

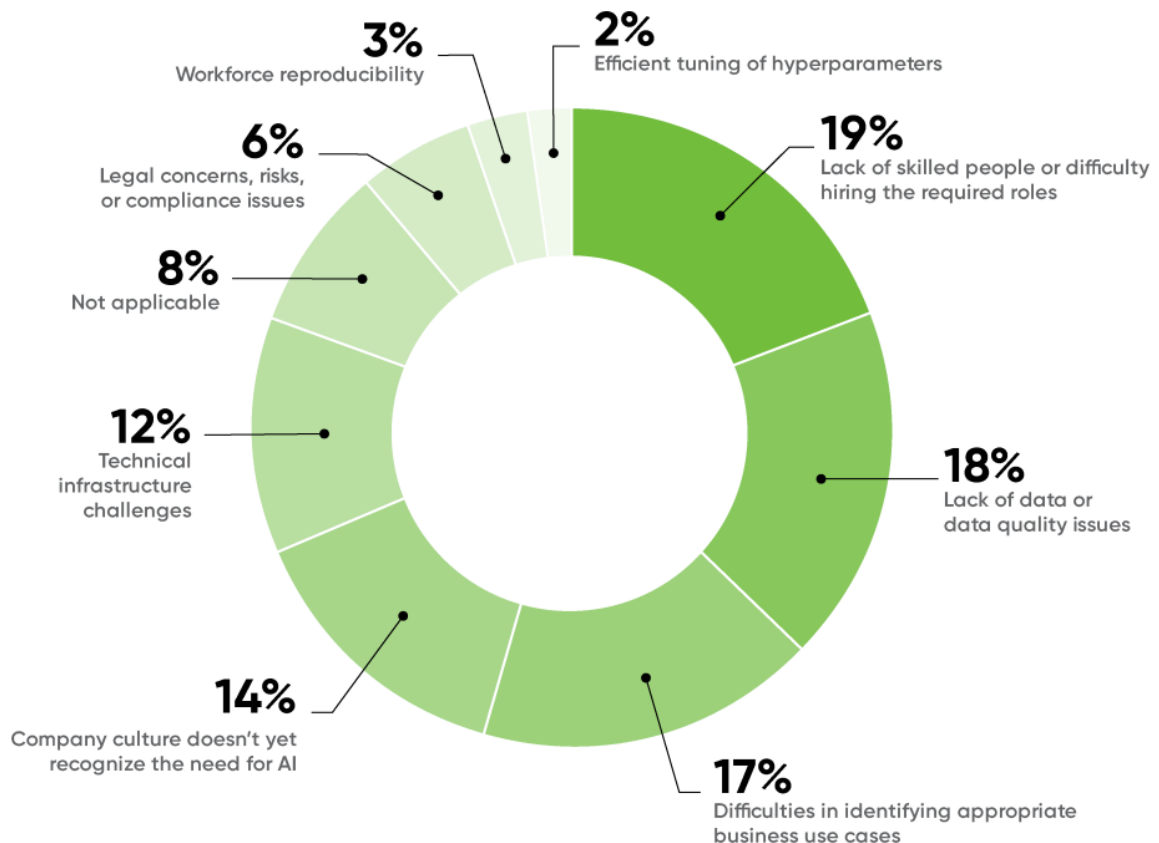
# Automate Your Process using Machine Learning Operations (MLOps)

## Overview

- The need for MLOps
- What is MLOps?!
- MLOps Example: CI/CD Pipelines
- CI/CD Pipelines Lab - GitHub Actions
- Optional Readings

## Challenges of AI Adoption to Organizations

- In 2021 survey studying the biggest problems facing companies trying to adopt AI, most common challenges included the lack of skilled people and difficulty in hiring (19%) and data quality (18%). It's no surprise that the demand for AI expertise has exceeded the supply, but it's important to realize that it's now become the biggest bar to wider adoption
- Note that about 12% of the AI projects face technical infrastructure challenges. Therefore, **MLOps** was introduced to help addressing such challenges.



AI Bottlenecks (from AI Adoption in the Enterprise (2021))

# Machine Learning Processes in the Enterprise



## What is MLOps?

- MLOps is a set of practices that aims to deploy and maintain machine learning models in production reliably and efficiently.
- The objective of an MLOps team is to **automate the deployment of ML models into the core software system or as a service component.**
- **Why do we need to automate the model deployment?** Because ML models and data change.

## Why do ML Models and Data Change? What are the common reasons?

- ML models can be retrained based upon new training data.
  - Models may be retrained based upon new training approaches.
  - Models may be self-learning.
  - Models may degrade over time.
  - Models may be deployed in new applications.
  - Models may be subject to attack and require revision.
- 
- Models can be quickly rolled back to a previous serving version.
  - Corporate or government compliance may require audit or investigation on both ML model or data, hence we need access to all versions of the productionized ML model.
  - Data may reside across multiple systems.
  - Data may only be able to reside in restricted jurisdictions.
  - Data storage may not be immutable.
  - Data ownership may be a factor.

# Automation

The level of automation of the Data, ML Model, and Code pipelines determines the maturity of the ML process. With increased maturity, the velocity for the training of new models is also increased. As stated earlier, the goal of MLOps is to automate the deployment of your ML models into the main system or as a service component.

## Levels of Automation

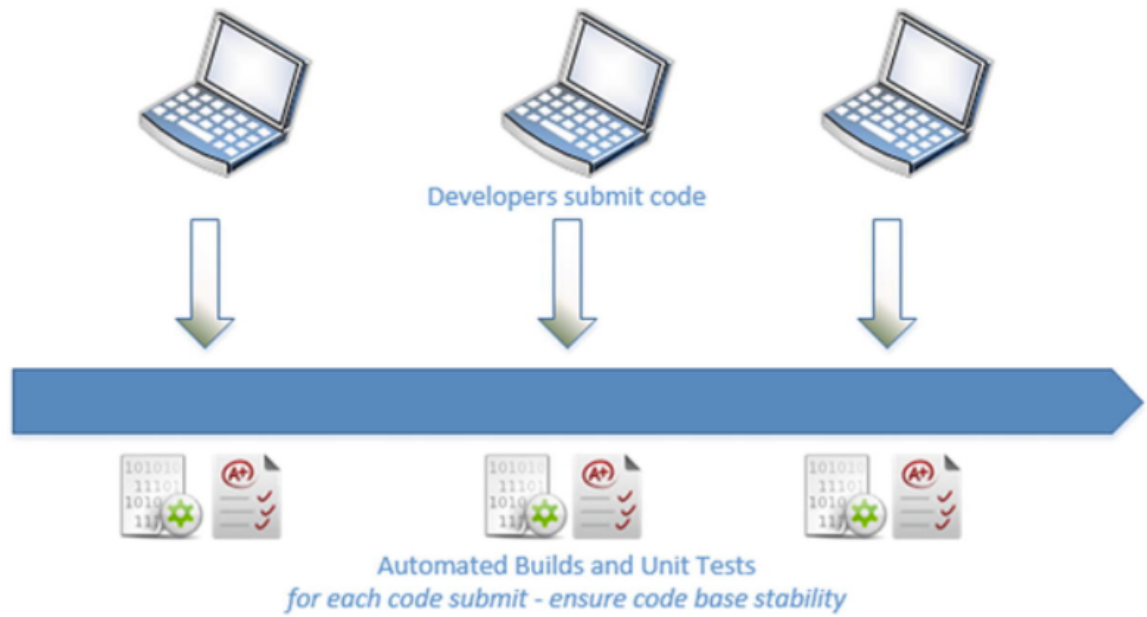
- **Manual process.** This is a typical data science process, which is performed at the beginning of implementing ML. This level has an experimental and iterative nature. Every step in the process, such as data preparation and validation, model training and testing, are executed manually. The common way to process is to use Rapid Application Development (RAD) tools, such as Jupyter Notebooks.
- **ML pipeline automation.** The next level includes the execution of model training automatically. Whenever new data are available, the process of model retraining is triggered. This level of automation also includes data and model validation steps.
- **CI/CD pipeline automation.** In the final stage, we introduce a CI/CD system to perform fast and reliable ML model deployments in production. The core difference from the previous step is that we now automatically build, test, and deploy the Data, ML Model, and the ML training pipeline components.

## CI/CD Pipeline Automation

- In software development, multiple team members develop code and contribute to creating the software's functionality. When multiple people contribute to a code base, it is important to maintain its integrity and ensure that any team member can retrieve the latest version and build and run it locally.
- Two important aspects should be maintained to assure the code base's stability. The first aspect is to ensure that the code is compiling without errors. The second aspect is to ensure that all unit tests validating code behavior pass, including the latest code changes, at a very high percentage.

A build pipeline should be defined to compile each check-in/commit to the code base and then execute all unit tests to validate the code base to ensure its stability; this is generally known as a CI build. If the build successfully compiles and all the unit tests pass, it generates and publishes output that is deployed to a target environment

# Continuous Integration/Continuous Delivery



CI/CD

# Example of CI/CD Pipelines: GitHub Actions

## Introduction to GitHub Actions

- GitHub Actions are a set of actions in a GitHub repository workflow. These actions allow you to customize and execute software development workflows. You can create actions or utilize existing actions and create and customize workflows to perform any job or automate software development life cycle processes, including CI/CD.
- Actions are individual tasks that can be combined to create a workflow. A workflow is one or more automated jobs with actions configured in a YAML file that can be stored in your GitHub repo. Let's discuss each key concept in more detail.

## GitHub Actions - Main Concepts

- Action
- Artifact
- Event
- GitHub Hosted Runner
- Job
- Task (or Step)
- Workflow
- Workflow File
- Workflow Run

### Action

The smallest building block of a workflow is an action, which can be identified as an individual task. These tasks or steps can be combined to create a job that can be executed in a workflow. Existing actions from the marketplace can create jobs and workflows, and you can customize or create your own actions. An action must be used as a step in a job to be used in a workflow.

You need to combine actions into a job to make up a workflow that can check out a repository, and build and publish artifacts.

### Artifact

The files generated when you build your software project or test your software project are artifacts. Artifacts may contain the binary packages required to deploy your software and any support files, such as configurations or infra-scripts required for deployment activities. Artifacts can be created in one job and used in another job for deployment actions in a workflow.

## Event

An event triggers a workflow in GitHub Actions. Once a code change is pushed, or a pull request is made, an event can be set up in GitHub Actions to trigger the workflow. You can configure external triggers using a repository dispatch webhook. You can also use many other webhooks, such as deployment, workflow dispatch, and check runs.

## GitHub Hosted Runners

Hosted runners are machines that are supported in Windows, Linux, and macOS. These machines are preinstalled with commonly used software. You cannot customize a hosted runner's hardware configuration. A GitHub-hosted runner virtual environment contains hardware configuration, operating system, and installed software information. You can find installed software and OS information at <https://github.com/actions/virtual-environments/tree/main/images>

## Job

A job is a set of steps set up to run in a single runner. A job can comprise one or more actions. Jobs can run in parallel in a single workflow, and you can set up dependencies to run jobs sequentially. A dependent job will not run if the dependencies fail. Each job in a workflow runs in a fresh instance of a runner. A job should specify the runner's OS and the version.

## Task

A task that is an action or a command and it is identified as a step. All steps in a job run in the same runner. The file system's information is shared with multiple steps (actions and commands) in a single job.

## Workflow

In a GitHub repo, the process set up in a YAML file defining the build, test, package, or deployment jobs is called a workflow. A workflow is scheduled to run based on triggers/events. A workflow may contain one or more jobs set up to run sequentially or in parallel, depending on the requirements.

## Workflow File

The YAML file stored in the github/workflows folder in your GitHub repository is a workflow file. The workflow file is defined with the workflow, which runs based on the events.

## Workflow Run

A workflow executes based on the preconfigured triggers/events. Logs tell you about failed jobs or successful job activities. Each workflow runs logs for the jobs and actions or commands executed.

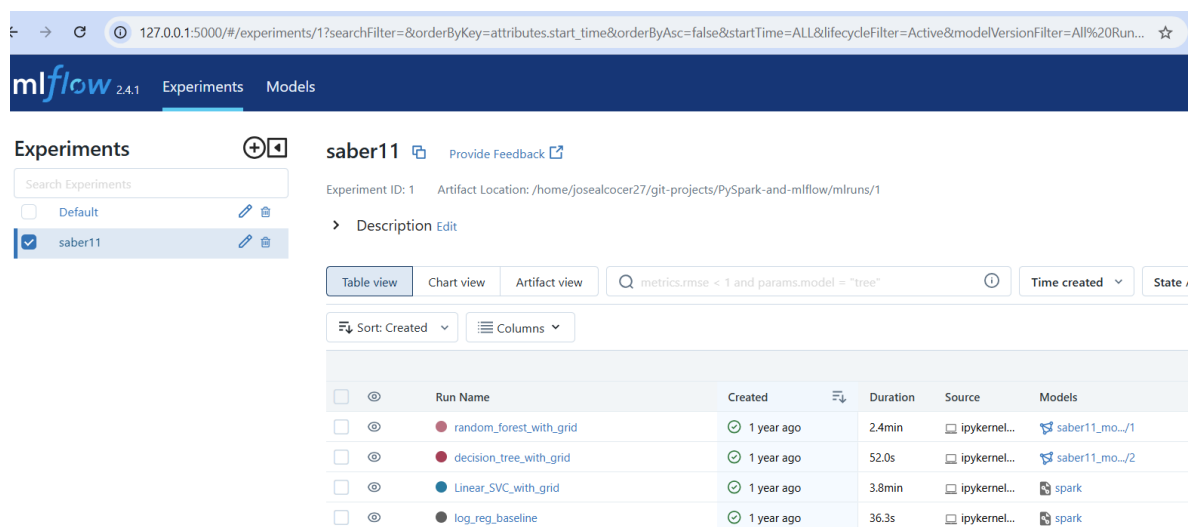
# Lab

- Let's use GitHub actions to build CI/CD pipeline that will compile your TF model upon pushing any new changes to your GitHub repository
- Our repository URL is: [https://github.com/mohamedfarag/test\\_repo](https://github.com/mohamedfarag/test_repo)
- The main python file is: **run\_ml\_model.py**
- You will need to create a file containing all the packages that should be installed and name it **requirements.txt**
- Your workflow file can be found under **.github/workflows** folder

## How to track ML Models?!

There are several tools to do so. One of the most popular and open source tools is MLflow. You can install MLflow locally by following the instructions in the following GitHub repository:

<https://github.com/mohamedfarag/local-mlflow>



Run Name	Created	Duration	Source	Models
random_forest_with_grid	1 year ago	2.4min	ipykernel...	saber11_mo.../1
decision_tree_with_grid	1 year ago	52.0s	ipykernel...	saber11_mo.../2
Linear_SVC_with_grid	1 year ago	3.8min	ipykernel...	spark
log_reg_baseline	1 year ago	36.3s	ipykernel...	spark

MLFlow on Local Machine

## Optional Readings

- [Evaluate your system automatically by calculating its ML Test Scores](#)
- [ML Model Management](#)
- [Evaluate Regression Models](#)
- [Evaluate Classification Models](#)