



3.1 Orchestration and Deployment

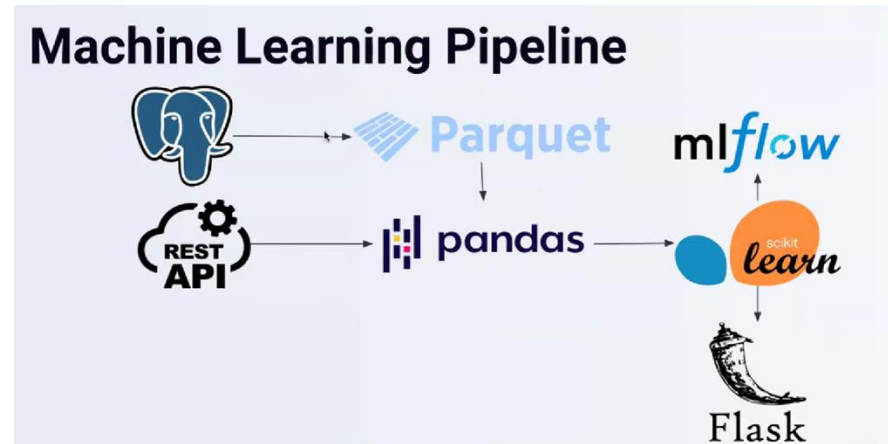
POSGRAD
MNA


A man with a beard and glasses, wearing large white headphones, is seated at a desk. He is wearing a blue button-down shirt. His hands are on a desk, and he appears to be working. In the background, there is a bookshelf filled with books. The image has a teal overlay with abstract geometric shapes in the corners. The word "Orchestration." is written in white text on the left side.

Orchestration.

What is ML Orchestration?

Machine learning (ML) orchestration refers to the process of automating the deployment, management, and monitoring of machine learning models at scale. It coordinates the numerous facets and processes within an ML pipeline, such as data preprocessing, feature engineering, model training, validation, and deployment.






Machine learning orchestration automates and streamlines ML model development, helping businesses speed up time-to-market and improve ML operations' accuracy and efficiency.

Orchestration platforms support various stages, including:

- Version and data control
- Model creation, testing, and validation
- Model deployment and operation
- Automated monitoring and alerting
- Integration with other data and application services

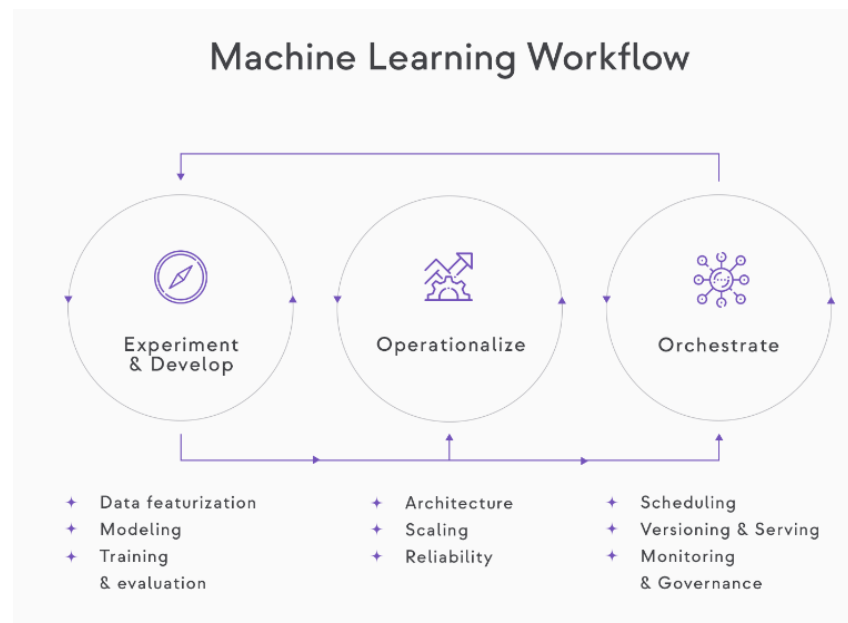
These platforms free data scientists and engineers from infrastructure management, allowing them to focus on model development and improvement.



How a typical ML pipeline looks

If we want our ML pipeline to work well in production, its components must be orchestrated. The dependencies and data flow between them must be coordinated reliably, regularly, and observably.

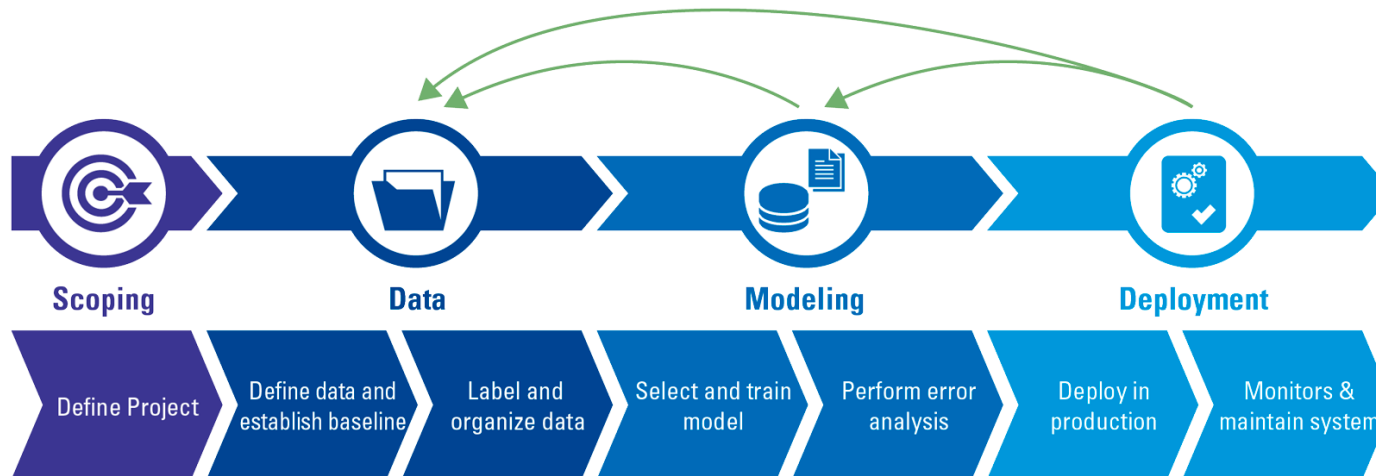
This practice is known as machine learning orchestration (MLOx).



Operationalization and System Design

When moving to production, the following aspects must be considered:

- Architecture and data flow
- Scaling and scalability.
- Availability and uptime
- Security.



Orchestration Elements.

The orchestration tasks involve the following elements:

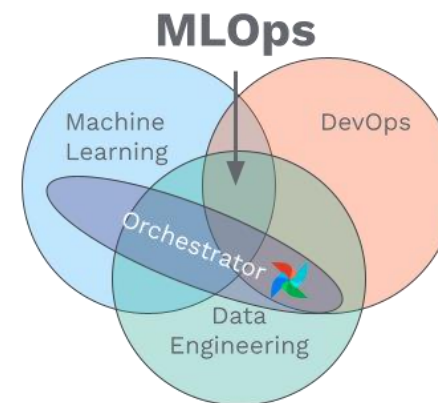
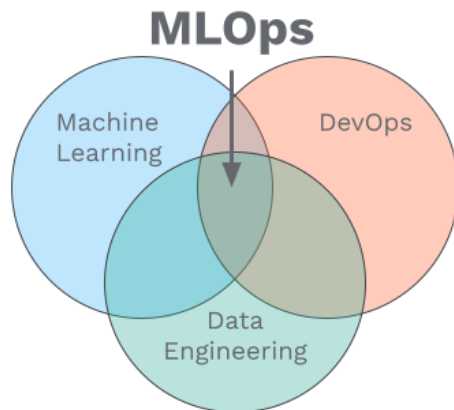
- Scheduling
- Syncing
- CI/CD and testing
- Saving and versioning
- Retention and cleanup schedule
- Monitoring
 - Data Drift
 - Concept Drift
 - Model Performance
- System health and reliability tracking
- Governance and Observability.

What is the difference between Orchestration and MLOps?

MLOps is a set of practices that combines Machine Learning, DevOps, and Data Engineering. It aims to reliably and efficiently deploy and maintain ML systems in production.

The orchestrator is an essential component in any MLOps stack, responsible for running your ML pipelines. The orchestrator provides an environment set up to execute your pipeline's steps.

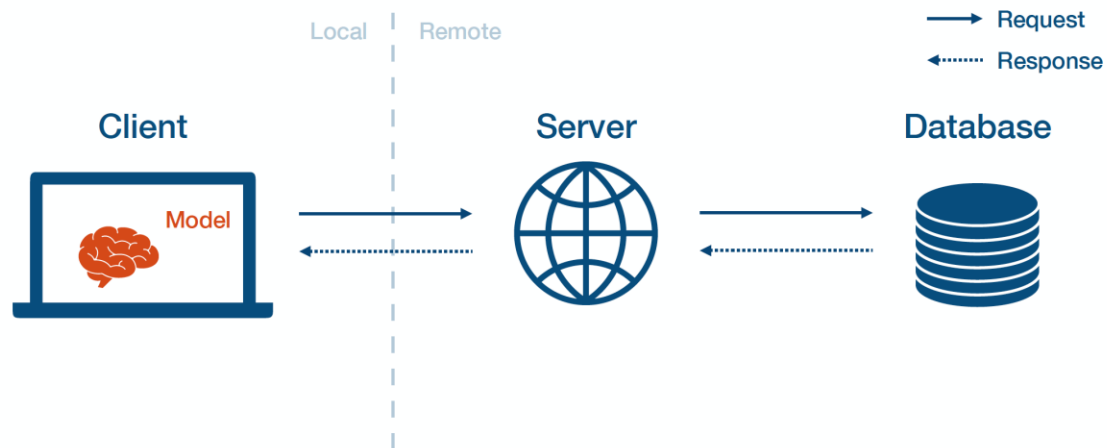
The orchestrator's role is to ensure that the various processes in the workflow execute on schedule, in the correct sequence, and deal appropriately with failure.



A man with a beard and glasses, wearing large white headphones, is seated at a desk. He is wearing a blue button-down shirt. His hands are on a desk, and he appears to be working. In the background, there is a bookshelf filled with books. The image has a teal overlay with abstract geometric shapes in the corners and a dotted pattern in the bottom right.

Deployment.

Introduction



Model deployment refers to integrating a trained machine-learning model into a **real-world system or application** to generate predictions or perform specific tasks automatically.

We may see that some models could have excellent performance in their development phase but once those are put into production, we could see issues like data skew, scalability inconsistencies and the sudden decrease of accuracy.

Important considerations.



ML pipeline deployment will depend on the following:

- System environment.
- Model type.
- DevOps practices.
- Scalability requirements.

Deployment is just as crucial as the development phase because ML pipeline deployment can only bring value to the organization once it faces real scenarios and deals with actual data.

ML Pipeline Deployment Criteria.



Before deploying a model, there are a couple of criteria that your ML pipeline needs to achieve before it's ready for deployment:

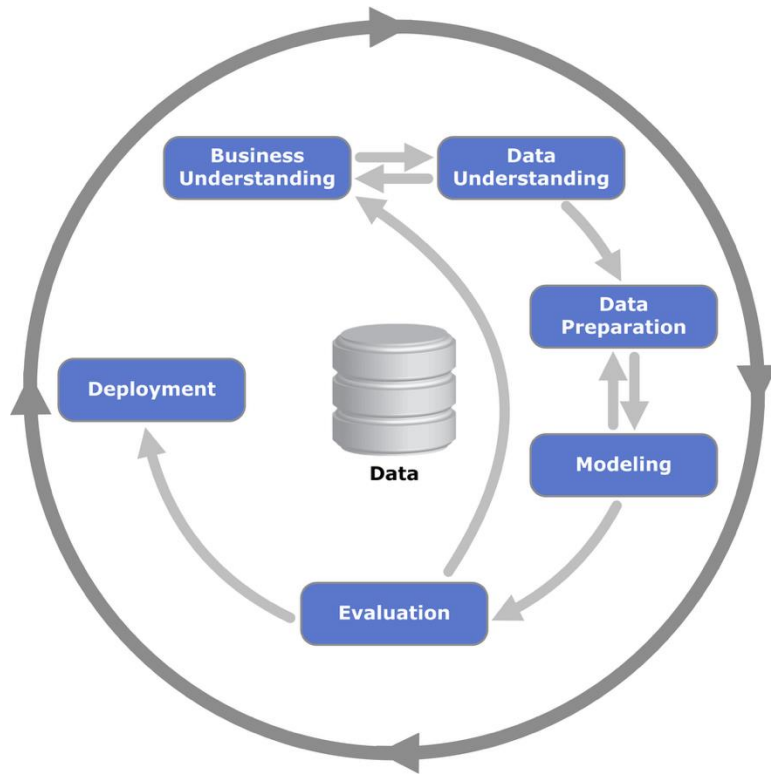
- **Portability:** This is the ability of your software to be transferred from one machine or system to another.
- **Scalability:** This refers to how large your model can scale. A scalable model is one that doesn't need to be redesigned to maintain its performance

ML Pipeline Architecture

As we have seen, there are four layers in an ML pipeline.

- Data Layer: It provides access to all the data sources the model will require.
- Feature Layer: Is responsible for generating feature data in a transparent, scalable, and usable manner.
- Scoring Layer: It transforms features into predictions. Scikit-learn is most commonly used and is the industry standard for scoring.
- Evaluation Layer: It checks the equivalence of two models and can be used to monitor production models.

ML Deployment Process



As we have seen during the course, the process of model deployment involves several steps, but we can redefine the Deployment stage to include the next steps:

- Model serialization.
- Deployment environment preparation
- Deployment API building
- Deployment testing and validation
- ML pipeline deployment

Model Serialization.



Once you have chosen the best model, it's crucial to serialize and save it for further use. Serialization converts the model into a binary format that can be stored on disk. The joblib library is commonly used for model serialization in Python.

Deployment Environment Preparation.



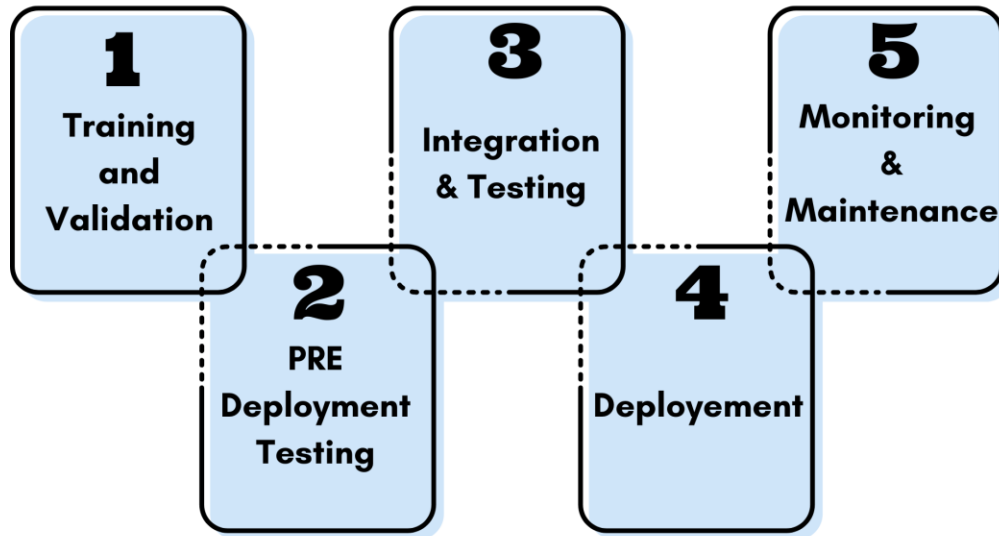
To deploy your model, you need to set up the deployment environment, this means that you need to create a virtual environment that manage the dependencies and install the required libraries. This ensures that your deployment environment is isolated and has the required packages installed.

Deployment API Building.



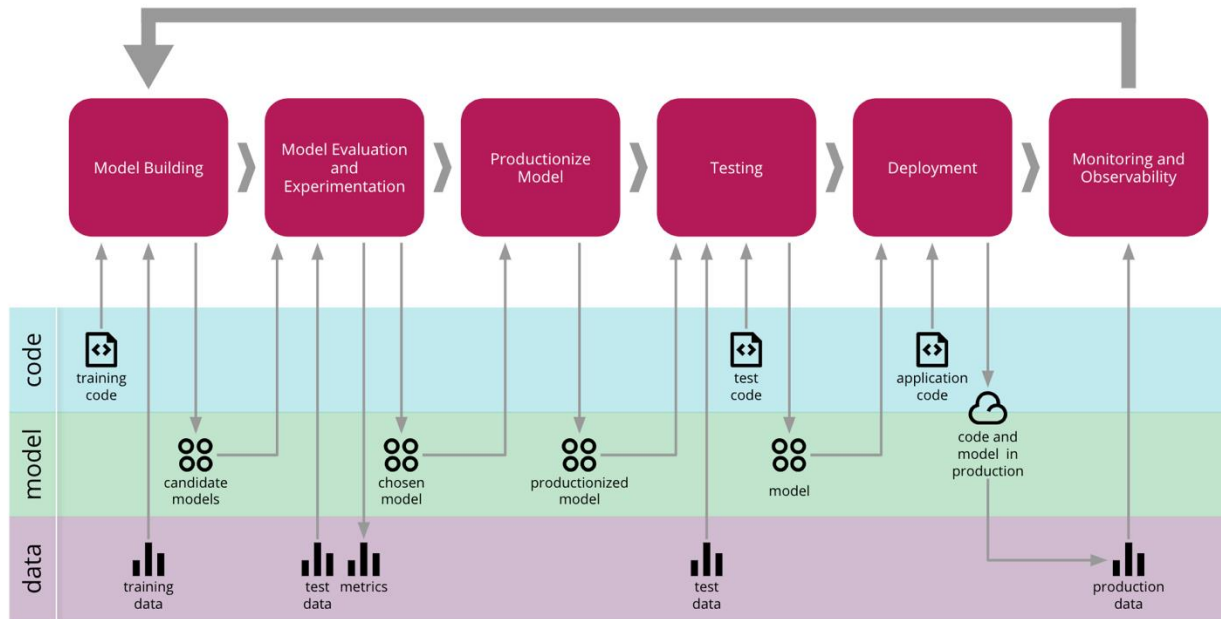
You must create a python script that defines the API endpoint that will handle the incoming request, preprocess and make the predictions.

Deployment Testing and Validation



This step involves sending sample requests to the API using tools like requests or curl and verifying the output against expected results during model monitoring. You must further compare the predictions made by the deployment with those made during model development to ensure consistency and accuracy across multiple models.

ML Pipeline Deployment

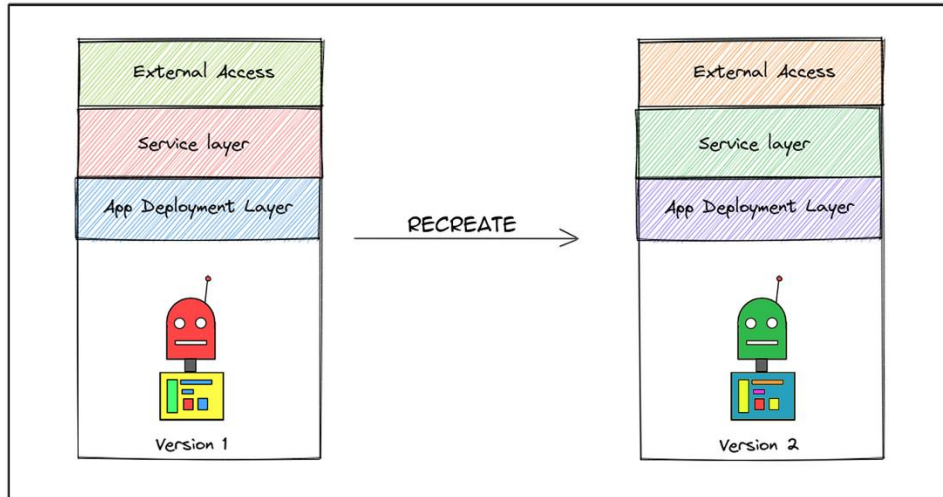


You must choose a server or cloud platform to deploy your ML application, such as Heroku, AWS, or Azure, as they provide the necessary infrastructure to host your deployment. You must also configure the server/cloud environment to handle incoming requests and route them to the API endpoint you defined in the previous step. When deploying your ML model in a production environment, you must always follow best practices for security, scalability, and availability.



Deployment Methods and Patterns.

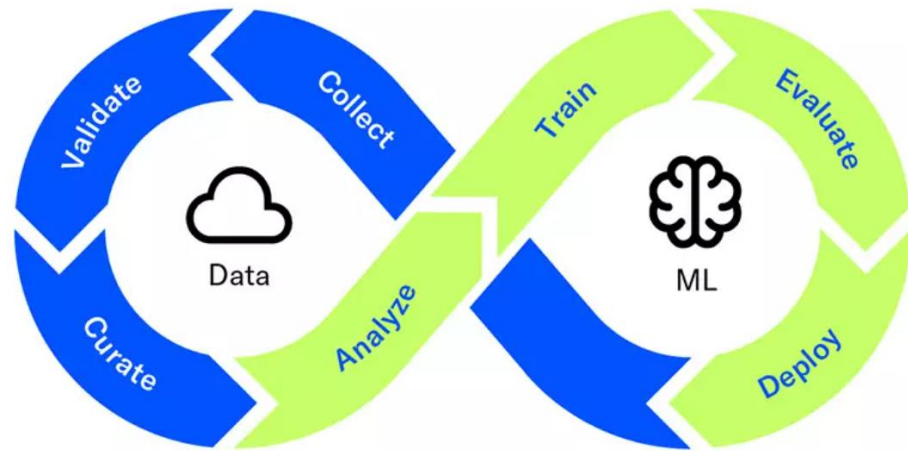
Deployment Methods.



These are techniques employed by DevOps teams to successfully launch a new version of the software solution they provide.

For Machine Learning pipelines, it refers to the method used to make the up-to-date model available for its users.

Deployment Patterns.



These are automated methods for introducing new application features to your users.

Deployment patterns enable you to test new features with a small group of users before making them available to everyone.

A man with a beard and glasses, wearing large white headphones, is seated at a desk. He is looking towards the right side of the frame. He is wearing a blue button-down shirt. In the background, there is a bookshelf filled with books. The image has a teal overlay with decorative geometric shapes: a teal circle and a blue circle in the top left, and a large teal circle with concentric dotted lines in the top right. A blue circle with a dotted pattern is in the bottom right.

Deployment Methods.

One-off.

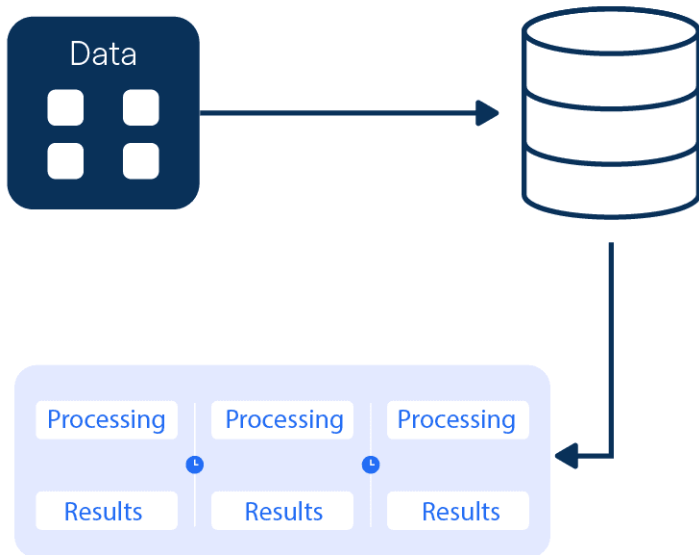


Sometimes, you don't need to continuously train a machine learning model to deploy it; it is only needed once or periodically.

In this case, the model can simply be trained ad-hoc when it's needed and pushed to production until it drifts enough so you need to fix it.

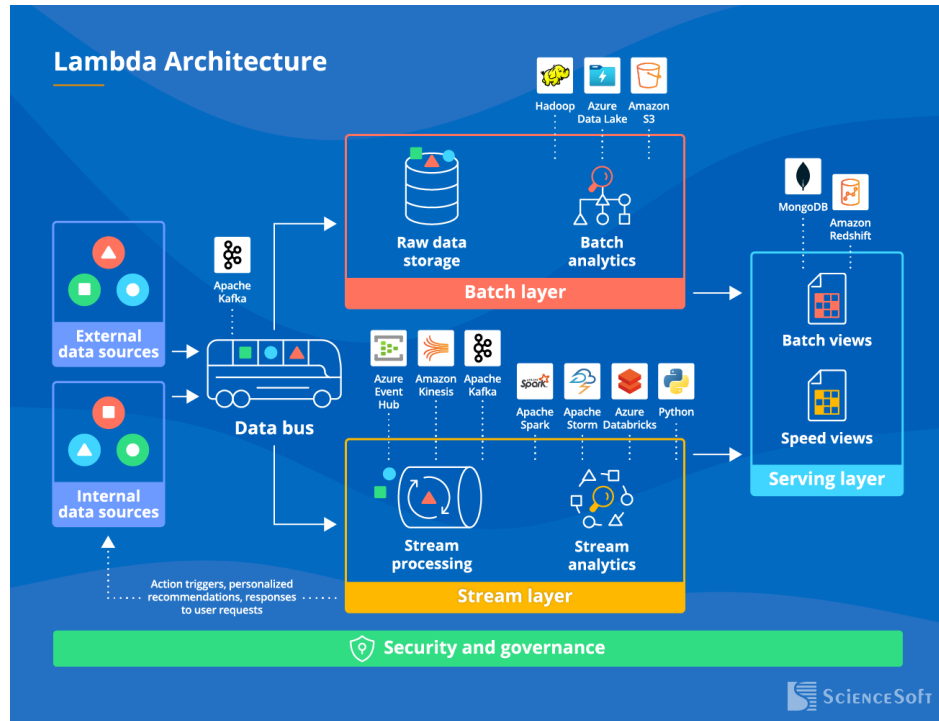
Batch.

Batch Processing



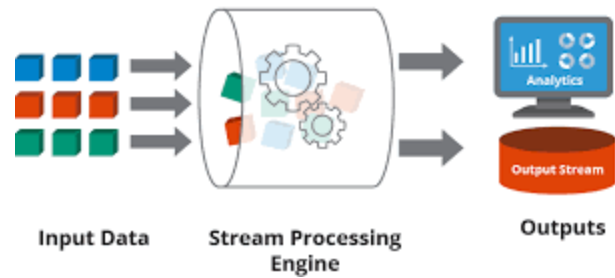
Batch training enables you to maintain an up-to-date version of your model continuously. This scalable approach uses a subset of the data for each update, avoiding the need to process the entire dataset every time. It's ideal if you regularly rely on the model but don't require real-time predictions

Real time.



In certain situations, real-time predictions are essential, such as identifying whether a transaction is fraudulent. This can be achieved by using online machine learning models, like linear regression with stochastic gradient descent.

Streaming.



Streaming deployment supports a more asynchronous workflow, allowing user actions to trigger prediction computations.

To implement this, the process is connected to a message broker, and the model handles requests as they become available.

This method decreases server load and optimizes computational resources by using an efficient queuing system.

Edge.



Edge deployment allows for faster model responses and offline predictions.

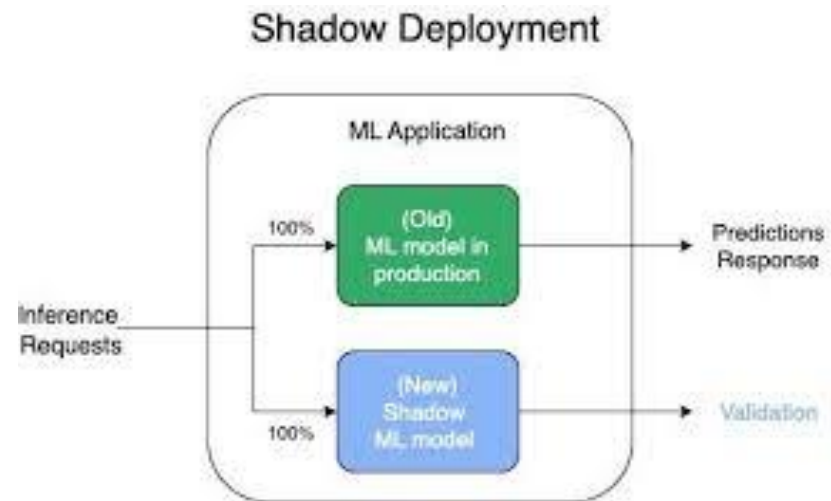
In this approach, the model is deployed directly on a client device, such as a smartphone or IoT device.

To make this feasible, models typically need to be compact enough to run on a smaller device.

A man with a beard and glasses, wearing large white headphones, is seated at a desk. He is looking towards the right side of the frame. He is wearing a blue button-down shirt. In the background, there is a bookshelf filled with books. The image has a teal overlay with geometric shapes: a large teal circle in the top left, a blue circle in the top right, and a purple circle in the bottom right. The text "Deployment Patterns." is overlaid in white, bold font on the left side of the image.

Deployment Patterns.

Shadow Deployment.

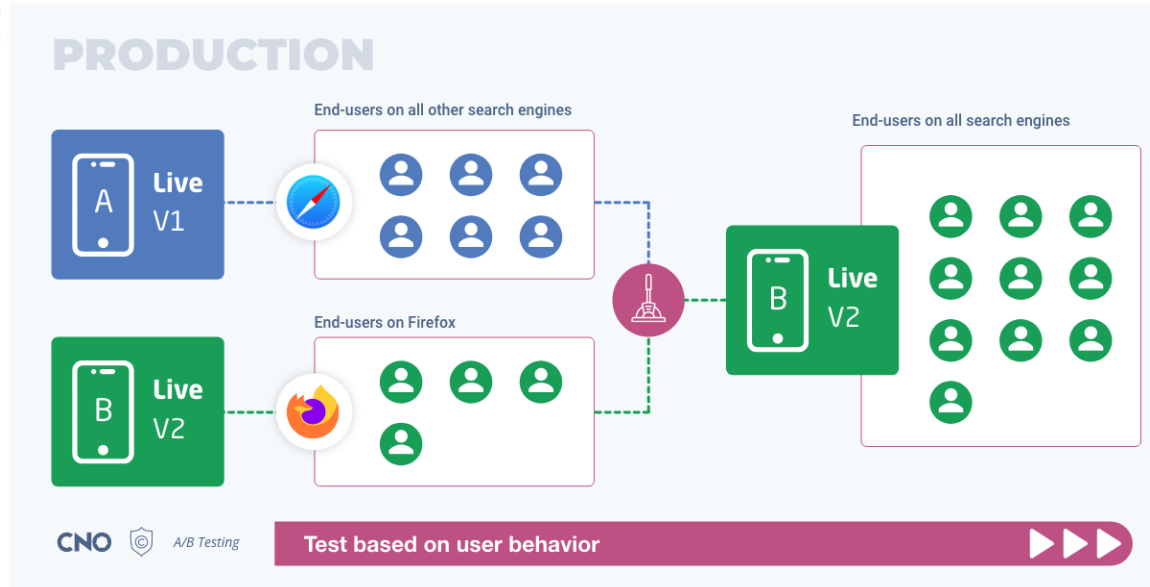


The new model is deployed with additional features alongside the active model, known in this case as a shadow model.

The shadow model processes all requests like the live model, but it isn't publicly accessible.

This approach allows for evaluating the shadow model on real-world data without disrupting the services provided by the live model.

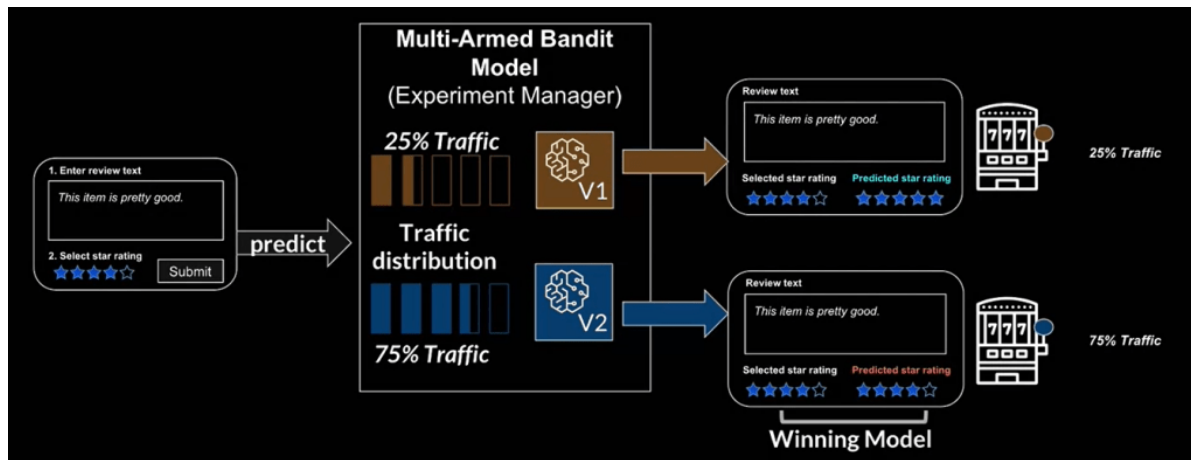
A/B Testing.



This is a data-driven testing method used to compare two models, A and B, to determine which performs better in a controlled setting.

Through A/B testing, data scientists can assess and select the optimal website design based on user data. The two models have slight feature differences and are targeted at different user groups.

Multi-Arm Bandits.

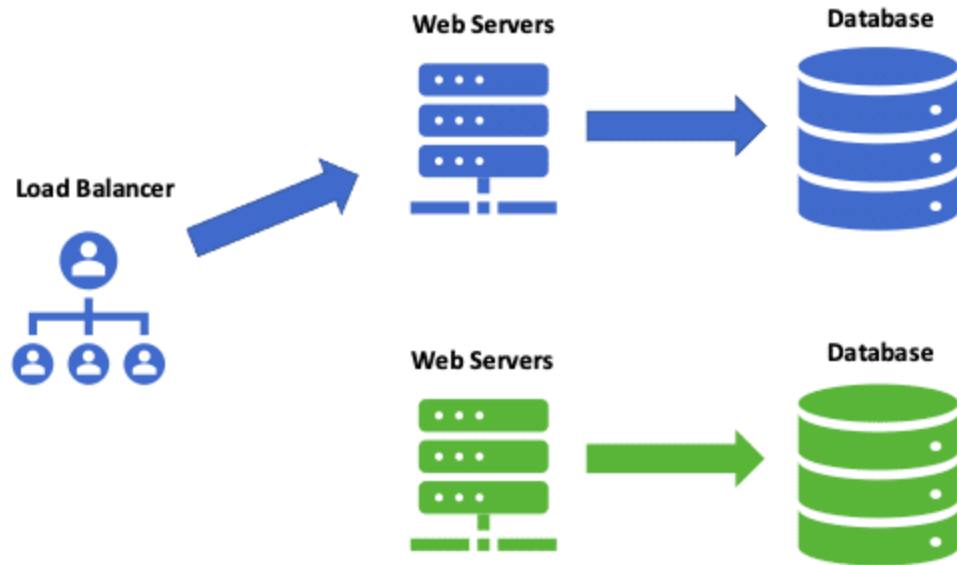


The Multi-Armed Bandit (MAB) is an advanced approach to A/B testing.

Drawing inspiration from reinforcement learning, it aims to balance exploration and exploitation to maximize the reward function.

MAB uses machine learning to optimize key performance indicators (KPIs), deploying the model that delivers the highest KPI across all users.

Blue-Green Deployment.



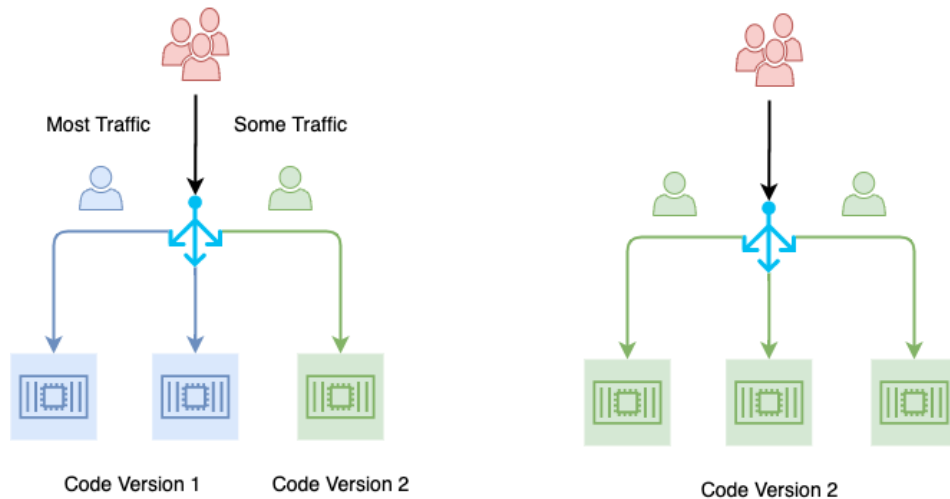
The Blue-green deployment strategy uses two production environments rather than only models.

The blue environment hosts the live model, while the green environment contains the updated model version. The green environment serves as a staging area with new features.

Once updates are tested and validated, user requests are redirected to the new environment, effectively replacing the old model with the new one.

Canary Testing.

Canary Deployment



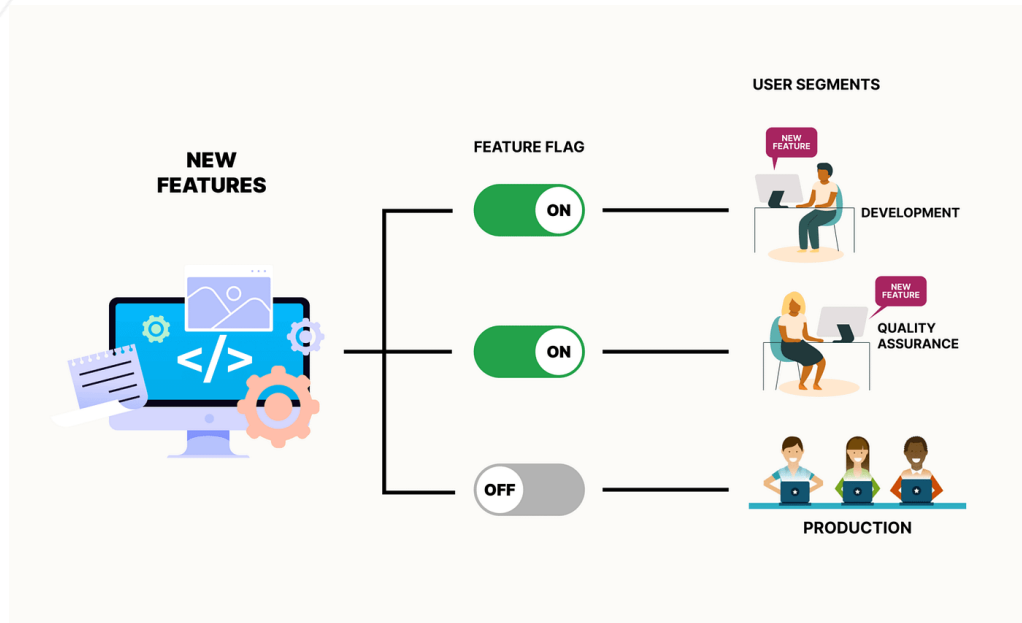
The canary release is a technique for identifying potential issues before they impact all users.

In a canary deployment, the new model version is released by gradually increasing the number of users.

This deployment strategy uses real users to test the new model.

Consequently, bugs and issues can be detected before rolling out the model globally.

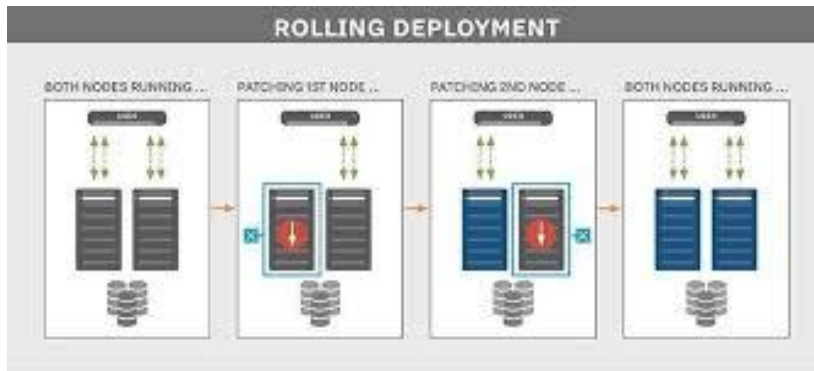
Feature Flag.



The feature flag is a technique that allows developers to merge code into the main branch while keeping specific features inactive.

The goal is to keep the feature dormant until it's fully ready, enabling collaboration on various ideas and iterations. Once the feature is complete, it can be activated and deployed.

Rolling Deployment.



The Rolling deployment is a strategy that gradually updates and replaces the older version of the model.

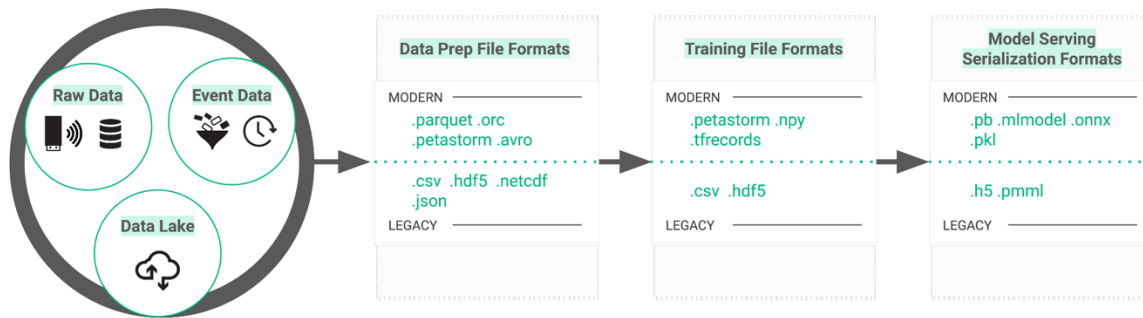
Rolling updates involve updating all instances of your model or application.

One by one the older pods are replaced with newer ones. There is no downtime with this approach.

A man with a beard and glasses, wearing large white headphones, is seated at a desk. He is looking slightly to the right. He wears a blue button-down shirt. In the background, there is a bookshelf filled with books. The image has a teal overlay with decorative geometric shapes: a dark teal shape in the top left, concentric dotted circles in the top right, and a purple dotted circle in the bottom right.

Model Packaging and Serving.

What is model packaging?

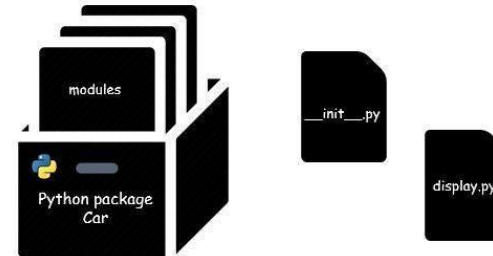


Model packaging bundles model artifacts, dependencies, configuration files, and metadata into a unified format for easy distribution, installation, and reuse. The primary goal is to streamline model deployment, ensuring a smooth transition to production.

Why is it important?

Model packaging ensures a machine-learning pipeline can be easily deployed and maintained in a production environment. When it is done correctly, it ensures:

- A well-packaged model is easy to install, which reduces the time and effort required for deployment.
- Ensures reproducibility across different environments, providing consistent results.
- Model packaging makes it easier to version models, track changes, and roll back to previous versions if needed.
- It includes clear code documentation that helps your peers understand how to use and modify it if required.



Challenges of creating a model package.

Packaging our ML pipelines is helpful when putting our ML models into production, however, it comes with some challenges as well:

- Model complexity.
- Environment diversity.
- Collaboration across teams.
- Dependency management.






There are a couple of considerations when packaging ML pipelines:

1. Web-based frameworks.

These are code libraries that makes web development faster, efficient and easier by providing common patterns for building reliable, scalable and maintainable web applications. Some examples of this are Flask, Django and FastAPI.

2. MLOps tools.

You can rely on tools that take some of the heavy lifting for this process, some examples are:

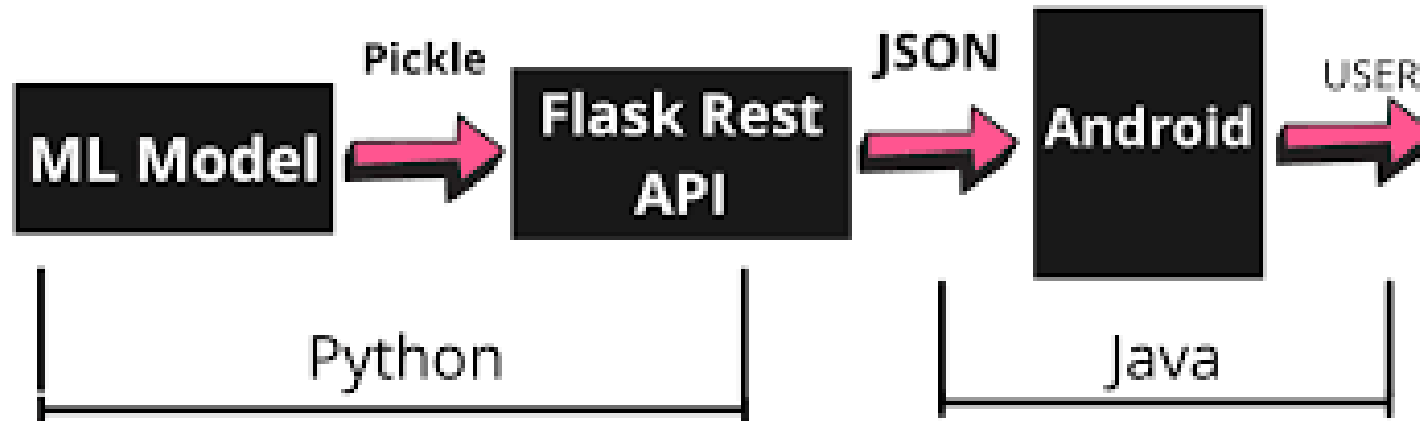
- a. Azure ML
 - b. GCP Kubeflow
 - c. AWS Sagemaker
 - d. ONNX
 - e. Tensorflow Serving
- 

A man with a beard and glasses, wearing large white headphones, is seated at a desk in an office environment. He is looking towards the right side of the frame. The background shows a blurred bookshelf filled with books. The image has a teal overlay with abstract geometric shapes in the corners and a pattern of concentric dotted circles on the right side.

Model Serving Options.

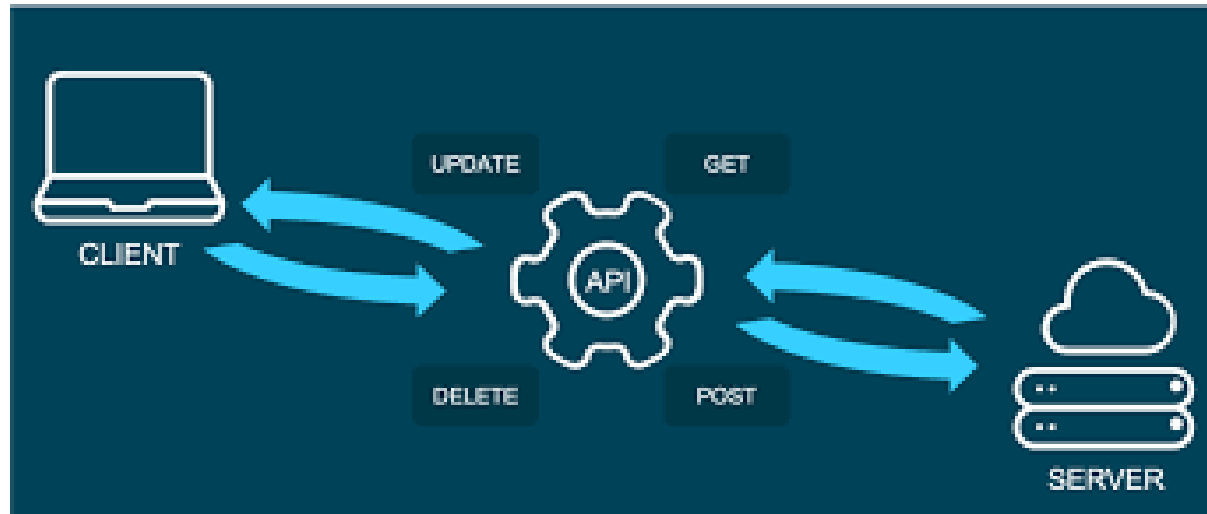
Model Embedded in the App.

The most straightforward way to integrate your model into an application is through model embedding. This approach involves embedding the model file within the application code, allowing the application to access it directly.



Model Served as an API.

This architecture separates the application from the model via an API, streamlining organizational workflows, model versioning, reuse, phased updates, and hardware independence. It enables the application and model server to use different hardware (CPU, GPU, FPGA) and scale independently based on demand.



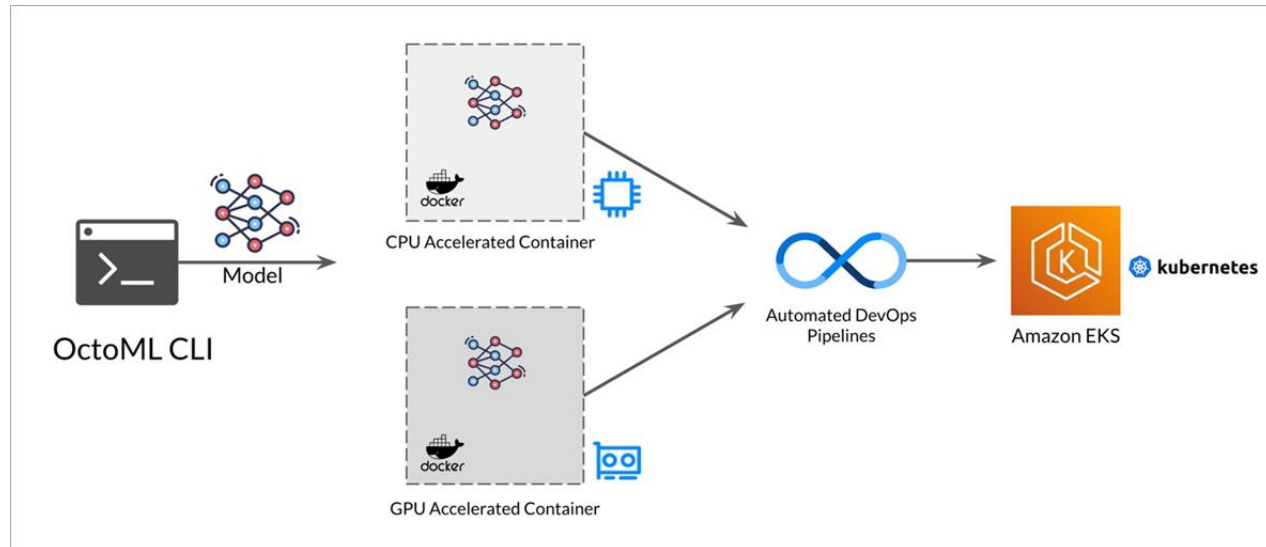
Model saved and used as a Library.

In this approach, the model is saved in a standardized format (such as TensorFlow SavedModel, PMML, PFA, or ONNX), making it programmatically accessible to any modern programming environment, language, or framework and enabling reuse across various applications.



Model saved in containers.

Containerization technologies like Docker and Kubernetes have transformed the way developers and organizations package, deploy, and manage applications. These tools have gained popularity due to their ability to package applications conveniently, eliminating concerns about dependencies and infrastructure. Their use has also expanded into machine learning, where developers leverage them to package and deploy ML models.



A man with a beard and glasses, wearing large white headphones, is sitting at a desk. He is wearing a blue button-down shirt. His hands are on a laptop. In the background, there is a bookshelf filled with books. The image has a teal overlay with geometric shapes: a large teal circle in the top left, a blue circle in the top right, and a purple circle in the bottom right. The text "Intro to Docker" is written in white on the left side.

Intro to Docker

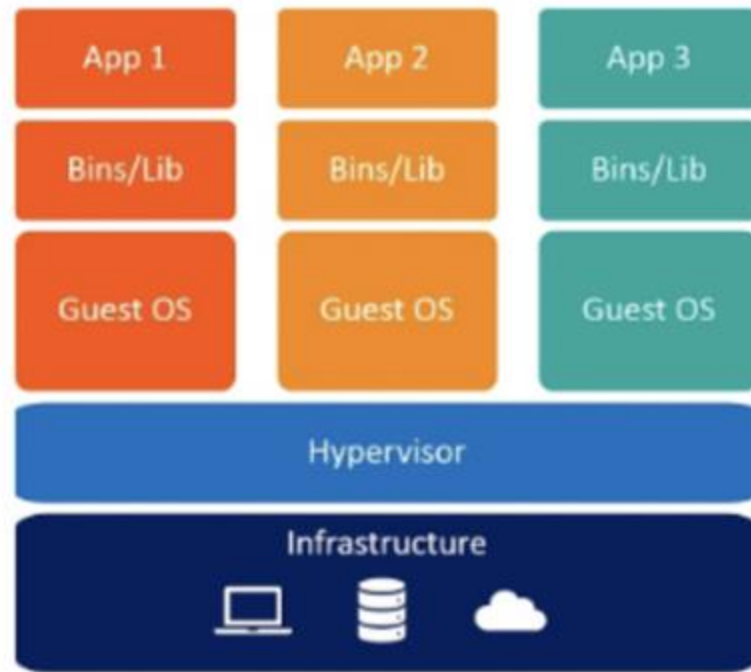
Docker Intro

Docker is a software platform that enables the rapid creation, testing, and deployment of applications. Docker packages software into standardized units called containers, which include everything needed to run the software, such as libraries, system tools, code, and runtime environments.

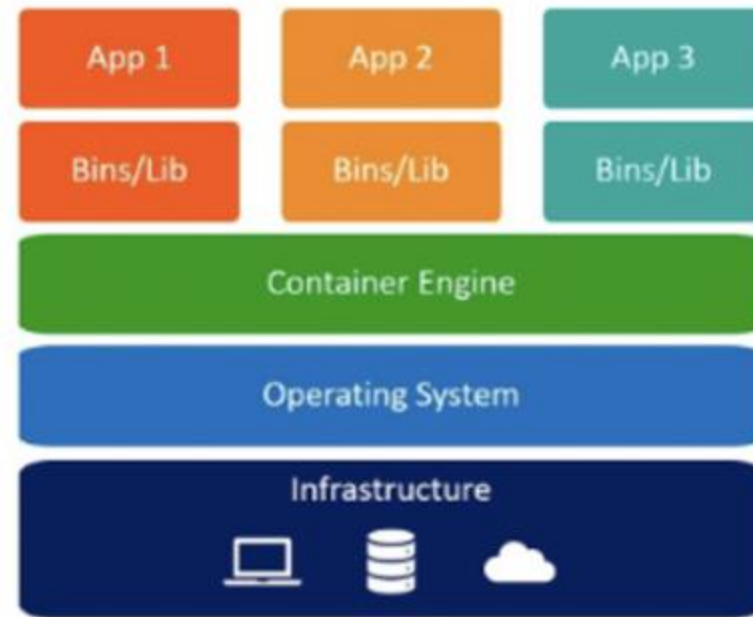
Docker provides a standardized way to run your code. It acts as an operating system for containers. Similar to how a virtual machine abstracts server hardware, containers abstract the server's operating system. Docker is installed on each server and offers simple commands to create, start, or stop containers.



Virtualization vs Containerization



Virtual Machines



Containers

Issues with Virtual Machines

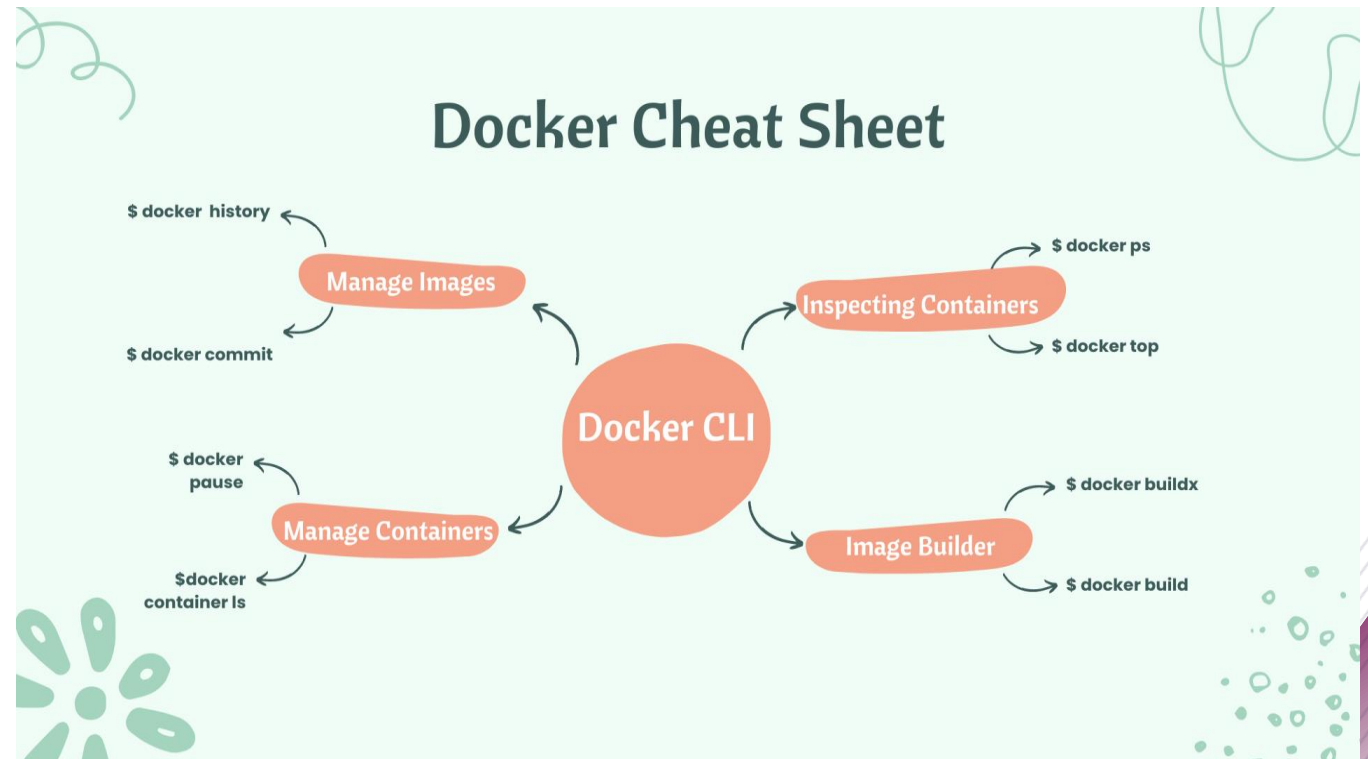
- Size: Duplicates standard files, leading to slow startup.
- Maintenance Cost: Requires maintenance, similar to any other computer.
- Multiple Formats: VID, VHD, raw, etc.

Advantages of containers

- Flexibility: Any application can be containerized.
- Lightweight: Containers use the host OS kernel.
- Portable: Run consistently on any machine.
- Scalable: Easily create more containers with identical functionality.
- Secure: A container can control access to system areas.

To package an ML model using Docker, follow the next steps:

1. Create a Dockerfile.
2. Build the Docker image.
3. Push the Docker image to a registry.
4. Pull the Docker image from the registry.
5. Run the Docker container.






References.



For further reference:

- [ML Model Packaging](#)
 - [Machine Learning Pipeline Orchestration](#)
 - [Machine Learning Orchestration vs MLOps](#)
 - [ML Workflow and Pipeline Orchestration Tools](#)
- 



D.R.© Tecnológico de Monterrey, México, 2024.
Prohibida la reproducción total o parcial
de esta obra sin expresa autorización del
Tecnológico de Monterrey.