gnunet-git – GNUNet support for git

Rogerio Carvalho
Matthias Kruk

June 17, 2014
Contents

1 Motivation
2 Related work
3 A git primer
4 Design overview
5 An illustrative example
Motivation

1 Centralised code storage is bad
  - Prone to governmental interference
  - Anonymity is hard to achieve
  - Hard to track interesting modifications
  - We don’t like GitHub

2 Consider this:
  - You’re anonymously developing a harddrive encryption tool
  - The NSA has deanonymised you
  - You’re being national-security’d to include a backdoor
Related work

1. **Git (duh!)**

2. **Gittorrent**
   - In a nutshell: BitTorrent transport for git
   - No working implementation
   - Project was abandoned in 2009
   - → Not too useful for us
A git primer

Git is a decentralised version control system, originally developed by Linus Torvalds to facilitate Linux kernel development.

- Doesn’t force you towards centralisation or decentralisation
- Very lightweight, tuned towards high performance
- Committing, branching, merging, etc. similar to SVN
- Any clone of a repository includes the full revision history, everything
- Decentralised structure mostly affects the workflow
A git primer - Centralised SCM

Diagram showing a central VCS server with version databases for versions 1, 2, and 3. Two computers, A and B, are shown with files being checked out from the server.
git – The git commandline client

- Communicates with local and remote repositories
  - Supports http, ssh, git, rsync, file transports
  - Pushing only with ssh, rsync transports
- Used for any operations
  - Committing changes
  - Branching, merging
  - Pulling, pushing commits
A git primer

**git** – The git commandline client

Git commands you’ll need the most:

- clone – fetch a repository, similar to svn co
- pull – update a previously cloned repository, like svn co
- add – mark file for inclusion in the next commit
- commit – save a set of changes to the repository
- branch – create and delete branches
- merge – merge branches
git-daemon – The git server

- Listens on port 9418 by default
- Services requests using the git protocol
  - For example $ git clone git://example.org/project
- Exports all or marked repositories in its base path
  - Something like /var/lib/git/ (on Debian)
Design overview – Some terminology

- **Local repository**: A repository that is hosted by the local git-daemon
- **Remote repository**: A repository that is hosted by the git-daemon on another machine
- **Local request**: A request for a local or remote repository that originates on the local machine
- **Remote request**: A request for a local or remote repository that originates on another machine
Design overview

The entire thing consists of two parts:
- gnunet-git – The commandline client
- gnunet-git-daemon – The backend, GNUnet service

Furthermore, *git* and *git-daemon* are used.
**gnunet-git** acts as a wrapper around **git**

**gnunet-git-daemon** listens on localhost:9418, GNUnet
- Host lookup using GNS
- CADET is used to communicate via GNUnet
  → Authentication and confidentiality

**git-daemon** listens on localhost:9419
**gnunet-git** – The commandline client

- It’s (for most parts) a wrapper around *git*
- Looks at the destination URL given by the user
- Translates it to a URL that is understood by *git* and *gnunet-git-daemon*
- Used to query meta information from *gnunet-git-daemon*
Design overview – *gnunet-git-daemon*

*gnunet-git-daemon* – The glue between GNUnet, sockets, and git

- Listens on localhost:9418 for local requests
- Listens on a GNUnet port for remote requests
- Connects to the local *git-daemon*
- Forwards remote requests from GNUnet to the local *git-daemon*
- Forwards local requests for remote repositories to the respective GNUnet peer
- Synchronises repository statuses with familiar *gnunet-git-daemons*
Design overview

- `git`
  - Invokes `gnunet-git`
  - Local operations (commit, branch, ...)
  - Clone, push, pull, ...
  - Load, store

- `gnunet-git`
  - Discover remote operations

- `gnunet-git-daemon`
  - Clone, push, pull, ...

- `/var/lib/git/`
  - foo
  - bar

GNUnet
Example: cloning a remote repository from GNUnet

- **Alice** invokes gnunet-git:
  
  ```bash
  $ gnunet-git clone gngit://bob.gnu/project
  ```

- gnunet-git invokes git:
  
  ```bash
  $ git clone git://localhost/bob.gnu/project
  ```

- gnunet-git-daemon connects to the gnunet-git-daemon at **bob.gnu**

- The gnunet-git-daemon at **bob.gnu** relays the request to localhost:9419, its local **git-daemon**

- **Alice**’s gnunet-git-daemon clones the repository into the base path of the local **git-daemon**
Bob’s gnunet-git-daemon remembers that Alice cloned

Alice automatically shares her clone of the repository

- Bob can pull interesting changes from Alice
- Bob can decide to automatically merge changes from Alice
What’s the advantage over plain git or tunneling git through TOR?

- GNUnet’s pet name system and authentication
- NAT traversal
- Anonymity with onion-routing in CADET
- Signalling between gnunet-git-daemons
- Arguably more user-friendly