Optimize Your Development with This Ultimate

Git Cheatsheet

CLONING & INITIALIZATION

Initialize an existing directory as a Git repository

```
$ git init
```

Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits

```
$ git clone [ repo_url ]
```

After using the git init command, link the local repository to an empty GitHub repository using the following command:

```
$ git remote add origin [url]
```

SETUP CREDENTIALS

Set a Username

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

Set an Email address

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

BRANCHES

To check which branch that is

```
$ git branch
```

Creates a new branch

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

Creates a new branch & switch to that branch using one command only

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

Deletes the specified branch

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

STAGE & SNAPSHOT

Show modified files in working directory, staged for your next commit

```
$ git status
```

Add a file as it looks now to your next commit (stage)

```
$ git add [file_name]
```

Add all changed files to staging area

```
$ git add .
```


Commit your staged content as a new commit snapshot

```
$ git commit -m “[descriptive message]”
```

Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. git pull is a combination of git fetch and git merge

```
$ git pull origin <branch_name>
```

Uploads all local branch commits to GitHub

```
$ git push origin <branch_name>
```


Synchronize your local repository with the remote repository on GitHub.com

```
$ git fetch
```

MERGING

Merge <branch> into the current branch

```
$ git merge  <branch_name>
```

THE .GITIGNOREFILE

Sometimes it may be a good idea to exclude files from being tracked with Git. This is typically done in a special file named .gitignore.

REDO COMMITS

Reset staging area to match most recent commit, but leave the working directory unchanged.

```
$ git reset
```

Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.

```
$ git reset --hard
```

Create new commit that undoes all of the changes made in , then apply it to the current branch.

```
$ git revert <commit>
```

TEMPORARY COMMITS

Put current changes from your working directory into stash for later use.

```
$ git stash
```

Apply stored stash content into working directory, and clear stash.

```
$ git stash pop
```

Delete a specific stash from all your previous stashes

```
$ git stash drop
```

Was this post helpful to you?

Follow for more