Université Paris Cité

Git : Part 2 Branches & Merges

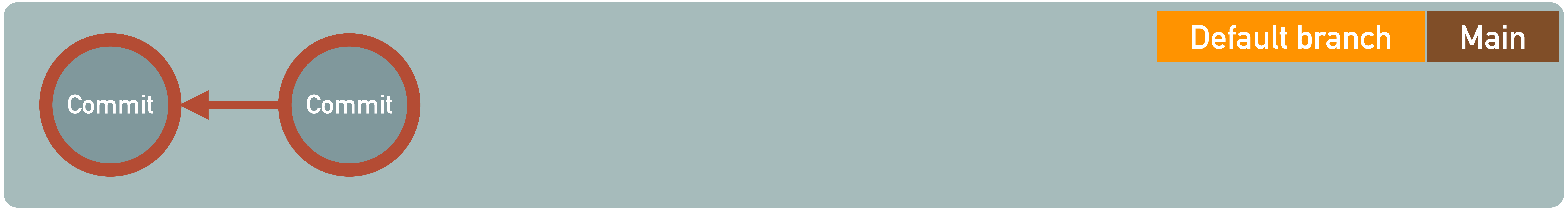
Amine FERDJAOUI

amine.ferdjaoui@sogeti.com

Année universitaire 2025/2026

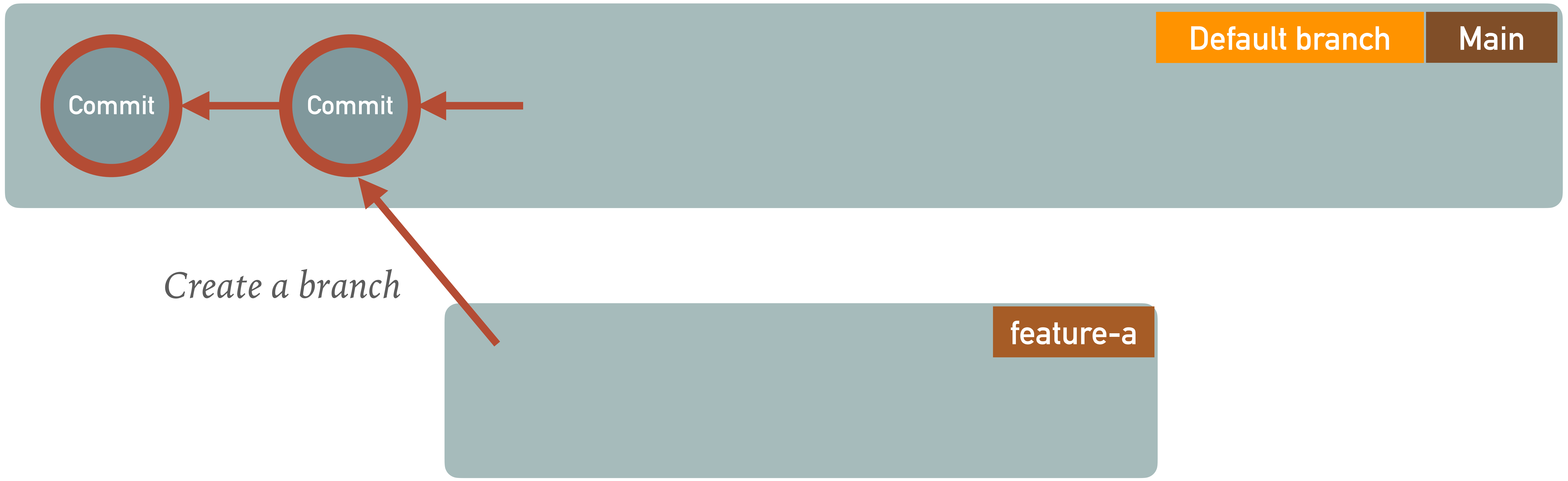
BRANCHES

Branches

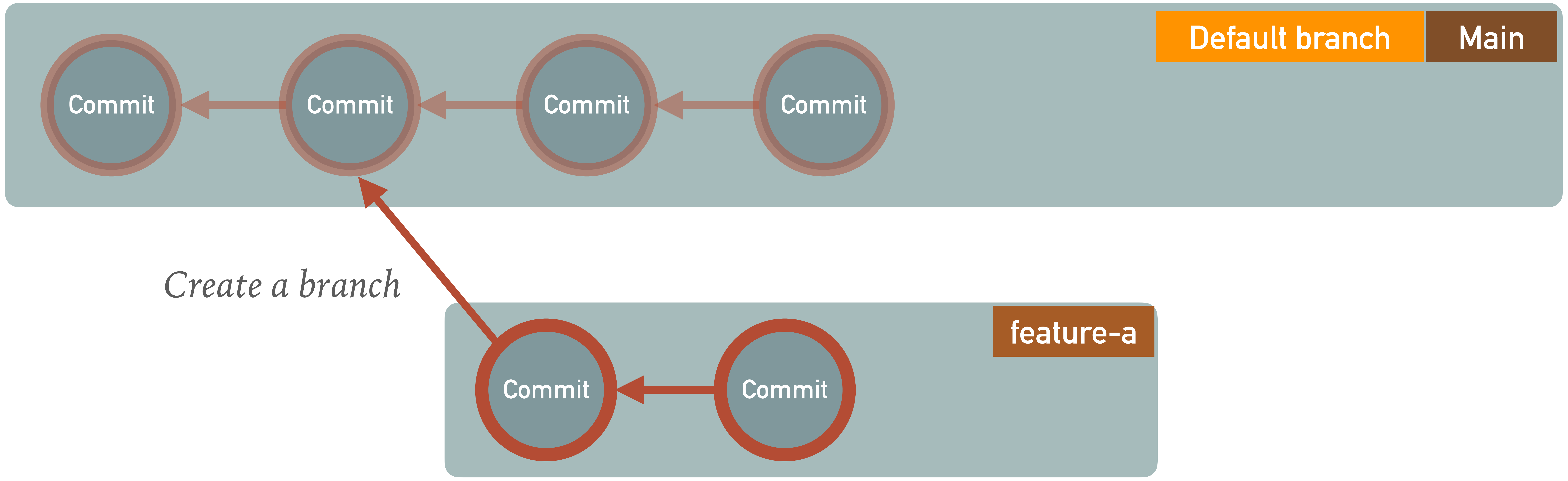


*in old git versions the default branch is called "*master*"

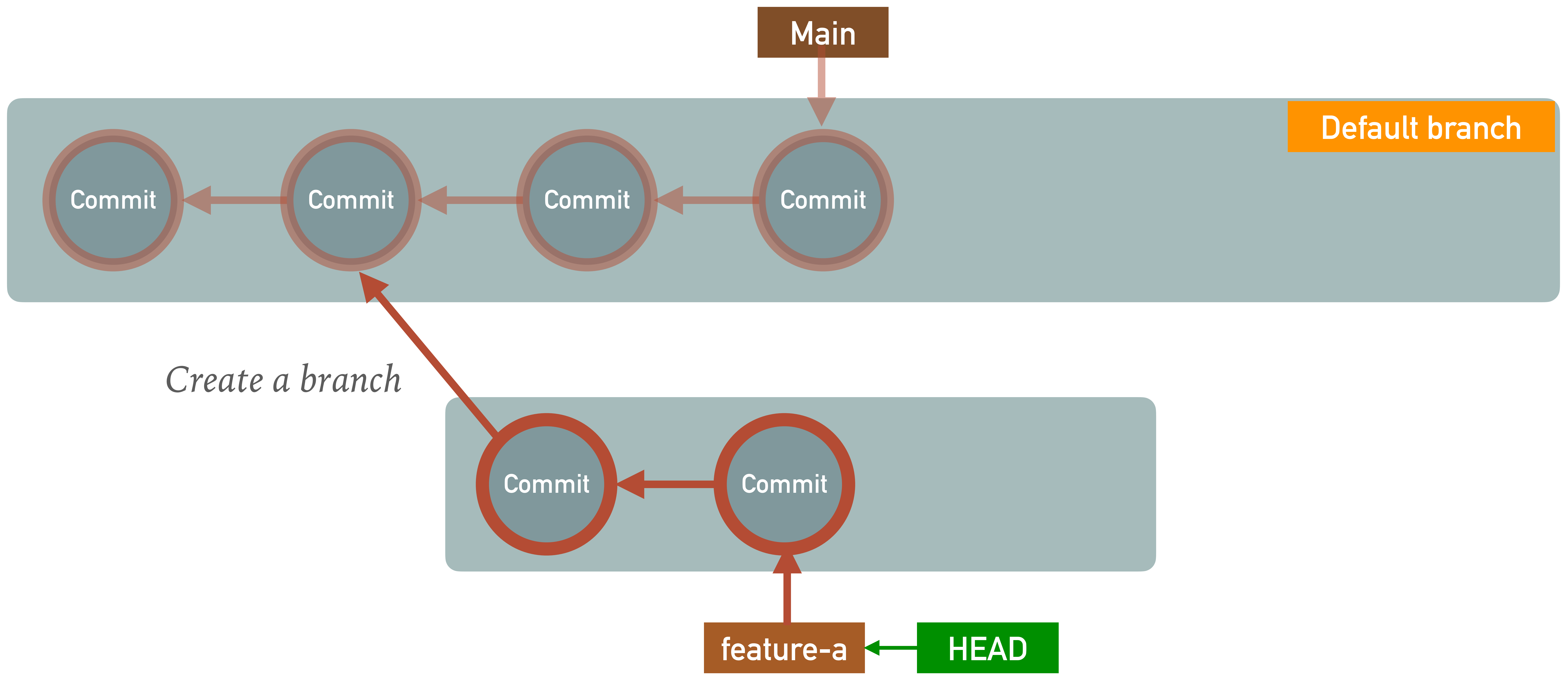
Branches


Default branch
Main
Commit
Commit
Create a branch
feature-a

Branches


Default branch
Main
Commit
Commit
Commit
Commit
Create a branch
feature-a
Commit
Commit

Branches



Git restores the working directory according to the position of the HEAD pointer

Git branches

Get the current branch

```
$ git branch
$ git branch -a #for all
```

Ps. Always commit before changing branch (otherwise use git stash et git stash apply)

Create new branch and switch

```
$ git branch <branch-name>
$ git switch <branch-name> #or git checkout
$ git switch - #go back to the last branch
$ git switch -C <branch-name> #create and jump into new branch / or git checkout -b
```

Delete a branch

```
$ git branch -d <branch-name> # to delete
```

Log & diff

Log commits of all branches

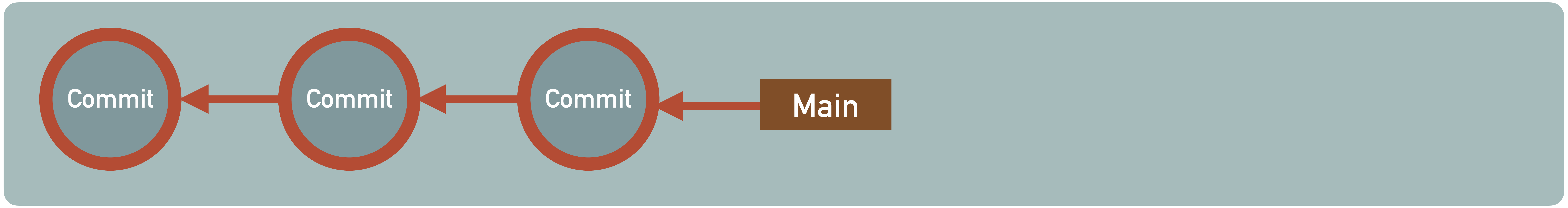
```
$ git log --oneline --all
$ git log --oneline --all --graph #you can save it as alias
$ git log <branch-A>..<branch-B> #commit difference between 2 branches
```

See differences between two branches

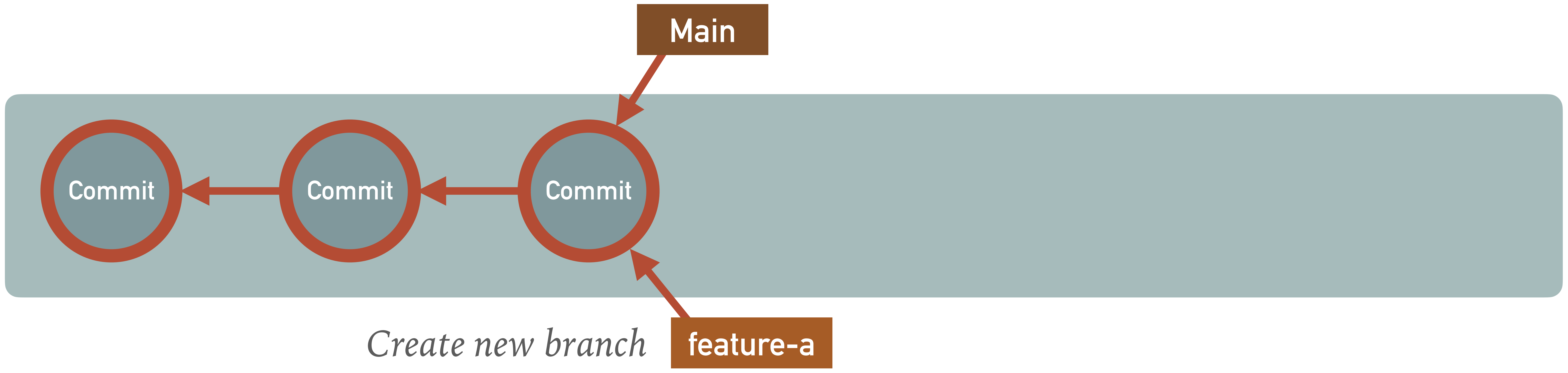
```
$ git diff <branch-A>..<branch-B> #incide file difference
$ git diff --name-status <branch-A>..<branch-B> #file difference
```

MERGE

Merges

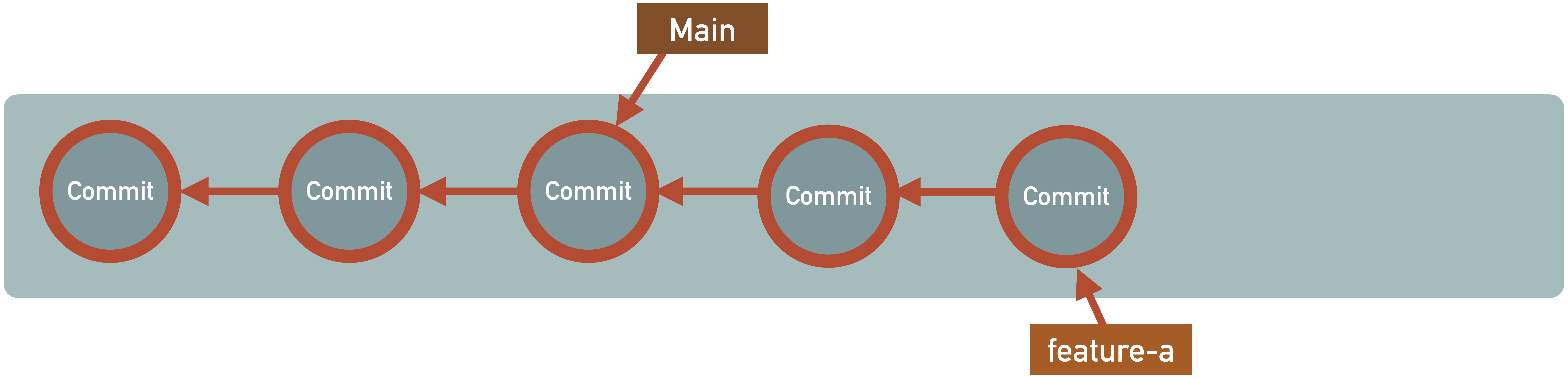


Merges

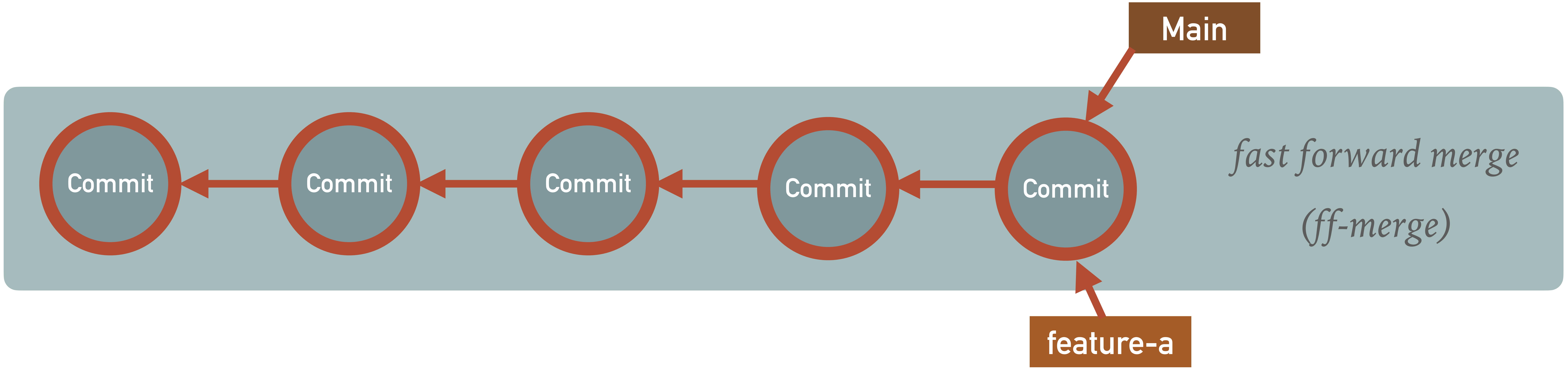


Create new branch

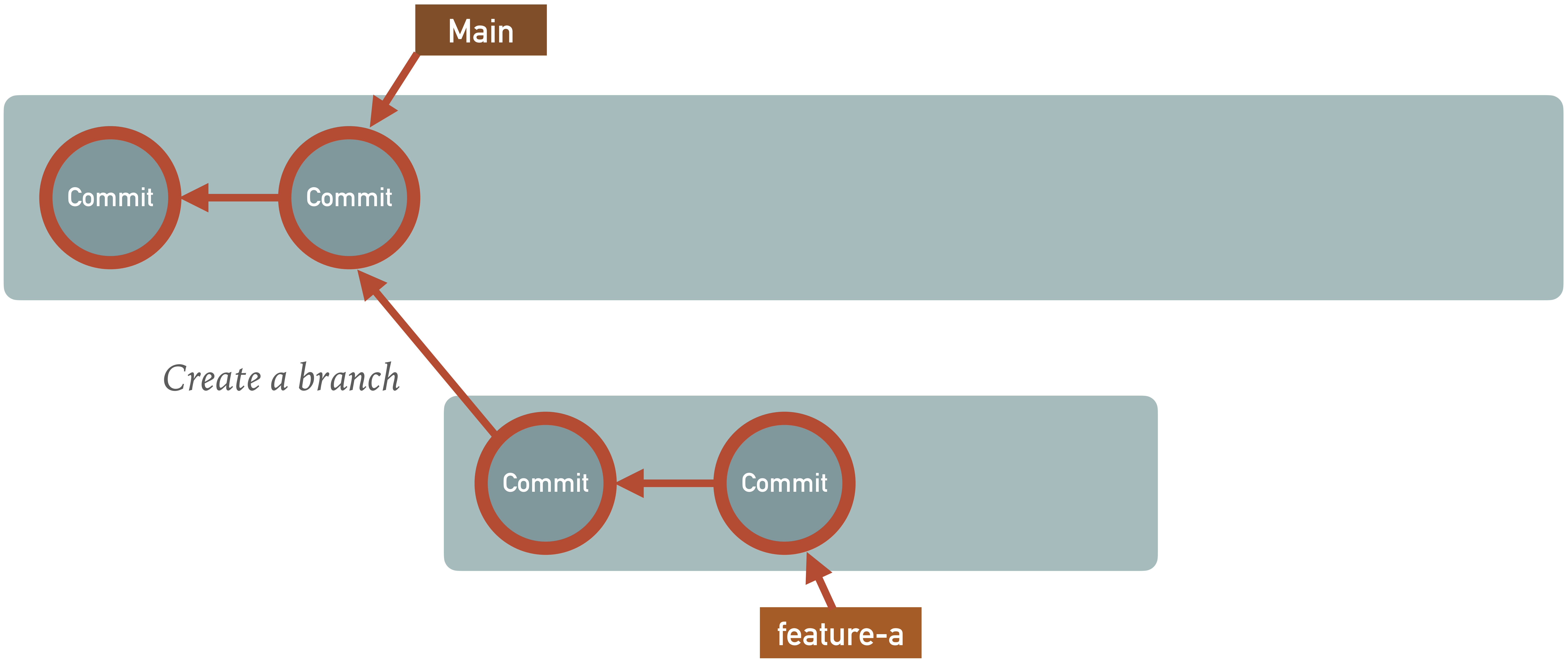
I. Fast forward merges : linear changes



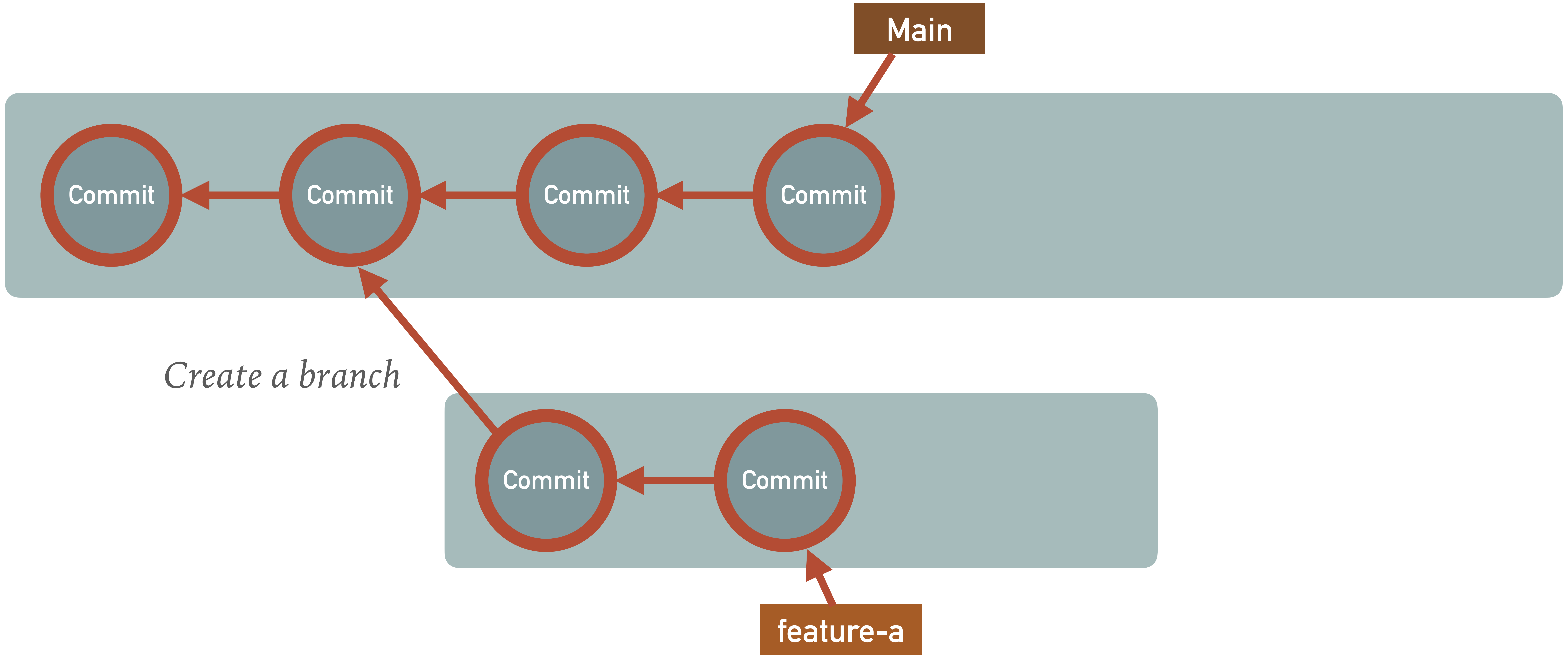
I. Fast forward merges : linear changes



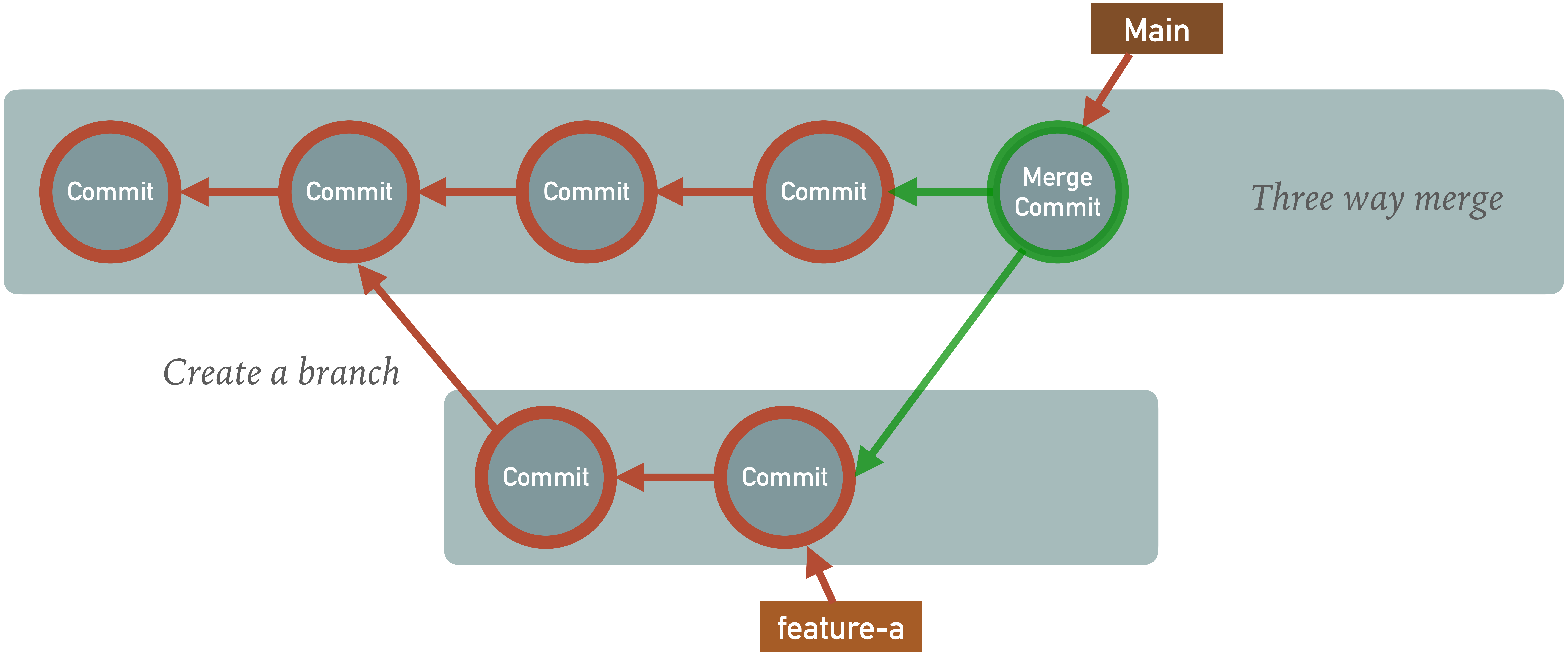
II. a. Three way merge



II. a. Three way merge



II. a. Three way merge



Git merge

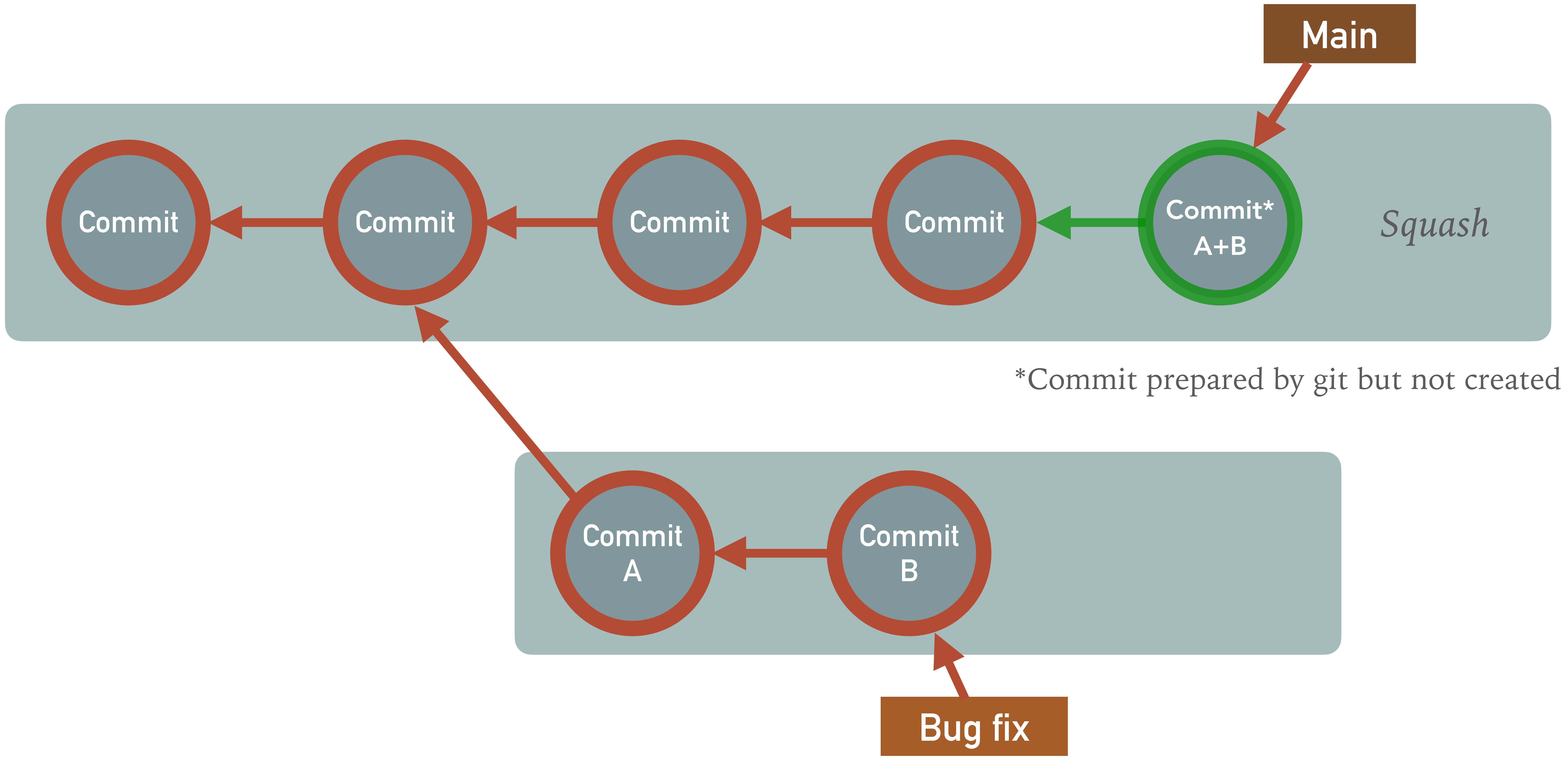
Go to the main branch and merge with another branch

```
$  git merge <name-of-the-branch>
```

```
$  git log --oneline --graph --all
```

```
$ git switch main
$ git merge classification
$ git branch -d classification #once your merge your branch with the main you can delete it
```

II. b. Git merge --squash



*Commit prepared by git but not created

Git merge --squash

In case you want to add branche's changes without merge you can do it using squash

```
$ git merge --squash <name-of-the-branch>
```

- Create a branch named : bug-fix
- Add two new files and commit changes
- Squash the merge

```
$ git switch main
$ git merge --squash bug-fix
$ git commit
$ git branch -D bug-fix #You have to force deletion because the branch is considered unmerged
```

Ps. Conflicts are resolved in the same way as merge

CONFLICTS

Conflicts

- Conflicts may occur for different reasons :
  - Different versions of the same line inside a file
  - File content different from one branch to another
  - File no longer existing

Fix conflict

```
$ vi pre-processing # Fix conflict
$ git add pre-processing #get the change back to the staging
$ git commit # without -m flag because the commit message is already pre-set by git
```

Nb. In case you don't want to resolve the conflict you can go back to the previous state

```
$ git merge --abort
```

TP2

A envoyer à : mlds.git@gmail.com

1. Créer un nouveau dossier avec 2 fichiers (main.py et README.md) et initialiser un git.
2. Ajouter les deux fichiers au stage et faire un commit.
3. Créer une nouvelle branche feature-a et se placer dedans.
4. Créer un fichier classification.py (le remplir avec un simple print).
5. Ajouter le fichiers au stage et faire un commit.
6. Faire un merge avec la main.
7. Modifier le fichier main dans les deux branches et faire un commit.
8. Faire de nouveau un merge.
9. Résoudre le conflit et commit.