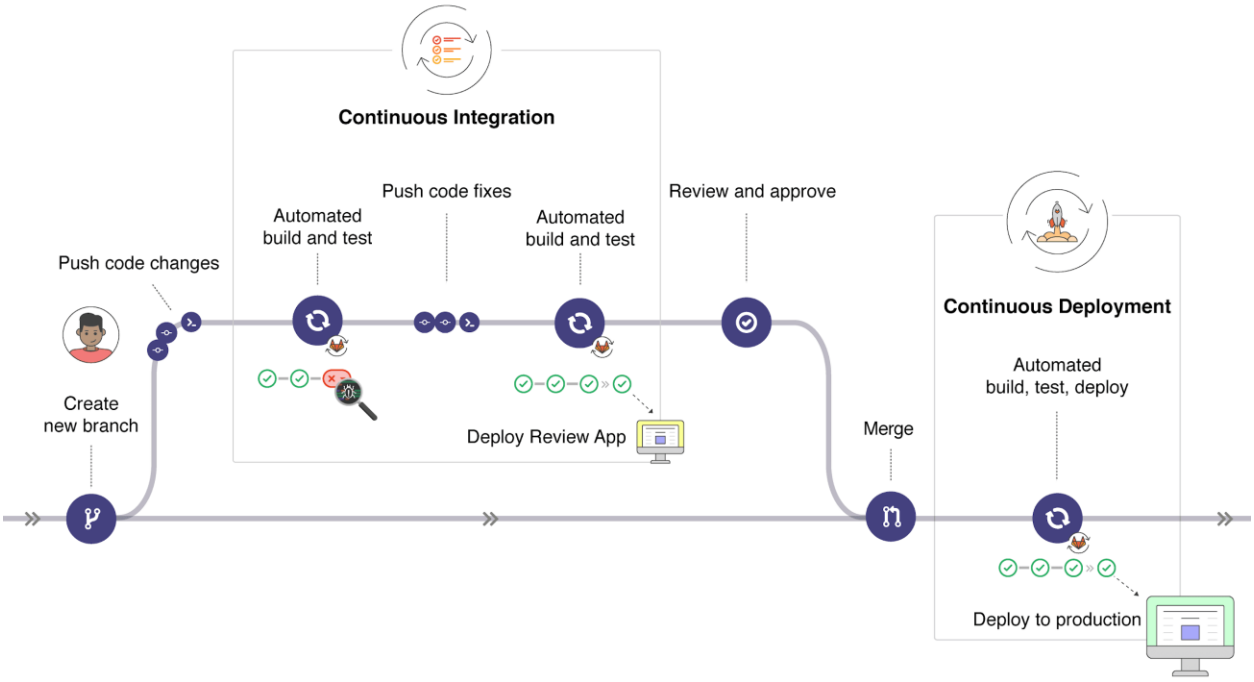
Final Project: ML Ops

Traditional Dev Ops

Traditional devops broadly refers to the set of practices and tools that developers use to build, test, and deploy code in production. Two core components of traditional devops are continuous integration (“CI”) and continuous deployment (“CD”). CI refers to the practice of building, testing, and merging code changes to a shared repository. CD then refers to that set of changes being deployed to end users in an automated fashion. The following diagram describes the typical CI/CD pipeline at a high level:



There are several existing solutions that help with CI/CD pipeline building and management, including Jenkins, Bamboo, CircleCI, and GitLab CI/CD.

ML Ops

The development of machine learning models to use in production applications shares some similarities with traditional devops, but a few key differences make existing tooling insufficient to fully meet the needs for this use case.

Data as key ingredient

Starting with the input to a CI/CD pipeline - in traditional development, code is the main input. In development of machine learning models, though, code *and data* are the key ingredients. The introduction of data as an input presents numerous challenges, as existing version control systems like git, and by extension the CI/CD solutions that rely on them, generally do not have

the capability to version control and verify data integrity as they do code. For example, the codebase for a traditional application is kept under version control and hosted in a repository platform like GitHub or GitLab. When a change to the existing code is made and the modified version is pushed to the hosted repository (typically in a pull request or merge request), it is obvious what has changed in the underlying source code. This visibility helps with quality control as developers can identify the source of any behavior change in the application (for example, if any of the specified tests do not pass) and revert to previous versions of the application if necessary. In a machine learning system, code changes are not the only source of potential behavior change - as changes in the underlying data used to train and validate the models may also change over time, and the weights in a machine learning model may also change as the model re-trains. When it comes to testing, quality assurance, and debugging machine learning models, having a complete record of these data and changes to them over time is a crucial component of practicing ML Ops at scale. Connected to this, data validation upon entry into the CI/CD pipeline is also necessary - this prevents data errors from affecting the performance of the production system and helps machine learning practitioners develop using the same environment and tooling as that used by the production system.

Training, validation, and performance monitoring

In a traditional application, the testing portion of a CI/CD pipeline consists of running integration and unit tests against the codebase to make sure any changes have not resulted in unexpected results or behaviors. These tests are typically defined by engineers as they build the system and serve as guardrails for its future evolution. In a machine learning system, these sorts of tests will typically also be performed, but the outputs of the machine learning model also need to be validated and checked for quality. This presents a conundrum, as if we knew a priori what the correct outputs of the system would be for all cases, we would not need to use machine learning in the first place and could hard code the outputs in the application. In a supervised learning context, a machine learning practitioner will typically train a model on one dataset, using another for validation and a final hold-out set to test the model's performance. These training, validation, and test datasets will have "labels" or the output variable that the system is attempting to classify or predict. From the validation and test datasets, various performance metrics can be calculated to give an indication of the model's effectiveness. In the CI / CD context, however, this workflow presents challenges. First, a product that depends on a machine learning component ideally does not train and validate the model only intermittently and manually, but continuously or at least regularly while the model is deployed in order to make use of new data and prevent the model's predictions from becoming stale. Therefore, the training and validation processes need to be integrated into the CI / CD pipeline and their results automatically assessed to determine whether new code makes it to the deployment phase. Maintaining fresh data in the training and validation steps is particularly crucial to avoid overfitting and deploying models that do not perform well when faced with live, current data.

To keep track of quality when a model is deployed, performance monitoring also needs to be baked into the system. In a traditional software deployment, the application monitors its performance so that engineers are alerted if a component begins to degrade or fails. The same should be true in the MLOps context, but implementing this well is more challenging, as both the

operational aspects of the system (e.g. prediction latency) and model performance need to be monitored. The number of metrics and the situations to which they apply are also highly variable depending on the particular use case of the ML system. This requires per-model configuration, rather than a global performance monitoring system.

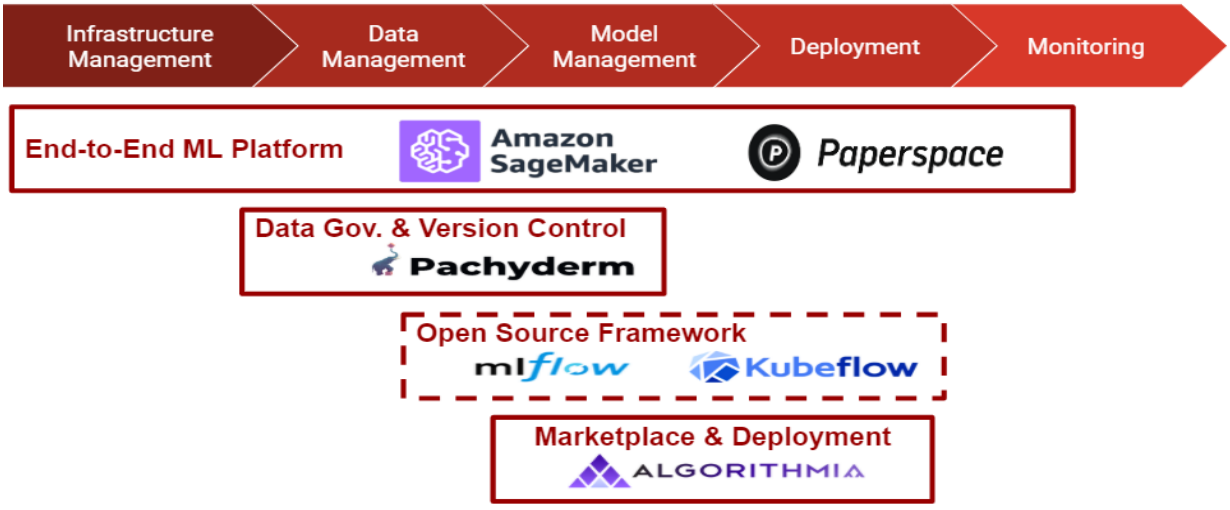
Finally, models, like the code that produces them, need to be version controlled to maintain audibility and be able to revert to previous versions of the system quickly if something fails.

Infrastructure management

To achieve the capabilities listed above, a CI/CD system also needs to be able to manage the computing resources necessary to perform the tasks, from data extraction and validation to model training. While such resource management is not unique to ML Ops (at a high level), the introduction of machine learning workloads make these processes more complex. For example, training models on GPUs rather than CPUs is generally much more performant, but such processors need not be used throughout the pipeline. The infrastructure requirements will also vary significantly between the development and production environments, as training can typically be done in batch processes and resources do not, generally, have to be perfectly elastic or responsive to the load they are handling, as they can be scaled up or down by the data scientist or engineer. In production, however, the requirements may be quite different, as prediction latency may be more important and the infrastructure should respond to end-user traffic. Additionally, monitoring infrastructure usage across a company becomes important for being able to properly allocate the cost associated with using ML models back to the business in a transparent way.

Existing ML Ops Tools

Along the ML workflow, we have identified several key ML Ops tool categories summarized in the chart below.



End-to-end ML platform - AWS Sagemaker, Paperspace

While Sagemaker and Paperspace are the more popular products in the market, there are a few of its kind. Along with AWS Sagemaker, Google Cloud AI Platform and Microsoft Azure ML Studio are the platform plays of the big three in the cloud computing space. Smaller players such as Paperspace and Floydhub tend to offer products with lower technical requirements with relatively limited scalability and flexibility. We compare AWS Sagemaker with Paperspace for a sense of how a big player and small player might focus on different aspects of product offerings.

	AWS Sagemaker	**Paperspace**
Ease of Setup	More complicated - appropriate shell scripts need to be run to configure EBS volume, set up dedicated IPs and install the required packages, software tools and deep learning libraries.	Basic instances can be set up within minutes
Infrastructure	Integrate well with the rest of AWS properties but require knowledge in permissions (e.g. IAM) and allocation of compute resources (e.g. EC2)	Gradient° as the core platform with powerful hashing is one of the key design patterns
Performance	Papersource has better performance under the same environment. (*Data provided by Rupak Thakur: "Sagemaker AWS p2.xlarge and Paperspace GPU+ have almost equivalent performance with AWS just inching ahead. If we use the Pascal versions on Paperspace, which are still cheaper than AWS, the model performance is expected to be 3x as fast as AWS."*)	
Cost	Start at $0.9/hr with 30GB free EBS volume under the Free Tier program. $13/month per 100GB SSD volume+ elastic IP	Serverless ML (on-demand billing); pay for instances run on-prem, on AWS, or on GCP

Open source framework - MLflow, Kubeflow

Open source frameworks like MLflow and Kubeflow compete to become the standard of the open-source landscape.

	MLflow	**Kubeflow**
Structure	- A single python package - A lighter tool with a laser-sharp focus on tracking and archiving	A combination of open-source libraries that depends on a Kubernetes cluster to provide a computing environment

Ecosystem	- Backed by Databricks	- Backed by Google and integrated with Google Cloud Platform
Collaboration	- Easy collaboration in remote or local environments - Simple logging process for exploratory data analysis (EDA) and development work - “Project” format offers an enhanced ability for users to share and try other's projects with minimum hassle	- Knowledge of Kubernetes is required - Direct control over infrastructure used for EDA, development, and production environment
Data management	Require external support for feature engineering, data pipeline development, and pipeline orchestration	- High scalability and hyperparameter tuning
Model management	- Simplified tracking - “Model Registry” enhances governance and core proposition of model management	-High scalability and hyperparameter tuning - Strong scheduling and orchestration capabilities
Hosting	- Basic hosting components but easy to push to cloud vendor API endpoints	- Wide range of hosting components but require more development effort

Marketplace / deployment - Algorithmia

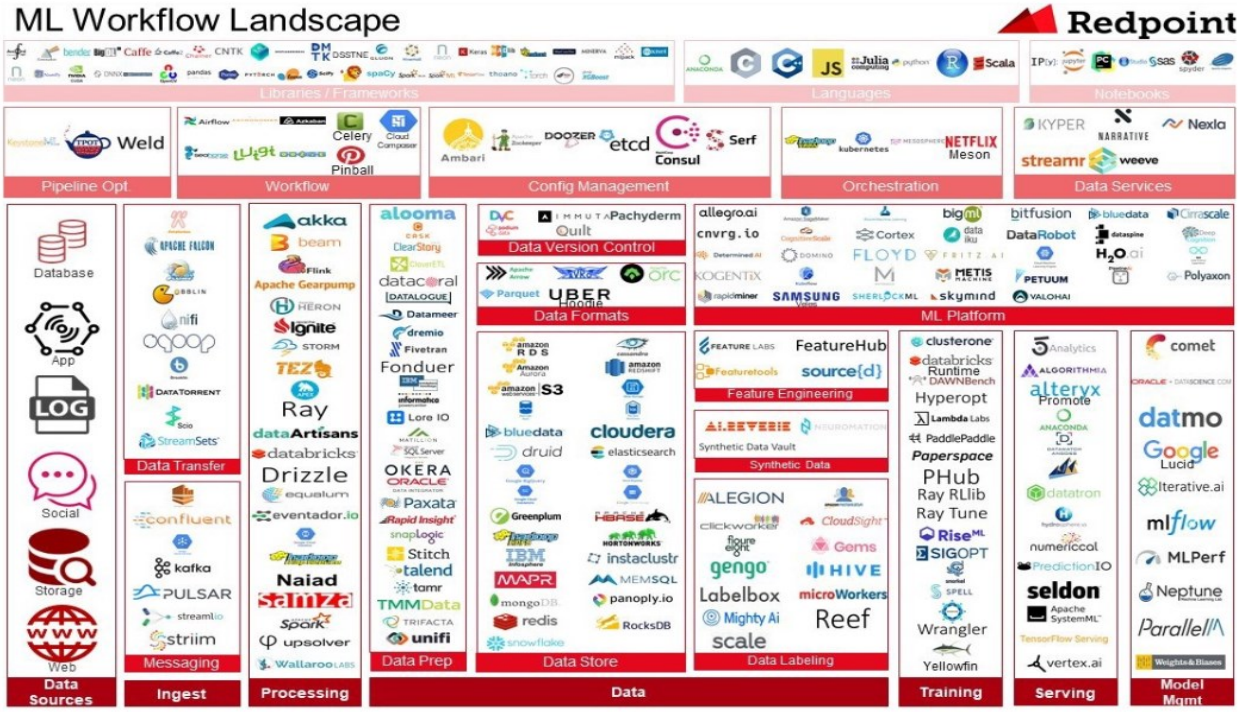
If one believes the prediction that ML algorithms will become commodities, an “app store”-like marketplace for algorithms as Algorithmia has the potential to be the Petri dish for that vision. Launched in 2015, Algorithmia started off with building its marketplace. It had accumulated over 3500 ML algorithms in the marketplace with almost 50,000 developers using the system within 2 years. The latest focus of Algorithmia is helping companies accelerate the process of getting their ML models into production. Essentially, the service helps deployment at scale. Algorithmia’s serverless microservice architecture involves a mix of Git, Kubernetes, and Docker. It can work with most training platforms, languages, and frameworks for deployment through Algorithmia Management API.

Google led Algorithmia’s $10.5M Series A and Norwest Venture Partners led the $25M Series B in May 2019.

Point solutions such as Pachyderm, Spark ML, etc.

In 2018, Redpoint VC put together the ML workflow landscape including about 280 point solutions along the ML value chain. We can see that existing solutions have a heavy focus on dealing with data from all aspects. For example, **Pachyderm** creates scalable and manageable ML pipelines on Kubernetes. It audits all data sources entering the data pipeline at every step

and manages version control. On the processing side, **Apache Spark ML** provides a platform for training full machine learning pipelines on these versioned/tracked datasets, including feature generation/extraction and predictive models.



Areas of Opportunity

While the area of ML Ops is nascent and there exist a slew of unresolved questions and competing solutions, some areas have attracted more attention than others. Data governance and versioning are being worked on by the likes of Pachyderm and others, such as Quilt. Managed development environments and infrastructure provision are a competitive space, with Paperspace, FloydHub, and other products like Google's Colab being some of the early leaders. The area in which we see the biggest scope of opportunity as machine learning workflows eventually approach traditional dev ops workflows is the step at the end of the process, namely performance monitoring and reporting. Companies like Data Dog have had great success developing monitoring solutions for traditional apps as infrastructure has consolidated to cloud providers, and we believe demand for a ML-specific suite of performance monitoring tooling will emerge as the CI/CD processes around machine learning mature. For example, a few areas that are crucial to watch in order to diagnose the health of a ML system include the following:

- **Traditional metrics** like AUC, accuracy, precision, recall, etc. for supervised learning systems. These metrics need labeled data in order to calculate them, so having a dynamically updating test dataset to use in production performance monitoring is necessary

- **Data validity**. As the product and user behavior change, the input data to a machine learning system may become different than what the model was trained on. Monitoring such changes may give early warning signs to degradation of model performance
- **Bias, differential impact, and model transparency**. In addition to model-level metrics like those mentioned above, it is also necessary to track how a given ML system is impacting different subsets of users. Especially in regulated industries, having a verified and trusted methodology for reporting, auditing, and alerting on potential disparate impact across groups of users will only become more important.
- **Resource consumption**. While there are existing solutions to track resource use, having visibility into cost at the model or even sub-model level may also become more important as the industry matures. For example, in a deployed deep learning system that runs an expensive training job every night, being able to see which layers of the neural network are most costly may provide useful insight into where to focus incremental data science resources
- **Numerical stability**. In some cases, machine learning models can exhibit aberrant behavior due to computational issues (e.g. weights in a neural network becoming very small or very large). If such issues arise in a production setting, detecting such failures is critical to maintaining performance.
- **Security**. Machine learning systems present unique security challenges. Classification models whose outputs are exposed to end users (e.g., a loan application decision) can be reverse engineered by observing the inputs and the outputs. Models that learn in an online fashion may be targeted with bad data as inputs. Thus, monitoring solutions that account the diversity of possible attacks against ML applications will be necessary.

There are also several quality of life features of a performance monitoring system that could make machine learning engineers more productive, such as:

- **Metrics repository**. Some typical metrics are easy to calculate and widely available via ML frameworks, but many times businesses may have more unique performance metrics that are also relevant indicators of a deployed ML system's performance, such as conversion rates or other types of particular user behavior. Being able to incorporate user-defined metrics such as these into the monitoring component of a CI/CD system would be valuable.
- **Alerting**. Existing solutions allow automated notifications to operations teams when, for example, a site is down or performance has slowed significantly. Alerts for ML-based systems, incorporating some of the factors listed above, would be similarly useful.
- **Visualization**. While practitioners of deep learning are certainly themselves able to create the visualizations necessary for monitoring, having an easy-to-use dashboarding solution that is tailored to the use case would save time, encourage detailed monitoring, and increase transparency in machine learning systems throughout the organization.

Conclusion

In summary, we see deployment monitoring as the biggest area of opportunity for a new ML Ops product. Fundamental issues in ML Ops, like data and model versioning, are being tackled by several players currently, making entry by a new player an uphill battle. This is doubly true for managed services for model development and deployment, as it will be difficult to compete with the cloud providers that can integrate all the way up the stack. This leaves deployment monitoring as an area for opportunity, as there are not any leading products currently (to our knowledge) and monitoring is certainly a use case where many companies would rather buy a product vs. build their own as it is not core to their product (see the success of Datadog, for example). As with existing dev ops tools, we see the ML Ops space ultimately being composed of low-level open source tools, integrated solutions from the cloud providers, and high-level point solutions in areas that are important but not core to most companies' product development. We see this latter area as the biggest white space generally, with deployment monitoring specifically as a concrete opportunity in the near term.

References

- https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning
- https://www.youtube.com/watch?v=VHojfWuqAlg&feature=youtu.be
- MLflow vs. Kubeflow https://medium.com/weareservian/the-cheesy-analogy-of-mlflow-and-kubeflow-715a45580fbe
- MLflow https://databricks.com/blog/2018/06/05/introducing-mlflow-an-open-source-machine-learning-platform.html
- Algorithmia https://www.datanami.com/2019/04/09/algorithmia-laser-focused-on-ml-deployment-and-management/
- ML platform https://medium.com/@rupak.thakur/aws-vs-paperspace-vs-floydhub-choosing-your-cloud-gpu-partner-350150606b39
- ML platform https://www.zdnet.com/article/guide-to-enterprise-ai-and-machine-learning-companies-and-applications/
- ML Dev & Ops breakdown https://www.forbes.com/sites/cognitiveworld/2020/03/08/the-emergence-of-ml-ops/#3da2fea84698
- Pachyderm https://blog.mi.hdm-stuttgart.de/index.php/2019/02/26/reproducibility-in-ml/
- Point solutions https://medium.com/memory-leak/introducing-redpoints-ml-workflow-landscape-312ca3c91b2f