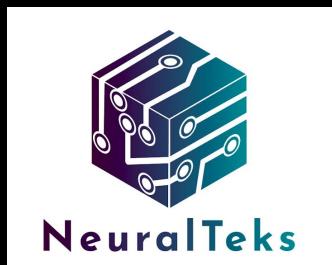


Deployment and maintenance of artificial intelligence models



Dr. Souheil Hanoune
2024/2025

souheil.hanoune@neuraltex.com

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, Alvancity, École des Mines de Paris)
- Editorialist in ActuA 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
- ActuA 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

€20 millions invested in R&D & product

HQ in Paris La Défense

80 employees (tech teams 50+ people)

8 years of R&D, 150 PoC

7 PhDs in AI

€22M in Series A

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



- 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 :

- Paramétrier vos cas d'usage en 5 clics
 - Sélection de l'usage
 - Définition de la classe
 - Définition de la zone d'analyse
 - Réglage de la temporalité
 - 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



©2021. Tous droits réservés à XXII GROUP.

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



©2022. Tous droits réservés à XXII GROUP.

XXII Alertes - Temps réel

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

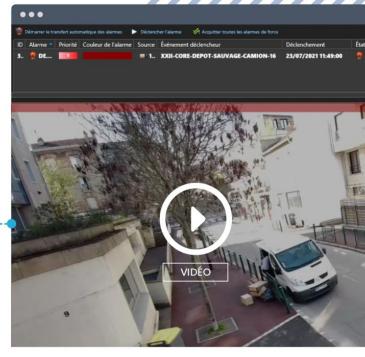
Fonctionnalités :

- 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 :



©2021. Tous droits réservés à XXII GROUP.

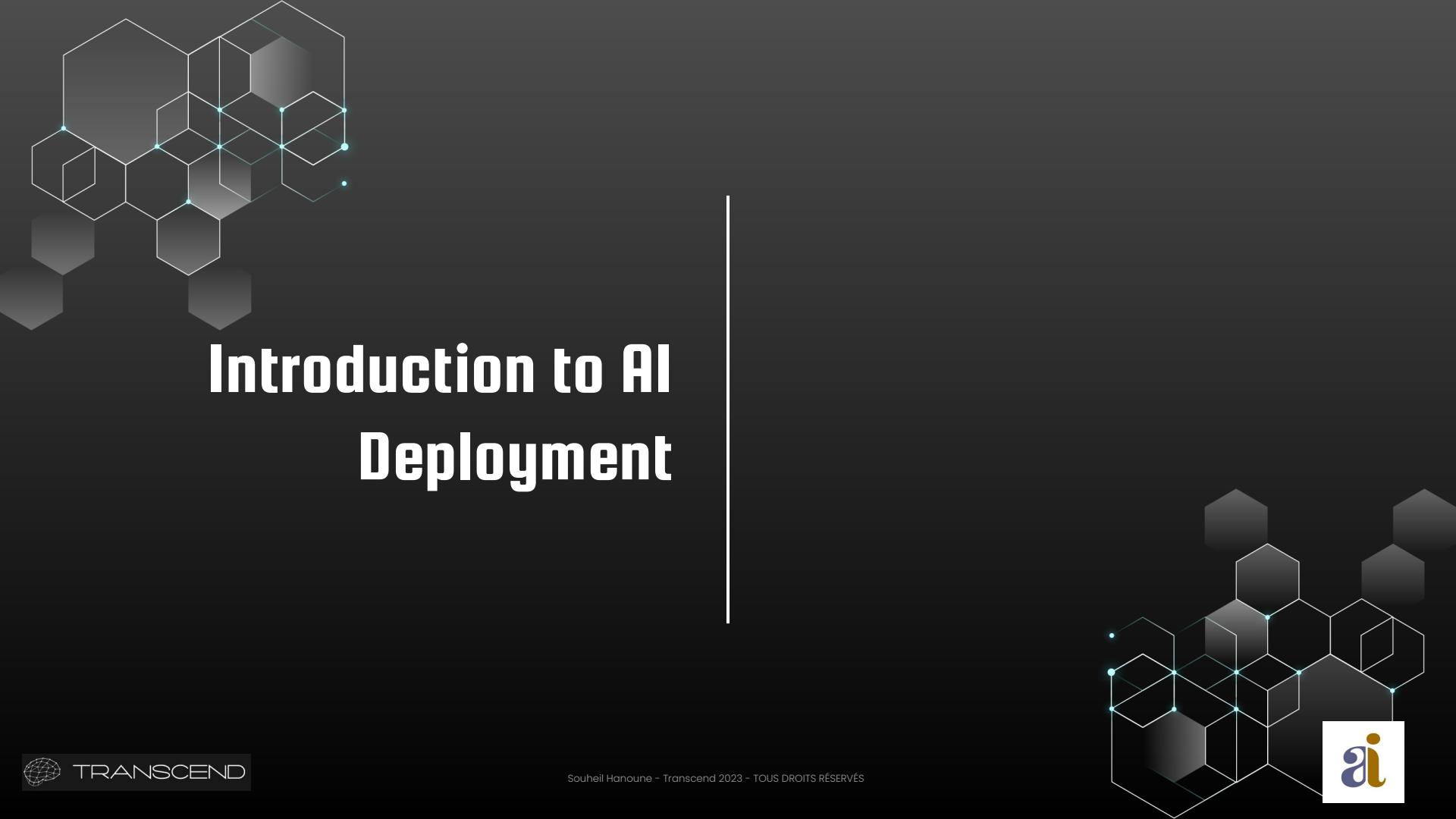


NeuralTeks & MicroTeks



SUMMARY

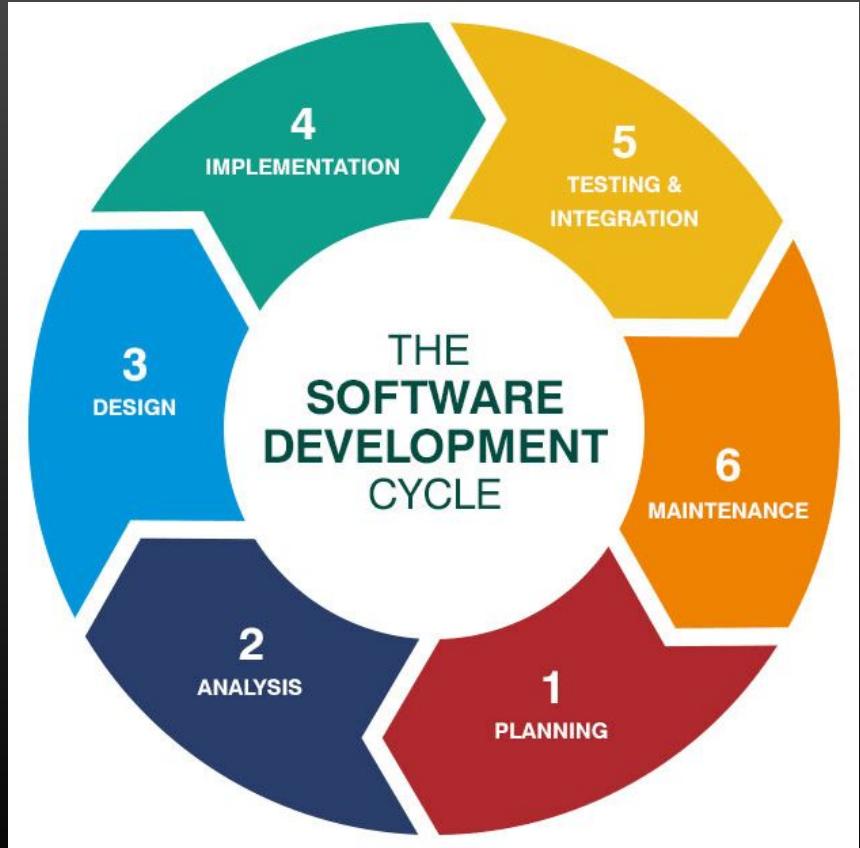
- Introduction to AI deployment
- Model selection and deployment options
- Deployment methods and technologies
- Deployment tools and frameworks
- Model management and version control
- Model monitoring and performance tuning
- Ethical considerations for AI models deployment



Introduction to AI Deployment

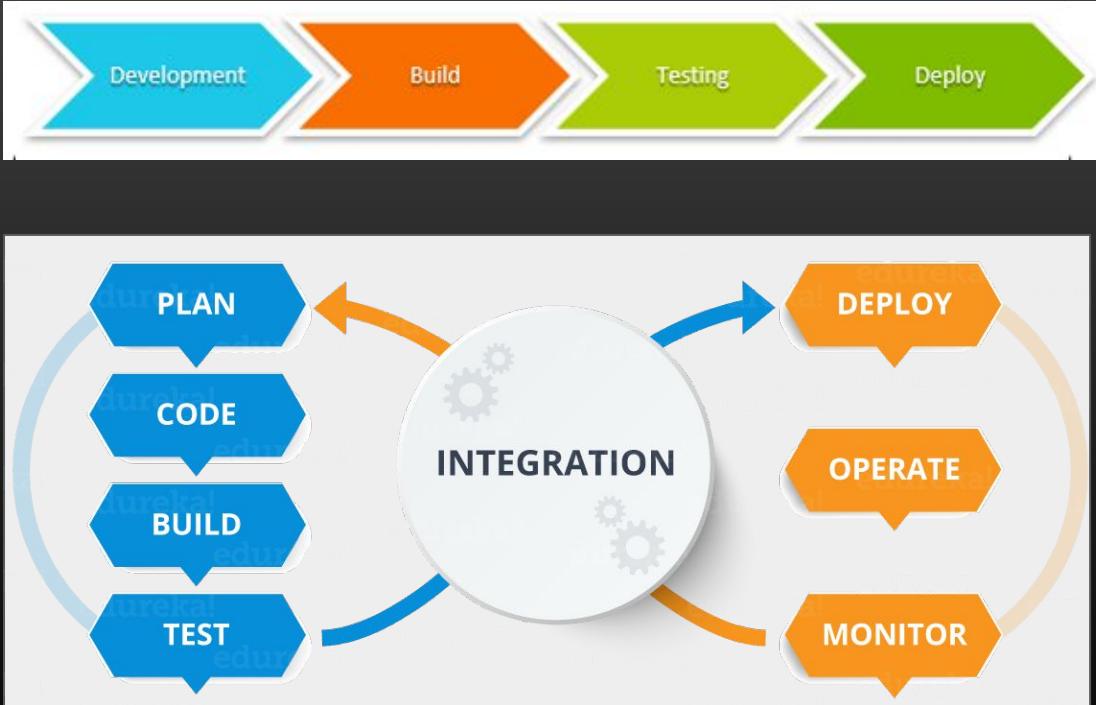
Software development cycle

- 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



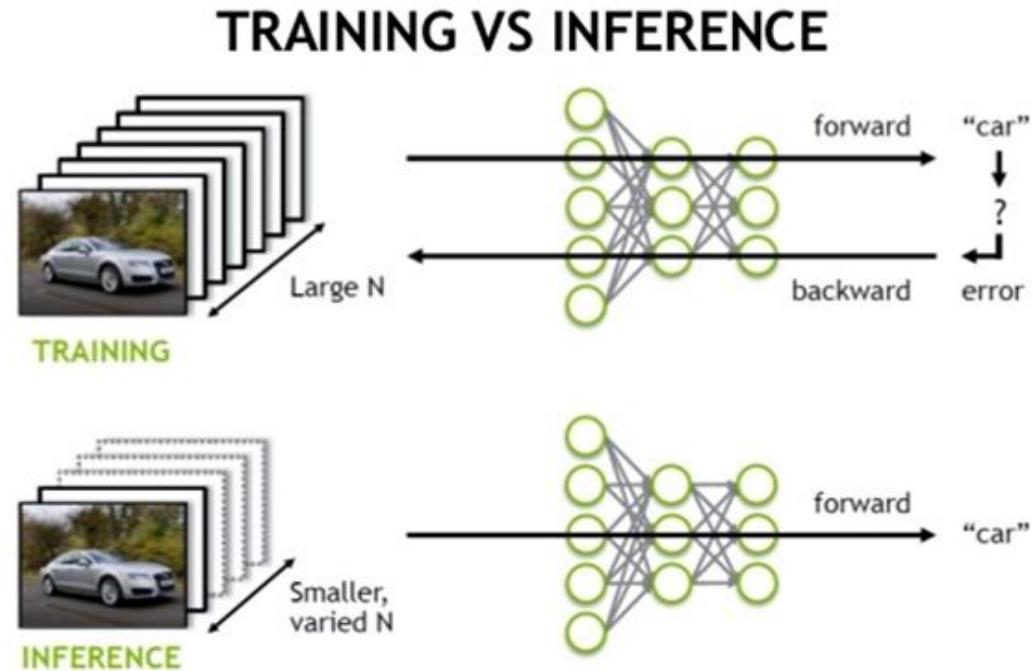
Definition of AI Deployment

- **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

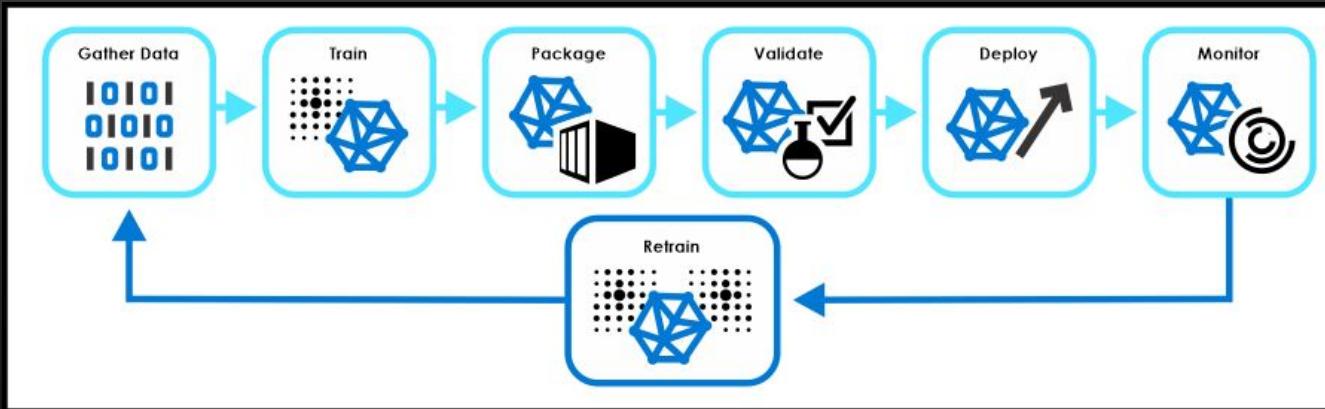
AI Deployment importance

- 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

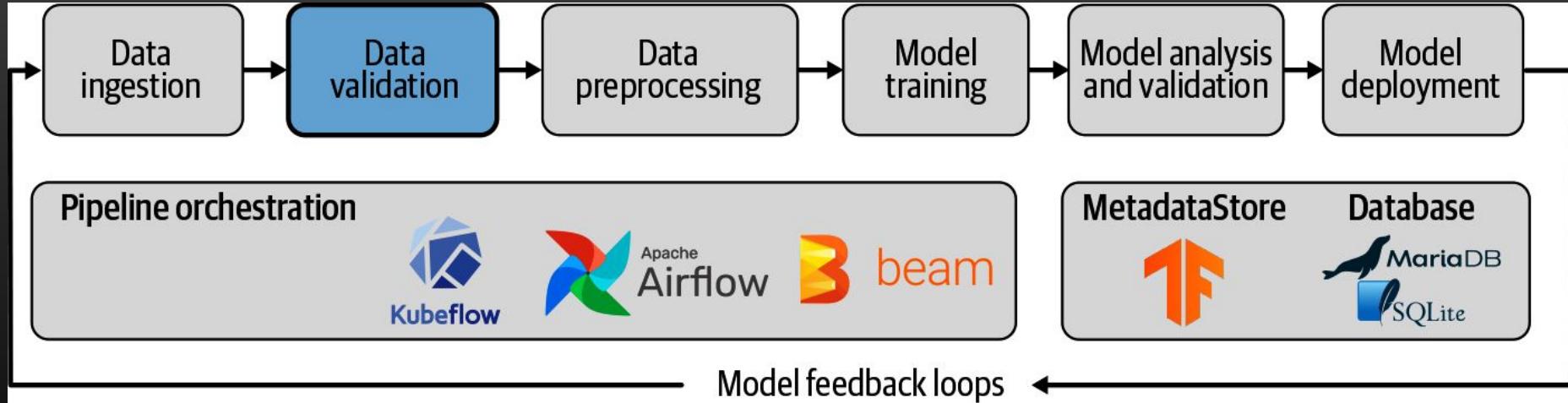
Training vs inference



Deployment VS training

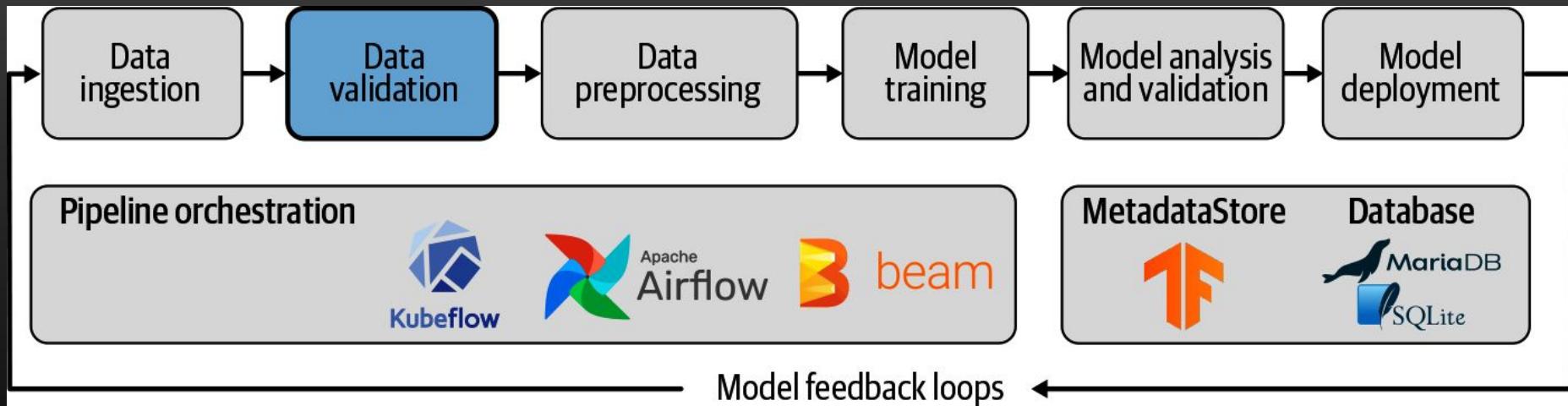


Example of AI pipeline



Example of AI pipeline

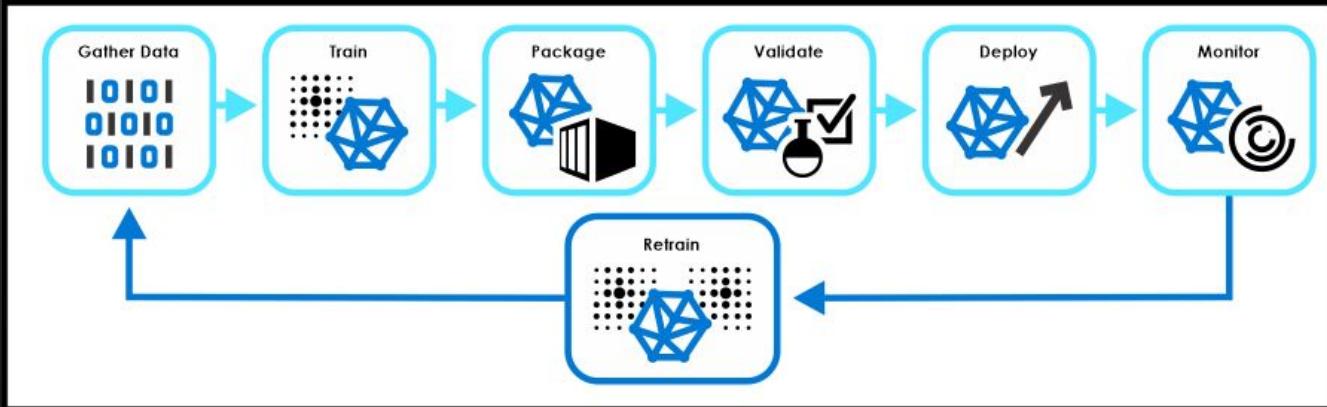
Take 10mn and imagine a use case to deploy and the various tools you know for each step of the pipeline



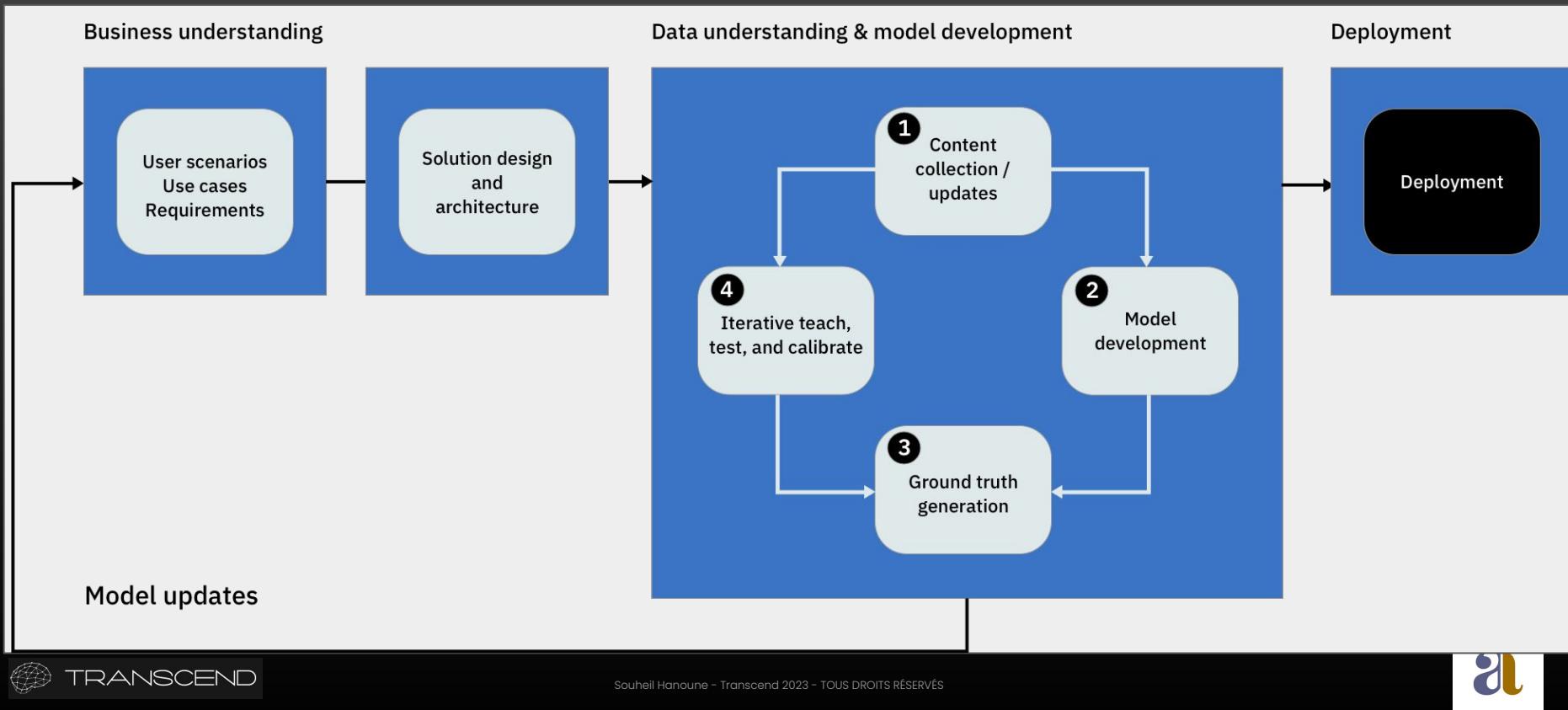


Model selection and deployment possibilities

Deployment VS training

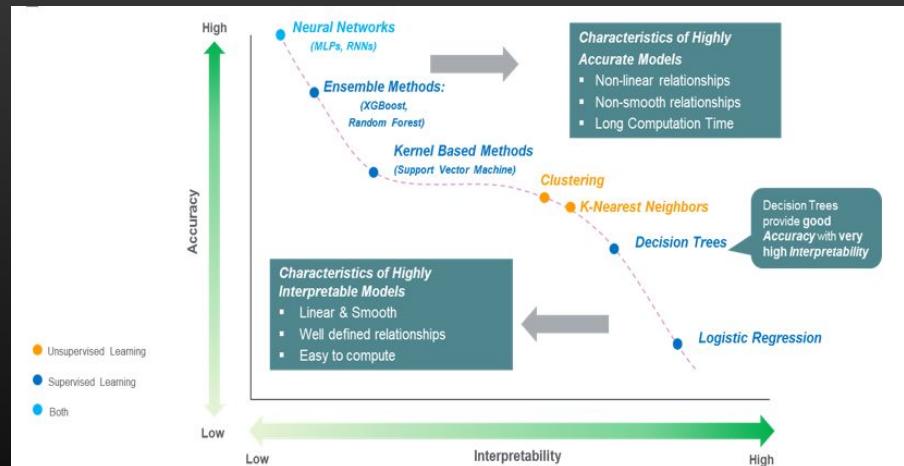


Continuous model deployment



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



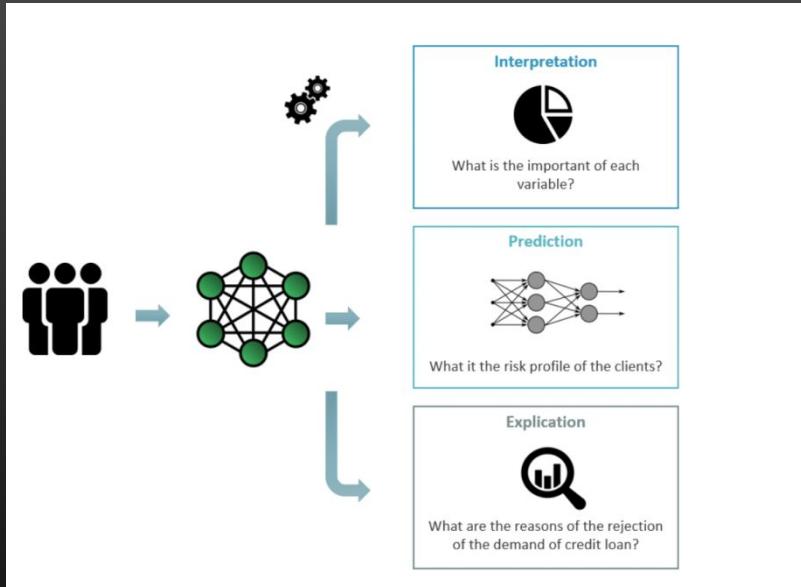
Explicability vs. Interpretability

- **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

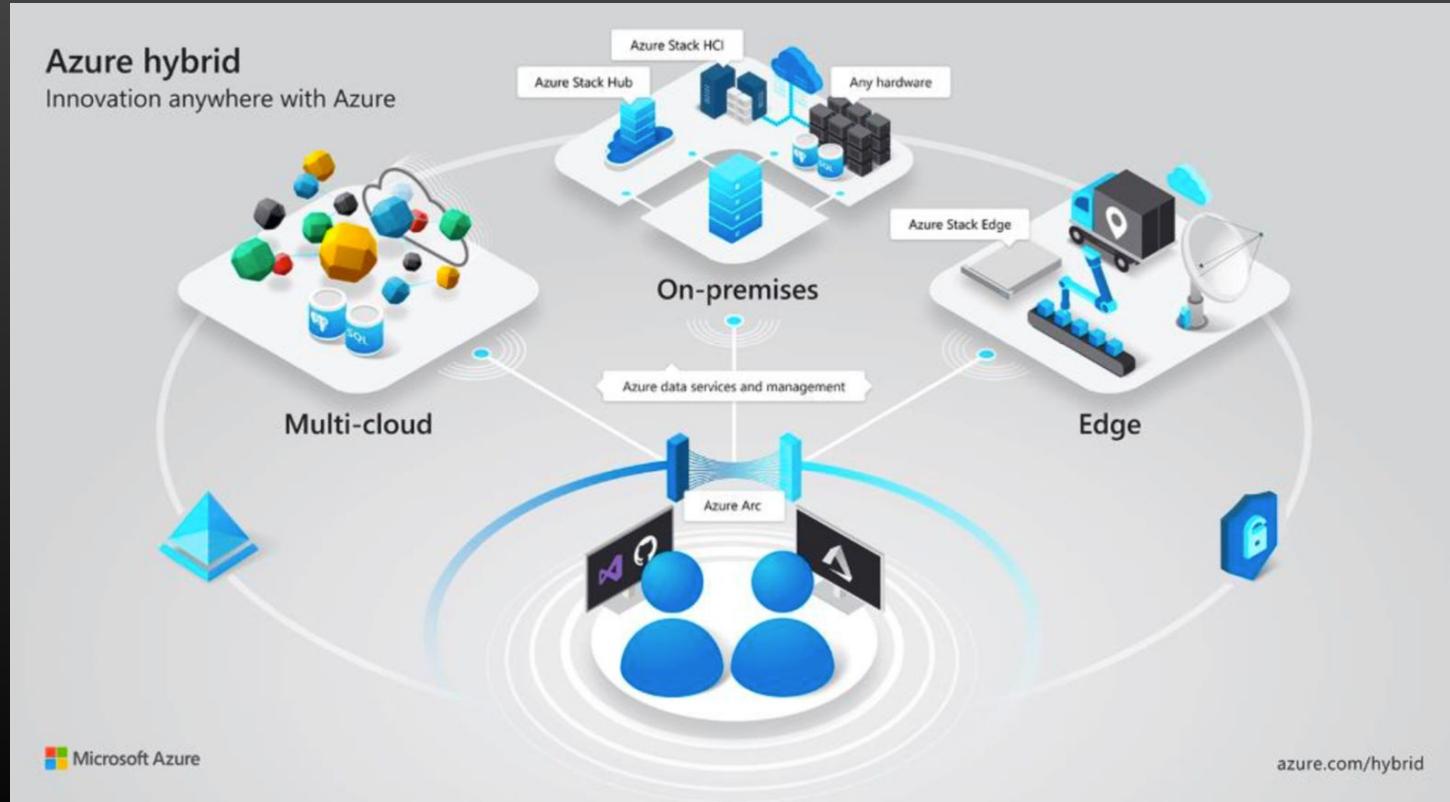
- **Interpretability**

- 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

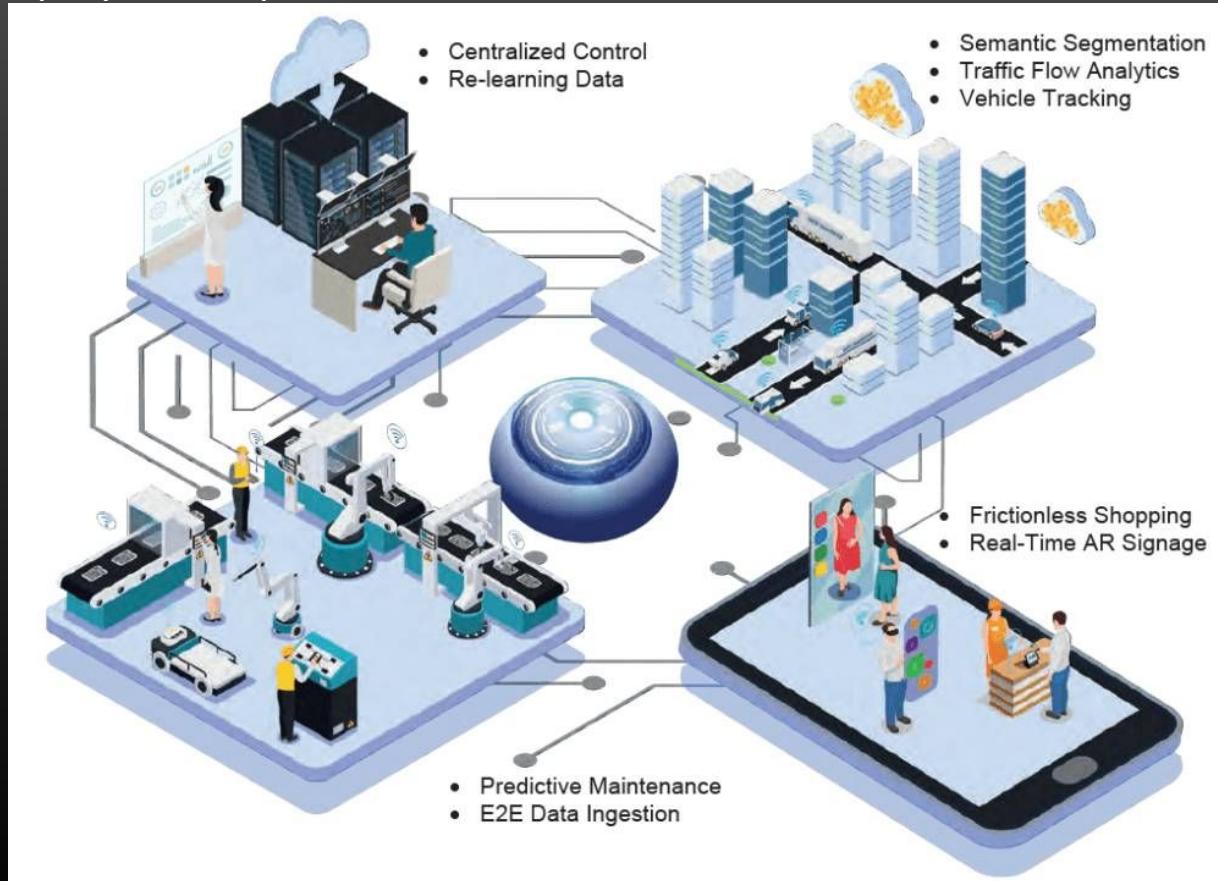


An explainable decision is
interpretable

Deployment options



Examples of deployment options



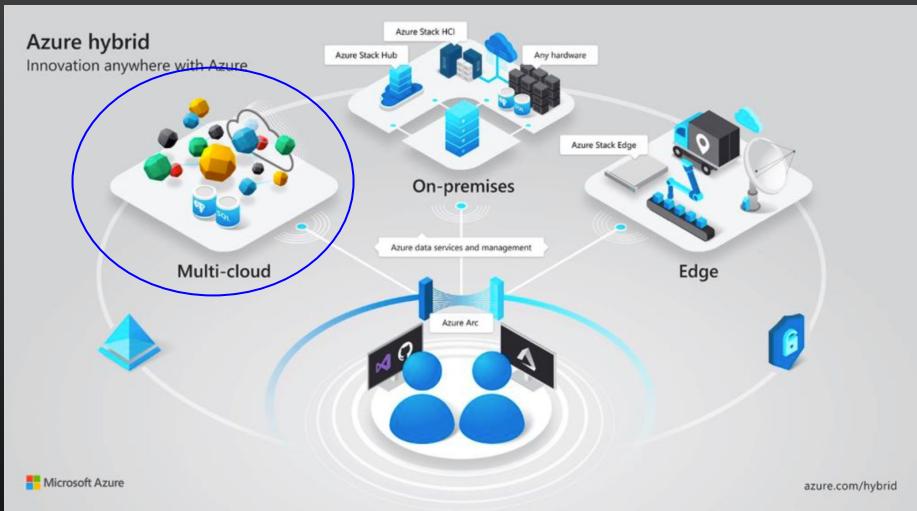
Cloud deployment

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)



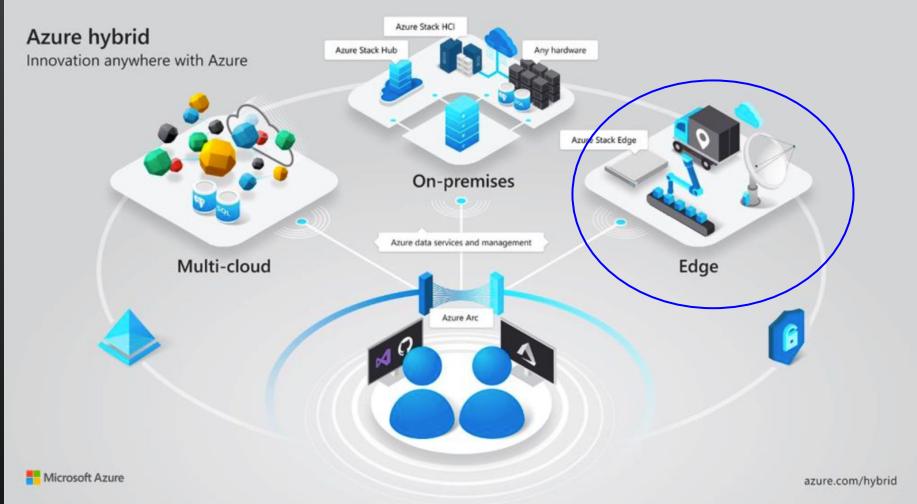
Edge deployment

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



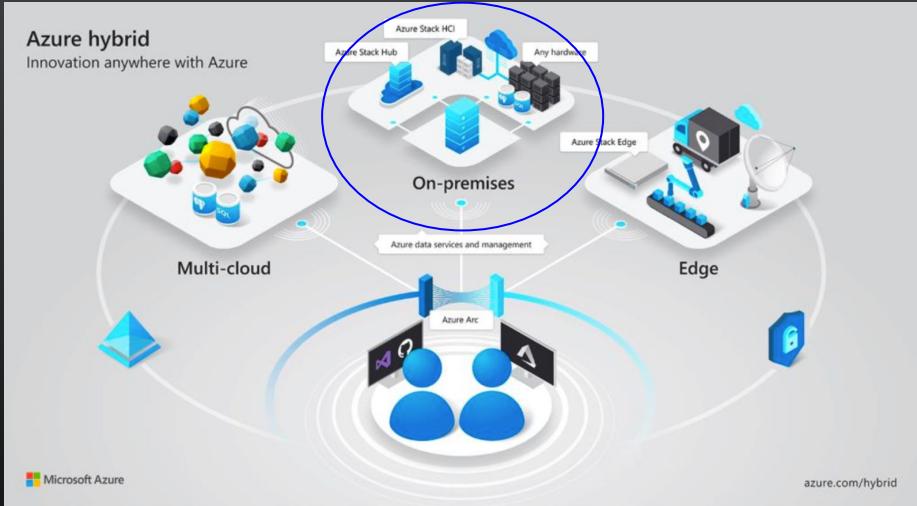
On premise deployment

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

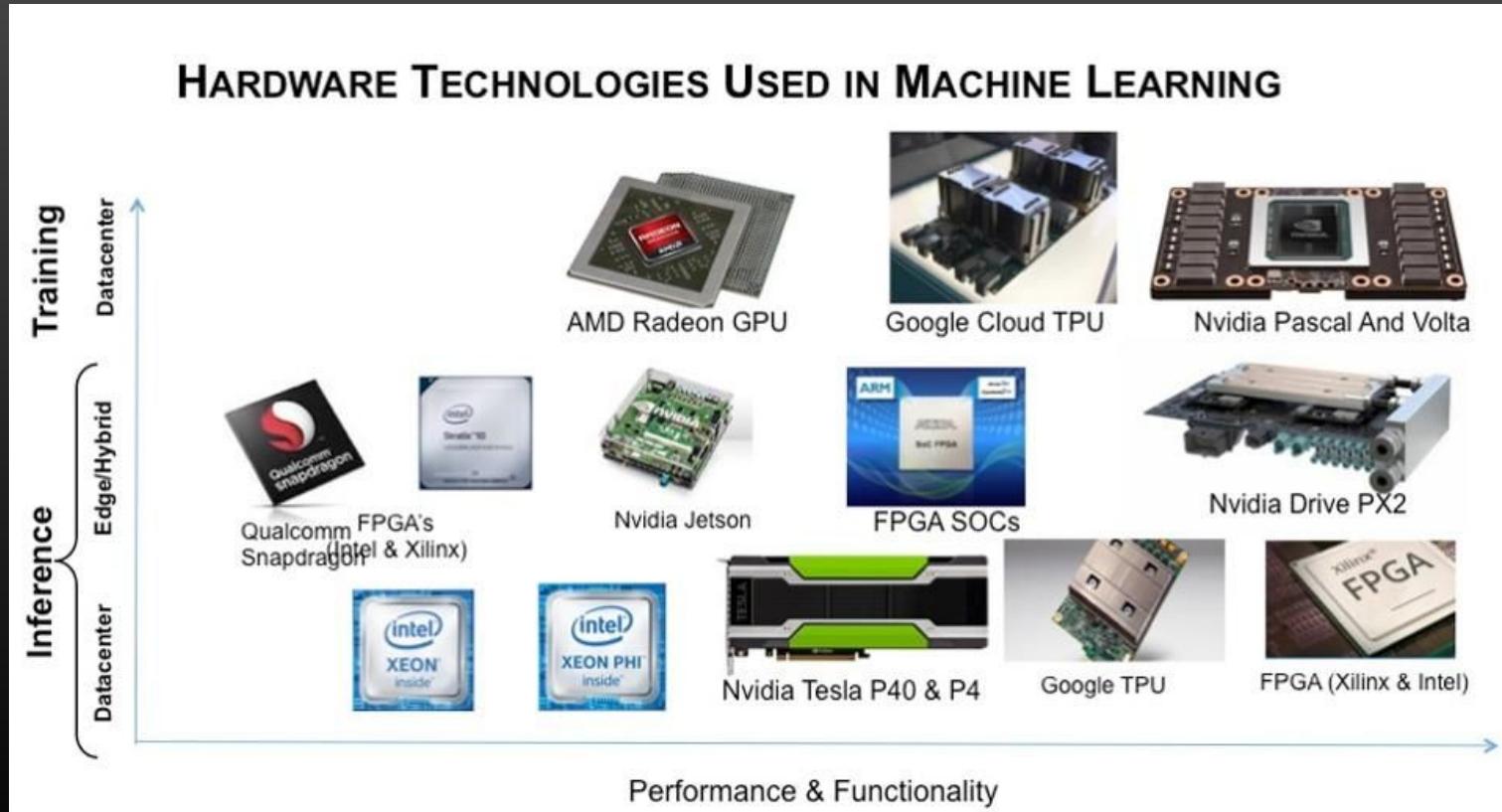
- **Factors to Consider When Determining Your Method of Deployment**
 - How frequently predictions will be generated and how urgent the results are needed?
 - If predictions should be generated individually or by batches
 - The latency requirements of the model, the computing power capabilities that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



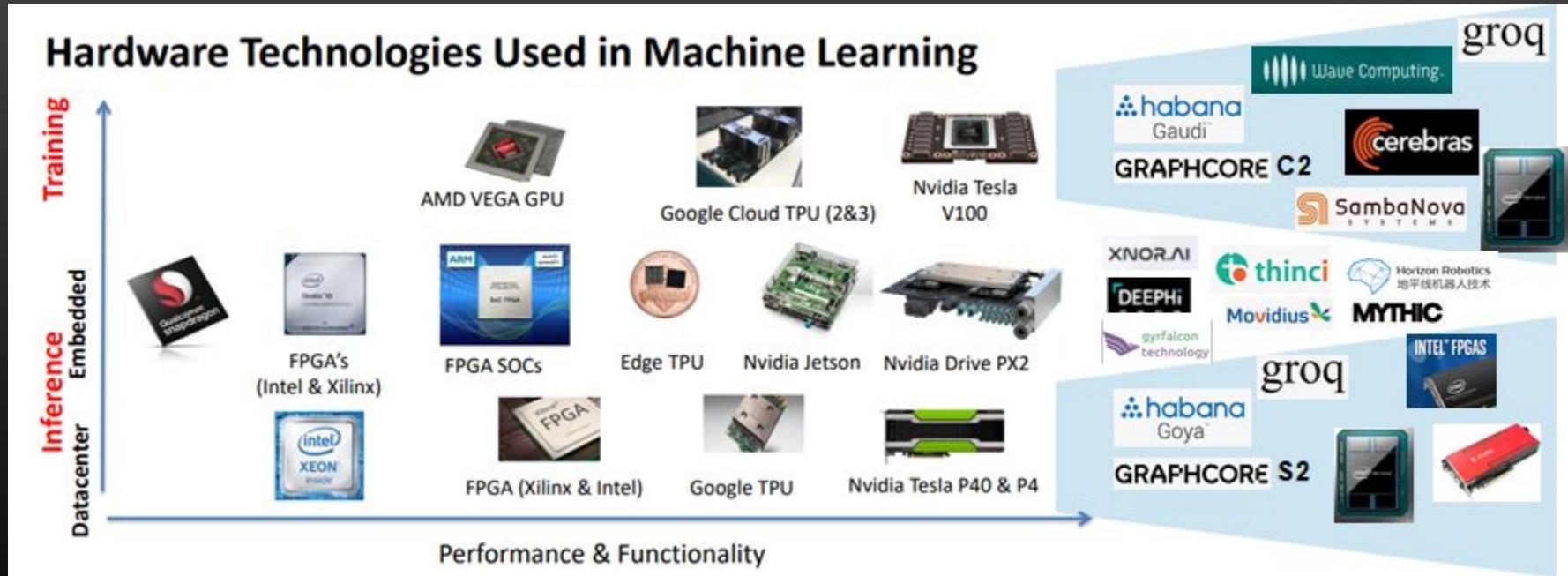
Types of Deployment in ML:

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

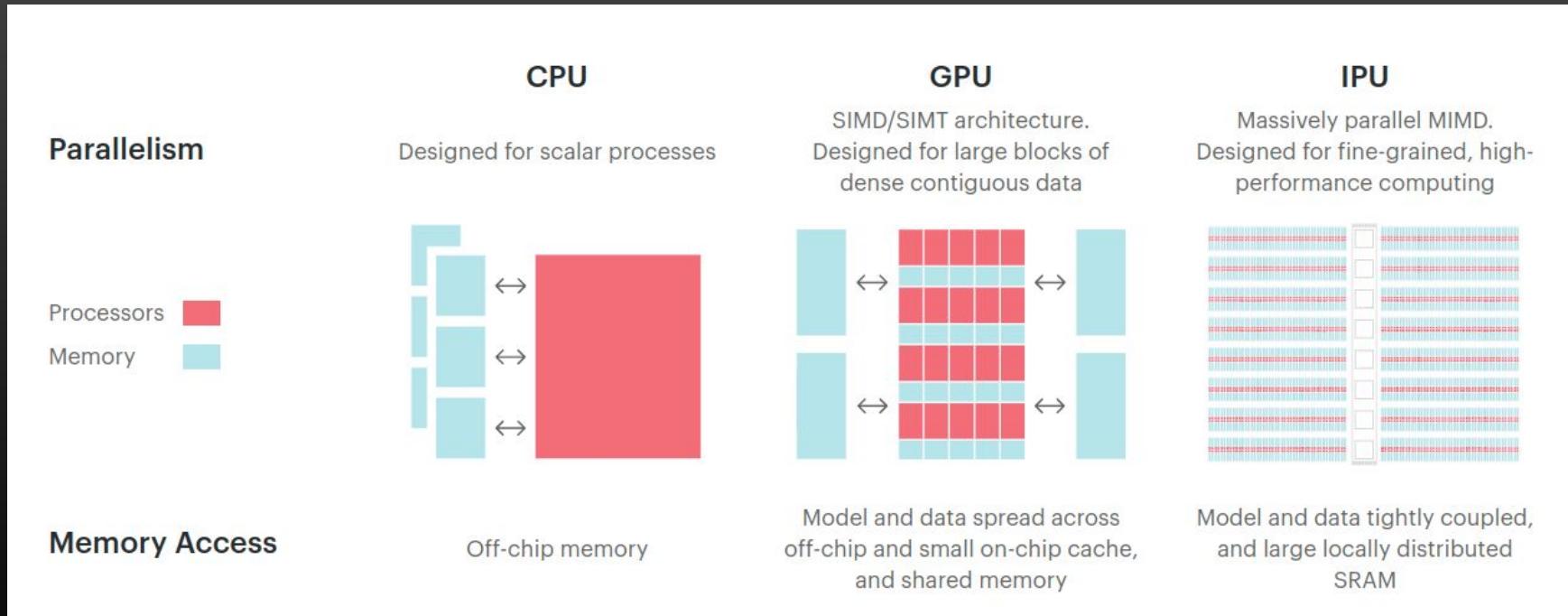
The compute



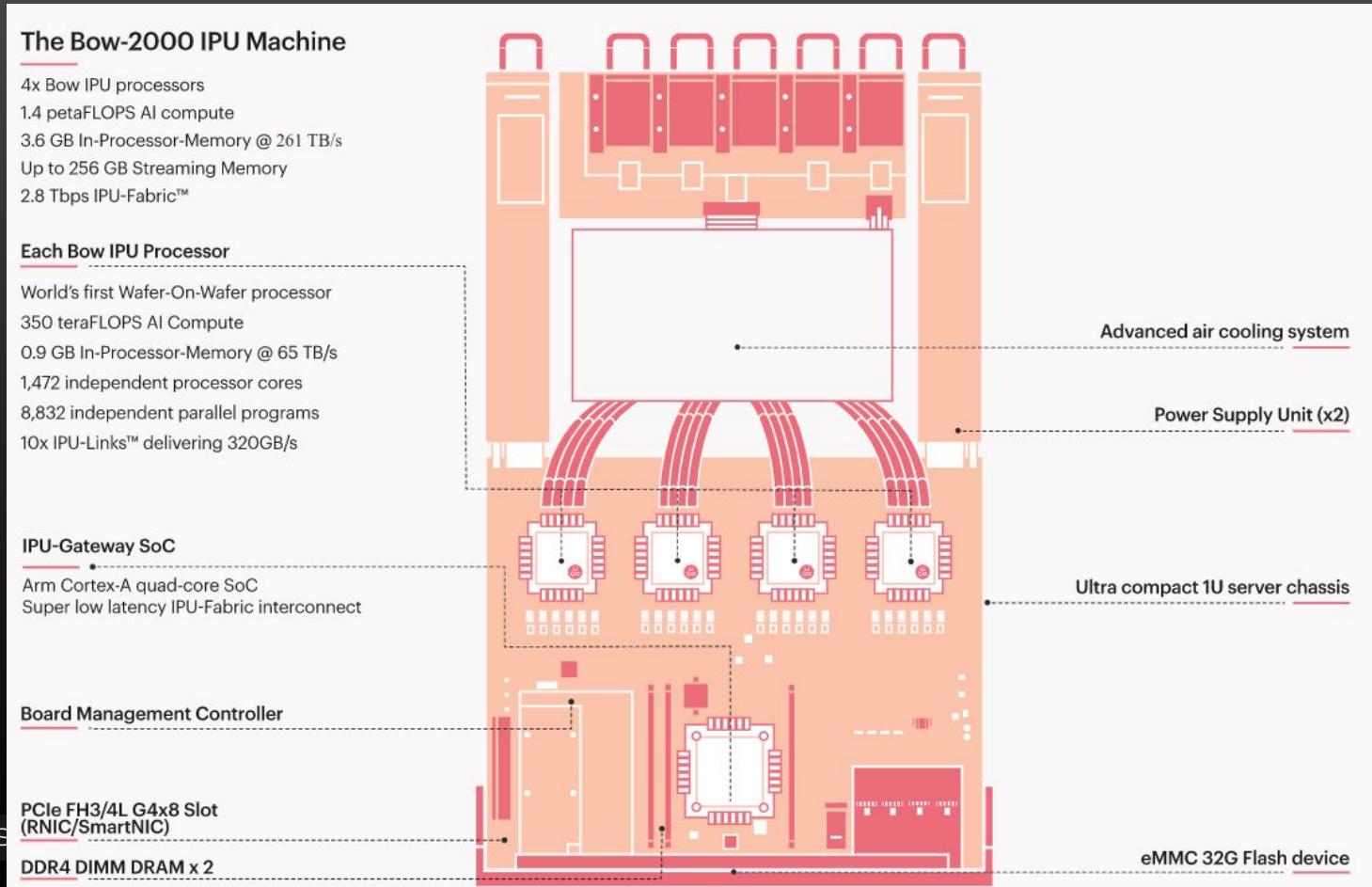
The compute

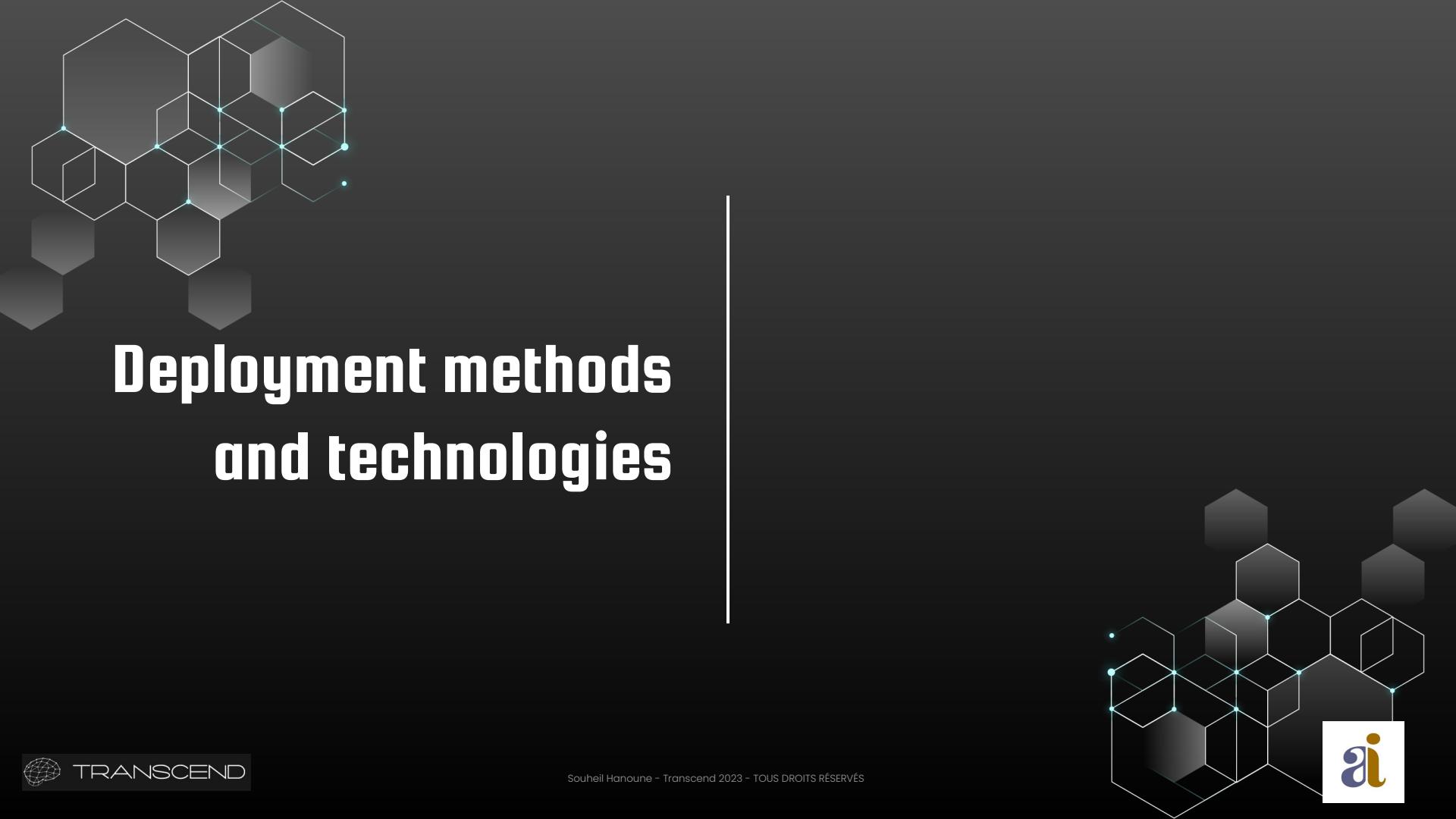


Specific AI Hardware : GraphCore case study (IPU)



Specific AI Hardware : GraphCore case study (IPU)





Deployment methods and technologies

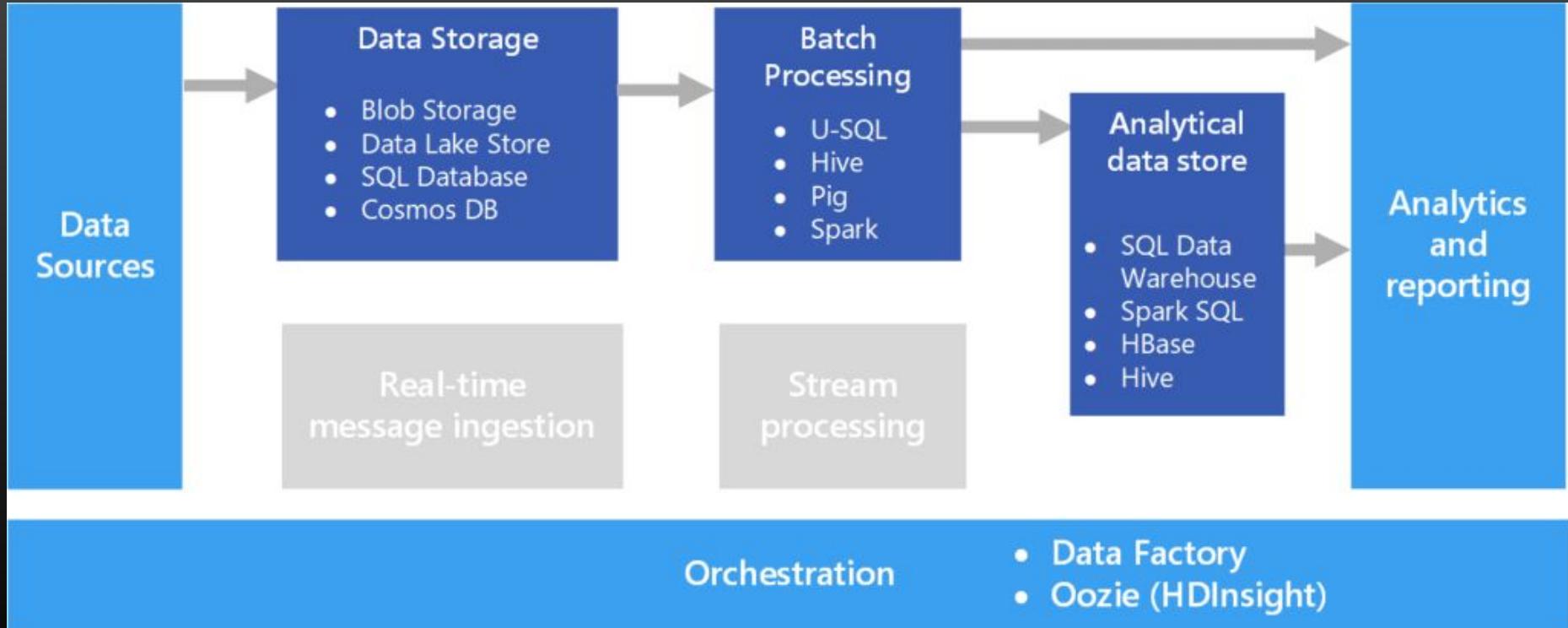
Methods to deploy depending on model update

- **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

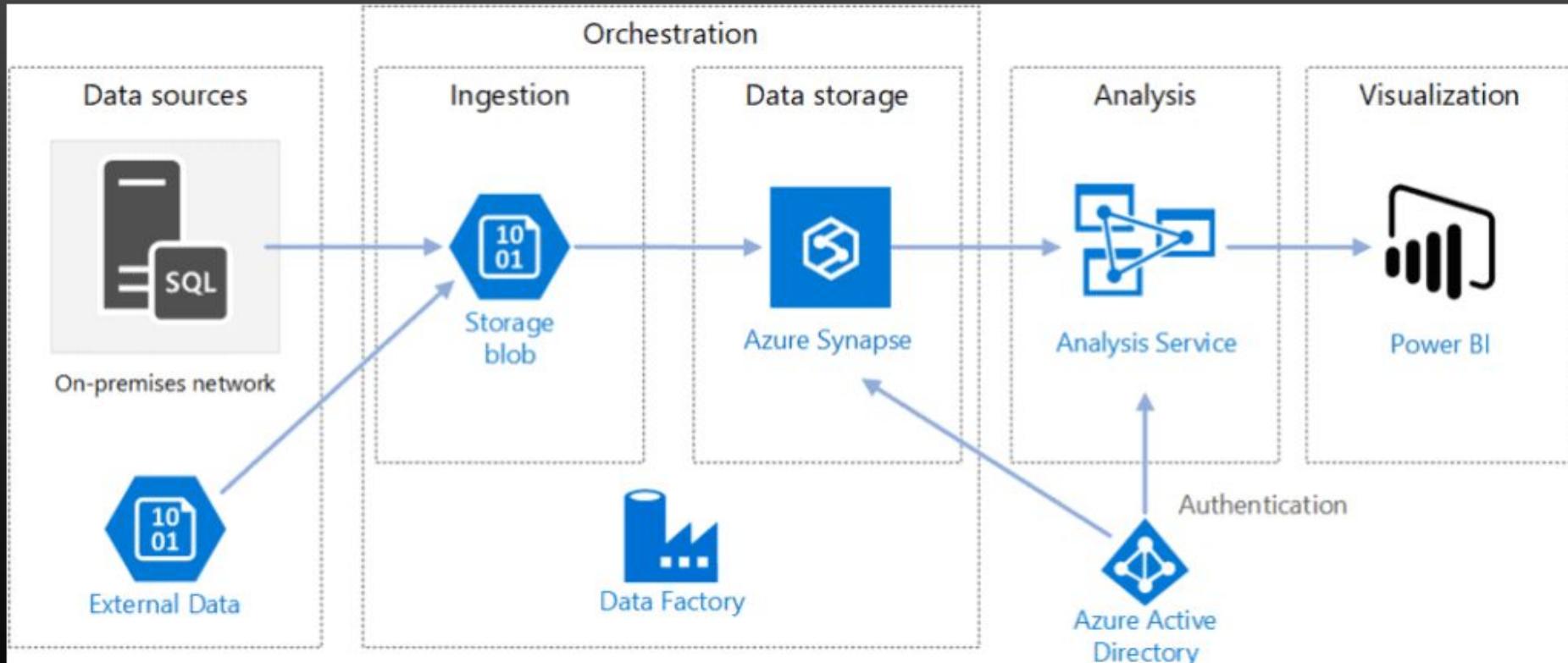
Methods to deploy a model for data consumption

- **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

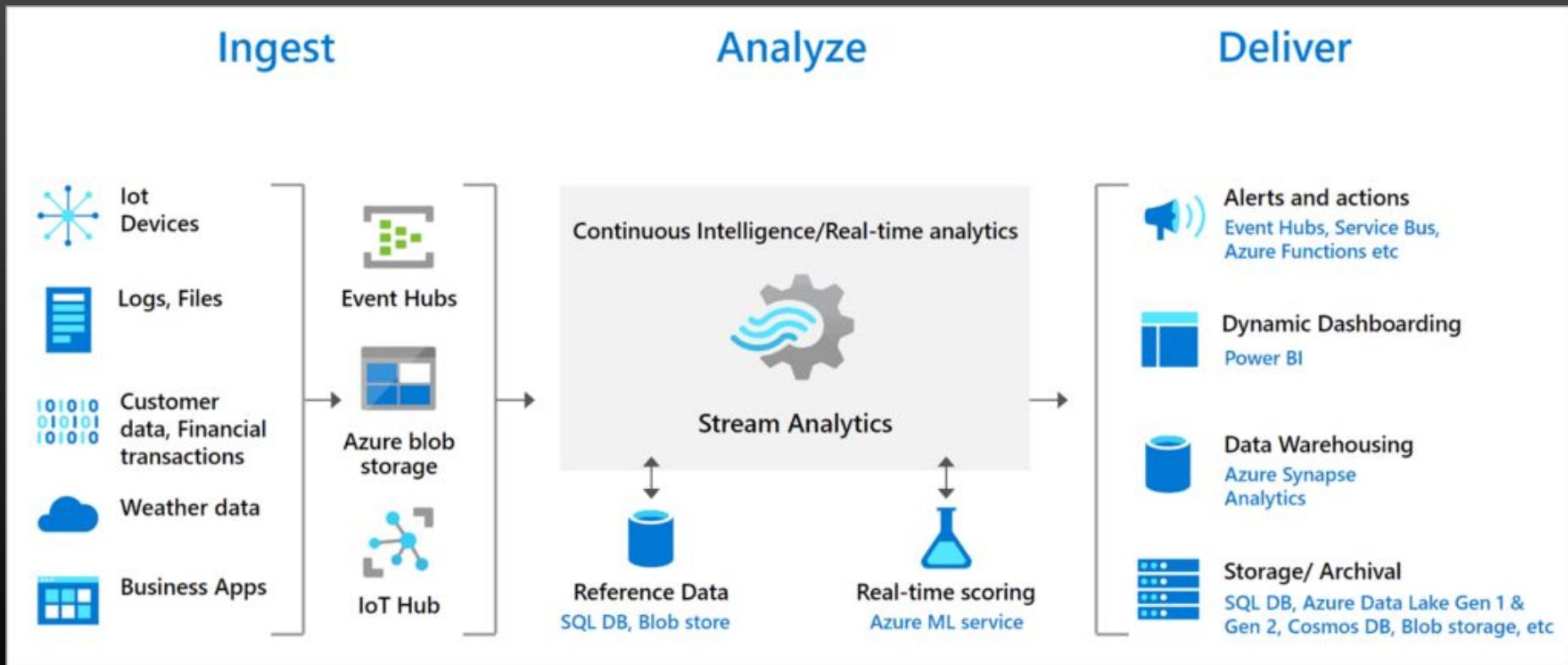
Batch & stream processing architecture



Example of batch processing architecture on Azure



Example of stream processing architecture on Azure

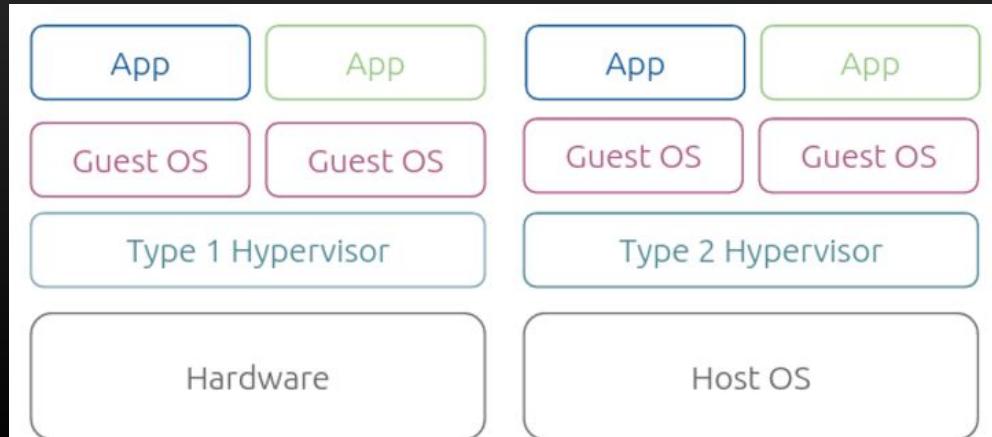
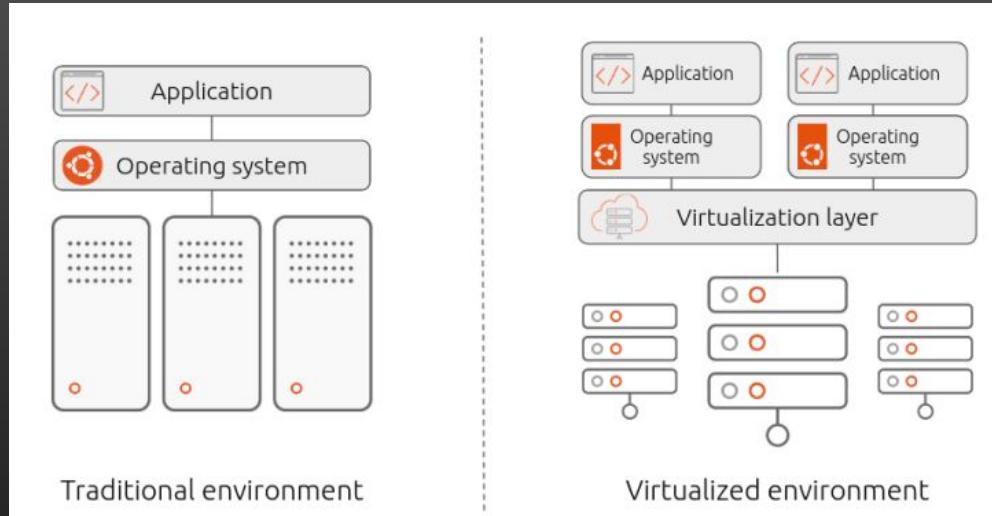


Key technologies used in AI deployment

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

Virtualization

- 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



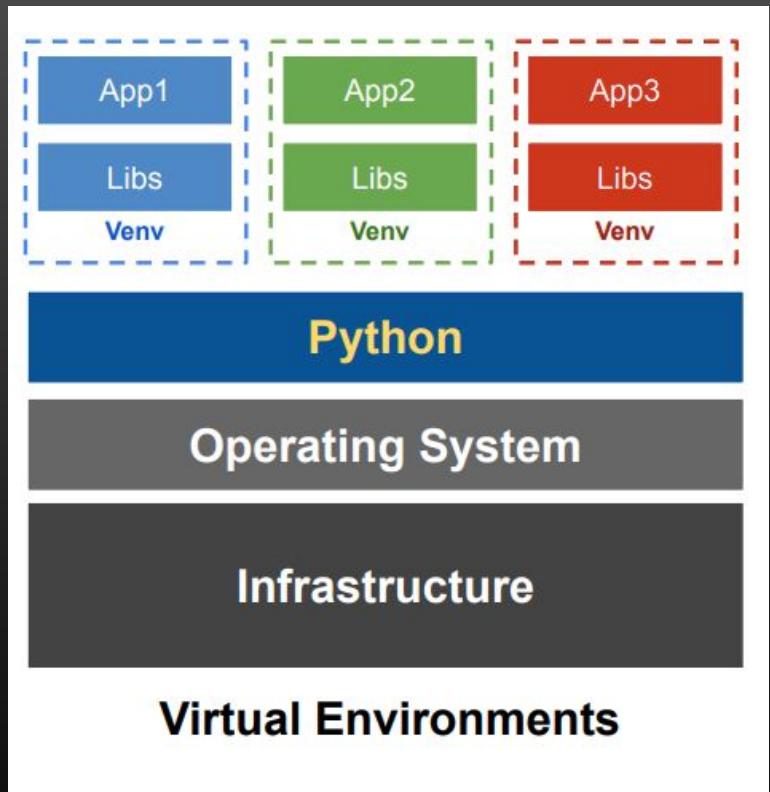
Virtual Environments

Pros

- Reproducible results
- Explicit dependencies
- Improved engineering collaboration

Cons

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



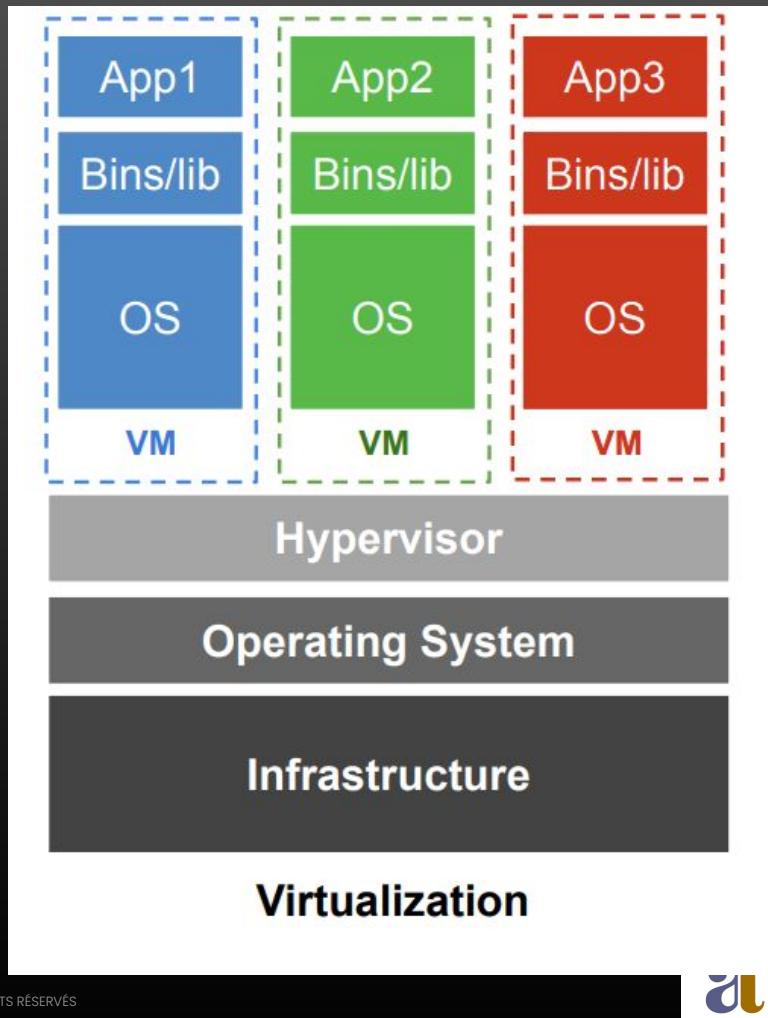
Virtual Machines (Virtualization)

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)

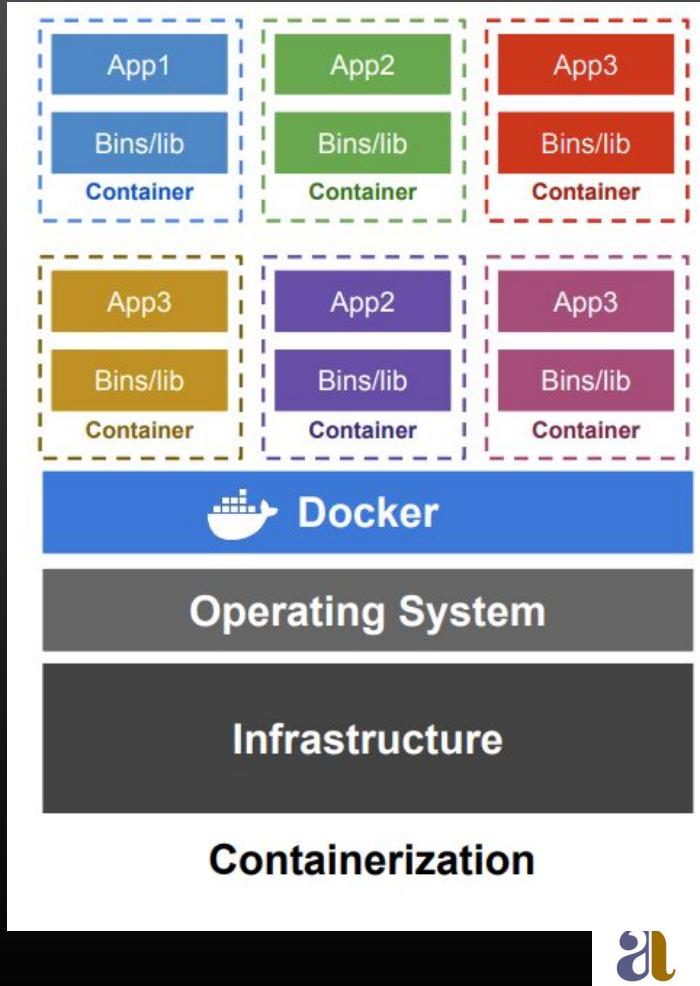
We want a system that:

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

What's a container ?

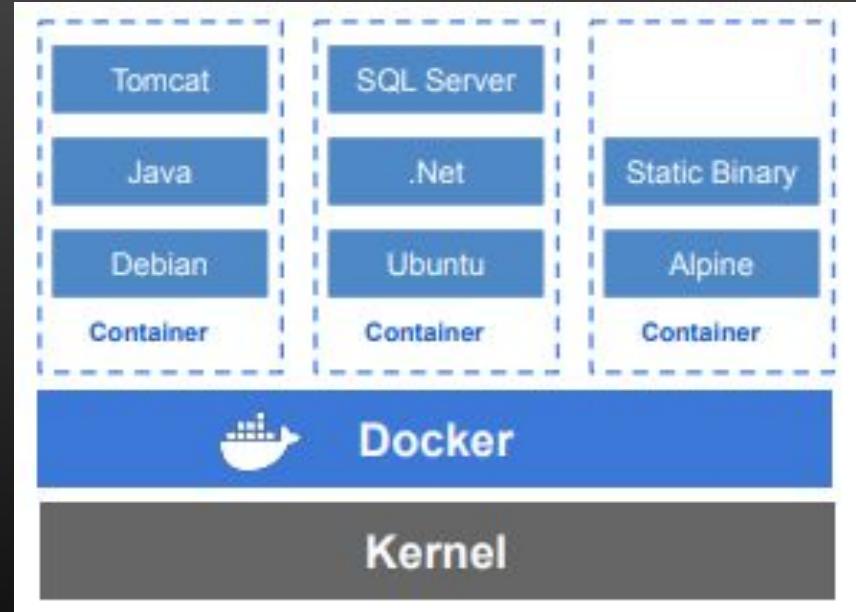
- Extremely **portable** and lightweight
- **Fully packaged** software with all dependencies included
- Can be used for **development, training, and deployment**
- Development teams can easily **share** containers

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



Containerization (Docker)

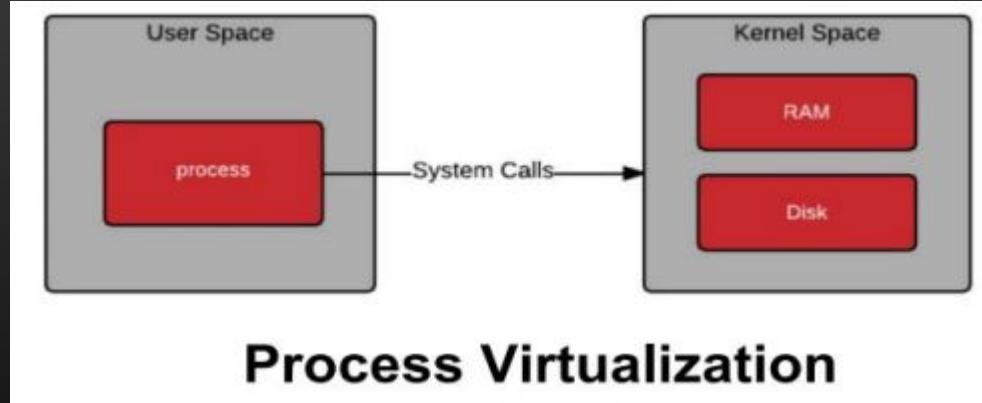
- **Standardized** packaging for software dependencies
- **Isolate** apps from each other
- **Works** for all major Linux distributions, MacOS, Windows



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

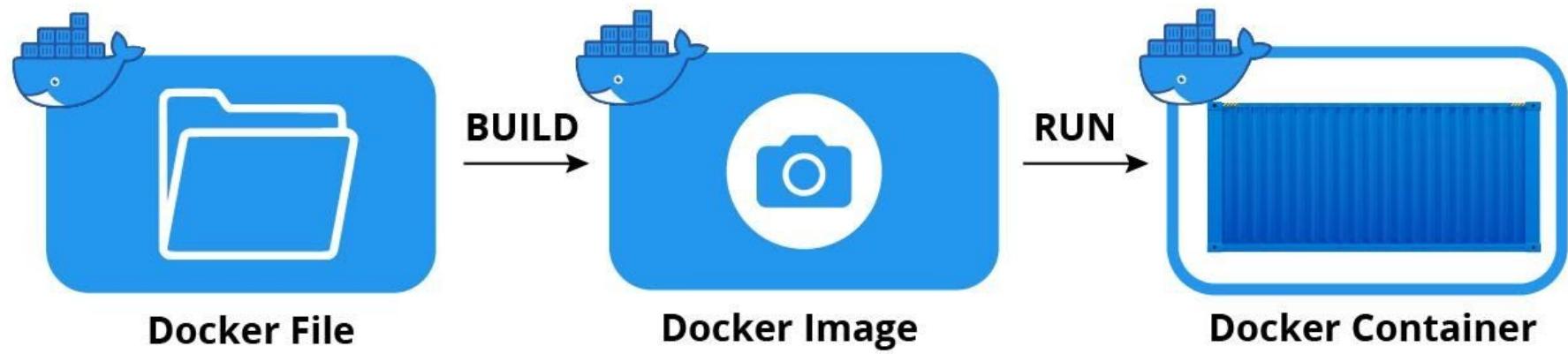
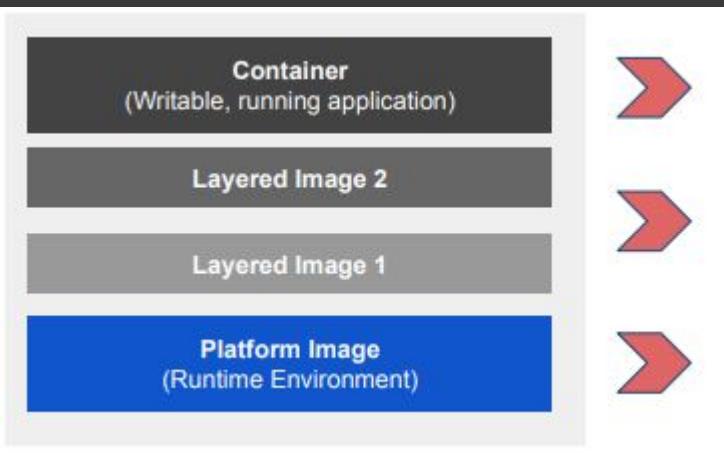


Image layering



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

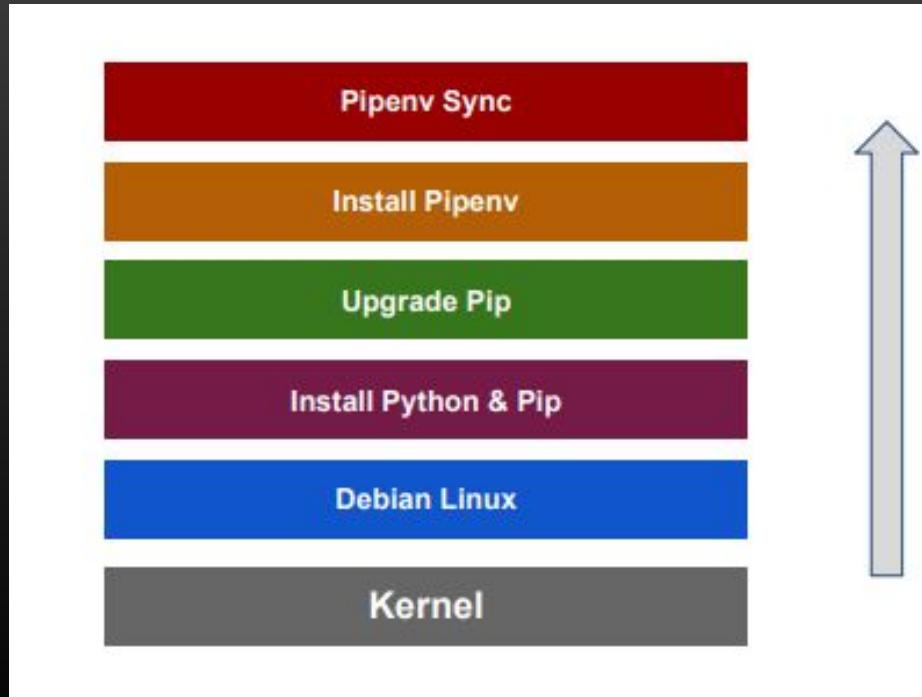
- Layer images are read-only
- Each image depends on one or more parent images

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

Images

How we store the application

Docker Container

The standard unit in which the application service resides and executes

Containers

How we run the application

Docker Engine

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

Engine

Where we run the application

Registry Service (Docker Hub or Docker Trusted Registry)

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

Registry

Where we store the application

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

```
docker build -t ac215-d1 -f Dockerfile .
```

Build the image

Name of the dockerfile in current directory “.”

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
```



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.

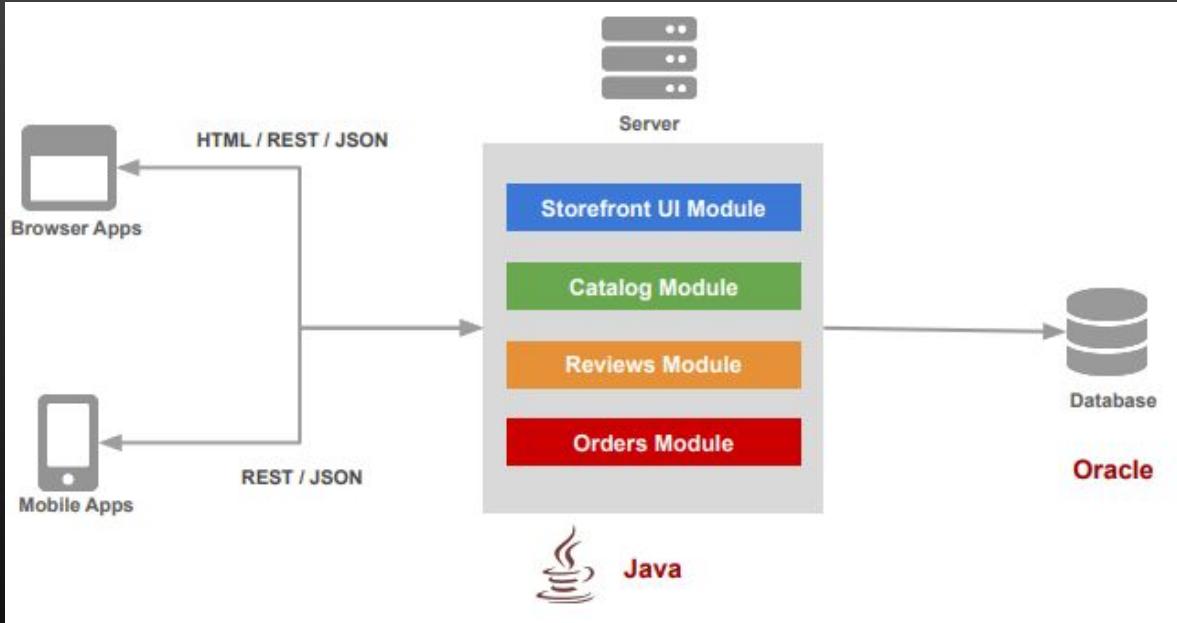
We will mainly focus on :

- Monolithic architecture
- Microservice architecture

Monolithic Architecture

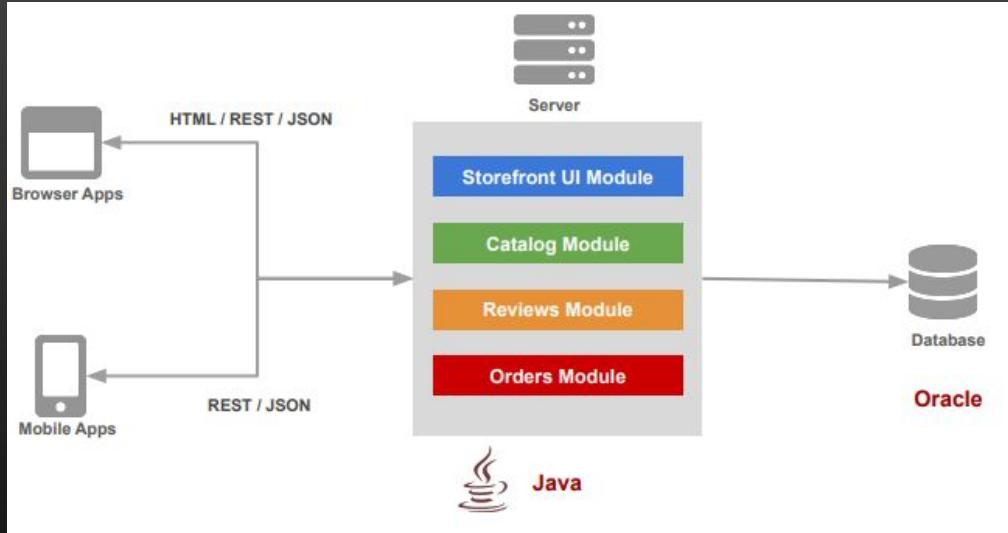
Simple to **Develop, Test, Deploy and Scale**:

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

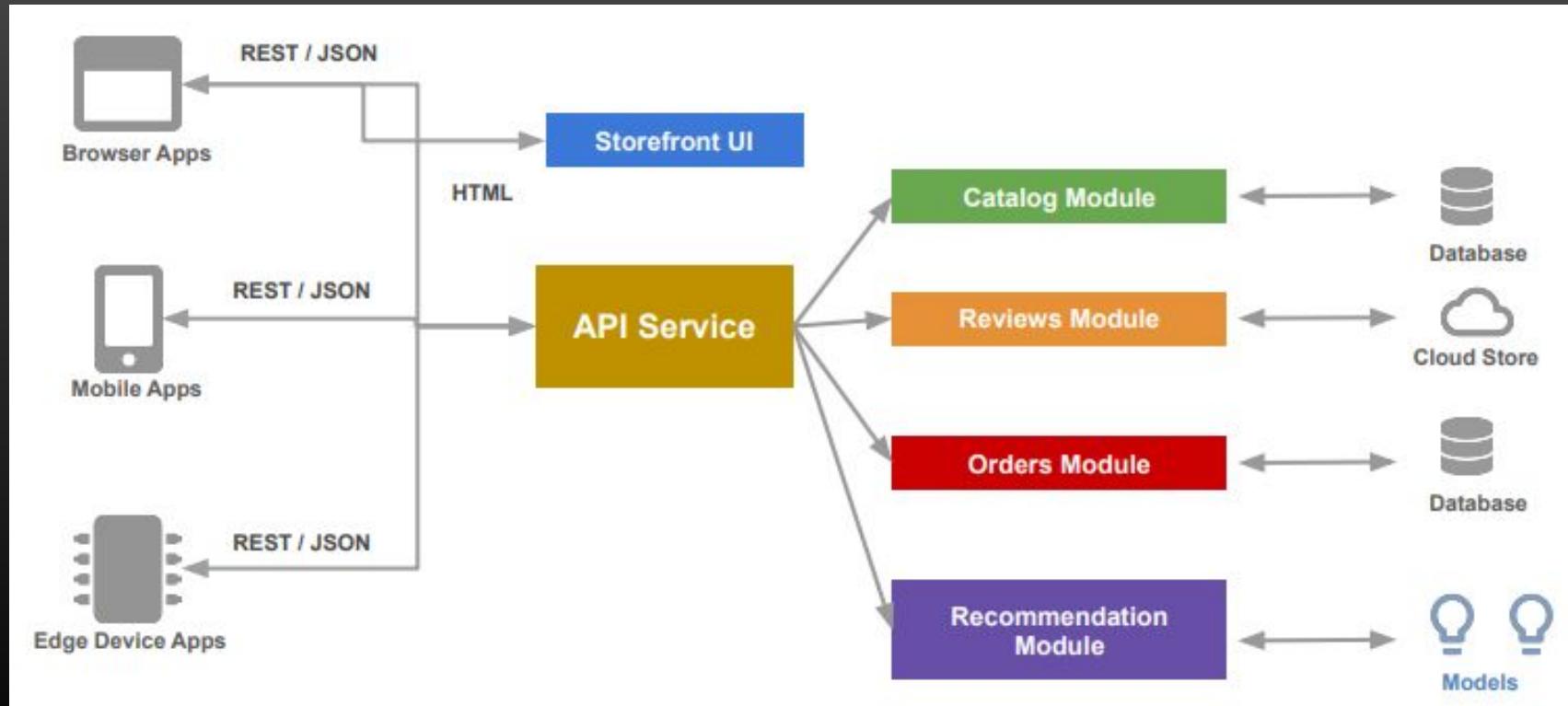
Newer version are deployed often

Deployed to a multitude of servers

Data Science & ML

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

Microservice Architecture

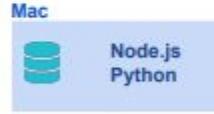
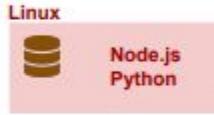
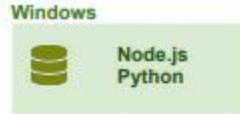


Software Development Workflow (no Docker)



OS Specific **installation** in
every developer machine

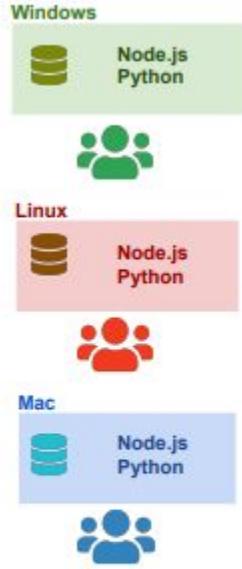
Software Development Workflow (no Docker)



Every team member moves code to source control

OS Specific **installation** in every developer machine

Software Development Workflow (no Docker)



OS Specific **installation** in
every developer machine

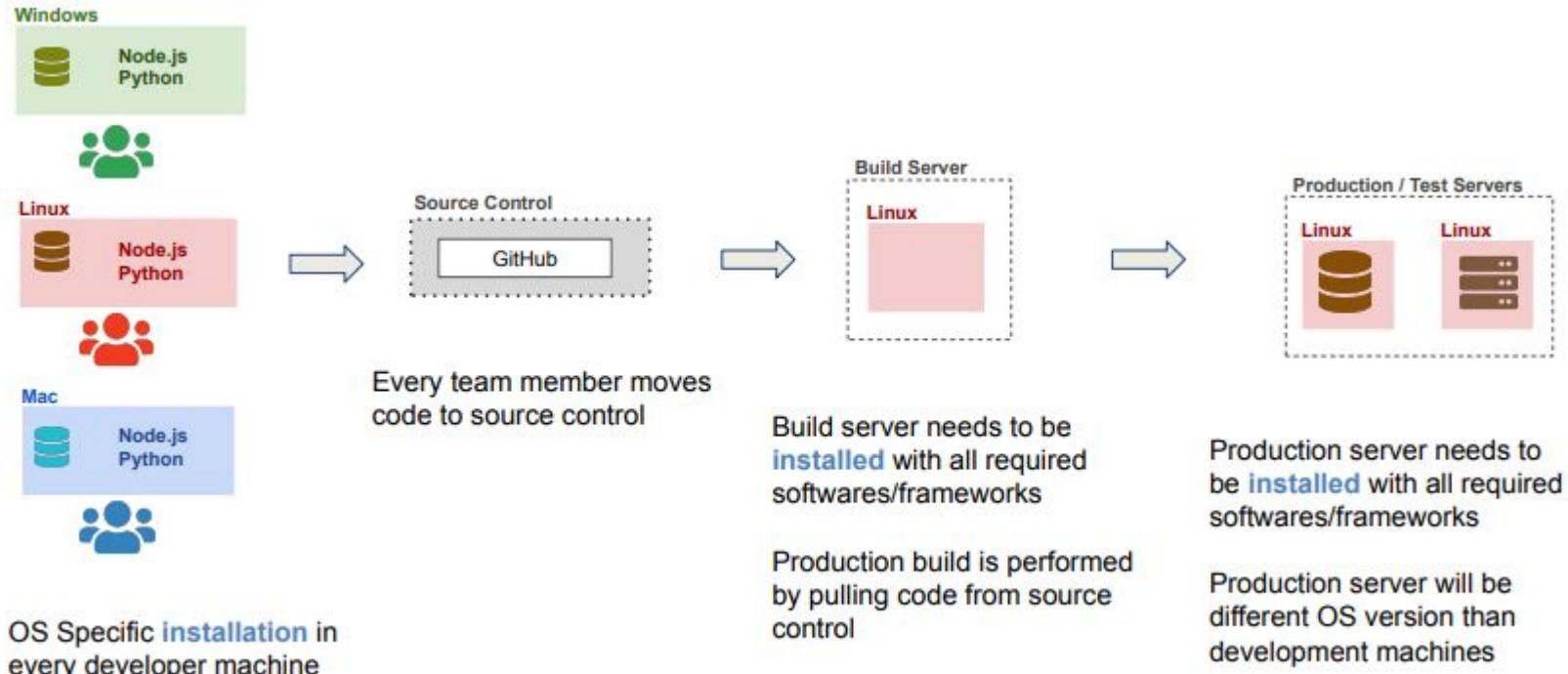
Every team member moves
code to source control

Build server needs to be
installed with all required
softwares/frameworks

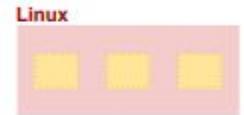
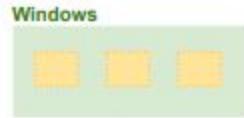
Production build is performed
by pulling code from source
control



Software Development Workflow (no Docker)



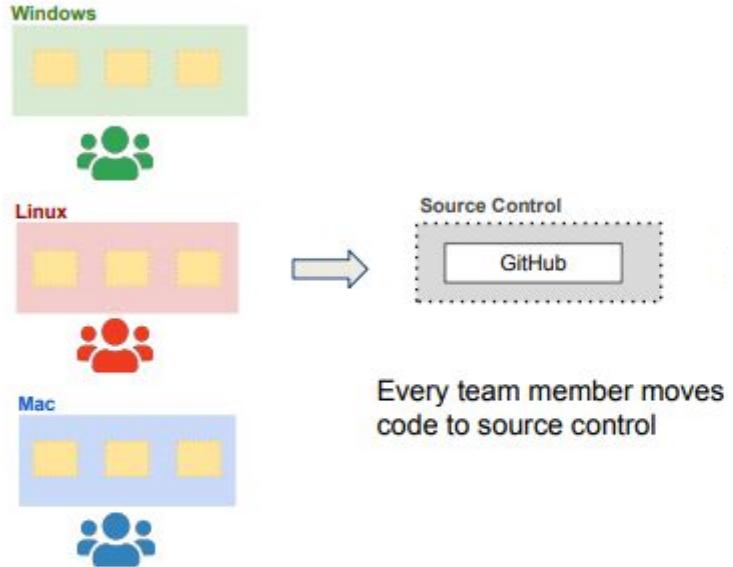
Software Development Workflow (with Docker)



Development machines only
needs **Docker installed**

Containers need to be setup
only once

Software Development Workflow (with Docker)

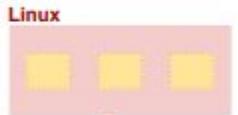
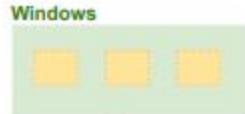


Every team member moves
code to source control

Development machines only
needs **Docker installed**

Containers need to be setup
only once

Software Development Workflow (with Docker)

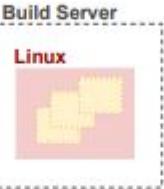


Development machines only needs **Docker installed**

Containers need to be setup only once



Every team member moves code to source control

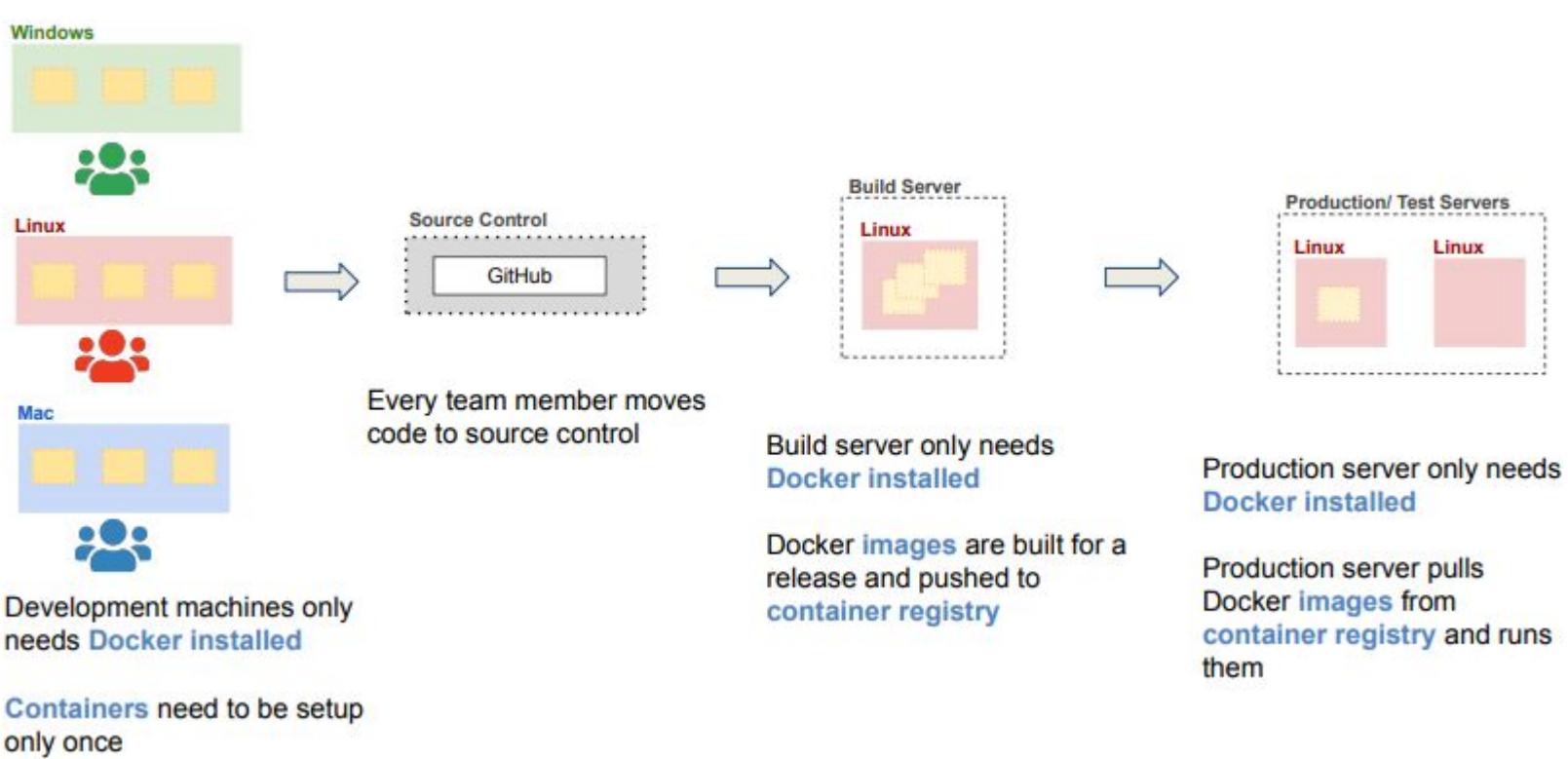


Build server only needs **Docker installed**

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

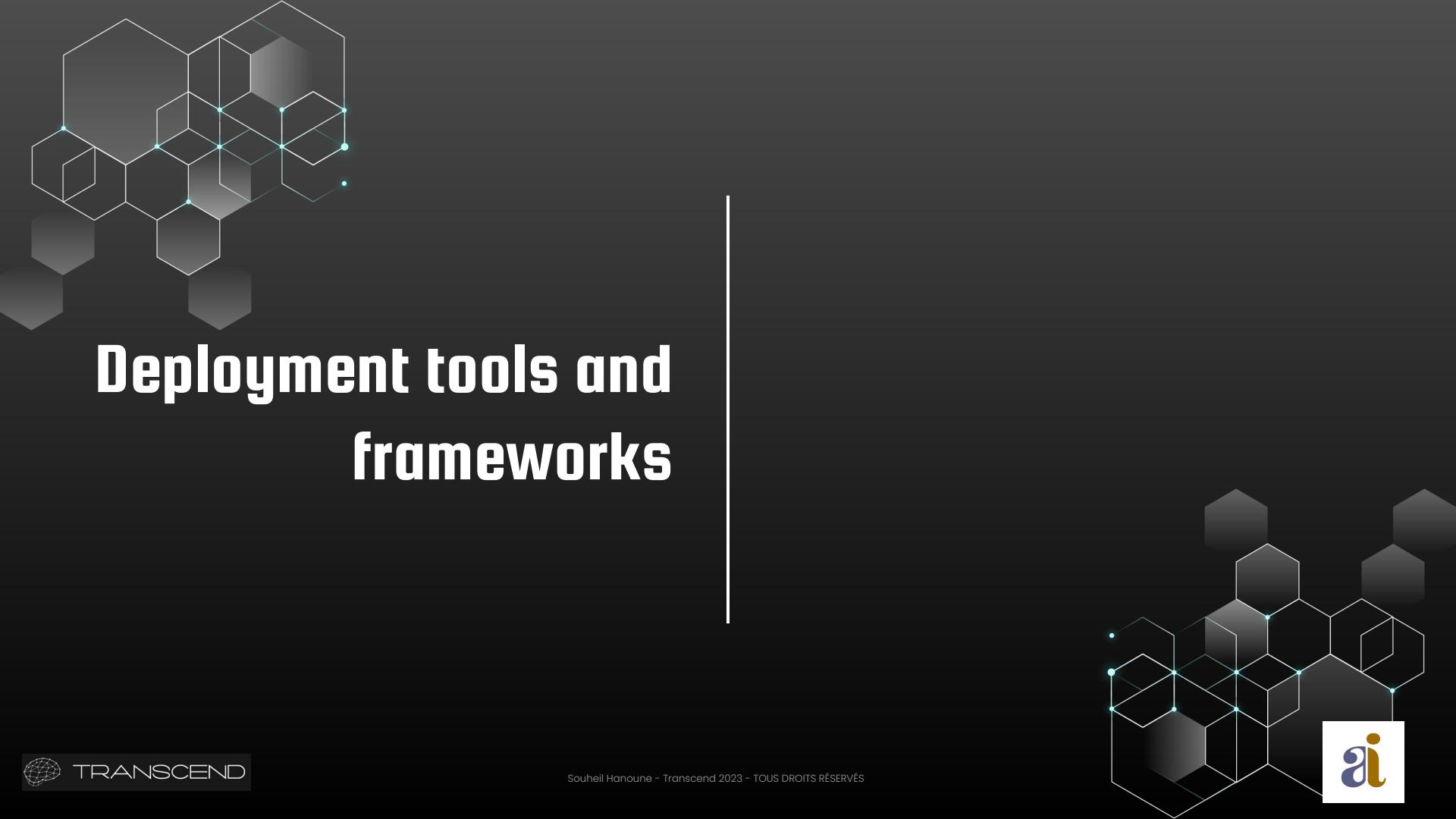


Software Development Workflow (with Docker)



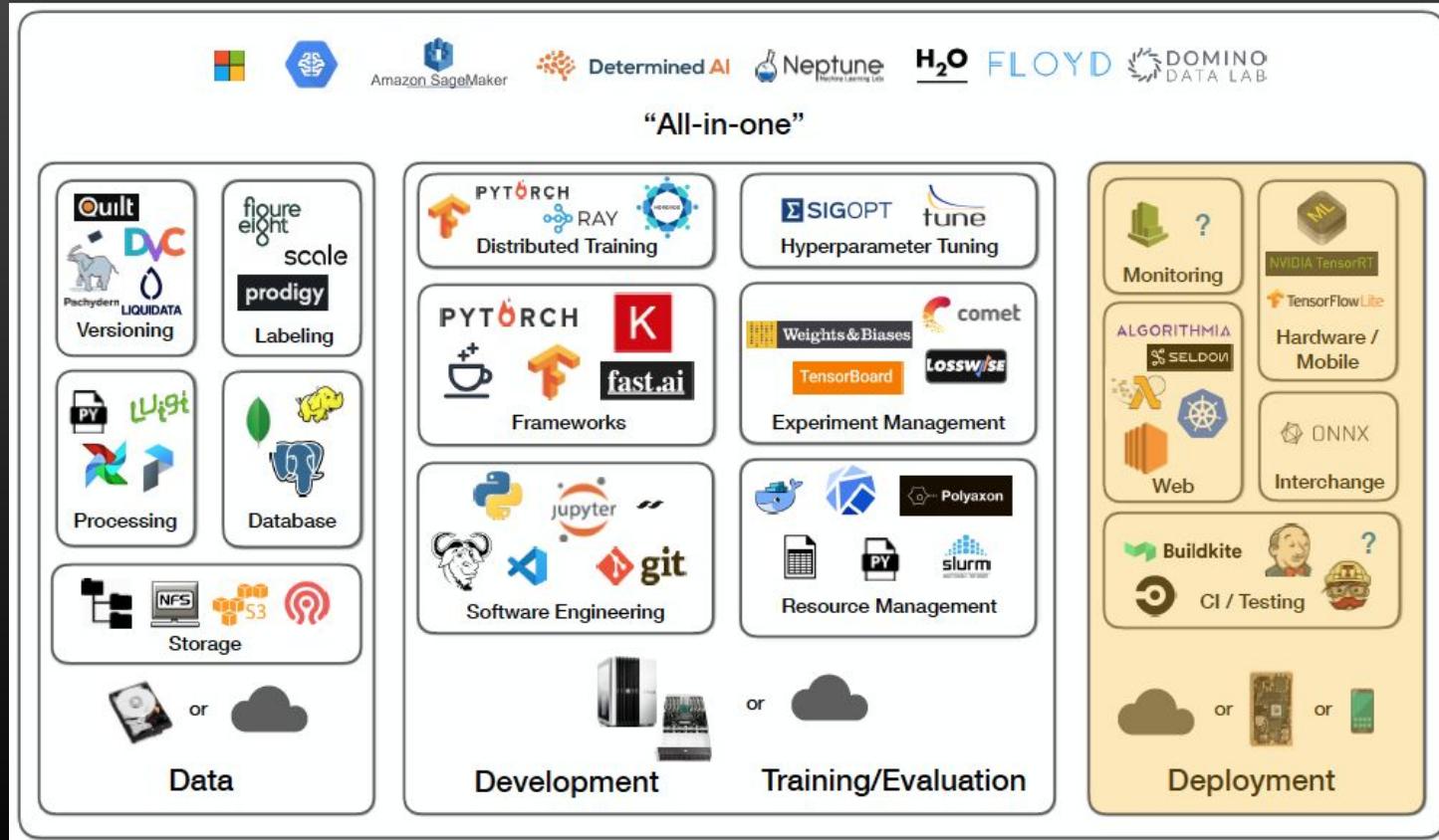
Software Development Workflow (with Docker)

	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

AI ecosystem and tools

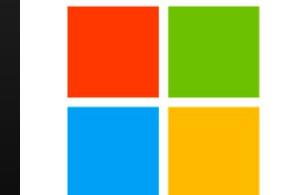


ML frameworks and deployment tools

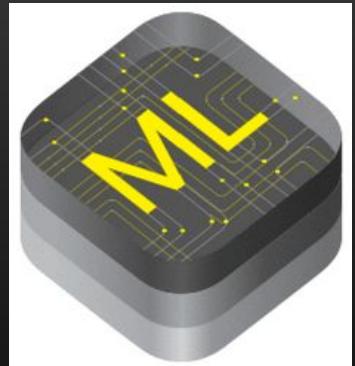
Machine learning frameworks are tools and libraries that allow developers to more easily build ML models or Machine Learning applications, without having to get into the nuts and bolts of the base or core algorithms. It provides more of an end-to-end pipeline for machine learning development.



TensorFlow



Microsoft
CNTK



DL frameworks and deployment tools

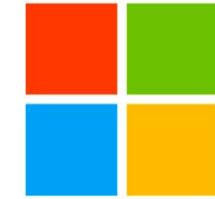
Deep learning frameworks are the machine learning frameworks that are specialized in deep learning applications.



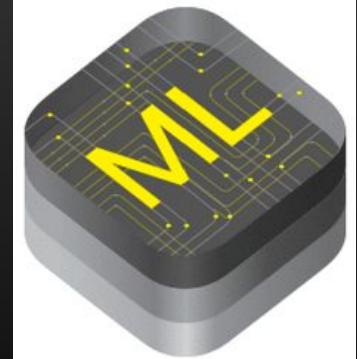
TensorFlow



PyTorch



Microsoft
CNTK



ML Model Serving

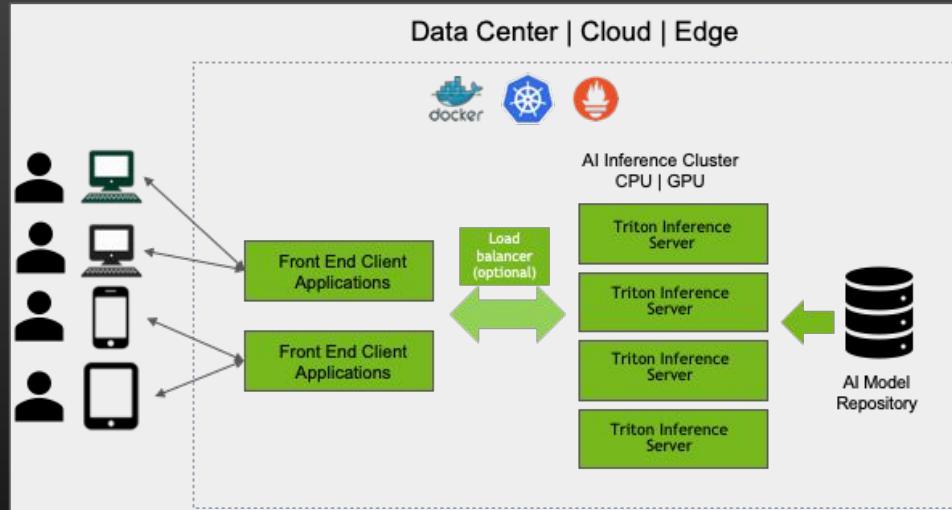
Tools for **model serving** in machine learning can provide you with solutions to many of the data engineers' and DevOp's concerns. They have many functionalities that make it easier to manage your models.

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.

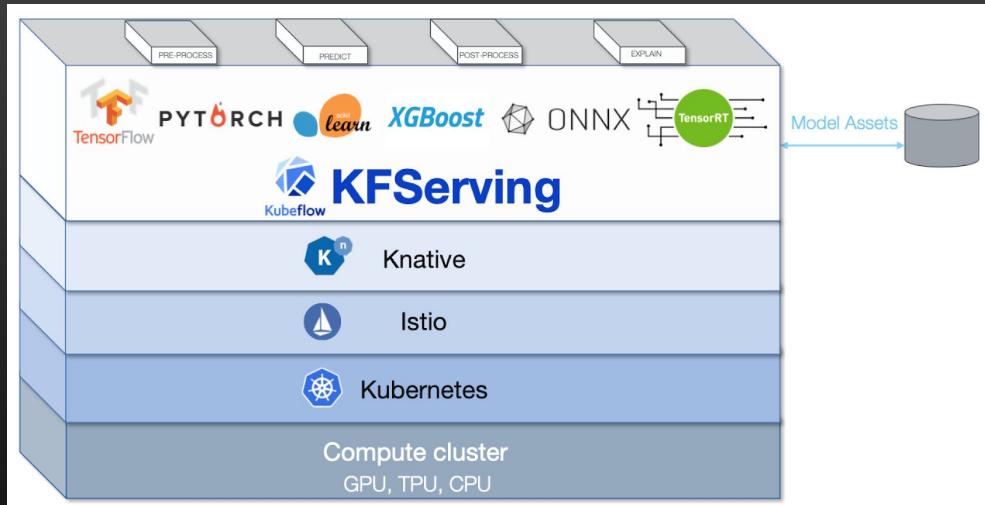
- Supports multiple deep-learning frameworks (TensorRT, TensorFlow GraphDef, TensorFlow SavedModel, ONNX, and PyTorch TorchScript)
- Simultaneous model execution on the same GPU or on multiple GPUs
- Manages stream and batch deployments
- Dynamic batching
- Extensible backends
- Supports model ensemble
- 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.

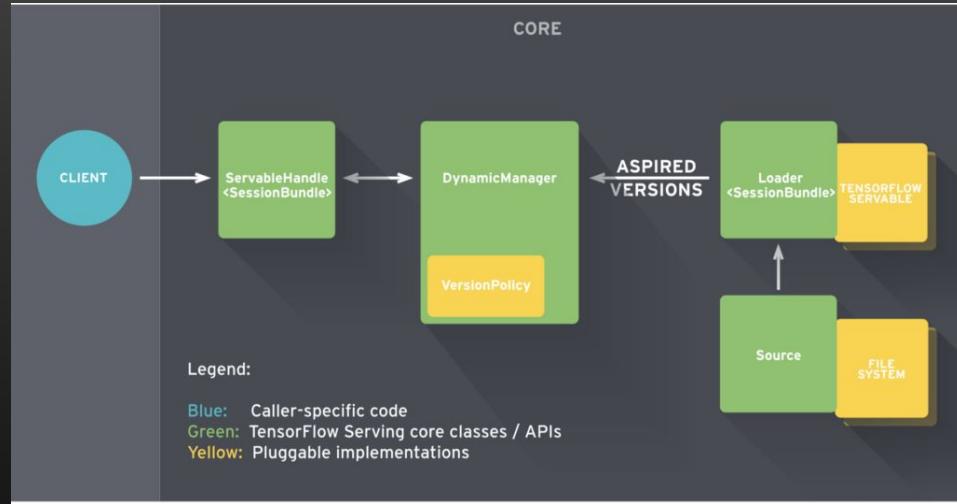
- Provides a simple, pluggable, and complete story for your production ML inference server by providing prediction, pre-processing, post-processing and explainability
- Customizable Inference Service to add your resource requests for CPU, GPU, TPU and memory requests and limits
- Batching individual 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.

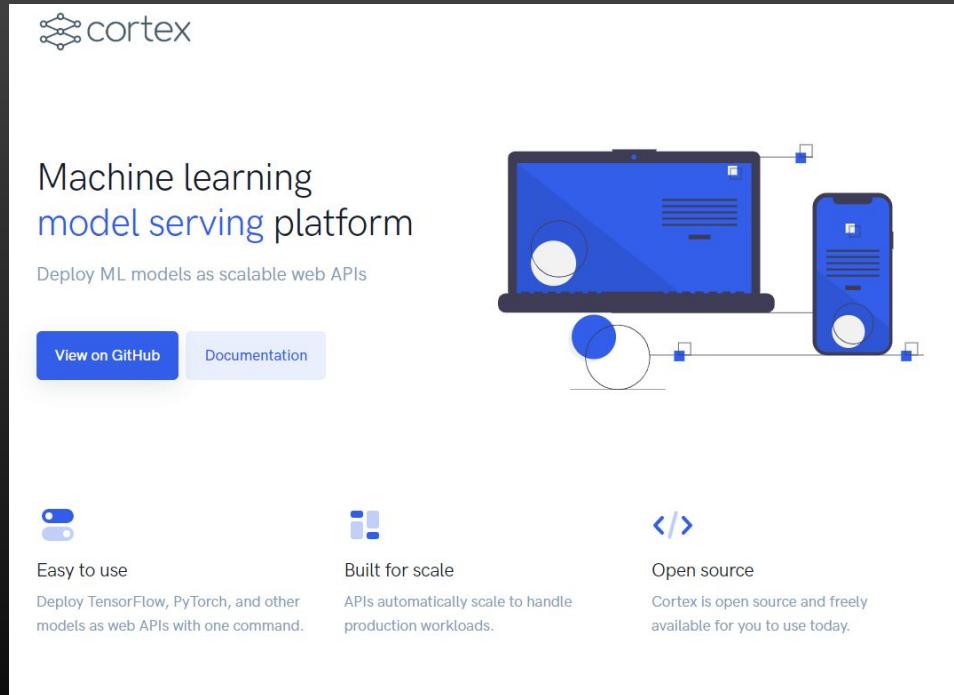
- Can serve multiple models, or multiple versions of the same model at the same time
- Exposes both gRPC and HTTP inference endpoints
- Allows deployment of new model versions without changing your code
- Lets you flexibly test experimental models
- Its efficient, low-overhead implementation adds minimal 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



The screenshot shows the Cortex landing page. At the top is the Cortex logo, which consists of a network icon followed by the word "cortex". Below the logo is the title "Machine learning model serving platform". Underneath the title is the subtitle "Deploy ML models as scalable web APIs". There are two buttons: "View on GitHub" and "Documentation". To the right of the text area is a diagram showing a laptop and a smartphone connected to a central circular node, symbolizing deployment to various devices. Below the diagram are four cards with icons and text: "Easy to use" (with a switch icon), "Built for scale" (with a bar chart icon), and "Open source" (with a code icon). Each card also has a detailed description below its icon.

cortex

Machine learning model serving platform

Deploy ML models as scalable web APIs

[View on GitHub](#) [Documentation](#)

Easy to use

Built for scale

Open source

Deploy TensorFlow, PyTorch, and other models as web APIs with one command.

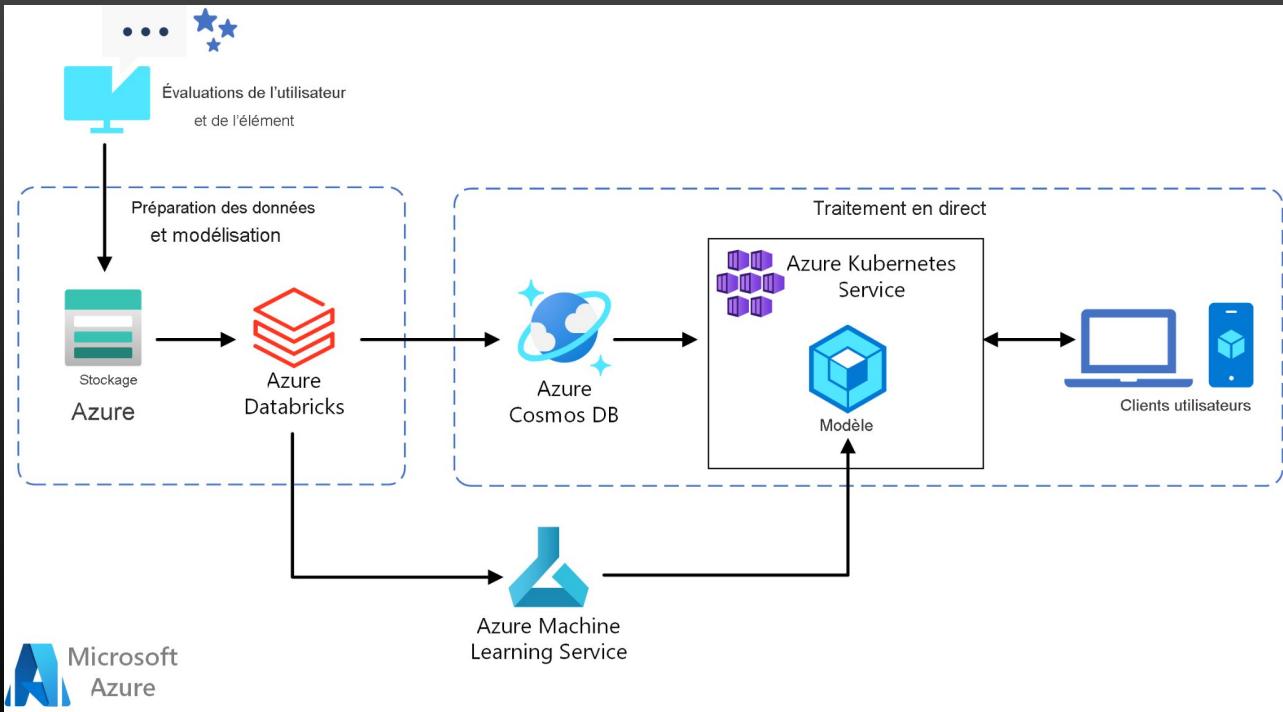
APIs automatically scale to handle production workloads.

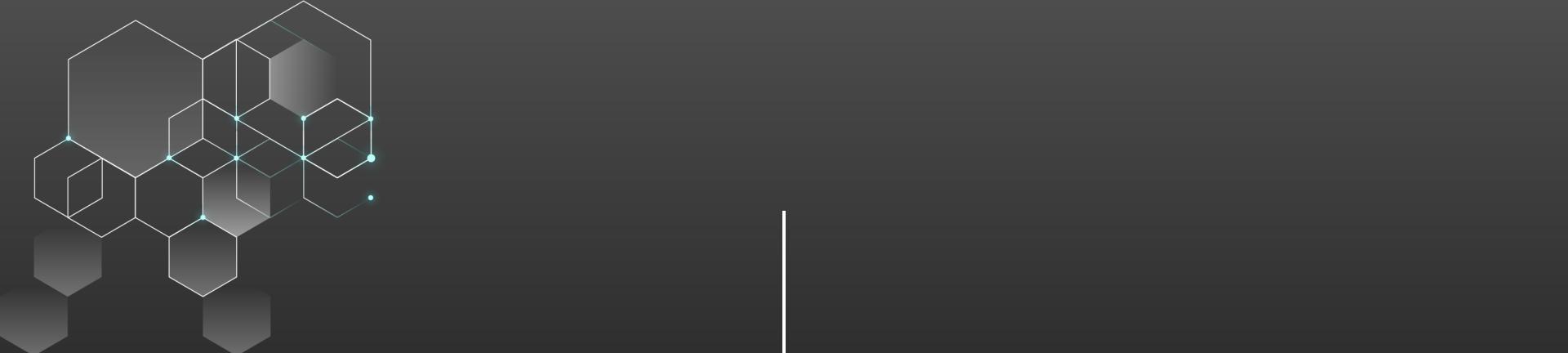
Cortex is open source and freely available for you to use today.

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





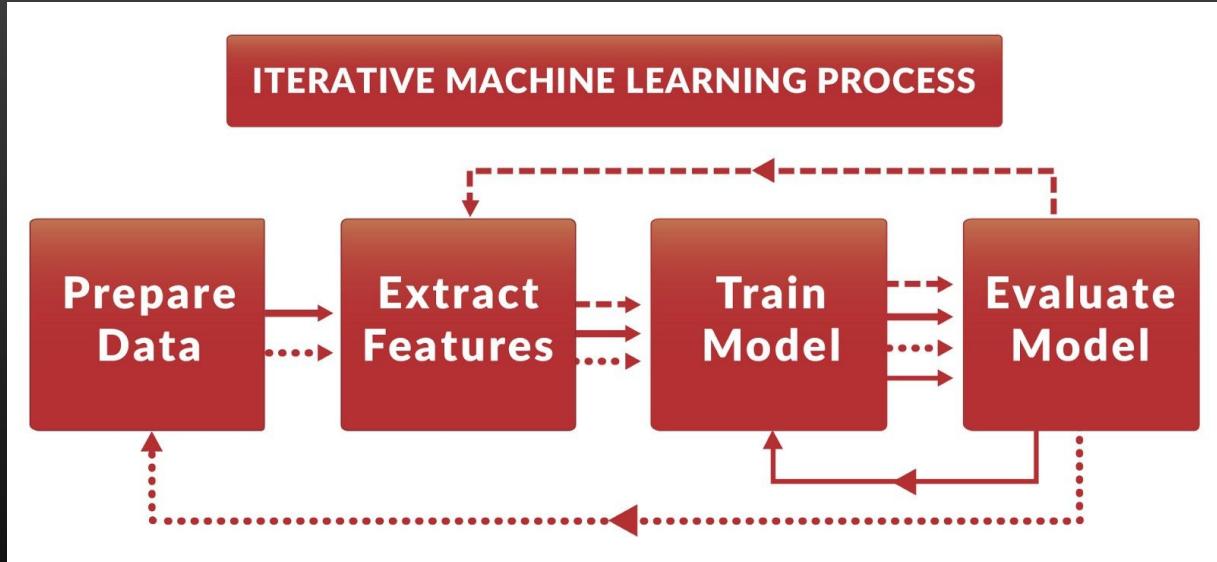
Model management and version control



Version control for ML

For AI models, we need to manage :

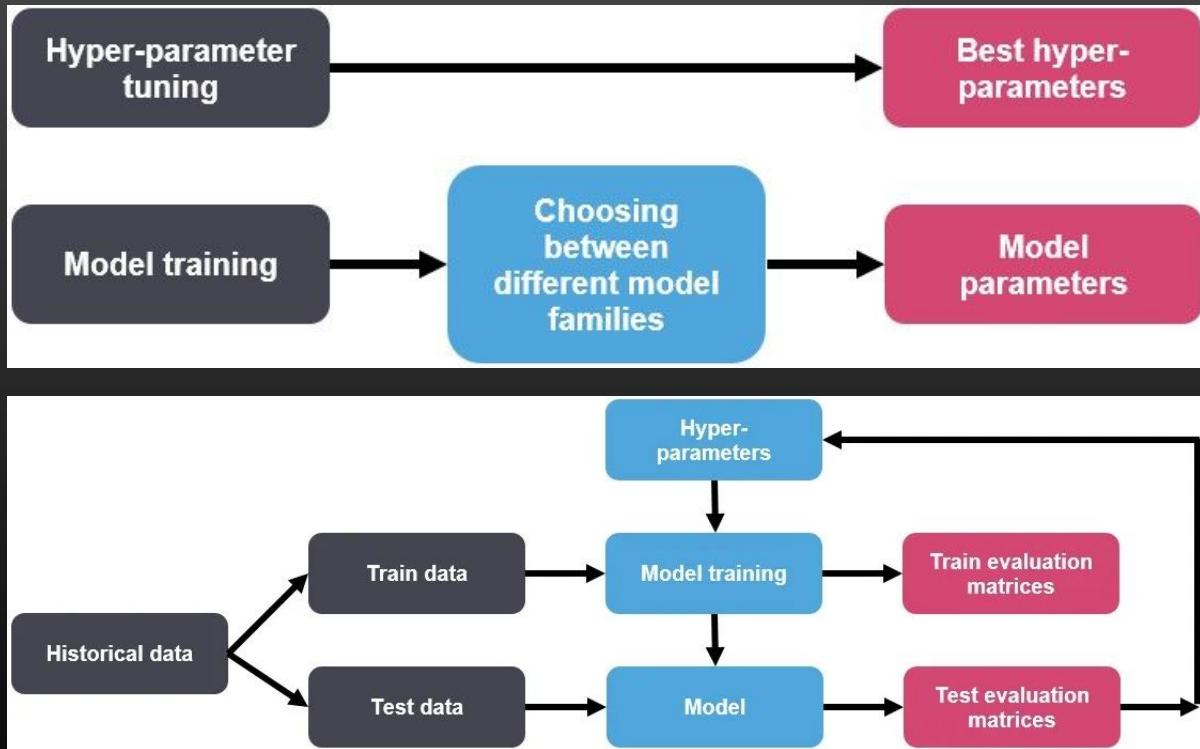
- Code
- Data
- Model



Version control for ML

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

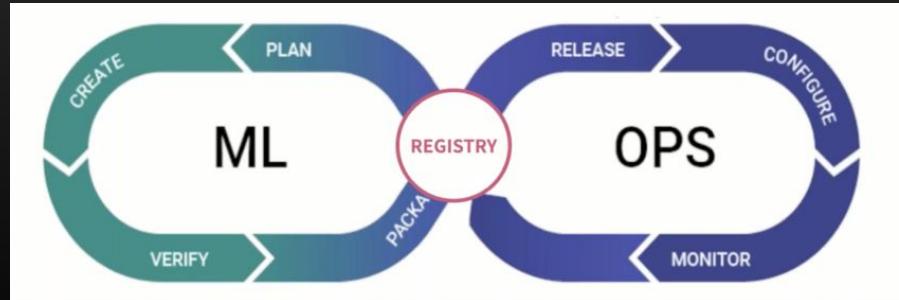
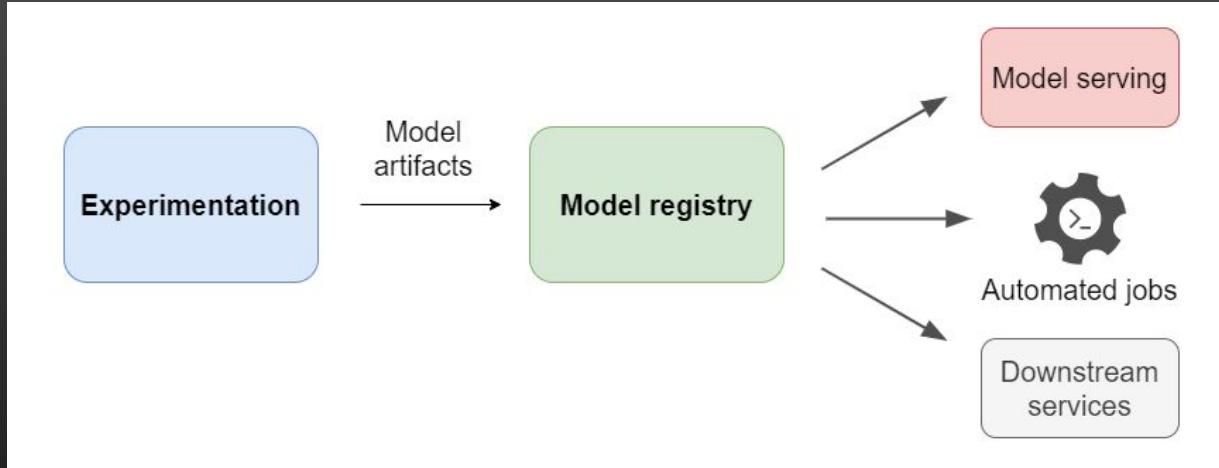
- Collaboration
- Versioning
- Reproducibility
- Dependency tracking
- Model updates



AI Model registry

The model registry provides:

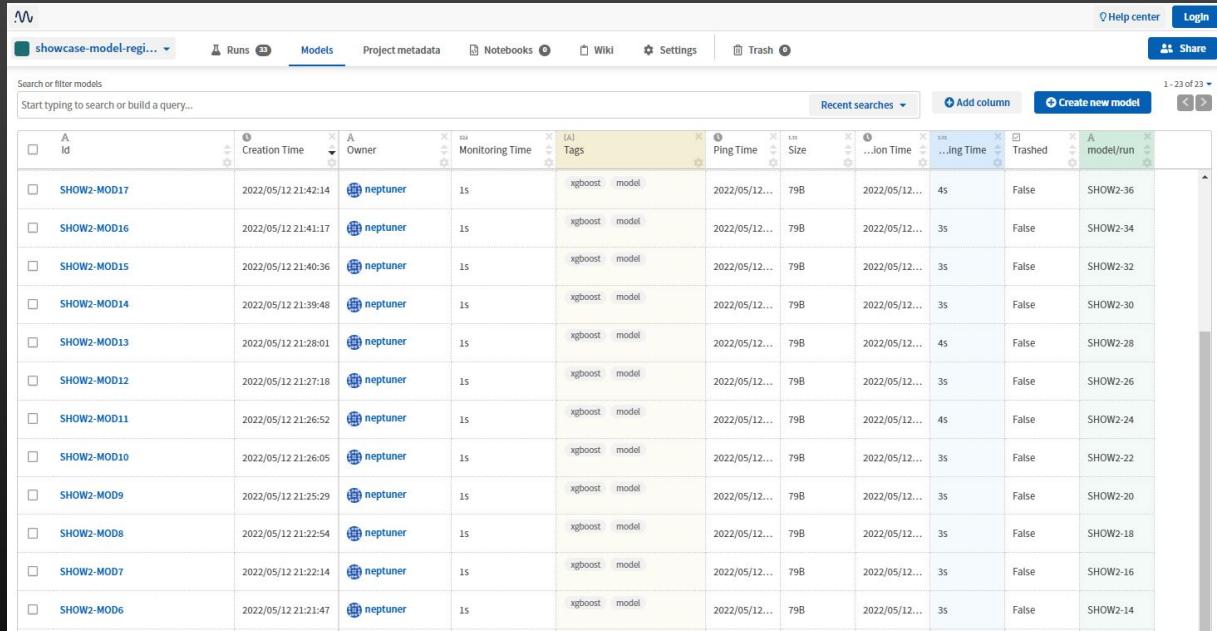
- Centralized storage for all types of models,
- And a collaborative unit for model lifecycle management.



AI Model registry : neptune.ai

The model registry provides:

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

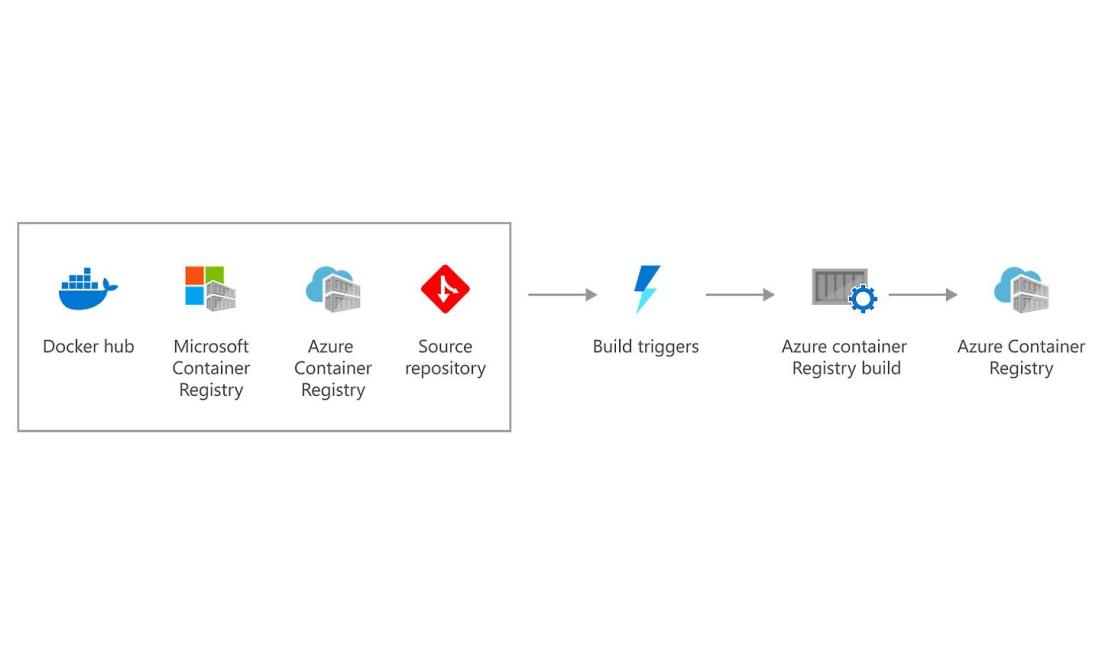


The screenshot shows the Neptune AI Model Registry interface. At the top, there's a navigation bar with tabs for 'Runs' (23), 'Models' (selected), 'Project metadata', 'Notebooks', 'Wiki', 'Settings', and 'Trash'. Below the navigation bar is a search bar with placeholder text 'Start typing to search or build a query...'. To the right of the search bar are buttons for 'Recent searches', 'Add column', and 'Create new model'. The main area is a table with 12 columns: Id, Creation Time, Owner, Monitoring Time, Tags, Ping Time, Size, Duration, Trashed, and a final column for 'model/run'. There are 12 rows in the table, each representing a model named 'SHOW2-MOD17' through 'SHOW2-MOD6'. Each row contains the same data: creation time (e.g., 2022/05/12 21:42:14), owner ('neptuner'), monitoring time (1s), tags ('xgboost model'), ping time (2022/05/12...), size (79B), duration (4s), trashed status (False), and a link to 'SHOW2-36'.

A	A	Creation Time	A	Owner	Monitoring Time	(A)	Tags	Ping Time	Size	Duration	Trashed	A
□	SHOW2-MOD17	2022/05/12 21:42:14	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-36	
□	SHOW2-MOD16	2022/05/12 21:41:17	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-34	
□	SHOW2-MOD15	2022/05/12 21:40:36	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-32	
□	SHOW2-MOD14	2022/05/12 21:39:48	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-30	
□	SHOW2-MOD13	2022/05/12 21:28:01	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-28	
□	SHOW2-MOD12	2022/05/12 21:27:18	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-26	
□	SHOW2-MOD11	2022/05/12 21:26:52	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-24	
□	SHOW2-MOD10	2022/05/12 21:26:05	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-22	
□	SHOW2-MOD09	2022/05/12 21:25:29	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-20	
□	SHOW2-MOD08	2022/05/12 21:22:54	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-18	
□	SHOW2-MOD07	2022/05/12 21:22:14	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-16	
□	SHOW2-MOD06	2022/05/12 21:21:47	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-14	

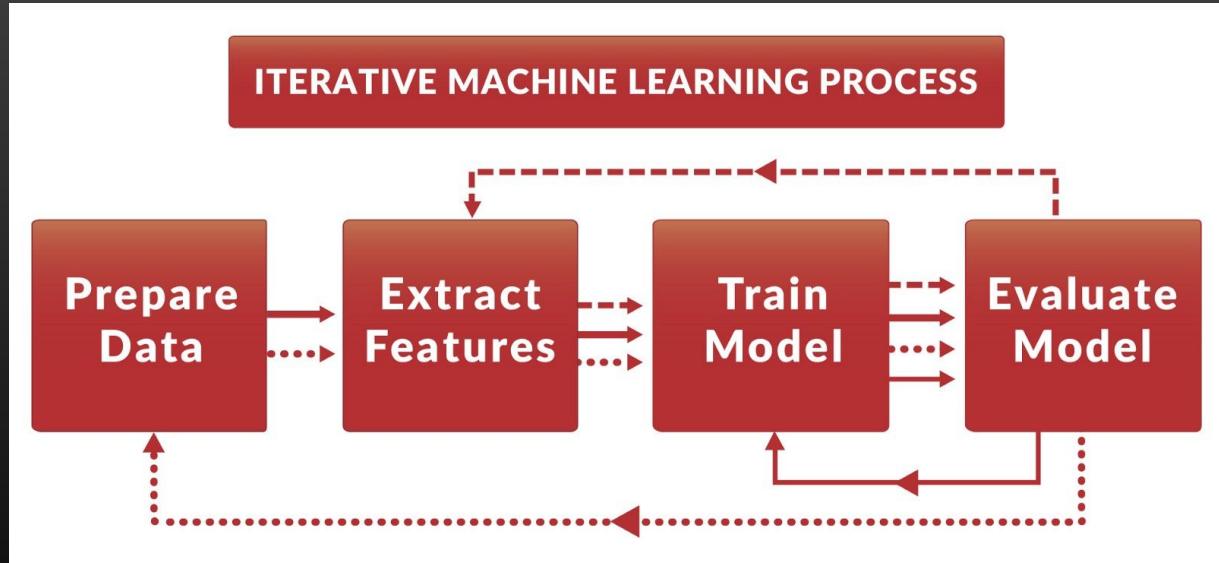
AI Model registry : Azure container registry

- **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



The MLflow logo features the word "mlflow" in a bold, lowercase sans-serif font. The letters are primarily black, except for the "f" which is blue.

Tracking

Record and query experiments: code, data, config, results

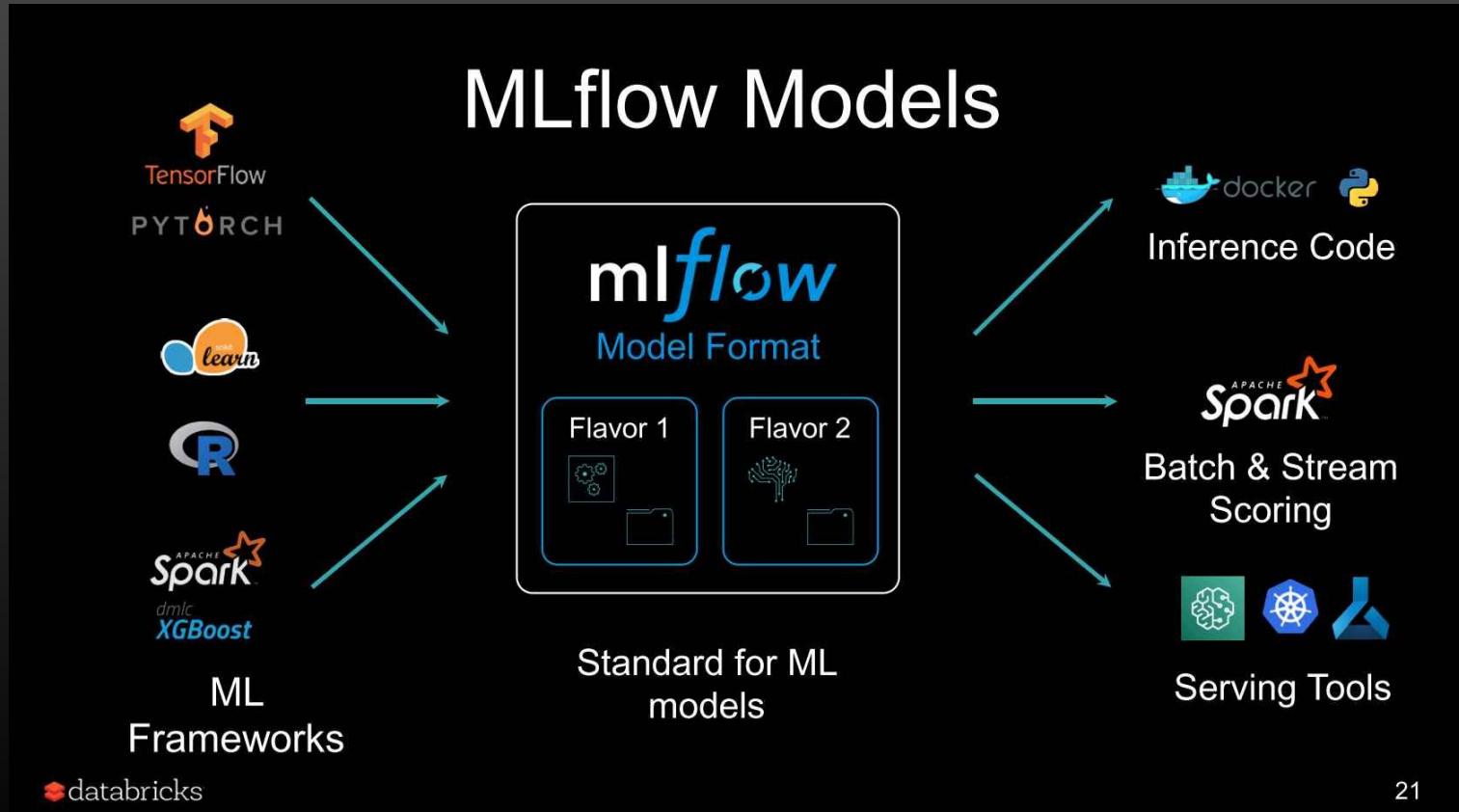
Projects

Packaging format for reproducible runs on any platform

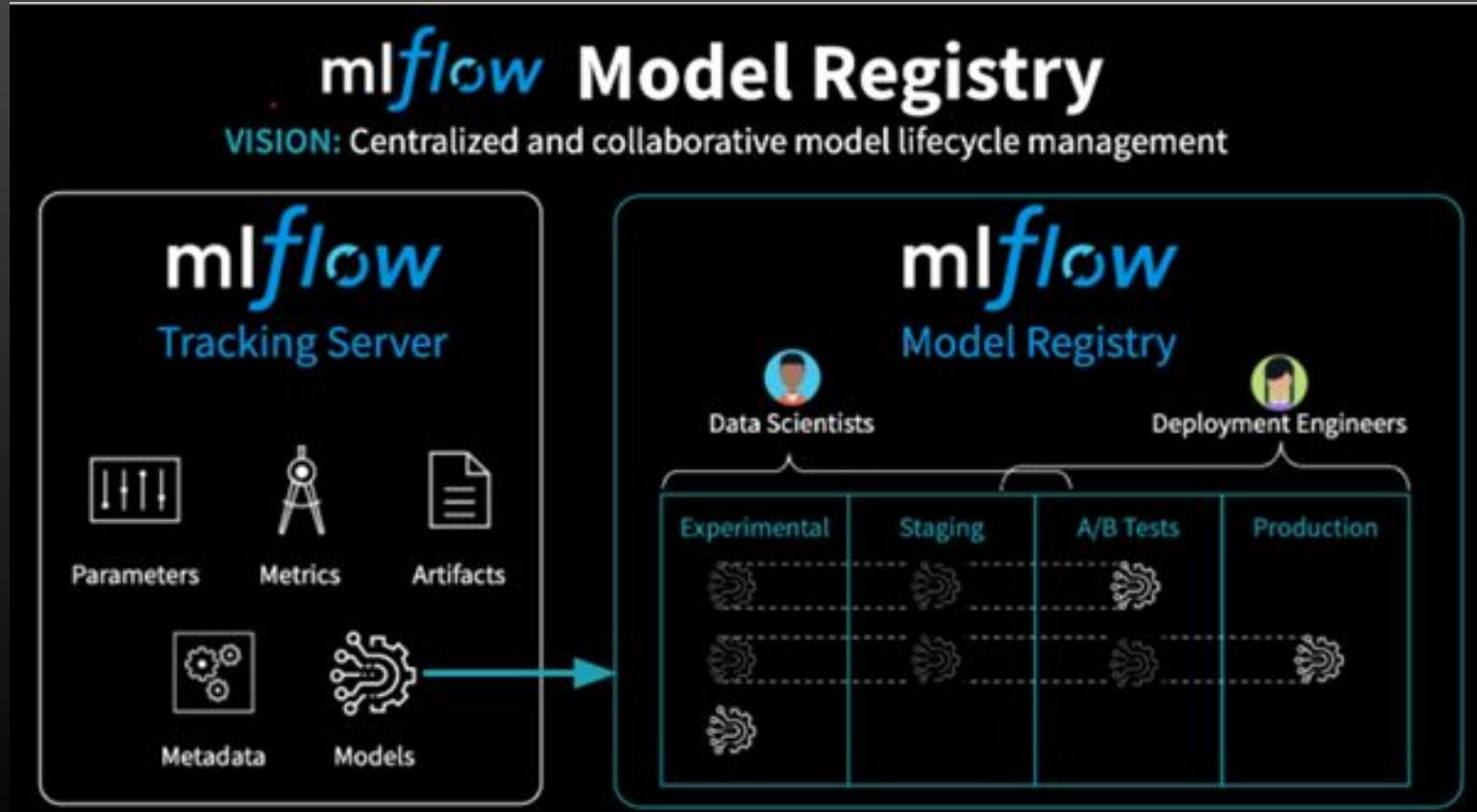
Models

General format for sending models to diverse deploy tools

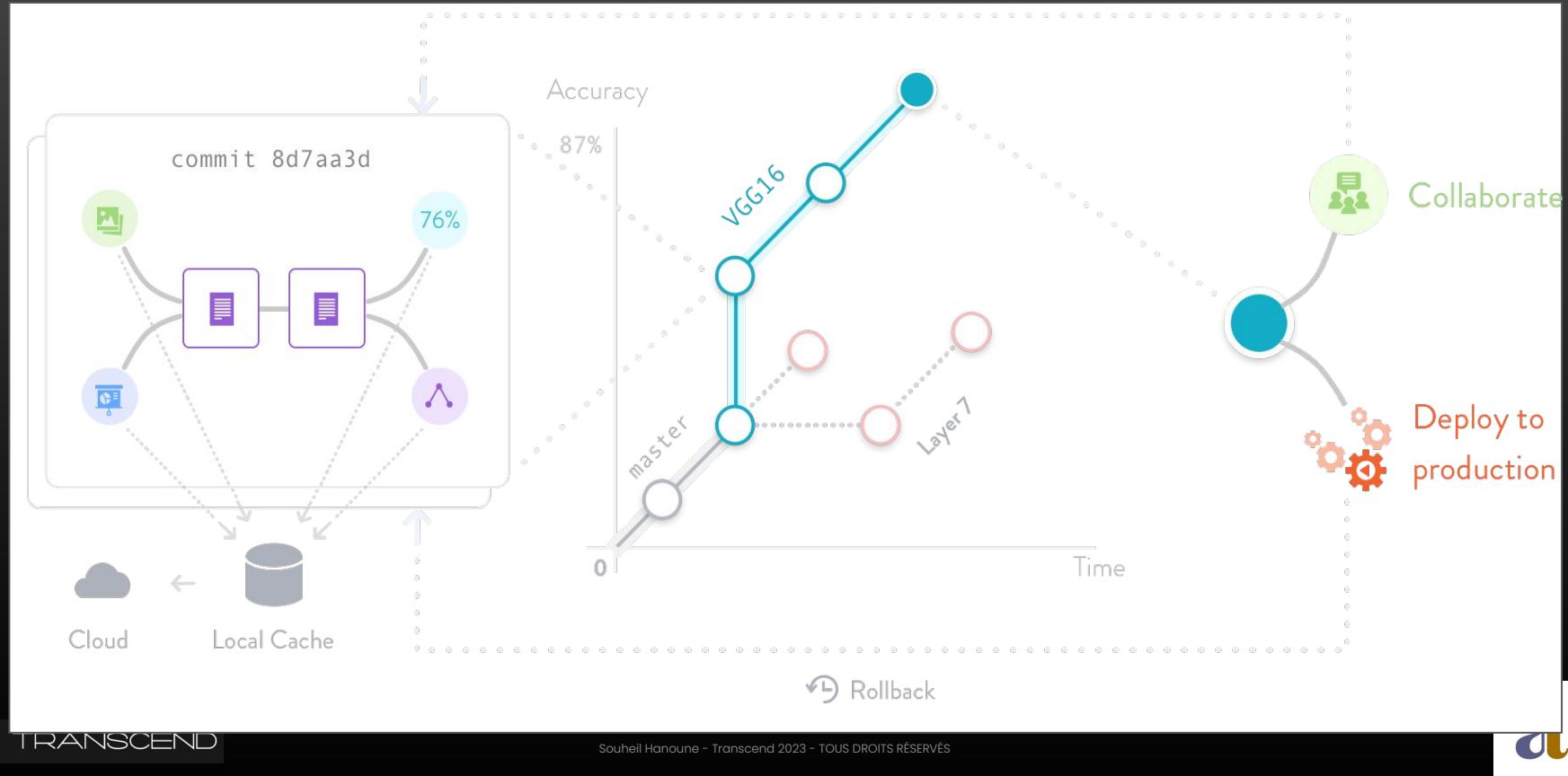
MLflow : A platform of the machine learning lifecycle



MLflow : A platform of the machine learning lifecycle



(Not Just) Data Version Control (DVC)



TRANSCEND



(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.

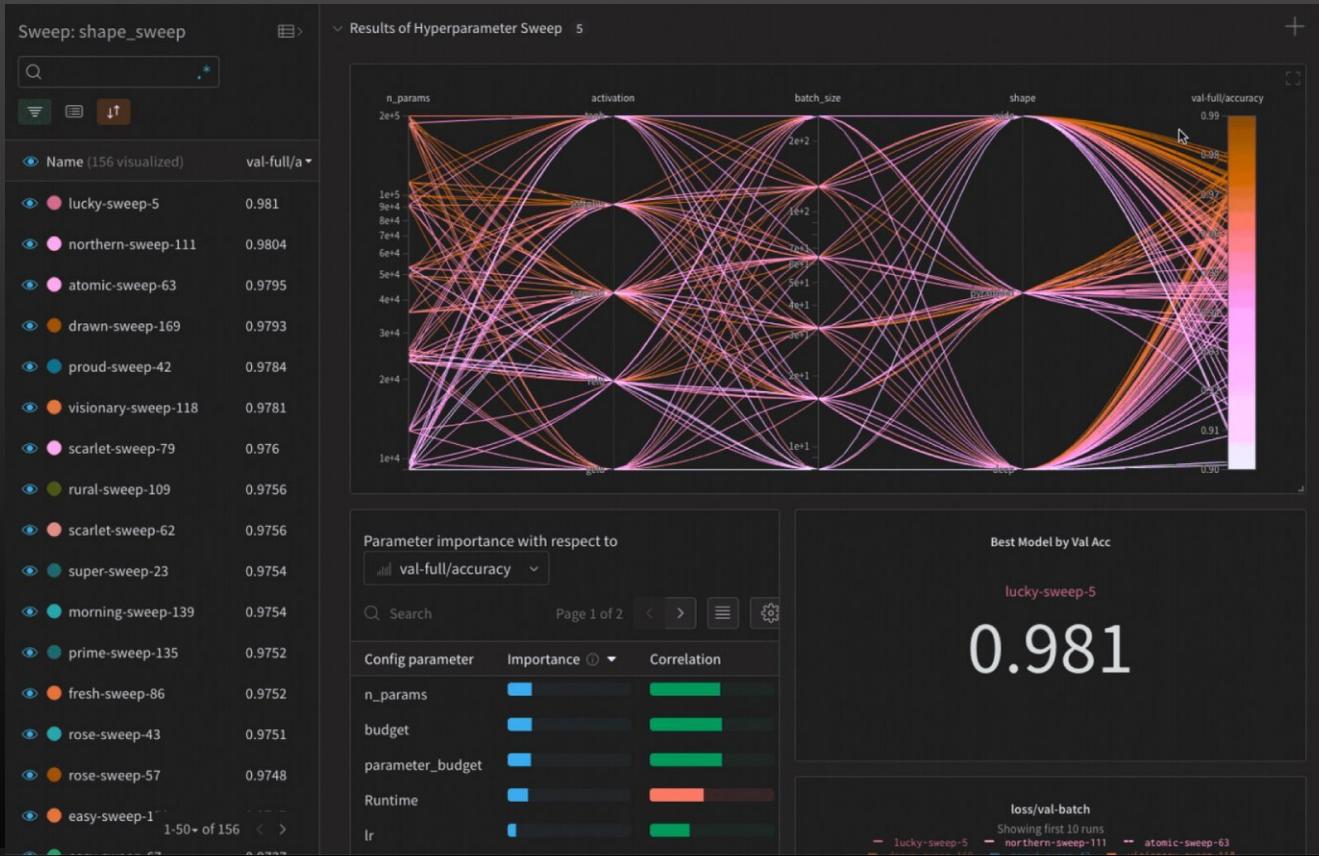
DVC was designed to keep branching as simple and fast as in Git — no matter the data file size. Along with first-class citizen metrics and ML pipelines, it means that a project has cleaner structure. It's easy to compare ideas and pick the best. Iterations become faster with intermediate artifact caching.

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



DU DEVELOPPEMENT AU DEPLOIEMENT

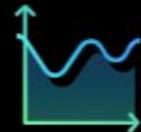
Un chaîne MLOps Complète



Pipelines automatisés



Gestion des données



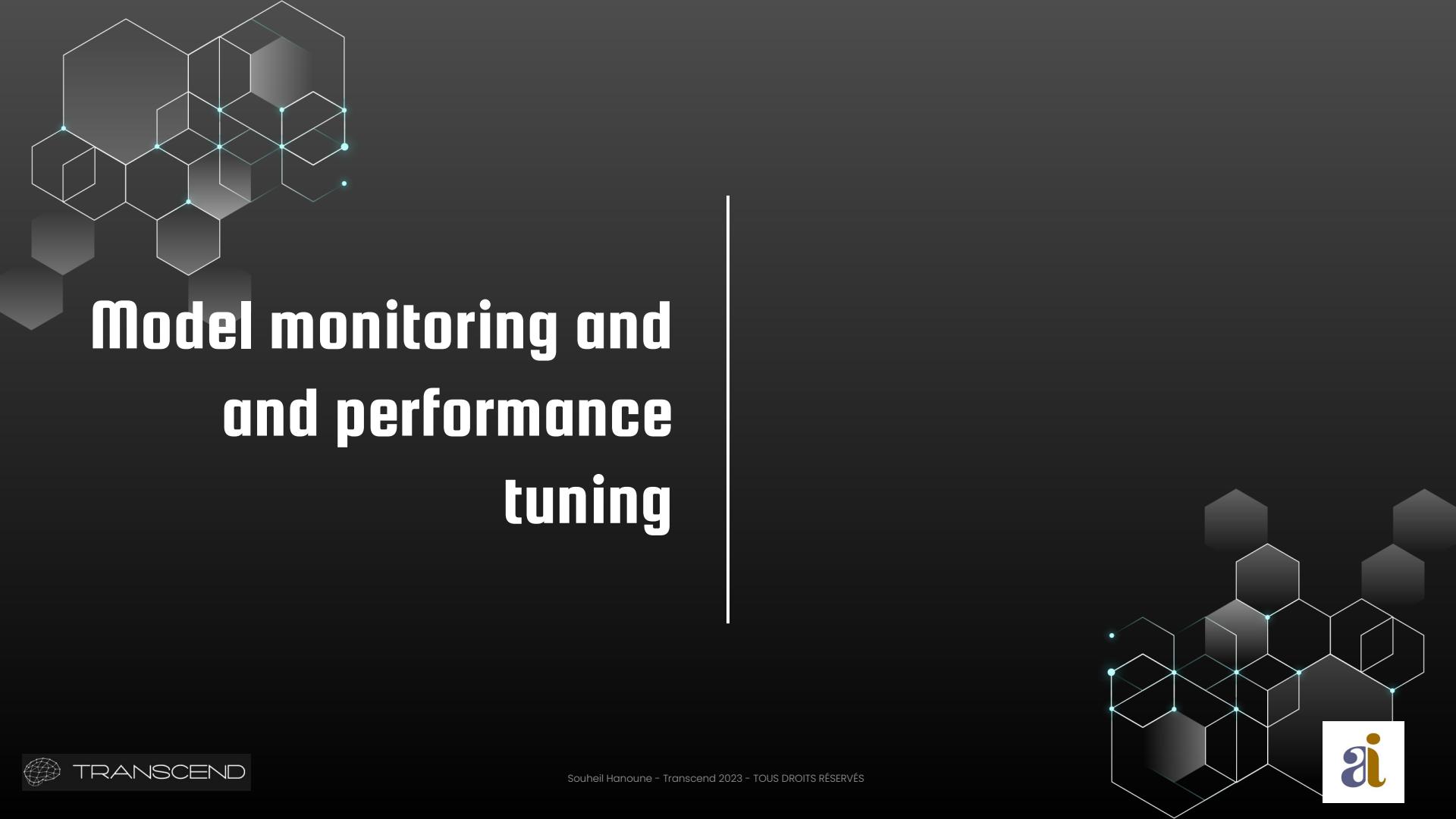
Expérience
Suivi



Modèle
Déploiement



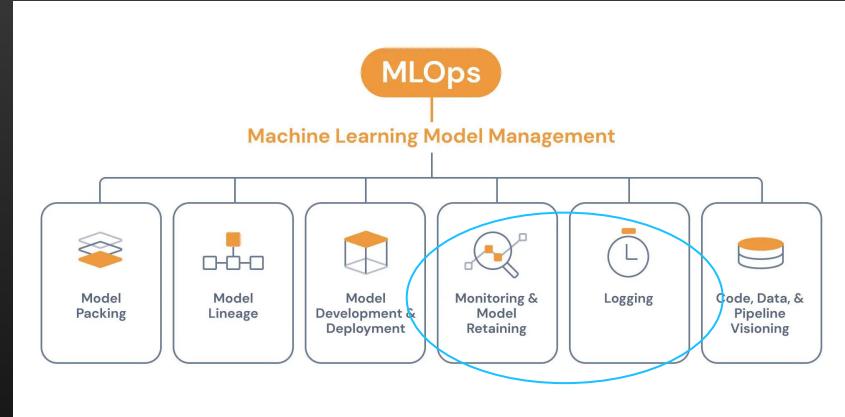
Modèle
Surveillance



Model monitoring and and performance tuning

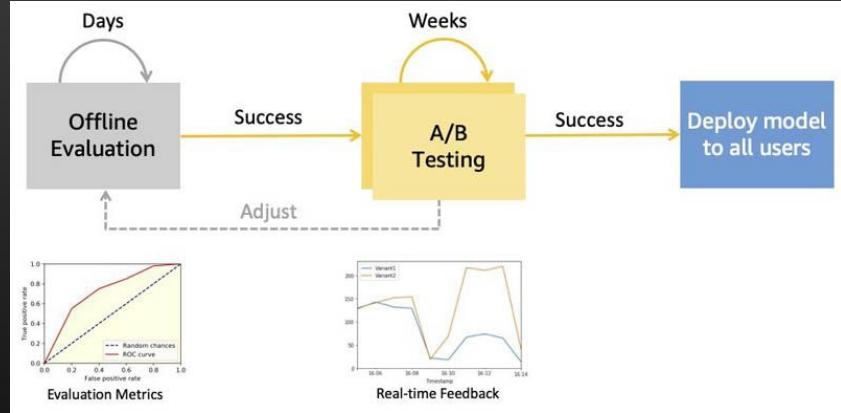
Monitoring deployed models

- **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



Transfer learning

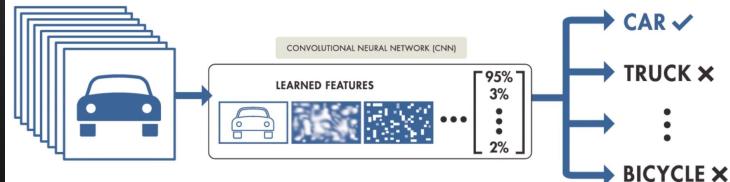
- Needs data
- Takes time
- Needs computing power

Traditional ML vs Transfer Learning

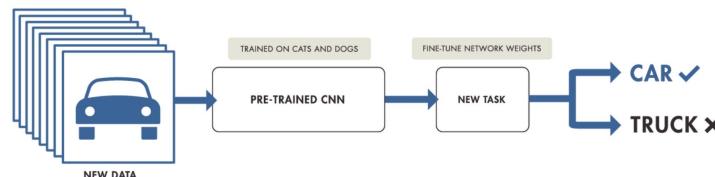
- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data



TRAINING FROM SCRATCH

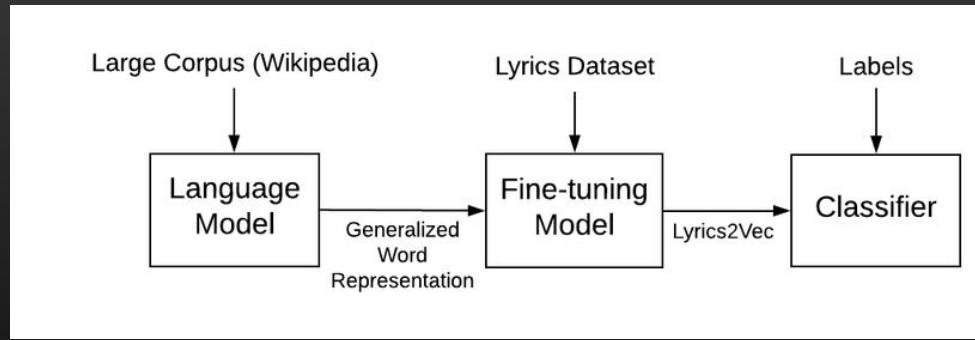


TRANSFER LEARNING



Fine tuning

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



Transfer learning vs Fine tuning

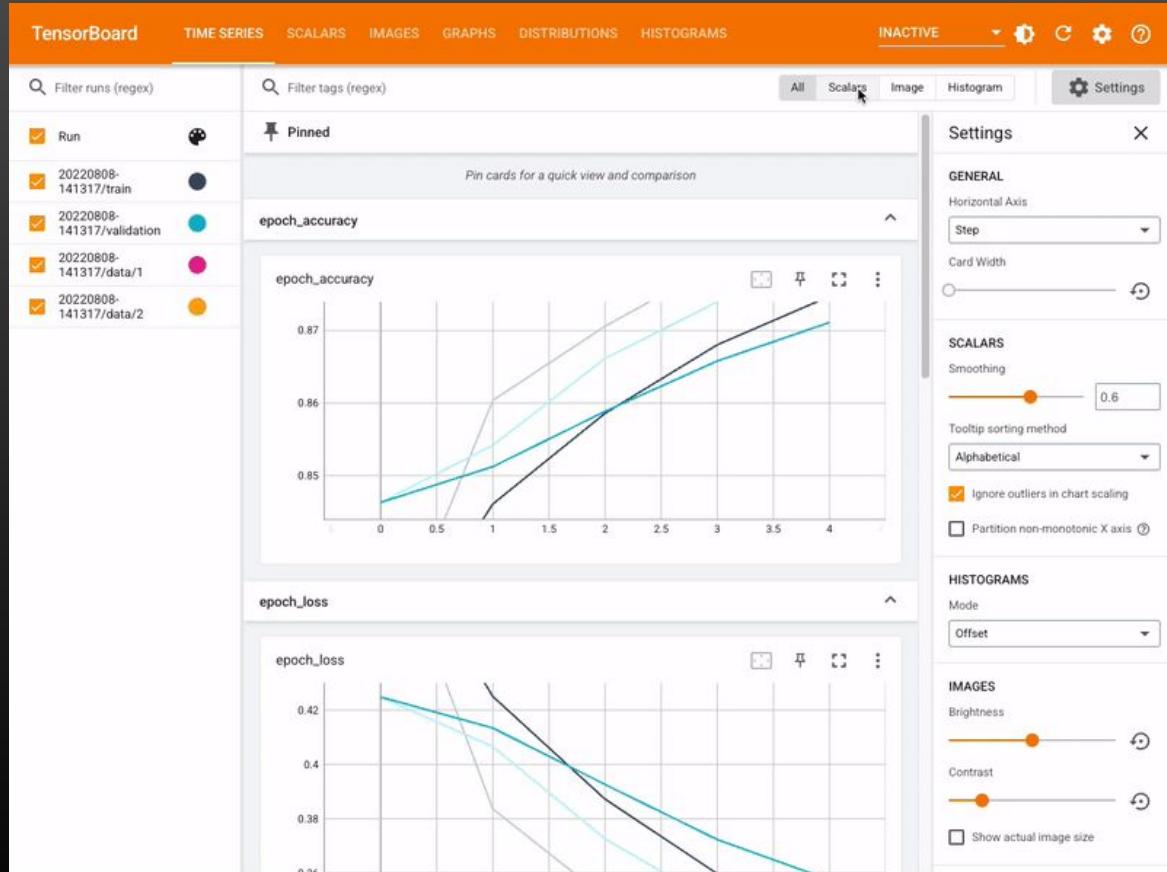
	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

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.

Great visualization

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

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.

Many client libraries

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

Powerful queries

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

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.

Precise alerting

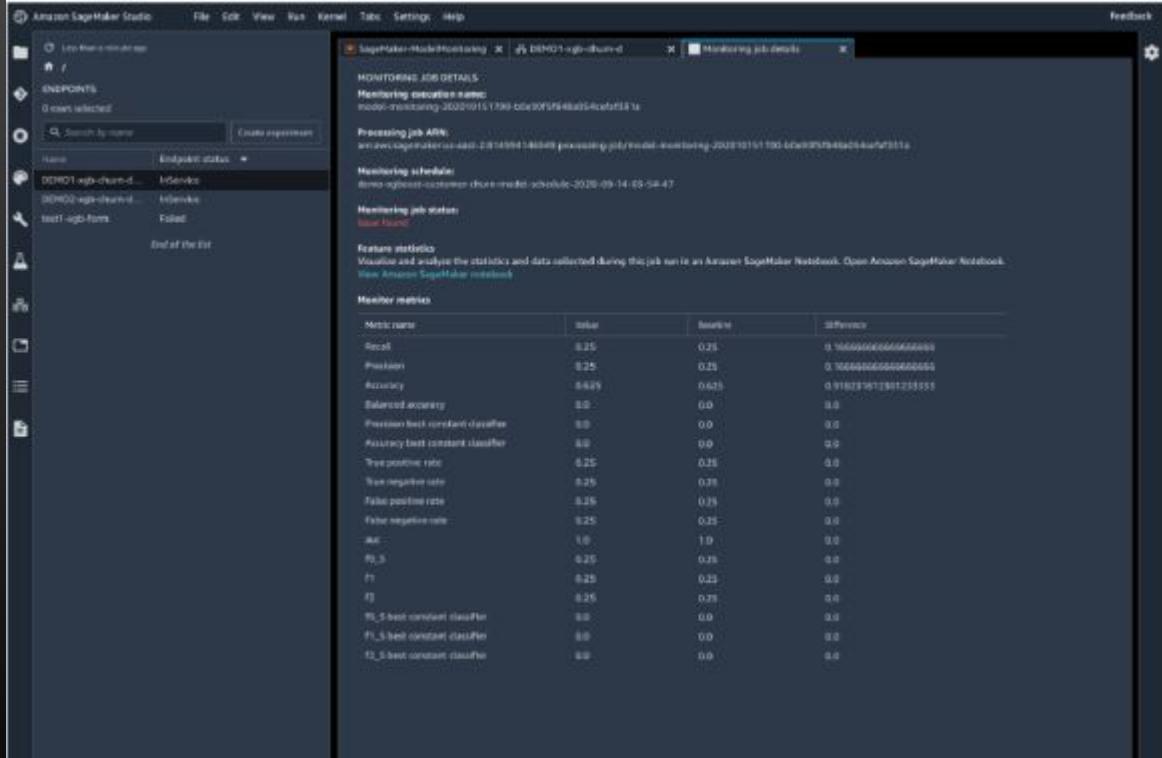
Alerts are defined based on Prometheus's flexible PromQL and maintain dimensional information. An alertmanager handles notifications and silencing.

Many integrations

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

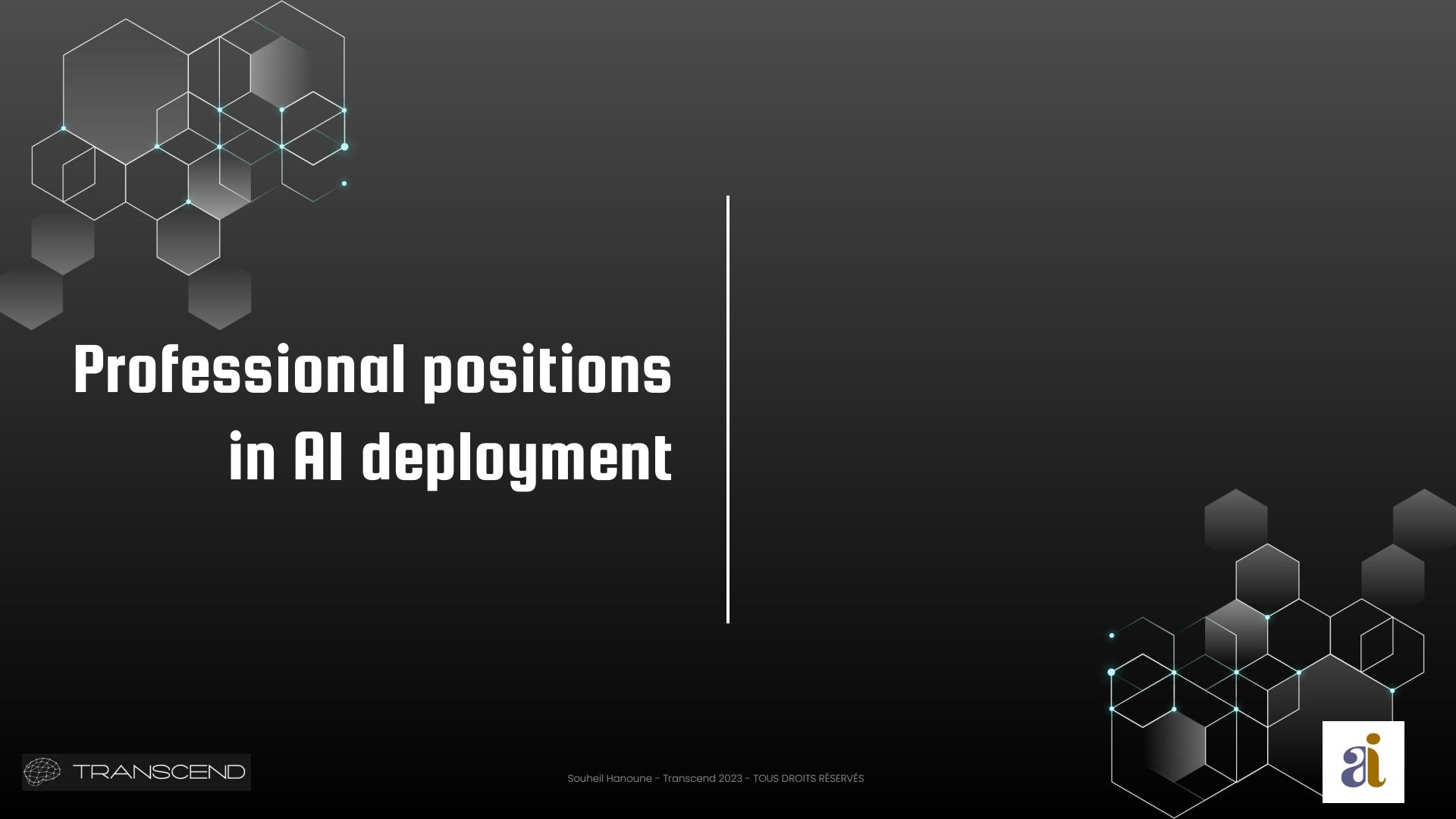
Amazon SageMaker Model Monitor

- **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	Census
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



Professional positions in AI deployment

Data scientist / ML Engineer / Data Engineer

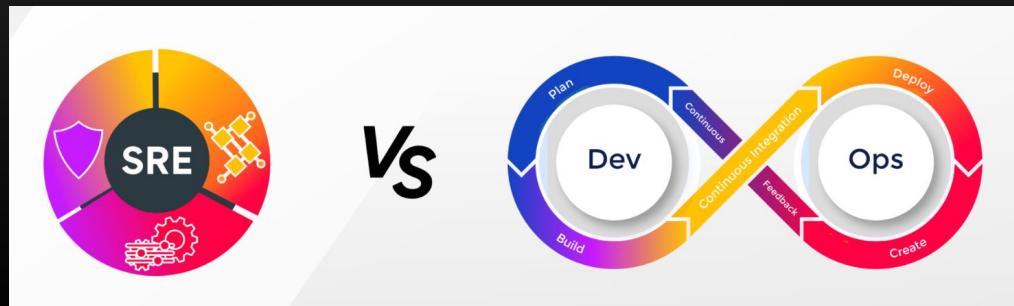
Data Scientist vs. Machine Learning Engineer vs. Data Engineer

	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	<ul style="list-style-type: none">✓ Knowledge of math and statistics✓ Decision making and data optimization skills✓ High proficiency in SQL✓ Scripting skills (R/Python)	<ul style="list-style-type: none">✓ Solid programming background✓ Data science skills✓ Knowledge of math and statistics✓ Rapid prototyping skills✓ Good problem-solving-skills✓ Proficiency in deep learning frameworks	<ul style="list-style-type: none">✓ 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



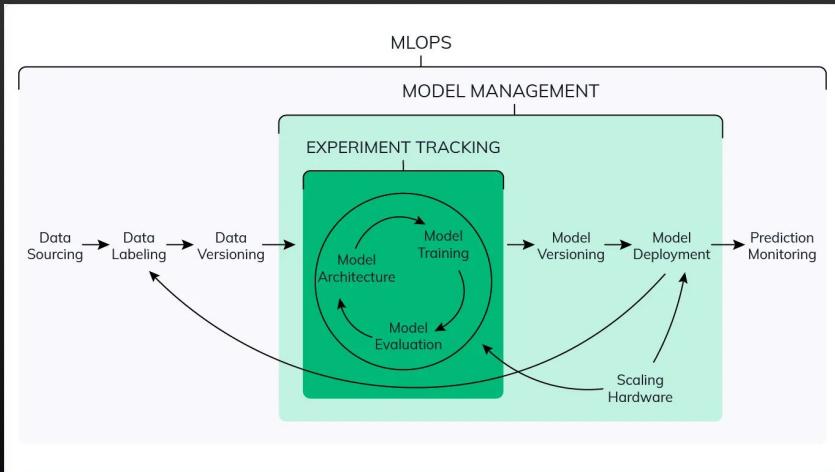
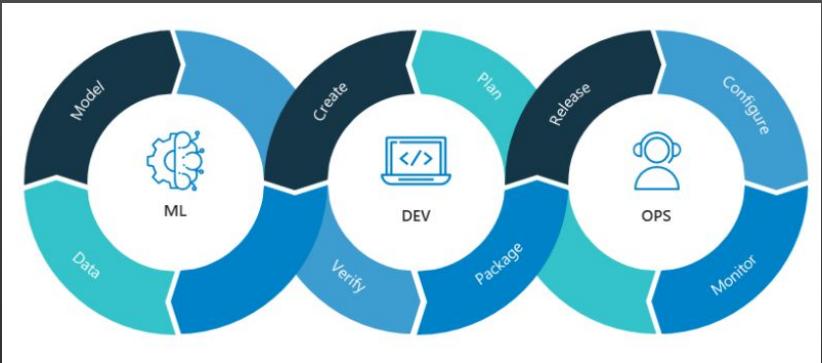
DevOps & SRE (Site reliability Engineer)

- 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).



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

Product Owner vs. AI Product Owner

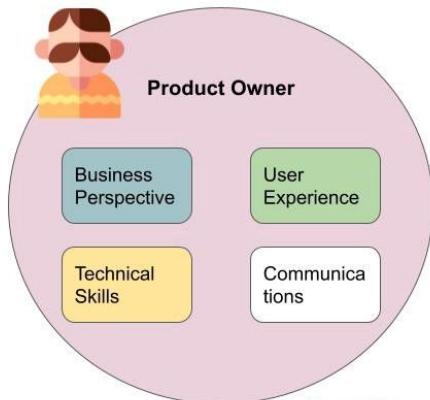
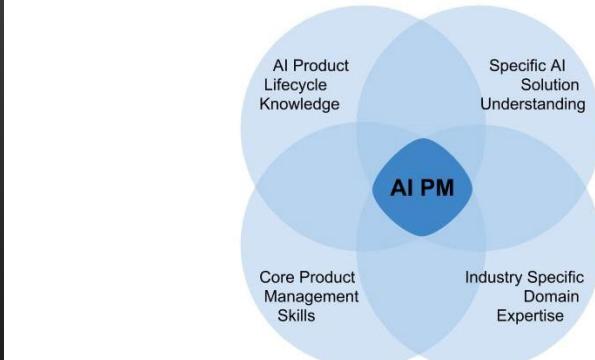


Image source: Icons made by [Freepik](#) from [www.flaticon.com](#)

The AI Product Manager



© 2018 Adnan Boz. All rights reserved. adnan.boz@gmail.com



Case studies and discussion

Example deploying svm on Azure

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