Langfristige Nutzung von Forschungssoftware

Rafael Gieschke, Dirk von Suchodoletz, Universität Freiburg;
Klaus Rechert, Hochschule Kehl

Universität Freiburg:
Rechenzentrum

Software Preservation

- Software preservation is an important requirement to ensure re-use certain digital (data-)objects

EaaSI

https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/

Software Preservation

- Software preservation is an important requirement to ensure re-use certain digital (data-)objects
- **But re-using preserved software requires infrastructure**
  - Runtime dependencies (operating system, libraries, etc…)
  - Hardware or hardware equivalents (emulation)

EaaSI

https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/

Software Preservation

- Software preservation is an important requirement to ensure re-use certain digital (data-)objects
- **But re-using preserved software requires infrastructure**
  - Runtime dependencies (operating system, libraries, etc…)
  - Hardware or hardware equivalents (emulation)
- And workflows, automation to make usable and reproducible

https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/

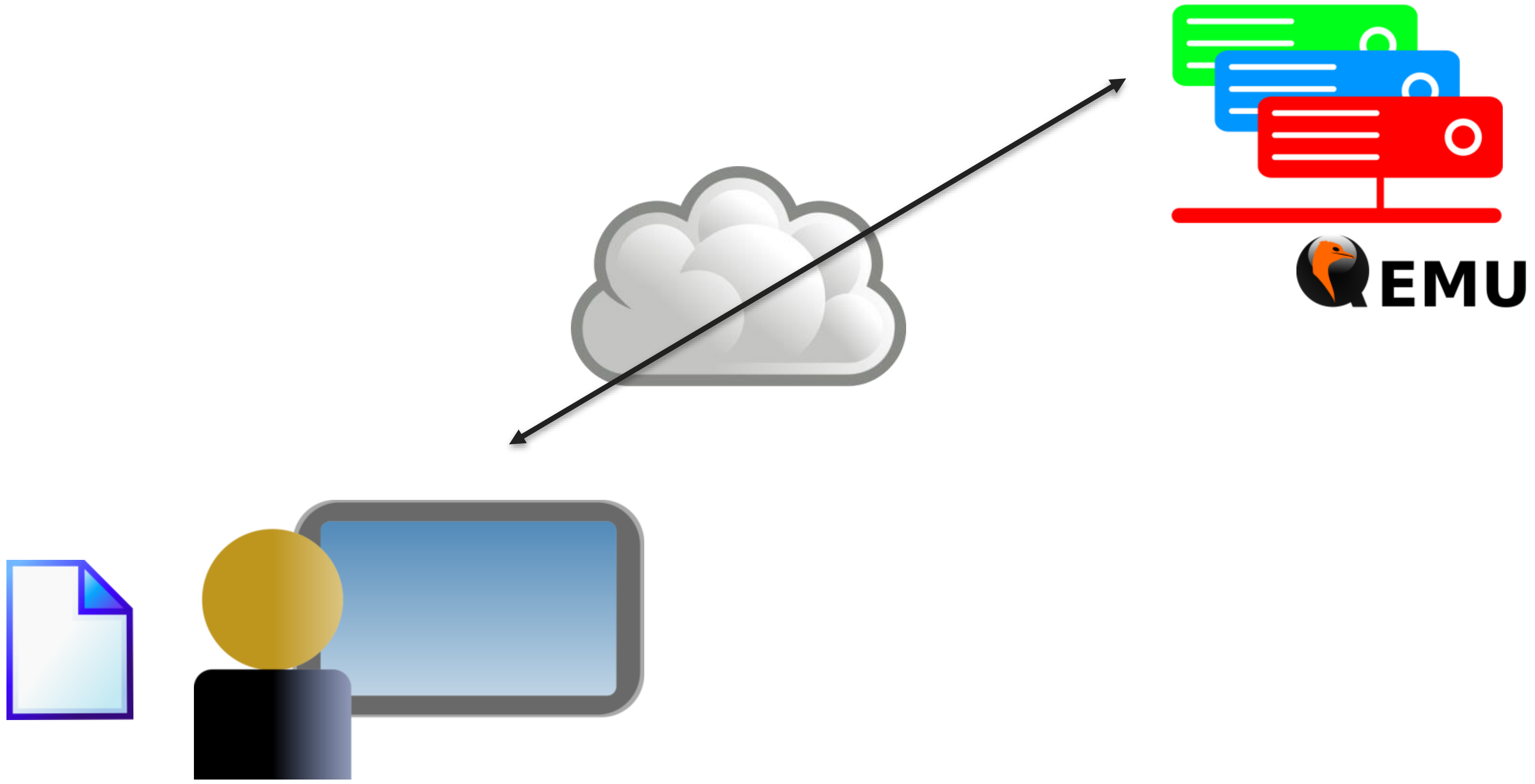
EaaS and CiTAR

- Preservation of born-digital objects from Apple II to IBM PC
  - Web-based
  - Used by libraries to allow visitors to, e.g.,
    - play archived computer games
    - access content (in historic file formats) on CD-ROMs
  - Migration often not possible or with potential loss of information
    - → emulation
    - provide and suggest pre-configured emulation environments (operating system, software to view/edit file format)
- Same problems in RDM
  - Migration not possible
  - Preservation must be done on different levels
    - software itself (e.g., container images)
    - operating system
    - hardware

Three example workflows

1. Base Emulation-as-a-Service system rendering digtial objects
2. Historic builds
3. Preserving containers

1. Emulation-as-a-Service rendering digital objects



Demo

https://b651fad4-55ac-4126-86f4-0298c23e8eb0.test.emulation.cloud/

2. Historic builds

Historic Builds available

Ensuring usability of a scientific code base

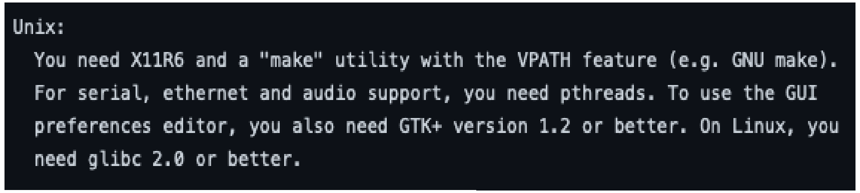
Klaus Rechert Uni Freiburg

Jurek Oberhauser Uni Freiburg

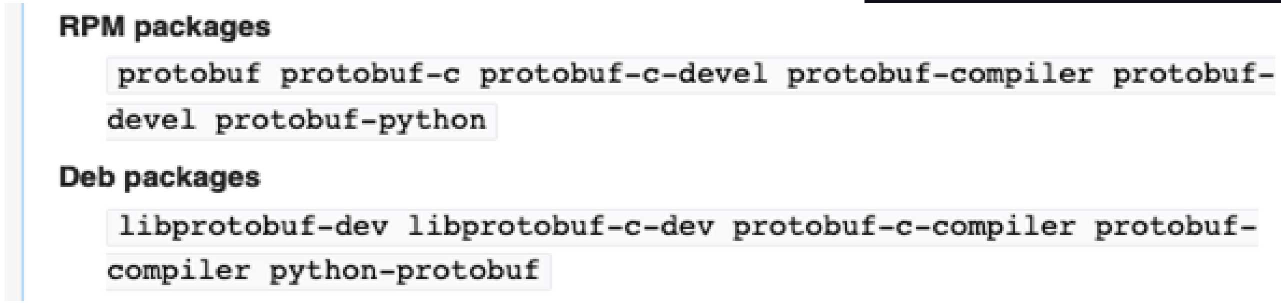
Rafael Gieschke Uni Freiburg

Archiving and Accessing Scientific Code

- Re-create (re-build) code from source requires another set of dependencies
  - → build dependencies
- Usually no formal description, implicit for a given time context



```
Unix:
  You need X11R6 and a "make" utility with the VPATH feature (e.g. GNU make).
  For serial, ethernet and audio support, you need pthreads. To use the GUI
  preferences editor, you also need GTK+ version 1.2 or better. On Linux, you
  need glibc 2.0 or better.
```



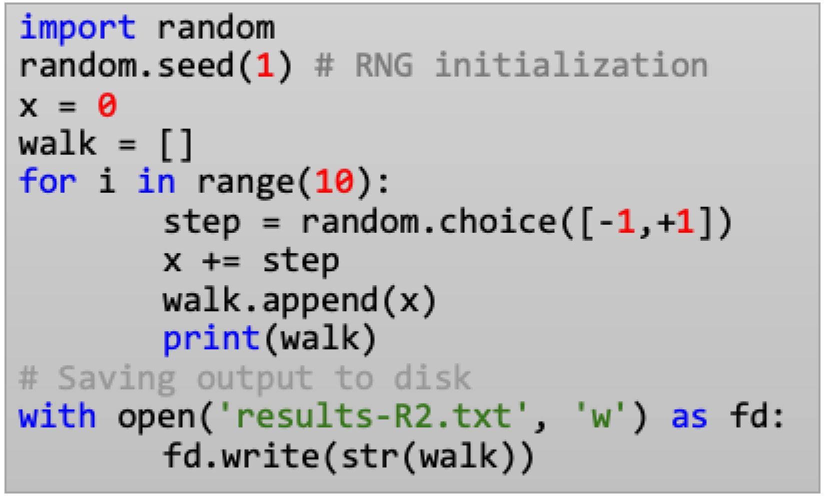
RPM packages

`protobuf protobuf-c protobuf-c-devel protobuf-compiler protobuf-devel protobuf-python`

Deb packages

`libprotobuf-dev libprotobuf-c-dev protobuf-c-compiler protobuf-compiler python-protobuf`

Example



```python
import random
random.seed(1) # RNG initialization
x = 0
walk = []
for i in range(10):
        step = random.choice([-1,+1])
        x += step
        walk.append(x)
        print(walk)
# Saving output to disk
with open('results-R2.txt', 'w') as fd:
        fd.write(str(walk))
```

Python <3.2:
−1, 0, 1, 0, −1, −2, −1, 0, −1, −2

Python >=3.3:
−1, −2, −1, −2, −1, 0, 1, 2, 1, 0

From:
“Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions” von Fabien C. Y. Benureau und Nicolas P. Rougier.

Infrastructure and Requirements

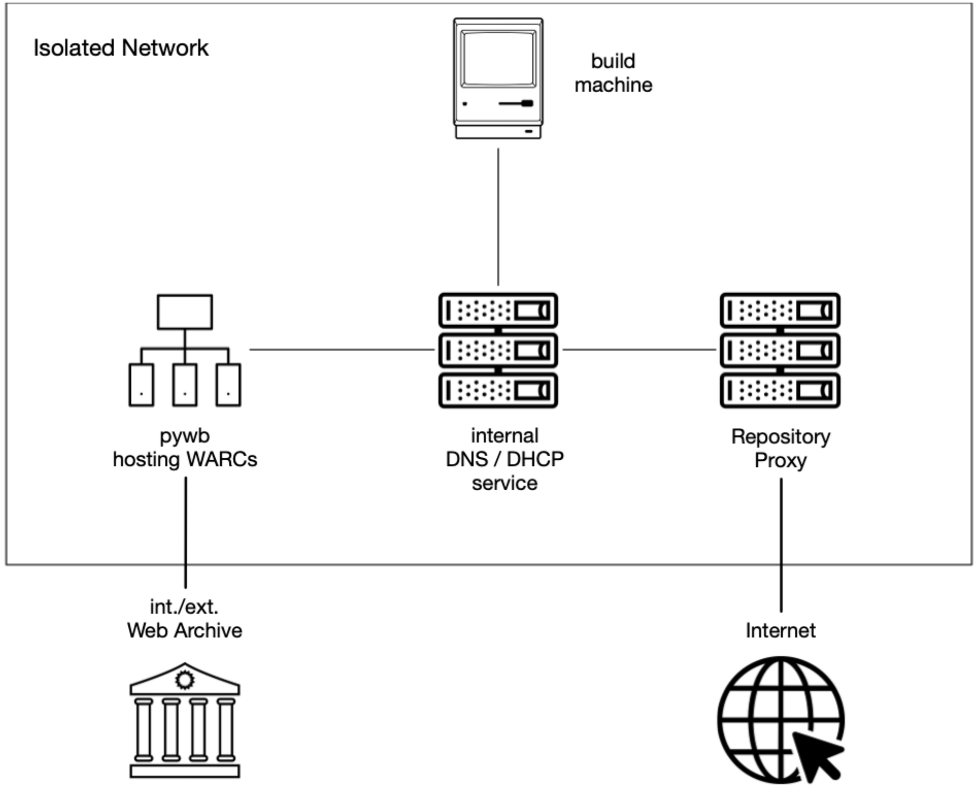
- Archival, management and access to historic (build) environments
  - Emulation infrastructure EaaS(I), CiTAR, etc.
  - Systematic collection of relevant “base” environments
  - Maintain/archive/manage external software repositories
    - Distribution repositories
    - Special “repositories” like npm, pip, CPAN, …

→ ideally have snapshots to allow implicit versioning

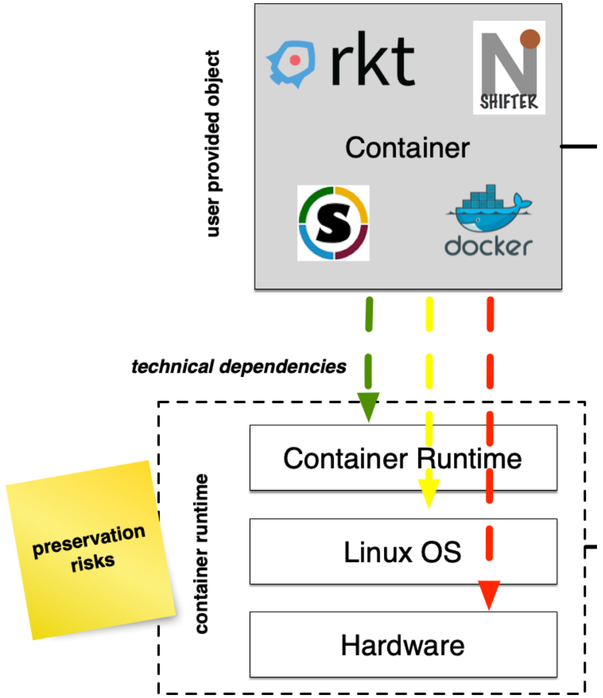
Historic Build Environments

Build environment

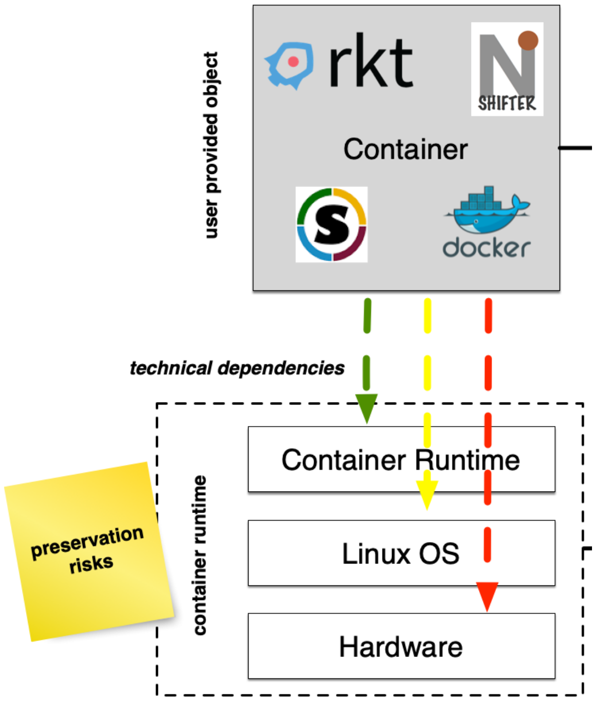
- Machine (emulator) + installed disk image
- Managed context
  - E.g. time/date settings
  - Isolated network with transparent repository mapping



3. Preserving containers



Preserving containers



Mostly standardized as Open Container Initiative (OCI) Image Format (layered .tar.gz or .tar.zst files)

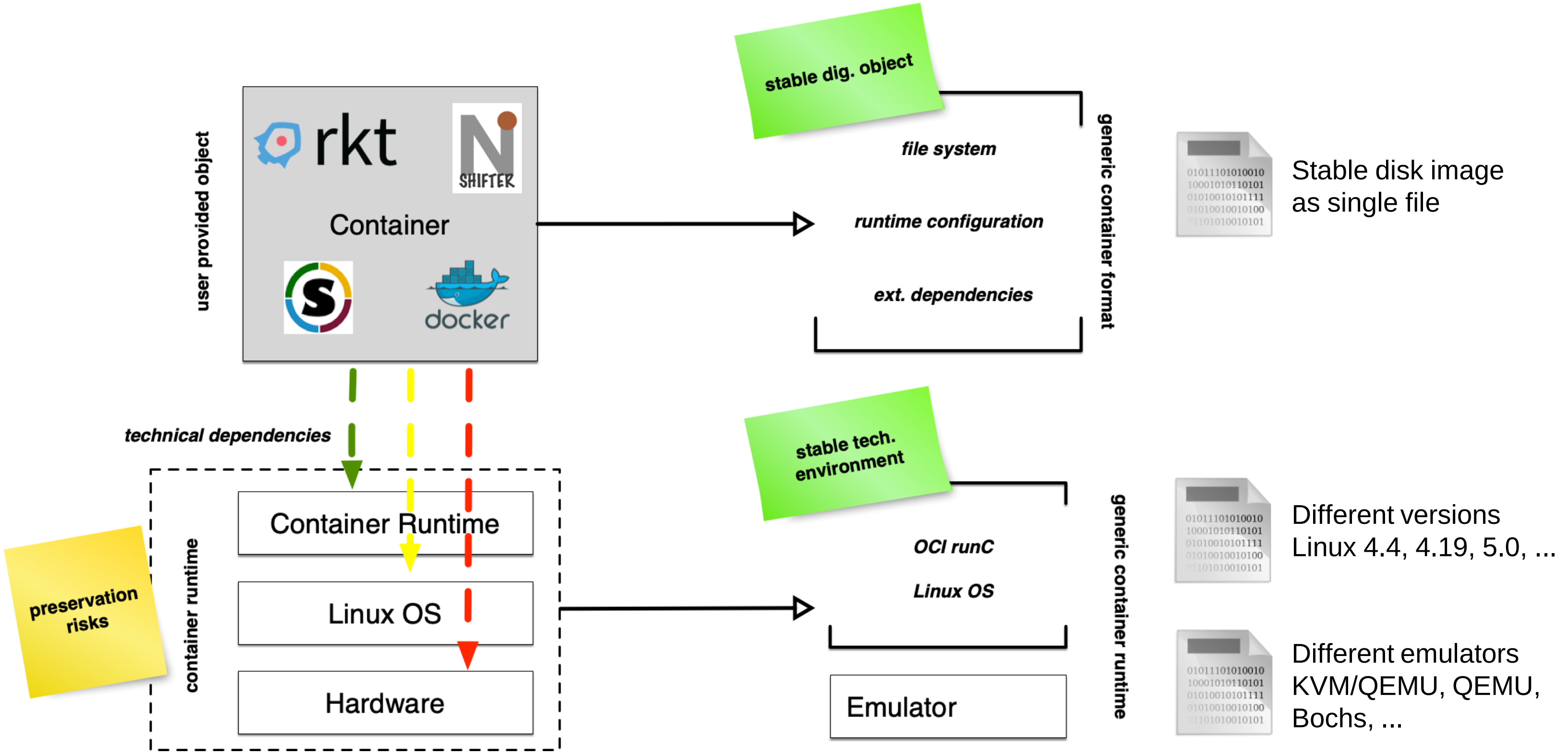
Might not be available forever on, e.g., Docker Hub (freemium), required version (“latest”) might not be obvious (or not be available anymore/overwritten)

Somehow standardized in Open Container Initiative (OCI) Runtime Specification

Generally provides very strong backward compatibility

Plattform: abstracted away by OS
ISA/CPU: x86(-64) not standardized, mainly two implementations; backward-(in)compatible extensions?

Preserving containers



Demo

https://a19b53c8-2990-43ef-8ccd-6353c370d056.test.emulation.cloud/

Conclusion

- Boundaries between software and data fluent
- Both are digital objects
- Make objects as uniform as possible to scale
- Need emulation for long-term access
- Need workflows to make usable
- Need integration into existing platforms
- Into preservation system
- Into user’s workflows: continuous access and preservation