We will learn what is MLOps or Machine Learning Operations. This article is for people who want to understand how an ML model is deployed to production, the stages, the process, and the tears it involves.

Let us begin!

What is MLOps?

Machine Learning Operations involves a set of processes or rather a sequence of steps implemented to deploy an ML model to the production environment. There are several steps to be undertaken before an ML Model is production-ready. These processes ensure that your model can be scaled for a large user base and perform accurately.

Why do we need MLOps?

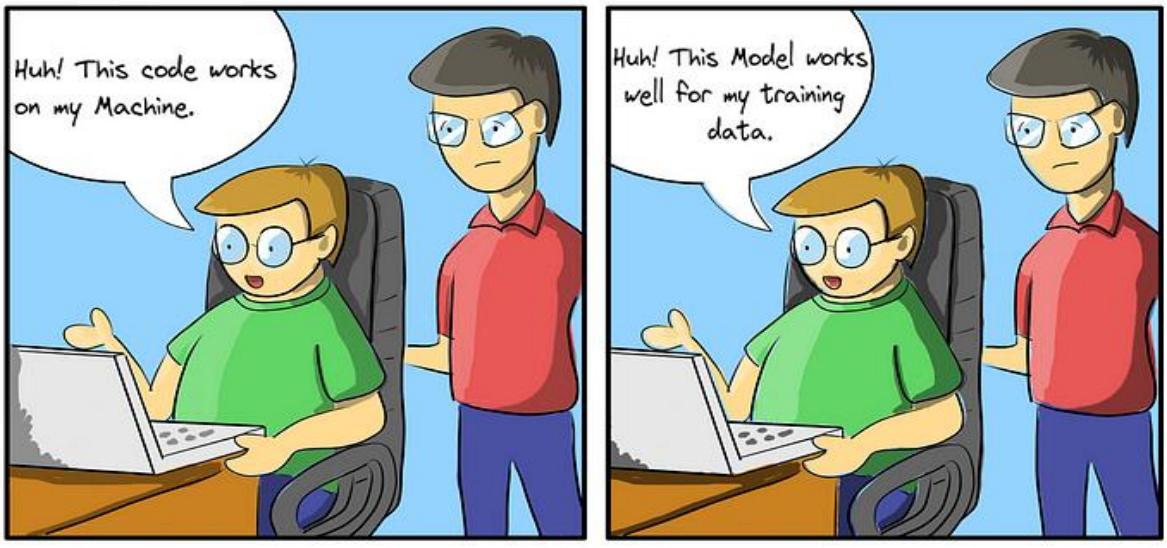
Creating an ML model that can predict what you want it to predict from the data you have fed is easy. *However, creating an ML model that is reliable, fast, accurate, and can be used by a large number of users is difficult.*

The necessity of MLOps can be summarized as follows:

- ML models ***rely on a huge amount of data***, difficult for a single person to keep track of.

- ***Difficult to keep track of parameters*** we tweak in ML models. Small changes can lead to enormous differences in the results.
- We have to keep track of the features the model works with, feature engineering is a separate task that contributes largely to model accuracy.
- Monitoring an ML model isn't like monitoring a deployed software or web app.
- ***Debugging an ML model is an extremely complicated art***
- Models rely on real-world data for predicting, ***as real-world data changes, so should the model.*** This means we have to keep track of new data changes and make sure the model learns accordingly.

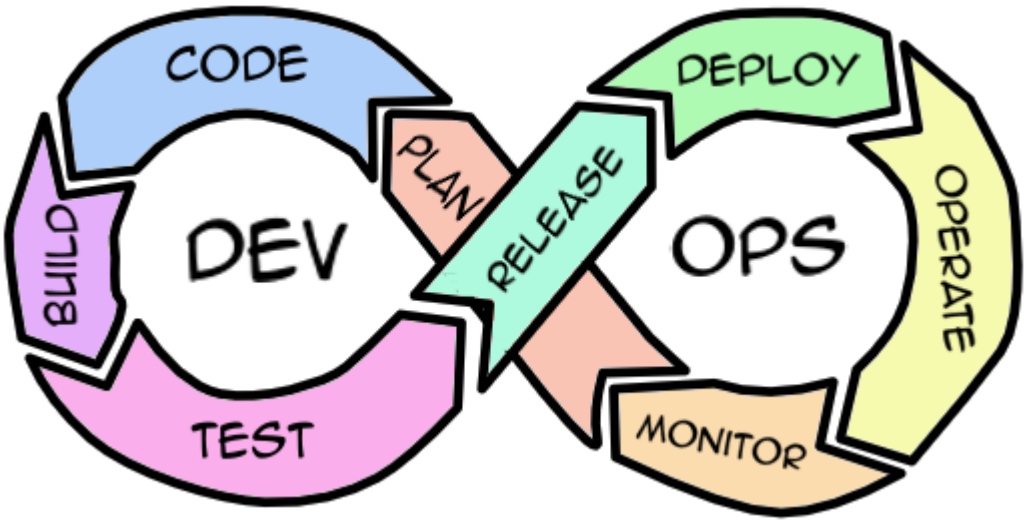
Remember the old excuse Software Engineers used. We want to avoid it.



Software Engineers vs ML Engineers

DevOps vs MLOps

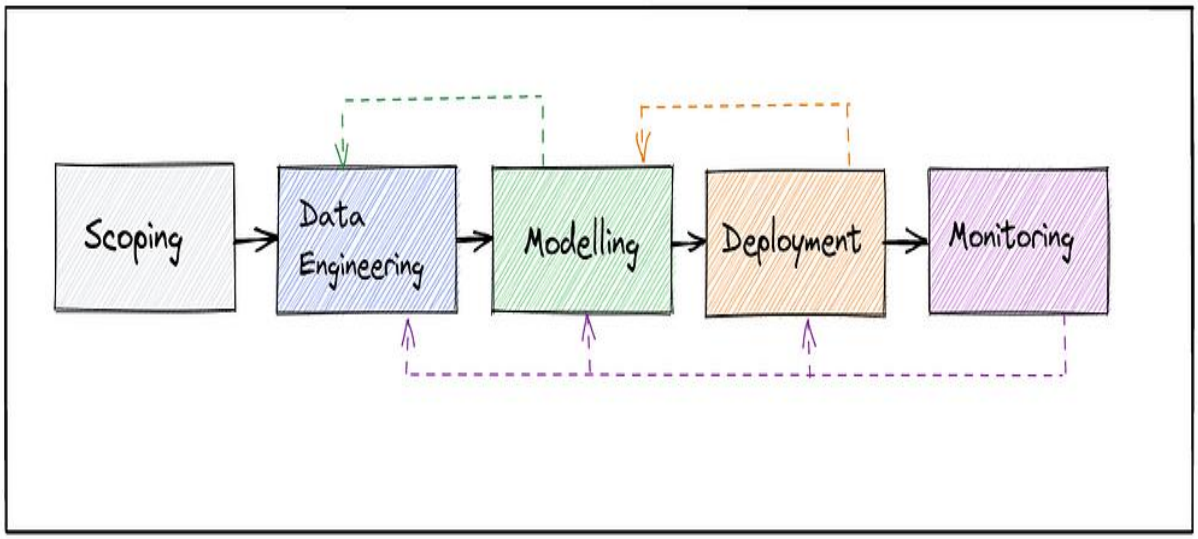
You must have heard of good old DevOps. The process to build and deploy software applications. You might wonder how MLOps is different.



DevOps Cycle (Image by Author)

The DevOps stages are targeted for developing a software application. You plan the features of the application you want to release, write code, build the code, test it, create a release plan and deploy it. You monitor the infrastructure where the app is deployed. And this cycle continues until the app is fully built.

In MLOps, things are different. We implement the following stages



ML Project Lifecycle

Scoping — We define the project, ***check if the problem requires Machine Learning to solve it***. Perform requirement engineering, check if the relevant data is available. Verify if the data is non-biased and reflects the real-world use case.

Data Engineering — This stage involves collecting data, establishing baselines, cleaning the data, formatting the data, labelling, and organizing the data.

Modelling — Now we come to the coding part, here we create the ML model. We train the model with the processed data. Perform error analysis, define error measurement, and track the model performance.

Deployment — Here we package the model, deploy it in the cloud or on edge devices as necessary. Packaging could be — model wrapped with an API server exposing REST or gRPC endpoints, a docker container deployed on cloud infrastructure, deployed on server-less cloud platform, or a mobile app for edge-based models.

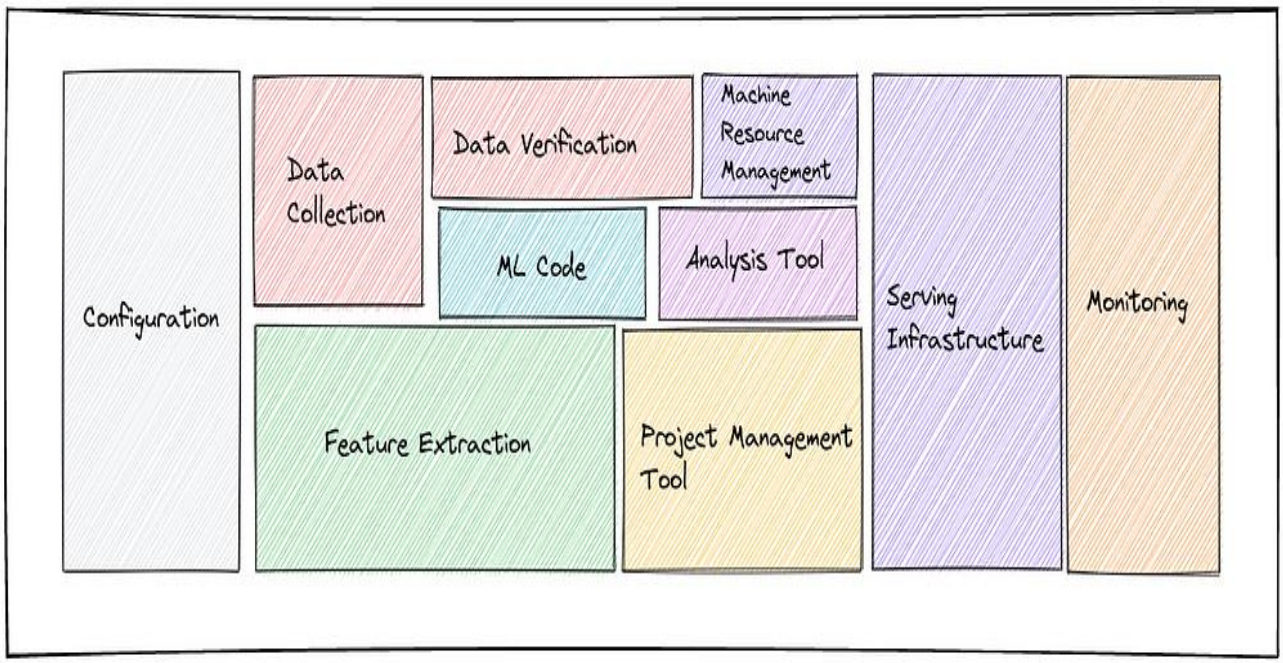
Monitoring — Once the deployment is done, we rely on a monitoring infrastructure to help us maintain and update the model. This stage has the following components:

1. Monitor the infrastructure where we deploy — for load, usage, storage, and health. This tells us about the environment where the ML model is deployed.
2. Monitor the model for its performance, accuracy, loss, bias, and data drift. This informs us if the model is performing as expected, valid for real-world scenarios or not.

There will sometimes be a feedback loop as some models might require learning from the user inputs and predictions it makes. This lifecycle is valid for most of the ML use cases.

Equipped with the knowledge of the basic lifecycle of an ML project, let’s take a look at how the infrastructure scene is on the ML side.

ML Production Infrastructure



ML Infrastructure

Now we learn what infrastructure setup we would need for a model to be deployed in production. You can see in the above picture, ML code is only a small part of it. Let us understand the components one by one.

Data Collection — This step involves collecting data from various sources. ML models require a lot of data to learn. Data collection involves consolidating all kinds of raw data related to the problem. i.e Image classification might require you to collect all available images or scrape the web for images. Voice recognition may require you to collect tons of audio samples.

Data Verification — In this step we check the validity of the data, if the collected data is up to date, reliable, and reflects the real world, is it in a proper consumable format, is the data structured properly.

Feature Extraction — Here, we select the best features for the model to predict. In other words, your model may not require all the data in its entirety for discovering patterns, some columns or parts of data might be not used at all. Some models perform well when a few columns are dropped. We usually rank the features with importance, features with high importance are included, lower ones or near zero ones are dropped.

Configuration — This step involves setting up the protocols for communications, system integrations, and how various components in the pipeline are supposed to talk to each other. You want your data pipeline to be connected to the database, you want your ML model to connect to database with proper access, your model to expose prediction endpoints in a certain way, your model inputs to be formatted in a certain way. All the necessary configurations required for the system need to be properly finalized and documented.

ML Code — Now we, come to the actual coding part. In this stage, we develop a base model, which can learn from the data and predict. There are tons of ML libraries out there with multiple language support. Ex: tensorflow, pytorch, scikit-learn, keras, fast-ai and many more. Once we have a model, we start improving its performance by tweaking the hyper-parameters, testing different

learning approaches until we are satisfied that the model is performing relatively better than its previous version.

Machine Resource Management — This step involves the planning of the resources for the ML model. Usually, ML models require heavy resources in terms of CPU, memory, and storage. Deep learning models are dependent on GPU and TPU for computation. Training ML models involves cost in terms of time and money. Slower CPUs involve more time, Powerful CPUs are pricier. The larger the model, the bigger the storage you will have to invest in.

Analysis Tool — Once your model is ready, how do you know if the model is performing up to mark. We decide on model analysis in this stage. How do we compute loss, what error measurement should we use, how do we check if the model is drifting, is the prediction result proper, has the model been overfitted or underfit? Usually, the libraries with which we implement the model ship with analysis kits and error measurements.

Project Management Tool — Tracking an ML project is very important. It's easy to get lost and mess up while dealing with huge data, features, ML code, resource management. Luckily there are a lot of project management tools out on the Internet to help us out.

Serving Infrastructure — Once the model is developed, tested, and ready to go, we need to deploy it somewhere the users can access it. The majority of the models are deployed on cloud. Public

cloud providers like AWS, GCP, and Azure even have specific ML-related features for easy deployment of models. Depending on the budget you can select the provider suited for your needs.

If we are dealing with an edge-based model, we need to decide on how the ML model can be used, it could be a mobile application for use cases like image recognition, voice recognition. We could also have a custom chip and processor for certain use case like autonomous driving as in the case of Tesla. Here we have to take into account how much computing capability is available and how large is our model size.

Monitoring — We need to implement a monitoring system to observe our deployed model and the system on which it runs. Collecting model logs, user access logs, and prediction logs will help in maintaining the model. There are several monitoring solutions like greylog, elasticstack, and fluentd available. Cloud providers usually ship their own monitoring systems.

A Walk-through

Now that we have understood how the ML project lifecycle works, how the infrastructure scene is in an ML production. We will learn how the ML model is deployed in production.

To understand this we will look at Jen and her quest for ML Engine.

Jen has a huge pumpkin patch, every year she sells the pumpkins to townsfolk and local pumpkin spice latte factory. Since she has a

huge demand every year, it became tedious for her to look at every pumpkin and check if it is good or bad.

So she approaches you to help her develop an ML Engine that will help her predict if a given pumpkin is good or bad.

Jen needs help classifying her pumpkins. (Image by Author)

Off the bat, this is a simple **classification problem.**

Let's discuss our approach to solve this problem.

1. First, we collect all the info about the pumpkins in Jen's patch. All the photos of good pumpkins, ok pumpkins, and bad pumpkins. We then ask the good townsfolk to send in pictures of the pumpkins they had bought from Jen. (Some of them send pictures of watermelon! Shame on them) **This step is EDA + Compiling dataset.**

2. Now that we have a good collection of pictures. It's time to label these, with Jen's help we label several hundred

pictures. We check the resolution of the pictures, set a standard resolution, discard the low-quality images, format the image contrast and brightness for better readability. **This step is Data Preparation**

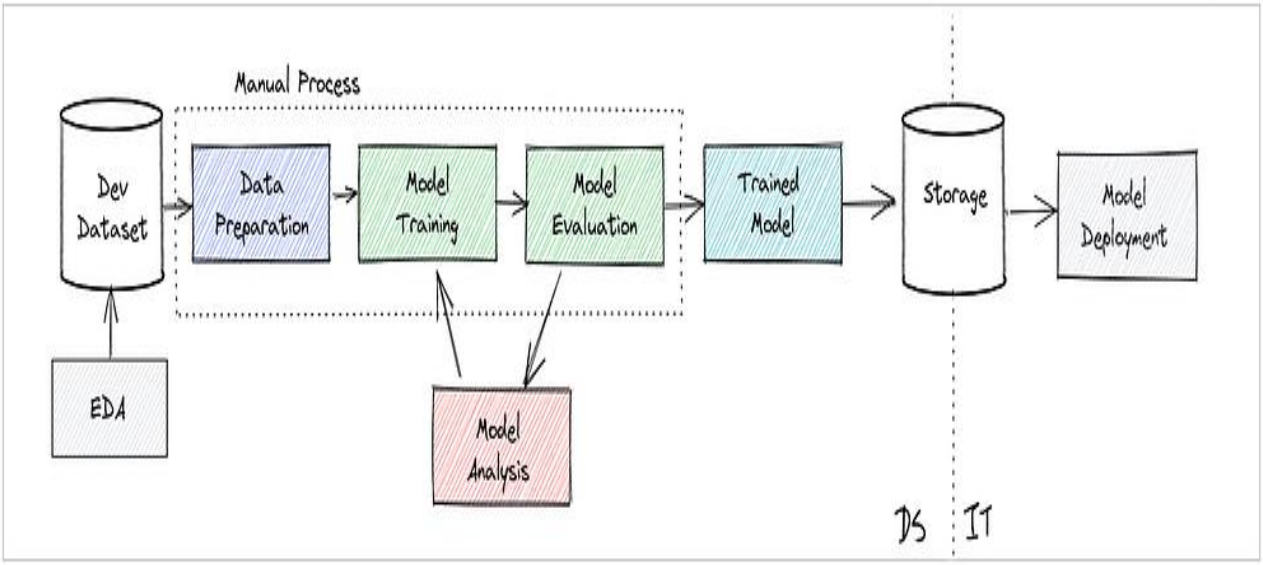
3. Now we train a tensorflow model to classify the images. Say we use a sequential neural net with ReLU activation. We define 1 input layer, 2 hidden layers, and 1 output layer, just a basic Convolution Neural Net. We split our image dataset into training and testing sets, provide the training data as input to the ConvoNet. The model is trained. **This step is Model Training.**

4. Once the model is trained, we evaluate the model by using the testing dataset. Based on the prediction, we compare the result and check for the accuracy of the prediction. **This step is Model Evaluation.**

5. We tweak the hyper parameters of the model, to increase the accuracy, retrain the model and evaluate it again. This iteration is done until we are satisfied that the model is good enough for Jen. **This step is Model Analysis.**

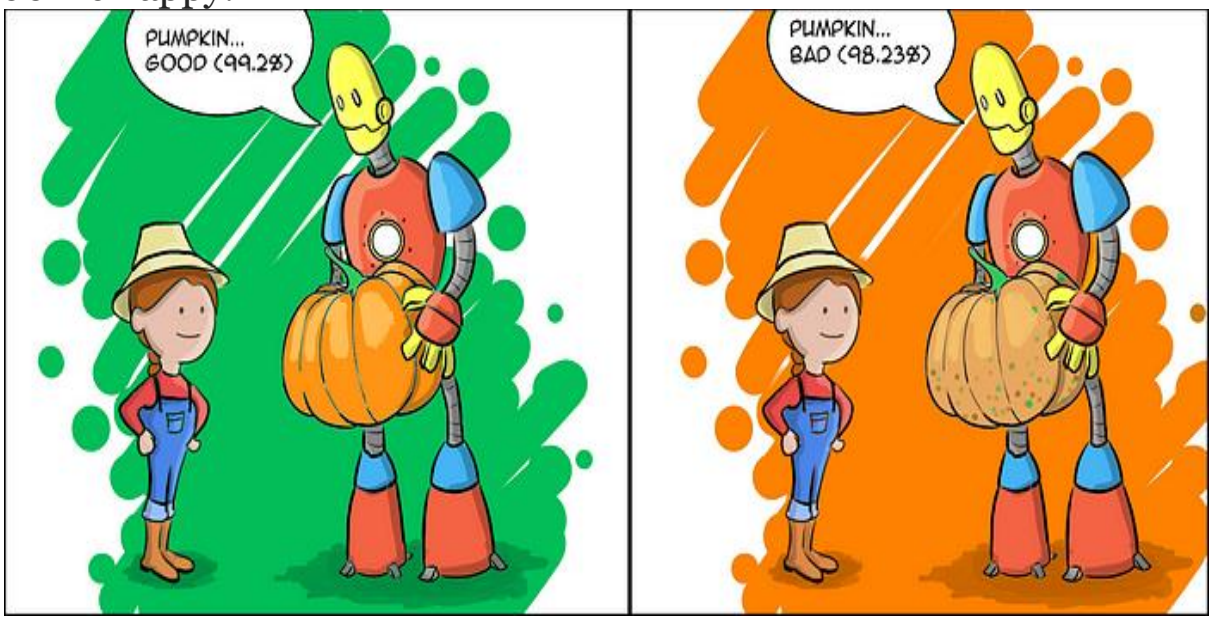
6. Now we have our working model, we deploy it so that Jen can use this ML Engine for her daily work. We create a server in the cloud with prediction APIs, create an app or a website where she can upload the images and get results in real-time. **This is Model Deployment.**

We have done all the work manually, from data preparation to deployment. **Congratulations! This process of MLOps is called Level-0.** We have achieved our deployment, but all things are done manually. You can refer to the diagram below.



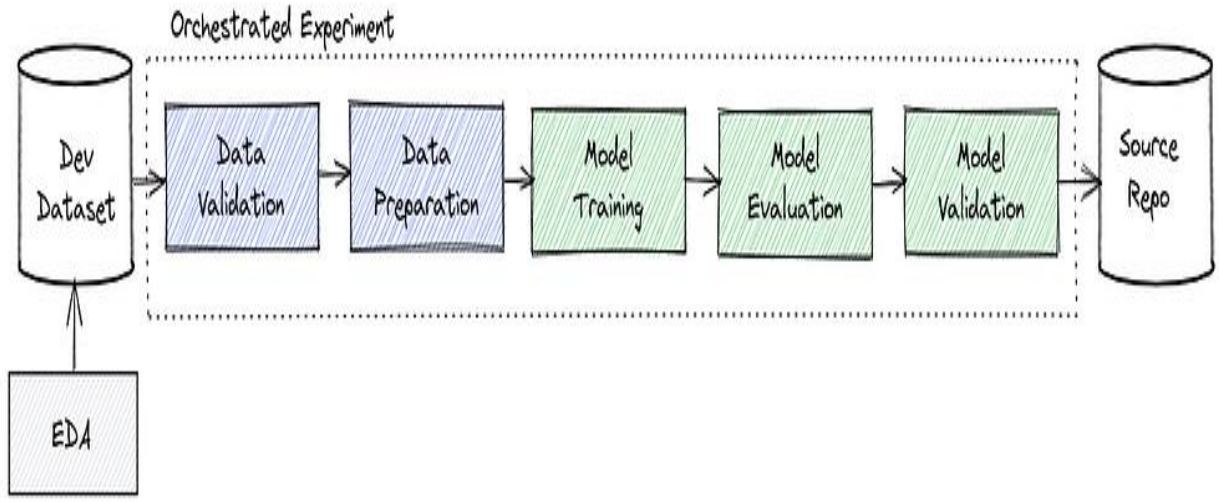
MLOps Level-0

Jen is happy.



Classification

Now the demand has begun to rise. You cannot train the model every day manually. So you create an automation pipeline, to validate data, prep it, and train the model. You also try to fetch the best available model, by comparing multiple error metrics. The pipeline takes care of it all. **This process is Level-1 of MLOps.** Here the training and analysis of the model are taken care of automatically. You just have to check if proper data is available and make sure there isn't a skewed dataset so that the model is trained properly.



MLOps Level-1

Most companies achieve this level of MLOps. This is also achievable by an individual data scientist or ML engineer. This is good enough when you test the model in your development environment.

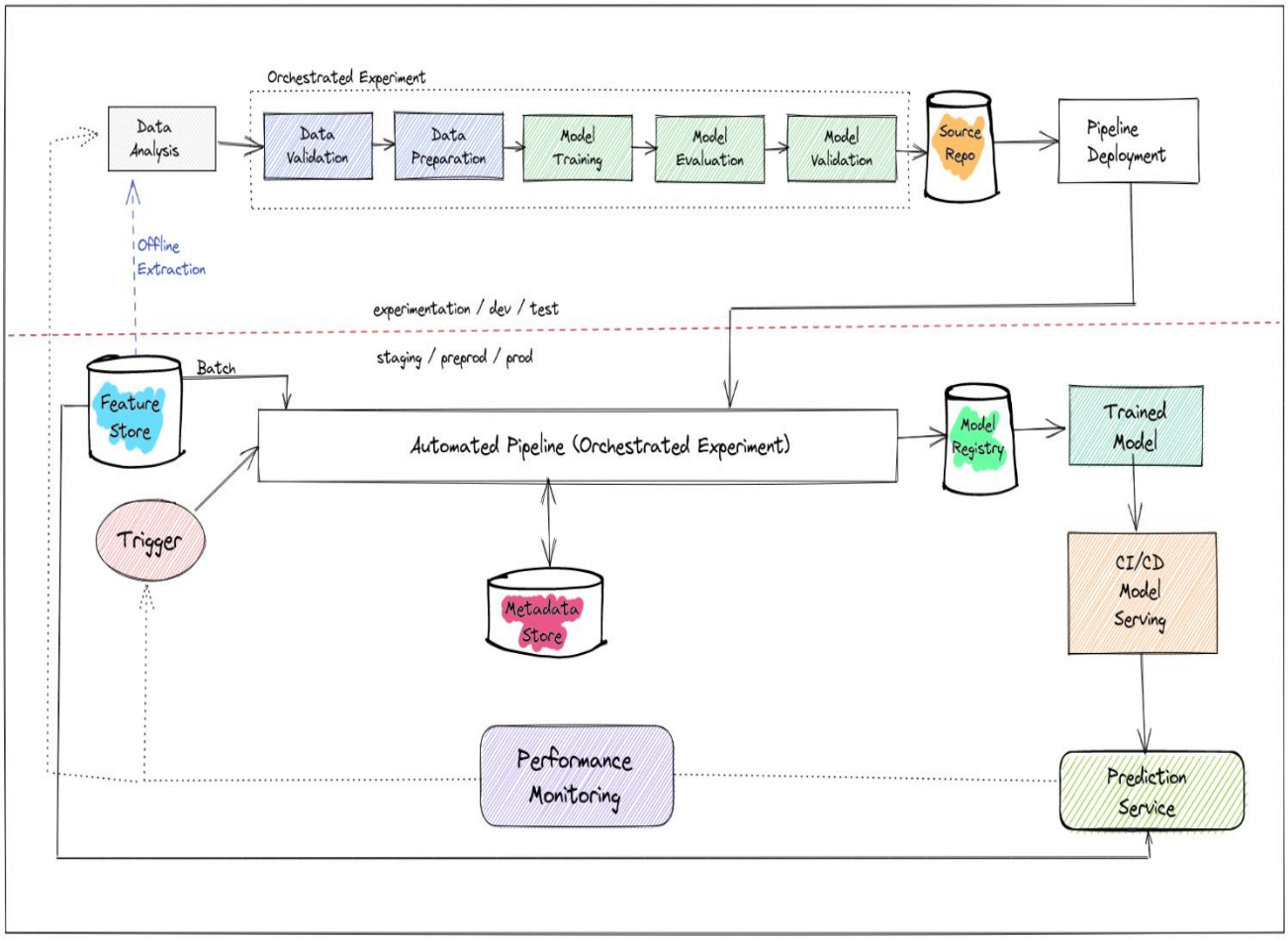
Let's now ask ourselves a few questions.

- Is your model able to replicate the result with different varieties of pumpkins?

- When new data is added to the dataset, is your model able to retrain?
- Can your model be used by hundreds of thousands of people at once? Scaled well?
- How do you keep track of models when you deploy them across a large region or even across the globe?

This leads us to **Level-2.**

It's time to **look at the bigger picture…**



MLOps Level-2

Let’s break down the process.

- Everything we do ***above the red line***, in the flowchart — is ***Level-1***.
- This entire ***Orchestrated Experiment*** is now part of the ***Automated ML Pipeline.***
- We introduce a ***Feature Store***, which pulls data from various sources and transforms the data to features required by the model. The ML pipeline uses the data from the Store in batches.
- The ML pipeline is connected to a ***Metadata Store***. Think of it as bookkeeping, since you don’t train the model manually — this store has the records of each stage in the pipeline. Once a stage is completed, the next stage looks up the record list, finds the previous stage records, picks up from there.
- The models are then stored in a ***Model Registry.*** We have a bunch of models with various accuracy stored here. Based on the requirement, the appropriate model is then sent to a ***CI/CD pipeline*** which deploys it as a ***Prediction Service.*** Authorized users are able to access the prediction service when desired.
- This system is monitored for performance. Say you have a new bunch of genetically modified pumpkins. ***Your model isn’t aware of this. This is a new dataset***

with a high probability of being wrongly classified. This drop in performance would set a trigger, that would lead to the retraining of the model on the new data.

- This cycle continues.

We retrain the model, when there is a drop in performance or there is new data available. It's good to keep the model up to date so that the real-world changes are reflected in the model. ***Ex: Your model shouldn't be recommending cassette tapes, when the world has moved onto digital streaming.***

Where to go from here?

We covered what is MLOps? Why would you use it? What would the production infrastructure setup look like? And, once you have the infrastructure, how would you implement it — the process.

You can start by creating simple models and automating the steps. Remember, it is an iterative process, will take time to get it right.

Do check out some of the MLOps Tools like MLFlow, Seldon Core, Metaflow, and Kubeflow Pipelines.