Developer Tools

git

Distributed Version Control System

Purwadhika
Startup and Coding School

Git

Git is a *distributed version control system*:

- It’s a system that records changes to our files over time, so we can recally specific version of those files at any given time.
- Many people can easily collaborate on a project and have their own version of project files on their computer.

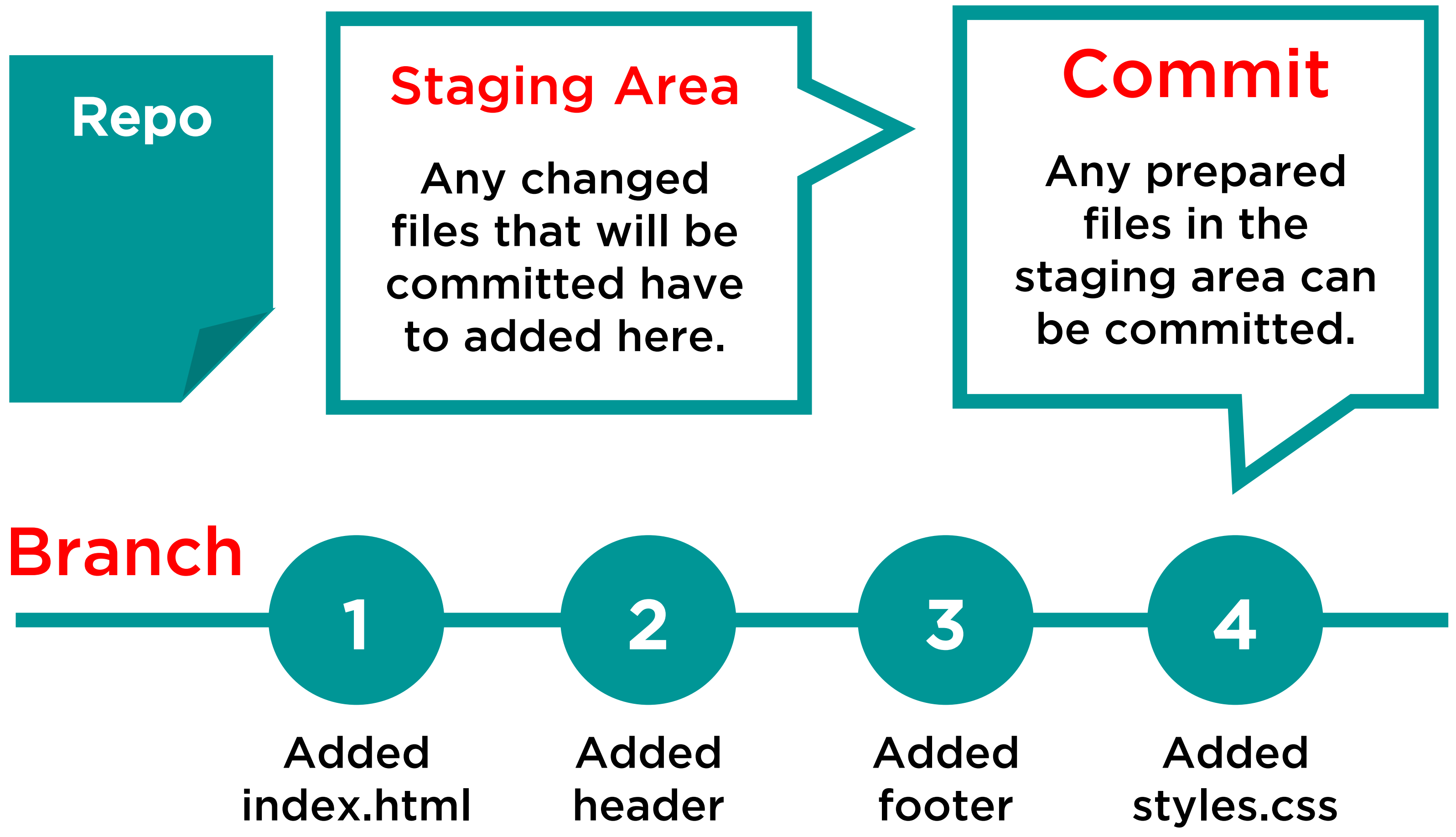
Purwadhika
Startup and Coding School

Why Use Git?

- Store revisions in a project history in just one directory.
- Rewind to any revision in the project we wanted to.
- Work on new features without messing up the main codebase.
- Easily collaborate with other programmers.

Purwadhika
Startup and Coding School

How to Use?



Repo

Staging Area

Any changed files that will be committed have to added here.

Commit

Any prepared files in the staging area can be committed.

Branch

1. Added index.html
2. Added header
3. Added footer
4. Added styles.css

Purwadhika
Startup and Coding School

Install Git

- Install Git from https://git-scm.com.

- Check git version

```
$ git --version
```

- Set username

```
$ git config --global user.name lintang
```

- Set email

```
$ git config --global user.email xyz@xyz
```

- Check username

```
$ git config user.name
```

Purwadhika
Startup and Coding School

Create Repo & Add to Staging

- On project directory (new or old), type:

```
$ git init
```

- Check file status

```
$ git status
```

- Add file to staging area

```
$ git add <namaFile.xyz>  // add just a file
$ git add .               // add all files
```

- Remove file from staging area

```
$ git rm --cached <namaFile.xyz>
```

Purwadhika
Startup and Coding School

Making Commits

- Making commit

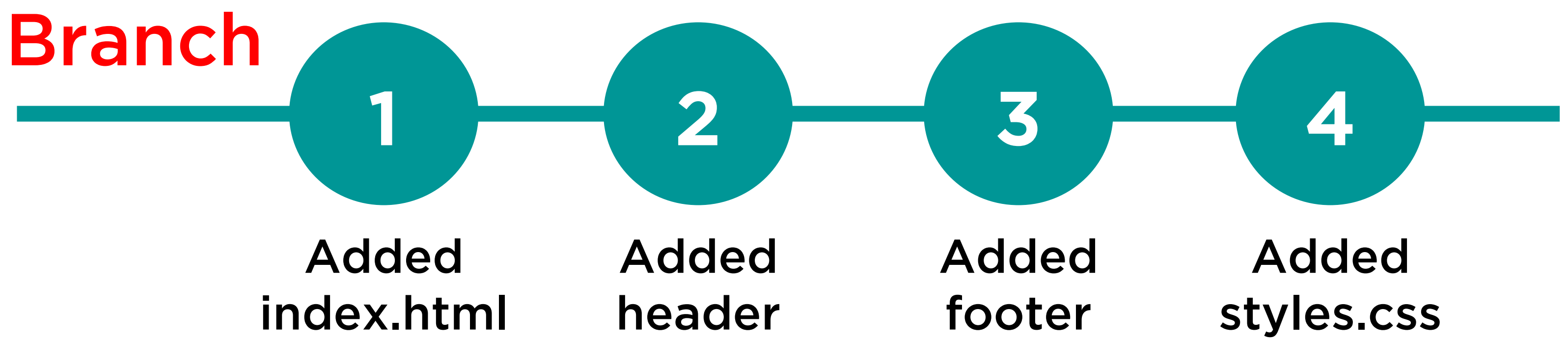
```
$ git commit
$ git commit –m “added css files”
```

- See commit history

```
$ git log
$ git log --oneline
```

Purwadhika
Startup and Coding School

Undo-ing Things



Checkout Commit

Revert Commit

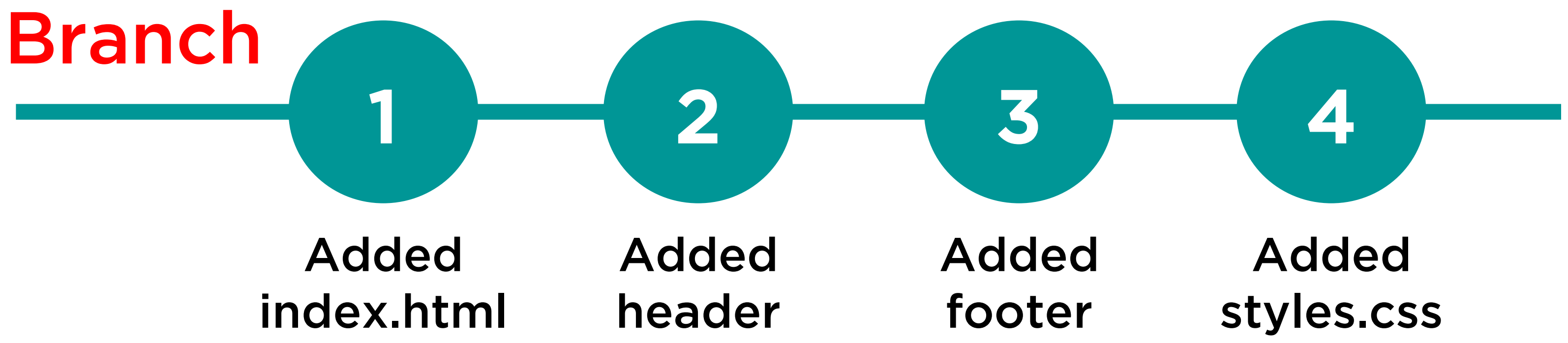
Reset Commit

Checkout Commit

- *Checkout a commit:*

```
$ git checkout <commit_id>  // checkout
$ git checkout master       // go back
```

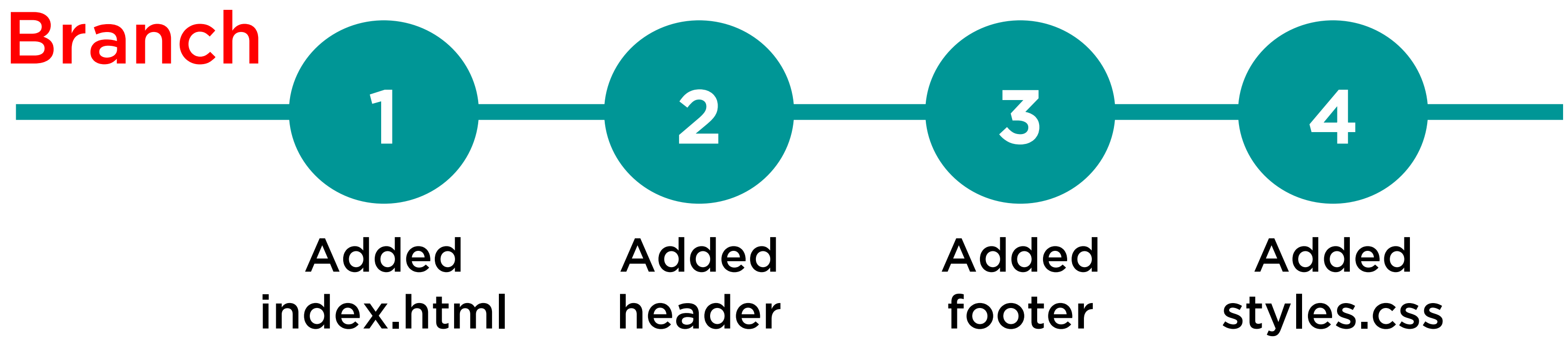
* It will rewind to <commit_id>, just to look around (read-only).



Revert Commit

- *Revert a commit (undo a particular commit):*

```
$ git revert <commit_id>
```



Reset Commit

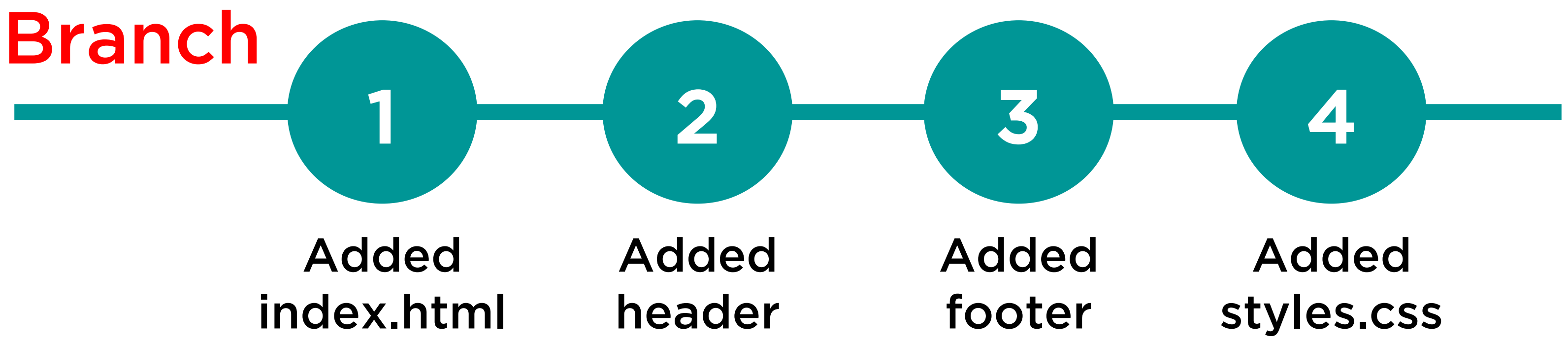
- *Reset a commit:*

```
$ git reset <commit_id>
```

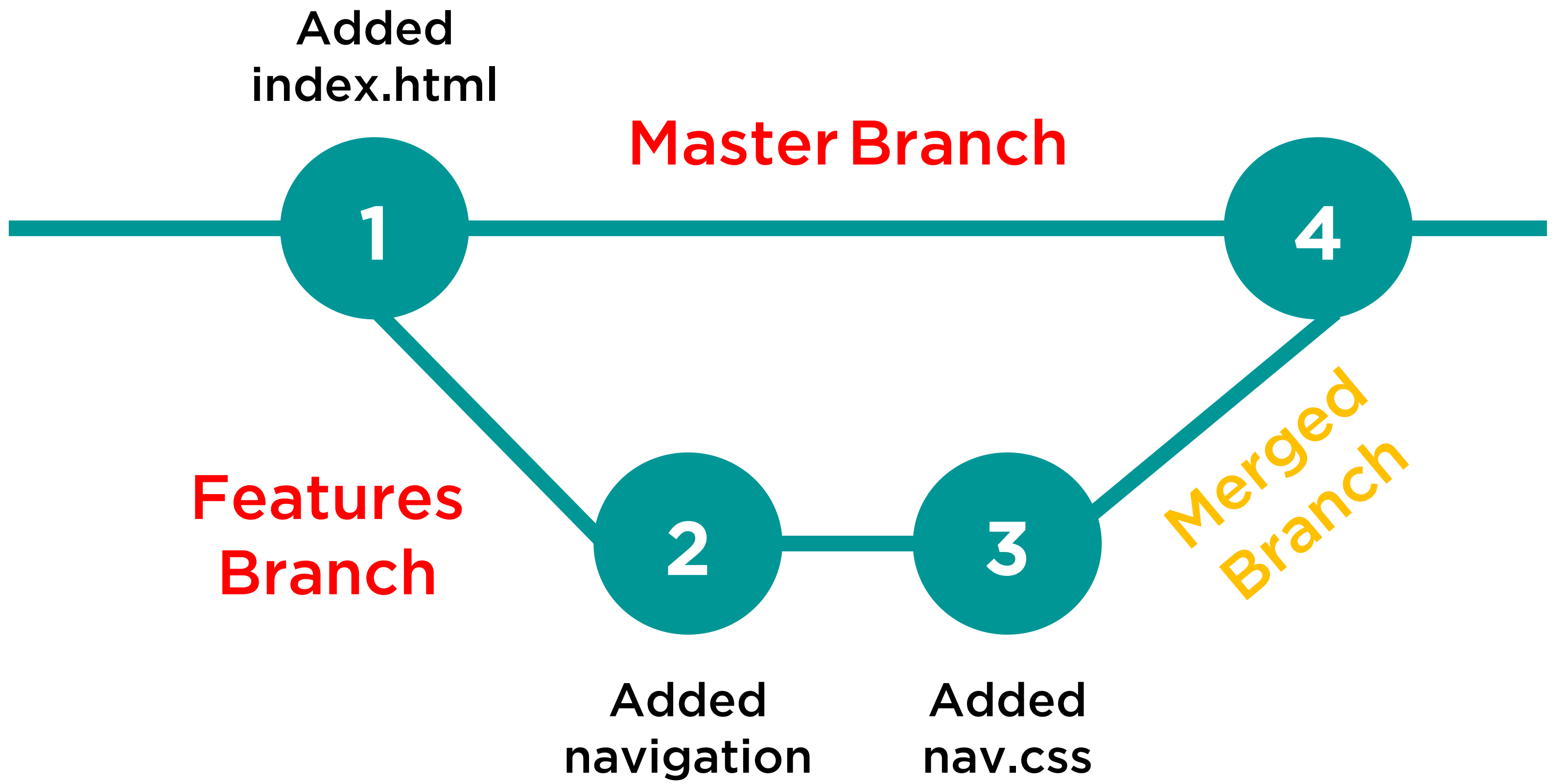
* It will reset to <commit_id>, delete commits after it, but the changes after it still there.

```
$ git reset <commit_id> --hard
```

* It will reset to <commit_id>, delete commits & changes after it.



Branches



Purwadhika
Startup and Coding School

Branches

- Making a feature branch:
```
$ git branch <namaBranch>
```
- See all branches (including master branch):
```
$ git branch –a
```
- Working on a branch:
```
$ git checkout <namaBranch>
```
- Shortcut to make & work on a branch:
```
$ git checkout -b <namaBranch>
```
- Delete a branch:

  1 - `$ git checkout master`

  2 - `$ git branch -D <namaBranch>`

Merging Branches

- Merge a feature branch to master branch:

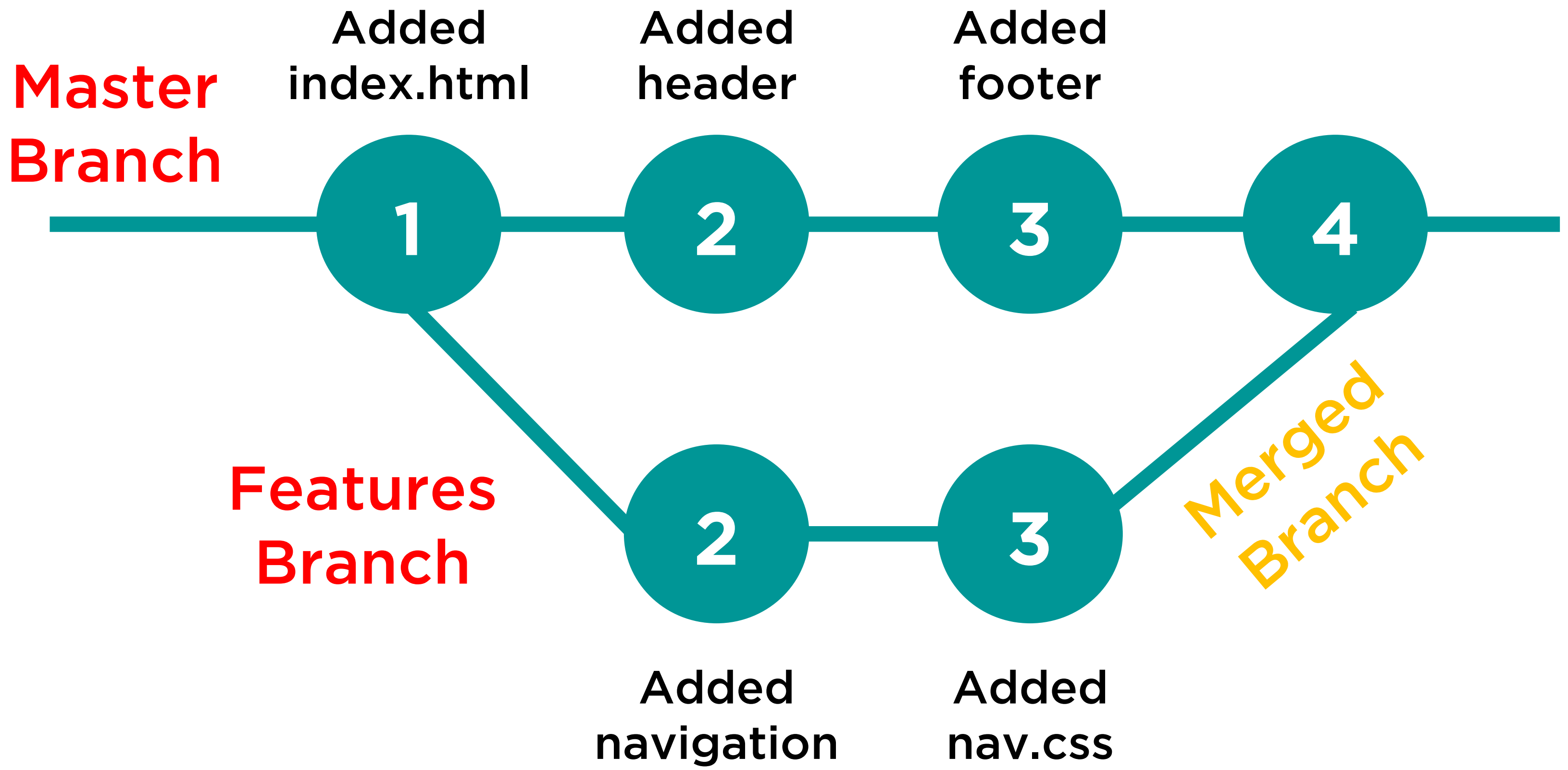
1 - `$ git checkout master`
2 - `$ git merge <namaBranch>`

Dealing with Conflicts

- If there's a conflict, edit files only from master branch. It's better to not work on master branch when the repo has branches.

Purwadhika
Startup and Coding School

Conflicts



Purwadhika
Startup and Coding School

Developer Tools

git

Distributed Version Control System

Purwadhika
Startup and Coding School