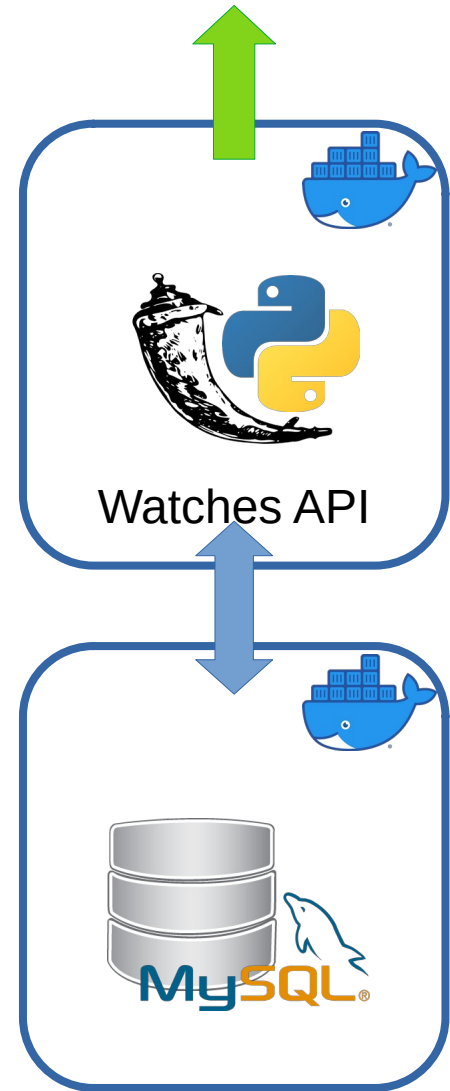# CLOUD COMPUTING PROJECT

WATCHES

WEBSERVICES

PART I

Part I - Objectives

- Develop a watch info service API
  - OpenAPI Spec
  - MySQL Data
  - Flask / Python

- Containerize
  - Docker
    - Rest API
    - MySQL

Watches API

Gitlab

- Project's files repository
  - Please insert your Gitlab username in this shared document to get access
    - https://docs.google.com/spreadsheets/d/11q2XYMPFUqAThhYi2do9kFkhycyQaiYCMZQn83OIDj0/edit?usp=sharing
  - Once access is granted
    - `$ git clone git@gitlab.com:lleonini/cloudcomputing-2021.git`
    - `$ cd cloudcomputing-2021/project`
  - Description of the API
    - `info_openapi_v1.yaml`
      - OpenAPI v3 (OAS3)
    - `watches.sql`
      - MySQL data
  - Material for the next parts will also be published in this repository

Swagger

- A tool to create and test OpenAPI specifications

- https://editor.swagger.io/

MySQL Data

| 📋 Browse | 🔧 Structure | 📄 SQL | 🔍 Search | 📥 Insert | 📤 Export | 📥 Import | 🔒 Privileges | 🔧 Operations | ⊙ Tracking | ⊞ Triggers |

✅ Showing rows 0 - 24 (4136 total, Query took 0.0011 seconds.)

SELECT * FROM `watches`

| 1 | > | >> | | Number of rows: | 25 ▾ | | Filter rows: | Search this table | | Sort by key: | None | ▾ |

+ Options

				sku	type	status	gender	year	dial_material	dial_color	case_material	case_form	bracelet_material	movement
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	ACBF2180	chrono	old	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	ACBF2A80	chrono	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	ACBF5A80	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	ACBF5A81	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	ACBF5A82	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	AWBF2180	watch	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	AWBF2A80	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	AWBF2A81	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1110.BA0850	chrono	old	man	2003	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1110.BT0705	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1111.BA0850	chrono	old	man	2003	STANDARD	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1111.BT0705	chrono	old	man	2004	STANDARD	WHITE	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1112.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1112.BT0705	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1113.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC111A.BA0850	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC111B.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC111D.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC111D.BT0705	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1310.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1310.FC6219	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	NIZZA	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1311.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAC1311.FC6220	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	NIZZA	MVT_QUARTZ
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAD5110.FC6177	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	LEATHER ALLIGATOR	CALIBRE_36
☐	🖉 Edit	⊐⊏ Copy	⊖ Delete	CAF1010.BA0821	chrono	old	man	2007	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ

MySQL Schema

- Data directly maps to API spec

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	sku 🔑	varchar(255)	utf8_unicode_ci		No	None			✎ Change ⊖ Drop ▼ More
2	type	set('watch', 'chrono')	utf8_unicode_ci		No	watch			✎ Change ⊖ Drop ▼ More
3	status	set('old', 'current', 'outlet')	utf8_unicode_ci		No	current			✎ Change ⊖ Drop ▼ More
4	gender	enum('man', 'woman')	utf8_unicode_ci		No	None			✎ Change ⊖ Drop ▼ More
5	year	int(11)			No	None			✎ Change ⊖ Drop ▼ More
6	dial_material	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More
7	dial_color	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More
8	case_material	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More
9	case_form	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More
10	bracelet_material	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More
11	movement	varchar(255)	utf8_unicode_ci		Yes	None			✎ Change ⊖ Drop ▼ More

Step #1 - Local install

- Install MySQL
  - `$ sudo apt install mysql-server mysql-client`
  - Install PHPMyAdmin to create a new user and database
    - `$ sudo apt install phpmyadmin`
  - Load the data (CLI or PHPMyAdmin)
    - `$ mysql -u <username> -p <database> < watches.sql`

- Install Python 3 & pip3
  - `$ sudo apt install python3 python3-pip`
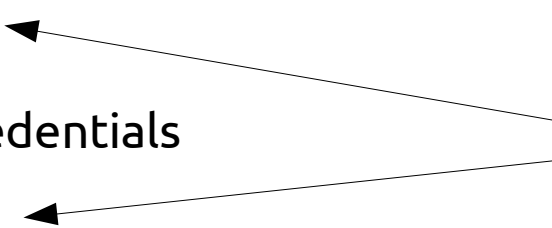
Step #2 - Local dev

- Develop your API in Python
  - Use pip3 for dependencies management
    - https://pip.pypa.io/en/stable/
    - Dependencies in: **requirements.txt**
      - Flask
      - PyMySQL
  - Server (single file): **server.py**
    - Listen on port **1080** (not privileged)

```
if __name__ == "__main__":
    app.run(port=1080)
```

Step #3 - ENV vars

- DB parameters
  - DB_HOST=127.0.0.1
  - DB_PORT=3306
  - DB_DBNAME=watches
  - DB_USER=watches
  - DB_PASS=watches
- HTTP basic auth credentials
  - HTTP_USER=cloud
  - HTTP_PASS=computing
- Create a shell script **run.sh**
  - Set ENV vars
  - Start: server.py
    - Using flask run

> Use exactly
> these ENV vars
> and auth values !

Step #4 - Validate API

- Using Swagger
  - https://editor.swagger.io/
  - Load info_openapi_v1.yaml
  - Set authorization
  - **Test all endpoints**
    - Curl commands are also generated
      - Adapt the port to use 1080 instead of 80
- Do not proceed with next steps until your API works as expected

Step #5 - Optimizations

- 1. Add indexes to the DB Data
  - In order to improve lookup speed
  - Update `watches.sql` with your changes (dump the DB)

- 2. Set HTTP expiration headers
  - All data GET should be valid 1 hour

Step #6 - Info-service in Docker

- Create a Dockerfile
  - Embed your Python development in a Docker image
  - Default command: start the server (**without using run.sh** in this case !)
  - $ docker build -t **info-service-v1** .
- $ docker run -d -p 1080:1080 --network=host \

      -e "HTTP_USER=cloud" \

      -e "HTTP_PASS=computing" \

      -e "DB_HOST=127.0.0.1" \

      -e "DB_PORT=3306" \

      -e "DB_DBNAME=watches" \

      -e "DB_USER=watches" \

      -e "DB_PASS=watches" \

  info-service-v1

> Using `--network=host`,
> your docker instance should
> be able to connect directly
> to the MySQL instance
> running on host machine

Step #7 - MySQL in Docker

- Use https://hub.docker.com/_/mysql/
  - Read the documentation
    - See how to load `watches.sql` at startup or via an external **volume**
  - If you have authentication problems, use MySQL version 5.7 instead of latest (8.0)
    - The "old" authentication mechanism is disabled by default in images with newer versions
    - More details and solutions:
      - https://medium.com/@crmcmullen/how-to-run-mysql-8-0-with-native-password-authentication-502de5bac661

Step #8 - Compose

- Write **docker-compose.yml**
  - Set ENV vars
  - Run images
    - `info-service-v1`
    - `mysql`
    - `PHPMyAdmin` (optionally to view database)
  - Connect them together
  - Persist DB data
- `$ docker-compose up`
  - Everything should start and work
  - More info soon about docker compose

Deliverables

- Python server
  - `server.py`
  - `requirements.txt`
  - `run.sh`
  - `watches.sql`
- Docker
  - `Dockerfile → info-service-v1`
  - `docker-compose.yml`
- README
  - Name of all participants (if group)
  - Comments to help running the project (if needed)

Committing

- Push your development in your Gitlab assignment repository
  - `/project` folder
  - Commit your work <u>step by step</u> and <u>regularly</u> with commit messages
  - Once the first part of the project is finished, `tag` the commit '`part1`'
  - If working by team (max 3 students)
    - Indicate all the participants and the main repository to assistants
    - Every member of the team should commit regularly using their own user
  - Grant access and share information with Rémi and Jämes until next week

Delay

- Part I (4 weeks)
  - Documentation: TODAY
  - Deadline: 2021-04-14T23:59:59+02:00
- Part II (3 weeks)
  - Documentation: 2021-04-15
  - Deadline: 2021-05-05T23:59:59+02:00
- Part III (3 weeks)
  - Documentation: 2021-05-06
  - Deadline: 2021-05-26T23:59:59+02:00