See our other MLOps content

Guides

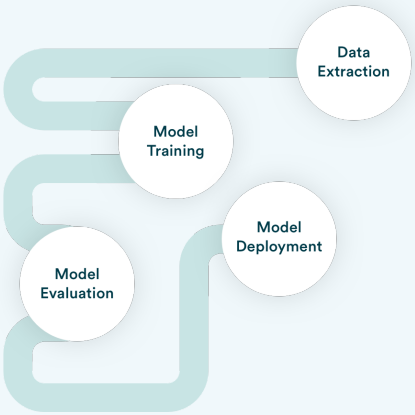
What is MLOps?
MLOps Platform: Build or Buy
MLOps Platforms Compared

Resources

The MLOps eBook
The MLOps Stack Template

Topics

Machine Learning Pipeline



What Is a Machine Learning Pipeline?

A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment.

For data science teams, the production pipeline should be the central product. It encapsulates all the learned best practices of producing a machine learning model for the organization's use-case and allows the team to execute at scale. Whether you are maintaining multiple models in production or supporting a single model that needs to be updated frequently, an end-to-end machine learning pipeline is a must.

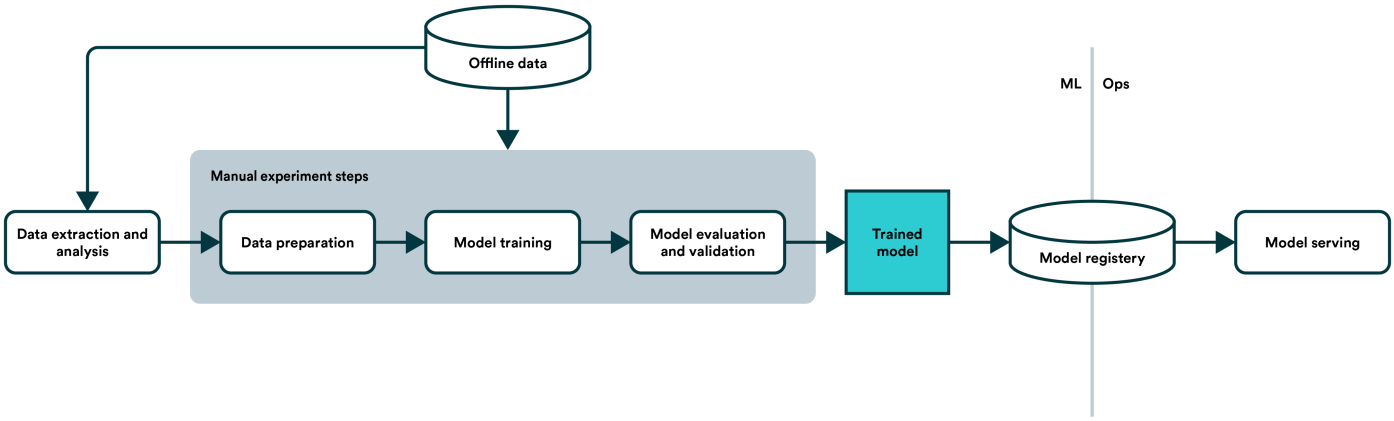
Topics covered on this page

What Are the Benefits of a Machine Learning Pipeline?

It is beneficial to look at the stages which many data science teams go through to understand the benefits of a machine learning pipeline. Implementing the first machine learning models tends to be very problem-oriented, and data scientists focus on producing a model to solve a single business problem, for example, classifying images.

The Manual Cycle

Teams tend to start with a manual workflow, where no real infrastructure exists. The data collection, data cleaning, model training and evaluation are likely written in a single notebook. The notebook is run locally to produce a model, which is handed over to an engineer tasked with turning it into an API endpoint. Essentially, in this workflow, **the model is the product.**



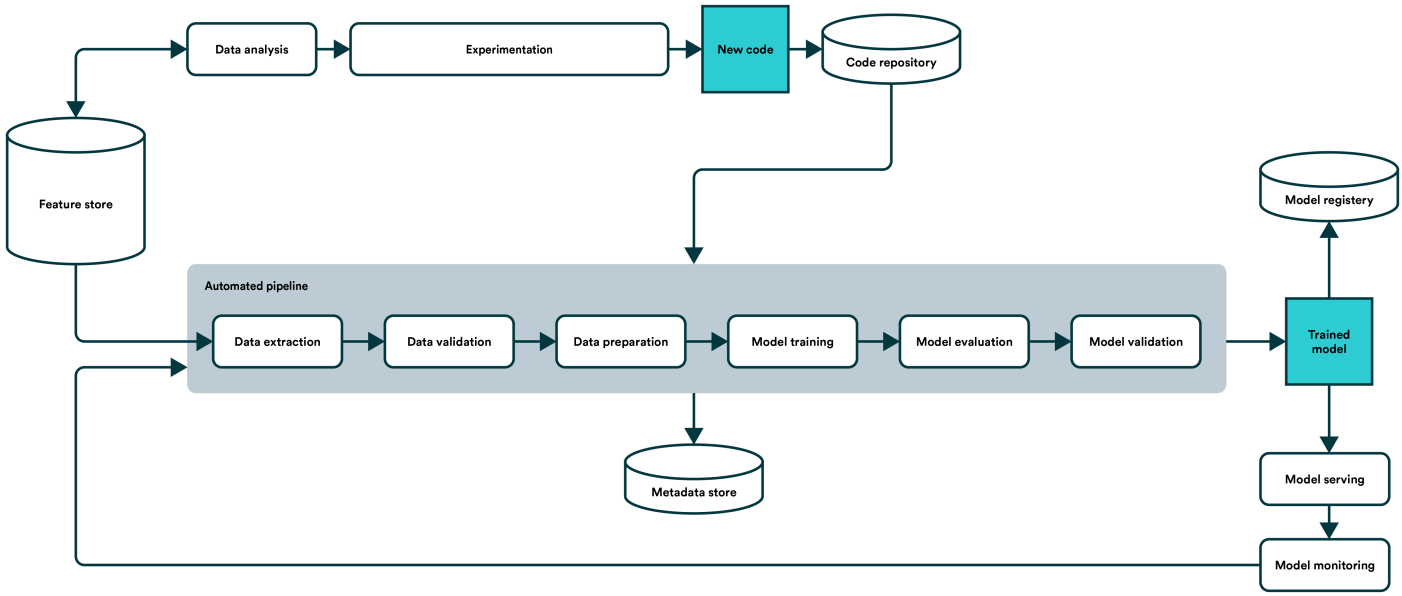
The manual workflow is often ad-hoc and starts to break down when a team begins to speed up its iteration cycle because manual processes are difficult to repeat and document. A code monolith, even in notebook format, tends to be unsuitable for collaboration.

Characteristics of a manual ML pipeline:

- **The model is the product**
- Manual or script-driven process
- A disconnect between the data scientist and the engineer
- Slow iteration cycle
- No automated testing or performance monitoring
- No version control

The Automated Pipeline

Once teams move from a stage where they are occasionally updating a single model to having multiple frequently updating models in production, a pipeline approach becomes paramount. In this workflow, you don't build and maintain a model. You develop and maintain a pipeline. **The pipeline is the product.**



An automated pipeline consists of components and a blueprint for how those are coupled to produce and update the most crucial component – the model.

In the automated workflow, solid engineering principles become more into play. The code is split into more manageable components, such as data validation, model training, model evaluation, and re-training triggering.

The system offers the ability to execute, iterate, and monitor a single component in the context of the entire pipeline with the same ease and rapid iteration as running a local notebook cell on a laptop. It also lets you define the required inputs and outputs, library dependencies, and monitored metrics.

This ability to split the problem solving into reproducible, predefined, and executable components forces the team to adhere to a joined process. A joined process, in turn, creates a well-defined language between the data scientists and the engineers and also eventually leads to an automated setup that is the ML equivalent of continuous integration (CI) – a product capable of auto-updating itself.

Characteristics of an automated ML pipeline:

- **The pipeline is the product**
- Fully automated process
- Co-operation between the data scientist and the engineer
- Fast iteration cycle
- Automated testing and performance monitoring
- Version-controlled

The Pipeline Approach Allows Machine Learning to Scale

Transitioning from a manual cycle to an automated pipeline may have many iterations in between depending on the scale of your machine learning efforts and your team composition. Ultimately, the purpose of a pipeline is to allow you to increase the iteration cycle with the added confidence that codifying the process gives and to scale how many models you can realistically maintain in production.

What to Consider when Building a Machine Learning Pipeline?

As stated above, the purpose is to increase the iteration cycle and confidence. Your starting point may vary; for example, you might have already structured your code. The following four steps are an excellent way to approach building an ML pipeline:

1. **Build every step into reusable components.**
   Consider all the steps that go into producing your machine learning model. Start with how the data is collected and preprocessed, and work your way from there. It's generally encouraged to limit each component's scope to make it easier to understand and iterate.

2. **Don't forget to codify tests into components.**
   Testing should be considered an inherent part of the pipeline. If you, in a manual process, do some sanity checks on how the input data and the model predictions should look like, you should codify this into a pipeline. A pipeline gives opportunities to be much, much more thorough with testing as you will not have to perform them manually each time.

3. **Build every step into reusable components.**
   There are many ways to handle the orchestration of a machine learning pipeline, but the principles remain the same. You define the order in which the components are executed and how inputs and outputs run through the pipeline. We, of course, recommend using Valohai for building your pipeline. The next section is a short overview of how to build a pipeline with Valohai.

4. **Automate when needed.**
   While building a pipeline already introduces automation as it handles the running of subsequent steps without human intervention, for many, the ultimate goal is also to automatically run the machine learning pipeline when specific criteria are met. For example, you may monitor model drift in production to trigger a re-training run or – simply do it more periodically, like daily.

Depending on your specific use case, your final machine learning pipeline might look different. For example, you might train, evaluate and deploy multiple models in the same pipeline. There are common components that are similar in most machine learning pipelines.

Examples of different components:

- Data validation
- Data cleanup
- Model training
- Model evaluation
- Model validation
- Re-training trigger

In addition, the pipeline also has static components like:

- Feature store
- Deployment endpoint
- Metadata store
- Source code version control

How to Build a Machine Learning Pipeline with Valohai?

In Valohai, pipelines are DAGs (Directed Acyclic Graph).

- **Directed** = Data flows one way
- **Acyclic** = Backward feedback loops are not allowed
- **Graph** = Nodes (step) are connected by connections (data flow)

The beneficial consequence of using DAGs is that every node executes once and only once. It means that every single node only has one set of inputs and outputs per running pipeline. This makes the pipeline simpler to define, understand, and debug.

Defining Valohai pipeline

Valohai pipelines are defined through YAML. You specify steps and connections between them.

Here is an example pipeline that:

1. Preprocesses dataset
2. Trains a model

```
- step:
  name: preprocess
  image: tensorflow/tensorflow:1.13.1-gpu-py3
  command: python preprocess.py
  inputs:
    - name: train-images
    - name: train-labels
    - name: test-images
    - name: test-labels
- step:
  name: train
  image: tensorflow/tensorflow:1.13.1-gpu-py3
  command: python train.py {parameters}
  parameters:
    - name: learning_rate
      description: Initial learning rate
      type: float
      default: 0.001
  inputs:
    - name: train-images
    - name: train-labels
    - name: test-images
    - name: test-labels
- pipeline:
  name: Training Pipeline
  connections:
    - [preprocess.output.train-images, train.input.train-images]
    - [preprocess.output.train-labels, train.input.train-labels]
    - [preprocess.output.test-images, train.input.test-images]
    - [preprocess.output.test-labels, train.input.test-labels]
```

Step

A single step in a graph represents a cloud machine running your code once.

Step definition contains:

- Docker image to be used
- Command(s) to be executed
- Input files
- Parameters

For each execution of a step, Valohai does the following:

- Spins up a cloud instance with a docker container
- Feeds the container input files and parameters
- Runs your code
- Grabs the output files, logs, and metadata
- Shuts down the docker container and the cloud instance
- Version-controls everything that just happened

Connection

A single connection in the graph represents data flow. The data flows from one step to another, and Valohai handles the data transfer for you. There are two types of data flows:

- **Files**
  - Input: Grab files from `/valohai/inputs`
  - Output: Put files into `/valohai/outputs`
- **Parameters**
  - Input: Parse `--myvar` from the command-line parameters
  - Output: Print JSON into logs `print(json.dumps({"myvar": 190}))`

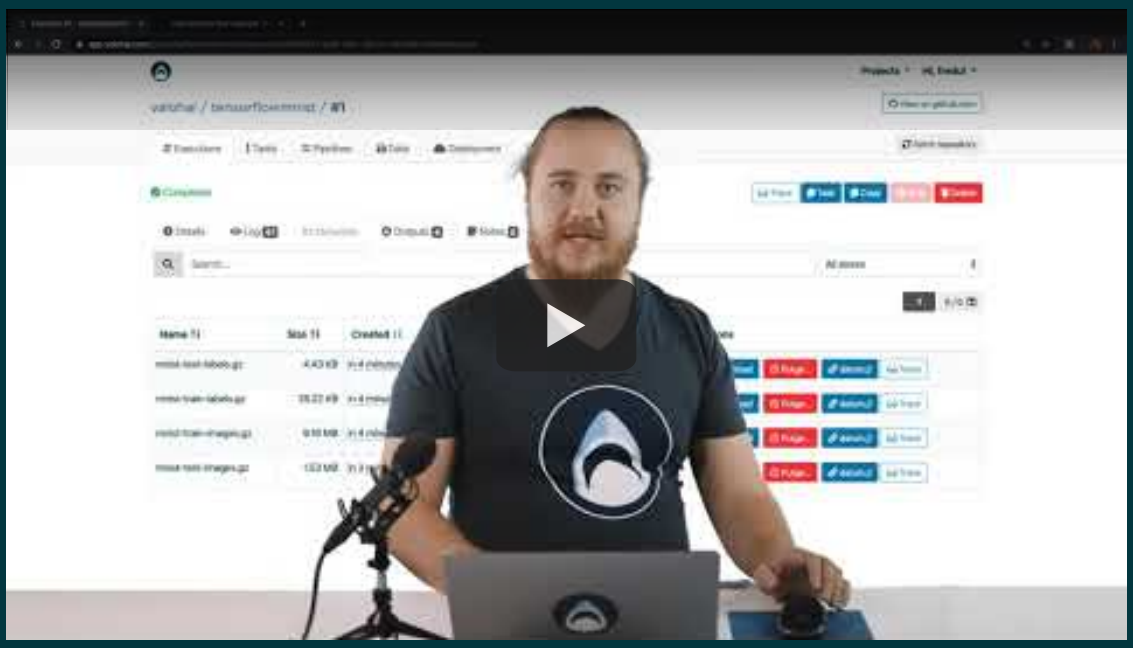
Last name*

Email*

Company name*

By submitting this form, you give consent for the authors (Valohai, SigOpt, and Tecton) to store the information provided and to contact you. You may unsubscribe at any time. For more information, see **Privacy Policy**.

Download the Free eBook!



Haven't heard of Valohai yet?

Valohai is the last MLOps platform you'll ever need. The platform allows you to build end-to-end ML pipelines that automate everything from data collection to deployment while tracking and storing everything.

Book a demo **Learn more**

Valohai

Product

Why Valohai?

MLOps

Features overview

All features

Get started

Pricing

Book a demo

Resources

Patch notes

Documentation

Open standards

Blog posts

Success stories

Nobody cares...