Lecture 2
Making Simple Commits

Sign in on the attendance sheet!

credit: https://xkcd.com/1296/

Review of Last Lecture

- git init – creates a git repo in the current directory
- git clone <git url> – copies the remote git repo into the current directory
- git log [--oneline] – lists all commits in the git repo, starting with the most recent one
- git help <command>, git <command> --help, man git <command> – brings up the man help page for the git command

Leftover topics from last lecture

- The .git folder

The .git folder

- Every git repository has a .git directory in the toplevel project directory
- This is where all git commit objects and metadata are stored
- **Don’t delete it!** Doing so deletes the repository
- Folders starting with a dot are hidden on UNIX

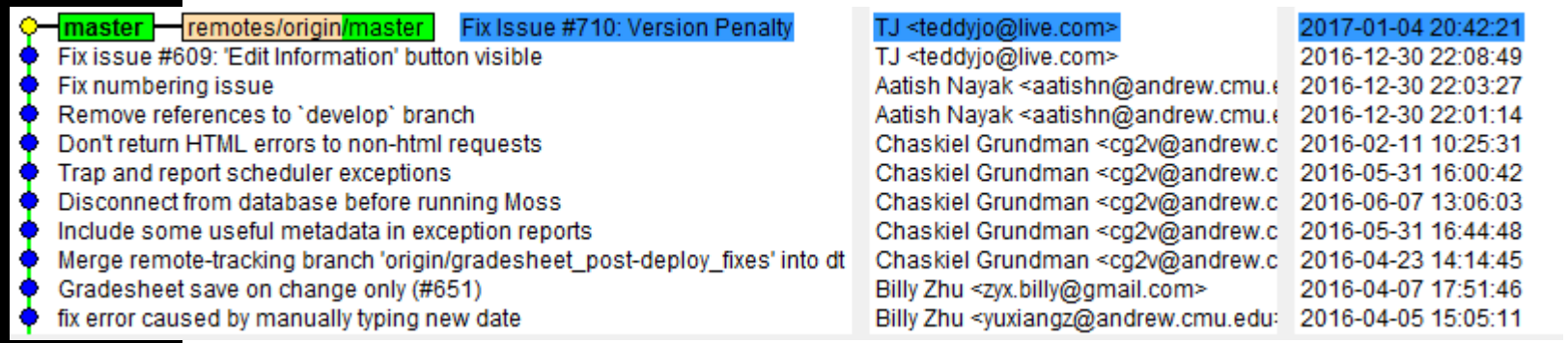
```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/www (master)
$ ls -a
.  ..  .git  css  f16  homework  index.html  lecturenotes  slides

Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/www (master)
$ ls .git
COMMIT_EDITMSG  config       hooks  info  objects      refs
HEAD            description  index  logs  packed-refs
```

Today: The Git Commit Workflow

- Review: git log
- git diff
- git status
- git add
- git commit
- git show

From Last Time: git log

```
commit 5a4b0ebb2b3b4ba5eb8cf13395ff83a88a700c09
Author: TJ <teddyjo@live.com>
Date:   Wed Jan 4 20:42:21 2017 -0500

    Fix Issue #710: Version Penalty

commit 135867b4b48fc30c591bbf0bfb4b506da2822803
Author: TJ <teddyjo@live.com>
Date:   Fri Dec 30 19:08:49 2016 -0800

    Fix issue #609: 'Edit Information' button visible

commit 652d0f9fe06b290ef4af11cb83485fb33df41b47
Author: Aatish Nayak <aatishn@andrew.cmu.edu>
Date:   Fri Dec 30 22:03:27 2016 -0500

    Fix numbering issue

commit 2eae45f09bf3267b1cf084ca452d35d01ecadb15
Author: Aatish Nayak <aatishn@andrew.cmu.edu>
Date:   Fri Dec 30 22:01:14 2016 -0500

    Remove references to `develop` branch

commit 1de15123f5175e9c5cfe2931871b91a77fee964f
Author: Chaskiel Grundman <cg2v@andrew.cmu.edu>
Date:   Thu Feb 11 10:25:31 2016 -0500

    Don't return HTML errors to non-html requests

    API requests (e.g. annotations) should not try to display
    donkey kong when they fail, since that just triggers another
    error

commit 467750d709e10e91c627e6f95a142b43329a0250
Author: Chaskiel Grundman <cg2v@andrew.cmu.edu>
Date:   Tue May 31 16:00:42 2016 -0400

    Trap and report scheduler exceptions

    fork with a block does not seem to propagate exceptions in the subprocess
    to rescue blocks outside the fork block, so exceptions inside scheduler
    modules were not being logged. Trap them inside the fork block, also trap
    ScriptError exceptions, and send the subprocess exceptions to
    ExceptionNotifier.
```



Also try `git log --oneline`:

```
652d0f9 Fix numbering issue
2eae45f Remove references to `develop` branch
1de1512 Don't return HTML errors to non-html requests
467750d Trap and report scheduler exceptions
8245996 Disconnect from database before running Moss
e6bc057 Include some useful metadata in exception reports
e715f07 Merge remote-tracking branch 'origin/gradesheet_post-deploy_fixes' into dt
0426a88 Gradesheet save on change only (#651)
c4af599 fix error caused by manually typing new date
da91ba8 Merge pull request #643 from autolab/gradesheet_get_feedback_on_demand
812ec17 Merge pull request #637 from autolab/searchable_dropdown_when_creating_subm
72c532d Merge pull request #636 from autolab/create_extension_with_new_due_date
8778100 Safely open local_submit files
f1f448d Autograde_done url should use https
```

What is `2eae45f`?

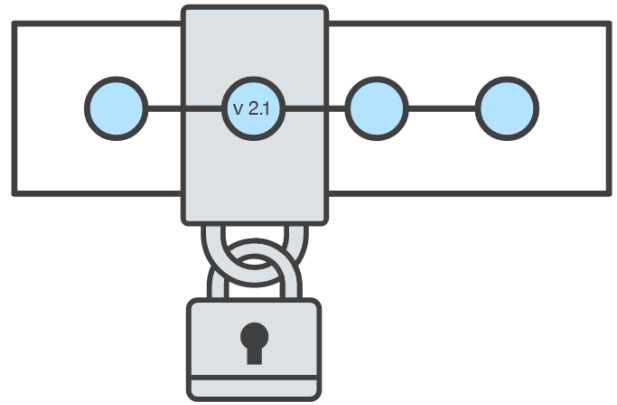
- Commits are uniquely represented by SHA-1 hashes
- The first 6-7 characters of a hash are usually enough to identify it uniquely from all the other commits in the repository
- This is called the **short hash**

Okay, so what is a commit?

1. A **snapshot** of all the files in a project at a particular time.
2. A **checkpoint** in your project you can come back to or refer to.

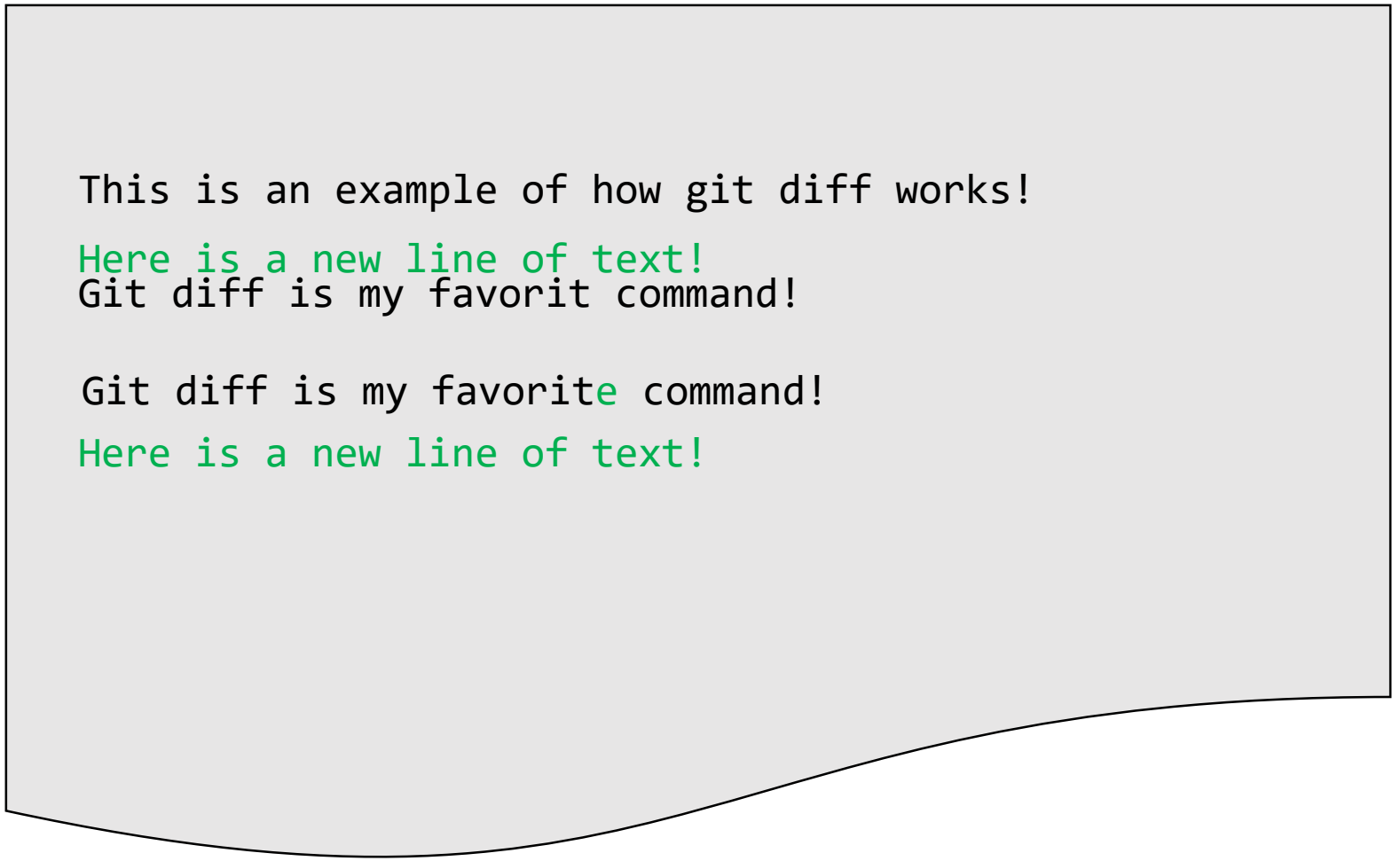
Anything else?

3. The **changes** a commit makes over the previous commit



Git Diff

demo.txt

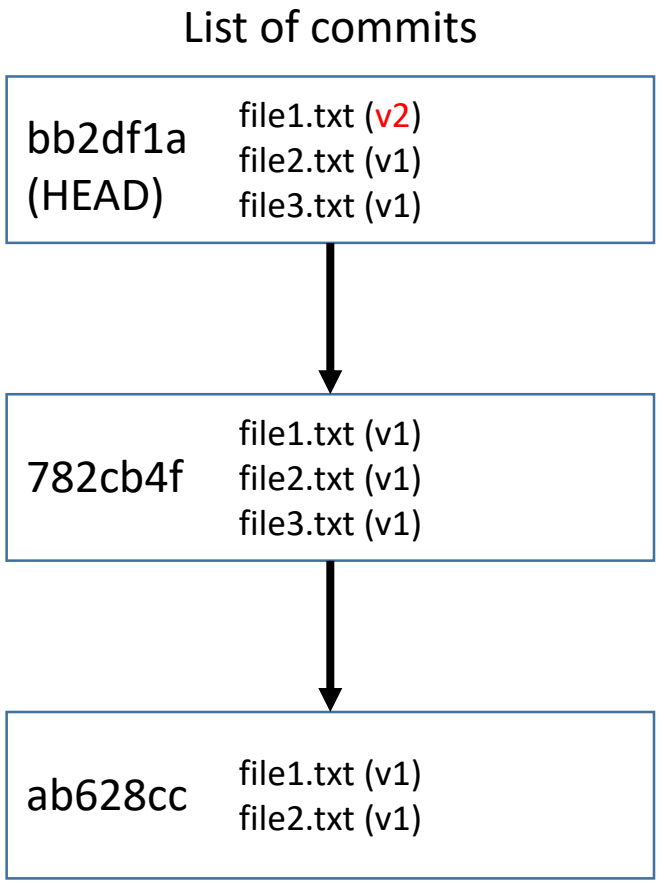


Git Diff

```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/testing (master)
$ git diff
diff --git a/demo.txt b/demo.txt
index 4fd054e..ff58225 100644
--- a/demo.txt
+++ b/demo.txt
@@ -1,3 +1,5 @@
 This is an example of how git diff works!
+Here is a new line of text!

-Git diff is my favorit command!
+Git diff is my favorite command!
+Here is another line of text!
```

Commits: Revisited

- Editing a file takes its state from 1 particular snapshot to the next
- When we edit a file, we can see it as a set of changes (a “diff”) from the snapshotted state of that file
- Commits bundle up sets of changes to a list of files



git show <commit hash>

```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/Autolab (master)
$ git show 13586
commit 135867b4b48fc30c591bbf0bfb4b506da2822803
Author: TJ <teddyjo@live.com>
Date:   Fri Dec 30 19:08:49 2016 -0800

    Fix issue #609: 'Edit Information' button visible

diff --git a/app/views/course_user_data/show.html.erb b/app/views/course_user_data/show.html.erb
index 942e9e3..9ecaa0a 100755
--- a/app/views/course_user_data/show.html.erb
+++ b/app/views/course_user_data/show.html.erb
@@ -11,4 +11,6 @@
   <li><b>Course Average Tweak</b> of <%=raw tweak(@requestedUser.tweak) %></li>
   <% end %>
 </ul>
+<% if @cud.instructor? then %>
 <%= link_to raw('<span class="btn primary">Edit Information</span>'), edit_course_course_user_datum_p
+<% end %>
diff --git a/app/views/course_user_data/user.html.erb b/app/views/course_user_data/user.html.erb
index a2ae9e3..be1513a 100755
--- a/app/views/course_user_data/user.html.erb
+++ b/app/views/course_user_data/user.html.erb
@@ -12,4 +12,6 @@
   <li><b>Course Average Tweak</b> of <%=raw tweak(@requestedUser.tweak) %></li>
   <% end %>
 </ul>
+<% if @user.instructor? then %>
 <%= link_to raw('<span class="btn primary">Edit Information</span>'), edit_course_user_path(@course,
+<% end %>
(END)
```

The Git Commit Workflow: Edit

Working Directory

file1.txt (v2)
file2.txt (v1)
file3.txt (v2)

Make changes to files
`vim file1.txt file3.txt`

List of Changes

- In file1.txt: add the line “here is a new line!” between lines 3 and 4
- In file3.txt: delete line 27

The Git Commit Workflow: Add

Working Directory

file1.txt (v2)
file2.txt (v1)
file3.txt (v2)

Add the current differences
`git add file1.txt file3.txt`

List of Changes

In file1.txt: add the line “here is a new line!” between lines 3 and 4

In file3.txt: delete line 27

Staging Area

The Git Commit Workflow: Commit

List of Changes

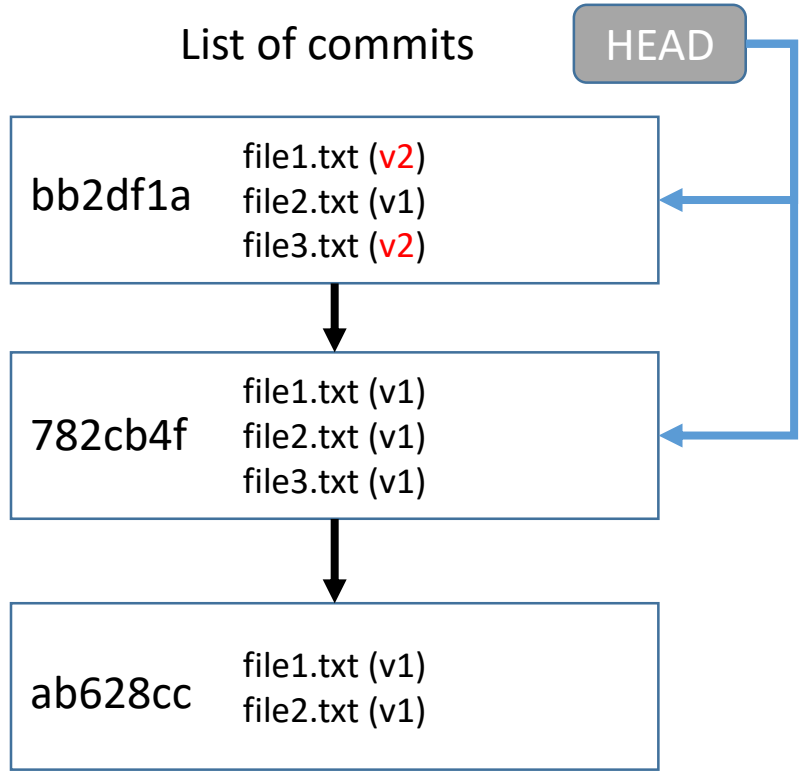
Staging Area

In file1.txt: add the line “here is a new line!” between lines 3 and 4

In file3.txt: delete line 27

Commit the currently staged differences

```
git commit –m "fixed bug in file1 and file3"
```



git commit

Example use:

git commit

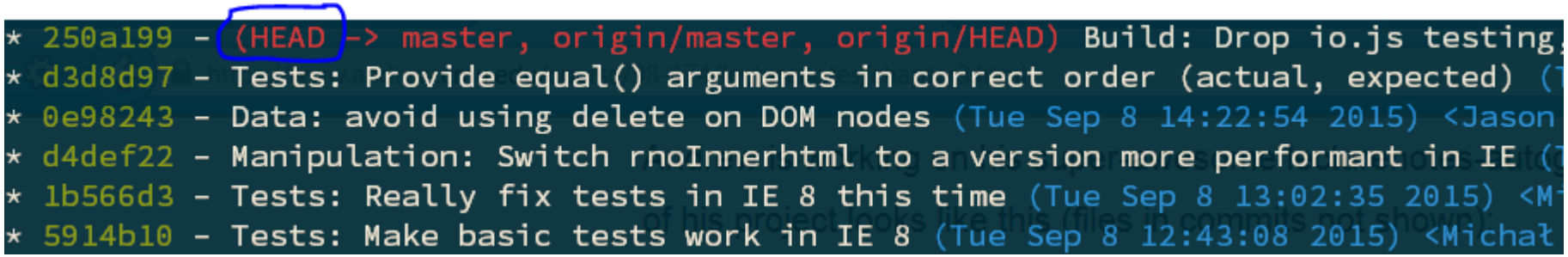
(or)

git commit –m “commit message goes here”

- Creates a commit out of a snapshot of the staging area, and updates HEAD.

Aside: commit HEAD

- The “most recent commit” has a special name: HEAD



Good commit messages

- Good:

Build: Don't install jsdom3 on Node.js 0.10 & 0.12 by default

- Bad:

bugfix lol get rekt

http://whatthecommit.com

git status

Shows files differing between the staging area and the working directory (i.e. unstaged changes), the staging area and HEAD (i.e. changes ready to commit), and untracked files

```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/testing (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   demo.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   demo.txt
        modified:   one.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        new_file.txt
```

git diff

Example use:

(show unstaged changes)

git diff

(show staged changes)

git diff --cached

- Shows unstaged changes or staged changes

git show

Example use:

git show [commit hash (default is HEAD)]

- Shows the changes in the specified commit

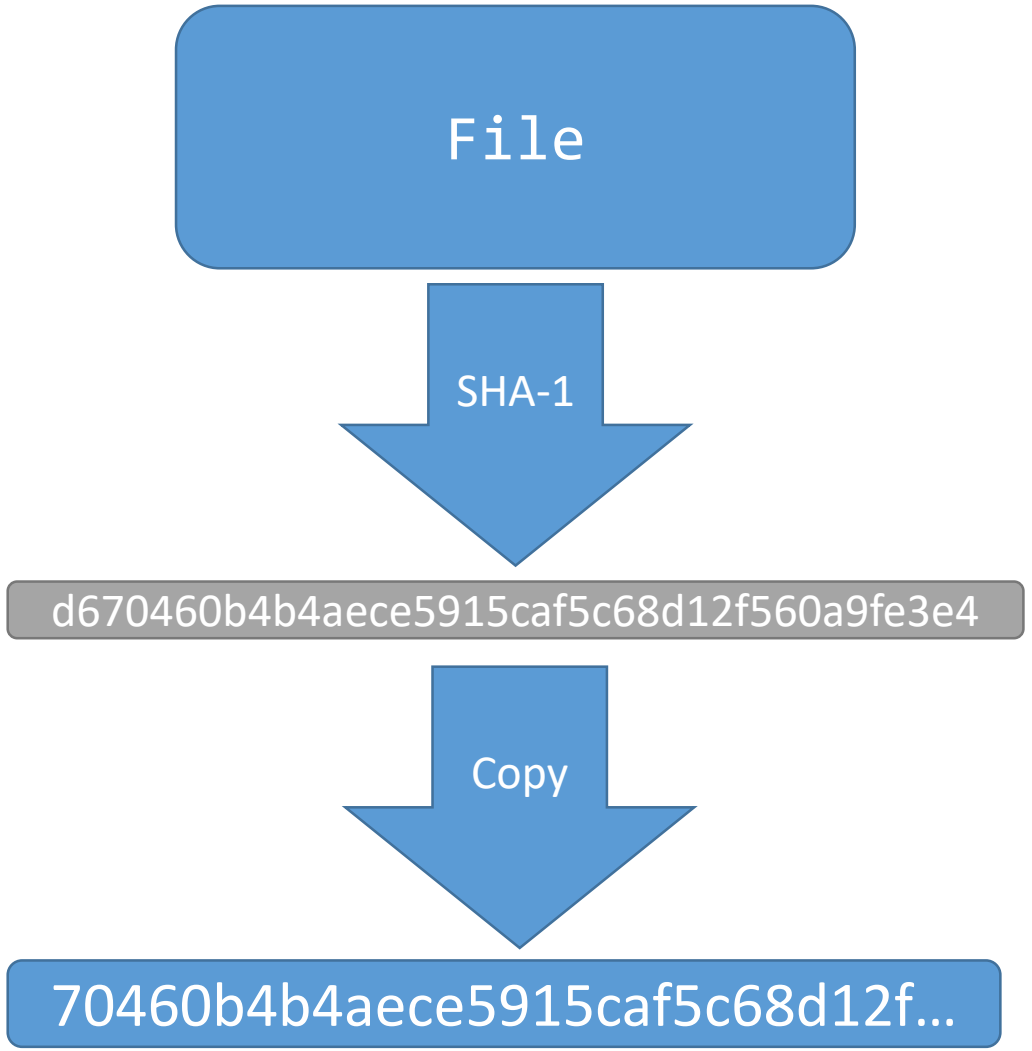
Activity: Practicing Making Commits

- Make a new folder, and create a new git repository inside.
- Create a file called “me.txt”. Inside, write your name and hometown.
- Make a commit with this new file.
- Make a new file called “neighbors.txt”.
- Now, find 3 people sitting near you. For each person,
  - Find out their name and hometown, and put it in neighbors.txt.
  - Check the output of git status and git diff and verify it makes sense.
  - git add neighbors.txt
  - Check the output of git status and git diff and verify it makes sense.
  - Commit the change.
  - Check the output of git show and verify it makes sense.

How Git Add+Commit Actually Works

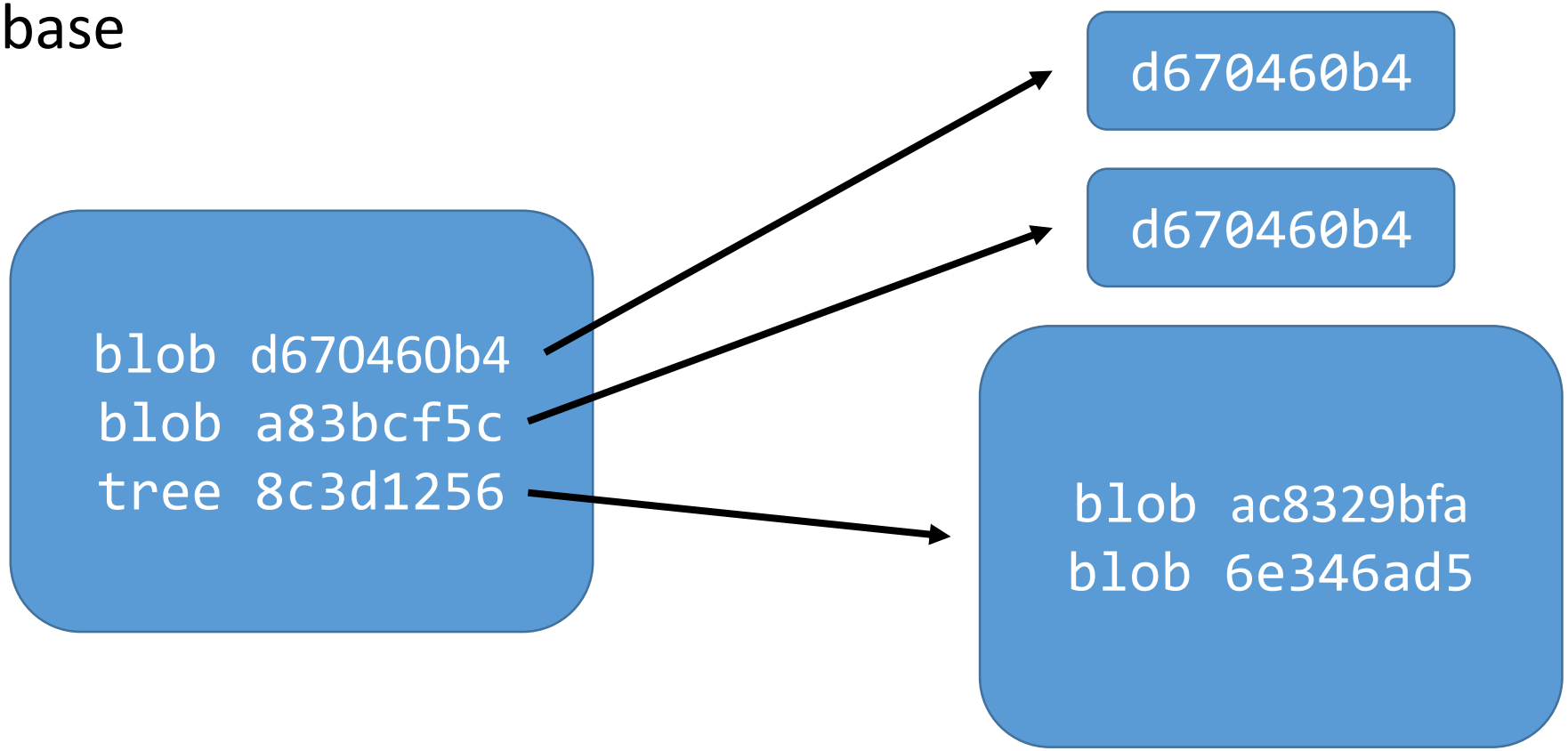
Step 1: Store a file in the git database

- Hash the file
- Store a copy of the file in .git/objects/<hash>



How Git Add+Commit Actually Works

Step 2: Store the state of a directory tree in the git database



How Git Add+Commit Actually Works

Step 3: Store a commit object with a reference to the top-level tree in the git database