Model serving & Cloud infrastructure

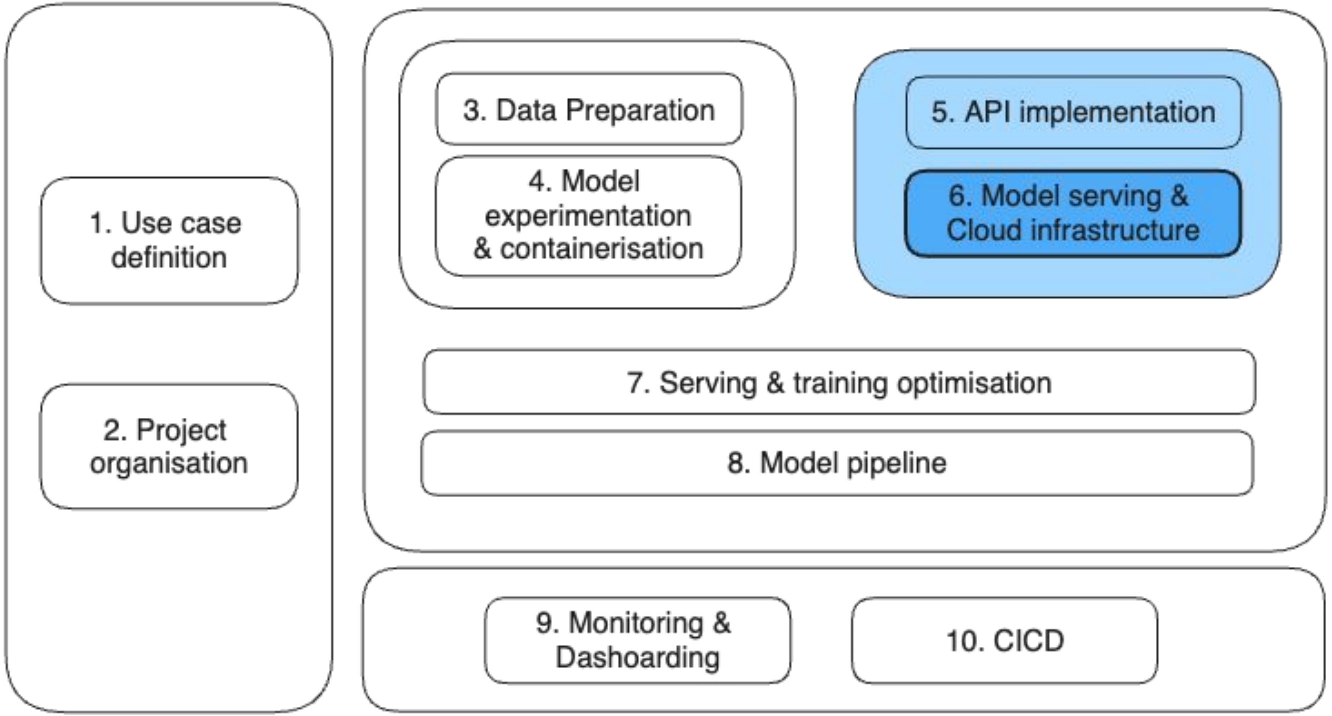
Sprint 3 - Week 6

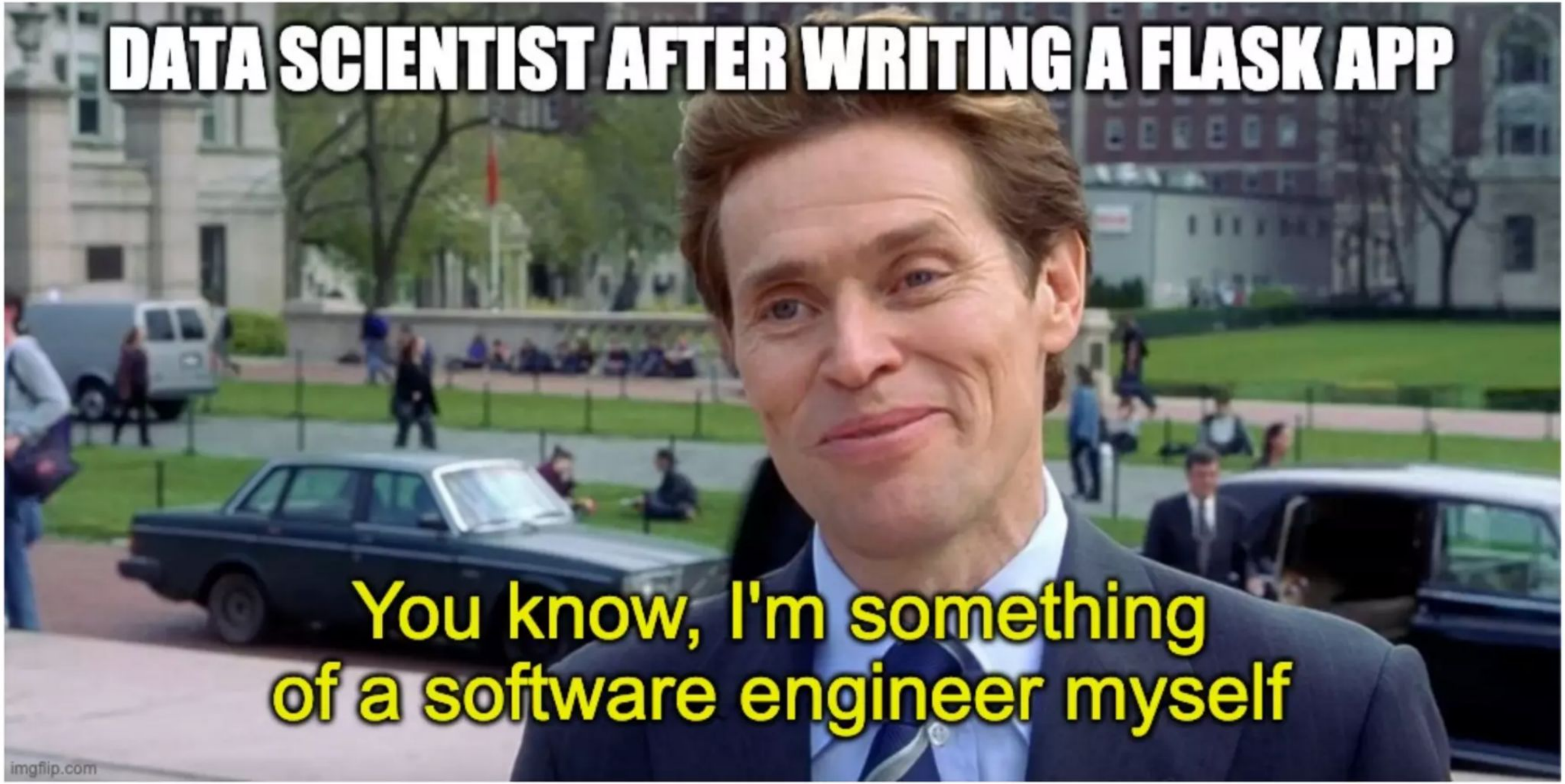
INFO 9023 - Machine Learning Systems Design

2024 H1

Thomas Vrancken (t.vrancken@uliege.be)
Matthias Pirlet (matthias.pirlet@uliege.be)

Status on our overall course roadmap




DATA SCIENTIST AFTER WRITING A FLASK APP
You know, I'm something of a software engineer myself
imgflip.com


DADDY, WHAT ARE CLOUDS MADE OF?
LINUX SERVERS, MOSTLY

Agenda

What will we talk about today

Guest Lecture (45 min)

1. Cloud infrastructure

Lecture (30 min)

2. ML Model serving
3. Cloud vs on-prem deployment

Lab (45 min)

1. Deploy an API in the Cloud

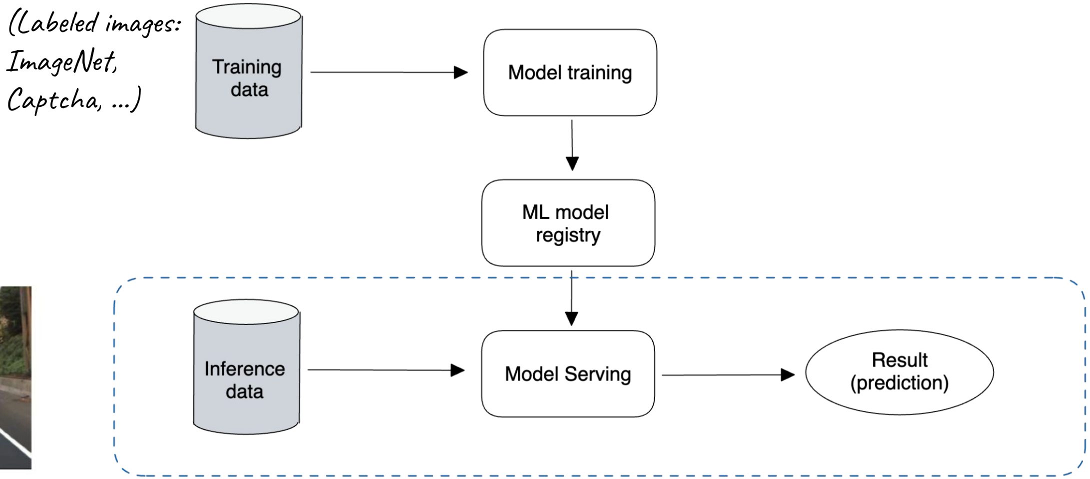
Guest lecture:
Cloud infrastructure

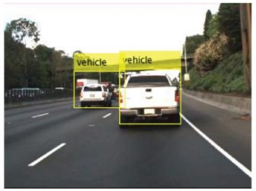
ML Model serving


YOU'RE DONE TRAINING YOUR MODEL?
TIME TO START SERVING IT !
imgflip.com

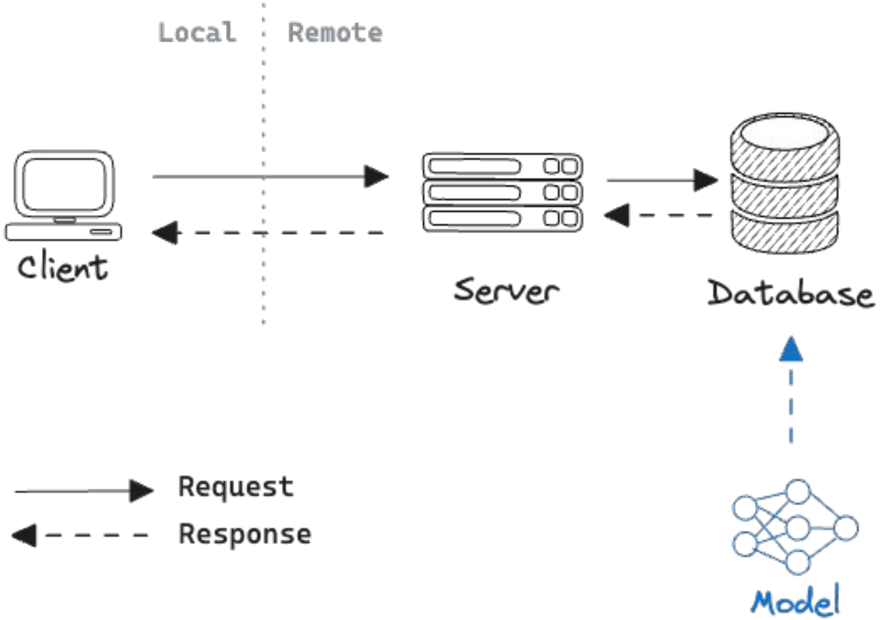
What is **model serving**?

After training an ML model, it can be called on **inference** by users during **model serving**.





Batch predictions



Batch predictions

Offline serving / batch: Model is used in a batch job on a large number of historical data points.

Model is used **periodically** (daily, weekly, ...) on all new or relevant data points.

Usually **less latency** (speed) dependent - optimisation techniques can still be useful for cost/compute reasons.

But requires **more throughput**.

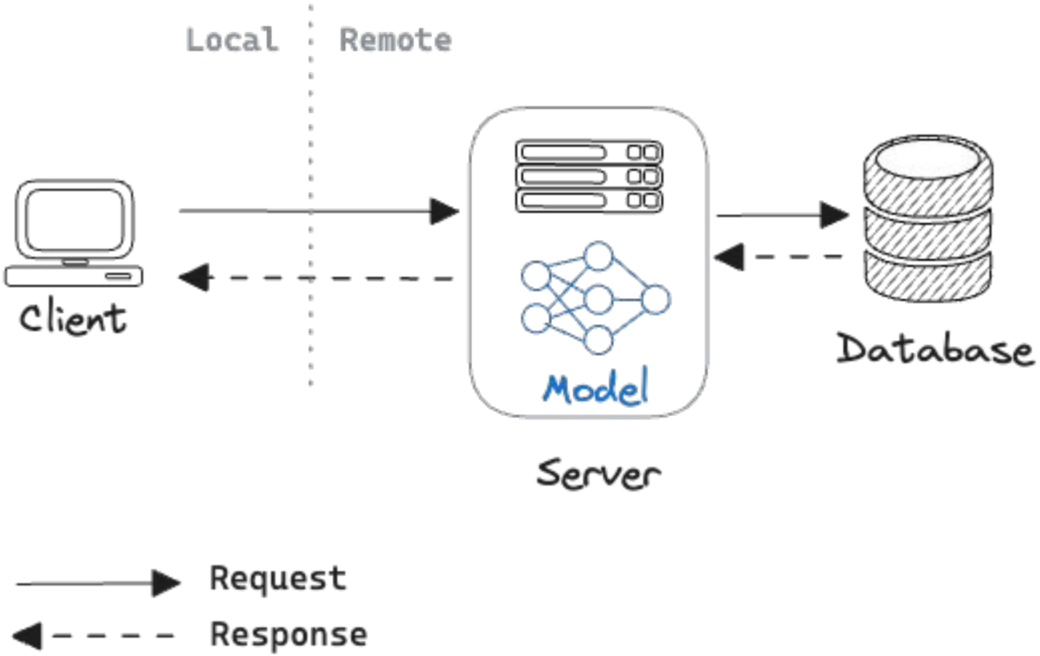
Examples:

- A lot of forecasting models (periodical run when you have new data - daily, weekly, …)
- Document AI on archive
- Sentiment analysis on customer feedback

Batch predictions

Pros	Cons
● Simple to implement ● Low latency to the user ● Less downtime risks	● Can only produce predictions for a fixed list of inputs ● Doesn’t scale to complex input types ● Users don’t get the most up-to-date predictions

Online predictions



Online predictions

Online serving (aka **real-time**): Model is used in a real-time by being called on a single data point and directly returning the result. You need the client's historical and current context.

Requires a model with a low enough latency.

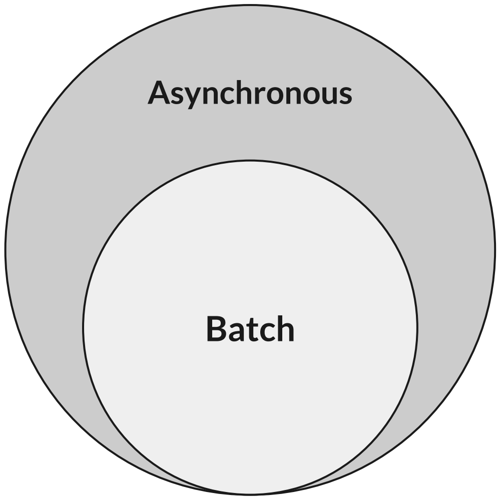
Examples: Detecting car traffic objects in images, ChatGPT, …

Attention: Online ≠ On the Cloud. You can do real-time (online) serving but on prem, so not on the Cloud

Online predictions

Pros	Cons
● Can be used on (pretty much) any new data point ● Latest possible prediction ● No scheduled pipeline or prediction data storage necessary	● Requires low latency model ● Model needs to always be up and running ● Serving infrastructure overhead (GPUs always up, load balancing, …)

Synchronous and asynchronous.

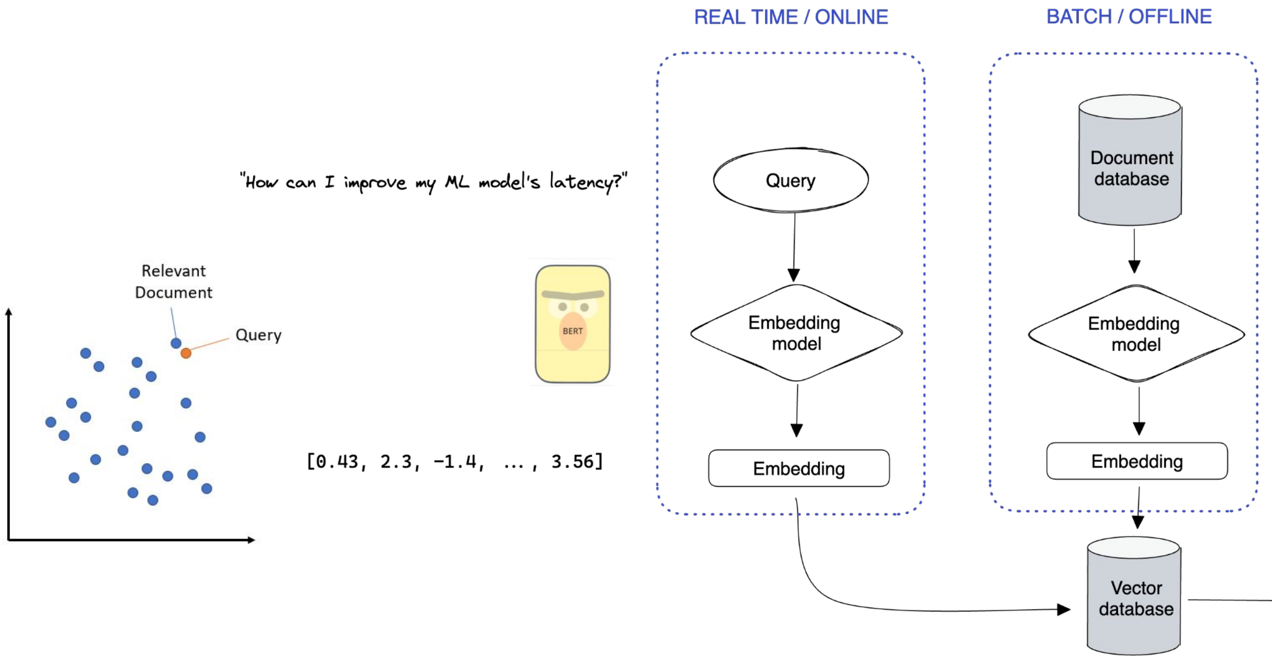


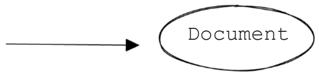
Asynchronous: The request for prediction (+ subsequent action) and the model computing result do not need to happen at the same time

- **Push**: The model generates predictions which are pushed as notifications to the user (e.g. fraud detection)
- **Pull**: The results are produced and stored in a database from where they can be retrieved in real-time for subsequent actions (e.g. recommendation engine, ...)

Batch serving is an example of asynchronous pull serving.

Search engine: Example of hybrid model serving.





Document embeddings: **Batch**

⇒ Batch: Historical load (big job of thousands of documents)

⇒ Asynchronous: If you receive a new document (embed it and store in vector DB)

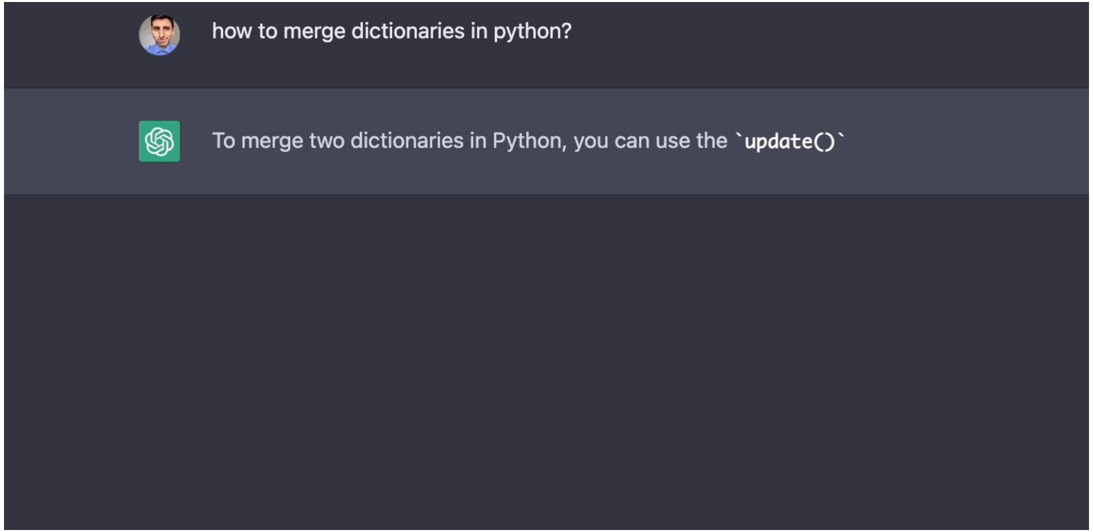
Query embeddings: **Real-time**

Credit: sbert.net

Streaming response

Some models produce outputs as a continuous stream of information.

Think of LLMs producing answers words per words instead of a full text after ~10 seconds.



Hybrid approach: Use both online and offline

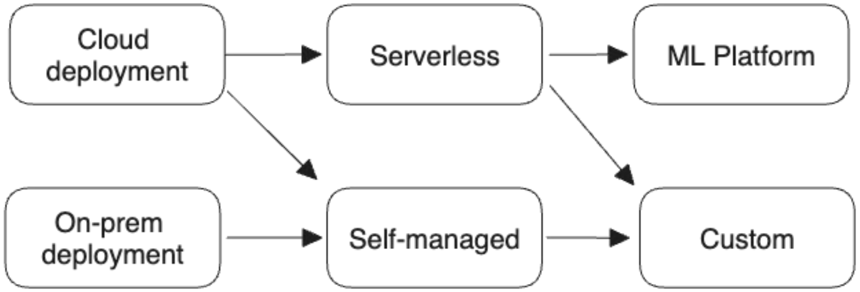
Pre-compute and store common queries

For example:

- DOORDASH
    - Restaurant recommendations use batch predictions
    - Within each restaurant, item recommendations use online predictions

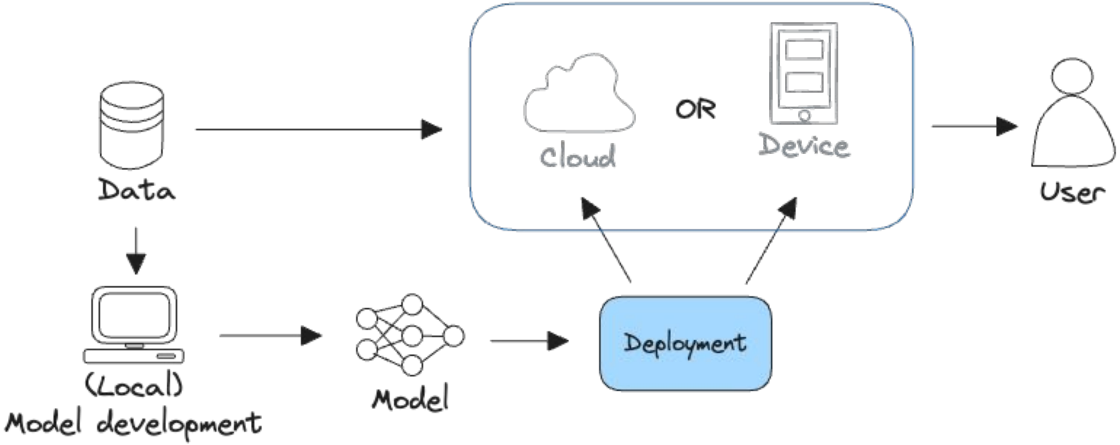
- NETFLIX
    - Title recommendations use batch predictions
    - Row orders use online predictions

Cloud vs on-prem deployment



Model deployment

You need to store your model somewhere so it can be called to do predictions on new data.

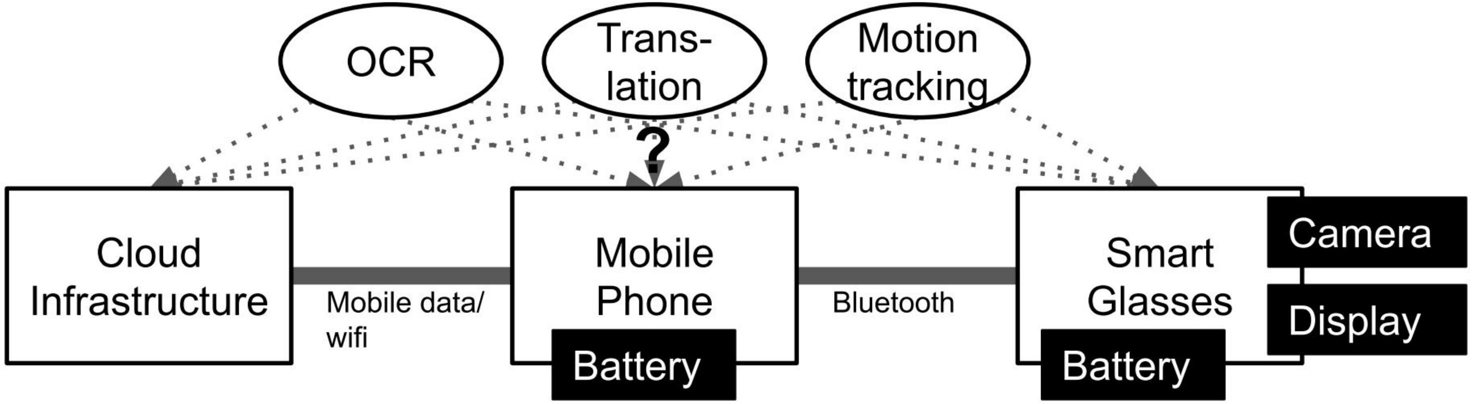


Example: Augmented reality translation

https://mlip-cmu.github.io/s2023/slides/09_deploying_a_model/deployment.pdf

Example: Augmented reality translation

Where should the models live?



Cloud? Phone? Glasses?
What qualities are relevant for the decision?

Considerations

- How much data is needed as input for the model?
- How much output data is produced by the model?
- How fast/energy consuming is model execution?
- What latency is needed for the application?
- How big is the model? How often does it need to be updated?
- Cost of operating the model? (distribution + execution)
- What happens if users are offline?

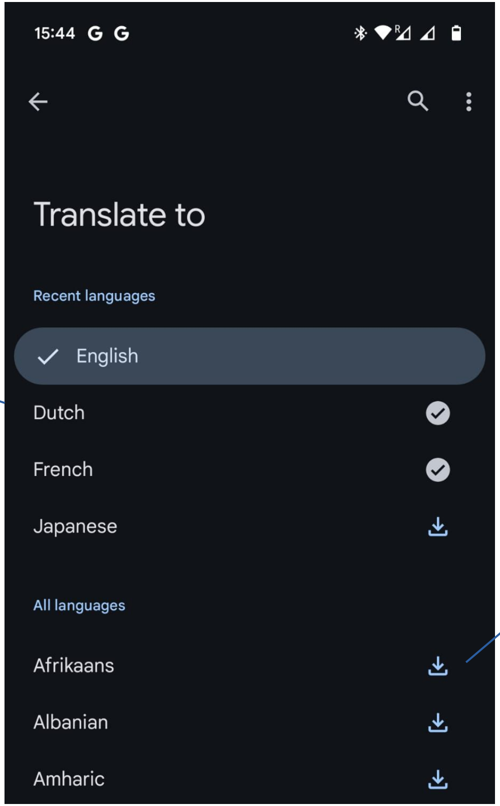
Cloud vs on-prem computing

	Cloud computing	On-prem computing
Computations	Done on **cloud** (servers). Model is stored on the Cloud and connected to through internet.	Model is stored on **on-prem** devices (own server, Jetson Nano, browsers, phones, smart watches, car, ...). Does not require internet connection, though it can help to update and maintain models.
Examples	• Most APIs • ChatGPT • Email spam filtering • …	• Next word predictions when typing in your smartphone • Self driving cars • Camera to detect where a rug needs to be cut

Example: Google Translate

Model downloaded on your phone (on-prem)!

Works offline.



Model on the cloud!

Needs internet.

Why on-prem deployment?

Pros	Cons
Privacy ● Sometimes hard requirement on data not leaving a specific location (hospitals, or personal data can't leave the EU due to GDPR) ● Less risk of data to be intercepted over network **Latency** ● No networking latency - can be faster ● Usually more of a monolith architecture which doesn't rely on microservices communicating to each other ● Even though **Unstable connexion** ● Can work where there is no internet connexion (e.g. self-driving car)	**Privacy** ● Actually pretty unsafe as someone can just physically intercept the data (steal the device) **Hardware** ● Can be really expensive to buy a performant hardware setup ● Don't have the "pay for what you use" - your training GPUs are wasted when nothing is running **Development** ● Often have to physically connect to the device ● Maintenance overhead (someone needs to physically go there if something crashes). **Scalability** ● Hard to increase to 20 new plants

Cloud deployment: ML Platform vs Custom deployment

Custom deployment

Containerise your model inference code as a **microservice** (as an **API**).

Spin up a compute instance on the Cloud and **host it yourself**.

Pros: Much more flexibility. Can add custom logic. Cheap.

Cons: Infrastructure implementation overhead. Time to deployment.

ML Platform deployment

Use a **standard model framework** (Pytorch, Tensorflow, Huggingface, …).

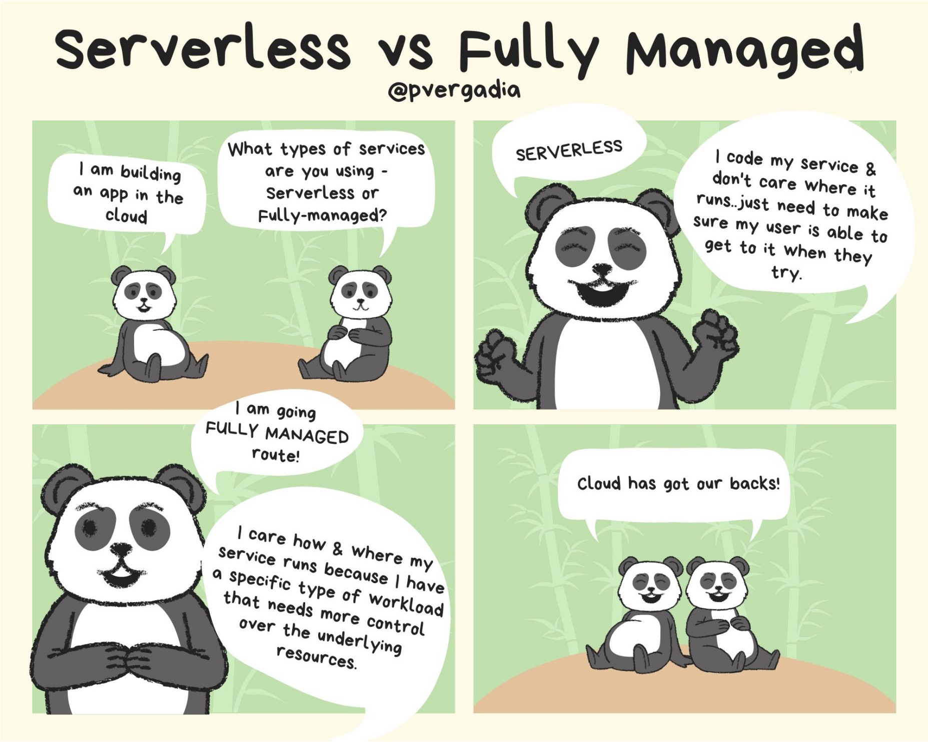
The **infrastructure** scaling and overhead is mostly **managed** by the **service provider**.

Provided on **platforms** such as Vertex, Azure ML or Sagemaker.

Pros: Less overhead, easy access to optimise infrastructure.

Cons: Limited customisation/flexibility. Hard to include custom logic. Expensive

Cloud custom deployment: Serverless



Cloud deployment deployment: Serverless

Serverless

- Cloud-native
- Developers don't have to manage servers
- Cloud provider abstracts away the provisioning, maintaining, and scaling the server infrastructure
- Developers just have to package their codes in a container
- Opposite is **self-managed**

<table>
<tr><th>Pros</th><th>Cons</th></tr>
<tr><td>

More efficient use of time for developers
Often cost effective (pay for what you use)
Simplified operations
Better adoption of DevOps / MLOps practices
Team collaboration / rotation
Stateless containerised application

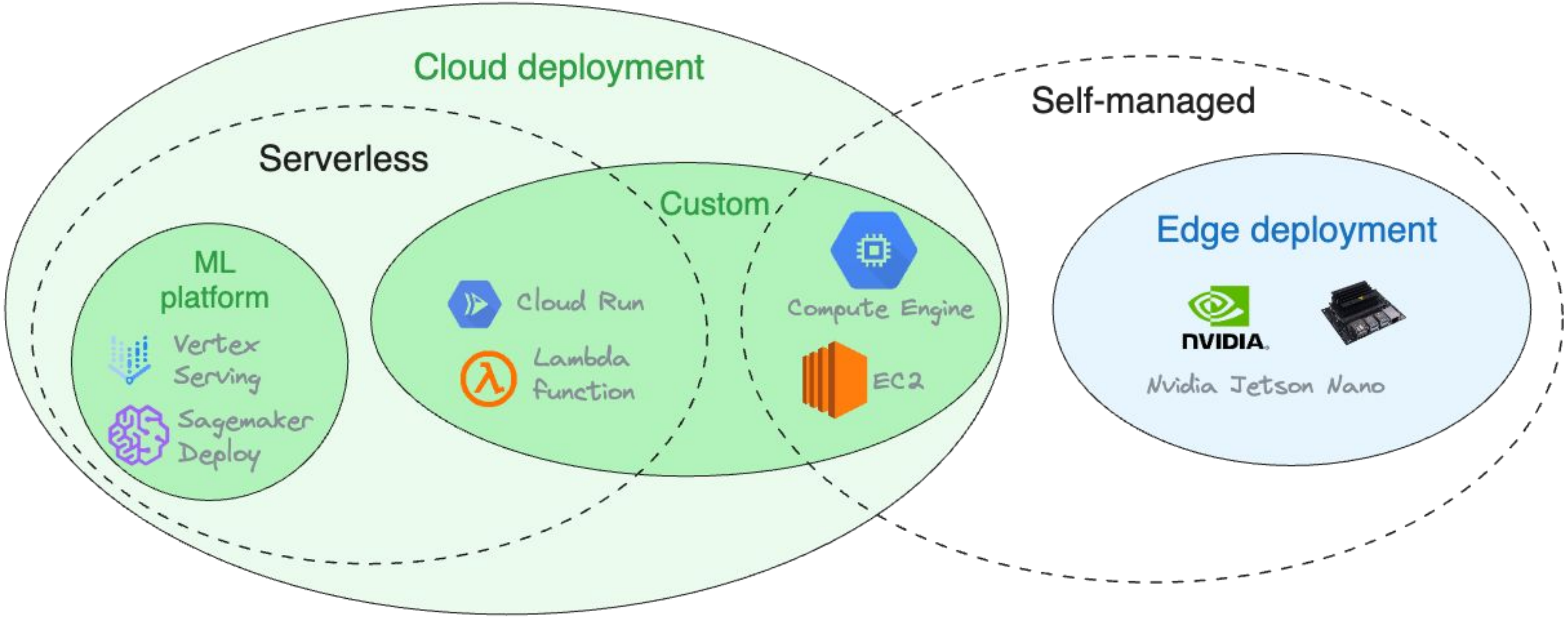
</td><td>

Less control over the server architecture
Type of compute
Interaction between components
Vendor lock-in
Debugging

</td></tr>
</table>

https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless

Example of different tools for different types of deployments



Parentheses: Cloud infrastructure incurs costs!

r/webdev • 1 yr. ago
PracticeEssay

Hate those moments when you accidentally spend $502 on Google Cloud

Discussion

Forgot brackets in my UseEffect call... Just running the npm live server in my browser sent 217 million requests to google firebase... Now I'm in $502 of debt...

https://preview.redd.it/1h6vl5k3xt2a1.png?width=949&format=png&auto=webp&s=f6fd6d6b6d969477bf1c8ee9da233f501e23c648

Hacker News new | past | comments | ask | show | jobs | submit login

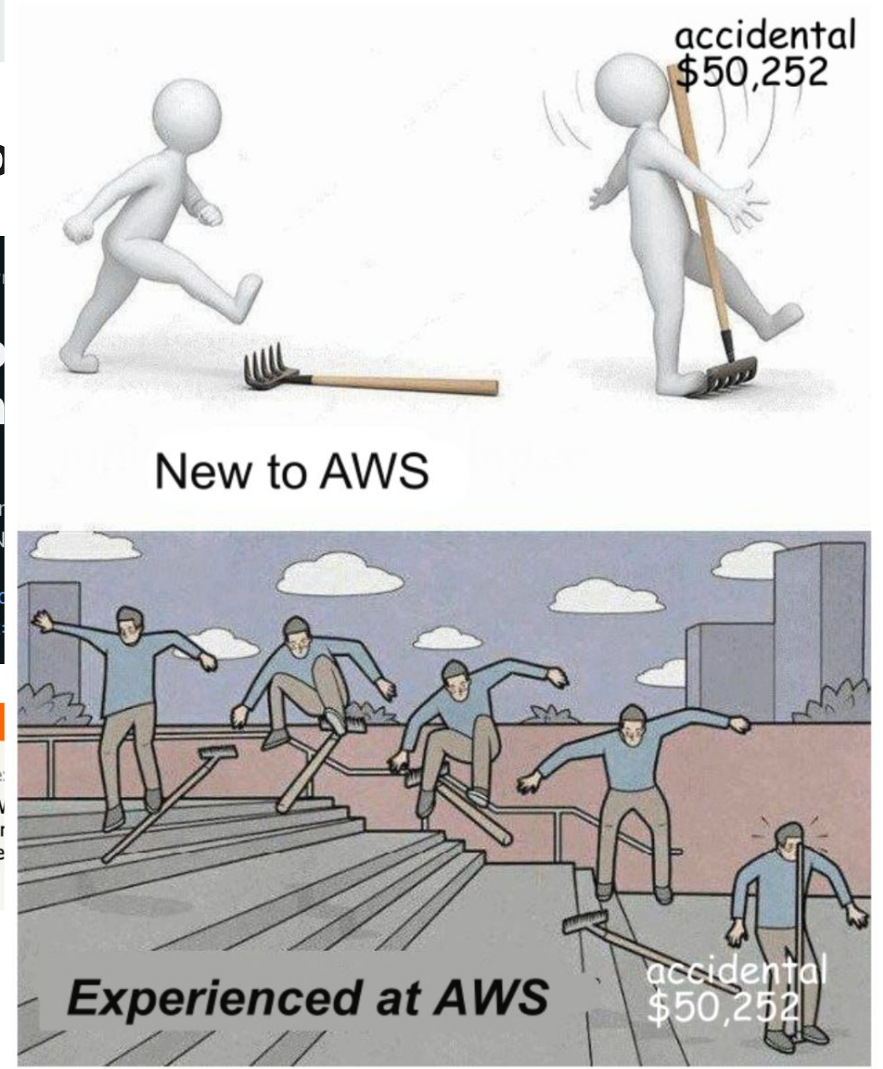
▲ appstorelottery on March 29, 2020 | parent | context | favorite | on: How to burn the most money with a single click in ...

A few years ago my startup was killed by a AWS mistake that ran overnight. The irony: my AWS expert at the time had made exactly the same provisioning mistake at his previous job - so I figured he'd never make a $80k mistake again. It turns out - his mistake with my startup was even more impressive. More positively - he did help shell out with me to cover the cost & overnight we were out of money. The mistake shocked me so much, and I've since heard *so* many stories of similar mistakes. The event hit me so hard I went back in time to PHP and shared hosting. Not kidding.

Parentheses: Clo

r/webdev • 1 y
PracticeEssay
Hate tho … end
$502 on
Discussion
Forgot brackets in … ests to
google firebase... N
https://preview.redd
width=949&format

Hacker News new | past | comments | … login
appstorelottery on March 29, 2020 | parent | conte
A few years ago my startup was killed by a AW … same provisioning mistake at his previous job - so I
figured he'd never make a $80k mistake agair … did help shell out with me to cover the cost &
overnight we were out of money. The mistake … hit me so hard I went back in time to PHP and shared
hosting. Not kidding.



Parentheses: Ways to manage Cloud costs

- Make sure to shutdown idle instances!
- Select coherent compute instance size
- Use services that **scale to 0**
  - Spins an instance only when getting requests. Shuts it down after a while.
- You can set parameters to your autoscaling :
  - Max concurrent instances
  - Max instance size
  - Max cloud spend
- You can set alerts (emails etc)
- Good to frequently have a look at your Cloud cost breakdown and identify avoidable costs
- Use batch instead of streaming when possible
- Use custom serving instead of managed serving

[GCP Cloud Run Pricing]

Let's look at some use cases where model latency is the difference between success or not...

Optimising latency in object detection on live combine harvest.

Finding Needles (and broken rice, leaves and ticks) in a literal haystack... but fast.

Use case: Installing a device on combines to automatically segment larger wheat objects and steering combine settings.

Type of serving: ??

Optimising latency in object detection on live combine harvest.

Finding Needles (and broken rice, leaves and ticks) in a literal haystack... but fast.

Use case: Installing a device on combines to automatically segment larger wheat objects and steering combine settings.

Type of serving: Real-time (synchronous) and on-edge

Improvements: ??

Optimising latency in object detection on live combine harvest.

Finding Needles (and broken rice, leaves and ticks) in a literal haystack... but fast.

Use case: Installing a device on combines to automatically segment larger wheat objects and steering combine settings.

Type of serving: On-edge and real-time (synchronous).

Improvements:

- Simplified model
- Optimised model (quantisation and pruning)
- Optimised framework (Tensorflow Lite)

(Deprecated)

Outcome:

- Faster than our competitors
- Cheaper hardware (200k $ difference)

LIÈGE université

Transcribing an archive database of call center audio recordings.

Use case: Large archive of call center a recordings to be transcribed using speech-to-text. Gain insights into phone calls in order to support and facilitate operators' tasks.

Type of serving: ??

LIÈGE université

Transcribing an archive database of call center audio recordings.

Use case: Large archive of call center a recordings to be transcribed using speech-to-text. Gain insights into phone calls in order to support and facilitate operators' tasks.

Type of serving: Batch.

Improvements:

- ??

Transcribing an archive database of call center audio recordings.

Use case: Large archive of call center a recordings to be transcribed using speech-to-text. Gain insights into phone calls in order to support and facilitate operators' tasks.

Type of serving: Batch.

Improvements:

- Not much possible with whisper out of the box
- Use lighter open source model (e.g. wav2vec)
- Hardware + cloud infrasctructure

Outcome: Blocking factor in the current track

https://huggingface.co/jonatasgrosman/wav2vec2-large-xlsr-53-english https://openai.com/research/whisper

Lab: Deploy an API in the Cloud

Wrap-up

Lecture summary

Topic	Concepts	To know for…	
		Project	Exam
Cloud infrastructure	• Guest lecture		
ML Model serving	• Batch vs real-time • Asynchronous vs synchronous • Push vs pull		Yes
Cloud vs on-prem deployment	• How and why for each option		Yes
Lab: Deploy an API in the Cloud		Yes	

Project objective for sprint 3

Note that optionally you can look into *managed* ML serving tools such as [Sagemaker Predict]() or [Vertex Predictions]().

#	Week	Work package	Requirement
3.1	W05	Build an **API** to **serve your model** and any extra logic that is needed to serve it (e.g. using Flask). You should be able to run the API locally.	Required
3.2	W05	Package your model serving API in a **Docker container**. This too should be run locally.	Required
3.3	W06	**Deploy** your model serving API in the Cloud. You should be able to call your model to generate new predictions from another machine. **Attention**: This can incur Cloud **costs**. Make sure to use a platform where you have credits and not burn through them. You can ask for support from the teaching staff in that regard.	Required

LIÈGE université