Data Engineering and MLOps in Business

Serverless User Interfaces for Machine Learning Systems

Primoz Konda

AAUBS

March 26, 2024

pk@business.aau.dk

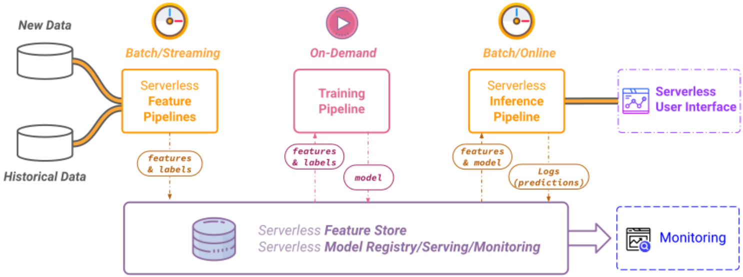
Plan for today

1. Serverless UIs for ML
2. Docker
3. Docker Concepts

Where are we now in MLOps journey?

- Technical issues?
- We will focus on Docker and our Penguins
- Homework: Get Hopswork Lab 4 working :)
- Recreate this app with your data

ML & Data Pipeline



Why we need UI solutions

- Translating model performance from technical to non-technical audiences.
- Demonstrating model value through ”decision intelligence.”
- Showcasing incremental progress and receiving feedback.
- Importance of understanding the business problem and its needs.

UI-Driven Development for Data Science

- Building a user-interface from day 1 to align business needs with ML model results.
- Following an iterative development process with incremental improvements.

Benefits of Serverless UIs for ML

- Scalability and Flexibility
- Reduced Infrastructure Overhead
- Enhanced Developer Productivity

Challenges and Considerations

- Cold start problem.
- Limited control over the environment.
- Security and privacy concerns.

UI Examples

- Streamlit
- Dash
- Gradio
- Bokeh (interactive data vizualization)
- Django (google, youtube, instagram...)
- Jupytr, Pluto, Shiny

UI Examples

Sometimes, the harder question is, where is the appropriate place to host it?

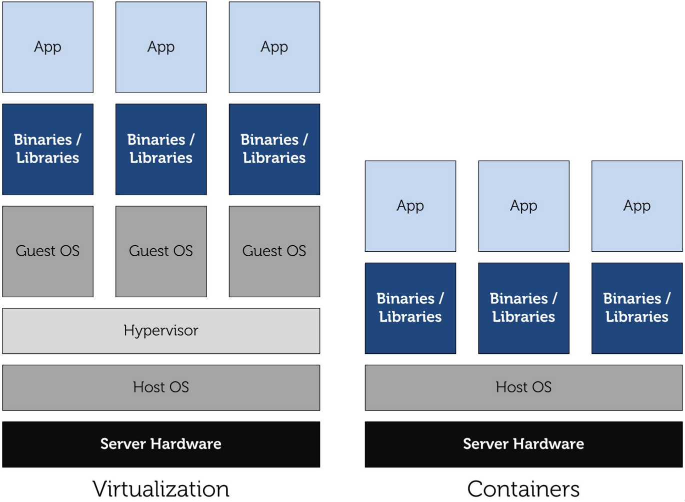
Issues:

- Privacy
- Security
- Costs
- Stability

What is Docker?

- Docker is an open-source platform that automates the deployment of applications inside lightweight containers.
- Allows applications to run in the same manner on any system that supports Docker.
- Simplifies configuration, increases reproducibility, and eases isolation of dependencies.

Virtual Machines VS Containers



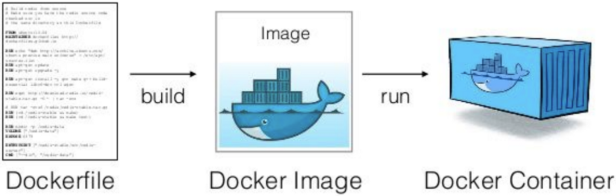
Why Docker?

- **Consistency:** Ensures consistent environments from development through to production.
- **Speed:** Containers can be spun up in seconds, making deployments and scaling faster.
- **Isolation:** Applications and their dependencies are isolated in containers, reducing conflicts.
- **Portability:** Containers can run anywhere - on a developer's laptop, on physical or virtual machines, in data centers, or in the cloud.
- **Microservices:** Facilitates the microservices architecture by allowing each service to be containerized and scaled independently.

Docker in Practice

- **Development:** Developers use Docker to run and test applications in environments identical to production.
- **CI/CD:** Docker simplifies the continuous integration and delivery process by ensuring consistent environments at all stages.
- **Application Deployment:** Easily deploy applications on any platform that supports Docker, ensuring reliability and consistency.

What is Docker?



Dockerfile

- A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image.
- Acts as a blueprint for building Docker images.
- Specifies the base image, software installations, environment variables, network ports, and other configurations needed for the container.
- **Build command:** `docker build -t my-image-name .`

How to structure Dockerfile?

```
# Use the official Python image from the Docker Hub
FROM python:3.8-slim

# Set the working directory inside the container to ’/app’.
WORKDIR /app
COPY core/requirements.txt .
RUN pip install –no-cache-dir -r requirements.txt
COPY app/ .

# Instruct Docker to listen on port 8501 at runtime.
EXPOSE 8501

# Set the command to run the app using Streamlit.
CMD [”streamlit”, ”run”, ”app/streamlit_app.py”, ”–server.port=8501”,
”–server.address=0.0.0.0”]
```

Docker Image

- A **Docker Image** is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and config files.
- Images are built from the instructions in a Dockerfile.
- Images are immutable, meaning once built, they do not change.
- Can be stored in a Docker registry such as Docker Hub, allowing them to be shared and deployed anywhere Docker is supported.
- **Usage:** Images become containers when they run on Docker Engine.

Docker Container

- A **Docker Container** is a runtime instance of a Docker image.
- Containers encapsulate the application and its environment at the runtime.
- Provide isolation from other containers and the host system, yet allow for network and storage connectivity.
- Can be started, stopped, moved, and deleted independently.
- Containers ensure that the software runs uniformly and consistently across any platform.

Q&A

Thank You!
Questions?