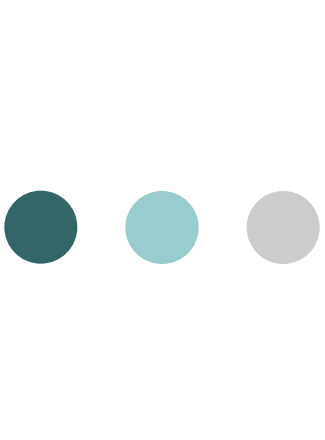


Hg Tutorial

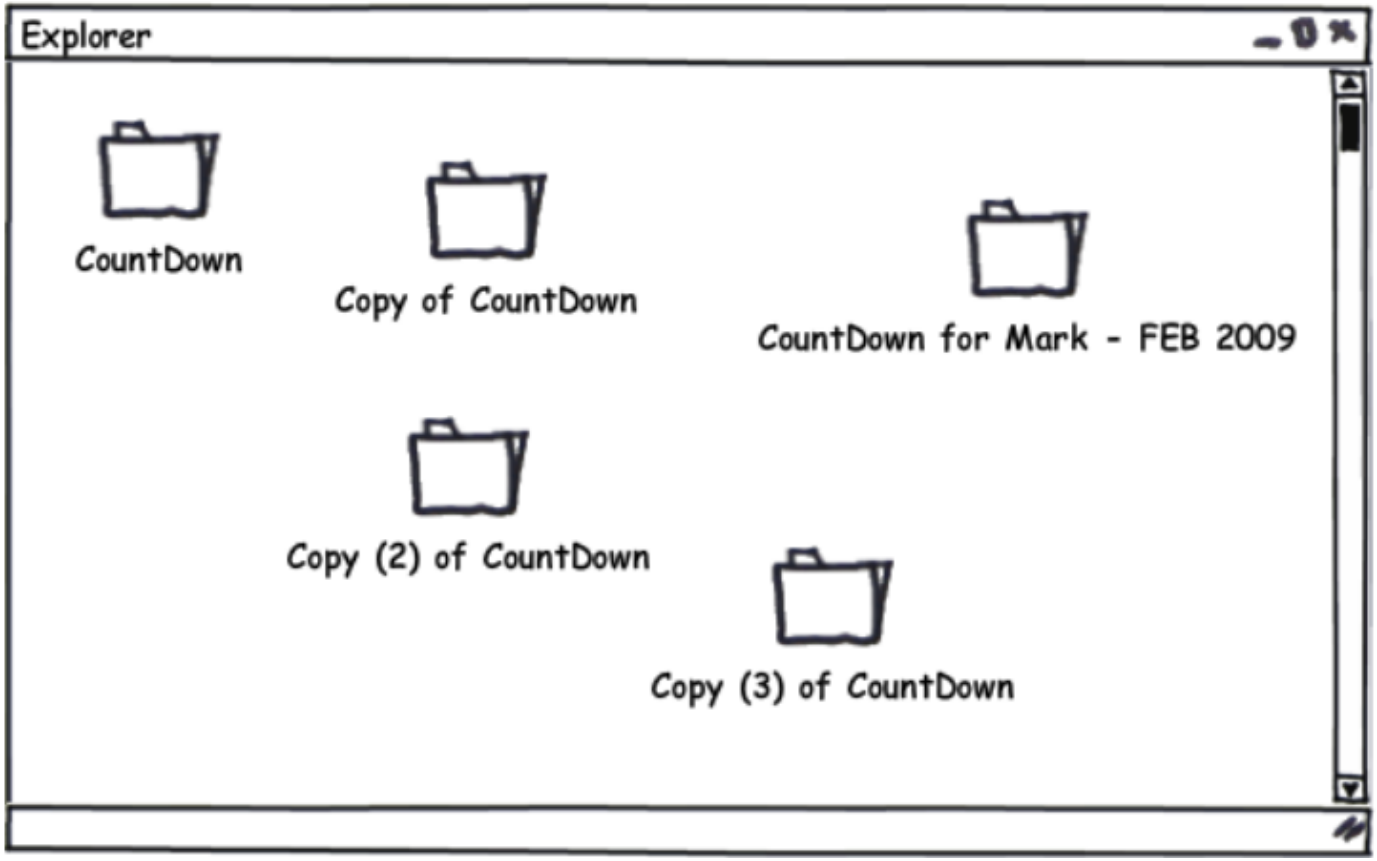
For : COP 3330.

Object oriented Programming (Using C++)

http://www.compgeom.com/~piyush/teach/3330

Biswas Parajuli

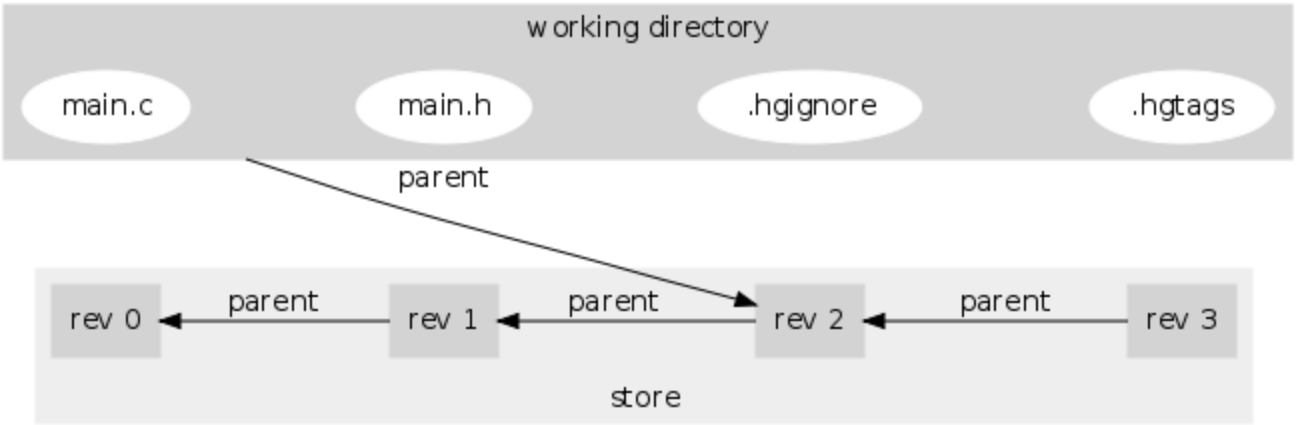
Need for Version Control



http://hginit.com/01.html

Repository

- Working directory:
  - has a copy of the project files in a certain version
- Store:
  - holds complete history of the project



https://www.mercurial-scm.org/wiki/UnderstandingMercurial

Example Project

- Say you have a directory *proj_0* with:
  - Makefile
  - hello.cpp
  - README.rst

Creating local hg repo

1. $ cd proj_0
2. $ ls –a
   - lists project files + hidden dirs ‘.’ and ‘..’
3. $ hg init
   - create a new repository in the current directory
4. $ ls –a
   - New hidden directory ‘.hg’
   - ‘.hg’ will hold the history of the working dir
5. $ ls .hg
   - shows files ‘00changelog.i’, ‘requires’ and directory ‘store’

Note:
‘$’ is not a part of the command.

Hg commands

$ hg
Mercurial Distributed SCM

basic commands:

add	add the specified files on the next commit
annotate	show changeset information by line for each file
clone	make a copy of an existing repository
commit	commit the specified files or all outstanding changes
diff	diff repository (or selected files)
export	dump the header and diffs for one or more changesets
forget	forget the specified files on the next commit
init	create a new repository in the given directory
log	show revision history of entire repository or files
merge	merge another revision into working directory
pull	pull changes from the specified source
push	push changes to the specified destination
remove	remove the specified files on the next commit
serve	start stand-alone webserver
status	show changed files in the working directory
summary	summarize working directory state
update	update working directory (or switch revisions)

(use "hg help" for the full list of commands or "hg -v" for details)

Per-repo config file

- Create a new file *.hg/hgrc* inside *proj_0*
  - Configuration file with sections
  - Each section led by [section] header followed by name = value entries
- “ui” section
  - User Interface controls
  - username => who made the changes?
- “paths” section
  - Alias for location of the repo
  - Can be a remote URL or a local directory

add to repo

```
$ echo “Print Hello” > README.rst
$ hg add README.rst
```

- README.rst will now be version controlled
- It will be added to the repo in the next check in or commit

hg add

- Do not add compiled binaries, .so or other files which are produced based on the source files
- Avoid adding files with sensitive info
- Add only the source files

check status

```
$ hg status
A README.rst
? Makefile
? hello.cpp
```

- Status before committing
- $ hg status
  - shows status of all files in the working dir
- $ hg st README.rst
  - shows status of the given file
- $ hg status –mar
  - Show only those files which were modified (m), added (a) or removed (r)
- For other status codes, run “hg help status”
  - ? = not tracked

hg commit

$ hg commit –m “Added title to README”

- Commit the change you made to the source files to the repo
- “-m” for commit => commit message
- Informative commit. Searchable commits.
- A new commit == A revision

add remaining and commit

```
$ echo “int main(){return 0;}” > hello.cpp
$ hg add hello.cpp
$ hg ci –m “Initialized hello.cpp”
```

```
$ echo “# Makefile to build hello.cpp” > Makefile
$ hg add Makefile
$ hg ci –m “Checking in Makefile”
```

- How many revisions so far?

History

```
$ hg log
changeset:   2:350b60c55f99
tag:         tip
user:        bparaj
date:        Sun Jan 15 19:32:15 2017 -0500
summary:     checking in Makefile

changeset:   1:a9204d84057c
user:        bparaj
date:        Sun Jan 15 19:31:36 2017 -0500
summary:     Initialized hello.cpp

changeset:   0:e3475d50b16a
user:        bparaj
date:        Sun Jan 15 19:29:16 2017 -0500
summary:     Added title to README
```

History

```
$ hg tip
changeset:   2:350b60c55f99
tag:         tip
user:        bparaj
date:        Sun Jan 15 19:32:15 2017 -0500
summary:     checking in Makefile

$ hg log -r 1:2
changeset:   1:a9204d84057c
user:        bparaj
date:        Sun Jan 15 19:31:36 2017 -0500
summary:     Initialized hello.cpp

changeset:   2:350b60c55f99
tag:         tip
user:        bparaj
date:        Sun Jan 15 19:32:15 2017 -0500
summary:     checking in Makefile
```

Navigate Revisions

- Update working dir with a specific version
- Like a time travel
- Where are we?
  - $ hg identify
- Lets go to revision 0
  - $ hg update –r 0
  - Check your working dir. Makefile and hello.cpp are gone!
- Can we go to the latest version?
  - $ hg update
  - Phew!

Tag Revisions

- $ hg tag -r 1 cpp_added
  - Give user defined symbolic name “cpp_added” to revision 1

```
$ hg log -r 1
changeset:   1:a9204d84057c
tag:         cpp_added
user:        bparaj
date:        Sun Jan 15 19:31:36 2017 -0500
summary:     Initialized hello.cpp
```

Mistakes

- Mistaken modification to a file but you have not committed the change yet
- Undo change to the file with “hg revert”.

```
proj_0$ echo "unwanted edit" >> hello.cpp
proj_0$ cat hello.cpp
int main(){return 0;}
unwanted edit
proj_0$ hg stat -mard
M hello.cpp
proj_0$ hg revert hello.cpp
proj_0$ hg stat -mard
proj_0$ cat hello.cpp
int main(){return 0;}
```

Mistakes

- Accidentally added “a.out” but you have not committed yet
- Untrack it with “hg forget”

```
proj_0$ hg add a.out
proj_0$ hg stat -mard
A a.out
proj_0$ hg forget a.out
proj_0$ hg stat -mard
proj_0$
```

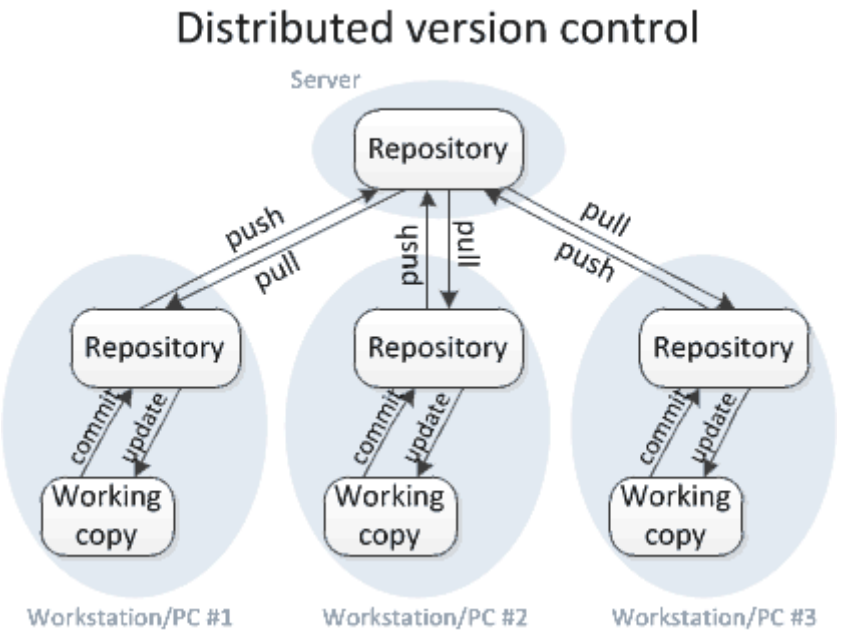
Mistakes

- You made an incomplete or a wrong commit.
- To fix it:
  1. Make correct changes
  2. Use “hg commit --amend” to overwrite/alter the previous commit

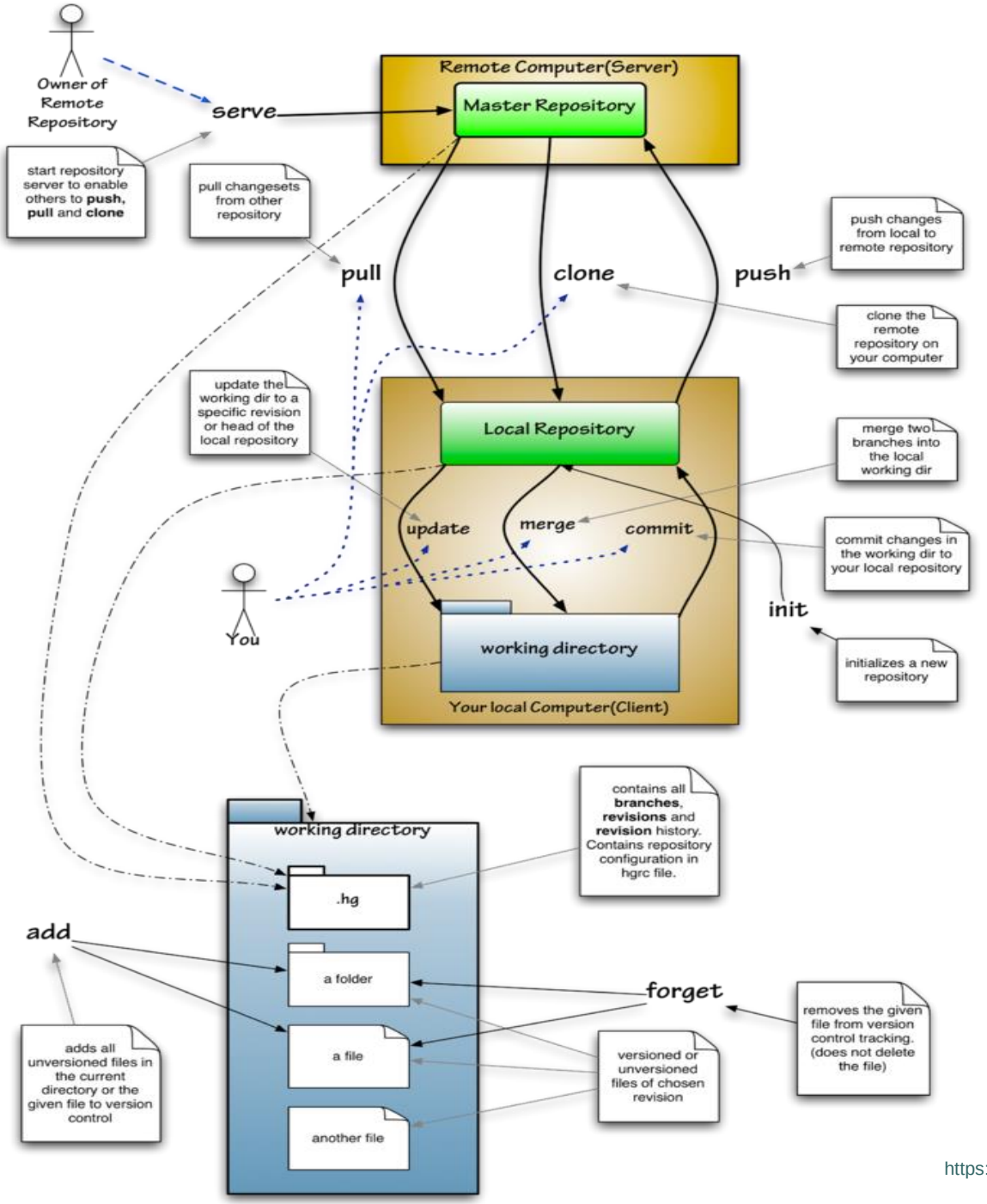
```
$ echo "Wrote wrong code" >> hello.cpp
$ hg commit -m "accidental commit"
$ vim hello.cpp  # make correct changes to hello.cpp
$ hg commit --amend
```

Distributed Version Control

- Collaborative software development
- Each dev copies the whole repo in her local machine



https://homes.cs.washington.edu/~mernst/advice/version-control.html



https://en.wikipedia.org/wiki/Mercurial

hg clone

- Obtain copy of a remote master repo
- $ hg clone *URL*

Push changes

```
$ hg push
```

- With “hg commit”, the updates are committed only on the local repo
- Other devs (developers) should have access to the changes you made
- Where to push? To the master repo specified in the default entry in [paths] section in “.hg/hgrc” file.

Pull and Update

$ hg pull

- Retrieve changes from the master repo to your local repo
- Basically, the two repos are synced … but not the actual source (working) files

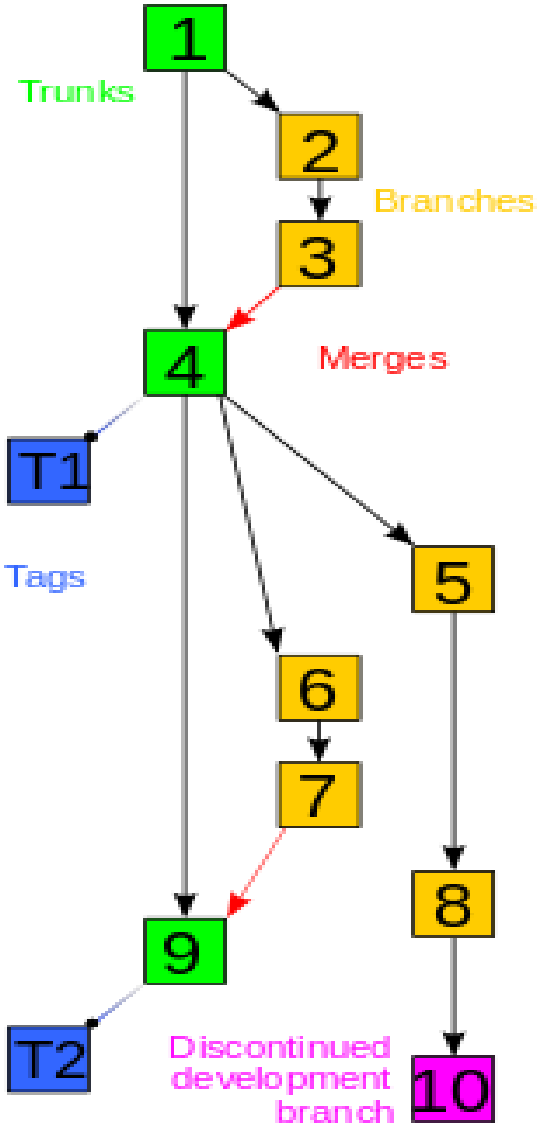
$ hg update

- Update the local source (working) files with the changesets pulled from master repo

hg merge

- Combine two changesets for multiple files or even the same files but on non-overlapping sections
- Graphically: join two branches at their current heads

Version Control: Graph



https://en.wikipedia.org/wiki/Version_control

Merge Conflict

- Two independent changesets on overlapping sections of the same files
- Dev should visually verify and select the correct changeset for the overlapping parts

Rule of Thumb:
Pull and update before you commit and push.

References

- Hg Init: a Mercurial tutorial (http://hginit.com)
- https://www.mercurial-scm.org/wiki/
- Mercurial: The Definitive Guide
  - http://hgbook.red-bean.com/read
- hg on command line