# Introduction to Git and GitHub

I. Introduction to source control

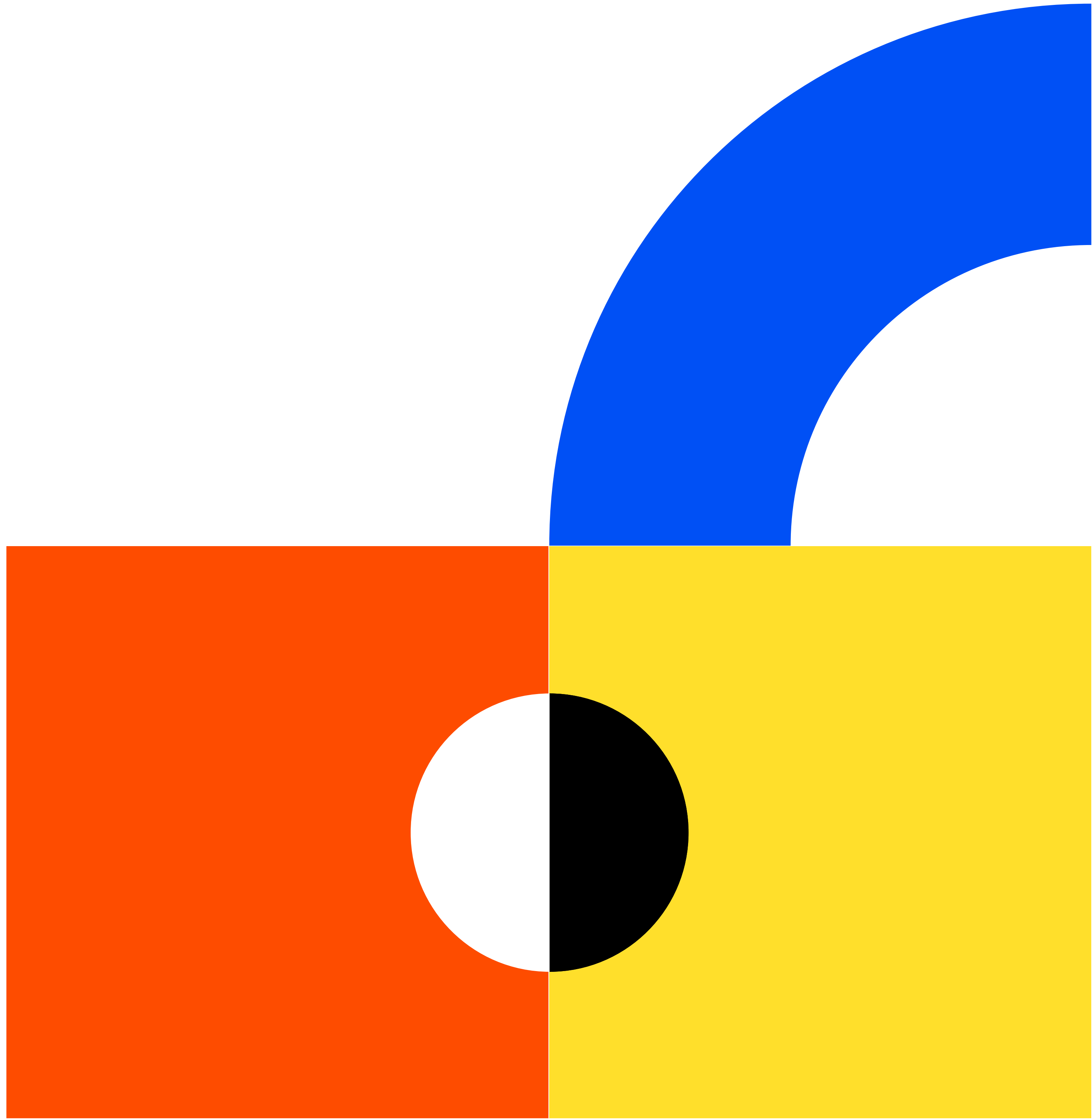
A. History and fundamental concepts behind source control
B. Centralized vs. distributed version control

II. Introduction to Git

A. What is Git? Basic Git concepts and architecture
B. Git workflows: Creating a new repo (adding, committing code)
C. HEAD
D. Git commands (checking out code)
E. Master vs branch concept
F. Creating a branch/switching between branches
G. Merging branches and resolving conflicts

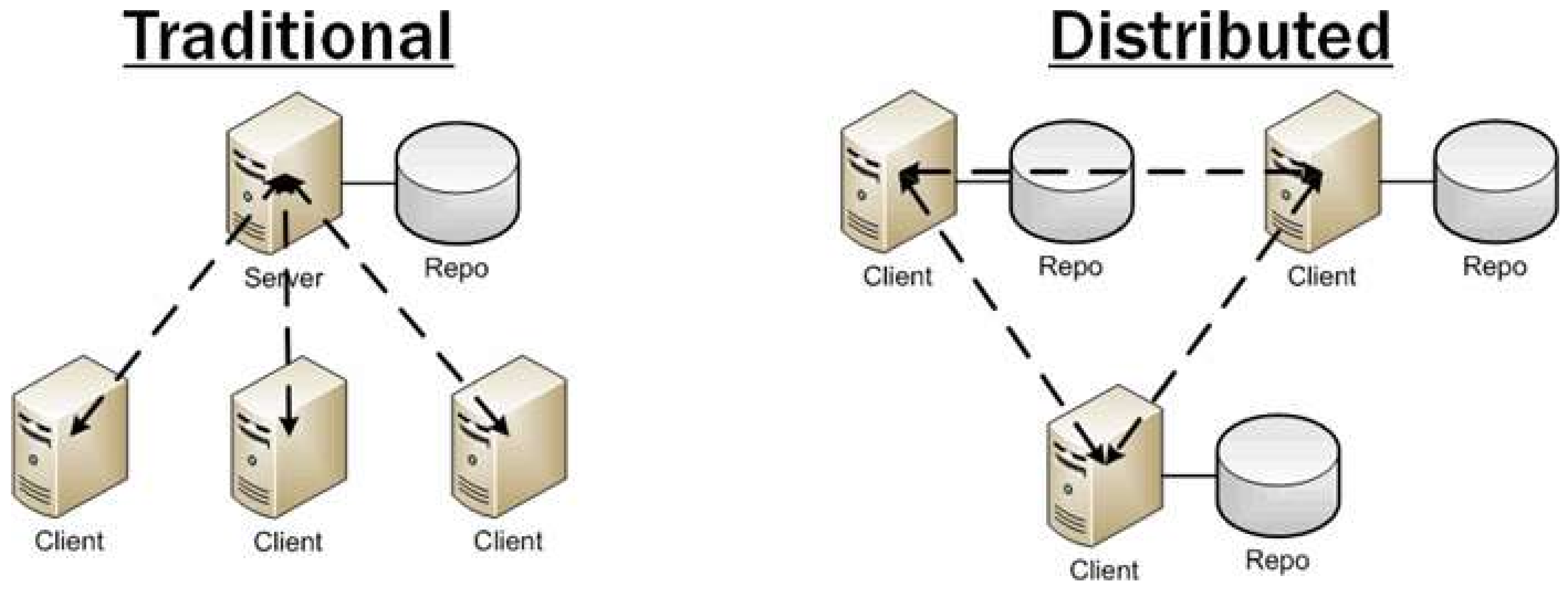
## WHAT IS A ‘VERSION CONTROL SYSTEM?’

- a way to manage files and directories
- track changes over time
- recall previous versions
- ‘source control’ is a subset of a VCS.



# Distributed version control

No central server
Every developer is a client, the server and the repository



## What is git?

- created by Linus Torvalds, April 2005
- replacement for BitKeeper to manage Linux kernel changes
- a command line version control program
- uses checksums to ensure data integrity
- distributed version control (like BitKeeper)
- cross-platform (including Windows!)
- open source, free

## Git distributed version control

“If you’re not distributed, you’re not worth using.” – Linus Torvalds

- no need to connect to central server
- can work without internet connection
- no single failure point
- developers can work independently and merge their work later
- every copy of a Git repository can serve either as the server or as a client
(and has complete history!)
- Git tracks changes, not versions
- Bunch of little change sets floating around

## Is Git for me?

- People primarily working with source code
- Anyone wanting to track edits (especially changes
to text files)
- review history of changes
- anyone wanting to share, merge changes
- Anyone not afraid of
command line tools

## Most popular languages used with Git

- HTML
- CSS
- Javascript
- Python
- ASP
- Scala
- Shell scripts
- PHP
- Ruby
- Ruby on Rails
- Perl
- Java
- C • C++
- C#
- Objective C
- Haskell
- CoffeeScript
- ActionScript

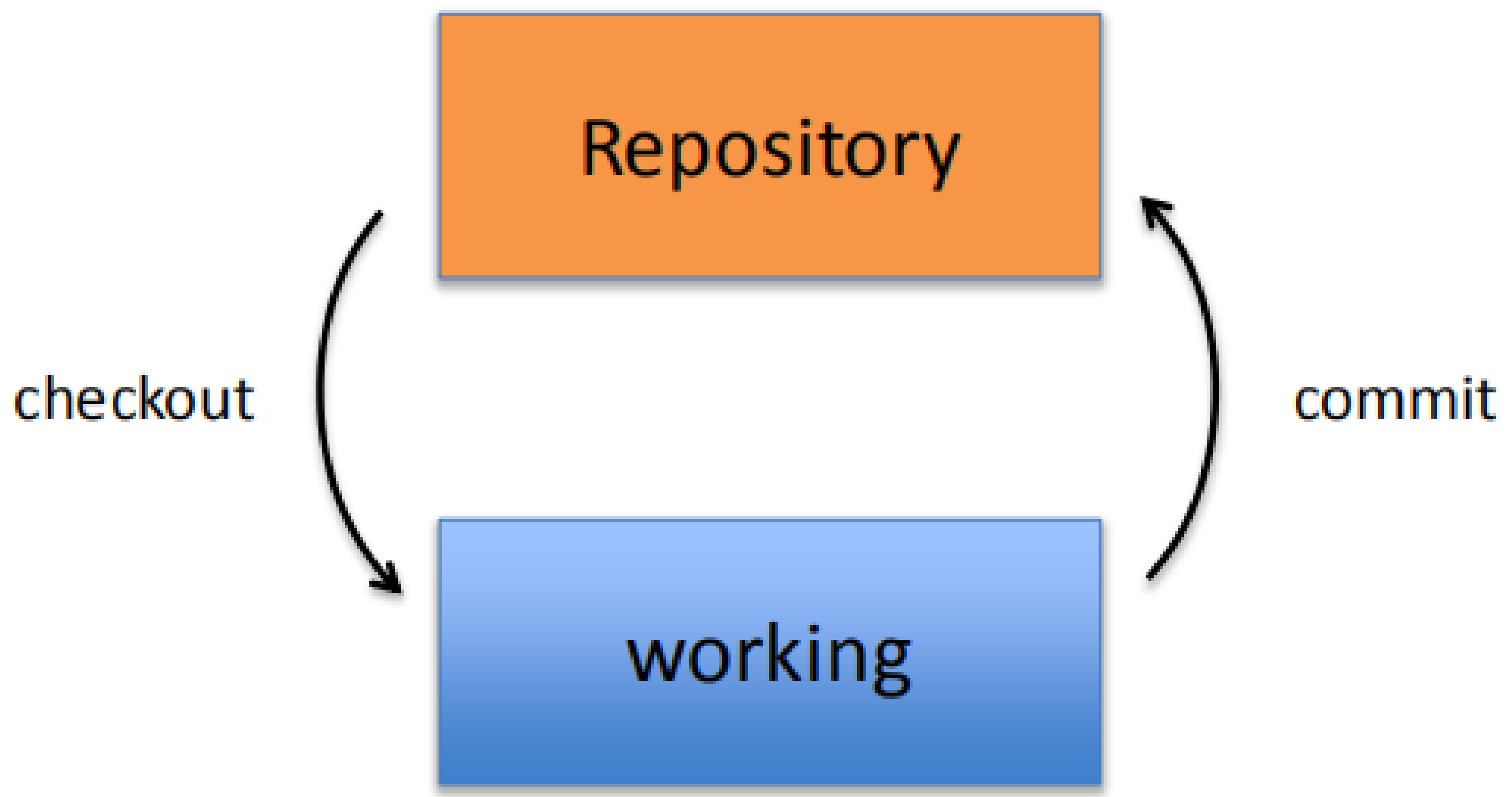
## How do I get it?

https://git-scm.com/

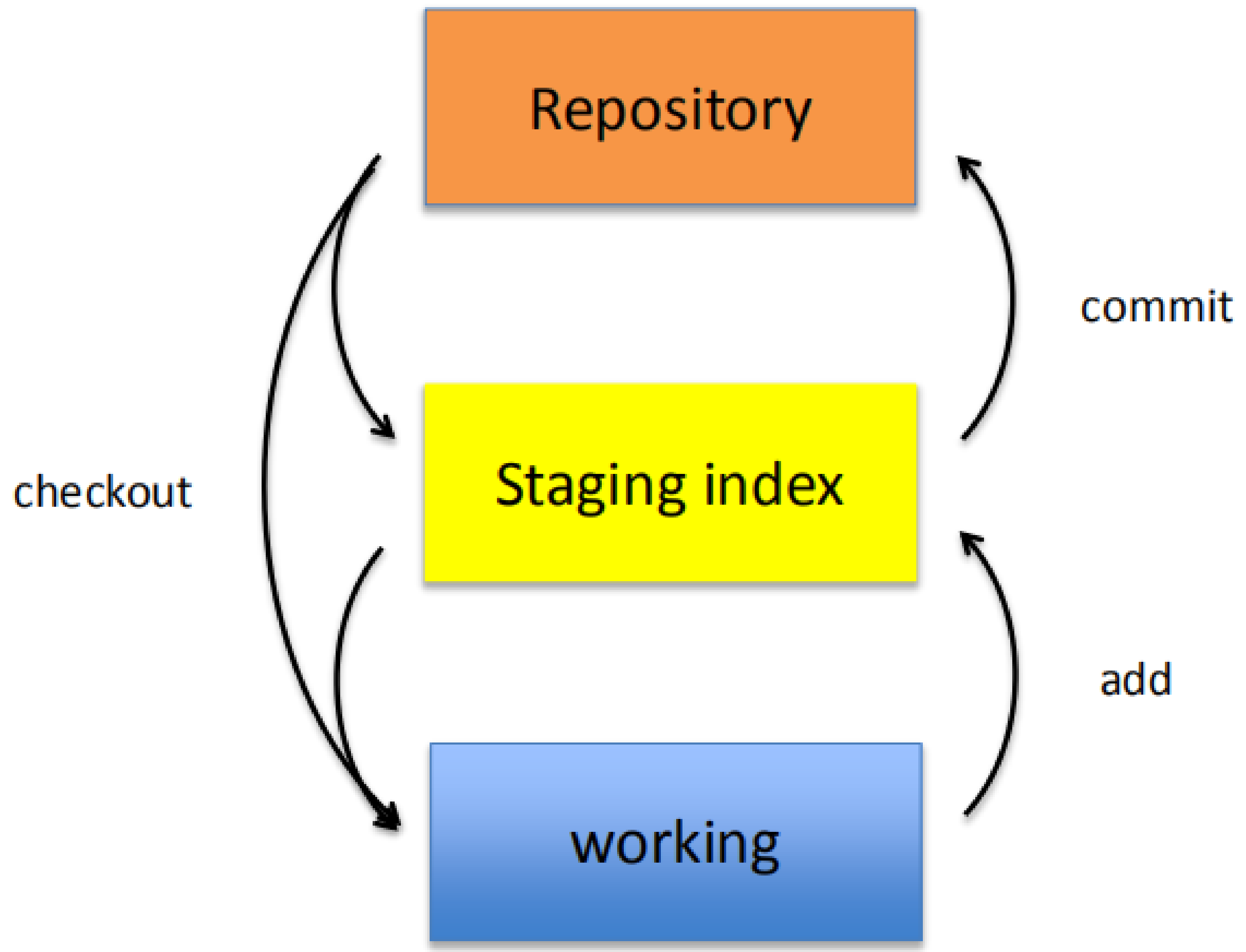
## What is a repository?

- “repo” = repository
- usually used to organize a single project
- repos can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs

## Two-tree architecture



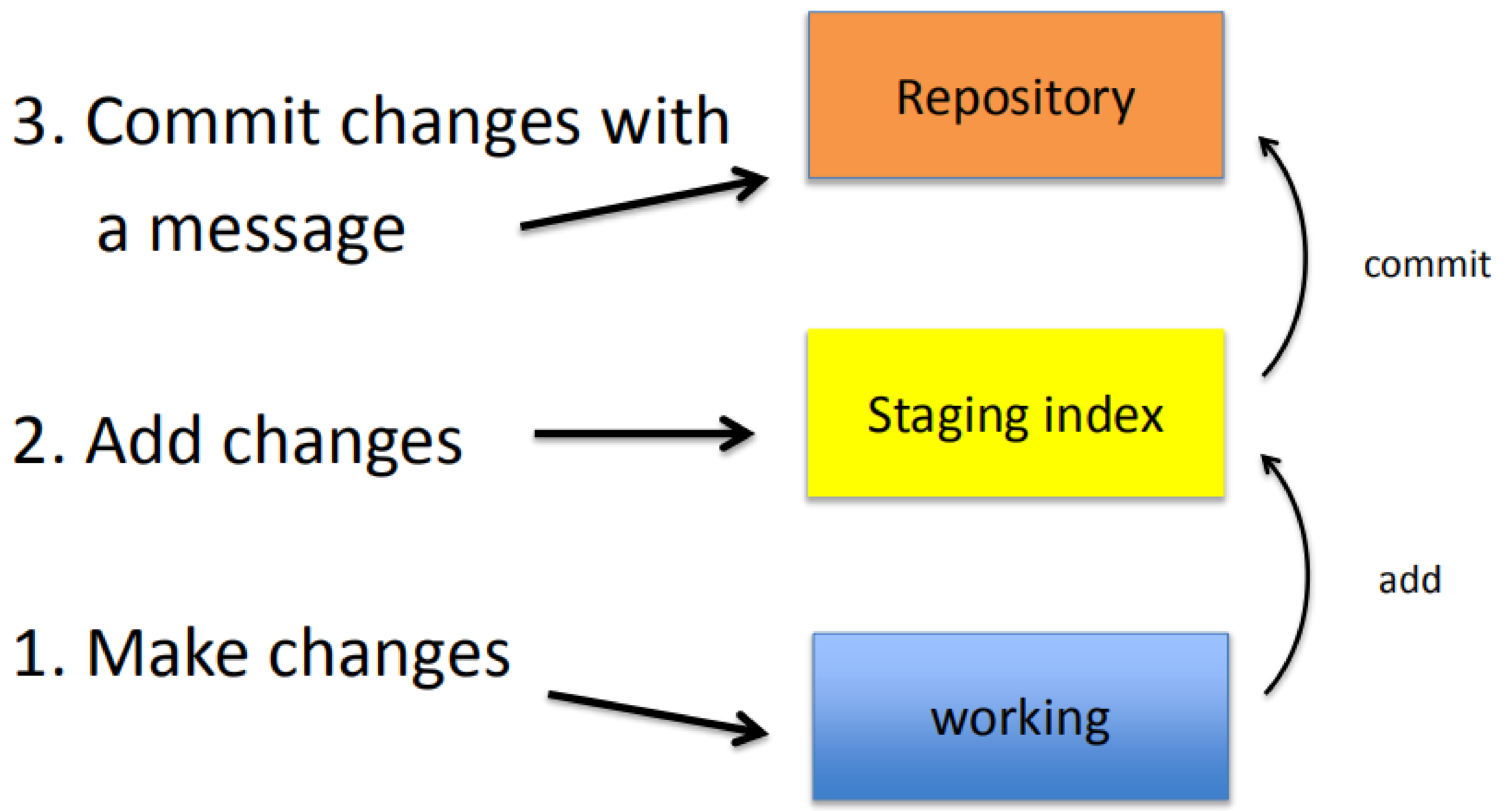
# Git uses a three-tree architecture



## A simple Git workflow

1. Initialize a new project in a directory:
git init
2. Add a file using a text editor to the directory
3. Add every change that has been made to the directory:
git add .
4. Commit the change to the repo:
git commit –m “important message here”

# After initializing a new git repo...

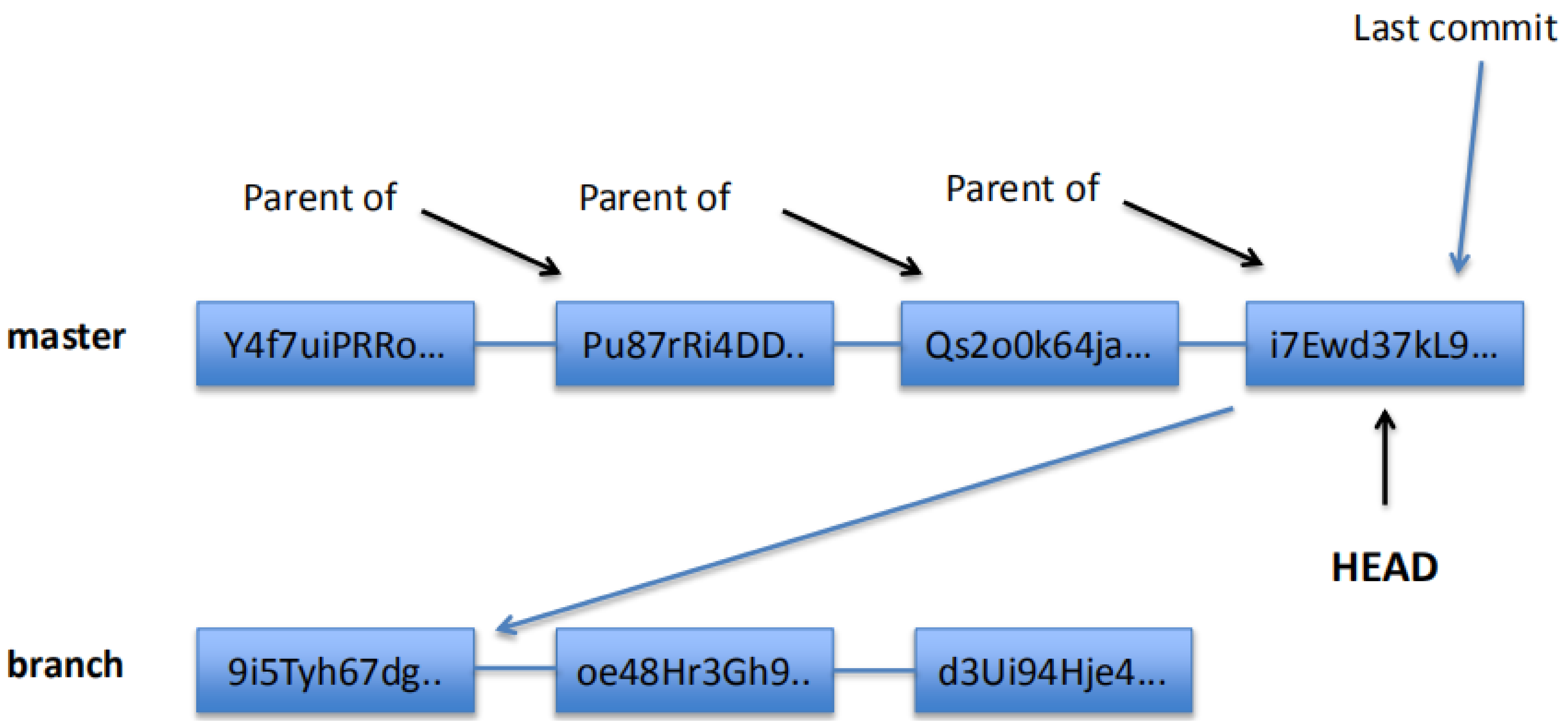


## A note about commit messages

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Keep lines to <= 72 characters
- Ticket or bug number helps

## The HEAD pointer

• points to a specific commit in repo
• as new commits are made, the pointer
changes
• HEAD always points to the “tip” of the
currently checked-out branch in the repo
• (not the working directory or staging index)
• last state of repo (what was checked out initially)
• HEAD points to parent of next commit (where writing the
next commit
takes place)


Last commit
Parent of
Parent of
Parent of
master
Y4f7uiPRRo…
Pu87rRi4DD..
Qs2o0k64ja…
i7Ewd37kL9…
HEAD
branch
9i5Tyh67dg..
oe48Hr3Gh9..
d3Ui94Hje4…

## Which files were changed and where do they sit in the three tree?

git status – allows one to see where files are in the three tree scheme

## Deleting files from the repo

git rm filename.txt

- moves deleted file change to staging area
- It is not enough to delete the file in your working directory. You must commit the change.

## Moving (renaming) files

```
git mv filename1.txt filename2.txt
```

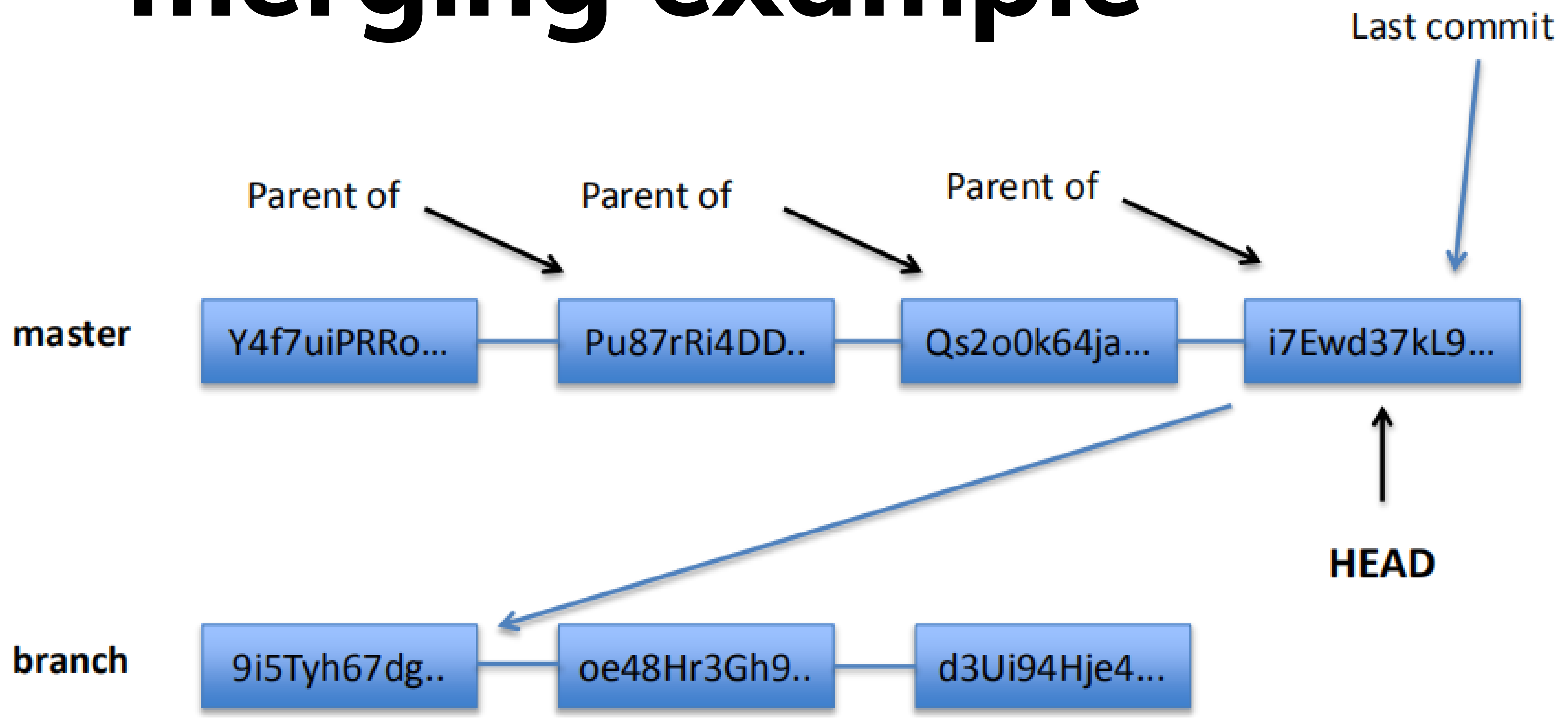
## 75% of the time you'll be using only these commands

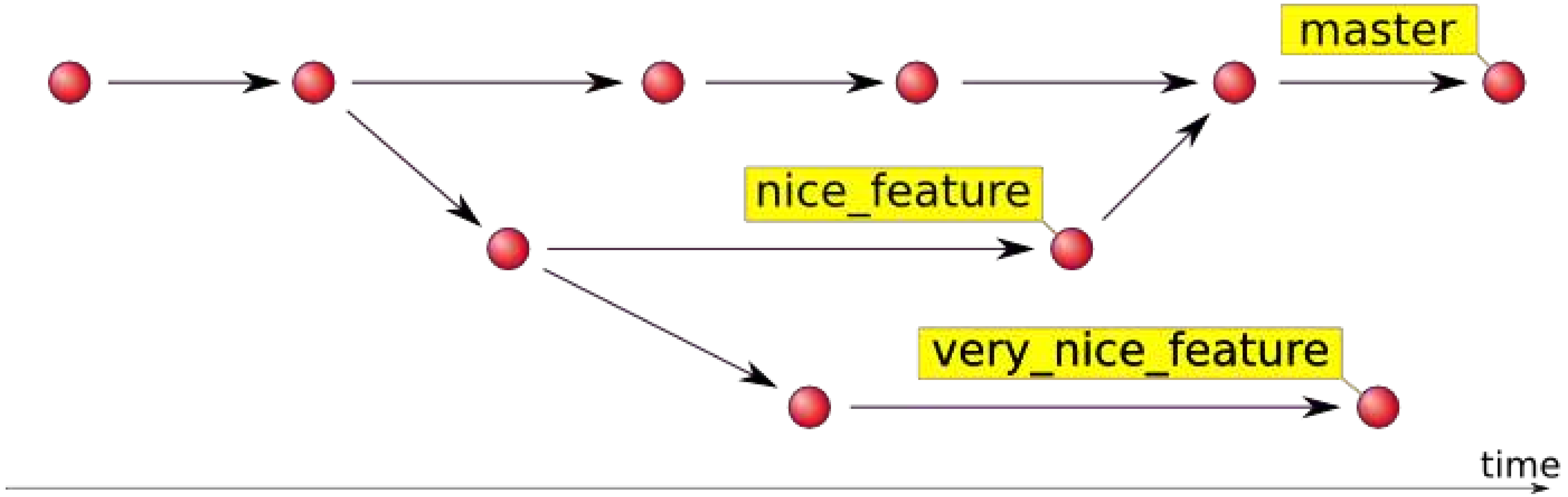
```
git init
git status
git log
git add
git commit
git diff
git rm
git mv
```

# Branching

- allows one to try new ideas
- If an idea doesn’t work, throw away the branch.
Don’t have to undo many changes to master branch
- If it does work, merge ideas into master branch.
- There is only one working directory

# Branching and merging example




master
nice_feature
very_nice_feature
time

## In which branch am I?

```
git branch
```

## How do I create a new branch?

```
git branch new_branch_name
```

**Note: At this point, both HEADs of the branches are pointing to the same commit (that of master)**

## How do I switch to new branch?

```
git checkout new_branch_name
```

**At this point, one can switch between branches, making commits, etc. in either branch, while the two stay separate from one another.**
**Note: In order to switch to another branch, your current working directory must be clean (no conflicts, resulting in data loss).**

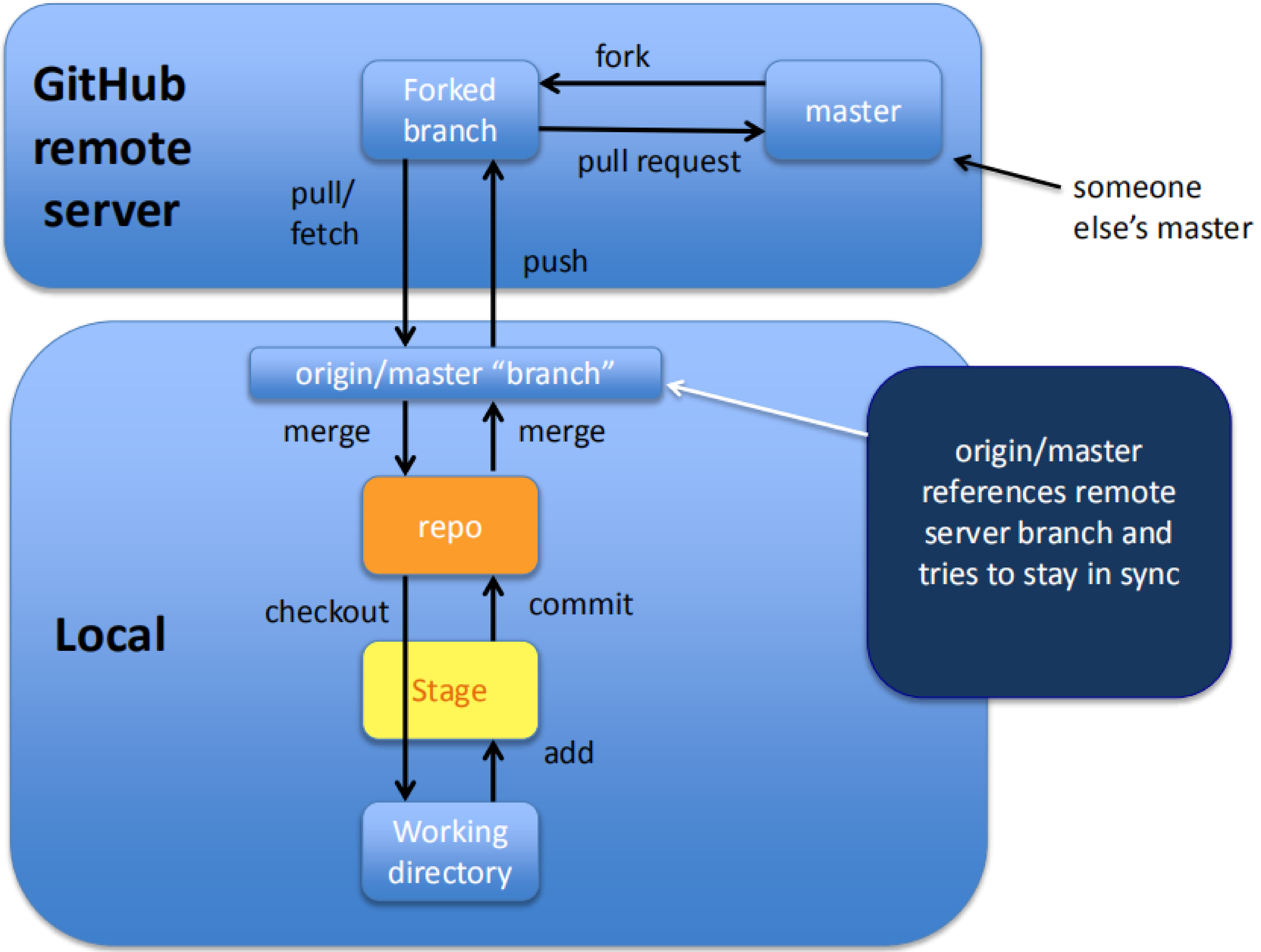
## How do I merge a branch?

```
git merge branch_to_merge
```

**Note: Always have a clean working directory when merging**

## What is GitHub?

- a platform to host git code repositories
- http://github.com
- launched in 2008
- most popular Git host
- allows users to collaborate on projects from anywhere
- GitHub makes git social!
- Free to start


GitHub remote server
fork
Forked branch
master
pull request
someone else's master
pull/ fetch
push
origin/master "branch"
merge
merge
origin/master references remote server branch and tries to stay in sync
repo
checkout
commit
Local
Stage
add
Working directory

## How do I link my local repo to a remote repo?

```
git remote add <alias> <URL>
```

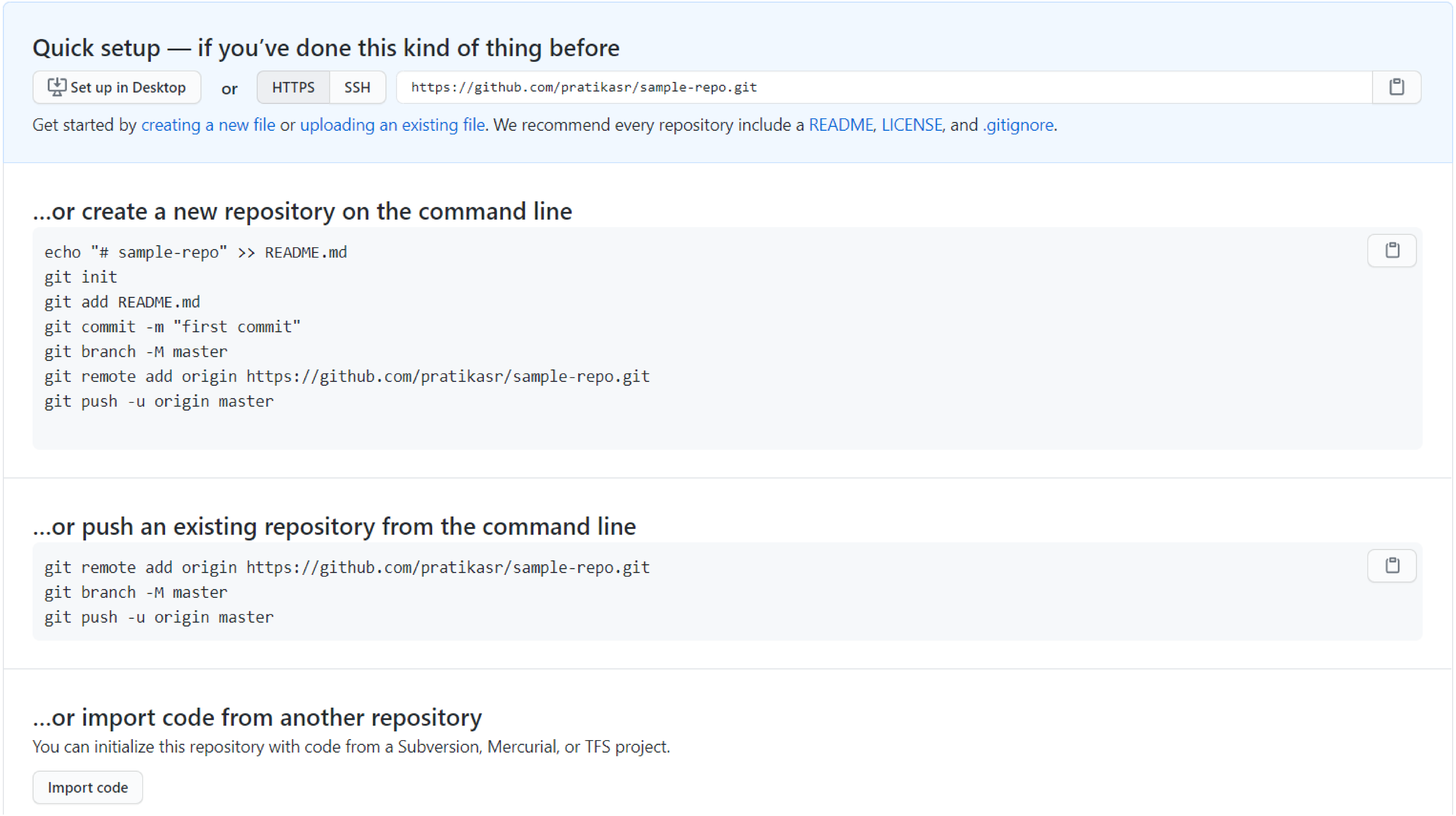
**Note: This just establishes a connection...no files are copied/moved**
**Note: Yes! You may have more than one remote linked to your local directory!**

## create a new repository on the command line

```
echo "# sample-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/pratikasr/sample-repo.git
git push -u origin master
```

## push an existing repository from the command line

```
git remote add origin https://github.com/pratikasr/sample-repo.git
git branch -M master
git push -u origin master
```



## Quick setup — if you've done this kind of thing before

https://github.com/pratikasr/sample-repo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

## ...or create a new repository on the command line

```
echo "# sample-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/pratikasr/sample-repo.git
git push -u origin master
```

## ...or push an existing repository from the command line

```
git remote add origin https://github.com/pratikasr/sample-repo.git
git branch -M master
git push -u origin master
```

## ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.