Git in the enterprise

Charles Bailey

25th April 2012

External hosting service

- Less configuration and management
- Reliance on your hosting provider
- Security of your intellectual property

Pre-packaged add-ons

- Many to choose from
- Make sure that your choice matches your way of working, or can me made to.
- Possibilities include:
  - gitolite
  - gitosis

Just vanilla Git

Total control... but everything extra that you want must be scripted yourself.
Consider development and maintenance costs.

Hosting it yourself

Vanilla `git-web` will only get you so far

Authentication through transport

- Git has no native authentication
- `ssh` is the transport of choice for strong authentication
- Only use `git` transport in read-only mode

Trust and keys

- Using passwords with Git+ssh is laborious
- Public key authentication is easier
- The ability to revoke keys is essential

Account configuration

- Universal git account with multiple keys
- Individual accounts

"Commit" access

Historically in Open Source, commit access was for the privileged. In the commercial environment, direct access to save to the repository is a requirement to "get the job done".

ssh access

- Direct write access means no controls
- `git update-ref -d refs/tags/v1.0` is dangerous
- ... but then so is `rm -rf repo.git`

Safer options

Levels of access

1. Direct `ssh` access to central server
2. Commands restricted by ssh key restrictions
3. Restricted `git-shell` access

Finer grain

- File system permissions can give per-repository read and write access
- Per-branch control is usually achieved through the `update` hook E.g. `contrib/hooks/update-paranoid`

Recommended configuration

```
git config receive.denyDeletes true
```

Auditing

Pusher vs author vs committer

- Enforce known "author" — but anyone can spoof another's commit and this disallows pushing others' commits in rebases and new branches
- Enforce known "committer" — still easy to forge
- Enforce "committer = pusher" (e.g. `update-paranoid`) — but now I can only push another's commit if I amend it
- Audit pushes

Trusting the content

- Signed tags
- Signed commits

Signed commits

- Requires Git >= 1.7.9
- Signature is stripped on *any* modification of the commit
- A signed commit guarantees the signer (not the committer or author)

Signed commit demonstration

Yes, Git supports all your workflows

- Git gives you great freedom.
- You may have to script your own enforcement.

Free-for-all

- Don't expect the ultimate in cleanliness
- Education is important
- Beware the "cheat sheet"

Locked down

- Slows down your development rate
- *May* increase quality
- *May* make for a cleaner history
- *May* backfire

Forms of lock-down

- "Golden" repository has restricted access
- Release branches have restricted access
- Commits need appropriate "signature"

Large objects

- Use Git for source
- Avoid placing binaries and large objects in your source tree

Pushes per day

Total push frequency is a constraint

1. Team size
2. Per developer push frequency
3. Test cycle length

Partitioning the world into repositories

- Git's "whole tree" versioning makes partitioning important.
- You can't get away with a "universe" repository.
- Put the source for each unit of release in to a single repository.

Submodules

To reflect a submodule change, a commit *must* be made in the parent repository.
This means that submodules don't solve the "contended `HEAD`" problem.

Reviews

- `Signed-off-by:` can mean what you need it to mean
- Review commits before pushing
- Allow reworks to be squashed in
- Decide whether to review merges

Questions

Questions?