aivancity
PARIS-CACHAN
SCHOOL FOR TECHNOLOGY, BUSINESS & SOCIETY

Deployment and maintenance of artificial intelligence models

Dr. Souheil Hanoune
2024/2025

souheil.hanoune@neuralteks.com

TRANSCEND

ai

Souheil Hanoune

CTO & co-founder - NeuralTeks
Previously CTO/CSO - XXII GROUP

Experience :

- Co-founder & CTO of NeuralTeks
- 7 years at XXII Group (R&D Director, CSO, CTO/CSO)
- 10 years as a professor (Université de Cergy, ISEP, Institut Léonard de Vinci, AIvancity, École des Mines de Paris)
- Editorialist in ActuIA magazine

Training :

- PhD STIC (AI, robotics & cognitive sciences), CY Tech University
- 3 Masters, AI, Robotics, Computer science, Knowledge engineering, Sorbonne University
- Msc Engineering

Activities :

- Chairman of the scientific committee XXII GROUP
- ActuIA scientific committee member
- Conseil de perfectionnement du MS Deeptech de Mines Paris - Université PSL
- Conseil de perfectionnement Msc CY Tech

Souheil Hanoune CTO - CSO XXII GROUP

ABOUT US

XXII, AI computer vision software publisher

Our mission : Improving operational excellence

Our method : Reproducing the tremendous power of human vision

Our creation : A platform analyzing thoroughly and precisely video streams

Key figures

Founded in 2015
HQ in Paris La Défense
8 years of R&D, 150 PoC
€22M in Series A

€20 millions invested in R&D & product
80 employees (tech teams 50+ people)
7 PhDs in AI
400 sites deployed worldwide

XXII Worldwide

France Europe
USA Middle East

XXII

La vision par ordinateur, une solution pour mieux faire face aux grands enjeux sociétaux

Enjeux sociétaux

Environnement
Energie
Mobilité
Sécurité

Technologie

Intelligence Artificielle

Augmentation du nombre de caméras

SOLUTION

XXIICORE

- Optimiser l'efficacité des équipes,
- Augmenter la réactivité opérationnelle,
- Mesurer la performance pour orienter les investissements futurs.

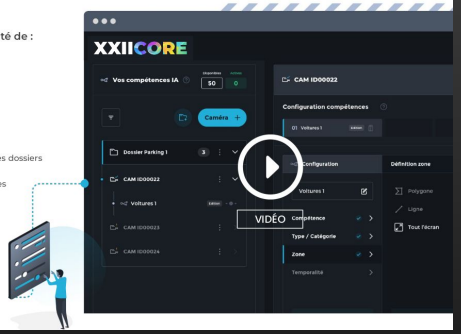
XXII CORE

XXII

Une interface simple répondant à vos usages

Notre interface de configuration offre la possibilité de :

- Paramétrer vos cas d'usage en 5 clics
  1. Sélection de l'usage
  2. Définition de la classe
  3. Définition de la zone d'analyse
  4. Réglage de la temporalité
  5. Gestion de la remontée
- Retrouver rapidement vos caméras avec la gestion des dossiers
- Programmer l'activation et la désactivation des usages
- Gérer les priorités d'alertes



XXII

Alertes - Temps réel

XXIICORE remonte les alertes en temps réel dans votre VMS ou HYPERVISEUR.

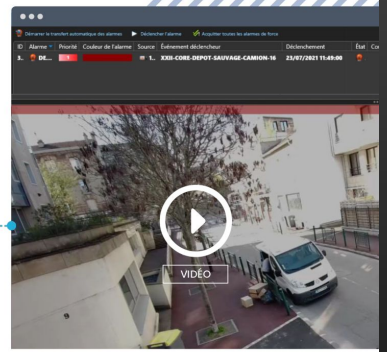
Fonctionnalités :

- Connexion simple et rapide à vos logiciels
- Configuration du type d'alerte
- Priorisation du traitement des alertes

Bénéfices :

- Alerte temps réel des situations sur tout le parc caméra
- Facilité de traitement de l'information
- Augmentation de la réactivité
- Réduction de la perte d'attention

Intégration :



XXII

Tableau de bord

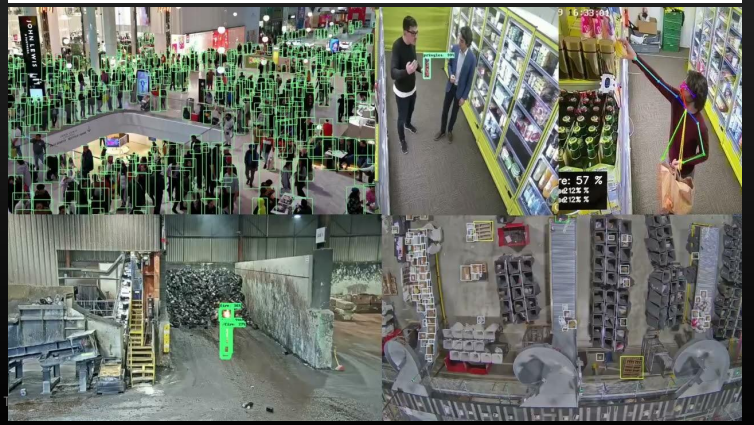
Interface de visualisation des données

Fonctionnalités :

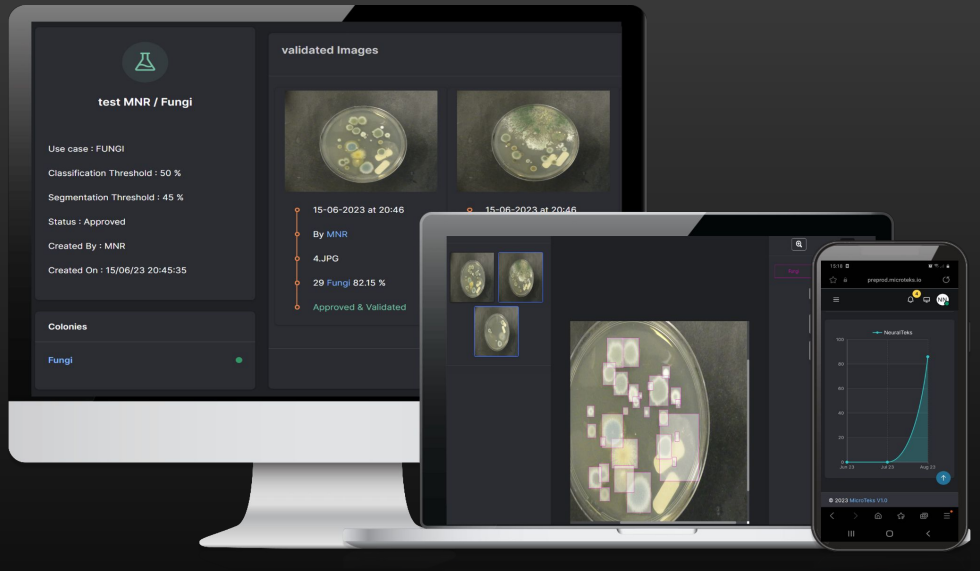
- Paramétrage des tableaux de bord
- Affichage multi-sources
- Gestion des seuils
- Comparaison par période
- Consultation à distance
- Génération et envoi de rapports
- Intégration dans vos systèmes

Bénéfices :

- Augmentation de la réactivité
- Visualisation de la donnée générée
- Connaissance en temps réel de la donnée



NeuralTeks & MicroTeks



TRANSCEND

SUMMARY

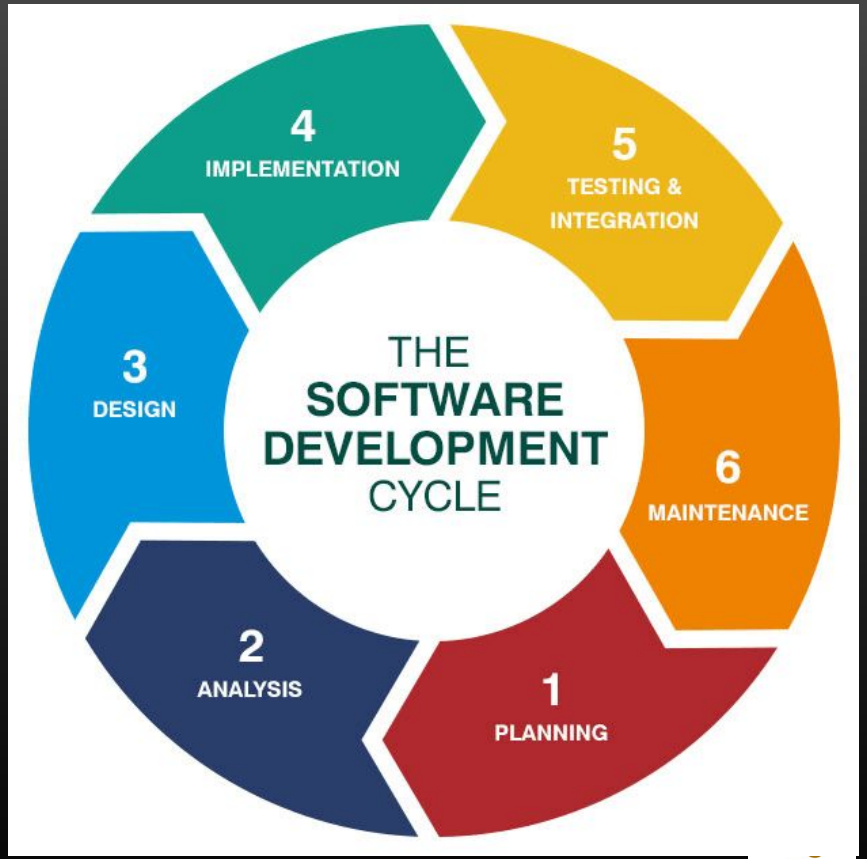
- Introduction to AI deployment
- Model selection and deployment options
- Deployment methods and technologies
- Deployment tools and frameworks
- Model management and version control
- Model monitoring and performance tuning
- Ethical considerations for AI models deployment

Introduction to AI Deployment

TRANSCEND

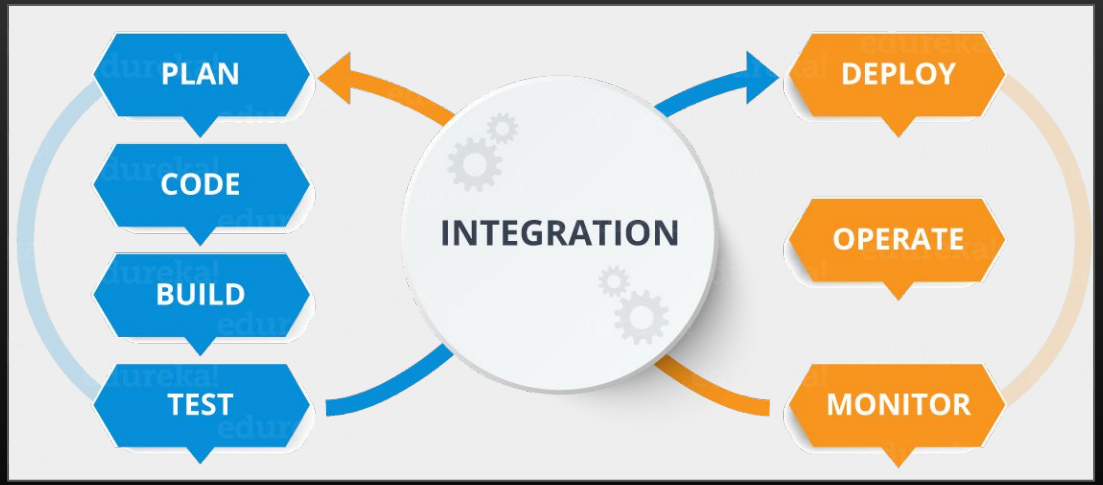
Software development cycle

- Software development is a **cyclic process**
- Usually, the **development team** and the **maintenance team** are different
- Separation of **tasks** facilitates **industriability**
- **Resilience** : Software -> existing bugs (probably)
- **Provide** value **continuously**



Software : feature development

- Each **feature** is developed separately
- The integration of the features incrementally **maximizes quality** and provides **value continuously**
- The **development** cycle and **deployment** cycle are **connected** to evolve the software



TRANSCEND

Definition of AI Deployment

- **Deployment** is the methods by which you **integrate** a **machine learning model** into an **existing production environment** to make practical business **decisions based on data**. It is one of the last stages in the machine learning life cycle

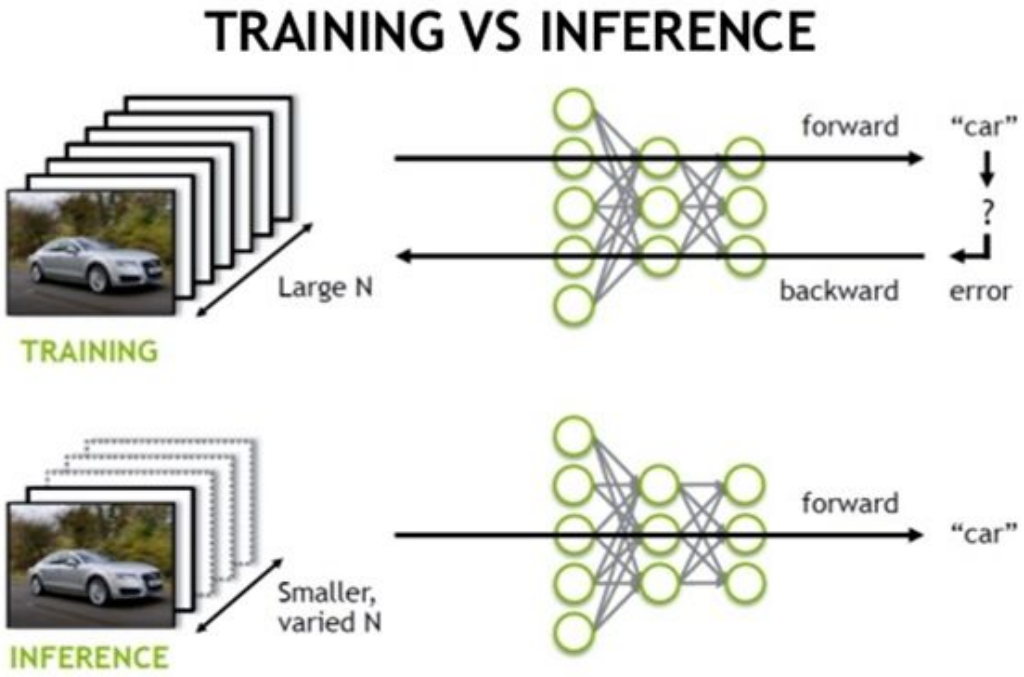
- Before you deploy a model, there are a couple of criteria that your machine learning model needs to achieve before it's ready for deployment:
    - **Portability**: this refers to the ability of your software to be transferred from one machine or system to another A portable model is one with a relatively low response time and one that can be rewritten with minimal effort
    - **Scalability**: this refers to how large your model can scale. A scalable model is one that doesn't need to be redesigned to maintain its performance

AI Deployment importance

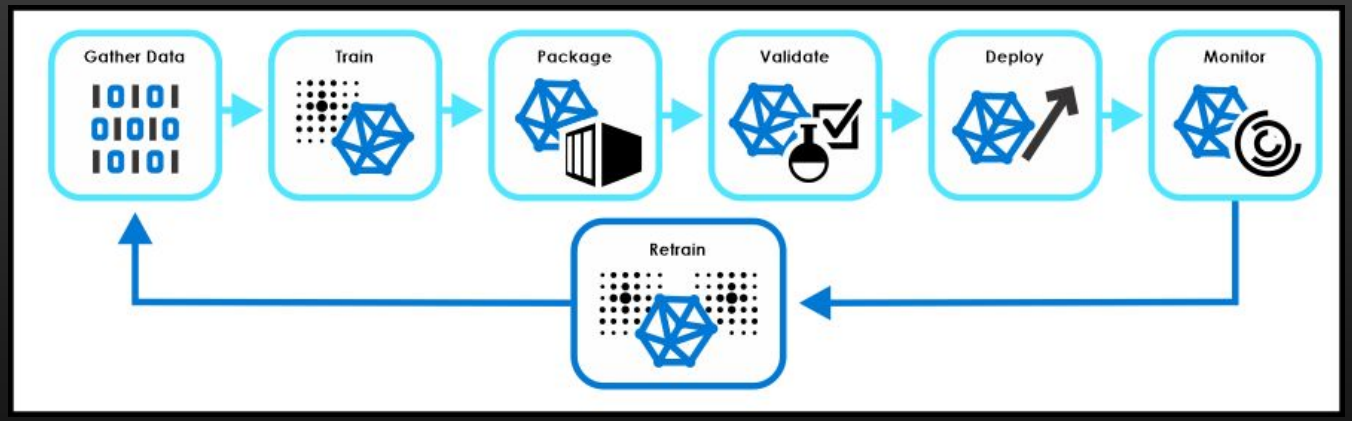
- Provide **value** to the **users**, the **clients** and the **market**
- **Update** model for better **performance**
- Address **uncompleted learnings** and **data drift**
- Ease **AI acceptability** (**integration** in the ecosystem)
- Minimize **cost** and maximize **value**
- Consider the **energetic** aspect and the **ecological** aspect

TRANSCEND

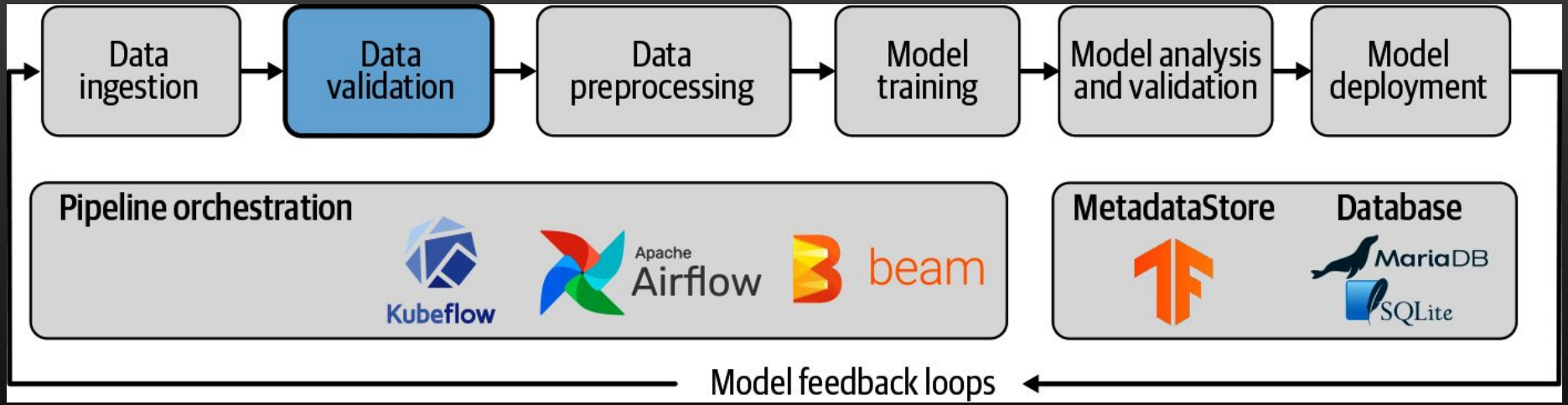
Training vs inference



Deployment VS training

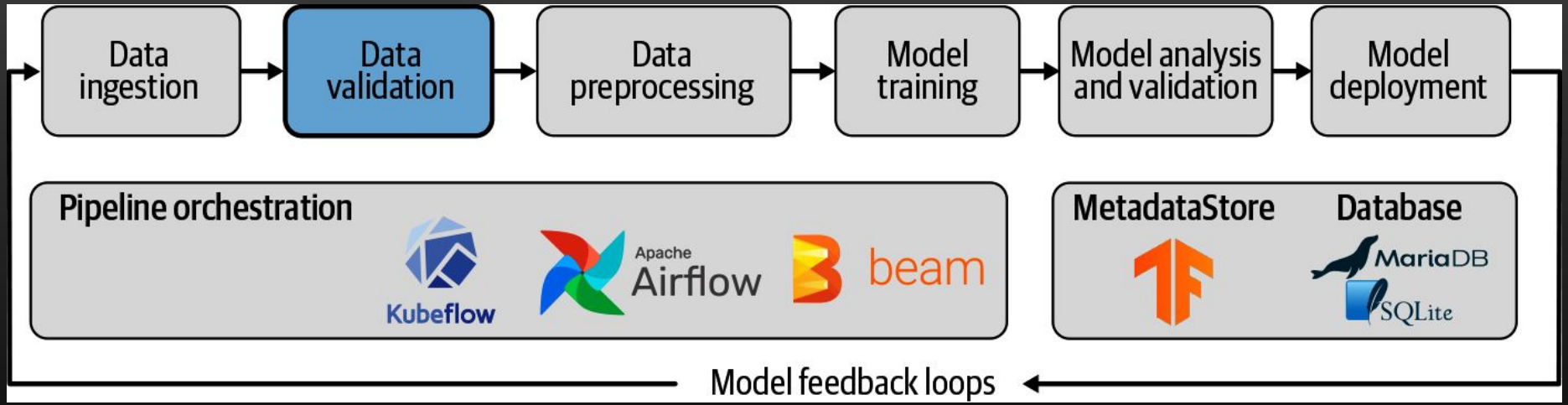


Example of AI pipeline



Example of AI pipeline

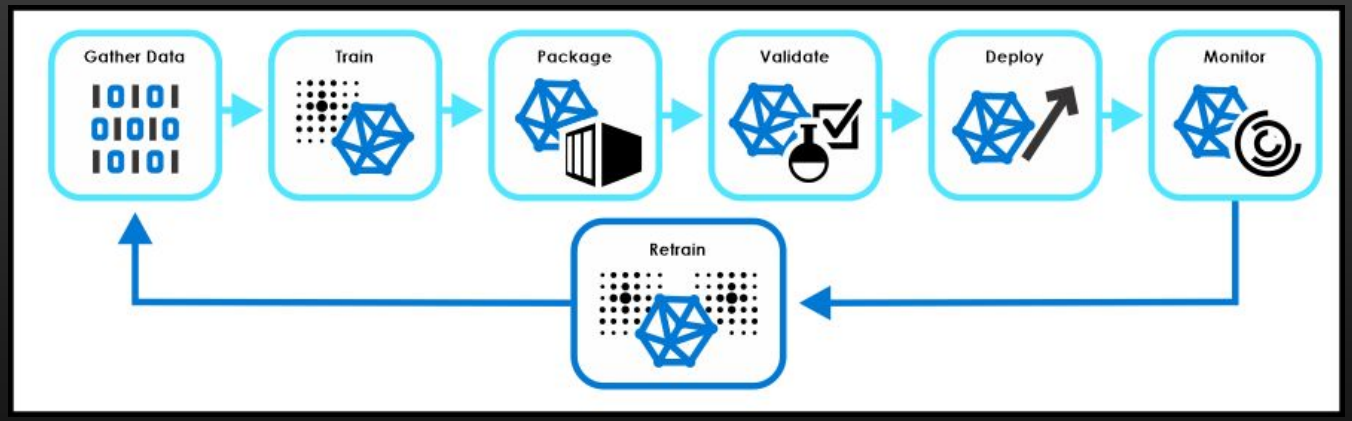
Take 10mn and imagine a use case to deploy and the various tools you know for each step of the pipeline



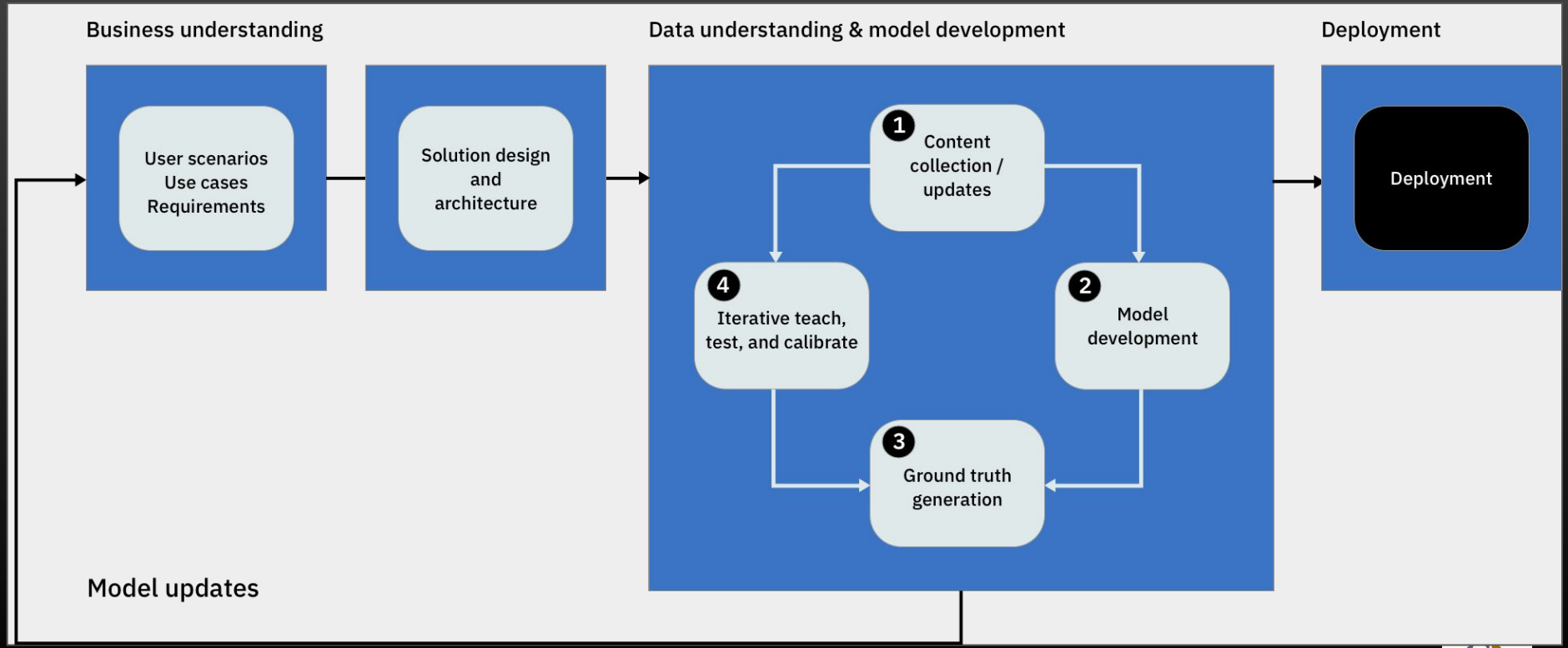
Model selection and deployment possibilities

TRANSCEND

Deployment VS training

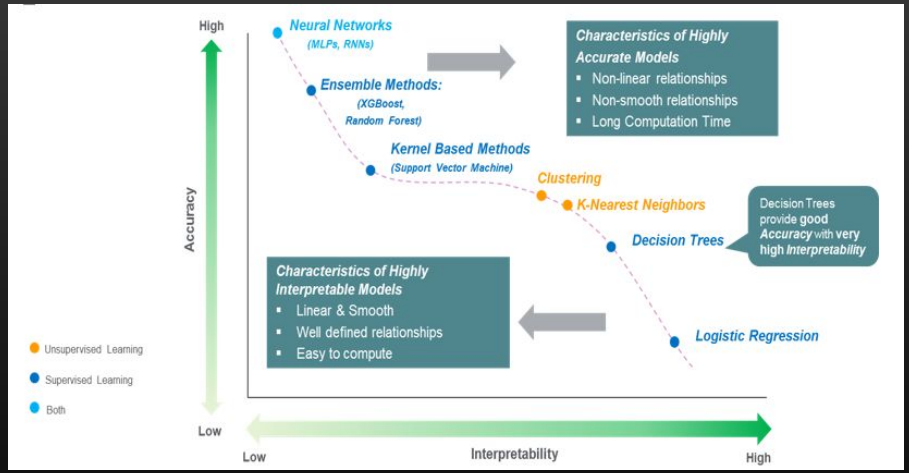


Continuous model deployment



Factors to consider when selecting a model

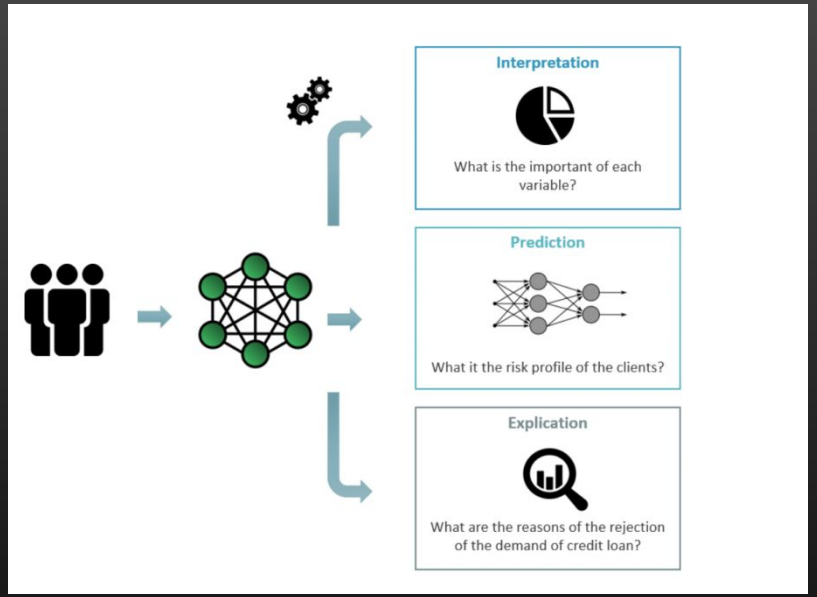
- **Accuracy**
  - How precise the model predictions are
- **Interpretability**
  - How understandable the decisions taken by the model are
- **Performance**
  - How fast and how much computing power does the model



Explicability vs. Interpretability

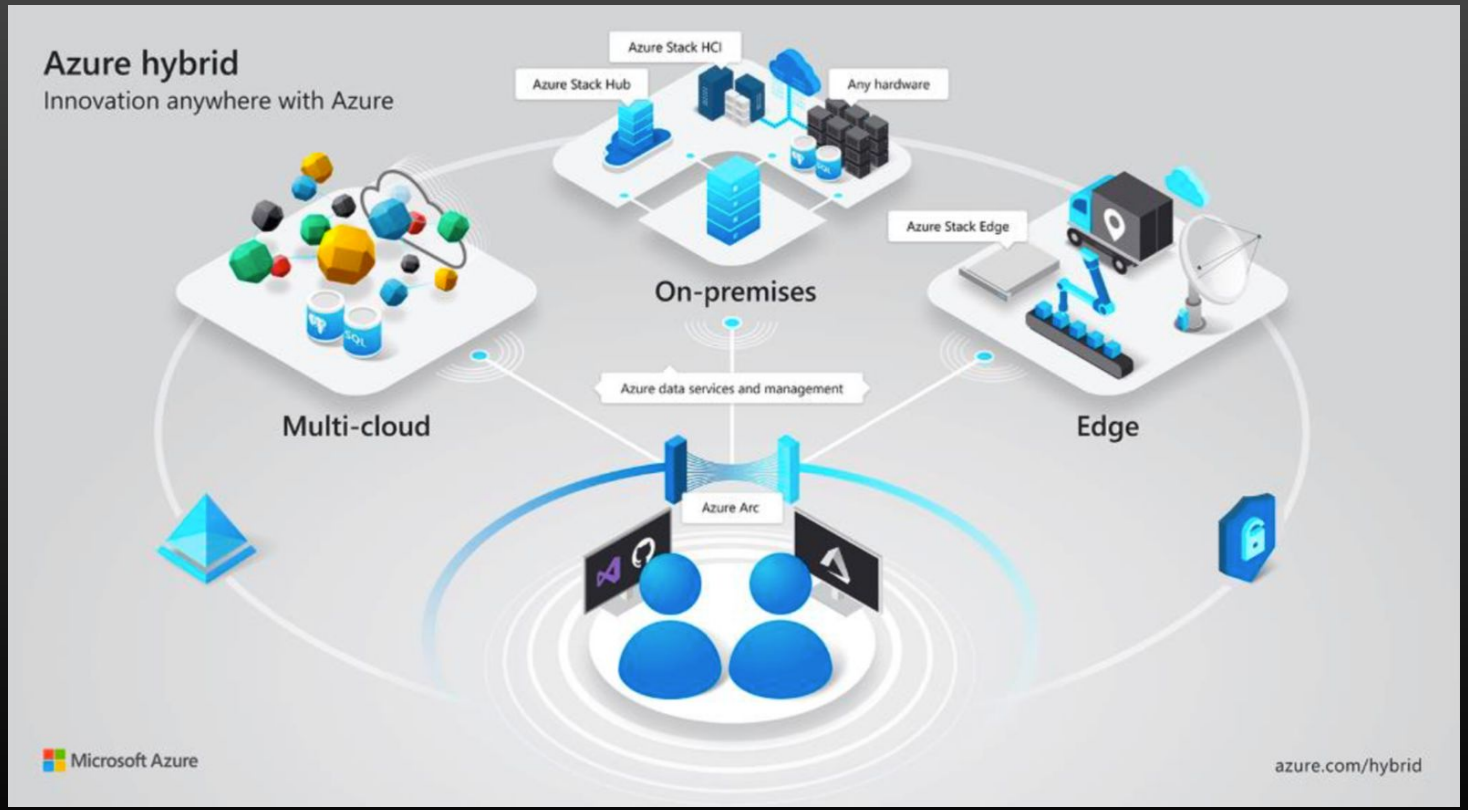
- **Explicability**
  - Can be **explicitly accounted** for on the basis of **known data** and **characteristics** of the situation
  - Possible to **relate** the **values taken** by certain variables (the characteristics) and **their consequences** on the prediction
  - For a bank loan the model is explainable if it explicitly indicates the relationship between the values taken by the variables (age, family situation, salary, etc.) and the final score

- **Interpretability**
  - Can identify the characteristics or variables that participate **most in the decision**, or even to quantify their importance
  - The model used for loan scoring is interpretable if it is able to measure the relative importance of the variables used (age, family situation, salary, etc.) in the determination of the final score

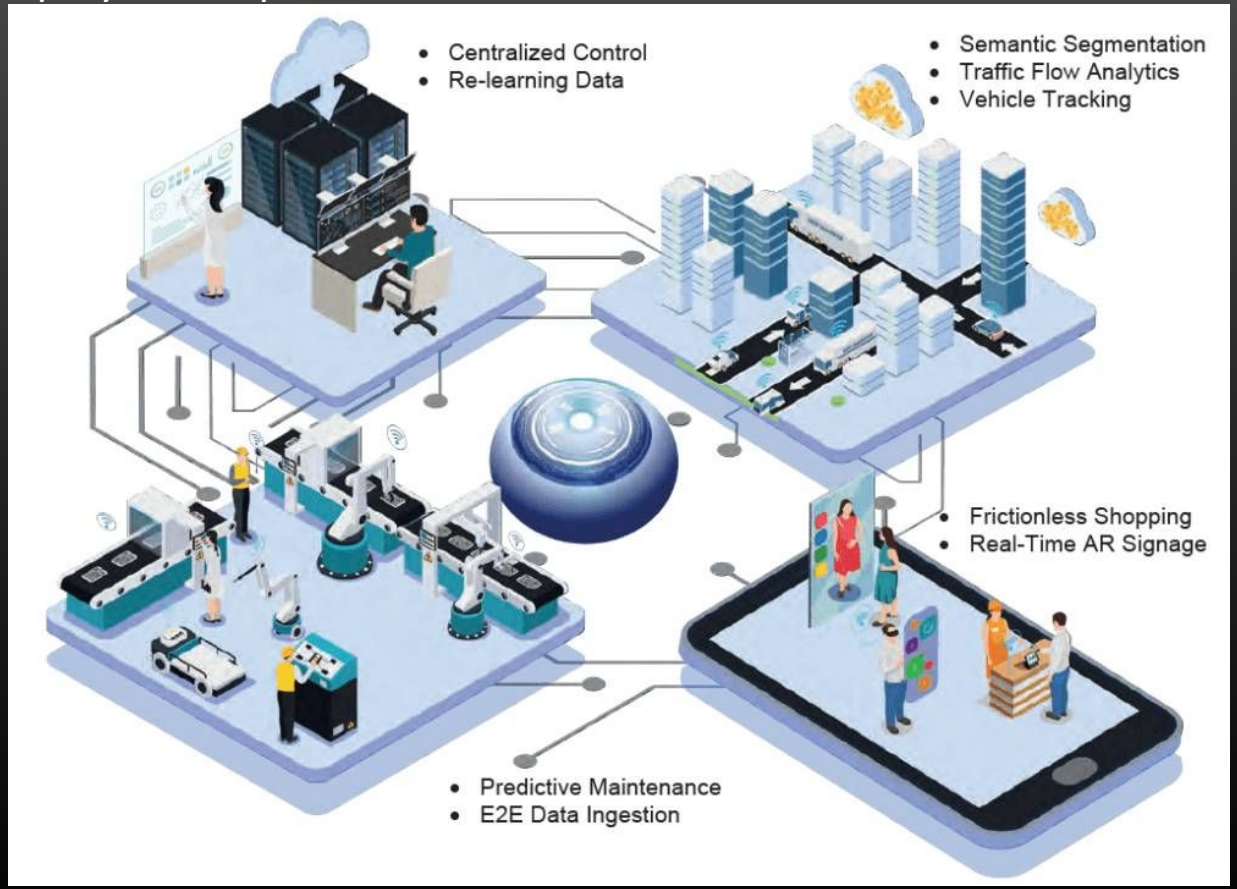


An explainable decision is interpretable

Deployment options



Examples of deployment options



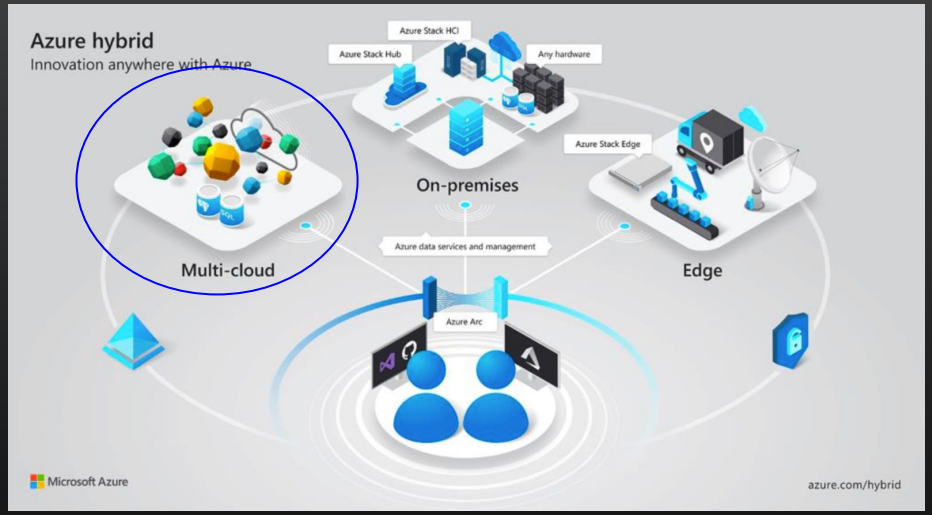
Cloud deployment

Pros

- Scalable computing power
- Hardware management
- Backups and storage
- Computing power ++++
- Hardware update scalable
- Deployment efforts

Cons

- Cost of operations without hardware ownership
- Lack of control
- Internet and network dependency
- Privacy concerns
- Latency (real-time use cases)



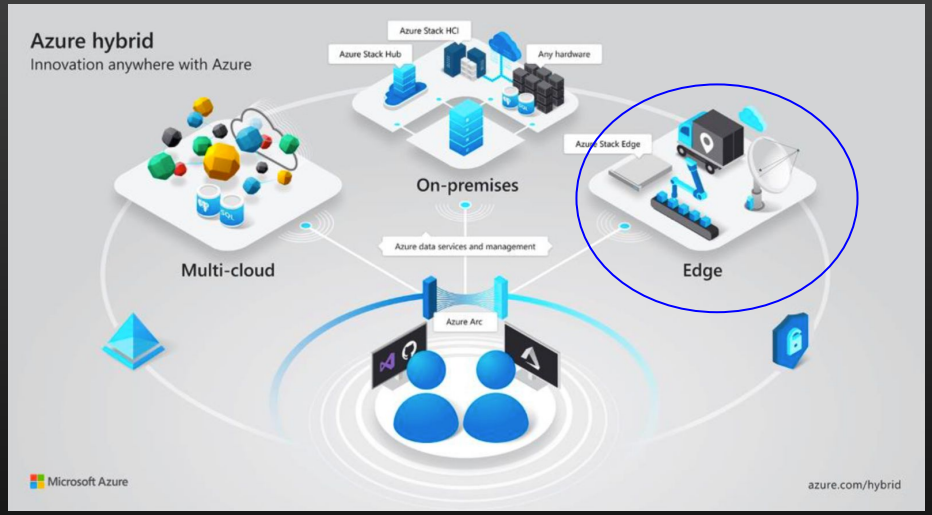
Edge deployment

Pros

- Extremely optimized
- Directly connected to the sensor
- High efficiency
- No network dependency and no delay
- Can manage real-time applications

Cons

- Cost of deployment
- Cost of hardware updates
- Deployment efforts
- Very limited computing power
- Usually software specific



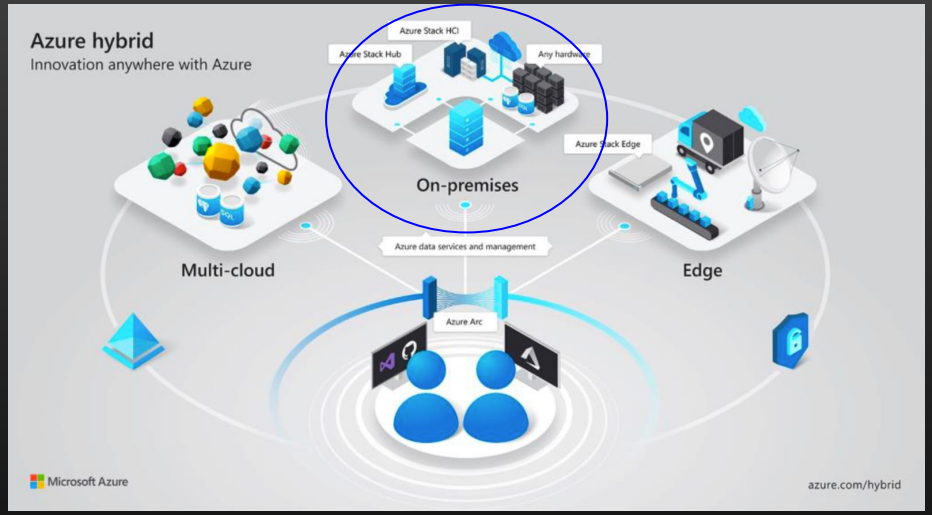
On premise deployment

Pros

- Compromise between cloud and edge deployments
- Close to the sensor and not internet dependent
- Can accept a wide range of computing power
- No network dependency and no delay
- Can manage real-time applications

Cons

- Hardware maintenance
- Cost of hardware
- Deployment efforts
- Fixed computing power



Methods to deploy a model

- **Factors to Consider When Determining Your Method of Deployment**
  - How frequently predictions will be generated and how urgent the results are needed?
  - If predictions should be generated individually or by batches
  - The latency requirements of the model, the computing power capabilities that one has, and the desired SLA
  - The operational implications and costs required to deploy and maintain the model

Types of Deployment in ML:

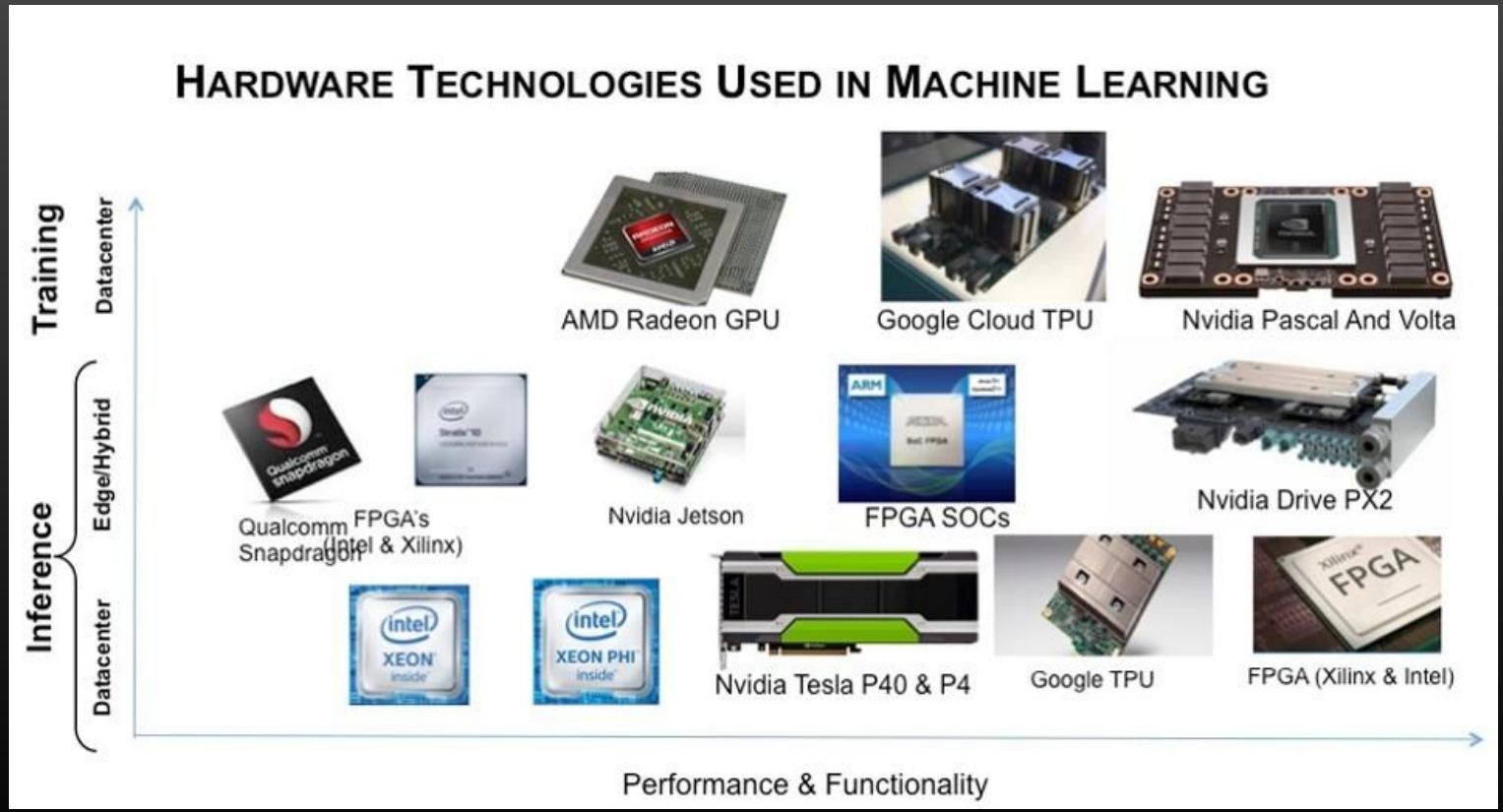
1. **On-Premises:** deployed within the organization's own infrastructure, behind their firewall. It offers maximum control and security, but requires high upfront costs and maintenance

2. **On cloud:** Model is hosted on a cloud service provider's infrastructure. flexibility, scalability, and low upfront costs, it may raises concerns around data privacy and security

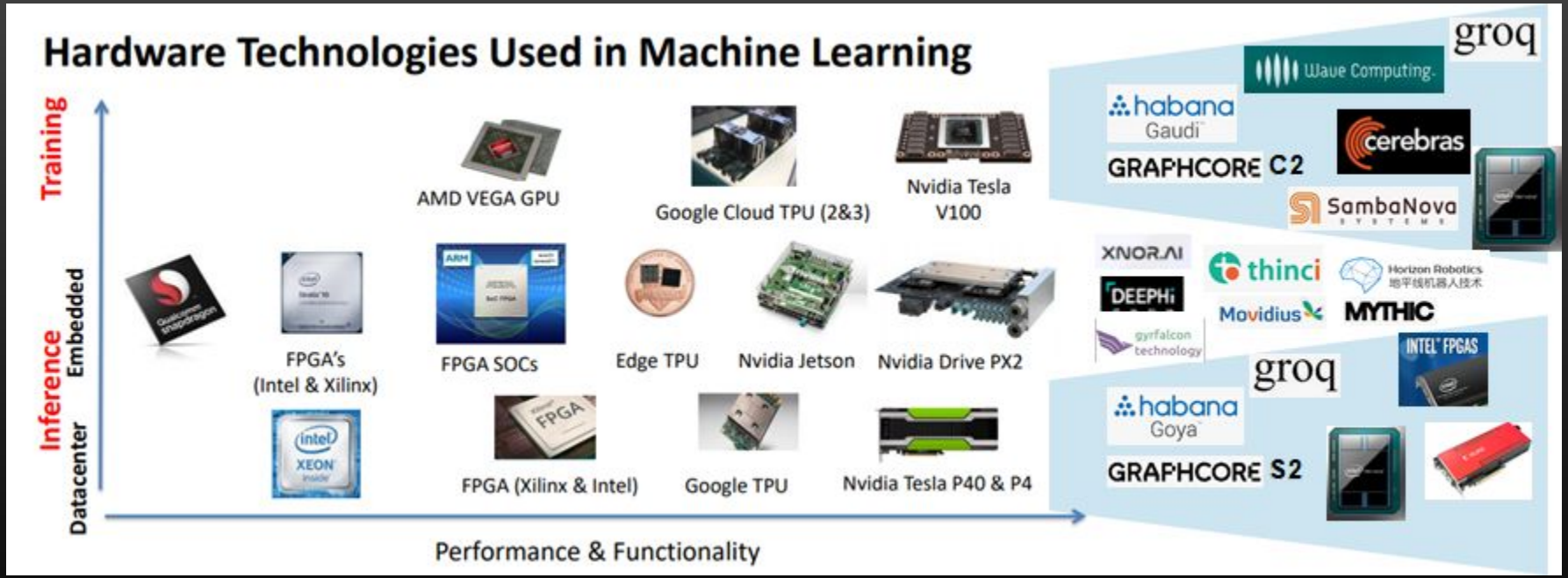
3. **Edge :** model is deployed on an edge device(Internet of Things (IoT)), such as a sensor or a camera, allowing for real-time analysis of data. low latency, cost-effective solutions, and reduces network traffic, but requires careful consideration of power and computational resources

4. **Mobile:** model is deployed on a mobile device, allowing for offline prediction capabilities. low latency and personalized experiences, but requires careful consideration of model size and computational resources

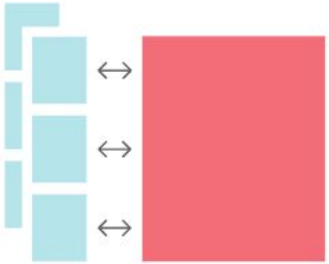
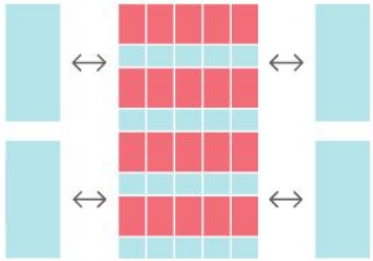
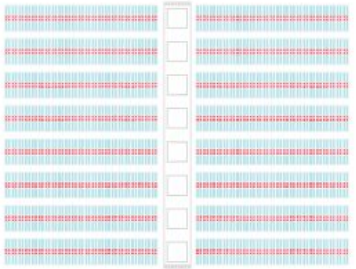
The compute



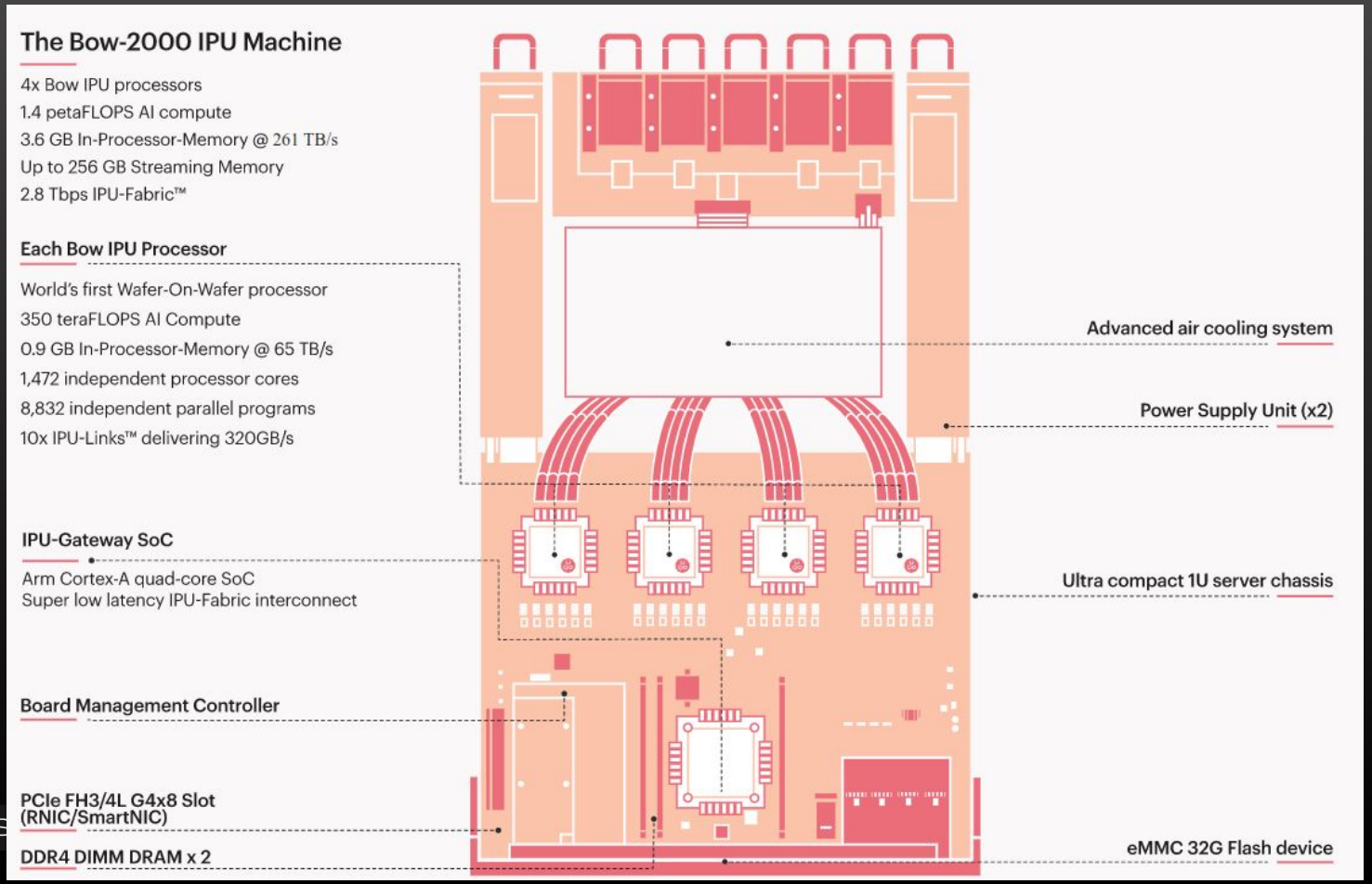
The compute



Specific AI Hardware : GraphCore case study (IPU)

	CPU	GPU	IPU
Parallelism	Designed for scalar processes	SIMD/SIMT architecture. Designed for large blocks of dense contiguous data	Massively parallel MIMD. Designed for fine-grained, high-performance computing
			
Memory Access	Off-chip memory	Model and data spread across off-chip and small on-chip cache, and shared memory	Model and data tightly coupled, and large locally distributed SRAM

Specific AI Hardware : GraphCore case study (IPU)



Deployment methods and technologies

TRANSCEND

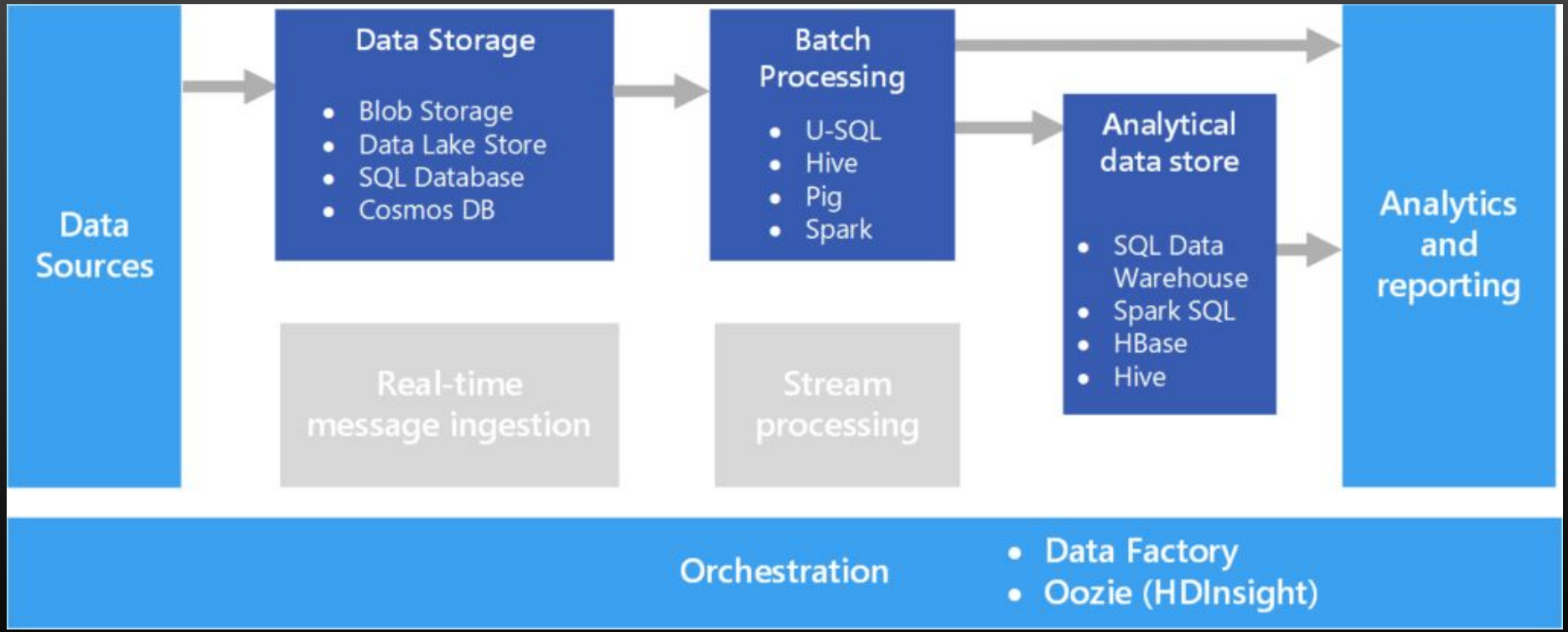
Methods to deploy depending on model update

- **Fixed model**
  - When the performances needed are identified and the process is stable
  - The model can simply be trained ad-hoc when it’s needed and pushed to production until it deteriorates enough to need some fixing
- **Updatable model**
  - Allows for constant up-to-date version of the model
  - Scalable method that uses a subsample of data at a time and eliminating the need to use the full data set for each update
  - A good method if you use the model on a consistent basis but without requiring the predictions in real-time

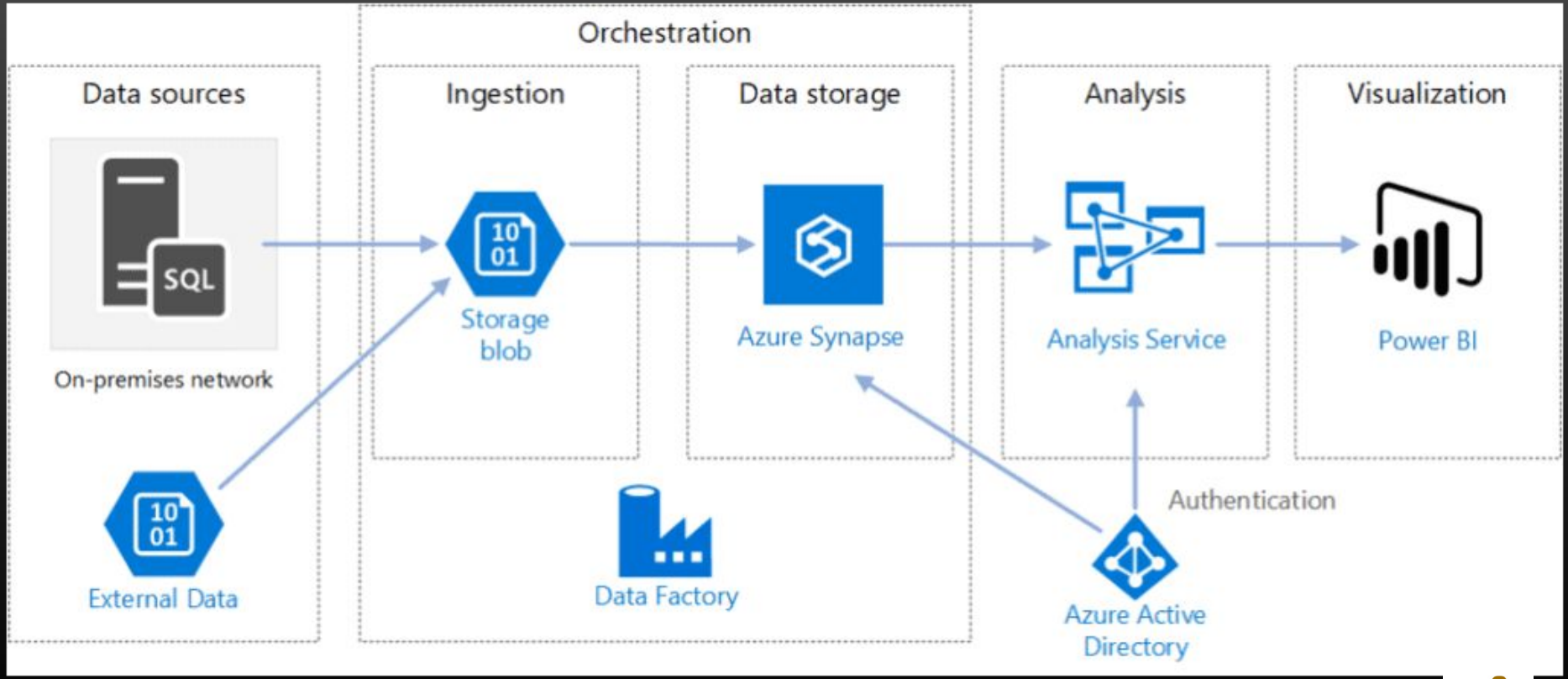
Methods to deploy a model for data consumption

- **One-off**
  - Sometimes a model is only needed once or periodically for a period of time
  - The model can simply be trained ad-hoc then deployed for specific use then the deployment is interrupted
  - Typically used during the development process
- **Batch**
  - Allows for constant up-to-date version of the model and for fixed versions
  - Process large amounts of data all at once
  - A good method if you use the model on a consistent basis but without requiring the predictions in real-time
  - Allows to optimize the requests from multiple users on the hardware
- **Stream or Real-time**
  - When real-time predictions are needed (determine if a transaction is fraudulent or not) or when streams are analysed (video analysis)
  - The data output rate must be just as fast as the data input
  - The model is continuously analyzing data and providing an inference with no down time

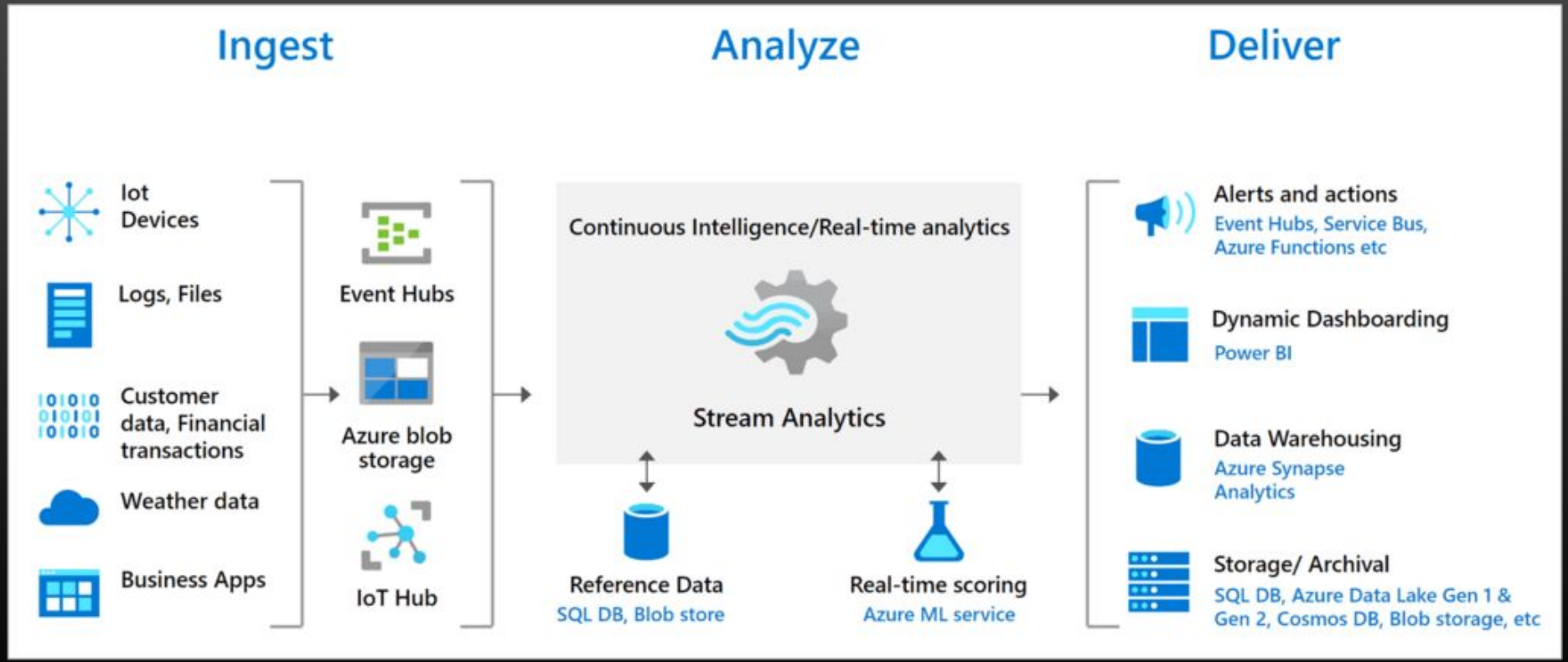
Batch & stream processing architecture



Example of batch processing architecture on Azure



Example of stream processing architecture on Azure

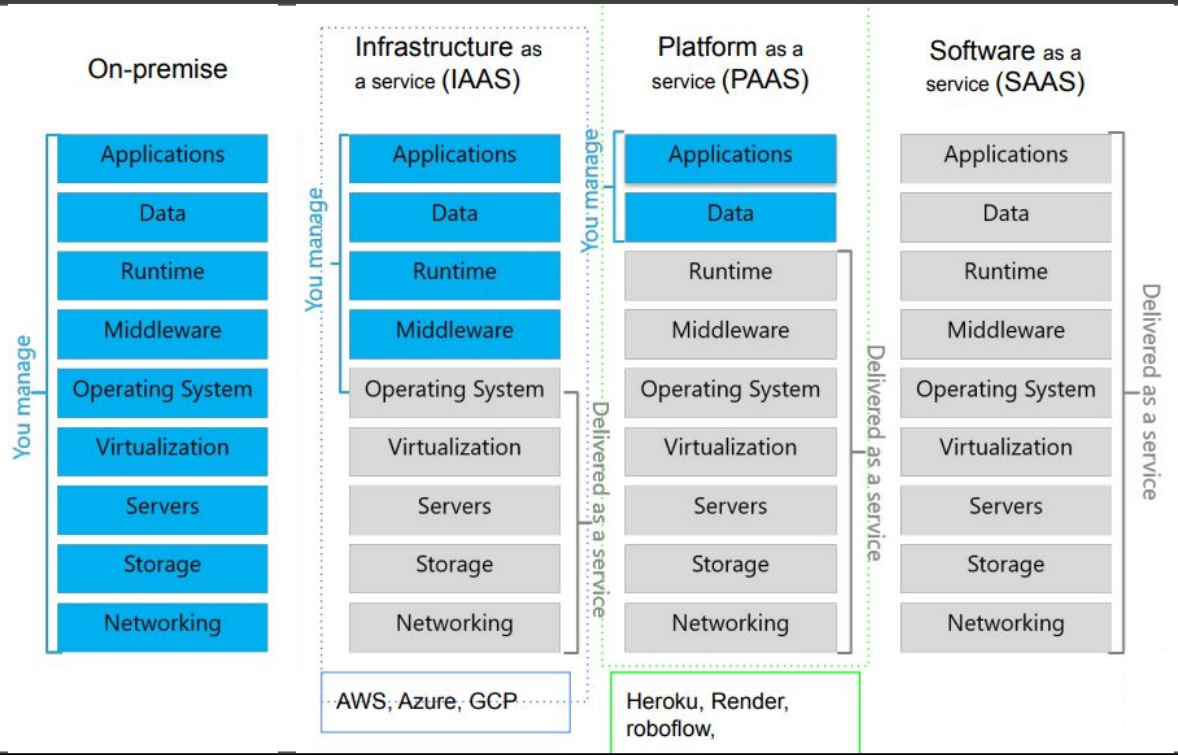


Key technologies used in AI deployment

- **Virtual environments**
- **Virtual machine**
- **Containers & containers management**
- **Containers deployment**
- **Monolithic VS microservice architectures**
- **Software development workflow with and without docker**

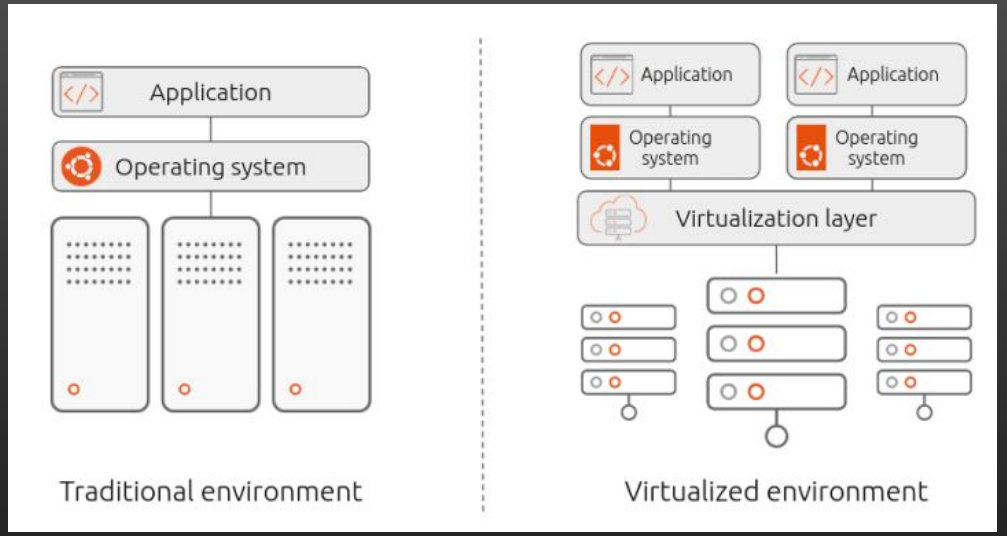
Virtualization

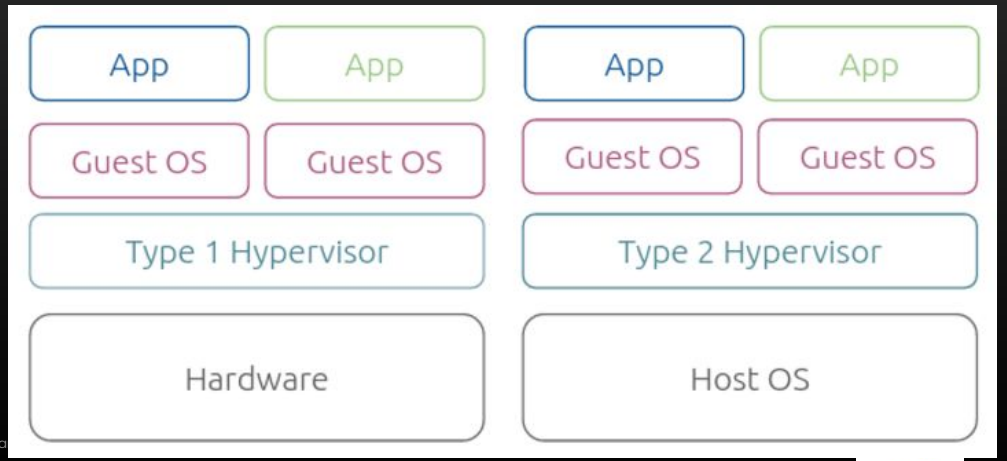
Cloud Service Models Diagram



Virtualization

- Provide a virtual environment over the operating system or over the hardware
- Allows for the management of the application dependencies, drivers and other requirements
- Facilitates the deployment
- Garanties reproducibility
- Type 1 hypervisor over hardware ; type 2 over host OS





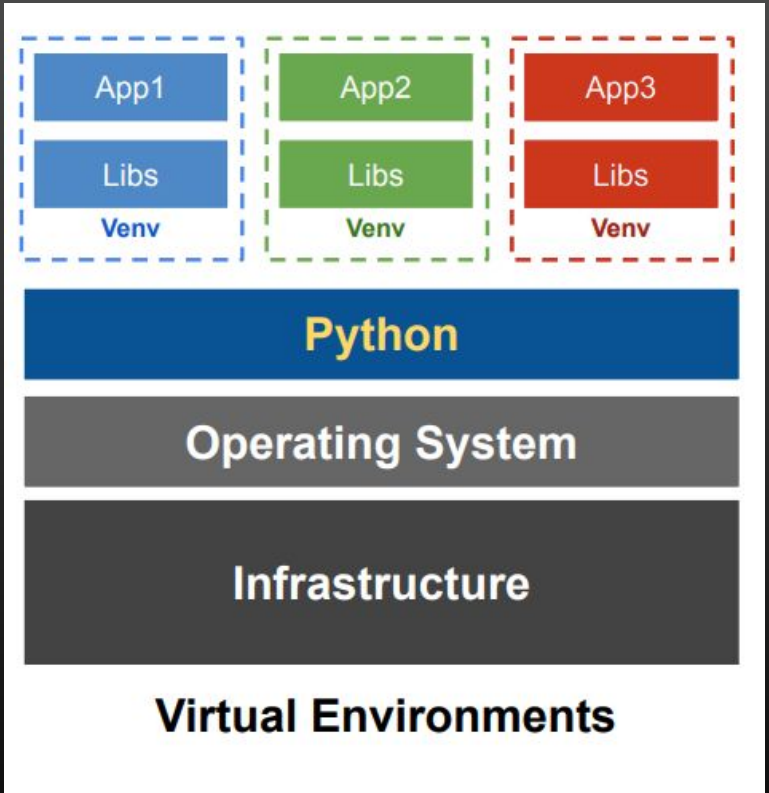
Virtual Environments

Pros

- Reproducible results
- Explicit dependencies
- Improved engineering collaboration

Cons

- Difficulty setting up your environment
- No isolation
- Does not always work across different OS



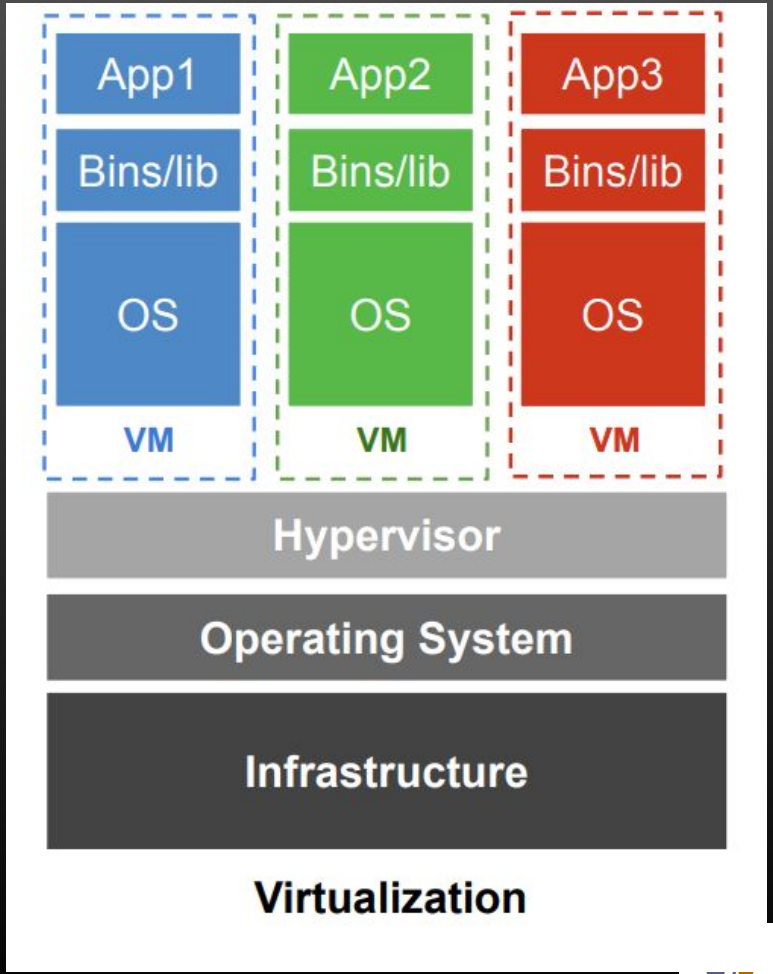
Virtual Machines (Virtualization)

Pros

- Full autonomy
- Very secure
- Lower costs
- Used by all Cloud providers for on demand server instances

Cons

- Uses hardware in local machine
- Not very portable since size of VMs are large
- There is an overhead associated with virtual machines



Containerization (Docker)

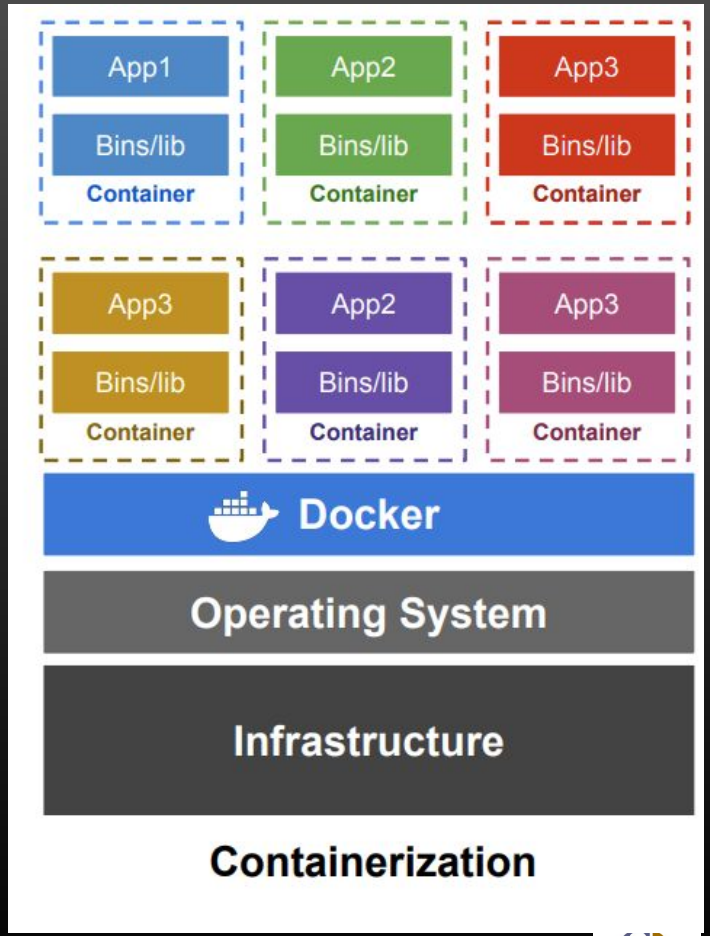
We want a system that:

- Automatically set up (installs) all OS and extra libraries and set up the python environment
- It is isolated
- Uses less resources
- Startups quickly

Whats is a container ?

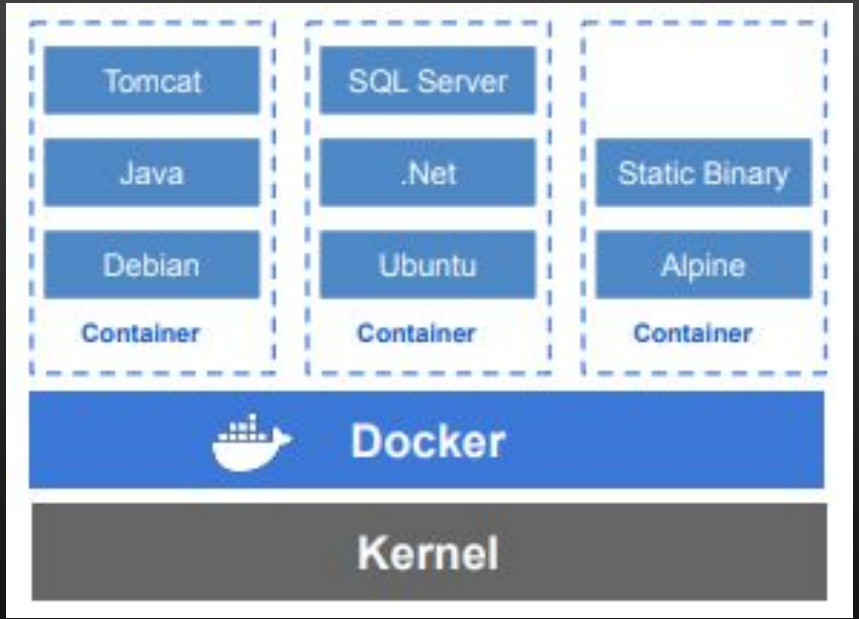
- Extremely **portable** and lightweight
- **Fully packaged** software with all dependencies included
- Can be used for **development, training, and deployment**
- Development teams can easily **share** containers

Docker is an open source platform for building, deploying, and managing containerized applications.



Containerization (Docker)

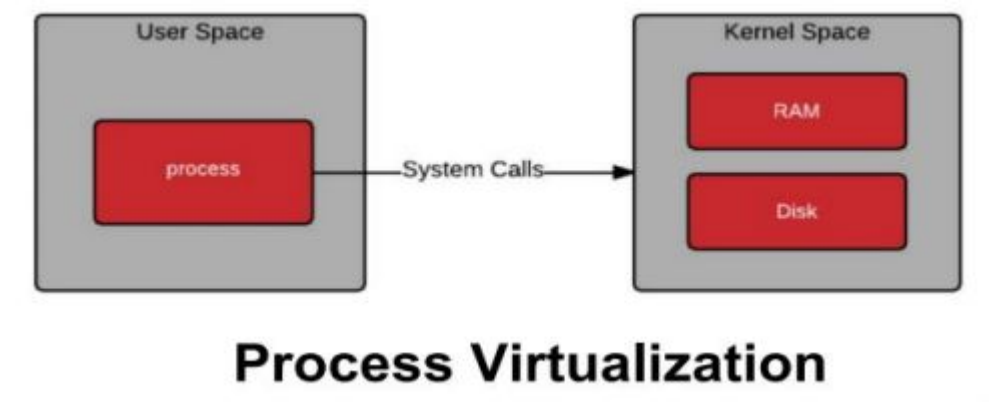
- **Standardized** packaging for software dependencies
- **Isolate** apps from each other
- **Works** for all major Linux distributions, MacOS, Windows



Containerization (Docker)

Container = User Space of OS

- User space refers to all of the code in an operating system that lives outside of the kernel



Docker pipeline

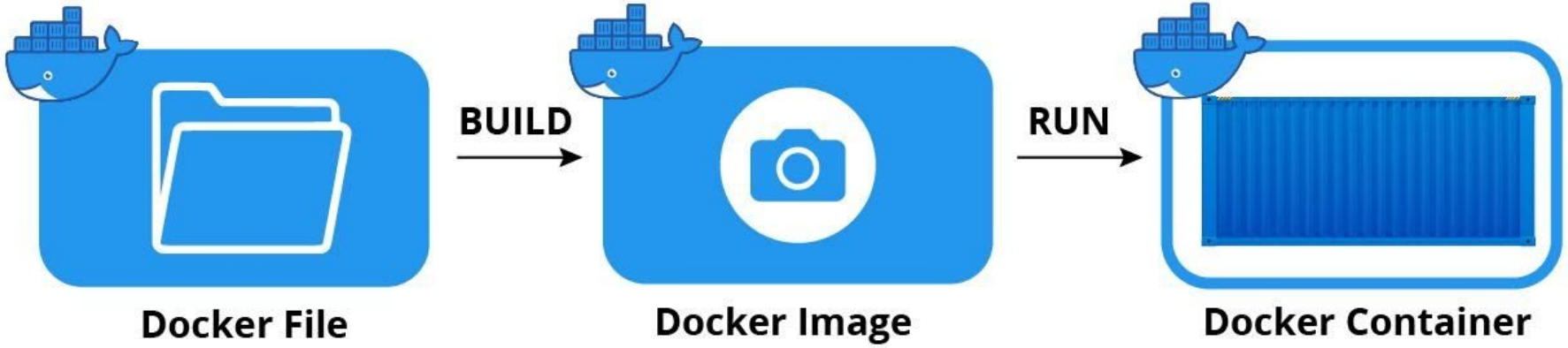
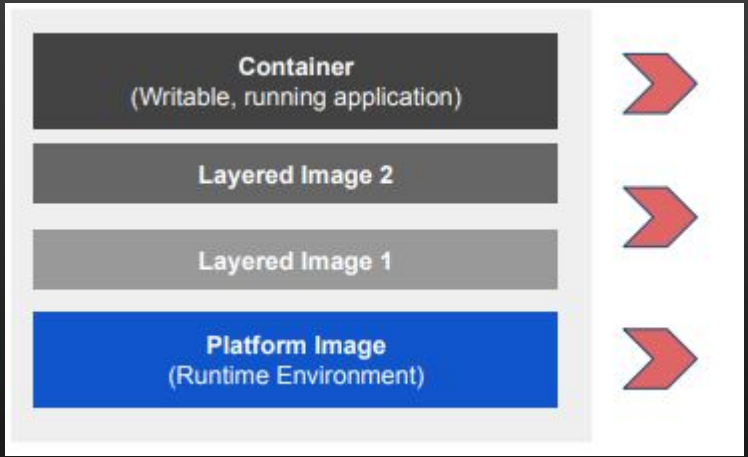


Image layering



A application sandbox

- Each container is based on an image that holds necessary config data
- When you launch a container, a writable layer is added on top of the image

A static snapshot of the container configuration

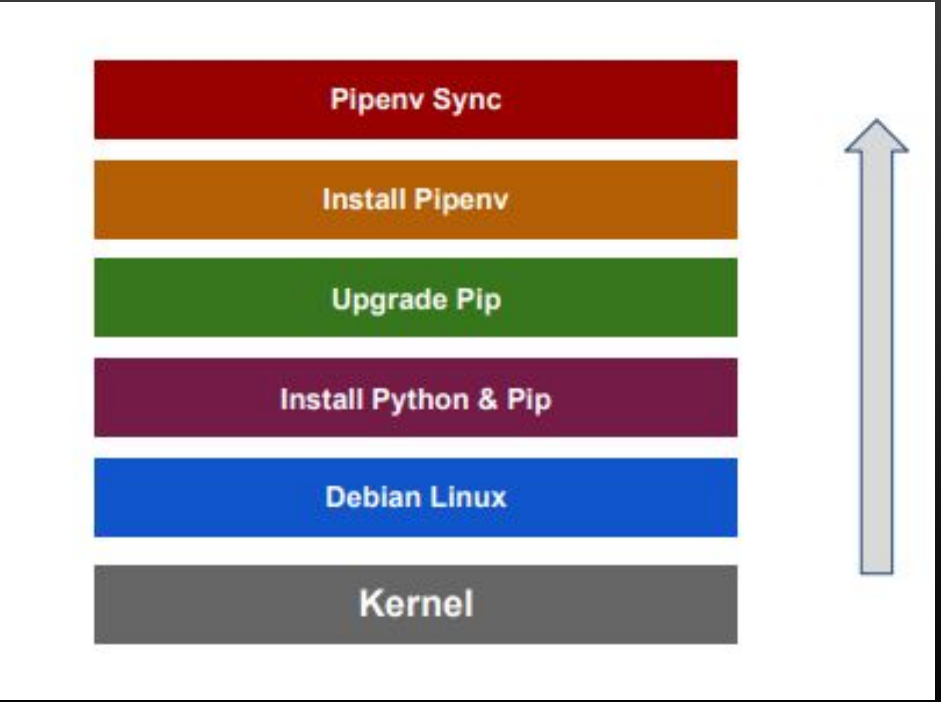
- Layer images are read-only
- Each image depends on one or more parent images

An Image that has no parent

- Platform images define the runtime environment, packages and utilities necessary for containerized application to run

Image layering

Docker layers for a container running debian and a python environment using Pipenv



Docker vocabulary

Docker Image

The basis of a Docker container. Represent a full application

Docker Container

The standard unit in which the application service resides and executes

Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

Registry Service (Docker Hub or Docker Trusted Registry)

Cloud or server-based storage and distribution service for your images

Images
How we **store** the application

Containers
How we **run** the application

Engine
Where we **run** the application

Registry
Where we **store** the application

TRANSCEND

Docker command example

Check docker version

docker	**--version**
docker command	Get docker CLI version

Docker command example

List all running docker containers

docker	**container**	**ls**
docker command	Docker command for containers	Docker option to list all containers

Docker command example

List all docker images

docker **image** **ls**

docker command — Docker command for image — Docker option to list all containers

Docker command example

Build an image based on a Dockerfile

Build the image

Name of the dockerfile in current directory “.”

docker build -t ac215-d1 -f Dockerfile .

docker command

Name of the image

Docker Image as Layers

```
>docker build -t hello_world_cmd -f Dockerfile_cmd .

Sending build context to Docker daemon  34.3kB
Step 1/4 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
54ee1f796a1e: Already exists
f7bfea53ad12: Already exists
46d371e02073: Already exists
b66c17bbf772: Already exists
Digest: sha256:31dfb10d52ce76c5ca0aa19d10b3e6424b830729e32a89a7c6eee2cda2be67a5
Status: Downloaded newer image for ubuntu:latest
 ---> 4e2eef94cd6b
Step 2/4 : RUN apt-get update
 ---> Running in e3e1a87e8d6e
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [67.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [231 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [1078 B]
```

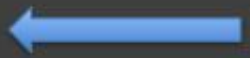
Step1: Instruction 1

Step2: Instruction 2

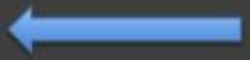
Docker Image as Layers

```
>docker build -t hello_world_cmd -f Dockerfile_cmd .

....
Step 3/4 : ENTRYPOINT ["/bin/echo", "Hello"]
 ---> Running in 52c7a98397ad
Removing intermediate container 52c7a98397ad
 ---> 7e4f8b0774de
Step 4/4 : CMD ["world"]
 ---> Running in 353adb968c2b
Removing intermediate container 353adb968c2b
 ---> a89172ee2876
Successfully built a89172ee2876
Successfully tagged hello_world_cmd:latest
```



Step3: Instruction 3



Step4: Instruction 4

Docker Image as Layers

```
> docker images
REPOSITORY        TAG             IMAGE ID        CREATED          SIZE
hello_world_cmd   latest          a89172ee2876    7 minutes ago    96.7MB
ubuntu            latest          4e2eef94cd6b    3 weeks ago      73.9MB
```

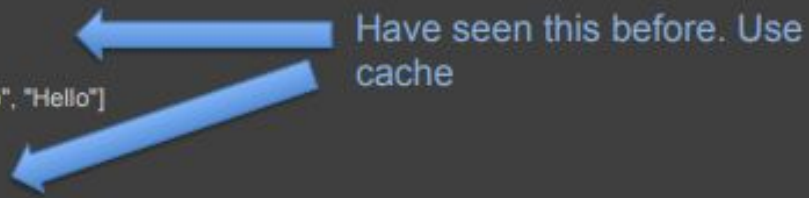
```
> docker image history hello_world_cmd
IMAGE          CREATED          CREATED BY                                      SIZE          COMMENT
a89172ee2876   8 minutes ago    /bin/sh -c #(nop)  CMD ["world"]                0B
7e4f8b0774de   8 minutes ago    /bin/sh -c #(nop)  ENTRYPOINT ["/bin/echo" "…   0B
cfc0c414a914   8 minutes ago    /bin/sh -c apt-get update                       22.8MB
4e2eef94cd6b   3 weeks ago      /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      3 weeks ago      /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>      3 weeks ago      /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   811B
<missing>      3 weeks ago      /bin/sh -c [ -z "$(apt-get indextargets)" ]     1.01MB
<missing>      3 weeks ago      /bin/sh -c #(nop) ADD file:9f937f4889e7bf646…   72.9MB
```

Why Layers?

Why build an image with multiple layers when we can just build it in a single layer?

Let's take an example to explain this concept better, let us try to change the Dockerfile_cmd we created and rebuild a new Docker image.

```
> docker build -t hello_world_cmd -f Dockerfile_cmd .
Sending build context to Docker daemon  34.3kB
Step 1/4 : FROM ubuntu:latest
 ---> 4e2eef94cd6b
Step 2/4 : RUN apt-get update
 ---> Using cache
 ---> cfc0c414a914
Step 3/4 : ENTRYPOINT ["/bin/echo", "Hello"]
 ---> Using cache
 ---> 7e4f8b0774de
Step 4/4 : CMD ["world"]
 ---> Using cache
 ---> a89172ee2876
Successfully built a89172ee2876
Successfully tagged hello_world_cmd:latest
```



As you can see that the image was built using the **existing** layers from our previous docker image builds. If some of these layers are being used in **other containers**, they can just use the existing layer instead of recreating it from scratch.

Your turn

Create a docker file that :

- Pulls an ubuntu image (with NVIDIA Libs if necessary)
- Installs Python, tensorflow and any other necessary library
- Downloads a Yolo detector (choose as you wish)
- Runs the Yolo detector on images available in a repertory in your computer
- Stores back the detection images in your computer

Software architectures

There are multiple types of software architecture that can be implemented.

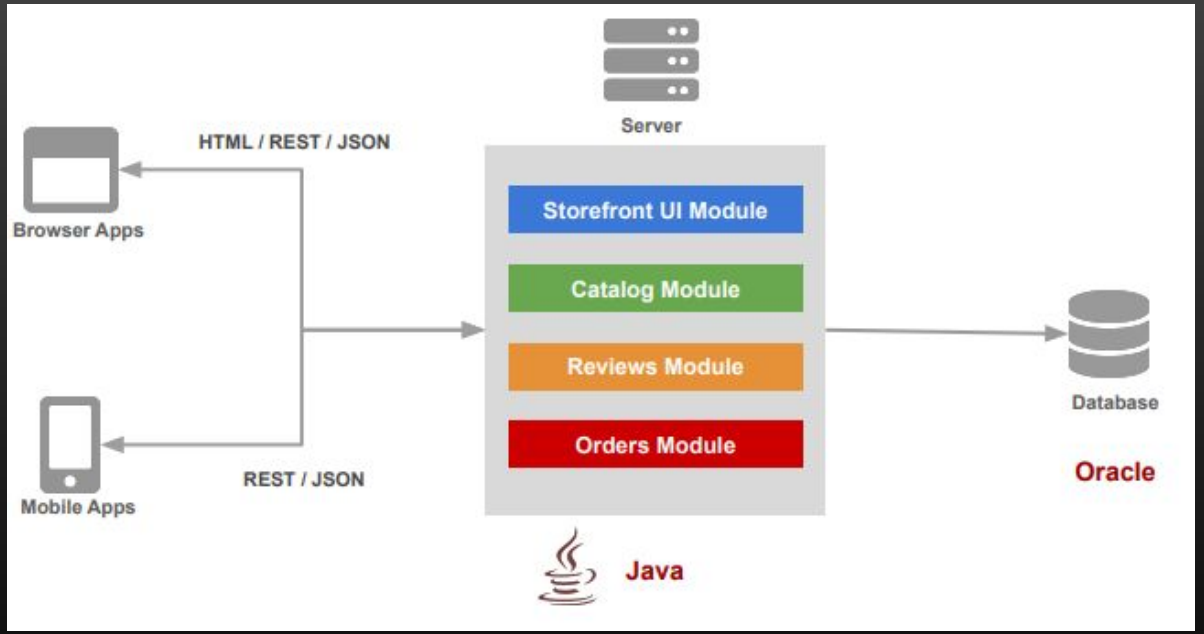
We will mainly focus on :

- Monolithic architecture
- Microservice architecture

Monolithic Architecture

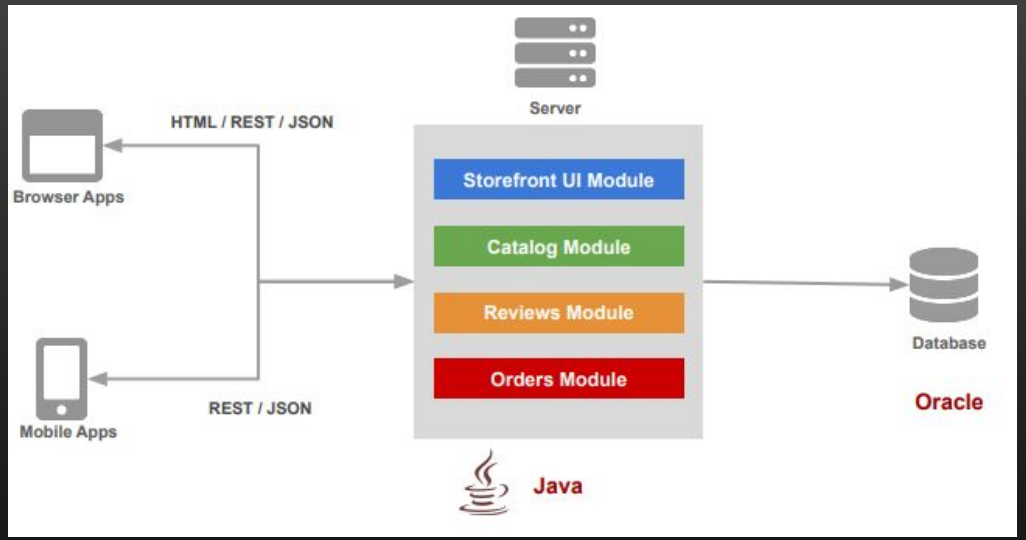
Simple to **Develop**, **Test**, **Deploy** and **Scale**:

1. **Simple** to **develop** because all the tools and IDEs support the applications by default
2. **Easy** to **deploy** because all components are packed into one bundle
3. **Easy** to **scale** the whole application



Monolithic Architecture

1. Very **difficult** to **maintain**
2. One **component failure** will cause the **whole** system to **fail**
3. Very **difficult** to create the **patches** for monolithic architecture
4. Adapting to **new technologies** is **challenging**
5. Take a long **time to startup** because all the components needs to get started



Apps development changed

A decade ago

Apps were monolithic

Built on a single stack (e.e. .NET or Java)

Long lived

Deployed to a single server

Today

Apps are constantly being developed

Build from loosely coupled components

Newer version are deployed often

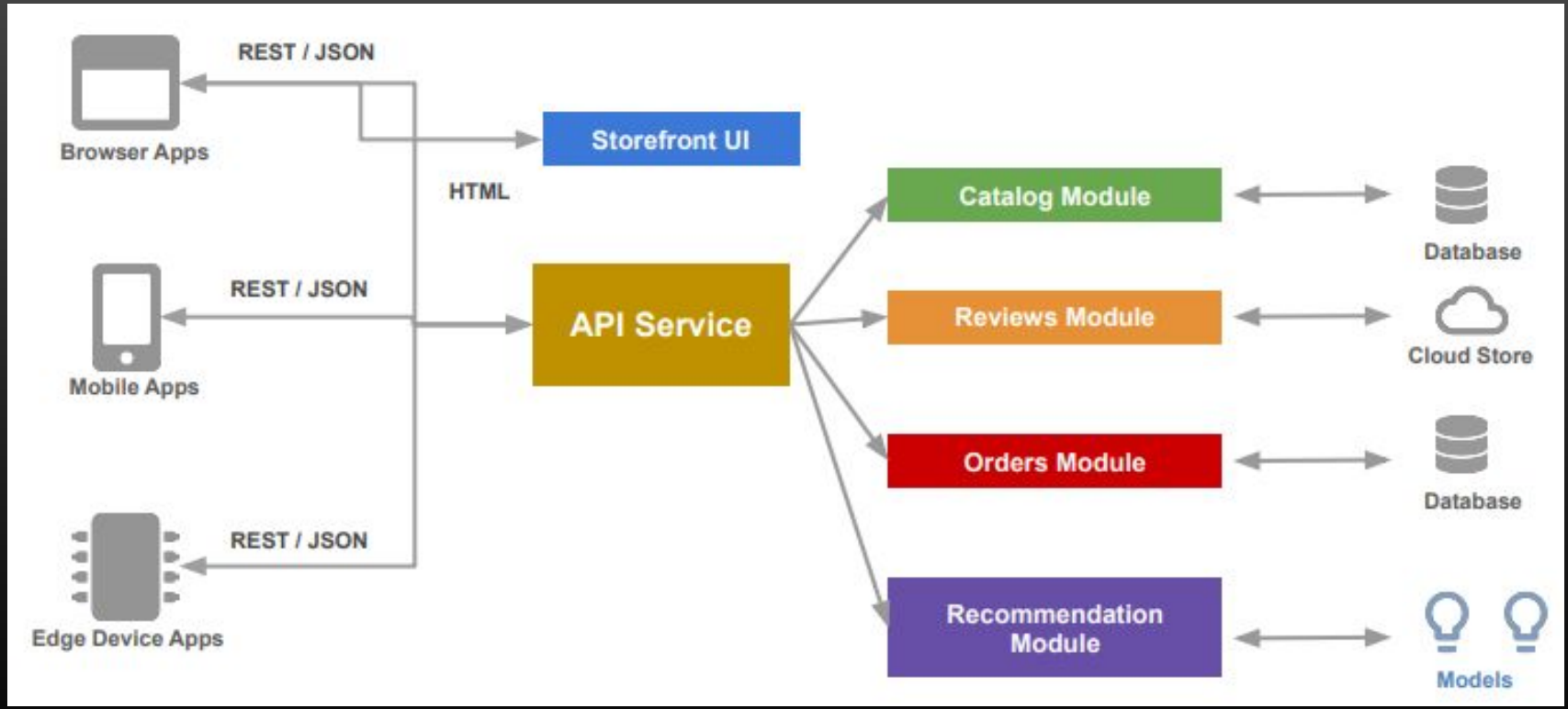
Deployed to a multitude of servers

Data Science & ML

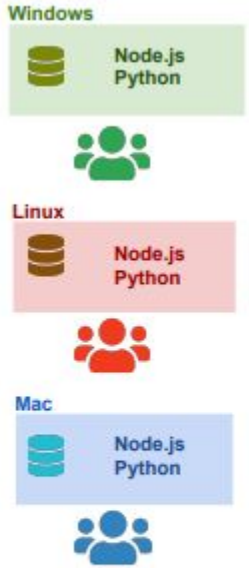
Apps are being integrated with various data types/sources and models

TRANSCEND

Microservice Architecture

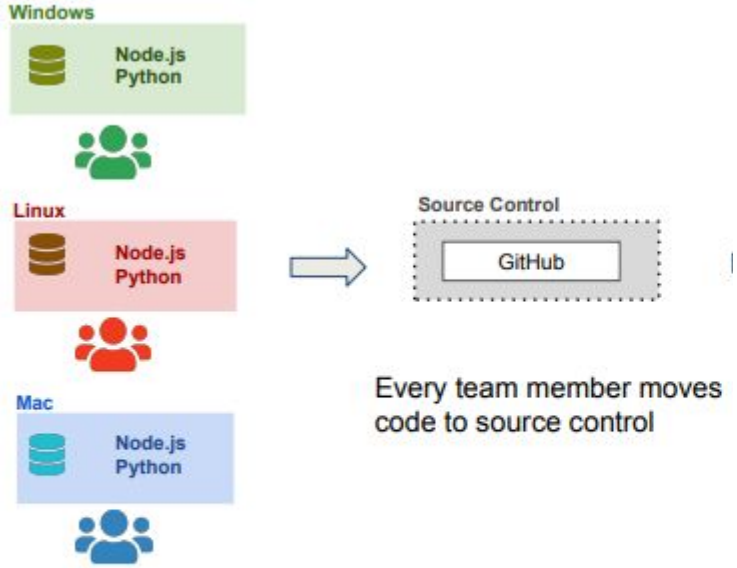


Software Development Workflow (no Docker)

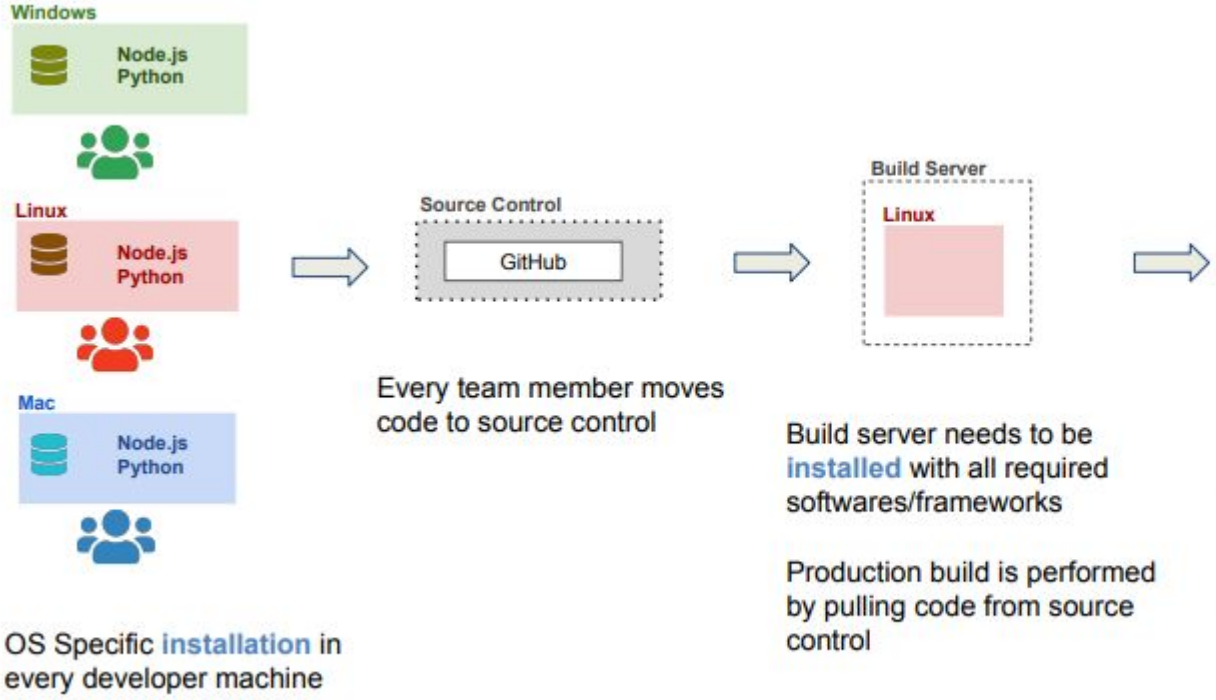


OS Specific **installation** in every developer machine

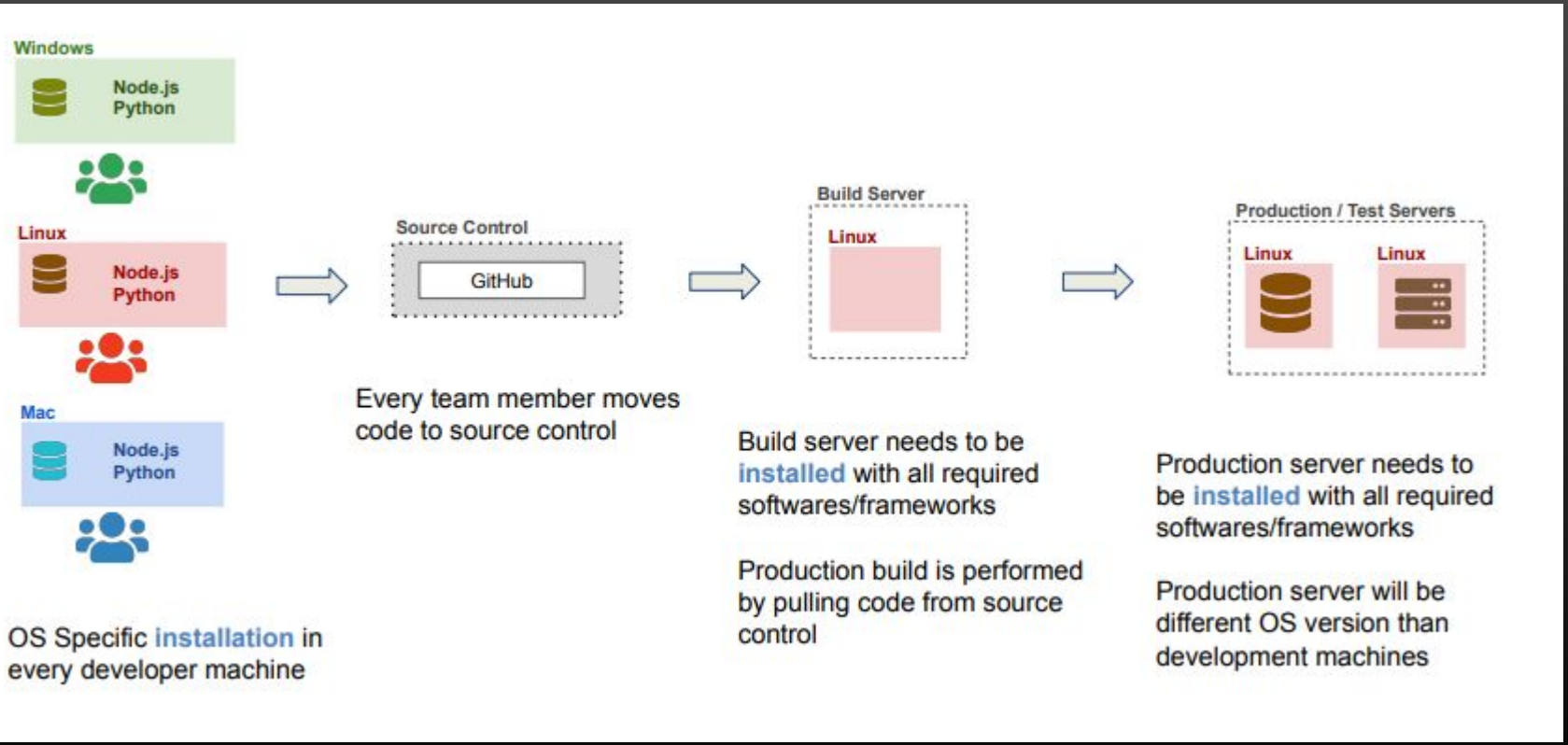
Software Development Workflow (no Docker)



Software Development Workflow (no Docker)



Software Development Workflow (no Docker)

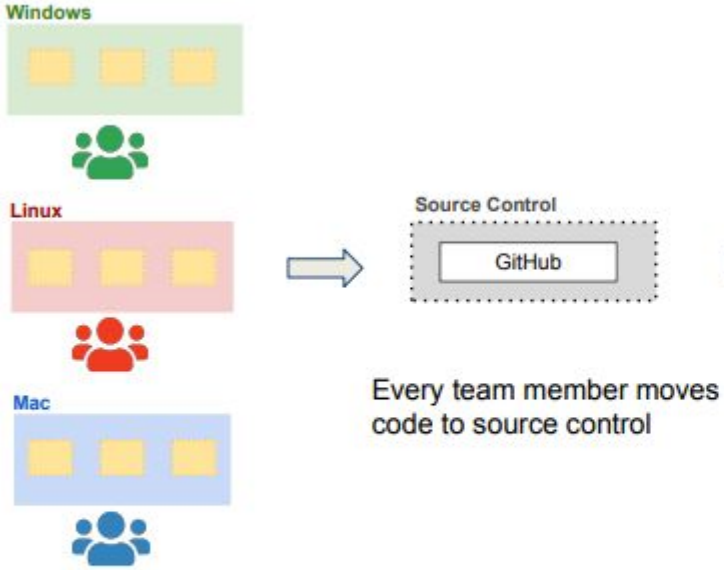


Software Development Workflow (with Docker)

Development machines only needs Docker installed

Containers need to be setup only once

Software Development Workflow (with Docker)

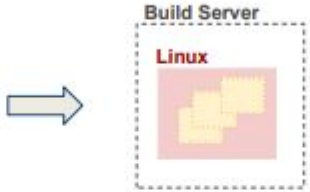


Software Development Workflow (with Docker)

Development machines only needs Docker installed

Containers need to be setup only once

Every team member moves code to source control



Build server only needs Docker installed

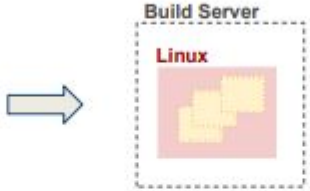
Docker images are built for a release and pushed to container registry

Software Development Workflow (with Docker)

Development machines only needs Docker installed

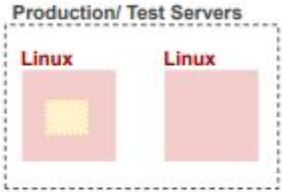
Containers need to be setup only once

Every team member moves code to source control



Build server only needs Docker installed

Docker images are built for a release and pushed to container registry



Production server only needs Docker installed

Production server pulls Docker images from container registry and runs them

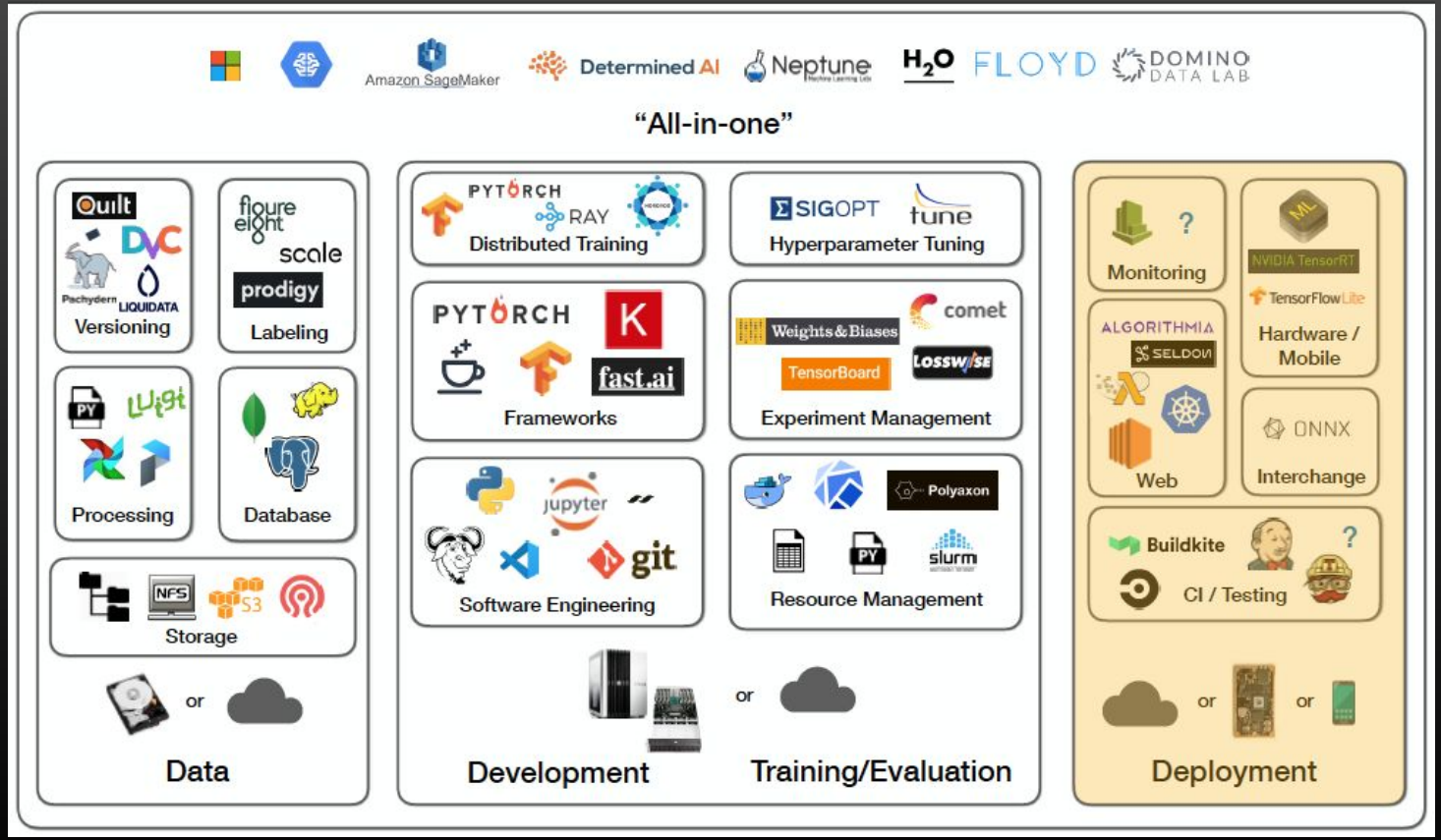
Software Development Workflow (with Docker)

	Virtual environment	Docker	Virtual Machine
Computational cost Memory footprint (without app)	Low	Medium Low	High
Deployment	Easy	Medium	Init high then easy
Versatility (Types of apps)	Medium	Medium High	Medium
Portability	Medium	High	High

Deployment tools and frameworks

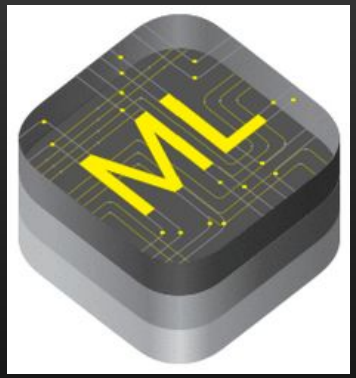
TRANSCEND

AI ecosystem and tools



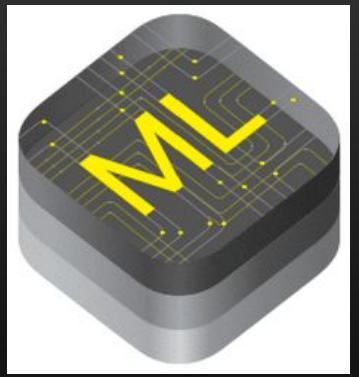
ML frameworks and deployment tools

Machine learning frameworks are tools and libraries that allow developers to more easily build ML models or Machine Learning applications, without having to get into the nuts and bolts of the base or core algorithms. It provides more of an end-to-end pipeline for machine learning development.



DL frameworks and deployment tools

Deep learning frameworks are the machine lear frameworks that are specialized in deep learning applications.



ML Model Serving

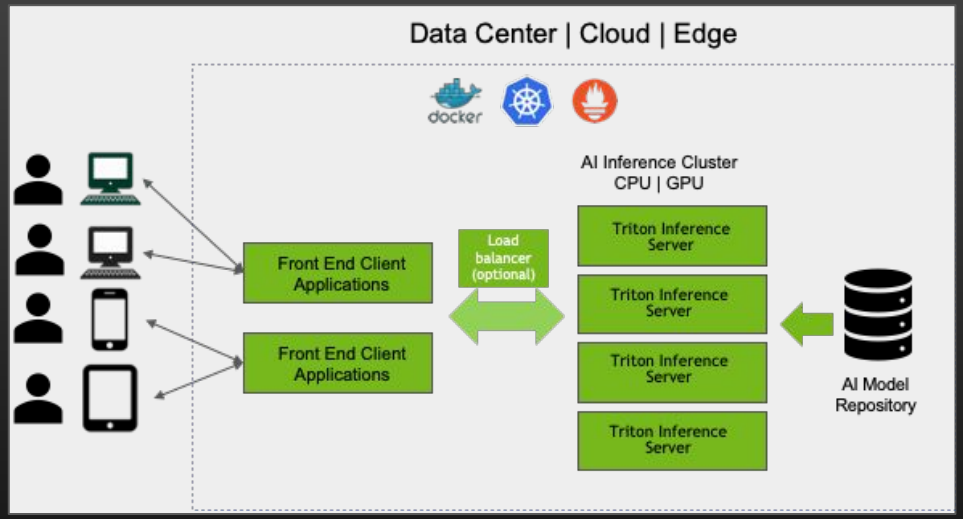
Tools for **model serving** in machine learning can provide you with solutions to many of the data engineers' and DevOp's concerns. They have many functionalities that make it easier to manage your models.

You can use them during the entire lifecycle of your ML project, beginning with building a trained model, to deploying, monitoring, providing easy accessibility, and production. They will automate and optimize your work, but also ensure there are no errors, make it easy to collaborate with others, and track changes in real-time.

ML Model Serving : NVIDIA Triton Inference Server

Triton Inference Server provides an optimized cloud and edge inferencing solution. It's optimized for both CPUs and GPUs. Triton supports an HTTP/REST and GRPC protocol that allows remote clients to request inferencing for any model being managed by the server.

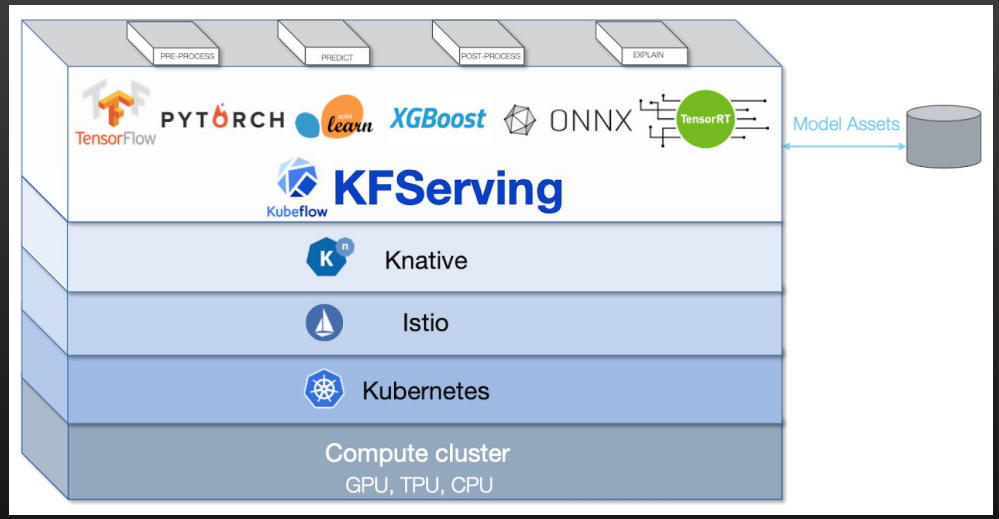
- Supports multiple deep-learning frameworks (TensorRT, TensorFlow GraphDef, TensorFlow SavedModel, ONNX, and PyTorch TorchScript)
- Simultaneous model execution on the same GPU or on multiple GPUs
- Manages stream and batch deployments
- Dynamic batching
- Extensible backends
- Supports model ensemble
- Metrics in Prometheus data format indicating GPU utilization, server throughput, and server latency



ML Model Serving : KFServing

It aims to solve production model serving use cases by providing performant, high abstraction interfaces for common ML frameworks like Tensorflow, XGBoost, ScikitLearn, PyTorch, and ONNX.

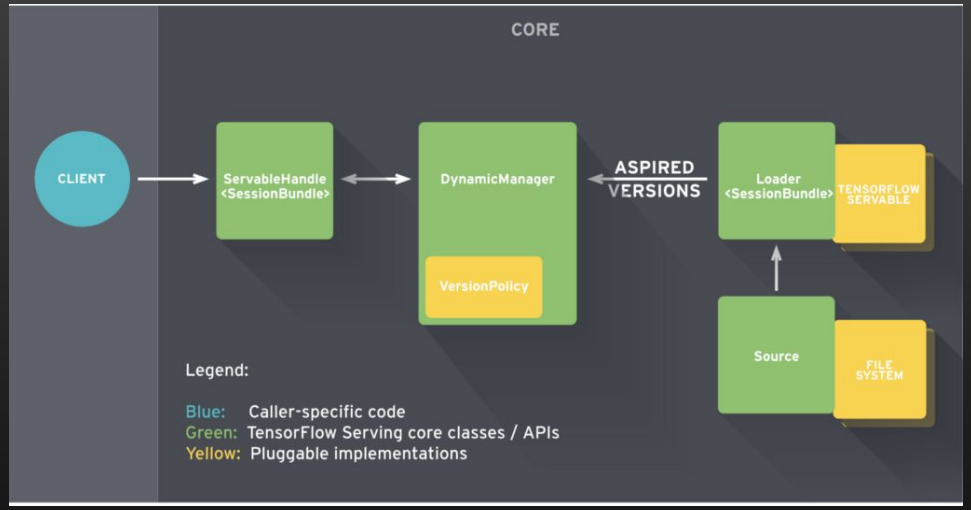
- Provides a simple, pluggable, and complete story for your production ML inference server by providing prediction, pre-processing, post-processing and explainability
- Customizable Inference Service to add your resource requests for CPU, GPU, TPU and memory requests and limits
- Batching individual model inference requests
- Traffic management
- Scale to and from Zero
- Revision management
- Request/Response logging
- Scalable Multi Model Serving



ML Model Serving : TensorFlow Serving

TensorFlow Serving is a flexible system for machine learning models, designed for production environments. It deals with the inference aspect of machine learning.

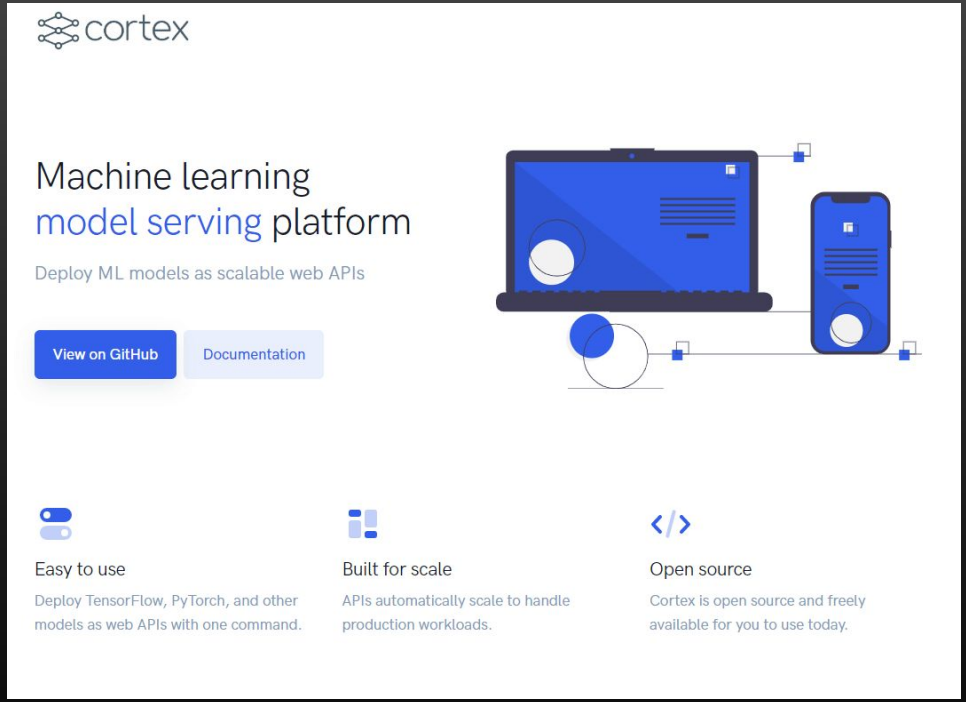
- Can serve multiple models, or multiple versions of the same model at the same time
- Exposes both gRPC and HTTP inference endpoints
- Allows deployment of new model versions without changing your code
- Lets you flexibly test experimental models
- Its efficient, low-overhead implementation adds minimal latency to inference time
- Supports many servables: Tensorflow models, embeddings, vocabularies, feature transformations, and non-Tensorflow-based machine learning models



ML Model Serving : Cortex

Cortex is an open-source platform for deploying, managing, and scaling machine learning models. It's a multi framework tool that lets you deploy all types of models.

- Automatically scale APIs to handle production workloads
- Run inference on any AWS instance type
- Deploy multiple models in a single API and update deployed APIs without downtime
- Monitor API performance and prediction results

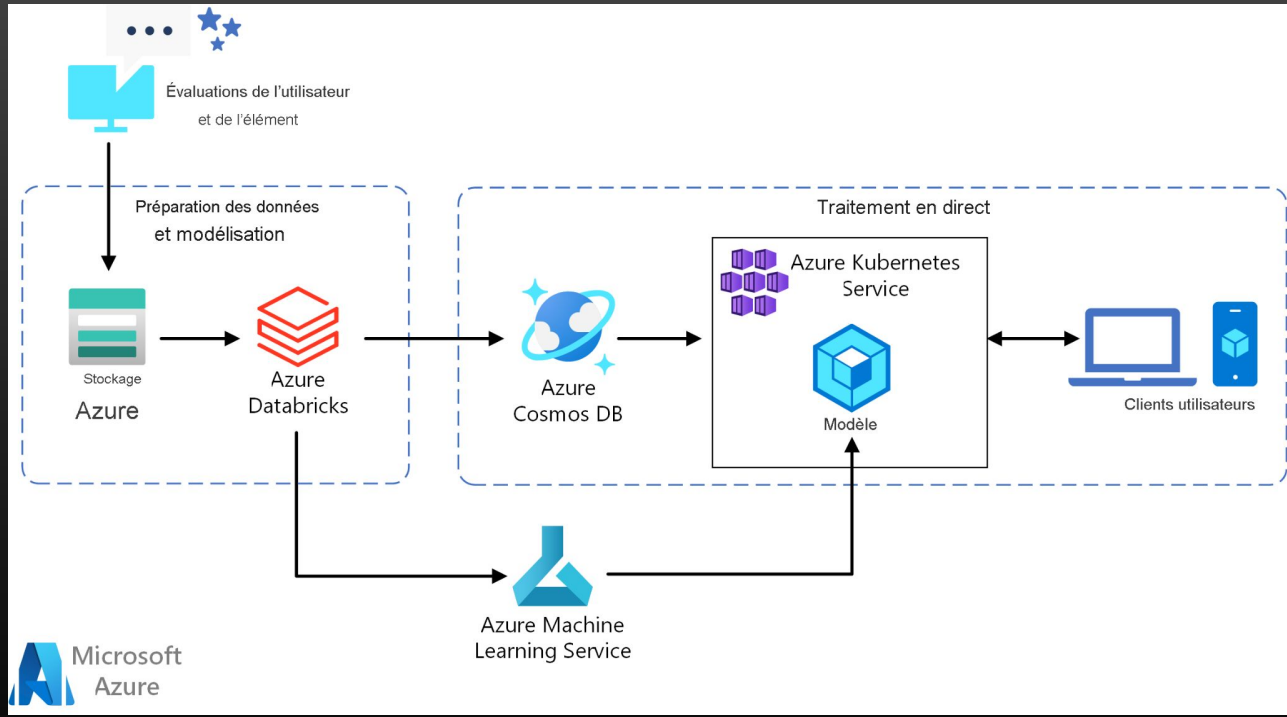


TRANSCEND

ML Model Azure

Azure (and other cloud providers) do not provide a unified ML serving server.

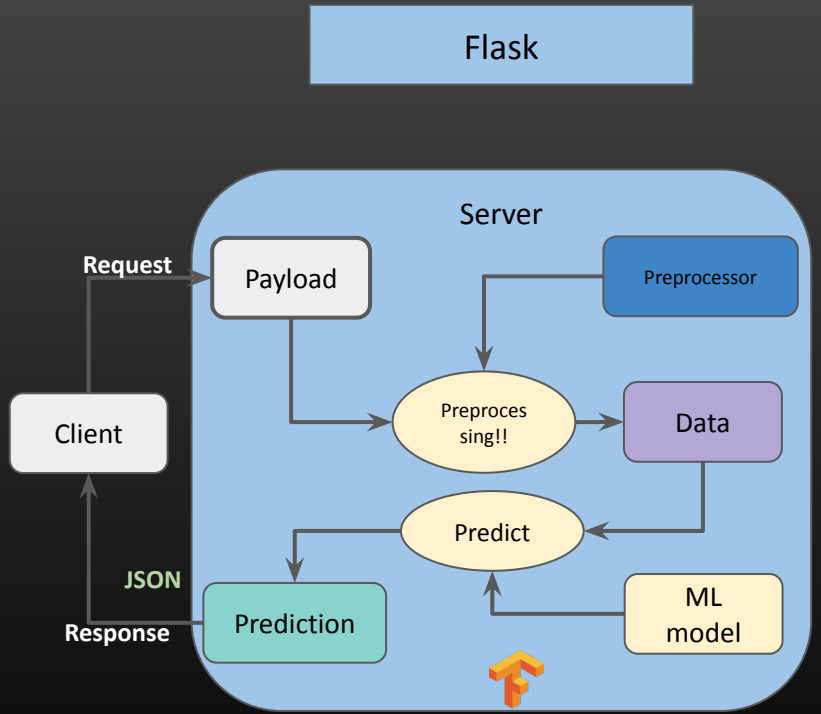
We need to develop and architecture for the serving service based on the cloud provider services



Flask

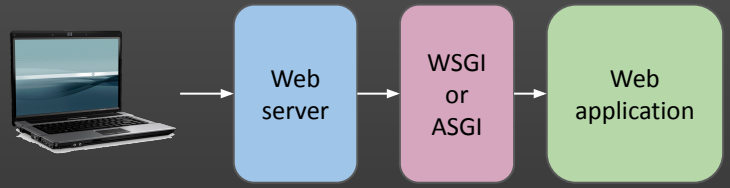
Deploy ML as a web service- Flask-WSGI (Web Service Gateway Interface)

- **What is Flask**
  - deploy ML models as web services quickly and easily
- **How:**
  - Train ML/DL model (data, train, validation), Save model to file ".pkl"
  - Create new flask application define routes
  - Use the flask run command to run the application on your development server
  - Deploy the application to a server or hosting service, such as a virtual private server (VPS) or a platform-as-a-service (PaaS) provider.
  - **Optional :** Containerize the application using Docker and deploy it to a container orchestration platform like Kubernetes



Flask

Project: Deploy ML as a web service using Flask- WSGI



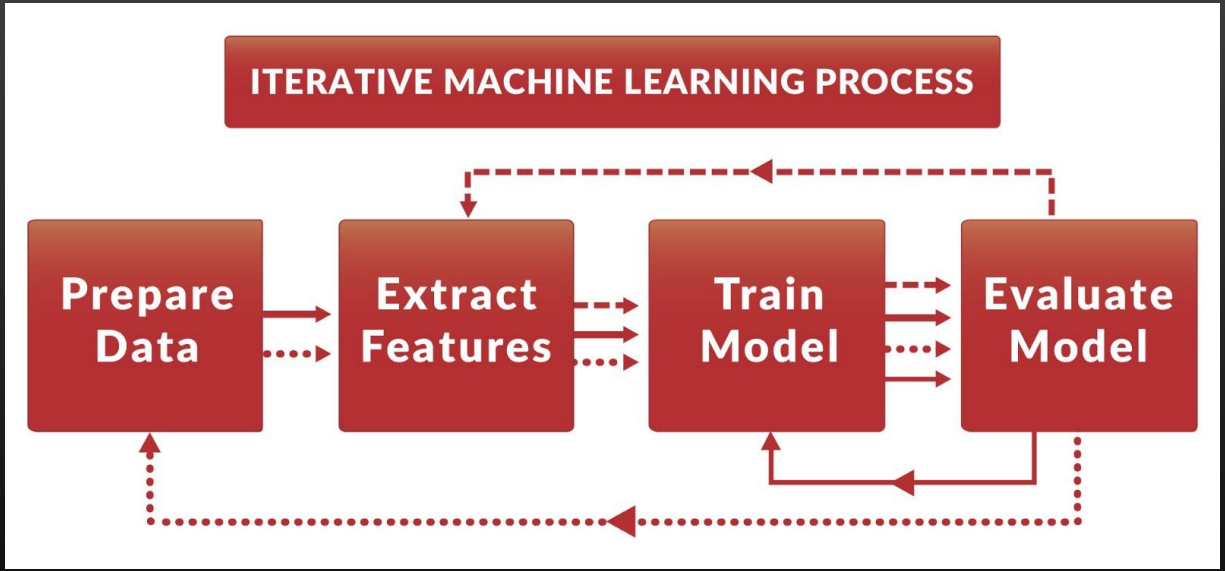
- Create your own account on github if you have not
- Clone the repo into your local machine: https://github.com/mymehio/deploy_flask_01.git
- Locally you will have a repo of the following files
  - Hiring.csv: a dataset of training
  - Model.py a script to train model using dataset
  - App.py a script to define API services using http requests. It's a creation of web app
  - templates/index.html: here you define the template that you want in html to request and predict your inputs
  - Model.pkl: the already trained model (if you don't want to train from the scratch)
- Copy all files to a new folder created in your PC
- Now, commit and push your files into your own github, after creating your repository
  - git init
  - git add . (to add all files)
  - git commit -m "message"
  - git remote add origin https://github.com/mymehio/repo_name.git
  - git push -u origin main
- Create a free account on server: Render, ~~heroku~~: are cloud based platforms as a service (PAAS) that allow easily to deploy, run and manage applications
- In PAAS, setup the launch command: **gunicorn** app:app
  - **gunicorn** : python lib for running web service in **WSGI** (web server Gateway Interface)
- Increase your dataset, retrain, push to repo, update web service

Model management and version control

Version control for ML

For AI models, we need to manage :

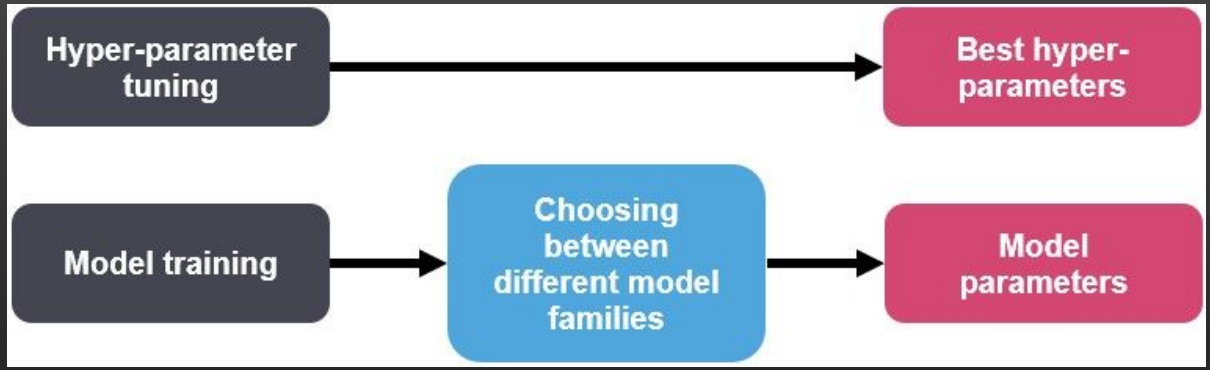
- Code
- Data
- Model

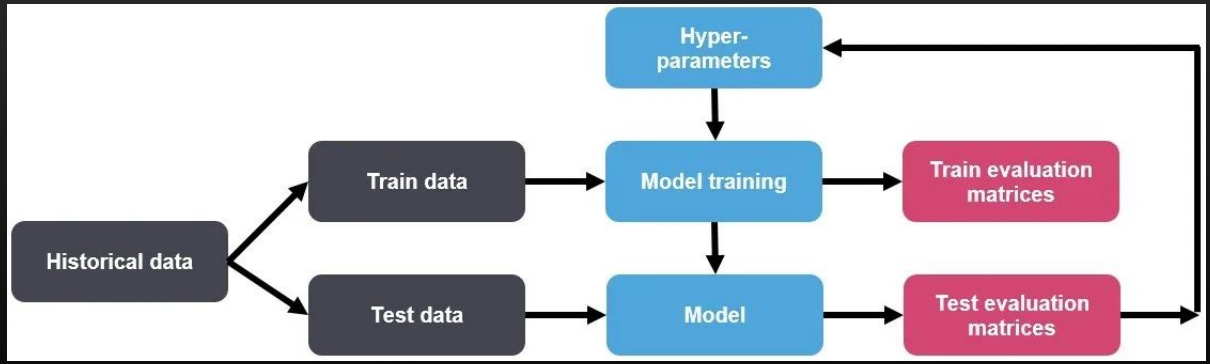


Version control for ML

The model connects all of the above with model parameters and hyperparameters.

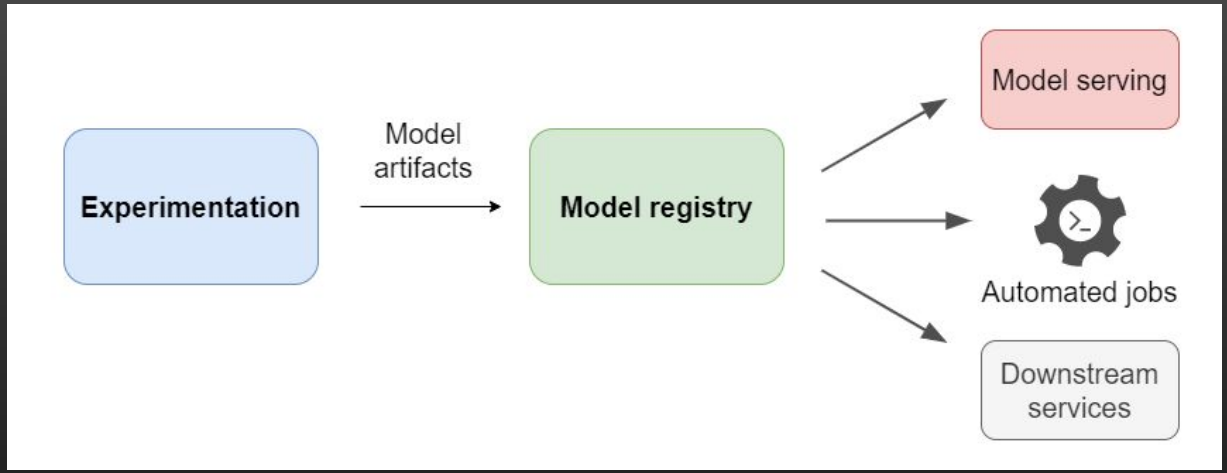
- Collaboration
- Versioning
- Reproducibility
- Dependency tracking
- Model updates





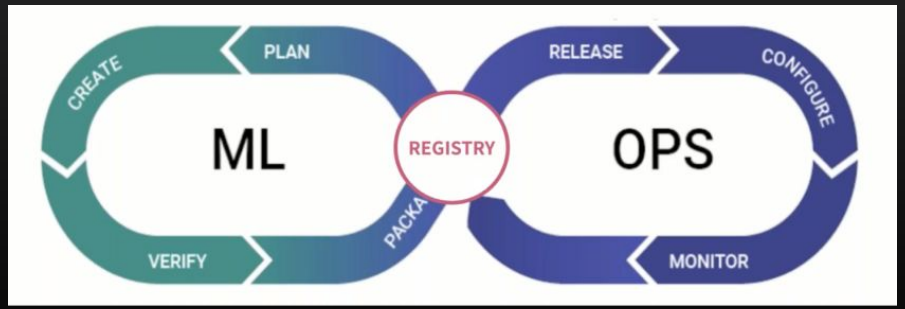
TRANSCEND

AI Model registry



The model registry provides:

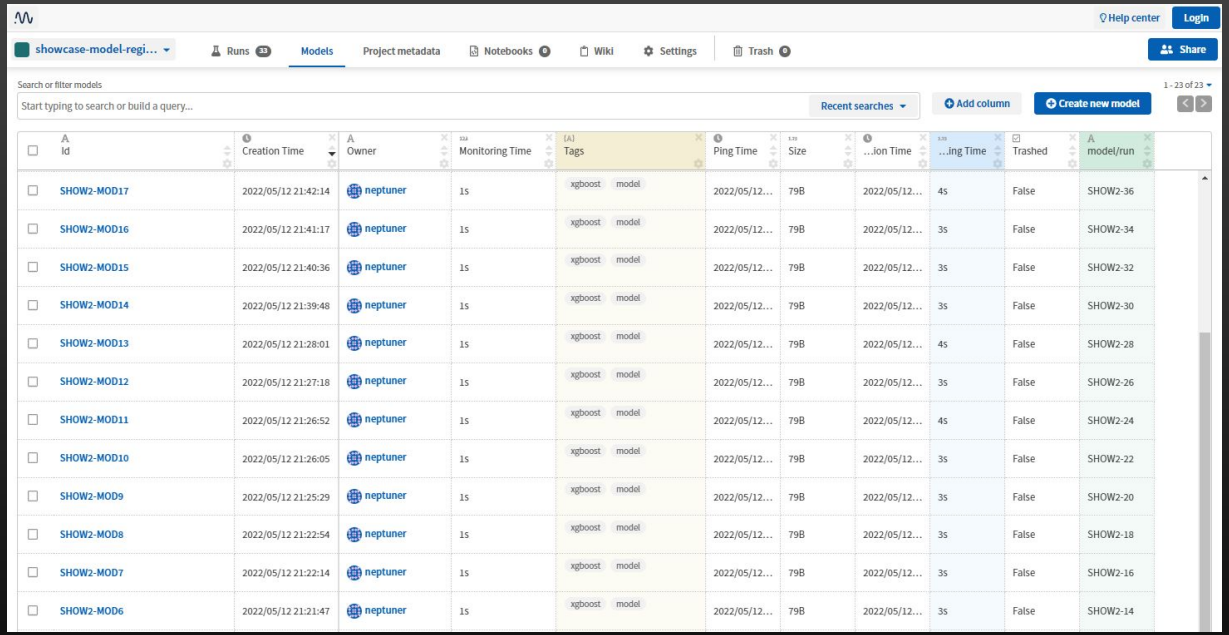
- Centralized storage for all types of models,
- And a collaborative unit for model lifecycle management.



AI Model registry : neptune.ai

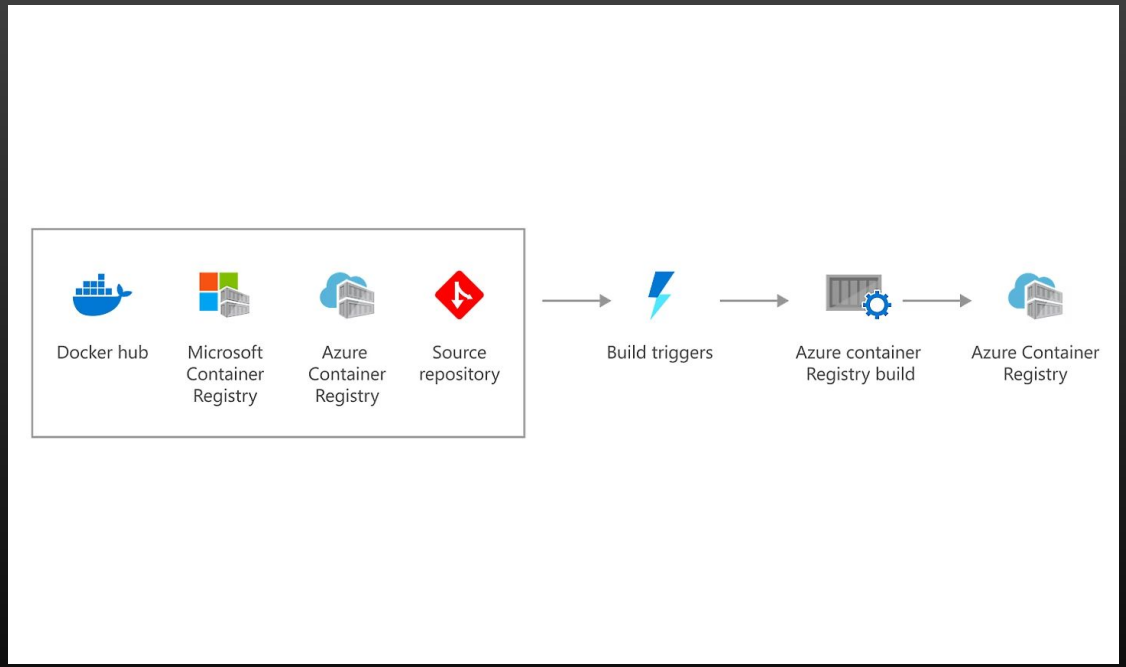
The model registry provides:

- Centralized storage for all types of models,
- And a collaborative unit for model lifecycle management.



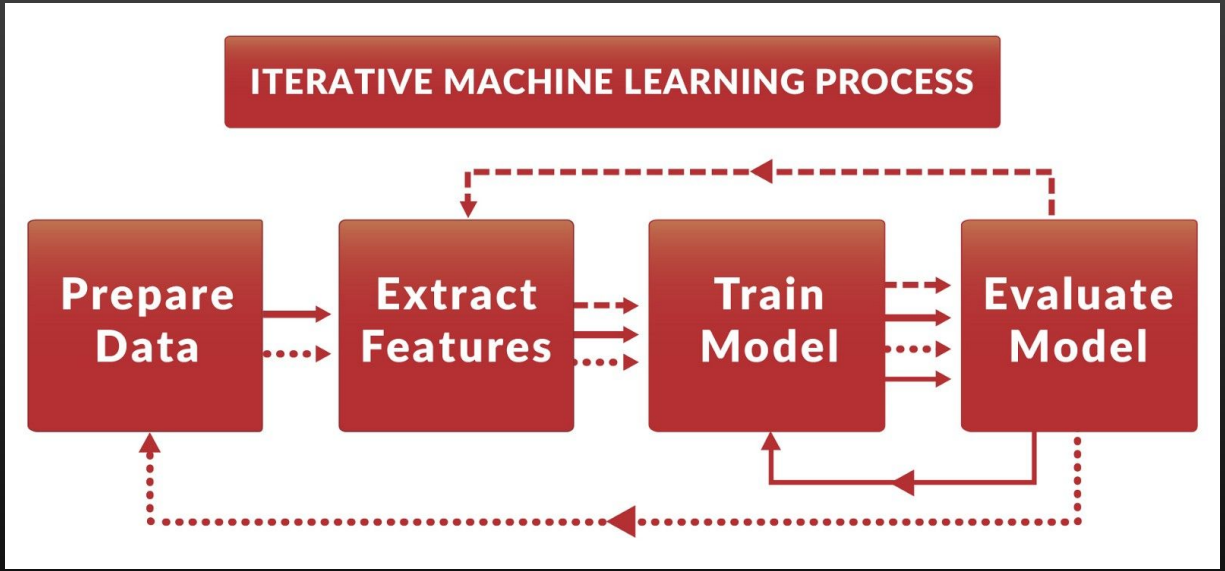
AI Model registry : Azure container registry

- **Geo-replication** to efficiently manage a single registry across multiple regions
- **OCI artifact repository** for adding Helm charts, Singularity support, and new OCI artifact-supported formats
- **Automated container building and patching** including base image updates and task scheduling
- **Integrated security** with Azure Active Directory (Azure AD) authentication, role-based access control, Docker Content Trust, and virtual network integration

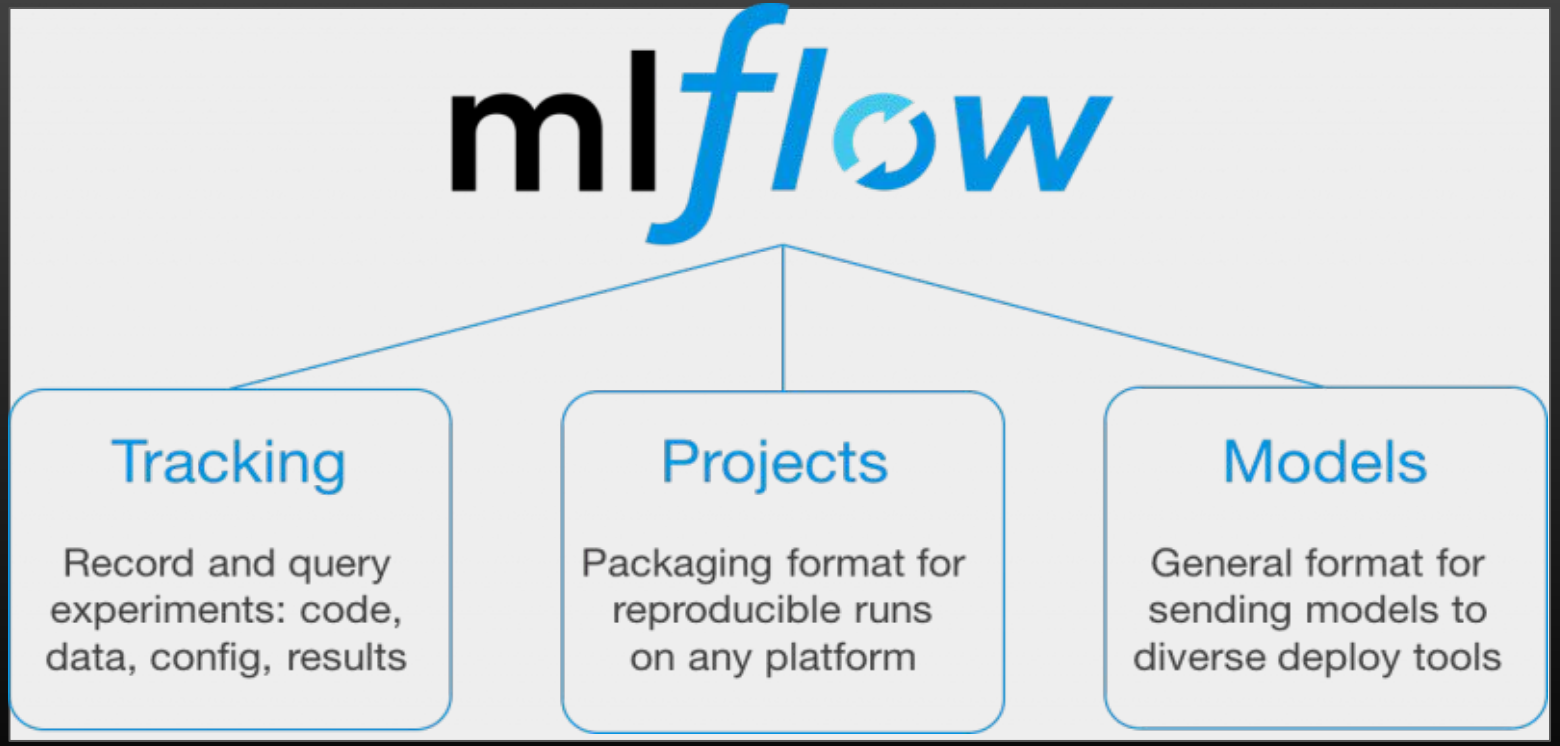


Version control for ML

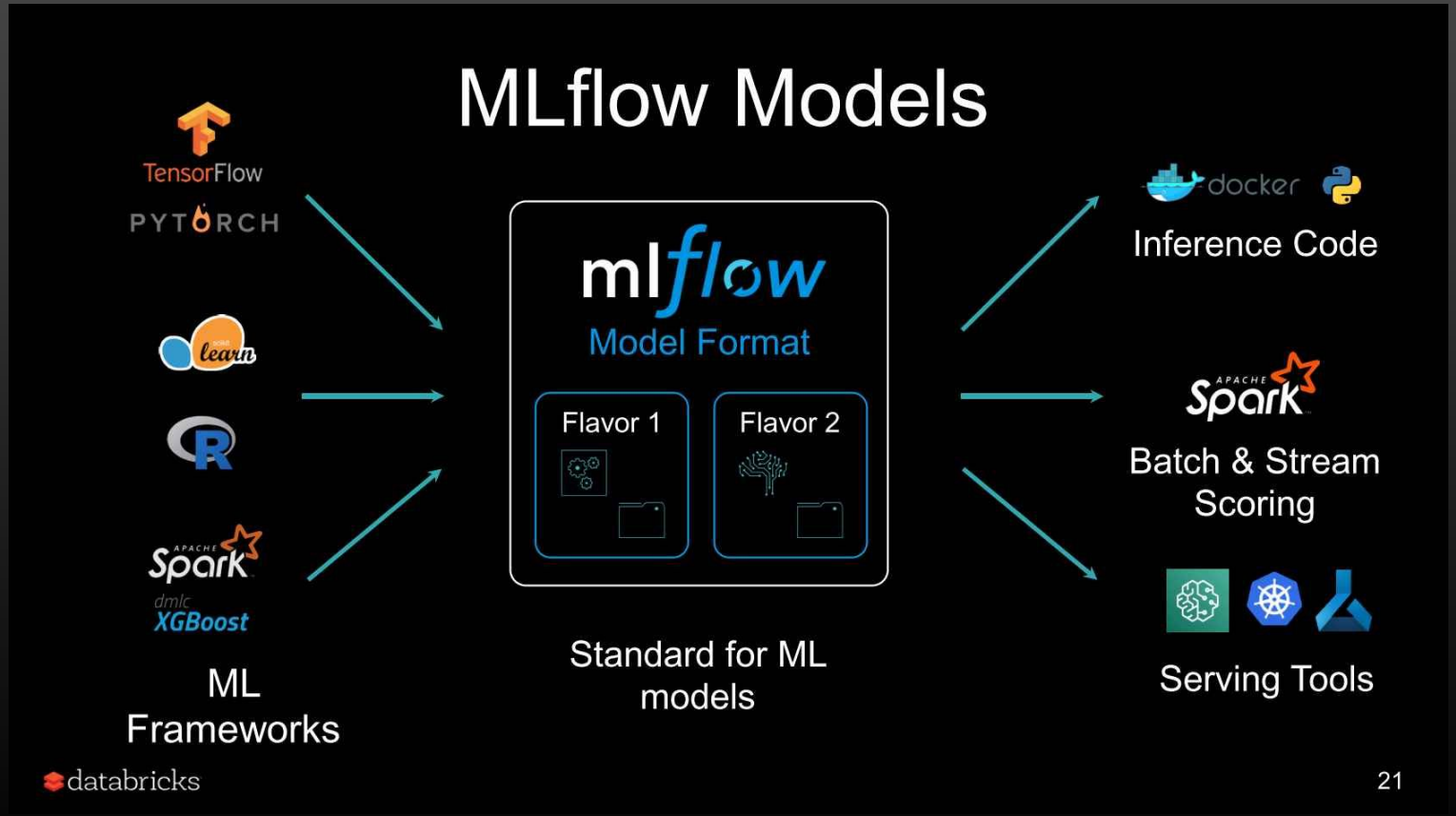
MLOps tools provide the features necessary for the management of ML pipelines



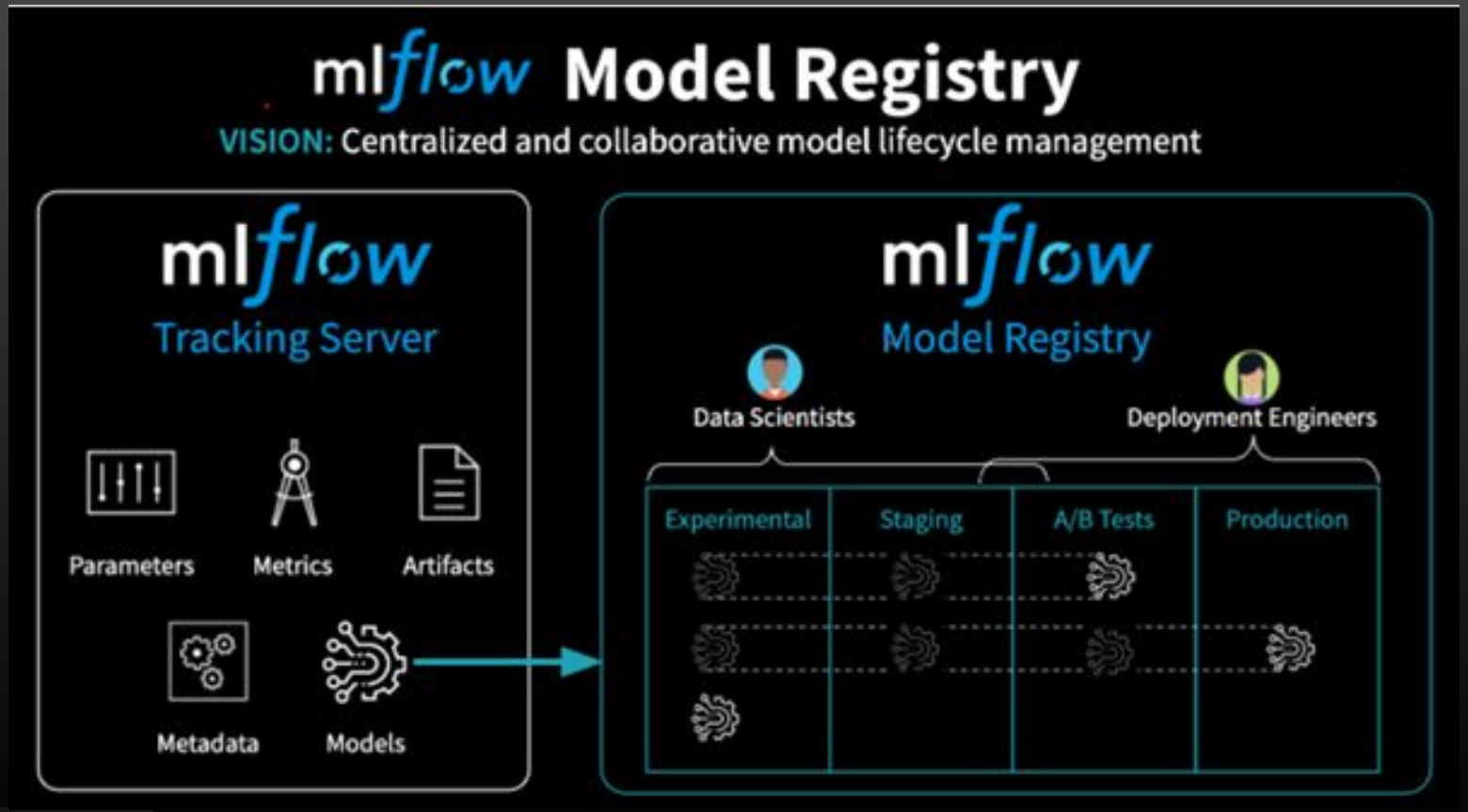
MLflow : A platform of the machine learning lifecycle



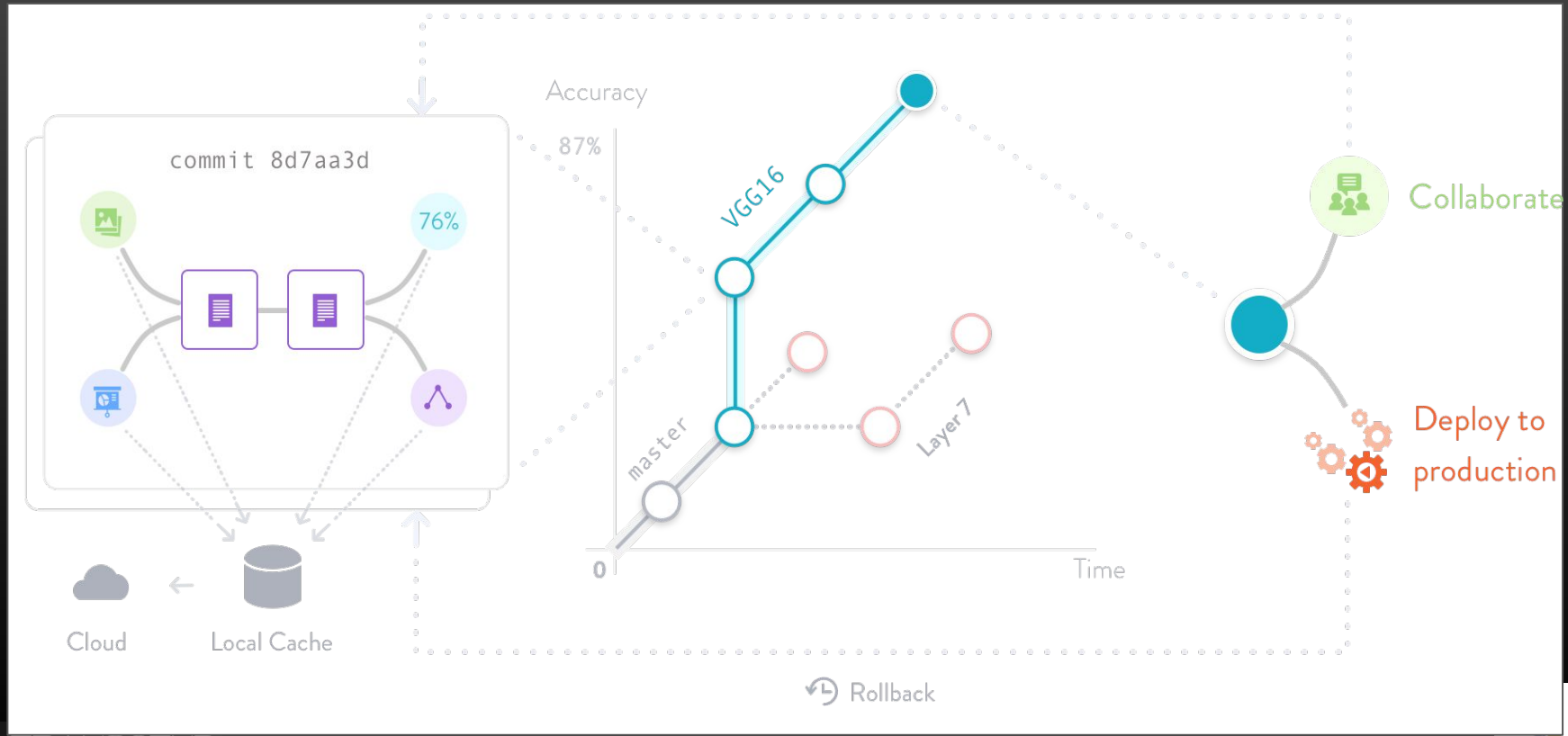
MLflow : A platform of the machine learning lifecycle



MLflow : A platform of the machine learning lifecycle



(Not Just) Data Version Control (DVC)



(Not Just) Data Version Control (DVC)

ML project version control

Version control machine learning models, data sets and intermediate files. DVC connects them with code, and uses Amazon S3, Microsoft Azure Blob Storage, Google Drive, Google Cloud Storage, Aliyun OSS, SSH/SFTP, HDFS, HTTP, network-attached storage, or disc to store file contents.

Full code and data provenance help track the complete evolution of every ML model. This guarantees reproducibility and makes it easy to switch back and forth between experiments.

ML experiment management

Harness the full power of Git branches to try different ideas instead of sloppy file suffixes and comments in code. Use automatic metric-tracking to navigate instead of paper and pencil.

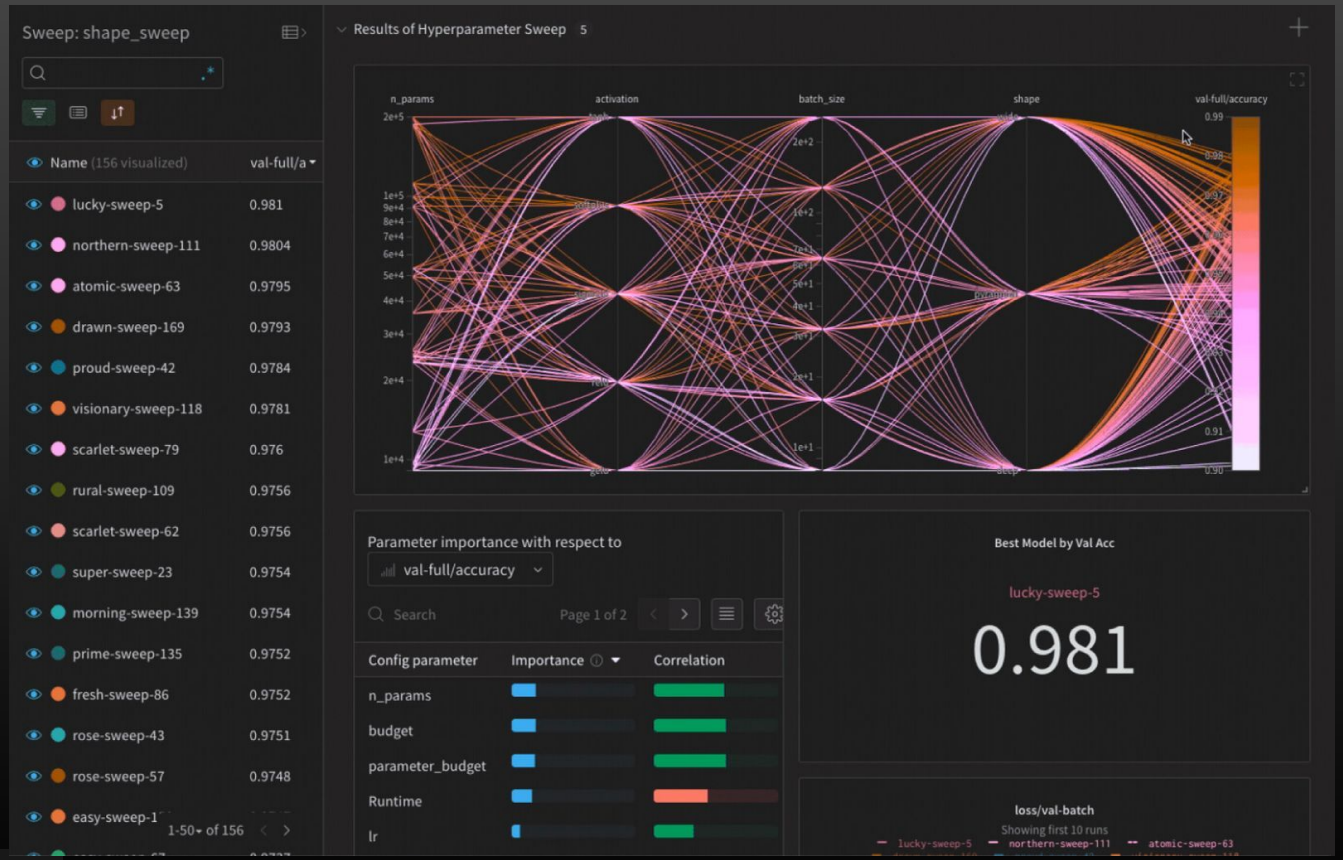
DVC was designed to keep branching as simple and fast as in Git — no matter the data file size. Along with first-class citizen metrics and ML pipelines, it means that a project has cleaner structure. It's easy to compare ideas and pick the best. Iterations become faster with intermediate artifact caching.

Deployment & Collaboration

Instead of ad-hoc scripts, use push/pull commands to move consistent bundles of ML models, data, and code into production, remote machines, or a colleague's computer.

DVC introduces lightweight pipelines as a first-class citizen mechanism in Git. They are language-agnostic and connect multiple steps into a DAG. These pipelines are used to remove friction from getting code into production.

Wandb.ai The developer-first MLOps platform



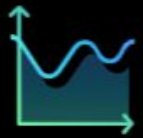
Picsellia Plateforme CVOps

DU DEVELOPPEMENT AU DEPLOIEMENT

Un chaîne MLOps Complète

Pipelines automatisés

Gestion des données

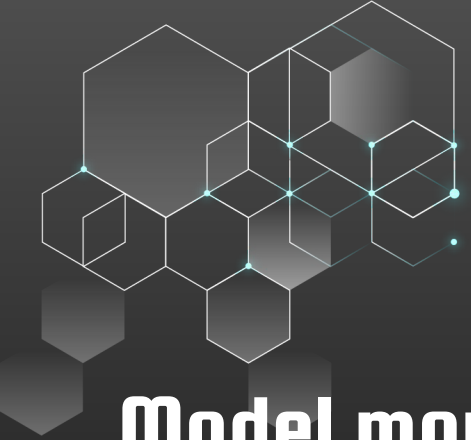


Expérience
Suivi

Modèle
Déploiement

Modèle
Surveillance

Model monitoring and and performance tuning



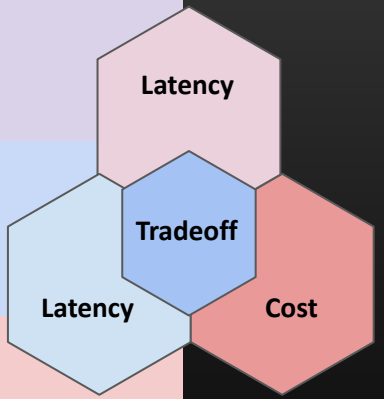
TRANSCEND

ML in production

Model Optimization

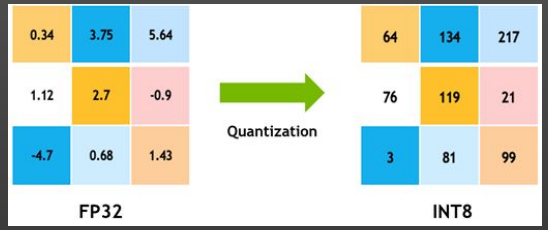
- Model architecture vs performance and complexity
- Optimization; the trade-offs between latency, throughput, and cost.

- Optimizing for latency
  - Refers to the time consumed to predict (inference)
  - Reduce the parameters number
  - Use lightweight models such as MobileNet
- Optimizing for Throughput
  - Refers to the number of predictions (inference) per second
  - Use the hardware accelerator, GPU, TPU
  - Use model parallelism to split the model across multiple devices
- Optimizing for Cost
  - Refers to the resources required to run the model (computing, memory, storage .)
  - Use the cloud based services (GCP, AWS..) t
  - Compress the model to reduce the size of the model and lower storage costs



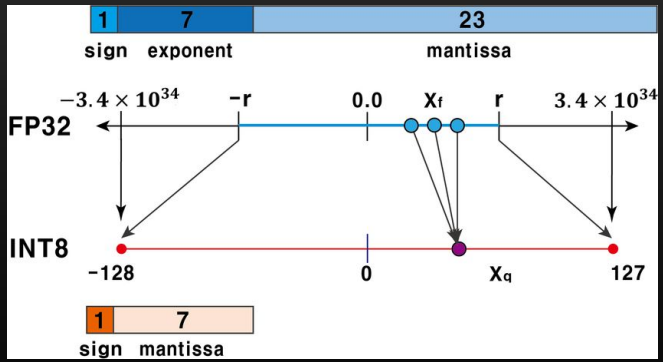
ML in production

Quantization: Tensorflow as a use case



- Quantization is a strategy for reducing the size and computational cost of a machine learning model
  - Finite number of bits represent a huge real numbers. Representation is decided by the number of bit. 32 bit floating point is in most of application
  - Weights are of 32 bit by defaults . Can they be of lower precision ?
  - The numbers are quantized (i.e. discretized) to some specific values, which we can then represent using integers
  - This results in a smaller model size, better portability and faster computation

- **Weight ranges:** Weight distribution for AlexNet shows how most values are concentrated in small ranges
- **Latency:** severely reduces the computation to run interference (arithmetic operations are slower on 32fp)



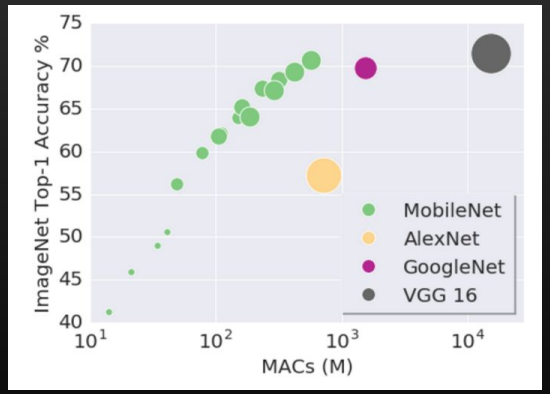
ML in production

Why Quantization in ML/DL

- Deep Learning Models has a trend of increasing complexity to give good results
- Models need to be deployed on embedded and Mobile devices which also put constraints on memory
- To reduce the footprint (Execution time and Memory) of neural network models to address these challenges

Network	Model size (MB)	GFLOPS
AlexNet*	233	0.7
VGG-16*	528	15.5
VGG-19*	548	19.6
ResNet-50*	98	3.9
ResNet-101*	170	7.6
ResNet-152*	230	11.3
GoogleNet#	27	1.6
InceptionV3#	89	6
MobileNet#	38	0.58
SequeezeNet#	30	0.84

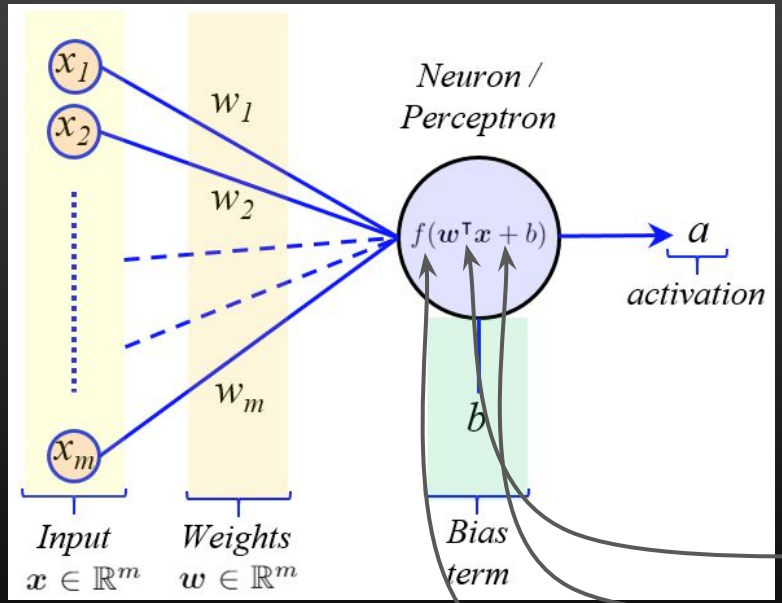
*: Characterization and Benchmarking of Deep Learning, Natalia Vassilieva
#: https://github.com/albanie/convnet-burden



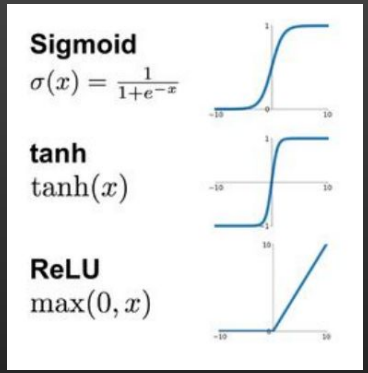
Model MACs and Predicate Accuracy of Networks. (MAC: Multiply ACcumulate)

ML in production

What to quantize



Possible Activation Functions:



*** Activation Functions are Non-Linear**

Operations:

- Matrix Multiplication
- Addition
- Non-Linear Function

TRANSCEND

ML in production

Quantization Calculation

- Given a Tensor (N-D Array) T for example: Tf = [0.1, 0.2, 0.3] where T range is [0, 0.5]
- Quantized tensor q_Tf in t_uint8 [0, 255] can be derived as follows:
  - q_Tf = round((1 / q_Tf .scale) * Tf - q_Tf .zero_point)
  - Where:
    - q_Tf.scale = (Tf .max - Tf .min) / (t_uint8.max - t_uint8.min) a scaling factor that determines the range of quantized values
    - q_Tf.zero_point = t_uint8.max- (Tf .max / q_Tf .scale)
  - Thus, in this case:
    - q_Tf.scale = ?
    - q_Tf.zero_point = ?
    - q_Tf = ?

ML in production

Quantization Calculation

- Given a Tensor (N-D Array) T for example: Tf = [0.1, 0.2, 0.3] where T range is [0, 0.5]

- Quantized tensor q_Tf in t_uint8 [0, 255] can be derived as follows:

    - q_Tf = round((1 / q_Tf .scale) * Tf - q_Tf .zero_point)
    - Where:
        - q_Tf .scale = (Tf .max - Tf .min) / (t_uint8.max - t_uint8.min) a scaling factor that determines the range of quantized values
        - q_Tf .zero_point = t_uint8.max- (Tf .max / q_Tf .scale)
    - Thus, in this case:
        - q_Tf.scale = (0.5 - 0) / (255 - 0) = 0.0019607843137255
        - q_Tf.zero_point = 255 - (0.5 / 0.00196) = 0
        - q_Tf = [51, 102, 153]

ML in production

Quantization Techniques

- Post-Training Quantization (PTQ)
  - after training, where the weights of the model are quantized from 32bit to 8 bits or any lower precision
  - No need to training
- Quantization Aware Training (QAT)
  - the model is trained with the knowledge that it will be quantized to perform well with low precision data
  - May lead to better accuracy

Model	Non-quantized Top-1 Accuracy	8-bit Quantized Accuracy
MobilenetV1 224	71.03%	71.06%
Resnet v1 50	76.3%	76.1%
MobilenetV2 224	70.77%	70.01%
Nasnet-Mobile	74%	73%
Resnet-v2 50	75.6%	75%

TRANSCEND

Deployment in DL Model

Project: Quantization Techniques

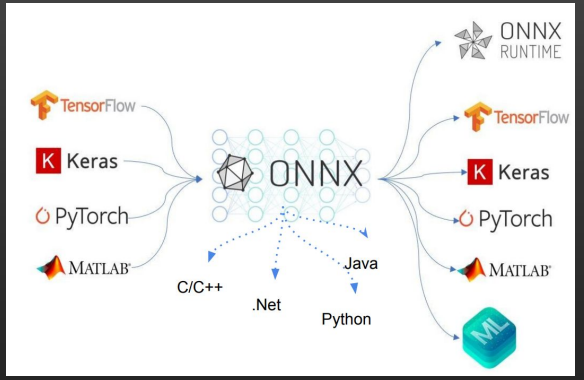
- Using tensorflow library we do the quantization tasks easily
- Your project is to develop a DL model from the scratch, then you quantize it by applying practically both techniques you learned
- Then you will be able to evaluate quantized models and compare them according to :
    - Complexity
    - Accuracy
    - Size

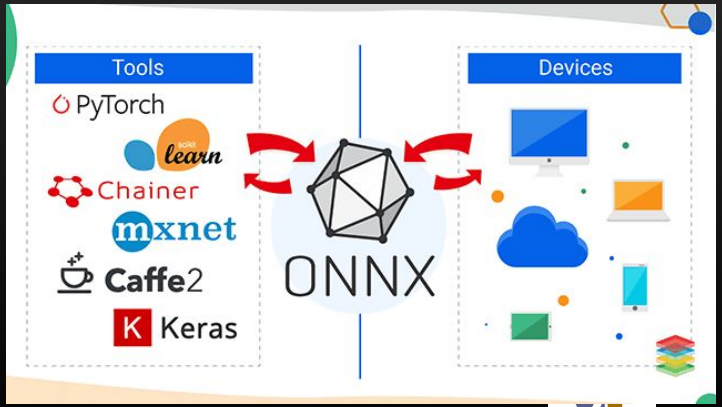
- **<u>In this practical project</u>**

Deployment in DL Model

ONNX Format

- The Open Neural Network Exchange (ONNX) format is a popular open-source standard for representing deep learning models. It offers:
  - ONNX models can be run on various deep learning frameworks
  - ONNX supports a wide range of neural network architectures
  - ONNX is an active community-driven project with regular updates and contributions from major deep learning domain
  - It provides a standard format for exporting models from training environments and deploying them to inference environments (cloud, devices, on premise…..)





Deployment in DL Model

ONNX Format - project

- **In this practical project**, you will learn to convert a pre-trained YOLO model to ONNX format for improved portability and inference speed
- You will then deploy the model on either an edge device or a cloud-based platform for object detection tasks
- You will also gain experience optimizing the model for their chosen deployment environment and evaluating its performance
- The project provides valuable skills for real-world applications and insights into the challenges of deploying machine learning models
- Useful application including the ONNX utility for deploying models

- **In this practical project**

TRANSCEND

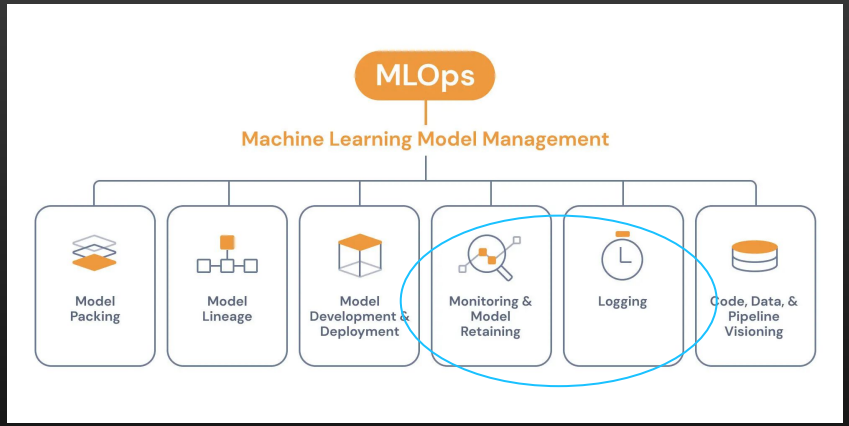
Goodies

- Project to deploy ONNX model in C++ Good stuff: https://learnopencv.com/object-detection-using-yolov5-and-opencv-dnn-in-c-and-python/

- Matlab stuff: https://fr.mathworks.com/help/deeplearning/ug/inference-comparison-between-onnx-and-imported-network-for-image-classification.html

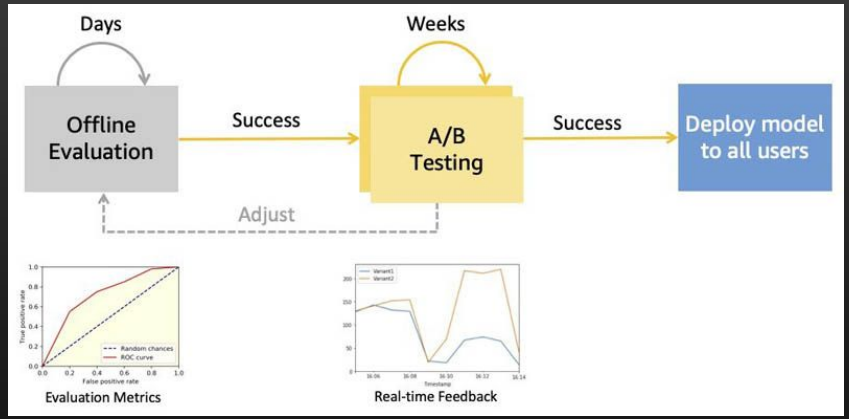
Monitoring deployed models

- **Monitor model performance in production:** see how accurate the predictions of your model are. See if the model performance decays over time, and you should re-train it.
- **Monitor model input/output distribution:** see if the distribution of input data and features that go into the model changed? Has the predicted class distribution changed over time? Those things could be connected to the data and concept drift.
- **Monitor model training and re-training:** see learning curves, trained model predictions distribution, or confusion matrix during training and re-training.
- **Monitor model evaluation and testing:** log metrics, charts, prediction, and other metadata for your automated evaluation or testing pipelines
- **Monitor hardware metrics:** see how much CPU/GPU or Memory your models use during training and inference.



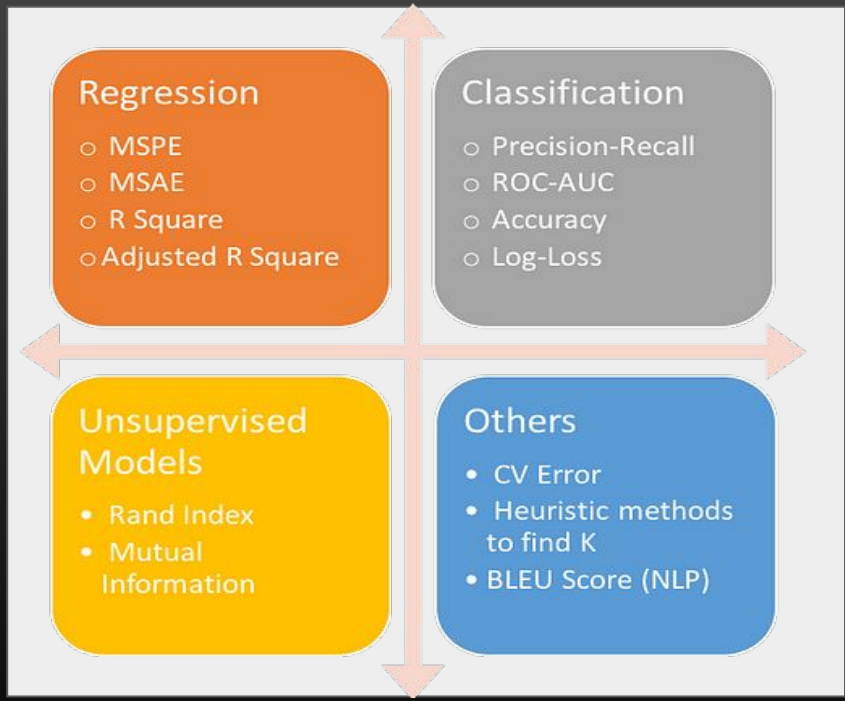
A/B testing for machine learning

- Performing A/B testing on the new model and the old model with production traffic can be an effective final step after offline evaluation
- By randomizing which users are in which group (A or B), you minimize the chances that other factors, like mobile vs. desktop, drive your metrics.



Metric-based tuning for machine learning model

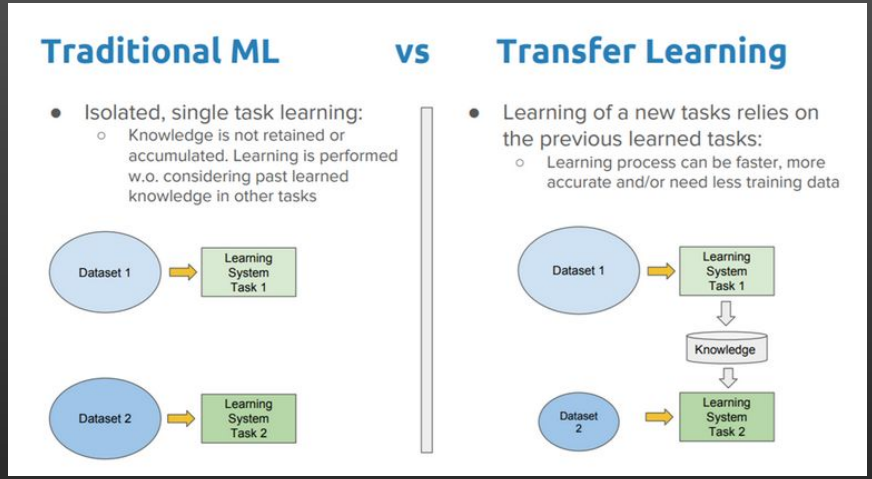
- We can continuously evaluate the model performance on new and old data
- The metric to select depends on the model and the data
- A drop in performance can automatically start a new learning phase or other actions

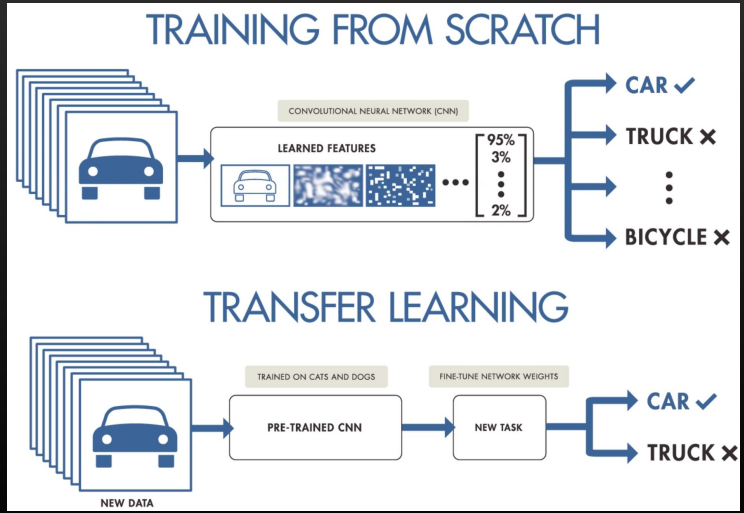


TRANSCEND

Transfer learning

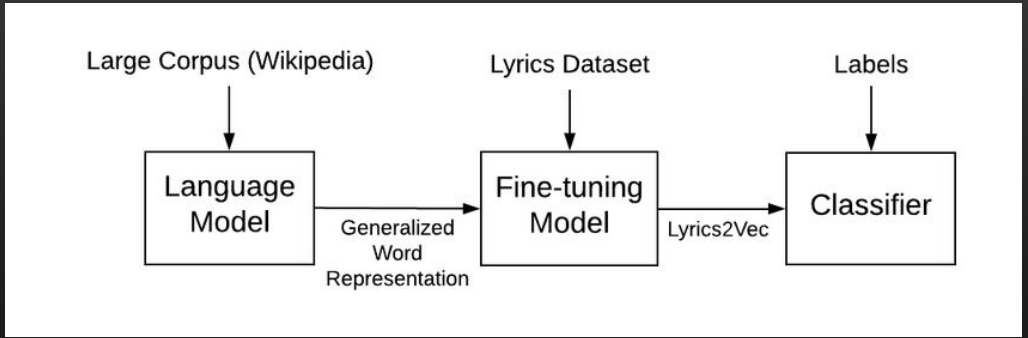
- Needs data
- Takes time
- Needs computing power





Fine tuning

- Data is difficult to acquire or annotate
- Limited time and computing resources



Transfer learning vs Fine tuning

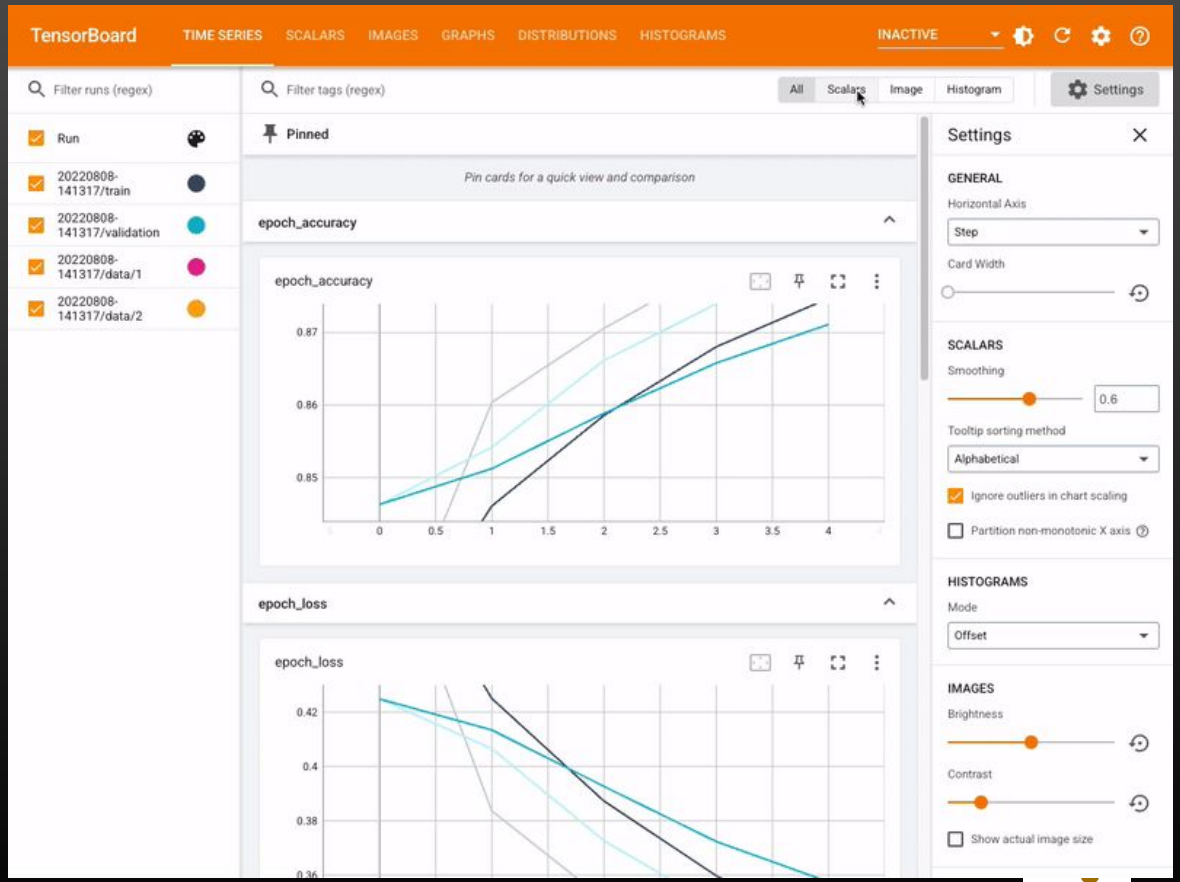
	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

Comparing ML model monitoring tools

- **ease of integration:** how easy is it to connect it to your model training and deployment tools
- **flexibility and expressiveness:** can you log and see what you want and how you want it
- **overhead:** how much overhead does the logging impose on your model training and deployment infrastructure
- **monitoring functionality:** can you monitor data/feature/concept/model drift? Can you compare multiple models that are running at the same time (A/B tests)?
- **alerting: does** it provide automated alerts when the performance or input goes crazy?

Tensorboard

- **Tracking and visualizing metrics** such as loss and accuracy
- Visualizing the **model graph** (ops and layers)
- Viewing **histograms** of weights, biases, or other tensors as they change over time
- Projecting **embeddings** to a lower **dimensional space**
- **Displaying** images, text, and audio data
- **Profiling** TensorFlow programs



Prometheus + Grafana

Dimensional data

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries

PromQL allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

Simple operation

Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

Precise alerting

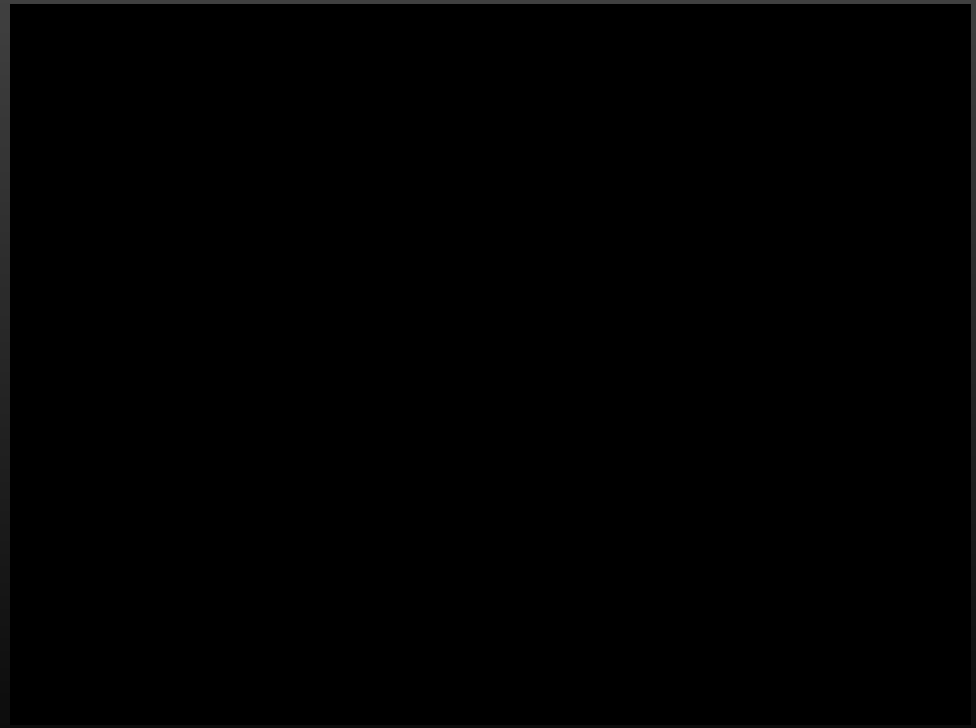
Alerts are defined based on Prometheus's flexible PromQL and maintain dimensional information. An alertmanager handles notifications and silencing.

Many client libraries

Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.

Many integrations

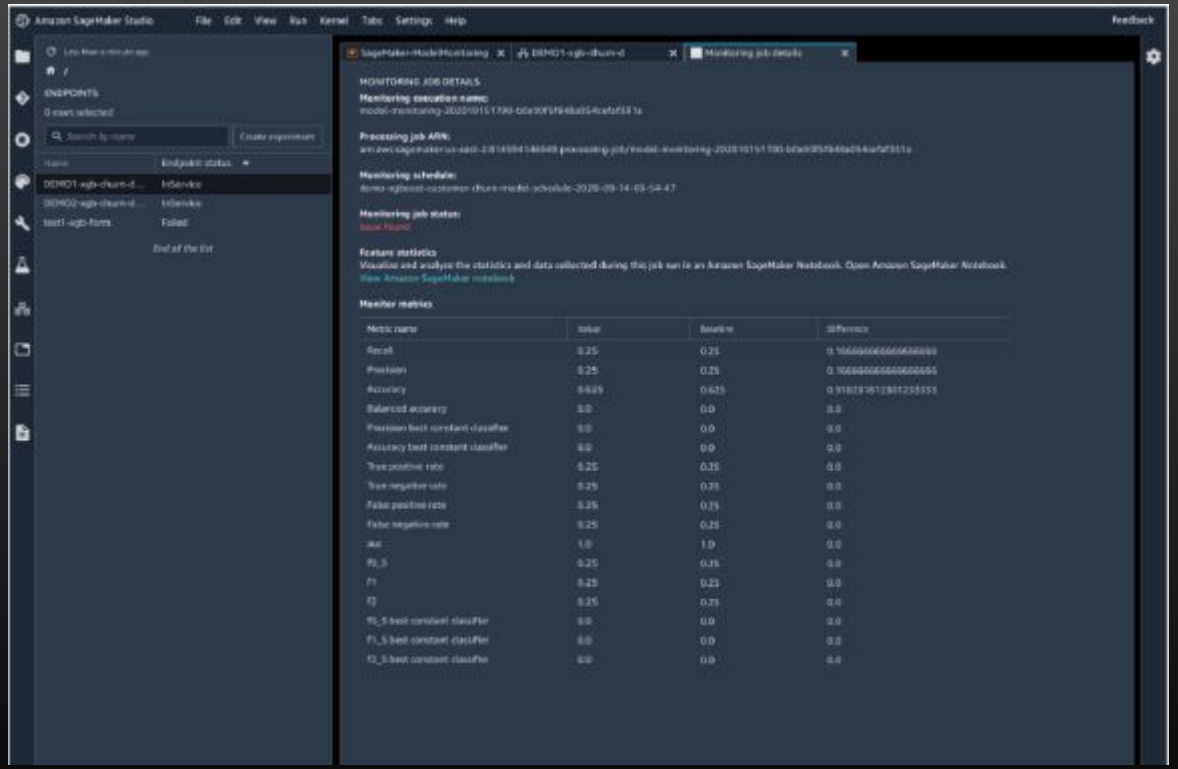
Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.



TRANSCEND

Amazon SageMaker Model Monitor

- **Customizable data collection and monitoring**
- **Detect drifts** in data and model quality
- Write custom rules and specify thresholds for each rule to **analyze model performance**
- **Visualization of metrics**
- **Model prediction** – import your data to compute model performance
- **Schedule** monitoring jobs
- The tool is integrated with Amazon SageMaker Clarify to identify potential bias in your ML models



ML model monitoring tools

	Neptune.ai	Arize AI	WhyLabs	Grafana + Prometheus	Evidently	Qualdo	Fiddler	Amazon SageMaker	Seldon Core	Censius
Model evaluation and testing	Yes	Limited	Limited	No	No	No	No	Yes	No	No
Hardware metrics	Yes	No	No	Yes	No	No	No	No	No	No
Model input/output distribution	No	Yes	Yes	Limited	Yes	Yes	Yes	Yes	Yes	Yes
Model training and re-training	Yes	Limited	Limited	No	No	No	No	Yes	No	No
Model performance in production	No	Yes	Yes	Limited	Yes	Yes	Yes	Yes	Yes	Yes
CI/CD pipelines for ML	Yes	No	No	No	No	No	No	Yes	No	No

TRANSCEND

Privacy challenges

Privacy Challenges in AI Development

How is privacy impacted by AI models

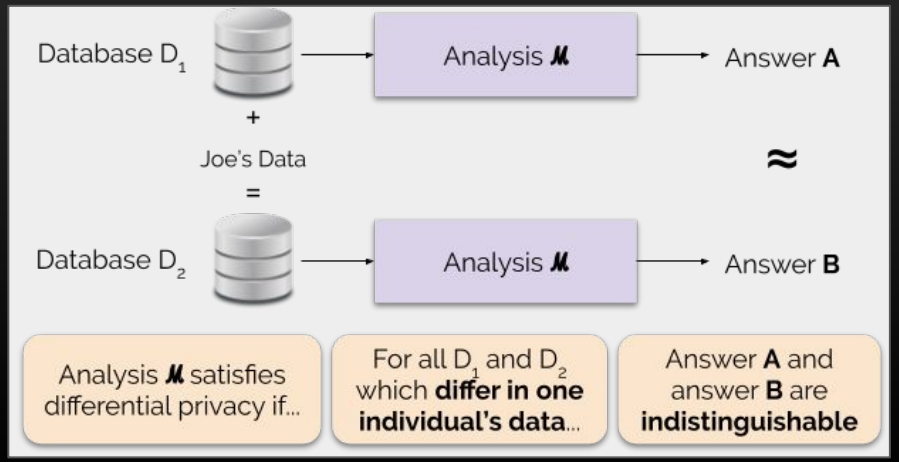
- **Privacy Risks**
  - AI systems often process large datasets, including sensitive information like health and financial data.
  - Higher data volume increases potential for misuse, unauthorized access, or data breaches.
- **Ethical Balance**
  - Essential to balance AI's capabilities and benefits to society with protection of individual rights.
  - Privacy protection needs to be integral, not an afterthought, in AI development and deployment.
- **Black-Box Models (explicability/interpretability)**
  - Many AI models, especially deep learning models, operate with opaque decision-making processes.
  - Limited transparency complicates efforts to track how data is processed and if privacy is respected.

TRANSCEND

Algorithmic Techniques for Privacy Preservation

How to technically preserve privacy

- **Differential Privacy**
  - Technique that adds random noise to datasets, protecting individual data points' anonymity.
  - Allows insights and analysis at the aggregate level without revealing specific data points.
  - Used in applications where aggregated data insights are valuable but privacy is crucial (e.g., census data).



TRANSCEND

Algorithmic Techniques for Privacy Preservation

Example Scenario: A Health Survey on Exercise Habits

Imagine a health organization wants to analyze the exercise habits of a population to understand patterns and trends. However, they also want to ensure that individual responses remain private, especially as the survey involves sensitive personal data.

Step 1: Adding Noise to the Data

- If a respondent reported they exercise 3 times per week, the system might add or subtract a small random value (say, ±1 or ±2) to this response.
- The altered answer (e.g., 5 or 1 times per week) is then stored, rather than the exact original answer.

Step 2: Aggregating the Noisy Data

When the organization aggregates all the data to determine the average number of exercise sessions per week, the noise averages out across the whole dataset. This allows them to observe population-level trends without revealing any specific individual's exact answer.

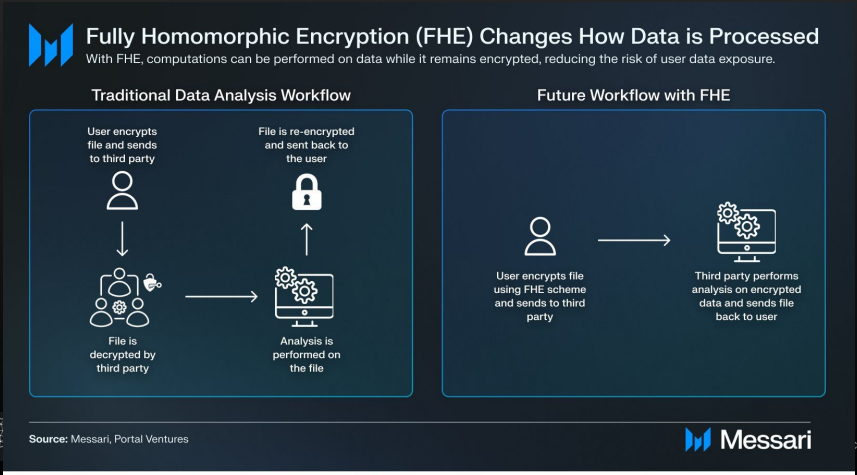
Result: Preserving Privacy with Useful Insights

- Privacy: Individual responses are obscured because of the added noise, so no one can precisely identify if a specific individual reported a certain exercise frequency.
- Utility: The organization can still get an accurate estimate of the average exercise frequency across the population, as the noise has little impact on the overall trend.

Algorithmic Techniques for Privacy Preservation

How to technically preserve privacy

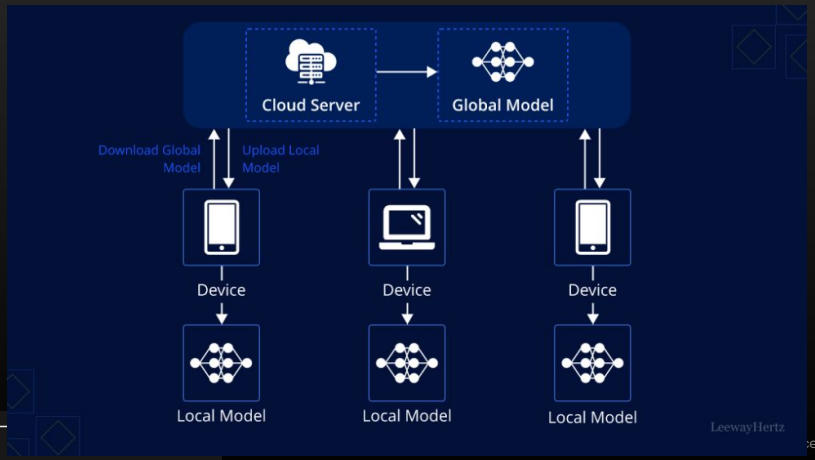
- **Homomorphic Encryption**
  - Enables computation on data while it remains encrypted, maintaining confidentiality throughout processing.
  - Supports secure data processing, particularly useful in untrusted or third-party environments (e.g., cloud).
  - Provides a paradigm for privacy-focused AI solutions without compromising data utility.

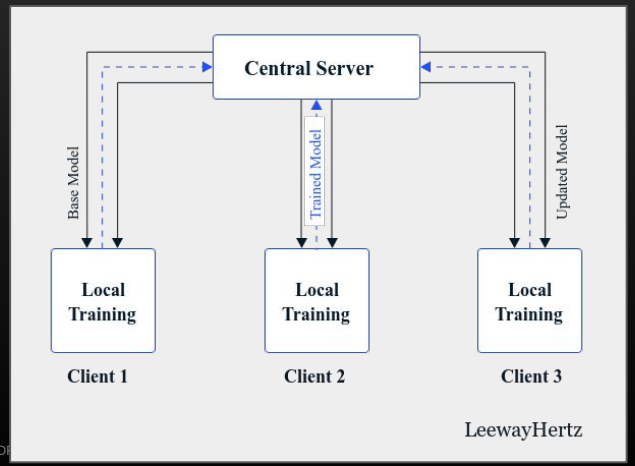


Algorithmic Techniques for Privacy Preservation

How to technically preserve privacy

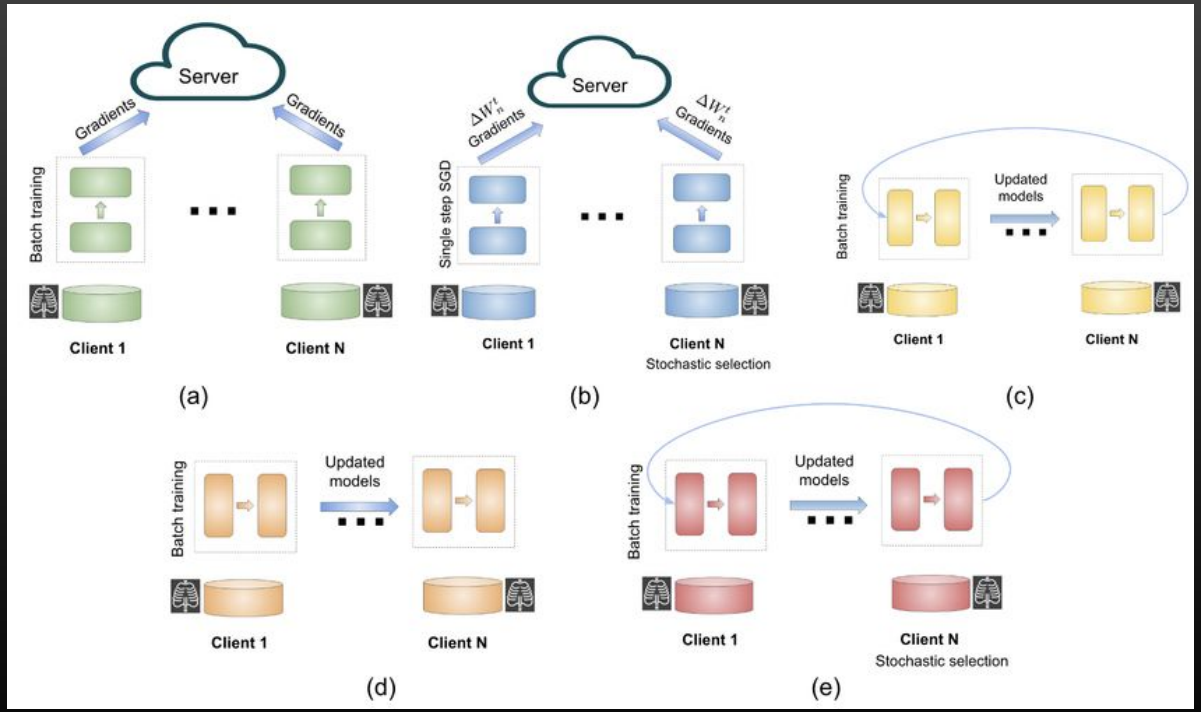
- **Federated Learning**
  - Decentralizes AI model training by keeping data localized on devices rather than sharing it centrally.
  - Model updates, not raw data, are shared; preserves privacy across decentralized networks.
  - Ideal for applications on mobile devices (e.g., predictive text, health monitoring) where data sensitivity is high.





Algorithmic Techniques for Privacy Preservation

Darz E.i et al. A Comparative Study of Federated Learning Models for COVID-19 Detection, 2023.

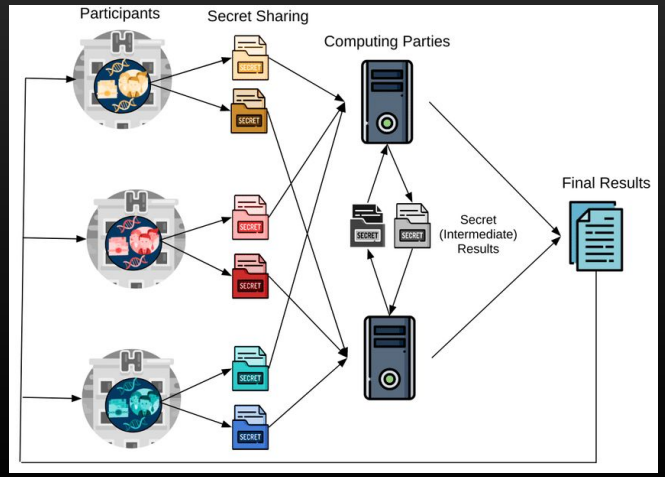


Schematic view of FL models and algorithms, (a) Federated averaging, clients train on local batch of data (b) FedSGD, a subsample of clients are selected, and each performs single step SGD and sends the model updates to the server (c) Cyclic Weight transfer (CWT), clients train locally, and pass the model to the next clients, and the cycle repeats (d) Single weight transfer (SWT) model passes each client only once. (e) Stochastic weight transfer (STWT), the model is passed sequentially through clients, and participating clients in each round are sampled randomly.

Algorithmic Techniques for Privacy Preservation

How to technically preserve privacy

- **Secure Multi-Party Computation (SMPC)**
  - Allows multiple parties to jointly compute functions over their data without exposing individual inputs.
  - Ensures collaborative data analysis without requiring raw data sharing.
  - Used in scenarios where entities collaborate but need to maintain data confidentiality (e.g., collaborative research).



K. Dodiya et al. Differential Privacy Techniques in Machine Learning for Enhanced Privacy Preservation JETIR February 2024.

TRANSCEND

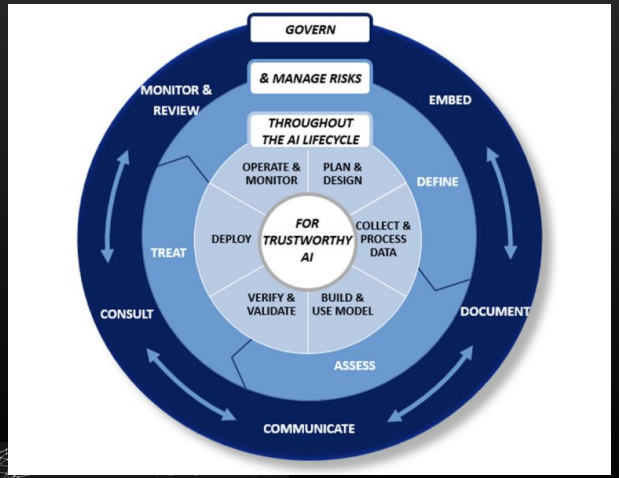
Addressing Bias, Transparency, and Accountability in AI

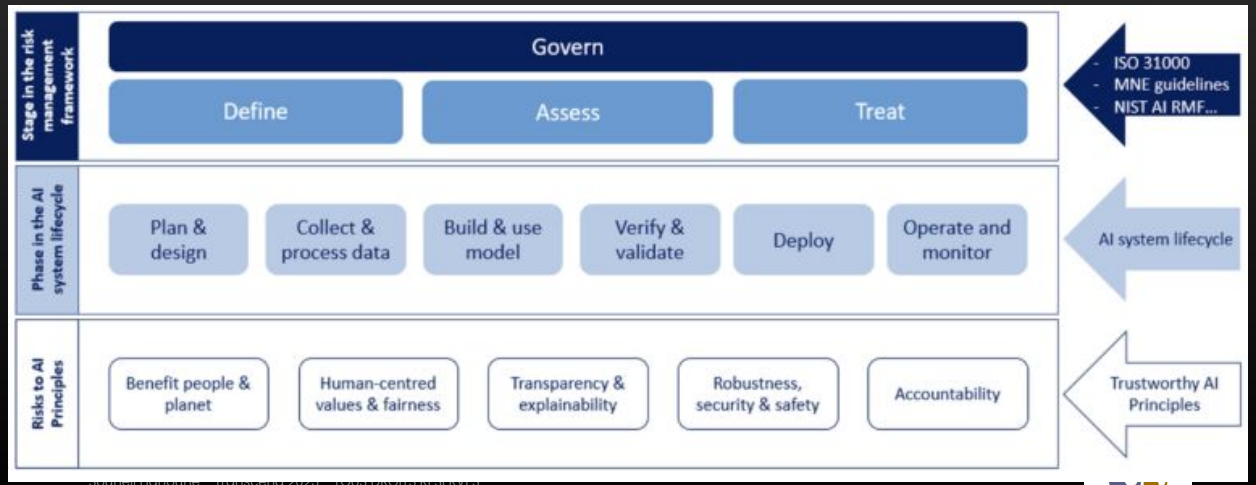
- **Transparency in AI Models:**
  - Transparency is critical, especially in high-stakes domains like healthcare, finance, and criminal justice.
  - Tools like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) offer insights into model decision-making.
  - Feature importance analysis helps identify the influence of different data features on AI predictions.
- **Bias in Datasets:**
  - Biased data can lead AI models to make unfair, prejudiced, or discriminatory decisions.
  - Important to use representative datasets that accurately reflect the diversity of the population.
  - Bias mitigation techniques, such as re-weighting, oversampling, or algorithmic adjustments, help reduce bias impact.

Addressing Bias, Transparency, and Accountability in AI

- **AI Lifecycle Accountability:**
  - Developer Responsibilities: Incorporate ethics and transparency into the model design and testing phases.
  - Company Role: Companies deploying AI must ensure ethical data practices and implement safeguards against misuse.
  - End-User Impact: Users and affected communities should demand transparency and accountability from AI developers and implementers, fostering responsible use of AI.

https://oecd.ai/en/accountability





Integrating Ethical and Regulatory Strategies in AI Development

Conclusion

- **Comprehensive Approach:**
  - Effective privacy protection requires a blend of technological solutions, regulatory compliance, and ethical guidelines.
  - Emphasis on collaboration among technologists, ethicists, policymakers, and civil society to address privacy in AI.
- **Future of Ethical AI Research:**
  - Focus areas include federated learning, differential privacy, and ongoing interdisciplinary studies on ethics.
  - Emphasis on adaptive research that addresses new ethical and privacy challenges as AI evolves.
- **International Standards:**
  - GDPR, OECD Guidelines, and other international standards set foundational principles for privacy and ethical AI.
  - Harmonized frameworks are essential to create consistent and universally respected AI practices.
- **Ethics by Design:**
  - Embedding ethical considerations directly into AI design stages ensures respect for privacy and human rights.
  - Proactive integration of ethics minimizes the need for post-deployment fixes and builds trust in AI systems.

Professional positions in AI deployment

TRANSCEND

Data scientist / ML Engineer / Data Engineer

Data Scientist vs. Machine Learning Engineer vs. Data Engineer

	Data scientist	Machine learning engineer	Data engineer
What they do	Build models that help business get better insights and make predictions from their data	Automate ML processes and make models work in a production environment.	Build, test and maintain data pipelines; provide ML models with quality data.
Skill set	✓ Knowledge of math and statistics ✓ Decision making and data optimization skills ✓ High proficiency in SQL ✓ Scripting skills (R/Python)	✓ Solid programming background ✓ Data science skills ✓ Knowledge of math and statistics ✓ Rapid prototyping skills ✓ Good problem-solving-skills ✓ Proficiency in deep learning frameworks	✓ Scripting skills (Linux commands) ✓ Knowledge of databases ✓ Knowledge of cloud technologies ✓ Proficiency in SQL ✓ Data modelling skills ✓ ELT development skills
Tools used	Python, R, Pandas, Jupyter notebooks, SQL	Python, PyTorch, TensorFlow, cloud services	SQL, Oracle, Hadoop, Amazon S3, Python

altexsoft
software r&d engineering

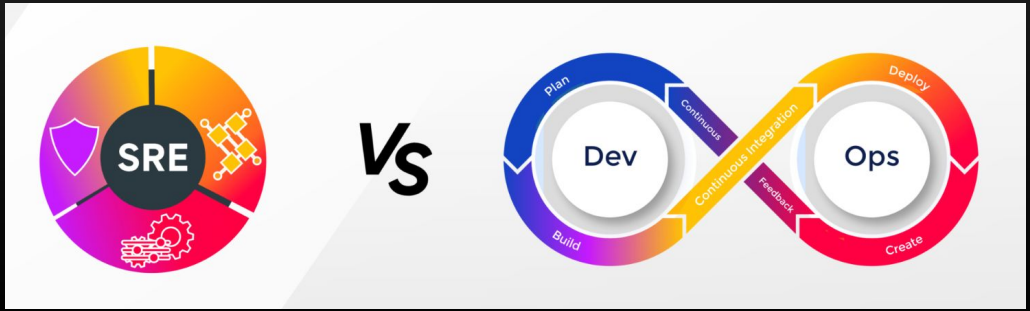
DevOps & SRE (Site reliability Engineer)

- **DevOps** is [the] set of **cultural norms** and **technology practices** that [enables] the **fast flow of planned work** from, among others, development, through tests into operations while preserving world-class reliability, operation and security. **DevOps isn't about what you do, but what your outcomes are.**

- Google's definition of SRE is quite simple:

  "SRE is what happens when you ask a software engineer to design an operations team."

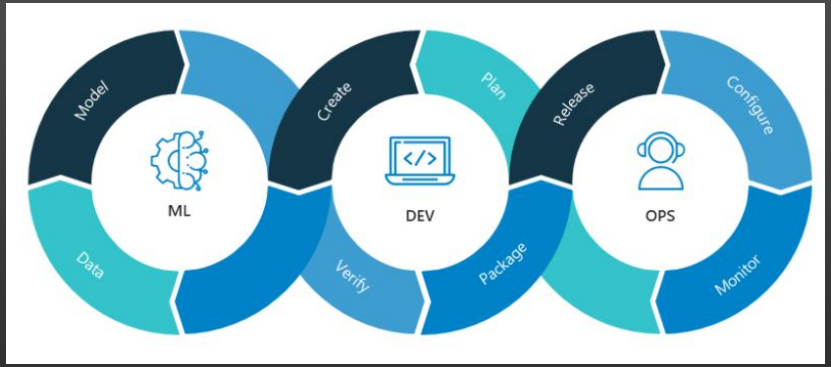
- **SREs** are operations folks with strong **development backgrounds**, and they apply **engineering practices** to solve **common problems** when **running systems in production**. SREs are **responsible for making systems available**, **resilient**, **efficient**, and **compliant** with the organization's policies (like change management).

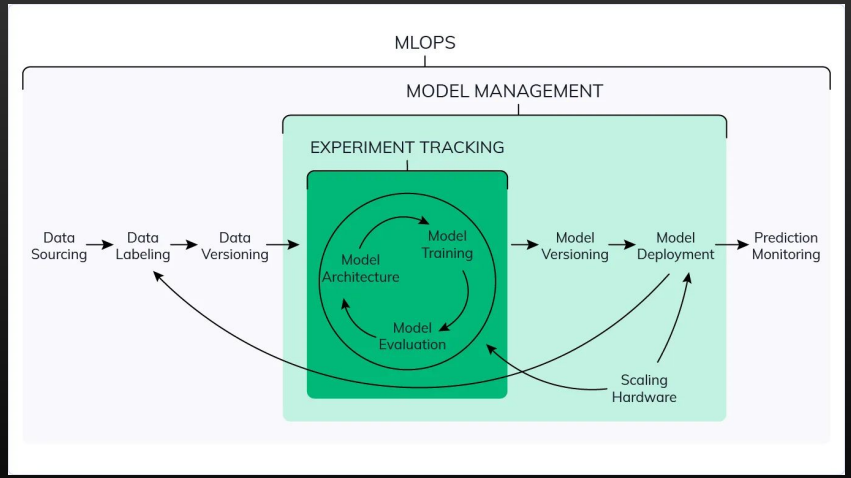


TRANSCEND

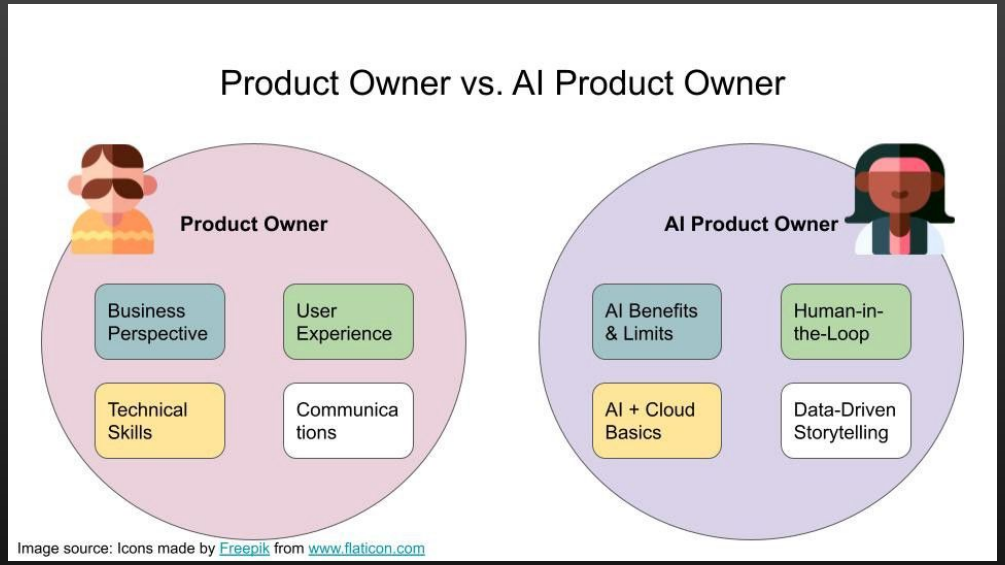
MLOps

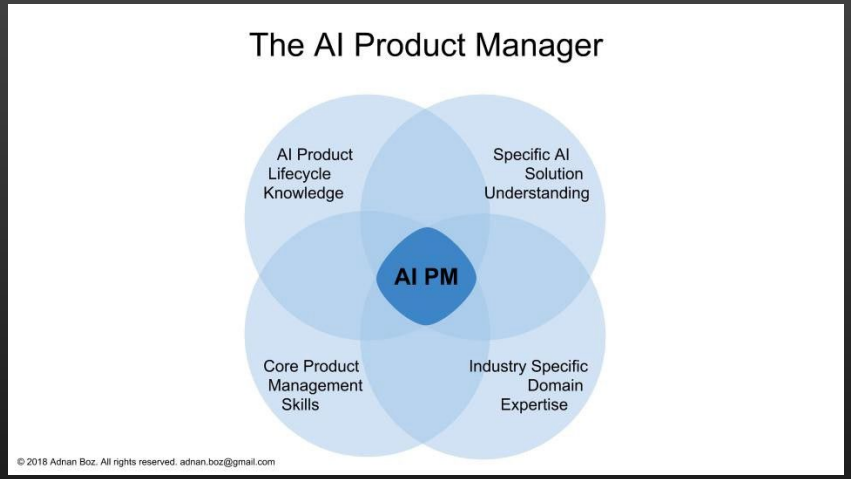
- **MLOps (Machine Learning Operations)** is a set of practices for collaboration and communication between data scientists and operations professionals. Applying these practices increases the quality, simplifies the management process, and automates the deployment of Machine Learning and Deep Learning models in large-scale production environments. It's easier to align models with business needs, as well as regulatory requirements.





Product Owner / Product Manager





Case studies and discussion

Example deploying svm on Azure

https://medium.com/aiguys/model-deployment-on-azure-as-a-web-app-b7bb5599bfbf