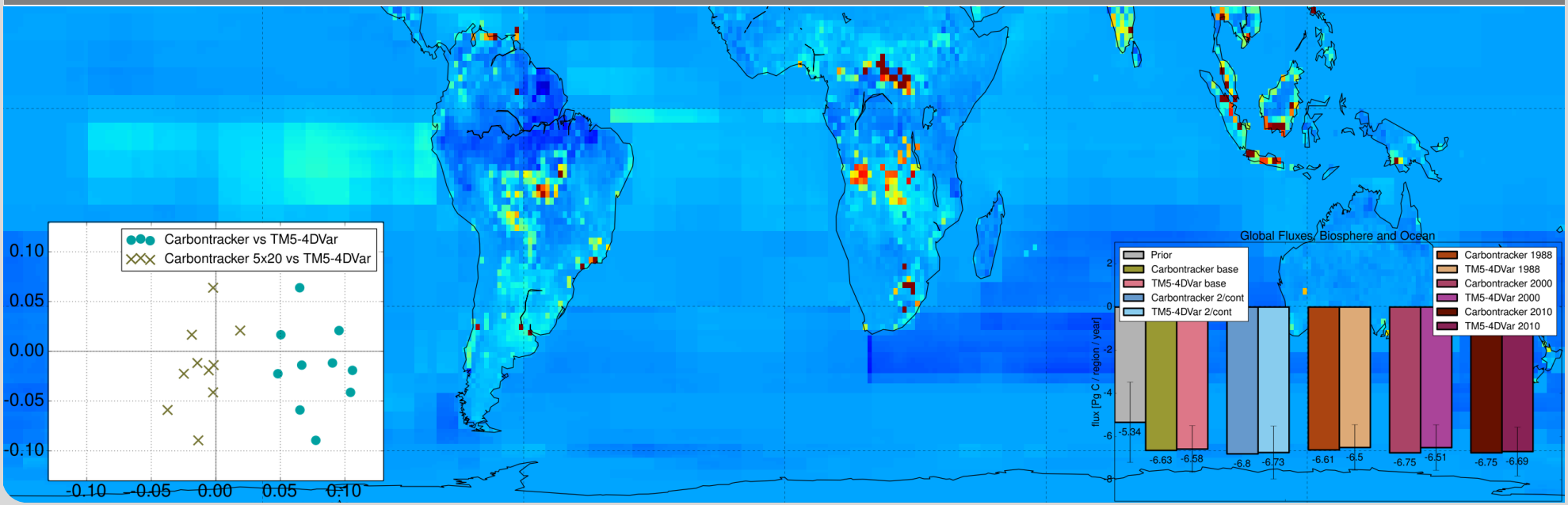
Karlsruher Institut für Technologie

hg init science

| 23. Juli 2015

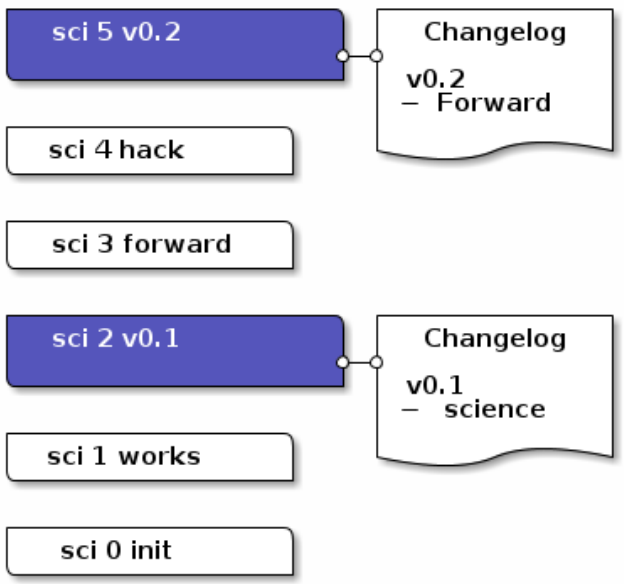
IMK-ASF, REMOTEC GHG-GROUP, VIA ANDRÉ BUTZ



Goals of my talk

- convey solid hg skills
- contrast hg usage to subversion
- present advantages of hg for software development in science

Step back: Backup Folders



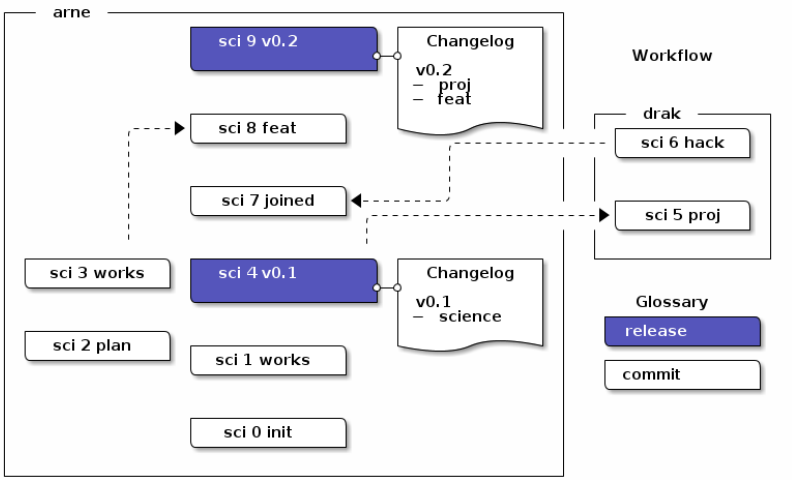
Process:

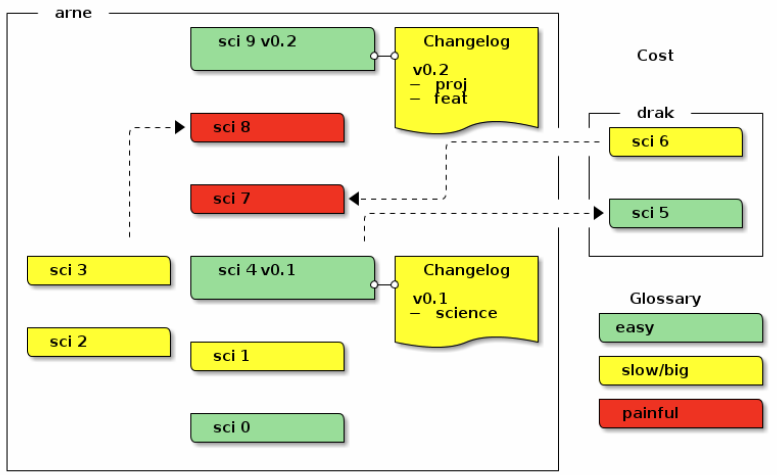
- One folder per version

Problems:

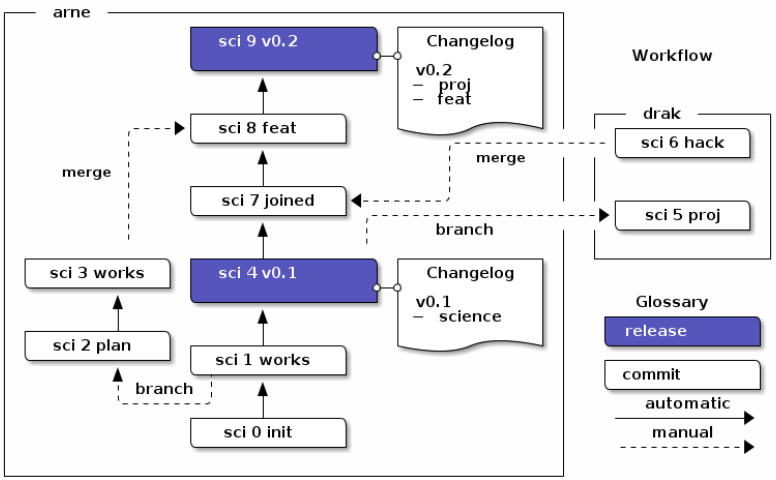
- Slow
- Bookkeeping
- non-linear changes
- diskspace
- collaborating

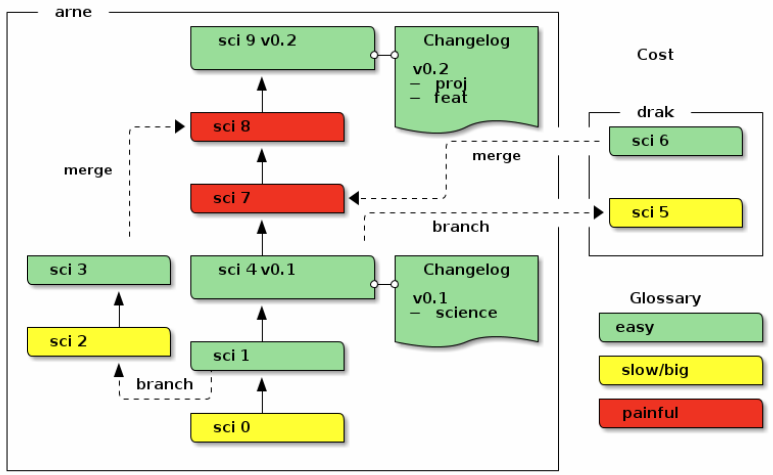
Workflow Cost: Backup Folders



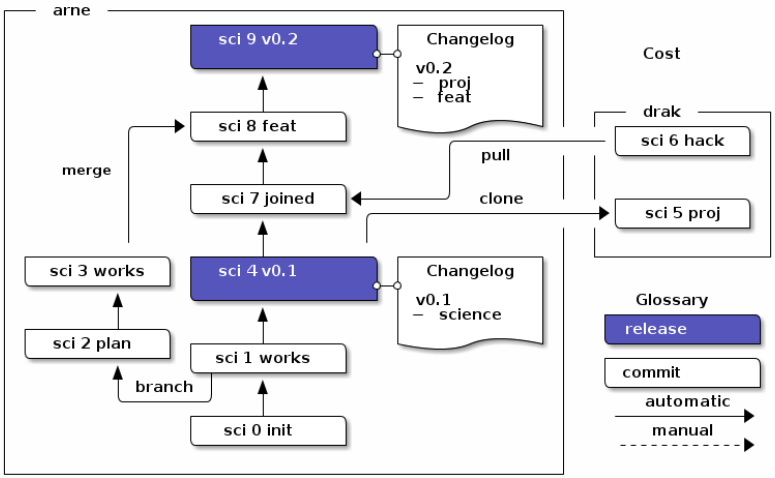


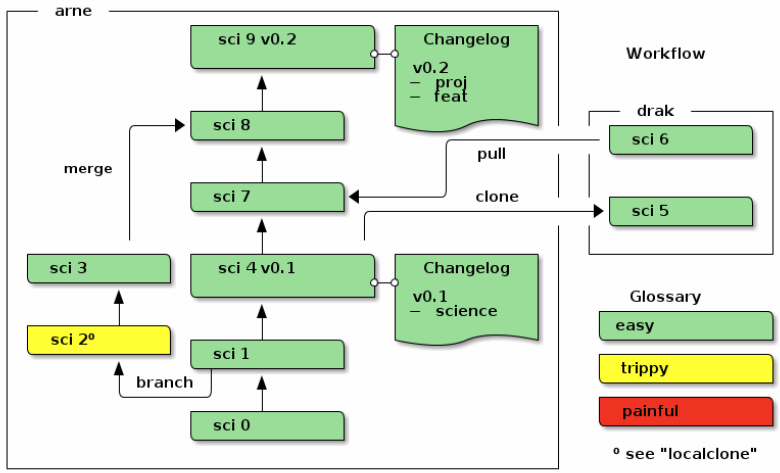
Workflow Cost: Subversion





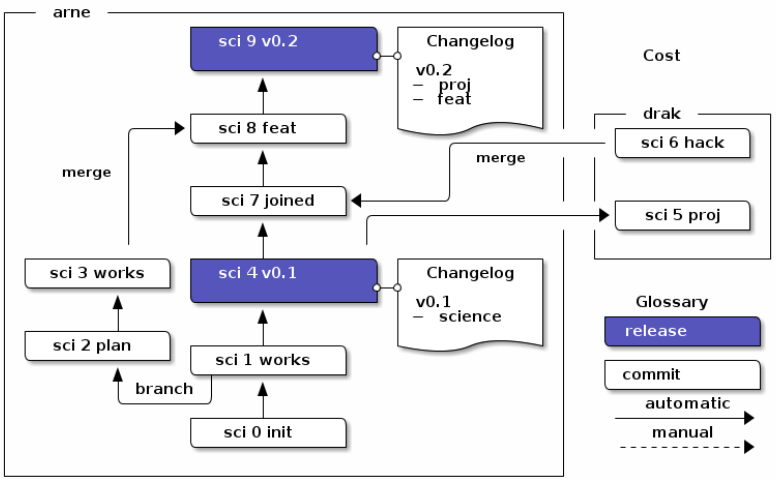
Workflow Cost: Git

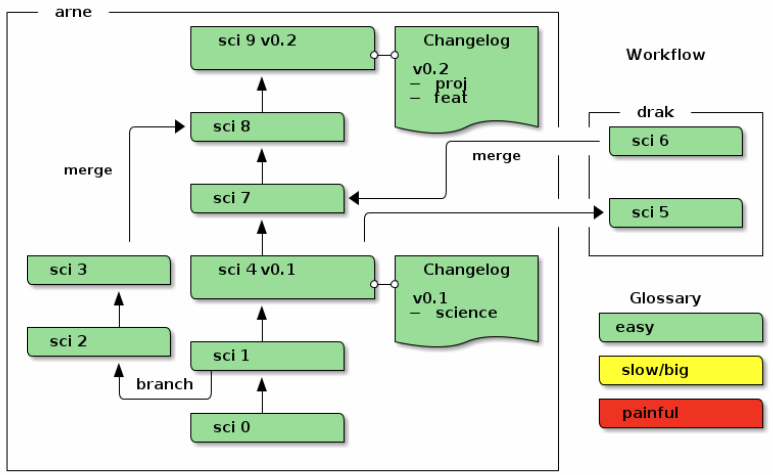




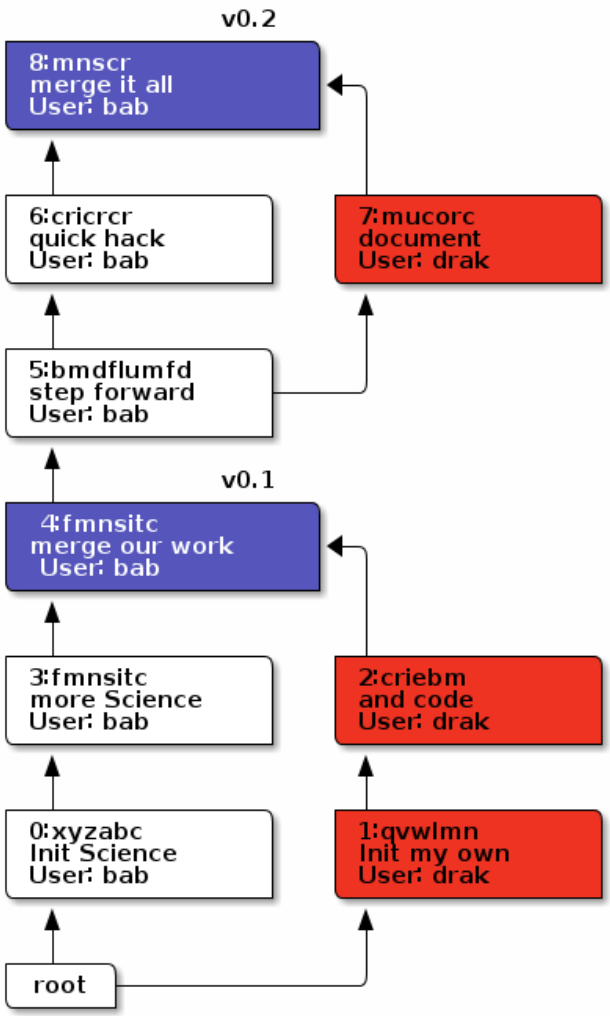
[0]localclone: When using local testing clones in git, an additional local branch is required. See DVCS workflow failures <draketo.de/node/561>

Workflow Cost: Mercurial



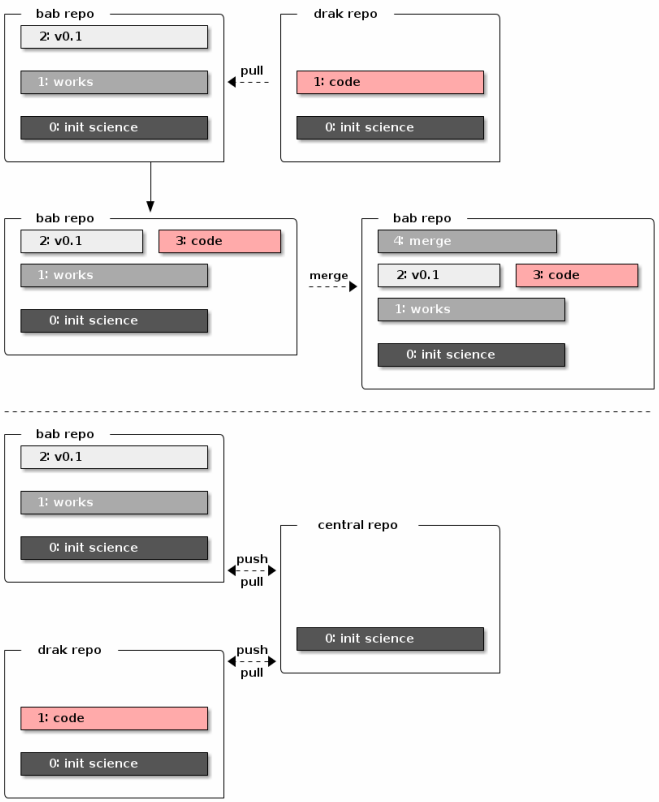


Branch and Merge: Asynchronous work



- Cheap snapshots
  - when you take a break
  - when something works
  - before you start debugging
  - ...
- Diff, Log and Merge
  - What changed?
  - When did we do *this*?
  - Combine our work!

Push and Pull: Collaborate



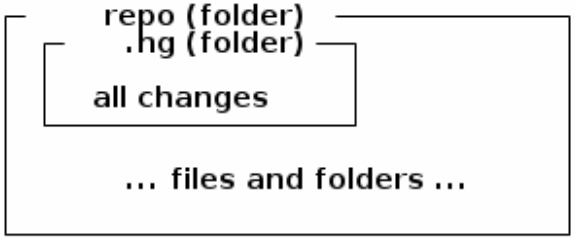
- exchange
- central repo
- push and pull
- merge

repo (“repository”):

A folder with history and a working copy.

working copy:

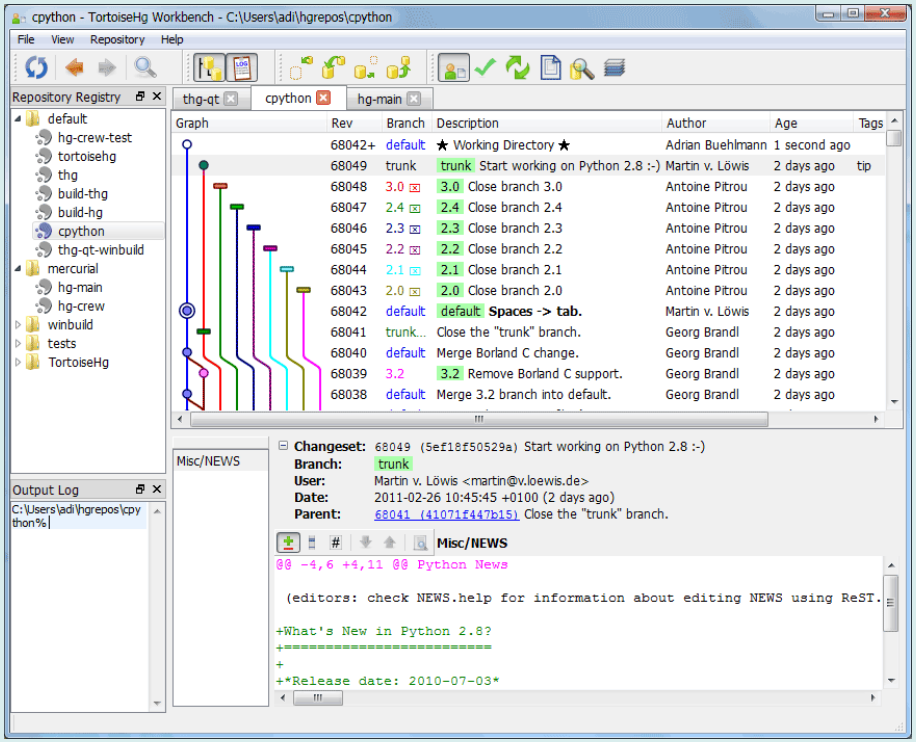
The files and folders you currently work on.



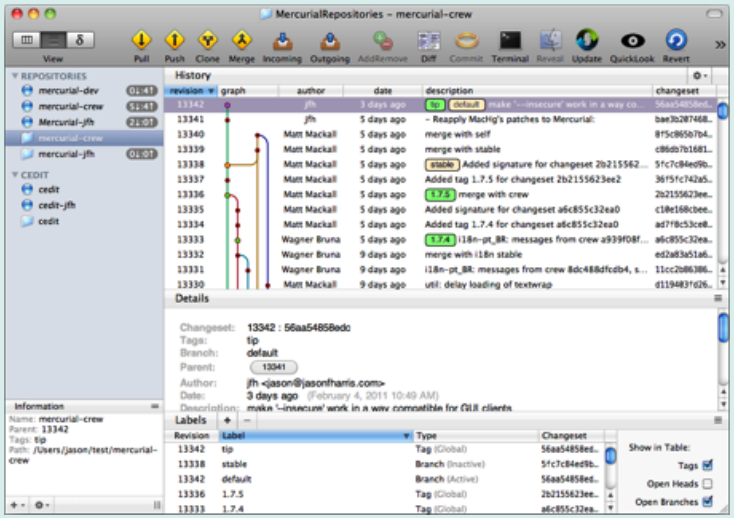
Summary

- log: automated bookkeeping
- commit: branch and merge
- collaborate: pull and push

GUI



TortoiseHg (Windows, GNU/Linux, MacOSX)
tortoisehg.bitbucket.org



MacHg (well, Mac :))
jasonfharris.com/machg/

I will not show these here. See their tutorials.

Local operation

Username in config!

```
[ui]
username = NAME <me@x.org>
```

config file

- Windows: %USERPROFILE%\mercurial.ini
- MacOSX, GNU/Linux: $HOME/.hgrc
- Per repo: <repo>/.hg/hgrc

hg help: essential

- `hg help <cmd>`
- first line: Usage
- end: The options

See also:

```
hg help templates
hg help revsets
hg help hgrc
hg help # topics
```

help example

Usage:

```
LC_ALL=C hg help commit | head -n 1
hg commit [OPTION]... [FILE]...
```

Options:

```
LC_ALL=C hg help commit | tail -n 7 | head -n 3
-l --logfile FILE        read commit message from file
-d --date DATE           record the specified date as commit dat
-u --user USER           record the specified user as committer
```

Create a new repository

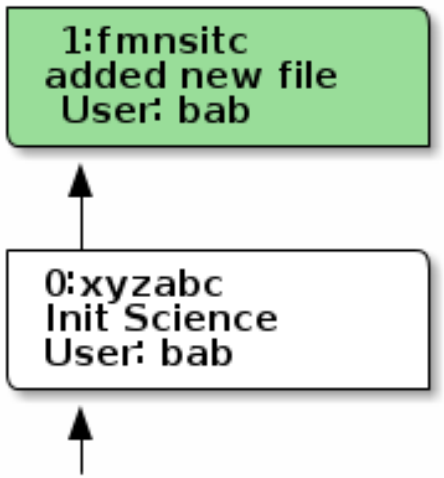
hg init

Example: Init

```
hg init testproject
cd testproject
```

Save and Inspect versions

- `add`: new files
- `commit`: save



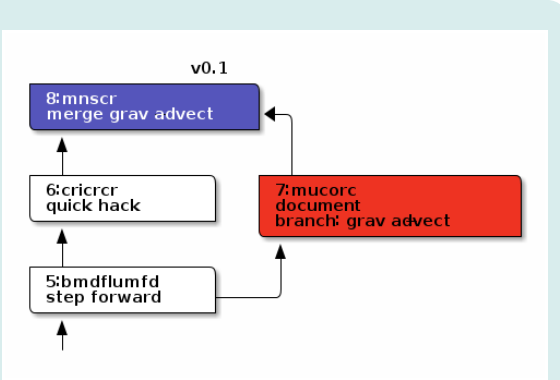
- `status`: modified files
- `diff`: what would be committed
- `log`: show versions (`-G`: Graph)

```
hg log -G -r -1:-3
@  Änderung:         46:4b44ffc841aa
|  Marke:           tip
|  Nutzer:          Arne Babenhauserheide <arne.babenhauserheide@kit.
|  Datum:           Thu Jul 23 19:34:48 2015 +0200
|  Zusammenfassung: Added signature for changeset 6a63a5c6594a
|
o  Änderung:         45:6a63a5c6594a
|  Nutzer:          Arne Babenhauserheide <arne.babenhauserheide@kit.
|  Datum:           Thu Jul 23 19:34:46 2015 +0200
|  Zusammenfassung: Added tag 0.2.3 for changeset a770e6cefebc
|
o  Änderung:         44:a770e6cefebc
|  Marke:           0.2.3
|  Nutzer:          Arne Babenhauserheide <arne.babenhauserheide@kit.
|  Datum:           Thu Jul 23 19:33:40 2015 +0200
|  Zusammenfassung: testing clones
|
```

Example: Save and Inspect

```
echo "co2" > tracers.txt
hg add tracers.txt
hg status
hg diff
hg commit -m "added co2 as tracer"
hg log -G # enable graphlog extension
          # for old versions of hg
```

Branch and Tag



- `update`: Change the working copy to another version
- `branch`: Collect the following commits under a name
- `merge`: Combine two branches

Example: Branch and Tag

```
echo "co" >> tracers.txt
hg commit -m "added co"
hg update 0
echo "ch4" >> tracers.txt
hg ci -m "added ch4"
hg log -G
hg merge
hg diff
hg commiit -m "merged co and ch4"
hg log -G
```

Exchange and Share

Clone and Serve

- `clone`: Copy a repository and remember the sourec
- `serve`: Share over the network for clone, pull and push

```
hg clone project experiment # local folder
hg clone ssh://imkpcabu/path/to/repo # friend machine
hg clone ssh://hg@bitbucket.org/remotec/sim # ssh server
hg clone http://draketo.de/proj/hgsite # static http
```

Pull and Push

- `pull`: get versions
- `pull -u`: get all versions and update the working copy[1]
- `push`: copy versions elsewhere
- `incoming/outgoing`: what would be pulled or pushed

[1]: merges uncommited changes: `diff` and `commit` before `pull -u`!

Example: Exchange

```
cd ..
hg clone testproject testethane
cd testethane
echo "c2h6" >> tracers.txt
hg ci -m "added ethane"
hg log -G
hg outgoing
cd ../testproject
hg incoming ../testethane
hg status # no changes to commit
hg pull -u
hg log -G
```

Bitbucket

SSH-key

```
ssh-keygen # hit enter a few times
cat ~/.ssh/id_rsa.pub
```

Upload on `https://bitbucket.org/account/user/USERNAME/ssh-keys/`

Persistent authentication with the ~/.hgrc

```
[auth]
bb.prefix = https://bitbucket.org/USERNAME/
bb.username = USERNAME
bb.password = PASS
```

`http://hgtip.com/tips/advanced/2009-10-01-configuring-user-auth-https/`

Authentication with the keyring

`http://mercurial.selenic.com/wiki/KeyringExtension`

Subrepos: Concepts and Pitfalls

Linked repositories: Useful but brittle

- share libraries in multiple repos
- broken subrepo, broken parent
- thin shell: Most resilient. Replaceable

path rewriting in `.hgsub`

```
SRC/remotec_core = SRC/remotec_core
SRC/sim_retrieval = SRC/sim_retrieval
SRC/sim_create = SRC/sim_create
[subpaths]
# SRC/<reponame> -> bb.org/<owner>/<reponame>
https://(.*)@bitbucket.org/(.*)/sim/SRC/(.*) = https://\1@bitbucket.org/\2/\3
ssh://hg@bitbucket.org/(.*)/sim/SRC/(.*) = ssh://hg@bitbucket.org/\1/\2
```

Summary: What you need regularly

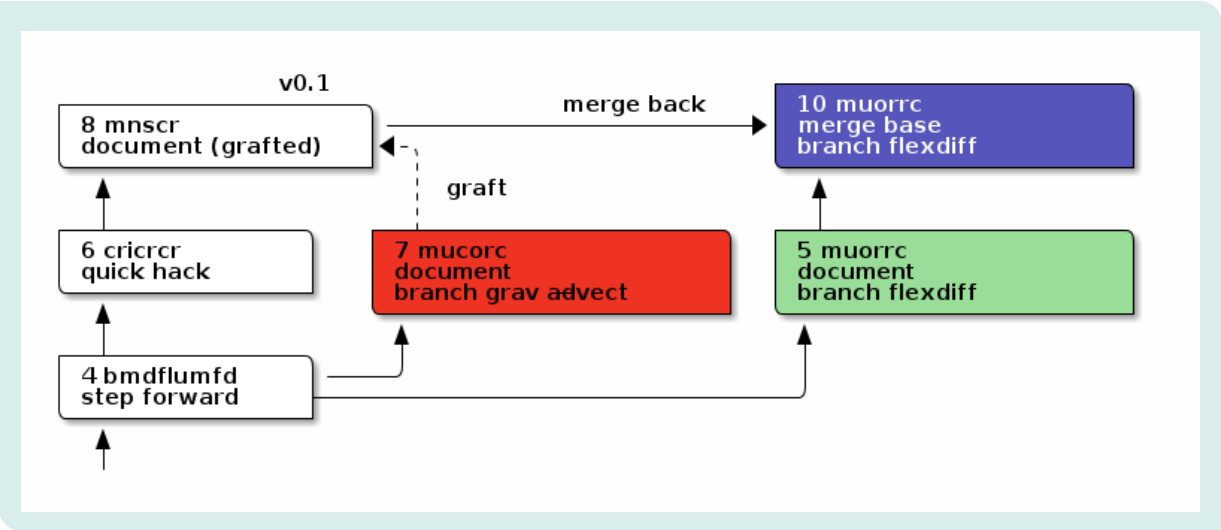
Local changes

- `init`: Create a new repository
- `diff`: See what you would commit
- `commit`: Save a new version
- `update`: Switch to another version
- `merge`: Recombine asynchronous work (do it often!)

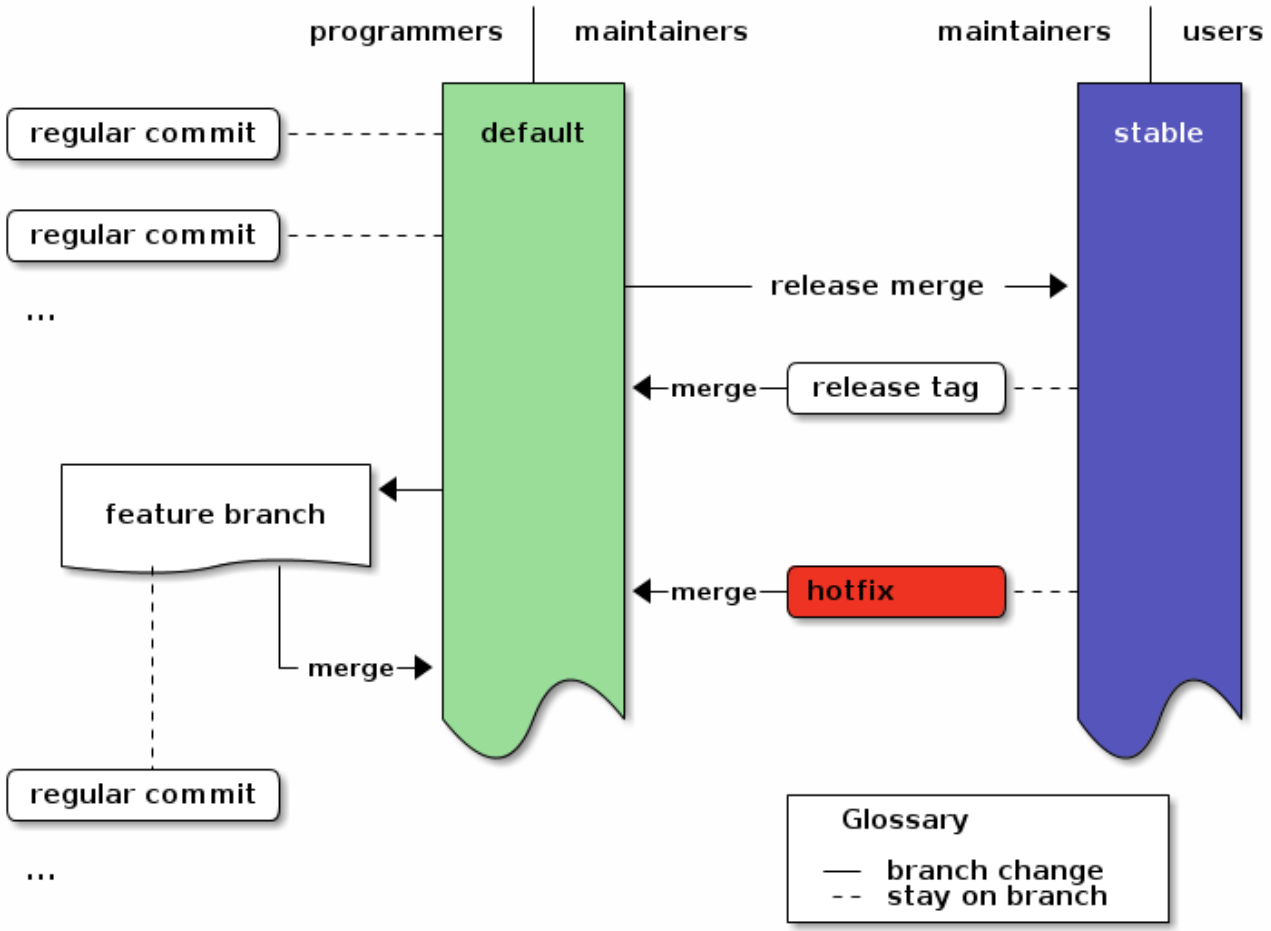
Exchange versions

- `clone`: Copy a repository
- `pull/push`: Exchange versions
- `incoming/outgoing`: See what you would pull or push

Project Branches



Advanced: Release-cycle



All development starts on default. Only maintainers touch stable.

Tasks and Commands Overview

create a project:

```
$ hg init project
$ cd project
$ (add some files)
$ hg add
$ hg commit
(enter the commit message)
```

do nonlinear development:

```
$ (do some changes)
$ hg commit
(enter the commit message)
$ hg update 0
$ (do some changes)
$ hg commit
(enter the commit message)
$ hg merge
$ (optionally hg resolve)
$ hg commit
(enter the commit message)
```

use feature clones:

```
$ cd ..
$ hg clone project grav-advect
$ cd grav-advect
$ (do some changes)
$ hg commit
(enter the commit message)
$ cd ../project
$ hg pull ../grav-advect
```

use named branches:

```
$ hg branch BRANCHNAME
$ hg update BRANCHNAME
$ hg merge OTHERBRANCH
$ hg commit --close-branch -m "finished"
```

share your repository via the integrated webserver:

```
$ hg serve &
$ cd ..
$ hg clone http://127.0.0.1:8000 project-clone
```

export changes to files:

```
$ cd project-clone
$ (do some changes)
$ hg commit
(enter the commit message)
$ hg export tip > ../changes.diff
```

import changes from patches:

```
$ cd ../project
$ hg import ../changes.diff
```

pull changes from a served repository (hg serve still runs):

```
$ cd ../grav-advect
$ hg pull http://127.0.0.1:8000
```

Use shared repositories on BitBucket:

```
$ (setup bitbucket repo)
$ hg push https://bitbucket.org/USER/REPO
(enter name and password in the prompt)
$ hg pull https://bitbucket.org/USER/REPO
```

Send patches by email:

```
$ hg help hgrc | grep "  \"smtp\"" -A 38
$ (configure in ~/.hgrc)
$ hg email --to group@example.com
```

See also

Activate extensions:

```
$ echo '[extensions]' >> ~/.hgrc
$ echo 'EXTENSION=' >> ~/.hgrc
```

Activate extension temporarily:

```
$ hg --config extensions.gpg= hg help sign
```

command abbreviations:

```
$ hg ci # commit
$ hg st # status
(any unambigous prefix)
```

create branch:

```
$ hg branch grav-advect
(do some changes)
$ hg commit
(write commit message)
```

switch to:

```
$ hg update grav-advect
```

include (merge):

```
$ hg update default
$ hg merge grav-advect
$ hg commit
(write commit message)
```

Finish (close):

```
$ hg update grav-advect
$ hg commit --close-branch -m "fixed lightpath"
```

Revsets:

```
$ hg log -r "5::8" # follow the commits
$ hg log -r "keyword(fixes)" # search
$ hg log -r "branch(stable) and \
  not merge() and \
  not keyword(signature) and \
  not keyword(tag)"
```

Tag a release:

```
$ hg tag v0.1
```

See Tags:

```
$ hg tags
```

Sign changes:

```
$ (setup gpg)
$ hg sign
$ hg sigs
$ hg sigcheck v0.1
```

Undo last action (destructive!):

```
$ hg rollback
```

Further Reading:

- Mercurial in Workflows:
  mercurial.selenic.com/guide
  15 minute basics
- hg init:
  hginit.com
  simple guide for windows
- Using Extensions:
  mercurial.selenic.com/wiki/UsingExtensions
- Branching Strategy:
  draketo.de/branching-strategy
- software carpentry

Thank you!