

KI

Service
Zentrum



Hasso
Plattner
Institut

Digital Engineering • Universität Potsdam

Containerize your intelligence: A hands-on workshop on deploying AI models with Docker

Felix Boelter and Johanna Reiml

SPONSORED BY THE



Federal Ministry
of Education
and Research

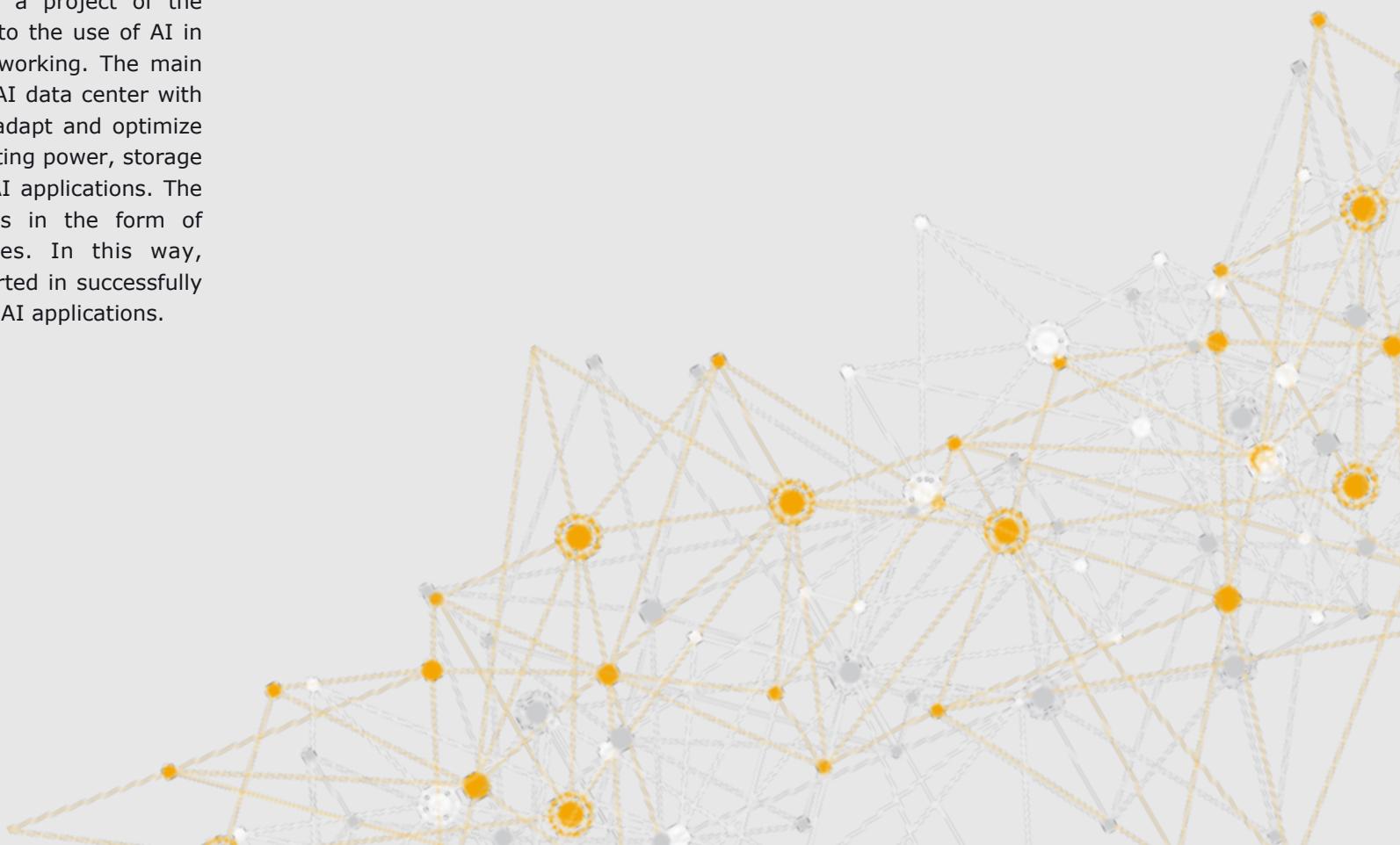
**Design IT.
Create Knowledge.**

www.hpi.de



KISZ-BB

The AI Service Center Berlin-Brandenburg (KISZ-BB) is a project of the Hasso-Plattner-Institute with the aim of lowering barriers to the use of AI in business and society through knowledge transfer and networking. The main research areas are operational research to investigate an AI data center with heterogeneous hardware and methodological research to adapt and optimize AI models. The KISZ-BB provides resources such as computing power, storage space, data and models for the development and use of AI applications. The KISZ-BB also offers educational and consulting services in the form of workshops, individual consultations and online courses. In this way, companies, start-ups and non-profit institutions are supported in successfully mastering the next steps towards the professionalization of AI applications.



AI Service Centers

MOTIVATION, TASK & SERVICES

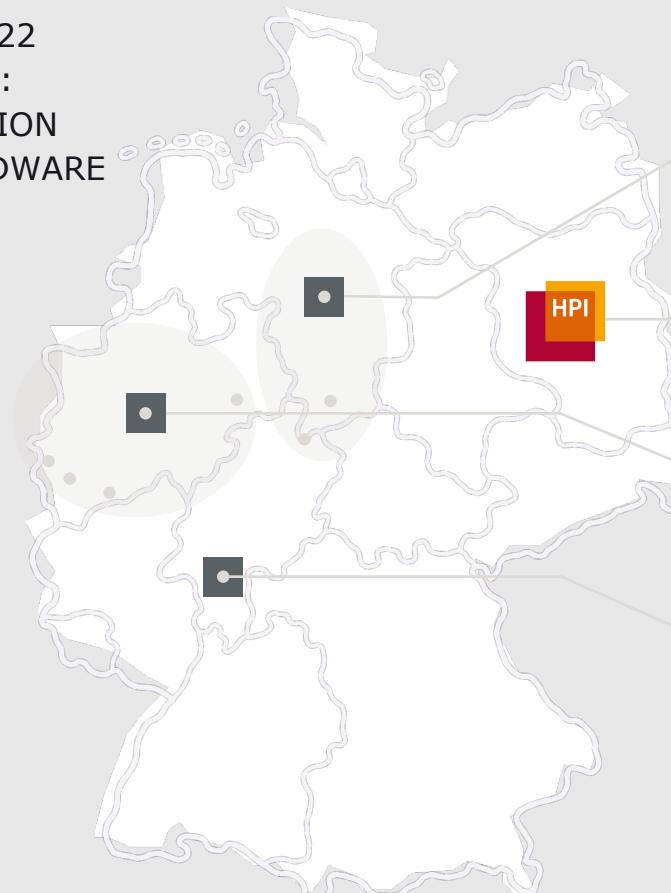
KICK-OFF: DEC 2022

FUNDING AMOUNT:

APPROX. €10 MILLION

17 PEOPLE + HARDWARE

5 SPECIALISTS



Germany



KISSKI

Hannover | Göttingen | Kassel
Sensitive and critical infrastructures
Focus: Medicine & Energy

KISZ-BB

Berlin | Brandenburg
Educational and advisory services
Use of AI in business and society

West AI

Dortmund | Bonn | Jülich | Aachen | Paderborn
Large and transferable AI models

hessian AI

Service Center

Darmstadt
Explainability, generalizability
and contextual adaptation

Discussion

Who are you?

Learning Goals

- Understand how we can use a **basic language detection model**
- Understand **APIs** and **FastAPI**
- Understand essential **Docker features**
- Be able to **deploy a basic ML application**

What we will build

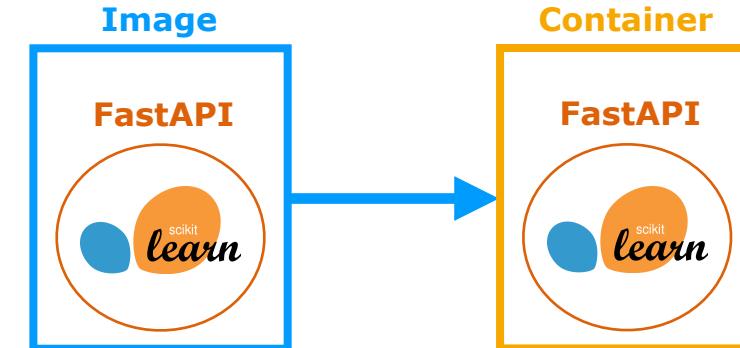
Step 1: Create Model API



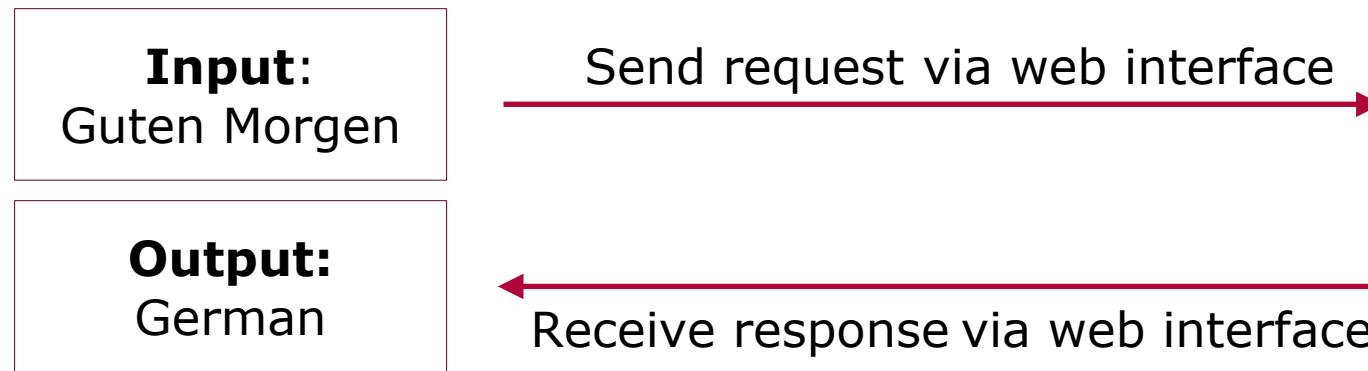
Step 2: Create Docker Image



Step 3: Build a Container and Deploy it



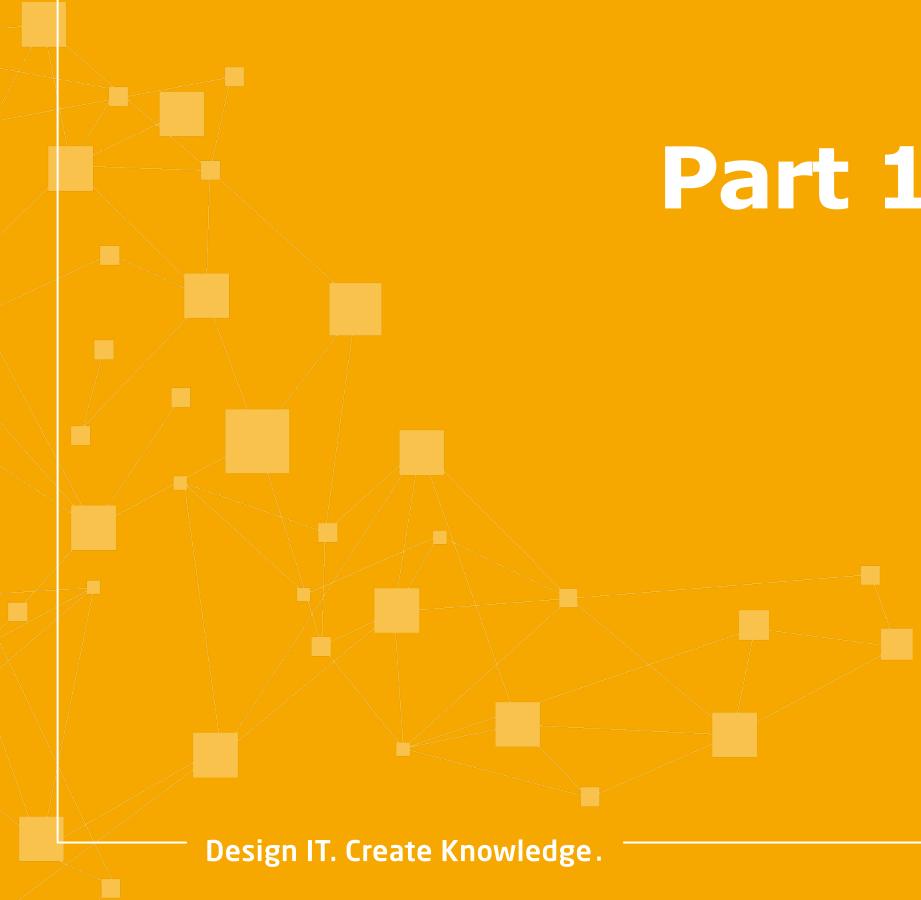
What we will build



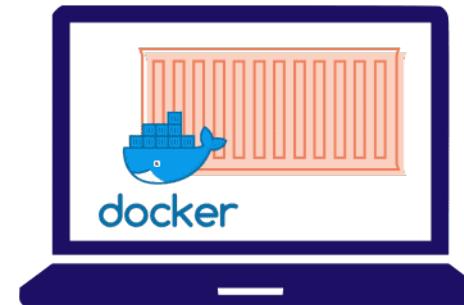
Hands-On: Setting up the repository

- Open your terminal
- Navigate to a directory for storing the workshop code using the `cd` command
- **Clone the repository:**
 - `git clone https://github.com/KISZ-BB/kisz-mlops-docker.git`
- **Navigate to the repository using:**
 - `cd kisz-mlops-docker`
- Check the contents with `ls` (Linux/Mac) or `dir` (Windows)
- Look at the `README.md` and the `handout.pdf`

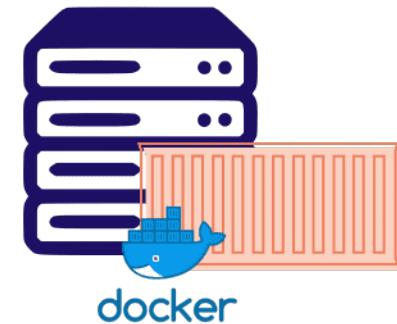
Part 1: Introduction to Docker



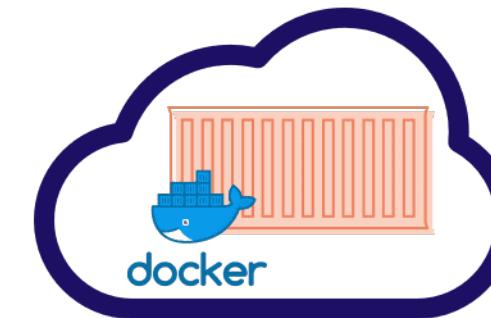
Background Talk: What is Docker?



” Works well
on my laptop

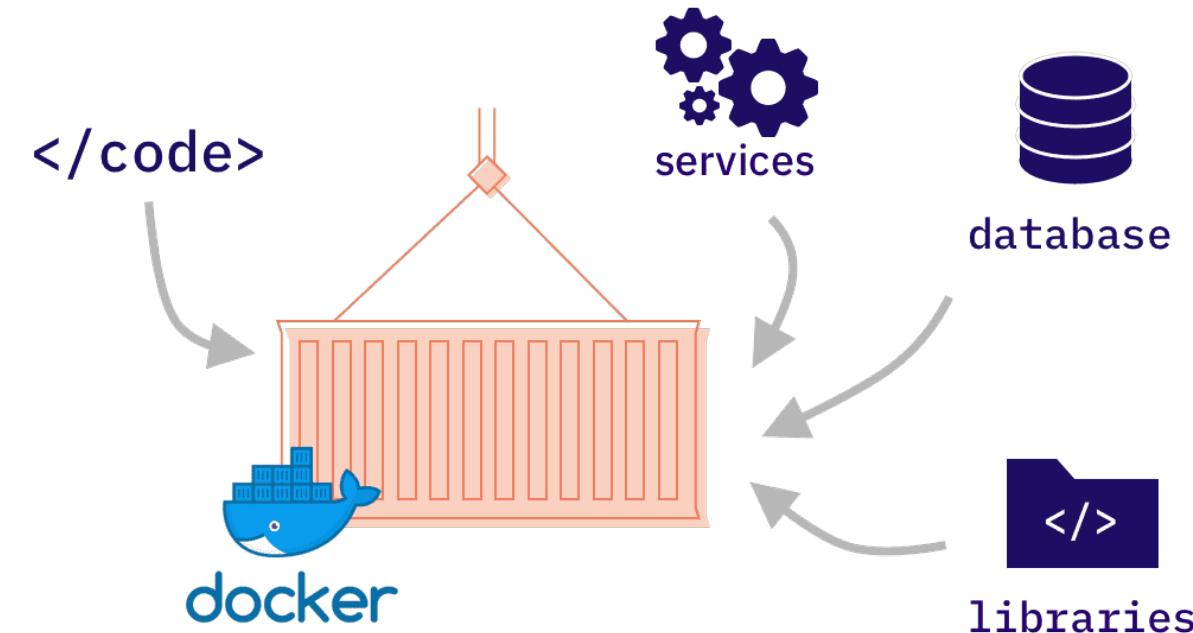
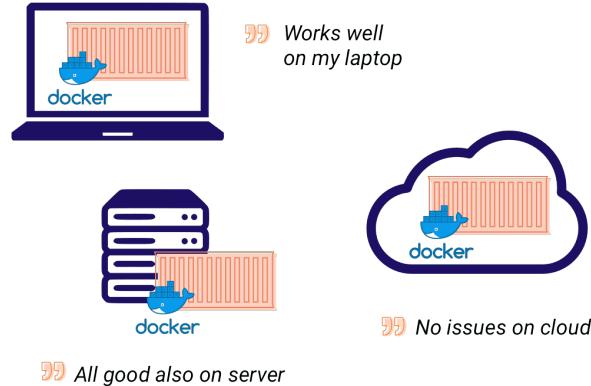


” All good also on server

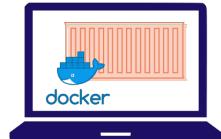


” No issues on cloud!

Background Talk: What is Docker?



Background Talk: What is Docker?



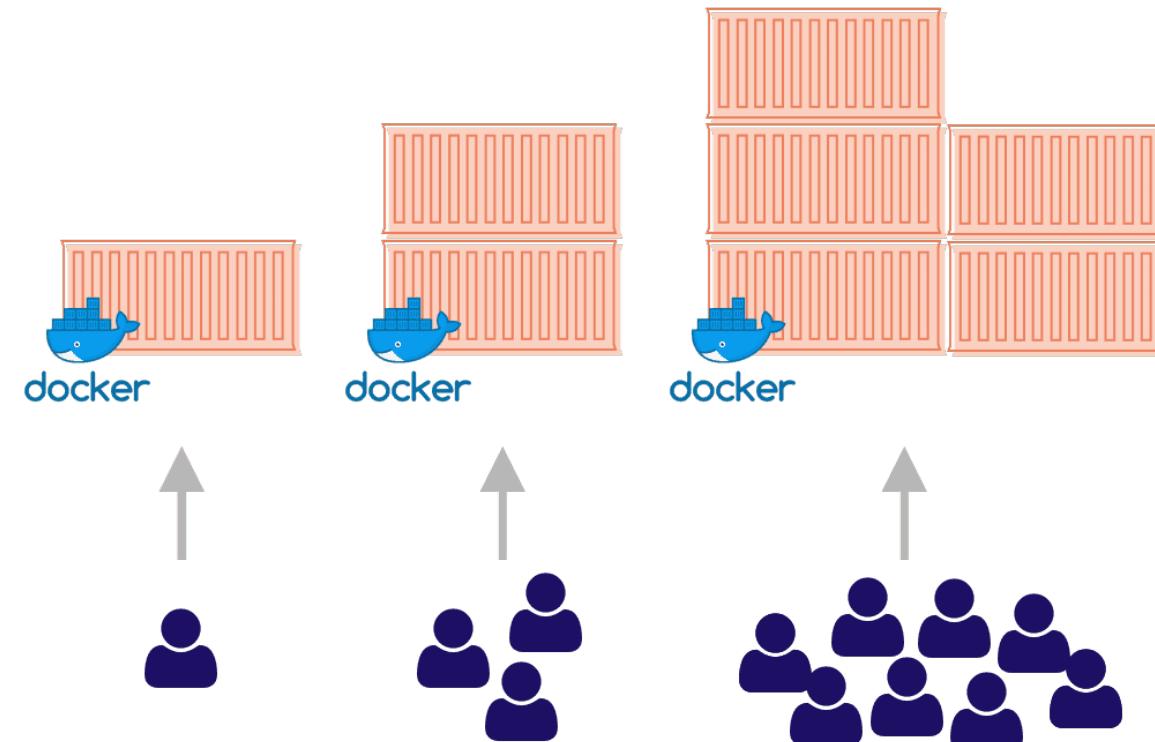
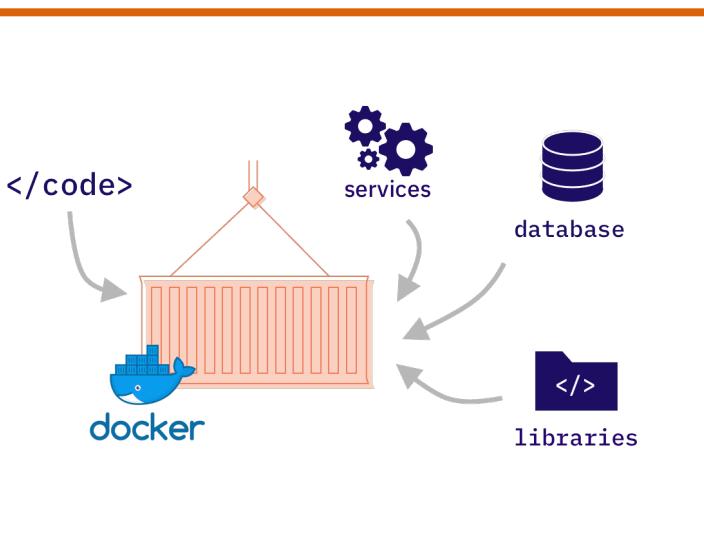
» Works well
on my laptop



» All good also on server

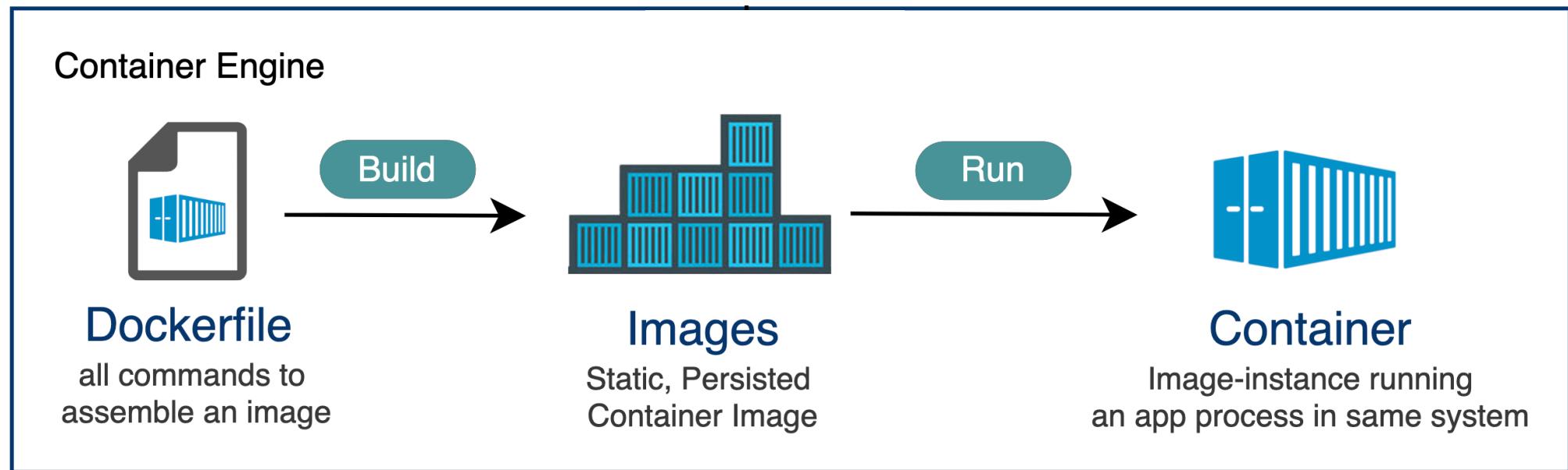


» No issues on cloud!



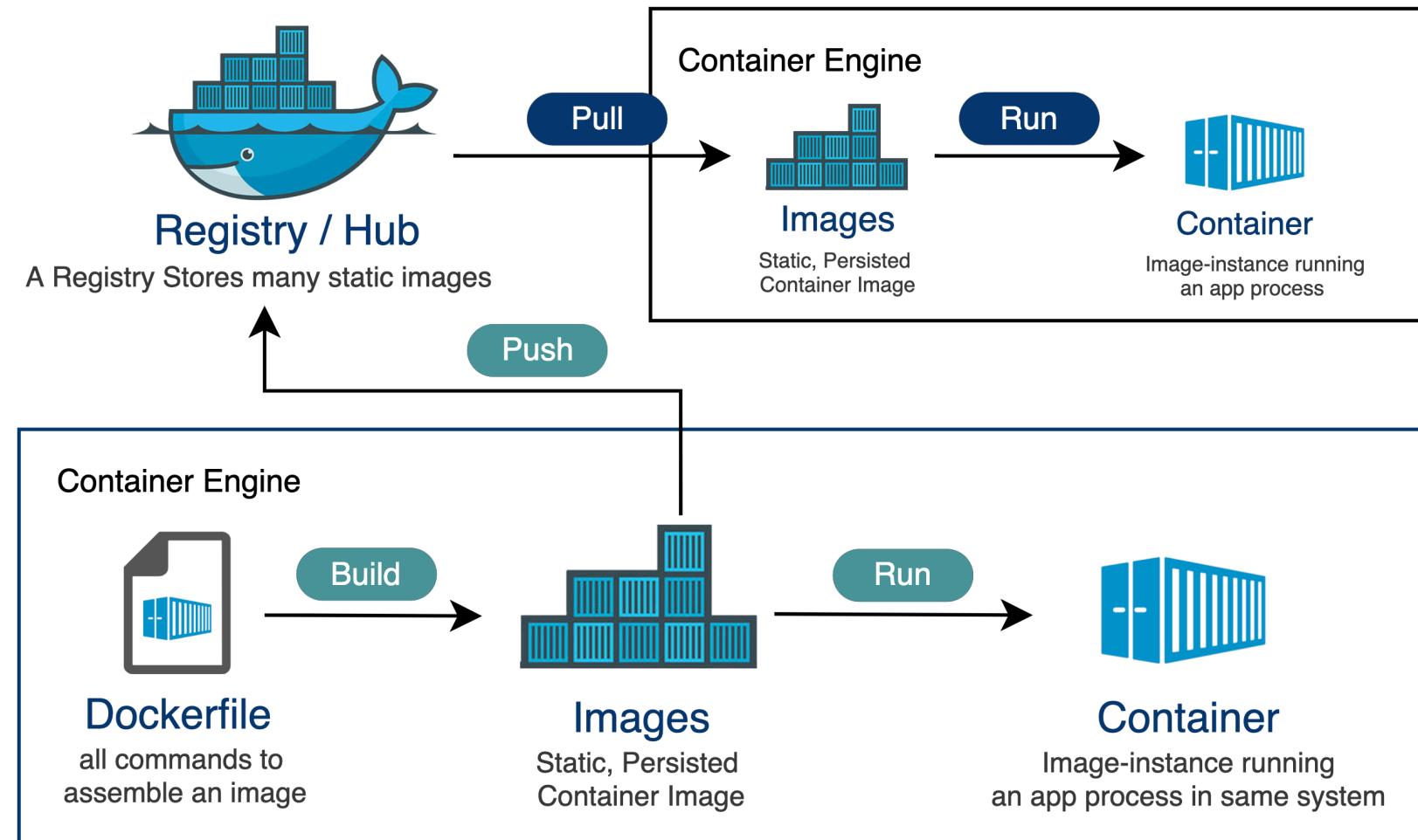
Background Talk: How does it work?

SPONSORED BY THE



Docker Hub and Registries

SPONSORED BY THE



Hands-On: Hello-World toy example

SPONSORED BY THE



Goal: Pulling an image, creating a container and removing the container.

**Find a detailed description of the steps in the GitHub repository under
["Hands-On 1: Hello-World toy example"](#).**

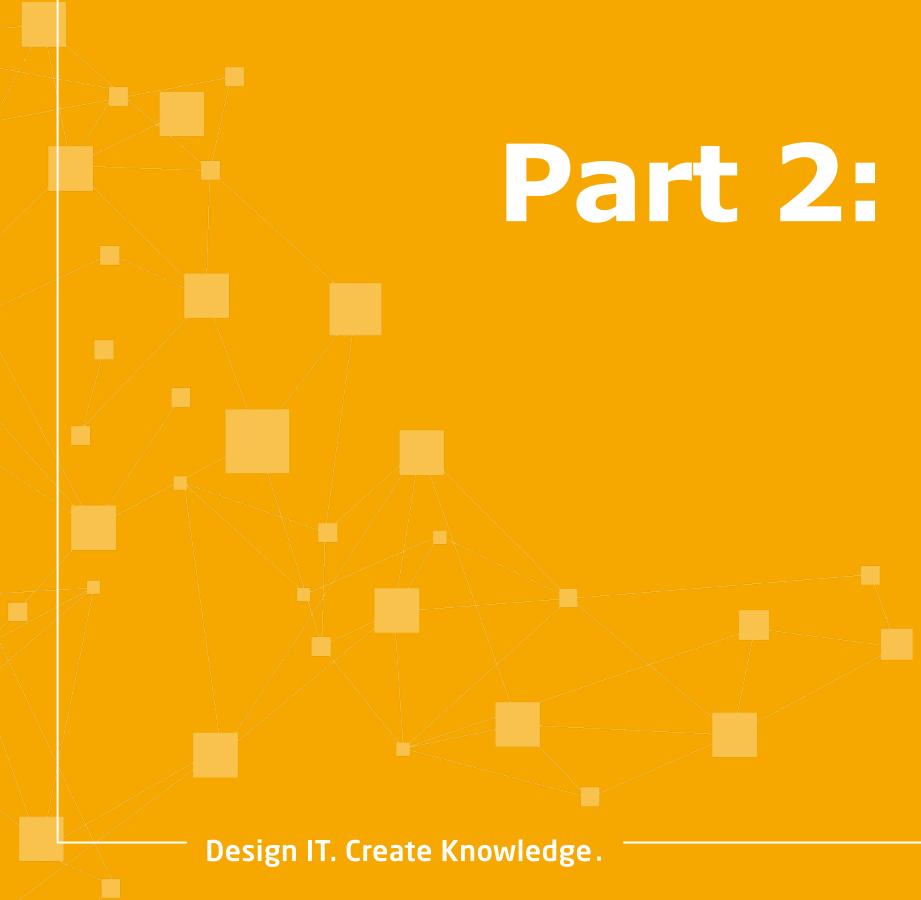
Discussion

How was your experience?

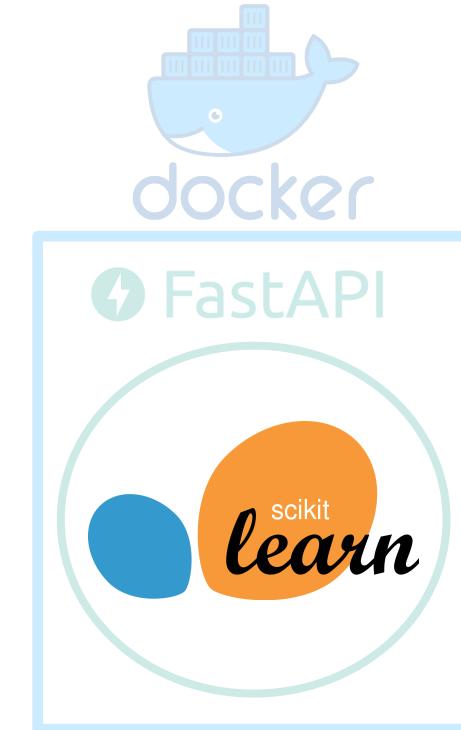
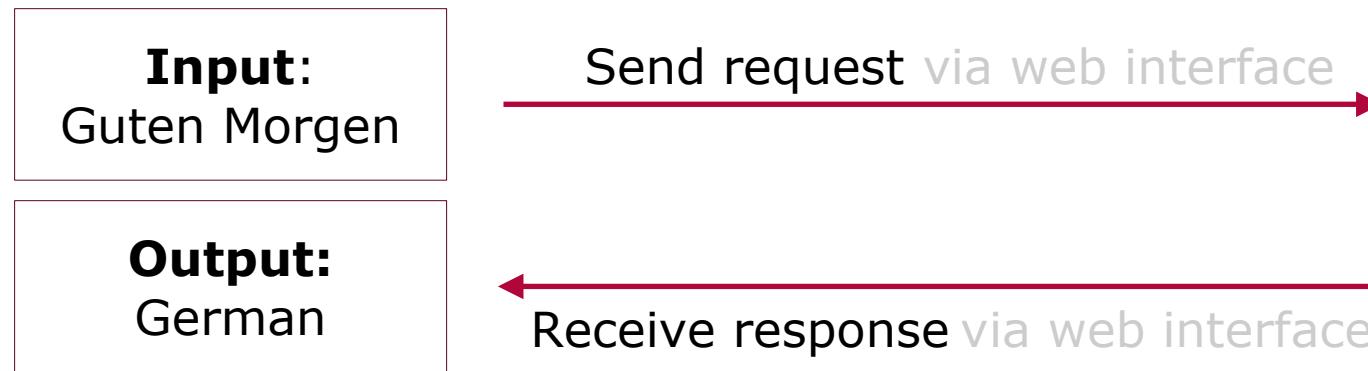
A photograph showing three hands holding cups. One hand holds a white cup with a latte featuring a circular design. Another hand holds a white cup with a latte featuring a leaf-like design. A third hand holds a dark, reflective cup. The background is blurred, suggesting a coffee shop setting.

Break

Part 2: Creating the ML Model



What we will build



Live Demo

SPONSORED BY THE



Using the Model

Language Detection Pipeline

SPONSORED BY THE



Input

A text for which we want to determine the language.

Ciao!



Machine learning model



Output

The best matching language.

Italian

Discussion: What is Jupyter?



SPONSORED BY THE



Live Demo

SPONSORED BY THE



Training & Testing the Model

Hands-On: Running Jupyter with Docker

SPONSORED BY THE



Goal: Running Jupyter Notebook in a Docker container

Find a detailed description of the steps in the GitHub repository under
"Hands-On 2: Running Jupyter with Docker"

Docker Mount Types

Volume Mount

Docker managed file system

Persistent Data Storage

Container Data Exchange

Performance Benefits

Bind Mount

Host file system

Real-time code changes

Access to System Files

Configuration Management

tmpfs Mount

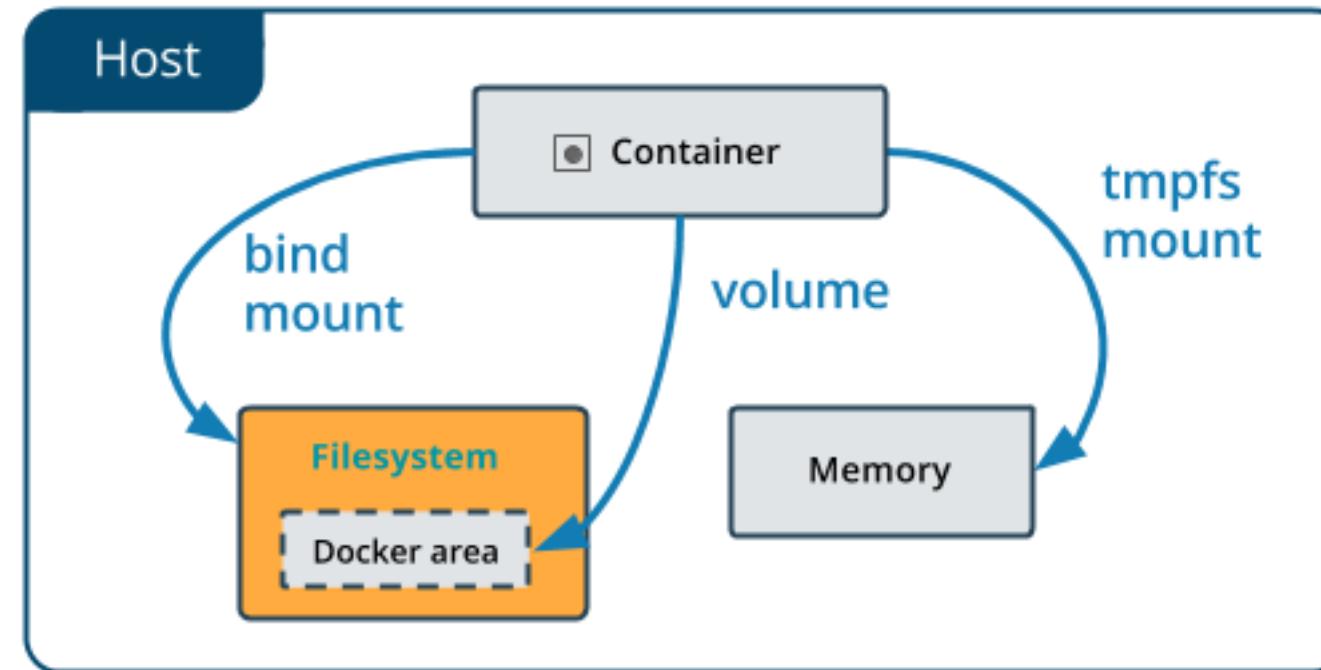
Host memory

Non-persistent Data

Fast Data Processing

In-Memory Caching

Docker Host Interaction



Hands-On: Bind Mounts

- **Problem:** The container has no access to our files
- **Next steps:**
 - Make sure we can access the `model` directory in our Jupyter notebook container
 - Start a Jupyter notebook with access to the data pre-processing and training code

Hands-On: Bind Mounts

SPONSORED BY THE



Goal: Use bind mounts to access our `model` directory in our Jupyter notebook

**Detailed steps listed on the GitHub Repository under
["Hands-On 3: Bind Mount"](#)**

Hint: Mount the `model` source folder from the clone of the workshop code to the `/home/jovyan/work` target folder in the Docker container.

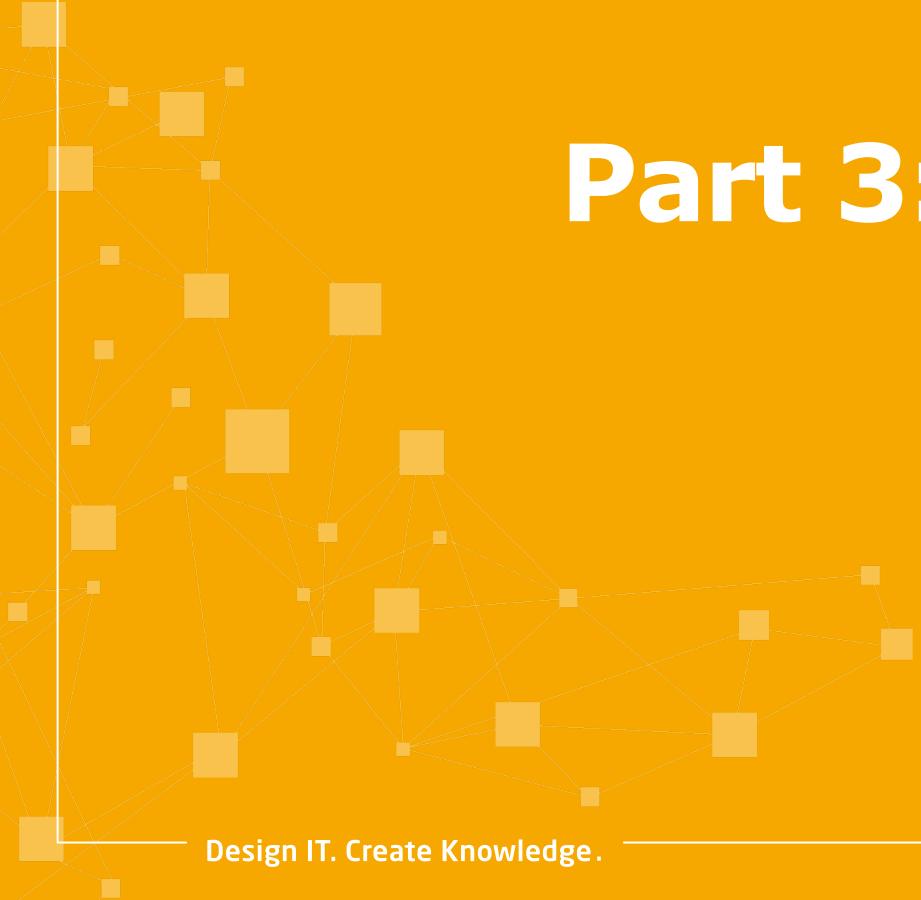
Discussion

How was your experience?

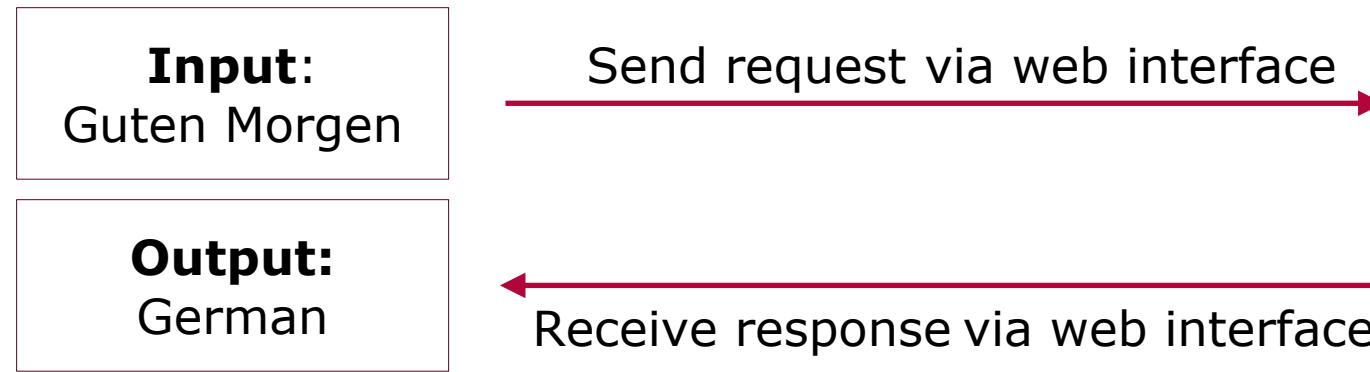
A photograph showing three hands holding cups. One hand holds a white cup with a latte featuring a circular design. Another hand holds a white cup with a latte featuring a leaf-like design. A third hand holds a dark-colored cup, possibly containing a soda or beer. The background is blurred, suggesting an indoor setting like a cafe.

Break

Part 3: Creating the ML API



What we will build



Introduction to APIs



SPONSORED BY THE



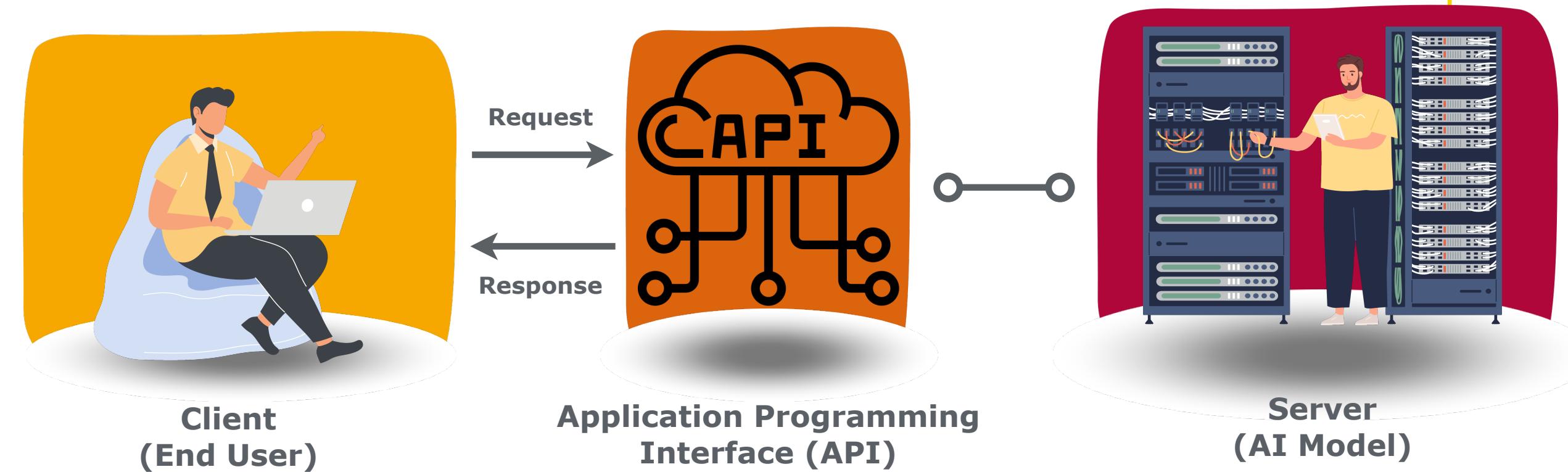
Introduction to APIs

SPONSORED BY THE



Introduction to APIs

SPONSORED BY THE



Live Demo

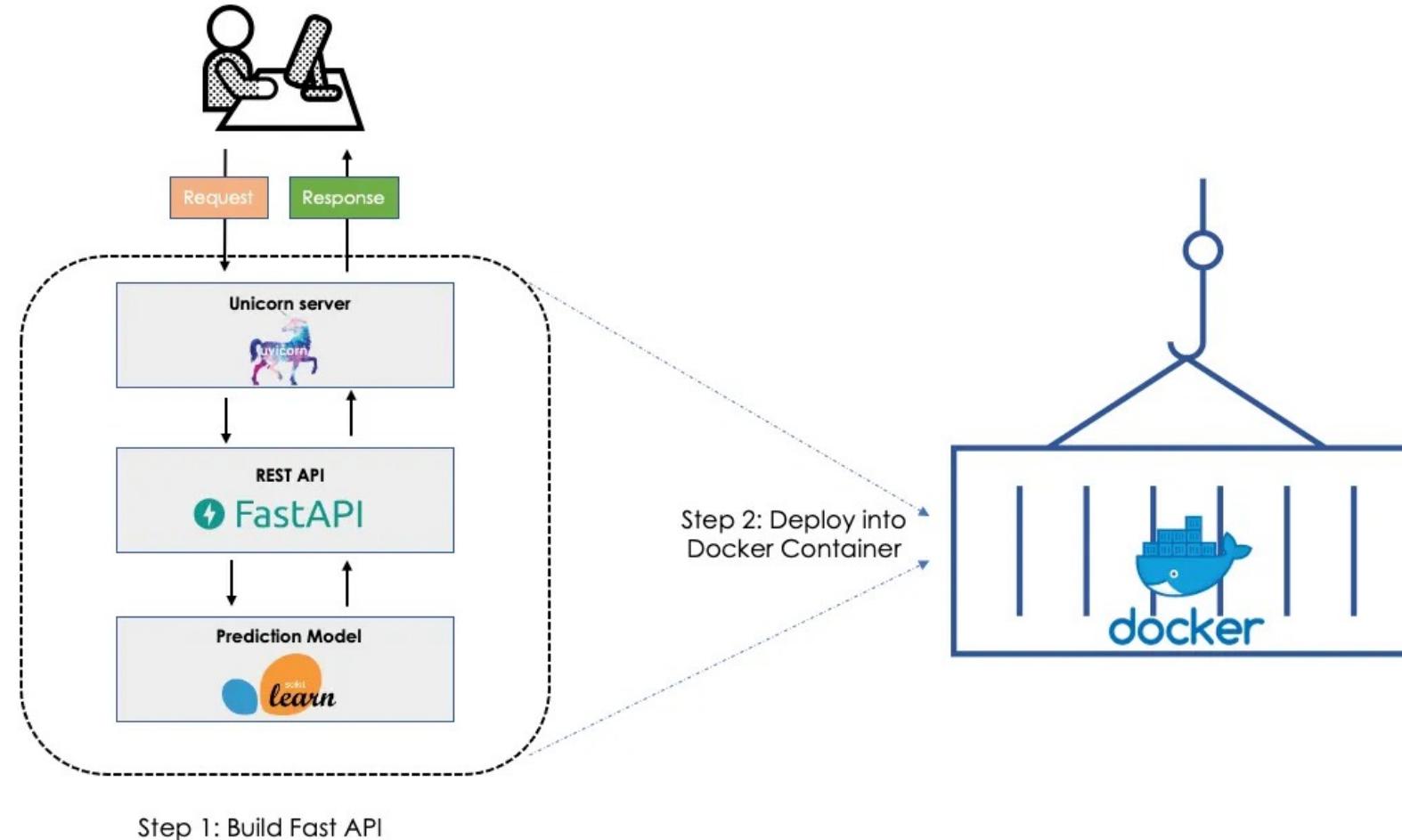
SPONSORED BY THE



API calls with FastAPI

Introduction to Web APIs for Machine Learning

SPONSORED BY THE



Discussion: Main API Methods for Machine Learning

POST

Submits Data

Use Cases

Model Prediction:

Submit input features to a model to get predictions.

Other Use Case:

?

GET

Retrieves Data

Use Cases

Model Overview:

Retrieve information about a trained model.

Other Use Case:

?

PUT

Update Data

Use Cases

Model Retraining:

Update a model with new training data.

Other Use Case:

?

DELETE

Deletes Data

Use Cases

Model Deletion:

Remove a trained model from the server.

Other Use Case:

?

Simplest FastAPI file

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Hands-On: Our FastAPI file

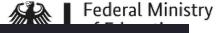
```
model_api = FastAPI()

class Result(BaseModel):
    language : str

def get_detection_model():
    full_detection_model = joblib.load("model/multinomial_language_detector.joblib")
    return full_detection_model

# TODO: Create a post request with the path "/predict" and the Result response_model.
@model_api.
async def predict(input_text, full_detection_model = Depends(get_detection_model)):
    detection_model, cv = full_detection_model
    vectorized = cv.transform([input_text])
    language_prediction = detection_model.predict(vectorized)[0]
    # TODO: Return a Result Object with the language being the predicted language.
    return
model_api.mount("/", StaticFiles(directory="app/static", html=True), name="static")
```

SPONSORED BY THE

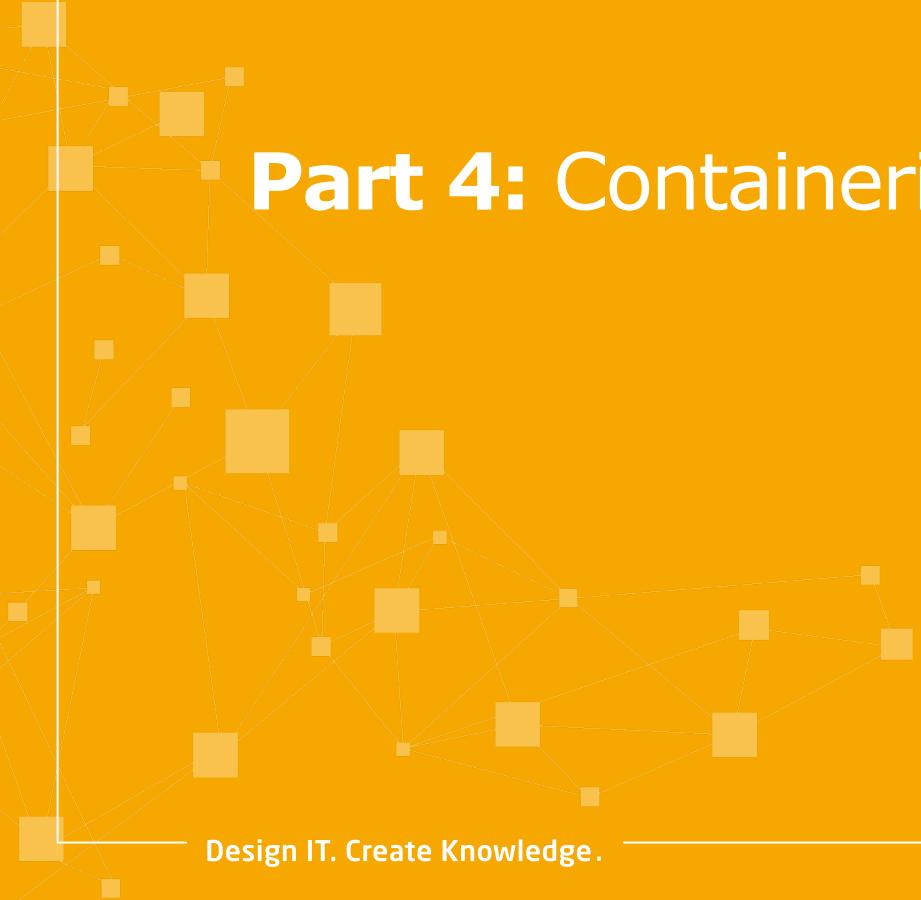


Discussion

How was your experience?

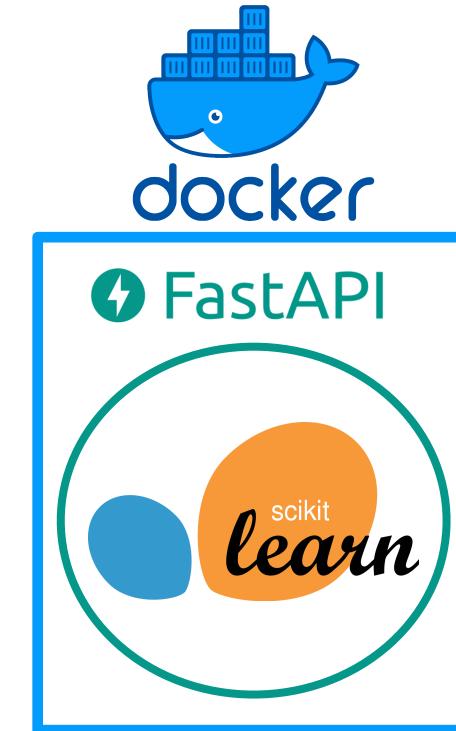
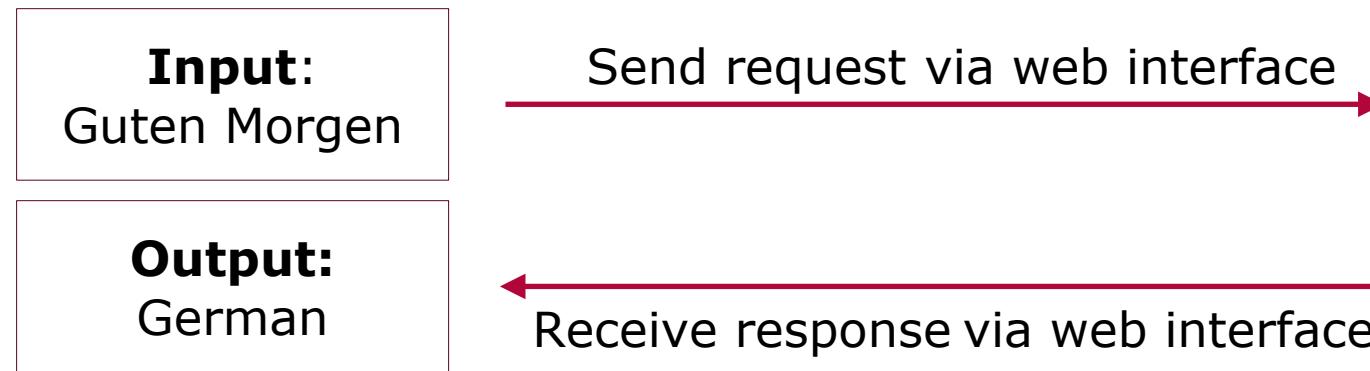
A photograph showing three hands holding cups. One hand holds a white cup with a latte featuring a circular design. Another hand holds a white cup with a latte featuring a leaf-like design. A third hand holds a dark, reflective cup. The background is blurred, suggesting a coffee shop setting.

Break



Part 4: Containerizing and Deploying the ML Model API

What we will build



The Dockerfile: Important Commands

FROM

Specifies a base image
`FROM python:3.11`

WORKDIR

Sets the working directory
`WORKDIR /app`

COPY

Copies files to image
`COPY ./ /app`

RUN

Executes terminal commands
`RUN pip install -r requirements.txt`

CMD

Initial container command. One per Dockerfile.
`CMD ["uvicorn", "app.api:app", "--host", "0.0.0.0"]`

Hands-On: Building a Docker Image

SPONSORED BY THE



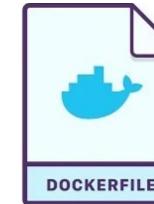
Build: docker build -t mlops/language-detection:v1 .

Start: docker run -p 8080:8000 mlops/language-detection:v1



Dockerfile

FROM Python Base Image



WORKDIR in the container set to /app

COPY the requirements.txt file into the container

RUN pip install for packages specified in requirements.txt

COPY the rest of the application into the container

CMD for starting the unicorn server through the terminal command:
`unicorn app.location:api_name --host 0.0.0.0`

Discussion

What tedious aspects of the build process did you observe?

What we noticed

- Slow build times due to big Docker images
- Running the container in attached mode is not suitable for production
- Using individual `build` and `run` commands is not useful in a production setting

Running the Container with Docker Compose

Build: docker build -t mlops/language-detection:v1 .

Start: docker run -p 8080:8000 mlops/language-detection:v1

Simpler:

docker compose up --build

```
version: "3"
services:
  api:
    image: mlops/language-detection:v1
    build:
      context: .
    ports:
      - "8000:8000"
    # volumes:
    #   - .:/app
```

Live-Demo: Deploying the Container with Docker Compose

- 1. Problem: Container shuts down when terminal is closed**
2. Run the container in detached mode: `docker compose up -d`

- 3. Problem: Container does not restart when the machine restarts**
4. Add `restart: always` to the `compose.yaml` file

- 5. Problem: The application has errors and I need to see the logs**
6. Check that the container is running by running `docker ps`
7. Check the logs of the specific container using: `docker logs YOUR_CONTAINER_ID`

8. Close the container using: `docker compose down`

Discussion

SPONSORED BY THE



How can we make the image smaller?

Competition: Who can create the smallest image size?

SPONSORED BY THE



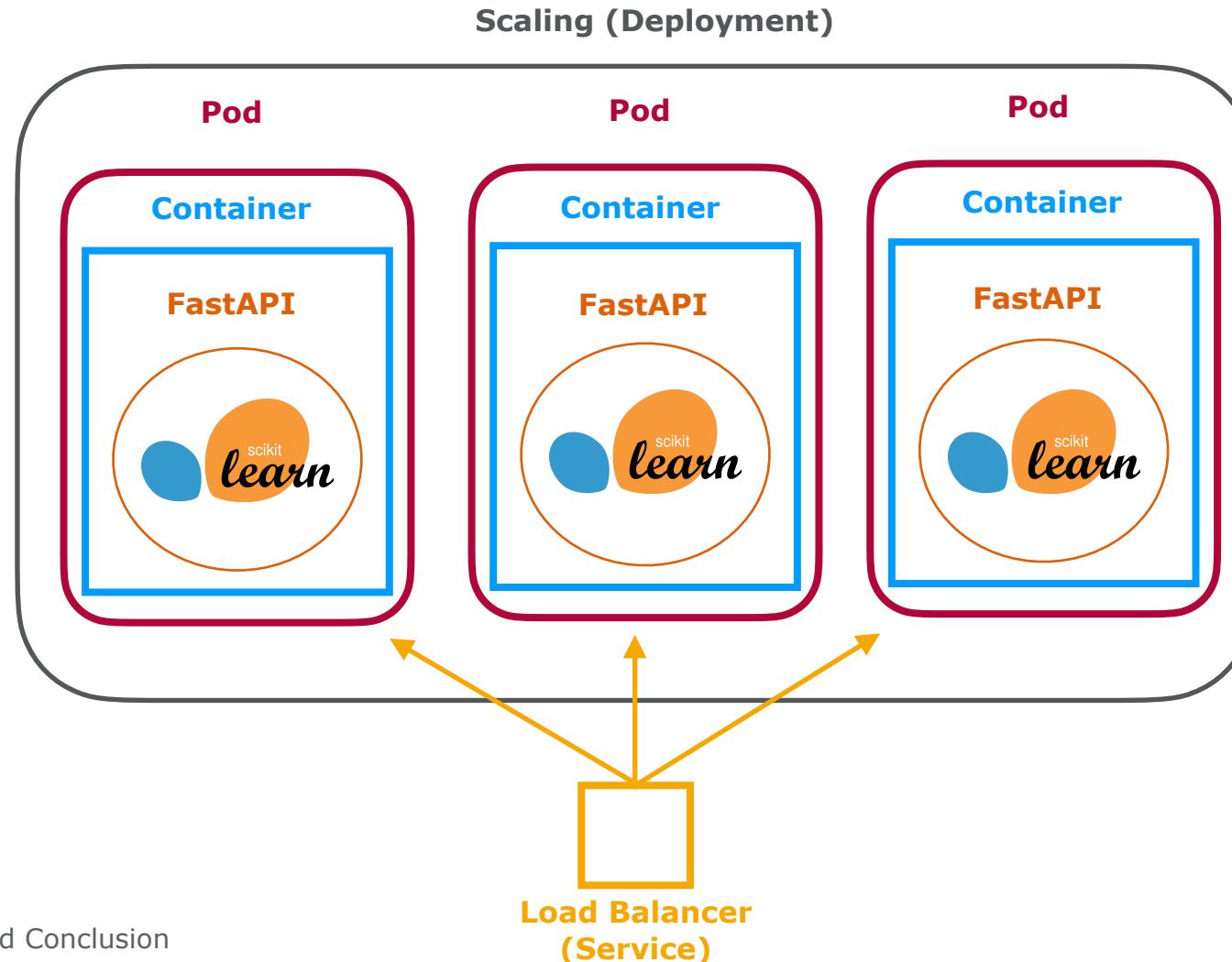
Pair up in teams of two and work together to create the smallest possible image size.

- Use a smaller base image
- Install only what is necessary for production
- Create a `.dockerignore` file to ignore unnecessary files
- Delete any cache files that may have been created

15:00

Outlook: Scaling and Load Balancing with Kubernetes

SPONSORED BY THE



Conclusion

- ✓ Understand how we can use a **basic language detection model**
- ✓ Understand **FastAPI** and **APIs**
- ✓ Understand essential **Docker features**
- ✓ Be able to **deploy a basic ML application**

Feedback and Q&A

forms.gle/JU6gsnDQhsY585mAA



SPONSORED BY THE

