

# Machine Learning Operations (MLOps)

## Introduction to MLOps

Zeham Management Technologies BootCamp  
by SDAIA  
September the 29<sup>th</sup>, 2024

# Introduction to MLOps

# Objectives

**By the end of this module, trainees will have a comprehensive understanding of:**

Master MLOps principles.

Distinguish between DevOps and MLOps Lifecycles.

Learn how to manage code changes.

Learn enhancing the ML pipeline and minimizing errors.

Explore different AutoML Examples.

Learn how to build and deploy ML models in production environments.



# Agenda



Introduction to MLOps



MLOps Lifecycle



Setting Up Local Environment



Version Control



Recent Trends in MLOps



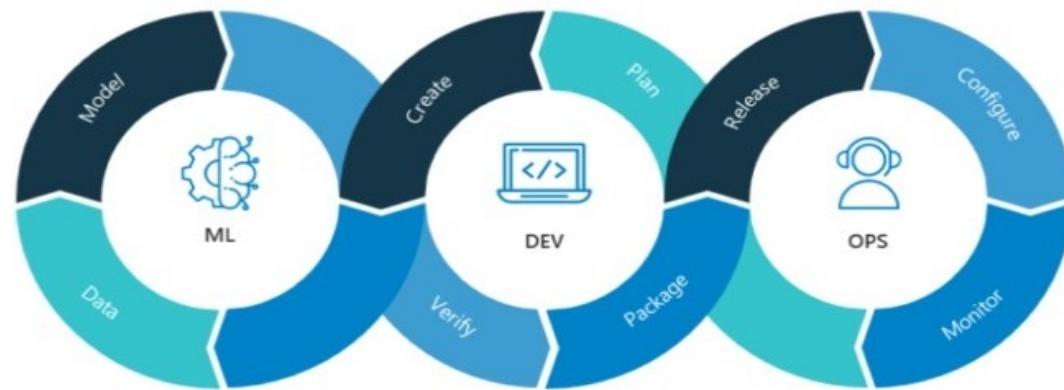
References





# What is MLOps ?

- Machine Learning Operations (MLOps) includes a series of steps to deploy an ML model into a production environment.
- It combines aspects of Machine Learning and DevOps
- Several steps must be completed before an ML model is ready for production.
- These steps ensure that your model can handle many users and work accurately.





# Why do we need MLOps ?

- Creating an ML model that can predict based on the data you provide is simple. However, making a ML model that is reliable, fast, accurate, and can be used by many users is challenging.





# Importance of MLOps

- ML models depend on a large amount of data, which is hard for one person to manage.
- It is challenging to keep track of the parameters we adjust in ML models. Small changes can significantly impact the results.
- We need to monitor the features the model uses, as feature engineering is a separate task that greatly contributes to the model's accuracy.





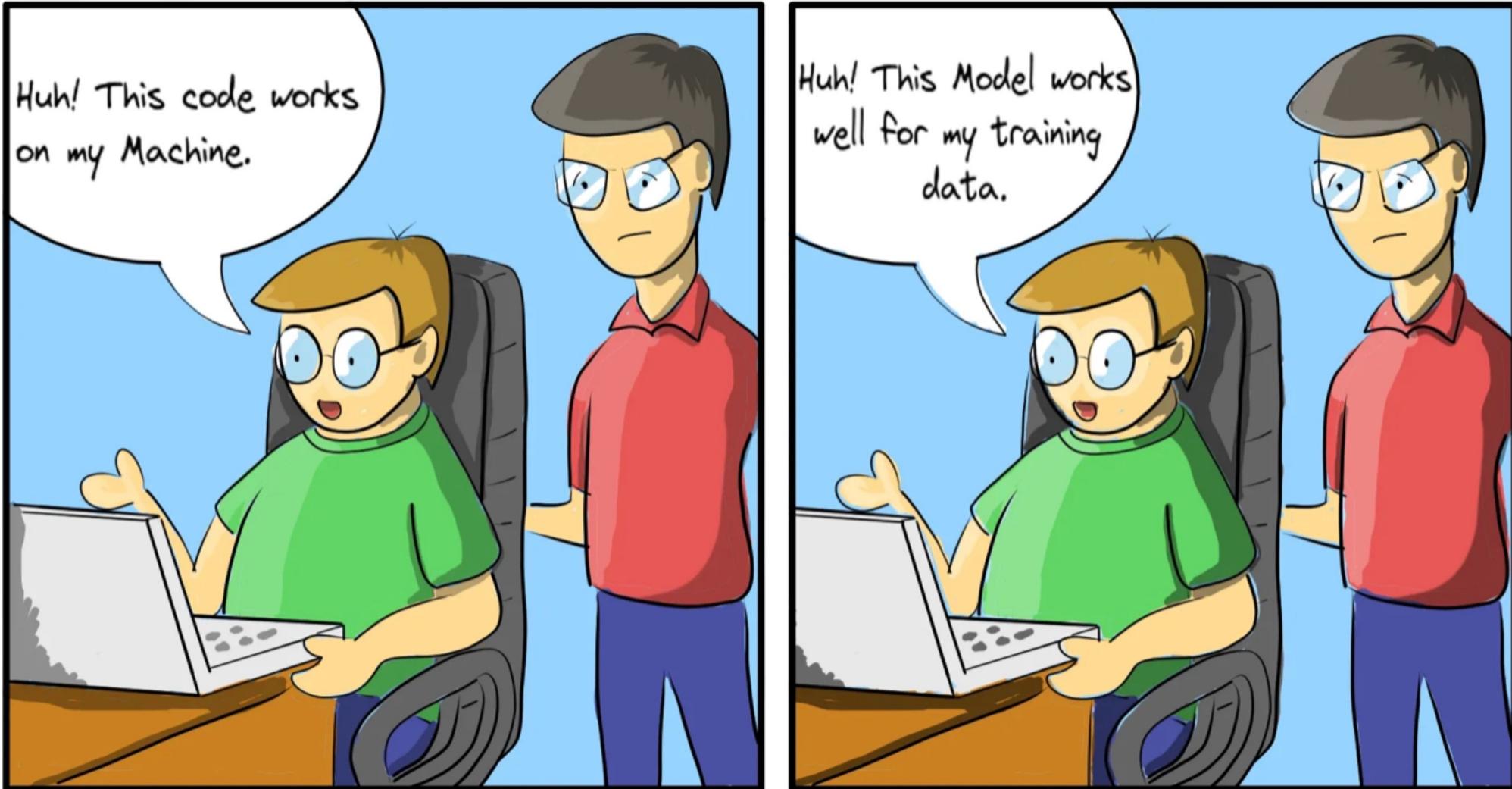
# Importance of MLOps

- Monitoring an ML model is different from monitoring a deployed software application or web app.
- Debugging an ML model is a highly complex skill.
- Models depend on real-world data for predictions, and as this data changes, the model must adapt. We need to monitor new data changes and ensure the model learns from them.





# Importance of MLOps



[Source](#)

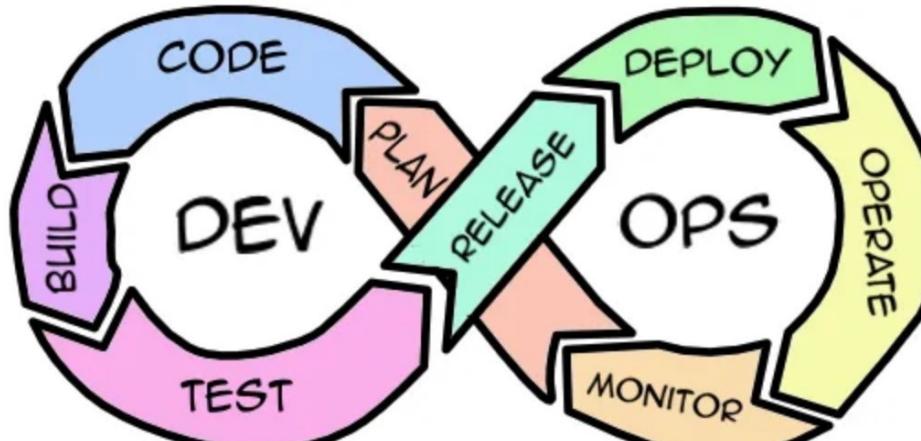


# MLOps Lifecycle



# DevOps life cycle

- You may be familiar with traditional DevOps, which involves building and deploying software applications. You might be curious about how MLOps differs from this process.



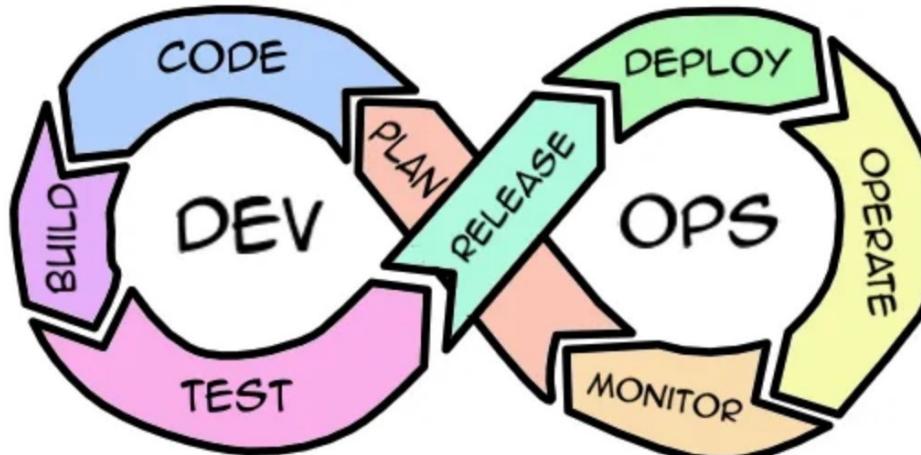
DevOps Cycle (Image by [Author](#))





# DevOps life cycle

- DevOps stages focus on developing a software application. You start by planning the application's features, writing the code, building it, testing it, creating a release plan, and deploying it. You then monitor the infrastructure where the app is deployed. This cycle repeats until the application is fully developed.



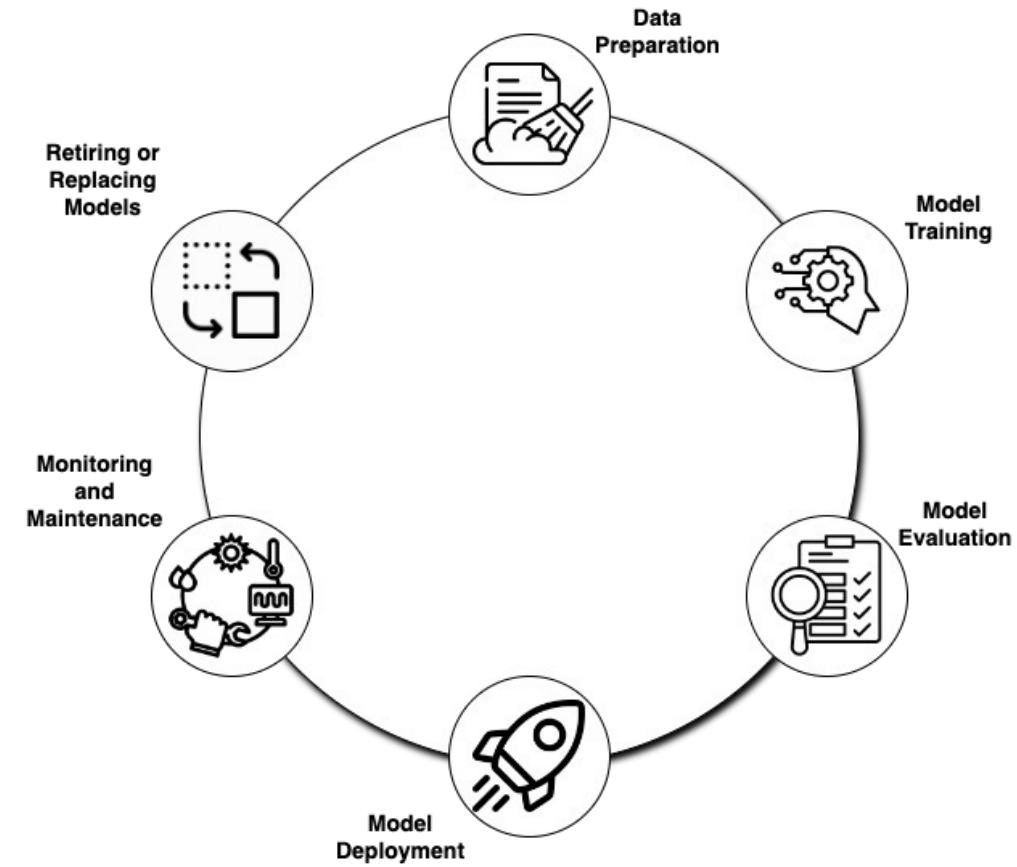
DevOps Cycle (Image by [Author](#))





# MLOps Lifecycle (Data Preparation)

**Data Preparation** is the process of gathering, cleaning, and transforming raw data into a structured format suitable for machine learning.

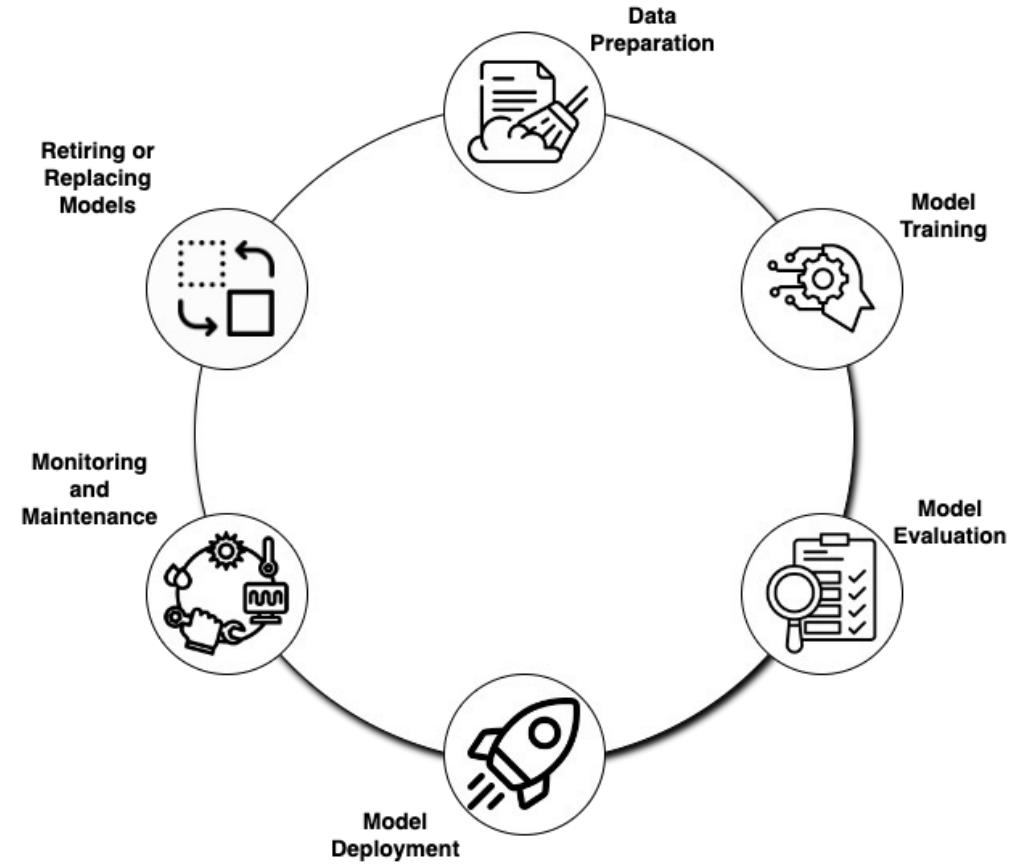




# MLOps Lifecycle (Data Preparation)

## Steps:

- **Data Collection:** Gather data from various sources like databases, APIs, sensors, etc.
- **Data Cleaning:** Remove or correct any errors, duplicates, or inconsistencies in the data.

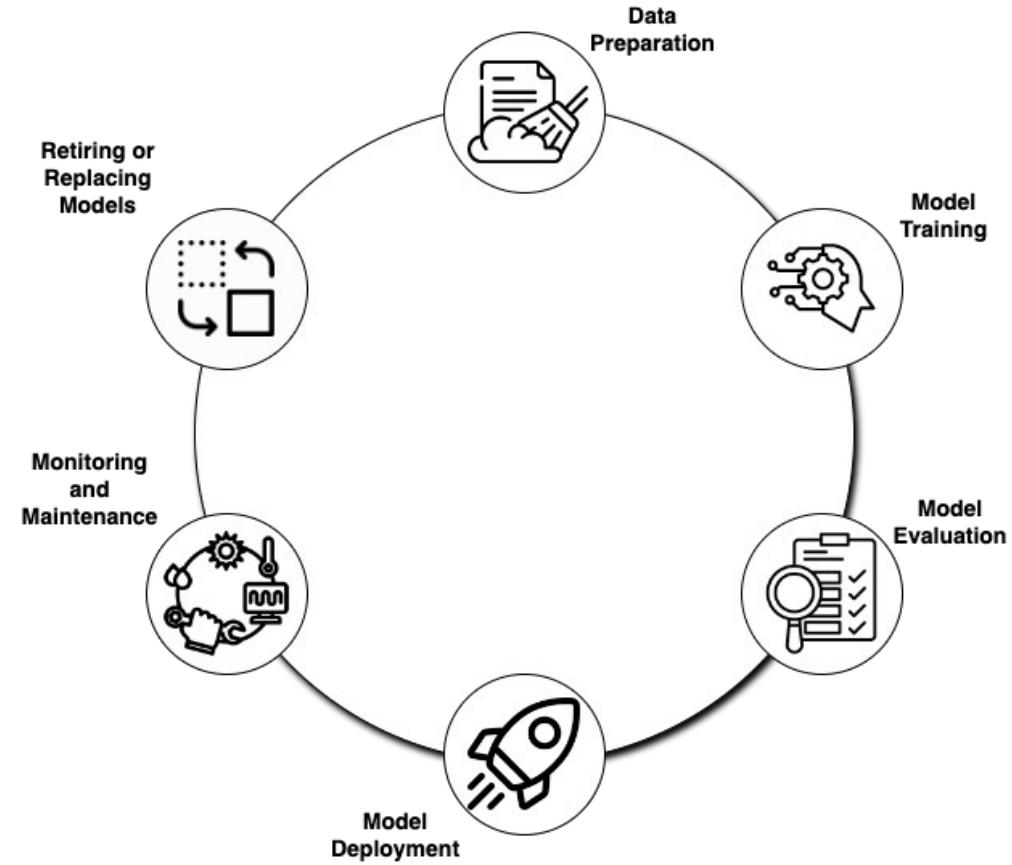




# MLOps Lifecycle (Data Preparation)

## Steps:

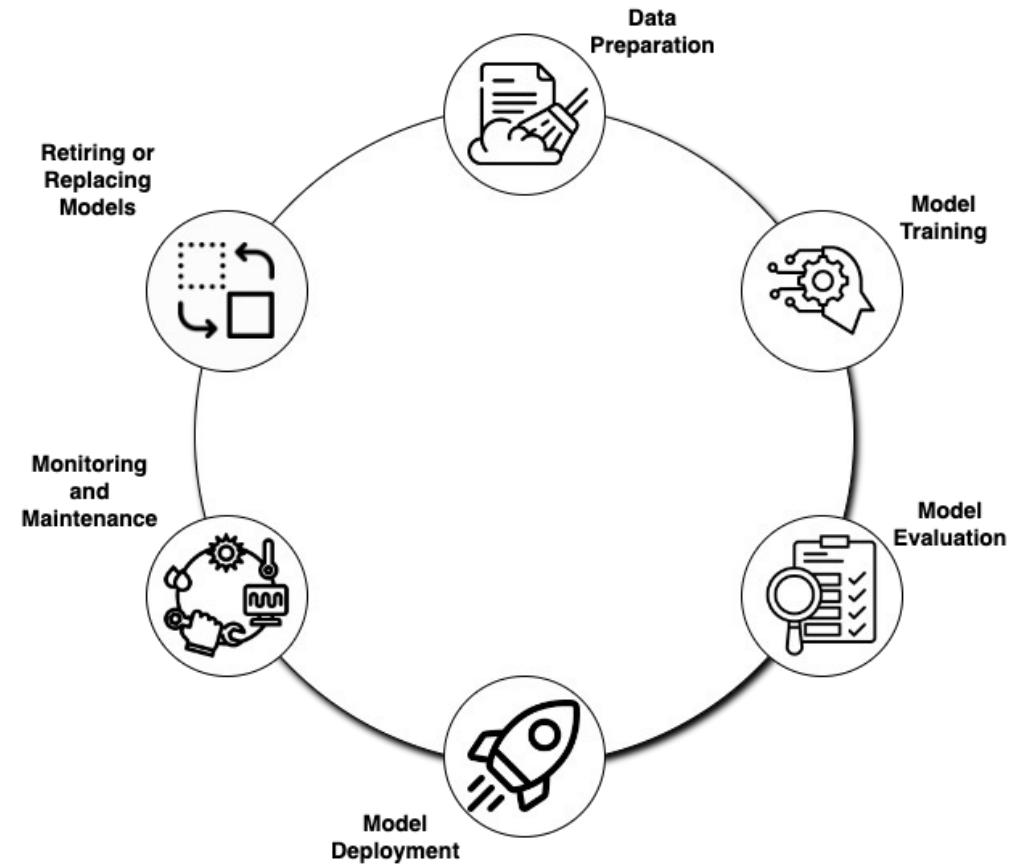
- **Data Transformation:** Convert data into the required format, such as normalizing, encoding categorical variables, and scaling.
- **Data Splitting:** Divide the data into training, validation, and test sets.





# MLOps Lifecycle (Model Training)

**Model Training** is the phase where data scientists and machine learning engineers design, build, and optimize machine learning models.

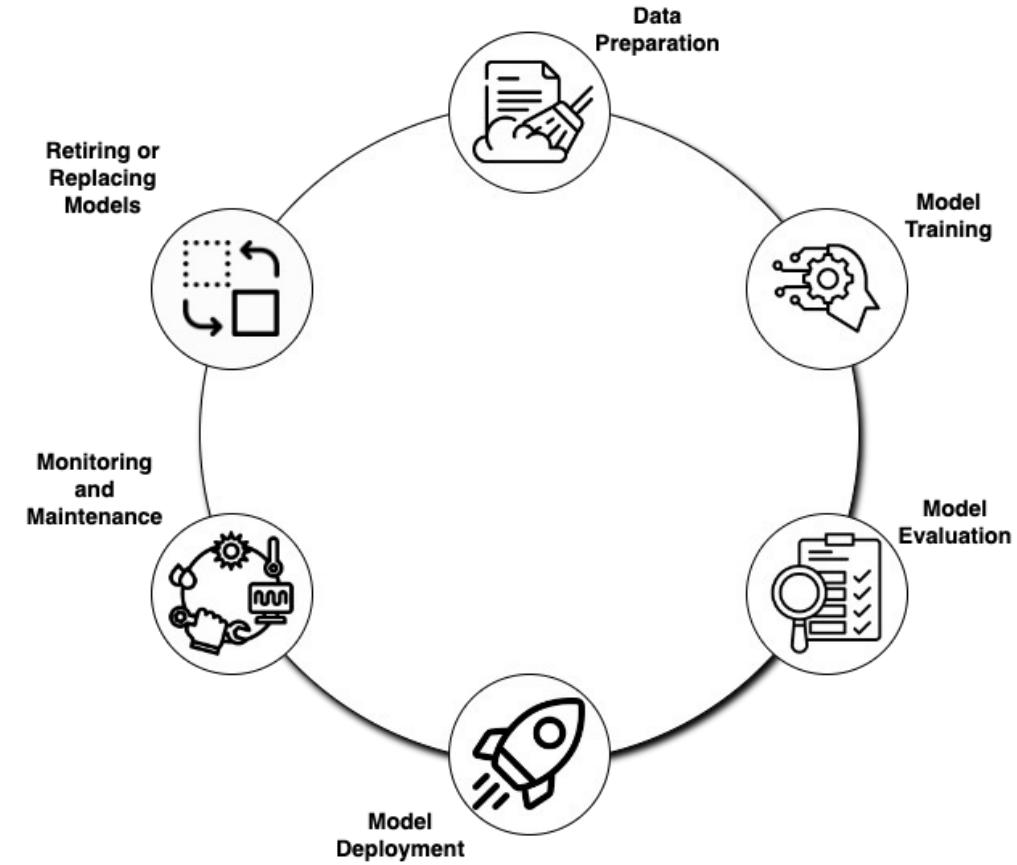




# MLOps Lifecycle (Model Training)

## Steps:

- **Algorithm Selection:** Choose the appropriate machine learning algorithm for the task.
- **Feature Engineering:** Create new features or select important features from the dataset to improve model performance.

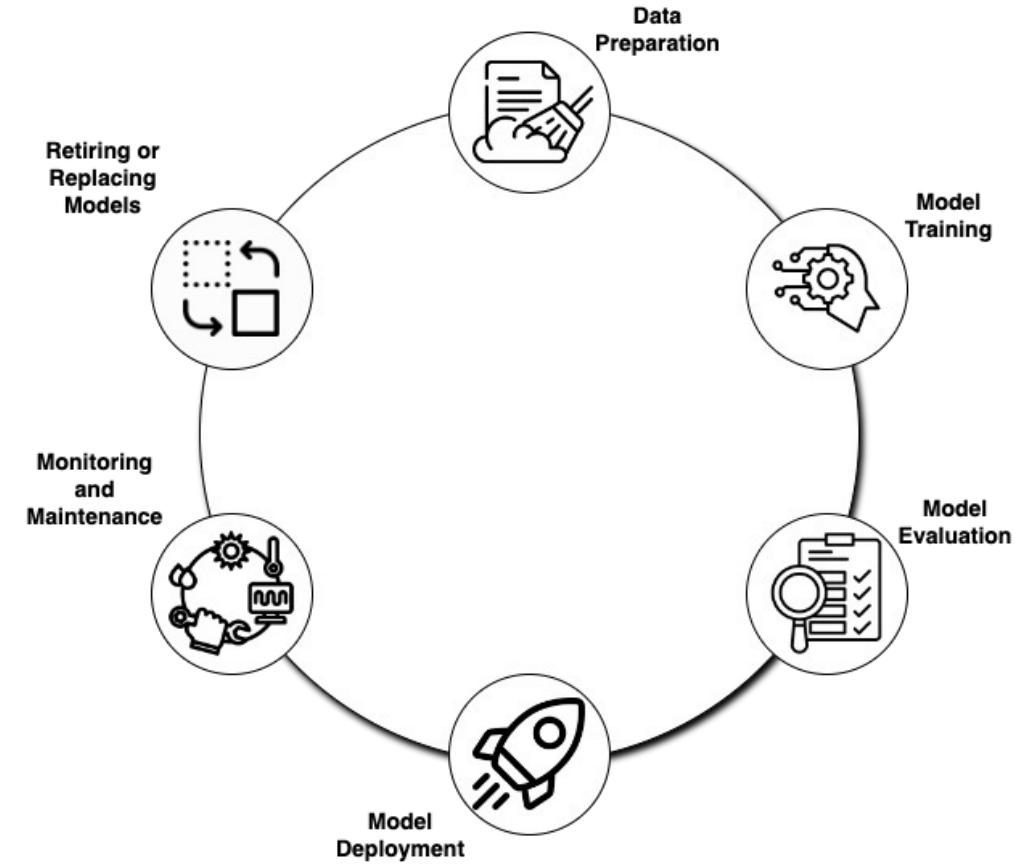




# MLOps Lifecycle (Model Training)

## Steps:

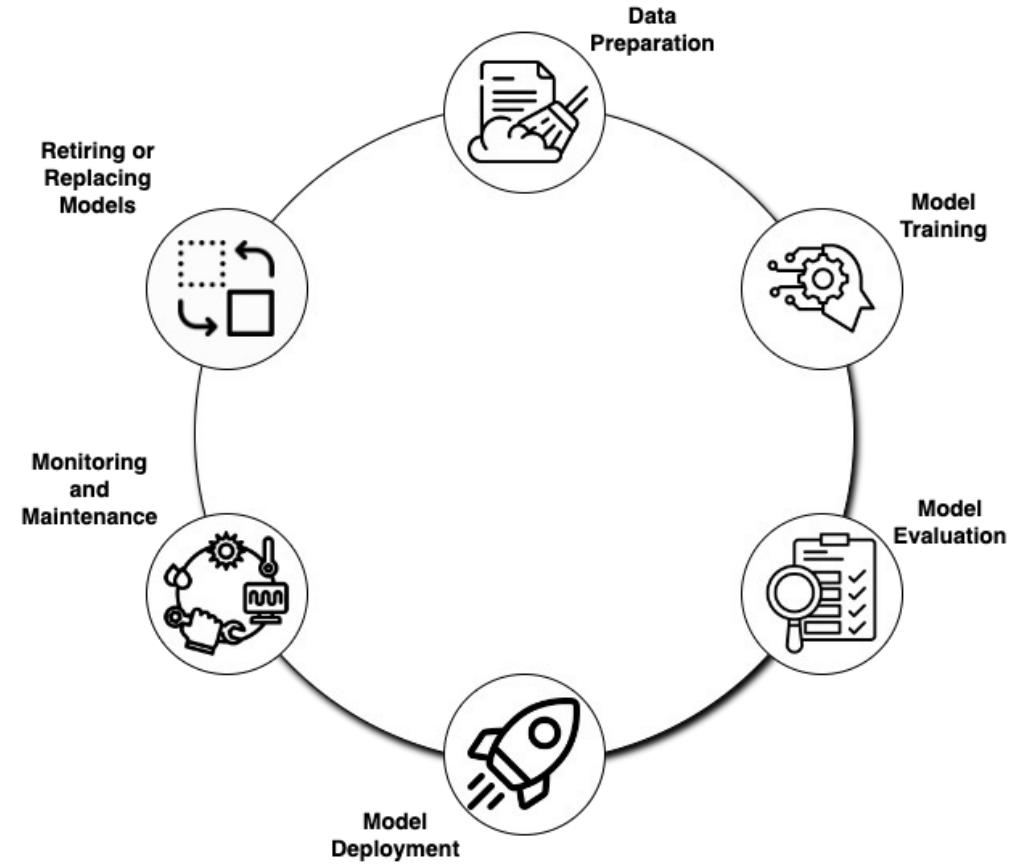
- **Model Building:** Develop the first version of the model using selected algorithm and features.
- **Hyperparameter Tuning:** Adjust the hyperparameters to improve model performance.





# MLOps Lifecycle (Model Evaluation)

**Model Evaluation** step involves evaluation the model's performance using various metrics to ensure it meets the required standards before deployment.



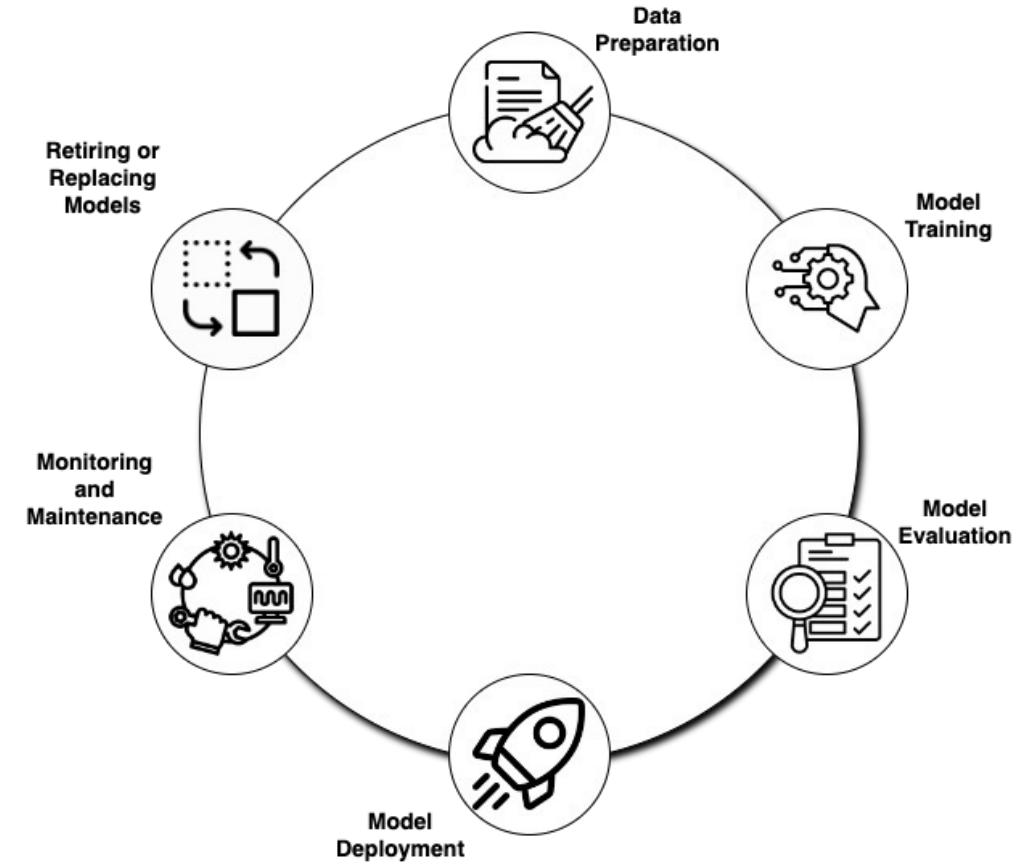


# MLOps Lifecycle (Model Evaluation)

## Steps:

**Validation Testing:** Evaluate the model using the validation dataset to fine-tune it.

**Performance Metrics:** Calculate metrics such as accuracy, precision, recall, and F1-score.



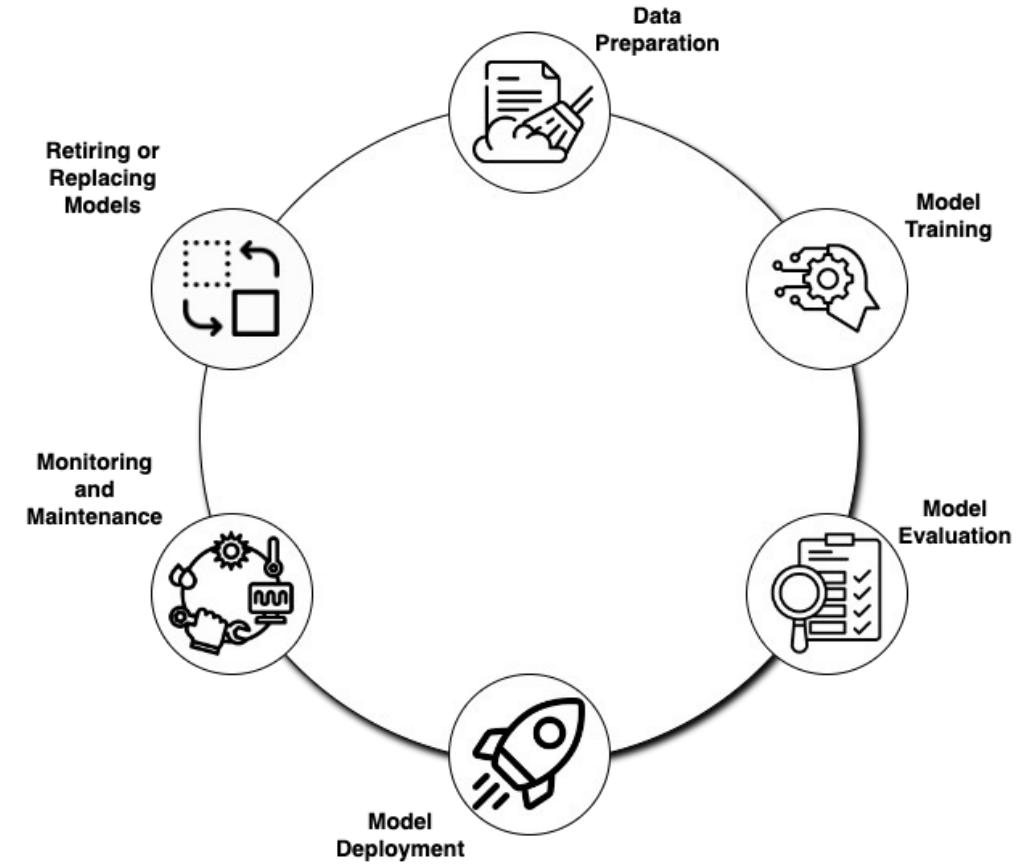


# MLOps Lifecycle (Model Evaluation)

## Steps:

**Cross-Validation:** Perform cross-validation to ensure the model's robustness.

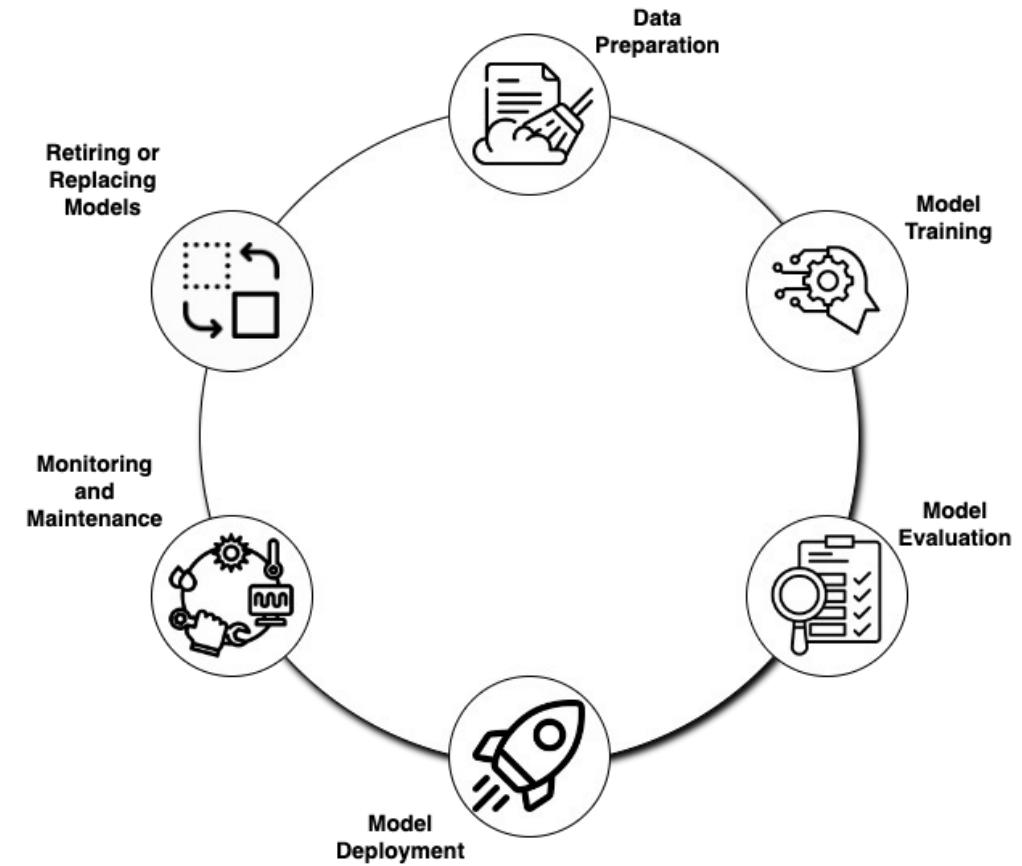
**Bias and Fairness Testing:** Check the model for biases and ensure fairness.





# MLOps Lifecycle (Model Deployment)

**Model Deployment** is deploying the validated model into a production environment where it can start making predictions on live data for example in Amazon's AWS, Google Cloud or even a private cloud.

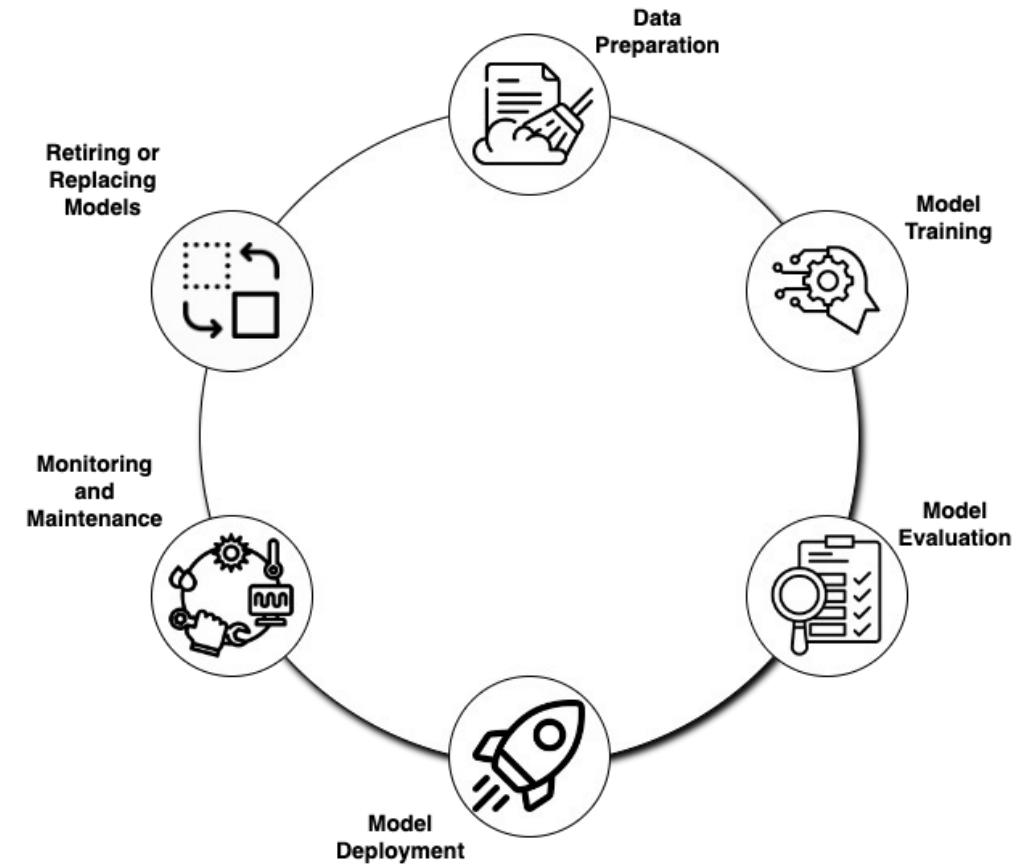




# MLOps Lifecycle (Model Deployment)

## Steps:

- **Infrastructure Setup:** Prepare the production environment and necessary infrastructure.
- **Model Integration:** Integrate the model with the existing systems and applications.

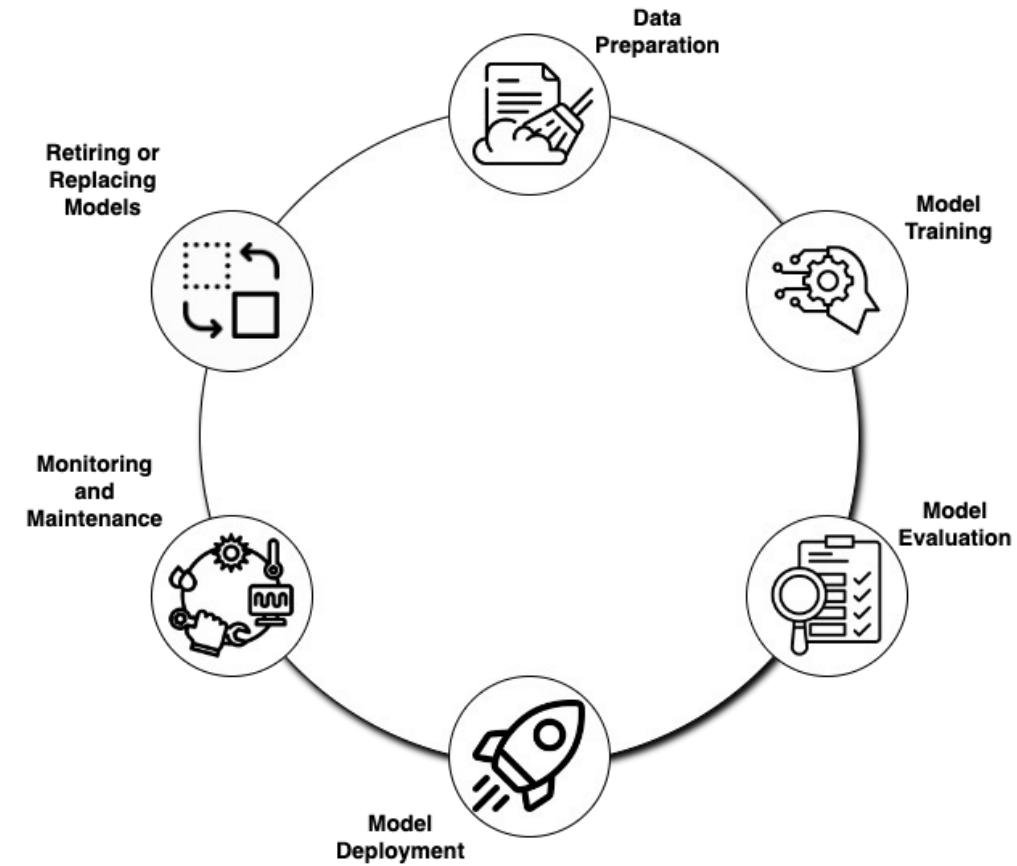




# MLOps Lifecycle (Model Deployment)

## Steps:

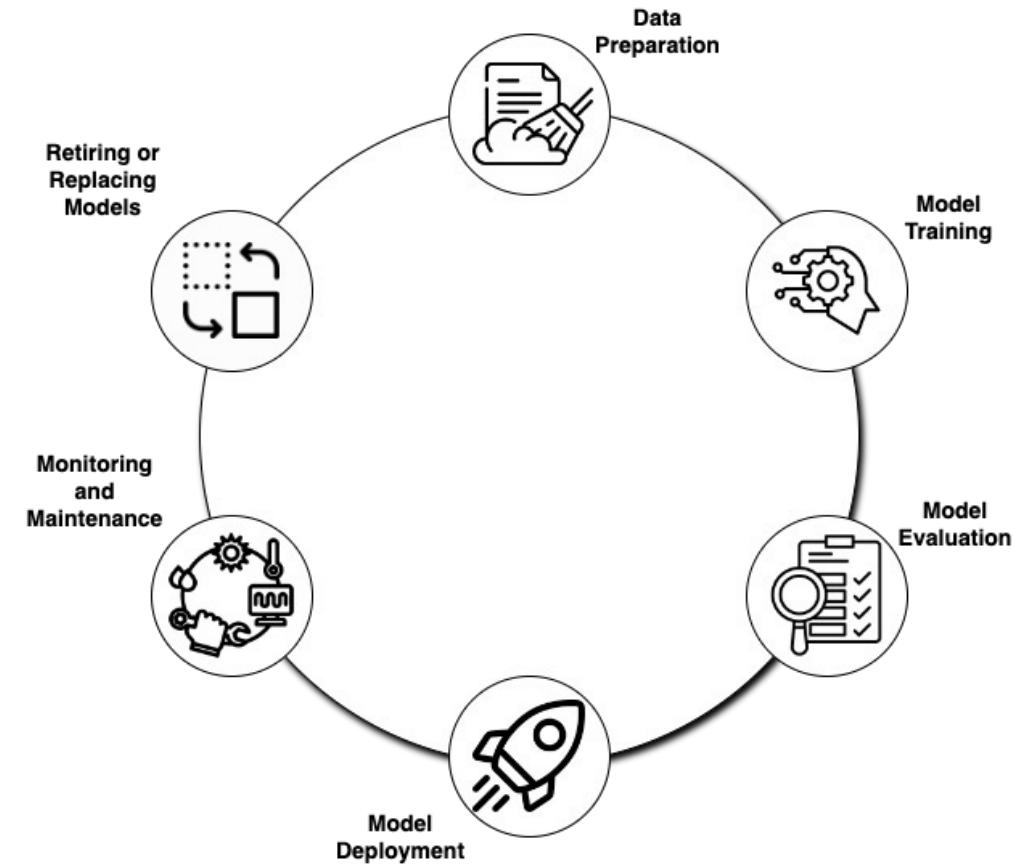
- **Deployment Testing:** Conduct tests to ensure the model works correctly in the production environment.
- **Release:** Roll out the model for production use.





# MLOps Lifecycle (Monitoring and Maintenance)

**Monitoring and Maintenance** is continuous monitoring the model's performance and maintaining it to ensure it continues to perform well over time.

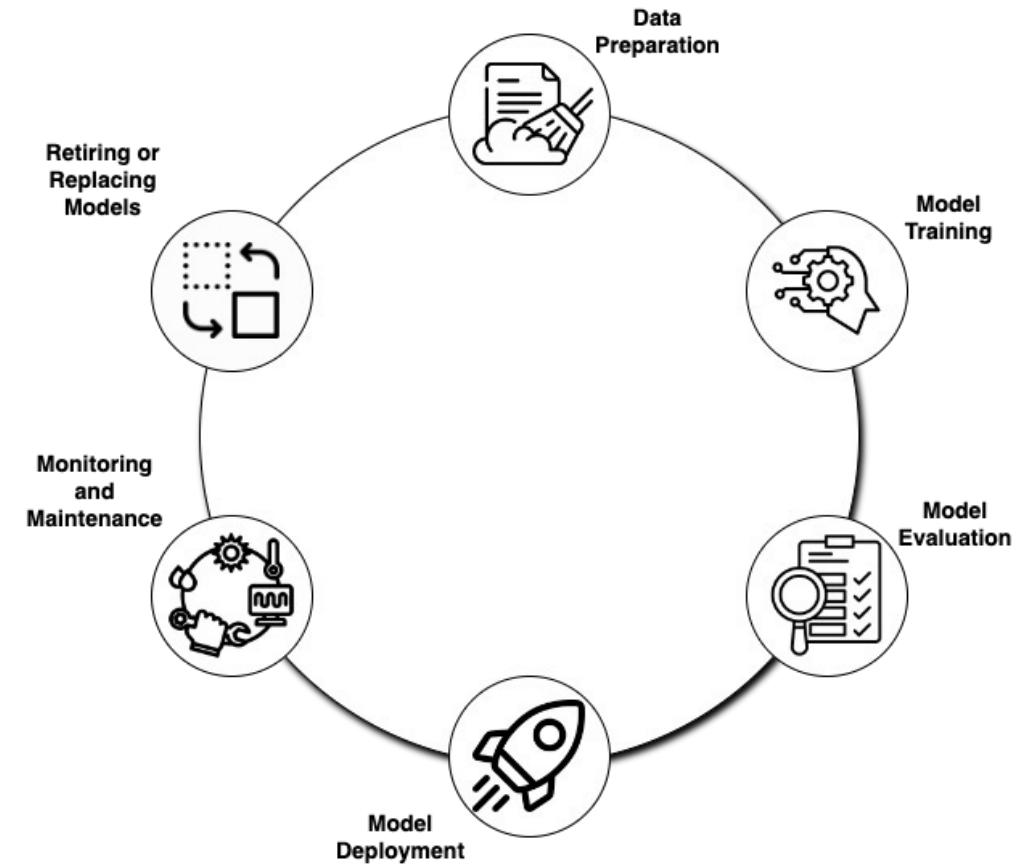




# MLOps Lifecycle (Monitoring and Maintenance)

## Steps:

- **Performance Monitoring:** Track the model's performance on live data.
- **Error Analysis:** Identify and analyze errors to understand model weaknesses.

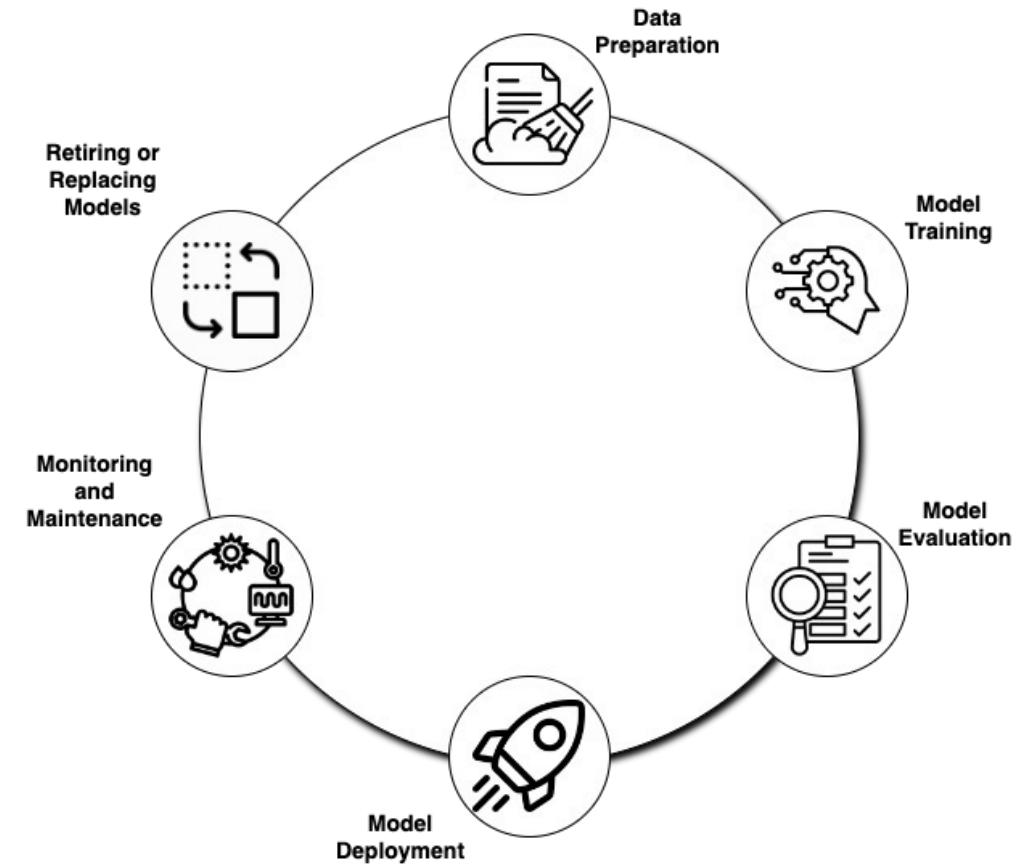




# MLOps Lifecycle (Monitoring and Maintenance)

## Steps:

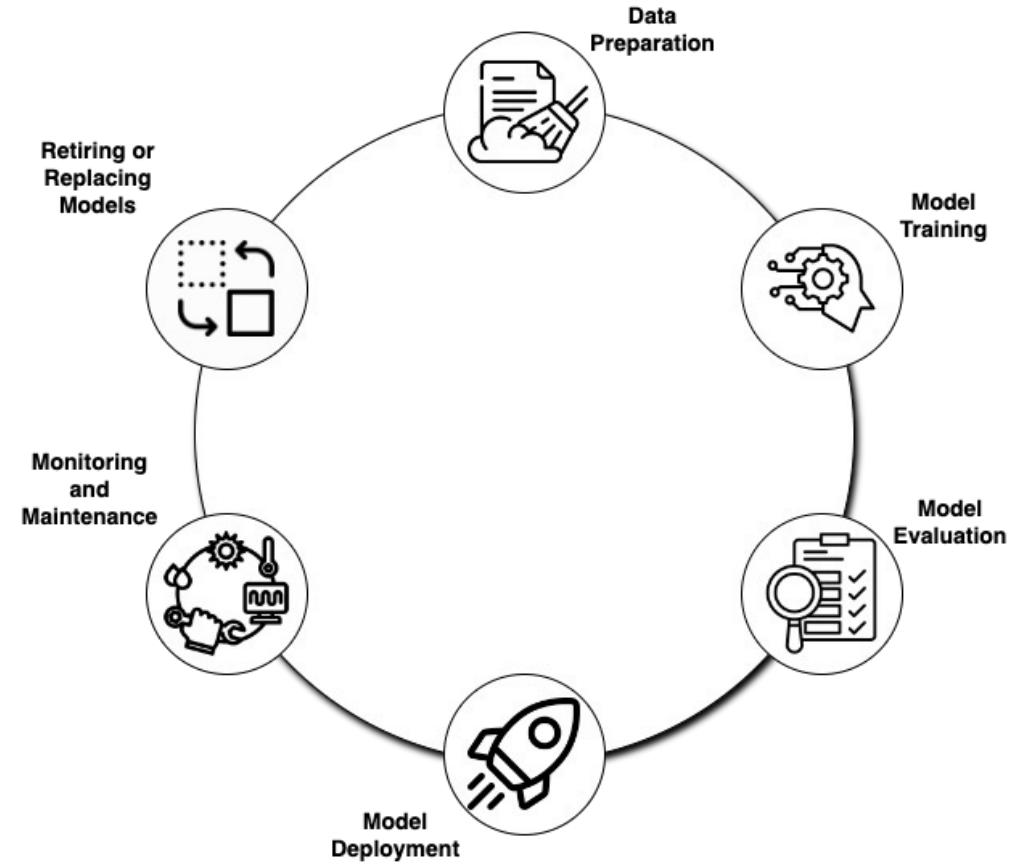
- **Data Drift Detection:** Detect shifts in data distribution that could affect model performance.
- **Model Retraining:** Periodically retrain the model with new data to maintain accuracy.





# MLOps Lifecycle (Retiring or Replacing Models)

**Retiring and Replacing** is when a model no longer meets performance standards or becomes obsolete, it must be retired and replaced with a new or updated model.

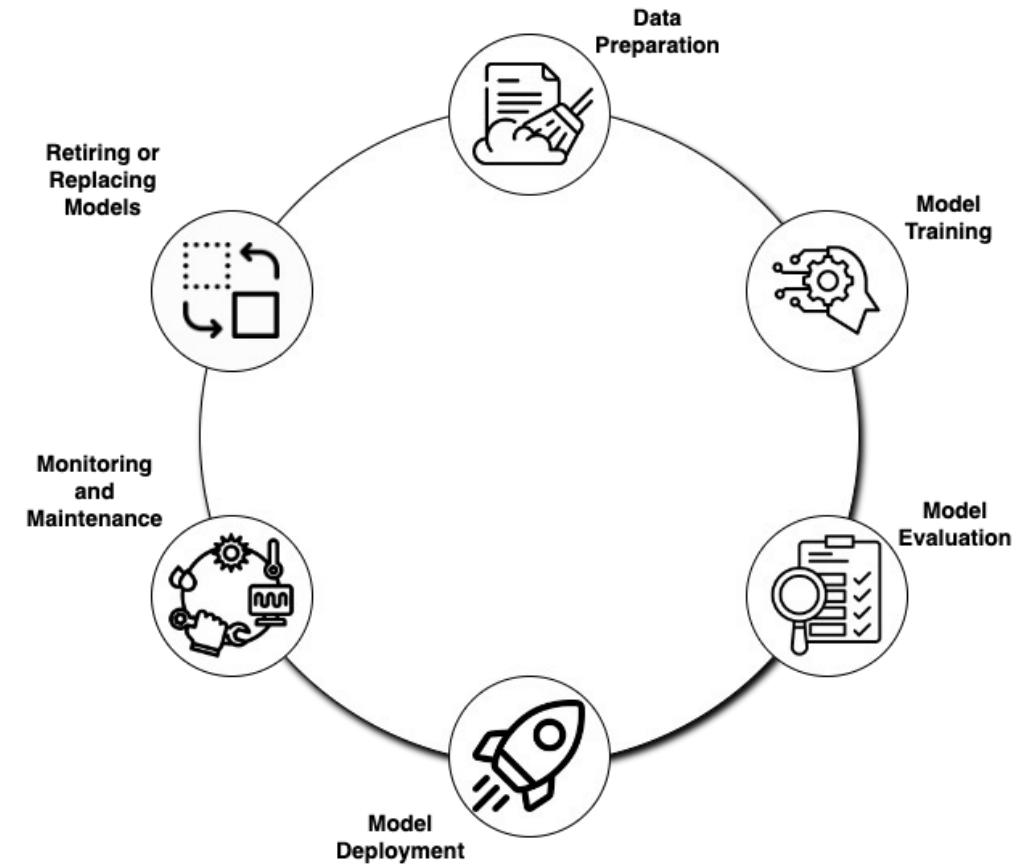




# MLOps Lifecycle (Retiring or Replacing Models)

## Steps:

- **Performance Review:** Regularly review the model's performance and relevance.
- **Replacement Strategy:** Plan for replacing the outdated model with a new one.

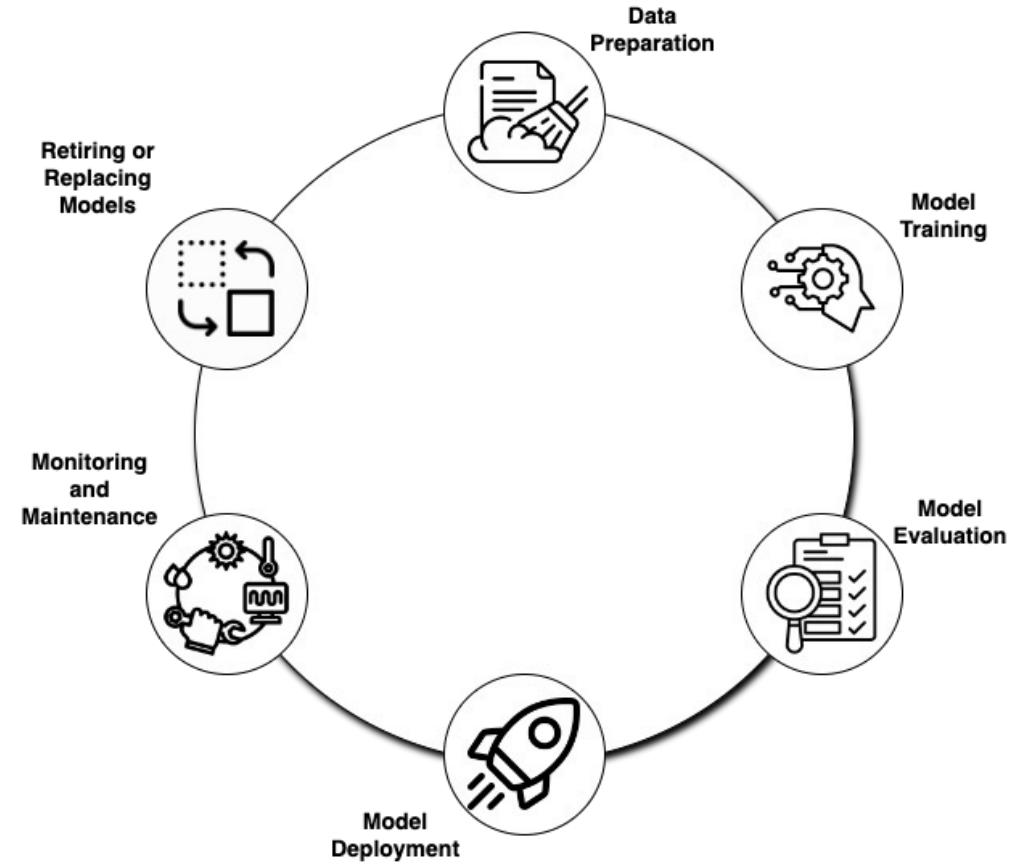




# MLOps Lifecycle (Retiring or Replacing Models)

## Steps:

- **Model Decommissioning:** Safely retire the old model, ensuring a smooth transition.
- **New Model Deployment:** Deploy the updated model following the deployment steps.

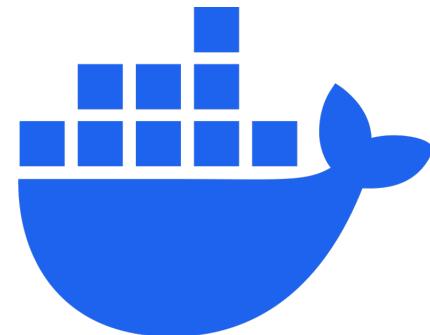
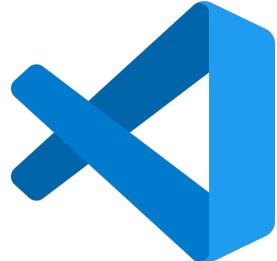


# Setting up local environment



# Overview of Local Development Environment

A local development environment is a setup on your local device that allows you to develop, test, and debug machine learning models before deploying them to production.

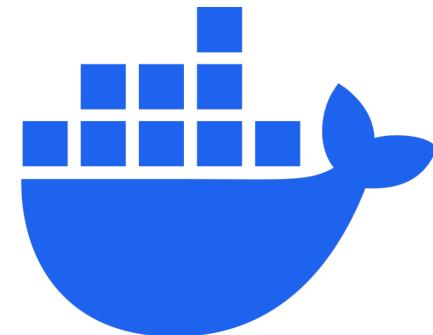
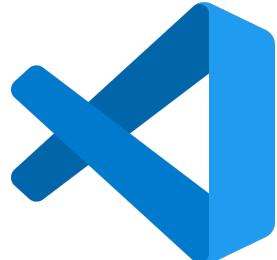




# Overview of Local Development Environment

Programming Languages: Python is the most common choice for ML development.

- **IDE/Text Editor:** Popular options include VS Code, PyCharm, Jupyter Notebook.
- **Package Managers:** pip, conda for managing Python packages.

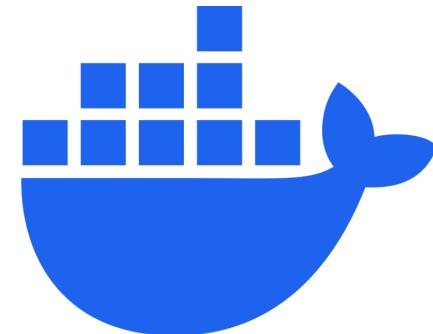
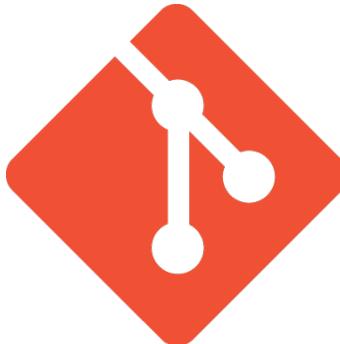
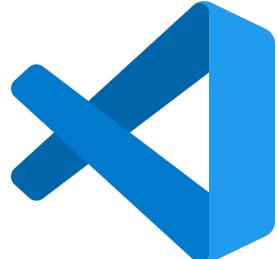




# Overview of Local Development Environment

Programming Languages: Python is the most common choice for ML development.

- **Version Control:** Git for tracking changes and collaboration.
- **Containerization:** Docker for creating consistent environments.





# Installing Essential Tools (Python Installation)

If you haven't ☺

To install the latest version of Python, visit the [official website](#).

The screenshot shows the Python official website's download page. A red arrow points to the 'Download Python 3.12.3' button. The page features a banner with two boxes labeled 'macOS' and 'Windows' with parachutes. Below the banner, there's a section for 'Active Python Releases' with a table of versions.

Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569





# Installing Essential Tools (Package Managers)

pip comes with Python by default, so you don't need to download it.

The screenshot shows the Python.org website. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is the Python logo and a search bar. A red arrow points to a yellow button labeled "Download Python 3.12.3". To the right of the button is a graphic of two parachutes descending from clouds. Below the button, text says "Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)". There's also a link for "Want to help test development versions of Python 3.13? [Prereleases](#), [Docker images](#)". At the bottom, there's a section titled "Active Python Releases" with a table of release information.

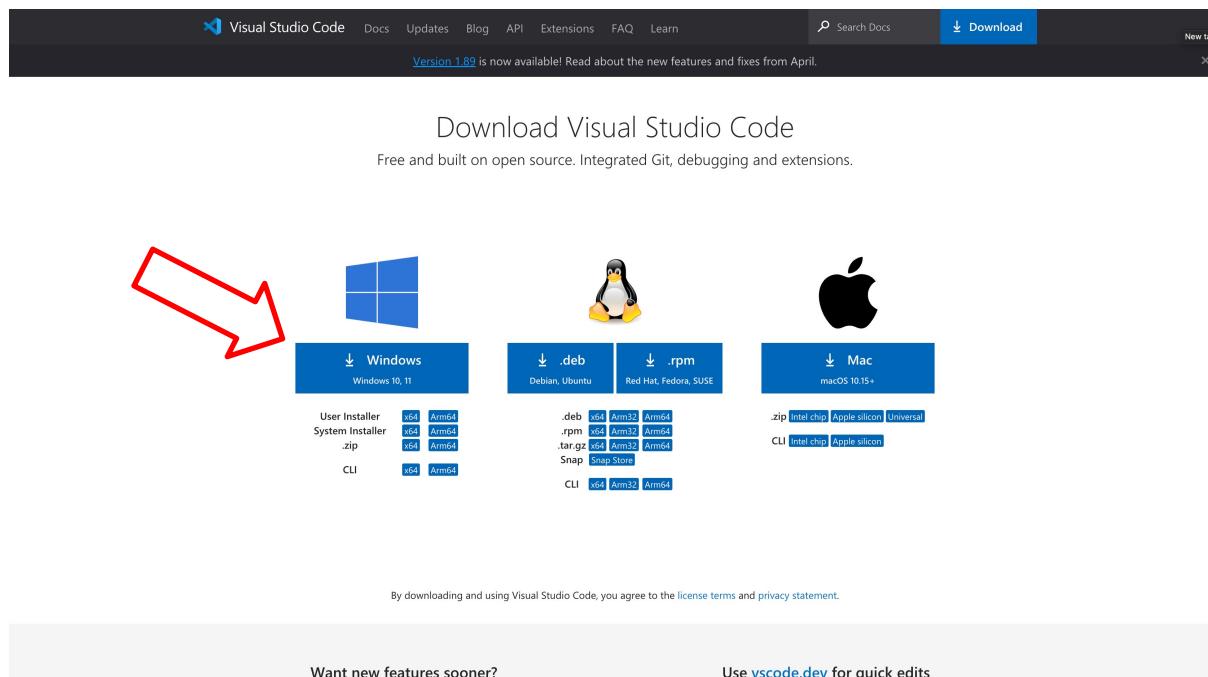
Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569





# Installing Essential Tools (VS Code)

To install VS Code, you will need to visit the [official website](#) and install the compatible version with your operating system.



The screenshot shows the official Visual Studio Code download page. At the top, there's a navigation bar with links for Docs, Updates, Blog, API, Extensions, FAQ, Learn, a search bar, a 'Download' button, and a 'New tab' link. A message at the top indicates 'Version 1.89 is now available! Read about the new features and fixes from April.' Below the navigation, the heading 'Download Visual Studio Code' is displayed, followed by the subtext 'Free and built on open source. Integrated Git, debugging and extensions.' To the left of the download links is a large red arrow pointing towards the Windows download section. The page features icons for Windows (Windows logo), Linux (Tux the Penguin), and macOS (Apple logo). Below each icon are download links for different operating systems and architectures. For Windows, there are links for User Installer (.zip) and System Installer (.msi), both in x64 and Arm64 formats. For Linux, there are links for .deb (Debian, Ubuntu) and .rpm (Red Hat, Fedora, SUSE), both in x64 and Arm64 formats. For macOS, there are links for .zip (Intel chip, Apple silicon, Universal) and CLI (Intel chip, Apple silicon), both in x64 and Arm64 formats. At the bottom of the page, a note states 'By downloading and using Visual Studio Code, you agree to the [license terms](#) and [privacy statement](#)'. A footer bar at the very bottom includes links for 'Want new features sooner?' and 'Use [vscode.dev](#) for quick edits'.





# Setting Up Version Control (git - win)

To install git, you will need to visit the [official website](#) and install the compatible version with your operating system. For windows visit [here](#).

The screenshot shows the official Git website ([git-scm.com/](https://git-scm.com/)). The main navigation bar includes links for 'About', 'Documentation' (with 'Book' highlighted), 'Downloads', and 'Community'. A search bar at the top right says 'Search entire site...'. The main content area is titled '1.5 Getting Started - Installing Git'. It contains several sections: 'Installing Git' (with a note about Git version 2), 'Installing on Linux' (with examples for DNF and APT), 'Installing on macOS' (with a note about Xcode Command Line Tools), and 'Installing on Windows' (with a note about the Windows Subsystem for Linux). The footer lists various language translations available for the documentation.

git --everything-is-local

About Documentation Chapters 2nd Edition

Search entire site...

1.5 Getting Started - Installing Git

Installing Git

Before you start using Git, you have to make it available on your computer. Even if it's already installed, it's probably a good idea to update to the latest version. You can either install it as a package or via another installer, or download the source code and compile it yourself.

Note This book was written using Git version 2. Since Git is quite excellent at preserving backwards compatibility, any recent version should work just fine. Though most of the commands we use should work even in ancient versions of Git, some of them might not or might act slightly differently.

Installing on Linux

If you want to install the basic Git tools on Linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you're on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

```
$ sudo dnf install git-all
```

If you're on a Debian-based distribution, such as Ubuntu, try apt:

```
$ sudo apt install git-all
```

For more options, there are instructions for installing on several different Unix distributions on the Git website, at <https://git-scm.com/download/linux>.

Installing on macOS

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run git from the Terminal the very first time.

```
$ git --version
```

If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is available at <https://git-scm.com/mac>.

This book is available in English. Full translation available in azərbaycan dilin, български език, Deutsch, Español, Français, Ελληνικά, 日本語, 한국어, Nederlands, Русский, Slovenščina, Tagalog, Українська 简体中文, Partial translations available in Čeština, Македонски, Polski, Српски.





# Setting Up Version Control (git - macOS)

To install git, you will need to visit the [official website](#) and install the compatible version with your operating system. For macOS use homebrew to install it via the command `brew install git`.

The screenshot shows the official Git website ([git-scm.com](#)) with the page title "1.5 Getting Started - Installing Git". The page content includes instructions for installing Git on Linux, Mac OS X, and Windows. It features code snippets for terminal commands like `$ sudo apt-get install git` and `$ sudo dnf install git`. A note at the bottom states: "If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is available at [https://git-scm.com/download/mac](#)". The left sidebar contains links for "About", "Documentation" (with "Book" highlighted), "Downloads", and "Community". The footer lists various language translations available for the documentation.

git --everything-is-local

About

Documentation

Reference

Book

Videos

External Links

Downloads

Community

This book is available in English. Full translation available in azərbaycan dilin, български език, Deutsch, Español, Français, Ελληνικά, 日本語, 한국어, Nederlands, Русский, Slovenščina, Tagalog, Українська, 简体中文, Partial translations available in Čeština, Македонски, Polski, Српски.

Chapters ▾ 2nd Edition

## 1.5 Getting Started - Installing Git

### Installing Git

Before you start using Git, you have to make it available on your computer. Even if it's already installed, it's probably a good idea to update to the latest version. You can either install it as a package or via another installer, or download the source code and compile it yourself.

**Note**

This book was written using Git version 2. Since Git is quite excellent at preserving backwards compatibility, any recent version should work just fine. Though most of the commands we use should work even in ancient versions of Git, some of them might not or might act slightly differently.

### Installing on Linux

If you want to install the basic Git tools on Linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you're on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use `dnf`:

```
$ sudo dnf install git-all
```

If you're on a Debian-based distribution, such as Ubuntu, try `apt`:

```
$ sudo apt install git-all
```

For more options, there are instructions for installing on several different Unix distributions on the Git website, at [https://git-scm.com/download/linux](#).

### Installing on macOS

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run `git` from the Terminal the very first time.

```
$ git --version
```

If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is available at [https://git-scm.com/download/mac](#).





# Setting Up Version Control (git - Linux)

To install git, you will need to visit the [official website](#) and install the compatible version with your operating system. For Linux install it via the command `apt-get install git`.

The screenshot shows the official Git documentation website at [git-scm.com/](https://git-scm.com/). The page title is "1.5 Getting Started - Installing Git". The main content discusses installing Git on various Linux distributions, providing terminal commands for DNF and APT. It also includes notes about Git's compatibility and a note about the version used for the documentation.

**1.5 Getting Started - Installing Git**

**Installing Git**

Before you start using Git, you have to make it available on your computer. Even if it's already installed, it's probably a good idea to update to the latest version. You can either install it as a package or via another installer, or download the source code and compile it yourself.

**Note**

This book was written using Git version 2. Since Git is quite excellent at preserving backwards compatibility, any recent version should work just fine. Though most of the commands we use should work even in ancient versions of Git, some of them might not or might act slightly differently.

**Installing on Linux**

If you want to install the basic Git tools on Linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you're on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

```
$ sudo dnf install git-all
```

If you're on a Debian-based distribution, such as Ubuntu, try apt:

```
$ sudo apt install git-all
```

For more options, there are instructions for installing on several different Unix distributions on the Git website, at <https://git-scm.com/download/linux>.

**Installing on macOS**

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run `git` from the Terminal the very first time.

```
$ git --version
```

If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is available at <https://git-scm.com/mac>.





# Containerization with Docker

To install Docker, visit the [official Docker website](#) and follow installation instructions.

The screenshot shows the Docker Desktop landing page. At the top, there's a dark header with the Docker logo, navigation links for Products, Developers, Pricing, Support, Blog, Company, and user options for Sign In and Get started. Below the header, the text "Docker Desktop" is displayed, followed by the bold statement "The #1 containerization software for developers and teams". A red arrow points from the text above to the "Download for Mac - Apple Chip" button. The download section also includes "Create an account" and links for "Download for Mac - Intel Chip", "Download for Windows", and "Download for Linux". A note at the bottom states that commercial use requires a paid subscription for companies with more than 250 employees or \$10 million in annual revenue.

Docker Desktop

The #1 containerization software for developers and teams

Your command center for innovative container development

Download for Mac - Apple Chip

Create an account

Download for Mac - Intel Chip

Download for Windows

Download for Linux

Commercial use of Docker Desktop at a company of more than **250 employees** OR more than **\$10 million** in annual revenue requires a paid subscription (Pro, Team, or Business).

[https://desktop.docker.com/mac/main/arm64/Docker.dmg?utm\\_source=docker&utm\\_medium=webreferral&utm\\_campaign=dd-smartbutton&utm\\_location=module](https://desktop.docker.com/mac/main/arm64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=dd-smartbutton&utm_location=module)

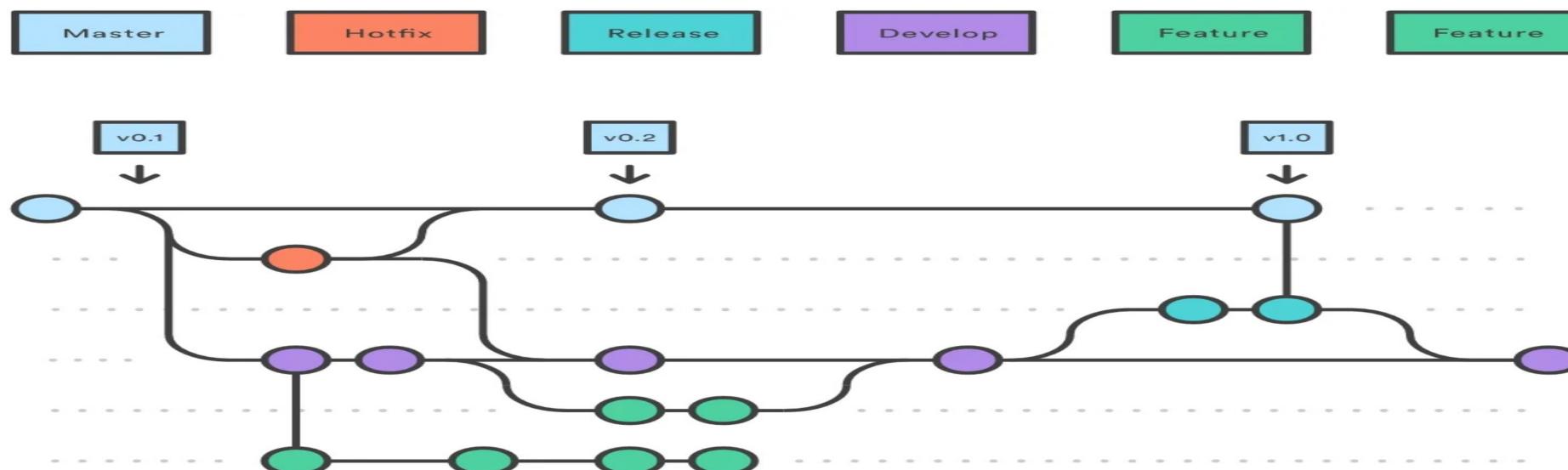


# Version Control



# What is Version Control ?

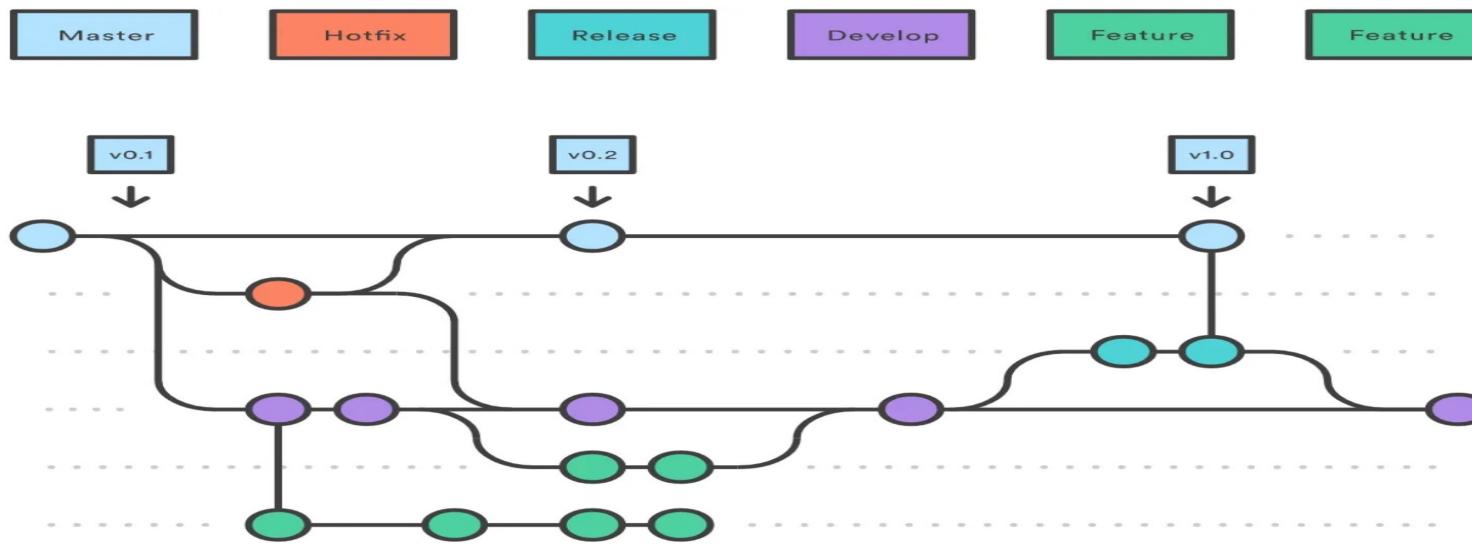
- **Version control** is the process of managing changes to a set of files over time. It enables developers to track changes, collaborate with others, and revert to previous versions if needed. Version control systems (VCS) like Git and SVN are commonly used in software development to manage code changes. In ML projects, version control is also used to track changes to datasets, preprocessing scripts, training scripts, and model files.





# Why is Version Control important ?

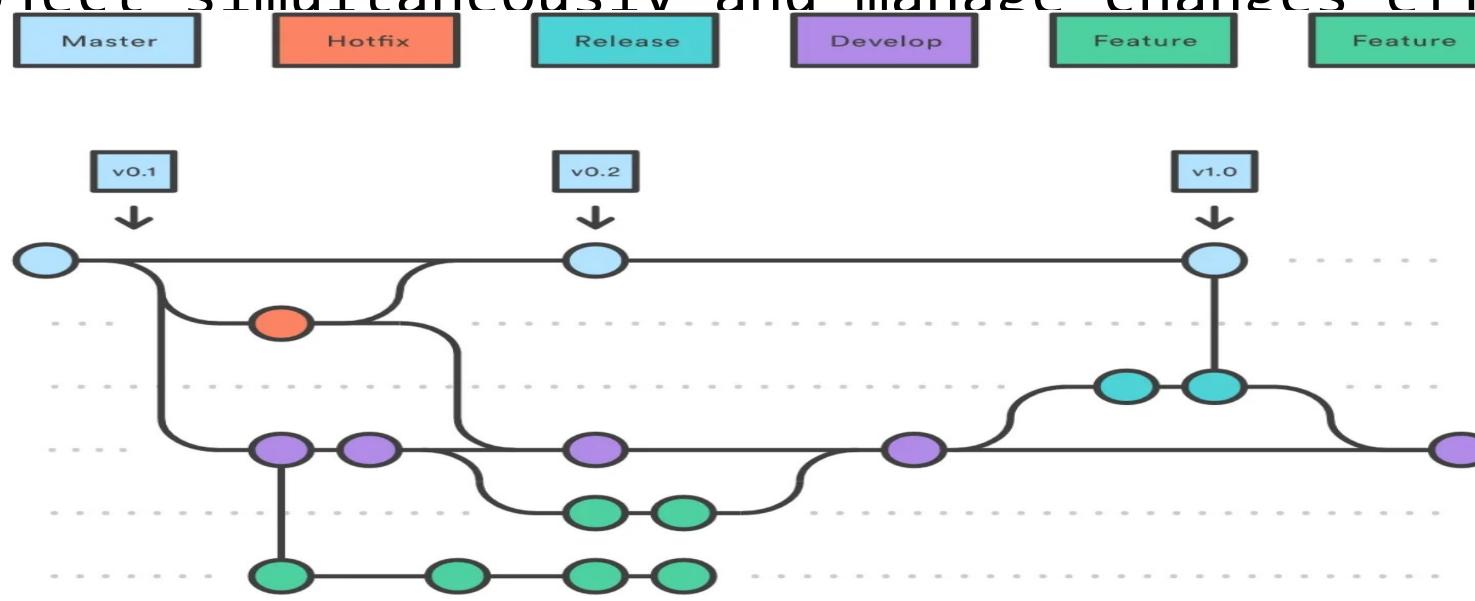
- **Version control** is especially crucial in ML projects because of the large volumes of data and the iterative nature of model development. Without version control, it becomes difficult to track changes to datasets, preprocessing steps, and model architectures.
- **Version control** ensures reproducibility, which is essential for research and development.





# Why is Version Control important ?

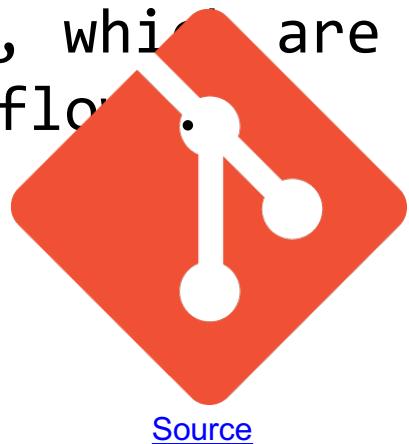
- Reproducibility allows other researchers to validate results, compare methods, and build upon existing work.
- Additionally, version control facilitates collaboration by allowing multiple team members to work on the same project simultaneously and manage changes efficiently.





# Choosing a Version control

- Several version control systems are available, including Git, SVN, and Mercurial.
- Git is by far the most popular and has become the standard for version controlling ML projects. As a distributed system, Git allows each user to have a complete copy of the repository, making collaboration and offline work easy.
- Git also offers powerful features like branching and merging, which are well-suited for managing complex ML workflows.



[Source](#)



[Source](#)





# Best practices for Version control

- Version control code using Git or a similar version control system.
- Version control data using a tool like DVC (see section 5 for more information on DVC).
- Track experiment results, such as performance metrics and visualizations, using a tool like MLflow or TensorBoard.



[Source](#)



# Recent Trends in MLOps



# AutoML

- AutoML (Automated Machine Learning) streamlines the setup of Machine Learning (ML) models by enhancing the ML pipeline and minimizing errors. This makes the process quicker and more efficient.





# AutoML Example

- Hugging face Auto NLP:

Hugging Face AutoNLP is an AutoML tool specifically designed for natural language processing (NLP) tasks, automating the process of training and fine-tuning NLP models.



[Source](#)





## AutoML Example

- H2o.ai (H2O AutoML)

H2o.ai offers a suite of AutoML tools, including H2O AutoML, which automates the machine learning model selection and hyperparameter optimization process.



[Source](#)





# AutoML Example

- Google cloud AutoML

Google Cloud AutoML provides a range of AutoML tools, such as AutoML Vision, AutoML Natural Language, and AutoML Tables, which automate the development of computer vision, NLP, and tabular data models.



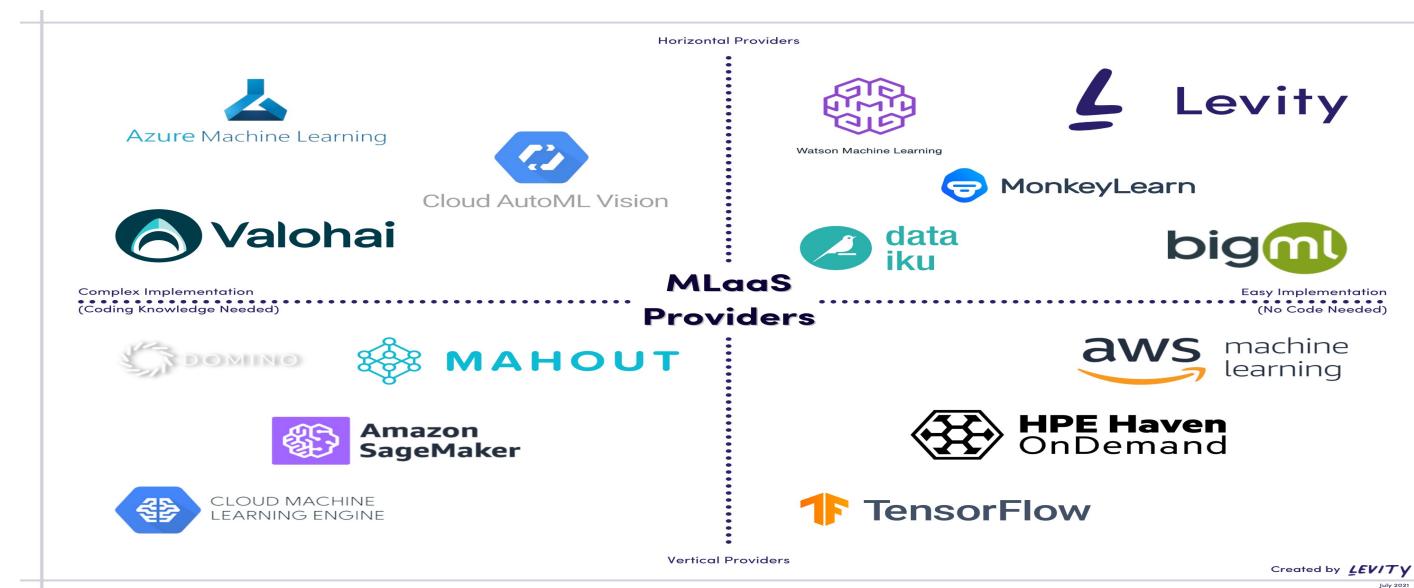
[Source](#)





# ML as a service

- **MLaaS** is the use of technology to automatically train, test, and deploy machine learning models for you, typically by leveraging an AI platform with offsite server farms (a “cloud”) to run them on behalf of the customer.





## MLaaS Example

### Amazon Machine Learning Services:

AWS Machine Learning Services let you create ML models without having to dabble with the algorithm. This makes it one of the most automated players in the MLaaS market and a service that's a good fit for deadline-centered businesses.



[Source](#)

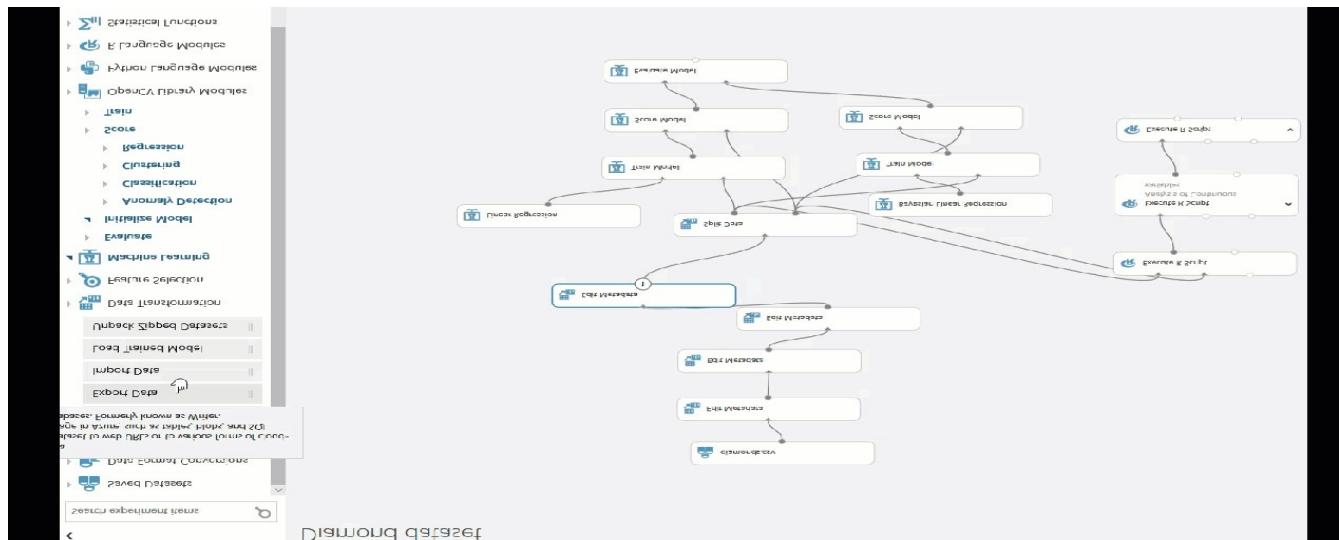




# MLaaS Example

# Microsoft Azure Machine Learning Studio:

A good selling point is ML Studio's variety of algorithms. It empowers users with over 100 methods that help with regression, classification, recommendation, Text Analysis, and anomaly detection.





# Future trends in MLOps

- **Explainable AI (XAI):**

Future MLOps workflows will likely integrate more tools and techniques for Explainable AI (XAI). These tools will help data scientists and stakeholders understand how ML models make decisions, thereby increasing transparency and trust in AI systems.

- **Model Governance and Compliance:**

As regulations around data privacy and model fairness continue to evolve, MLOps will increasingly emphasize model governance and compliance. Organizations will need to establish robust frameworks for tracking and auditing model behavior to ensure adherence to these regulations.





# References:

- <https://towardsdatascience.com/a-gentle-introduction-to-mlops-7d64a3e890ff>
- <https://www.almabetter.com/bytes/tutorials/mlops/version-control-ml-projects>
- <https://levity.ai/blog/mlaas-platforms-comparative-guide>
- <https://techvify-software.com/best-automl-tools/>
- <https://vedant-dhote.medium.com/top-10-automl-examples-819ac33bf852>
- <https://www.akkio.com/post/machine-learning-as-a-service-what-it-is-and-how-to-use-it#:~:text=deploy%20basic%20models.-,What%20is%20ML%20as%20a%20service%3F,on%20behalf%20of%20the%20customer>
- <https://medium.com/@vinodvamanbhat/devops-for-machine-learning-mlops-4fd280ca2ffd>

