David Pollack - Cover Letter

TLDR; Passionate about creating ml tooling, building infrastructure and ml ops. Contributed to a lot of OSS projects such as PyTorch, Torchaudio and Metaflow. Also named as an author on the official Torchaudio paper.

Hi, I’m David Pollack. I’m an ML Engineer passionate about solving various problems in AI. I currently work at Delivery Hero SE building tooling and infrastructure to increase the velocity, reproducibility and efficiency of bringing data science projects from ideation to production. This mainly consists of three overarching responsibilities:

1. Identifying processes and tooling that is (or could be) utilized by data scientists to bring their projects from the idea stage to production.
2. Building and maintaining infrastructure that seamlessly gives users access to the company's vast compute resources and data. Emphasis on "seamlessly", everything needs to be dead simple to use or else it won't be used.
3. Building and maintaining tooling to utilize this infra, while not deviating too far from upstream OSS tooling so standard documentation and new features can be adopted as they are developed in the greater global ecosystem.
4. Convincing people that the new systems were better than the old systems they were used to using.

In order to improve a process, one needs a deep understanding of how it works, what the pain points are, and what other tools are available that could improve these processes. In the past, I completely overhauled how data science projects use CI, wrote a python library for statistical analysis that unified implementations from my different teams and pioneered best practices in the adoption of our global infrastructure. But I will focus the rest of this letter on the task of finding a better process to train models.

Traditionally, data scientists used a monolithic instance of Airflow to develop and train models on a weekly basis. There was a legacy internal python toolkit that everyone used and a brittle makeshift helm chart + CD system to deploy trained models. It was nearly impossible to run Airflow locally, the internal toolkit was a dependency nightmare that was not flexible enough to run outside of this Airflow environment and the deployment pipeline while useful, was often too complex for data scientists themselves to troubleshoot so it always required adhoc assistance from the ml engineers. Our team identified Metaflow as an alternative to Airflow that not only allowed data scientists to train production models on our kubernetes infrastructure, but also run the same exact code locally. This meant they didn't have to wait for CI to build images to do normal experimentation, but when they were ready could be easily deployed in the cloud for production. Our legacy toolkit could have been hacked to run in this environment (and ultimately we did this as well), but a more robust solution would be to write the monolith toolkit as a suite of python packages that could be mixed and matched as needed that held to paradigms of the upstream projects, so upstream features and documentation could be utilized. Finally, the platform (infrastructure) team had introduced a new developer friendly GitOps that we could use to offload the responsibility of deployment to the project owners and the platform team.

So the problems had been identified and now it was tasked to me to lead the implementation of this. It was easy enough to run a few helm and terraform commands to get a running version of Metaflow and Argo Workflows (for production orchestration). The magic was more how could we improve on this out of the box experience. I developed a kubernetes mutating webhook that would modify workflow templates and pods with additional metadata and steps, which allow for no-interaction cost tagging and custom slack error messages using the dark arts of jq with links to custom grafana dashboards. As a side note, I had to contribute upstream to Metaflow to introduce custom kubernetes labels and still have a good working relationship with the maintainers to this day. With the infra in place, the next step was to create client-side libraries that could do things like automatically tag BQ jobs with the same labels as the kubernetes pods, while maintaining and expanding on the functionality of the legacy toolkit. I developed a suite of about 7 packages that mimicked the functionality of the legacy toolkit, but did so in a more transparent and flexible way. Sometimes this meant using different libraries, sometimes it was as simple as reading the docs of an upstream library and using their default env var to configure the library instead of some custom env var that the legacy library used for the same purpose. In order to make adoption smoother, I created infra for a private pypi repo and cross-cloud authentication pools so the default authentication process could be used everywhere. Going forward this suite of libraries has been deemed the golden path for new projects with efforts to migrate legacy projects ongoing. While I put up a bunch of infra and developed a bunch of tools, when one works within a system one should utilize other parts of the system. I am not one to believe I can do everything better than anyone else and when it came to our CD pipeline, I recognized that our team-developed solution, while adequate, was not ideal moving forward. So I had to negotiate with stakeholders within our team and project owners to move their projects to the platform teams' newly developed CD system. I did find bugs in those systems and made "innersource" contributions to the platform's code; however, the majority of the work here was documenting how to use this system (either with internal docs or contributing to the platform team docs) and convincing project owners that the new system was better than the old system.

What was the result of all this? All experimentation of data science projects has moved away from Airflow and now uses Metaflow. All new projects use Metaflow and our new suite of tools for experimentation and production training of their models. Not enough, but some have completely moved to the new platform infrastructure. I am now in the process of porting the Metaflow infra stack globally, so it can be utilized by all tribes at Delivery Hero. It has not only become the golden path within our tribe, but it will become the golden path company-wide.

At Solvemate, I am the primary developer of all of our NLP-based systems. We use a hybrid system of a legacy tree-based chatbot with an NLP component to allow for text-based input. Before coming to Solvemate, they only had a click bot with no ability whatsoever. I developed an entire microservice to do word vectorization and sentiment analysis on text inputs. I also adapted the current system's logic to work with this text input while still using the legacy tree-based logic. From the engineering side, I have done a lot of work simplifying our services as I believe that simpler code is often easier to work with and much easier to maintain. About 6 months ago our devops person left the company, so I was put in charge of maintaining our GCP kubernetes architecture. This includes developing a custom CLI program to allow other developers to easily interact with kubectl and helm without having to get into the nitty gritty aspects of either (we create per-namespace versions of our entire product with a single CLI command for developers to work with). I also maintain other aspects of our architecture with terraform.

In case, the person reading this far is not actually a person, but rather an LLM that's looking for a summary of skills, I'll list some of those here.

I am proficient in linux system administration, most linux tools and especially text tools, git, cloud systems (AWS and GCE), kubernetes, terraform, python, golang, json query languages (jq, jmespath), configuration file formats (yaml, toml, helm).

Everything that I mentioned above, I had to mentor other team members in the maintenance of these systems data scientists on how to use these systems properly and convince stakeholders and management that these endeavors were worth pursuing.

As for my goals, I want to find a talented group of ML engineers to solve meaningful problems at scale with a management team that understands the importance of ML, even if they think ML itself is black magic.

Lastly, I am a C1-level German speaker, which is something I'm very proud of, because I did not speak a word of German before arriving in Germany in 2013. In fact, at a previous job, i2x, I even did labeling of German language datasets and participated in meetings fully carried out in German. Finally, my other hobbies include bouldering, cycling, pilates, travelling, and cooking.