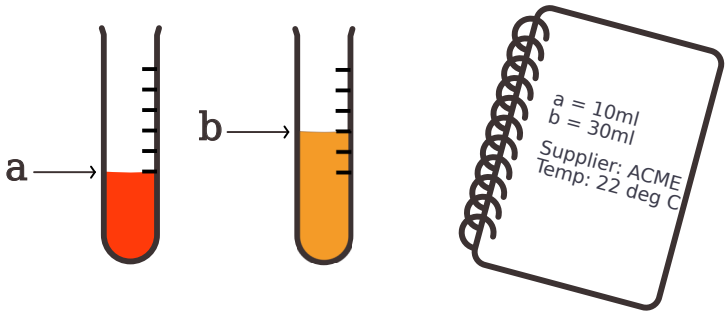
Guix-HPC Activity Report 2021–2022

Céline Acary-Robert, Ludovic Courtès, Yann Dupont, Marek Felšöci, Konrad Hinsen, Ontje Lünsdorf, Pjotr Prins, Philippe Swartvagher, Simon Tournier, Ricardo Wurmus

GuixHPC

Reproducible software deployment for high-performance computing.



Activity Report 2021–2022

10 February 2023

Céline Acary-Robert, Ludovic Courtès, Yann Dupont, Marek Felšöci,
Konrad Hinsen, Ontje Lünsdorf, Pjotr Prins, Philippe Swartvagher, Simon Tournier,
Ricardo Wurmus

2.

Guix-HPC is a collaborative effort to bring reproducible software deployment to scientific workflows and high-performance computing (HPC). Guix-HPC builds upon the GNU Guix[1] software deployment tools and aims to make them useful for HPC practitioners and scientists concerned with dependency graph control and customization and, uniquely, reproducible research.

Guix-HPC was launched in September 2017 as a joint software development project involving three research institutes: Inria[2], the Max Delbrück Center for Molecular Medicine (MDC)[3], and the Utrecht Bioinformatics Center (UBC)[4]. GNU Guix for HPC and reproducible science has received contributions from additional individuals and organizations, including CNRS[5], Université Paris Cité[6], the University of Tennessee Health Science Center[7] (UTHSC), the Leibniz Institute for Psychology[8] (ZPID), Cornell University[9], and a growing number of organizations deploying Guix on their HPC clusters.

---

[1] *https://guix.gnu.org*

[2] *https://www.inria.fr/en/*

[3] *https://www.mdc-berlin.de/*

[4] *https://ubc.uu.nl/*

[5] *https://www.cnrs.fr/en*

[6] *https://u-paris.fr/en/*

[7] *https://uthsc.edu/*

[8] *https://leibniz-psychology.org/*

[9] *https://www.csl.cornell.edu/*

This report highlights key achievements of Guix-HPC between our previous report[10] a year ago and today, February 2023. This year was marked by exciting developments for HPC and reproducible workflows: the release of GNU Guix 1.4.0 in December[11], the celebration of ten years of Guix with a three-day conference[12], several releases of the Guix Workflow Language (GWL), more work on supporting RISC-V processors, and more publications relying on Guix as a foundation for reproducible computational workflows.

[10] *https://hpc.guix.info/blog/2022/02/guix-hpc-activity-report-2021/*

[11] *https://guix.gnu.org/en/blog/2022/gnu-guix-1.4.0-released/*

[12] *https://10years.guix.gnu.org*

Outline

Guix-HPC aims to tackle the following high-level objectives:

- *Reproducible scientific workflows.* Improve the GNU Guix tool set to better support reproducible scientific workflows and to simplify sharing and publication of software environments.
- *Cluster usage.* Streamlining Guix deployment on HPC clusters, and providing interoperability with clusters not running Guix.
- *Outreach & user support.* Reaching out to the HPC and scientific research communities and organizing training sessions.

The following sections detail work that has been carried out in each of these areas.

6.

Reproducible Scientific Workflows

Supporting reproducible research workflows is a major goal for Guix-HPC.

Guix Workflow Language

The Guix Workflow Language[13] (or GWL) is a scientific computing extension to GNU Guix's declarative language for package management. It allows for the declaration of scientific workflows, which will always run in reproducible environments that GNU Guix automatically prepares. The general idea with the GWL is a simple inversion of priorities: put reproducible software deployment *first* and *extend* the deployment infrastructure provided by Guix with tools to declare and run workflows. As a consequence, the GWL benefits directly from the continued development of Guix's salient features pertaining to software reproducibility and reliable, predictable deployment. Much of the work on the GWL is thus aimed at recasting these features through the lens of a domain-specific language for describing workflows as a graph of processes that are inextricably linked with their associated software stacks.

The year 2022 saw three releases of the Guix Workflow Language: version 0.4.0 on January 28, version 0.5.0 on July 21, and version 0.5.1 on November 13, representing the cumulative efforts of four contributors. The changes include fixes to errors discovered in active use of the GWL for scientific workflows, adjustments in the details of how the GWL extends Guix, and laying the groundwork for improved performance.

The German National Research Data Infrastructure—specifically its engineering sciences branch NFDI4Ing[14]— recognizes workflow management systems as an important tool towards reproducible and reusable scientific

[13] *https://workflows.guix.info*

[14] *https://nfdi4ing.de/*

workflows. A special interest group discussed and compared several workflow management systems, including GWL, along three different user story perspectives. The discussion paper entitled "Evaluation of tools for describing, reproducing and reusing scientific workflows" highlights GWL's abilities to easily reproduce compute environments and to provide precise software provenance tracking as well as its flexible workflow definition. The special interest group recommends the GWL to specialists with high requirements on software reproducibility and integrity. The preprint of the discussion paper is available here[15].

Reproducible GNU R Environments

The R language is widely used for statistics in general and notably in bioinformatics. A common practice for installing R packages, from within the `R` session, is to run the `install.packages` utility: it allows users to download and install packages from CRAN[16] and CRAN-like repositories such as Bioconductor[17], or from local files.

While convenient, use of `install.packages` raises the question of the level of control over the software "supply chain". Some R packages are not just plain R scripts and instead also contain C, C++, or Fortran parts, mainly for performance, or require external system-wide dependencies unmanaged by `install.packages`, such as linear algebra libraries. Therefore, computational environments populated with the builtin utility `install.packages` might not be reproducible from one machine to another.

This is where the `r-guix-install` package comes in. `r-guix-install`, which is available on CRAN[18], allows users to install R packages *via* Guix from within the running `R` session, similarly as `install.packages` but where the complete supply chain is controlled by Guix. In addition, if the requested R package does not exist in Guix at this time, the package and all

[15] *https://preprints.inggrid.org/repository/view/5/*

[16] *https://cran.org*

[17] *https://bioconductor.org/*

[18] *https://CRAN.R-project.org/package=guix.install*

its missing dependencies will be imported recursively and the generated package definitions will be written to `~/.Rguix/packages.scm`. This record of imported packages can be used later to reproduce the environment, and to add the packages in question to a proper Guix channel (or to Guix itself). `guix.install()` not only supports installing packages from CRAN, but also from Bioconductor or even arbitrary Git or Mercurial repositories, replacing the need for installation *via* `devtools`.

While this approach works well for individual users, Guix installations with a larger user base, for instance institution-wide, would benefit from the default availability of the entire CRAN package collection with pre-built substitutes to speed up installation times. Additionally, reproducing environments would include fewer steps if the package recipes were available to anyone by default.

The new `guix-cran`[19] channel was built to address that issue. It extends the package collection by providing all CRAN packages missing in Guix proper and has all of the properties mentioned above.

Creating and updating `guix-cran` is fully automated and happens without any human intervention. The channel itself is always in a usable state, because updates are tested with `guix pull` before committing and pushing them. However, some packages may not build or work, because build or runtime dependencies (usually undeclared in CRAN itself) are missing. Any improvement to the already very good Guix CRAN importer, like enhanced auto-detection of these missing dependencies, also improves the channel's quality. More details are available in a blog post[20].

Packages

As of this writing, Guix comes with more than 22,000 packages, which makes it one of the ten biggest free software distributions according to

[19] *https://github.com/guix-science/guix-cran*

[20] *https://hpc.guix.info/blog/2022/12/cran-a-practical-example-for-being-reproducible-at-large-scale-using-gnu-guix/*

Repology[21]. This is the result of more than 15,000 commits made by 343 people since last year—an impressive level of activity sustained thanks to the Guix tooling and continuous integration services.

Many scientific packages have been added or upgraded in Guix. As an example, Bioconductor, the R suite for bioinformatics, was upgraded to 3.16; OCaml 5 with support for shared memory parallelism and effect handlers was introduced; the `snakemake` package in Guix received an important bug fix, making `snakemake` usable for parallel execution on HPC clusters. The most common scientific and HPC packages were updated and improved: Open MPI and its many dependencies, SLURM, OpenBLAS, Scotch, SUNDIALS, and GROMACS, to name a few. The Julia package set is still growing; Julia was upgraded to 1.6.7 and then to 1.8.3, with fixes for i686 and improvements of the Julia build system.

In addition to the growing collection of curated packages provided as part of the main Guix channel, we maintain a number of special-purpose channels that provide additional packages for scientific and high-performance computing. An up-to-date list of Guix channels maintained by members of the Guix HPC effort is available on the project page[22]. The on-line package package browser[23] also makes it easier to navigate channels.

The Guix-Science channel[24], initiated in 2021, now provides more than 600 packages, complementing the rich scientific package collection available in Guix proper. Chief among the changes it received this year are an update of R Studio[25] and improvements to the Jupyter Lab and Jupyter Hub packaging, and the addition of Integrative Genomics Viewer (IGV).

Ensuring Source Code Availability

[21] *https://repology.org*

[22] *https://hpc.guix.info/channels/*

[23] *https://hpc.guix.info/browse*

[24] *https://github.com/guix-science/guix-science*

[25] *https://hpc.guix.info/package/rstudio*

The 10 Years of Guix[26] event was an opportunity for developers of Guix and Software Heritage (SWH) to discuss *intrinsic identifiers*. An intrinsic identifier only depends on the data content itself and it requires three ingredients for its computation: a representation of the *structure* of this content (serializer), a cryptographic hash algorithm, and an encoding for the resulting byte string. While converting from one encoding to another is trivial—e.g., between base64 and base32—it is, naturally, impossible to "convert" a cryptographic hash to the hash computed by a different function. All three parameters can be selected with command-line options to the `guix hash` command.

By default Guix computes a SHA256 hash over the Nar serialization of source archives and version-control checkout ("Nar" stands for *normalized archive*; it is the serialization format inherited from Nix). Instead, the SWH archive computes the SHA1 hash of a Git-serialized representation of the files[27]. This discrepancy deprives Guix of a simple and reliable way to query the SWH archive by content hash. This led to a discussion[28] about the possibility for SWH to compute and preserve Nar hashes as additional information for code it archives—so-called `ExtID` identifiers. Doing so could improve archive coverage for code source referenced by Guix packages, in particular for Subversion checkouts as used by most of the TeX Live packages.

As discussed in last year's report, Guix contributor Timothy Sample was awarded a grant by Software Heritage and the Alfred P. Sloan Foundation[29] to further their work on Disarchive. Disarchive[30] bridges the gap between source code archives (*tarballs*) packages refer to and content stored in the SWH archive. It does so by providing a command to *extract* the

---

[26] *https://10years.guix.gnu.org/*

[27] *https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html*

[28] *https://gitlab.softwareheritage.org/swh/meta/-/issues/4538*

[29] *https://www.softwareheritage.org/2022/01/13/preserving-source-code-archive-files/*

[30] *https://ngyro.com/software/disarchive.html*

metadata of a tarball, and another command to *reassemble* metadata and content, thereby restoring the original tarball. This work is key to improving source code availability for the many packages built from source code tarballs.

Last year, the Guix project deployed infrastructure to continuously build and publish a Disarchive database at `disarchive.guix.gnu.org`[31]. Guix is able to combine Disarchive and SWH as a fallback when downloading a tarball from its original URL fails, significantly improving source code archival coverage.

This work was initiated[32] a few years back and is still ongoing. A proposal to integrate Disarchive into the SWH archive is being discussed[33]. We believe Disarchive integration would be a great step forward, not just for Guix, but for all the distributions and tools that rely on source tarball availability.

Reproducible Research in Practice

This section highlights scientific productions made with GNU Guix.

Guix was used to ensure the reproducibility of experiments for the study of memory contention between computations and communications on several different HPC clusters. A public companion[34] explains how to reproduce the experiments with and without GNU Guix.

- Alexandre Denis et al., *Predicting Performance of Communications and Computations under Memory Contention in Distributed HPC Systems*[35]

The reproducible paper[36] about the impact of tracing on complex HPC application executions, mentioned in the previous Guix-HPC Activity Re-

[31] *https://disarchive.guix.gnu.org/*

[32] *https://gitlab.softwareheritage.org/swh/devel/swh-model/-/issues/2430#note_23708*

[33] *https://gitlab.softwareheritage.org/swh/devel/swh-model/-/merge_requests/267*

[34] *https://gitlab.inria.fr/pswartva/paper-model-memory-contention-r13y*

[35] *https://hal.inria.fr/hal-03871630v1*

port, is still under review for publication. However, first feedbacks from reviewers requested several complementary experiments. These complementary experiments were made about a year after the initial experiments presented in the paper. Having a complete workflow based on GNU Guix really helped to dive back into the experimental context and configurations used a year before!

Philippe Swartvagher defended his PhD thesis[37] on the interactions between HPC task-based runtime systems and communication libraries. In an appendix of the manuscript, he explains how he used on different HPC clusters GNU Guix, packages from the Guix-HPC channel and Software Heritage, to ensure reproducibility of his experiments.

The PhD thesis of Marek Felšöci (to be defended in February 2023), which is part of a collaboration between Inria and Airbus, is set in an industrial aeroacoustic context and deals with direct methods for solving coupled sparse/dense linear systems. Within the thesis, the author dedicates a full chapter to the topic of reproducibility. Throughout this chapter, he addresses the challenges of ensuring a reproducible research study in computer science in general and in the context of the thesis in particular. The questions related to the usage of non-free software are discussed as well. The author then presents the strategy he adopts to face these challenges including working principles, software tools and their alternatives. To share the resulting guidelines, he provides a minimal working example of a reproducible research study on solvers for coupled sparse/dense systems. Moreover, he introduces and references examples of actual studies from the thesis following the advocated principles and techniques for improving reproducibility.

The latest addition to the PiGx framework of reproducible scientific workflows backed by Guix[38] is PiGx SARS-CoV-2, a pipeline for analysing

[36] *https://gitlab.inria.fr/pswartva/paper-starpu-traces-r13y*

[37] *https://theses.hal.science/tel-03989856*

data from sequenced wastewater samples and identifying given lineages of SARS-CoV-2. The output of the PiGx SARS-CoV-2 pipeline is summarized in a report which provides an intuitive visual overview about the development of variant abundance over time and location. This is the first of the released PiGx pipelines that comes with concise yet comprehensive instructions on how to use `guix time-machine` to reproduce the software environment used for the analyses presented in the paper:

- Vic-Fabienne Schumann et al., *SARS-CoV-2 infection dynamics revealed by wastewater sequencing analysis and deconvolution*[39]

Guix was used as the computational environment manager of biomedical research on the administration of azithromycin drug after allogeneic hematopoietic stem cell transplantation for hematologic malignancies. Studying 240 samples from patients randomized in this phase 3 controlled clinical trial was a unique opportunity to better understand the mechanisms underlying relapse, the first cause of mortality after transplantation. The various data processing scripts and associated computational environments using `manifest.scm` and `channels.scm` files for use with `guix time-machine`[40] and `guix shell`[41] are available here[42], there[43] or there[44].

- Nicolas Vallet et al. *Azithromycin promotes relapse by disrupting immune and metabolic networks after allogeneic stem cell transplantation*[45]

---

[38] *https://bioinformatics.mdc-berlin.de/pigx/*

[39] *https://doi.org/10.1016/j.scitotenv.2022.158931*

[40] *https://guix.gnu.org/manual/en/html_node/Invoking-guix-time_002dmachine.html*

[41] *https://guix.gnu.org/manual/en/html_node/Invoking-guix-shell.html*

[42] *https://gitlab.com/nivall/azimut-blood*

[43] *https://gitlab.com/nivall/azimut-in-vitro*

[44] *https://gitlab.com/nivall/azimutscrna*

[45] *https://doi.org/10.1182/blood.2022016926*

Cluster Usage and Deployment

As part of our effort to streamline Guix deployment on HPC clusters, we updated and improved our cluster installation guide[46], which is now part of the Guix Cookbook. The guide describes the steps needed to get Guix running on a typical HPC cluster where nodes come with a distribution other than Guix System, such as CentOS or Rocky Linux.

The sections below highlight the experience of cluster administration teams and report on tooling developed around Guix for users and administrators on HPC clusters.

Genetics Research Cluster at UTHSC

At the University of Tennessee Health Science Center (UTHSC) in Memphis (USA), we are running an 11-node large-memory HPC Octopus cluster[47] (264 cores) dedicated to pangenome and genetics research. In 2022 more storage added. Notable about this HPC is that it is *administered by the users themselves.* Thanks to GNU Guix we install, run and manage the cluster as researchers (and roll back in case of a mistake). UTHSC's information technology (IT) department manages the infrastructure—i.e., physical placement, routers and firewalls—but beyond that there are no demands on IT. Thanks to out-of-band access we can completely (re)install machines remotely.

Octopus runs GNU Guix on top of a minimal Debian install and we are experimenting with Guix System nodes that can be run on demand. LizardFS is used for distributed network storage. Almost all deployed software has been packaged in GNU Guix and can be installed by regular users on the cluster without root access, see the guix-bioinformatics[48] channel.

---

[46] *https://guix.gnu.org/cookbook/en/html_node/Installing-Guix-on-a-Cluster.html*

[47] *http://genenetwork.org/facilities/*

[48] *https://gitlab.com/genenetwork/guix-bioinformatics*

Tier-2 Research Cluster at GliCID

GliCID[49], a Tier-2 cluster in Nantes (France), will have a new computing cluster installed in the summer of 2023. To retain control over the system and avoid proprietary tools specific to this type of facility, GliCID chose to build an independent cluster infrastructure into which the newly delivered cluster will be integrated.

This infrastructure consists of virtual machines (VMs) generated from Guix operating system definitions and providing services such as: identity management, databases, monitoring, high availability, login machines, slurms servers, documentation servers—over 20 VMs in total. The generated images are directly pushed on Ceph RBD volumes and consumed by KVM hypervisors, which avoids a deployment phase. Now fully operational, this infrastructure is entering a test phase. The choice of Guix has proven to be perfectly adapted to control the whole infrastructure and to obtain redeployable, reproducible and easily scalable machines.

Compute nodes are a mix of virtual compute machines running Guix System, and physical machines from a previous cluster running another distribution. Making native and "foreign" Guix installations cohabit while guaranteeing the consistency of the profiles turned out to be challenging. One specific issue GliCID overcame was managing a shared independent `/gnu/store`, shared by all the nodes as per the standard cluster setup instructions[50], and merging the `/gnu/store` directory of native nodes via overlayfs.

In 2023, GliCID plans to increase the share of infrastructure machines running Guix System, to factor more code and improve the quality of operating system definitions, packages, and services that have been developed internally[51], and to contribute more of these upstream.

---

[49] *https://www.glicid.fr/*

[50] *https://guix.gnu.org/cookbook/en/html_node/Installing-Guix-on-a-Cluster.html*

[51] *https://gitlab.univ-nantes.fr/glicid-public/guix-glicid*

Packages as Environment Modules

To support seasoned HPC users and system administrators, we developed Guix-Modules[52], a tool to generate *environment modules*[53]. Environment modules are a venerable tool that lets HPC users "load" the software environment of their choice *via* the `module load`. This gives a lot of flexibility: users can use their favorite software packages without interfering with one another, and they can also manipulate different environments. The downside of this tool that modules are all too often handcrafted on each cluster: an `openmpi/4.1.4` module might be called differently on another cluster, or it might be a different version, or it might be built with different options. In other words, use of modules is usually specific to one cluster, and users have to "port" their code when switching to a different clusters as they cannot expect to find the same modules.

Nevertheless, the `module` command remains widespread, well-known, and convenient. Guix-Modules generates modules for the chosen Guix packages, such that users can then run `module load` to use them, without having any knowledge of Guix. For system administrators, the benefit is obvious: instead of having to build and maintain tens of modules for scientific software, they can instead generate them all at once and provide users with battle-tested packages found in Guix. For users, the immediate benefit is a smooth transition to Guix, but also reproducibility and provenance tracking: the generated modules record provenance information, which allows users to deploy the exact same software elsewhere or at a different point in time.

A similar interoperability layer was previously developed for the Spack and EasyBuild package managers with similar motivations. In the case of Guix, we hope this will help user accustomed to `module` migrate

[52] *https://hpc.guix.info/blog/2022/05/back-to-the-future-modules-for-guix-packages/*

[53] *http://modules.sourceforge.net/*

towards reproducible deployment without having to change their habits overnight.

Containers, Singularity, and Docker

For HPC environments that do not support running native Guix software deployment Guix supports building lightweight reproducible containers that only have the software that is really needed. At UTHSC we are distributing binary deployments as Docker containers that run on state-of-the art compute HPCs. These containers were developed and tested first on a separate computer with GNU Guix installed, and produced with `guix pack`.

Research teams at Inria resort to `guix pack` as well when targeting supercomputers where Guix is not installed. Scientists can deploy their software using Guix directly on clusters that support it, such as Grid'5000, PlaFRIM, and some of the Tier-2 clusters; when they need to deploy it on Tier-1 supercomputers, they build a Singularity image that they ship and run there. This is both a productivity boost—no need to manually rebuild software!—and the guarantee that they *are* running the same software.

Having Guix available on those supercomputers would of course make the process even smoother; we plan to engage with those cluster administration teams to make Guix available in the future.

Supporting POWER9 and RISC-V CPUs

While it is perhaps early days to call RISC-V an HPC platform, there are indicators that this may happen in the near future with investments from the USA[54], the EU[55], India, and China. RISC-V hardware platforms and vendors will become common in the coming years.

---

[54] *https://www.tomshardware.com/news/risc-v-cluster-demonstrated*

[55] *https://www.european-processor-initiative.eu/epi-epac1-0-risc-v-test-chip-samples-delivered/*

Together with Chris Batten of Cornell and Michael Taylor of the University of Washington, Erik Garrison and Pjotr Prins are UTHSC PIs responsible for leading the NSF-funded RISC-V supercomputer for pangenomics[56]. It will incorporate GNU Guix and the GNU Mes bootstrap[57], with input from Arun Isaac, Efraim Flashner and others. NLNet[58] is funding RISC-V support for GNU Guix with Efraim Flashner and the GNU Mes RISC-V bootstrap project with Ekaitz Zarraga and Jan Nieuwenhuizen. We aim to continue adding RISC-V support to GNU Guix at a rapid pace. After the Guix days in Paris, Alexey Abramov was the first to bootstrap GNU Guix for RISC-V on the Polarfire platform.

Why is the combination of GNU Mes and GNU Guix exciting for RISC-V? First of all, RISC-V is a very modern modular open hardware architecture that provides further guarantees of transparency and security. It extends reproducibility to the transistor level and for that reason generates interest from the Bitcoin community, for example. Because there are no licensing fees involved, RISC-V is already a major force in IoT and will increasingly penetrate hardware solutions, such as storage microcontrollers and network devices, going all the way to GPU-style parallel computing and many-core solutions with thousands of cores on a single die. GNU Mes and GNU Guix are particularly suitable for RISC-V because Guix can optimize generated code for different RISC-V targets and is able to parameterize deployed software packages for included/excluded RISC-V modules.

---

[56] *https://news.cornell.edu/stories/2021/11/5m-grant-will-tackle-pangenomics-computing-challenge*

[57] *https://guix.gnu.org/en/blog/2019/guix-reduces-bootstrap-seed-by-50/*

[58] *https://nlnet.nl/project/current.html*

Outreach and User Support

Guix-HPC is in part about "spreading the word" about our approach to reproducible software environments and how it can help further the goals of reproducible research and high-performance computing development. This section summarizes articles, talks, and training sessions given this year.

Articles

The following refereed articles about Guix were published:

- Nicolas Vallet, David Michonneau, and Simon Tournier, *Toward practical transparent verifiable and long-term reproducible research using Guix*[59], Nature Scientific Data, volume 9 issue 1, October 2022
- Ludovic Courtès, *Reproducibility and Performance: Why Choose?*[60], IEEE CiSE volume 4, issue 3, June 2022
- Ludovic Courtès, *Building a Secure Software Supply Chain with GNU Guix*[61], Programming Journal, volume 7 issue 1, June 2022

The following refereed articles about research that uses Guix were published:

- Alexandre Denis et al., *Predicting Performance of Communications and Computations under Memory Contention in Distributed HPC Systems*[62]
- Vic-Fabienne Schumann et al., *SARS-CoV-2 infection dynamics revealed by wastewater sequencing analysis and deconvolution*[63], Science of the Total Environment, volume 853, December 2022

[59] *https://doi.org/10.1038/s41597-022-01720-9*

[60] *https://hal.inria.fr/hal-03604971*

[61] *https://doi.org/10.22152/programming-journal.org/2023/7/1*

[62] *https://hal.inria.fr/hal-03871630v1*

[63] *https://doi.org/10.1016/j.scitotenv.2022.158931*

- Nicolas Vallet et al. *Azithromycin promotes relapse by disrupting immune and metabolic networks after allogeneic stem cell transplantation*[64]

Over the year we published six articles on the Guix-HPC blog[65] touching topics such as environment modules, reproducible R environments, and reproducibility.

Talks

Since last year, we gave the following talks at the following venues:

- *Concise Common Workflow Language—Concision and Elegance in a Workflow Language Using Lisp*[66], FOSDEM, Feb. 2022 (Arun Isaac)
- *Using Guix in Computer Architecture Research* at both the gem5 users' workshop and the Sixth Workshop on Computer Archiecture Research with RISC-V (CARRV'22) in New York City, NY[67], June 2022, Christopher Batten (Cornell University), Pjotr Prins, Efraim Flashner, Arun Isaac (The University of Tennessee Health Science Center), Jan van Nieuwenhuizen (Joy of Source), Ekaitz Zarraga (ElenQ Technologies), Tuan Ta, Austin Rovinski (Cornell University), Erik Garrison (The University of Tennessee Health Science Center)
- *GNU Guix and the RISC-V Future*[68], Ten Years of Guix, Sep. 2022, (Pjotr Prins)
- *GNU Guix, vers la reproductibilité computationnelle*[69], BlueHats session of Open Source Experience[70], Nov. 2022 (Simon Tournier)

---

[64] *https://doi.org/10.1182/blood.2022016926*

[65] *https://hpc.guix.info/blog/*

[66] *https://archive.fosdem.org/2022/schedule/event/commonworkflowlang/*

[67] *https://carrv.github.io/2022/*

[68] *https://10years.guix.gnu.org/program/#gnu-guix-and-the-risc-v-future*

[69] *https://communs.numerique.gouv.fr/posts/annonce-programme-journee-bluehats-2022/*

[70] *https://www.opensource-experience.com/en/*

- *Toward practical transparent verifiable and long-term reproducible research using Guix*[71], bioinfo seminar at Institut de Biologie de l'École Normale Supérieure (IBENS)[72], Dec. 2022 (Simon Tournier)

Events

As in previous years, Pjotr Prins spearheaded the organization of the "Declarative and minimalistic computing" track[73] at FOSDEM 2022, which was home to several Guix talks.

This year was also the tenth year of Guix as a project. Its first lines of code were written in April 2012, and it has since received code contributions by more than 800 people at an impressive rate, not to mention non-coding contributions in many areas—from helping out newcomers, to designing graphics, to translating documentation.

To celebrate, we organized *Ten Years of Guix*[74], a three-day event that took place in Paris, France, in September 2022, with support from research and non-profit organizations[75]. About 50 people came in Paris and the event was also live-streamed.

This event was one of kind: it brought together scientists and free software hackers, two communities that evidently have shared values—as the *open science* movement demonstrates—and that benefit from one another. The program was organized as follows:

- Friday, September 16th, was dedicated to *reproducible deployment for reproducible research*. Scientists and practitioners shared their experience building reproducible research workflows, using Guix and other tools.

---

[71] *https://www.ibens.ens.fr/spip.php?article172&lang=en*

[72] *https://www.ibens.ens.fr/?lang=en*

[73] *https://archive.fosdem.org/2022/schedule/track/declarative_and_minimalistic_computing/*

[74] *https://10years.guix.gnu.org/*

[75] *https://10years.guix.gnu.org/sponsors*

- Saturday focused on development *with* Guix and *on* Guix, as well as community topics.
- Sunday had more in-depth presentations of Guix as well as informal discussions and skill-sharing sessions.

A total of 34 talks were given and videos are available on-line[76]—many thanks to the Debian video team for making it possible!

Oh and of course, we ate not one but two birthday cakes.

Training Sessions

For the French HPC Guix community, we continued the monthly on-line event called "Café Guix"[77], originally started in October 2021. Each month, a user or developer informally presents a Guix feature or workflow and answers questions. These sessions are now recorded and are available on the webpage.

A mini-tutorial about Guix was presented by Simon Tournier on May 19, 2022 during the French Higher Education and Research Days on Networking (JRES). The 1h video[78] and the support[79] are available (in French). In June, INRAE[80] (the French institute for research in agriculture, food, and environment) organized in Montpellier a training session covering tools such as Kubernetes and OpenStack, and hosted a session dedicated to computational reproducibility where Simon Tournier presented[81] how Guix can help.

---

[76] *https://10years.guix.gnu.org/program/*

[77] *https://hpc.guix.info/events/2022/café-guix/*

[78] *https://replay.jres.org/w/3TuYmocHwKtzs7q1VtL1GB*

[79] *https://conf-ng.jres.org/2021/document_revision_2595html?download*

[80] *https://www.inrae.fr/en*

[81] *https://gitlab.com/zimoun/fcc-inrae*

On May 30, 2022 the Max Delbrück Center for Molecular Medicine in the Helmholtz Association (MDC) hosted a Guix workshop as part of the Data Science Café in Berlin. The workshop was entitled "Managing reproducible and transparent software environments with GNU Guix" and was presented by Ricardo Wurmus.

The Inria research center in Nancy (France) periodically organizes afternoon technical seminars, referred to as "Tuto Techno", about a technology or programming language. On June 14, 2022 the research center hosted Marek Felšöci who gave a presentation[82] on the use of Guix combined with literate programming with Org Mode for building reproducible research studies. The presentation was followed by a hands-on session. Attendees were guided through the process of constructing a standalone Git repository containing a research study entirely reproducible thanks to Guix and the literate description of the experimental environment, source code and methods in Org mode. At the end of the hands-on session, participants learned how to use Software Heritage to guarantee a long-term availability of their work. The tutorial is self-contained and publicly available[83] for anyone who would like to try it out.

A training session was given during the Open Science Days[84], which took place in Grenoble, France, 13–15 December 2022. Entitled "*Déploiement reproductible des logiciels scientifiques avec GNU Guix*"[85] ("Reproducible scientific software deployment with GNU Guix") and given by Ludovic Courtès, Konrad Hinsen, and Simon Tournier, the session introduced the use of `guix shell` and `guix time-machine` as the building blocks of reproducible workflows. Training material is available on-line[86].

---

[82] *https://tuto-techno-guix-hpc.gitlabpages.inria.fr/slides/tuto-techno-guix-hpc.pdf*

[83] *https://tuto-techno-guix-hpc.gitlabpages.inria.fr/guidelines/*

[84] *https://osd-uga-2022.sciencesconf.org/*

[85] *https://archive.softwareheritage.org/swh:1:dir:731f3a71e4676c7ec0ab83c72aa060c9a094630a*

[86] *https://gitlab.inria.fr/guix-hpc/open-science-days-tutorial*

Another training session was organized by SARI (part of the DevLog knowledge network at CNRS) in Grenoble on the 8th of December 2022. It aimed to help people use Guix on the GriCAD HPC cluster.

Work has started on a sequel to the Reproducible Research MOOC[87] by Inria Learning Lab, which will include an introduction to Guix for managing software environments for reproducible research.

[87] *https://www.fun-mooc.fr/en/courses/reproducible-research-methodological-principles-transparent-scie/*

Personnel

GNU Guix is a collaborative effort, receiving contributions from more than 90 people every month—a 50% increase compared to last year. As part of Guix-HPC, participating institutions have dedicated work hours to the project, which we summarize here.

- Inria: 2.5 person-years (Ludovic Courtès; contributors to the Guix-HPC channel: Emmanuel Agullo, Luca Cirrottola, Marek Felšöci, Marc Fuentes, Nathalie Furmento, Gilles Marait, Florent Pruvost, Matthieu Simonin, Philippe Swartvagher, Mathieu Verite; system administrator in charge of Guix on the PlaFRIM and Grid'5000 clusters: Julien Lelaurain)
- Max Delbrück Center for Molecular Medicine in the Helmholtz Association (MDC): 2 person-years (Ricardo Wurmus and Mădălin Ionel Patrașcu)
- University of Paris Cité: 0.75 person-year (Simon Tournier)
- University of Tennessee Health Science Center (UTHSC): 3+ person-years (Efraim Flashner, Bonface Munyoki, Fred Muriithi, Arun Isaac, Jorge Gomez, Erik Garrison and Pjotr Prins)
- CNRS and UGA (GRICAD): 0.3 person-year (Céline Acary-Robert, Pierre-Antoine Bouttier, Oliver Henriot)

Perspectives

With UNESCO's Recommendation on Open Science[88] and the many Open Science initiatives at the national and institutional levels, awareness of the Open Science and reproducible research principles is on the rise. Its implications are also better understood in particular when it comes to software: software publication and licensing, issues of software deployment, provenance tracking, and reproducibility are becoming central to scientific practices. Addressing these issues requires commitment of the scientific community at large: scientists, but also research software engineers (RSEs) and system administrators.

The Guix-HPC effort is unique in its ability to connect these communities. This Activity Report as well as the program of the Ten Years of Guix event earlier this year are proof that researchers, engineers, and system administrators all have a stake in what we are building. Together, we shape tools and practices that further Open Science and make reproducible research workflows practical.

Bringing these tools and practices to the scientific community is a key challenge for the project. While Guix gets more recognition as an enabler for reproducible research, misconceptions persist: that Guix only caters to the needs of "reproducibility professionals"[89], or that it reproducibility is antithetical to performance[90]. In the coming year, we want to reach out to broader user communities—again scientists, engineers, and system administrators—and to provide training sessions. It is our mission to put the tools we build in the hands of practitioners at large.

There are technical challenges ahead for the coming year, in line with what we have been doing: improving the user experience for scientists, improving the user story when running software on a Guix-less cluster, bridging the gap with users that do not interact with software *via* the

---

[88] *https://en.unesco.org/science-sustainable-future/open-science/recommendation*

[89] *https://hpc.guix.info/blog/2022/07/is-reproducibility-practical/*

[90] *https://hal.inria.fr/hal-03604971/*

command line or Jupyter, bringing Guix System and `guix deploy` to HPC cluster administrators, and achieving 100% coverage of package source code in the Software Heritage archive.

The GNU Guix project turned ten this year. It started with the development of a "package manager" and is now providing a complete *deployment toolbox*: a package manager, but also a development environment manager, a container provisioning tool, a standalone operating system, and a cluster deployment tool. Besides its technical achievements, it has raised the bar of what one can expect in terms of software deployment—reproducibility, provenance tracking, and transparency. We are determined to make more strides in that direction.

There's a lot we can do and we'd love to hear your ideas[91]!

[91] *https://hpc.guix.info/about*

28.