VERSION CONTROL

With Git

WHY VERSION CONTROL?

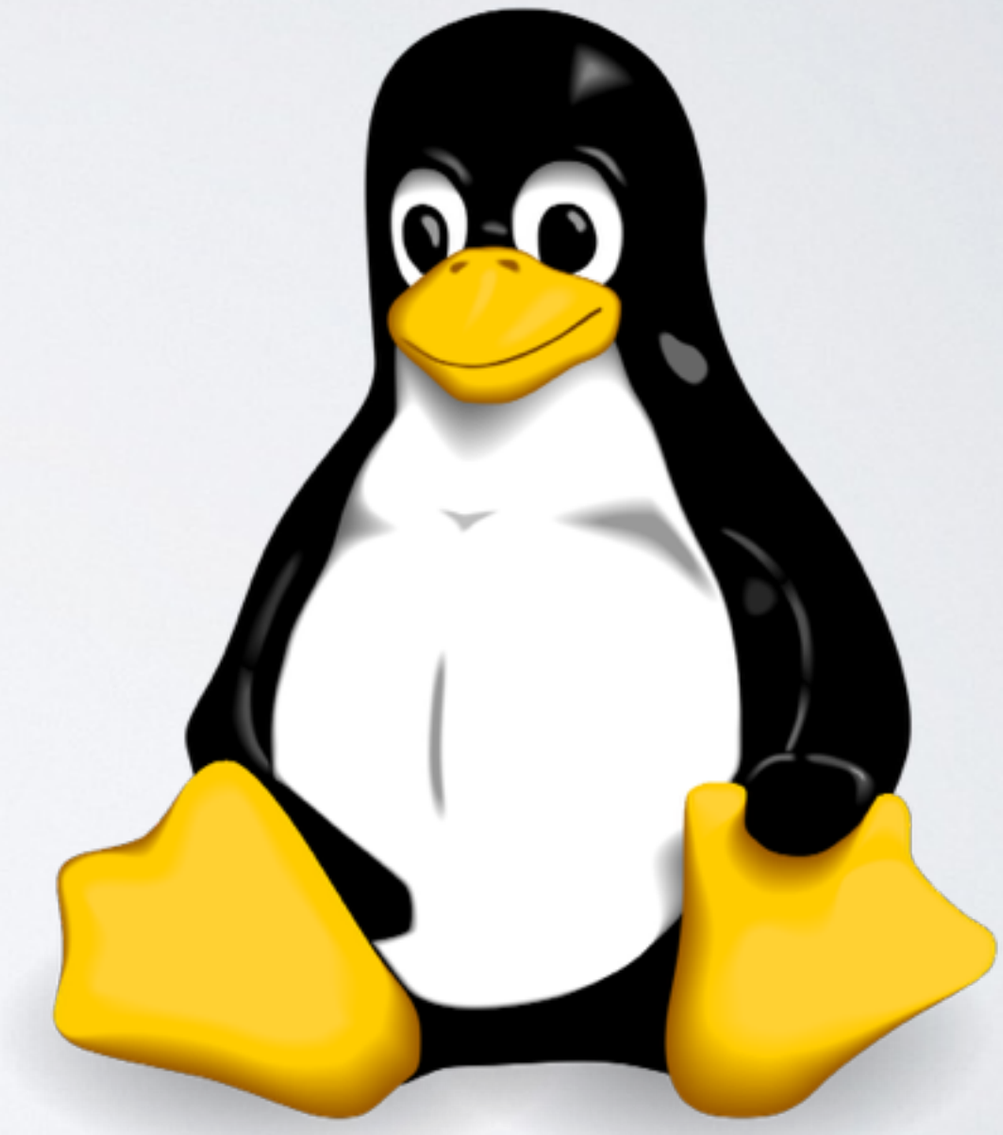
- Software changes (frequently)
- Most software is developed in teams
- Agile development methods thrive on revision

HISTORICAL METHODS

- Basic Version Control: Archiving, RCS
- Centralized Version Control: CVS, Subversion
- Distributed Version Control: Git, Mercurial, Bazaar

GIT

- Written by Linus Torvalds in 2005 for use with the Linux kernel
- Completely distributed
- Very fast
- Allows for many contributors and parallel work



INSTALLING GIT

OSX

- Install Xcode Developer Tools from App Store

Linux

- Install from package manager:

Fedora / Red Hat

- sudo yum install git-core

Ubuntu / Debian

- sudo apt-get install git

INSTALLING GIT

WINDOWS

- Visit the Git website: http://git-scm.com/downloads

GETTING STARTED

- Initial configuration

- Telling git who we are:

  ```
  git config --global user.name “John Doe”
  git config --global user.email jdoe@ryerson.ca
  ```

- Extra configuration (optional)

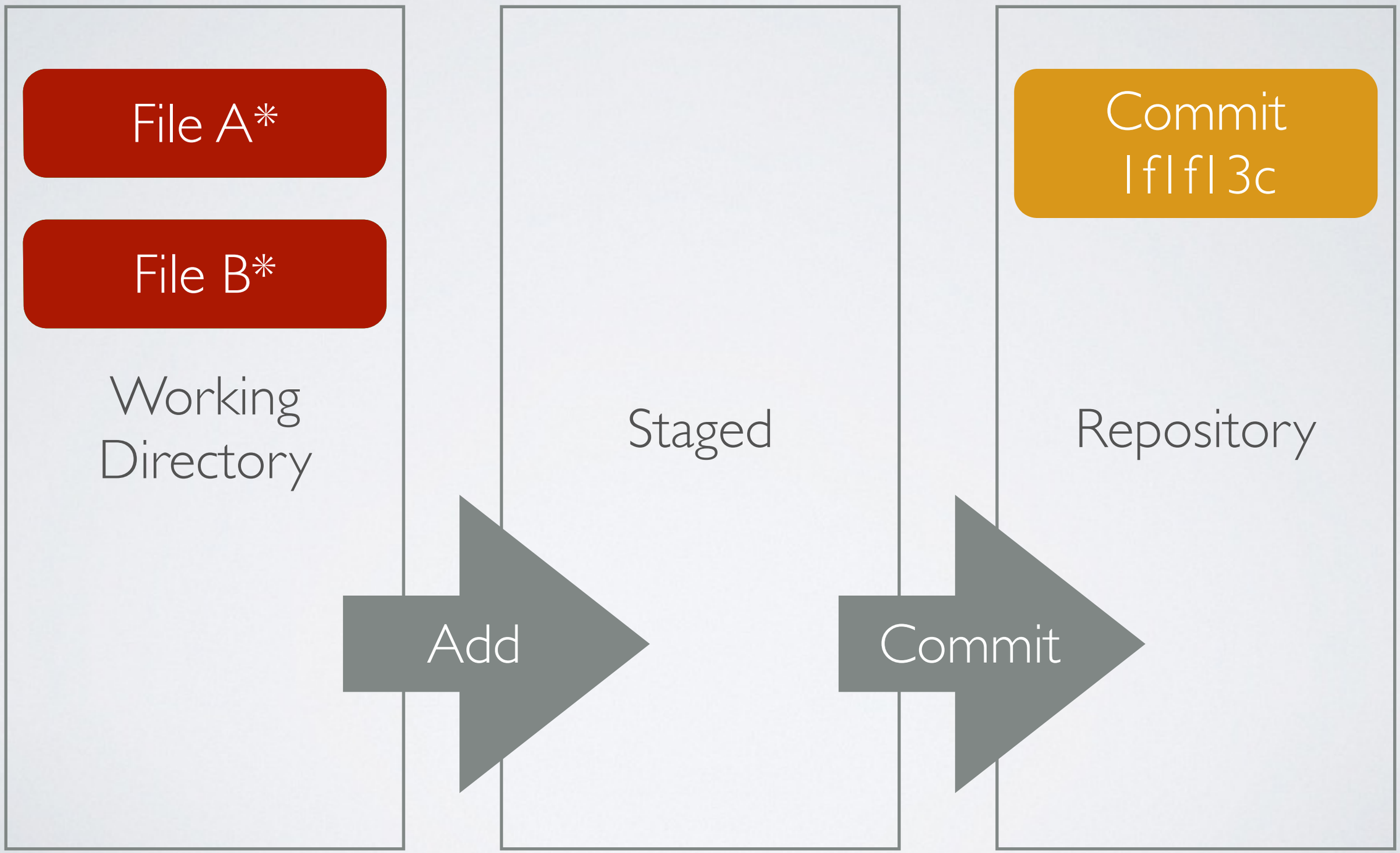
  ```
  git config --global  core.editor vim
  git config --global  merge.tool vimdiff
  ```

STARTING A REPO

- Create a folder and initialize Git:

```
mkdir myfirstrepo
cd myfirstrepo
git init
```

THE GIT MODEL



INITIAL COMMIT

- Create a file:

  ```
  echo “Hello World” >> readme.txt
  git status
  ```

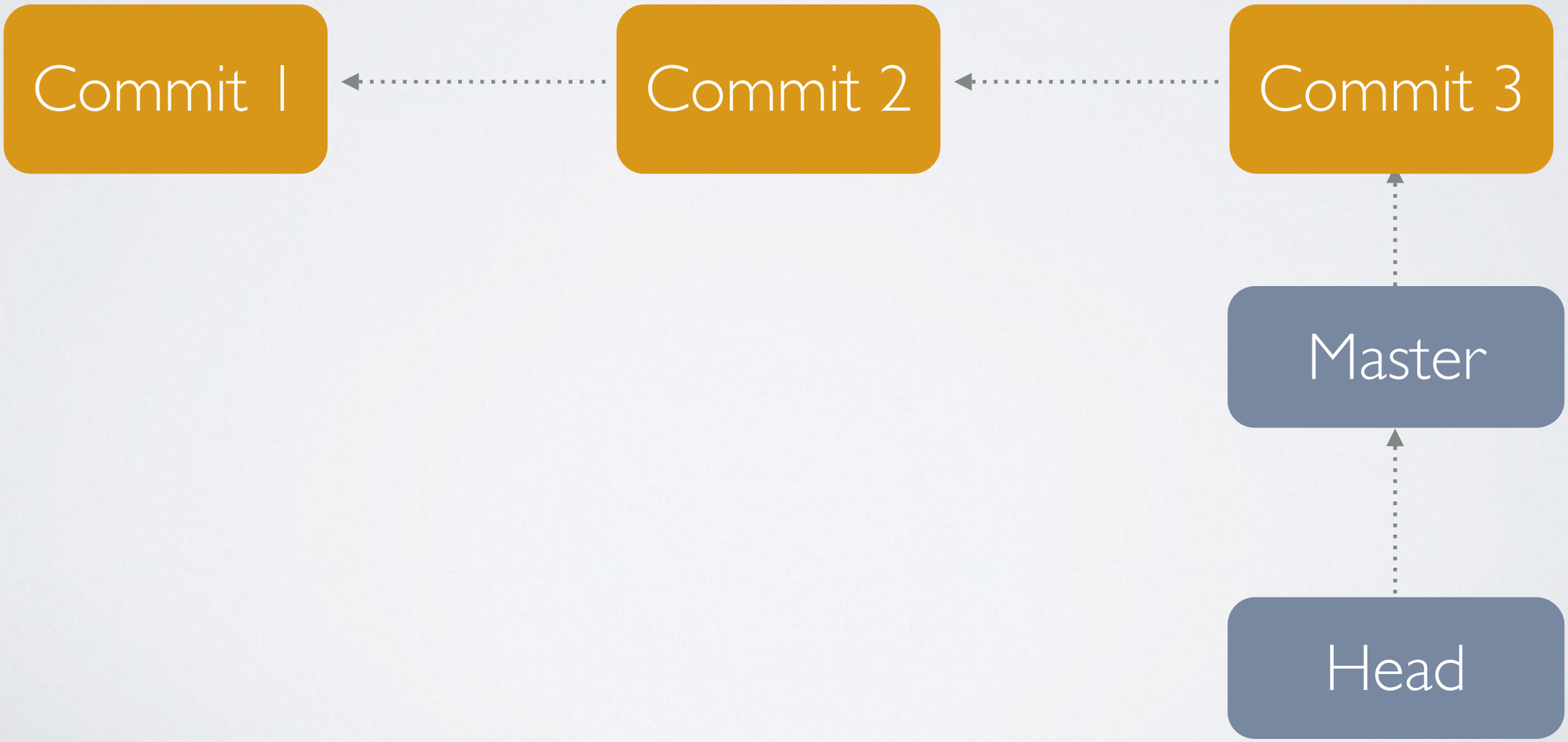
- Add to staging area:

  ```
  git add readme.txt
  git status
  ```

- Commit the staging area:

  ```
  git commit
  git status
  ```

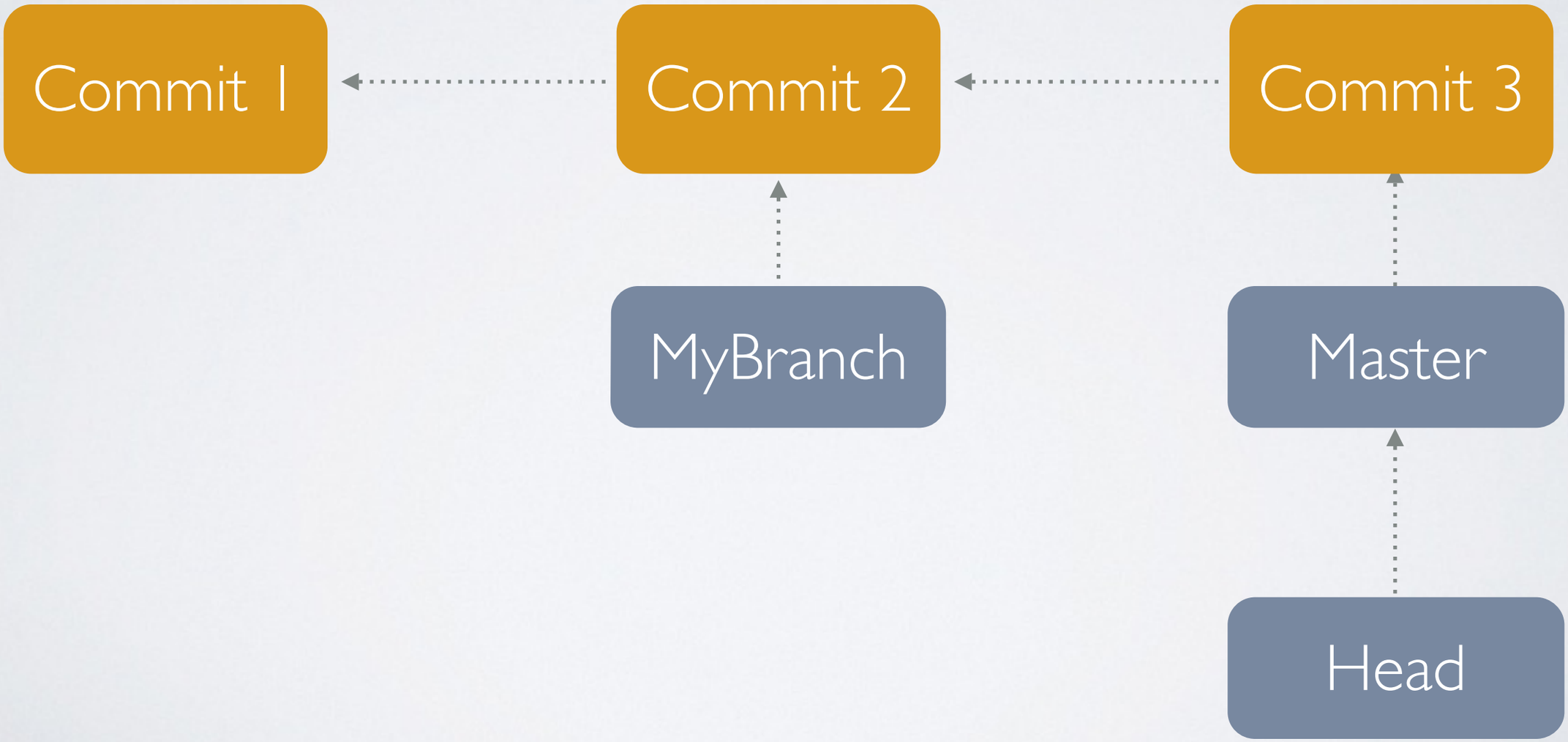
THE HISTORY



BRANCHING

- Git allows us to create alternate paths in our history called branches
- The default branch is called ‘master’
- This enables very agile workflows

THE HISTORY REVISED



WORKING WITH BRANCHES

- Creating branches

  git branch [branch-name]

- Switching to a branch

  git checkout [branch-name]

MERGING

- To merge a branches code into our working branch

  `git merge [branch-name]`

- Git tries to merge the code using a three-way merge

- For anything Git can't automerge, it leaves you to merge the changes and commit the result

- To view the result graphically:

  `git log --graph`

SHARING CODE

- Cloning a repository:

  git clone [path-to-repository]

- A simple example repository:

  git clone https://github.com/adpopescu/north-american-octo-sansa.git

- Viewing the history:

  git log

REMOTES

- Remotes are non-local repositories associated with your repo

- Git creates a remote called origin when you clone a repository:

  git remote

- You can add remotes:

  git remote add <name> <url>

PULLING AND PUSHING

- We can update our working branch from a remote by doing:

  `git pull [remote-name] [remote-branch]`

- We can update a remote branch from our local branch using:

  `git push [remote-name] [remote-branch]`

REFERENCES

- Git Website: http://git-scm.com

- Pro Git