Introduction to Version Control

Ralf Hemmecke

Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria

25-Oct-2012

Outline

Outline

Old Style

- Files are in a directory
- lots of backup files
- ordered by manual version number or manual date or ???

```
/home/hemmecke/myproject/
                  myproject/myproject-1.0/
                  myproject/myproject-1.2/
                  myproject/myproject-20061014.tar.gz
                  myproject/myproject-20061117.tar.gz
                  myproject/myproject-20061122.tar.gz
```

- generated files like .dvi, .ps etc. are also stored
- HD space is cheap, but manual administration costs time

New Style

Source Code Management systems

Use an SCM system to store versioned files and history in a repository.

Backup vs. Version Control

- backup = save a snapshot
  - automatically (daily, monthly, etc.)
  - manually (at release of a program, article, book)
- version control is not versioning of software
  - several commit between two releases of a program
- version control = save snapshots with meaning
  - task/idea driven
  - more fine grained than backups
  - can reconstruct development history of the product

Common use cases for SCM systems

Source Code Management can be beneficial for

- single user
  - keep history and evolution of files
  - doing work on different machines
  - develop a program with several releases
- multiple user
  - writing a joint article with other authors
  - develop a program in a group

Free Source Code Management Systems

- for central development
  - RCS
  - CVS
  - Subversion (SVN)
- for distributed development
  - SVK (uses SVN as backend)
  - GNU Arch, Bazaar-NG
  - Git (used for Linux kernel)
  - Mercurial
  - Darcs

Outline

Repository

A Repository can be considered as a collection of snapshots of a directory in the file system together with dates and log information.

Creating a repository in a directory

The following creates a subdirectory `.git` with all the relevant information.

```
cd SOMEDIRECTORY
git init
git add . # ((Note the dot.))
git commit -m 'initial version'
```

Adding new files to a repository

```
git add FILEA FILEB
git commit
```

Updating files in a Repository

```
# edit files FILEA and FILEB
git add FILEA FILEB
git commit
```

or simply

```
# edit files FILEA and FILEB
git commit -a
```

to commit all modified files

Removing files from a repository

```
git rm FILEB
git commit
```

Since git records the history, the file can still be recovered in any previous version.

Getting Information

Help

git help

History

git log

What is the Current Situation

git status

Graphically investigate the history

gitk -a

Simplified Storage Picture

- content is addressed by its hash value (sha-1, a 40 digits hex number)
- commits are stored as a directed acyclic graph (DAG)
- initial commit is the root DAG
- leaves of the DAG are called branches
- creating a new leaf is called branching
- there is usually a master branch
- HEAD points to the current branch
- joining two commits is called a merge

The real Start

Make yourself known to git

```
git config -global user.name 'John Doe'
git config -global user.email doe@example.com
```

Creation of a Git Repository

- Create a directory and a file.

```
cd MagicRings
emacs magicrings.tex ForgingOfTheRings.tex
emacs MagicOverMagic.tex Introduction.tex
```

- Put this data under version control.

```
git init
git add .
git commit -m 'initial commit'
```

- Check that everything is fine.

```
git log
git show
gitk
```

Simple Workflow

- Create a Makefile

```
emacs Makefile
git status
git diff
git add Makefile
git status
git diff --cached
git commit -m 'add automatic compilation'
```

- Set *log message editor* and add Forging section

```
GIT_EDITOR='emacs -nw'
emacs magicrings.tex
git status
git diff
gitk
git rm ForgingOfTheRings.tex
git commit -a
git log --stat
```

Simple Workflow 2

- Add MagicOverMagic section

```
emacs magicrings.tex
git commit -a
git log --stat
```

- Oops. Commits should be logically connected. `MagicOverMagic.tex` should have been deleted in the previous commit, i.e. undo last commit and do it right.

```
git rm MagicOverMagic.tex
git commit --amend -C HEAD
```

Simple Workflow (checking history)

- What was the situation 2 commits ago?

```
git checkout HEAD~2
gitk
```

- Go back to the master branch.

```
git checkout master
gitk
```

Go on travel and don't stop with your work

- Clone the repository to your laptop via ssh.

```
git clone hemmecke@speedy.risc.jku.at:MagicRings
```

- or (equivialently)

```
U=hemmecke; H=speedy.risc.jku.at
D=/home/hemmecke/MagicRings
git clone ssh://$U@$H$D
```

- or put the lines

```
Host rh-risc
  IdentityFile ~/.ssh/id_rsa
  HostName speedy.risc.jku.at
  User hemmecke
```

into `~/.ssh/config` and clone via

```
git clone rh-risc:MagicRings
```

Work on your laptop

- Add introduction during travel

```
cd MagicRings
emacs magicrings.tex
git rm Introduction.tex
git commit -a -m 'add introduction'
```

Back at the office

- On your laptop you could say

  ```
  git push origin master
  ```

  but you have to take care if the “origin” was not a bare repository.
- Better go to your office computer pull in the changes made on the laptop

  ```
  git pull mylaptop.risc.jku.at:MagicRings
  ```

  or (equivalently)

  ```
  git remote add origin mylaptop.risc.jku.at:MagicRings
  git pull origin master
  ```

Use Git as a backup machine

- Create a bare repository on a USB stick or an external hard drive.

```
cd ~/USB
mkdir MagicRings.git
cd MagicRings.git
git init --bare
```

- push your work to the backup repository

```
cd ~/MagicRings
git remote -v
git remote add backup ~/USB/MagicRings.git
git push backup master
```

Use Git as a backup machine 2

- Make more changes.

```
cd ~/MagicRings
emacs magicrings.tex
git commit -a -m 'fix typo'
```

- physically connect your backup storage and push again

```
git push backup master
```

- The whole history that leads to *master* is now in a safe place.
- You can now throw away all you working copies and are still able to reconstruct all files including their history from `MagicRings.git`.

Git branching an merging

- Create a new branch.

```
git branch mybranch
```

- Add new commit on “master” branch.

```
emacs magicrings.tex
git commit -am 'weird magic'
```

- Switch to “mybranch”.

```
git checkout mybranch
emacs magicrings.tex
git commit -am 'new feature'
```

- Merge “mybranch” back into master.

```
git checkout master
git merge mybranch
```

Other important commands

```
git gc
git whatchanged
git branch
git tag
git merge
git fetch
git reset
git blame
git gui blame
git stash
```

Git Documentation

- Git books

  `http://book.git-scm.com/`
  `http://www-cs-students.stanford.edu/~blynn/gitmagic/`

- Git can read and write SVN (Subversion) repositories

  `google: git svn`

- git cheat sheet

  `http://cheat.errtheblog.com/s/git`

Git Hosting

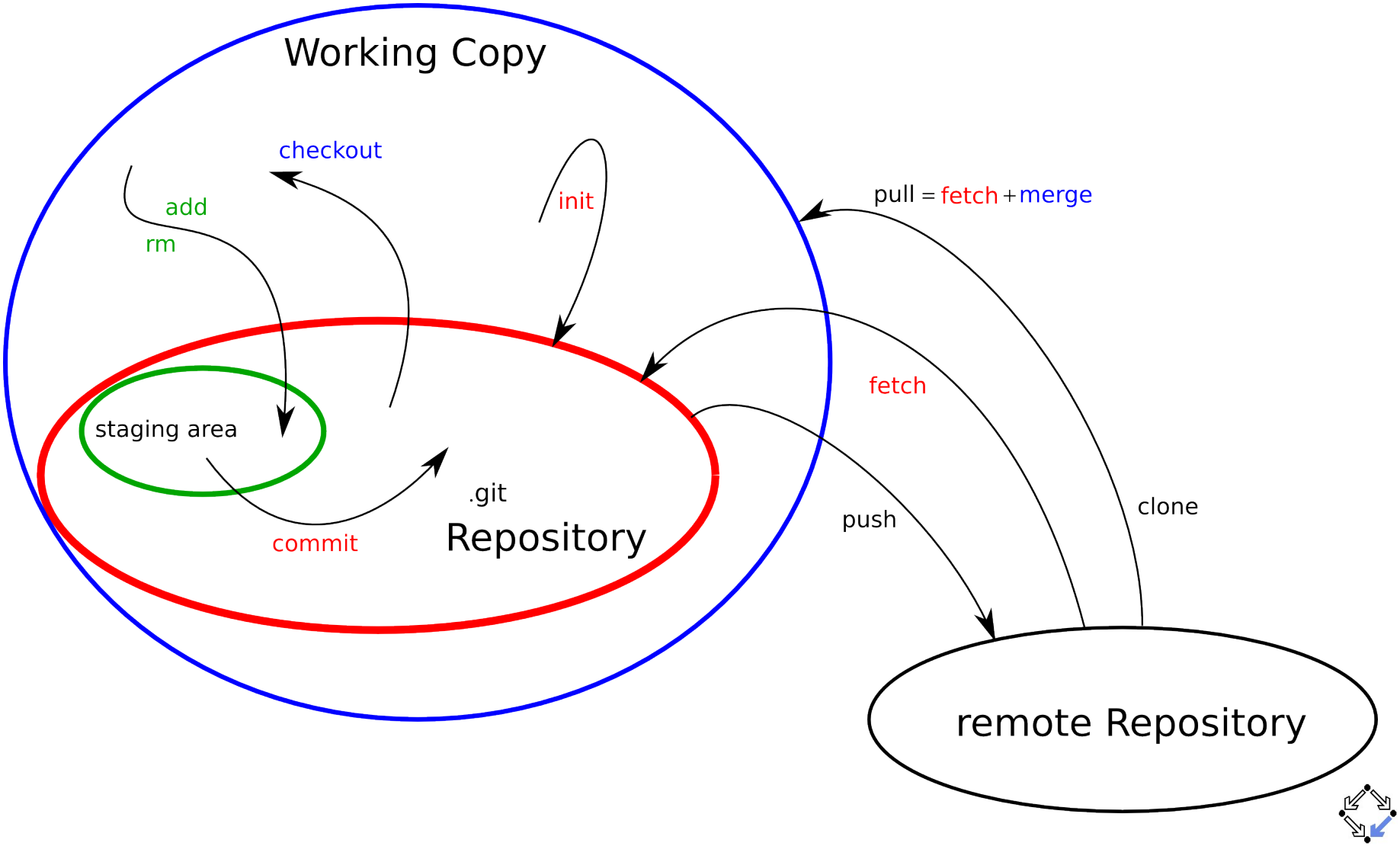
- Public git hosting

  `https://git.wiki.kernel.org/index.php/GitHosting`
  `http://github.com/`
  `http://repo.or.cz/`

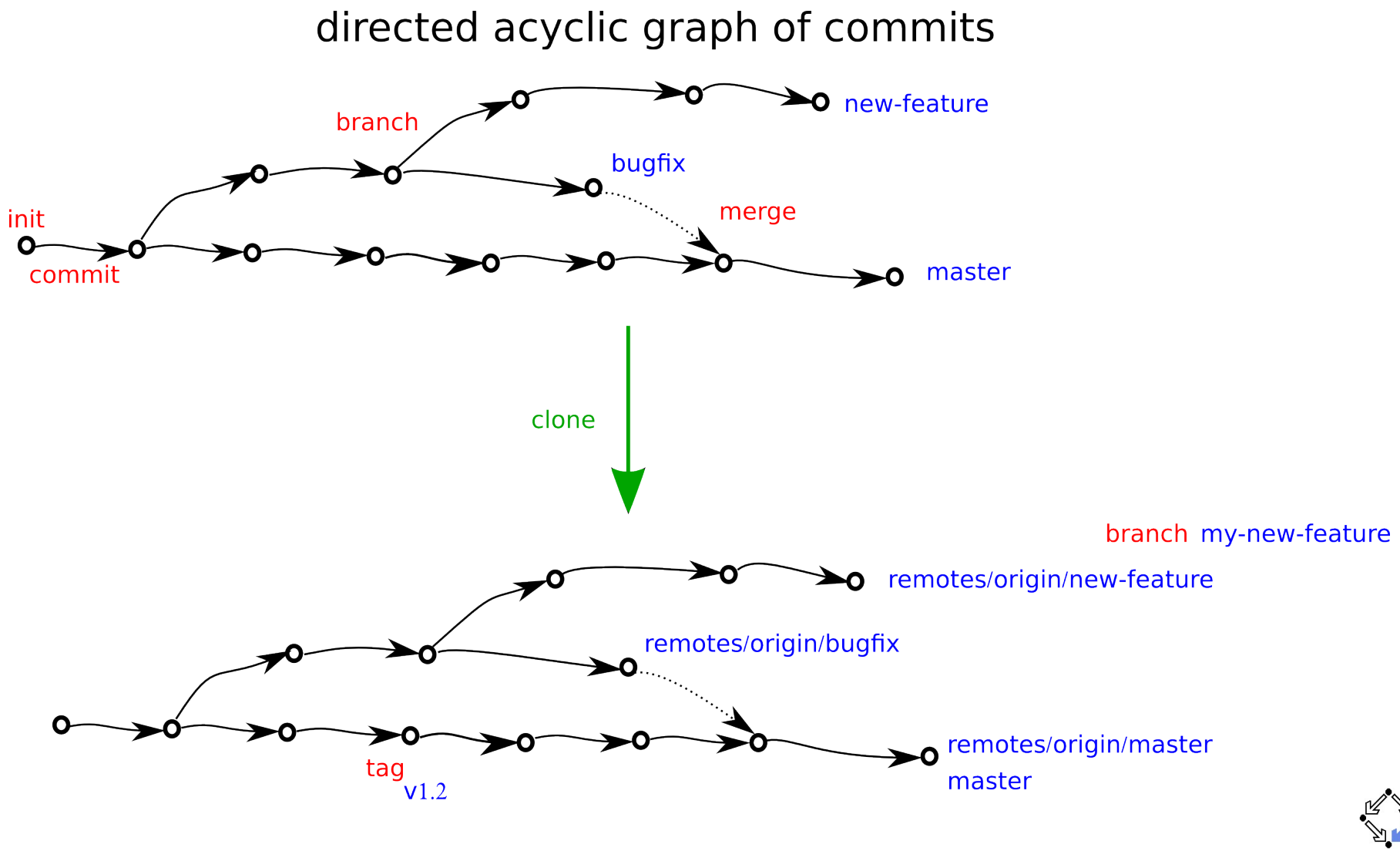
- Privately shareing git repositories

  `https://portal.risc.jku.at/search?SearchableText=gitolite`

Git Commands



Git DAG


directed acyclic graph of commits
new-feature
branch
bugfix
merge
init
commit
master
clone
branch my-new-feature
remotes/origin/new-feature
remotes/origin/bugfix
remotes/origin/master
master
tag
v1.2