Version Control

Focus on GIT

Version Control Types

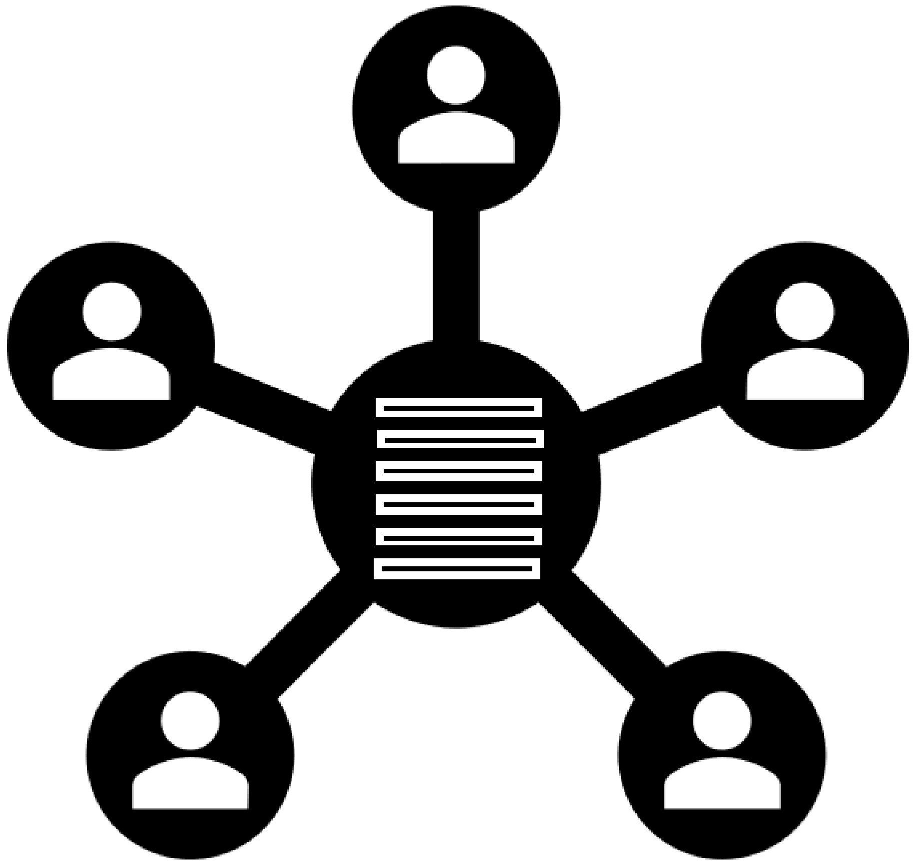
Centralized Repository Version Control (Traditional)

Distributed Repository Version Control

Centralized Repository Version Control (Traditional)

Examples

- Team Foundation Version Control
- SVN
- CVS
- ClearCase
- Team City



Centralized Repository

Pros	Cons
Easy to understand	Single point of failure
Exclusive Checkout by Default	Needs to be always connected
Supports Large Files and Repositories	Relatively slow

Distributed Repository Version Control

Examples

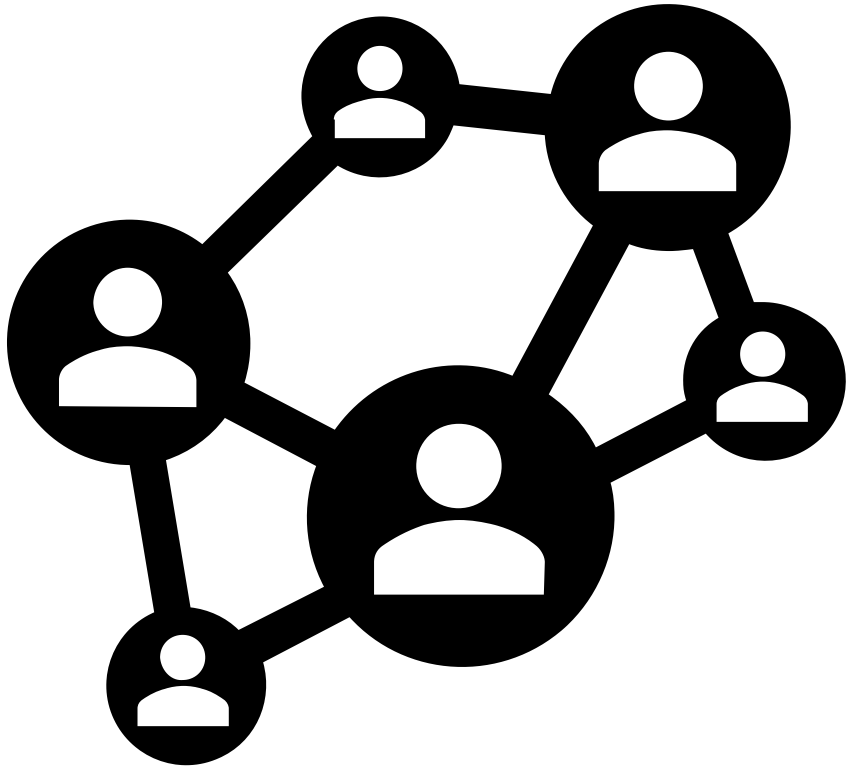
GIT

- Azure DevOps GIT
- GitHub
- GitLab
- Atlassian Bitbucket

Mercurial

Fossil

GNU Bazaar



Distributed Repository

Pros

- Fast as 90% local interaction
- No Single point of failure
- Many Options – Industry Standard

Cons

- Has some learning curve
- Necessary to follow best practices
- Difficult to handle binaries

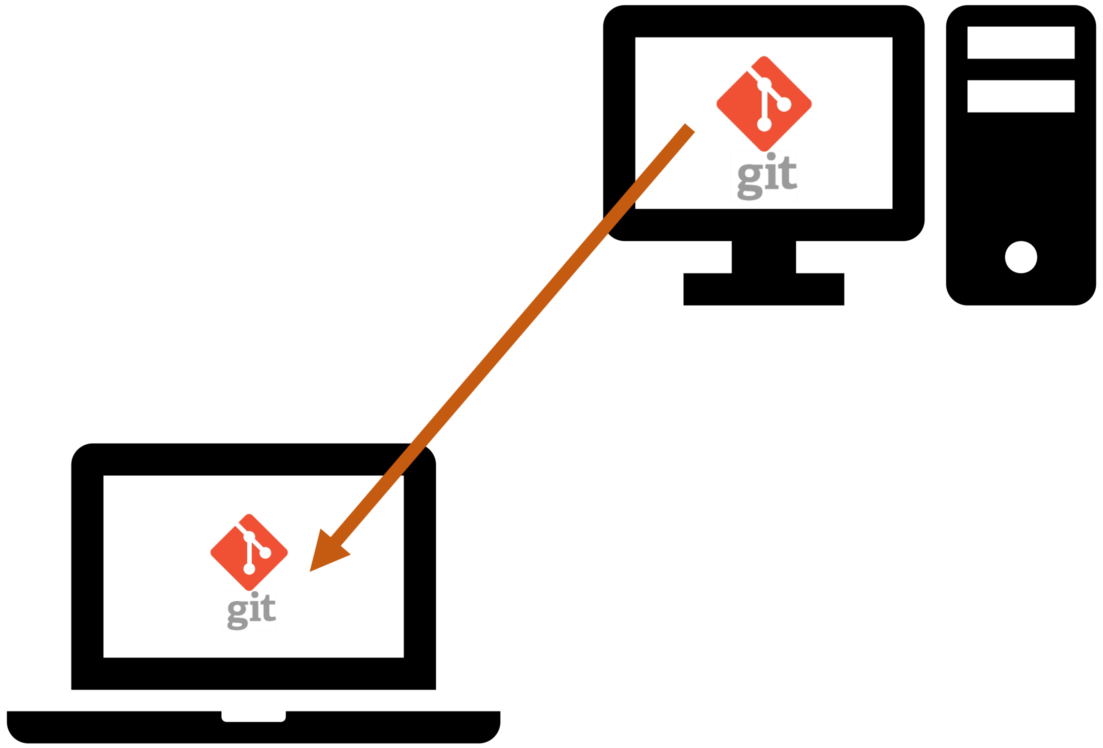
Basic Operations

Clone

Entire repository

Creates Local Repository if not existing

Sets Remote Repo linked to Local



```
git clone https://dev.azure.com/fabrikam/DefaultCollection/_git/Fabrikam C:\Repos\FabrikamFiber
```

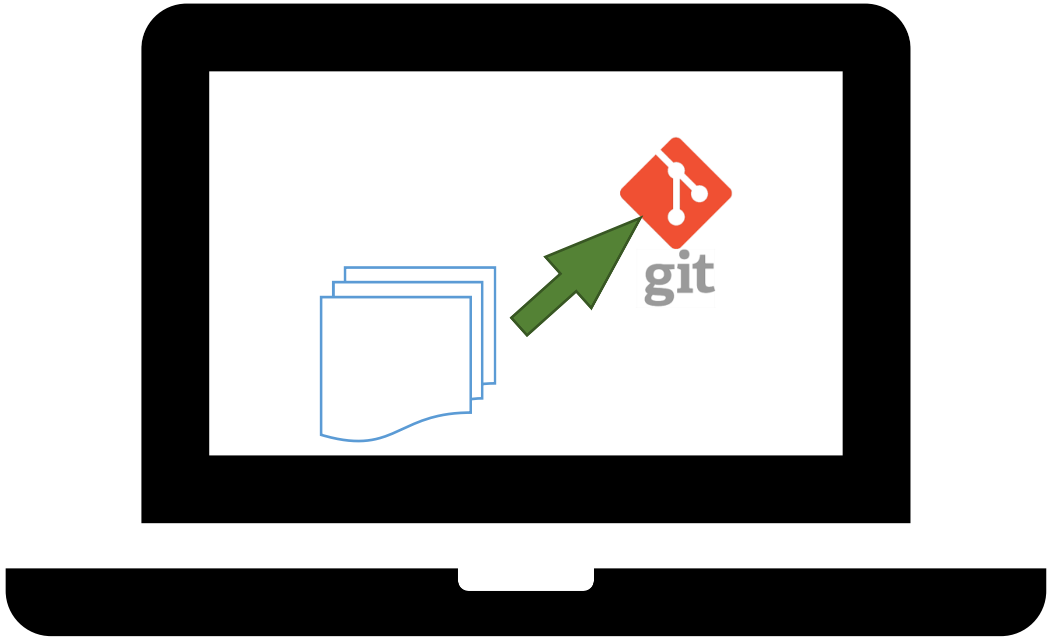
Commit

Always to local repository

All Files or Staged Files

Unique ID

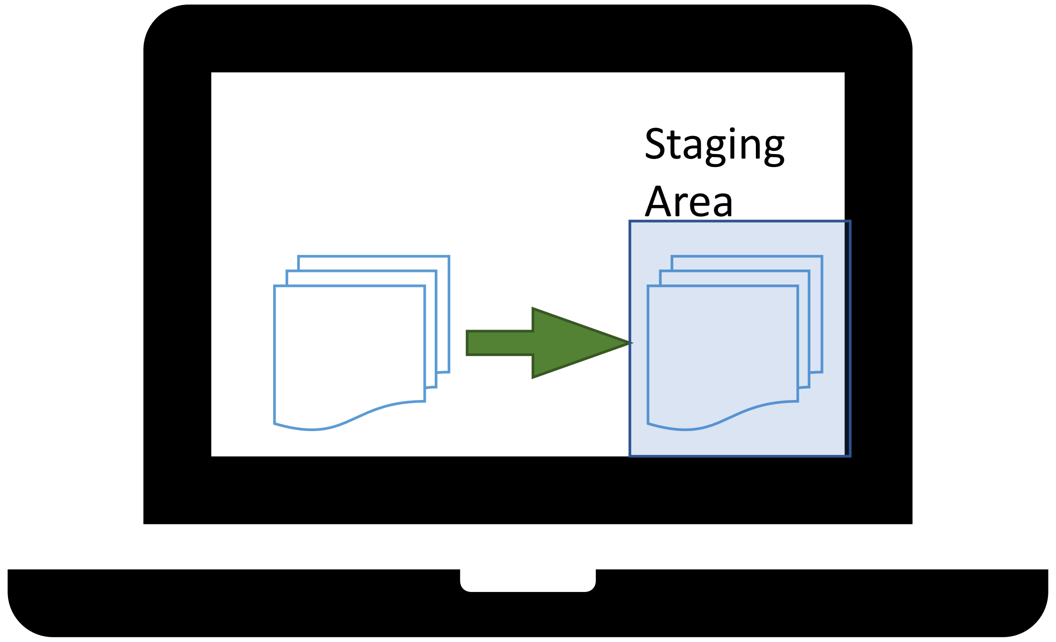
```
git commit –m "Commit Message"
```



Stage

Stores the selected files in the staging area

Indication that these files should be committed in the next commit



```
git add <file name>

git add -p
```

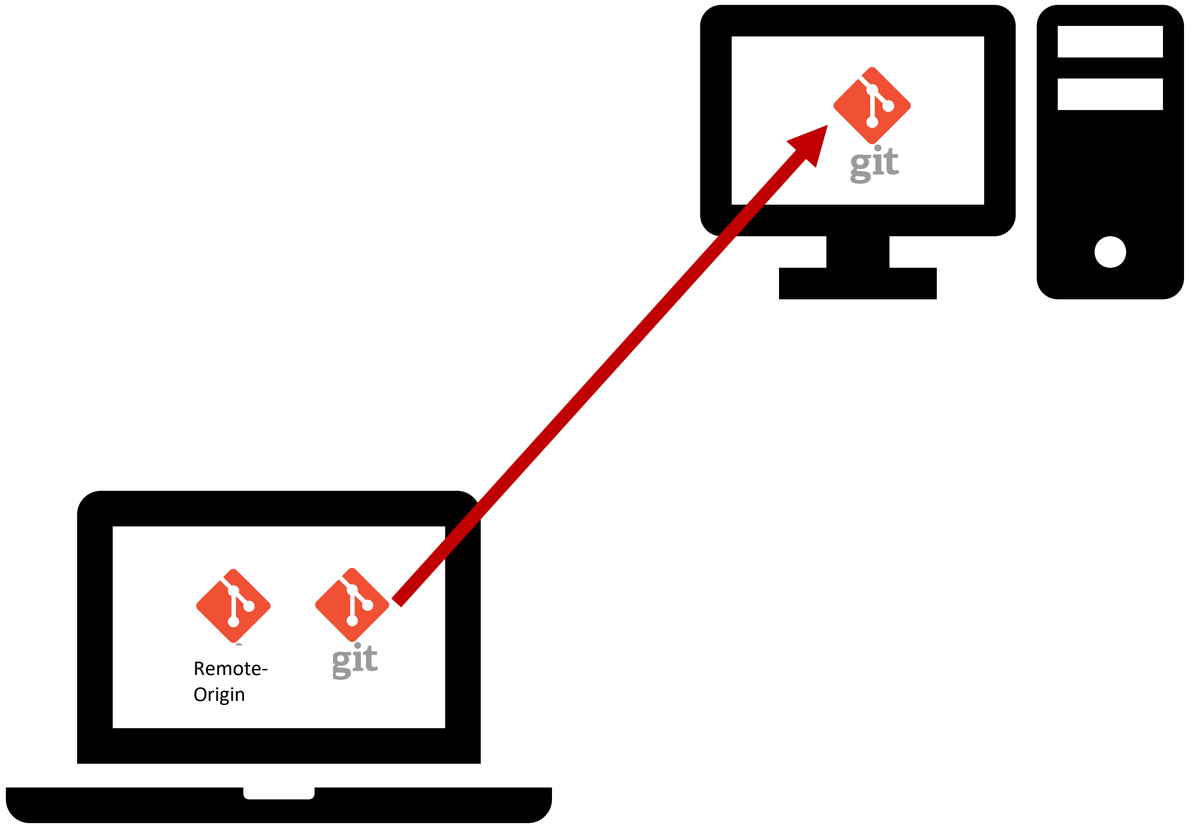
Push

From local repository to remote repository

By default, all commits

Creates remote/origin

```
git push -u origin users/frank/bugfix
```



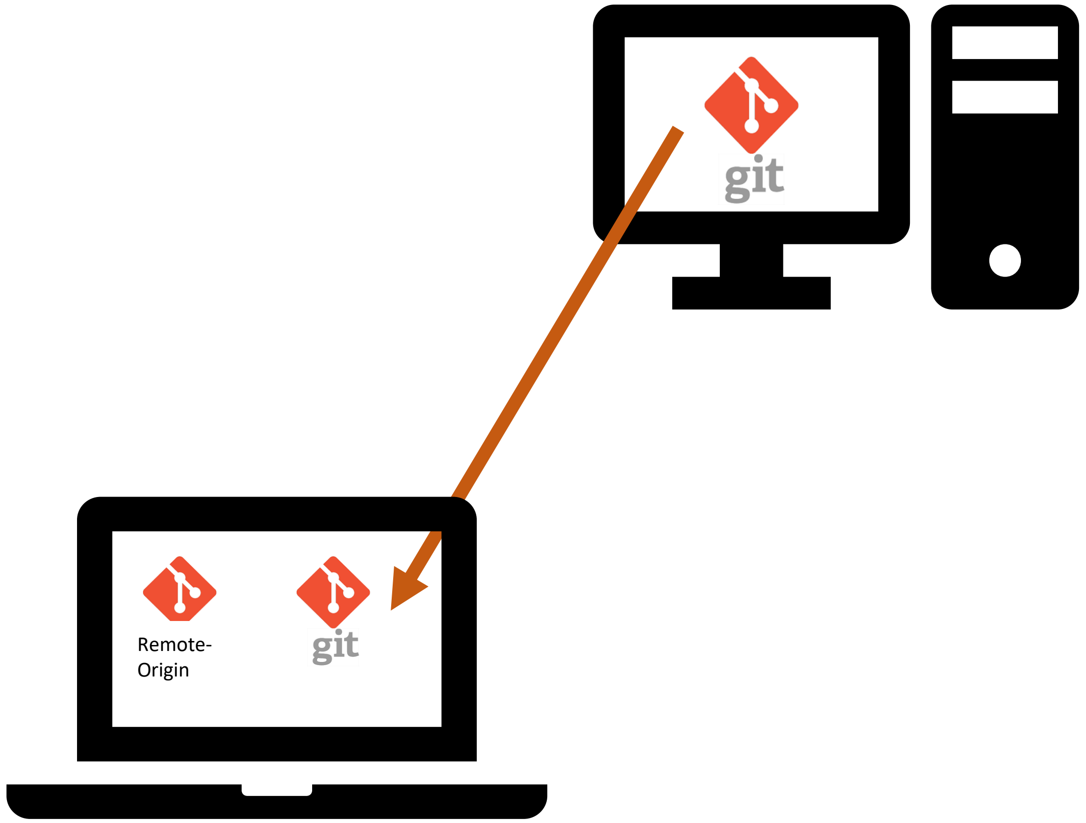
Pull

Gets the code that is pushed by other developers

Merges the code directly with the respective branch

Creates a local branch if it does not exist

`git pull origin users/frank/bugfix`



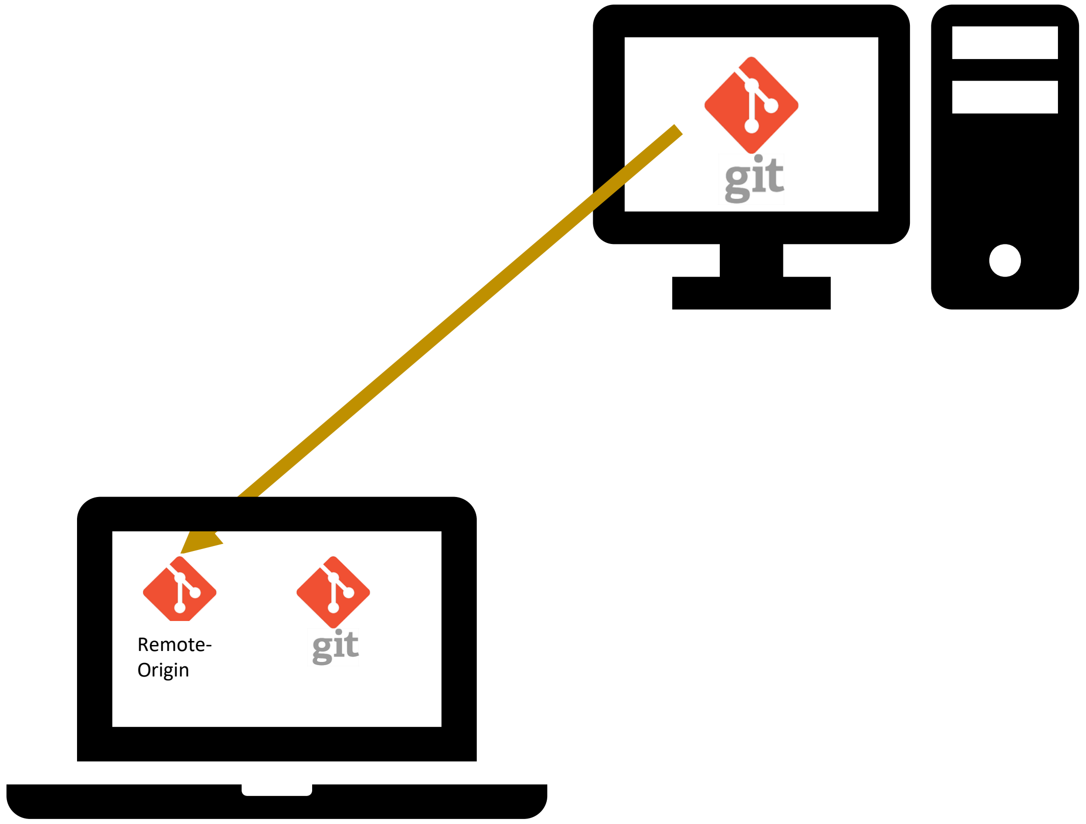
Fetch

Gets the code that is pushed by other developers

Code is saved in remote/origin, not merged

Fetched code can be viewed but not edited

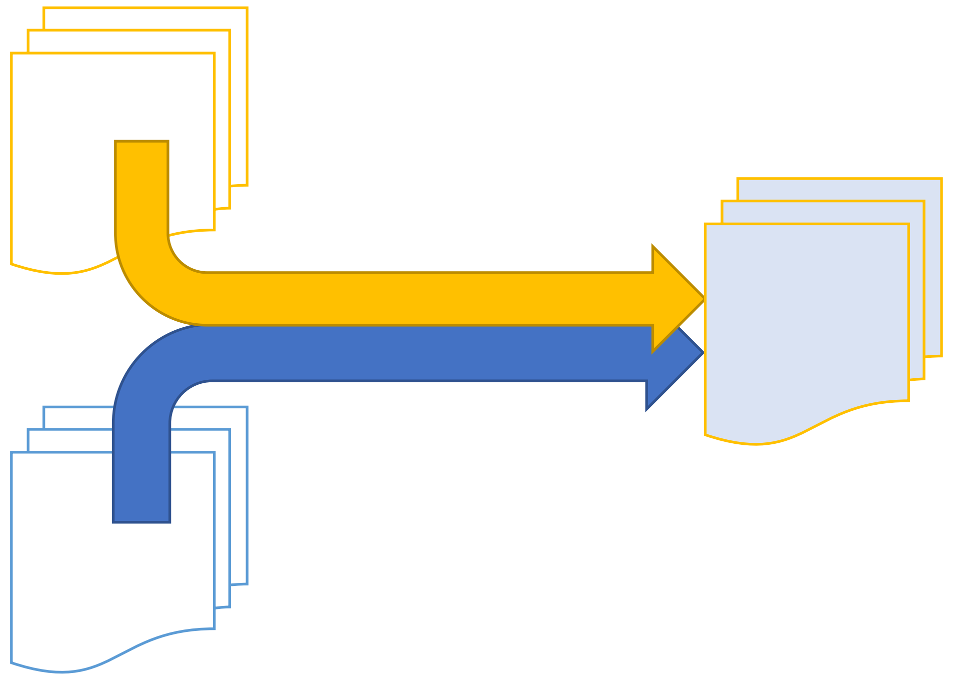
```
git fetch
```



Merge

From branch to branch within repository

From branch in remote/origin to branch in repository

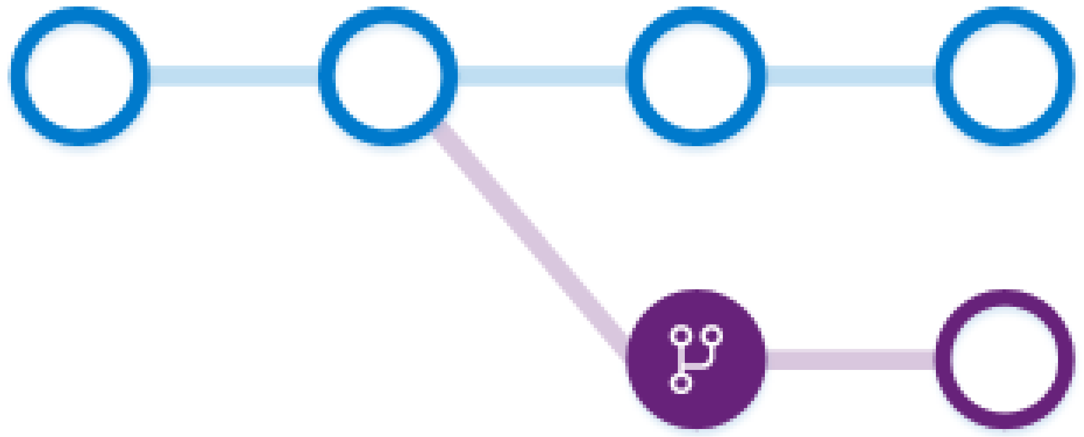


Branch

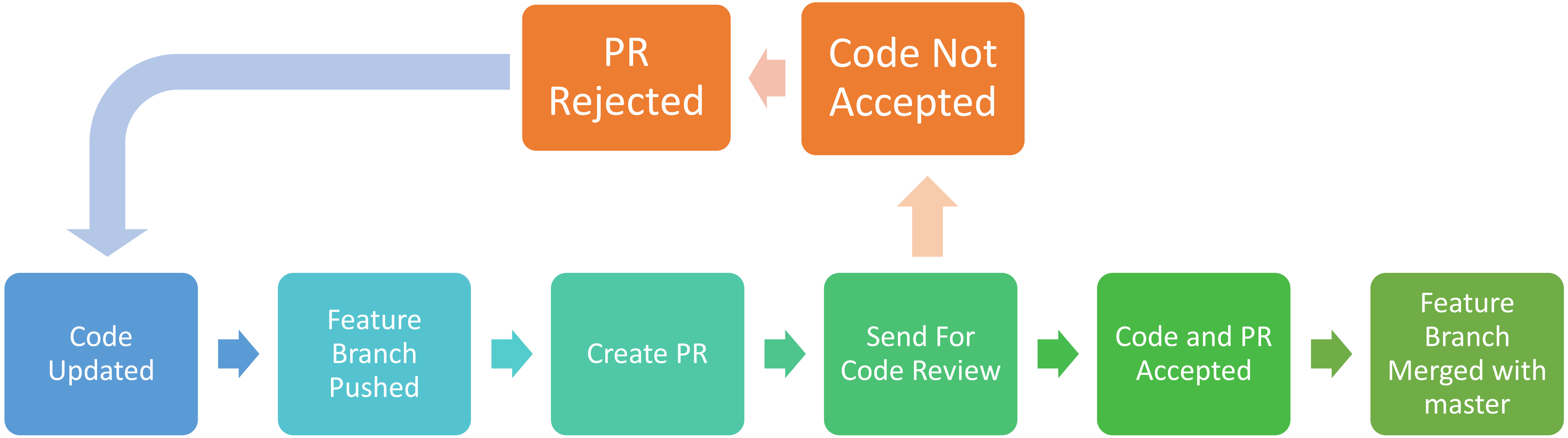
Create copy of the code for working in isolation

Local Branches are light weight

```
git branch feature1
git checkout feature1
```

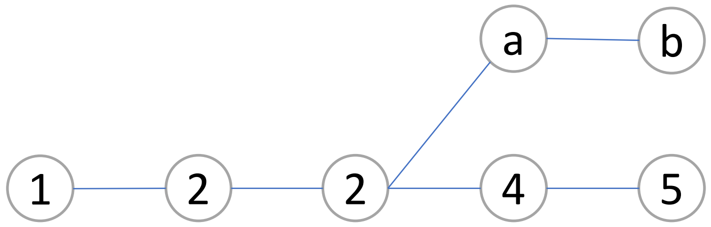


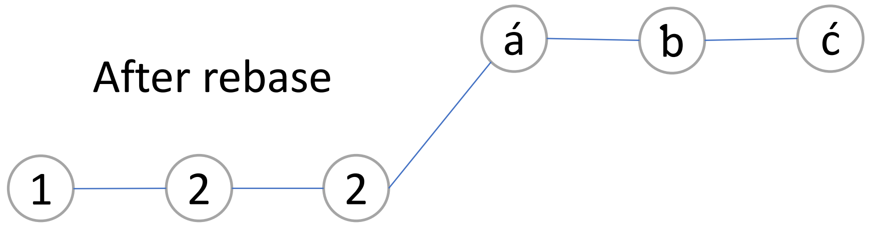
Pull Request

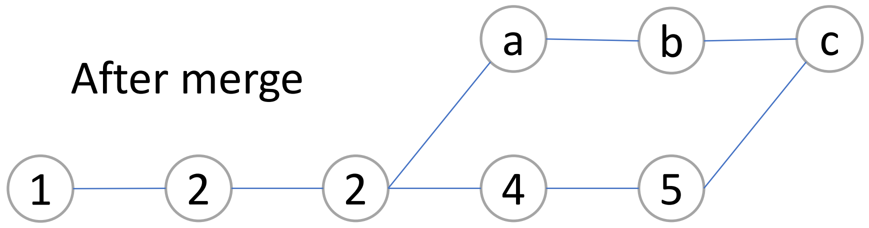


Rebase

- Similar to Merge
- History of target does not show the commits that were made in source after the branch was created.



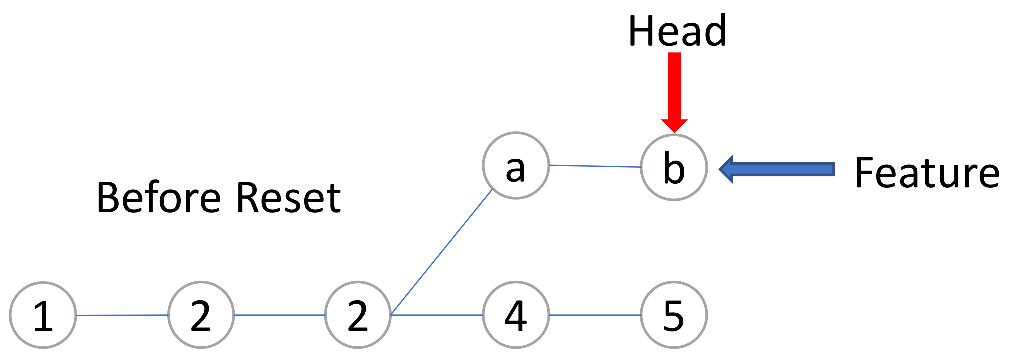


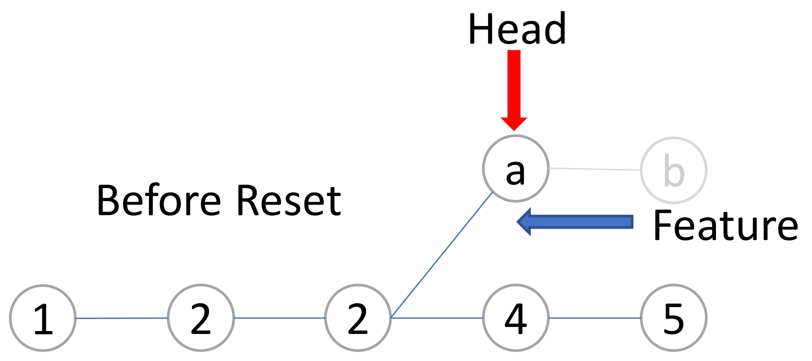


Reset

- Changes the status of branch and head to the commit that is specified in the command

git reset a

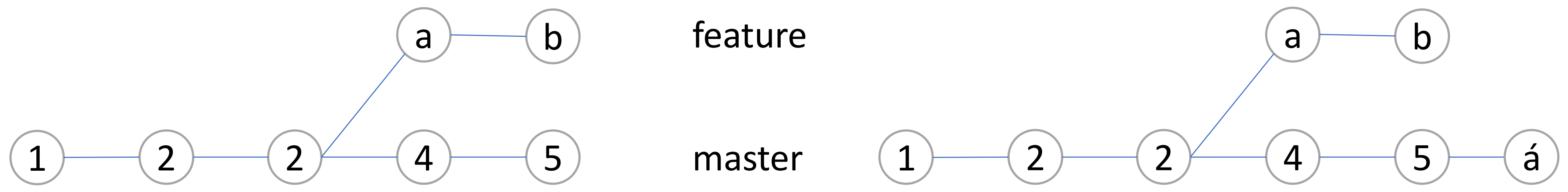




Cherry Pick

- Brings only some commits from the source branch to target branch
- Applies the changes introduced by the named commit on the current branch.
- It will introduce a new, distinct commit.

```
git checkout master
git cherry-pick feature~a
```



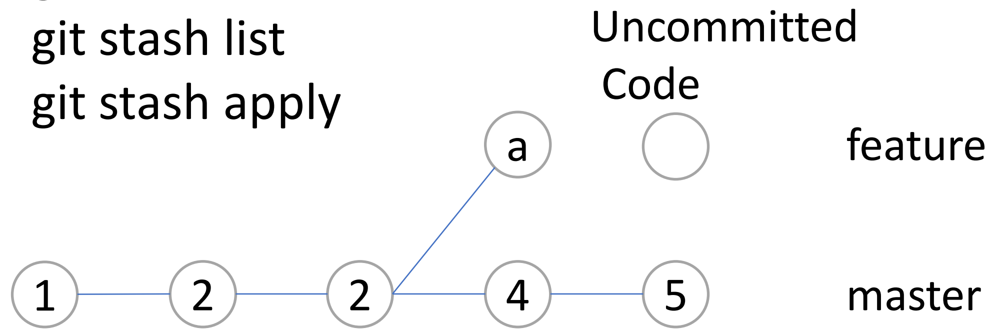
Before Cherry-pick

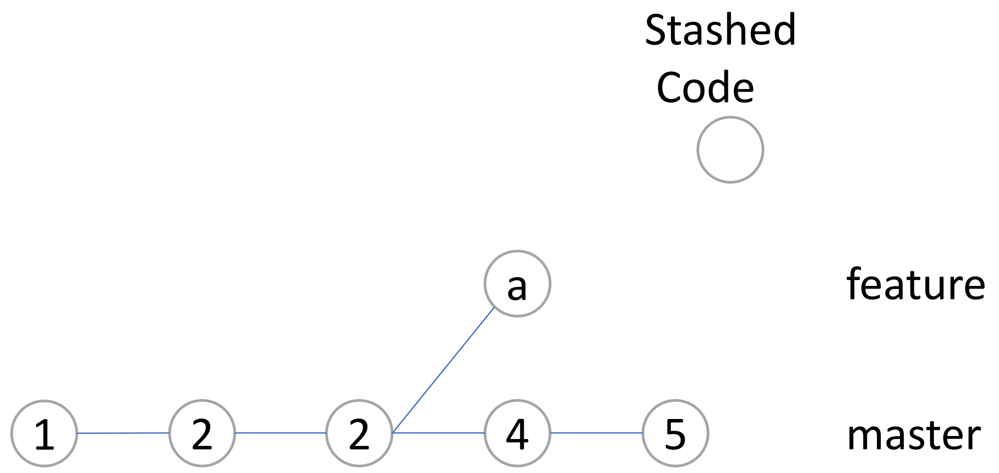
After Cherry-pick

Stash

- Stores the current state of files without commit and returns the working directory to HEAD (Earlier successful commit).

```
git stash
git stash list
git stash apply
```





Make changes in commit

```
git commit -m 'Initial commit'
git add forgotten_file
git commit --amend
```