

Gitolite

Access Control for Git Servers

Sitaram Chamarty

TCS Innovation Labs
Hyderabad

Nov 4-5 / FUDCon Pune 2011

Git – Main Features Recap

- Distributed
- Very powerful
- Very scalable, and very fast
- Attracts large teams working on large projects

Access Control: Developers can...

- ...push arbitrary branches to the server
- ...rewind or delete branches on the server
- ...push arbitrary files to the server

Access Control: File System ACLs don't help

- a "branch" is not necessarily a separate (set of) file(s)
- even if it were, you now need Unix userids for everyone

Access Control: 'git-shell' doesn't cut it

- useful to prevent *non* git commands from being run
- but *within* the repo, you can do anything

Authentication and Authorisation

- Authentication
 - ensuring that you are who you claim to be
- Authorisation
 - once we know who you are, deciding if you're allowed to do what you want to do

Authentication is hard

- In general, Authentication is harder to do safely
- Almost always requires serious stuff like crypto
- And you **don't** want to write and maintain that, trust me!
- Best thing to do is let someone else deal with it
 - sshd
 - httpd/apache
 - ???
- (They've been at it longer and know far more than you or I do!)

Gitolite is...

- ...an authorisation tool
- (**not** an authentication tool!)

Feature Goals

- access control down to the branch level
- including rewind (force-push) control

Implementation Goals

- should not require root
- should be installable on any Unix-like OS
 - as long as git is already installed
- should have a much better config language than ...

(later goals)

- all access *and* ACL changes should be auditable and reviewable
- ACL management should not need shell access

Main Features

- uses a single "real" user on the server
- allows many "virtual" users
 - `sshd authorized_keys` file size is the only limit
- control access to many repos
 - "read" control at "repo" level
 - "write" control at "branch/tag" level
 - including create, push, rewind, and delete
- lots – and I mean **lots** – of documentation

Advanced Features for the Admin

- simple, but very powerful, config language
 - scale to thousands of repos and users
 - allow grouping of users and repos for convenience
- can also control who can change what files/dirs
- delegate parts of the config to sub-admins

Advanced Features for the User

- Personal branches
- Personal repos
- Custom commands (defined by the admin, run by the user)

Technical Support

- ssh
- confusion between git and gitolite
- Windows and Mac
- UGFWIINI
- oddball situations

Documentation

- ... can fill a whole day ...
- ... best to skip it! ...

Fedora

- Scaling up
 - more than 11,000 repos
 - more than 1000 users
 - the current version of gitolite (at that time) would grab all RAM and die!
 - sometimes it would take the machine with it :-)
 - redesign the internals a little more efficiently
 - ...and "big config" mode was born!

KDE

- fork command, plus lots of others (help, watch, delete-branch, who-pushed)
- custom permissions categories
- hub – allowing merge requests between users
- self-service key management

- Dan Carpenter's security audit
- "deny" rules for the entire repo
- make "playing with gitolite" easier

Academia

- Prof Hiren Patel, U Waterloo
 - "wildcard" repositories
 - the whole concept comes from his use-case/problem
 - the SACMAT paper
 - (my first conference paper, yaaaay!)

At TCS

- For a large product group
 - many cities, many products/repos, many developers
- Advanced mirroring capabilities
 - the "NUMA" thing
 - almost "active-active" mirroring
 - partial mirrors and local repos
 - laggy mirrors
 - autonomous mirrors