Migrating

GitHub importer

For Internet-accessible projects, GitHub.com provides Importer for automatic migration and repository creation from Subversion, Team Foundation Server, Mercurial, or alternatively-hosted Git version controlled projects.

The process is as simple, needing only you to sign into your GitHub account, if you aren't already, entering your existing project's version control URL in the repository field, and initiating the conversion.

Depending on the detected version control system, Importer may request additional information for migration. This includes a mapping file for associating Subversion usernames with Git fields.

Read more about how to import your project into GitHub by looking at the full GitHub Importer documentation.

SVN2Git Utility

When access limitations or non-public Subversion repositories need porting to Git, the SVN2Git utility is the command line utility of choice and provides control through every step of the process.

Subversion presents distinct differences in structure to that of a Git repository, and SVN2Git provides the flexibility and configuration for traditional and custom Subversion layouts. This ensures the resulting Git repository aligns with standard best practices for commits, branches, and tags for the entire project's history.

Notable features of SVN2Git include:

- Converting all SVN conventions to traditional Git structure - Providing SVN users field to name and email data in Git commits
- Permitting exclusion patterns for precise Git repository content

Learn more about SVN2Git at the project's official home page:

https://github.com/nirvdrum/svn2git

Bridging

Leveraging Git’s support of SVN

Often times, during a transition to Git, the Subversion infrastructure remains in place while users become acquainted with local Git repository interactions, local workflows, and desktop applications.

The `git svn` command permits users to synchronize with a centralized Subversion repository host while taking advantage of all the benefits local Git command line and graphical clients have to offer.

To acquire a Subversion repository as a resulting local Git repository, download the project in its entirety with this command:

```
git svn clone [svn-repo-url] --stdlayout
```

Make certain you are familiar with the targeted Subversion repository’s structure and whether it follows the standard layout or not. For non-traditional `trunk`, `branches`, and `tags` layouts, the following option switches should be specified during the `svn clone`:

- `T [trunk]` for alternate main source convention
- `b [branches]` for alternate branch location
- `t [tags]` for alternate tag structure location

Once the clone operation completes, you can proceed with any local Git interactions on the command line or with graphical clients.

Synchronizing with Subversion

Publishing local Git history back to a central Subversion repository acquired with git svn clone is performed with one command:

```
git svn dcommit
```

If the hosted Subversion repository’s history possesses commits not yet in the local Git repository, the `dcommit` operation will be rejected until the commits are acquired with this command:

```
git svn rebase
```

Keep in mind this action rewrites your local Git history and your commit identifiers will be different.

Understanding

Subversion and Git share similar vocabularies, but the commonality often is only is command names. Behavior and functionality are quite distinct given the unique qualities Git provides as a distributed version control system when compared to the centralized aspects of Subversion.

SVN command	Git command	Git behavior
`status`	`status`	Report the state of working tree
`add`	`add`	Required for each path before making a commit
`commit`	`commit`	Store prepared changes in local revision history
`rm` , `delete`	`rm`	Prepare paths for deletion in next commit
`move`	`mv`	Prepare relocated content for next commit
`checkout`	`clone`	Acquire the entire history of a project locally for the first time
	`branch`	Create local context for commits
	`merge`	Join branch histories and changes to working tree
	`log`	No network required
	`push`	Upload commit history to GitHub/centralized Git host
	`pull`	Download and integrate GitHub repository history with local on
	`fetch`	Download GitHub repository history with no other action