Software Engineering 101

Building Dockerized API

https://github.com/joelluijmes/Workshop-JADS

whoami

- Joël Luijmes
- >5 years (part-time) working experience
  - Topicus - Software Development
  - Gynzy - DevOps / Data Engineer

Content

- Demo
- Basic Principles
- HTTP
- “Hello World!”
- Docker Essentials
- Writing an API

Why?

You want to deliver a **product**, not a notebook.

“An **application programming interface (API)** is an interface or communication protocol between a client and a server intended to simplify the building of client-side software.”

https://en.wikipedia.org/wiki/Application_programming_interface

Python HTTP Frameworks

- Flask
- Django
- **aiohttp**
- Many, many more: https://hackr.io/blog/python-frameworks

What is a process?

Basic Principles

“In computing, a **process** is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity. ”

https://en.wikipedia.org/wiki/Process_(computing)

“A **central processing unit** (**CPU**), also called a **central processor** or **main processor**, is the electronic circuitry within a computer that carries out the instructions of a computer program …”

https://en.wikipedia.org/wiki/Central_processing_unit

I/O-bound vs CPU-bound

Basic Principles

I/O

- I/O = Input / output
- Communication between systems / programs / devices
  - Mouse; webcam; speakers
  - RAM; Disk; Network

I/O-Bound or CPU-Bound?

- Reading from database
- Discovering the next prime number
- Making HTTP request to remote service
- Parsing HTML files to scrape information
- Training deep-learning model for cat detection

Blocking operations

Basic Principles

“A process that is **blocked** is one that is waiting for some event, such as a resource becoming available or the completion of an I/O operation.”

https://en.wikipedia.org/wiki/Blocking_(computing)

Blocking Example

00_request_blocking.py > ...

```python
import requests
from pprint import pprint

r = requests.get('https://httpbin.org/delay/2')
pprint(r.json())

```

Multi-threading

Basic Principles

“In computer architecture, **multithreading** is the ability of a central processing unit (CPU) (or a single core in a multi-core processor) to provide multiple threads of execution concurrently, supported by the operating system.”

https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)

Asynchronous

Basic Principles

Callback

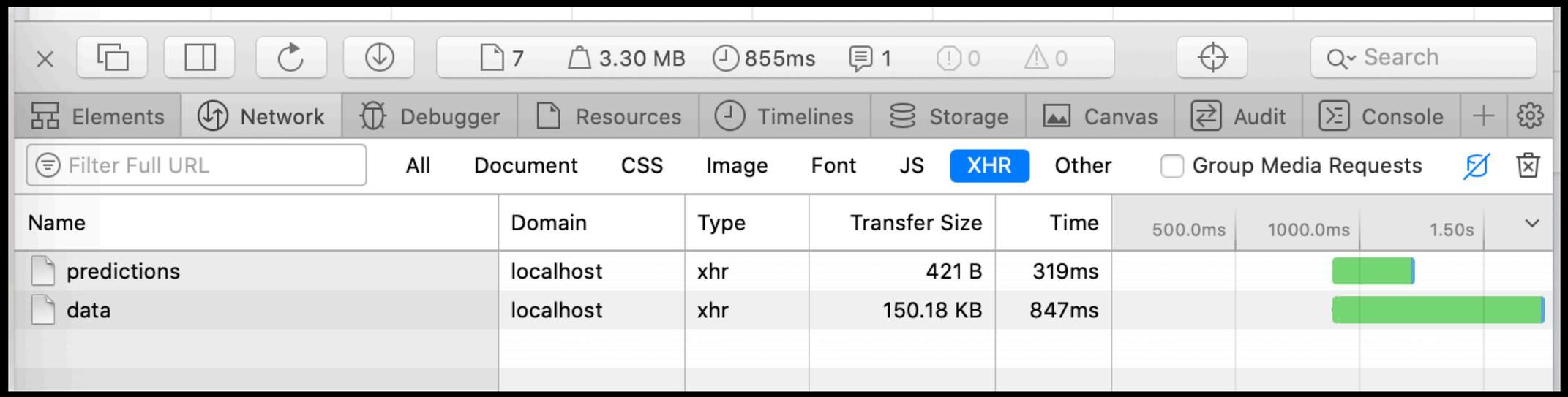
```
this.httpService.get(`https://httpbin.org/get`)
    .subscribe(() => {
        console.log('CALLBACK');
    });

console.log('REST OF THE PROGRAM');
```

https://nodejs.dev/javascript-asynchronous-programming-and-callbacks

Multi-tasking

```
52      Promise.all([
53        axios.get("http://localhost:7001/data"),
54        axios.get("http://localhost:7001/predictions?days=7")
55      ]).then(([{ data: weather }, { data: predictions }]) => {
56        // Do stuff
57      });
```



Callback Based

TS 01_request_callback.ts > ...

```ts
this.http.get('http://oauth/refresh-token', (token) => {
    this.http.post('http://ai/train-model', token, (model) => {
        this.mongo.getConnection((connection) => {
            connection.getCollection('models').update(model, () => {
                console.log('model updated! 😀')
            });
        });
    });
});
```

“In computer programming, the **async/await pattern** is a syntactic feature of many programming languages that allows an asynchronous, non-blocking function to be structured in a way similar to an ordinary synchronous function.”

https://en.wikipedia.org/wiki/Async/await

Await

TS 01_request_callback.ts > ...

```ts
this.http.get('http://oauth/refresh-token', (token) => {
    this.http.post('http://ai/train-model', token, (model) => {
        this.mongo.getConnection((connection) => {
            connection.getCollection('models').update(model, () => {
                console.log('model updated! 😀')
            });
        });
    });
});
```

TS 02_request_async.ts > ...

```ts
const token = await this.http.get("http://oauth/refresh-token");
const model = await this.http.post("http://ai/train-model", token);

const connection = await this.mongo.getConnection();
connection.getCollection("models").update(model);
console.log("model updated! 😀");
```

HTTP

HTTP

- HyperText Transfer Protocol

- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP

```
> GET /predictions?days=7 HTTP/1.1
> Host: localhost:7001
> User-Agent: insomnia/7.0.3
> Accept: */*

< HTTP/1.1 200 OK
< Content-Type: application/json; charset=utf-8
< Content-Length: 133
< Date: Sat, 09 Nov 2019 15:29:03 GMT
< Server: Python/3.6 aiohttp/3.5.4
```

HTTP

- Methods:
  - POST / GET / PUT / DELETE
- Status Codes:
  - 2XX Success (200 OK; 201 Created; 204 No Content)
  - 3XX Redirect (301 Permanent; 302 Temporary)
  - 4XX Client Error (400 Bad Request; 401 Unauthorised)
  - 5XX Server Error (500 Internal; 502 Bad Gateway)

- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

REST

REST

- REpresentation State Transfer
- https://www.codecademy.com/articles/what-is-rest
- https://en.wikipedia.org/wiki/Representational_state_transfer

“Hello World!”

Weather API

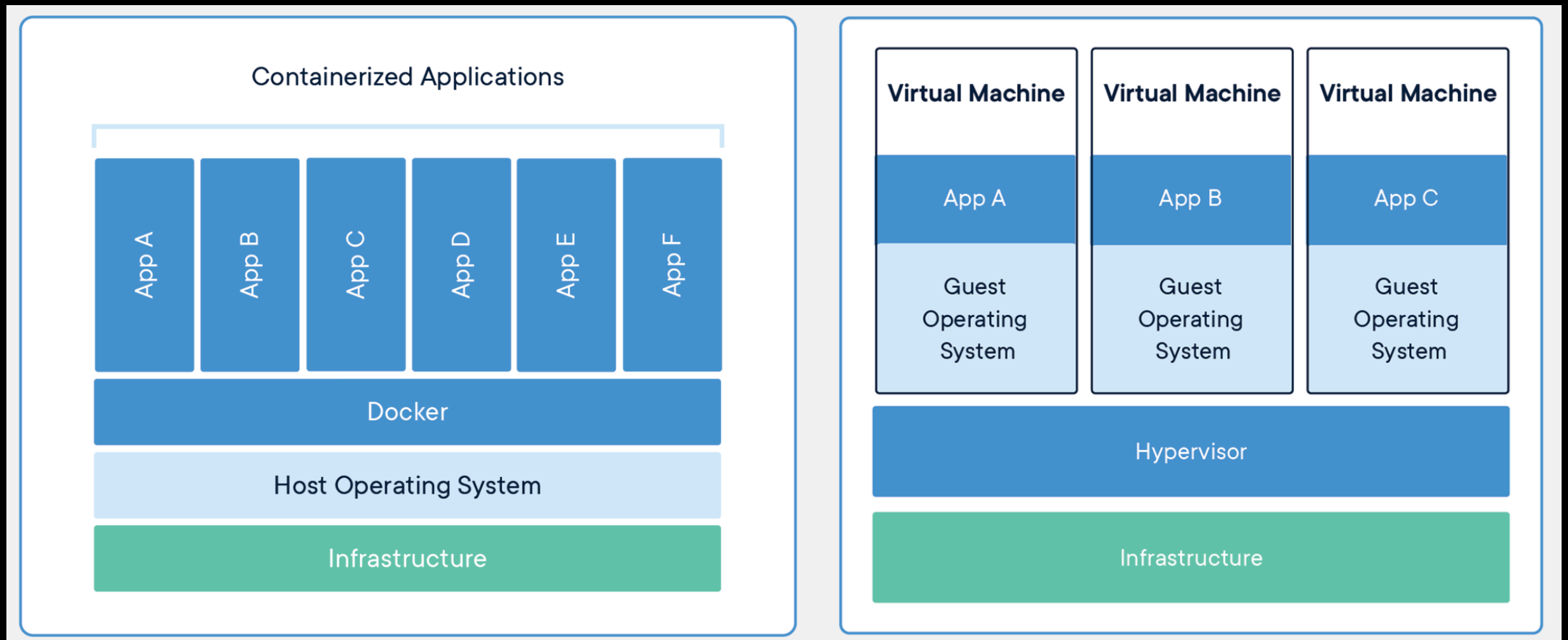
- Route links a resource (url) to function
- Wrap synchronous CPU-bound operation by delegating to different thread
- CORS allows different host to access API
- https://docs.aiohttp.org/en/stable/index.html
- https://docs.python.org/3/library/asyncio.html

Docker

“A **container** is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.”

https://www.docker.com/resources/what-container

Container vs VM



- Source: https://www.docker.com/resources/what-container

Docker Networking

- Windows Docker Toolbox - IP is mentioned when starting
- *Real* Docker - localhost
- https://docs.docker.com/network/

Docker References

- https://docs.docker.com/engine/reference/commandline/cli/
- https://docs.docker.com/engine/reference/builder/
- https://docs.docker.com/compose/compose-file/
- https://hub.docker.com/search?q=&type=image