

CLOUD COMPUTING PROJECT

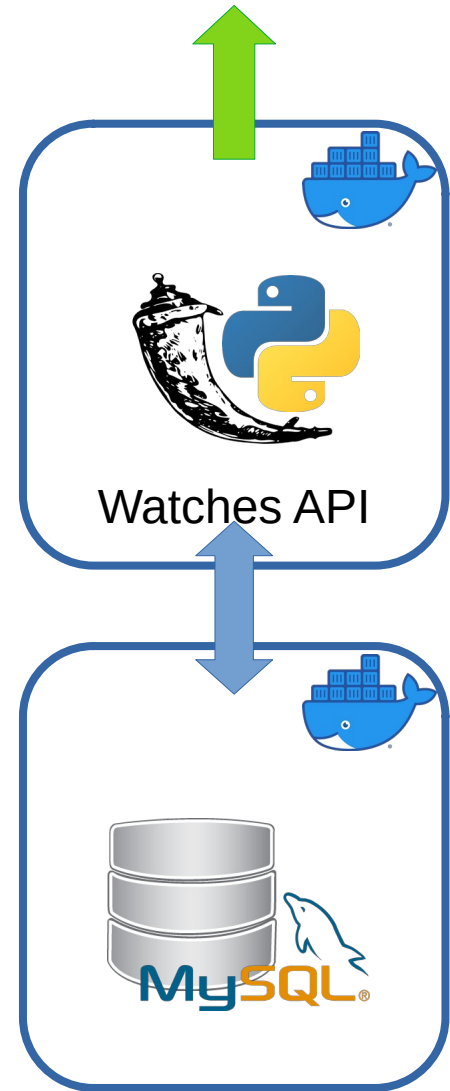
WATCHES

WEBSERVICES

PART I

Part I - Objectives

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



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/>

The screenshot displays the Swagger Editor web application. The left pane shows the OpenAPI specification in YAML format, and the right pane shows the rendered API documentation.

```
1 openapi: 3.0.0
2 info:
3   title: Watch info service
4   version: '1.0'
5 servers:
6   - url: 'http://localhost/watches/v1'
7 security:
8   - basicAuth: []
9 paths:
10  /watch:
11    post:
12      summary: Add a new watch to the store
13      requestBody:
14        description: Watch object that needs to be added
15        to the store
16        required: true
17        content:
18          application/json:
19            schema:
20              $ref: '#/components/schemas/Watch'
21      responses:
22        '200':
23          description: Successful operation
24        '400':
25          description: Invalid input
26  '/watch/{sku}':
27    get:
28      summary: Return watch data
29      parameters:
30        - name: sku
31          in: path
32          description: SKU of the watch to return
33          required: true
34          schema:
35            type: string
36      responses:
37        '200':
38          description: Successful operation
39          content:
40            application/json:
41              schema:
42                $ref: '#/components/schemas/Watch'
43        '404':
44          description: Watch not found
45    put:
46      summary: Updates a watch in the store with form
47      data
48      parameters:
49        - name: sku
```

Watch info service 1.0 OAS3

Servers
http://localhost/watches/v1 **Authorize**

default

- POST** /watch Add a new watch to the store
- GET** /watch/{sku} Return watch data
- PUT** /watch/{sku} Updates a watch in the store with form data
- DELETE** /watch/{sku} Deletes a watch
- GET** /watch/complete-sku/{prefix} Get list of SKUs matching a prefix
- GET** /watch/find Finds watches by any criteria

Schemas

- Watch

MySQL Data

Server: localhost » Database: cloud » Table: watches											
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
<input type="checkbox"/> Edit Copy Delete	ACBF2180	chrono	old	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
<input type="checkbox"/> Edit Copy Delete	ACBF2A80	chrono	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
<input type="checkbox"/> Edit Copy Delete	ACBF5A80	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
<input type="checkbox"/> Edit Copy Delete	ACBF5A81	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
<input type="checkbox"/> Edit Copy Delete	ACBF5A82	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
<input type="checkbox"/> Edit Copy Delete	AWBF2180	watch	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
<input type="checkbox"/> Edit Copy Delete	AWBF2A80	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
<input type="checkbox"/> Edit Copy Delete	AWBF2A81	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
<input type="checkbox"/> Edit Copy Delete	CAC1110.BA0850	chrono	old	man	2003	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1110.BT0705	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1111.BA0850	chrono	old	man	2003	STANDARD	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1111.BT0705	chrono	old	man	2004	STANDARD	WHITE	STEEL	ROUND	RUBBER	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1112.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1112.BT0705	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	RUBBER	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1113.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC111A.BA0850	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC111B.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC111D.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC111D.BT0705	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1310.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1310.FC6219	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	NIZZA	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1311.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	STEEL	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAC1311.FC6220	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	NIZZA	MVT_QUARTZ
<input type="checkbox"/> Edit Copy Delete	CAD5110.FC6177	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	LEATHER ALLIGATOR	CALIBRE_36
<input type="checkbox"/> Edit Copy Delete	CAF1010.BA0821	chrono	old	man	2007	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ

MySQL Schema

- Data directly maps to API spec

Server: localhost » Database: cloud » Table: watches

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Tracking](#)

[Table structure](#) [Relation view](#)

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

☐ Check all With selected: [Browse](#) Change Drop Primary Unique Index Fulltext Add to central

Print Propose table structure Track table Move columns Normalize

Add column(s)

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	sku	4221	A	No	

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`

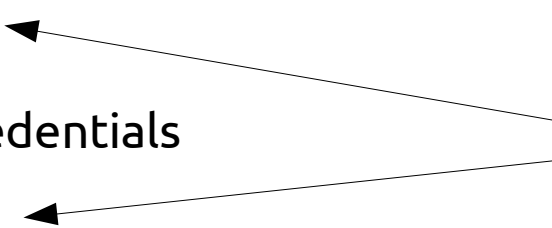
Step #2 - Local dev

- 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/@crmcullen/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 step by step and regularly 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