is retrieved from a registry or comes from the directory cited in the file. The rest of the file contains instructions for defining the compilation of the constituent images, for linking containers and for defining the environment.

Once it has been fully built, the application can then be run very simply using the docker-compose command, which reacts in the same way as the **docker** command, but this time on all the containers defined in the **docker-compose.yml** file.

In this way it is possible to start the application with the **docker-compose up** command, stop it with the **docker-compose stop** command or restart it with the **docker-compose restart** command. In the same way as the **docker** command, the docker-compose command provides access to logs via the **docker-compose logs** command.

1.1 - Installation

Retrieve docker-compose with **curl** :

```
root@debian11:~# curl -L "https://github.com/docker/compose/releases/download/v2.0.1/docker-compose-$(uname -s)-
$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 24.7M  100 24.7M    0     0  45.5M      0 --:--:-- --:--:-- --:--:-- 45.5M
```

Check the installed version:

```
root@debian11:~# docker-compose --version
Docker Compose version v2.0.1
```

Make **/usr/local/bin/docker-compose** executable:

```
root@debian11:~# chmod +x /usr/local/bin/docker-compose

root@debian11:~# ls -l /usr/local/bin/docker-compose
-rwxr-xr-x 1 root root 25907200 Dec 16 12:15 /usr/local/bin/docker-compose
```

The options for the **docker-compose** command are:

```
root@debian11:~# docker-compose --help

Usage: docker compose [OPTIONS] COMMAND

Docker Compose

Options:
      --ansi string Control when to print ANSI control
                                   characters ("never"|"always"|"auto")
                                   (default "auto")
      --compatibility Run compose in backward compatibility mode
      --env-file string Specify an alternate environment file.
  -f, --file stringArray Compose configuration files
      --profile stringArray Specify a profile to enable
      --project-directory string Specify an alternate working directory
                                   (default: the path of the Compose file)
  -p, --project-name string Project name

Commands:
  build Build or rebuild services
  convert Converts the compose file to platform's canonical format
  cp Copy files/folders between a service container and the local filesystem
  create Creates containers for a service.
  down Stop and remove containers, networks
  events Receives real time events from containers.
  exec Execute a command in a running container.
  images List images used by the created containers
  kill Force stop service containers.
  logs View output from containers
  ls List running compose projects
  pause pause services
  port Print the public port for a port binding.
  ps List containers
  pull Pull service images
  push Push service images
  restart Restart containers
  rm Removes stopped service containers
  run Run a one-off command on a service.
  start Start services
  stop Stop services
  top Display the running processes
  unpause unpause services
  up Create and start containers

Run 'docker compose COMMAND --help' for more information on a command.
```

2.2 - Installing Wordpress with Docker Compose

Now create the **wordpress1** directory in /root :

```
root@debian9:~# mkdir wordpress1
```

Go into the directory and create the **docker-compose.yaml** file.

```
root@debian9:~# cd wordpress1
```

```
root@debian9:~/wordpress1# vi docker-compose.yaml
root@debian9:~/wordpress1# cat docker-compose.yaml
version: "3.3"
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: fenestros
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
```

Run the docker-compose command :

```
root@debian11:~/wordpress1# docker-compose up -d
[+] Running 12/12
 ⠿ db Pulled
```

```
16.0s
  ⠿ 20e4dcae4c69 Pull complete
5.7s
  ⠿ 1c56c3d4ce74 Pull complete
5.8s
  ⠿ e9f03a1c24ce Pull complete
5.9s
  ⠿ 68c3898c2015 Pull complete
6.2s
  ⠿ 6b95a940e7b6 Pull complete
6.3s
  ⠿ 90986bb8de6e Pull complete
6.3s
  ⠿ ae71319cb779 Pull complete
7.4s
  ⠿ ffc89e9dfd88 Pull complete
7.4s
  ⠿ 43d05e938198 Pull complete
15.2s
  ⠿ 064b2d298fba Pull complete
15.3s
  ⠿ df9a4d85569b Pull complete
15.3s
[+] Running 4/4
 ⠿ Network wordpress1_default        Created
0.1s
 ⠿ Volume "wordpress1_db_data"       Created
0.0s
 ⠿ Container wordpress1-db-1         Started
10.7s
 ⠿ Container wordpress1-wordpress-1  Started
1.5s
```

Check that wordpress is running:

```
root@debian11:~/wordpress1# lynx --dump http://10.0.3.46:8000
   WordPress
   Select a default language [English (United States)__________]

   Continue
root@debian11:~/wordpress1# docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED              STATUS
PORTS                                   NAMES
9475874569e1   wordpress:latest                          "docker-entrypoint.s…"   51 seconds ago       Up 49
seconds   0.0.0.0:8000->80/tcp, :::8000->80/tcp   wordpress1-wordpress-1
5983cdf711ec   mysql:5.7                                 "docker-entrypoint.s…"   About a minute ago   Up 50
seconds   3306/tcp, 33060/tcp                     wordpress1-db-1
cf27f30654d2   dockersamples/examplevotingapp_worker   "dotnet Worker.dll"      20 hours ago         Up 20 hours
worker
33a264a36bdc   dockersamples/examplevotingapp_result   "/usr/bin/tini -- no…"   20 hours ago         Up 20 hours
0.0.0.0:5001->80/tcp, :::5001->80/tcp   result
81e6fcb9f692   dockersamples/examplevotingapp_vote     "gunicorn app:app -b…"   20 hours ago         Up 20 hours
0.0.0.0:5000->80/tcp, :::5000->80/tcp   vote
5083545dcbf8   postgres:9.4                              "docker-entrypoint.s…"   20 hours ago         Up 20 hours
5432/tcp                                db
227554e3e4c1   redis                                     "docker-entrypoint.s…"   20 hours ago         Up 20 hours
6379/tcp                                redis
63fec083f4d6   wordpress                                 "docker-entrypoint.s…"   21 hours ago         Up 21 hours
10.0.3.46:80->80/tcp                    wordpress
db3732939266   mysql:latest                              "docker-entrypoint.s…"   21 hours ago         Up 21 hours
3306/tcp, 33060/tcp                     wordpressdb
57e92a8b25d7   centos                                    "/bin/bash"              21 hours ago         Up 21 hours
centos3
fc417b22a20d   centos                                    "/bin/bash"              21 hours ago         Up 21 hours
centos2
cb2875ab1059   centos                                    "/bin/bash"              21 hours ago         Up 21 hours
centos1
2126924504d8   centos                                    "/bin/bash"              21 hours ago         Up 21 hours
resotest
```

```
root@debian11:~/wordpress1# docker inspect wordpress1-wordpress-1 | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.18.0.3",
root@debian11:~/wordpress1# lynx --dump http://172.18.0.3
   WordPress
   Select a default language [English (United States)__________]

   Continue
```

Important - The **docker-compose up** command is an alias to the **docker-compose build && docker-compose run** commands. The **-d** option has the same effect as its **docker** command counterpart.

LAB #2 - Docker Machine

2.1 - Overview

Docker Machine is a tool that lets you install docker on virtual hosts and manage the hosts using commands specific to docker-machine. It is therefore possible to use this tool to create docker hosts locally, on the network, in a data centre or in the cloud (Azure, AWS, Digital Ocean for example).

The docker-machine command set allows you to start, monitor, stop and restart a managed host, update the docker client/daemon and configure a docker client so that it 'talks' to your host machine.

To install docker-machine on your **debian11** VM, use the following command:

```
root@debian11:~/wordpress1# cd ~

root@debian11:~# curl -L https://github.com/docker/machine/releases/download/v0.16.2/docker-machine-`uname -s`-
```

```
`uname -m` >/tmp/docker-machine && chmod +x /tmp/docker-machine && cp /tmp/docker-machine /usr/local/bin/docker-
machine
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 32.6M  100 32.6M    0     0  48.5M      0 --:--:-- --:--:-- --:--:-- 48.5M
```

2.2 - Creating Docker Virtual Machines

Creating a machine is done simply by using the **docker-machine** command with the **create** subcommand. This sub-command takes the **-driver** or **-d** option, which indicates the provider to be used:

Provider	**driver**
Amazon Web Services	amazonec2
Digital Ocean	digitalocean
Exoscale	exoscale
Google Compute Engine	google
IBM Softlayer	softlayer
Microsoft Hyper-V	hyperv
Microsoft Azure	azure
OpenStack	openstack
Oracle VirtualBox	virtualbox
Rackspace	rackspace
VMware Fusion	vmwarefusion
VMware vCloud Air	vmwarevcloudair
VMware vSphere	vmwarevsphere

Start by installing Oracle VirtualBox:

```
root@debian11:~# apt install virtualbox-6.1 -y
Reading package lists... Done
```

```
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libopengl0 linux-headers-5.10.0-15-amd64 linux-headers-5.10.0-15-common
Use 'apt autoremove' to remove them.
Recommended packages:
  linux-image
The following packages will be upgraded:
  virtualbox-6.1
1 upgraded, 0 newly installed, 0 to remove and 16 not upgraded.
Need to get 95.8 MB of archives.
After this operation, 53.2 kB disk space will be freed.
Get:1 http://download.virtualbox.org/virtualbox/debian bullseye/contrib amd64 virtualbox-6.1 amd64
6.1.48-159471~Debian~bullseye [95.8 MB]
Fetched 95.8 MB in 3s (31.4 MB/s)
apt-listchanges: Reading changelogs...
Preconfiguring packages ...
(Reading database ... 166797 files and directories currently installed.)
Preparing to unpack .../virtualbox-6.1_6.1.48-159471~Debian~bullseye_amd64.deb ...
Unpacking virtualbox-6.1 (6.1.48-159471~Debian~bullseye) over (6.1.46-158378~Debian~bullseye) ...
Setting up virtualbox-6.1 (6.1.48-159471~Debian~bullseye) ...
addgroup: The group `vboxusers' already exists as a system group. Exiting.
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for shared-mime-info (2.0-1) ...
Processing triggers for mailcap (3.69) ...
Processing triggers for desktop-file-utils (0.26-1) ...
```

Create the **/etc/vbox/networks.conf** file:

```
root@debian11:~# vi /etc/vbox/networks.conf
root@debian11:~# cat /etc/vbox/networks.conf
* 10.0.0.0/8 192.168.0.0/16
```

Now create the **manager1** virtual machine:

```
root@debian11:~# docker-machine create --driver virtualbox manager1
Running pre-create checks...
(manager1) Image cache directory does not exist, creating it at /root/.docker/machine/cache...
(manager1) No default Boot2Docker ISO found locally, downloading the latest release...
(manager1) Latest release for github.com/boot2docker/boot2docker is v19.03.12
(manager1) Downloading /root/.docker/machine/cache/boot2docker.iso from
https://github.com/boot2docker/boot2docker/releases/download/v19.03.12/boot2docker.iso...
(manager1) 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
Creating machine...
(manager1) Copying /root/.docker/machine/cache/boot2docker.iso to
/root/.docker/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 this virtual machine, run: docker-
machine env manager1
```

The options for the **docker-machine** command are:

```
root@debian11:~# docker-machine --help
Usage: docker-machine [OPTIONS] COMMAND [arg...]
```

```
Create and manage machines running Docker.

Version: 0.16.2, build bd45ab13

Author:
  Docker Machine Contributors - <https://github.com/docker/machine>

Options:
  --debug, -D Enable debug mode
  --storage-path, -s "/root/.docker/machine" Configures storage path [$MACHINE_STORAGE_PATH]
  --tls-ca-cert CA to verify remotes against [$MACHINE_TLS_CA_CERT]
  --tls-ca-key Private key to generate certificates [$MACHINE_TLS_CA_KEY]
  --tls-client-cert Client cert to use for TLS [$MACHINE_TLS_CLIENT_CERT]
  --tls-client-key Private key used in client TLS auth [$MACHINE_TLS_CLIENT_KEY]
  --github-api-token Token to use for requests to the Github API [$MACHINE_GITHUB_API_TOKEN]
  --native-ssh Use the native (Go-based) SSH implementation. [$MACHINE_NATIVE_SSH]
  --bugsnag-api-token BugSnag API token for crash reporting [$MACHINE_BUGSNAG_API_TOKEN]
  --help, -h show help
  --version, -v print the version
Commands:
  active Print which machine is active
  config Print the connection config for machine
  create Create a machine
  env Display the commands to set up the environment for the Docker client
  inspect Inspect information about a machine
  ip Get the IP address of a machine
  kill Kill a machine
  ls List machines
  provision Re-provision existing machines
  regenerate-certs Regenerate TLS Certificates for a machine
  restart Restart a machine
  rm Remove a machine
  ssh Log into or run a command on a machine with SSH.
  scp Copy files between machines
  mount Mount or unmount a directory from a machine with SSHFS.
  start Start a machine
  status Get the status of a machine
  stop Stop a machine
  upgrade Upgrade a machine to the latest version of Docker
  url Get the URL of a machine
  version Show the Docker Machine version or a machine docker version
  help Shows a list of commands or help for one command
Run 'docker-machine COMMAND --help' for more information on a command.
```

The options for the **create** subcommand of the **docker-machine** command are:

```
root@debian11:~# docker-machine create --help
Usage: docker-machine create [OPTIONS] [arg...]

Create a machine

Description:
   Run 'docker-machine create --driver name --help' to include the create flags for that driver in the help text.

Options:
   --driver, -d "virtualbox" Driver to create machine with. [$MACHINE_DRIVER]
   --engine-env [--engine-env option --engine-env option] Specify environment variables to set in the engine
   --engine-insecure-registry [--engine-insecure-registry option --engine-insecure-registry option] Specify
insecure registries to allow with the created engine
   --engine-install-url "https://get.docker.com"                                                    Custom
URL to use for engine installation [$MACHINE_DOCKER_INSTALL_URL]
   --engine-label [--engine-label option --engine-label option] Specify labels for the created engine
   --engine-opt [--engine-opt option --engine-opt option] Specify arbitrary flags to include with the created
engine in the form flag=value
   --engine-registry-mirror [--engine-registry-mirror option --engine-registry-mirror option] Specify registry
mirrors to use [$ENGINE_REGISTRY_MIRROR]
   --engine-storage-driver Specify a storage driver to use with the engine
   --swarm Configure Machine to join a Swarm cluster
   --swarm-addr addr to advertise for Swarm (default: detect and use the machine IP)
   --swarm-discovery Discovery service to use with Swarm
   --swarm-experimental Enable Swarm experimental features
   --swarm-host "tcp://0.0.0.0:3376" ip/socket to listen on for Swarm master
   --swarm-image "swarm:latest" Specify Docker image to use for Swarm [$MACHINE_SWARM_IMAGE]
   --swarm-join-opt [--swarm-join-opt option --swarm-join-opt option] Define arbitrary flags for Swarm join
   --swarm-master Configure Machine to be a Swarm master
   --swarm-opt [--swarm-opt option --swarm-opt option] Define arbitrary flags for Swarm master
   --swarm-strategy "spread" Define a default scheduling strategy for Swarm
   --tls-san [--tls-san option --tls-san option] Support extra SANs for TLS certs
   --virtualbox-boot2docker-url The URL of the boot2docker image. Defaults to the latest available version
[$VIRTUALBOX_BOOT2DOCKER_URL]
   --virtualbox-cpu-count "1" number of CPUs for the machine (-1 to use the number of CPUs available)
[$VIRTUALBOX_CPU_COUNT]
   --virtualbox-disk-size "20000" Size of disk for host in MB [$VIRTUALBOX_DISK_SIZE]
   --virtualbox-host-dns-resolver Use the host DNS resolver [$VIRTUALBOX_HOST_DNS_RESOLVER]
   --virtualbox-hostonly-cidr "192.168.99.1/24" Specify the Host Only CIDR [$VIRTUALBOX_HOSTONLY_CIDR]
   --virtualbox-hostonly-nicpromisc "deny" Specify the Host Only Network Adapter Promiscuous Mode
[$VIRTUALBOX_HOSTONLY_NIC_PROMISC]
   --virtualbox-hostonly-nictype "82540EM" Specify the Host Only Network Adapter Type
[$VIRTUALBOX_HOSTONLY_NIC_TYPE]
   --virtualbox-hostonly-no-dhcp Disable the Host Only DHCP Server [$VIRTUALBOX_HOSTONLY_NO_DHCP]
   --virtualbox-import-boot2docker-vm The name of a Boot2Docker VM to import [$VIRTUALBOX_BOOT2DOCKER_IMPORT_VM]
   --virtualbox-memory "1024" Size of memory for host in MB [$VIRTUALBOX_MEMORY_SIZE]
   --virtualbox-nat-nictype "82540EM" Specify the Network Adapter Type [$VIRTUALBOX_NAT_NICTYPE]
   --virtualbox-no-dns-proxy Disable proxying all DNS requests to the host [$VIRTUALBOX_NO_DNS_PROXY]
   --virtualbox-no-share Disable the mount of your home directory [$VIRTUALBOX_NO_SHARE]
   --virtualbox-no-vtx-check Disable checking for the availability of hardware virtualization before the vm is
started [$VIRTUALBOX_NO_VTX_CHECK]
   --virtualbox-share-folder Mount the specified directory instead of the default home location. Format: dir:name
[$VIRTUALBOX_SHARE_FOLDER]
   --virtualbox-ui-type "headless" Specify the UI Type: (gui|sdl|headless|separate) [$VIRTUALBOX_UI_TYPE]
```

Now create 5 workers - **worker1** through **worker5** :

```
root@debian11:~# docker-machine create --driver virtualbox worker1
Running pre-create checks...
Creating machine...
(worker1) Copying /root/.docker/machine/cache/boot2docker.iso to
/root/.docker/machine/machines/worker1/boot2docker.iso...
(worker1) Creating VirtualBox VM...
(worker1) Creating SSH key...
(worker1) Starting the VM...
(worker1) Check network to re-create if needed...
(worker1) 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 this virtual machine, run: docker-
machine env worker1
```

```
root@debian11:~# docker-machine create --driver virtualbox worker2
...
root@debian11:~# docker-machine create --driver virtualbox worker3
...
root@debian11:~# docker-machine create --driver virtualbox worker4
...
root@debian11:~# docker-machine create --driver virtualbox worker5
...
```

2.3 - Listing Docker VMs

To list Docker VMs and their states, use the **ls** subcommand of the **docker-machine** command:

```
root@debian11:~# docker-machine ls
NAME       ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER      ERRORS
manager1   -        virtualbox   Running   tcp://192.168.99.100:2376           v19.03.12
worker1    -        virtualbox   Running   tcp://192.168.99.101:2376           v19.03.12
worker2    -        virtualbox   Running   tcp://192.168.99.102:2376           v19.03.12
worker3    -        virtualbox   Running   tcp://192.168.99.103:2376           v19.03.12
worker4    -        virtualbox   Running   tcp://192.168.99.104:2376           v19.03.12
worker5    -        virtualbox   Running   tcp://192.168.99.105:2376           v19.03.12
```

2.4 - Obtaining the IP address of VMs

Another way of obtaining the IP addresses of the VMs is to use the **ip** subcommand:

```
root@debian11:~# docker-machine ip manager1
192.168.99.100
root@debian11:~# docker-machine ip worker1
192.168.99.101
root@debian11:~# docker-machine ip worker2
192.168.99.102
root@debian11:~# docker-machine ip worker3
192.168.99.103
root@debian11:~# docker-machine ip worker4
192.168.99.104
root@debian11:~# docker-machine ip worker5
192.168.99.105
```

2.5 - Connecting to a Docker VM

To connect to a Docker Machine VM, use the **ssh** subcommand of the **docker-machine** command:

```
root@debian11:~# docker-machine ssh manager1
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@manager1:~$ exit
logout
root@debian11:~#
```

Important - Note that the VM distribution is **Boot2Docker**. This distribution is based on **Tiny Core Linux**, runs entirely in RAM, weighs 27MB and boots in approximately 5 seconds.

Now install the **mlocate** package:

```
root@debian11:~# apt install mlocate -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libopengl0 linux-headers-5.10.0-15-amd64 linux-headers-5.10.0-15-common
Use 'apt autoremove' to remove them.
Suggested packages:
  nocache
The following NEW packages will be installed:
  mlocate

```

```
0 upgraded, 1 newly installed, 0 to remove and 16 not upgraded.
Need to get 98.3 kB of archives.
After this operation, 517 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main amd64 mlocate amd64 0.26-5 [98.3 kB]
Fetched 98.3 kB in 0s (1,985 kB/s)
Selecting previously unselected package mlocate.
(Reading database ... 166797 files and directories currently installed.)
Preparing to unpack .../mlocate_0.26-5_amd64.deb ...
Unpacking mlocate (0.26-5) ...
Setting up mlocate (0.26-5) ...
update-alternatives: using /usr/bin/mlocate to provide /usr/bin/locate (locate) in auto mode
Adding group `mlocate' (GID 123) ...
Done.
Created symlink /etc/systemd/system/timers.target.wants/mlocate.timer → /lib/systemd/system/mlocate.timer.
mlocate.service is a disabled or a static unit, not starting it.
Processing triggers for man-db (2.9.4-2) ...
```

Having been created by root, Docker Machine VMs and their associated files are stored in the **/root/.docker/machine/machines/** directory :

```
root@debian11:~# updatedb

root@debian11:~# locate manager1
/root/.docker/machine/machines/manager1
/root/.docker/machine/machines/manager1/boot2docker.iso
/root/.docker/machines/manager1/ca.pem
/root/.docker/machine/machines/manager1/cert.pem
/root/.docker/machine/machines/manager1/config.json
/root/.docker/machine/machines/manager1/disk.vmdk
/root/.docker/machine/machines/manager1/id_rsa
/root/.docker/machine/machines/manager1/id_rsa.pub
/root/.docker/machine/machines/manager1/key.pem
/root/.docker/machine/machines/manager1/manager1
/root/.docker/machine/machines/manager1/server-key.pem
/root/.docker/machine/machines/manager1/server.pem
```

```
/root/.docker/machine/machines/manager1/manager1/Logs
/root/.docker/machine/machines/manager1/manager1/manager1.vbox
/root/.docker/machines/manager1/manager1/manager1.vbox-prev
/root/.docker/machine/machines/manager1/manager1/Logs/VBox.log
```

LAB #3 - Docker Swarm

3.1 - Overview

Docker Swarm is a utility that allows you to manage a cluster to deploy containers by enabling an imitation of docker behaviour on a single machine.

3.2 - Initializing Docker Swarm

To initialise Docker swarm, use the **docker swarm init** command from the Docker VM **manager1**, specifying the IP address of manager1:

```
root@debian11:~# docker-machine ssh manager1
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@manager1:~$ docker swarm init --advertise-addr 192.168.99.100
Swarm initialized: current node (y0war0lijmwhnexrfhfflulsd) is now a manager.

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

    docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377

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

Note that the following ports must be open on a manager node: 22/tcp, 2376/tcp, 2377/tcp, 7946/tcp, 7946/udp and 4789/udp.

3.3 - Leader status

View the status of the Docker manager1 VM:

```
docker@manager1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
y0war0lijmwhnexrfhfflulsd *   manager1            Ready               Active              Leader
19.03.12
```

At any given time there can only be one **Leader**. It is possible to create other manager nodes by joining them to swarm using the token provided for this purpose. However, these management nodes remain on standby for any failure of the current Leader.

To find out the token required to join swarm as a management node, enter the following command:

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

    docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-1teue04djnc4vu2eufzty59ys 192.168.99.100:2377
```

3.4 - Join the Swarm

Join the 5 swarm worker machines using the **worker** token :

```
docker@manager1:~$ exit
logout

root@debian11:~# docker-machine ssh worker1
```

```
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker1:~$ docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377
This node joined a swarm as a worker.

docker@worker1:~$ exit
logout

root@debian11:~# docker-machine ssh worker2
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker2:~$ docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377
This node joined a swarm as a worker.

docker@worker2:~$ exit
logout

root@debian11:~# docker-machine ssh worker3
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker3:~$ docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377
This node joined a swarm as a worker.

docker@worker3:~$ exit
logout
```

```
root@debian11:~# docker-machine ssh worker4
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker4:~$ docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377
This node joined a swarm as a worker.

docker@worker4:~$ exit
logout

root@debian11:~# docker-machine ssh worker5
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker5:~$ docker swarm join --token
SWMTKN-1-25tbmsyx452cuhpiedymuc8n9llo9jbjcbza93npdo35q1aaff-9teuxcpe09xw7v7xz1tnwpw4g 192.168.99.100:2377
This node joined a swarm as a worker.

docker@worker5:~$ exit
logout

root@debian11:~#
```

Note that the following ports must be open on a worker node: 22/tcp, 2376/tcp, 7946/tcp, 7946/udp and 4789/udp.

The status of Docker VMs can be viewed by using the **docker node ls** command again:

```
root@debian11:~# docker-machine ssh manager1
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net
```

```
docker@manager1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
y0war0lijmwhnexrfhfflulsd *   manager1            Ready               Active              Leader
19.03.12
v5ai62lmhfsdcauccqmjyu6qk     worker1             Ready               Active
19.03.12
j4mr3d2ji30t7hu0trob5dpgz     worker2             Ready               Active
19.03.12
pouo7nuvirq0qkuvvrp04a47h     worker3             Ready               Active
19.03.12
mo0dd5ech6ifdgd8pa6cjz896     worker4             Ready               Active
19.03.12
5am2vd39pybytu1nd3oooabtq     worker5             Ready               Active
19.03.12
```

Note that you cannot use this command from a worker:

```
docker@manager1:~$ exit
logout

root@debian11:~# docker-machine ssh worker5
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker5:~$ docker node ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or modify
cluster state. Please run this command on a manager node or promote the current node to a manager.

docker@worker5:~$ exit
logout
exit status 1
```

```
root@debian11:~#
```

====3.5 - Viewing Swarm==== Information

It is possible to view information about the swarm using the **docker info** command:

```
root@debian11:~# docker-machine ssh manager1
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@manager1:~$ docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: active
  NodeID: y0war0lijmwhnexrfhfflulsd
  Is Manager: true
  ClusterID: w3mq6i01k4siboyqv3w3nbyu7
  Managers: 1
  Nodes: 6
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
   Snapshot Interval: 10000
   Number of Old Snapshots to Retain: 0
   Heartbeat Tick: 1
   Election Tick: 10
  Dispatcher:
   Heartbeat Period: 5 seconds
  CA Configuration:
   Expiry Duration: 3 months
   Force Rotate: 0
  Autolock Managers: false
  Root Rotation In Progress: false
  Node Address: 192.168.99.100
  Manager Addresses:
   192.168.99.100:2377
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.19.130-boot2docker
```

```
Operating System: Boot2Docker 19.03.12 (TCL 10.1)
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 985.4MiB
Name: manager1
ID: UIL3:SNFZ:J2XE:4QVD:7XYM:MPB7:2LHC:NMY7:B4CU:QMUK:3A66:H3G6
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
 provider=virtualbox
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

Important - When the Docker Engine is in swarm mode, management nodes implement the **Raft Consensus Algorithm** to manage cluster state.

3.6 - Starting a Service

In this example, we will start the **nginx** service with the following properties:

- Mapping the nginx port to port 80 on the host machine,
- 5 instances of the service,
- A unique name of **web**.

```
docker@manager1:~$ docker service create --replicas 5 -p 80:80 --name web nginx
oree977o1unlk5ndos0y44i2h
overall progress: 5 out of 5 tasks
1/5: running [==================================================>]
2/5: running [==================================================>]
3/5: running [==================================================>]
4/5: running [==================================================>]
5/5: running [==================================================>]
verify: Service converged
```

To check the status of this service, use the **docker service ls** command :

```
docker@manager1:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
oree977o1unl        web                 replicated          5/5                 nginx:latest        *:80->80/tcp
```

This service runs in Docker containers:

```
docker@manager1:~$ docker service ps web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
son0vgc73drb        web.1               nginx:latest        worker4             Running             Running 48
seconds ago
ojqyweuo65jw        web.2               nginx:latest        worker5             Running             Running 49
seconds ago
mb40onnaxd0u        web.3               nginx:latest        manager1            Running             Running 49
seconds ago
4vwsho5x7i36        web.4               nginx:latest        worker2             Running             Running 49
seconds ago
sk9hr6j2u47c        web.5               nginx:latest        worker3             Running             Running 50
seconds ago
```

Important - Note that there is no container on worker1.

To see that the nginx daemon has been launched, run the **docker ps** command on the **manager1** machine:

```
docker@manager1:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS               PORTS
NAMES
244fecd46312        nginx:latest        "/docker-entrypoint.…"   About a minute ago   Up About a minute    80/tcp
web.3.mb40onnaxd0u8t0nhbzx9rdih
```

On the other hand, the same command executed on **worker1** gives the following result:

```
docker@manager1:~$ exit
logout

root@debian11:~# docker-machine ssh worker1
   ( '>')
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@worker1:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
docker@worker1:~$ exit
logout

root@debian11:~#
```

Connect to each Docker Machine VM to see that the nginx service is accessible:

```
root@debian11:~# docker-machine ssh manager1
   ( '>')
```

```
  /) TC (\ Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\) www.tinycorelinux.net

docker@manager1:~$ curl 192.168.99.100
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

docker@manager1:~$ curl 192.168.99.101
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
```

```
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

docker@manager1:~$ curl 192.168.99.102
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

```
<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

docker@manager1:~$ curl 192.168.99.103
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

```
docker@manager1:~$ curl 192.168.99.104
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

docker@manager1:~$ curl 192.168.99.105
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
```

```
</style>
</head>
<body>
<h1>Welcome to nginx</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Important - Note that the service is even available on worker1.

3.7 - Scaling Up and Scaling Down the Service

Currently, there are 5 containers running. To scale up to 8 containers, the **docker service scale** command should be used:

```
docker@manager1:~$ docker service scale web=8
web scaled to 8
overall progress: 8 out of 8 tasks
1/8: running [==================================================>]
2/8: running [==================================================>]
3/8: running [==================================================>]
4/8: running [==================================================>]
5/8: running [==================================================>]
```

```
6/8: running [==================================================>]
7/8: running [==================================================>]
8/8: running [==================================================>]
verify: Service converged
```

Note that the **docker service ls** command confirms the fact that there are 8 replicas :

```
docker@manager1:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
oree977o1unl        web                 replicated          8/8                 nginx:latest        *:80->80/tcp
```

Of the three additional replicas, two were run on worker5 while the third was run on worker1 :

```
docker@manager1:~$ docker service ps web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
son0vgc73drb        web.1               nginx:latest        worker4             Running             Running 9
minutes ago
ojqyweuo65jw        web.2               nginx:latest        worker5             Running             Running 9
minutes ago
mb40onnaxd0u        web.3               nginx:latest        manager1            Running             Running 9
minutes ago
4vwsho5x7i36        web.4               nginx:latest        worker2             Running             Running 9
minutes ago
sk9hr6j2u47c        web.5               nginx:latest        worker3             Running             Running 9
minutes ago
pui4dnkepu27        web.6               nginx:latest        worker1             Running             Running about
a minute ago
yj0kexdcuo5u        web.7               nginx:latest        worker5             Running             Running about
a minute ago
93jtbxqj2dyz        web.8               nginx:latest        worker1             Running             Running about
a minute ago
```

3.8 - Checking the status of a node

To find out the status of the current node, use the **docker node inspect** command with the **self** keyword:

```
docker@manager1:~$ docker node inspect self
[
    {
        "ID": "y0war0lijmwhnexrfhfflulsd",
        "Version": {
            { "Index": 9
        },
        "CreatedAt": "2023-12-16T13:38:18.600836601Z",
        "UpdatedAt": "2023-12-16T13:38:19.126257379Z",
        "Spec": {
            "Labels": {},
            "Role": "manager",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "manager1",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 1000000000,
                "MemoryBytes": 1033252864
            },
            "Engine": {
                "EngineVersion": "19.03.12",
                "Labels": {
                    "provider": "virtualbox"
                },
                "Plugins": [
                    {
                        "Type": "Log",
                        "Name": "awslogs"
                    },
                    {
                        "Type": "Log",
                        "Name": "fluentd"
                    },
                    {
                        "Type": "Log",
                        "Name": "gcplogs"
                    },
                    {
                        "Type": "Log",
                        "Name": "gelf"
                    },
                    {
                        "Type": "Log",
                        "Name": "journald"
                    },
                    {
                        "Type": "Log",
                        "Name": "json-file"
                    },
                    {
                        "Type": "Log",
                        "Name": "local"
                    },
                    {
                        "Type": "Log",
                        "Name": "logentries"
                    },
                    {
```

```
                    "Type": "Log",
                    "Name": "splunk"
                },
                {
                    "Type": "Log",
                    "Name": "syslog"
                },
                {
                    "Type": "Network",
                    "Name": "bridge"
                },
                {
                    "Type": "Network",
                    "Name": "host"
                },
                {
                    "Type": "Network",
                    "Name": "ipvlan"
                },
                {
                    "Type": "Network",
                    "Name": "macvlan"
                },
                {
                    "Type": "Network",
                    "Name": "null"
                },
                {
                    "Type": "Network",
                    "Name": "overlay"
                },
                {
                    "Type": "Volume",
                    "Name": "local"
                    }
                ]
            },
            "TLSInfo": {
                "TrustRoot": "-----BEGIN CERTIFICATE-----
\nMIIBazCCARCgAwIBAgIUbi2tpJHqoqK+BA/p9c+Y9AmtQSAwCgYIKoZIzj0EAwIw\nEzERMA8GA1UEAxMIc3dhcm0tY2EwHhcNMjMxMjE2MTMzM
zAwWhcNNDMxMjExMTMz\nMzAwWjATMREwDwYDVQQDEwhzd2FybS1jYTBZMBMGByqGSM49AgEGCCqGSM49AwEH\nA0IABKuD7Svum+bER9CszNtFt7
ASMr5gj6Vea0oM7SCrlyYMCn8ryaHXQ9J+iEIh\nfWrpKmjNtei3/j+leOVF0flpg2OjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMB\nAf8EBTA
DAQH/MB0GA1UdDgQWBBRqlCTTUy9JODtzaVecgmjID/M5kzAKBggqhkjO\nPQQDAgNJADBGAiEA06sGunqGwd23lyjcUoczMWGQGln9nv0pmm/riJ
QvB80CIQC9\nvXNZUbC6U5lSQ7eGhfmmHi1JAfb88wGau0AlreBczw==\n- ----END CERTIFICATE-----\n",
                "CertIssuerSubject": "MBMxETAPBgNVBAMTCHN3YXJtLWNh",
                "CertIssuerPublicKey":
"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEq4PtK+6b5sRH0KzM20W3sBIyvmCPpV5rSgztIKuXJgwKfyvJoddD0n6IQiF9aukqaM216Lf+P6V4
5UXR+WmDYw=="
            }
        },
        "Status": {
            "State": "ready",
            "Addr": "192.168.99.100"
        },
        "ManagerStatus": {
            "Leader": true,
            "Reachability": "reachable",
            "Addr": "192.168.99.100:2377"
        }
    }
]
```

To find out about the status of another node, use the **docker node inspect** command with the name of the node concerned:

```
docker@manager1:~$ docker node inspect worker1
[
    {
        "ID": "v5ai62lmhfsdcauccqmjyu6qk",
        "Version": {
            { "Index": 15
        },
        "CreatedAt": "2023-12-16T13:40:08.301827885Z",
        "UpdatedAt": "2023-12-16T13:40:08.453463845Z",
        "Spec": {
            "Labels": {},
            "Role": "worker",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "worker1",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 1000000000,
                "MemoryBytes": 1033252864
            },
            "Engine": {
                "EngineVersion": "19.03.12",
                "Labels": {
                    "provider": "virtualbox"
                },
                "Plugins": [
                    {
                        "Type": "Log",
                        "Name": "awslogs"
                    },
                    {
                        "Type": "Log",
                        "Name": "fluentd"
                    },
                    {
                        "Type": "Log",
                        "Name": "gcplogs"
                    },
                    {
                        "Type": "Log",
                        "Name": "gelf"
                    },
                    {
                        "Type": "Log",
                        "Name": "journald"
                    },
                    {
                        "Type": "Log",
                        "Name": "json-file"
                    },
                    {
                        "Type": "Log",
                        "Name": "local"
                    },
                    {
                        "Type": "Log",
                        "Name": "logentries"
                    },
                    {
                        "Type": "Log",
                        "Name": "splunk"
                    },
                    {
                        "Type": "Log",
                        "Name": "syslog"
                    },
                    {
                        "Type": "Network",
                        "Name": "bridge"
                    },
                    {
                        "Type": "Network",
                        "Name": "host"
                    },
                    {
                        "Type": "Network",
                        "Name": "ipvlan"
                    },
                    {
                        "Type": "Network",
                        "Name": "macvlan"
                    },
                    {
                        "Type": "Network",
                        "Name": "null"
                    },
                    {
                        "Type": "Network",
                        "Name": "overlay"
                    },
                    {
                        "Type": "Volume",
                        "Name": "local"
                    }
                ]
            },
            "TLSInfo": {
                "TrustRoot": "-----BEGIN CERTIFICATE-----
    \nMIIBazCCARCgAwIBAgIUbi2tpJHqoqK+BA/p9c+Y9AmtQSAwCgYIKoZIzj0EAwIw\nEzERMA8GA1UEAxMIc3dhcm0tY2EwHhcNMjMxMjE2MTMzM
    zAwWhcNNDMxMjExMTMz\nMzAwWjATMREwDwYDVQQDEwhzd2FybS1jYTBZMBMGByqGSM49AgEGCCqGSM49AwEH\nA0IABKuD7Svum+bER9CszNtFt7
    ASMr5gj6Vea0oM7SCrlyYMCn8ryaHXQ9J+iEIh\nfWrpKmjNtei3/j+leOVF0flpg2OjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMB\nAf8EBTA
    DAQH/MB0GA1UdDgQWBBRqlCTTUy9JODtzaVecgmjID/M5kzAKBggqhkjO\nPQQDAgNJADBGAiEA06sGunqGwd23lyjcUoczMWGQGln9nv0pmm/riJ
```

```
QvB80CIQC9\nvXNZUbC6U5lSQ7eGhfmmHi1JAfb88wGau0AlreBczw==\n- ----END CERTIFICATE-----\n",
                "CertIssuerSubject": "MBMxETAPBgNVBAMTCHN3YXJtLWNh",
                "CertIssuerPublicKey":
"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEq4PtK+6b5sRH0KzM20W3sBIyvmCPpV5rSgztIKuXJgwKfyvJoddD0n6IQiF9aukqaM216Lf+P6V4
5UXR+WmDYw=="
            }
        },
        "Status": {
            "State": "ready",
            "Addr": "192.168.99.101"
        }
    }
]
```

The **-pretty** option produces more easily readable output:

```
docker@manager1:/$ docker node inspect --pretty worker1
ID:                     1f5qtolgtonqmhjk5ppwc8x1b
Hostname:               worker1
Joined at:              2017-09-08 11:48:42.011596185 +0000 utc
Status:
 State:                 Ready
 Availability:          Active
 Address:               192.168.99.101
Platform:
 Operating System:      linux
 Architecture:          x86_64
Resources:
 CPUs:                  1
 Memory:                995.8MiB
Plugins:
 Log:               awslogs, fluentd, gcplogs, gelf, journald, json-file, logentries, splunk, syslog
 Network:               bridge, host, macvlan, null, overlay
 Volume:                local
```

```
Engine Version:         17.06.2-ce
Engine Labels:
 - provider=virtualbox
TLS Info:
 TrustRoot:
-----BEGIN CERTIFICATE-----
MIIBajCCARCgAwIBAgIUNuU4I89kxId2QXulofRKxJa9XRcwCgYIKoZIzj0EAwIw
EzERMA8GA1UEAxMIc3dhcm0tY2EwHhcNMTcwOTA4MTEzOTAwWhcNMzcwOTAzMTEz
OTAwWjATMREwDwYDVQQDEwhzd2FybS1jYTBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABEqgLUbyjyNuP35aAzW+aqVB8AkghvpF5hq1KnMveHbl4Ilr+EyDjlYZkbnt
Gb/xmsy/tOP8uz598ZX/JlR4fZyjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMB
Af8EBTADAQH/MB0GA1UdDgQWBBSzoKGrN0ELfEIMsjxuYj5LAckD2jAKBggqhkjO
PQQDAgNIADBFAiB34DOvDtIYjJ+GzbPMGu9Dd/cJGvy7CJg1tNUG3SoOrAIhAJZ4
TJBucTomFSDsj5Y/R6TfhcpXpsksk7JwYgEglu44
-----END CERTIFICATE-----

 Issuer Subject:    MBMxETAPBgNVBAMTCHN3YXJtLWNh
 Issuer Public Key:
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAESqAtRvKPI24/floDNb5qpUHwCSCG+kXmGrUqcy94duXgiWv4TIOOVhmRue0Zv/GazL+04/y7Pn3xl
f8mVHh9nA==
```

3.9 - High Availability

When a node is active, it is able to receive new tasks from the manager:

- during a scale-up,
- during a progressive upgrade,
- when another node receives an instruction to be unavailable,
- when a service fails on another node.

Remember that the swarm contains 6 Docker VMs:

```
docker@manager1:~$ docker node ls
```

```
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
y0war0lijmwhnexrfhfflulsd *   manager1            Ready               Active              Leader
19.03.12
v5ai62lmhfsdcauccqmjyu6qk     worker1             Ready               Active
19.03.12
j4mr3d2ji30t7hu0trob5dpgz     worker2             Ready               Active
19.03.12
pouo7nuvirq0qkuvvrp04a47h     worker3             Ready               Active
19.03.12
mo0dd5ech6ifdgd8pa6cjz896     worker4             Ready               Active
19.03.12
5am2vd39pybytu1nd3oooabtq     worker5             Ready               Active
19.03.12
```

And that out of the 6 Docker VMs, there are 8 containers,

```
docker@manager1:~$ docker service ps web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
son0vgc73drb        web.1               nginx:latest        worker4             Running             Running 12
minutes ago
ojqyweuo65jw        web.2               nginx:latest        worker5             Running             Running 12
minutes ago
mb40onnaxd0u        web.3               nginx:latest        manager1            Running             Running 12
minutes ago
4vwsho5x7i36        web.4               nginx:latest        worker2             Running             Running 12
minutes ago
sk9hr6j2u47c        web.5               nginx:latest        worker3             Running             Running 12
minutes ago
pui4dnkepu27        web.6               nginx:latest        worker1             Running             Running 4
minutes ago
yj0kexdcuo5u        web.7               nginx:latest        worker5             Running             Running 4
minutes ago
```

```
93jtbxqj2dyz        web.8               nginx:latest        worker1             Running             Running 4
minutes ago
```

two of which are on worker1:

```
docker@manager1:~$ docker node ps worker1
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
pui4dnkepu27        web.6               nginx:latest        worker1             Running             Running 4
minutes ago
93jtbxqj2dyz        web.8               nginx:latest        worker1             Running             Running 4
minutes ago
```

Put worker1 into unavailability mode using the **–availability drain** option:

```
docker@manager1:~$ docker node update --availability drain worker1
worker1
```

Notice that the web service has been moved to two other nodes, **manager1** and **worker4** :

```
docker@manager1:~$ docker service ps web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
son0vgc73drb        web.1               nginx:latest        worker4             Running             Running 14
minutes ago
ojqyweuo65jw        web.2               nginx:latest        worker5             Running             Running 14
minutes ago
mb40onnaxd0u        web.3               nginx:latest        manager1            Running             Running 14
minutes ago
4vwsho5x7i36        web.4               nginx:latest        worker2             Running             Running 14
minutes ago
sk9hr6j2u47c        web.5               nginx:latest        worker3             Running             Running 14
minutes ago
ag41oh489h4t        web.6               nginx:latest        manager1            Running             Running 3
```

```
seconds ago
pui4dnkepu27         \_ web.6           nginx:latest        worker1             Shutdown            Shutdown 4
seconds ago
yj0kexdcuo5u        web.7               nginx:latest        worker5             Running             Running 5
minutes ago
kv7ax6cwzpkf        web.8               nginx:latest        worker2             Running             Running 3
seconds ago
93jtbxqj2dyz         \_ web.8           nginx:latest        worker1             Shutdown            Shutdown 4
seconds ago
```

3.10 - Removing a Service

To remove a service you should use the **docker service rm** command.

```
docker@manager1:~$ docker service rm web
web

docker@manager1:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS

docker@manager1:~$ docker service inspect web
[]
Status: Error: no such service: web, Code: 1
```

3.11 - Backing up Docker Swarm

The Docker Swarm configuration is contained in the **/var/lib/docker/swarm** directory of each Manager in the Swarm :

```
docker@manager1:~$ sudo su -
root@manager1:~# ls -l /var/lib/docker/swarm
total 20
```

```
drwxr-xr-x 2 root root 4096 Dec 16 13:38 certificates
-rw------- 1 root root 215 Dec 16 13:38 docker-state.json
drwx------ 4 root root 4096 Dec 16 13:38 raft
-rw------- 1 root root 70 Dec 16 13:38 state.json
drwxr-xr-x 2 root root 4096 Dec 16 13:38 worker
```

The backup process requires there to be at least **two** Managers in the Swarm. The backup procedure is :

- stop the Docker service on the Manager to be backed up,
- backup the **/var/lib/docker/swarm** directory,
- restart the Docker service on the Manager concerned.

3.12 - Restoring Docker Swarm

The restore procedure is :

- stop the Docker service on a new Manager,
- delete the contents of the **/var/lib/docker/swarm** directory in the new Manager,
- restore the **/var/lib/docker/swarm** directory in the new Manager from the backup,
- run the **docker swarm init –force-new-cluster** command on the new Manager,
- Add Managers and Workers to Swarm.
