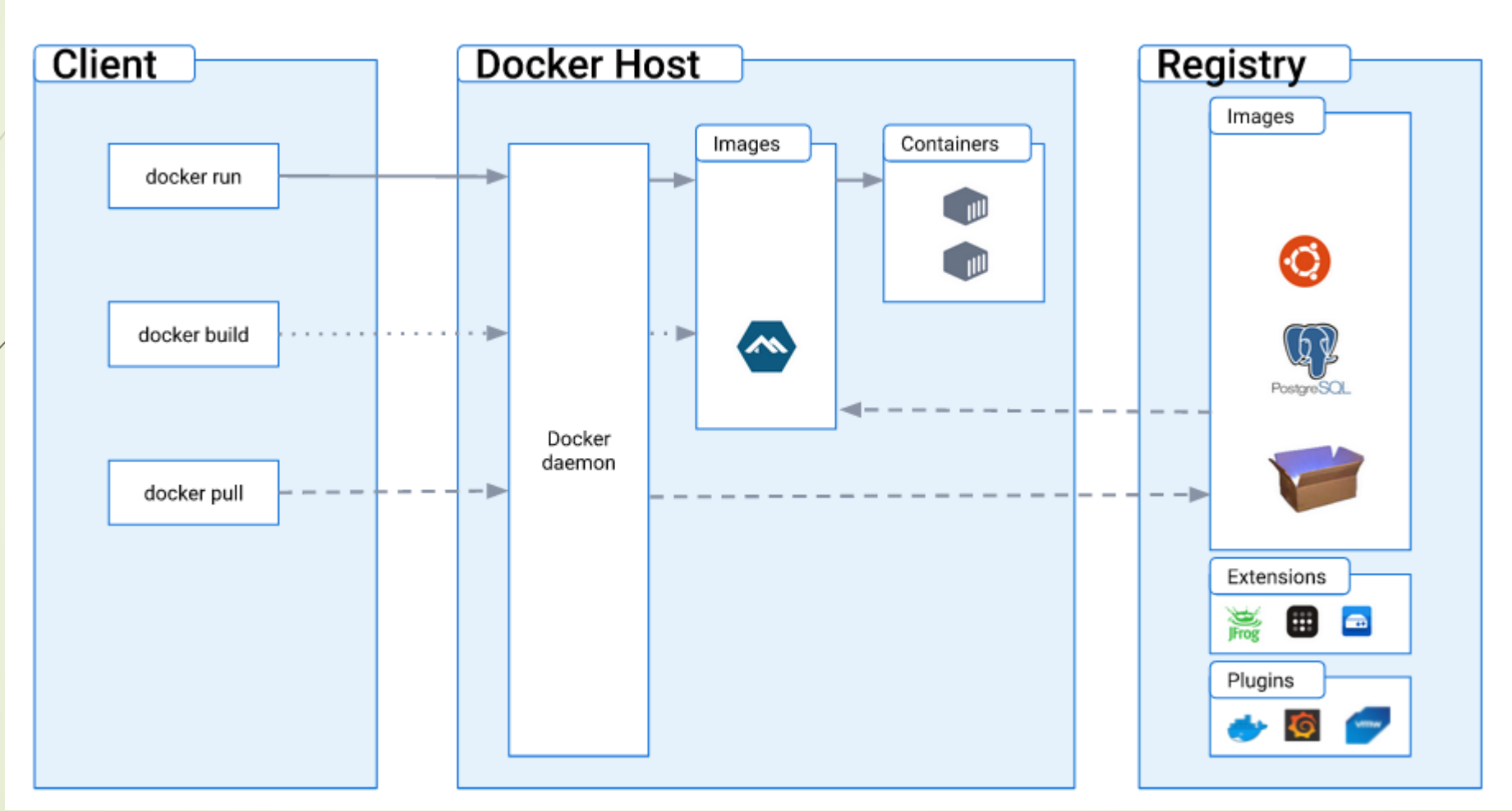
VIR1

- CPNV – ES
- Software development orientation
- 4th quarter - 2022-2023

- Glassey Nicolas

Docker Architecture?



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

nicolas.glassey@cpnv.ch

Writing Dockerfiles

Best practices

- A Docker images consist of:
  - Read-only layers (instruction).
  - Each one is a delta of the changes from the previous layer.

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

nicolas.glassey@cpnv.ch

Writing Dockerfiles

Best practices

- A Docker images consist of:
  - Read-only layers (instruction).
  - Each one is a delta of the changes from the previous layer.

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

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

nicolas.glassey@cpnv.ch

Writing Dockerfiles

Best practices

- A Docker images consist of:
  - Read-only layers (instruction).
  - Each one is a delta of the changes from the previous layer.

layer

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

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

nicolas.glassey@cpnv.ch

Writing Dockerfiles

Best practices

- A Docker images consist of:
  - Read-only layers (instruction).
  - Each one is a delta of the changes from the previous layer.

layer

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

- A running container (the container layer):
  - Is on top of the underlying layers
  - Is writable.

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

nicolas.glassey@cpnv.ch

Writing Dockerfiles

Best practices

- Create ephemeral containers
  - Can be stopped/destroyed/rebuild
  - With an absolute minimum of set up and configuration
- Don't install unnecessary packages
- Decouple applications
  - Each container should have only one concern
- Volume
  - Configuration storage
  - Database storage
  - Secrets
  - Files/Folders

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

nicolas.glassey@cpnv.ch

Docker Multi-stage builds

- Keep the size of images down
- Without manual effort

- Targets
  - Production -> only your application and the dependencies
  - Development -> everything needed to build your application

- Layer vs Multi-stage ?

https://docs.docker.com/build/building/multi-stage/

nicolas.glassey@cpnv.ch

Docker Multi-stage builds

```
# syntax=docker/dockerfile:1

FROM golang:1.16
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go ./
RUN CGO_ENABLED=0 go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis/href-counter/app ./
CMD ["./app"]
```

https://docs.docker.com/build/building/multi-stage/

nicolas.glassey@cpnv.ch

Docker Multi-stage builds

```
# syntax=docker/dockerfile:1

FROM golang:1.16
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go ./
RUN CGO_ENABLED=0 go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis/href-counter/app ./
CMD ["./app"]
```

https://docs.docker.com/build/building/multi-stage/

nicolas.glassey@cpnv.ch

Docker Multi-stage builds

```
# syntax=docker/dockerfile:1

FROM golang:1.16
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go ./
RUN CGO_ENABLED=0 go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis/href-counter/app ./
CMD ["./app"]
```

https://docs.docker.com/build/building/multi-stage/

nicolas.glassey@cpnv.ch

Docker Multi-stage builds

```
# syntax=docker/dockerfile:1

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
```

https://docs.docker.com/build/building/multi-stage/

nicolas.glassey@cpnv.ch

Docker - Development

Best practices

- **How to keep your images small**
  - Start with an appropriate base image
  - Use multistage builds
  - Create your own base image
  - Consider using the production image as debug base image
  - Use appropriate tag. Do not rely on automatic “latest” tag.

https://docs.docker.com/develop/dev-best-practices/

nicolas.glassey@cpnv.ch

Docker - Development

Best practices

- **Where and how to persist application data**
  - Avoid storing application data directly in container -> use volumes
  - During development, use bind mounts
  - Use Secrets to store sensitive application data
  - Use Configs to store non-sensitive application data

https://docs.docker.com/develop/dev-best-practices/

nicolas.glassey@cpnv.ch

Docker - Development

Best practices

- **Use CI/CD for testing and deployment**

  - Use Docker Hub (or another CI/CD) to automatically build and tag your Docker image

  - Sign images before deployment into production.

https://docs.docker.com/develop/dev-best-practices/

nicolas.glassey@cpnv.ch

Docker - Development

Best practices

- **Dev vs Prod Environments**

Development	Production
Use bind mounts to give your container access to your source code.	Use volumes to store container data.
Use Docker Desktop for Mac or Docker Desktop for Windows.	Use Docker Engine, if possible with userns mapping for greater isolation of Docker processes from host processes.
Don't worry about time drift.	Always run an NTP client on the Docker host and within each container process and sync them all to the same NTP server. If you use swarm services, also ensure that each Docker node syncs its clocks to the same time source as the containers.

Isolate containers with a user namespace

https://docs.docker.com/develop/dev-best-practices/

nicolas.glassey@cpnv.ch

Docker – Security

Best practices

- Use image from trusted source and keep it smaller as possible
- Use multi-stage builds
- Rebuild images
- Check your images for vulnerabilities

https://docs.docker.com/develop/security-best-practices/

nicolas.glassey@cpnv.ch

Docker – Security

Best practices

- Use image from trusted source and keep it smaller as possible

Base image	Size	Time to install tcpdump
alpine:3.11	5.6 MB	1-2s
archlinux:20200106	409 MB	7-9s
centos:8	237 MB	5-6s
debian:10	114 MB	5-7s
fedora:31	194 MB	35-60s
ubuntu:18.04	64 MB	6-8s

https://jpetazzo.github.io/2020/03/01/quest-minimal-docker-images-part-2/

nicolas.glassey@cpnv.ch

Docker – Security

Best practices

- Use multi-stage builds

https://docs.docker.com/develop/security-best-practices/

nicolas.glassey@cpnv.ch

Docker – Security

Best practices

- Use image from trusted source and keep it smaller as possible
- Use multi-stage builds
- Rebuild images
- Check your images for vulnerabilities

https://docs.docker.com/develop/security-best-practices/

nicolas.glassey@cpnv.ch

Bibliography

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

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

https://docs.docker.com/build/building/multi-stage/

https://docs.docker.com/develop/dev-best-practices/

nicolas.glassey@cpnv.ch