group meeting march 2013

Burkhard Ritter. March 8, 2013.

Contents

1. Git in 13½ minutes
    1. Intro to Version Control Systems
    2. Hands-on Git
    3. Hands-on Bitbucket
2. Everything not physics in computational physics
    1. Common workflow
    2. Provenance
    3. Data management
    4. Thoughts on a Python-centric workflow

Before we start

- Computational physics: develop software
- Not professionals in software development
- Spend little time thinking about software development
- Collaboration on code level is rare
- Established best practices and proven concepts: rarely adapted
- Examples
  - Revision control
  - Unit tests and test-driven development

Version Control Systems

- Every notable software project uses VCS
- Manage evolving code or documents (e.g. Latex)
- Indispensable when working together in a team
- Keep track of changes: "revisions"

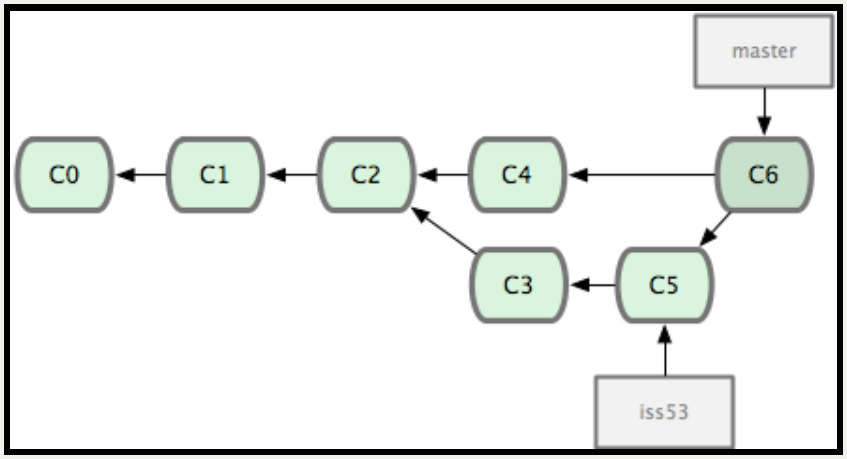


Image from: **Pro Git book**, **http://www.git-scm.com/book**

Version Control Systems

- Examples of VCS
  - Google Docs
  - Subversion
  - Distributed: Git, Mercurial
- Git
  - Originally developed for the Linux kernel
  - Made hugely popular by Github (**https://github.com/**)
- Version control can be complex and complicated
- Common use cases: dead simple, no reason not to use it

Hands-on: Git 1

```
$ mkdir my_cool_project
$ cd my_cool_project/
$ git init
Initialized empty Git repository in /Users/burkhard/my_cool_project/.git/
```

Hands-on: Git 2

```
$ vim funky_program.cpp
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add ..." to include in what will be committed)
#
#       funky_program.cpp
nothing added to commit but untracked files present (use "git add" to trac
k)
$ git add funky_program.cpp
$ git commit -a -m "Initial commit."
[master (root-commit) 1588d78] Initial commit.
 1 file changed, 4 insertions(+)
 create mode 100644 funky_program.cpp
```

Hands-on: Git 3

```
$ vim funky_program.cpp
$ git diff
diff --git a/funky_program.cpp b/funky_program.cpp
index c557e87..32e958e 100644
--- a/funky_program.cpp
+++ b/funky_program.cpp
@@ -1,4 +1,5 @@
 int main ()
 {
+    int whizbiz = 0;
     return 0;
 }
$ git commit -a -m "Implemented the new whizbiz feature."
[master 8859ecf] Implemented the new whizbiz feature.
 1 file changed, 1 insertion(+)
```

Hands-on: Git 4

```
$ git log
commit 8859ecf6b0cbcd29407ddbfde3bc0f3ae5c953b2
Author: Burkhard Ritter
Date:   Thu Mar 7 18:36:30 2013 -0700

    Implemented the new whizbiz feature.

commit 1588d78bb6ee615499441a76ab3a8fb6a62241c5
Author: Burkhard Ritter
Date:   Thu Mar 7 18:34:29 2013 -0700

    Initial commit.
```

Hands-on: Git 5

```
$ git help
[...verbose help message...]
$ git help tag
[...helpful manpage...]
$ git tag -a -m "Version 1." v1
$ git tag
v1
$ git describe
v1

$ vim funky_program.cpp
$ git commit -a -m "More exciting features."
[master 7684818] More exciting features.
 1 file changed, 1 insertion(+)
$ git describe
v1-1-g7684818
```

Hands-on: Git 6

```
$ git help diff
$ git diff head^
diff --git a/funky_program.cpp b/funky_program.cpp
index 32e958e..0403944 100644
--- a/funky_program.cpp
+++ b/funky_program.cpp
@@ -1,5 +1,6 @@
 int main ()
 {
+    bool evil_bug = true;
     int whizbiz = 0;
     return 0;
 }
```

Hands-on: Git 7

- Immediate advantages
  - Simple
  - Keep track of changes
  - No different copies of code floating around in directories
  - Which code produces which result
  - Keep track of progress
  - Track down bugs

Collaboration

- Distributed VCS: everybody has full copy of repository
- Different collaboration workflows possible
- More details: **Pro Git book**
- Simple example: Hagen and Kriemhild work together on an epic poem, the Nibelungenlied (in Latex)

...

```
hagen@worms:~/nibelungenlied$ git pull kriemhild
hagen@worms:~/nibelungenlied$ git add chapter4.tex
hagen@worms:~/nibelungenlied$ edit chapter3.tex
hagen@worms:~/nibelungenlied$ git commit -a
hagen@worms:~/nibelungenlied$ git push kriemhild
```

...

```
kriemhild@worms:~/nibelungenlied$ edit chapter3.tex
kriemhild@worms:~/nibelungenlied$ git commit -a
kriemhild@worms:~/nibelungenlied$ git pull hagen
kriemhild@worms:~/nibelungenlied$ git pull ssh://kriemhild@worms/home/hage
n/nibelungen
```

Collaboration

More complex example: central server / repository

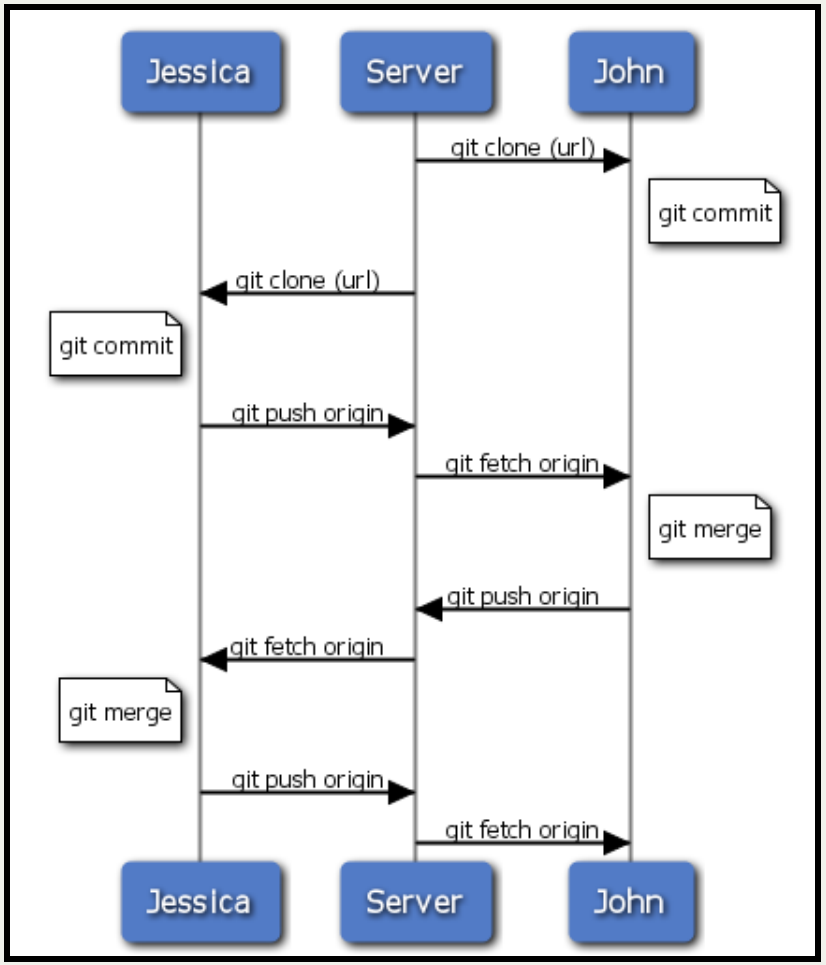


Image from: **Pro Git book**

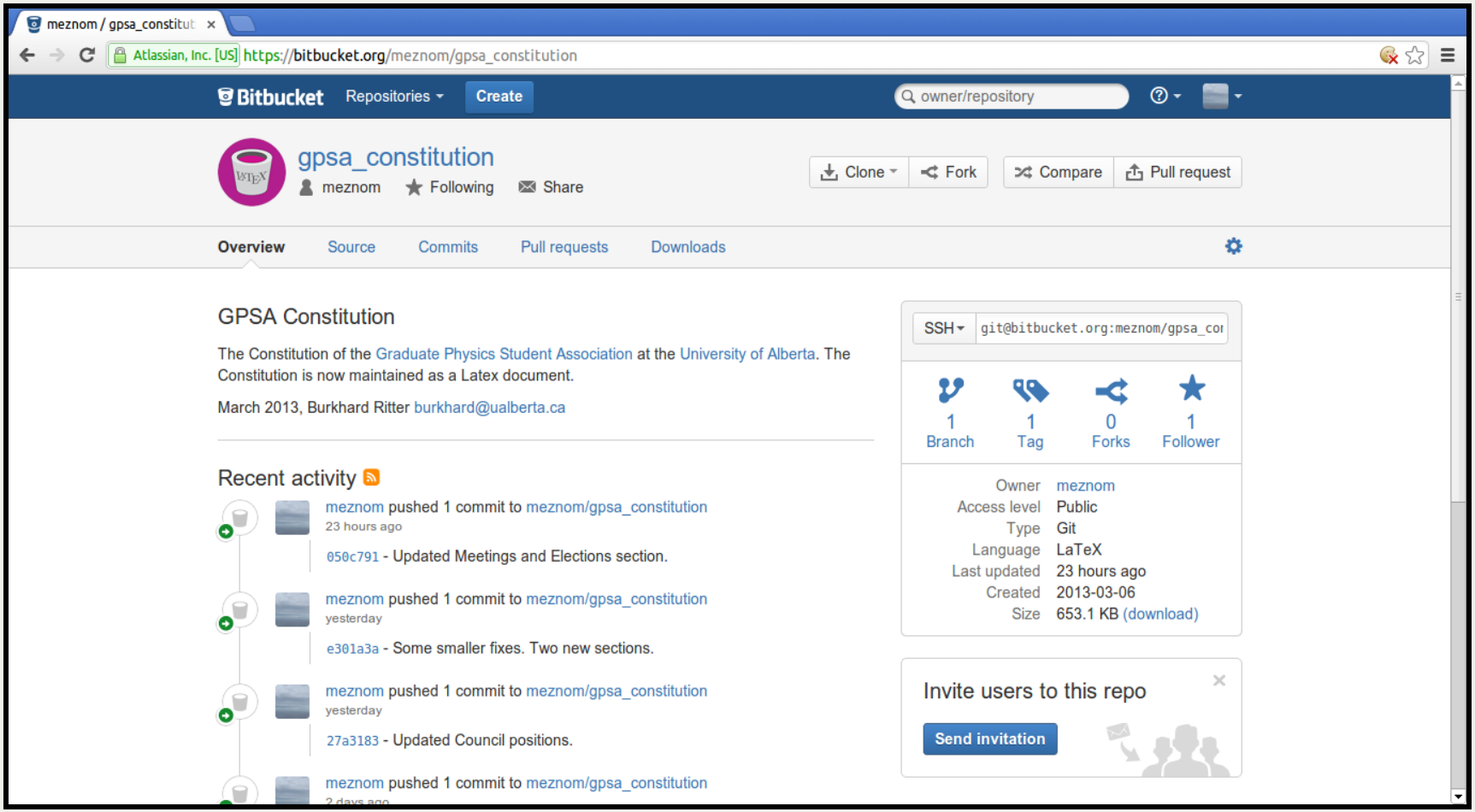
Bitbucket

- Github
  - Extremely popular
  - Social coding
  - "Revolutionalized open source" (**Wired article**)
  - Focus on individuals, not projects
  - Trend: publish everything, everything is public (!= free/libre software)
- Bitbucket
  - Similar, but not hip
  - Unlimited private repositories, 5 collaborators
  - Unlimited academic plan!
- Could or should our code be public?

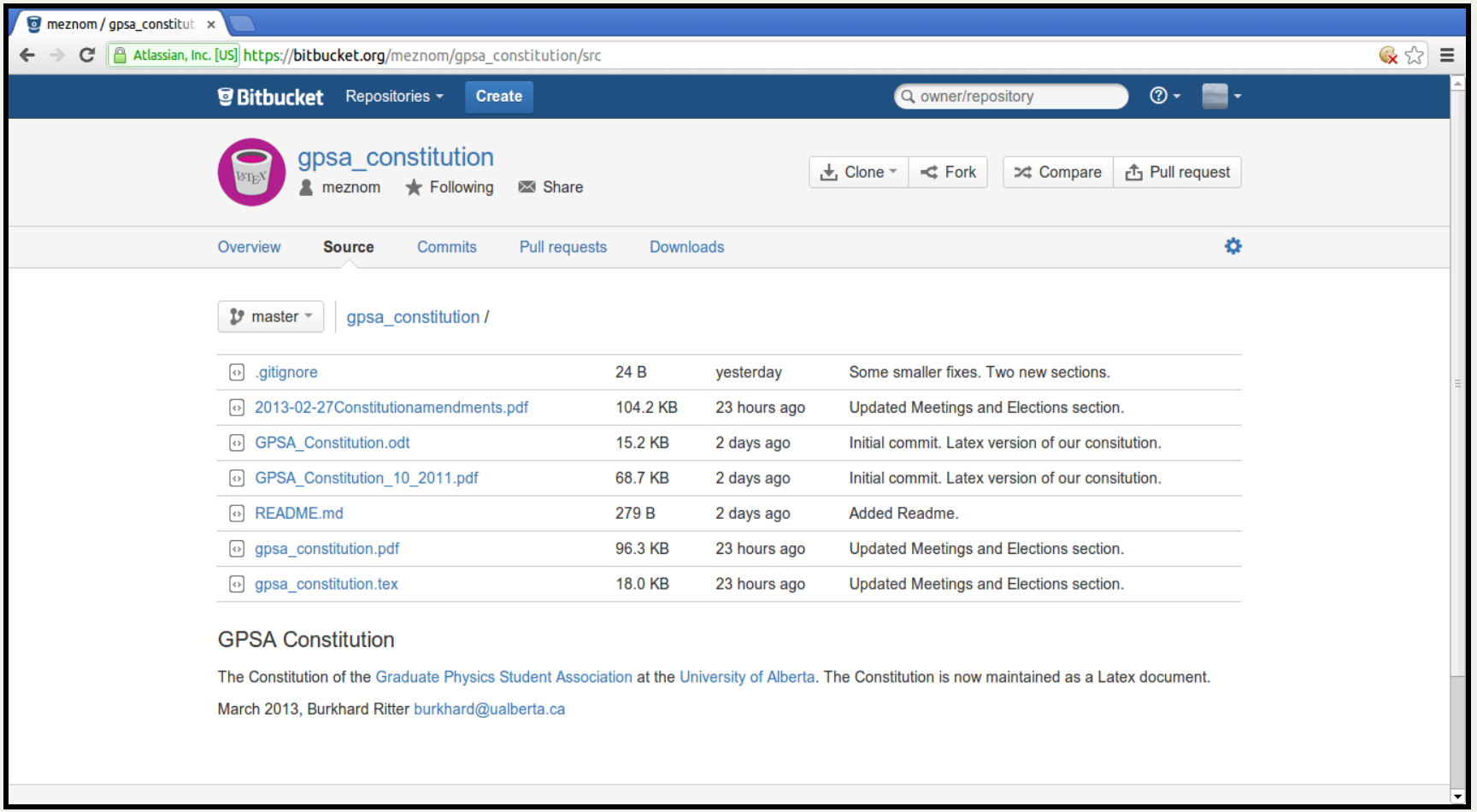
Bitbucket

- Why use Bitbucket?
  - Even easier to use Git
  - Web GUI
  - Backup
  - Sync between computers
  - Collaborate
    - Direct write access
    - Fork, create pull request
    - Teams

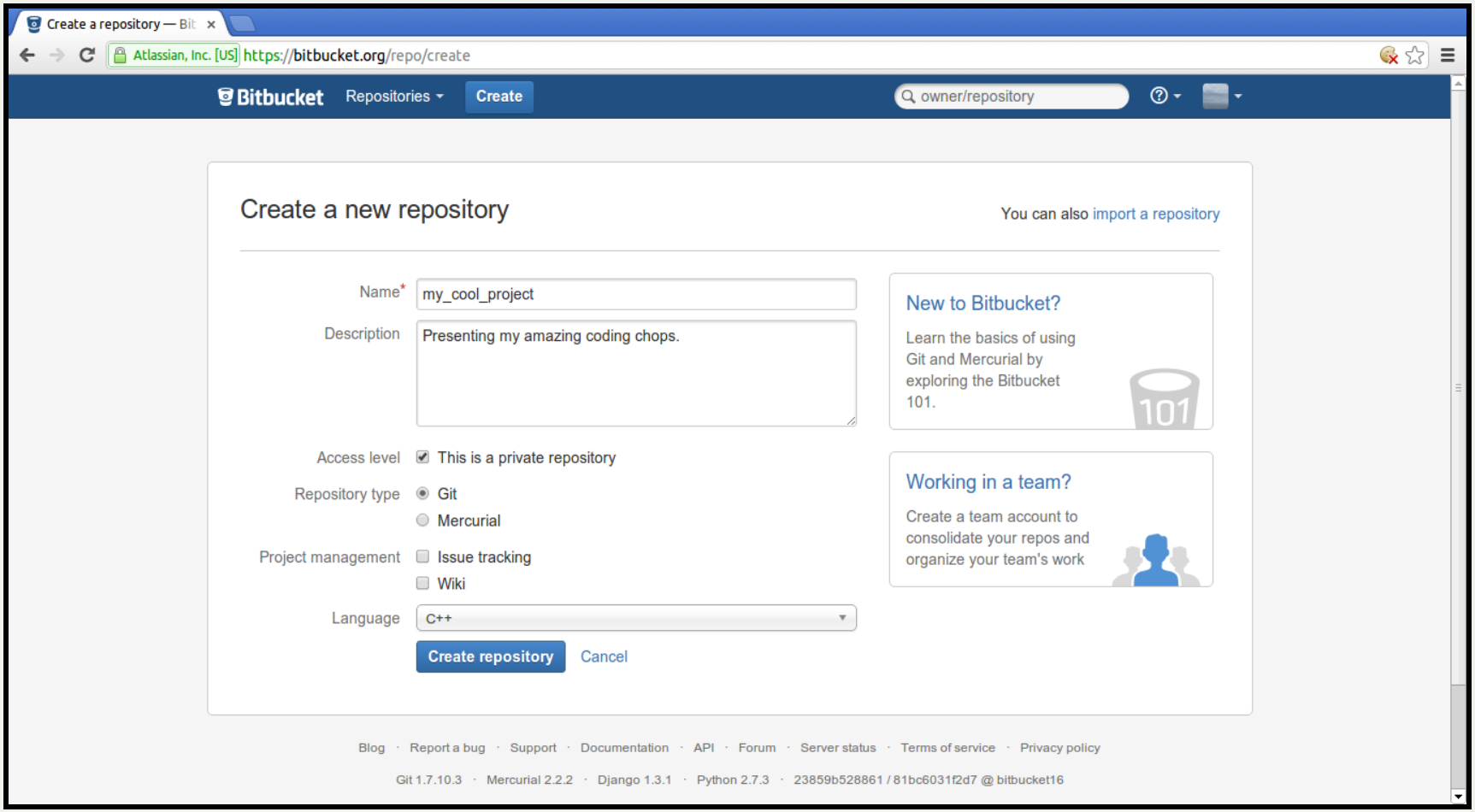
Hands-on: Bitbucket 1



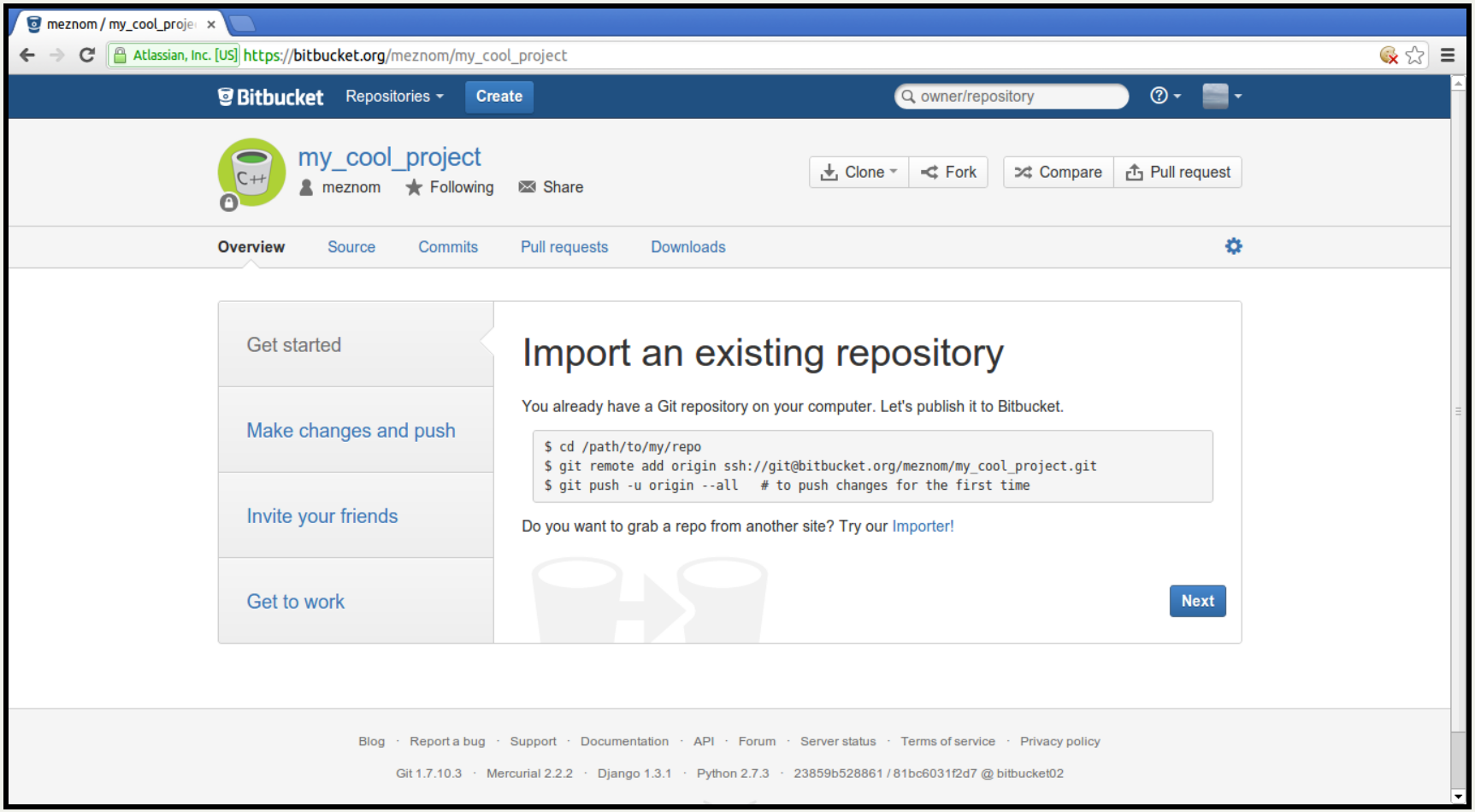
Hands-on: Bitbucket 2



Hands-on: Bitbucket 3



Hands-on: Bitbucket 4



Hands-on: Bitbucket 5

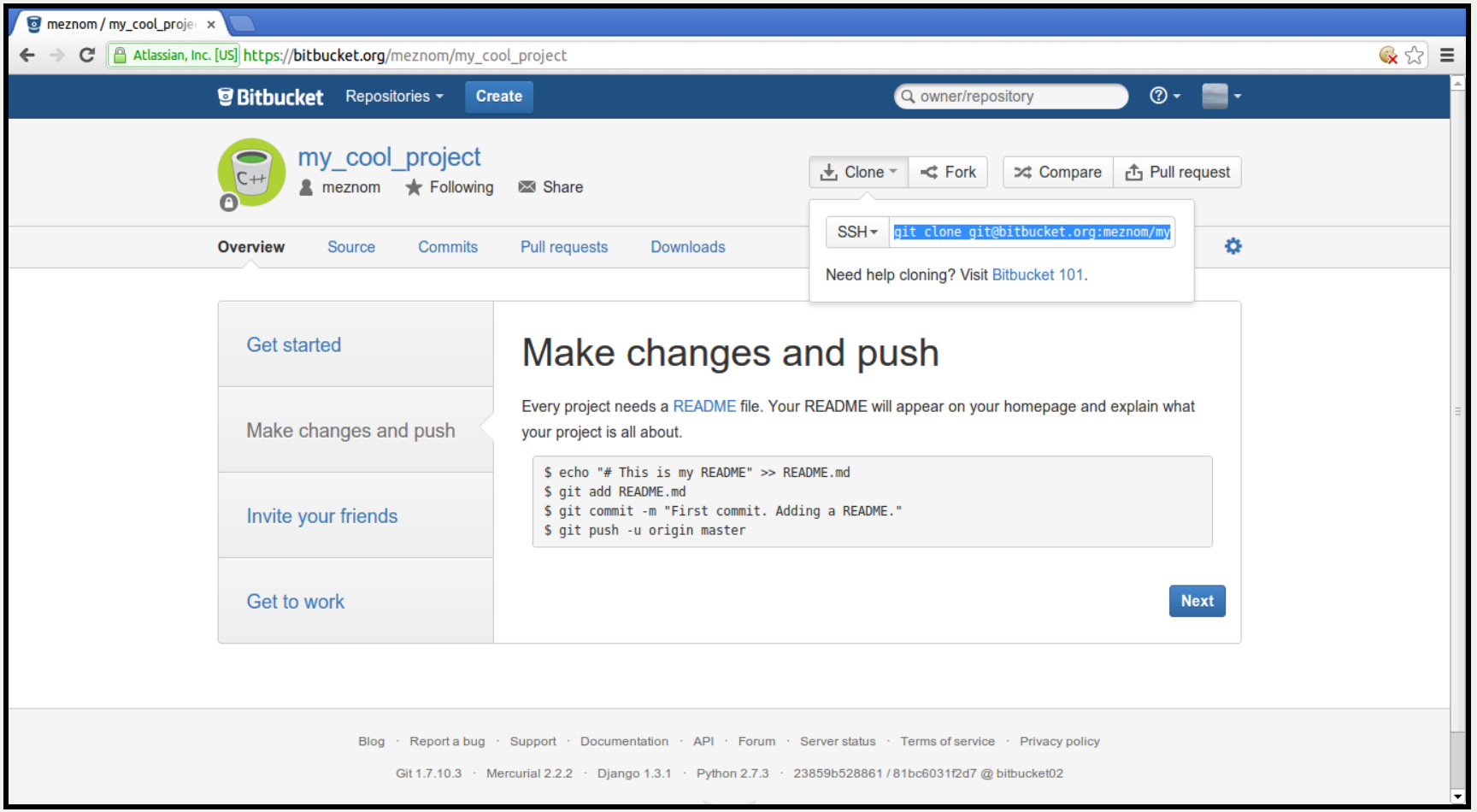
```
burkhard@macheath:~/my_cool_project$ git remote add origin ssh://git@bitbu
cket.org/meznom/my_cool_project.git

burkhard@macheath:~/my_cool_project$ git push -u origin --all
[...]

burkhard@macheath:~/my_cool_project$ git status
# On branch master
nothing to commit, working directory clean

burkhard@macheath:~/my_cool_project$ git pull
Already up-to-date.
```

Hands-on: Bitbucket 6



Hands-on: Bitbucket 7

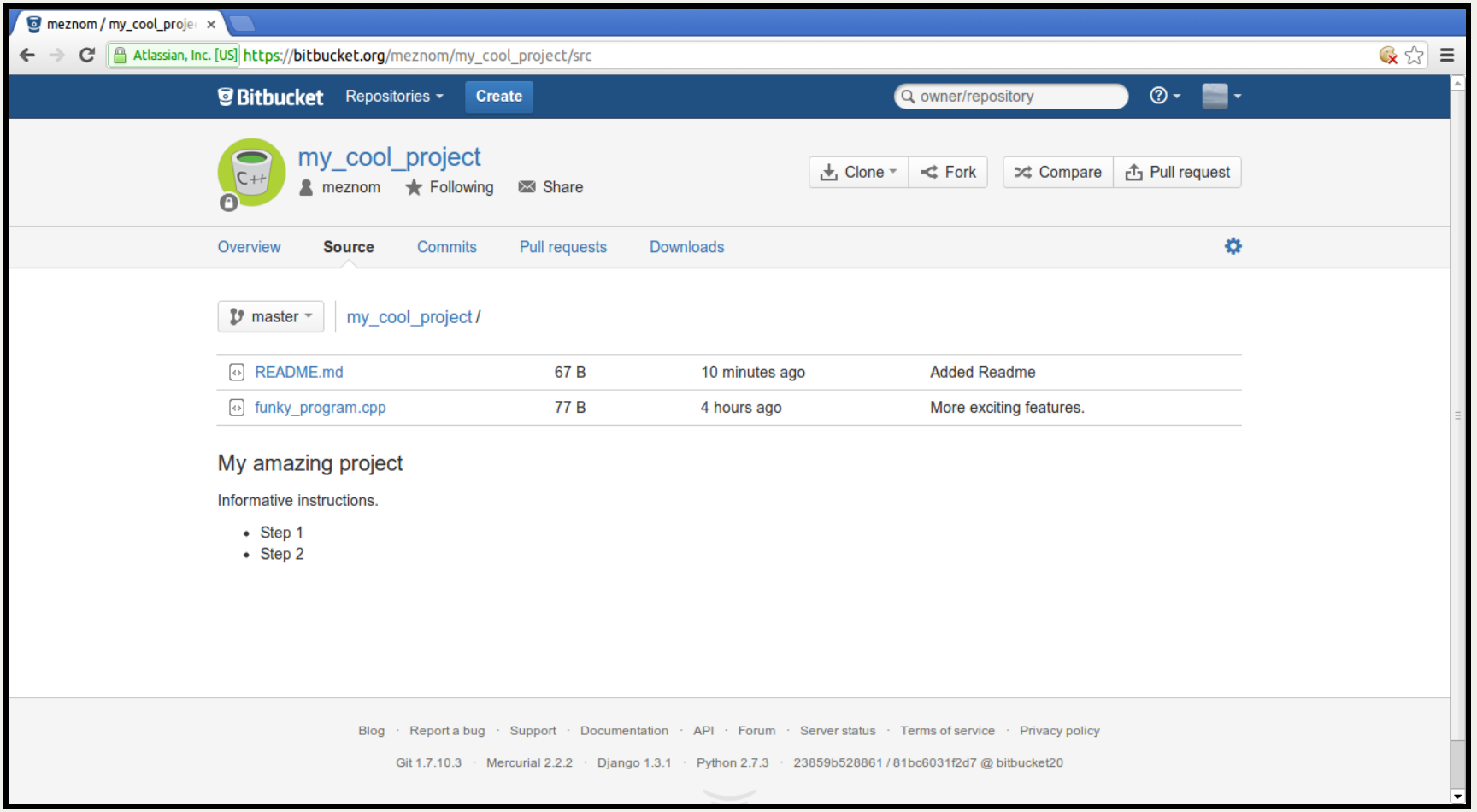
```
burkhard@lise:~$ git clone git@bitbucket.org:meznom/my_cool_project.git
[...]
burkhard@lise:~$ cd my_cool_project/
burkhard@lise:~/my_cool_project$ vim README.md
burkhard@lise:~/my_cool_project$ git add README.md
burkhard@lise:~/my_cool_project$ git commit -a -m "Added Readme"
[...]
burkhard@lise:~/my_cool_project$ git push
[...]
```

...

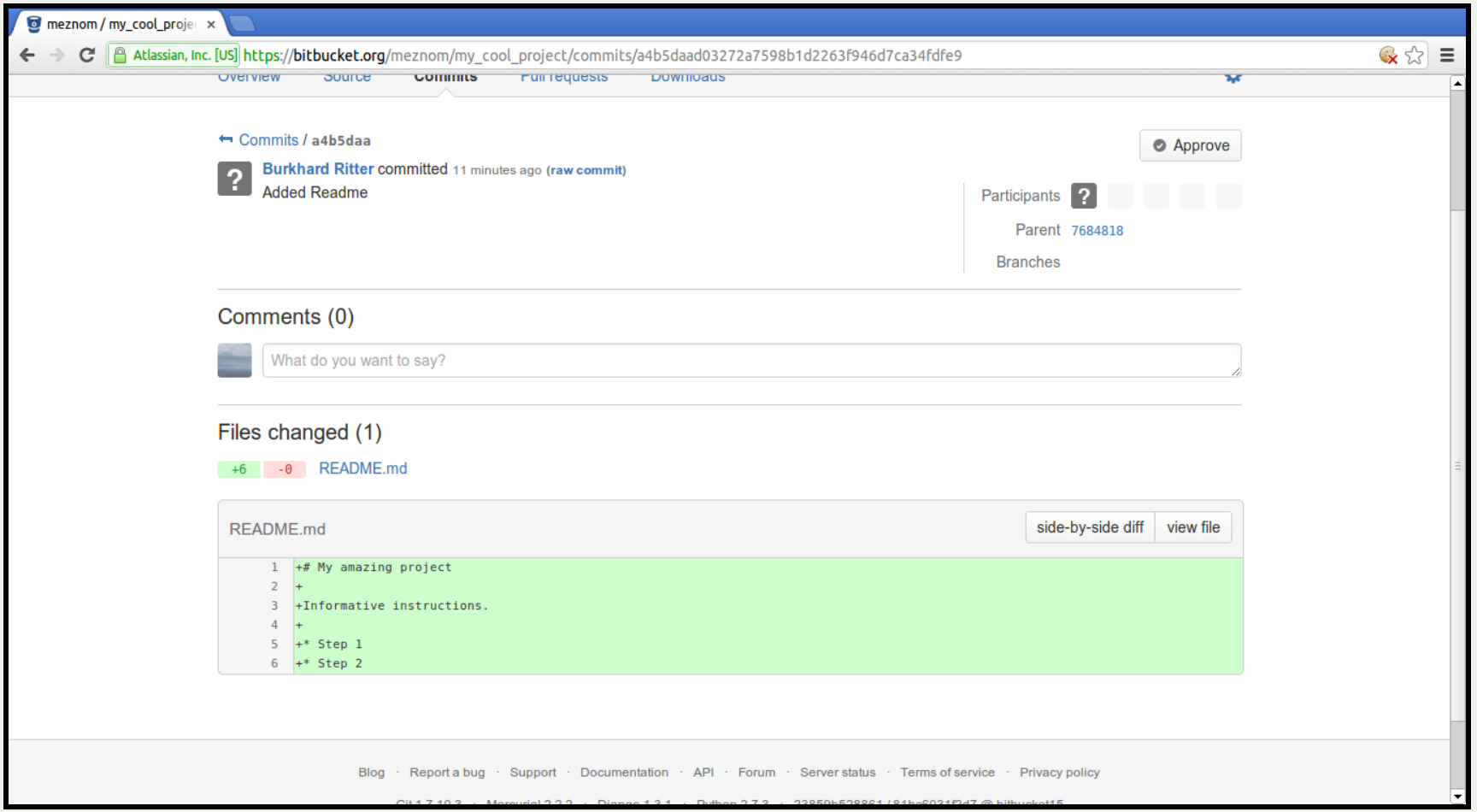
```
burkhard@macheath:~/my_cool_project$ git pull
[...]
From ssh://bitbucket.org/meznom/my_cool_project
   7684818..a4b5daa  master     -> origin/master
Updating 7684818..a4b5daa
[...]
burkhard@macheath:~/my_cool_project$ git log -n 1
commit a4b5daad03272a7598b1d2263f946d7ca34fdfe9
Author: Burkhard Ritter
Date:   Thu Mar 7 22:37:55 2013 -0700

    Added Readme
```

Hands-on: Bitbucket 8



Hands-on: Bitbucket 9



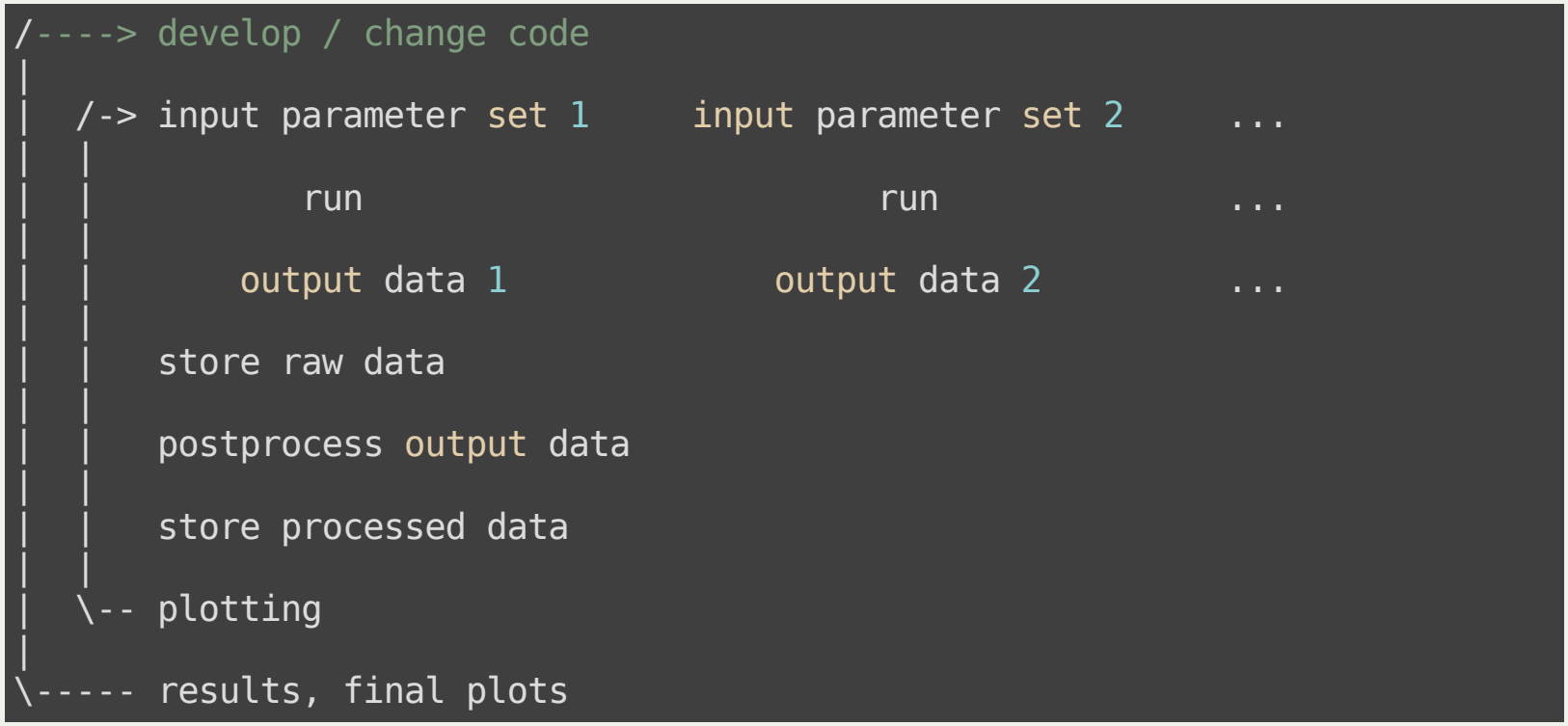
Everything not physics in computational physics

- In everyday work: activities, procedures, issues not related to physics
- Similar for all of us
- Problem and project independent
- Examples
  - Software management
  - Data management
  - Plotting
  - Publishing
- Similarities are a result of common workflow

Everything not physics in computational physics

- We do not often talk about these non-physics issues
- Goal
  - Exchange ideas
  - Establish best practices
  - Possibly share code
- This talk
  - Open discussion
  - No answers

Common workflow



Common workflow

- Common aspects
  - Fast evolving code
  - Results depend crucially on code version
  - Multiple input parameter sets (possibly: parallelize)
  - Input set -> one output data point
  - Postprocess output
  - Plot processed output
  - Store raw and processed data, results and plots
- Opportunities for best practices and collaboration

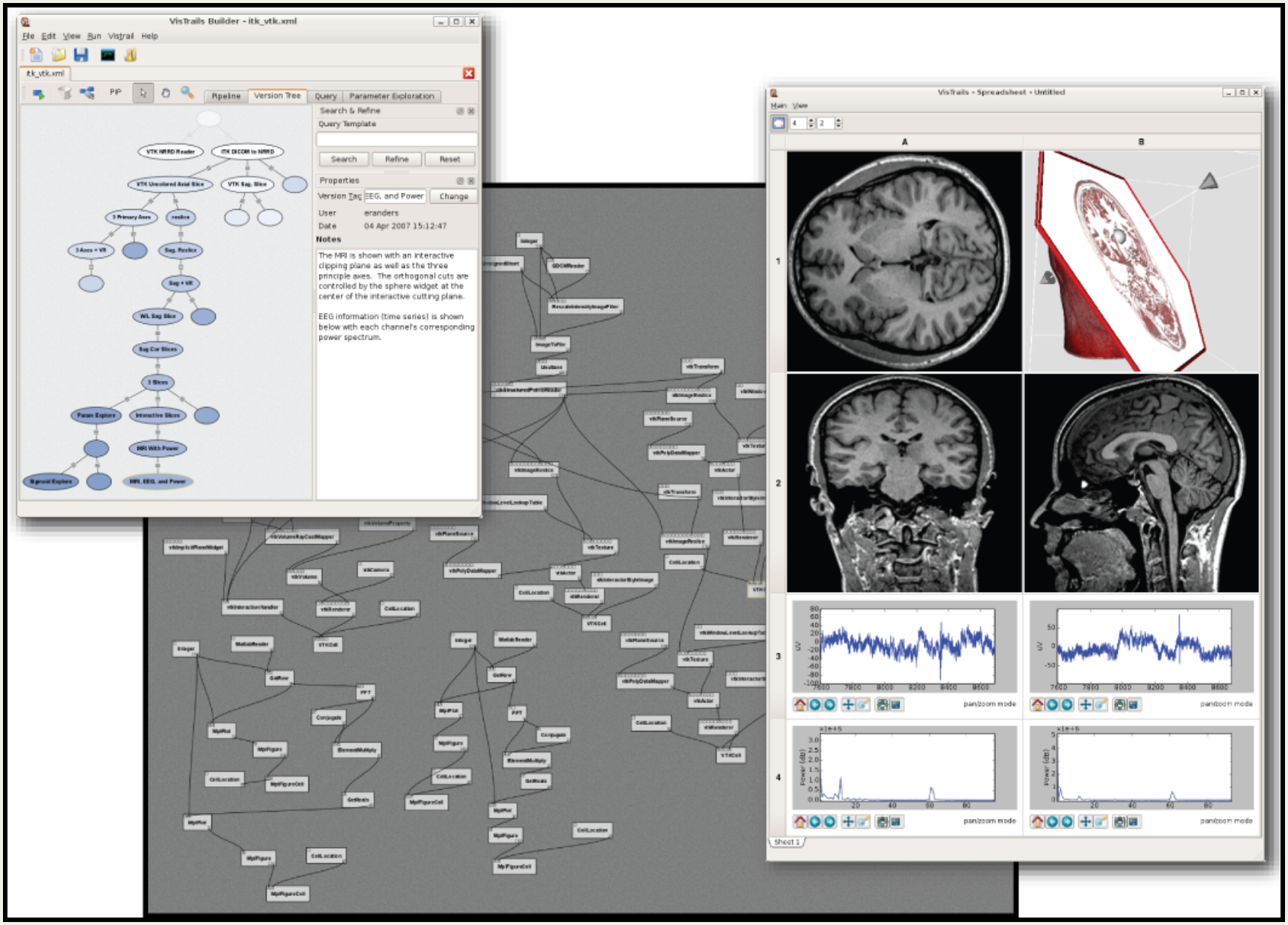
Provenance

- Verifiablility and reproducibility are at the heart of science
- Anybody at any time in the future
  - Take any (published) result
  - Go back and be able to reproduce and verify it
- Requires
  - Meticulously document every single step that led to the result
  - Quite involved for computer simulations

Provenance

- Pushed by Alps (ETH Zürich, **http://alps.comp-phys.org/**)
- VisTrails
  - **vistrails.org**, Polytechnic Institute of New York University
  - "Scientific workflow and provenance system"
  - Evolve workflows, document workflows
  - Example
    - Graph in paper
    - VisTrails opens workflow
    - Reproduce (rerun simulations)
- Complete and complex provenance system (too complex?)
- Take inspiration and learn

Provenance


VisTrails Builder - itk_vtk.xml
File Edit View Run Vistrail Help
itk_vtk.xml
Pipeline
Version Tree
Query
Parameter Exploration
Search & Refine
Query Template
Search
Refine
Reset
Properties
Version Tag EEG, and Power
Change
User eranders
Date 04 Apr 2007 15:12:47
Notes
The MRI is shown with an interactive clipping plane as well as the three principle axes. The orthogonal cuts are controlled by the sphere widget at the center of the interactive cutting plane.
EEG information (time series) is shown below with each channel's corresponding power spectrum.
VisTrails - Spreadsheet - Untitled
Main View
A
B
panizoom mode
Sheet 1

Provenance

- For every result, keep
  - Code version, `git describe`
  - Input parameters
  - Scripts / programs (or their configuration) for processing raw data
  - Plotting scripts / instructions
- Central code repository
- Publish code (i.e. open source)

Data management

> “
>
> *In the long run, your data matters more than your code. It's worth investing some effort to keep your data in good shape for years to come.*
>
> ”

Konrad Hinsen, "Caring for Your Data," Computing in Science and Engineering, vol. 14, no. 6, pp. 70-74, Nov.-Dec., 2012; available on **computer.org**

Data management

- Motiviation
  - Publications: data shown and discussed
  - Traditionally
    - Data management low priority (e.g. processing, storage)
    - Data formats undocumented
    - Format conversion error prone
    - Down the road: hard or impossible to interpret data

Data management

- Data model design
  - Equivalent to software design
  - Describe data model in abstract but plain language
    - Equivalent to pseudo code
    - Specification
    - Documentation
  - Guidelines
    - Avoid redundancy
    - Keep extensibility in mind
    - More details in article

Data management

- Distinguish data model and representations of data model
- Representations
  - In memory (i.e. in program code)
  - On disk (i.e. file formats)
- In memory
  - Code easier to understand
  - Encourages code modularity
  - Different representations in different languages
- On disk: file formats
  - Different formats for different requirements
    - Binary for performance, e.g. HDF5
    - Ascii for readability, e.g. XML, JSON
  - Same data model: convert between formats easily and without loss

Data management

Example: XML

```
<molecule name="water">
  <atoms>O H1 H2</atoms>
  <bonds>
    <bond atoms="O H1" order=1 />
    <bond atoms="O H2" order=2 />
  </bonds>
</molecule>
```

Example: JSON

```
{
  "type": "molecule",
  "name": "water",
  "atoms": ["O", "H1", "H2"],
  "bonds": [{"order": 1, "atoms": ["O", "H1"]},
            {"order": 1, "atoms": ["O", "H2"]}]
}
```

Data management

- Can we establish guidelines and best practices?
- Document our data models and formats
- Central data repository?
- Same class of problems: similar data models
  - E.g. Monte Carlo
- Use standardized file formats
  - HDF5
  - JSON
- Possibly: share code for data handling, input, output

Data management

Examples / ideas for Monte Carlo: HDF 5

```
/experiment
  attributes: id, description
  /measurement
    attributes: count
    /0
      attributes: config
      /observables
        /Magnetization
          /data
          /jack
          /bins
      /run
        attributes: count
        /0
          /observables
            /Magnetization
              /data
```

Data management

Examples / ideas for Monte Carlo: JSON

```
{
  "info": {
    "program": "SSEMonteCarlo",
    "version": "unknown",
    "state": "done",
    "seedvalue": 42,
    "run": {
      "0": {
        "startdate": "2013-03-08T09:41:43Z",
        "enddate": "2013-03-08T09:41:43Z"
      },
      "1": {
        "startdate": "2013-03-08T09:46:13Z",
        "enddate": "2013-03-08T09:46:13Z"
      }
    }
  },
```

Data management

Examples / ideas for Monte Carlo: JSON (continued)

```
"type": "ssemontecarlo.montecarlo.MonteCarlo",
"params": {
  "type": "ssemontecarlo.montecarlo.Struct",
  "N": 10,
  "beta": 1,
  "h": 10,
  "J": {
    "type": "ssemontecarlo.montecarlo.NNAFHeisenbergInteraction",
    "J": 1
  }
},
"mcparams": {
  "type": "ssemontecarlo.montecarlo.Struct",
  "t_warmup": 100,
  ...
},
"observables": [...],
```

Data management

Examples / ideas for Monte Carlo: JSON (continued)


```
  "data": {
    "ExpansionOrder": {
      "mean": 246.39998046875,
      "error": 0.9593617279141327,
      "binCount": 200,
      "binSize": 1
    },
    "Magnetization": {
      "mean": -0.060000009536743164,
      "error": 0.18963782783471672,
      "binCount": 200,
      "binSize": 1
    }
  }
}
```

A Python-centric workflow

- IPython is awesome (**ipython.org**)
- Browser-based notebooks
  - Similar to Mathematica
  - Might be good fit for some workflows
- Comprehensive library and tools for parallelization

A Python-centric workflow

- Idea: (versioned) Python scripts control complete workflow
  - Set input parameters
  - Run program (in parallel)
  - Postprocess data
  - Store raw and processed data (possibly in MongoDB)
  - Do plotting (matplotlib)
- Core program itself
  - Written in Python (SciPy; PyPy -- **pypy.org**)
  - Written in C++, compiled as Python module (e.g. with Boost)

The End