#2 Install

#3 GitHub Desktop

desktop.github.com

#3 Git for All Platforms

git-scm.com

#2 Configure tooling

Configure user information for all local repositories

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

Sets the name you want attached to your commit transactions

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

Sets the email you want attached to your commit transactions

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

Enables helpful colorization of command line output

#2 Branches

Branches are an important part of working with Git. Any commits you make will be made on the branch you're currently "checked out" to. Use `git status` to see which branch that is.

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

Creates a new branch

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

Switches to the specified branch and updates the working directory

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

Combines the specified branch's history into the current branch. This is usually done in pull requests, but is an important Git operation.

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

Deletes the specified branch

#2 Create repositories

A new repository can either be created locally, or an existing repository can be cloned. When a repository was initialized locally, you have to push it to GitHub afterwards.

```
$ git init
```

The git init command turns an existing directory into a new Git repository inside the folder you are running this command. After using the `git init` command, link the local repository to an empty GitHub repository using the following command:

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

Specifies the remote repository for your local repository. The url points to a repository on GitHub.

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

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

#2 The .gitignore file

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`. You can find helpful templates for `.gitignore` files at github.com/github/gitignore.

#2 Synchronize changes

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

```
$ git fetch
```

Downloads all history from the remote tracking branches

```
$ git merge
```

Combines remote tracking branches into current local branch

```
$ git push
```

Uploads all local branch commits to GitHub

```
$ git pull
```

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`

#2 Make changes

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 the specified commit

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

Snapshots the file in preparation for versioning

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

Records file snapshots permanently in version history

#2 Redo commits

Erase mistakes and craft replacement history

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

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

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

Discards all history and changes back to the specified commit

> CAUTION! Changing history can have nasty side effects. If you need to change commits that exist on GitHub (the remote), proceed with caution. If you need help, reach out at github.community or contact support.

#2 Glossary

- **git**: an open source, distributed version-control system
- **GitHub**: a platform for hosting and collaborating on Git repositories
- **commit**: a Git object, a snapshot of your entire repository compressed into a SHA
- **branch**: a lightweight movable pointer to a commit
- **clone**: a local version of a repository, including all commits and branches
- **remote**: a common repository on GitHub that all team members use to exchange their changes
- **fork**: a copy of a repository on GitHub owned by a different user
- **pull request**: a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more
- **HEAD**: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`