redhat.

03
Gitshop
Git - repositories

Jiří Jabůrek (jjaburek)
Red Hat 2012

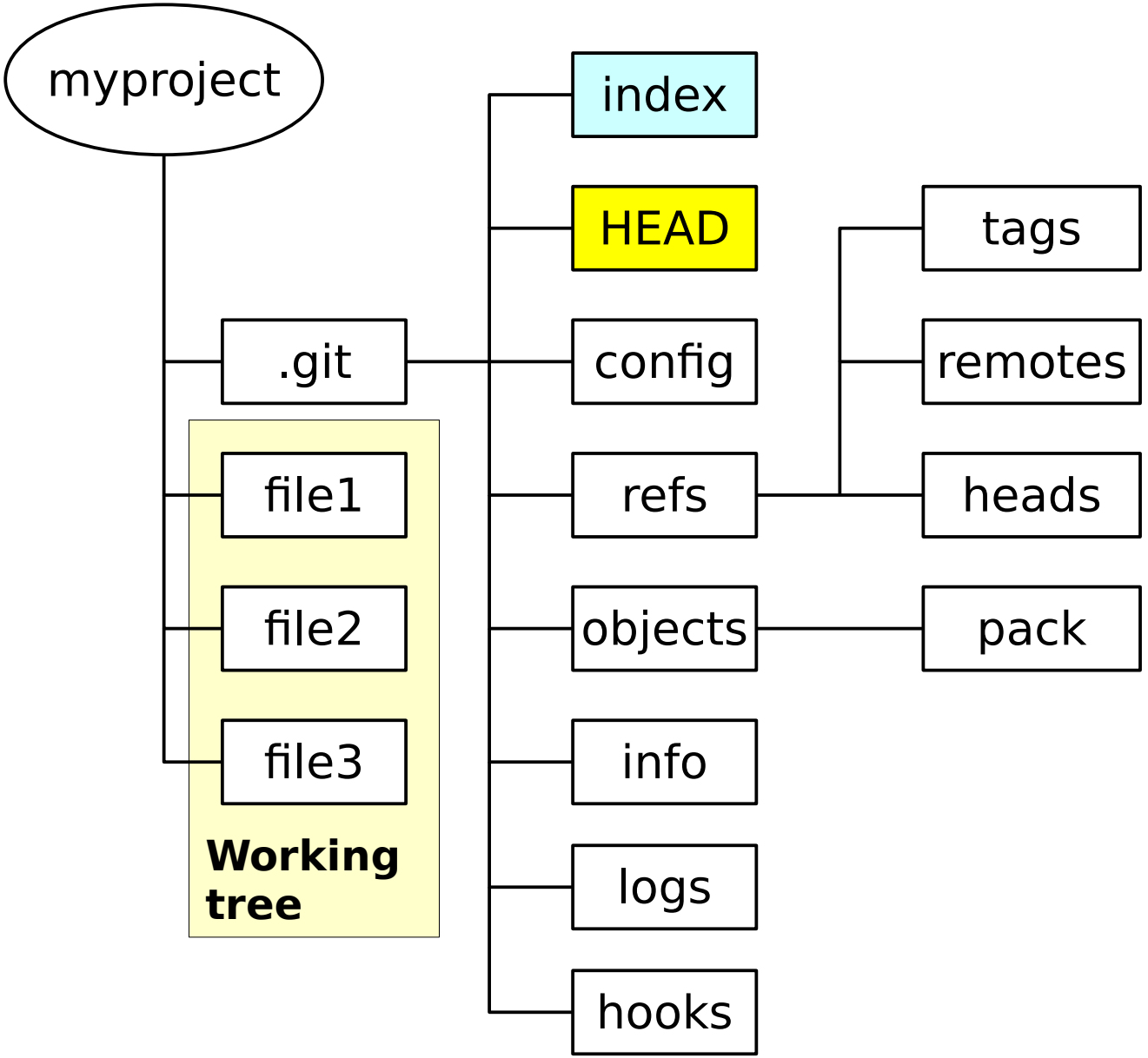
Git repositories

- `man 7 gitglossary`

```
repository
  A collection of refs together with an object database
  containing all objects which are reachable from the refs,
  possibly accompanied by meta data from one or more
  porcelains. A repository can share an object database
  with other repositories via alternates mechanism.
```

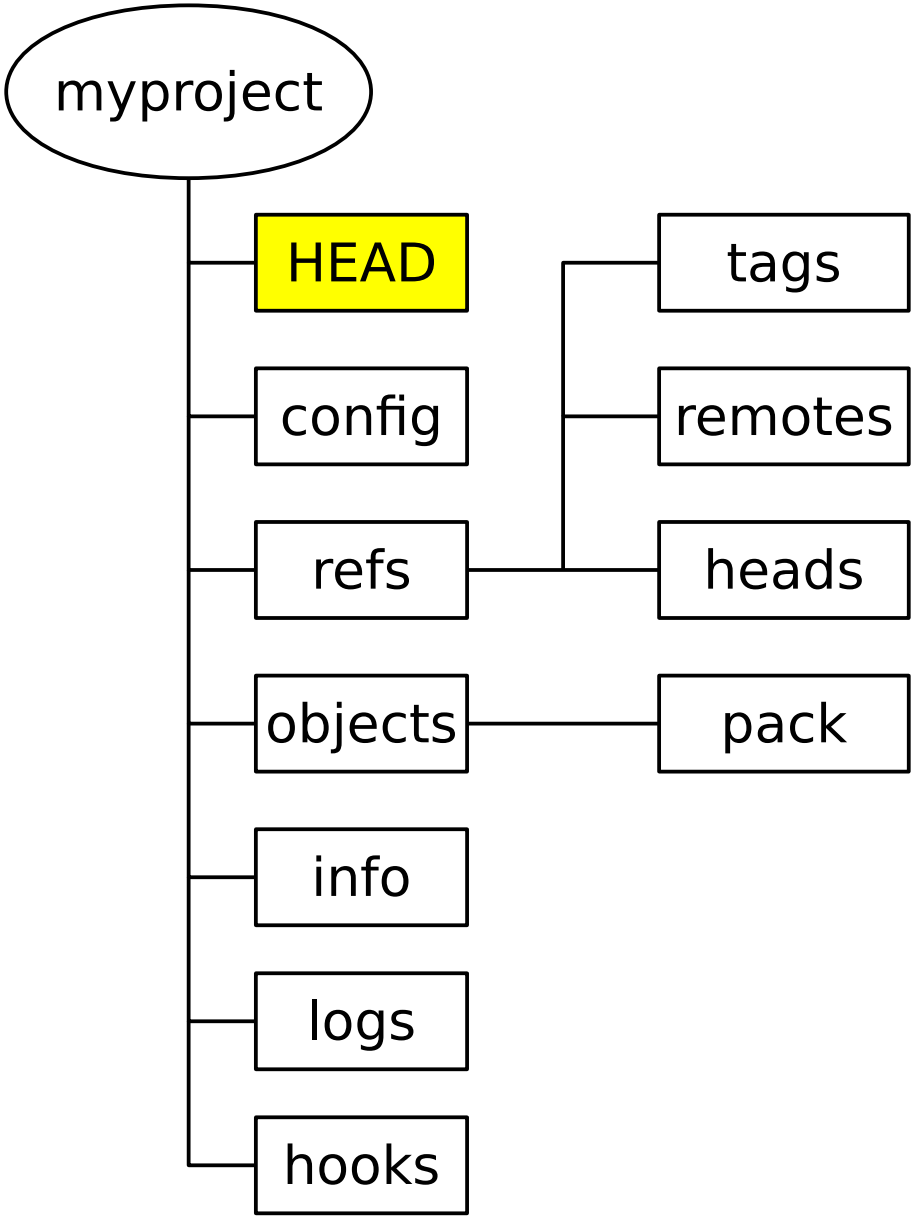
Repositoy types, structure

Normal repository



Bare repository

```
git init --bare
```



HEAD in bare ???

Basic repository structure

```
COMMIT_EDITMSG    hooks/
FETCH_HEAD        index
HEAD              info/
ORIG_HEAD         logs/
branches/         objects/
config            packed-refs
description       refs/
```

```
refs/
refs/heads
refs/heads/branch1
refs/heads/branch2
refs/remotes
refs/remotes/origin
refs/remotes/origin/hisbranch1
refs/tags
refs/tags/sometag
```

```
objects/
objects/info
objects/info/packs
objects/eb
objects/eb/56dc85d2336d927252909f21949d2734041889
objects/94
objects/94/a14c0f90ffcaed7a583aef409ea8a33093011d
objects/94/c2d5c38364c817d79759c2a35ff1fb1693440f
objects/f9
objects/f9/350dab48335cbcc81d9a3ea46c3d121dcf1808
objects/pack
objects/pack/pack-e7e74fef42f3583516bf887e0ddd559224fbb665.pack
objects/pack/pack-e7e74fef42f3583516bf887e0ddd559224fbb665.idx
```

Distributed

Distributed (nature)

- Everything done locally, always
- Inter-repo communication explicitly requested
- Ad-hoc, no permanent daemon needed
  - except git-daemon
- Network of trust
- Backups everywhere

Backends, protocols, URL types

Protocols

- git
  - smart - transfers only missing objects
  - uses deltas whenever possible
  - creates pack of objects to be transfered on the transmitting side, transfers the pack to the receiver
  - works only with referencable objects
- "dumb" http
  - work over restrictive firewalls
  - has no extra requirements on the server
  - supports push as well, via DAV

Protocols using "git" as backend

- ssh
  - authentication via password/keys/kerberos/..
  - push
    - local send-pack spawns receive-pack on remote side
  - fetch
    - local fetch-pack spawns upload-pack on remote side
- "smart" http
  - works over restrictive firewalls as well
  - requires server-side CGI script; git-http-backend
    - can serve both "smart" and "dumb" http
  - able to push (send-pack) and fetch (fetch-pack)

URL types

- git

```
git://host.xz[:port]/path/to/repo.git/
```

- ssh

```
    ssh://[user@]host.xz[:port]/path/to/repo.git/
git+ssh://[user@]host.xz[:port]/path/to/repo.git/
ssh+git://[user@]host.xz[:port]/path/to/repo.git/
[user@]host.xz:path/to/repo.git/
```

- http, ftp, rsync, ...

```
http[s]://host.xz[:port]/path/to/repo.git/
ftp[s]://host.xz[:port]/path/to/repo.git/
rsync://host.xz/path/to/repo.git/
```

- local

```
       /path/to/repo.git/
file:///path/to/repo.git/
```

URL types (less known)

- git-remote-* backends
  - http, https, ftp, ftps, rsync, testgit, ext, fd
  - git-remote-ext
    - `ext::<command>[ <arguments>...]`

```
git clone "ext::socat stdio unix-connect:sockfile" myrepo
```

  - git-remote-fd
    - `fd::<infd>[,<outfd>][/<anything>]`

```
git clone "fd::42/path/to/repo" myrepo
```

(porcelain)
Commands

Commands

- git-push
- git-fetch
- (git-pull) = fetch + merge
- (git-clone) = out of context

Commands

- git-push
  - "Update remote refs along with associated objects"
  - many uses, see manpage
  - usage: `git push <URL> <refspec>...`
    - `git push <URL> src:dst    # src can be empty`

```
# push refs/heads/master to remote refs/heads/master
git push <URL> master

# push refs/heads/master to remote refs/heads/boom
git push <URL> master:boom

# push commit sha 1234567 to new branch "newbr" on remote
git push <URL> 12345678:refs/heads/newbr

# push commit before head to a new tag "newtag" on remote
git push <URL> HEAD~1:refs/tags/newtag
```

Commands

- git-fetch
  - "Download objects and refs from another repository"
  - sets the FETCH_HEAD reference
  - can fetch only references
  - usage: `git fetch <URL> <refspec>...`

```
# fetch branch "br1" from repository at <URL1>
git fetch URL1 br1

# see commits on it
gitk FETCH_HEAD

# create a local branch on the same commit "br1" is on
git branch mybranch FETCH_HEAD
```

Remotes

Remotes

- Provide a way to manage remote repositories
- Create references for remote refs in local repo
  - keep fetched objects referenced
- Fetched objects and tag refs become local
- Stored locally, synced only on request
  - git-fetch , git-remote <name> prune, ...
  - .git/config

```
[remote "origin"]
	fetch = +refs/heads/*:refs/remotes/origin/*
	url = git://host.dom/path/to/repo.git/
```

  - .git/refs/*

```
remotes/origin/master
remotes/origin/anotherbranch
```

Remotes

- Examples

```
# add new remote "myremote"
git remote add myremote git://some.url/path/to/repo.git

# fetch all branches (and related tags) from the remote
git fetch myremote

# inspect "master" remote branch
gitk myremote/master

# create local branch based on myremote/devel + checkout it
git branch coolfeature myremote/devel
git checkout coolfeature

# make some changes, commits

# push commits made to myremote/devel
git push myremote coolfeature:devel
```

Tracking branches

- A way to link local branch to a remote one
- No hard dependencies, only hints
  - Hints based on locally-available data, no auto fetch
- Usage

```
git branch --track mine origin/theirs
```

- Storage - .git/config

```
[branch "mine"]
        remote = origin
        merge = refs/heads/theirs
```

- Example hints - on checkout

```
Your branch is ahead of 'origin/theirs' by 1 commit.
Your branch is behind 'origin/theirs' by 12 commits, and can
be fast-forwarded.
```

Additional points

Additional points

- git-clone ?
  - git init
  - git remote add origin <PROVIDED_URL>
  - git fetch origin
  - BR = (looks up HEAD on remote)
  - git branch --track $BR origin/$BR
  - git checkout $BR
- Force push?
  - git push -f
  - git push ... +master:master

Presentation: Questions?

Workshop >>

use workshop-03 repository/branch for this workshop

Workshop

goal: **selectively publish changes,**
using git-push,
creating branches/tags on remote

1) **See changes**
   - git log --oneline
   - gitk
2) **Create a bare repository outside** `git init --bare`
3) **Add it as a remote (optional)**
4) **Using git-push:**
   - create tags (v1.0, v1.1, v1.2) for each commit marked with "!!!" in the remote repository
   - create "devel" branch from the commit marked v1.2 on the remote
   - BONUS: perform both actions without moving HEAD or creating branches/tags in the local repo (ie. using `src:dst`)
5) **Go to the bare repository and set HEAD to point to "devel"**
   - using `git symbolic-ref HEAD refs/heads/devel`
6) **Go outside and clone the bare repo locally**
7) **See the changes in the cloned repo**
   - important commits should be tagged
   - local branch named "devel", tracking remote "origin/devel"

The End

Thanks for listening

Links

- Git-scm.com - "transfer protocols"

```
http://git-scm.com/book/en/Git-Internals-Transfer-Protocols
```