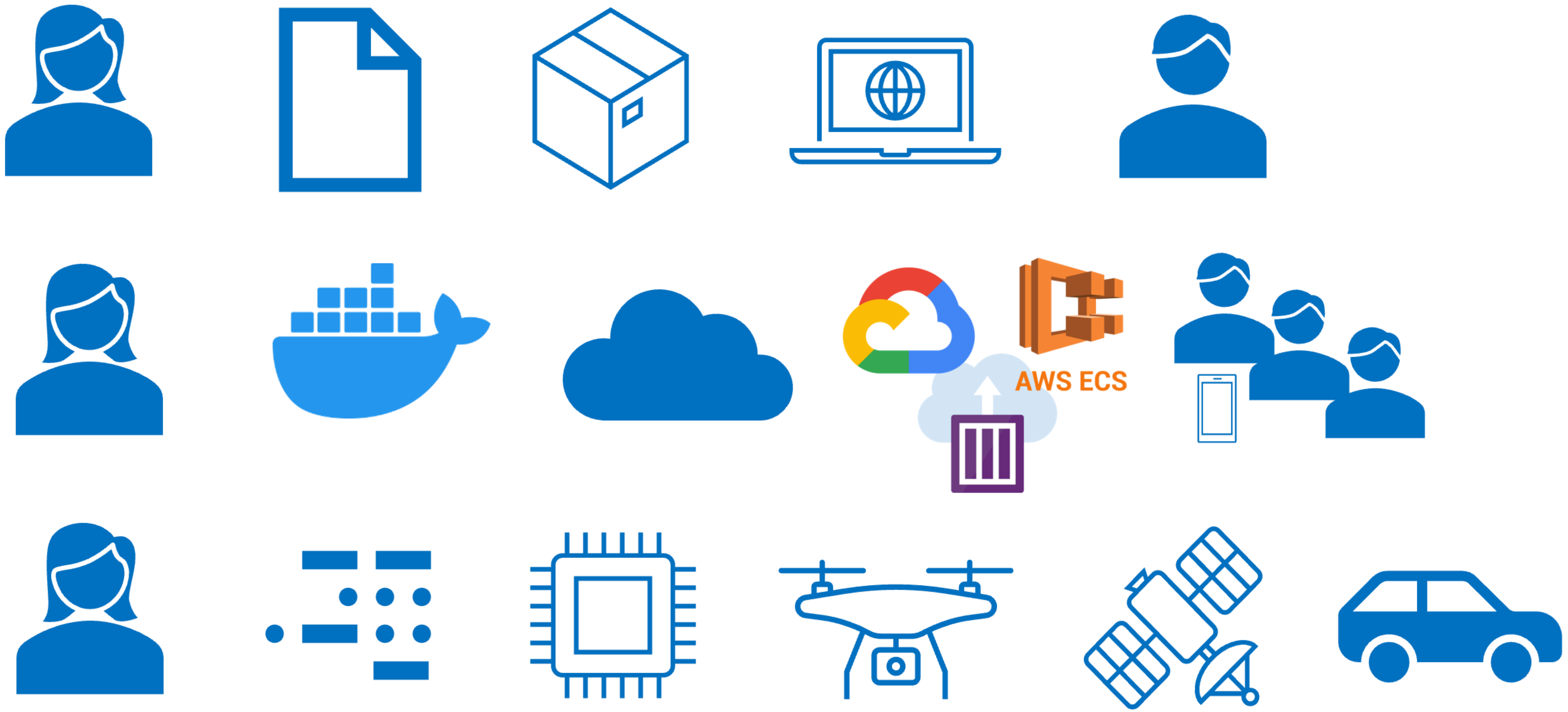
Machine Learning Model Deployment & App Development with Streamlit

ANTHONY ONOJA, PHD

DataEdge Academy

YouTube @tonyonoja7880

Deploying Machine Learning pipeline

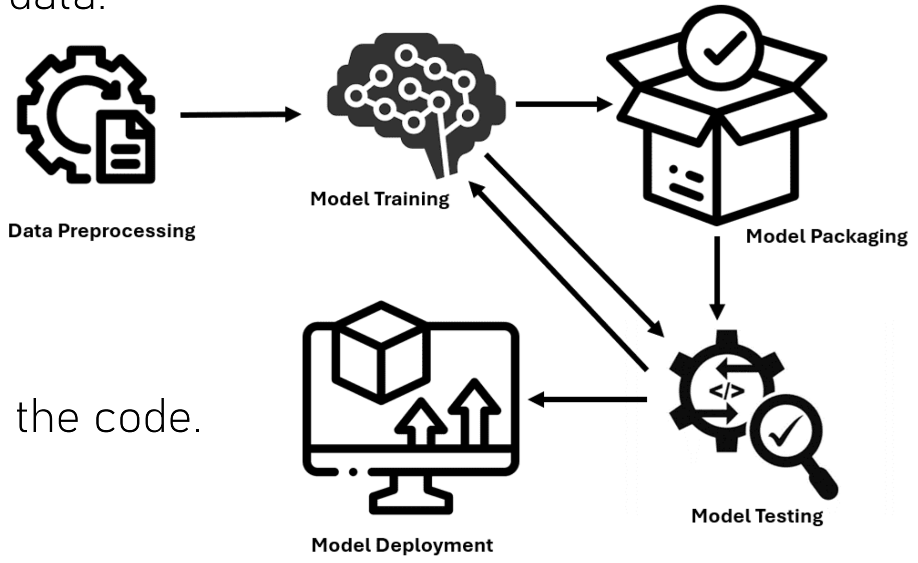


Credit: *https://apmonitor.com/pds/index.php/Main/DeployMachineLearning*

What is Model Deployment?

Model deployment is the process of integrating a trained machine learning model into a production environment where it can make predictions based on real-world data.

Why deploy models?

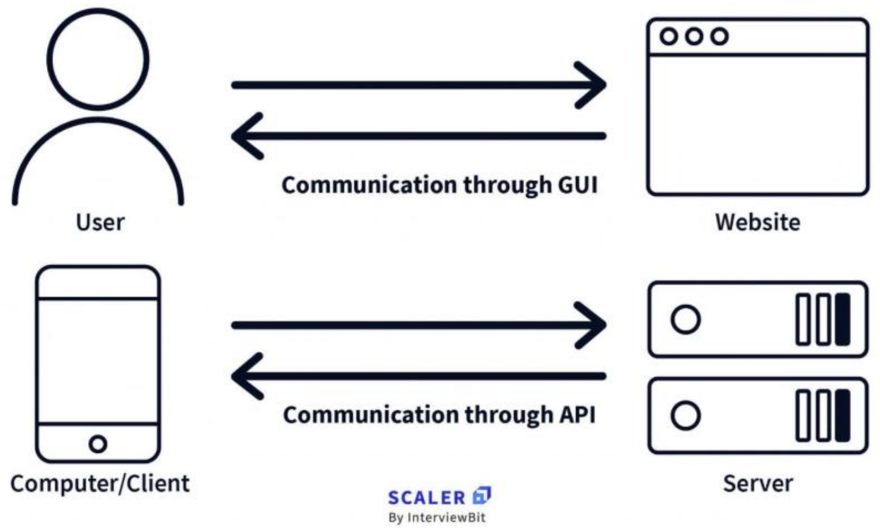


- ❑ To allow stakeholders to use the model without understanding the code.
- ❑ To move from experimentation to decision-making.
- ❑ To build user-facing tools (e.g., dashboards, recommendation systems, chatbots).

Common deployment platforms

Deploying ML is the process of making the model available to produce results for people or computers to access the model predictions. It involves managing data flow to the application for training or prediction. Data flow can be in batches or a continuous stream.

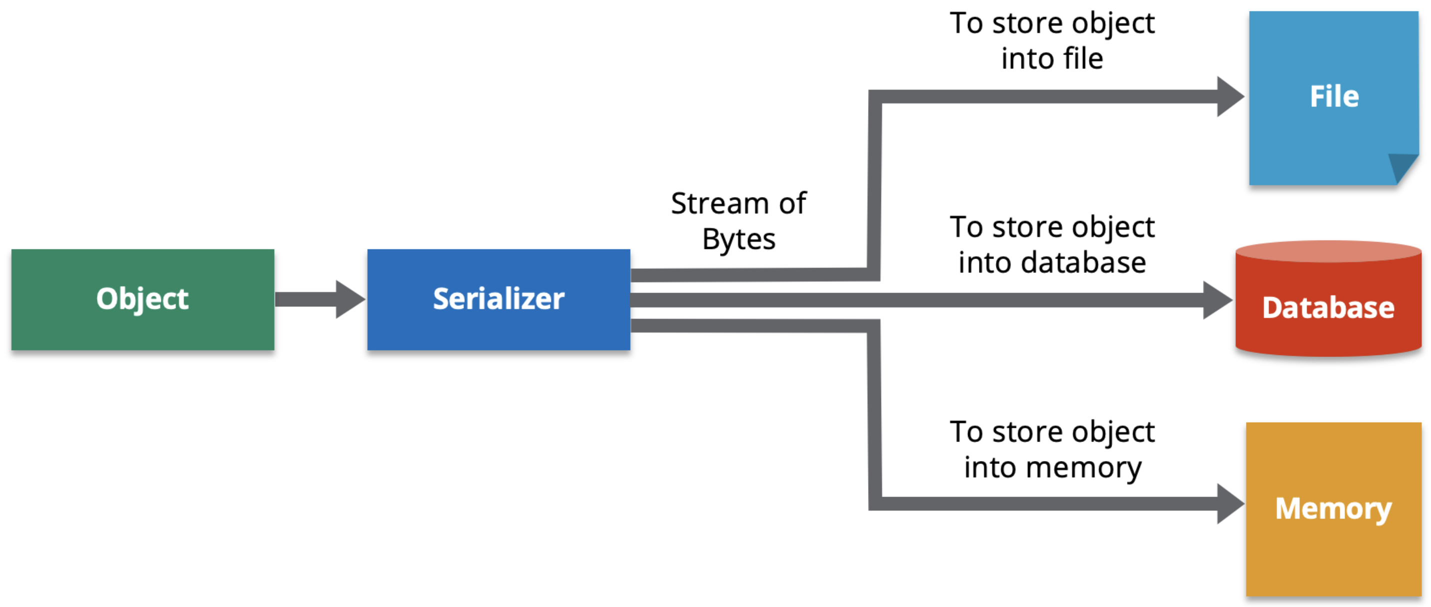
- ❑ Web frameworks (Flask, FastAPI)
- ❑ Streamlit for rapid prototyping & data apps
- ❑ Docker containers + cloud (AWS, GCP, Azure)
- ❑ Application Programming Interface (APIs)



An API is a connection between computers or between computer programs.

Model Serialisation & Containerisation

When deploying ML models, you need to save your trained models in a way that allows them to be loaded and used later, even outside your Jupyter notebook.

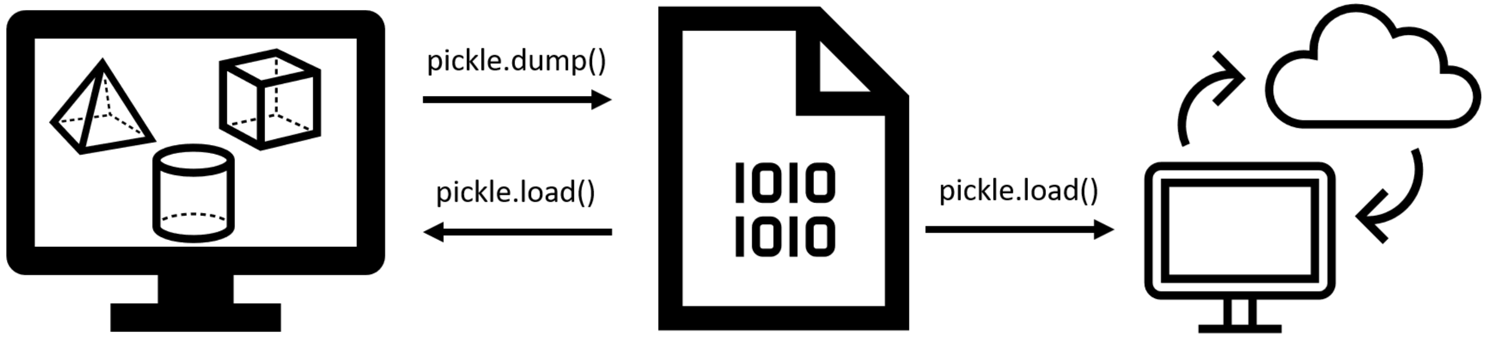

Object
Serializer
Stream of Bytes
To store object into file
File
To store object into database
Database
To store object into memory
Memory

- ✓ Pickle
- ✓ Joblib
- ✓ HDF5 (Hierarchical Data Format)
- ✓ Docker

If the ML is trained offline, the model is encapsulated and transferred to the hosting target. A scalable deployment means that the computing architecture can handle increased demand such as through a cloud computing host with on-demand scale-up. A Docker container can be deployed to cloud services. Google Cloud, Azure Container Instances (ACI), Amazon Elastic Container Service (ECS).

How to Pickle Files

Standard Python module for serialising (pickling) Python objects.



https://apmonitor.com/pds/index.php/Main/DeployMachineLearning

- ❑ Saves entire model pipelines, preprocessors, etc.
- ❑ Not optimised for large NumPy arrays.

Note: You need to serialise general Python objects or small models.

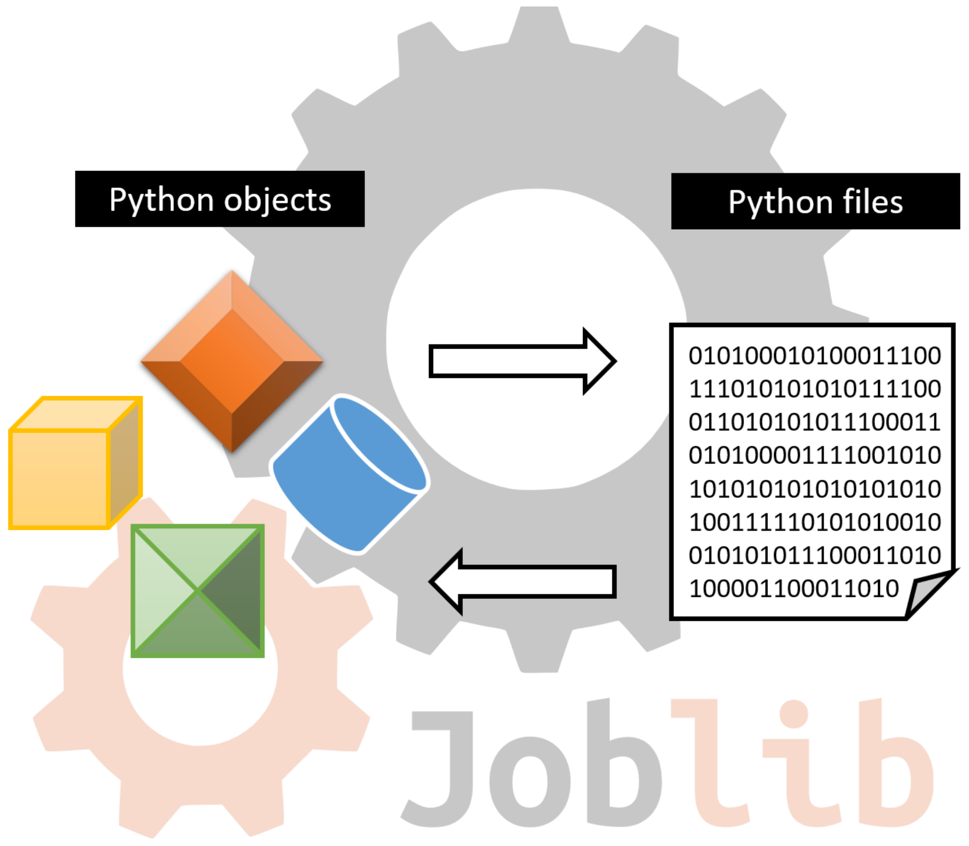
<u>Python code</u>

```
import pickle

# Save
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)

# Load
with open("model.pkl", "rb") as f:
    model = pickle.load(f)
```

Joblib serialisation



- ✓ Built for **efficient storage of large numpy arrays**.
- ✓ Often faster and smaller than pickle for scikit-learn models.
- ✓ Note: You're saving large models or using scikit-learn.

Python code

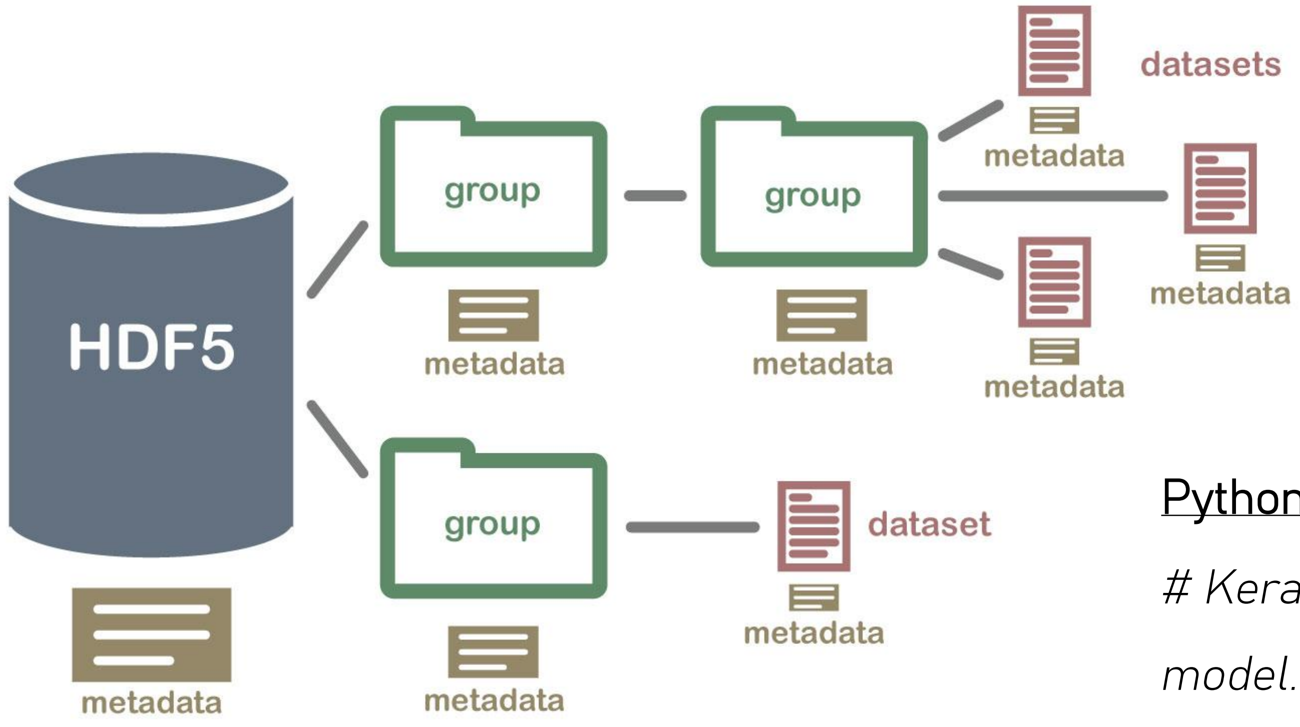
```
import joblib

# Save
joblib.dump(model, "model.joblib")

# Load
model = joblib.load("model.joblib")
```

HDF5 (Hierarchical Data Format)

A binary file format for storing large datasets (used by TensorFlow/Keras).



Note: You're working with deep learning models (TensorFlow/Keras).

Python code

```
# Keras example
model.save("model.h5")         # Save
model = keras.models.load_model("model.h5")  # Load
```

Stores model architecture, weights, optimiser state.

Containerising with Docker

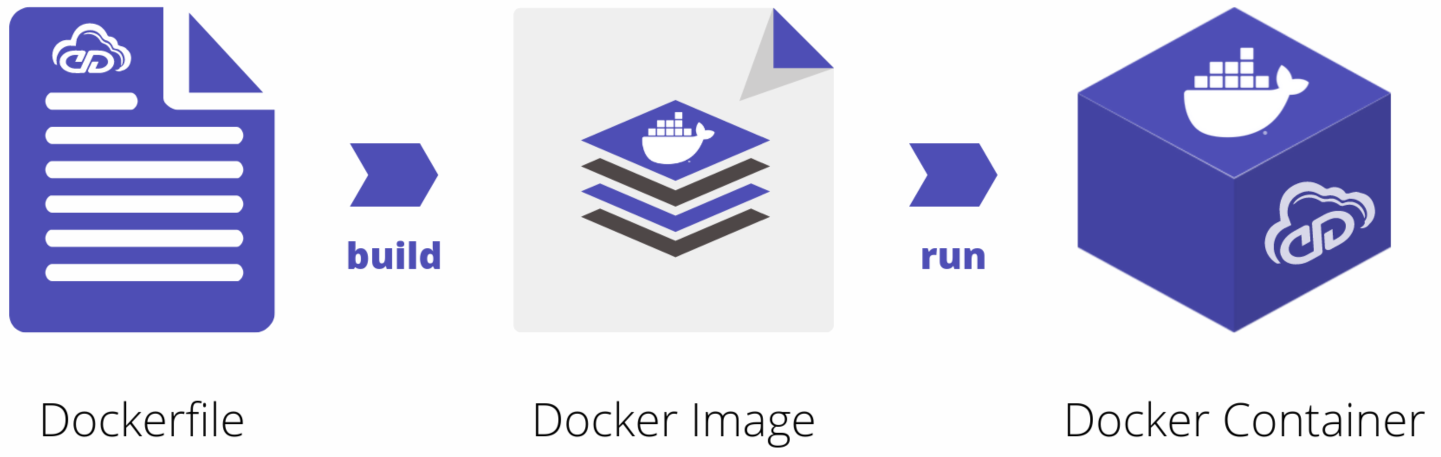
When deploying to production, we want repeatable environments. That's where Docker comes in.

What is Docker?

A containerization tool that packages:

- ✓ Code
- ✓ Dependencies
- ✓ System libraries

Into one isolated environment.



Why Use Docker?

- ✓ Consistent across machines
- ✓ Easy deployment to cloud (e.g., AWS, GCP, Azure)
- ✓ Simplifies collaboration
- ✓ Secure, portable, and scalable

Introduction to Streamlit

Streamlit is a Python framework designed to build interactive data apps quickly.

Why Streamlit?

- ✓ Minimal learning curve.
- ✓ Integrates with data science stacks: Pandas, scikit-learn, TensorFlow, PyTorch.
- ✓ Great for demo apps, dashboards, model interfaces.
- ✓ Hosted easily on Streamlit Cloud or Hugging Face Spaces.

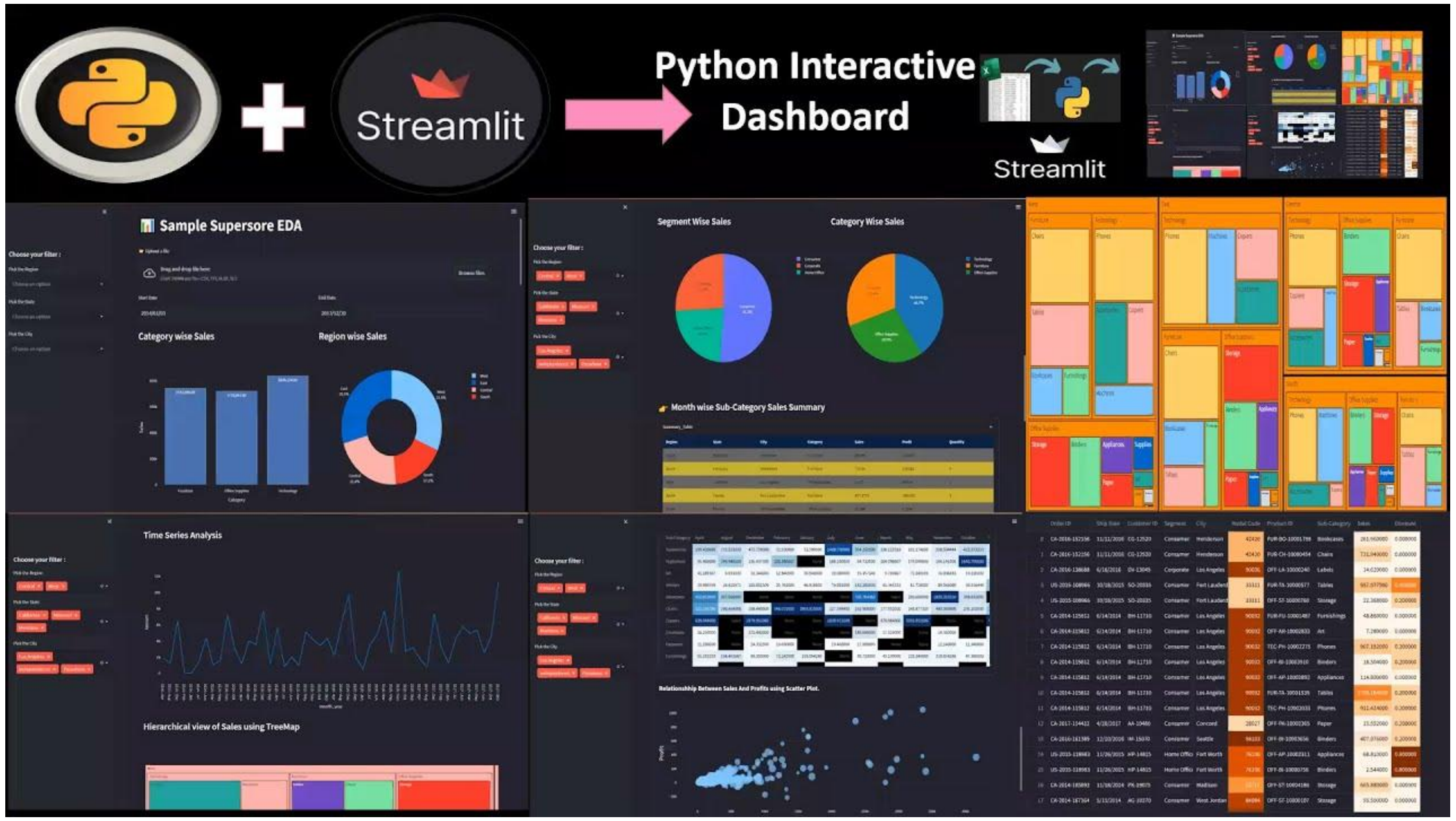
Basic Streamlit commands:

```
import streamlit as st

st.title("My First Data App")
st.write("Welcome to model deployment!")
```

Adding Features to Your App

- ✓ File upload (st.file_uploader)
- ✓ Charts (st.line_chart, st.bar_chart)
- ✓ Sidebar navigation
- ✓ Page structure with multipage apps


Streamlit
Python Interactive Dashboard
Streamlit
Sample Supersore EDA
Category wise Sales
Region wise Sales
Segment Wise Sales
Category Wise Sales
Month wise Sub-Category Sales Summary
Time Series Analysis
Hierarchical view of Sales using TreeMap
Relationship Between Sales And Profits using Scatter Plot.

Deployment Options

a) Streamlit Community Cloud (free)

- ✓ Push app to GitHub.
- ✓ Go to https://share.streamlit.io
- ✓ Link your repo → Deploy.

b) Hugging Face Spaces (for public apps)

- ✓ Use Streamlit as the runtime.
- ✓ Add requirements.txt, app.py, and optionally README.md.

What is GitHub?

GitHub is a platform for hosting and sharing code using Git, a version control system.

It helps you:

- ✓ Track changes in your code
- ✓ Collaborate with others
- ✓ Publish your projects
- ✓ Deploy apps (like Streamlit) directly from GitHub

Why Should Data Scientists Use GitHub?

Benefit	Description
✅ Version Control	Track every change you make in your code
✅ Collaboration	Work with team members without overwriting each other's work.
✅ Backup	Keep your work safe in the cloud
✅ Portfolio	Show off your data science projects to employers
✅ Deployment	Connect GitHub to services like Streamlit Cloud, Hugging Face, Heroku.

Basic Git & GitHub Terms

Term	What it Means
Repo	A project or folder in GitHub
Commit	A saved version of your code
Push	Upload your changes to GitHub
Clone	Copy a GitHub repo to your local machine
Branch	A separate line of development
Merge	Combine changes from different branches

Step-by-step: Create and push a Streamlit app

Create GitHub repo:

- ✓ Go to https://github.com
- ✓ Click New Repository
- ✓ Name it, e.g., iris-streamlit-app

Set up Git locally (only once):

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

Create a project folder locally:

```
mkdir iris-streamlit-app
cd iris-streamlit-app
```

Initialize Git and connect to GitHub:

```
git init
git remote add origin
https://github.com/yourusername/iris-streamlit-app.git
```

Add files and push:

```
git add .
git commit -m "Initial commit"
git push -u origin main
```

Best Practices for Deployment

- ✓ Use requirements.txt for dependency management.
- ✓ Handle exceptions and invalid inputs.
- ✓ Avoid retraining model in real time — load pre-trained model.
- ✓ Keep apps modular and readable.
- ✓ Secure sensitive info (e.g., API keys via st.secrets).

Deploy from GitHub to Streamlit Cloud

1. Go to https://share.streamlit.io
2. Connect your GitHub account
3. Select your repo and the app file (e.g., app.py)
4. Click Deploy

Hand-on session:Use Cases of Streamlit in Industry

- ❑ Explainable AI dashboards
- ❑ Fraud detection portals
- ❑ Health diagnostic apps
- ❑ Customer segmentation visualisation
- ❑ Real-time social media sentiment tracking