Talk Overview

Motivation  
git History  
git Usage  
git Implementation  
Documentation + further reading  
Creating a Repository  
Basics: Configuration  
Latest Status  
Adding Files  
Clone a Repository  
Viewing Changelog  
Exploring History  
Managing Branches  
Exploring History II  
Cleanup  
Cooperation  
Mail Patches  
Shuffle Patches
Motivation

- non-linear development
- distributed development
- striking community
- scalability + speed
- cryptographic authentication of history
- protocol support: git, ssh, rsync, http, ftp
git History

three parents: BK, cache, Monotone

timeline:
April 2005: officially used to track Linux
June 2005: Linux 2.6.12 release managed by git
December 2005: git 1.0 release
February 2007: git 1.5 release
September 2007: git 1.5.3 release

“And then realize that nothing is perfect.
Git is just *closer* to perfect than any other SCM out there.” -linus
git Usage

- linux-2.6
- xorg
- wine
- olpc
- alioth
- ...
git Implementation

data structure: index + blob object

**index**: the mutable index caches information about the working directory and the next revision to be committed

**blob**: append-only blob object of each file-revision. Each object is identified by a SHA-1 hash of its content. A blob object is combined into packs to save space using delta compression.
Documentation + further reading

git user manual

git tutorial

git wiki

everyday git with 20 commands or so

git help

git Wikipedia
Creating a Repository

Either extract your favourite project

```
$ tar xzf project.tar.gz
$ cd project
$ git init
Initialized empty Git repository in .git
```

or start with a blank dir

```
$ mkdir ~/src/project
$ cd ~/src/project
$ git init
Initialized empty Git repository in .git
```
Creating a Repository II

Tell git to track any file below the current directory

$ git add .

Finally commit the current state with interactive commit message

$ git commit -a

maybe directly with commit message

$ git commit -a -m "Initial commit of <project>"
**Basics: Configuration**

- **Global configuration**: `.gitconfig`
  - `$ git config --global user.name "maximilian attems"`
  - `$ git config --global user.email maks@debian.org`

- **Set email per repository**
  - `$ git config user.email maks@debian.org`

- **List config variables**
  - `$ git config --list`
check if anything changed

```bash
$ git status
```

really no diff!?

```bash
$ git diff
```

hmm ok we are lazy, show latest commit

```bash
$ git show
```
Adding Files

create file

```
$ cat > hello.c << EOF
#include <stdio.h>

int main()
{
    printf("hello world\n");
    return 0;
}
EOF
```

tell git which files you want to commit

```
$ git add test1.c
$ git commit test1.c
```
Clone a Repository

on the example of initramfs-tools

```
$ git clone \
  git://git.debian.org/git/kernel/initramfs-tools.git
```

developer access for pushing

```
$ git clone \
  git+ssh://maks@git.debian.org/git/kernel/initramfs-tools.git
```
At any point you can checkout the history of changes

```bash
$ git log
```

Show the diff of each step too

```bash
$ git log -p
```

summary overview

```bash
$ git log --stat --summary
```
Exploring History

see the great grandparent of HEAD

$ git show HEAD~4

see a specific commit

$ git show <object>
$ git show b71721f02b6b46fddfc624888f61aafbc2399129

use short notation of the object

$ git show b71721f02b6

(first few chars are enough to uniquely define that commit)
Managing Branches

list current available branches
(the asterisk marks the branch you are on):

$ git branch

switch to specific branch

$ git checkout <branchname>
$ git checkout david

merge specific branch

$ git merge <branchname>

create new branch from current state on

$ git checkout -b <branchname>
see log between two versions

```bash
$ git log 0.86..0.87
```

list commit that where made on the david branch but not on maks branch

```bash
$ git log maks..david
```

see what changed in the boot scripts the last six weeks

```bash
$ git log --since="6 weeks ago" scripts
```

crazy search

```bash
$ git log --since="June 5, 2005" \ 
    --grep="regex" --author="guy@domain.org" \ 
    --pretty=oneline some-branch -- path
```
reset the last commit

$ git reset
$ git reset HEAD~1

reset the last commit and discard any change

$ git reset --hard

revert changes to a file

$ git checkout file
Cooperation

define remote repository

$ git remote add bob /home/bob/myrepo

fetching remote repository

$ git fetch bob

after some checks ;) merging this repository into master

$ git merge bob/master
Mail Patches

prepare patches for e-mail submission, creates patches in $PWD

```
$ git format-patch -M -s master
```

inspect them, vi 00*

fire them off

```
$ git send-email --to upstream-maint@project.org
```
Shuffle Patches

interactive patch shuffeling: allows to merge, pick and reorder commits

```bash
$ git rebase -i HEAD~10
```