Reproducible Research at the Cloud Era

Overview, Hands-on and Open Challenges

Sébastien Varrette, PhD

Parallel Computing and Optimization Group (PCOG),
University of Luxembourg (UL), Luxembourg

http://RR-tutorials.rtfd.io

Before the tutorial starts: Visit
https://goo.gl/l9mCsM

for *preliminary setup instructions*!

uni.lu
UNIVERSITÉ DU
LUXEMBOURG

Summary

About me

https://varrette.gforge.uni.lu

- **2003 – 2007**: PhD between INP Grenoble & UL
  - ↪ *Security in Large Scale Distributed Systems: Authentication and Result Checking*
- **2007 – now**: *Research Associate* at UL
  - ↪ Part of the PCOG Team led by Prof. P. Bouvry
  - ↪ Manager of the UL High Performance Computing Facility
    - ✓ ≃ 197 **T**Flops (2017), 5.844 **P**B, 4 sysadmins

Research Interests: Distributed Computing Platforms

- Security (crash/cheating faults, obfuscation) in DGVCS
- Performance of HPC/cloud platforms
  - ↪ Energy Efficiency, Performance, Cost...

Disclaimer: Acknowledgements

- A **large** part of these slides were **courtesy** borrowed, with permission, from:
  - ↪ Lucas Nussbaum (INRIA, Univ. Lorraine)
  - ↪ Arnaud Legrand (INRIA, Univ. Grenoble)
  - ↪ Valentin Plugaru (Univ. of Luxembourg)
  - ↪ and many others...

- In particular, to know more about **Reproducible Research**:
  - ↪ Webinars on Reproducible Research `https://github.com/alegrand/RR_webinars`
  - ↪ Reproducible build `https://reproducible-builds.org/`
    - ✓ initiative of various free software projects

...

Agenda: Dec. 12th, 2016

Time	Session
09:00 – 10:00	Reproducible Research in Computer Science
10:00 – 10:30	Hands-On: Build these slides using Vagrant
10:30 – 11:00	**Coffee Break**
11:00 – 11:30	Hands-On: Reproducible Software Environment with Easybuild
	Hands-On: Docker
	Reproducible Results
12:15 –	**Lunch**

Tutorial Pre-Requisites / Setup

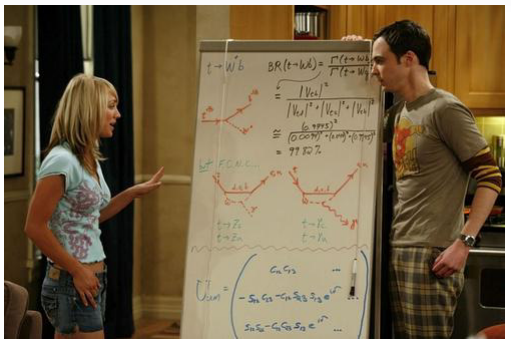
http://RR-tutorials.readthedocs.io/en/latest/setup/

- Create (if need) accounts for the **cloud services** we will use:
  - ↪ Github, Vagrant Cloud and Docker Hub
- Install **mandatory software**, *i.e.* (apart from Git):
  - ↪ Virtual Box https://www.virtualbox.org/
  - ↪ Vagrant https://www.vagrantup.com
  - ↪ Docker https://www.docker.com/
- Check installed software and download the boxes we will use:

```
$> git clone https://github.com/Falkor/RR-tutorials.git
$> cd RR-tutorials
$> make setup
$> vagrant up &&  docker pull ubuntu:14.04  # might take some time...
```

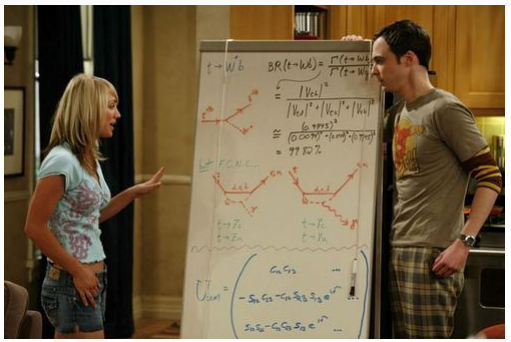
Summary

Validation in (Computer) Science



- Two classical approaches for validation:
  - ↪ **Formal**: equations, proofs, etc.
  - ↪ **Experimental**, on a scientific instrument
- Often a mix of both:
  - ↪ In Physics
  - ↪ In Computer Science
- Quite a lot of **formal** work in Computer Science
- But also quite a lot of experimental validation
  - ↪ Distributed computing, networking
    - ✓ testbeds: IoT-LAB, Grid'5000...
  - ↪ Language/image processing ⇝ evaluations using large corpuses

Validation in (Computer) Science



- Two classical approaches for validation:
  - ↪ **Formal**: equations, proofs, etc.
  - ↪ **Experimental**, on a scientific instrument
- Often a mix of both:
  - ↪ In Physics
  - ↪ In Computer Science
- Quite a lot of **formal** work in Computer Science
- But also quite a lot of experimental validation
  - ↪ Distributed computing, networking
    - ✓ testbeds: IoT-LAB, Grid'5000...
  - ↪ Language/image processing ⇝ evaluations using large corpuses

How good are we at performing experiments?

(Poor) State of Experimentation in CS

- **1994**: survey of 400 papers[1]
  - ↪ among published CS articles in ACM journals
  - ↪ **40%-50%** of those **requiring** an experimental validation **had none**
- **1998**: survey of 612 papers[2]
  - ↪ too many papers have **no experimental validation at all**
  - ↪ too many papers use an informal (assertion) form of validation
  - ↪ 2009 update: situation is improving[3]

[1] Paul Lukowicz et al. "Experimental Evaluation in Computer Science: A Quantitative Study". In: **Journal of Systems and Software** 28 (1994), pages 9–18.

[2] M.V. Zelkowitz and D.R. Wallace. "Experimental models for validating technology". In: **Computer** 31.5 (May 1998), pages 23–31.

[3] Marvin V. Zelkowitz. "An update to experimental models for validating computer technology". In: **J. Syst. Softw.** 82.3 (Mar. 2009), pages 373–376.

(Poor) State of Experimentation in CS

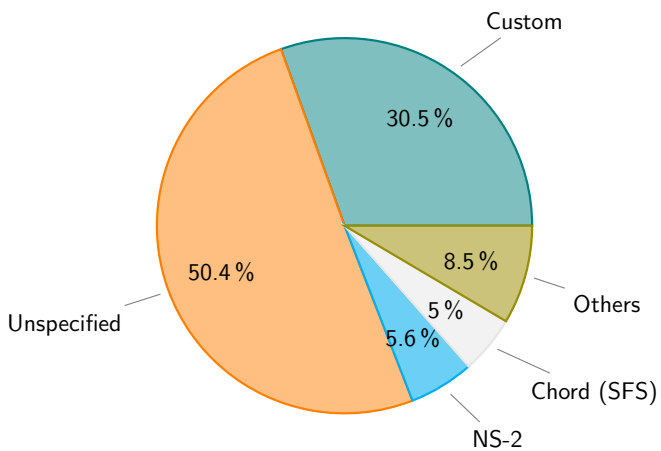
- Most papers **do not use** even basic statistical tools
  - ↪ Papers published at the Europar conference[4]

Year	#Papers	With error bars	Percentage
2007	89	5	5.6%
2008	89	3	3.4%
2009	86	2	2.4%
2010	90	6	6.7%
2011	81	7	8.6%
2007-2001	**435**	**23**	**5.3%**

[4] Study carried out by E. Jeannot.

(Poor) State of Experimentation in CS

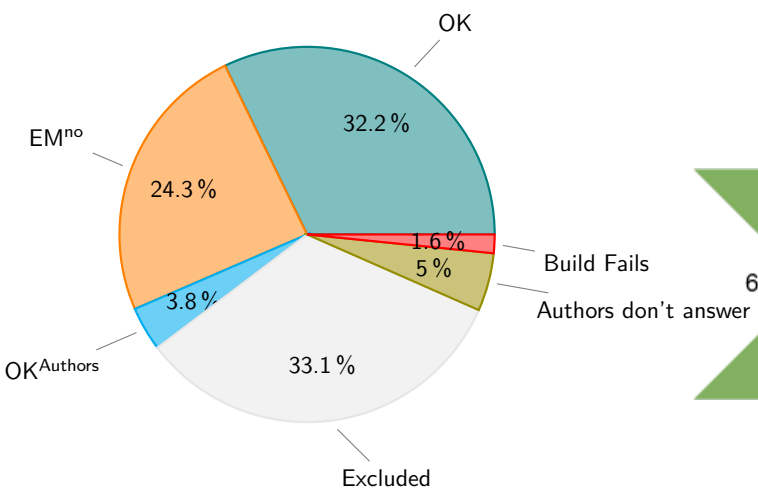
- **2007**: Survey of simulators used in P2P research[5]
  - ↪ 287 papers surveyed on P2P networking subject
  - ↪ **141** of these papers **reports the use of a simulator**
    - ✓ 30% use a custom tool
    - ✓ 50% don't report the used tool!

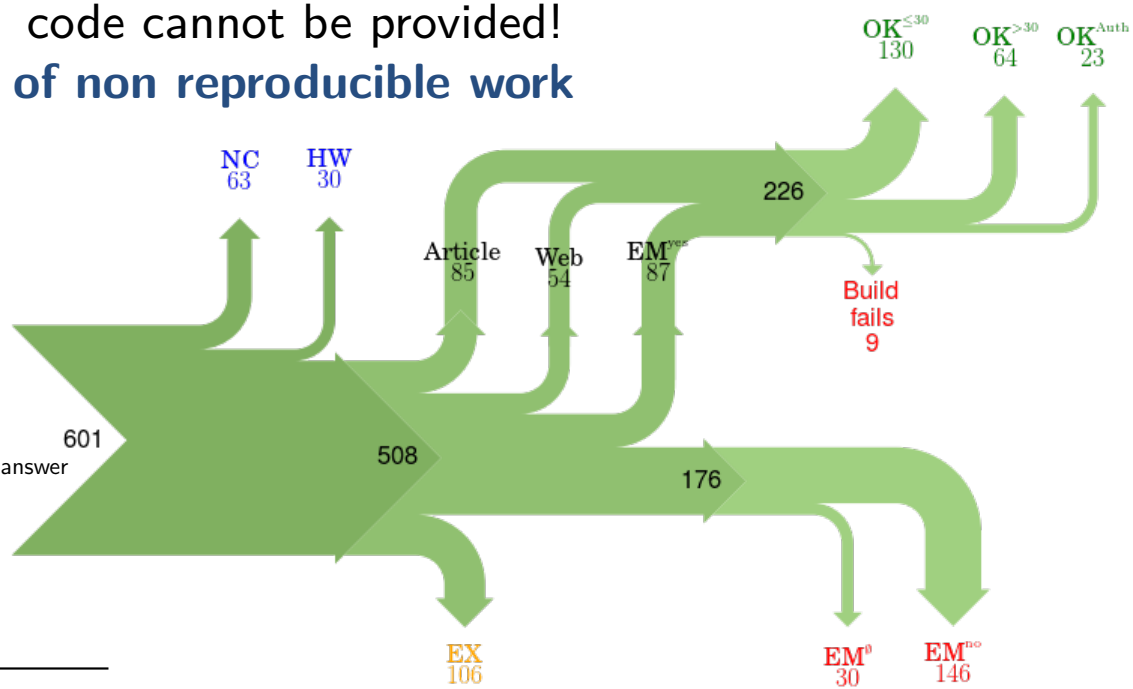


[5] S. Naicken et al. "The state of peer-to-peer simulators and simulations". In: **SIGCOMM Comput. Commun. Rev.** 37.2 (Mar. 2007), pages 95–98.

(Poor) State of Experimentation in CS

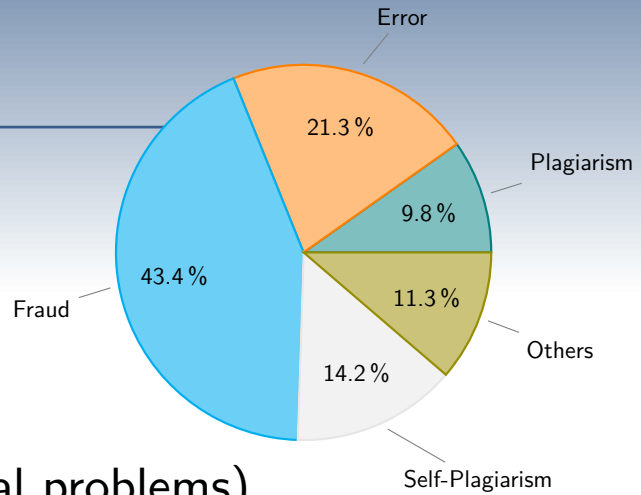
- **2015**: **601 papers** from ACM conferences and journals analysed[6]
  - ↪ **Obj.**: attempt to locate any source code that backed up the published results; **if found, try to build the code**.
  - ↪ EM^{no} (**146 papers!**): code cannot be provided!
  - ↪ Original study: **80% of non reproducible work**





[6] Christian Collberg et al. **Repeatability and Benefaction in Computer Systems Research**. Technical report. http://reproducibility.cs.arizona.edu/. Feb. 2015.

And in Other Sciences?



- **Biology**: Increase in **retracted papers**[7],
  - ↪ **Fraud** (data fabrication or falsification)
  - ↪ **Error** (plagiarism, scientific mistake, ethical problems)
    - ✓ see also Reproducibility: A tragedy of errors[8]
    - ✓ cf.Duke University scandal with scientific misconduct on lung cancer
  - ↪ High number of **failing clinical trials**
    - ✓ Do We Really Know What Makes Us Healthy?, 2007
    - ✓ Lies, Damned Lies, and Medical Science, 2010
- **Psychology**:
  - ↪ unreplicable study about extrasensory perception (ESP)
- **Machine Learning**: Trouble at the lab, The Economist, 2013

  *According to some estimates, three-quarters of published scientific papers in the field of machine learning are bunk because of this "overfitting". **Sandy Pentlan, MIT***

[7] R Grant Steen. "Retractions in the scientific literature: is the incidence of research fraud increasing?" In: **J Med Ethics** 37 (2011). http://dx.doi.org/10.1136/jme.2010.040923, pages 249–253.

[8] David B. Allison et al. **Reproducibility: A tragedy of errors**. http://www.nature.com/news/reproducibility-a-tragedy-of-errors-1.19264. Feb. 2016.

And in Other Sciences?

- **Medicine**: Study shows lower fertility for mices exposed to transgenic maize (AFSSA report[9])
  - ↪ Several calculation errors have been identified
  - ↪ led to a false statistical analysis & interpretation

[9] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al.

And in Other Sciences?

- **Medicine**: Study shows lower fertility for mices exposed to transgenic maize (AFSSA report[9])
  - ↪ Several calculation errors have been identified
  - ↪ led to a false statistical analysis & interpretation

- **Physics**: CERN / OPERA Experiment (2011)
  - ↪ **faster-than-light neutrinos**
    - ✓ People started gossiping about relativity violation...
  - ↪ caused by timing system failure in 2012

[9] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al.

And in Other Sciences?

- **Medicine**: Study shows lower fertility for mices exposed to transgenic maize (AFSSA report[9])
  - ↪ Several calculation errors have been identified
  - ↪ led to a false statistical analysis & interpretation

- **Physics**: CERN / OPERA Experiment (2011)
  - ↪ **faster-than-light neutrinos**
    - ✓ People started gossiping about relativity violation...
  - ↪ caused by timing system failure in 2012

- ☹: Not everything is perfect
- ☺: But some errors are properly identified
  - ↪ Stronger experimental culture in other (older?) sciences?
  - ↪ Long history of costly experiments, scandals, ...

[9] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al.

What About You (as Rewiever) ?

> *"This may be an interesting contribution but. . . "*

- This **average value** must hide something
- As usual, there is no **confidence interval**,
  - ↪ I wonder about the variability and whether the difference is **significant** or not

What About You (as Rewiever) ?

> *"This may be an interesting contribution but. . . "*

- This **average value** must hide something
- As usual, there is no **confidence interval**,
  - ↪ I wonder about the variability and whether the difference is **significant** or not

- Why is this graph in **logscale**? How would it looks like otherwise?
- That can't be true, I'm sure they **removed some points**
- The authors decided to show only a subset of the data.
  - ↪ I wonder what the rest looks like

What About You (as Rewiever) ?

> *"This may be an interesting contribution but. . . "*

- This **average value** must hide something
- As usual, there is no **confidence interval**,
  - ↪ I wonder about the variability and whether the difference is **significant** or not

- Why is this graph in **logscale**? How would it looks like otherwise?
- That can't be true, I'm sure they **removed some points**
- The authors decided to show only a subset of the data.
  - ↪ I wonder what the rest looks like

- There is no label/legend/. . . What is the **meaning of this graph**?
  - ↪ If only I could access the generation script

What About You (as Author) ?

- I thought I used the *same parameters*...
  - ↪ but I'm **getting different results**!

What About You (as Author) ?

- I thought I used the *same parameters*...
  - ↪ but I'm **getting different results**!

- The new student wants to compare with my **last year' method**
- My advisor asked me whether I took care of setting this or this...
  - ↪ but **I can't remember**
- The damned fourth reviewer asked for a major revision...
  - ↪ he wants me to **change figure 3** ☹

What About You (as Author) ?

- I thought I used the *same parameters*...
  - ↪ but I'm **getting different results**!

- The new student wants to compare with my **last year' method**
- My advisor asked me whether I took care of setting this or this...
  - ↪ but **I can't remember**
- The damned fourth reviewer asked for a major revision...
  - ↪ he wants me to **change figure 3** ☹

- **Which code / data set** did I use to generate this figure?
- It **worked yesterday**!
- 6 months later: just **why** did I do that?

Why is it Hard to Reproduce? (any Scientific Work)

- **Human error**:
  - ↪ Experimenter **bias** *crowdsourced research?*
  - ↪ Programming **errors** or data manipulation **mistakes**
  - ↪ Poorly selected statistical test
- There is just **no real incentive in doing so**:
  - ↪ Legal barriers, **copyright** *Many ongoing discussions in US*
  - ↪ **Competition** issue *researchware, bibliometry, ...*
  - ↪ Publication **bias** *only the idea matters, not the gory details...*
  - ↪ Rewards for **positive/novel results**, not for *consolidating* results

Why is it Hard to Reproduce? (any Scientific Work)

- **Human error**:
  - ↪ Experimenter **bias** *crowdsourced research?*
  - ↪ Programming **errors** or data manipulation **mistakes**
  - ↪ Poorly selected statistical test
- There is just **no real incentive in doing so**:
  - ↪ Legal barriers, **copyright** *Many ongoing discussions in US*
  - ↪ **Competition** issue *researchware, bibliometry, ...*
  - ↪ Publication **bias** *only the idea matters, not the gory details...*
  - ↪ Rewards for **positive/novel results**, not for *consolidating* results

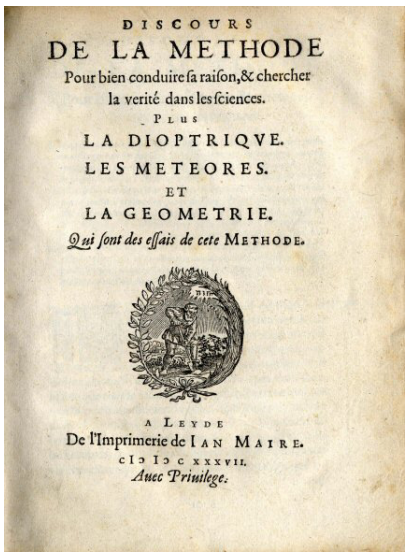
- **Technical difficulty**:
  - ↪ ~~Hardware and software evolve too quickly. It's not worth it~~
  - ↪ ~~No resources for storing so much data/information~~
  - ↪ ~~Lack of easy-to-use tools~~

Summary

Reproducible Research Movement

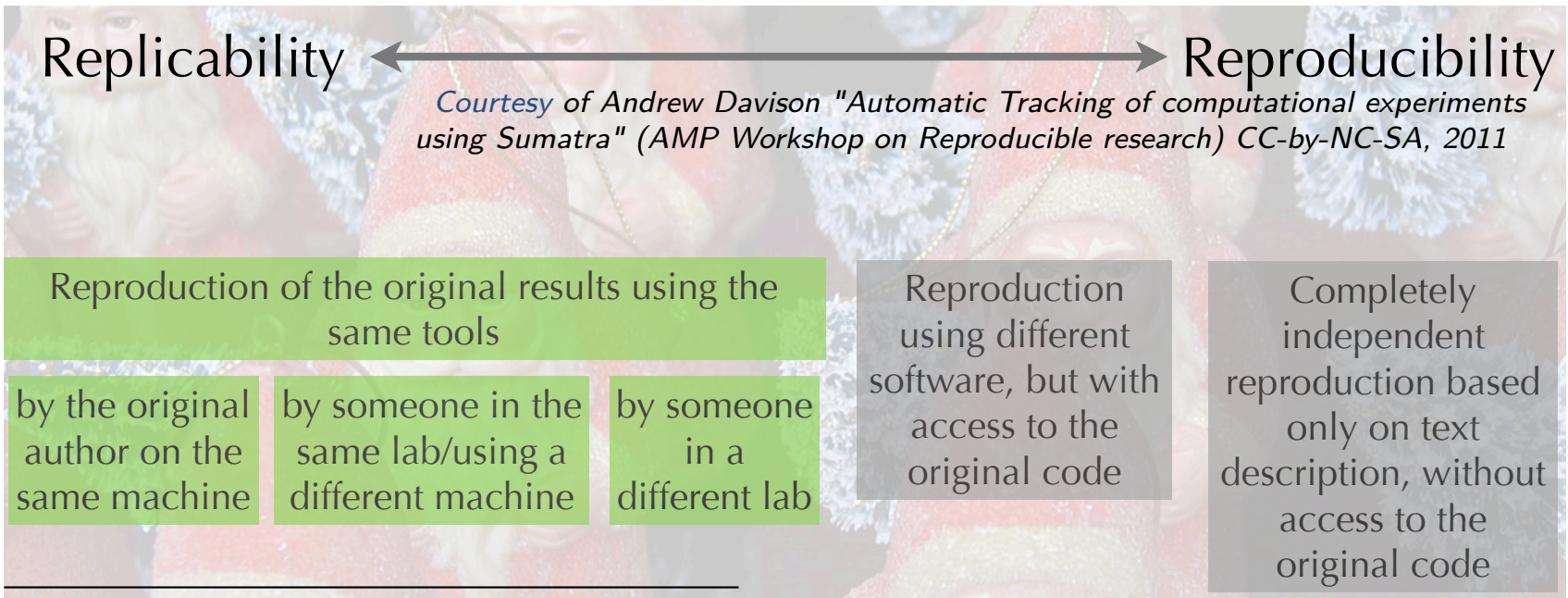
- Originated mainly in **Computational Sciences**
  - ↪ Computational biology, data-intensive physics, etc.
- Explores methods and tools to enhance experimental practices
  - ↪ Enable others to reproduce and build upon one's work

- **Nothing New**
  - ↪ Fundamental basis of the scientific method
  - ↪ K. Poppler, 1934: *non-reproducible single occurrences are of no significance to science*


DISCOURS
DE LA METHODE
Pour bien conduire sa raison,& chercher la verité dans les sciences.
PLUS
LA DIOPTRIQVE.
LES METEORES.
ET
LA GEOMETRIE.
Qui sont des essais de cette METHODE.
A LEYDE
De l'Imprimerie de IAN MAIRE.
cIↃ IↃ c XXXVII.
Avec Privilege.

Replicability vs. Reproducibility

- Terminology varies[10]
  - ↪ **Replicability** ∼ same result
  - ↪ **Reproducibity** ∼ same scientific conclusions



[10] Dror G. Feitelson. **From Repeatability to Reproducibility and Corroboration**. Technical report. http://www.cs.huji.ac.il/~feit/papers/Repeat15SIGOPS.pdf. Hebrew University of Jerusalem, 2015.

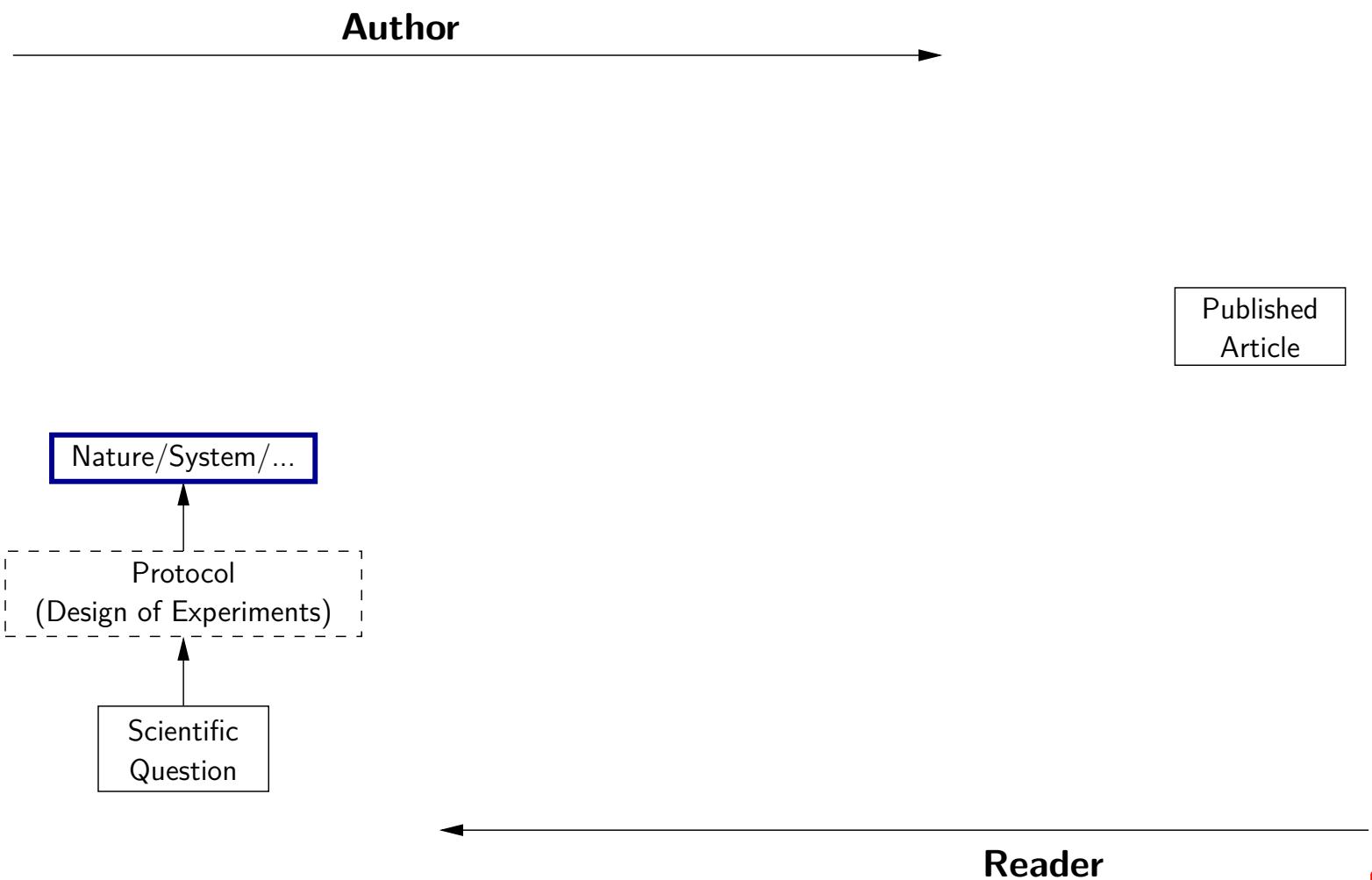
Reproducibility in Practice

Reproducibility (Wikipedia)

- the ability of an entire experiment or study to be **reproduced**,
  - ↪ either by the researcher
  - ↪ or by someone else working independently.
- One of the main principles of the scientific method.

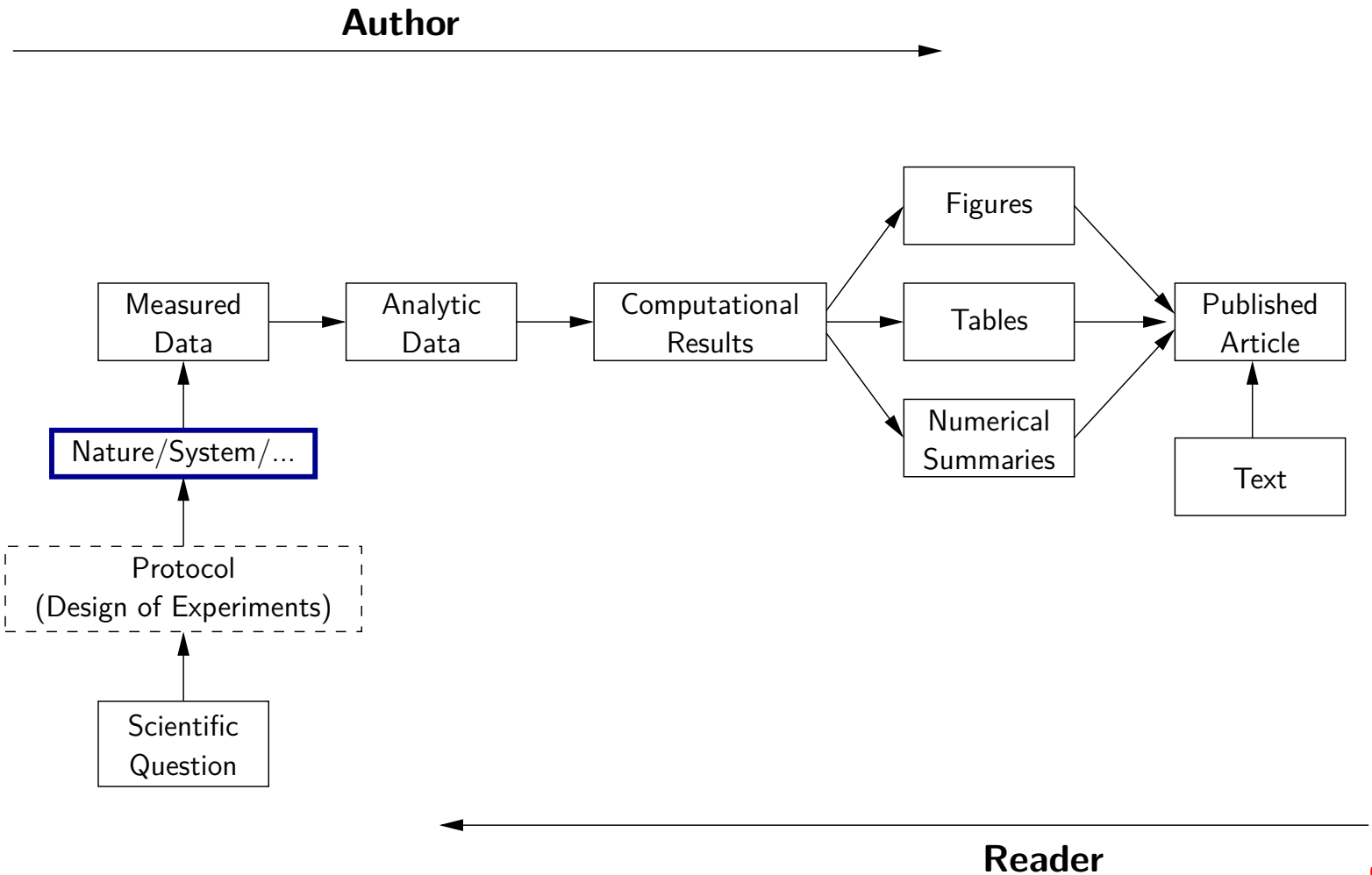
- For an experiment involving software, reproducibility means:
  - ↪ **open access** to the scientific article describing it
  - ↪ **open data** sets used in the experiment
  - ↪ **source code** of all the components
  - ↪ **environment** of execution
  - ↪ **stable references** between all this

The Research Pipeline



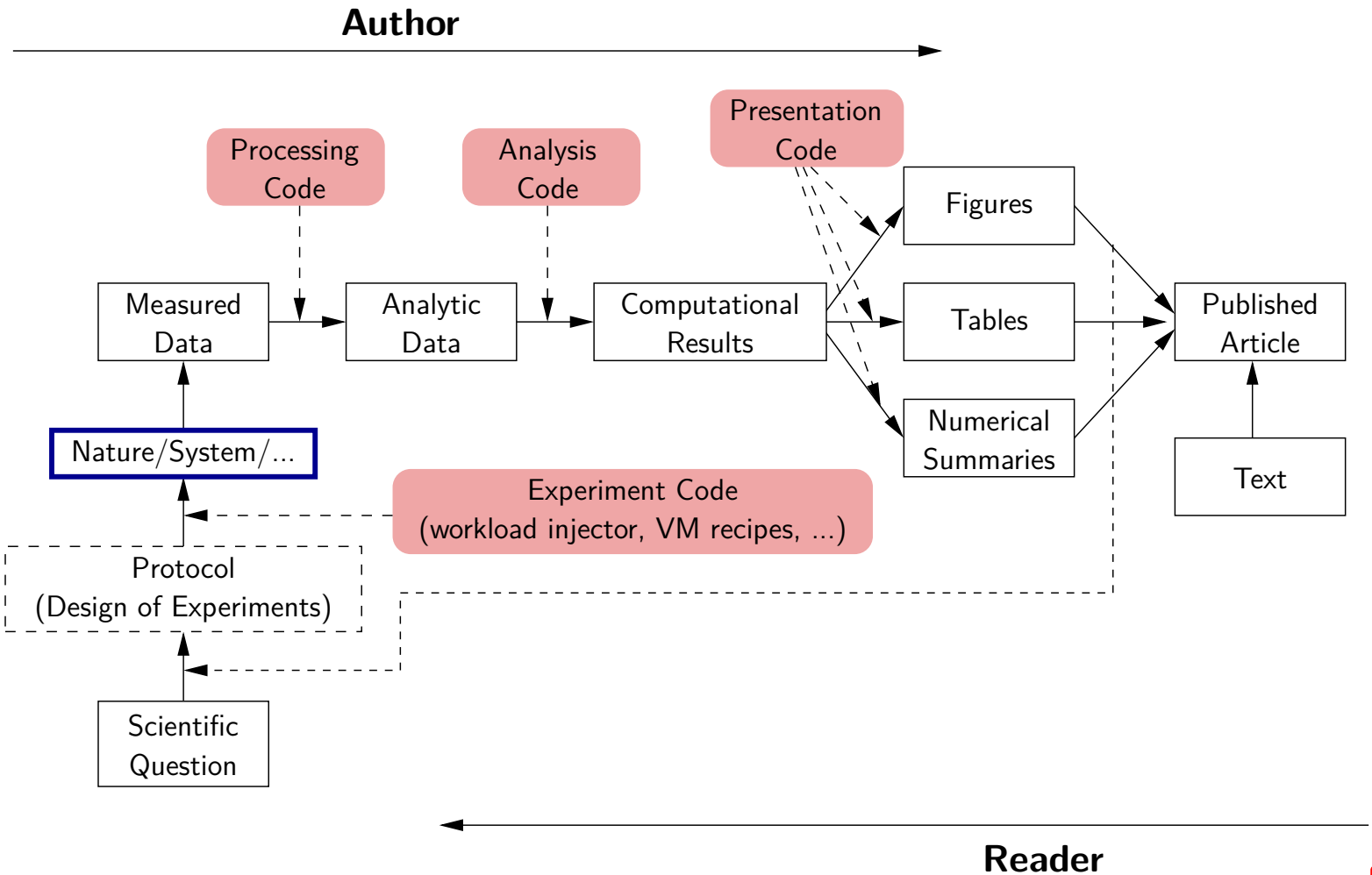
Courtesy of A. Legrand, inspired by Roger D. Peng's lecture on reproducible research, May 2014

The Research Pipeline



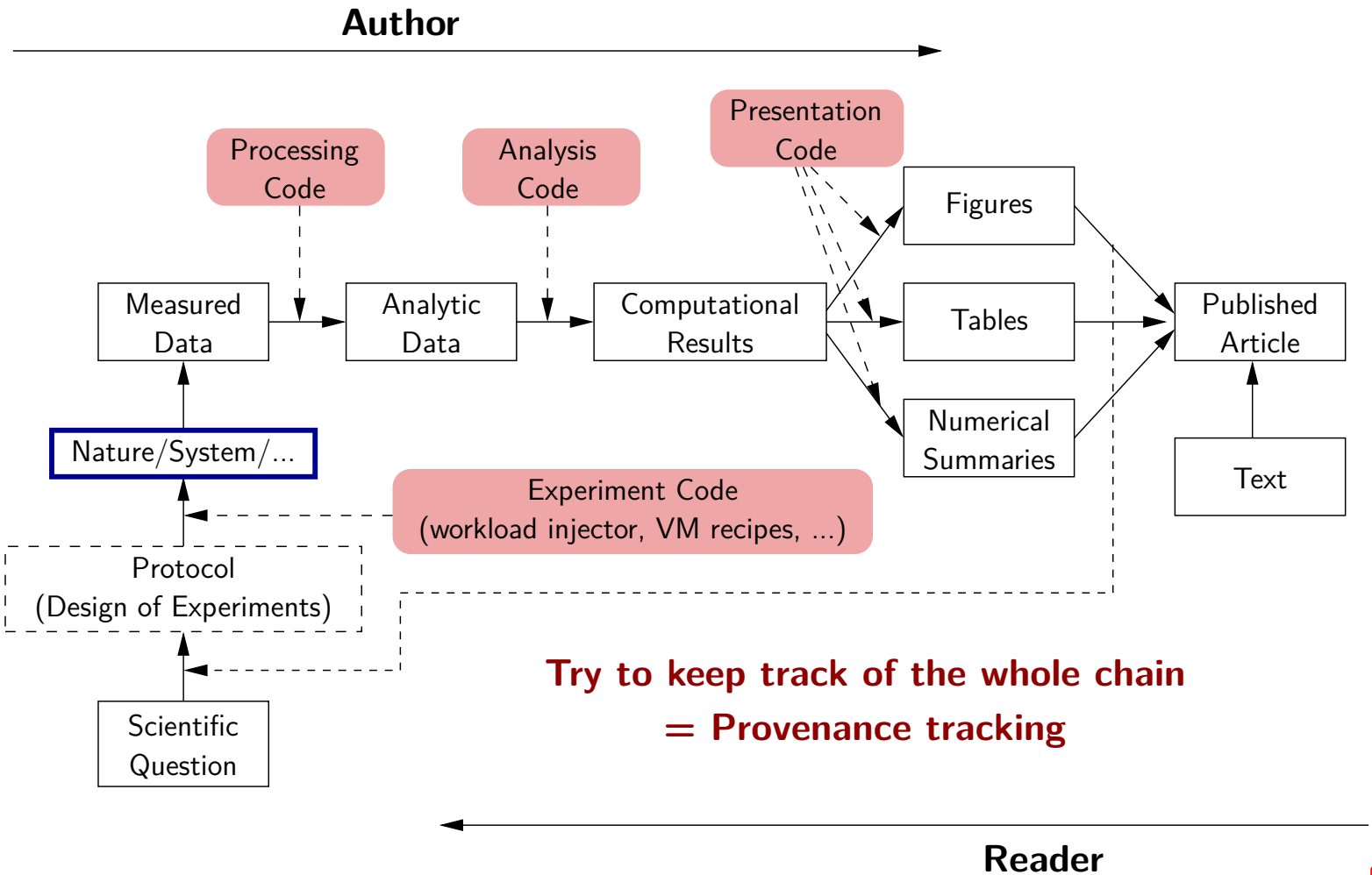
Courtesy of A. Legrand, inspired by Roger D. Peng's lecture on reproducible research, May 2014

The Research Pipeline



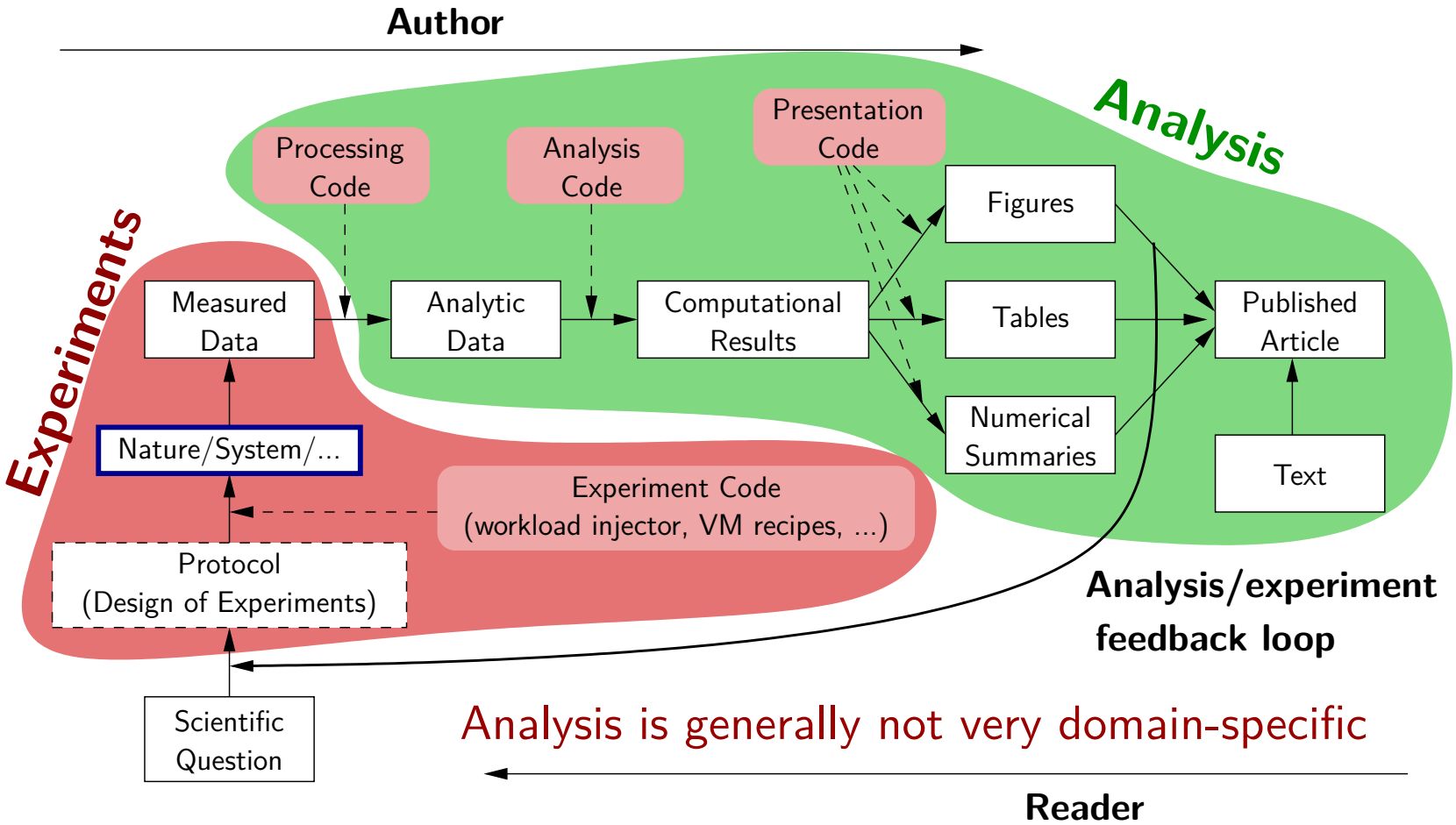
Courtesy of A. Legrand, inspired by Roger D. Peng's lecture on reproducible research, May 2014

The Research Pipeline



Courtesy of A. Legrand, inspired by Roger D. Peng's lecture on reproducible research, May 2014

The Research Pipeline



Courtesy of A. Legrand, inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research Challenges

- The **Distributed/Cloud Computing point-of-view**:
  - ↪ **Experiments** remains **the HARD part** and is very domain-specific
    - ✓ Rely on large, distributed, hybrid, prototype hardware/software
    - ✓ Measure execution times (makespans, traces, . . .)
    - ✓ Many parameters, very costly and hard to **reproduce**

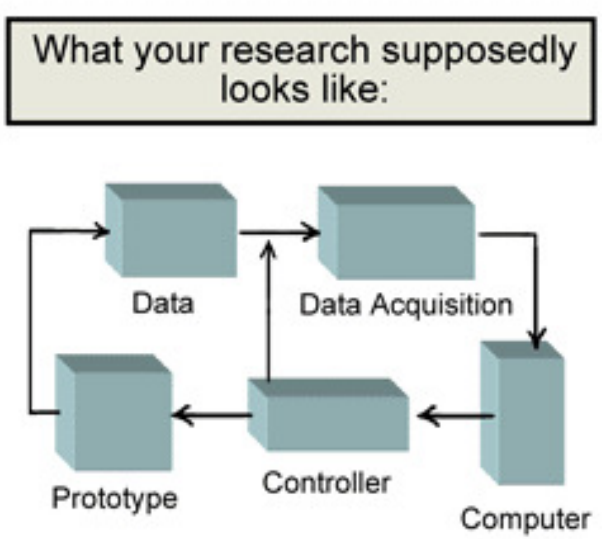


Figure 1. Experimental Diagram

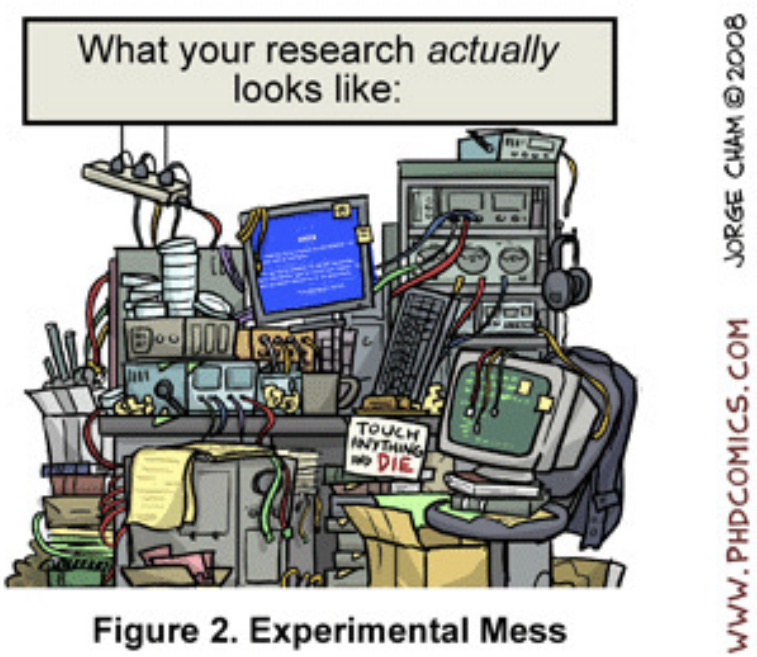


Figure 2. Experimental Mess

Environment Management

Controlling/Providing your Environment

- An **environment** is a set of tools and materials that permits a complete reproducibility of *part/whole* experiment process.

Environment Management

Controlling/Providing your Environment

- An **environment** is a set of tools and materials that permits a complete reproducibility of *part/whole* experiment process.

Q1: How to describe/provide the software environment used?

"I used OpenFOAM with OpenMPI on Debian"

- Obvious solution: **Virtual Machines**
  - ↪ Easy way to [automatically] test recipes
  - ↪ **Yet** provides **only** the final result, **not the logic behind**

RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...

RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...
- **Sharing Code and Data**
  - ↪ `git`, Github, Bitbucket, Gitlab...

RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...
- **Sharing Code and Data**
  - ↪ `git`, Github, Bitbucket, Gitlab...
- **Mastering your environment clean and automated** by:
  - ↪ Using common building tools make, cmake etc.
  - ↪ Using a constrained environment
    - ✓ Sandboxed Ruby/Python,Vagrant, Docker
  - ↪ Automate its building through cross-platform recipes
  - ↪ Automatically test your recipes for Environment configuration

RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|tal… Docs**
  - ↪ Markdown, mkdocs, org-mode, Read… …
- **Sharing Code and Data**
  - ↪ `git`, Github, Bitbucket, C…
- **Mastering your envi… clean and automated** by:
  - ↪ Using commo… tools make, cmake etc.
  - ↪ Using a co… nvironment
    - ✓ Sa… by/Python,Vagrant, Docker
  - ↪ Auto… uilding through cross-platform recipes
  - ↪ A… y test your recipes for Environment configuration

All covered in this tutorial!

Summary

Easy-to-{Read | Take | Share} Docs

- **Reproducible** research assumes accurate and organized Docs
- You need to **document** your:
  - ↪ **Hypotheses**: keep track of your ideas/line of thoughts
  - ↪ **Experiments**: details on how and why an experiment was run
    - ✓ including failed or ambiguous attempts.
  - ↪ **Initial analysis or interpretation** of these experiments
    - ✓ was the outcome conform to the expectation or not?
    - ✓ does it (in)validate the hypothesis?
  - ↪ **Organization**: keep track of things to do/ x/test/improve
- **Stucture**:
  - ↪ General information about the document
  - ↪ **commonly used commands** and how to set up experiments
  - ↪ Experiment results
    - ✓ by **date** (`tags`)
    - ✓ by **experiment campaigns** (`date/time`)

Documentation Tools / Format

Recommandation

- Plain-text with Markdown syntax
  - ↪ Easy to **track over Git** (`text` files, **not** Word/RFT etc.)
  - ↪ Easy to **export** to any format using `pandoc` / `multimarkdown`
  - ↪ **Supports online/offline** Wikis / Blogging platforms

- **Focus on writing**, viewers for all platform
  - ↪ Mac OS: MOU, Marked 2
  - ↪ Linux: Remarkable, Retext
  - ↪ Windows: MarkdownPad, Remarkable
- Git Based Markdown Blogging
  - ↪ Octopress, Jekyll

Git-based Markdown Wiki

- **Permits to work offline**
  - ↪ Gollum, as embedded in GitLab
    - ✓ run `gollum` (from root directory) `http://localhost:4567`

Recommandation: MkDocs

`http://www.mkdocs.org/`

- Better for **Hierarchical structure** of the docs
  - ↪ fully configured by `mkdocs.yml` and files in `docs/`
  - ↪ local [interpreted] site: `mkdocs serve` (from root directory)
    `http://localhost:8000`
- compliant with Read the Docs
  - ↪ trigger **automatic doc rebuild** upon [git] push
  - ↪ cf `http://rr-tutorials.readthedocs.io/` ☺

Mkdocs Workflow

```
$> mkdocs new     # initialize 'mkdocs.yml' and docs/ directory
```

Mkdocs Workflow

```
$> mkdocs new     # initialize 'mkdocs.yml' and docs/ directory
```

```
# mkdocs.yml -- MkDocs configuration, all *.md files relative to docs/
site_name: My Environment Documentation
pages:
- Home: 'index.md'
- Tools:
    - SSH: 'tools/ssh.md'
    - Git: 'tools/git.md'
- Configuration:
    - CA Certificates: 'config/certificates/README.md'
theme: readthedocs
```

Mkdocs Workflow

```
$> mkdocs new     # initialize 'mkdocs.yml' and docs/ directory
```

```
# mkdocs.yml -- MkDocs configuration, all *.md files relative to docs/
site_name: My Environment Documentation
pages:
- Home: 'index.md'
- Tools:
    - SSH: 'tools/ssh.md'
    - Git: 'tools/git.md'
- Configuration:
    - CA Certificates: 'config/certificates/README.md'
theme: readthedocs
```

```
$> mkdocs serve     # Run LOCAL builtin server http://localhost:8000
```

Hands-On 1: Markdown & MkDocs

Your Turn!

http://rr-tutorials.readthedocs.io/en/latest/hands-on/docs/

- **Easy-to-{Read | Take | Share} Docs with MkDocs**
  - ↪ installation of MkDocs — http://www.mkdocs.org/#installation
  - ↪ initialization — `mkdocs new .`
  - ↪ Markdown basis
  - ↪ **Local** serve — `mkdocs serve`

Summary

Sharing Code and Data

What kinds of systems are available?

- *Good*: The cloud — Dropbox, Google Drive, Figshare...
- **Better** - **Version Control systems** (VCS)
  - ↪ SVN, Git and Mercurial
- **Best** - **Version Control Systems** on the **Public/Private Cloud**
  - ↪ GitHub, Bitbucket, Gitlab

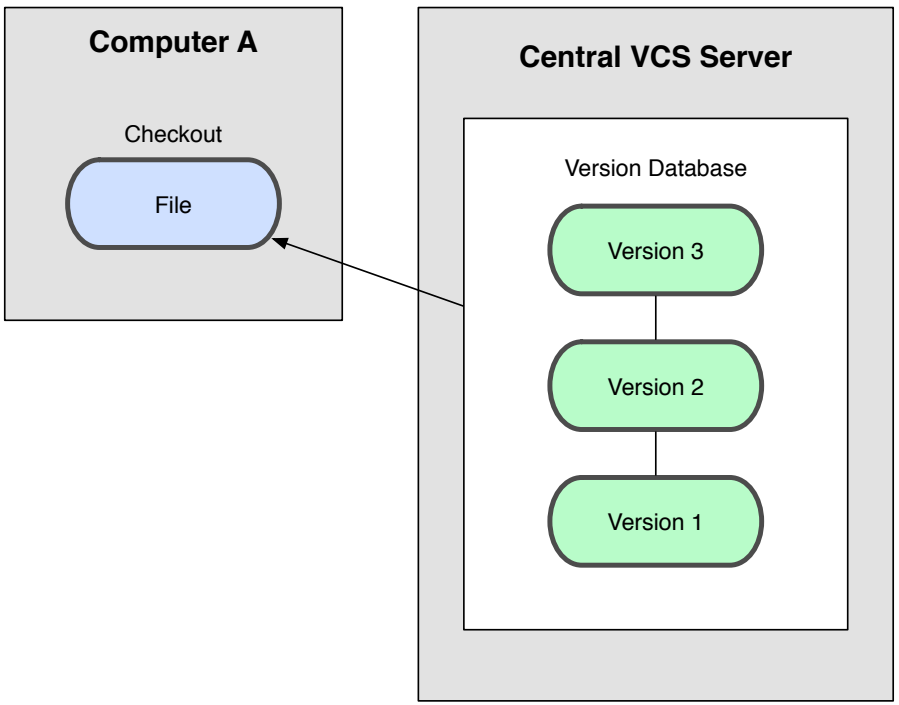
Sharing Code and Data

What kinds of systems are available?

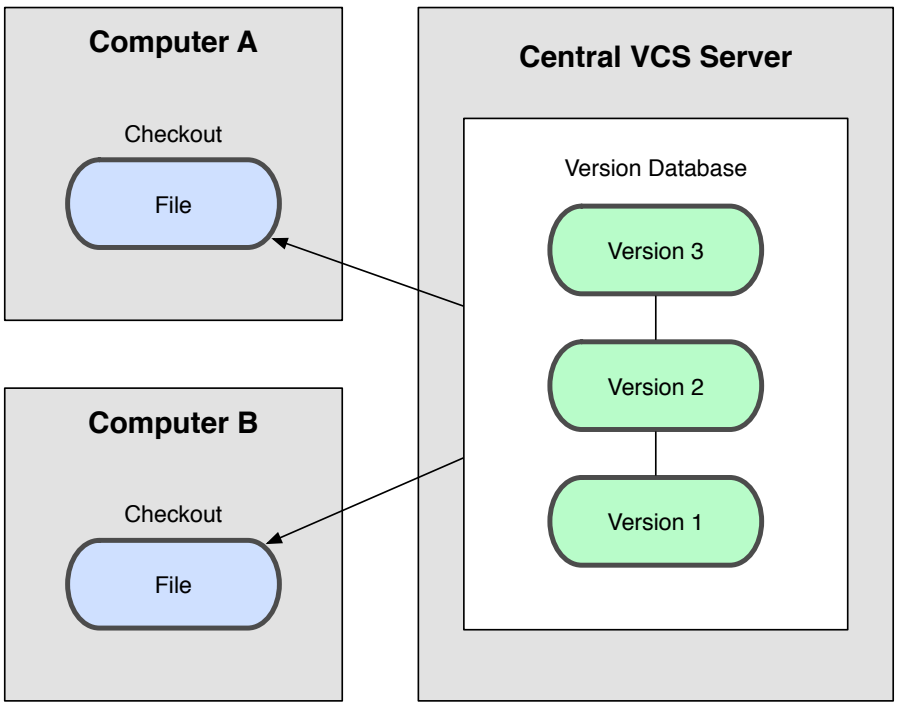
- *Good*: The cloud — Dropbox, Google Drive, Figshare...
- **Better** - **Version Control systems** (VCS)
  - ↪ SVN, Git and Mercurial
- **Best** - **Version Control Systems** on the **Public/Private Cloud**
  - ↪ GitHub, Bitbucket, Gitlab

- **Which one?**
  - ↪ Depends on the level of privacy you expect
    - ✓ ... but you probably already know these tools ☺
  - ↪ **Few handle GB files...**

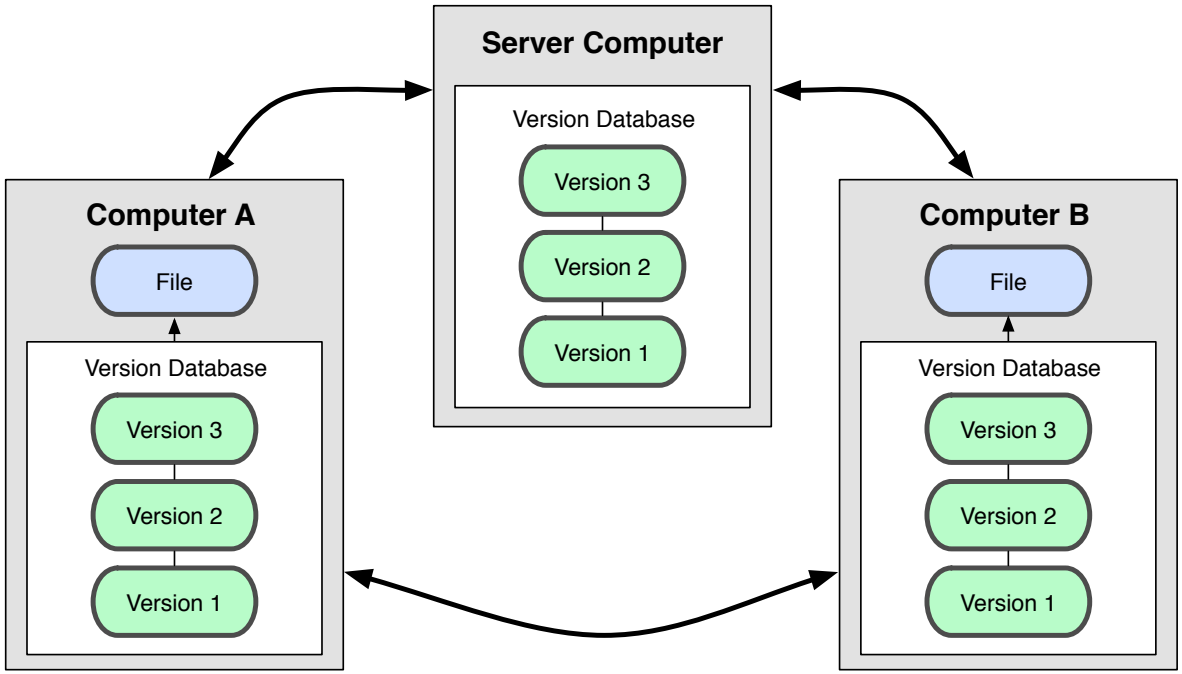
Centralized VCS – CVS, SVN



Centralized VCS – CVS, SVN

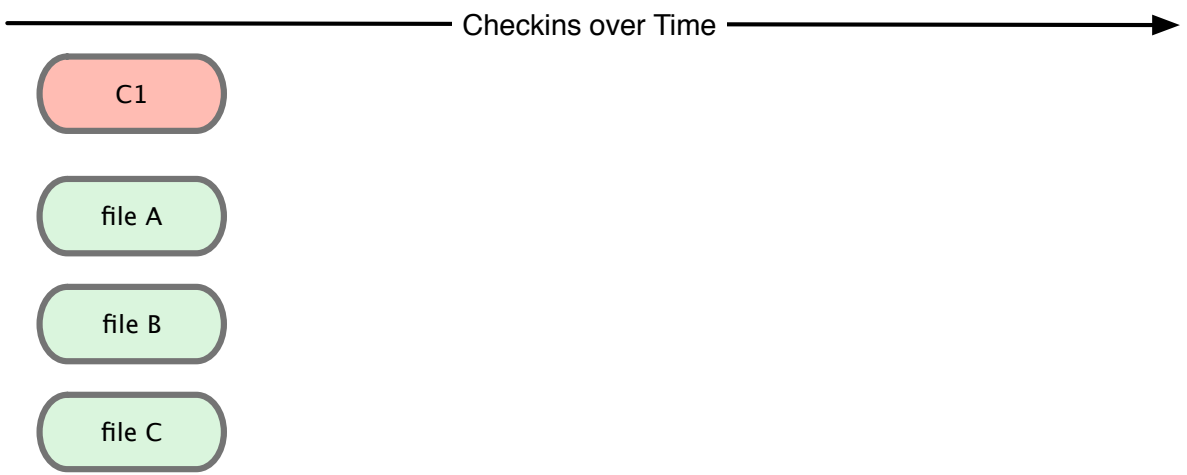


Distributed VCS – Git

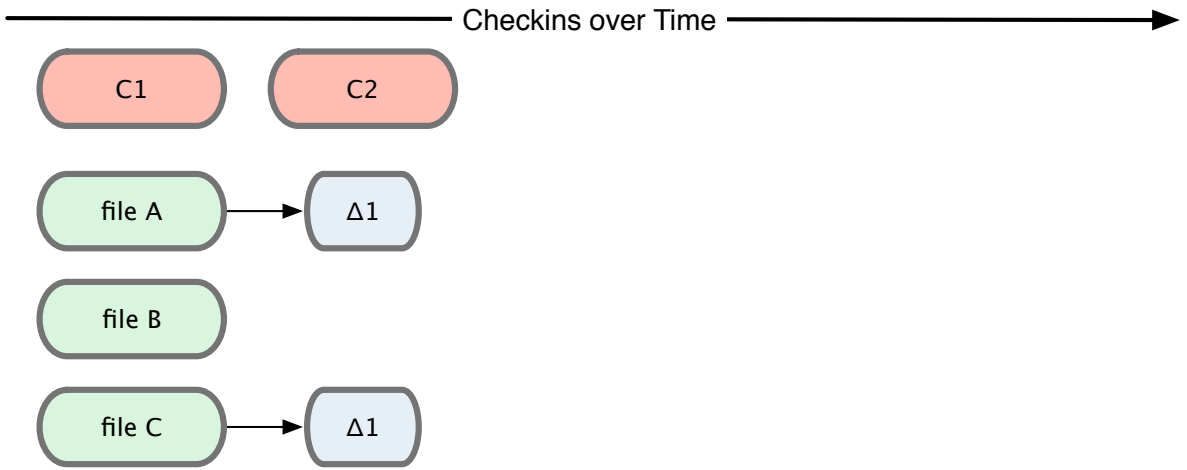


Everybody has the full history of commits

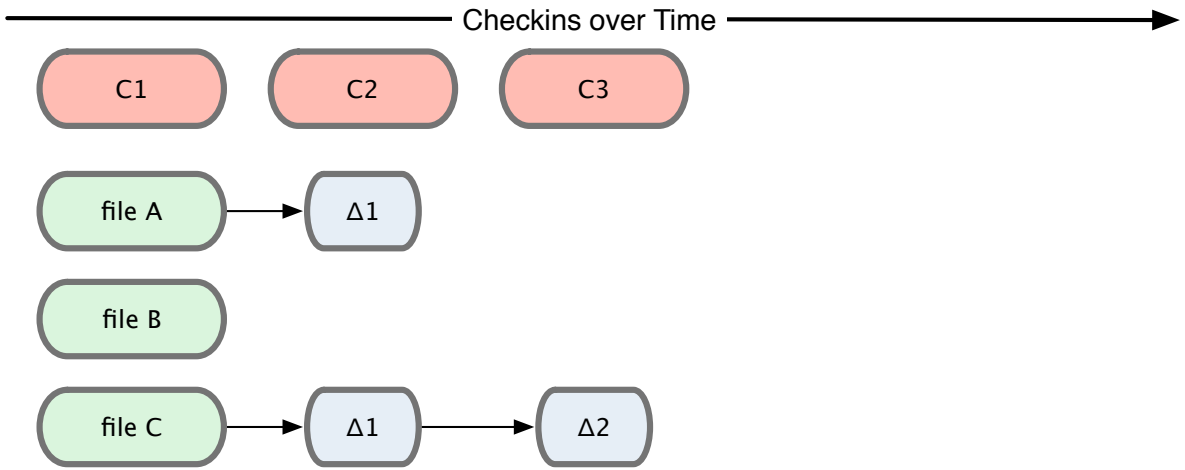
Tracking changes (most VCS)



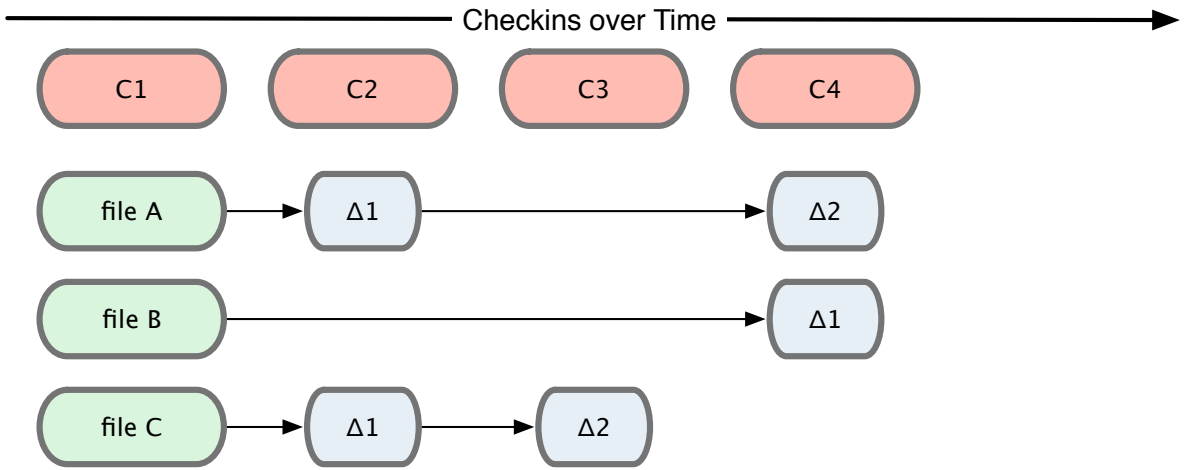
Tracking changes (most VCS)



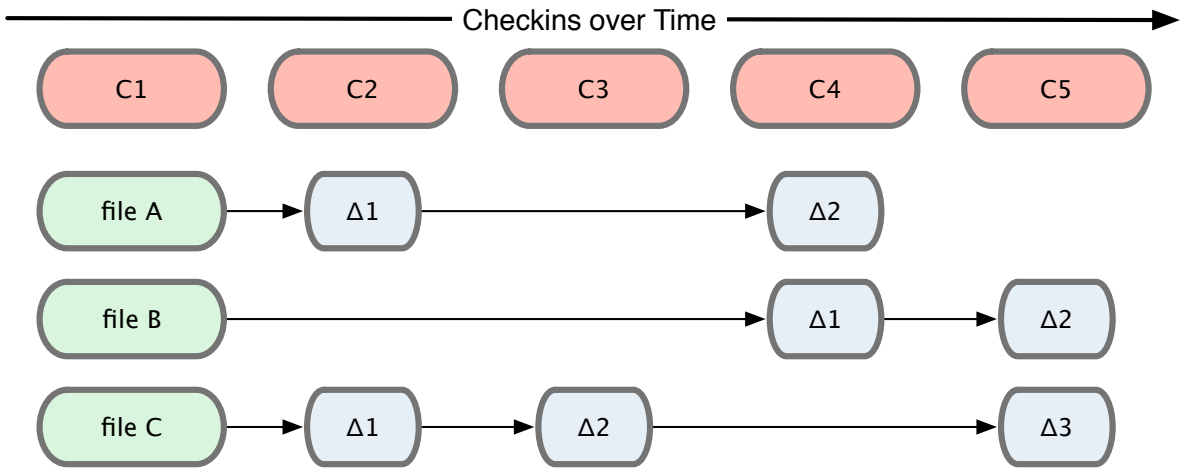
Tracking changes (most VCS)



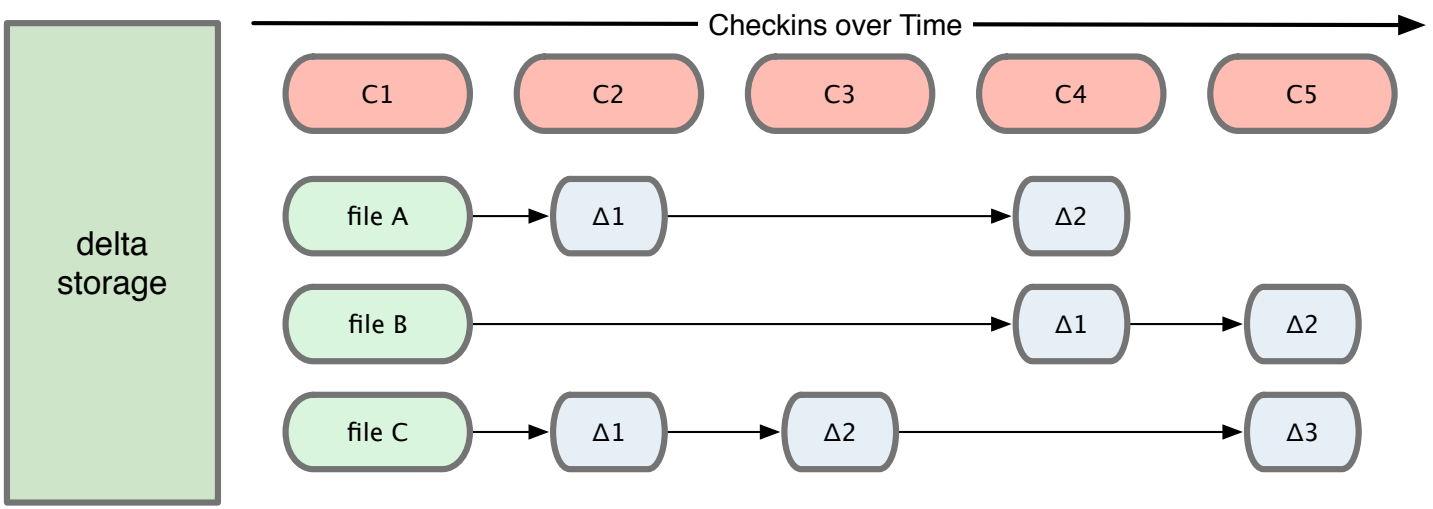
Tracking changes (most VCS)



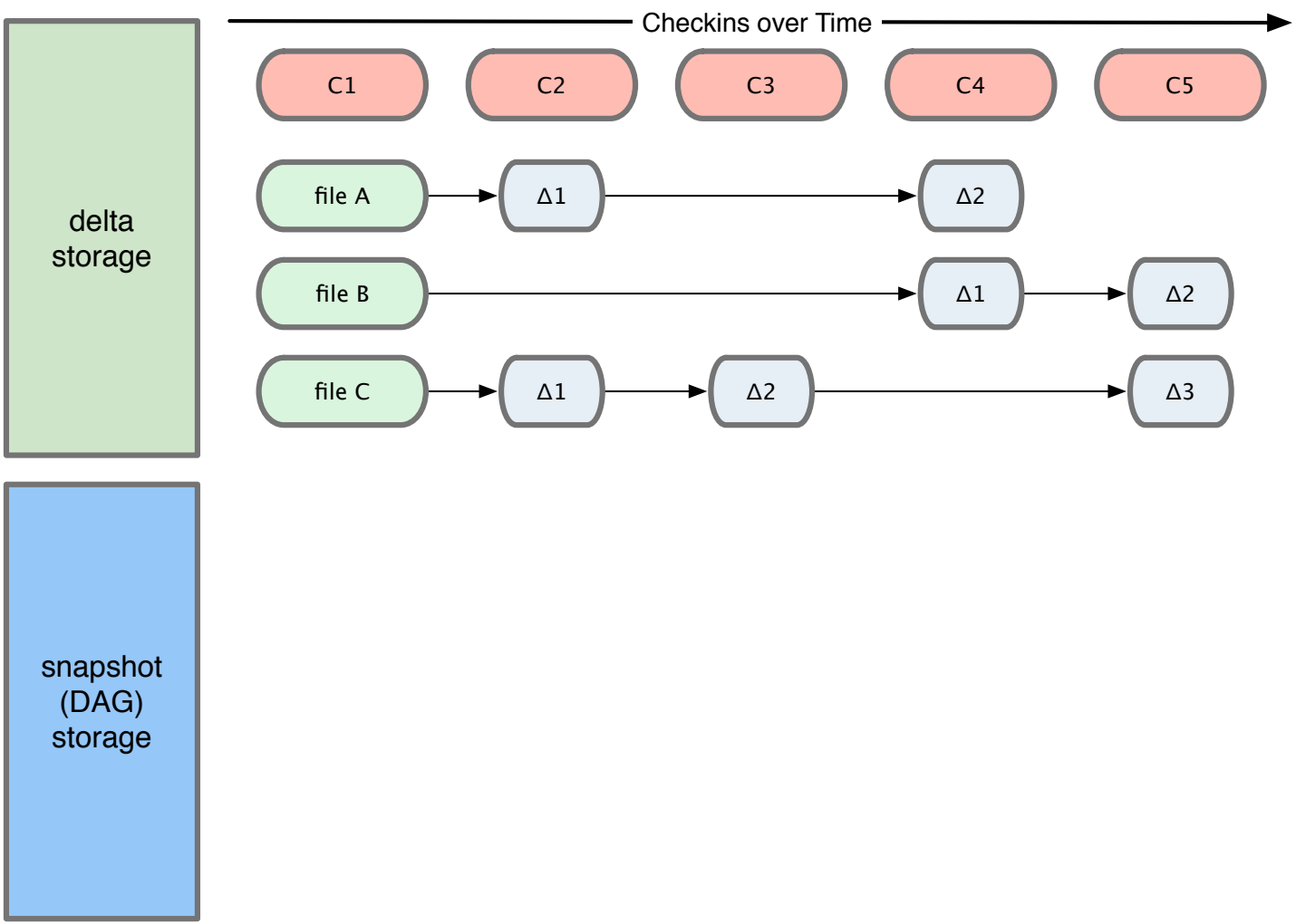
Tracking changes (most VCS)



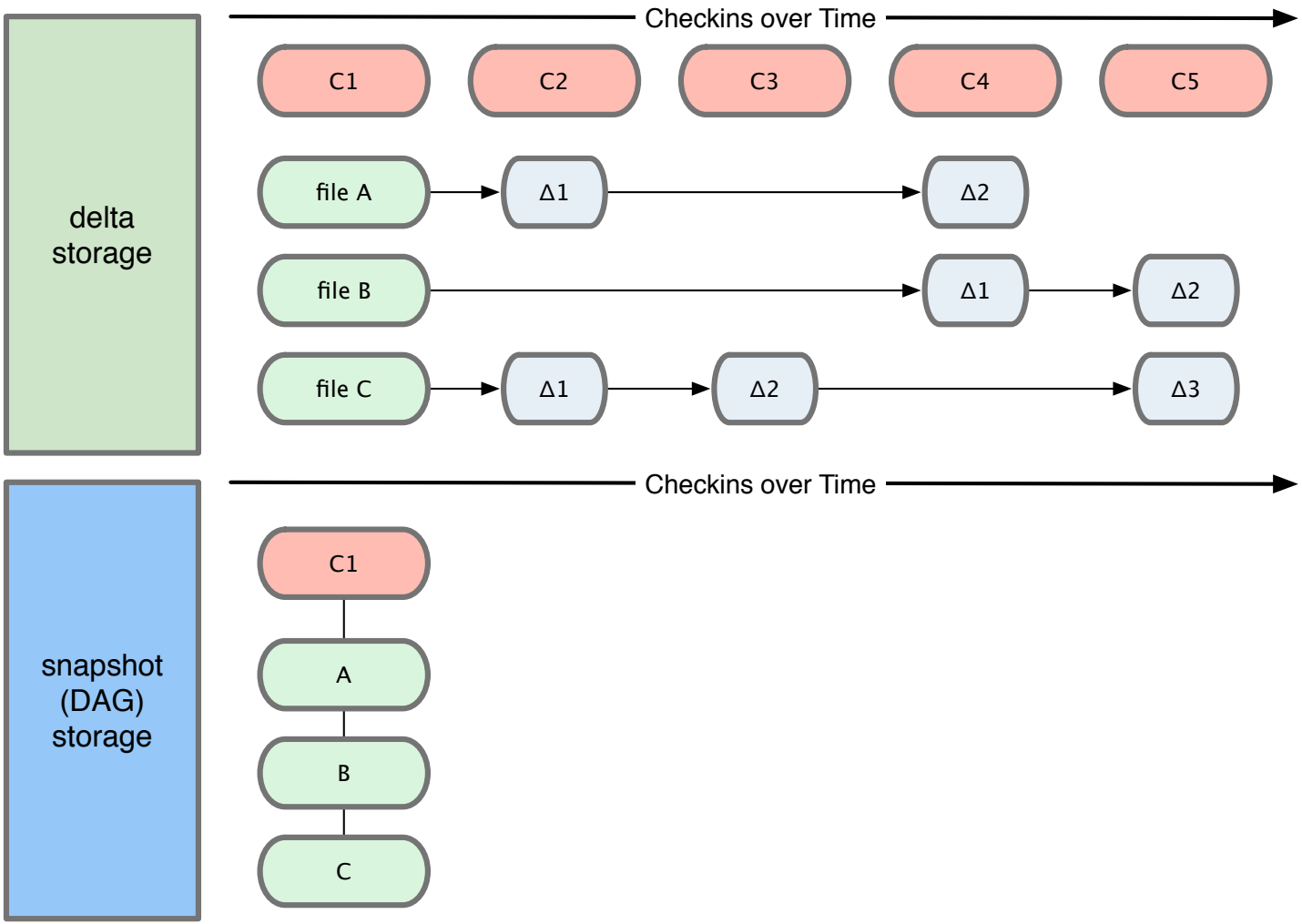
Tracking changes (most VCS)



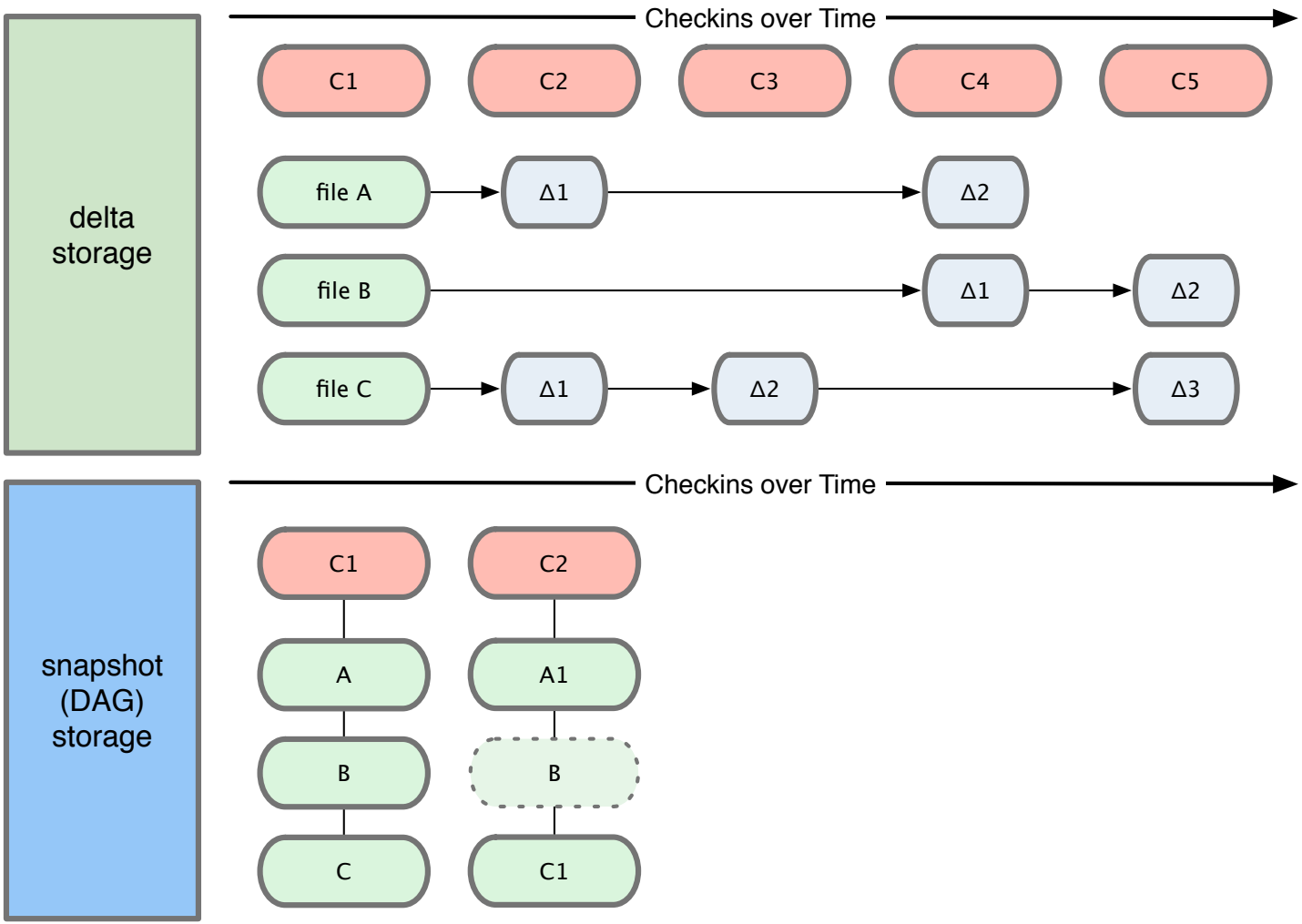
Tracking changes (Git)



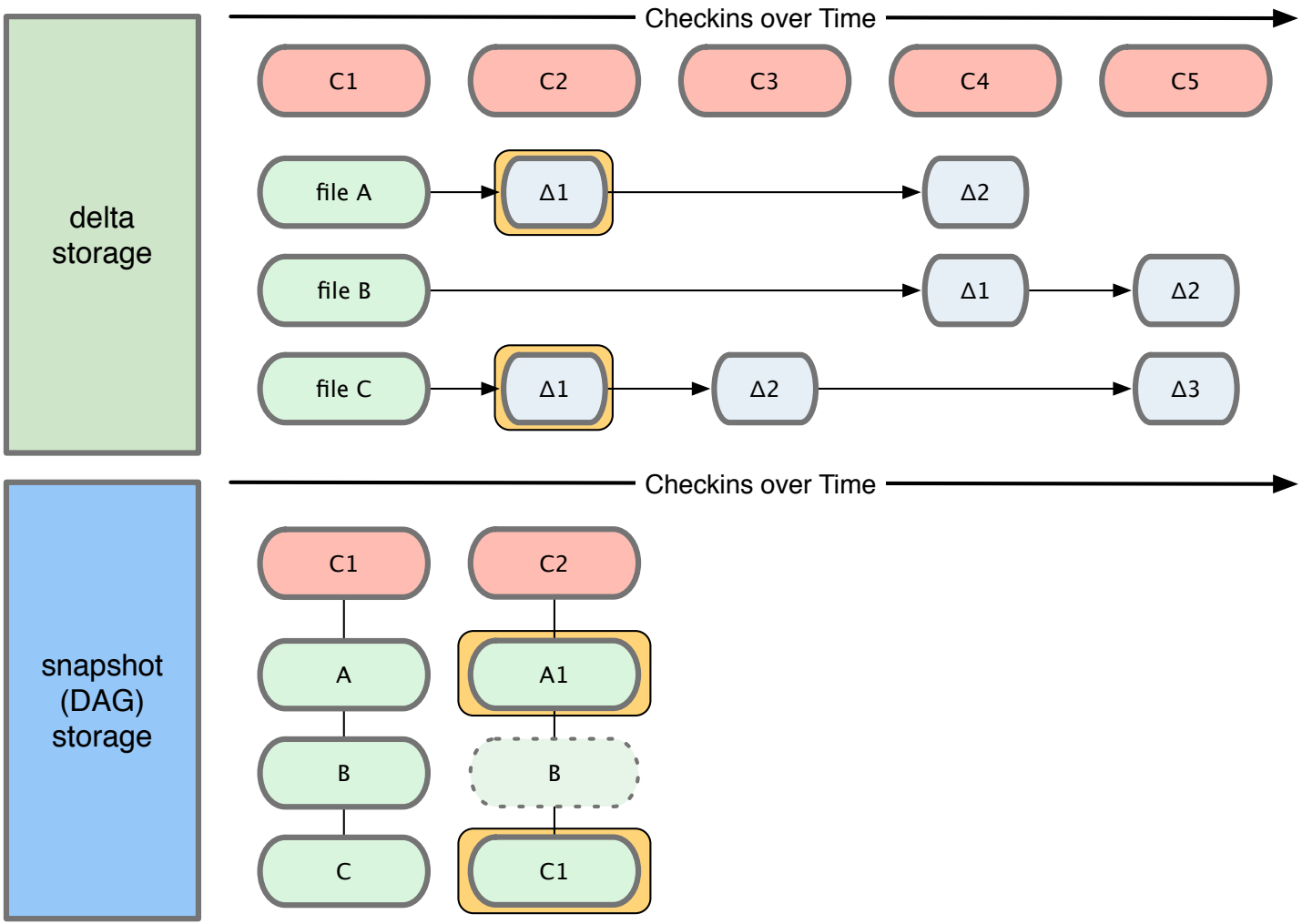
Tracking changes (Git)



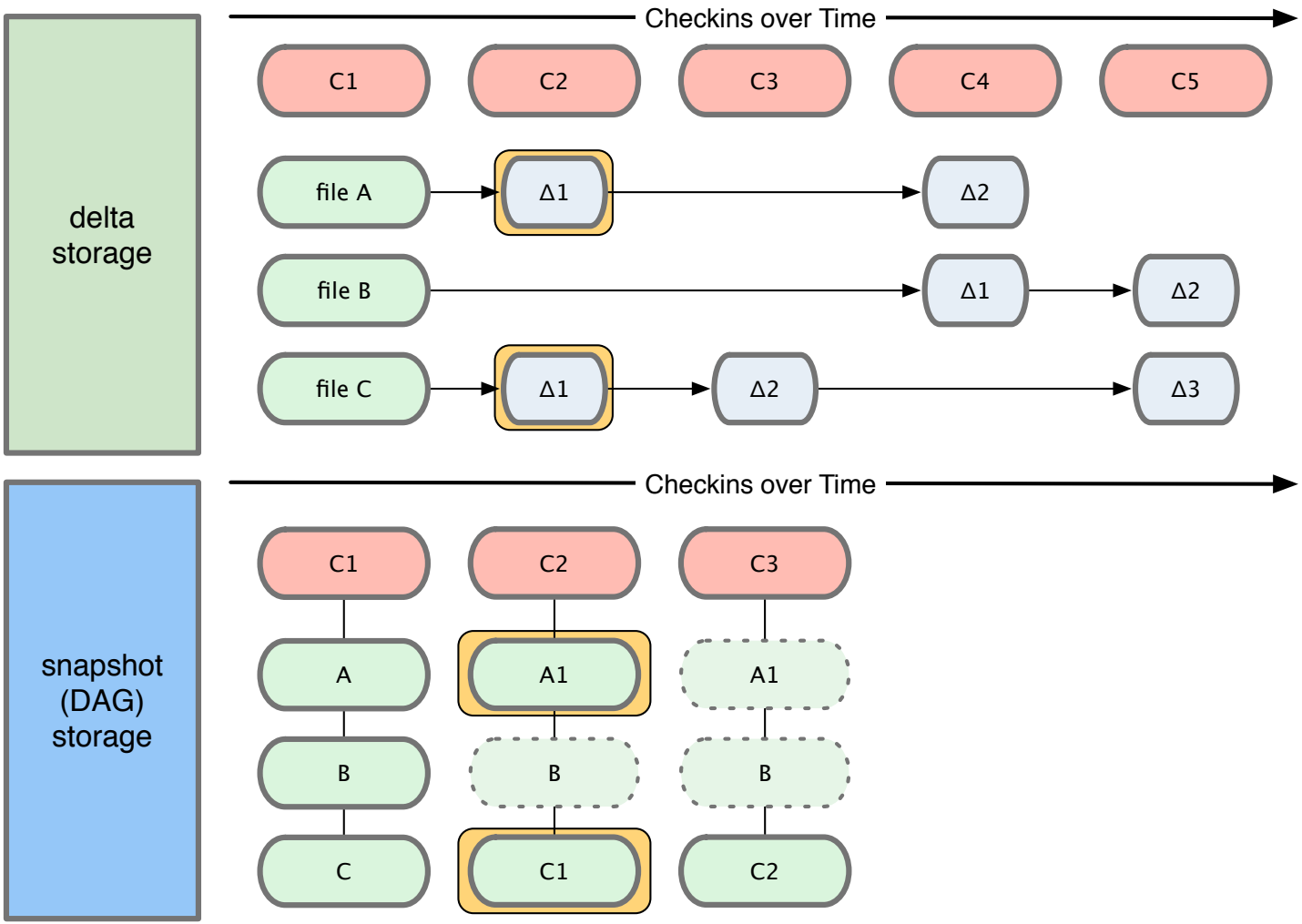
Tracking changes (Git)



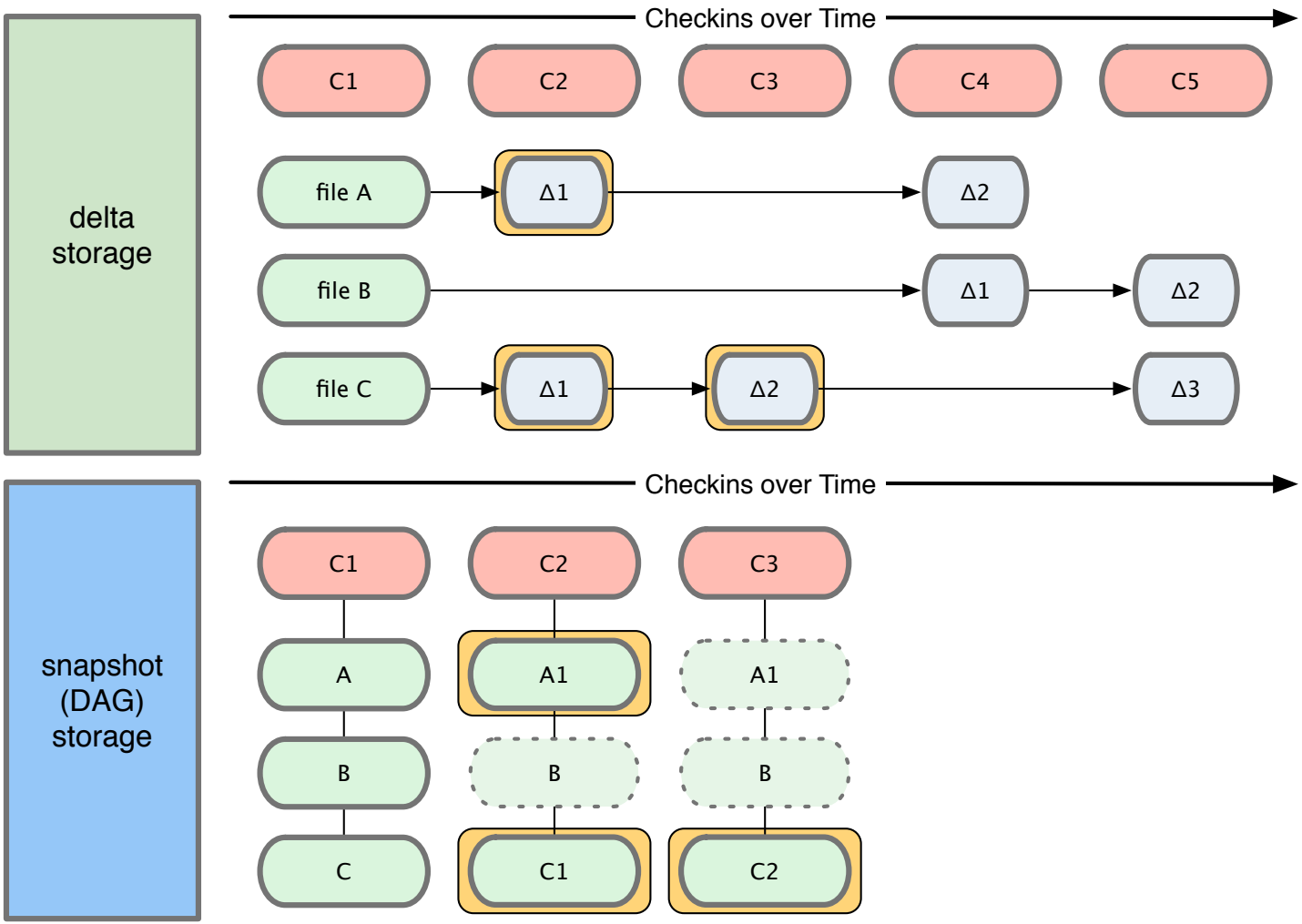
Tracking changes (Git)



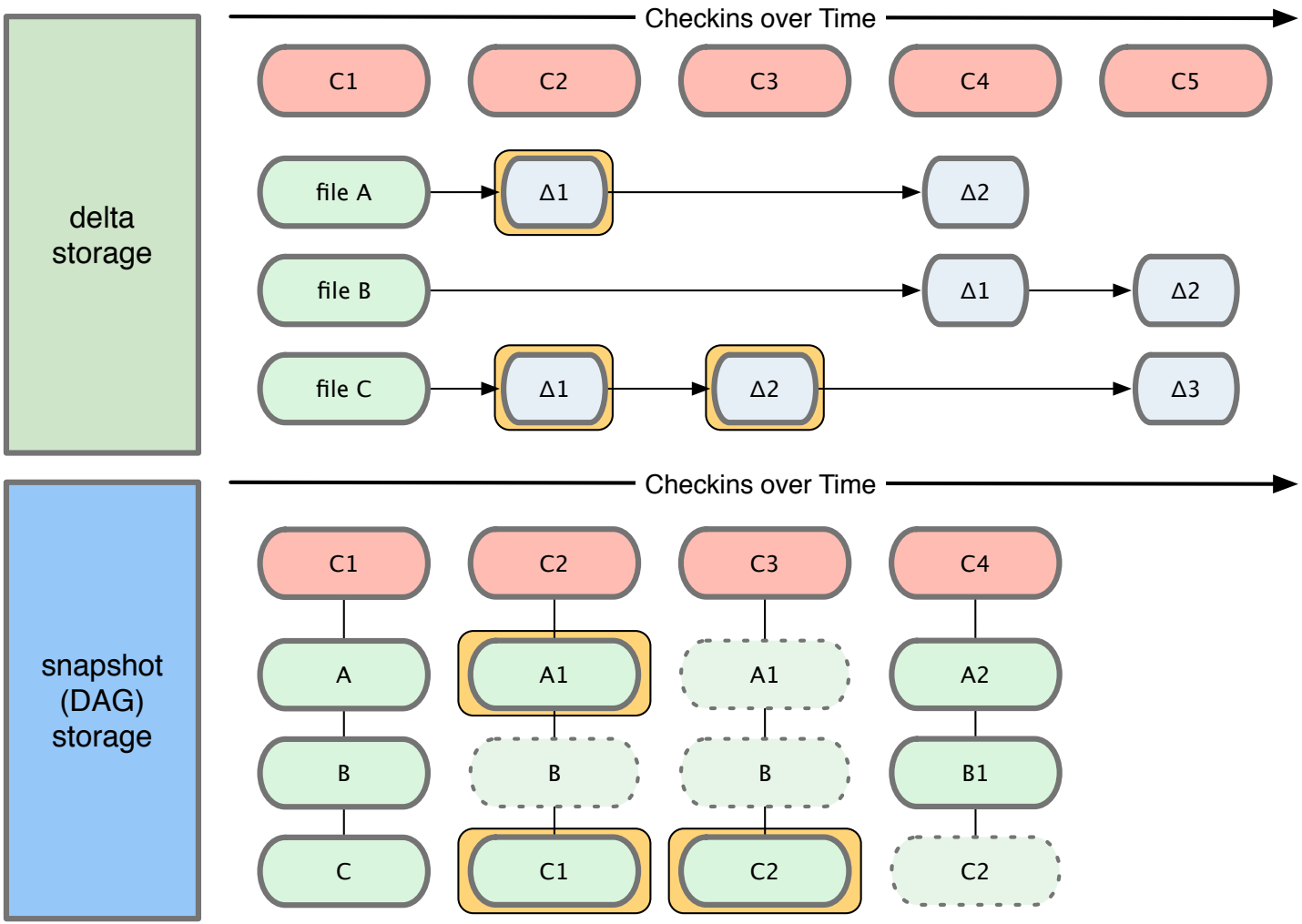
Tracking changes (Git)



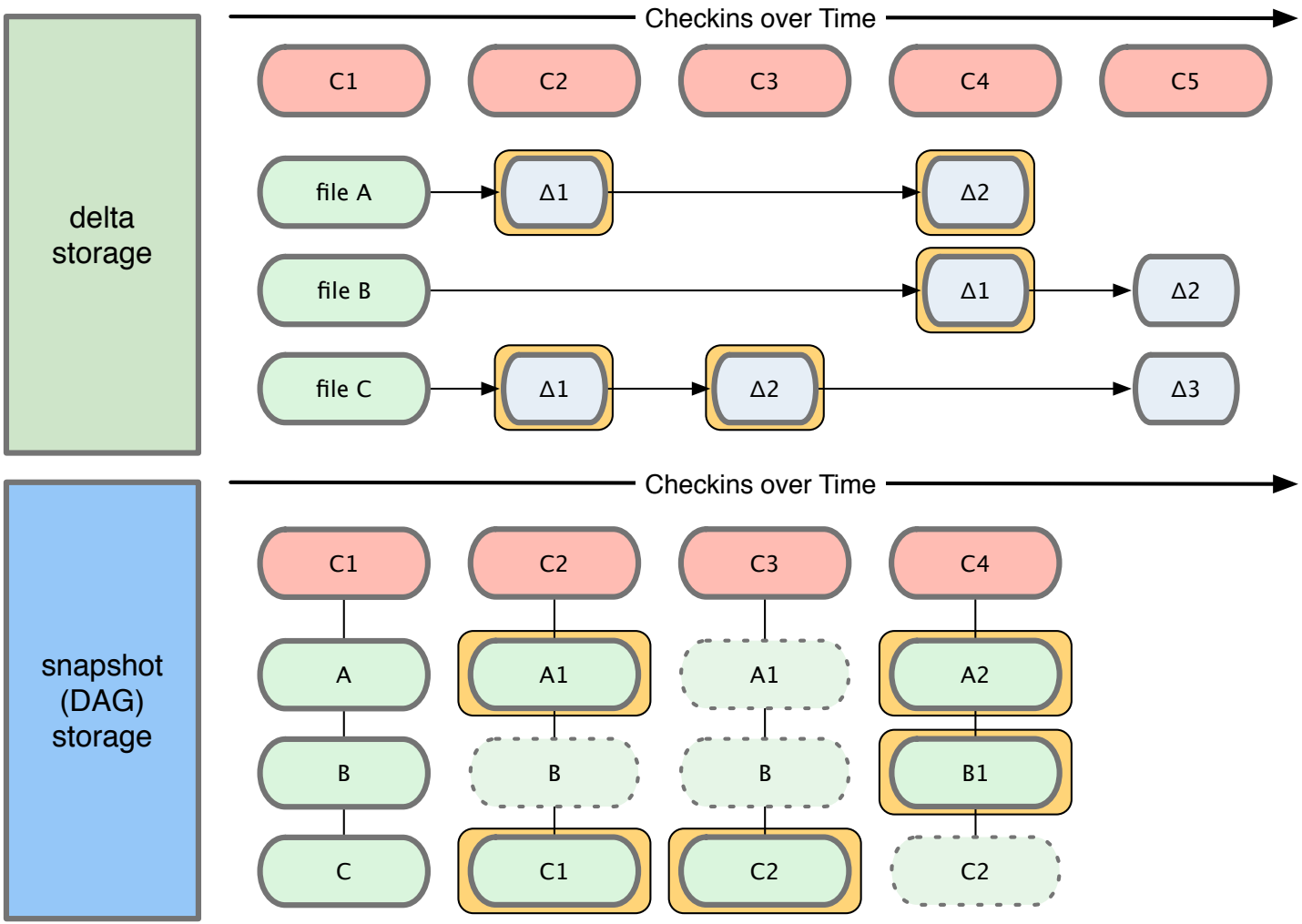
Tracking changes (Git)



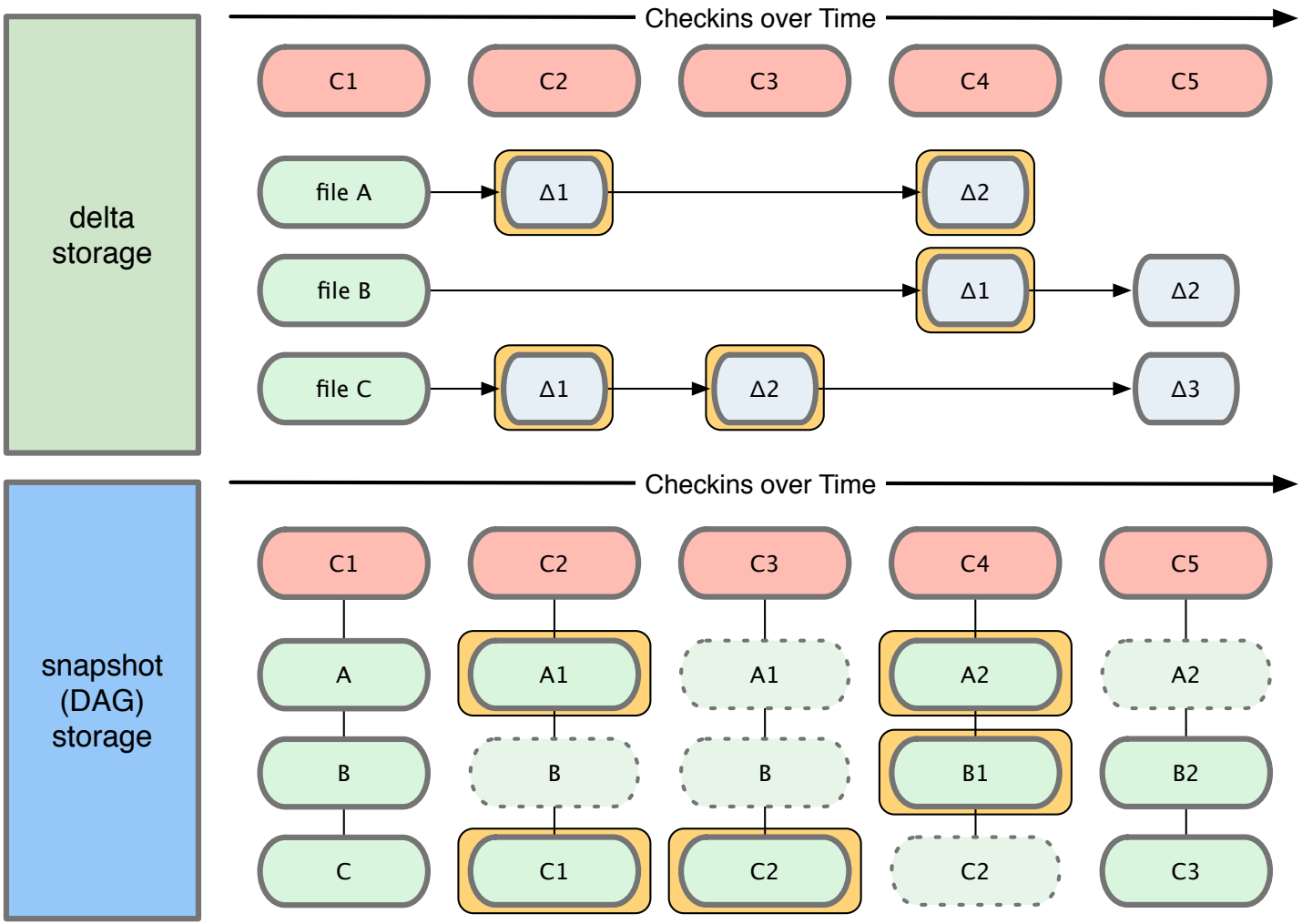
Tracking changes (Git)



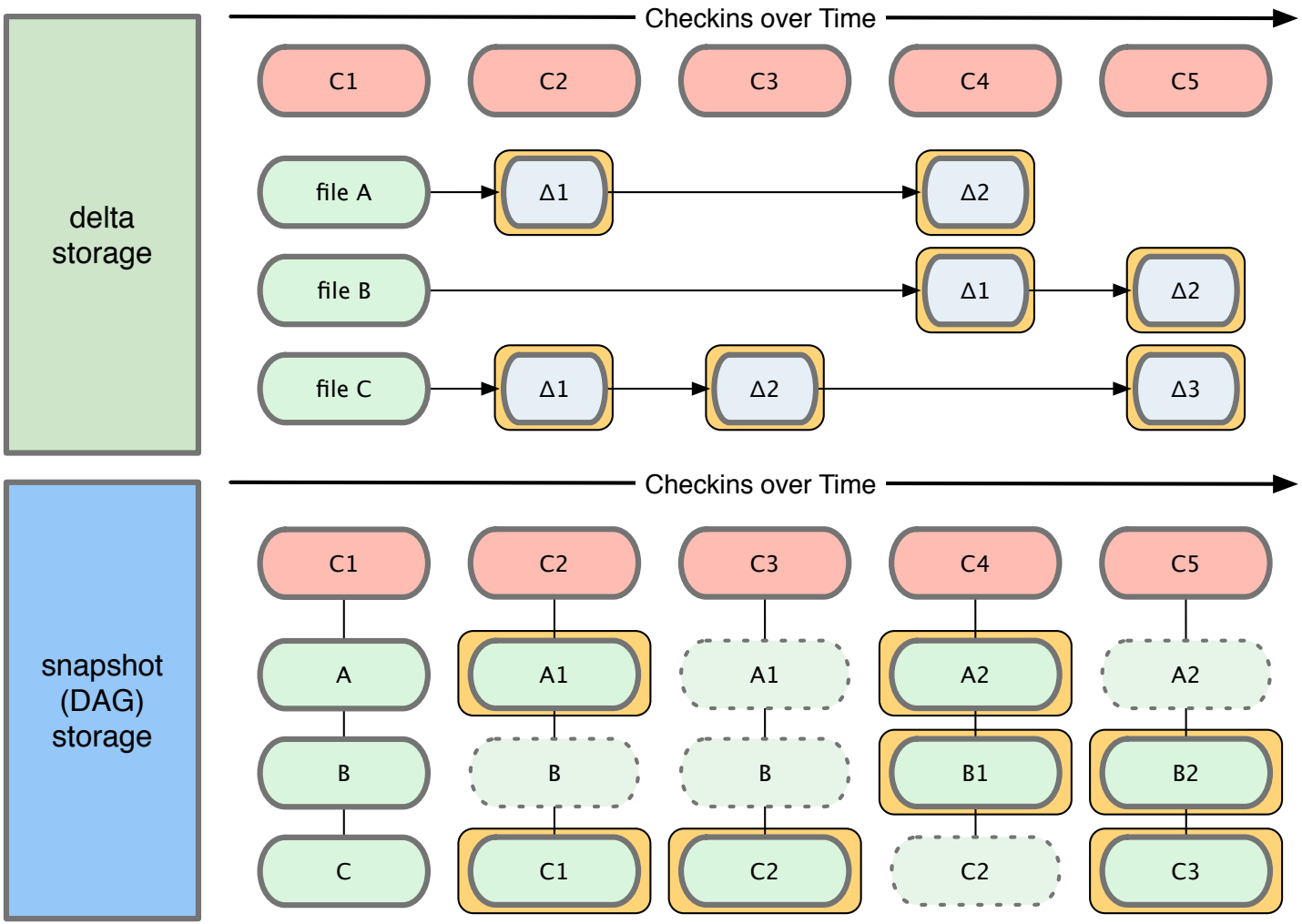
Tracking changes (Git)



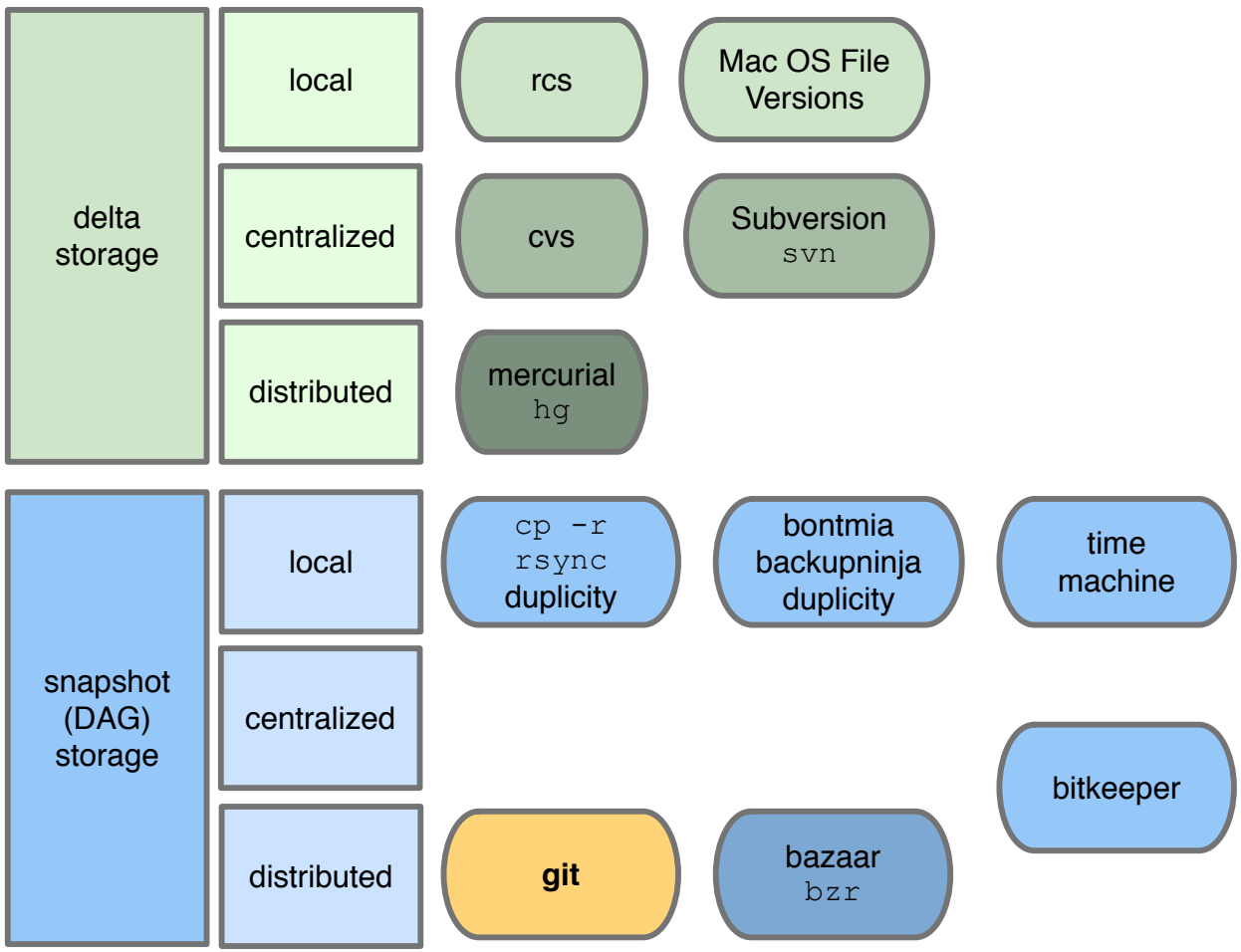
Tracking changes (Git)



Tracking changes (Git)



VCS Taxonomy



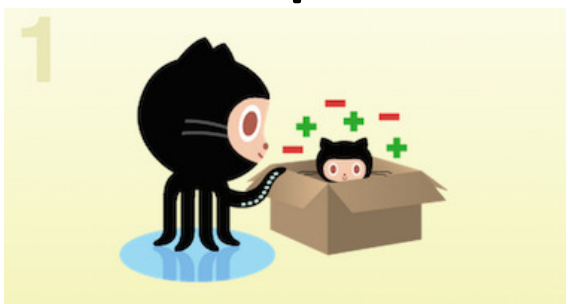
Git at the heart of RR

http://git-scm.org

Git on the Cloud: Github `github.com`

(Reference) web-based Git repository hosting service

Set up Git



Create Repository

Fork repository

Work together

So what makes Git so useful?

(almost) Everything is local

- everything is fast
- every clone is a backup
- you work **mainly offline**

Ultra Fast, Efficient & Robust

- Snapshots, not patches (deltas)
- **Cheap branching and merging**
  - ↪ Strong support for thousands of parallel branches
- Cryptographic integrity everywhere

Other Git features

- **Git doesn't delete**
  - ↪ **Immutable** objects, Git generally only adds data
  - ↪ If you mess up, you can usually recover your stuff
    - ✓ Recovery can be tricky though

Other Git features

- **Git doesn't delete**
  - ↪ **Immutable** objects, Git generally only adds data
  - ↪ If you mess up, you can usually recover your stuff
    - ✓ Recovery can be tricky though

Git Tools / Extension

- cf. **Git submodules** or **subtrees**
- **Introducing git-flow**
  - ↪ workflow with a strict branching model
  - ↪ offers the `git` commands to follow the workflow

```
$> git flow init
$> git flow feature { start, publish, finish } <name>
$> git flow release { start, publish, finish } <version>
```

Hands-on 2: Practical Git http://git-scm.com/downloads

Installation on Linux / Mac OS

```
$> apt-get install git-core git-flow          # On Debian-like systems
$> yum install git gitflow                    # On CentOS-like systems
$> brew install git git-flow                  # On Mac OS, using Homebrew
```

Hands-on 2: Practical Git http://git-scm.com/downloads

Installation on Linux / Mac OS

```
$> apt-get install git-core git-flow        # On Debian-like systems
$> yum install git gitflow                  # On CentOS-like systems
$> brew install git git-flow                # On Mac OS, using Homebrew
```

Installation on Windows MsysGit

- Incl. Git Bash/GUI & Shell Integration
  - ↪ install **Git bash** + command prompt
  - ↪ select checkout windows / commit unix

Hands-on 2: Practical Git http://git-scm.com/downloads

Installation on Linux / Mac OS

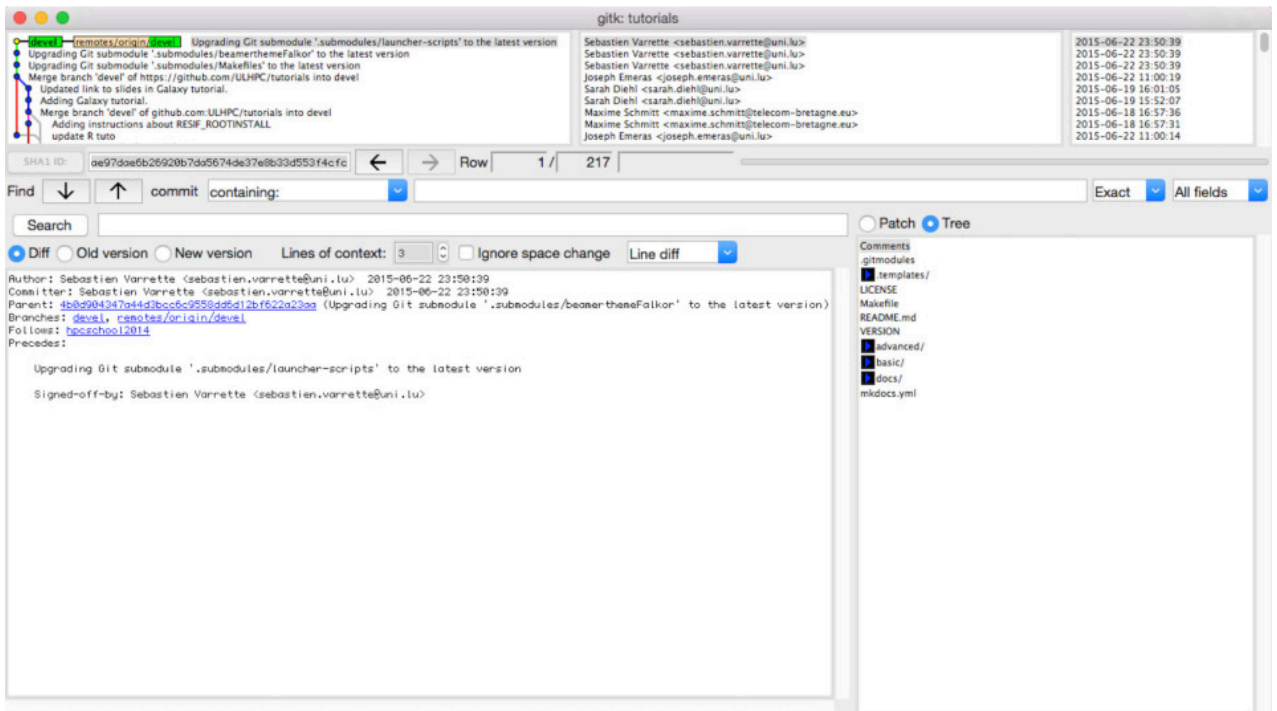
```
$> apt-get install git-core git-flow      # On Debian-like systems
$> yum install git gitflow                # On CentOS-like systems
$> brew install git git-flow              # On Mac OS, using Homebrew
```

Installation on Windows MsysGit

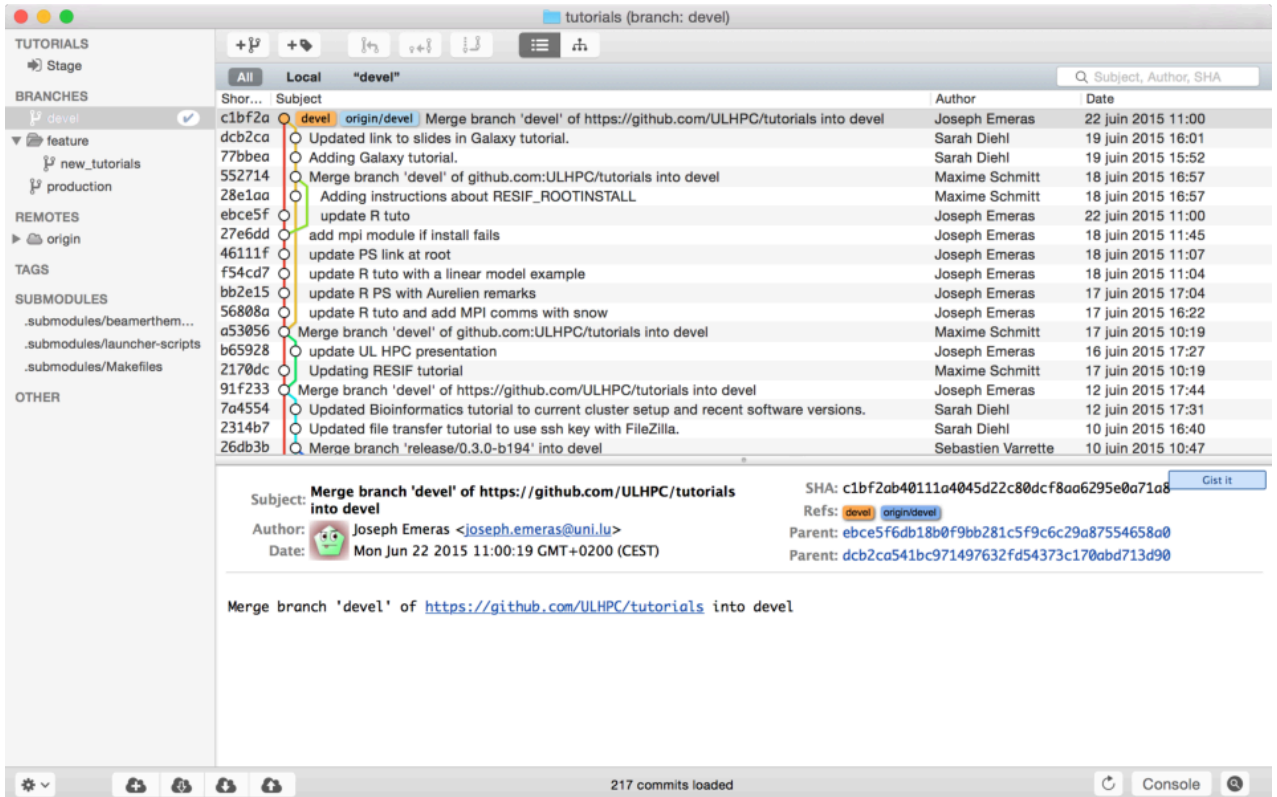
- Incl. Git Bash/GUI & Shell Integration
  - ↪ install **Git bash** + command prompt
  - ↪ select checkout windows / commit unix

- **Your turn!** http://rr-tutorials.readthedocs.io/en/latest/setup/
  - ↪ Ensure you have git installed

Git GUI (default) Gitk

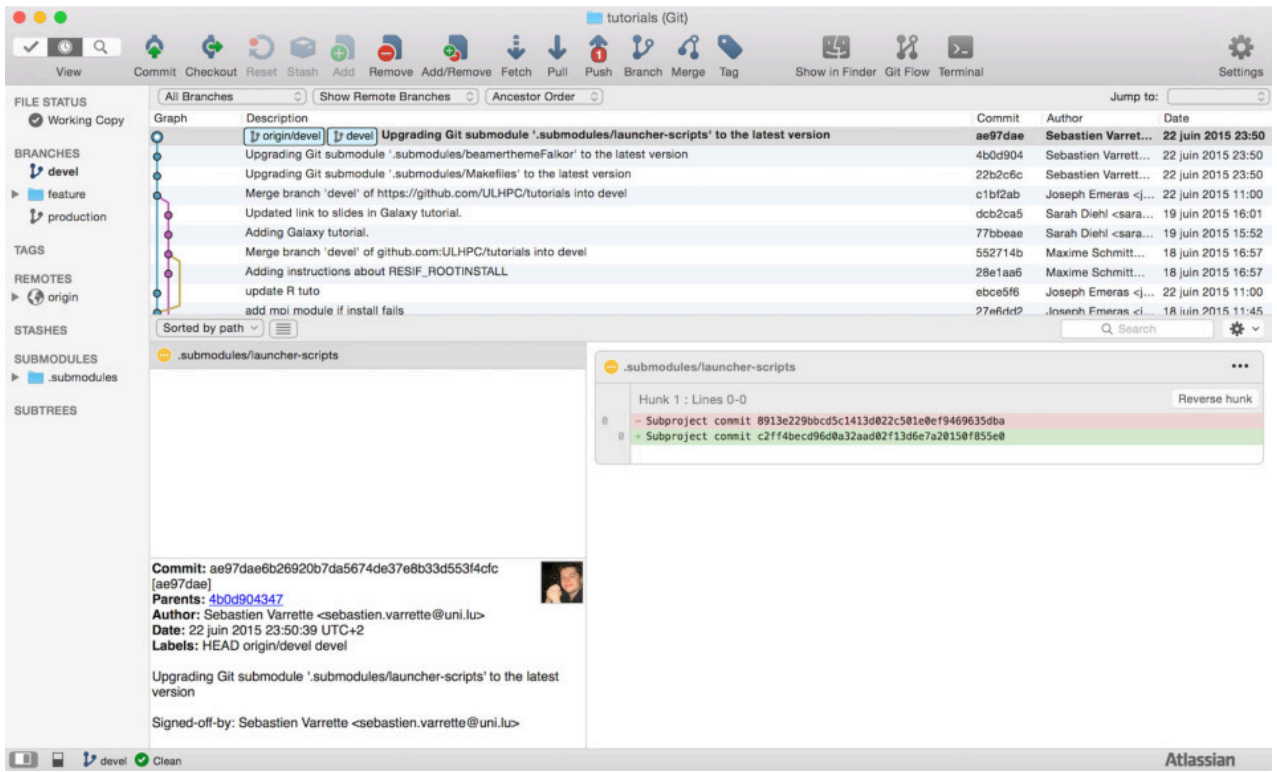


Git GUI (Mac OS) GitX-dev



http://rowanj.github.io/gitx/

Git GUI (Windows/Mac) SourceTree



http://www.sourcetreeapp.com/

1. Let it install a default git ignore file
2. make it load your SSH key created with Putty

Preliminary Configurations

- Global Git configuration are stored in `~/.gitconfig`
  - ↪ **Ex**: see my personal `.gitconfig`
- You **SHOULD** at least configure your name and email to commit
  - ↪ open a terminal (Git bash under windows) for the below commands

```
$> git config --global user.name "Firstname LastName"
$> git config --global user.email "Firstname.LastName@uni.lu"
$> git config --global color.ui true                    # Colors
$> git config --global core.editor vim                  # Editor
```

Preliminary Configurations

- Global Git configuration are stored in `~/.gitconfig`
  - ↪ **Ex**: see my personal `.gitconfig`
- You **SHOULD** at least configure your name and email to commit
  - ↪ open a terminal (Git bash under windows) for the below commands

```
$> git config --global user.name "Firstname LastName"
$> git config --global user.email "Firstname.LastName@uni.lu"
$> git config --global color.ui true                    # Colors
$> git config --global core.editor vim                  # Editor
```

Your Turn!

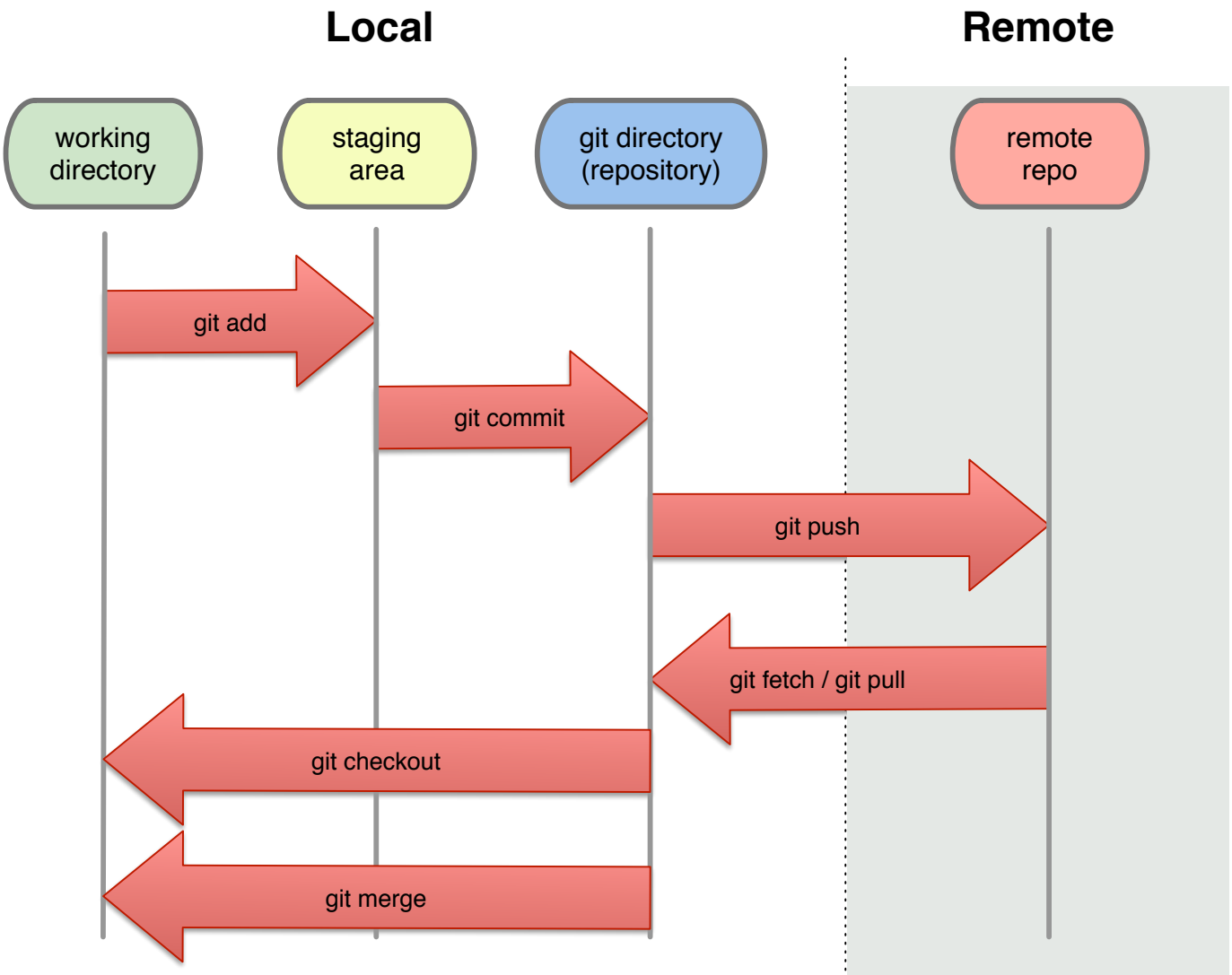
- Then check the changes by: `git config -l | grep user`

Git Commands Aliases

- You can also create git command aliases in `~/.gitconfig`.
  - ↪ **Ex** copy/paste from my personal `.gitconfig`

```
[alias]
    up = pull origin
    pu = push origin
    st = status
    df = diff
    ci = commit -s
    co = checkout
    br = branch
    w  = whatchanged --abbrev-commit
    ls = ls-files
    gr = log --graph --oneline --decorate
    amend = commit --amend
```

Git Workflow



Creating a Repository

```
$> git [flow] init
```

- Initializes a new git (flow) repository in the current directory

Creating a Repository

```
$> git [flow] init
```

- Initializes a new git (flow) repository in the current directory

Your Turn!

```
$> cd /tmp
$> mkdir firstproject
$> cd firstproject

$> git init
Initialized empty Git repository in /private/tmp/firstproject/.git/
```

Cloning a Repository

```
$> git clone [--recursive] <url> [<path>]
```

Type	URL Format / Example	Port
Local	`/path/to/project.git`	n/a
SSH	`git+ssh://user@server:port/project.git`	22
Git	`git://server/project.git`	9418
HTTPS	`https://github.com/Falkor/falkorlib.git`	443

Cloning a Repository

```
$> git clone [--recursive] <url> [<path>]
```

Your Turn!

```
$> cd /tmp
$> git clone https://github.com/Falkor/RR-tutorials.git
Cloning into 'tutorials'...
remote: Counting objects: 1247, done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 1247 (delta 32), reused 0 (delta 0), pack-reused 1181
Receiving objects: 100% (1247/1247), 15.74 MiB | 3.08 MiB/s, done.
Resolving deltas: 100% (588/588), done.
Checking connectivity... done.
$> git clone --recursive \
       https://github.com/Falkor/RR-tutorials.git /tmp/tutorials2
```

Inspecting a Repository

```
$> git status [-s]                    # -s:  short / simplified output
```

Inspecting a Repository

```
$> git status [-s]                # -s:  short / simplified output
```

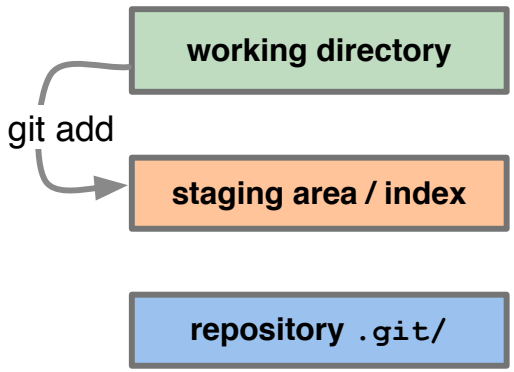
Your Turn!

```
$> cd /tmp/firstproject
$> git status
On branch master

Initial commit

nothing to commit

# Create an empty file
$> touch README.md
```

```
$> git status
On branch master

Initial commit

Untracked files:
    README

nothing added to commit but untracked
files present
$> git status -s
?? README
```

Add / Tracking [new] file(s)

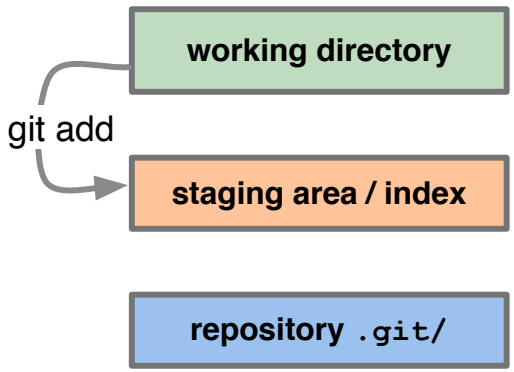
```
$> git add [-f] <pattern>
```



- Adds changes to the index
  - ↪ Add a specific file: `git add README`
  - ↪ Add a set of files: `git add *.py`
- Beware that empty directory cannot be added **directly**
  - ↪ due to the internal file representation (**blobs**)
  - ↪ **Tips**: add an hidden file `.empty` (or `.gitignore`)

Add / Tracking [new] file(s)

```
$> git add [-f] <pattern>
```



- Adds changes to the index
  - ↪ Add a specific file: `git add README`
  - ↪ Add a set of files: `git add *.py`
- Beware that empty directory cannot be added **directly**
  - ↪ due to the internal file representation (**blobs**)
  - ↪ **Tips**: add an hidden file `.empty` (or `.gitignore`)

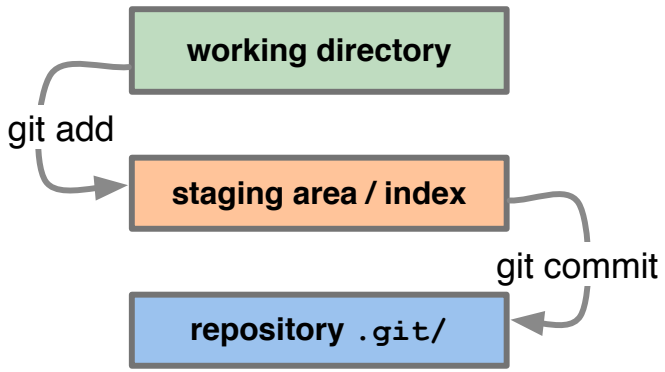
Your Turn!

```
$> cd /tmp/firstproject
$> git status -s
?? README
```

```
$> git add README
$> git status -s
A  README
```

Committing your changes

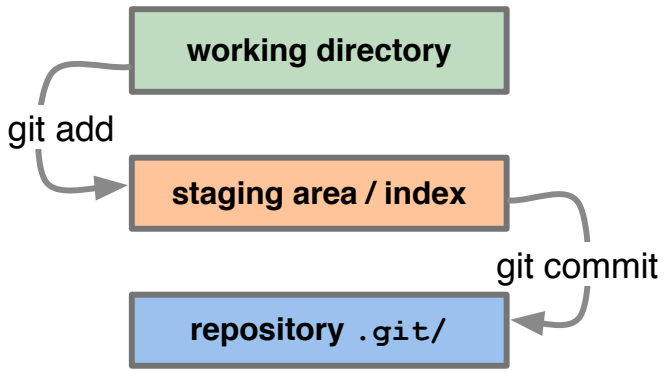
```
$> git commit [-s] [-m "msg"]
```



- Commit all changes: `git commit -a`

Committing your changes

```
$> git commit [-s] [-m "msg"]
```



- Commit all changes: `git commit -a`

Your Turn!

```
$> cd /tmp/firstproject
$> git commit -s -m "add README"    # OR git ci -m "add README"
[master (root-commit) ee60f53] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README
$> git status               # OR git st
On branch master
nothing to commit, working directory cleant
```

Removing Files

```
$> git rm [-rf] [--cached] <file>
```

- `--cached`: remove from Staging area
  - ↪ otherwise (default): from index **and** file system

Ignoring Files

Ignoring files from staging: '.gitignore'

- you can create a `.gitignore` file listing patterns to ignore
  - ↪ Blank lines or lines starting with \# are ignored
  - ↪ End pattern with slash (/) to specify a directory
  - ↪ Negate pattern with exclamation point (!)
- Collection of useful `.gitignore` templates

```
.DS_Store
*~
```

```
*.asv
*.m~
*.mex*
tmp/*
```

- LaTeX `.gitignore`
- Python `.gitignore`
- Ruby `.gitignore`

Moving Files

```
$> git mv <source> <destination>
```

```
# Equivalent of:
mv <source> <destination>
git rm <source>
git add <destination>
```

Moving Files

```
$> git mv <source> <destination>
```

```
# Equivalent of:
mv <source> <destination>
git rm <source>
git add <destination>
```

Your Turn!

```
$> cd /tmp/firstproject
$> git mv README README.md
$> git status
On branch master
Changes to be committed:
    renamed:    README -> README.md
$> git commit -m "a first move"
```

Check the Commit History

```
$> git log [-p] [--stat] [--graph --oneline --decorate]
```

- `-p` / `--stat`: show the differences introduced in each commit
- You can also perform some date filtering

  ```
  $> git log --since=2.weeks
  ```

- Ncurses-based text-mode interface: `tig`

Check the Commit History

```
$> git log [-p] [--stat] [--graph --oneline --decorate]
```

- `-p` / `--stat`: show the differences introduced in each commit
- You can also perform some date filtering

  ```
  $> git log --since=2.weeks
  ```

- Ncurses-based text-mode interface: `tig`

Your Turn!

```
$> cd /tmp/firstproject
$> git log --oneline --graph --decorate     # OR git gr
* f1f0c27 (HEAD -> master) a first move
* ee60f53 add README
$> git log -p -1          # only the last commit OR git show
$> tig
```

Show differences

```
$> git diff [--cached] [<ref>]
```

- Check **un-staged** changes: `git diff`
  - ↪ `--cached`: check **staged** changes
- Relative to a specific revision:

```
$> git diff 1776f5
$> git diff HEAD^
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

- Restore to the last committed/cloned version: **all** changes are lost!

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

```
$> git revert <commit>                   # revert a <commit>
```

- Make a new commit that undoes all changes made in <commit>

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

```
$> git revert <commit>                   # revert a <commit>
```

Your Turn!

```
$> cd /tmp/firstproject
$> git commit --amend
$> echo 'toto' >> README.md
```

```
$> cat README.md && git status
$> git checkout -- README
$> git status
```

Git Summary

Basic Workflow

Edit files	`vim` / `emacs` / `subl` ...
Stage the changes	`git add`
Review your changes	`git status`
Commit the changes	`git commit`

Git Summary

For cheaters: A Basicerer Workflow

Edit files	`vim` / `emacs` / `subl` ...
Stage & commit the changes	`git commit -a`

Git Summary

For cheaters: A Basicerer Workflow

Edit files	`vim` / `emacs` / `subl` ...
Stage & commit the changes	`git commit -a`

- **Advices**: **Commit early, commit often!**
  - ↪ commits = save points
    - ✓ use descriptive commit messages
  - ↪ Don't get out of sync with your collaborators
  - ↪ Commit the sources, not the derived files

- **Not covered here (by lack of time)**
  - ↪ Branches, tags, remotes, submodules, subtrees, etc...

Summary

Environment Management

- RR assumes that you **Master your environment**
- Keep it **clean and automated** by:
  - ↪ Using common building tools — make, cmake etc.
  - ↪ Using a constrained environment
    - ✓ Sandboxed Ruby environment — `bundler`, `Gemfile`
    - ✓ Sandboxed Python — `pip freeze`, `pyenv`, `virtualenv`
    - ✓ VMs or Containers — Vagrant, Docker
  - ↪ Automate its building through cross-platform recipes
  - ↪ Automatically test your recipes for Environment configuration

Controlled Ruby Environment

- Consider using RVM, rbenv and more importantly Bundler
  - ↪ Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
  - ↪ Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Controlled Ruby Environment

- Consider using RVM, rbenv and more importantly Bundler
  - ↪ Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
  - ↪ Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Typical setup of a freshly cloned project:

```
$> gem install bundler # assuming it is not yet available
$> bundle              # clone ruby deps/env as defined in Gemfile*
$> rake -T             # To list the available tasks
```

Controlled Ruby Environment

- Consider using RVM, rbenv and more importantly Bundler
  - ↪ Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
  - ↪ Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Typical setup of a freshly cloned project:

```
$> gem install bundler # assuming it is not yet available
$> bundle              # clone ruby deps/env as defined in Gemfile*
$> rake -T             # To list the available tasks
```

Recommended Gems

`rake`, `bundler`, `falkorlib`

Controlled Python Environment

- `pip`: Python package manager
  - ↪ "nice" python packages: `mkdocs`...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

Controlled Python Environment

- `pip`: Python package manager
  - ↪ "nice" python packages: `mkdocs`...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

```
$> pip install -U pip                       # upgrade on Linux/Mac OS
```

Controlled Python Environment

- `pip`: Python package manager
  - ↪ "nice" python packages: `mkdocs`...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

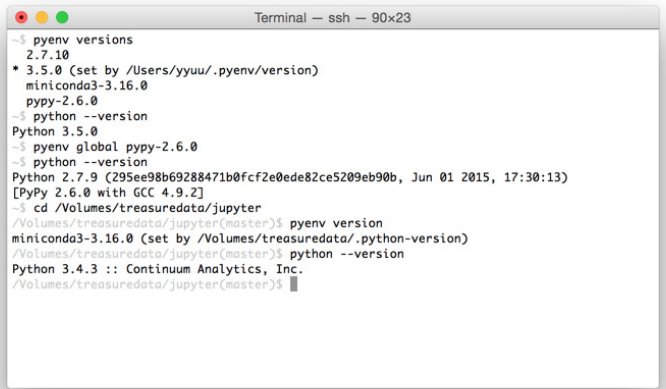
```
$> pip install -U pip                       # upgrade on Linux/Mac OS
```

- Dump python environment to a requirements file

```
$> pip freeze -l > requirements.txt         # as Ruby Gemfiles
```

Pyenv / VirtualEnv / Autoenv

- pyenv: ≃ RVM/rbenv for Python
- virtualenv ≃ RVM Gemset
- (optional) autoenv
  - ↪ Directory-based shell environments
  - ↪ easy config through .env file. **Ex**:




```
# (rootdir)/.env : autoenv configuration file
pyversion=‘head .python-version‘
pvenv=‘head .python-virtualenv‘

pyenv virtualenv --force --quiet ${pyversion} ${pvenv}-${pyversion}
# activate it
pyenv activate ${pvenv}-${pyversion}
```

Constrained VM environment

- Let's see how to reproduce a simple yet practical example in a **constrained** and **reproducible** VM environment.

Challenge 1: Reproduce the Build of these Slides

- Several **tricky** issues illustrating previous best practices
  - ↪ grab the sources — `git`
  - ↪ use of a constrained environment — Vagrant
  - ↪ installing the prerequisite software environment — `apt-get`
    - ✓ [un]common mix here: `make`, `latex-beamer`, `biber`, `pandoc`...
    - ✓ generally the **major challenge** in reproducing computations...

Constrained VM environment

- Let's see how to reproduce a simple yet practical example in a **constrained** and **reproducible** VM environment.

Challenge 1: Reproduce the Build of these Slides

- Several **tricky** issues illustrating previous best practices
  - ↪ grab the sources — `git`
  - ↪ use of a constrained environment — Vagrant
  - ↪ installing the prerequisite software environment — `apt-get`
    - ✓ [un]common mix here: `make`, `latex-beamer`, `biber`, `pandoc`...
    - ✓ generally the **major challenge** in reproducing computations...

`http://rr-tutorials.readthedocs.io/en/latest/hands-on/vagrant/`

IF NOT YET DONE: `http://rr-tutorials.readthedocs.io/en/latest/setup/`

Grab the [Code/Data] Source

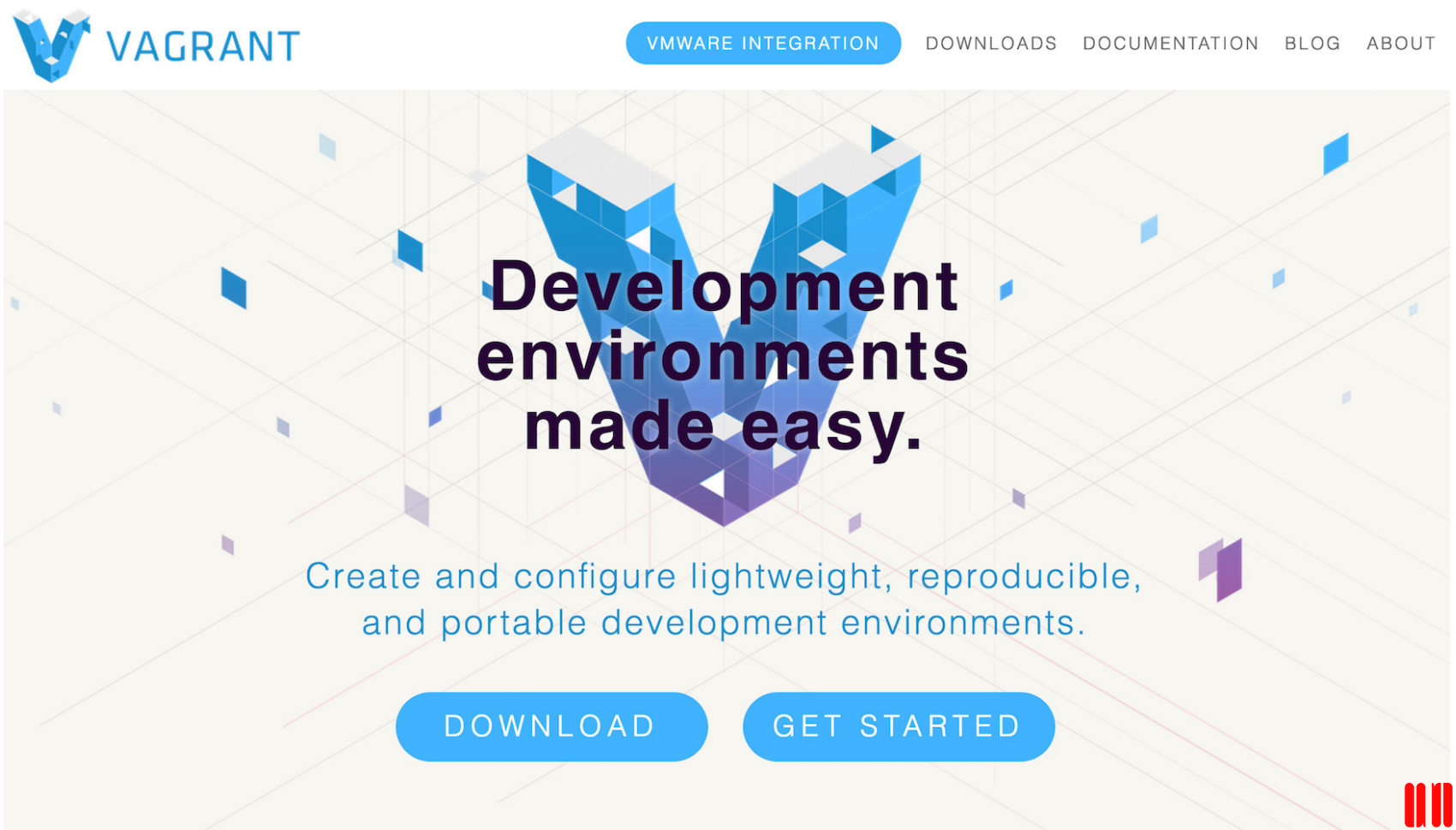
- You should have now Git installed
- Get the `RR-tutorials` repository from Github

```
$> git clone https://github.com/Falkor/RR-tutorials.git
$> cd RR-tutorials
$> make setup  # OR git submodule init && git submodule update
```

- **Notable** elements within this cloned repository:
  - ↪ the LaTeX slides sources `slides/2016/cloudcom2016/src/`
  - ↪ Documentation sources `mkdocs.yml` and `docs/`
  - ↪ Vagrant configuration for **this** project `Vagrantfile`
  - ↪ Bats unit tests `tests/`
  - ↪ Continuous Integration settings through Travis-CI `.travis.yml`

Use a Constrained Environment

http://vagrantup.com/



What is Vagrant ?

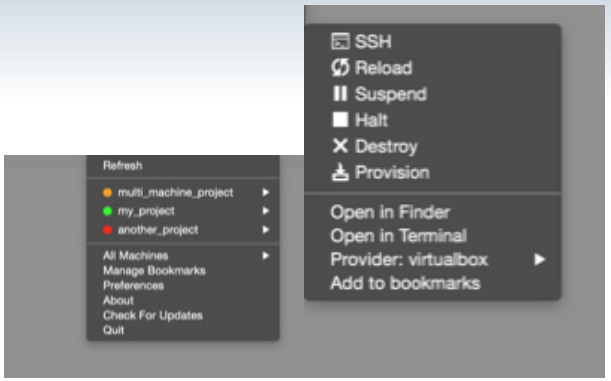
Create and configure **lightweight**, **reproducible**, and **portable** development environments

- **Command line** tool `vagrant [...]`
- Easy and Automatic per-project VM management
  - ↪ Supports many hypervisors: VirtualBox, VMWare...
  - ↪ Easy text-based configuration (Ruby syntax) `Vagrantfile`
- Supports **provisioning** through configuration management tools
  - ↪ Shell
  - ↪ Puppet `https://puppet.com/`
  - ↪ Salt... `https://saltstack.com/`

Cross-platform: runs on Linux, Windows, MacOS

Installation Notes

http://rr-tutorials.readthedocs.io/en/latest/setup/



- **Mac OS X**:
  - ↪ best done using Homebrew and Cask

```
$> brew install caskroom/cask/brew-cask
$> brew cask install virtualbox    # install virtualbox
$> brew cask install vagrant
$> brew cask install vagrant-manager # cf http://vagrantmanager.com/
```

- **Windows / Linux**:
  - ↪ install Oracle Virtualbox and the Extension Pack
  - ↪ install Vagrant

Why use Vagrant?

- Create new VMs quickly and easily: only one command!
  - ↪ `vagrant up`
- Keep the number of VMs under control
  - ↪ All configuration in `VagrantFile`
- **Reproducibility**
  - ↪ Identical environment in development and production
- **Portability**
  - ↪ **avoid** sharing 4 GB VM disks images
  - ↪ Vagrant Cloud to share your images
- **Collaboration made easy**:

```
$> git clone ...
$> vagrant up
```

Minimal default setup

```
$> vagrant init [-m] <user>/<name> # setup vagrant cloud image
```

- A `Vagrantfile` is configured for box `<user>/<name>`
  - ↪ Find existing box: Vagrant Cloud https://vagrantcloud.com/
  - ↪ You can have multiple (named) box within the **same** `Vagrantfile`
    - ✓ See `ULHPC/puppet-sysadmins/Vagrantfile`

```
Vagrant.configure(2) do |config|
   config.vm.box = '<user>/<name>'
   config.ssh.insert_key = false
end
```

Box name	Description
`ubuntu/trusty64`	Ubuntu Server 14.04 LTS
`debian/contrib-jessie64`	Vanilla Debian 8 "Jessie"
`centos/7`	CentOS Linux 7 x86_64
`svarrette/RR-tutorials`	IEEE CloudCom 2016 Tuto

Pulling and Running a Vagrant Box

```
$> vagrant up          # boot the box(es) set in the Vagrantfile
```

- Base box is downloaded and stored locally `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template
  - ↪ The VM is booted and (eventually) provisioned
  - ↪ Once within the box: `/vagrant` = directory hosting `Vagrantfile`

Pulling and Running a Vagrant Box

```
$> vagrant up          # boot the box(es) set in the Vagrantfile
```

- Base box is downloaded and stored locally `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template
  - ↪ The VM is booted and (eventually) provisioned
  - ↪ Once within the box: `/vagrant` = directory hosting `Vagrantfile`

```
$> vagrant status                  # State of the vagrant box(es)
```

Pulling and Running a Vagrant Box

```
$> vagrant up          # boot the box(es) set in the Vagrantfile
```

- Base box is downloaded and stored locally `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template
  - ↪ The VM is booted and (eventually) provisioned
  - ↪ Once within the box: `/vagrant` = directory hosting `Vagrantfile`

```
$> vagrant status                  # State of the vagrant box(es)
```

```
$> vagrant ssh                  # connect inside it, CTRL-D to exit
```

Stopping Vagrant Box

```
$> vagrant { destroy | halt }                    # destroy / halt
```

- Once you have finished your work within a *running* box
  - ↪ save the state for later with `vagrant halt`
  - ↪ reset changes / tests / errors with `vagrant destroy`
  - ↪ commit changes by generating a new version of the box

Back to Hands-on 1

Your Turn! http://rr-tutorials.readthedocs.io/en/latest/hands-on/vagrant/

- **Steps [1-4]** to cover the following elements:
  - ↪ **Basic Usage of Vagrant**
  - ↪ **Build these Slides**
    - ✓ find the prerequisite software environment `apt-get`
    - ✓ [un]common mix here: `make`, `latex-beamer`, `biber`, `pandoc...`

- **Hints**:
  - ↪ if a package is missing, find the appropriate one `apt-cache search`
  - ↪ Ubuntu Package Search for a missing `*.sty` http://packages.ubuntu.com/
    - ✓ `Search the contents of packages` for Distribution `Trusty`

Vagrant Box Provisioning

- Now you have *hopefully* a working **documented procedure**
  - ↪ it's time to **bundle it** for provisioning the box upon boot
  - ↪ key for sustainable reproducible environment
- Simple case: **inline** provisioning **i.e.** list commands to run

Vagrant Box Inline Provisioning

- Now you have *hopefully* a working **documented procedure**
  - ↪ it's time to **bundle it** for provisioning the box upon boot
  - ↪ key for sustainable reproducible environment
- Simple case: **inline** provisioning **i.e.** list commands to run

```
config.vm.provision "shell", inline: <<-SHELL
  sudo apt-get update --fix-missing
  sudo apt-get upgrade
  # Complete the below list of missing packages
  apt-get -yq --no-install-suggests --no-install-recommends install \
        git make latex-beamer biber latex-make [...]
SHELL
```

Vagrant Box Inline Provisioning

- Now you have *hopefully* a working **documented procedure**
  - ↪ it's time to **bundle it** for provisioning the box upon boot
  - ↪ key for sustainable reproducible environment
- Simple case: **inline** provisioning **i.e.** list commands to run

```
config.vm.provision "shell", inline: <<-SHELL
  sudo apt-get update --fix-missing
  sudo apt-get upgrade
  # Complete the below list of missing packages
  apt-get -yq --no-install-suggests --no-install-recommends install \
        git make latex-beamer biber latex-make [...]
SHELL
```

```
$> vagrant provision          # test your provisioning config
```

Vagrant Box Inline Provisioning

Your Turn! http://rr-tutorials.readthedocs.io/en/latest/hands-on/vagrant/

- **Steps 5**:
  - ↪ adapt the `Vagrantfile` to embed your commands
  - ↪ recall that relative paths are expanded relative to the location of the root `Vagrantfile`
  - ↪ inline command are run as the `vagrant` user, **not** `root`

- **IMPORTANT**:
  - ↪ all your commands should run **in a non-interactive** way

```
apt-get install -y <package>    # Debian / Ubuntu
yum     install -y <package>    # CentOS/ Redhat
```

Vagrant Box Shell Provisioning

- Embed your inline commands in a **Shell/Python/Ruby** script
  ↪ see sample script `vagrant/bootstrap.sample.sh`

```
config.vm.provision "shell", path: "<script>.{sh|py|rb}"
```

Your Turn! http://rr-tutorials.readthedocs.io/en/latest/hands-on/vagrant/

- **Steps 6**: copy and adapt `vagrant/bootstrap.sample.sh`
  ↪ adapt the `Vagrantfile` to provision the VM with your script
  ↪ test a reproducible provisioning from scratch

```
$> vagrant destroy && vagrant up && vagrant ssh
$> make -C make -C /vagrant/slides/2016/cloudcom2016/src/
```

Vagrant Box Packaging

- At some moment, you probably want to diffuse your custom box!
  - ↪ **Ex**: `svarrette/RR-tutorials` used for this tutorial
  - ↪ use Vagrant Cloud as a global storage media
  - ↪ `VBoxManage list runningvms` to get the real box name

```
$> vagrant package --base <real-box-name> --output <name>.box
```

Vagrant Box Packaging

- At some moment, you probably want to diffuse your custom box!
  - ↪ **Ex**: `svarrette/RR-tutorials` used for this tutorial
  - ↪ use Vagrant Cloud as a global storage media
  - ↪ `VBoxManage list runningvms` to get the real box name

```
$> vagrant package --base <real-box-name> --output <name>.box
```

- **BEFORE packaging your box**:
  - ↪ Use official insecure SSH key `config.ssh.insert_key=false`
  - ↪ **Purge** the VM to reduce its size see `vagrant/purge.sh`
    - ✓ remove useless [big] packages `aptitude purge [...]`
    - ✓ Empty logs/history etc.
    - ✓ Zero out the free space `dd if=/dev/zero of=/EMPTY bs=1M`
  - ↪ **Up-to-date** Virtualbox **Guest additions** `vagrant vbguest`

Detailed Pre-Packaging Steps (1/2)

- Ensure you **DO NOT** reset the default (insecure) SSH key
  - ↪ default **expected** setting to SSH your box
  - ↪ **before** `vagrant up`, ensure replacement of SSH keys **is not done**

```
config.ssh.insert_key = false    # in Vagrantfile
```

- **Purge** the VM, in particular to **Zero out the free space**
  - ↪ see `vagrant/purge.sh`

```
# Remove APT cache
apt-get clean -y   && apt-get autoclean -y && apt-get autoremove -y
# Remove bash history
unset HISTFILE
rm -f /root/.bash_history && rm -f /home/vagrant/.bash_history
# Zero out free space to aid VM compression
dd if=/dev/zero of=/EMPTY bs=1M
rm -f /EMPTY
```

Detailed Pre-Packaging Steps (2/2)

- Ensure an **Up-to-date Virtualbox Guest additions**
  - ↪ ensure optimized usage of the box
  - ↪ simplified management with the vbguest plugin

```
# Install the 'vbguest' plugin
$> vagrant plugin install vagrant-vbguest
$> vagrant vbguest --status
GuestAdditions versions on your host (5.1.8) and guest (4.3.36)
do not match.
# Upgrade the GuestAdditions
$> vagrant vbguest --do install --auto-reboot [--force]
```

- If you want the **manual** way:
  - ↪ copy `/Applications/VirtualBox.app/Contents/MacOS/VBoxGuestAdditions.iso`
  - ↪ mount in **within** the VM
  - ↪ execute `VBoxLinuxAdditions.run`

Vagrant Box Packaging

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"RR-tutorials_default_1481463725786_57301" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

Vagrant Box Packaging

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"RR-tutorials_default_1481463725786_57301" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

- Now you can upload the generated box on Vagrant Cloud.
  - ↪ select '`New version`', enter the new version number
  - ↪ add a new box provider (`Virtualbox`)
  - ↪ upload the generated box

Vagrant Box Packaging

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"RR-tutorials_default_1481463725786_57301" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

- Now you can upload the generated box on Vagrant Cloud.
  - ↪ select '`New version`', enter the new version number
  - ↪ add a new box provider (`Virtualbox`)
  - ↪ upload the generated box

- Upon successful upload: **release** the uploaded box
  - ↪ by default it is unreleased
  - ↪ Now people using the `<user>/<name>` box will be notified of a pending update

Vagrant Box Packaging

Your Turn! http://rr-tutorials.readthedocs.io/en/latest/hands-on/vagrant/

- **Steps 7-8**: Package your box and diffuse it on Vagrant Cloud
  - ↪ Make preliminary checks
  - ↪ Purge the VM
  - ↪ Package it and Upload to Vagrant Cloud

Vagrant Box Generation

- You might rely on Falkor/vagrant-vms
  - ↪ use it at your own risks
  - ↪ based on packer and veewee

```
$> git clone https://github.com/Falkor/vagrant-vms.git
$> cd vagrant-vms
$> gem install bundler && bundle install
$> rake setup
```

Vagrant Box Generation

- You might rely on Falkor/vagrant-vms
  - ↪ use it at your own risks
  - ↪ based on packer and veewee

```
$> git clone https://github.com/Falkor/vagrant-vms.git
$> cd vagrant-vms
$> gem install bundler && bundle install
$> rake setup
```

```
# initiate a template for a given Operating System:
$> rake packer:{Debian,CentOS,openSUSE,scientificlinux,ubuntu}:init
# Build a Vagrant box
$> rake packer:{Debian,CentOS,openSUSE,scientificlinux,ubuntu}:build
# If things goes fine:
$> vagrant box add packer/<os>-<version>-<arch>/<os>-<version>-<arch>.box
```

Advanced Provisioning: Puppet

- Shell provisioning is a reasonable good basis but **not sufficient**
  - ↪ **hard to be cross-platform** `apt-get` vs. `yum`
- You quickly something more **consistent**
  - ↪ Puppet `https://puppet.com/`
  - ↪ Salt... `https://saltstack.com/`

Advanced Provisioning: Puppet

- Shell provisioning is a reasonable good basis but **not sufficient**
  - ↪ **hard to be cross-platform** `apt-get vs. yum`
- You quickly something more **consistent**
  - ↪ Puppet `https://puppet.com/`
  - ↪ Salt... `https://saltstack.com/`

Puppet: Reproducible/Cross-Platform IT Environment

- Advanced configuration management and **IT Automation**
  - ↪ cross-platform w. Puppet's Resource Abstraction Layer (RAL)
  - ↪ Git-based workflow
- Embed environment management in **manifests** and **modules**
  - ↪ **nodes manifests**: nodes definitions
  - ↪ **modules**: (reusable) set of recipe to configure a given service
    - ✓ Large Community Recipes / Modules `https://forge.puppet.com/`

Puppet Operational modes

- **Masterless** - apply Puppet manifests directly on the target system.
  - ↪ No need of a complete client-server infrastructure.
  - ↪ Have to distribute manifests and modules to the managed nodes.

```
$> puppet apply --modulepath /modules/ /manifests/file.pp
```

Puppet Operational modes

- **Masterless** - apply Puppet manifests directly on the target system.
  - ↪ No need of a complete client-server infrastructure.
  - ↪ Have to distribute manifests and modules to the managed nodes.

```
$> puppet apply --modulepath /modules/ /manifests/file.pp
```

- **Master / Client** Setup
  - ↪ server (running as `puppet`) listening on 8140 on the Puppet Master
  - ↪ client (running as `root`) on each managed node.
    - ✓ Run as a service (default), via cron (with random delays), manually or via MCollective
  - ↪ Client and Server have to share SSL certificates
    - ✓ certificates must be signed by the Master CA

```
$> puppet agent --test [--noop] [--environment <environment>]
```

Puppet DSL

- A Declarative Domain Specific Language (DSL)
  - ↪ defines **STATES** (and **not** procedures)
- Puppet code is written in **manifests** `<file>.pp`
  - ↪ **declare resources** that affect elements of the system
    - ✓ each resource has a type (`package`, `service`, `file`, `user`, `exec` ...)
    - ✓ each resource has a **uniq** title
  - ↪ resources are grouped in **classes**
- Classes and configuration files are organized in **modules**
- **Example** of resources types:

```
file { '/etc/motd':
  content => "Toto"
}
```

```
package { 'openssh':
  ensure => present,
}
```

```
service { 'httpd':
  ensure => running,
  enable => true,
}
```

Puppet Classes

- **Containers** of different resources
  - ↪ Can have parameters since Puppet 2.6

```
class mysql (
  $root_password = 'default_value',
  $port          = '3306',
) {
  package { 'mysql-server':
    ensure => present,
  }
  service { 'mysql':
    ensure    => running,
  }
  [...]
}
```

Puppet Classes Declaration

- To use a class previously defined, we **declare** it
- "Old style" class declaration, without parameters:

```
include mysql
```

- "New style" (from Puppet 2.6) with explicit parameters:

```
class { 'mysql':
  root_password => 'my_value',
  port          => '3307',
}
```

- A class is **uniq** to a given node

Puppet Defines

- Similar to parametrized classes . . .
  - ↪ . . . but can be used multiple times (with different titles).

```
# Definition of a define
define apache::virtualhost (
  $ensure   = present,
  $template = 'apache/virtualhost.conf.erb' ,
  [...] ) {
  file { "ApacheVirtualHost_${name}":
    ensure  => $ensure,
    content => template("${template}"),
  }
}
# Declaration of a define:
apache::virtualhost { 'www.uni.lu':
  template => 'site/apache/www.uni.lu-erb'
}
```

Puppet Variables and Facts

- Can be defined in different places and by different actors:
  - ↪ by client nodes as facts
  - ↪ defined by users in Puppet code, on Hiera on in the ENC
  - ↪ built-in and be provided directly by Puppet
- Facts using `facter`:
  - ↪ runs on clients and collects facts that the server can use as variables

```
$> facter
architecture => x86_64
fqdn => toto.uni.lu
kernel => Linux
memorytotal => 16.00 GB
operatingsystem => Centos
operatingsystemrelease => 6.3
osfamily => RedHat
virtual => physical
[...]
```

- Can be used outside Puppet
- Good tool to **abstract** your environment
  - ↪ permits **reproducible** and cross-platform developments

Puppet User Variables

- In Puppet manifests:

```
$role = 'mail'
$package = $::operatingsystem ? {
    /(?i:Ubuntu|Debian|Mint)/ => 'apache2',
    default                   => 'httpd',
}
```

- In an External Node Classifier (ENC)
  - ↪ Common ENC: Puppet DashBoard, the Foreman, Puppet Enterprise.
- In an Hiera backend

```
$syslog_server = hiera(syslog_server)
```

Puppet Nodes

- A **node**/system is identified by its **certname**
  - ↪ defaults to the node's fqdn

```
node 'web01' {
  include apache
}
```

```
node /^www\d+$/ {
  include apache
}
```

- Nodes classification can be done by External Node Classifier (ENC)
  - ↪ Puppet DashBoard, The Foreman and Puppet Enterprise
- Nodes classification can be done also by Hiera
  - ↪ In `/etc/puppet/manifests/site.pp`

```
hiera_include('classes')
```

Vagrant Puppet Provisionning

- Operate in **masterless** mode
- Embed your manifests and modules in your repository
  - ↪ grab community modules with `librarian-puppet`, `r10K`

```
config.vm.provision :puppet  do |puppet|
    puppet.hiera_config_path = 'hieradata/hiera.yaml'
    puppet.working_directory = '/vagrant'
    puppet.manifests_path    = "manifests"
    puppet.module_path       = "modules"
    puppet.manifest_file     = "init.pp"
    puppet.options = [ '-v','--report','--show_diff','--pluginsync' ]
end
```

Vagrant Puppet Provisionning

- Operate in **masterless** mode
- Embed your manifests and modules in your repository
  - ↪ grab community modules with `librarian-puppet`, `r10K`

```
config.vm.provision :puppet  do |puppet|
    puppet.hiera_config_path = 'hieradata/hiera.yaml'
    puppet.working_directory = '/vagrant'
    puppet.manifests_path    = "manifests"
    puppet.module_path       = "modules"
    puppet.manifest_file     = "init.pp"
    puppet.options = [ '-v','--report','--show_diff','--pluginsync' ]
end
```

Your Turn!

Vagrant Puppet Provisionning

- Operate in **masterless** mode
- Embed your manifests and modules in your repository
  - ↪ grab community modules with `librarian-puppet`, `r10K`

```
config.vm.provision :puppet  do |puppet|
    puppet.hiera_config_path = 'hieradata/hiera.yaml'
    puppet.working_directory = '/vagrant'
    puppet.manifests_path    = "manifests"
    puppet.module_path       = "modules"
    puppet.manifest_file     = "init.pp"
    puppet.options = [ '-v','--report','--show_diff','--pluginsync' ]
end
```

Your Turn! ... **Or not ☺(no time)**

Software/Modules Management

- **Software Management Challenge**
  - ↪ Not so much standardization
    - ✓ every machine/app has a different software stack / installation procedure
    - ✓ Sites share unique hardware among teams with very different requirements
    - ✓ **You** want to experiment with many exotic architectures
- Software Flavor vs. Dependency nightmare vs Performance
  - ↪ **Ex**: 3 compilers + 3 MPI + n software
  - ↪ Complex set of CLI options,
  - ↪ One of the main limits for RR
- Some Tools can help you!
  - ↪ Easybuild `http://easybuild.readthedocs.io/`
  - ↪ Spack `http://spack.readthedocs.io/`
  - ↪ CDE
  - ↪ Kameleon `http://kameleon.imag.fr/`

EasyBuild

http://easybuild.readthedocs.io/

- Easybuild: open-source framework to (automatically) build scientific software
- **Why?**: *"Could you please install this software on the cluster?"*
  - ↪ Scientific software are often **painful** to build
    - ✓ non-standard build tools / incomplete build procedure
    - ✓ hardcoded parameters and/or poor/outdated documentation
  - ↪ EasyBuild helps to facilitate this task
    - ✓ consistent software build and installation framework
    - ✓ automatically generates LMod modulefiles

EasyBuild Installation

http://easybuild.readthedocs.io/

```
# pick an installation prefix to install EasyBuild to
export EASYBUILD_PREFIX=$HOME/.local/easybuild
# download script
curl -O goo.gl/RK3Gpf    # Get bootstrap_eb.py
# bootstrap EasyBuild
python bootstrap_eb.py $EASYBUILD_PREFIX
# update $MODULEPATH, and load the EasyBuild module
module use $EASYBUILD_PREFIX/modules/all
module load EasyBuild
```

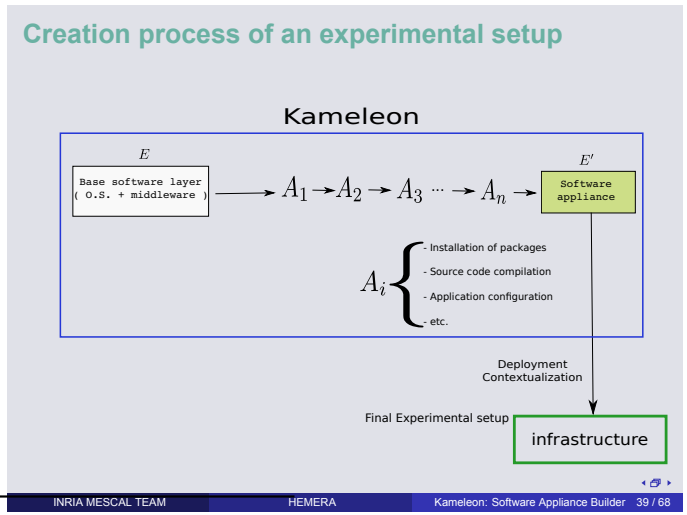
EasyBuild Usage

http://easybuild.readthedocs.io/

```
# Load EasyBuild module
module load EasyBuild
# Check version
eb --version
# Look for HPL
eb -S HPL
# Check what needs to be built to compile HPL 2.1 with Intel compiler
HPL-2.1-intel-2016b.eb
# Check what needs to be built to compile HPL 2.1 with GCC/OpenMPI/...
eb HPL-2.1-foss-2016b.eb -Dr
# Build HPL and its dependencies
eb HPL-2.1-foss-2016b.eb -r
# See available HPL now
module avail HPL
# Amending an existing easyconfig
eb HPL-2.1-foss-2016b.eb --try-software-version=2.2
```

Kameleon: Reproducible SW[11]

- Uses *recipes* (high-level description)
  - ↪ Similar to cfengine, Puppet, Chef in the sysadmin world
- Persistent cache to allow re-generation without external resources
  - ↪ Linux distribution mirror ⇝ self-contained archive
  - ↪ Supports LXC, Docker, VirtualBox, qemu, Kadeploy images, etc.



[11] Cristian Camilo Ruiz Sanabria et al. "Reproducible Software Appliances for Experimentation". In: **TRIDENTCOM'2014**.

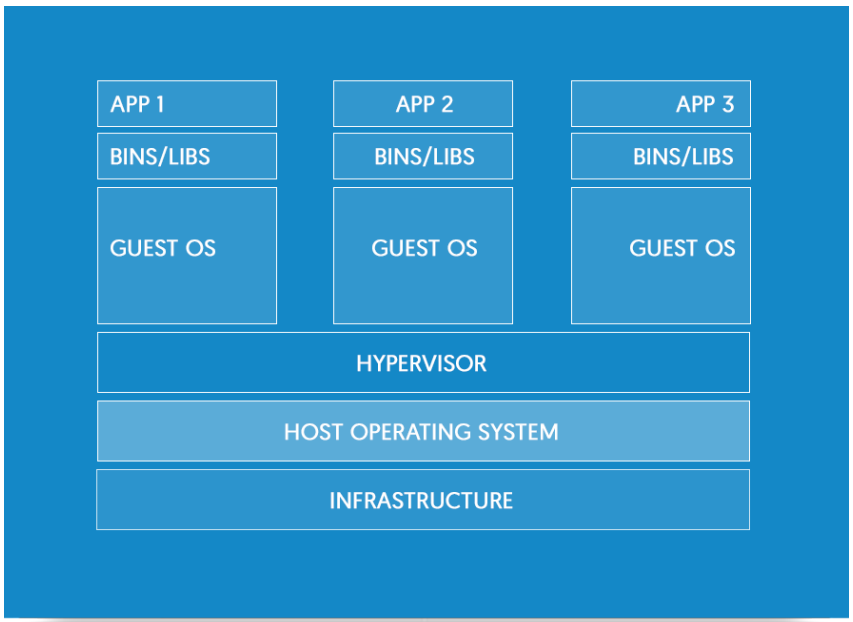
Courtesy of L. Nussbaum

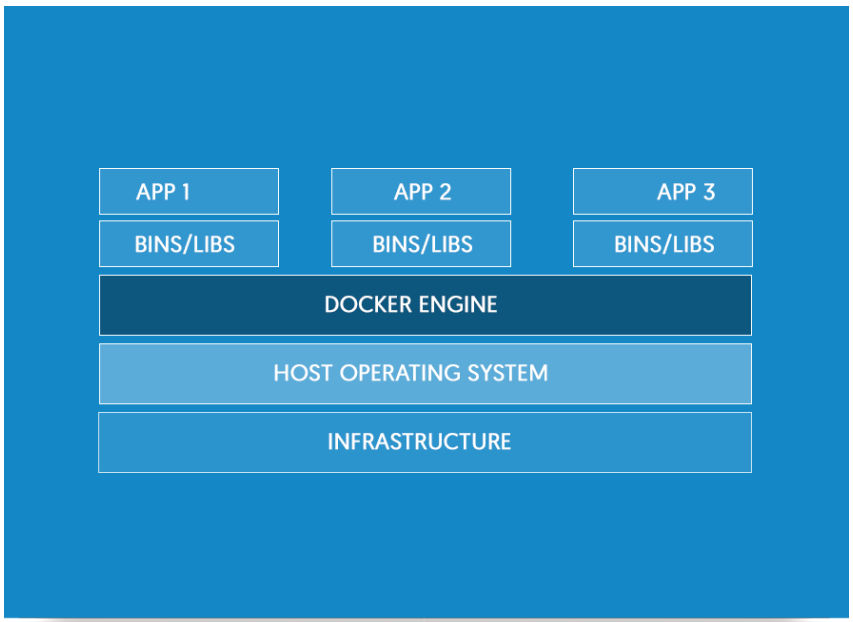
Lighweight Constrained Env.: Docker

http://www.docker.com

- Open-source engine
- Automates the deployment of any application
  - ↪ **lightweight**, **portable**, **self-sufficient** container
  - ↪ will run virtually anywhere
- Tries to achieve deterministic builds by isolating your service
  - ↪ build done from a snapshotted OS and running imperative steps on top of it
- **Dependency hell**:
  - ↪ Docker works with images that consume minimal disk space
  - ↪ all images are **versioned**, **archivable**, and **shareable** DockerHub
- `Dockerfiles`: resolving imprecise documentation

VM vs. Containers





- **Virtual machines**
  - ↪ app + binaries + libraries
  - ↪ incl. an entire guest OS
- **Container**
  - ↪ app + binaries + libraries
  - ↪ kernel shared
  - ↪ run on **any** computer

Pulling and Running Images

```
$> docker pull <name>:<tag>
```

- Pull a **public** image such as ubuntu or centos
  ↪ if a tag is not specified, use “latest”.

Pulling and Running Images

```
$> docker pull <name>:<tag>
```

- Pull a **public** image such as ubuntu or centos
  ↪ if a tag is not specified, use “latest”.

```
$> docker run -it <name>
```

Pulling and Running Images

```
$> docker pull <name>:<tag>
```

- Pull a **public** image such as ubuntu or centos
  ↪ if a tag is not specified, use "latest".

```
$> docker run -it <name>
```

```
$> docker commit <ID> <name>
```

Pulling and Running Images

```
$> docker pull <name>:<tag>
```

- Pull a **public** image such as ubuntu or centos
  ↪ if a tag is not specified, use "latest".

```
$> docker run -it <name>
```

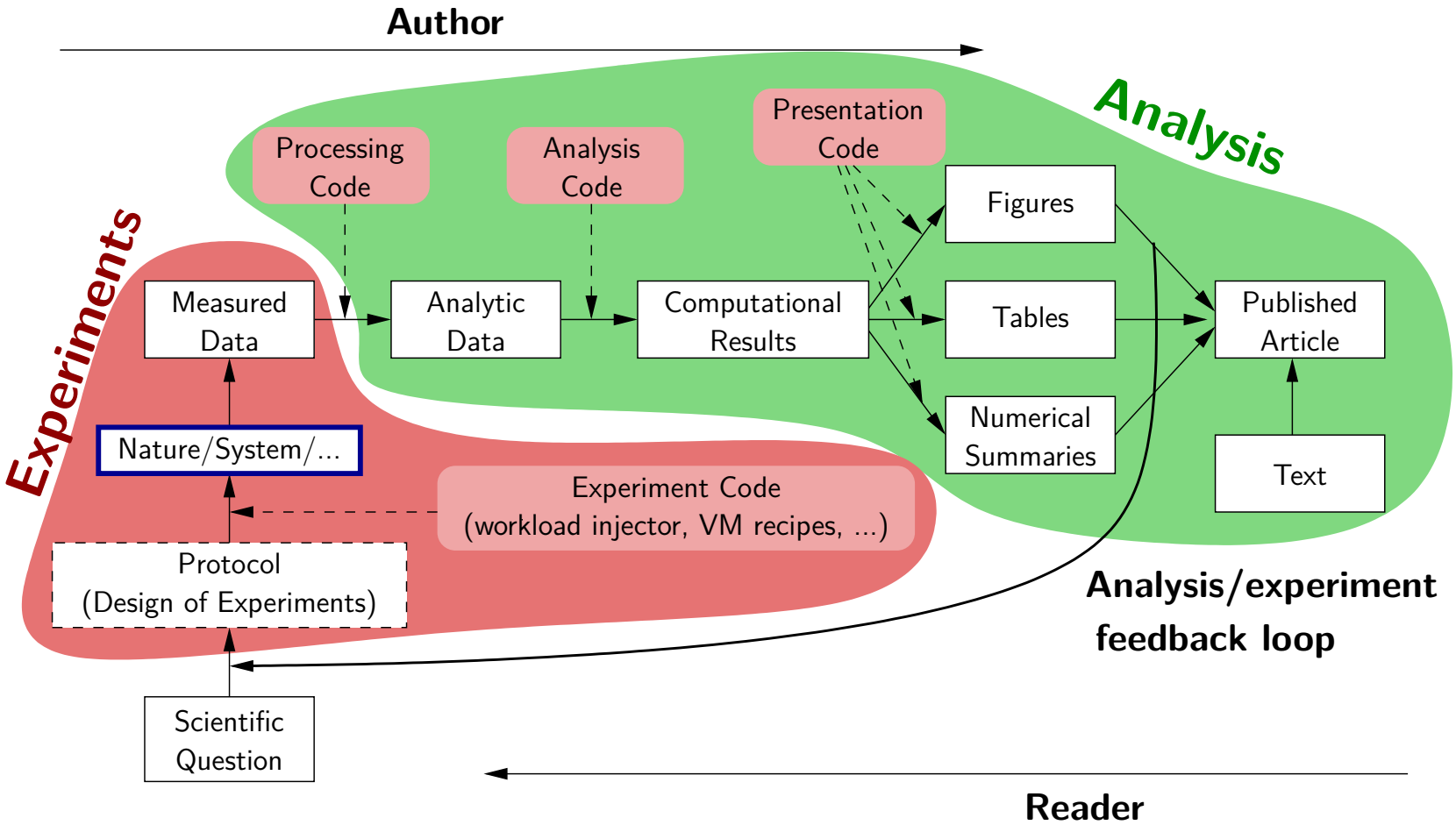
```
$> docker commit <ID> <name>
```

Your Turn!

http://rr-tutorials.readthedocs.io/en/latest/hands-on/docker/

Summary

The Research Pipeline



RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...

RR: Trying to Bridge the Gap

- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...
- **Sharing Code and Data**
  - ↪ `git`, Github, Bitbucket, Gitlab...

RR: Trying to Bridge the Gap

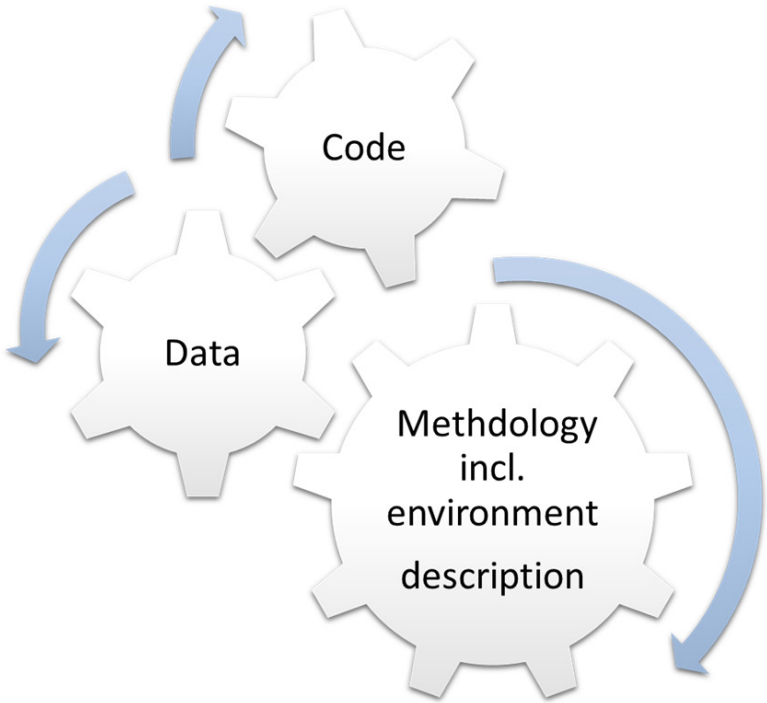
- Accurate, organized and **easy-to{read|take|share} Docs**
  - ↪ Markdown, mkdocs, org-mode, Read the Docs...
- **Sharing Code and Data**
  - ↪ `git`, Github, Bitbucket, Gitlab...
- **Mastering your environment clean and automated** by:
  - ↪ Using common building tools make, cmake etc.
  - ↪ Using a constrained environment
    - ✓ Sandboxed Ruby/Python,Vagrant, Docker
  - ↪ Automate its building through cross-platform recipes
  - ↪ Automatically test your recipes for Environment configuration

Sharing Code and Data

- Is this enough?
  1. Use a work ow that **documents both data and process**
  2. Use the machine readable **CSV format**
  3. Provide **raw** data and **meta** data, not just statistical outputs
  4. **Never** do data manipulation and statistical tests **by hand**
  5. Use R, Python or another free software to read and process raw data
     - ✓ ideally to produce complete reports with code, results and prose

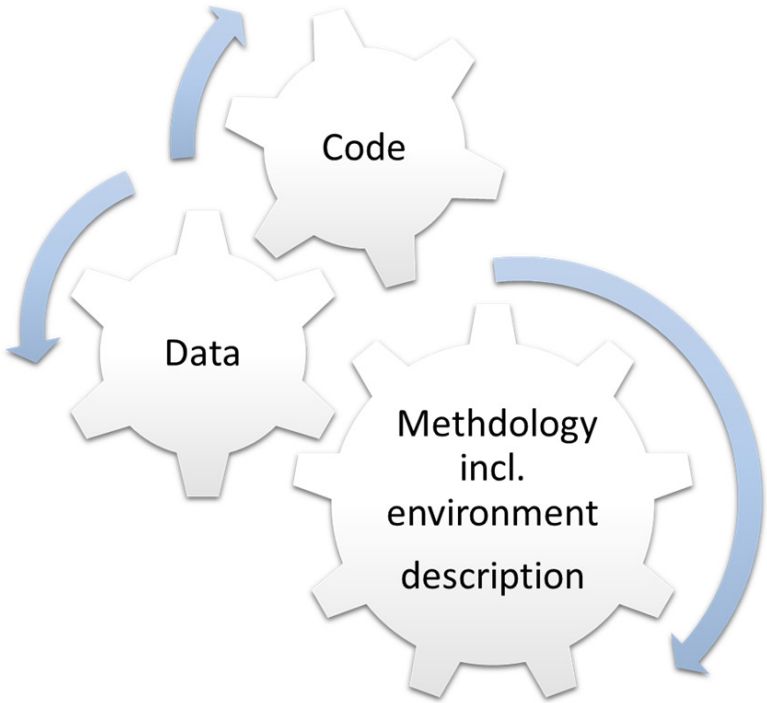
Reproducibility axes

- Always keep track of:
  - ↪ your **methodology**
  - ↪ your **code**
  - ↪ your (input) **data**
- Can you later come back and:
  - ↪ reproduce your experiment
  - ↪ including its environment
  - ↪ . . . and obtain the same results?



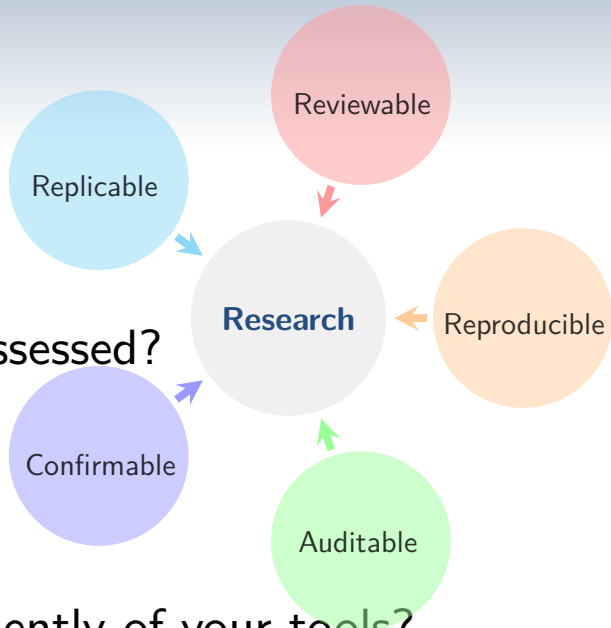
Reproducibility axes

- Always keep track of:
  - ↪ your **methodology**
  - ↪ your **code**
  - ↪ your (input) **data**
- Can you later come back and:
  - ↪ reproduce your experiment
  - ↪ including its environment
  - ↪ . . . and obtain the same results?



- If **not**, then **now** is the best time to start
  - ↪ documenting your processes
  - ↪ describing your environment (software **and hardware**!)
  - ↪ versioning and tagging your code and data
  - ↪ (. . . and keep backups of it all)

Reproducibility levels



Is your research[12]:

- reviewable
  - ↪ desc. of your methods can be independently assessed?
- replicable
  - ↪ are the tools available to duplicate the results?
- confirmable
  - ↪ can the main conclusions be attained independently of your tools?
- auditable
  - ↪ do you have records such that your research can be later defended?
  - ↪ . . . or differences between independent confirmations resolved?
- **open or reproducible**, such that
  - ↪ the procedures can be fully audited **and**
  - ↪ the results can be replicated or independently reproduced **and**
  - ↪ the results can be extended or the method applied to new problems

[12] ICERM Report 2013: "Reproducibility in Computational and Experimental Mathematics"

Open challenges

Sometimes you need to:

- Continue your computation **elsewhere**
  - ↪ another HPC node/cluster, supercomputer, cloud instance
- Continue your computation **in a different environment**
  - ↪ another software stack (*just* OS, *some* libraries / compiler flags)
- Use a **different version** of a commercial or community software

Open challenges

Sometimes you need to:

- Continue your computation **elsewhere**
  - ↪ another HPC node/cluster, supercomputer, cloud instance
- Continue your computation **in a different environment**
  - ↪ another software stack (*just* OS, *some* libraries / compiler flags)
- Use a **different version** of a commercial or community software

Are your results consistent?

Open challenges

Sometimes you need to:

- Continue your computation **elsewhere**
  - ↪ another HPC node/cluster, supercomputer, cloud instance
- Continue your computation **in a different environment**
  - ↪ another software stack (*just* OS, *some* libraries / compiler flags)
- Use a **different version** of a commercial or community software

Are your results consistent?

Be wary of:

- Comparing algorithms running on diverse hw. infrastructures
- Restarting calculation with the same code but on diff. sw. env.
- . . . different (usually newer. . .) version of the code

Open challenges

Sometimes you need to:

- Continue your computation **elsewhere**
  - ↪ another HPC node/cluster, supercomputer, cloud instance
- Continue your computation **in a different environment**
  - ↪ another software stack (*just* OS, *some* libraries / compiler flags)
- Use a **different version** of a commercial or community software

Are your results consistent?

Be wary of:

- Comparing algorithms running on diverse hw. infrastructures
- Restarting calculation with the same code but on diff. sw. env.
- . . . different (usually newer. . .) version of the code

Keep track of your environment changes!

Questions?

Sebastien Varrette
mail: Sebastien.Varrette@uni.lu
Office E-007
Campus Kirchberg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg