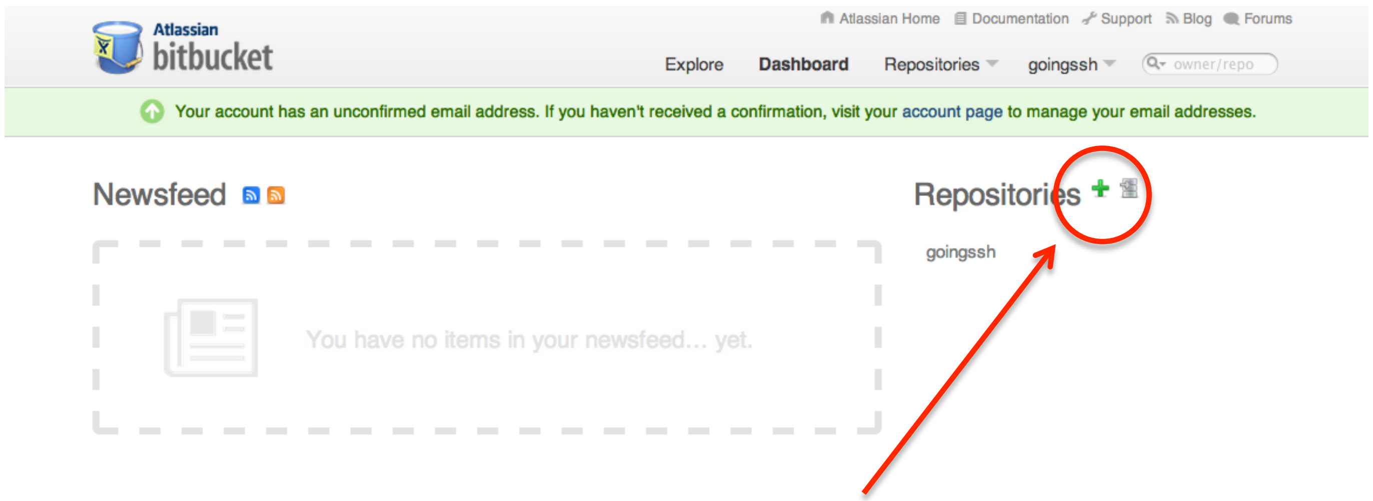
Getting Started with Mercurial, Bitbucket, and MacHG

This lab will help get your group set up using the Mercurial version control system. MacHG is a nice free, open-source GUI for Mercurial. Bitbucket is a free (for small groups and academics) web-based project hosting service that supports Mercurial. The following will walk you through the steps to use these 3 services for your group's project development. **Make sure you follow these in order**, and ask questions along the way if any part does not make sense.

Step 1 – Set up a bitbucket account and repository

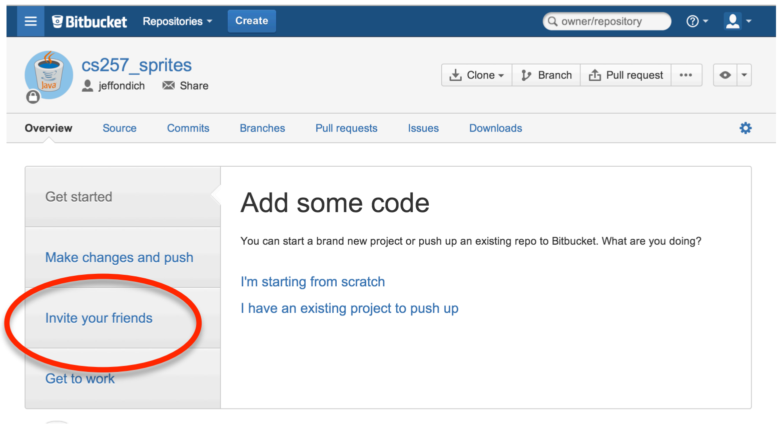
(This step will probably go fastest if 1 group member who will be setting up the actual repository has his/her own computer and the other group members work on separate computers):

a. Every member of your group should make their own account on bitbucket.org. **Make sure you use your Carleton email address to sign up** so that you can get the free, unlimited academic license, which has no restrictions on the number of team members associated with a repository. I also recommend using a new password just for this.
b. One member of your group should create a new repository on bitbucket. After creating an account bitbucket should bring you to a page similar to the following:


Atlassian Home
Documentation
Support
Blog
Forums
Atlassian
bitbucket
Explore
Dashboard
Repositories
goingssh
owner/repo
Your account has an unconfirmed email address. If you haven't received a confirmation, visit your account page to manage your email addresses.
Newsfeed
Repositories
goingssh
You have no items in your newsfeed... yet.

To create a new repository click the + sign (circled above) next to "Repositories" towards the upper right corner of the website. Then give your project a name, make sure the "Mercurial" box is checked, and also check the "issue tracking" box. Select whatever language your Comps group will be using in the "language" drop-down menu, enter a description for your project, and leave the website box blank. Finally, click "Create Repository".

Bitbucket should now take you to your new repository's page, which will look something like the following:



Add some code

You can start a brand new project or push up an existing repo to Bitbucket. What are you doing?

I'm starting from scratch

I have an existing project to push up

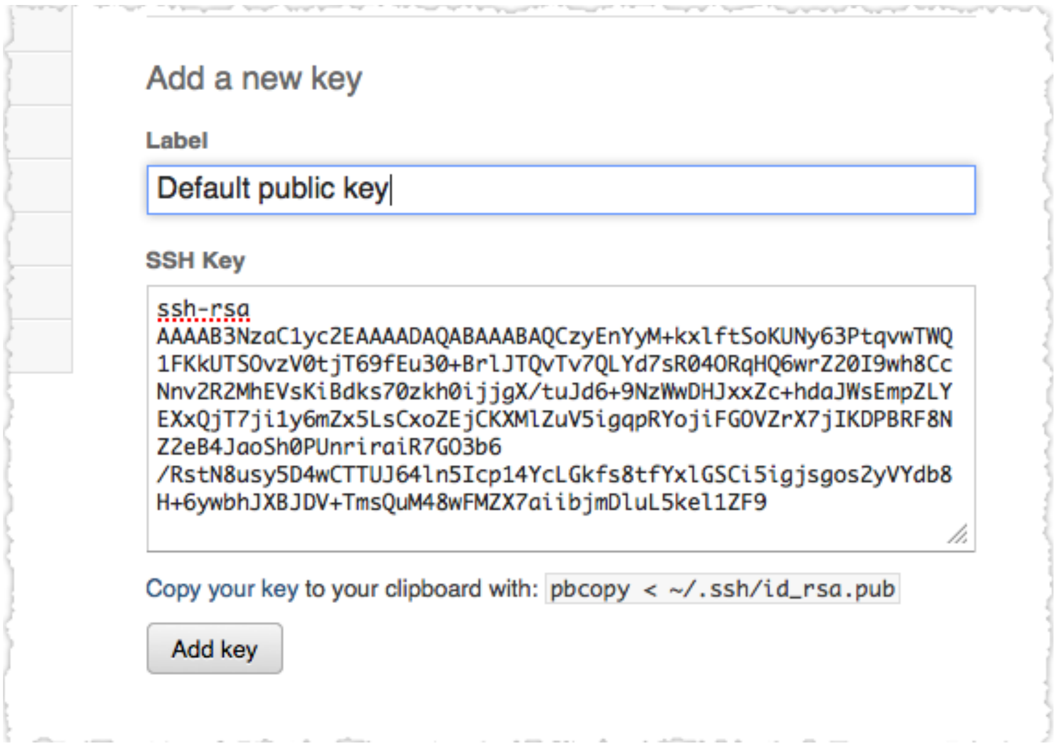
Invite your other group members to your repository by clicking the "Invite your friends" button, circled above. Enter the emails each group member used for their bitbucket account, and **make sure to choose "admin" access for each group member invited**.

c. The rest of the group members should accept invitations to the project they receive through email.

Step 2 – Set up and install your public SSH key

To securely exchange files between the bitbucket repository and your local repository, we'll be using SSH. In order to use SSH, you'll need to generate an SSH public key and add it to your bitbucket account. Each person in your group should do the following:

a. Open a terminal window.
b. Type in the following command: `ssh-keygen -b 4096 -t rsa` You will be prompted for a file name, which will be something like `/Accounts/username/.ssh/id_rsa` Go ahead and accept the default filename. You'll then be asked to enter a passphrase. **Do not use your Carleton password for this!** You can use the same password you used for bitbucket, but the two are not linked so it doesn't matter what password you use (as long as you can remember it).
c. Go back to your web browser. Log into bitbucket if you are not currently logged in.
d. Click on your your ***username*** on the upper left of your screen, then click on the **Manage Account** button. The system should now display the account settings page.
e. Click **SSH keys.** The SSH key page should now be displayed. It shows a list of any existing keys (this will be blank unless you've used SSH keys on other bitbucket projects).
f. Go back to your terminal window. Copy the contents of the public key file. On Linux, you can type `cat ~/.ssh/id_rsa.pub`. In Mac OSX the following command copies the key to the clipboard: `pbcopy < ~/.ssh/id_rsa.pub`
g. Go back to your browser, click on **Add Key**, and enter a label for your public key, such as `Default public key`.
h. Paste the copied public key into the **Key** field:


Add a new key

Label

Default public key

SSH Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCzyEnYyM+kxlftSoKUNy63PtqvwTWQ
1FKkUTSOvzV0tjT69fEu30+BrlJTQvTv7QLYd7sR04ORqHQ6wrZ20I9wh8Cc
Nnv2R2MhEVsKiBdks70zkh0ijjgX/tuJd6+9NzWwDHJxxZc+hdaJWsEmpZLY
EXxQjT7ji1y6mZx5LsCxoZEjCKXMlZuV5igqpRYojiFGOVZrX7jIKDPBRF8N
Z2eB4JaoSh0PUnriraiR7GO3b6
/RstN8usy5D4wCTTUJ64ln5Icp14YcLGkfs8tfYxlGSCi5igjsgos2yVYdb8
H+6ywbhJXBJDV+TmsQuM48wFMZX7aiibjmDluL5kel1ZF9
```

Copy your key to your clipboard with: `pbcopy < ~/.ssh/id_rsa.pub`

Add key

i. Press **Add key**. The system adds the key to your account.

Step 3 – Set your MacHG preferences

Now that you have a bitbucket repository and SSH public key set up, you can use MacHG to access that repository locally. At some point each group member is going to want to log on to their own account (or own computer) to do the following:

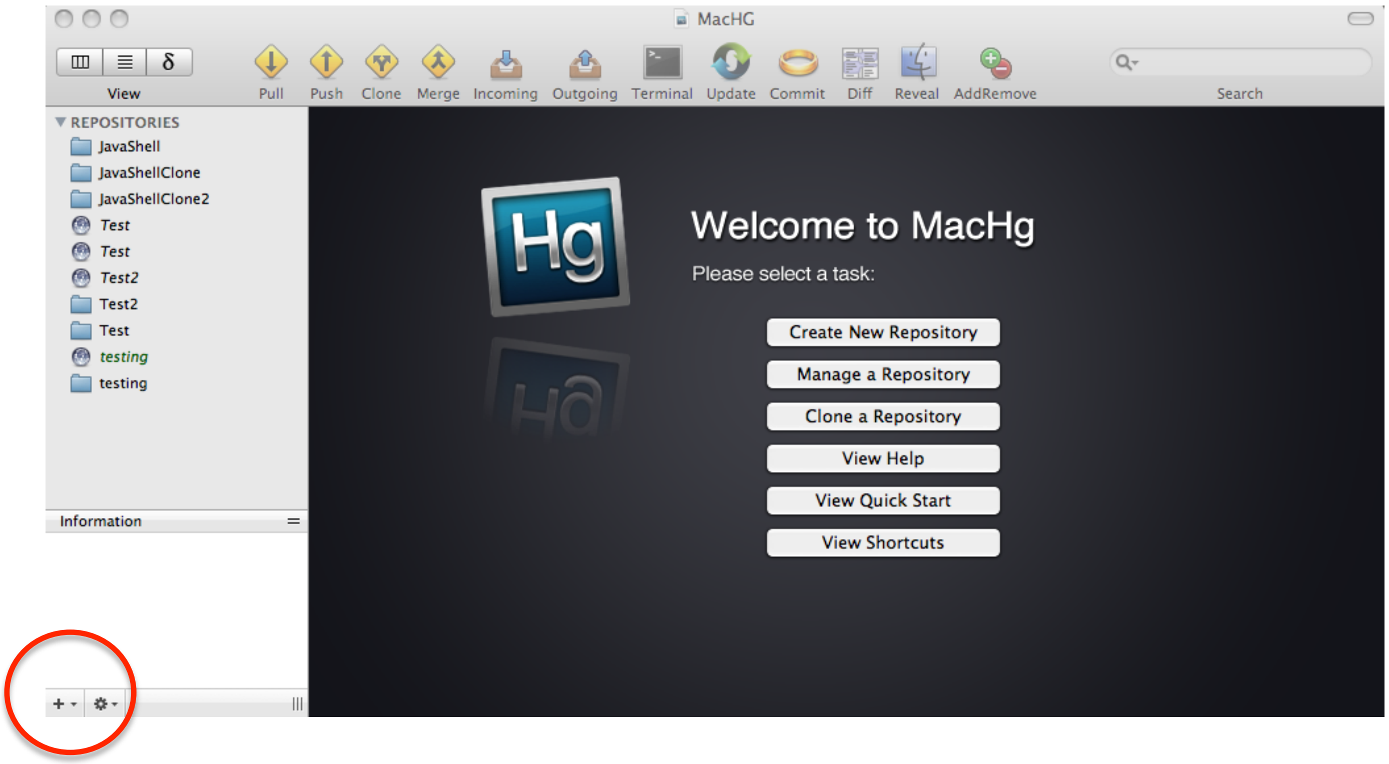
a. Open MacHG from Carleton Apps (shortcut under Applications). (Or download it for your Mac, if you have one, at http://jasonfharris.com/machg/downloads/.)
b. If a box pops up asking for some username info, put in your name and <email address> and you can skip part f below. Ignore, for the time being, any other error messages that pop up.
c. Right click on the toolbar area and choose "customize toolbar".
d. Drag the following buttons to the toolbar: Update, Commit, Diff, Reveal, & AddRemove.
e. (optional but likely what you want) – Choose MacHG->Preferences from the menu at the very top of your screen. You should be on the "General" tab. Change "Double Click" to "Open" and "Command Double Click" to "Diff".
f. Still in Preferences, go to the "Advanced" tab. Click the "Edit" button next to '~/Application Support/MacHg/hgrc'. In the file that opens, the very last line should say "username = ...". Change whatever is there to your name and <email address>. For instance the last two lines of Jeff's file is

   ```
   [ui]
   username = Jeff Ondich <jondich@carleton.edu>
   ```

   This is what shows up every time you commit a change to the repository, making it easy for others working on your project to know who did what and to email someone if they have a question about a commit.
g. Note that MacHG may ask if you want to save a file on exit or other times. This file stores your info about local repositories and username/password/URL for server repositories so you do want to save it, preferably with a better name than "untitled".

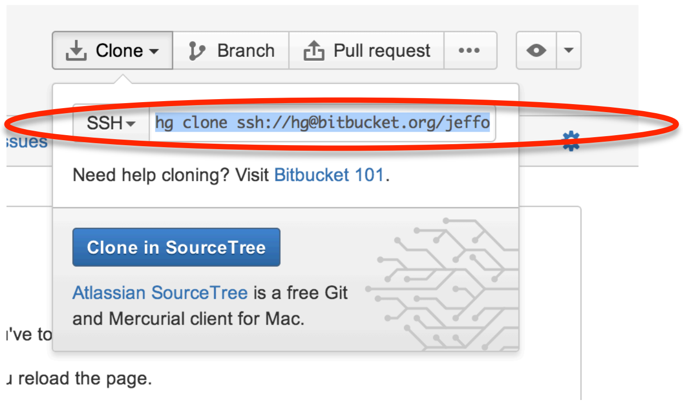
Step 4 – Create a local copy of the repository

Each group member should do this:



a. In MacHG choose "Add Server Repository" by clicking the + in the lower left-hand corner (circled above).

b. Fill out the form that pops up on the screen. The short name can be whatever you want. The server name you can get from bitbucket. If you go to the repository page, under the Overview tab, you should see a label that says "Clone this repository" (see the picture below). Click on SSH. You should now see the name of the repository, starting with ssh://. Use this URL (including the ssh:// for the server name. Leave `hg` as the username.

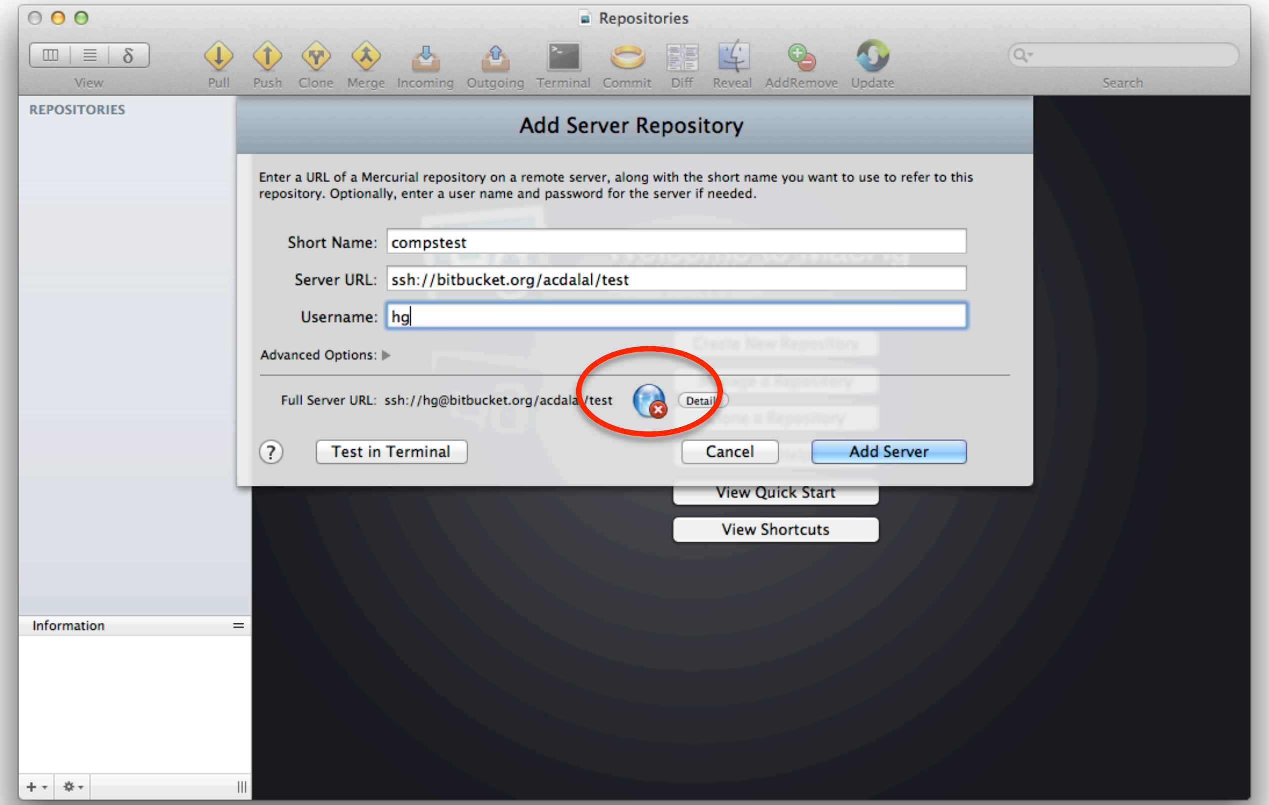


```
hg clone ssh://hg@bitbucket.org/jeffo
```

Need help cloning? Visit Bitbucket 101.

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

c. At this point, the MacHG window should show a "denied" status, indicated by the world icon with the red X in the picture below.



Add Server Repository

Enter a URL of a Mercurial repository on a remote server, along with the short name you want to use to refer to this repository. Optionally, enter a user name and password for the server if needed.

Short Name: compstest

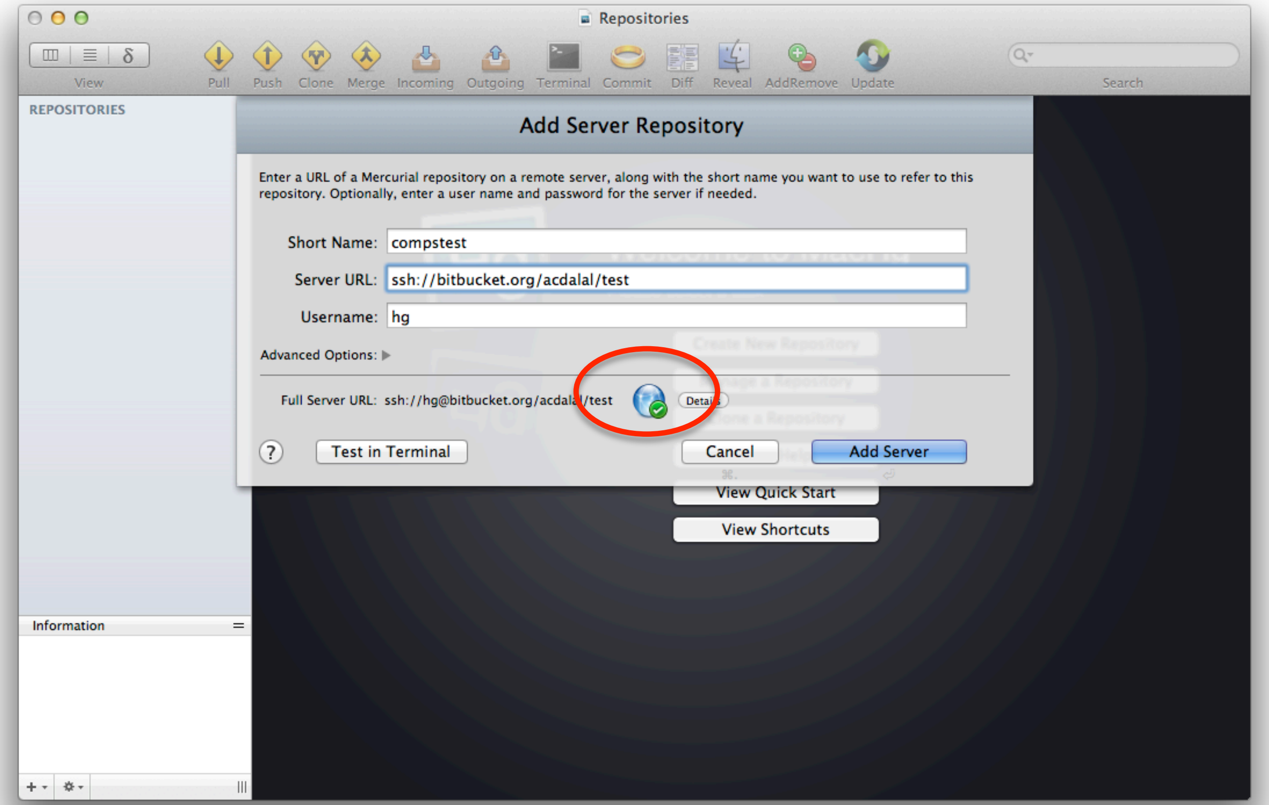
Server URL: ssh://bitbucket.org/acdalal/test

Username: hg

Advanced Options:

Full Server URL: ssh://hg@bitbucket.org/acdalal/test

Click on the “Test in Terminal” button, and enter your SSH passphrase when prompted. (You can save this in your keychain if you’d like.) Assuming your password is accepted, close the terminal window. Then go back to the server URL box, delete part of the name (say, “test”) and retype it in the window. You should now see a green check, indicating that you’ve set up the server repository correctly and that you are authenticating to the server correctly (as in the image below).



Add Server Repository

Enter a URL of a Mercurial repository on a remote server, along with the short name you want to use to refer to this repository. Optionally, enter a user name and password for the server if needed.

Short Name: compstest

Server URL: ssh://bitbucket.org/acdalal/test

Username: hg

Advanced Options:

Full Server URL: ssh://hg@bitbucket.org/acdalal/test

d. In MacHG click on the server repository you just created, then click the **Clone** icon (*at the top of the MacHG window, **not** the Clone A Repository button in the middle)* to create your local copy. Save the directory wherever you would like on your account.

Step 4 – Using mercurial to add, modify, update, etc. project files

a. To add files to your project, copy them to your local working directory (the one you chose in the previous step to save your repository to on your own account/computer). They will show up in MacHG with ? icons next to their names. Select the files to be added and click the “AddRemove” icon in the toolbar (or right click and choose “AddRemove”). Click on the commit icon in the toolbar (or choose “Actions->Commit” from the menu at the top of your screen). Click on the “push” icon to make your changes on the central copy of your project on bitbucket. **Note that until you choose “push”, only your local copy has changed---no one else will have access to those changes.**

b. To get changes others have made to your project:
   - Click on “pull”
   - Check the message pull produces to see if there were conflicts or not with the changes others made and the changes you’ve made in your local copy (good design will avoid this as much as possible). If the last line of the pull message says "run hg update to get a working copy" that means there were no conflicts. Otherwise there were conflicts and when you close the pull message you should notice that the “merge” icon is now available instead of grayed out.
   - If there were NOT conflicts, click the update icon. Select the most recent revision in the list shown and click “update”. This will actually make the changes to your local copy. Note by default the version selected when you click the update icon will be your last commit, not the most recent version of the project, so don’t forget to click on the most recent (top) revision in the list before clicking the “update” button.
   - If there WERE conflicts, click the merge icon. Again select the most recent revision in the list and click “merge”. This will open the mergefile program that will show you the 2 versions of the conflicted file side-by-side with any differences marked. For each conflict, choose what you want to happen from the “actions” pull-down menu in lower right corner. SAVE this merging, either with <ctrl> - s or file->savemerge, then close the mergefile program. Click the “merge” button to finalize the merge. You should still then have to open the merged file and check that it looks like you want it to.

c. To make changes available to the rest of your group:
   - **IMPORTANT!** First follow the steps in part b to get any changes others in your group have pushed since you last pulled. If there are any conflicts you MUST merge those first or Mercurial will not let you push. If there were no conflicts you should still update to the most recent version, then submit your changes.
   - Choose all files that have been modified (they'll have the blue modified icon by them)
   - Click on the commit icon in the toolbar or choose Actions->Commit
   - Click the push icon

The very basics of the above steps to remember are:

1. **To get changes**: make sure any of your changes are committed first, then pull, then update or merge depending on conflicts. Don’t forget to click on most recent revision in list in either case!
2. **To commit changes**: commit any changed files, then make sure you have gotten any changes per step 1, then commit any merged files if necessary, then push.

PRACTICE these steps with your group. Have one member add some files to the project (it doesn't matter what they are). Make sure everyone else can get them correctly. Have one member make some changes to a file and make sure others can get those changes. Finally figure out how to cause a conflict by having 2 group members modify the same line of a file each in their local copy of the project. Then the first member to commit and push shouldn't have any problems, but the 2nd member will have conflicts when they pull the first member's committed changes. Go through the merge process and commit/push the merged files.

Advanced: Using Mercurial from the command line
If you would like to use mercurial from the command line, you can use the hg command. MacHG uses its own internal version of hg named mhg. If you'd like to use that version, you can either click the terminal icon in MacHG and then use the aliased command mhg instead of hg, or you can set that alias yourself in your .bash_profile file using the following steps.

a. Click the terminal icon in MacHG
b. Type "alias mhg" and copy the line the terminal prints---it should be something like "alias mhg=... ${mhgResources}..."
   Paste this line into some other text document
c. Type "echo $mhgResources"
d. Copy the resulting path and replace the ${mhgResources} in your previous alias line with this entire path. ONLY replace ${mhgResources}, the rest of the line leave unchanged.
e. Finally open your .bash_profile file in a regular terminal window (or create it if you don't already have one in your home directory) and copy the resulting line from part d into it. Save it and type "source .bash_profile" in the regular terminal window. You should now be able to use the command mhg from any terminal, not just the MacHG one.

For example, to add the file bunnies.txt to your bitbucket repository (assuming the file bunnies.txt already exists):

```
hg add bunnies.txt
hg commit –m "Adding bunny file to the repository"
hg push
```

To update your local repository based on changes your teammates have made:

```
hg pull
hg update
```

Refer to the Mercurial (HG) Cheat Sheet for some common mercurial commands.