

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 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 .

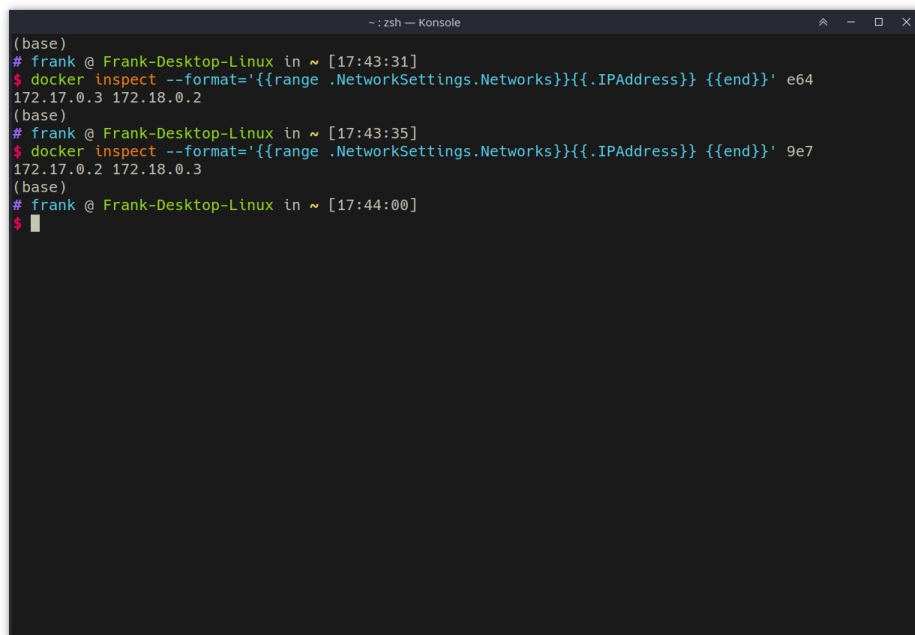
- `macvlan`
 - Assign MAC address to the container, making it appears to be a physical machine. Also provides a more VM-like environment.
 - Use case: When running applications that requires or expects to be physically connected to a network.
 - `none`
 - Disable all network settings.
 - Use case: Using 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/>

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`".
- `-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.

```
dreamdream@nasa [~] docker run -p 3000:3000 \ #
-w /app -v ${PWD}:/app \
--network nasa-net \
-e MYSQL_HOST=mysql \
-e MYSQL_USER=root \
-e MYSQL_PASSWORD=secret \
node:12-alpine \
sh -c "echo helloworld"

dreamdream@nasa [~] docker run \
--network nasa-net \
-v mysql-data:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=secret \
mysql:5.7
```