Fast, Flexible and Fun: Revision Control with Mercurial

Martin Geisler
<mg@aragost.com>

PyZH: Zurich Python User Group
June 29th, 2010
About the Speaker

Martin Geisler:
- huge fan of Python : -)
- core Mercurial developer:
  - reviews patches from the community
  - helps users in our IRC channel
- PhD in Computer Science from Aarhus University, DK
  - spent 2005 as an exchange student at ETH Zürich
- now working at aragost Trifork, Switzerland
Outline

Introduction

Using Mercurial
- Workflows
- Branches
- The Underlying Model
- Using History

Cool Extensions
- Changing History
- Talking to Other Systems
- Third-Party Tools

Demonstration

Wrapping Up
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
**Key Concepts**

- **hello.c**
- **Makefile**
- **commit**
- **update**

Alice
Key Concepts

hello.c
Makefile

commit
update

push
pull

Alice
**Key Mercurial Commands**

Local commands:
- `hg commit`: save a snapshot into the current repository.
- `hg update`: checkout revision into working directory.
- `hg merge`: join different lines of history.

Network commands:
- `hg pull`: retrieve changesets from another repository.
- `hg push`: send your changesets to another repository.
**Moving Changesets Around**

People have read-only access (e.g., `hg serve`):

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th></th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carla</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

0 / 41
Moving Changesets Around

People have read-only access (e.g., `hg serve`):

Alice 0 → $A_1$

Bob 0

Carla 0
Moving Changesets Around

People have read-only access (e.g., hg serve):

Alice

0 → A₁ → A₂

Bob

0

Carla

0
Moving Changesets Around

People have read-only access (e.g., hg serve):

Alice

Bob

Carla

hg serve
Moving Changesets Around

People have read-only access (e.g., `hg serve`):

Alice

0 → $A_1$ → $A_2$ → $B_1$

Bob

0 → $B_1$

pull

Carla

0
Moving Changesets Around

People have read-only access (e.g., `hg serve`):

Alice

0 → A₁ → A₂ → A₃

Bob

0 → B₁

Carla

0

aragost Trifork
Moving Changesets Around

People have read-only access (e.g., hg serve):

Alice

0 → $A_1$ → $A_2$ → $A_3$

Bob

0 → $B_1$

Carla

0 → $C_1$
Moving Changesets Around

People have read-only access (e.g., `hg serve`):

Alice

0 → $A_1$ → $A_2$ → $A_3$ → $B_1$

Bob

0 → $B_1$

Carla

0 → $C_1$ → $C_2$
People have read-only access (e.g., `hg serve`):
People have read-only access (e.g., `hg serve`):

Alice

0 → A₁ → A₂ → A₃

Bob

0 → B₁

Carla

0 → C₁ → C₂ → C₃

B₁
People have read-only access (e.g., *hg serve*):

Alice

0 → A₁ → A₂ → A₃

Bob

0 → B₁

Carla

0 → C₁ → C₂ → C₃

pull

A₁ → A₂ → A₃

B₁ → B₁
People have read-only access (e.g., `hg serve`):
OUTLINE

INTRODUCTION

USING MERCUKAL
   Workflows
   Branches
   The Underlying Model
   Using History

COOL EXTENSIONS
   Changing History
   Talking to Other Systems
   Third-Party Tools

DEMONSTRATION

WRAPPING UP
Introduction

Using Mercurial

Workflows
Branches
The Underlying Model
Using History

Cool Extensions

Changing History
Talking to Other Systems
Third-Party Tools

Demonstration

Wrapping Up
Distributed Revision Control

Mercurial duplicates the history on many servers:
Distributed Revision Control

Mercurial duplicates the history on many servers:

- Alice
- Alice’s Laptop
- Bob
- Carla

Diagram showing connections between Alice, Alice’s Laptop, Bob, and Carla.
Distributed Revision Control

Mercurial duplicates the history on many servers:

- Alice
- Alice’s Laptop
- Server
- Bob
- Carla
Distributed Revision Control

Mercurial duplicates the history on many servers:
Mercurial scales from a single team...
Workflow Between Company Divisions

... to enterprise-wide development...:
Workflow Between Two Computers

...to working with yourself:

Alice's Desktop

bitbucket.org

Alice's Laptop
Outline

Introduction

Using Mercurial
- Workflows
- Branches
  - The Underlying Model
  - Using History

Cool Extensions
- Changing History
- Talking to Other Systems
- Third-Party Tools

Demonstration

Wrapping Up
Branches

A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
Branches

A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
**Branches**

A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
**Branches**

A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
Branches

A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes
A key concept:

- parallel lines of development
- used to track releases
- used to isolate disruptive changes

[Diagram showing parallel lines of development with releases 1.0, 1.0.1, and 1.0.2]
Merging

The opposite of branching:

- combines two branches
- used to merge back bugfixes
- used to integrate feature branches
The opposite of branching:
- combines two branches
- used to merge back bugfixes
- used to integrate feature branches
Merging

The opposite of branching:

- combines two branches
- used to merge back bugfixes
- used to integrate feature branches
**Merging**

The opposite of branching:
- combines two branches
- used to merge back bugfixes
- used to integrate feature branches

![Diagram](image-url)
Branches in SVN

Subversion knows nothing about branches!
  ▶ but SVN has a cheap copy mechanism
  ▶ used for tags and branches
Branches in SVN

Subversion knows nothing about branches!

▶ but SVN has a cheap copy mechanism
▶ used for tags and branches

r10

trunk/
  hello.c
  Makefile
branches/
tags/
Subversion knows nothing about branches!
- but SVN has a cheap copy mechanism
- used for tags and branches

\[ \text{r10} \]
- trunk/
  - hello.c
  - Makefile
- branches/
- tags/

\[ \text{r11} \]
- trunk/
  - hello.c
  - Makefile
- branches/
  - goodbye/
  - hello.c
  - Makefile
- tags/
Branches in SVN

Subversion knows nothing about branches!

▶ but SVN has a cheap copy mechanism
▶ used for tags and branches

r10
trunk/
  hello.c
  Makefile
branches/
tags/

→

r11
trunk/
  hello.c
  Makefile
branches/
  goodbye/
  hello.c
  Makefile
tags/

→

r12
trunk/
  hello.c
  Makefile
branches/
  goodbye/
  hello.c
  goodbye.c
  Makefile
tags/
Merging Branches in SVN

The support is incomplete and fragile:

▶ renamed files are not merged correctly
▶ old clients will not update the merge info
Merging Branches in SVN

The support is incomplete and fragile:
  ▶ renamed files are not merged correctly
  ▶ old clients will not update the merge info

From the SVN Book:

The bottom line is that Subversion’s merge-tracking feature has an extremely complex internal implementation, and the `svn:mergeinfo` property is the only window the user has into the machinery. Because the feature is relatively new, a number of edge cases and possible unexpected behaviors may pop up.

—Version Control with Subversion

(Mercurial has robust built-in support for merging branches.)
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
A Mercurial changeset conceptually consist of:

- 0–2 parent changeset IDs:
  - root changeset has no parents
  - normal changesets have one parent
  - merge changesets have two parents
- date, username, commit message
- difference from first parent changeset
- changeset ID is computed as SHA-1 hash of the above
- makes it impossible to inject malicious code on server
Introduction

Using Mercurial
- Workflows
- Branches
- The Underlying Model
- Using History

Cool Extensions
- Changing History
- Talking to Other Systems
- Third-Party Tools

Demonstration

Wrapping Up
BROWSING THE HISTORY OF A FILE

The `hg annotate` command is invaluable:

- you see when each line was introduced
- you can quickly jump back to earlier versions

History of Mercurial’s README file:

```
3942: Basic install:
  445:
3942: $ make # see install targets
3942: $ make install # do a system-wide install
3942: $ hg debuginstall # sanity-check setup
3942: $ hg # see help
  0:
# ...
```

Better interface in `hg serve`
Searching File Content

Ever wondered when a function was introduced?

▶ hg grep can help you!

Example: When was hg forget introduced?

```
% hg grep --all 'def forget' commands.py
commands.py:8902:+:def forget(ui, repo, *pats, **opts):
commands.py:3522:-:def forget(ui, repo, *pats, **opts):
commands.py:814:-:def forget(ui, repo, file1, *files):
commands.py:814:+:def forget(ui, repo, *pats, **opts):
# ...
```
You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:
Revision Graph Bisection

You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:
You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:
You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:
You’ve found a bug! When was it first introduced? Use **hg bisect** to mark good and bad revisions:
Revision Graph Bisection

You’ve found a bug! When was it first introduced?
Use `hg bisect` to mark good and bad revisions:

```
good
  ↓
good
  ↓
good
  ↓
bad
```

- good
- good
- good
- bad

`test`
You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:
**Revision Graph Bisection**

You’ve found a bug! When was it first introduced? Use `hg bisect` to mark good and bad revisions:

```
good
  ↓
  └───→

good
  ↓
  └───→

bad
  ↑
  └───→

test
  ↑
  └───→

bad
```

24/41
You’ve found a bug! When was it first introduced?
Use hg bisect to mark good and bad revisions:
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up

aragost Trifork
You can add new functionality to Mercurial:

- ships with 30+ extensions
- wiki lists 75+ extensions
- extensions can change basically everything
- helps to keep the core small and focused
OUTLINE

INTRODUCTION

USING MERCURIAL
  Workflows
  Branches
  The Underlying Model
  Using History

COOL EXTENSIONS
  Changing History
  Talking to Other Systems
  Third-Party Tools

DEMONSTRATION

WRAPPING UP
MOVING CHANGESETS AROUND

Tired of all those merges? Use the `rebase` extension!

- Revision graph:

```
A → B → C
```
Moving Changesets Around

Tired of all those merges? Use the rebase extension!

- Revision graph:

\[\begin{array}{c}
A & \rightarrow & B & \rightarrow & C & \rightarrow & D & \rightarrow & E
\end{array}\]
Moving Changesets Around

Tired of all those merges? Use the rebase extension!

- Revision graph:

```
A → B → C → D → E
   ↓   ↓   ↓   ↓
   X   Y   Z
```

Beware: public changes should never be rebased.
Moving Changesets Around

Tired of all those merges? Use the rebase extension!

- Revision graph:

  - A → B → C → D → E
  - X → Y → Z

- Merge:

  - A → B → C → D → E → M
  - X → Y → Z
Moving Changesets Around

Tired of all those merges? Use the `rebase` extension!

- Revision graph:

```
A -> B -> C -> D -> E
  |   |   |   |
  X -> Y -> Z
```

- Merge:

```
A -> B -> C -> D -> E -> M
  |   |   |   |
  X -> Y -> Z
```

- Rebase:

```
A -> B -> C -> D -> E
  |   |   |   |
  X -> Y -> Z -> D' -> E'
```
Tired of all those merges? Use the rebase extension!

- **Revision graph:**
  
  \[
  \begin{array}{cccccc}
  A & B & C & D & E \\
  \quad & X & Y & Z & \\
  \end{array}
  \]

- **Merge:**
  
  \[
  \begin{array}{cccccc}
  A & B & C & D & E & M \\
  \quad & X & Y & Z & \\
  \end{array}
  \]

- **Rebase:**
  
  \[
  \begin{array}{cccccc}
  A & B & C & D & E \\
  \quad & X & Y & Z & D' & E' \\
  \end{array}
  \]

- **Beware:** public changes should never be rebased.
Inspired by *git rebase -i*, **histedit** lets you

- reorder changesets:

  \[ A \rightarrow B \rightarrow C \quad \sim \rightarrow \quad A \rightarrow C' \rightarrow B' \]
Editing History

Inspired by `git rebase -i`, **histedit** lets you

- reorder changesets:
  
  \[
  A \rightarrow B \rightarrow C \quad \sim \sim \quad A \rightarrow C' \rightarrow B'
  \]

- fold changesets:
  
  \[
  A \rightarrow B \rightarrow C \quad \sim \sim \quad A \rightarrow BC
  \]
Inspired by \texttt{git rebase -i}, \texttt{histedit} lets you

- reorder changesets:
  
  \[
  \xymatrix{A \ar[r] & B \ar[r] & C \ar[r] & A \ar[r] & C' \ar[r] & B' }
  \]

- fold changesets:
  
  \[
  \xymatrix{A \ar[r] & B \ar[r] & C \ar[r] & A \ar[r] & BC }
  \]

- drop changesets:
  
  \[
  \xymatrix{A \ar[r] & B \ar[r] & C \ar[r] & A \ar[r] & C' }
  \]
Inspired by `git rebase -i`, `histedit` lets you

- reorder changesets:
  \[ A \rightarrow B \rightarrow C \leadsto \quad A \rightarrow C' \rightarrow B' \]

- fold changesets:
  \[ A \rightarrow B \rightarrow C \leadsto \quad A \rightarrow BC \]

- drop changesets:
  \[ A \rightarrow B \rightarrow C \leadsto \quad A \rightarrow C' \]

- edit changesets:
  \[ A \rightarrow B \rightarrow C \leadsto \quad A \rightarrow X \rightarrow B' \rightarrow C' \]
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
The convert extension can import history:

- CVS, SVN, Git, Bazaar, Darcs, ...
- incremental conversion
- many options for fiddling with branches, authors, ...

Interestingly, convert can import from Mercurial:

- --filemap lets you exclude and rename files
- --branchmap lets you rename branches
The **convert** extension can import history:

- CVS, SVN, Git, Bazaar, Darcs, ...
- incremental conversion
- many options for fiddling with branches, authors, ...

Interestingly, **convert** can import from Mercurial:

- `--filemap` lets you exclude and rename files
- `--branchmap` lets you rename branches
Interfacing with Subversion

The `hgsubversion` extension let’s you:

- use `hg clone` on a SVN URL
- use `hg pull` to convert new SVN revisions
- use `hg push` to commit changesets to SVN server

Goal: make `hg` a better Subversion client than `svn`!
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
THIRD-PARTY TOOLS

Tools with Mercurial support:

▶ Shell integration: TortoiseHg (Windows, Mac, Linux)
▶ IDEs: Eclipse, NetBeans, IntelliJ, Visual Studio, Emacs…
▶ Project Support: Trac, JIRA, Maven, Hudson, BuildBot…
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
Live Demo!
Outline

Introduction

Using Mercurial
  Workflows
  Branches
  The Underlying Model
  Using History

Cool Extensions
  Changing History
  Talking to Other Systems
  Third-Party Tools

Demonstration

Wrapping Up
Mercurial in a Nutshell

Mercurial changes the way you develop:

▶ simple yet strong model for both branching and merging
▶ power tool instead of necessary evil
▶ light-weight and snappy
More Information

- Mercurial homepage:
  http://mercurial.selenic.com/

- *Mercurial: The Definitive Guide:*
  http://hgbook.red-bean.com/

- Getting Started:
  http://mercurial.aragost.com/kick-start/
  http://mercurial.ch/
  http://hginit.com/

- Some free Mercurial hosting sites:
  http://bitbucket.org/
  http://code.google.com/
  http://sourceforge.net/
  http://www.codeplex.com/ (Microsoft)
Contact

Please get in touch if you have more questions or have considered using Mercurial in your organization:

▶ Email: mg@aragost.com
▶ IRC: mg in #mercurial on irc.freenode.net
Mercurial Contributors

From http://ohloh.net/p/mercurial/map:

Showing 50 of 325 contributors.
Mercurial Contributors

From http://ohloh.net/p/mercurial/map:

Thank you!

Showing 50 of 325 contributors.

aragost Trifork