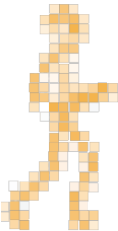


使用Mercurial进行分布式版本管理

Martin Geisler
<mg@aragost.com>

CCDC
Cambridge, UK
June 25th--27th, 2012

译者：施展
<g.shizhan.g@gmail.com>
华中科技大学 武汉光电国家实验室
October 22, 2012



About the Original Speaker

Martin Geisler:

- ▶ core Mercurial developer:
 - ▶ reviews patches from the community
 - ▶ helps users in our IRC channel
- ▶ works at aragost Trifork, Zurich:
 - ▶ offers professional Mercurial support
 - ▶ customization, migration, training
 - ▶ advice on best practices



关于中译稿

施展:

- ▶ 动机:

- ▶ Mercurial, Python 确实很有用, 满足各种兴趣爱好、还有研究和教学需要。
- ▶ 结构化写作 —— LaTeX。
- ▶ 好想法孕育好工具, 好工具帮助塑造好习惯。

- ▶ 在武汉光电国家实验室工作, 副研究员:

- ▶ 专业方向: 计算机系统结构
- ▶ 研究领域: 海量信息存储、存储服务、存储管理
- ▶ 课程教学: 计算机系统结构、外部设备、人机交互



大纲

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Mercurial 是什么?

主要特点:

- ▶ 快速、分布式的版本管理系统
 - ▶ 强壮的分支 与 合并
 - ▶ 自由与开放
 - ▶ 有商业支持: aragost Trifork and others



Mercurial 是什么?

主要特点:

- ▶ 快速、分布式的版本管理系统
 - ▶ 强壮的分支 与 合并
 - ▶ 自由与开放
 - ▶ 有商业支持: aragost Trifork and others
- ▶ 可在各种平台下安装 Windows, Mac OS X, Linux, ...
 - ▶ TortoiseHg 跨平台图形界面
 - ▶ MacHg 还有 SourceTree 快速高效的 Mac OS X 原生界面
 - ▶ 提供插件支持 for MS Visual Studio, Eclipse, ...



Mercurial 是什么?

主要特点:

- ▶ 快速、分布式的版本管理系统
 - ▶ 强壮的分支 与 合并
 - ▶ 自由与开放
 - ▶ 有商业支持: aragost Trifork and others
- ▶ 可在各种平台下安装 Windows, Mac OS X, Linux, ...
 - ▶ TortoiseHg 跨平台图形界面
 - ▶ MacHg 还有 SourceTree 快速高效的 Mac OS X 原生界面
 - ▶ 提供插件支持 for MS Visual Studio, Eclipse, ...
- ▶ 很好用
 - ▶ 所有命令均内建 (可扩展) 帮助
 - ▶ 命令集与 CVS 和 SVN 相仿
 - ▶ 危险命令交给扩展来完成



谁在用?

Mercurial 拥有这些知名用户:

- ▶ Oracle for Java, OpenSolaris, NetBeans, OpenOffice, ...
- ▶ Mozilla for Firefox, Thunderbird, ...
- ▶ Google
- ▶ 真不少...



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

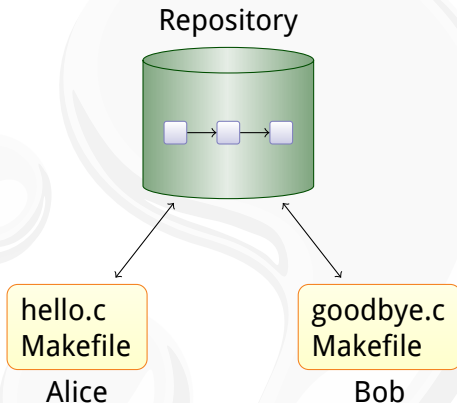
扩展

与 Git 比较

归纳总结

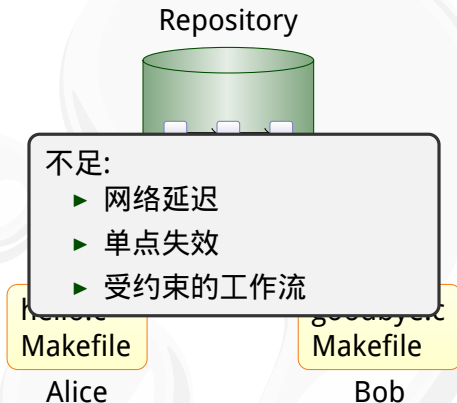
集中式版本管理

单个库，多个工作副本：



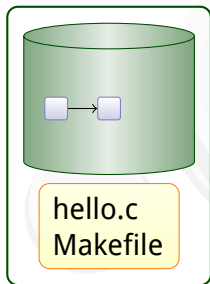
集中式版本管理

单个库，多个工作副本：

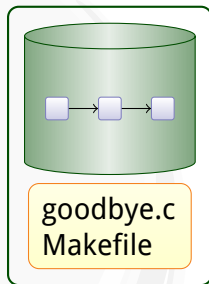


分布式版本管理

Mercurial 在许多服务器上均保存历史:



Alice

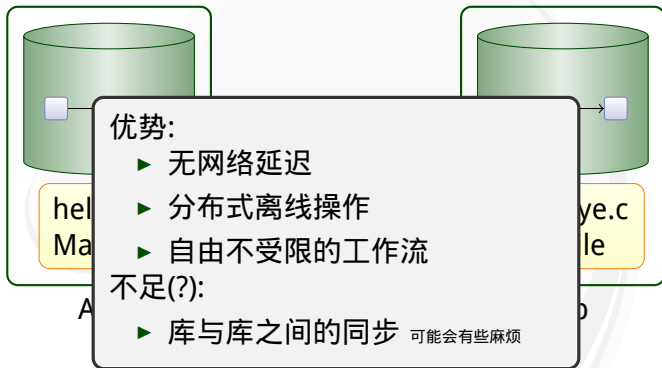


Bob



分布式版本管理

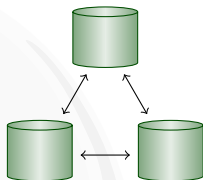
Mercurial 在许多服务器上均保存历史:



为何需要分布式?

分布式版本管理工具可以提供:

- ▶ 离线提交
- ▶ 本地操作丰富而快捷
- ▶ 很灵活



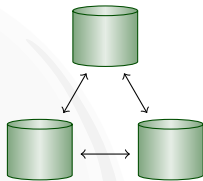
为何需要分布式?

分布式版本管理工具可以提供:

- ▶ 离线提交
- ▶ 本地操作丰富而快捷
- ▶ 很灵活

还有:

- ▶ 细粒度 fine-grained 提交
- ▶ 可检索历史记录
- ▶ 分支和合并不再那么可怕 (均有副本)
- ▶ 工作流 workflows 更好



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice



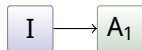
Bob



传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice



`commit`

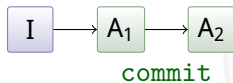
Bob



传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice



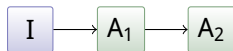
Bob



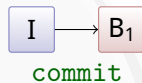
传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice

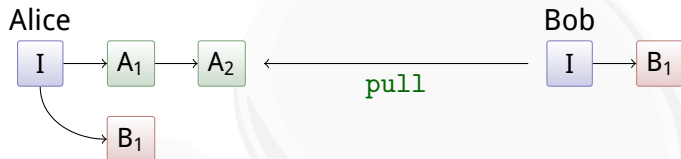


Bob



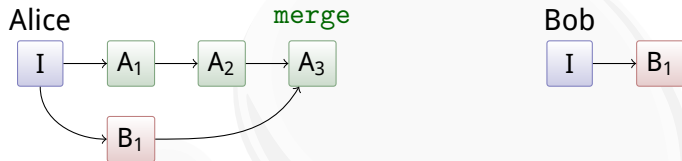
传递变更集 Changesets

提取 Pull 和 合并 merge:



传递变更集 Changesets

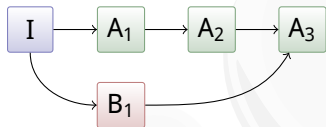
提取 Pull 和 合并 merge:



传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice



Bob



变更集 Changesets:

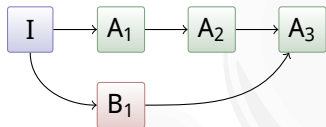
- ▶ 原子的，库范围的快照
- ▶ 与 ClearCase UCM 里的 **baseline** 相仿



传递变更集 Changesets

提取 Pull 和 合并 merge:

Alice



Bob



变更集 Changesets:

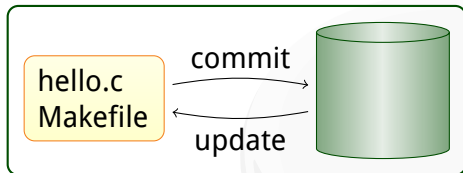
- ▶ 原子的，库范围的快照
- ▶ 与 ClearCase UCM 里的 **baseline** 相仿

合并 Merging:

- ▶ 找出这些变更集 A_2 and B_1 的共同祖先: I
- ▶ 在这些变更集上归并 I , A_2 , and B_1
- ▶ 持续归并 \Rightarrow 使共同祖先接近于库顶 tip



Mercurial 关键命令



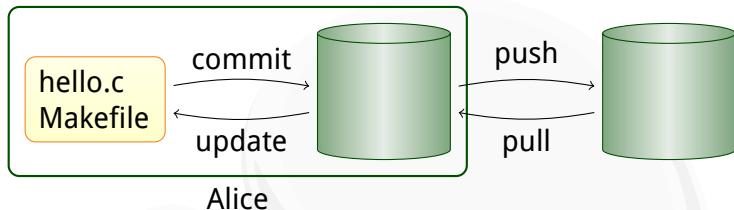
Alice

本地命令:

- ▶ `hg commit`: 在当前库中保存一份快照
- ▶ `hg update`: 从库中取出一份版本作为工作副本
- ▶ `hg merge`: 合并不同版本历史路径



Mercurial 关键命令



本地命令:

- ▶ `hg commit`: 在当前库中保存一份快照
- ▶ `hg update`: 从库中取出一份版本作为工作副本
- ▶ `hg merge`: 合并不同版本历史路径

网络命令:

- ▶ `hg pull`: 从远程库取变更集
- ▶ `hg push`: 将变更集推送至远程库



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

workflow Workflows

分支 Branches

底层模型

使用历史

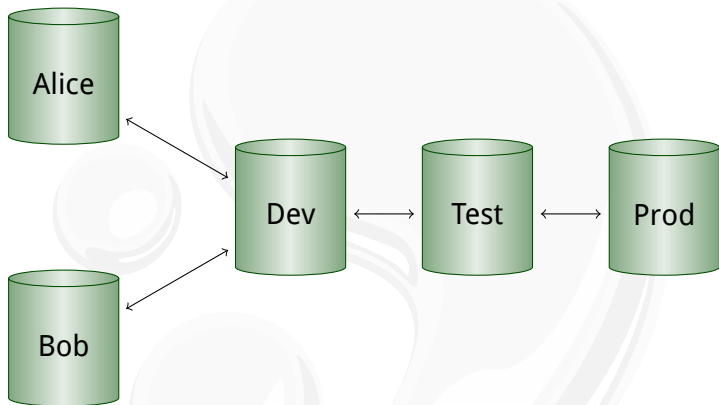
扩展

与 Git 比较

归纳总结

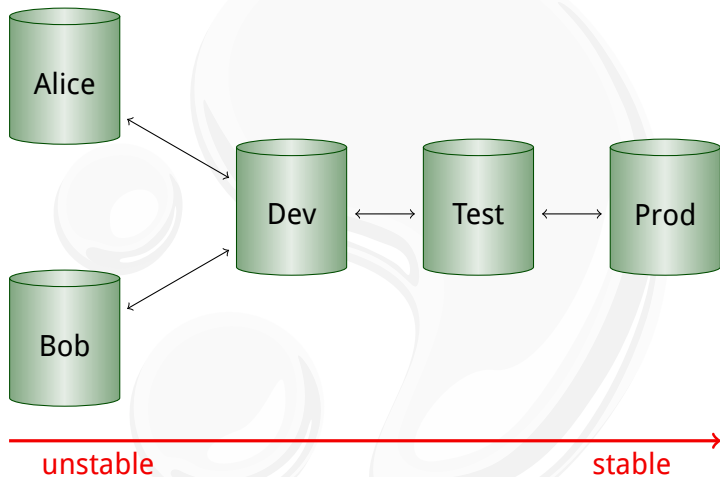
小组工作流

一个小组怎么用 Mercurial…:



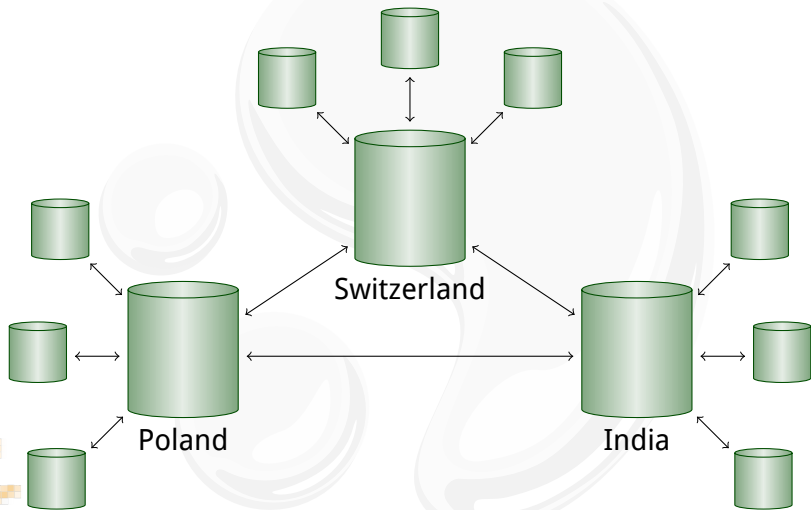
小组工作流

一个小组怎么用 Mercurial…:



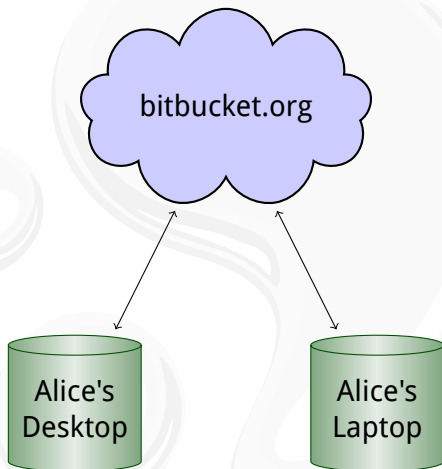
分部工作流

…对于包含若干分部的大公司而言…:



个人 workflow

…如果单干的话:



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

发布分支

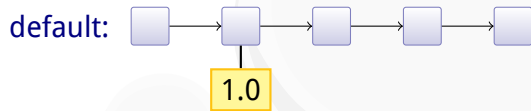
default: 



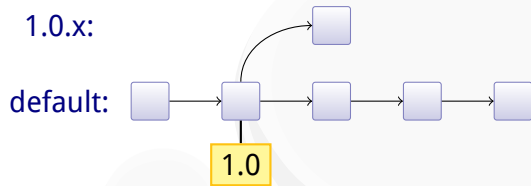
发布分支



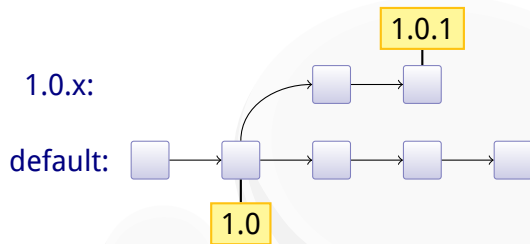
发布分支



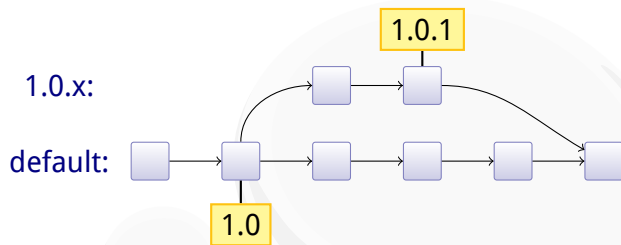
发布分支



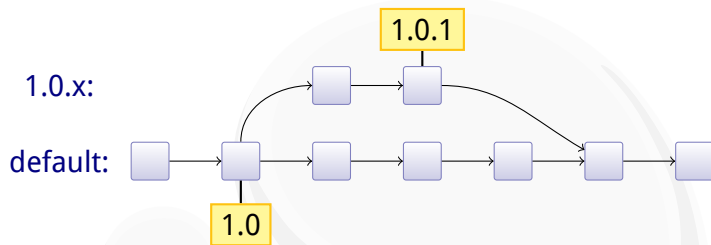
发布分支



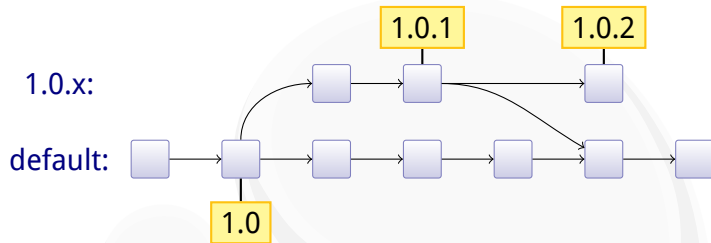
发布分支



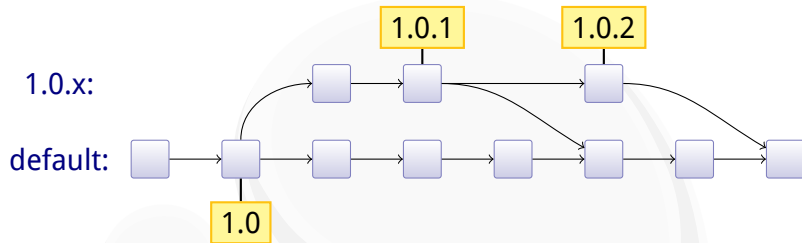
发布分支



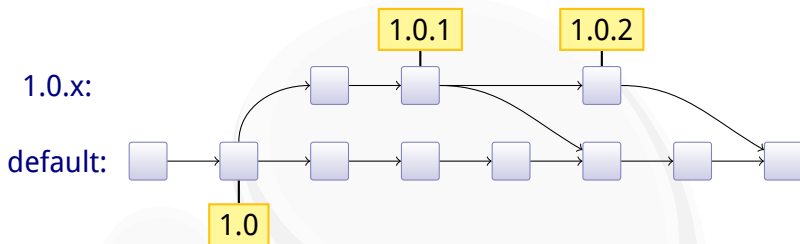
发布分支



发布分支



发布分支



已命名分支:

- ▶ 重量级分支，对后续变更集影响最大
- ▶ 针对历史记录的跟踪审计



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

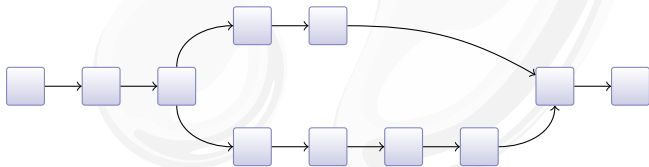
与 Git 比较

归纳总结

底层模型

概念上，Mercurial 变更集包含：

- ▶ 0-2 个父变更集标识 IDs:
 - ▶ 根 root 变更集没有父变更集
 - ▶ 一般变更集有一个父变更集
 - ▶ 合并变更集有双亲变更集
- ▶ 日期 date, 用户名 username, 提交消息 commit message
- ▶ 与父变更集之间的区别
- ▶ 变更集标识由上述信息 SHA-1 哈希计算得来 自然形成校验
- ▶ 无法向服务器注入 恶意代码



永久性历史记录

以 SHA-1 哈希为变更集标识还有这样的意义:

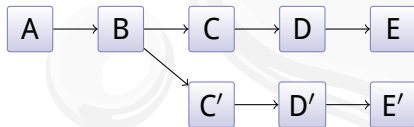
- ▶ 变更集标识即代表“迄今为止”全部的历史
- ▶ 若变更历史记录则影响全部后续变更集
- ▶ 历史记录不可变，仅能增添条目:



永久性历史记录

以 SHA-1 哈希为变更集标识还有这样的意义:

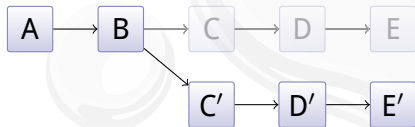
- ▶ 变更集标识即代表“迄今为止”全部的历史
- ▶ 若变更历史记录则影响全部后续变更集
- ▶ 历史记录不可变，仅能增添条目:



永久性历史记录

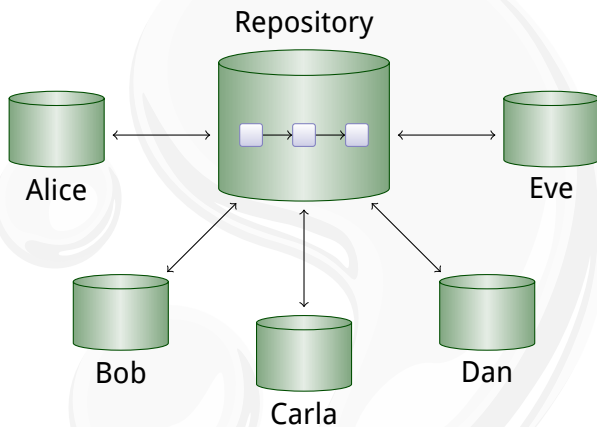
以 SHA-1 哈希为变更集标识还有这样的意义:

- ▶ 变更集标识即代表“迄今为止”全部的历史
- ▶ 若变更历史记录则影响全部后续变更集
- ▶ 历史记录不可变，仅能增添条目:



中心化 workflow

全体成员均拥有中心库的写权限:



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

浏览文件历史

The `hg annotate` 命令很有用:

- ▶ 可精确查阅每一行修改
- ▶ 可快速跳回历史版本

Mercurial 之 README 文件历史:

```
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:
# ...
```

用 `hg serve` 还有更便利的 Web 图形化界面



在文件内容中查找

琢磨某个函数是何时写的?

► `hg grep` 来帮忙!

例: `hg forget` 是何时实现的?

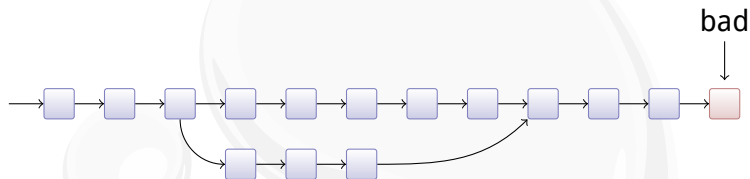
```
$ hg grep --all 'def forget' commands.py
commands.py:8902+:def forget(ui, repo, *pats, **opts):
commands.py:3522:-:def forget(ui, repo, *pats, **opts):
commands.py:814:-:def forget(ui, repo, file1, *files):
commands.py:814+:def forget(ui, repo, *pats, **opts):
# ...
```



版本图分拆

找到一个BUG! 最早何时出现?

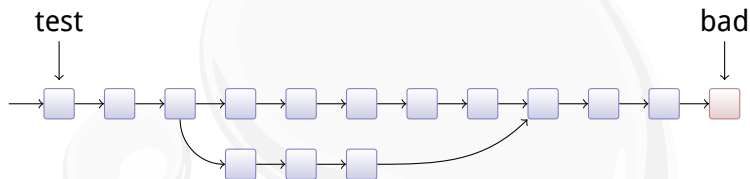
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

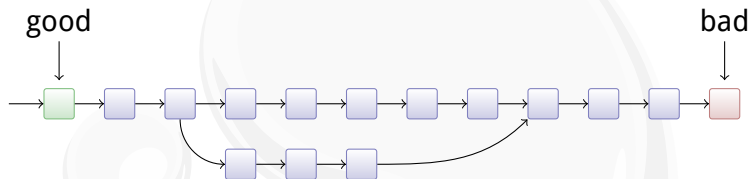
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

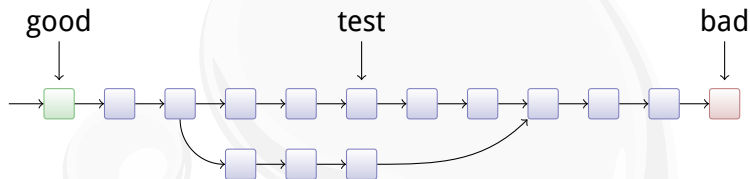
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

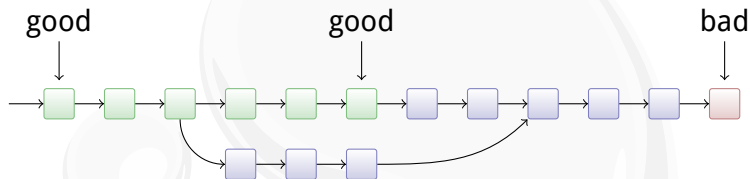
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

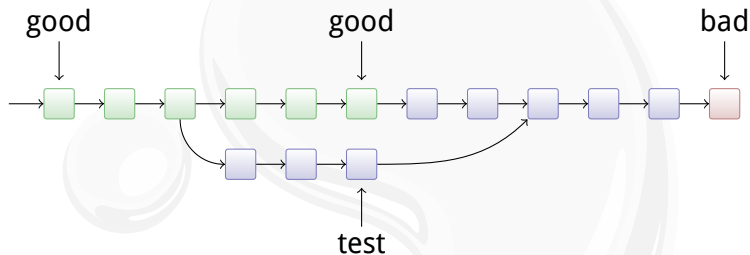
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

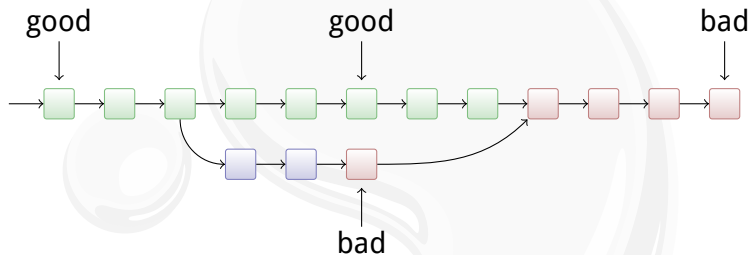
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

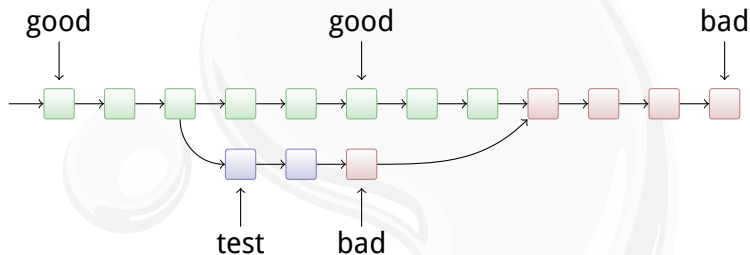
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

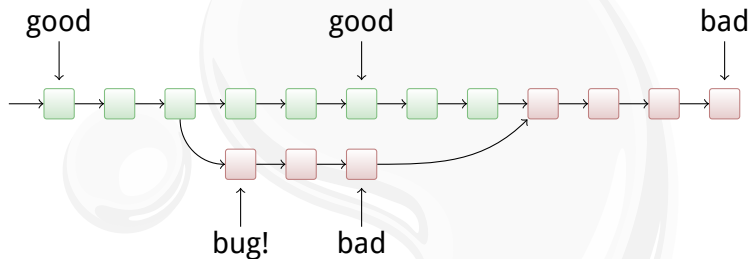
使用 `hg bisect` 来标记“好、坏”版本:



版本图分拆

找到一个BUG! 最早何时出现?

使用 `hg bisect` 来标记“好、坏”版本:



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Mercurial 容易扩展

可以给 Mercurial 添加新功能:

- ▶ 自带 30+ 扩展 extension
- ▶ wiki 上还有 75+ 扩展
- ▶ 扩展基本可操作各种内容
- ▶ 帮助使核心更为小巧专注



变更集“重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展!

► 版本图:



变更集“重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展！

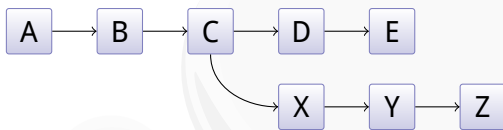
► 版本图：



变更集“重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展！

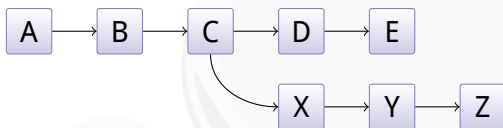
► 版本图：



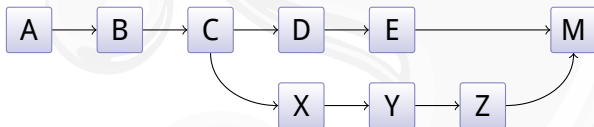
变更集“重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展！

► 版本图：



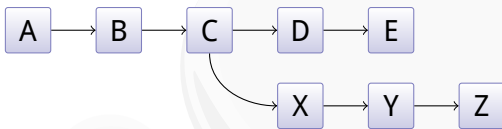
► 合并 Merge:



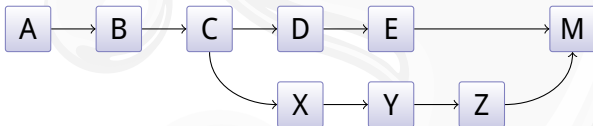
变更集 “重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展！

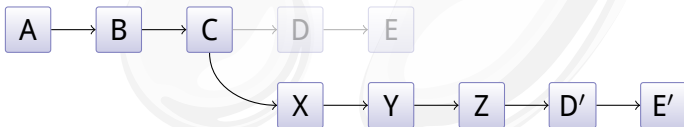
► 版本图:



► 合并 Merge:



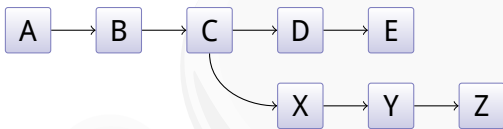
► 重起 Rebase:



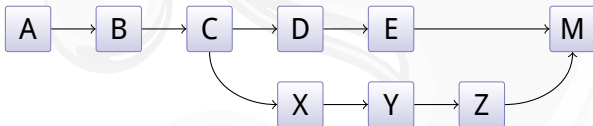
变更集 “重起”

不想再做“合并 Merge”？不妨试试 **rebase** 扩展！

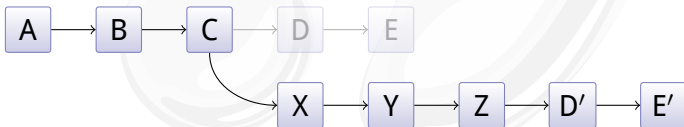
▶ 版本图：



▶ 合并 Merge:



▶ 重起 Rebase:



▶ 留神：已发布变更不能被“重起”。

维护补丁序列

mq 扩展可简化补丁序列维护:



推送内容之前可以方便在本地进行修改。



维护补丁序列

mq 扩展可简化补丁序列维护:



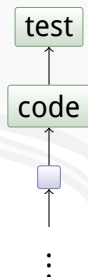
code

推送内容之前可以方便在本地进行修改。



维护补丁序列

mq 扩展可简化补丁序列维护:

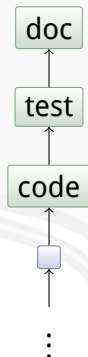


推送内容之前可以方便在本地进行修改。



维护补丁序列

mq 扩展可简化补丁序列维护:

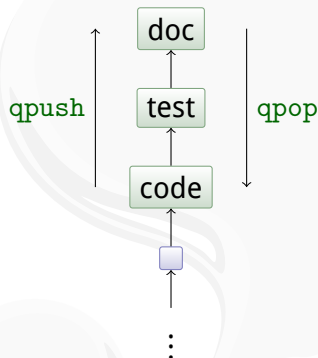


推送内容之前可以方便在本地进行修改。



维护补丁序列

mq 扩展可简化补丁序列维护:



推送内容之前可以方便在本地进行修改。



编辑历史记录

受 `git rebase -i` 启发, `histedit` 允许

- 为变更集排序:



编辑历史记录

受 `git rebase -i` 启发, `histedit` 允许

- 为变更集排序:



- 折叠变更集:



编辑历史记录

受 `git rebase -i` 启发, `histedit` 允许

- 为变更集排序:



- 折叠变更集:



- 删除变更集:



编辑历史记录

受 `git rebase -i` 启发, `histedit` 允许

- 为变更集排序:



- 折叠变更集:



- 删除变更集:



- 插入变更集:



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Git

Git 由(大神) Linus Torvalds 制作以用于 Linux 内核开发:

- ▶ 专注于速度
- ▶ 与 Mercurial 概念一致:
 - ▶ 支持副本间使用 push/pull 交互之完全分布式架构
 - ▶ 提交以变更集形式组织



Git

Git 由(大神) Linus Torvalds 制作以用于 Linux 内核开发:

- ▶ 专注于速度
- ▶ 与 Mercurial 概念一致:
 - ▶ 支持副本间使用 push/pull 交互之完全分布式架构
 - ▶ 提交以变更集形式组织

Git 相当灵活, 然而灵活有其代价:

- ▶ `git log` 有足足 150 个不同的命令行参数, 连其帮助文本也有 1496 行
- ▶ 缺省命令集不乏危险命令
- ▶ 学习曲线陡峭 --- 所谓 “暂存区域 staging area” ?



Git

Git 由(大神) Linus Torvalds 制作以用于 Linux 内核开发:

- ▶ 专注于速度
- ▶ 与 Mercurial 概念一致:
 - ▶ 支持副本间使用 push/pull 交互之完全分布式架构
 - ▶ 提交以变更集形式组织

Git 相当灵活, 然而灵活有其代价:

- ▶ `git log` 有足足 150 个不同的命令行参数, 连其帮助文本也有 1496 行
- ▶ 缺省命令集不乏危险命令
- ▶ 学习曲线陡峭 --- 所谓 “暂存区域 staging area” ?

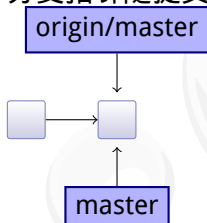
Git 仍然以 Linux 平台为主:

- ▶ 由 Linux Kernel hackers 一手打造, 服务于 Linux Kernel hackers
- ▶ 随后 Git 移植来 Windows
- ▶ 然而 Windows 平台上性能依然不足

Git分支

与 Mercurial 里的 bookmarks 相似:

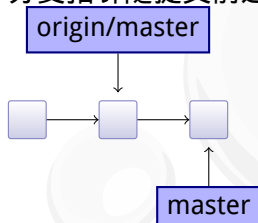
- ▶ 分支指针随提交前进:



Git分支

与 Mercurial 里的 bookmarks 相似:

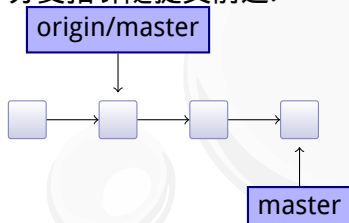
- ▶ 分支指针随提交前进:



Git分支

与 Mercurial 里的 bookmarks 相似:

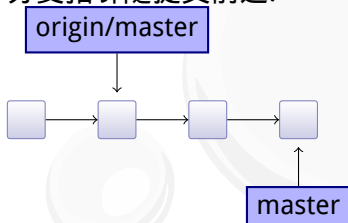
- ▶ 分支指针随提交前进:



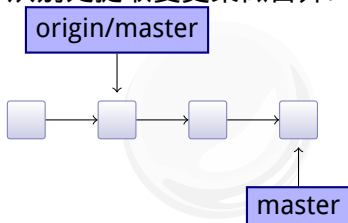
Git分支

与 Mercurial 里的 bookmarks 相似:

- ▶ 分支指针随提交前进:



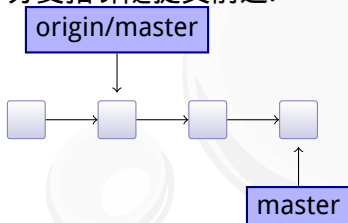
- ▶ 从别处提取变更集做合并:



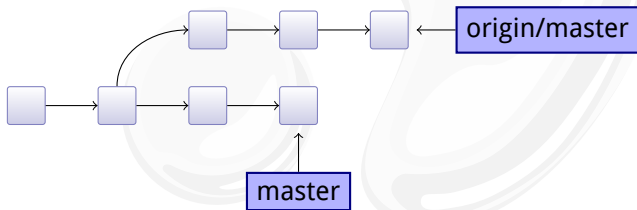
Git分支

与 Mercurial 里的 bookmarks 相似:

- ▶ 分支指针随提交前进:



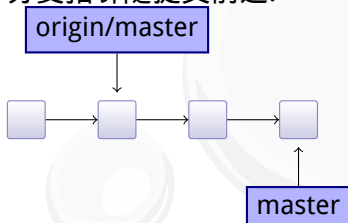
- ▶ 从别处提取变更集做合并:



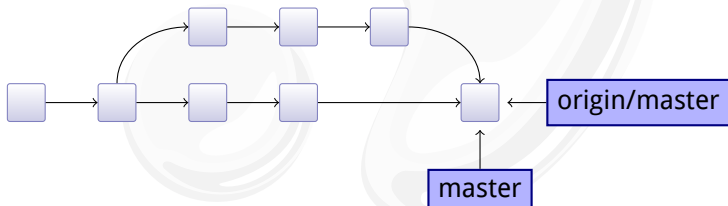
Git分支

与 Mercurial 里的 bookmarks 相似:

- ▶ 分支指针随提交前进:



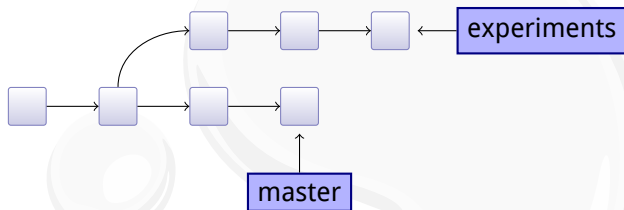
- ▶ 从别处提取变更集做合并:



删除一个 Git 分支

Git 可以 **垃圾回收 garbage collect** 变更集:

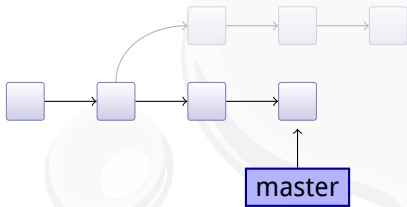
- ▶ 有实验分支的库:



删除一个 Git 分支

Git 可以 **垃圾回收 garbage collect** 变更集:

- ▶ 有实验分支的库:



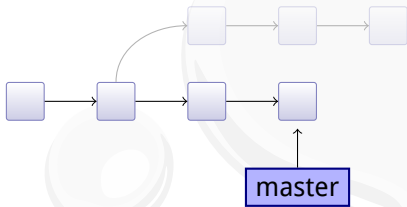
- ▶ 用 `git branch -D experiments` (标记) 删除实验性分支
- ▶ 下一次执行 `git gc` 时会一次性清除



删除一个 Git 分支

Git 可以 **垃圾回收 garbage collect** 变更集:

- ▶ 有实验分支的库:



- ▶ 用 `git branch -D experiments` (标记) 删除实验性分支
- ▶ 下一次执行 `git gc` 时会一次性清除
- ▶ 在服务器一端删除 要用 `git push origin :experiments`



Outline

介绍

Mercurial

集中 vs 分布

Mercurial 关键概念

使用 Mercurial

工作流 Workflows

分支 Branches

底层模型

使用历史

扩展

与 Git 比较

归纳总结

Mercurial 归纳

Mercurial 改变着开发形式:

- ▶ 用简单而强大的模型来 **同时** 支持分支与合并
- ▶ 提供强大的工具而非“必要的邪恶”？ (necessary evil)
- ▶ 轻量而快速



更多资料

- ▶ Mercurial 官方主页:
<http://mercurial.selenic.com/>
- ▶ Mercurial: The Definitive Guide:
<http://hgbook.red-bean.com/>
- ▶ 上手指南:
<http://mercurial.aragost.com/kick-start/>
<http://mercurial.ch/>
<http://hginit.com/>
- ▶ 一些免费 Mercurial 供应站:
<http://bitbucket.org/>
<http://code.google.com/>
<http://sourceforge.net/>
<http://www.codeplex.com/>



Mercurial 贡献者

源 <http://ohloh.net/p/mercurial/map:>



Mercurial 贡献者

源 <http://ohloh.net/p/mercurial/map:>

