Retour d'expérience sur l'utilisation de Guix dans le cadre d'un programme de recherche

Benjamin Arrondeau, Dylan Bissuel

7 novembre 2024

Who am I?

- I obtained a PhD in CFD (Computational Fluid Dynamics) less than a year ago

  so mainly a physics background with some basic programming (Python, Fortran90)

- After my PhD, I wanted to continue in the research world

  to be a support to the research but not leading it!

Who am I?

- I obtained a PhD in CFD (Computational Fluid Dynamics) less than a year ago

  so mainly a physics background with some basic programming (Python, Fortran90)

- After my PhD, I wanted to continue in the research world

  to be a support to the research but not leading it!

Currently, I am an **HPC support Engineer** at Gricad but mostly working for the PEPR DIADEM (DIAMOND project) for whom I **containerize/package** codes/workflows, **manage/set-up** the platform infrastructures and **support** the other engineers.

Table of content

Table of content

Guix context

GRICAD

- Guix available for users on head-nodes and compute-nodes since **2019**
- Custom channel for specific user requests since **2020**
- General documentation
- Support to users

Guix context

GRICAD

- Guix available for users on head-nodes and compute-nodes since **2019**
- Custom channel for specific user requests since **2020**
- General documentation
- Support to users

ME

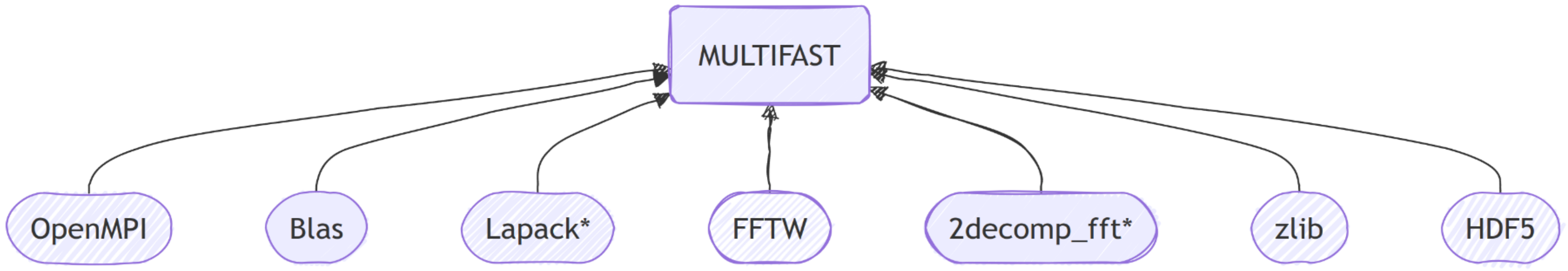
- Guix was completely *unknown*
- First training the 7 of March by @PAB
- Learning approach:
  - Package a known-code
  - Benchmark the perfs (to be fully convinced)

Table of content

First steps: Test case

> *Try to package the code I used during my thesis:*

- common dependencies so little needs to package extra-stuff
- common build chain so little needs to change phases



First steps: Test case

```
(define-public multifast
  (package
  (name "multifast")
  (version "1")
    (source ... )
    (propagated-inputs ... )
    (build-system gnu-build-system)
    (arguments '(#:phases (modify-phases %standard-phases
                   (add-after 'unpack 'fix_binsh ... )
                   (replace 'configure ... )
                   (replace 'install ... )
                   ...
                   )))
    (home-page "https://github.com/Benji12358/multifast")
    (synopsis "MULTIFAST synopsis")
    (description "MULTIFAST description")
    (license license:gpl3+)))
```

First steps: Performance benchmark

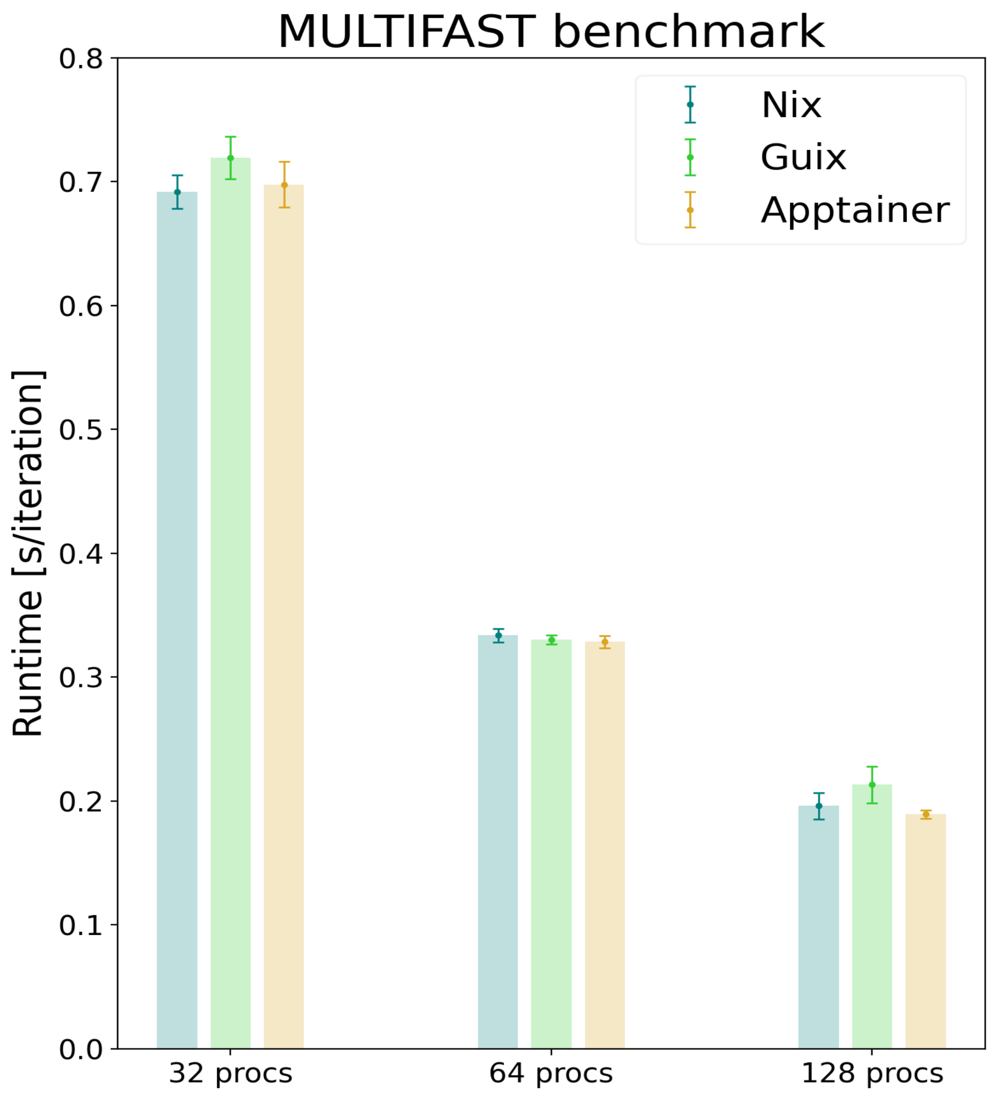
> *The main idea was to compare performance between Nix environment, Apptainer image and Guix package*

- 1000 iterations of a turbulent channel flow with MULTIFAST
- Tests on 32 procs (1 node) and 64 and 128 procs (2 nodes)
- 12 runs on Dahu cluster (Gricad) for each case
- Same nodes (Gold Processors 6130, 32 cores per node, 192 Go RAM)

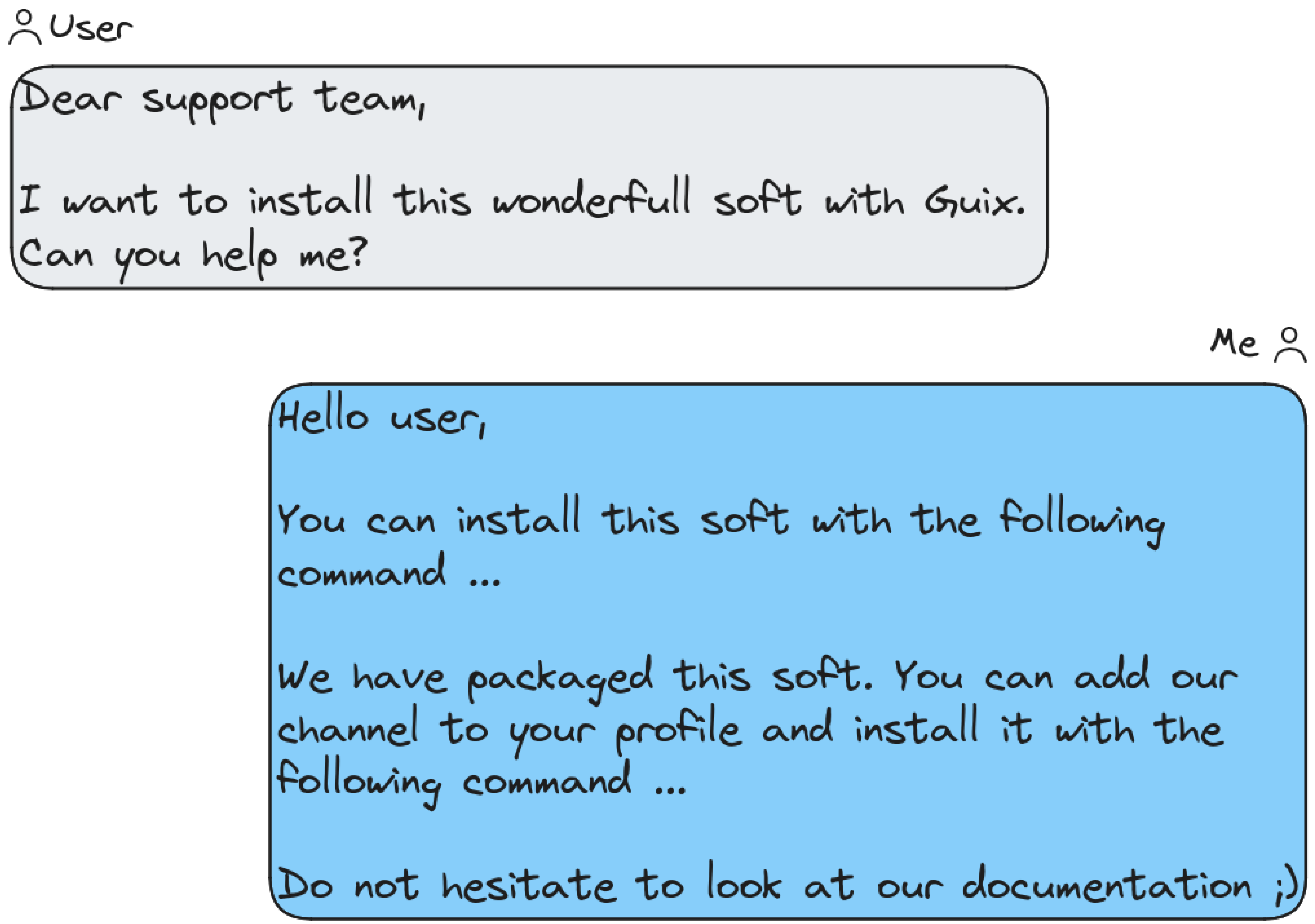
First steps: Performance benchmark

> *The main idea was to compare performance between Nix environment, Apptainer image and Guix package*

- 1000 iterations of a turbulent channel flow with MULTIFAST
- Tests on 32 procs (1 node) and 64 and 128 procs (2 nodes)
- 12 runs on Dahu cluster (Gricad) for each case
- Same nodes (Gold Processors 6130, 32 cores per node, 192 Go RAM)



First steps: User support



User

Dear support team,

I want to install this wonderfull soft with Guix.
Can you help me?

Me

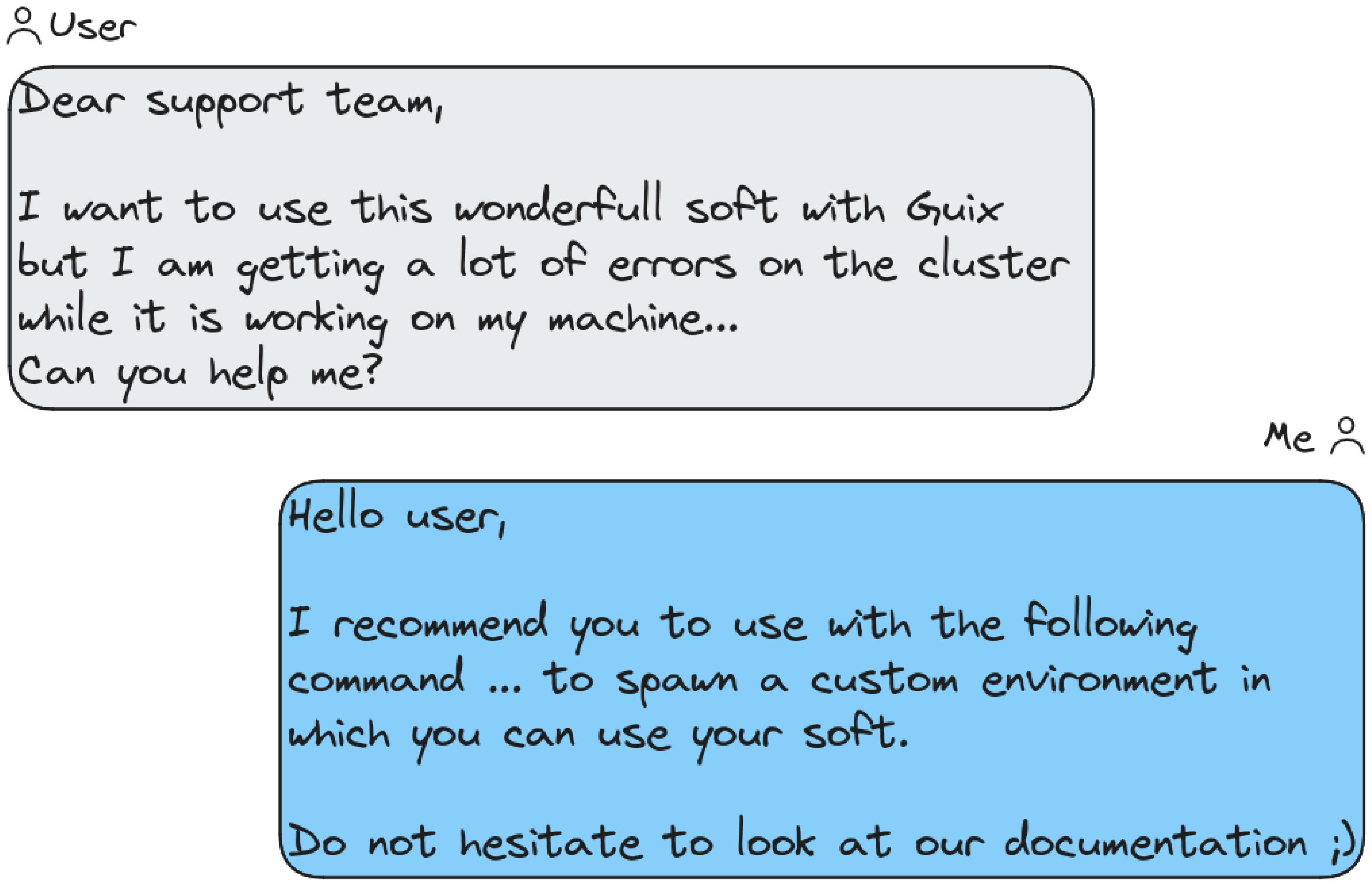
Hello user,

You can install this soft with the following command ...

We have packaged this soft. You can add our channel to your profile and install it with the following command ...

Do not hesitate to look at our documentation ;)

First steps: User support



User

Dear support team,

I want to use this wonderfull soft with Guix but I am getting a lot of errors on the cluster while it is working on my machine...
Can you help me?

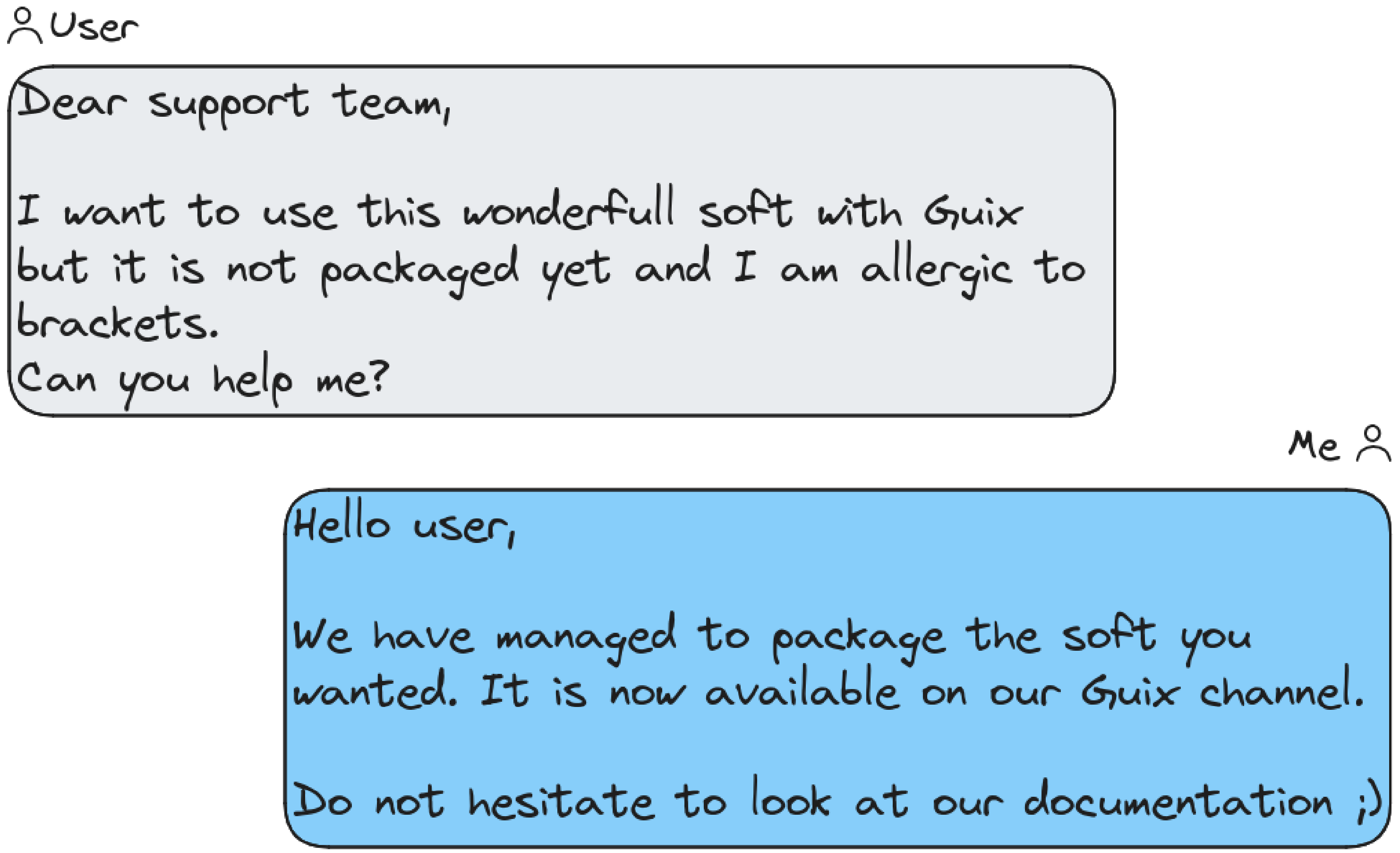
Me

Hello user,

I recommend you to use with the following command ... to spawn a custom environment in which you can use your soft.

Do not hesitate to look at our documentation ;)

First steps: User support



User

Dear support team,

I want to use this wonderfull soft with Guix but it is not packaged yet and I am allergic to brackets.
Can you help me?

Me

Hello user,

We have managed to package the soft you wanted. It is now available on our Guix channel.

Do not hesitate to look at our documentation ;)

Table of content

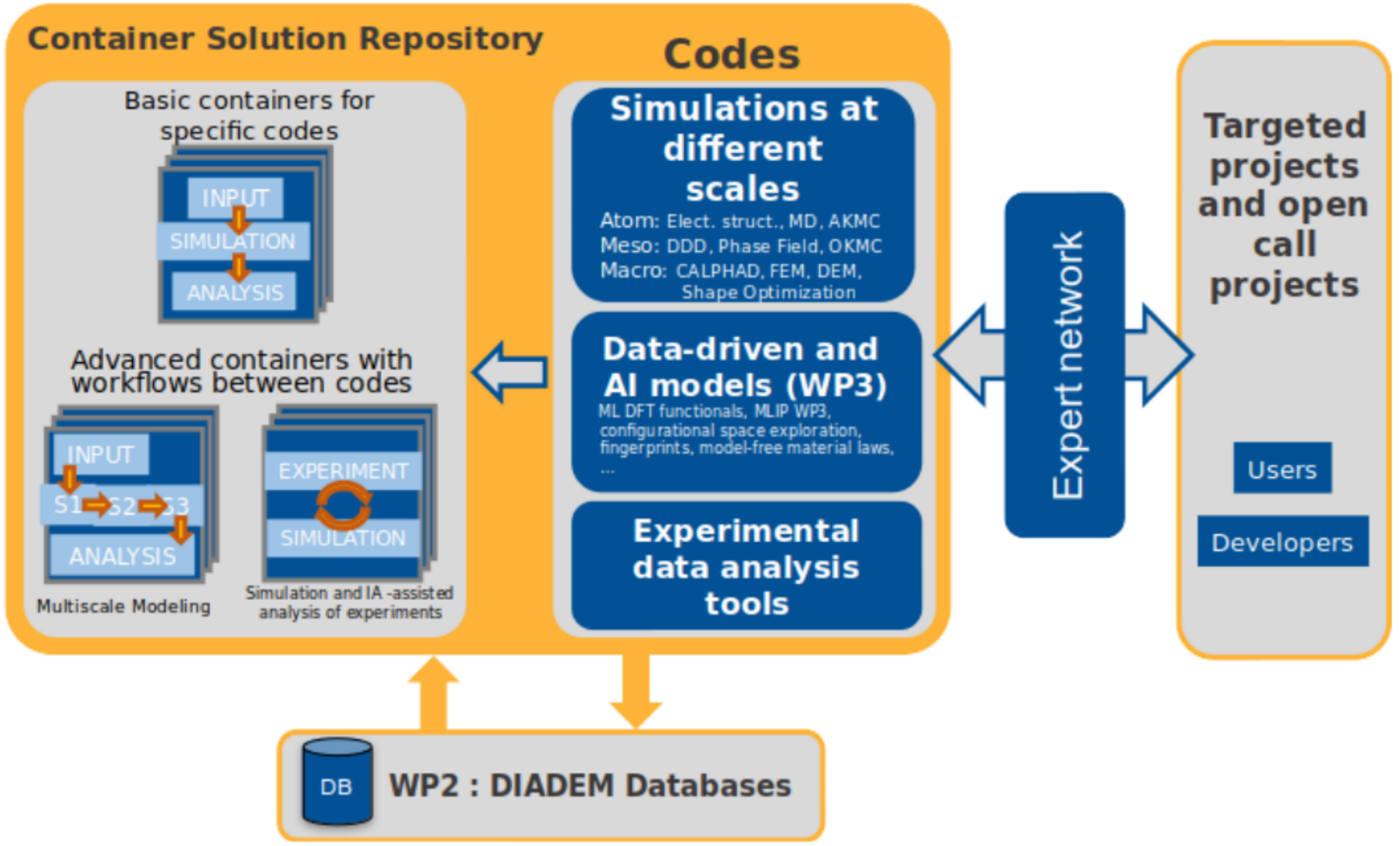
Application to DIAMOND project (PEPR DIADEM)

*DIADEM stands for **DI**scovery **A**cceleration for the **D**eployment of **E**merging **M**aterials*

*DIAMOND stands for **D**ata management and **I**nfrastructures for **A**I, **M**odelling, **O**ptimization and **N**umerical **D**esign*

- **Containers:** making it easier to install, distribute and use code
- Container **repository**
- **Workflows:** enable code chaining, automation and therefore high throughput, including analysis of experimental data

Meso-center: GRICAD (Grenoble)


Container Solution Repository
Basic containers for specific codes
INPUT
SIMULATION
ANALYSIS
Advanced containers with workflows between codes
INPUT
S1
S2
S3
ANALYSIS
Multiscale Modeling
EXPERIMENT
SIMULATION
Simulation and IA-assisted analysis of experiments
Codes
Simulations at different scales
Atom: Elect. struct., MD, AKMC
Meso: DDD, Phase Field, OKMC
Macro: CALPHAD, FEM, DEM, Shape Optimization
Data-driven and AI models (WP3)
ML DFT functionals, MLIP WP3, configurational space exploration, fingerprints, model-free material laws, ...
Experimental data analysis tools
Expert network
Targeted projects and open call projects
Users
Developers
DB
WP2 : DIADEM Databases

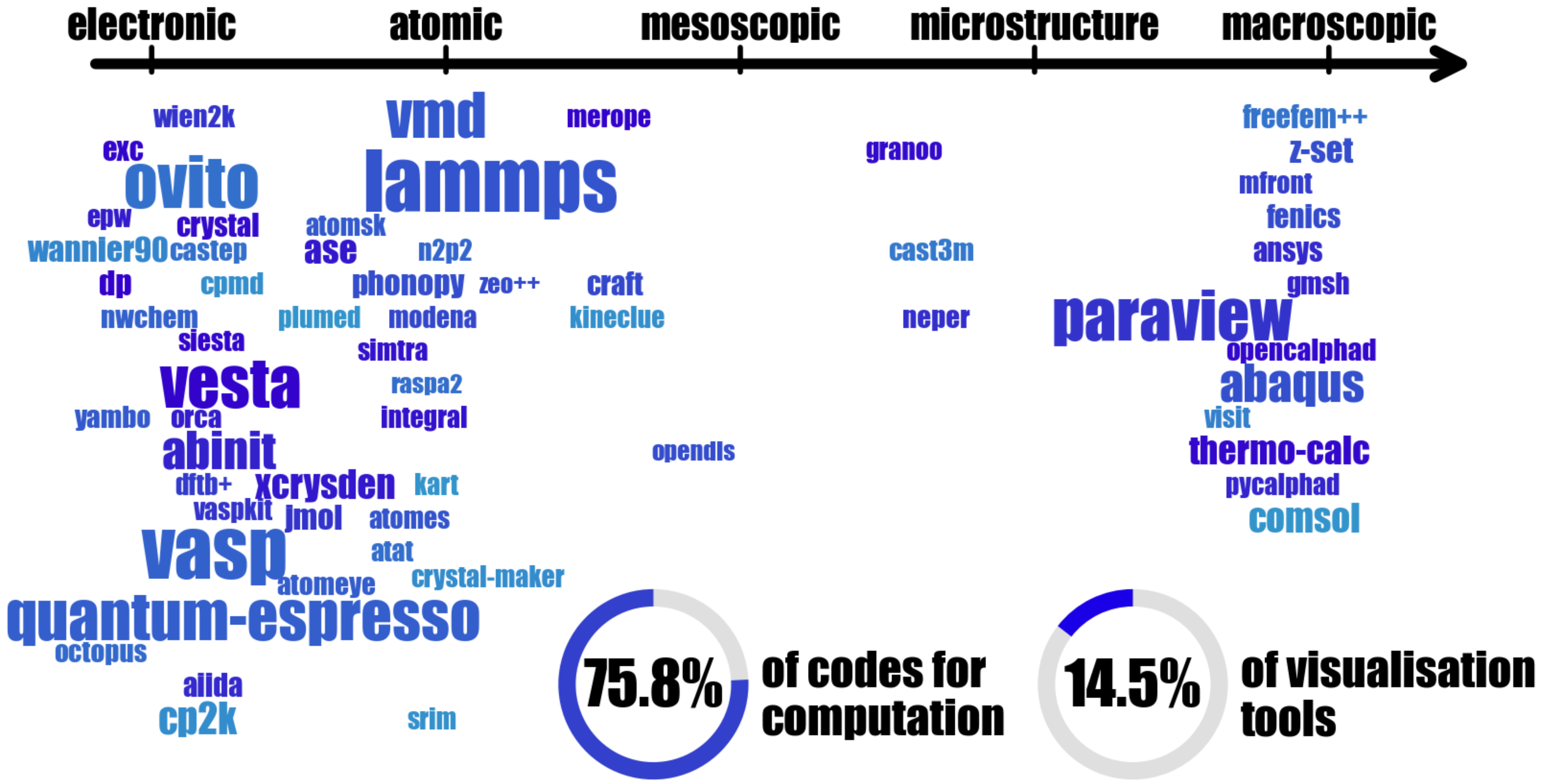
Application to DIAMOND project

In brief:

- PEPR DIADEM aims to accelerate the **development** and **production** of more efficient and sustainable materials
- WP1 of DIAMOND (one project of DIADEM) aims to set-up a **numerical infrastructure** for this instance
- This infrastructure should provide **codes** (for simulation or visualisation), **workflows** (to automate series of calculation) and access to a **database** (grouping both experimental and simulation data)

Application to DIAMOND project

Results from a survey conducted by @BissuelD during the summer 2023



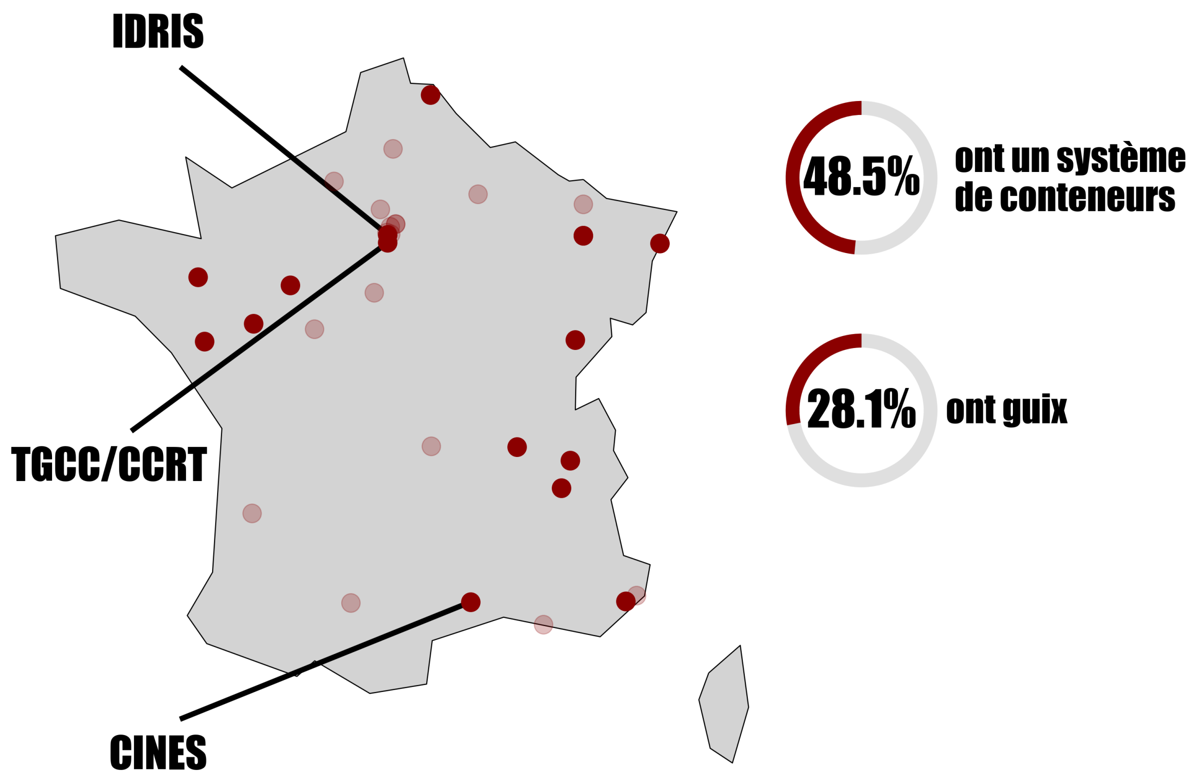
Application to DIAMOND project

Apptainer as containerization solution

> *to embed a code and its dependencies in lightweight images*

Guix as packaging solution

> *to automate install and update of a code and its dependencies*
> *... and for software reproducibility*



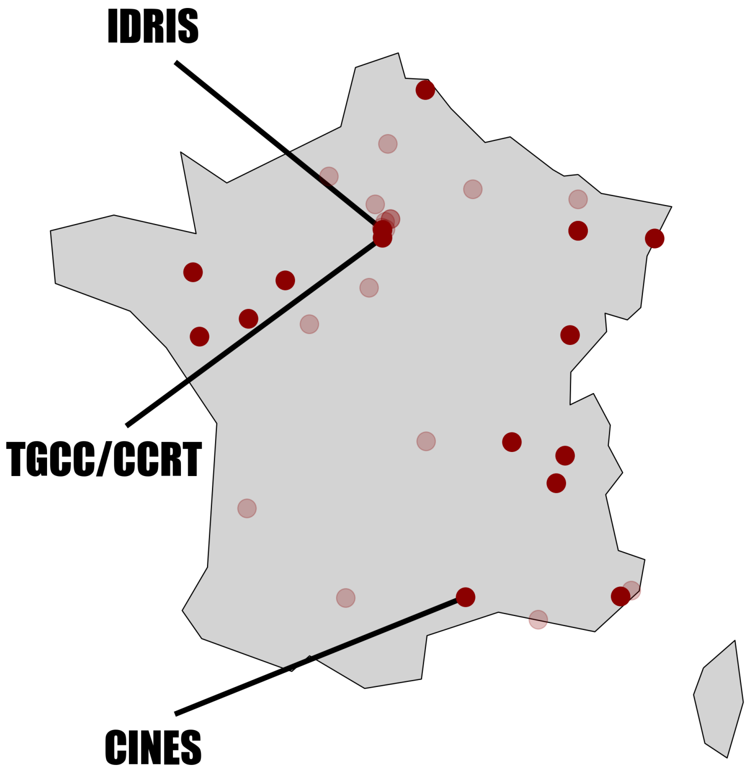
Application to DIAMOND project

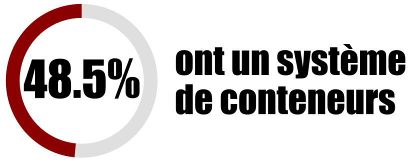
Apptainer as containerization solution

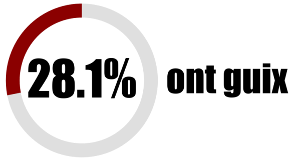
> *to embed a code and its dependencies in lightweight images*

Guix as packaging solution

> *to automate install and update of a code and its dependencies*
> *... and for software reproducibility*

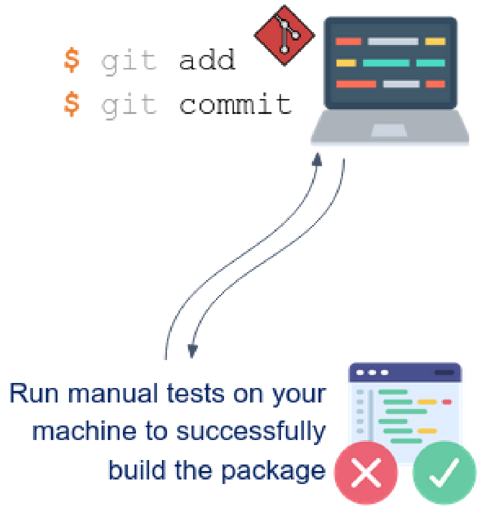




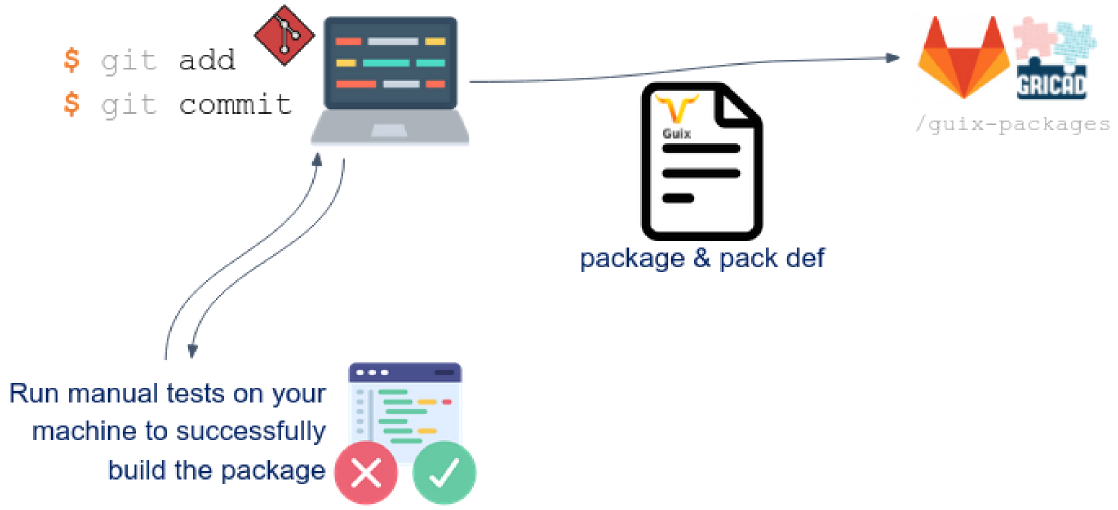


How can we combine **Apptainer** images and **Guix** packages?

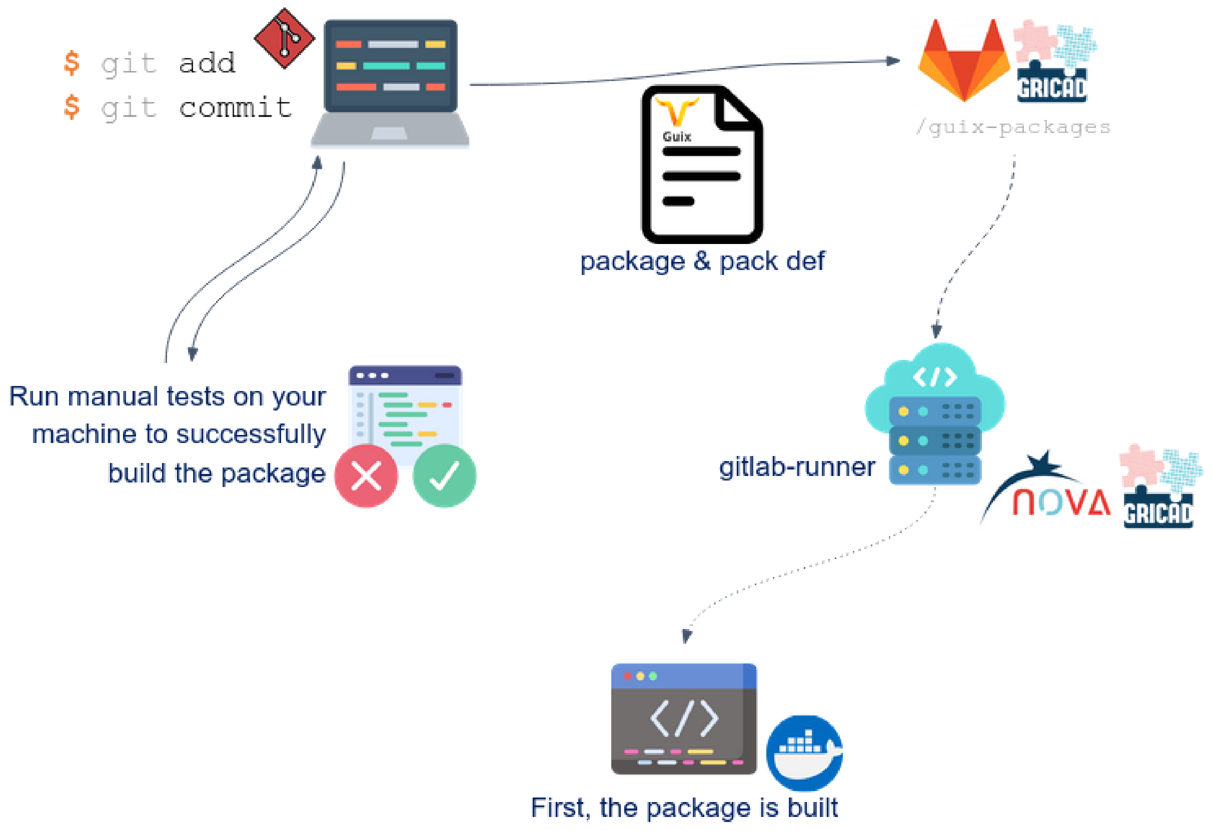
Application to DIAMOND project: CI/CD pipeline


$ git add
$ git commit
Run manual tests on your machine to successfully build the package

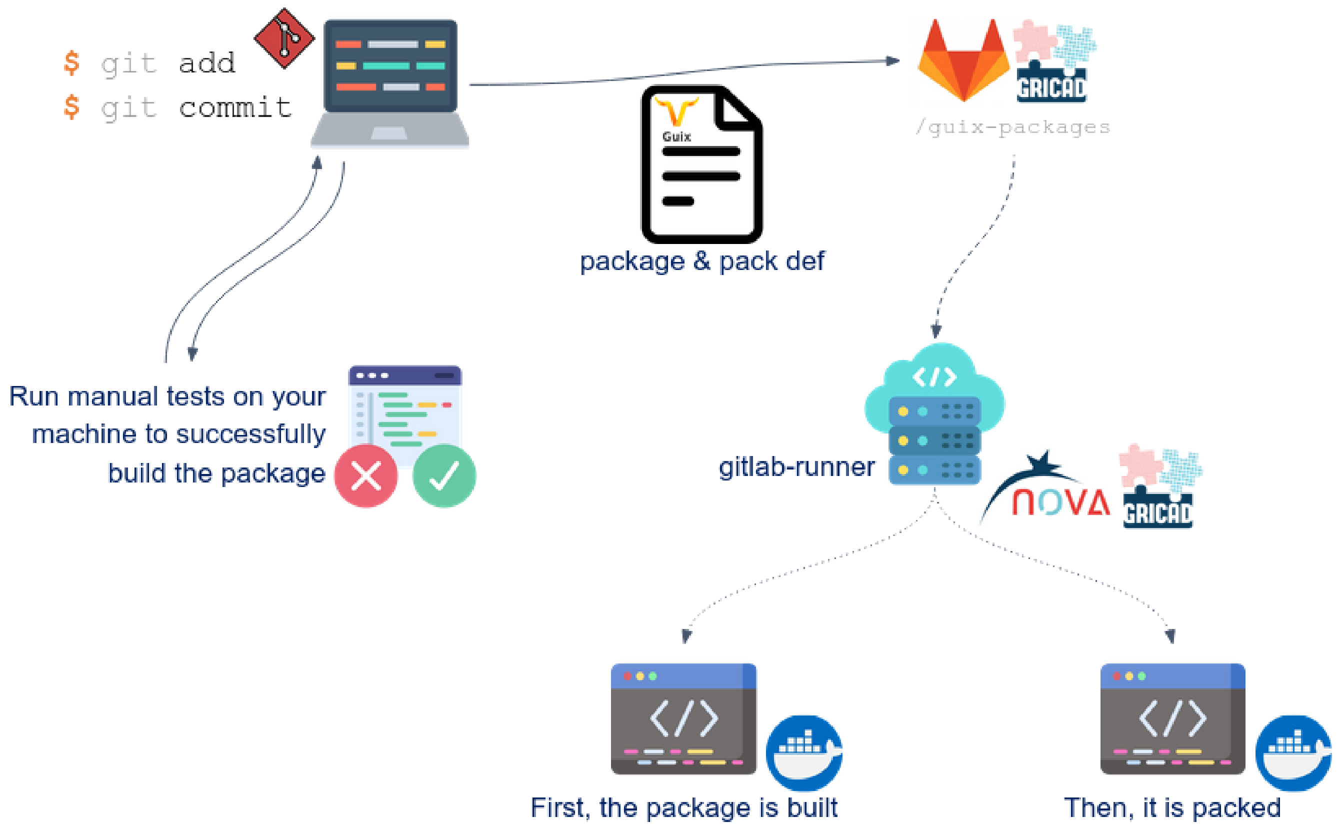
Application to DIAMOND project: CI/CD pipeline


$ git add
$ git commit
Guix
package & pack def
GRICAD
/guix-packages
Run manual tests on your machine to successfully build the package

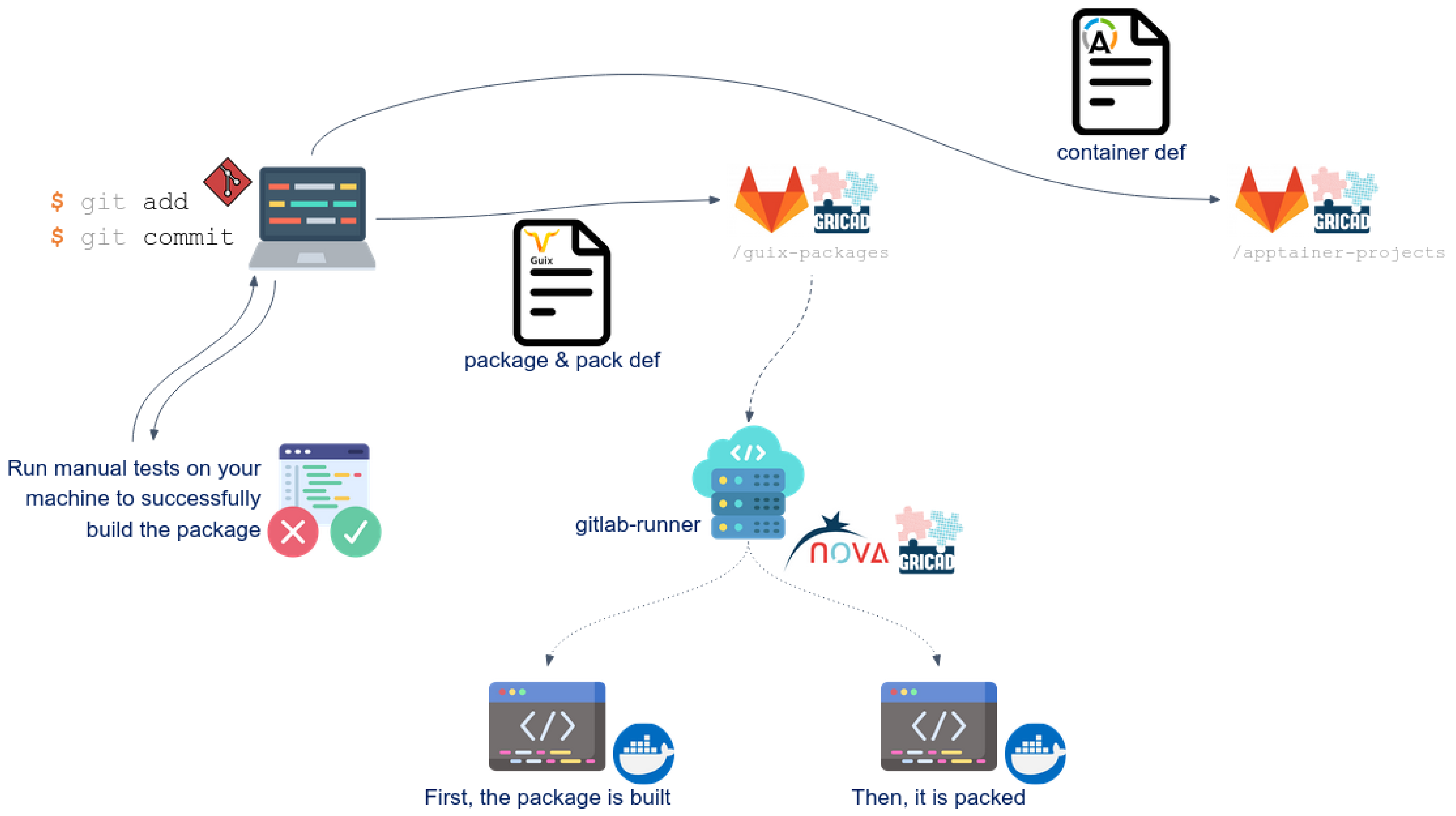
Application to DIAMOND project: CI/CD pipeline


$ git add
$ git commit
Guix
package & pack def
GRICAD
/guix-packages
Run manual tests on your machine to successfully build the package
gitlab-runner
NOVA
GRICAD
First, the package is built

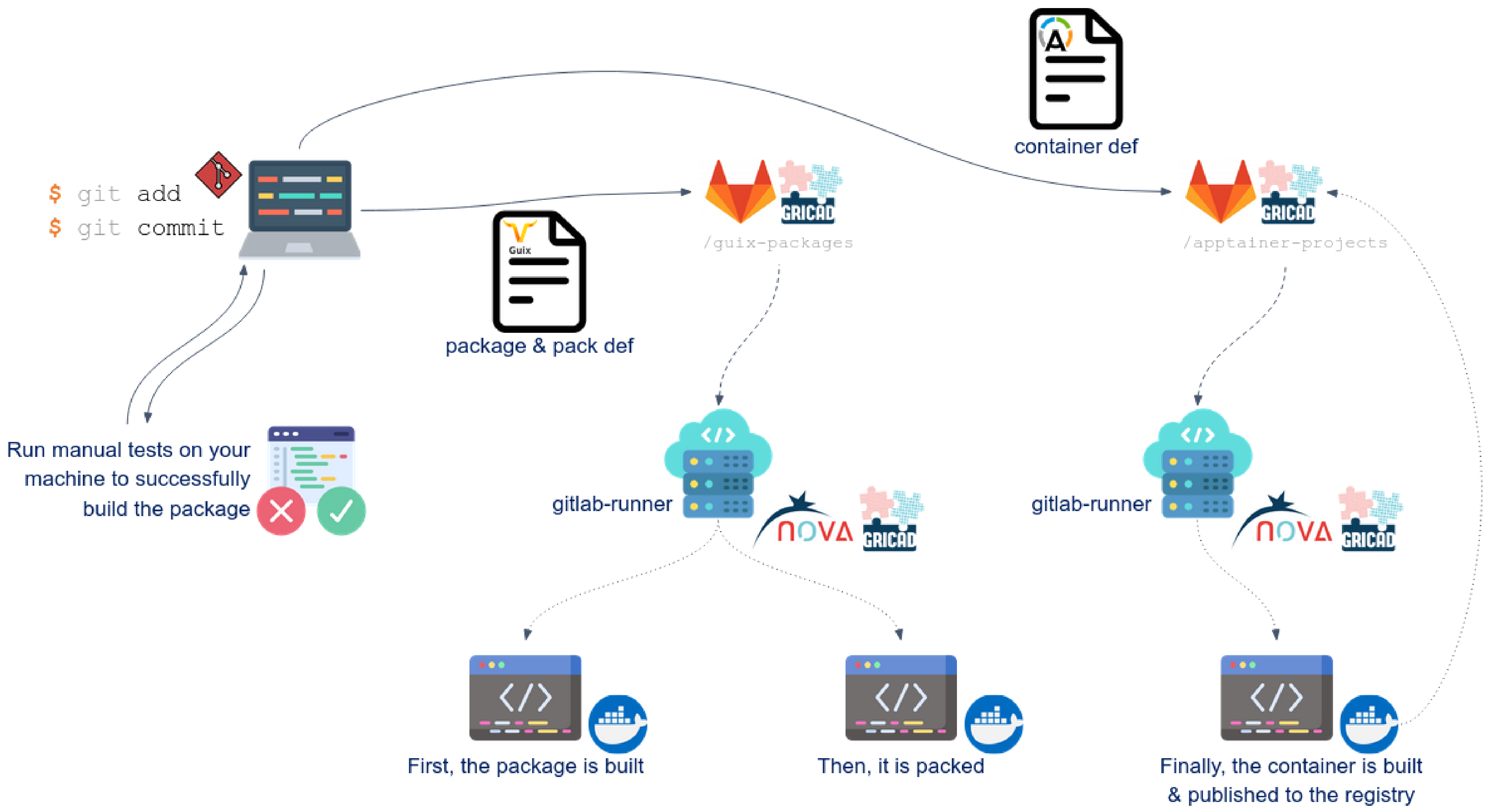
Application to DIAMOND project: CI/CD pipeline



Application to DIAMOND project: CI/CD pipeline



Application to DIAMOND project: CI/CD pipeline


$ git add
$ git commit
container def
package & pack def
Guix
GRICAD
/guix-packages
/apptainer-projects
Run manual tests on your machine to successfully build the package
gitlab-runner
NOVA
gitlab-runner
First, the package is built
Then, it is packed
Finally, the container is built & published to the registry

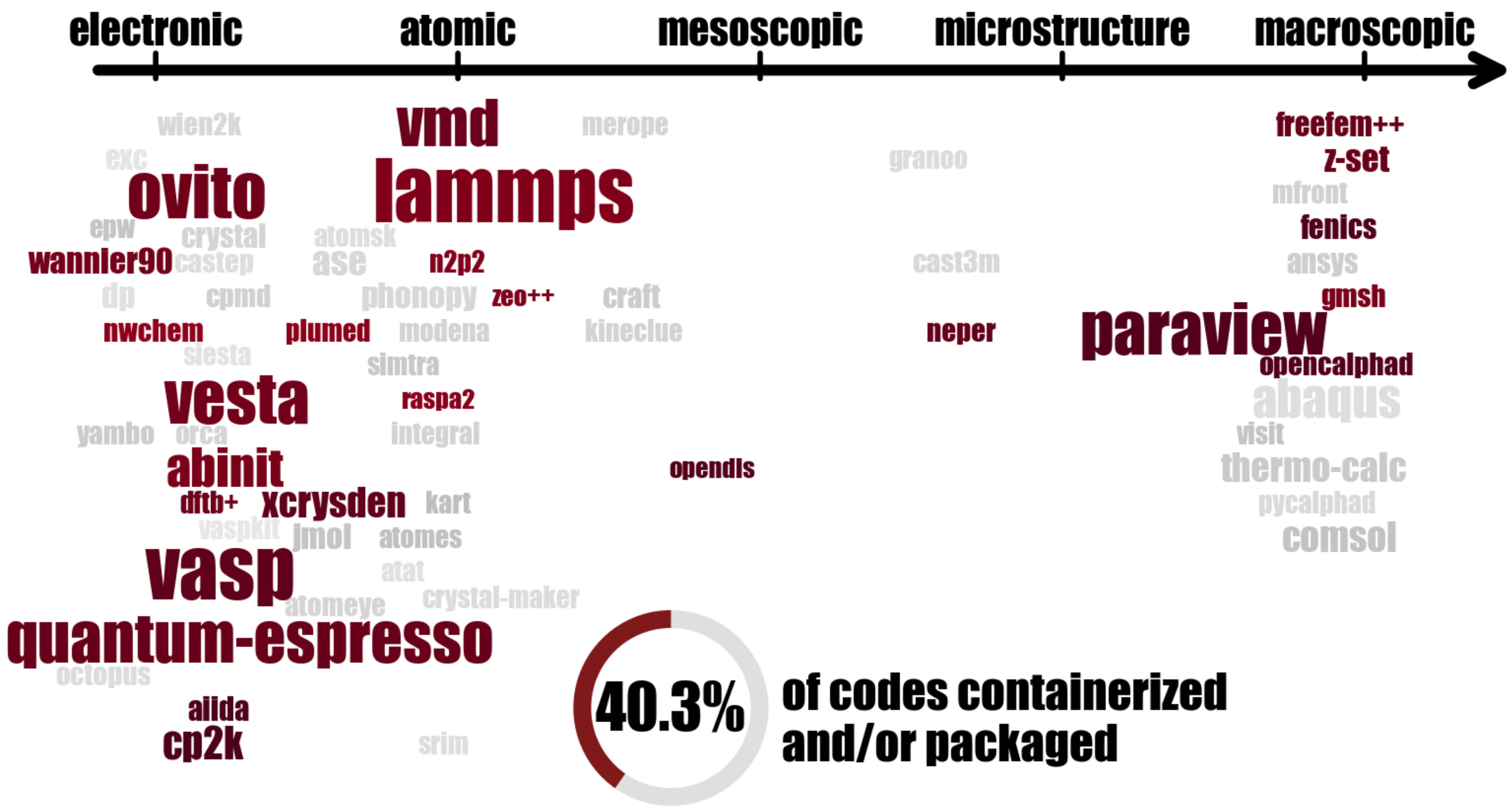
Application to DIAMOND project: CI/CD pipeline

```
# For gitlab-ci : regex=lammps-openmpi-openssh
Bootstrap: localimage
From: /gnu/store/5w1mqjgia33yy3qy9wjwvi37hvdbfzzl-lammps-plumed-bash-minimal-squashfs-pack.gz.squashfs

%files
    empty_file /etc/passwd # to remove warning about /etc/passwd
    empty_file /etc/group  # to remove warning about /etc/group

%post
    profile_path=$(ls /gnu/store | grep profile)
    mkdir -p /usr/share/lammps/potentials
    cp /gnu/store/$profile_path/share/lammps/potentials/* /usr/share/lammps/potentials

%environment
    export LAMMPS_POTENTIALS=/usr/share/lammps/potentials

%help
    This container embedds LAMMPS (2 August 2023 stable version, update 2) with OpenMPI support.
    For more information about this image, please run "apptainer inspect <this-image>"
    ...

%labels
    Owner Sandia Corporation
    Author dylan.bissuel@univ-lyon1.fr
    Label LAMMPS_stable_2Aug2023_update2
    EntryPoint https://www.lammps.org/
```

Table of content

What is the progress so far?

Guix packages available 38 | Guix squashfs images available 21 | Apptainer images available 23 | Apptainer WM images available 6



What is the progress so far?

In practice:

- A channel was created to store all the **Guix** packages
- All **Apptainer** images are stored on the Gitlab registry
- Some codes have not been packaged with Guix yet

AiiDA

- Some codes are proprietary licence

VASP, ovito, z-set

- The hell of dependencies ...

23 packages definition for cp2k

What is the progress so far?

```
(define-public costa
  (package
    (name "costa")
    (version "2.2.2")
    (source
      (origin
        (method git-fetch)
        (uri (git-reference
              (url "https://github.com/eth-cscs/COSTA")
              (commit "bb84528d023db9a6b00ad729fb44b8c3cef8c981")))
        (file-name (git-file-name name version))
        (sha256 (base32 "026svdxdihh1zrm4ydwpq417f4y69d8l1v72kj22phmvf1jk484f"))))
    (build-system cmake-build-system)
    (propagated-inputs
     (list openmpi
           gfortran
           scalapack))
    (arguments
        '(#:phases
          (modify-phases %standard-phases
              ;; Remove check phase
              (delete 'check)
        )))
    (synopsis synopsis)
    (description description)
    (home-page "https://github.com/eth-cscs/COSTA")
    (license license:bsd-3)))
```

What is the progress so far?

```
(define-public quip
  (package
    (name "quip")
    (version "v0.9.14")
    (source
      (origin
        (method git-fetch)
        (uri (git-reference
              (url "https://github.com/libAtoms/QUIP")
              (commit "72aaf3fbc9403fe22361bf3fdb4295c516dd1094")
              (recursive? #t)))
        (file-name (git-file-name name version))
        (sha256 (base32 "17nask73vj0xy797pjz8h8w6frq14czysmcxjni3zvh7gx26zjx3"))))
    (build-system gnu-build-system)
    (propagated-inputs
        (list gfortran
              openblas-openmp
              lapack
              openmpi
              python-sans-pip
              python2-minimal
              perl
              tcsh))
    (arguments
        '(;; Disable parallel build
          #:parallel-build? #f
          #:phases
              (modify-phases %standard-phases
                  ;; Remove check phase
                  (delete 'check)
                  ;; Add a phase to fix bash interpreter
                  (add-after 'unpack 'fix_interpreters
                      (lambda* (#:key inputs #:allow-other-keys)
                          (for-each (lambda (f)
                              (invoke "sed" "-i" (string-append "s@/bin/bash@" (which "bash") "@g") f))
                                  (find-files "."))))
                  ;; Add a phase to prepare the Makefile used during the build
                  (add-before 'configure 'prepare_makefile
                      (lambda _
                          (invoke "sed" "-i" "s@QUIPPY_FCOMPILER = gnu95@QUIPPY_FCOMPILER = gfortran@g" "arch/Makefile.linux_x86_64_gfortran")))
                  ;; Replace the configure phase
                  ;; as the user behavior (pressing enter) cannot mimick, the default options are forced
                  ;; to do so, the output file is hard written
                  ;; this file was found by reversing the source code ...
                  (replace 'configure
                      (lambda* (#:key inputs outputs #:allow-other-keys)
                          (setenv "QUIP_ARCH" "linux_x86_64_gfortran_openmp")
                          (setenv "QUIP_ROOT" (getcwd))
                          (setenv "Makefile_dir" (string-append "build/" (getenv "QUIP_ARCH")))
                          (setenv "Makefile_file" (string-append (getenv "Makefile_dir") "/Makefile.inc"))
                          (mkdir-p (getenv "Makefile_dir"))
                          (call-with-output-file (getenv "Makefile_file")
                              (lambda (port)
                                  (format port "# Place to override setting elsewhere, in particular things set in ~a
# look in ~a for defaults set by arch
#
# F77=gfortran
# F90=gfortran
# F95=gfortran
# CC=gcc
# CPLUSPLUS=g++
# FPP=gfortran -E -x f95-cpp-input
# LINKER=gfortran
# LIBTOOL=
# OPTIM=
# COPTIM=
# QUIPPY_INSTALL_OPTS=
# DEBUG=-O0 -g -DDUMP_CORE_ON_ABORT -DDEBUG -fbounds-check
# DEBUG=
# CDEBUG=
MATH_LINKOPTS=~a
PYTHON=~a
PIP=~a
EXTRA_LINKOPTS=
HAVE_CP2K=0
HAVE_VASP=0
HAVE_TB=0
HAVE_PRECON=1
HAVE_ONIOM=0
HAVE_LOCAL_E_MIX=0
HAVE_QC=0
HAVE_GAP=0
HAVE_DESCRIPTORS_NONCOMMERCIAL=0
HAVE_QR=1
HAVE_SCALAPACK=0
HAVE_THIRDPARTY=0
HAVE_FX=0
HAVE_SCME=0
HAVE_MTP=0
HAVE_MBD=0
HAVE_TTM_NF=0
HAVE_CH4=0
HAVE_NETCDF4=0
HAVE_MDCORE=0
HAVE_ASAP=0
HAVE_KIM=0
HAVE_CGAL=0
HAVE_METIS=0
HAVE_LMTO_TBE=0
SIZEOF_FORTRAN_T=2
```

```
"
                                (getenv "QUIP_ARCH")
                                (string-append (getcwd) "arch/Makefile." (getenv "QUIP_ARCH"))
                                "-llapack -lblas"
                                (string-append (assoc-ref %build-inputs "python2-minimal") "/bin/python")
                                (string-append (assoc-ref %build-inputs "python2-minimal") "/bin/pip"))))
                        ;; there is still an option to be added to this file: the size_t_fortran
                        ;; to add it, the config command of the source code is run after commented all the lines of the Makefile.config
                        (invoke "sed" "-i" "1,31 s/^/#/" "Makefile.config")
                        (invoke "sed" "-i" "35,474 s/^/#/" "Makefile.config")
                        (invoke "make" "config")))
                   ;; Add a phase to create a new Makefile for the build
                   (add-before 'build 'patch_makefile
                      (lambda _
                        (copy-file (string-append "arch/Makefile." (getenv "QUIP_ARCH")) (string-append "arch/Makefile." (getenv "QUIP_ARCH") "_forbuild"))
                        (invoke "sed" "-i" "s@include arch/Makefile.linux_x86_64_gfortran@include arch/Makefile.linux_x86_64_gfortran@g" (string-append "arch/Makefile." (getenv "QUIP_ARCH") "_forbuild"))
                      ))
                   ;; Add a phase to copy the missing files for the build
                   (add-after 'patch_makefile 'copy_missing_files
                      (lambda _
                        (copy-file "Makefile.rules" "src/fox/Makefile.rules")
                        (copy-file "Makefile.rules" (string-append "build/" (getenv "QUIP_ARCH") "/Makefile.rules"))
                        (copy-file (string-append "arch/Makefile." (getenv "QUIP_ARCH") "_forbuild") (string-append "build/" (getenv "QUIP_ARCH") "/Makefile." (getenv "QUIP_ARCH")))
                        (copy-file (string-append "arch/Makefile." (getenv "QUIP_ARCH") "_forbuild") (string-append "src/fox/Makefile." (getenv "QUIP_ARCH")))
                        (copy-file (string-append "build/" (getenv "QUIP_ARCH") "/Makefile.inc") "src/fox/Makefile.inc")
                      ))
                   ;; Add a phase to patch shell interperter of fox configure
                   (add-after 'copy_missing_files 'fix_fox_build
                      (lambda _
                        (invoke "sed" "-i" (string-append "s@/bin/sh@" (which "sh") "@g") "src/fox/configure")
                      ))
                   ;; Add a phase to patch the path of the source code in the Makefile
                   (add-after 'fix_fox_build 'fix_makefile
                      (lambda _
                        (invoke "sed" "-i" "s#${PWD}/src#${PWD}/source/src#g" "Makefile")
                      ))
                   ;; Replace the install phase
                   (replace 'install
                      (lambda _
                          (let* ((out (getenv "out"))
                                 (incdir (string-append out "/include"))
                                 (libdir (string-append out "/lib")))
                              (mkdir-p incdir)
                              (mkdir-p libdir)
                              ;; install .mod file
                              (install-file (string-append "build/" (getenv "QUIP_ARCH") "/quip_unified_wrapper_module.mod")  incdir)
                              ;; install .a file from quip ...
                              (for-each (lambda (f) (install-file f libdir))
                                      (find-files (string-append "build/" (getenv "QUIP_ARCH")) (lambda (file stat) (ar-file? file))))
                              ;; ... and from fox
                              (for-each (lambda (f) (install-file f libdir))
                                      (find-files (string-append "src/fox/objs." (getenv "QUIP_ARCH") "/lib") (lambda (file stat) (ar-file? file))))
                   )))
                   ; ;; Add a phase to install sources
                   ; (add-after 'install 'install-src
                   ;    (lambda _
                   ;        (let* ((out (getenv "out"))
                   ;               (srcdir (string-append out "/src")))
                   ;            (mkdir-p srcdir)
                   ;            (copy-recursively "src" srcdir))
                   ; ))
                   ;; Add a phase to build and install libquip.a
                   (add-after 'install 'build_install_libquipa
                      (lambda _
                          (setenv "QUIP_INSTALLDIR" (getenv "out"))
                          (invoke "make" "install")
                          (invoke "make" "libquip")
                          (install-file (string-append "build/" (getenv "QUIP_ARCH") "/libquip.a") (string-append (getenv "out") "/lib"))))
          )))
    (synopsis "libAtoms/QUIP molecular dynamics framework")
    (description "The QUIP package is a collection of software tools to carry out molecular dynamics simulations.
      It implements a variety of interatomic potentials and tight binding quantum mechanics, and is also able to
      call external packages, and serve as plugins to other software such as LAMMPS, CP2K and also the python
      framework ASE. Various hybrid combinations are also supported in the style of QM/MM, with a particular focus
      on materials systems such as metals and semiconductors.")
    (home-page "https://libatoms.github.io")
    (license license:gpl3+)))
```

Table of content

What is next?

- Design a solution for the workflows

  more or less identical to that of codes

- Deploy the cuirass service

  useful to come back to the last working commit

- Continue packaging ...

  all codes in codecloud and others ... and provide GPU support

- Learn more about Guix system

  for system reproducibility ...

Acknowledgement

the dev team:

- Dylan Bissuel

  has packaged/containerized most of the codes/workflows

- David Martin-Calle

  has designed and deployed the DIAMOND website

- Arthur Hardiagon, João Paulo Mendonça, Jonathan Daubin

  have developed workflows and professionalised codes

- Cinthya Herrera, Akshay Krishna

  for the different discussions across DIAMOND WPs

Thanks for your attention!

Any questions?