**Machine Learning**:

* The **gradient** is the generalization of the derivative to multivariate functions. It captures the local slope of the function, allowing us to predict the effect of taking a small step from a point in any direction.

**Facial Recognition**:

* CNN(Convolutional Nueral Network) a substream of DNN (Deep Nueral Network) - most commonly applied to analyzing visual imagery.
* SVM(Support Vector Machine) - A supervised machine learning algorithm used for both classification and regression
* DLIB
  + State-of-the-art **face recognition** built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark
  + A toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings
* Feature Extraction methods
  + **SIFT** - is a feature extraction method that **reduces the image content to a set of points used to detect similar patterns in other images**
  + **HOG** - **Histogram of oriented gradients** - The technique counts occurrences of gradient orientation in localized portions of an image
* MTCNN(Multi-Task Cascaded Convolutional Neural Networks)
  + Stage 1 : Face Classification
  + Stage 2: Bounding box regression
  + Stage 3 : Facial Landmark Localization
* K-Nearest Neighbors(KNN) Algorithms for Facial Recognition
  + k-NN is one of the most basic classification algorithms in machine learning. It belongs to the supervised learning category of machine learning. k-NN is often used in search applications where you are looking for “similar” items. The way we measure similarity is by creating a vector representation of the items, and then compare the vectors using an appropriate distance metric (like the Euclidean distance, for example).

It is generally used in data mining, pattern recognition, recommender systems and intrusion detection.

* + <https://www.linkedin.com/pulse/assessing-k-nearest-neighbors-knn-algorithm-accuracy-facial-fericy/>
  + <https://thesai.org/Downloads/Volume8No11/Paper_15-Efficient_K_Nearest_Neighbor_Searches.pdf>
  + <https://www.geeksforgeeks.org/ml-implement-face-recognition-using-k-nn-with-scikit-learn/>
* A Novel **Distributed Approximate Nearest Neighbour** Method for Real-time Face Recognition a.k.a Novel distributed approximate nearest neighbor (**ANN**)
  + Efficient for large datasets.
  + #Maximum Likelihood , #Clustering
  + This algorithm falls in Lazy Algorithm category where there is no training involved.
  + <https://arxiv.org/pdf/2005.05824v1.pdf>
  + <https://paperswithcode.com/paper/a-novel-distributed-approximate-nearest>
  + If the number of classes is large the total number of reference images will be large too. So considering all the reference images to choose the nearest one will be time-consuming. In such cases CNN or SVM algorithms are slow and not effective.
  + These type of algorithms are categorized into
    - “Fast” -
    - “Brute force” - Distance between the input image and all the reference images and choose the nearest one to the input image.

ML Model Formats and Frameworks:

* **ONNX(Open Neural Network Exchange)** is an open format built to represent machine learning models
* Tensorflow
* XGBoost
* ScikitLearn
* PyTorch

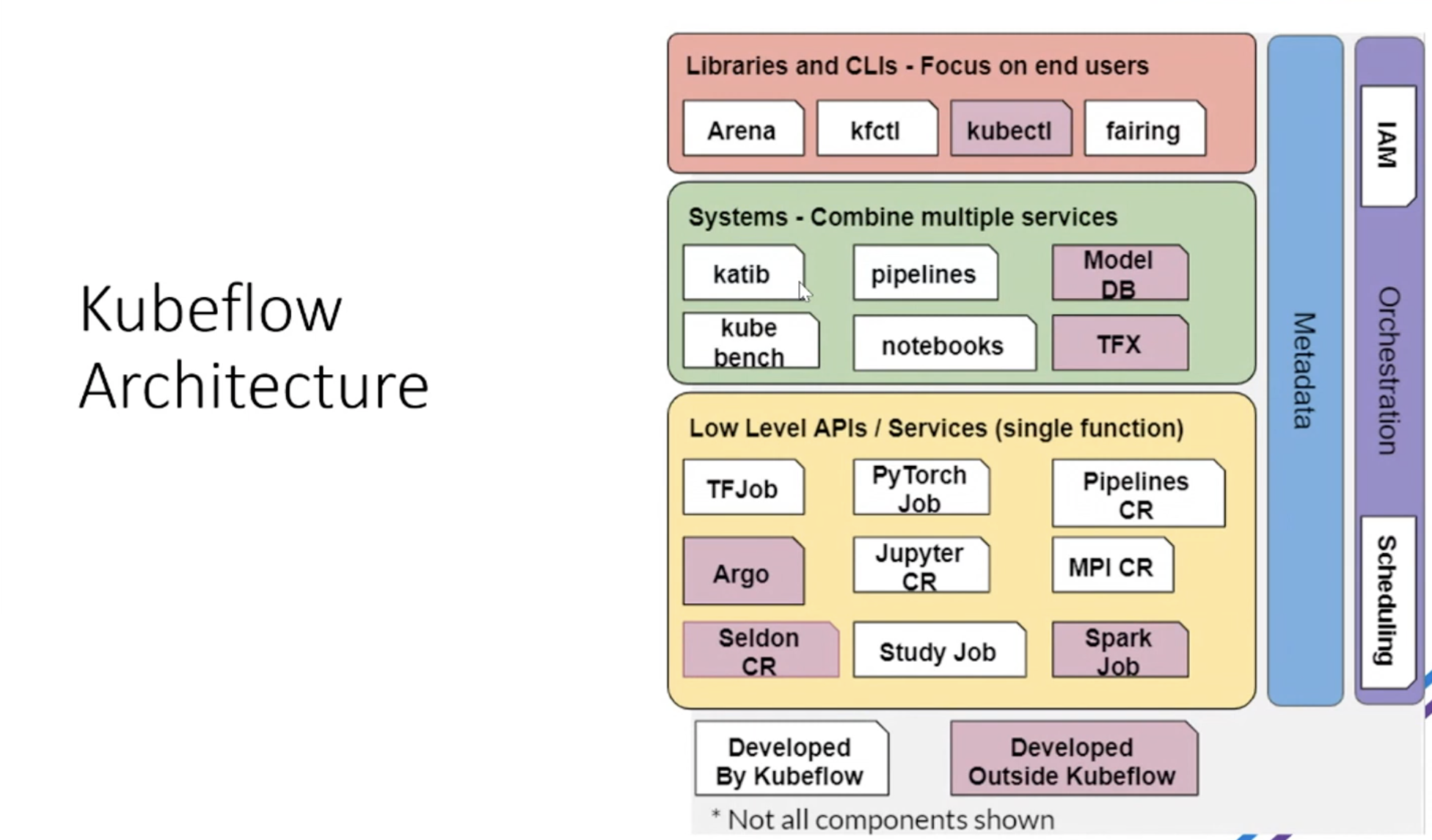
Tracks/Opportunities available with[Vivek Raghavan](mailto:vivek@ekstep.org):

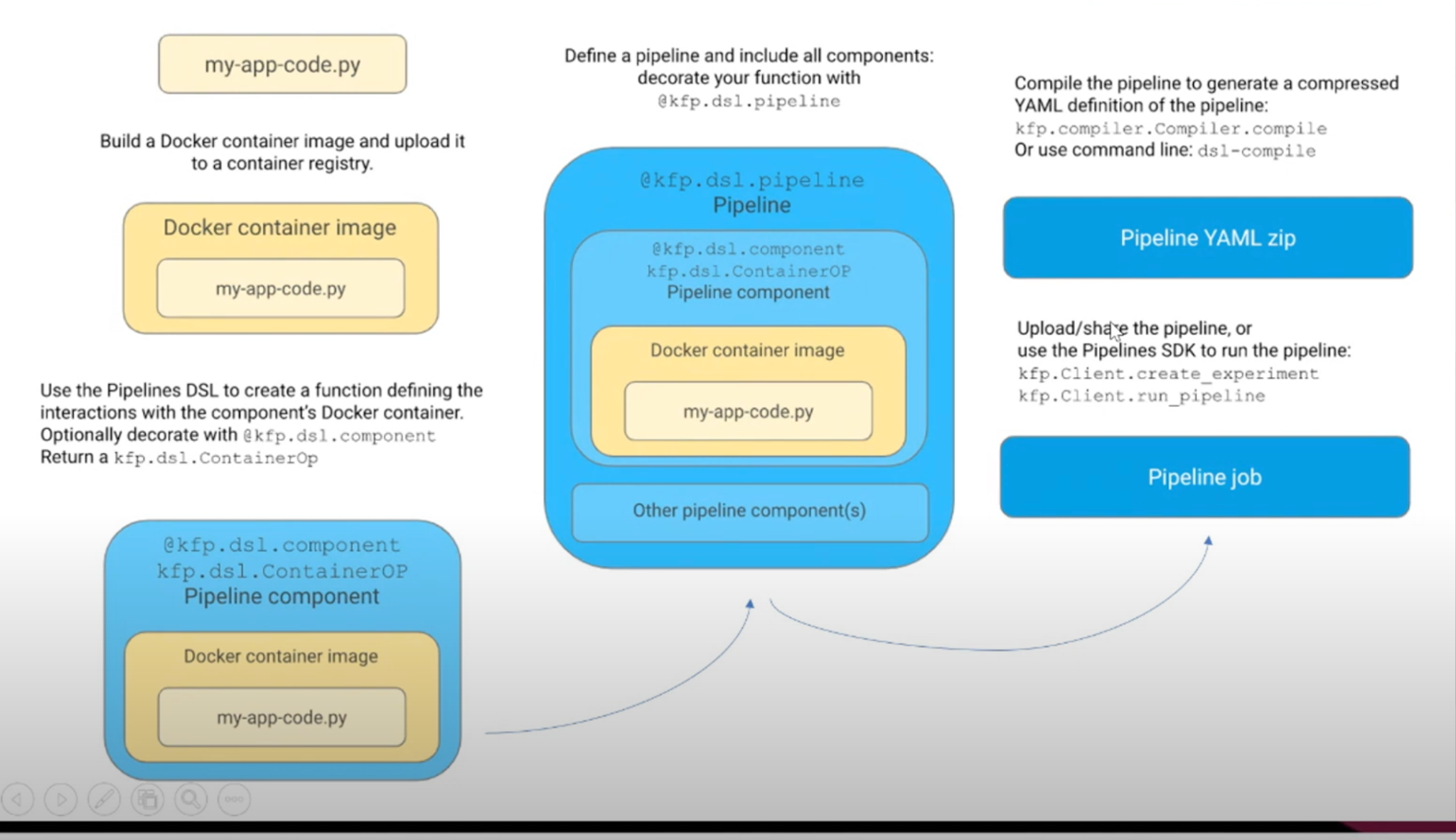
**Novel Distributed Approximate Nearest Neighbour Method for Real-time Face Recognition a.k.a ANN**

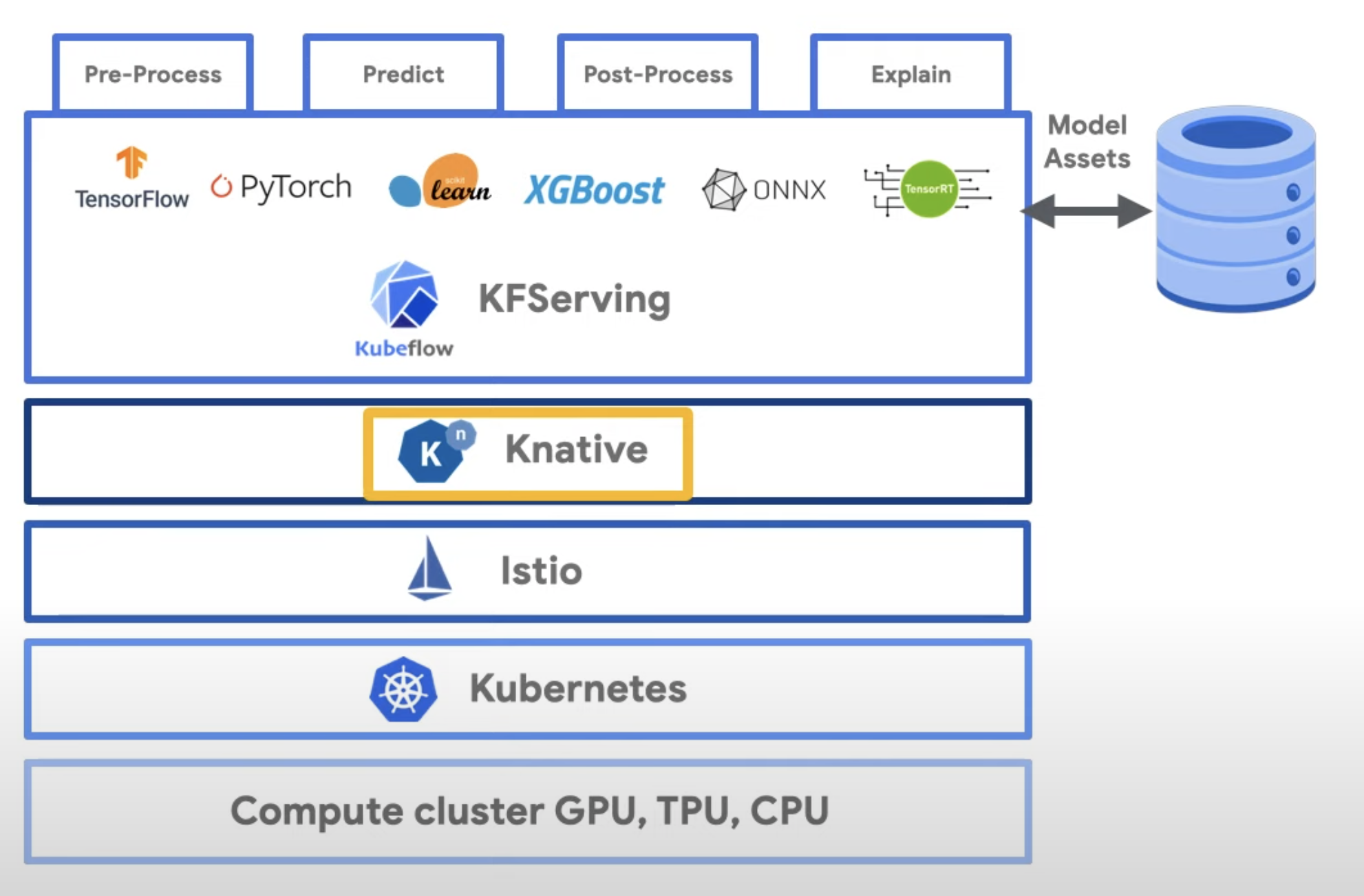
**ML Serverless and resource optimizations**:

* Efficiency and cost effectiveness
* Clear ML <https://clear.ml/>
* **Kubeflow** <https://www.kubeflow.org/>
  + Kubeflow is *the ML toolkit for Kubernetes*.
  + Goal is not to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems for ML to diverse infrastructures
  + <https://www.youtube.com/playlist?list=PLIivdWyY5sqLS4lN75RPDEyBgTro_YX7x>
  + **Katib** : For tuning ML hyper parameters
  + **Pipelines**: help in building **step** by step ML flows. Steps like collecting data , serving model which are build as reusable components. Use kubeflow DSL to define pipelines using annotations.
  + **Jupyter Notebooks**
  + Matadata Management: Information about runs,model info , datasets,descriptions ,type of model.
  + **KFServing(Kubeflow Serving)** : To serve ML models in production. Supports GPUs,Auto-scaling , Cannary rollouts. SeldonCore can be used instead of KFServing ?
  + **Kustomize** is a configuration management solution that leverages layering to preserve the base settings of your applications and components by overlaying declarative yaml artifacts (called patches) that selectively override default settings without actually changing the original files.
  + **TF-Job** : A distributed TensorFlow job typically contains 0 or more of below processes. Its custom resource defined by Kubeflow.
    - Chief , Parameter Server, Worker,Evaluator
  + Kale - External Addon provides UI in the form of a JupyterLab extension

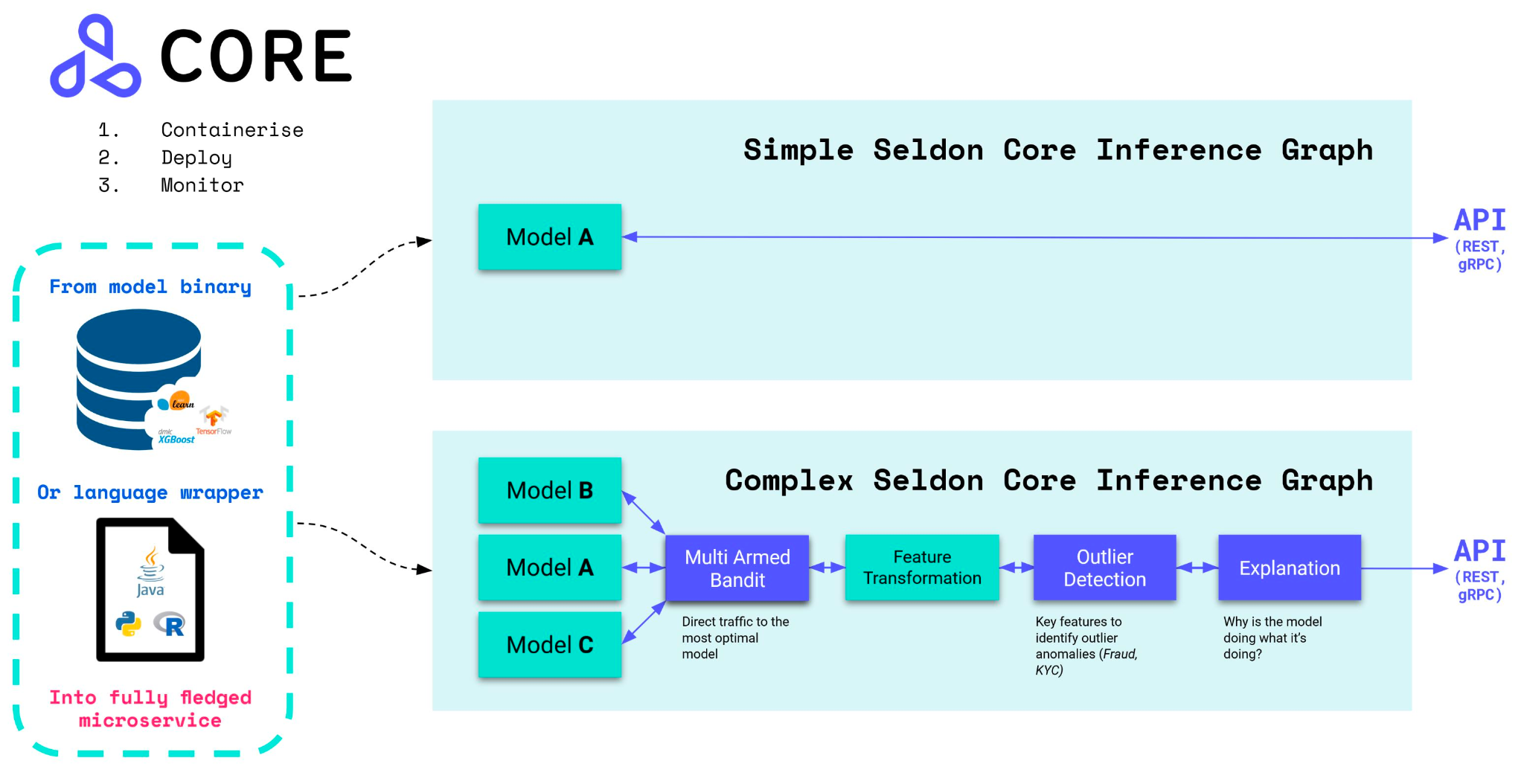
<https://www.kubeflow.org/docs/started/architecture/>

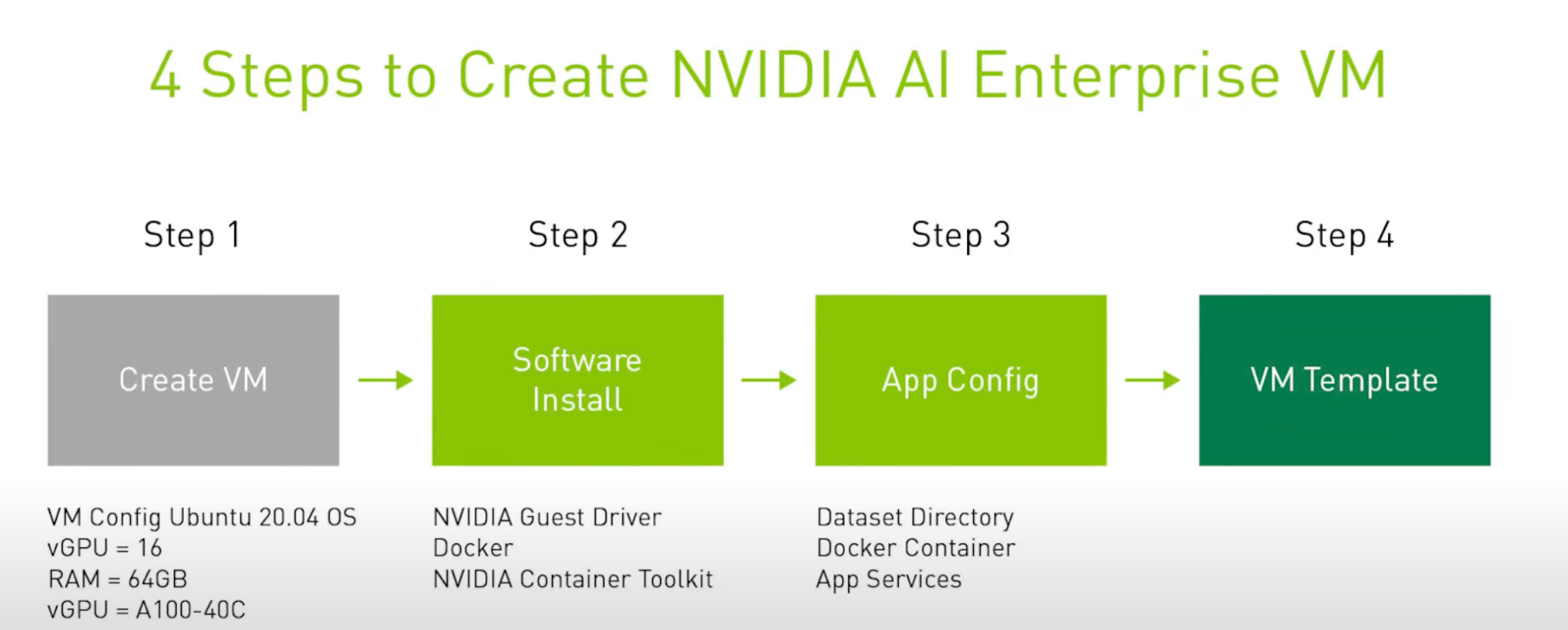






* **KServe** <https://kserve.github.io/website/0.9/>
  + <https://github.com/kserve/kserve/>
  + Highly scalable and standards based Model Inference Platform on Kubernetes for Trusted AI
* **SeldonCore** : <https://github.com/SeldonIO/seldon-core>
  + An open source platform to deploy your machine learning models on Kubernetes at massive scale.



* Triton <https://github.com/triton-inference-server/server/>
  + Triton Inference Server provides a cloud and edge inferencing solution optimized for both CPUs and GPUs
  + NVIDIA LaunchPad labs
  + Configurable Model specific custom operations
  + Scheduling and Batching
  + Model Repository
  + [Model Analyzer tool](https://github.com/triton-inference-server/model_analyzer) for profiling and optimizing your models
  + <https://github.com/triton-inference-server/server>
    - Triton Inference Server is an **open source inference serving software that streamlines AI inferencing**. Triton enables teams to deploy any AI model from multiple deep learning and machine learning frameworks, including TensorRT, TensorFlow, PyTorch, ONNX, OpenVINO, Python, RAPIDS FIL, and more. Triton supports inference across cloud, data center,edge and embedded devices on NVIDIA GPUs, x86 and ARM CPU, or AWS Inferentia. Triton delivers optimized performance for many query types, including real time, batched, ensembles and audio/video streaming.
  + NVIDIA GPU Cloud (NGC)

<https://www.youtube.com/watch?v=guYIxRnTXfs>

* Model Optimization
* Allows on-demand serverless



**Amazon SageMaker** is a cloud machine-learning platform that was launched in November 2017. SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems and edge-devices.

**Azure Machine Learning Service**

**ML Model formats:**

#### **TensorFlow** - It has a collection of pre-trained models and is one of the most popular machine learning frameworks that help engineers, deep neural scientists to create [deep learning algorithms](https://www.projectpro.io/article/deep-learning-algorithms/443) and models. From Google.

**Pros of Using Tensorflow**

* It can render subtle computational graph visualizations.
* Since Google backs it, it is efficient, provides seamless performance, and caters to quick updates.
* It also allows executing subparts of a graph, which enable machines to extract discrete data.

**Cons of Using Tensorflow**

* If you compare TensorFlow with its competitors, it lacks in both speed and usage.
* It is not supportable in Windows systems.

#### **Keras** - Open Source framework developed on top of TensorFlow.

**Pros of Using Keras**

* It has a collection of pre-trained models.
* Keras is a high-level API that supports multi-platform and multi-backend integrations.
* Developers use it for rapid prototyping.

**Cons of Using Keras**

* The errors produced by the Keras framework are not easily identifiable.
* Keras fails to handle low-level computation.

#### **MXNet** - MXNet is a choice of all Deep Learning developers. It supports scalability for a wide range of GPUs and programming languages

**Pros of Using MXNet**

* It is scalable, efficient, and fast.
* It supports programming languages like R, Scala, Python, JavaScript, C++, etc.

**Cons of Using MXNet**

* MXNet has less open-source community support as compared to TensorFlow.
* Bug fixes and feature updates take a longer time due to a shortage of community support.

#### **Caffe - Convolutional Architecture for Fast Feature Embedding**

**Pros of Using Caffe**

* It is well-accepted for computationally constrained platforms.
* It is open-source and developer-friendly.

**Cons of Using Caffe**

* It has a steep learning curve.
* It is not a good choice while dealing with sequence modeling and recurrent neural networks.

#### **H2O -** It is another open-source, business-oriented machine learning Framework. It helps in implementing predictive analytics with mathematics to make decisions based on granular data

**Pros of Using H2O**

* H2O is customizable.
* Performing auto ML through H2O is effective.
* It is super-easy to use for programmers coming from diverse programming backgrounds.

**Cons of Using H2O**

* Scalability is too less in the H2O framework.

#### **Scikit Learn -** Python-based ML framework written with programming languages like C, C++, Python, and Cython

**Pros of Using SciKit Learn**

* It allows the clustering of unlabeled data.
* It has an [ensemble feature](https://www.projectpro.io/article/a-comprehensive-guide-to-ensemble-learning-methods/432) that helps in combining predictions of several supervised models.

**Cons of Using SciKit Learn**

* It is not an excellent alternative for in-depth learning.
* It is not a good choice for web apps.

#### **PyTorch** - It is a popular, lightweight, open-source ML and DL framework developed by Facebook on top of the Torch library, and hence the name

**Pros of Using PyTorch**

* It's beginner's friendly.
* It has multi-GPU support.
* It does smooth integration with Python for data science operations also.

**Cons of Using PyTorch**

* It lacks data visualization and monitoring tools.

#### Amazon Machine Learning

* Azure ML
  + Managed Endpoints with Triton Inference Service

<https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-with-triton?tabs=azure-cli%2Cendpoint>

**Platform : Dhruva**

NVIDIA - Internal Reva platform

Compare Azure ML vs Seldon with K8S deployments to be presented End of this month. <https://learn.microsoft.com/en-us/azure/machine-learning/concept-endpoints#managed-online-endpoints-vs-kubernetes-online-endpoints>

<https://neptune.ai/blog/best-mlops-platforms-to-manage-machine-learning-lifecycle>

<https://neptune.ai/blog/azure-ml-alternatives-for-mlops>

[**Triton Inference Server scaling with K8S**](https://developer.nvidia.com/blog/deploying-nvidia-triton-at-scale-with-mig-and-kubernetes/#:~:text=Query%20NVIDIA%20T%5B%E2%80%A6%5Detrics%20using%20Prometheus)

Multi-Instance GPU (MIG)

feedback from AML team on how to use Triton on Azure -

1. Deploying through NCD (no code deployment) option

[azureml-examples/deploy-triton-managed-online-endpoint.sh at main · Azure/azureml-examples (](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FAzure%2Fazureml-examples%2Fblob%2Fmain%2Fcli%2Fdeploy-triton-managed-online-endpoint.sh&data=05%7C01%7Cpratykumar%40microsoft.com%7C0a60bfce58a347b864be08dadc6a3f90%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C638064646303162083%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=KwbaAiblbEhdPZBE82ONwYub089BWRNlZGtbMMthANI%3D&reserved=0)[github.com](http://github.com/)[)](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FAzure%2Fazureml-examples%2Fblob%2Fmain%2Fcli%2Fdeploy-triton-managed-online-endpoint.sh&data=05%7C01%7Cpratykumar%40microsoft.com%7C0a60bfce58a347b864be08dadc6a3f90%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C638064646303162083%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=KwbaAiblbEhdPZBE82ONwYub089BWRNlZGtbMMthANI%3D&reserved=0)

2. BYOC (custom container) option

[azureml-examples/deploy-custom-container-triton-single-model.sh at main · Azure/azureml-examples (](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FAzure%2Fazureml-examples%2Fblob%2Fmain%2Fcli%2Fdeploy-custom-container-triton-single-model.sh&data=05%7C01%7Cpratykumar%40microsoft.com%7C0a60bfce58a347b864be08dadc6a3f90%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C638064646303318731%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=QVPm2bLCEzGSfCMaEE4nSAJ8U6bJqOu%2F21n0vzR8xYg%3D&reserved=0)[github.com](http://github.com/)[)](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FAzure%2Fazureml-examples%2Fblob%2Fmain%2Fcli%2Fdeploy-custom-container-triton-single-model.sh&data=05%7C01%7Cpratykumar%40microsoft.com%7C0a60bfce58a347b864be08dadc6a3f90%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C638064646303318731%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=QVPm2bLCEzGSfCMaEE4nSAJ8U6bJqOu%2F21n0vzR8xYg%3D&reserved=0)

Cloud ML Platforms vs Open Platform Technologies

Reference : <https://neptune.ai/blog/azure-ml-alternatives-for-mlops>

| **Features** | **AzureML** | **AWS SageMaker** | **Open Platform**  **Kubeflow/Seldon/Triton/** |
| --- | --- | --- | --- |
| Experiment Tracking |  |  |  |
| Model Management |  |  |  |
| Model Deployment |  |  |  |
| Model Lineage |  |  |  |
| Model Monitoring |  |  |  |
| Data Labelling |  |  |  |
| Cloud Agnostic |  |  |  |