Intro to GIT

Install git and setup SSH key

Install git on Ubuntu:

```
$ sudo apt install git-all
```

Generating a new SSH key

1. Open Terminal.
2. Paste the text below, substituting in your GitHub email address.

   ```
   $ ssh-keygen -t ed25519 -C " your_email@example.com"
   ```

   Note: If you are using a legacy system that doesn't support the Ed25519 algorithm, use:

   ```
   $ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
   ```

   This creates a new ssh key, using the provided email as a label.

   ```
   > Generating public/private ed25519 key pair.
   ```

3. When you're prompted to "Enter a file in which to save the key," press Enter. This accepts the default file location.

   ```
   > Enter a file in which to save the key (/home/ you/.ssh/id_ed25519): [Press enter]
   ```

4. At the prompt, type a secure passphrase. For more information, see "Working with SSH key passphrases".

   ```
   > Enter passphrase (empty for no passphrase): [Type a passphrase]
   > Enter same passphrase again: [Type passphrase again]
   ```

1. Start the ssh-agent in the background.

   ```
   $ eval "$(ssh-agent -s)"
   > Agent pid 59566
   ```

2. Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace *id_ed25519* in the command with the name of your private key file.

   ```
   $ ssh-add ~/.ssh/id_ed25519
   ```

3. Add the SSH key to your GitHub account.

Configure your Git username/email

You typically configure your global username and email address after installing Git. However, you can do so now if you missed that step or want to make changes. After you set your global configuration, repository-specific configuration is optional.

Git configuration works the same across Windows, macOS, and Linux.

To set your global username/email configuration:

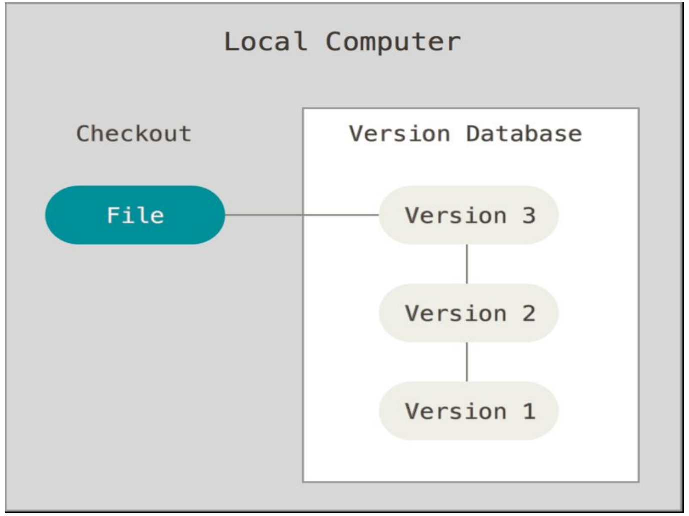
1. Open the command line.
2. Set your username:
   ```
   git config --global user.name "FIRST_NAME LAST_NAME"
   ```
3. Set your email address:
   ```
   git config --global user.email "MY_NAME@example.com"
   ```

To set repository-specific username/email configuration:

1. From the command line, change into the repository directory.
2. Set your username:
   ```
   git config user.name "FIRST_NAME LAST_NAME"
   ```
3. Set your email address:
   ```
   git config user.email "MY_NAME@example.com"
   ```
4. Verify your configuration by displaying your configuration file:
   ```
   cat .git/config
   ```

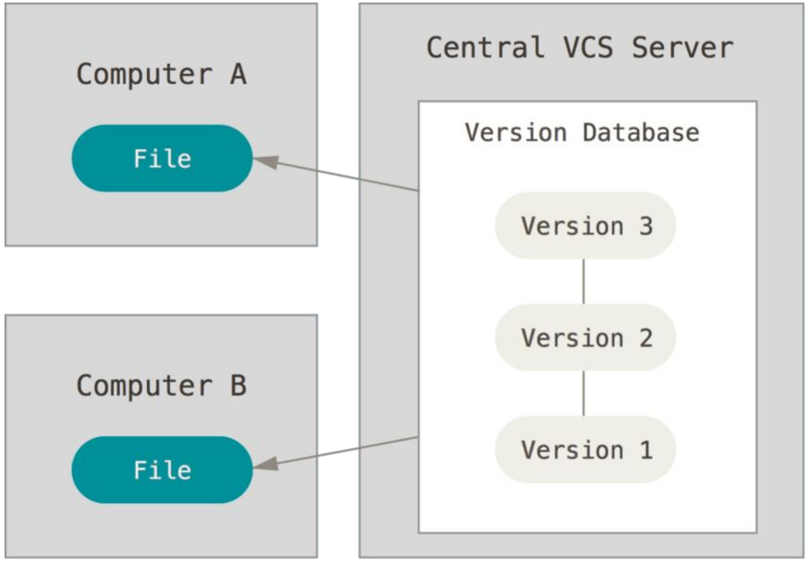
What is a Version Control System

Local version control systems:



+ Plain
- Makes cooperation impossible
- Prone to mistakes

Centralized version control systems:

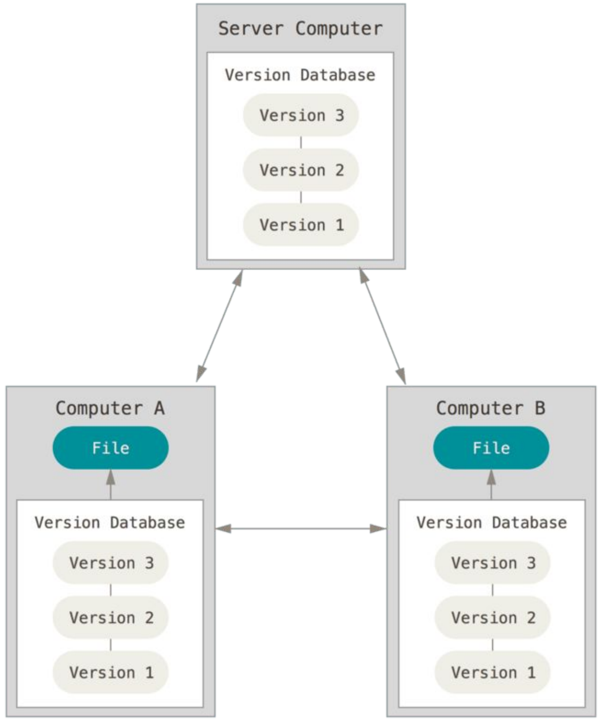


+ Administrable
+ Understandable
+ Transparent

- Lagging
- Internet connection required
- Unstable

What is a Version Control System

Distributed version control systems:



+ Very fast

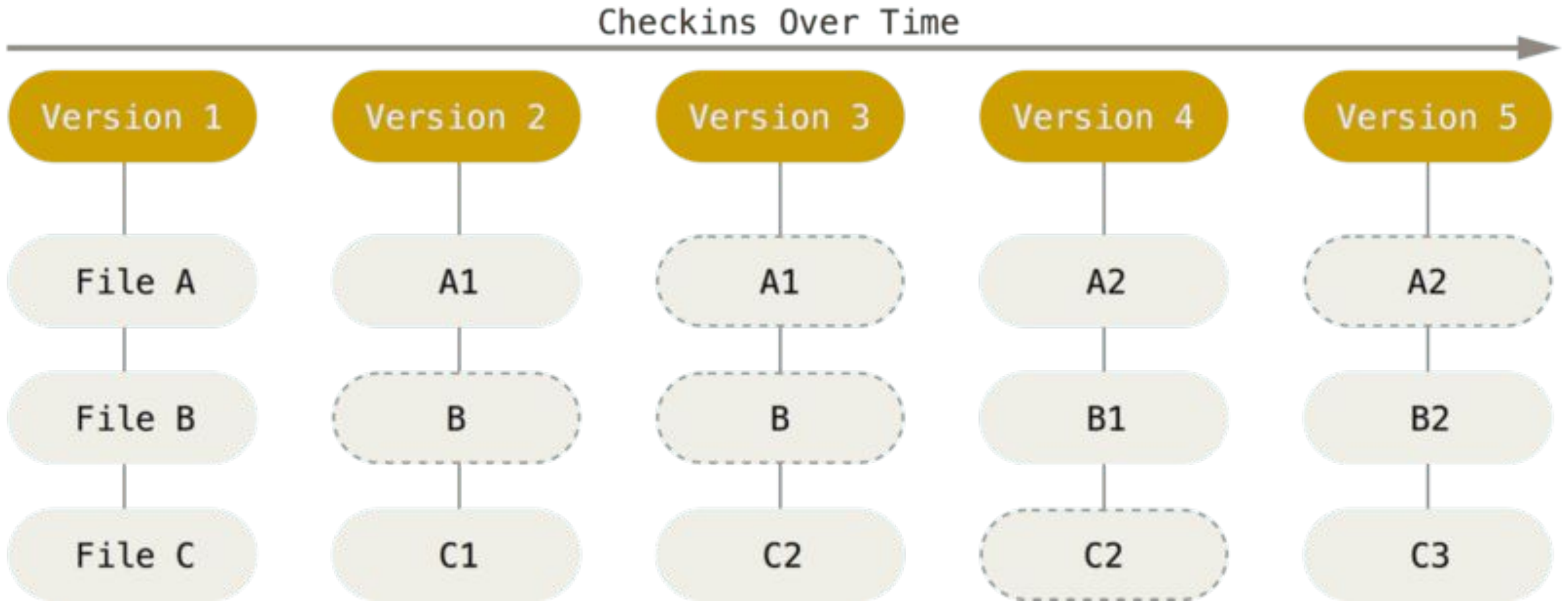
+ Most of the operations are local

+ It's hard to make a mistake or delete something

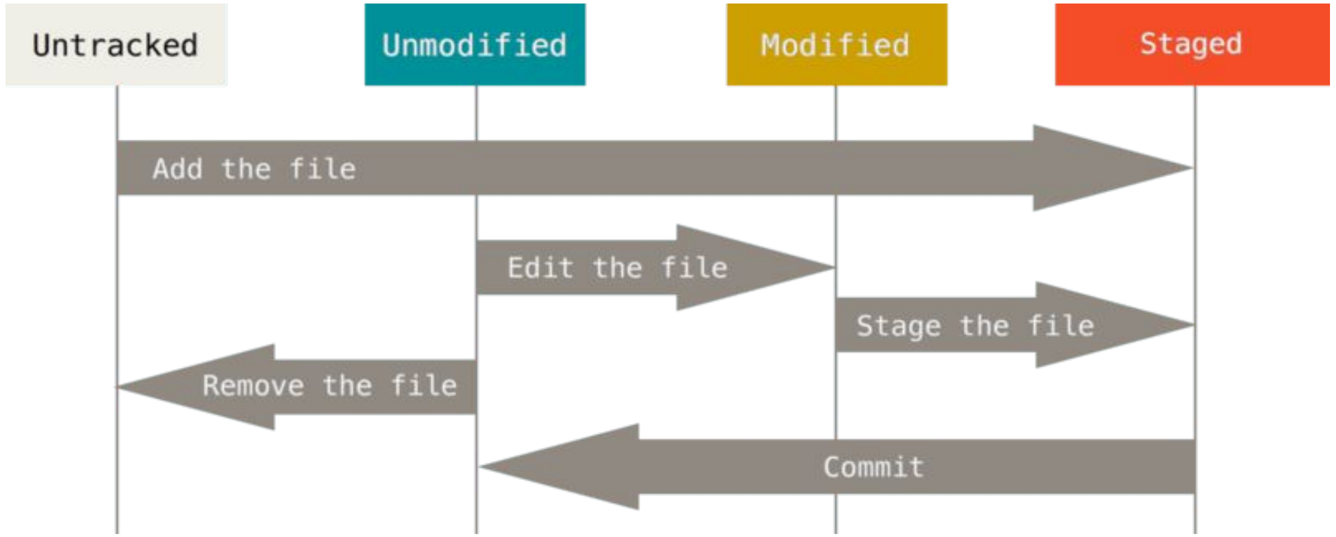
+ Multiple repositories.

- Difficult to understand

Git approach



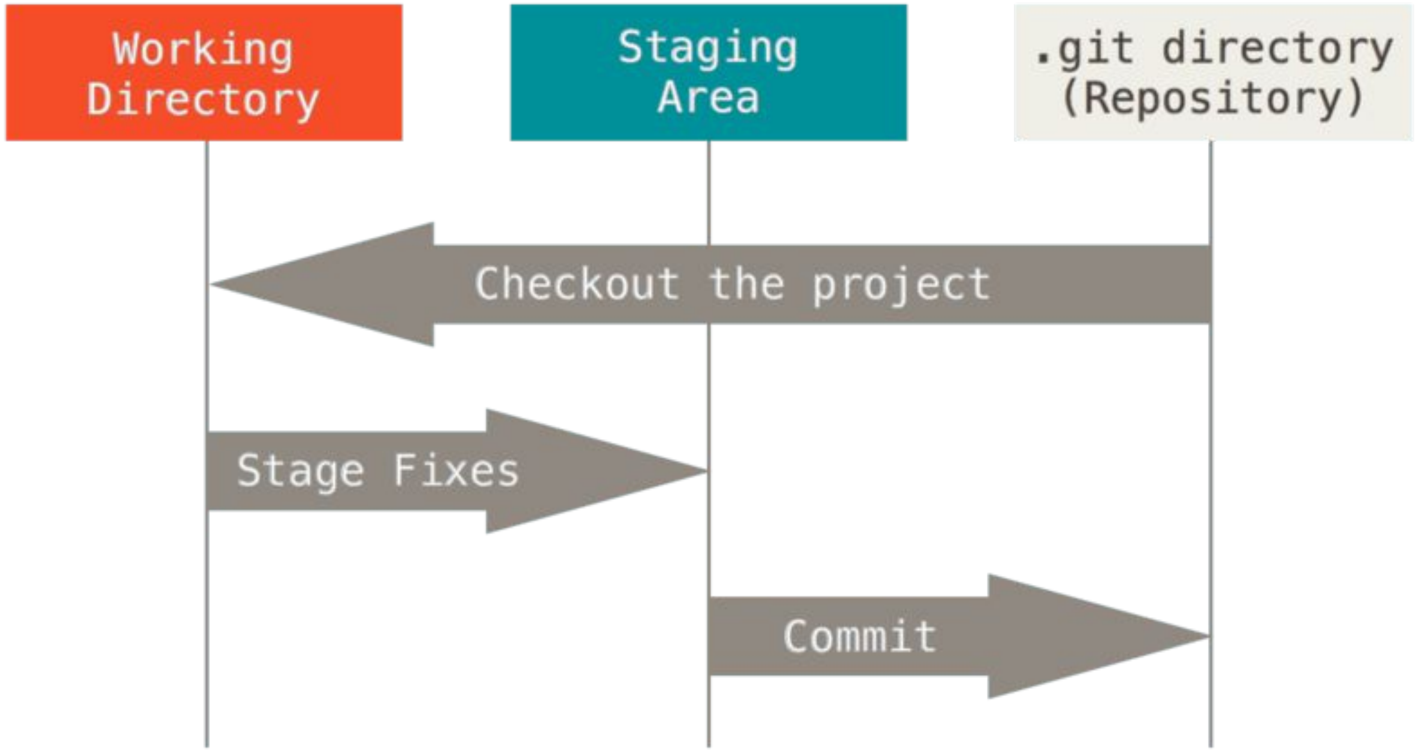
Git file states



File in Git can be either:

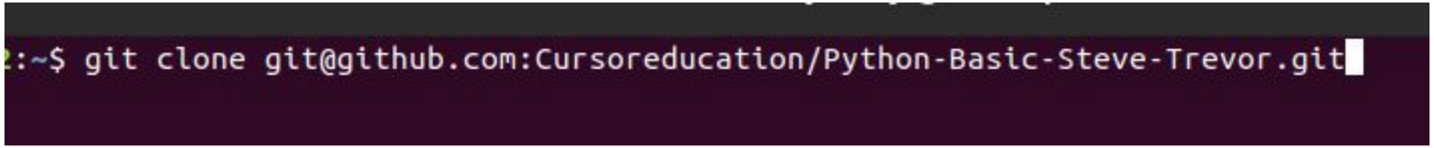
- Untracked – Git sees it in the repository but knows nothing about it
- Staged – It was added to the staging area with ‘git add’
- Unmodified – Normal unchanged file in the repository
- Modified(Deleted/Renamed) – File from the repository was modified.

Git Directories



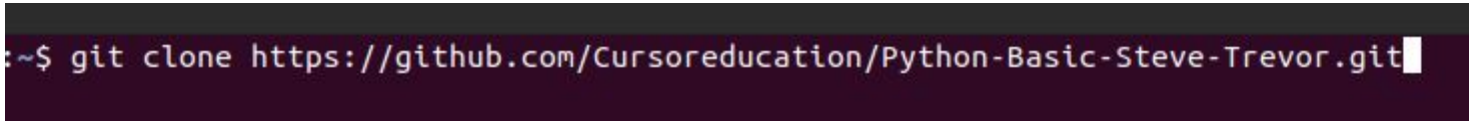
Clone project

Authorization via SSH key:



```
:~$ git clone git@github.com:Cursoreducation/Python-Basic-Steve-Trevor.git
```

Authorization via email and password:



```
:~$ git clone https://github.com/Cursoreducation/Python-Basic-Steve-Trevor.git
```

Common Git Commands

We will divide the common Git commands into two primary categories:

Local: git init, git touch, git status, git add, git commit, git rm ...

Remote: git remote, git pull, and git push ...

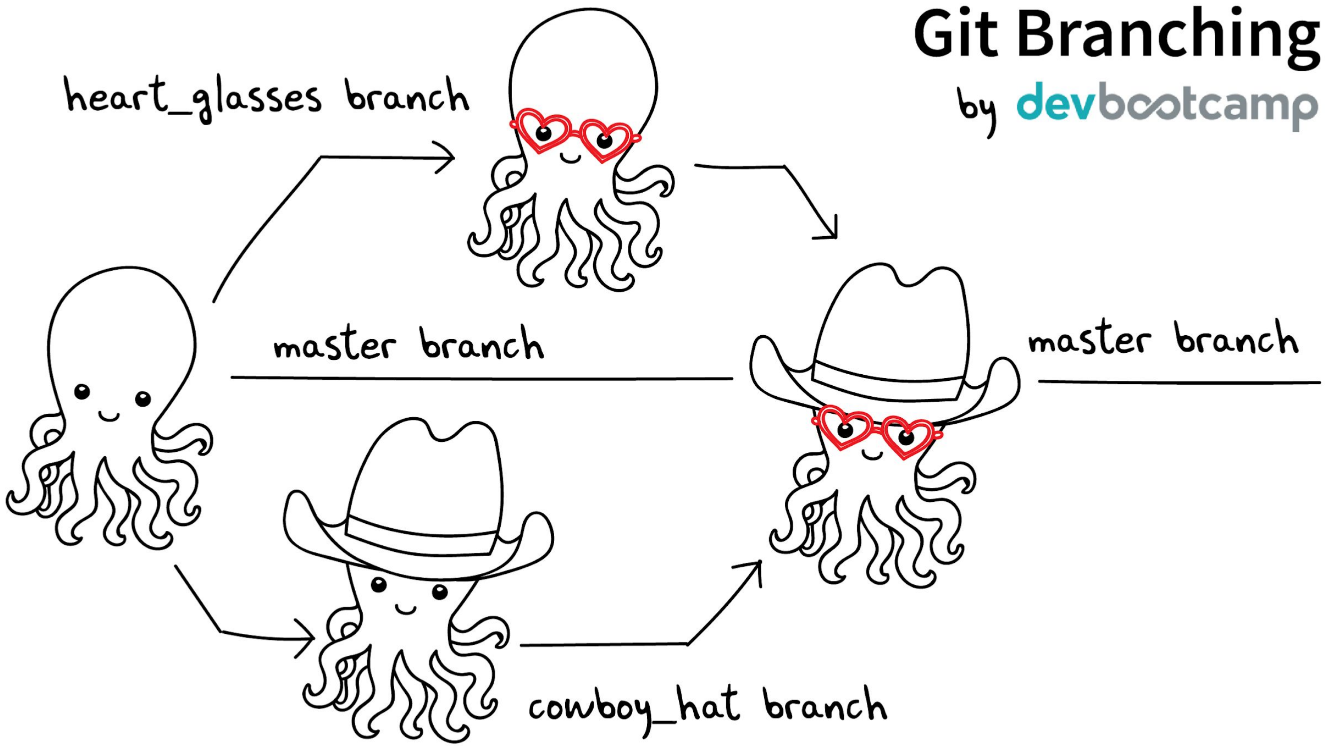
Git Basic Commands

git status – check the repository status
git add <file name> - add file to the staging area.
git commit –m "Add some changes" – record changes to the git directory
git log – check commit history
git push – push changes to remote
git pull – pull changes from remote
git fetch - download objects and refs from another repository
git rm <file name> - remove file from git
git grep <pattern> - print lines matching a pattern
git tag - Create, list, delete or verify a tag object signed with GPG
git stash - stash the changes in a dirty working directory away
git stash pop - Remove a single stashed state from the stash list and apply it on top of the current working tree state
git remote -v – check your remote git url

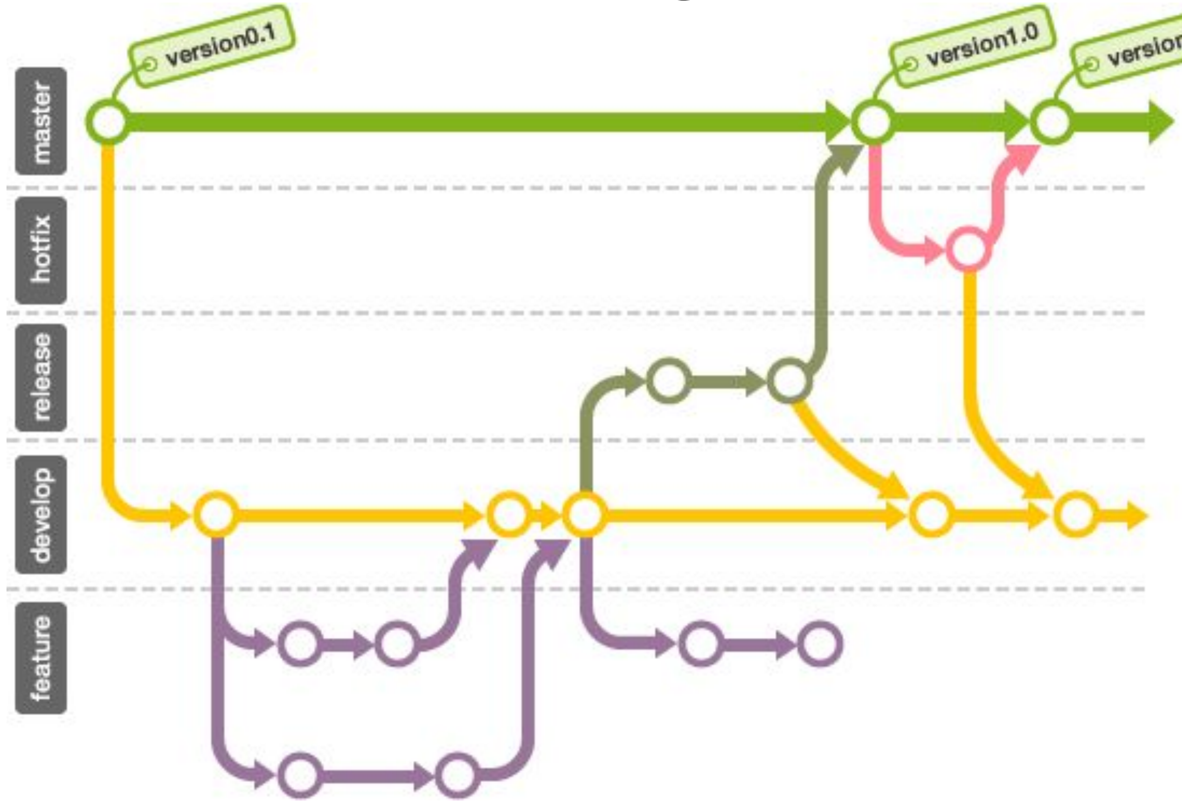
Git Basic Commands

git branch – get a list of all of the local git branches
git branch <branch name> – add new local git branch
git checkout <branch name> -- checkout to the <branch name>
git checkout -b <branch name> – created and checkout to the <branch name>
git merge <branch name> – merge changes from the <branch name> to current branch
git rebase <branch name> - re-committing your changes on top of the other branch
git cherry-pick <commit> - apply the changes introduced by some existing commits

Branches


Git Branching
by devbootcamp
heart_glasses branch
master branch
master branch
cowboy_hat branch

Branching model



In case of failure

git reset HEAD <file> – remove file from the index
git commit –-amend – fix the last commit without creating a new one
git reset {some-commit-hash} - reset to a specific commit, it means that all the changes were made after this commit will be untracked
--hard {some-commit-hash} – hard reset to a specific commit, it means that all the changes were made after this commit will be permanently removed
--soft {some-commit-hash} – soft reset to a specific commit, it means that all the changes were made after this commit will be in stage state
git push --force-with-lease - is a safer option that will not overwrite any work on the remote branch if more commits were added to the remote branch

References

https://learngitbranching.js.org/

Thank you for your attention!