Хичээл 2: Контейнер (docker)

Г.Гантулга

2024 оны 11-р сарын 28

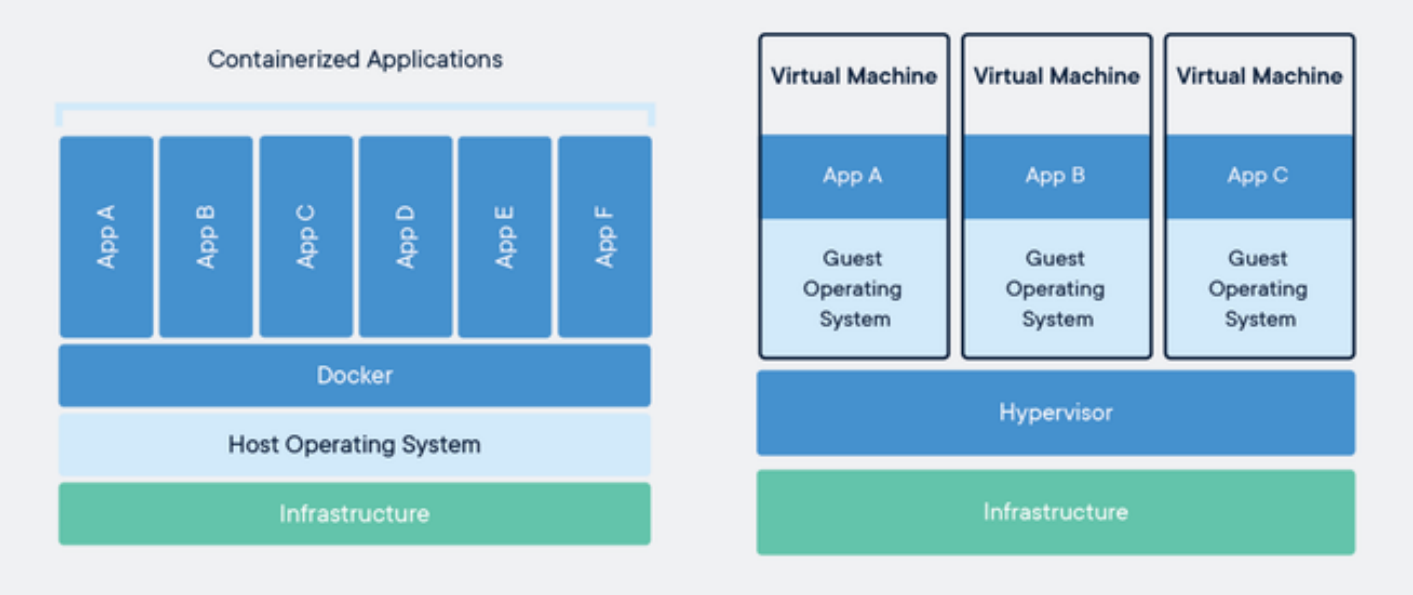
Debian package management

1. Контейнер танилцуулга
2. Docker
3. BuildX
   - Image-ийг бүртгэлд байршуулах
   - Үелэх build
   - Орчны хувьсагч
   - Нууц
   - Exporting build
4. Docker compose

Орчин үеийн веб апп ажиллуулахад гардаг асуудлууд

- Код, зөв тохируулга
- Сангууд, зөв хувилбар
- Компайлдсан кодыг ажиллуулах програм, interpreter, сервер
- Орчин, үйлдлийн системийн тохиргоо

VM vs Container

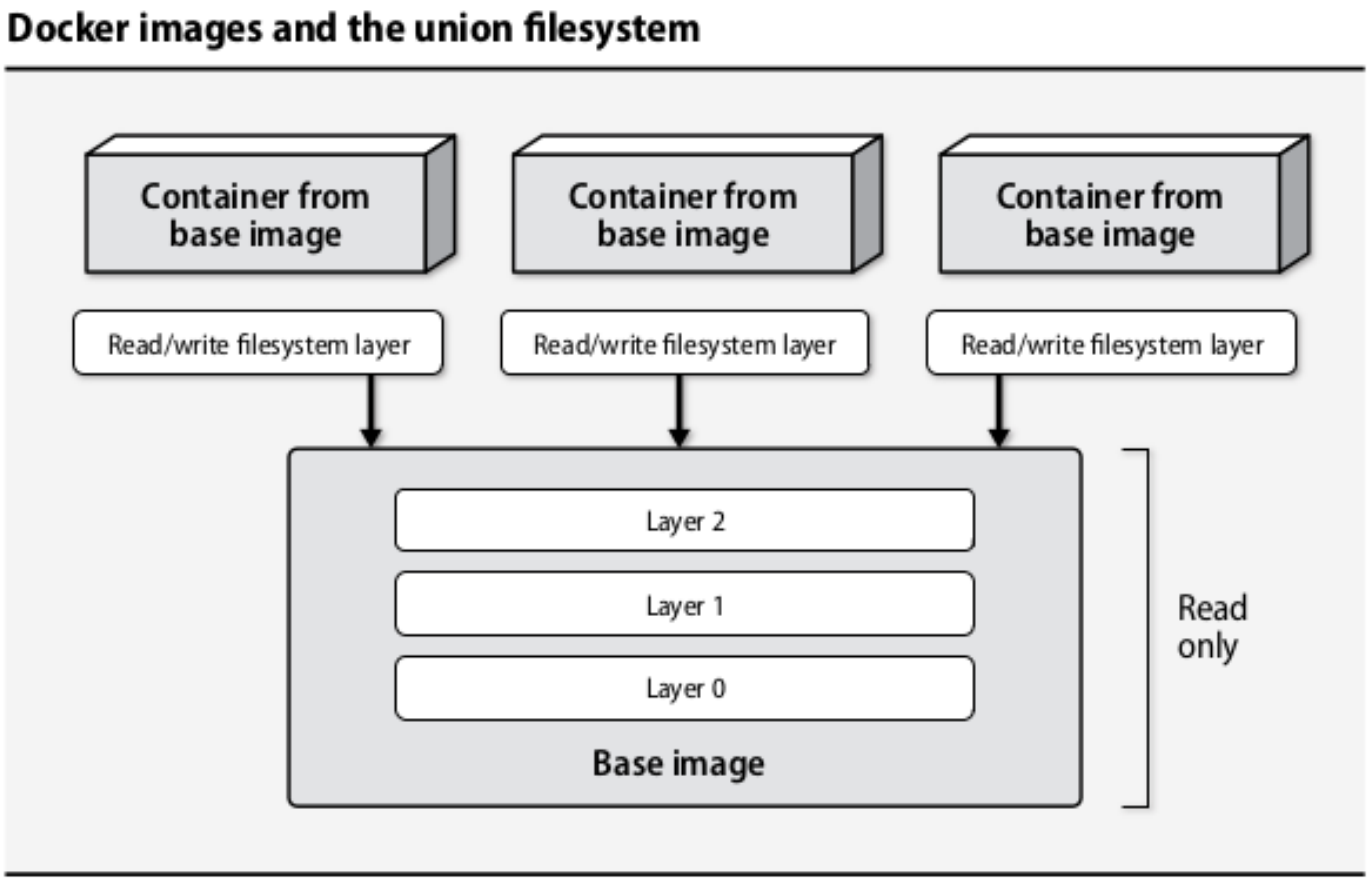


Зураг: VM vs Container

Үндсэн бүтэц

- Namespace
  - processes
  - filesystem
  - mounts
- Cgroups
  - CPU
  - Disk
  - Ram
- Capabilities
  - syscalls
  - ability to change system settings
- Secure computing mode (seccomp)
  - fine grained access to syscalls

Үндсэн бүтэц



Зураг: Контейнер тусгал (image)

Контейнер хувилбарууд

- Docker
- systemd-nspawn
- rkt
- containerd
- podman

Асуулт

Хэрэв нэг сервер дээр дараах 2-ын аль нэгийг ажиллуулах болсон бол аль нь илүү их ачаалал серверт өгөх вэ?

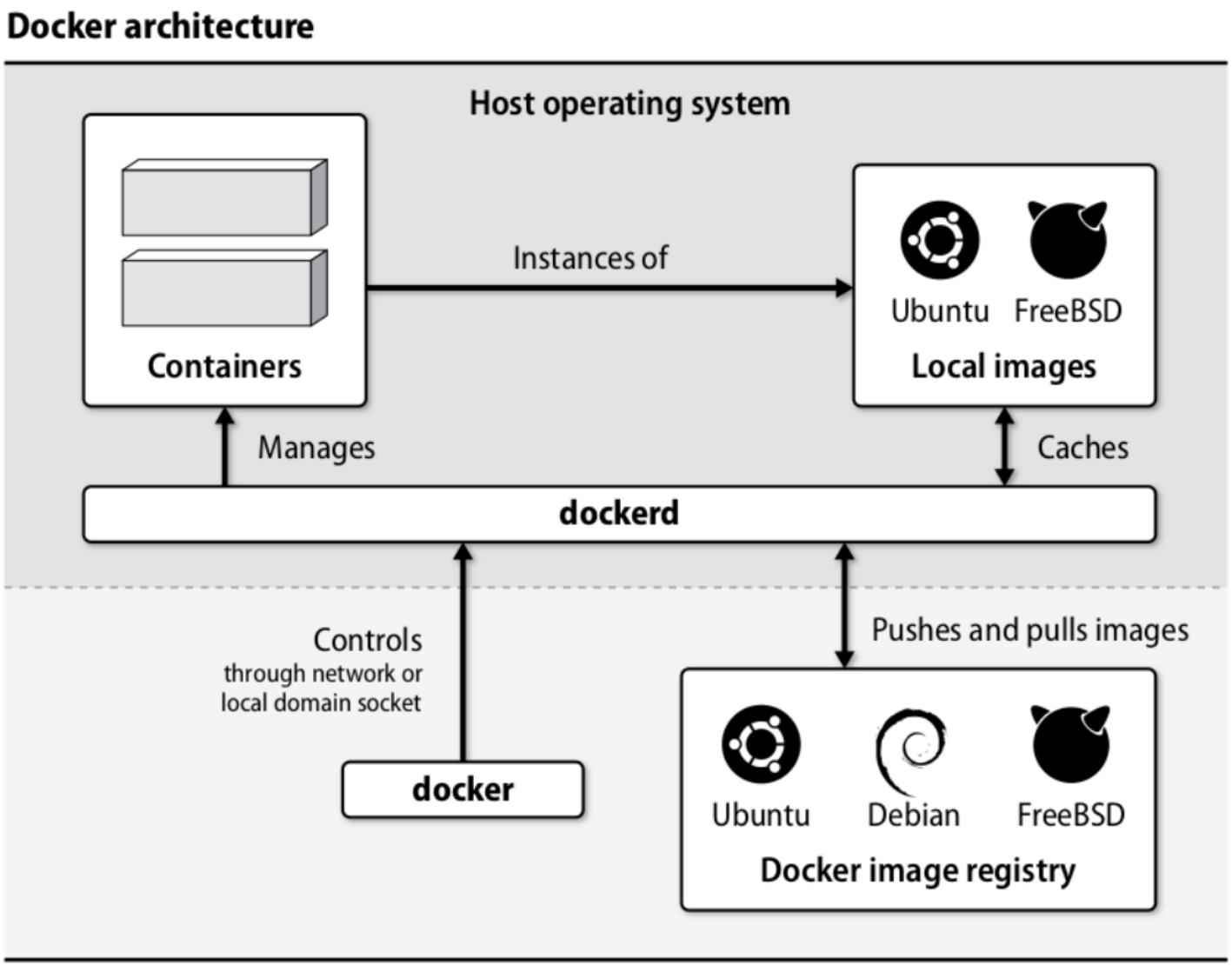
1. 2 веб сервисийг тус тусад нь 2 виртуал машин дотор ажиллуулах
2. 2 веб сервисийг тус тусад нь 2 контейнер дотор ажиллуулах

Асуулт

Docker-ийг аль үйлдлийн систем дээр ажиллуулж болох вэ?

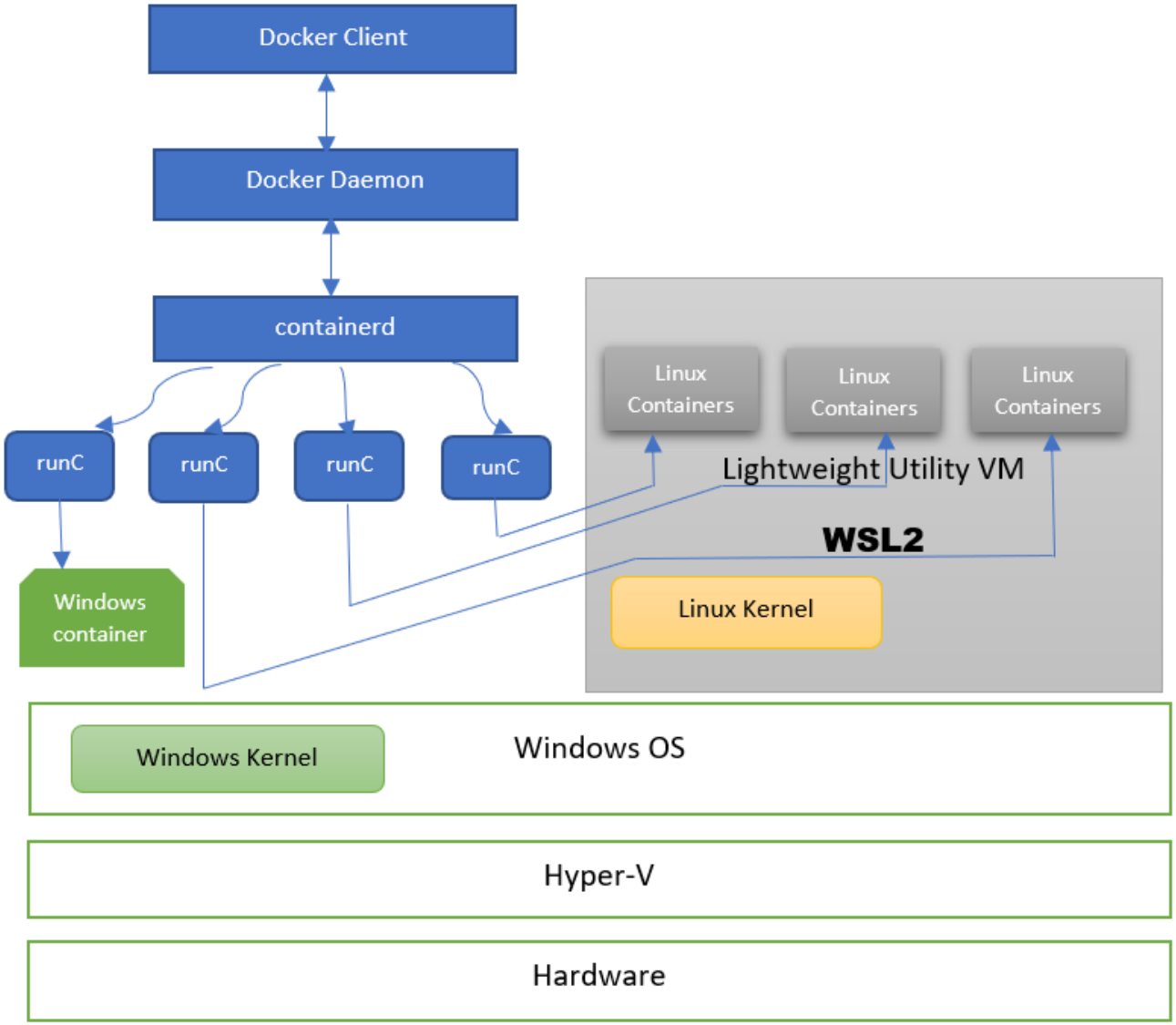
1. Windows
2. Linux
3. Mac OSX

Docker архитектур



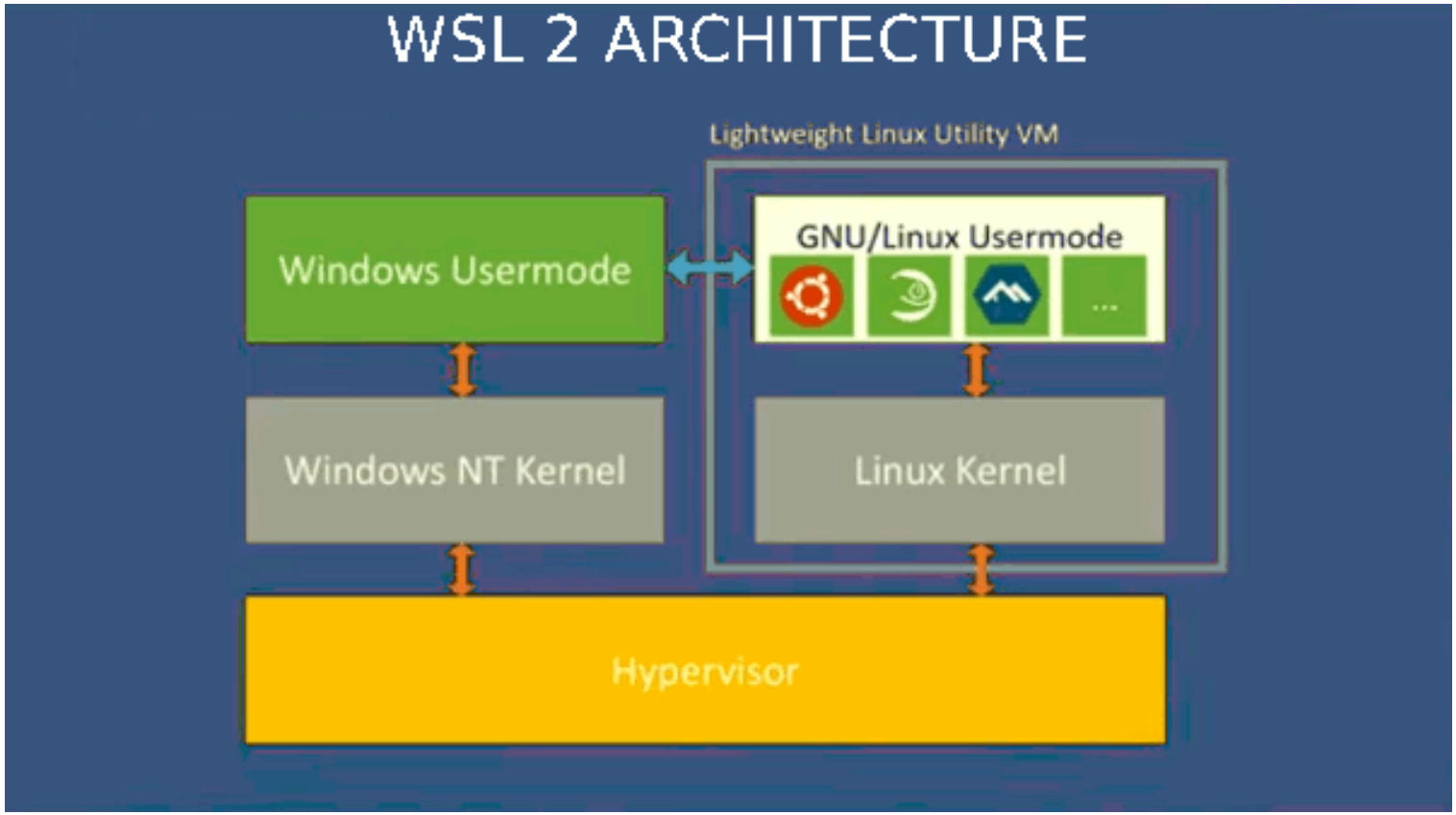
Зураг: Docker архитектур

Docker windows архитектур



Зураг: Docker архитектур

Windows Subsystem Linux (WSL)



Зураг: WSL архитектур

Docker суулгах, тохируулах

Тохируулга

docker бүлэгт нэмэхийн оронд sudo ашиглаж docker-т хандах эрх өгөх.

```
sudo systemctl start docker
```

Тохируулга

Итгэлцэлгүй docker image-ийг шууд татахыг хориглодог. Үүнийг дараах тохиргоогоор болиулах боломжтой

```
export DOCKER_CONTENT_TRUST=1
```

/etc/sudoers файлд оруулж өгнө.

```
Defaults env_keep += "DOCKER_CONTENT_TRUST"
```

Сервер дээр байгаа docker-той ажиллах

Remote docker

Сервер дээр байгаа docker daemon-ийг удирдахыг хүсвэл

```
export DOCKER_HOST=tcp://10.0.0.10:2376
```

орчингийн хувьсагчийг тохируулж өгнө.
TLS сертификат тохируулж өгч encrypted authentication хэрэглэхийг анхаар.

TLS arguments common to docker and dockerd

Argument	Meaning or argument
--tlsverify	Require authentication
--tlscert [a]	Path to a signed certificate
--tlskey [a]	Path to a private key
--tlscacert [a]	Path to the certificate of a trusted authority

a. Optional. The default locations are **~/.docker/{cert,key,ca}.pem**.

Docker командууд

Frequently used docker subcommands

Subcommand	What it does
docker info	Displays summary information about the daemon
docker ps	Displays running containers
docker version	Displays extensive version info about the server and client
docker rm	Removes a container
docker rmi	Removes an image
docker images	Displays local images
docker inspect	Displays the configuration of a container (JSON output)
docker logs	Displays the standard output from a container
docker exec	Executes a command in an existing container
docker run	Runs a new container
docker pull/push	Downloads images from or uploads images to a remote registry
docker start/stop	Starts or stops an existing container
docker top	Displays containerized process status

Зураг: Docker командууд

Image VS Container

Container

- Image-ээс үүсгэгдэнэ.
- Системийн нөөцийг эзэлнэ (CPU, RAM, network)
- Ажиллаж байгаа контейнерийг image болгон хадгалах боломжтой.
- Үндсэн үйлдлийн системээр баяжуулагдана.

Image

- Үйлдлийн системийн root файл системийг агуулна.
- Ажиллах програмын interpreter, сангуудыг агуулна.
- Амьгүй (ip хаяггүй, ямар ч системийн нөөц эзлэхгүй)

Жишээ командууд

```
$ sudo docker pull debian
Using default tag: latest
latest: Pulling from library/debian
f50f9524513f: Download complete
d8bd0657b25f: Download complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6...
Status: Downloaded newer image for debian:latest
```

Жишээ командууд

```
$ sudo docker images
REPOSITORY TAG       IMAGE ID        CREATED         SIZE
ubuntu     latest    07c86167cdc4    2 weeks ago     187.9 MB
ubuntu     wily      b5e09e0cd052    5 days ago      136.1 MB
ubuntu     trusty    97434d46f197    5 days ago      187.9 MB
ubuntu     15.04     d1b55fd07600    8 weeks ago     131.3 MB
centos     7         d0e7f81ca65c    2 weeks ago     196.6 MB
centos     latest    d0e7f81ca65c    2 weeks ago     196.6 MB
debian     jessie    f50f9524513f    3 weeks ago     125.1 MB
debian     latest    f50f9524513f    3 weeks ago     125.1 MB

$ sudo docker run debian /bin/echo "Hello World"
Hello World
```

hub.docker.com хуудсаас татах боломжтой image-үүдийг хайж болно.

Жишээ командууд

```
bob@host$ sudo docker run --hostname debian -it debian
                                                /bin/bash
root@debian:/# ls
bin dev home lib64 mnt proc run srv tmp var
boot etc lib media opt root sbin sys usr
root@debian:/# ps aux
USER PID %CPU %MEM  VSZ  RSS TTY STAT START TIME COMMAND
root 1   0.5 0.4 20236 1884   ?  Ss   19:02 0:00 /bin/bash
root 7   0.0 0.2 17492 1144   ?  R+   19:02 0:00 ps aux
root@debian:/# uname -r
3.10.0-327.10.1.el7.x86_64
root@debian:/# exit
exit
bob@host$ uname -r 3.10.0-327.10.1.el7.x86_64
```

Жишээ командууд

```
$ sudo docker run -p 80:80 --hostname nginx --name nginx -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fdd5d7827f33: Already exists
a3ed95caeb02: Pull complete
e04488adab39: Pull complete
2af76486f8b8: Pull complete
Digest: sha256:a234ab64f6893b9a13811f2c81b46cfac885cb141dcf4e275ed3
ca18492ab4e4
Status: Downloaded newer image for nginx:latest
0cc36b0e61b5a8211432acf198c39f7b1df864a8132a2e696df55ed927d42c1d
$ sudo docker ps
IMAGE COMMAND                  STATUS         PORTS
nginx "nginx -g 'daemon off"   Up 2 minutes   0.0.0.0:80->80/tcp
```

Жишээ командууд

- logs

```
$ sudo docker logs nginx
```

- image дээр програм суулгах

```
$ sudo docker
```

- зогсоох, эхлүүлэх устгах

```
$ sudo docker stop nginx
$ sudo docker start nginx
$ sudo docker ps
$ sudo docker rm
```

Volumes

Host систем дээрх бичиж болдог хавтсыг docker контейнерт оруулах боломжтой. Контейнер устсан ч уг volume дахь мэдээлэл хадгалагдан үлдэнэ.

```
$ sudo docker run -v /data --rm --hostname web
                                --name web -d nginx
$ sudo docker run -v /mnt/data:/data --rm
                                --name web -d nginx
$ sudo docker inspect nginx
```

Volumes

Data Volumes

Volume-ийг тусад нь үүсгээд, үүсгэсэн volume-ийг контейнерийг ажиллахад оруулах боломжтой.

```
# creating data-only container
$ sudo docker create -v /mnt/data:/data
                --name nginx-data nginx

# creating a new container using data-only
# container
$ sudo docker run --volumes-from nginx-data
              -p 80:80 --name web -d nginx
```

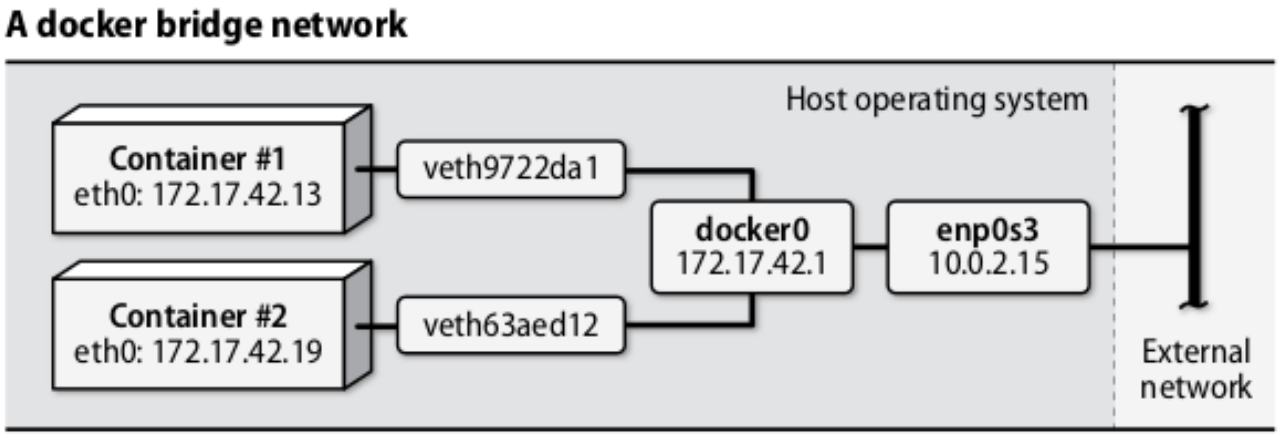
Ажиллаж байгаа контейнерын тохиргоог өөрчлөх

Commit

Тохиргоог өөрчлөхийн тулд хадгалаад, өөрчлөөд шинэ контейнер үүсгээд, хуучныг устгана.

```
$ docker commit web web:2
```

Docker Network



Зураг: Docker network

```
docker network ls
```

Docker Network

Docker container-ийн үйлчилгээнд нэрээр хандах

Нэрээр хандах боломжгүй. IP хаягаар л хандана. Гэхдээ нэг сүлжээнд байвал

```
$ docker network create mynet
$ docker run --net mynet --name test busybox \
                    nc -l 0.0.0.0:7000
$ docker run --net mynet busybox ping test
```

Docker debug

```
$ docker ps -a
$ docker logs <container_id>
```

Асуулт

Дараахын аль нь худал вэ?

1. Шинэ контейнер үүсгэхдээ *docker run* ашиглана.
2. Image татахдаа *docker pull*
3. Контейнер дотор ажиллаж байгаа процессийг *docker top*
4. Ажиллаж байгаа контейнерыг хадгалахдаа *docker save*

Асуулт

Docker контейнер үүсгэсэн боловч *docker ps*-д харагдахгүй байвал хамгийн түрүүнд юу хийх вэ?

1. Ахиад асаах гэж оролдоно.
2. Интернетээс хайна.
3. *docker logs* дуудна.

Асуулт

Контейнерыг зогсооход юу болох вэ?

1. Ажиллаж байсан процесс зогсоод, системийн төлөв хадгалагдан үлдэнэ.
2. Ажиллаж байсан процесс устаж, мэдээлэл нь хамт устана.
3. Тийм үйлдэл хийх боломжгүй.

Өөрийн image-ийг үүсгэх

Dockerfile нэртэй файлыг үүсгээд дотор нь ажиллуулах командуудыг бичнэ.

```
FROM debian:bookworm
RUN apt-get update \
&& apt-get install -y nginx \
&& rm -rf /var/lib/apt/lists/*
# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log
EXPOSE 80 443
VOLUME /data
CMD ["nginx", "-g", "daemon off;"]
```

```
$ docker build -t nginx:byme .
```

Өөрийн image-ийг үүсгэх

Abbreviated list of Dockerfile instructions

Instruction	What it does
ADD	Copies files from the build host to the image[a]
ARG	Sets variables that can be referenced during the build but not from the final image; not intended for secrets
CMD	Sets the default commands to execute in a container
COPY	Like ADD, but only for files and directories
ENV	Sets environment variables available to all subsequent build instructions and containers spawned from this image
EXPOSE	Informs **dockerd** of the network ports exposed by the container
FROM	Sets the base image; must be the first instruction
LABEL	Sets image tags (visible with **docker inspect**)
RUN	Runs commands and saves the result in the image
STOPSIGNAL	Specifies a signal to send to the process when told to quit with **docker stop**; defaults to SIGKILL
USER	Sets the account name to use when running the container and any subsequent build instructions
VOLUME	Designates a volume for storing persistent data
WORKDIR	Sets the default working directory for subsequent instructions

Зураг: Dockerfile-д ашиглаж болох командууд

Өөрийн image-ийг үүсгэх

```
# syntax=docker/dockerfile:1
FROM python:3
RUN pip install awscli
RUN --mount=type=secret,id=aws,target=/root/.aws/credentials \
  aws s3 cp s3://... ...
```

Өөрийн image-ийг registry-д байршуулах

Өөрийн image-ийн registry-д байршуулахад tag нь registry-ийнхаа URL-тай адил байх ёстой.

```
$ docker tag nginx:byme \
          registry.admin.com/nginx:byme .
$ docker login https://registry.admin.com
Username: bat
Password: ********************
$ docker push registry.admin.com/nginx:byme
```

Өөрийн image-ийг registry-д байршуулах

gitlab registry-д байршуулах

```
$ docker login registry.gitlab.com
Username: ggtulga
Password: <token>
$ docker build -t registry.gitlab.com/ggtulga/test .
$ docker push registry.gitlab.com/ggtulga/test
```

Build-д нэр өгч, үелсэн бүйлд хийх

```
FROM golang:1.23
WORKDIR /src
COPY <<EOF ./main.go
package main

import "fmt"

func main() {
  fmt.Println("hello, world")
}
EOF
RUN go build -o /bin/hello ./main.go

FROM scratch
COPY --from=0 /bin/hello /bin/hello
CMD ["/bin/hello"]
```

```
$ docker build -t hello .
```

Build-д нэр өгч, үелсэн бүйлд хийх

Өөр build-ээс файл хуулан авах боломж

```
COPY --from=nginx:latest /etc/nginx/nginx.conf /nginx.conf
```

Build-д нэр өгч, үелсэн бүйлд хийх

Өмнөх үеэс үргэлжлүүлэх

```
FROM alpine:latest AS builder
RUN apk --no-cache add build-base

FROM builder AS build1
COPY source1.cpp source.cpp
RUN g++ -o /binary source.cpp

FROM builder AS build2
COPY source2.cpp source.cpp
RUN g++ -o /binary source.cpp
```

Орчны хувьсагч

ARG, ENV хоёрын ялгаа: ENV бол image-ээс контейнерт хадгалагдаж үлддэг.

```
ARG NODE_VERSION="20"
ARG ALPINE_VERSION="3.20"

FROM node:${NODE_VERSION}-alpine${ALPINE_VERSION} AS base
WORKDIR /src

FROM base AS build
COPY package*.json ./
RUN npm ci
RUN npm run build

FROM base AS production
ENV NODE_ENV=production
COPY package*.json ./
RUN npm ci --omit=dev && npm cache clean --force
COPY --from=build /src/dist/ .
CMD ["node", "app.js"]
```

Нууц

```
RUN --mount=type=secret,id=aws \
    AWS_SHARED_CREDENTIALS_FILE=/run/secrets/aws \
    aws s3 cp ...

RUN --mount=type=secret,id=aws,target=/root/.aws/credentials \
    aws s3 cp ...

RUN --mount=type=secret,id=token,env=API_TOKEN,target=/app/token
```

```
$ docker build --secret id=aws,src=~/.aws/credentials .
$ docker build --secret id=token,env=API_TOKEN .
```

Export build

```
FROM alpine:latest AS build
RUN apk --no-cache add build-base
COPY hello.c hello.c
RUN gcc -o /hello hello.c
ENTRYPOINT ["/hello"]

$ docker build --platform=linux/amd64,linux/arm64 \
                    --output=out .
$ ls out/
```

Асуулт

nginx web серверийн тохиргоог өөрчлөн, web app-аа хуулж, шинэ image build хийх болсон бол Dockerfile-д бичигдэх дараах командуудыг зөв дараалалд оруул.

1. COPY nginx.conf /etc/nginx/nginx.conf
2. RUN mkdir /webapp
3. FROM nginx:latest
4. COPY webapp/* /webapp

Асуулт

Дараах spring boot Dockerfile-ийн доторхыг зөв дараалалд оруул.

1. `COPY ${DEPENDENCY}/META-INF /app/META-INF`
2. `USER spring:spring`
3. `ENTRYPOINT ["java","-cp","app:app/lib/*","hello.Applicatio`
4. `FROM openjdk:8-jdk-alpine`
5. `COPY ${DEPENDENCY}/BOOT-INF/lib /app/lib`
6. `RUN addgroup -S spring && adduser -S spring -G spring`
7. `ARG DEPENDENCY=target/dependency`
8. `COPY ${DEPENDENCY}/BOOT-INF/classes /app`

Docker compose

Dockerfile

```
FROM python:3.10-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run", "--debug"]
```

Docker compose

compose.yml

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
    develop:
      watch:
        - action: sync
          path: .
          target: /code
  redis:
    image: "redis:alpine"
```

```
$ sudo docker-compose up -d
$ sudo docker-compose down
```

Docker compose

compose.yml

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
    develop:
      watch:
        - action: sync
          path: .
          target: /code
  redis:
    image: "redis:alpine"
```

```
$ sudo docker-compose up -d
$ sudo docker-compose down
```