USING GIT FOR REVISION CONTROL

INTRODUCTION

Git is a distributed revision control system. Its purpose is to provide a structured way to track changes to files as well as manage contributions from many collaborators.

THE NOUNS OF GIT

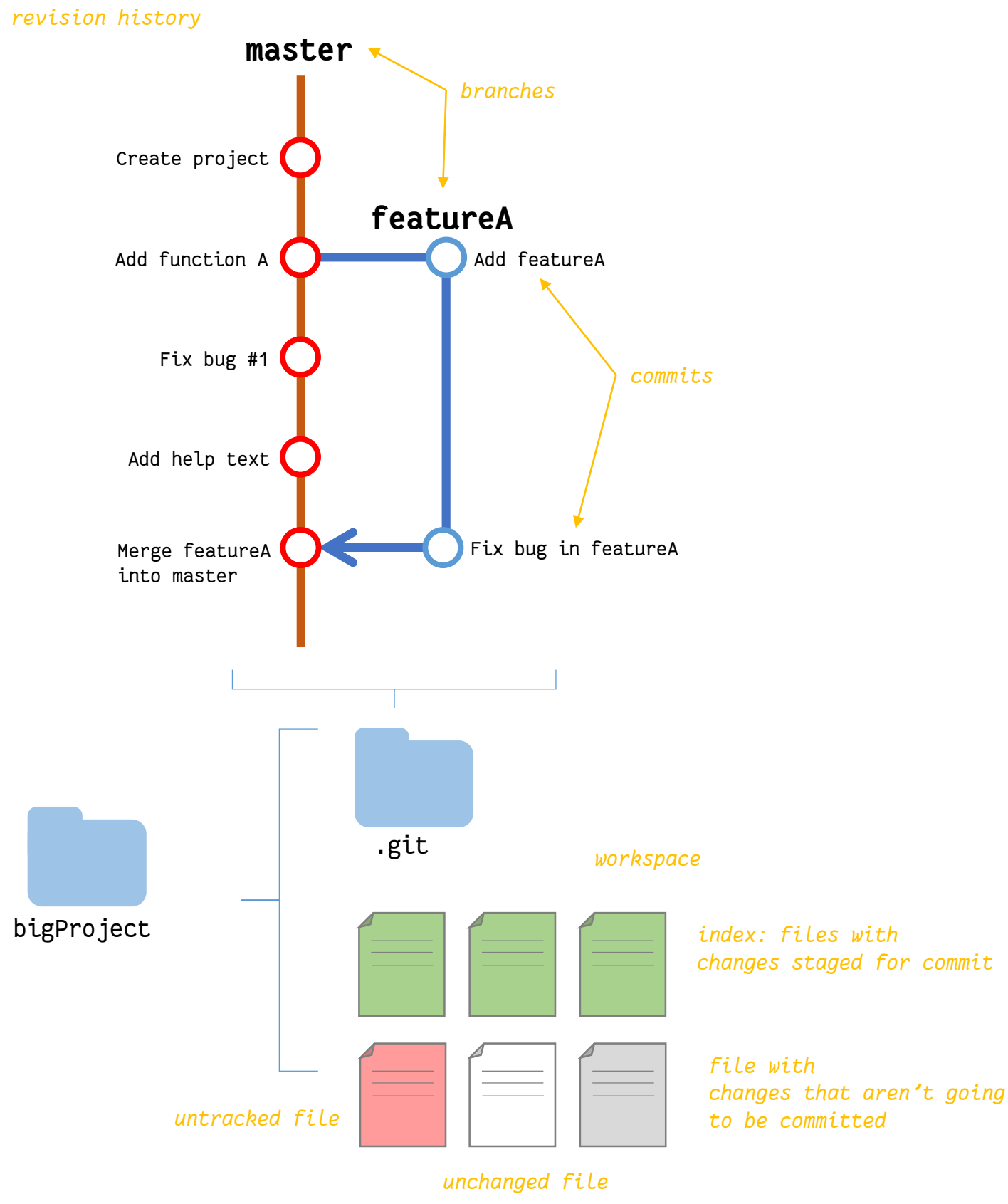
Repository Revision-controlled directory. Inside a "repo", the hidden .git folder contains metadata about the project, its upstream source, and all the changes that have been made to it. The distinction between Git and centralized revision control systems is that each developer uses Git to make a local copy of the "remote" (upstream or official repo), makes their changes, and then records them using commits. Synchronization between the remote and everyone's local copy allows each developer's changes to propagate to their peers.

Commit The fundamental unit of a revision control system. A snapshot of a repository at a specific time, annotated with a log entry ("commit message") that describes the changes made since the last commit.

Branch A timeline or series of commits. The mainline branch is usually called the "master" or "trunk." Branches can diverge from the trunk, creating an alternate timeline to develop a new feature without touching the stable code. These changes can later be merged into the main branch.

Workspace The project's working directory. The files in the workspace reflect the state of the current branch as of the last commit, plus any changes since then. Untracked files are not associated with a particular branch, and show up in all of them.

Index Staging area for commits. Adding a file to the index caches its changes and tells Git to snapshot it the next time a commit is made. *Changes not staged in the index are not committed.*



USING GIT FOR REVISION CONTROL

THE VERBS OF GIT

git <command> <-flags> <inputs>
git checkout -b featureA

add <files> stage files for commit

mv <oldName> <newName> rename a file, preserving its history at the old name. Index the change.

rm <file> delete a file, recording the change in the index.

commit -m "<message>" commit indexed changes

branch <name> create a new branch

checkout <name> switch to branch

merge <branch> play back commits from "branch" onto the current branch

rebase <parent> rewind commits on current branch, add commits from the parent since the branches diverged, then add changes back in

revert <commit> undo commit

reset --hard roll back changes since commit

stash save changes for later, reverting edits

stash pop restore stashed changes

status show list of changes and untracked files

log -p pretty-print commit log

diff show line-by-line differences between files in the workspace and last commit

diff --cached show line-by-line differences between staged files and last commit

init create empty git repository

clone <remote url> copy a remote repository

fetch <remote> get changes from remote branch

pull fetch and merge changes from remote branch

push publish local commits to remote

request-pull generates a patch message that can be sent to others describing your changes and where they can be downloaded

```
>> git clone git@github.com:user/project.git
```

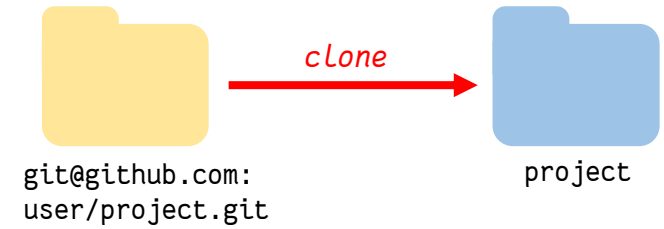
```
>> cd project
>> ls
   file1
   file2
```

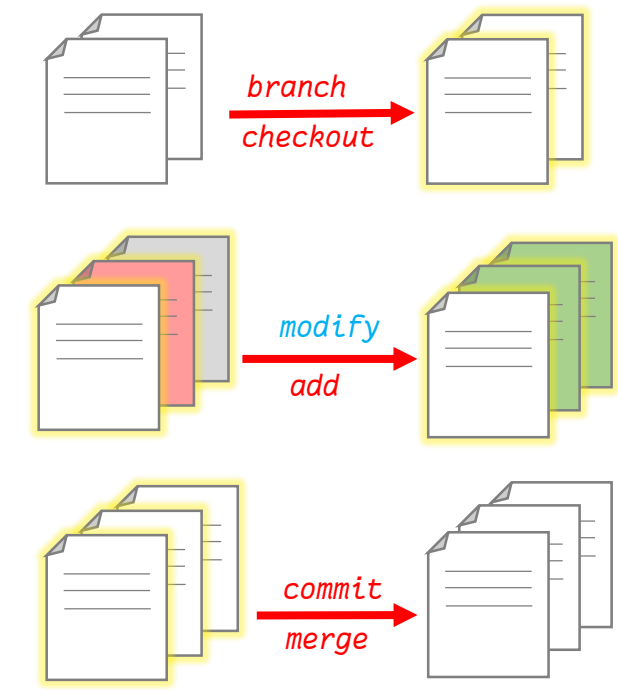
```
>> git checkout -b bugfixBranch
>> touch file3
>> echo "new code" >> file1
>> git status
   modified:        file1
   untracked files: file3
>> git add .
>> git status
   modified: file1
   new file: file3
>> git commit -m "fixed stuff"
```

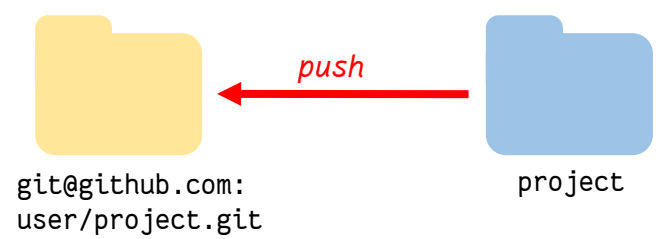
```
>> git checkout master
>> git diff bugfixBranch
   modified file1
   +fixed stuff
   created file3
>> git merge bugfixBranch
```

```
>> git push origin master
```

```
>> git request-pull bigProjectv1.0 \
https://github.com/user/project master \
> pullrequest.txt
```







MORE RESOURCES

Code hosting & collaboration
github.com
bitbucket.org

Official documentation
git-scm.com/doc

Tutorials & cheatsheets
atlassian.com/git