Part 3.
Inputs/Outputs

- Data volumes (persistent data)
  - mounted from the host filesystem
  - named volumes (internal + volume plugins)
- Devices
- Links
- Publishing ports (NAT)

`docker run` — mount external volumes

```
docker run -v /hostpath:/ctrpath[:ro] ...
docker run
    --mount type=bind,src=/hostpath,dst=/ctrpath[,ro] ...
```

mounts the location */hostpath* from the host filesystem at the location */ctrpath* inside the container

With the "`ro`" option, the mount is read-only

Purposes:

- store persistent data outside the container
- provide inputs: data, config files, ... (read-only mode)
- inter-process communication (unix sockets, named pipes)

Note: `-v` creates */ctrpath* automatically, `--mount` does not

mount examples (1/2)

Persistent data

```
$ docker run --rm -t -i -v /tmp/persistent:/persistent debian
root@0aeedfeb7bf9:/# echo "blahblah" >/persistent/foo
root@0aeedfeb7bf9:/# exit
$ cat /tmp/persistent/foo
blahblah
$ docker run --rm -t -i -v /tmp/persistent:/persistent debian
root@6c8ed008c041:/# cat /persistent/foo
blahblah
```

Inputs (read-only volume)

```
$ mkdir /tmp/inputs
$ echo hello > /tmp/inputs/bar
$ docker run --rm -t -i -v /tmp/inputs:/inputs:ro debian
root@05168a0eb322:/# cat /inputs/bar
hello
root@05168a0eb322:/# touch /inputs/foo
touch: cannot touch `/inputs/foo': Read-only file system
```

mount examples (2/2)

Named pipe

```
$ mkfifo /tmp/fifo
$ docker run -d -v /tmp/fifo:/fifo debian sh -c 'echo blah blah> /fifo'
ff0e44c25e10d516ce947eae9168060ee25c2a906f62d63d9c26a154b6415939
$ cat /tmp/fifo
blah blah
```

Unix socket

```
$ docker run --rm -t -i -v /dev/log:/dev/log debian
root@56ec518d3d4e:/# logger blah blah blah
root@56ec518d3d4e:/# exit
$ sudo tail /var/log/messages | grep logger
Jan 21 08:07:59 halfoat logger: blah blah blah
```

`docker run` — named volumes

Named volumes

- stored inside `/var/lib/docker`
- lifecycle managed with the `docker volume` command
- plugin API to provide shared storage over a cluster/cloud[8]

```
$ docker volume create my-volume
my-volume
$ docker volume ls
DRIVER              VOLUME NAME
local               my-volume
$ docker run --rm -t -i -v my-volume:/vol busybox
/ # echo foo > /vol/bar
/ # ^D
$ docker volume inspect my-volume|grep Mountpoint
        "Mountpoint": "/var/lib/docker/volumes/my-volume/_data",
$ docker run --rm -t -i -v my-volume:/vol busybox cat /vol/bar
foo
$ docker volume rm my-volume
my-volume
```

[8] https://docs.docker.com/engine/tutorials/dockervolumes/

initialisation: bind volumes vs named volumes

- bind volumes are created empty
- named volumes are created with a copy of the image content at the same mount point

```
$ docker run --rm -t alpine ls /etc/apk
arch                keys                protected_paths.d  repositories       world

$ docker run --rm -t -v /tmp/dummy:/etc/apk alpine ls /etc/apk
$ ls /tmp/dummy/
$

$ docker run --rm -t -v dummy:/etc/apk alpine ls /etc/apk
arch                keys                protected_paths.d  repositories       world
$ ls /var/lib/docker/volumes/dummy/_data
arch                keys                protected_paths.d  repositories       world
```

`docker run` — grant access to a device

By default devices are not usable inside the container

```
$ docker run --rm debian  fdisk -l /dev/sda
fdisk: cannot open /dev/sda: No such file or directory

$ docker run --rm debian  sh -c 'mknod /dev/sda b 8 0 && fdisk -l /dev/sda'
fdisk: cannot open /dev/sda: Operation not permitted

$ docker run --rm -v /dev/sda:/dev/sda debian  fdisk -l /dev/sda
fdisk: cannot open /dev/sda: Operation not permitted
```

They can be whitelisted with `--device`

`docker run --device` *`/hostpath`*`[:`*`/containerpath`*` ]` *...*

```
$ docker run --rm --device /dev/sda debian  fdisk -l /dev/sda
Disk /dev/sda: 250.1 GB, 250059350016 bytes
...
```

`docker run` — inter-container links (legacy links[9])

Containers cannot be assigned a static IP address (by design)
→ service discovery is a must

Docker "links" are the most basic way to discover a service

`docker run --link` *ctr*:*alias* ...

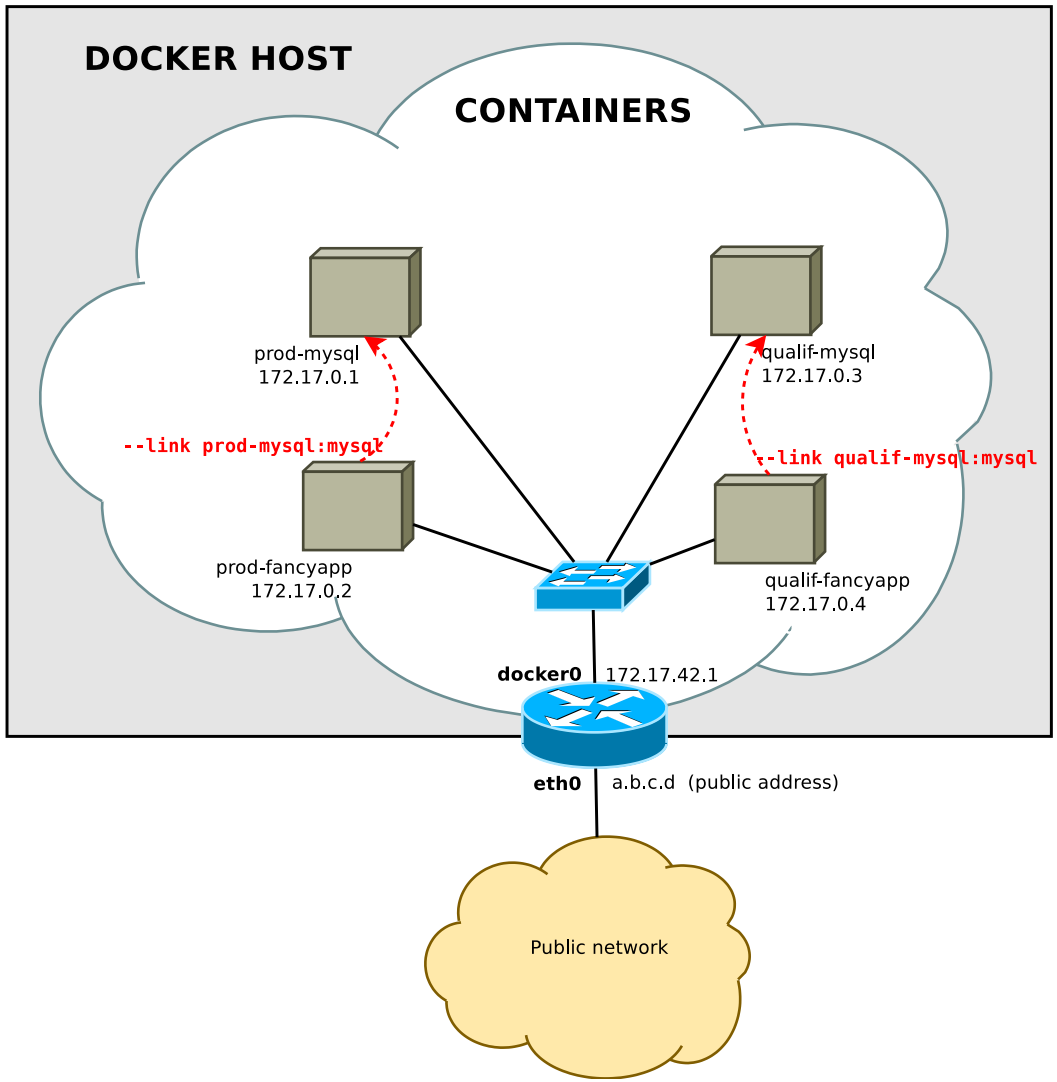
→ container *ctr* will be known as *alias* inside the new container

```
$ docker run --name my-server debian sh -c 'hostname -i && sleep 500' &
172.17.0.4

$ docker run --rm -t -i --link my-server:srv debian
root@d752180421cc:/# ping srv
PING srv (172.17.0.4): 56 data bytes
64 bytes from 172.17.0.4: icmp_seq=0 ttl=64 time=0.195 ms
```

[9] since v1.9.0, links are superseded by user-defined networks

Legacy links

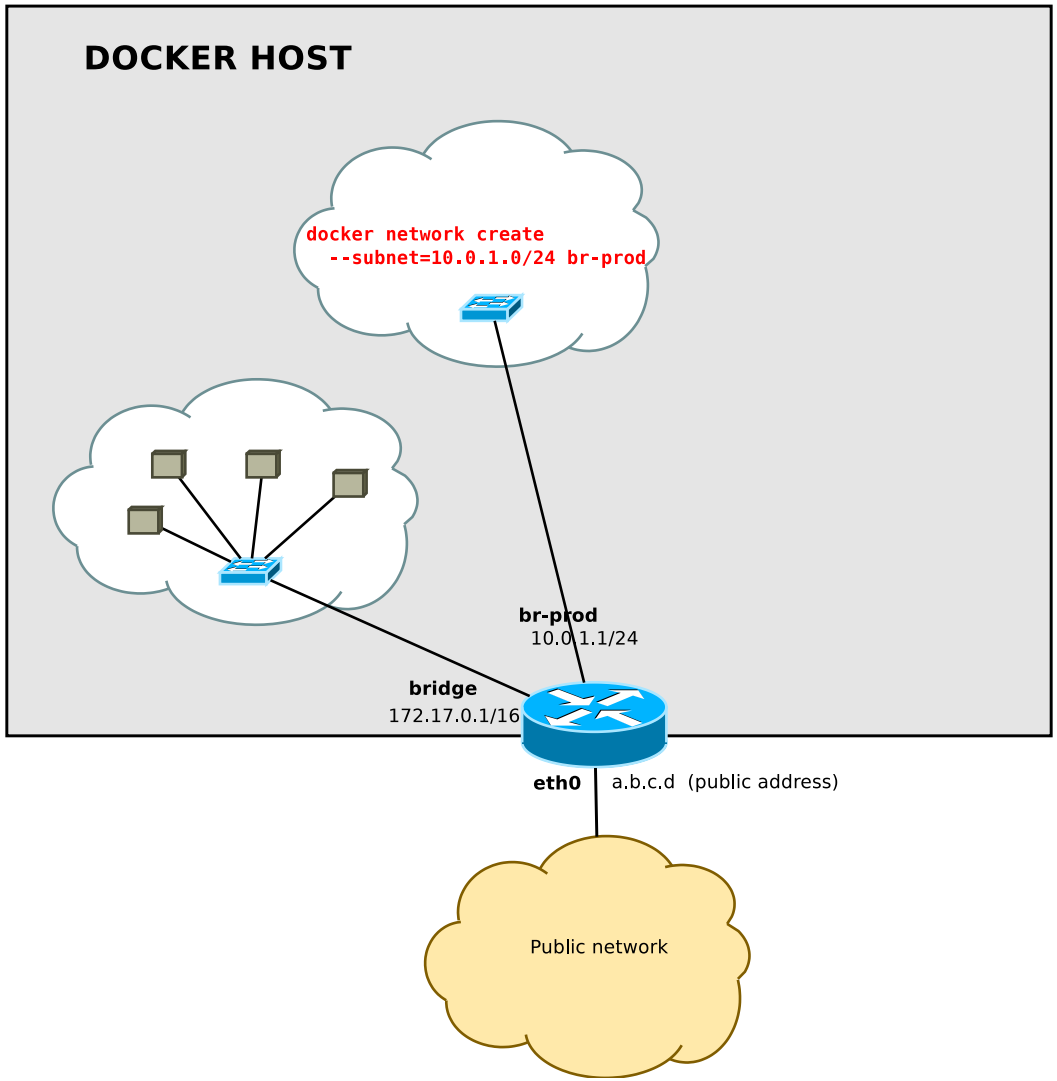
⚠ deprecated feature



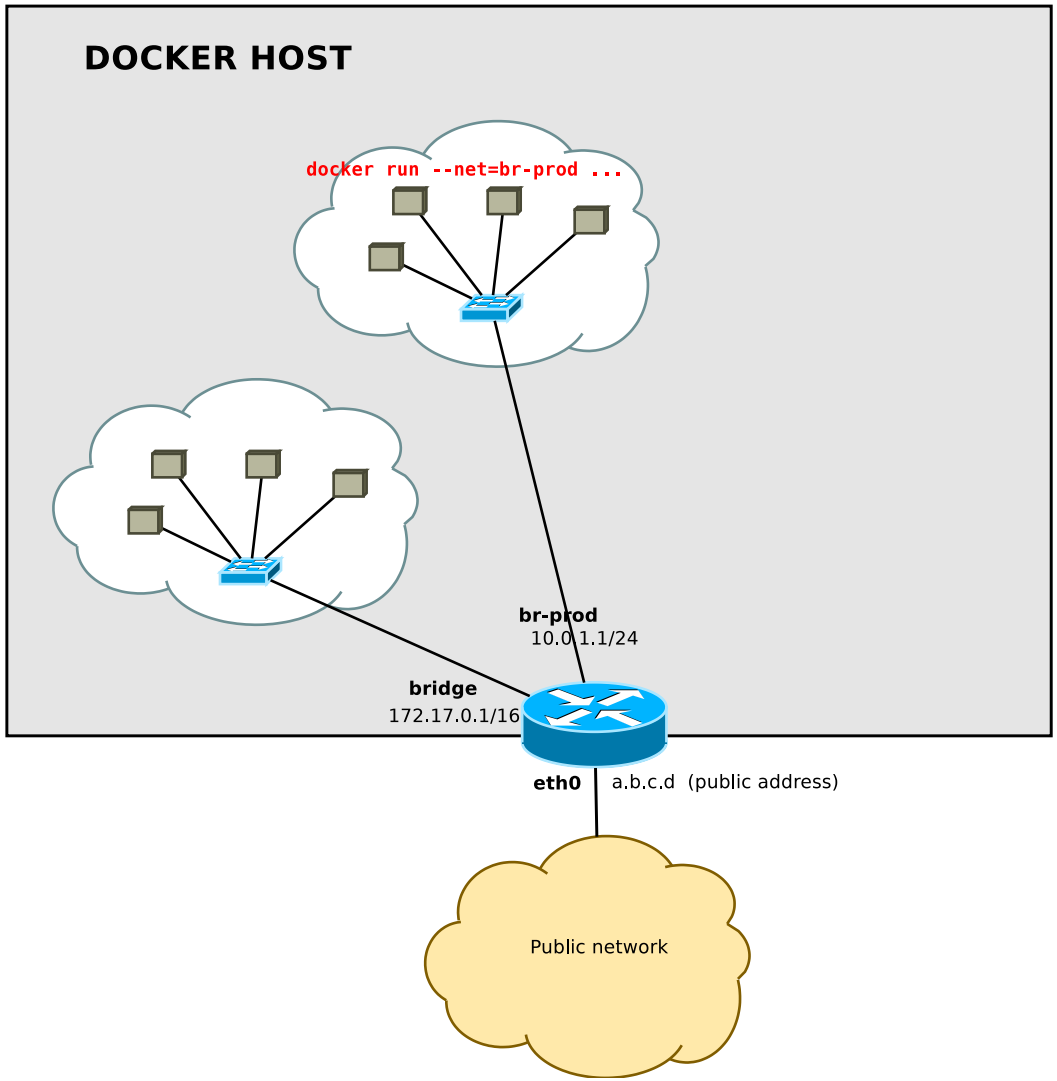
User-defined networks (since v1.9.0)

- by default new containers are connected to the main network (named "bridge", 172.17.0.0/16)
- the user can create additional networks:
  `docker network create` *NETWORK*
- newly created containers are connected to one network:
  `docker run --net=`*NETWORK*
- container may be dynamically attached/detached to any network:
  `docker network connect` *NETWORK CONTAINER*
  `docker network disconnect` *NETWORK CONTAINER*
- networks are isolated from each other, communications is possible by attaching a container to multiple networks

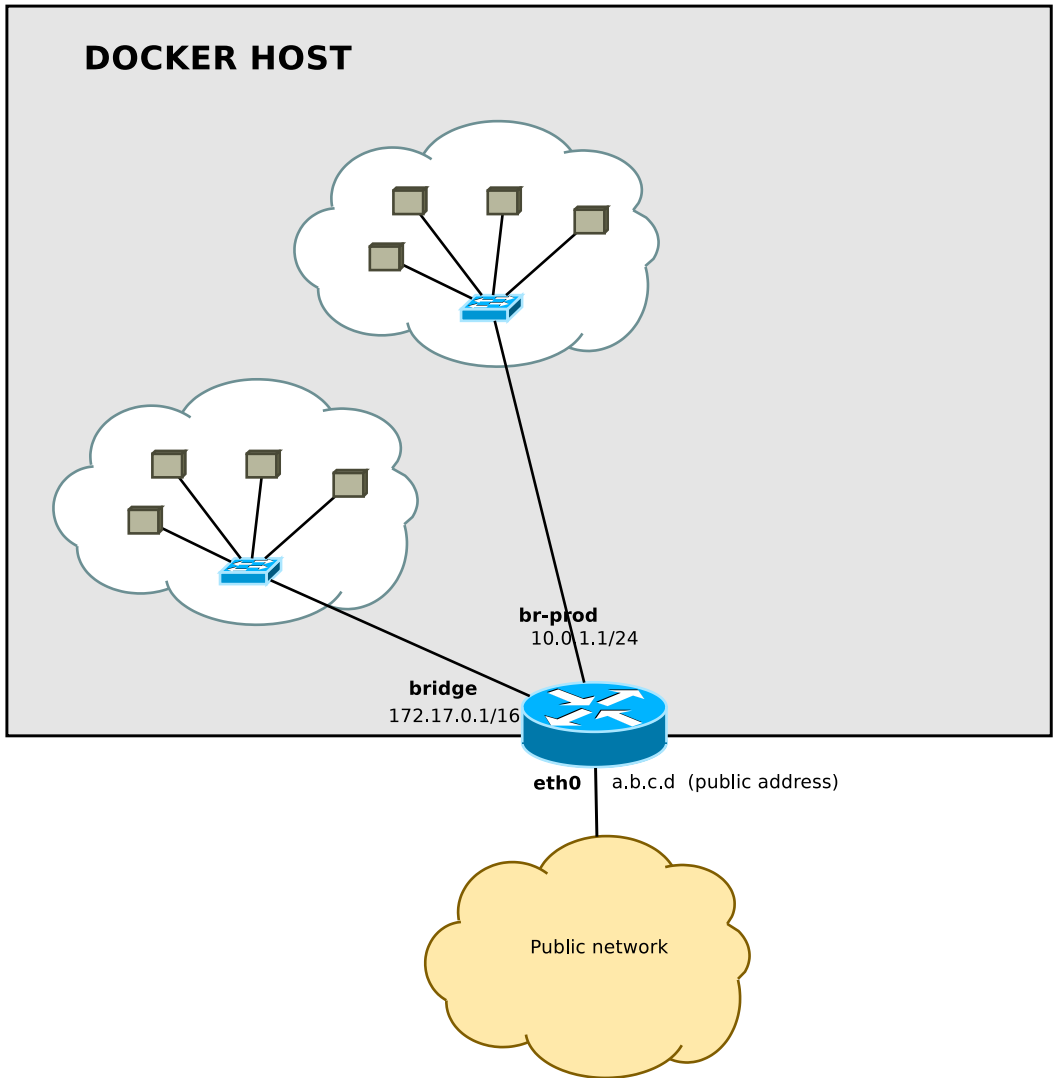
User-defined networks example



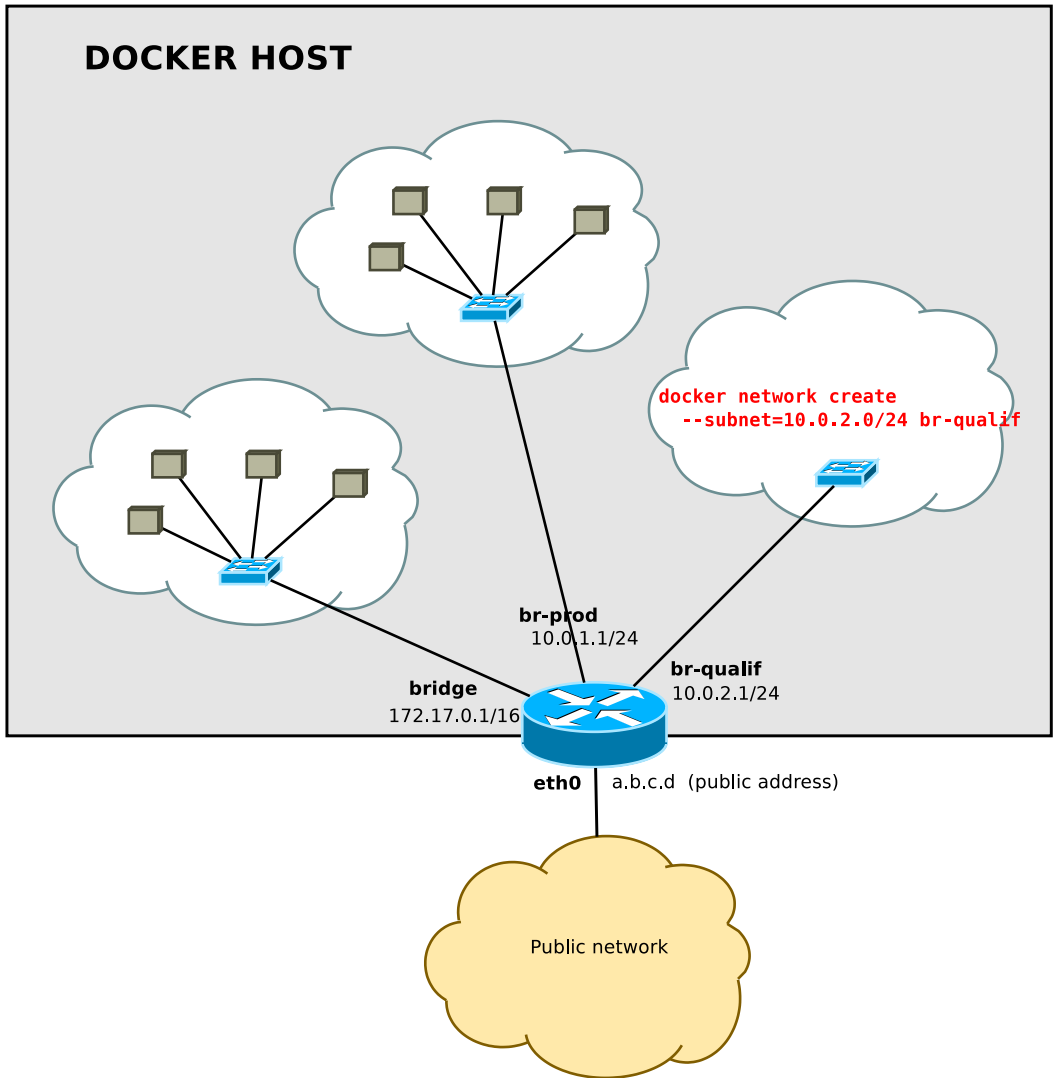
User-defined networks example



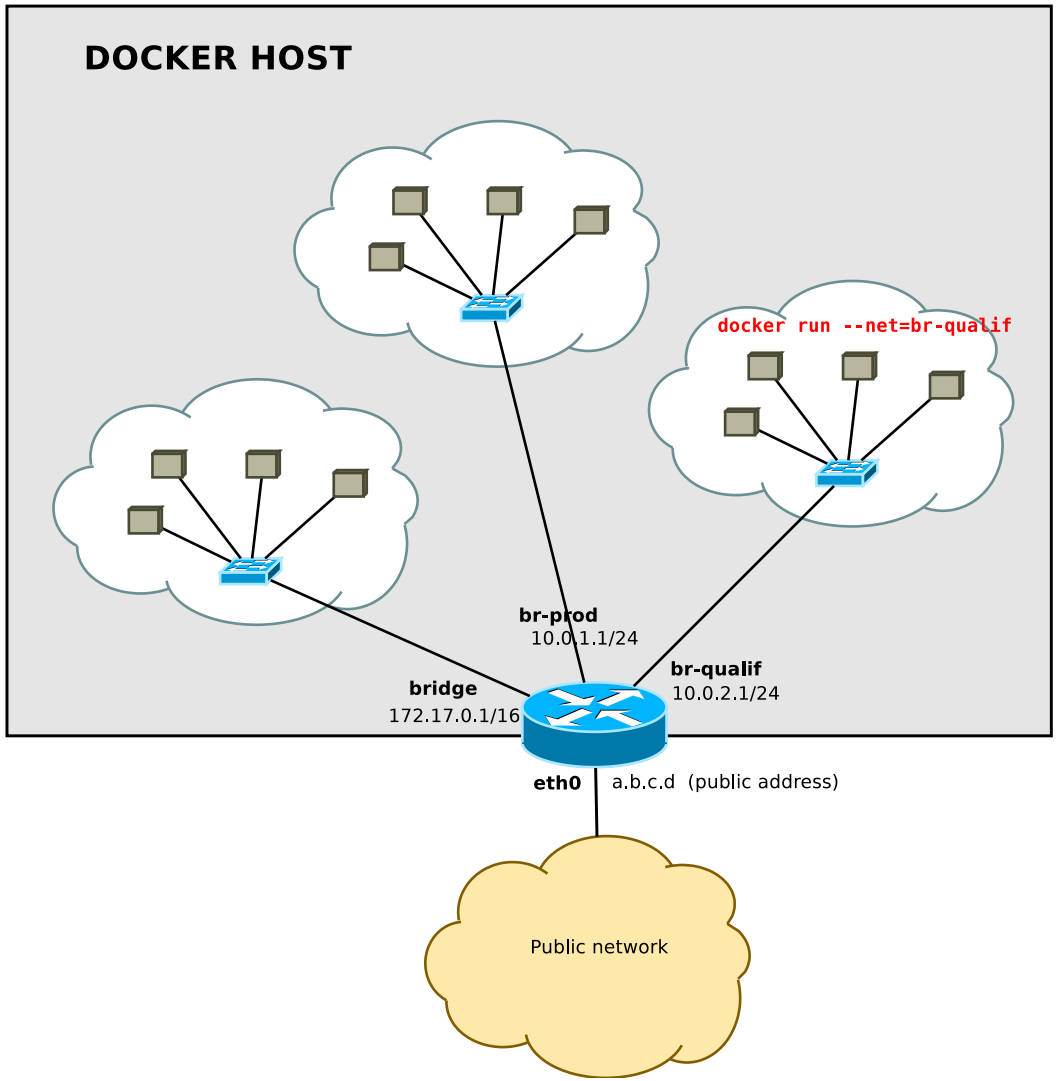
User-defined networks example



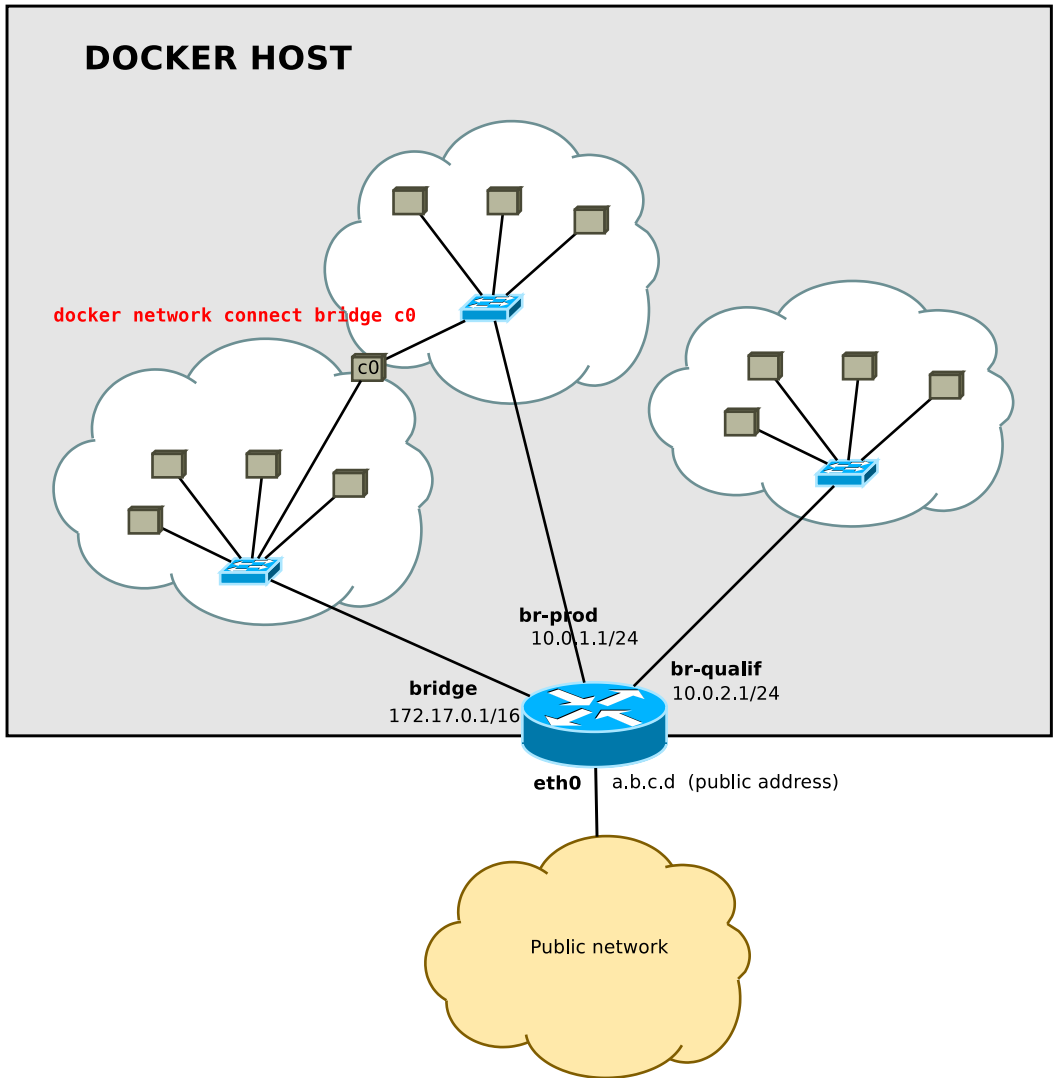
User-defined networks example



User-defined networks example



User-defined networks example



`docker run` — publish a TCP port

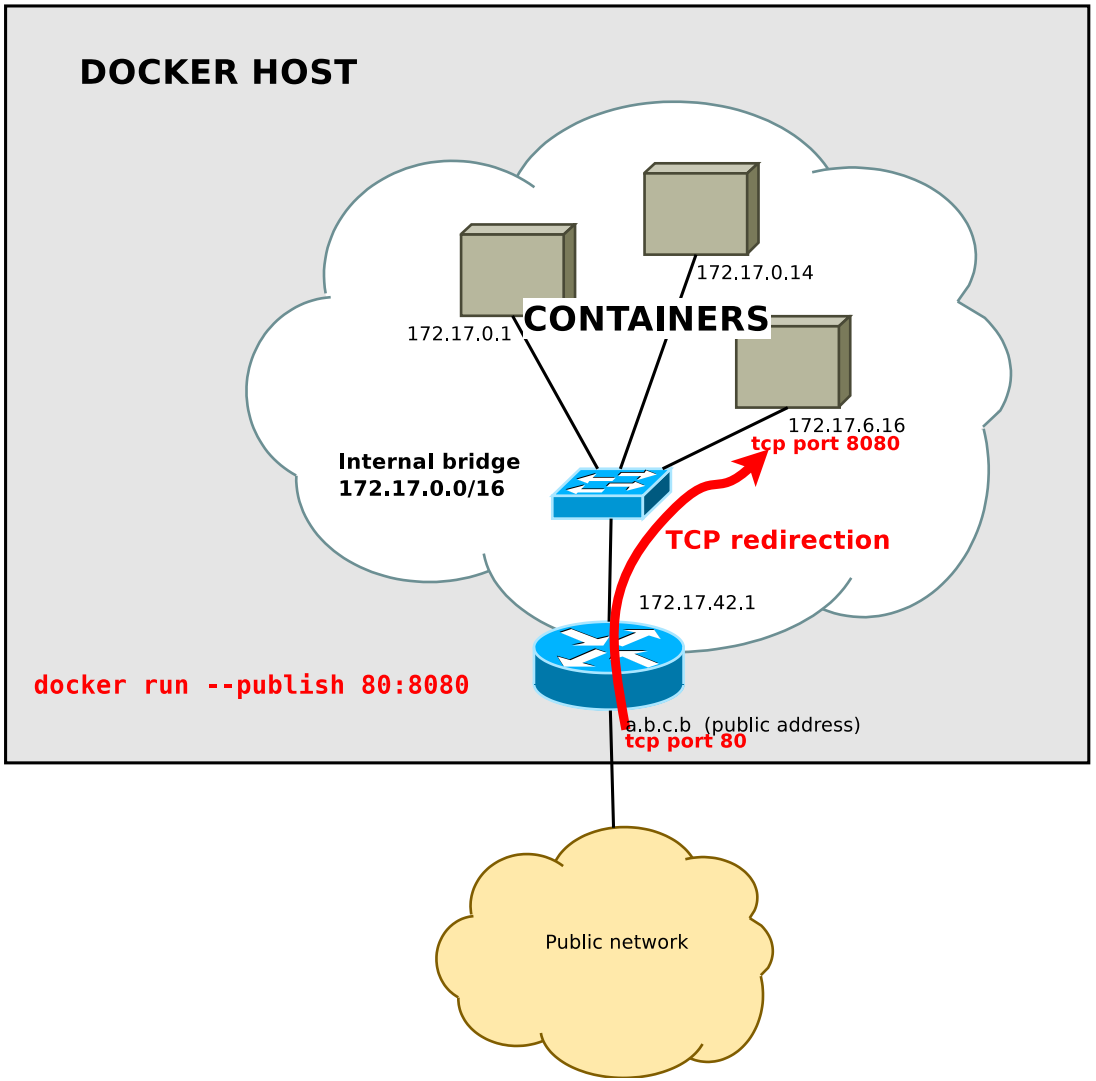
Containers are deployed in a private network, they are not reachable from the outside (unless a redirection is set up)

```
docker run -p [ipaddr:]hostport:containerport
```

→ redirect incoming connections to the TCP port *hostport* of the host to the TCP port *containerport* of the container

The listening socket binds to 0.0.0.0 (all interfaces) by default or to *ipaddr* if given

publish example



publish example

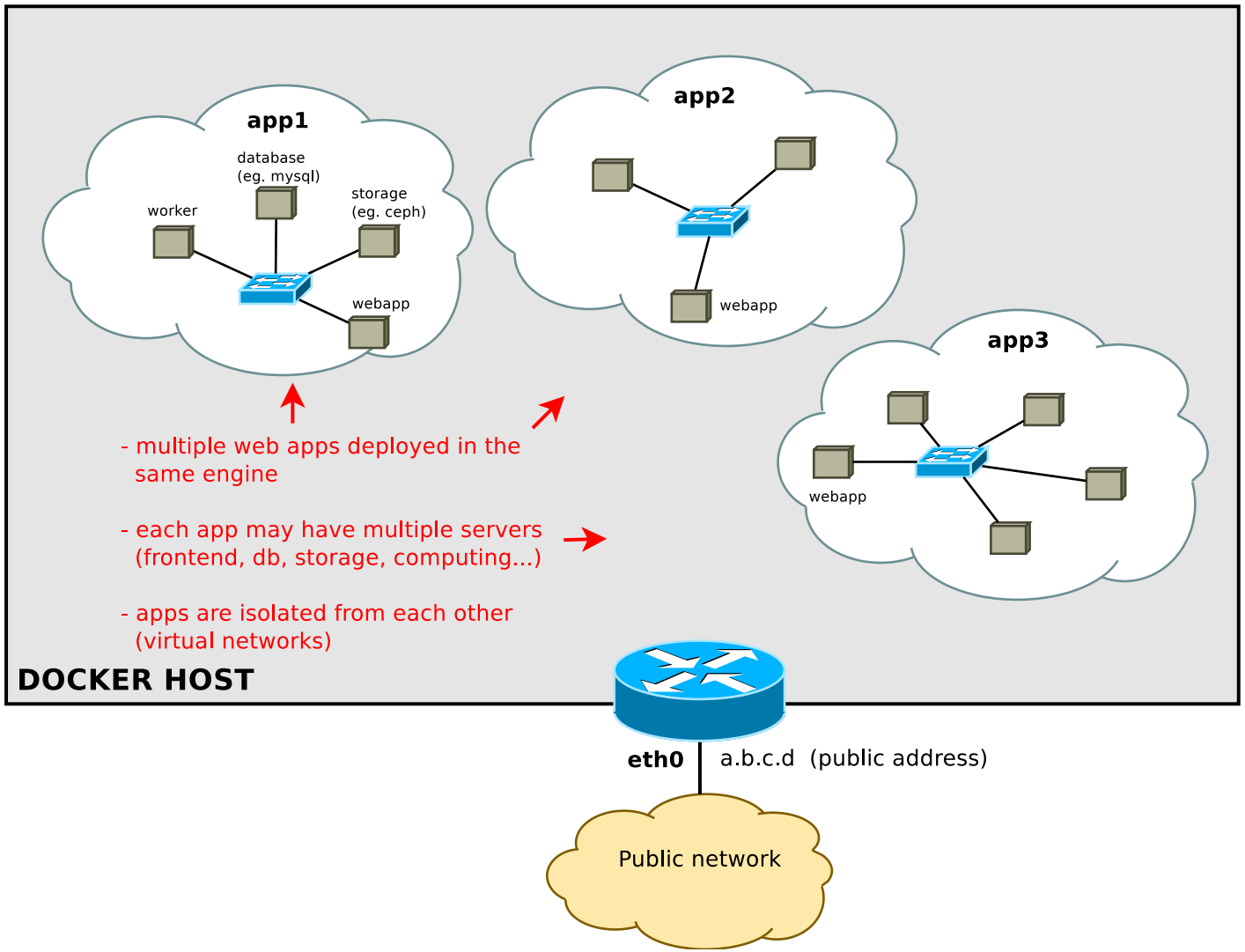
bind to all host addresses

```
$ docker run -d -p 80:80 nginx
52c9105e1520980d49ed00ecf5f0ca694d177d77ac9d003b9c0b840db9a70d62

$ wget -nv http://localhost/
2016-01-12 18:32:52 URL:http://localhost/ [612/612] -> "index.html" [1]
$ wget -nv http://172.17.42.1/
2016-01-12 18:33:14 URL:http://172.17.42.1/ [612/612] -> "index.html" [1]
```

bind to 127.0.0.1

```
$ docker run -d -p 127.0.0.1:80:80 nginx
4541b43313b51d50c4dc2722e741df6364c5ff50ab81b828456ca55c829e732c

$ wget -nv http://localhost/
2016-01-12 18:37:10 URL:http://localhost/ [612/612] -> "index.html.1" [1]
$ wget http://172.17.42.1/
--2016-01-12 18:38:32--  http://172.17.42.1/
Connecting to 172.17.42.1:80... failed: Connection refused.
```

The whole picture



The whole picture