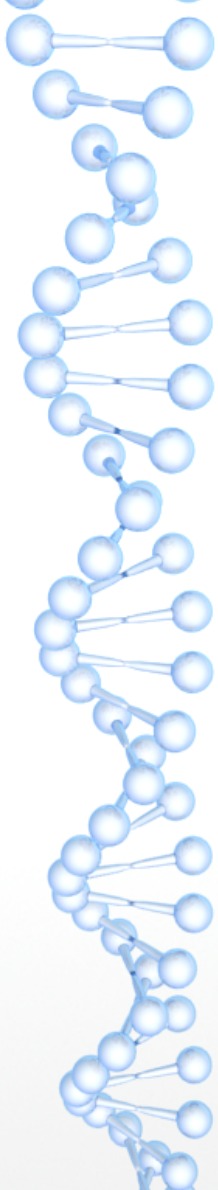
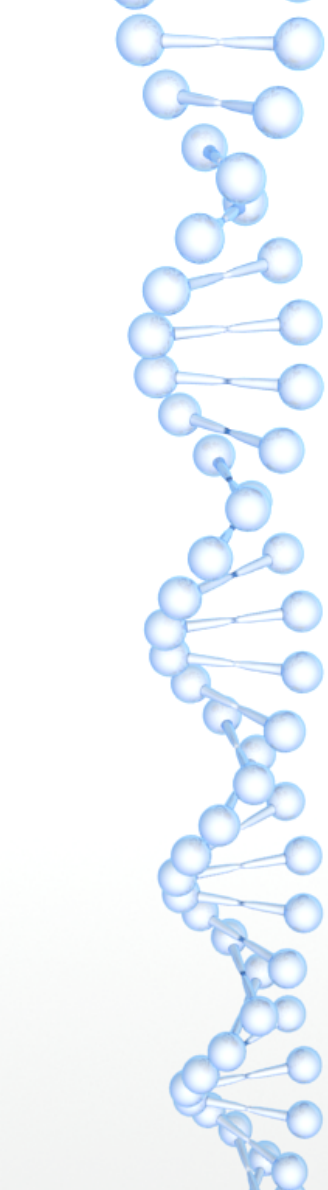


REPRODUCIBILITY

... the difference between science and magic

František Kluknavský
fkluknav@redhat.com
2017

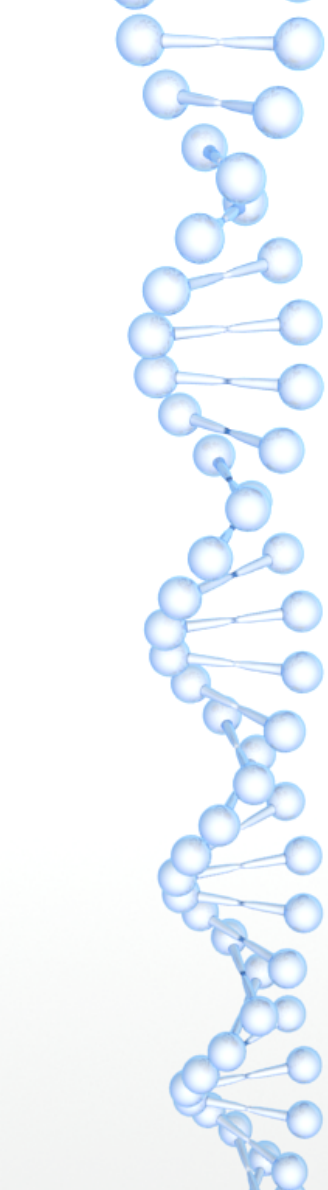


Science

- Knowledge in the form of testable predictions about the universe
  - Reproducible experiments
  - Theory that explains experiments
  - Expectations about future experiments follow from the theory
- Reproducibility is the key. Do not create myths.

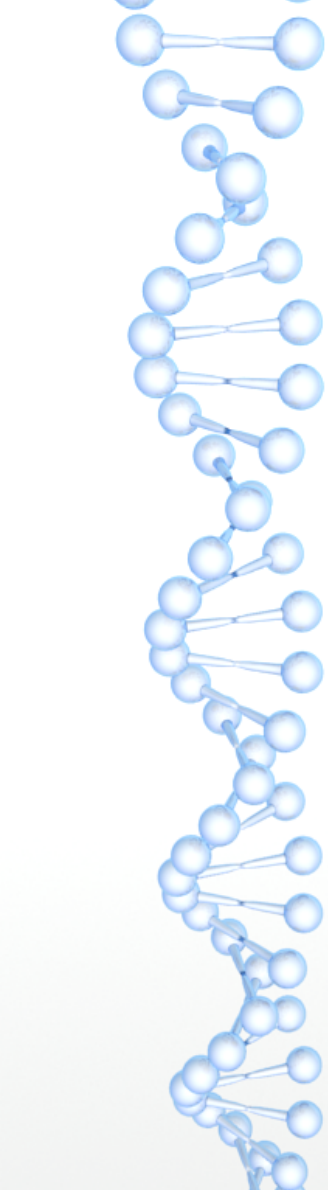
Magic

- Let us (mis)use the term to descibe non-science
  - Includes irreproducible unexplained elements.
    - This holds for any kind of magic. David Copperfield, Gandalf, genie in a bottle... even the most friendly witch in the most friendly fairytale gives you only one magic herb and you can not get more.
  - Apologies to any magician who has a different definition.
- Magic surely is a great thing. However, it should never ever be confused with science.
- Are particle colliders and computers science or magic?



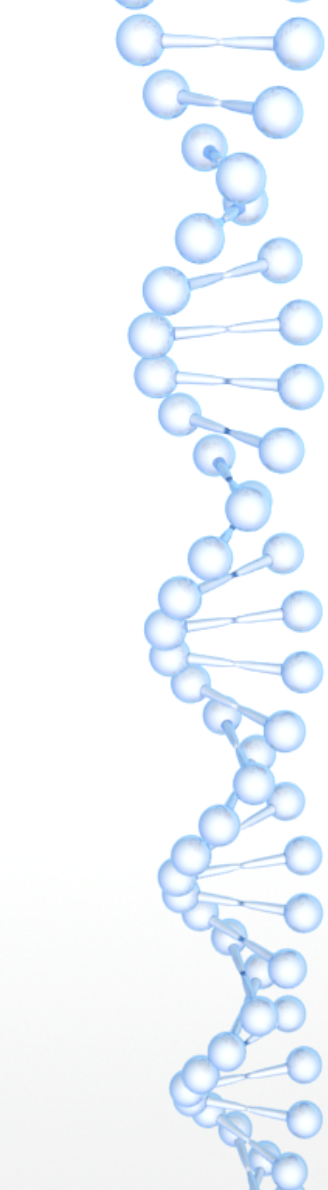
Magical software

- Software you can not inspect and run is as good as magical
  - Free, Libre, Open Source – crucial but not today‘s topic. The smaller the blackbox is, the better. Experiment, operating system, firmware, hardware...
- Software that is impractically difficult to run is no better
  - Motivation for this talk – reconstructing a computation from a scientific paper is too often a real pain or outright impossible
  - Start with bachelor and master thesis



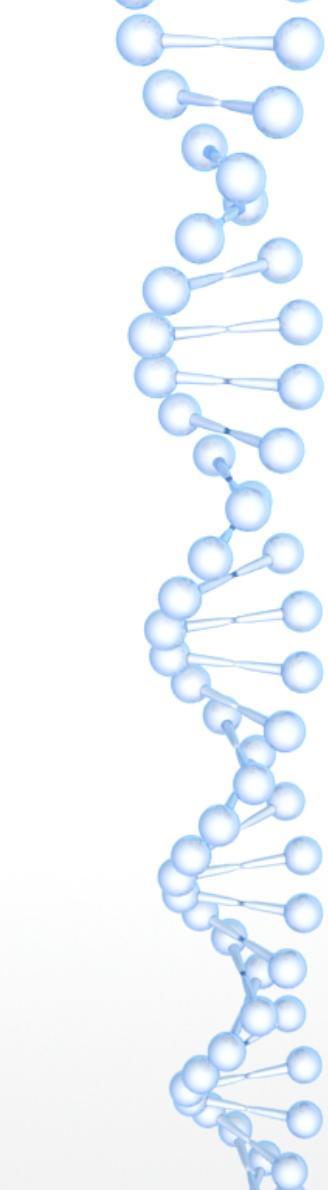
It is magic and not science if you...

- Publish conclusions without publishing source data
- Publish data but not software
  - Other people might use similar methods and get different results
  - If you can work with graphical tools only, learn some basic scripting. Seriously. The sooner the better.
- Publish software that is hard to run – as if you did not publish it at all
  - Languages and compilers evolve
  - Libraries change API or disappear completely
  - Hardware changes
  - Operating system changes
  - Configuration files, environment variables, filesystem structure...



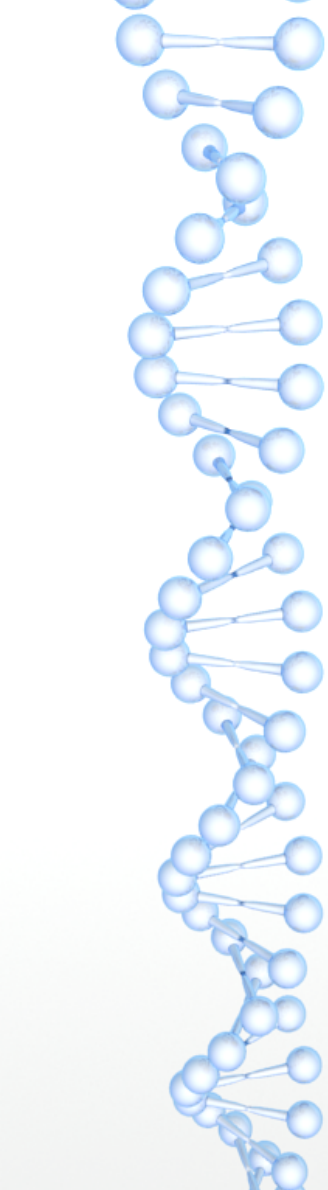
Instant soup – just add water

- A (the best?) compromise
  - Mostly complete package, relatively small and light
  - Water is the largest fraction
  - Water is easy to get and effortless to add
- Let us make running software as easy as cooking instant soup
- Linux kernel is stable and widespread – like water
  - Time horizont of <single digit> years
  - Even on M$
  - Some features not immediately available on new/rare hardware
- Linux kernel is too big to ship with every „Hello, world“ application – like water



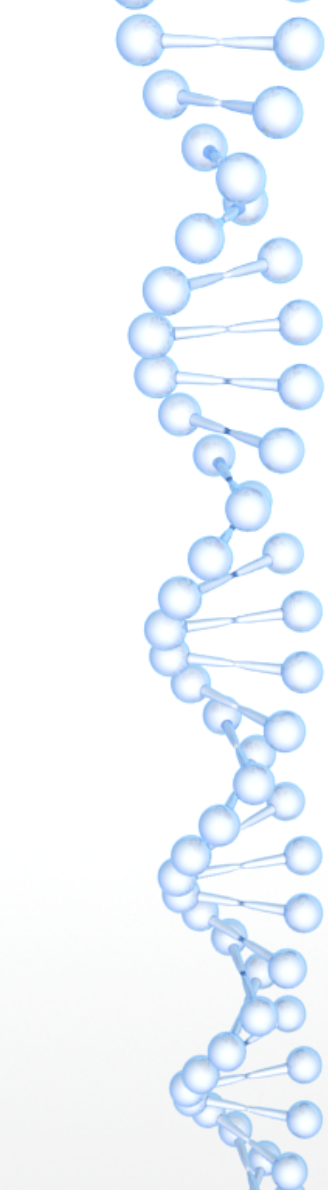
... stir and cook 5 minutes

- Do not build a new kitchen for cooking one new meal
- Do not learn to cook from scratch
- We need a standard for running applications – make the kernel easy to add
  - Tarball, chroot, LXC, docker, rocket, flatpack, libvirt, QCOW2...
- Too many incompatible tools – almost as bad as having no tool at all



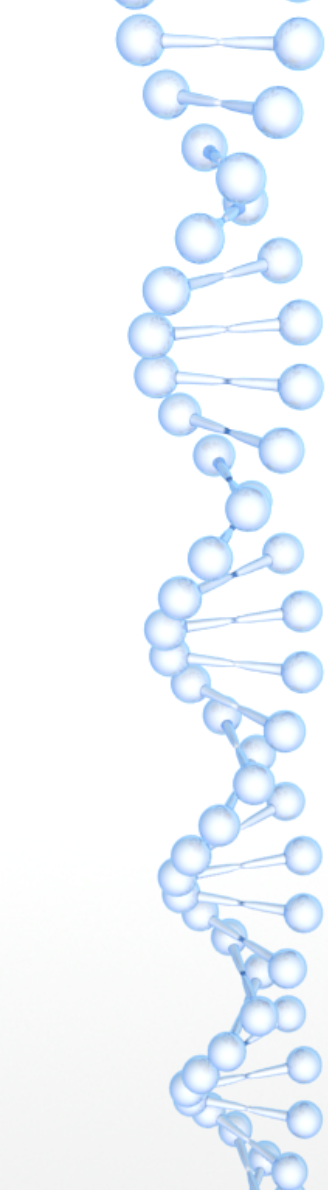
OCI

- Open Container Initiative, www.opencontainers.org
- Standard – the PDF of containers
- Build and run anywhere – ecosystem of tools
  - In a virtual machine if necessary
- Having multiple compatible tools is a good thing.
- Having multiple incompatible formats is a bad thing.



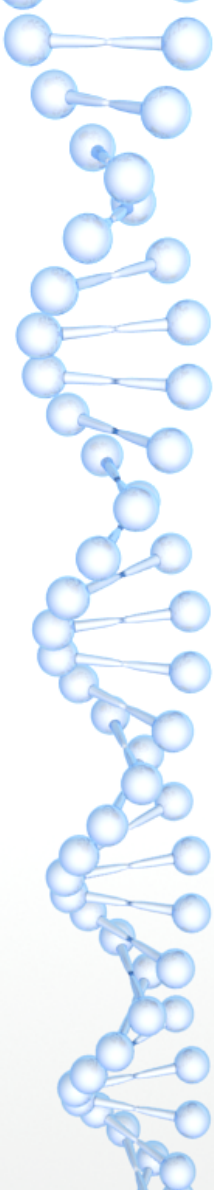
The simplest scenario - one-shot jobs

- Takes input, runs a computation, writes output
  - Preferably the same output for the same input
- No <buzzword>
  - No microservices
  - No CI
  - No CD
  - No open hybrid cloud, not even cloud
  - No pet vs cattle
- Scientific computing, simulations, financial reports, demos, publications...



How to

- Include README
- Include libs for your software
  - Lapack (ATLAS, OpenBLAS...), Scipy, Numpy, Tensorflow...
  - Compilers if needed
- Include source code of your software in a tarball
- Unpack and compile using tools already in the image
- Include the script that did all of the above
- Bind-mount storage with data – stdin might be difficult to use
  - /input
  - /output



Discussion: Alternative suggestions welcome

Java? No.
Unikernels? No.
Dotnet? No.
...