

600.120 Intermediate Programming, Spring 2017

Misha Kazhdan

Announcements

- Assignment 0 was released on Wednesday (Piazza).
 - Due this coming Tuesday night
 - Solo, written assignment

Outline

- Working with `git`
- Tarring a directory
- Transferring files

Working with `git`

Last time:

We cloned the existing course repo `cs120sp17-public` into the `cs120` directory, and then copied its contents to `cs120-personal`.

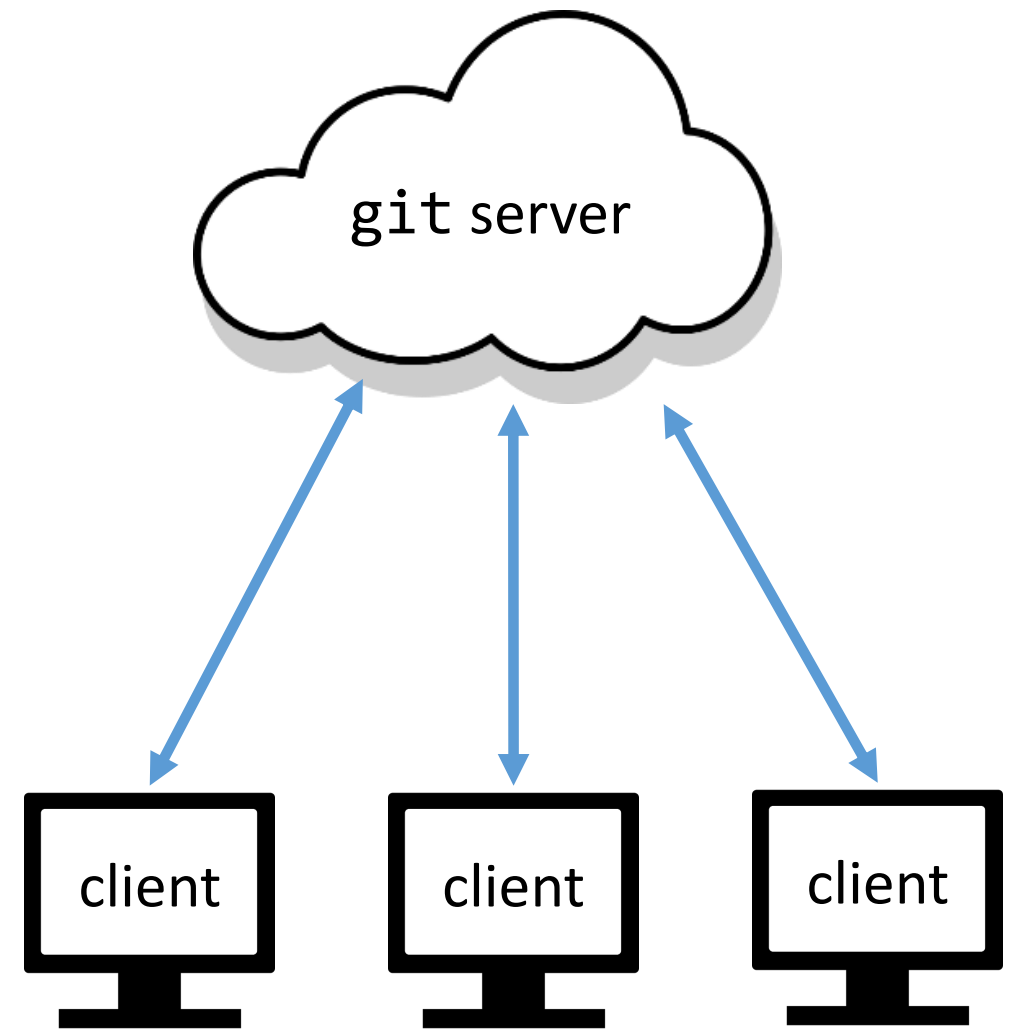
Working with `git`

This time:

We will create a repo from scratch and copy its contents between machines.

Repo (why)?

- A copy on a remote `git` server
- Supports collaboration / sharing
- Version control
 - Stores the history as a collection of concise, semi-automatically annotated, “snapshots” of the repo



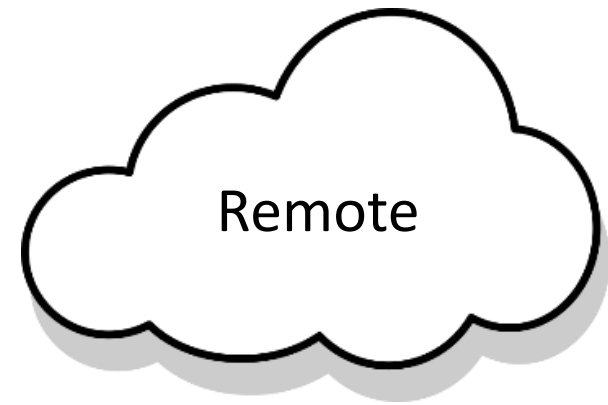
Working with `git`

This time:

We will create a repo from scratch and copy its contents between machines.

Repo (what)?

- A copy of the entire history of the files/project
 - A *local* copy is stored on your (client) machine.
 - A *remote* copy is stored on the server (e.g. `bitbucket.org` or `github.org`)



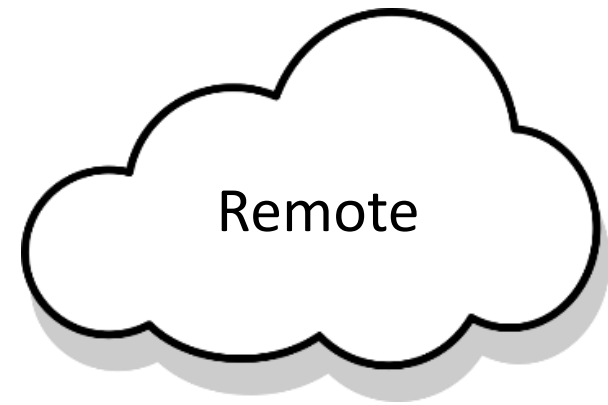
Working with `git`

This time:

We will create a repo from scratch and copy its contents between machines.

Repo (how)?

- The repo directory contains:
 - Local Repo: A copy of the entire repository (in the `.git` subdirectory)
 - Working copy: A replica of all the “latest” files.

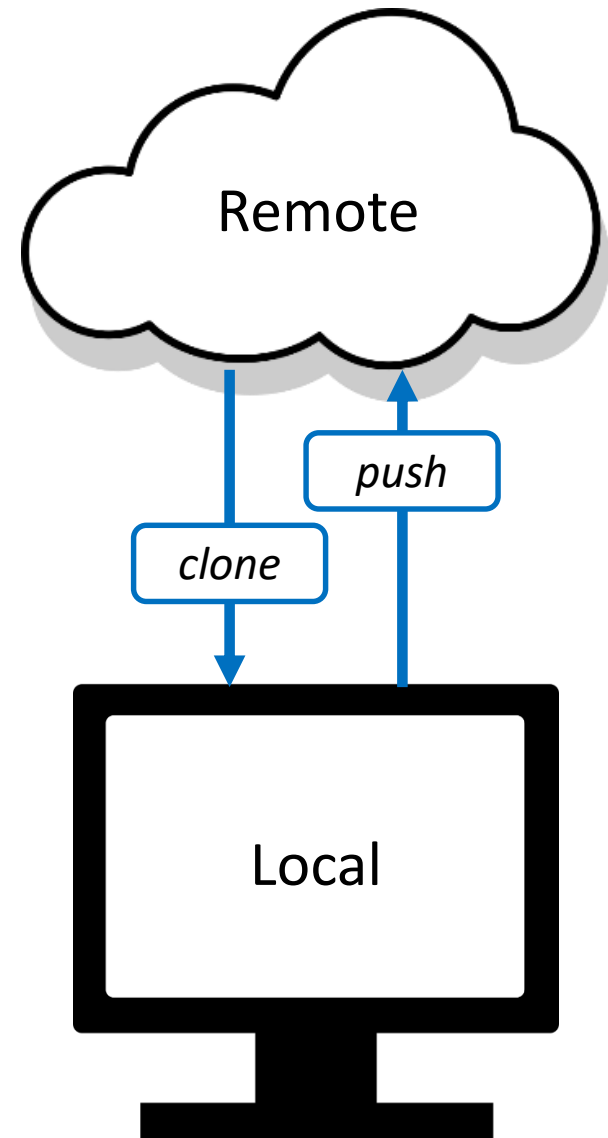


Working with `git`

This time:

We will create a repo from scratch and copy its contents between machines:

- Create an account on `bitbucket.org`
- Create an empty repository
- Clone the repository to our `cs120` folder
- Add / modify local files
- Push the edited files in the repo.

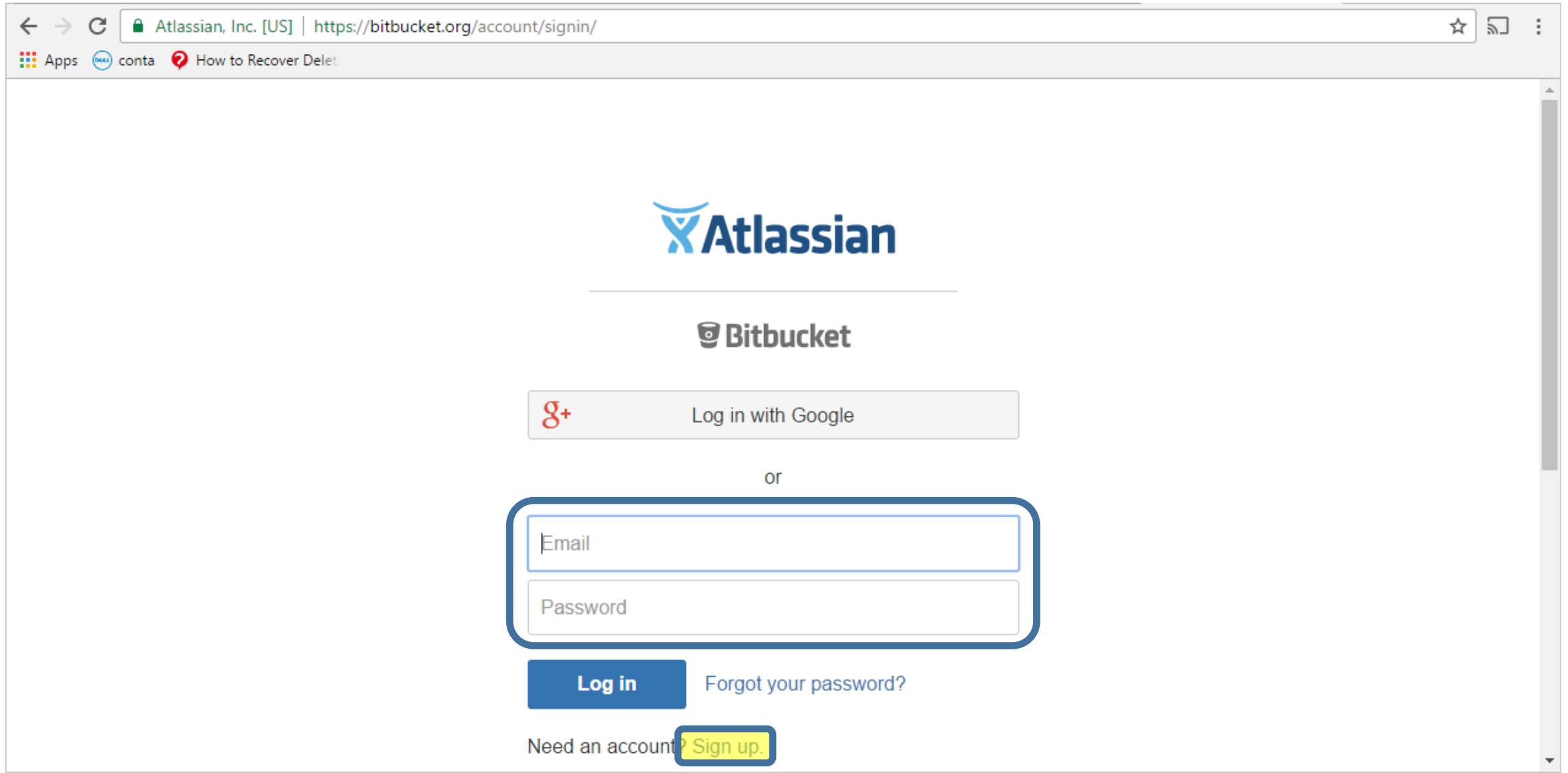


Working with `git`

Before we start:

- We need to tell the `git` system (on ugradx) a bit about ourselves:
 - What name do we want the system to know us by:
`git config --global user.name "<your name>"`
(quotes are required if there is a space in <your name>)
 - What email do we want the system to know us by:
`git config --global user.email <your email>`

Create an account / log in



A screenshot of a web browser showing the Bitbucket login page. The browser's address bar displays the URL <https://bitbucket.org/account/signin/>. The page features the Atlassian logo at the top, followed by the Bitbucket logo. Below the logos is a "Log in with Google" button. Underneath this is the word "or". A blue rounded rectangle highlights the login form, which contains an "Email" input field and a "Password" input field. Below the form is a blue "Log in" button and a link for "Forgot your password?". At the bottom, the text "Need an account?" is followed by a yellow "Sign up." button.

Atlassian

Bitbucket

g+ Log in with Google

or

Email

Password

Log in [Forgot your password?](#)

Need an account? [Sign up.](#)

Create a new repository

The screenshot shows the Bitbucket dashboard interface. At the top, the navigation bar includes the Bitbucket logo, 'Teams', 'Projects', 'Repositories', and 'Snippets'. A search bar on the right says 'Find a repository...'. Below the navigation bar, the 'Repositories' dropdown menu is open, showing 'RECENTLY VIEWED' and two options: 'Create repository' (highlighted with a yellow box) and 'Import repository'. The main content area on the left has a 'Dashboard' section with an 'Overview' tab selected. It features a message: 'Everything's awesome! All your pull requests and reviews are done and dusted.' with a 'View all pull requests' link. On the right, a 'Repository' list shows five items: 'Yotam slides', 'cs120sp17-staging-sara-misha', 's4 class resources', 'ParameterizingEvolvingMeshes', and 'Yotam class material', each with a lock icon. The browser's address bar shows the URL 'https://bitbucket.org/dashboard/overview#'.

Atlassian, Inc. [US] | <https://bitbucket.org/dashboard/overview>

Apps conta How to Recover Deleted

Bitbucket Teams Projects Repositories Snippets Find a repository... ?

Dashboard

Overview Repositories Pull requests

Everything's awesome!
All your pull requests and reviews are done and dusted.

[View all pull requests](#)

RECENTLY VIEWED

Create repository
Import repository

Repository

	Yotam slides Last updated 8 hours ago	
	cs120sp17-staging-sara-misha Last updated 8 hours ago	
	s4 class resources Last updated yesterday	
	ParameterizingEvolvingMeshes Last updated Jan 26 2017	
	Yotam class material Last updated Jan 25 2017	

<https://bitbucket.org/dashboard/overview#>

Clone the repository

The screenshot shows the Bitbucket web interface for a repository named 'cs120-personal' by user 'mkazhdan'. The left sidebar contains navigation links: Overview (selected), Source, Commits, Branches, and Pull requests. The main content area is titled 'Overview' and features a 'Clone' button with a download icon. Below the 'Clone' button, there is a text box containing the command `git clone https://mkazhdan@bitbucket`, which is highlighted with a yellow background. To the right of the command box, there is a link to 'Learn how to clone a repository.' Below this, there is a button labeled 'Clone in SourceTree' and a description: 'Atlassian SourceTree is a free Git and Mercurial client for Windows.' To the right of the 'Clone in SourceTree' button, there is an illustration of a laptop with a document icon and a code icon. Below the illustration, there is a section titled 'Put some bits in your bucket' with the text 'Add some code or content and start bringing your ideas to life. [Learn how](#)'. At the bottom, there is a section titled 'Get started the easy way' with the text 'Creating a README or a .gitignore is a quick and easy way to get something into your repository.'

Atlassian, Inc. [US] | <https://bitbucket.org/mkazhdan/cs120-personal>

Apps | conta | How to Recover Delet

Bitbucket Teams Projects Repositories Snippets Find a repository...

cs120-personal

ACTIONS

- Clone
- Create branch
- Create pull request
- Compare
- Fork

NAVIGATION

- Overview
- Source
- Commits
- Branches
- Pull requests

mkazhdan / cs120-personal

Overview

HTTPS `git clone https://mkazhdan@bitbucket`

Need help cloning? Learn how to [clone a repository](#).

Clone in SourceTree

Atlassian SourceTree is a free Git and Mercurial client for Windows.

Put some bits in your bucket

Add some code or content and start bringing your ideas to life. [Learn how](#)

Get started the easy way

Creating a README or a .gitignore is a quick and easy way to get something into your repository.

<https://bitbucket.org/mkazhdan/cs120-personal#clone>

Clone the repository

The image shows a Bitbucket web interface for a repository named 'cs120-personal' by user 'mkazhdan'. The left sidebar contains 'ACTIONS' (Clone, Create branch, Create pull request, Compare, Fork) and 'NAVIGATION' (Overview, Source, Commits, Branches, Pull requests). The main area shows the 'Overview' page with a 'Clone' button and the repository URL: `https://mkazhdan@bitbucket.org/mkazhdan/cs120-personal.git`.

Overlaid on the bottom right is a terminal window titled 'misha@ugradx:~/cs120'. It shows the command `git clone https://mkazhdan@bitbucket.org/mkazhdan/cs120-personal.git` being entered at the prompt `[misha@ugradx ~/$]`.

Working with `git`

Once you have a repository, you can:

- commit changes you've made locally (to create a restore point)
- push those back up to the git server (to synchronize your local repo)

Note:

- `git` does not know what changes you want to commit.
- You will have to let it know explicitly.

Working with `git`

To run a git command you type “`git`”, followed by the command, followed by the parameters:

- `git clone <address>`:
 - clones a repository into the local directory
- `git pull`:
 - fetches the most recent version of the repo from the server and merges it with the repo (and working copy) on the local machine
- `git status`:
 - display the current status of the repository
- `git log`:
 - display the history of changes (commits)

Working with `git`

To run a git command you type “`git`”, followed by the command, followed by the parameters:

- `git add <file name>`:
 - add the specified file to the list of files that you will be committing (do this to add a file to the repo or if you’ve changed an existing file)
- `git commit -m “<commit message>”`:
 - commit the changes (locally), including a brief message describing the changes.
- `git commit -am “<commit message>”`:
 - like `git commit -m “<commit message>”` but automatically adds any files that have been modified (not added) since the last commit
- `git push`:
 - push your local repo back to the server

Working with `git`

Note that you cannot modify a repo directly using standard `mv` or `rm`; all interactions are via the `git` command:

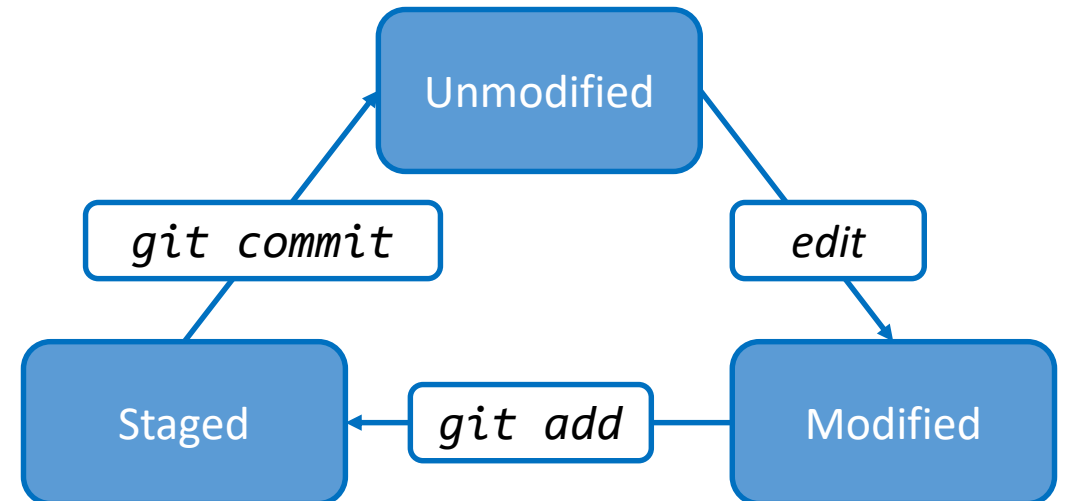
- `git mv <source file> <target file>`:
 - rename a file
- `git rm <file>`:
 - remove a file (delete it)

For a more complete list, see:

<https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

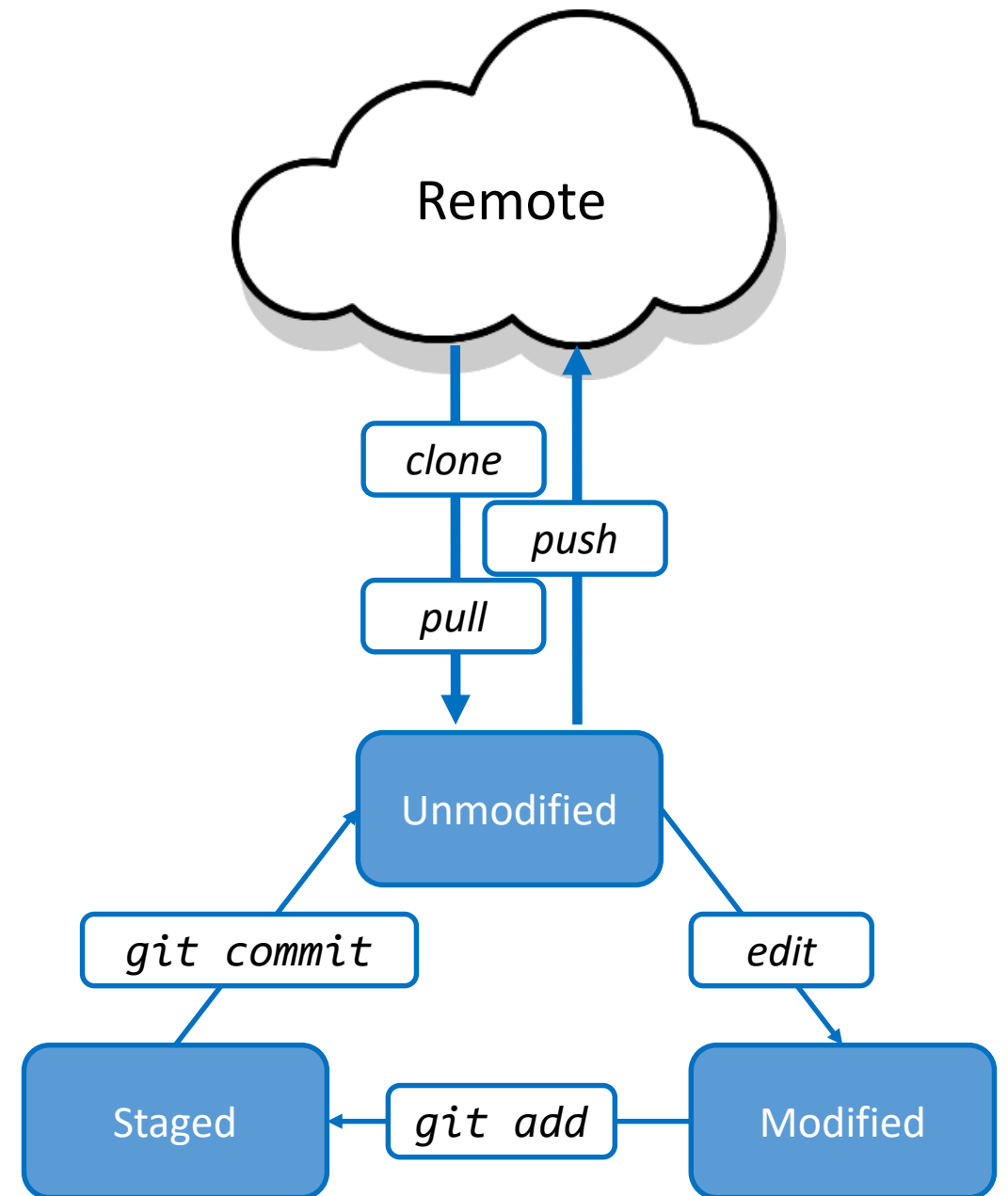
Working with `git`

- Files in your working copy (you `git add`'ed them) are *tracked* and can be in one of several states:
 - Unmodified* (same as copy in local repo)
 - Modified* (different from copy in local repo but not yet staged)
 - Staged* (next `git commit` will update the repo)



Working with `git`

- The `git push` / `pull` / `clone` commands synchronize the local repository with the remote one.
- These commands synchronize the local/global repos, so be sure that your working copy matches the local repo before calling them (i.e. all files are unmodified)



Working with `git`

- Files that are *not* yet part of your project are *untracked*
 - When you create a new file; it's *unstaged* until you `git add` it
 - This is true even if the file is in the directory containing your working copy
 - However, `git` will notice it, and it will appear as unstaged if you `git status`
- Some untracked files are files that we want `git` to ignore because we'll never want to include them in the remote repo
 - Tell `git` to ignore a file by adding it to `.gitignore` file
 - Good candidates for ignoring might be `a.out`, `gitlog.txt`

Working with `git`

Workflow Suggestions:

- Start each session with `git pull`, to ensure your local copy is up-to-date
- After you complete work on a small task, `git commit` it
- Include a message with every commit to explain what changes you committed (use `-m`, or you might be forced into vim editor to create one!)
- Make sure you `git commit` and `git push` at the end of each work session

Working with `git`

- Don't be discouraged if `git` concepts are elusive at first
 - You can get by with just a few key ideas
 - Tutorials and explanations linked from Resources section of Piazza (go to General Resources area, then click on Tools Reference)
 - Lots of help available from TAs/CAs, instructors, Google



Recycle Bin



PuTTY



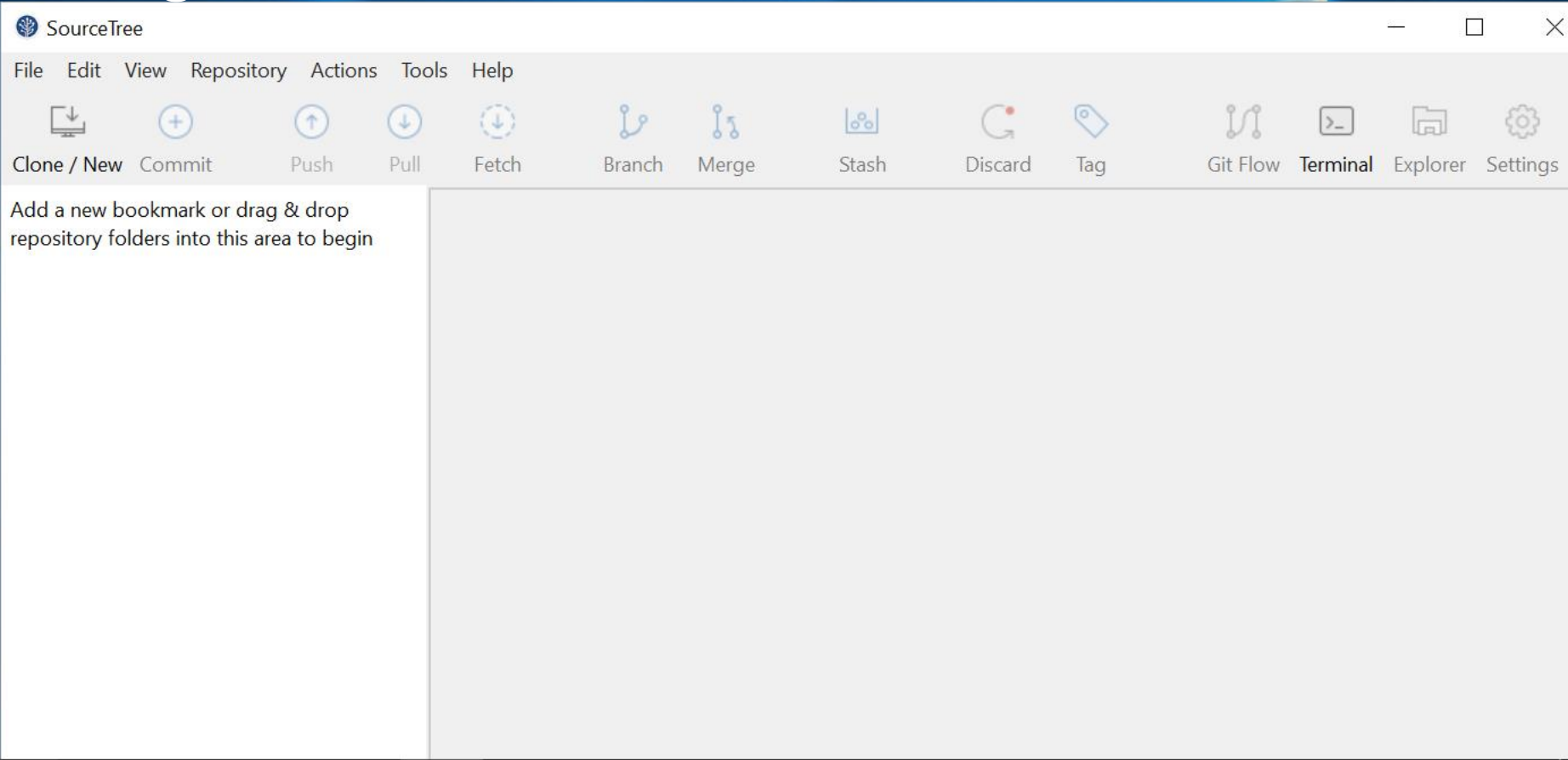
WinSCP



SourceTree

Working with git

There are also GUIs for git, like SourceTree.



Type or talk to search



12:03 AM
2/3/2017



Tarring a directory

The tar command lets us package everything in a directory (and all its sub-directories) into a single file.

Why?

- It is easier to move a single file

Tarring a directory

Basic operations:

- Create a “tar-ball”, filtered through gzip, and put it in a file.

```
tar -c -z -f <tar-ball name>.tgz <directory name>
```

```
tar -czf <tar-ball name>.tgz <directory name>
```

- Extract the contents of a tar-ball file that was filtered with gzip

```
tar -xzf <tar-ball name>.tgz
```

- List the contents of a tar-ball file that was filtered with gzip

```
tar -tzf <tar-ball name>.tgz
```

Transferring files

While we will do our work on the ugradx machines, we will sometimes want to copy our files to other machines.

- Audio
- Video
- Printing
- etc.

Or we may want to move content from our machines to ugradx...

Since the content on our ugradx is ours, we need to do this securely (using authentication to validate that the files are ours).

Transferring files

To perform local/local copying, we use the copy command, `cp`.

To perform local/remote or remote/local copying we use the secure copy command, `[p]scp`.

`[p]scp misha@ugradx.cs.jhu.edu:<source> <target>`
moves from ugradx to local machine.

`[p]scp <source> misha@ugradx.cs.jhu.edu:<target>`
moves to ugradx from local machine.

Transferring files

Example:

On Linux, if the source file is `cs120/foo.c` on `ugradx` and we want to copy it to the current directory on our local machine:

```
scp misha@ugradx.cs.jhu.edu:cs120/foo.c .
```

On Windows, if we want to copy the file to directory `foo`:

```
pscp misha@ugradx.cs.jhu.edu:cs120/foo.c foo\
```



Recycle Bin



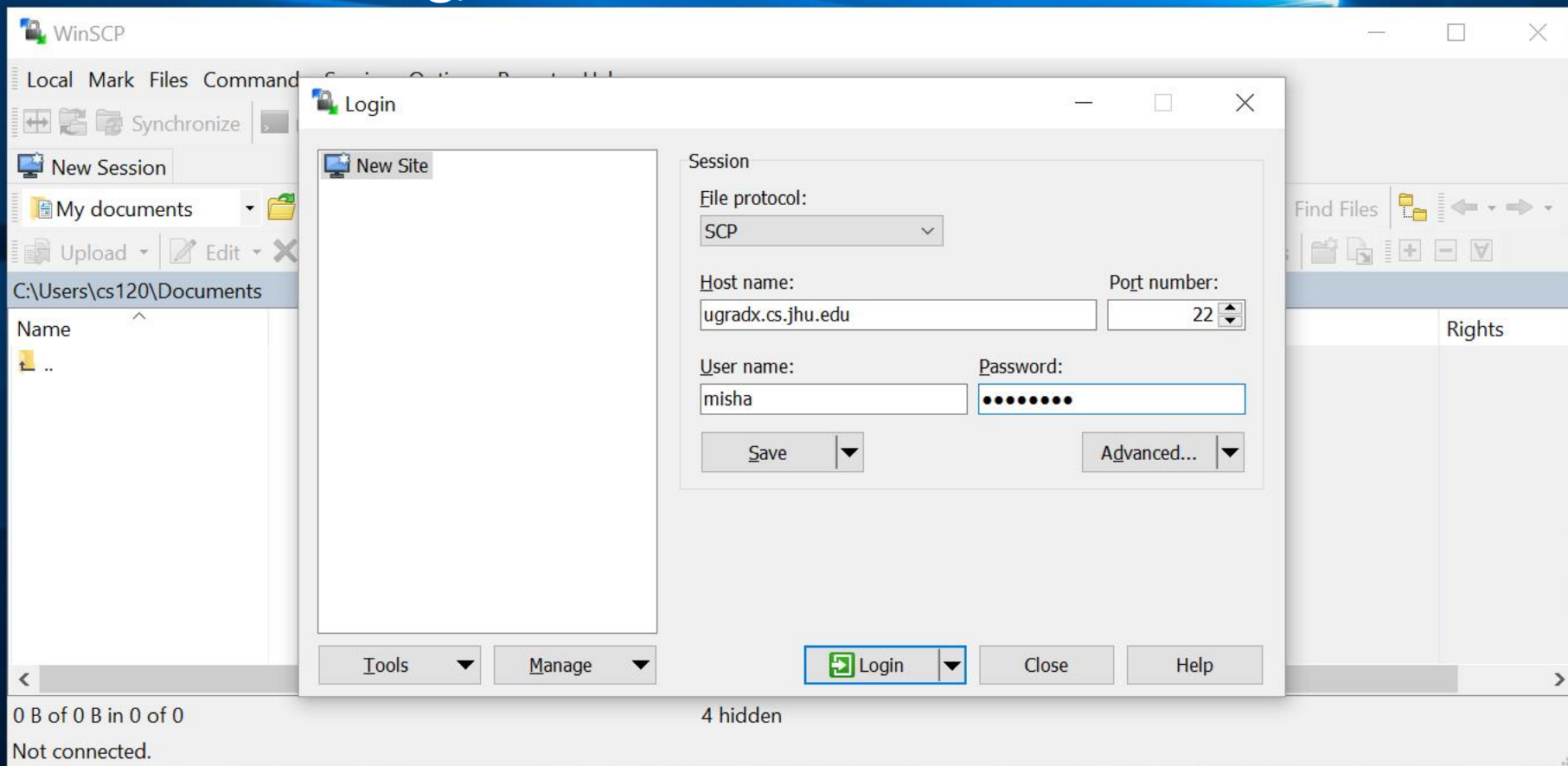
PuTTY



WinSCP

Transferring files

There are also GUIs for transferring, like WinSCP on windows machines.



Type or talk to search



11:53 PM
2/2/2017



In-Class Exercises

- On Piazza, find Resources section, then click Resources tab
- Scroll down to section for this course section
- Find link for Exercise 1-2 and follow it
- Follow the instructions; raise your hand if you get stuck
- Make sure you check in with a course staff member sometime during this session