Experiences Following the Popper Convention

Michael A. Sevilla, Ivo Jimenez, Carlos Maltzahn
University of California
{msevilla, ivo}@soe.ucsc.edu, {carlosm}@ucsc.edu

***Abstract*—Popper is pretty hard.**

I. Introduction

II. Popper-compliant Papers

We have authored five Popper-compliant papers:

- quiho: Automated Performance Regression Testing Using Inferred Resource Utilization Profiles []
- Cudele: An API and Framework for Programmable Consistency and Durability in a Global Namespace [1]
- Malacology: A Programmable Storage System [2]
- Characterizing and Reducing Cross-Platform Performance Using OS-level Virtualization
- Programmable Caches with a Data Management Language & Policy Engine

We use the reader/reviewer sample workflow outlined in [3] and shown in Figure 1. For the visualization component (1), we use Jupyter notebooks. The notebooks themselves are versioned with Git and users interact with local copies by cloning the repository and launching a Jupyter Docker container. The paper is written in LaTeXand built with a Docker container. For the code component (2), both the source code for the system itself and the deploy/experiment code is stored on GitHub. When running experiments, we use Docker containers to isolate libraries and binaries. For the multi-node component (3), we use CloudLab machines and Ansible to script deployment and experiment orchestration. For the data set components (4), we use GitHub to store results files; our inputs and results are small enough that we do no need a larger capacity. GitHub allows files up to 50MB and stores data on S3.

Our experiments start with a baseline; to describe the process, we reference our ceph-popper-template set up on CloudLab. Users setup SSH keys and deploy CloudLab nodes using our CephFS Profile. The profile has the nodes automatically install Docker on bootup using our install script. After the nodes finish booting (i.e. their status on the CloudLab GUI read as READY), users push SSH keys using a convenience script.

The deploy code is based on ceph-ansible, a tool that configures hardware and software for Ceph. We forked the project and made it less dependent on Python. To run an experiment, users log in into the head node and clone the ceph-popper-template. This repository has submodules that point to ceph-ansible and our own custom roles; configuration files for our Ceph setup; and helper scripts written in bash that deploy Ceph and run the benchmarks. For more information, see the

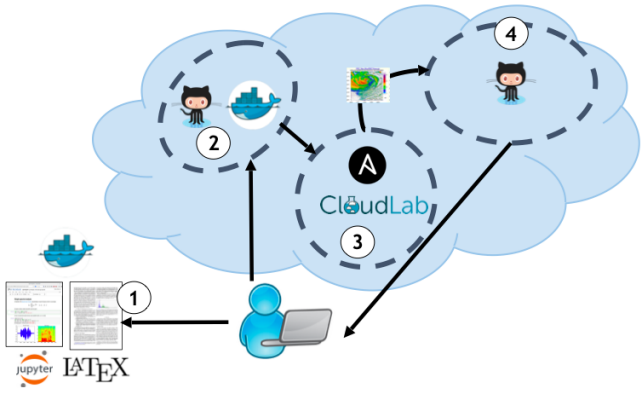

Fig. 1: Illustration of the Popper workflow used in our papers; Figure is adapted from [3]. For (1) - the visualization component - we use Jupyter, LaTeX, and Docker; for (2) - the code component - we use GitHub and Docker; for (3) - the multi-node component - we use Ansible and CloudLab; and for (4) - the data set component - we use GitHub.

README. For more information on the baseline and pipelines terminology, read more on the Popper Convention. Then users configure their cluster by specificing IPs and user names in the `hosts` file.

Finally, users specify the Ceph services that should be deployed using the `ansible/` directory. This directory has code for deploying Ceph and its components: `ansible.cfg`, `ceph.yml`, `cleanup.yml`, `group_vars`, `monitor.yml`, and `workloads`. The *.yml files are Ansible playbooks that start and configure components: ceph.yml starts Ceph, cleanup.yml tears Ceph down, and monitor.yml starts daemons that monitor performance. We separate these components into different playbooks so users can mix and match Ceph services. The workloads directory has scrips for running the baseline benchmarks. The other files and directories are Ansible configuration files used by the playbooks.

Users can change the ansible/ceph.yml to specify which Ceph daemons to launch in the cluster. It uses the hosts file we set up above and is based off the ceph-ansible site file. High level configurations are in the vars.yml. This file is heavily commented. For CloudLab, uncomment all blocks labeled “Uncomment for CloudLab”.

To configure the Ceph cluster with the variables in the Ceph configuration file documentation, change the Ansible `group_vars/all` file. We also need to specify the Docker

image. These can be specified in each configuration file for the daemon but for simplicity we put everything in the global variable file. Finally, users run the `run.sh` script.

III. Reproducibility Must Be a 1st Class Citizen

IV. Organized Repositories and Documentation

V. Well-Defined Collaboration Roles

VI. Maintaining Pointers

VII. Conference Requirements

VIII. Conclusion

References

[1] M. A. Sevilla, I. Jimenez, N. Watkins, S. Finkelstein, J. LeFevre, P. Alvaro, and C. Maltzahn, "Cudele: An API and Framework for Programmable Consistency and Durability in a Global Namespace," in *Proceedings of the 32nd IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '18, 2018.

[2] M. A. Sevilla, N. Watkins, I. Jimenez, P. Alvaro, S. Finkelstein, J. LeFevre, and C. Maltzahn, "Malacology: A Programmable Storage System," in *Proceedings of the European Conference on Computer Systems*, ser. EuroSys '17, 2017.

[3] I. Jimenez, M. A. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "The Popper Convention: Making Reproducible Systems Evaluation Practical," in *Proceedings of the International Parallel and Distributed Processing Symposium Workshop*, ser. IPDPSW '17.