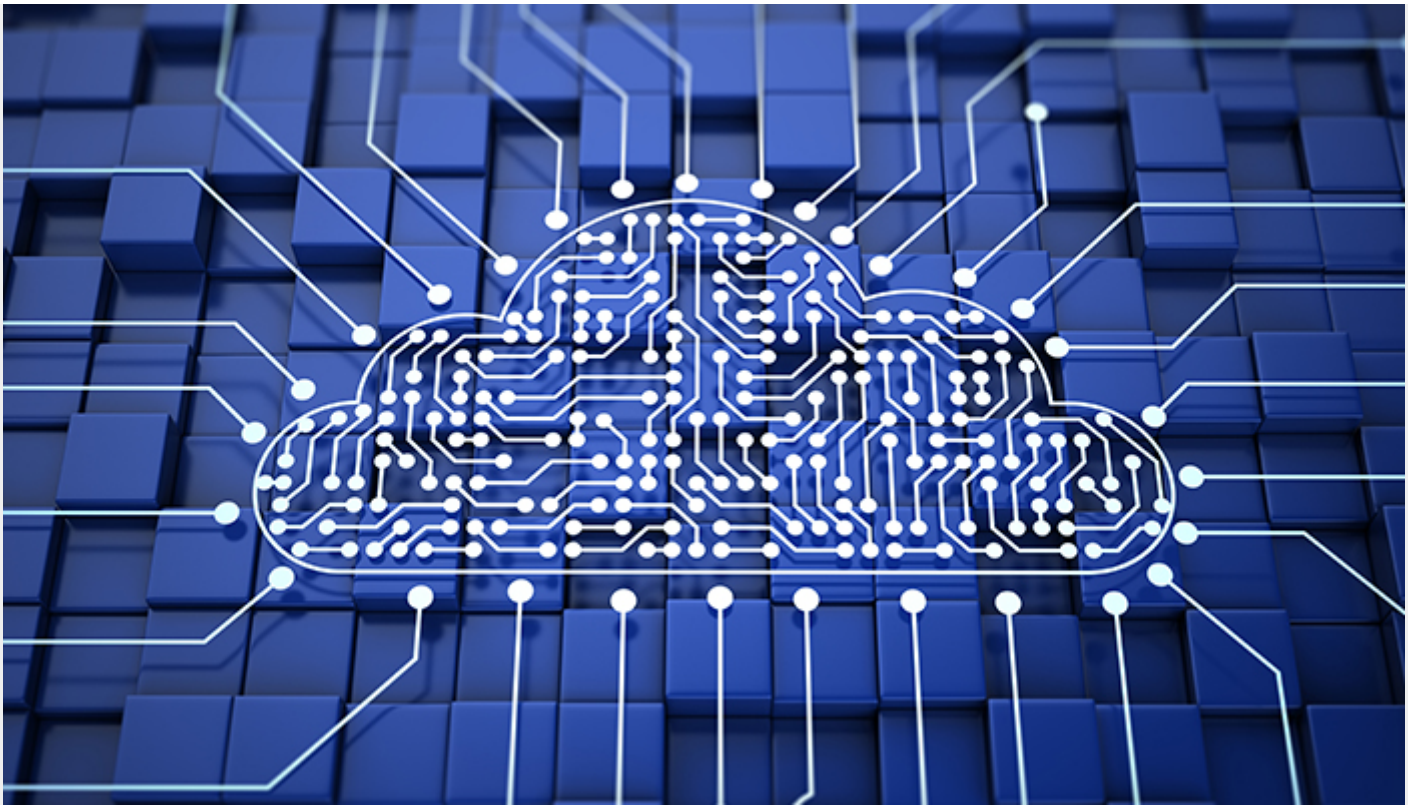
LET MACHINE LEARNING PIPELINES BENEFIT FROM AUTOMATION



A recent book published by O'Reilly, *Building Machine Learning Pipelines—Automating Machine Learning with TensorFlow*, provides practical, useful guidance on how to create machine learning (ML) pipelines, which is something many BMC customers that I talk with are heavily engaged in. At 367 pages, with step-by-step guidance on how to set up a Kubernetes cluster on Google Cloud and tips for running Kubeflow pipelines, this book is not a light read. We'll spare the technical details here, but do want to highlight the principles of effective ML development efforts highlighted by authors Hannes Hapke and Catherine Nelson.

- According to the authors, " to be more intelligent and make more use of AI, machine learning pipelines need to become more like CI/CD pipelines." Becoming more like modern CI/CD pipelines includes avoiding or at least minimizing custom coding or custom processes for each iteration of the pipeline. Organizations need to avoid doing things differently at different stages every time something new comes through the pipeline. As the authors note, "The connections between tasks are often brittle and cause the pipelines to fail."
- ML pipelines need to be automated, scalable and reproducible. This can't be accomplished if custom code is used extensively or processes are tweaked each time they run.
- Achieving these principles for automated, scalable ML pipeline development requires modern approaches. In their attempt to streamline development, enterprises have allowed, and maybe even encouraged, tooling to proliferate, in the belief that you can never have too much automation. For example, the 2021 State of ModelOps Report found that of 81 percent of

financial services companies were using multiple ML ModelOps solutions, and 42 percent had more than five.

The more tools that are used, the greater the challenge to integrate them. Scaling and automation become very difficult if tools can't work together, which is a reason why so many ML models never make it all the way to production. To address this problem, organizations are turning to orchestrators like Apache Airflow, Apache Beam, TensorFlow, and KubeFlow Pipelines. The book describes how these orchestrators can be highly effective for automating tasks in their target environment. However, their emergence has created a modern data management problem: How to integrate the multiple, domain-specific data and ML orchestrators that are now being used across the enterprise for its different development and data infrastructure environments?

Not having integration into a common environment greatly limits the ability to automate, troubleshoot, and scale pipelines. The integrated environment needs to provide visibility into dependencies, including those involving traditional business applications that may be generating input to or consuming output from these emerging ML applications.

It can be done. In the Control-M world, machine development is already like modern CI/CD because Control-M natively supports multiple cloud and ML environments and tools, including Apache Airflow.

ML orchestration needs to be comprehensive because it touches on the three core principles of becoming more like CI/CD pipelines—applying CI/CD principles, minimizing custom code, and pursuing automation and scalability. Let's look at these principles more closely.

Applying CI/CD Principles to ML

There are many domain-specific tools for almost every conceivable segment of technology and there is ample justification for almost all of them. This book makes clear that no one would consider using CI/CD tools for ML pipelines, even though at some level there appears to be a great deal of similarity. Multiple tools are necessary, so organizations need a way to see across all these domains and see the relationships among them.

Regardless of the tools used, or how many are used, organizations need a way to bring their workflows into a common environment where they can work together and be managed. That need reflects one of O'Reilly's fundamental principles for effective ML pipeline development. Luckily, there is an example that organizations can look to for guidance, as Hapke and Nelson note, "With machine learning pipelines becoming more streamlined in the coming months, we will see machine learning pipelines moving toward more complete CI/CD workflows. As data scientists and machine learning engineers, we can learn from software engineering workflows."

One thing many organizations have learned about software workflows is that Control-M enables automation and orchestration starting in development and continuing through each stage of the lifecycle. The solution is a well-proven, agnostic environment to support many development environments and business application workflows, and it extends those capabilities into the ML world. Control-M has native integration for Apache Airflow plus more than 90 other integrations. Now you can use Control-M to manage specialized ML code and models just like any other job. And you can do it in the way that works best for you, i.e., jobs-as-code, using JSON or Python, doing it all in Control-M, or using a combination through integration with your tool of choice.

Minimizing Custom Code

Custom code usually does a good job with tasks like executing a data import from one system to another. The trouble often comes when the environment needs to scale. Developing and maintaining multiple custom integrations becomes too time-consuming as ML programs grow. Automation is essential for scalability, and custom, non-standard code is an obstacle to automation. As the authors state: "Automated pipelines allow data scientists to develop new models, the fun part of their job. Ultimately, this will lead to higher job satisfaction and retention in a competitive job market."

Creating ML Models that are Automated, Scalable, and Reproducible

Through automation, organizations can remove the need for one-off, self-developed integrations that consume a lot of developer time, create a dependency on individual developers, and limit scalability. The beauty of Control-M is that it allows data science, development, and operations professionals to work with their familiar tools, then brings everything together through single interface.

Control-M not only automates workflow execution; it can also automate data ingestion, including file transfers. Embedding jobs-as-code makes workflows self-executing, which saves time in promoting to production. Control-M also has specific features that simplify working with big data environments. Users in multi-tool environments appreciate its core functionality that provides a clear, end-to-end view across very complex and diverse technology and applications landscapes.

Users can also act on this visibility, with Control-M's native functionality for pausing, killing, and restarting processes as needed, providing business-oriented and technical views of workflows, guidance for site standards, granular security, reporting, and workflow analytics for continuous improvement. These and other author actions can be automated based on events, conditions, and other triggers that you define.

To recap, the people that literally wrote the book on how to develop ML pipelines and get them into production more quickly and effectively cite the need to use specialized tools and automation while minimizing custom code. Historically, the complication has been that integrating specialty tools and automating their interactions has required custom code. Control-M solves that for ML and other environments, orchestrating across tools, clouds, and other components so pipeline development and execution can be automated. We'll publish a follow-up blog with more specifics on how Control-M enhances Apache Airflow.