AI VIETNAM
All-in-One Course
(TA Session)

Apache Airflow

Extra Class: MLOps

AI VIET NAM
@aivietnam.edu.vn

Nguyen-Thuan Duong – TA

Kha-Vi Nguyen - sTA

Year 2024

code&data demo

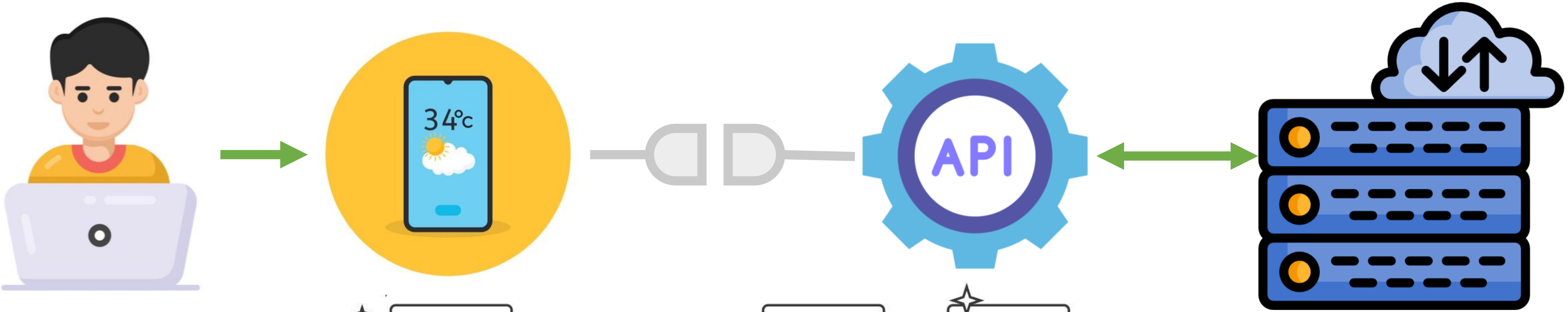
Outline

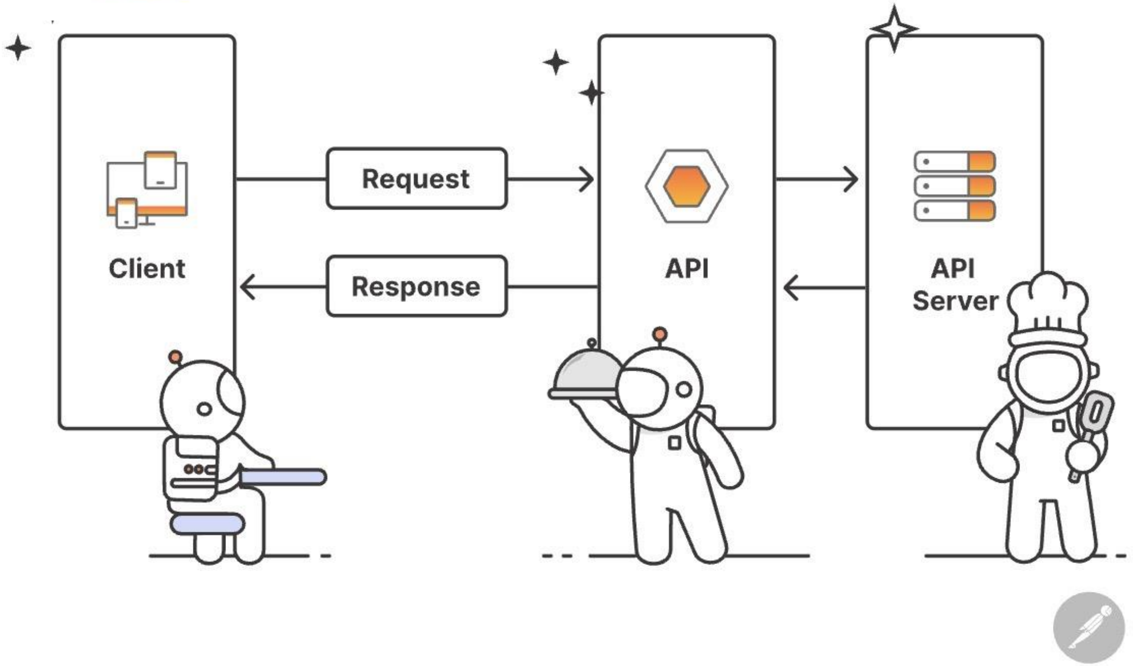
- Recap
- Introduction
- Airflow
- Pipeline
- Practice
- Question

Recap

Recap

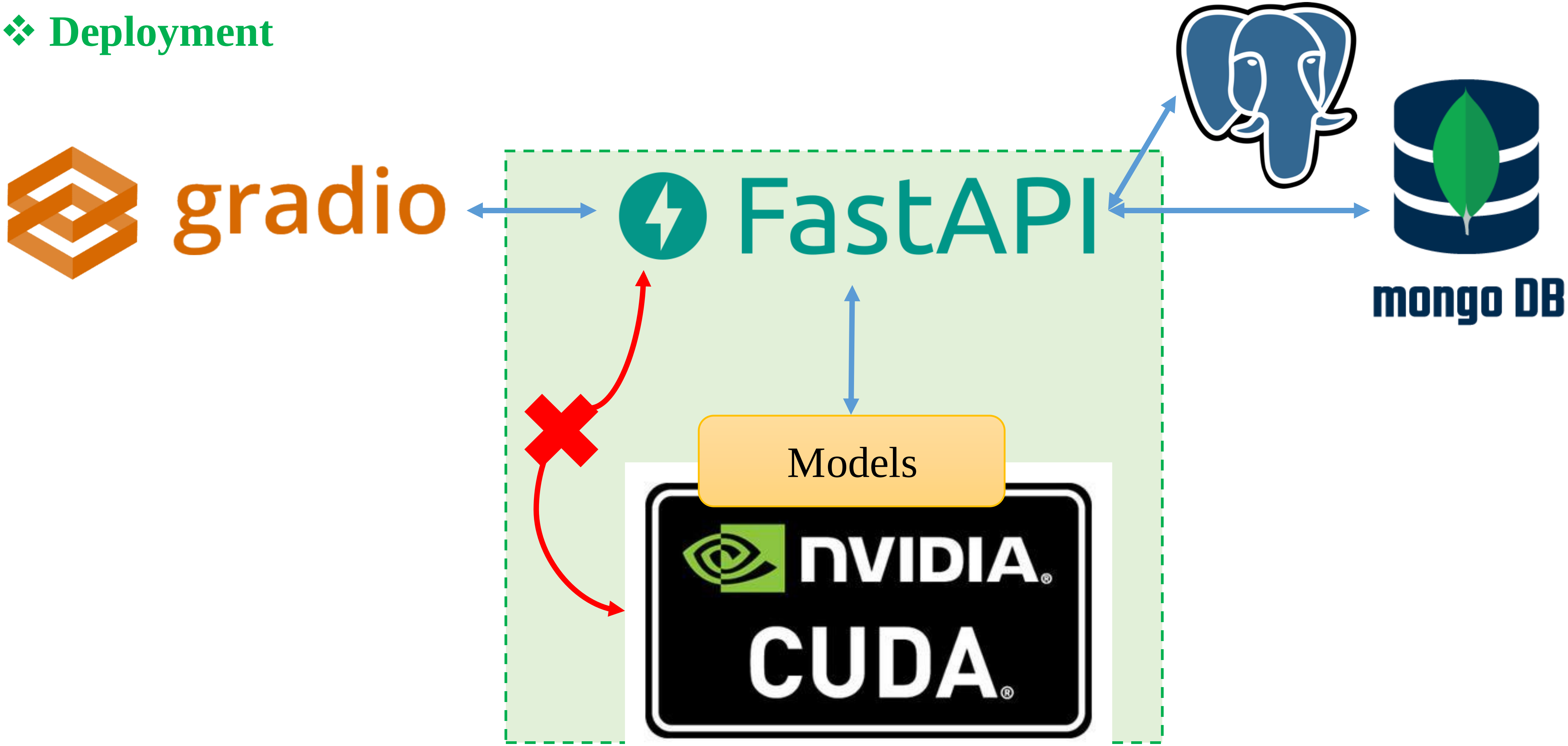
❖ Deployment





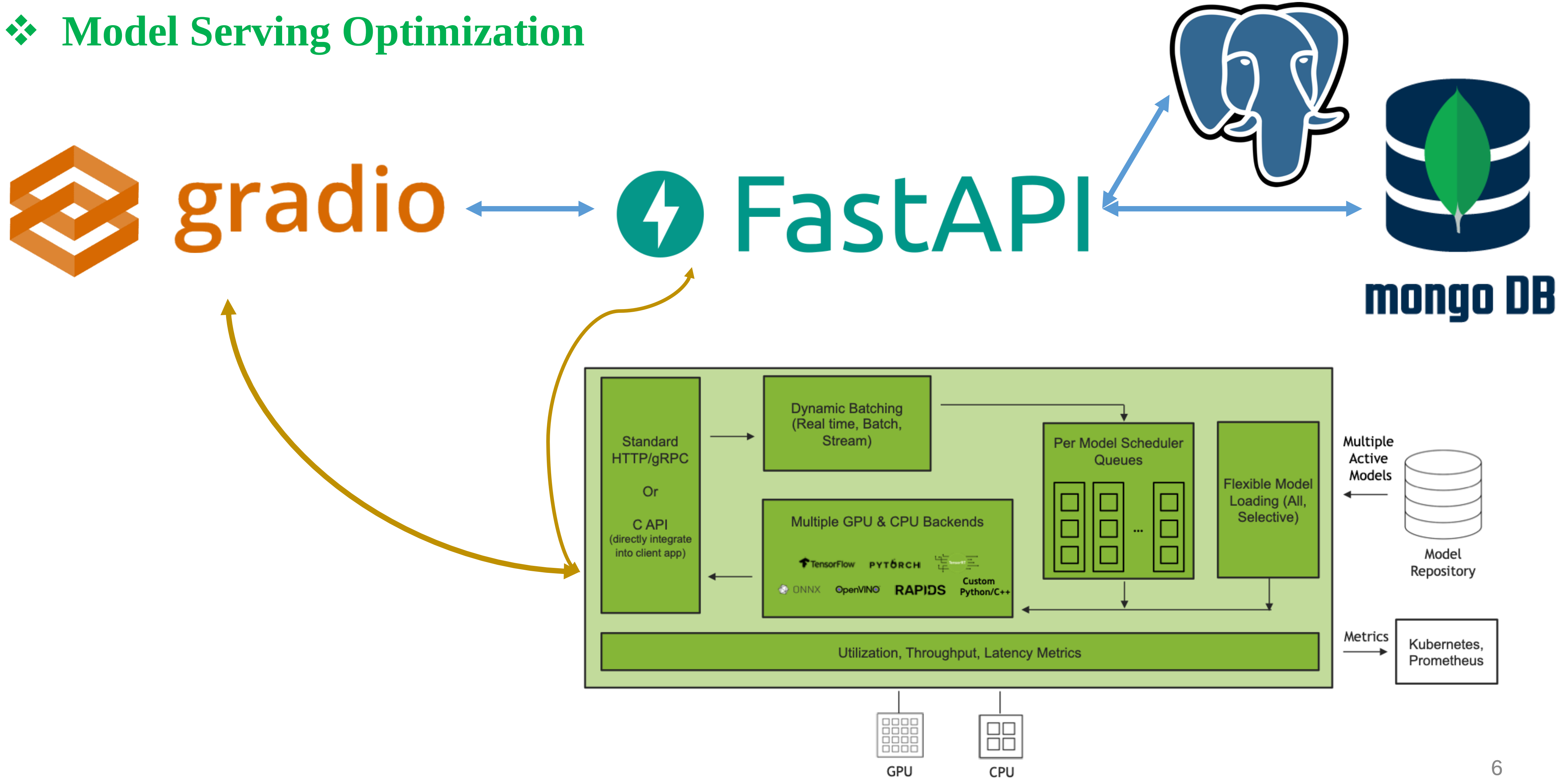
Recap

❖ Deployment



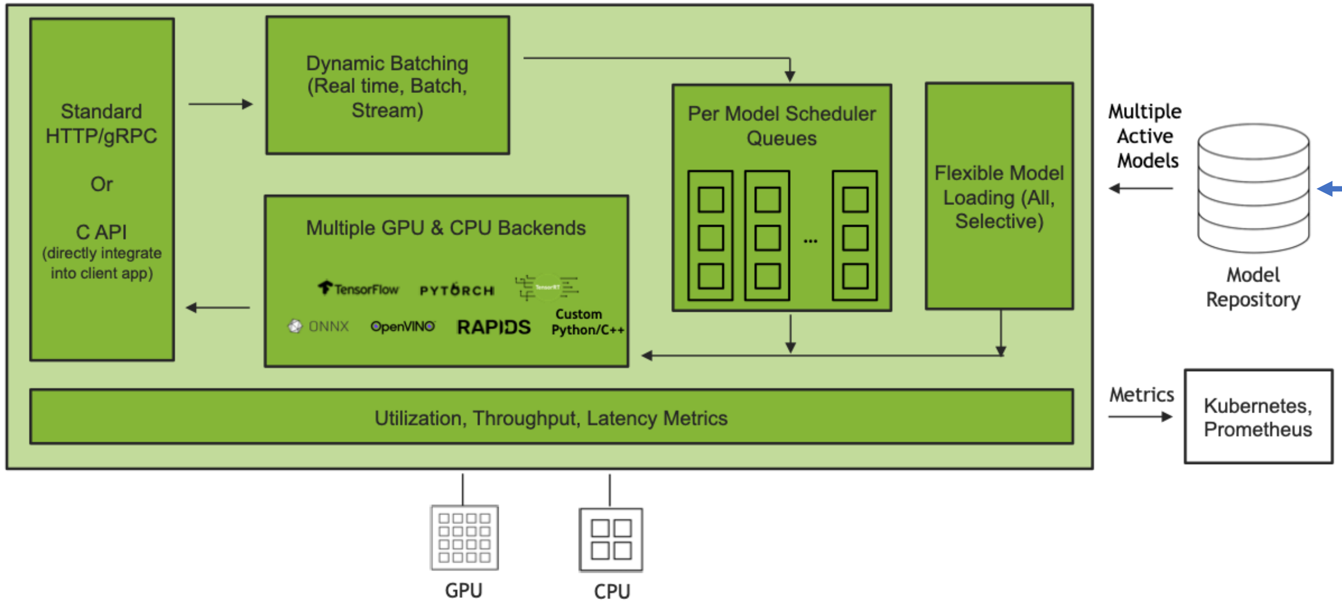
Recap

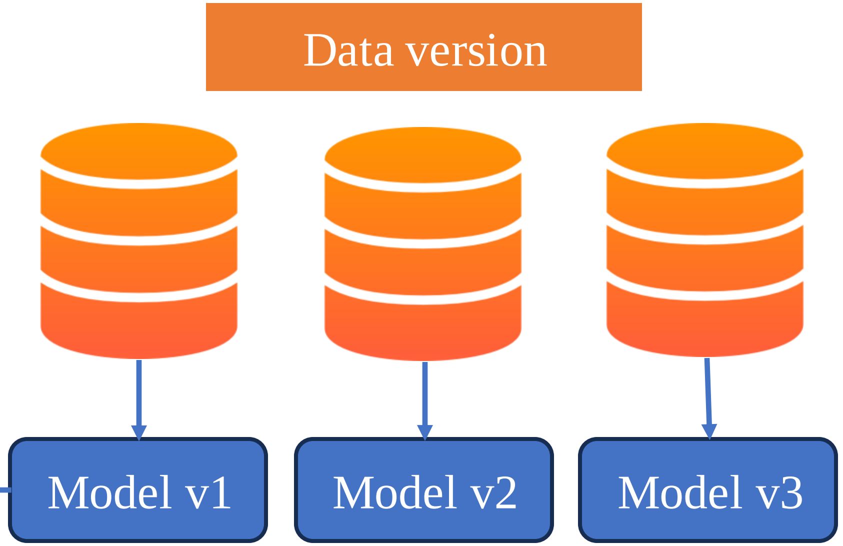
❖ Model Serving Optimization

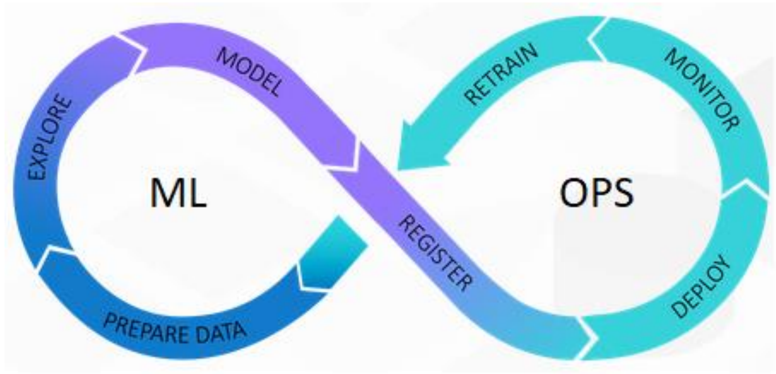


Recap

❖ Machine Learning Lifecycle

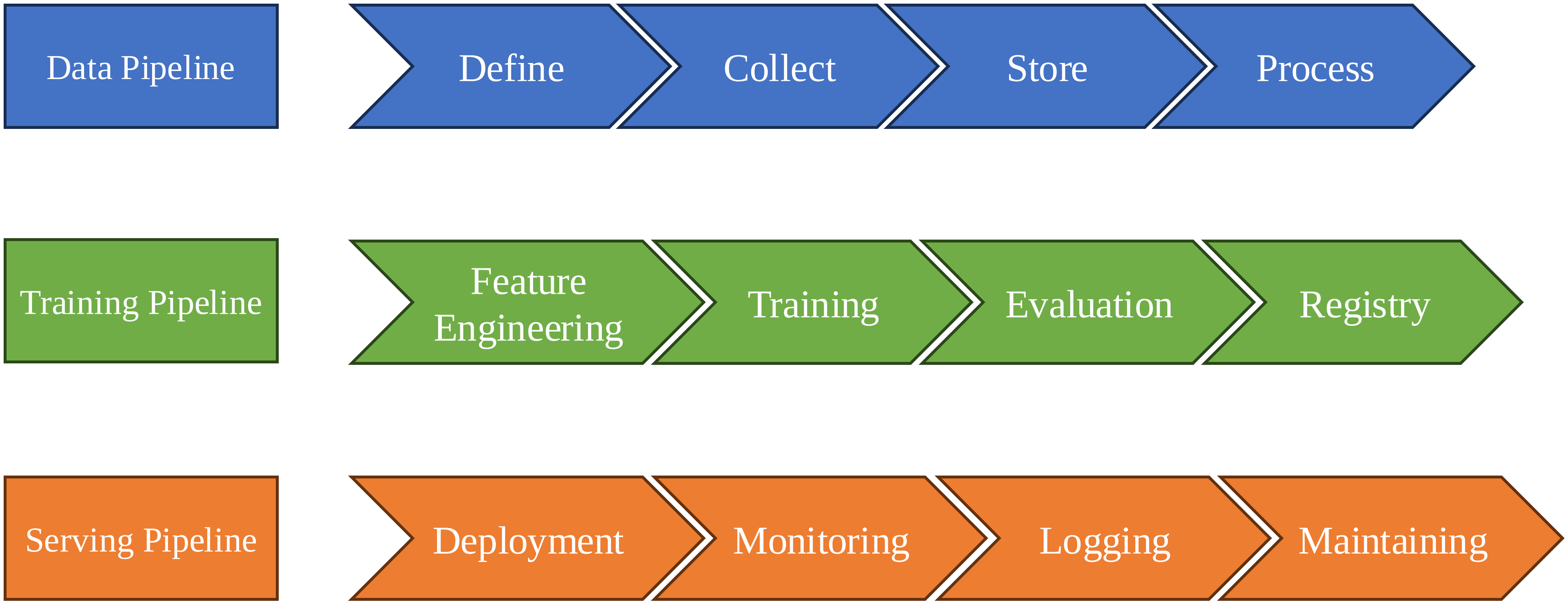






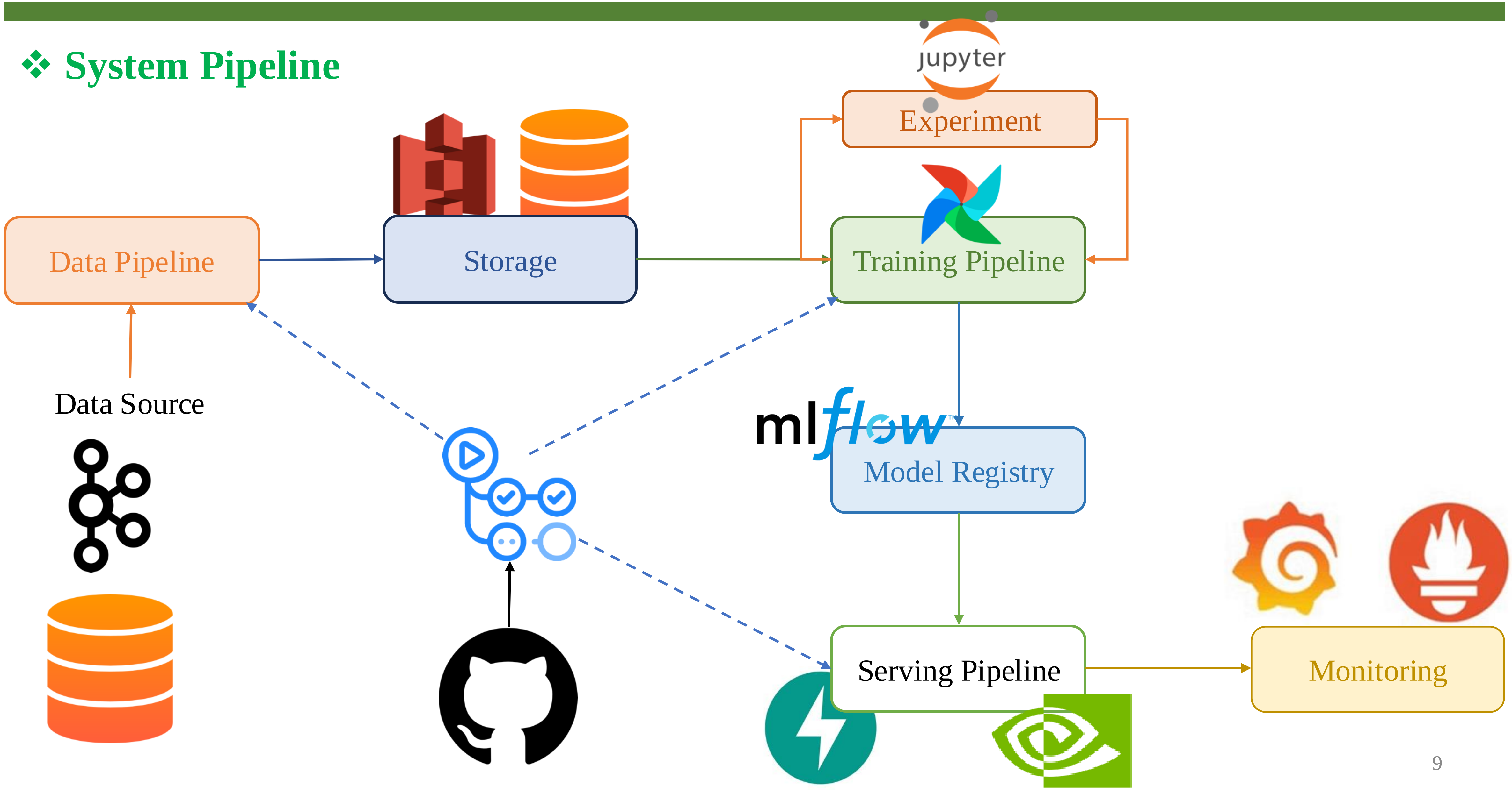
Recap

❖ System Pipeline



Recap

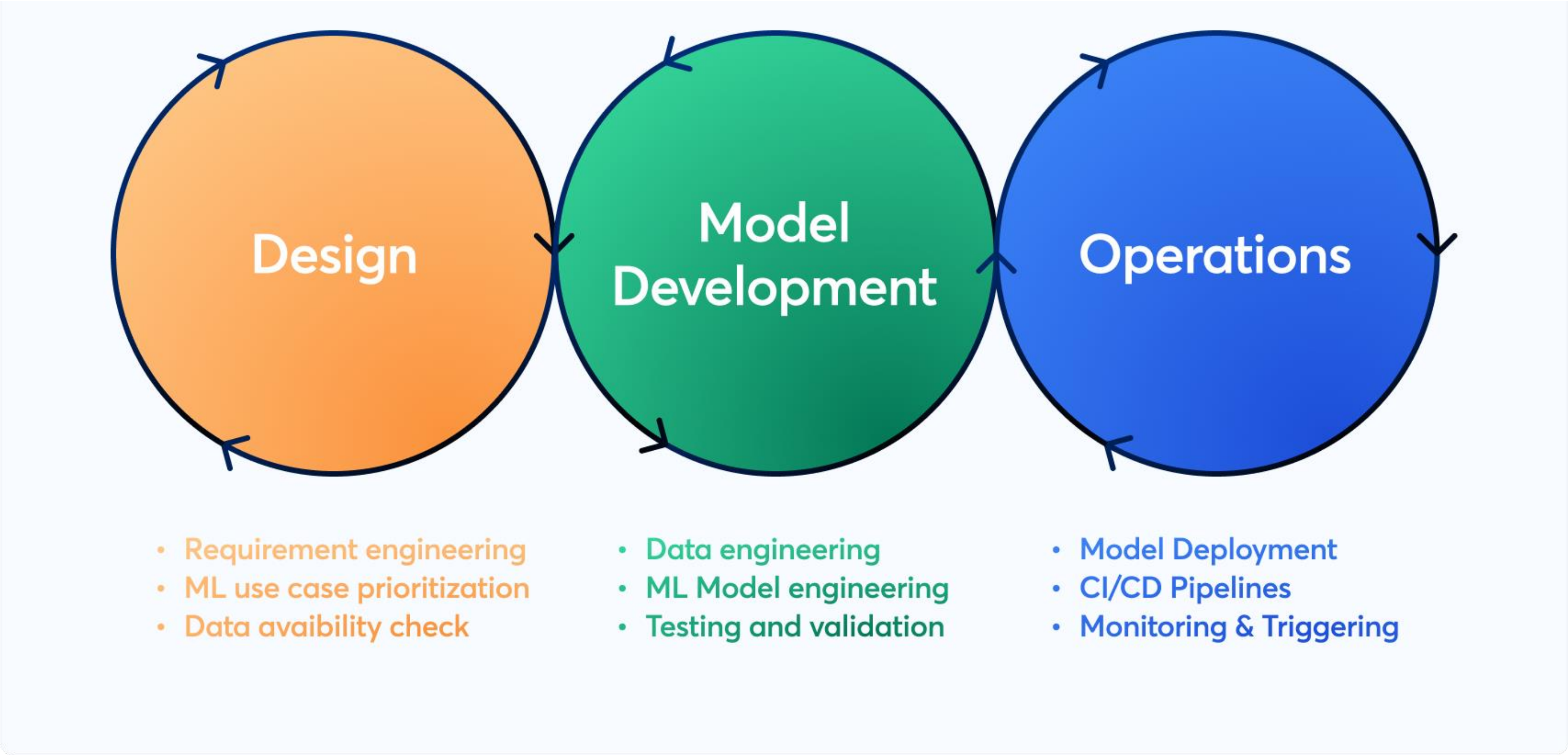
❖ System Pipeline



Introduction

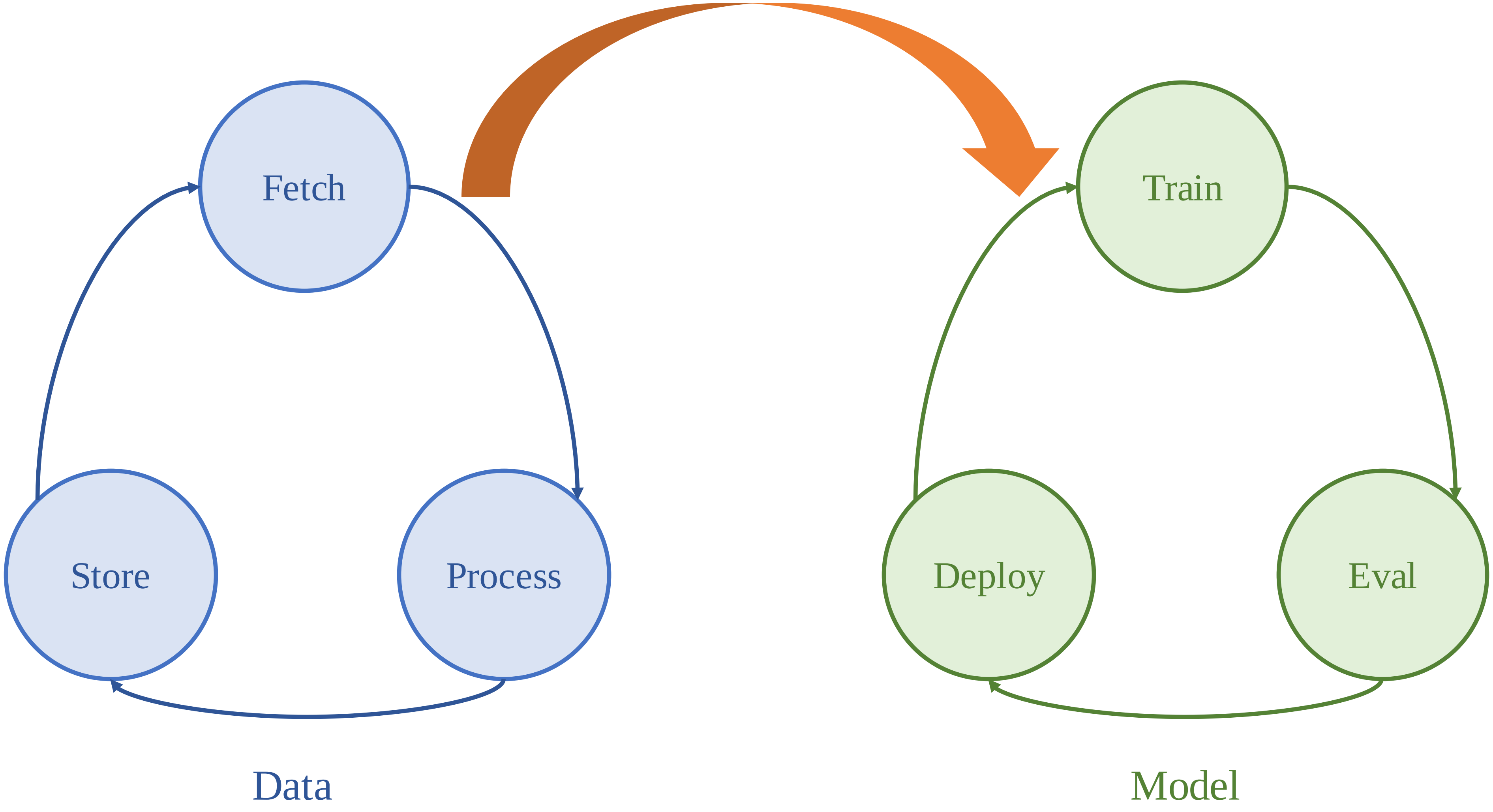
Introduction

❖ Getting Started



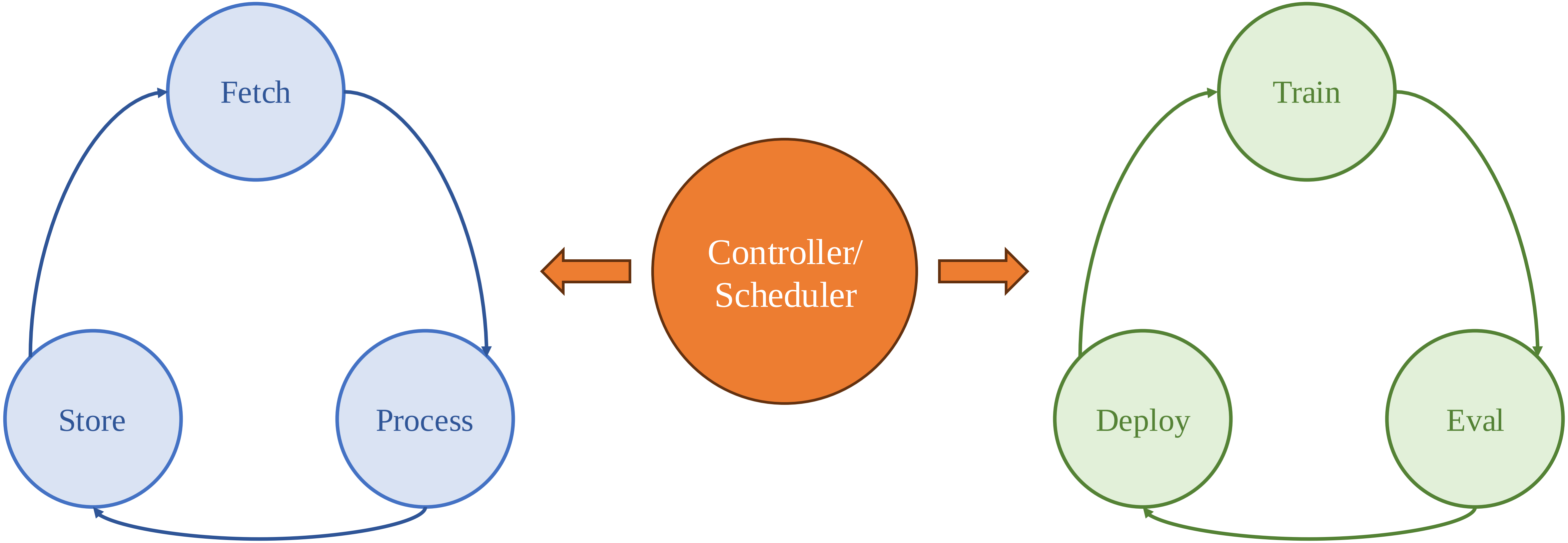
Introduction

❖ Cycle Processing



Introduction

❖ Controller/Scheduler



Airflow

Airflow

❖ Getting Started

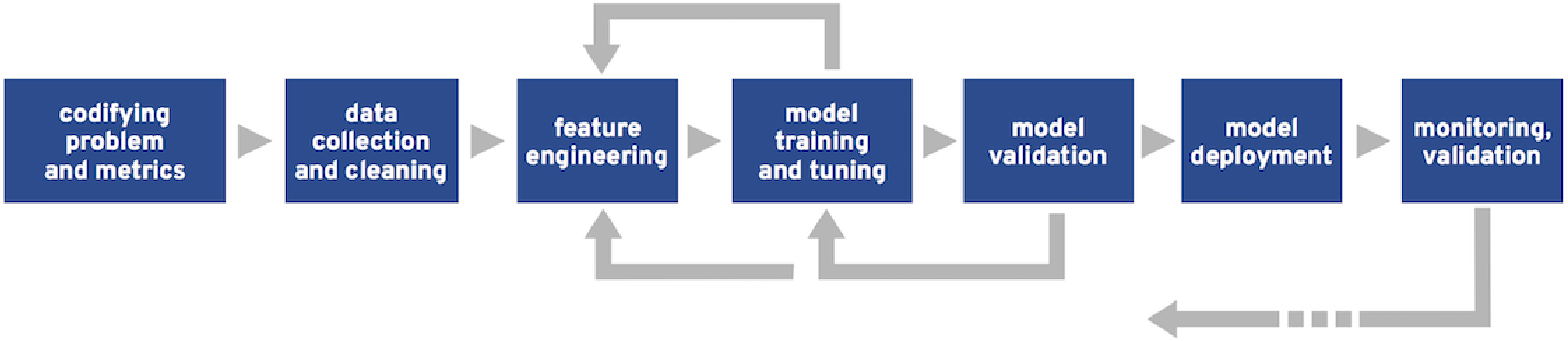
An open-source workflow management platform for data engineering pipelines

- It started at Airbnb in 2014
- Manage the company's complex workflows
- Workflows as code

Airflow

❖ Workflow

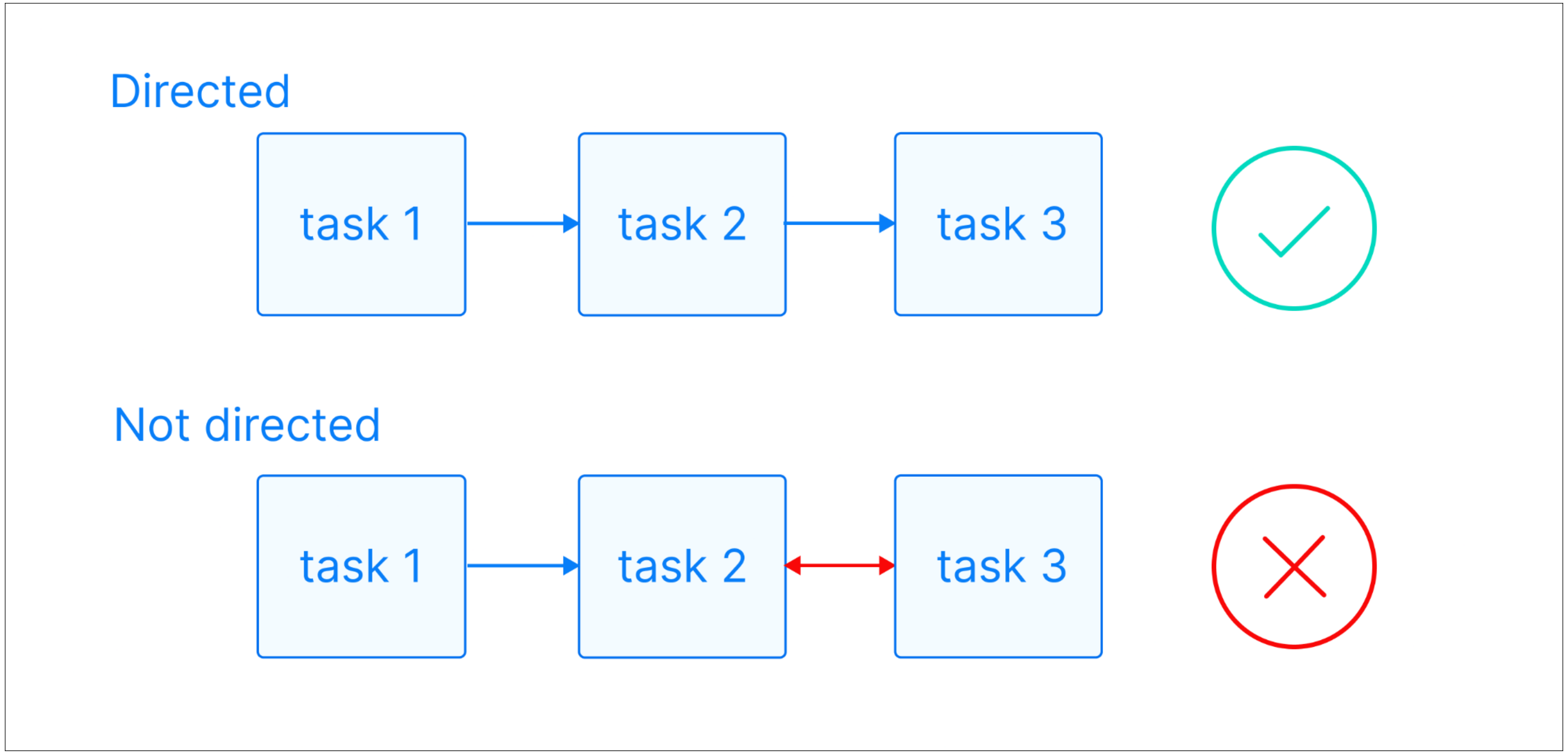
A workflow is a series of tasks that are arranged in a DAG. The DAG specifies the order in which the tasks should be executed.



Airflow

❖ Directed Acyclic Graph

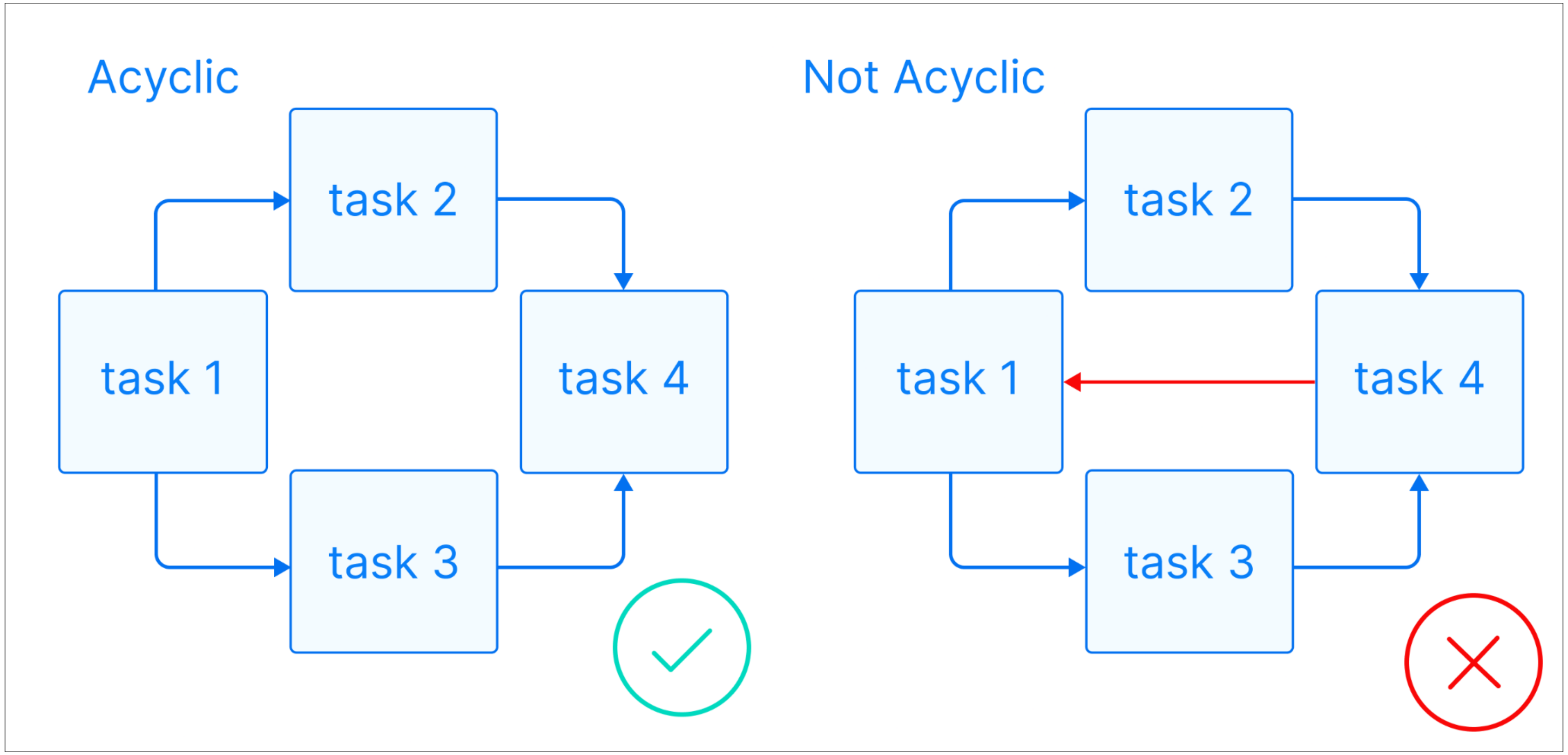
A task can be either upstream, downstream, or parallel to another task (clear direction)



Airflow

❖ Directed Acyclic Graph

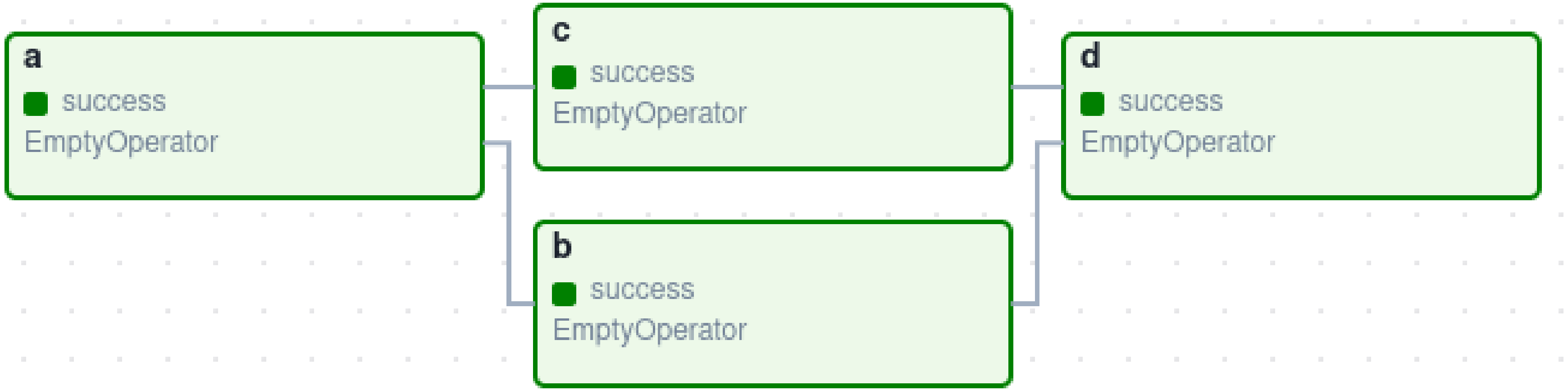
A task cannot depend on itself, nor can it depend on a task that ultimately depends on it (NO circular dependencies)

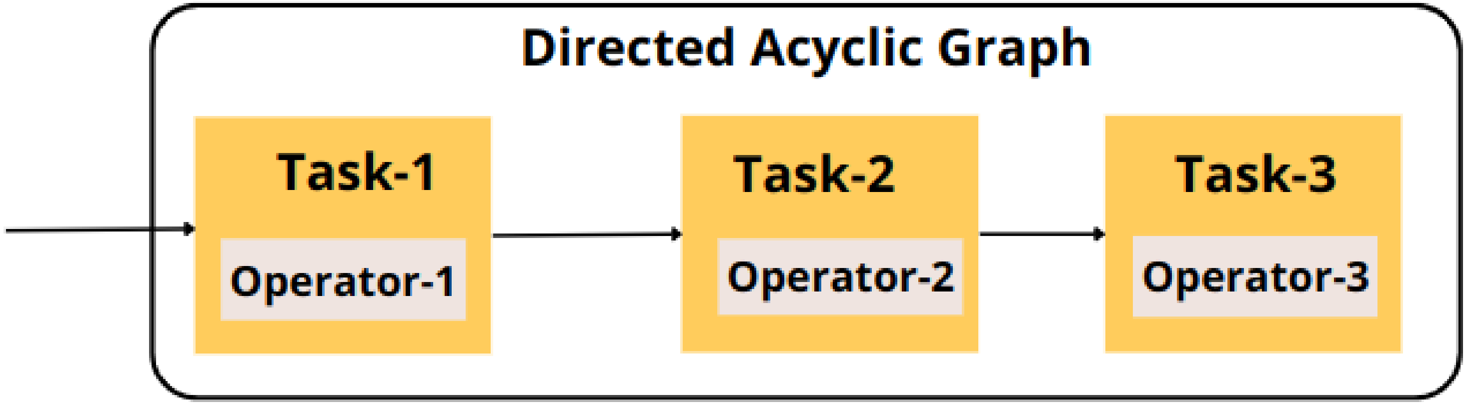


Airflow

❖ Directed Acyclic Graph

A DAG is a graph, which is a structure consisting of nodes and edges.



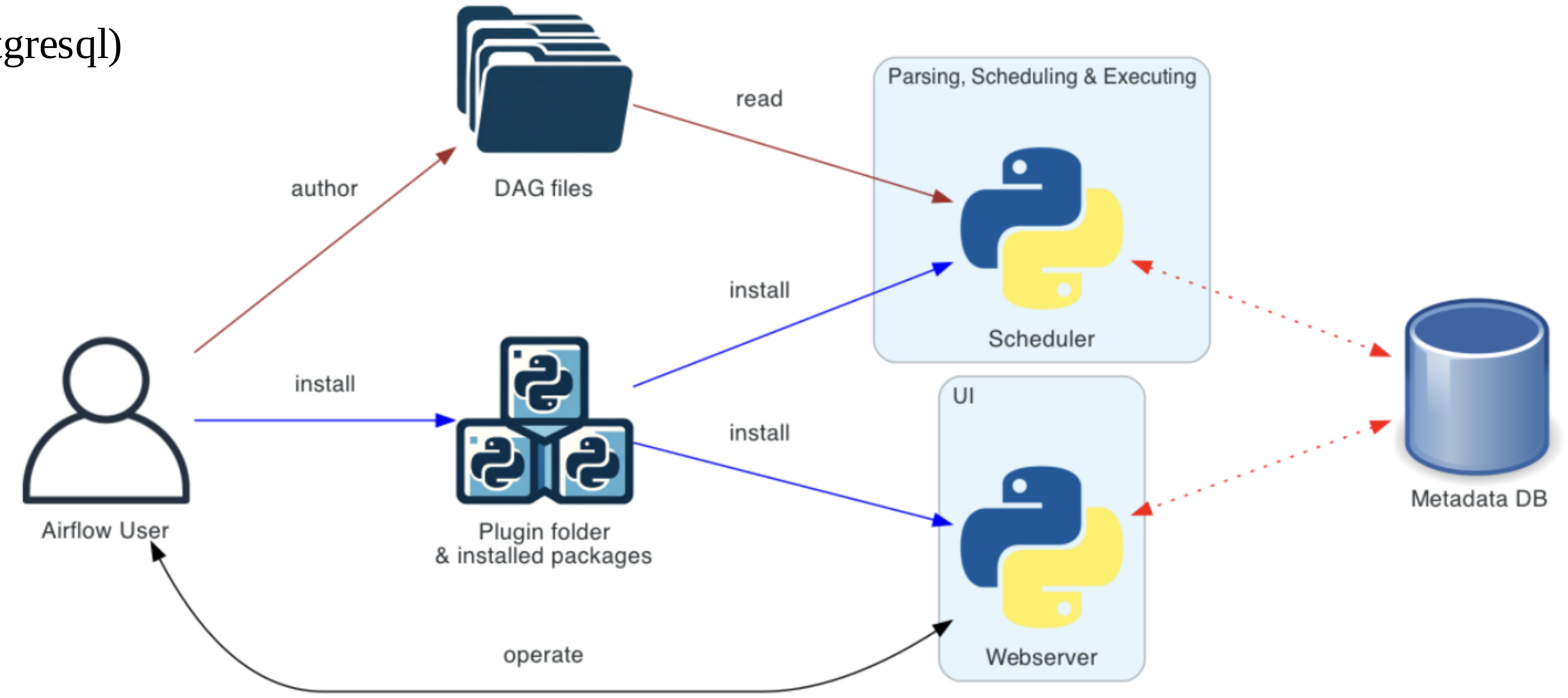


Airflow

❖ Airflow Architecture

A minimal Airflow installation consists of the following components:

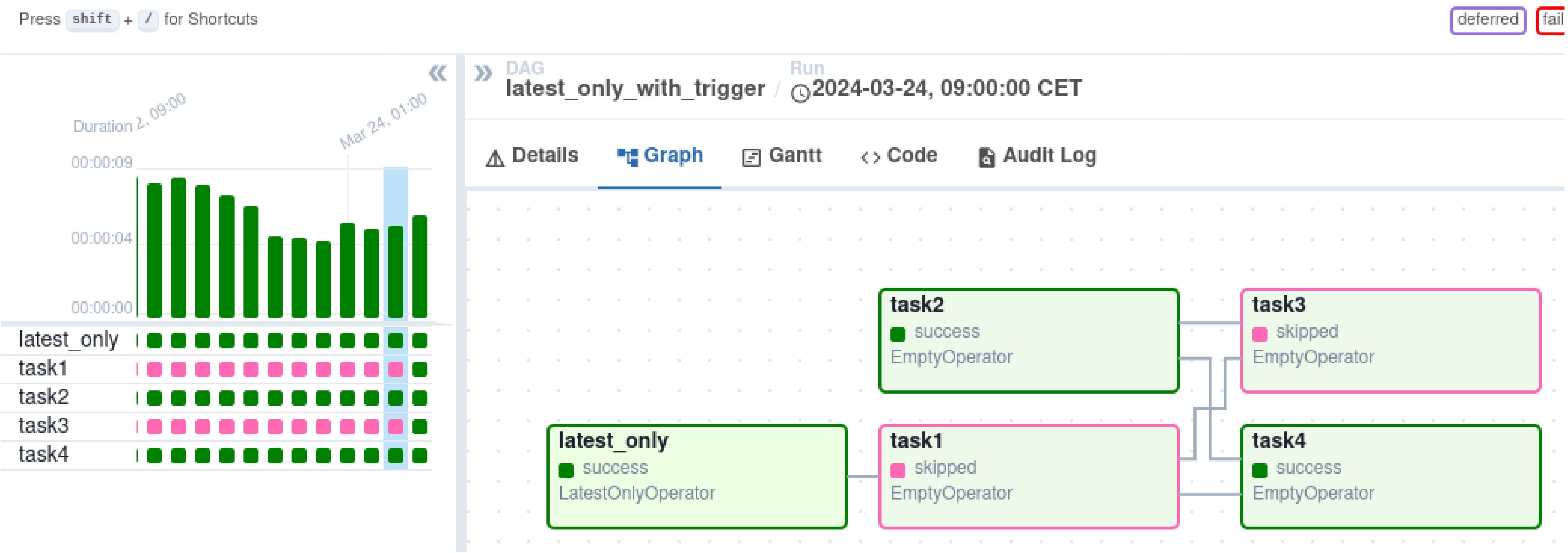
- Scheduler
- Webserver
- A metadata database (Postgresql)
- DAG files
- Message bus (Redis)



Airflow

❖ Core Concepts

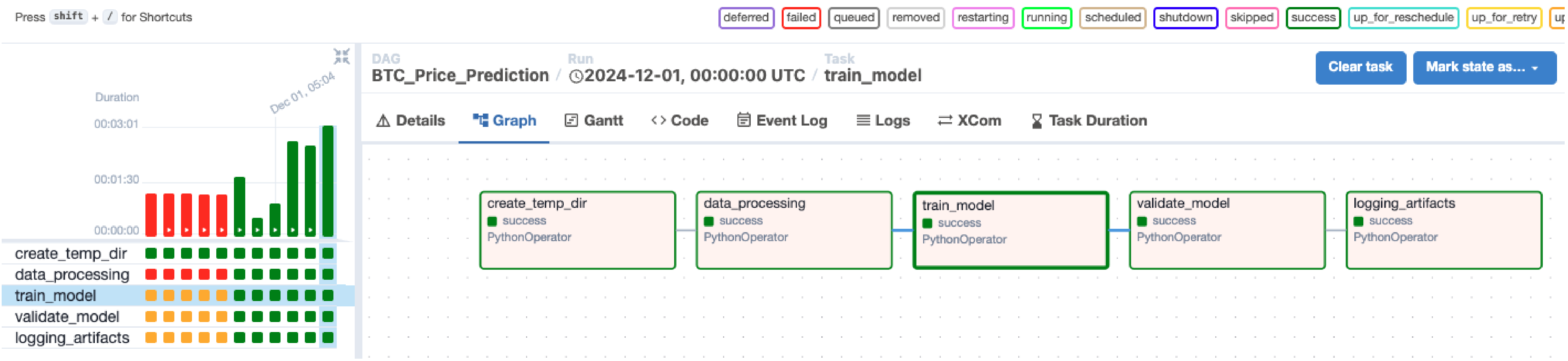
DAGs: Organize tasks with dependencies and relationships to say how they should run.



Airflow

❖ Core Concepts

DAGs: Organize tasks with dependencies and relationships to say how they should run.



Airflow

❖ Core Concepts

Task: Arranged into DAGs, and then have upstream and downstream dependencies set between them in order to express the order they should run in.

```
create_temp_dir_task >> data_processing_task >> train_model_task >> validate_model_task >> logging_artifacts_task

```

Individual task dependencies

```
create_temp_dir_task \
>> [fetch_BTC_data_task, data_processing_task] \
>> [fetch_ETH_data_task, data_processing_task] \
>> [fetch_Gold_data_task, data_processing_task] \
>> train_model_task >> validate_model_task >> logging_artifacts_task
```

Group task dependencies

Airflow

❖ Core Concepts

Operator: A template for a predefined Task
Eg. BashOperator, PythonOperator, EmailOperator, …

BashOperator	Executes a bash command
PythonOperator	Calls an arbitrary Python function
DockerOperator	Execute a command inside a docker container
MySqlOperator	Executes sql code in a specific MySQL database

Airflow

❖ Core Concepts

Operator: A template for a predefined Task
Eg. BashOperator, PythonOperator, EmailOperator, …

EmailOperator	Sends an email
HTTPOperator	Calls an endpoint on an HTTP system to execute an action
S3FileTransformOperator	Copies data from a source S3 location to a temporary location on the local
SlackAPIOperator	Send a file to a Slack channel

Airflow

❖ Core Concepts

```python
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'trigger_rule': 'all_success',
    'start_date': days_ago(1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=1),
}

```

```python
with DAG(
    'BTC_Price_Prediction',
    default_args=default_args,
    description='A simple ML pipeline demonstration',
    schedule_interval=timedelta(days=1),
    ) as dag:

    data_processing_task = PythonOperator(
        task_id='data_processing',
        python_callable=data_processing,
        dag=dag,
    )
    train_model_task = PythonOperator(
        task_id='train_model',
        python_callable=train_model,
        dag=dag,
    )
    validate_model_task = PythonOperator(
        task_id='validate_model',
        python_callable=validate_model,
        dag=dag,
    )
```

Airflow

❖ Pass data between tasks

Sharing data between tasks is a very common use case in Airflow.

Pass data via XCom

Pass data via External System

Ref: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/xcoms.html

Airflow

❖ Pass data between tasks

Pass data via XCom

- A built-in Airflow feature for sharing task data.
- XComs should be used to pass small amounts of data between tasks.
- Serialization: JSON, pandas DataFrame, Delta Lake tables, Apache Iceberg tables.
- Custom XCom Backend for more customized configuration

Airflow

❖ Pass data between tasks

Pass data via External System

External System: Cloud file storage such as S3, GCS, or Azure Blob Storage.

Save data to External System at the end of one task, then reading data from that system in the next task.

Great way to pass data that is too large
Eg. raw data file, large dataframe…

Pipeline

Pipeline

❖ Version Control

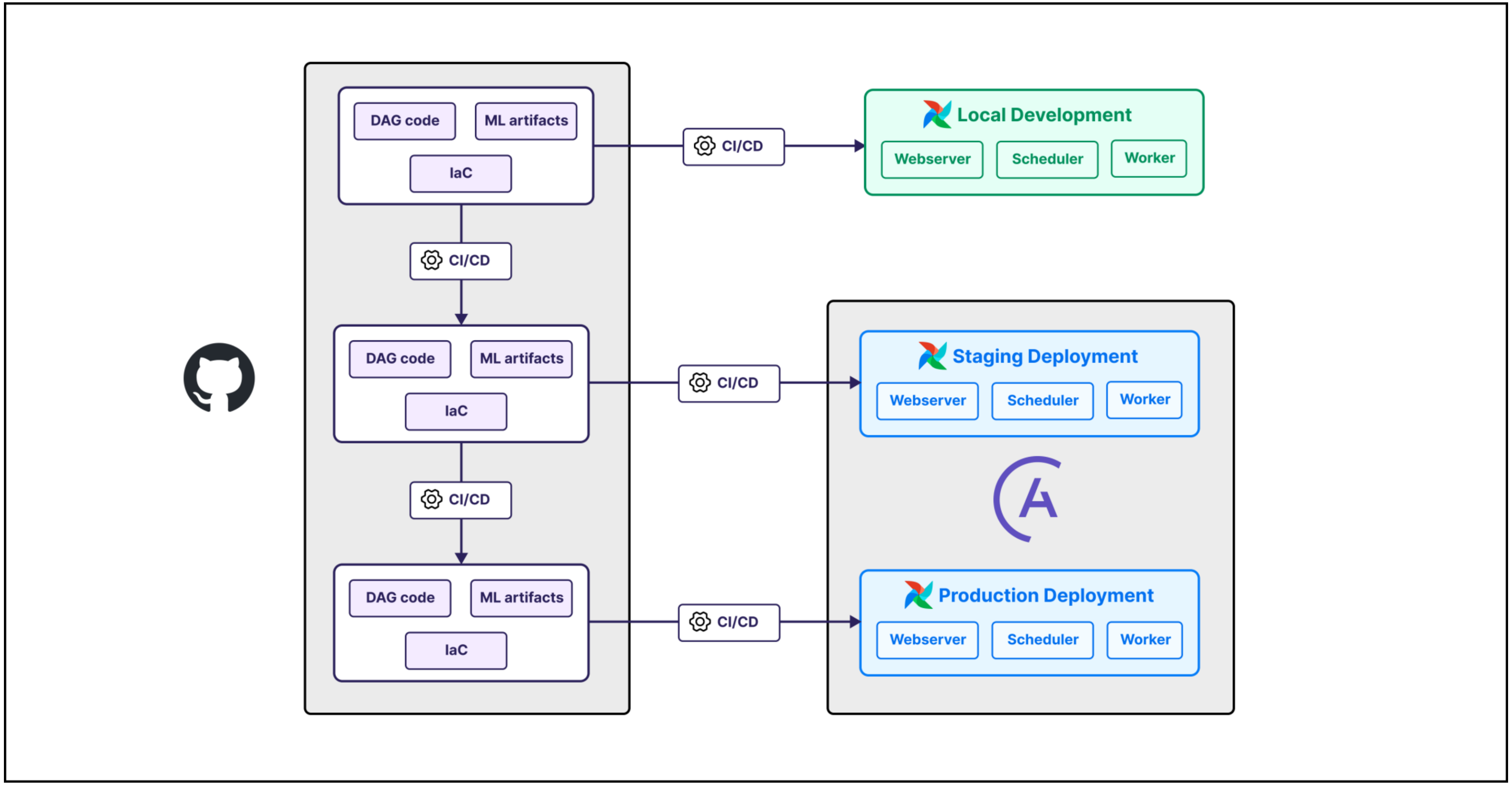
- Store Airflow code and configuration in Git
- Set up **dev**, **UAT** (User Acceptance Testing), and **production** branches in Git and mapping them to associated Airflow environments
- **Test** and **lint** for all Airflow code before deployment

Ref: https://www.astronomer.io/docs/learn/airflow-mlops/#airflow-features-for-mlops

Pipeline

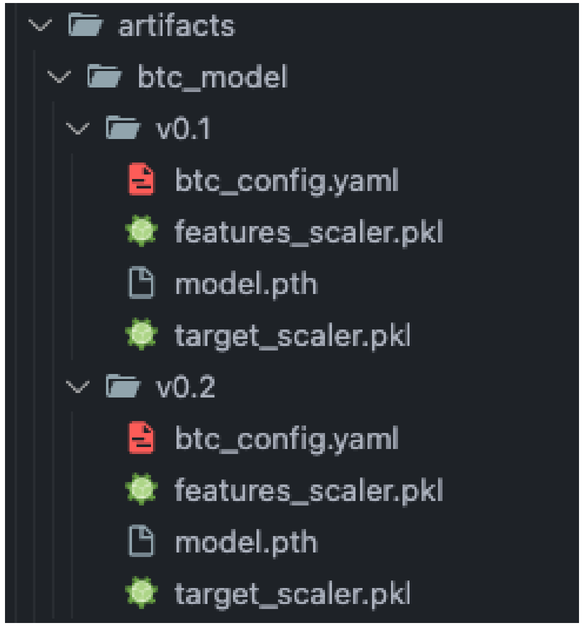
❖ Infrastructure as code (IaC)

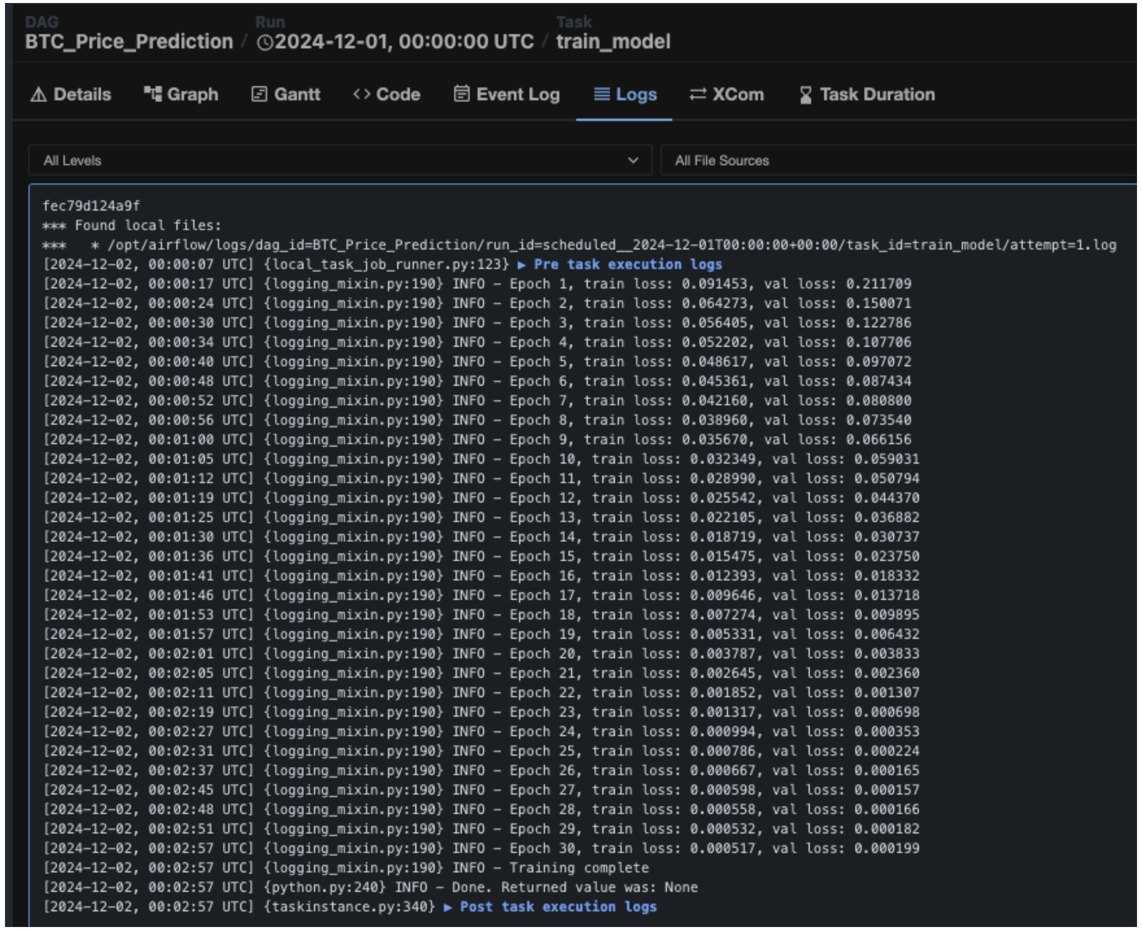
Use the same CI/CD process by defining IaC



Pipeline

❖ ML models registry (next)



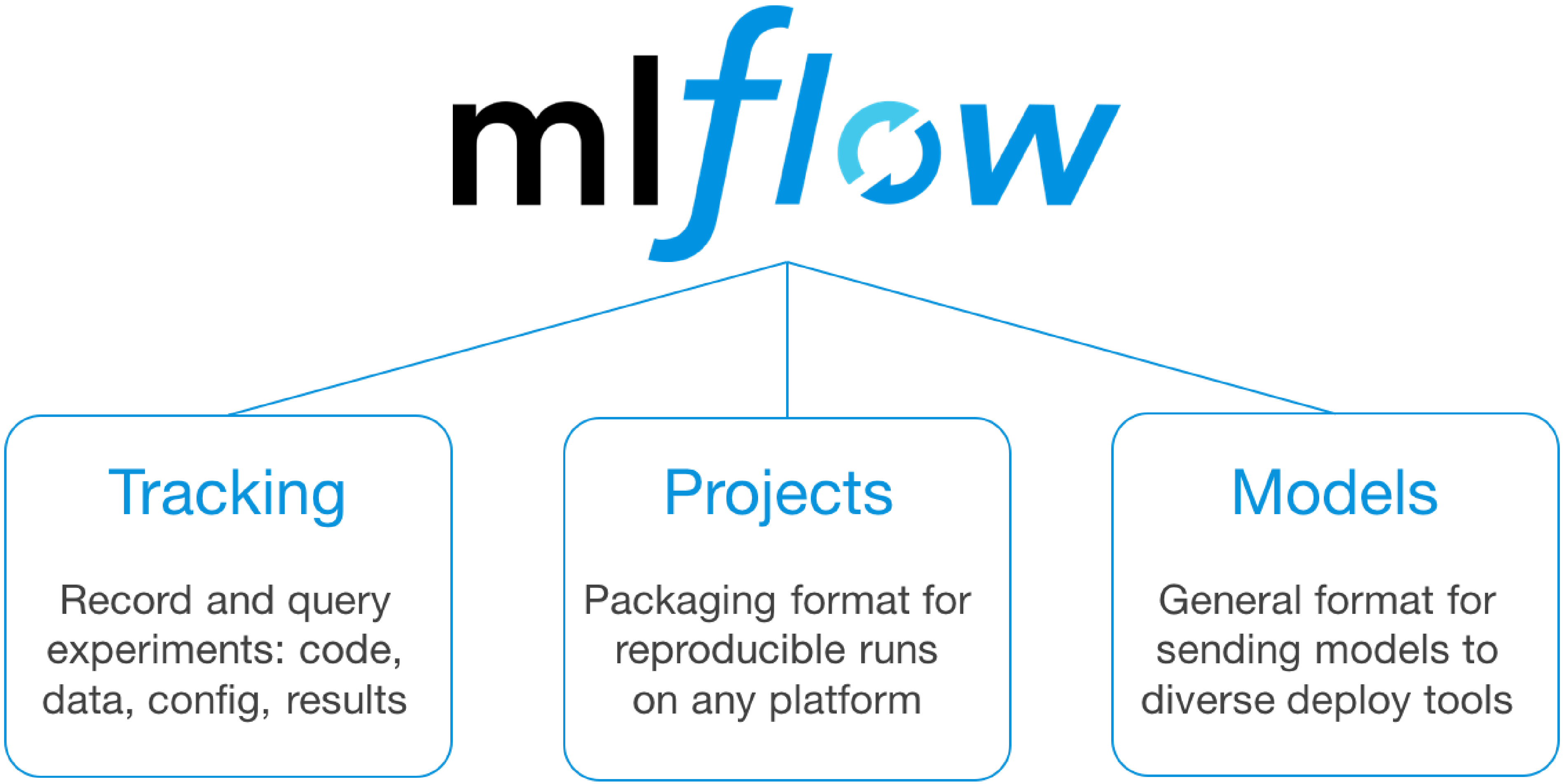


mlflow™

Logging to

Pipeline

❖ Mlflow (next)



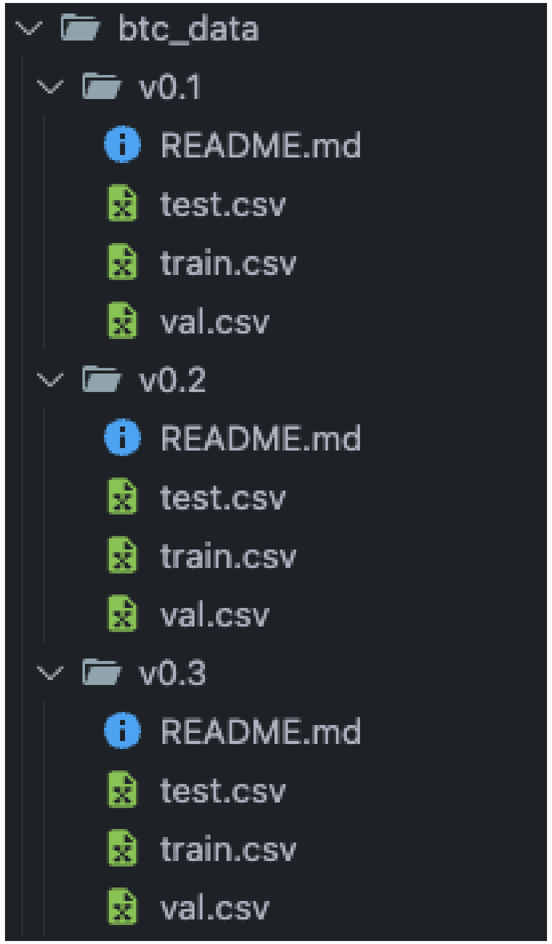
Practice

Practice

❖ Dataset

BTC price data is downloaded from investing.com website. The dataset is collected from 1/2016 to 10/2024 and split to 3 versions: v0.1, v0.2 and v0.3.

- v0.1: 1/2016 to 12/2023
- v0.2: 1/2016 to 2/2024
- v0.3: 1/2016 to 10/2024



Ref: https://www.investing.com/crypto/bitcoin/historical-data

Practice

❖ Model

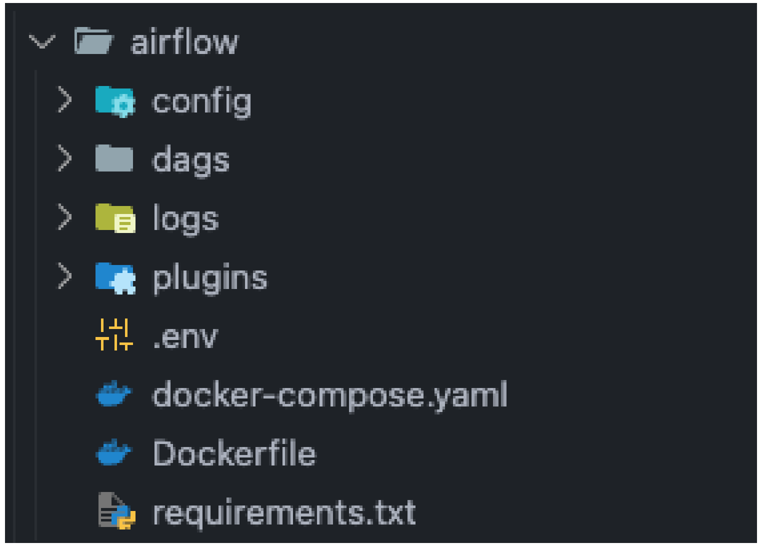
Model is used for training time-series data is RNN. The model contains 2 RNN layer and 2 fully-connected layers to predict the price of BTC in the future.

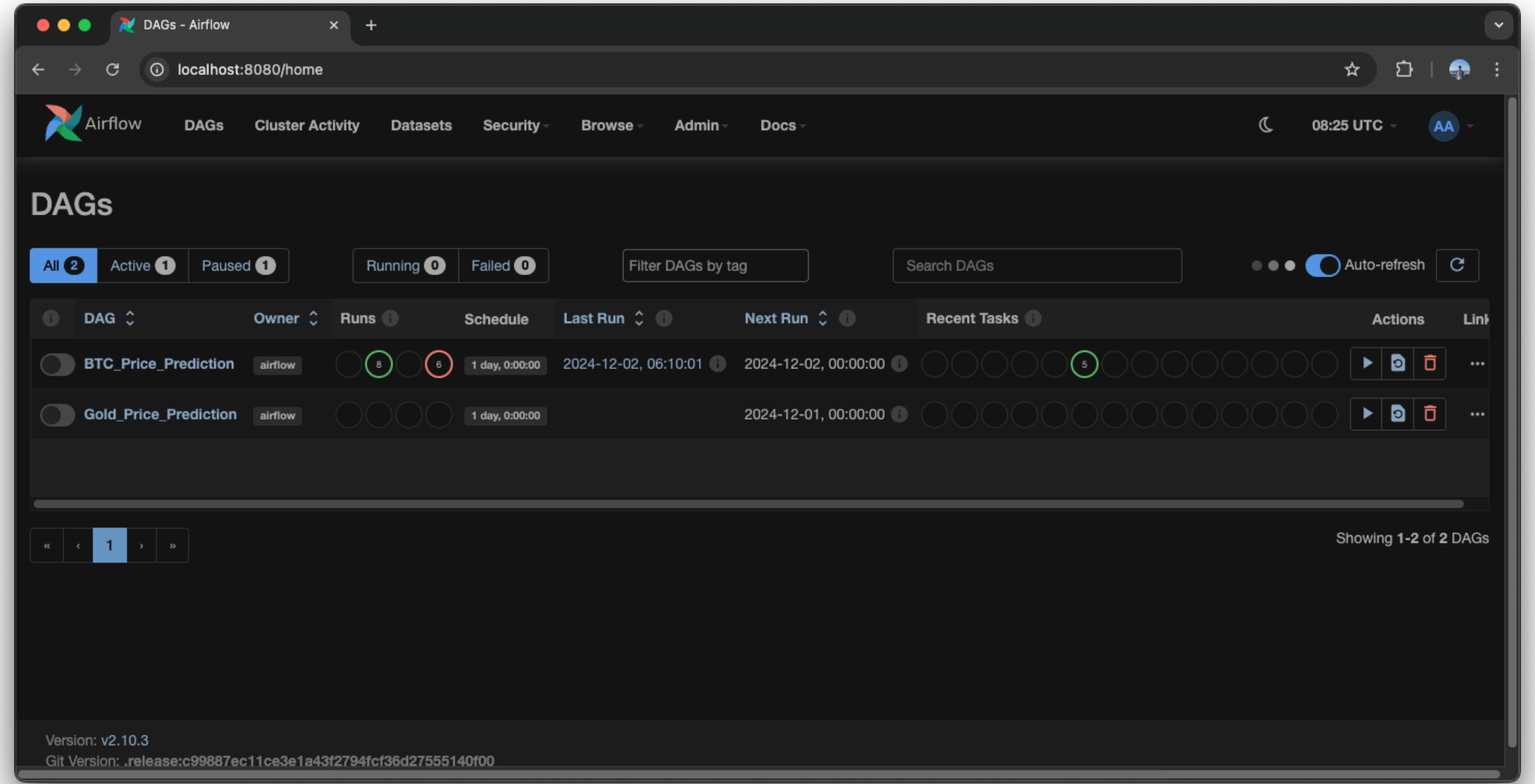
```python
class RNN_Model(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers):
        super().__init__()
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True, bidirectional=True)
        self.fc1 = nn.Linear(hidden_size * 2, hidden_size)
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.relu = nn.ReLU()

    def forward(self, x):
        out, _ = self.rnn(x)
        x = self.fc1(out[:, -1, :])
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Practice

❖ Airflow server

Use Docker and Docker Compose to start Airflow server and related components.





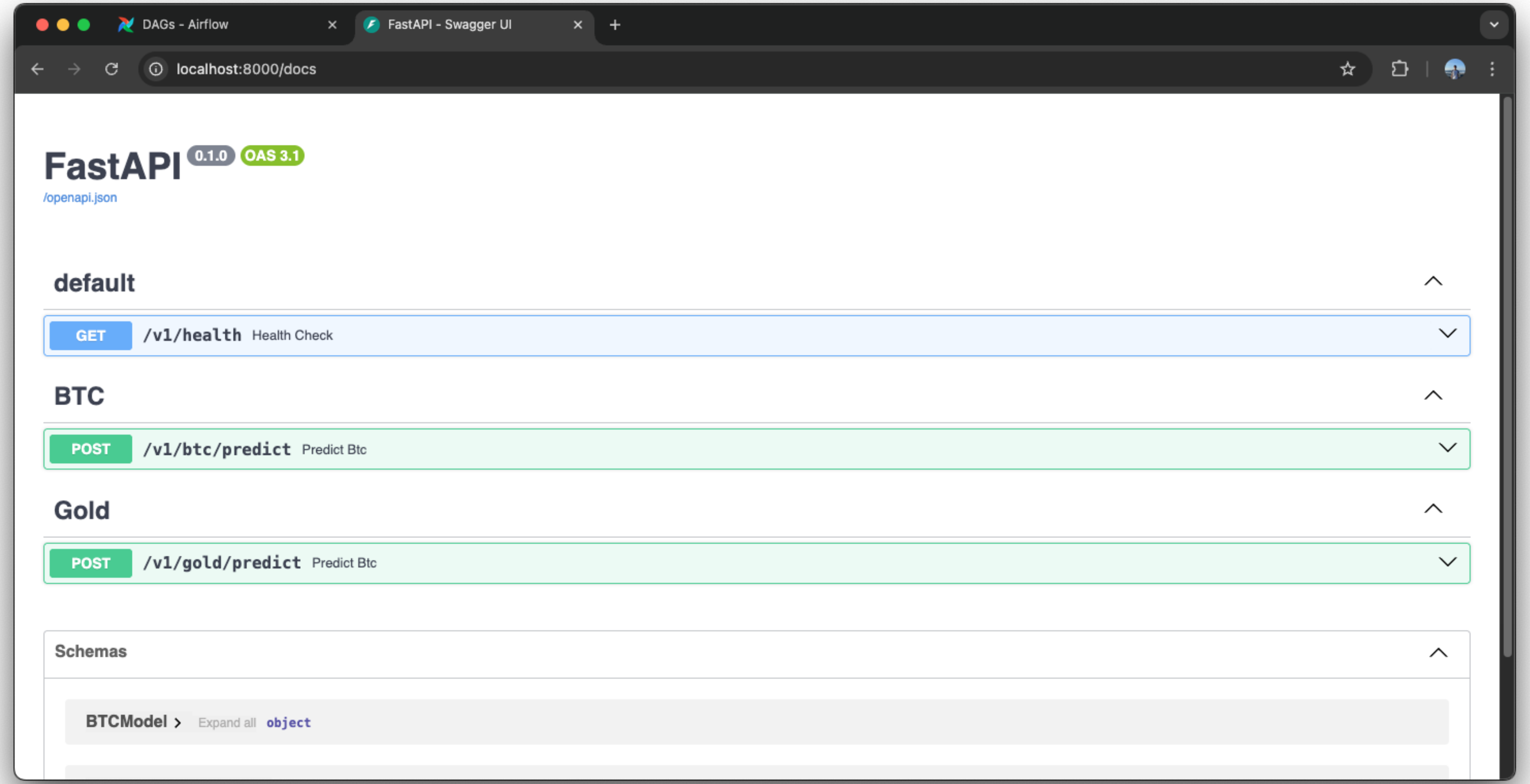
Default user and password of airflow account is: airflow

Ref: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html

Practice

❖ Backend server

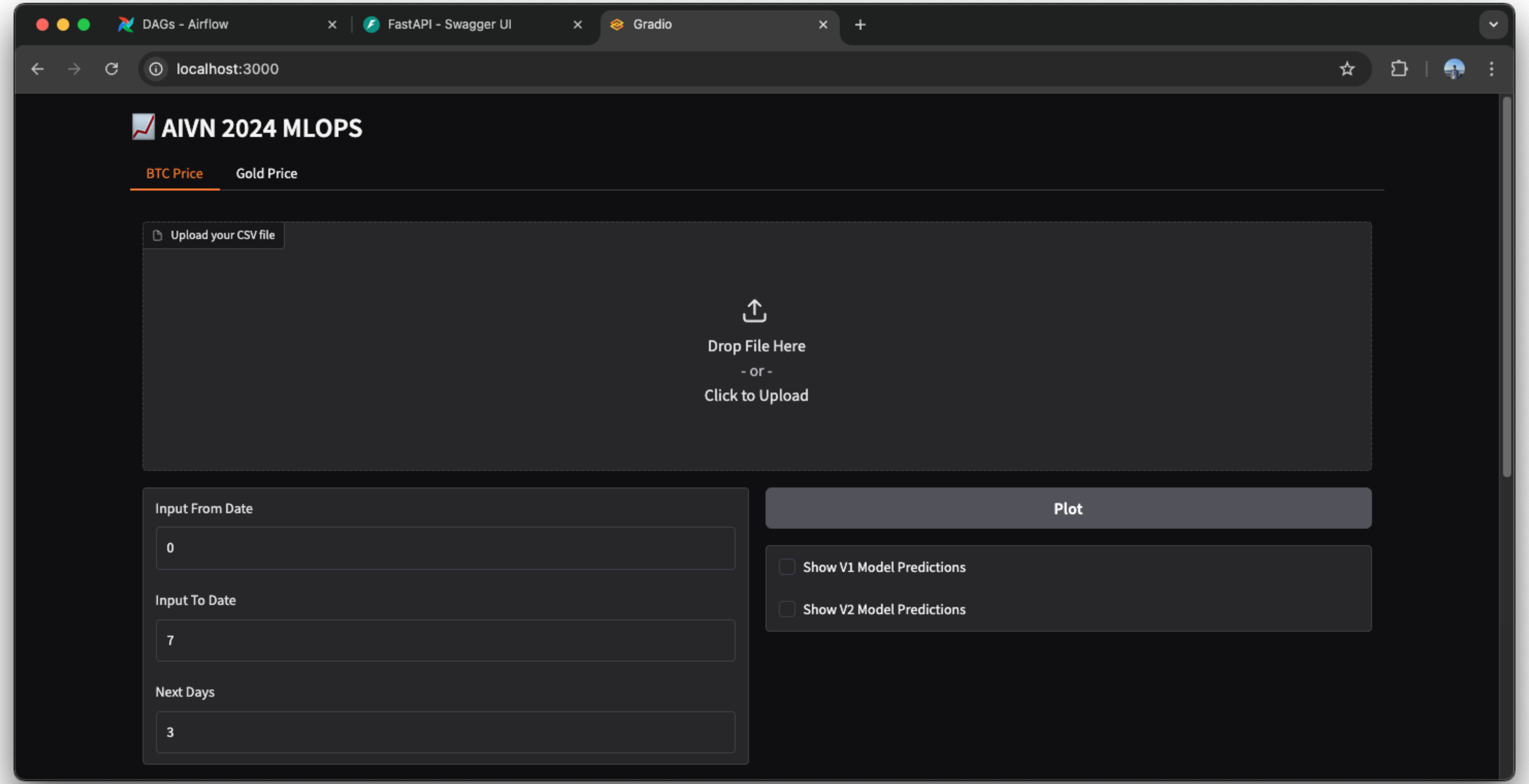
FastAPI is used for backend server



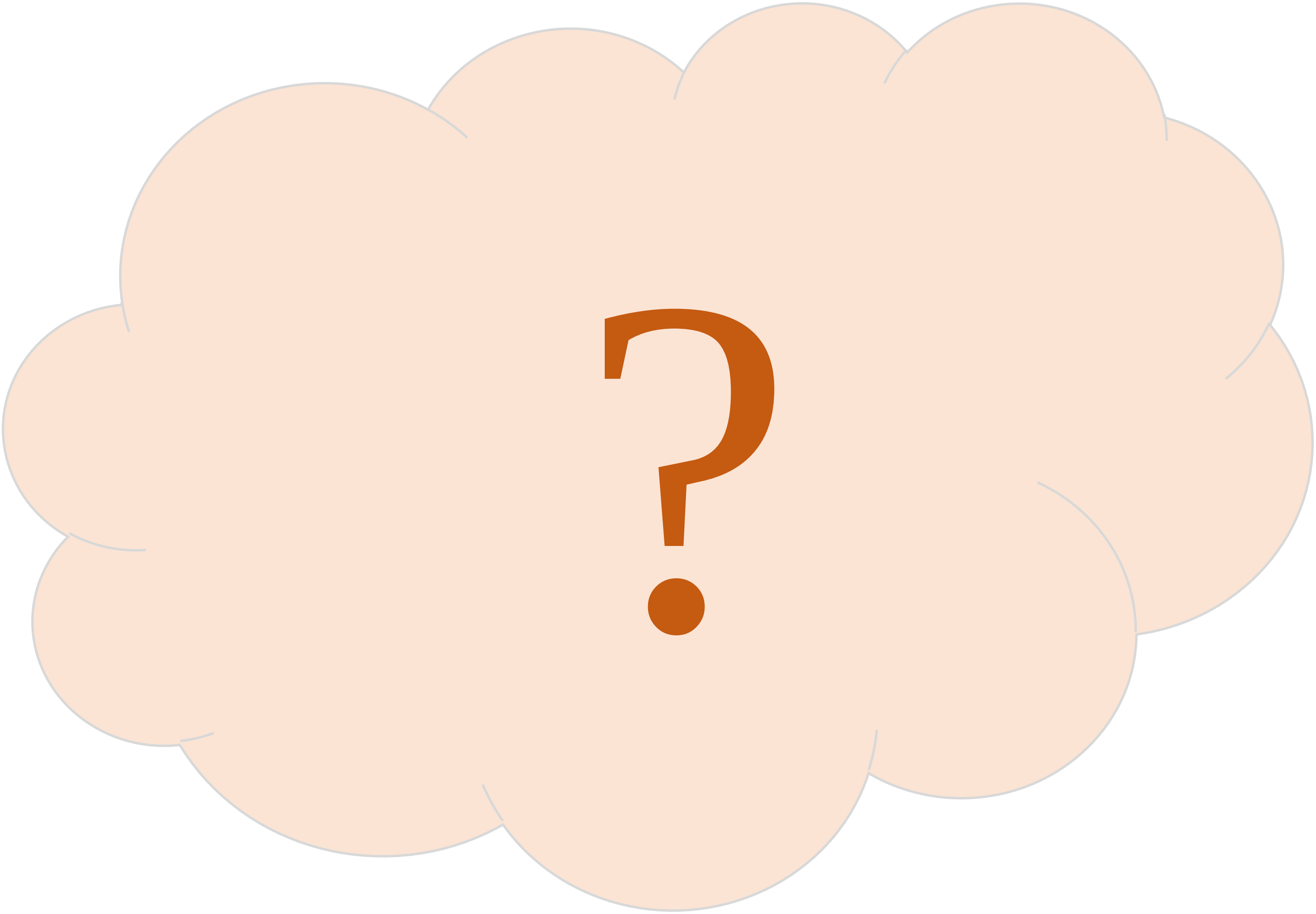
Practice

❖ Frontend UI

Gradio is used for frontend inferface



Question




Thank you!