High Available OpenNMS

Alejandro Galue, agalue@opennms.com

1.0-SNAPSHOT
Last updated 2015-11-27 23:33:08 +01:00

Table of Contents

GIT CHEAT SHEET 1
- 1. Install Git 2
- 2. Configure Tooling 3
- 3. Create Repositories 4
- 4. Make Changes 5
- 5. Group Changes 6
- 6. Refactoring Filenames 7
- 7. Suppress Tracking 8
- 8. Save Fragments 9
- 9. Review History 10
- 10. Redo Commits 11
- 11. Synchronize Changes 12

GIT CHEAT SHEET

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

Chapter 1. Install Git

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

https://windows.github.com

https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM site.

http://git-scm.com

Chapter 2. Configure Tooling

Configure user information for all local repositories.

Sets the name you want attached to your commit transactions

```
$ git config --global user.name "[name]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Enables helpful colorization of command line output

```
$ git config --global color.ui auto
```

Chapter 3. Create Repositories

Start a new repository or obtain one from an existing URL.

Creates a new local repository with the specified name

```
$ git init [project-name]
```

Downloads a project and its entire version history

```
$ git clone [url]
```

Chapter 4. Make Changes

Review edits and craft a commit transaction.

Lists all new or modified files to be committed

```
$ git status
```

Shows file differences not yet staged

```
$ git diff
```

Snapshots the file in preparation for versioning

```
$ git add [file]
```

Shows file differences between staging and the last file version

```
$ git diff --staged
```

Unstages the file, but preserve its contents

```
$ git reset [file]
```

Records file snapshots permanently in version history

```
$ git commit -m "[descriptive message]"
```

Chapter 5. Group Changes

Name a series of commits and combine completed efforts.

Lists all local branches in the current repository

```
$ git branch
```

Creates a new branch

```
$ git branch [branch-name]
```

Switches to the specified branch and updates the working directory

```
$ git checkout [branch-name]
```

Combines the specified branch’s history into the current branch

```
$ git merge [branch]
```

Deletes the specified branch

```
$ git branch -d [branch-name]
```

Chapter 6. Refactoring Filenames

Relocate and remove versioned files

Deletes the file from the working directory and stages the deletion

```
$ git rm [file]
```

Removes the file from version control but preserves the file locally

```
$ git rm --cached [file]
```

Changes the file name and prepares it for commit

```
$ git mv [file-original] [file-renamed]
```

Chapter 7. Suppress Tracking

Exclude temporary files and paths.

A text file named .gitignore suppresses accidental versioning of files and paths matching the specific patterns.

```
*.log
build/
temp-*
```

Lists all ignored files in this project

```
$ git ls-files --other --ignored --exclude-standard
```

Chapter 8. Save Fragments

Shelve and restore incomplete changes.

Temporarily stores all modified tracked files

```
$ git stash
```

Restores the most recently stashed files

```
$ git stash pop
```

Lists all stashed changesets

```
$ git stash list
```

Discards the most recently stashed changeset

```
$ git stash drop
```

Chapter 9. Review History

Browse and inspect the evolution of project files.

Lists version history for the current branch

```
$ git log
```

Lists version history for a file, including renames

```
$ git log --follow [file]
```

Shows content differences between two branches

```
$ git diff [first-branch]...[second-branch]
```

Outputs metadata and content changes of the specified commit

```
$ git show [commit]
```

Chapter 10. Redo Commits

Erase mistakes and craft replacement history

Undoes all commits after `[commit]`, preserving changes locally

```
$ git reset [commit]
```

Discards all history and changes back to the specified commit

```
$ git reset --hard [commit]
```

Chapter 11. Synchronize Changes

Register a repository bookmark and exchange version history.

Downloads all history from the repository bookmark

```
$ git fetch [bookmark]
```

Combines bookmark’s branch into current local branch

```
$ git merge [bookmark]/[branch]
```

Uploads all local branch commits to GitHub

```
$ git push [alias] [branch]
```

Downloads bookmark history and incorporates changes

```
$ git pull
```