ILT 5 - TensorFlow Data and Deployment

Ground Rules

Observe the following rules to ensure a supportive, inclusive, and engaging classes

- Give full attention in class
- Mute your microphone when you're not talking
- Keep your camera on
- Turn on the CC Feature on Meet
- Use raise hand or chat to ask questions
- Make this room a safe place to learn and share

Outline Session

- Introduction to Machine Learning Deployment
- Data Pipelines
- Federated Learning

bangkit

Introduction to ML Deployment

bangkit

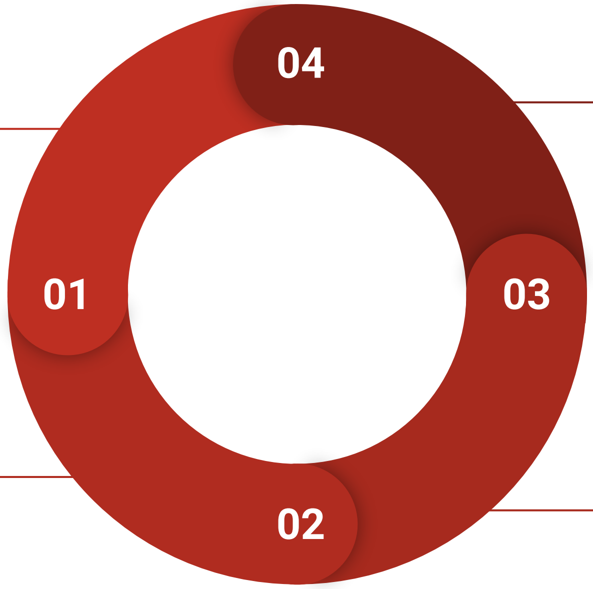
Lifecycle of a ML Project

Project Planning & Setup

At this phase, we want to decide the problem to work on, determine the requirements and goals, as well as figure out how to allocate resources properly

Data Collection & Labeling

At this phase, we want to collect & organize data (images, text, tabular, etc.) & potentially annotate them with ground truth, depending on the specific sources where they come from



Deployment & Monitoring

At this phase, we put the model into production, write the software needed to make the model run, and make predictions. We need also to monitor and maintain the system. If the data changes, we need to update the model

Model Training & Debugging

At this phase, we want to implement baseline models quickly, find and reproduce state-of-the-art methods for the problem domain, debug our implementation, and improve the model performance for specific tasks

The Challenges of Model Deployment

- ML models are sensitive to the semantics, quantity, and completeness of incoming data
- The performance of ML models in production degrades over time due to changes in real data
- ML models only work with data from specific demographic

Model Deployment Options

Centralize model in server

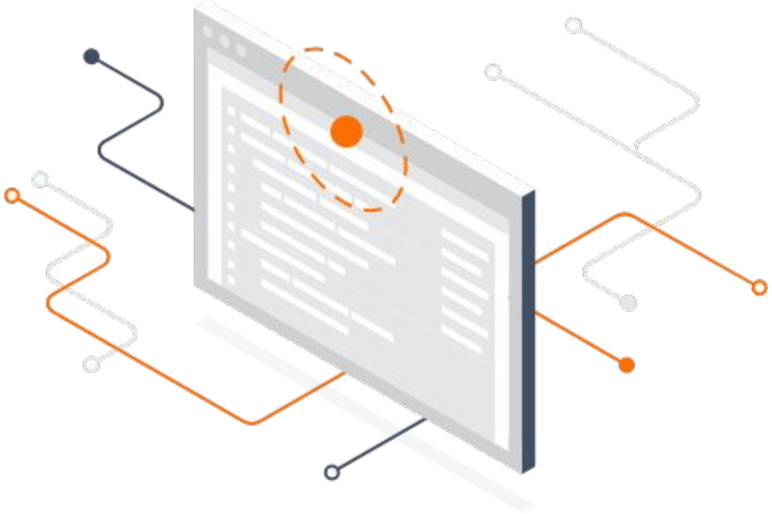
Distribute model on user device

ML Deployment
TensorFlow.js

What is TensorFlow.js?

An open-source JavaScript library for training and deploying machine learning models in the client's browser or Node.js server

- Run existing models
- Retrain existing models
- Develop ML with JavaScript



General Steps in TensorFlow.js

Build a Model → **Save Model** → **Convert** → **Deploy**

Build a Model	Save Model	Convert	Deploy
Build & train a ML model with TensorFlow	Save model in SavedModel or .h5 format	Convert it to JSON format	Deploy the Converted model on a webpage

Convert Models into JSON Format

TensorFlow → Save Model → TF js Converter → Model JSON Format

```
model.save(saved_model_path)
```

```
!tensorflowjs_converter
--input_format=keras
saved_model_path tfjs_model_path
```

What Can TensorFlow.js Do?

* Teachable Machine Project

ML Deployment
TensorFlow Lite

What is TensorFlow Lite?

An open-source deep learning framework to run TensorFlow models on-device

- Optimized for on-device machine learning
- Multiple platform support
- Diverse language support
- Hardware acceleration & model optimization

The Challenges of on-Device Models

Running on-device models means we need to handle diverse devices & we do not have access to actual data

Need to balance between:

- Power efficiency
- Inference latency
- Model accuracy & complexity

General Step in TensorFlow Lite

Build a Model → **Save Model** → **Convert** → **Deploy**

Build a Model
Build & train a ML model with TensorFlow

Save Model
Save model in SavedModel or .h5 format

Convert
Convert it to TFLite model

Deploy
Deploy the converted model on a mobile/edge device

Convert Models into TF Lite Model

TensorFlow → Saved Model → TF Lite Converter → TF Lite Model

```
tf.saved_model.save(model, saved_model_path)
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(export_dir)
tflite_model = converter.convert()
tflite_model_file = pathlib.Path('model.tflite')
tflite_model_file.write_bytes(tflite_model)
```

What Can TensorFlow Lite Do?

bangkit

ML Deployment
TensorFlow Serving

What is TensorFlow Serving?

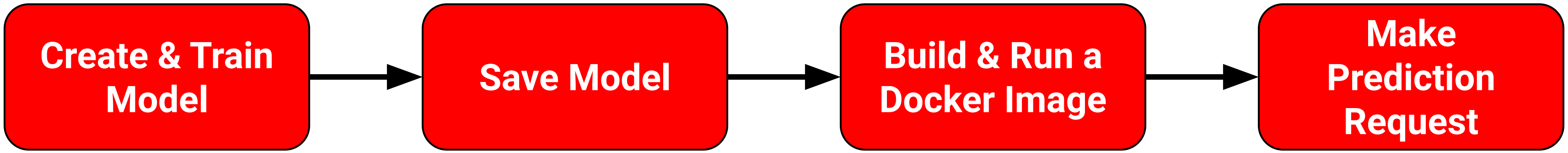
TensorFlow Serving is a flexible & high-performance serving system for machine learning models, designed for production environments

TensorFlow Serving

TensorFlow Serving allows us to have a centralized model

- Easy to manage model version
- Easy to manage hardware resources based on demand
- Can have multiple serving processes

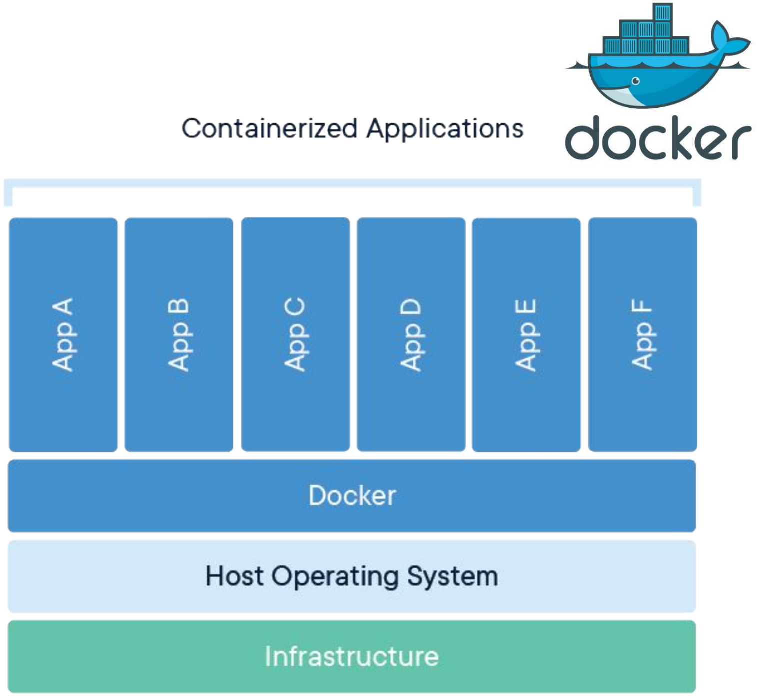
Deploy Model with TF Serving



Whats is Container? Docker?

Container is a standard unit of software that packages up code and all its dependencies so the application runs portably.

- Portable
- Lightweight
- Isolation



Install Docker Desktop on Windows

How to Build & Run a Docker Image?

Dockerfile

```
FROM tensorflow/serving:latest

COPY . /models
ENV MODEL_NAME=fashion-mnist
```

Pull TF Serving image

Copy the current directory

Define the MODEL_NAME

Run the Docker image

```
docker build -t
fashion-mnist-tf-serving .

docker run -p 8080:8501
fashion-mnist-tf-serving
```

How to Make Prediction Request?

This example demonstrates how to make prediction requests.

```
import json
import requests
json_data = json.dumps({"instances": image.tolist()})

endpoint = "http://localhost:8080/v1/models/fashion-mnist:predict"

response = requests.post(endpoint, data=json_data)
prediction = tf.argmax(response.json()[“predictions”][0]).numpy()
print(prediction)
```

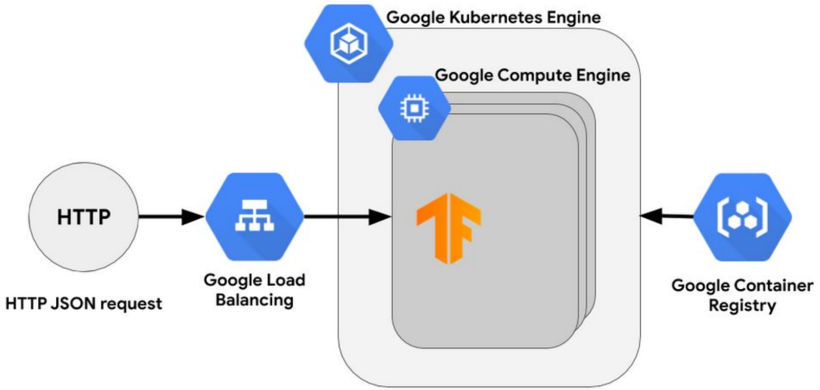
How to Serve Your ML Models on GCP?

Google Cloud Platform (GCP) provides multiple ways for deploying inference in the cloud

- Compute Engine
- Vertex AI
- Cloud Functions
- Cloud Run

Scaling ML System with Kubernetes

- In a production setting, you want to be able to scale as the load is increasing on your app
- Kubernetes can help in orchestrating and scaling multiple docker containers



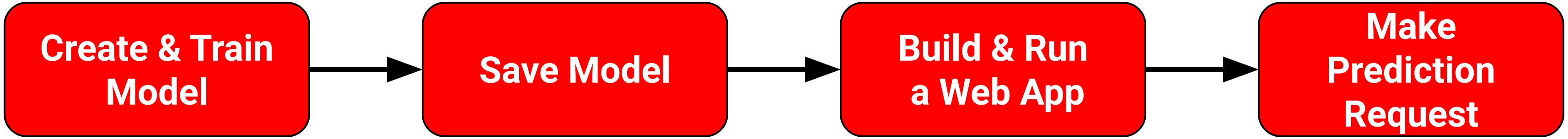
ML Deployment
Flask

What is Flask?

Flask is a minimalist web application framework written in Python

- Easy to use
- Lightweight

Deploy Model with Flask



How to Build & Run a Web App?

A Simple Flask Web App (main.py)

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello_world():
    return "Hello, World!"
```

Define the web application

Define the route

Run the Web App

```
export FLASK_APP=main.py
flask run
```

How to Make Prediction Request?

This example demonstrates how to make a prediction route using Flask.

```
model = joblib.load("iris_model.joblib")
@app.route("/predict", methods=["POST"])
def predict():
    request_json = request.json
    prediction = model.predict(request_json.get("data"))
    prediction_string = [str(d) for d in prediction]
    response_json = {
        "data": request_json.get("data"),
        "prediction": list(prediction_string)
    }
    return json.dumps(response_json)
```

How to Make Prediction Request?

This example demonstrates how to make a request prediction

```
import requests
import json


json_data = json.dumps({"data": [[4.9, 3.0, 1.4, 0.2]]})
endpoint = "http://localhost:5000/predict"
headers = {"content-type": "application/json"}
response = requests.post(endpoint, data=json_data, headers=headers)
print(response.json())
```

Data Pipelines

What is Data Pipeline?

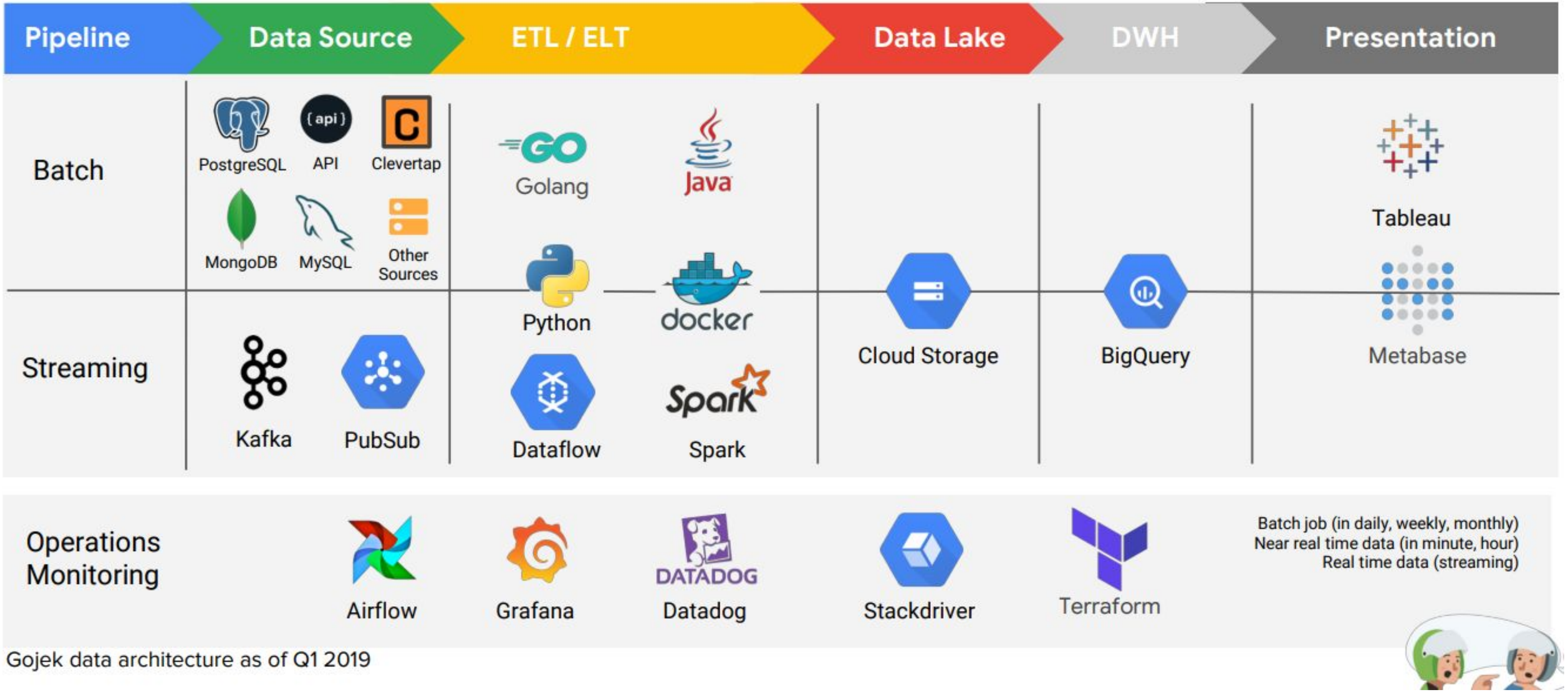
TensorFlow Datasets

- TensorFlow datasets (TFDS) is a tool provided by TensorFlow to build the ETL process with a consistent API
- TFDS also contains a collection of public research datasets of various types such as audio, text, image, video, etc

Load

```
For data in dataset.take(10):
…
```

Gojek's Data Warehouse Architecture



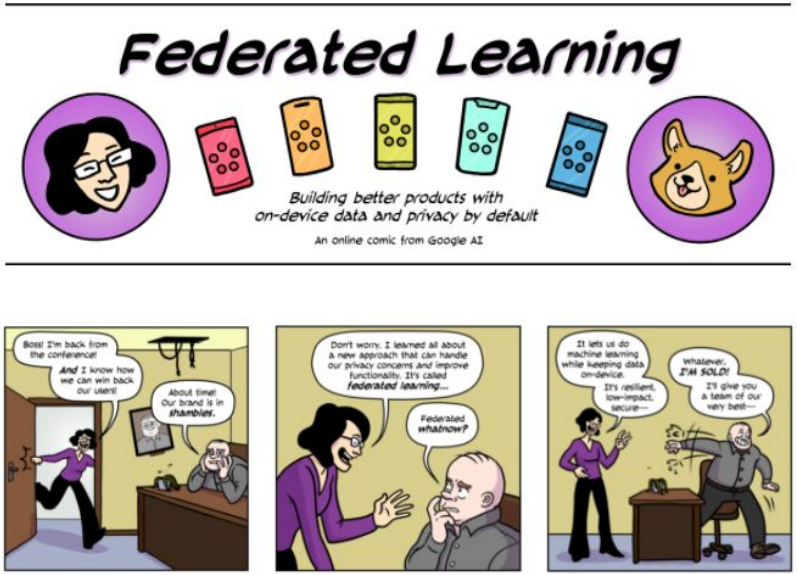
Gojek data architecture as of Q1 2019

Federated Learning

What is Federated Learning?

Federated learning allows each client independently train its own model using its own data right on the device

- Lower latency
- Less power consumption
- Ensuring privacy



* Fun comic that explains the concept

How Federated Learning Works?

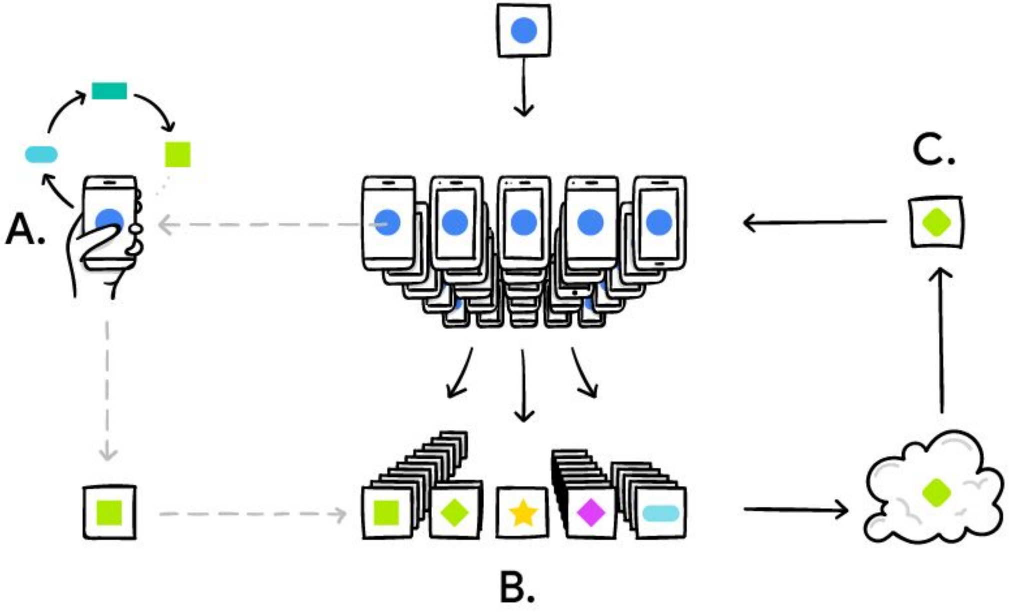


Image source

TensorFlow Federated

TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data

- Federated Learning (FL) API
- Federated Core (FC) API

```
import tensorflow_federated as tff
import nest_asyncio
nest_asyncio.apply()

@tff.federated_computation
def hello_world():
  return "Hello, World!"

hello_world()
```

Deployment Option Summary

	TF JS	TF Lite	TF Serving	TFF
Model runtime	Node.JS server / client's browser	On-device	Server	On-device
Computing power	Depends on usage	Low	High	Low
Latency	Depends on the model complexity	Low	Depends on the infrastructure and model complexity	Low
Model complexity	Depends on usage	Lighter	Heavier	Lighter
Need server connection	Anytime	One time only / once in a while update	Anytime	One time only / once in a while update
Privacy	Depends on model runtime	No need to send data to the server	Need to send data to the server	Ensuring user privacy

Sharing Session

Demo Link

Demo deployment use TF Serving:
https://github.com/dicodingacademy/demo-ilt-ml-bangkit/tree/main/ILT-5/deploy-tf-serving

Demo deployment use Flask:
https://github.com/dicodingacademy/demo-ilt-ml-bangkit/tree/main/ILT-5/deploy-flask

Demo deployment use TFDS:
https://colab.research.google.com/drive/1PnOrYjooTPAa9N4bmejpAxdlLjqZJsGG?usp=sharing

Quiz

Discussions

Thank You