# Docker
# [SWE2021]

**JinYeong Bak**
jy.bak@skku.edu
Human Language Intelligence Lab, SKKU

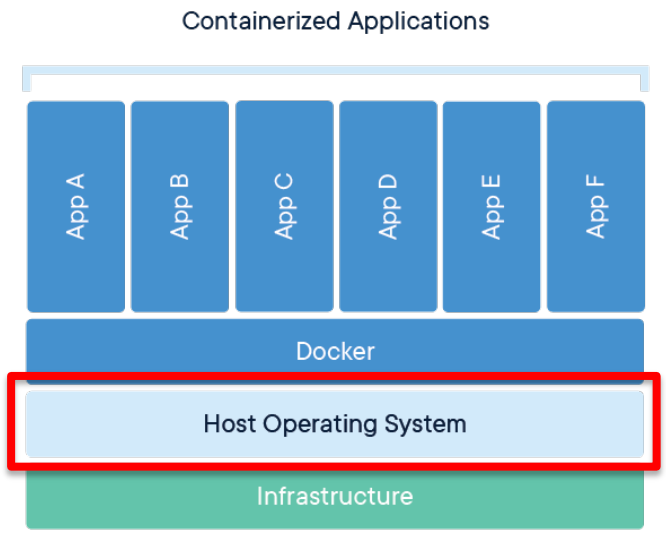
Slide credit: YeongJun Hwang / hmtyj2@g.skku.edu

# Docker

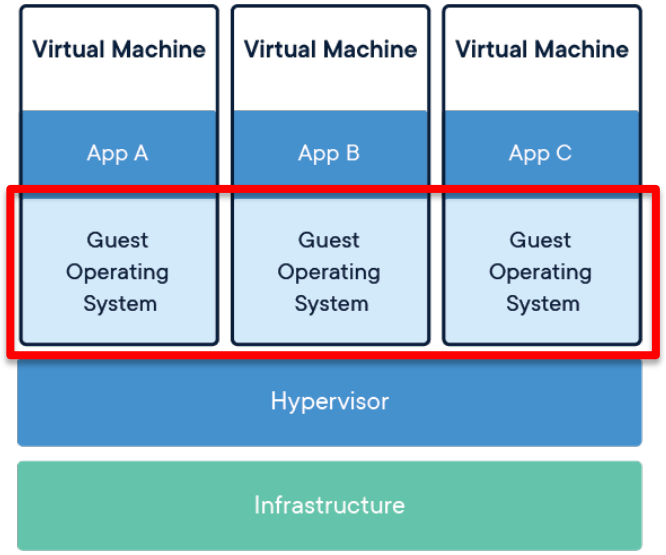
- Platform as a service (PaaS) product which use OS-level virtualization
- Open platform for developing, shipping, and running applications
- Containerization platform that is used to package your application in form of containers
- Delivering software quickly by separating applications from infrastructure

# OS-level Virtualization

- OS paradigm in which the kernel allows multiple isolated user instances (i.e., Partitioning OS to create multiple isolated virtual machines)
- Provides flexible and efficient way to run multiple applications improving resource utilization, application isolation



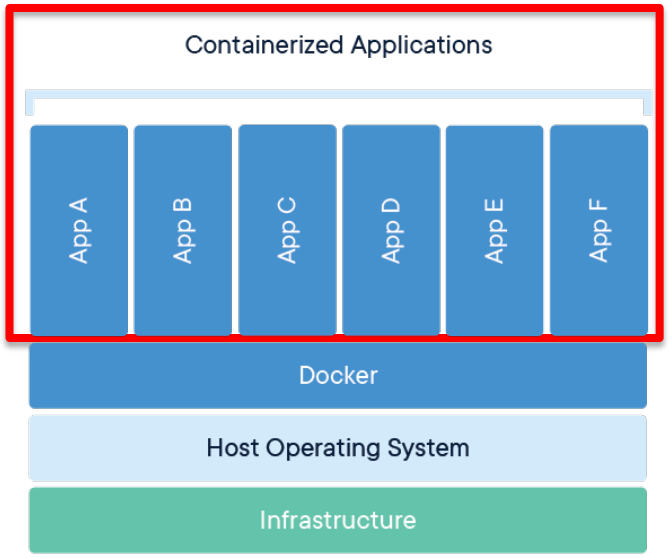

OS-level virtualization

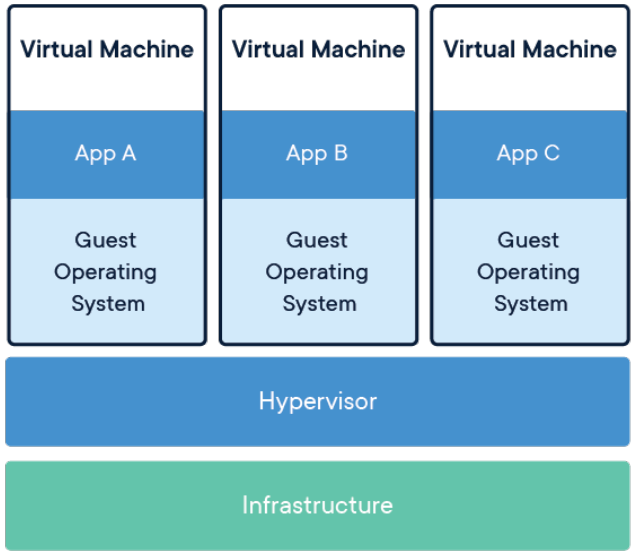



Hardware virtualization

# Containerization

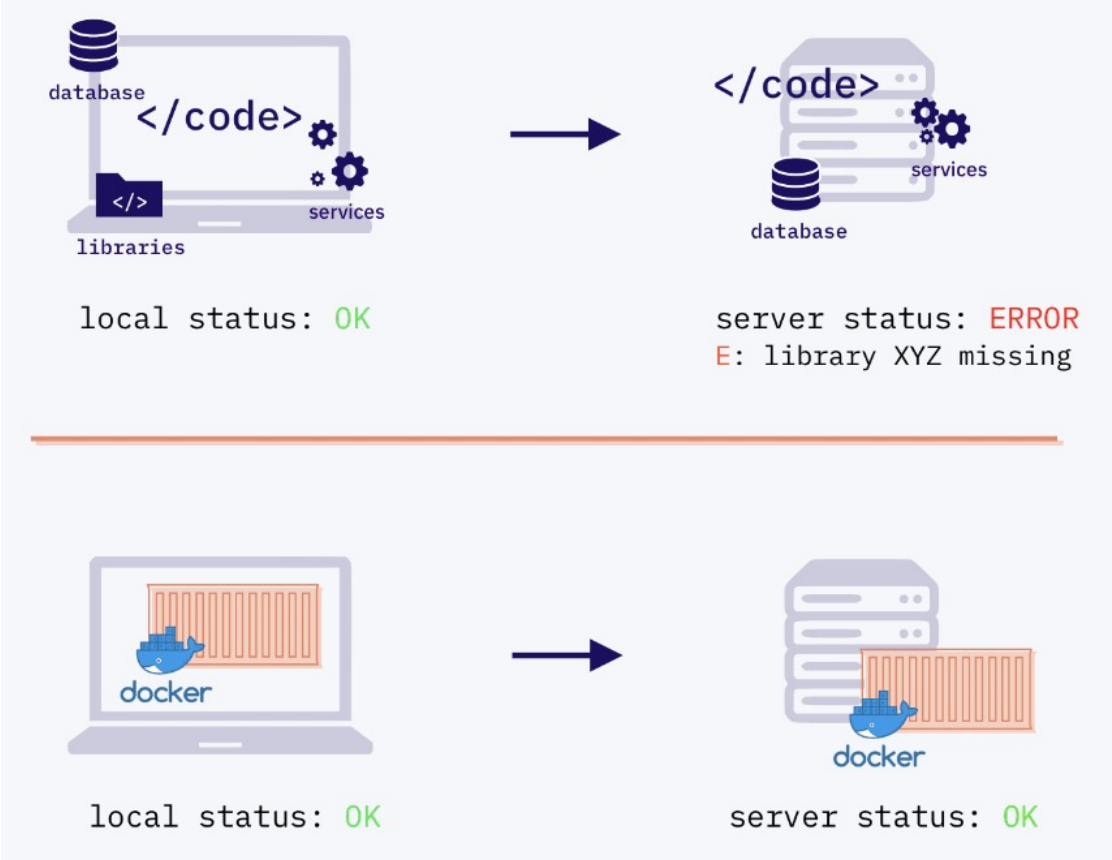
- OS-level virtualization that creates multiple virtual units called containers in the user space
- Standard unit container consists software that packages up code and its dependencies

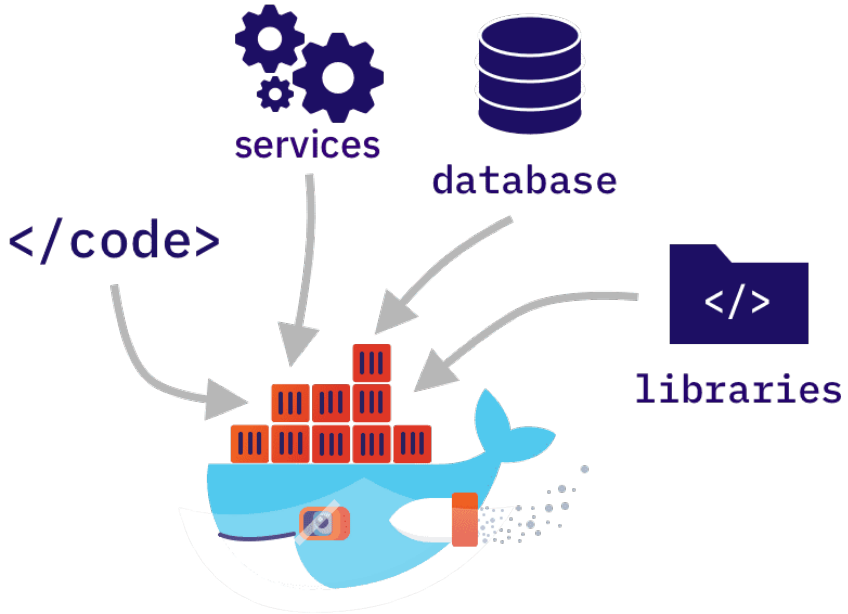


# Why Docker?

- Consistent delivery of applications



# Why Docker? (cont'd)

- Responsive deployment and scaling
  - Highly portable workloads
  - Can run container in local laptop, data center, or cloud providers (e.g., Amazon Web Service, Azure)

- Running more workloads on the same hardware
  - Lightweight and fast
  - Can run multiple workloads with different setting on single laptop

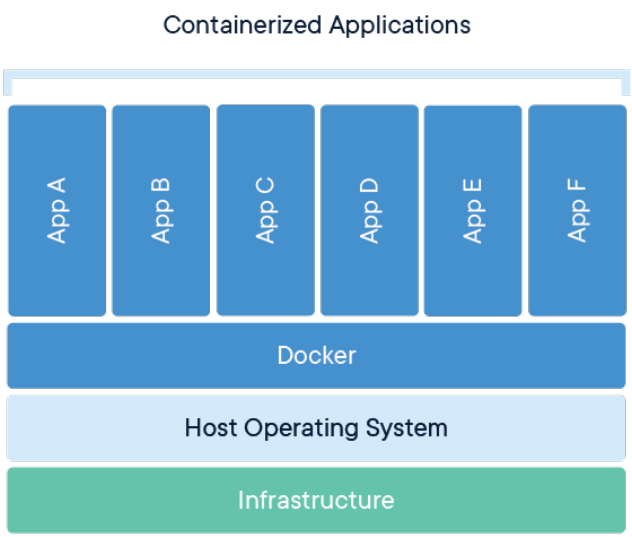
# Docker vs Virtual Machines

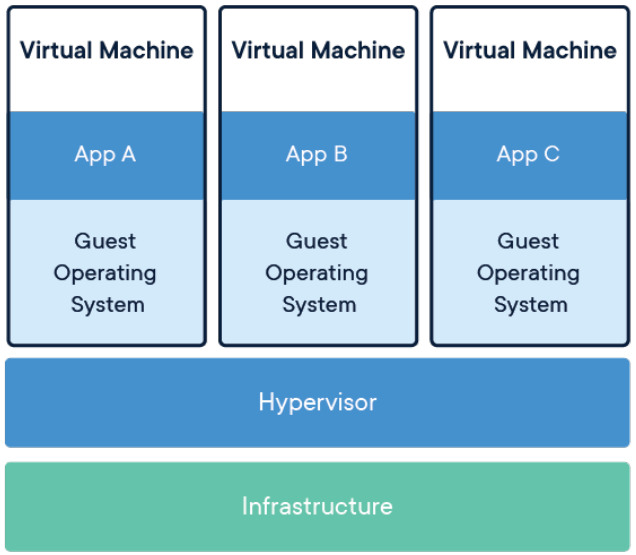
## Docker

- Abstract at app layer
- Multiple containers share OS kernel with other containers

## Virtual Machines

- Abstract of physical hardware
- Each VMs include full copy of OS



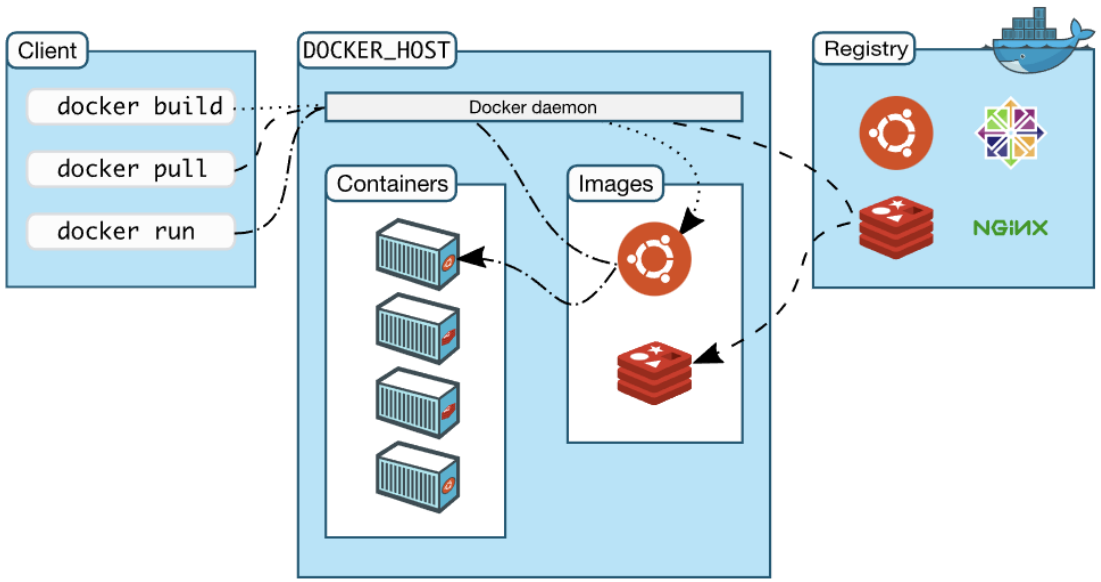



# Docker vs Virtual Machines (cont'd)

| Docker | Virtual Machines |
| --- | --- |
| • Less memory space | • A lot of memory space (OS load) |
| • Short boot-up time (few seconds) | • Long boot-up time (minutes) |
| • Data volume cannot be shared | • Data volume can be shared |
| • Easy to scale up | • Difficult to scale up |

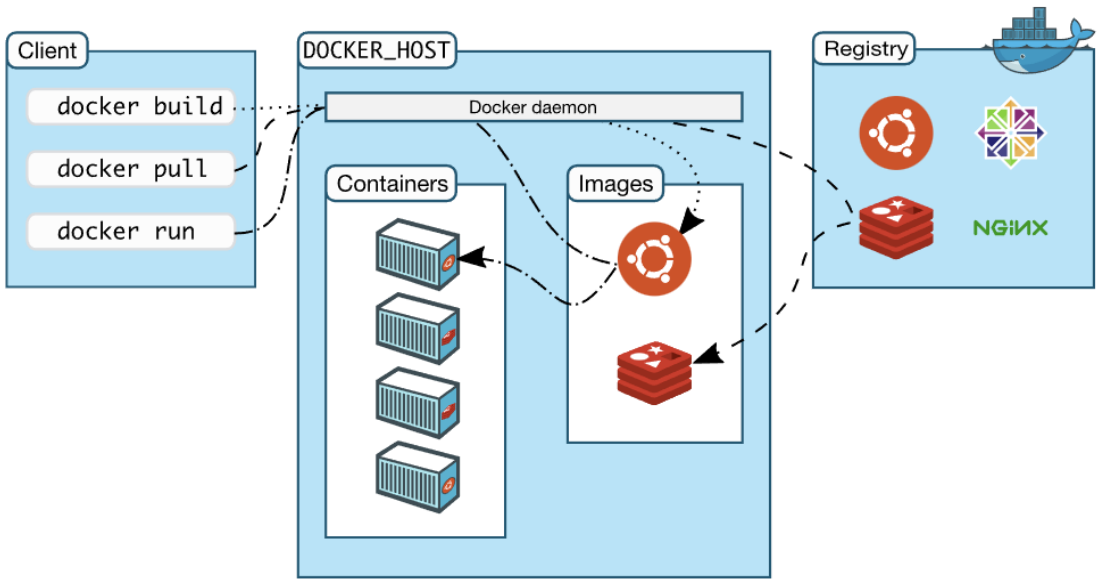
# Docker Architecture

- Client–server architecture with REST API
- Docker client
  - User interface for communicating docker system
- Docker host
  - The machine managing the containers and images
- Registry
  - Highly scalable server-side application that stores and lets you distribute Docker images (https://hub.docker.com/)



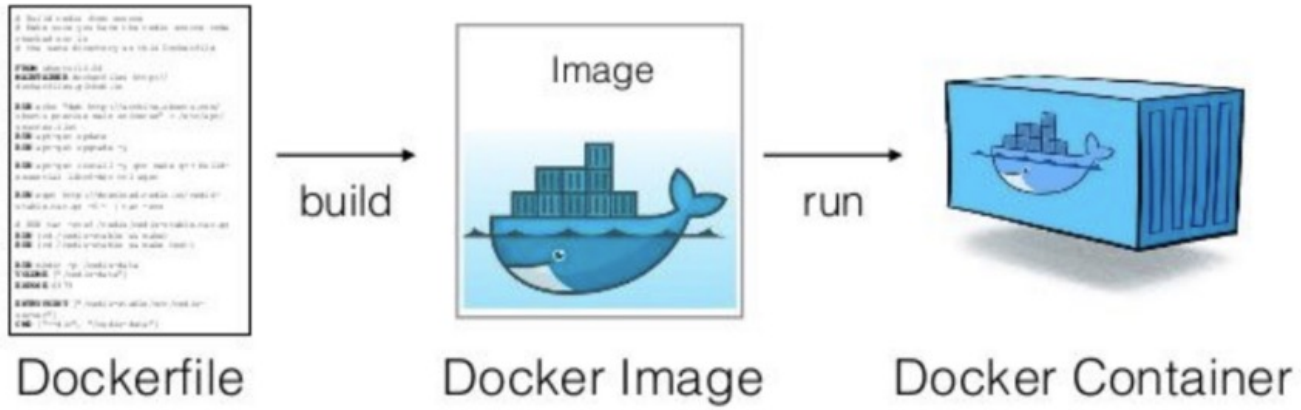
# Docker Architecture (cont'd)

- Docker daemon
  - Listens docker API requests and manages docker objects (images, containers, network, …)
- Images
  - Read-only template for creating docker containers
- Containers
  - Living, runnable instance of a docker image



# Docker Image and Container

- Docker images
  - Read-only template with instructions for creating a docker container
  - Specific point of an application and its virtual environment  (snapshot)
- Docker containers
  - Living & runnable instance of a docker image
  - Defined by its images with configuration options when created and started



# Docker Installation

- https://docs.docker.com/get-docker/

## Get Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.

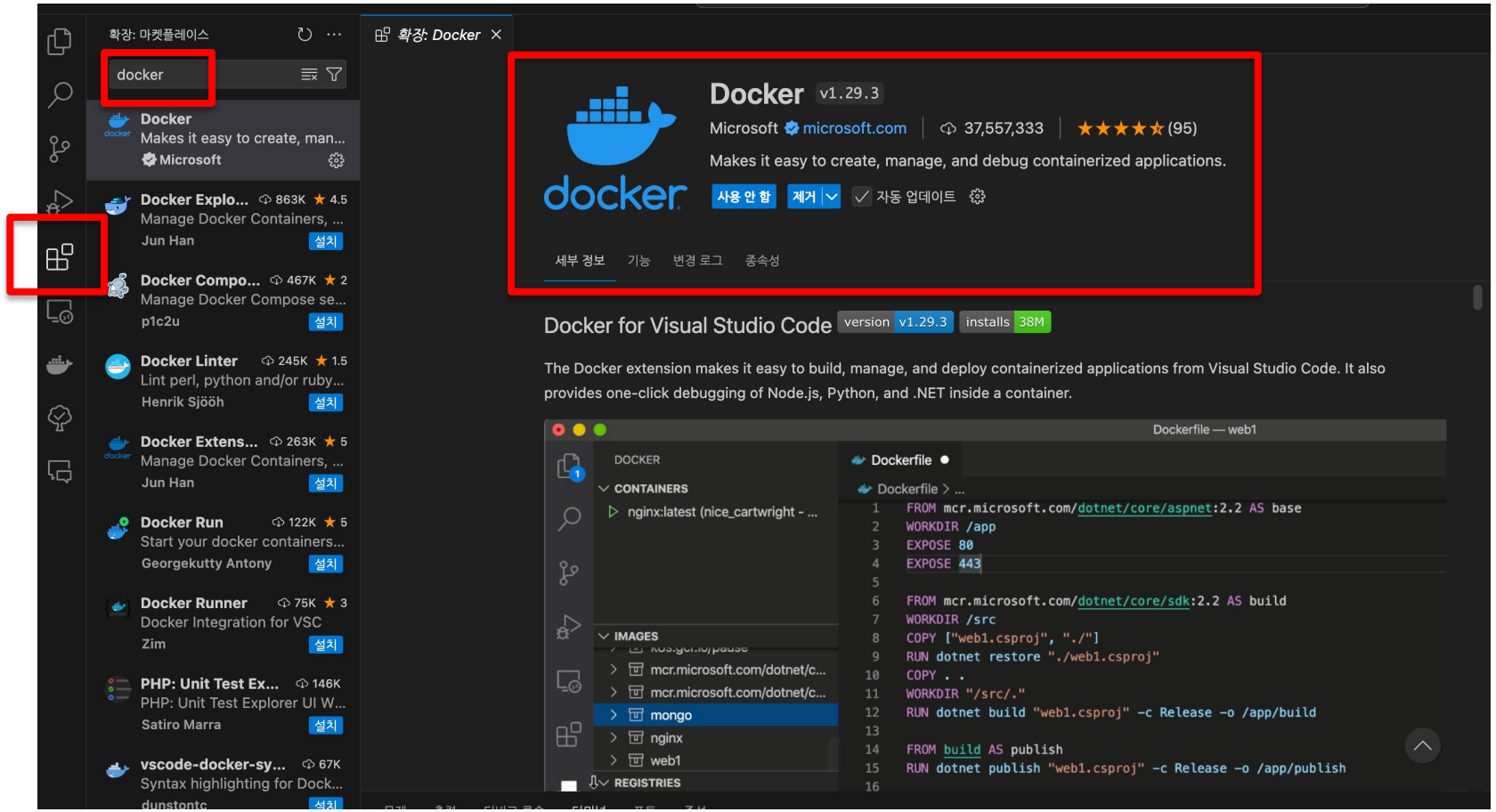
**Docker Desktop for Mac**
A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

**Docker Desktop for Windows**
A native Windows application which delivers all Docker tools to your Windows computer.

**Docker Desktop for Linux**
A native Linux application which delivers all Docker tools to your Linux computer.

# Docker Installation – VSCode extension

- Extension to manage docker environments in VSCode

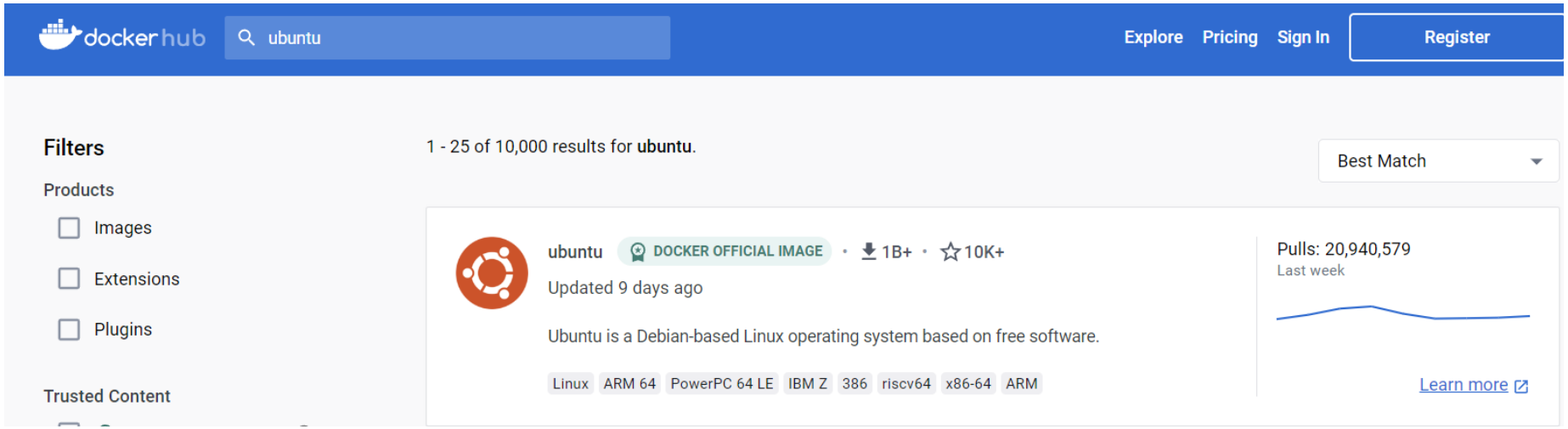


# Docker Pull

- ‘docker images’ will print all the docker images we have
  - We don’t have any docker images for now

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
```

- Go to docker hub(https://hub.docker.com/) and find ‘ubuntu’ images



# Docker Pull (cont'd)

- 'docker pull <image_name>:<tag_name>' will pull docker images from docker hub to local (e.g., docker pull ubuntu:22.10)
  - If we don't provide tag, the default tag (latest in this case) will be used

**ubuntu** DOCKER OFFICIAL IMAGE · 1B+ · 10K+

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

```
docker pull ubuntu
```

Overview Tags

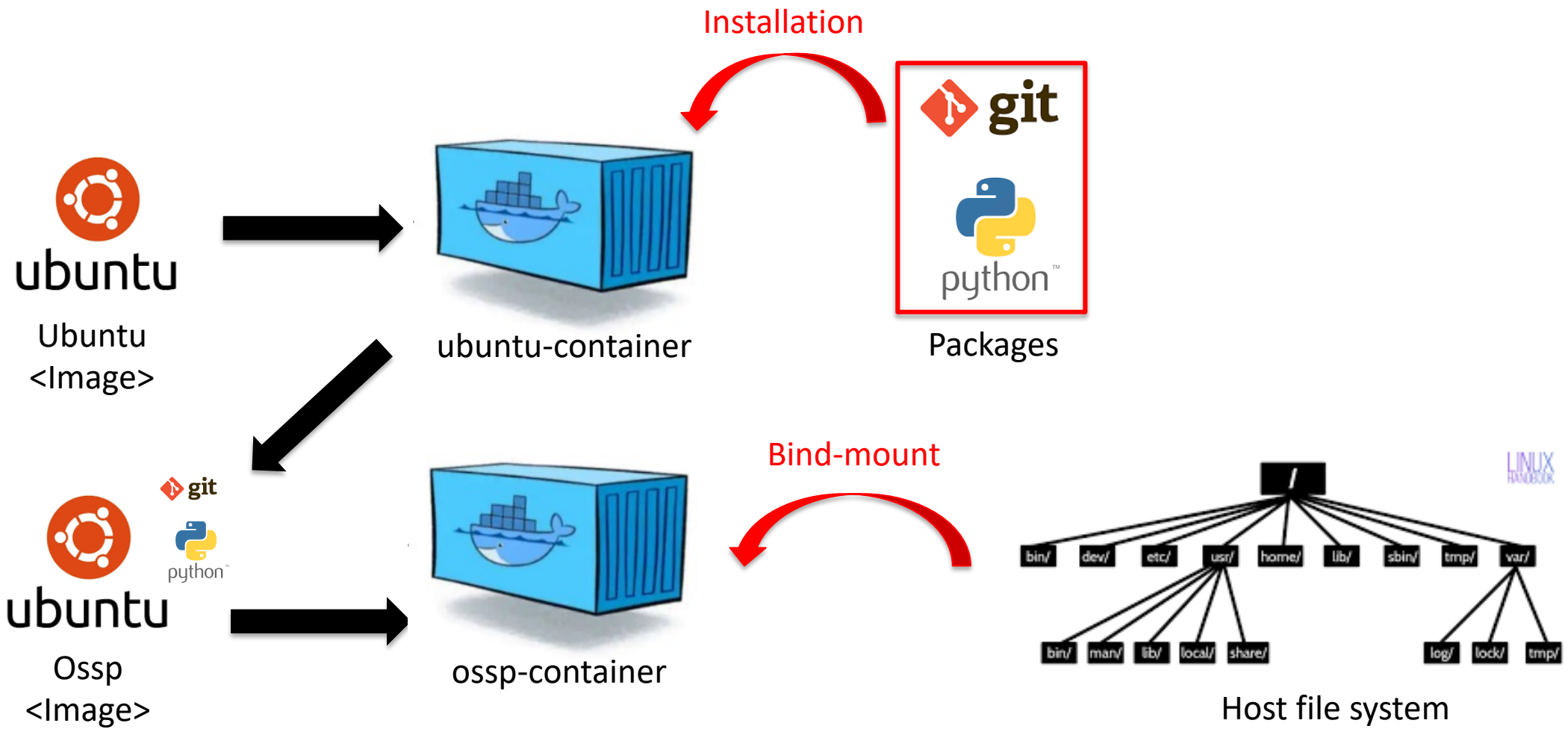
## Supported tags and respective `Dockerfile` links

- 18.04, bionic-20230308, bionic
- 20.04, focal-20230412, focal
- 22.04, jammy-20230308, jammy, latest
- 22.10, kinetic-20230412, kinetic
- 23.04, lunar-20230415, lunar, rolling

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
6e59cb05818e: Pull complete
Digest: sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe44391539243703c72766894fa834a
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

# Docker Workflow

- Today's task setup (Goal)



# Docker Run

- Now we have ‘ubuntu’ docker images

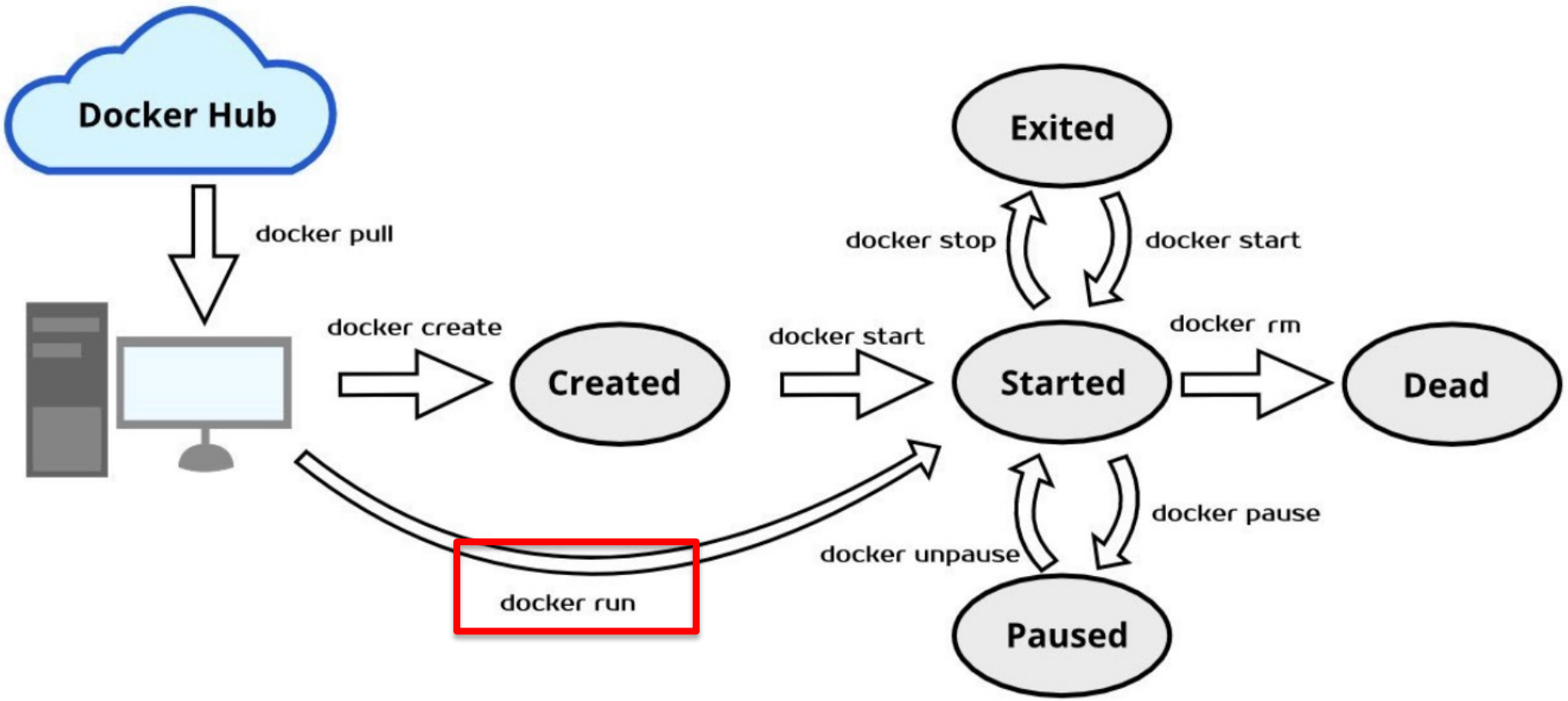
```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
ossp_env          latest    2eedf28684ec   28 minutes ago   665MB
ubuntu            latest    2b1b17d5e5a2   4 weeks ago      101MB
personal_image    latest    e448d519c29b   3 months ago     538MB
ubuntu            <none>    fabf3a8d4949   5 months ago     98.8MB
```

- Let’s check docker containers
  - ‘docker ps’ command will print the list of running containers

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS   PORTS   NAMES
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$
```

# Docker Container Lifecycle

- We don’t have ‘container’ yet



# Docker Run (cont'd)

- Run 'docker run <options> --name <container_name> <image_name>'
  - 'docker run -i -t -d --name ubuntu-container ubuntu'
  - This command will create docker container named 'ubuntu-container' from images 'ubuntu'

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS         PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   2 seconds ago    Up 1 second              ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$
```

# Docker Run (cont'd)

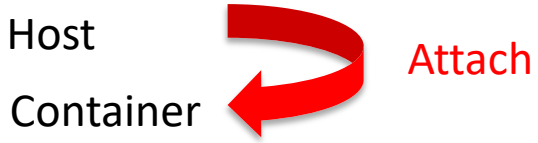
- What happens when you use command 'docker run -i -t -d --name ubuntu-container ubuntu'
  1) Docker create new container 'ubuntu-container' (from image 'ubuntu')
  2) Docker allocate a read-write filesystem to a container which enables the container to create of modify in its local filesystem
  3) Docker creates a network interface to connect the container to the default network
  4) Docker starts the container and executes '/bin/bash' (by default)
- More information on Docker Run (with options)
  – https://docs.docker.com/engine/reference/commandline/run/

# Docker Attach

- We run the container but, how we work in the container?

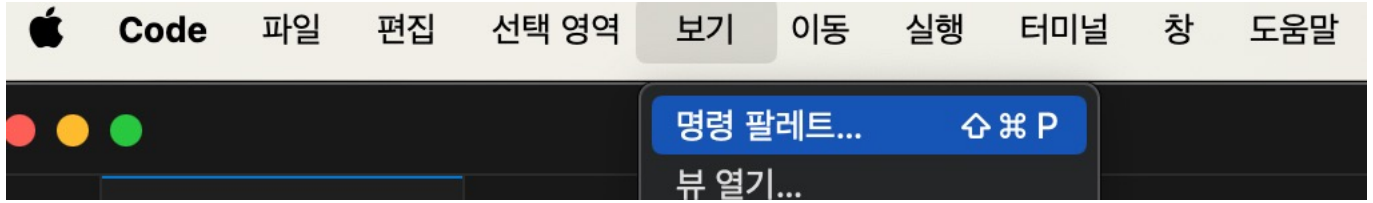
- ‘docker attach <container_name>’ will attach your terminal to the container
  - ‘docker attach ubuntu-container’

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker attach ubuntu-container
root@b6e854aa368d:/#
```

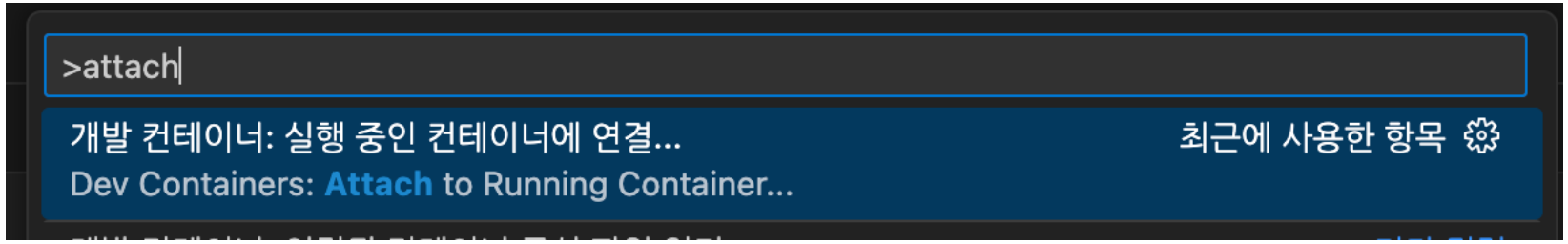


# Docker Attach – VSCode extention

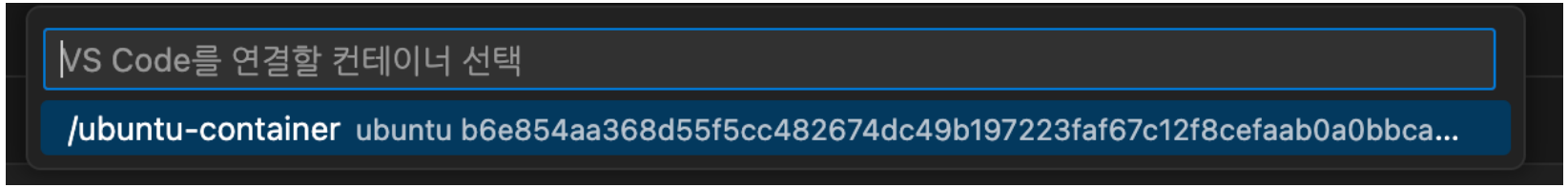
- View – Command palette




- Search ‘Dev Containers: Attach to Running Container’

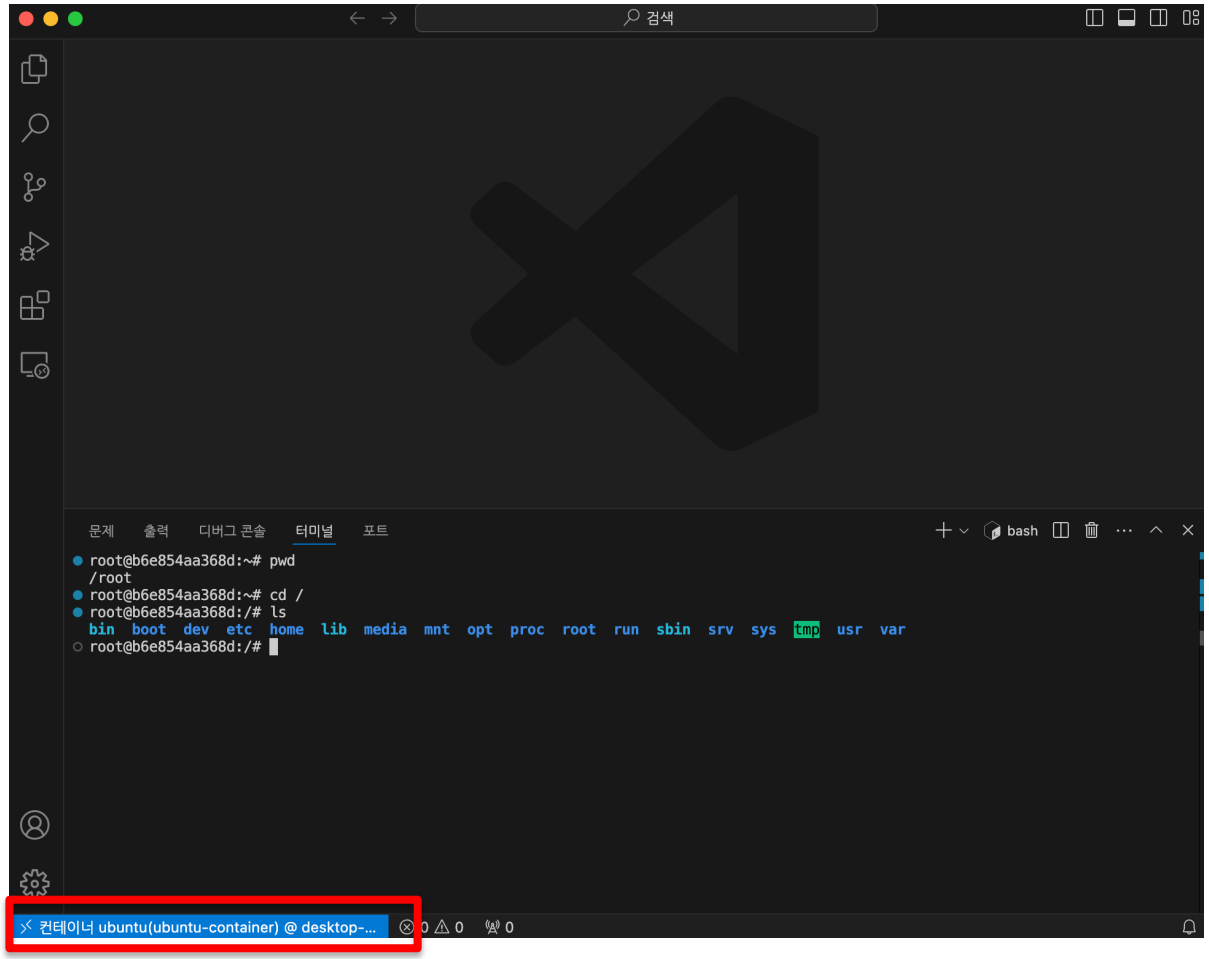



- Select container to attach (ubuntu-container)

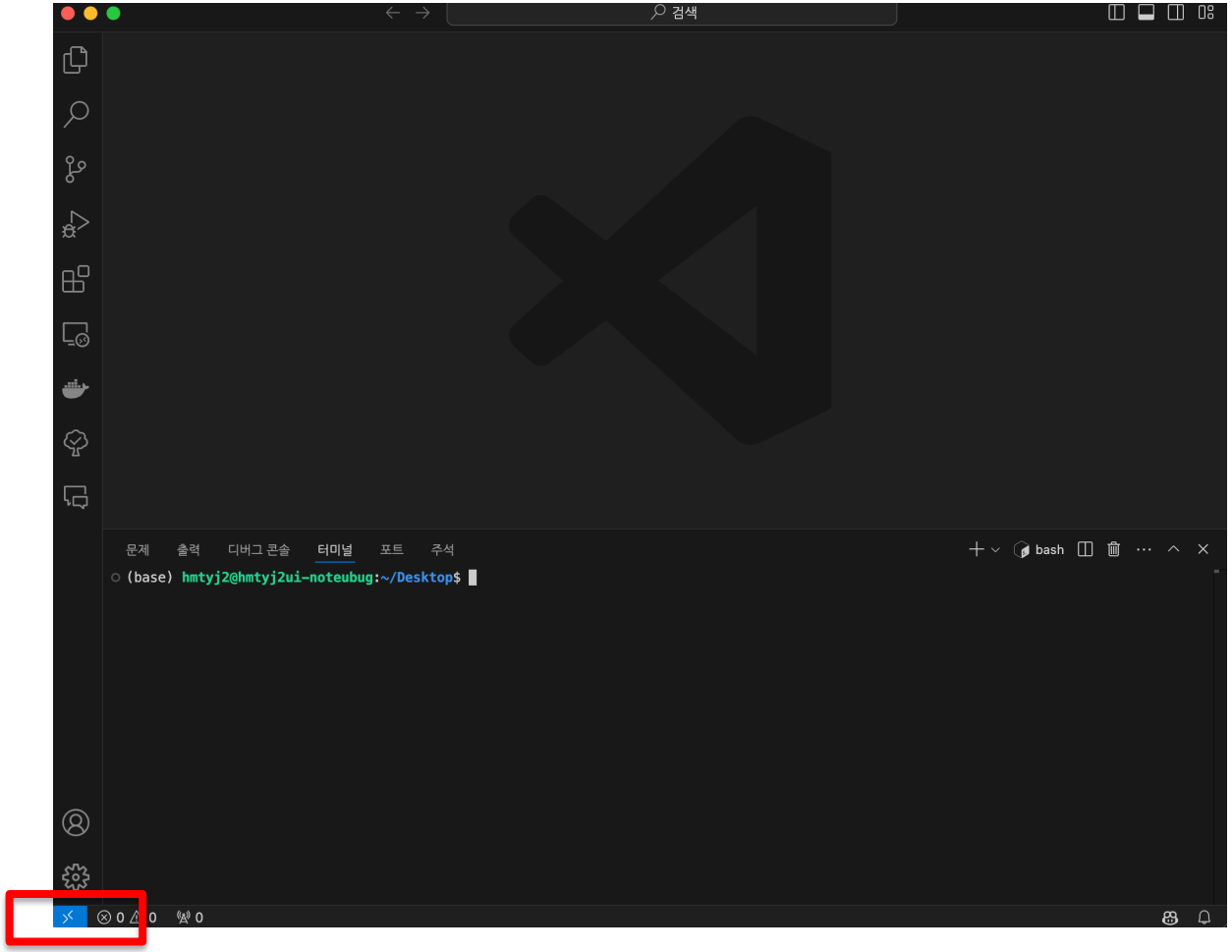


# Docker Attach – VSCode extention

- New window which is connected to container will be opened

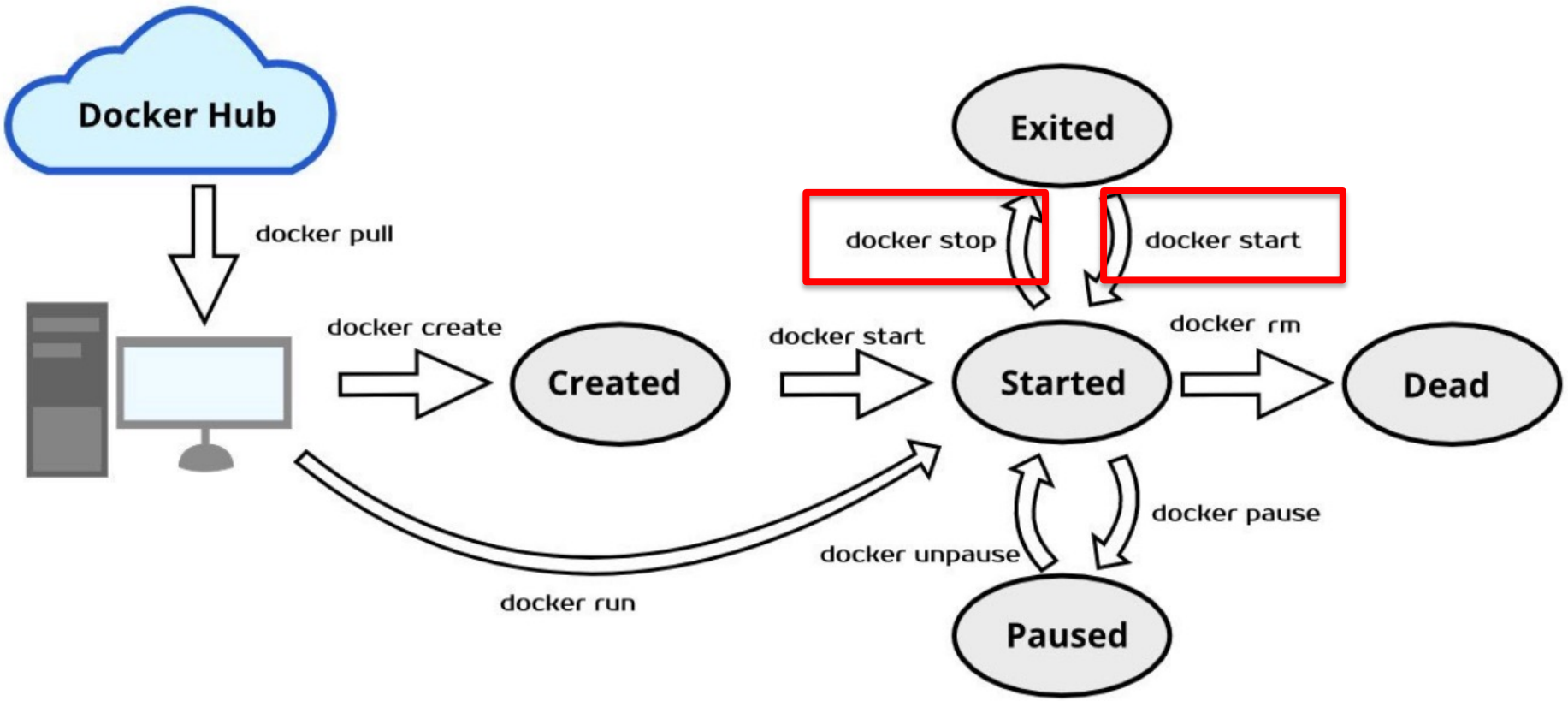



Container: ubuntu-container




Host machine (local machine)

# Docker Stop / Start



# Docker Stop / Start (cont'd)

- 'docker stop <container_name>'
  - Stop the running container
  - Status changed to 'Exited' from 'Up (running)'

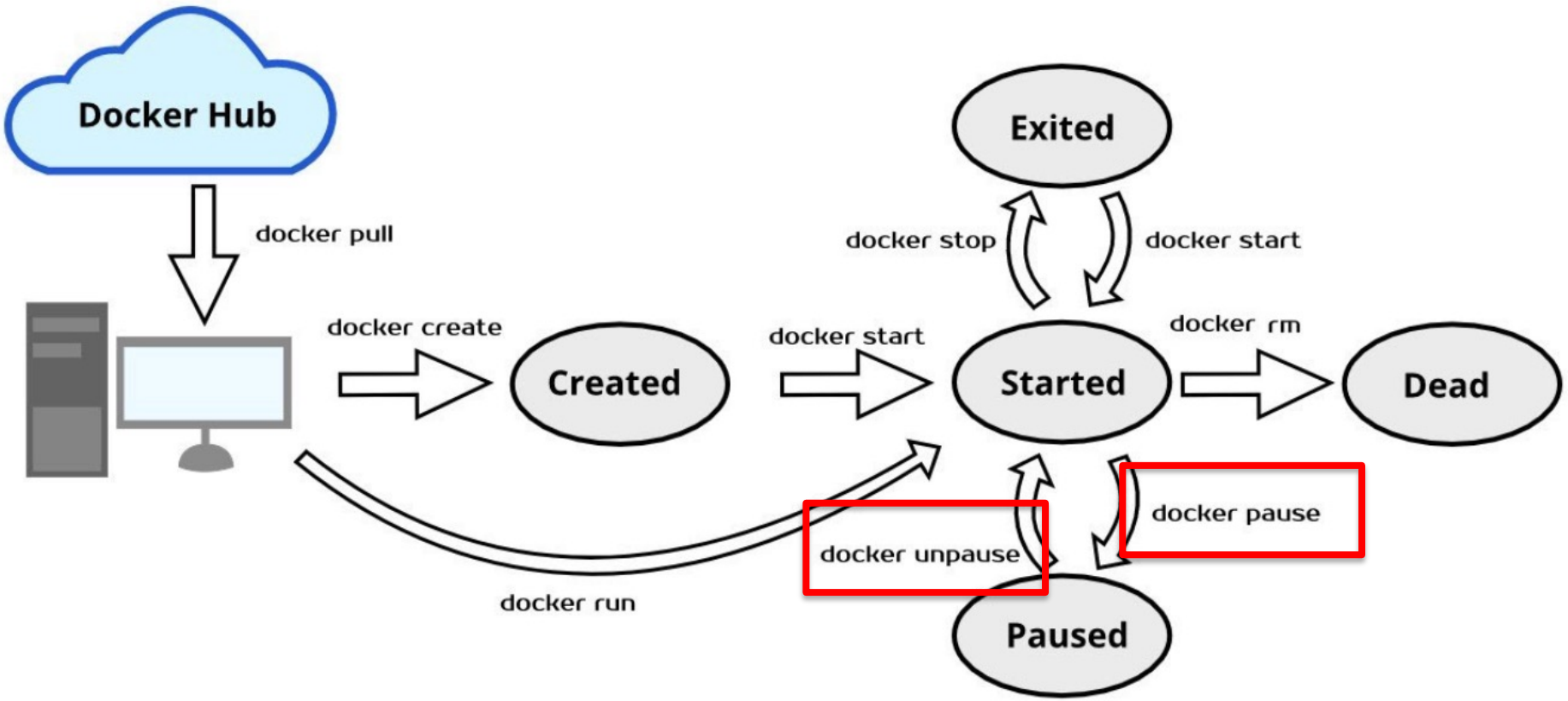
```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS         PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 4 seconds             ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker stop ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED       STATUS                      PORTS     NAMES
b6e854aa368d   ubuntu         "/bin/bash"   3 hours ago   Exited (137) 6 seconds ago            ubuntu-container
```

# Print all containers (-a option)

- 'docker start <container_name>
  - Start the exited / created container
  - Status changed to 'Up (running)' from 'Created' or 'Exited'

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker start ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS         PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 2 seconds             ubuntu-container
```

# Docker Pause / Unpause



# Docker Pause / Unpause (cont'd)

- 'docker pause <container_name>'
  - Pause the running container

- 'docker unpause <container_name>
  - Unpause the paused container

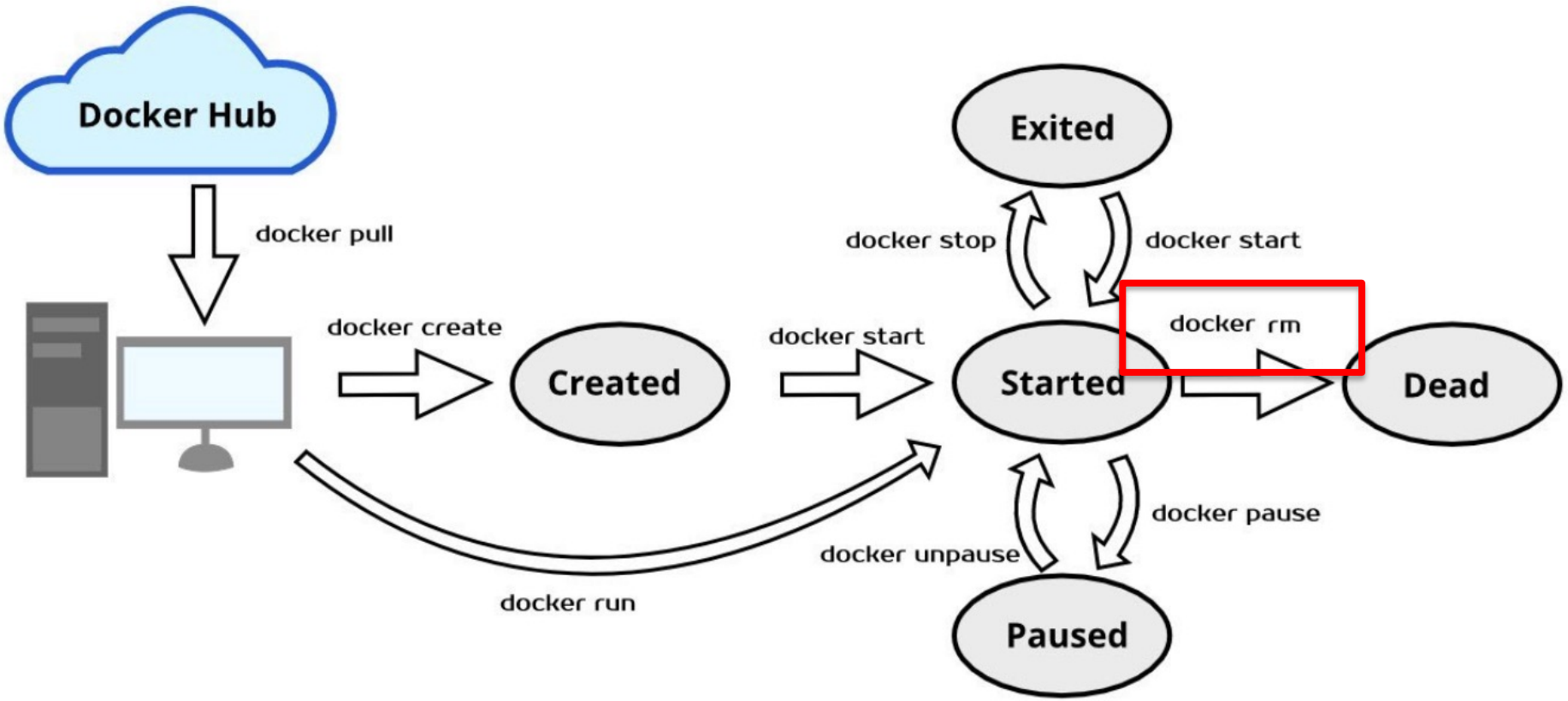
```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS         PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 2 seconds             ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker pause ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS                     PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 30 minutes (Paused)             ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker unpause ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS         PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 31 minutes            ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$
```

# Docker Stop & Pause

- Differences between stop and pause
  - ‘stop’ terminates all processes in container (SIGTERM, then SIGKILL), while ‘pause’ stops all processes (SIGSTOP)

- Why stop and pause
  - Save resources of host system
  - Graceful shutdown, clean up resources (stop)
  - Pausing process, quick resume (pause)

# Docker Remove



# Docker Remove (cont'd)

- 'docker rm <container_name>' will delete the 'stopped' or 'created' containers

- With force option (-f), we can delete containers in all status (e.x. running)

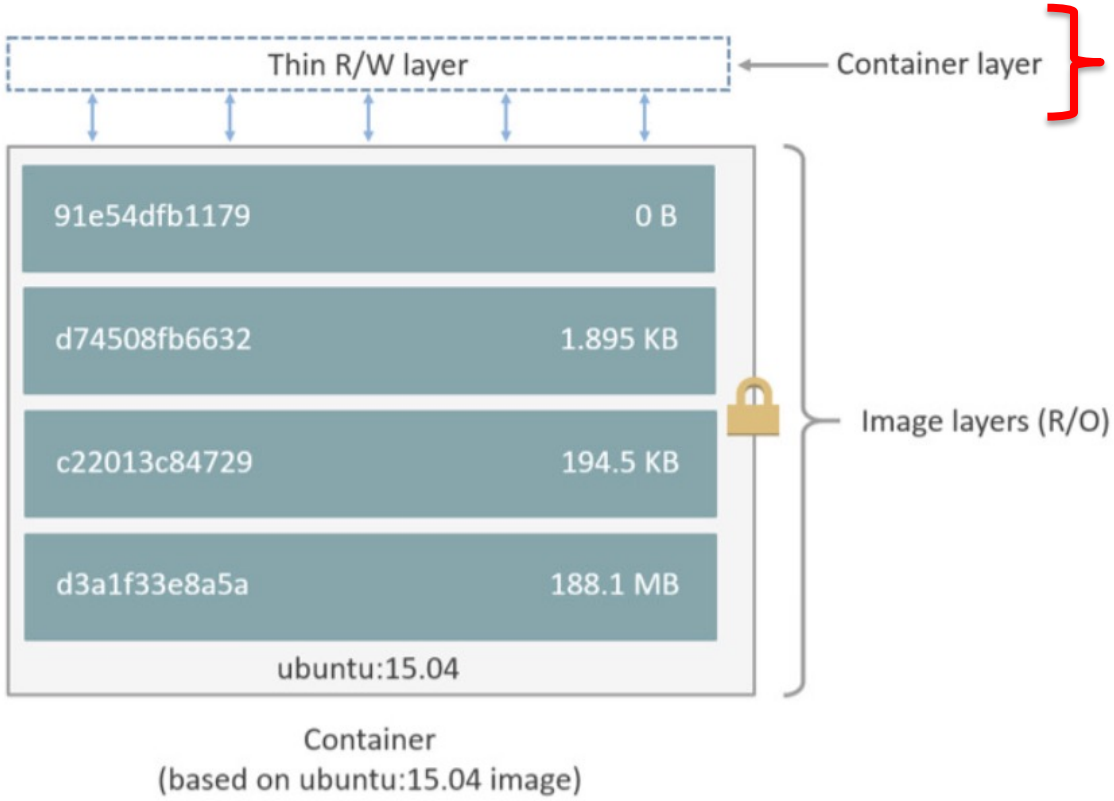
```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS          PORTS     NAMES
b6e854aa368d   ubuntu    "/bin/bash"   3 hours ago   Up 33 minutes             ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker stop ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker rm ubuntu-container
ubuntu-container
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED        STATUS                     PORTS     NAMES
```

# Stop the Container

# Container no more exists

# Docker Commit - Save

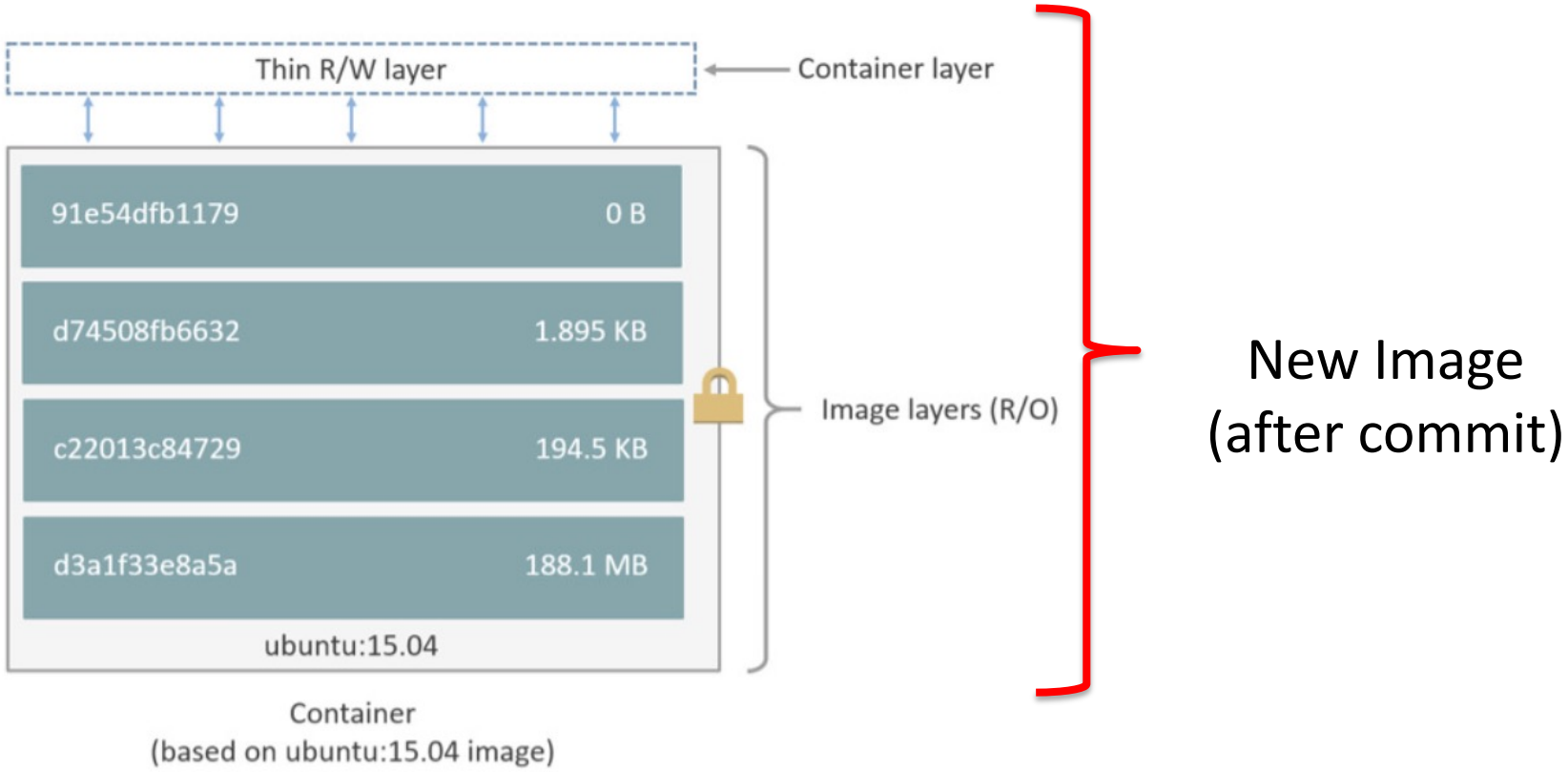
- If you delete a docker container, this will remove all the modified or created files which are written in container layer




Your code, installed libraries. …

# Docker Commit (cont'd)

- To save modifications in container (write code, install library, ..), we have to commit a container's change and setting into a new images



# Docker Commit (cont'd)

- Install 'Git' in container

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker attach ubuntu-container
root@56e69bfd9dc6:/# git --version
bash: git: command not found
root@56e69bfd9dc6:/# apt-get update && apt-get install git -y
```

# Attach to container (if not running, start the container)
# Check Git version (not installed, yet)
# Install git

- 'docker commit <container_name> <image_name>:<tag>' will commit the container to given image name with tag
  - 'docker commit ubuntu-container ossp'

```
root@56e69bfd9dc6:/# exit
exit
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker commit ubuntu-container ossp
sha256:72f1ee64584dea040e4be7cfdf4540fca69a0335ae6e29ab6ba44d3f0b8f34e7
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
ossp              latest    72f1ee64584d   4 seconds ago    233MB
```

# Exit container (container -> host)

# Check created image (ossp)

# Docker Commit (cont'd)

- Run container 'ossp-container' from image 'ossp'
  - 'docker run -dit --name ossp-container ossp'

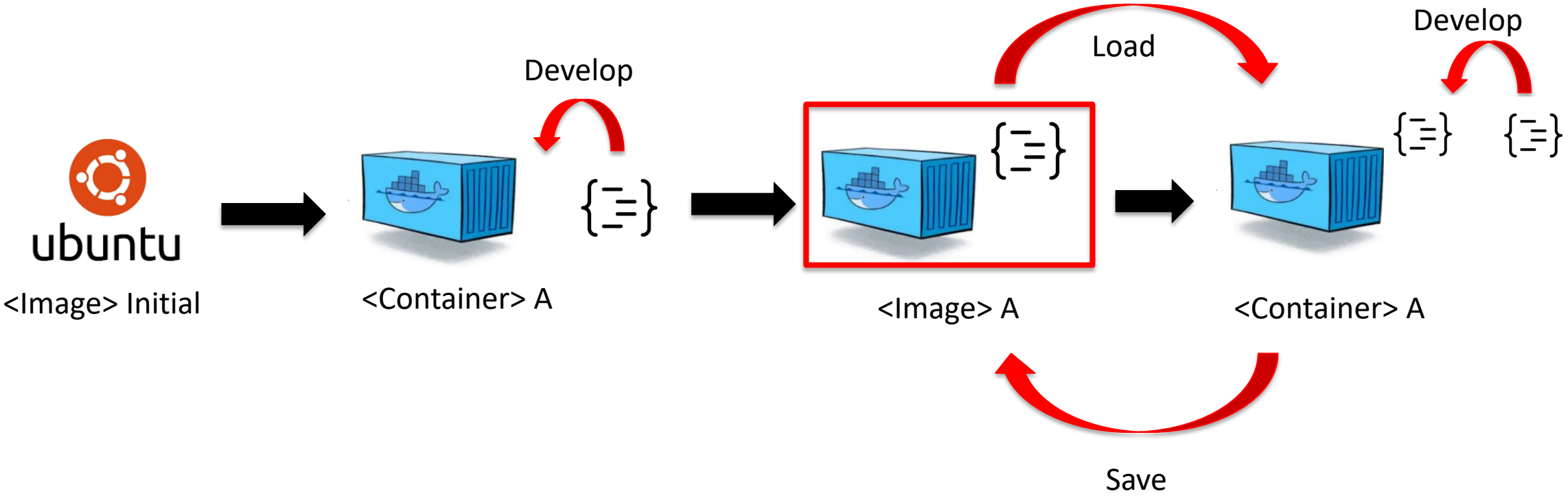
- Check 'git' in container 'ossp-container'

```
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker run -dit --name ossp-container ossp
3435c9265fdbe1be1489a602f80914d718466133da468fc611400ed9e5732e38
(base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker attach ossp-container
root@3435c9265fdb:/# git --version
git version 2.43.0
root@3435c9265fdb:/#
```

# Attach
# Check git version

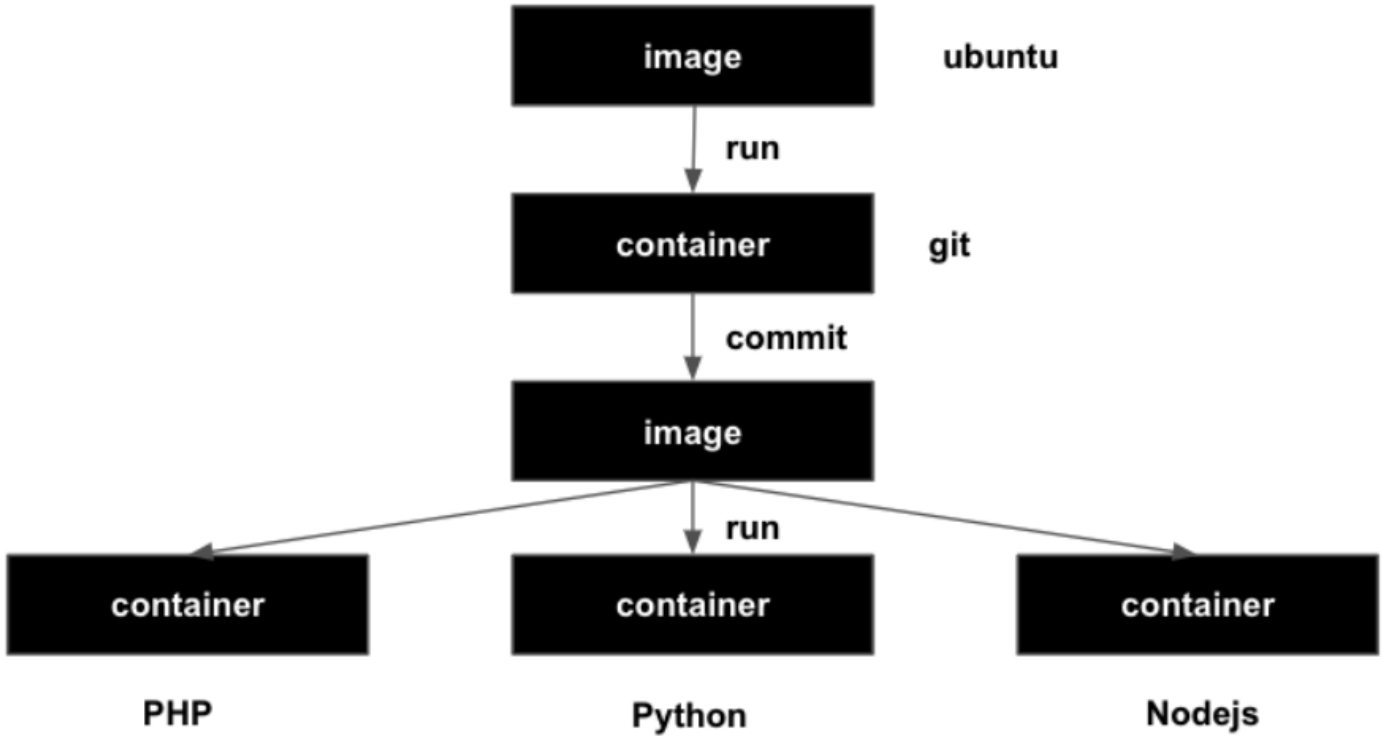
# Docker Commit (cont'd)

- In this way, you can save your work (container) as image and load when you need further work



# Docker Commit (cont'd)

- Committed container is stored as an image, so you can run other containers from the images and make multiple projects



# Docker Commands

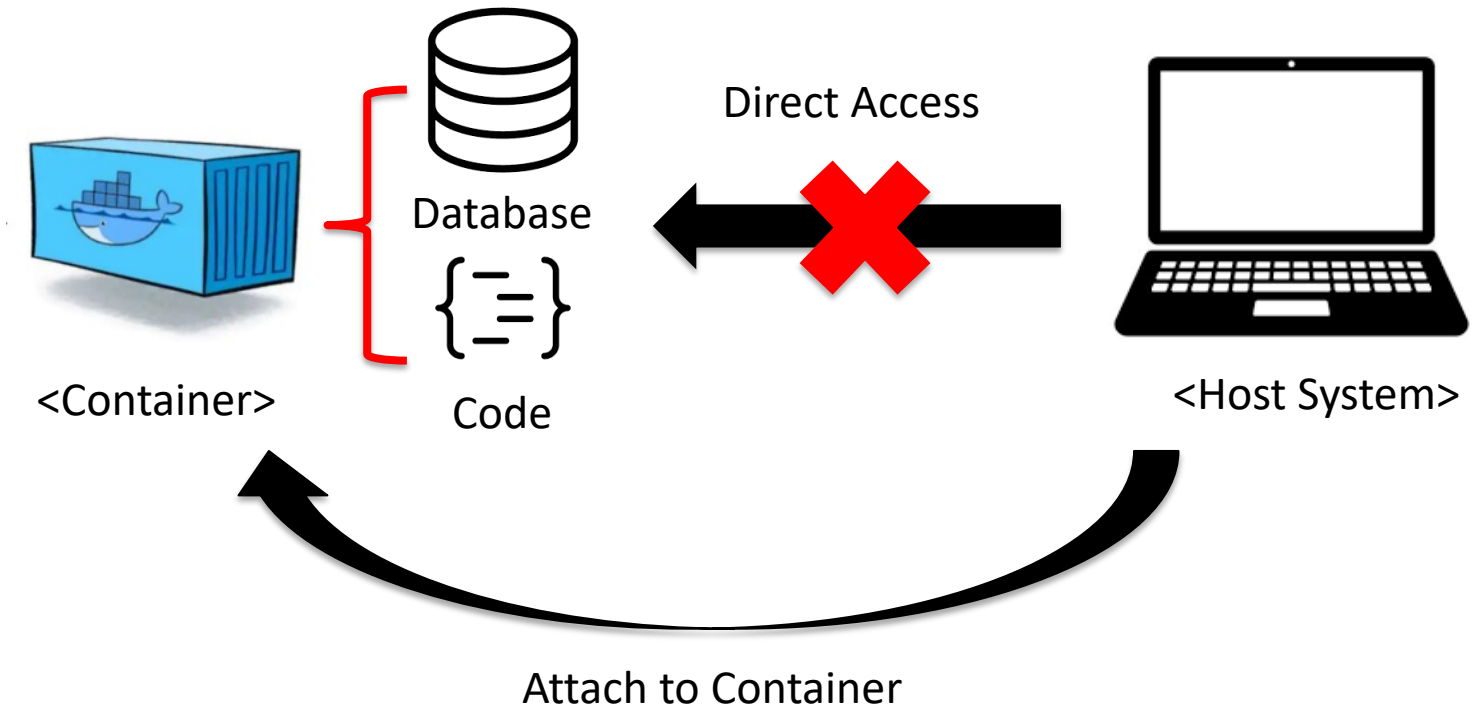
- Docker commands we study today (pull, run, stop, …) are basic commands to start docker

- You can see more commands with options (https://docs.docker.com/engine/reference/commandline/docker/)

- For example, with ‘docker compose’ you can run multiple containers for single application

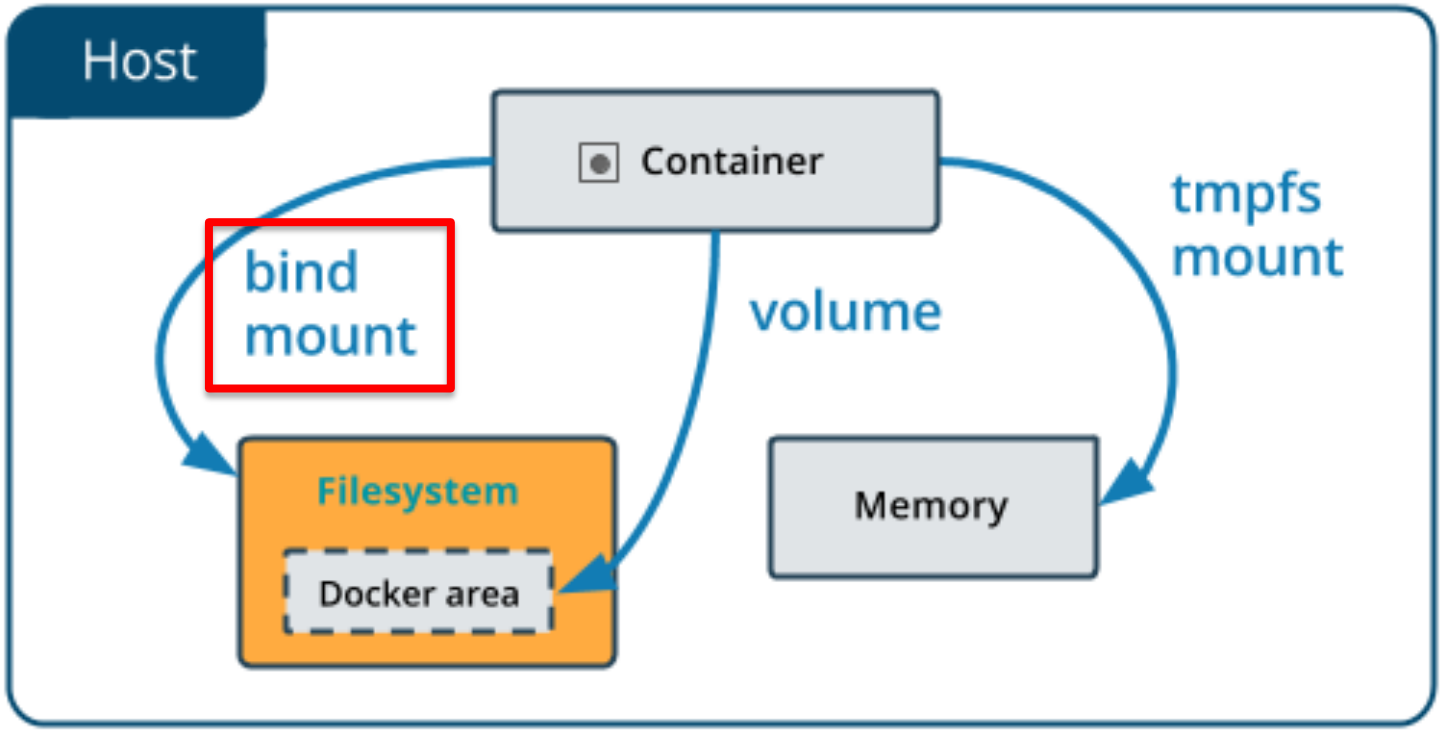
# Docker Storage - Bind mounts

- How to check your contents in container?
  - Host system cannot directly access to container's file system (storage)
  - Hard to transfer files between the container and the host system



# Docker Storage - Bind mounts

- When you use a bind mount, a file or directory on the host machine is mounted into a container
- The file or directory is referenced by its absolute path on the host machine



# Docker Storage - Bind mounts

- Command for bind mounts
  - ‘docker run -dit --name ossp-container **-v <host_dir_path>:<container_dir_path>** ossp’ will mount <host_dir> to <container_dir> and then share the files

```
● (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker rm -f ossp-container
  ossp-container
● (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ mkdir ossp_host_dir
● (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ touch ossp_host_dir/shared_file.txt
● (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ ls ossp_host_dir/
  shared_file.txt
● (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker run -dit --name ossp-container -v ./ossp_host_dir:/mnt/ossp_container_dir ossp
  9423b2e10b2256b353d175083d7a6db68fa33d0ecbcb28b38a7f0d50b5d9b0a5
○ (base) hmtyj2@hmtyj2ui-noteubug:~/Desktop$ docker attach ossp-container
  root@9423b2e10b22:/# ls /mnt/ossp_container_dir/
  shared_file.txt
  root@9423b2e10b22:/#
```

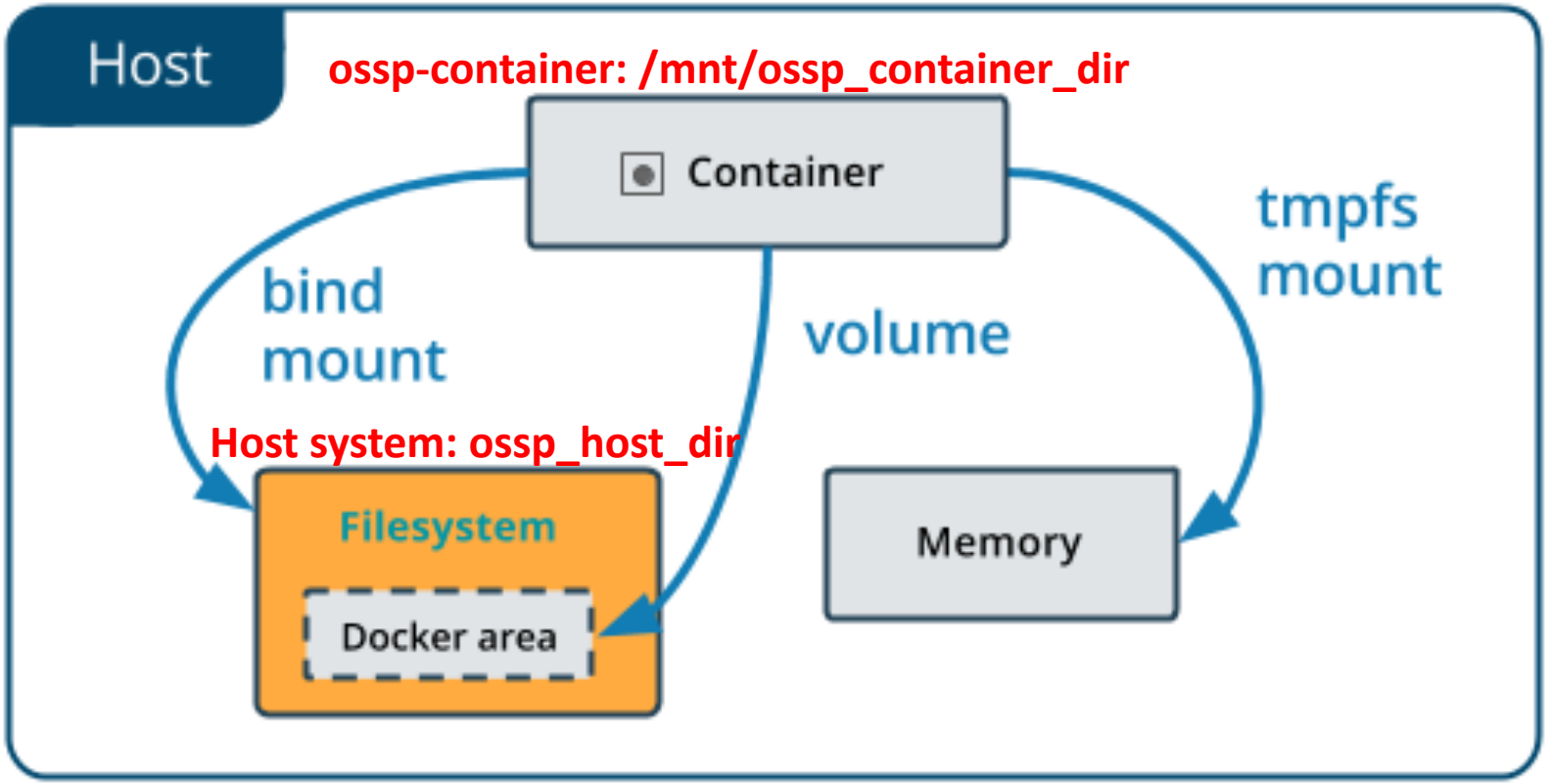
# Remove existing container (ossp-container)

# Set directory on host system

# Run container with mount option
ossp_host_dir <-> /mnt/ossp_container_dir

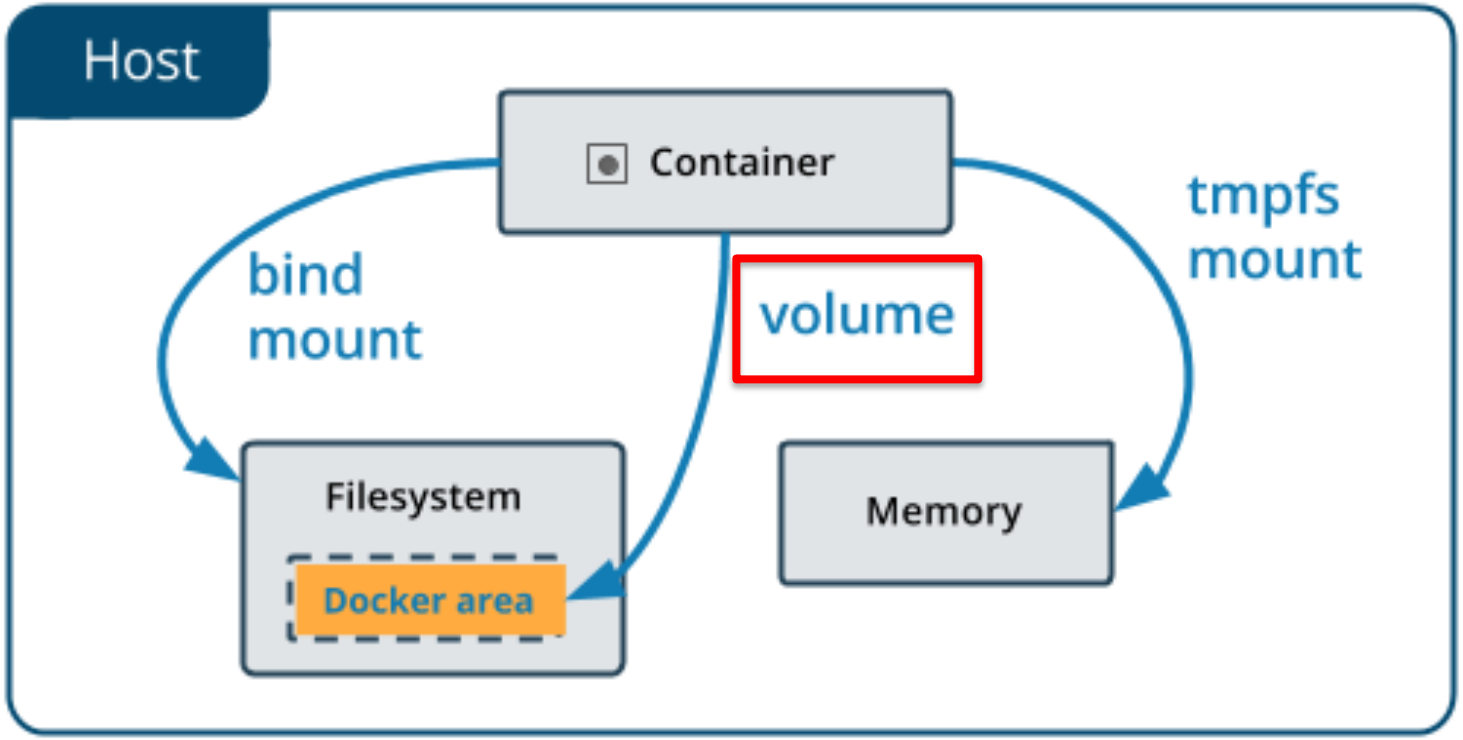
# Check container dir

# Docker Storage - Bind mounts



# Docker Storage - Volumes

- While bind mounts are dependent on the directory structure and OS of the host machine, volumes are managed by Docker
- Volumes are easier to back up or migrate than bind mounts

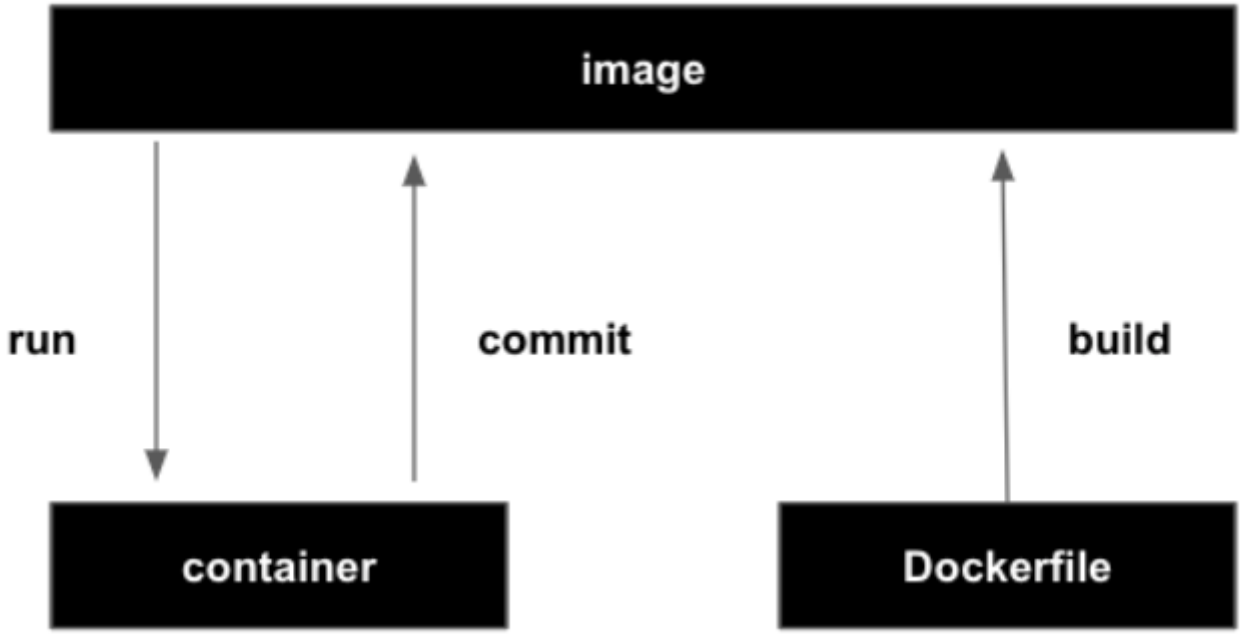



https://docs.docker.com/storage/volumes/

# Dockerfile

- Text document with instructions to build docker images

- Contains all the command lines to assemble images



# Dockerfile Format

- Dockerfile consists instruction and argument


- FROM
  - specify base image
- RUN
  - executes commands during the image build process
- WORKDIR
  - specific instructions (RUN, CMD, …) get executed in that directory

# Dockerfile Example

- Make a dockerfile which build a docker image same with ‘ubuntu_git’ from ‘ubuntu’ image

- Make directory and create ‘Dockerfile’ in that directory

Dockerfile ×

C: > Users > hmtyj2 > ubuntu_git_dir > Dockerfile

```
FROM ubuntu:latest

RUN apt-get update
RUN apt-get install -y git
```

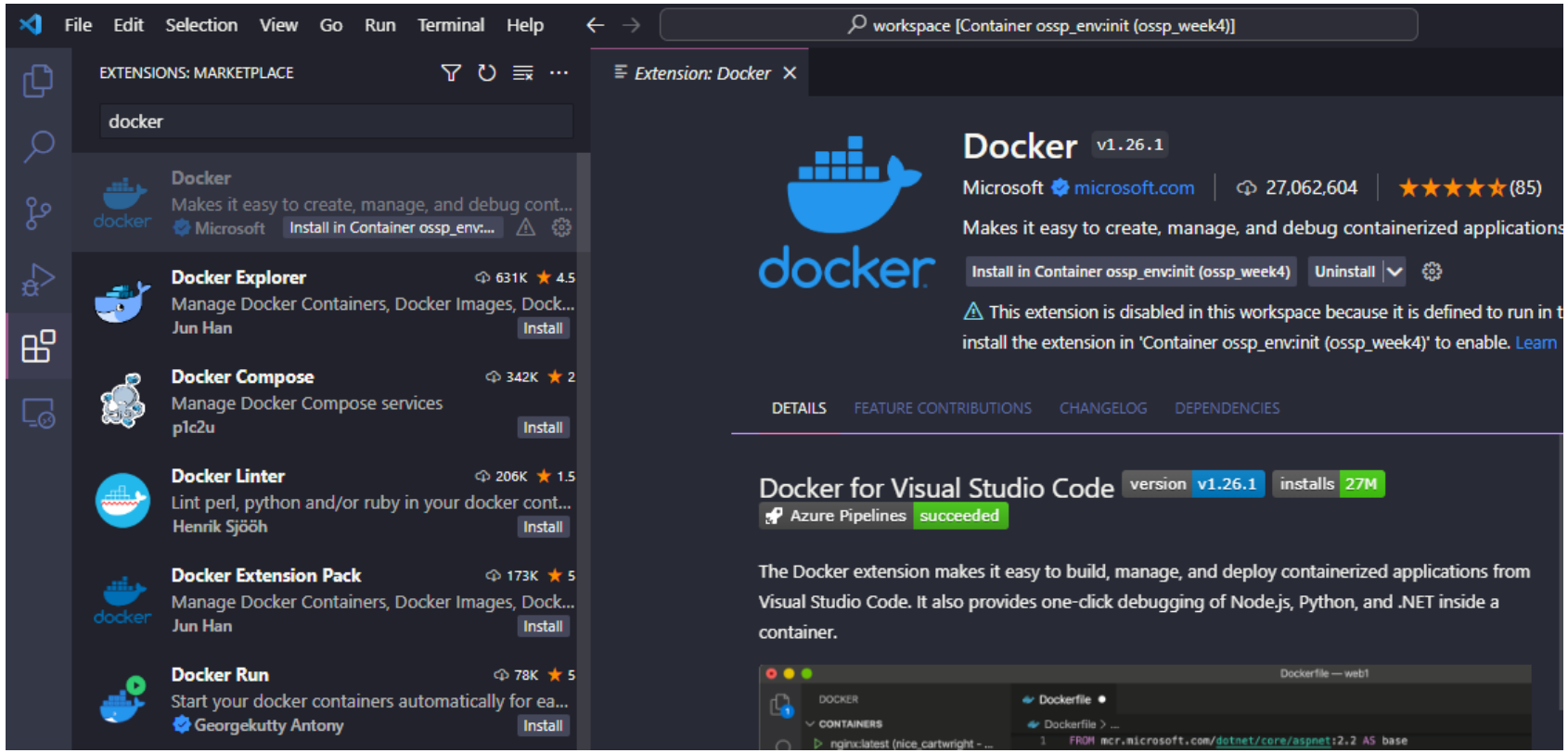
# Dockerfile Example (cont'd)

- Run 'docker build -t ubuntu_git:init . '
  - This will run 'Dockerfile' in current directory (' . ') and create new image named 'ubuntu_git' with 'init' tag

- Check and test the image

```
PS C:\Users\hmtyj2\ubuntu_git_dir> docker images
REPOSITORY   TAG       IMAGE ID       CREATED             SIZE
ubuntu_git   init      38b88d9b13b7   2 minutes ago       197MB
ubuntu       git       1660e4483171   About an hour ago   197MB
ubuntu       latest    08d22c0ceb15   7 weeks ago         77.8MB
PS C:\Users\hmtyj2\ubuntu_git_dir> docker run -dit --name my_ubuntu ubuntu_git:init
d914299c45dd1c897b9d2afa797c7038e9e1eee38616fe636a8edb19f28f59fd
PS C:\Users\hmtyj2\ubuntu_git_dir> docker attach my_ubuntu
root@d914299c45dd:/# git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
```

# Docker Extension - VSCode




- https://code.visualstudio.com/docs/containers/overview