NASA HW4

b09902004 郭懷元

Network Administration

Short Answers

1.

> Refs:
>
> https://docs.netgate.com/pfsense/en/latest/firewall/fundamentals.html#block-vs-reject

When using `block`, the packets received are dropped silently without sending any message to the source. When using `reject`, the firewall will return some message to inform the source that the packet has been dropped.

Generally speaking, `block` is preferred on WAN settings and `reject` is preferred on LAN settings.

2.

> Refs:
>
> https://www.reddit.com/r/PFSENSE/comments/jt8be5/whats_the_difference_between_using_lan_net_and/gc42ogx/

`interface net` means all addresses in the same subnet, and `interface address` means the address of the interface on pfsense. For example, suppose an interface `vlan5` is on `192.168.42.1/24`, then `vlan5 net` is `192.168.42.1-255` and `vlan5 address` is `192.168.42.1`.

3.

> Refs:
>
> https://lin0204.blogspot.com/2017/01/blog-post_30.html
> https://docs.netgate.com/pfsense/en/latest/firewall/fundamentals.html#stateful-filtering

The firewall in pfsense is a *stateful firewall*. A *stateful firewall* will keep track of traffics going through, and allow expected respond packets that are not directly allowed in rules. For example, if I send a TCP request to a website, the firewall will allow the respond packet from that website.

pfSense

1.

> Refs:
>
> Lab slides

`Interfaces` -> `Assignments` -> `VLANs` -> `Add`, create one vlan with tag `5` and one with tag `99`.

Go to `Interface Assignments` to add those two vlan interfaces.

`Interfaces` -> `OPT1`, and do the following configs:

- **Enable**: check the box
- **Description**: `VLAN5`
- **IPv4 Configuration Type**: `Static IPv4`
- **IPv4 Address**: `10.5.255.254/16`

`Interfaces` -> `OPT2`, and do the following configs:

- **Enable**: check the box
- **Description**: `VLAN99`
- **IPv4 Configuration Type**: `Static IPv4`
- **IPv4 Address**: `192.168.99.254/24`

`Services` -> `DHCP Server` -> `VLAN5`, and do the following configs:

- **Enable**: check the box
- **Range**: From `10.5.0.1` to `10.5.255.253`
- **DNS Servers**: `8.8.8.8`, `8.8.4.4`

`Services` -> `DHCP Server` -> `VLAN99`, and do the following configs:

- **Enable**: check the box
- **Range**: From `192.168.99.1` to `192.168.99.253`
- **DNS Servers**: `8.8.8.8`, `8.8.4.4`

2.

> Refs:
>
> https://forums.serverbuilds.net/t/guide-aliases-in-pfsense/5777

`Firewall` -> `Aliases`

Add one entry with the following configs:

- **Name**: `GOOGLE_DNS`
- **Type**: `Host`
- **IP or FQDN**: `8.8.8.8`, `8.8.4.4`

Add one entry with the following configs:

- **Name**: `ADMIN_PORTS`
- **Type**: `Port`
- **Port**: `22`, `80`, `443`

Add one entry with the following configs:

- **Name**: `CSIE_WORKSTATIONS`
- **Type**: `Host`
- **IP or FQDN**: `linux1.csie.org`, `linux2.csie.org`, `linux3.csie.org`, `linux4.csie.org`, `linux5.csie.org`

3.

> Refs:
>
> https://blog.51cto.com/fxn2025/1943916

`System` -> `Advanced` -> navigate to `Secure Shell`

Check the box for enabling ssh

`Firewall` -> `Rules` -> `VLAN99`

Add a new entry with the these config:

- **Action**: `Pass`
- **Interface**: `VLAN99`
- **Address Family**: `IPv4`
- **Protocol**: `TCP`
- **Source**: `VLAN99 net`
- **Destination**: `VLAN99 Address`
- **Destination Port Range**: `ADMIN_PORTS`

4.

> Refs:
>
> None

`Firewall` -> `Rules` -> `VLAN99`

Add some entries with these configs:

- Entry 1
  - **Action**: `Pass`
  - **Interface**: `VLAN99`
  - **Address Family**: `IPv4`
  - **Protocol**: `Any`
  - **Source**: `VLAN99 net`
  - **Destination**: `VLAN5 net`
- Entry 2
  - **Action**: `Pass`
  - **Interface**: `VLAN99`
  - **Address Family**: `IPv4`
  - **Protocol**: `Any`
  - **Source**: `VLAN99 net`
  - **Destination**: `Single host or alias`, `GOOGLE_DNS`
- Entry 3
  - **Action**: `Pass`
  - **Interface**: `VLAN99`
  - **Address Family**: `IPv4`
  - **Protocol**: `Any`
  - **Source**: `VLAN99 net`
  - **Destination**: `Single host or alias`, `CSIE_WORKSTATIONS`
- Entry 4
  - **Action**: `Block`
  - **Interface**: `VLAN99`
  - **Address Family**: `IPv4`
  - **Protocol**: `Any`
  - **Source**: `VLAN99 net`
  - **Destination**: `any`

And put entry 4 at the bottom.

5.

> Refs:
>
> https://www.reddit.com/r/PFSENSE/comments/7srwxc/question_about_multiple_interfaces_and_firewall/
>
> https://docs.netgate.com/pfsense/en/latest/firewall/floating-rules.html

`Firewall` -> `Rules` -> `Floating`

add an entry with these configs:

- **Action**: `Block`
- **Interface**: `WAN`, `LAN`, `VLAN5`, `VLAN99`
- **Direction**: `any`
- **Address Family**: `IPv4`
- **Protocol**: `any`

- **Source**: invert match `VLAN99 net`
- **Destination**: `VLAN99 net`

6.

> Refs:
>
> https://docs.netgate.com/pfsense/en/latest/firewall/time-based-rules.html

`Firewall` -> `Schedules`

add an entry like this:

- **Schedule Name**: `block_VLAN5`
- **Month**: `May_21`
- **Date**: `11`
- **Time**: `0:00` ~ `23:59`
- click `add time`

`Firewall` -> `Rules` -> `VLAN5`

add an entry like this:

- **Action**: `Block`
- **Interface**: `VLAN5`
- **Address Family**: `IPv4`
- **Protocol**: `Any`
- **Source**: `any`
- **Destination**: `any`
- click `Display Advanced`
- **Schedule**: `block_VLAN5`

7.

> Refs:
>
> None

`Firewall` -> `Rules` -> `VLAN5`

add an entry to the bottom with these configs:

- **Action**: `Pass`
- **Interface**: `VLAN5`
- **Address Family**: `IPv4`
- **Protocol**: `Any`
- **Source**: `VLAN5 net`
- **Destination**: `any`

8.

> Refs:
>
> None

`Diagnostics` -> `Backup & Restore`

System Administration

1. 關於 Container

> Refs:
>
> https://medium.com/@jinghua.shih/container-%E6%A6%82%E5%BF%B5%E7%AD%86%E8%A8%98-b0963ae2d7c6
> https://ithelp.ithome.com.tw/articles/10216215
> https://ithelp.ithome.com.tw/articles/10218127
> https://ithelp.ithome.com.tw/articles/10219102
> https://computingforgeeks.com/docker-vs-cri-o-vs-containerd/
> https://www.tutorialworks.com/difference-docker-containerd-runc-crio-oci/
> https://thenewstack.io/a-security-comparison-of-docker-cri-o-and-containerd/
> https://medium.com/@xroms123/docker-%E5%BB%BA%E7%AB%8B-nginx-%E5%9F%BA%E7%A4%8E%E5%88%86%E4%BA%AB-68c0771457fb

1.

When to use containers

- A web backend environment that uses specific versions of Python, MySQL and Node.js.
- An environment packed with your application to avoid any dependency issues.
- An environment for students to practice programming without worrying compiler version issues
- An web server environment

When to use VMs instead of containers

- Playing with malwares and virus
- Testing applications on a different OS
- Specifying hardware resources you want to use

2.

OCI is a project that design and maintain specifications, about how different solutions of container should create and run containers. CRI is an interface between a container-orchestration system (like `Kubernetes`) and a container runtime (like `Docker`).

`Docker` runs containers with OCI specs, and interacts with system `Kubernetes` through CRI.

3.

`CRI-O` is a lightweight container runtime that is designed to work with `Kubernetes`. It provides only the necessary services to run a container and reduces excessive inter-process communications that other solutions might have.

`CRI-O` vs `Docker`

Common

- Uses `runC` at the bottom level
- Can be used with `Kubernetes`
- Open Source

Differences

- `CRI-O` directly uses `runC`. But `Docker Engine` calls `containerd` then `containerd` calls `runC`.
- `CRI-O` directly talks to `Kubernetes` through CRI, but `Docker Engine` requires `Dockershim` (deprecated now).
- `CRI-O` removes many linux capabilities such as SSH, but `Docker` keeps them.

4.

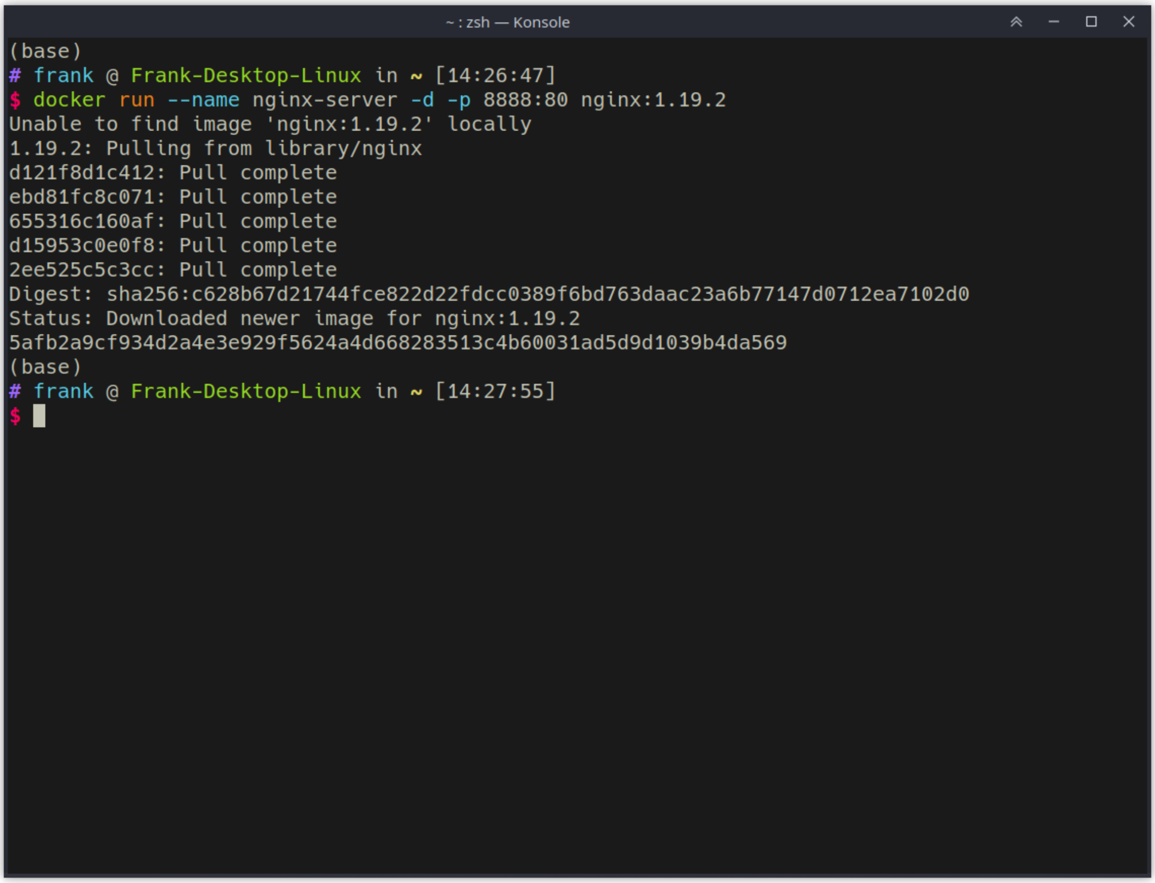
```
docker run --name nginx-server -d -p 8888:80 nginx:1.19.2
```

`–name nginx-server` set the name of this container.

`-d` means run the container in background and print container ID.

`-p 8888:80` means we forward local port `8888` to container's port `80`.

`nginx:1.19.2` is the image we are using.



```
(base)
# frank @ Frank-Desktop-Linux in ~ [14:26:47]
$ docker run --name nginx-server -d -p 8888:80 nginx:1.19.2
Unable to find image 'nginx:1.19.2' locally
1.19.2: Pulling from library/nginx
d121f8d1c412: Pull complete
ebd81fc8c071: Pull complete
655316c160af: Pull complete
d15953c0e0f8: Pull complete
2ee525c5c3cc: Pull complete
Digest: sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
Status: Downloaded newer image for nginx:1.19.2
5afb2a9cf934d2a4e3e929f5624a4d668283513c4b60031ad5d9d1039b4da569
(base)
# frank @ Frank-Desktop-Linux in ~ [14:27:55]
$ 
```

2. Docker Basics

> Refs:
>
> https://www.codenotary.com/blog/extremely-useful-docker-commands/
> https://docs.docker.com/engine/reference/commandline/system_prune/
> https://stackoverflow.com/questions/17157721/how-to-get-a-docker-containers-ip-address-from-the-host
> https://docs.docker.com/engine/reference/commandline/inspect/
> https://docs.docker.com/engine/reference/commandline/stats/
> https://docs.docker.com/config/containers/container-networking/
> https://docs.docker.com/engine/reference/commandline/exec/

1.

```
docker kill $(docker ps -q)
```

`docker ps -q` lists all container IDs. `docker kill <container id>` stops the container.

2.

```
docker rmi $(docker images -q)
```

`docker images -q` lists all image IDs. `docker rmi <image id>` removes the image.

3.

```
docker system prune -a --volumes
```

`-a` removes all unused resources (only dangling ones are removed by default). `--volumes` removes volumes (volumes are kept by default).

4.

```
docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
5b0f1ed0dcb8
```

`docker inspect` shows the information the container. `-f <format>` specify the output format.

5.

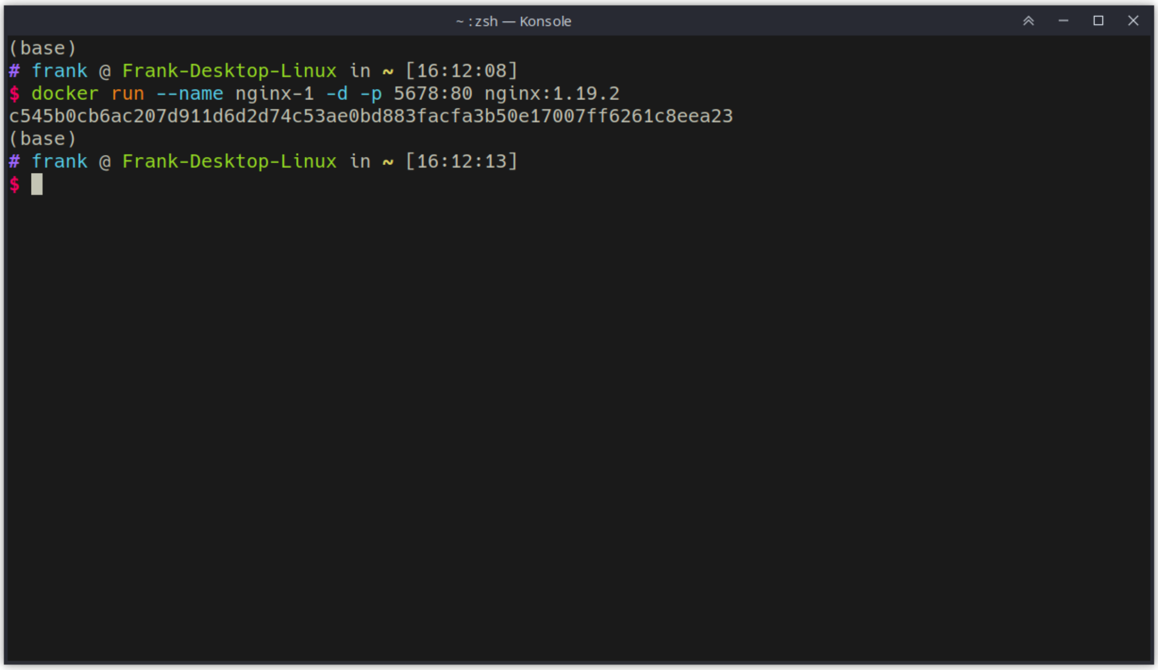
```
docker stats -a
```

`-a` shows all containers' resources usage (including not running ones).

6.

```
docker run --name nginx-1 -d -p 5678:80 nginx:1.19.2
```

`-p 5678:80` means we forward local port `5678` to container's port `80`.

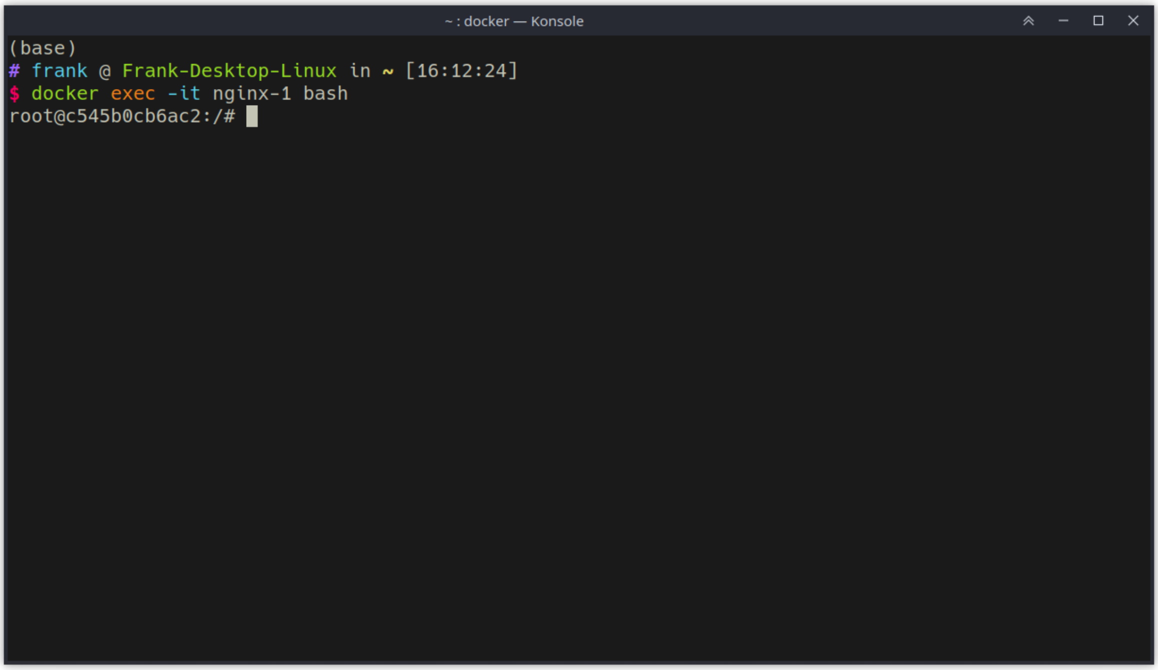


```
(base)
# frank @ Frank-Desktop-Linux in ~ [16:12:08]
$ docker run --name nginx-1 -d -p 5678:80 nginx:1.19.2
c545b0cb6ac207d911d6d2d74c53ae0bd883facfa3b50e17007ff6261c8eea23
(base)
# frank @ Frank-Desktop-Linux in ~ [16:12:13]
$ 
```

7.

```
docker exec -it nginx-1 bash
```

Executes `bash` shell in `nginx-1`.

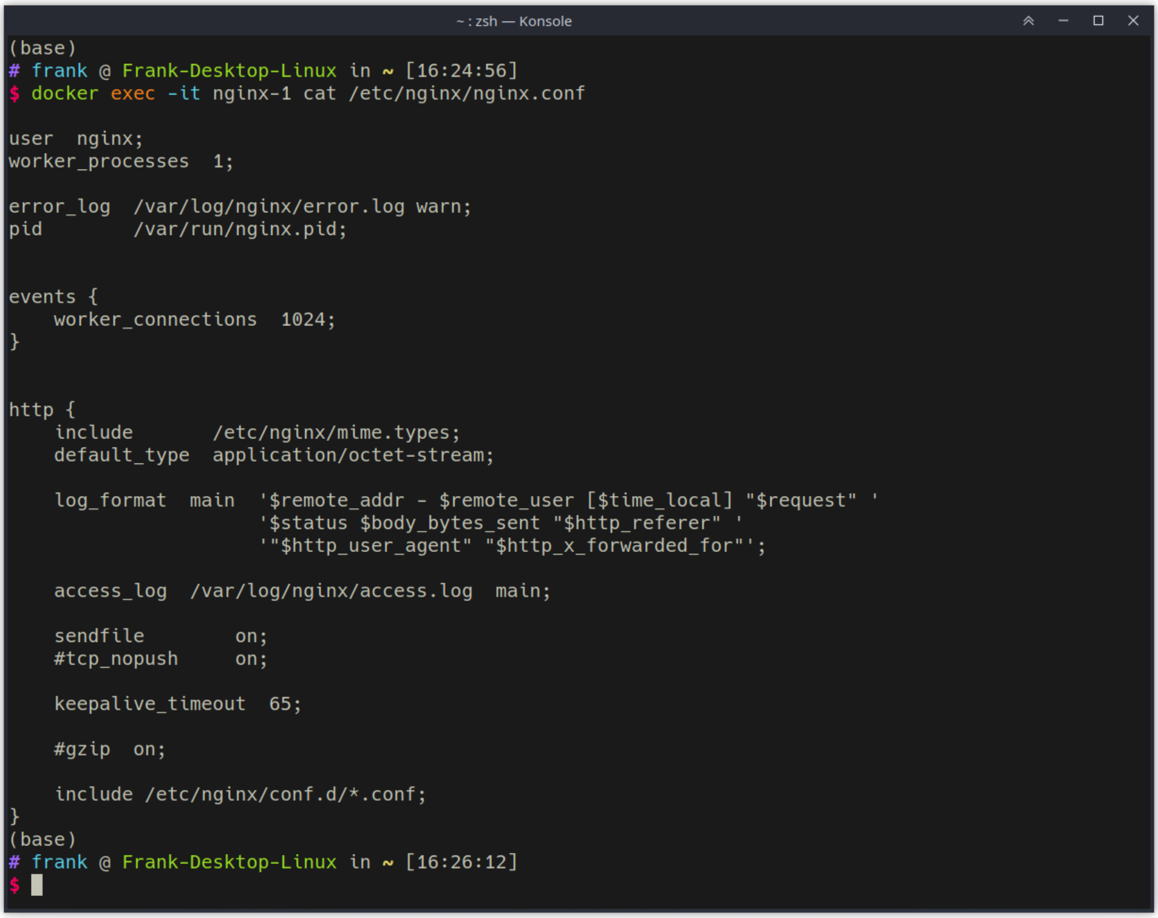


```
(base)
# frank @ Frank-Desktop-Linux in ~ [16:12:24]
$ docker exec -it nginx-1 bash
root@c545b0cb6ac2:/# 
```

8.

```
docker exec -it nginx-1 cat /etc/nginx/nginx.conf
```

Usage is `docker exec -it <container name> <command>`. So just put `cat /etc/nginx/nginx.conf` in the `<command>` part.



```
(base)
# frank @ Frank-Desktop-Linux in ~ [16:24:56]
$ docker exec -it nginx-1 cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
(base)
# frank @ Frank-Desktop-Linux in ~ [16:26:12]
$ 
```

---

3. Docker Network

> Refs:
>
> https://docs.docker.com/network/
> https://ithelp.ithome.com.tw/articles/10193457
> https://docs.docker.com/network/bridge/#manage-a-user-defined-bridge
> https://nickjanetakis.com/blog/docker-tip-65-get-your-docker-hosts-ip-address-from-in-a-container

1.

Docker network

- `bridge`

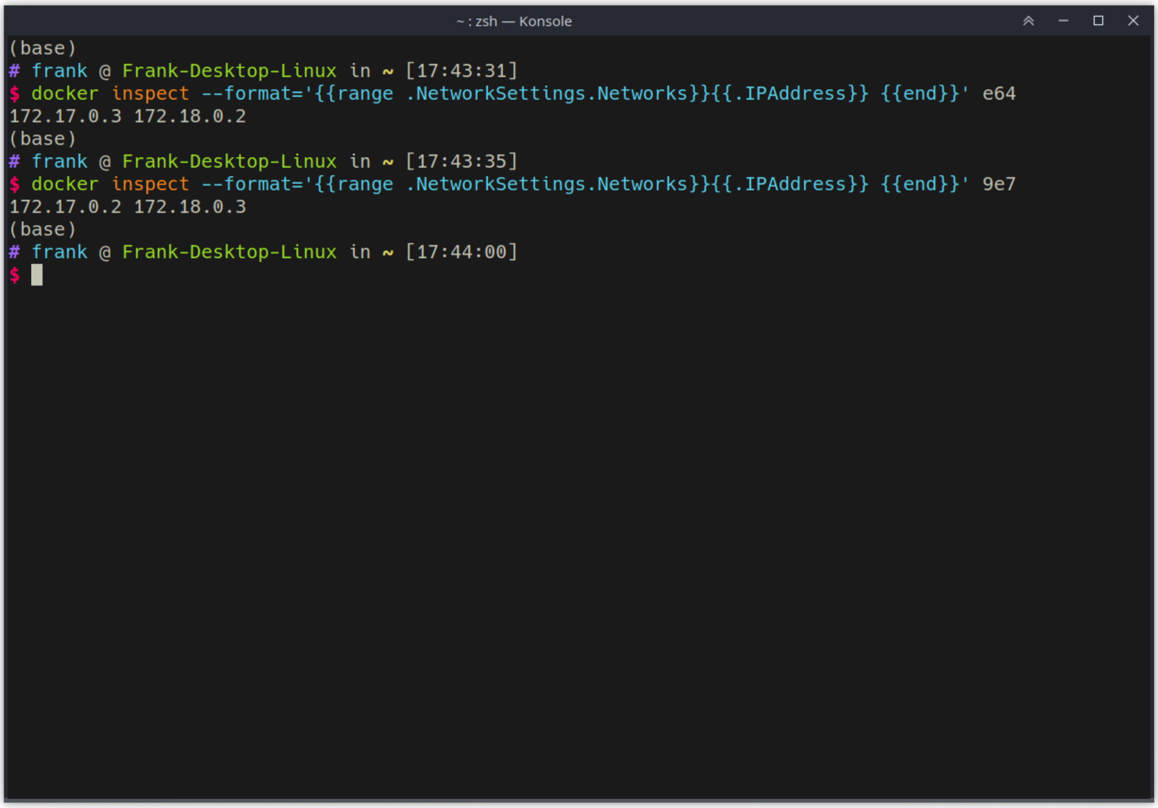
    - Kind of like NAT in VM network settings. Each container will be isolated can can communicate to other containers.
    - Use case: When you have multiple containers like web servers on one Docker host, and you want them to communicate with each other.
- `host`
    - Using host machine's network directly.
    - Use case: Testing software under host's network configs in a isolated environment.
- `overlay`
    - Allowing containers on different Docker hosts to communicate.
    - Use case: Two people can have their container running on each person's own machine and communicate .
- `macvlan`
    - Assign MAC address to the container, making it appears to be a physical machine. Also provides a more VM-like environment.
    - Use case: When running applications that requires or expects to be physically connected to a network.
- `none`
    - Disable all network settings.
    - Use case: Using a custom network driver for the container.

2.

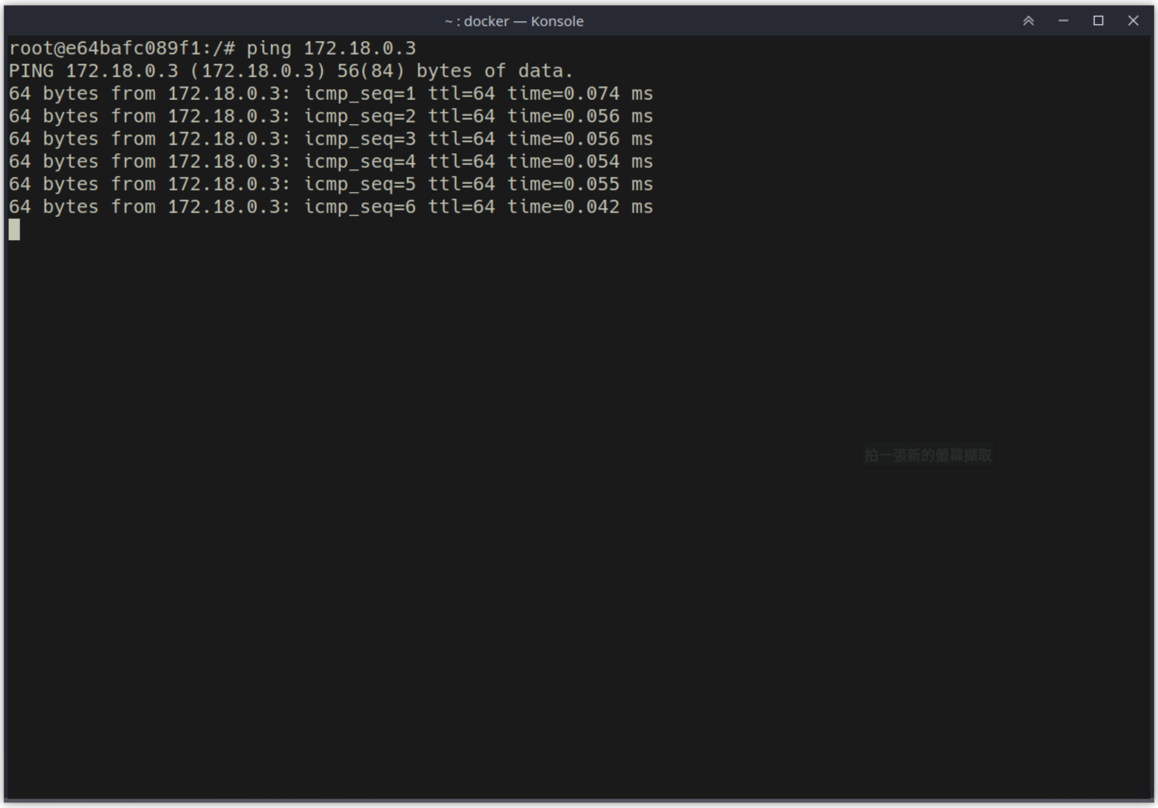
```
docker run --name nginx-2 -d nginx:1.19.2
docker network create nasa-net
docker network connect nasa-net nginx-1
docker network connect nasa-net nginx-2
```

`docker network create` creates a user-defined bridge.

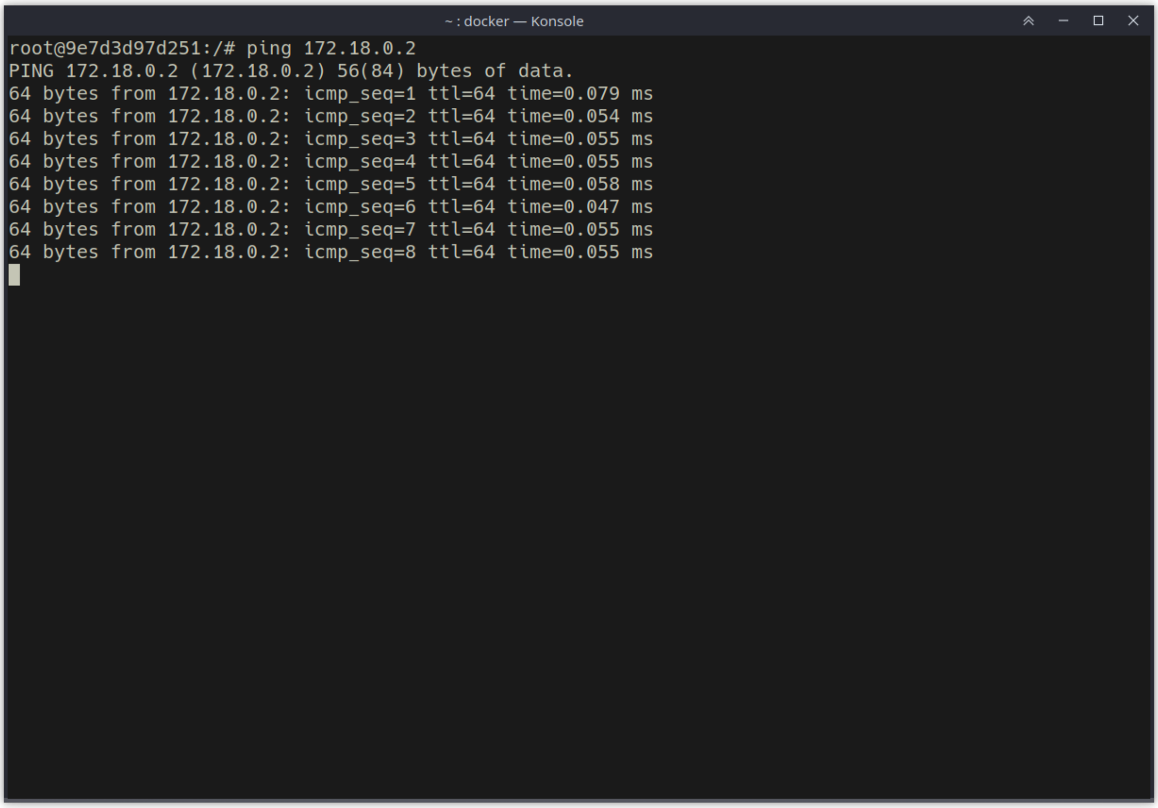
`docker network connect` connects a container to a bridge.



```
(base)
# frank @ Frank-Desktop-Linux in ~ [17:43:31]
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}} {{end}}' e64
172.17.0.3 172.18.0.2
(base)
# frank @ Frank-Desktop-Linux in ~ [17:43:35]
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}} {{end}}' 9e7
172.17.0.2 172.18.0.3
(base)
# frank @ Frank-Desktop-Linux in ~ [17:44:00]
$ 
```



```
root@e64bafc089f1:/# ping 172.18.0.3
PING 172.18.0.3 (172.18.0.3) 56(84) bytes of data.
64 bytes from 172.18.0.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 172.18.0.3: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 172.18.0.3: icmp_seq=3 ttl=64 time=0.056 ms
64 bytes from 172.18.0.3: icmp_seq=4 ttl=64 time=0.054 ms
64 bytes from 172.18.0.3: icmp_seq=5 ttl=64 time=0.055 ms
64 bytes from 172.18.0.3: icmp_seq=6 ttl=64 time=0.042 ms
```

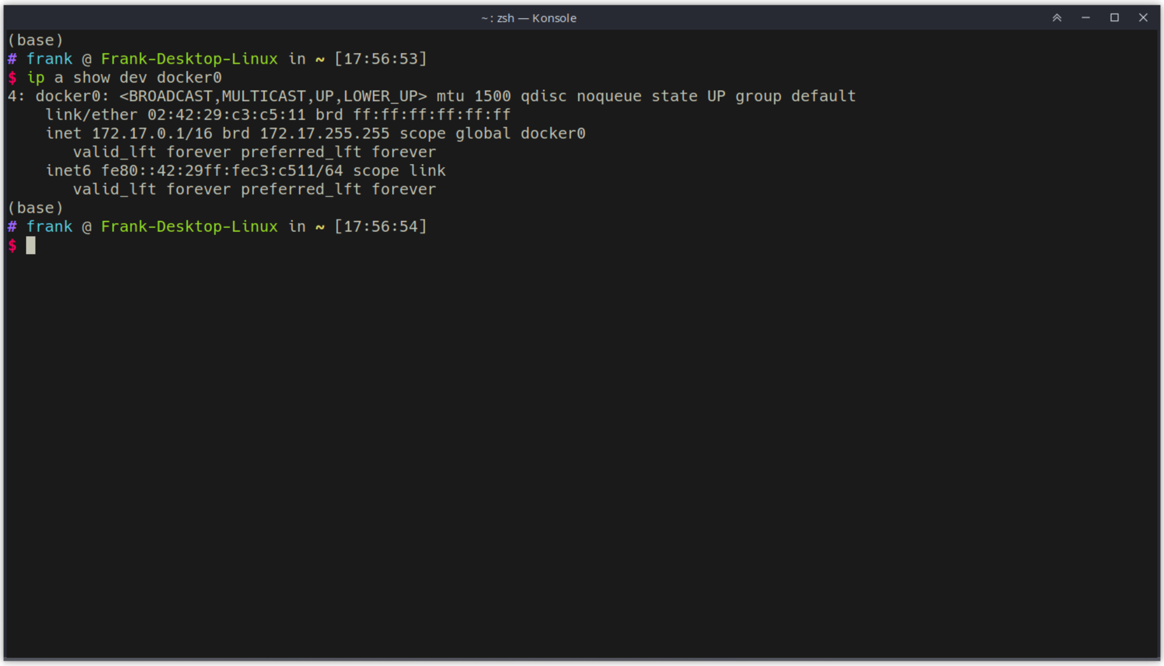


```
root@9e7d3d97d251:/# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.079 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from 172.18.0.2: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 172.18.0.2: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 172.18.0.2: icmp_seq=5 ttl=64 time=0.058 ms
64 bytes from 172.18.0.2: icmp_seq=6 ttl=64 time=0.047 ms
64 bytes from 172.18.0.2: icmp_seq=7 ttl=64 time=0.055 ms
64 bytes from 172.18.0.2: icmp_seq=8 ttl=64 time=0.055 ms
```

3.

```
ip a show dev docker0
```

Because I am running Docker directly on linux, a network adapter `docker0` will be added. We can use `ip a show dev <device name>` to see it's info.



```
(base)
# frank @ Frank-Desktop-Linux in ~ [17:56:53]
$ ip a show dev docker0
4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:29:c3:c5:11 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:29ff:fec3:c511/64 scope link
       valid_lft forever preferred_lft forever
(base)
# frank @ Frank-Desktop-Linux in ~ [17:56:54]
$
```

4. Build Application

> Refs:
>
> https://www.ctl.io/developers/blog/post/dockerfile-entrypoint-vs-cmd/
> https://docs.docker.com/engine/reference/builder/
> https://docs.docker.com/compose/
> https://docs.docker.com/engine/reference/commandline/run/#extended-description
> https://docs.docker.com/storage/bind-mounts/
> https://docs.microsoft.com/zh-tw/visualstudio/docker/tutorials/use-docker-compose
> https://docs.docker.com/compose/networking/
> https://docs.docker.com/compose/reference/
> https://docs.docker.com/compose/reference/down/

1.

Differences:

- `ENTRYPOINT` is used when this image is an wrapped application. `CMD` is used to pass user-set arguments to `ENTRYPOINT` or execute a temporary command.
- In `docker run`, overriding `CMD` is simply appending it to the end of command. Overriding `ENTRYPOINT` requires using the flag `--entrypoint`.
- If `ENTRYPOINT` is written in `SHELL` from in the Dockerfile, any `CMD` will not take effect.

Use case: Use `CMD` to pass arguments to `ENTRYPOINT`, which executes `ping`.

```
FROM alpine:3
RUN apk update && apk add iputils
ENTRYPOINT ["/bin/ping", "-c", "5"]
CMD ["localhost"]
```

2.

`Docker Compose` is a tool to start and manage multiple docker containers as an application.

`Docker` or `Docker Engine` provides a way to start a single container.

3.

First command:

- . `-p 3000:3000` forward port 3000 on host to port 3000 on container.
- `-w /app` set working directory in the container to `/app`.
- `-v ${PWD}:/app` "bind mounts: the current working directory on your host to container's `/app`.
- `--network nasa-net` connects the container to `nasa-net` network.
- `-e MYSQL_HOST=mysql`, `-e MYSQL_USER=root`, and `-e MYSQL_PASSWORD=secret` set environment variables in the container.

- `node:12-alpine` is a `Node.js` image on alpine linux.
- `sh -c "echo helloworld"` is the `CMD` we are using.

Second command:

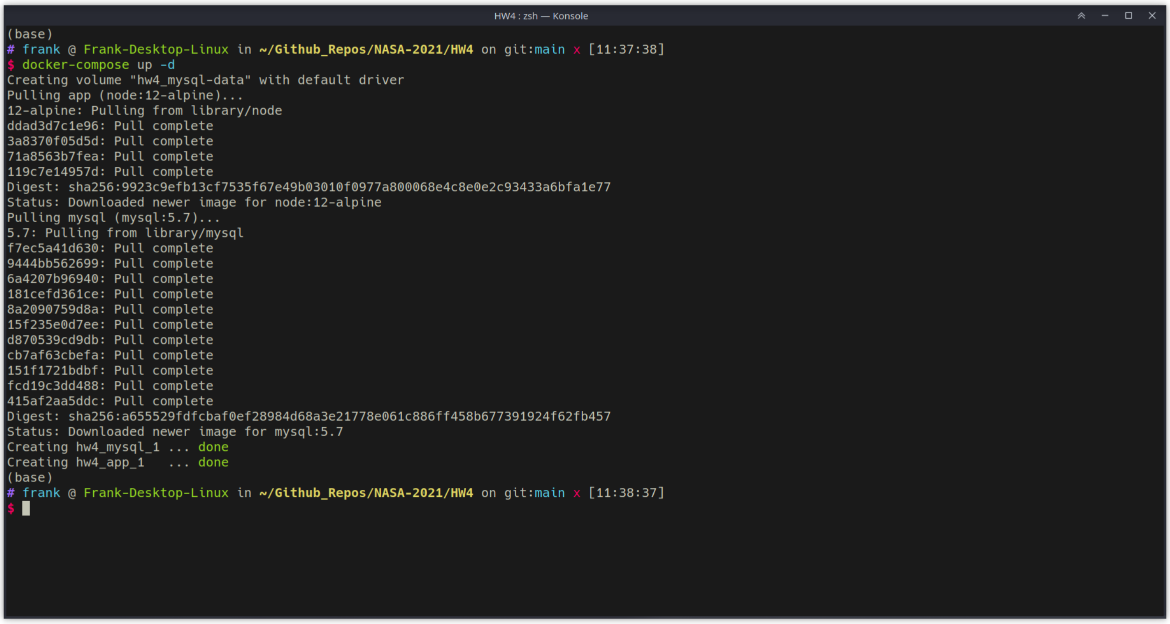
- `–network nasa-net` connects the container to `nasa-net` network.
- `-v mysql-data:/var/lib/mysql` bind mounts `mysql-data` on host to `/var/lib/mysql` on the container.
- `-e MYSQL_ROOT_PASSWORD=secret` sets environment variable in the container.
- `mysql:5.7` is a `MySQL` image.

`docker-compose.yml` :

```
version: "3.8"
services:
  app:
    image: node:12-alpine
    command: sh -c "echo helloworld"
    ports:
      - 3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USE: root
      MYSQL_PASSWORD: secret
  mysql:
    image: mysql:5.7
    volumes:
      - mysql-data:/var/liyamlb/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret

volumes:
  mysql-data:

networks:
  default:
    external:
      name: nasa-net
```



```
(base)
# frank @ Frank-Desktop-Linux in ~/Github_Repos/NASA-2021/HW4 on git:main x [11:37:38]
$ docker-compose up -d
Creating volume "hw4_mysql-data" with default driver
Pulling app (node:12-alpine)...
12-alpine: Pulling from library/node
ddad3d7c1e96: Pull complete
3a8370f05d5d: Pull complete
71a8563b7fea: Pull complete
119c7e14957d: Pull complete
Digest: sha256:9923c9efb13cf7535f67e49b03010f0977a800068e4c8e0e2c93433a6bfa1e77
Status: Downloaded newer image for node:12-alpine
Pulling mysql (mysql:5.7)...
5.7: Pulling from library/mysql
f7ec5a41d630: Pull complete
9444bb562699: Pull complete
6a4207b96940: Pull complete
181cefd361ce: Pull complete
8a2090759d8a: Pull complete
15f235e0d7ee: Pull complete
d870539cd9db: Pull complete
cb7af63cbefa: Pull complete
151f1721bdbf: Pull complete
fcd19c3dd488: Pull complete
415af2aa5ddc: Pull complete
Digest: sha256:a655529fdfcbaf0ef28984d68a3e21778e061c886ff458b677391924f62fb457
Status: Downloaded newer image for mysql:5.7
Creating hw4_mysql_1 ... done
Creating hw4_app_1   ... done
(base)
# frank @ Frank-Desktop-Linux in ~/Github_Repos/NASA-2021/HW4 on git:main x [11:38:37]
$ 
```

4.

(a)

```
docker-compose up -d
```

(b)

```
docker-compose pause
```

(c)

```
docker-compose down -v
```

`-v` is added to remove volumes. External networks and volumes won't be removed.

5. Docker in Docker

> Refs:
>
> https://docs.docker.com/engine/install/ubuntu/
> https://itnext.io/docker-in-docker-521958d34efd
> https://ithelp.ithome.com.tw/articles/10191139

1.

```
FROM ubuntu:18.04
RUN apt update
RUN apt install -y apt-transport-https ca-certificates curl gnupg lsb-release
RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
RUN echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
RUN apt update
RUN apt install -y docker-ce docker-ce-cli containerd.io
CMD docker run hello-world
```

2.

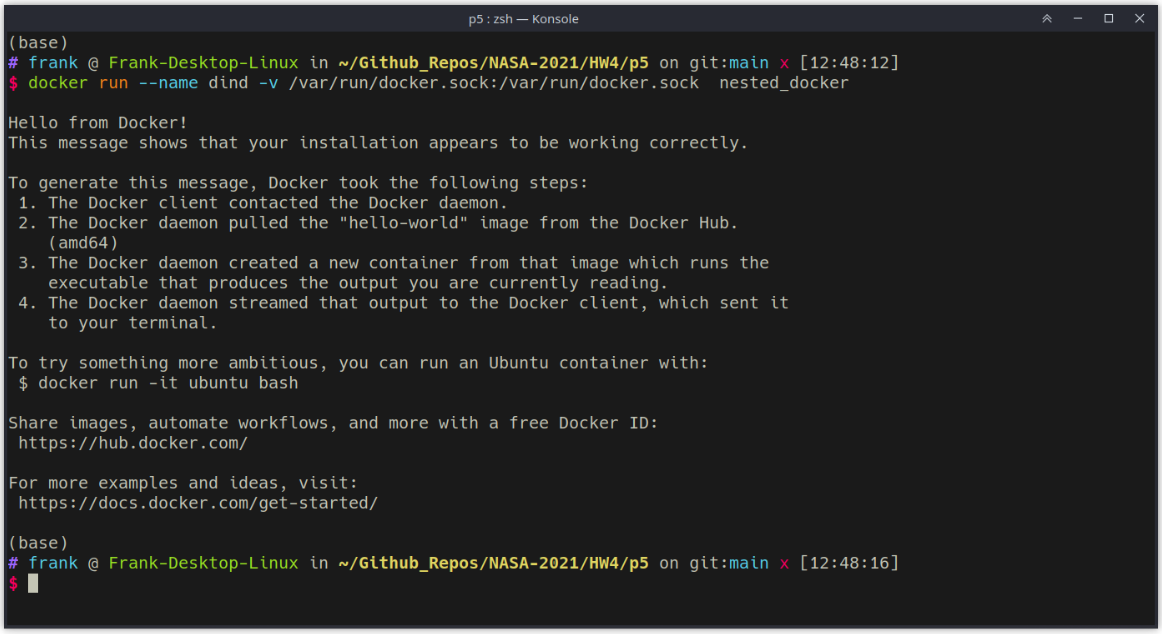
```
docker build -t nested_docker .
```

`-t` is used to name the image. `.` is the path to `Dockerfile`.

3.

```
docker run --name dind -v /var/run/docker.sock:/var/run/docker.sock  nested_docker
```

Due to some low-level issues and how docker is implemented, running an completely isolated docker inside a docker container might requires some nasty hacks. However, in most cases we don't necessary need an completely isolated docker engine. If we expose docker socket of the outer docker to the inner (`-v /var/run/docker.sock:/var/run/docker.sock`), we would still be able to use docker inside a container.



```
(base)
# frank @ Frank-Desktop-Linux in ~/Github_Repos/NASA-2021/HW4/p5 on git:main x [12:48:12]
$ docker run --name dind -v /var/run/docker.sock:/var/run/docker.sock  nested_docker

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

(base)
# frank @ Frank-Desktop-Linux in ~/Github_Repos/NASA-2021/HW4/p5 on git:main x [12:48:16]
$ 
```

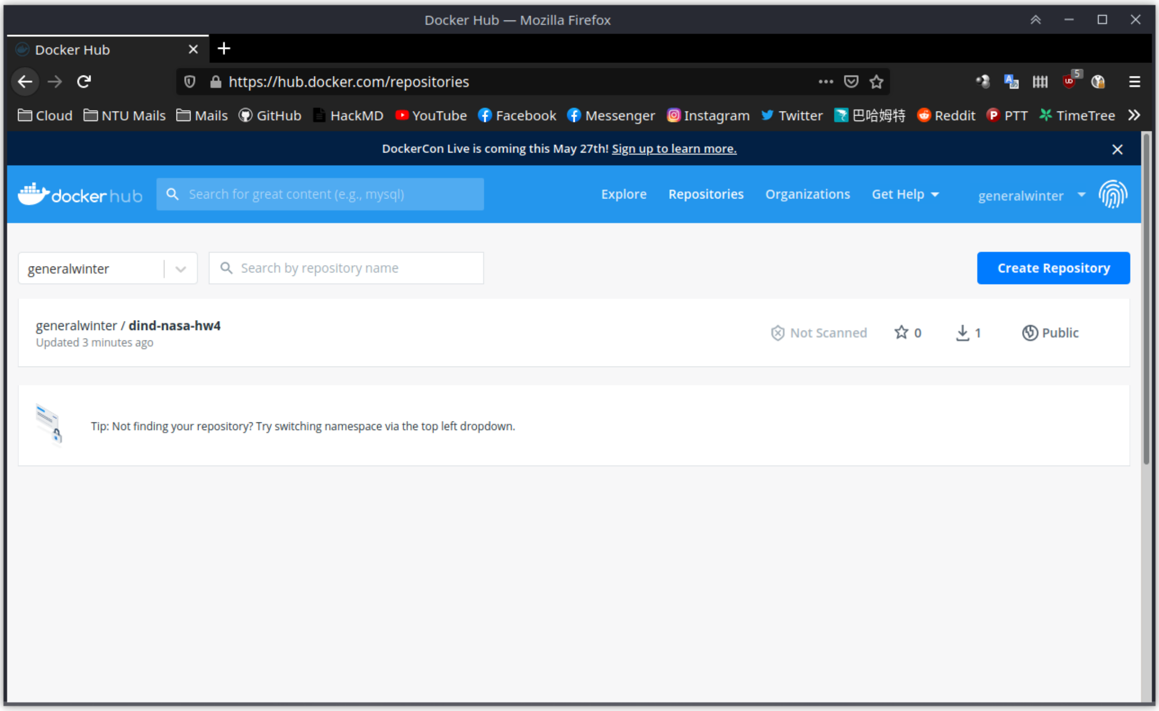
4.

```
docker login
docker tag nested_docker generalwinter/dind-nasa-hw4:v1.0.0
docker push generalwinter/dind-nasa-hw4:v1.0.0
```

`docker login` to login the account that you will push your image to.

`docker tag <original image name> <account name>/<upload image name>:<tag>`

`docker push <account name>/<upload image name>:<tag>`



DockerCon Live is coming this May 27th! Sign up to learn more.

generalwinter

generalwinter / dind-nasa-hw4
Updated 3 minutes ago

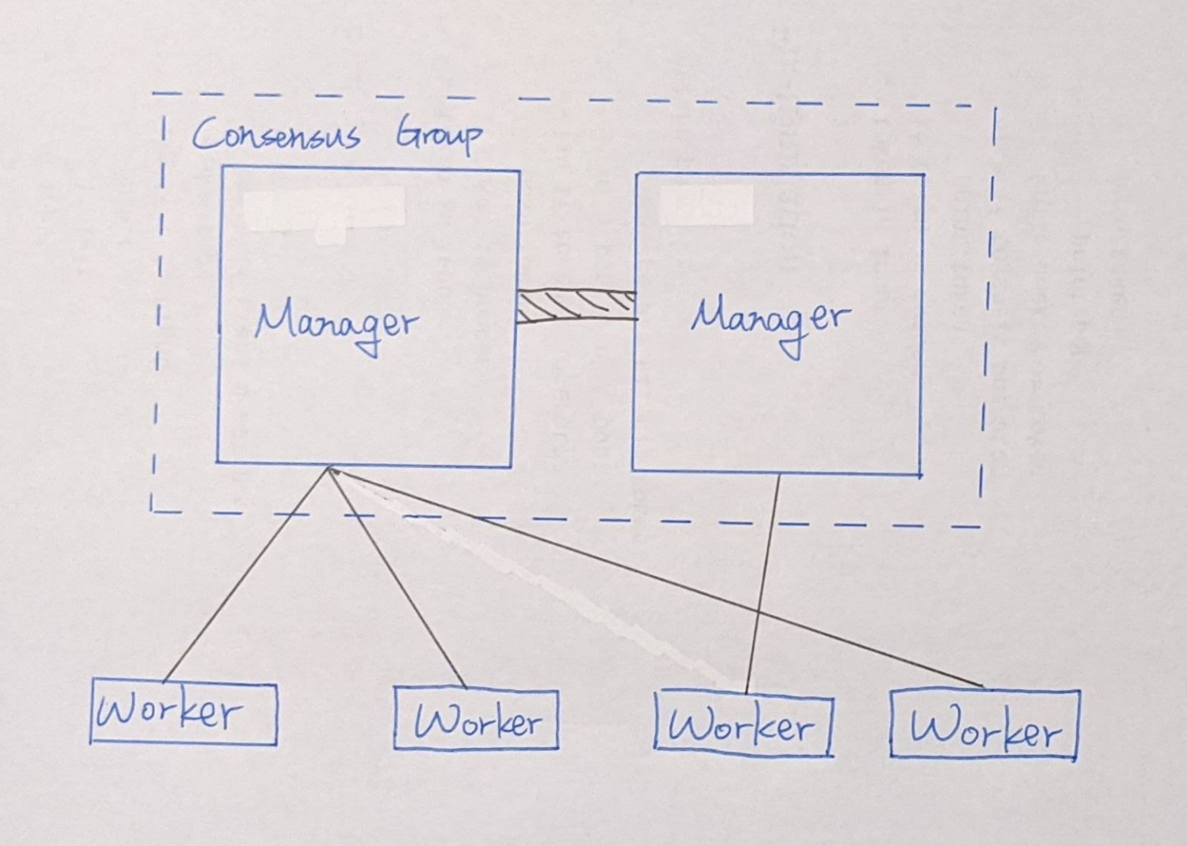
Not Scanned ☆ 0 ⤓ 1 Public

Tip: Not finding your repository? Try switching namespace via the top left dropdown.

6. Docker & Distributed System

1.

> Refs:
>
> https://github.com/twtrubiks/docker-swarm-tutorial
> https://columns.chicken-house.net/2017/12/31/microservice9-servicediscovery/
> https://web.archive.org/web/20200612023642if_/https://success.docker.com/article/networking#swarmnativeservicediscovery



A Docker Swarm is constructed with nodes, and there are two types of nodes, `Manager` and `Worker`. A `Manager` node deploys tasks to `Worker` nodes. When there are multiple `Manager` nodes, one of them would be the "leader" and other nodes will follow the leader. A `Worker` node receives tasks, do them, and tell `Manager` its status.

Service discovery in Docker Swarm is done with the DNS server embed in the Docker Engine. Since containers are started with Docker, the engine can easily keep track of containers and update its DNS table for internal services. Queries are done by sending DNS query to engine's DNS server.

2.

> Refs:

> https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/
> https://github.com/twtrubiks/docker-swarm-tutorial
> https://docs.genesys.com/Documentation/System/latest/DDG/InstallationofDockeronAlpineLinux
> https://docs.docker.com/engine/swarm/manage-nodes/#add-or-remove-label-metadata
> https://docs.docker.com/engine/swarm/stack-deploy/
> https://docs.docker.com/compose/compose-file/compose-file-v3/

(a)

I choose alpine linux as the os for vm. Some special steps to install Docker on alpine are:

```
# before install
vi /etc/apk/repositories # uncomment the url for community repositories
# after install
rc-update add docker boot # start docker at boot
service docker start # manually start docker
```

(b)

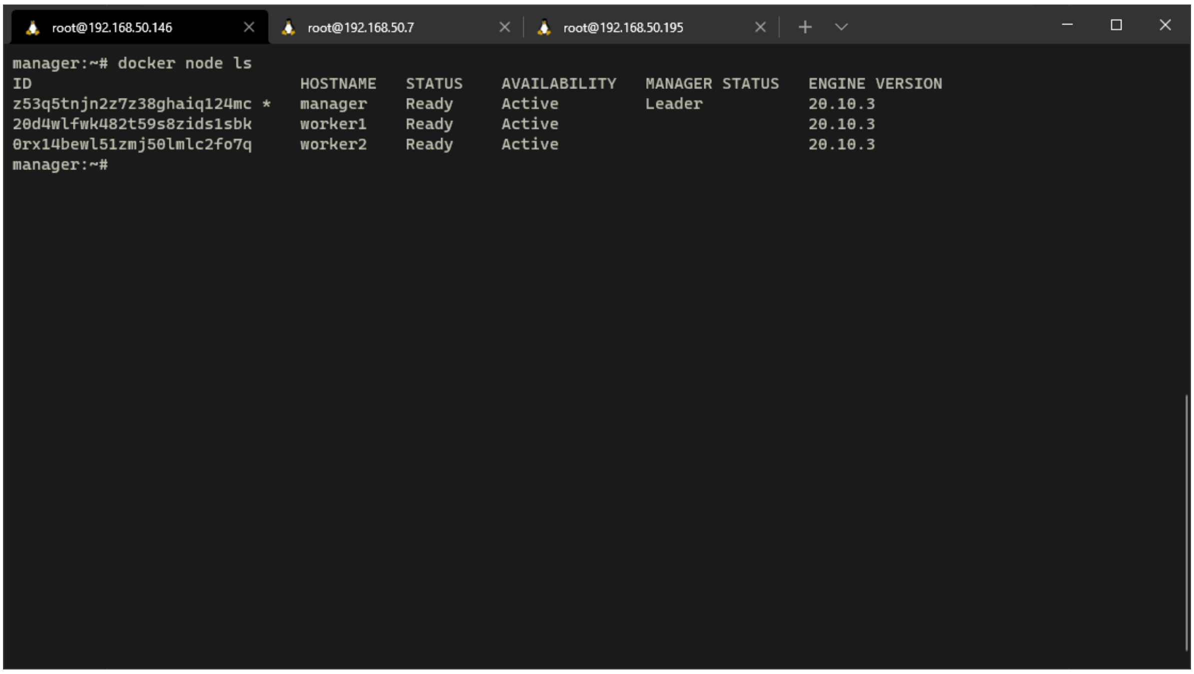
On manager vm:

```
docker swarm init --advertise-addr 192.168.50.146
```

`192.168.50.146` is the IP address of my manager vm. Docker will prompt some information after this command, and the command for joining as a worker would be shown.

On worker vms

```
docker swarm join --token SWMTKN-1-
0j4x20nk85imkbx005ry77uy1e3pksmjz9gl9wrgr8crmed76z-9yw549tc77iw8dv7f1kb7uk9y
192.168.50.146:2377
```

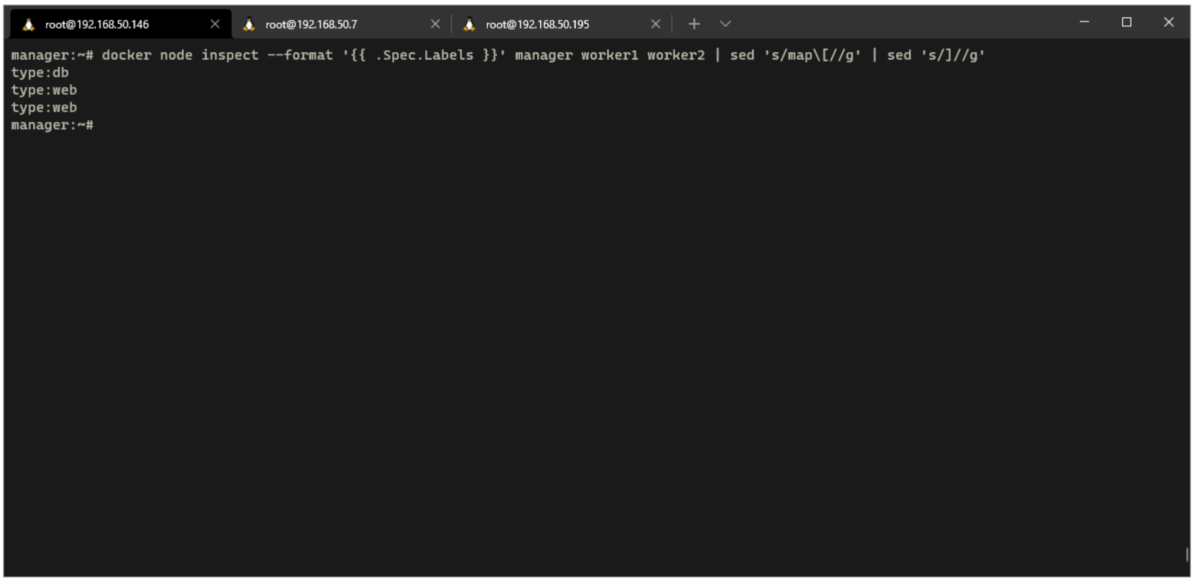


```
manager:~# docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
z53q5tnjn2z7z38ghaiq124mc *   manager    Ready     Active         Leader           20.10.3
20d4wlfwk482t59s8zids1sbk     worker1    Ready     Active                          20.10.3
0rx14bewl51zmj50lmlc2fo7q     worker2    Ready     Active                          20.10.3
manager:~#
```

(c)

Add labels with:

```
docker node update --label-add <label name> <node name>
```



```
manager:~# docker node inspect --format '{{ .Spec.Labels }}' manager worker1 worker2 | sed 's/map\[//g' | sed 's/]//g'
type:db
type:web
type:web
manager:~#
```

(d)

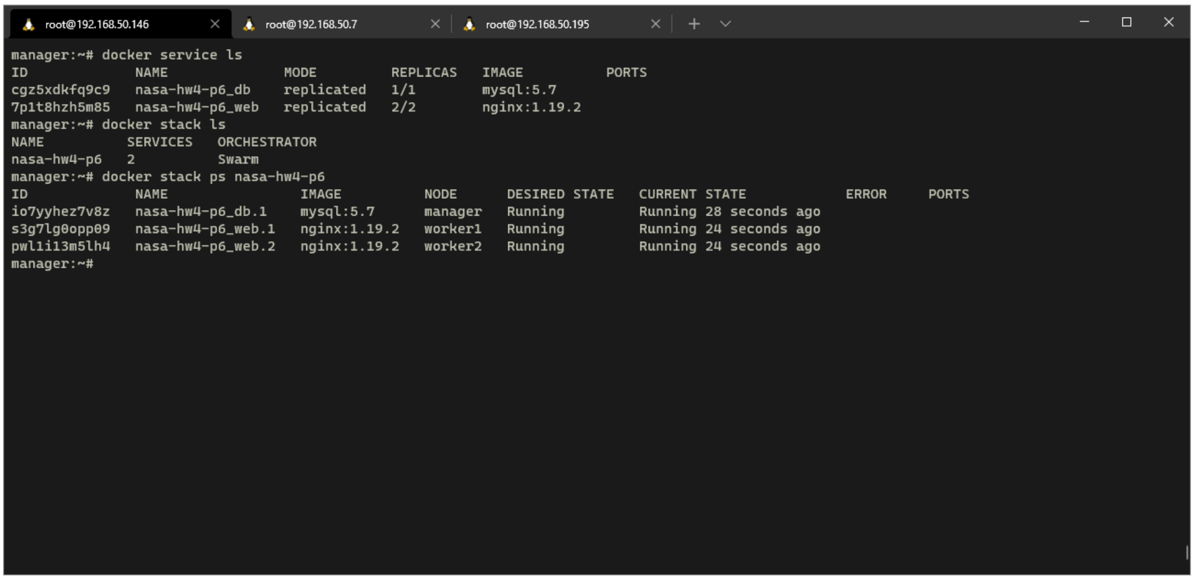
`docker-compose.yml`

```
version: "3.8"
services:
  db:
    image: mysql:5.7
    volumes:
      - /data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
    deploy:
      mode: replicated
      placement:
        constraints: [node.labels.type == db]
      replicas: 1

  web:
    image: nginx:1.19.2
    deploy:
      mode: replicated
      placement:
        constraints: [node.labels.type == web]
      replicas: 2
```

On manager vm:

```
# docker-compose dependency
apk add py-pip python3-dev libffi-dev openssl-dev gcc libc-dev rust cargo make
pip install docker-compose
# test docker-compose file before deploy
docker-compose up -d
docker-compose down
# deploy to swarm
docker stack deploy --compose-file docker-compose.yml nasa-hw4-p6
```



```
manager:~# docker service ls
ID             NAME              MODE         REPLICAS   IMAGE          PORTS
cgz5xdkfq9c9   nasa-hw4-p6_db    replicated   1/1        mysql:5.7
7p1t8hzh5m85   nasa-hw4-p6_web   replicated   2/2        nginx:1.19.2
manager:~# docker stack ls
NAME          SERVICES   ORCHESTRATOR
nasa-hw4-p6   2          Swarm
manager:~# docker stack ps nasa-hw4-p6
ID             NAME                IMAGE          NODE      DESIRED STATE   CURRENT STATE            ERROR     PORTS
io7yyhez7v8z   nasa-hw4-p6_db.1    mysql:5.7      manager   Running         Running 28 seconds ago
s3g7lg0opp09   nasa-hw4-p6_web.1   nginx:1.19.2   worker1   Running         Running 24 seconds ago
pwl1i13m5lh4   nasa-hw4-p6_web.2   nginx:1.19.2   worker2   Running         Running 24 seconds ago
manager:~#
```