

SECTION 2:

HW3 Setup

cse331-staff@cs.washington.edu

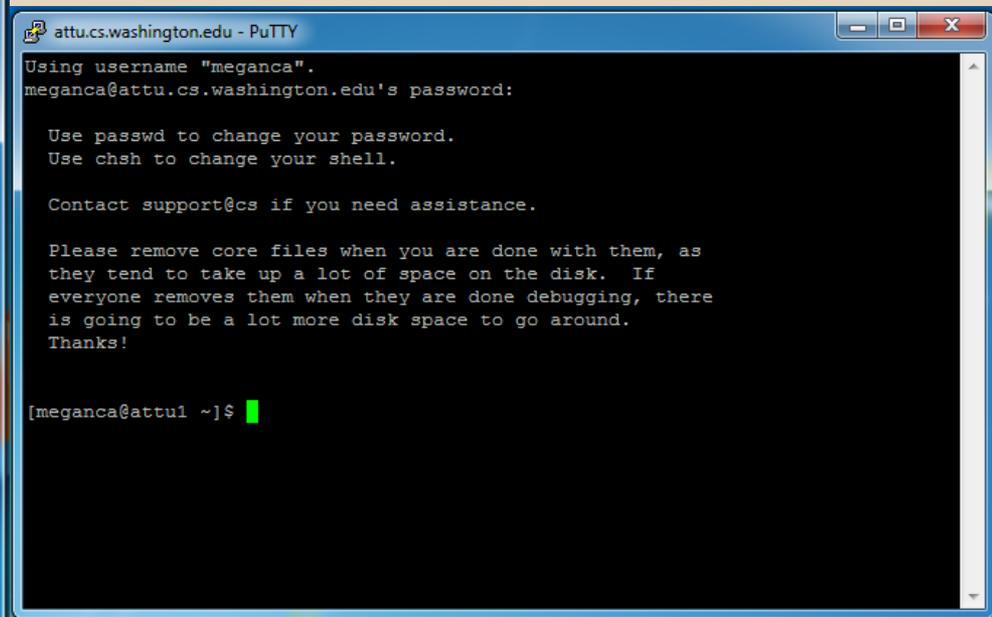
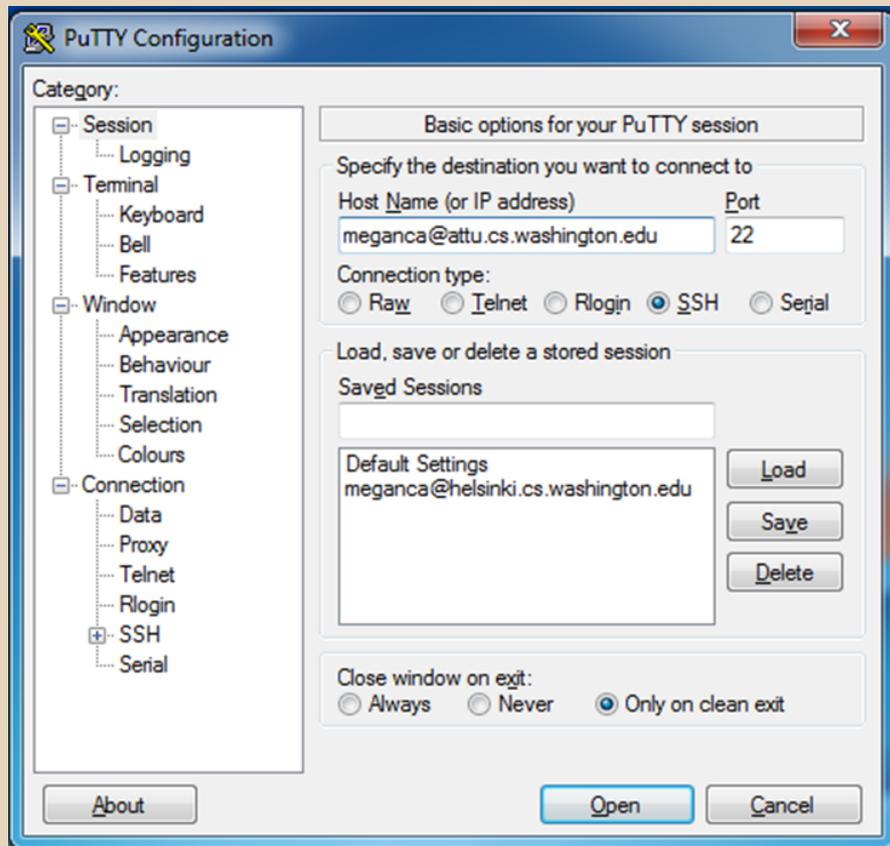
DEVELOPER TOOLS

- **Remote access**
- **Eclipse and Java versions**
- **Version Control**

WHAT IS AN SSH CLIENT?

- **Uses the secure shell protocol (SSH) to connect to a remote computer**
 - Enables you to work on a lab machine from home
 - Similar to remote desktop
- **Windows users: PuTTY and WinSCP**
 - PuTTY: ssh connection
 - WinSCP: transfer or edit files
- **Mac/Linux users: Terminal application**
 - Go to Applications/Utilities/Terminal
 - Type in “ssh cseNetID@attu.cs.washington.edu”
- **More Information: <http://courses.cs.washington.edu/courses/cse331/15sp/tools/WorkingAtHome.html#remote-attu>**

PuTTY



TERMINAL (LINUX, MAC)

```
meganca@charmader: ~  
meganca@charmader:~$ ssh meganca@attu.cs.washington.edu  
meganca@attu.cs.washington.edu's password:  
Last login: Wed Sep 24 17:13:13 2014 from c-24-19-57-209.hsd1.wa.comcast.net  
  
Use passwd to change your password.  
Use chsh to change your shell.  
  
Contact support@cs if you need assistance.  
  
Please remove core files when you are done with them, as  
they tend to take up a lot of space on the disk. If  
everyone removes them when they are done debugging, there  
is going to be a lot more disk space to go around.  
Thanks!  
  
[meganca@attu3 ~]$ █
```

ECLIPSE and Java

- Get Java 7
- Important: Java separates compile and execution, eg:
 - `javac Example.java` $\xrightarrow{\text{produces}}$ `Example.class`
 - Both compile and execute have to be the same Java!
- Please use **Eclipse 4.4**
- Instructions: http://courses.cs.washington.edu/courses/cse331/15sp/tools/WorkingAtHome.html#Step_1

ECLIPSE and Java

.java files

- Human readable 'code' file



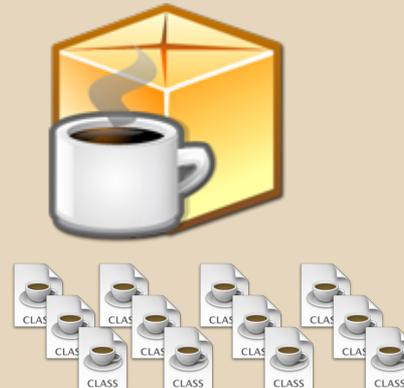
.class files

- Compiled version of .java files. Typically represented as Byte code to run on the Java Virtual Machine (JVM)

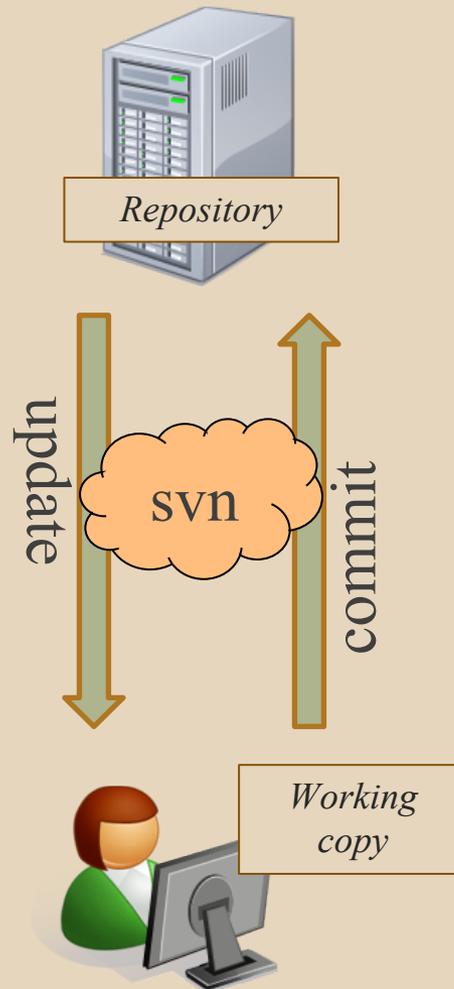


.jar files

- Packaged aggregate of .class files and metadata

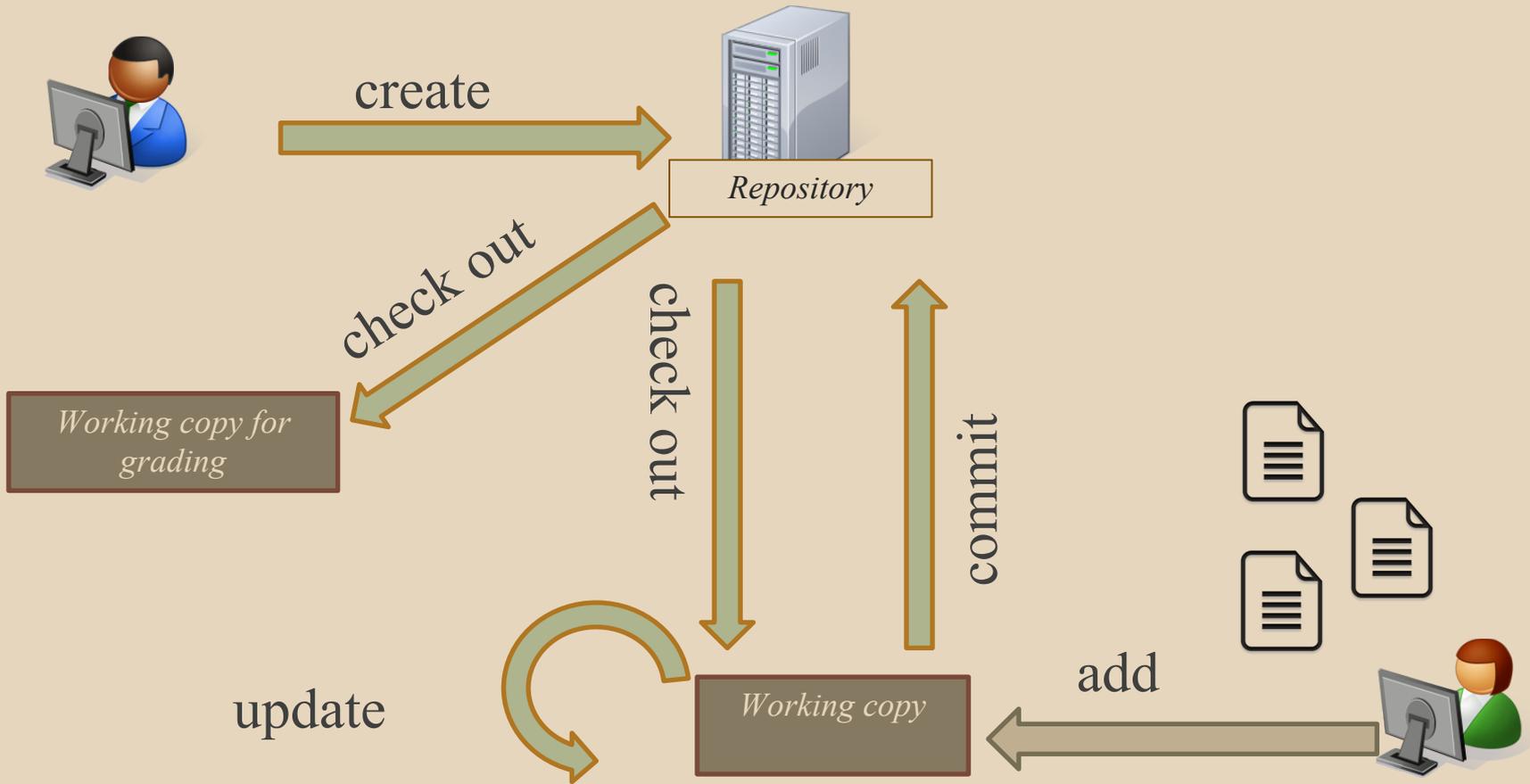


VERSION CONTROL REVIEW



Refer to Section 1 slides for more information on Version Control.

331 VERSION CONTROL



VERSION CONTROL: COMMAND-LINE

command	description
<code>svn co <i>repo</i></code>	check out
<code>svn ci [<i>files</i>]</code>	commit / check in changed files
<code>svn add <i>files</i></code>	schedule files to be added at next commit
<code>svn help [<i>command</i>]</code>	get help info about a particular command
<code>svn merge <i>source1 source2</i></code>	merge changes
<code>svn revert <i>files</i></code>	restore local copy to repo's version
<code>svn resolve <i>files</i></code>	resolve merging conflicts
<code>svn update [<i>files</i>]</code>	update local copy to latest version
others: blame, changelist, cleanup, diff, export, ls/mv/rm/mkdir, lock/unlock, log, propset	

THIS QUARTER

- We distribute starter code by adding it to your **repo**
- You will **code** in Eclipse
- The version control system we will be using is **subversion**
 - You turn in your files by **adding** them to the repo and **committing** your changes
- You will **validate** your homework by **SSHing** onto attu and running an Ant build file

331 VERSION CONTROL

- Your main repository is at
/projects/instr/15sp/cse331/YourCSENetID/REPOS/cse331
- Only check out once (unless you're working in a lot of places)
- Don't forget to add files!!
- Check in your work!

HOW TO USE SUBVERSION

- Eclipse plugin: Subclipse
 - **Recommended!**
- GUI interface: TortoiseSVN
- Command line: PuTTY

IMPORTANT DETAILS

- **Windows** users
 - Need to download [TortoiseSVN](#) and [Putty](#) anyways, to avoid errors known to come up in the Eclipse plug-in, Subclipse
- **Mac** users do not need to do this step.

CHECKING OUT YOUR REPO

- To check out a local copy of your repository on Eclipse
 - First need to install Subclipse: <http://courses.cs.washington.edu/courses/cse331/15sp/tools/WorkingAtHome.html#Step3Eclipse>
 - Next, need to checkout a local copy of your repository through Subclipse: <https://courses.cs.washington.edu/courses/cse331/15sp/tools/versioncontrol.html#SetUpEclipse>

HW 3

- Many small exercises to get you used to version control
- More information on homework instructions: <http://courses.cs.washington.edu/courses/cse331/15sp/hws/hw3/hw3.html>
- Committing changes: [Instructions](#)
 - How you turn in your assignments
- Updating changes: [Instructions](#)
 - How you retrieve new assignments

Turning in HW3

- Instructions
- Done by simply committing your changes
 - Good to do this early and often
 - Most recent commit before the deadline will be used for grading
- Before final commit, remember to run ant validate

Ant Validate

- **What will this do?**
 - Checks out a fresh local copy of your repository with all your changes
 - Makes sure you have all the **required** files such as `hw3/answers/problem6.txt`
 - Make sure your homework builds without errors
 - Passes specification and implementation tests in the repository
 - **Note:** this does not include the additional tests we will use when grading
 - This is just a sanity check that your current tests pass

Ant Validate

- **How do you run ant validate?**
 - Has to be done on attu from the command line since that is the environment your grading will be done on
 - Do not use the Eclipse ant validate build tool!

Ant Validate

- **How do you run ant validate?**
 - Steps
 - Log into attu via [SSH](#)
 - In attu, checkout a local copy of your repository through the [command-line](#) if you have not already
 - **Note:** Now, you should have two local copies of your repository, one on your computer through Eclipse and one in attu
 - Go to the hw folder which you want to validate through the 'cd' command
 - For example: `cd ~/cse331/src/hw3`
 - Run ant validate

Ant Validate

- **How do you know it works?**
 - If successful, will output **Build Successful** at the bottom
 - If unsuccessful, will output **Build Failed** at the bottom with information on why
 - If ant validate failed, fix and commit changes through eclipse, go to the copy of your repo on attu, and do 'svn update', and try ant validate again

Ant Validate

- **For the future**
 - Now have two local copies of your repository
 - One on your computer through Eclipse
 - One on attu through the command-line
 - Code and commit changes through Eclipse
 - Afterwards, go to repo on attu and do a 'svn update' command to retrieve all the changes you made from Eclipse
 - Run ant validate