MICROSERVICES
ATYOURSERVICE

Bridging the gap between NLP Research and Industry

Docker & Web Service API
Intro Workshop

Lingsoft®
LANGUAGE
MANAGEMENT

14 June 2021

Lingsoft®

Schedule Day 1

Activity	Helsinki Time
Tools and terms intro	11:00-11:45
Docker & API, example 1	11:45-12:15
Break	12:15-13:15
Docker & API, example 2	13:15-13:45
Q&A	13:45-14:00

Lingsoft®

Schedule Day 2

Activity	Helsinki Time
In case we didn’t have time for something on Day 1	14:00-14:30
Q&A - Bring your problem - We’ll try to help	14:00-16:00

SaaS < PaaS < IaaS

- Google Cloud, Azure, AWS
- And, the European Language Grid (ELG)
    - "Our main objective is to address fragmentation in the European Language Technology business and research landscape by **establishing the ELG as the primary platform for Language Technology in Europe** and to strengthen European LT business with regard to the competition from other continents."
    - "The ELG will be **a platform for commercial and non-commercial Language Technologies**, both functional (running services and tools) and non-functional (data sets, resources, models)."

Project Objectives

1. Identify suitable (open source) NLP tools
2. Dockerise the NLP tools
3. Share information about the methods and results

Co-financed by the Connecting Europe Facility of the European Union

Lingsoft®

Objective 2: Dockerize NLP Tools

1. **Add a web service API to the tool**
2. **Create a Docker image for the tool**
3. Create how-to instructions for new users
4. Store the Docker image in a docker registry
5. Integrate the image with the European Language Grid
6. List the image on ELRC-SHARE repository

Lingsoft®

Web Service API & Docker Workshop

Lingsoft

Pre-Workshop

- **Find an open source tool that you would like to dockerise**, ideally one you are familiar with how it works.
    - In the simplest case, you have access to an installable script/package which does something with the tool (e.g. decode, annotate, …)
    - The workshop will focus on Python, but it shouldn't be too difficult for you to adapt it for e.g. Java
- Docker documentation: https://docs.docker.com/
- Docker getting started: https://docs.docker.com/get-started/
- Installing Docker engine: https://docs.docker.com/engine/install/
    - **Please, install** (at least) **Docker engine before the workshop**
    - The workshop examples are made in some Linux version
- **This workshop is aimed at getting beginners started.** More advanced people are welcome, especially to assist the organisers with helping the beginners

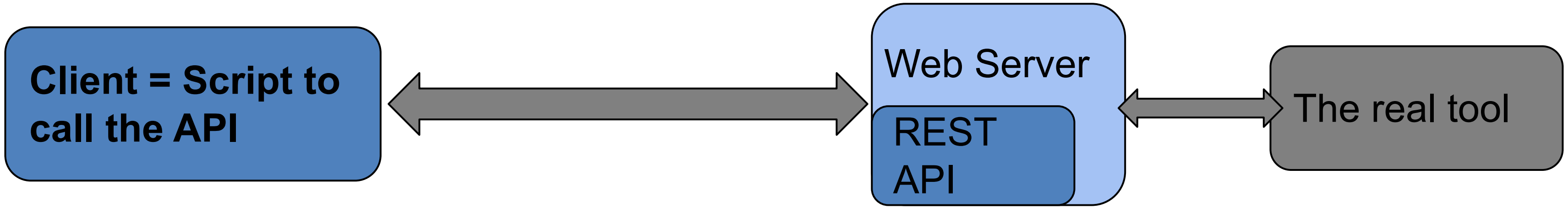
Workshop Target

- Introduce some terminology and tools
- Introduce some links to more material
- Get you prepared for trying it yourself

Disclaimer - this workshop does not include - but we'll mention some hints for the future

- CPU & memory - you want your image as small as possible
- Details on WebSocket, REST, … - Google it
- Nvidia-docker - if you got GPU tools
- Security aspects - is Docker safe?
- Docker Hub - like gitHub for docker images - needed for ELG
- Kubernetes or Docker compose - container orchestration & scalability

SaaS > Web Service API > REST API



Client	REST API
• Often a sequence of calls with e.g. ○ **Curl** - commandline linux ○ Python requests • Call structure: **Header(s), Data, Method, Address** • **curl -H** "Authorization: Bearer $TOKEN" -H "Content-Type: text/plain" --data-binary "@./path/**somefile**" -X **POST** **https://api.xyz.fi/REST/nmt**	• **HTTP end point** https://api.xyz.fi/REST/nmt • Defines which HTTP methods the client can call: ○ **POST** - send some data, e.g. text or file, to backend tool ○ **GET**, PUT, DELETE • Needs documentation for the client developer

Lingsoft®

Doing Your Own Web Service API

- The Application Programming Interface (API) - i.e. you need some tool behind the interface too
- The ELG API specs for the REST API for your tool
    - When things go well
    - When things go wrong
    - Standard codes (404 Not Found - sounds familiar?)
- The Python Flask package for implementing a REST API for your tool
- The ELG Python SDK, including a package for creating the docker & web service with Flask and an example

Lingsoft®

What is Docker?

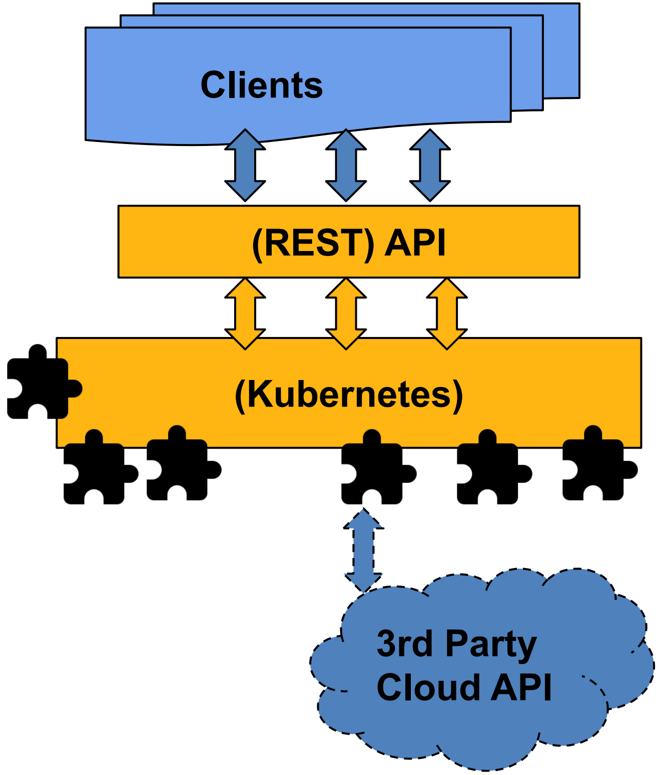
- Wikipedia says: “[Docker is] a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.”
- **Image** (the shareable software package) - **Container** (the running software package)

- Makes it easy to install tools with different dependencies/OS on same **host** machine
- Makes it easy for others to install & use the tool
  - Previous years machine translation/parsing/whatnot systems for scientific challenges
  - Previous years software development examples for students
  - ...
- Makes it easy to create microservice architecture

Lingsoft

Microservice Software Architecture

- 🧩 = NLP component in Docker container
- Easy integration of open source and 3rd party components
- Easy extension to new languages and functionality
- Easy replacement of components



Docker Images and Containers

- Docker **pull** ubuntu:20.04
  - Get your base image from a Docker registry e.g. https://hub.docker.com/
- **Dockerfile**
  - Usually starting from a “base” image, e.g. Ubuntu 20.04
  - List of instructions: tools to install, commands to run, ...
- Docker **build** --help
  - Build a distributable image from the Dockerfile
- Docker **run** --help
  - Create a running container with whatever you packed in the image, e.g. a text tagger, a machine translation web-demo, …
- Docker **push**
  - Publish your updated image on e.g. https://hub.docker.com/
- (That container orchestration thing that is not included in this workshop)

Lingsoft®

Some Useful Commands, part 1

Command	Comment
docker image/container list	Lists images/containers on the machine, and info on e.g. memory & connected port
docker ps -a	lists containers more thoroughly than above command
docker system df [-v]	How much space does the images/containers take. Good to check. One can easily fill up space by accident as a beginner.
docker image prune [-a]	Deletes unused “dangling” images and frees up space.
docker stop container_name/ID	Stops “pauses” a running container
docker rm container_name/ID	Deletes a container

Some Useful Commands, part 2

Command	**Comment**
Docker **run** --volume local_dir:container_dir --publish localport:containerport --name container_name image_name	Usually put in a shell script
Docker **exec** -it container_name bash	In my case, to get a bash prompt. But, you can use it to run (any?) commands in the container
Docker **attach** container_name	Attach stdin & stdout & stderr. See what's going on inside the container.
Ctrl-p ctrl-q	Detach ("hop off") from a running container. Ctrl-D might also do what you need.

Step-by-step: FinBERT simple

FinBERT Simple Flask & Docker Example

- The sample code was emailed in a zip file. If you did not get it, please email: sebastian.andersson@lingsoft.fi
- On some systems, you need sudo to use docker commands
- Check on your system with: (sudo) **docker image list**
  - If you don't have ubuntu 18.04 as image, then:
  - Docker **pull** ubuntu:18.04
    - It pulls it from https://hub.docker.com/
  - The example probably works with other ubuntu/python base images too, but then you need to Edit the first line "From…" in the Dockerfile

FinBERT Simple: Needed files

- In folder docker_example
  - Dockerfile #Instructions for docker build
  - serve.py #REST API definition & run web server
  - templates/index.html #needed by serve.py
  - templates/result.html #needed by serve.py
  - static/finbert.png #needed by serve.py

FinBERT Simple: Dockerfile

- **Dockerfile**

```
#Specify the base image
FROM ubuntu:18.04
#Install basic tools
RUN apt-get update -y
RUN apt-get install -y python3-pip python3-dev
RUN pip3 install --upgrade pip
RUN pip3 install happytransformer flask
#Copy files onto the container, and run the serve.py
EXPOSE 8866
COPY ./ ./
CMD ["python3", "serve.py"]
```

Lingsoft®

FinBERT Simple: build, run & test

1. Build the container. Needs to be run in the same folder as the Dockerfile: **sudo docker build -t finbert-demo .** (NOTE: the dot is needed)
2. Run the container in the background. Can be done in any folder. **sudo docker run -d -p 0.0.0.0:8866:8866 --name localbert-demo finbert-demo**
3. List containers: **sudo docker ps -a**
4. Test with curl: **curl -X POST -d 'sentencein=esimerkiksi SANA on viimeaikoina ollut esillä .' http://0.0.0.0:8866/predict_json**
5. More Finnish example sentences:
   - Tämän viikonloppuna vietetään pääsiäistä, johon kuuluu paljon SANA ja herkkuja.
   - Se oli silti yli 40 prosenttia korkeampi kuin viime viikon maanantaina, jolloin todettujen SANA määrä alkoi nousta.
   - Huomenna minä menen SANA .

Lingsoft®

Step-by-step: Turku Neural Parser Pipeline

Turku Neural Parser Pipeline

- Complex parser pipeline with several steps running as sub-processes and requiring a GPU to run fast enough
- Code:
  - https://github.com/TurkuNLP/Turku-neural-parser-pipeline/tree/diaparser
  - Note: “diaparser” branch is the correct one
  - This is a new branch which uses the diaparser dependency parser at its core
  - Not yet fully complete but will do for this tutorial, as it is much easier to install

Turku Neural Parser Pipeline

- Steps to install
  - Clone code from GitHub (possibly install using setup.py)
  - Pip-install required packages
  - Fetch a trained model for your language
- Steps to run
  - Directly in python via import
  - Let us have a look
  - https://github.com/TurkuNLP/Turku-neural-parser-pipeline/blob/diaparser/tnpp-parse
  - Simple http API done with flask
- Everything needed to run it is summarized here:
  - https://colab.research.google.com/github/TurkuNLP/Turku-neural-parser-pipeline/blob/diaparser/docs/tnpp_diaparse.ipynb

Lingsoft®

TNPP - flask

- Minimal flask APP to run the parser
- Let us walk through it:
- https://github.com/TurkuNLP/Turku-neural-parser-pipeline/blob/diaparser/tnpp_serve.py

Lingsoft®

TNPP - dockerfile

- A simple Docker file for the parser
- Let us walk through it
- https://github.com/TurkuNLP/Turku-neural-parser-pipeline/blob/diaparser/Dockerfile.server
- `docker build -f Dockerfile.server -t tnpp-fi-server .`
- This builds based on the .server docker file, tags the image as tnpp-fi-server, the build is based on the current directory (so this needs to be run in the top directory of the parser)

Lingsoft®

TNPP - run in docker

- `docker run -it -p 5000:7689 tnpp-fi-server`
- This will run interactive (easy to ctrl-c)
- Map container’s port 7689 onto local machine’s port 5000
- ...and there it should be running
- You can try GET on http://localhost:5000
- You can try POST like this:
- `curl -X POST -d 'Minulla on kissa' localhost:5000`

TNPP - GPU acceleration

- https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html
- https://ngc.nvidia.com/catalog
- You can base your image off one of these
- `docker build -f Dockerfile.server.gpu --build-arg MODEL=fi_tdt_dia -t tnpp-fi-server:latest-gpu2 .`
- The only change in the Dockerfile:

```
FROM nvcr.io/nvidia/pytorch:20.06-py3
```

- How to run:

```
nvidia-docker run --rm -it -p 5000:7689 tnpp-fi-server:latest-gpu2
```

Lingsoft®

Q & A from live sessions

Lingsoft®

Question	Answer
Is there a way to reduce build context?	Yes. https://docs.docker.com/engine/reference/builder/#dockerignore-file
Does the Cuda versions need to match between host & docker image?	No. But, the host GPU drivers needs to match the cuda version on the image. See also GPU example in this presentation.
How do I give a compiled version of my tool with the image?	In the Dockerfile, but there are many ways: add compile commands as instructions or copy the executable onto the image or… Example: WORKDIR MyThing/build RUN cmake MyThing && make -j
How does one reduce the image size?	There are many ways. Here are some guidelines for the Dockerfile using multi-stage build

Contact

Sebastian Andersson
Solution Architect
sebastian.andersson@lingsoft.fi

www.lingsoft.fi

Eteläranta 10, FI-00130 Helsinki, Finland

Kauppiaskatu 5A, FI-20100 Turku, Finland

+358 2 2793 300