CONTAINER & DOCKER

This slide is part of class material
2110415 Software Defined Systems

By Veera Muangsin

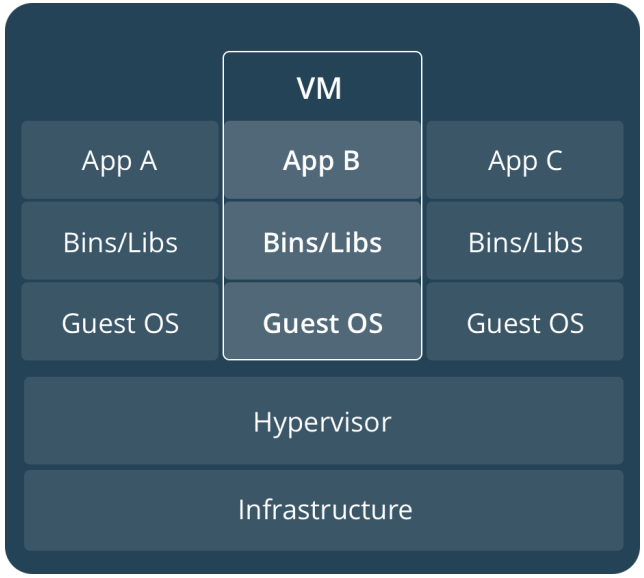
Docker

- Docker is a platform for developers to **develop, deploy, and run** applications with **Linux** containers.

- Note: *Docker for Mac* and *Docker for Windows* run Linux containers in a Linux Virtual Machine.

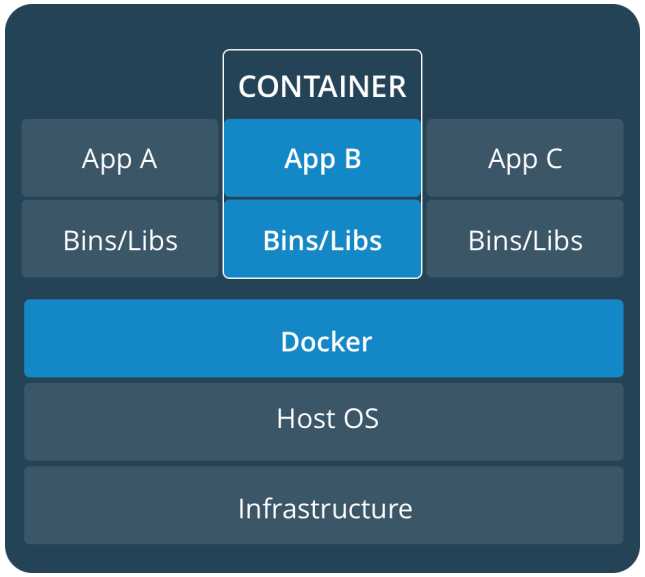
Virtual Machine

- An abstraction of physical hardware.
- Each VM includes a full copy of an operating system, application code and libraries.
- The hypervisor manages multiple VMs.
- Large (GBs). Slow to boot up.



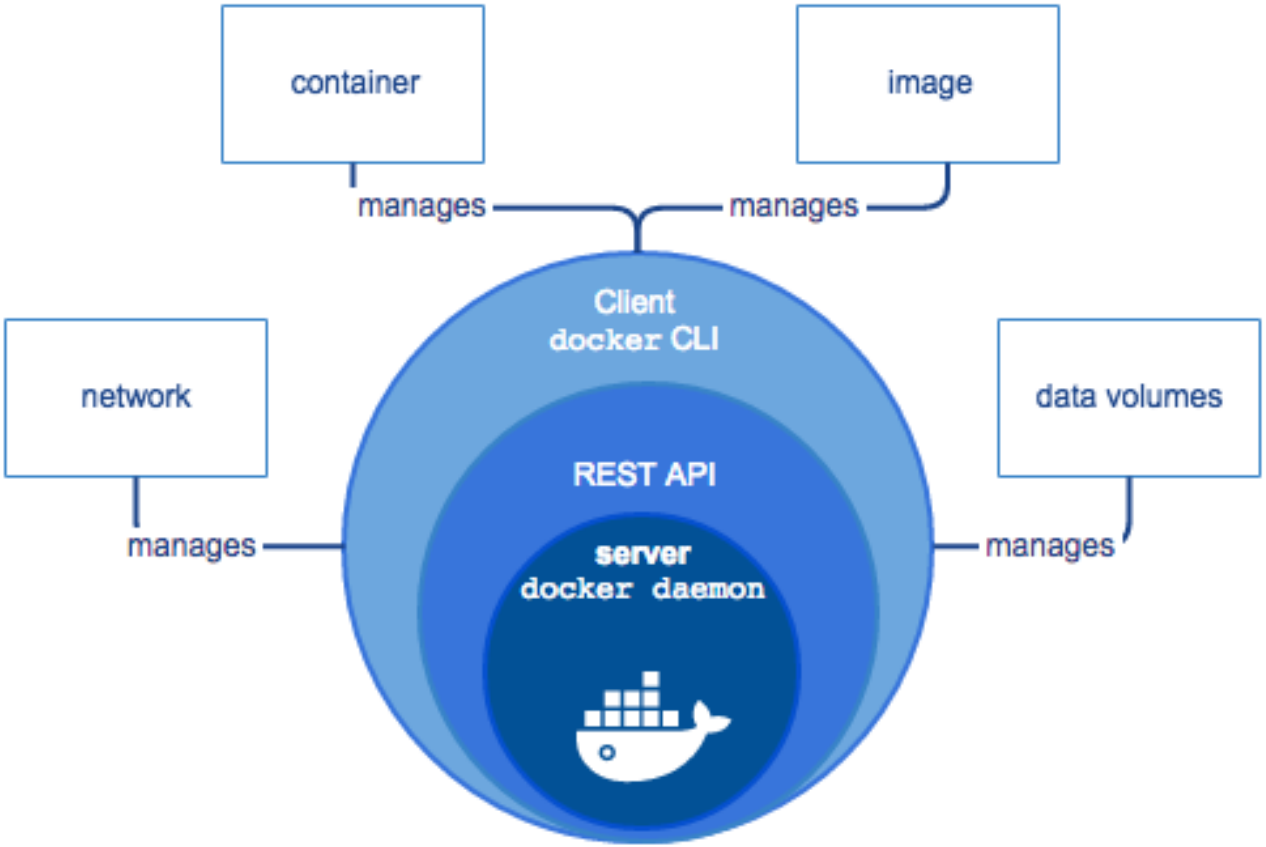
Container

- An abstraction of OS resources.
- Each container includes application code, libraries, settings.
- Containers share OS kernel, each running as isolated processes in user space.
- The container runtime (Docker Engine) manages multiple containers.
- Small (MBs). Fast to start.

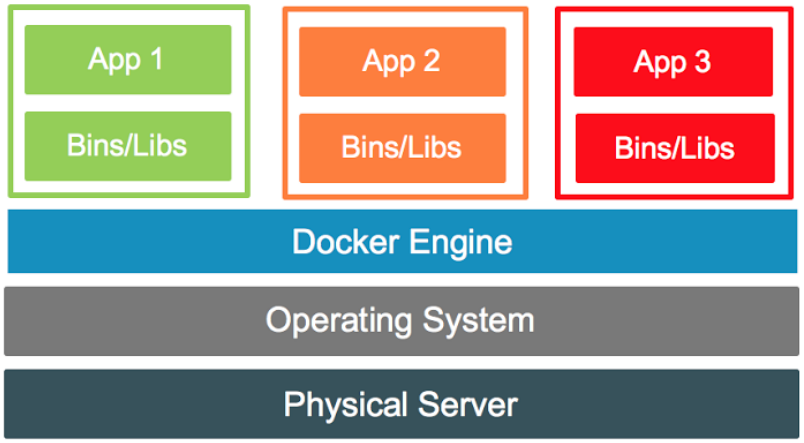


https://www.docker.com/resources/what-container

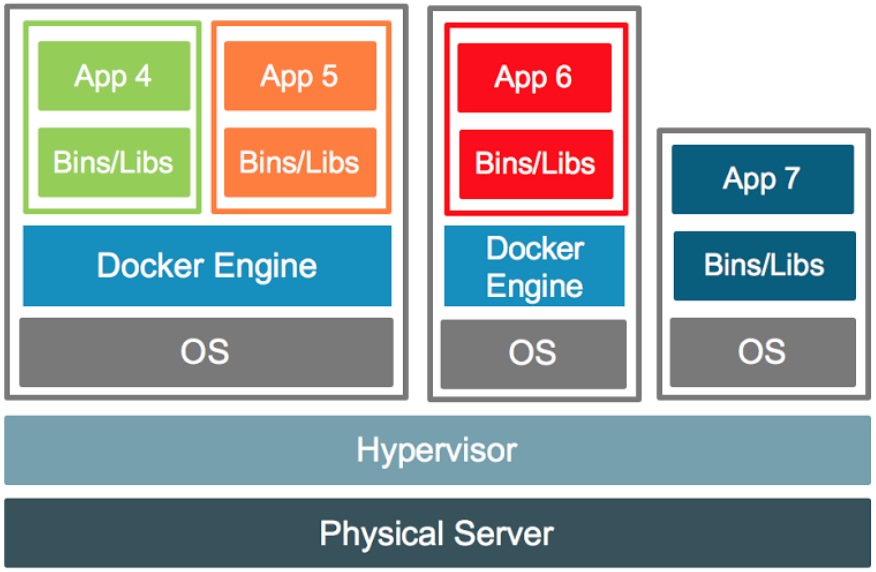
Docker Engine



Containers and Virtual Machines Combinations



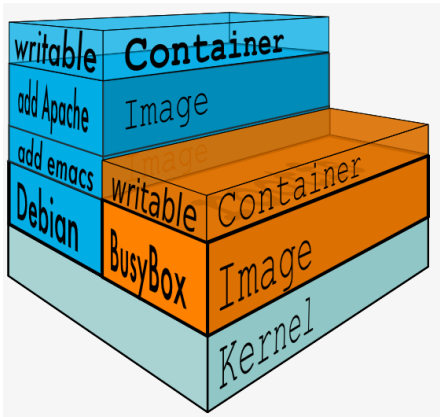
Docker without VM

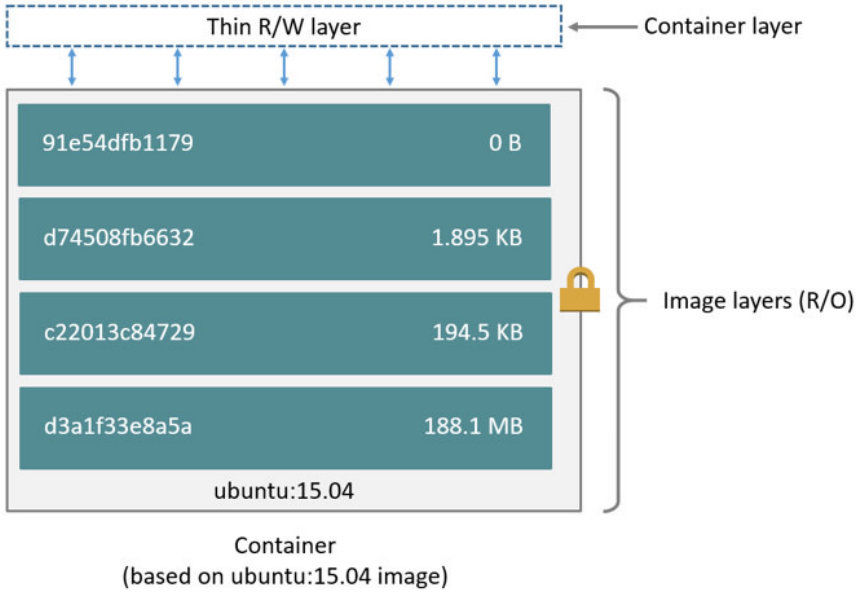


Docker with VMs

Docker Image

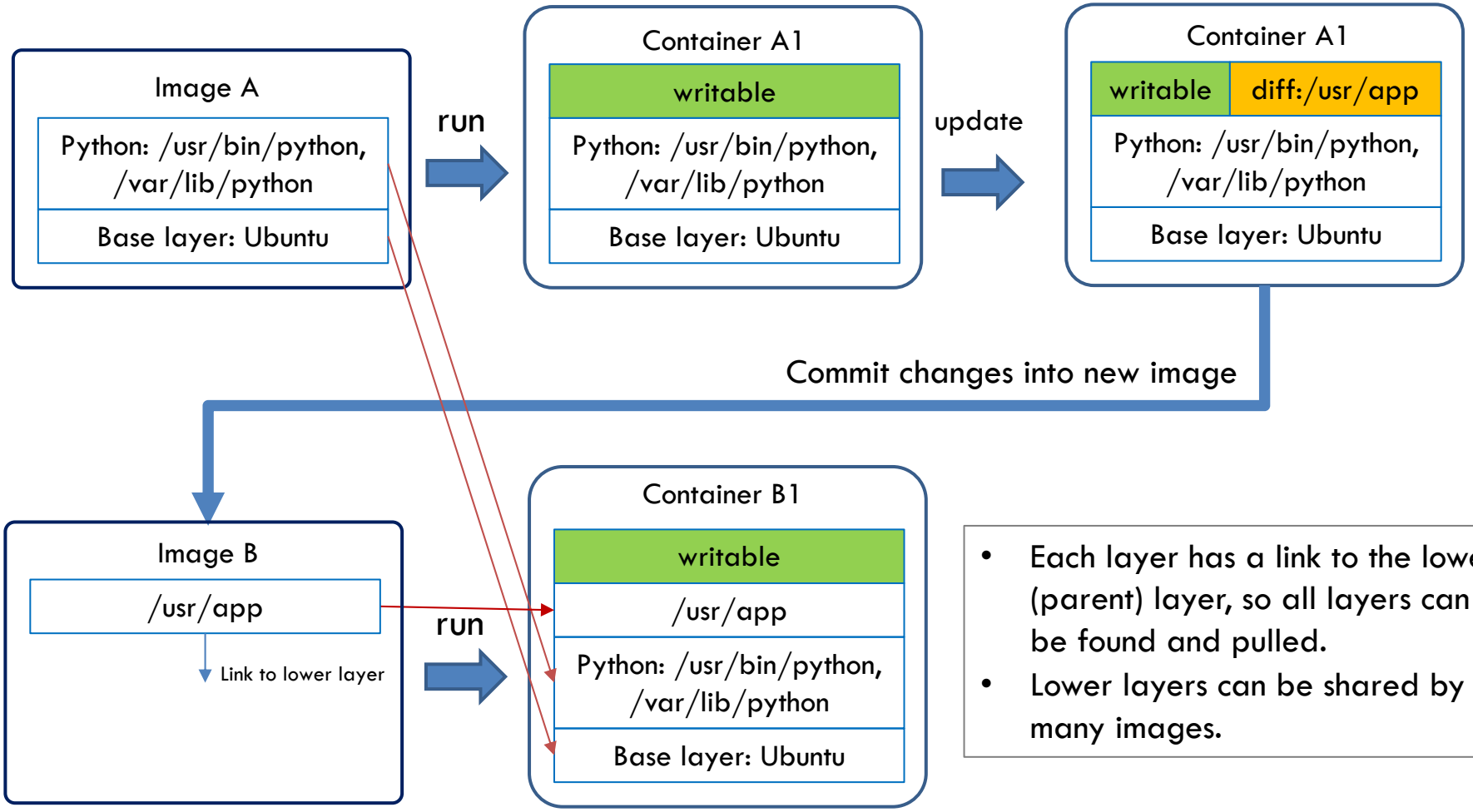
- A Docker image is a read-only template for creating a Docker container.
- Docker uses a Union File System (e.g. AUFS, Overlay2) to combine image layers into a single image.
- The bottom layer is called the base image. It defines the runtime environment.
- When an image is built, other layers are written on top of each other.
- Image layers are immutable (read only) and shared.
- When a container is created from an image, a read-write filesystem is mounted on top.





https://docs.docker.com/storage/storagedriver/

How image evolves

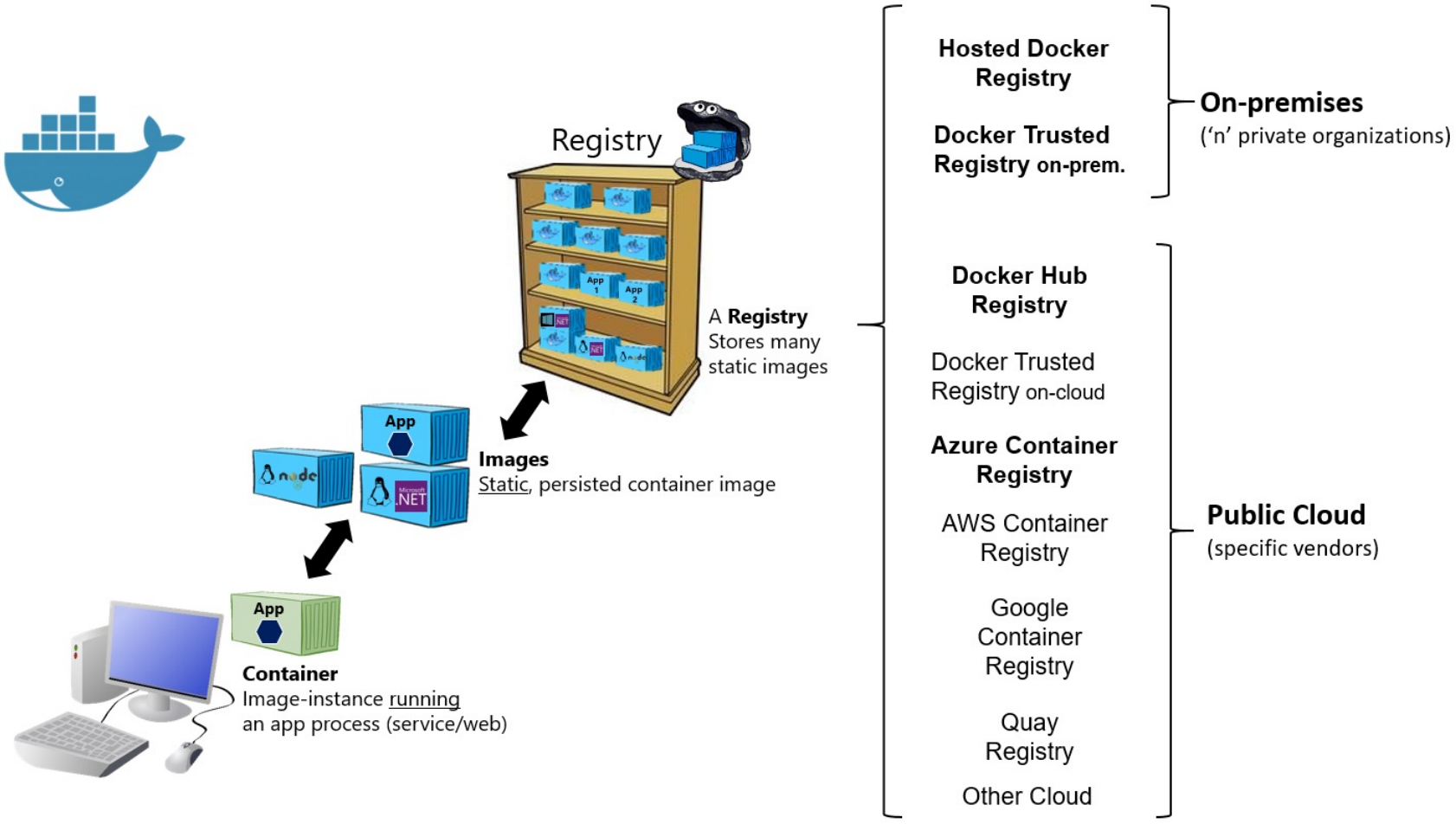


- Each layer has a link to the lower (parent) layer, so all layers can be found and pulled.
- Lower layers can be shared by many images.

Docker Registry

- To facilitate image sharing, Docker images are stored and distributed via "*registries*".
- A registry is a storage and content delivery system, holding named Docker images.
- Docker maintains a public registry Docker Hub
  - https://hub.docker.com/
  - Other vendors provide registries for different collections of images.

Registries



- Docker maintains a public registry via Docker Hub; other vendors provide registries for different collections of images.

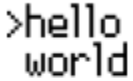
https://docs.microsoft.com/th-th/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries

Docker Images on Docker Hub

ubuntu DOCKER OFFICIAL IMAGE

Updated 2 days ago

Ubuntu is a Debian-based Linux operating system based on free software.



hello-world DOCKER OFFICIAL IMAGE

Updated 25 days ago

Hello World! (an example of minimal Dockerization)

ubuntu/nginx VERIFIED PUBLISHER

By Canonical • Updated 12 days ago

Nginx, a high-performance reverse proxy & web server. Long-term tracks maintained by Canonical.

grafana/grafana VERIFIED PUBLISHER

By Grafana Labs • Updated 2 days ago

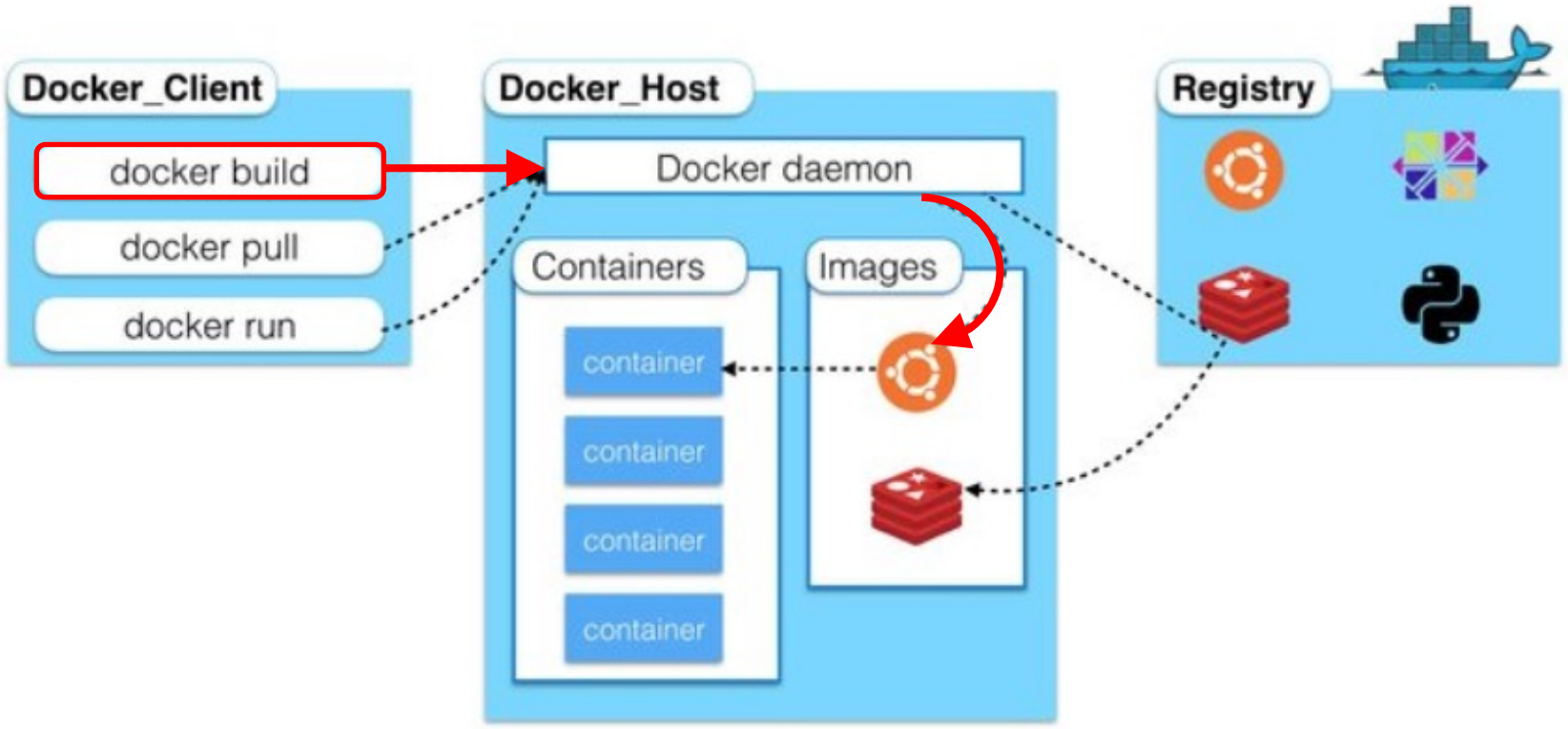
The official Grafana docker container

Naming convention:

For official images: only “name”

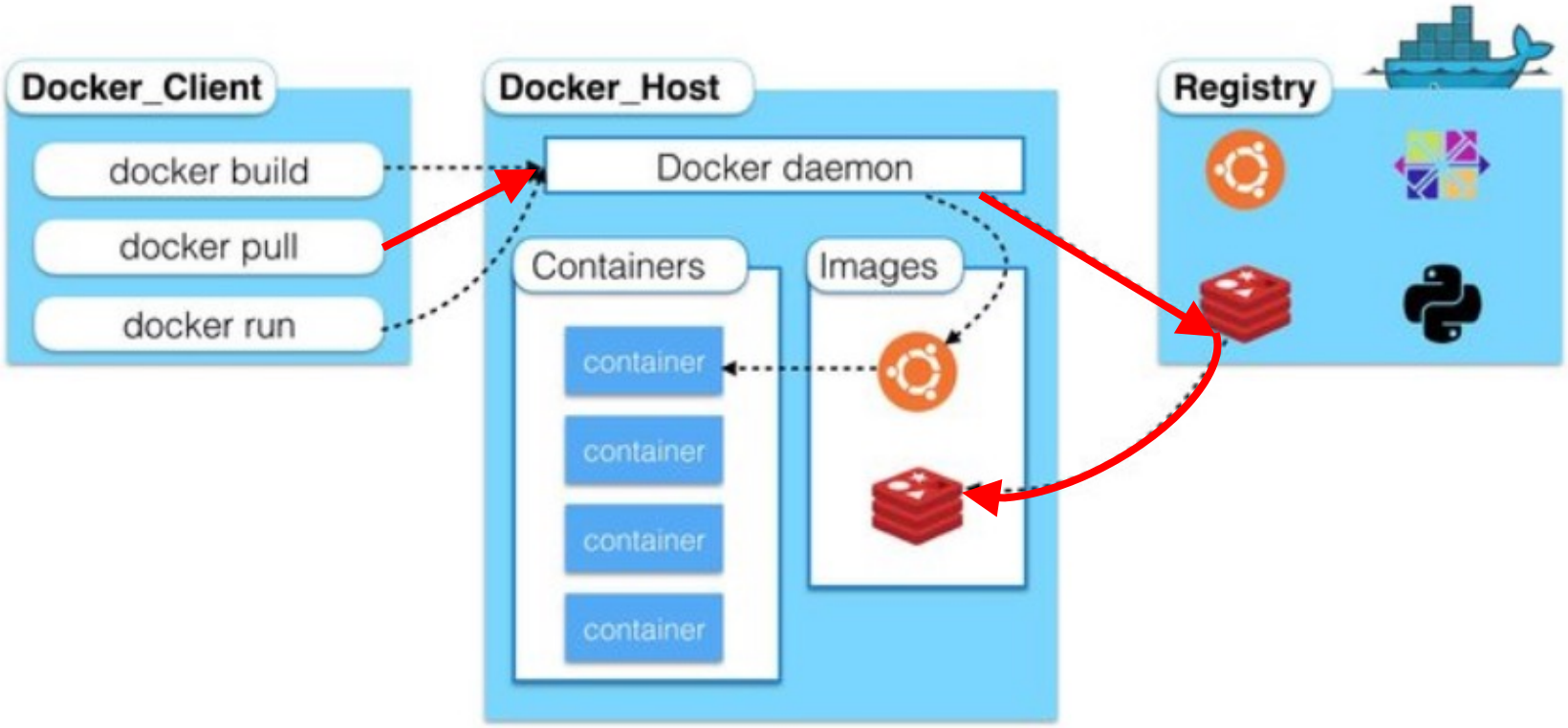
For others: “user/name”

Basic operation: building an image

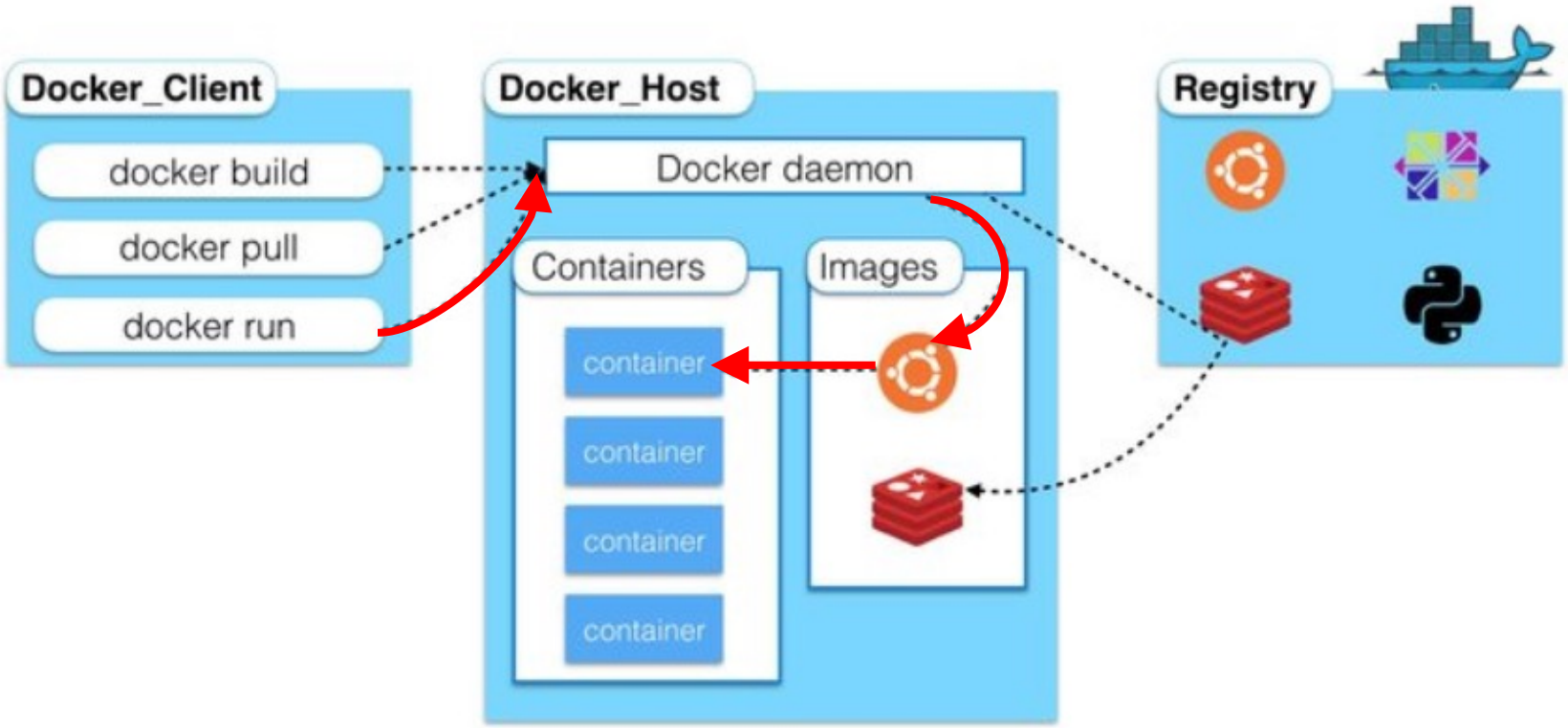


Basic operation: pulling an image from registry

- If an image is not found locally, it can be pulled from a registry.



Basic operation: running an image



Docker Commands

Basic Docker Commands

```
## List Docker CLI commands
docker
docker COMMAND --help

## Pull Docker image from a registry
docker pull IMAGE

## Execute Docker image (create a container)
## Its default command can be override by COMMAND option
docker run IMAGE [COMMAND]
docker run --rm IMAGE [COMMAND]
docker run --name NAME IMAGE [COMMAND]

## List Docker images
docker image ls             same as  docker images
docker image ls --all       same as  docker images -a

## List Docker containers
docker container ls          same as   docker ps
docker container ls --all    same as   docker ps -a
```

Docker command: run

```
$ docker run [OPTIONS] IMAGE [COMMAND] [ARGS]
```

- Most common options

`--name` assign a name to the container

`-d` detached mode (in background)

`-it` foreground with attached pseudo-tty and STDIN (interactive)

`-expose=[]` expose container's ports to other containers on the same network

`-p=[]` publish container's ports to host machine and external clients

`-v` use a persistence storage

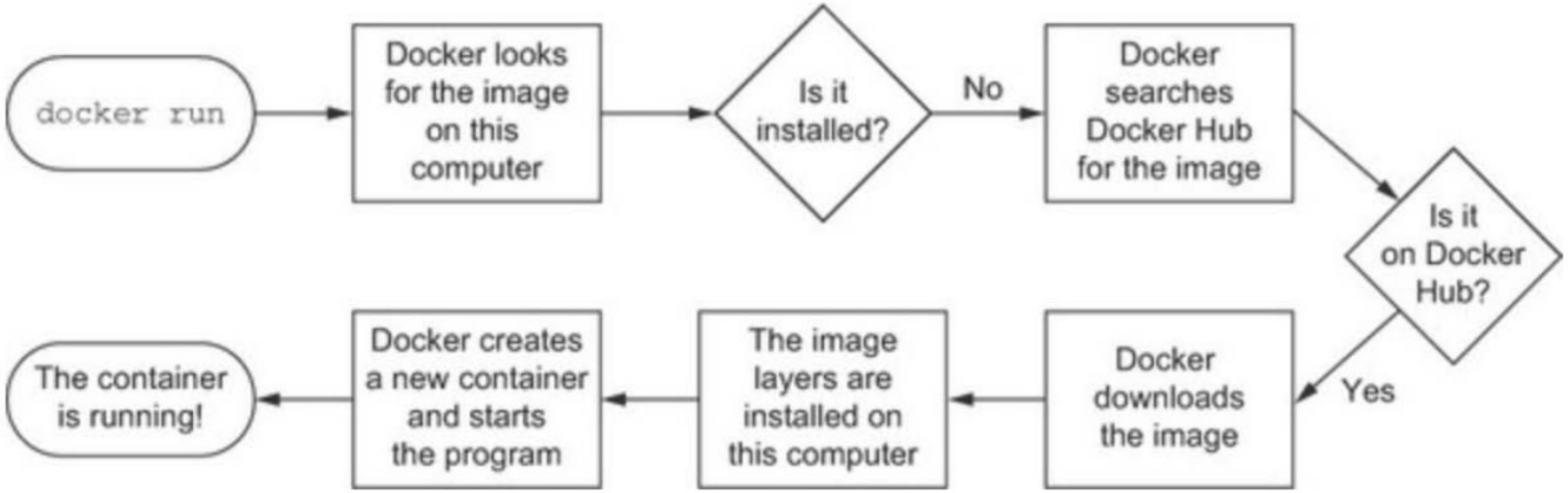
`-e` set environment variables

`-link=[]` link to other containers

- The "Hello World" container

```
$ docker run ubuntu /bin/echo 'Hello world'
```

docker run



From “Docker in Action”, J. Nickoloff

Pull and run an image

```
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
veera@ubuntu-vm:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f2a5301c00
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

registry/username/image_name:tag

`docker pull hello-world` is a shortcut for

`docker pull docker.io/library/hello-world:latest`

```
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        8 months ago        1.84kB
veera@ubuntu-vm:~$ docker run hello-world

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.
```

Run an image for the first time

```
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
veera@ubuntu-vm:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f2a5301c00
Status: Downloaded newer image for hello-world:latest

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/
```

Run an image interactively

```
docker run --interactive --tty IMAGE [COMMAND]
docker run -it IMAGE [COMMAND]
```

```
veera@ubuntu-vm:~$ docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
35c102085707: Pull complete
251f5509d51d: Pull complete
8e829fe70a46: Pull complete
6001e1789921: Pull complete
Digest: sha256:d1d454df0f579c6be4d8161d227462d69e163a8ff9d20a847533989cf0c94d90
Status: Downloaded newer image for ubuntu:latest
root@f7bd2f332fa7:/#
root@f7bd2f332fa7:/# hostname
f7bd2f332fa7
root@f7bd2f332fa7:/# exit
exit
veera@ubuntu-vm:~$
veera@ubuntu-vm:~$ docker image ls --all
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              a2a15febcdf3        3 weeks ago         64.2MB
hello-world         latest              fce289e99eb9        8 months ago        1.84kB
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
f7bd2f332fa7        ubuntu              "/bin/bash"         4 minutes ago       Exited (0) 43 seconds ago                       trusting_aryabhata
d2aeb346082e        hello-world         "/hello"            27 minutes ago      Exited (0) 27 minutes ago                       modest_lumiere
9701313ec8ac        hello-world         "/hello"            37 minutes ago      Exited (0) 37 minutes ago                       happy_easley
```

If not specified, containers are given random names.

Remove a container

```
docker container rm CONTAINER [CONTAINER…]
```

```
veera@ubuntu-vm:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
veera@ubuntu-vm:~$
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                          PORTS               NAMES
2a63848173ee        ubuntu              "/bin/bash"         14 hours ago        Exited (255) 17 minutes ago                         zealous_poitras
f7bd2f332fa7        ubuntu              "/bin/bash"         15 hours ago        Exited (127) 14 hours ago                           trusting_aryabhata
d2aeb346082e        hello-world         "/hello"            15 hours ago        Exited (0) 15 hours ago                             modest_lumiere
9701313ec8ac        hello-world         "/hello"            15 hours ago        Exited (0) 15 hours ago                             happy_easley
veera@ubuntu-vm:~$ docker container rm zealous_poitras trusting_aryabhata
zealous_poitras
trusting_aryabhata
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
d2aeb346082e        hello-world         "/hello"            15 hours ago        Exited (0) 15 hours ago                         modest_lumiere
9701313ec8ac        hello-world         "/hello"            15 hours ago        Exited (0) 15 hours ago                         happy_easley
```

Remove all container

```
veera@ubuntu-vm:~$ docker container rm $(docker container ls --all --quiet)
d2aeb346082e
9701313ec8ac
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

Same as

```
docker rm $(docker ps -a -q)
```

Run image in background

```
docker run --detach [--tty] --name NAME IMAGE
docker exec CONTAINER COMMAND
```

```
veera@ubuntu-vm:~$ docker run --detach --tty --name ubuntu ubuntu
b91555fadae3c5f1cd65b4625528e900c8677babd59c5019677ea2216181ed7
veera@ubuntu-vm:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b91555fadae3        ubuntu              "/bin/bash"         5 seconds ago       Up 4 seconds                            ubuntu
veera@ubuntu-vm:~$ docker exec ubuntu hostname
b91555fadae3
```

```
veera@ubuntu-vm:~$ docker exec -it ubuntu bash
root@b91555fadae3:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@b91555fadae3:/#
root@b91555fadae3:/# exit
exit
veera@ubuntu-vm:~$
```

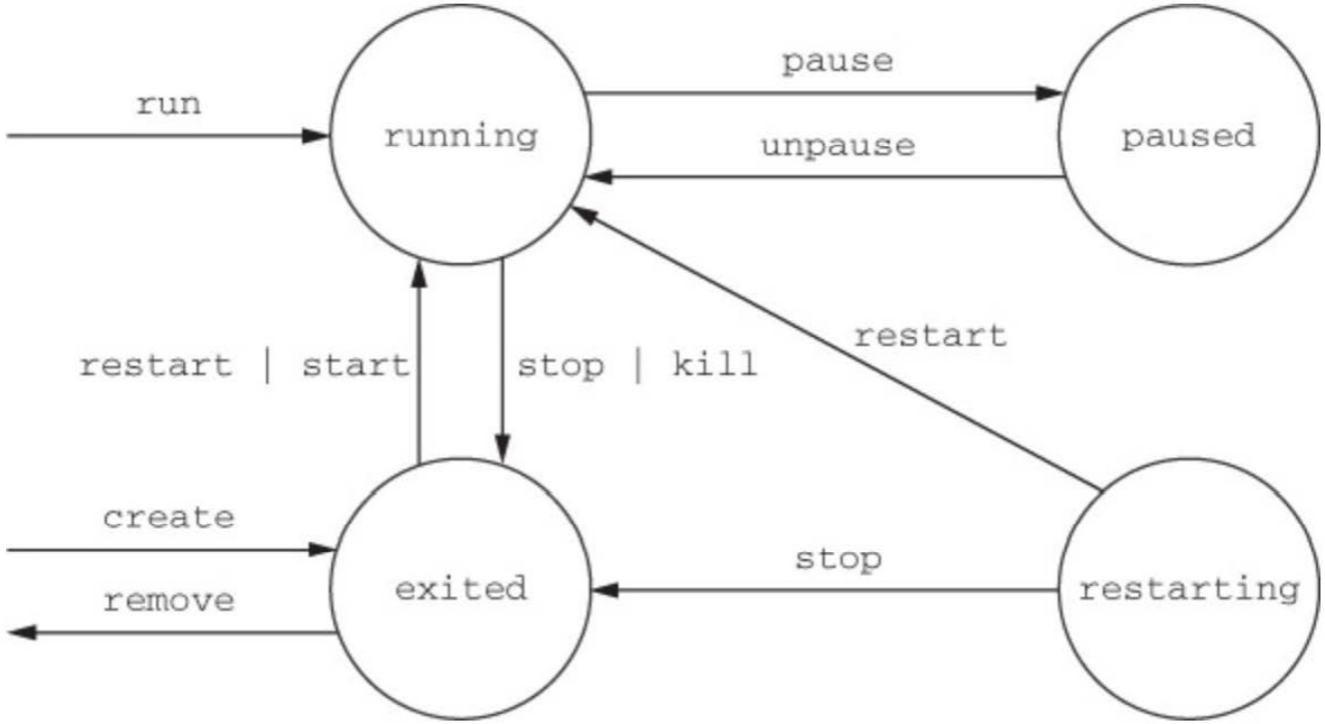
Stop and start a container

```
docker stop CONTAINER
docker start CONTAINER
```

```
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b91555fadae3        ubuntu              "/bin/bash"         7 hours ago         Up 7 hours                              ubuntu
veera@ubuntu-vm:~$ docker stop ubuntu
ubuntu
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
b91555fadae3        ubuntu              "/bin/bash"         7 hours ago         Exited (0) 4 seconds ago                        ubuntu
veera@ubuntu-vm:~$ docker start ubuntu
ubuntu
veera@ubuntu-vm:~$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b91555fadae3        ubuntu              "/bin/bash"         7 hours ago         Up 2 seconds                            ubuntu
```

- *docker stop* kills processes in the container but leave code & data in the container.
- *docker start* on a stopped container will restart the container with the code & data that remain in the container.
- *docker start* will run the same command that was run when the container was created
  - The CMD command in dockerfile, or in *docker run image* COMMAND
  - Run command *docker inspect CONTAINER* → see "Config" section

State transitions of Docker containers



From “Docker in Action”, J. Nickoloff

Commit changes to an image

```
docker commit CONTAINER IMAGE_NAME
```

```
veera@ubuntu-vm:~$ docker run -it ubuntu
root@f33b176d0054:/# ifconfig
bash: ifconfig: command not found
root@f33b176d0054:/# apt update
root@f33b176d0054:/# apt install net-tools
root@f33b176d0054:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
```

While the container is running:

```
veera@ubuntu-vm:~$ docker ps
CONTAINER ID   IMAGE                                   COMMAND                  CREATED              STATUS
f33b176d0054   ubuntu                                  "/bin/bash"              2 minutes ago        Up 2 minutes
veera@ubuntu-vm:~$ docker diff f33b176d0054
C /sbin
A /sbin/mii-tool
A /sbin/slattach
A /sbin/ifconfig
veera@ubuntu-vm:~$ docker commit f33b176d0054 ubuntu-nettools
sha256:2eca02e842367861f911c06c5f9a09be398b5554b3c037b12d5074439d4b18dc
veera@ubuntu-vm:~$ docker images
REPOSITORY                              TAG       IMAGE ID       CREATED          SIZE
ubuntu-nettools                         latest    2eca02e84236   8 seconds ago    103MB
veera@ubuntu-vm:~$ docker run -it ubuntu-nettools
root@2fdc959bfad3:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
```

Remove an image

```
docker image rm IMAGE
```

```
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              a2a15febcdf3        4 weeks ago         64.2MB
veera@ubuntu-vm:~$ docker image rm ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:d1d454df0f579c6be4d8161d227462d69e163a8ff9d20a847533989cf0c94d90
Deleted: sha256:a2a15febcdf362f6115e801d37b5e60d6faaeedcb9896155e5fe9d754025be12
Deleted: sha256:fdc47e80ad3dbe1767a5c2141442c0c7aa93a14a817357a0d4353cd8ae48ee58
Deleted: sha256:2b4ed599a73ad82b15d4e3488d95af4f387037b90401d63ad6cb433374b3e3d3
Deleted: sha256:8f06e3a624319de717230f0d766dd8922d048441adb9e0387860a8047d000409
Deleted: sha256:6cebf3abed5fac58d2e792ce8461454e92c245d5312c42118f02e231a73b317f
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

Before removing an image, all containers that run from that image must be removed first.

Remove all images

```
docker image rm $(docker image ls -q)
```

Dangling Images

A dangling image is an unused image with no name and tag.
An image become dangling when it is overwritten with a new image of the same name and tag.

```
pom@X280:~$ docker image ls
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
docker101tutorial           latest              ca94c52211b0        5 hours ago         27.3MB
mveera/docker101tutorial    latest              ca94c52211b0        5 hours ago         27.3MB
<none>                      <none>              d85c91c4b7f9        5 hours ago         85.6MB
<none>                      <none>              9e9299b94217        5 hours ago         224MB
<none>                      <none>              26a89c3197d3        5 hours ago         72MB
python                      alpine              0f03316d4a27        4 days ago          42.7MB
ubuntu                      latest              4e2eef94cd6b        3 weeks ago         73.9MB
nginx                       alpine              6f715d38cfe0        4 weeks ago         22.1MB
node                        12-alpine           18f4bc975732        6 weeks ago         89.3MB
docker/getting-started      latest              1f32459ef038        2 months ago        26.8MB
hello-world                 latest              bf756fb1ae65        8 months ago        13.3kB
pom@X280:~$ docker image ls -f "dangling=true"
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              d85c91c4b7f9        5 hours ago         85.6MB
<none>              <none>              9e9299b94217        5 hours ago         224MB
<none>              <none>              26a89c3197d3        5 hours ago         72MB
```

Remove all *dangling* (<none>:<none>) images

```
docker image rm $(docker image ls -f "dangling=true" -q)
```

Pruning

```
## Remove stopped containers
docker container prune


## Remove unused images
docker image prune
```

Push an image to a local registry

Run a local registry

```
veera@ubuntu-vm:~$ docker run -d -p 5000:5000 --name registry registry
```

Create an image

```
veera@ubuntu-vm:~$ cd docker/hello_python/
veera@ubuntu-vm:~/docker/hello_python$ docker build -t hello .
veera@ubuntu-vm:~/docker/hello_python$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               latest              192ad07dd1d2        4 seconds ago       918MB
```

Create a tag

```
veera@ubuntu-vm:~/docker/hello_python$ docker image tag hello localhost:5000/my-image
veera@ubuntu-vm:~/docker/hello_python$ docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
hello                     latest              192ad07dd1d2        44 seconds ago      918MB
localhost:5000/my-image   latest              192ad07dd1d2        44 seconds ago      918MB
```

Push the image to the registry

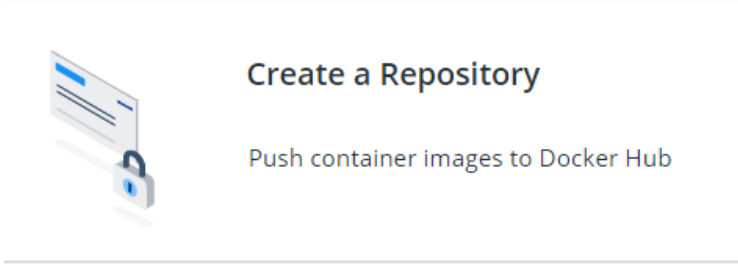
```
veera@ubuntu-vm:~/docker/hello_python$ docker push localhost:5000/my-image
Using default tag: latest
The push refers to repository [localhost:5000/my-image]
dd6f71fba59f: Pushed
7dd4eea0238a: Pushed
63a0a46eada9: Pushed
fbf93044a68a: Pushed
51ec4ea28b75: Pushed
2db44bce66cd: Pushed
latest: digest: sha256:1c9a5ef5dbfdda9018f435e0027a21f142fc6fb5cc735094247690c64c2e5e10 size: 1577
```

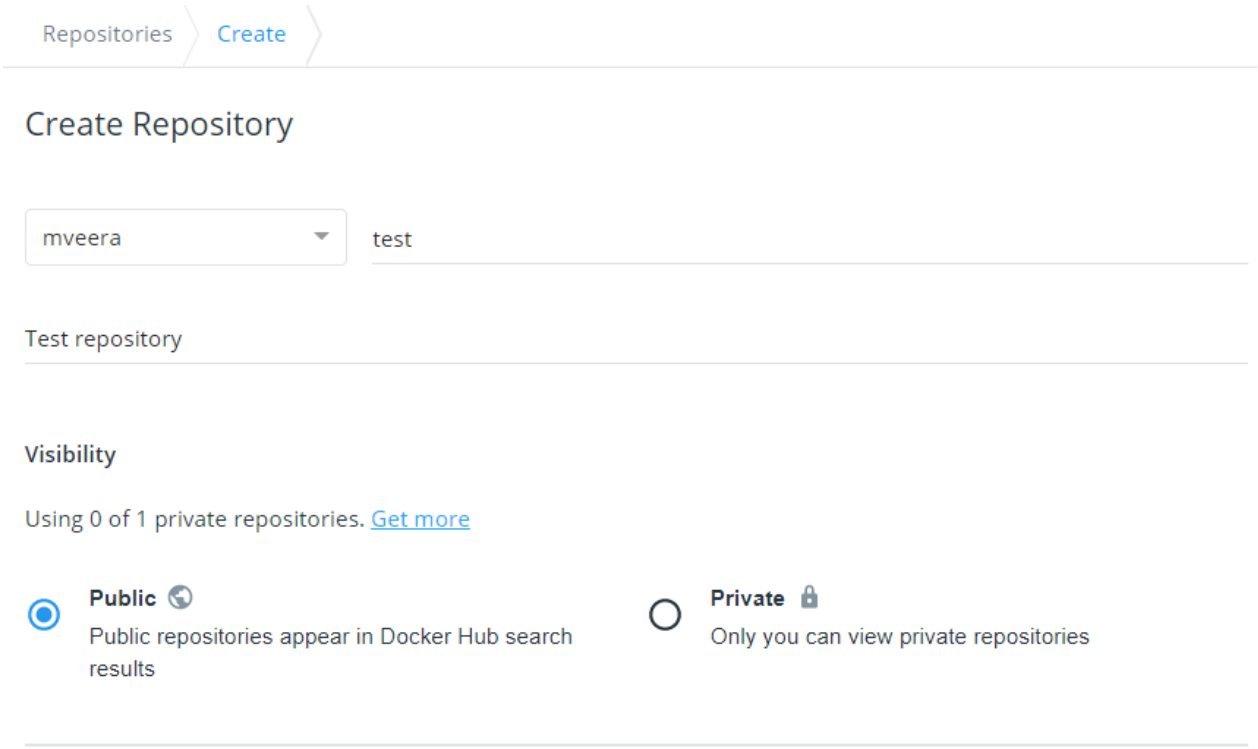
Pull the image from the registry

```
veera@ubuntu-vm:~/docker/hello_python$ docker pull localhost:5000/my-image
```

Create a repository on Docker Hub

- Sign-up and create a repository
- https://hub.docker.com/

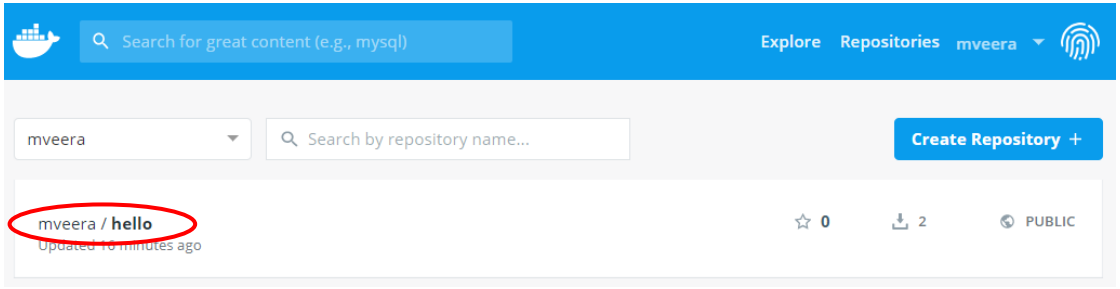




Push an image to Docker Hub

Push an image to Docker Hub

```
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello_web           latest              2c01fcf27f96        About an hour ago   927MB
hello               latest              b5df83789743        22 hours ago        918MB
veera@ubuntu-vm:~$ docker tag hello mveera/hello
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello_web           latest              2c01fcf27f96        About an hour ago   927MB
mveera/hello        latest              b5df83789743        22 hours ago        918MB
veera@ubuntu-vm:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a
Username: mveera
Password:
veera@ubuntu-vm:~$ docker push mveera/hello
The push refers to repository [docker.io/mveera/hello]
```



Find the image in Docker Hub

Pull the image from Docker Hub

```
veera@ubuntu-vm:~$ docker image rm mveera/hello
veera@ubuntu-vm:~$ docker image pull mveera/hello
veera@ubuntu-vm:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello_web           latest              2c01fcf27f96        About an hour ago   927MB
mveera/hello        latest              b5df83789743        22 hours ago        918MB
veera@ubuntu-vm:~$ docker run mveera/hello
hello
```

docker inspect

- Show low-level information on Docker objects (image, container, network, volume, etc.)

```
docker inspect [OPTIONS] NAME|ID
```

```
pom@X280:~/docker$ docker image ls nginx
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    2b7d6430f78d   2 weeks ago   142MB
pom@X280:~/docker$ docker inspect 2b7d6430f78d
[
    {
        "Id": "sha256:2b7d6430f78d432f89109b29d88d4c36c868cdbf15dc31d2132ceaa02b993763",
```

```
pom@X280:~/docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                    NAMES
44407e86ac38   nginx     "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   0.0.0.0:4000->80/tcp   nginx
pom@X280:~/docker$ docker inspect 44407e86ac38
[
    {
        "Id": "44407e86ac388d93da1ed2a0f06186d2c1766d1010798b11c4bb15d26f4b6155",
```

```
pom@X280:~/docker$  docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'  44407e86ac38
172.17.0.2
```

https://docs.docker.com/engine/reference/commandline/inspect/#examples

Debugging Commands

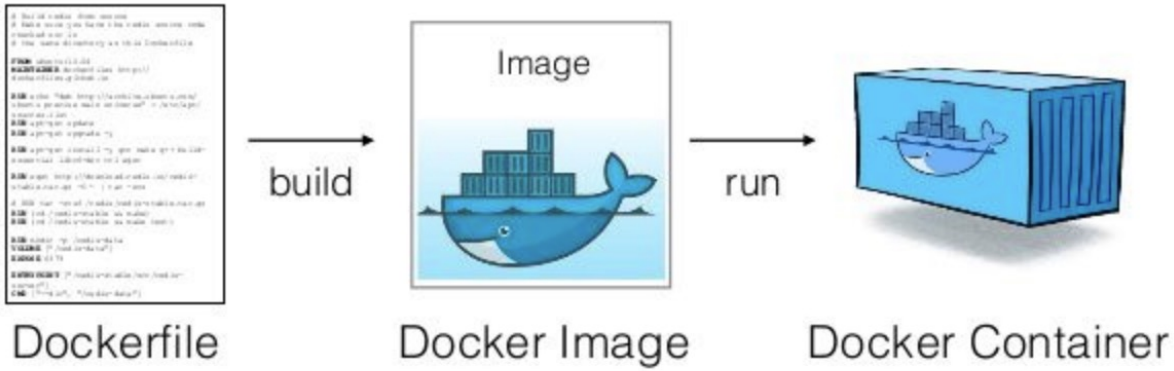
```
## Run a shell in a container
docker exec -it CONTAINER bash

## Show logs
docker logs CONTAINER
```

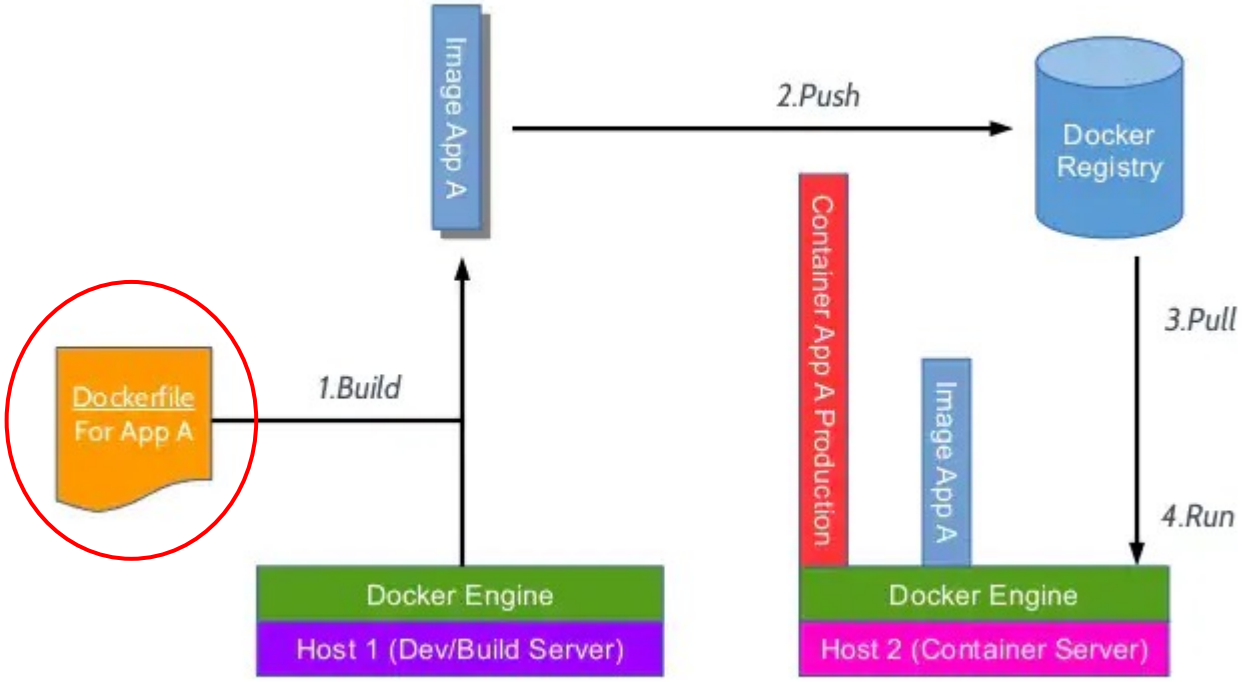
Dockerfile

Dockerfile

- A Dockerfile is a text file that tells Docker engine how to build an image.
  - Run commands
  - Add files or directories
  - Create environment variables
  - What to run when launching container



Dockerfile Usecase



Dockerfile Instructions

FROM — specifies the base (parent) image. Always the first instruction in a dockerfile.
COPY — copies files and directories to the container.
ADD — copies files and directories to the container.
WORKDIR — sets the working directory for the instructions that follow.
RUN —runs a command and creates an image layer. Used to install packages into containers.
CMD — provides a command and arguments for an executing container. CMD is executed when no command is given to *docker run*.
ENTRYPOINT — provides command and arguments for an executing container. Unlike CMD, ENTRYPOINT is always executed.
ENV — sets a persistent environment variable.
ARG — defines a variable to pass to Docker at build-time.
EXPOSE — exposes a port.
VOLUME — creates a directory mount point to access and store persistent data.
LABEL —provides metadata including maintainer info.

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Build and run an image

Create a file named “dockerfile” containing:

```
FROM ubuntu

CMD ["bash"]
```

Run these commands in the same directory as docker file

```
docker build -t ubuntu_bash .
docker image ls
docker run -it ubuntu_bash
```

```
docker build -t ubuntu_bash .
```

Tell docker engine to create a new image

-t give this image a tag for later reference

. use the current directory to find the Dockerfile

Build and run an image

```
veera@ubuntu-vm:~/docker$ cat dockerfile
FROM ubuntu
CMD ["bash"]

veera@ubuntu-vm:~/docker$ docker build -t ubuntu_bash .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM ubuntu
 ---> a2a15febcdf3
Step 2/2 : CMD ["bash"]
 ---> Running in e3bffa3d087c
Removing intermediate container e3bffa3d087c
 ---> c7824c8113f5
Successfully built c7824c8113f5
Successfully tagged ubuntu_bash:latest
veera@ubuntu-vm:~/docker$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu_bash         latest              c7824c8113f5        21 seconds ago      64.2MB
python              latest              780bbecceca6        32 hours ago        919MB
ubuntu              latest              a2a15febcdf3        4 weeks ago         64.2MB
hello-world         latest              fce289e99eb9        8 months ago        1.84kB
veera@ubuntu-vm:~/docker$ docker run -it ubuntu_bash
root@a002cb5b70b2:/#
```

Build image of Ubuntu + tools

dockerfile

```
FROM ubuntu

# add tools

RUN apt-get update

RUN apt-get install -y iputils-ping

RUN apt-get install -y net-tools

RUN apt-get install -y iproute2

RUN apt-get install -y curl

CMD ["bash"]
```

Useful commands and packages

- ping (package iputils-ping)
- ifconfig (package net-tools)
- ip (package iproute2)
- curl (package curl)

```
veera@ubuntu-vm:~/docker/ubuntu_tools$ docker build -t ubuntu_tools .
Sending build context to Docker daemon  2.048kB
Step 1/7 : FROM ubuntu
 ---> a2a15febcdf3
```

```
Step 7/7 : CMD ["bash"]
 ---> Running in 02414ac7fa3d
Removing intermediate container 02414ac7fa3d
 ---> fe8fe2ef9bff
Successfully built fe8fe2ef9bff
Successfully tagged ubuntu_tools:latest
```

```
veera@ubuntu-vm:~/docker/ubuntu_tools$ docker image ls
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
ubuntu_tools                        latest    fe8fe2ef9bff   3 minutes ago    122MB
veera@ubuntu-vm:~/docker/ubuntu_tools$ docker run -it --name ubuntu_tools ubuntu_tools
root@8ce4e32704ae:/#
```

Building an image for a python program

Create *hello.py* in directory *hello*

```
print("hello")
```

Create *dockerfile* in directory *hello*

```
# Get the ubuntu image from Docker Hub
FROM python:slim
# Copy current folder which contains python code
# into directory /usr/src/hello of the Docker image
COPY . /usr/src/hello
# Specify working directory
WORKDIR /usr/src/hello
# Run program
CMD ["python", "hello.py"]
```

Run these commands in directory *hello*

```
cd hello
docker build -t hello .
docker run hello
```

Building an image for a python web app

Create *index.py* in directory *hello_web*

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello World!"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int("5000"), debug=True)
```

Create *dockerfile* in directory *hello_web*

```
FROM python:slim
COPY . /app
WORKDIR /app
RUN pip install flask
EXPOSE 5000
ENTRYPOINT ["python", "./index.py"]
```

Run these commands in directory *hello_web*

```
cd hello_web
docker build -t hello_web .
docker run --name hello_web --detach -p 5000:5000 hello_web
```

Open http://localhost:5000 on a web browser

Run another web app on another port

```
docker run --name hello_web_2 --detach -p 80:5000 hello_web
```

```
veera@ubuntu-vm:~/docker/hello_web$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
f1bbdc530e92        hello_web           "/bin/sh -c 'python …"   4 minutes ago       Up 4 minutes        0.0.0.0:80->5000/tcp     hello_web_2
7fceda6a8b38        hello_web           "/bin/sh -c 'python …"   23 minutes ago      Up 23 minutes       0.0.0.0:5000->5000/tcp   hello_web
```

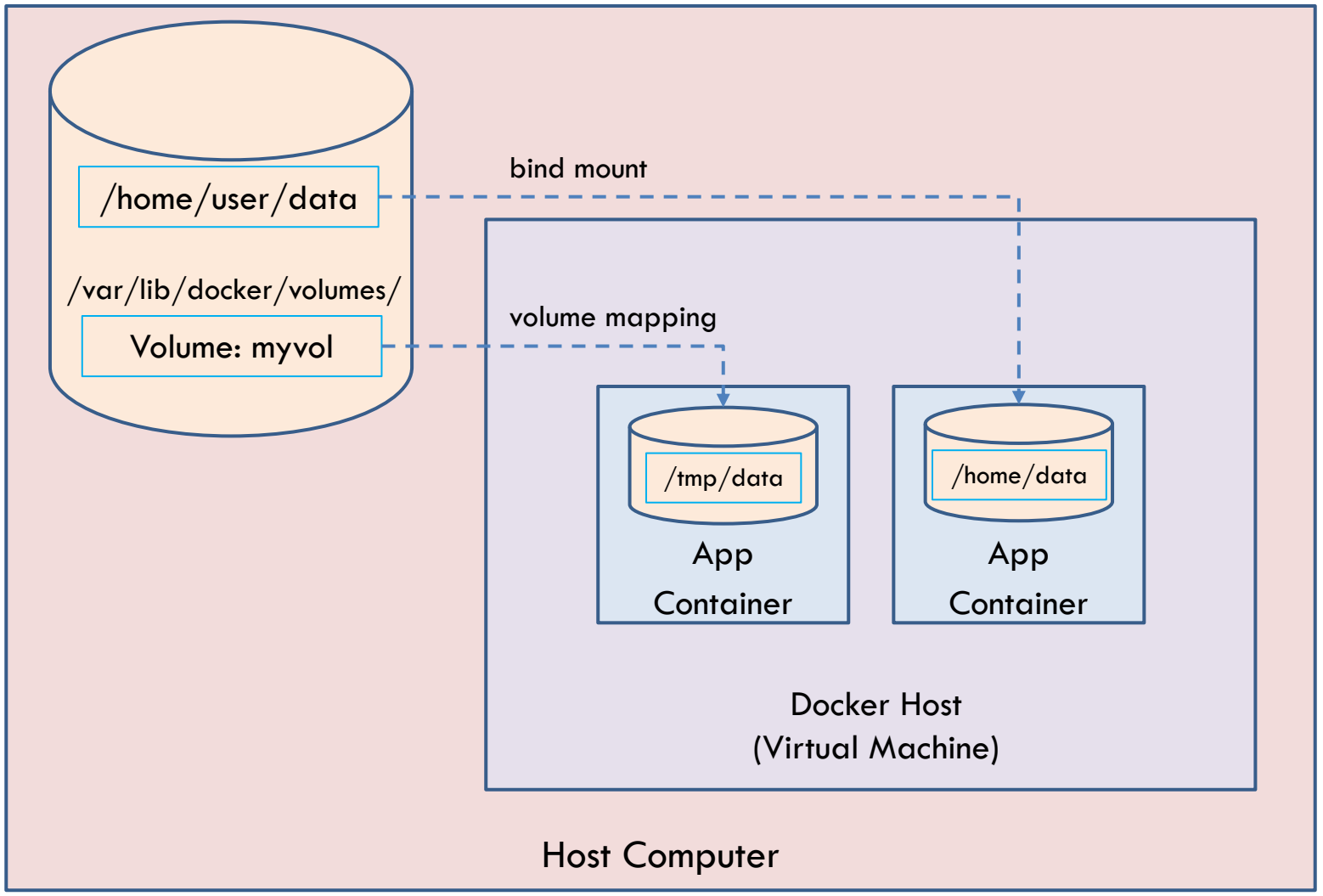
Open http://localhost/ on a web browser

Persistent Storage

Problem

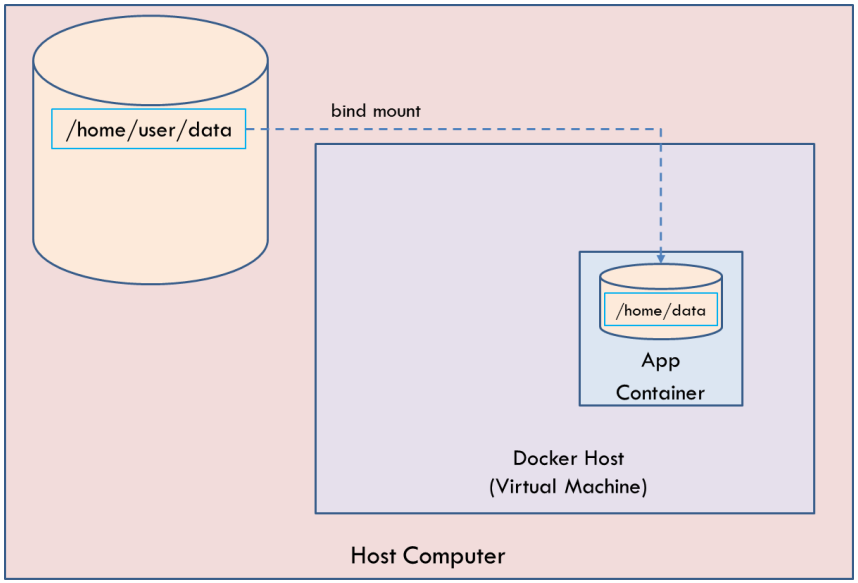
- Files in a container will disappear when the container stops.
- To keep the files, they must be stored in a persistent storage.
- Docker provides two methods
  - Bind mount
  - Volume mapping

Persistent Data



Method 1: Bind Mount

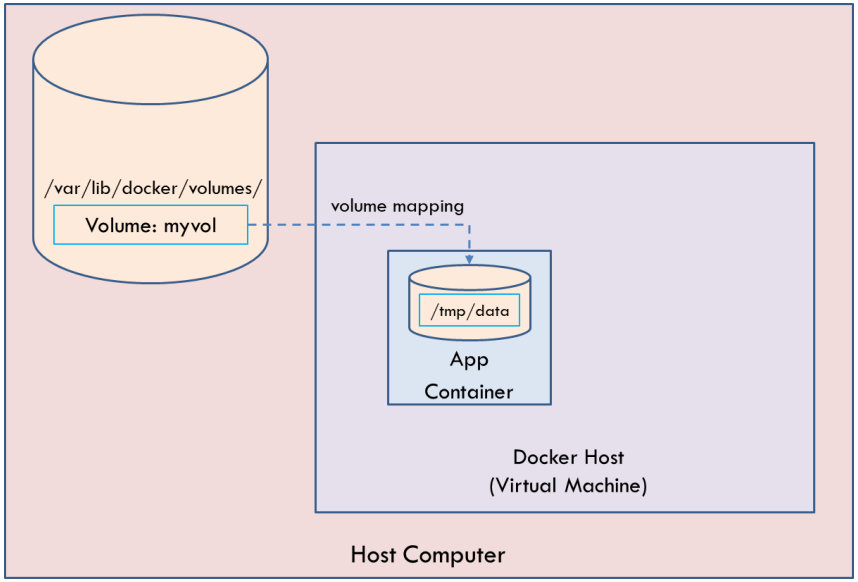
- Mount host directory to container directory




```
veera@ubuntu-vm:~$ ls /home/veera/container/
veera@ubuntu-vm:~$
veera@ubuntu-vm:~$ docker run --name ubuntu -it -v /home/veera/container:/home ubuntu
root@3cfc393b56a3:/# cat > /home/test.txt
test
root@3cfc393b56a3:/# exit
exit
veera@ubuntu-vm:~$ docker rm ubuntu
ubuntu
veera@ubuntu-vm:~$ cat /home/veera/container/test.txt
test
```

Method 2: Volume Mapping

- Volume is a persistent storage location created outside container.
- Volumes are stored in host's filesystem and managed by Docker.
- Recommended option



Method 2: Volume Mapping

```
docker volume create myvol
docker run --name ubuntu -it -v myvol:/home ubuntu
```

```
veera@ubuntu-vm:~$ docker volume create myvol
myvol
veera@ubuntu-vm:~$ docker volume ls
DRIVER              VOLUME NAME
local               myvol
veera@ubuntu-vm:~$ docker volume inspect myvol
[
    {
        "CreatedAt": "2019-09-16T00:01:18+07:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvol/_data",
        "Name": "myvol",
        "Options": {},
        "Scope": "local"
    }
]
veera@ubuntu-vm:~$ docker run --name ubuntu -it -v myvol:/home ubuntu
root@c376a358e77a:/# cat > /home/test
test
root@c376a358e77a:/# exit
exit
veera@ubuntu-vm:~$ docker rm ubuntu
ubuntu
veera@ubuntu-vm:~$ sudo cat /var/lib/docker/volumes/myvol/_data/test
test
```

Docker Networking

Docker Network

- Each new Docker container is automatically assigned an IP address.

```
veera@ubuntu-vm:~$ docker container inspect nginx | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
```

IP of container

Publish container ports

- A container is isolated by default. So, it does not allow external access unless it explicitly publishes specific ports.
- Multiple containers may use the same port numbers (e.g. default ports for web server = 80, MySQL = 3306), so they must be mapped to different ports on the host.
- `docker run -p [host_port]:[container_port] IMAGE`

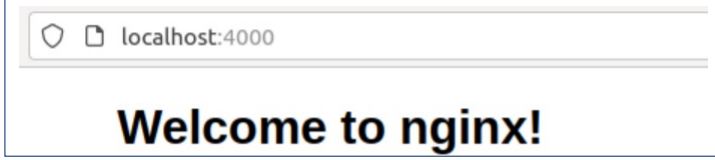
Example

Create a nginx container with host port = 4000 and container port = 80

docker run -d --name nginx -p 4000:80 nginx

```
veera@ubuntu-vm:~$ docker run -d --name nginx -p 4000:80 nginx
533898a25687d5cda2911b935053c9740d646984d99025e2e2a3254b7f1bd6b4
veera@ubuntu-vm:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
533898a25687   nginx          "/docker-entrypoint.…"   18 seconds ago   Up 16 seconds   0.0.0.0:4000->80/tcp, :::4000->80/tcp   nginx
```

Connect to the web server from the host via port 4000



Connect to the web server from the host via port 4000

```
veera@ubuntu-vm:~$ curl localhost:4000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Connect to the web server from the same container via port 80

```
veera@ubuntu-vm:~/docker$ docker exec -it nginx bash
root@0f5e4c2df1fb:/# curl localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Ex. Create a Web server container

Create a web server container (nginx)

```
pom@X280:~/docker$ docker run -d -p 4000:80 --name nginx nginx
```

```
pom@X280:~/docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                   NAMES
9dc8ed7ab142   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:4000->80/tcp    nginx
```

```
pom@X280:~/docker$ docker container inspect nginx | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
```

Ex. Create Ubuntu + tools container

dockerfile

```
FROM ubuntu
# add tools
RUN apt-get update
RUN apt-get install -y iputils-ping
RUN apt-get install -y net-tools
RUN apt-get install -y iproute2
RUN apt-get install -y curl
CMD ["bash"]
```

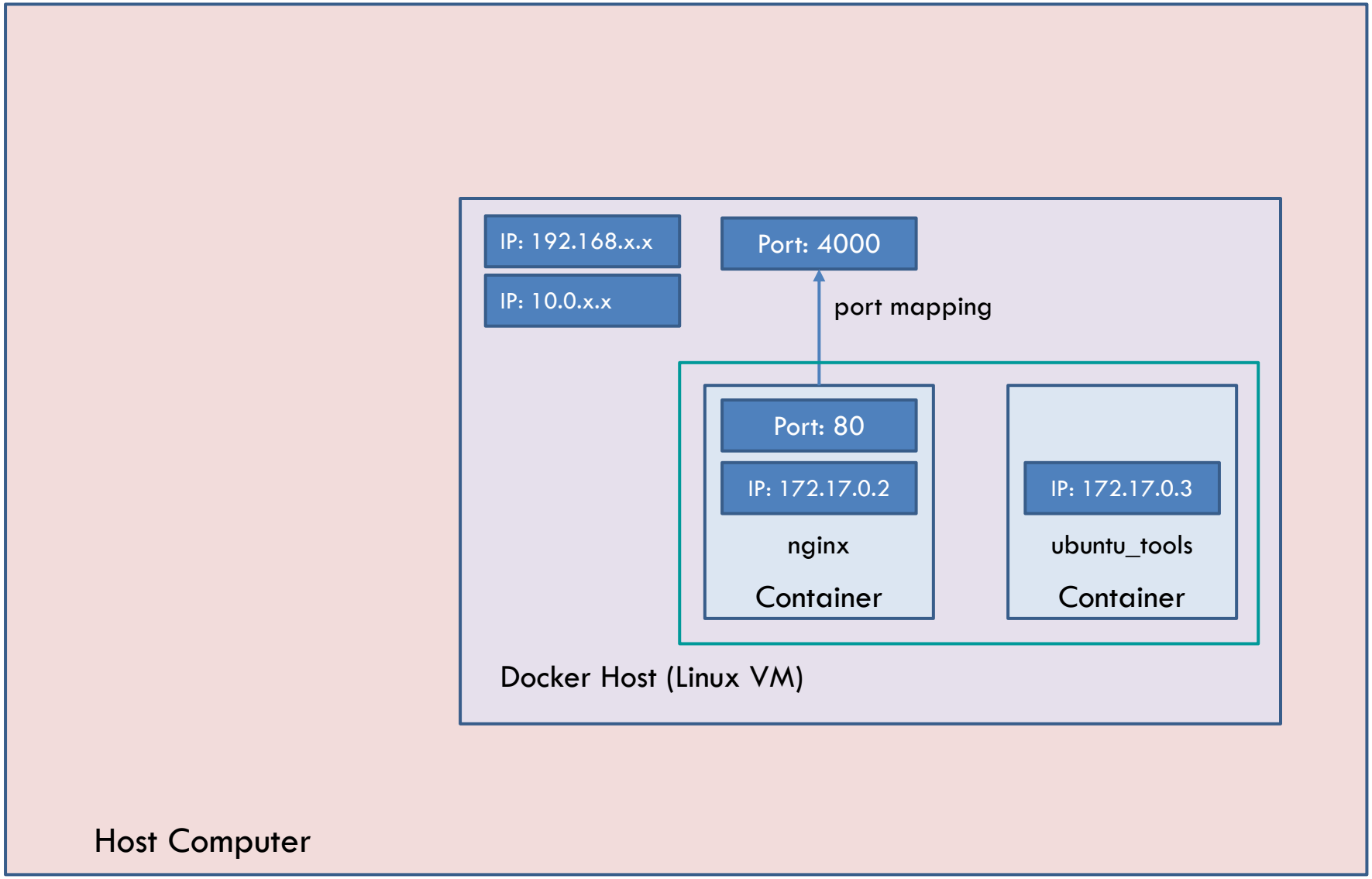
Useful commands and packages

- ping (package iputils-ping)
- ifconfig (package net-tools)
- ip (package iproute2)
- curl (package curl)

```
pom@X280:~/docker/ubuntu_tools$ docker build -t ubuntu_tools .
```

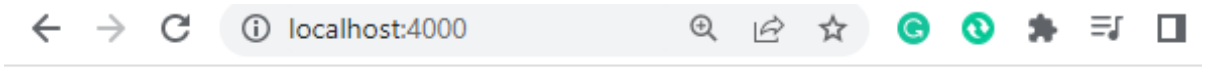
```
pom@X280:~/docker/ubuntu_tools$ docker run -it --name ubuntu_tools ubuntu_tools
root@863d667ab61d:/# ifconfig | grep inet
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
```

Ex: Web server & browser



Access the web server container

via a web browser on host computer → localhost:4000



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

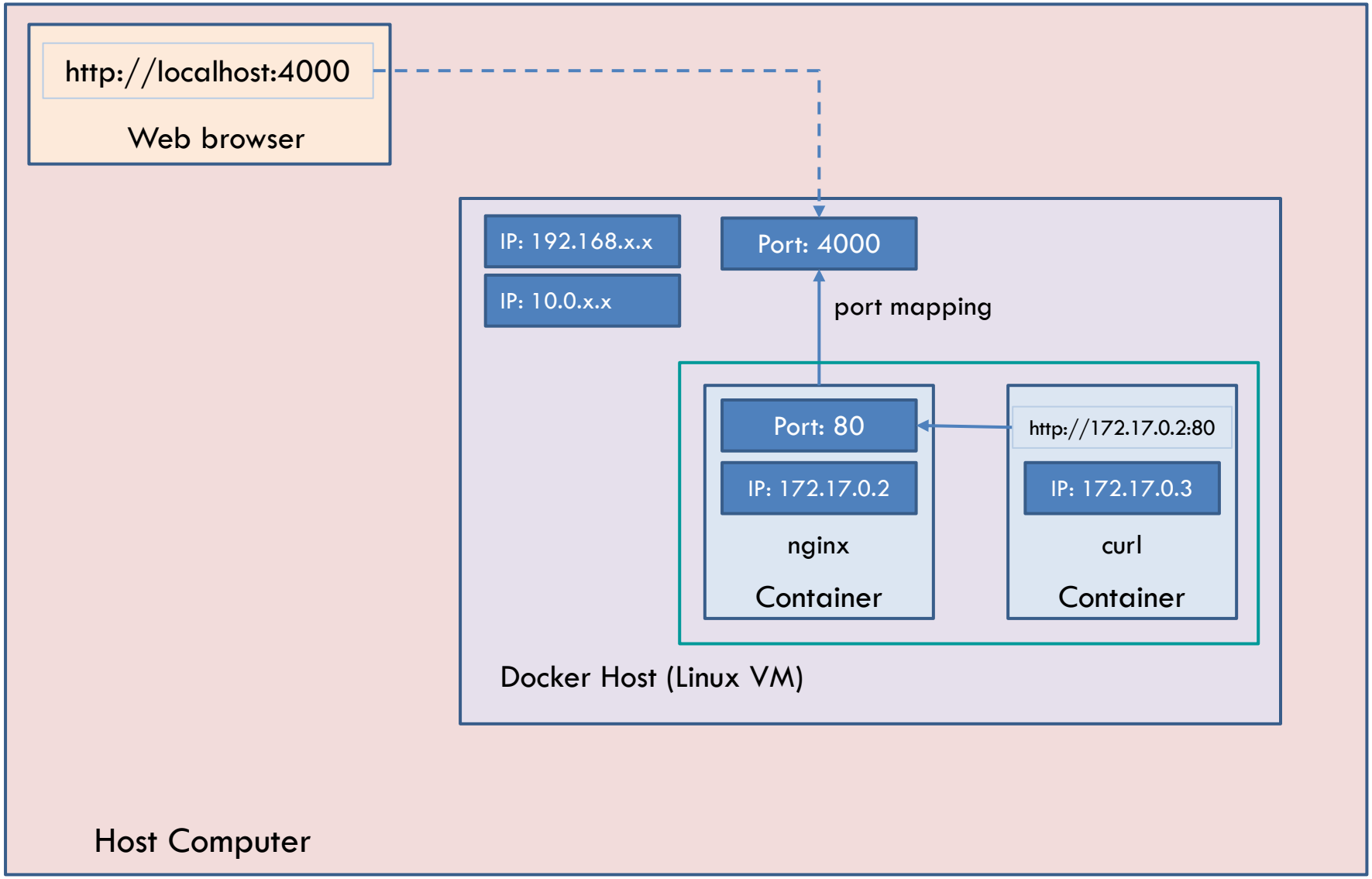
via a web browser (curl) on another container → 172.17.0.2:80

On ubuntu_tools container:

```
root@863d667ab61d:/# curl 172.17.0.2:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

```
root@863d667ab61d:/# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.238 ms
```

Ex: Web server & browser



They don't know each other by name.

Web browser cannot access web server via hostname or container ID.

On ubuntu_tools container:

```
root@863d667ab61d:/# curl nginx:80
curl: (6) Could not resolve host: nginx
```

```
root@863d667ab61d:/# curl 9dc8ed7ab142:80
curl: (6) Could not resolve host: 9dc8ed7ab142
root@863d667ab61d:/# ping 9dc8ed7ab142
ping: 9dc8ed7ab142: Name or service not known
```

`/etc/hosts` file contains a mapping of IP addresses to hostnames.
No mapping to Web server (nginx)

```
root@863d667ab61d:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.3      863d667ab61d
```

Link

Run container with `--link <name or id>[:alias]` option to link to another container.

```
pom@X280:~/docker/ubuntu_tools$ docker run -it --name ubuntu_tools2 --link nginx ubuntu_tools
```

Now Web browser can access web server via hostname or container ID.

On ubuntu_tools2 container:

```
root@96f27af36283:/# curl nginx
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

```
root@96f27af36283:/# curl 9dc8ed7ab142
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

How does --link work?

`/etc/hosts` now contains a mapping of IP address to hostname of Web server (nginx)

On ubuntu_tools2 container:

```
root@96f27af36283:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      nginx 9dc8ed7ab142
172.17.0.4      96f27af36283
```

However, Web server still don't know web browser by hostname.

In other words, *--link* is one-way mapping.

On nginx container:

```
root@9dc8ed7ab142:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      9dc8ed7ab142
```

Note: --link option is a legacy feature of Docker.
It will be replaced by user-defined networks.

Docker Networks

```
pom@X280:~$ docker network ls
NETWORK ID     NAME                                      DRIVER    SCOPE
08d98d8ae48b   bridge                                    bridge    local
e1eeed546e58   host                                      host      local
731053211c6d   none                                      null      local
```

Bridge Network

- A bridge network allows containers connected to the same bridge network to communicate. Containers on different bridge networks cannot communicate directly with each other.

- When you start Docker, a default bridge network (also called *bridge*) is created automatically, and newly-started containers connect to it unless otherwise specified.

- Bridge networks apply to containers running on the same Docker daemon host.

```
pom@X280:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "08d98d8ae48b66016b32e520f228556dbc42903751b093793fff391819c5a97b",
        "Created": "2022-09-10T07:10:43.4425995Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "863d667ab61d8f6f3b2ad89621b63ae96844e4f454ff4e06ab1bfc109be95f0e": {
                "Name": "ubuntu_tools",
                "EndpointID": "e9ecf44ec11b345ebe8b35932021cb76a9ea30d3c6a2e202a4596e0c81eb0cb5",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "9dc8ed7ab142037e81fa769077dc46b0aa89d91efeaaa85055edefd6d3a5970d": {
                "Name": "nginx",
                "EndpointID": "71c31d55143aeb153077525f57501fbdcbf8231bd8d3bade6cc79286c201a098",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
```

Containers that attach to this network

User-defined bridge network

- We can also create user-defined bridge networks.

- User-defined bridges provide automatic DNS resolution between containers.
  - Containers on the *default bridge network* can only access each other by IP addresses, unless you use the --link option.
  - On a *user-defined bridge network*, containers can resolve each other by name or alias.

Creating a user-defined bridge network

```
pom@X280:~$ docker network create mynet
045e8187e372b7778af6d8517a096e52032ca69605b23dc65055187801023090
pom@X280:~$ docker network ls
NETWORK ID     NAME                                      DRIVER    SCOPE
08d98d8ae48b   bridge                                    bridge    local
e1eeed546e58   host                                      host      local
045e8187e372   mynet                                     bridge    local
731053211c6d   none                                      null      local
```

User-defined bridge network

```
pom@X280:~$ docker network inspect mynet
[
    {
        "Name": "mynet",
        "Id": "045e8187e372b7778af6d8517a096e52032ca69605b23dc65055187801023090",
        "Created": "2022-09-11T08:00:40.8564448Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Connect existing container to network

```
pom@X280:~$ docker network connect mynet nginx
pom@X280:~$ docker inspect nginx
```

Scroll down to the "Networks" section

```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "08d98d8ae48b66016b32e520
        "EndpointID": "71c31d55143aeb153077525
        "Gateway": "172.17.0.1",
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:02",
        "DriverOpts": null
    },
```

```
    "mynet": {
        "IPAMConfig": {},
        "Links": null,
        "Aliases": [
            "9dc8ed7ab142"
        ],
        "NetworkID": "045e8187e372b7778af6d851
        "EndpointID": "802002b4cbe90eba34ff64c
        "Gateway": "172.18.0.1",
        "IPAddress": "172.18.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:12:00:02",
        "DriverOpts": {}
    }
}
```

Connect existing container to network

```
pom@X280:~$ docker network connect mynet ubuntu_tools
pom@X280:~$ docker inspect ubuntu_tools
```

Scroll down to the "Networks" section

```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "08d98d8ae48b66016b32e520f
        "EndpointID": "e9ecf44ec11b345ebe8b3593
        "Gateway": "172.17.0.1",
        "IPAddress": "172.17.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:03",
        "DriverOpts": null
    },
```

```
    "mynet": {
        "IPAMConfig": {},
        "Links": null,
        "Aliases": [
            "863d667ab61d"
        ],
        "NetworkID": "045e8187e372b7778af6d8517
        "EndpointID": "981b5a05680f99898fe50c73
        "Gateway": "172.18.0.1",
        "IPAddress": "172.18.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:12:00:03",
        "DriverOpts": {}
    }
}
```

Connect new container to network

Docker run command with `--network network`

```
pom@X280:~$ docker run -it --name ubuntu_tools3 --network mynet ubuntu_tools
root@f6deac8fc947:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.0.4  netmask 255.255.0.0  broadcast 172.18.255.255
```

New container can connect to web server by hostname

```
root@f6deac8fc947:/# curl nginx
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Note: The new container does not connect to the default “bridge” network.

```
pom@X280:~$ docker network inspect mynet
[
    {
        "Name": "mynet",
        "Id": "045e8187e372b7778af6d8517a096e52032ca69605b23dc65055187801023090",
        "Created": "2022-09-11T08:00:40.8564448Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "863d667ab61d8f6f3b2ad89621b63ae96844e4f454ff4e06ab1bfc109be95f0e": {
                "Name": "ubuntu_tools",
                "EndpointID": "981b5a05680f99898fe50c7315b8ca49f932320c5edd7aef9d3a504c426725a0",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "9dc8ed7ab142037e81fa769077dc46b0aa89d91efeaaa85055edefd6d3a5970d": {
                "Name": "nginx",
                "EndpointID": "802002b4cbe90eba34ff64c9951dce0775c685997d3e9f9f07351c142974bf3a",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "f6deac8fc9473a64752c40ae304ac6ea173a899aef1659ad22030959107e301a": {
                "Name": "ubuntu_tools3",
                "EndpointID": "0ebf7abf0ad398d0784a27a4ff957cc7c63702565a58559c81d54dca498e4083",
                "MacAddress": "02:42:ac:12:00:04",
                "IPv4Address": "172.18.0.4/16",
                "IPv6Address": ""
            }
```

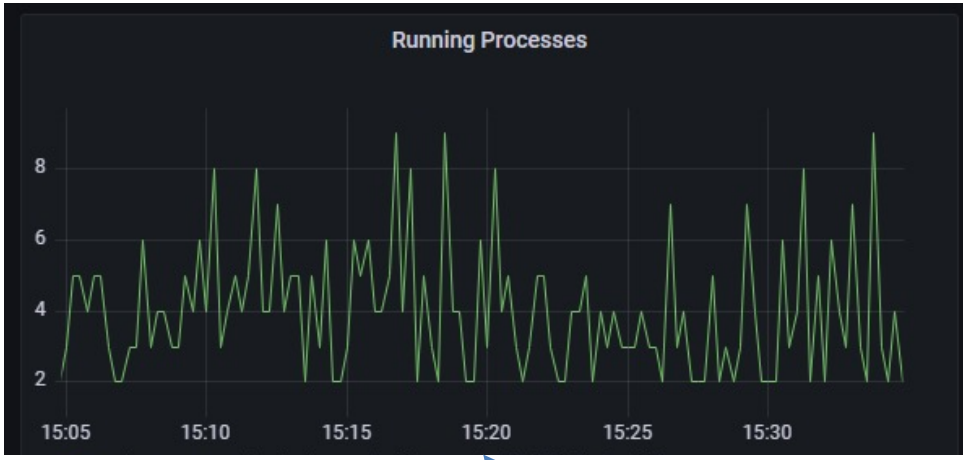
Disconnect from network

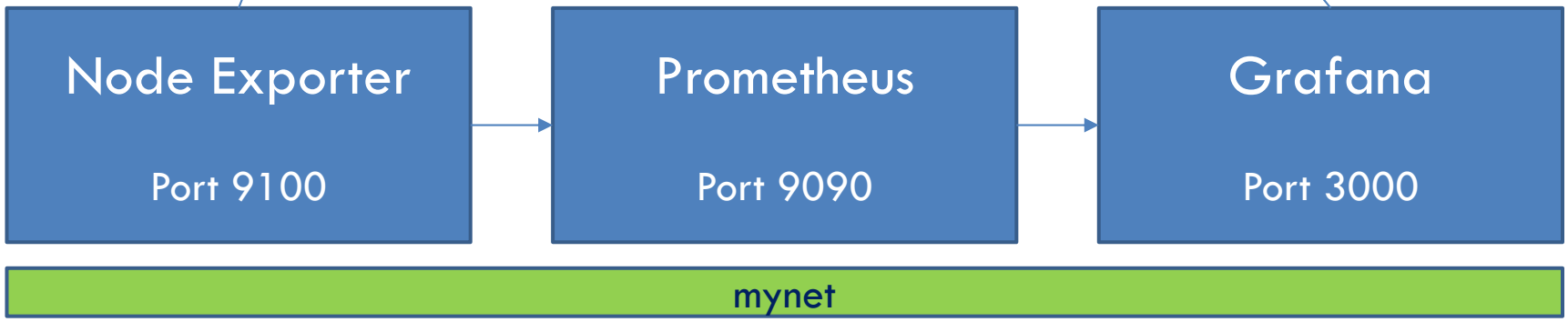
```
pom@X280:~$ docker network disconnect bridge nginx
pom@X280:~$ docker network disconnect bridge ubuntu_tools
```

```
pom@X280:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "08d98d8ae48b66016b32e520f228556dbc42903751b093793fff391819c5a97b",
        "Created": "2022-09-10T07:10:43.4425995Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
```

Example: Resource Monitoring

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.6e-05
go_gc_duration_seconds{quantile="0.25"} 9.52e-05
go_gc_duration_seconds{quantile="0.5"} 0.0001528
go_gc_duration_seconds{quantile="0.75"} 0.0001877
go_gc_duration_seconds{quantile="1"} 0.0012912
go_gc_duration_seconds_sum 0.0047064
go_gc_duration_seconds_count 22
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 4.847872e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 5.20074e+07
```

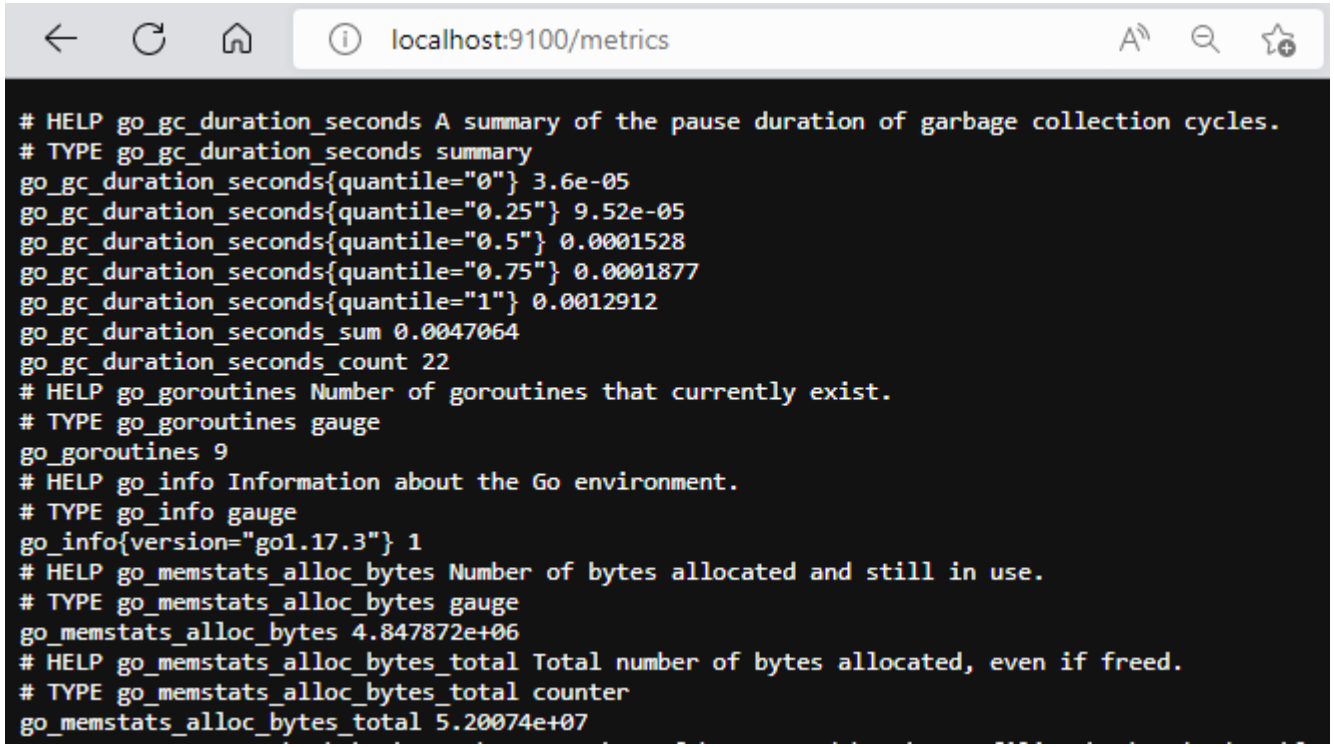




Node Exporter

- Node Exporter is a program that collects hardware and OS metrics. It is used for measuring various server resources such as RAM, disk space, and CPU utilization.

```
pom@X280:~/docker/monitoring$ docker run -d -p 9100:9100 --network mynet --name node-exporter prom/node-exporter
```



Refresh to see changes

Prometheus

- Prometheus is a monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results.

prometheus.yml

```
global:
  scrape_interval: 15s

scrape_configs:
- job_name: node
  static_configs:
  - targets: ['node-exporter:9100']
```

Scrape data from node-exporter

```
pom@X280:~/docker/monitoring$ docker run -d -p 9090:9090 --network mynet --name prometheus \
> -v /home/pom/docker/monitoring/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
```

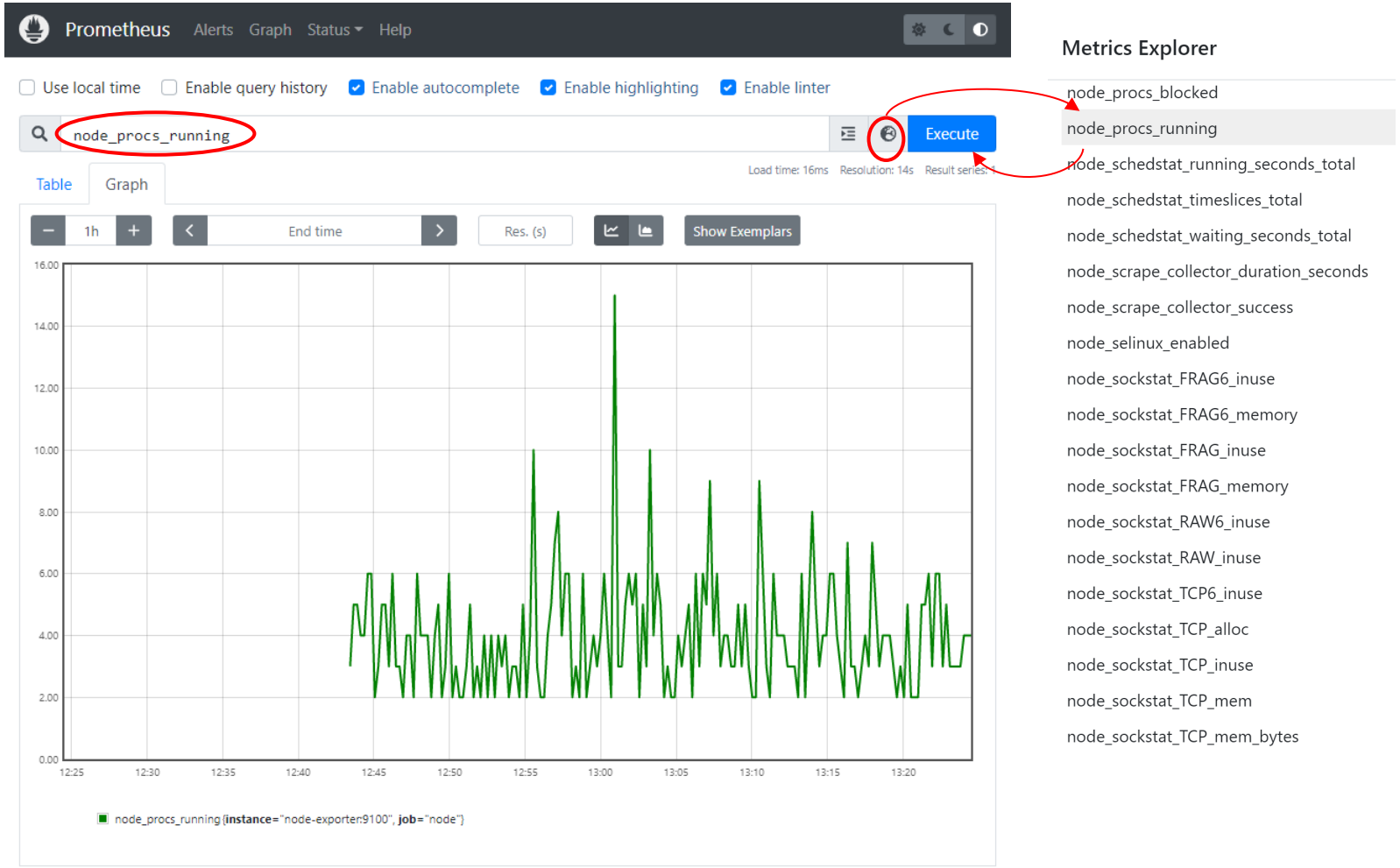
or

```
pom@X280:~/docker/monitoring$ docker run -d -p 9090:9090 --network mynet --name prometheus \
> -v /home/pom/docker/monitoring:/etc/prometheus prom/prometheus
```

Prometheus GUI

http://localhost:9090/

Select metrics to monitor



Grafana

- Grafana is an analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources.

Persistent Storage

- Grafana store dashboard database in `/var/lib/grafana`
- To keep the database in a persistent storage, a volume must be created.

```
pom@X280:~/docker/monitoring$ docker volume create grafana-vol
```

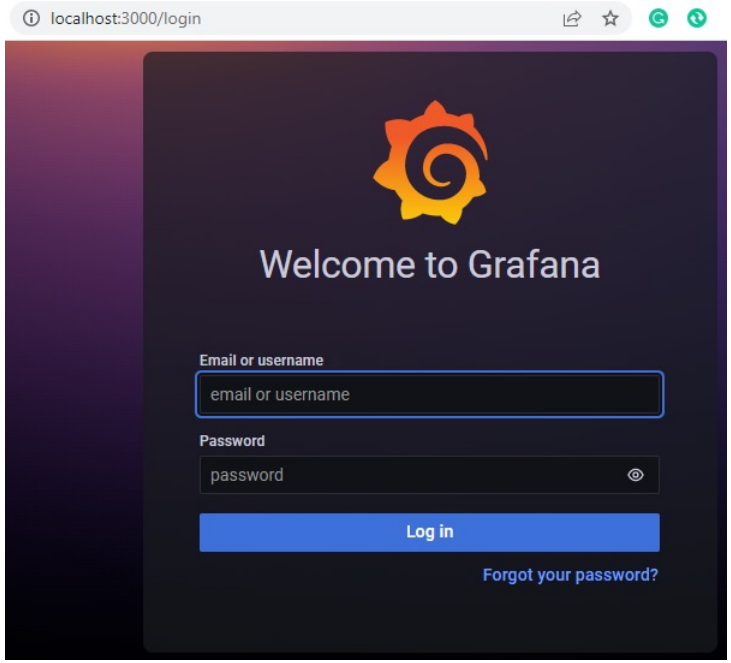
```
pom@X280:~/docker/monitoring$ docker volume ls | grep grafana
local     grafana-vol
```

```
pom@X280:~/docker/monitoring$ docker inspect grafana-vol
[
    {
        "CreatedAt": "2022-09-11T16:48:58Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/grafana-vol/_data",
        "Name": "grafana-vol",
        "Options": {},
        "Scope": "local"
    }
]
```

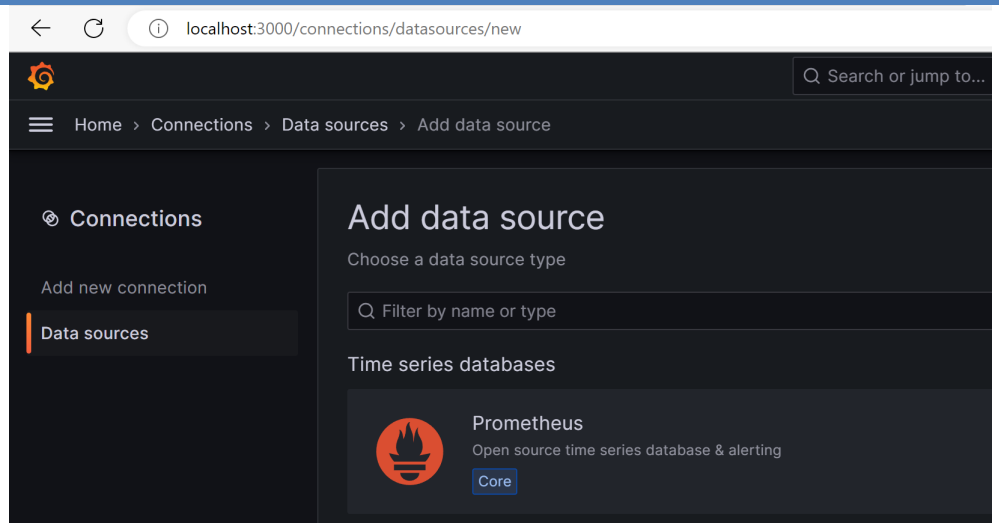
Running Grafana

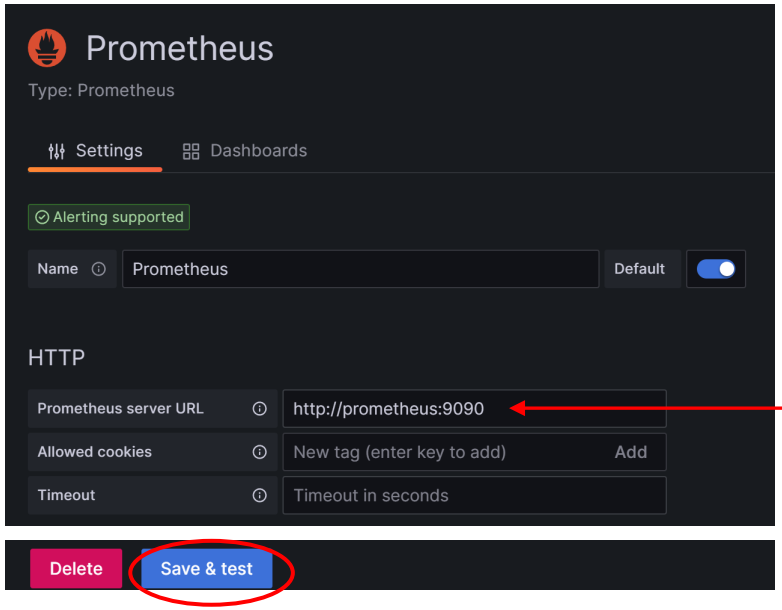
```
pom@X280:~/docker/monitoring$ docker run -d -p 3000:3000 --network mynet \
> -v grafana-vol:/var/lib/grafana --name grafana grafana/grafana
```

- http://localhost:3000
- Username: admin
- Password: admin
- When asked to change password → skip



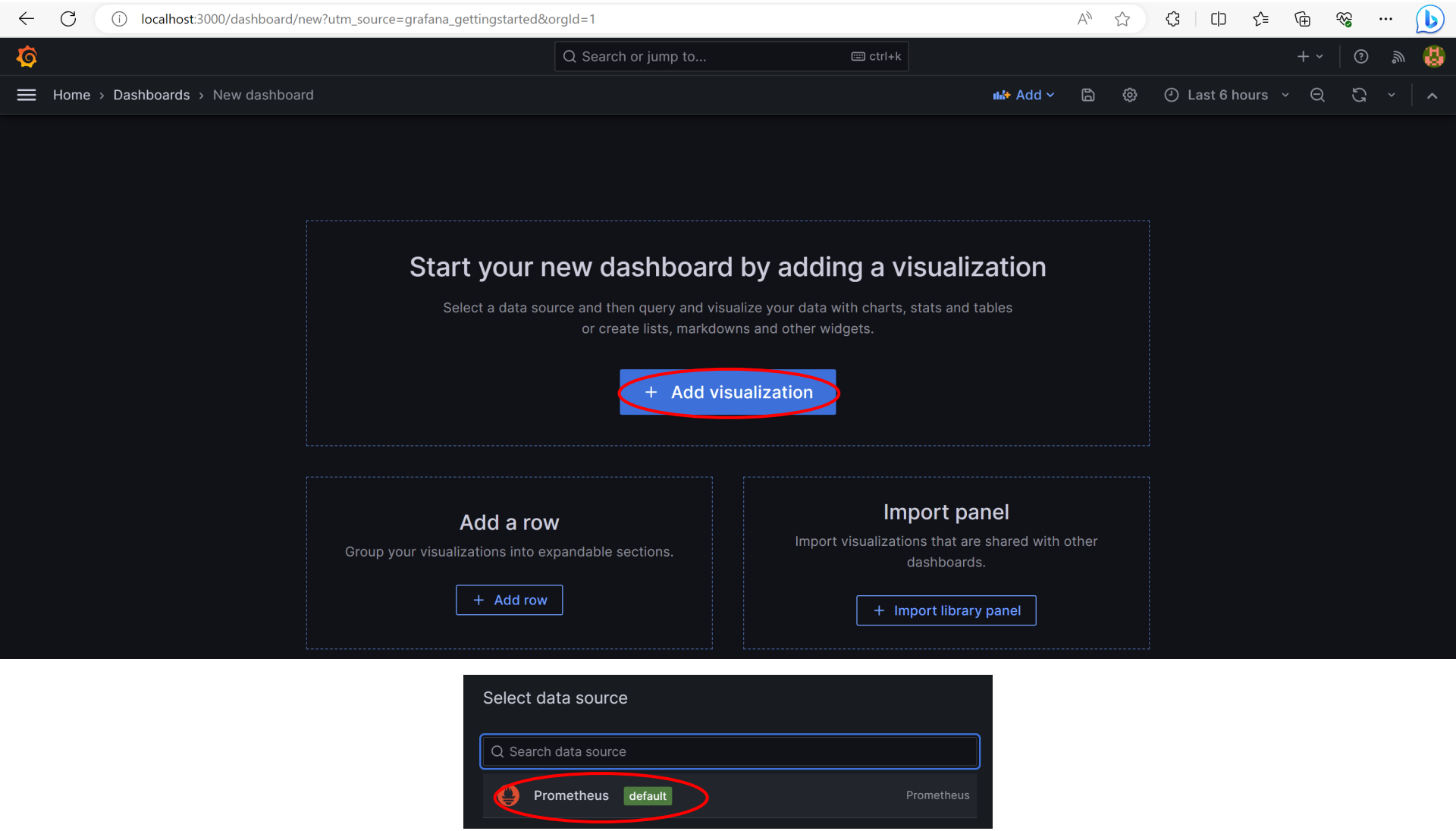
Add data source



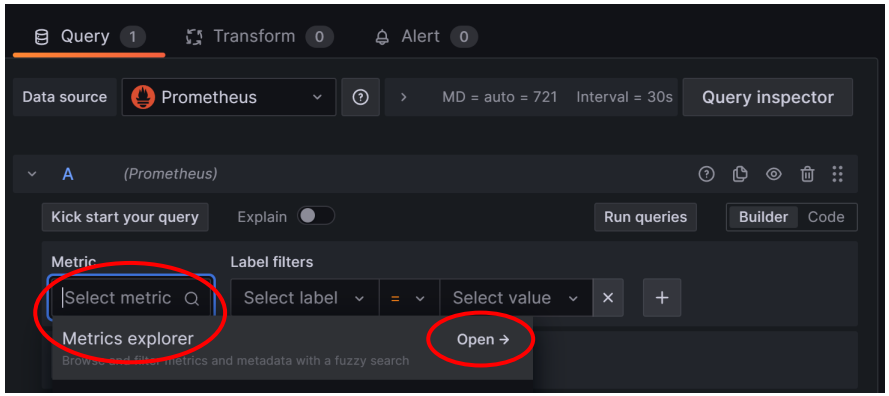


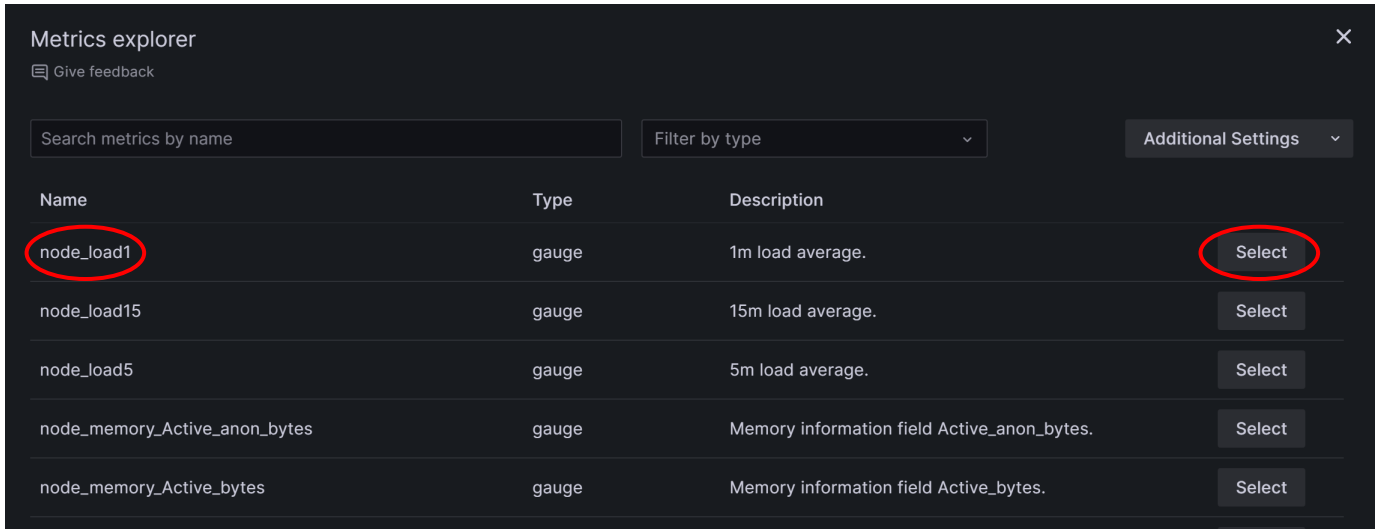
http://prometheus:9090

Create dashboard

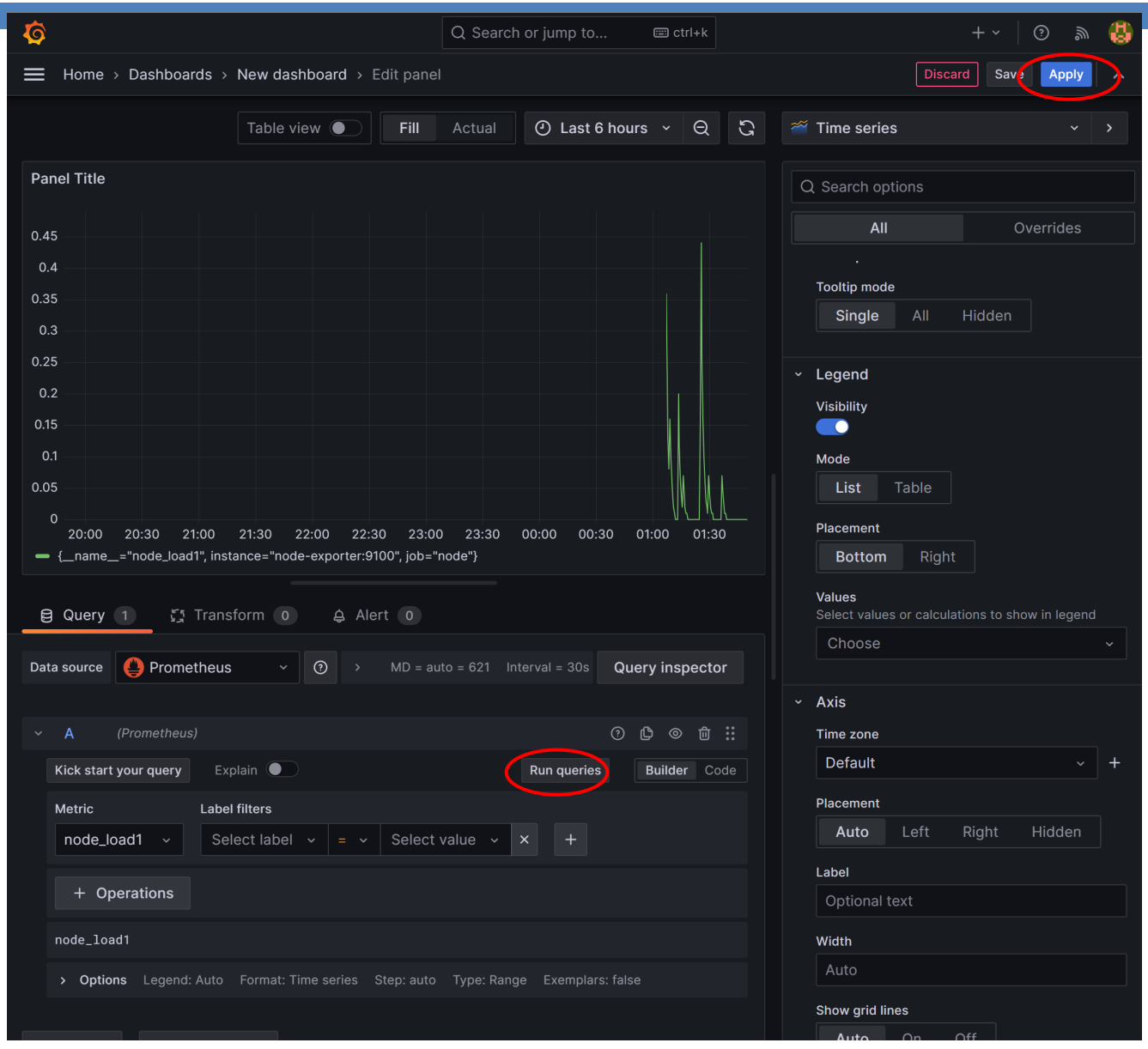


Select metric

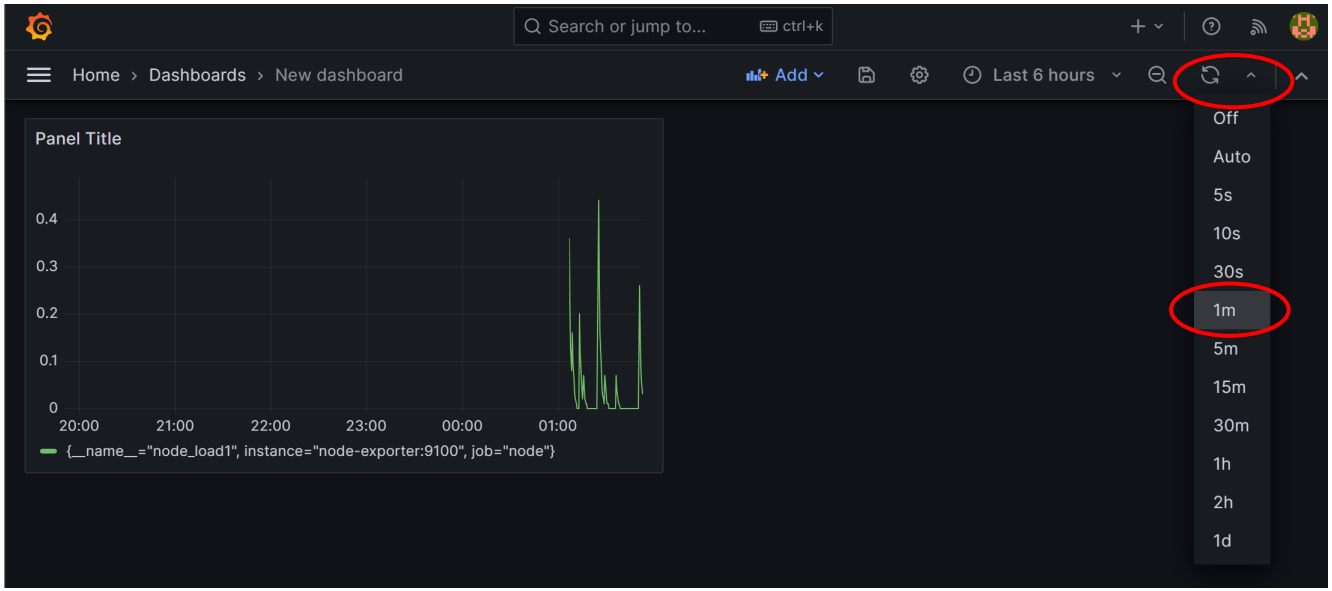




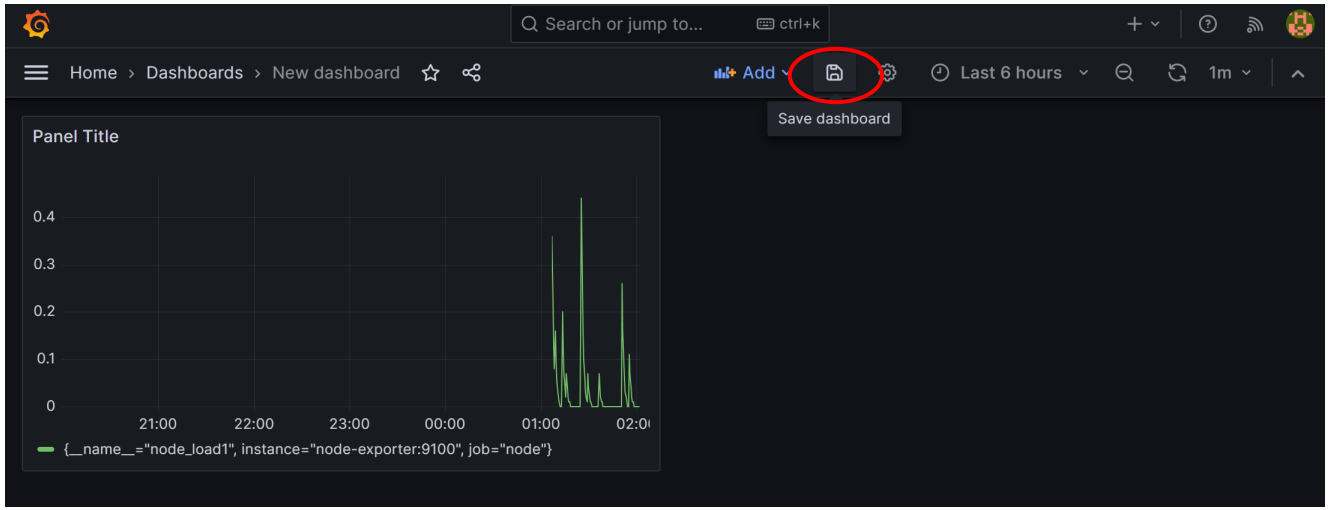
Apply Query


Search or jump to... ctrl+k
Home › Dashboards › New dashboard › Edit panel
Discard
Save
Apply
Table view
Fill
Actual
Last 6 hours
Time series
Panel Title
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30
00:00
00:30
01:00
01:30
{__name__="node_load1", instance="node-exporter:9100", job="node"}
Search options
All
Overrides
Tooltip mode
Single
All
Hidden
Legend
Visibility
Mode
List
Table
Placement
Bottom
Right
Values
Select values or calculations to show in legend
Choose
Axis
Time zone
Default
Placement
Auto
Left
Right
Hidden
Label
Optional text
Width
Auto
Show grid lines
Query 1
Transform 0
Alert 0
Data source
Prometheus
MD = auto = 621
Interval = 30s
Query inspector
A
(Prometheus)
Kick start your query
Explain
Run queries
Builder
Code
Metric
Label filters
node_load1
Select label
=
Select value
Operations
node_load1
Options
Legend: Auto
Format: Time series
Step: auto
Type: Range
Exemplars: false

Set refresh rate



Save Dashboard



Managing Multi-container Application

With Docker Compose

Problem

Docker commands (e.g. docker run, docker stop) manage each container separately.

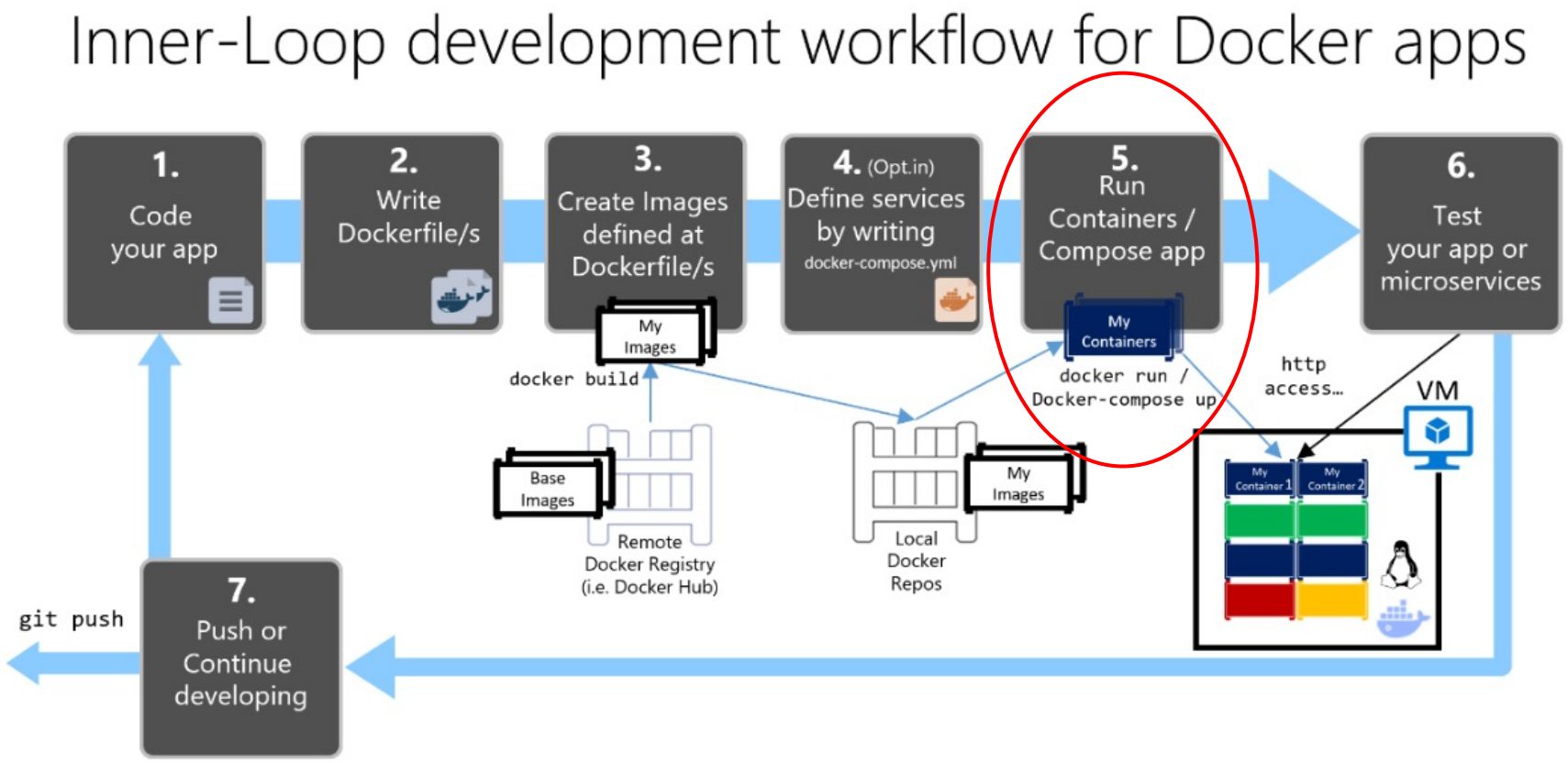
For an application with multiple containers, it is more complex and error-prone.

So, can we manage multiple containers as a single entity?

Docker Compose

- A tool for defining and running multi-container applications
- An application is composed of services
- Each service is defined with
  - container image
  - commands to execute
  - network
  - port
  - CPU and memory limits
  - number of replicas
- Docker Compose uses a YAML (Yet Another Markup Language) file to configure the application's services.

Workflow for developing Docker apps



https://learn.microsoft.com/en-us/dotnet/architecture/microservices/docker-application-development-process/docker-app-development-workflow

Installing Docker Compose

- Check if Docker Compose is installed

```
docker compose version
```

- Windows and Mac
  - Docker Desktop already includes Docker Compose
- Linux

```
sudo apt-get install docker-compose-plugin
```

https://docs.docker.com/compose/install/

Docker Compose basic commands

```
## List Docker-Compose CLI commands
docker compose
docker compose COMMAND --help

## Build images for services
docker compose build [SERVICE...]

## Run services
docker compose up -d [SERVICE...]
docker compose up --build -d [SERVICE...]

## Stop services
docker compose down

## List services
docker compose ps

## View logs [SERVICE...]
docker compose logs
```

Docker Compose File

- The compose file is a YAML file defining:
  - Version
  - Services
  - Networks
  - Volumes
  - Configs
  - Secrets
- Default name is compose.yaml or compose.yml
  - Older version: docker-compose.yaml or docker-compose.yml

compose.yml

```
version: '3'

networks:
  monitoring:
    driver: bridge

volumes:
  grafana-vol:
    external: true

services:
  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    expose:
      - 9100
    networks:
      - monitoring

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    expose:
      - 9090
    networks:
      - monitoring

  grafana:
    image: grafana/grafana
    container_name: grafana
    volumes:
      - grafana-vol:/var/lib/grafana
    ports:
      - 3000:3000
    networks:
      - monitoring
```

Top-level elements

Create a bridge network named “monitoring”

Declare volumes to be used.
extenal:true means the volumes already created
Without this flag, a new volume will be created on the first invocation and reused in subsequence invocations.

Declare services and their configurations and requirements.

https://docs.docker.com/compose/compose-file/03-compose-file/

compose.yml

```
version: '3'

networks:
  monitoring:
    driver: bridge

volumes:
  grafana-vol:
    external: true

services:
  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    expose:
      - 9100
    networks:
      - monitoring

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    expose:
      - 9090
    networks:
      - monitoring

  grafana:
    image: grafana/grafana
    container_name: grafana
    volumes:
      - grafana-vol:/var/lib/grafana
    ports:
      - 3000:3000
    networks:
      - monitoring
```

Services

image

This port will be accessible by other services connected to the same network but won't be published on the host machine.

Bind mount

Volume mapping

Publish port

https://docs.docker.com/compose/compose-file/03-compose-file/

Start all containers with Docker Compose

Create and start containers

```
docker compose up -d
```

List containers

```
docker compose ps
```

Stop containers

```
docker compose stop
```

```
veera@Yoga:~/docker/prometheus$ docker compose up -d
[+] Running 3/3
 ✓ Container grafana        Started                                                    0.0s
 ✓ Container node-exporter  Started                                                    0.0s
 ✓ Container prometheus     Started                                                    0.0s
veera@Yoga:~/docker/prometheus$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                    NAMES
902bf8adad62   prom/prometheus      "/bin/prometheus --c…"   22 minutes ago   Up 4 seconds    9090/tcp                 prometheus
905ad63b2d75   prom/node-exporter   "/bin/node_exporter"     22 minutes ago   Up 4 seconds    9100/tcp                 node-exporter
ca43733d5ffa   grafana/grafana      "/run.sh"                22 minutes ago   Up 4 seconds    0.0.0.0:3000->3000/tcp   grafana
veera@Yoga:~/docker/prometheus$ docker compose stop
[+] Stopping 3/3
 ✓ Container prometheus     Stopped                                                    0.9s
 ✓ Container grafana        Stopped                                                    0.6s
 ✓ Container node-exporter  Stopped                                                    0.8s
```

Docker Compose Network

Docker Compose creates a bridge network for the containers.

```
veera@Yoga:~/docker/prometheus$ docker network ls
NETWORK ID     NAME                   DRIVER    SCOPE
8441dd401405   bridge                 bridge    local
45d7c6c2c3f9   host                   host      local
39559390a0d1   mynet                  bridge    local
1750f3cb1cc4   none                   null      local
bc742ce85b51   prometheus_monitoring   bridge    local
```