Git Cheat Sheet

Git is the open source distributed version control system C3 IoT strongly recommends using. This cheat sheet summarizes commonly used Git commands.

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config -- global user.name “[name]”
```

Sets the name you want attached to your commits

```
$ git config --global user.email “[email]”
```

Sets the email you want attached to your commits

CREATE REPOSITORIES

Start a new repo or obtain one from an existing URL

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

Creates a new local repository with the specified name

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

Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit

```
$ git status
```

Lists all new or modified files to be committed

```
$ git diff
```

Shows file differences not yet staged

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

Snapshots the file in preparation for versioning

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

Shows differences between staging and last file version

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

Un-stages the file, but preserves its contents

```
$ git commit -m “[message]”
```

Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine efforts.

```
$ git branch
```

Lists all local branches in the current repository

```
$ git branch -r
```

Lists all remote branches

```
$ git branch [new_branch_name]
```

Creates a new branch

```
$ git checkout -b [new_branch_name]
```

Creates a new branch

```
$ git checkout [branch_name]
```

Switches to chosen branch and updates working directory

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

Combines chosen branch’s history into current branch

```
$ git branch -v
```

Shows the last commit on each branch

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

Deletes the chosen branch

SAVE FRAGMENTS

Shelve and restore incomplete changes

```
$ git stash
```

Temporarily stores all modified tracked files

```
$ git stash pop
```

Restores the most recently stashed files

```
$ git stash list
```

Lists all stashed changesets

```
$ git stash drop
```

Discards the most recently stashed changeset

REDO COMMITS

Start a new repo or obtain one from an existing URL

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

Undoes all commits after the commit, preserving changes locally

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

Discards all history and changes back to the specified commit

```
$ git reset HEAD [staged_file_name]
```

Un-stages specified file (or all if none specified), but does not delete or remove the file from the repository

```
$ git revert [commit]
```

Creates new commit that undoes all the changes made in this commit, and applies it to current branch

SYNCHRONIZE CHANGES

Browse and inspect the evolution of project files

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

Downloads all history from the repo bookmark

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

Combines bookmark's branch into current local branch

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

Uploads all local branch commits to GitHub

```
$ git pull
```

Downloads bookmark history and incorporates changes

REWRITING GIT HISTORY

Relocate and remove versioned files

```
$ git commit --amend
```

Replaces last commit with staged changes and latest commit combined

```
$ git rebase [base]
```

Rebase current branch onto [base]

```
$ git reflog
```

Shows log of changes to the local repo's HEAD

REFACTOR FILENAMES

Relocate and remove versioned files

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

Deletes the file from working directory and stages deletion

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

Removes file from version control but preserves file locally

```
$ git mv [file_originaly] [file_renamed]
```

Changes the file name and prepares it for commit

SUPPRESS TRACKING

Exclude temporary files and paths

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

A text file named `.gitignore` suppresses accidental versioning of files and paths matching specified patterns

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

Lists all ignored files in this project

REVIEW HISTORY

Browse and inspect the evolution of project files

```
$ git log
```

Lists version history for the current branch

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

Lists version history for a file, including renames

```
$ git diff [first_branch]…[second_branch]
```

Shows content differences between two branches

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

Outputs metadata and content changes of commit

Reference:

- https://services.github.com/on-demand/resources/cheatsheets/
- https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet