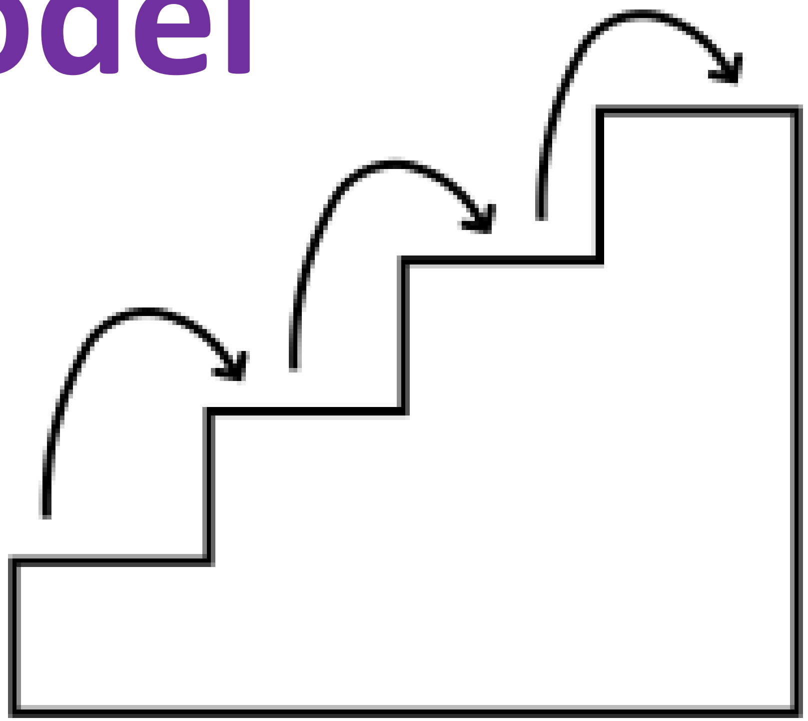
Psitron

Module 2

Introduction to ML and MLOps stages

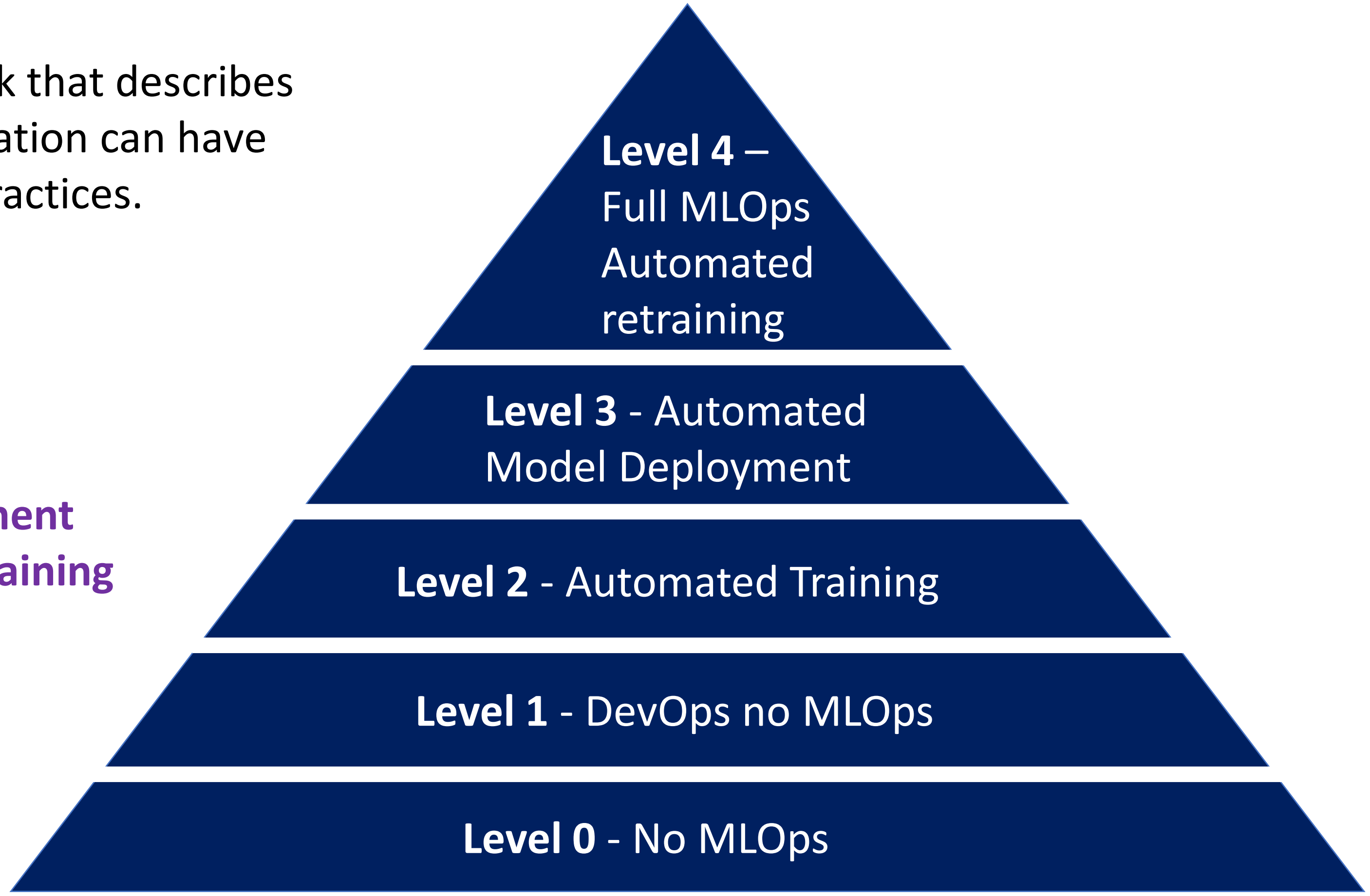
MLOps maturity model



Operational hierarchy of MLOps maturity model

- ✓ The MLOps maturity model operates entirely on **5 levels** with different **responsibilities** and **functionalities**
- ✓ The MLOps maturity model is a framework that describes the different levels of maturity an organization can have when it comes to implementing MLOps practices.

  - ✓ **Level 0: No MLOps**
  - ✓ **Level 1: DevOps no MLOps**
  - ✓ **Level 2: Automated Training**
  - ✓ **Level 3: Automated Model Deployment**
  - ✓ **Level 4: Full MLOps Automated retraining**



• Level 0: No MLOps

Model creation:

- Data is **gathered manually** and **preprocessed**
- Once the data is efficient a dummy-like model is developed to evaluate certain predictions

Model release:

- Models are released manually
- Model scoring script is manually scripted after certain **experiments** and it is **mainly used to validate the data available**

Application integration:

- Heavily dependent on data scientist interpretations from the model developed
- This level basically involves data gathering and model development
- Monitoring of the model is not taken up

Psitron

• Level 1: DevOps no MLOps

Model creation:

- Pipelines will have ability to generate data automatically
- Model parameters will be tracked and monitored a fewer number of times only

Model release:

- Models in the pipeline are **evaluated**
- The scores are **scripted** and passed on to the team of software engineers

Application integration:

- Models developed will be undergone various testing
  - ✓ Integration
  - ✓ Unit testing
  - ✓ Suitably evaluated according to software testing principles

- # Level 2: Automated Training

Model creation:

- Gathering data automatically from the pipeline
- Models developed are monitored and validated continuously

Model release:

- Models are released manually
- The model’s parameters are evaluated continuously with certain test parameters

Application integration:

- Models developed will be undergone various testing
  - ✓ Integration
  - ✓ Unit testing
  - ✓ Suitably evaluated according to software testing principles

- # Level 3: Automated Model Deployment

Model creation:

- **Effective modeling** and **managing** the models create
- **Training model code** and the **resulting model parameters** are efficiently handled

Model release:

- Model performance is scripted based on the outcomes of tests
- Entirely managed by the CI/CD pipelines

Application integration:

- Models are monitored
- Deployed in the form of an application
- Model deployed will be monitored on the basis of software testing principles

- # Level 4: Full MLOps Automated retraining

Model creation:

- Responsible for **triggering the model for retraining** with respect to the **feedback received after continuously monitoring** the model that is present in production

Model release:

- Model metrics are scripted after monitoring in the pipeline is received and the model is retrained accordingly

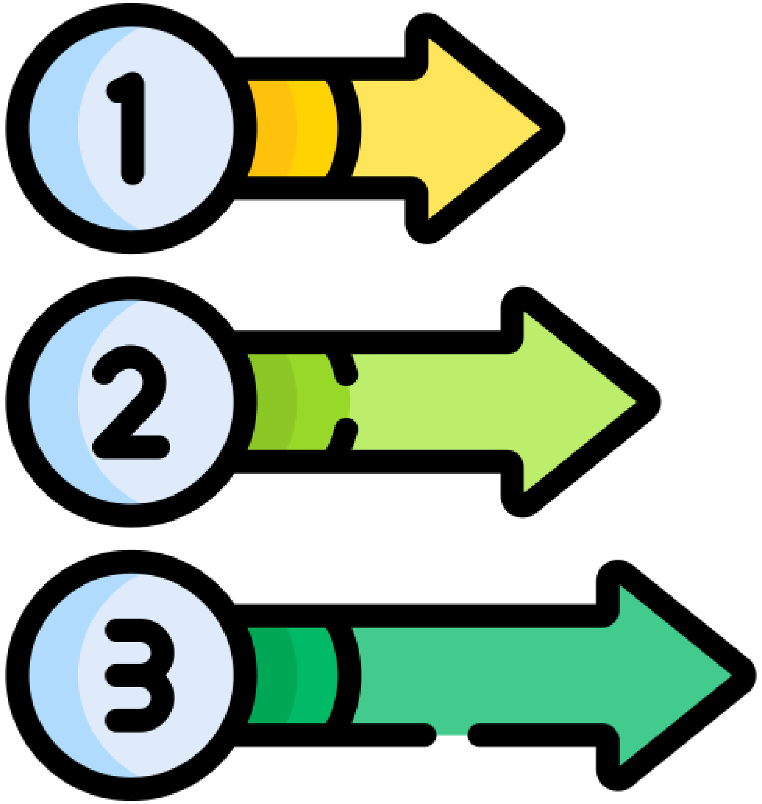
Application integration:

- Model is only evaluated
- Monitored continuously through unit tests and software testing principles.

Detailed MLOps and stages

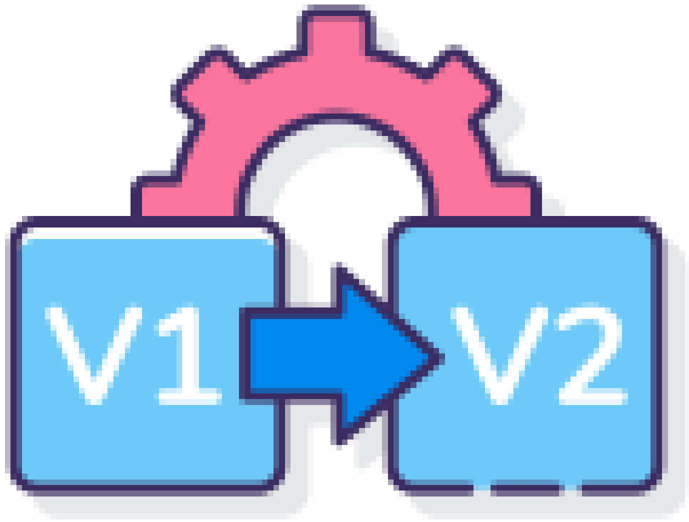
The stages of MLOps include

- ✓Versioning
- ✓Testing
- ✓Automation (CI/CD)
- ✓Reproducibility
- ✓Deployment
- ✓Monitoring



Versioning

- ✓ In MLOps, versioning is a critical aspect of managing machine learning models in production.
- ✓ key practice to keep track of changes to **data**, **code**, **models**, **features**, and **containers**.
  - Versioning Data
  - Versioning Code
  - Versioning Models
  - Versioning Features
  - Versioning Containers



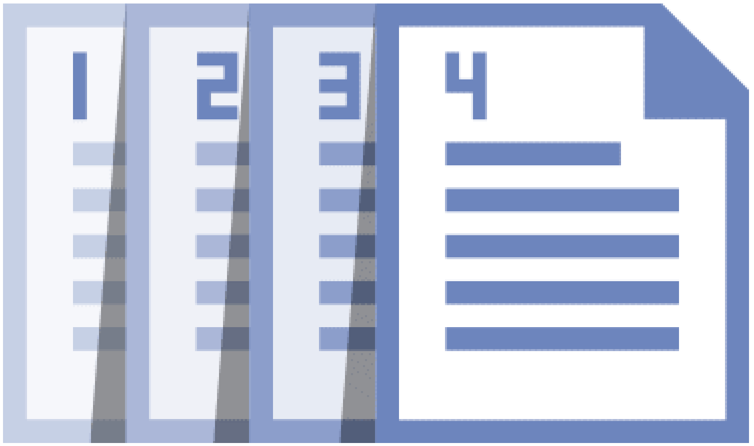
Versioning Data

- ✓ Keeping track of different versions of data is important to ensure that the model is being **trained** and **tested** on the **correct data.**

- ✓ Versioning data allows for easy rollback to previous versions of data if necessary.

- ✓ Version control systems such as **DVC** can be used to **track** and **version data files.**

- ✓ Cloud providers such as **AWS**, **Azure**, and **GCP** offer versioning capabilities for data stored on their platforms.



Tools for Versioning Data

- ✓ DVC (Data Version Control)
- ✓ DataRobot
- ✓ Git-LFS (Large File Storage)
- ✓ DataKit
- ✓ MLflow

Versioning Code

Psitron

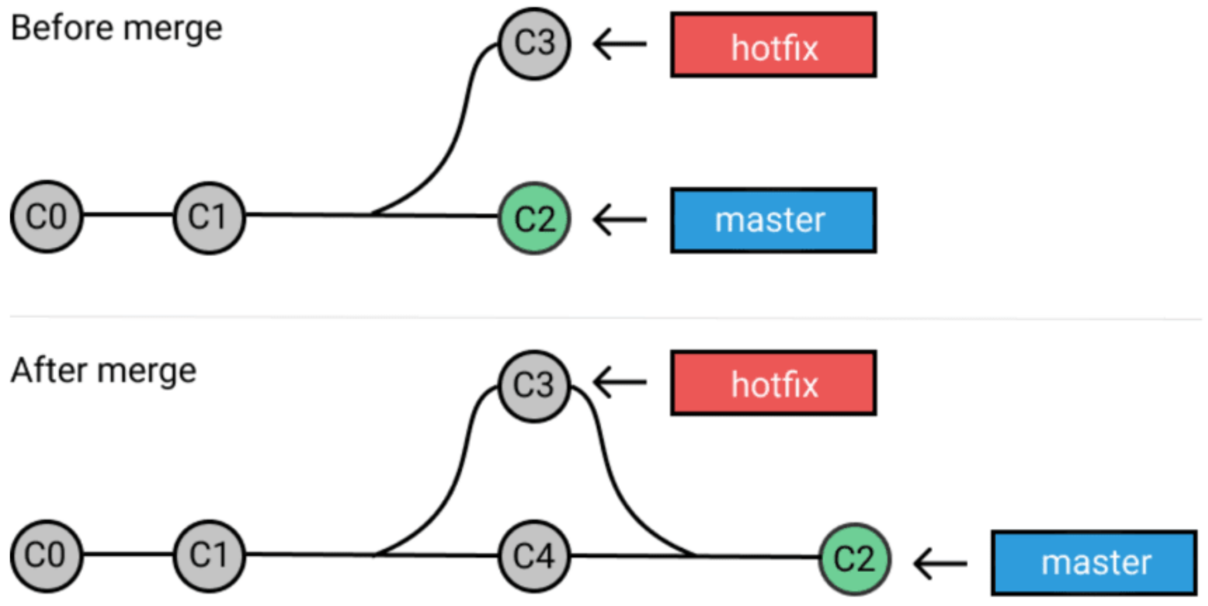
- ✓ In MLOps, **versioning code** is a key practice to keep track of **changes to the codebase** used for machine learning development.
- ✓ Versioning code helps to ensure **reproducibility** and **traceability** in the process.
- ✓ Version control systems such as **Git**, **AWS codecommit** can be used to achieve versioning code in MLOps.

Version Control Systems

- Version control systems allow for **tracking** and **versioning** of code changes over time.
- They enable **multiple developers to work on the same codebase simultaneously**.
- Version control systems maintain a **clear history of changes.**
- **Git** is the most widely used **version control** system in the industry.

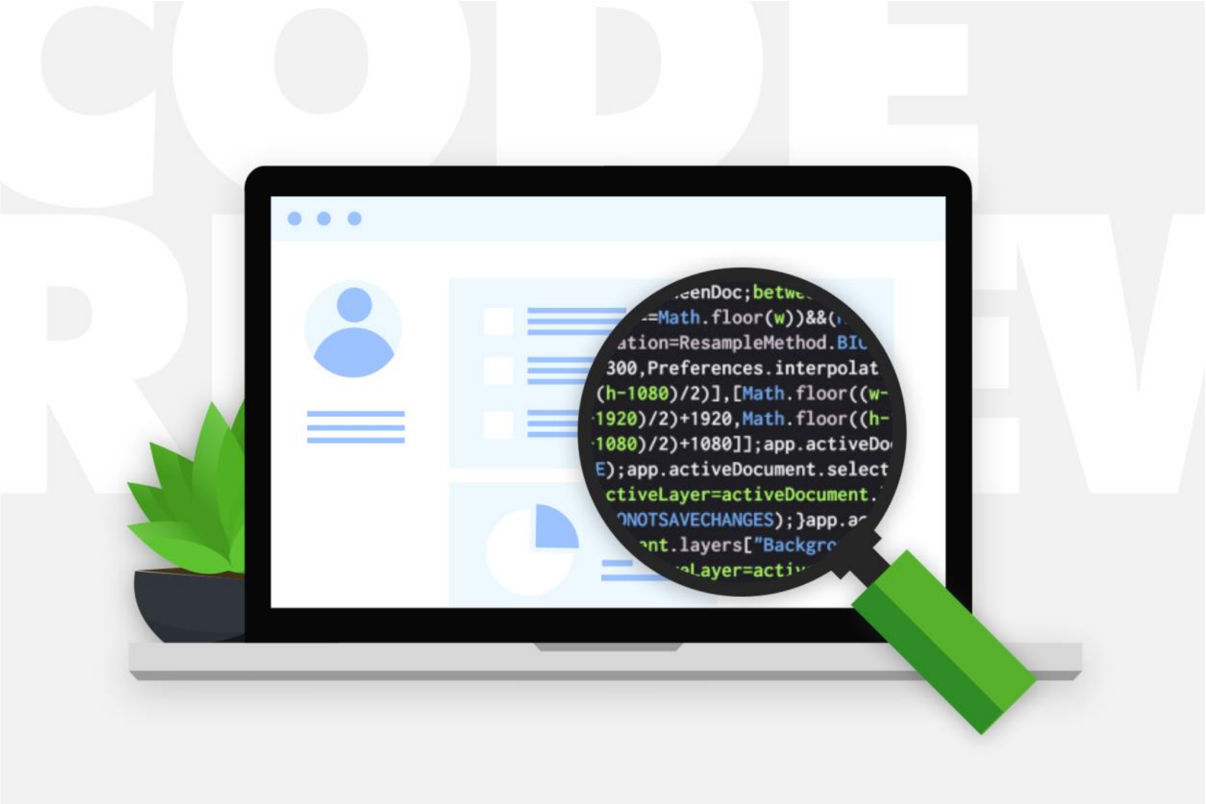
Branching and Merging

- Version control systems allow for the **creation of branches.**
- Branches are **independent copies** of the codebase.
- Branches can be developed separately.
- This allows for **isolated development** and **testing** of new features.
- The **main codebase remains stable** when using branches.



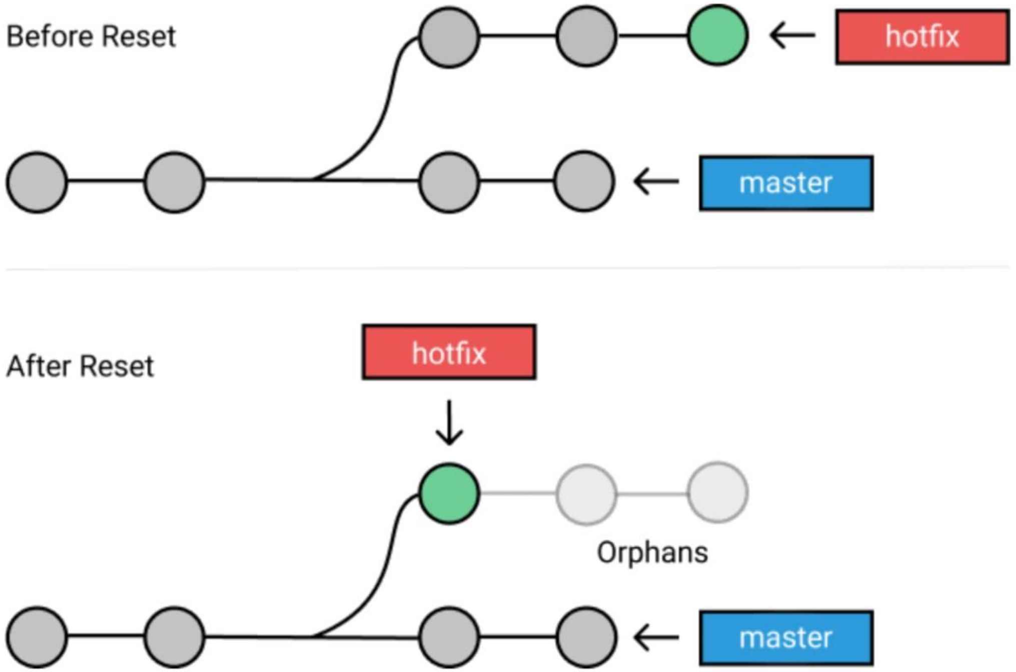
Code Review

- Version control systems allow for **code review.**
- Other **developers can review** and **provide feedback on code changes**.
- Code review happens **before code changes are merged into the main codebase.**
- Code review helps to ensure high code quality.
- Code review helps to **identify** and **address issues** early on.



Rollback

- Versioning code allows for easy **rollback** to **previous versions** if necessary.
- Rollback is used in case **of bugs or issues.**
- Previous versions of code can be accessed by checking out.
- Specific changes can be reverted.



Tools for Versioning code

- ✓ GitHub
- ✓ Bitbucket
- ✓ Apache Subversion
- ✓ AWS CodeCommit
- ✓ Azure Repos
- ✓ GCP Source Repositories

GitHub

SUBVERSION

GCP Source Repositories

ATLASSIAN Bitbucket

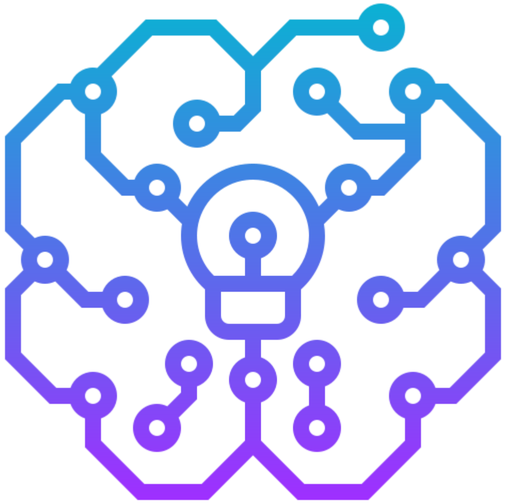
Azure Repos

AWS CodeCommit

Psitron

Versioning Models

- ✓ Keeping track of **different versions of models** is important for **reproducibility** and **traceability**.
- ✓ Versioning models allows for easy **rollback** to previous versions if necessary.
- ✓ Cloud providers such as **AWS**, **Azure**, and **GCP** offer **versioning capabilities for models stored on their platforms.**



Model Metadata Management

- ✓ Tools like **MLflow**, **DVC**, and **Comet** allow to **store model metadata.**
- ✓ Model metadata includes **version number**, **training data**, **evaluation metrics**, and **hyperparameters** used.
- ✓ Metadata can be stored in a database or version control system.
- ✓ Metadata is stored alongside the model files.

Model Registry

- ✓ Tools like **Seldon**, **MLflow**, and **AWS SageMaker** Model Registry provide a centralized repository for storing and managing models.
- ✓ They allow for easy discovery and management of models.
- ✓ They can be integrated with **version control systems** to **track** and **version models.**

Psitron

Cloud Providers

- ✓ Cloud providers such as **AWS**, **Azure**, and **GCP** offer versioning capabilities for models.
- ✓ These services can automatically version models as they are **trained** and **deployed**.
- ✓ These services provide an easy way to **roll back** to previous versions if necessary.

Tools for Versioning model

mlflow™

DVC

SELDON

Amazon SageMaker

AWS SageMaker Model Registry

Azure Machine Learning

Model Management

TensorFlow Extended

neptune.ai

Versioning Features

- ✓ Versioning features refers to the process of keeping **track of different versions of the features** used in the **model throughout the development process**
- ✓ This helps to ensure **reproducibility** and **traceability**.
- ✓ Allows for easy **rollback** to previous versions if necessary.

Version Control Systems

- ✓ Version control systems can be used to **track** and **version feature files.**
- ✓ This allows for a **clear history of changes to the features.**
- ✓ This makes it easier to **reproduce results**

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. (	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mr:	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, M	female	26	0	0	STON/O2. 31	7.925		S
5	4	1	1	Futrelle, Mrs	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. Wi	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. J	male		0	0	330877	8.4583		Q

Psitron

Feature Metadata

- ✓ Keeping track of feature metadata is important for **reproducibility** and **traceability**.
- ✓ Feature metadata includes **version number**, **creation date** and **author**.
- ✓ This metadata can be stored in a **database** or **version control system.**
- ✓ The metadata is **stored alongside the feature files.**

Feature Versioning

- ✓ Keeping track of different **versions of features** is important for **reproducibility** and **traceability**.
- ✓ Feature versioning allows to **compare features**.
- ✓ Feature versioning allows to **understand their characteristics.**
- ✓ Feature versioning allows to **choose the most suitable one for a given task.**

Psitron

Feature Management

- ✓ A system that **manages the features is important.**
- ✓ This system can include **a feature store.**
- ✓ **Feature store allows** to **organize**, **discover** and **version features.**
- ✓ **Automating feature versioning** and management process can help to avoid human errors.

Feature Management tools:

FEAST

Hugging Face

Hugging Face Feature Store

Featuretools

DataRobot

Azure

Amazon SageMaker

DVC

Automation

✓ Automating **feature versioning** and **management process** can speed up the process of creating and managing features.

Versioning Containers

- ✓ Keeping track of different **versions of containers** is important for **reproducibility** and **traceability**.
- ✓ Allows for easy **rollback** to previous versions of containers if necessary.
- ✓ This can be achieved through the use of **container registry such as Docker Hub.**
- ✓ Container images can be **tracked** and **versioned** using **container registry.**
- ✓ Versioning containers is important for maintaining the integrity of the **model**, **data**, and **code throughout** the MLOps pipeline.

Tools for Versioning Containers

docker hub

Amazon ECR

JFrog Artifactory

HARBOR

GitLab Container Registry

Testing in MLOps

Testing in MLOps

- ✓ Testing is an important part of MLOps, it helps to ensure the **quality** and **reliability** of the **machine learning models.**
- ✓ There are several types of testing that can be performed in MLOps:
  - ✓ Unit Testing
  - ✓ Integration Testing
  - ✓ Functional Testing
  - ✓ Performance Testing
  - ✓ A/B Testing
  - ✓ Continuous Testing
  - ✓ Exploratory Testing
  - ✓ User Acceptance Testing
  - ✓ Data Quality Testing
  - ✓ Security Testing
  - ✓ Deployment Testing

Unit Testing

- ✓ It is done on **individual components** or **functions of the codebase** to ensure that they work as expected.

Integration Testing

- ✓ It is done to ensure that the **different components of the codebase work together correctly.**

Functional Testing

- ✓ It is done to ensure that the **machine learning model works as expected and produces** the desired output.

Performance Testing

- ✓ It is done to ensure that the **machine learning model performs** well in terms of **speed**, **memory usage**, and **other performance** metrics.

A/B Testing

- ✓ A/B TestingIt is done to **compare the performance** of **two or more machine learning models**, by running them on the same data and comparing their results.

Continuous Testing

- ✓ It is done to **automate the testing process** and ensure that the machine learning models are **thoroughly tested and validated before deployment.**

Exploratory Testing

- ✓ It is done to test the model's ability to generalize and adapt to **new data**, and to **identify any** **errors** or **biases** in the model.

Psitron

User Acceptance Testing

- ✓ It is done to ensure that the machine learning model meets the **requirements of the end-users** and **stakeholders**, and that it is usable and reliable.

Data Quality Testing

- ✓ It is done to ensure that the **data used to train and test machine learning models** is **accurate**, **reliable**, and **free from errors** or **biases**.

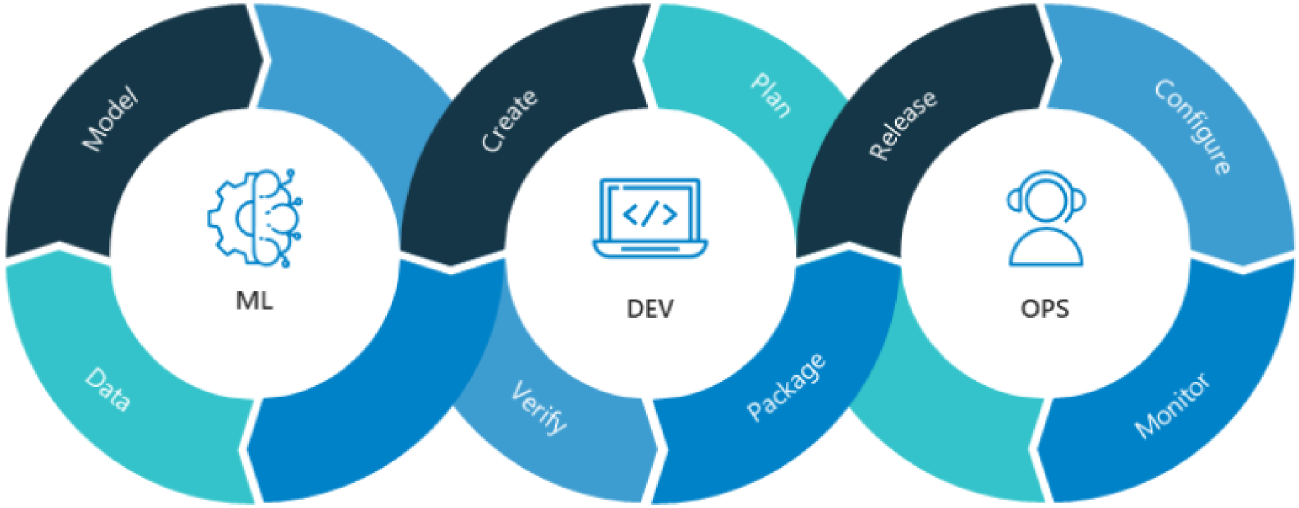
Security Testing

- ✓ It is done to ensure that the **machine learning models** and the **data** they use are **secure from any kind of unauthorized access or attacks.**

Deployment Testing

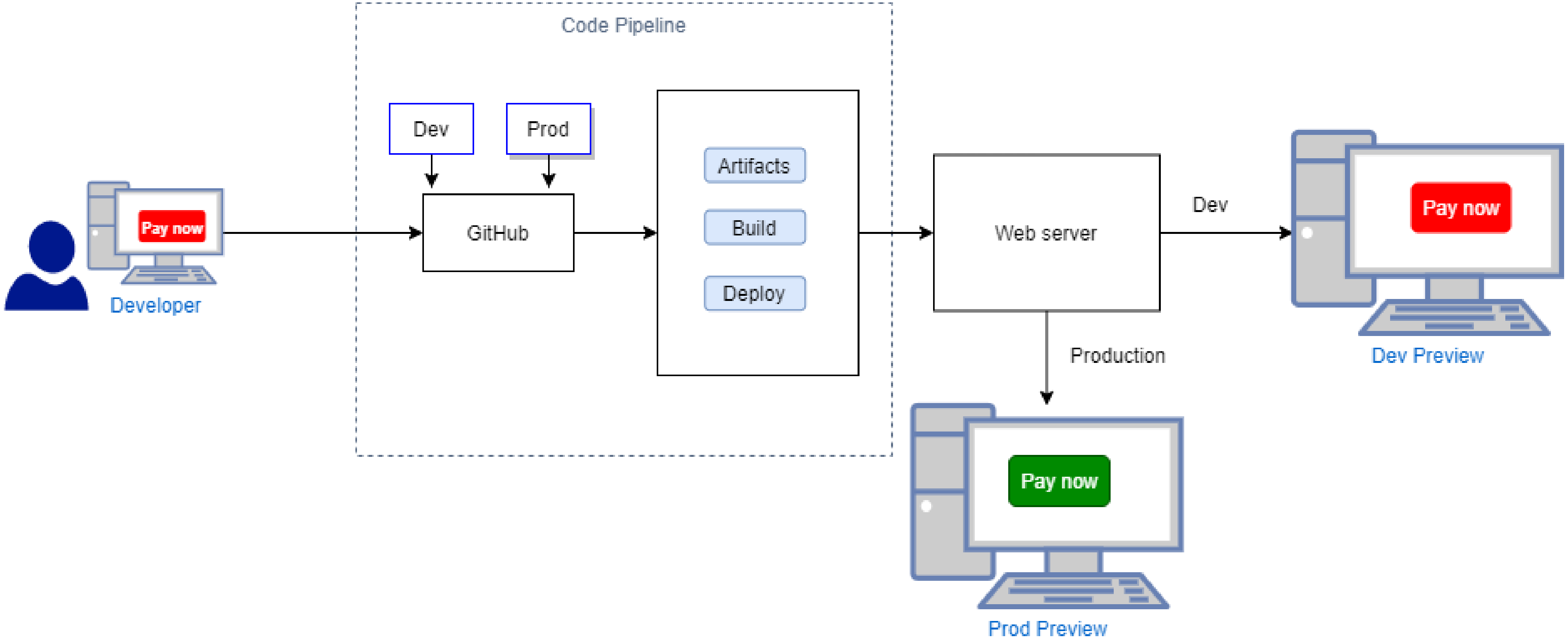
- ✓ It is done to ensure that the **machine learning model is deployed correctly** and **is working as expected in the production environment.**

Automation (CI/CD)



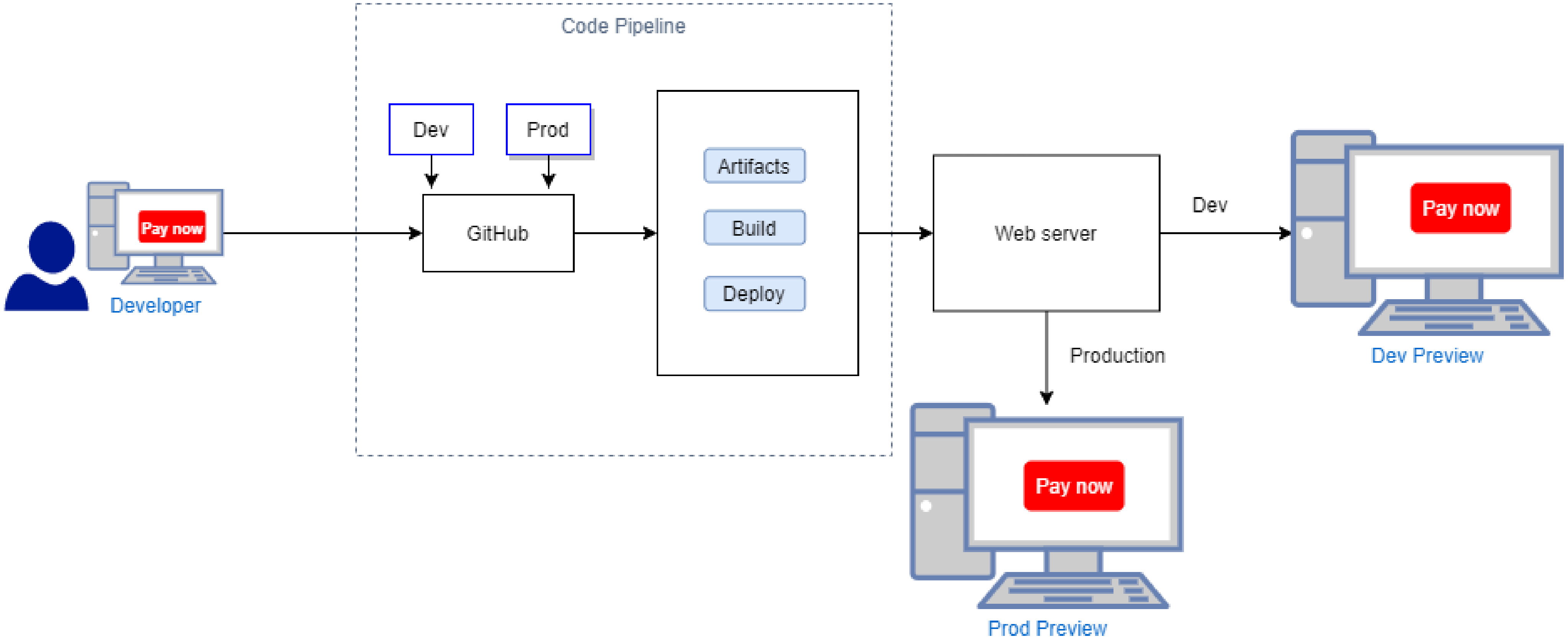
Continuous integration and Continuous deployment (CI/CD)

CI/CD stands for "Continuous Integration/Continuous Deployment" and is a practice that automates the process of building, testing, and deploying software changes.



© 2023 Psitron Technologies Pvt.Ltd

Continuous integration and Continuous deployment (CI/CD)



Reproducibility

- ✓ Reproducibility refers to the ability to **recreate** or **replicate**.
- ✓ Machine learning reproducibility is replicating the **ML workflow previously** carried out in a paper, tutorial, and producing the same results as the original work.
- ✓ **Reproducibility is crucial from large-scale deployments perspectives.**

There are several practices that can be used to improve reproducibility in MLOps:

- ✓ Version control
- ✓ Documenting the process
- ✓ Automating the process
- ✓ Using standard libraries and frameworks
- ✓ Keeping track of the metadata

Deployment

- ✓ There are several methods to deploy machine learning models in a **production environment, including:**

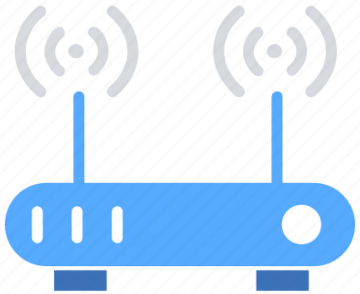
Cloud-based deployment

- ✓ Deploying the model to a cloud-based platform such as **AWS**, **Azure**, or **GCP**.
- ✓ Allows for easy scaling and management of the model.
- ✓ Can be done using containerization or serverless technologies.

Containerization

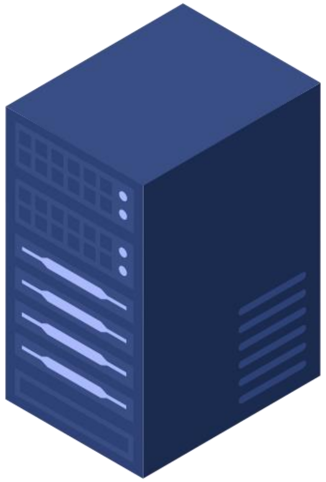
- ✓ Packaging the **model** and its **dependencies** in a **container**, such as a **Docker container.**
- ✓ Deploying it to a **container orchestration platform**, such as **Kubernetes**.
- ✓ This allows for **easy scaling** and **management** of the model.
- ✓ Can be done **on-premises** or in the **cloud**.

Edge deployment



- ✓ Deploying the **model to edge devices**, such as **IoT devices** or **mobile devices.**
- ✓ To perform **inferences locally.**
- ✓ This can be done using technologies such as **TensorFlow Lite** or **OpenVINO**.

On-premises deployment



- ✓ Deploying the **model to a physical or virtual machine** in an **on-premises data center.**
- ✓ This can be done using technologies such as **Kubernetes** or **OpenShift**.

Psitron

Function-as-a-service (FaaS)

- ✓ Deploying the model to a **function-as-a-service platform**, such **as AWS Lambda** or **Google Cloud Functions.**
- ✓ This allows for **easy scaling and management of the model.**
- ✓ Can be done using **serverless technologies.**

AWS Lambda

Google Cloud Functions

Hybrid deployment

- ✓ **Combining different methods** to deploy the model in multiple environments such as **cloud**, **edge** or **on-premises.**

Monitoring

✓ **Monitoring in MLOps** refers to the practice of **observing**, **measuring**, and **analyzing** the **performance**, **health**, and **behaviour** of **machine learning models** and **systems in a production environment**.

✓ To ensure that the **models are performing as expected** and to **identify** and **address** any **issues that arise.**

- Model performance monitoring
- Model health monitoring
- Model behaviour monitoring
- Model drift monitoring
- Infrastructure monitoring
- Logging and alerting
- Anomaly detection

Model performance monitoring

✓ Tracking the **performance of the model** in terms of metrics such as

- Accuracy
- Precision
- Recall
- F1-score

…and comparing it to the expected performance.

Model health monitoring

✓ Tracking the health of the **model by monitoring** its resource usage, such as **memory** and **CPU usage**, and identifying any potential issues that might cause the model to fail.

Model behaviour monitoring

✓ Tracking the **behaviour of the model by monitoring** its **input** and **output data**

✓ As well as its **interactions with other systems**, **to identify any potential issues or anomalies.**

Model drift monitoring

- ✓ Tracking the **changes of the model over time** and **identifying any drift from the expected behavior.**

Infrastructure monitoring

- ✓ **Monitoring the infrastructure that supports the model**, such as **servers** and **networks**, to ensure that it is functioning properly.

Logging and alerting

- ✓ **Logging the events** related to the **model** and the **infrastructure**
- ✓ And setting **alerts to notify** the **relevant teams when something unusual happens.**

Anomaly detection

- ✓ **Identifying unusual patterns** or **behaviors** in the **data** that might indicate a problem with the **model** or the **infrastructure**.

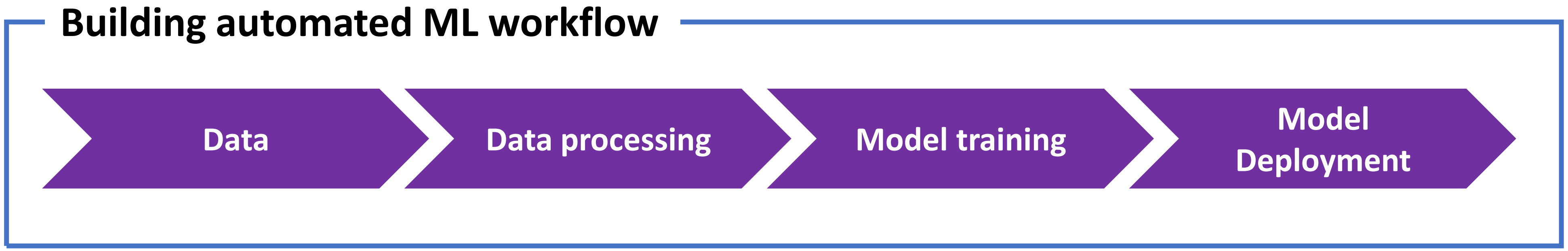
Monitoring Tools

- ✓ Evidently
- ✓ Grafana + Prometheus
- ✓ Amazon SageMaker
- ✓ TensorFlow Extended (TFX)
- ✓ Seldon Core
- ✓ Censius
- ✓ Neptune.ai
- ✓ Arize AI
- ✓ WhyLabs
- ✓ Qualdo
- ✓ Fiddler

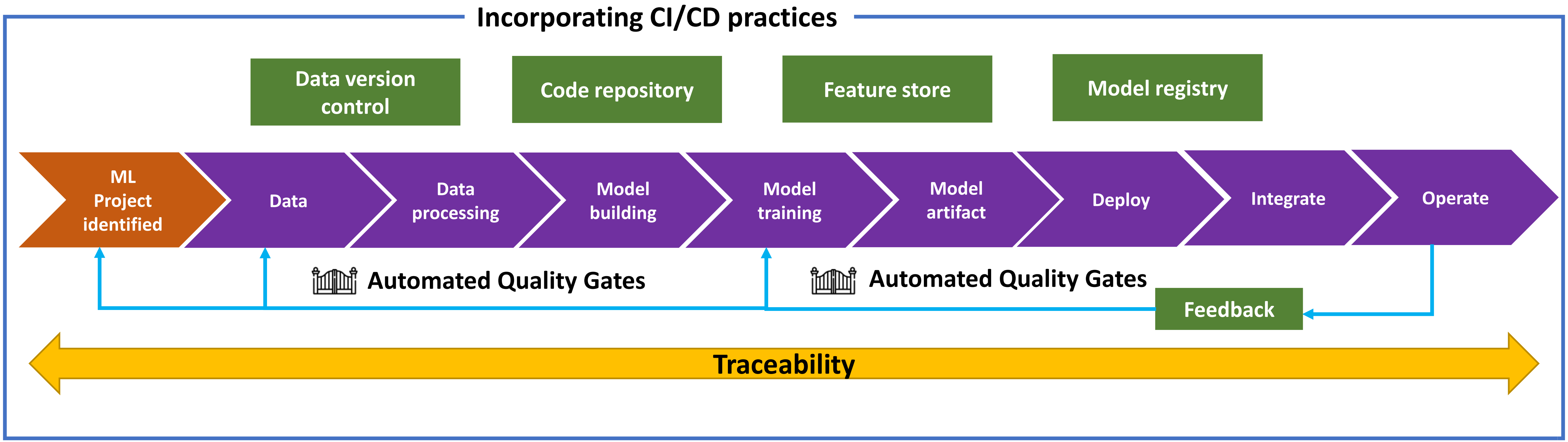
Automated ML pipelines vs CI/CD ML pipelines

Psitron

Moving from…



to…



MLOps Architectures

Architectures - Open Source tools

MLOps Architectures

Kubeflow

- ✓ Kubeflow is an open-source project created by Google.
- ✓ It makes it easy to use machine learning in a Kubernetes cluster.
- ✓ It allows for deploying and scaling machine learning projects on any infrastructure.
- ✓ It is designed to work with existing systems.

Jupyter notebooks

- Create and customize Jupyter notebooks
- Immediately see the results of running your code
- Create interactive analytics reports.

Custom TensorFlow job operator

- Helps **train your model**
- Apply a **TensorFlow** or **Seldon Core** serving container **to export the model to Kubernetes**

Simplified containerization

- Kuberflow **eliminates the complexity** involved in **containerizing the code.**
- Data scientists can perform **data preparation**, **training**, and **deployment in less time.**

MLflow

- **MLflow is an open-source platform** for machine learning engineers.

- It **helps to manage the ML lifecycle** through **experimentation**, **deployment**, and **testing**.

- It is useful for **tracking the performance of models**.

- It **acts as a dashboard** where you can

    ✓ Monitor the ML pipeline,
    ✓ Store model metadata, and
    ✓ Pick the best-performing model.

Four components of MLflow

Tracking

- ✓ The **MLflow Tracking component** is an **API** and **UI** for logging information about your **models**.
- ✓ It allows to **log parameters**, **code versions**, **metrics**, and **output files** from running the code.
- ✓ It **provides visualization** for the results.
- ✓ It allows you to **log** and **query experiments** using different **APIs** like **Python**, **REST**, **R**, and **Java**.
- ✓ It enables to **record the results.**

Four components of MLflow

Project

- ✓ **MLflow Project** is a tool that helps ML teams **organize** and **manage** their projects
- ✓ It allows for **reusable** and **reproducible projects**
- ✓ It has **API** and **command-line** tools
- ✓ It can run on any platform.

Model

- ✓ MLflow Model is a tool that helps package and deploy machine learning models
- ✓ It supports different tools like Apache Spark
- ✓ It facilitates the usage of models in diverse serving environments.

Model Registry

- ✓ **MLflow Model Registry** is a tool that helps teams manage the **lifecycle of an MLflow model**
- ✓ It allows for **versioning** and **storage of models**
- ✓ It provides **tracking of model lineage** and **transitioning between stages**
- ✓ It includes a **UI** and **APIs** for better collaboration.

Metaflow

Psitron

- ✓ Netflix created **Metaflow**
- ✓ Metaflow is an open-source MLOps platform
- ✓ Metaflow is used for **building** and **managing large-scale** data science projects
- ✓ Metaflow is meant for **enterprise-level projects**
- ✓ Data scientists can use Metaflow for **end-to-end development** and **deployment** of their **machine learning models.**

METAFLOW

Great library support

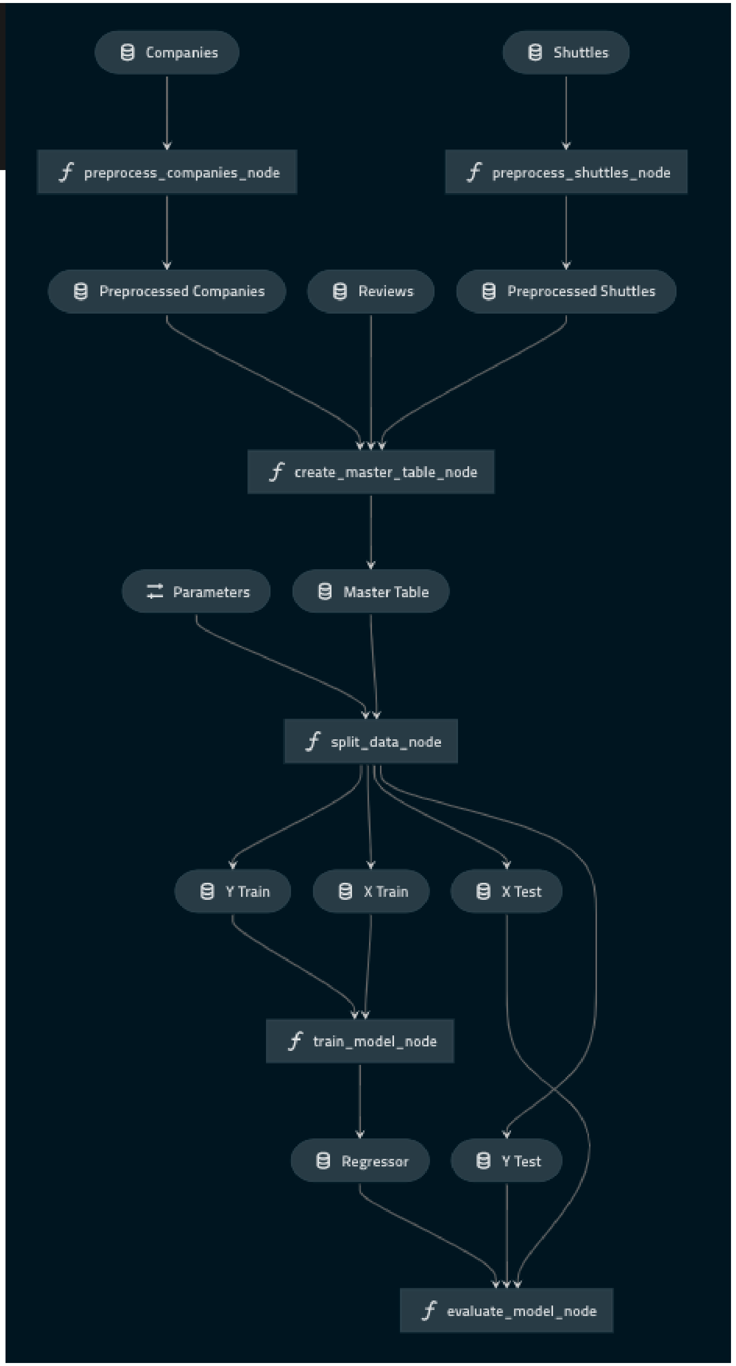
- ✓ Metaflow supports all popular **data science libraries**, such as **TensorFlow** and **scikit-learn.**
- ✓ Users can continue to use their **preferred tool.**
- ✓ Metaflow supports both **Python** and **R** programming languages.
- ✓ This provides flexibility in terms of **library** and **package choice.**

Powerful version control toolkit

- ✓ **Metaflow automatically versions** and keeps track of all **experiments**, so nothing important is lost.
- ✓ Users can **inspect** the results of all **experiments in notebooks.**
- ✓ Metaflow was specifically **designed for large-scale machine learning development**.
- ✓ The solution is powered by the **AWS cloud** and **includes built-in integrations** for **storage**, **compute**, **and machine learning services.**
- ✓ There is no need to **rewrite** or **change code** to use these services.

Kedro

- ✓ **Kedro is a Python framework** for **machine learning engineers** and **data scientists** to create **reproducible** and **maintainable** code.
- ✓ It helps to organize the **data pipeline** and **makes ML project development** more efficient.
- ✓ Kedro limits the need for **code rewrites,** allowing more time to focus on **robust pipelines.**
- ✓ It also helps teams establish **collaboration standards** to limit **delays** and **build scalable**, **deployable projects.**



Project templates

- ✓ Usually, a lot of time is required to understand how to set up an **analytics project**.
- ✓ Kedro provides a **standard template which saves time.**

Data management

- ✓ Kedro will help you **load** and **store data** to stop being alarmed about the **reproducibility** and **scalability** of your code

Configuration management

- ✓ Kedro is a necessary tool when working with **complex software systems**, it helps to avoid serious **reliability** and **scalability problems**.
- ✓ Kedro promotes a **data-driven approach** to ML development.
- ✓ It maintains **industry-level standards** while decreasing operational **risks for business**.

ZenML

- ✓ **ZenML is an open-source platform** for **machine learning (ML)** and **artificial intelligence (AI)** development.
- ✓ It aims to **simplify** and **automate** the entire ML development process, from **data preparation** and **feature engineering** to **model training** and **deployment**.
- ✓ ZenML provides a unified interface to work with multiple ML frameworks such as **TensorFlow**, **PyTorch**, and **scikit-learn**, and allows users to easily switch between them.
- ✓ It provides a set of tools for **data pipeline management**, **versioning**, and **monitoring**, allowing users to easily **track** and **reproduce their experiments**.
- ✓ ZenML is a **flexible**, **reusable**, and **extensible platform** to **develop**, **test**, and **deploy** Machine Learning models.

Psitron

Preprocess data

- ✓ ZenML helps you convert **raw data** into **analysis-ready data.**

Train your models

- ✓ ZenML platform uses **declarative pipeline configs** as a convenient tool for training.
- ✓ The configs enable easy **switching between on-premise** and **cloud environments**.

Conduct split testing

- ✓ The creators of ZenML claim that the platform's key benefits are **automated tracking of the experiments.**
- ✓ The platform also guarantees comparability between experiments.

Evaluate the results

- ✓ ZenML focuses on making ML development reproducible and straightforward for both individual developers and large teams.

- ✓ This framework frees you from all the troubles of delivering machine learning models with traditional tools.

- ✓ If you struggle with providing enough experiment data that proves the reproducibility of results, want to reduce waste and make the reuse of code simpler, ZenML will help.

MLRun

- ✓ **MLRun is a serverless**, open-source platform for **machine learning** and **data science development.**
- ✓ It is built on **top of Kubernetes** and allows **easy management** and **scaling of machine learning workloads.**
- ✓ It provides a unified interface for different **machine learning frameworks** and tools for **data pipeline management**, **versioning**, and **monitoring**.
- ✓ The platform supports distributed **training** and **inference** and allows easy deployment to various environments.
- ✓ It allows easy, cost-effective management of machine learning workloads without dedicated infrastructure.

Psitron

MLRun has a layered architectures

Feature and artifact store

- ✓ This layer helps you to handle the **preparation** and **processing** of data and store it across **different repositories.**

Elastic serverless runtimes layer

- ✓ **Convert simple code** into **microservices** that are easy to **scale** and **maintain**.
- ✓ It is compatible with standard runtime engines like **Kubernetes jobs**, **Dask**, and **Apache Spark.**

Automation layer

- ✓ Pipeline automation tool helps with **data preparation** and **testing**
- ✓ Helps with **real-time deployment**
- ✓ Allows the user to focus on **training the model** and **fine-tuning the hyperparameters**
- ✓ User supervision is needed to create a **state-of-the-art ML solution.**

Central management layer

- ✓ **Unified dashboard to manage entire workflow**
- ✓ Convenient **user interface**, **CLI**, and **SDK**
- ✓ Ability to **write code once** and **run on different platforms**
- ✓ Tool manages **build process**, **execution**, **data movement**, **scaling**, **versioning**, **parameterization**, **output tracking**, and **more**.

CML

- ✓ **CML (Continuous Machine Learning)** is a library that helps with the **continuous integration** and **delivery of machine learning** projects.
- ✓ It was developed by the creators of **DVC**, an open-source library for **versioning ML models** and **experiments**.
- ✓ Together with **DVC**, **Tensorboard**, and **cloud services**, CML aims to make it easier to **develop** and **implement ML models** into **products**.

Automate pipeline building

- ✓ CML aims to automate some of the tasks of **ML engineers**
- ✓ It includes automation of **training experiments**, **model evaluation**, **datasets** and their additions

Integrate APIs

- ✓ CML is a library that supports **GitFlow for data science projects.**
- ✓ It allows **automatic generation of reports**.
- ✓ It hides the complex details of using external services like **cloud platforms** and infrastructure tools like **DVC**, **docker**, and **Terraform**.
- ✓ It helps in managing the infrastructural aspect of ML projects.

Apache Airflow

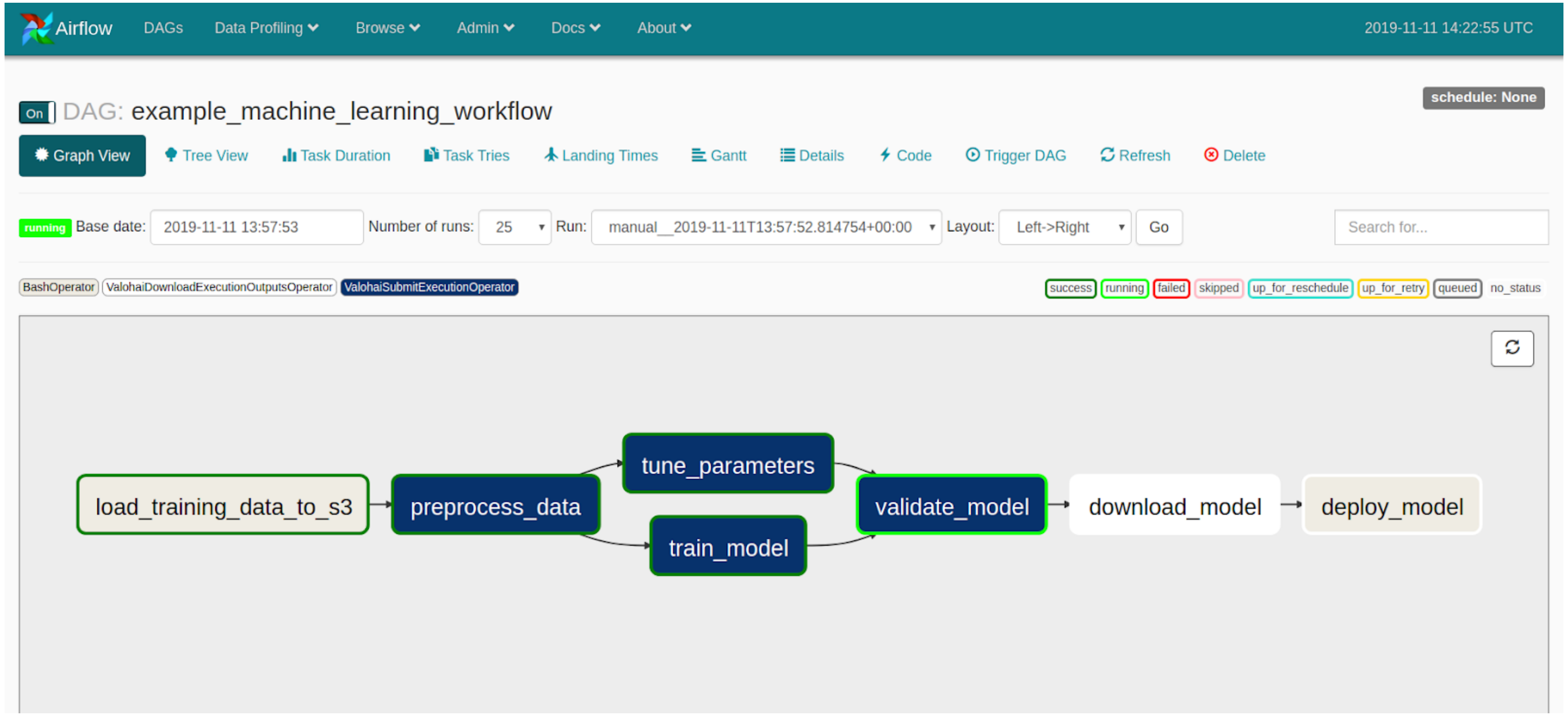
✓ **Apache Airflow**: Open-source tool for creating, scheduling, and tracking workflows.
✓ **DAGs**: Workflows structured as directed graphs; tasks are nodes, dependencies are edge
✓ **Origin**: Created by Airbnb in 2014, later made open-source.

Key Features:

- Directed Acyclic Graphs (DAGs)
- Operators
- Schedulers
- Web Interface
- Extensible
- Dynamic Configuration
- Integration

Apache Airflow for ML pipelines

- **Airflow for ML Pipelines**: Manages and automates ML pipelines.
- **ML Pipelines**: Sequences of data processing and modeling.
- **Automation**: Airflow automates and monitors pipeline execution.



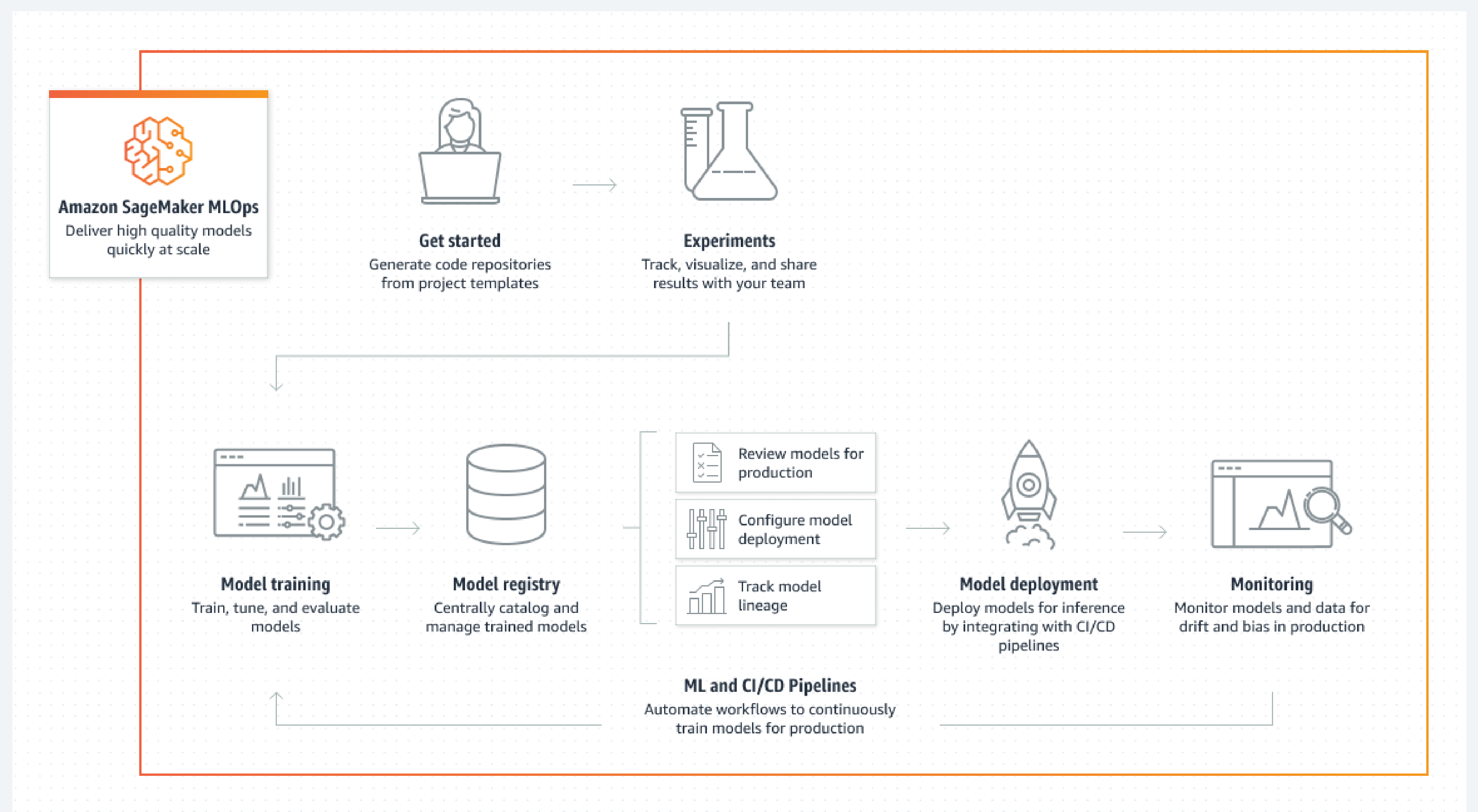
Apache Airflow

MLOps Architectures

Architectures - Cloud Native tools

AWS for MLOps

Amazon SageMaker for MLOps



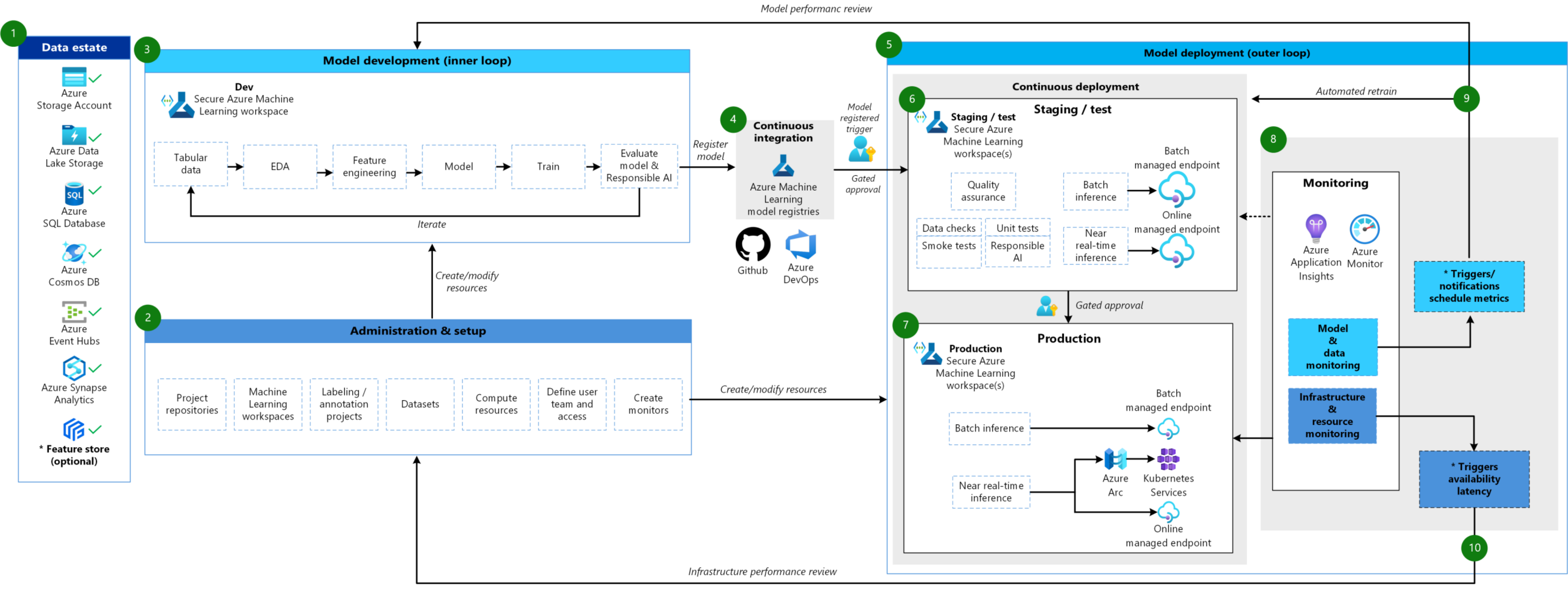
AWS for MLOps

Amazon SageMaker for MLOps

Amazon SageMaker is a fully managed service provided by Amazon Web Services (AWS) for **building**, **deploying**, and **managing machine learning models**.

- ✓ It can be used as a part of an **MLOps pipeline** to streamline the deployment of machine learning models in a cloud environment.
- ✓ SageMaker also provides a **variety of tools** for **monitoring** and **tuning models**, including **automatic model tuning and real-time monitoring of model performance**.
- ✓ SageMaker provides **Jupyter notebooks** for developing models and integrates with other AWS services for **data storage**, **warehousing**, and **container management.**

Azure for MLOps



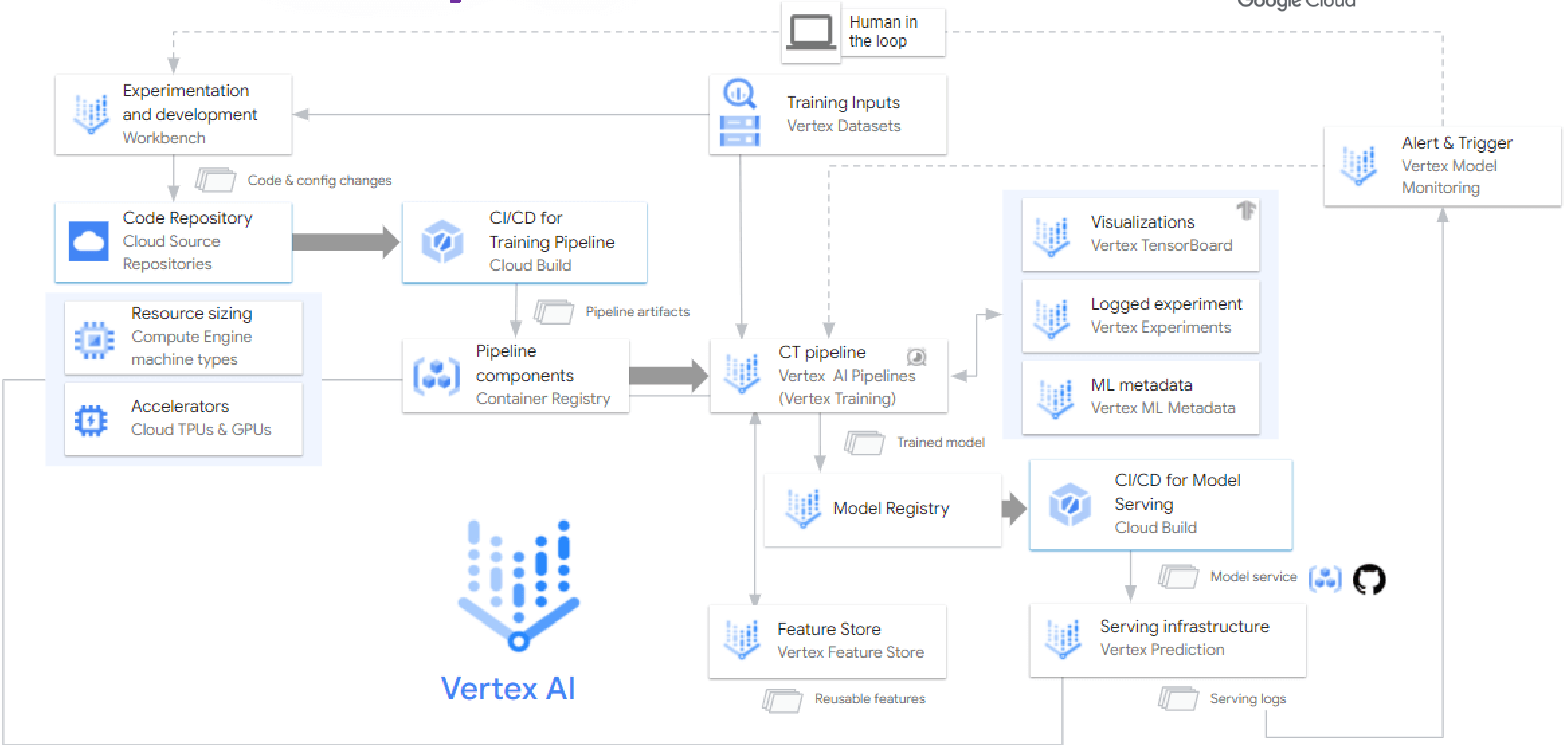
Legend: Data engineer | Data scientist | Machine learning engineer | Infrastructure team

* Future accelerator implementation

✓ Recommended practice

Microsoft Azure

GCP for MLOps



Google Cloud

Psitron

Comparison among cloud-native tools

Name of Services	Additional info	Amazon SageMaker	GCP Vertex AI	Azure Machine Learning
Notebook Support		✅	✅	✅
Jupiter lab	Azure has custom interface	✅	✅	❌
Language support (Notebook, Experiment, pipelines, custom model, endpoint etc)	AWS Definitely has more support like MXNet,TF, PyTorch Azure has direct integration with VS code , so we can easily work on local setup GCP in comparatively new , it has basic version but has direct integration with git	Python R TensorFlow PyTorch Apache MXNet Spark Keras	Python 3 Python (conda encroot)	Notebook (*ipynb) Python(*.py) R(*.R) Bash(*.sh) Text(*.txt) other

Psitron

	Additional Info	Amazon SageMaker	GCP Vertex AI	Azure Machine Learning
Compute Instance	Azure gives more control over compute	At set-up time	At set-up time	Can change via notebook
Workflow Pipelines Support		✅	✅	✅
Studio/Low Code/GUI and Drag & Drop support	Azure is a pioneer AWS recently started and GCP yet to start	✅	❌	✅
In-built Feature store	GCP looks has advanced here, based on feast.io	✅	✅	✅
Automatic ML		✅ Auto Pilot	✅ Auto ML	✅ Automated ML
Label the data		✅ Ground Truth	✅ Labeling Tasks	✅ Data Labeling

	Additional Info	Amazon SageMaker	GCP Vertex AI	Azure Machine Learning
GPU Support		✅ Framework Optimized	✅ NVIDIA Tesla based	✅ Can install GCP drivers in compute based
TXF(Transfer Learning Framework) support		❌ Custom Docker	✅	❌ Custom Docker
Real-time endpoint		✅	✅	✅
Offline Job		✅ Batch Transform	✅ Batch Prediction	✅ Pipeline
In-built support for IoT		✅ Neo	❌	❌ Possible via IoT Edge
A/B testing		✅ Traffic Routing	✅ Traffic Routing	❌ Custom Code
Reinforcement learning		✅	✅	✅

	Additional Info	Amazon SageMaker	GCP Vertex AI	Azure Machine Learning
Kubernetes Support		✅	✅	❌
Multiple Model on Same Endpoint to save cost		✅	❌	❌
Automatic Model Debugging , tuning		✅	❌	❌
Auto Scaling, edge optimization of endpoint		✅	✅	✅
Model Monitoring (Data quality, model quality, data drift, bias etc)	AWS is ahead in the game followed by Azure & GCP	✅	✅	✅
Responsible AI	Multiple different matrices available	✅ Model Explainability	✅ Explainable AI	✅ Explainable interpretability

Cost-benefit approach of each architecture- pay as you go model

aws

https://aws.amazon.com/sagemaker/pricing/

Azure

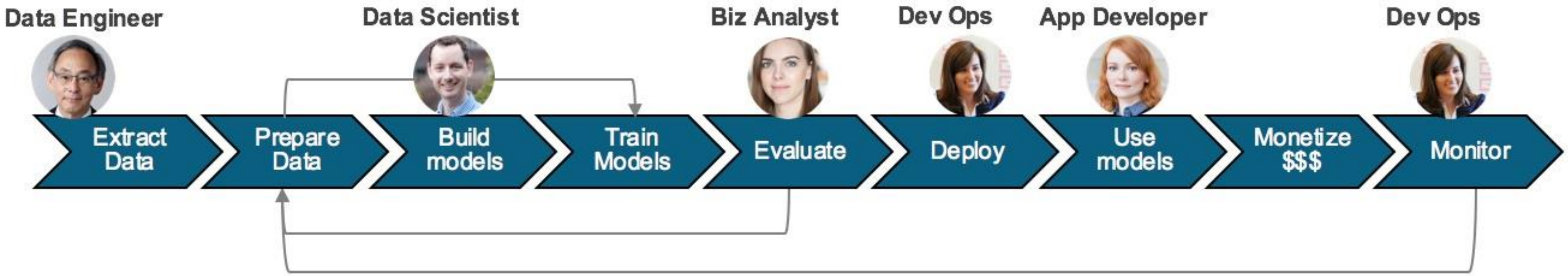
https://azure.microsoft.com/en-in/pricing/details/machine-learning/

Google Cloud

https://cloud.google.com/vertex-ai/pricing#vertex-ai-pricing

Different Roles involved in MLOps

A single person cannot answer all the above questions. Hence, a matured ML team typically consists of the following:



- Data Analysts
- Data Engineers
- Data Scientist
- Research/Applied Scientists
- ML Engineers
- Developers