

Docker KBTG Day 2

- Training Day -

Get Slide

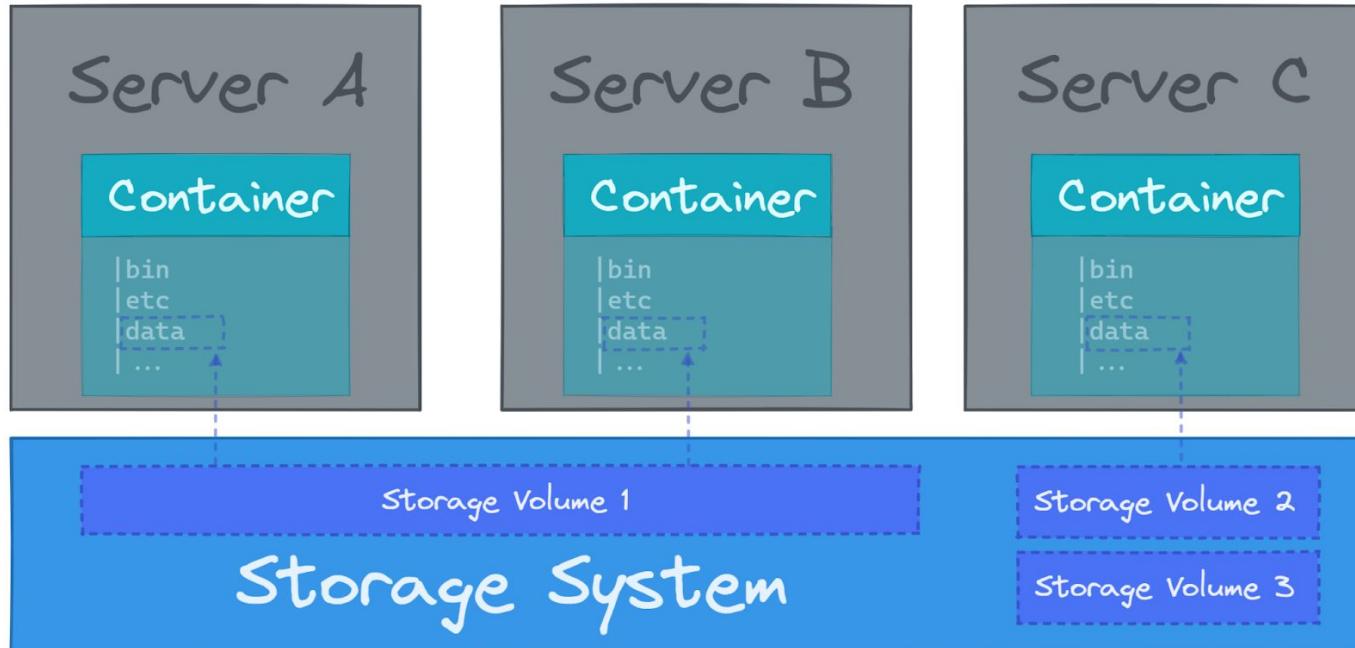


SCAN ME

Advanced Docker Usage

Storage (Cont.)

- Data is shared between two containers on the same host:



Activity Time

- With Paper Base -

Output what you learned in **Practical Action**

Hands-on Workshop:

Network and Storage

Linking Docker Containers

1. Change directory to advance/link-container
2. Run mongo container
3. Run mongo-express container
4. Explore mongo-express
 - a. Open web browser with localhost:8081
 - user: admin
 - password: pass
5. Execute shell to mongo-express container
6. Try to ping to mongo for check dns resolve
7. Print environment
8. Exit

Initialize data in MongoDB (Mount volume)

1. Change directory to advance/mount-volume
2. Run mongo container with mount volume initial data
3. Run mongo-express with connect mongo container with mount volume
4. Explore mongo-express with world database
 - a. Open web browser with localhost:8082
 - user: kbtg
 - password: gtbk

Docker Security Best Practices

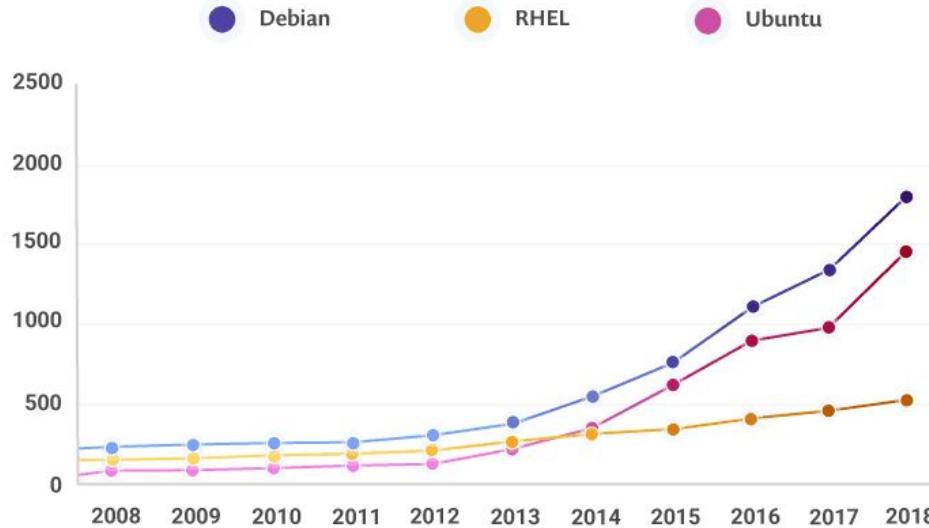
Not only it work

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

Docker Vulnerability

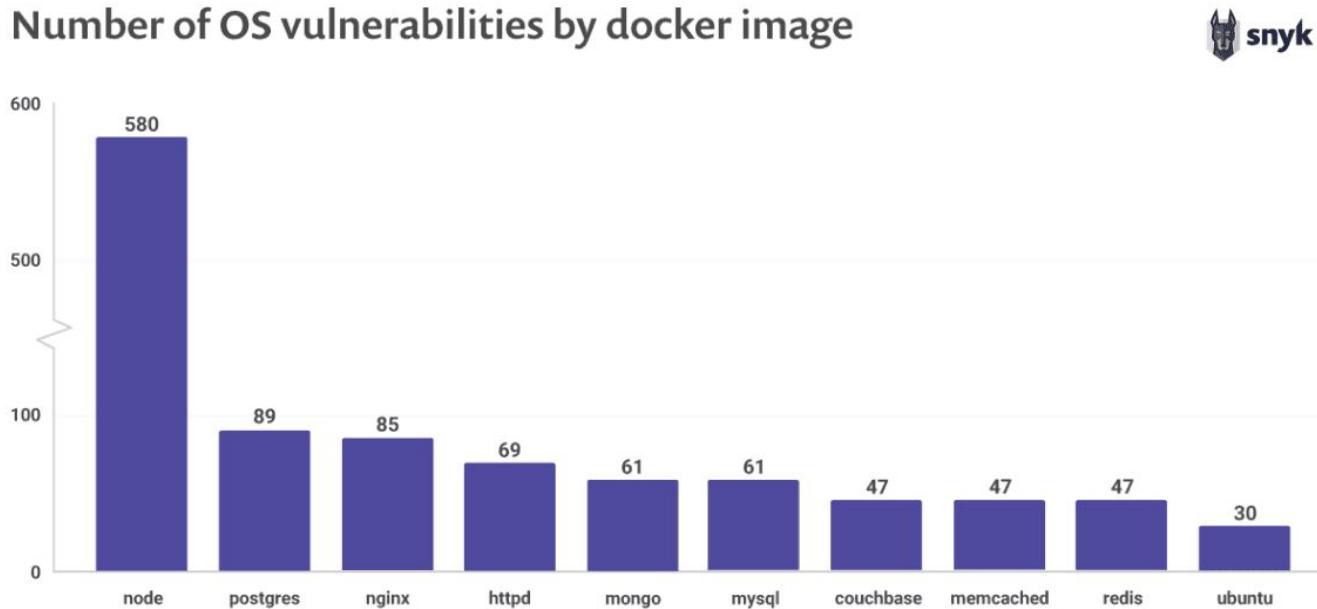
Linux OS vulnerabilities steadily increasing



Reference Pictures:

- [Top ten most popular docker images each contain at least 30 vulnerabilities](#)

Docker Vulnerability (Cont.)



Reference Pictures:

- [Top ten most popular docker images each contain at least 30 vulnerabilities](#)

Security best practices

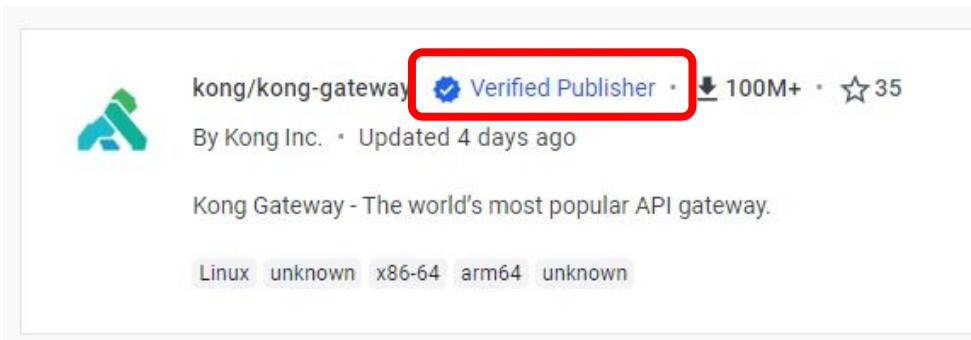
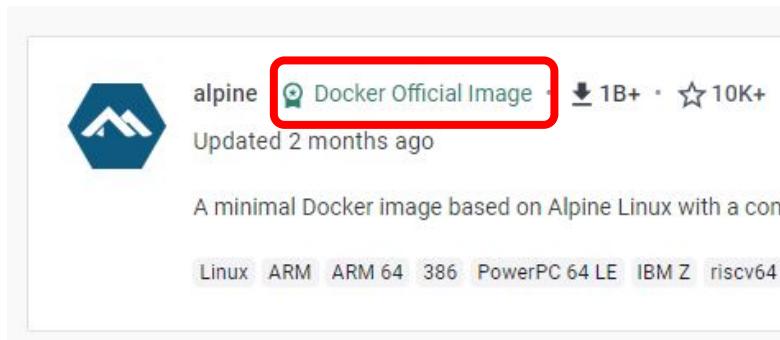
1. [Choosing the right base image](#)
2. [Using multi-stage builds](#)
3. [Rebuilding images](#)
4. [Checking your image for vulnerabilities](#)

Reference:

- [Building best practices](#)

Choosing the right base image

- Official Image and Verified Publisher badges.
- A small image with minimal dependencies can considerably lower the attack surface.



Reference:

- [Building best practices](#)

Container Security

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Container Security(Cont.)

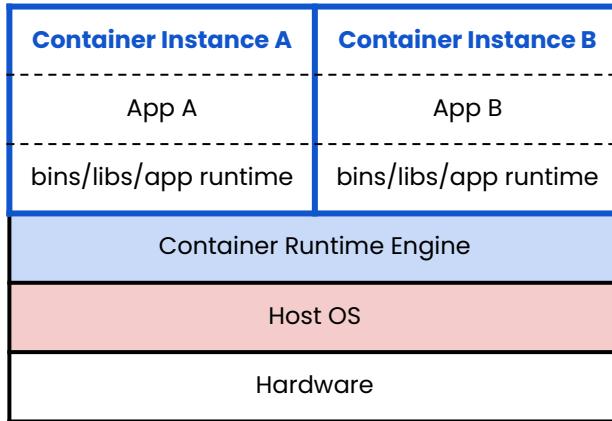
- It is important to understand that containers have **different security requirements than virtual machines.**

Reference:

- [Linux kernel capabilities](#)

Container Security(Cont.)

- It is important to understand that containers have different security requirements than virtual machines. A lot of people **rely on the isolation** properties of containers, but that can **be very dangerous**.

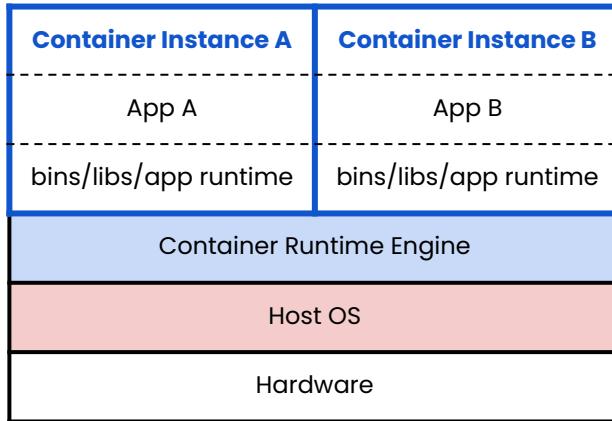


Reference:

- [Linux kernel capabilities](#)

Container Security(Cont.)

- When containers are started on a machine,

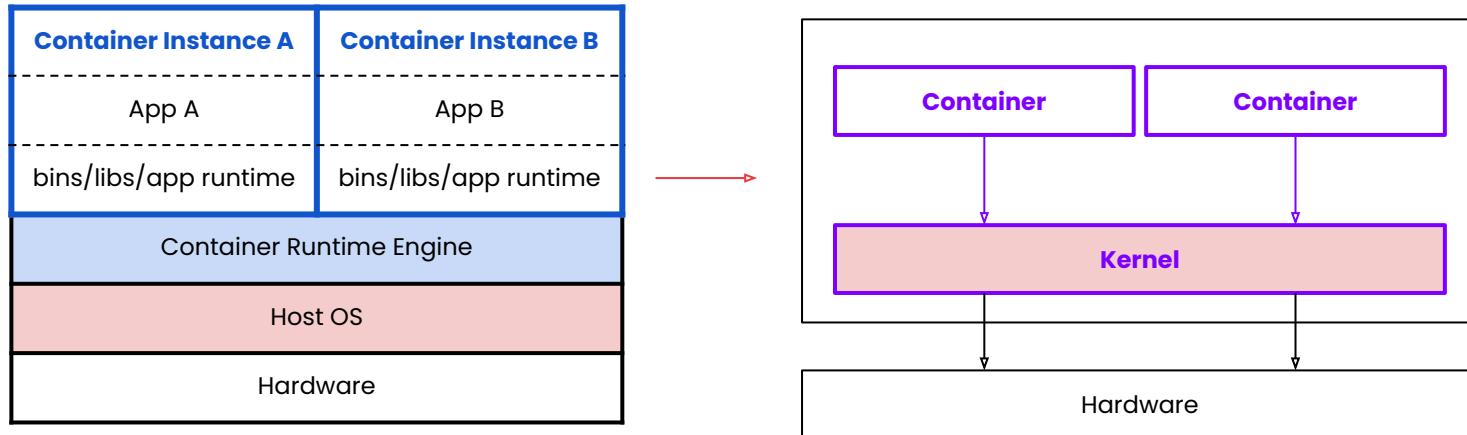


Reference:

- [Linux kernel capabilities](#)

Container Security(Cont.)

- When containers are started on a machine, they **always share the same kernel**, which then becomes a risk for the whole system,

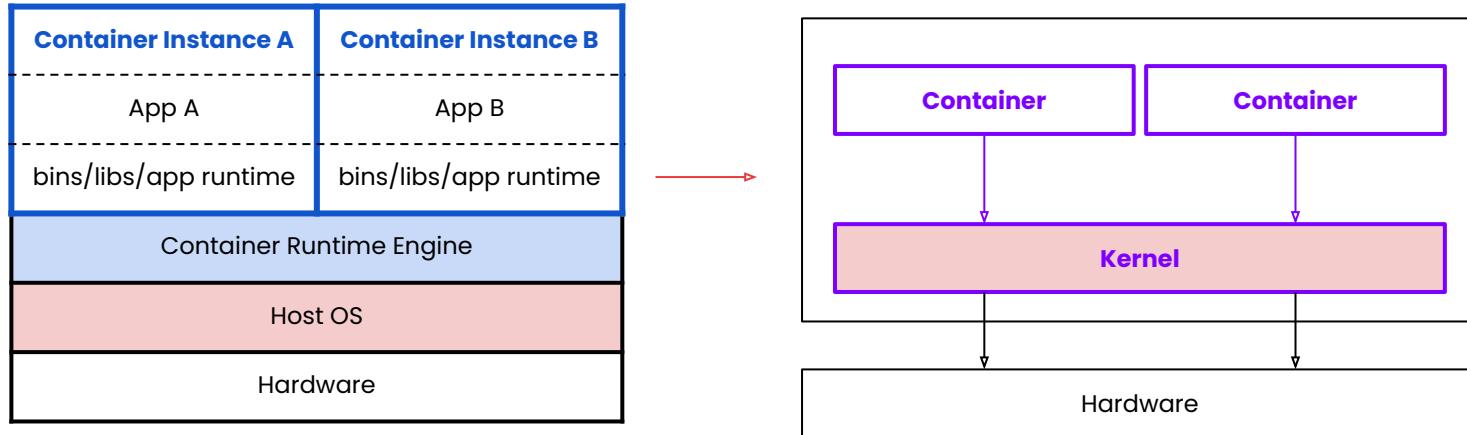


Reference:

- [Linux kernel capabilities](#)

Container Security(Cont.)

- When containers are started on a machine, they always share the same kernel, which then becomes a risk for the whole system, if containers are allowed to call kernel functions

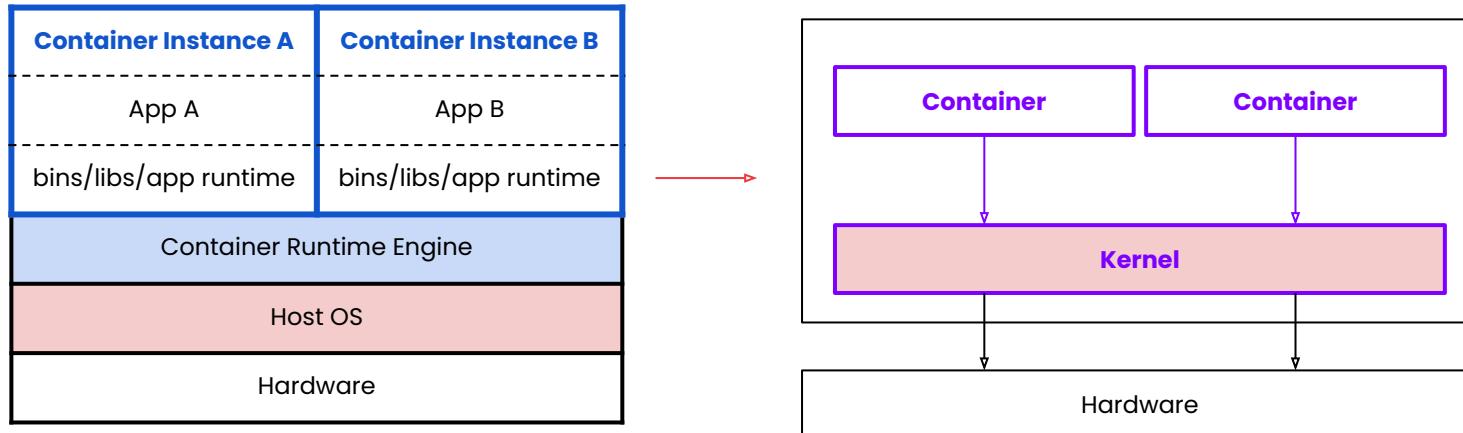


Reference:

- [Linux kernel capabilities](#)

Container Security(Cont.)

- When containers are started on a machine, they always share the same kernel, which then becomes a risk for the whole system, if containers are allowed to call kernel functions like for example **killing other processes or modifying the host network** by creating routing rules.



Reference:

- [Linux kernel capabilities](#)

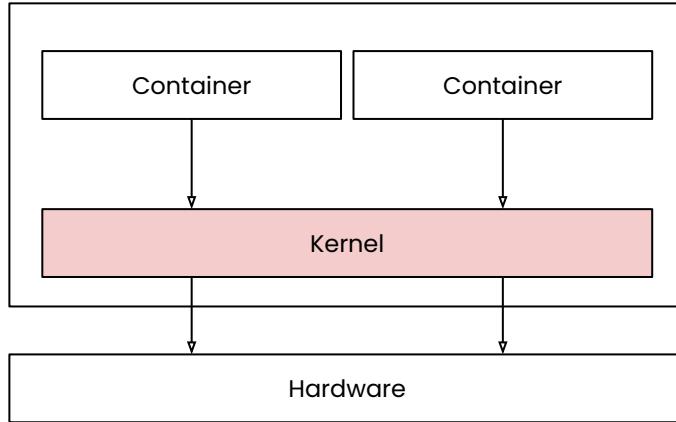
Privilege Escalation

- Container Security -

Privilege Escalation

Privilege Escalation (Cont.)

- One of the greatest security risks,

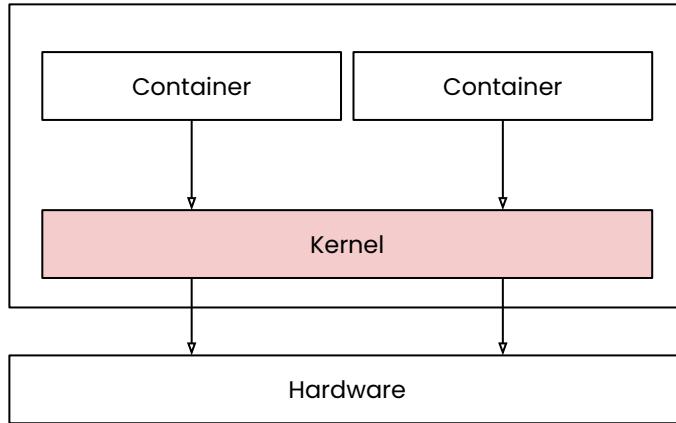


Reference:

- [Linux kernel capabilities](#)

Privilege Escalation (Cont.)

- One of the greatest security risks, **not only in the container area**, is the execution of processes with **too many privileges**,

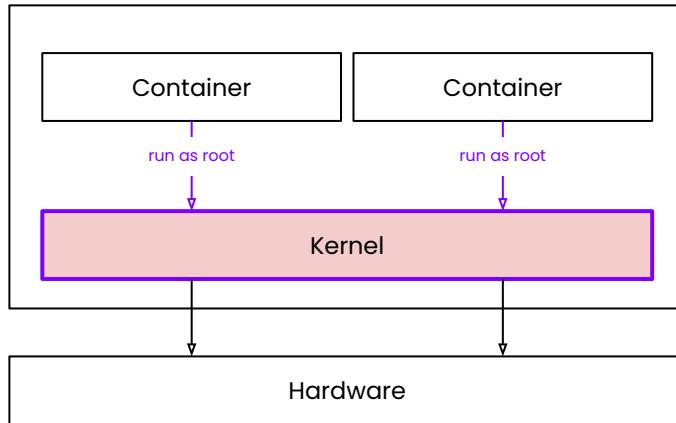


Reference:

- [Linux kernel capabilities](#)

Privilege Escalation (Cont.)

- One of the greatest security risks, not only in the container area, is the execution of processes with **too many privileges**, especially starting processes **as root or administrator**.

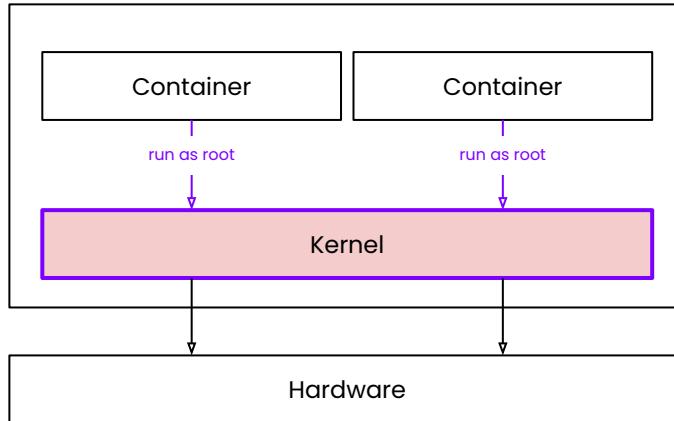


Reference:

- [Linux kernel capabilities](#)

Privilege Escalation (Cont.)

- One of the greatest security risks, not only in the container area, is the execution of processes with **too many privileges**, especially starting processes as root or administrator. Unfortunately, this is a **problem that got ignored a lot in the past** and there are a lot of containers out there that **run as root users**.



Reference:

- [Linux kernel capabilities](#)

Surface Attack

- Container Security -

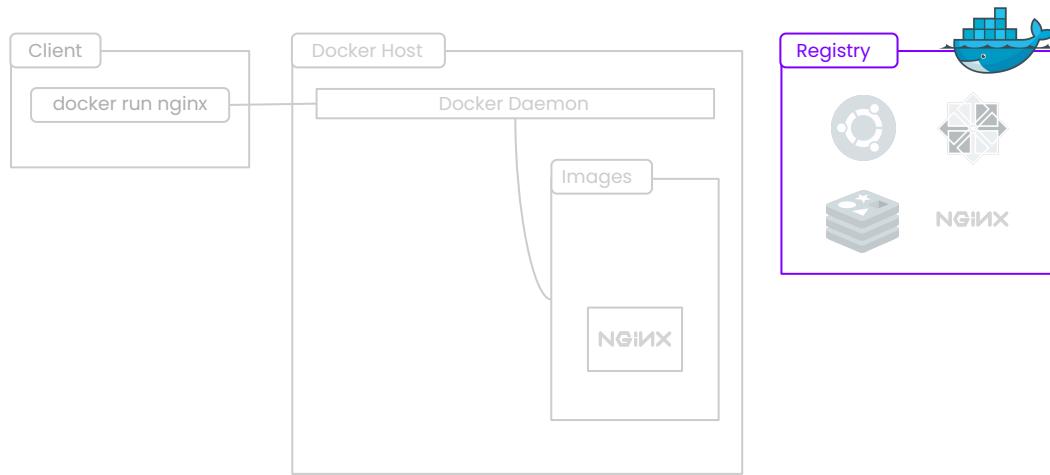
Surface Attack

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ร่องรอยนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

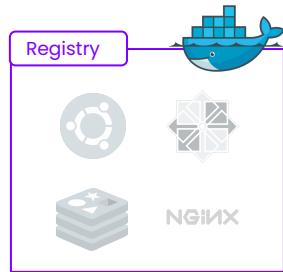
Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).



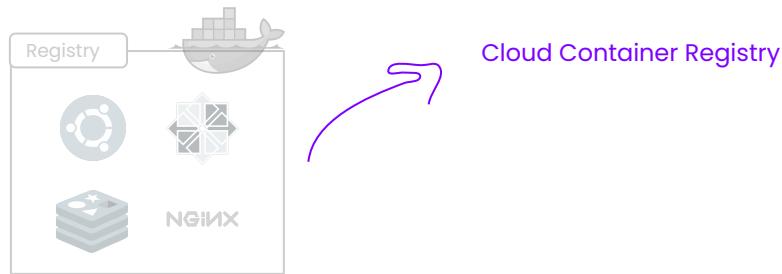
Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).



Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).



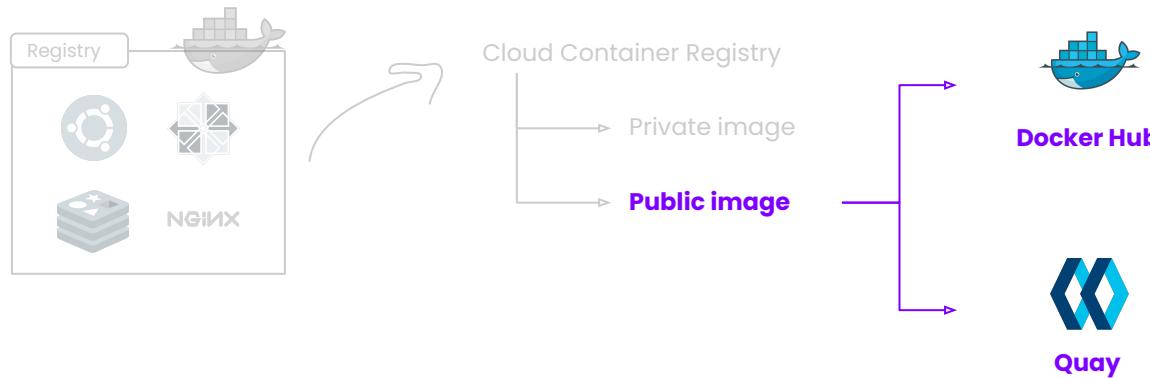
Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).



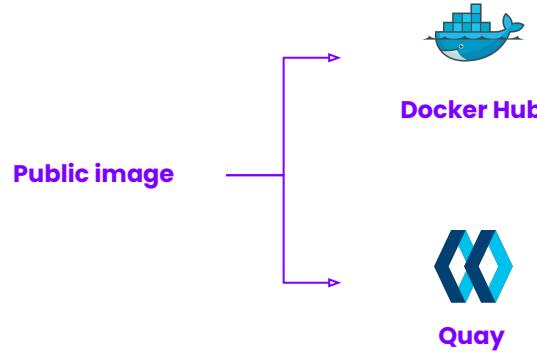
Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).
- Two of the most **popular public image registries** are **Docker Hub** and **Quay** and while it's great that they provide publicly accessible images



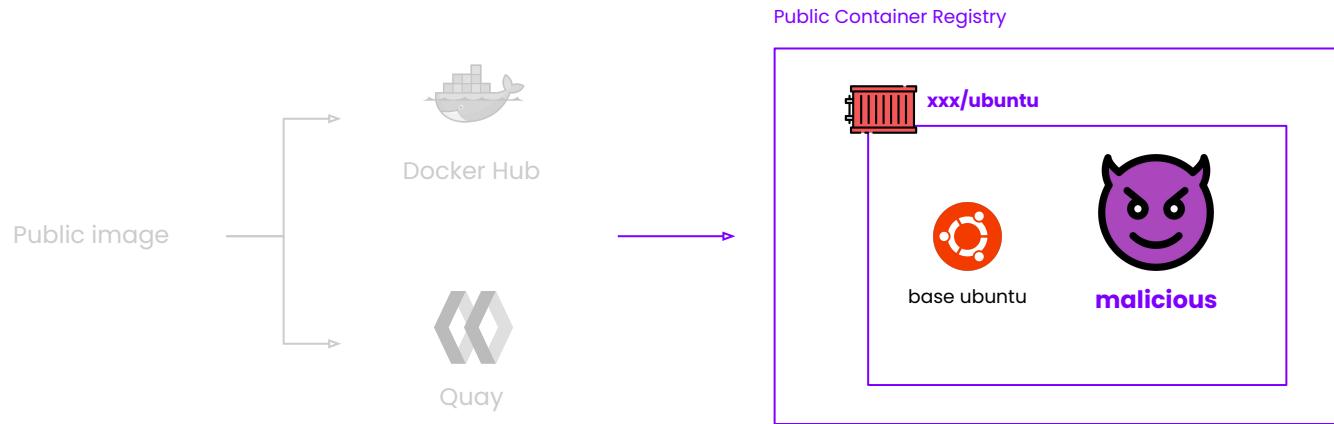
Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).
- Two of the most popular public image registries are **Docker Hub** and **Quay** and while it's great that they provide publicly accessible images, you have to **make sure** that



Surface Attack (Cont.)

- A fairly new attack surface that was introduced with containers is the use of **public images** (cloud container registry).
- Two of the most popular public image registries are **Docker Hub** and **Quay** and while it's great that they provide publicly accessible images, you have to make sure that these images were not modified to include malicious software.



Distroless Image

- 4. Container Security -

Distroless Image

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้รับสุนั�น์เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Distroless Image (Cont.)

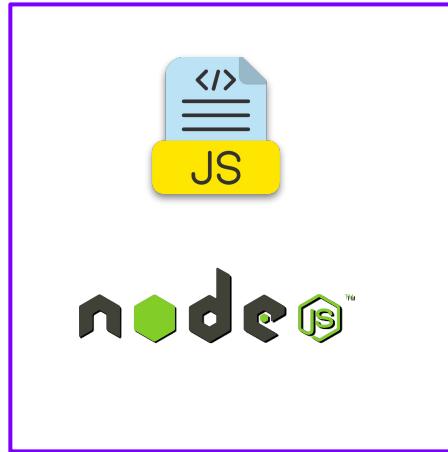
- Contains only **Application Runtime (if required)** and **Application (executable file)** and it does not provide any OS packages even shell commands.

Reference:

- [Distroless Image](#)

Distroless Image (Cont.)

- Contains only **Application Runtime (if required)** and **Application (executable file)** and it does not provide any OS packages even shell commands.



Reference:

- [Distroless Image](#)

Distroless Image (Cont.)

- Contains only **Application Runtime (if required)** and **Application (executable file)** and it does not provide any OS packages even shell commands.

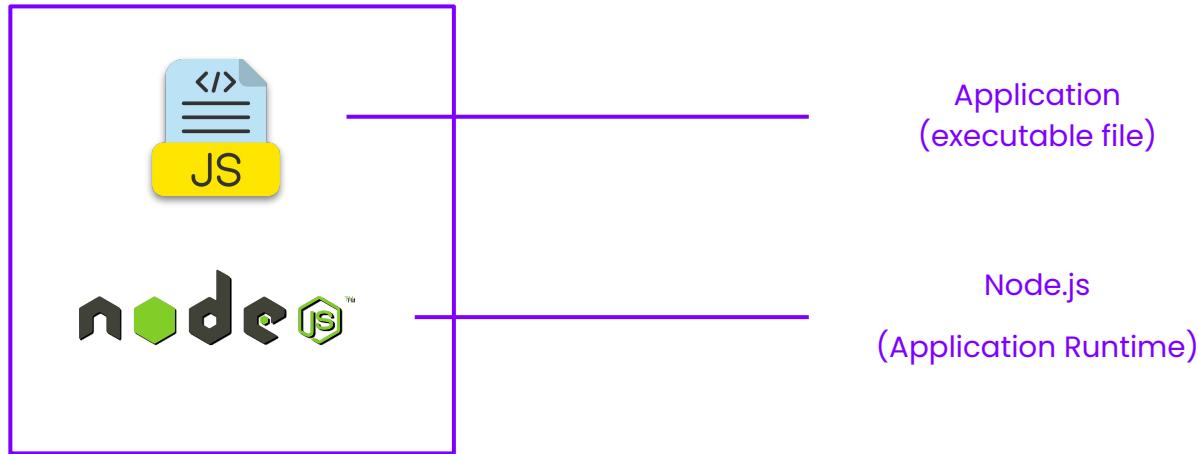


Reference:

- [Distroless Image](#)

Distroless Image (Cont.)

- Contains only **Application Runtime (if required)** and **Application (executable file)** and it does not provide any OS packages even shell commands.



Reference:

- [Distroless Image](#)

Red Hat UBI Micro Image

- 4. Container Security -

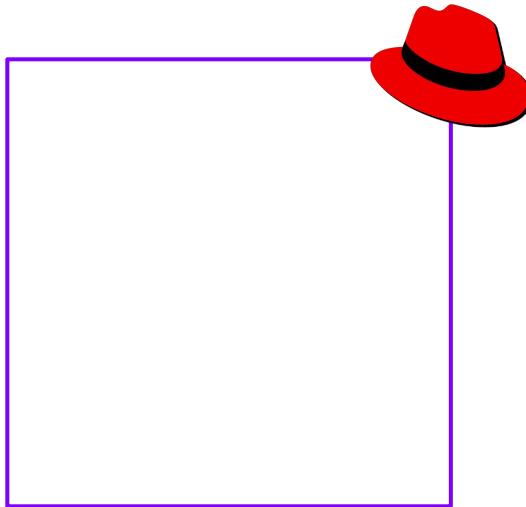
Red Hat UBI Micro Image

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

Red Hat UBI Micro Image (Cont.)

- reduce the attack surface of Linux containers
(FROM registry.access.redhat.com/ubi9/ubi-micro:9.2-15)

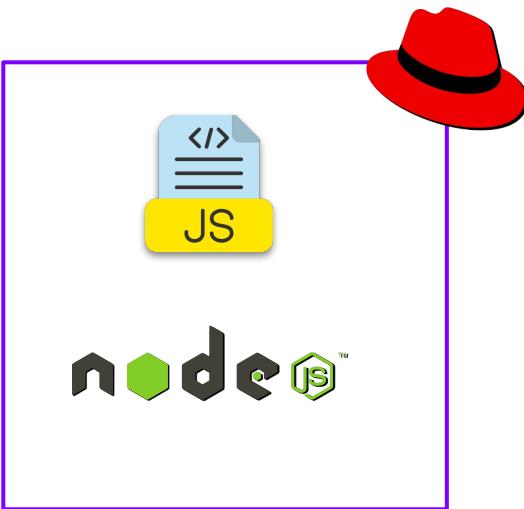


Reference:

- [Red Hat UBI Micro Image](#)

Red Hat UBI Micro Image (Cont.)

- reduce the attack surface of Linux containers
(FROM registry.access.redhat.com/ubi9/ubi-micro:9.2-15)



Home > Containers > Browse Containers > Red Hat Universal Base Image 9 Micro

Scratch image

Red Hat Universal Base Image 9 Micro

Provided by Red Hat

ubiq/ubi-micro

Architecture: amd64 Tag: 9.2-15 Repository structure: Single-stream

9.2-15 latest 9.2

Overview Security Technical Information Packages Dockerfile Get this image

Description

Universal Base Image Micro (UBI Micro) is a stripped down image that uses the package manager on the underlying host to install packages, typically using Buildah, or Multi-stage builds with Podman. This base image is freely redistributable, but Red Hat only supports Red Hat technologies through subscriptions for Red Hat products. This image is maintained by Red Hat and updated regularly.

Documentation

[Understanding the UBI micro images](#)
[Using the UBI micro images](#)

Published 17 days ago

Release category Generally Available

Health index

Size 7.3 MB (24.9 MB uncompressed)

Reference:

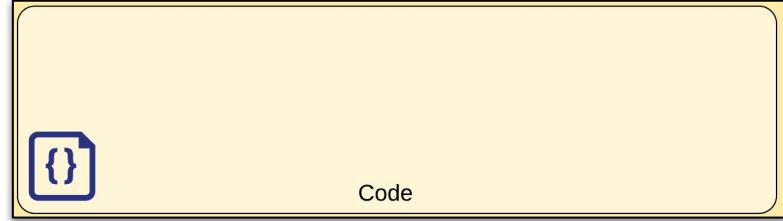
- [Red Hat UBI Micro Image](#)

The 4C's of Cloud Native security

- Container Security -

The 4C's of Cloud Native Security (Cont.)

1. **C**ode

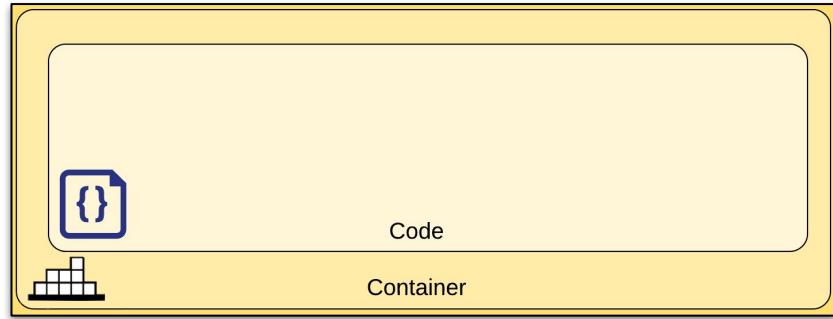


Reference:

- [The 4C's of Cloud Native Security](#)

The 4C's of Cloud Native Security (Cont.)

1. **C**ode
2. **C**ontainer

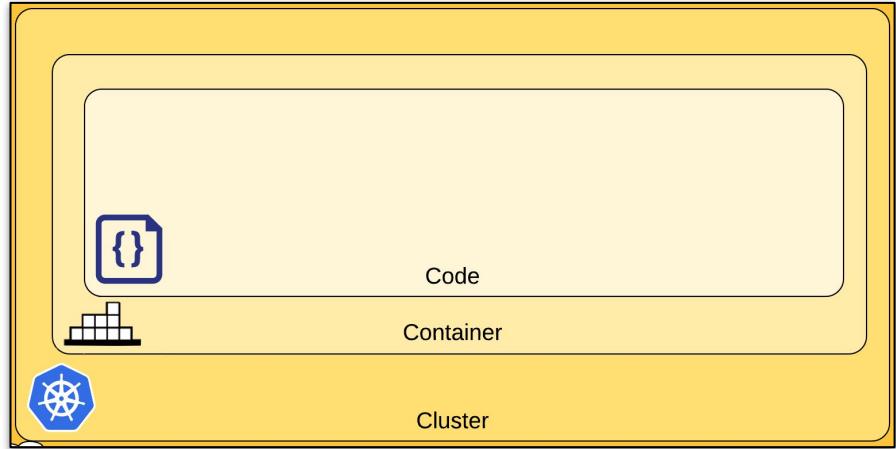


Reference:

- [The 4C's of Cloud Native Security](#)

The 4C's of Cloud Native Security (Cont.)

1. **C**ode
2. **C**ontainer
3. **C**luster
 - Components of the Cluster
 - Components in the cluster
(your application)

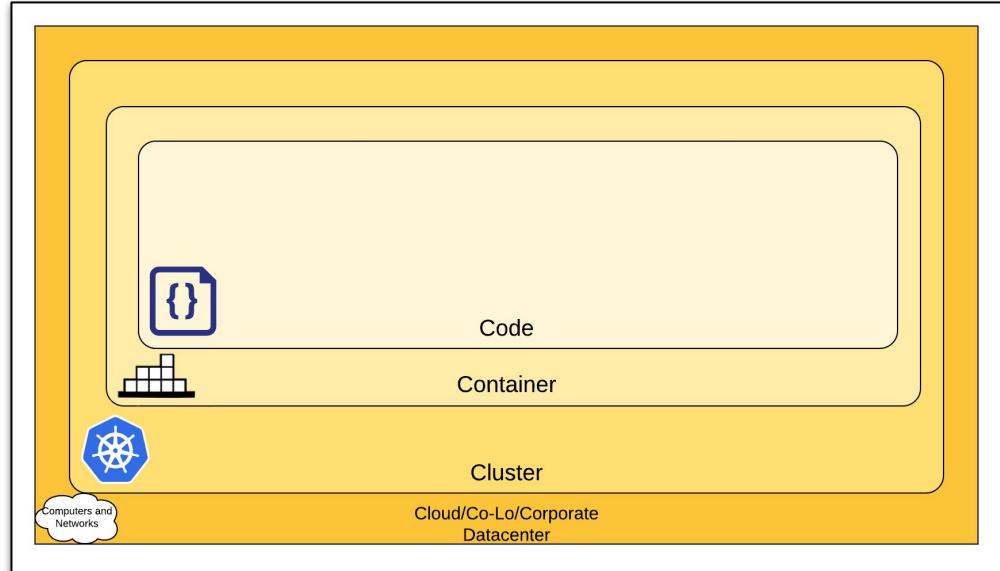


Reference:

- [The 4C's of Cloud Native Security](#)

The 4C's of Cloud Native Security (Cont.)

1. **C**ode
2. **C**ontainer
3. **C**luster
 - Components of the Cluster
 - Components in the cluster
(your application)
4. **C**loud
 - Cloud provider security
 - Infrastructure security



Reference:

- [The 4C's of Cloud Native Security](#)

Multi-Stages

Multi-Stages(Cont.)

- In Node.js, you usually don't need to have node modules if the project is built into a static web application that usually can be run by nginx.

Multi-Stages(Cont.)

- In Node.js, you usually don't need to have node modules if the project is built into a static web application that usually can be run by nginx.

```
FROM node:20.7.0 AS builder
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . ./.
RUN npm run build
```

Multi-Stages(Cont.)

- In Node.js, you usually don't need to have node modules if the project is built into a static web application that usually can be run by nginx.

```
FROM node:20.7.0 AS builder
WORKDIR /app
COPY package.json .
RUN npm install
COPY . ../
RUN npm run build
```



Stage 1: builder (ชื่อ stage ตั้งได้ตามต้องการ)

Multi-Stages(Cont.)

- In Node.js, you usually don't need to have node modules if the project is built into a static web application that usually can be run by nginx.

```
FROM node:20.7.0 AS builder
WORKDIR /app
COPY package.json .
RUN npm install
COPY . ../
RUN npm run build
FROM nginx:1.25.2 as runner
WORKDIR /usr/share/nginx/html
RUN rm -rf ./*
COPY --from=builder /app/build .
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

Multi-Stages(Cont.)

- In Node.js, you usually don't need to have node modules if the project is built into a static web application that usually can be run by nginx.

```
FROM node:20.7.0 AS builder
WORKDIR /app
COPY package.json .
RUN npm install
COPY . ../
RUN npm run build
FROM nginx:1.25.2 as runner
WORKDIR /usr/share/nginx/html
RUN rm -rf ./*
COPY --from=builder /app/build .
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

→ Stage 1: builder (ชื่อ stage ตั้งได้ตามต้องการ)

→ Stage 2: runner (ชื่อ stage ตั้งได้ตามต้องการ)

Single Stage vs Multi-stages

Single Stage vs Multi-stages

Reference:

- [Multi-Stage](#)
- [Multi-Stage build](#)

Single Stage vs Multi-stages

Image Size

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-normal	latest	e3ae63525098	About a minute ago	275MB
docker-multistages	latest	eb3cde029044	About a minute ago	133MB

Reference:

- [Multi-Stage](#)
- [Multi-Stage build](#)

Single Stage vs Multi-stages

Image Size

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-normal	latest	e3ae63525098	About a minute ago	275MB
docker-multistages	latest	eb3cde029044	About a minute ago	133MB

- From the picture above, you would see that Docker Building **Multi-stages Strategy** would help you **reduce and optimize** the size of Docker image compared to Docker Building Normal Strategy.

Reference:

- [Multi-Stage](#)
- [Multi-Stage build](#)

Activity Time

- With Paper Base -

Output what you learned in **Practical Action**

Hands-on Workshop:

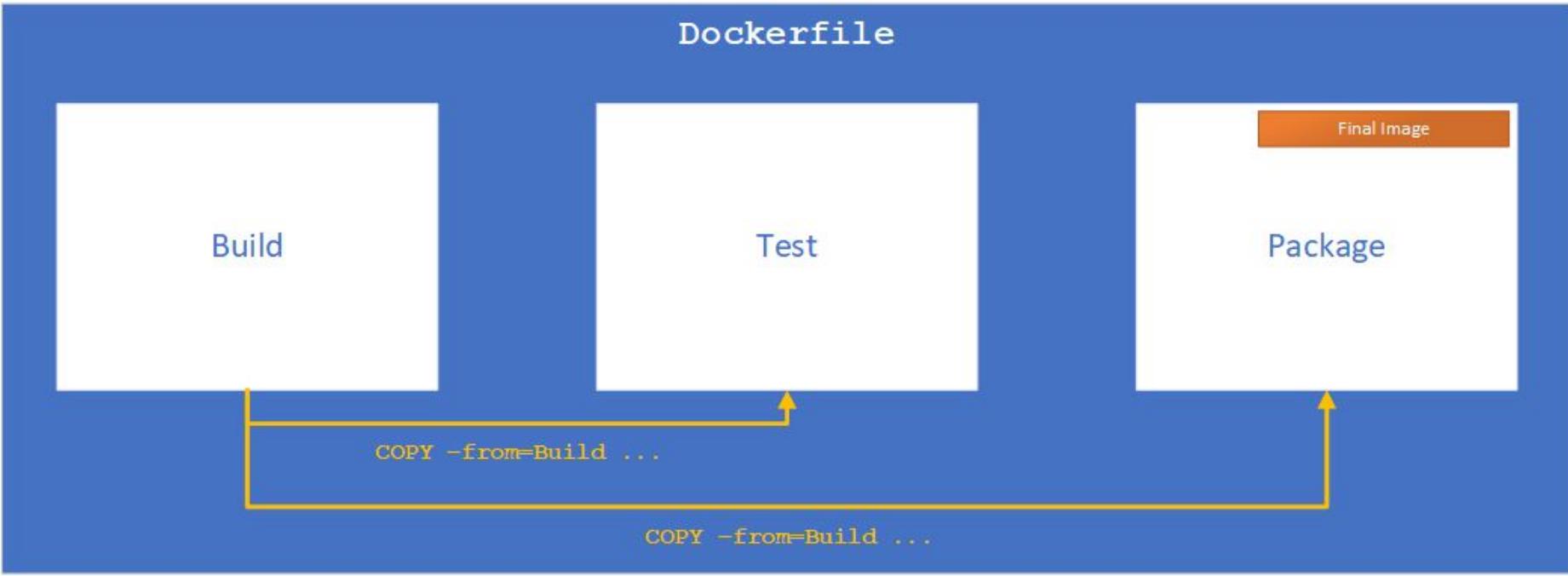
Multi-stage builds

Use multi-stage builds

- 'cherry pick' your artifacts without inheriting the vulnerabilities from the base images on which they rely.

Reference:

- [Building best practices](#)



Reference Pictures:

- [Docker Tutorials: Image – Optimize faster builds and smaller docker images using multistaging build?](#)

Explore Multi-stage builds demo 1

```
FROM node:20.9-slim AS build
WORKDIR /usr/src/app
COPY package* ./
RUN yarn
COPY . .
RUN yarn run build

FROM node:20.9-alpine
WORKDIR /usr/src/app
COPY package* ./
RUN yarn --prod
COPY --from=build /usr/src/app/dist ./
EXPOSE 3000
ENV NODE_ENV production
ENTRYPOINT [ "yarn", "run" ]
CMD [ "start:prod" ]
```

Explore Multi-stage builds demo 2

```
FROM node:20.9-slim AS build
WORKDIR /app
COPY package* ./
RUN npm i
COPY ..
RUN npm run build

FROM nginx:stable-alpine
COPY /nginx/nginx.conf /etc/nginx/conf.d/default.conf
COPY bin/docker-entrypoint.sh bin/generate_config_js.sh /bin/
RUN chmod u+x bin/docker-entrypoint.sh bin/generate_config_js.sh
COPY --from=build /app/build /usr/share/nginx/html
ENTRYPOINT [ "/bin/docker-entrypoint.sh" ]
```

Explore Multi-stage builds demo 3

```
FROM node:18-alpine AS build
WORKDIR /app
RUN npm i -g pnpm husky
COPY .npmrc package.json pnpm-lock.yaml ./
RUN pnpm i --frozen-lockfile
COPY ..
RUN pnpm build
RUN pnpm prune --prod

FROM gcr.io/distroless/nodejs18-debian12:nonroot
WORKDIR /app
USER nonroot
COPY --from=build --chown=nonroot:nonroot /app/node_modules ./node_modules
COPY --from=build --chown=nonroot:nonroot /app/dist/ ./dist/
EXPOSE 3000
ENV NODE_PORT 3000
ENV NODE_ENV production
CMD [ "dist/main.js" ]
```

Rebuilding images

- Each container should have only one responsibility.
- Containers should be immutable, lightweight, and fast.
- Don't store data in your containers. Use a shared data store instead.
- Containers should be easy to destroy and rebuild.
- Use a small base image (such as Linux Alpine). Smaller images are easier to distribute.
- Avoid installing unnecessary packages. This keeps the image clean and safe.
- Avoid cache hits when building.
- Auto-scan your image before deploying to avoid pushing vulnerable containers to production.
- Analyze your images daily both during development and production for vulnerabilities. Based on that, automate the rebuilding of images if necessary.

```
$ docker build --no-cache -t myImage:myTag myPath/
```

Reference:

- [Building best practices](#)

Check your image for vulnerabilities

- Docker Hub supports an automatic [vulnerability scanning](#) feature (Requires a [Docker subscription](#).)
- Docker Hub also supports an early-access [advanced image analysis](#) feature
- CLI docker scout
- Docker Desktop has a detailed image view for images in your local image store, that visualizes all of the known vulnerabilities affecting an image.

Reference:

- [Building best practices](#)

Docker scout

1. Run docker scout local image

Reference:

- [Docker Scout](#)

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Docker Security Cheat Sheet (1)

1. Keep Host and Docker up to date
2. Do not expose the Docker daemon socket (even to the containers)
3. Set a user
4. Limit capabilities (Grant only specific capabilities, needed by a container)
5. Add `--no-new-privileges` flag
6. Disable inter-container communication (`--icc=false`)
7. Use Linux Security Module (seccomp, AppArmor, or SELinux)

Reference:

- [Docker Security Cheat Sheet](#)

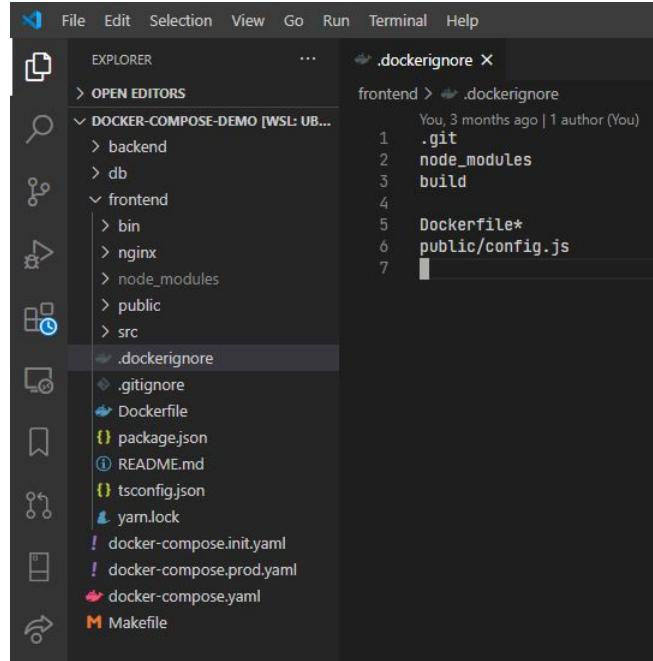
Docker Security Cheat Sheet (2)

8. Limit resources (memory, CPU, file descriptors, processes, restarts)
9. Set filesystem and volumes to read-only
10. Use static analysis tools
11. Set the logging level to at least INFO
12. Lint the Dockerfile at build time
13. Run Docker in root-less mode

Reference:

- [Docker Security Cheat Sheet](#)

Docker Ignore



Reference:

- [.dockerignore file](#)

Let's get back to Dockerfile

FROM instruction (Cont.)

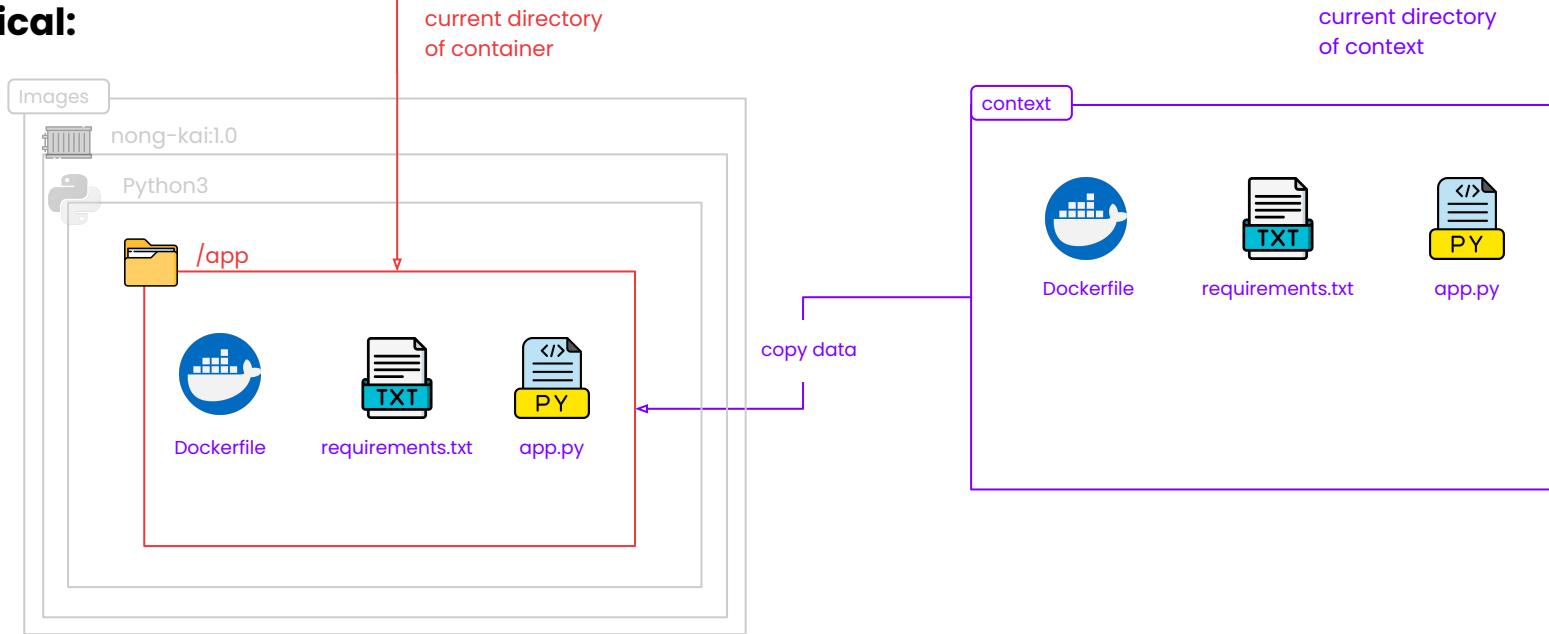
Real World:

```
COPY .  
...  
...
```

Command:

```
$ docker build -t nong-kai:1.0 .
```

Logical:



FROM instruction (Cont.)

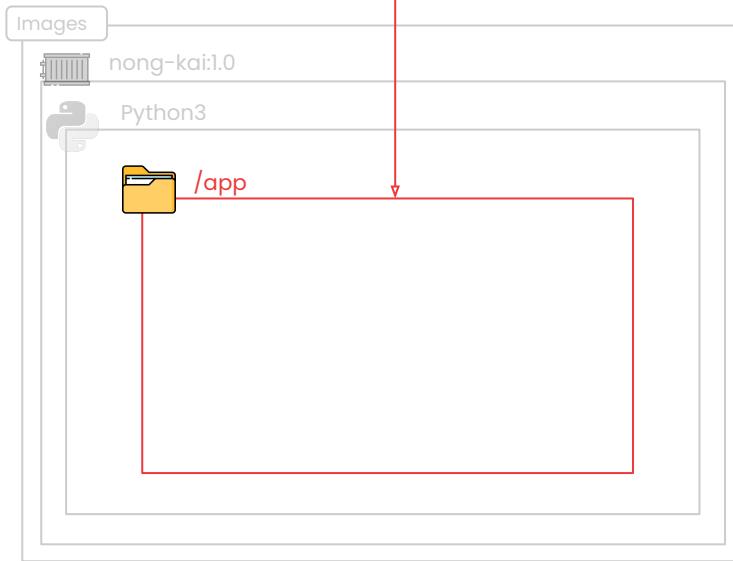
Real World:

```
COPY .  
...
```

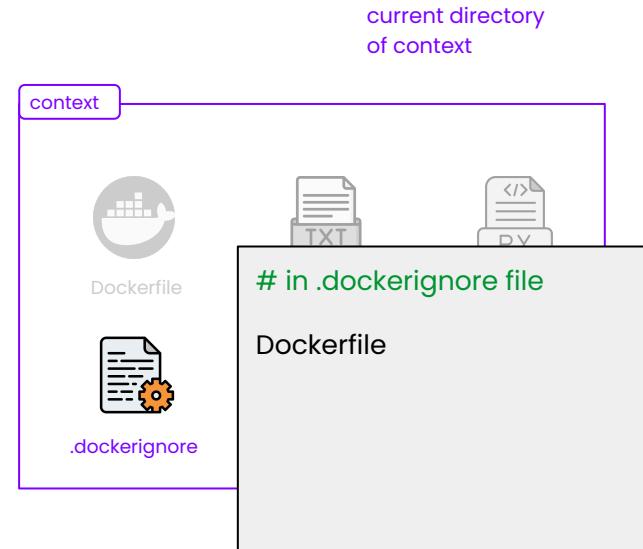
Command:

```
$ docker build -t nong-kai:1.0 .
```

Logical:



current directory
of container



current directory
of context

FROM instruction (Cont.)

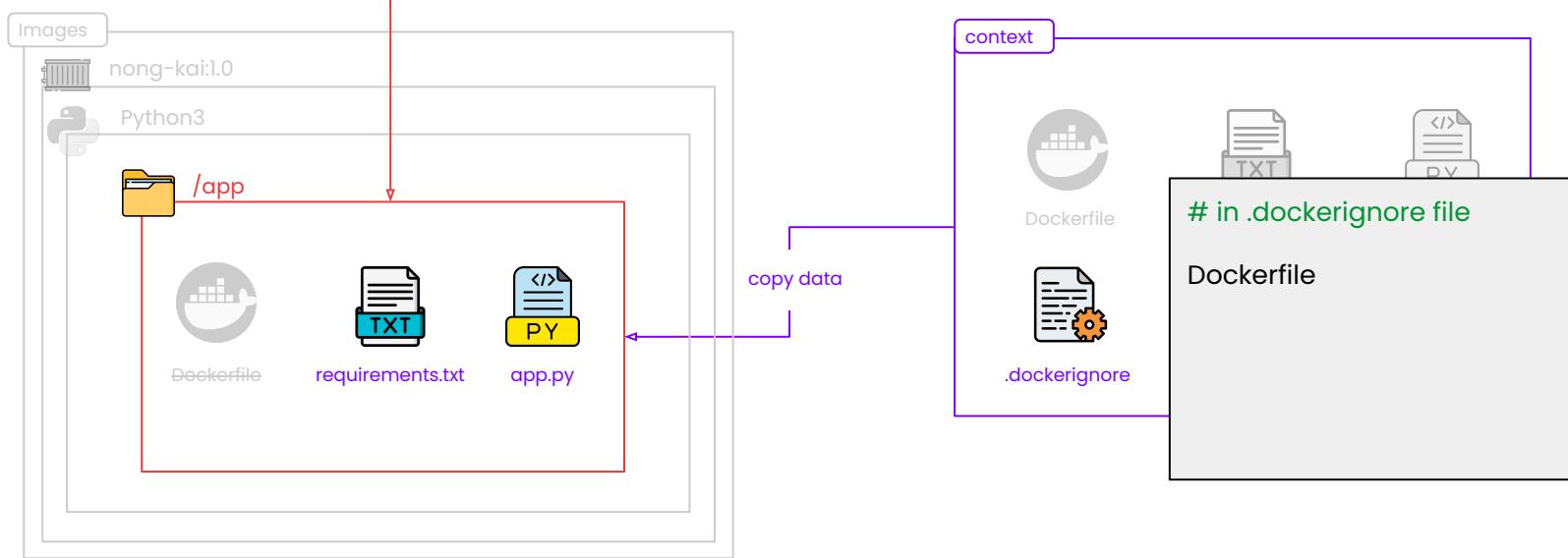
Real World:

```
COPY .  
...  
...
```

Command:

```
$ docker build -t nong-kai:1.0 .
```

Logical:



Docker Resources

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Docker Runtime Metrics

1. Display a live stream of container(s) resource usage statistics

Reference:

- [Runtime metrics](#)

Limit Resources

1. Open new terminal session
2. Run container with limit resources
3. Monitor resource usage statistics
4. Exit

Activity Time

- With Paper Base -

Output what you learned in **Practical Action**

Hands-on Workshop:

Docker Resources

Step back to see

Jumpbox®

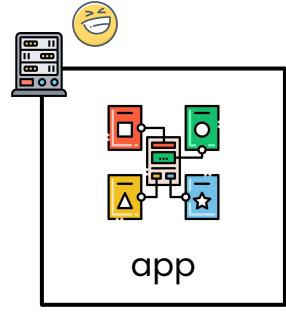
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ร่องรอยนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

Step back to see
the Big Picture

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ร่องรอยนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

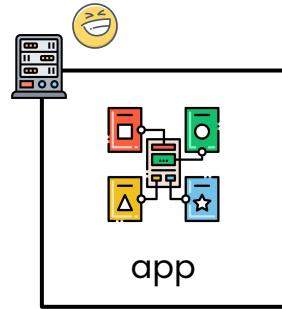
How to manage a lot of traffic



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเนต จะถูกดำเนินคดีตามกฎหมาย



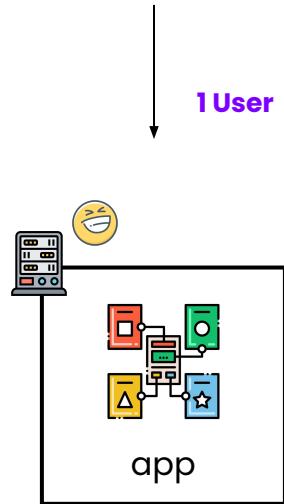
User

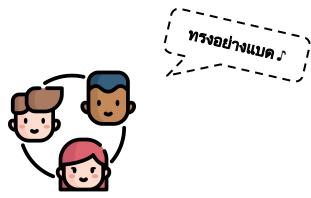




User

1 User

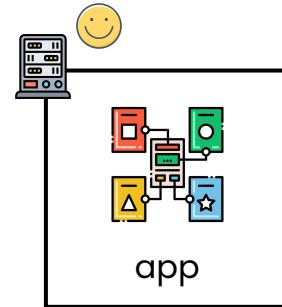
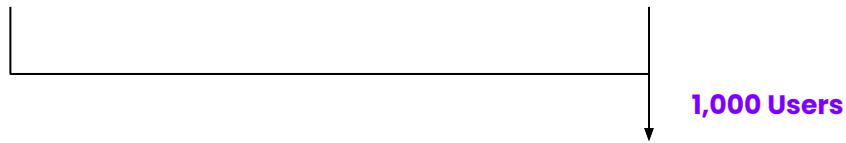




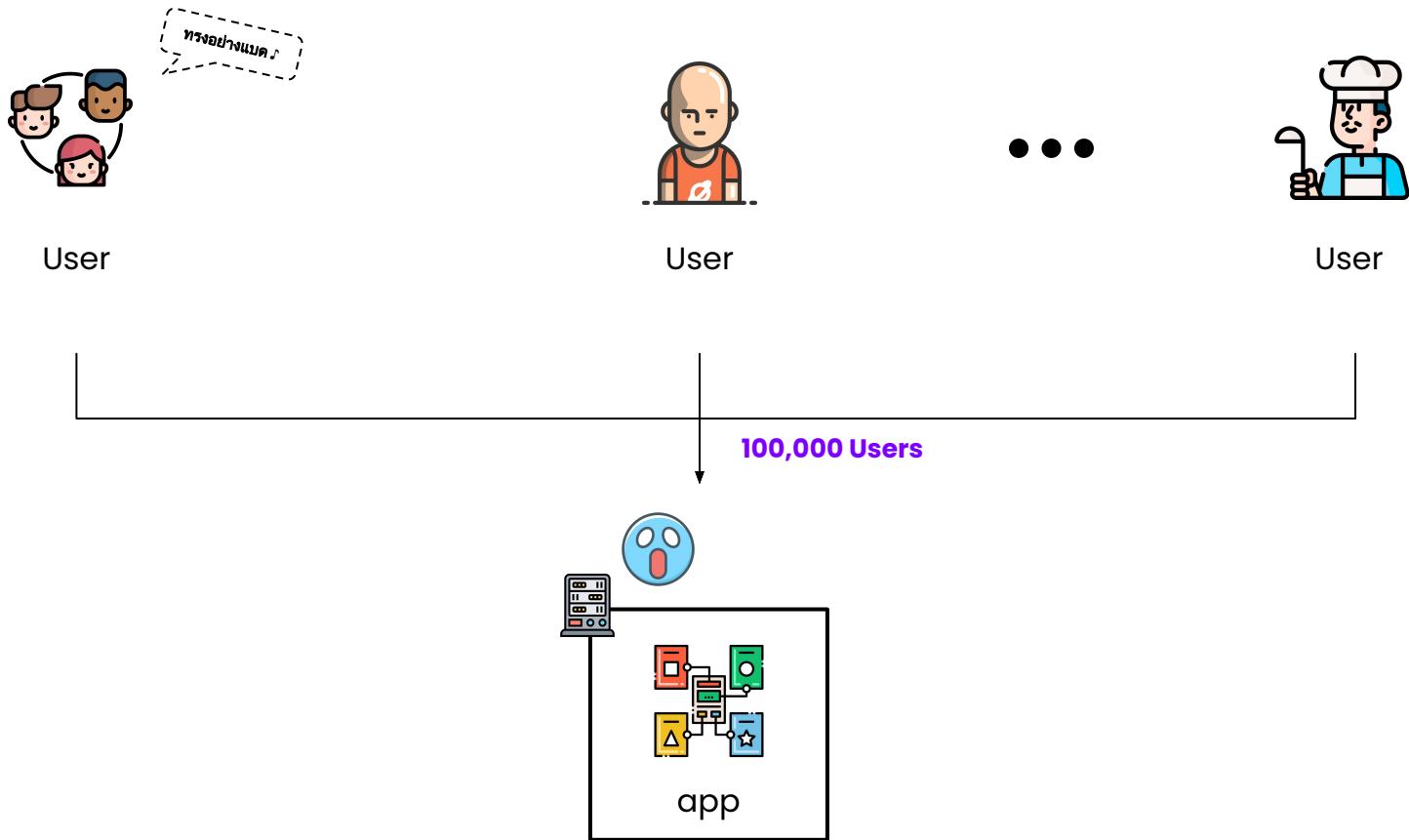
User



User

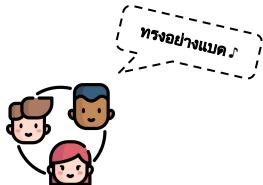


เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคือความหมาย

and now Load Balancer comes in



User

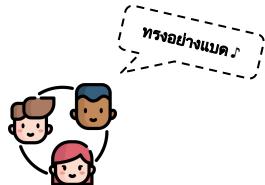


User

• • •



User



User

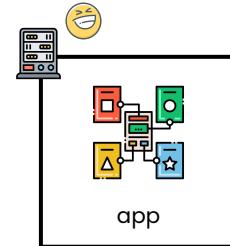


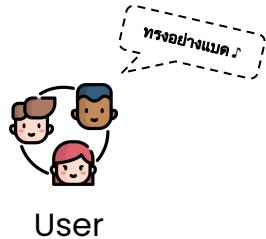
User

• • •

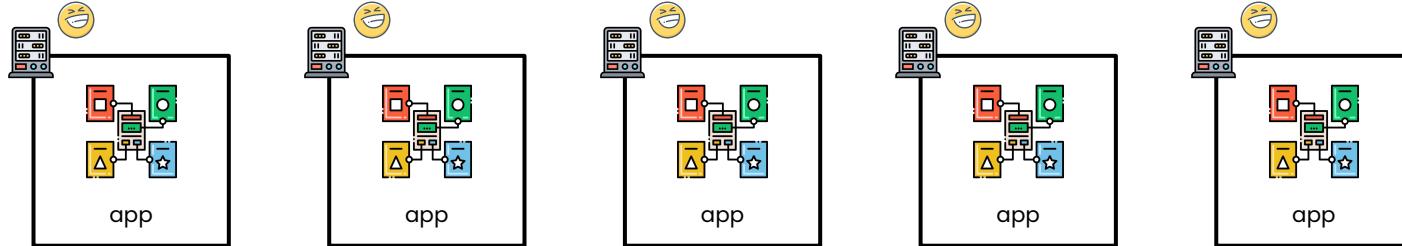


User





• • •



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ด้านบุคคลทั่วไป และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคือความภูมาย

Jumpbox®

กรุณาปางแม่ค้า



User



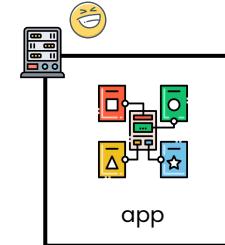
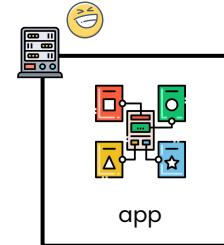
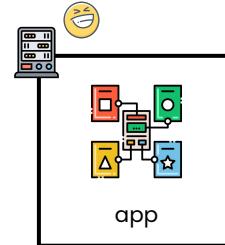
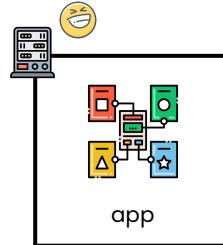
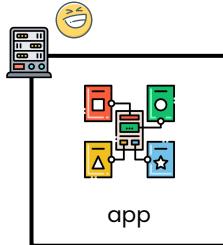
User

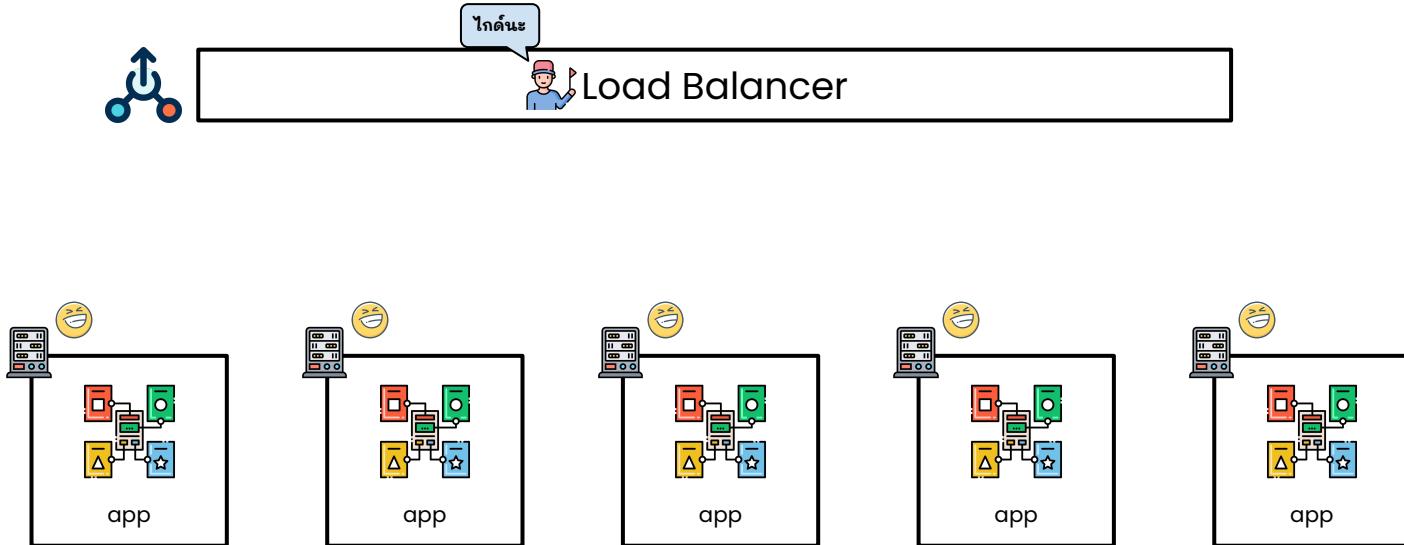
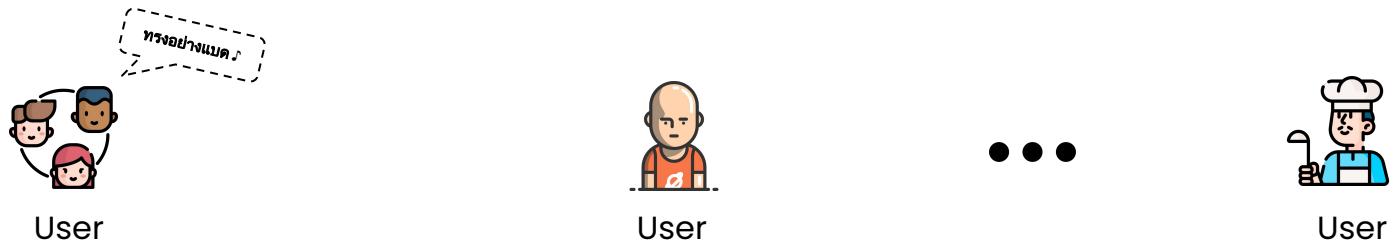


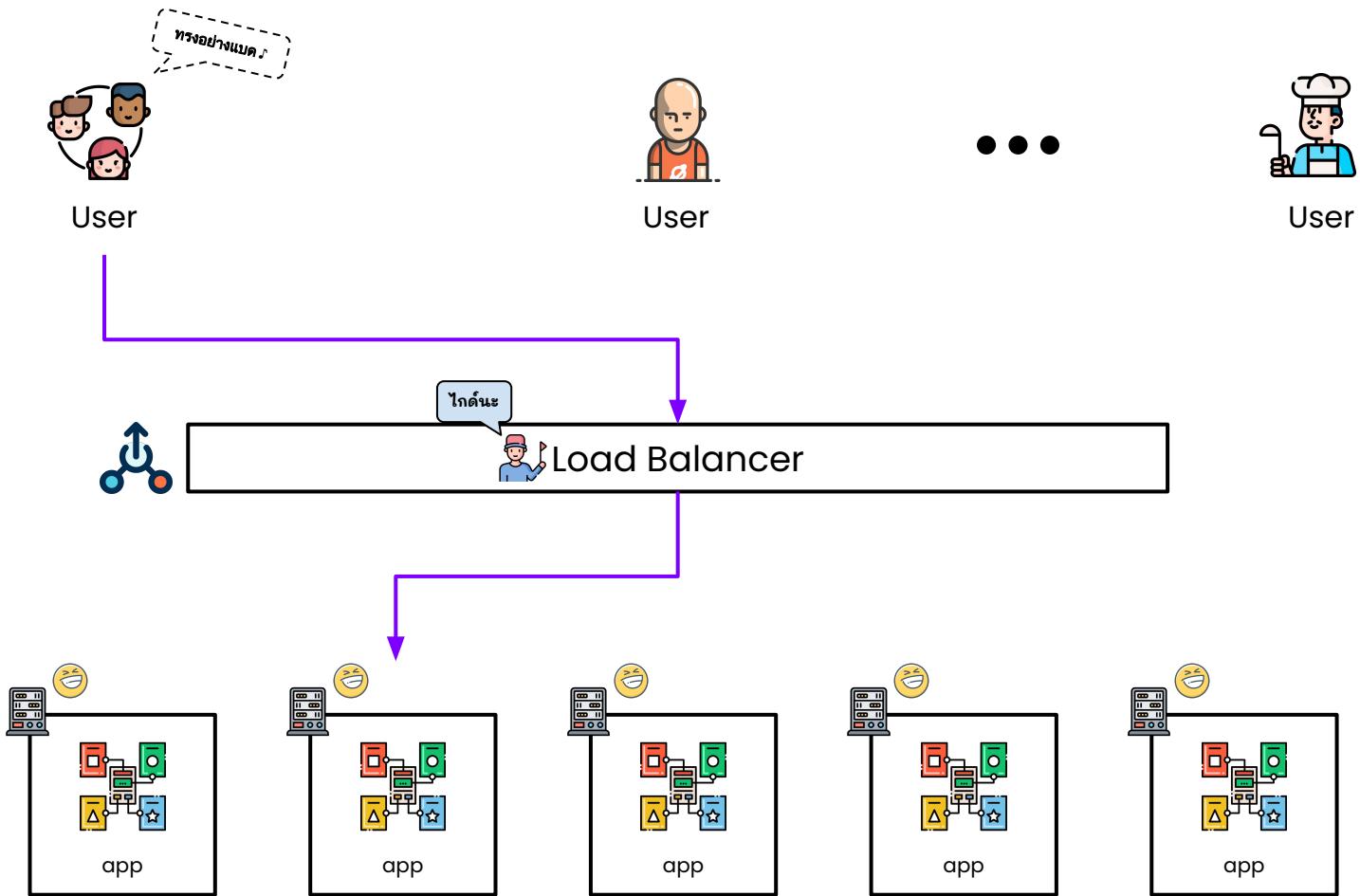
User

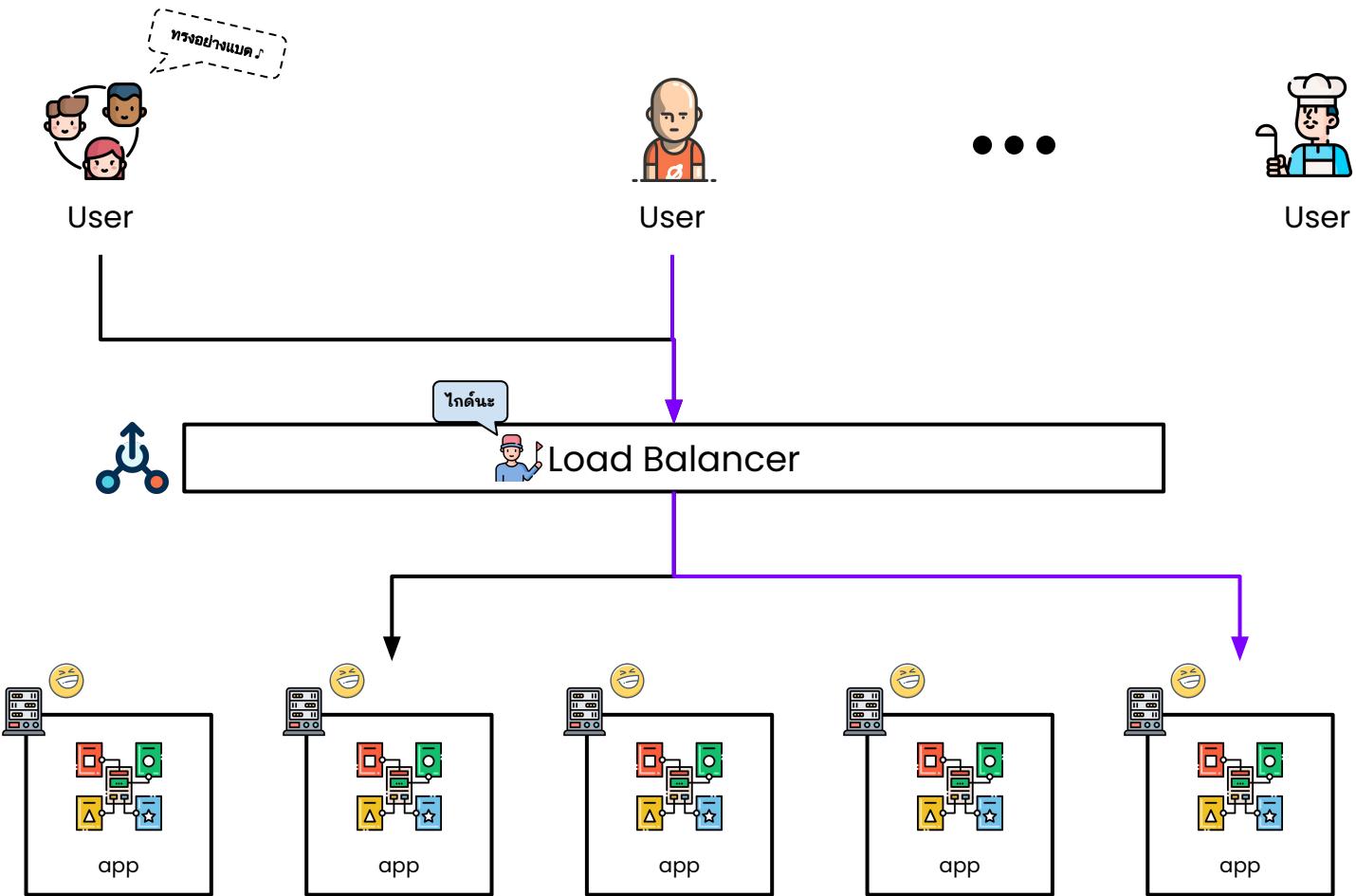


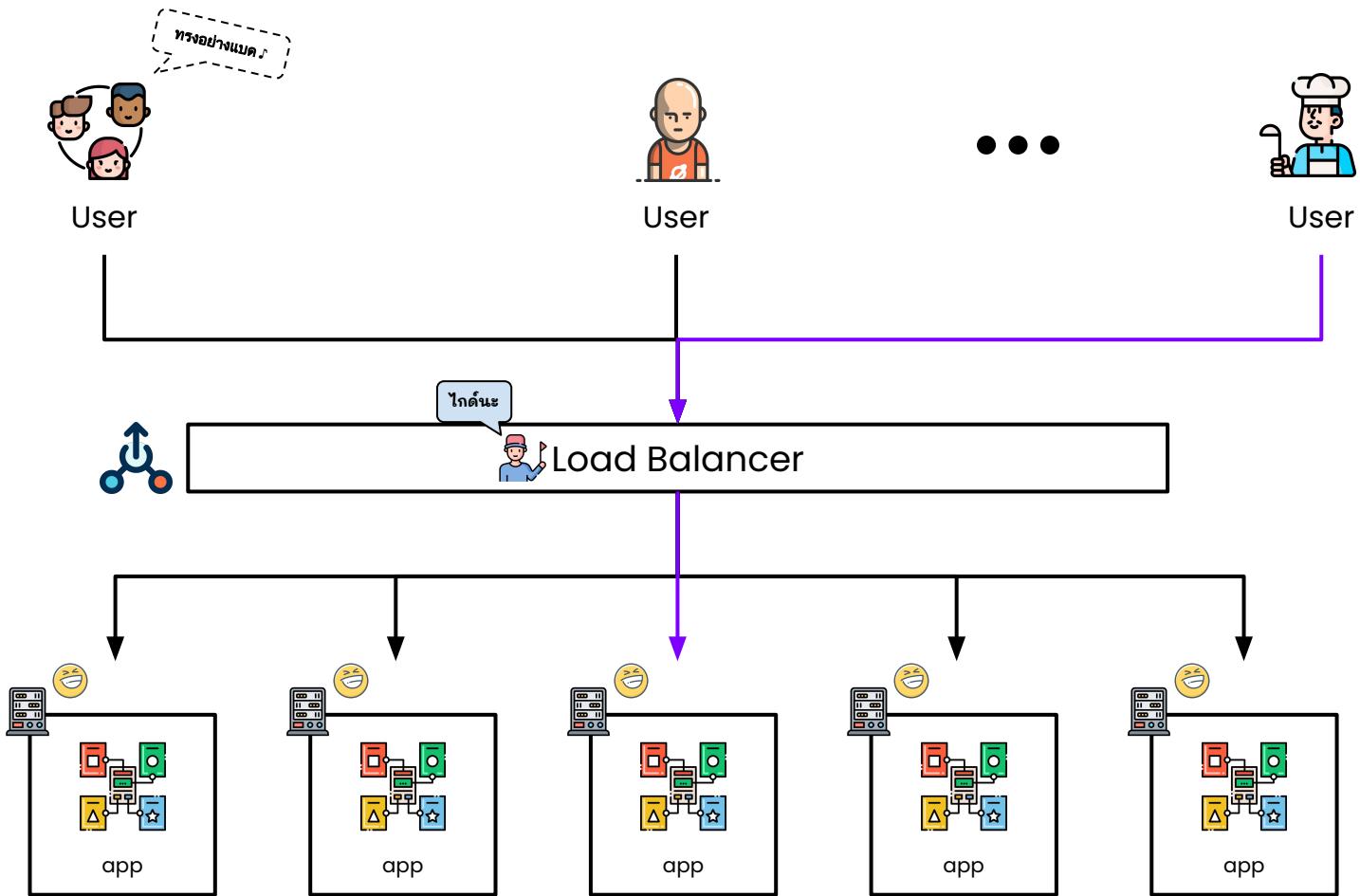
Load Balancer

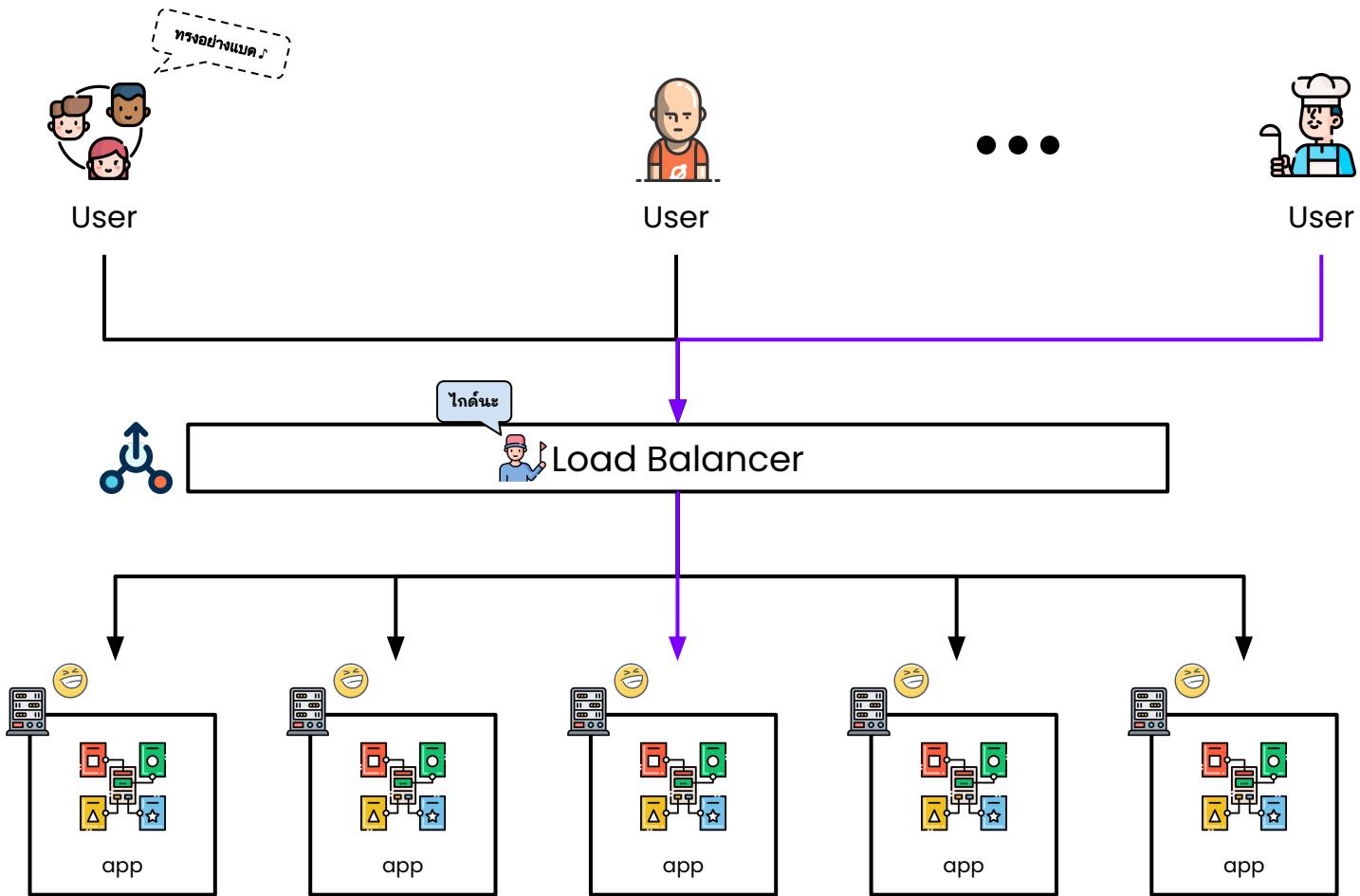












Hidden Problem?

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ร่องรอยนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

Hidden Problem?

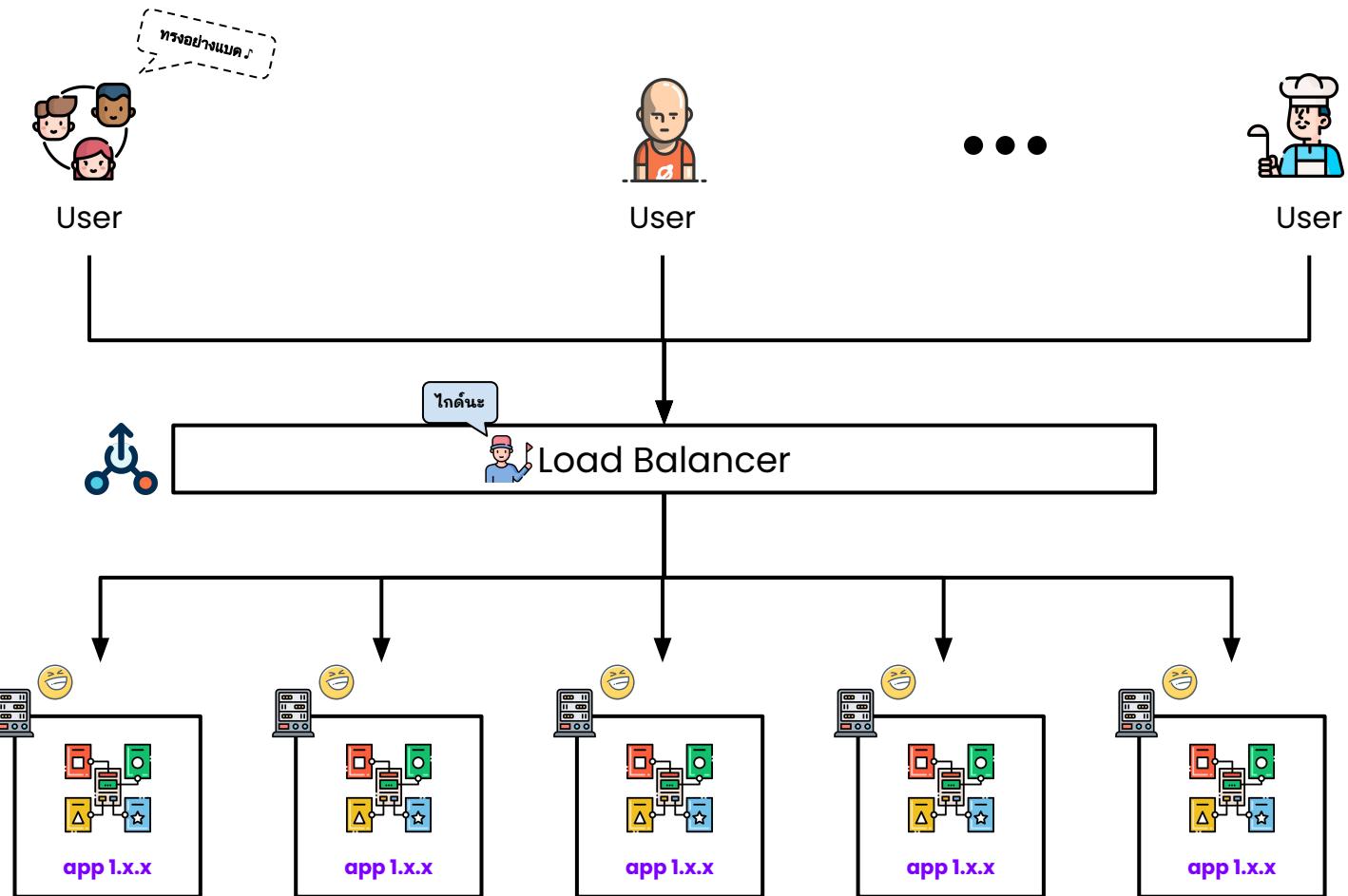
- Library version ไม่ตรงกัน

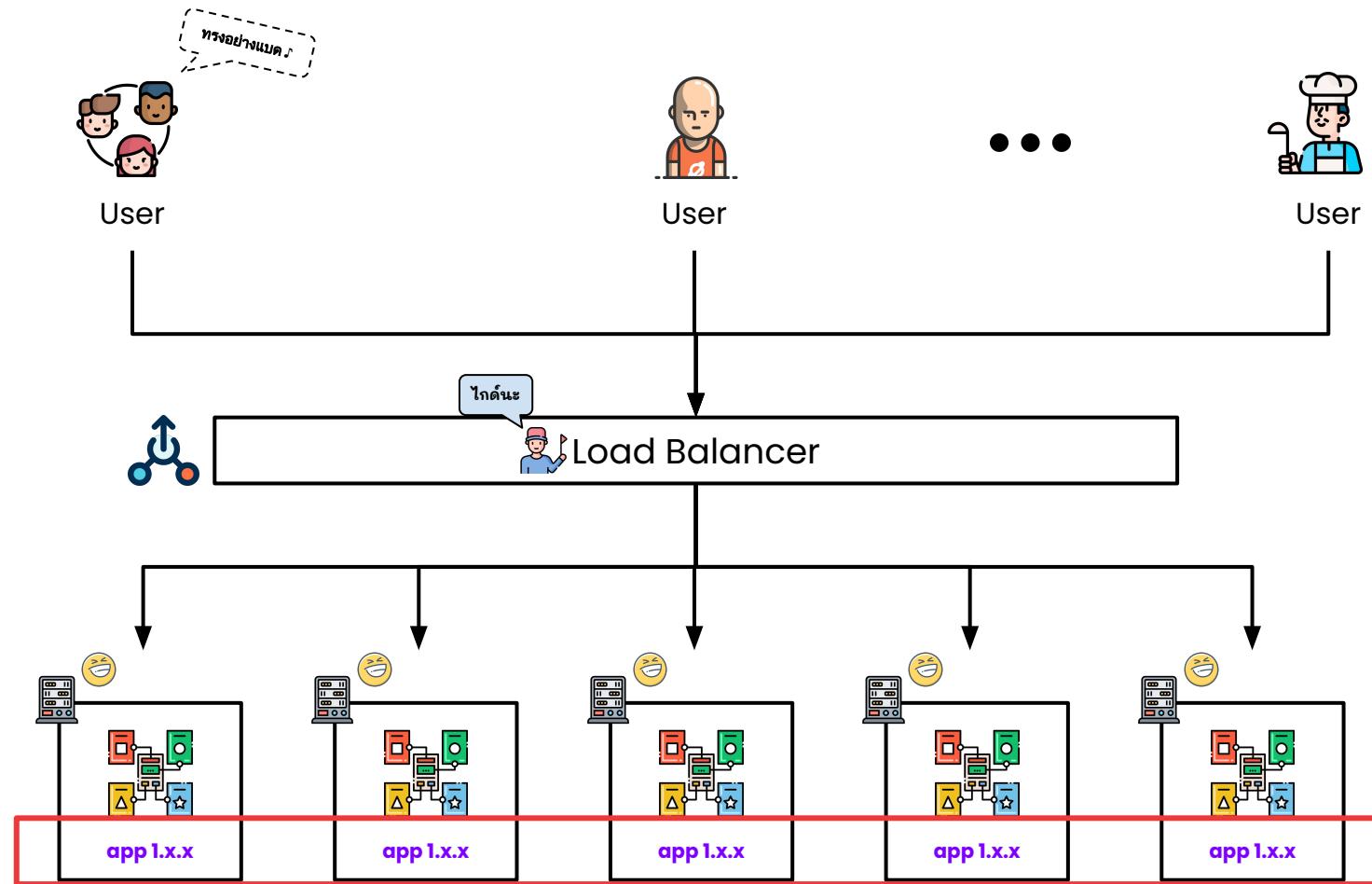
Hidden Problem?

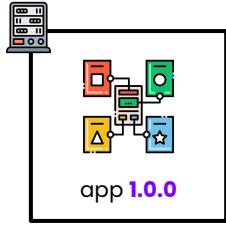
- Library version ไม่ตรงกัน
- Application version ไม่ตรงกัน

Hidden Problem?

- Library version ไม่ตรงกัน
- Application version ไม่ตรงกัน
- Application Runtime version ไม่ตรงกัน

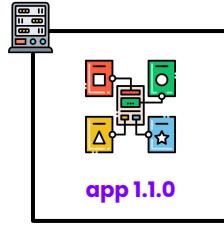
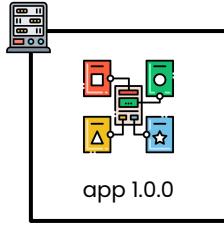


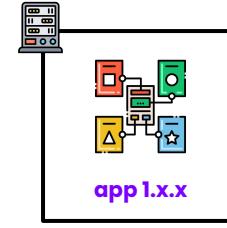
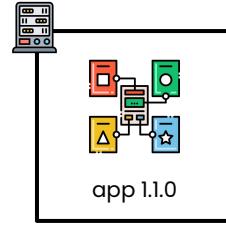
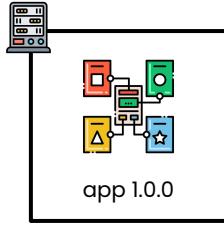




เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®





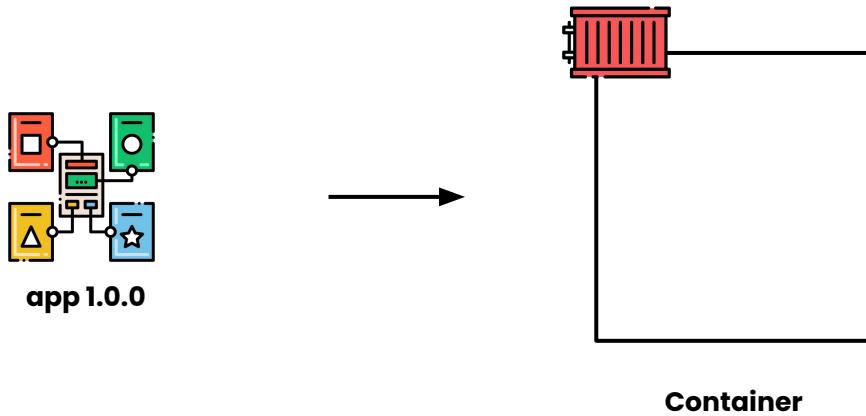
and now Container comes in

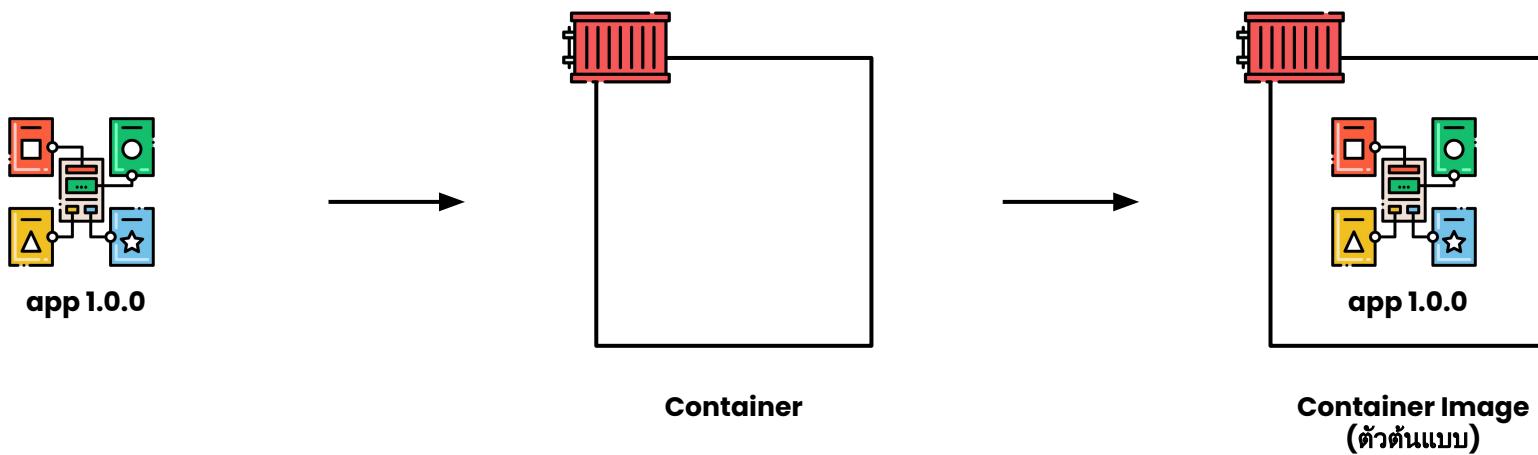


app 1.0.0

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

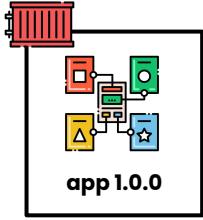




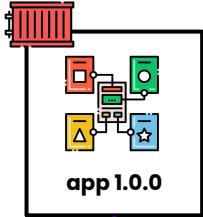
Scaling app with Container

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย



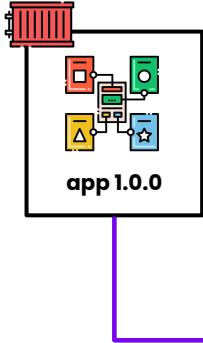
Container Image



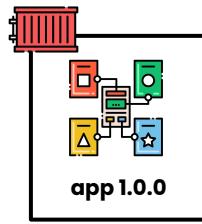
Container Image



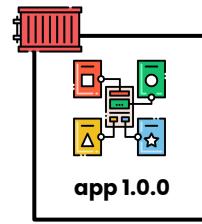
Container Instance



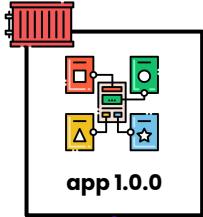
Container Image



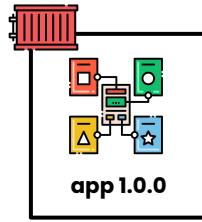
Container Instance



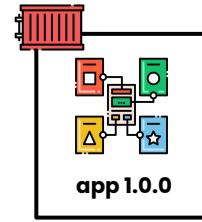
Container Instance



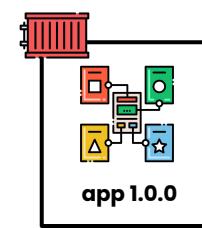
Container Image



Container Instance



Container Instance



Container Instance

Library version ไม่ตรงกัน

Application version ไม่ตรงกัน

Application Runtime version ไม่ตรงกัน



~~Library version~~ ไม่ตรงกัน

Application version ไม่ตรงกัน

Application Runtime version ไม่ตรงกัน



~~Library version~~ ไม่ตรงกัน

~~Application version~~ ไม่ตรงกัน

Application Runtime version ไม่ตรงกัน



~~Library version~~ ไม่ตรงกัน

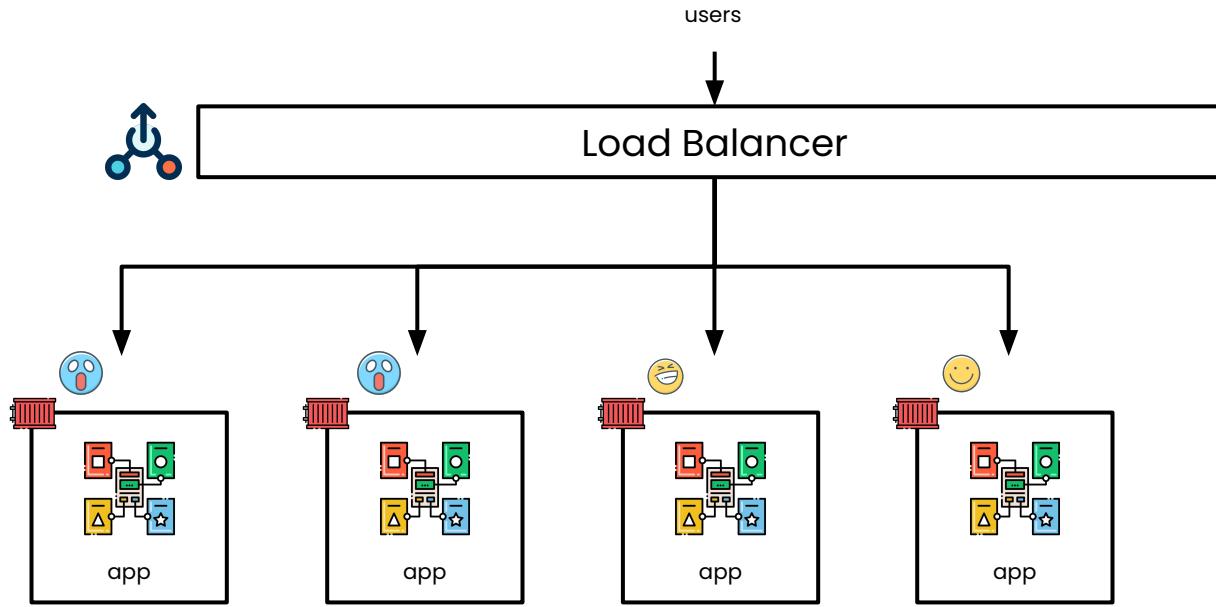


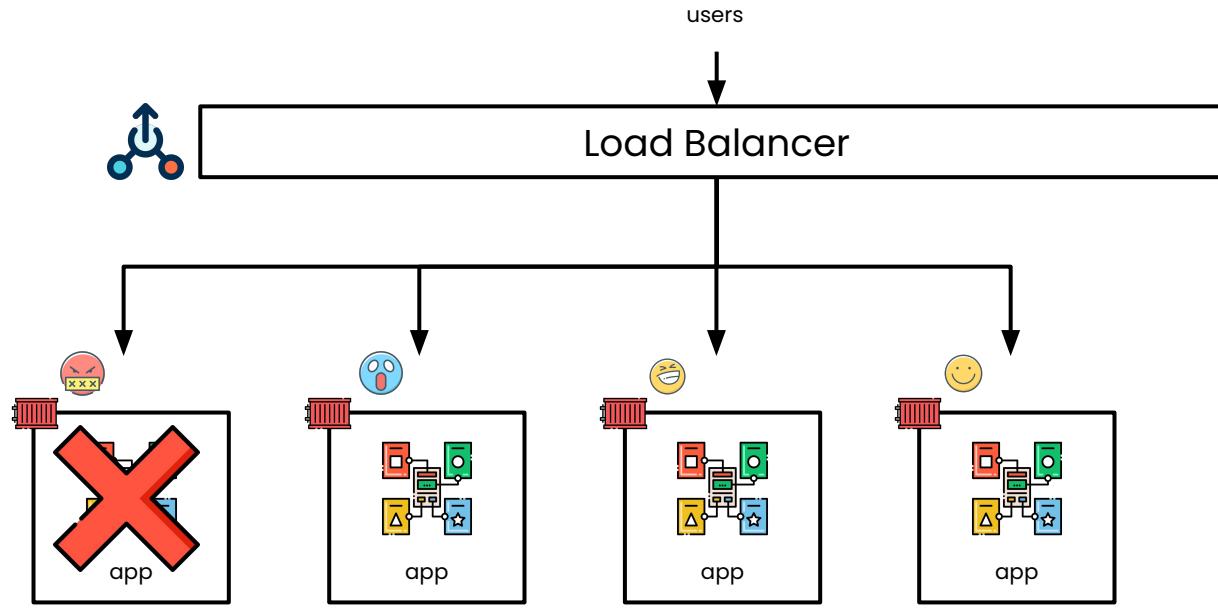
~~Application version~~ ไม่ตรงกัน

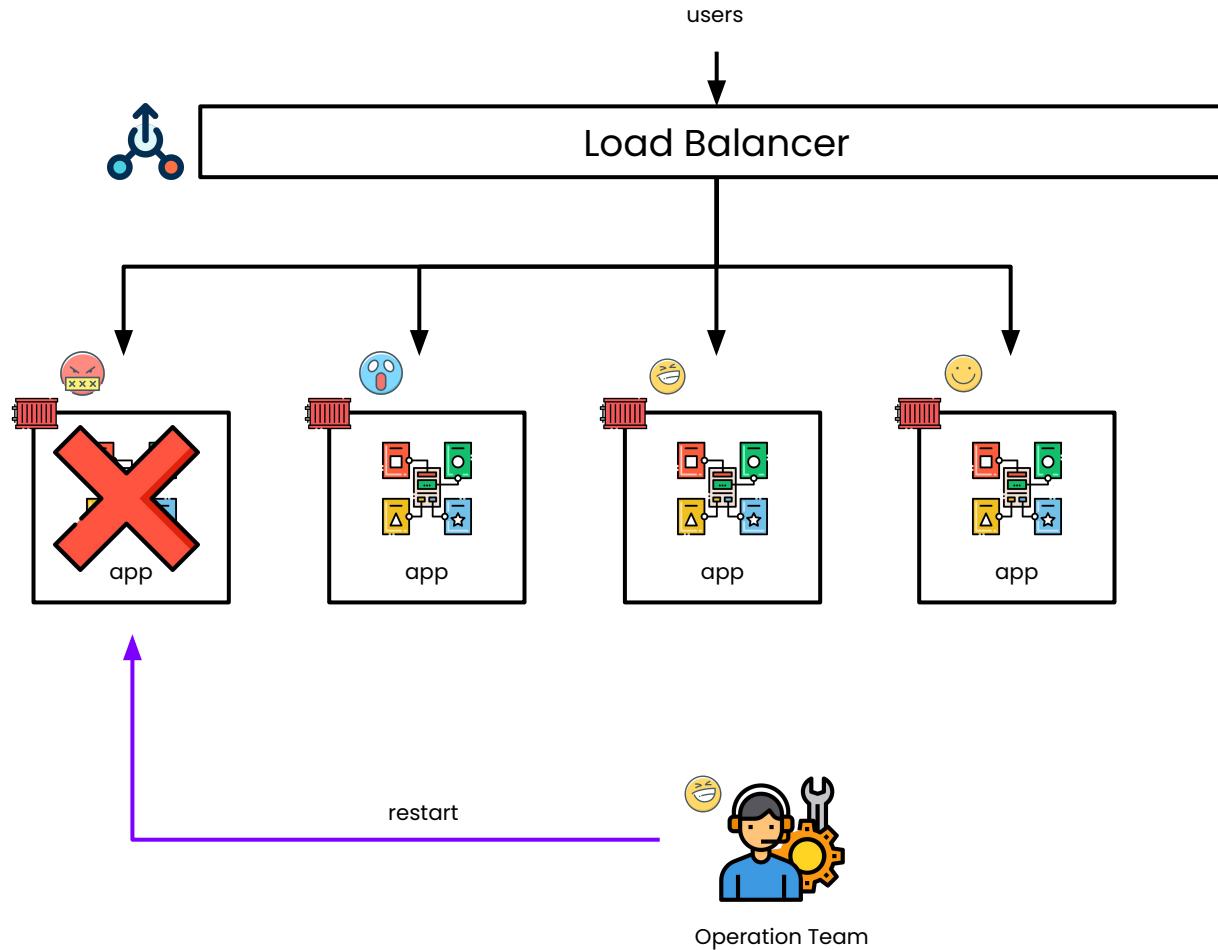


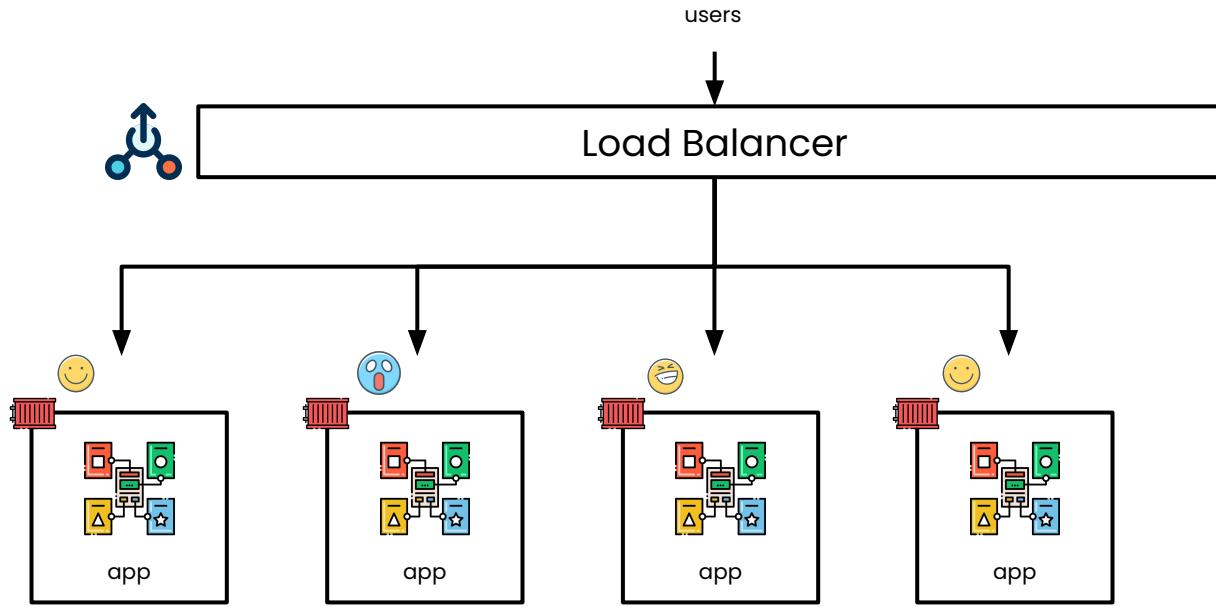
~~Application Runtime version~~ ไม่ตรงกัน

Come back to the **real world use case**

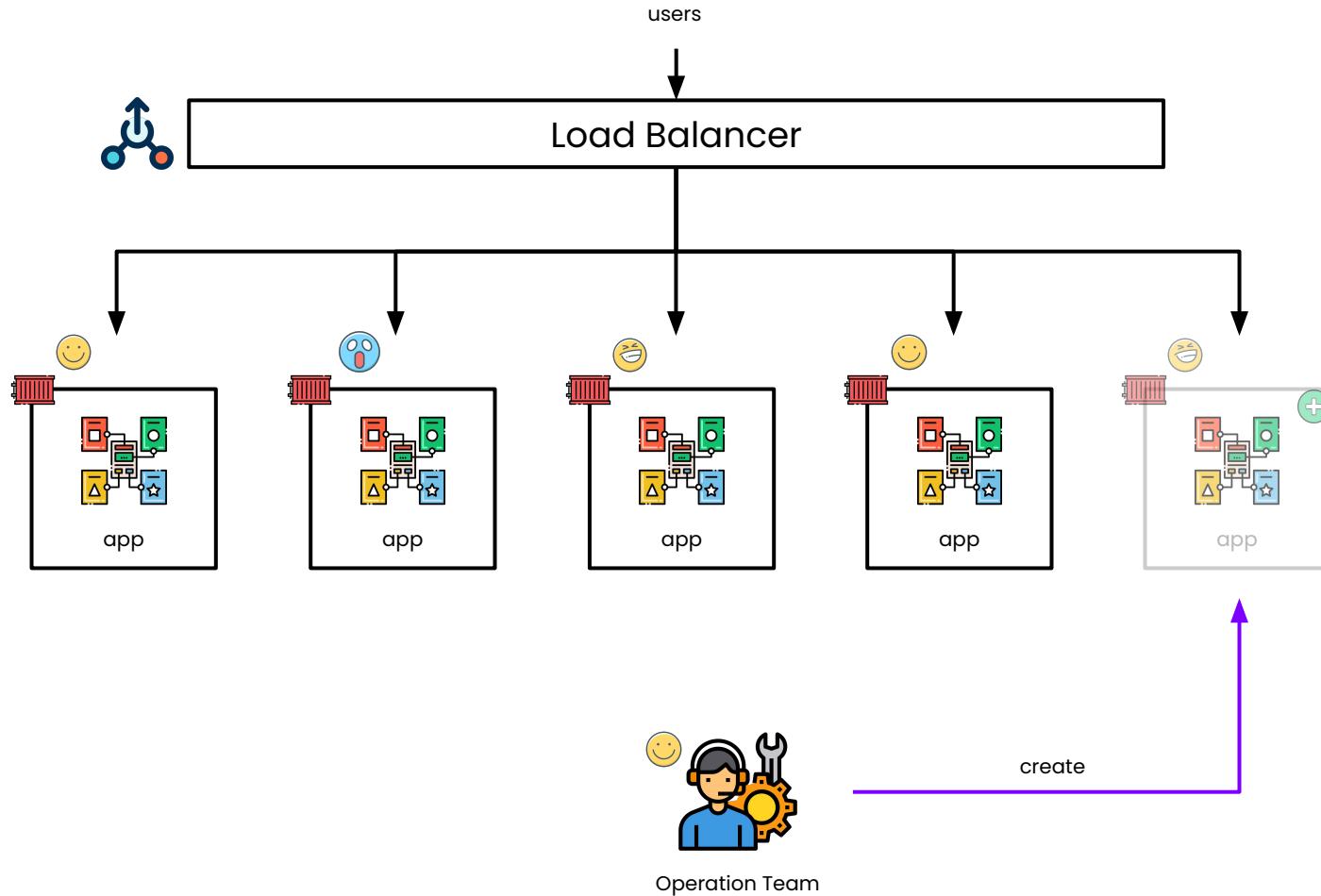


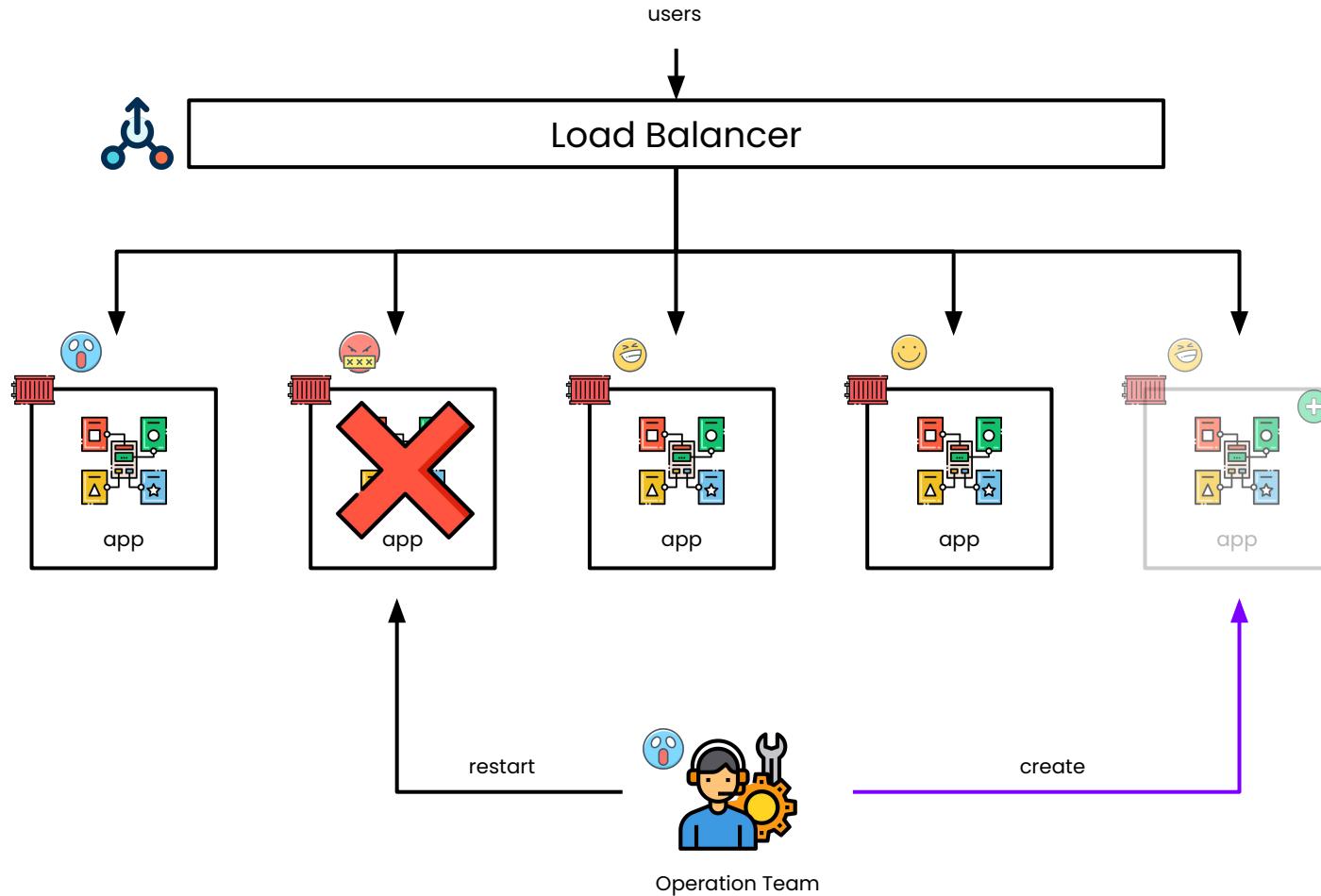




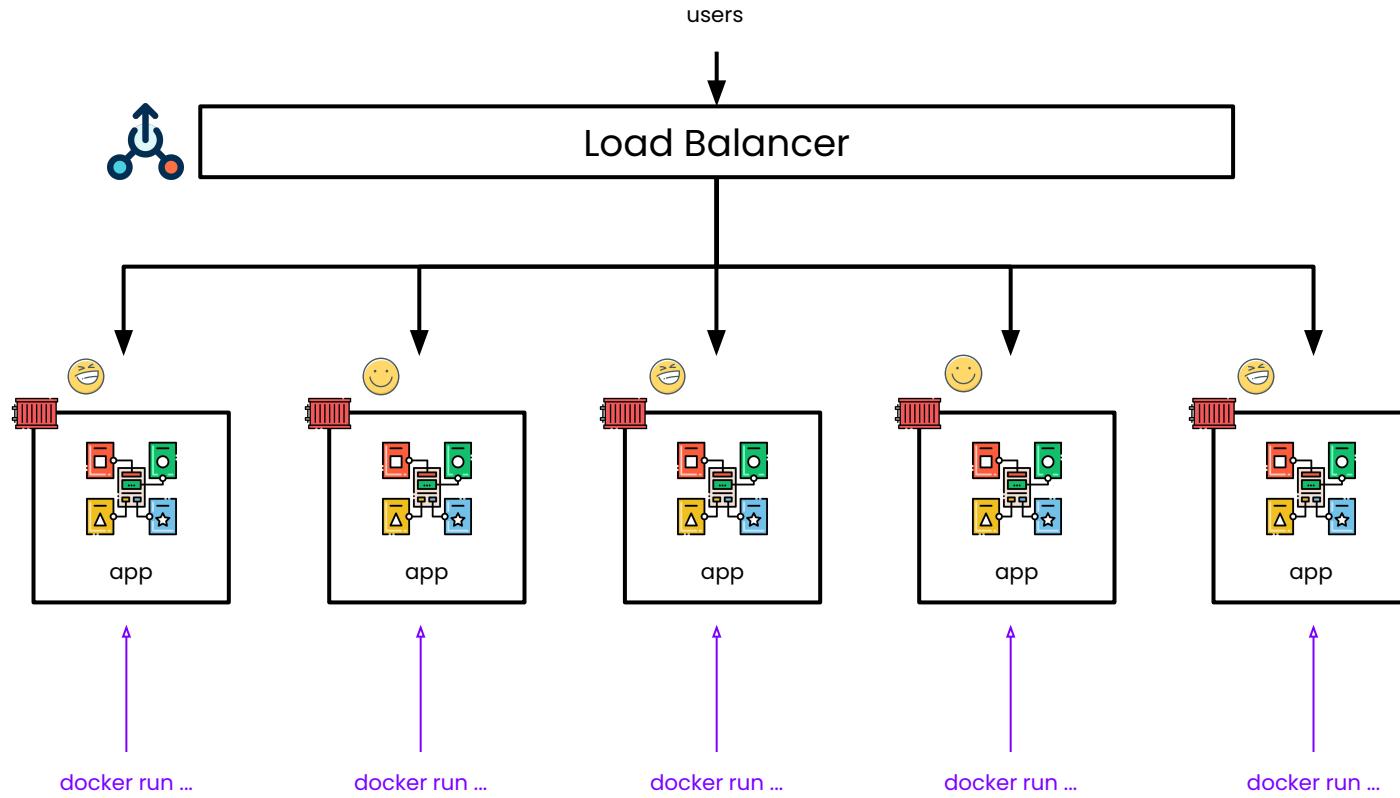


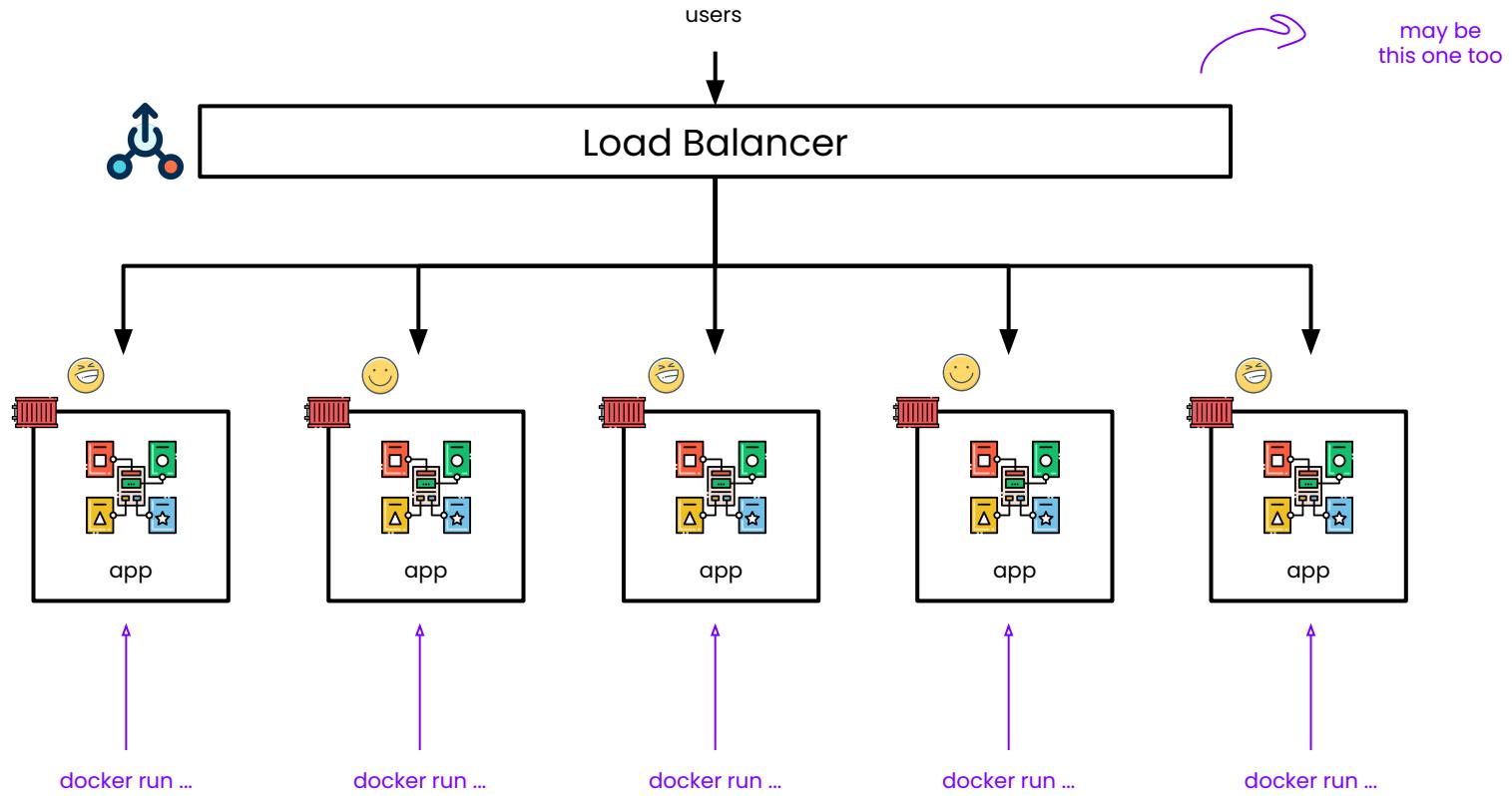
Operation Team





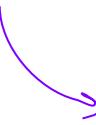
Every application **MUST** run this command





If we have a ton of application

If we have a ton of application



Write ton of docker script

If we have a ton of application



Write ton of docker script



Meh.. not fun

This is where **docker compose** comes in

Compose Background

- Docker Compose -

Compose Background (Cont.)

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

Compose Background (Cont.)

- When Docker was new, a company called Orchard built a tool called Fig that made it really easy to manage multi-container microservices apps in a **single YAML file**.

Fast, isolated development environments using Docker.

Define your app's environment with a **Dockerfile** so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in **fig.yml** so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type **fig up**, and Fig will start and run your entire app:



YAML

Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- You could even [use Fig to deploy and manage the entire lifecycle](#) of the app with the fig command-line tool.

Fast, isolated development environments using Docker.

Define your app's environment with a [Dockerfile](#) so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in [fig.yml](#) so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type [fig up](#), and Fig will start and run your entire app:



YAML



fig CLI

Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- Behind the scenes, Fig would read the YAML file and call the appropriate Docker commands

Fast, isolated development environments using Docker.

Define your app's environment with a `Dockerfile` so it can be reproduced anywhere:

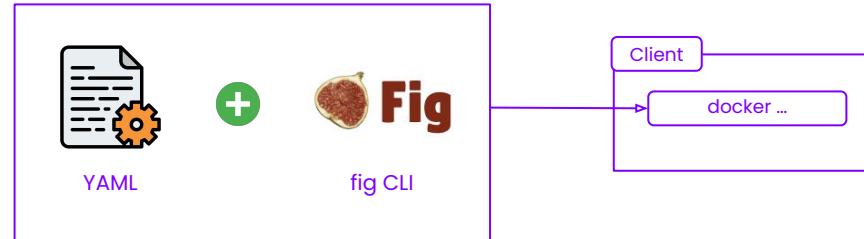
```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in `fig.yml` so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type `fig up`, and Fig will start and run your entire app:



Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- Behind the scenes, Fig would read the YAML file and call the appropriate Docker commands **to deploy and manage it.**

Fast, isolated development environments using Docker.

Define your app's environment with a `Dockerfile` so it can be reproduced anywhere:

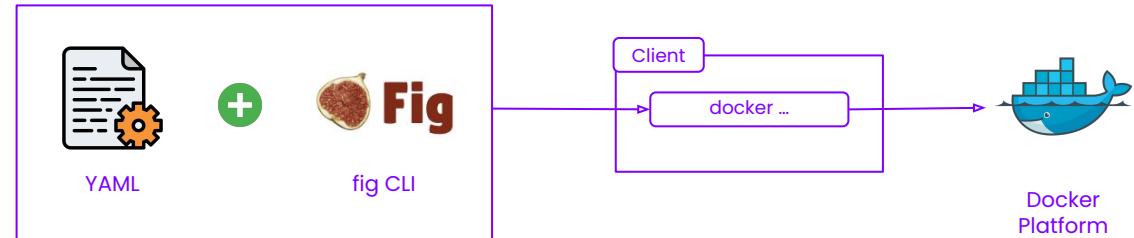
```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in `fig.yml` so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type `fig up`, and Fig will start and run your entire app:



Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- In JUL 2014, Docker Inc have announced their **acquisition of Orchard Labs**

Fast, isolated development environments using Docker.

Define your app's environment with a [Dockerfile](#) so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in [fig.yml](#) so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type [fig up](#), and Fig will start and run your entire app:

JUL 23, 2014 • 2 MIN READ

by



Chris Swan

[FOLLOW](#)

Engineer, Atsign

Docker Inc have [announced](#) their acquisition of [Orchard Labs](#), a provider of hosted Docker services. Orchard are also the developers of [Fig](#), a composition and orchestration tool for multi container Docker applications. The London based Orchard team is two strong, with prolific developers [Ben Firshman](#) and [Aanand Prasad](#).

Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- In JUL 2014, Docker Inc have announced their **acquisition of Orchard Labs**

Fast, isolated development environments using Docker.

Define your app's environment with a [Dockerfile](#) so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in [fig.yml](#) so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type [fig up](#), and Fig will start and run your entire app:

JUL 23, 2014 • 2 MIN READ

by



Chris Swan

[FOLLOW](#)

Engineer, Atsign

Docker Inc have [announced](#) their acquisition of [Orchard Labs](#), a provider of hosted Docker services. Orchard are also the developers of [Fig](#), a composition and orchestration tool for multi container Docker applications. The London based Orchard team is two strong, with prolific developers [Ben Firshman](#) and [Aanand Prasad](#).

Fig has been replaced by Docker Compose, and is now deprecated. The new documentation is on the [Docker website](#).

Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Compose Background (Cont.)

- In JUL 2014, Docker Inc have announced their **acquisition of Orchard Labs**

Fast, isolated development environments using Docker.

Define your app's environment with a [Dockerfile](#) so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in [fig.yml](#) so they can be run together in an isolated environment:

```
web:
  build: .
  command: python app.py
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

(No more installing Postgres on your laptop!)

Then type [fig up](#), and Fig will start and run your entire app:

JUL 23, 2014 • 2 MIN READ

by



Chris Swan

[FOLLOW](#)

Engineer, Atsign

Docker Inc have [announced](#) their acquisition of [Orchard Labs](#), a provider of hosted Docker services. Orchard are also the developers of [Fig](#), a composition and orchestration tool for multi container Docker applications. The London based Orchard team is two strong, with prolific developers [Ben Firshman](#) and [Aanand Prasad](#).

Fig has been replaced by [Docker Compose](#), and is now deprecated. The new documentation is on the [Docker website](#).

Reference:

- [Docker acquired Orchard Labs, 2014](#)
- [Fig tools](#)

Docker Compose

- Docker Compose -

Docker Compose

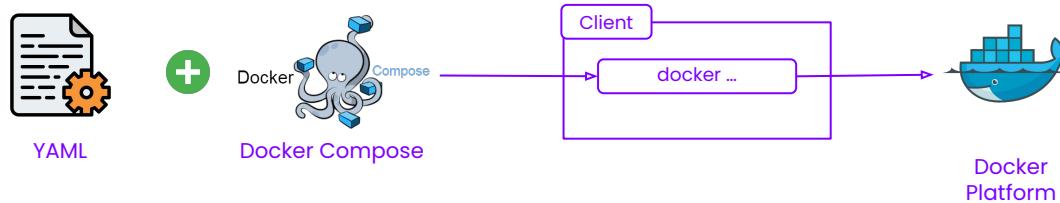
Jumpbox®

Docker Compose (Cont.)

- The Compose file is a [YAML file defining](#) services, networks and volumes. The default path for a Compose file is [./docker-compose.yml](#).
- You can use either a [.yml](#) or [.yaml](#) extension for this file. They both work.

Docker Compose (Cont.)

- The Compose file is a **YAML file defining** services, networks and volumes. The default path for a Compose file is `./docker-compose.yml`.
- You can use either a `.yml` or `.yaml` extension for this file. They both work.



Explore and resolve with the compose way

1. https://hub.docker.com/_/mongo-express
2. <https://github.com/laradock/laradock/blob/master/docker-compose.yml>

Docker compose command

- docker compose up [-d]
 - docker compose down
 - docker compose restart
 - docker compose ps
 - docker compose logs [-f]
 - docker compose exec (don't use -it)
-
- Options
 - -d or --detach
 - -f or --file

Docker Compose I: Introduction and YAML Basics

- YAML is a digestible data serialization language often used to create configuration files with any programming language.
- Designed for human interaction, YAML is a strict superset of JSON, another data serialization language. But because it's a strict superset, it can do everything that JSON can and more. One major difference is that newlines and indentation actually mean something in YAML, as opposed to JSON, which uses brackets and braces.

Reference:

- [What is yaml a beginner](#)

JSON

```

1 {
2   "json": [
3     "rigid",
4     "better for data interchange"
5   ],
6   "yaml": [
7     "slim and flexible",
8     "better for configuration"
9   ],
10  "object": {
11    "key": "value",
12    "array": [
13      {
14        "null_value": null
15      },
16      {
17        "boolean": true
18      },
19      {
20        "integer": 1
21      },
22      {
23        "alias": "aliases are like variables"
24      },
25      {
26        "alias": "aliases are like variables"
27      }
28    ],
29  },
30  "paragraph": "Blank lines denote\nparagraph breaks\n",
31  "content": "Or we\ncan auto\nconvert line breaks\ninto save space",
32  "alias": {
33    "bar": "baz"
34  },
35  "alias_reuse": {
36    "bar": "baz"
37  }
38 }
```

YAML

```

1 ---
2 # <- yaml supports comments, json does not
3 # did you know you can embed json in yaml?
4 # try uncommenting the next line
5 # { foo: 'bar' }
6
7 json:
8   - rigid
9   - better for data interchange
10 yaml:
11   - slim and flexible
12   - better for configuration
13 object:
14   key: value
15   array:
16     - null_value:
17     - boolean: true
18     - integer: 1
19     - alias: &example aliases are like variables
20     - alias: *example
21 paragraph: >
22   Blank lines denote
23
24   paragraph breaks
25 content: |-
26   Or we
27   can auto
28   convert line breaks
29   to save space
30 alias: &foo
31   bar: baz
32 alias_reuse: *foo
```

Reference Pictures:

- [JSON to YAML Convert JSON to YAML online](#)

Activity Time

- With Paper Base -

Output what you learned in **Practical Action**

Hands-on Workshop:

Docker Compose

Docker Compose II: Running Multi-container Apps

1. Change directory to docker-compose-demo/multi-container-app
2. Explore app.py
3. Explore compose.yaml
4. Run compose
5. Explore web application
 - a. Open web browser with localhost:8000
6. Exit

Reference:

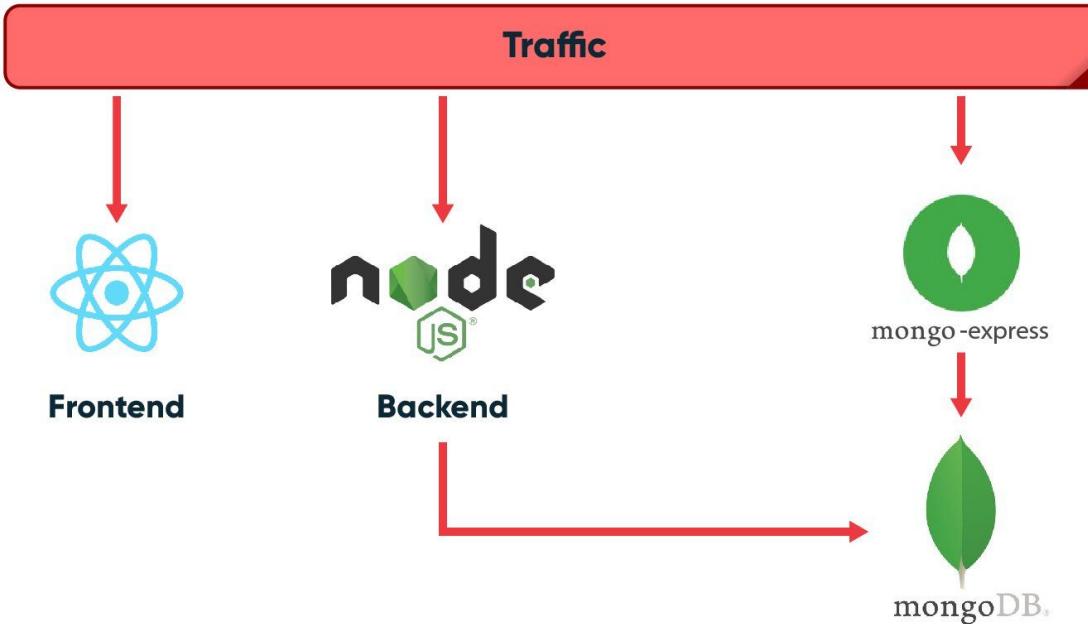
- [Docker Compose Quickstart](#)

Docker Compose III: Networking and Data in Compose

1. Change directory to docker-compose-demo/network-and-data
2. Explore compose.yaml
3. Run compose
4. Explore mongo-express with world database
 - a. Open web browser with localhost:8082
5. List all docker network
6. List all docker volume

Hands-on Workshop: Real-world Application Deployment

Application Overview & Running Application in Docker



Deploying a web service with a database backend using Docker & Docker Compose

Docker Compose with development stack (1)

1. Change directory to docker-compose-real-world
2. Run init install dependencies
3. Run all application stack in compose.yaml
4. Fix **conflict** already running port
 - a. Open compose.yaml
 - b. Change 8082 → 8083
5. Restart stack
 - a. docker compose down
 - b. docker compose up -d
6. Explore random World Cities website
 - a. Open web browser with localhost:80
 - b. Try refresh for random country in website

Docker Compose with development stack (2)

7. Down stack
8. Interchangeable by change mongo image
 - a. Open compose.yaml
 - b. Change mongo image 6.011 → 7.0.3
9. Re-run Stack
10. Explore random World Cities website
 - a. Open web browser with localhost:80
 - b. Try refresh for random country in website
11. Down stack

Docker compose watch

Prerequisites

In order to work properly, `watch` relies on common executables. Make sure your service image contains the following binaries:

- `stat`
- `mkdir`
- `rmdir`
- `tar`

Reference Pictures:

- [Use Compose Watch](#)

Tips & Tricks: Common Problems and Solutions

Avoid mistake

- Running apt-get
- Using ADD instead of COPY
- Adding your entire application directory in one line
- Using :latest
- Using external services during the build
- Adding EXPOSE and ENV at the top of your Dockerfile
- Multiple services running in the same container
- Build images for every environment

Reference:

- [9 common dockerfile mistakes](#)
- [Handling multiple environments in React with Docker](#)

Best practices

- ENTRYPOINT exec form
- Understand how CMD and ENTRYPOINT interact
- Docker ignore
- Use multi-stage builds
- Leverage build cache

Reference:

- [Building best practices](#)
- [Dockerfile: RUN vs CMD vs ENTRYPOINT](#)

Next Topic:

- Docker secret
- Cgroups
- Healthcheck
- Open Container Initiative (OCI)

Express official Health Checks and Graceful Shutdown

The screenshot shows a dark-themed web page for the Express.js documentation. At the top, there's a navigation bar with the word "Express" on the left and a search bar with a magnifying glass icon. To the right of the search bar are links for "Home", "Getting started", "Guide", "API reference", "Advanced topics" (which is bolded), and "Resources". Below the navigation, the main title "Health Checks and Graceful Shutdown" is displayed in large, bold, white font. Underneath it, a section titled "Graceful shutdown" is shown. A paragraph explains that when deploying a new application version, the previous one must be replaced, and the process manager will send a SIGTERM signal to notify the application it's being killed. Once it receives the signal, it stops accepting new requests, finishes ongoing ones, and cleans up resources like database connections and file locks. Another section, "Health checks", is also present, mentioning that a load balancer uses health checks to determine if an application instance is healthy and can accept requests. It notes that Kubernetes has two types of health checks: liveness and readiness. A third-party solutions section follows, containing a warning message in a red box: "Warning: This information refers to third-party sites, products, or modules that are not maintained by the Expressjs team. Listing here does not constitute an endorsement or recommendation from the Expressjs project team."

Reference Pictures:

- [Health Checks and Graceful Shutdown](#)

Healthcheck Discuss



Docker Healthchecks: Why Not To Use `curl` or `iwr`

Reference Pictures:

- [Docker Healthchecks: Why Not To Use `curl` or `iwr`](#)

Let's Encrypt

The screenshot shows the Let's Encrypt homepage. At the top, there is a navigation bar with links for Documentation, Get Help, Donate, About Us, and Languages. The main header features the Let's Encrypt logo and the text: "A nonprofit Certificate Authority providing TLS certificates to **363 million** websites." Below this, a call-to-action button says "Get Started" and a "Sponsor" button. A section titled "FROM OUR BLOG" lists several articles with their publication dates and titles. To the right, a section titled "MAJOR SPONSORS AND FUNDERS" displays logos from various organizations.

A non-profit Certificate Authority providing TLS certificates to **363 million** websites.

Read all about our nonprofit work this year in our [2023 Annual Report](#).

[Get Started](#) [Sponsor](#)

FROM OUR BLOG

May 20, 2024
Let's Encrypt Continues Partnership with Princeton to Bolster Internet Security
Increasing defense against BGP attacks thanks to support from the Open Technology Fund.
[Read more](#)

May 1, 2024
Taliscale's Adoption of ACME Renewal Info (ARI) enables easy and automated cert re-creation and replacement.
[Read more](#)

Apr 25, 2024
An Engineer's Guide to Integrating ARI into Existing ACME Clients
Six steps developers can take to integrate ARI into an existing ACME client.
[Read more](#)

Apr 12, 2024
Deploying Let's Encrypt's New Issuance Chains
Using our new RSA & ECDSA intermediates to sign certificates starting June 6th.
[Read more](#)

Mar 19, 2024
New Intermediate Certificates
Adding new intermediates for security, efficiency, and agility.
[Read more](#)

[Subscribe via RSS](#)

MAJOR SPONSORS AND FUNDERS

chrome AWS mozilla CISCO
EFF OHMUS Meta IdenTrust
Cloudflare Shopify IBM SAP
Cloudflare AUTOMATIC Akamai cyon
infomaniak hostinger Shoutem VULTR
PlanHost fastly cPanel
redash JIMDO zendesk netlify
dnsimple Discourse
CloudFlare OutCircle brave
overflx dreamhost GitHub UNRAID
HEINETIC HAProxy OpenShift LIVEPORT
NeverCloud render Spinifex
Red Hat cPanel WIX.com Envato
SNIPERIT exalidrop uplayer Hostcore
HOSTSTAR one.com CARGO
Cloudflare AMERIBB duda ads
cdmon Teleport NGINX Varnish
Cloudflare OpenShift talend Bitnami
IBAN yubico Cloudflare DOKK1
entri FormLabs osm ngrk
SerpApi

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จดถูกต้องกับเจ้าของลิขสิทธิ์

Jumpbox®

Let's Encrypt is SSL Grade A+

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > [jitrak.dev](#)

SSL Report: **jitrak.dev** (34.87.31.208)

Assessed on: Mon, 21 Sep 2020 19:14:12 UTC | [Hide](#) | [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



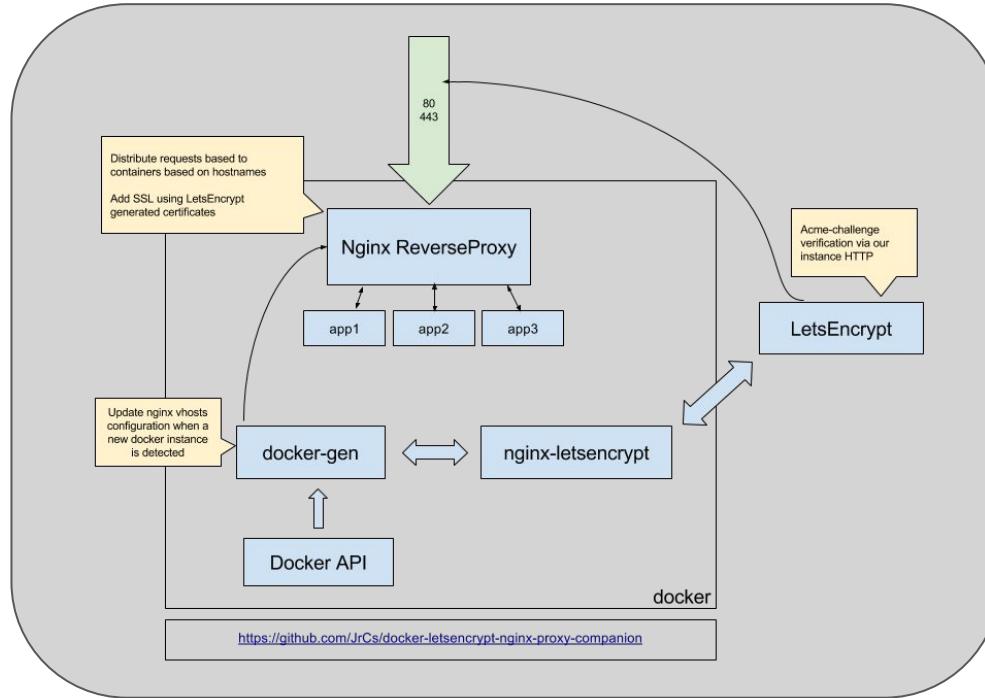
Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This site works only in browsers with SNI support.

This server supports TLS 1.3.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. [MORE INFO »](#)

acme-companion



Reference Pictures:

- [acme-companion](#)

Handling multiple environments in React with Docker

1. Run npm run build when the server starts
2. Build JS when Docker build runs
3. Use a relative API URL
4. Use a separate config file

Reference:

- [Handling Multiple Environments](#)

Next Steps & Advanced Tools

Docker Desktop license agreement

Docker Desktop license agreement

Docker Desktop is licensed under the Docker Subscription Service Agreement. When you download and install Docker Desktop, you will be asked to agree to the updated terms.

Our [Docker Subscription Service Agreement](#) states:

- Docker Desktop is free for small businesses (fewer than 250 employees AND less than \$10 million in annual revenue), personal use, education, and non-commercial open source projects.
- Otherwise, it requires a paid subscription for professional use.
- Paid subscriptions are also required for government entities.
- The Docker Pro, Team, and Business subscriptions include commercial use of Docker Desktop.

Read the [Blog](#) and [Docker subscription FAQs](#) to learn how this may affect companies using Docker Desktop.

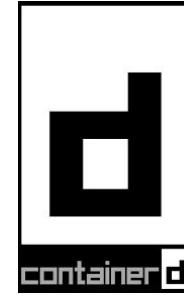
● Note

The licensing and distribution terms for Docker and Moby open-source projects, such as Docker Engine, aren't changing.

Reference:

- [Docker Desktop license agreement](#)

Alternative - Container Runtime and Container Platform

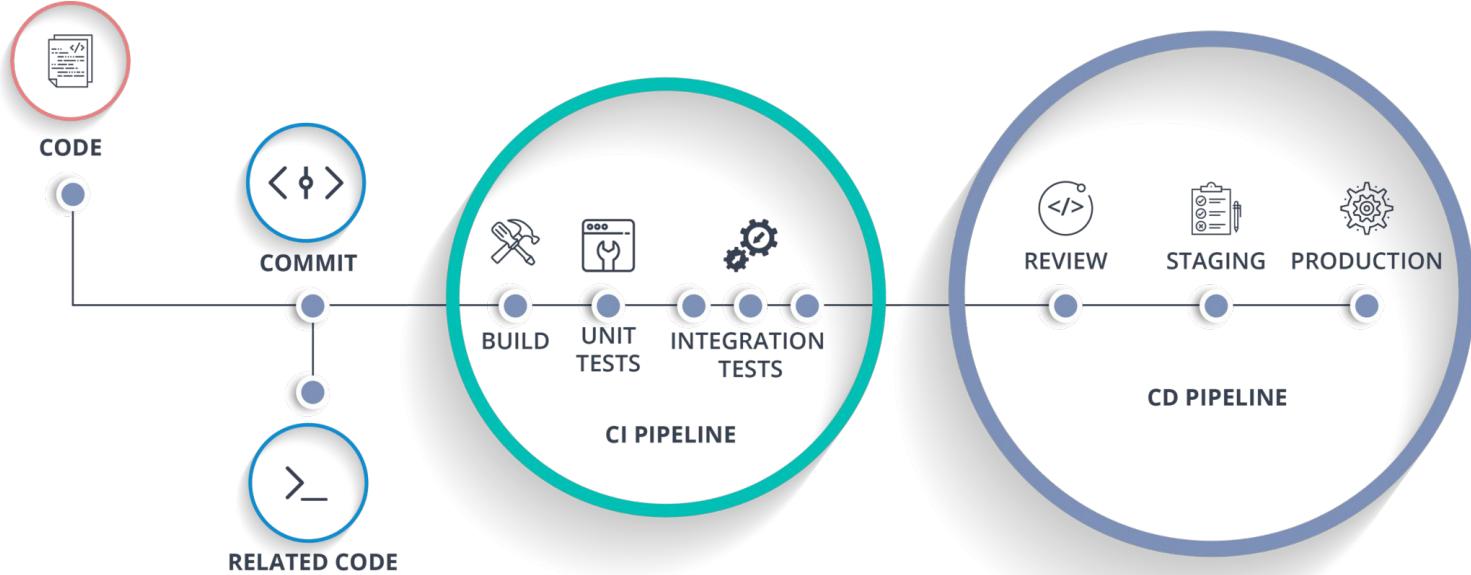


Reference:

- [6 Docker Alternatives to Look Out for in 2023](#)

Docker in CI/CD

CI/CD

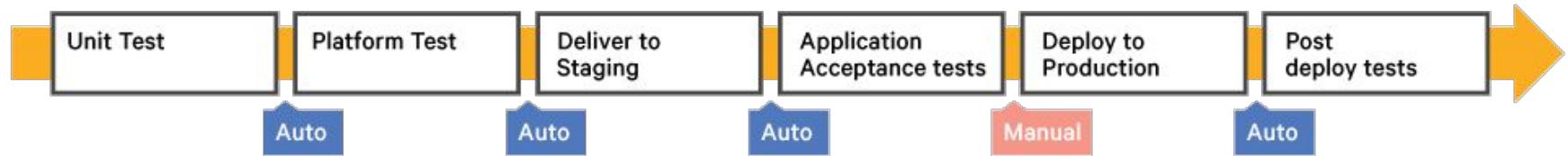


Reference Pictures:

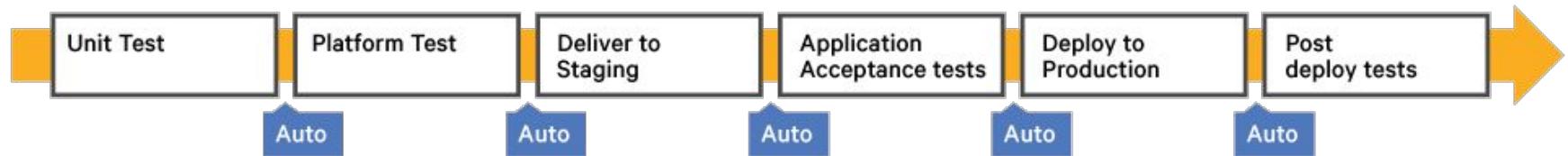
- [Learn How to Set Up a CI/CD Pipeline From Scratch](#)

CD

Continuous Delivery



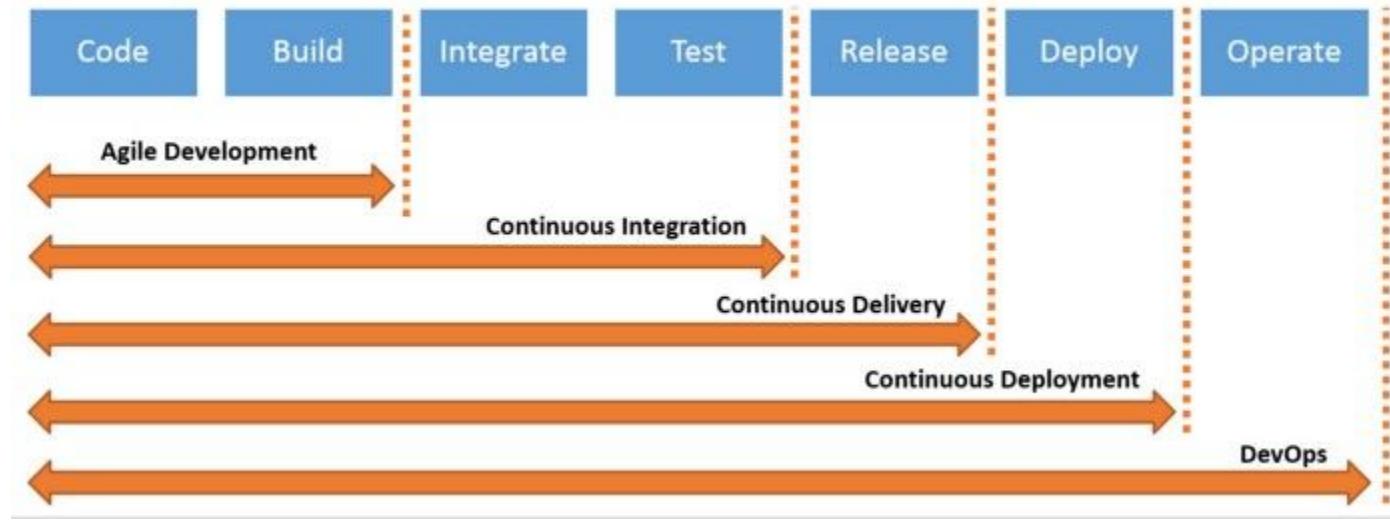
Continuous Deployment



Reference Pictures:

- [Continuous Delivery vs. Continuous Deployment: An Overview](#)

DevOps



Reference Pictures:

- [Continuous Integration / Delivery / Deployment \(CI / CD \)](#)

How to **auto build** Container image?

DockerHub

Source Type	Source	Docker Tag	Dockerfile location	Build Caching
Branch	master	latest	Dockerfile	<input checked="" type="checkbox"/> 
Tag	/^[0-9.]+\$/	{sourceref}	Dockerfile	<input checked="" type="checkbox"/> 

GitHub Action

```
name: ci

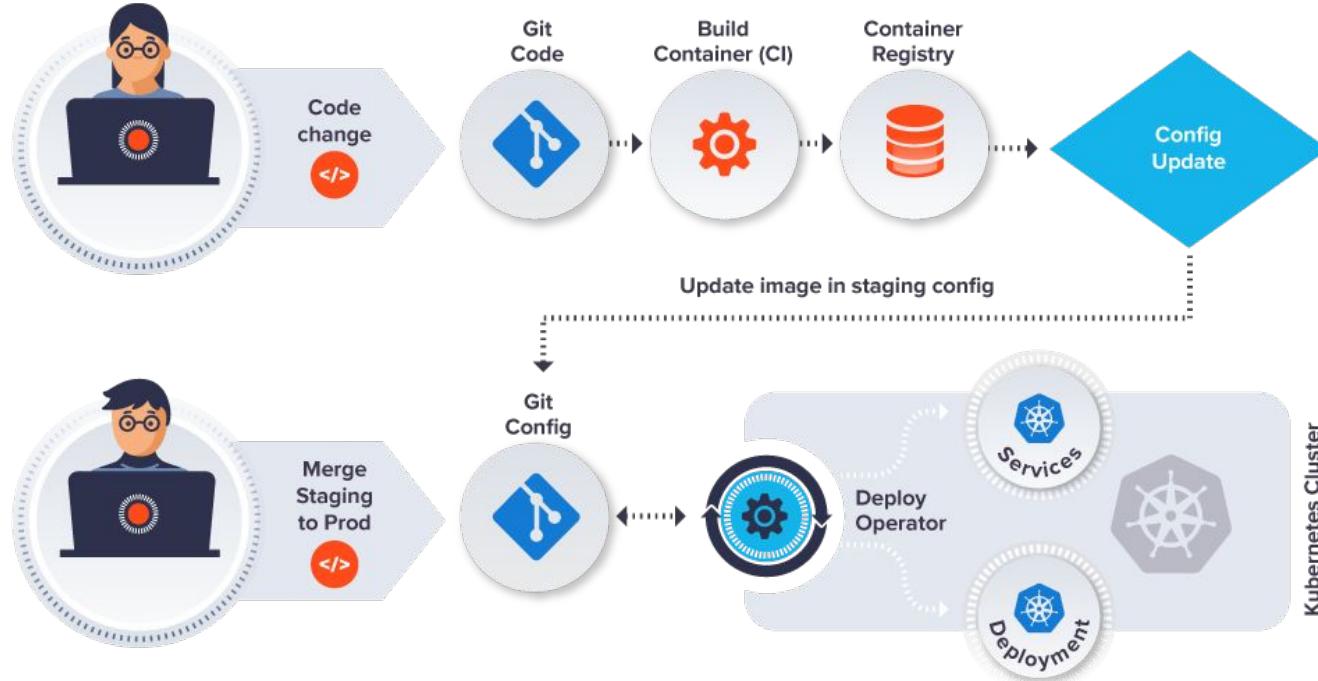
on:
  push:

jobs:
  buildx:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        # Add support for more platforms with QEMU (optional)
        # https://github.com/docker/setup-qemu-action
        name: Set up QEMU
        uses: docker/setup-qemu-action@v3
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
```

Reference:

- [setup-buildx-action](#)

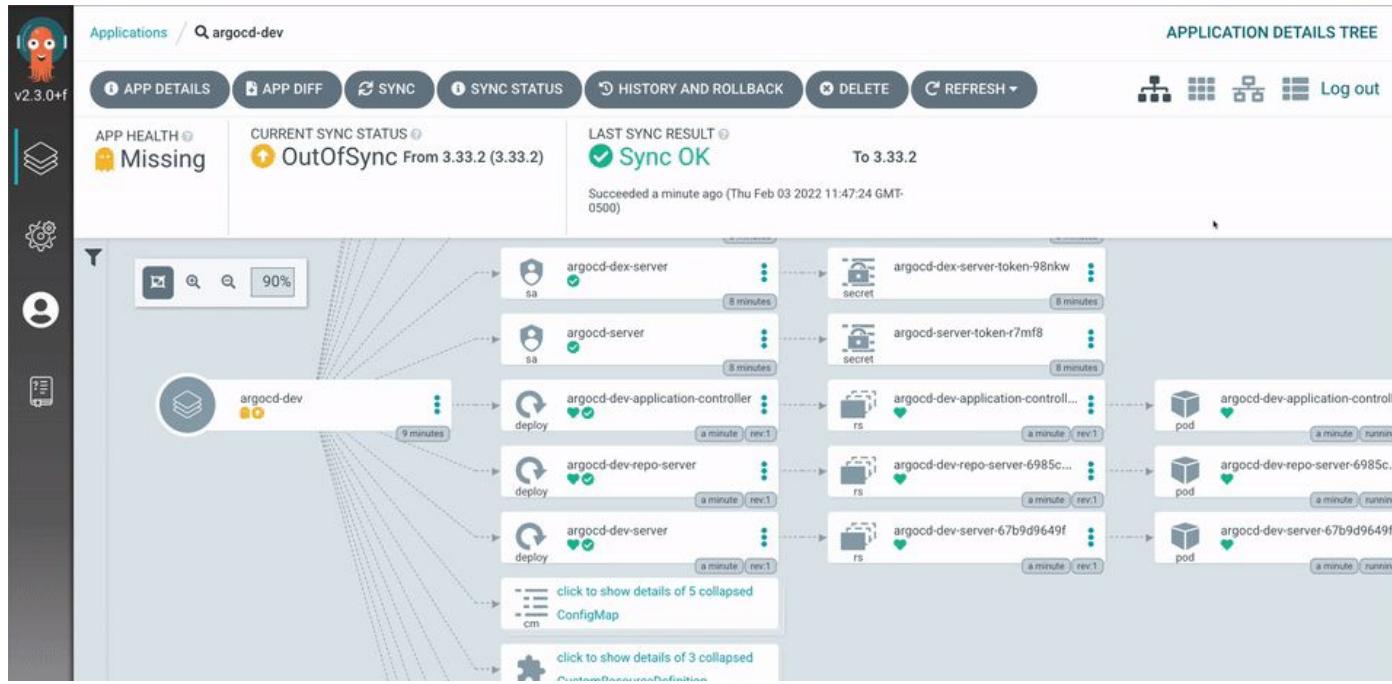
GitOps



Reference:

- [GitOps](#)

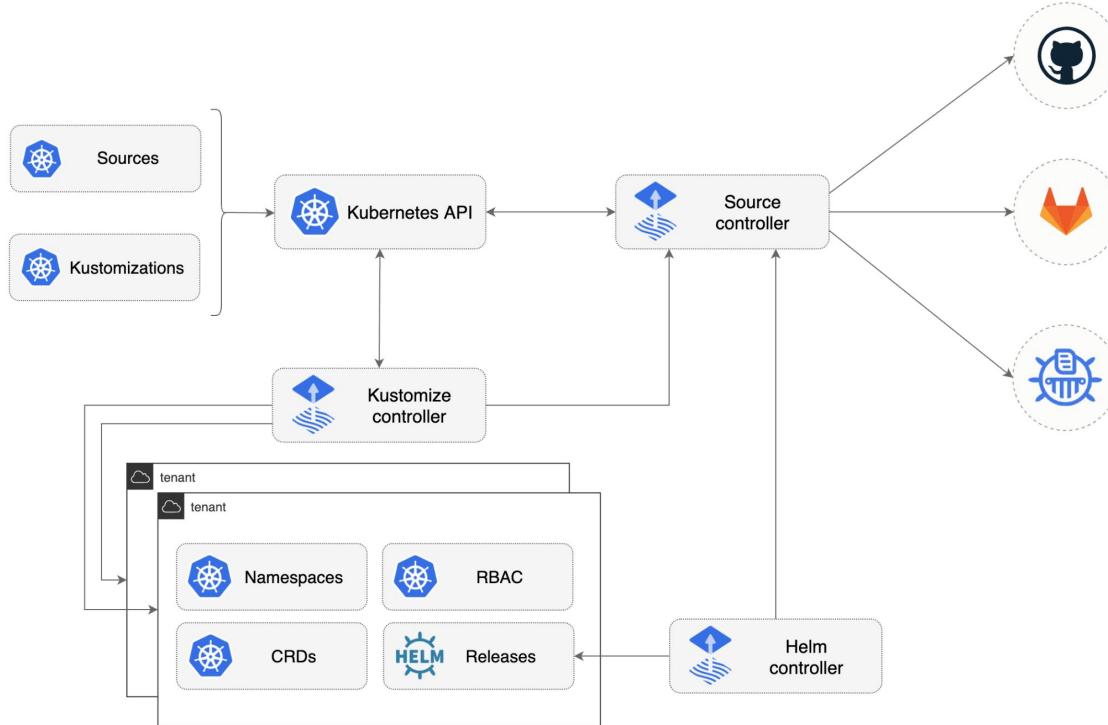
Argo CD



Reference:

- [What Is Argo CD?](#)

Flux CD



Reference:

- [Flux version 2](#)

GCP Cloud Run



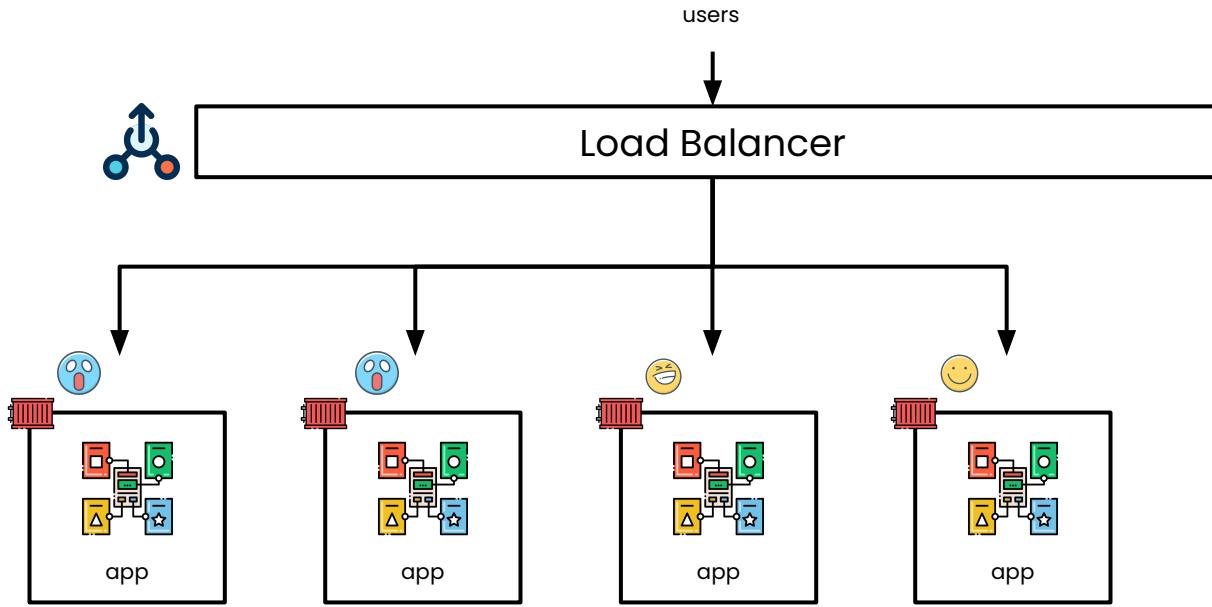
Reference:

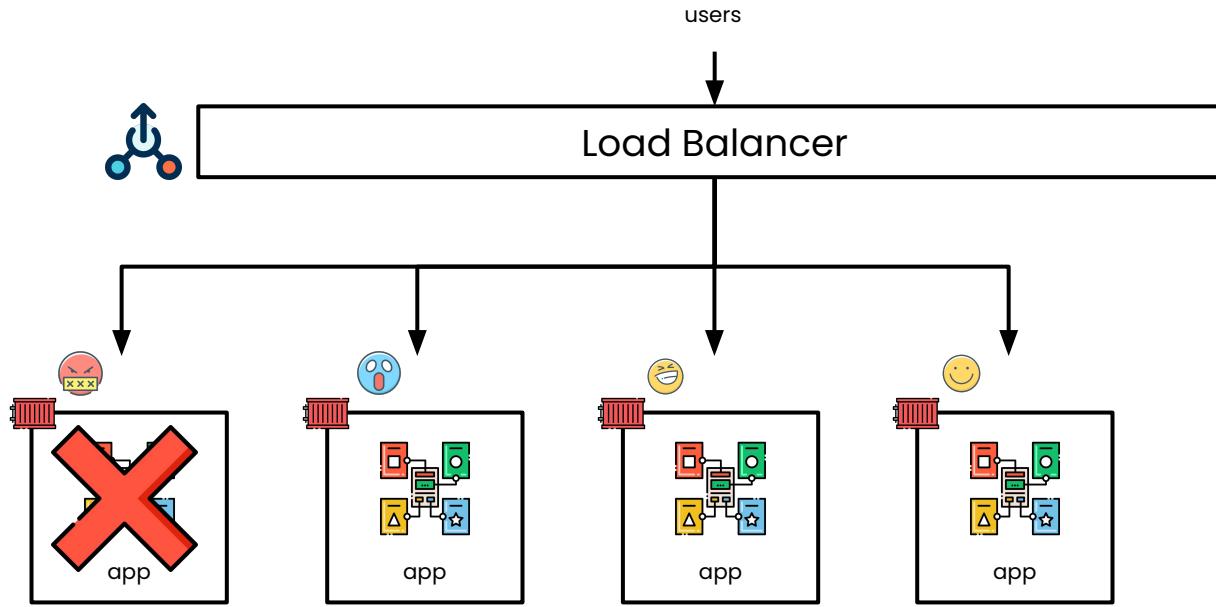
- [What is Cloud Run?](#)

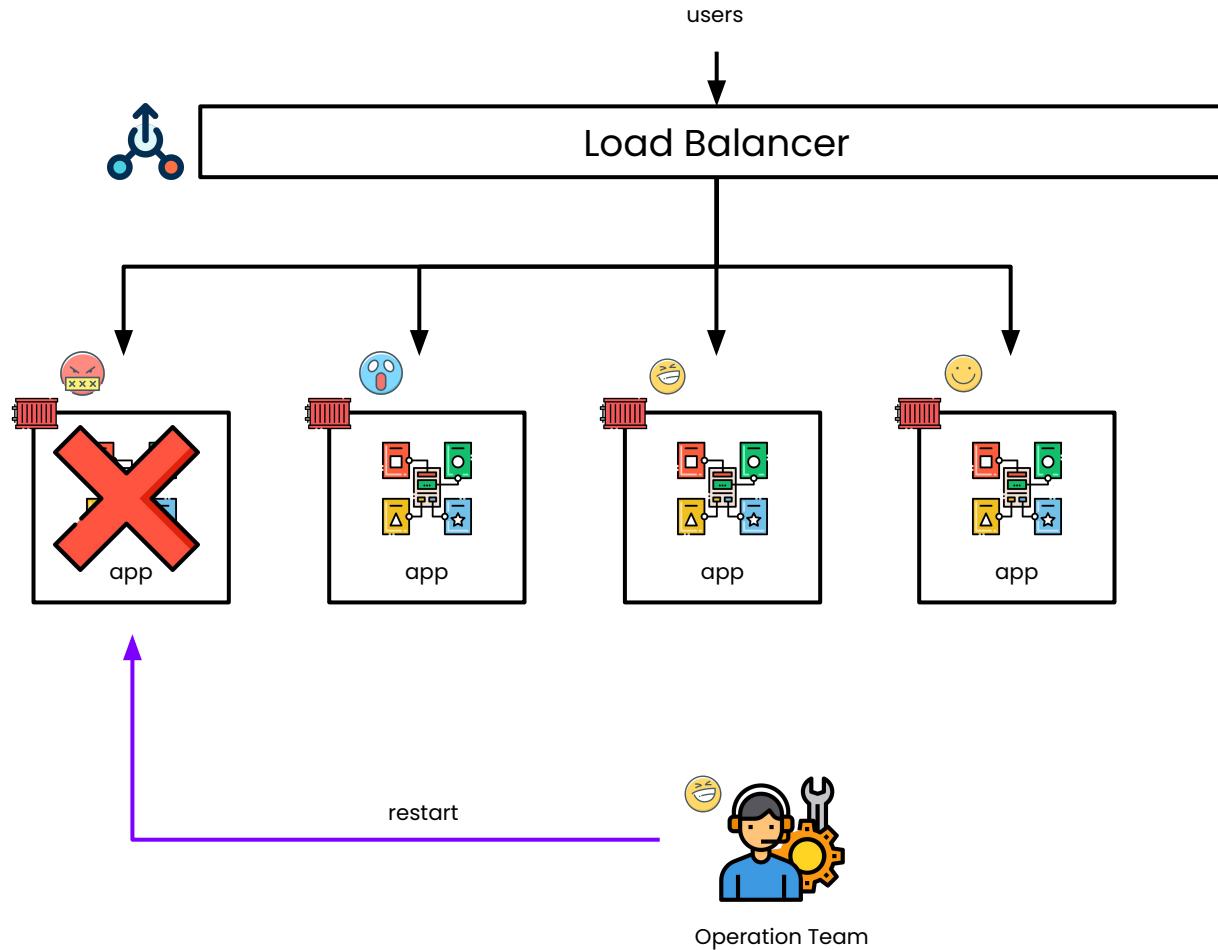


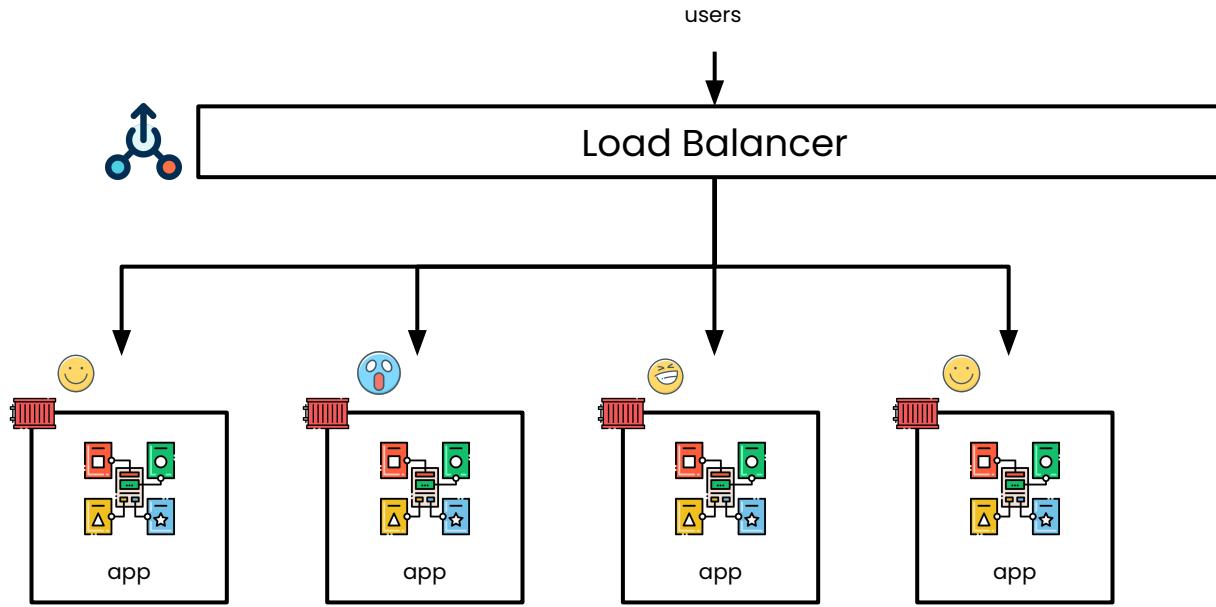
Demo

Come back to the **real world use case**

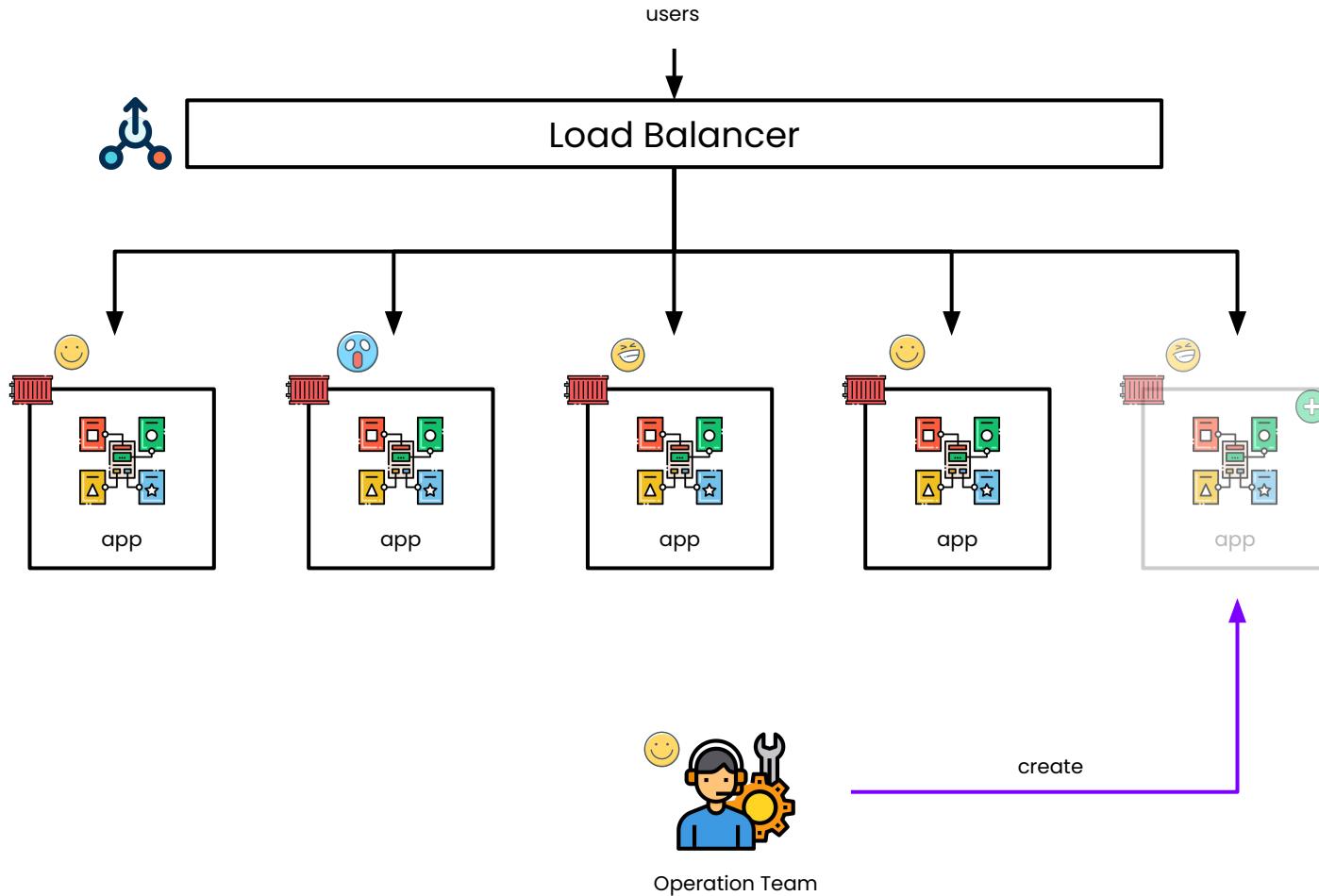


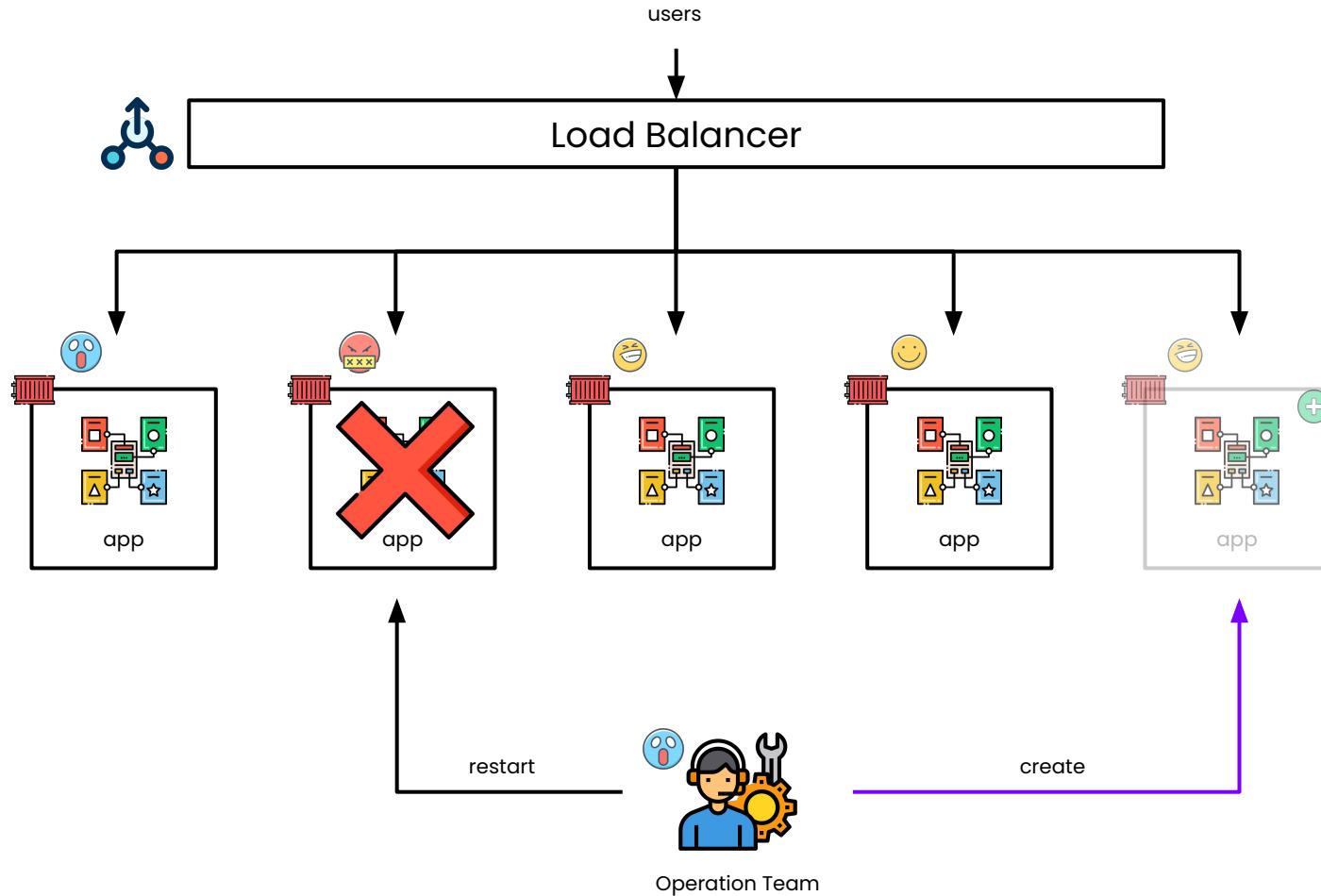




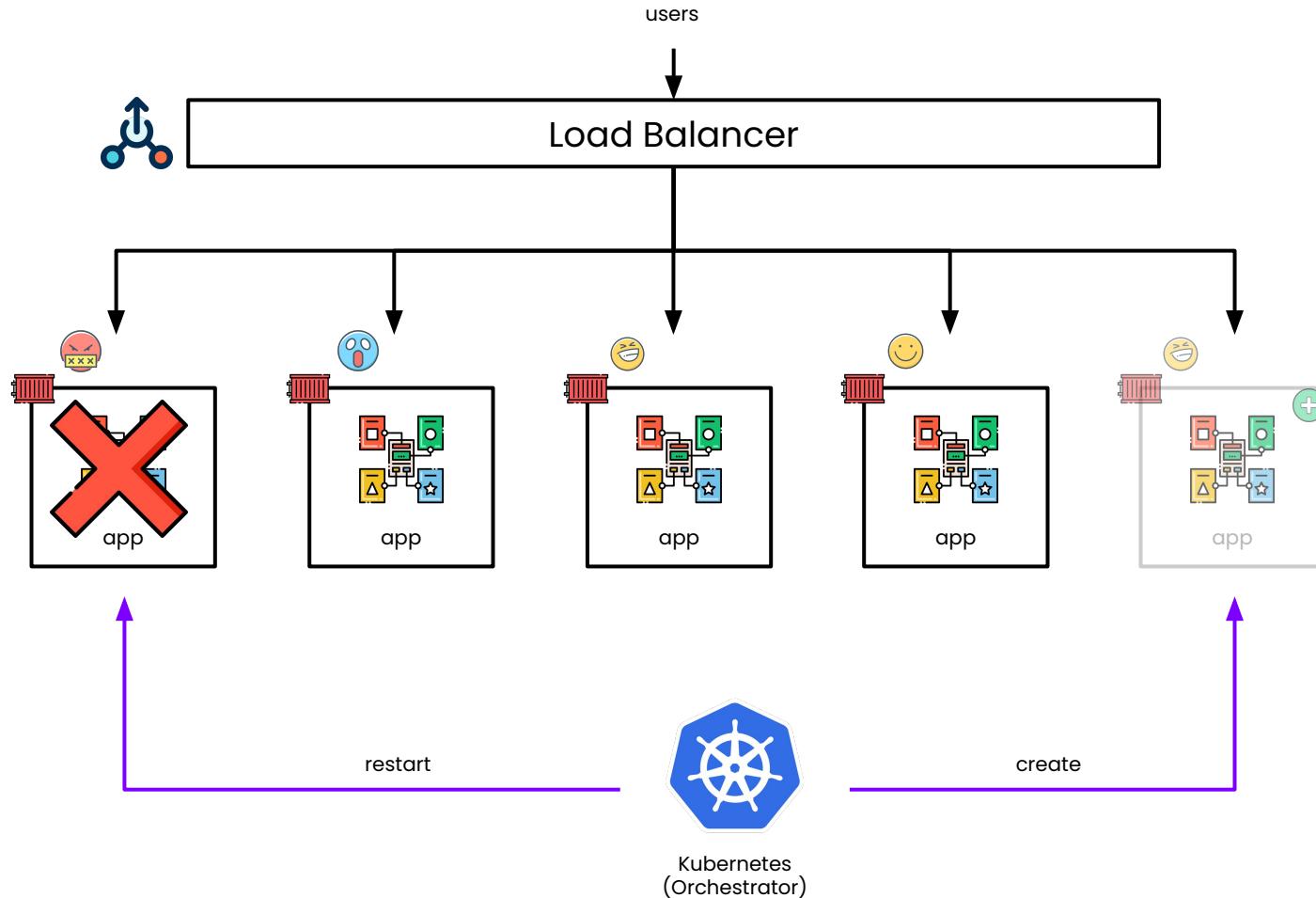


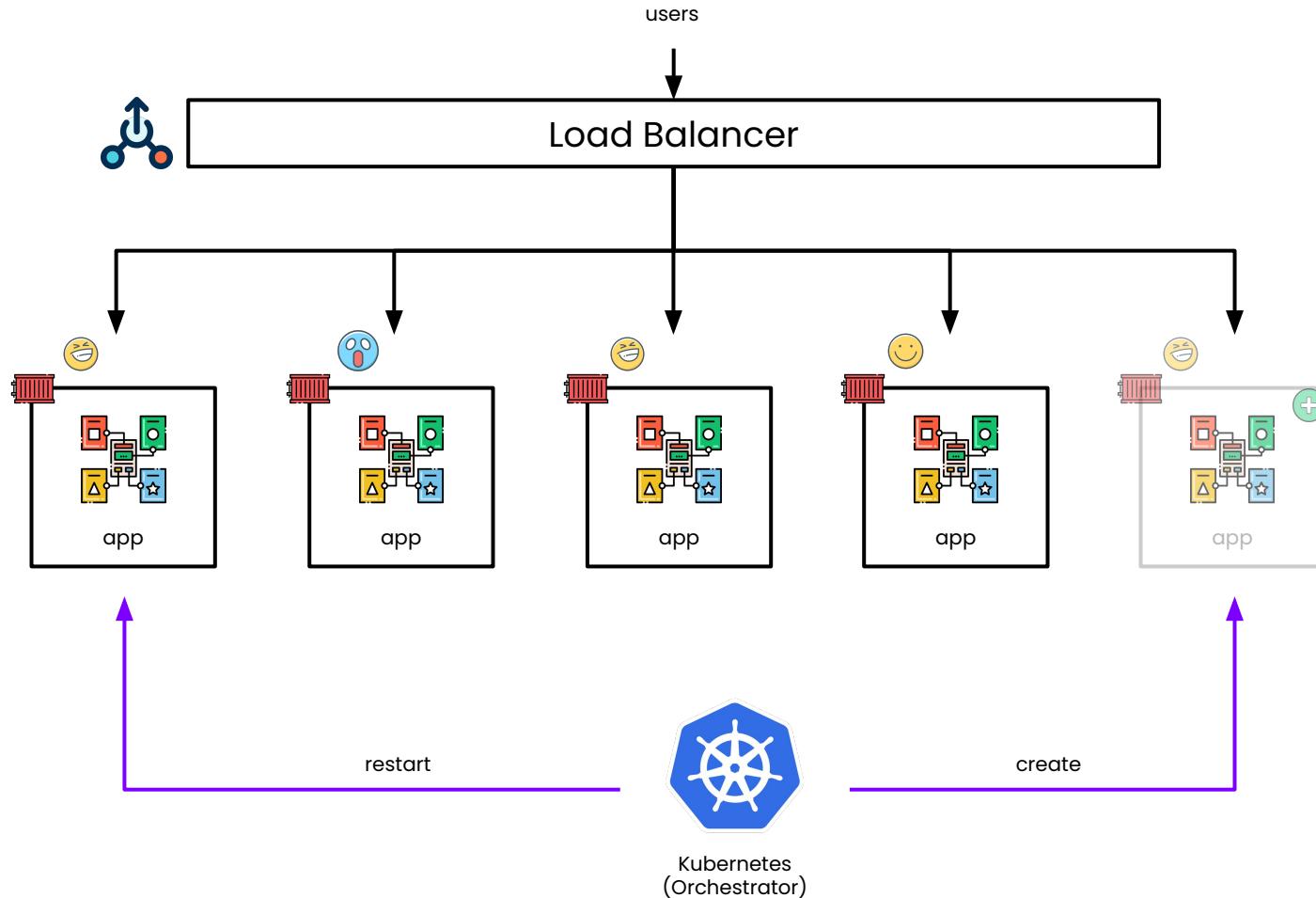
Operation Team

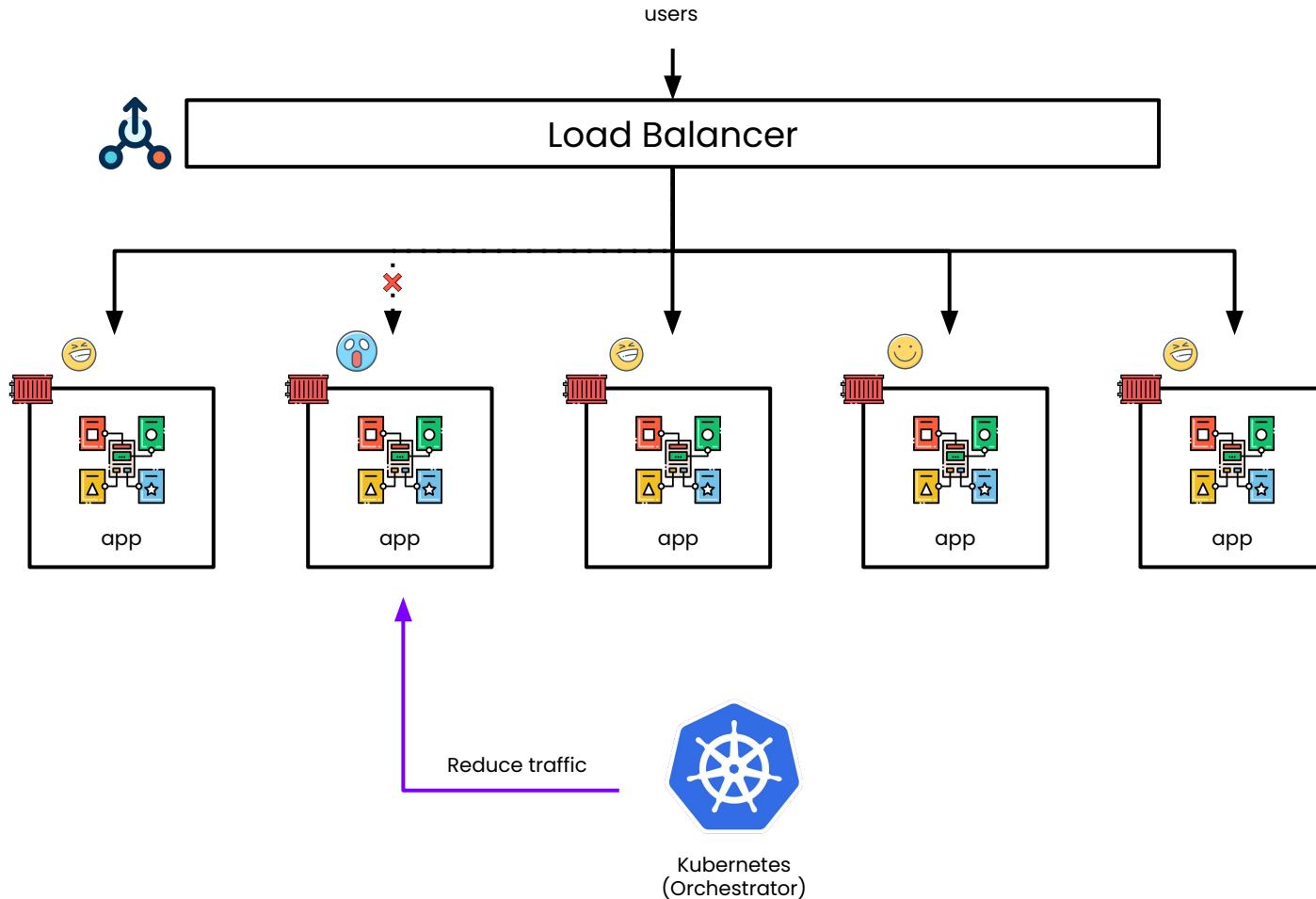


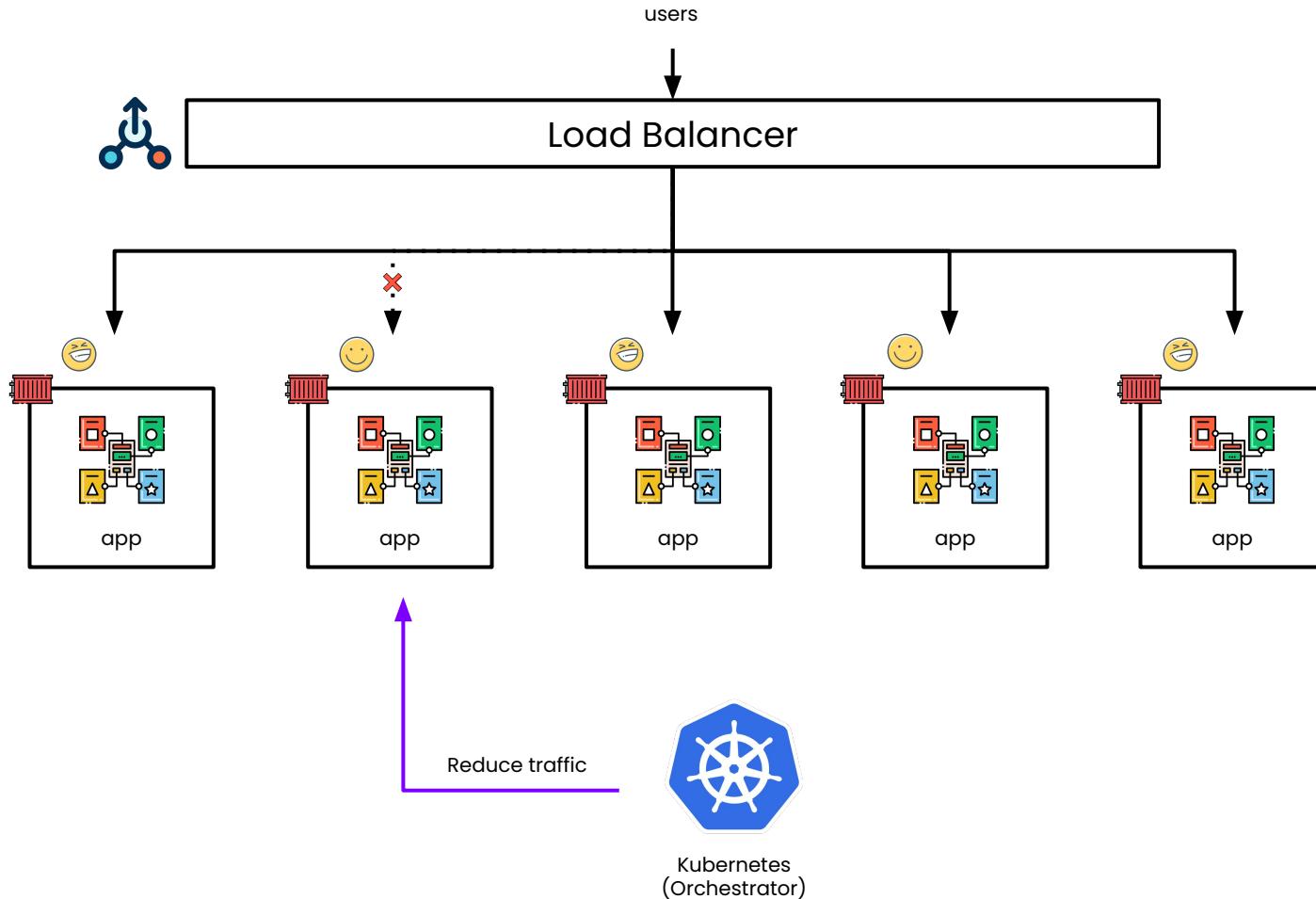


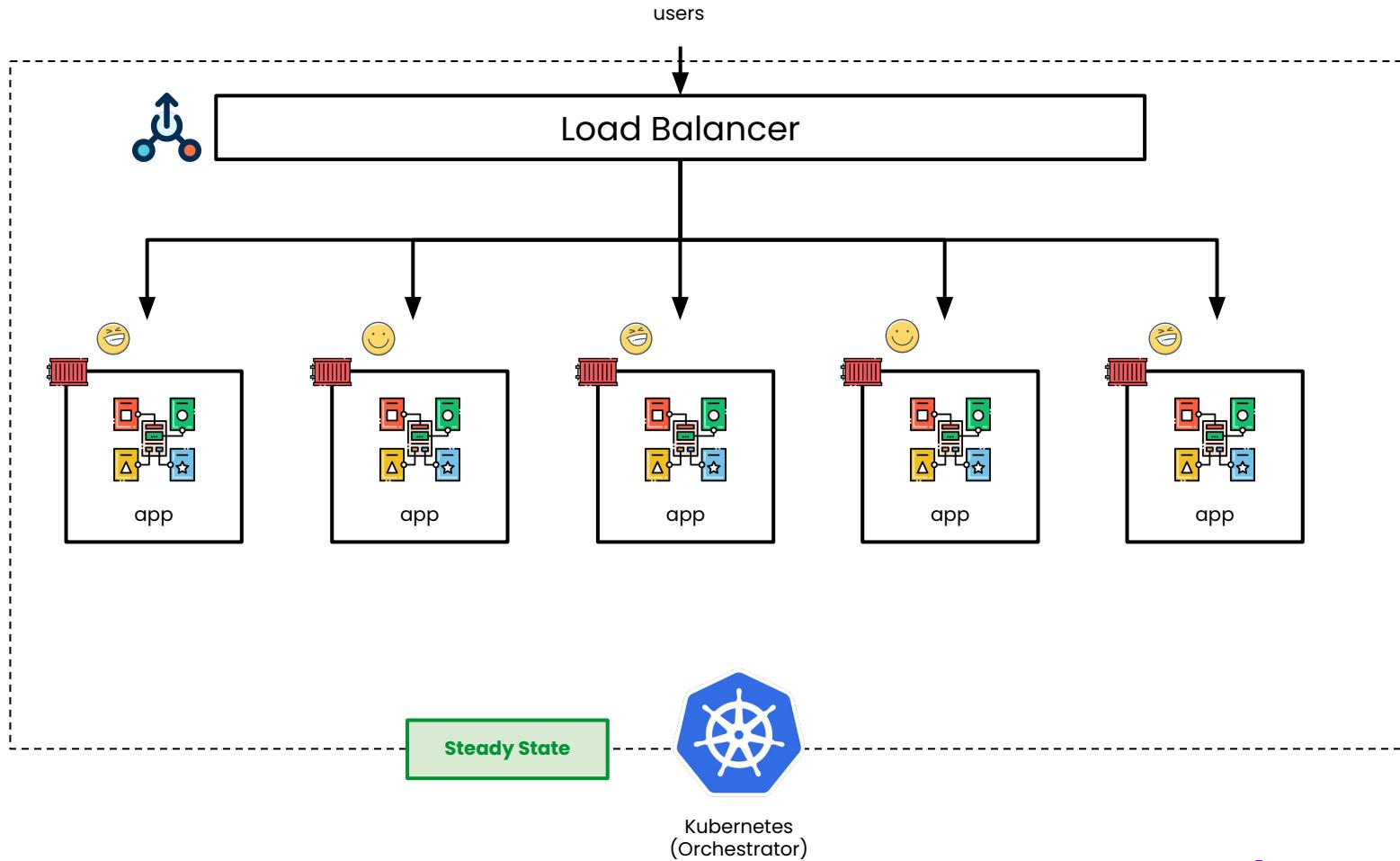
and now **Kubernetes comes in**







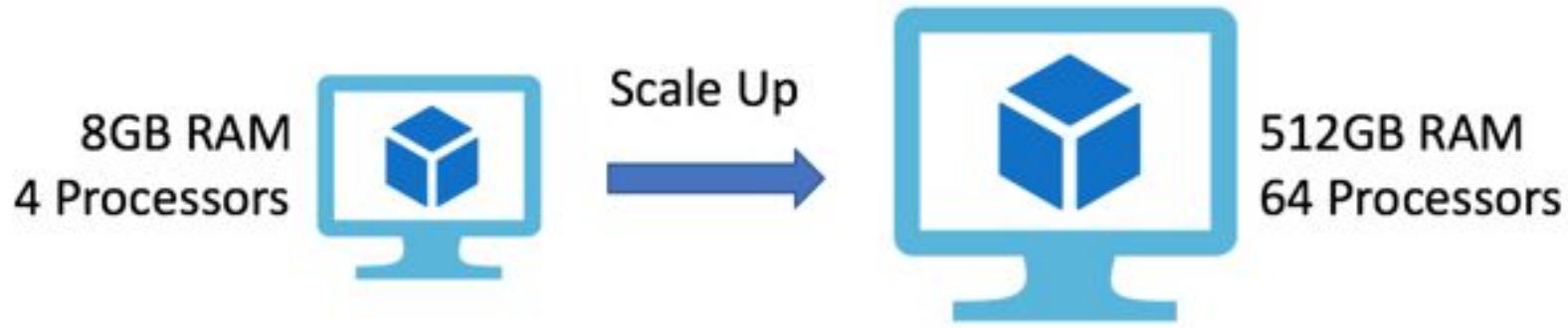




Introduction to Orchestrator

Introduction to Orchestrator Docker Swarm & Kubernetes

Business Growth

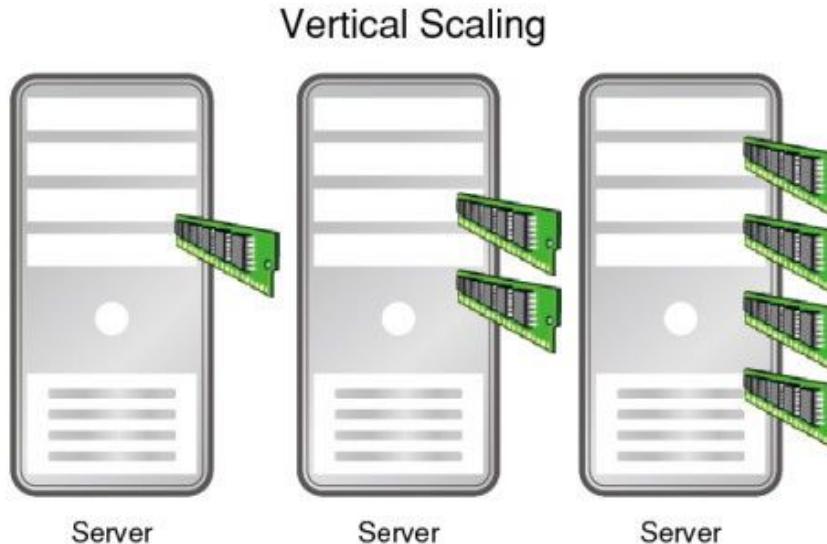


Reference Pictures:

- [Scaling Applications in the Cloud](#)

Vertical Scaling

Performance Enhancement
through Scale-Up

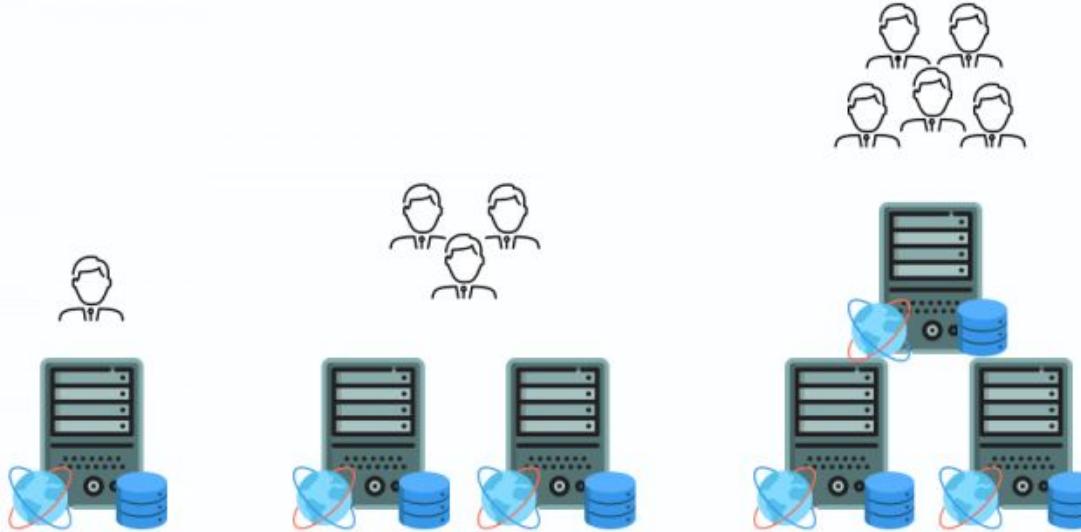


Reference Pictures:

- [Scaling a monolith vertical scaling, horizontal scaling simply defined](#)

Horizontal Scaling

Horizontal Scaling



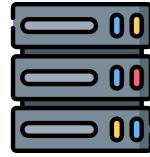
Reference Pictures:

- [Scaling a monolith vertical scaling, horizontal scaling simply defined](#)

Vertical Scaling

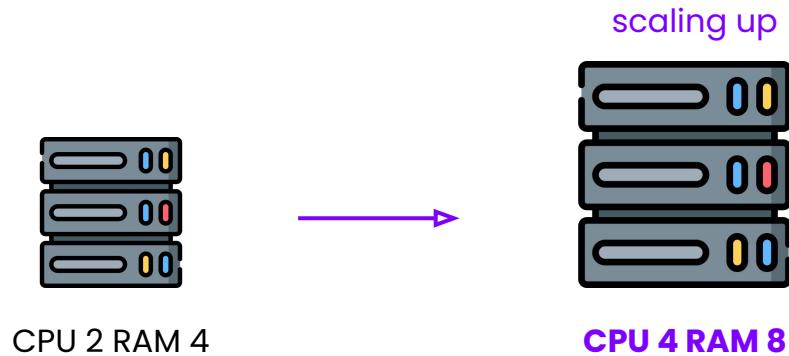
- Autoscaling -

Vertical Scaling



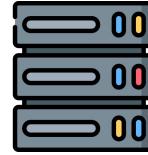
CPU 2 RAM 4

Vertical Scaling (Cont.)

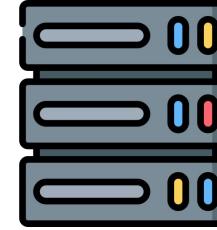


Vertical Scaling (Cont.)

scaling down



CPU 2 RAM 4

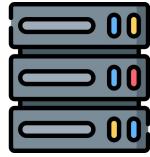


CPU 4 RAM 8

Horizontal Scaling

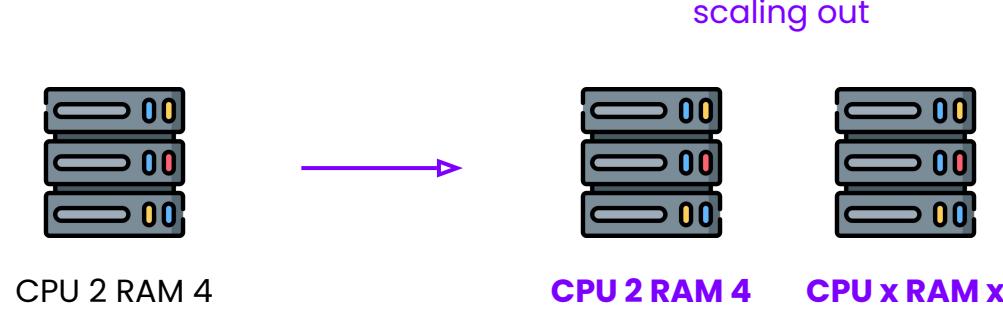
- Autoscaling -

Horizontal Scaling



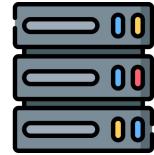
CPU 2 RAM 4

Horizontal Scaling (Cont.)

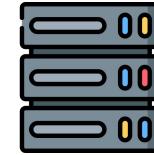


Horizontal Scaling (Cont.)

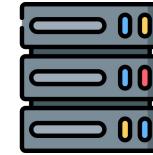
scaling in



CPU 2 RAM 4



CPU 2 RAM 4



CPU x RAM x

Vertical vs Horizontal scaling

- 4. Autoscaling -

Vertical vs Horizontal scaling

Vertical vs Horizontal scaling (Cont.)

Vertical

Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

Vertical vs Horizontal scaling (Cont.)

Vertical

Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

Vertical vs Horizontal scaling (Cont.)

Vertical



Horizontal

Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

Pros:

- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

Vertical vs Horizontal scaling (Cont.)

Vertical



Horizontal

Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

Pros:

- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

Cons:

- It can be more complex.
- It generally costs more up-front

Vertical vs Horizontal scaling (Cont.)

Vertical



Horizontal

Pros:

- It can sometimes be more cost effective
- It can sometimes be simpler to work with
- It can be easier to maintain

Cons:

- You still have a single point of failure
- It is inherently limited
- It can sometimes cost more

Pros:

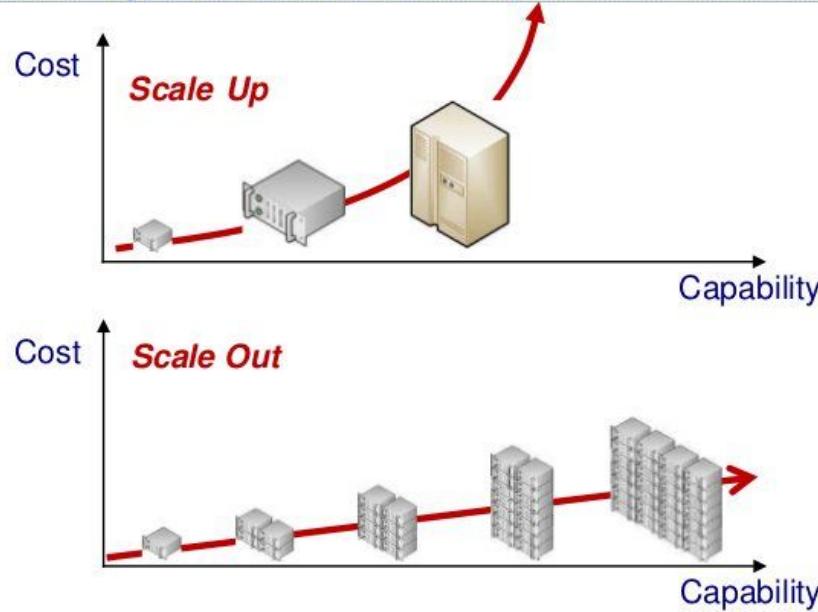
- It increases availability and resilience/fault-tolerance.
- It can be more cost-effective.
- It can increase performance.

Cons:

- It can be more complex.
- It generally costs more up-front

Economies of scale

Scale Up vs. Scale Out



www.scispike.com

Copyright © SciSpike 2015

6

Reference Pictures:

- [Scale Up vs. Scale Out](#)

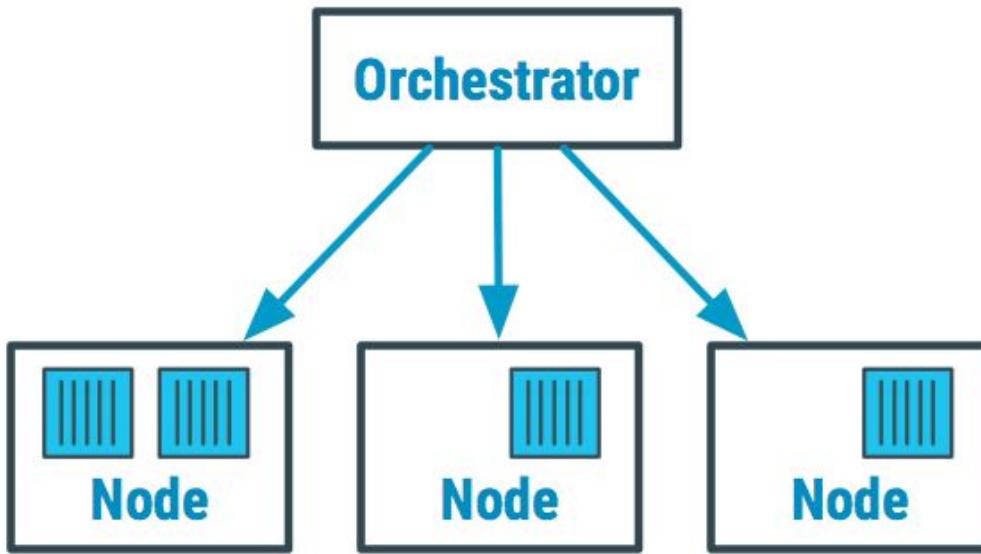


Reference Pictures:

- <https://eduinpro.com/blog/dont-run-your-holiday-ppc-ads-until-you-read-this/>

Use containers how to scale out?

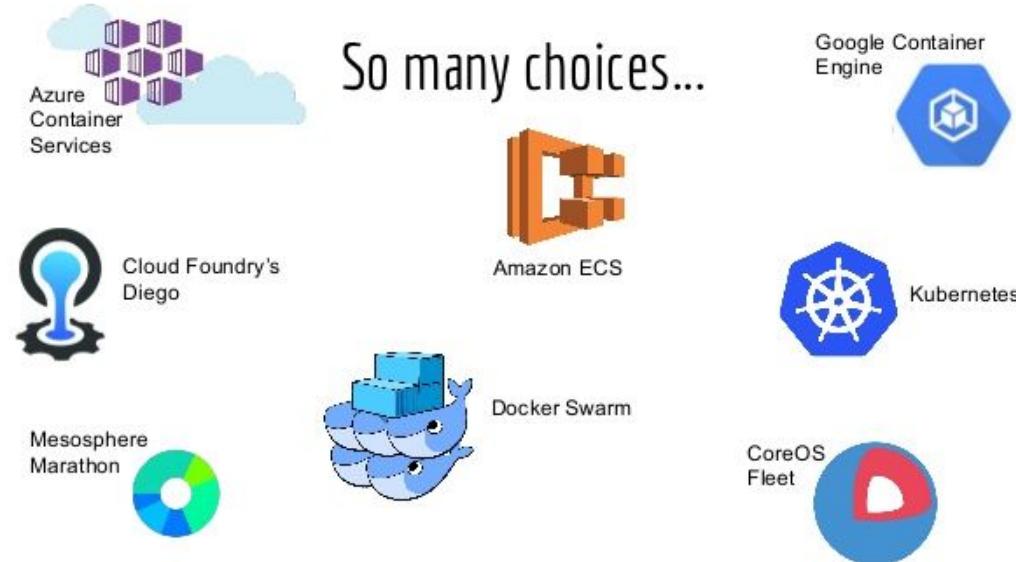
Container Orchestration



Reference Pictures:

- <https://devopedia.org/images/article/37/8281.1530784485.png>

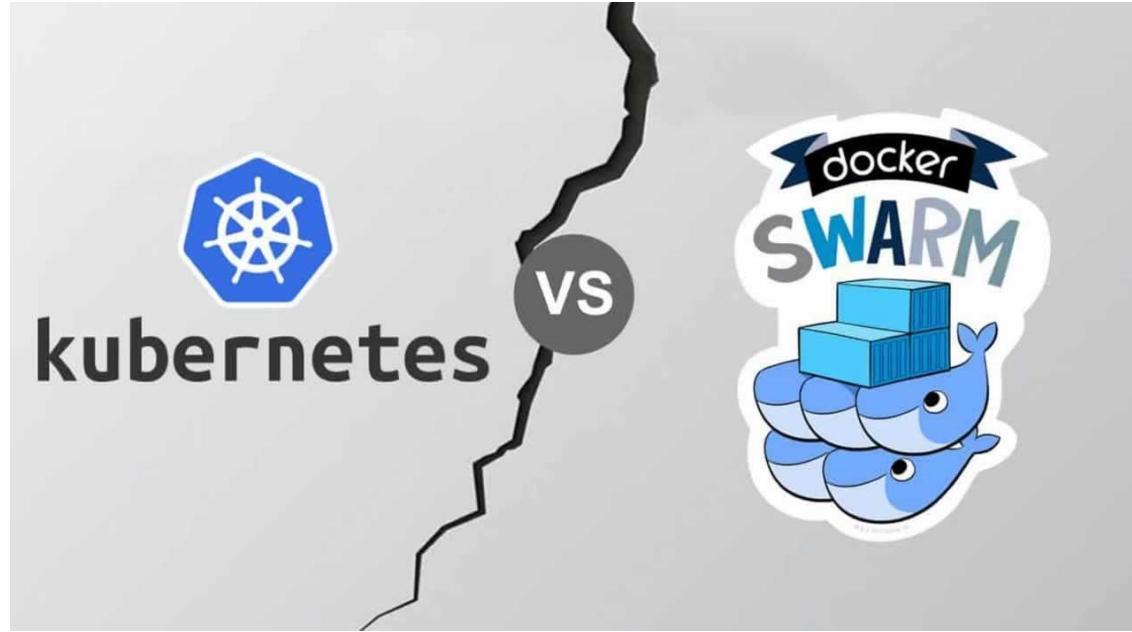
Container Orchestration tools



Reference Pictures:

- [Managed Container Orchestration with Amazon ECS](#)

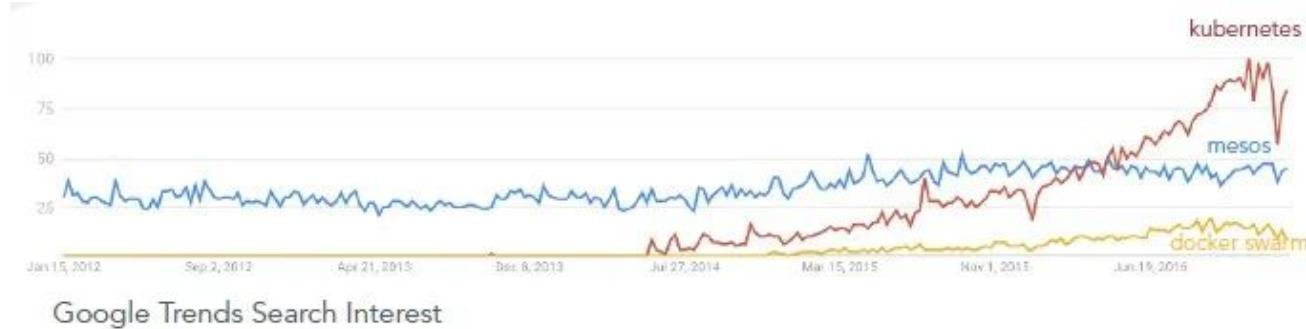
Kubernetes vs Docker Swarm



Reference Pictures:

- [Kubernetes vs Docker Swarm | Comparison of Docker and Kubernetes | Kubernetes Training | Edureka](#)

Kubernetes was the clear winner



Project Maturity

- Built by the same team that built google's [borg](#), matured within google and outside

Enterprise Adoption

- Strong - well known [customers](#) running in production

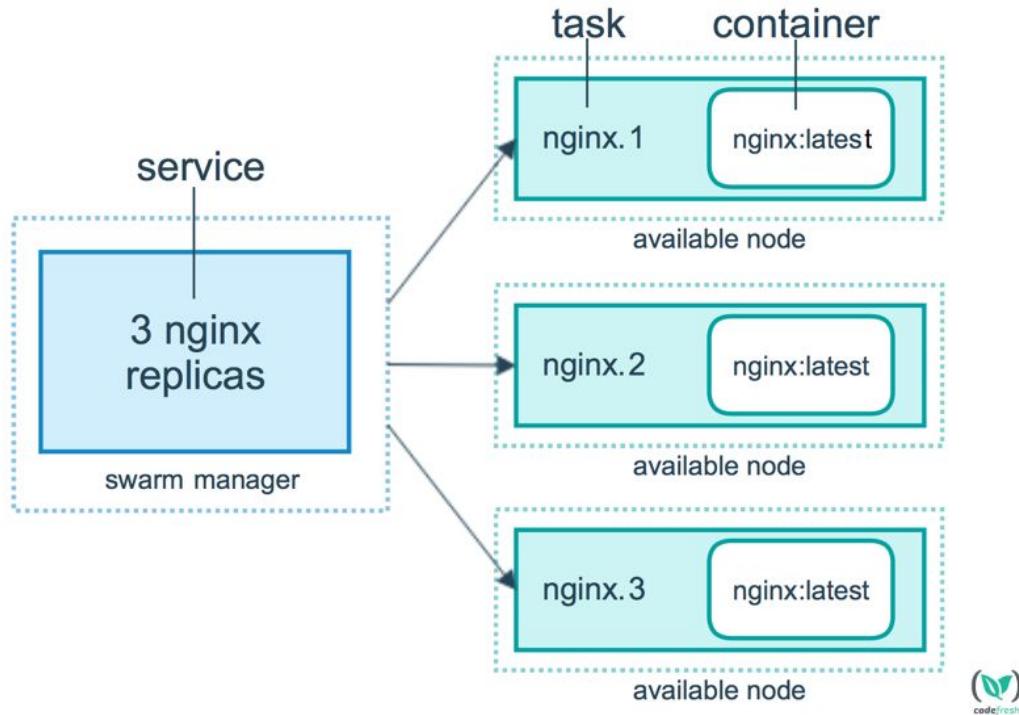
Community Strength

- > 1000 contributors
- 20K github stars
- Most active OSS community - google, red hat, intel, [box](#). Astounding pace of innovation.

Reference Pictures:

- [The Gravity of Kubernetes](#)

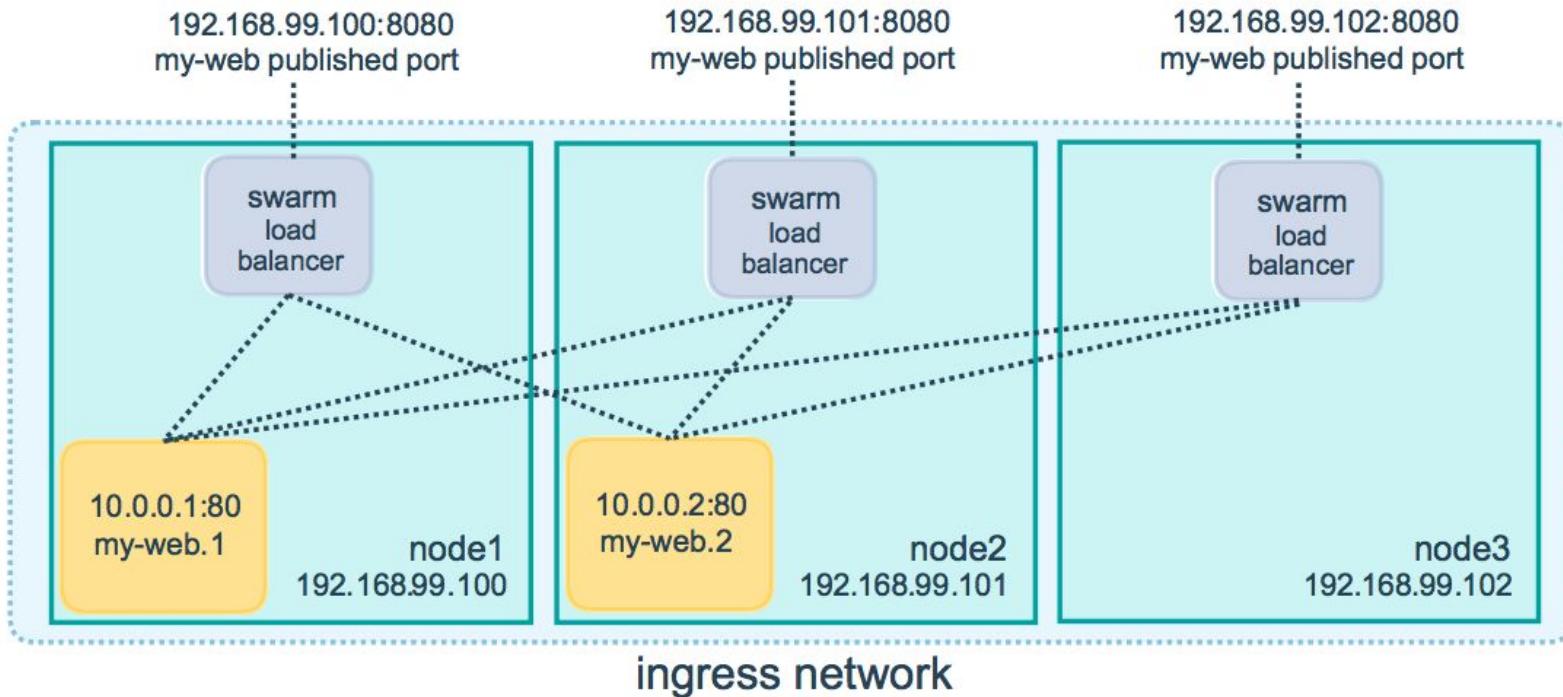
Docker Swarm



Reference:

- [Doing it the Swarm Way](#)

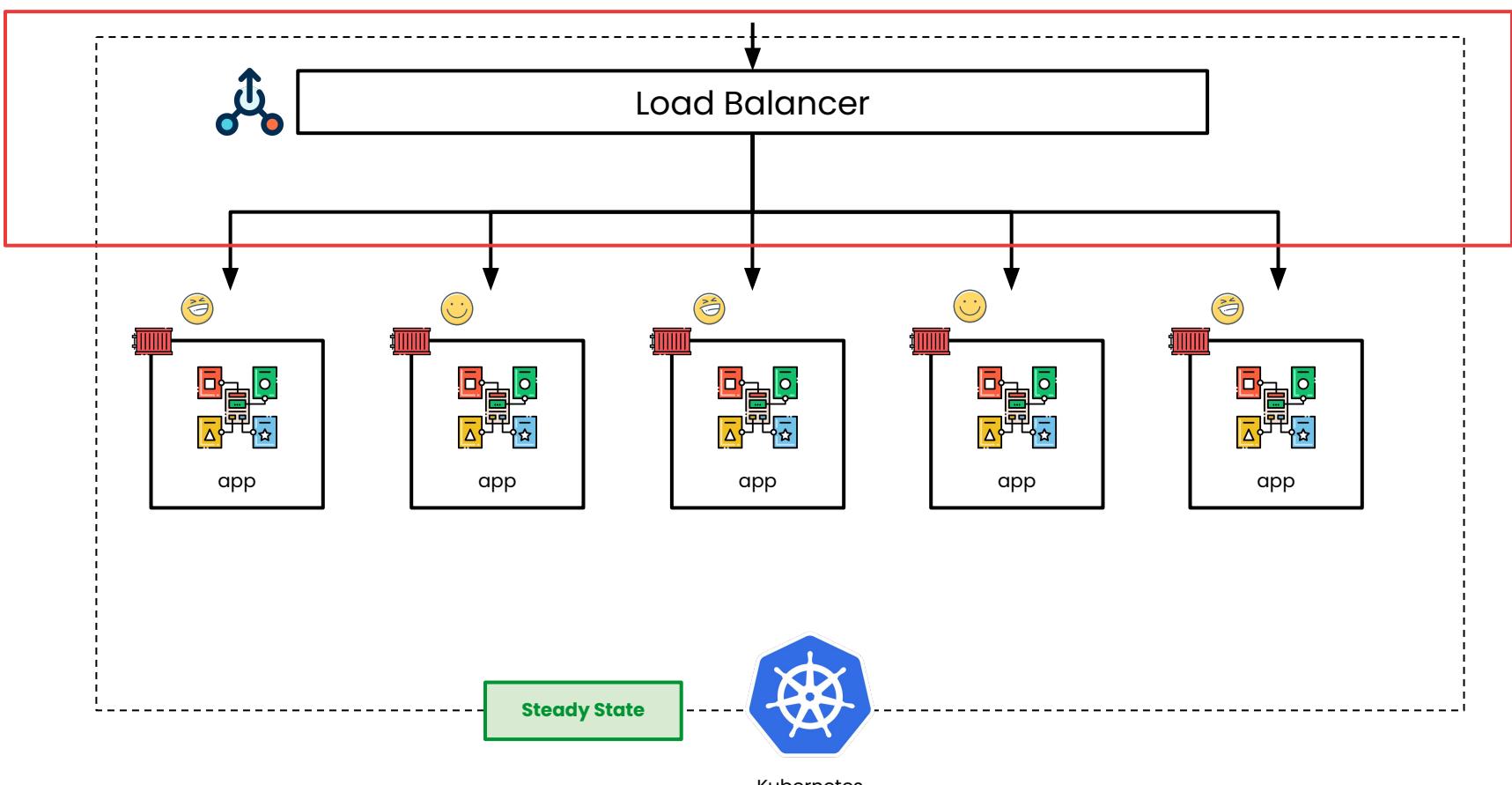
Docker Swarm (Cont.)



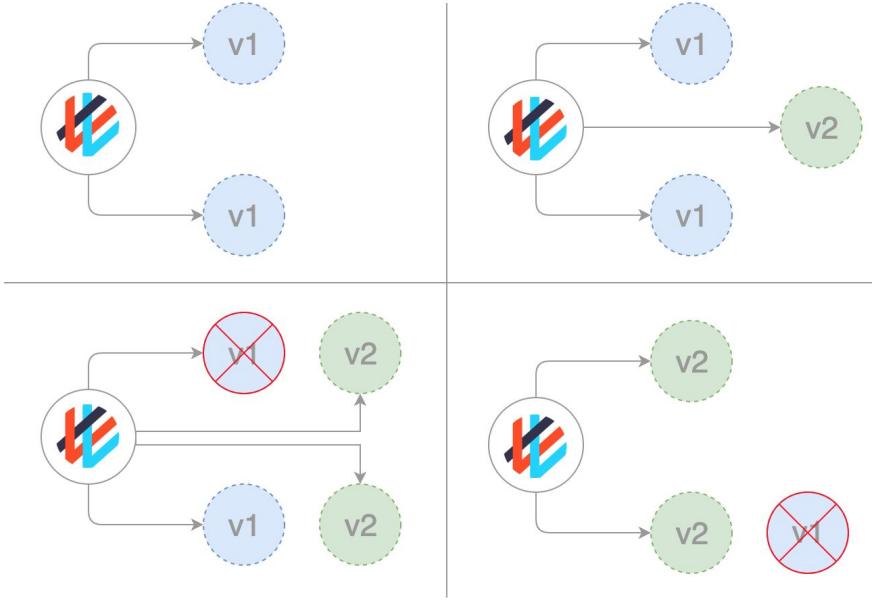
Reference:

- [Deploy to Swarm](#)

Deployment strategies



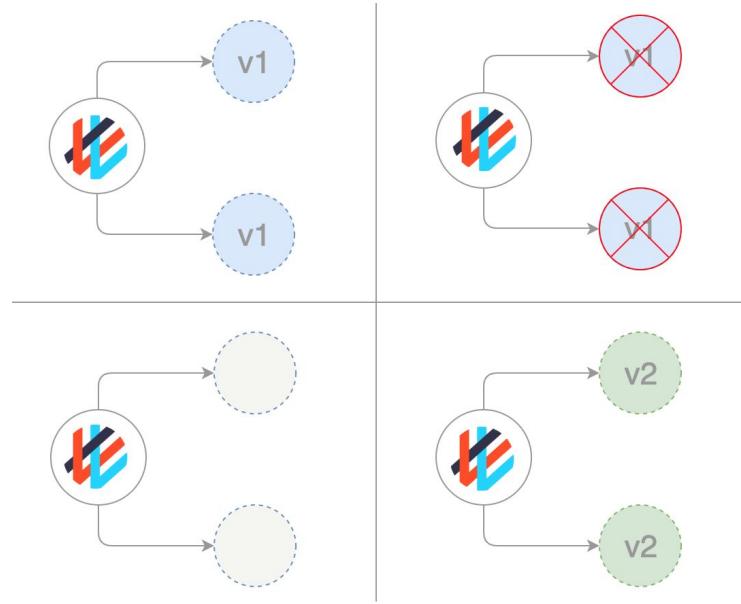
Rolling Deployment



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

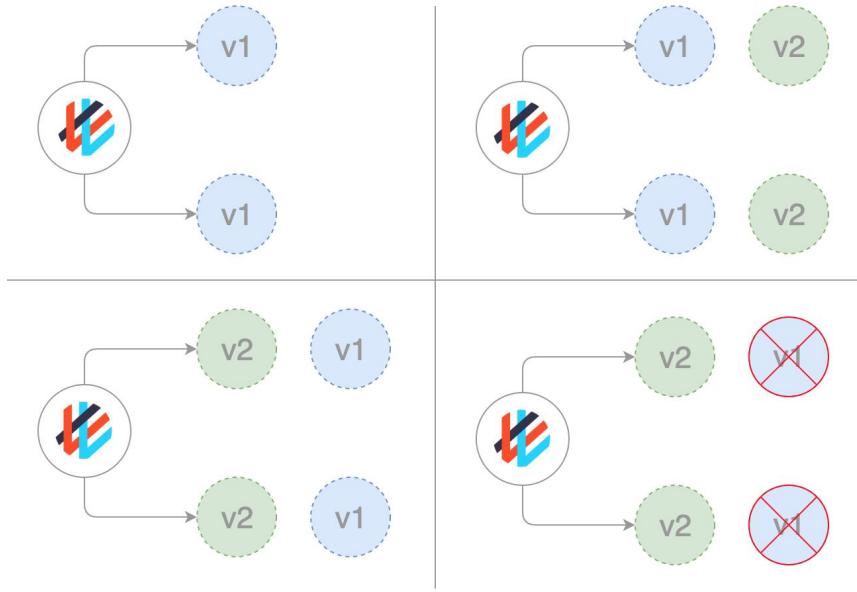
Recreate



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

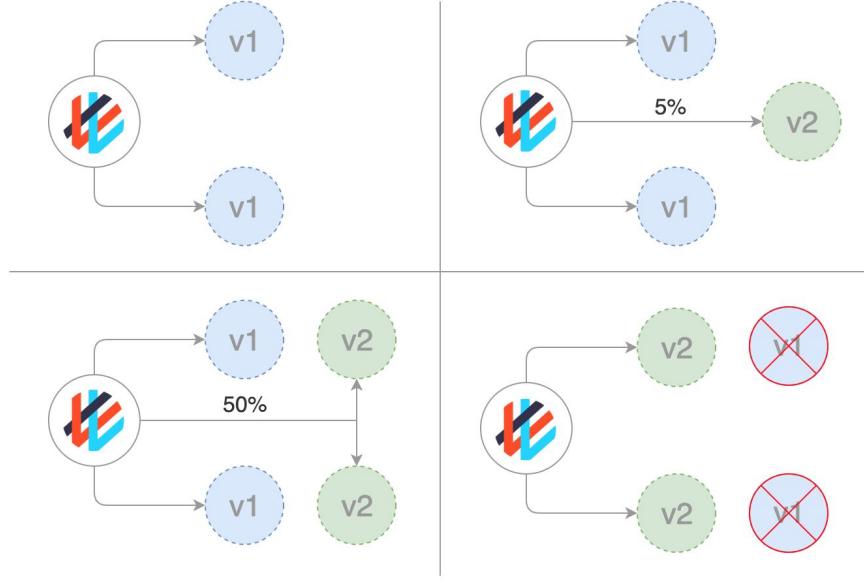
Blue / Green deployments



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

Canary



Reference Pictures:

- [Kubernetes Deployment Strategies](#)

Cloud Native Architecture Fundamentals

Cloud Native Architecture

Cloud Native Architecture Fundamentals

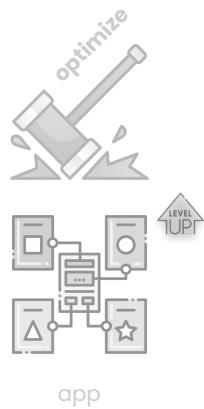
Cloud Native Architecture Fundamentals (Cont.)

- **cloud native architecture** is to optimize your software for



Cloud Native Architecture Fundamentals (Cont.)

- **cloud native architecture** is to optimize your software for **cost efficiency**, **reliability** and **faster time-to-market**



Cost efficiency



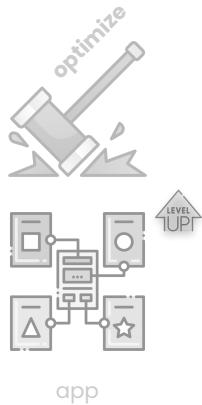
Reliability



Faster time-to-market
(ttm)

Cloud Native Architecture Fundamentals (Cont.)

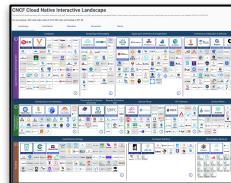
- **cloud native architecture** is to optimize your software for cost efficiency, reliability and faster time-to-market by using a combination of cultural, technological and architectural design patterns.



Culture

	IPF	PROTOTYPES	WATERFALL	AGILE	COLLABORATION	KNOWLEDGE
INDEPENDENCE	Individual	Peerless	Neutral	Collaborative	Experimental	
TEAM	Non-hierarchical	Layered	Hierarchy	Cross-functional	Decentralized	All about
PROCESS	Random	Waterfall	Agile	Scrum/XP	Design Thinking	Lean/Kanban
ARCHITECTURE	Single monolithic	Tight coupling	Decentralized	Microservices	Cloud-native	Platform
INTEGRATION	Integrated	Centralized	Decentralized	Microservices	Cloud-native	Platform
RELAY	Implicit	Periodic	Continuous	Continuous	Continuous	Continuous
POSITIONING	Native	Optimized	Config Project Decentralized	Optimized	Hybrid	Service Mesh
INFRASTRUCTURE	Single host	Multiple servers	Inter-host	Containerized	Cloud-native	Edge computing

Design patterns



Technological



Cost efficiency



Reliability



Faster time-to-market (ttm)

Cloud Native Architecture Fundamentals (Cont.)

- **cloud native architecture** is to optimize your software for cost efficiency, reliability and faster time-to-market by using a combination of cultural, technological and architectural design patterns.



Culture



Design patterns

Technological



Reliability



Cost efficiency



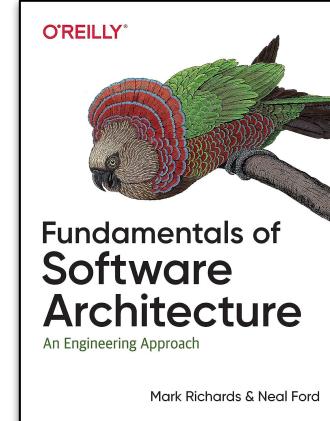
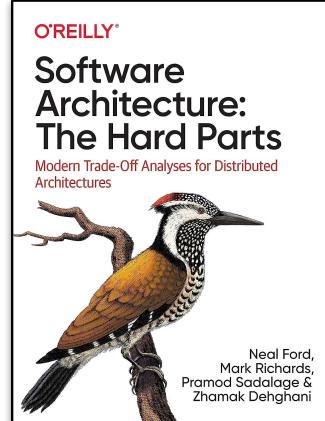
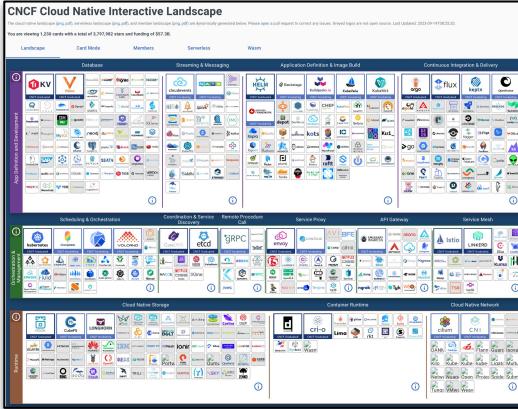
Faster time-to-market
(ttm)

Cloud Native Architecture Fundamentals (Cont.)

- cloud native architecture is to optimize your software for cost efficiency, reliability and faster time-to-market by using a combination of cultural, technological and architectural design patterns.
- The term cloud native can be found in various definitions,

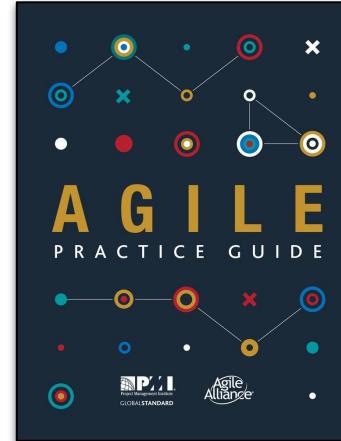
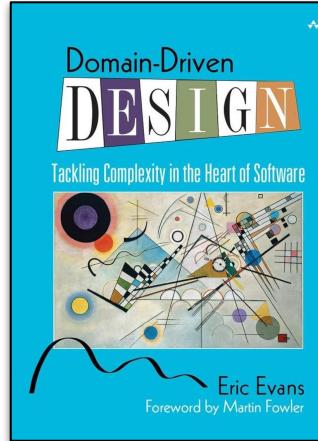
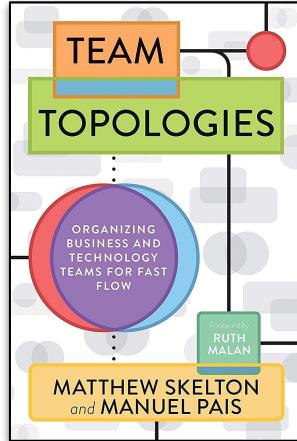
Cloud Native Architecture Fundamentals (Cont.)

- cloud native architecture is to optimize your software for cost efficiency, reliability and faster time-to-market by using a combination of cultural, technological and architectural design patterns.
- The term cloud native can be found in various definitions, some of which have a focus on technologies,



Cloud Native Architecture Fundamentals (Cont.)

- cloud native architecture is to optimize your software for cost efficiency, reliability and faster time-to-market by using a combination of cultural, technological and architectural design patterns.
- The term cloud native can be found in various definitions, some of which have a focus on technologies, while others may **focus on the cultural side of things**.



"Cloud native technologies **empower** organizations

- CNCF Definition v1.0 -
Approved by TOC: 2018-06-11

"Cloud native technologies empower organizations to build and run scalable applications in modern, **dynamic environments** such as public, private, and hybrid clouds.

- CNCF Definition v1.0 -
Approved by TOC: 2018-06-11

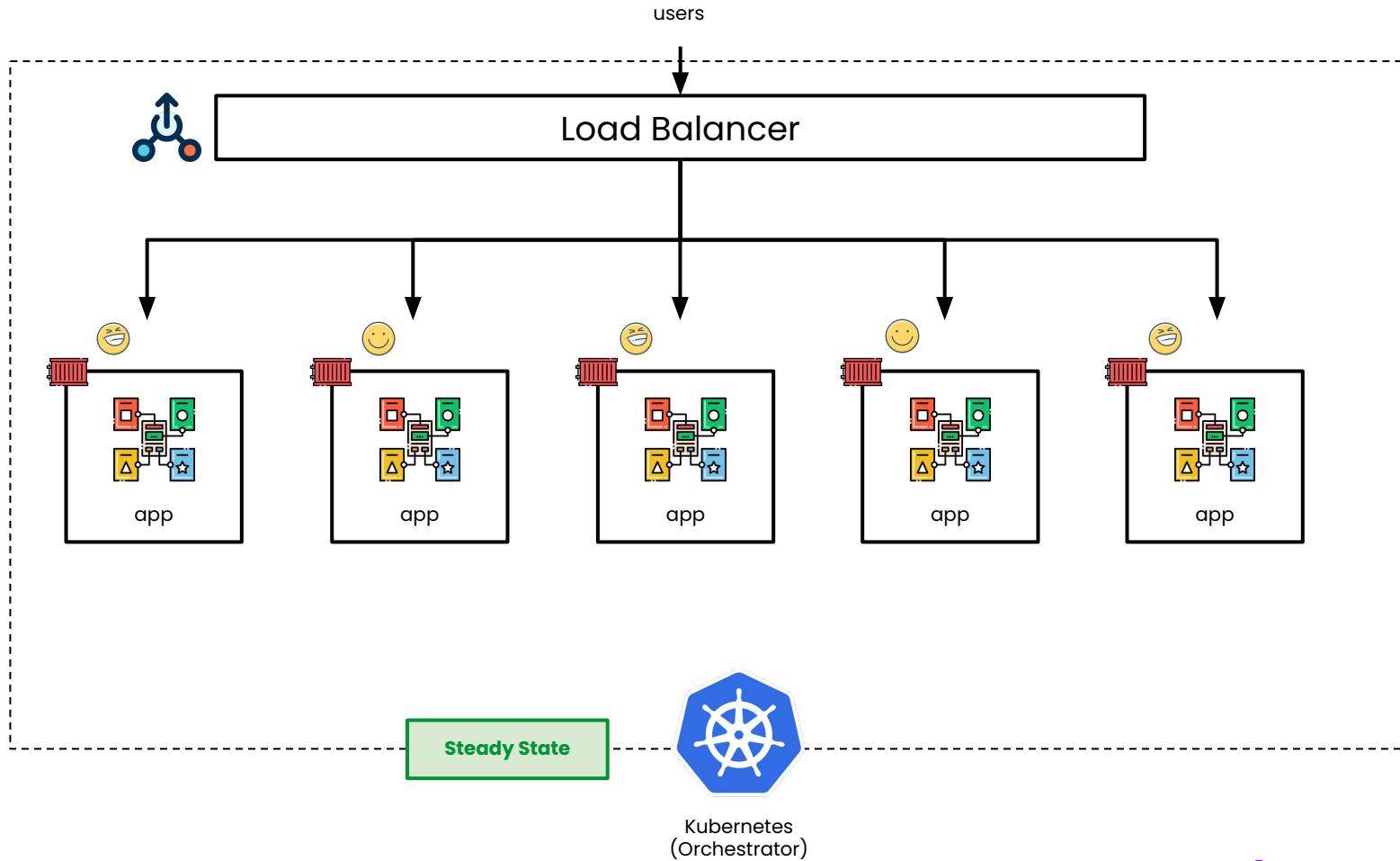
"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach"

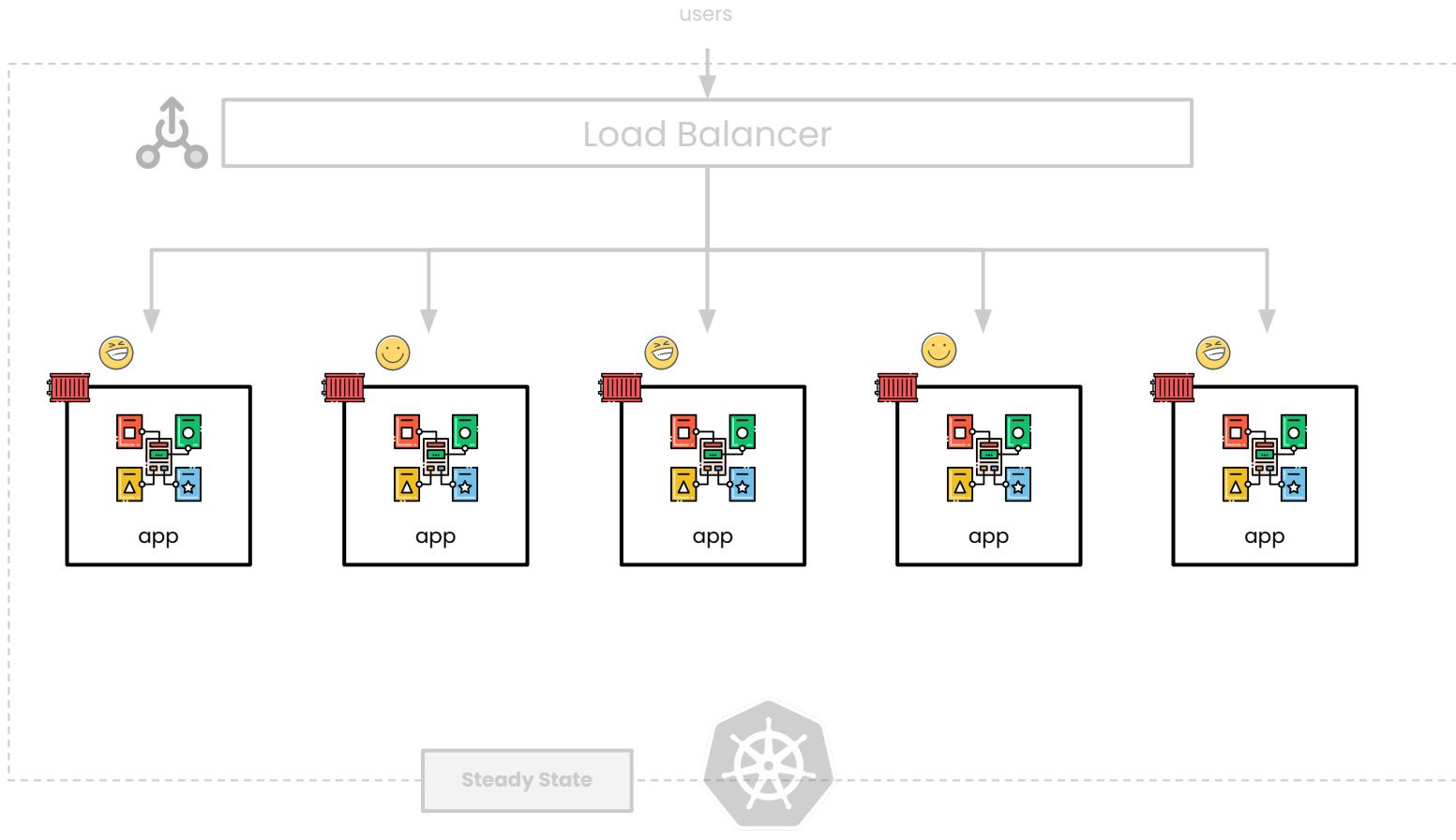
- CNCF Definition v1.0 -
Approved by TOC: 2018-06-11

"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach"

- CNCF Definition v1.0 -
Approved by TOC: 2018-06-11

Let's get back to the **Architect** Overview



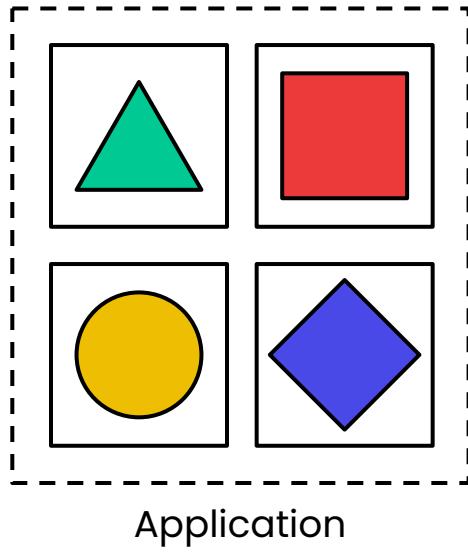


Let's take a look at Application

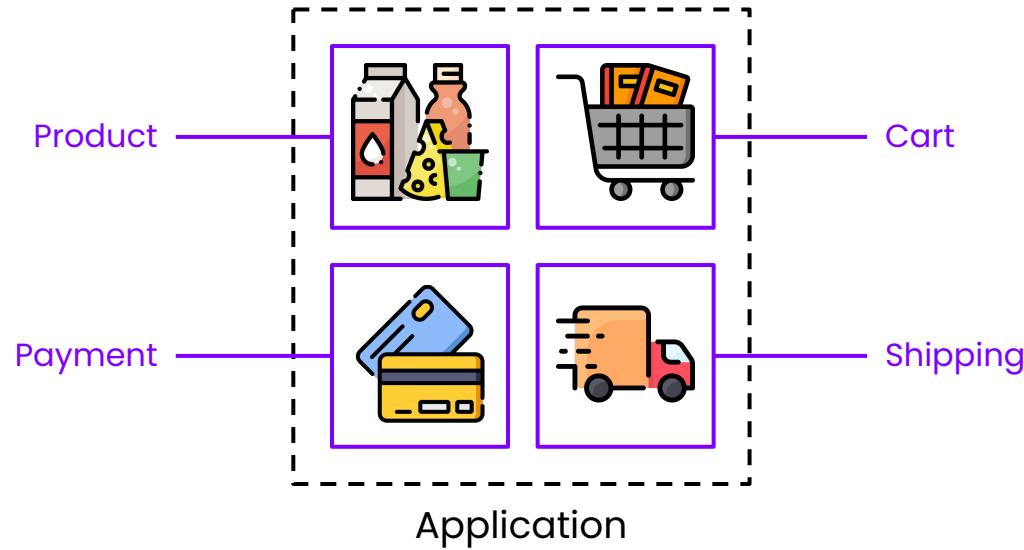
Traditional or legacy applications are usually designed with a monolithic approach

Example with E-Commerce software for an online shop

Example with E-Commerce software for an online shop (Cont.)



Example with E-Commerce software for an online shop (Cont.)



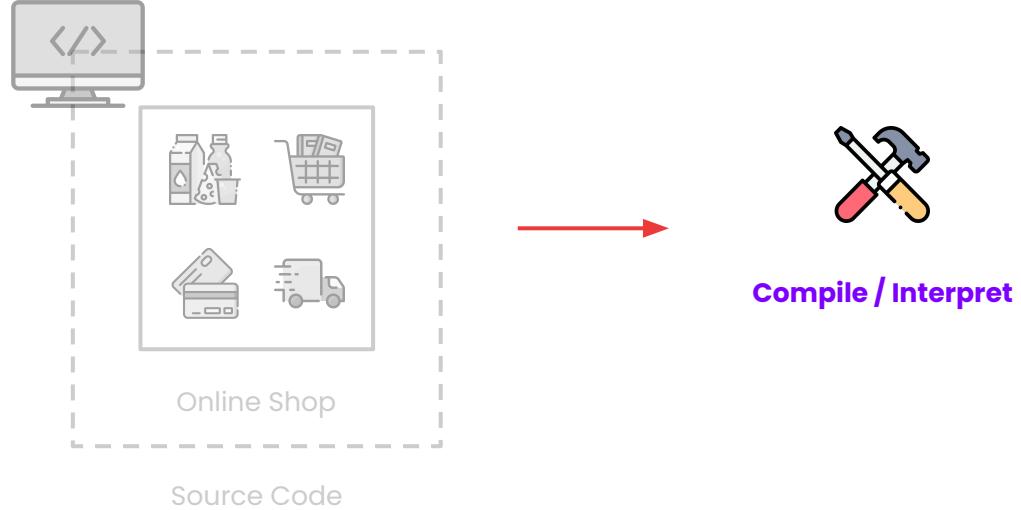
Monolithic Application

- Cloud Native Architecture Fundamentals -

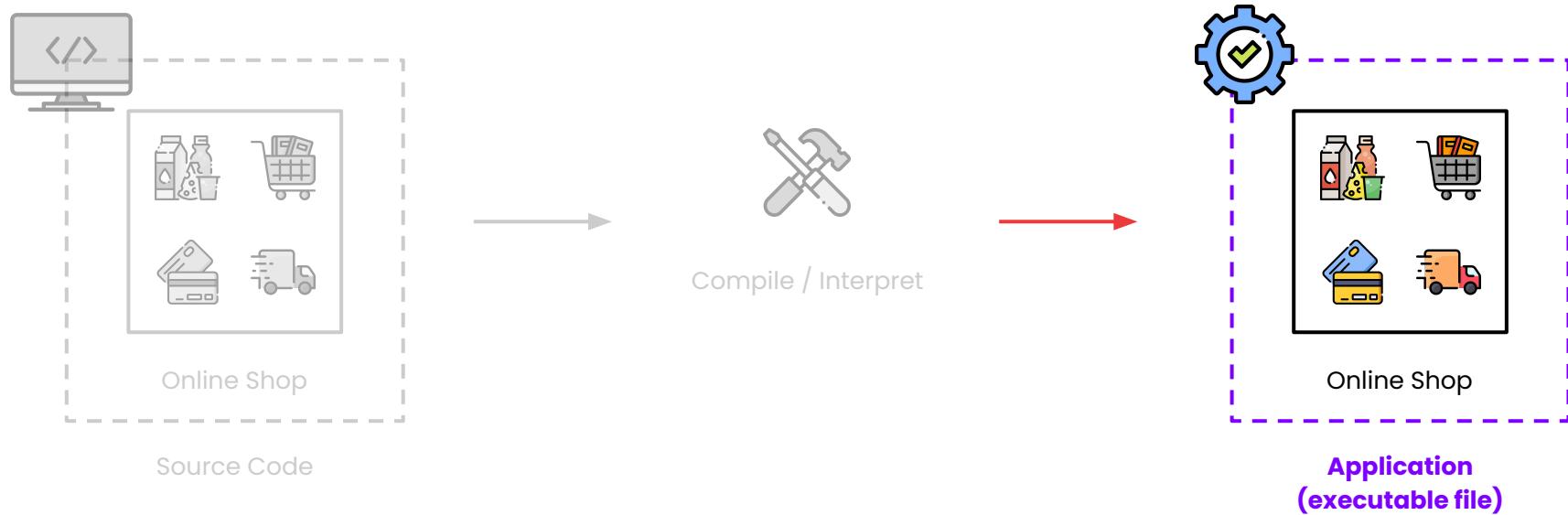
Monolithic Application



Monolithic Application (Cont.)



Monolithic Application (Cont.)

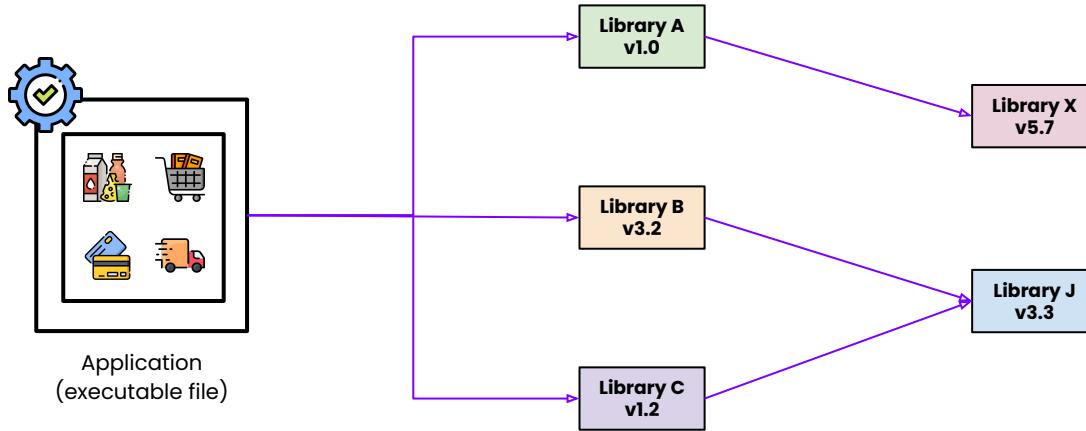


Monolithic Dependencies Management

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จนถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Monolithic Application (Cont.)



Monolithic Containerization

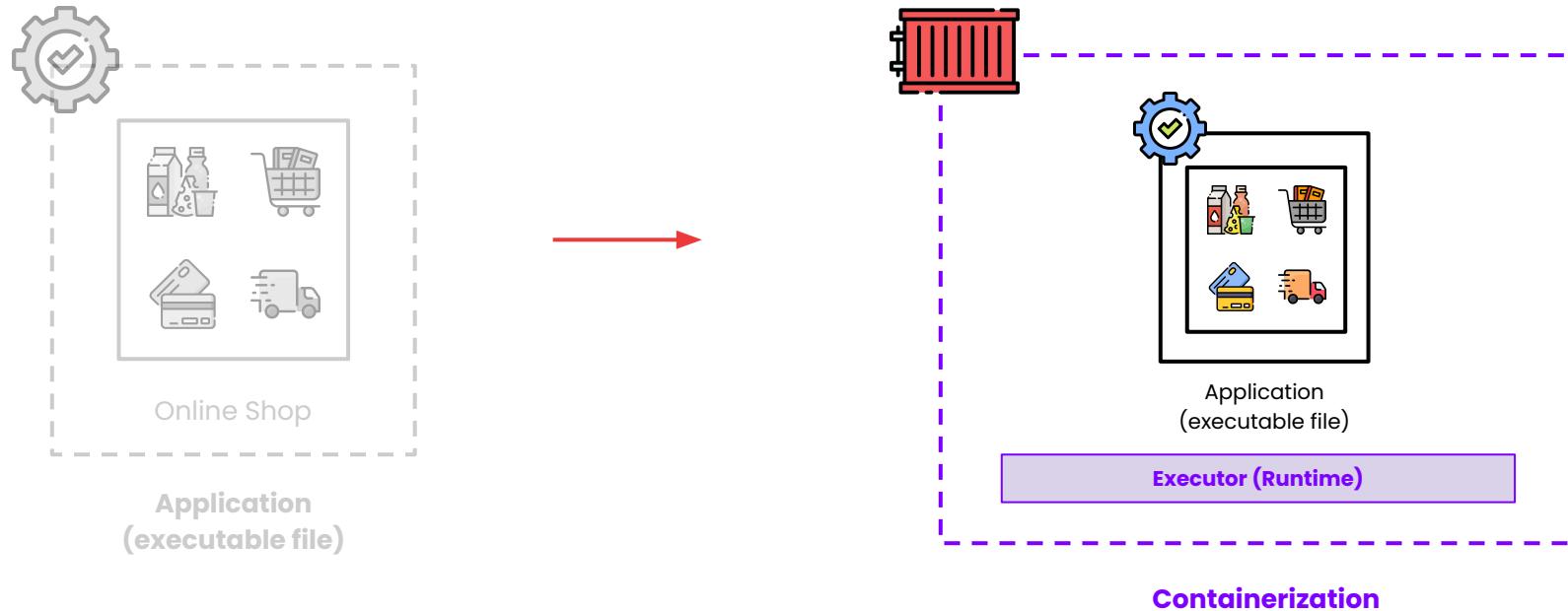
Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Monolithic Application (Cont.)



Monolithic Application (Cont.)



Microservice Application

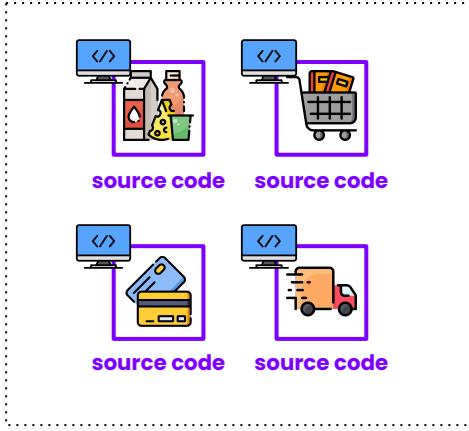
- Cloud Native Architecture Fundamentals -

Microservice Application

Jumpbox®

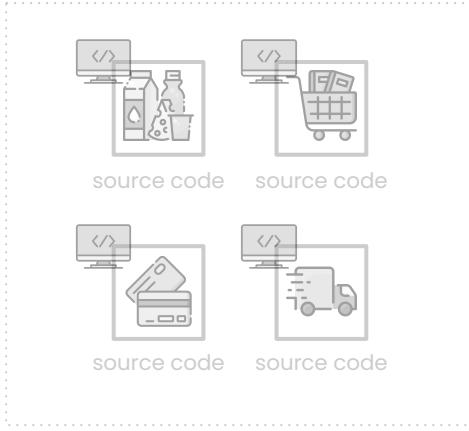
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Microservice Application (Cont.)



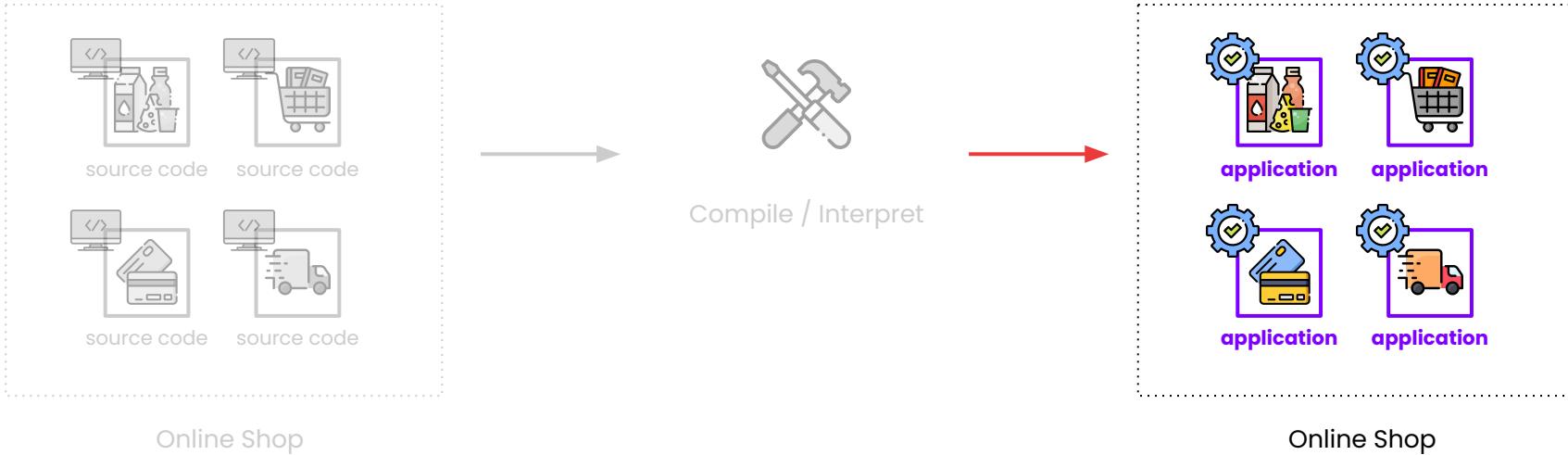
Online Shop

Microservice Application (Cont.)



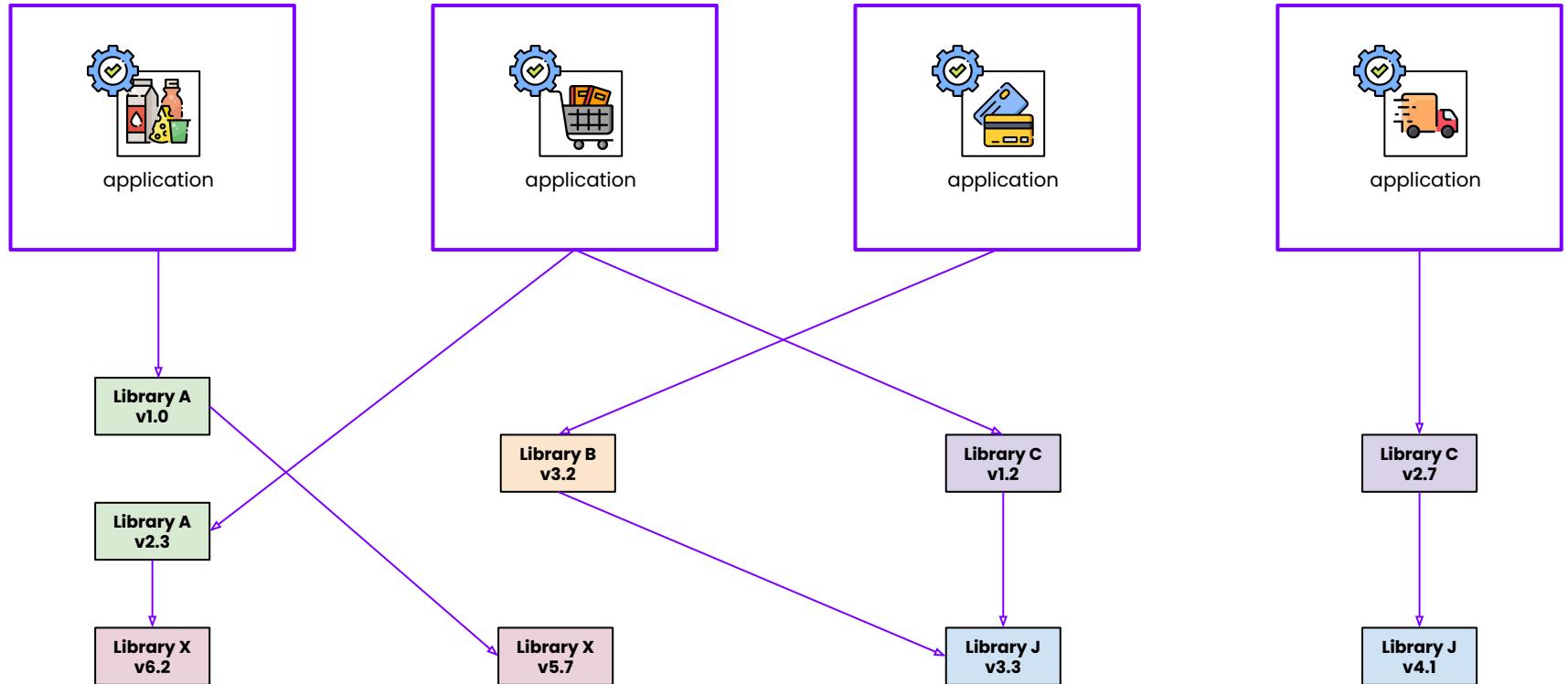
Compile / Interpret

Microservice Application (Cont.)



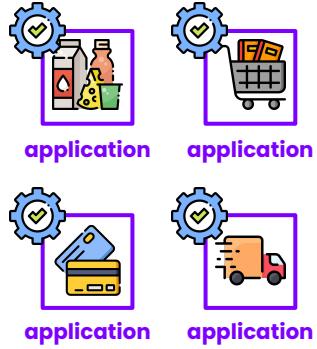
Microservice Dependencies Management

Microservice Application (Cont.)

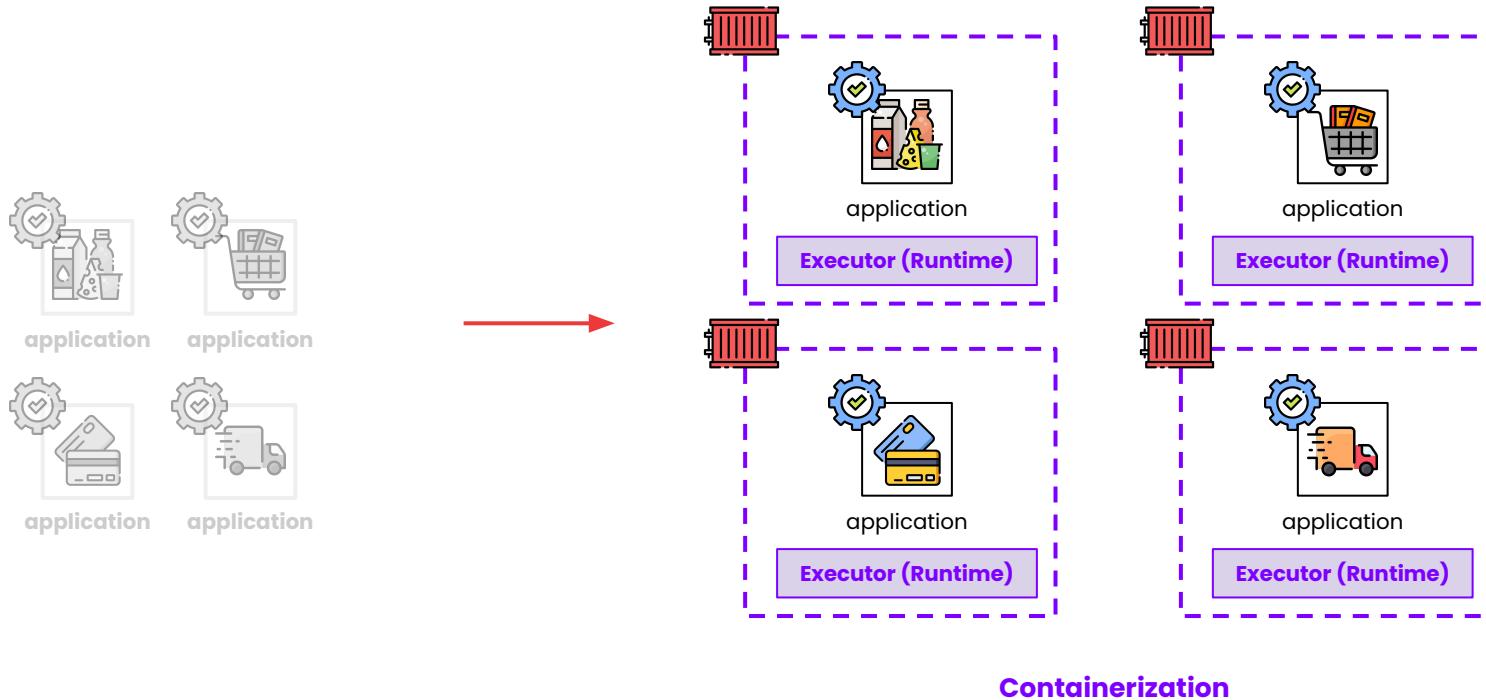


Microservice Containerization

Microservice Application (Cont.)



Microservice Application (Cont.)



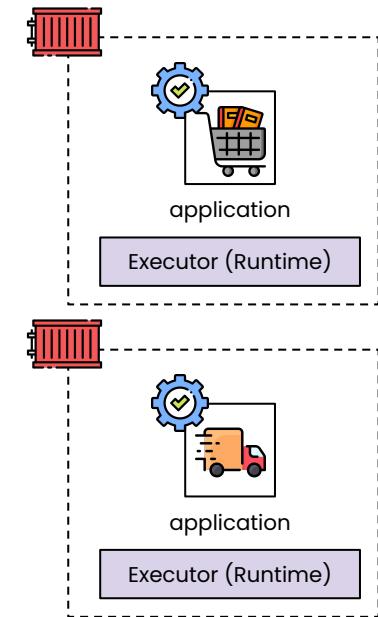
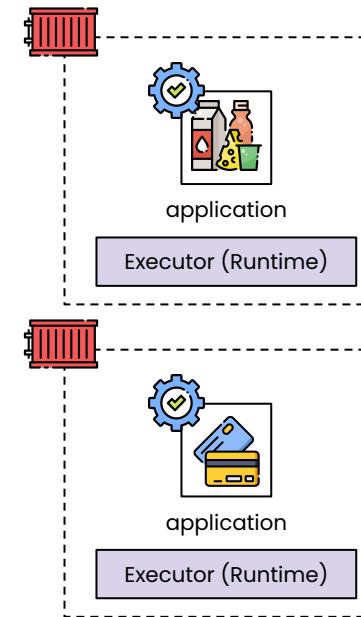
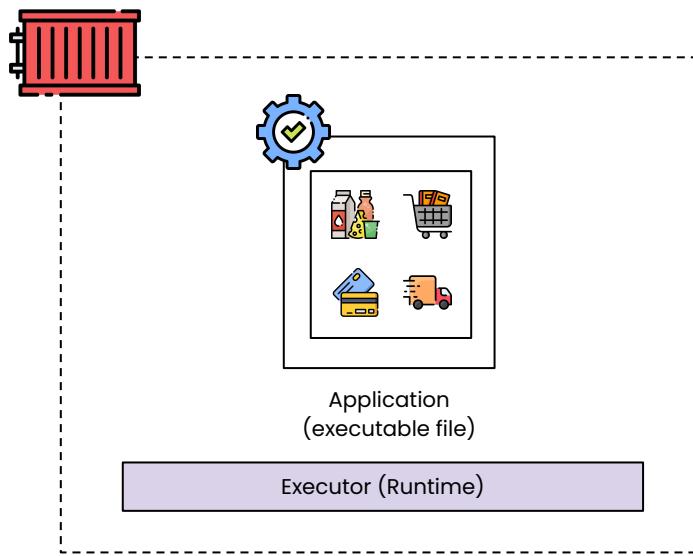
Monolithic vs Microservice

- Cloud Native Architecture Fundamentals -

Monolithic



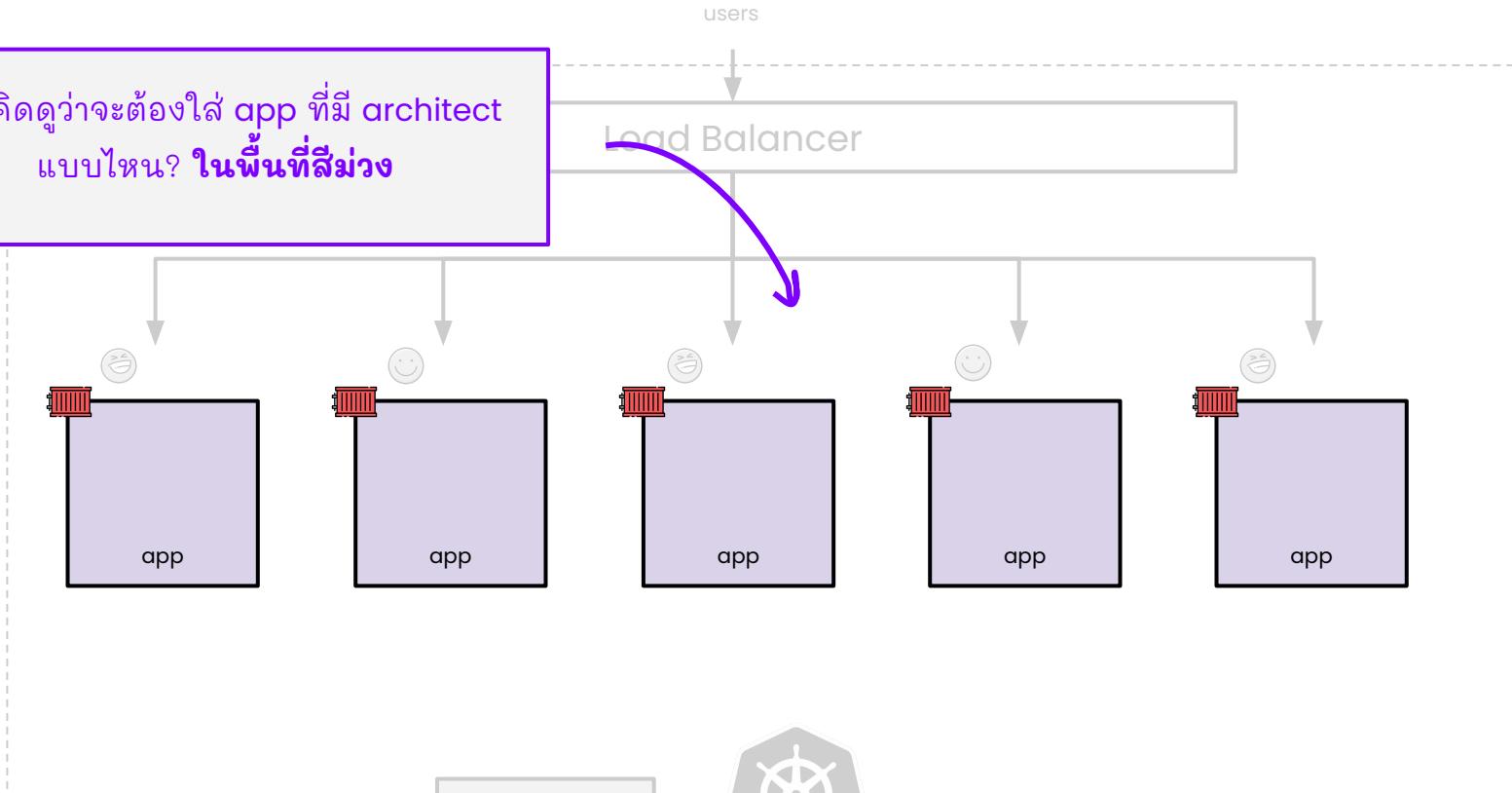
Microservice



This is First criteria to

This is First criteria to
Design **High Level Architecture**

ลองคิดดูว่าจะต้องใส่ app ที่มี architect
แบบไหน? ในพื้นที่สีม่วง



Characteristics of Cloud Native Architecture

- Cloud Native Architecture -

Characteristics of Cloud Native Architecture

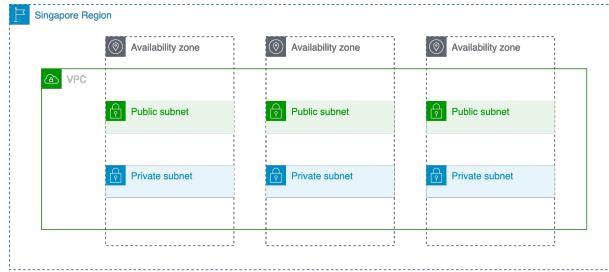
Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation
2. Self healing
3. Scalable
4. (Cost-) Efficient
5. Easy to maintain
6. Secure by default

Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation

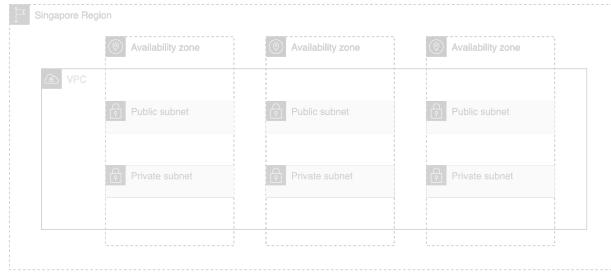
Network Infrastructure



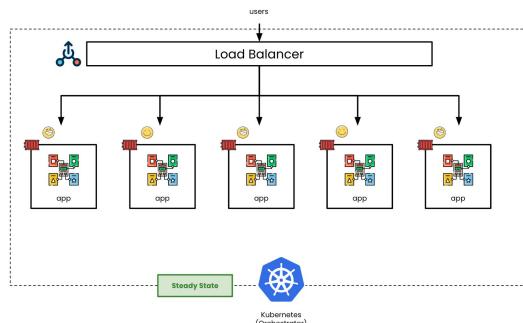
Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation

Network Infrastructure



High Level Infrastructure

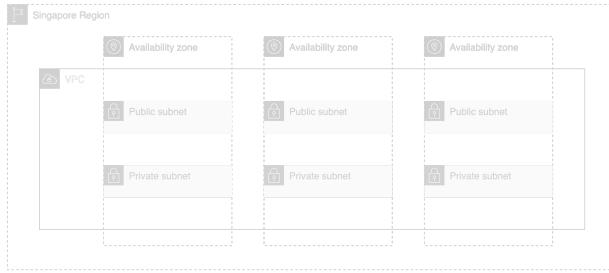


ເຈົ້າຂອງລືຫສິທີ ອຸນໝາດໃຫ້ວຽກແນ້ນເປົ້າການເບີນຮູ້ສັນບຸກຄອນທ່ານັ້ນ ແລະຂອງສຽນລືຫສິທີ ທັນມີໄຟເຫຍພຣີນິກໍສາກາຣນະ ຜູ້ອະນິດ ຈະຖຸກດຳເນີນຕົດຕາມງູ່ໝາຍ

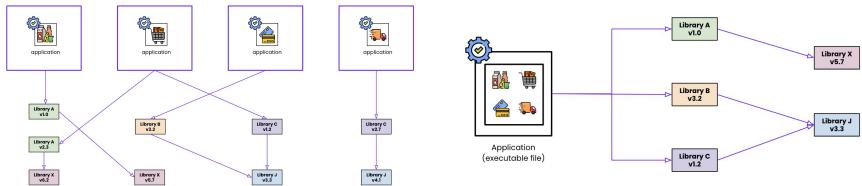
Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation

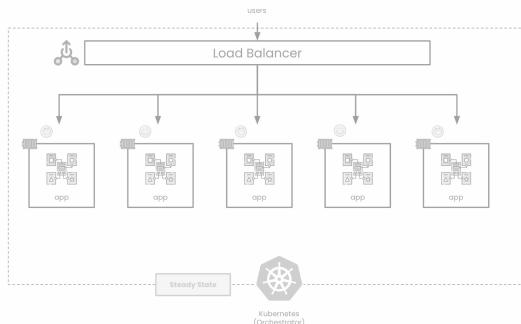
Network Infrastructure



Application Infrastructure



High Level Infrastructure

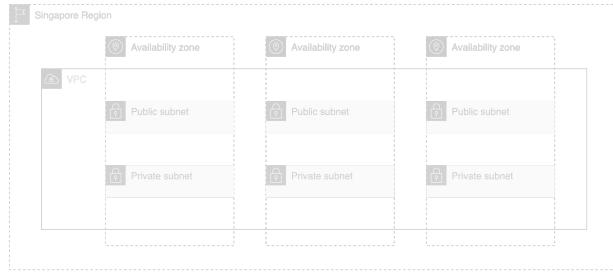


เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อ่านเนต จดถูกต้องกับเงื่อนไขด้านล่าง

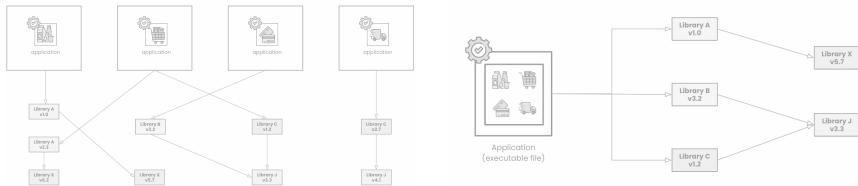
Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation

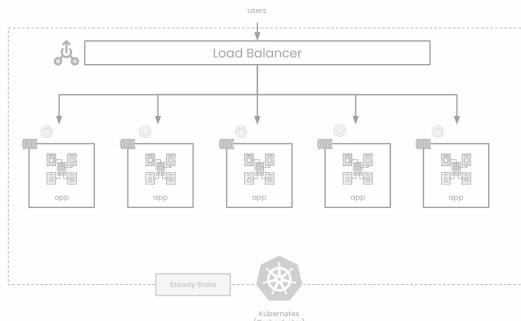
Network Infrastructure



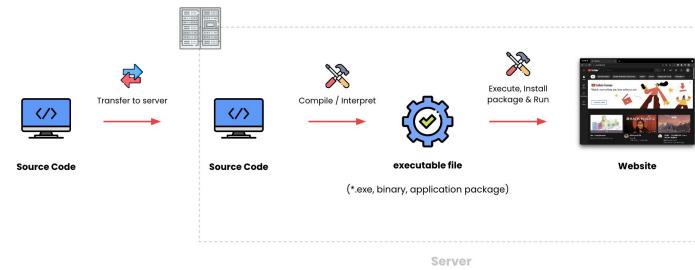
Application Infrastructure



High Level Architecture Infrastructure



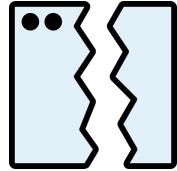
Deployment Process



เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนที่ของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อ่านเนต จดถูกด้วยเงื่อนไขด้านล่าง

Characteristics of Cloud Native Architecture (Cont.)

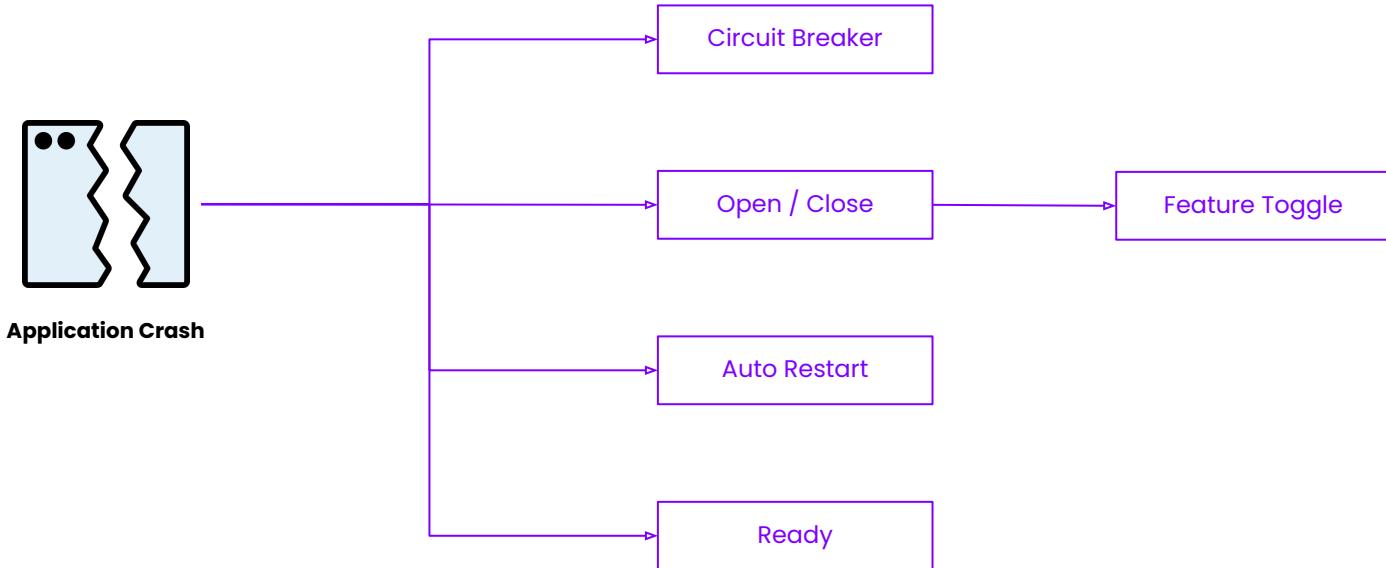
1. High level of automation
2. Self healing



Application Crash

Characteristics of Cloud Native Architecture (Cont.)

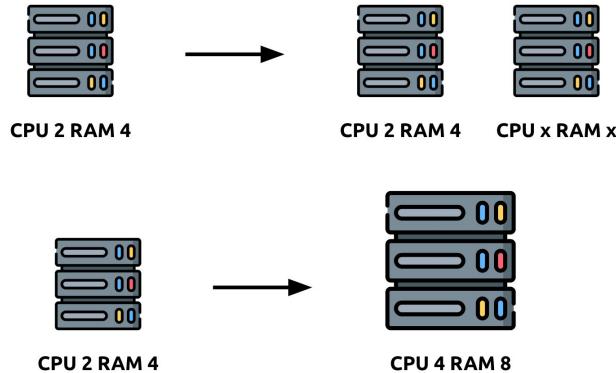
1. High level of automation
2. Self healing



Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation
2. Self healing
3. Scalable

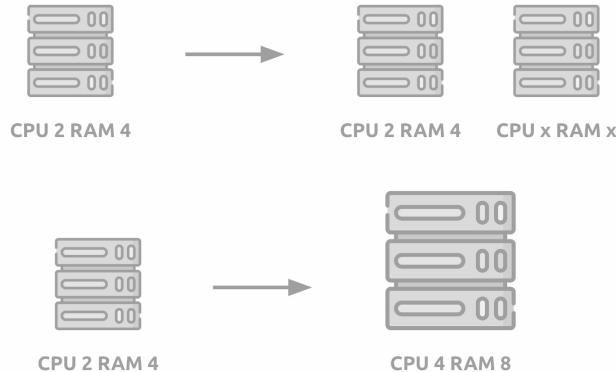
Scalable Infrastructure



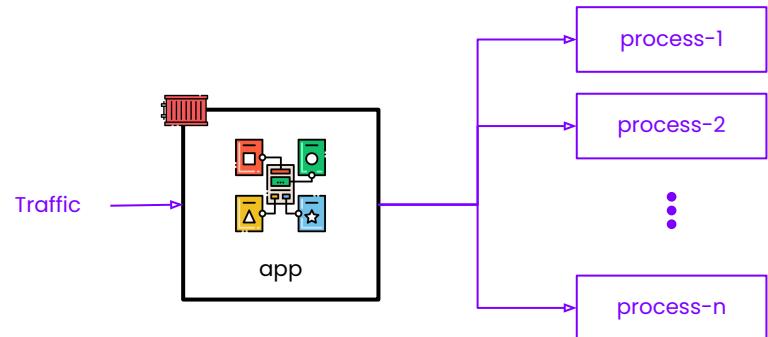
Characteristics of Cloud Native Architecture (Cont.)

1. High level of automation
2. Self healing
3. Scalable

Scalable Infrastructure

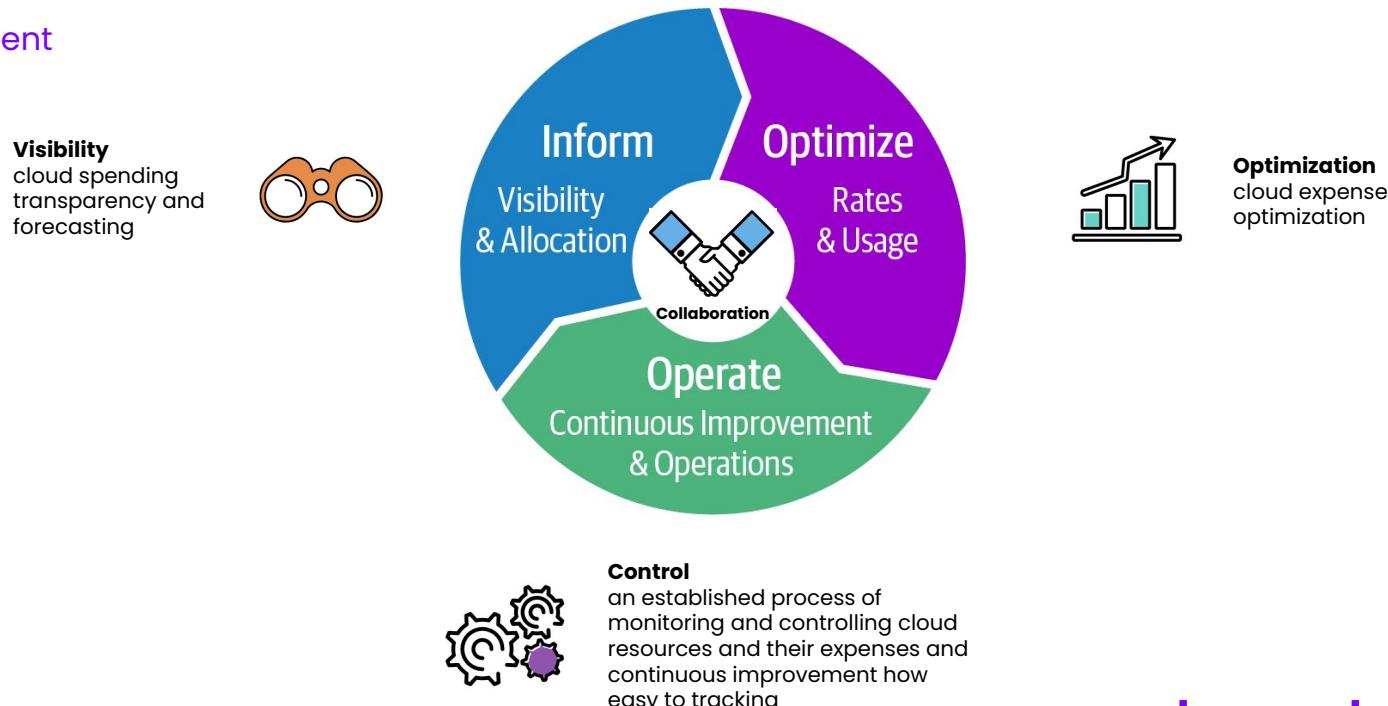


Scalable Application



Characteristics of Cloud Native Architecture (Cont.)

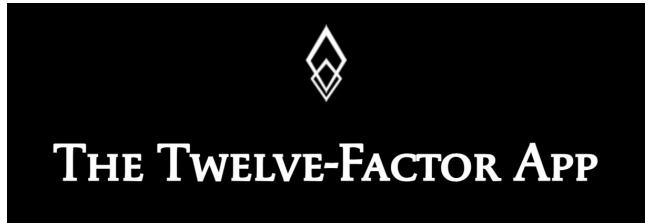
1. High level of automation
2. Self healing
3. Scalable
4. (Cost-) Efficient



Characteristics of Cloud Native Architecture (Cont.)

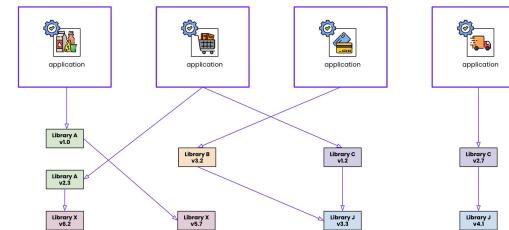
1. High level of automation
2. Self healing
3. Scalable
4. (Cost-) Efficient
5. Easy to maintain
6. Secure by default

12 Factor



<https://12factor.net/>

Architecture Scalable



Zero trust computing



"What doesn't kill you,
will make you stronger"

-The Quarry-





Jumpbox®

Tech Passion | Sharing | Society

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่ทำการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเมตตา จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

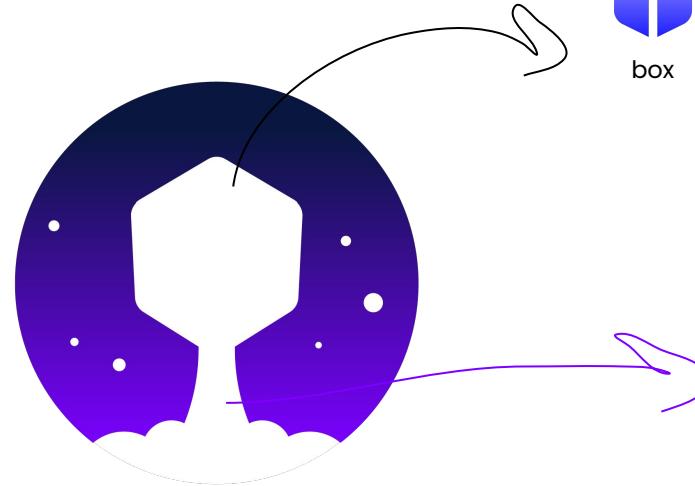


Jumpbox®

Tech Passion | Sharing | Society

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®



Jumpbox®

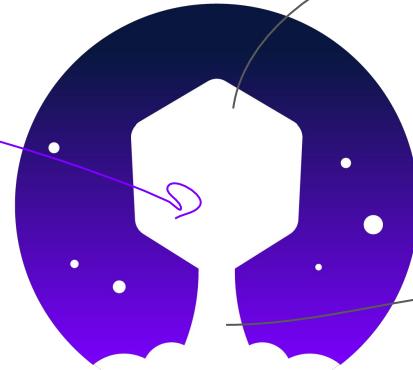
Tech Passion | Sharing | Society

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่ทำการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จนถูกดำเนินคดีตามกฎหมาย

Jumpbox®



We



box



Jet

Jumpbox®

Tech Passion | Sharing | Society

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่ทำการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จนถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Jumpbox

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Jumpbox = กล่องกระโดด

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้รับสั่นนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

Jumpbox = กล่องกระโดด
(การเติบโตอย่างก้าวกระโดด)

"เราเชื่อว่า การเรียนรู้ทำให้ชีวิตคุณดีขึ้น"

-Jumpbox Team-

ช่วยส่ง **Feedback** ให้กับผู้สอน เพื่อเราได้พัฒนาตัวเองให้ดีขึ้นครับ... 

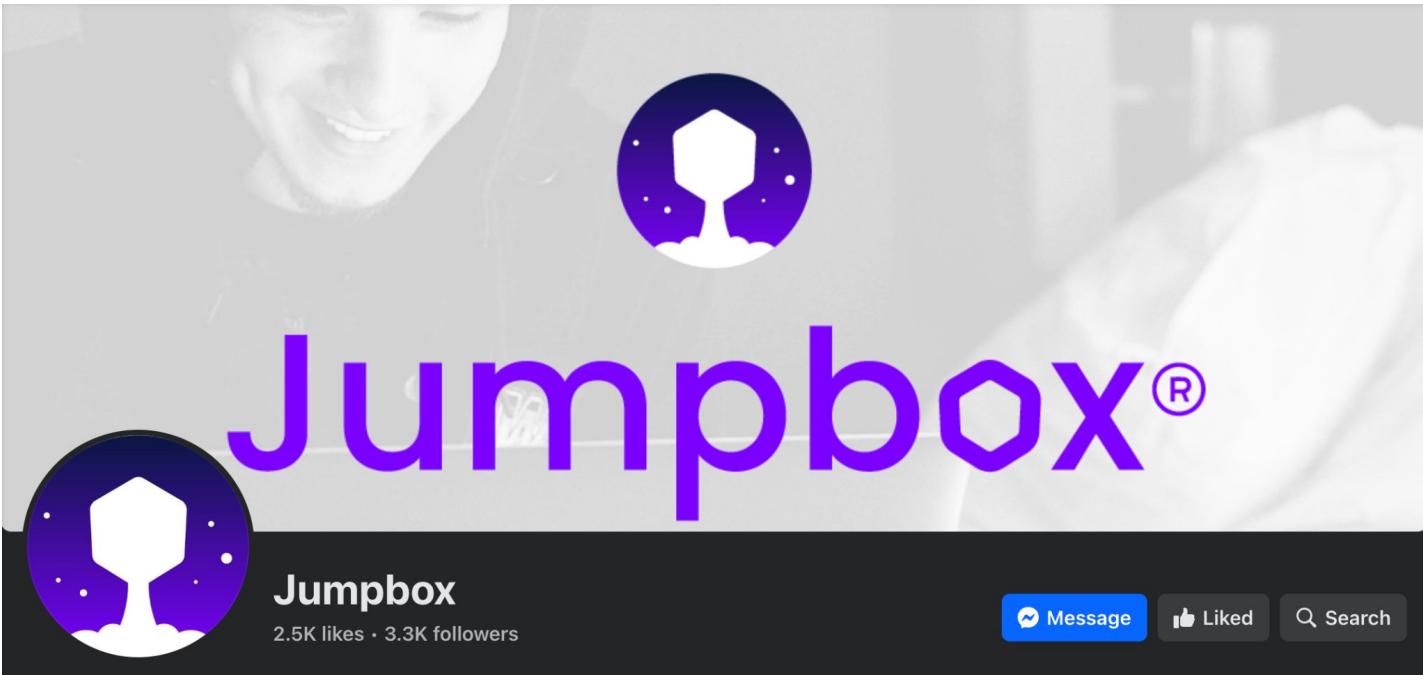


[Feedback Link](#)

เจ้าของลิสต์ที่ให้ไว้ล้วนถูกคัดท่านนั้น และขอสงวนลิสต์ที่หันมามีให้เชยแพรในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

facebook



<https://www.facebook.com/jumpbox.academy>

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง เดินทางโดยไม่มีความยุติธรรม

Jumpbox®



Jumpbox

@jumpbox.academy • 1.92K subscribers • 30 videos

More about this channel ...[more](#)

<https://www.youtube.com/@jumpbox.academy>

Contact Us



Jumpbox



@jumpbox



admin@jumpbox.co



063-245-2168 (JoJo)

062-796-1559 (Beau)





Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่มีการเรียบเรียงด้วยบุคคลท่านนี้ และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Addition

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สอดคล้องกับการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

1. Application Characteristic

- Basic Observability -

- Program Process
- Application Building
- Application Output
- Application Package Manager

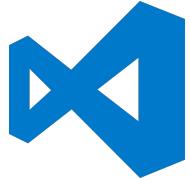
Program Process

- 1. Application Characteristic -

Software Engineer

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นแน่นอน จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®



Code Editor

Software Engineer

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®



Code Editor



Coding

Software Engineer



Code Editor



Coding

Software Engineer



Problem Solving



Code Editor

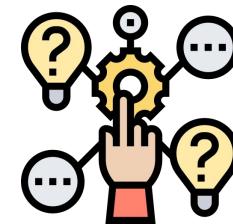


Coding

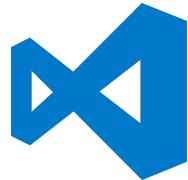
Software Engineer



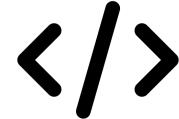
Problem Solving



Programming



Code Editor



Coding

Software Engineer



Problem Solving



Programming

Software Engineer

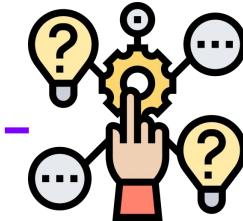


Problem Solving

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ส่วนหนึ่งของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดิน จดถูกต้องเป็นคัดลอกกฎหมาย

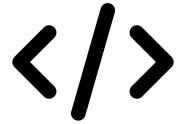


Coding

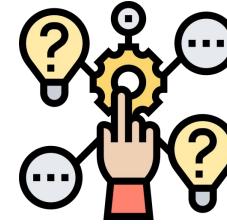


Programming

Jumpbox®



Coding



Programming



Problem Solving



Program

collection of instructions



Program

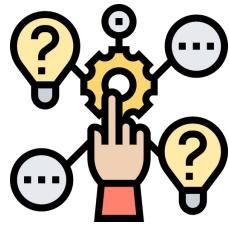
collection of instructions



Program



collection of instructions



Program



collection of instructions



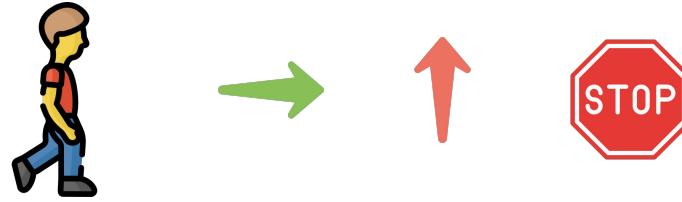
Program



Program

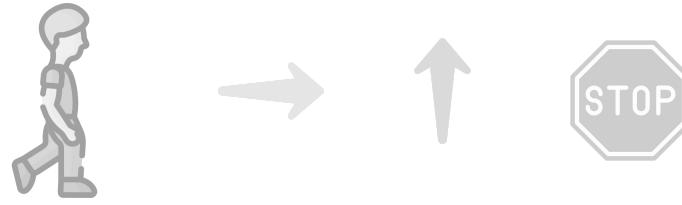


collection of instructions
executed by Human





collection of instructions
executed by Human

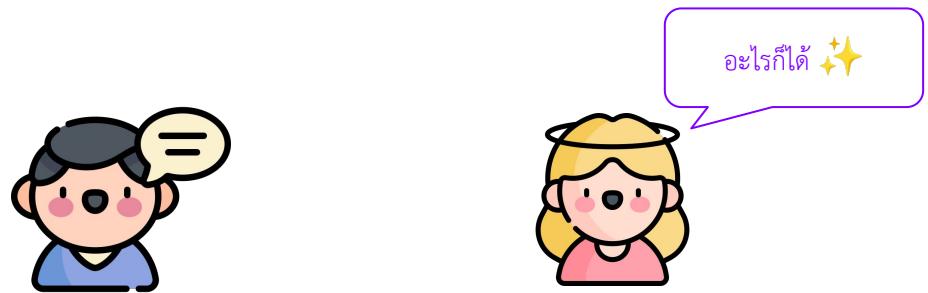
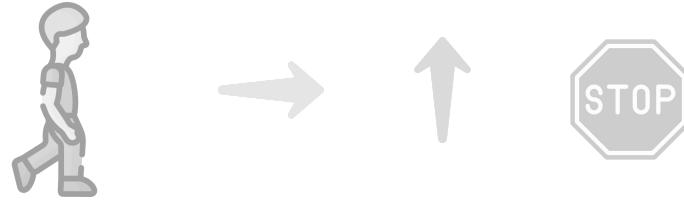


ເຢັ້ນນີ້ກິນອະໄຮດີ?



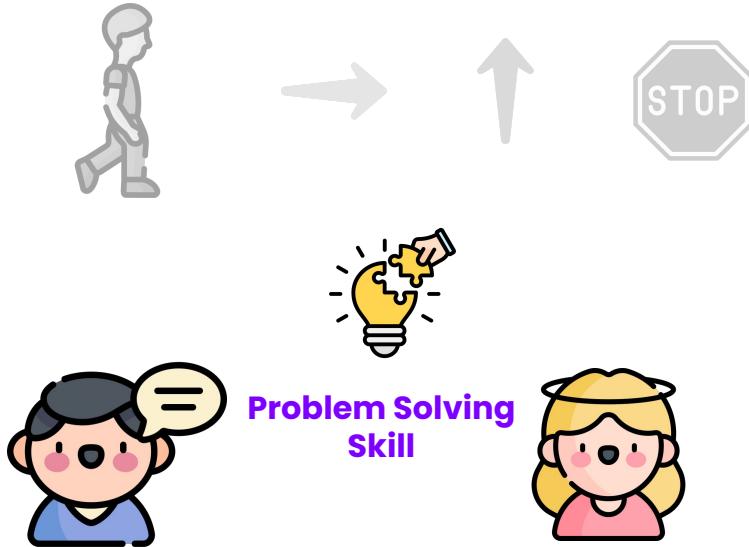


collection of instructions
executed by Human





collection of instructions
executed by Human



Program vs Application

Program vs Application (Cont.)

(Computer) Program

Program vs Application (Cont.)

(Computer) Program

collection of instructions executed by computer.

Program vs Application (Cont.)

(Computer) Program

collection of instructions executed by computer.

Application

Program vs Application (Cont.)

(Computer) Program

collection of instructions executed by computer.

Application

programs designed for end users.

Reference:

- [Program and Application](#)

Program Process

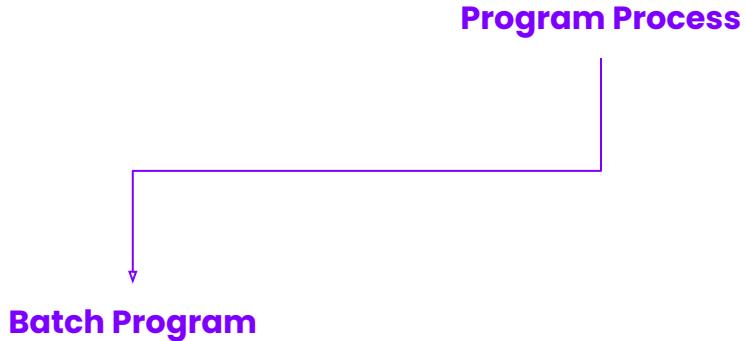
Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

Program Process (Cont.)

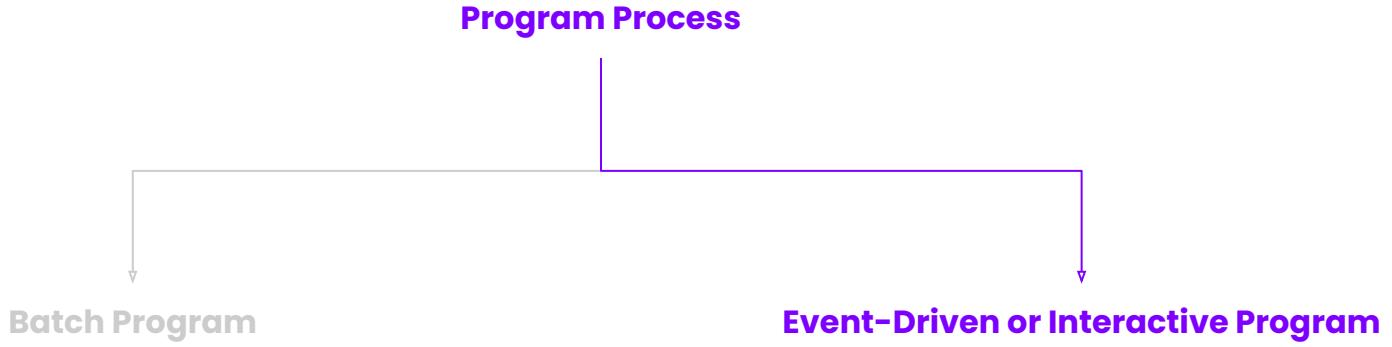
Program Process

Program Process (Cont.)



- These are designed to run to completion without needing user interaction or to wait for external events.
- They execute a series of instructions or tasks and then terminate.

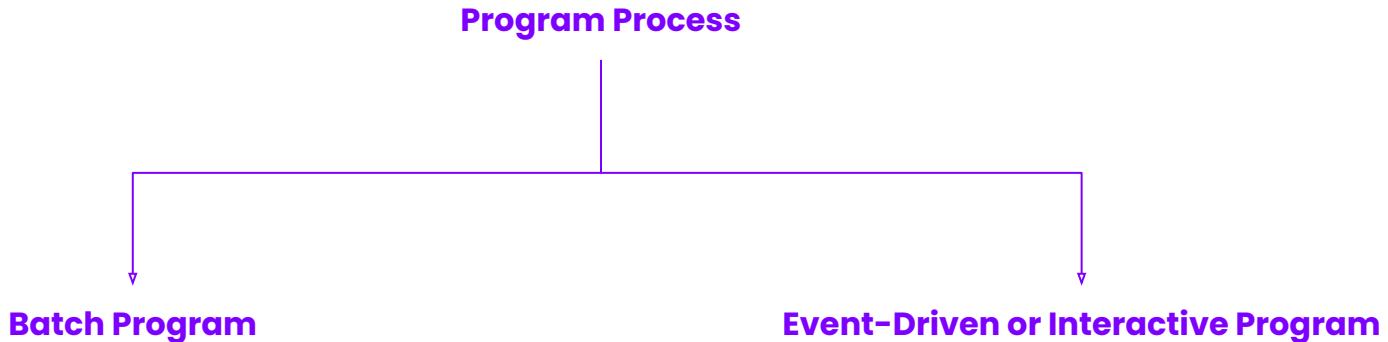
Program Process (Cont.)



- These are designed to run to completion without needing user interaction or to wait for external events.
- They execute a series of instructions or tasks and then terminate.

- These are designed to run indefinitely, waiting for events to occur.
- The program reacts to inputs or triggers like mouse clicks, keyboard presses, network requests, etc.
- The program remains active until it is explicitly terminated by the user or by an internal condition.

Program Process (Cont.)



- These are designed to run to completion without needing user interaction or to wait for external events.
- They execute a series of instructions or tasks and then terminate.
- These are designed to run indefinitely, waiting for events to occur.
- The program reacts to inputs or triggers like mouse clicks, keyboard presses, network requests, etc.
- The program remains active until it is explicitly terminated by the user or by an internal condition.

Activity Time !!

- Kata -

Batch or Event Driven

API?

Batch or Event Driven

API

Job?

Batch or Event Driven

API

Job

Batch or Event Driven

Schedule?

API

Job

Batch or Event Driven

Schedule

Web App?

API

Job

Batch or Event Driven

Schedule

Web App?

Application Building

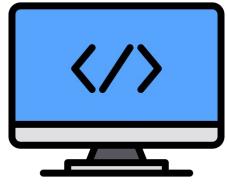
- 1. Application Characteristic -

Application Building

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ลงทะเบียน จะถูกดำเนินคดีตามกฎหมาย

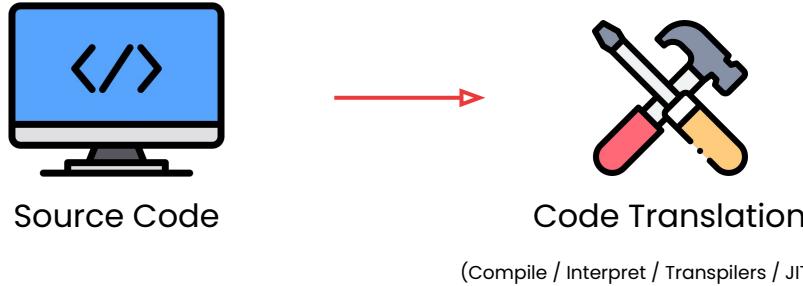
Jumpbox®

Application Building (Cont.)

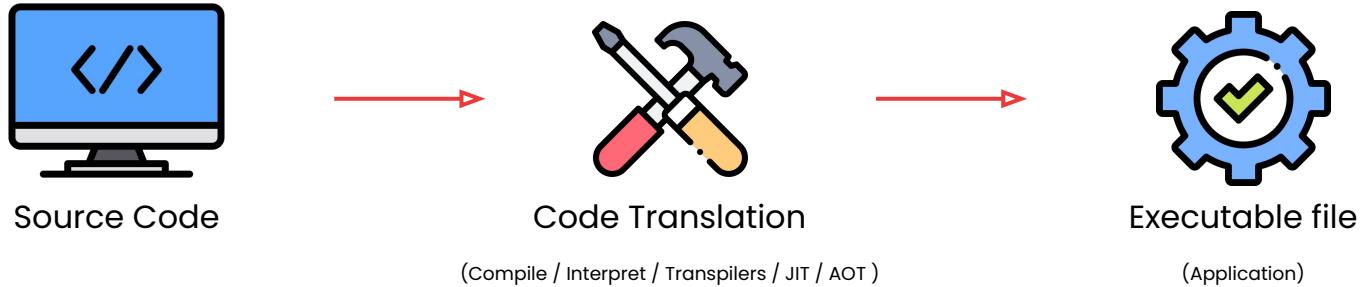


Source Code

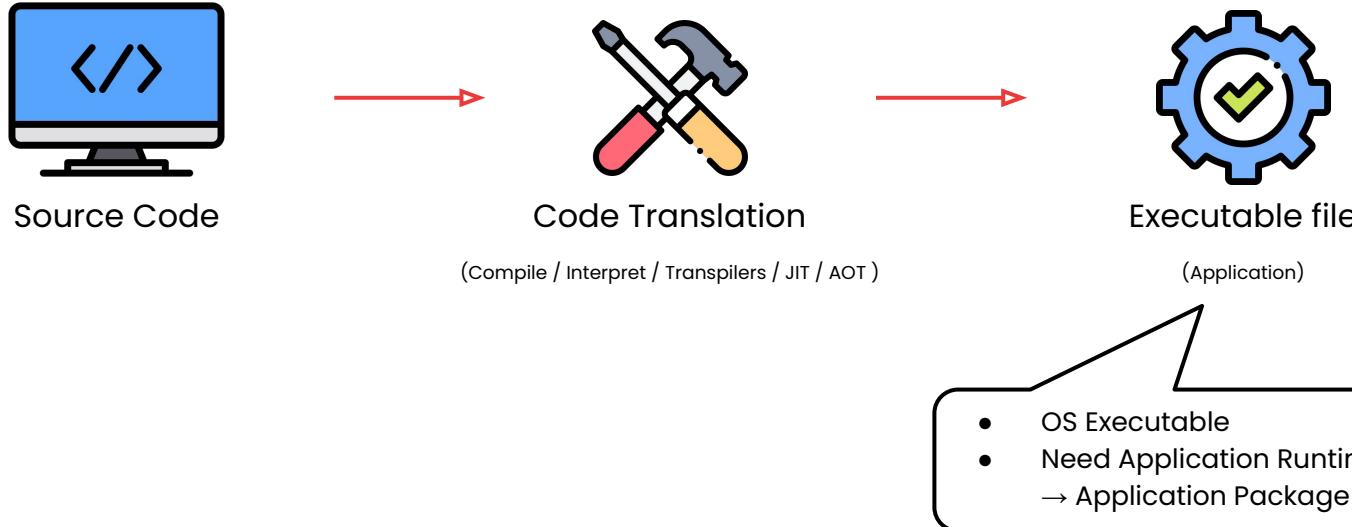
Application Building (Cont.)



Application Building (Cont.)



Application Building (Cont.)



Executable file: OS Executable (Cont.)

Executable file: OS Executable (Cont.)

System Programming Language:

Executable file: OS Executable (Cont.)

System Programming Language:



Executable file

(Application)

Executable file: OS Executable (Cont.)

System Programming Language:



Executable file

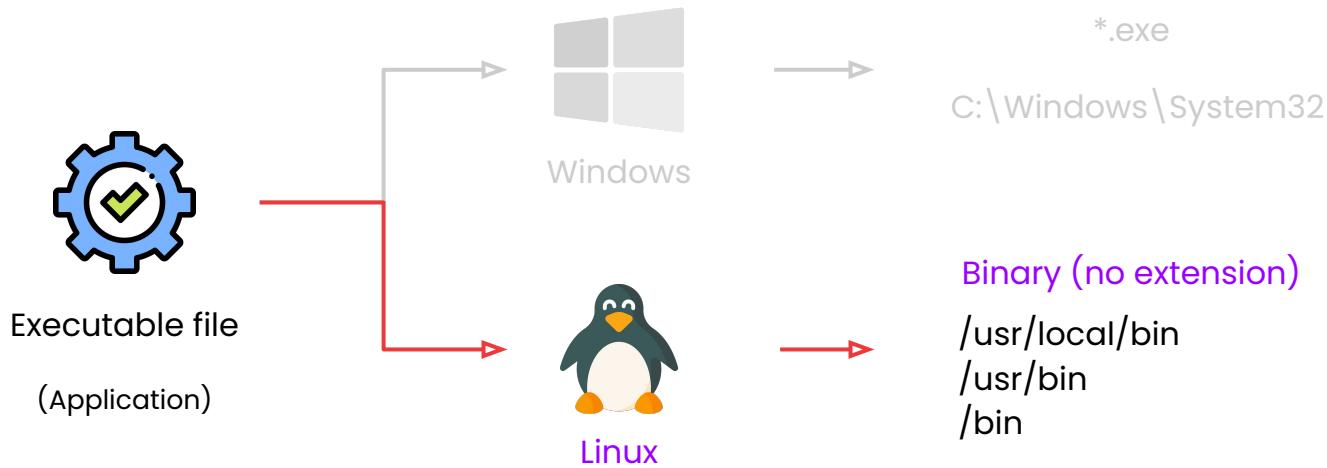
(Application)

*.exe

C:\Windows\System32

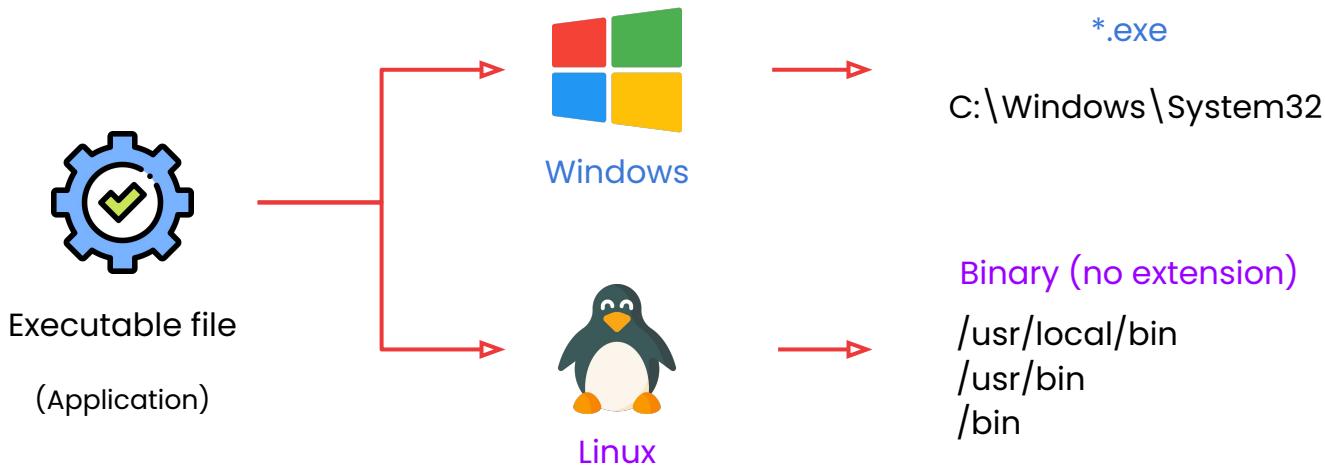
Executable file: OS Executable (Cont.)

System Programming Language:



Executable file: OS Executable (Cont.)

System Programming Language:



Executable file: Application Package

Executable file: Application Package (Cont.)

High-level programming language:

Executable file: Application Package (Cont.)

High-level programming language:



Application Package

Executable file: Application Package (Cont.)

Application

High-level programming language:



Application Package

Executable file: Application Package (Cont.)

High-level programming language:

Application

Package



Application Package

Executable file: Application Package (Cont.)

High-level programming language:



Application Package

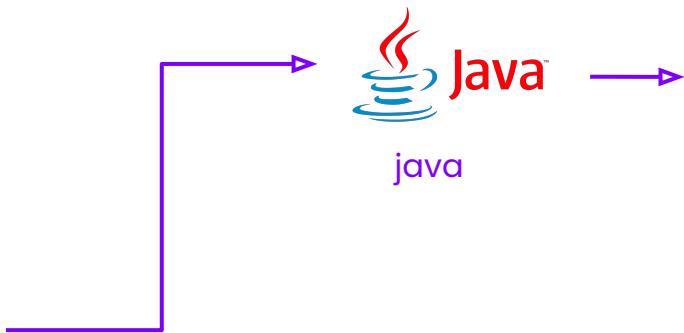
Application	
Package	Runtime

Executable file: Application Package (Cont.)

High-level programming language:



Application Package



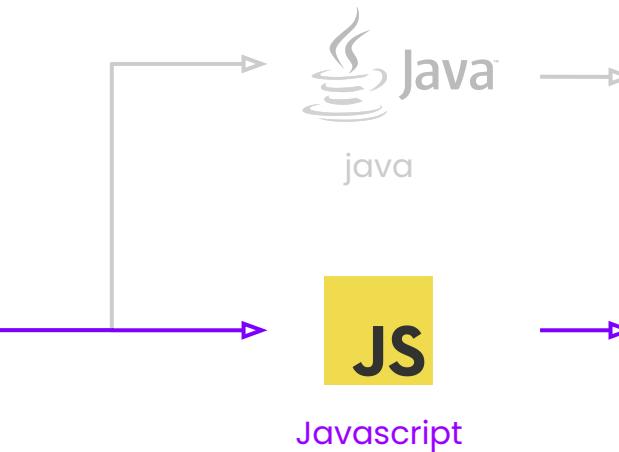
Application	
Package	Runtime
*.jar, *.war	JDK (JRE and JVM)

Executable file: Application Package (Cont.)

High-level programming language:



Application Package



Application

Package

Runtime

*.jar, *.war

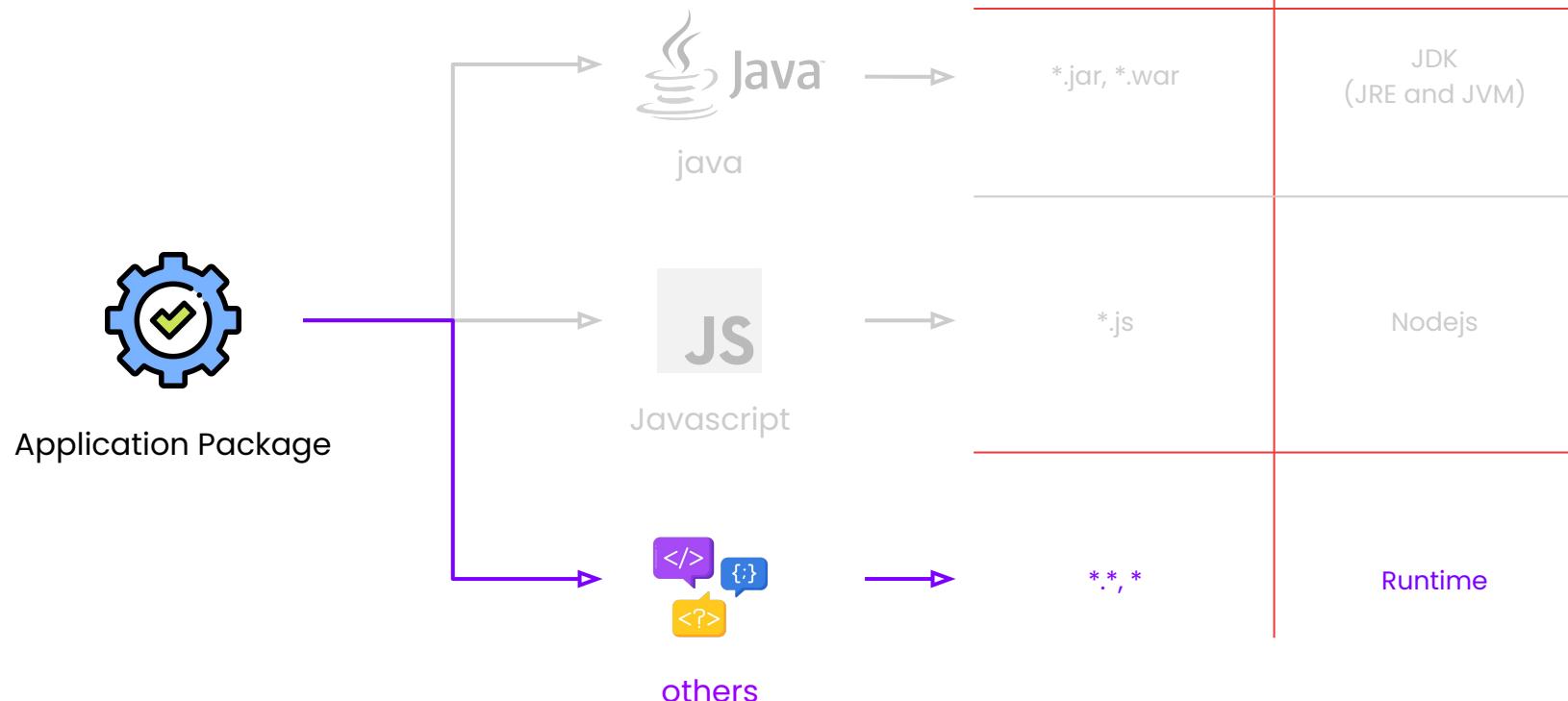
JDK
(JRE and JVM)

*.js

Nodejs

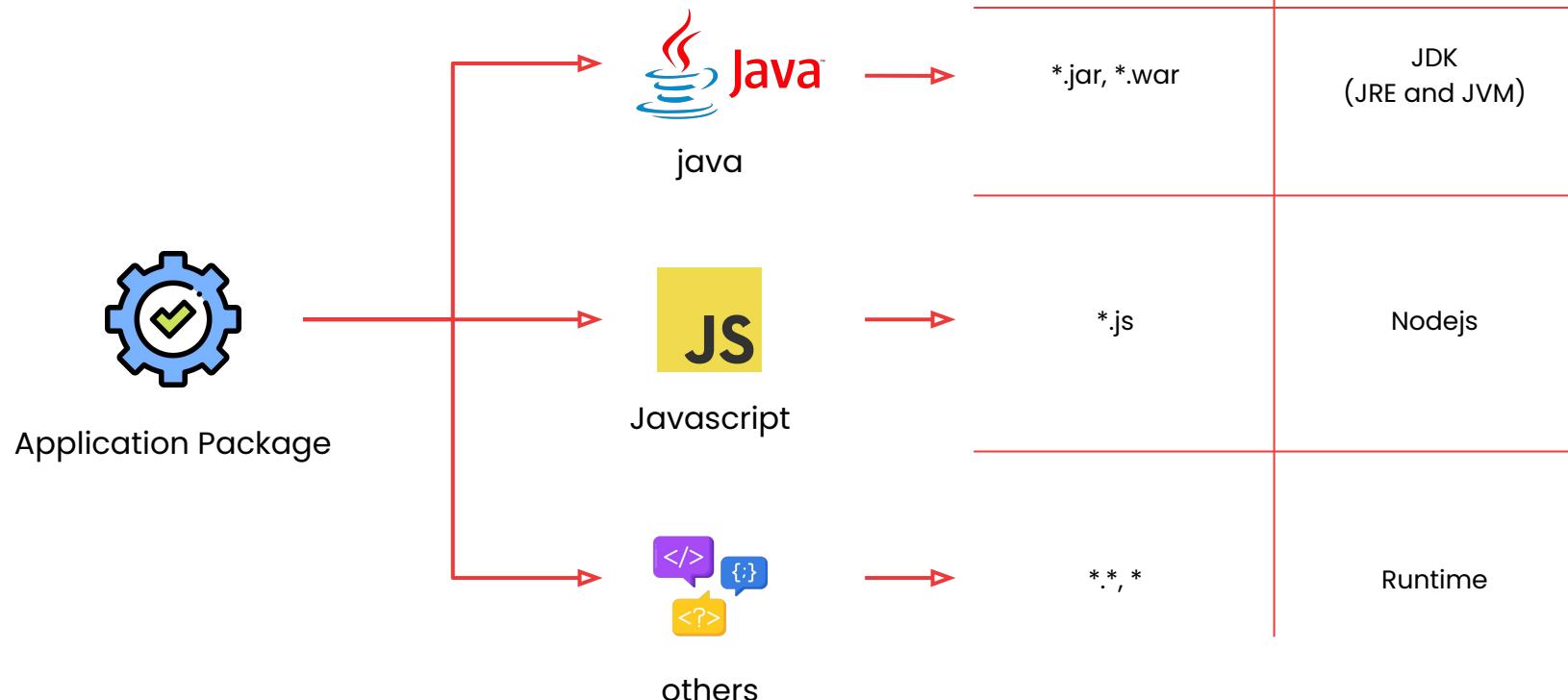
Executable file: Application Package (Cont.)

High-level programming language:



Executable file: Application Package (Cont.)

High-level programming language:



Summary Executable file

Summary Executable file (Cont.)

System Programming Language: OS Executable

- Window: *.exe
- Linux, MacOS: binary

Summary Executable file (Cont.)

System Programming Language: OS Executable

- Window: *.exe
- Linux, MacOS: binary

High-level Programming Language: Application Package (Need Application Runtime)

- JDK: Java
- Nodejs: Javascript
- Dotnet: C#
- Others: others

Summary Executable file (Cont.)

System Programming Language: OS Executable

- Window: *.exe
- Linux, MacOS: binary

High-level Programming Language: Application Package (Need Application Runtime)

- JDK: Java
- Nodejs: Javascript
- Dotnet: C#
- Others: others

Trend Application Programming with OS Executable

Jumpbox®

Trend Application Programming with OS Executable (Cont.)

Node.js v20.2.0 | ► Table of contents | ► Index | ► Other versions | ► Options

▼ Table of contents

- Single executable applications
 - Generating single executable preparation blobs
 - Notes
 - `require(id)` in the injected module is not file based
 - `__filename` and `module.filename` in the injected module
 - `__dirname` in the injected module
 - Single executable application creation process
 - Platform support

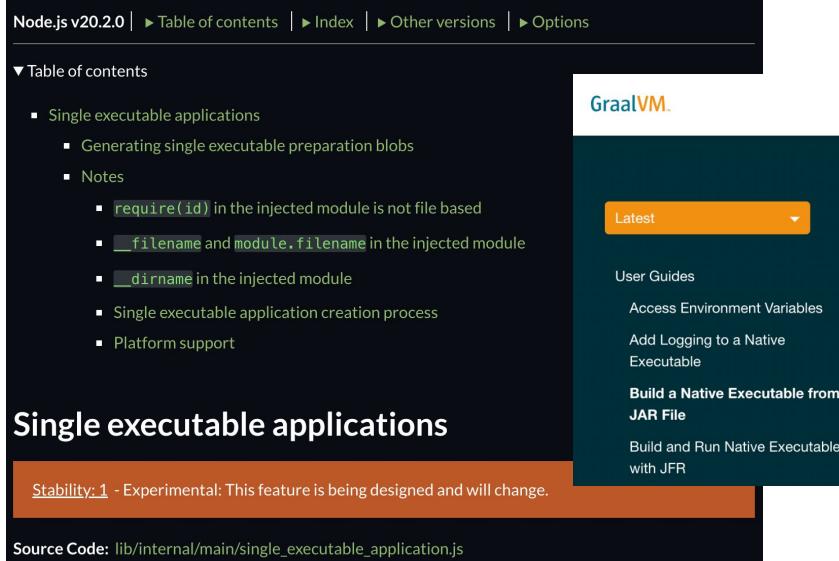
Single executable applications

Stability: 1 - Experimental: This feature is being designed and will change.

Source Code: [lib/internal/main/single_executable_application.js](#)

Nodejs Single executable: [link](#)

Trend Application Programming with OS Executable (Cont.)



Node.js v20.2.0 | ▶ Table of contents | ▶ Index | ▶ Other versions | ▶ Options

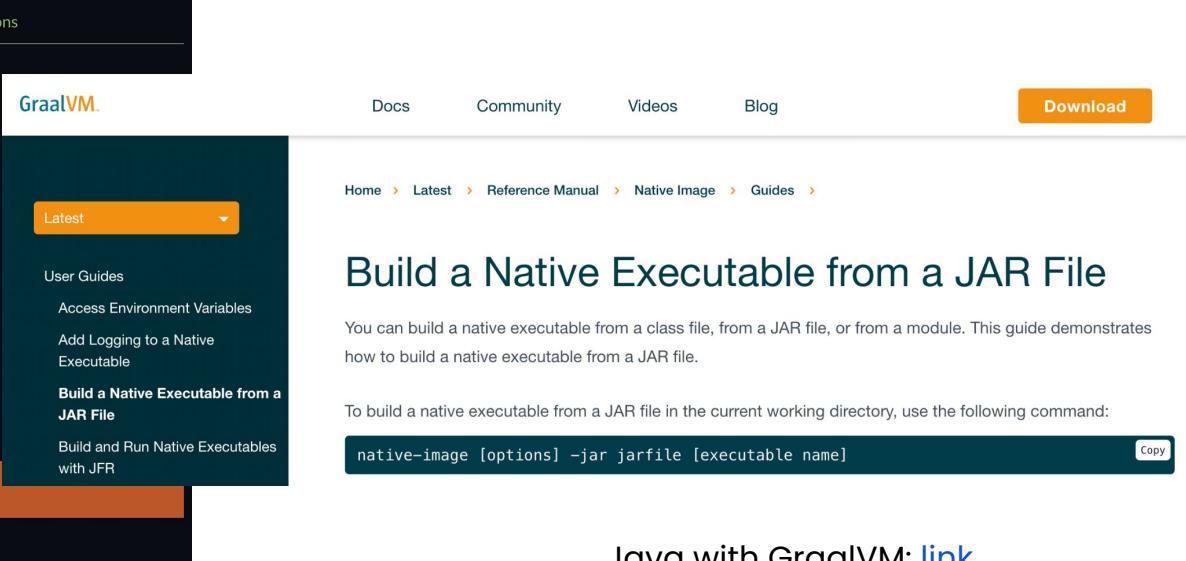
▼ Table of contents

- Single executable applications
 - Generating single executable preparation blobs
 - Notes
 - `require(id)` in the injected module is not file based
 - `__filename` and `module.filename` in the injected module
 - `__dirname` in the injected module
 - Single executable application creation process
 - Platform support

Single executable applications

Stability: 1 - Experimental: This feature is being designed and will change.

Source Code: [lib/internal/main/single_executable_application.js](#)



GraalVM

Docs Community Videos Blog Download

Home > Latest > Reference Manual > Native Image > Guides >

Build a Native Executable from a JAR File

You can build a native executable from a class file, from a JAR file, or from a module. This guide demonstrates how to build a native executable from a JAR file.

To build a native executable from a JAR file in the current working directory, use the following command:

```
native-image [options] -jar jarfile [executable name]
```

Copy

Java with GraalVM: [link](#)

Nodejs Single executable: [link](#)

Trend Application Programming with OS Executable (Cont.)

Trend Application Programming with OS Executable (Cont.)

Pros

- No Need Application Runtime
- Lightweight
- High Performance

Trend Application Programming with OS Executable (Cont.)

Pros

- No Need Application Runtime
- Lightweight
- High Performance

Cons

- Need export to Cross Multiple Architecture and Platform

Trend Application Programming with OS Executable (Cont.)

Pros

- No Need Application Runtime
- Lightweight
- High Performance

Cons

- Need export to Cross Multiple Architecture and Platform

Application Output

- 1. Application Characteristic -

Tree Output

Tree Output (Cont.)



Tree

Tree Output (Cont.)



Tree

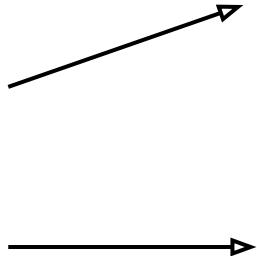


Gas

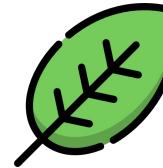
Tree Output (Cont.)



Tree



Gas

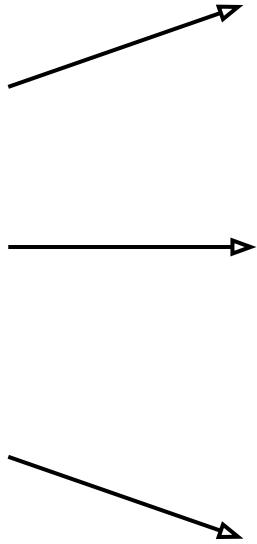


Leaf

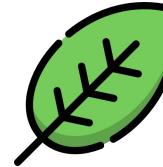
Tree Output (Cont.)



Tree



Gas



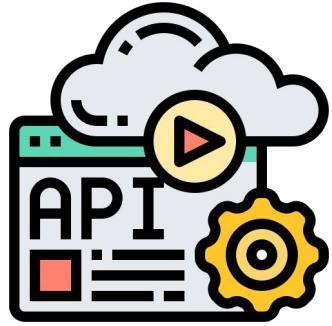
Leaf



Fruit

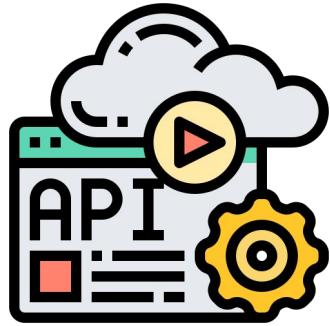
Application Output

Application Output (Cont.)



App

Application Output (Cont.)

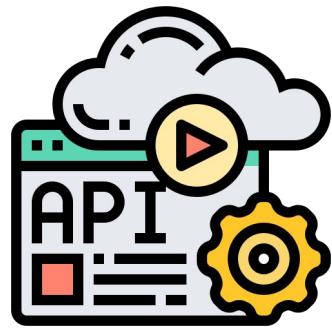


App

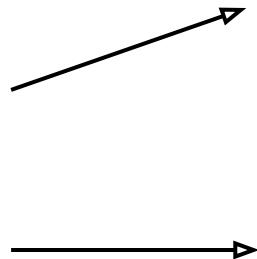


Logs

Application Output (Cont.)



App

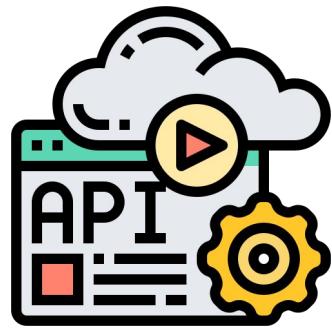


Logs

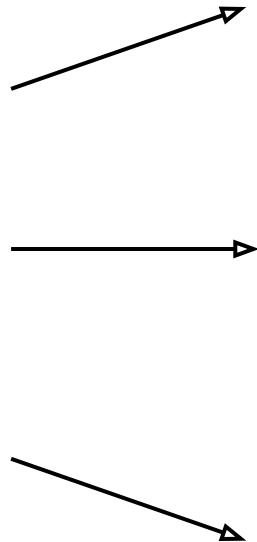


Traces

Application Output (Cont.)



App



Logs

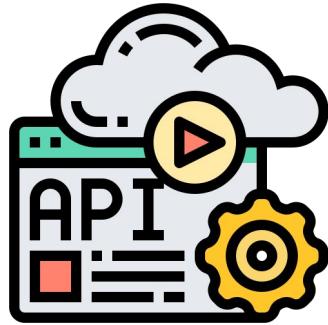


Traces

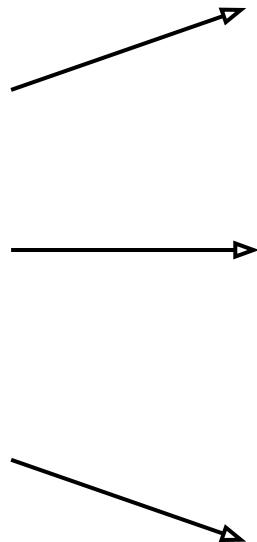


Metrics

Application Output (Cont.)



App



Logs



Traces



Metrics

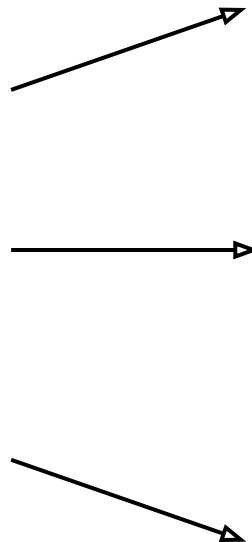


Jumpbox®

Application Output (Cont.)



App



Logs



Traces



Metrics

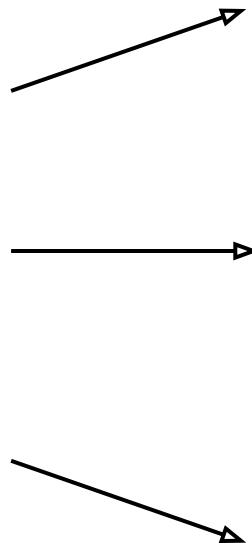


Jumpbox®

Application Output (Cont.)



App



Logs



Traces



Metrics



Jumpbox®

Application Package Manager

- 1. Application Characteristic -

Application Package Manager

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

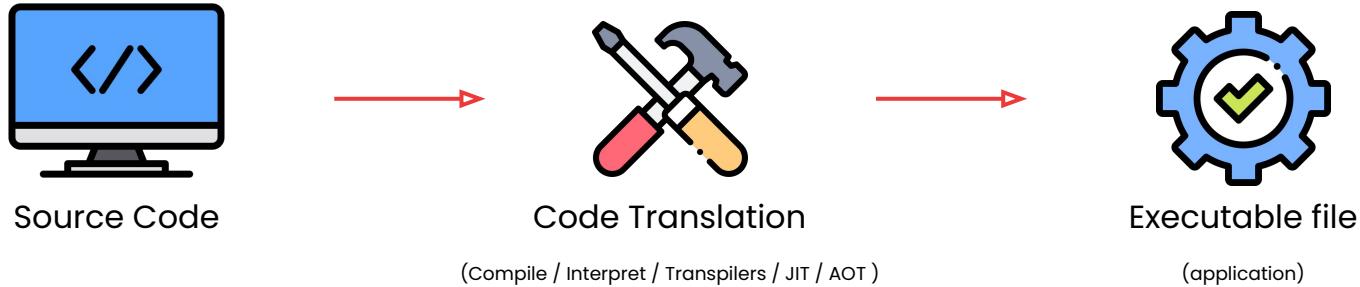
Let's get back to

Jumpbox®

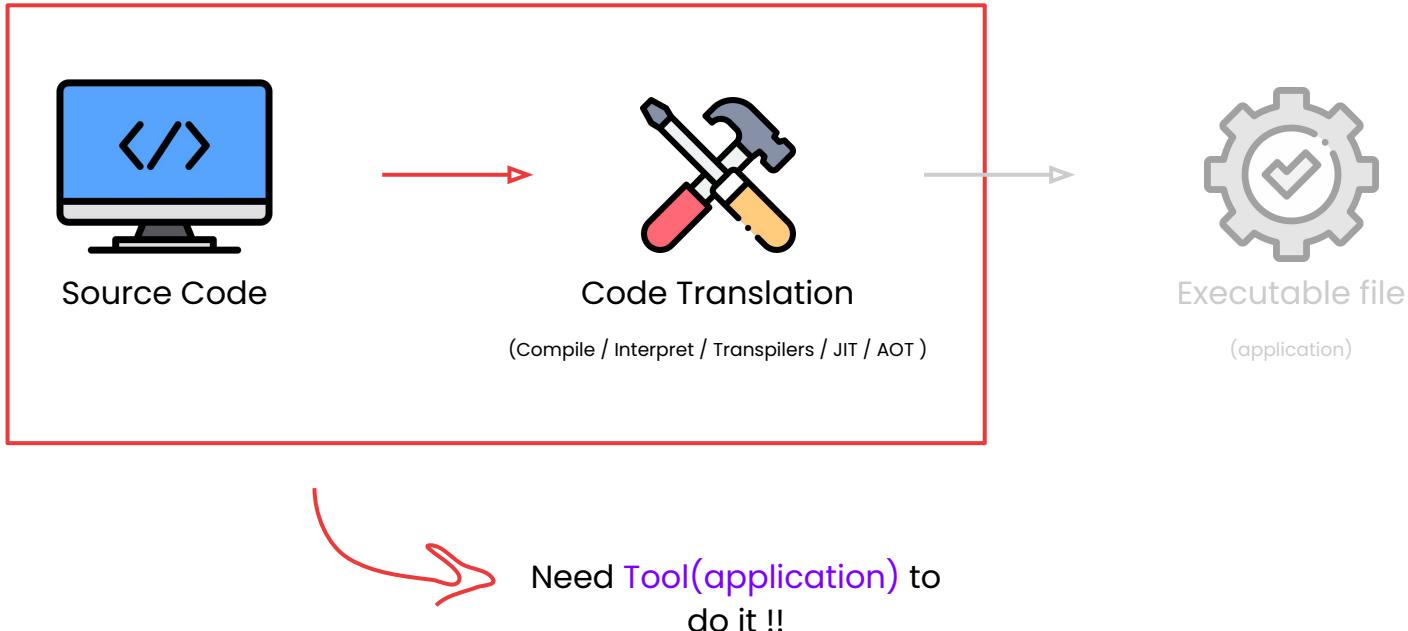
เจ้าของลิขสิทธิ์ อนุญาติให้ใช้ร่องรอยนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเมิด จะถูกดำเนินคดีตามกฎหมาย

Let's get back to
the Application Building Process

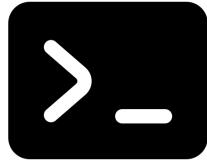
Application Package Manager(Cont.)



Application Package Manager(Cont.)

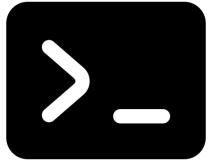


Application Package Manager(Cont.)



Tool = CLI = application

Application Package Manager(Cont.)



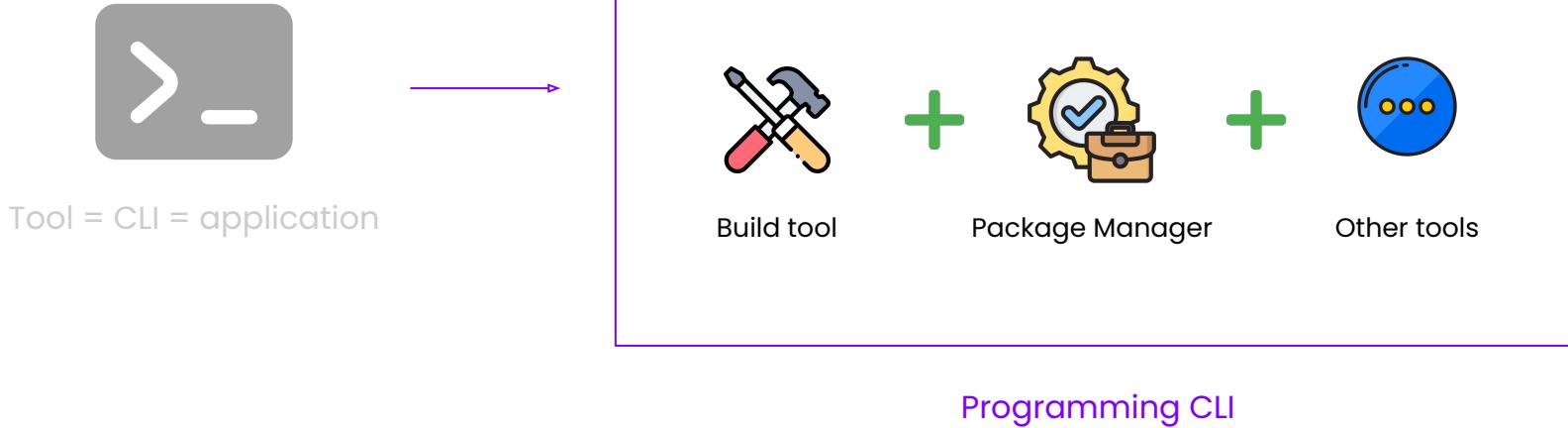
Tool = **CLI** = application



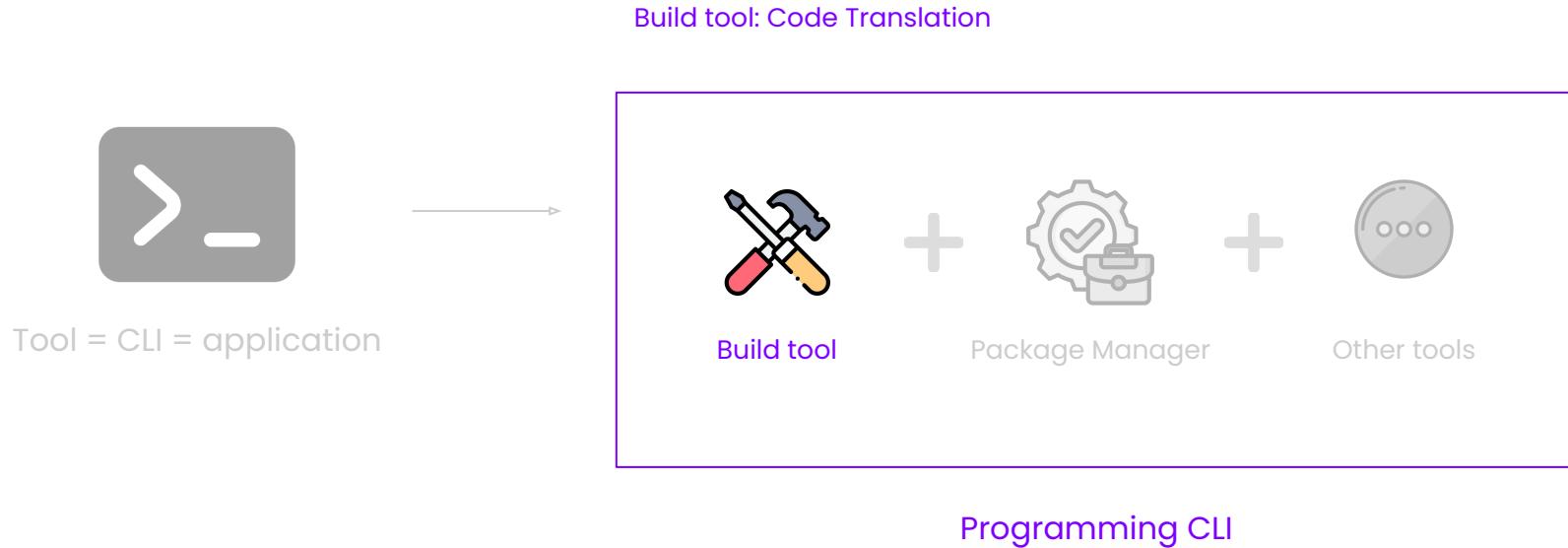
Command Line Interface

(ช่องทางในการติดต่อสื่อสารด้วย command ภายใน 1 บรรทัด)

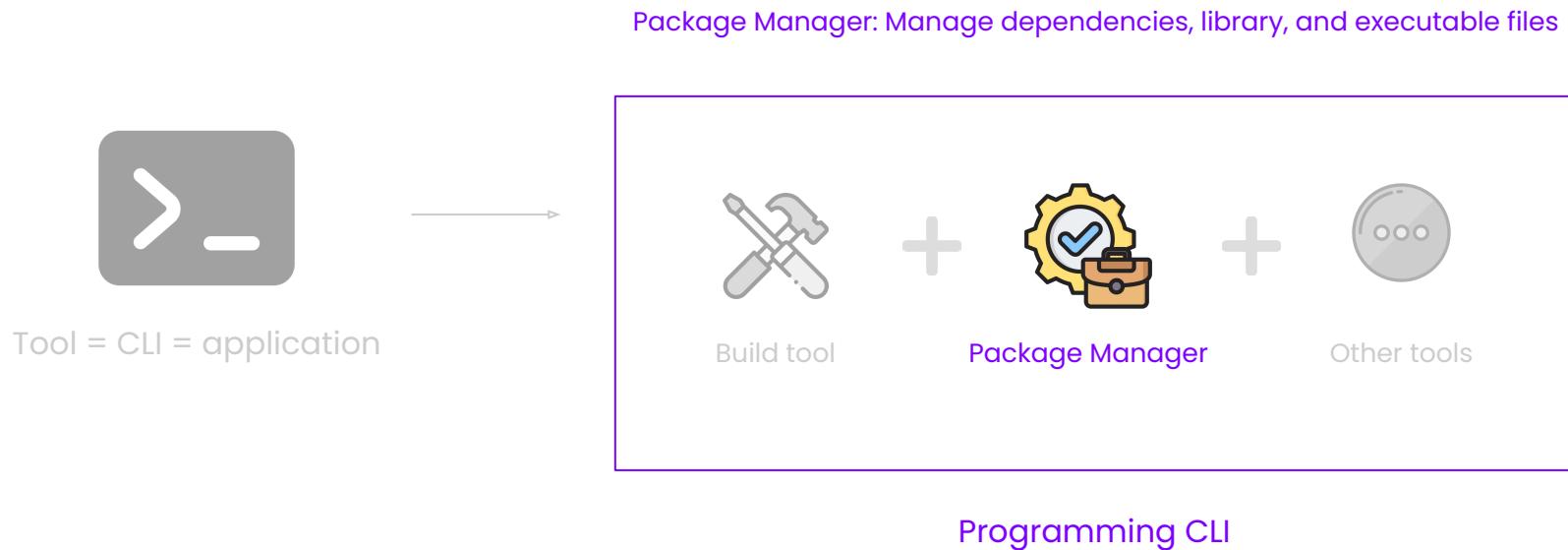
Application Package Manager(Cont.)



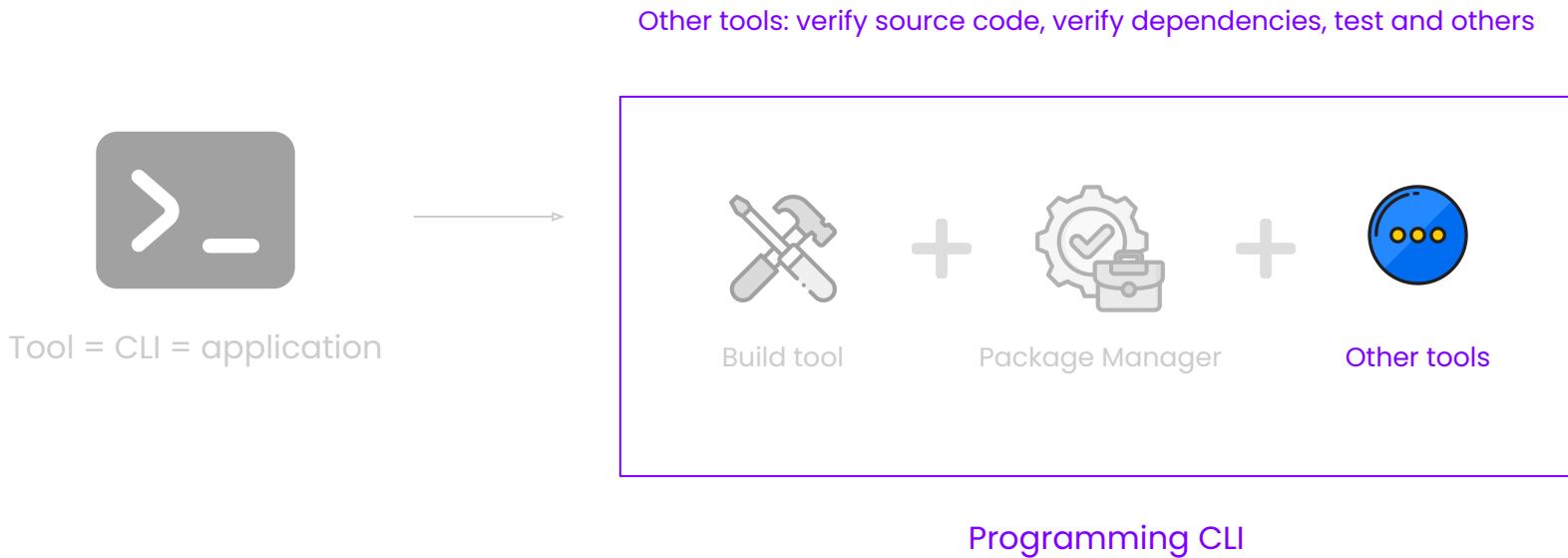
Application Package Manager(Cont.)



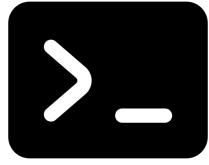
Application Package Manager(Cont.)



Application Package Manager(Cont.)

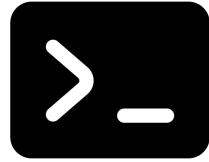


Application Package Manager(Cont.)



Package Manager

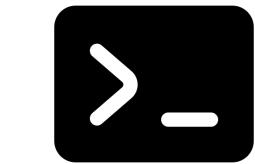
Application Package Manager(Cont.)



Package Manager

Javascript
npm install -g <program / application> e.g. npm install -g yarn

Application Package Manager(Cont.)



Package Manager

Javascript

npm install -g <program / application> e.g. npm install -g yarn

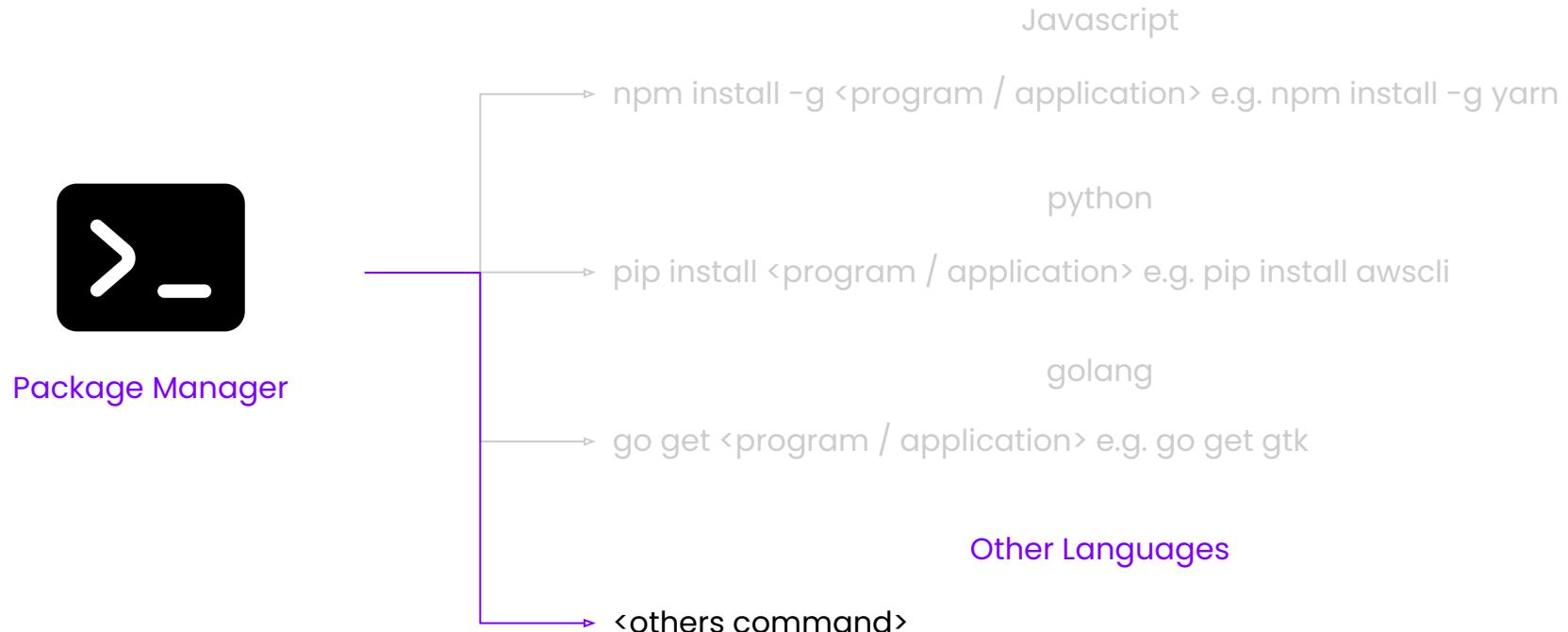
python

pip install <program / application> e.g. pip install awscli

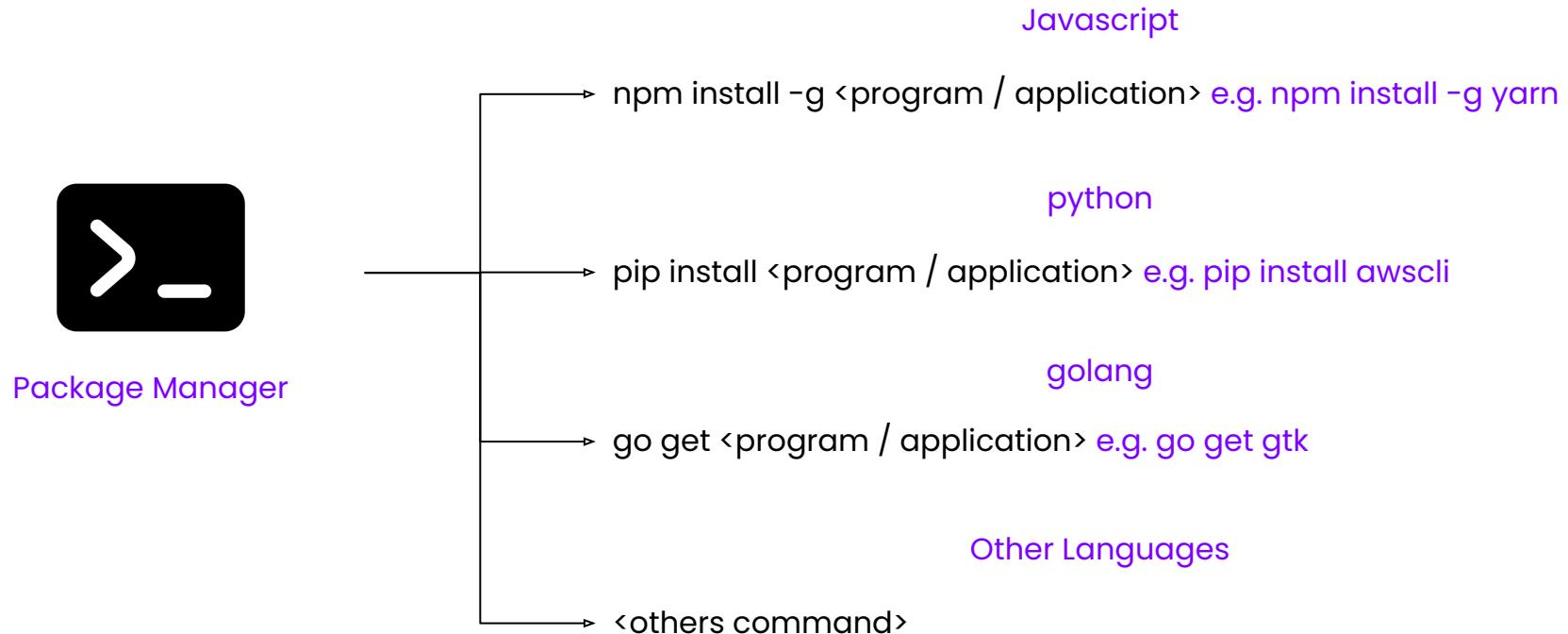
Application Package Manager(Cont.)



Application Package Manager(Cont.)



Application Package Manager(Cont.)



Observability

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จะถูกดำเนินคดีตามกฎหมาย

Observability

Observe + ability

Observe

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งที่มีลักษณะเด่นๆ ของการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นแน่นอน จะถูกดำเนินคดีตามกฎหมาย

Jumpbox®

Observe = สังเกต

Ability

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้ละเอียด จะถูกดำเนินคดีตามกฎหมาย

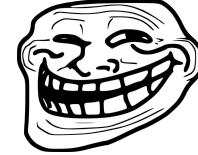
Jumpbox®

Ability = ความสามารถ

Observability

ความสามารถในการสังเกต

Telemetry



Observability

ความสามารถในการสังเกต

Telemetry

Jumpbox®

เจ้าของลิขสิทธิ์ อนุญาติให้ใช้สิ่งดังนี้เพื่อการเรียนรู้ส่วนบุคคลเท่านั้น และขอสงวนลิขสิทธิ์ ห้ามมิให้เผยแพร่ในที่สาธารณะ ผู้อื่นเดินทาง จนถูกดำเนินคดีตามกฎหมาย

Telemetry

Tele (ไกล) + Metron (การวัด)

Telemetry

Tele (ไกล) + Metron (การวัด)

นั่นหมายถึง กระบวนการส่งข้อมูลการวัดจากที่ห่างไกล เพื่อนำมาวิเคราะห์ ก่อนเราจะมีความสามารถในการสังเกตได้ หรือที่เรียกว่า Observability

For what?



รู้ว่าผู้ใช้งานเข้ามายังในระบบ ว่าอ
ยู่ส่วนไหน (Trace)

For what?



รู้ว่าผู้ใช้งานเข้ามายังระบบ ว่าอยู่ส่วนไหน (Trace)



เกิดอะไรขึ้น, ทำอะไรไปบ้าง
(Log)

For what?



รู้ว่าผู้ใช้งานเข้ามาในระบบ ว่าอยู่
ส่วนไหน (Trace)



เกิดอะไรขึ้น, ทำอะไรไปบ้าง
(Log)

For what?



ทำแล้วกี่ครั้ง / ใช้งานกี่คน
(Metric)



Where is it
happening

รู้ว่าผู้ใช้งานเข้ามาในระบบ ว่าอยู่
ส่วนไหน (Trace)

What's happening



เกิดอะไรขึ้น, ทำอะไรไปบ้าง
(Log)

For what?

Something is
happening



ทำแล้วกี่ครั้ง / ใช้งานกี่คน
(Metric)