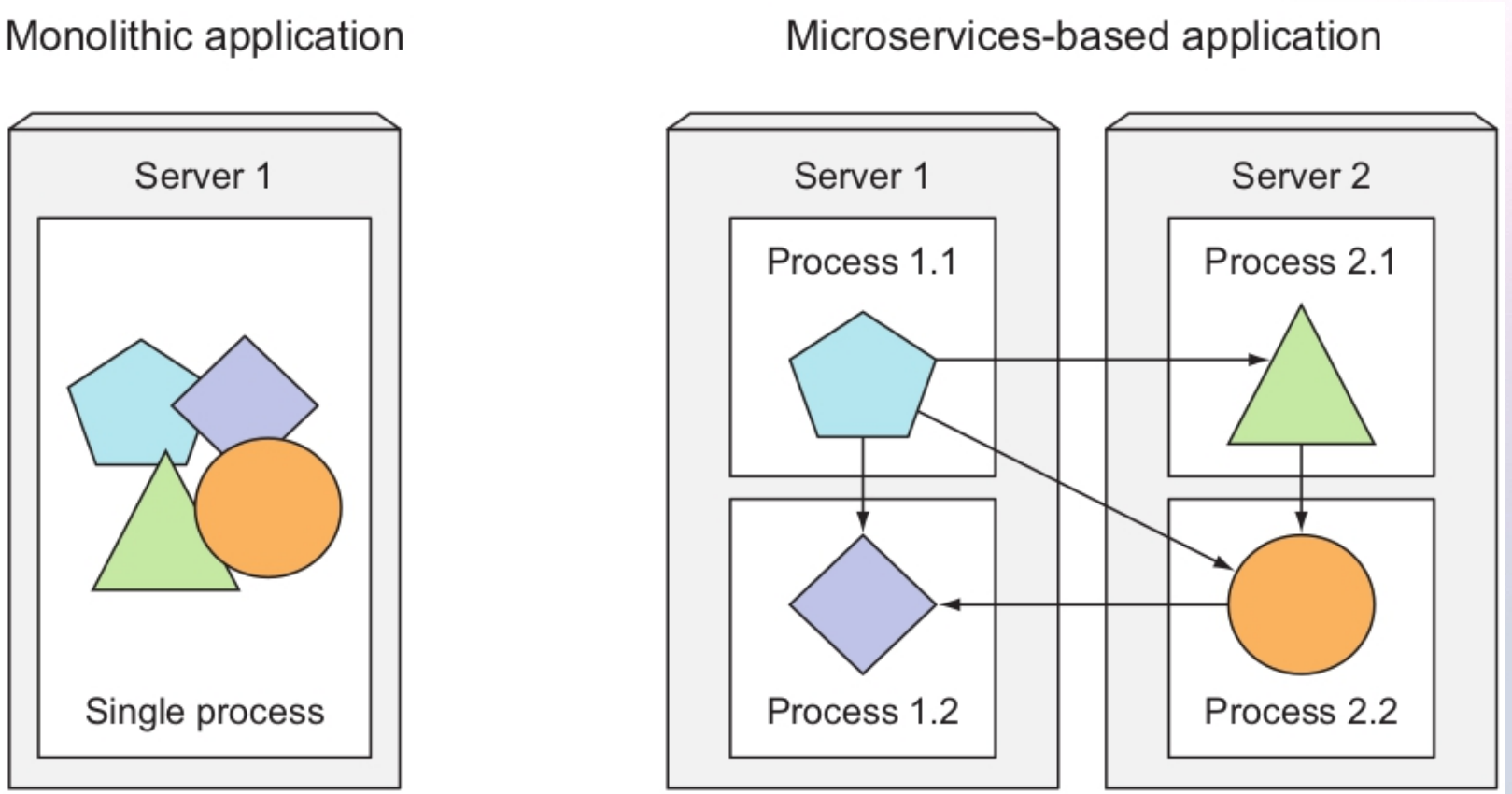
sourcemind

Docker basics

Tigran Vardanyan

ML Engineer

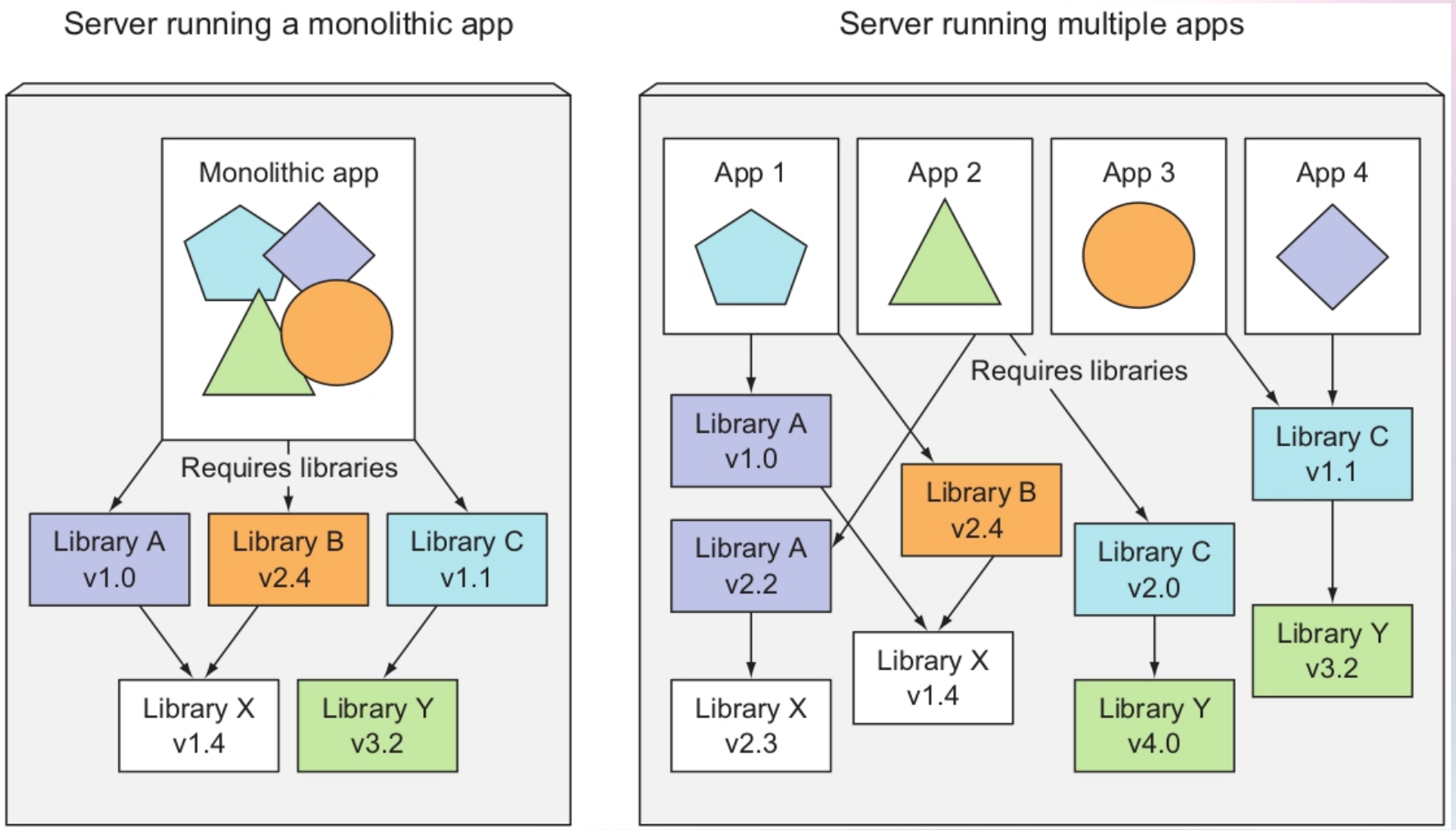
Moving from monolithic apps to microservices



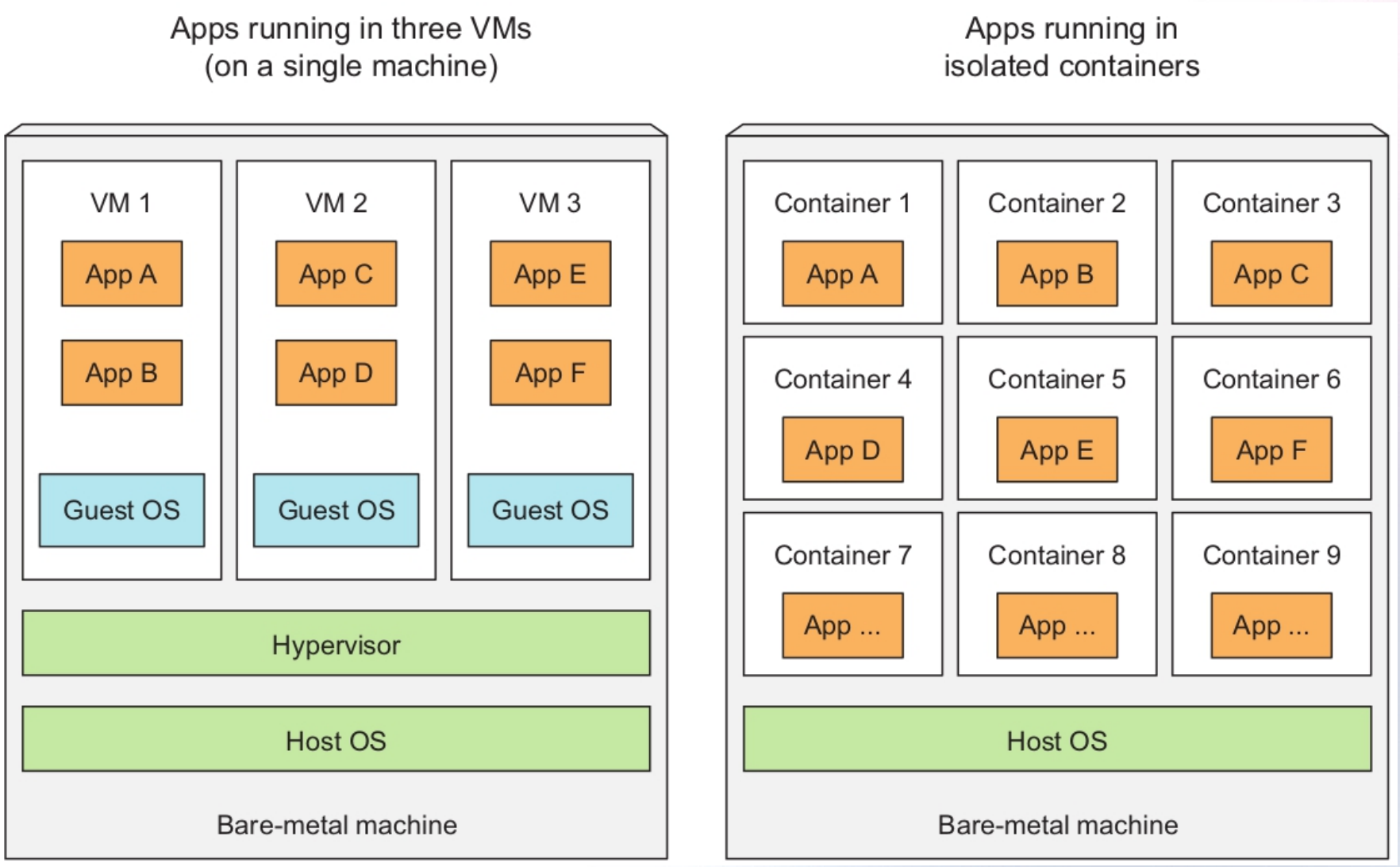
- Each microservice runs as an independent process and communicates with other microservices through well-defined interfaces (APIs).
- F. e. through synchronous protocols such as **HTTP**, over which they usually expose **REST**ful (Representational State Transfer) APIs.

sourcemind

Divergence of environment requirements



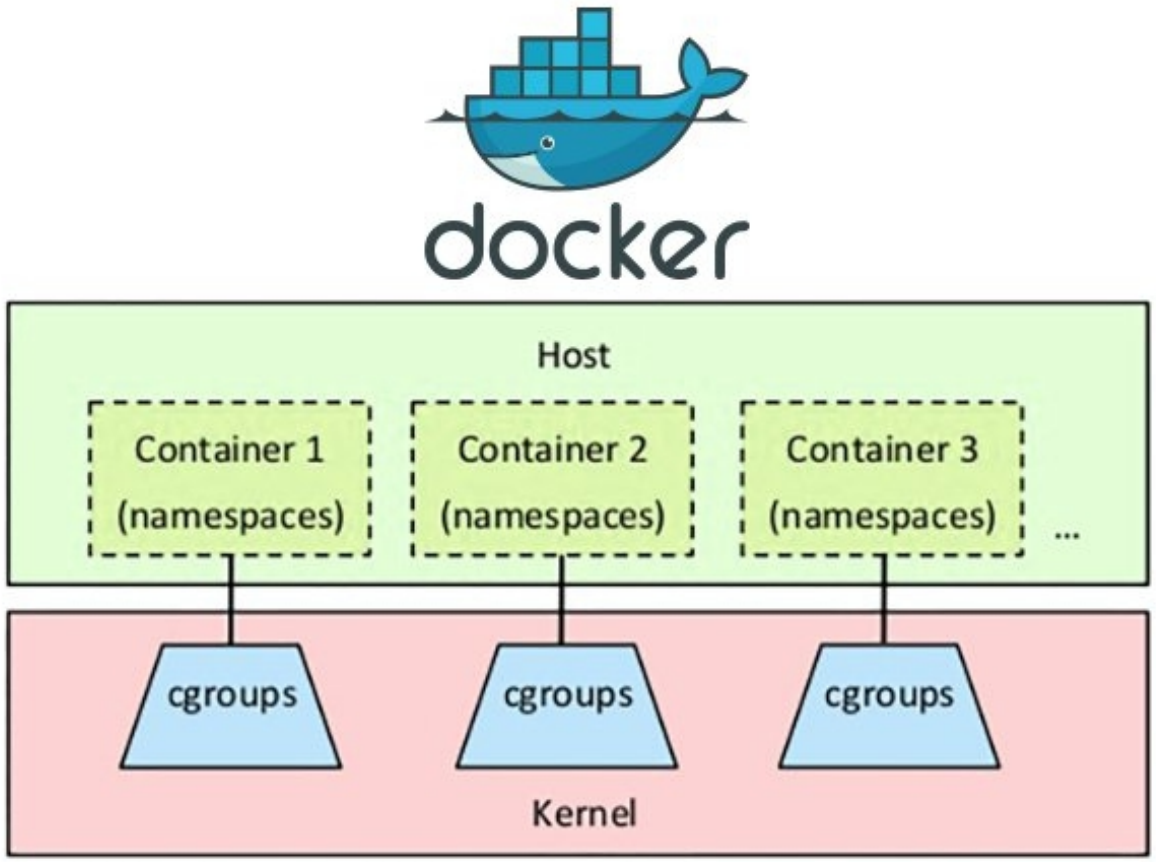
Comparing virtual machines to containers



- containers are much more lightweight
- allows you to run higher numbers of software components on the same hardware
- each VM needs to run its own set of system processes which requires additional compute resources

sourcemind

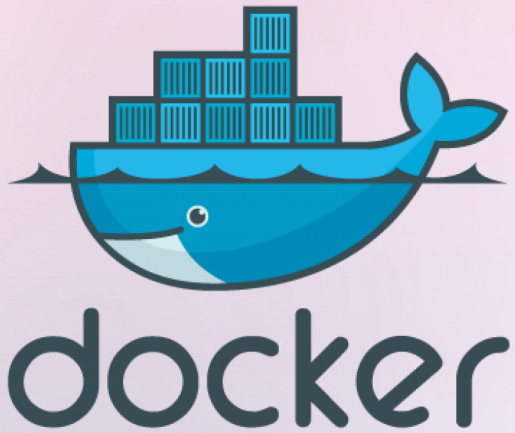
Mechanisms that make container isolation possible



- **Linux Namespaces** – makes sure each process sees its own personal view of the system (files, processes, network interfaces, hostname, and so on)
- **Linux Control Groups** (**cgroups**) – limits the amount of resources the process can consume (CPU, memory, network bandwidth, and so on)

What is a docker

- Docker is an open platform for developing, shipping, and running applications.
- Docker was the first container system that made containers easily portable across different machines.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- A big difference between Docker–based container images and VM images is that container images are composed of **layers**, which can be shared and reused across multiple images.



Docker installation

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

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.

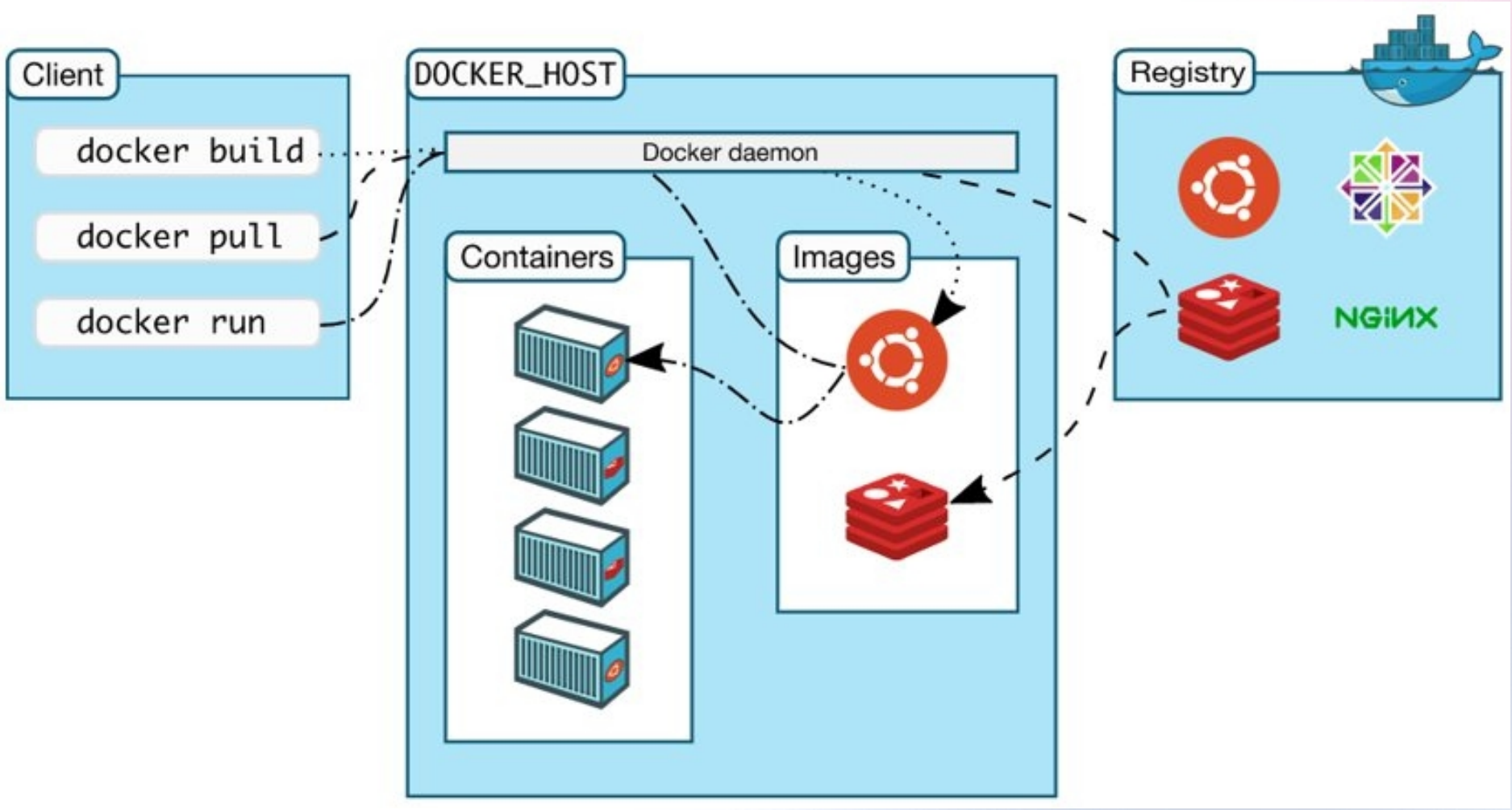
Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

Docker main concepts

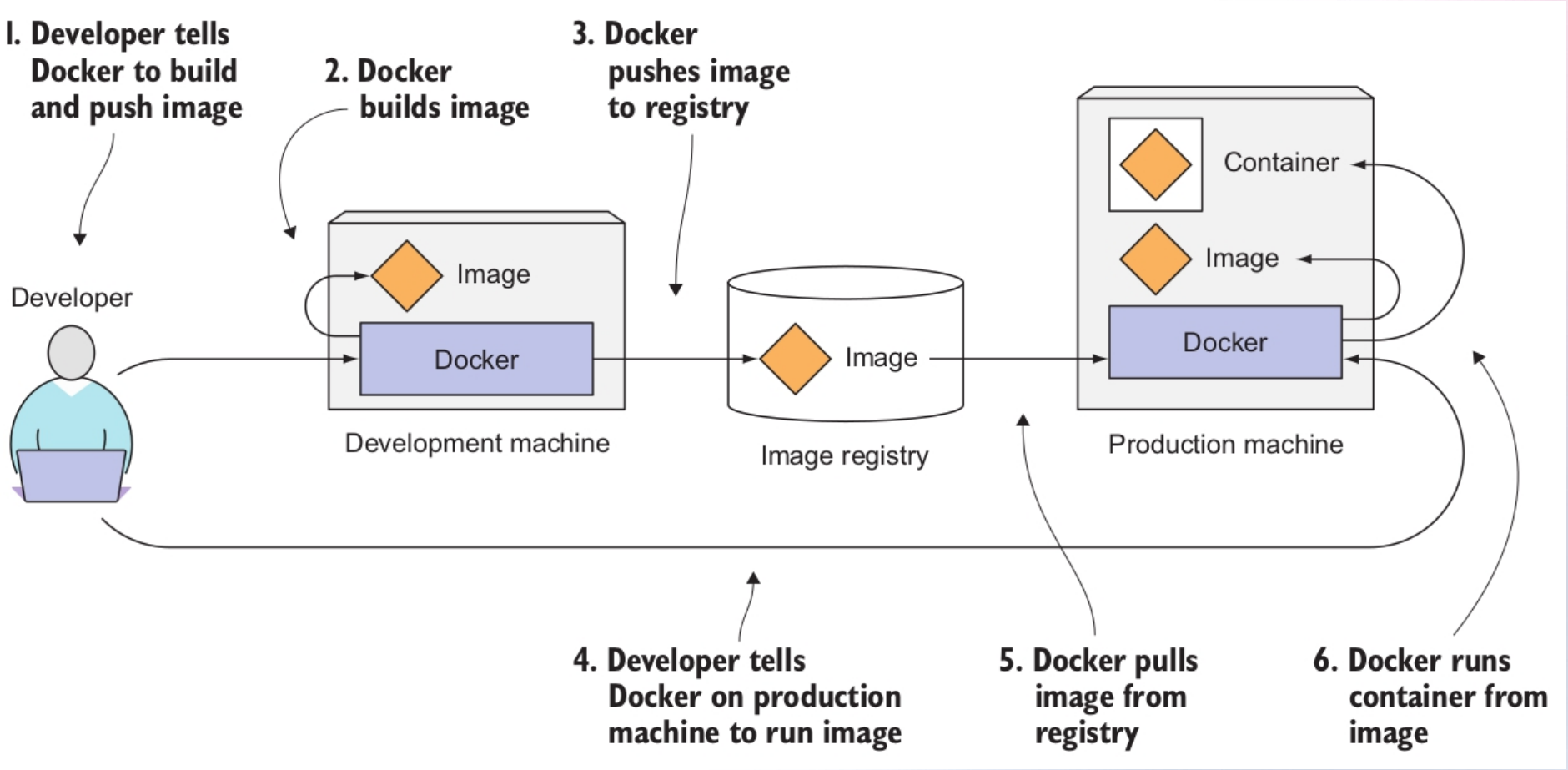
- *Images* – A Docker-based container image is something you package your application and its environment into. It contains the filesystem that will be available to the application and other metadata, such as the path to the executable that should be executed when the image is run.
- *Registries* – A Docker Registry is a repository that stores your Docker images and facilitates easy sharing of those images between different people and computers. Certain registries are public, allowing anyone to pull images from it, while others are private, only accessible to certain people or machines.
- *Containers* – A Docker-based container is a regular Linux container created from a Docker-based container image. A running container is a process running on the host running Docker, but it's completely isolated from both the host and all other processes running on it.

Docker architecture



- docker uses a client–server architecture
- docker client and daemon communicate using a REST API
- docker *registry* stores Docker images

Building, distributing, and running a docker image



Docker creates an isolated container based on the image and runs the binary executable specified as part of the image.

Dockerfile

```
FROM tensorflow/tensorflow:latest-gpu-py3

## ensure locale is set during build
ENV LANG C.UTF-8

RUN pip install keras && \
    pip install uproot && \
    pip install jupyter && \
    pip install matplotlib && \
    pip install papermill && \
    mkdir /afs

RUN apt-get update && apt-get install -y nano

COPY . /btagging
WORKDIR /btagging
```

Order matters for caching

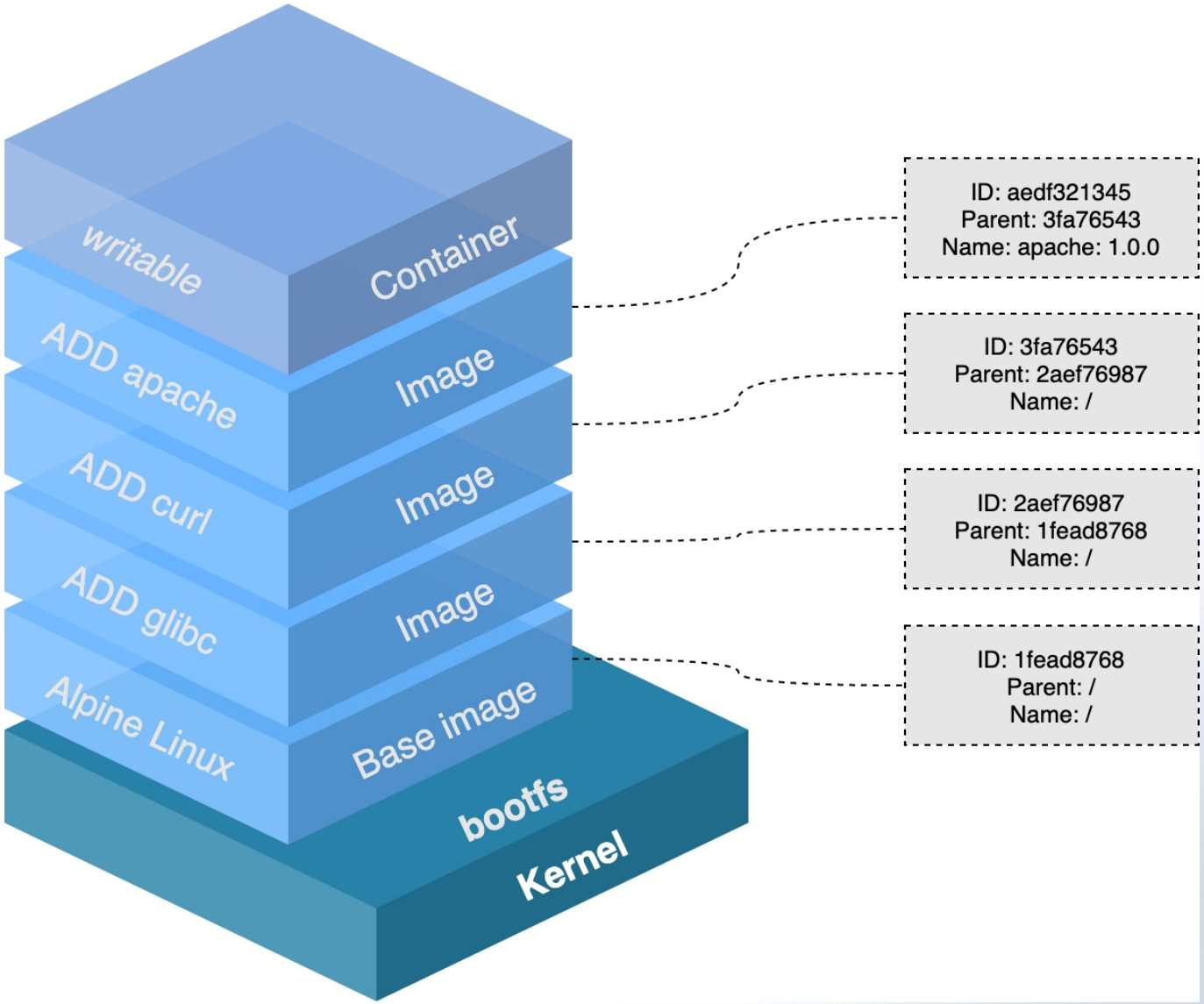
```
FROM debian
COPY . /app
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk ssh vim
COPY . /app
CMD ["java", "-jar", "/app/target/app.jar"]
```

Order from least to most frequently changing content.

- Docker can build images automatically by reading the instructions from a *Dockerfile*.
- A *Dockerfile* is a text document that contains all the commands a user could call on the command line to assemble an image.

sourcemind

Understanding image layers



- Layers make distribution of images across the network more efficient
- Layers reduce the storage footprint of images
- Container image layers are read-only

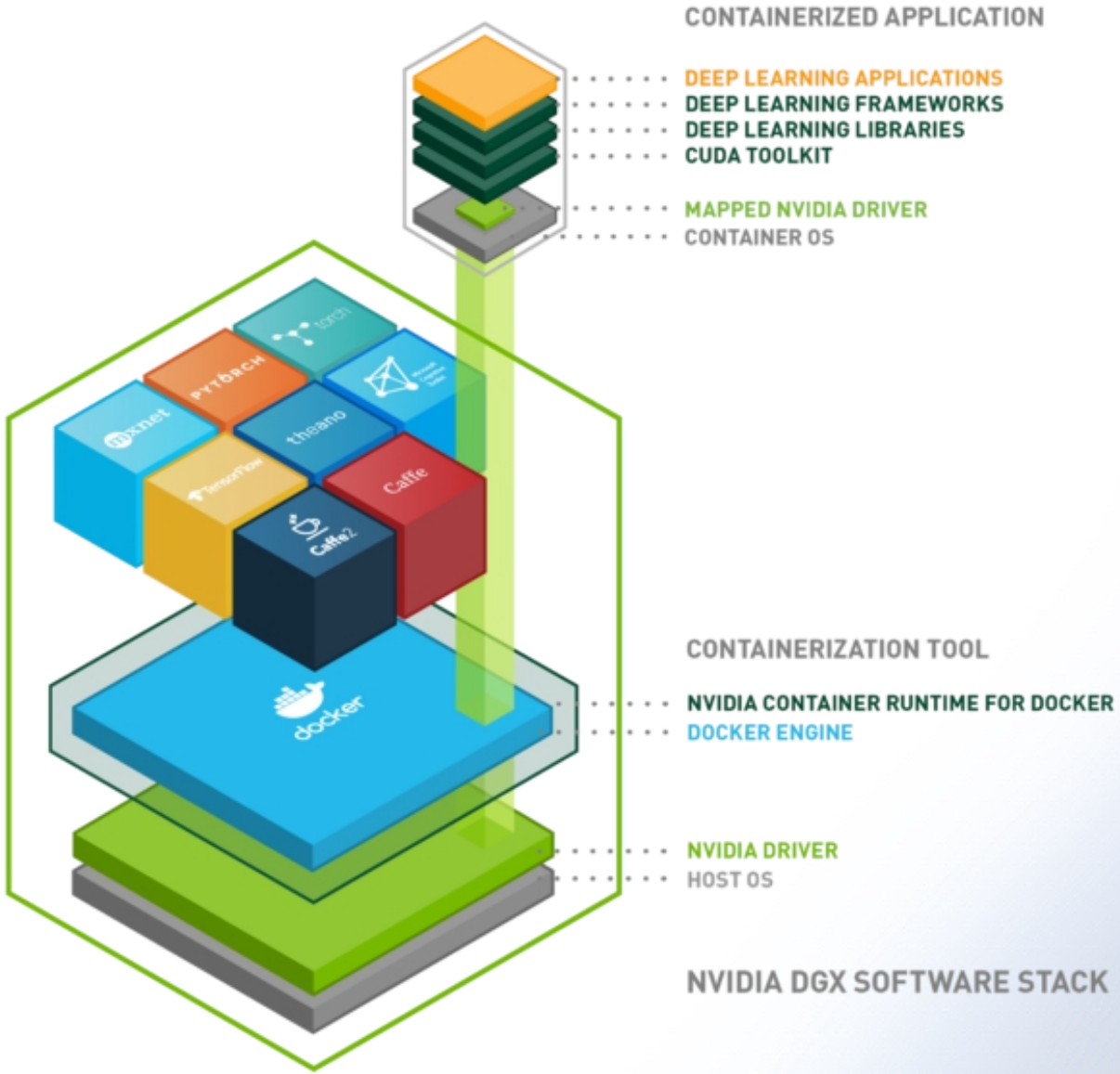
docker history [image_name]

sourcemind

Share docker container

- docker commit CONTAINER_NAME YOUR–REPO–NAME/app–image
- docker push YOUR–REPO–NAME/app–image
- docker save YOUR–REPO–NAME/app–image –o app–image.tar.gz
- docker load –i app–image.tar.gz

GPU within a Docker Container



apt-get install **nvidia-container-runtime**

docker run -it --rm **--gpus all** ubuntu nvidia-smi

sourcemind

Thank you !

sourcemind