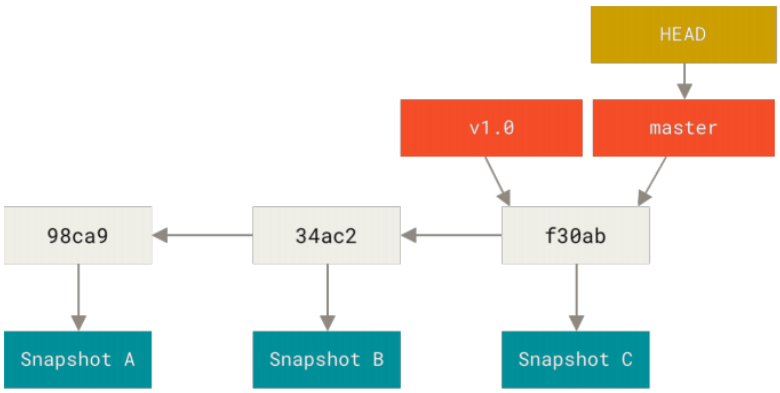
04. Git Branching

A branch in Git is simply a lightweight movable pointer to one of the commits.



The best practice is to create a **branch**, always. A **branch** is like a new space you create to make your developments, to not disturb the main road which is also called the **master** branch in GitLab and the **main** branch in GitHub. Think about the **master** branch as the place where everyone agreed that the work is of enough quality to be shared and cloned.

▼ Basics of Git Branching

- **Creating a new branch at the current commit —**

```
$ git branch <branch name>
```

- **List all of your branches —**

```
$ git branch
# the branch you are currently on will have * left to it
```

- **Switching between branches —**

```
# Switch to an existing branch
$ git switch <branch name>

# Create a new branch and switch to it
```

```
$ git switch -c <new-branch>
# Return to your previously checked-out branch
$ git switch -
```

While switching between branches remember that HEAD moves with each checkout/switch.

- **To see the last commit on each branch —**

```
$ git branch -v
```

- **Merging two branches —**

```
# merge the branch, 'branch-a' to the branch you are currently on
$ git merge <branch-a>

# If you wanted to merge 'branch-a' to the 'master' (or, main) branch
# first you would need to switch to the 'master' branch and then run this command
```

```
# List the branches that you have merged into the current branch
$ git branch --merged
```

```
# List the branches that you have not yet merged into the current branch
$ git branch --no-merged
```

- **Deleting a branch —**

  After you've merged two branches you would want to delete the branch that was merged as you no longer need it.

```
$ git branch -d <branch name>
```

▼ Working with Branches in a Remote Repository

Git doesn't push your branches when you push your local repository to a remote so you can work on private branches that you don't want to share yet.

```
# To share a branch to the remote
$ git push <remote name> <branch to push>
```

```
# To use a different name for the branch at your remote
$ git push <remote name> <branch to push>:<remote branch name>
```

```
# To delete a remote branch
$ git push <remote name> --delete <branch name on the remote>
```

▼ Basic Merge Conflicts

Merge conflicts happen when you've changed the same part of the same file differently in the two branches you're merging. Due to different changes to the same file git can't automatically decide which changes to keep and thus won't be able to merge them cleanly.

- **To see which files are unmerged at any point after a merge conflict —**

  ```
  git status
  ```

  This will show the file paths that have conflicts.

- **Resolving conflicts —**

  You can either manually edit the file to resolve the conflicts or use a GUI tool to solve the issues.

  ```
  # To launch a GUI
  $ git mergetool.
  ```

  ```
  # After you have resolved the conflicts you have to mark the files as resolved so that git knows
  # that the files are ready to merge.
  $ git add <file path>
  ```

  Note: If you use the `mergetool`, after you exit Git asks you if the merge was successful. If you tell the script that it was, it stages the file to mark it as resolved for you. You can run `$ git status` again to verify that all conflicts have been resolved

  ```
  # To finalize the merge do a commit
  $ git commit
  ```