LIRIS

UMR 5205 CNRS

Journée Reproductibilité en Sciences

Eric Boix 4 Avril 2024

cnrs INSA Lyon 1 UNIVERSITÉ LUMIÈRE LYON 2 ÉCOLE CENTRALE LYON

Conference: reproducibility guidelines

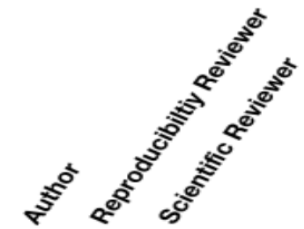
Website: https://osf.io/phmce/
Version: December 2020
DOI: 10.17605/OSF.IO/CB7Z8

AGILE Association of Geographic Information Laboratories in Europe

REPRODUCIBLE PAPER GUIDELINES

Full and short papers submitted to the AGILE conference **have** to include a **Data and Software Availability** section which documents data, software, and computational infrastructure to support reproduction, or mentions reasons for not publishing them.

The above requirement is the only one to comply with the AGILE Reproducible Paper Guidelines. The remainder of the document provides concrete recommendations for all involved stakeholders to increase transparency, reproducibility, and openness of computational GIScience research. The following table of contents shows the recommended parts for different readers. Familiarity with all sections is, of course, beneficial.



Conference: reproducibility guidelines

INCLUDING COMPUTATIONAL WORKFLOWS IN RESEARCH PAPERS

	Minimum requirements	Recommended practices
What? Computational environment	• Describe the used environment and computational infrastructure, e.g., hardware specs, operating system • List software versions • Cite used software[14]	• Provide the actual environment, e.g., a Dockerfile + container[15] or a Virtual Machine (e.g., using OSGeo-Live) • Provide a pinned freeze of your dependencies (structured configuration files with dependency information) • Add a colophon or "reproducibility receipt"[16] to your notebooks • Installation and execution instructions for different operating systems
Computation steps	• Document the detailed steps in a text file and/or flowchart (every action/click) • Document expected execution times given computing power unless negligible • Ask a colleague to try out the instructions	• Scripts/models and a README file that explains their use • All figures are fully scripted and a peer has read your README's instructions (incl. interactive visualisations and interactive adjustments • Multi-panel plots are composited with scripts[17] • Software package with structured metadata[18], tests/CI[19], and a pipeline framework[20] or workflow language[21] • Live documents for analyses, e.g., Binder[22] • Live demo of APIs/online applications (e.g., anonymous cloud resources, such as Google Cloud Run or AWS) • Subset or a synthetic dataset for quick evaluation

Conference: reproducibility guidelines

Versioned code repository, such as GitHub or GitLab [...]

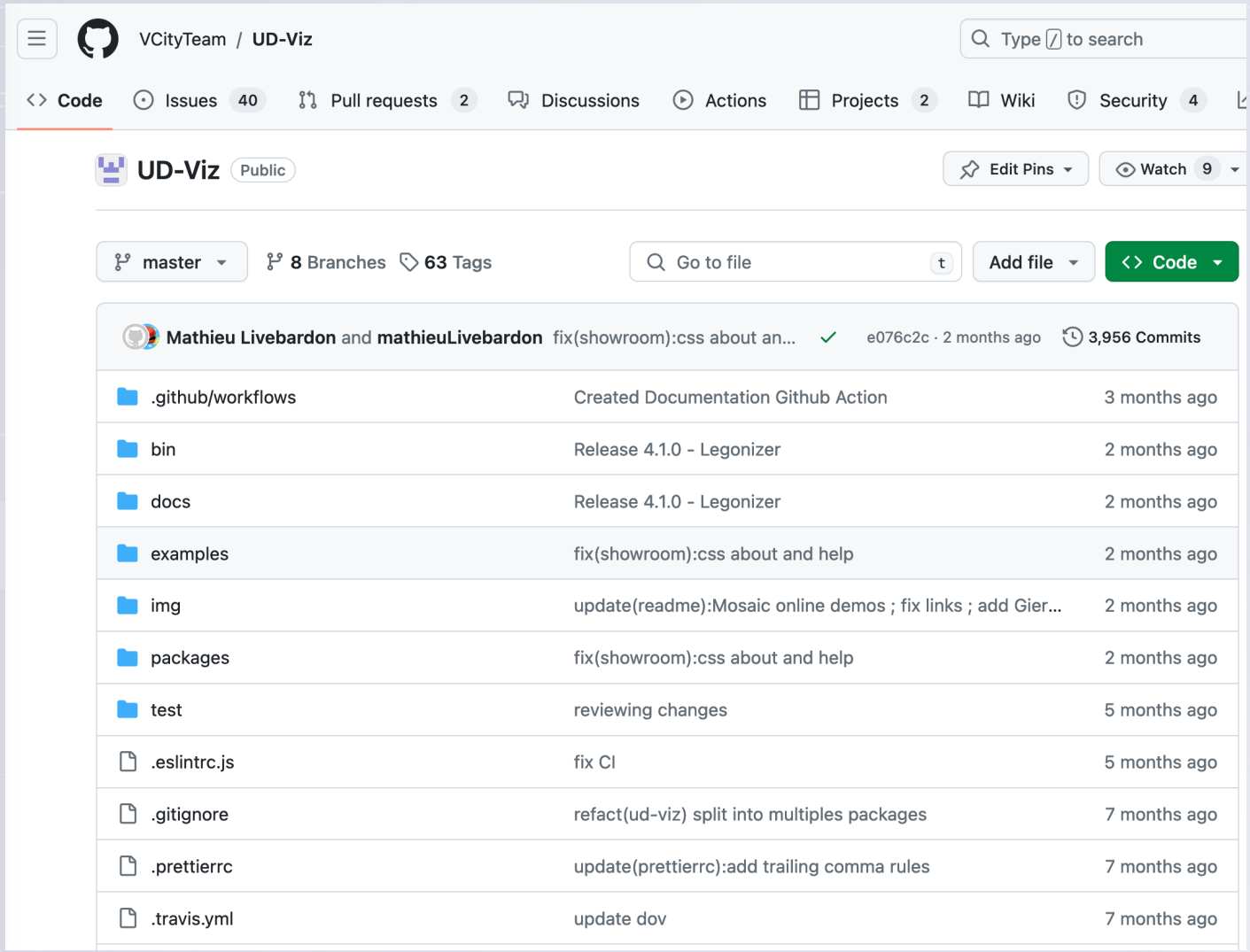
Computational environment

- Provide the actual environment, e.g., a Dockerfile
- Provide a pinned freeze of your dependencies
- Installation and execution instructions for different operating systems

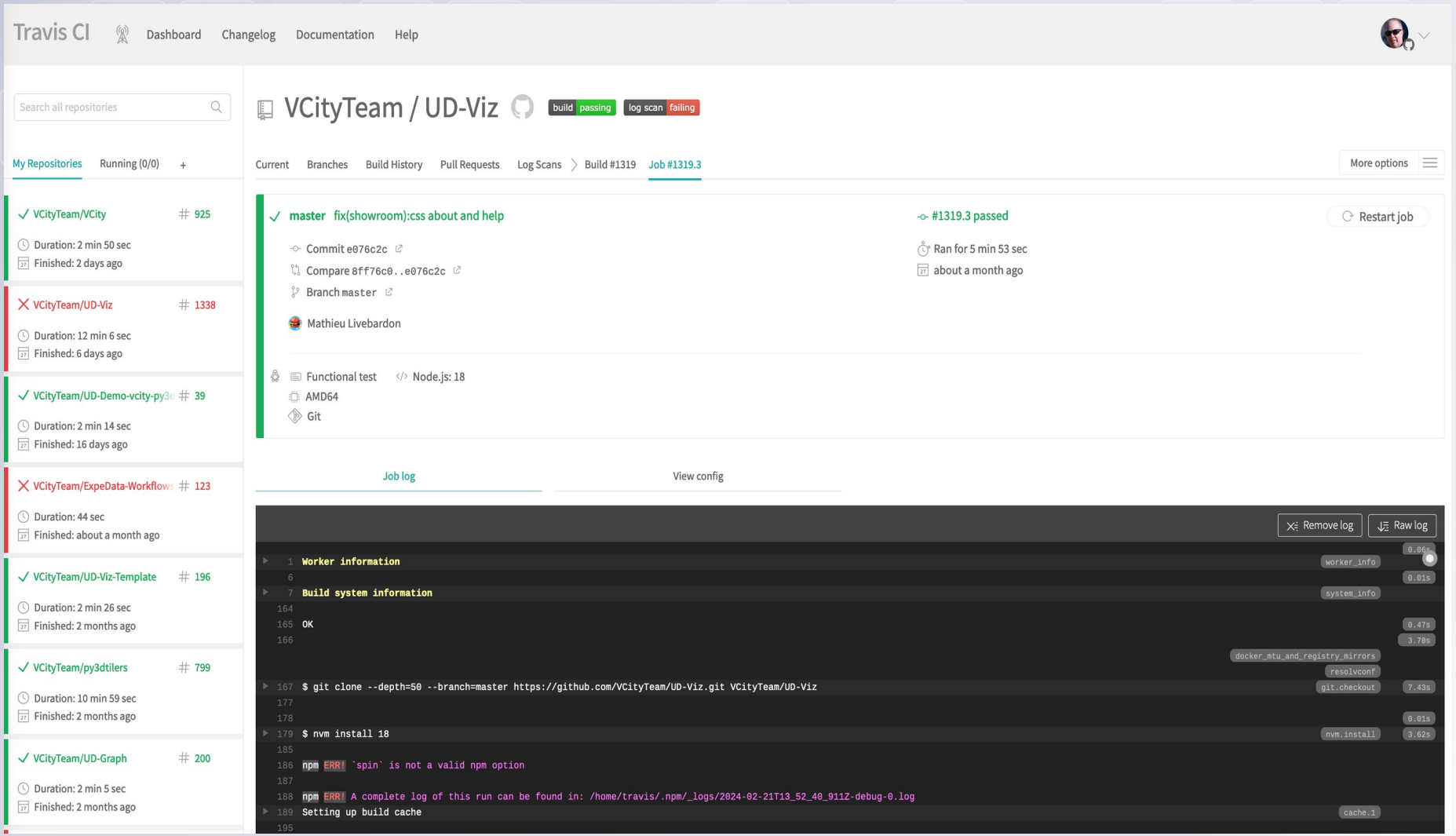
Computational steps

- Document the detailed steps in a text file and/or flowchart (every action/click)
- Scripts/models and a README file that explains their use

Continuous Integration tools should nail it



CI worker logs



Executed script: your job to write

```
# .travis.yml file
language: cpp
dist: trusty # Ubuntu 14.04 Trusty Tahr

os:
  - linux
  - osx

compiler:
  - gcc
  - clang

# For C++ projects, the env, compiler and os (provided as arrays)
# multiply to construct the build matrix.
matrix:
  fast_finish: true
  exclude:
    - os: osx
      compiler: gcc

addons:
  apt:
    packages:
      - libboost-date-time-dev
      - libboost-filesystem-dev

install:
  - if [ "$TRAVIS_OS_NAME" == "linux" -a "$QT" == "4" ]; then
    sudo apt-get -y install libqt4-dev libqt4-opengl-dev; fi
  - git clone MY_DEPENDCY_LIB
  - path MY_DEPENDCY_LIB

script:
  - ... < Build your binaries, run them, test them > ...
```

CI has some limitations

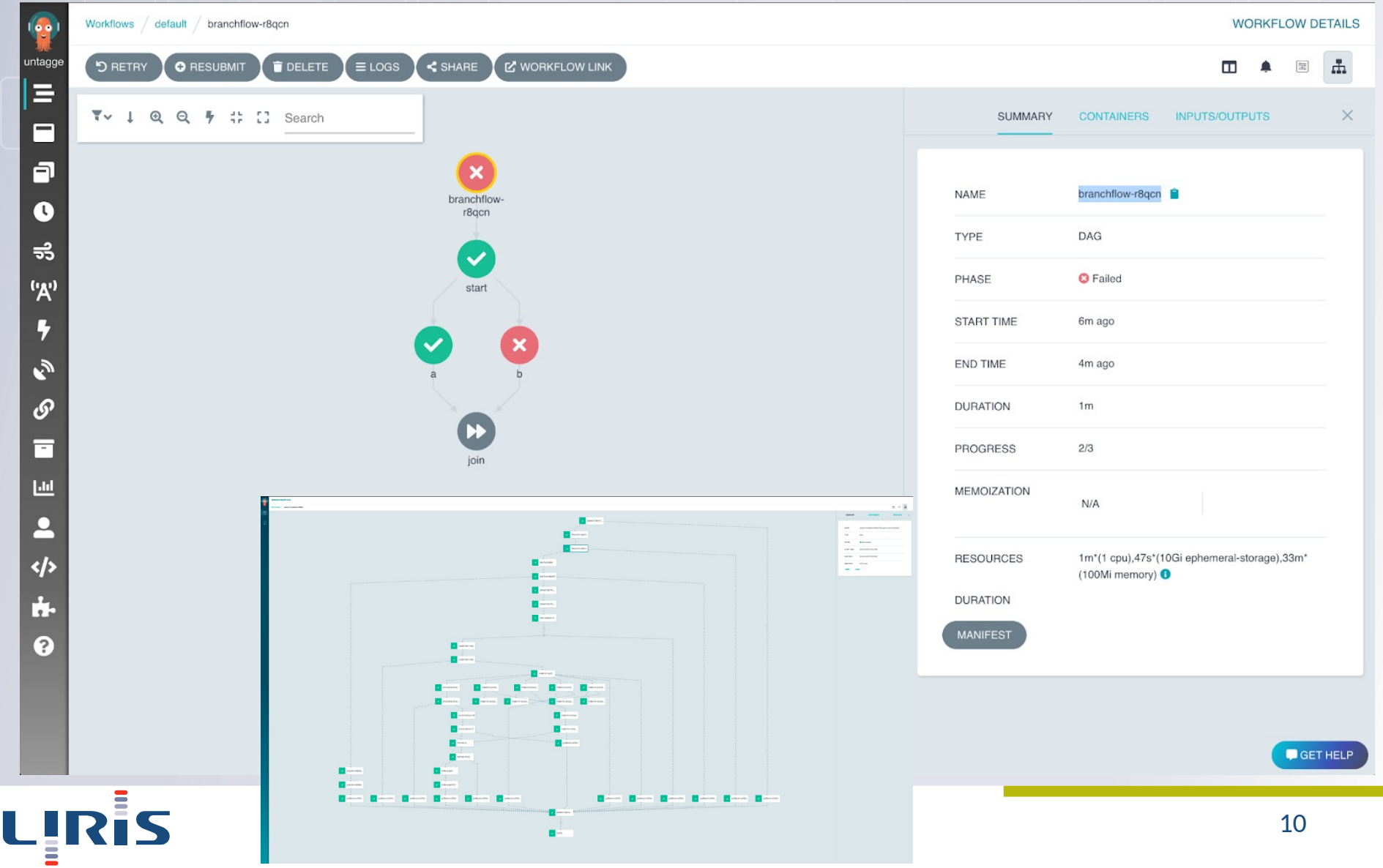
- Runs are independent from one another
  - Not conceived for expressing numerical experiments
  - How to combine many run results into my graphic ?
- Exfiltration of resulting artifacts is not easy
  - Getting ones hand on results (tables, graphics) is a job
- Description of the context is _really_ limited
  - What if need an upstream streaming feed ?
  - What is I need a patched database ?
- Containers (Docker) domination: **the numerical experiment building block can (will?) be docker-based**

Today's numerical experiments use many sub-softwares

```
# Shell script calling docker
echo -e "\n*** CLEANING UP PREVIOUS RUN GARBAGE ***\n"
docker-compose run --rm py3dtilers  rm -rf /datademo/lods_3dtiles

echo -e "\n*** STEP1 ***\n"
docker run [...] pc2vol -i /data/galeries.pts --gridstep 0.025 -o /data/gal.vol
docker run [...] dgtal volSurfaceRegularization -i /data/gal.vol -o /data/gal.obj

echo -e "\n*** STEP2 ***\n"
docker run [...] mepp2  triangulate_faces /data/gal.obj /data/gal_triangulated.obj
docker run [...] py3dtilers obj-tiler -i /data/gal_triangulated.obj -o /data/lods_3dtiles
```

- py3tilers, pc2vol, dgtal, mepp2 are independent softwares, with their own source repository, build chain, dependencies…
- A publication reviewer has better chances at verifying results

Experiment with a Direct Acyclic Graph and docker components



Workflow expression still hurts

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
spec:
  entrypoint: main
  volumes:
  - name: workdir
    persistentVolumeClaim:
      claimName: vcity-pvc
      readOnly: false
  arguments:
    parameters:
    # Numerical experiment related
    - name: boroughs
    - name: pattern
    # Derived parameters
    - name: database_dump_filename
      value: "{{workflow.parameters.experiment_output_dir}}/result-{{workflow.parameters.database_name}}.sql"

  templates:
  - name: main
    steps:

    ### Looping to start databases as DAEMON/services
    - - name: 3dcitydb-start-db-loop
        template: 3dcitydb-daemon-vintaged
        arguments:
          parameters:
          - name: vintage
            value: "{{item}}"
          - name: database_name
            value: "{{workflow.parameters.database_name}}-{{item}}"
          - name: password
            value: "{{workflow.parameters.database_password}}"
          - name: user
            value: "{{workflow.parameters.database_user}}"
          - name: port
            value: "{{workflow.parameters.database_port}}"
        withParam: "{{workflow.parameters.vintages}}"
```

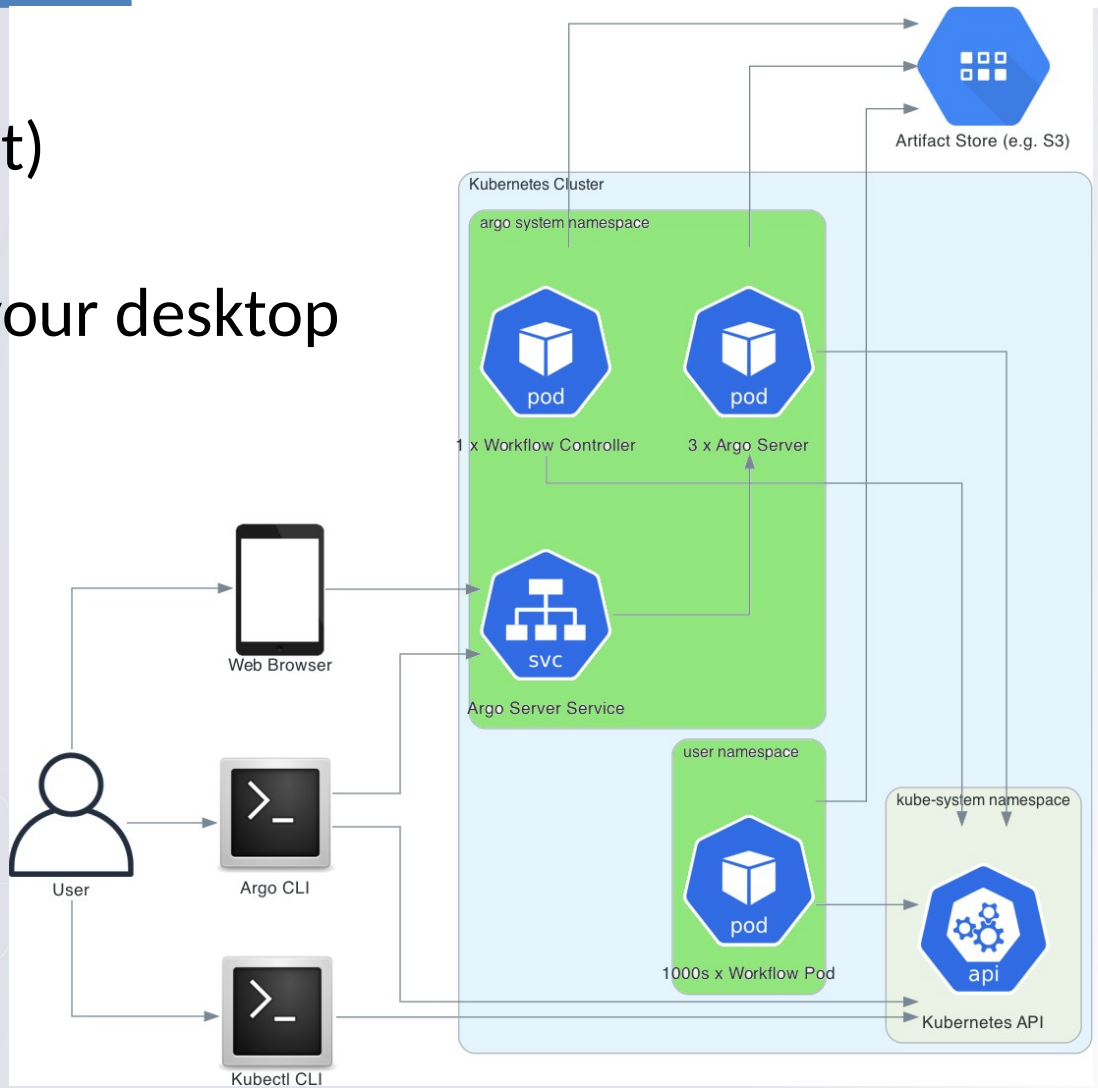
Workflow expression getting better

```
from hera.workflows import DAG, Task, models, Parameter, Workflow

with Workflow(generate_name="import-gml-", entrypoint="main") as w:

    with DAG(name="main"):
        for vintage in inputs.parameters.vintages:
            start_db_t = Task(
                name="start-db-daemon-" + str(vintage),
                template=threedcitydb_containers[vintage],
            )
            import_vintage_boroughs_t = Task(
                name="import-" + str(vintage) + "-boroughs",
                template_ref=models.TemplateRef(
                    name="workflow-import-" + str(vintage),
                    template=db_import_boroughs_template_names[vintage],
                ),
                arguments={"dbhostaddr": start_db_t.ip},
            )

            start_db_t >> import_vintage_boroughs_t
w.create()
```

Reproducing computations requires platforms (just like CI) … which hurts

Hard to deal with (abstract)

- the Kubernetes layer
- This includes installing your desktop
- Code generation layer



Automatized numerical reproducibility is coming.

Au moins la table ronde permet d'en parler :-)