DEVOPS 2.0

APPLYING MACHINE
LEARNING IN THE
CI/CD CHAIN

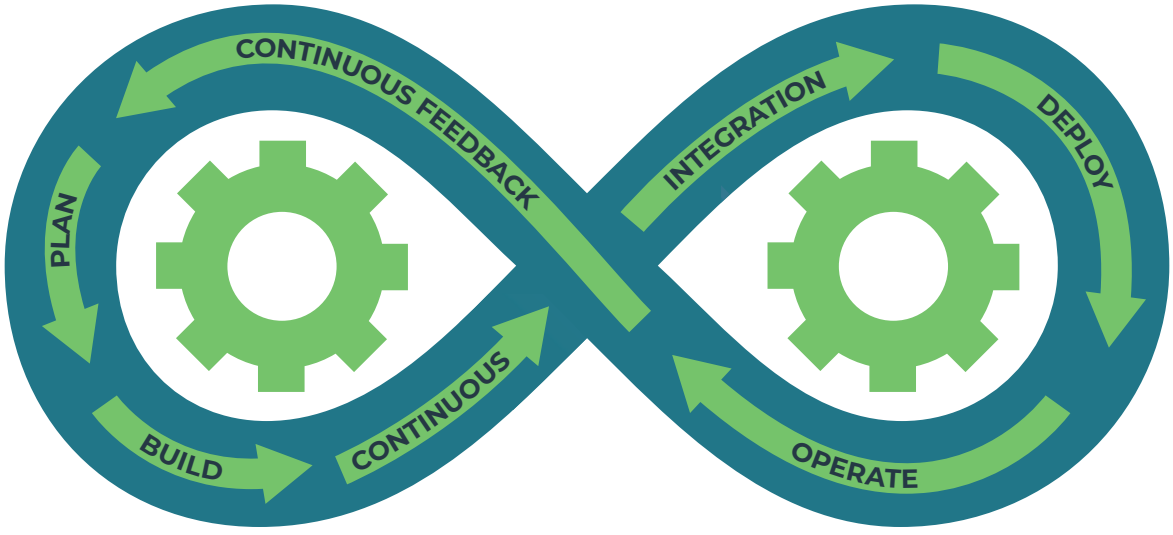
ActiveState®

DEVOPS 2.0

For a topic that's really just about math, statistics and algorithms, Machine Learning (ML) is generating a lot of interest with business people. But the ability to predict defects, failures and trends is what makes ML of particular interest to DevOps teams. ML-powered applications can make us all that much more efficient and empowered in our daily lives. And there's no role in software that can benefit from greater efficiency than DevOps.

Automation is a key design principle for DevOps teams. In order to speed up throughput in the software development lifecycle (SDLC), DevOps automates everything from builds to testing to monitoring. But most automated tasks are centered around a simple threshold (e.g. >50% of CPU usage) or pass/fail (e.g. # of critical warnings found) criteria. This means DevOps can end up chasing false positives or not catching subtleties in the code that trip alarms later, typically in production.

ML enables the automated assessment of test results for far more complex criteria, such as defining thresholds based on statistical significance rather than just presence/absence of specific criteria. All without slowing down the SDLC.



OFF-THE-SHELF VS DIY SOLUTIONS

While most next-gen DevOps tools have incorporated ML (such as Splunk, New Relic, etc), they essentially function as black boxes. And since the underlying algorithms are not exposed it's difficult to trust the results. While some tools do expose controls to a greater or lesser extent, it may be more beneficial to create your own solution rather than spending resources on tooling that may not fit your requirements or provide the level of visibility and trust you require.

However, adding the ML skill set to a DevOps team that is already expected to know a wide range of technologies can be a huge challenge. Because any time an organization adds new capabilities, they typically add new people, new teams, and a new set of processes that need to be followed. Change at this scale is never easy. As a result, the key to success for any ML initiative is the need for executive buy-in, as well as extensive communication in order to ensure all of the pieces work together.

Recall when you began your DevOps journey: the key to success was getting the development and operations folks to work together. You can expect a similar thing to happen in your ML journey when you add a data science capability to your organization. In this case, you need your data engineers to coordinate with your data modellers who need to understand how your DevOps people work in order to come together and deliver a great solution.

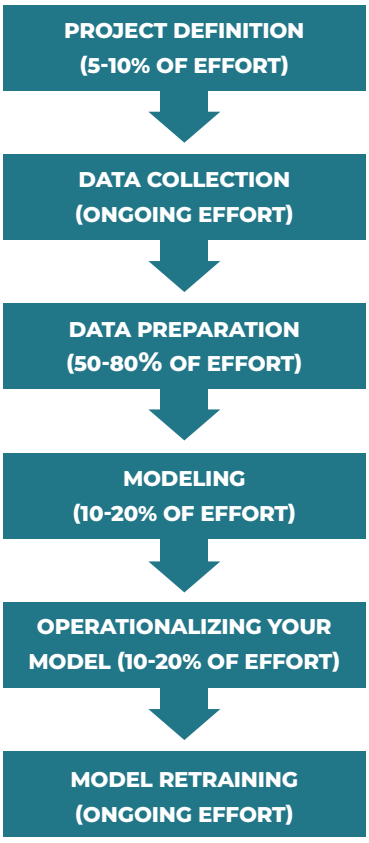
UNDERSTANDING THE ML PROJECT WORKFLOW

PROJECT DEFINITION (5-10% OF EFFORT)

Whether you build your own solution or purchase one, the first step in any ML initiative is figuring out where you can get the biggest bang for your buck. DevOps collects reams of data on a daily basis that can be used to streamline workflows and orchestration, monitor applications in production, and diagnose faults or other issues. Look for the most critical aspect of your workflow that is suffering from a lack of actionable information. Then create a project definition for it to ensure everyone is aligned toward solving the same goal.

Data Collection (ongoing effort)

For DevOps, data is collected through multiple channels, but text-based log files represent a significant source. If you're like most DevOps teams, you can quickly get overwhelmed by the shear volume. As a result, combing through logs looking for exceptions is typically all your team has time for. That's where ML comes in. ML models require a substantial dataset in order to train them up -- data that should (ideally) be labeled in a consistent format so your algorithms can derive correlations, identify trends and deliver insights not visible to the naked eye or traditional analytics solutions.



With labeled data such as time-series log data, supervised learning can be used to train a predictive model. This means you can focus on historical data and use it to predict future events like memory leaks, system failures or other disruptive anomalies. While unlabeled data can also be used in conjunction with ML, it requires a different methodology called unsupervised learning. Rather than predictability, the focus here is primarily on uncovering the hidden structures in the data through a method known as clustering.

DATA PREPARATION (50-80% OF EFFORT)

The next step is to prepare the data by cleansing (e.g. removing outliers that might skew the data, deleting or filling in missing values, etc.) and normalizing it (e.g. ensuring that all dates are formatted the same way). While you could just write a script to handle data preparation, there are various tools that can make the job easier such as Python's Pandas library, which is terrific for indexing, renaming and mapping, as well as doing joins and mergers. Be forewarned, however, that even with proper tooling data preparation can constitute as much as 80% of the time and effort in any ML initiative.

Modeling (10-20% of effort)

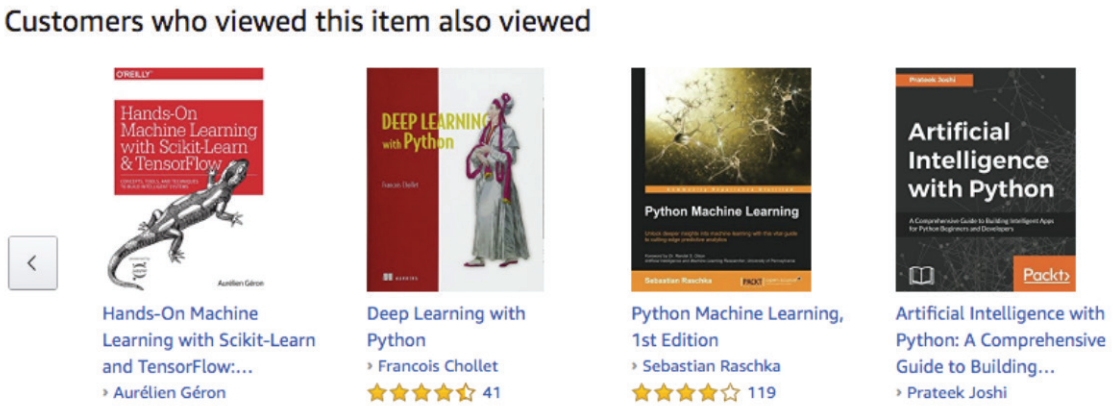
Once the data is prepared the modeling can begin. Creating a model involves "fitting" an algorithm to the training data, and then using the model to make predictions against the test data. Previously, data science teams would create and test their own algorithms. Today, you can leverage a number of well-known frameworks that provide access to proven algorithms out of the box. For example, Python's Sci-kit Learn library allows you to apply one or more of its standard algorithms against your training data until a model can be found that closely approximates your data points. The smaller the error between your actual data point and the algorithm's calculation, the better the model is likely to be. The process of finding this approximation is called "fitting."

OPERATIONALIZING YOUR MODEL (10-20% OF EFFORT)

Once the model is created, the last mile involves incorporating it into an application. How you incorporate your trained model into your application really depends on where you've stored the model. For example, it could be in a database, or a flat file that consists of protobufs or JSON. And how you expose your model for use by an application is really a function of the architecture you're implementing. Whether that's a monolithic architecture that builds the model right in, or whether it's a more agile architecture that would access the model via an API. There's no one best methodology or set of tools for the job. Rather, it depends on your organization's standards, developer strengths, and fitness to task.

Model Retraining (Ongoing effort)

But that's not the end of the project. DevOps data changes over time, which means your model will eventually become outdated. As a result, you'll need to plan on retraining your model. How often you do retraining depends on how often your data changes. For example, Amazon's Recommendation Engine, which recommends related products during a purchase is updated as many as 4 times per day. But Amazon is dealing with a massive amount of data that is highly time sensitive since Amazon wants to ensure they're not missing any trends in the marketplace. For DevOps, retraining may be more appropriately done on a weekly or monthly basis, depending on a number of factors including the frequency of events like releases, failures, process changes, etc.



HOW TO GET STARTED

Now that you have some basic knowledge about what's involved in applying ML to your DevOps data, the next big question is where do you start? While you can get started today with your existing personnel, you'll get further faster with dedicated ML staff. Previously, that meant hiring ML experts, but that's no longer the case. ML is a space that's changing rapidly. In fact, it's gotten to the point where you no longer need to be an expert in algorithms in order to, for example, work with image or voice recognition. Today, you can work with APIs that provide access to a pre-trained model that's ready to use. For example, Amazon has a voice and text recognition service called Lex, which is what drives Amazon's virtual assistant, Alexa. For DevOps, Lex might be useful for reading log files and extracting insights in real time – all of which can be done by a good technical team without the need for a data scientist.

Technology in general is becoming more democratized, but it's especially noticeable with ML. ML tools are getting better and more high-level at an extremely rapid pace. As a result, you no longer need to know the low-level details required to build up a neural network. Instead, you can just use a framework like TensorFlow, specify your inputs and the number of layers you want, and (as long as your data is prepped) start training a model right away. TensorFlow abstracts away much of the complexity of neural network creation, simplifying tasks like image recognition, speech, and natural language processing. Another framework, Scikit-Learn would be more appropriate for performing anomaly detection, such as when looking for root cause or detecting exceptions in "normal" processing.

BUILDING THE RIGHT ML FOUNDATION

It's this kind of rapid advancement in open source tools that's driving the majority of the innovation in ML, and, in turn, also driving the growing number of commercial solutions. Many of today's proprietary ML solutions are leveraging open source in order to create their offerings. These off-the-shelf solutions often provide algorithmic implementations out of the box, along with datasets and even pre-trained models.

However, the right foundation is neither solely open source nor commercial but a mix of the two. For example, it's unlikely you would want to build your own text recognition system when Amazon's Lex is readily available. However, you may want to use Python to support not only your data engineering and data science teams, but also the deployment of trained models into various applications. Python is widely considered the open source tool of choice for data science projects in general, and ML initiatives in particular. While R is still popular with statisticians, Python has the bulk of the ML and data engineering libraries, as well as strong Web and API frameworks.

For DevOps, Python's ML functionality is key to helping identify trends over time from time-series data, correlating information across different monitoring tools, or predicting failures. But Python's strong Web and API frameworks can also help overcome issues with the last mile, namely operationalizing models in production. For example, whether you need a full stack Web framework like Django or a lightweight, extensible one like Flask, both have excellent Web API libraries that can simplify the creation and exposure of your model.

Having consistent tooling across your data engineering, modeling and application development groups removes significant barriers to collaboration, enhancing your chances for success. Whether you choose to standardize on Python, R, Julia or another language, avoid version proliferation by adopting a single, standard distribution for all your teams in order to make support and maintenance manageable, and eliminate compatibility issues.

CONCLUSIONS

DevOps is in a bind. Automation is a key design principle for DevOps teams. But despite having automated so much of the release process, DevOps is still counted on to deliver ever more frequent releases. The problem lies in the fact that an automated Software Development Life Cycle (SDLC) generates volumes of distributed, dynamic, opaque and ephemeral data that are more than humans can comprehend. There are significant efficiencies to be gained if the data can be analyzed and acted on in a timely fashion.

Data collected through each release cycle, such as velocity, burn rate, and defects found, as well as data collected by CI/CD tools, such as successful integrations, number of integrations, time between integrations, and defects per integration, all have value if they can be properly correlated and evaluated. Recent advances in ML make it a viable – and valuable – tool for DevOps teams struggling with these kinds of signal-to-noise problems, enabling them to take a proactive approach to issues based on accurate predictions.

Given the tremendous benefits that an ML-driven DevOps team can bring to the release process, managers need to be ready to take the next steps to boost their team's ML capabilities through hiring, training and laying a proper foundation that starts with open source.

ACTIVESTATE – PYTHON BUILDS SINCE 1999

For more than 20 years, ActiveState has been providing commercially-backed, secure, stable and comprehensive open source language distributions that have become renowned for quality, and are now the de-facto standards for millions of developers around the world. 100% compatible with community open source code, ActiveState's open source language distributions can be freely downloaded, but crucially also offer guaranteed support SLAs and regular maintenance updates, as well.

ActiveState®

website: www.activestate.com
Toll-free in NA: 1.866.631.4581
email: solutions@activestate.com