Productization - Day 9

Topics

- Different types of deployments
- Deployment Process
- Cloud Infrastructure
- Deployment Considerations
- Local Machine Setup

Goals

- Distinguish between the different types of model deployments
- Comprehend technical requirements for deploying models
- Become familiar with the deployment process
- Understand potential deployment problems and optimize against them
- Set-up your local machine to execute Python code

Types of Deployments

Thinking about how we interact w/ AI

Think back to our first lesson on AI in our everyday lives.

- What kinds of models are we interacting with?
- How are we interacting with these models?
  - Are we actually sending data to a model?
- Where are these models hosted?

Interactive vs Non-Interactive

Interactive

- Client is sending data to the model and receiving results

Non-Interactive

- Client is not sending data to the model

- Majority of the time we are actually using non-interactive deployments, in our daily lives.
- Data is collected about you, then models optimize for your attention.
- You are not sending data directly to a model

Batch vs Single Record

Batch Deployment:

- Model is receiving batches of data and returning multiple results

Single Record:

- Model is receiving and returning a single-record at a time.

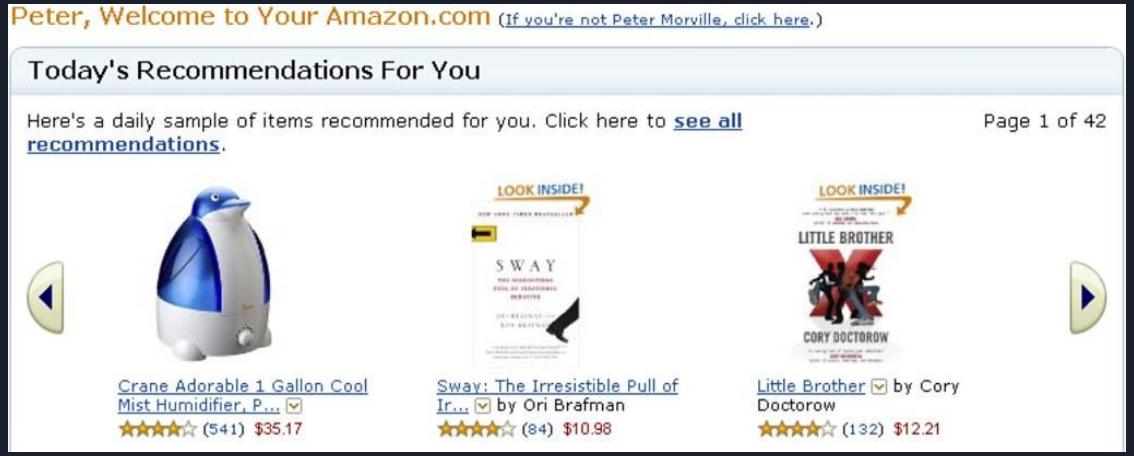
- There isn't much of a difference here; how many rows in our input matrix to our model 'predict' method is all.
- All of these are important when considering how a model will be used in production

Examples

Example request | text-davinci-002 | python | Copy

```
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Completion.create(
  model="text-davinci-002",
  prompt="Say this is a test",
  max_tokens=6,
  temperature=0
)
```

- OpenAI GPT3
- Interactive
- Single-record
- Making direct request to model



- Amazon recommender
- Non-interactive
- Batch
- Model will show you data upon page loading

More Examples

- Interactive
- Single record

- Embedded systems
- non-interactive

Process - Infrastructure

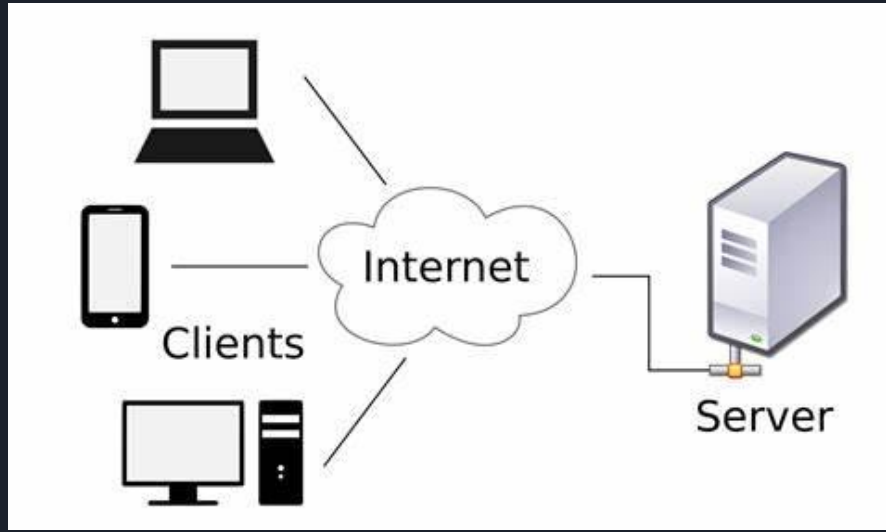
Components of ML Deployments

1. The ML model - model we want to deploy
2. API (flask, fastapi, etc) - api to make calls to the model
3. Hosting (Heroku, GCP, Azure) - cloud service provider
4. Infrastructure (VMs/worker jobs/database) - cloud application infrastructure

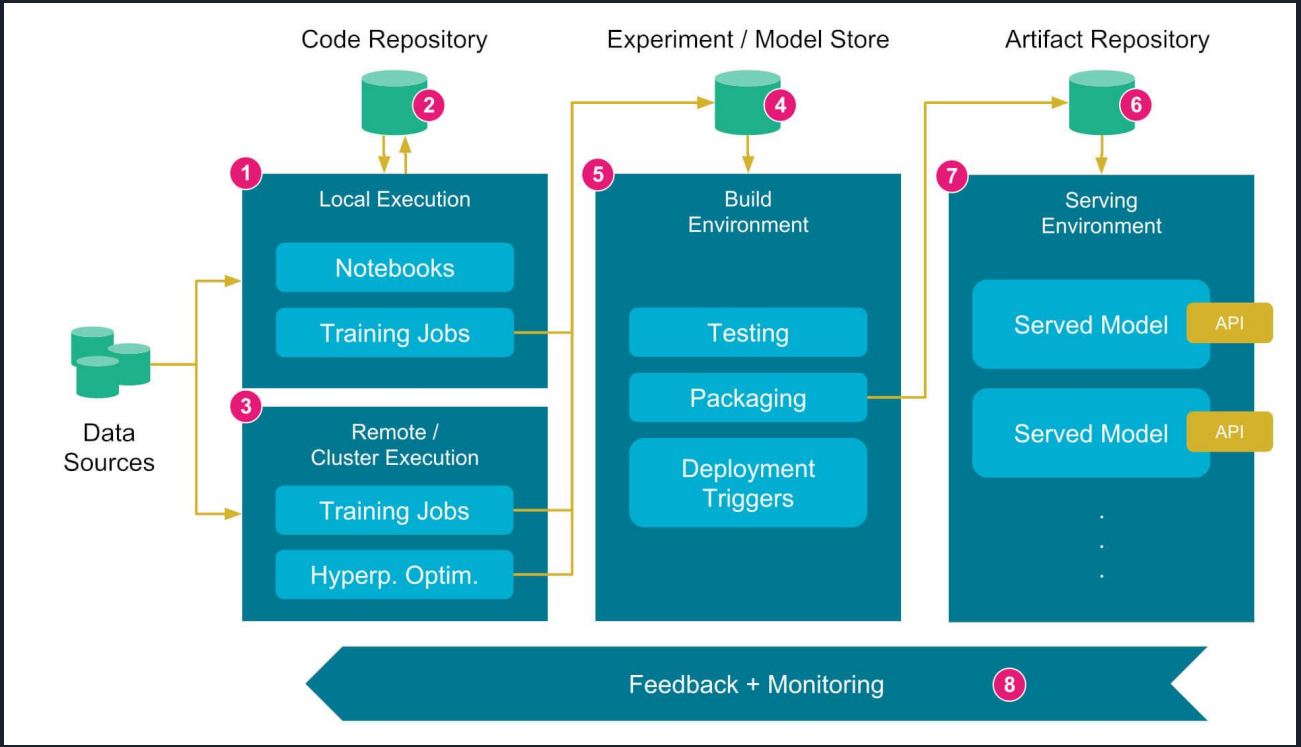
In that order!

Clients & Servers

Client	***Server***
• What the user interacts with • GUI (graphical user interface) • Application, has buttons boxes, design, etc • Client can access/see code • Contains public keys that map to secret keys	• The code that carries out the request made by the client • Client never sees/accesses server-side code • Contains secret keys to run application



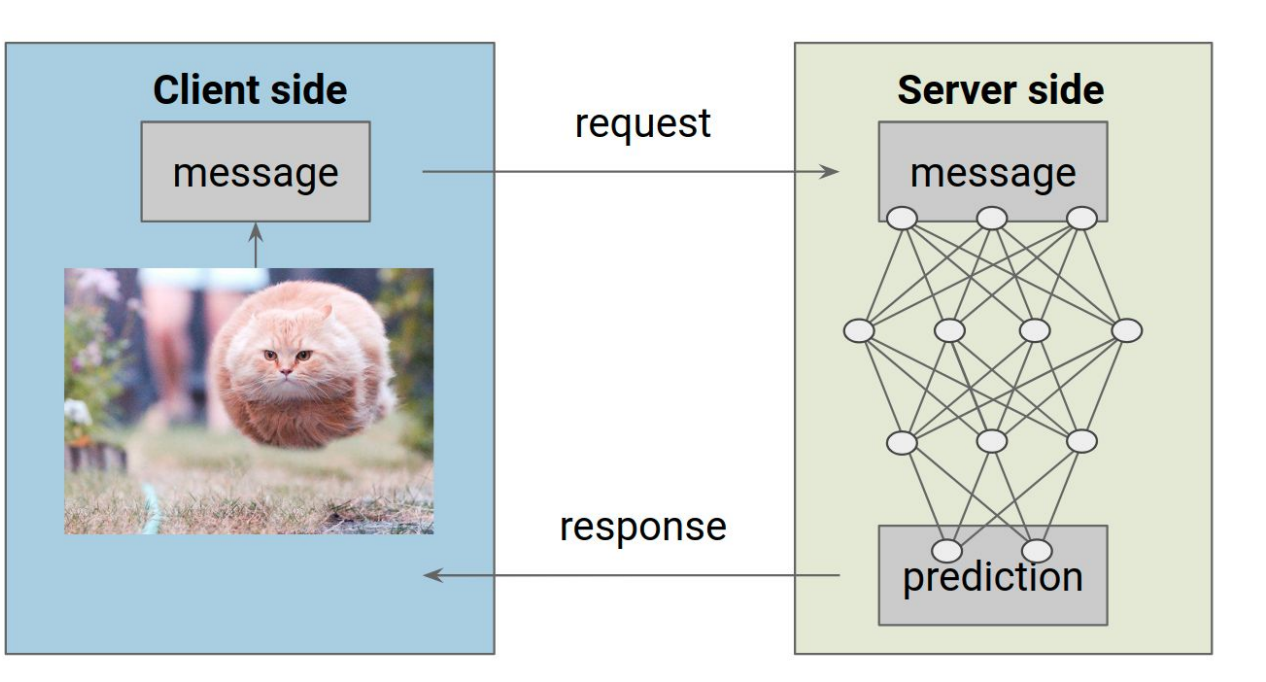
Development Process



- So far we have accomplished 1,2,3.
- We have created notebooks/models and completed training jobs.

- We will now move on to the following parts of the process

Deployment Overview



- Example interaction with production model
- A request is initiated by the client, sent to the server, fed to the ml model, and the response returned to the UI.

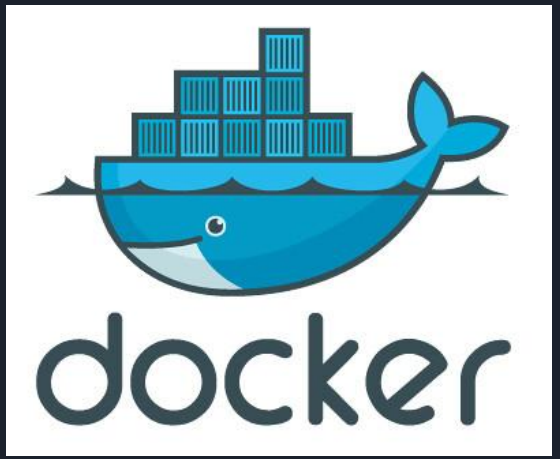
Infrastructure - Overview

- In development, as we will see shortly, we deploy our apps locally. We can do all dev/testing in a local environment.
- However, in a production environment, our code will not run locally. We want to publish it to the web so anyone can access & use it.
- All an app does is execute some code, based on what the user wants to do.
- ***Q:*** Code is executed sequentially, so how would one deploy an app so that many users can send requests simultaneously?
- ***A:*** VMs (virtual machines), & worker jobs (or queues).

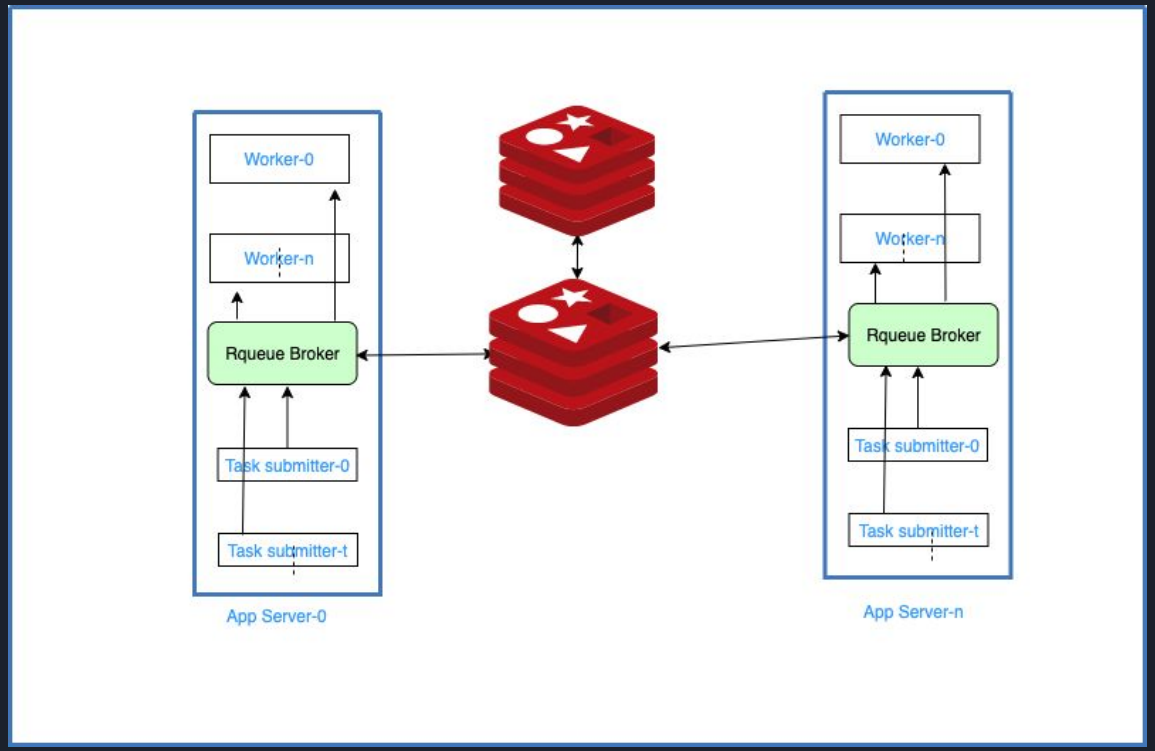
Redis, Docker, and Kubernetes

- The inner-workings of these services are quite complex.
- For now, just know these tools are how we manage application dependencies and deploy ML models to the web.
- Conceptualize these services as “cloud” computers, with individual configurations.
- This will make more sense after setting up an environment locally.

redis



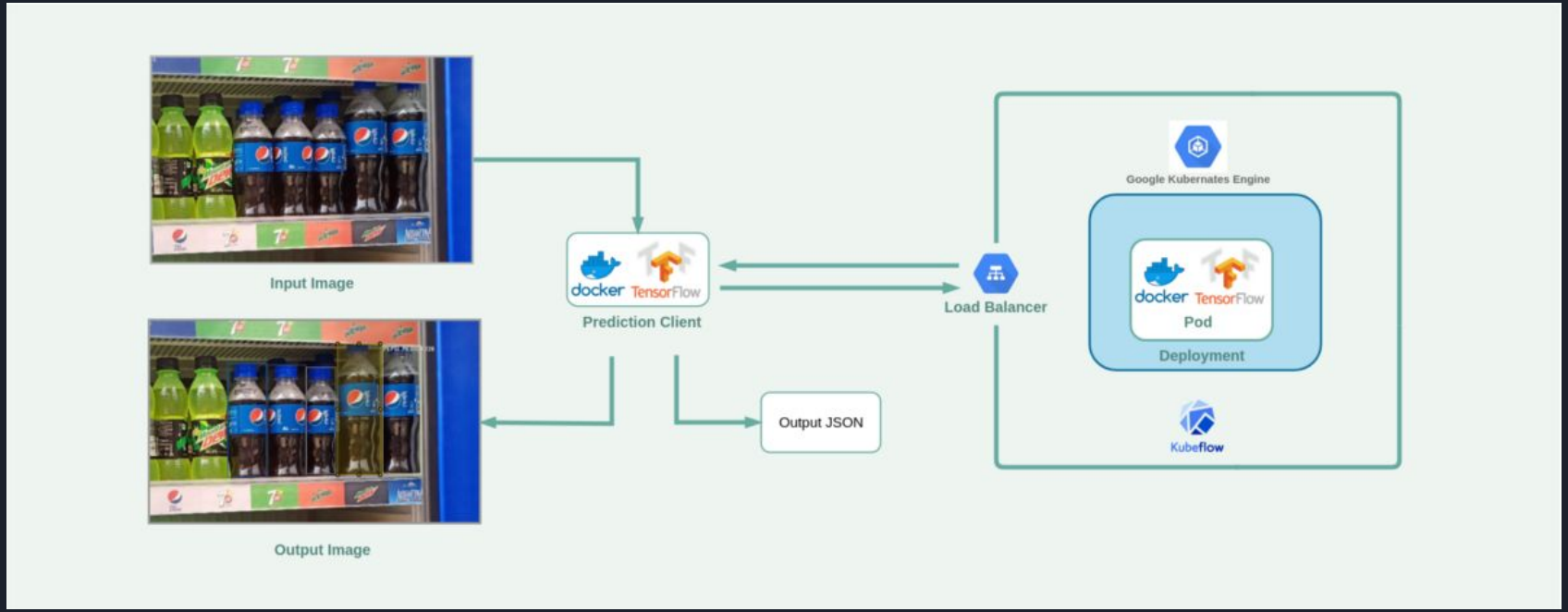
Infrastructure - Worker Jobs



- We use VMs (virtual machines) to execute our code, & carry out the requests sent to the application.
  - This example illustrates Redis usage.
  - We need a queue, to process requests simultaneously
  - This ensures seamless concurrent use of our application

Deployment Example

A request is being sent to the docker VM configuration running this application, and returning the model results. As we can see, the request contains an image, and we are sending this to a keras ANN.



Production Considerations

Infrastructure Considerations

As we've already seen in previous slides, there is a vast amount of options to choose from when deploying ML applications.

- Will this need a GPU VM, or will CPU suffice?
- How many VMs will we need?
  - What is the expected traffic for this application or API?
- How will the data be fed to the model?
- Do we need a database? If so, what are the security protocols? (it is task specific)
  - What kind of database?
  - What data are we storing?

Model Drift

A common misconception about AI is that models continuously learn. However, this is not true, a model learns when it is trained.

So, what is model drift?

- Imagine this, a large retailer wants to analyze beverage sales during the winter months.
- They gather all relevant data and train a regression model to predict the upcoming sales for this winter.
- However, the last two years the world was in an unprecedented pandemic, and now that has largely subdued, maybe it is time to analyze the model again.

Model Re-Training

There are a number of things to consider when deciding to retrain a production model.

- Is the model performance decreasing?
- Has the data significantly changed? (model drift)
- Has a significant amount of new data become available?
- Was there a new major advancement in the tech-stack you're utilizing?

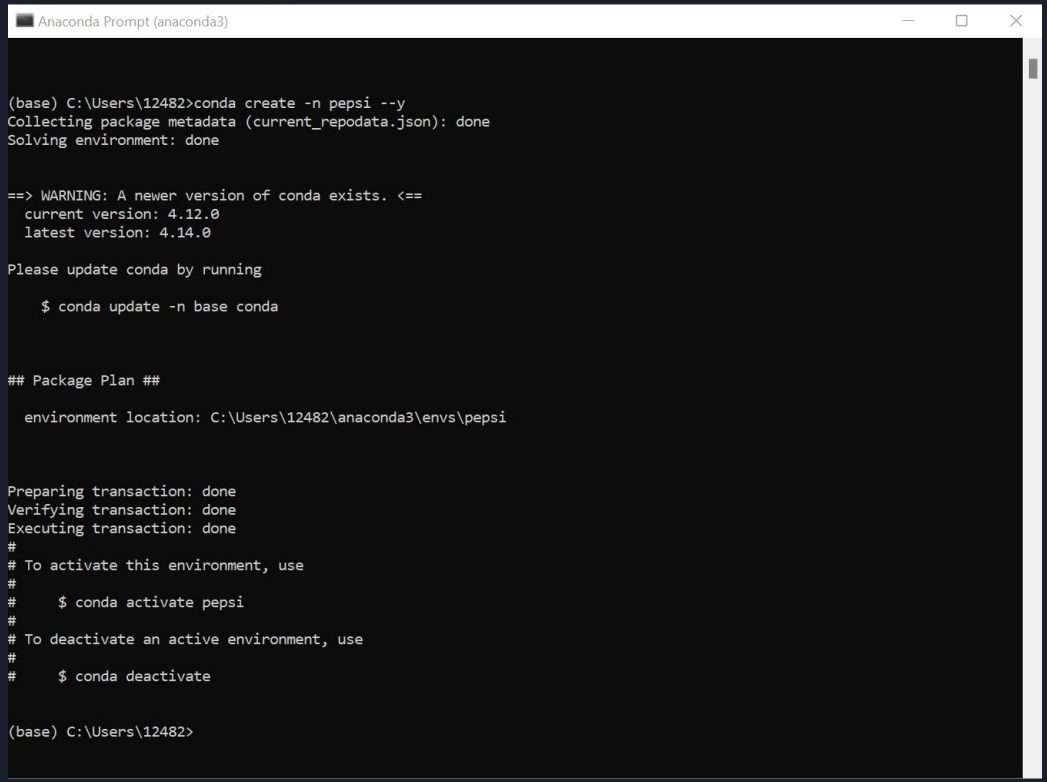
Local Machine Setup

Software

- Anaconda
- Microsoft VS-Code

Setup - 1

1. **Navigate to the ‘Anaconda Prompt’**
   a. Go to computer search bar, and search ‘anaconda prompt’.
2. **Run the command**
   a. >>> conda create -n pepsi - - y



Setup - 2

1. Activate the environment created in step 1
   a. >>> conda activate pepsi
2. Install the txt file I have provided to you. This will install all the dependencies your computer needs to run a ML app locally.
   a. >>> cd desktop
   b. >>> mkdir training-course
   c. >>> cd training-course
   d. >>> pip install -r requirements.txt

Setup - 3

Run some tests to be sure you have properly installed the dependencies

1. >>> conda activate pepsi
2. >>> python
3. >>> import pandas as pd, numpy as np
4. >>> from sklearn.linear_model import LinearRegression
5. >>> from flask import Flask

Conclusion

Please follow along in VS-Code for the remainder of the lesson.