Linux Academy

Git Quick Start

Cheat Sheet

Stosh Oldham stosh@linuxacademy.com

December 12, 2018

Contents

Git Introduction and Architecture 1

Git Installation and Configuration 1

Creating a Repository and Adding Content 2

Ignoring Content 2

Cloning 2

Branching 3

Logging 3

Merging and Pushing Updates 4

Git Introduction and Architecture

- The `git` program is a source control tool created by Linus Torvalds.
- Simply stated, `git` manages content snapshots, checksums, and metadata to keep track of changes in files.
- Some basic terminology (we will go into more detail as we work through!):
  - Each state is known as a commit.
  - The current commit is called the head.
  - Commits are affiliated with repositories and branches.
  - The head may be moved between commits.
- This course covers the basics to get you up and running with Git.
- More detailed information as well as more advanced topics can be found in Linux Academy's *Source Control with Git* course.

Git Installation and Configuration

- To install Git:
  - `sudo yum install git` or `sudo apt-get install git`
- Git configuration may be managed using `git config`:
  - `git config --global user.name <username>`
  - `git config --global user.email <email_addr>`
  - `git config --system core.editor vim`
- Alternatively, you may store system-wide configuration values in the file directly:
  - `/etc/gitconfig` which corresponds to `--system`
  - `~/.gitconfig` or `~/.config/git/config` which corresponds to `--global`
  - `.git/config` in a repository which corresponds to `--local`
  - **Note**: Files lower in the list override files higher in the list.
  - File Example:

```
[user]
 name = john smith
 email = john@example.com
```

Creating a Repository and Adding Content

- Use `git init <repo-directory>` to create or "initialize" a new repository in the specified directory.
  - The init subcommand will create a `.git` directory in the repository used for Git metadata (see Git Deep Dive course for more information).
- `git add <filename>` may be used to indicate relevant files for tracking in the repository.
- Tracking information on files within the directory may be obtained using `git status`.
- Once all relevant changes have been noted, run `git commit -m "text describing changes"` to commit the changes to the repository.
- If a file no longer requires tracking, the command `git rm <file>` may be used to stop tracking a provided file.

Ignoring Content

- Sometimes there are artifacts in your project which are transient or do not otherwise need to be tracked in source control.
- There is a mechanism to have certain files disregarded for source control purposes.
- The file `.gitignore` (local to a repository or system global) may contain a list of file names or patterns which Git will ignore.
- It is also possible to exclude certain paths using:
  - `git config --global core.exludesfile <path>`

Cloning

- Typically, the repository of record is not used for "working" purposes.
- Changes to source are managed either in a separate branch or a remote repository.
- Git repositories may be cloned or have branches.
- It is possible to clone a repository locally on the file system using:
  - `git clone <path_of_original> <clone_repo_location>`
- Git also supports cloning over SSH and HTTP/S.
  - Clone over SSH:
    - `git clone user@server:<original_repo_path_relative_to_user_home> <local_repo_path>`
  - Clone over HTTPS:

        * `git clone https://github.com/username/reponame <local_repo_path>`

- The cloned repository keeps repository logs separately making the origin repository log less cluttered.
- After local work is complete, the changes may be pushed back to the origin repository using:
    - `git push origin master`
- Branching achieves a similar effect within a single repository (more information in branching section).

Branching

- Branching allows you to create an effective copy of the master branch within the repository that can be worked on without interfering with the master.
- This declutters the master branch.
- Branching tends to be an expensive operation in other source control tools but it is incredibly efficient in Git.
- Branches can be merged back into the master branch when work is complete.
- How to:
    - Create a branch:
        * `git branch <new branch>`
    - Begin working in the new branch:
        * `git checkout <branch>`
    - Do both at once:
        * `git checkout -b <new branch>`
- It is also possible to branch from an existing branch.

Logging

- `git log` will print information regarding commits and changes within the repository.
- An abbreviated change log can be listed using:
    - `git log --oneline`
- More detailed log information may be viewed with:
    - `git log -p`
- It is also possible to look at information on a particular file:

  - `git log -- <filename>`
  - `git log --oneline <filename>`

- To have a graph printed, you can use the `--graph` and `--decorate` options:
  - `git log --graph --decorate`
  - **Note**: This is particularity useful when you have multiple branches.

Merging and Pushing Updates

- Branches may be pushed to remote sources, just like master.
- How to push branches to origin:
  - `git push origin --all`
- Merging brings branches together (must be in the branch you are merging into):
  - `git merge <target branch>`
- Merging branches where files may have changed in diverging ways is called a **merge conflict** (see the Git Deep Dive for more on this).