Artificial Intelligence I: Introduction to Data Science and Machine Learning

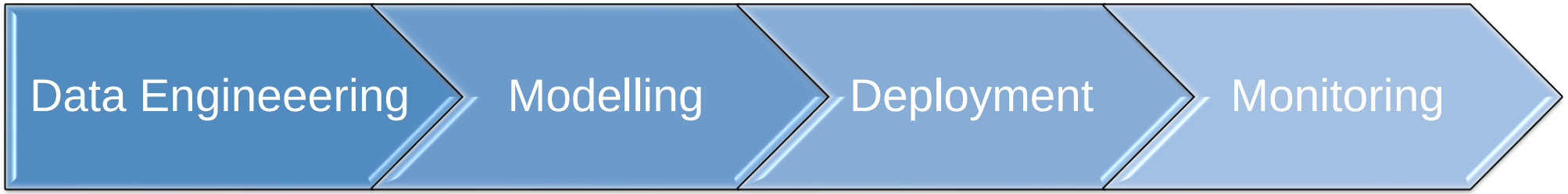
Assoc. Prof. Dr. Taner Arsan
H. Fuat Alsan, PhD(c)
Sena Kılınç, PhD(c)

Outline

- **What is MLOps?**
- **What is a Web Service?**
- **What is API?**
- **HTTP Rest API**
- **Microservices**
- **FastAPI**
- **Docker**
- **MLFlow**
- **Implementing Machine Learning Models as Web Services**

MLOps

- MLOps (Machine Learning Operations) is a discipline that encompasses the process of developing, training, deploying, managing and maintaining machine learning models.
  - Faster and more reliable model distribution
  - Increased model performance
  - Better model management
  - Improved collaboration



Data Engineering

- Data Preprocessing [Jupyter Notebook]

Modelling

- Model Selection and Evaluation [Scikit-Learn, Pytorch, Tensorflow]
- Model Selection and Optimization

Deployment

- Automation and CI/CD [Github, Jenkins]
- Model Packaging and Distribution [Docker, Kubernetes]

Monitoring

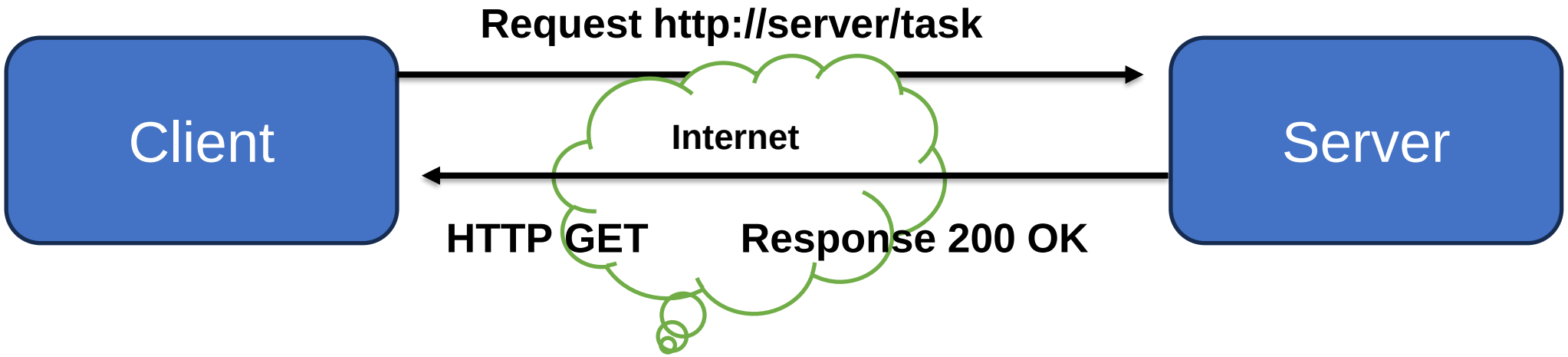
- Tracking Models and Optimization [MLFlow]
- Model Management and Version Control [Git, Docker]

MLOps

- Starbucks India uses data-driven strategies to reduce customer churn and increase sales opportunities; Thanks to its integrated data platform, it **increases revenue** in targeted campaigns and **reduces customer churn**.
- While Senko Logistics Group has adopted MLOps to improve shipment volume accuracy with AI-powered predictions; It reduces workload by optimizing operational procedures and streamlines transactions by **increasing forecast accuracy**.
- PadSquad applies MLOps to **improve ad performance and reduce costs**; It eases the workload with automatic processes and provides the opportunity for rapid marketing by improving advertising performance.
- Philips, Netherlands, healthcare: **Hours saved** by experimental monitoring and automatic logging.
- Ecolab, in the chemical industry: **Reduced model deployment times** from 12 months to 30-90 days.

Web Services

- These are the technologies that enable the communication of software applications between different platforms.
- They are services provided by the server and used by the client applications.
- They usually communicate over the HTTP protocol and use data formats such as XML or JSON.
- They are platform independent and provide integration between applications with different technologies.



XML ve JSON

```
<bookstore>
  <book category="fiction">
    <title lang="en">Harry Potter</title>
    <author>J.K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="fiction">
    <title lang="en">The Hobbit</title>
    <author>J.R.R. Tolkien</author>
    <year>1937</year>
    <price>19.99</price>
  </book>
</bookstore>
```

```
{
  "bookstore": {
    "book": [
      {
        "category": "fiction",
        "title": "Harry Potter",
        "author": "J.K. Rowling",
        "year": 2005,
        "price": 29.99
      },
      {
        "category": "fiction",
        "title": "The Hobbit",
        "author": "J.R.R. Tolkien",
        "year": 1937,
        "price": 19.99
      }]
  }
}
```

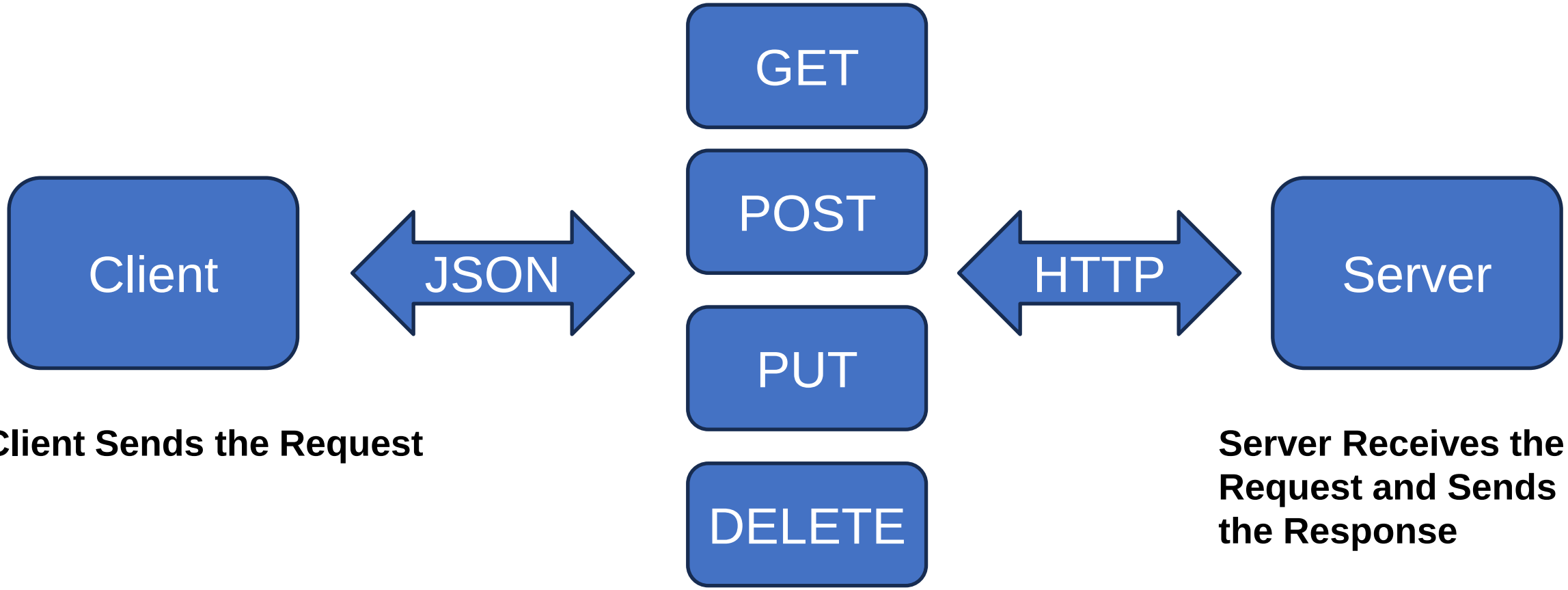
API

- APIs or Application Programming Interfaces are sets of rules and protocols that enable software applications to communicate with each other.
- APIs define the methods and data formats that applications can use to exchange information and use their functionality.
- An API is a set of protocols that allow an application to interact with another application.

What is the Difference?

- Web Services are a type of API that must be accessed over a network connection.
- APIs are application interfaces that allow different applications to communicate with each other in a standard way.
- All Web services are APIs, but not all APIs are web services.
- Web services are often associated with SOAP, while APIs can use any communication style.
- While APIs can return data in formats such as JSON or XML, web services primarily use JSON.
- Web services are heavier; APIs, on the other hand, can have a lightweight architecture suitable for devices with limited bandwidth, such as smartphones.

HTTP Rest API

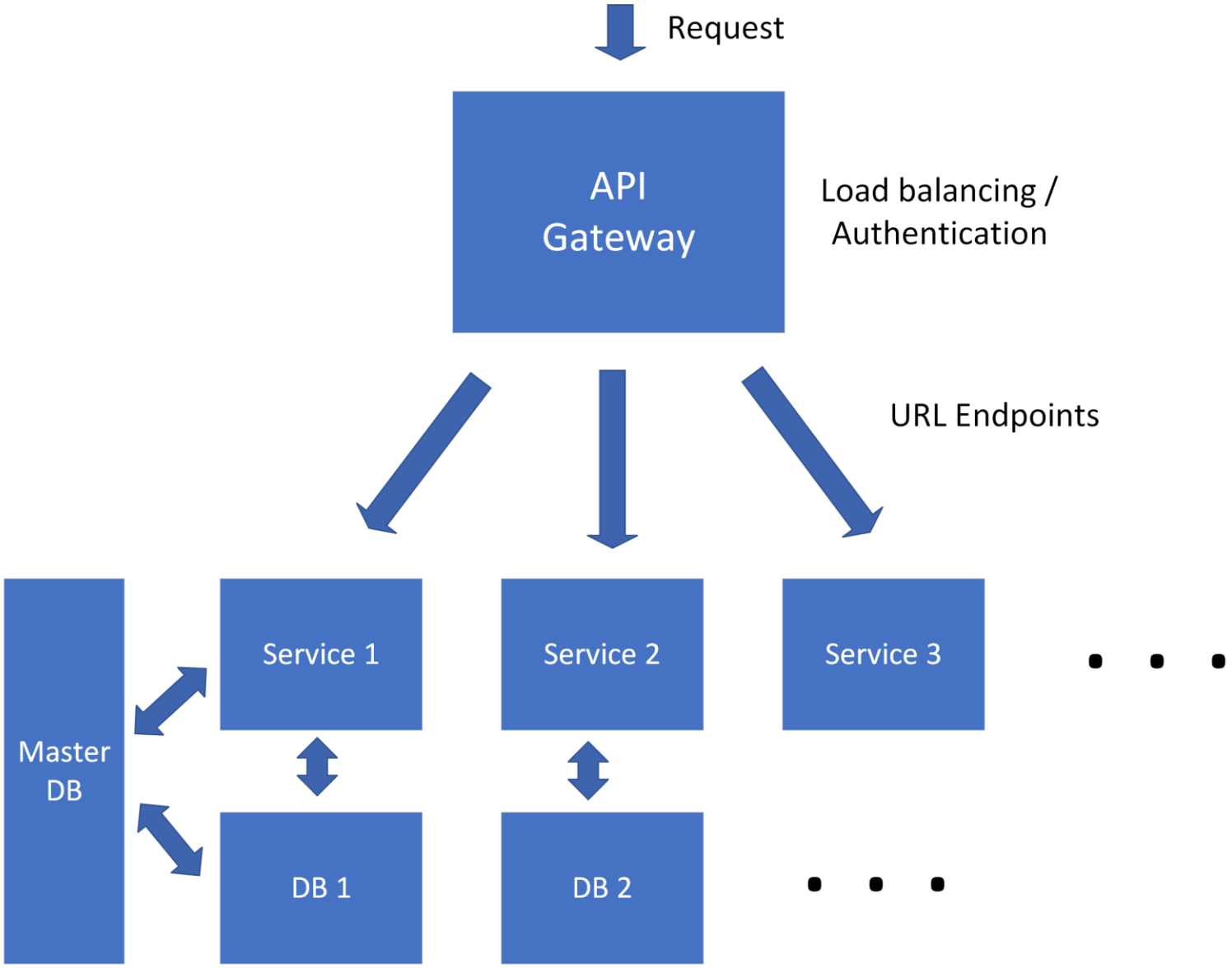


Client Sends the Request

Server Receives the Request and Sends the Response

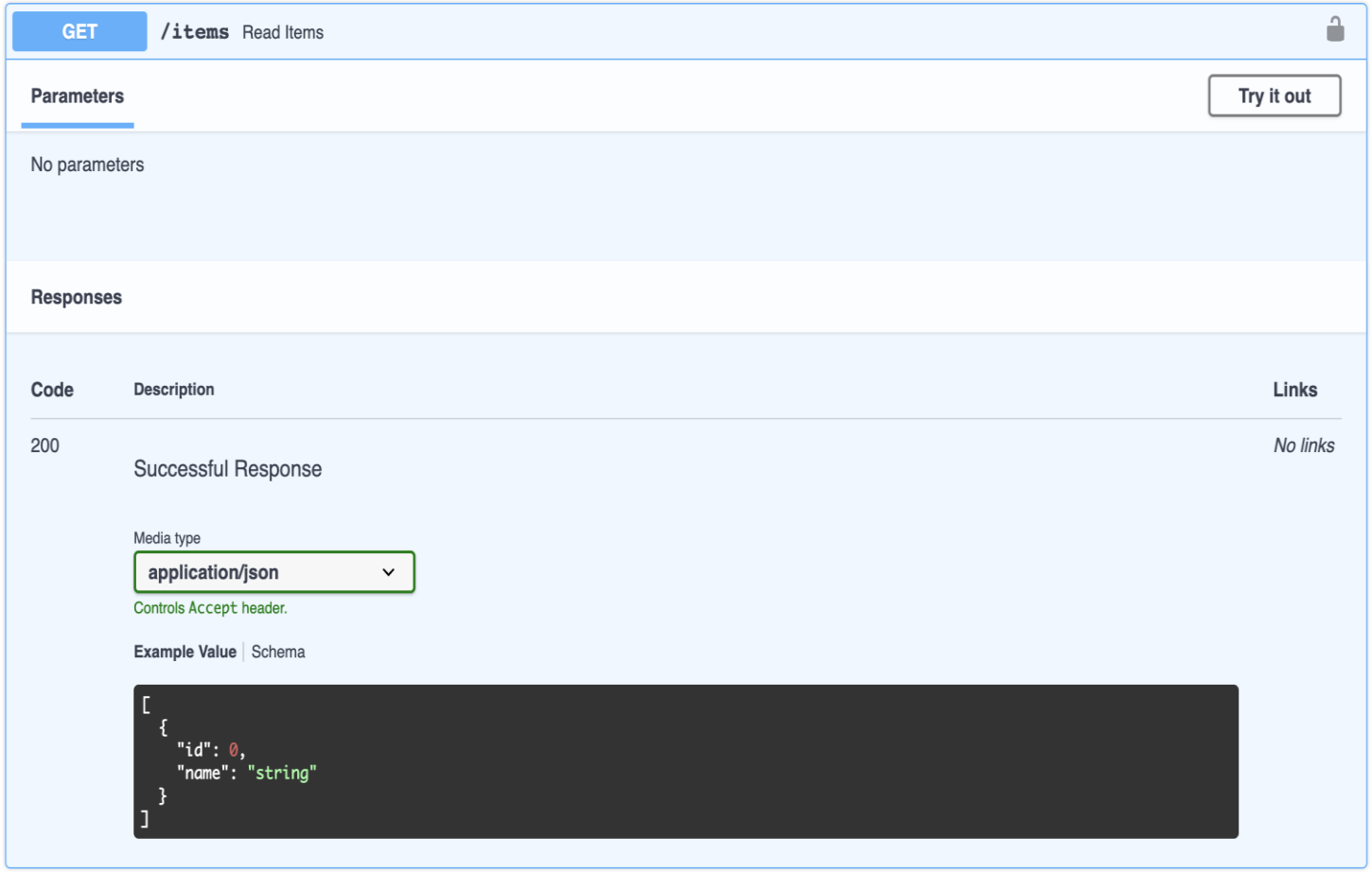
HTTP Methods

Microservices



FastAPI

- FastAPI is a modern web library for building RESTful APIs in Python.
- High Performance
- Ease of Use



Docker

- Application containers (isolated runtime, similar to virtual machines)
- Launches faster than traditional virtual machines
- Contains the underlying runtime for applications (libraries, files, etc.)
  - In our case: the running environment of the random forest model
- Can run as a daemon process in the background
- Images From Docker hub
  - In our case: MLFlow and FastAPI etc.
- Custom Docker images can be built with Dockerfile
  - In our case: randomforest_image

Docker Basics

- **Containers:** Lightweight and fast Docker elements that run applications in an isolated environment.
- **Images:** Templates that define the executable state of Docker containers.
- **Dockerfile:** Files that configure Docker images step by step.
- **Docker Compose:** Tool used to define and manage multiple Docker containers.
- **Docker Daemon and Docker Client:** Background process and user interface that processes Docker commands.

Important Docker Concepts

- **Docker containers do not store persistent data. When the container stops, all data in it is deleted.**
  - Solution: Volumes are created and mounted to Docker containers for persistence.
- **Docker containers cannot communicate directly.**
  - Solution: Virtual Docker networks are created, and virtual IPs are assigned.
  - Bridge: An isolated Docker network is created so that multiple containers can communicate on the same Docker host.

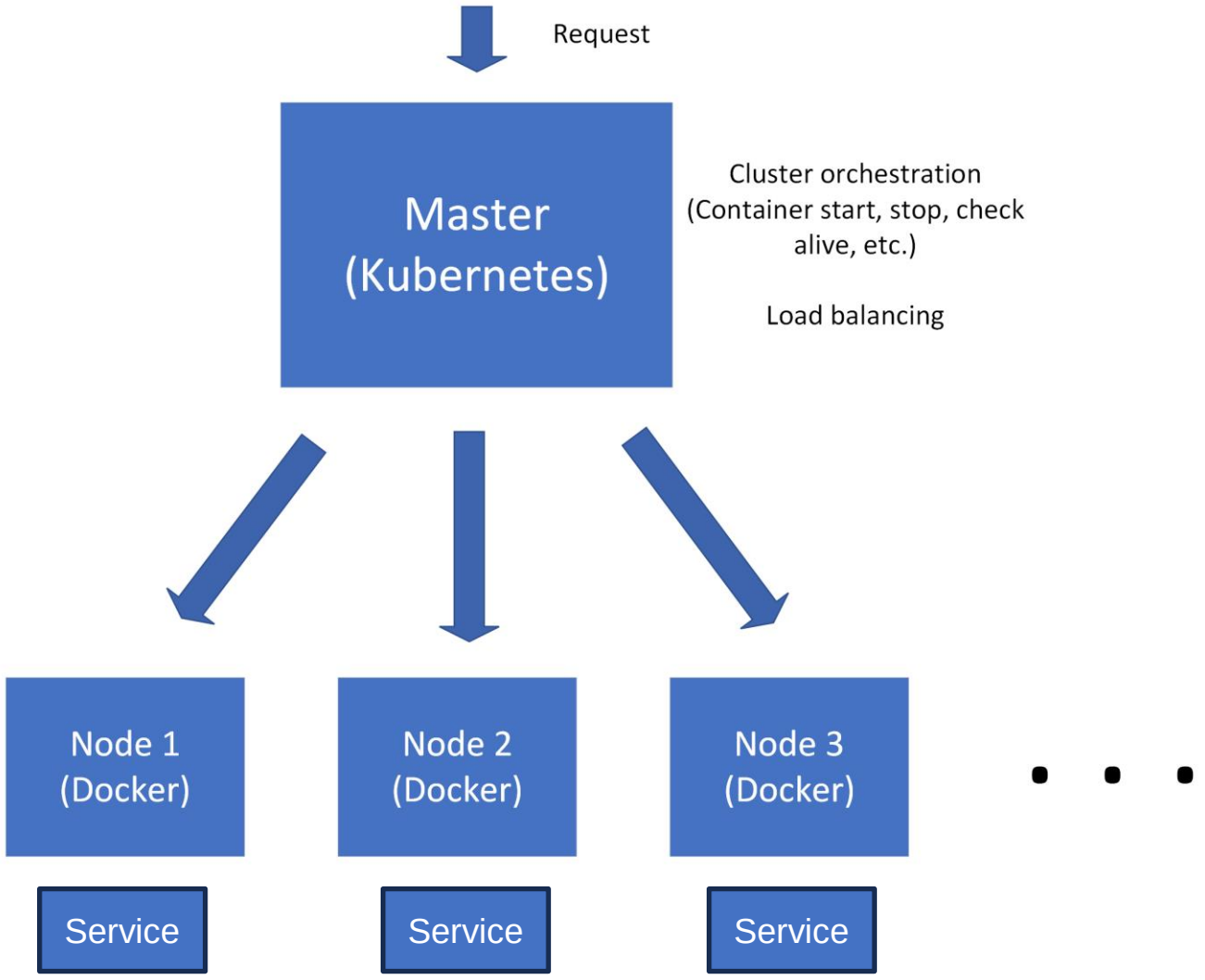
Docker Compose

- Docker Compose is a lightweight Kubernetes alternative.
- It is an orchestration tool for multiple Docker images.
- Starts and stops Docker containers easily.
- The YAML configuration file is docker-compose.yml.
- The configuration file determines the following:
  - Base Docker image
  - Port binding & forwarding
  - Permanent Volumes (Persistence)
  - Virtual networks
  - Runtime dependencies
    - (other Docker images)

Kubernetes

- **Mikroservisler hızlı bir şekilde karmaşık hale gelebilir, bu nedenle yönetim, orkestrasyon ve bakım gerektirir.**
- Yüksek Kullanılabilirlik (High Availability)
- Yük Dengeleme (Load Balancing)
- Ölçeklenebilirlik (Scalability)
  - Yatay Ölçeklenebilirlik (Horizontal Scalability)
- Kendini İyileştirme (Self-Healing)
  - Felaket Kurtarma (Disaster Recovery)

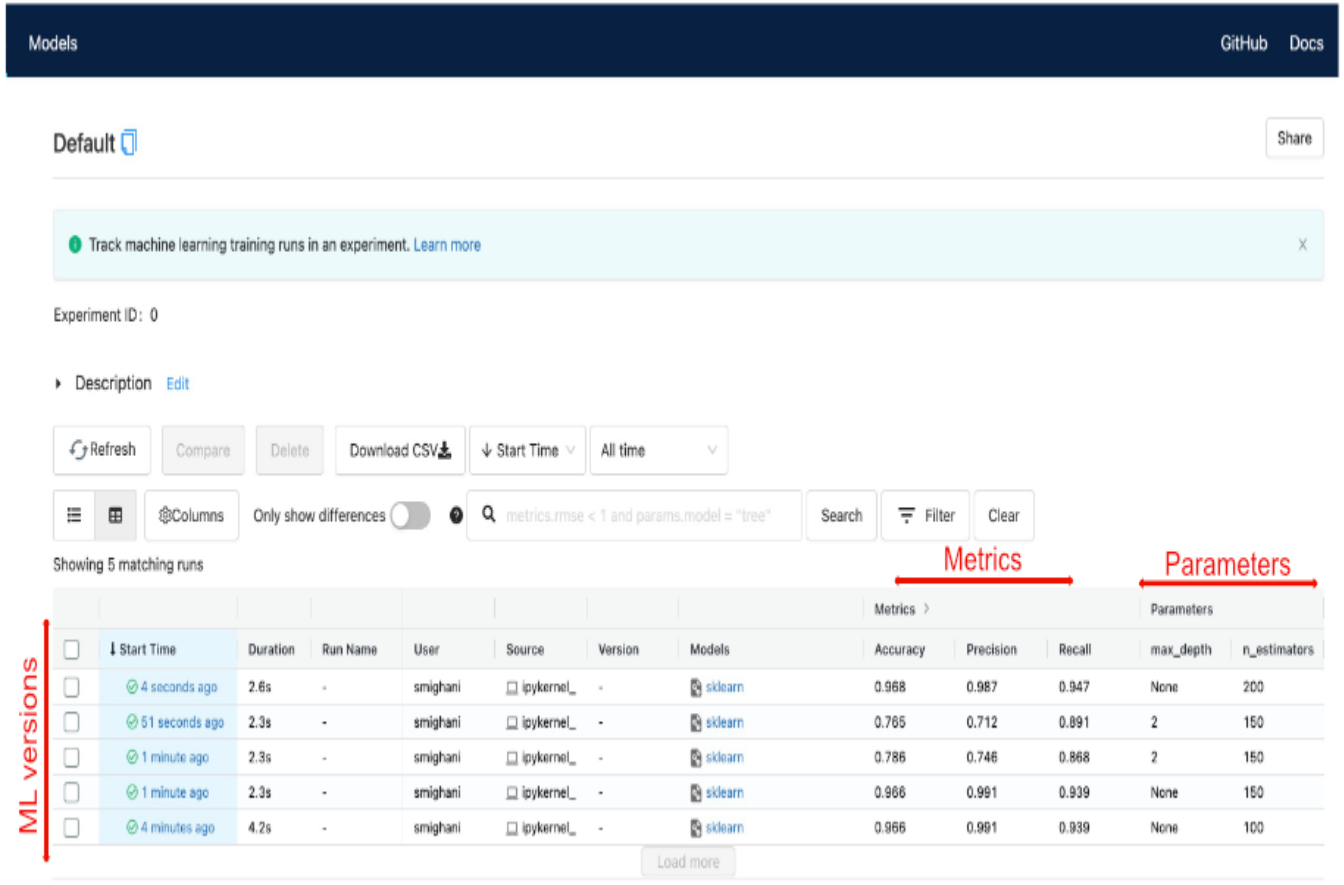
Scalability and Load Demand



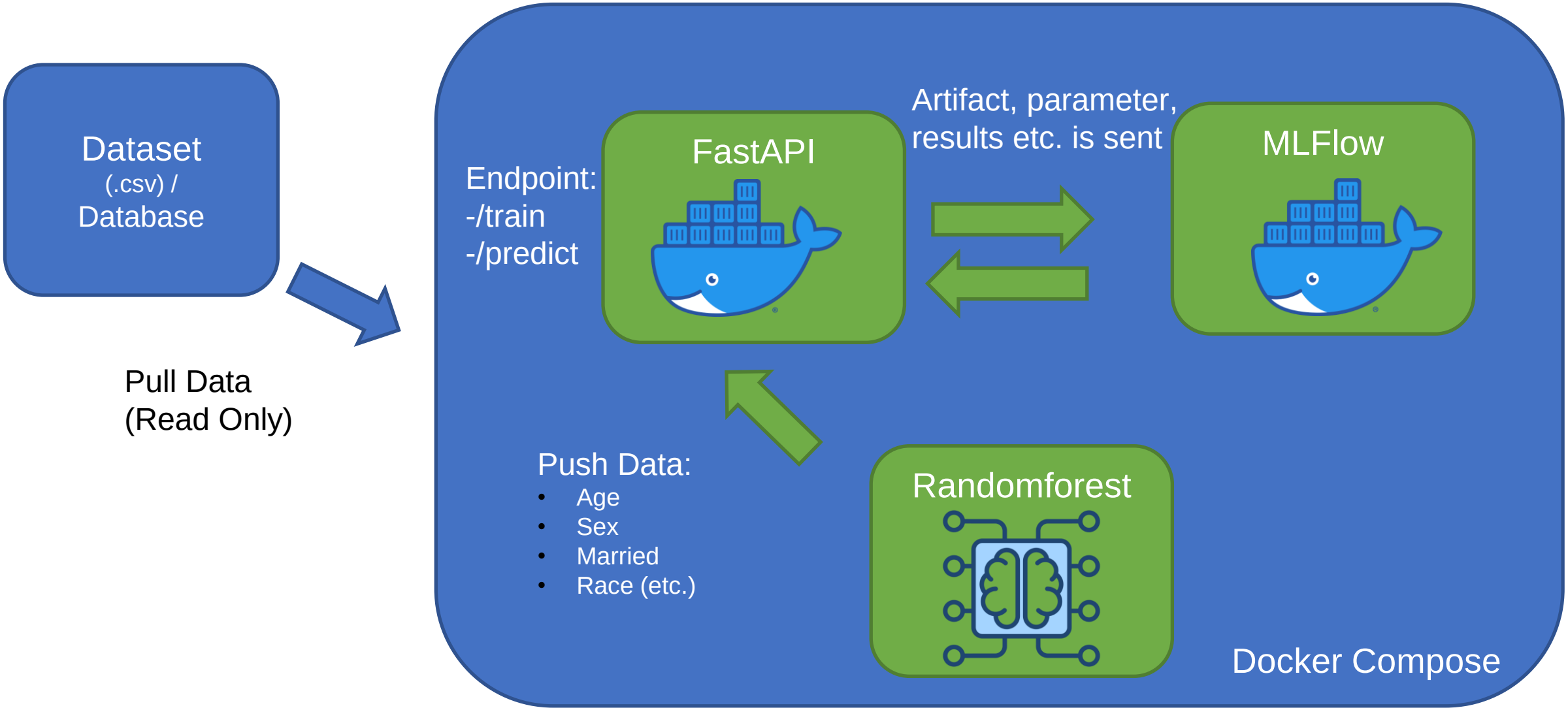
Horizontally scalable containers
(Add nodes as required)

MLFlow

- MLflow is a platform for tracking, creating and making machine development projects reproducible.
- **Monitoring and Recording:** Records every step of the project and visualizes data and results.
- **Model Management:** Provides tools for saving, versioning, and distributing trained models.



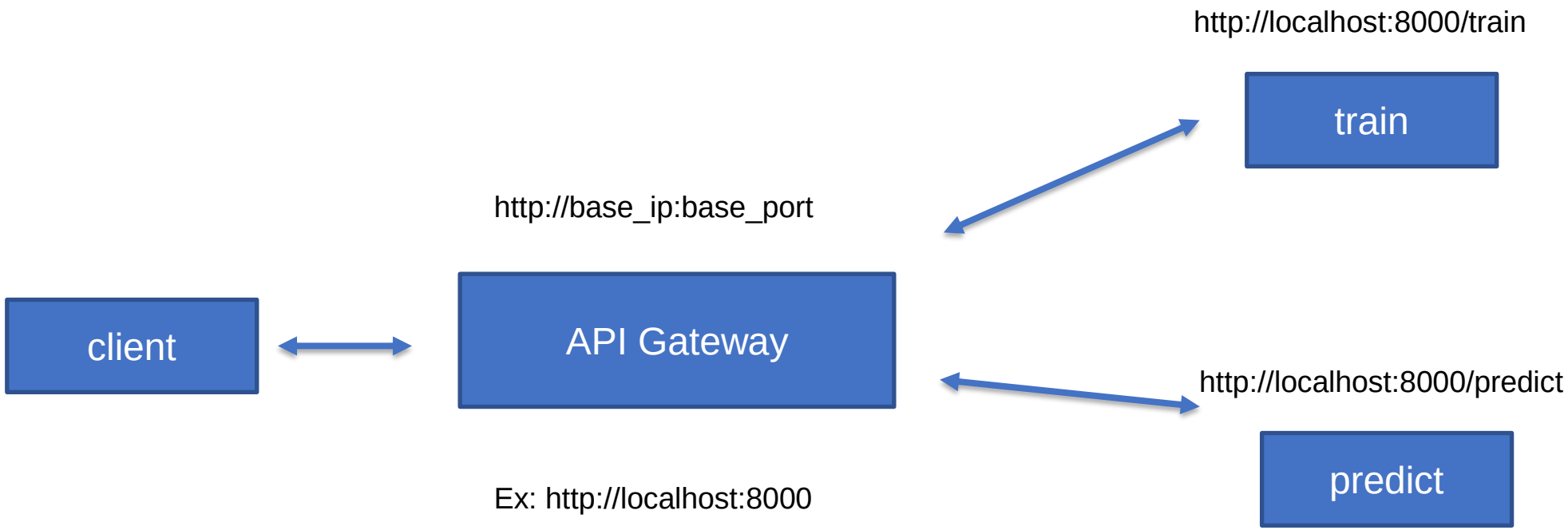
Software Architecture



Architecture Summary

- Microservice architecture
  - Each service is an independently running process
  - They can communicate with each other via API and use JSON as data format
- Docker containers will contain everything related to the random forest classifier runtime
- Process Flow:
  - **Data Extraction:** Data is read from the CSV file.
  - **Training:** The data is sent to the FastAPI endpoint and the model is trained.
  - **Prediction:** Data is sent to the FastAPI endpoint and the model makes the prediction.
  - **Tracking:** All information about the model is recorded in Mlfow.

URL Endpoints (Architecture)



URL Endpoints

- FastAPI (acts as a gateway)
- HTTP POST request to predefined URLs
- Content format JSON (request/response)
- Supported URLs:
  - /train
  - /predict

Overall Project Structure

- **randomforest.ipynb:** A Jupyter Notebook file used for exploratory data analysis (EDA).
- **randomforest.py:** A Python file containing your model. A file in which your random forest model is defined and trained.
- **app.py:** The main file that runs the FastAPI application. It also trains and predicts and identifies specific URLs using functions in randomforest.py.
- **dockerfile_fastapi:** Docker file that creates the Docker image of the FastAPI application.
- **dockerfile_mlflow:** The Docker file that creates the Docker image used by MLflow.
- **requirements.txt:** A requirements file containing the libraries used in the app.py file.
- **Docker-compose.yaml:** Docker Compose file for interoperating MLflow and FastAPI. This file is used to stand up both services and connect them together.

Data Preprocessing

- preprocess_data:
  - '?' Missing values are assigned instead of characters.
  - Independent and dependent variables are determined and the data set is divided into training and testing sets.
  - Missing values are filled in with the most common values.
  - One-Hot Encoding is applied for categorical variables.
  - Scaling is accomplished using RobustScaler.
- Feature_importance:
  - Determines the feature importance scores of the model and returns a Series sorted by importance.

DEMO

Thank you!