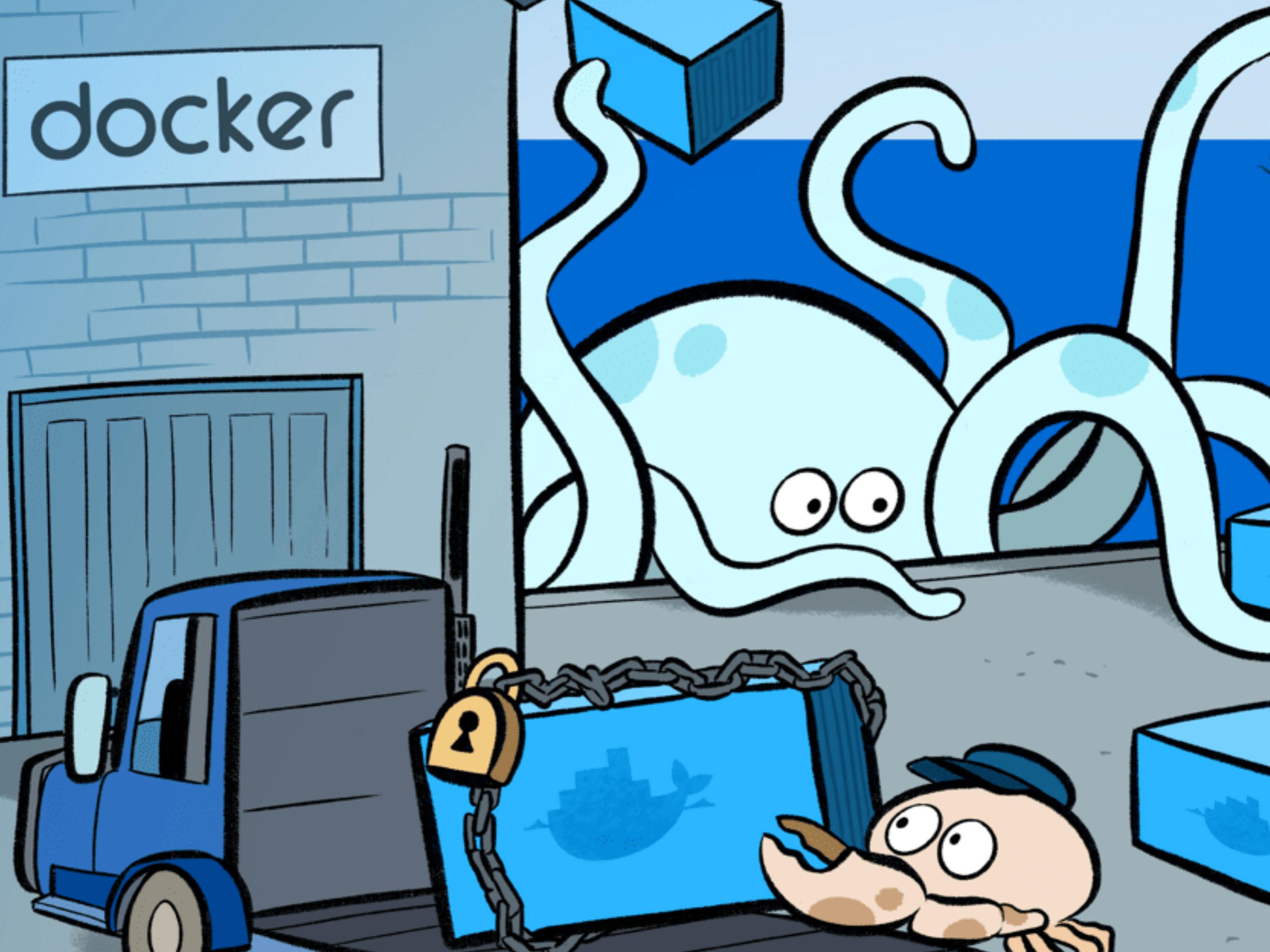


Web Application Development

Alexander Menshchikov

Docker

docker

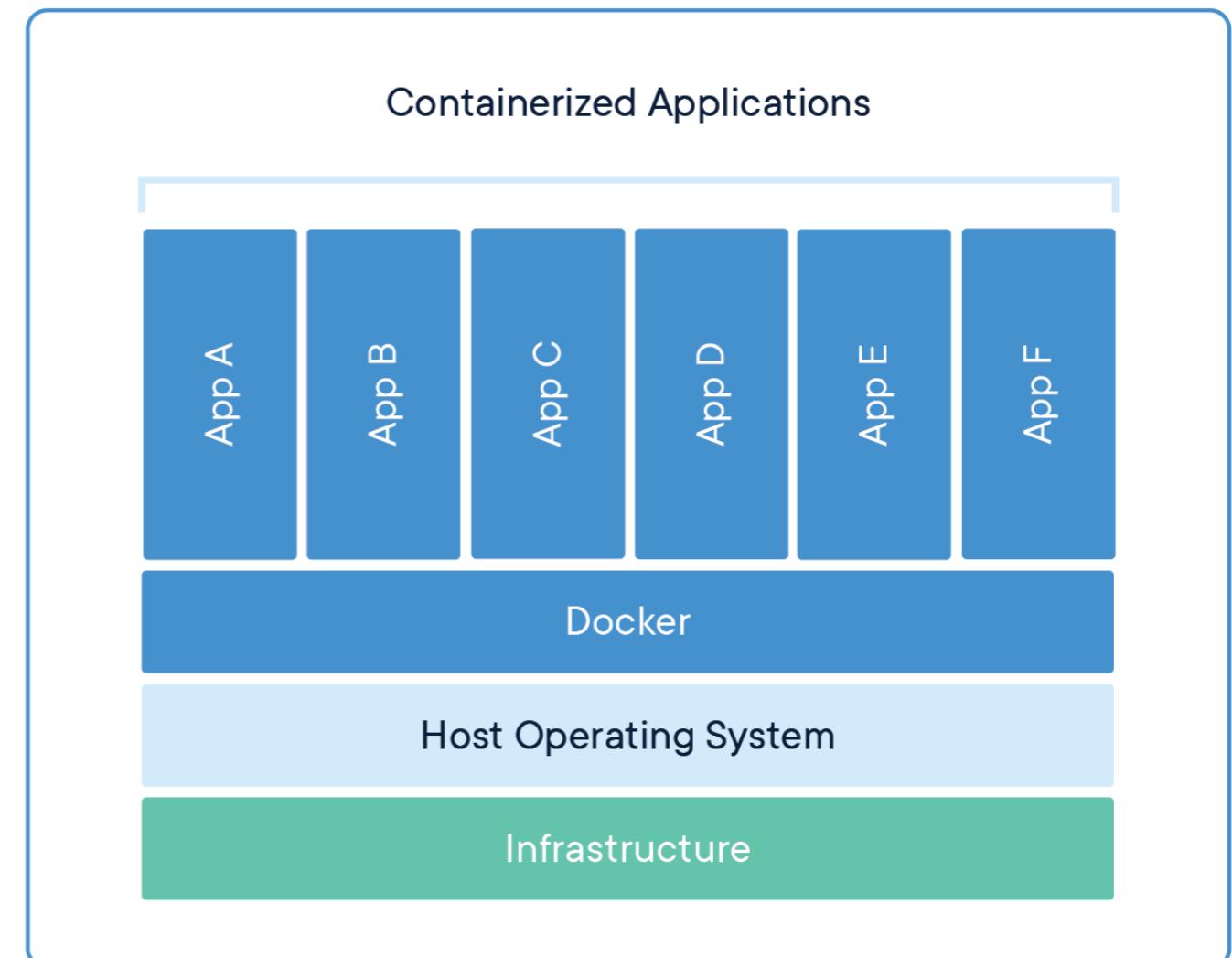
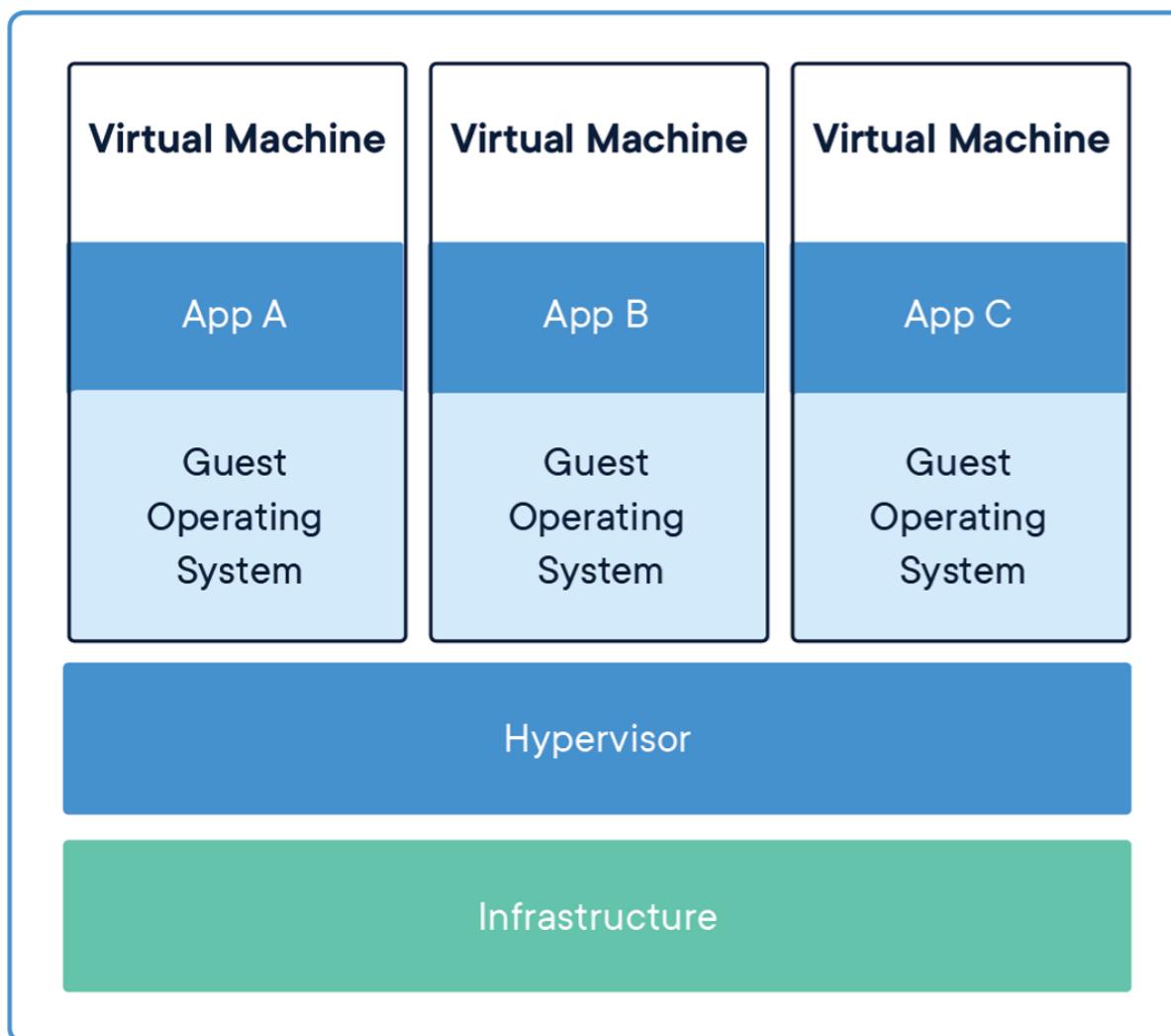


Why virtualization?

- Resource utilisation
- Security
- Isolation
- Scalability/Portability
- Easy backups
- Abstraction from hardware



Virtualisation containerisation

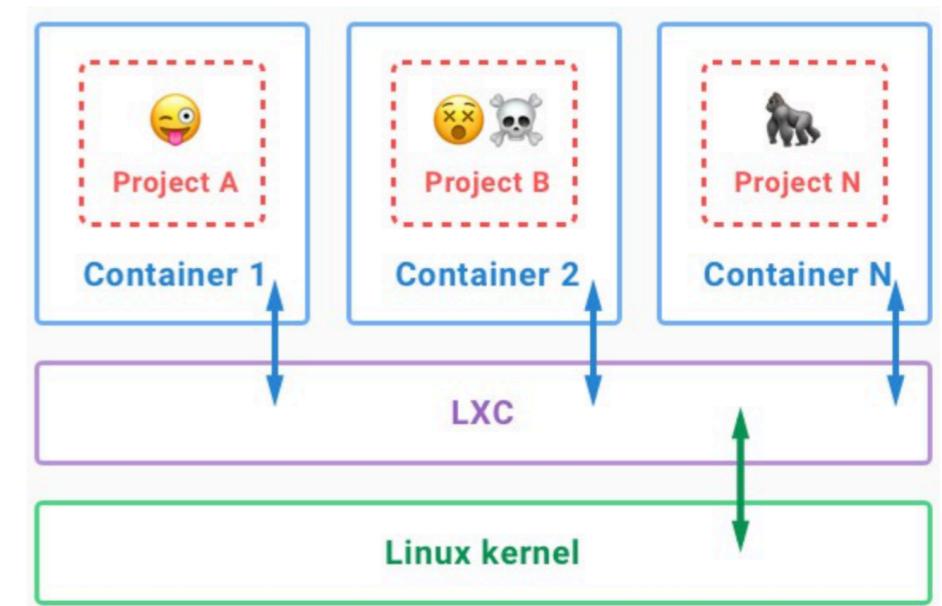


Virtual machines

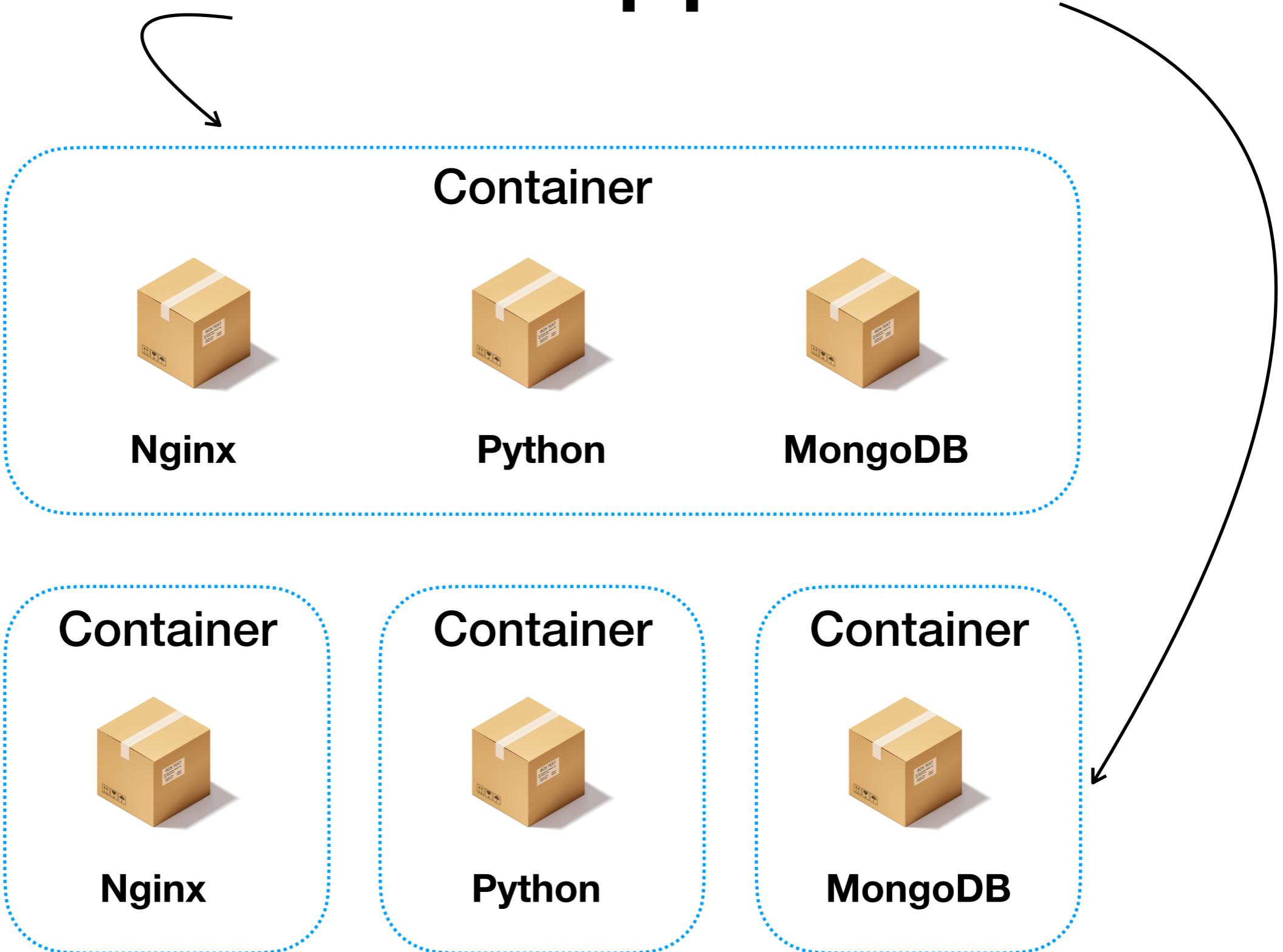
Containers

Containers != virtualization

-  Better performance
-  File sharing between containers
- Limited OS support

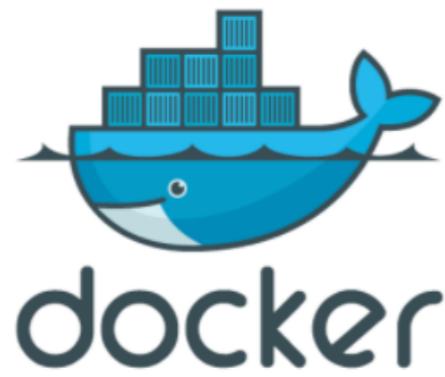


Machine or application

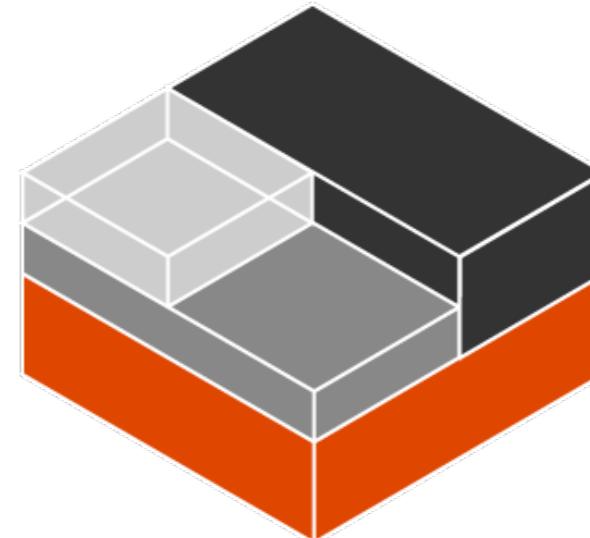


LXD or Docker

Docker hosts application containers



LXD hosts machine containers



Docker commands



Installation

```
apt install -y apt-transport-https ca-certificates curl software-properties-common  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"  
apt install -y docker-ce
```

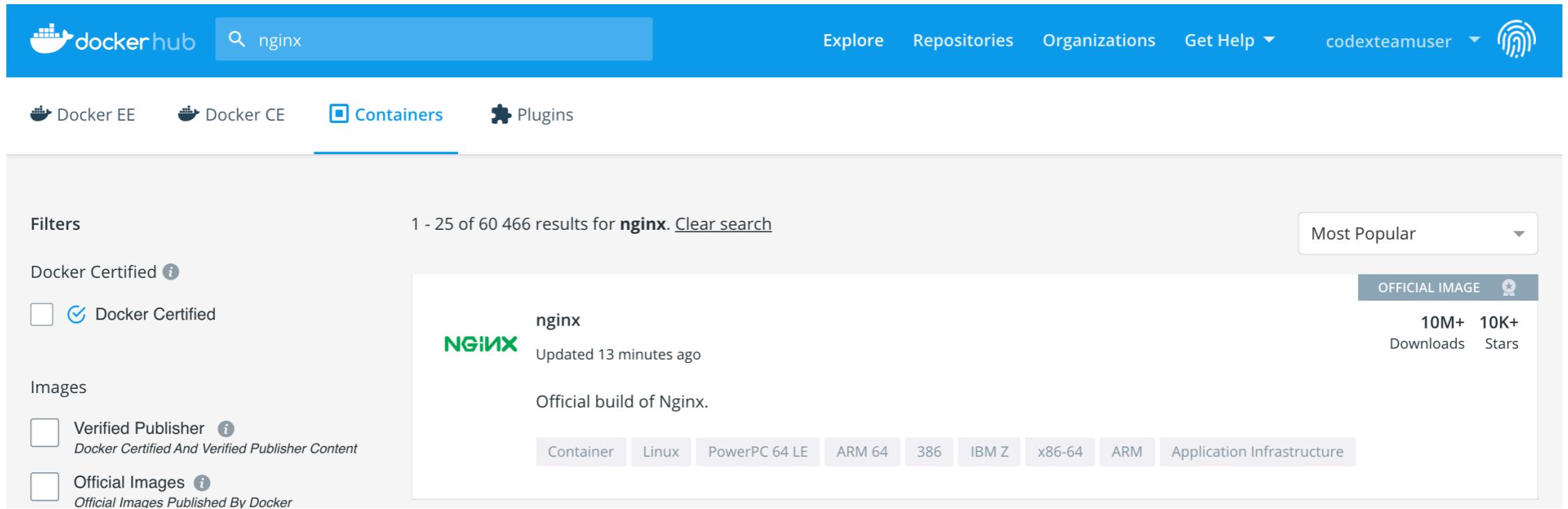
<https://docs.docker.com/get-docker/>

Elements

- Image
- Container
- Network
- Volume



Dockerhub



A screenshot of the Dockerhub website showing search results for "nginx". The top navigation bar includes links for Explore, Repositories, Organizations, Get Help, codexteamuser, and a profile icon. Below the search bar, tabs for Docker EE, Docker CE, Containers (which is selected), and Plugins are visible. On the left, there are filters for Docker Certified (with an unchecked checkbox for Docker Certified), Images (with checkboxes for Verified Publisher and Official Images), and a dropdown for sorting by Most Popular. The main search results show the official Nginx Docker image, which has been updated 13 minutes ago. It is labeled as an OFFICIAL IMAGE with 10M+ Downloads and 10K+ Stars. The image details include Container, Linux, PowerPC 64 LE, ARM 64, 386, IBM Z, x86-64, ARM, and Application Infrastructure.



docker/whalesay ☆

By [docker](#) • Updated 5 years ago

An image for use in the Docker demo tutorial

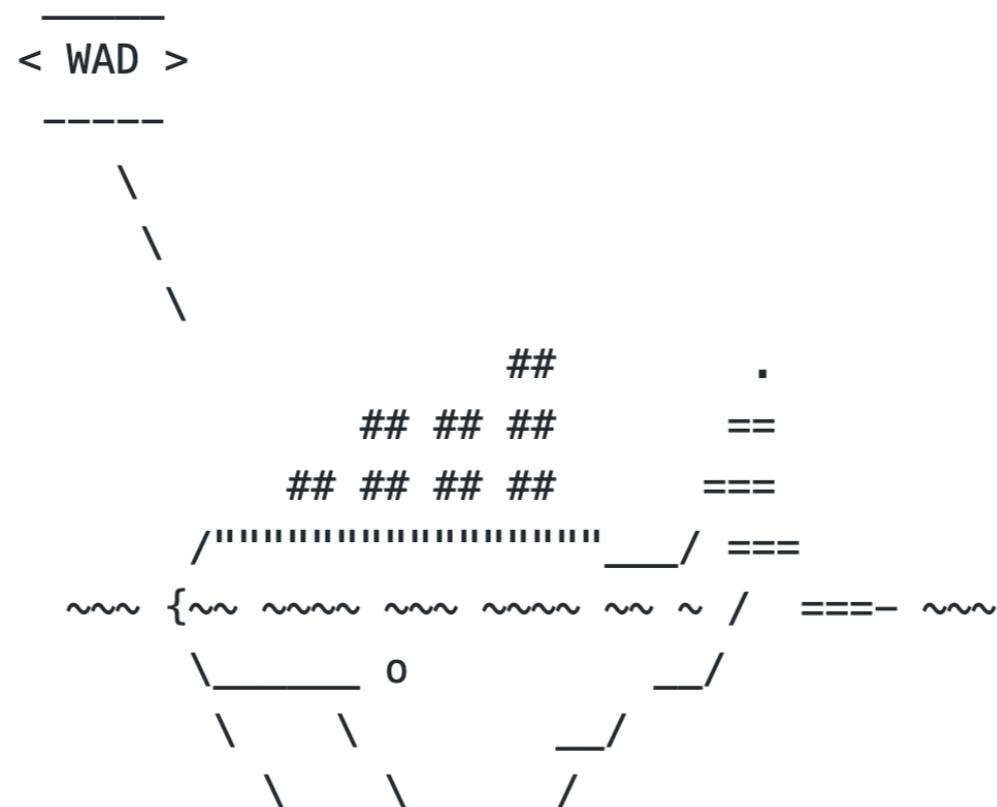
Container

<https://hub.docker.com/r/docker/whalesay/>

Hello world

```
> docker pull docker/whalesay
```

```
> docker run docker/whalesay cowsay WAD
```



Images

```
> docker image ls
```

```
> docker image inspect docker/whalesay
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
root_flask	latest	10b38c58159b	About an hour ago	210MB
root_flask-simple	latest	8695c3e97616	About an hour ago	210MB
codexteamuser/hawk-collector	prod	b921deac96e6	3 days ago	20.3MB
consul	latest	197999eb696c	11 days ago	116MB
ubuntu	latest	1d622ef86b13	11 days ago	73.9MB
nginx	latest	602e111c06b6	12 days ago	127MB
python	3.8.1-slim-buster	b99890b7a7dc	2 months ago	193MB
docker/whalesay	latest	6b362a9f73eb	4 years ago	247MB

Containers

```
> docker run -d ubuntu bash -c "while true; do sleep 3; done"
```

```
> docker ps
```

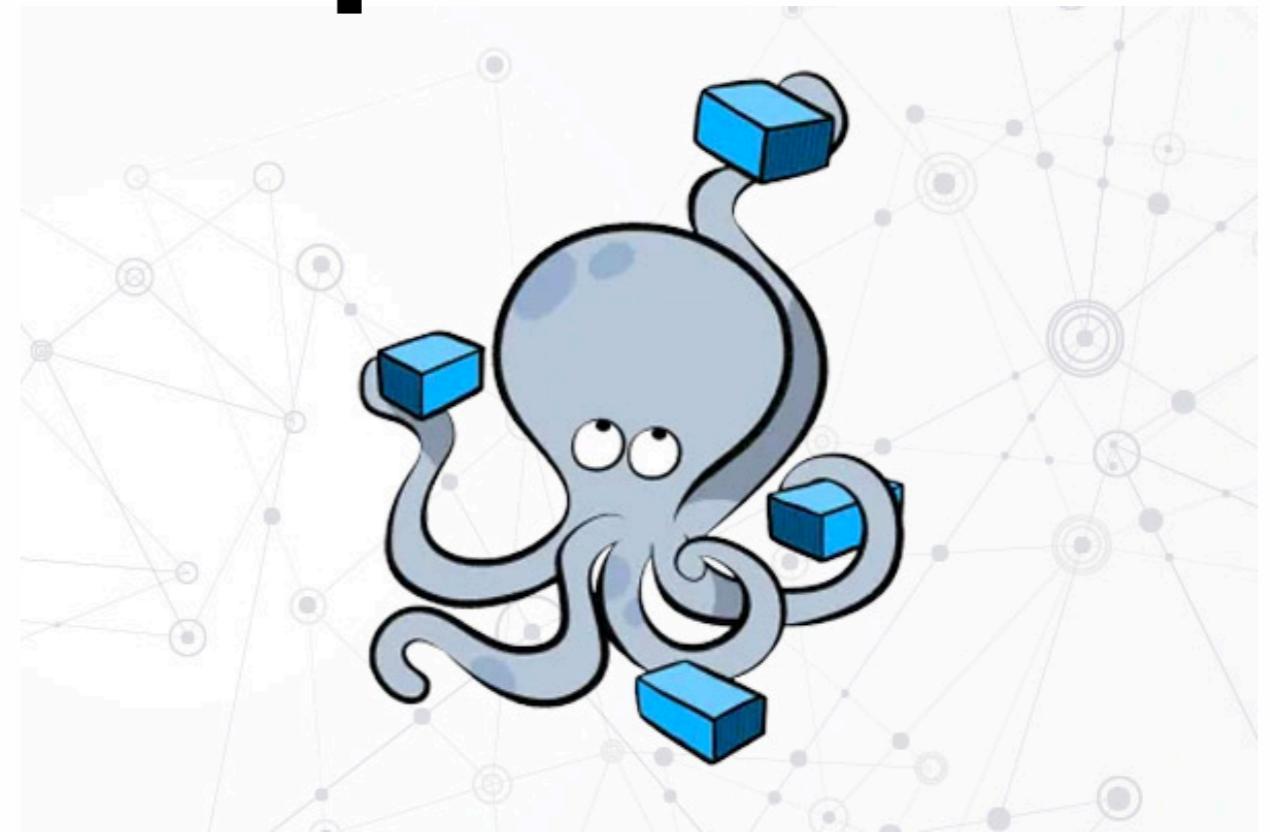
```
> docker ps --format "{{.ID}}\t{{.Image}}\t{{.Names}}"
```

```
d4021aab2350    ubuntu    cool_saha
```

```
> docker exec -it d4021aab2350 bash
```

```
root@d4021aab2350:/# ps uaxf
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      66 13.3  0.1    4112  3400 pts/0      Ss  12:53  0:00 bash
root      74  0.0  0.1    5884  2908 pts/0      R+  12:53  0:00  \_ ps uaxf
root      1  0.2  0.1    3980  3156 ?        Ss  12:50  0:00 bash -c while true; do sleep 3; done
root     73  0.0  0.0    2512   588 ?        S   12:53  0:00 sleep 3
```

Docker-compose



Installation

Find current stable version: <https://github.com/docker/compose/releases> and replace 1.25.5 word **in** the following command.

```
curl -L https://github.com/docker/compose/releases/download/1.25.5/docker-compose-  
`uname -s`-`uname -m` -o /usr/local/bin/docker-compose  
chmod +x /usr/local/bin/docker-compose
```



<https://docs.docker.com/compose/install/>

Docker-compose

```
version: "3.2"
services:
  nginx:
    image: nginx
    ports:
      - "80:80"
  volumes:
    - ./index.html:/usr/share/nginx/html/
index.html
```

Docker-compose cli

- > docker-compose ps
- > docker-compose build
- > docker-compose up
- > docker-compose down
- > docker-compose up -d

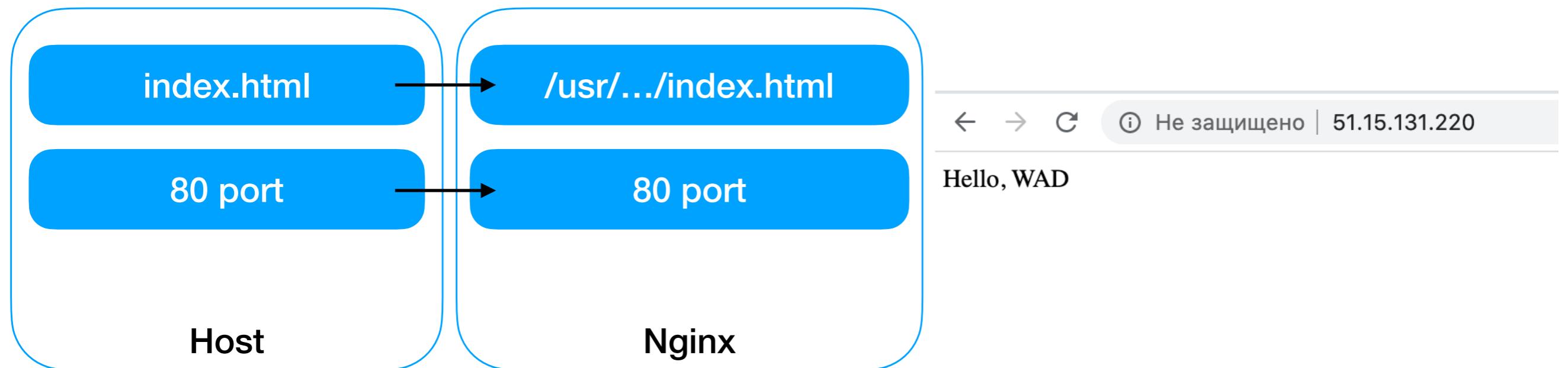
Docker-compose

```
root@scw-recursing-tesla:~# docker ps
```

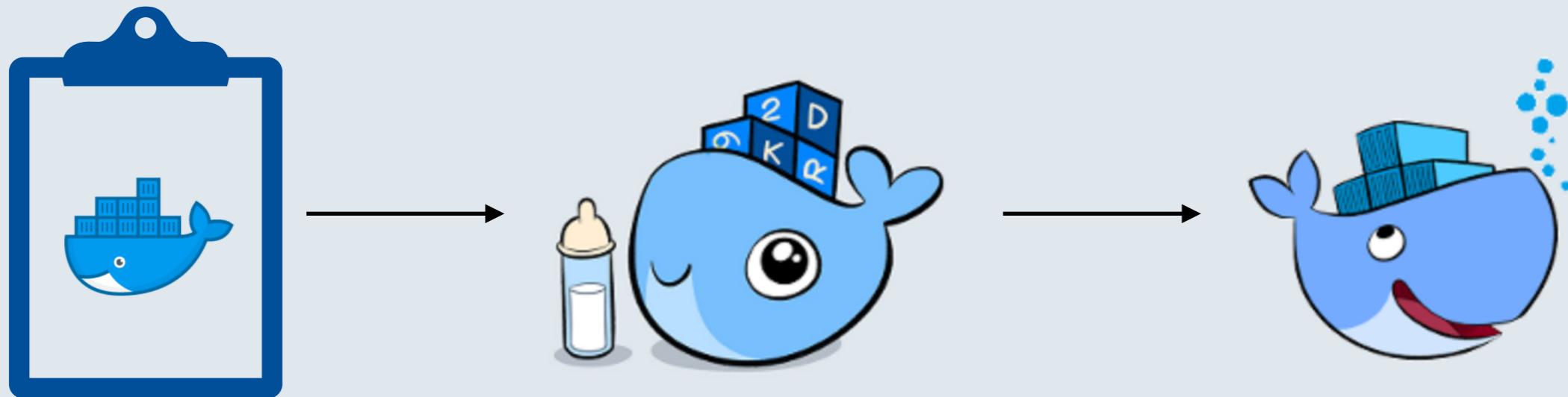
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
15432721d0f6	nginx	"nginx -g 'daemon off;"	About a minute ago	Up 1 second	0.0.0.0:80->80/tcp	root_nginx_1

```
root@scw-recursing-tesla:~# cat index.html
```

```
Hello, WAD
```



Dockerfile



Flask app

```
from flask import Flask, jsonify

app = Flask(__name__)

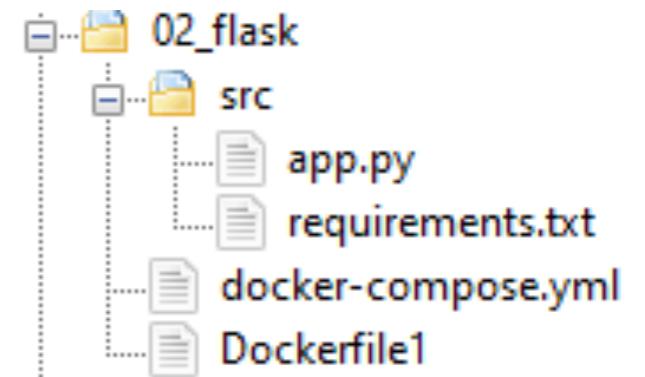
@app.route("/")
def hello_world():
    return jsonify(hello="world")

if __name__ == "__main__":
    app.run(port=5000, host="0.0.0.0")
```

./src/app.py

flask

./src/requirements.txt



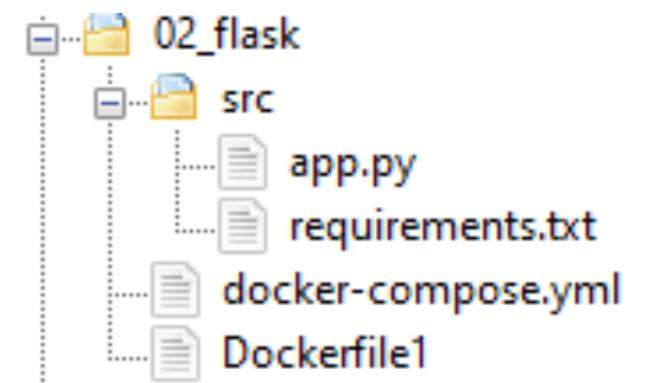
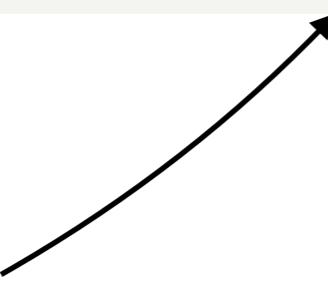
Dockerfile

```
FROM python:3.8.1-slim-buster

# set work directory
WORKDIR /usr/src/app

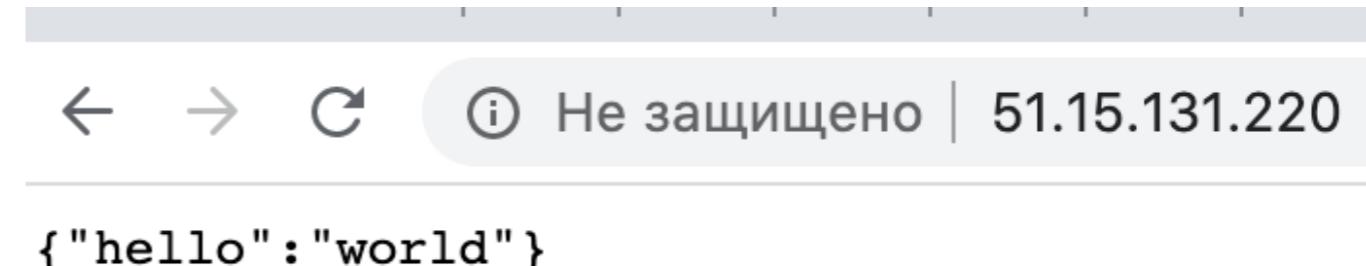
# install dependencies
RUN pip install --upgrade pip
COPY ./src/requirements.txt /usr/src/app/requirements.txt
RUN pip install -r requirements.txt
```

Dockerfile1



Compose

```
version: "3.2"
services:
  flask:
    build:
      dockerfile: Dockerfile1
      context: .
    ports:
      - "80:5000"
    command: python app.py
    volumes:
      - ./src:/usr/src/app/
```



docker-compose.yml

```
root@scw-recursing-tesla:~# docker-compose up
WARNING: Found orphan containers (root_nginx_1) for this project. If you removed or renamed thi
Starting root_flask_1 ... done
Attaching to root_flask_1
flask_1  | * Serving Flask app "app" (lazy loading)
flask_1  | * Environment: production
flask_1  |   WARNING: This is a development server. Do not use it in a production deployment.
flask_1  |   Use a production WSGI server instead.
flask_1  | * Debug mode: off
flask_1  | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

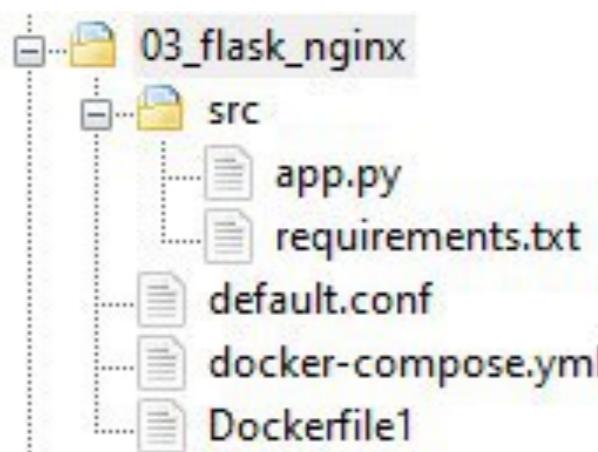
Flask + Nginx in Docker

Flask app

```
server {  
    listen      80;  
    server_name localhost;  
  
    location / {  
        proxy_pass http://flask-simple:5000/;  
    }  
}
```

docker-compose.yml

default.conf



```
version: "3.2"  
services:  
    nginx:  
        image: nginx  
        ports:  
            - "80:80"  
        volumes:  
            - ./default.conf:/etc/nginx/conf.d/default.conf  
    flask-simple:  
        build:  
            dockerfile: Dockerfile1  
            context: .  
        command: python app.py  
        volumes:  
            - ./src:/usr/src/app/
```

Result

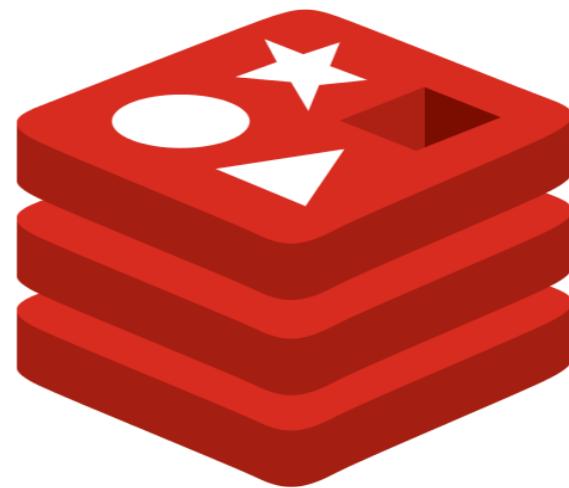
```
root@scw-recursing-tesla:~# docker-compose up
WARNING: Found orphan containers (root_flask_1) for this project. If you removed o
Starting root_flask-simple_1 ... done
Starting root_nginx_1      ... done
Attaching to root_nginx_1, root_flask-simple_1
flask-simple_1  | * Serving Flask app "app" (lazy loading)
flask-simple_1  | * Environment: production
flask-simple_1  |   WARNING: This is a development server. Do not use it in a pro
flask-simple_1  |   Use a production WSGI server instead.
flask-simple_1  | * Debug mode: off
flask-simple_1  | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
nginx_1        | 178.70.131.84 - - [05/May/2020:13:46:20 +0000] "GET / HTTP/1.1"
fari/537.36" "-"
flask-simple_1  | 172.18.0.2 - - [05/May/2020 13:46:20] "GET / HTTP/1.0" 200 -
flask-simple_1  | 172.18.0.2 - - [05/May/2020 13:46:20] "GET / HTTP/1.0" 200 -
nginx_1        | 178.70.131.84 - - [05/May/2020:13:46:20 +0000] "GET / HTTP/1.1"
fari/537.36" "-"
```

Literature

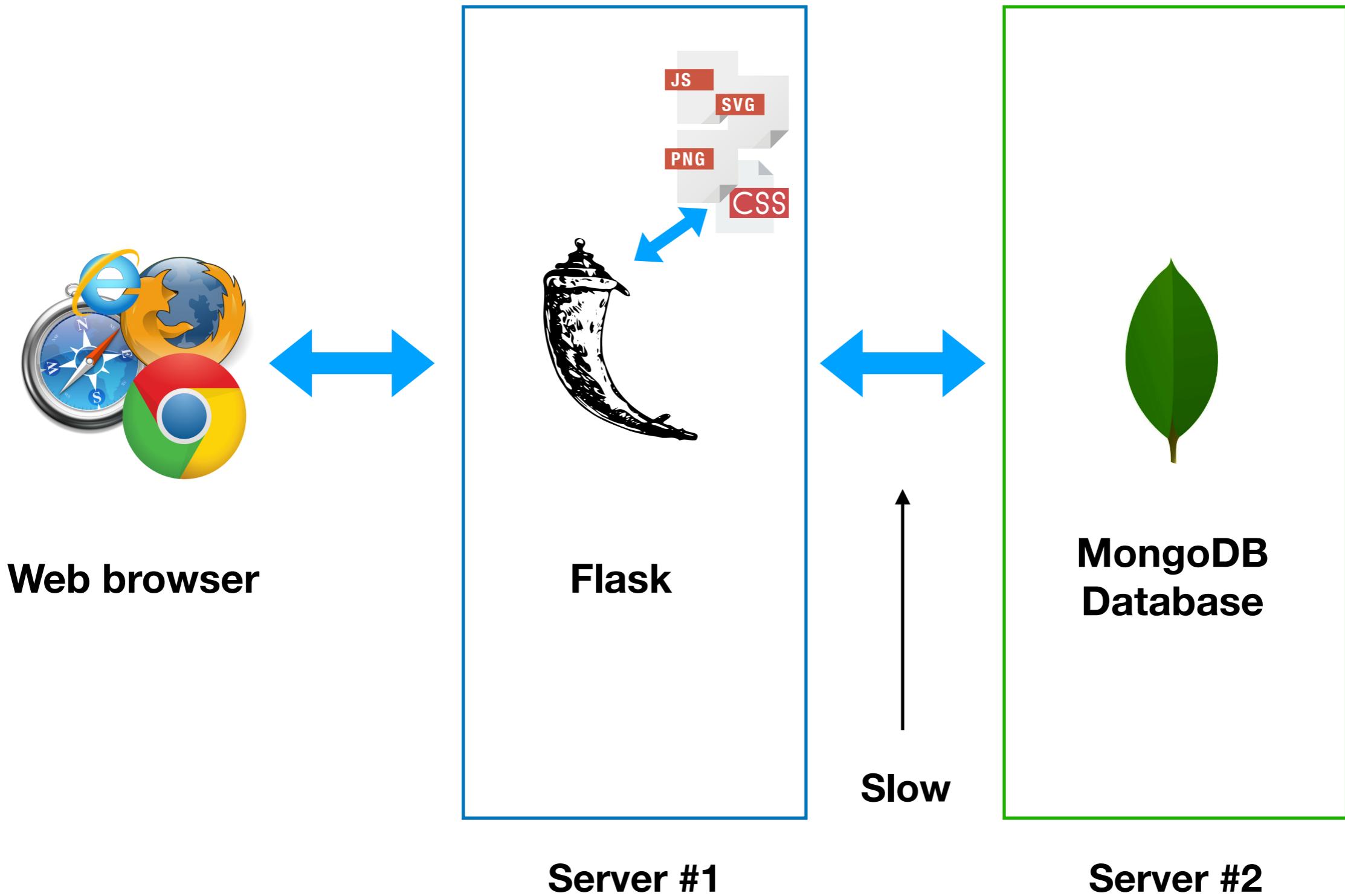
- Docker install: <https://docs.docker.com/get-docker/>
- Get started: <https://docs.docker.com/get-started/>
- Docker-compose install: <https://docs.docker.com/compose/install/>

Demo

Redis



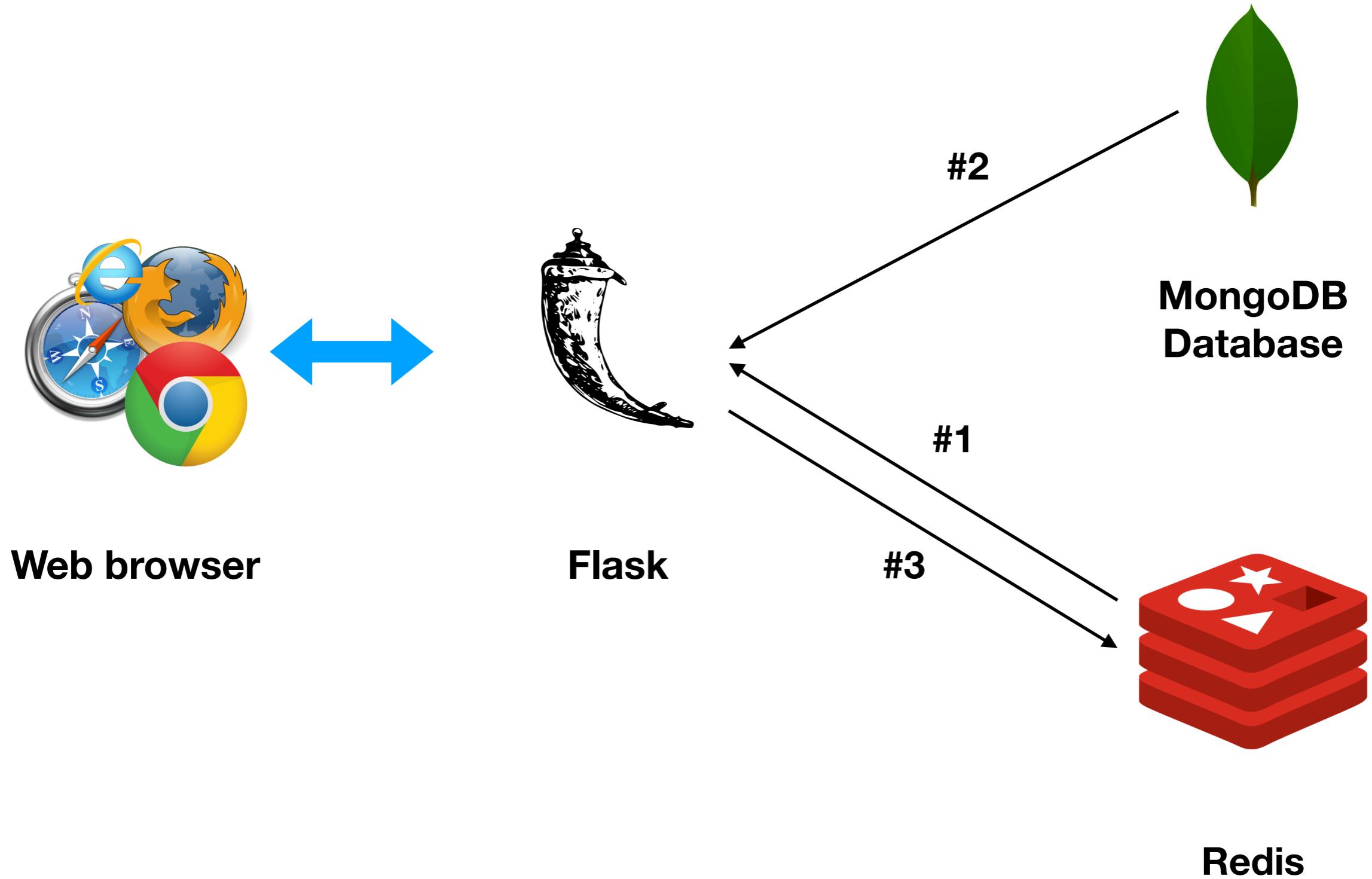
Web Application Architecture



Cache. Why?

- Faster delivery
- Reduce load

Web Application Architecture



Redis advantages

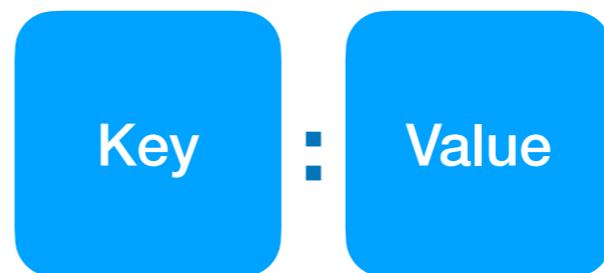
Redis

- In-memory
- Different data structures
- On-disk persistence
- Replication, transactions, high availability

Key-value

Data model

- Key
 - ASCII
- Value
 - Strings, Hashes, Lists, Sets, Sorted sets



SET/GET

```
127.0.0.1:6379> set user:1 '{"password": 123, "age": 20}'
```

```
OK
```

```
127.0.0.1:6379> set user:2 '{"password": 123, "age": 21}'
```

```
OK
```

```
127.0.0.1:6379> set user:3 '{"password": 123, "age": 22}'
```

```
OK
```

```
127.0.0.1:6379> GET user:1
```

```
"{ \"password\": 123, \"age\": 20}"
```

HSET

```
127.0.0.1:6379> hset users 1 '{"password": 123, "age": 20}' → (integer) 1
```

```
127.0.0.1:6379> hset users 2 '{"password": 123, "age": 21}' → (integer) 1
```

```
127.0.0.1:6379> hset users 3 '{"password": 123, "age": 22}' → (integer) 1
```

```
127.0.0.1:6379> hexists users 1 → (integer) 1
```

```
127.0.0.1:6379> hexists users 333 → (integer) 0
```

```
127.0.0.1:6379> hget users 1 → '{"password": 123, "age": 20}'
```

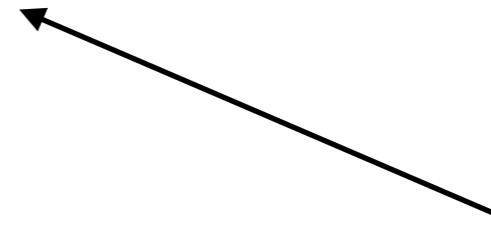
```
127.0.0.1:6379> hgetall users →
```

```
"{\"password\": 123, \"age\": 20}"  
1) "1"  
2) "{\"password\": 123, \"age\": 20}"  
3) "2"  
4) "{\"password\": 123, \"age\": 21}"  
5) "3"  
6) "{\"password\": 123, \"age\": 22}"
```

Redis in Flask

```
from flask_caching import Cache
...
app = Flask(__name__)
cache = Cache(app, config={
    'CACHE_TYPE': 'redis',
    'CACHE_REDIS_URL': 'redis://redis:6379/0'
} )

@app.route("/<int:num>")
@cache.cached(timeout=10)
def index(num):
    for i in range(2, num):
        if num % i == 0:
            return f"Found divisor: {i}"
```



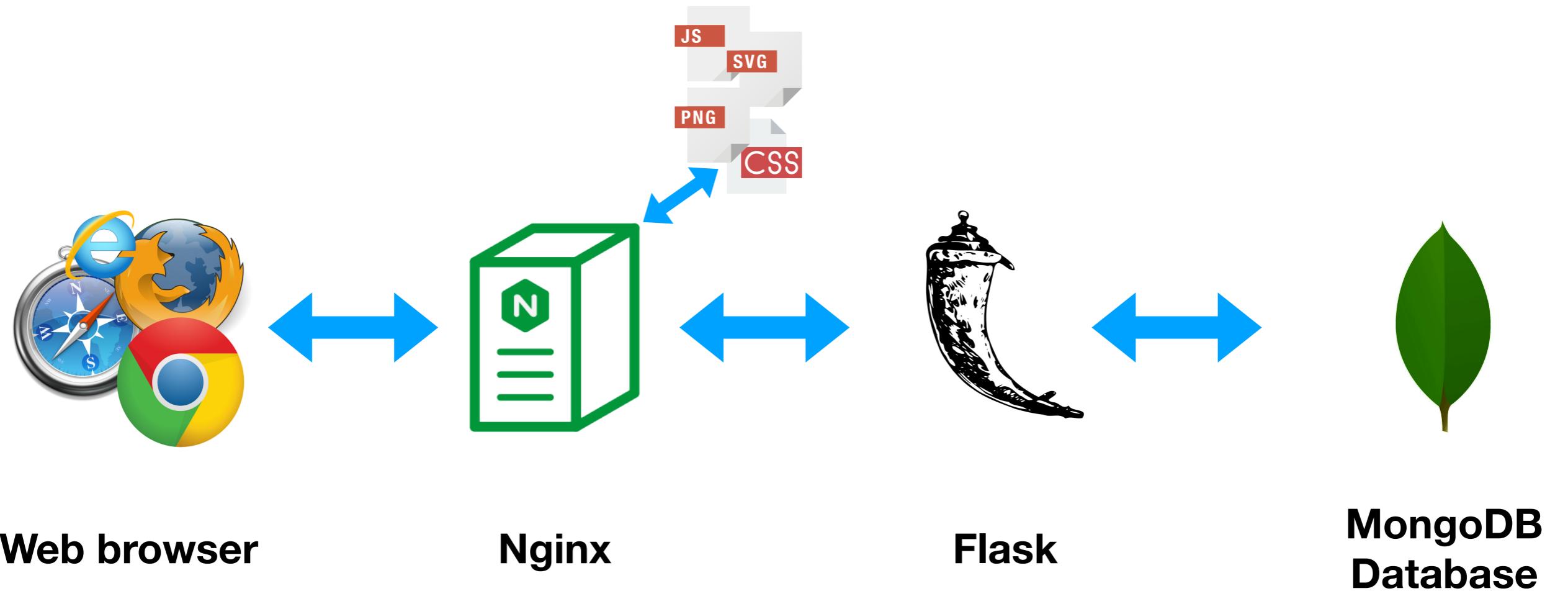
localhost?

Redis in Docker

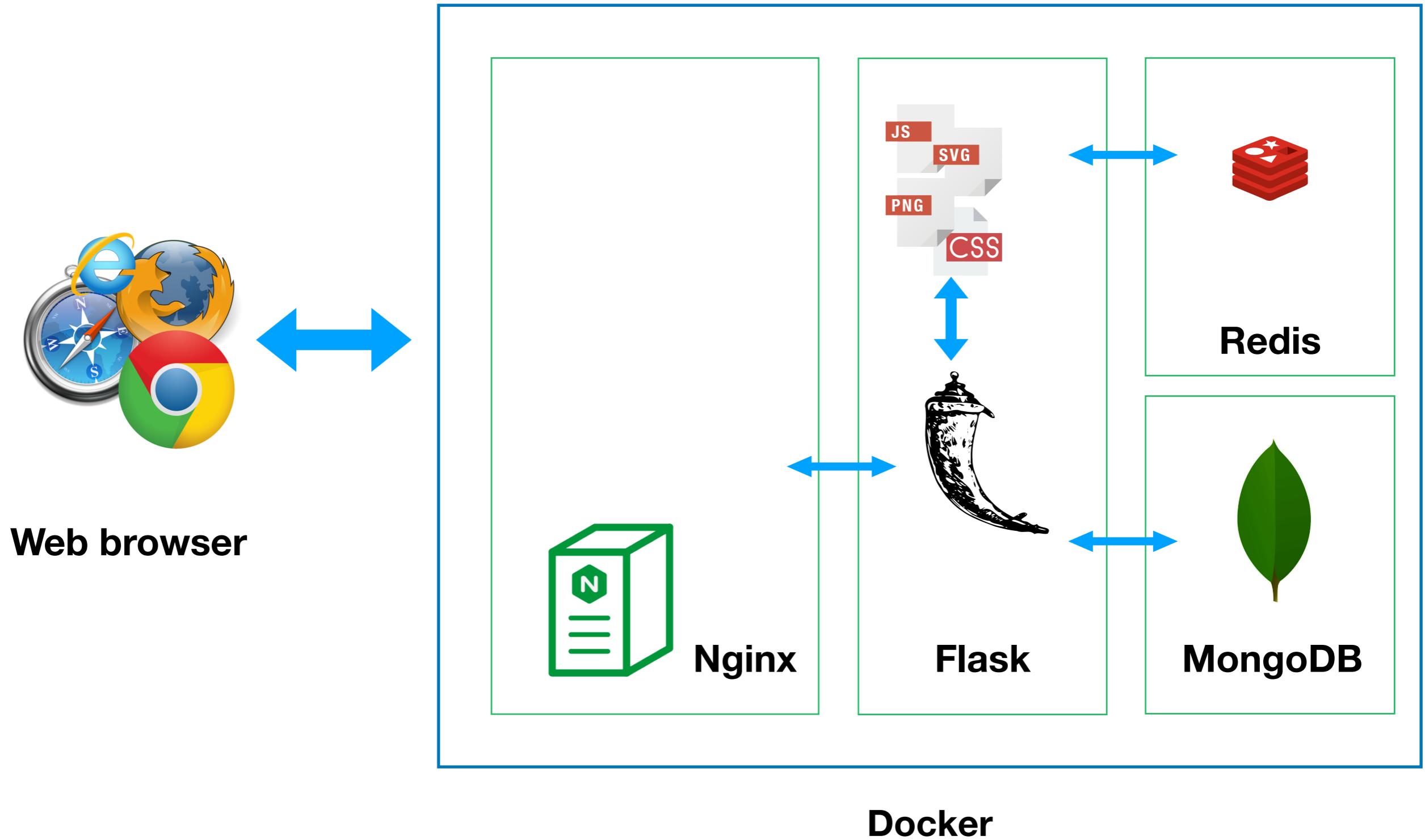
```
version: "3.2"
services:
  flask-simple:
    build:
      dockerfile: Dockerfile1
      context: .
    command: python app.py
    ports:
      - "80:5000"
    volumes:
      - ./src:/usr/src/app/
  redis:
    image: redis
    volumes:
      - redisdata:/data
    ports:
      - "6379:6379"
volumes:
  redisdata:
```

```
> docker-compose ps
> docker exec -it
root_redis_1 redis-cli
```

Old Architecture



New Architecture



Literature

- Redis commands: <https://redis.io/commands/>

Demo

Assignment #6

Task #6 - Docker

Coding → Deploy → Code-review

Coding

Preparations

1. Install Python programming language v3.8.2: <https://www.python.org/downloads/>
2. Install Docker: <https://docs.python.org/3/installing/index.html#basic-usage>
3. Install Docker-compose: <https://docs.docker.com/compose/install/>

What you already have after task #5

1. Flask web application, which can authenticate user with password:
 - Listen on `localhost:5000`
 - Render authentication form on `http://localhost:5000/`
 - Return static images and files on `http://localhost:5000/static/<image_name>`
 - Has secret page for authenticated users on `http://localhost:5000/cabinet`
2. Valid usernames and passwords are stored in MongoDB database
3. Image upload function in cabinet `http://localhost:5000/cabinet/`
4. Images are being saved to `upload` folder

Basic part

1. Create `docker-compose.yml` and `Dockerfile` to run your application in Docker
2. Move your sources to `src` directory. Don't forget to create `requirements.txt` file
3. In `docker-compose.yml` there are gonna be two containers named: `mongodb` , `flask-simple`

Optimal part

1. Add persistent volume for MongoDB
2. Mount `upload` directory to `upload` directory inside `flask-simple` container

Challenging part

1. Add Nginx which will proxy requests to flask application
2. Add Redis cache for uploaded images

Deploy

1. Register on GitHub: <https://github.com/>
2. Join our organization: <https://github.com/itmo-wad/>
3. You can create new personal repository or use repository for the previous tasks (but don't remove old files, just append new directories)
4. Commit and push your sources to GitHub. And don't forget to describe shortly what have you done in `README.md` file. Use Markdown format:
<https://guides.github.com/features/mastering-markdown/>

Code-review

1. Communicate in Telegram chat
2. Help others to complete the assignment

Teams

Team 1	Time picker	https://github.com/itmo-wad/time-picker
Team 2	Who when can  	https://github.com/itmo-wad/Who_when_can
Team 3	Ur-opinion  	https://github.com/itmo-wad/Ur-opinion
Team 4	CommentCloud  	https://github.com/itmo-wad/CommentCloud
Team 5	Cozy quiz  	
Team 6	Swiss knife  	https://github.com/itmo-wad/Swiss-knife
Team 7	Emoji picker  	https://github.com/itmo-wad/Emoji-picker

Practice time