Longitudinal Study of Software Environments Produced by Dockerfiles from Research Artifacts: Initial Design

ACM REP'25, Vancouver, BC, Canada

Quentin GUILLOTEAU, Antoine WAEHREN, Florina M. CIORBA

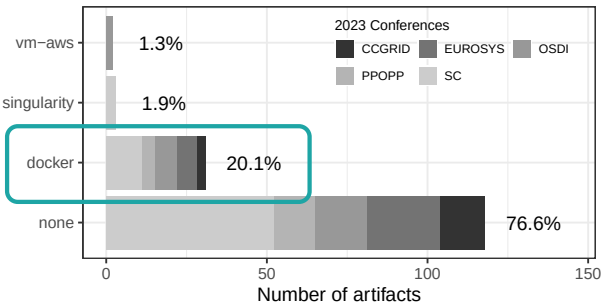
2025-07-30

University of Basel, Switzerland

University of Basel

Longevity (ACM REP'24)

- Who are artifacts for? ⇝ **Future researchers**
- Math proof will not **disappear** or **change**
- Conferences recommend using containers



Longevity of Artifacts in Leading Parallel and Distributed Systems Conferences: a Review of the State of the Practice in 2023

Quentin Guilloteau
Florina M. Ciorba
Quentin.Guilloteau@unibas.ch
Florina.Ciorba@unibas.ch
University of Basel
Basel, Switzerland

Millian Poquet
Millian.Poquet@ens.fr
Univ. Toulouse, CNRS, IRIT
Toulouse, France

Dorian Goepp
Olivier Richard
Dorian.Goepp@imta.fr
Olivier.Richard@imta.fr
Univ. Grenoble Alpes, Inria, CNRS, LIG
Grenoble, France

ABSTRACT

Reproducibility is the cornerstone of science. Many scientific communities have been struck by the reproducibility crisis, and computer science is no exception. Its answer has been to require artifact evaluations along with accepted articles and award badges to reward authors for their efforts to support 'reproducibility.' Authors voluntarily submit artifacts associated with a submission to reviewers who double-check 'reproducibility' properties. We argue that the notion of 'reproducibility' considered by such badges is limited and misses important aspects of the reproducibility crisis. In this article, we survey almost 300 articles from five leading conferences on parallel and distributed computing (based on publication rates), ASPLOS, PPoPP, and SC). For each article, we gather information about its artifacts (how it was shared, under which permissions), and how the software environment was generated and shared, as well as the reproducibility badges awarded. By reviewing the methods and tools used to create and share artifacts in a technical, in-depth, and article-context-agnostic manner, we found that the state of practice does not address reproducibility in terms of artifact longevity and we expose eight observations that support this finding. To address the longevity of artifacts, we propose a new badge based on source code, experimental setup, and software environment. The criteria will allow rewarding artifacts expected to withstand the test of time. This work aims to shed light on the issue of long-term reproducibility in parallel and distributed systems and to start discussion in the community towards addressing the issue.

CCS CONCEPTS

• **General and reference** → **Empirical studies**

KEYWORDS

Reproducibility, Artifact Evaluation, Badges, Longevity

ACM Reference Format:

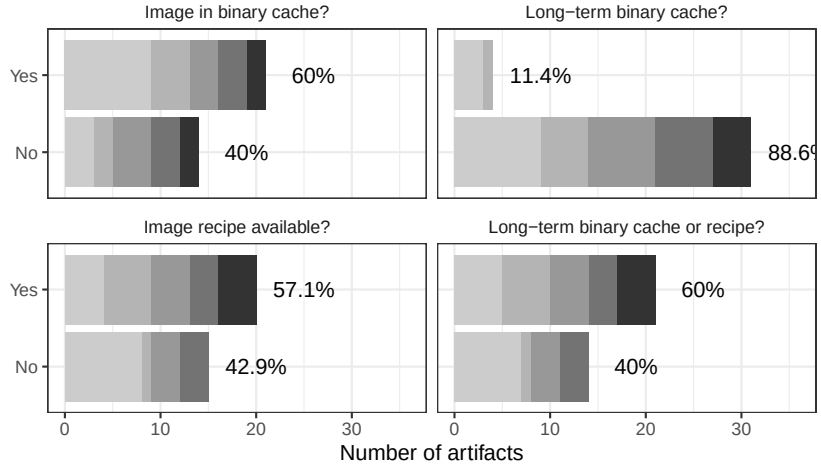
Quentin Guilloteau, Florina M. Ciorba, Millian Poquet, Dorian Goepp, and Olivier Richard. 2024. Longevity of Artifacts in Leading Parallel and Distributed Systems Conferences: a Review of the State of the Practice in 2023. In *ACM Conference on Reproducibility and Applicability (ACM REP '24), June 18–20, 2024, Rennes, France.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3641525.3663631

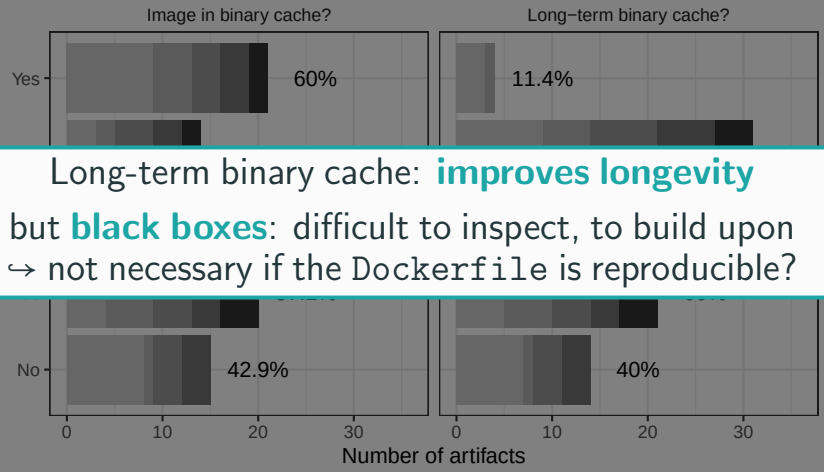
1 INTRODUCTION

The scientific community as a whole is traversing a reproducibility crisis for the last decade. Computer science is not an exception to this crisis [4, 65]. The reproducibility of research is essential to build solid knowledge and increase reliability and confidence in the results, while limiting the methodology and analysis bias. In 2015, Collberg et al. [15] studied the reproducibility of 402 experimental articles published in system conferences and journals of 2011 and 2012. Each of the articles studied linked the source code used to perform their experiments. Of the 402 articles, 46% were not reproducible. The main causes were: (i) the source code was not available, (ii) the code did not compile or run, (iii) the experiments required specific hardware.

To reward authors of reproducible articles, several publishers, such as ACM or Springer, set up a peer review-based artifact evaluation for each submission. This peer review process of the experimental artifact can award one or several badges to the authors based on the level of reproducibility of their artifacts.

Docker, Longevity, and Sustainability





Long-term binary cache: **improves longevity**

but **black boxes**: difficult to inspect, to build upon
↪ not necessary if the `Dockerfile` is reproducible?

Research Questions

Is Docker really suitable for **longevous and reproducible** research?

(Should Reproducibility Chairs stop recommending Docker as a suitable solution to authors?)

Research Questions

Is Docker really suitable for **longevous and reproducible** research?

(Should Reproducibility Chairs stop recommending Docker as a suitable solution to authors?)

↪ How do the software environments produced by `Dockerfiles` **evolve through time**?

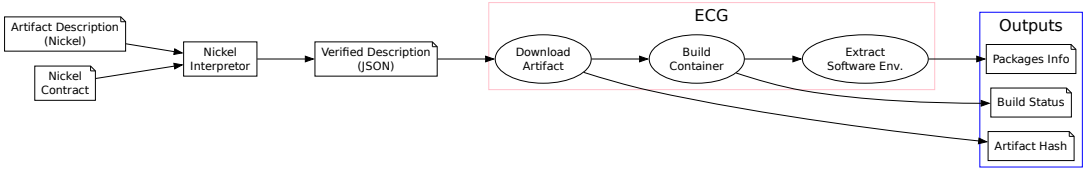
Is Docker really suitable for **longevous and reproducible** research?

(Should Reproducibility Chairs stop recommending Docker as a suitable solution to authors?)

Study: Take `Dockerfiles` from Research Artifacts, **build** them periodically, and **capture** the resulting software environment.

↪ How do the software environments produced by `Dockerfiles` **evolve through time**?

Workflow, Data collected, and Frequency



What are we capturing?

- **Artifact Hash**: Did the content of the artifact change? (what's behind the link)
- **Build Status**: Did the container build successfully? What were the errors?
- **Packages Info**: What are the versions of the packages in the SW environment?
  - ↪ **Package Managers** (`apt`, `dpkg`, `pip`, `conda`), **Manual Installs.** (`git`, `curl`/`wget`)

When are we capturing? And for how long?

At the start of **each month**, for a **full year** (13 captures)

Scope of this Preliminary Study

5 artifacts from Euro-Par 2025 ⇝ all the artifacts using a `Dockerfile`

- Conference in our field (HPC) ⇝ **we are familiar with SW stacks**
- Artifacts published when we finished to develop our workflow ⇝ **"fresh" artifacts**

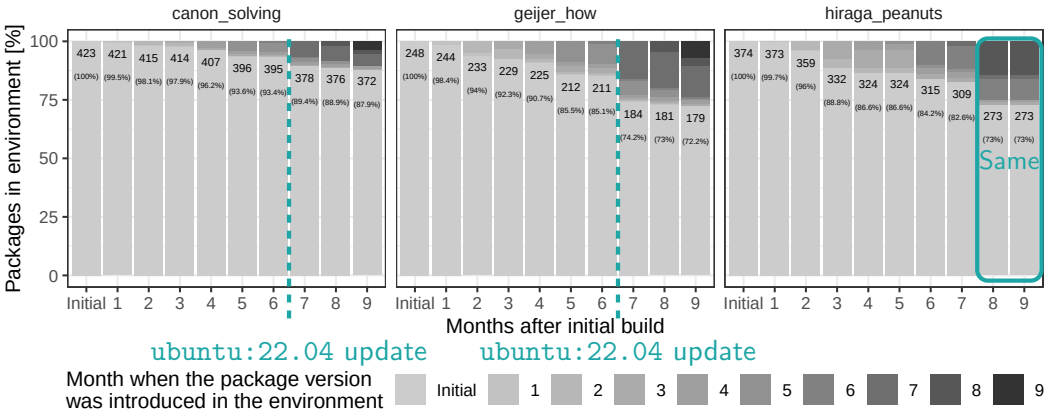
Artifact	Docker Base Image used		Calling `apt update`?
	Name	Version	
`canon_solving`	`ubuntu`	`22.04`	Yes
`geijer_how`	`ubuntu`	`22.04`	Yes
`hiraga_peanuts`	`devcontainers/cpp`	`1-debian-12`	Yes
`munoz_fault`	`ubuntu`	`22.04`	Yes
`wolff_fast`	`ubuntu`	`22.04`	Yes

Table 1: Information about the `Dockerfiles` from the study.

From October 1st 2024 to ~~April 1st 2025~~ July 1st 2025 ⇝ **10 months**

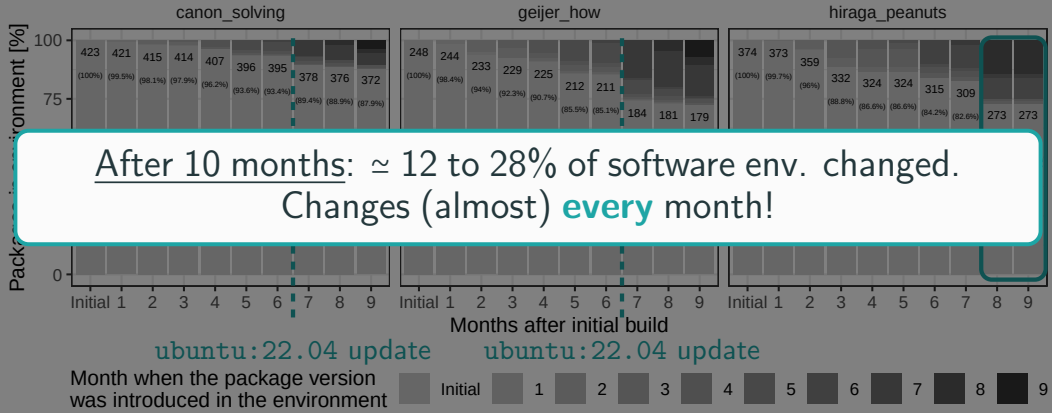
Preliminary Results – Per Artifact

Evolution of the packages versions over time for selected Dockerfile studied



Preliminary Results – Per Artifact

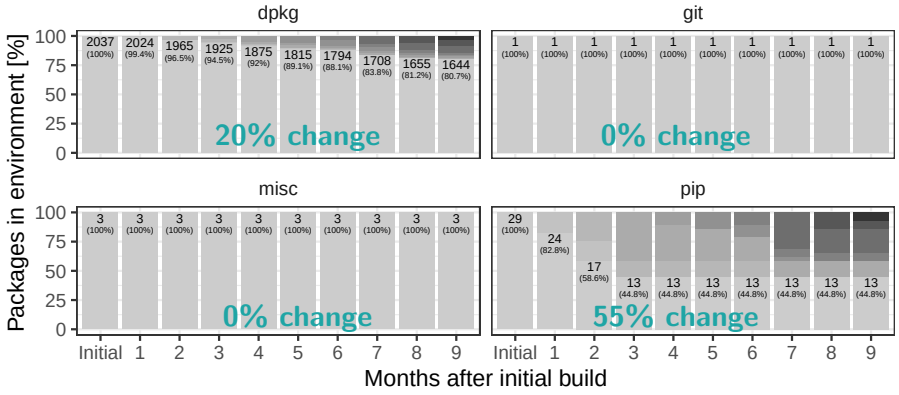
Evolution of the packages versions over time for selected Dockerfile studied



After 10 months: ≃ 12 to 28% of software env. changed.
Changes (almost) **every** month!

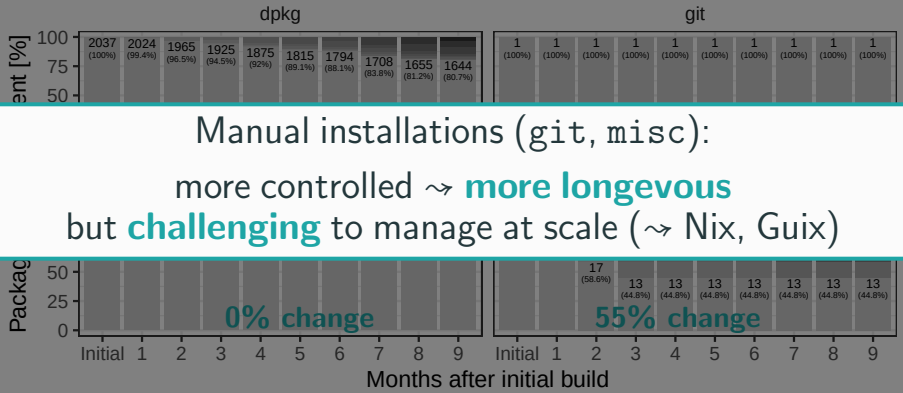
Preliminary Results – Per Tool

Evolution of the packages versions over time



Preliminary Results – Per Tool

Evolution of the packages versions over time



Manual installations (`git`, `misc`):

more controlled ⇝ **more longevous**
but **challenging** to manage at scale (⇝ Nix, Guix)

Month when the package version was introduced in the environment

Initial 1 2 3 4 5 6 7 8 9

Conlusion and Future Work

How do the software environments produced by `Dockerfiles` from Artifacts evolve through time?

Preliminary Results

- **Software Env. changed within a month! $\rightsquigarrow$ same period than the AE !**
- Only 5 artifacts ☹ (how significant / representative ?)

Future Work

- Design of the **large scale study** (How many artifacts? Which conferences?)
- Capture the hash of the base Docker image
- Other containerization tools? Other package managers?
- **Wanna help? Contact us!** ☺