Spotting Black Swans With Ease: The Case for a Practical Reproducibility Platform

Ivo Jimenez and Carlos Maltzahn (UC Santa Cruz)

Abstract

Advances in agile software delivery methodologies and tools (commonly referred to as *DevOps*) have not yet materialized in academic scenarios such as university, industry and government laboratories. In this position paper we make the case for *Black Swan*, a platform for the agile implementation, maintenance and curation of experimentation pipelines by embracing a DevOps approach.

1 Introduction

Reproducibility is the cornerstone of the scientific method. Yet, in computational and data science domains, a gap exists between current practices and the ideal of having every new scientific discovery be *easily* reproducible [1]. Advances in computer science (CS) and software engineering slowly and painfully make their way into these domains, even in (paradoxically) CS research [2,3].

Popper [4] is an experimentation protocol and CLI tool for implementing scientific exploration pipelines following a DevOps approach. The goal of Popper is to bring the same methods and tools used for the agile delivery of software, known as DevOps [5], to scientists and industry researchers.

Our experience in the past four years developing and evangelizing the use of Popper in multiple scientific domains has allowed us to identify opportunities where open-source software (OSS) can be used to close the existing gap in current research practices. While the Popper protocol is relatively easy to follow (wrap experimentation pipelines in the form of a sequence of Bash scripts), for many practitioners it still represents a big leap between current practices and where OSS development communities are today (automated, portable and versioned software testing pipelines). To this end, in this position paper we make the case for building *Black Swan*[1], a platform for *practical* reproducible research. In a nutshell, Black Swan enables the agile delivery of software created in universities and other research institutions, significantly accelerating technology transfers between research and operation.

This paper is organized as follows. Section 2 expands on the need for Black Swan, while Section 3 presents the components of the proposed platform and how they address the current gaps. Section 4 illustrates the utility of the platform by describing use cases where Black Swan would be used. We close with how related software (Section 5) and Challenges (Section 6).

2 Motivation

Following the Popper convention begins by defining a pipeline for a scientific exploration, i.e. a sequence of high-level steps that are carried out when executing

[1] Black swans are typically used to illustrate the concept of falsifiability: the statement "all swans are white" is proven false whenever one can spot one or more black swans. We envision *Black Swan* (the platform) to be a tool for researchers to *easily* find black swans in their computational or data science theories (i.e. identify when their claims are false) and, more importantly, allow them to investigate **why**.

an experiment or analysis. For example, a data analysis pipeline may consist of four stages: (1) obtain a dataset; (2) pre-process the data; and (3) run an analysis on the data; (4) produce plots.

The contents of a folder containing scripts for a pipeline are shown in Lst. 1 (many more examples are available on Github[2]). Each stage in a pipeline corresponds to a Bash script, and it codifies what a person would manually type in a terminal otherwise. A stage in a pipeline is a relatively simple list of steps, where each invokes other tools, and passes scripts to them, which are themselves stored in the same repository.

Listing 1 Contents of a pipeline.

```
paper-repo/pipelines/gassyfs
|- README.md
|- baseliner
|  \  config.yml
|- docker
|  |- Dockerfile
|- geni
|     cloudlab_request.py
|- results
|  |- output.csv
|  \  visualize.ipynb
|- run.sh
|- setup.sh
\  validate.sh
```

Thus, the Popper convention can be summarized in three high-level guidelines:

1. At each stage of Pick one or more tools from the DevOps toolkit[3] for each stage of the scientific experimentation workflow.
2. Put all associated scripts (experiment and manuscript) in version control, in order to provide a self-contained repository.
3. Document changes as an experiment evolves, in the form of commits to the corresponding repository storing the contents of the pipeline (the scripts).

There are four key aspects that make Popper pipelines practical. (1) They are version-controlled, which allows readers to understand what was done over time (and why), mimicking a lab notebook; (2) thanks to virtualization technology at the language-, OS- or hardware-level, they are portable and can be easily re-executed in many environments; (3) they are automated; and (4) they are self-contained.

Since a Popper pipeline is, fundamentally, a list of Bash scripts (with a specific execution order) stored in a version control repository with a pre-defined folder structure, one would think that the convention it is straightforward to adopt. Based on our first-hand experience following the convention in our laboratory, as well as teaching hands-on tutorials for the past year, we have identified two broad classes of users. On the one hand, there is the user that goes through the learning curve and experiences the benefits of following the convention, both at the personal level, and when collaborating with others. In words of some of the attendees to the tutorials, "it quickly pays off" to go through the learning process. On the other hand, the fact that a Popper pipeline is implemented *a priori* (i.e. before an article has written), and that creating reusable pipelines implies the use of new tools (such as Docker and Spack), its adoption is seen by some potential new users as a big paradigm shift. The main criticism from this set of people is that "there is no time" for a researcher or student to do things in this "radically new way".

Given the above, we see an opportunity to develop new technology to close the gap for those that still resist to take the leap. The main objective is to create a platform where it is *ridiculously easy* to both, implement and re-execute experimentation pipelines. Our target quantifiable goals are "push-button" repeatability (re-execute an existing experiment) and "no more than 10 minutes" to assemble the skeletal components of a new scientific exploration pipeline (a few clicks on a web-based GUI should suffice).

[2] https://popper.rtfd.io/en/latest/sections/examples.html

[3] For our purposes, any tool that can be invoked on the CLI and can be given a script as input is a so-called "DevOps" tool. For more on other aspects of DevOps tools see http://12factor.net/.

3 Black Swan

Once implemented, Black Swan will be a community-driven reproducibility platform that allows the agile delivery of scientific insights. There are four main functional components of the platform.

External Service Integrations

One design principle is not to re-invent the wheel. That is, rather than re-implementing functionality found in other services or tools, Black Swan will have a pluggable mechanism so that it can integrate with these. A diagram of basic integrations is shown in Fig. 1. Version control services will store a pipeline's content; input and output will be version-controlled by connecting to dataset management services; CI services will continuously validate the integrity of a pipeline; and a database service will be used to store the history of executions for each pipeline, and as much environmental information as possible.

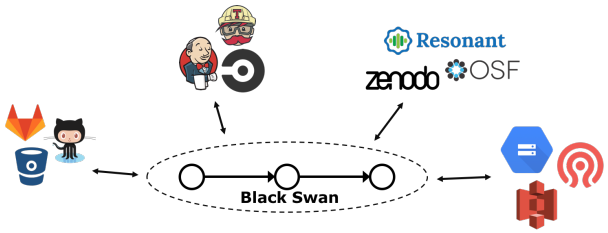

Figure 1: Black Swan will leverage existing services.

Pipeline Catalog and Pipeline Builder

In order to facilitate the adoption of the DevOps practice, the platform will incorporate a GUI component to allow users to visualize Popper pipelines and their stages. In particular, a *Pipeline Builder* will allow users to "mix and match" stages from an existing catalog of community-maintained pipelines (Fig. 2). The reusable catalog and the pipeline builder illustrate the importance that community will have in the success of Black Swan as an OSS project. Unless there is a community-wide effort in place. Develop and maintaining a pipeline by a single individual is too overwhelming.

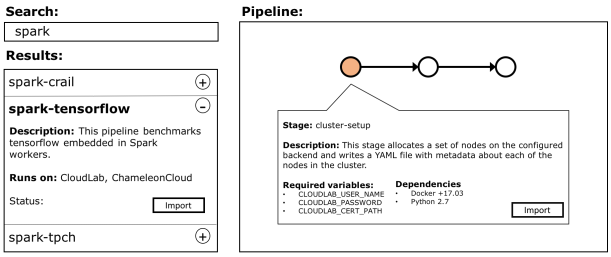

Figure 2: Sketch of the pipeline builder GUI.

Automated Validation

A third key component is the notion of pipeline validations. A validation is domain-agnostic way of checking that domain-specific results are valid. In other words, these are checks that verify that the outcome of the execution of a pipeline is as the original authers expected it to be. Black Swan will incorporate facilities to quickly visualize the status of a pipeline with respect to domain-specific validations. Fig. 3 shows a concept of this feature.

ivotron / my-paper-repository

experiment	commit	message	time	date	status
spark-ml	d4baa54	Using parameter...	23 mins 10 secs	2016-12-20T11:30:32Z	GOLD
spark-ml	fa3135ac	Fixed bug in funct...	22 mins 58 secs	2016-12-20T09:13:47Z	SUCCESS
spark-ml	b92560d	Finished experim...	2 mins 46 secs	2016-12-19T23:56:01Z	FAIL

Figure 3: A sketch of a dashboard with validations as first-class citizens in the GUI.

Environment Capture and Automated Analysis

When the validation stage for a pipeline fails, we have found a black swan, i.e. the domain-specific expectations of the re-execution of a pipeline have not hold. The next task is to find why. Black Swan will incorporate facilities to automatically capture the execution environment for a pipeline, in machine readable format so that a post-processing step can analyze it and compare it against previous successful executions in order to try to determine root causes of irreproducibility, or to aid researchers in finding them

4 Use cases

We describe three use cases for which we envision Black Swan to be applicable. We envision many more but we see these as key use cases.

Technology Transfer

Organizations with *Research and Development* units such as tech companies and government-funded institutions (DOE labs, NASA, universities) spend significant amounts of resources transferring technology from R&D to production environments. Deploying an instance *Black Swan* internally, using a public one (or connecting a public with a private one), would allow organizations to streamline tech (and knowledge) transfers.

Research Curation

In the past decade, institutional libraries have invested a significant amount of resources to the creation of *Data Repositories*. These efforts are aimed at systematically managing the output of research. For example, EU's OpenAIRE initiative is a catalog of software and data containing more than 80 million entries (each being a code repository or dataset). Similar efforts are underway in the US such as the Center for Open Science's Open Science Framework. We see *Black Swan* complementing these efforts, since Popper enables the curation of research by enabling all these existing repositories to be easily executed and validated over time.

Self-validation of Academic Artifacts

A growing number of Computer Science conferences and journals incorporate an artifact evaluation process in which authors of an article submit an artifact description[4] (AD) that is tested by a committee, in order to verify that experiments presented in a paper can be re-executed by others. Instead of manually creating and testing an AD, Black Swan can be leveraged to automatically test for the validity of an experiment, assuming the pipeline(s) corresponding to an article have codified expectations on their results.

[4] http://ctuning.org/ae/submission.html

5 Related Software

Continuous Integration (CI) Software

Black Swan will leverage CI services to execute and validate research pipelines. OSS communities such as those developing and using Gitlab, Github and Jenkins can benefit from this platform.

Code/Dataset Management Software

OSS communities developing code and data management repositories such as DOE Code, Zenodo and OSF can leverage Black Swan to add executable features to the repositories that they curate.

6 Challenges

Managing changes to code using version-control systems; managing data with dataset management systems; continuously integrating (CI) and deploying (CD) software; all have become standard practices in OSS communities, not because of a fad but because of the quantifiable benefits that following best practices imply. To the contrary, in R&D settings, these practices are seen as a burden. The biggest challenge we face lies in changing the culture within organizations and teams. Finding the right incentives so that these users can make the leap and adopt agile practices, so they can enjoy the benefits that come from embracing DevOps.

References

[1] D.L. Donoho, A. Maleki, I.U. Rahman, M. Shahram, and V. Stodden, "Reproducible Research in Computational Harmonic Analysis," *Computing in Science & Engineering*, vol. 11, Jan. 2009.

[2] G. Fursin, "Collective Mind: Cleaning up the research and experimentation mess in computer engineering using crowdsourcing, big data and machine learning," *arXiv:1308.2410 [cs, stat]*, Aug. 2013. Available at: http://arxiv.org/abs/1308.2410.

[3] C. Collberg and T.A. Proebsting, "Repeatability in computer systems research," *Communications of the ACM*, vol. 59, 2016, pp. 62–69.

[4] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "The popper convention: Making reproducible systems evaluation

practical," *Parallel and distributed processing symposium workshops (ipdpsw), 2017 ieee international*, IEEE, 2017, pp. 1561–1570.

[5] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook*, O'Reilly Media, 2016.