2022 • GIORGIA BERTACCHINI

MLOps

Machine Learning Model Operationalization Management

Part 2

Concepts

MLOps Maturity

AI MATURITY

The smart applications of AI enable organizations to improve, to scale, and to accelerate the decision-making process across most business functions, so as to work both more efficiently and more effectively.

Launching ML-powered software solutions, denotes a company’s maturity in incorporating AI into the business.

Google Cloud's AI Adoption

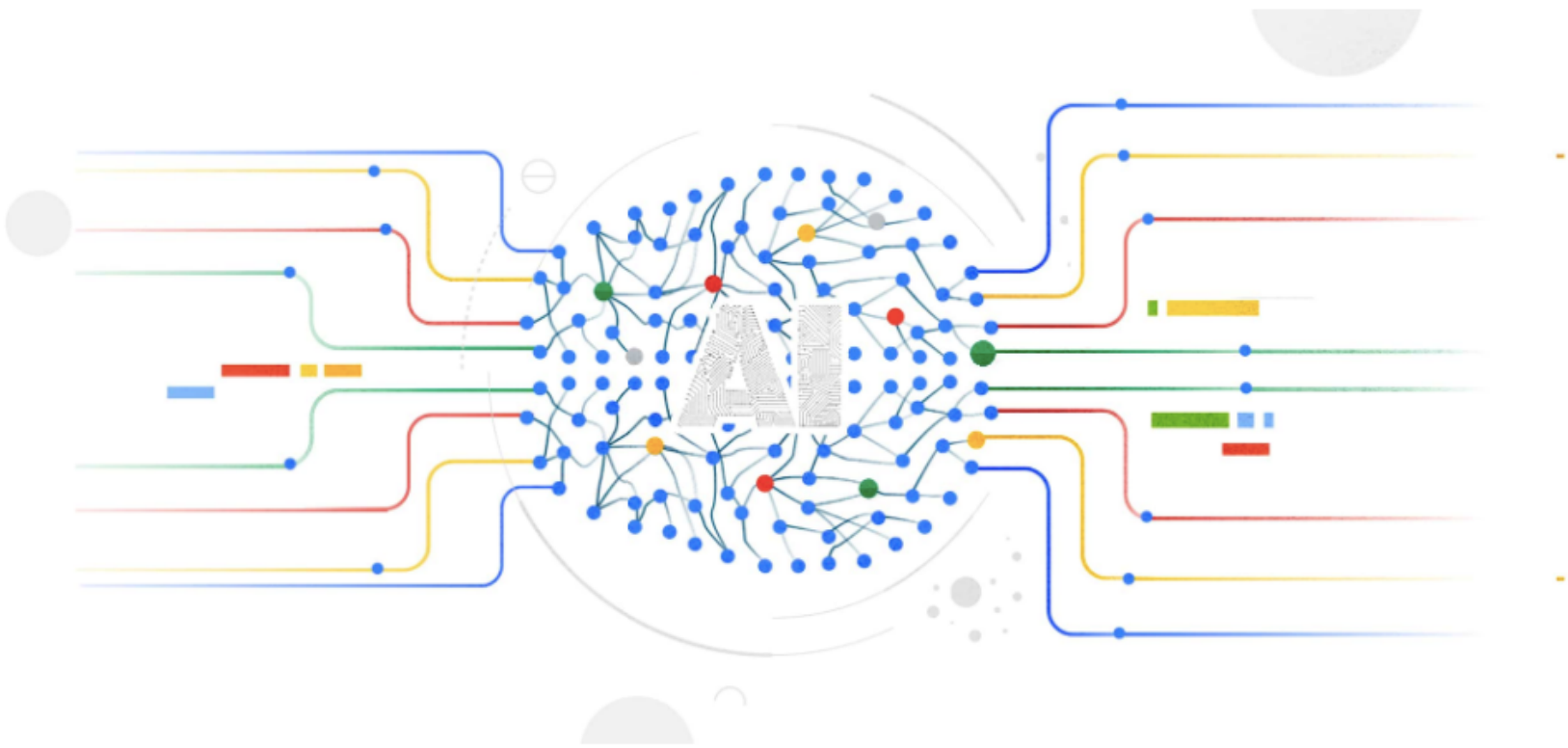
MLOps • 2022

Google Cloud's AI Adoption

FRAMEWORK

With Google Cloud's AI Adoption Framework, you'll be able to create and evolve your own transformative **AI capability**.
You'll have a structure for building **scalable AI capabilities** to create better insights from big data with powerful algorithms across the entire business.

With Google Cloud as **guide**, the path to AI is considerably smoother.



MLOps • 2022

AI maturity Themes

AREAS AND THEMES

Google Cloud's AI Adoption Framework is anchored in the familiar rubric of **people, process, technology, and data**. The interplay between these four key areas gives rise to six themes: **Learn, Lead, Access, Scale, Automate, and Secure**.
These themes are foundational to the AI Adoption Framework.

AI maturity Themes

LEARN

Learn concerns the quality and scale of learning programs to upskill your staff, hire external talent, and augment your data science and ML engineering staff with experienced partners.

LEAD

Lead concerns the extent to which your data scientists are supported by a mandate from leadership to apply ML to business use cases, and the degree to which the data scientists are cross-functional, collaborative, and self-motivated.

ACCESS

Access concerns the extent to which your organization recognizes data management as a key element to enable AI and the degree to which data scientists can share, discover, and reuse data and other ML artifacts.

AI maturity Themes

SCALE

Scale concerns the extent to which you use cloud-native ML services that scale with large amounts of data and large numbers of data processing and ML jobs, with reduced operational overhead.

SECURE

Secure concerns the extent to which you understand and protect your data and ML services from unauthorized and inappropriate access, in addition to ensuring responsible and explainable AI.

AUTOMATE

Automate concerns the extent to which you are able to deploy, execute, and operate technology for data processing and ML pipelines in production efficiently, frequently, and reliably.

AI Maturity Phases

In general, the amount of MLOps practices and infrastructure components will depend on the organization's ML-maturity.

For each theme, current business practices will fall into one of the following phases: **tactical, strategic, transformational**.

AI Maturity Phases: Tactical

GENERAL

The initial phase, known as Tactical, denotes that the organizations **explore the capabilities of ML/AI technologies**.
The reasonable way is to start with **non-critical use cases and short-term projects**.
Furthermore, all processes are **mainly manual** because there is little to no MLOps infrastructure and ML skills are being developed.

At this phase, too, there may be **no process to scale solutions** consistency, nor the skill set to solve complex analytics problems.

USE CASES

Developers are typically leveraging **exploratory data analysis (EDA)** tools and **ready-to-use AI** and ML services for proofs of concept and prototyping, for example, using a prebuilt computer vision service to detect printed and handwritten text, or using descriptive analytics to create a customer segmentation model.

FOCUS

This phase's focus is clearly **on learning and understanding** what value ML might generate.
The focus is on easy adoption, minimal disruption, and quick wins.

AI Maturity Phases: Tactical

OPPORTUNITIES

Data Access

At this phase, organizations can benefit substantially from better **access to data**. Integrated, cleaner, and fresher data leads, in turn, to actionable insights better tailored to your needs, which translates to sharper and more informed decision-making.

Data lake

To drive effective collaboration across an organization and to innovate with data from many sources, a key next move is to start bringing together siloed data from the many lines of business into a central, **unified data lake**, but with **decentralized access**. With a central data lake, it's easier to derive insights from unstructured data and easier also to perform batch integration for reporting. Data is available through **centralized tooling**, but teams do not lose autonomy of access management or data ownership.

Standard

The quality of the analytics solutions improves when the organization focuses on developing core standards:

- A set of unified standards and technical practices that ensures the **security of all data access and protection**
- A set of common principles and procedures that prevents building any ML that may **harm your brand** (such as, for example, inadvertently building socially biased models)

AI Maturity Phases: Strategic

GENERAL

The next phase, called Strategic, implies that business goals coordinate the **amount of ML use cases** and **parts of the processes are automated**.

Typically, there are **basic ML skills** in the team and **basic infrastructure to get ML models into production**.
The main distinction to the previous phase is utilizing **pipelines** for data preparation and model training. Additionally, the ML system provides end points, such as REST API, to expose ML models to the target application. **Basic ML monitoring and alerting functionality** are in place as well.

FOCUS

Organizations at the strategic phase are focused on delivering **sustainable business value**, with several ML systems deployed and maintained **in production** that leverage both ready-to-use and custom models.

AI Maturity Phases: Strategic

ANALYTICS TEAM

Organizations typically require a degree of centralized coordination and so they will often create an advanced **analytics team** with the right skill set to build solutions for various ML use cases across business functions. This team can be part of a broader **Cloud Center of Excellence (COE)**, where engineers and data scientists with domain expertise work side by side, drawing on business subject matter experts, as required by the use case.

FRAMEWORKS

At this phase, teams use existing **frameworks, methods, and techniques** to solve a variety of use cases; and they often deploy **custom ML models** in production.
Teams retrieve their information from a single source: an **enterprise data warehouse (EDW)**, with complete, consistent, correct, and concurrent data. Extract, transform, load (ETL) and extract, load, transform (ELT) routines are **automated and scalable**.

OPPORTUNITIES

At this phase, organizations can benefit substantially from developing an AI capability that is more **tailored to their business model and their distinct business needs**.

Common practices and guidelines are established for building secure, **ethics-complaint AI solutions**.

AI Maturity Phases: Transformational

GENERAL

In the Transformational phase, organizations use ML **to stimulate innovation,** support agility and help to cultivate a culture where experimentation and learning is continuous and encouraged.
In this phase, ML is productionized as a **fully automated process**, which implies a sophisticated data platform, widely adopted common patterns for ML development, and CI/CT/CD practices.
Models are built and deployed from a unified ML platform, making ML accessible to everyone in the organization.

Furthermore, organizations utilize **advanced MLOps components**, such as feature store, ML model and data versioning, and model monitoring and alerting to trigger model re-training.

There is a mechanism in place for scaling and promoting ML capabilities across the organization

AI Maturity Phases: Transformational

TEAMS

Usually, the ML expertise is a part of a **cross-functional team** to maintain the productionized model.

At this phase, organizations typically model a hybrid approach to AI, with **functional or product-specific AI teams** embedded into the **broader product teams** — supported by the advanced analytics team. The **central team** enables a mechanism for scaling and cultivating these capabilities across the organization.

ROLES

The role of the centralized advanced analytics team becomes more confined to establishing **common patterns and best practices and to providing standard tools and libraries** for accelerating ML projects. By contrast, the data science teams embedded in product groups or lines of business are responsible for building their **function-specific ML models**. This division of responsibility drives consistency in building **high-quality**, technical solutions with real business impact.

AI Maturity Phases: Transformational

PLATFORM

All teams are empowered through a platform that enables access to **useful datasets, prepared features, reusable components, and trained models**. That platform is supported by scalable and serverless compute for batch and online data ingestion and processing, distributed ML training and serving, access to databases with specialized storage and querying capabilities, and hardware accelerators.

OPPORTUNITIES

At this phase, organizations can benefit substantially from focusing on best practices, ensuring that AI practices are responsible, that they are based on sound principles, and that AI systems are safe and robust.

The **ML platform** is supported with tools for continuous integration, continuous training, and continuous model serving and monitoring. Building and maintaining such a platform is a shared responsibility between ML and software engineers with skills in infrastructure.

Data management

DATA ARCHITECTURE

The enterprise software want to make data more accessible and available to business users by directly connecting data owners, data producers, and data consumers.
This is materialize with the Data mesh.

DATA MESH

Data mesh is a new approach to thinking about data based on a distributed architecture for data management.

Data Mesh Architecture

MLOps • 2022

Before

CENTRAL DATA TEAM

Many organizations have invested in a central data lake and a data team with the expectation to drive their business based on data. However, they notice that the central data team often becomes a bottleneck. The team cannot handle all the analytical questions of management.

Tasks
They need to spend too much time fixing broken data pipelines after operational database changes.
The data team has to discover and understand the necessary domain data. For every question, they need to learn domain knowledge to give meaningful insights.

AUTONOMOUS DOMAIN TEAMS

Organizations have also invested in **domain-driven design**, **autonomous domain teams** (also known as **stream-aligned teams** or **product teams**) and a **decentralized microservice architecture**.
These domain teams own and know their domain, including the information needs of the business. They design, build, and run their web applications and APIs on their own.

Why Data mesh

SCALE UP DATA ANALYTICS

With the eventual growth of the organization, the situation of the domain teams and the central data team becomes worse.

A way out of this is to shift the responsibility for data from the central data team to the domain teams. This is the core idea behind the data mesh concept, born in 2019: **Domain-oriented decentralization for analytical data**.

The data mesh architecture is a decentralized approach that enables domain teams to perform cross-domain data analysis on their own and interconnects data, similar to APIs in a microservice architecture.

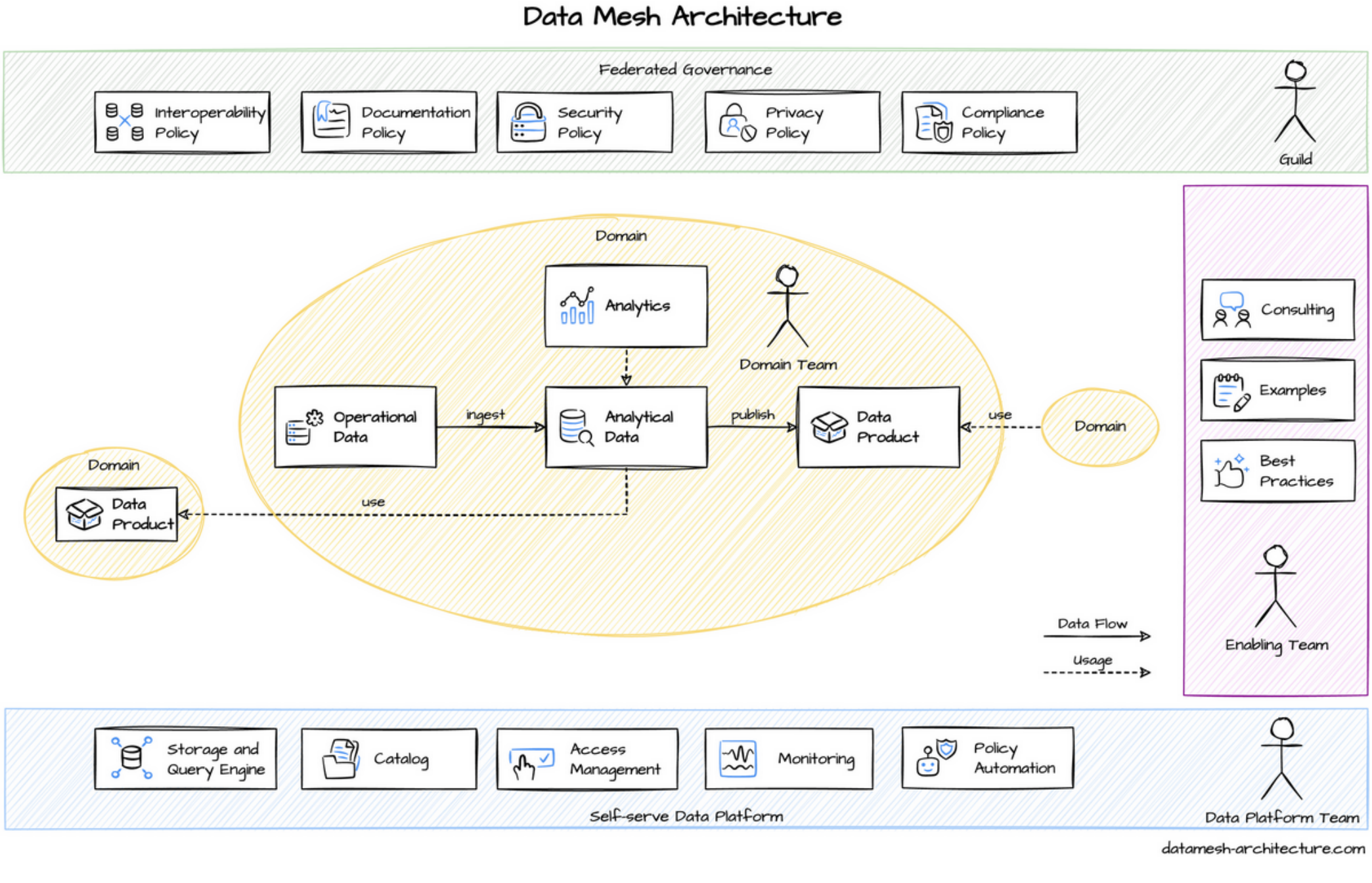
What is Data mesh

PRINCIPLES

Data mesh is based on four fundamental principles:

1. **Domain ownership**
    - The domain teams take responsibility for their data.
    - Following the domain-driven distributed architecture, analytical and operational data ownership is moved to the domain teams, away from the central data team.
2. **Data as a product**
    - This principle means that there are consumers for the data beyond the domain. The domain team is responsible for satisfying the needs of other domains by providing high-quality data.
    - Domain data should be treated as any other public API.
3. **Self-serve data infrastructure platform**
    - A dedicated data platform team provides domain-agnostic functionality, tools, and systems to build, execute, and maintain interoperable data products for all domains.
    - With its platform, the data platform team enables domain teams to seamlessly consume and create data products
4. **Federated governance**
    - The federated governance principle achieves interoperability of all data products through standardization.
    - The main goal of federated governance is to create a data ecosystem with adherence to the organizational rules and industry regulations.

Data Mesh Architecture



MLOps • 2022

Data Mesh Architecture

DOMAIN

At its core is the domain with its **responsible team** and its **operational and analytical data**.

- The domain team ingests **operational data** and builds **analytical data models** to perform their own **analysis**. It uses analytical data to build **data products** based on other domains' needs.

FEDERATED GOVERNANCE

The domain team agrees with others on global policies in a federated governance guild, so that domain teams know how to discover, understand and use data products available in the data mesh.

- **Interoperability Policy**. They allow other domain teams to use data products in a consistent way. For example, define the standard way to provide data.
- **Documentation Policy**. Some form of documentation to discover and understand available data products. A simple policy for this could be a wiki page with a predefined set of metadata (such as owner of the data product, location URL).
- **Security Policy**. A uniform way to access the actual data product in a secure way could be using role-based access.
- **Privacy Policy, Compliance Policy**. Think about protection of personally identifiable information (PII) or industry-specific legal requirements.

Data Mesh Architecture

DATA PLATFORM TEAM

The **self-serve domain-agnostic data platform**, provided by the data platform team, enables domain teams to easily build their own data products and do their own analysis effectively.

ENABLING TEAM

An enabling team guides domain teams on **how to model analytical data, use the data platform, and build and maintain interoperable data products**.
The enabling team consists of specialists with extensive knowledge on data analytics, data engineering, and the self-serve data platform

MLOps • 2022

Data Mesh Architecture

DATA PRODUCT

A data product usually is a published **data set** that can be accessed by other domains, similar to an API. For example,

- the history of inventory updates in a Google BigQuery table
- a daily JSON file with purchase orders on an AWS S3 bucket
- a sales report containing KPIs and charts as a PDF
- a machine learning model to predict shipping dates as an ONNX file.

It is described with **metadata**, including ownership, data location and access, update frequency, and a specification of the data model.

The domain team is responsible for the **operations** of the data product during its entire lifecycle. The team needs to continuously monitor and ensure data quality and availability.

To design data products, we recommend to use the Data Product Canvas.

Data Mesh Architecture

OPERATIONAL DATA

Operational data is often ingested as some kind of raw and unstructured data.

ANALYTICAL DATA

Raw data is cleaned and structured into events and entities.

- **Events** are small, immutable, and highly domain oriented.
- **Entities and aggregates** represent business objects, and their state changes over time so they are mutable.

In practice, we often see **manually** entered or imported data.
Data from other teams are integrated as **external data**, that are well governed so this integration might be implemented in a very lightweight way.

The published data product is derived by aggregating a subset of the events, entities, manual, and external data.

Data Mesh Architecture

INGESTING

Domain teams ingest their operational data into the data platform. In this platform are inserted the events and the entities.
Data can be collected, processed, and forwarded to the data platform in real time. With this streaming ingestion, data are immediately available for analytics when they arrive.

Examples for streaming ingestion: Kafka Connect, Kafka Streams, AWS Lambda.

Data Mesh Architecture

CLEAN DATA

Raw data that is ingested into the data platform is usually imported in its original raw and unstructured format.
Now it can be preprocessed to get data clean:

- Structuring: Transform unstructured and semi-structured data to the analytical data model.
- Mitigation of structural changes: When data structures have changed, mitigate them.
- Deduplication: Remove all duplicate entries.
- Completeness: Ensure that data contain agreed periods.
- Fix outliers: Invalid data that may occur through bugs get identified and corrected.

These preprocessing steps can be implemented as simple **SQL views** that project the raw data. An alternative, the cleaning steps can be implemented as **lambda functions** that operate on topics.
More complex pipelines can be built with frameworks like dbt or Apache Beam that offer an advanced programming model.

Data Mesh Architecture

ANALYTICS

Domain teams query, process, and aggregate their analytical data together with relevant data products from other domains.

SQL is the foundation for most analytical queries. It provides powerful functions to connect and investigate data. The data platform should perform join operations efficiently, even for large data sets.
Aggregations are used to group data.
Examples: Jupyter Notebooks, Presto.

There are a number of great data visualization tools that build beautiful charts, key performance indicator overviews, dashboards and reports. They provide an easy-to-use UI to drill down, filter, and aggregate data.
Examples: Looker, Tableau, Metabase, Redash.

Data science and machine learning methods enable correlation analyzes, prediction models, and other advanced use cases.
Examples: scikit-learn, PyTorch, TensorFlow.

Data Mesh Architecture

DATA PLATFORM

The self-serve data platform may vary for each organization. Vendors are starting to add data mesh capabilities to their existing offerings.

- **Analytical capabilities**. The data platform needs functions to ingest, store, query, and visualize data as a self-service.
- **Data product capabilities**. For creating, monitoring, discovering, and accessing data products. A data catalog can be implemented in different ways: as a wiki, git repository, or there are even already vendor solutions for a cloud-based data catalog such as Select Star, Google Data Catalog, or AWS Glue Data Catalog.
- **Policy automation**. The policies are automatically enforced through the platform

A shared platform with a single query language and support for separated areas is a good way to start as everything is highly integrated. This could be Google BigQuery with tables in multiple projects that are discoverable through Google Data Catalog.
In a more decentralized and distributed data mesh, a distributed query engine such as Presto can still perform cross-domain joins without importing data, but they come with their own limitations.

Technology

There are a lot of different ways to implement a data mesh architecture.

Here is a selection of typical tech stacks that we saw:

- Google Cloud BigQuery
- AWS S3 and Athena
- Azure Synapse Analytics
- dbt and Snowflake
- Starburst Enterprise (TBD)
- Databricks (TBD)

Model management

ARCHITECTURAL PATTERNS

ML workflows can employ different forms.
This is defined by Model Architectural Patterns.

Model Architectural Patterns

MLOps • 2022

Two dimensions

Operating an ML model might assume several architectural styles.

We discuss four architectural patterns which are classified along two dimensions:

1. **ML Model Training**,
2. **ML Model Prediction**,
   - For simplicity, we disregard the third dimension:
3. **ML Model Type**, which denotes the type of machine learning algorithm such as supervised, unsupervised, semisupervised, and Reinforcement Learning.

ML Model Training

There are 2 ways how we perform ML Model Training:

1. **Offline learning** (aka **batch** or **static learning**):
    - The model is trained on a set of already collected data.
    - After deploying to the production environment, the ML model remains constant until it re-trained because the model will see a lot of real-live data and becomes stale. This phenomenon is called '**model decay**' and should be carefully monitored.
2. **Online learning** (aka **dynamic learning**):
    - The model is regularly being re-trained as new data arrives, e.g. as data streams.
    - This is usually the case for ML systems that use time-series data, such as sensor, or stock trading data to accommodate the temporal effects in the ML model.

ML Model Prediction

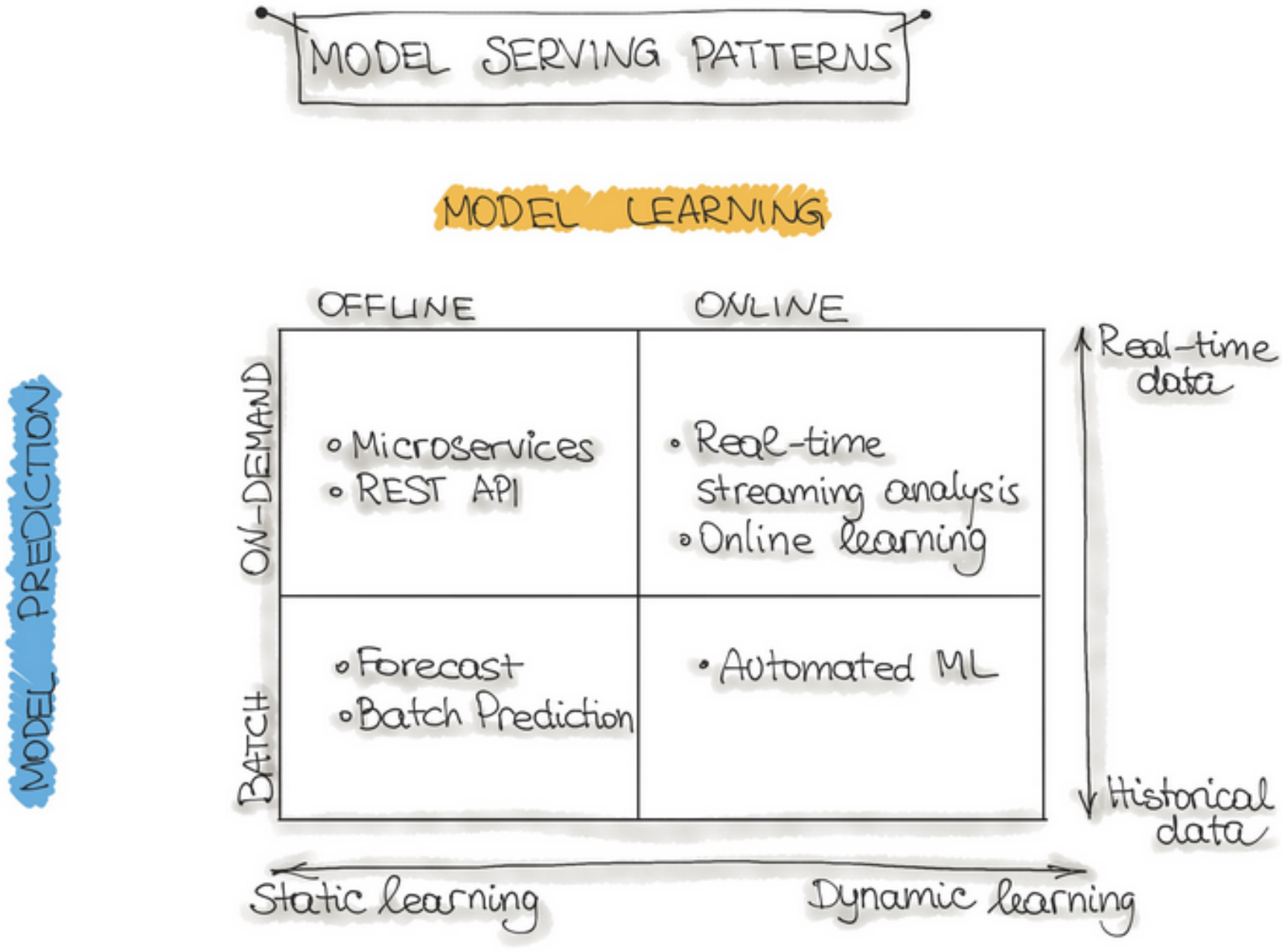
This denotes the mechanics of the ML model to makes predictions.

Here we also distinguish two modes:

1. **Batch predictions**
    - The deployed ML model makes a set of predictions based on **historical input data**.
    - This is often sufficient for data that is not time-dependent, or when it is not critical to obtain real-time predictions as output.
2. **Real-time predictions** (aka **on-demand predictions**)
    - Predictions are generated in real-time using the **input data that is available** at the time of the request.

4 ML architecture patterns

We can classify the operationalization of machine learning models into four ML architecture patterns.



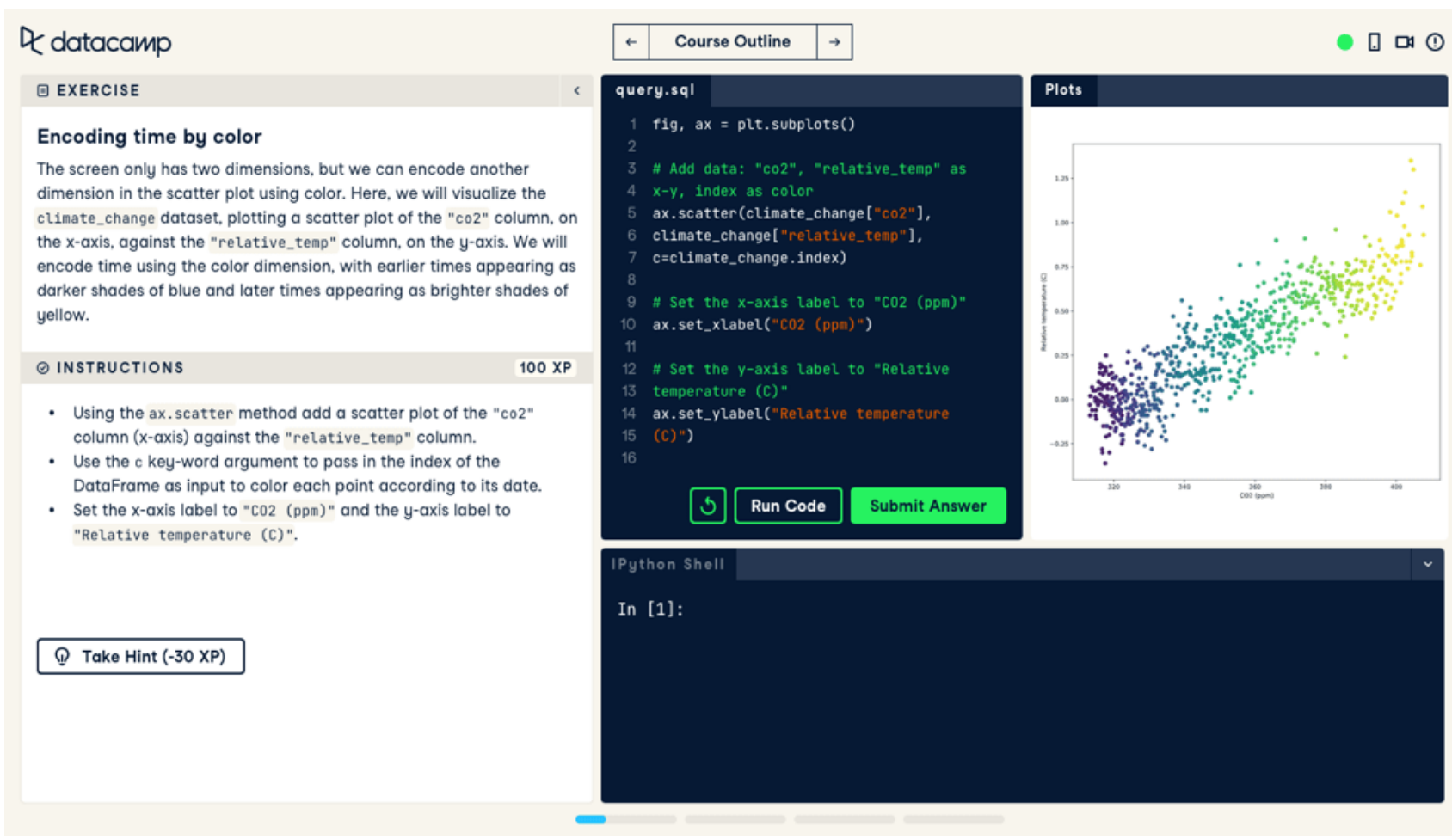
Forecast

Model Learning: Offline.
Model Prediction: Batch.

This ML workflow is not very useful and, therefore, not common in an industry setting for production systems.

This type of machine learning workflow is widely spread in **academic research or data science education** (e.g., Kaggle or DataCamp).

This form is used to experiment with ML algorithms and data as it is the easiest way to create a machine learning system. Usually, we take an available dataset, train the ML model, then run this model on another (mostly historical) data, and the ML model makes predictions.

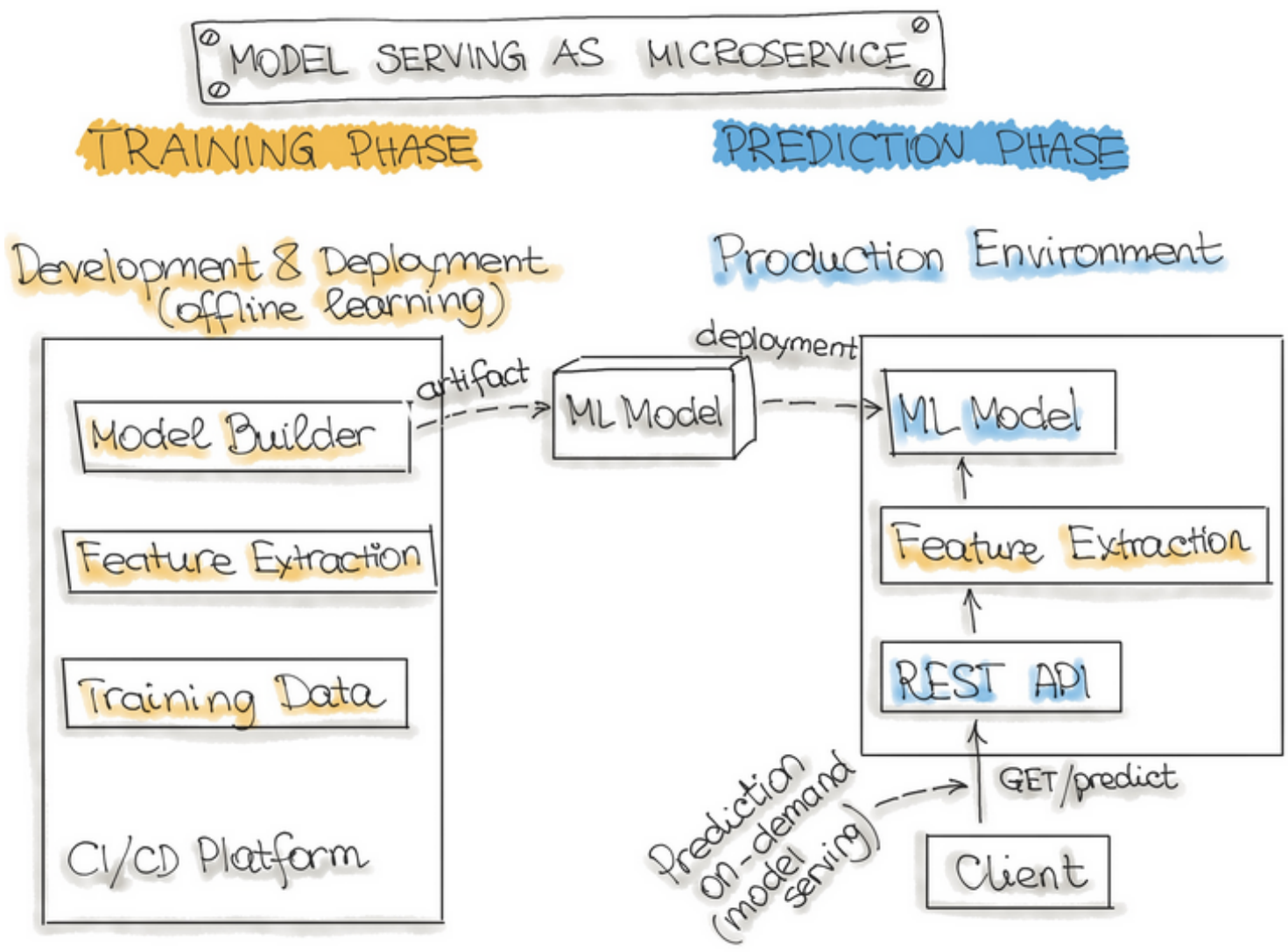


Web-Service

Model Learning: Offline.
Model Prediction: On-demand.

The most commonly described deployment architecture for ML models is a web service (**microservise**).

The model is trained offline on historical data, but it uses real-live data to make predictions. The difference from a forecast (batch predictions) is that the ML model runs near real-time and handles a single record at a time instead of processing all the data at once. But the model remains constant until it is re-trained and re-deployed into the production system.



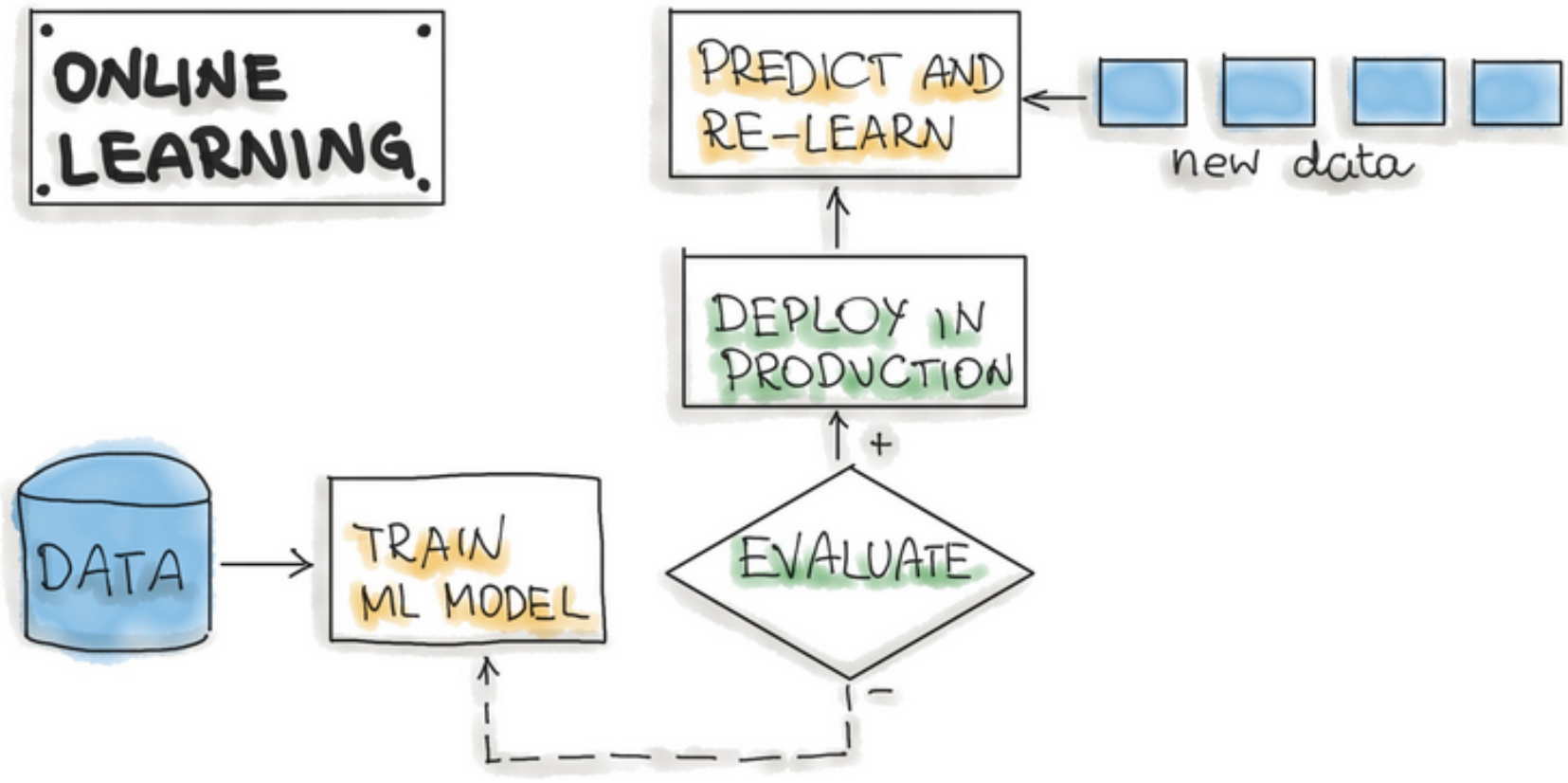
Online Learning or Real-time Streaming Analysis

Model Learning: Online.
Model Prediction: On-demand.

The most dynamic way to embed machine learning into a production system is to implement **online learning**, which is also known as r**eal-time streaming analytics**.
The ML learning algorithm is continuously receiving a data stream, either as **single data points** or in small groups called **mini-batches**. The system learns about new data **on the fly** as it arrives, so the ML model is incrementally being re-trained with new data. This continually re-trained model is instantly available as a web service.

Technically, this type of ML system works well in big data systems. The model would typically run as a service on a Kubernetes cluster or similar.

A big difficulty with the online learning system in production is that if **bad data** is entering the system, the ML model, as well as the whole system performance, will increasingly decline.



Automated Machine Learning or AutoML

Model Learning: Online.
Model Prediction: Batch.

AutoML is getting a lot of attention and is considered the next advance for enterprise ML.
The user needs to provide data, and the AutoML system automatically selects an ML algorithm, such as **neural network architecture**, and configures the selected algorithm.

For now, this is a very **experimental way** to implement ML workflows. AutoML is usually provided by big cloud providers, such as Google or MS Azure. However, models build with AutoML need to reach the level of accuracy required for real-world success.

Model management

DISTRIBUTABLE FORMAT

The ML model should be present and should be executable as an independent asset. This means that the ML models should work outside of the model-training environment.

ML Model Serialization Formats

MLOps • 2022

ML Model Serialization Formats

PORTABLE FORMATS

One approach to reducing dependencies on the production environment is to export the model to a **portable format** such as PMML, PFA, ONNX, or POJO. These aim to improve model portability between systems and simplify deployment.

In the following, we describe **Language-agnostic** and **Vendor-specific exchange formats** for ML models.

Language-agnostic exchange formats

AMALGAMATION

Amalgamation is the simplest way to export an ML model. The model and all necessary code to run are bundled as **one package**. Usually, it is a single source code file that can be compiled on nearly any platform as a standalone program.

- For example, we can create a standalone version of an ML model by using SKompiler.

With some easy ML algorithms, this format is compact and might have good performance.
However, the **ML model code and parameters** need to be managed together.

SKOMPILER

This python package provides a tool for transforming **trained Scikit-learn models** into other forms, such as **SQL queries, Excel formulas, Portable Format for Analytics (PFA) files, or SymPy expressions**. The last can be translated to **code** in a variety of languages, such as **C, Javascript, Rust, Julia, etc.**

Language-agnostic exchange formats

PMML

PMML (Predictive Model Markup Language) is a format for model serving based on **XML** with the file extension .pmml.
PMML has been standardized by the Data Mining Group (DMG). Basically, **.ppml describes a model and pipeline in XML**. The PMML supports not all of the ML algorithms, and its usage in open source-driven tools is limited due to licensing issues.

PFA

PFA (Portable Format for Analytics) is designed as a replacement for PMML.
From DMG: “A PFA document is a string of **JSON-formatted text** that describes an executable called a **scoring engine**. Each engine has a well-defined **input**, a well-defined **output**, and **functions** for combining inputs to construct the output in an **expression-centric syntax tree**”. PFA capabilities include:

- control structures, such as conditionals, loops, and user-defined functions,
- expressed within JSON, and can be easily generated and manipulated by other programs,
- fine-grained function library supporting extensibility callbacks.

Language-agnostic exchange formats

ONNX

ONNX (Open Neural Network eXchange) is an ML framework **independent file format.**
ONNX was created to allow any ML tool to share a single model format. This format is supported by many big tech companies such as Microsoft, Facebook, and Amazon.

Once the ML model is serialized in the ONNX format, it can be consumed by **onnx-enabled runtime libraries** (also called inference engines) and then make predictions.

Most deep learning tools have **ONNX support**.

Tools that can use ONNX format:

- CoreML
- MATLAB
- Menoh
- MXNet
- Tensorflow
- TensorRT
- Windows ML
- Other: https://github.com/onnx/tutorials#scoring-onnx-models

Vendor-specific exchange formats

EXAMPLES

- **Scikit-Learn** saves models as pickled **python objects**, with a .pkl file extension.
- **H2O** allows you to convert the models you have built to either **POJO (Plain Old Java Object) or MOJO (Model Object, Optimized)**.
- **SparkML** models that can be saved in the **MLeap file** format and served in real-time using an MLeap model server. The MLeap runtime is a **JAR** that can run in any **Java application**. MLeap supports **Spark**, **Scikit-learn**, and **Tensorflow** for training pipelines and exporting them to an MLeap Bundle.
- **TensorFlow** saves models as .pb files; which is the protocol buffer files extension.
- **PyTorch** serves models by using their proprietary **Torch Script** as a .pt file. Their model format can be served from a **C– application**.
- **Keras** saves a model as a .h5 file, which is known in the scientific community as a data file saved in the **Hierarchical Data Format (HDF)**. This type of file contains multidimensional arrays of data.
- **Apple** has its proprietary file format with the extension .mlmodel to store models embedded in iOS applications. The **CoreML** framework has native support for **Objective-C and Swift** programming languages. Applications trained in other ML frameworks, such as TensorFlow, Scikit-Learn, and other frameworks need to use tools like such as **coremltools** and **Tensorflow converter** to translate their ML model files to the .mlmodel format for use on iOS.

See:
https://towardsdatascience.com/guide-to-file-formats-for-machine-learning-columnar-training-inferencing-and-the-feature-store-2e0c3d18d4f9

Code management

DEPLOYING IN A PRODUCTION ENVIRONMENT

Three components should be considered when we serve an ML model in a production environment. The **inference** is the process of getting data to be ingested by a model to compute predictions. This process requires a **model**, an **interpreter** for the execution, and **input data**.

MODEL SERVING PATTERNS

Model serving is a way to integrate the ML model in a software system.

Model Serving Patterns

MLOps • 2022

Model Serving Patterns

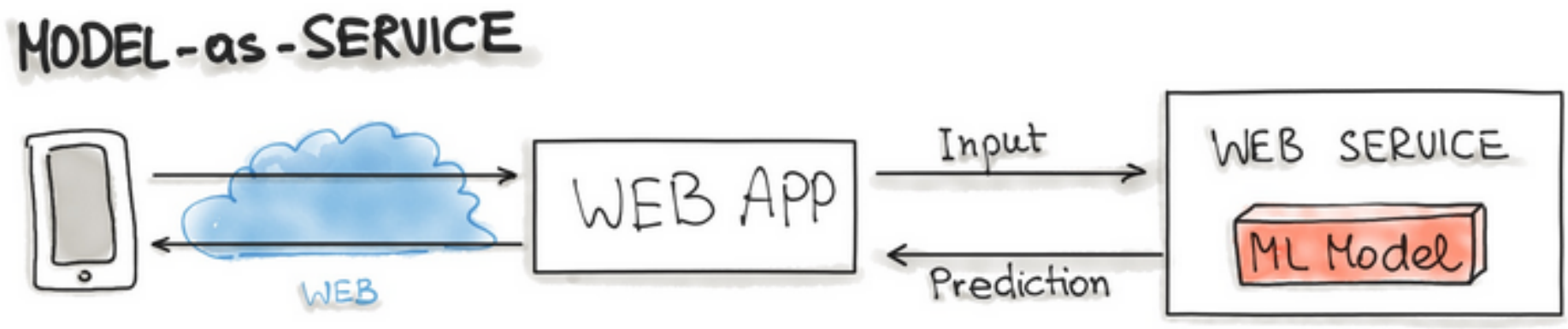
Model serving is a way to integrate the ML model in a software system.

We distinguish between five patterns to put the ML model in production: Model-as-Service, Model-as-Dependency, Precompute, Model-on-Demand, and Hybrid-Serving.

Model-as-Service

Model-as-Service, or **live-scoring model,** is a common pattern for wrapping an ML model as an independent service. We can wrap the ML model and the interpreter within a dedicated **web service** that applications can request through a **REST API** or consume as a gRPC service.

This pattern can be used for various ML workflows, such as Forecast, Web Service, Online Learning.

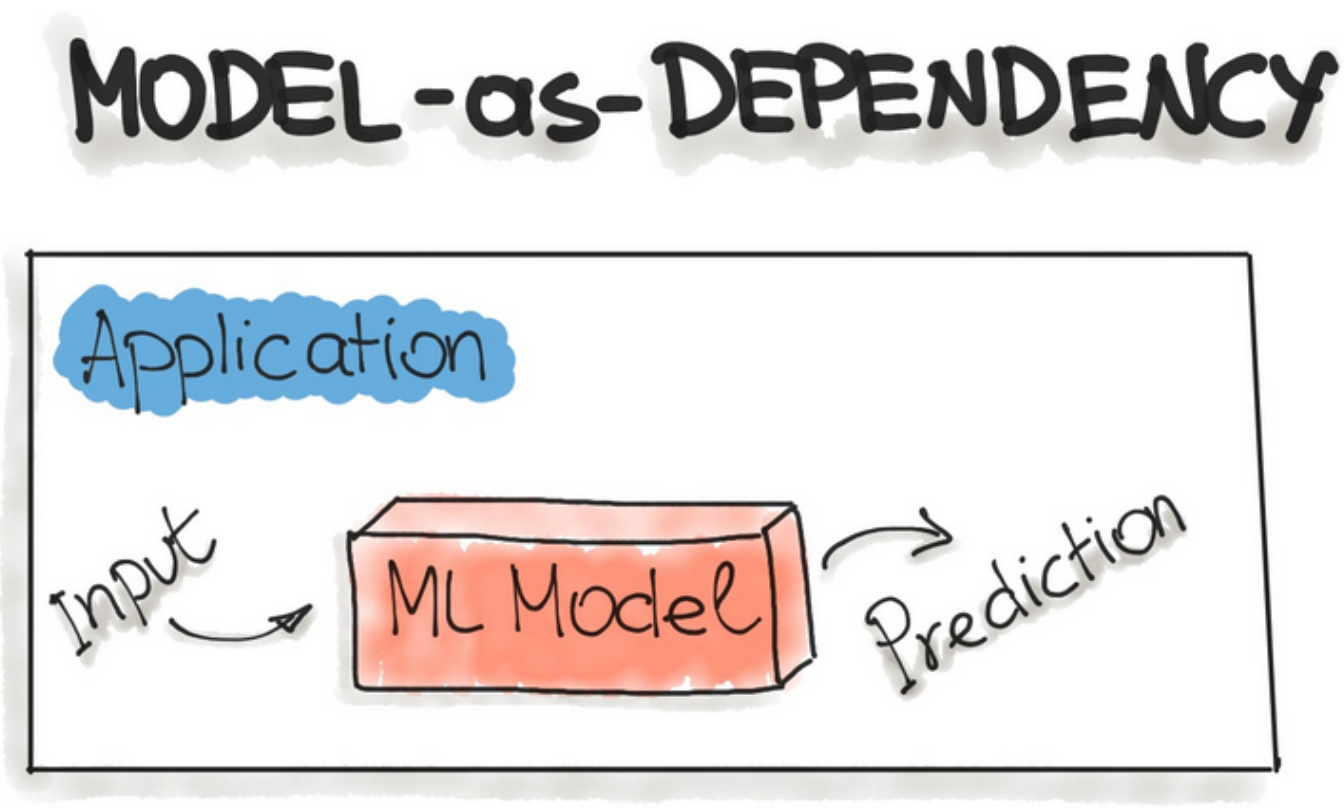


Model-as-Dependency

Model-as-Dependency is probably the most straightforward way to package an ML model. A packaged ML model is considered as a **dependency within the software application**.

For example, the application consumes the ML model like a conventional **jar dependency by invoking the prediction method and passing the values**. The return value of such method execution is some prediction that is performed by the previously trained ML model.

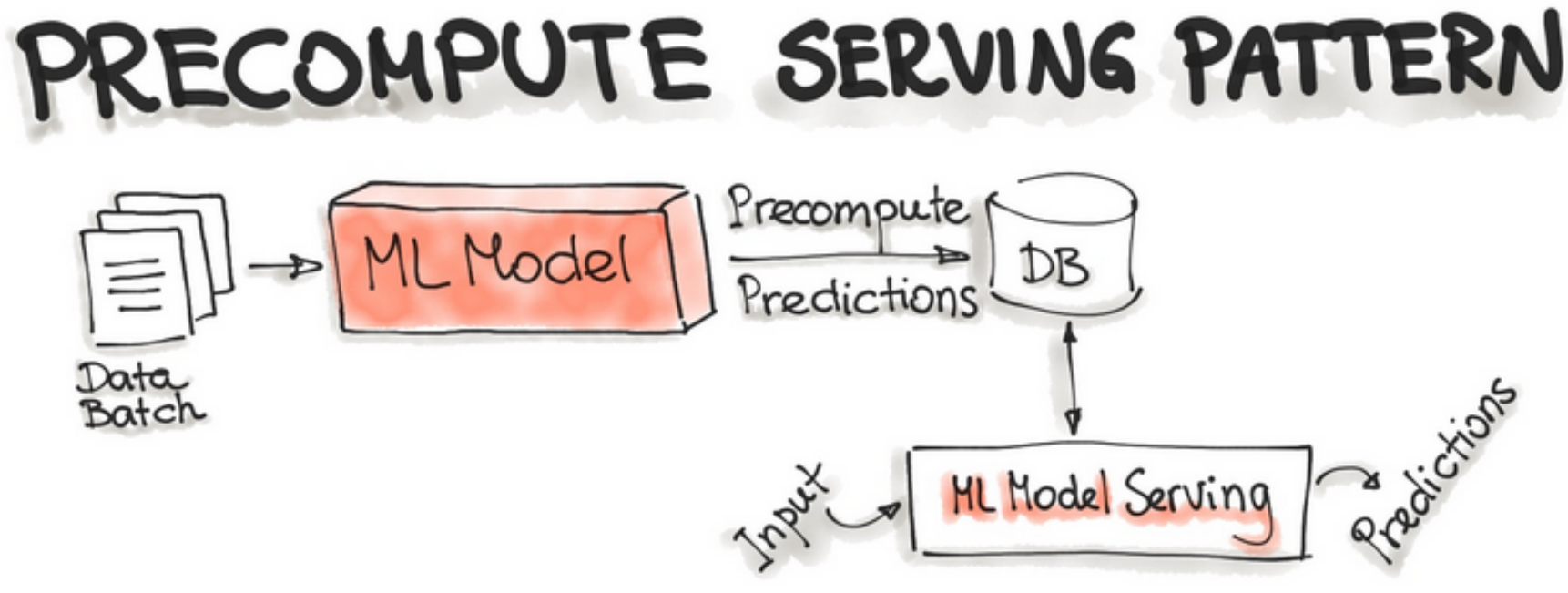
The Model-as-Dependency approach is mostly used for implementing the Forecast pattern.



Precompute Serving Pattern

This type of ML model serving is tightly related to the Forecast ML workflow.

With the Precompute serving pattern, we use an already trained ML model and **precompute the predictions for the incoming batch of data**.
The resulting predictions **are persisted in the database**. Therefore, for any input request, we query the database to get the prediction result.



See:
https://www.slideshare.net/mikiobraun/bringing-ml-to-production-what-is-missing-amld-2020

Model-on-Demand

The Model-on-Demand pattern also treats the ML model as a dependency that is available at runtime. This ML model, contrary to the Model-as-Dependency pattern, has its own **release cycle** and is **published independently**.

The **message-broker** architecture is typically used for such on-demand model serving. The message-broker topology architecture pattern contains two main types of architecture components:

- a **broker** component
- an **event processor** component.

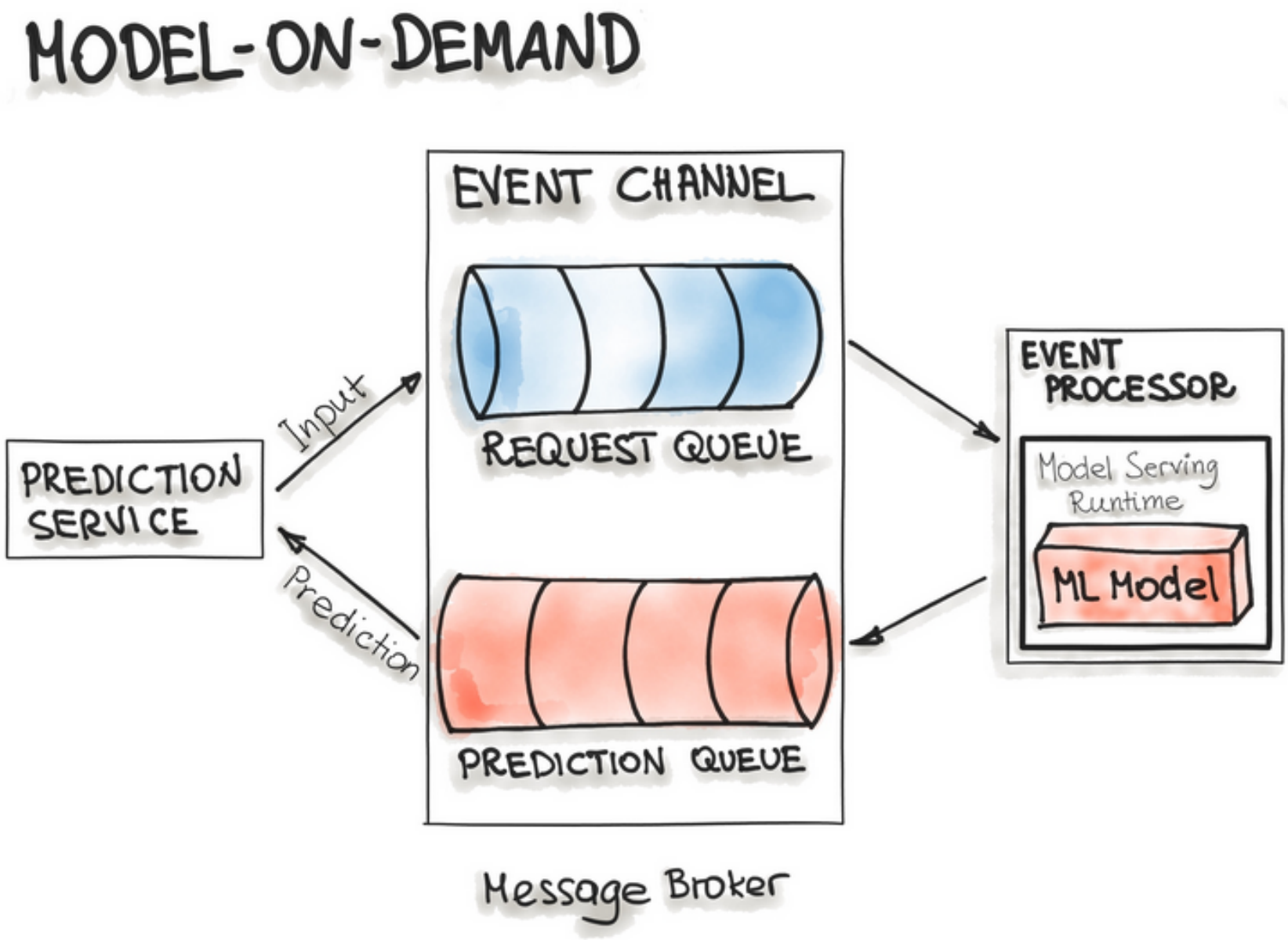
The broker component

The broker component is the central part that contains the **event channels** that are utilised within the **event flow**. The event channels, which are enclosed in the broker component, are **message queues**. We can imagine such architecture containing input- and output-queues. A message broker allows one process to write prediction-requests in an input queue.

The event processor

The event processor contains the **model serving runtime** and the **ML model**. This process **connects** to the broker, **reads** these requests in batch from the queue and **sends** them to the model to make the predictions. The model serving process runs the **prediction generation** on the input data and writes the **resulted predictions** to the output queue. Afterwards, the queued prediction results are pushed to the prediction service that initiated the prediction request.

Model-on-Demand



Hybrid-Serving (Federated Learning)

It is unique in the way it does, there is **not only one model** that predicts the outcome, but there are also lots of it.

Let us start with the unique model, the one on the server. The **model on the server-side** is trained only once with the real-world data. It sets the initial model for each user. Also, it is a relatively general trained model so it fits for the majority of users.

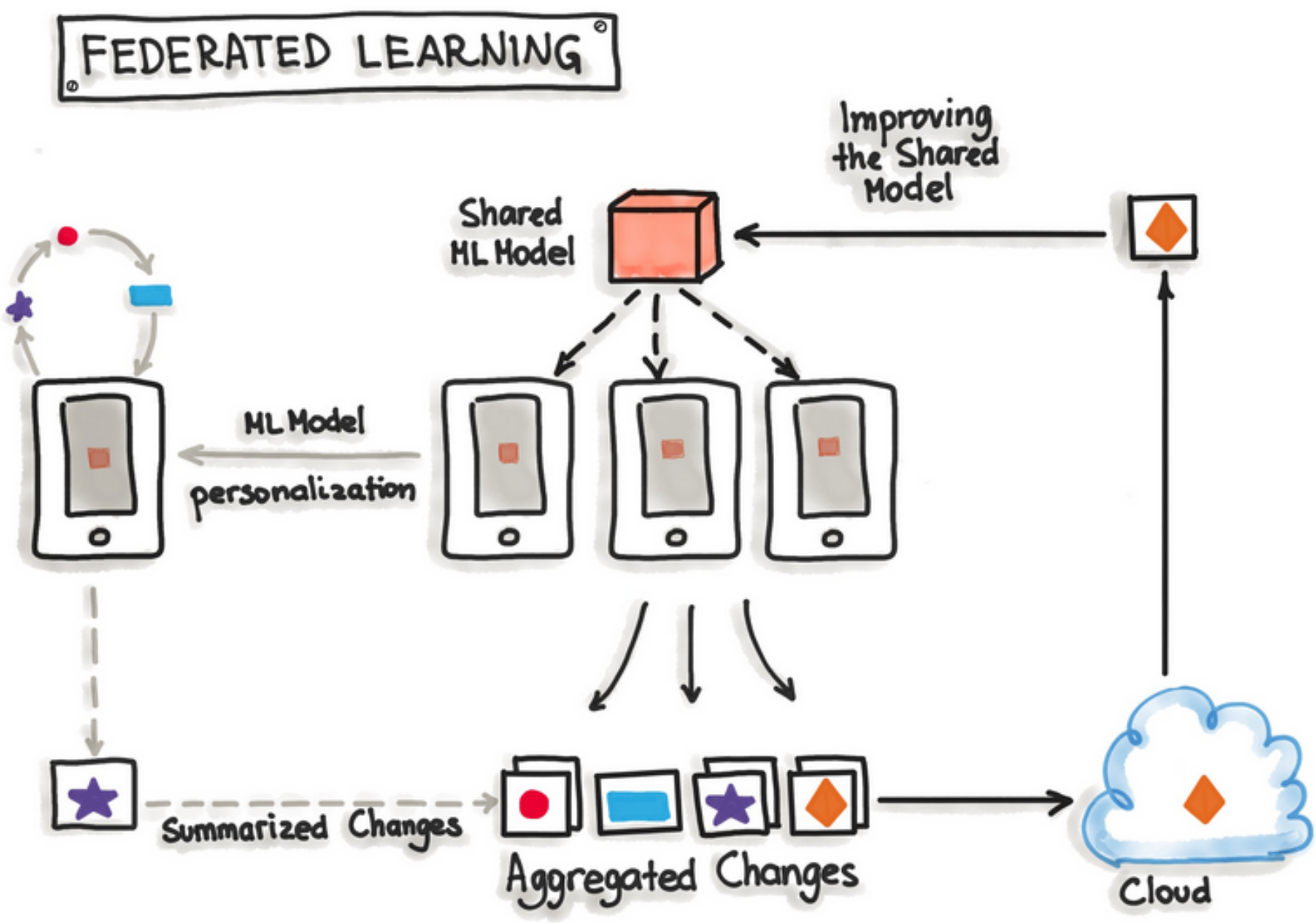
On the other side, there are the **user-side models,** which are the real unique models. Due to the raising hardware standards on mobile devices, it is possible for **the devices to train their own models**. Like that the devices will train their **own highly specialized model** for their own user. Once in a while, **the devices send their already trained model data to the server**.

The server model is set to be the new initial model that all devices are using. For not having any downsides for the users, while the server model gets updated, this happens only when the **device is idle, connected to WiFi and charging**. Also, the testing is done on the devices, therefore the newly adopted model from the server is sent to the devices and tested for functionality.

Hybrid-Serving (Federated Learning)

The big benefit of this is that the **data** used for training and testing, which is **highly personal**, never leaves the devices while still capturing all data that is available. This way it is possible to **train highly accurate models** while not having to store tons of data in the cloud.

With Federated Learning there are other circumstances, the **mobile devices** are **less powerful**, the **training data is distributed** across millions of devices and **these are not always available** for training. Exactly for this TensorFlow Federated (TFF) has been created. TFF is a lightweight form of TensorFlow created for Federated Learning.



Code management

DEPLOYING ML MODELS

For wrapping trained models as deployable services, there are different Deployment Strategies.

MLOps • 2022

Deployment Strategies

MLOps • 2022

Deployment Strategies

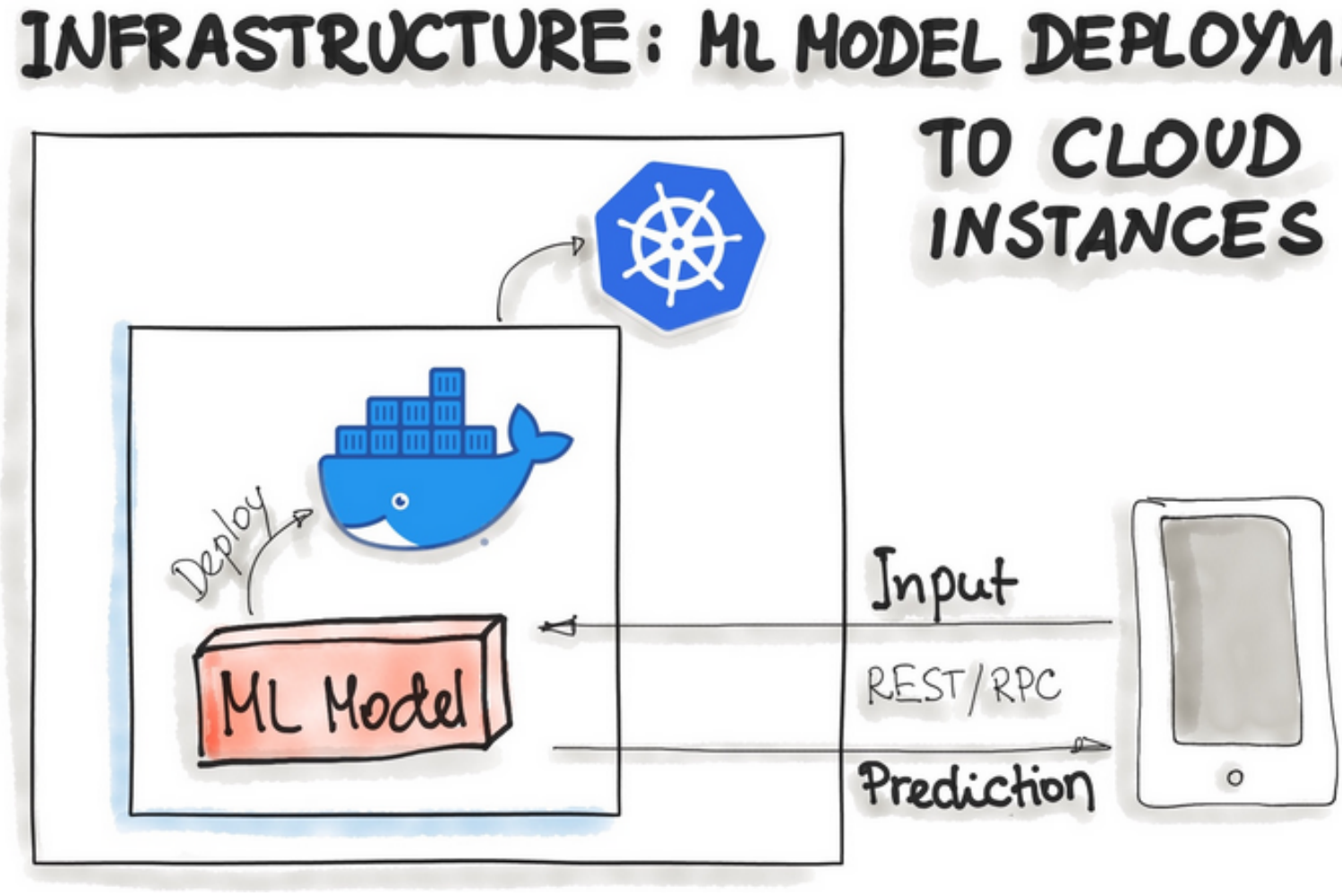
There are different ways for wrapping trained models as deployable services.

For example, **deploying ML models as Docker Containers to Cloud Instances** and **as Serverless Functions**.

Deploying ML Models as Docker Containers

As of now, there is no standard, open solution to ML model deployment. As ML model inference being considered stateless, lightweight, and idempotent, **containerization** becomes the de-facto standard for delivery. This means we deploy a container that wraps an ML model inference code.
For on-premise, cloud, or hybrid deployments, Docker is considered to be **de-facto standard containerization technology**.

One ubiquitous way is to package the **whole ML tech stack (dependencies) and the code for ML model prediction** into a Docker container.
Then Kubernetes or an alternative (e.g. AWS Fargate) does the orchestration. The ML model functionality, such as prediction, is then available through a REST API (e.g. implemented as Flask application).

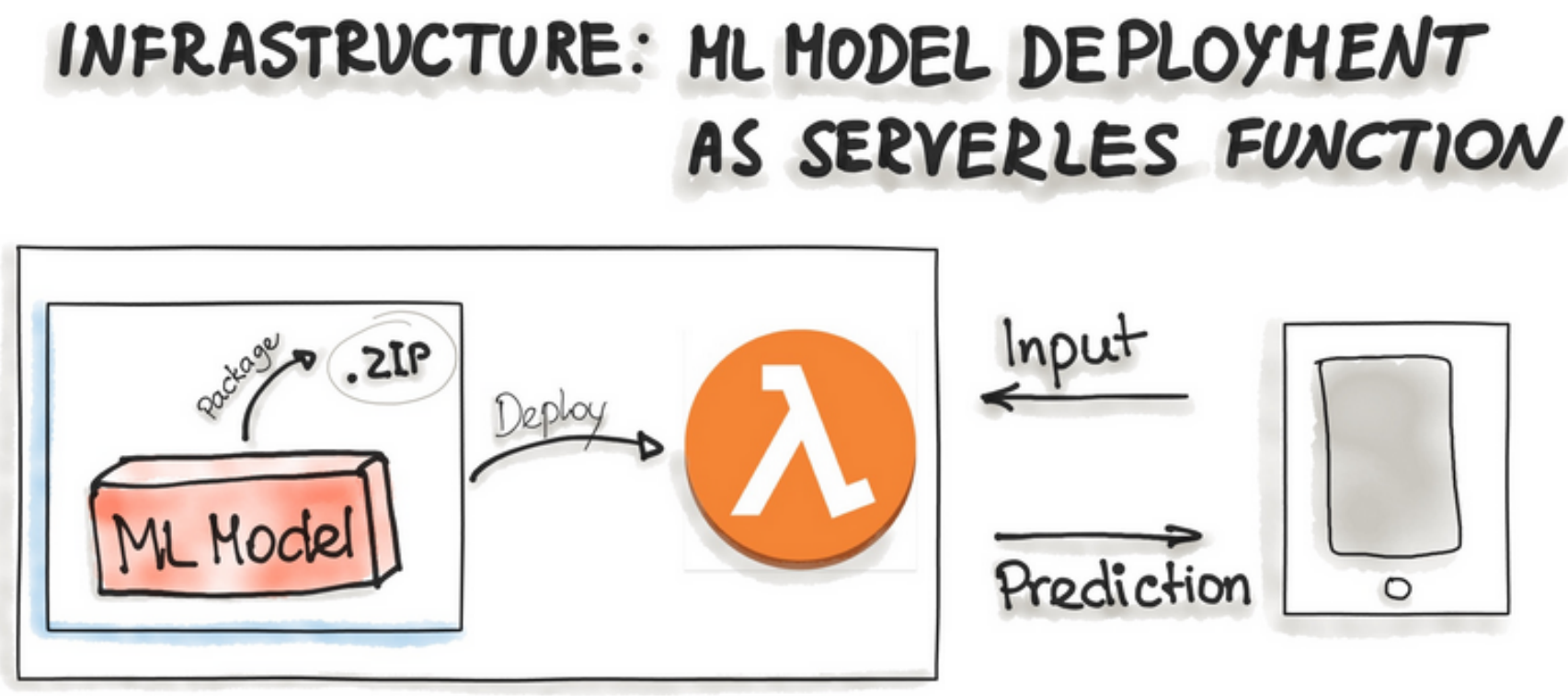


Deploying ML Models as Serverless Functions

Various cloud vendors already provide **machine-learning platforms**, and you can deploy your model with their services. Examples are Amazon AWS Sagemaker, Google Cloud AI Platform, Azure Machine Learning Studio, and IBM Watson Machine Learning, to name a few. **Commercial cloud services** also provide containerization of ML models such as AWS Lambda and Google App Engine servlet host.

In order to deploy an ML model as a serverless function, the **application code and dependencies** are packaged into **.zip files**, with a **single entry point function**. This function then could be managed by major cloud providers such as Azure Functions, AWS Lambda, or Google Cloud Functions.

However, attention should be paid to possible **constraints of the deployed artifacts** such as the size of the artifact.



Choosing the Right Metric for Evaluating

MLOps • 2022

Deployment Strategies

See:

- https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4
- https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428

Tools

Cloud Providers & Tools

Existing cloud providers already working on offering machine learning platforms such as **AI Platform by Google Cloud, AzureML**, and **SageMaker by AWS**.
Initiated at Google, the **Kubeflow** project presents an option to manage a set of open-source tools for MLOps and assemble them on **Kubernetes**.

Tools for non-cloud systems like **MLFlow, Sacred, or DVC**.

- https://mlflow.org/
- https://github.com/IDSIA/sacred
- https://dvc.org/

The structured way to proceed in the MLOps tech stack selection is to use the **MLOps Stack Template**. This template breaks down a machine learning workflow into nine components, as described in the MLOps Principles.

- https://valohai.com/blog/the-mlops-stack/

The **Linux Foundation's LF AI** project created a visualization for the ML/AI and MLOps tools. Another curated list of the production machine learning tools is maintained by the **Institute for Ethical AI**.

- https://landscape.lfai.foundation/
- https://github.com/EthicalML/awesome-production-machine-learning

Cloud Providers & Tools

Spending on cloud infrastructure services reached a record $30 billion in the second quarter of 2020, with **Amazon Web Services (AWS), Microsoft, and Google Cloud** accounting for half of customer spend.

These companies invest in research & development of specialized hardware, software, and SaaS applications, but also MLOps software. Two great examples come to mind:

- **AWS** with its **Sagemaker**, a fully managed end-to-end cloud ML-platform that enables developers to create, train, and deploy machine-learning models in the cloud, embedded systems, and edge-devices.
- **Google** with its recently announced **AI Platform Pipelines** for building and managing ML pipelines, leveraging **TensorFlow Extended (TFX's)** pre-built components and **templates** that do a lot of model deployment work for you.

End-to-end MLOps solution

These are fully managed services that provide developers and data scientists with the ability to build, train, and deploy ML models quickly.

AMAZON

- **Amazon Sagemaker**, a suite of tools to build, train, deploy, and monitor machine learning models.
  - https://aws.amazon.com/it/sagemaker/

MICROSOFT AZURE:

- **Azure Machine Learning** to build, train, and validate reproducible ML pipelines
  - https://azure.microsoft.com/en-us/services/machine-learning/
- **Azure Pipelines** to automate ML deployments
  - https://azure.microsoft.com/en-us/services/devops/pipelines/
- **Azure Monitor** to track and analyze metrics
  - https://docs.microsoft.com/en-us/azure/azure-monitor/overview
- **Azure Kubernetes Services** and other additional tools
  - https://azure.microsoft.com/en-us/services/kubernetes-service/

End-to-end MLOps solution

GOOGLE CLOUD

- **Dataflow** to extract, validate, and transform data as well as to evaluate models
  - https://cloud.google.com/dataflow
- **AI Platform Notebook** to develop and train models
  - https://cloud.google.com/vertex-ai-workbench
- **Cloud Build** to build and test machine learning pipelines
- **TFX** to deploy ML pipelines
  - https://www.tensorflow.org/tfx
- **Kubeflow Pipelines** to arrange ML deployments on top of **Google Kubernetes Engine** (GKE).
  - https://www.kubeflow.org/docs/components/pipelines/introduction/
  - https://cloud.google.com/kubernetes-engine

Some MLOps Tools

TOOLS

- **Project Jupyter**
  - https://jupyter.org/
- **Nbdev**
  - https://github.com/fastai/nbdev
- **Airflow**
  - https://airflow.apache.org/
- **Kubeflow**
  - https://www.kubeflow.org/
- **MLflow**
  - https://mlflow.org/
- **Optuna**
  - https://optuna.org/
- **Cortex**
  - https://www.cortex.dev/
- **Neptune** (for its easy and fast experiment tracking and compatibility with a lot of tools like Sagemaker and MLflow; if there isn't an integration guide or pre-built solution, you can use their Python client API to build a custom integration)
  - https://neptune.ai/home

Cloud Providers & Tools

TOOLS

- https://github.com/EthicalML/awesome-production-machine-learning

TOOLS by Category

- https://mlops.community/learn/
- https://twimlai.com/solutions/
- https://neptune.ai/blog/best-mlops-tools

Youtube

- ML Ops Best Practices on Google Cloud (Cloud Next '19)
    - https://neptune.ai/blog/mlops

MLOps Stack

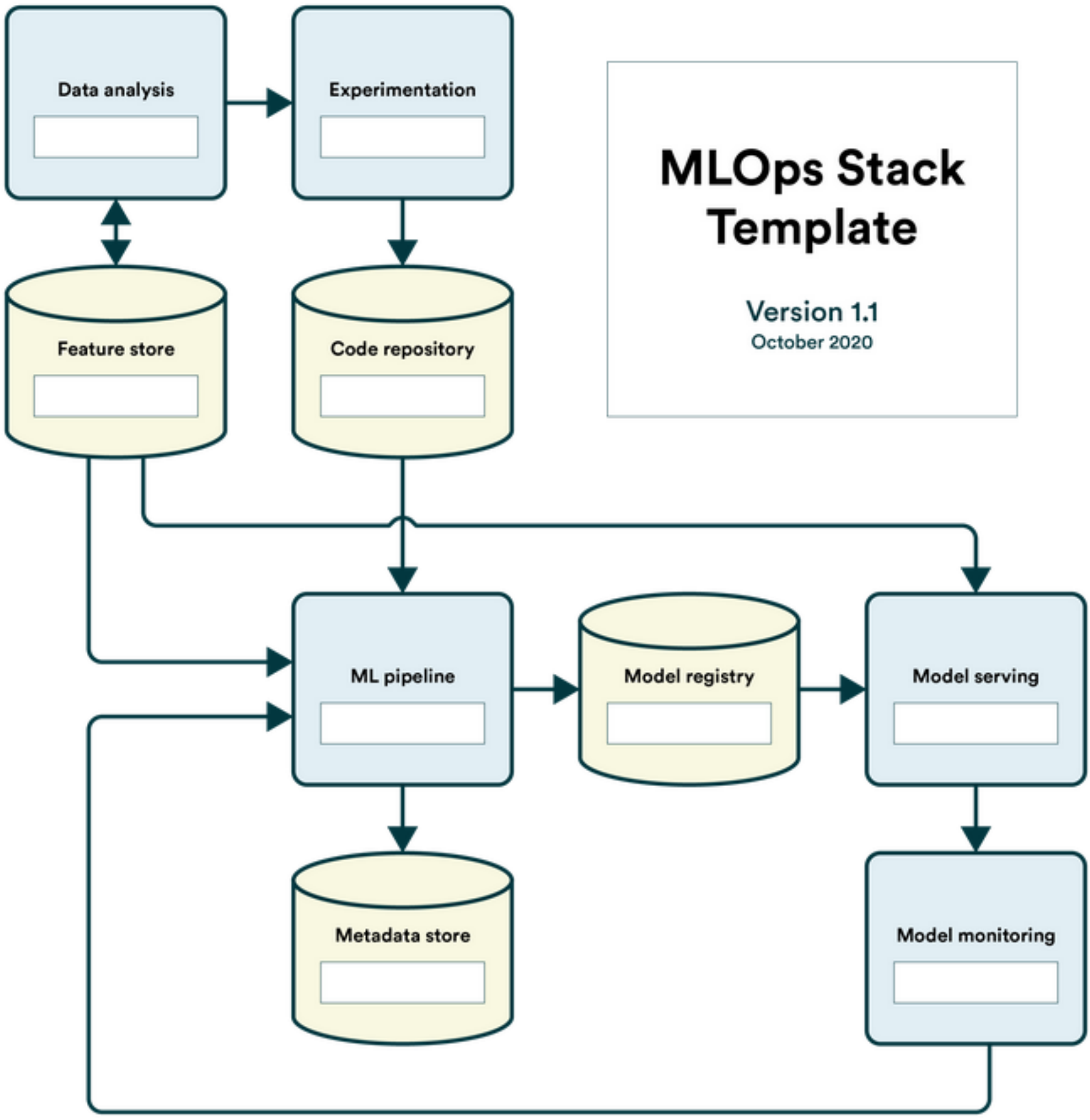
WHAT AND WHERE USE TOOLS

To make it easier to consider what tools your organization could use to adopt MLOps, there is **MLOps Stack Templete**, a simple template that breaks down a machine learning workflow into components.

MLOps Stack Template

MLOps • 2022

MLOps Stack Template



MLOps Stack Template

This template allows you to consider where you need tooling.

As a machine learning practitioner, you'll have a lot of choices in technologies.

First, some technologies cover multiple components, while others are more singularly focused.
Second, some tools are open-source and can be implemented freely, while others are proprietary but save you the implementation effort.

Download the MLOps Stack Template

- https://valohai.com/assets/files/mlops-stack.pdf

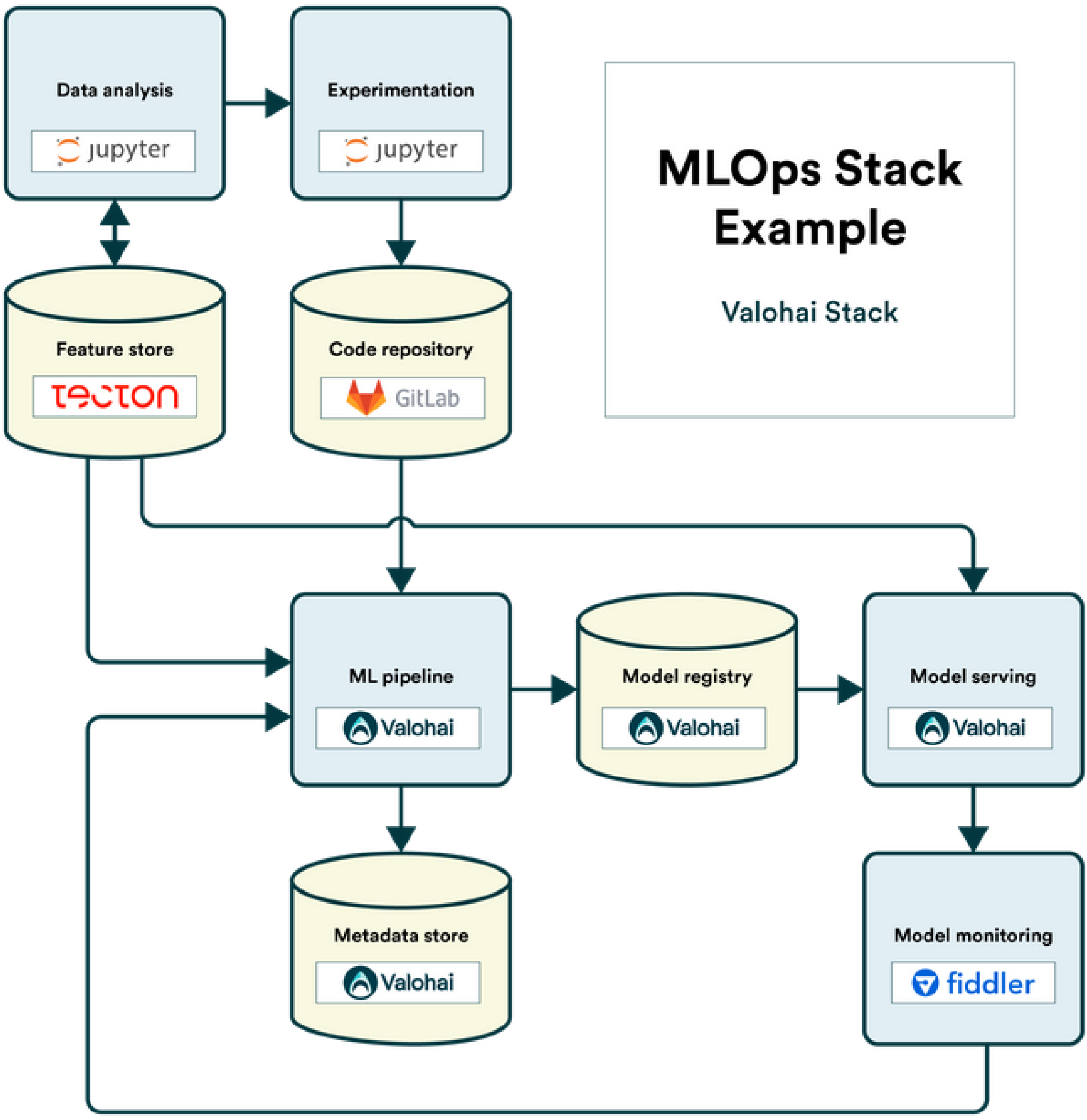
MLOps Stack Template

9 COMPONENTS

The MLOps Stack template is loosely based on **Google's article on MLOps and continuous delivery** (https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning), but we tried to simplify the workflow to a more manageable abstraction level. There are nine components in the stack which have varying requirements depending on your specific use case.

Component	Requirements	Tooling
Data analysis		
Experimentation		
Feature store		
Code repository		
ML pipeline		
Metadata store		
Model registry		
Model serving		
Model monitoring		

Example


Data analysis
jupyter
Experimentation
jupyter
MLOps Stack Example
Valohai Stack
Feature store
tecton
Code repository
GitLab
ML pipeline
Valohai
Model registry
Valohai
Model serving
Valohai
Metadata store
Valohai
Model monitoring
fiddler

Example

As an example, we've put together a technology stack containing our MLOps platform, Valohai, and some of our favorite tools that work complementary to it, including:

- **JupyterHub** / Jupyter Notebook for data analysis and experimentation
    - https://jupyter.org/hub
- **Tecton** for feature stores
    - https://www.tecton.ai/
- **GitLab** for code repositories
    - https://about.gitlab.com/
- **Fiddler Lab**s for model monitoring
    - https://www.fiddler.ai/
- **Valohai** for training pipelines, model serving, and associated stores
    - https://www.fiddler.ai/

Tools by Service Category

The service categories provided below group the function sets necessary to build scalable machine lean

Service Category	Service Description	Available Implementations
ML Platform	Provisioned as our opinionated preference for ML workflows running on a highly scalable software infrastructure.	Kubeflow, Kubernetes
ML Frameworks	Select your machine learning and deep learning framework, toolkit, and libraries.	TensorFlow, Keras, Cafee, PyTorch
Storage Volume Management	Choose from software and tools for storage to meet your high performance ML needs	Local FS, AWS EFS, AWS EBS, Ceph (block and object), Minio, NFS, HDFS
Container Image Governance	Choose from software and tools that register, secure and manage the distribution of container images.	AWS ECR, Harbor, GitLab
Workflow Engine	Provisioned by default to govern scheduling and coordination of jobs	Argo
Model Training	Include collaboration tooling and interactive model training as part of your template	JupyterHub, TensorBoard, Argo workflow templates
Model Serving	Pick the tool to expose trained models to business applications.	Seldon, tf-serving
Model Validation	Set by default, models will be evaluated against test data as part of your ML pipeline.	Argo workflow templates
Data Storage Services	Choose from storage options befitting the performance of other ML services.	Mini, AWS S3, MongoDB, Cassandra, HDFS
Data Preparation & Processing	Select your tooling to manage the data processing event of your ML pipeline	Argo, NATS, workflow application templates
Infrastructure Monitoring	Elect which reporting and dashboarding tool gives you the better optics into your stack performance.	Prometheus, Grafana
Model Monitoring	Find and choose the appropriate tool to watch model accuracy over time.	Prometheus, Grafana, Isto
Load Balancing & Ingress	Determine the appropriate tool to expose cluster services broadly to other application services.	ELB, Traefik, Ambassador
Security	Find the right tooling for you to manage certificates, passwords and secrets tuned for RBAC across all hybrid-cloud environments.	Okta, Hashicorp Vault, AWS Certificate Manager
Log Management	Make logging easier by choosing pre-integrated tools for ingest, analysis and reporting.	ElasticSearch, Fluentd, Kibana

Download

- https://ml-ops.org/img/mlops-agile-stack.png

CRISP-ML(Q)

INFRASTRUCTURE STACK

One of the main issues for ML-adopters is technology integration and compatibility.

To specify an architecture and infrastructure stack for Machine Learning Operations we can use frameworks and tools.

Tool: MLOps Stack Canvas

MLOPS STACK CANVAS

MLOps Stack Canvas is a framework designed to be application- and industry-neutral, that support the understanding of requirements for the ML system and the audit process of the infrastructure components.

MLOps Stack Canvas

MLOps Stack Canvas

This canvas condenses the main elements of a whole technology stack for an ML application into a single page.
This framework guides the development teams through the MLOps building blocks and lets them answer the MLOps infrastructure-related questions and identify the necessary tools chain.

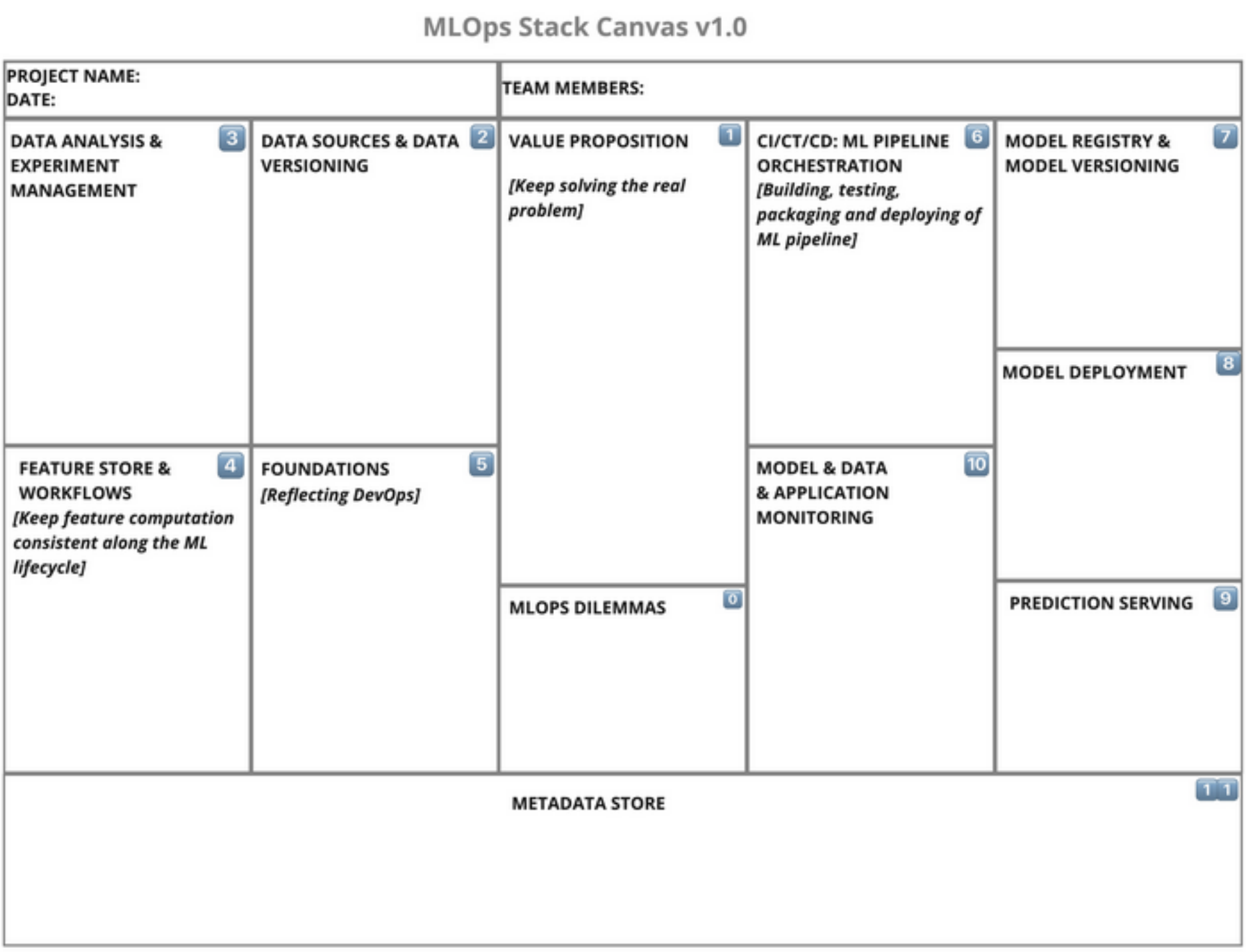
The purpose of the MLOps Stack Canvas is to help you to **structure workflows, architecture, and infrastructure components** for the MLOps stack in the ML project.

The **scope** of MLOps Stack Canvas:

- Planning the cost of the infrastructure components for the MLOps stack by considering three main areas:
  - Data and code management
  - ML model management, and
  - Metadata management
- Planning the cost of the orchestration of the ML system to manage its life-cycle and maintainability by considering
  - Continuous integration/training/delivery for ML assets
  - Monitoring to ensure the ML is achieving the business objectives
  - Alerting to deal with model failures.
- Designing the ML system to fulfill:
  - Reproducibility: versioning, feature store, and pipelines,
  - Reliability: models should have few outages and safe failovers,
  - Efficiency: model predictions are fast and as cost-effective as possible.

MLOps Stack Canvas

Generally, the MLOps Stack Canvas consists of three main areas: Data and Code Management, Model Management, and Metadata Management. Each of these areas contains its own building blocks.



1. Value Proposition

Generally, a value proposition is a statement that outlines **why a customer would benefit** from using our software or service.
An ML system generates predictions that are a foundation for a later decision to increase productivity or improve user experience.

The value proposition section summarizes the key things that make up your software product and why end-users should use it.
The goal is to focus on solving the **real problem**.

Generally, to formulate the value proposition for the ML project:

- What are we trying to do for the end-user(s)?
- What is the problem?
- Why is this an important problem?
- Who is our persona? (ML Engineer, Data Scientist, Operation/Business user)

Recommended applying the Machine Learning Canvas through the initial phase of the CRISP-ML(Q) process.

Another practical tool to specify the value proposition is Value Proposition Canvas;
https://www.peterjthomson.com/2013/11/value-proposition-canvas/

---

2. Data Source and Data Versioning

The overall goal is to **estimate the cost** of **data acquisition, storage, and processing**, which are the main activities during the Business and Data Understanding and Data Preparation phases of the CRISP-ML(Q).

In addition, data versioning might be a regulatory requirement to explain predictions for every version of the learned model.
Generally, we distinguish three levels of data versioning:

1. Data is versioned as a **snapshot at training time**;
2. Versioning data and code as **one asset**;
3. Using **specialized data versioning systems**.

Look: https://www.dropbox.com/s/3l9cp75esgvqpxp/Chapter3.pdf?dl=0

The following considerations:

- What data sources are available?
- What is the storage for the above data? (e.g., data lake, DWH)
- Is manual labeling required? Do we have human resources for it?
- How to version data for each trained model?
- What tooling is available for data pipelines/workflows?

Recommend using the Data Landscape Canvas to create an overview of the available, accessible, and required data sources for the ML project.

3. Data Analysis and Experiment Management

Tasks **to translate business objectives to ML tasks, to collect and understand data**, and **to assess the feasibility** of the project. To achieve these tasks, we run experiments and implement a proof of concept.

The focus is the **applicability** of the ML technology for the specified business goals and **data preparation**.

The following questions:

- What programming language to use for analysis? (R, Python, Scala, Julia. Or is SQL sufficient for analysis?)
- Are there any infrastructure requirements for model training?
- What ML-specific and business evaluation metrics need to be computed?
- Reproducibility: What metadata about ML experiments is collected? (data sets, hyperparameters)
- What ML Framework know-how is there?

4. Feature Store and Workflows

Feature engineering is a process of transforming raw input data into feature vectors that are suitable input formats for machine learning algorithms before model training and prediction.

A **feature store** is defined as an interface between data engineering and ML model engineering to separate the feature engineering from the CRISP-ML(Q) model development process. Feature stores promise the speed up in the development and operationalization of ML models.

- Do we have a data governance process such that feature engineering has to be reproducible?
- How are features computed (workflows) during the training and prediction phases?
- What are infrastructure requirements for feature engineering?
- What databases are involved in feature storage?
- Do we design APIs for feature engineering?

5. Foundations (Reflecting DevOps)

Is about the inventory of the available DevOps infrastructure and raising awareness about the current **DevOps principles** in the ML project. The intuition behind this activity is that we might extrapolate **DevOps best practices** to the MLOps activities.

Following the DevOps principles has a direct impact on the software delivery performance.

- How do we maintain the code? What source version control system is used?
- How do we monitor the system performance?
- Do we need versioning for notebooks?
- Deployment and testing automation: What is the CI/CD pipeline for the codebase? What tools are used for it?
- Do we track deployment frequency, lead time for changes, mean time to restore, and change failure rate metrics?

6. CI/CT/CD: ML Pipeline Orchestration

Continuous Integration, Training, and Deployment: ML Pipeline Orchestration.

Generally, **continuous integration** denotes the building, testing, and packaging of data and model pipelines.
Continuous training is a new property, unique to ML systems concerned with automatically retraining ML models.

We utilize the pipeline pattern, a software design pattern that implements the construction and execution of a sequence of operations. Typically, data and ML pipelines include operations such as data verification, feature and data selection, data cleaning, feature engineering, and model training. The set of acyclic pipelines construct a directed acyclic graph (DAG) and denote the overall workflow job.

Depending on the maturity level, we might want to automate the data and model training pipeline workflows to operationalize the model.

- How often are models expected to be retrained? What is the trigger for it (scheduled, event-based, or ad hoc)?
- Where does this happen (locally or on a cloud)?
- What is the formalized workflow for an ML pipeline? (e.g., Data prep -> model training -> model eval & validation) What tech stack is used?
- Is distributed model training required? Do we have an infrastructure for the distributed training?
- What is the workflow for the CI pipeline? What tools are used?

7. Model Registry and Model Versioning

Establishing a model and data versioning practice is the foundation for **reproducibility** in machine learning. Depending on your use case, a change in code or data might trigger a model **re-training**.

The common reason for the machine learning model update is the “**model decay**”, where the model performance declines with time as new data arrives.
All ML models should be versioned and protocolled in regulated industries such as health, finance, or military.

By tracking several versions of the ML model, it is possible to implement different deployment strategies such as **“canary”- or “shadow”-deployment** by analyzing the latest trained model’s performance improvement.

- The reproducibility requirement might be the reason that you need the model versioning.
- Where should new ML models be stored and tracked?
- What versioning standards are used? (e.g., semantic versioning)

8. Model Deployment

Deploying an ML model denotes making it available on the target environment for receiving prediction requests.

Continuous Deployment (CD) is an automatic deployment of ML models into the target environment based on predetermined evaluation metrics.specify all model exposure strategies and infrastructure aspects of CD.

- What is the delivery format for the model?
- What is the expected time for changes? (Time from commit to production)
- What is the target environment to serve predictions?
- What is your model release policy? Is A/B testing or multi-armed bandits testing required? (e.g., for measuring the effectiveness of the new model on business metrics and deciding what model should be promoted into the production environment)
- What is your deployment strategy? (e.g. shadow/canary deployment required?)

9. Prediction Serving

Deals with the ML model serving as a process of applying a machine learning model **to the new input data**.

Generally, we distinguish between **online and batch modes of prediction serving**. They can be implemented using five patterns to put the ML model in production: Model-as-Service, Model-as-Dependency, Precompute, Model-on-Demand, and Hybrid-Serving. Each of these patterns would require different **infrastructure settings**.

To identify the environment for model serving:

- What is the serving mode? (batch or online)
- Is distributed model serving required?
- Is multi-model prediction serving required?
    - See: https://www.oreilly.com/content/efficient-machine-learning-inference/
- What is the expected target volume of predictions per month or hours?

10. ML Model, Data, and System Monitoring

Once the ML Model is deployed, it must be constantly monitored to ensure the model quality, meaning that the model serving produces correct results.

- Do you need to monitor your model for performance degradation and trigger an alert if your model starts performing badly? Is the model retraining based on events such as data or concept drift?
- What ML metrics are collected?
- What domain-specific metrics are collected?
- How is the model performance decay detected? (Data Monitoring)
- What operational aspects need to be monitored? (e.g., model prediction latency, CPU/RAM usage)
- What is the alerting strategy? (thresholds)
- What triggers the model re-training? (ad hoc, event-based, or scheduled)

3 Dilemmas of MLOps

There are also organizational aspects of MLOps, which belong to the general discussion about building the ML projects' infrastructure.

1. **Tooling**: Should we buy, use existing open-source or build in-house tools for any of the MLOps components? What are the risks, trade-offs, and impacts of each of the decisions?
2. **Platforms**: Should we agree on one MLOps platform or create a hybrid solution? What are the risks, trade-offs, and impacts of each of the decisions?
3. **Skills**: How expensive is it to either acquire or educate our own machine learning engineering talents?

Business questions

WORK FLOW DECOMPOSITION

Each task of the entire business process needs to be decomposed into its constituent elements in order to see where prediction (ML model) can be introduced.

Tools: the AI Canvas or the Machine Learning Canvas

AI CANVAS OR MACHINE LEARNING CANVAS

This tool assist and help to structure the breakdown process. They also help to articulate exactly what is needed to predict and how we react on errors made by the prediction algorithm.

AI Canvas

MLOps • 2022

AI Canvas

The AI Canvas is a simple decision-making tool.
Each space on the canvas contains one of the requirements for machine-assisted decision making, beginning with a prediction.
The AI Canvas is a simple tool that helps you organize what you need to know into seven categories in order to systematically make that assessment.

Clarifying these seven factors for each critical decision throughout your organization will help you get started on identifying opportunities for AIs to either reduce costs or enhance performance.
Filling out the AI Canvas won't tell you whether you should make your own AI or buy one from a vendor, but it will help you clarify what the AI will contribute (the prediction), how it will interface with humans (judgment), how it will be used to influence decisions (action), how you will measure success (outcome), and the types of data that will be required to train, operate, and improve the AI.

PREDICTION	JUDGMENT	ACTION	OUTCOME
What do you need to know to make the decision?	How do you value different outcomes and errors?	What are you trying to do?	What are your metrics for task success?

INPUT	TRAINING	FEEDBACK
What data do you need to run the predictive algorithm?	What data do you need to train the predictive algorithm?	How can you use the outcomes to improve the algorithm?

Seven Factors

The top row of the canvas — prediction, judgment, action, and outcome — describes the critical aspects of a decision.

Prediction
You specify what you are trying to predict.

Judgment
you need to know the cost of a false prediction. This will depend on the situation and requires human judgment. There are many factors to consider; determining their relative weights requires judgment.
Such judgment can change the nature of the prediction machine you deploy.
Judgment sometimes requires determining the relative value of factors that are difficult to quantify and thus compare.

Action
you identify the action that is dependent on the predictions generated. This may be a simple “dispatch/don’t dispatch” decision, or it may be more nuanced. Perhaps the options for action include not just dispatching someone but also enabling immediate remote monitoring.

Outcome
An action leads to an outcome. We are able to see for each decision whether the right response occurred. Knowing this is important for evaluating whether there is scope to improve predictions over time.

Seven Factors

On the bottom row of the canvas are three final considerations. They all relate to data.

Input
You need to know what is going on at the time a decision needs to be made.

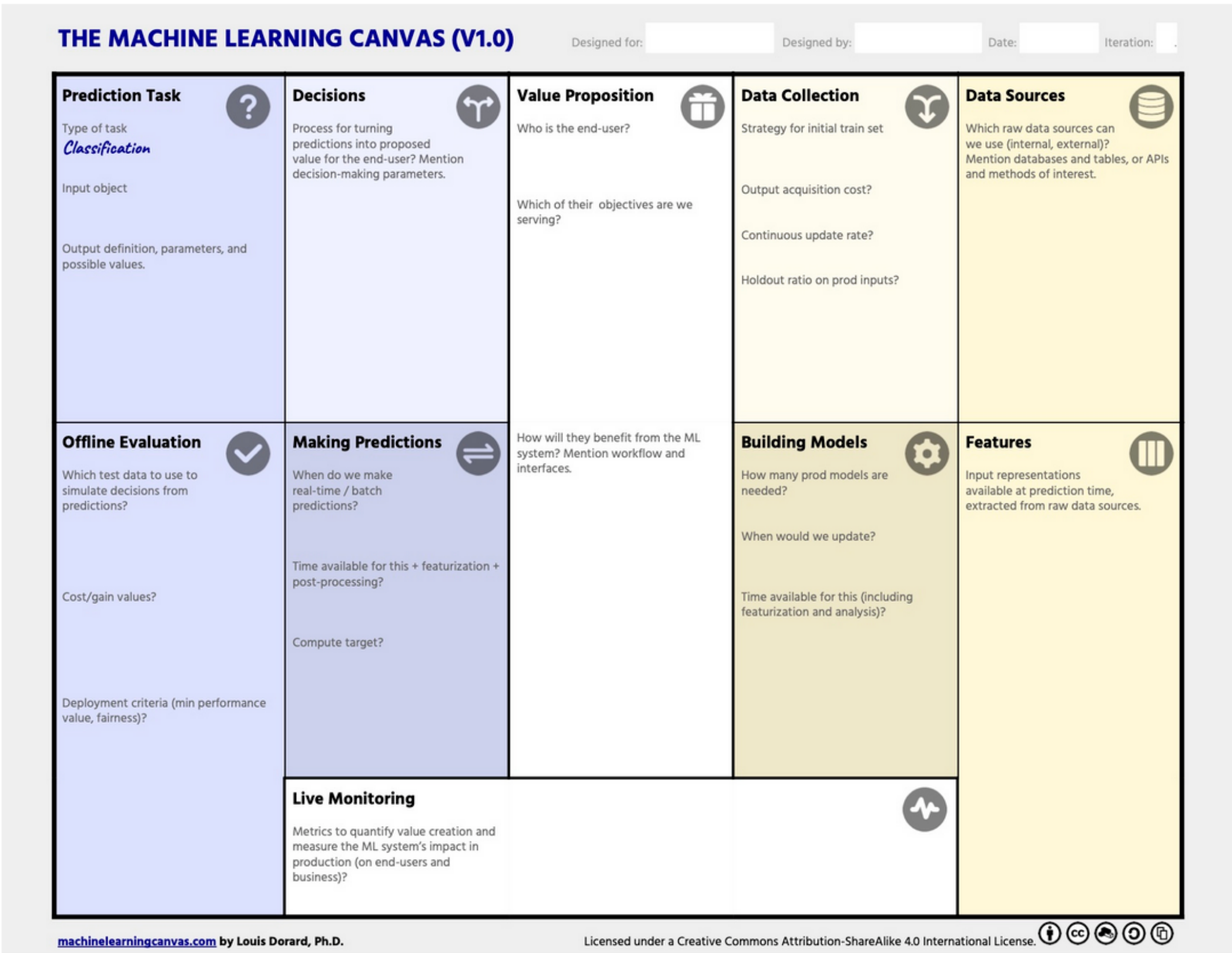
Training
To develop the prediction machine in the first place, you need to train a machine learning model. The richer and more varied that training data, the better your predictions will be out of the gate. If that data is not available, then you might have to deploy a mediocre prediction machine and wait for it to improve over time.

Feedback
Feedback data is data that you collect when the prediction machine is operating in real situations. Feedback data is often generated from a richer set of environments than training data.
So, you can improve the accuracy of predictions further with continual training using feedback data.

Machine Learning Canvas

MLOps • 2022

Machine Learning Canvas



The ML Canvas guides through the prediction and learning phases of the ML application. In addition, it enables all stakeholders to specify data availability, regulatory constraints, and application requirements such as robustness, scalability, explainability, and resource demand.

What is

This canvas structures the ML project and helps to specify the core requirements to realise the project.
It is used for complete the AI canvas when it ask to specify both the vision for the ML system and the specifics of the system.
The effort to fill out this canvas might initiate an existential discussion regarding the real objective and hidden costs for the ML-software.

STRUCTURE

Initially, we identify the objectiven, the **business goal**.
After there is the **Value Proposition** building block, which describes products or services that create some value for customers.

The remaining canvas is divided into three broad categories:

- **Learning**. specify how the ML model will be learned.
- **Prediction**. describes how the prediction is performed.
- **Evaluation**. contains methods and metrics for the ML model and the system evaluation.

10 BLOCKS

The Machine Learning Canvas is structured as ten compound blocks, each of those blocks is focused on one aspect of the future ML application: Value Proposition, Data Sources, Prediction Task, Features (Engineering), Offline Evaluation, Decisions, Making Predictions, Collecting Data, Building Models, and Live Evaluation and Monitoring.

1. Value Proposition

THREE IMPORTANT QUESTIONS

1. What is the problem? What objective are we serving? What are we trying to do for the end-user?
2. Why is it important?
3. Who is the end-user? Can we specify the persona?

GEOFFREY MOORE'S VALUE POSITIONING STATEMENT

Geoffrey Moore worked as a consultant for The McKenna Group and has accumulated a wide variety of knowledge about companies of Silicon valley.
He offers a template for a positioning statement, that we can complete for this canvas' block.

The template is:
For (target customer) who (statement of the need or opportunity), the (product name) is a (product category) that (statement of key benefit - that is, compelling reason to buy). Unlike (primary competitive alternative), our product (statement of primary differentiation).

2. Data Sources

In this block, we clarify all available and possible data sources to be used for the ML task. Furthermore, we should clarify the hidden costs of a machine learning application.

As an example, we might consider using:

- **Internal/external databases.**
- Data marts, OLAP cubes, data warehouses, OLTP systems.
- **Hadoop clusters.**
    - is a special type of computational cluster designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment.
- **REST APIs to gather data.**
- **Static files, spreadsheets.**
- **Web scraping.**
    - is a specialized tool designed to accurately and quickly extract data from a web page. This information is collected and then exported into a format that is more useful for the user.
- **The output of other (ML) systems.**
- **Open-source data sets.**
    - Useful publicly available datasets: Kaggle Datasets, Google's Dataset Search, UCI Repository, or Wikipedia's list of datasets for machine-learning research

2. Data Sources

DATA WAREHOUSES

A data warehouse is a system that **aggregates data from multiple sources into a single, central, consistent data store** to support data mining, artificial intelligence (AI), and machine learning.
Data warehouse solutions consolidate data from the different sources to make it available in one unified form.

DATA MARTS

Data marts is a system that aggregates data from multiple sources **focused on a particular line of business, department, or subject area**.
Data marts make specific data available to a defined group of users, which allows those users to quickly access critical insights without wasting time searching through an entire data warehouse.

2. Data Sources

OLAP CUBES

The OLAP (online analytical processing) cube is an **array-based multidimensional database** that makes it possible to process and analyze multiple data dimensions much more quickly and efficiently than a traditional relational database. The OLAP cube extends the single table with additional layers, each adding additional dimensions —usually the next level in the “concept hierarchy” of the dimension.

OLTP SYSTEMS

OLTP (online transaction processing) enables the **real-time execution of large numbers of database transactions** by large numbers of people, typically over the internet. A database transaction is a change, insertion, deletion, or query of data in a database.
OLTP systems (and the database transactions they enable) drive many of the financial transactions we make every day.

3. Prediction Task

This block is focuse what type of ML should be used.

Examples, for clarify the ML Task:

- Supervised or unsupervised learning?
- Is the problem about which option should be taken? (recommendation)
- Do we need to predict a continuous value? (regression)
- Which category need to be predicted? (classification)
- Do we need to group our data? (clustering)
- What is the input for a prediction task?
- What is the output of the prediction task?

4. Features (Engineering)

As every ML algorithm requires input data in the form of features, we should clarify how should the input data be represented.

Consider to include domain experts to specify what data aspects are most important for the particular ML task.

5. Offline Evaluation

We would need to specify and set up the methods and metrics to evaluate the system before deployment. Here we would need to specify:

- Domain specific metrics that justify the deployment of the ML model.
- What technical evaluation metrics should be used? Precision, Recall, F1 score.
    - **Precision**: A measurement that specifies what fraction of the machine learning model's output was accurate when compared to the human annotator output. Precision is determined by the number of correctly labeled annotations divided by the total number of annotations added by the machine learning model.
    - **Recall**: A measurement that specifies how many mentions that should have been annotated by a given label were actually annotated with that label. Recall is determined by the number of correctly labeled annotations divided by the number of annotations that should have been created.
    - **F1 score:** A measurement that considers both precision and recall to compute the score. The F1 score can be interpreted as a weighted average of the precision and recall values, where an F1 score reaches its best value at 1 and worst value at 0.
- What is our test data?
- How much test data do we need?

6. Decisions

The next is to specify:

- How are prediction used to make decisions?
- How does the end-user or the system interacts with the model predictions?
- Are there hidden costs in decision making, such as **human in the loop**?

Such information is required to later decide on how to deploy the ML model.

HUMAN-IN-THE-LOOP MACHINE LEARNING (HITL)

human-in-the-loop machine learning refers to the stages of the model development process that **require a person to inspect, validate, or change some part of the process** to train and deploy a model into production.

HITL also can extend to the people who prepare and structure data for machine learning.

In fact, human-in-the-loop machine learning includes all of the people who work with data that will be used, including the people who collect, label, and conduct quality control (QC) on data.

7. Making Predictions

This block includes information about when we make a prediction on new inputs.

- When should predictions be available?
  - New predictions are made each time when the user opens the app, such as recommendations.
  - New predictions are made on request.
  - New predictions are made on schedule.
- Are predictions made on the fly for each data point or for a batch of the input data?
- How computationally complex could the model inference get in the application?
- Is there a human in the loop to support in making predictions?

8. Collecting Data

The Collecting Data block gathers information about new data that should be collected in order **to re-train the ML model**. In this way, we specify how we prevent the ML model decay phenomenon.
Further questions to answer in this block are:

- How do we label the new data?
- How expensive is it to collect new data?
- How expensive is it to process rich media like images, sound, or video?
- Is there human in the loop for the manual cleaning and labelling of the incoming data?

9. Building Models

The Building Models answers questions regarding updating the ML models, because different ML tasks require different frequencies of model **re-training**:

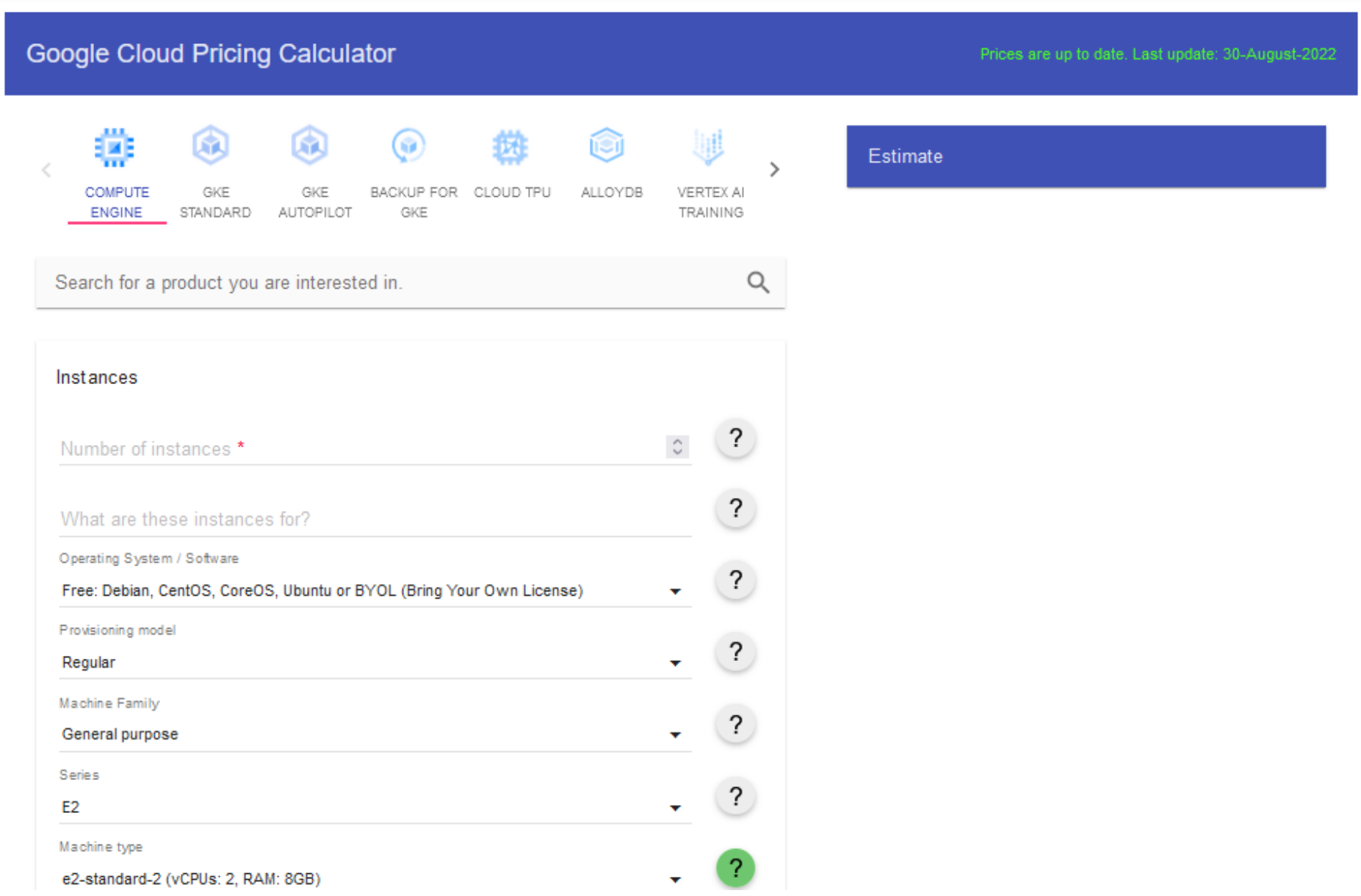
- How often the model should be retrained?
- What are the hidden costs for model re-training?
  - what is the price policy of the cloud vendor?
  - how should we perform hardware cost estimation?
  - common Cloud Pricing Calculators are Google Cloud Calculator, Amazon ML Pricing, Microsoft Azure Calculator
- How long will it take to re-train the model?

9. Building Models

GOOGLE CLOUD CALCULATOR

With the Google Cloud Pricing Calculator tool, developers will be able to **calculate in advance what it will cost to deploy an application** on Google's cloud platform.

The Google Cloud Calculator provides an accurate breakdown of the services and the costs they entail. It should be noted, however, that the prices issued by the calculator are not to be seen as final and only serve as a basis for discussion and negotiation.



10. Live Evaluation and Monitoring

The ML model should be evaluated and here we would need to specify both model and business metrics, which should correlate.

Generally, the metrics should follow the S.M.A.R.T methodology.

- How do we track the system performance? e.g. A/B Testing.

S.M.A.R.T METHODOLOGY

SMART goals are meant to address all of your major job responsibilities.
Goals are intended to focus attention and resources on what is most important so that you can be successful in achieving your priorities.

S - Specific
When setting a goal, be specific about what you want to accomplish.
Think about this as the mission statement for your goal.
It should include an answer to the popular ‘w’ questions:

- Who needs to be involved to achieve the goal.
- What you are trying to accomplish.
- When, at least set a time frame.
- Which any related obstacles or requirements.
- Why, the reason for the goal.

10. Live Evaluation and Monitoring

S.M.A.R.T METHODOLOGY

M – Measurable

What metrics are you going to use to determine if you meet the goal? This provides a way to measure progress.

- The M is a direct indicator of what success for a particular goal will look like.
- Data collection efforts needed to measure a goal can be included in that goal's action plan.
- Measurement methods can be both quantitative (productivity results, money saved or earned, etc.) and qualitative (client testimonials, surveys, etc.).

A – Achievable

This focuses on how important a goal is to you and what you can do to make it attainable and may require developing new skills and changing attitudes. The goal is meant to inspire motivation, not discouragement.

R – Relevant

Relevance refers focusing on something that makes sense with the broader business goals. For example, if the goal is to launch a new program or service, it should be something that's in alignment with the overall business/department objectives.

10. Live Evaluation and Monitoring

S.M.A.R.T METHODOLOGY

T – Time-Bound

Anyone can set goals, but if it lacks realistic timing, chances are you're not going to succeed.

Providing a target date for deliverables is imperative. Ask specific questions about the goal deadline and what can be accomplished within that time period.

Providing time constraints also creates a sense of urgency.

MLOps Architecture

DOCUMENTING

One of the effective ways to document the MLOps architecture is by using Architecture Decision Records (ARD) construct.

Architecture Decision Records

MLOps • 2022

Architecture Decision

An **Architectural Decision (AD)** is a software design choice that addresses a functional or non-functional requirement that is architecturally significant.

An **Architecturally Significant Requirement (ASR)** is a requirement that has a measurable effect on a software system's architecture and quality.

An **Architectural Decision Record (ADR)** captures a single AD, such as often done when writing personal notes or meeting minutes; the collection of ADRs created and maintained in a project constitute its decision log.
All these are within the topic of **Architectural Knowledge Management (AKM)**.

ADR

ARD is a brief description of the architectural decision.

- (An architecture decision record is a short text file in a format similar to an Alexandrian pattern.)

Use a lightweight text formatting language like Markdown or Textile.

ADRs will be numbered sequentially and monotonically. Numbers will not be reused.

Architectural Decision Records

“LIGHTWEIGHT” ADR

A “lightweight” ADR consists of:

- Title
  - These documents have names that are short noun phrases.
- Status
  - What is the status, such as proposed, accepted, rejected, deprecated, superseded, etc.?
- Context
  - A short description of the problem.
  - This section describes the forces at play, including technological, political, social, and project local. These forces are probably in tension, and should be called out as such.
  - What is the issue that we're seeing that is motivating this decision or change?
- Decision
  - Describe the actual architectural decision with a detailed explanation.
  - What is the change that we're proposing and/or doing?
- Consequences
  - This section provides any implications of the architecture decision. This section is also a good place to discuss any architectural trade-offs.
  - A particular decision may have positive, negative, and neutral consequences, but all of them affect the team and project in the future.
  - What becomes easier or more difficult to do because of this change?

Tools

Y-STATEMENT

To reduce design documentation effort, we experimented with lean, minimalistic documentation.

One approach was our **(WH)Y approach**, that reduces the documentation to a statement in the **Y-statement**.

In short, the Y-statement is as follows:

> In the context of `<use case/user story u>`, facing `<concern c>` we decided for `<option o>` to achieve `<quality q>`, accepting `<downside d>`.

The long form of it is as follows (extra section “because”):

> In the context of `<use case/user story u>`, facing `<concern c>` we decided for `<option o>` and neglected `<other options>`, to achieve `<system qualities/desired consequences>`, accepting `<downside d/undesired consequences>`, because `<additional rationale>`.

Tools

MADR

MADR: The Markdown Architecture Decision Records.

MADR is a lean template to capture **any decisions in a structured way**. The template originated from capturing architectural decisions and developed to a template allowing to capture any decisions taken.

There are debates what is an architecturally-significant decision and which decisions are not architecturally significant. Since we believe that any (important) decision should be captured in a structured way, we offer the MADR template to capture any decision.

```
# Use Plain JUnit5 for advanced test assertions


## Context and Problem Statement


How to write readable test assertions?

How to write readable test assertions for advanced tests?


## Considered Options


* Plain JUnit5

* Hamcrest

* AssertJ


## Decision Outcome


Chosen option: "Plain JUnit5", because it is a standard framework and the features of the other frameworks
```

Data management

TOOLS FROM DATA MESH

- Data Product Canvas
- dbt and Snowflake
- Google Cloud BigQuery
- Google Dataplex

MLOps • 2022

Data Product Canvas

MLOps • 2022

Designing Data Products

A MESH

Conceptually, a mesh is a graph, a network, consisting of **nodes** and **connecting edges**. Each node in a data mesh is called **data product**. Every data product exists within a **bounded context** and one bounded context might contain several data products.

SOCIOTECHNICAL PERSPECTIVE

Furthermore, because we have consumers for our data, these consumers expect a certain quality. Therefore, our data need to be accessible, and everyone should know where to find data.

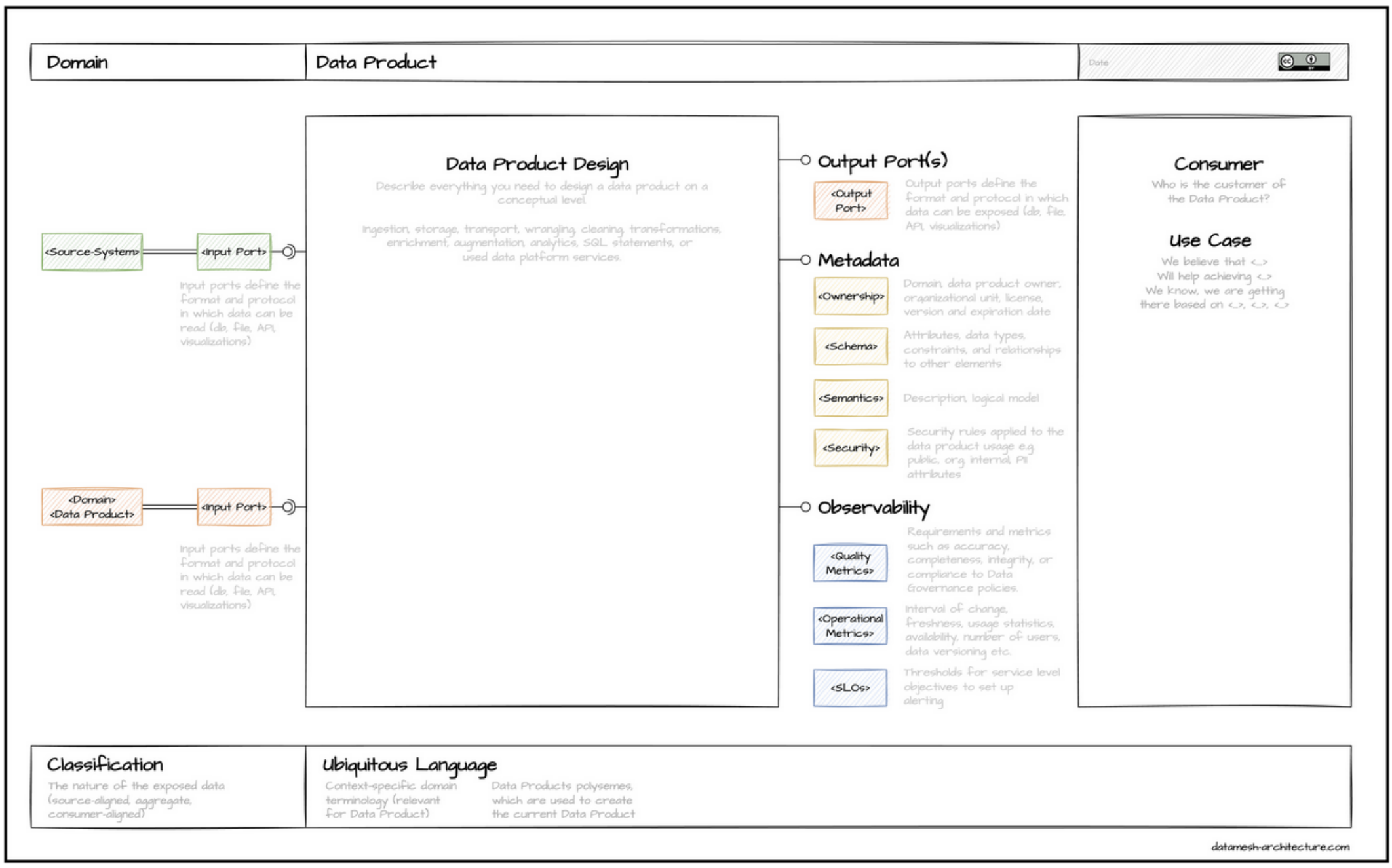
A DATA PRODUCT

A data product is an autonomous, read-optimized, standardized data unit containing at least one Domain Dataset, created for satisfying user needs.

Any data representation that has value for its consumers can be a data product. For example:

- A database table or view
- Raw unstructured files: e.g., images or videos
- Data stream of data entities from a transaction system
- Data stream representing the history of changes to the application
- REST API: Read-optimized data exposed from applications
- Features for the machine learning models

Data Product Canvas



A Data Product Canvas is a visual framework that guides your team through the data product specification.

The Data Product Canvas consists of ten building blocks:

1. Domain
2. Data Product Name
3. Consumer and Use Case
4. Output Port
5. Metadata
6. Input Ports
7. Data Product Design
8. Observability
9. Ubiquitous Language
10. Classification

Ten building Blocks

1. DOMAIN

Each data product should be implemented, evolved, and maintained by one domain team only.
The following possible questions are relevant in this building block:

- Who is accountable for the data product?
- Who specifies its requirements?
- Who will answer questions about the data product?
- Who fixes it when it breaks?

2. DATA PRODUCT NAME

Each data product has a unique name to be identified and accessed within an organization.

3. CONSUMER AND USE CASE(S)

This building block describes the reason behind the existence of the data product. To identify the purpose of the data product, we describe analytical use cases and organizational objectives. Understanding the use cases is essential to specify the data required to implement the use cases.
The consumer of the future data product might be either our own domain team or different domain teams.

Ten building Blocks

4. OUTPUT PORTS

The output ports define the format and consumption protocol in which data can be exposed.
For example, the output port can be a database table, file, API, or visualizations.

5. METADATA

The data product should define its metadata.

- The ownership part describes domain name, product owner, organizational unit, license, version and probable expiration date of the data product.
- The data schema part describes attributes, data types, constraints, and relationships to other elements in the data unit.
- The semantics part provides description about the logical model of the data unit.
- The security block describes security rules applied to the data product usage.

6. INPUT PORTS

This building block describes the input data for the future data product. The input ports are receiving mechanism for data that will constitute the data product. The input ports define the format and protocol in which data can be read.
We distinguish here between operational source systems and other data products, which might be either internal or coming from other domains.

Ten building Blocks

7. DATA PRODUCT DESIGN

To specify everything between the input and output ports.
For instance, you might describe data ingestion, storage, transport, wrangling, cleaning, transformations, enrichment, augmentation, analytics, SQL statements, or data platform services.

8. OBSERVABILITY

- Quality Metrics outlines data quality requirements and metrics such as accuracy, completeness, integrity, as well as compliance to data governance policies.
- Operational Metrics might include interval of change, freshness, usage statistics, availability, number of users, data versioning etc.
- SLOs for data product enable the discipline of building trustworthiness in each data product. We specify here the thresholds for metrics to trigger alarms.

9. UBIQUITOUS LANGUAGE

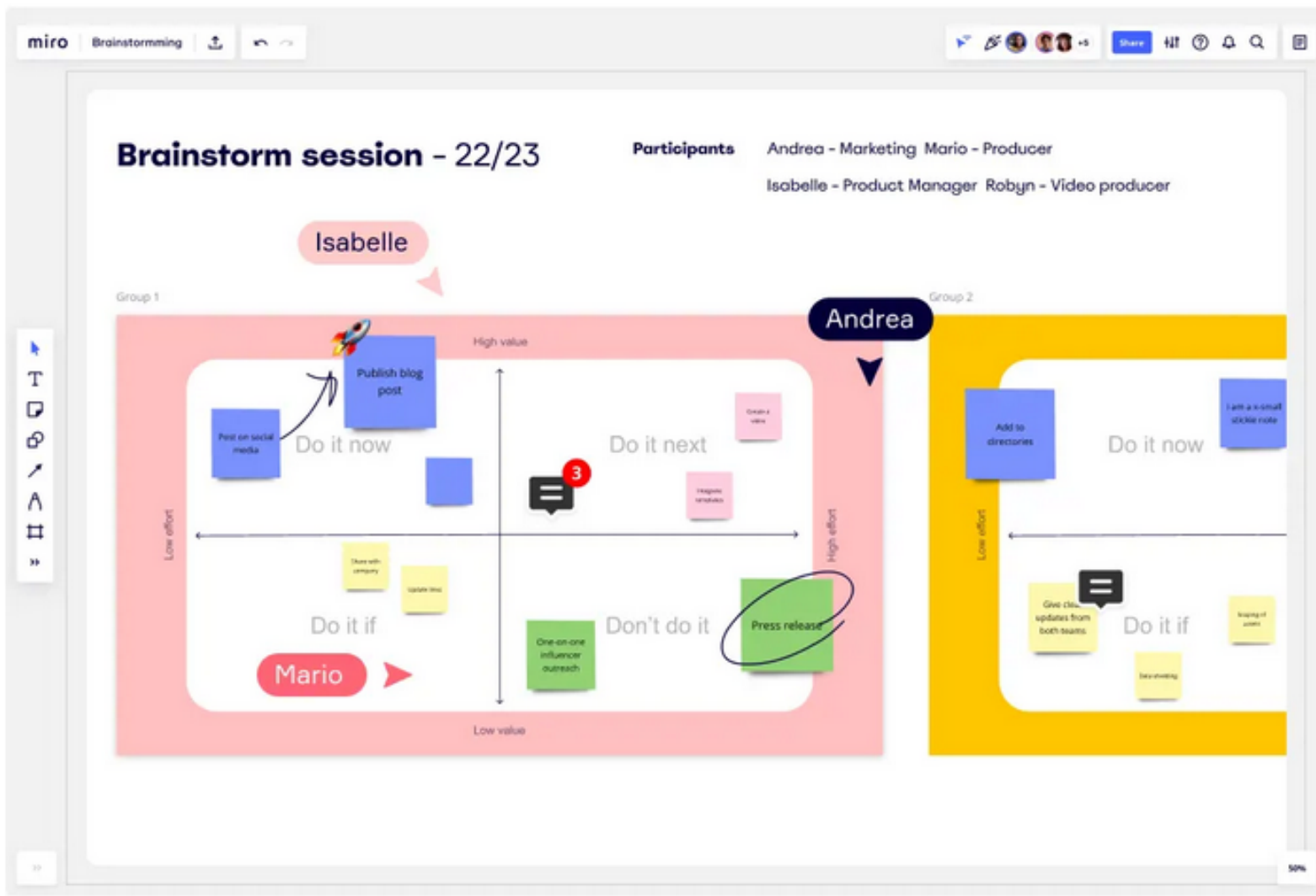
Describe here a common language that is shared between everyone involved in the project. This is usually a context-specific domain terminology that is relevant for operational systems and data product.

10. CLASSIFICATION

We specify the nature of the exposed data, we classify our data product as either, source-aligned, aggregate, or consumer-aligned.

Templates

The proposed canvas is suitable for working collaboratively on data products design.
The canvas might be easily used either plotted on paper or with the **common online whiteboards** like Miro or Mural.



Once your organization has created a collection of data products using the proposed data product canvas, you can start connecting all data products and produce an actual mesh.

dbt and Snowflake

MLOps • 2022

dbt and Snowflake

DBT

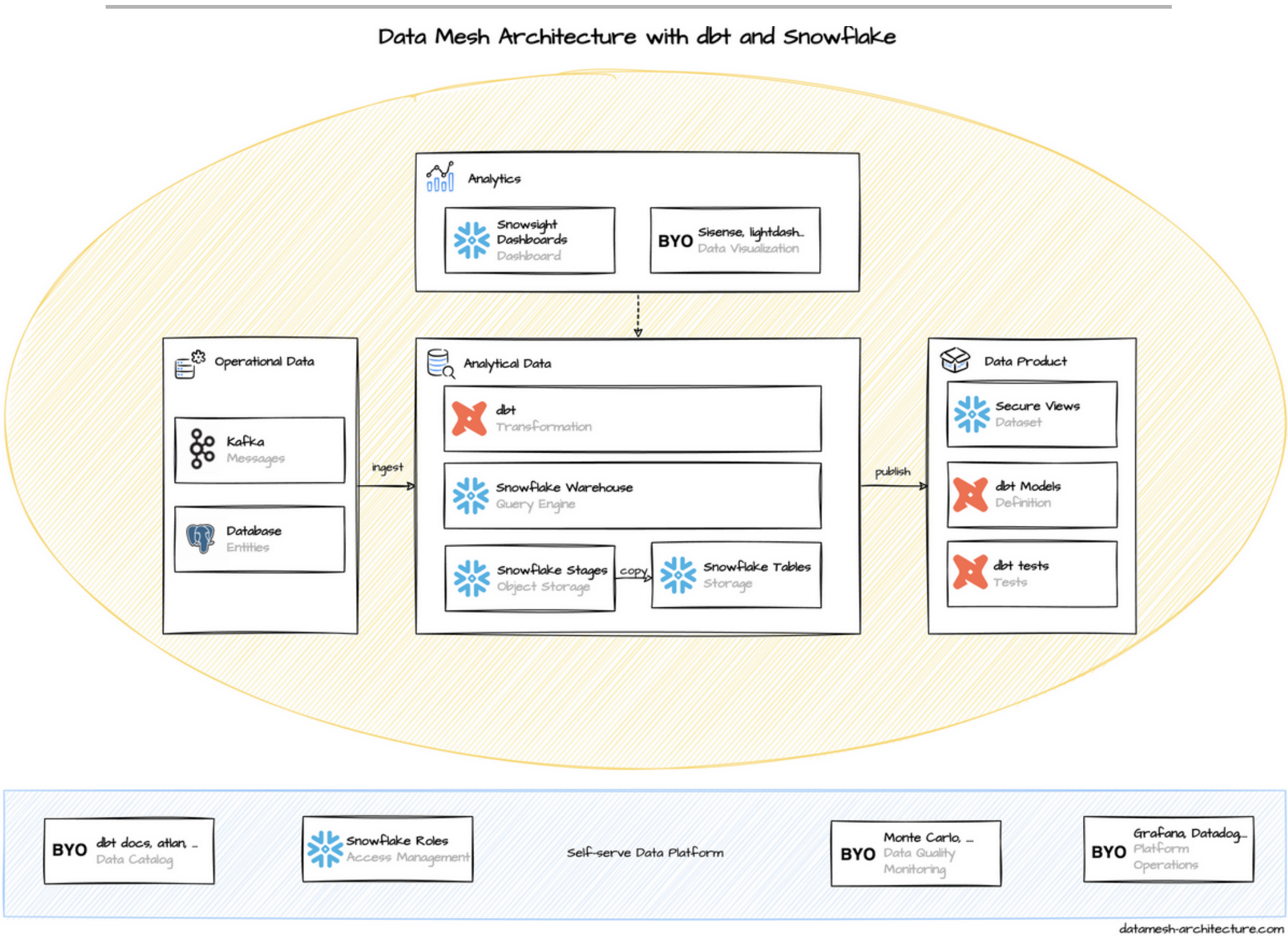
dbt has emerged as the default **framework to engineer analytical data**. This is where you define and test your **models**.
Compare it with Spring Boot in the microservices world.
dbt has adapters for most data warehouses, databases, and query engines.

SNOWFLAKE

Snowflake is a **modern data warehouse**. The key concepts to store and access data are tables and views, along with SQL queries.
Snowflake can be compared with Google BigQuery, AWS Redshift and Azure Synapse.
Snowflake is a great fit for dbt.
Snowflake is built on cost-efficient object storage services to **store data** and compute instances to **execute queries and DML operations**.

- DML(Data Manipulation Language) operations: Inserting and Updating, Upserting, Merging, Deleting, Restoring Deleted records and Converting Leads.

Architecture



DBT

- Trasformation
- Definition
- Test

SNOWFLAKE

- Query Engine
- Object storage, Storage
- Dashboard
- Dataset
- Access Management

Analytical Data

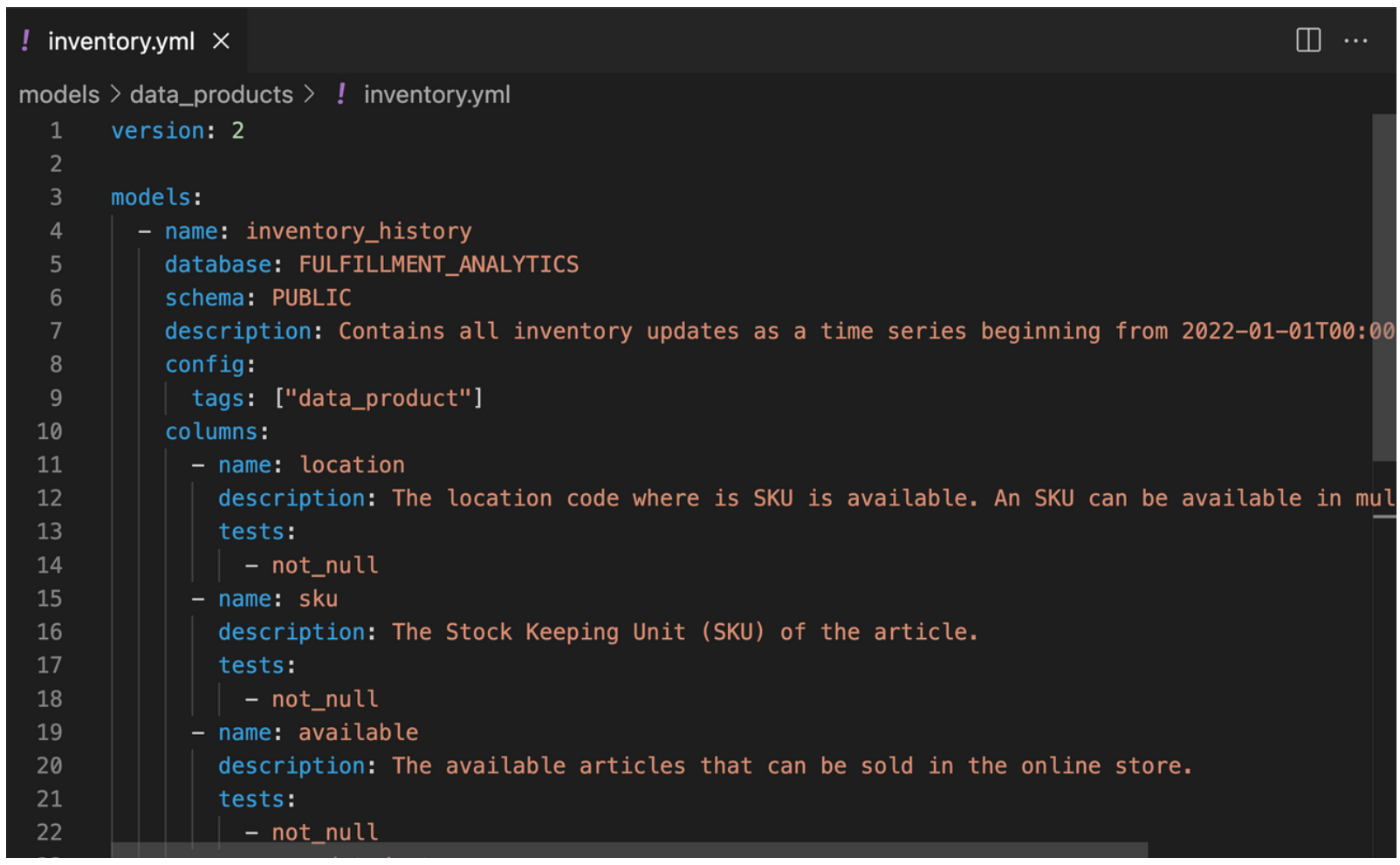
DBT - TRASFORMATION

dbt is a framework to transform, clean, and aggregate data within your data warehouse.

Transformations are written as plain **SQL statements** and result in models that are SQL views, materialized views, or tables.

Instead, models are defined, configured, and documented in simple **YML or YAML structures**.

YAML ("Yet Another Markup Language") is a text format used to specify data related to configuration.



Models can refer to other models, and dbt builds a usage dependency graph before executing the models.
The templating engine Jinja can be used to define these references and also build your own macros, that are reusable functions.

Data Product

DBT - TEST

Also, dbt embraces tests to verify data when running any transformation, both for sources and results. More complex tests are written as **SQL statements** that need to return an empty result for a test to pass.
The extension **dbt-expectations** adds powerful test methods to verify the data quality. dbt tests are great start to have **data quality monitored**.

DBT - DEFINITION

With data mesh, every domain team manages its own dbt project(s) and they own their CI pipelines to run the models.

When publishing models as data products, a model should be tagged as **data_product**, and it is a good idea to put them into a folder like **/models/data_products/**.

You can use **macros** and lifecycle **hooks** to automate policies that you agreed on in the federated governance group, like granting and checking permissions, or hashing columns that are tagged as pii-sensitve data.

Analytical Data

SNOWFLAKE - STORAGE AND TABLES

Snowflake stores data in **tables** that are logically organized in databases and schemas.

Data is usually loaded into Snowflake using stages that are references to object stores (such as S3 buckets), either managed internally through Snowflake or external references. Pipes continuously copy these data into tables.

There are pre-built solutions to ingest Kafka messages, and it is quite common to use a managed service like Fivetran or Stitch to replicate data from databases or SaaS providers, such as Google Analytics or Salesforce.

SNOWFLAKE - WAREHOUSES

dbt transformations, data loads, and all other SQL queries are executed by a warehouse.
Warehouses are compute instances, and the query performance scales with their underlying CPU cores and memory. Warehouses are charged while they are active.

A warehouse can be attached to specific transformations and queries to give a high level of isolation.

Data Production & Data Platform

SNOWFLAKE - DATA ORGANIZATION

In Snowflake, data can be organized by these layers:

- Organization Account
- Account
- Databases (a database is just a logical grouping)
- Schemas (a schema is also just a logical grouping)
- Tables, Views, Sequence, Stages (references to files in an object store), ...

In general:

- one single account per environment for all teams,
- one logical database per team,
- teams can use schemas to internally organize their data.

SNOWFLAKE - ACCESS MANAGEMENT

As **permission management** might get complex, an alternative is that each domain team has their own Snowflake account under the same organization account.

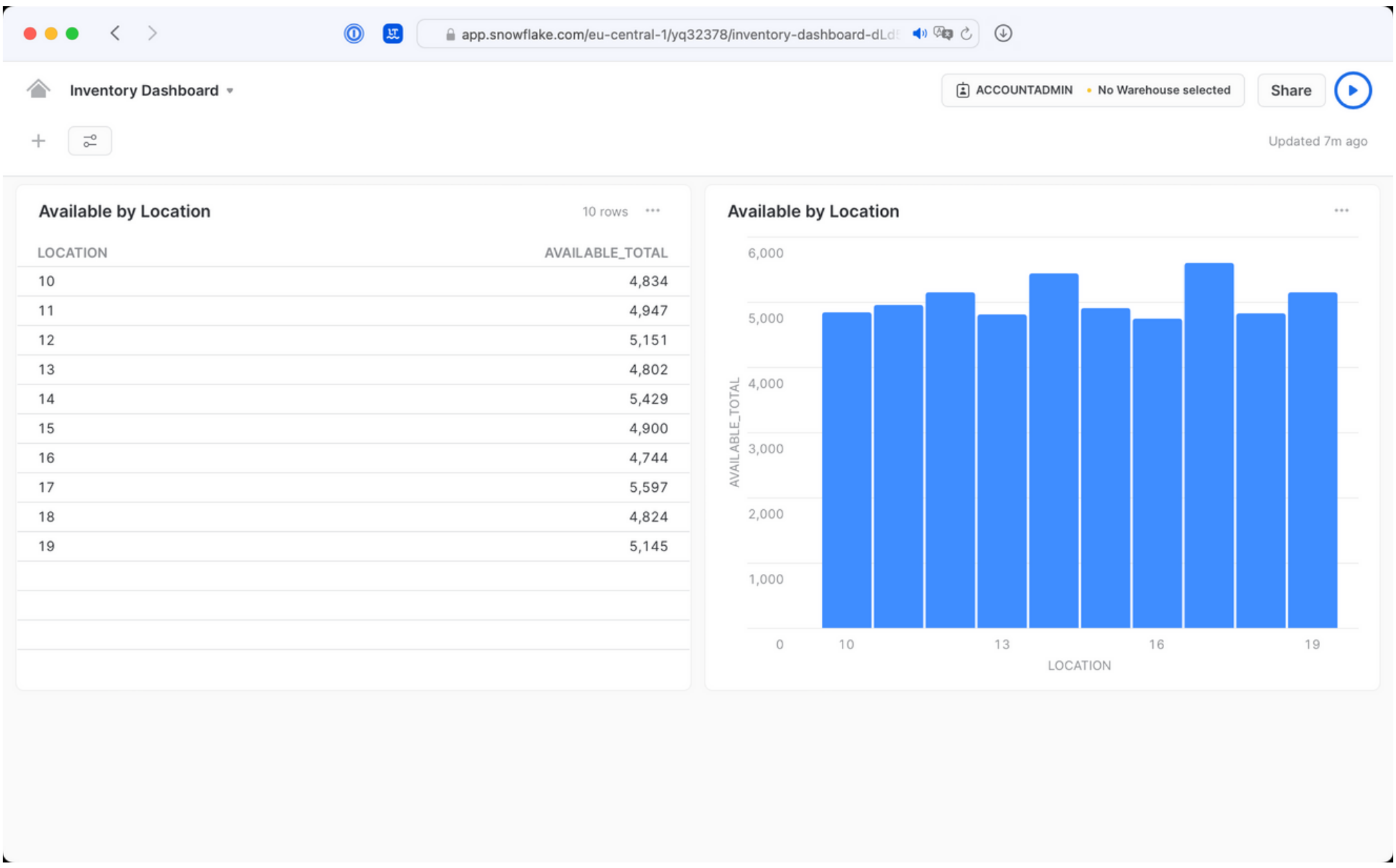
Unfortunately, there is no simple way to share data across the whole organization, so a **dbt hook** or some **terraform configurations** are required. This option enhances the isolation and security, as everything is clearly separated, but it is more management overhead to share and use data products.

Analytics

SNOWFLAKE - DASHBOARDS

Snowflake`s Web UI Snowsight has some basic support to visualize data in **dashboards**.
Again, Snowflake integrates well with more advanced BI tools like Sisense, Tableau, or lightdash.

Business intelligence (BI)
BI is an term for the technology that enables data preparation, data mining, data management, and data visualization. Business intelligence tools and processes allow end users to identify actionable information from raw data, facilitating data-driven decision-making within organizations across various industries.



Data Platform

SNOWFLAKE - NO DATA CATALOG

Snowflake has no built-in **data catalog** to discover and govern published data products.

For a start, the generated documentation of dbt models might be sufficient, when linked on a common wiki page. Most advanced data catalogs, such as atlan, integrated well, both in Snowflake and dbt projects.

SNOWFLAKE - PLATFORM

To monitor the platform itself, Snowflake comes with essential statistics on consumed credits, data storage and transfers. Resource monitors help to notify and suspend when a defined budget exceeds. More detailed usage, access history and especially query execution statistics are accessible through a system database via SQL.

These can be integrated in monitoring tools, such as Grafana or Datadog.

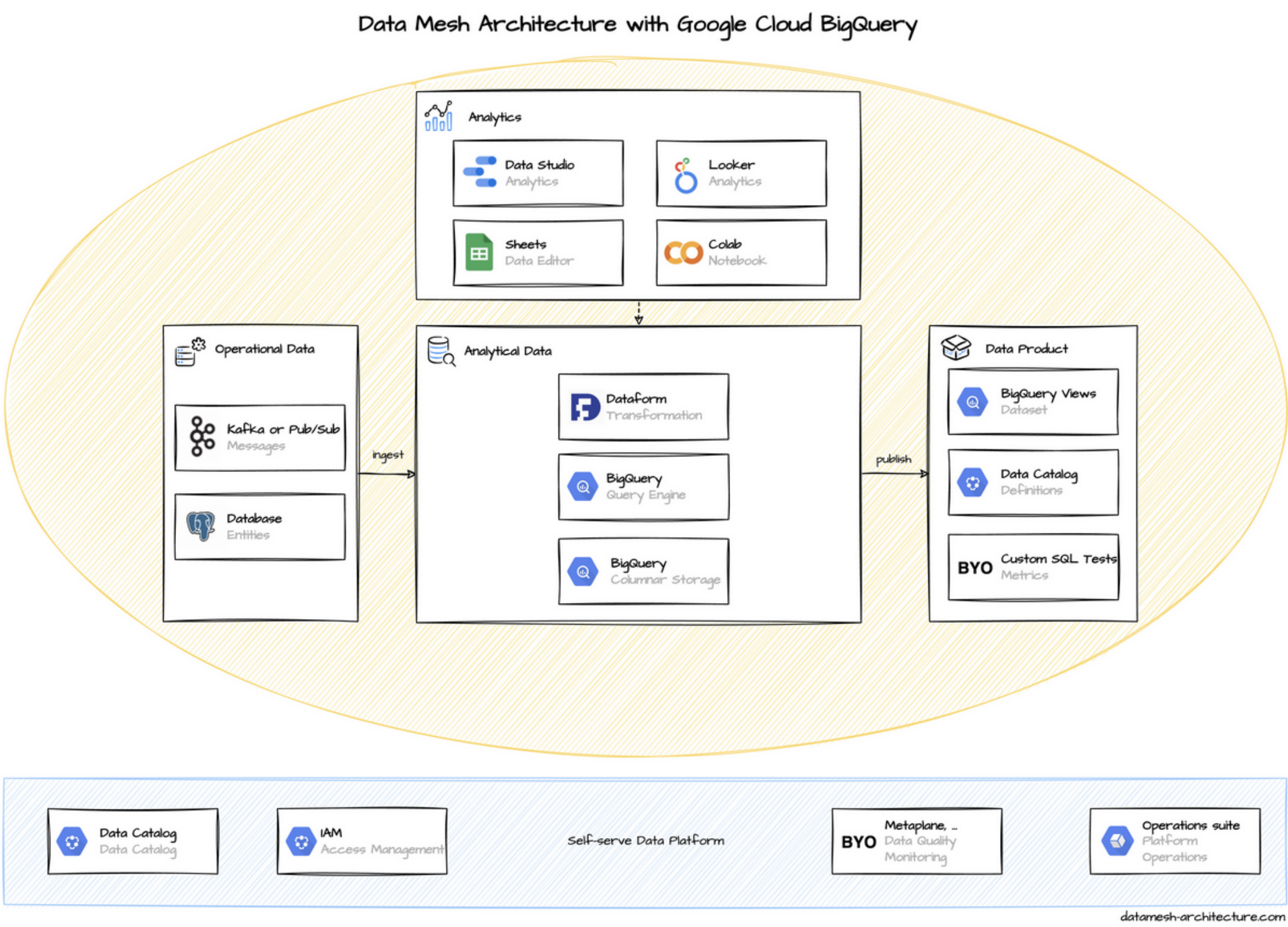
Google Cloud BigQuery

MLOps • 2022

Google Cloud BigQuery

Google Cloud Platform is a **Serverless, highly scalable, and cost-effective multicloud data warehouse** designed for business agility.

Many Data Mesh implementations rely on Google Cloud Platform (GCP) as a common infrastructure, at least for analytical data. Everything is available as developer-friendly self-service. The on-demand query performance of BigQuery is remarkable, especially for large data sets.



BigQuery & Operational Data

BIGQUERY

BigQuery is the central component for **storing analytical data**.
BigQuery is a columnar data store and can perform efficient **JOIN operations** with large data sets.

EXTERNAL AND MANUAL DATA

BigQuery supports access to files stored in **Google Cloud Storage** as external tables , so it can also be used as a data lake.
Manual data is easily edited with Google Sheets.

INGEST

When the operational system architecture relies on Apache Kafka, then streaming through Kafka Connect Google BigQuery Sink Connector is recommended.

CDC DATABASE

CDC Replication is a replication solution that **captures database changes** as they happen and delivers them to target databases.
For CDC database replication to bigquery, consider Data Fusion and Dataflow, based on your operational database.

Analytical Data & Analytics

SQL ON RAW DATA

The simplest way to preprocess and clean raw data is with SQL views. The SQL views can be implemented to perform directly on raw data or triggered as scheduled queries that materialize in a new table. If this does not suffice, more complex pipelines can be implemented with Dataflow (Apache Beam) or Dataform (a dbt clone).

SQL QUERIES

BigQuery supports SQL queries with a number of available user-defined functions, link: https://github.com/GoogleCloudPlatform/bigquery-utils/tree/master/udfs/community

VISUALIZATION

For basic visualization and report generation, Google Data Studio is well-integrated and can be used for free.

OTHER SUPPORTS

A good integration also exists for Google Sheets and Colab, an online Jupyter Notebook service.
More advanced analytics is done with Looker, which was acquired by Google in 2019.

Data product

BIGQUERY VIEWS

Data Products are usually simple **BigQuery tables or views** in a special dataset. The access to these data is managed by the domain team with Google IAM roles.

DEFINITIONS

Google has an integrated Data Catalog, which can get pretty cluttered. It can be used to document and tag datasets as data products, but we see that many users still stick on a Wiki page with available data products listed.

Features

- **Serverless**: you can focus on data and analysis rather than worrying about upgrading, securing, or managing the infrastructure.
- **Built-in ML and AI integrations**: Besides bringing ML to your data with BigQuery ML, integrations with Vertex AI and TensorFlow enable you to train and execute powerful models on structured data in minutes, with just SQL.
- **Real-time analytics**: making your latest business data immediately available for analysis.
- **Real-time change data capture and replication**: Synchronize data across heterogeneous databases, storage systems, and applications reliably and with minimal latency.
- **Automatic high availability**: BigQuery transparently and automatically provides highly durable, replicated storage in multiple locations
- **Standard SQL**: BigQuery supports a standard SQL dialect.
- Materialized Views: allowing you to quickly get answers to your questions.
- **Storage and compute separation**: With BigQuery's separated storage and compute, you have the option to choose the storage and processing solutions.
- **Automatic backup and easy restore**: BigQuery automatically replicates data and keeps a seven-day history of changes.
- **BigQuery data transfer service**: The BigQuery Data Transfer Service automatically transfers data from external data sources, like Google Marketing Platform, Google Ads, YouTube, and partner SaaS applications to BigQuery.

Google Dataplex

MLOps • 2022

Announcement

Google announced Dataplex to centrally manage, monitor and govern data and promotes it for data mesh use cases.

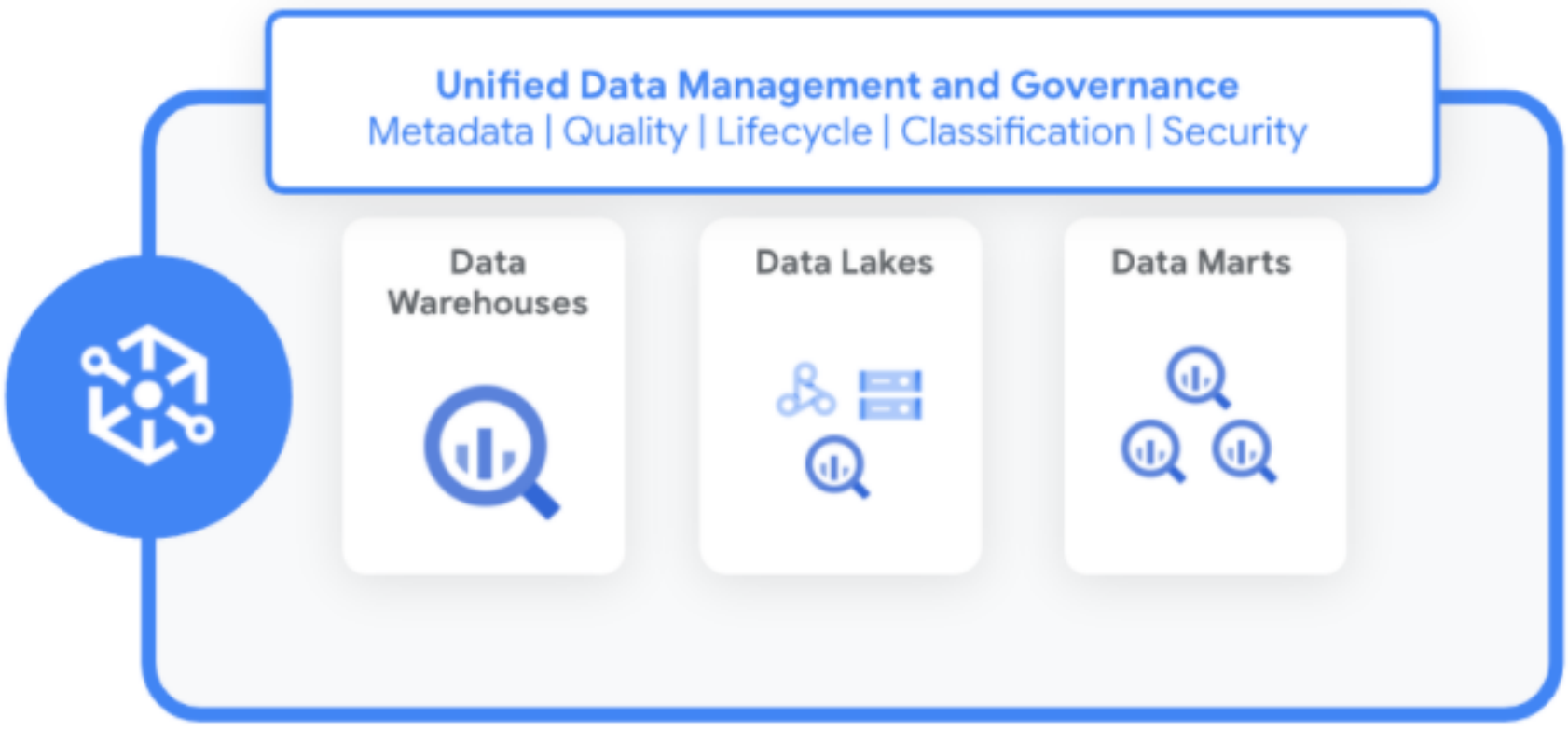
Dataplex is an **intelligent data fabric** that helps you unify distributed data and automate data management and governance across that data to power analytics at scale.

Dataplex enables standardization and unification of metadata, security policies, governance, classification, and data lifecycle management across this distributed data.

Unification

Dataplex enables you to unify data–distributed across data lakes, data warehouses, and data marts.

- **Data warehouse**: is a system that aggregates data from multiple sources into a single, central, consistent data store to support artificial intelligence (AI) and machine learning.
- **Data marts**: is a subset of a data warehouse focused on a particular line of business, department, or subject area. Data marts make specific data available to a defined group of users, which allows those users to quickly access critical insights without wasting time searching through an entire data warehouse.
- **Data lake**: provides massive storage of unstructured or raw data fed via multiple sources, but the information has not yet been processed or prepared for analysis. As a result of being able to store data in a raw format, data lakes are more accessible and cost-effective than data warehouses.



Deployment management

TOOLS

- Kubernetes

Kubernetes

MLOps • 2022

Kubernetes (k8s)

CONTAINERIZATION

Kubernetes, an open source platform that has gained a lot of traction in the past few years and is becoming the standard for container orchestration, greatly simplifies these issues and many others. It provides a powerful declarative API to run applications in a group of Docker hosts, called a Kubernetes cluster. The word declarative means that rather than trying to express in code the steps to set up, monitor, upgrade, stop, and connect the container, users specify in a configuration file the desired state, and Kubernetes makes it happen and then maintains it.

Kubernetes will allocate the hosts, start the containers, monitor them, and start a new instance if one of them fails. Finally, the major cloud providers all provide managed Kubernetes services; users do not even have to install and maintain Kubernetes itself. If an application or a model is packaged as a Docker container, users can directly submit it, and the service will provision the required machines to run one or several instances of the container inside Kubernetes.

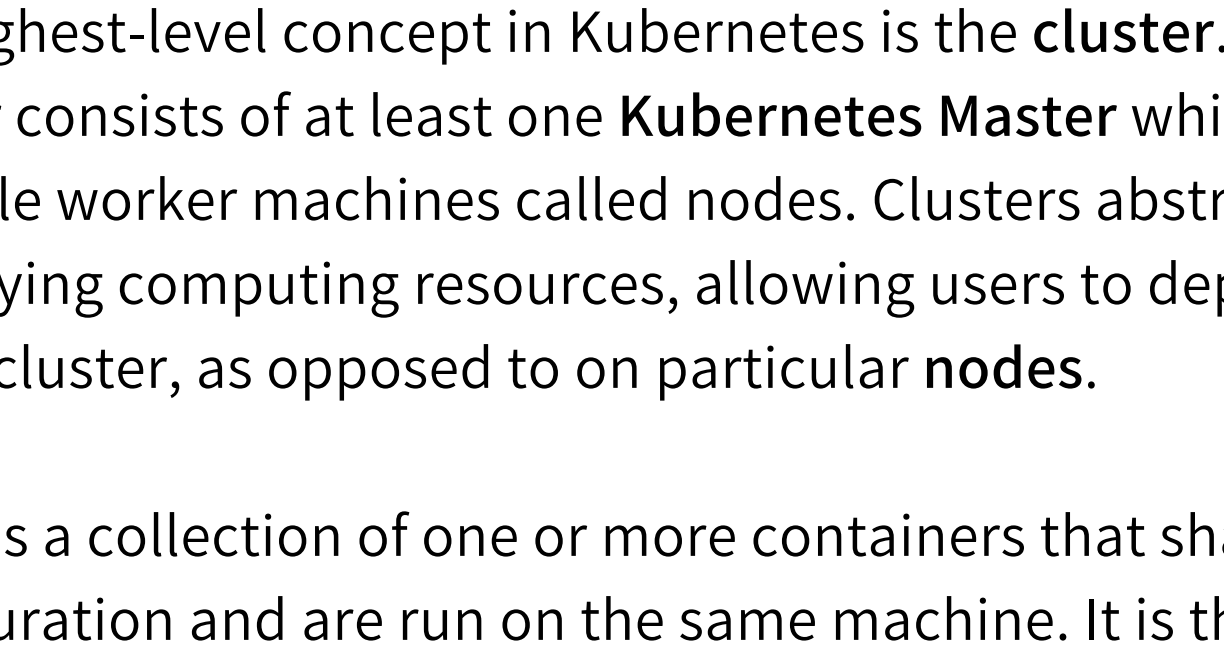
Docker with Kubernetes can provide a powerful infrastructure to host applications, including ML models. Leveraging these products greatly simplifies the implementation of the deployment strategies—like blue-green deployments or canary releases— although they are not aware of the nature of the deployed applications and thus can’t natively manage the ML performance analysis. Another major advantage of this type of infrastructure is the ability to easily scale the model’s deployment.

Kubernetes architecture

Kubernetes takes a hierarchical approach to managing the various resources that it is responsible for, with each layer hiding the complexity beneath it.

The highest-level concept in Kubernetes is the **cluster**. A Kubernetes cluster consists of at least one **Kubernetes Master** which controls multiple worker machines called nodes. Clusters abstract their underlying computing resources, allowing users to deploy workloads to the cluster, as opposed to on particular **nodes**.

A **pod** is a collection of one or more containers that share common configuration and are run on the same machine. It is the basic workload unit in Kubernetes. All containers within a pod share the same context, resources, and lifecycle. Resources include local and remote storage volumes and networking. All containers in a pod share an IP address and port space. To run containers in pods, Kubernetes uses a container runtime, such as Docker, running on each node.

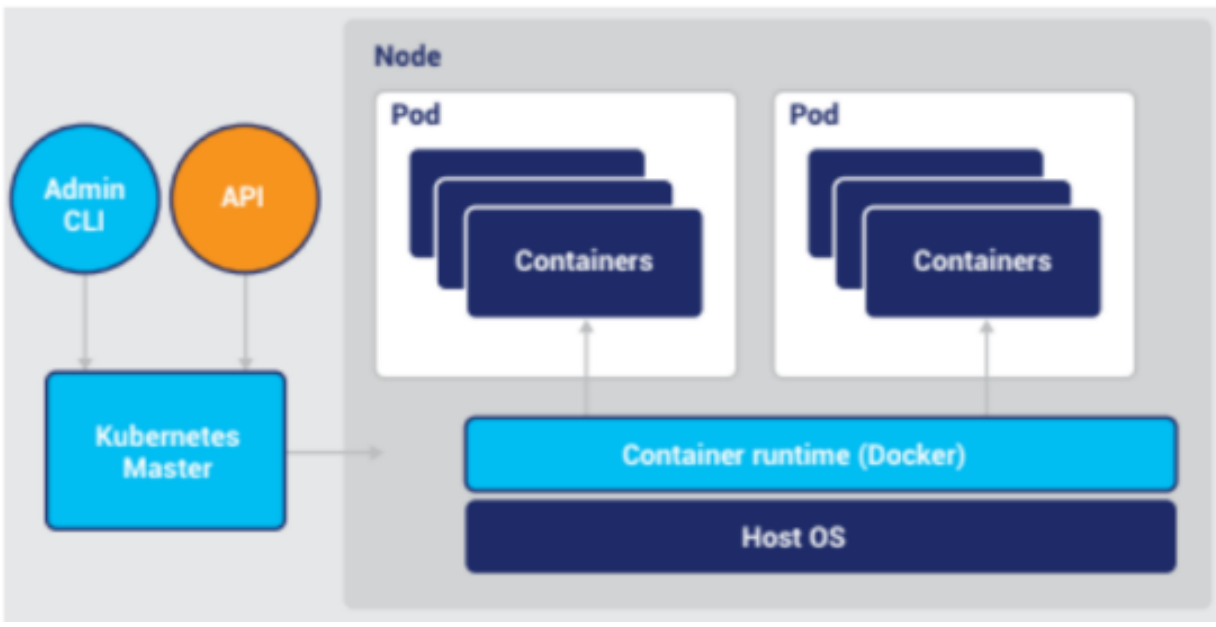


Kubernetes architecture

The master can be thought of as the “brain” of the cluster. It responds to cluster configuration and management requests submitted via the Kubernetes client or API. It is responsible for determining which pods are deployed to which nodes based on their respective requirements and capabilities, a process called **scheduling**.

Nodes in a Kubernetes cluster run an agent called the **kubelet** that listens for instructions from the master and creates, runs, and destroys containers accordingly.

The user provides a plan that defines which pods to create and how to manage them. This plan can be specified via configuration documents which are sent to the cluster via Kubernetes’ APIs or client libraries. In this way, Kubernetes is said to be a **“declarative” system**; users declare the state of the system they want, and Kubernetes tries to affect that state given the resources at its disposal.
When pods are scheduled to a node, the node pulls the appropriate container images from an image registry and coordinates with the local container runtime to launch the container.



Kubernetes for Machine and Deep Learning

Simplifying Data Management

- Kubernetes provides connectors for diverse data sources and manages volume lifecycle
- Data workflow & pipeline abstractions allow complex data transformations
- Data fabrics extend scalable, multi-format storage to cluster

Driving Efficient Resource Use

- Provides elasticity, allowing cluster and workloads to be easily scaled up/down
- Users and tools can programmatically deploy and control workloads

Hiding Complexity

- Containers provide a convenient format for packaging workloads and declaring dependencies
- Kubernetes abstracts infrastructure, allowing users to think of cluster as unit of compute

Kubeflow

Kubeflow has since published a general availability (GA) release of the project, and now aspires to support a wide variety of end-to-end machine learning workflows on Kubernetes.

Kubeflow is an open source project designed to make machine learning workflows on Kubernetes **simple, portable and scalable**. Kubeflow is sponsored by Google and inspired by TensorFlow Extended, or TFX, the company's internal machine learning platform.

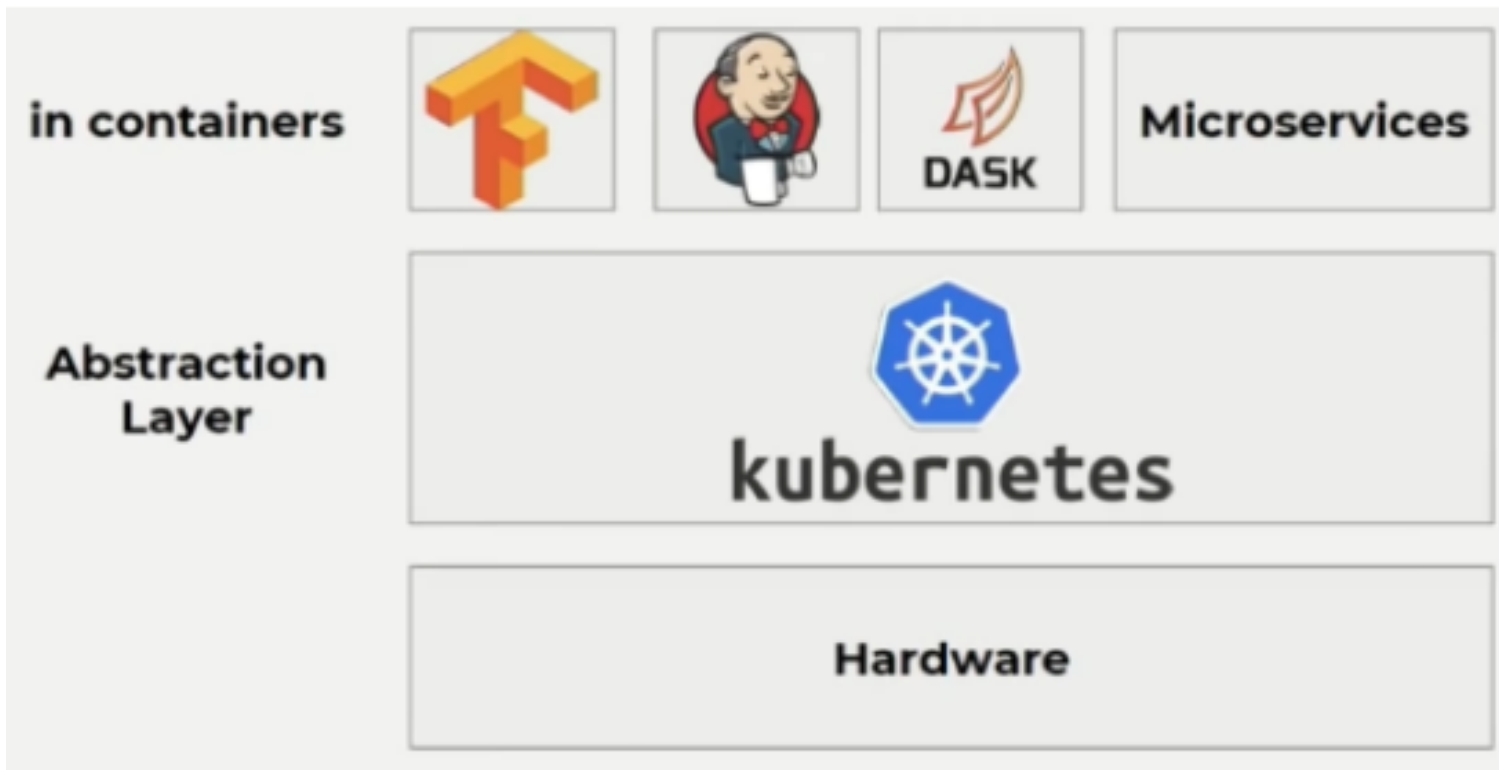
Kubeflow provides a **central dashboard** through which users can access and manage the various components of a Kubeflow deployment, support for multi-tenancy and multi-user isolation, and through Kubeflow Fairing, the ability to more easily build, train, and deploy training jobs across a hybrid cloud environment directly from a Jupyter notebook.

Example Kubernetes

k8s is basically an abstraction between **your hardware and whatever you want to run** (Dask, microservices, TensorFlow jobs, Jenkins).

k8s doesn't really care what is it running in most cases. It only ran containers. It doesn't really care what's inside these containers and that's the good thing about it that you can have multiple workloads of different types running on the same cluster.

Complete example: https://www.youtube.com/watch?v=fYGu1ideQkw



Example Kubernetes

We need 2 things:

- **Dockerfile**, which is basically the definition of the container.
    - you cloned the TensorFlow graph alongside the config file that you have, you install the dependencies, and then you basically start the tuning.

```
# ---- Dockerfile
FROM python3:latest
ADD start.sh /opt

ENV RNN_REPO http://repo.git
ENV RNN_BRANCH master
CMD ["bash", "/opt/start.sh"]
```

- **k8s config**, which is basically a deployment file that we use in order to tell k8s what to do in this case. What should be the scheduling constraints, what should be the scheduling features that we interested in to run this job.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: rnn-training
spec:
      ....
      containers:
      - name: rnn-training
        image: v0.1.0
      ...
      affinity:
        # schedule only one rnn training job on one node
        # TF is configured to consume all CPUs
        podAntiAffinity:
          ...
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - rnn-training
            topologyKey: "kubernetes.io/hostname"
```

```
# schedule only on "machine_learning" nodes
nodeAffinity:
    ...
    nodeSelectorTerms:
    - matchExpressions:
      - key: machine-type
        operator: In
        values:
        - "machine_learning"
```

Example Kubernetes

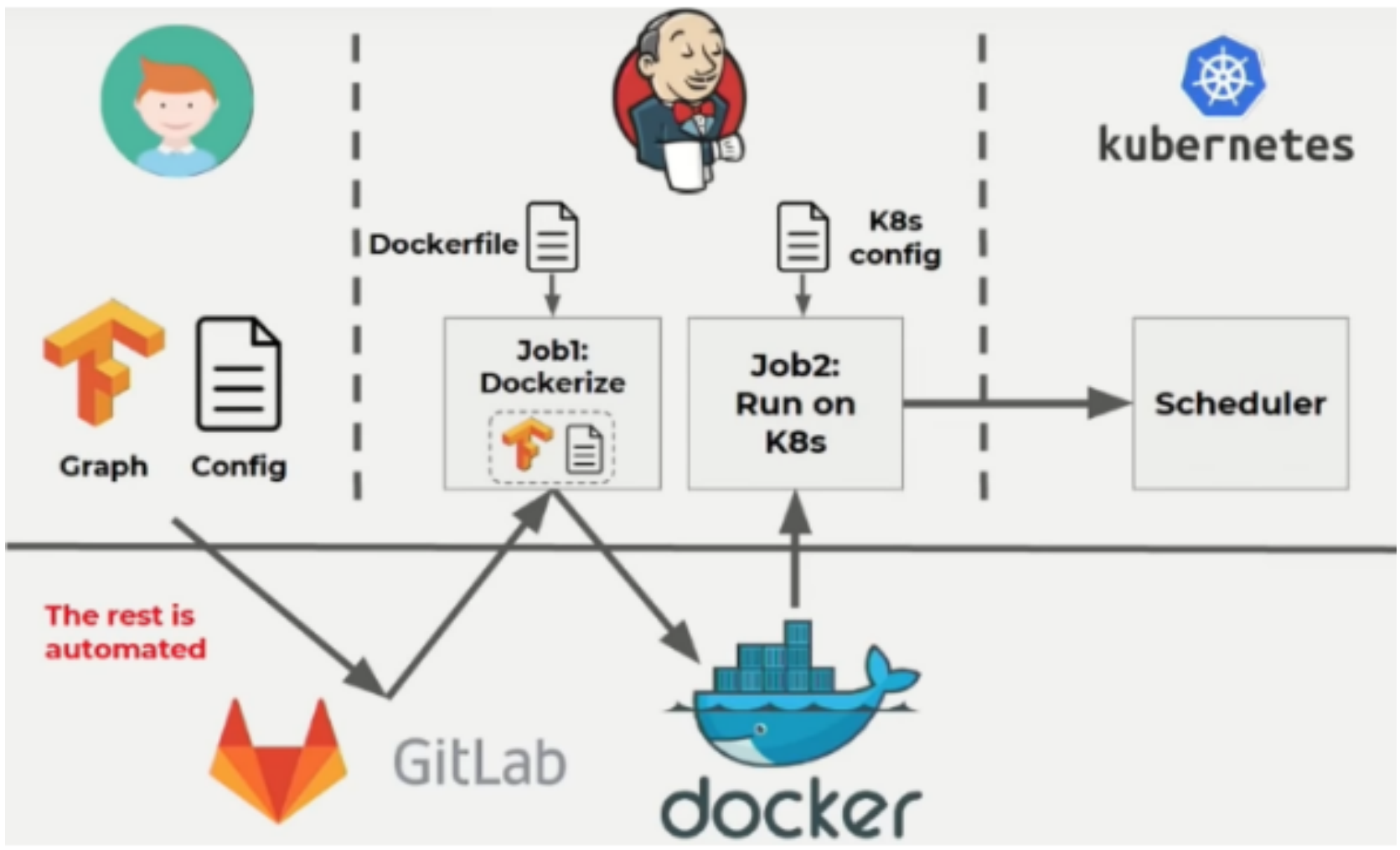
We see **Jenkins**, which just scheduled a job to kubernetes.

Initially we have a **TensorFlow Graph** and **a Config file,** and we push in the **GitLab**. And the rest in GitLab is basically automated.

Now we have a job which pulls in the **Docker file** and builds the **Docker image**. This also contains the dependencies, the TensorFlow Graph.

After we build a **Docker image**, that we push to a Docker Registry, and then this is pulled by k8s alongside a deployment file, in order to start this in k8s.

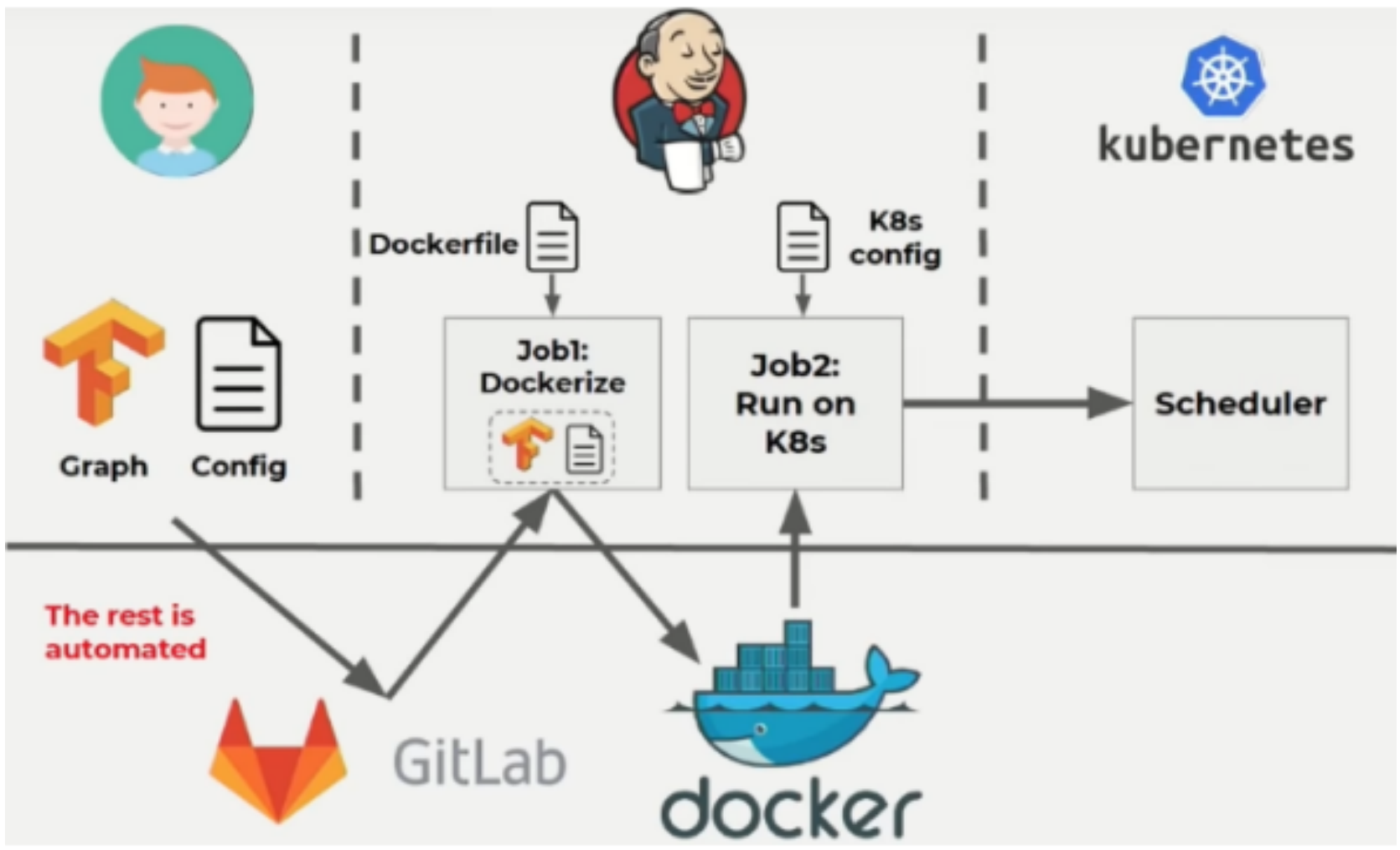
Keep in mind we have everything **documented** so we have the docker file, the deployment file, everything is saved so we can reproduce whatever the data scientist has done and if these models are versioned then we have can backtrack the whole process basically.



Example Kubernetes

Recap:

- data scientist modeling and the operations is being done automatically behind the scenes by using tools like Docker, K8s and Jenkins to do the scheduling to fire this jobs.
- And managing environment using k8s because these environment are pre-build as docker images and you can have reproduce ability in the sense of you can trace back what happened by checking and get what the data scientist have just pushed and what Jenkins has just scheduled.

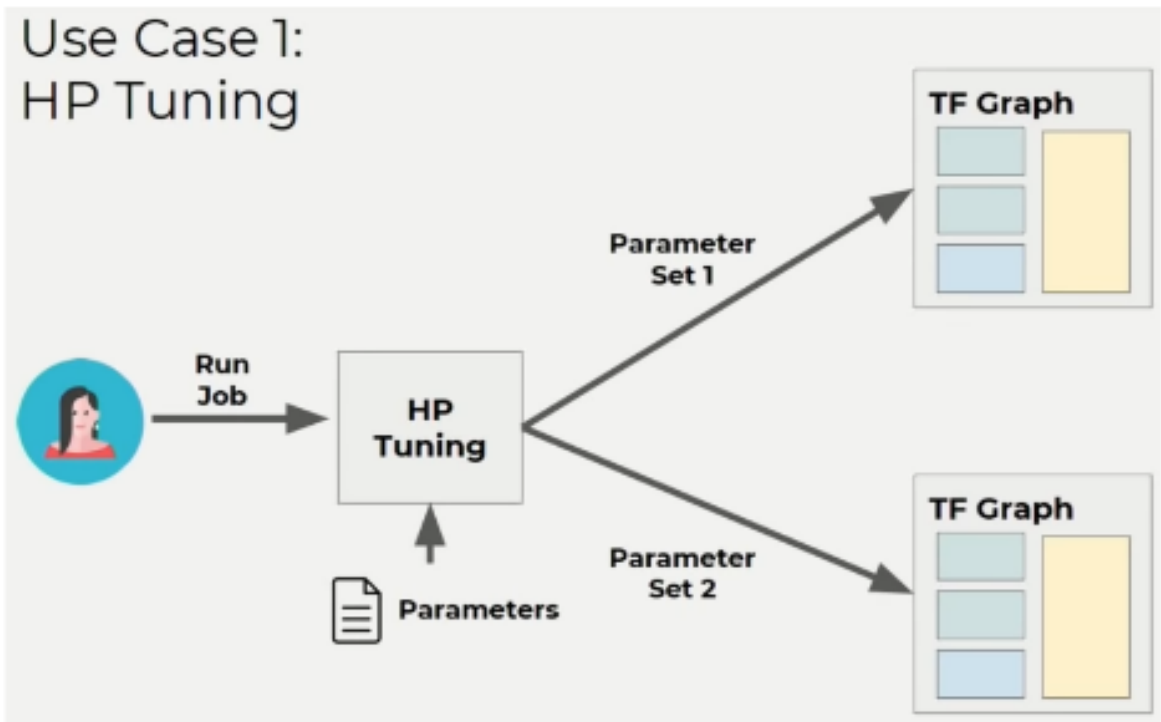


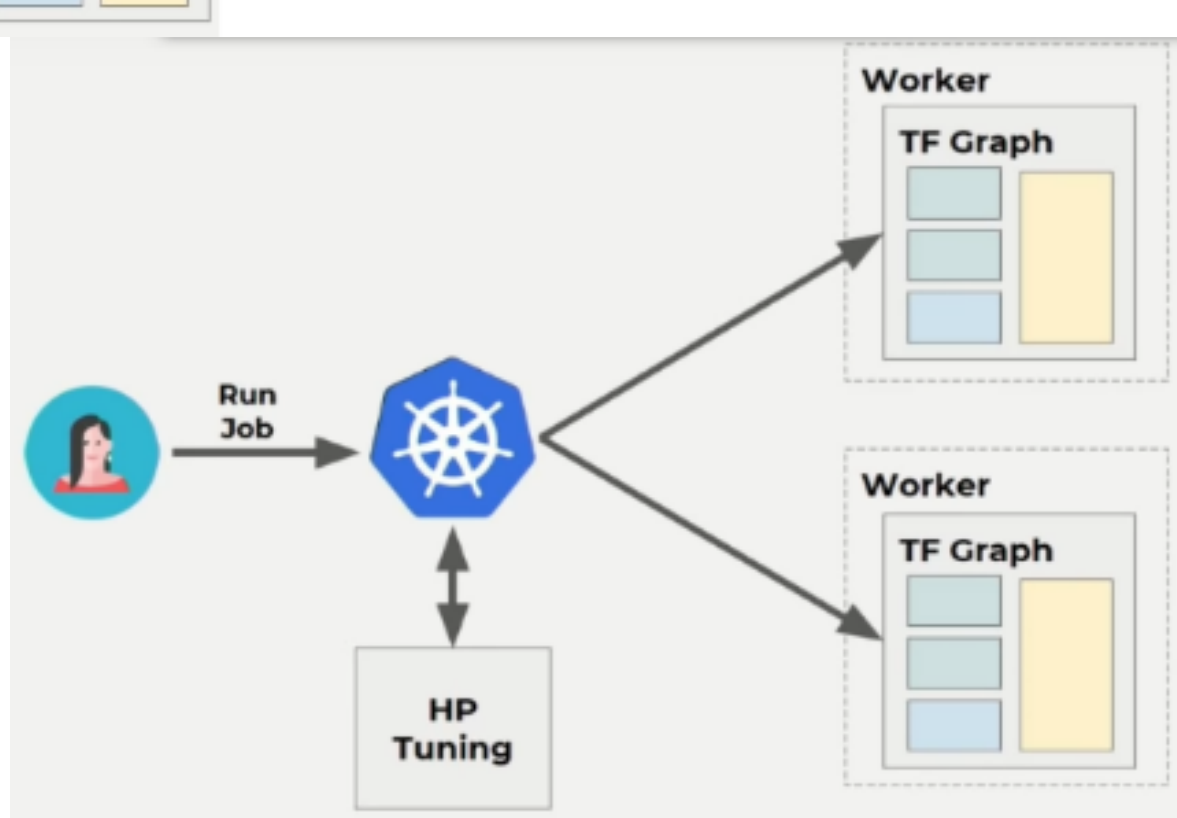
Example Kubernetes

We check the scalable solution, for having to use more hw.

We have the **Hyperparameter tuning solution**: which means that you get set of parameters and then you start different tunes with distinct set of parameters

We have also **KubeFlow**: which separates the TensorFlow graph between different nodes (machines). Embedding the Hyperparameters tuning training logic within the kubernetes scheduler. K8s as a API that everyone can use and the scheduler has an API. You can write your scheduler in 10 lines of bash that can know where and when to run the next tune run on a separate machine.





Documentation

TOOLS

- Data Sheets and Model Cards.
- -> https://arxiv.org/pdf/1803.09010.pdf
- -> https://arxiv.org/pdf/1810.03993.pdf

DATA SHEETS

Data Sheets record which kind of mechanisms or procedures were used for data collection or whether ethical review procedures took place.

MODEL CARDS

Model Cards complement Data Sheets and provide information about the development of a model, the assumptions that have been made, and expectations about model behavior across different cultural, demographic, or phenotypic groups.

Metadata View

MODEL INFORMATIONS

A good UX (User Experience) for take informations about the models, is

Moddel Service Catalog

MODEL SERVICE CATALOG

The Model Service Catalog is an internal marketplace of all ML models in the enterprise. The catalog should have a good UX, it should be connected to the location of model storage, and it should display relevant metadata for a model such as its latest version, inputs, and outputs. Employees with appropriate permissions can access the catalog, search for models, and retrieve requested information about the models.

ML security management

ADVERSARIAL ML THREAT MATRIX

Using this Adversarial ML Threat Matrix might be helpful: it is comparable to the classic Attack Chain Modeling and builds on the well-established MITRE Att&CK, a globally accessible knowledge base of attacks and techniques. MITRE Att&CK is used as a guideline for the development of specific threat models and methods in the private sector, in governments, and in the field of cyber security products and services. The Adversarial ML Threat Matrix is a similar concept for ML security – it contains a collection of known weaknesses and the associated attacks.

See: https://github.com/mitre/advMLthreatmatrix

References

References

Link

- https://ml-ops.org/content
- https://services.google.com/fh/files/misc/ai_adoption_framework_whitepaper.pdf
- https://www.oracle.com/it/integration/what-is-data-mesh/
- https://www.datamesh-architecture.com/
- https://neptune.ai/blog/mlops
- https://valohai.com/blog/the-mlops-stack/
- https://hbr.org/2018/04/a-simple-tool-to-start-making-decisions-with-the-help-of-ai
- https://the.gt/geoffrey-moore-positioning-statement/
- https://www.ibm.com/cloud/learn
- https://cloud.ibm.com/docs/watson-knowledge-studio?topic=watson-knowledge-studio-evaluate-ml
- https://cloudpricingcalculator.appspot.com/
- https://www.ucop.edu/local-human-resources/_files/performance-appraisal/How%20to%20write%20SMART%20Goals%20v2.pdf
- https://adr.github.io/madr/
- https://www.oracle.com/it/integration/what-is-data-mesh/
- https://www.datamesh-architecture.com
- https://www.ibm.com/uk-en/analytics/business-intelligence
- https://cloud.google.com

Book

- Introducing MLOps How to Scale Machine Learning in the Enterprise (Mark Treveil, Nicolas Omont, Clément Stenac etc.)
- Kubernetes for MLOps - Scaling Enterprise Machine Learning, Deep Learning, and AI (Sam Charrington)