Lab: Mercurial

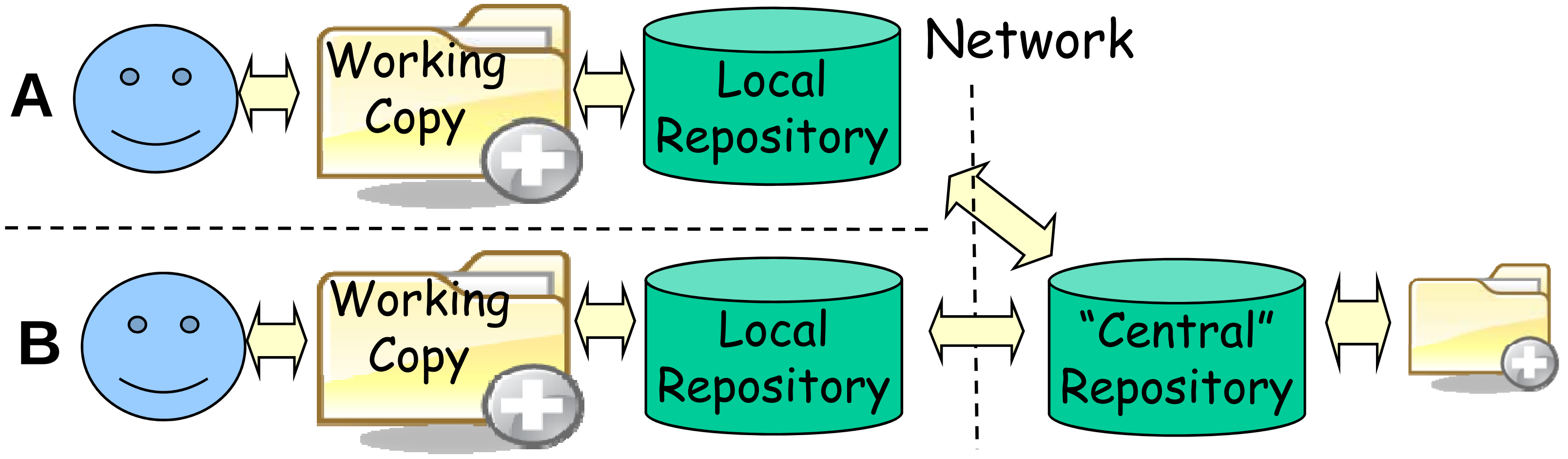
First: Recap

- Distributed Version control

Distributed Version Control

All developers have their own local repository (a.k.a. “decentralized version control”)

1. Developers work on their working copy
2. Developers commit changes of the working copy to their own local repository first
3. Changes can be exchanged between repositories (“pushed” and “pulled”)



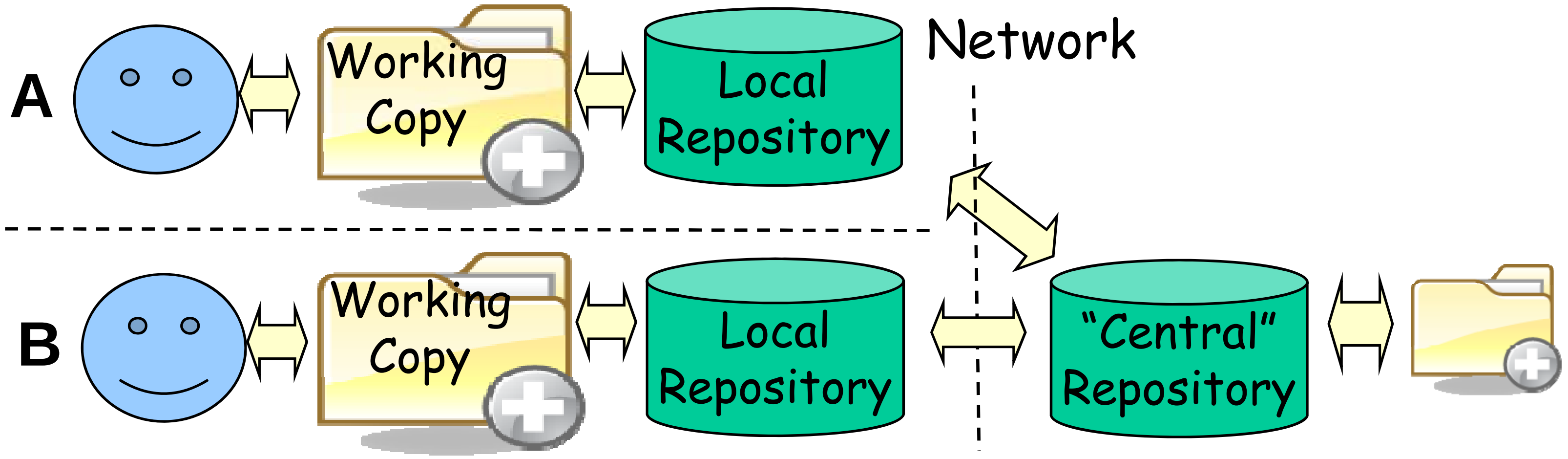
Push and Pull

Push

- Once developers have committed versions on their local repository, they can push them to another repo
- Versions are pushed from local branches into corresponding remote branches
- Like "commit" from one repo to another

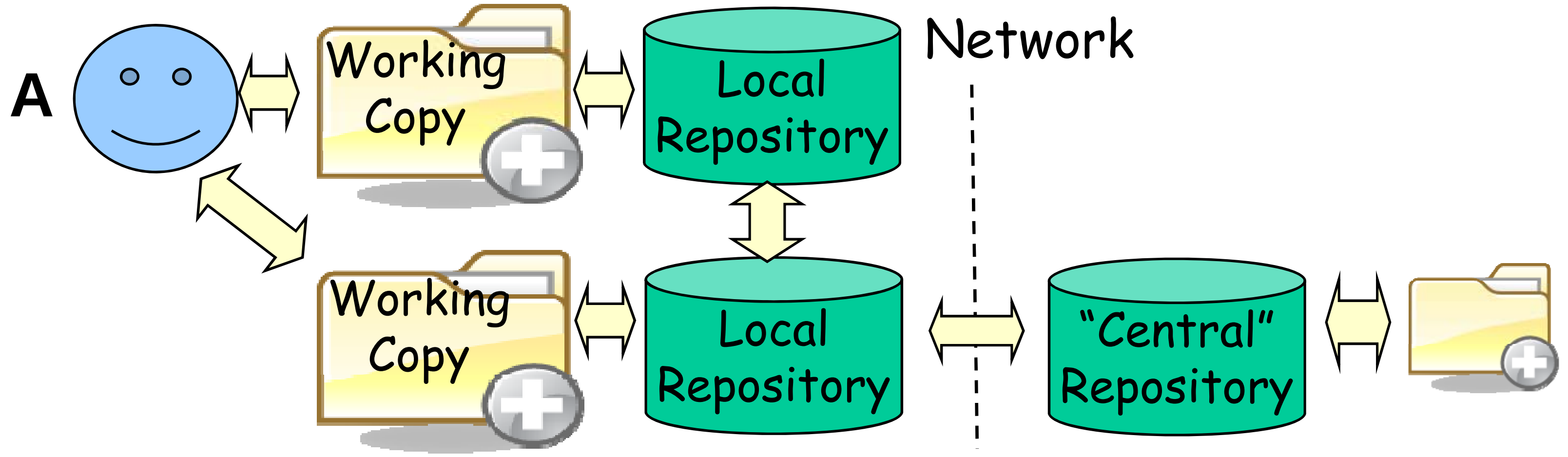
Pull

- Latest versions are pulled from remote branches and put into the corresponding local branches
- Like "update" from one repo to another



Distributed Version Control

- Local and remote repositories are technologically identical.
- Chaining of repositories is possible.
- Several personal repositories can be used:
- Good for testing of new commits:
- one repo for development, one for staging



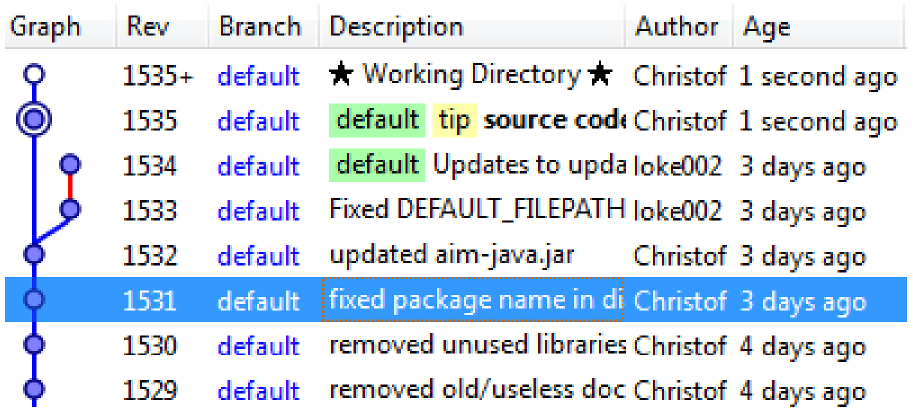
Mercurial

- Open-source project, started around 2005
- Used for many open-source projects

- Every developer has a repository, which is a folder
- Repo folder contains working copy,
  and a subfolder `.hg` which contains the version data
- Versions are identified locally by natural numbers
  and globally by hash values,
  e.g. 5c240805ac2d9530b780cbd514574af398c0cdd6
- Good tool support (TortoiseHg)
- Fairly easy to use

Working with Hg

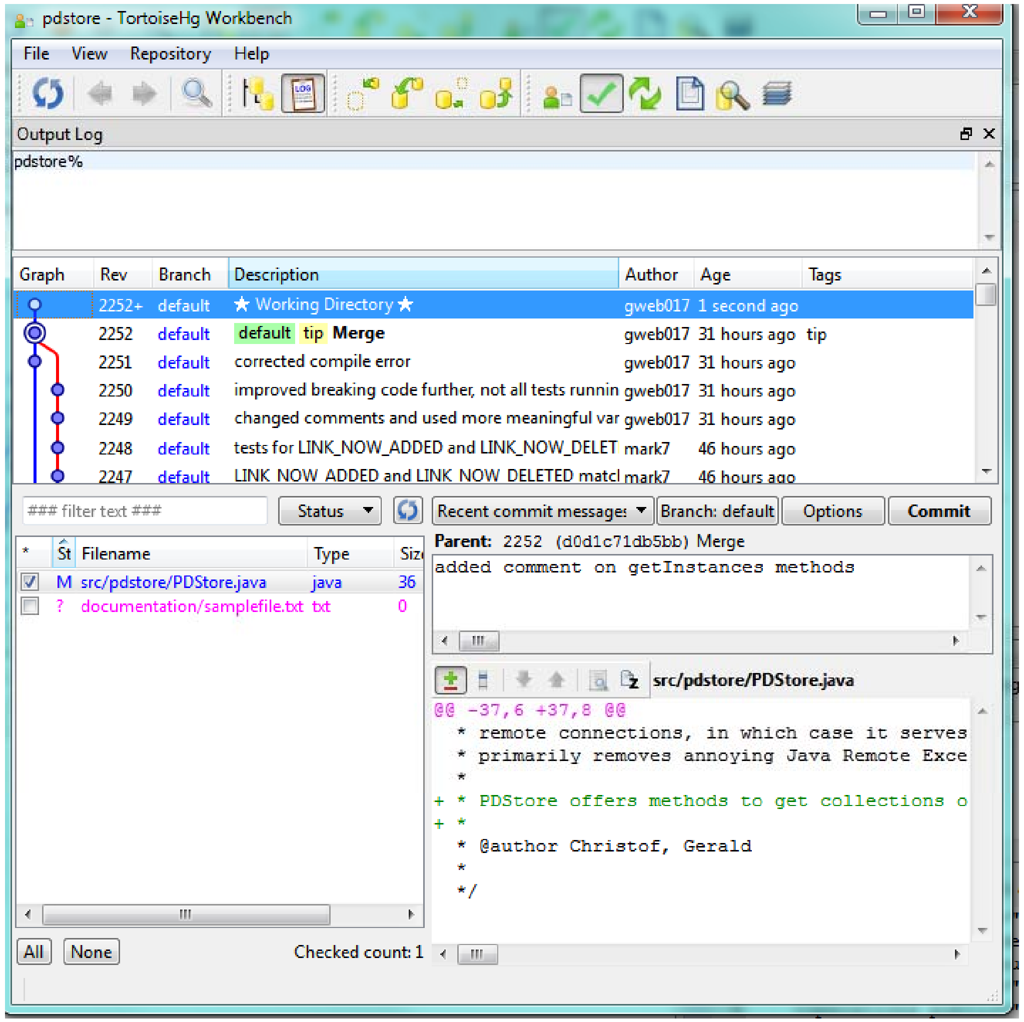
1. Start by **cloning** existing repo, or **creating** new one
   - New repo has only "default branch" (like trunk)
   - After cloning you have local copies of all branches of parent repo
2. Modify working copy and **commit** to create new versions in your local repo
3. **Pull** to load new versions from parent repo into local repo
   - Does not change working copy
   - Pulled versions are put in separate branch from your local versions



Graph	Rev	Branch	Description	Author	Age
	1535+	default	★ Working Directory ★	Christof	1 second ago
	1535	default	default tip source code	Christof	1 second ago
	1534	default	default Updates to upda	loke002	3 days ago
	1533	default	Fixed DEFAULT_FILEPATH	loke002	3 days ago
	1532	default	updated aim-java.jar	Christof	3 days ago
	1531	default	fixed package name in di	Christof	3 days ago
	1530	default	removed unused libraries	Christof	4 days ago
	1529	default	removed old/useless doc	Christof	4 days ago

Most important tool: Tortoise explorer

- Integrated control GUI for mercurial
- In some of the lab tasks you are asked to explore the functions a bit on your own and figure out how certain things work!
- After all, every GUI is intuitive, right?



pdstore%

Graph	Rev	Branch	Description	Author	Age	Tags
	2252+	default	★ Working Directory ★	gweb017	1 second ago	
	2252	default	default tip Merge	gweb017	31 hours ago	tip
	2251	default	corrected compile error	gweb017	31 hours ago	
	2250	default	improved breaking code further, not all tests runnin	gweb017	31 hours ago	
	2249	default	changed comments and used more meaningful var	gweb017	31 hours ago	
	2248	default	tests for LINK_NOW_ADDED and LINK_NOW_DELET	mark7	46 hours ago	
	2247	default	LINK_NOW_ADDED and LINK_NOW_DELETED matcl	mark7	46 hours ago	

filter text

*	St	Filename	Type	Size
☑	M	src/pdstore/PDStore.java	java	36
☐	?	documentation/samplefile.txt	txt	0

Checked count: 1

Parent: 2252 (d0d1c71db5bb) Merge

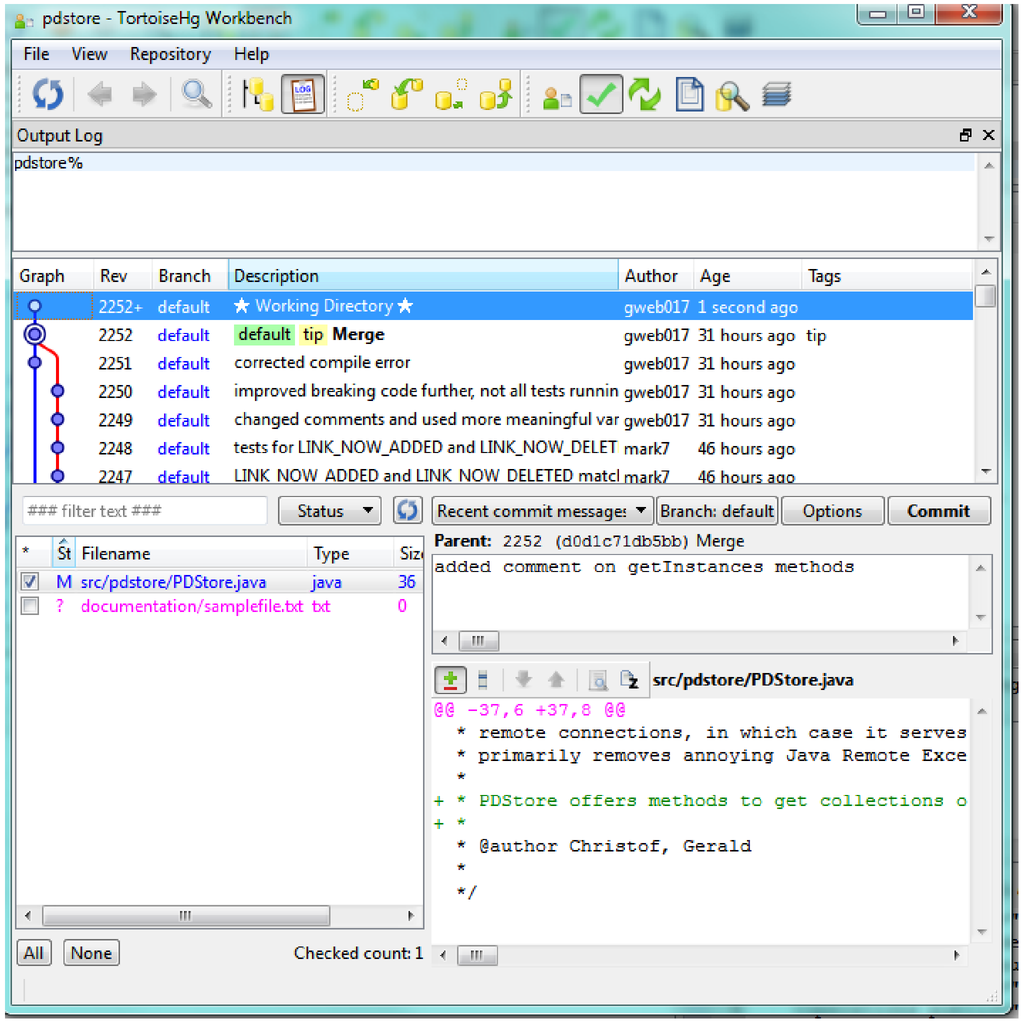
added comment on getInstances methods

src/pdstore/PDStore.java

```
@@ -37,6 +37,8 @@
  * remote connections, in which case it serves
  * primarily removes annoying Java Remote Exce
  *
+ * PDStore offers methods to get collections o
+ *
  * @author Christof, Gerald
  *
  */
```

commit

- Creates a new version on the local repository.
- Best practices:
- Always review your changes!
- Make sure to add new (a.k.a. untracked) files.
- One commit should be only one logical change.
- Never break the build!
- Never break the tests.



Graph	Rev	Branch	Description	Author	Age	Tags
	2252+	default	★ Working Directory ★	gweb017	1 second ago	
	2252	default	default tip **Merge**	gweb017	31 hours ago	tip
	2251	default	corrected compile error	gweb017	31 hours ago	
	2250	default	improved breaking code further, not all tests runnin	gweb017	31 hours ago	
	2249	default	changed comments and used more meaningful var	gweb017	31 hours ago	
	2248	default	tests for LINK_NOW_ADDED and LINK_NOW_DELET	mark7	46 hours ago	
	2247	default	LINK_NOW_ADDED and LINK_NOW_DELETED matcl	mark7	46 hours ago	

*	St	Filename	Type	Siz
✓	M	src/pdstore/PDStore.java	java	36
☐	?	documentation/samplefile.txt	txt	0

Parent: 2252 (d0d1c71db5bb) Merge

```
added comment on getInstances methods
```

src/pdstore/PDStore.java

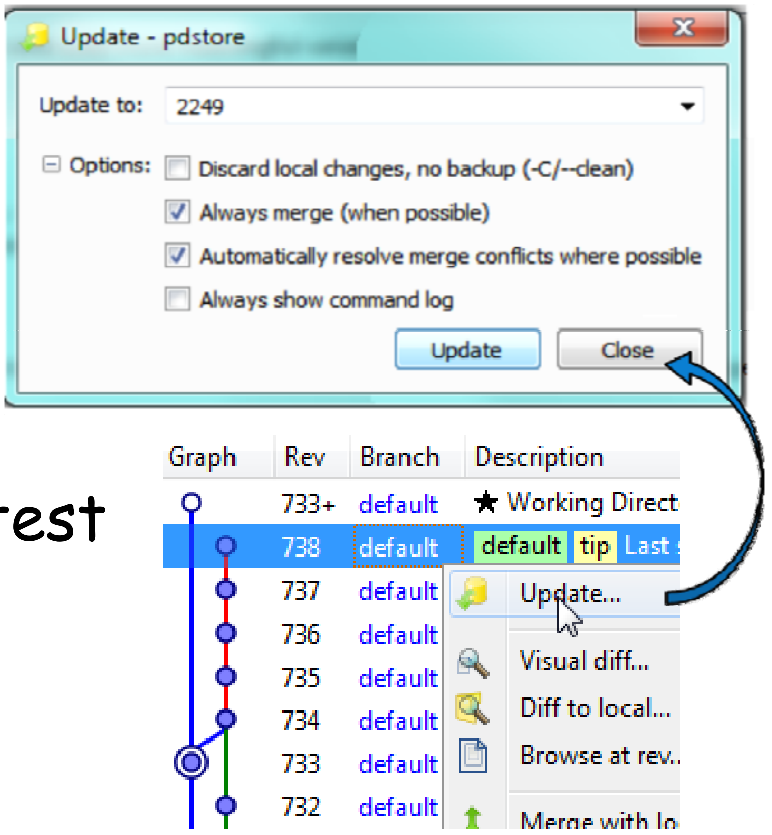
```
@@ -37,6 +37,8 @@
  * remote connections, in which case it serves
  * primarily removes annoying Java Remote Exce
  *
+ * PDStore offers methods to get collections o
+ *
  * @author Christof, Gerald
  *
  */
```

Hg Pull

Pull regularly to stay up to date.

Have you committed local versions on some branch?

1. If no, you can **update** to the latest pulled version
   - Changes in working copy are merged with pulled version
   - Unless you choose to "discard local changes"
2. If you have committed local versions on some branch, they should be **merge**d with pulled versions on same branch

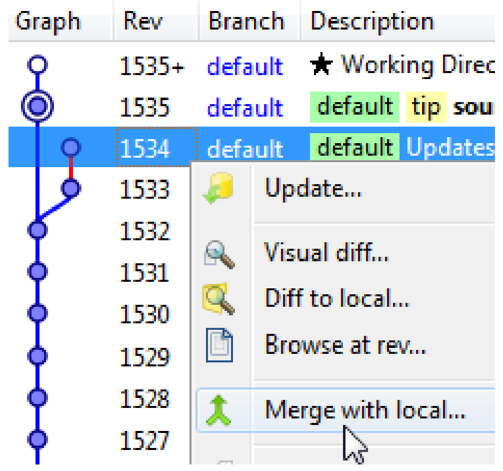


Update - pdstore

Update to: 2249

Options:
- [ ] Discard local changes, no backup (-C/--clean)
- [✓] Always merge (when possible)
- [✓] Automatically resolve merge conflicts where possible
- [ ] Always show command log

Graph	Rev	Branch	Description
	733+	default	★ Working Direct
	738	default	default tip Last
	737	default	
	736	default	
	735	default	
	734	default	
	733	default	
	732	default	



Graph	Rev	Branch	Description
	1535+	default	★ Working Direc
	1535	default	default tip sou
	1534	default	default Updates
	1533		
	1532		
	1531		
	1530		
	1529		
	1528		
	1527		

Lab Setup

- Create one repo: repo1
- Create one file in the working copy of repo1
- Add it to version control, can you figure out how?
- Commit it.
- Clone the repo: repo2

Lab Task 1

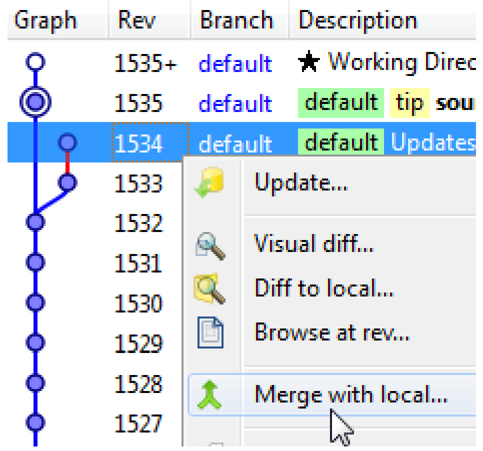
- Change a file in the working copy of repo1
- Commit to repo1.
- What is on repo2?
- Pull on repo 2. What do you see?
- Update on repo 2

Hg Push

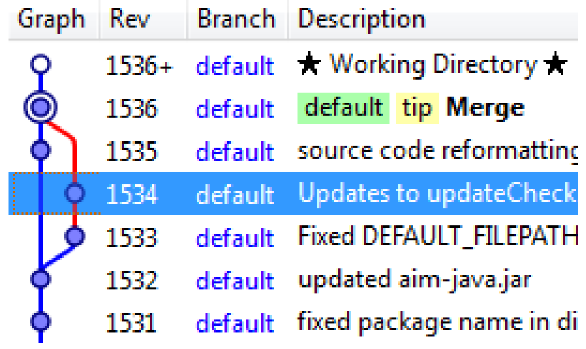
Push regularly to integrate your changes.

Have others committed versions on a remote branch that you have committed to locally?

1. If no, **push** will succeed and the local versions will be in the remote repo
2. If yes, i.e. others have committed versions on a branch you have committed to locally:
   - You need to **merge** your versions with their versions
   - When local branches and corresp. remote branches are merged, **push** succeeds



Graph	Rev	Branch	Description
	1535+	default	★ Working Direc
	1535	default	default tip sou
	1534	default	default Updates
	1533		
	1532		
	1531		
	1530		
	1529		
	1528		
	1527		



Graph	Rev	Branch	Description
	1536+	default	★ Working Directory ★
	1536	default	default tip Merge
	1535	default	source code reformatting
	1534	default	Updates to updateCheck
	1533	default	Fixed DEFAULT_FILEPATH
	1532	default	updated aim-java.jar
	1531	default	fixed package name in di

Lab Task 2

- Change something in the working copy of repo2.
- Commit to repo2
- What is on repo1?
- Push in repo 2 to repo1. What do you see?

Lab Task 3

- Change a file in the working copy of repo1
- Commit to repo1.
- Change a different file in the working copy of repo2.
- Commit to repo2
- Pull on repo 2. What do you see?
- Push in repo 2 to repo1. What happens?

Lab Task 4

- Reproduce the conflict from Lab task 3:
- Use “merge” on repo2, applying it to the two latest commits. Can you figure out how to do that?
- Push in repo 2 to repo1. What happens?

Lab Task 5

- Change a file in the working copy of repo1
- Commit to repo1.
- Change a different file in the working copy of repo2.
- Pull on repo 2. What do you see?
- Update, using “merge local changes”
- Now commit on repo2.
- Push in repo 2 to repo1. What happens?
- What is the difference in the version space to lab task 4?

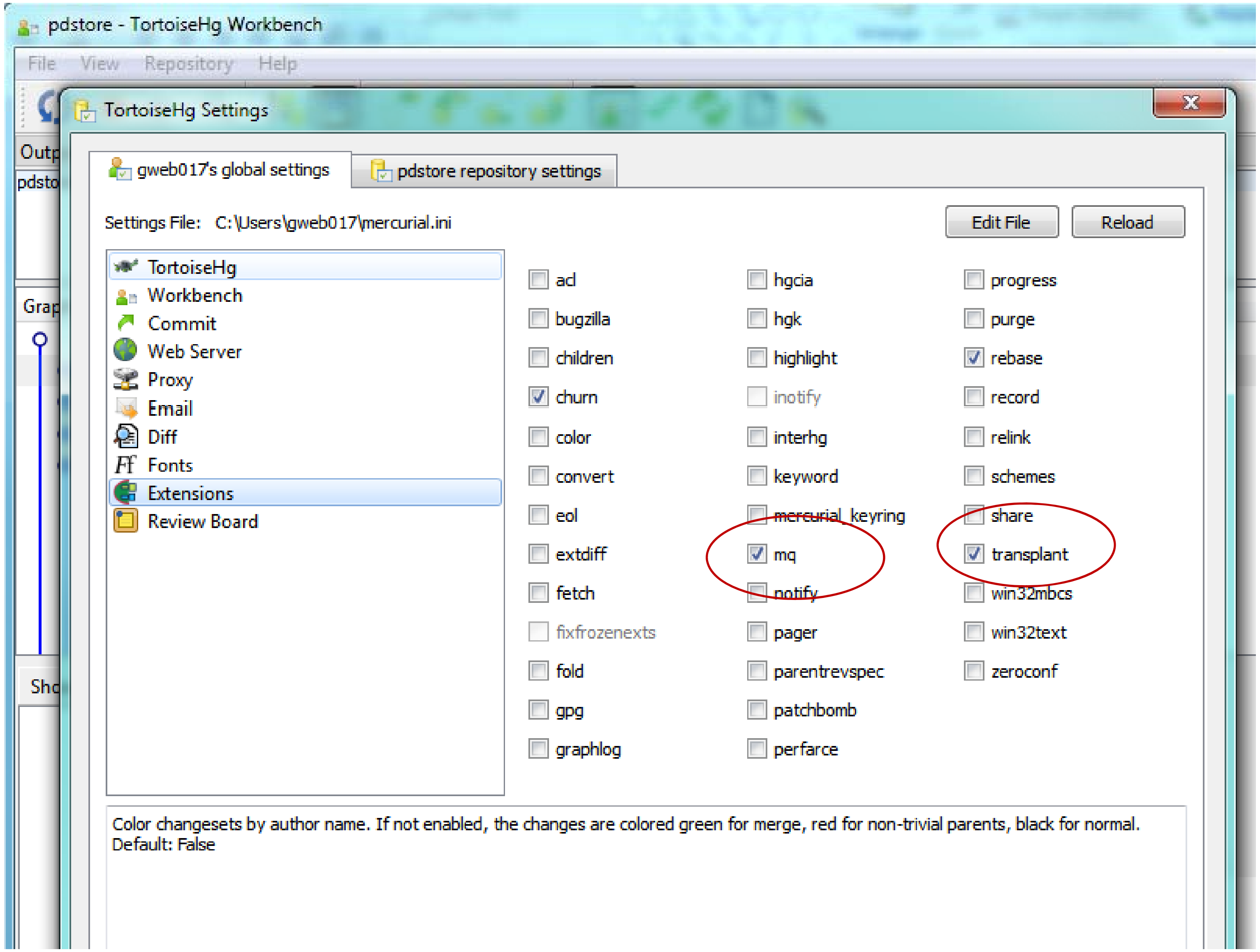
Lab Task 6

- Change a file in the working copy of repo1
- Commit to repo1.
- Change a different file in the working copy of repo2.
- Commit to repo2 under a named branch **tryout**
- Pull on repo 2. What do you see?
- Push in repo 2 to repo1. What happens?
- What is the difference to before?

Lab Task 7

- Reproduce the final situation of Lab Task 6:
- Use merge on repo2, on default branch and tryout branch
- Push in repo 2 to repo1. What happens?
- Compare with Lab Task 4

Extensions Tab


pdstore - TortoiseHg Workbench

TortoiseHg Settings

gweb017's global settings | pdstore repository settings

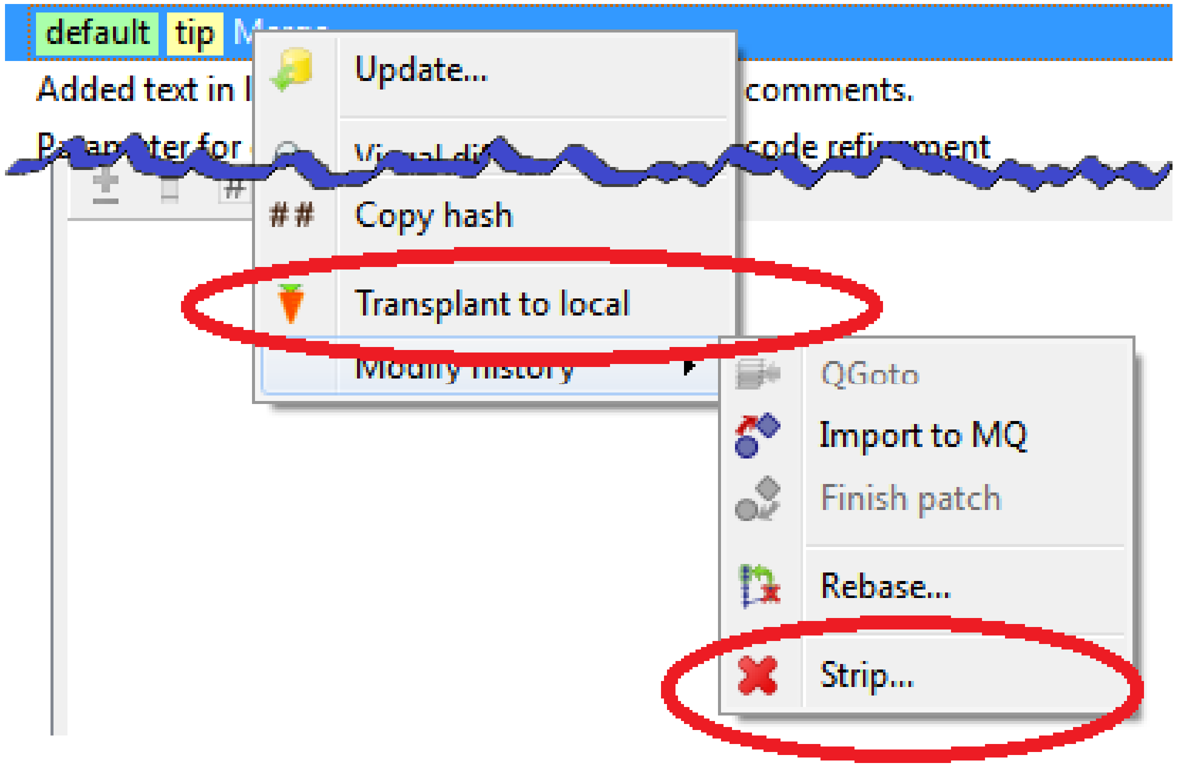
Settings File: C:\Users\gweb017\mercurial.ini

☐ acl	☐ hgcia	☐ progress
☐ bugzilla	☐ hgk	☐ purge
☐ children	☐ highlight	✓ rebase
✓ churn	☐ inotify	☐ record
☐ color	☐ interhg	☐ relink
☐ convert	☐ keyword	☐ schemes
☐ eol	☐ mercurial_keyring	☐ share
☐ extdiff	✓ mq	✓ transplant
☐ fetch	☐ notify	☐ win32mbcs
☐ fixfrozenexts	☐ pager	☐ win32text
☐ fold	☐ parentrevspec	☐ zeroconf
☐ gpg	☐ patchbomb	
☐ graphlog	☐ perfarce	

Color changesets by author name. If not enabled, the changes are colored green for merge, red for non-trivial parents, black for normal.
Default: False

Extensions

- Transplant: Moving a commit from one branch to another
- Strip: deleting one or several commits on a local repository.



Lab Task E1

- Reproduce the conflict from Lab task 3:
- Strip one head. Can you figure out how to do that?
- Push in repo 2 to repo1. What happens?
- Stripping changes is NOT the standard way to deal with conflicts, the standard way is merging.

Lab Task E2

- Reproduce the conflict from Lab task 3:
- Transplant one head to the other branch. Can you figure out how to do that?
- Strip one head, which one makes sense?
- Push in repo 2 to repo1.
- Compare with merging.

Work routine for committing

- If you think you are ready to commit

1. Run tests. If they fail, do not commit
2. **Pull** commits
3. If there are changes, **update** to the latest commit, back to 1.
4. Check all files that need to be committed.
5. **Commit**
6. **Push** (this is crucial, commit alone doesnt make it available for others.

Version Control Best Practices

1. Complete **one change** at a time and commit it
   - If you committing several changes together you cannot undo/redo them individually
   - If you don't commit and your hard disk crashes…
   - Continuous integration (see XP)
2. Only commit changes that preserve system **integrity**
   - No "breaking changes" that make compilation or tests fail
3. Commit only **source files** (e.g. not **`.class`** files)
4. Write a **log entry** for each change
   - What has been changed and why
5. **Communicate** with the other developers
   - See who else is working on a part before changing it
   - Discuss and agree on a design
   - Follow the project guidelines & specifications