git-as-svn User Manual

Artem Navrotskiy, Marat Radchenko, Andrew Thornton

Version 1.21.9, 2019-11-30

Table of Contents

Chapter 1. About project

1.1. What is it?

git-as-svn (https://github.com/bozaro/git-as-svn) emulates Subversion repository on top of Git repository.

It allows you to work with Git repositories using any tool compatible with Subversion 1.8+: console `svn`, TortoiseSVN, SvnKit, SmartSVN, etc.

1.2. Features

This implementation allows the majority of Subversion-users to work without thinking about what they actually use Git-repository.

- Nearly all Subversion commands:
  - svn checkout, update, switch, diff
  - svn commit
  - svn copy, move [1: Operations are supported, but copy/move information is not explicitly saved to repository. Instead, it is auto-calculated from Git commits]
  - svn cat, ls
  - svn lock, unlock
  - svnsync
- Path-based authorization
- Transparent mapping of .gitattributes/.gitignore to Subversion properties
- Git LFS, including locks
- Git submodules [2: Git submodule data available in readonly mode.]
- LDAP user authentication
- GitLab integration
- Gitea integration

Empty directories are not supported

1.3. What is project goal?

The project is designed to allow you to work with the same repository as Git, Subversion and style.

Git style

The basic idea is that the developer works in the local branch. His changes do not affect the work of other developers, but nonetheless they can be tested on CI farm, review by another developer and etc.

This allows each developer to work independently, as best he can. He can change and saving intermediate versions of documents, taking full advantage of the version control system (including access to the change history) even without network connection to the server.

Unfortunately, this approach does not work with not mergeable documents (for example, binary files).

Subversion style

The use of a centralized version control system is more convenient in the case of documents do not support the merge (for example, with binary files) due to the presence of the locking mechanism and a simpler and shorter publication cycle changes.

The need to combine Git and Subversion style work with one repository arises from the fact that different employees in the same project are working from fundamentally different data. If you overdo, you Git programmers, and artists like Subversion.

1.4. Why do we need it?

This project was born out of division teams working on another project into two camps:

- People who have tasted the Git and do not want to use Subversion (eg programmers);
- People who do not get from Git practical use and do not want to work with him, but love Subversion (eg designers).

To divide the project into two repository desire was not for various reasons.

At this point, saw the project http://git.q42.co.uk/git_svn_server.git with Proof-of-concept implementation svn server for git repository. After this realization svn server on top of git and didn't seem completely crazy idea (now it's just a crazy idea) and started this project.

Chapter 2. Installation

Subversion versions prior to 1.8 are **not** supported because git-as-svn relies on inherited properties Subversion feature.

2.1. .gitattributes

By default, Git uses native line ending for text files and determines whether file is text or not using heuristics that do not match Subversion behavior.

In order to fix this discrepancy, add the following to your `.gitattributes` file:

.gitattributes

```
*   -text
```

This will force Git to store files as-is unless end-of-line conversion is explicitly configured for them. See gitattributes documentation for additional info.

2.2. Installation on Debian/Ubuntu

You can install git-as-svn repository on Debian/Ubuntu using the following commands:

```
# Add bintray GPG key
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
379CE192D401AB61

# Add repository and fetch its contents
echo "deb https://dl.bintray.com/bozaro/git-as-svn debian main" | sudo tee
/etc/apt/sources.list.d/git-as-svn.list
sudo apt-get update

# Install git-as-svn
sudo apt-get install git-as-svn

# You only need this if you plan to use git-as-svn builtin Git-LFS server
sudo apt-get install git-as-svn-lfs
```

2.2.1. git-as-svn package

This package contains the git-as-svn.

After you install git-as-svn is run in daemon mode and is available on the svn-protocol on port 3690. The daemon runs as `git` user.

To access the server, you can use the user:

Login: test

Password: test

You can check configuration with command like:

```
svn ls --username test --password test svn://localhost/example/
```

Used directories

This package by default is configured to use the following directories:

/etc/git-as-svn
: This directory contains git-as-svn configuration files.

/usr/share/doc/git-as-svn
: This directory contains git-as-svn documentation.

/var/git/lfs
: This directory contains Git Large File Storage files.

  It must be writable by user `git`.

/var/git/repositories
: This directory is used by default to store the Git-repositories.

  It must be writable by user `git`.

/var/log/git-as-svn
: This directory is used to record log files.

  It must be writable by user `git`.

  See logging documentation on log configuration.

/var/cache/git-as-svn
: This directory is used to store the git-as-svn cache.

  It must be writable by user `git`.

  The loss of the contents of this directory is not critical for operation and does not entail the loss of user data.

2.2.2. git-as-svn-lfs package

This package contains the `git-lfs-authenticate` script required for git-as-svn builtin LFS server

2.3. Manual download

To try git-as-svn you need:

1. Install Java 8 or later;
2. Download archive from site https://github.com/bozaro/git-as-svn/releases/latest;
3. After unpacking the archive change working path to the uncompressed directory and run the command:

   ```
   bin/git-as-svn -c doc/examples/config.yml
   ```

This will start git-as-svn server with following configuration:

1. The server is accessible via svn-protocol on port 3690.

   You can check server with command like:

   ```
   svn ls svn://localhost/example
   ```

2. To access the server, you can use the user:

   Login: test

   Password: test

3. Cache and repository will be created in `build` directory:
   - `example.git` — repository directory, accessible via svn-protocol;
   - `git-as-svn.mapdb*` — cache files for expensive computed data.

Chapter 3. Command-line parameters

git-as-svn supports the following command-line parameters:

`-?` | `-h` | `--help`
: print help for command-line parameters.

`-c <file>` | `--config <file>`
: use an alternative configuration *file* instead of a default file.

`-t`
: test the configuration file: git-as-svn checks the configuration for correct syntax.

`-t` doesn't perform full git-as-svn initialization, so it is still possible that git-as-svn will fail to startup due to invalid configuration even though `-t` passed successfully.

`-T`
: same as `-t`, but additionally dump configuration files to standard output.

`-v` | `--version`
: print git-as-svn version.

Chapter 4. GitLab integration

git-as-svn supports integration with GitLab >= 9.3.3.

This includes:

- User authentication against GitLab
- Access control depending on user permissions in GitLab
- Usage of GitLab LFS server for transparent handling of LFS files for svn users
- Automatic discovery of new repositories created in GitLab
- Running GitLab repository hooks if any installed

4.1. Configuration

This chapter assumes that GitLab is installed using standard Omnibus installation to `/opt/gitlab`.

git-as-svn uses direct file access to Git repositories, so it needs to run from the same user as GitLab (normally, `git`). If you're installing both git-as-svn and Gitlab from Debian packages, no additional actions are required.

git-as-svn requires sudo access token permission.

1. Add variables required for GitLab hooks to `/etc/default/git-as-svn`:

```
PATH="/bin:/usr/bin:/opt/gitlab/embedded/bin"
GITLAB_SHELL_DIR=/opt/gitlab/embedded/service/gitlab-shell
GITALY_GITLAB_SHELL_DIR=/opt/gitlab/embedded/service/gitlab-shell
```

2. Change `userDB` to `!gitlabUsers`. This will tell git-as-svn to authenticate users against GitLab server:

```
userDB: !gitlabUsers {}
```

3. Configure builtin git-as-svn webserver:

```
shared:
  - !web

    # git-as-svn base url. Leave empty for autodetect.
    # Default: empty
    #
    # baseUrl: http://localhost:8123/

    listen:
      - !http

        # The network interface where git-as-svn web server binds to as an IP
address or a hostname.  If 0.0.0.0, then bind to all interfaces.
        # Default: localhost
        #
        # host: localhost

        # Port where git-as-svn web server listens on.
        # Default: 8123
        #
        # port: 8123

        # HTTP idle timeout milliseconds. If not a single byte is sent or received
over HTTP connection, git-as-svn closes it.
        # -1 = Use Jetty default
        # 0 = Disable timeout
        # Default: -1
        #
        # idleTimeout: -1

        # Tells git-as-svn to handle X-Forwarded-* headers.
        # Enable this if git-as-svn web server is running behind reverse HTTP proxy
(like nginx)
        # Default: false
        #
        # forwarded: false
```

4. Configure GitLab address and token:

```
shared:
  - !gitlab

    # GitLab base URL. This must match GitLab EXTERNAL_URL.
    # Default: http://localhost/
    #
    # url: http://localhost/

    # Tells git-as-svn to use GitLab for LFS objects and file locking
    # Default: !httpLfs {}
    #
    # lfsMode: !httpLfs {}

    # GitLab access token. Note that git-as-svn requires sudo access.
    token: <GitLab Access Token>
```

5. Configure git-as-svn to use GitLab as repository list source:

```
repositoryMapping: !gitlabMapping

  # Filesystem location where GitLab stores repositories
  # Note that git-as-svn requires write access
  # Default: /var/opt/gitlab/git-data/repositories/
  #
  path: /var/opt/gitlab/git-data/repositories/

  # Common settings for all repositories exposed to svn://
  #
  template:
    pusher: !pushEmbedded
      # This tells git-as-svn where GitLab commit hooks are located
      hooksPath: /opt/gitlab/embedded/service/gitaly-ruby/git-hooks
```

6. Add `git-as-svn:<branch>` topics to whatever repositories you want to add to git-as-svn via "Settings → General → Topics" in GitLab project settings. For example, add `git-as-svn:master` to expose `master` branch. If you want to expose more than one branch, add multiple `git-as-svn:<branch>` topics separated by commas.

4.2. Supported Git LFS modes

1. git-as-svn uses GitLab LFS API for all LFS operations

```
lfsMode: !httpLfs {}
```

2. git-as-svn doesn't use LFS at all

```
lfsMode: null
```

3. git-as-svn uses GitLab LFS API for write operations and direct disk access for read operations

```
lfsMode: !fileLfs
  # Directory where GitLab stores LFS ojects
  path: /var/opt/gitlab/gitlab-rails/shared/lfs-objects
```

4.3. Full configuration file example

/etc/git-as-svn/git-as-svn.conf

```
!config:

# Specifies IP to listen to for svn:// connections
# Default: 0.0.0.0
#
# host: 0.0.0.0

# Specifies port number to listen to for svn:// connections
# Default: 3690
#
# port: 3690

# Subversion realm name. Subversion uses this for credentials caching
# Default: git-as-svn realm
#
# realm: git-as-svn realm

# Traffic compression level. Supported values: LZ4, Zlib, None
# Default: LZ4
#
# compressionLevel: LZ4

# If enabled, git-as-svn indexed repositories in parallel during startup
# This results in higher memory usage so may require adjustments to JVM memory options
# Default: true
#
# parallelIndexing: true

# Sets cache location
cacheConfig: !persistentCache
  path: /var/cache/git-as-svn/git-as-svn.mapdb

# Tells git-as-svn to use GitLab API for repository list
repositoryMapping: !gitlabMapping
```

```
  # Filesystem location where GitLab stores repositories
  # Note that git-as-svn requires write access
  # Default: /var/opt/gitlab/git-data/repositories/
  #
  # path: /var/opt/gitlab/git-data/repositories/

  # Common settings for all repositories exposed to svn://
  #
  template:
    pusher: !pushEmbedded
      # This tells git-as-svn where GitLab commit hooks are located
      hooksPath: /opt/gitlab/embedded/service/gitaly-ruby/git-hooks

# Tells git-as-svn to authenticate users against GitLab
userDB: !gitlabUsers {}

shared:

  # git-as-svn builtin web server
  # It is used for GitLab system hook for repository creation/deletion notifications
  # Also, git-as-svn builtin LFS server is served through it
  - !web

    # git-as-svn base url. Leave empty for autodetect.
    # Default: empty
    #
    # baseUrl: http://localhost:8123/

    listen:
      - !http {

        # The network interface where git-as-svn web server binds to as an IP address
or a hostname.  If 0.0.0.0, then bind to all interfaces.
        # Default: localhost
        #
        # host: localhost

        # Port where git-as-svn web server listens on.
        # Default: 8123
        #
        # port: 8123

        # HTTP idle timeout milliseconds. If not a single byte is sent or received
over HTTP connection, git-as-svn closes it.
        # -1 = Use Jetty default
        # 0 = Disable timeout
        # Default: -1
        #
        # idleTimeout: -1

        # Tells git-as-svn to handle X-Forwarded-* headers.
```

```
        # Enable this if git-as-svn web server is running behind reverse HTTP proxy
(like nginx)
        # Default: false
        #
        # forwarded: false
      }

  # Configures GitLab access for git-as-svn
  - !gitlab

    # GitLab base URL
    # Default: http://localhost/
    #
    # url: http://localhost/

    # Tells git-as-svn to use GitLab for LFS objects and file locking
    # Default: !httpLfs {}
    #
    # lfsMode: !httpLfs {}

    # GitLab access token. Note that git-as-svn requires sudo access.
    token: <GitLab Access Token>
```

Chapter 5. Gitea integration

git-as-svn supports integration with Gitea >= v1.7.2.

This includes:

- User authentication against Gitea
- Access control depending on user permissions in Gitea
- Usage of Gitea LFS server for transparent handling of LFS files for svn users
- Automatic discovery of new repositories created in Gitea
- Running Gitea repository hooks if any installed

git-as-svn requires Sudo Gitea token

git-as-svn uses direct file access to Git repositories, so it needs to run from the same user as Gitea

5.1. Configuration file example

/etc/git-as-svn/git-as-svn.conf

```
!config:

# Specifies IP to listen to for svn:// connections
# Default: 0.0.0.0
#
# host: 0.0.0.0

# Specifies port number to listen to for svn:// connections
# Default: 3690
#
# port: 3690

# Subversion realm name. Subversion uses this for credentials caching
# Default: git-as-svn realm
#
# realm: git-as-svn realm

# Traffic compression level. Supported values: LZ4, Zlib, None
# Default: LZ4
#
# compressionLevel: LZ4

# If enabled, git-as-svn indexed repositories in parallel during startup
# This results in higher memory usage so may require adjustments to JVM memory options
# Default: true
#
```

```
# parallelIndexing: true

# Sets cache location
cacheConfig: !persistentCache
  path: /var/cache/git-as-svn/git-as-svn.mapdb

# Tells git-as-svn to use Gitea API for repository list
repositoryMapping: !giteaMapping

  # Filesystem location where Gitea stores repositories
  # Note that git-as-svn requires write access
  path: /data/git/repositories

  # Common settings for all repositories exposed to svn://
  #
  # template:
  #   branches:
  #     - master
  #   renameDetection: true

# Tells git-as-svn to use Gitea API for user authentication
userDB: !giteaUsers {}

shared:
  # Configures Gitea API for git-as-svn
  - !gitea

    # URL where your Gitea instance API is available
    url: http://localhost:3000/api/v1

    # Tells git-as-svn to store Git-LFS objects through Gitea LFS API
    # Note that this needs to be in sync with Gitea LFS_START_SERVER config option
    lfs: false

    # Gitea access token
    # Note that git-as-svn requires Gitea Sudo permission in order to authenticate
users
    token: 90c68b84fb04e364c2ea3fc42a6a2193144bc07d
```

Chapter 6. LFS server

git-as-svn has built-in Git Large File Storage server

6.1. Configuration file example

/etc/git-as-svn/git-as-svn.conf

```
shared:

  # git-as-svn builtin web server
  # It is used for GitLab system hook for repository creation/deletion notifications
  # Also, git-as-svn builtin LFS server is served through it
  - !web

    # git-as-svn base url. Leave empty for autodetect.
    # Default: empty
    #
    # baseUrl: http://localhost:8123/

    listen:
      - !http {

        # The network interface where git-as-svn web server binds to as an IP address
or a hostname.  If 0.0.0.0, then bind to all interfaces.
        # Default: localhost
        #
        # host: localhost

        # Port where git-as-svn web server listens on.
        # Default: 8123
        #
        # port: 8123

        # HTTP idle timeout milliseconds. If not a single byte is sent or received
over HTTP connection, git-as-svn closes it.
        # -1 = Use Jetty default
        # 0 = Disable timeout
        # Default: -1
        #
        # idleTimeout: -1

        # Tells git-as-svn to handle X-Forwarded-* headers.
        # Enable this if git-as-svn web server is running behind reverse HTTP proxy
(like nginx)
        # Default: false
        #
        # forwarded: false
      }

  # Git LFS server
  - !localLfs
    # Secret token for git-lfs-authenticate script
    # secretToken:
    path: /var/git/lfs
```

6.2. git-lfs-authenticate

Script `git-lfs-authenticate` (provided by git-as-svn-lfs package) is used by git-lfs to obtain credentials for HTTP access to Git LFS server for Git-users working with Git repository by SSH (https://github.com/github/git-lfs/blob/master/docs/api/README.md).

To check the settings of the script can be run locally on the server the following command:

```
# Set environment variable defined in configuration file
export GL_ID=key-1
# Check access to repository
sudo su git -c "git-lfs-authenticate example download"
```

Or on the client the following command:

```
ssh git@remote -C "git-lfs-authenticate example download"
```

The output should look something like this:

```
{
  "href": "https://api.github.com/lfs/bozaro/git-as-svn",
  "header": {
    "Authorization": "Bearer SOME-SECRET-TOKEN"
  },
  "expires_at": "2016-02-19T18:56:59Z"
}
```

6.3. Running git-a-svn behind Nginx reverse proxy

- Add git-as-svn upstream server:

  /etc/nginx/nginx.conf

  ```
  upstream gitsvn {
    server      localhost:8123  fail_timeout=5s;
    keepalive   100;
  }
  ```

- Add resource redirection:

/etc/nginx/nginx.conf

```
location ~ ^.*\.git/info/lfs/ {
  proxy_read_timeout      300;
  proxy_connect_timeout   300;
  proxy_redirect          off;

  proxy_http_version  1.1;
  proxy_set_header    Connection          "";

  proxy_set_header    Host                $http_host;
  proxy_set_header    X-Real-IP           $remote_addr;
  proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
  proxy_set_header    X-Forwarded-Proto   $scheme;
  proxy_set_header    X-Frame-Options     SAMEORIGIN;

  proxy_pass http://gitsvn;
}
```

Also you need to set `baseUrl` parameter in `!web` section of git-as-svn configuration file to external URL accessible to LFS users.

Chapter 7. LDAP (Lightweight Directory Access Protocol)

git-as-svn supports LDAP for user authentication. Refer to your LDAP server documentation to find out what configuration is appropriate in your case.

Internally, git-as-svn uses UnboundID LDAP SDK for Java for all LDAP communication.

/etc/git-as-svn/git-as-svn.conf

```
# Authenticates a user by binding to the directory with the DN of the entry for that
user and the password
# presented by the user. If this simple bind succeeds the user is considered to be
authenticated.
userDB: !ldapUsers

  # LDAP server URL
  # It usually specifies the domain name of the directory server to connect to,
  # and optionally the port number and distinguished name (DN) of the required root
naming context.
  # For secure connections, use ldaps://
  #
  connectionUrl: ldap://localhost:389/ou=groups,dc=mycompany,dc=com

  # Optional LDAP SSL certificate for secure LDAP connections
  #
  # ldapCertPem: /path/to/ldap.pem

  # Pattern specifying the LDAP search filter to use after substitution of the
username.
  #
  searchFilter: (&(objectClass=person)(objectClass=user))

  # LDAP bind configuration
  #
  # [see next documentation section]

  # LDAP attribute, containing user login.
  # Default: sAMAccountName
  #
  # loginAttribute: sAMAccountName

  # LDAP attribute, containing user name.
  # Default: name
  #
  # nameAttribute: name

  # LDAP attribute, containing user email.
  # Default: mail
  #
  # emailAttribute: mail
```

7.1. Supported LDAP bind methods

7.1.1. ANONYMOUS

Performs SASL ANONYMOUS bind as described in RFC 4505.

This is default bind type.

```
userDB: !ldapUsers
  bind: !ANONYMOUS {}
```

7.1.2. CRAM-MD5

Performs SASL CRAM-MD5 bind as described in draft-ietf-sasl-crammd5.

```
userDB: !ldapUsers
  bind: !CRAMMD5
    authenticationID: <required>
    password: <required>
```

7.1.3. DIGEST-MD5

Performs SASL DIGEST-MD5 bind as described in RFC 2831.

```
userDB: !ldapUsers
  bind: !DIGESTMD5
    authenticationID: <required>
    authorizationID: <optional>
    password: <required>
    realm: <optional>
```

7.1.4. EXTERNAL

Performs SASL EXTERNAL bind as described in RFC 4422.

```
userDB: !ldapUsers
  bind: !EXTERNAL
    authenticationID: <optional>
```

7.1.5. PLAIN

Performs SASL PLAIN bind as described in RFC 4616.

```
userDB: !ldapUsers
  bind: !PLAIN
    authenticationID: <required>
    authorizationID: <optional>
    password: <required>
```

7.1.6. Simple

Performs LDAPv3 simple bind operation.

```
userDB: !ldapUsers
  bind: !Simple
    bindDn: <optional>
    password: <optional>
```

Chapter 8. Logging

git-as-svn uses Apache Log4j 2 for logging. Configuration file is located in `/etc/git-as-svn/log4j2.xml`. Please, refer to Log4j 2 documentation on this file format and available options.

By default, all messages with INFO priority and higher are logged to `/var/log/git-as-svn/git-as-svn.log` and rotated at startup and per each 10 MB. Also, all messages with ERROR priority and higher are logged to `/var/log/git-as-svn/git-as-svn.error.log` with same rotation policy.

8.1. Loggers available in git-as-svn

- `git` - messages related to operations with Git repositories
- `gitea` - messages related to communication with Gitea
- `gitlab` - messages related to communication with GitLab
- `ldap` - messages related to LDAP user authentication
- `lfs` - messages related to Git LFS, including both internal and external LFS server if any of them is configured
- `misc` - few unsorted messages, mostly related to startup/shutdown procedures
- `svn` - messages related to incoming SVN connections
- `web` - messages related to builtin HTTP server

Additionally, git-as-svn uses some third-party libraries, most notable are JGit and Jetty, that can also log various stuff. Please, refer to their appropriate documentation on available loggers for these projects.

Chapter 9. Path-based authorization

This feature is currently only supported for `repositoryMapping: !listMapping`

git-as-svn supports path-based authorization that allows granting (or denying) permissions to users, very similar to Subversion path-based authorization feature. Typically this is done over the entire repository: a user can read the repository (or not), and they can write to the repository (or not).

It's also possible, however, to define finer-grained access rules. One set of users may have permission to write to a certain directory in the repository, but not others; another directory might not even be readable by all but a few special people. It's even possible to restrict access on a per file basis.

9.1. Getting Started with Path-Based Access Control

Here's a simple example demonstrating a piece of the access configuration which grants read access Sally, and read/write access to Harry, for the path `/path/to/directory/` (and all its children) in the repository `calc`:

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /path/to/directory:
          harry: rw
          sally: r
```

Permissions are inherited from a path's parent directory. That means we can specify a subdirectory with a different access policy for Sally. Let's continue our previous example, and grant Sally write access to a child of the directory that she's otherwise permitted only to read:

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /path/to/directory:
          harry: rw
          sally: r
        /path/to/directory/subdirectory:
          sally: rw
```

Now Sally can write to subdirectory, but can still only read other parts. Harry, meanwhile, continues to have complete read/write access to the whole directory.

It's also possible to explicitly deny permission to someone via inheritance rules, by using empty string or `none`:

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /path/to/directory:
          harry: rw
          sally: r
        /path/to/directory/secret:
          harry: none
```

In this example, Harry has read/write access to the entire directory, but has absolutely no access at all to the `secret` subdirectory within it.

The thing to remember is that the most specific path always matches first. The server tries to match the path itself, and then the parent of the path, then the parent of that, and so on. The net effect is that mentioning a specific path in the access file will always override any permissions inherited from parent directories.

By default, nobody has any access to any repository at all. If you want to give at least read permission to all users at the roots of the repositories. You can do this by using the asterisk variable (`*`), which means "all users":

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /:
          '*': r
```

Note that while all of the previous examples use directories, that's only because defining access rules on directories is the most common case. You may similarly restrict access on file paths, too.

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /README.md:
          harry: rw
          sally: r
```

You may also specify grant or restrict access only to specific branches.

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /README.md:
          harry: r
        master:/README.md:
          harry: rw
```

In this example, Harry has read access to file on all branches but has read/write access on master branch.

9.2. Access Control Groups

git-as-svn also allows you to define whole groups of users. To do this, describe your groups within `groups` section of `git-as-svn.conf` :

git-as-svn.conf

```
repositoryMapping: !listMapping
  groups:
    calc-developers:
      - harry
      - sally
      - joe
    paint-developers:
      - frank
      - sally
      - jane
```

Groups can be granted access control just like users. Distinguish them with an "at sign" (`@`) prefix:

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calc:
      access:
        /:
          '@calc-developers': rw
    paint:
      access:
        /:
          'jane': r
          '@paint-developers': rw
```

Another important fact is that group permissions are not overridden by individual user permissions. Rather, the *combination* of all matching permissions is granted. In the prior example, Jane is a member of the `paint-developers` group, which has read/write access. Combined with the `jane = r` rule, this still gives Jane read/write access. Permissions for group members can only be extended beyond the permissions the group already has. Restricting users who are part of a group to less than their group's permissions is impossible.

Groups can also be defined to contain other groups:

git-as-svn.conf

```
repositoryMapping: !listMapping
  groups:
    calc-developers:
      - harry
      - sally
      - joe
    paint-developers:
      - frank
      - sally
      - jane
    everyone:
      - '@calc-developers'
      - '@paint-developers'
```

User needs read/write access to `/` path of `master` branch in order to be able to download/upload files from git-as-svn internal LFS server.

9.3. Advanced Access Control Features

git-as-svn also supports some "magic" tokens for helping you to make rule assignments based on the user's authentication class. One such token is the `$authenticated` token. Use this token where you would otherwise specify a username or group name in your authorization rules to declare the permissions granted to any user who has authenticated with any username at all. You may also use `$authenticated:Local`/`$authenticated:GitLab`/`$authenticated:Gitea`/`$authenticated:LDAP` to refer to users authenticated against specific user database. Similarly employed is the `$anonymous` token, except that it matches everyone who has not authenticated with a username.

git-as-svn.conf

```
repositoryMapping: !listMapping
  repositories:
    calendar:
      access:
        /:
          '$anonymous': r
          '$authenticated': rw
```

Chapter 10. SVN Properties

The main `svn properties` trouble that they should be maintained in the synchronous state between Git and Subversion.

Because of this arbitrary `svn properties` is not supported. To value `svn properties` correspond Git-view, they are generated on the fly based on the repository content. `

Wherein:

- the commit verifies that `svn properties` file or directory exactly match what should be according to the data repository;
- Subversion does not tool allows you to change most of the properties (exception: `svn:executable`, `svn:special`);
- if a file affects the `svn properties` other files after changing it `svn properties` of the files in the same change.

For user convenience, git-as-svn is actively using the inherited properties.

This feature requires Subversion 1.8 or later.

Otherwise there will be problems with the svn properties for new files and directories.

10.1. File .gitignores

This file affects the property `svn:ignore` and `svn:global-ignores` for the directory and its subdirectories.

For example, a file in the directory `/foo` with the contents:

```
.idea/libraries
*.class
*/build
```

Mapped to properties:

- for directory `/foo`:

  ```
  svn:global-ignores: *.class
  ```

- for directory `/foo/*`:

  ```
  svn:ignore: build
  ```

- for directory `/foo/.idea`:

```
svn:ignore: libraries build
```

For Subversion has no way to make an exception for directories, as a result, for example, the rules of `/foo` (file or directory `foocode>) and /foo/ (directory foo) in Subversion will work the same way, though to Git they have different behavior.`

Terms like "all but" not supported on mapping to the `svn:global-ignores` property.

10.2. File .gitattributes

This file affects the properties of the `svn:eol-style` and `svn:mime-type` files from this directory and `svn:auto-props` from the directory itself.

For example, a file with contents:

```
*.txt           text eol=native
*.xml           eol=lf
*.bin           binary
```

Add property to the directory `svn:auto-props` with the contents:

```
*.txt = svn:eol-style=native
*.xml = svn:eol-style=LF
*.bin = svn:mime-type=application/octet-stream
```

And files in this directory:

- for suffix `.txt` add property `svn:eol-style = navtive`
- for suffix `.xml` add property `svn:eol-style = LF`
- for suffix `.bin` add property `svn:mime-type = application/octet-stream`

10.3. File .tgitconfig

This file only changes the properties of the directory in which it is located.

Properties are mapped one-to-one, for example, a file with the contents:

```
[bugtraq]
    url = https://github.com/bozaro/git-as-svn/issues/%BUGID%
    logregex = #(\\d+)
    warnifnoissue = false
```

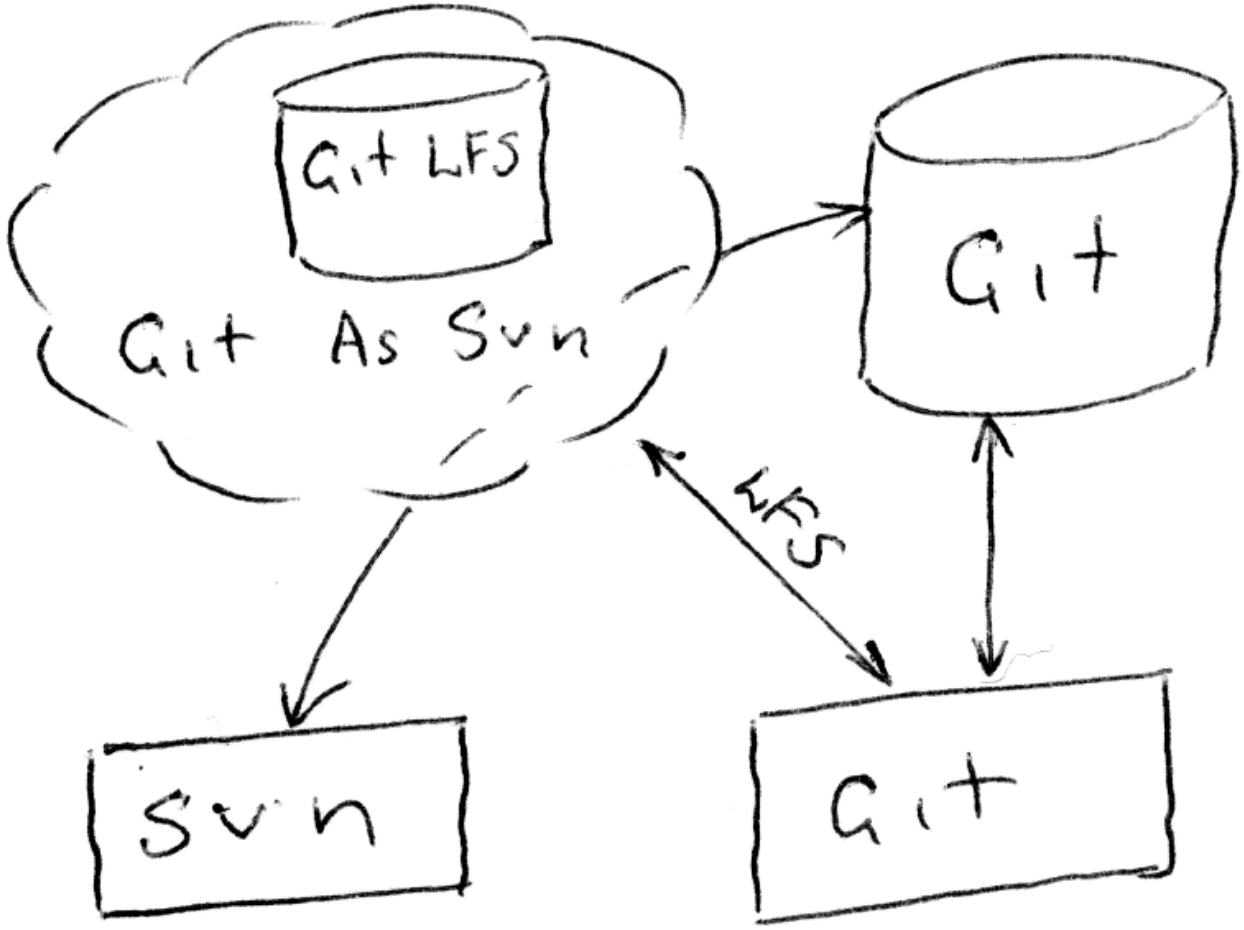
It will be converted to properties:

- `bugtraq:url = https://github.com/bozaro/git-as-svn/issues/%BUGID%`
- `bugtraq:logregex = #(\\d+)`
- `bugtraq:warnifnoissue = false`

If you use bugtraq svn properties, it is highly recommended that you use TortoiseSVN 1.9 or later.

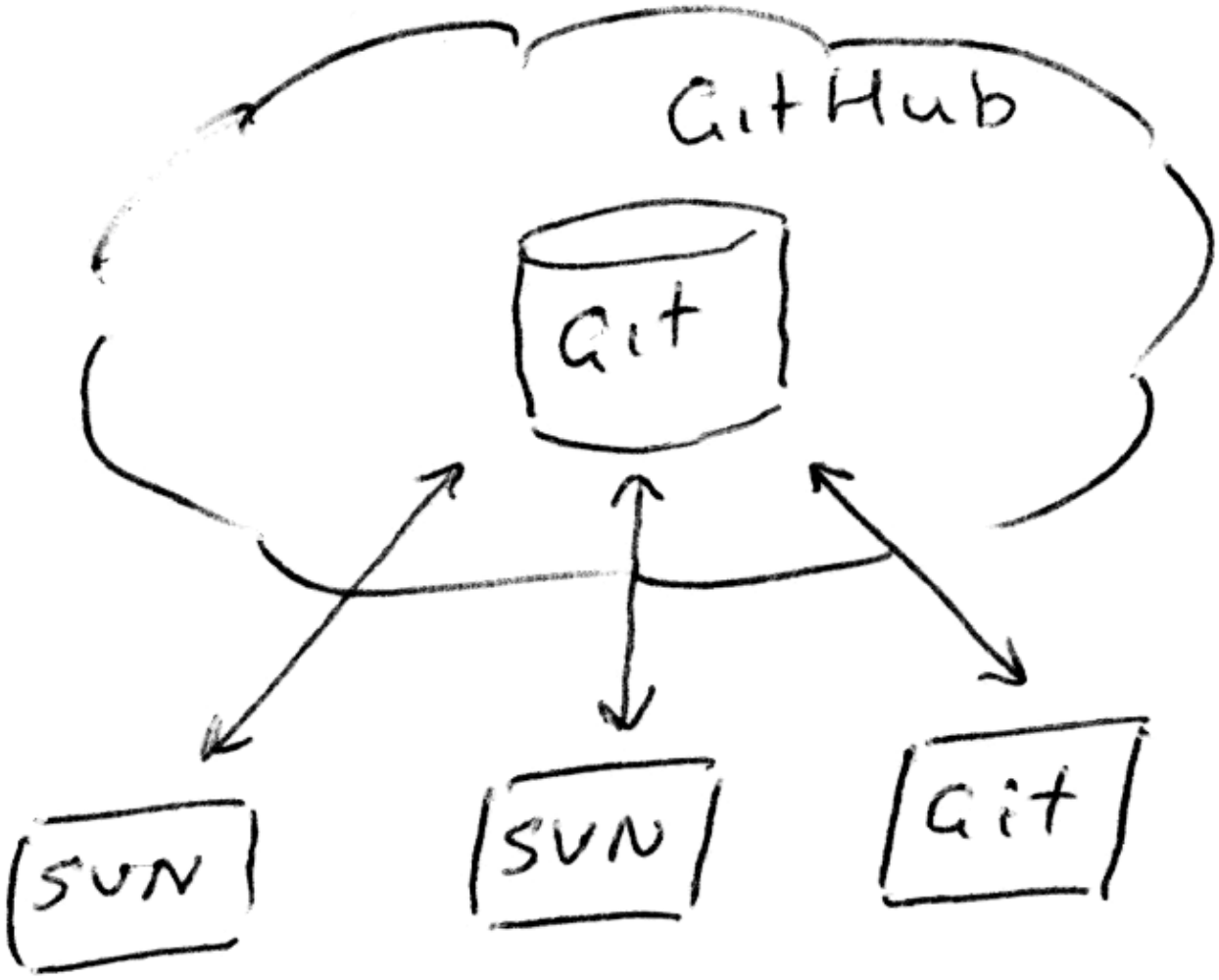
Otherwise TortoiseSVN will attempt to set these parameters for all newly created directories instead of use inherited properties.

Chapter 11. Alternatives



The problem of combining Git and Subversion work style with a version control system can be solved in different ways.

11.1. GitHub Subversion support



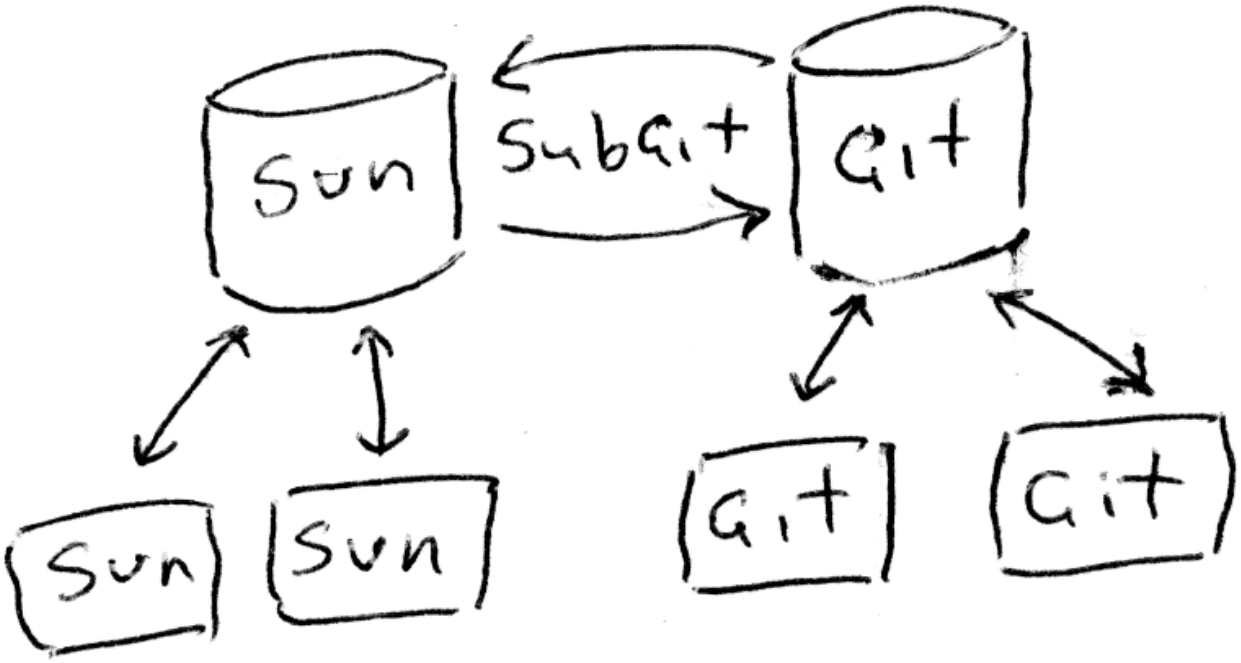
Website: https://help.github.com/en/articles/support-for-subversion-clients

This is probably the closest analogue.

The main problem of this implementation is inseparable from GitHub. Also, all of a sudden, this implementation does not support Git LFS.

In the case of GitHub it is also not clear where the stored mapping between Subversion-revision and Git-commit. This can be a problem when restoring repositories after emergency situations.

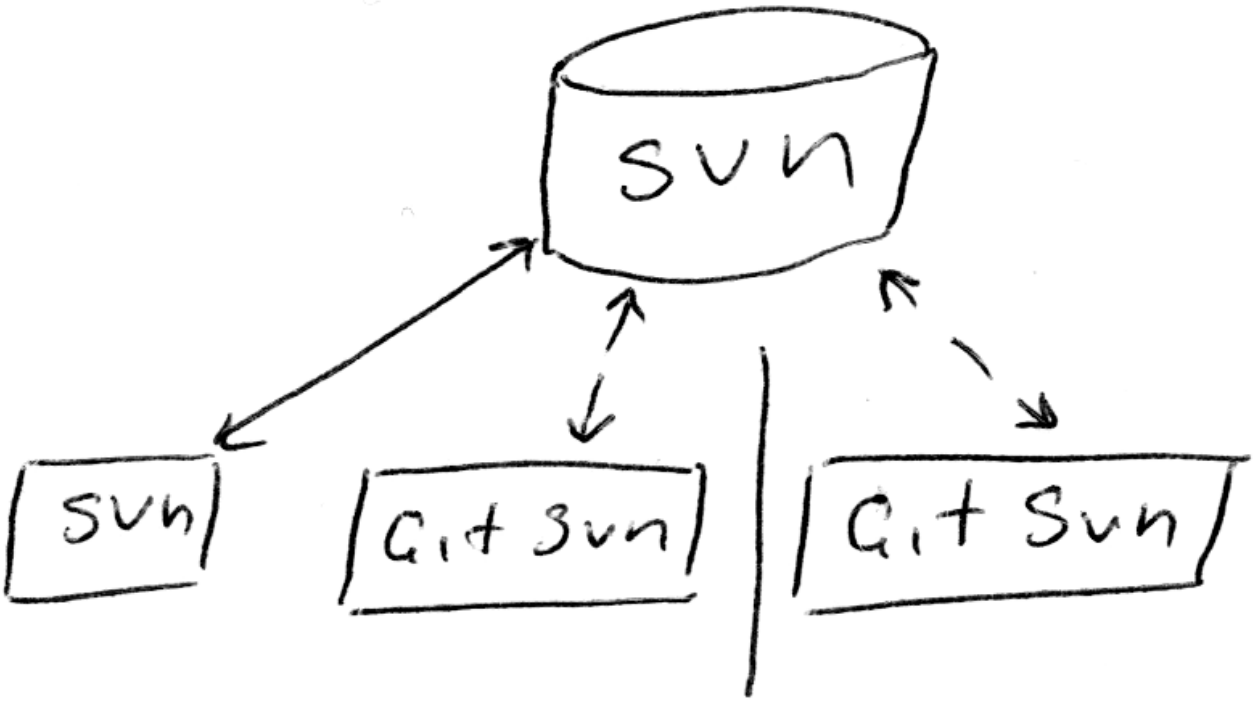
11.2. SubGit



Website: https://subgit.com

Quite an interesting implementation which supports master-master replication with Git and Subversion repositories. Thereby providing synchronization of repositories is not clear.

11.3. Subversion repository and git svn



This method allows you to use Git with Subversion repository, but using a shared Git repository between multiple developers very difficult.

At the same time, the developer has to use a specific command-line tool for working with the repository.

Chapter 12. SVN+SSH

12.1. Rationale

The SVN protocol is totally unencrypted, and due to the way git-as-svn has to proxy authentication through to git servers, almost all authentication happens in plaintext.

Clearly this is undesirable, not only is potentially private code exposed over the svn protocol, but so are passwords and usernames.

Traditionally SVN has two ways of preventing this:

- Use HTTPS
- Use svn+ssh

The HTTP protocol is substantially different from the SVN protocol and is currently unimplemented in git-as-svn

Thus leaving the svn+ssh mechanism.

12.2. How does SVN+SSH work?

Normally when a client calls `svn <command> svn://host/path`, for an appropriate `<command>`, the subversion client will open a connection to the `host` server on port 3690. After an initial handshake as per the SVN protocol the server will ask the client to authenticate.

If possible the client will attempt to perform its actions anonymously, and if necessary the server will then ask for reauthentication.

If a client calls `svn <command> svn+ssh//username@host/path`, the subversion client will internally ask ssh to open connection using something equivalent to: `ssh username@host "svnserve -t"`.

If ssh succesfully connects, the SSH will run `svnserve -t` on the host, which will then proceed with the SVN protocol handshake over its `stdin` and `stdout`, and the client will use the `stdin` and `stdout` of the ssh connection.

When the server asks the client to authenticate, the server will offer the `EXTERNAL` authentication mechanism. (Possibly with the `ANONYMOUS` mechanism.)

If the client uses `EXTERNAL` mechanism, the server sets the user to be either the currently logged in user from the ssh, (or an optional tunnel-user parameter.)

Securing the `svnserve -t` call and protecting against semi-malicious uses of the `--tunnel-user` option or even the calling of other commands in cases of multiple users for a single repository requires some thought.

Often this is protected through the use of a suitable `command=""` parameter in the `authorized_keys` file, coupled with other options. e.g.

`command="/usr/bin/svnserve -t --tunnel-user username",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty ssh-rsa ⋯`

Of note, in this example the command provided by the client is ignored but it could be checked and managed as appropriately. In fact these techniques are used in the `authorized_keys` files of most `git` servers.

This provides a simple first way to handle `svn+ssh`, if we set `command="nc localhost 3690"` then whenever we connect by ssh we will be passed directly to the git-as-svn server. The downside being that the client will be asked to authenticate.

12.3. A better `git-as-svn-svnserve`

Handling the `EXTERNAL` authentication mechanism properly without creating a new port to listen on and a new adjusted SVN protocol is not possible.

However there is another way:

We can stand in the middle of the SVN protocol stream, catch the authentication handshake, proxy it before stepping back and letting the client and server talk to each other.

We can create a new authentication mechanism on the `git-as-svn` server that requires a secret token known only by us, to allow us to pass in the external username (or other identifier) as the user authentication using `sshKeyUsers` to proxy the `UserDB`

We can then use `git-as-svn-svnserve-tunnel SECRET EXTERNAL_USERNAME` as a replacement for `svnserve -t` or `nc localhost 3690` in the `command=""` option in authorized_keys.

Of course we need to keep the `authorized_keys` file up-to-date

12.4. GitLab & git-as-svn-svnserve

There are two ways that Gitlab manages ssh access.

- Updating the git user's `authorized_keys` every time a SSH key is changed.
- The use of an SSH `AuthorizedKeysCommand`

First, let's look at the `authorized_keys` case.

Gitlab will update the `authorized_keys` file over time.

If you set the option: `gitlab_shell['auth_file']` in the `gitlab.rb` configuration file to a different location, you can catch changes to this file, and change the `command=""` option to something that will check whether we are trying to perform svn and handle it if so.

The suggested config, at least for Gitlab docker and assuming that git-as-svn has been installed in `/opt/git-as-svn` is:

/etc/gitlab/gitlab.rb

```
...
################################################################################
## gitlab-shell
################################################################################
...
# gitlab_shell['auth_file'] = "/var/opt/gitlab/.ssh/authorized_keys"
gitlab_shell['auth_file'] = "/var/opt/gitlab/ssh-shadow/authorized_keys"
...
```

/etc/git-as-svn/git-as-svn.conf

```
!config:

# Specifies IP to listen to for svn:// connections
# Default: 0.0.0.0
#
# host: 0.0.0.0

# Specifies port number to listen to for svn:// connections
# Default: 3690
#
# port: 3690

# Subversion realm name. Subversion uses this for credentials caching
# Default: git-as-svn realm
#
# realm: git-as-svn realm

# Traffic compression level. Supported values: LZ4, Zlib, None
# Default: LZ4
#
# compressionLevel: LZ4

# If enabled, git-as-svn indexed repositories in parallel during startup
# This results in higher memory usage so may require adjustments to JVM memory options
# Default: true
#
# parallelIndexing: true

# Sets cache location
cacheConfig: !persistentCache
  path: /var/cache/git-as-svn/git-as-svn.mapdb

# Tells git-as-svn to use GitLab API for repository list
repositoryMapping: !gitlabMapping

  # Filesystem location where GitLab stores repositories
  # Note that git-as-svn requires write access
```

```
  # Default: /var/opt/gitlab/git-data/repositories/
  #
  # path: /var/opt/gitlab/git-data/repositories/

  # Uncomment following to only handle repositories with specified tags (add them to
repositories via Settings -> General -> Tags in GitLab)
  #
  # repositoryTags:
  #   - git-as-svn

  # Common settings for all repositories exposed to svn://
  template:
    pusher: !pushEmbedded
      # This tells git-as-svn where GitLab commit hooks are located
      hooksPath: /opt/gitlab/embedded/service/gitaly-ruby/git-hooks

# Use GitLab user database
userDB:
  !sshKeyUsers
  userDB: !gitlabUsers {}
  sshKeysToken: CHANGE_THIS_TO_SOMETHING_SECRET

shared:

  # git-as-svn builtin web server
  # It is used for GitLab system hook for repository creation/deletion notifications
  # Also, git-as-svn builtin LFS server is served through it
  - !web

    # git-as-svn base url. Leave empty for autodetect.
    # Default: empty
    #
    # baseUrl: http://localhost:8123/

    listen:
      - !http {

        # The network interface where git-as-svn web server binds to as an IP address
or a hostname.  If 0.0.0.0, then bind to all interfaces.
        # Default: localhost
        #
        # host: localhost

        # Port where git-as-svn web server listens on.
        # Default: 8123
        #
        # port: 8123

        # HTTP idle timeout milliseconds. If not a single byte is sent or received
over HTTP connection, git-as-svn closes it.
        # -1 = Use Jetty default
```

```
        # 0 = Disable timeout
        # Default: -1
        #
        # idleTimeout: -1

        # Tells git-as-svn to handle X-Forwarded-* headers.
        # Enable this if git-as-svn web server is running behind reverse HTTP proxy
(like nginx)
        # Default: false
        #
        # forwarded: false
      }

  # Configures GitLab access for git-as-svn
  - !gitlab

    # GitLab base URL
    # Default: http://localhost/
    #
    # url: http://localhost/

    # Tells git-as-svn to use GitLab for LFS objects and file locking
    # Default: !httpLfs {}
    #
    # lfsMode: !httpLfs {}

    # GitLab access token. Note that git-as-svn requires sudo access.
    token: <GitLab Access Token>

  # Manage authorized_keys if your Gitlab instance creates this file
  - !sshKeys
    shadowSSHDirectory: /var/opt/gitlab/ssh-shadow
    realSSHDirectory: /var/opt/gitlab/.ssh
    originalAppPath: /opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell
    svnservePath: /opt/git-as-svn/bin/git-as-svn-svnserve
  # If your gitlab instance is using AuthorizedKeysCommand
  # look at tools/git-as-svn-authorized-keys-command
```

/opt/git-as-svn/bin/git-as-svn-svnserve

```
#!/bin/bash

################################################################
# git-as-svn-svnserve
#
# Shadow the default gitlab/gitea shell and allow svnserve
################################################################

################################################################
# For Gitlab Docker:
################################################################
```

```
# SHADOW_SHELL_PATH="/opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell"
# TUNNEL_PATH="/opt/git-as-svn/bin/git-as-svn-svnserve-tunnel"
# KEY="$1"
# REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

################################################################
# For Gitea Docker:
################################################################
SHADOW_SHELL_PATH="/app/gitea/gitea"
TUNNEL_PATH="/app/git-as-svn/git-as-svn-svnserve-tunnel"
KEY="$2"
SUBCOMMAND="$1"
REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

if [ "$SUBCOMMAND" != "serv" ]; then
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi

################################################################
# Other options:
################################################################
# For either, you can move the shadowed binary to something like
# /app/gitea/gitea.shadow and rename this script to /app/gitea/gitea.
#
# If you follow his approach you do not need to rewrite the
# authorized_keys file, but may still need to process it.
#
# You would need to set the REAL_SHELL_PATH to point to this file
# and restore the shadowing on updates to the application
################################################################
SECRET="CHANGE_THIS_TO_SOMETHING_SECRET"

SSH_ORIGINAL_COMMANDS=($SSH_ORIGINAL_COMMAND)

if [ -n "$SSH_ORIGINAL_COMMAND" ] && [ "${SSH_ORIGINAL_COMMANDS[0]}" = "svnserve" ] ;
then
    ## TUNNEL TO OUR SVNSERVER WITH MAGIC AUTHENTICATION ##
    exec "$TUNNEL_PATH" "$SECRET" "$KEY"
else
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi
```

/opt/git-as-svn/bin/git-as-svn-svnserve-tunnel

```
#!/bin/bash

################################################################
# git-as-svn-svnserve-tunnel
#
# Use a bit of bash hackery to implement svnserve -t by
# pushing stdin to the svn port (3690) but hijack the
```

```
# authentication phase to pass in the ssh key id
############################################################

SECRET="$1"
KEY="$2"
FAKE_AUTH="( success ( ( EXTERNAL ) 16:Git-as-svn Realm ) )"

function failed {
    echo "$0: Unable to connect to svn service! Is it running?" 1>&2
    exit
}
trap failed err

OUR_PID=$$
function finish {
    pkill -P $OUR_PID
    exec 3>&- 3<&-
}
trap finish EXIT

exec 3<>/dev/tcp/localhost/3690

trap finish err

function read_bracket {
    BEEN_IN=false
    NBRACK=0

    while ! $BEEN_IN || [ $NBRACK != 0 ]; do
        IFS= read -n1 -r -d '' FROM
        case $FROM in
            '('
            NBRACK=$(($NBRACK + 1))
            BEEN_IN=true
            ;;
            ')')
            NBRACK=$(($NBRACK - 1))
            ;;
            '')
            break
        esac
        echo -ne "$FROM"
    done
    IFS= read -n1 -r -d '' FROM
    echo -ne "$FROM"
    if [ "X$FROM" = "X" ]; then
        exec 0<&-
        exit
    fi
}
```

```
# Send server capabilities to client
read_bracket <&3 >&1

# Send client capabilities to server
read_bracket <&0 >&3

# Get the server authentication
AUTH_LIST_FROM_SERV=$(read_bracket <&3)

# Send the server our information
AUTHBODY=$(echo -ne "\0$SECRET\0$KEY" | base64)
AUTHBODY_LENGTH=${#AUTHBODY}
echo "( KEY-AUTHENTICATOR ( $AUTHBODY_LENGTH:$AUTHBODY ) )" >&3
if ! { command >&3; } 2>/dev/null; then
    exit
fi

# send the fake auth list to the client
echo "$FAKE_AUTH" >&1
if ! { command >&1; } 2>/dev/null; then
    exit
fi

# throwaway the client's response
read_bracket <&0 > /dev/null

# THEN PRETEND THAT THE REST OF IT WENT THAT WAY
(
    cat <&3 >&1 &
    CAT_PID=$!
    function on_exit {
        kill $CAT_PID
    }
    trap on_exit EXIT
    wait
    kill $OUR_PID
) &

cat <&0 >&3
pkill -P $OUR_PID
```

In the second case, if we proxy the `AuthorizedKeysCommand`, and just replace the `command=""` option as above then we have a working solution.

We have two main options, we can keep the same user, e.g. `git` for both subversion and git, or we could create another user.

The first option requires that we proxy the original app and replace it with our own. The second is similar but we leave the original response alone for git, just replacing it for svn

The first option is described below.

/assets/sshd_config

```
...
# AuthorizedKeysCommand /opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell-
authorized-keys-check git %u %k
# AuthorizedKeysCommandUser git
AuthorizedKeysCommand /opt/git-as-svn/bin/git-as-svn-authorized-keys-command git %u %k
AuthorizedKeysCommandUser git
...
```

/opt/git-as-svn/bin/git-as-svn-authorized-keys-command

```
#!/bin/bash

################################################################
# git-as-svn-authorized-keys_command
#
# Shadow the default ssh AuthorizedKeysComand and adjust its
# output to replace the original command with our svnserve
################################################################

################################################################
# For Gitlab Docker:
################################################################
ORIGINAL_AUTHORIZED_COMMAND="/opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-
shell-authorized-keys-check"
ORIGINAL_APP_PATH="/opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell"
SVN_SERVE_PATH="/opt/git-as-svn/bin/git-as-svn-svnserve"

################################################################
# Gitea does not have AuthorizedKeysCommand at present
################################################################

exec -a "$ORIGINAL_AUTHORIZED_COMMAND"  "$ORIGINAL_AUTHORIZED_COMMAND" "$@" | sed -e
's|command="'"$ORIGINAL_APP_PATH"'|command="'"$SVN_SERVE_PATH"'|'
```

/etc/git-as-svn/git-as-svn.conf

```
!config:

# Specifies IP to listen to for svn:// connections
# Default: 0.0.0.0
#
# host: 0.0.0.0

# Specifies port number to listen to for svn:// connections
# Default: 3690
#
# port: 3690
```

```
# Subversion realm name. Subversion uses this for credentials caching
# Default: git-as-svn realm
#
# realm: git-as-svn realm

# Traffic compression level. Supported values: LZ4, Zlib, None
# Default: LZ4
#
# compressionLevel: LZ4

# If enabled, git-as-svn indexed repositories in parallel during startup
# This results in higher memory usage so may require adjustments to JVM memory options
# Default: true
#
# parallelIndexing: true

# Sets cache location
cacheConfig: !persistentCache
  path: /var/cache/git-as-svn/git-as-svn.mapdb

# Tells git-as-svn to use GitLab API for repository list
repositoryMapping: !gitlabMapping

  # Filesystem location where GitLab stores repositories
  # Note that git-as-svn requires write access
  # Default: /var/opt/gitlab/git-data/repositories/
  #
  # path: /var/opt/gitlab/git-data/repositories/

  # Common settings for all repositories exposed to svn://
  #
  template:
    pusher: !pushEmbedded
      # This tells git-as-svn where GitLab commit hooks are located
      hooksPath: /opt/gitlab/embedded/service/gitaly-ruby/git-hooks

# Tells git-as-svn to authenticate users against GitLab
userDB: !gitlabUsers {}

shared:

  # git-as-svn builtin web server
  # It is used for GitLab system hook for repository creation/deletion notifications
  # Also, git-as-svn builtin LFS server is served through it
  - !web

    # git-as-svn base url. Leave empty for autodetect.
    # Default: empty
    #
    # baseUrl: http://localhost:8123/
```

```
    listen:
      - !http {

        # The network interface where git-as-svn web server binds to as an IP address
or a hostname.  If 0.0.0.0, then bind to all interfaces.
        # Default: localhost
        #
        # host: localhost

        # Port where git-as-svn web server listens on.
        # Default: 8123
        #
        # port: 8123

        # HTTP idle timeout milliseconds. If not a single byte is sent or received
over HTTP connection, git-as-svn closes it.
        # -1 = Use Jetty default
        # 0 = Disable timeout
        # Default: -1
        #
        # idleTimeout: -1

        # Tells git-as-svn to handle X-Forwarded-* headers.
        # Enable this if git-as-svn web server is running behind reverse HTTP proxy
(like nginx)
        # Default: false
        #
        # forwarded: false
      }

  # Configures GitLab access for git-as-svn
  - !gitlab

    # GitLab base URL
    # Default: http://localhost/
    #
    # url: http://localhost/

    # Tells git-as-svn to use GitLab for LFS objects and file locking
    # Default: !httpLfs {}
    #
    # lfsMode: !httpLfs {}

    # GitLab access token. Note that git-as-svn requires sudo access.
    token: <GitLab Access Token>
```

- `/opt/git-as-svn/bin/git-as-svn-svnserve` and `/opt/git-as-svn/bin/git-as-svn-svnserve-tunnel` same as above.

12.5. Gitea

There are two ways that Gitea manages ssh access.

- If Gitea is deferring to an external SSHD. It will update the git user's `authorized_keys` every time a SSH key is changed.
- If Gitea is using its own internal SSHD. It will run the `gitea serv` command each time.
- The use of an SSH `AuthorizedKeysCommand` in Gitea v1.7.0+

First, let's look at the `authorized_keys` case.

Gitea will update the `authorized_keys` file over time.

If you set the option: `SSH_ROOT_PATH` in the `[server]` of the gitea `app.ini` to a shadow location you can catch changes to this file, and change the `command=""` option to something that will check whether we are trying to perform svn and handle it if so.

The suggested config, at least for Gitea docker, and assuming that git-as-svn has been installed in `/app/git-as-svn` is:

/data/gitea/conf/app.ini

```
...
[server]
...
SSH_ROOT_PATH=/data/git/ssh-shadow
...
```

/app/git-as-svn/config.yaml

```
!config:

# Specifies IP to listen to for svn:// connections
# Default: 0.0.0.0
#
# host: 0.0.0.0

# Specifies port number to listen to for svn:// connections
# Default: 3690
#
# port: 3690

# Subversion realm name. Subversion uses this for credentials caching
# Default: git-as-svn realm
#
# realm: git-as-svn realm

# Traffic compression level. Supported values: LZ4, Zlib, None
# Default: LZ4
```

```
#
# compressionLevel: LZ4

# If enabled, git-as-svn indexed repositories in parallel during startup
# This results in higher memory usage so may require adjustments to JVM memory options
# Default: true
#
# parallelIndexing: true

# Sets cache location
cacheConfig: !persistentCache
  path: /var/cache/git-as-svn/git-as-svn.mapdb

# Tells git-as-svn to use Gitea API for repository list
repositoryMapping: !giteaMapping

  # Filesystem location where Gitea stores repositories
  # Note that git-as-svn requires write access
  path: /data/git/repositories

  # Common settings for all repositories exposed to svn://
  #
  # template:
  #   branches:
  #     - master
  #   renameDetection: true

userDB:
  !sshKeyUsers
  # Tells git-as-svn to use Gitea API for user authentication
  userDB: !giteaUsers {}
  sshKeysToken: CHANGE_THIS_TO_SOMETHING_SECRET

shared:

  # Configures Gitea API for git-as-svn
  - !gitea

    # URL where your Gitea instance API is running
    url: http://localhost:3000/api/v1

    # Tells git-as-svn to store Git-LFS objects through Gitea LFS API
    # Note that this needs to be in sync with Gitea LFS_START_SERVER config option
    lfs: false

    # Gitea access token
    # Note that git-as-svn requires Gitea Sudo permission in order to authenticate
users
    token: 90c68b84fb04e364c2ea3fc42a6a2193144bc07d

  - !sshKeys
```

```
    shadowSSHDirectory: /data/git/ssh-shadow
    realSSHDirectory: /data/git/.ssh
    originalAppPath: /app/gitea/gitea
    svnservePath: /app/gitea/git-as-svn-svnserve
  # If your gitea instance is using AuthorizedKeysCommand
  # look at tools/git-as-svn-authorized-keys-command
  # You don't need sshKeys in that case
```

/app/git-as-svn/bin/git-as-svn-svnserve

```
#!/bin/bash

################################################################
# git-as-svn-svnserve
#
# Shadow the default gitlab/gitea shell and allow svnserve
################################################################

################################################################
# For Gitlab Docker:
################################################################
# SHADOW_SHELL_PATH="/opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell"
# TUNNEL_PATH="/opt/git-as-svn/bin/git-as-svn-svnserve-tunnel"
# KEY="$1"
# REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

################################################################
# For Gitea Docker:
################################################################
SHADOW_SHELL_PATH="/app/gitea/gitea"
TUNNEL_PATH="/app/git-as-svn/git-as-svn-svnserve-tunnel"
KEY="$2"
SUBCOMMAND="$1"
REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

if [ "$SUBCOMMAND" != "serv" ]; then
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi

################################################################
# Other options:
################################################################
# For either, you can move the shadowed binary to something like
# /app/gitea/gitea.shadow and rename this script to /app/gitea/gitea.
#
# If you follow his approach you do not need to rewrite the
# authorized_keys file, but may still need to process it.
#
# You would need to set the REAL_SHELL_PATH to point to this file
# and restore the shadowing on updates to the application
################################################################
```

```
SECRET="CHANGE_THIS_TO_SOMETHING_SECRET"

SSH_ORIGINAL_COMMANDS=($SSH_ORIGINAL_COMMAND)

if [ -n "$SSH_ORIGINAL_COMMAND" ] && [ "${SSH_ORIGINAL_COMMANDS[0]}" = "svnserve" ] ;
then
    ## TUNNEL TO OUR SVNSERVER WITH MAGIC AUTHENTICATION ##
    exec "$TUNNEL_PATH" "$SECRET" "$KEY"
else
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi
```

- `/app/git-as-svn/bin/git-as-svn-svnserve-tunnel` should be the same as in the gitlab case.

For the second case, we need to shadow the gitea binary

So we would need to move the original gitea from `/app/gitea/gitea` to `/app/gitea/gitea.shadow`

And either create `/app/gitea/gitea` as a symbolic link or just copy the below `/app/git-as-svn/bin/git-as-svn-svnserve` as it.

/app/git-as-svn/bin/git-as-svn-svnserve

```
#!/bin/bash

################################################################
# git-as-svn-svnserve
#
# Shadow the default gitlab/gitea shell and allow svnserve
################################################################

################################################################
# For Gitlab Docker:
################################################################
# SHADOW_SHELL_PATH="/opt/gitlab/embedded/service/gitlab-shell/bin/gitlab-shell"
# TUNNEL_PATH="/opt/git-as-svn/bin/git-as-svn-svnserve-tunnel"
# KEY="$1"
# REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

################################################################
# For Gitea Docker:
################################################################
SHADOW_SHELL_PATH="/app/gitea/gitea"
TUNNEL_PATH="/app/git-as-svn/git-as-svn-svnserve-tunnel"
KEY="$2"
SUBCOMMAND="$1"
REAL_SHELL_PATH="$SHADOW_SHELL_PATH"

if [ "$SUBCOMMAND" != "serv" ]; then
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi
```

```
#################################################################
# Other options:
#################################################################
# For either, you can move the shadowed binary to something like
# /app/gitea/gitea.shadow and rename this script to /app/gitea/gitea.
#
# If you follow his approach you do not need to rewrite the
# authorized_keys file, but may still need to process it.
#
# You would need to set the REAL_SHELL_PATH to point to this file
# and restore the shadowing on updates to the application
#################################################################
SECRET="CHANGE_THIS_TO_SOMETHING_SECRET"

SSH_ORIGINAL_COMMANDS=($SSH_ORIGINAL_COMMAND)

if [ -n "$SSH_ORIGINAL_COMMAND" ] && [ "${SSH_ORIGINAL_COMMANDS[0]}" = "svnserve" ] ;
then
    ## TUNNEL TO OUR SVNSERVER WITH MAGIC AUTHENTICATION ##
    exec "$TUNNEL_PATH" "$SECRET" "$KEY"
else
    exec -a "$REAL_SHELL_PATH" "$SHADOW_SHELL_PATH" "$@"
fi
```

`/app/git-as-svn/bin/git-as-svn-svnserve-tunnel` should be the same as in the gitlab case.

Managing the `AuthorizedKeysCommand` is similar to that in the Gitlab case.

Chapter 13. Internal implementation details

13.1. Where Subversion data is stored?

To represent Subversion repository need to store information about how Subversion-revision number corresponds to which Git-commit. We can't compute this information every time on startup, because first `git push --force` change revision order. This information stored persistent in git reference `refs/git-as-svn/*`. In particular because it does not require a separate backup Subversion data. Because of this separate backup Subversion data is not necessary.

Also part of the data necessary for the Subversion repository, is very expensive to get based on Git repository.

For example:

- the revision number with the previous change file;
- information about where the file was copied;
- MD5 file hash.

In order not to find out their every startup, the data is cached in files. The loss of the cache is not critical for the operation and backup does not make sense.

File locking information is currently stored in the cache file.

13.2. Описание хранения информации в хранилище

13.3. Для чего нужно хранилище

Хранилище данных о ревизиях нужно для:

- Сохранения информации об списке измененных файлов (в svn очень часто нужно выяснять, в какой из предыдущих ревизий поменялся файл);
- Сохранения информации о том, к какой ревизии/ветке относится коммит;
- Сохранения идентификатора svn-репозитория.

Так же побочным эффектом служит формирование дерева с svn layout-ом для более простой работы svn update/switch/log/diff и т.п.

13.3.1. Формат хранения информации о ревизиях

Для хранения информации о ревизиях используется ссылка `refs/git-as-svn/v1`.

Эта ссылка содержит в себе набор коммитов, выстроенных в цеполчку по первому родителю. Порядковый номер коммита от начала соответвует номеру ревизии в SVN.

Содержимое коммита

Данные самого коммита:

- Первый родитель всегда ссылается на предыдущий коммит;
- Комментарий коммита берется из оригинального коммита. Именно этот комментарий выводится в svn log;
- Автор коммита берется из оригинального коммита;

Дерево коммита:

- svn (tree) - дерево, которое соответсвует svn layout-у данного коммита;
- commit.ref (commit) - ссылка на оригинальный коммит (для коммитов, которые связаны с удалением/созданием веток может отсутствовать);

Revision 0

Ревизия 0 несколько отличается от остальных коммитов.

В отличие от остальных коммитов:

- Она всегда содержит путое дерево svn;
- В ней лежит файл `uuid` с идентификатором репозитория.

13.4. How does commit work?

One of the most important parts of the system — to save the changes.

In general, the following algorithm:

1. At the moment the command `svn commit` client sends to the server of your changes. The server remembers them. At this point comes the first check the relevance of customer data.
2. The server takes the branch HEAD and begins to create new commit on the basis of client received delta. At this moment there is yet another check of the relevance of customer data.
3. Validating svn properties for changed data.
4. The server tries to push the new commit in the current branch of the same repository via console Git client. Next, the result of a push:
    - if commits pushed successfully — loading the latest changes from git commits and rejoice;
    - if push is not fast forward — load the latest changes from git commits and go to step 2;
    - if push declined by hooks — inform the client;
    - on another error — inform the client;

Thus, through the use console Git client for push, we avoid the race condition pouring directly change Git repository, and get the native hooks as a nice bonus.

Chapter 14. Changelog

Unreleased

- Systemd unit now correctly waits for git-as-svn to shut down. #275

1.21.9

- Catastrophically speedup rename detection (~50x). #306

1.21.8

- Write empty LFS files in a compatible with Git-LFS way
- Update dependencies

1.21.7

- Fix Git LFS lock paths not handled properly, making it possible to lock same file multiple times
- Send human-readable error message when locking fails due to already existing lock

1.21.6

- Add cleanup of bogus locks created with git-as-svn versions prior to 1.21.5

1.21.5

- Multiple fixes to remote LFS locking

1.21.4

- Fix commit of files larger than 8MB

1.21.3

- Fixes to `lfsMode: !fileLfs`.

1.21.2

- Fix bogus slashes in branch names for GitLab mapping

1.21.1

- Reduce log spam (LDAP and client disconnects)

- Log client version on connect

1.21.0

- Do not write to `/tmp` when streaming files from remote LFS server to SVN clients. #288
- Experimental `lfsMode: !fileLfs` LFS mode for GitLab
- `lfs: false` replaced with `lfsMode: null` in `!gitlab` section

1.20.5

- Log all exceptions when talking to SVN clients
- Fixed double buffering of client I/O
- Fix downloading of large files from remote LFS server. Broken in 1.20.4

1.20.4

- Fix multiple file descriptor leaks

1.20.3

- Fix `svn blame` failing with "Malformed network data" error

1.20.2

- Fix LFS files returning -1 size for remote LFS. #282

1.20.1

- Fix `git lfs unlock <path>` not finding LFS lock

1.20.0

- Fix inability to unlock files through Git-LFS
- Fix lock paths having leading slash when listing locks via Git-LFS
- Now path-based authorization supports branch-specific access

1.19.3

- Add `$authenticated:Local`/`$authenticated:GitLab`/`$authenticated:Gitea`/`$authenticated:LDAP` to refer to users authenticated against specific user database in path-based ACL
- Fix git-lfs failing with "Not Acceptable" error when uploading files

1.19.2

- Improve GitLab configuration defaults

1.19.1

- Fix path-based ACL entry search. #276

1.19.0

- Add support for LZ4 compression. `compressionEnabled=true/false` option replaced with `compressionLevel=LZ4/Zlib/None`. #163
- Fix severe performance loss on commit. Broken in 1.8.0

1.18.0

- Add option to expose user-defined branches for GitLab. See GitLab configuration documentation. #188
- `repositoryTags` is no longer supported for `!gitlabMapping`

1.17.0

- Drop ability to configure custom hook names in `!pushEmbedded` because Git doesn't have such feature. Instead, add `hooksPath` option that works as an override to `core.hooksPath` Git configuration option.
- Fix uploads of already existing files to remote LFS server

1.16.0

- Update Jetty to 9.4.19
- Update Log4j to 2.12.0
- Update git-lfs-java to 0.13.3
- Add support for `core.hooksPath` Git configuration variable. #267

1.15.0

- Now groups can be defined to contain other groups for path-based authorization
- JGit updated to 5.4.0
- UnboundID LDAP SDK updated to 4.0.11
- google-oauth-client updated to 1.30.1
- Remove `hookUrl` from `!gitlab` section, it is now automatically determined from `baseUrl` in `!web` section.

1.14.0

- Experimental path-based authorization
- `-t` and `-T` command-line switches. See Command-line parameters documentation
- `-s`/`--show-config` command-line switches removed. Use `-T` instead.

1.13.0

- Changed LDAP bind configuration. See LDAP documentation.
- Organize logs into categories and add logging documentation.

1.12.0

- Experimental support for LFS locking API Now git-as-svn forwards locking requests to LFS server. git-as-svn internal LFS server now supports LFS locks. Locks are now scoped to whole repositories instead of being per-branch. All existing svn locks will expire after upgrade.
- URL scheme has changed, now it is `svn://<host>/<repo>/<branch>`. Use `svn relocate` to fix existing SVN working copies.
- It is no longer valid to map a single repository under multiple paths. Use `branches` tag to expose multiple branches of a single repository to SVN.

1.11.1

- `!giteaSSHKeys` is no longer supported
- Fix date formatting to be compatible with git-lfs. Was broken in 1.11.0

1.11.0

- Add support for Gitea LFS server. Gitea >= 1.7.2 is required now.
- `!gitlabLfs {}` was replaced with `lfs: true` parameter in `!gitlab` section

1.10.1

- Fix PLAIN auth not working with passwords longer than 51 character. #242

1.10.0

- File locking code cleanup. All existing svn locks will expire after upgrade.
- Implement `get-file-revs` command. This is expected to speed up `svn blame` severely. #231
- Prospective blame support added

1.9.0

- Major code cleanup
- `repository: !git` changed to just `repository:` in git-as-svn.conf
- `access: !acl` changed to just `acl:` in git-as-svn.conf
- `svn stat` is now compatible with native svn for nonexistent paths

1.8.1

- Update dependencies: jgit-5.3.0, svnkit-1.10.0, jetty-9.4.15, java-gitea-api-1.7.4, unboundid-ldapsdk-4.0.10 and others

1.8.0

- `!lfs` renamed to `!localLfs` in git-as-svn.conf
- Experimental support for GitLab LFS (`!gitlabLfs {}`). #175, #212, #213.

1.7.6.1

- Fix broken URL construction in git-lfs-authenticate

1.7.6

- git-lfs-authenticate no longer silently falls back to anonymous mode if it failed to obtain user token
- git-lfs-authenticate now properly handles absolute repository paths

1.7.5

- Ensure hook stdout is closed when using embedded pusher

1.7.4

- Revert #215, causes tens of thousands of CLOSE_WAIT connections in Jetty
- Update Jetty to 9.4.14

1.7.3

- Reduce number of threads by using same thread pool for svn:// and http://. #215
- Fix compatibility with latest Gitea. #218

1.7.2

- Reduce lock contention during commit
- Log how long commit hooks take
- Do not log exception stacktraces on client-side issues during commit

1.7.1

- Revert offloading file → changed revisions cache to MapDB (PR#207) as an attempt to fix (or, at least, reduce) issues with non-heap memory leaks

1.7.0

- Dramatically improve memory usage by offloading file → changed revisions cache to MapDB
- --unsafe option no longer exists, all "unsafe" functionality was removed
- git-lfs-authenticate.cfg format has changed. Now, git-lfs-authenticate talks to git-as-svn via http and uses shared token.
- !api no longer exists in git-as-svn.conf
- !socket no longer exists in git-as-svn.conf
- LFS storage is no longer silently created, instead LfsFilter will error out when encounters LFS pointer without configured LFS storage
- JGit updated to 5.1.2
- GitLab API updated to 4.1.0

1.6.2

- [Gitea] Support uppercase letters in usernames / repository names. #196

1.6.1

- Update dependencies. #190
- [Gitea] Fixes to directory watcher. #192
- Deploy Debian packages to Bintray. #194

1.6.0

- Java 9/10/11 compatibility
- Gitea integration added

1.5.0

- Add tag-based repository filtering for GitLab integration

1.4.0

- Update JGit to 5.0.1.201806211838-r
- Update SVNKit to 1.9.3
- Reduce memory usage
- Improve indexing performance

1.3.0

- Switch to GitLab API v4. Fixes compatibility with GitLab >= 11. #176

1.2.0

- x10 speedup of LDAP authentication
- Drop dependency on GSon in favor of Jackson2
- Update unboundid-ldapsdk to 4.0.3
- Fix post-receive hook failing on GitLab 10 #160

1.1.9

- Update MapDB to 3.0.5 #161

1.1.8

- Fix git-as-svn unable to find prefix-mapped repositories (broken in 1.1.2)
- Fix PLAIN authentication with native SVN client (broken in 1.1.4)

1.1.7

- Use OAuth2 to obtain user token. Fixes compatibility with GitLab >= 10.2 #154

1.1.6

- Update various third-party libraries
- Upgrade to Gradle 4.4
- Fix GitLab repositories not becoming ready on git-as-svn startup #151
- Improve logging on git-as-svn startup

1.1.5

- Fix submodules support (was broken in 1.1.3)
- Invalidate caches properly if renameDetection setting was changed

1.1.4

- Upgrade Kryo to 4.0.1 #121
- Add option to disable parallel repository indexing on startup #121

1.1.3

- Fix ISO 8601 date formatting.
- Fix unexpected error message on locked file update #127.
- Increase default token expire time to one hour (3600 sec).
- Add string-suffix parameter for git-lfs-authenticate script.
- Index repositories using multiple threads on startup #132

1.1.2

- Add reference to original commit as parent for prevent commit removing by `git gc` #118.
- Fix repository mapping error #122.
- Fix non ThreadSafe Kryo usage #121.
- Add support for combine multiple authenticators.
- Add support for authenticator cache.
- Fix tree conflict on Windows after renaming file with same name in another case #123.
- Use commit author instead of commiter identity in svn log.
- Don’t allow almost expired tokens for LFS pointer requests.

1.1.1

- Fix "E210002: Network connection closed unexpectedly" on client update failure #114.

1.1.0

- Use by default svn:eol-style = native for text files (fix #106).
- Upload .deb package to debian repository.

1.0.17-alpha

- Add PDF, EPUB manual.
- Add support for anonymous authentication for public repositories.

1.0.16-alpha

- Rewrite GitLab authentication #110.
- Fix some permission check issues #110.
- Generate token in LFS server instead pass original authentication data #105.
- Ignore unknown GitLab hook data.

1.0.15-alpha

- Add support for GitLab 8.2 LFS storage layout #109.

1.0.14-alpha

- Add debian packaging.
- Add configurable file logging.

1.0.13-alpha

- Embedded git-lfs server
- Git-lfs batch API support.
- Add support for LDAP users without email.
- Add support for X-Forwarded-* headers.
- Add HTTP-requests logging.
- Change .gitignore mapping: ignored folder now mask all content as ignored.
- Fix git-lfs file commit.
- Fix quote parsing for .tgitconfig file.

1.0.12-alpha

- Initial git-lfs support (embedded git-lfs server).
- Initial GitLab integration.
- Import project list on startup.
- Authentication.
- Add support for embedded git push with hooks;

- Git-as-svn change information moved outside git repostitory #60.
- Configuration format changed.
- Fixed some wildcard issues.

1.0.11-alpha

- Fix URL in authentication result on default port (Jenkins error: `E21005: Impossibly long repository root from server`).
- Fix bind on already used port with flag SO_REUSEADDR (thanks for @fcharlie, #70).
- Add support for custom certificate for ldaps authentication.

1.0.10-alpha

- Fix get file size performance issue (`svn ls`).
- Fix update IMMEDIATES to INFINITY bug.
- Fix NPE on absent email in LDAP.

1.0.9-alpha

- Fix svn update after aborted update/checkout.
- Fix out-of-memory when update/checkout big directory.
- Show version number on startup.

1.0.8-alpha

- Support commands: `svn lock`/`svn unlock`.
- Multiple repositories support.

1.0.7-alpha

- More simple demonstration run
- `svnsync` support

1.0.6-alpha

- Add autodetection binary files (now file has `svn:mime-type = application/octet-stream` if it set as binary in .gitattribues or detected as binary).
- Expose committer email to svn.
- Fix getSize() for submodules.
- Fix temporary file lifetime.

1.0.5-alpha

- Add persistent cache support.
- Dumb locks support.
- Fix copy-from permission issue.

1.0.4-alpha

- Improve error message when commit is rejected due to wrong properties.

1.0.3-alpha

- Fix spaces in url.
- Add support get-locations.
- Add mapping binary to `svn:mime-type = svn:mime-type`

1.0.2-alpha

- Fix some critical bugs.

1.0.1-alpha

- Add support for more subversion commands
- Fix some bugs.

1.0.0-alpha

- First release.