



Getting Started With Bazaar

Ian Clatworthy
Canonical

Agenda



- History
- Core Concepts
- Getting Started
- Personal Version Control
- Sharing with Peers
- Team Development
- Community Development
- Further Information

History

Building on the shoulders of giants



Generation 1: SCCS, RCS



- Focus:
 - compact storage of history
 - local filesystems
 - file locking to prevent concurrent access
- Issues:
 - no support for collections of files
 - Performance over time
- Note:
 - still useful today, e.g. Twiki, W3C

Generation 2: CVS



- Focus:
 - Tree operations
 - Standard tool instead of in-house scripts (over RCS)
 - Added value:
 - Central server for storage & backup
 - Optimistic concurrency - merging
- Issues:
 - Non-atomic commits
 - Rename management
 - Integration difficulties

Generation 3: Subversion



- Focus:
 - “CVS done right”
 - atomic commits
 - rename tracking
 - easy to integrate
 - cross platform
- Issues:
 - same model
 - just well implemented now
 - non-trivial set-up
 - limited merging
 - limited renaming



Generation 3+: Early DVCS



- Focus
 - No central server
 - Working offline
 - Smart merging
- Issues:
 - ease of use (Arch)
 - performance (most)
 - 3rd party support



Generation 4: DVCS Mark II



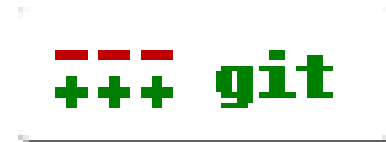
- Focus
 - Easy setup
 - Easy to use
 - High performance
- Issues:
 - maturity
 - no clear “winner”
 - 3rd party support



Bazaar

Hg

mercurial



Core Concepts

Foundations matter



Revision



- A **snapshot of a tree** of files and directories, including content and shape
- Includes metadata:
 - date created
 - author
 - commit message
 - parent revisions
 - etc.

Working Tree



- A **version controlled directory** containing files and directories the user can edit
- Many commands use the working tree as their context
 - e.g. `commit` makes a new revision using the current content of files in the working tree

Branch



- An **ordered series of revisions**
- The latest revision in a branch is known as the *“head”*
- Branches may be split apart and *“merged”* back together
- Branches are related if they have a *“common ancestor”*

Repository



- A **store of revisions**
- By default, each branch has its own repository
 - to save disk space and improve performance, branches can share a repository
 - more on this later ...

Getting Started

Installation, Configuration & UI



Installing Bazaar



- Linux packages available for (e.g.):
 - Ubuntu/Debian (apt)
 - Redhat/Fedora (yum)
- Other operating systems:
 - Windows – GUI installer
 - Windows Cygwin - standard package
 - Mac OS X – MacPorts or Fink
 - Free BSD - FreshPorts

Running from Source



- Requires Python 2.4 or later
- Very easy to do:
 - unpack using tar, FileRoller, WinZip, etc.
 - put directory on your PATH
- To verify your install:
 - `bzr version`
- Advanced tip (only if you're worried):
 - `bzr selftest`

Configuring Bazaar



- Tell Bazaar about yourself ...
 - `bzr whoami "Bill Bloggs <bill@bloggs.com>"`
- Verifying your identity ...
 - `bzr whoami`
`Bill Bloggs <bill@bloggs.com>`
- or
 - Linux: `vi ~/.bazaar/bazaar.conf`
 - Windows: `bzr-config` tool or notepad ...

Entering Commands



- General syntax:
 - `bzr [global-opts] command [opts & args]`
- Things to note:
 - Options can appear before or after arguments
 - Global options can appear anywhere
- Option types:
 - boolean, enumerated (parameter less)
 - string, list (parameter required)

Common options



- -h, --help
- -v, --verbose
- -q, --quiet
- -r, --revision
- show help
- show more info
- show less info
- select revision range: x..y
 - depending on context, either:
 - from x to y inclusive
 - y vs x (e.g. for diff)
 - both optional

Command Aliases



- Many commands have aliases:
 - reduced typing
 - easier for cvs and svn users to learn and use
- Can create your own as well, e.g.

```
[ALIASES]
```

```
slog=log --short
```

Plugins



- Plugins add new commands, new options, new formats or completely new features

The screenshot shows a web browser window displaying the Bazaar website. The browser's address bar shows the URL <http://bazaar-vcs.org/BzrPlugins>. The website header features the Bazaar logo (a yellow diamond with a black arrow pointing up) and the text "Bazaar GPL Distributed Version Control Software". Navigation links for "Docs", "Download", "Community", and "Plugins" are visible. The "Plugins" link is highlighted with a green arrow. Below the header, there is a search bar labeled "Search Bazaar" and a "go" button. A section titled "Wiki Tools" lists several tools: IanClatworthy, User Preferences, FindPage, RecentChanges, Edit Page, Page History, and Subscribe. At the bottom of the page, there is a status bar with the following information: Done, DWL: 21.48%, New Zealand: Thu 14:48, Sydney: Thu 12:48, UK: Thu 03:48, Chicago: Wed 21:48.

File Edit View History Bookmarks Tools Help

← → ↻ × + TAG http://bazaar-vcs.org/BzrPlugins Google

 **Bazaar**
GPL Distributed Version Control Software

Docs Download Community Plugins

Plugins for Bazaar

This page lists plugins for Bazaar that are available under a free software licence. Please do not add non-free plug-ins to this page unless they are read-only, in the sense that they are designed primarily to provide a migration pathway from a non-free revision control system to Bazaar.

RSS feed generators for bazaar

There are several methods to produce rss feeds of your bazaar repositories. See [FeedGenerators](#) for more information.

Search Bazaar go

 **Wiki Tools**

- * IanClatworthy
- * User Preferences
- * FindPage
- * RecentChanges
- * Edit Page
- * Page History
- * Subscribe

Done DWL: 21.48% New Zealand: Thu 14:48 Sydney: Thu 12:48 UK: Thu 03:48 Chicago: Wed 21:48

Installing Plugins



- OS packages available for the more popular ones, e.g. bzrtools, bZR-gtk
- Otherwise, the default location is `$BAZAAR_HOME/plugins`
 - Be sure to check the README for the correct top level directory name
- To see the installed plugins:
 - `bZR plugins`

Graphical Interfaces

The screenshot displays an Ubuntu desktop environment with several windows open:

- Terminal:** Shows the following commands and output:

```
syndicate scott% cd co/canonical/bzr.dev
syndicate bzr.dev% bzr viz
syndicate bzr.dev% bzr viz
```
- Commit History Table:** A table with columns for Message, Committer, and Date. It lists two commits:

Message	Committer	Date
merge from upstream	Robert Collins <robertc@robertcollins.ne...>	Tue ...
lock during fetch, which is a separate cod...	Robert Collins <robertc@robertcollins.ne...>	Tue ...
- Graphical Commit Graph:** A visual representation of the commit history, showing a series of nodes connected by lines, with a highlighted path.
- Diff Viewer:** A window titled "bzr.dev - bzrk diff" showing a comparison of files. The "Modified" section lists:
 - bzrlib/changeset.py
 - bzrlib/merge.py** (highlighted)
 - bzrlib/merge_core.py
 - bzrlib/selftest/test_mergeThe diff content for `bzrlib/merge.py` is shown:

```
=== modified file 'bzrlib/merge.py'
--- bzrlib/merge.py
+++ bzrlib/merge.py
@@ -412,16 +412,6 @@
         source_file.interesting = source_file.id in interesting_ids

-def generate_cset_optimized(tree_a, tree_b, interesting_ids=None):
-    """Generate a changeset.  If interesting_ids is supplied, only changes
-    to those files will be shown.  Metadata changes are stripped.
-    """
-    cset = generate_changeset(tree_a, tree_b, interesting_ids)
-    for entry in cset.entries.itervalues():
-        entry.metadata_change = None
-    return cset

def merge_inner(this_branch, other_tree, base_tree, tempdir,
                ignore_zero=False, merge_type=ApplyMerge3, backup_files=False,
                interesting_ids=None):
@@ -438,7 +428,7 @@
     return tree.tree.inventory

inv_changes = merge_flex(this_tree, base_tree, other_tree,
                          generate_cset_optimized, get_inventory,
+                          generate_changeset, get_inventory,
                          MergeConflictHandler(this_tree, base_tree,
                                                other_tree, ignore_zero=ignore_zero),
                          merge_factory=merge_factory,
```
- Commit Details Panel:** Located at the bottom left, it provides information for the selected commit:
 - Revision:** robertc@robertcollins.net-
 - Committer:** Robert Collins <robertc@r...
 - Timestamp:** Mon 2005-10-10 13:42:29
 - Parents:** robertc@robertcollins.net-
 - Message:** merge Gustavos executable2 patch

Getting Help



- Bazaar comes with online help built-in
- To see the list of topics:
 - `bzr help`
- To get help on command xxx:
 - `bzr help xxx`



Summary: Getting Started



- Key commands:

whoami

help

- Other commands:

version

plugins



Exercise 1: Getting Help

Time allowed: 10 minutes



Personal Version Control

Who said VCS tools are only for teams?



Solo Workflow



- ① create project
- ② record changes
- ③ browse history
- ④ package release

Computer
Copyright by Andrew Hopson



Starting a Project



- To put an existing tree under VC:
 - `cd my-stuff`
 - `bzr init`
 - `bzr add`
 - `bzr commit -m "Initial import"`
- A copy of all files is now archived under the `.bzr` directory
 - top level only (unlike cvs or svn)

Viewing Changes



- Change summary:
 - `bzr status`
- Actual differences:
 - `bzr diff`

Registering Files and Directories



- Registering all files and directories:
 - `bzr add`
- Registering some files and directories:
 - `bzr add foo.py bar/`

Recording Changes



- Recording all changes:
 - `bzr commit`
- Recording only the changes to selected files and directories:
 - `bzr commit README doc/`
- The `-m` option can be used to provide a message on the command line
 - otherwise an editor is launched for this

Undoing Things



- Deregistering a file or directory:
 - `bzr remove foo.py`
- Undoing the last commit:
 - `bzr uncommit`
- Undoing changes since the last commit:
 - `bzr revert`
 - `bzr revert foo.py`

Viewing History



- What is the history of this branch?
 - `bzr log`
- Condensed view:
 - `bzr log --short`
- GUI views available via plugins ...
- `bzr-gtk`
 - `bzr visualize`
- `Qbzr`
 - `bzr qllog`

QBzr - Log

Rev	Date	Author	Message
336	5/7/2007 9:47:33 AM	Lukáš Lalinský <lalinsky@gmail.com>	Merge SVN.
335	5/7/2007 8:18:13 AM	Lukáš Lalinský <lalinsky@gmail.com>	MP4, ASF: Check if the file is valid before readin...
334	5/7/2007 8:06:12 AM	Lukas Lalinsky <lalinsky@gmail.com>	File::readBlock: Don't try to read empty blocks.
333	5/7/2007 8:04:28 AM	Lukas Lalinsky <lalinsky@gmail.com>	Fix file opening on Windows.
332	5/7/2007 7:53:51 AM	Lukas Lalinsky <lalinsky@gmail.com>	Fix compilation under MSVC 2003.
331	3/16/2007 11:37:2...	Lukáš Lalinský <lalinsky@gmail.com>	Merge from SVN
279.1.9	2/16/2007 8:15:55 ...	wheeler	SVN_SILENT whoops, those were tabs
279.1.8	2/16/2007 6:29:40 ...	wheeler	Delete d-pointer.
330	2/13/2007 10:19:1...	Lukáš Lalinský <lalinsky@gmail.com>	Merge from SVN
279.1.7	2/13/2007 2:19:20 ...	wheeler	SVN_SILENT whoops
279.1.6	2/13/2007 2:18:30 ...	wheeler	Read the frame data length as a synch-safe int...
279.1.5	2/13/2007 1:55:56 ...	wheeler	Don't try to parse invalid frames.
279.1.4	2/13/2007 1:26:31 ...	wheeler	Add another sanity check -- don't let invalid fra...
279.1.3	2/13/2007 12:40:5...	wheeler	Fix the byte ordering for UTF 16BE
279.1.2	2/12/2007 11:21:4...	wheeler	Rework the read-only check so that it gets alon...
279.1.1	2/12/2007 9:43:10 ...	wheeler	Actually overwrite the value, as documented.
329	2/11/2007 5:43:41 ...	Lukáš Lalinský <lalinsky@gmail.com>	Change documentation title to TagLib.Ext
328	2/11/2007 5:39:05 ...	Lukáš Lalinský <lalinsky@gmail.com>	ASF: More documentation.
327	2/11/2007 5:27:01 ...	Lukáš Lalinský <lalinsky@gmail.com>	ASF, MP4: Documentation, API fixes, nitpicking,...
326	2/11/2007 5:05:34 ...	Lukáš Lalinský <lalinsky@gmail.com>	Fix the API documentation CSS for new versions...
325	2/11/2007 3:49:51 ...	Lukáš Lalinský <lalinsky@gmail.com>	Oh, well, changing the license of ASF and MP4 c...
324	2/11/2007 3:44:33 ...	Lukáš Lalinský <lalinsky@gmail.com>	Add doxygen files to generate the documentati...

Revision: [lalinsky@gmail.com-20070213181913-izh4fwgcu0b0bgy9](#)

Parent revisions: [lalinsky@gmail.com-20070211134341-itk2n8letnx337n](#), [ycs-imports@canonical.com-20070213101920-98810479d3e860ad](#)

Author: Lukáš Lalinský <lalinsky@gmail.com>

Branch nick: extra

Merge from SVN

```
taglib/mpeg/id3v2/frames/textidentificationframe.cpp
taglib/mpeg/id3v2/id3v2framefactory.cpp
taglib/toolkit/tfile.cpp
taglib/toolkit/tfile.h
taglib/toolkit/tmap.tcc
taglib/toolkit/tstring.cpp
```

Close

Viewing History of a File



- List changes that touched a file:
 - `bzr log foo.py`
- Content of file at revision X:
 - `bzr cat -r X foo.py`

Packaging a Release



- Creating a tar.gz or zip file:
 - `bzr export ../releases/my-stuff-0.1.tar.gz`
 - `bzr export ../releases/my-stuff-0.1.zip`
- Tagging a revision:
 - `bzr tag RELEASE-0.1`
- Viewing existing tags:
 - `bzr tags`

Some Points to Note



- To track a moved file or directory:
 - `bzr mv source destination`
- Symlinks can be versioned
- Permissions:
 - Execute bit is tracked
 - Others are not
- Binary files are supported



Summary: Personal Version Control



- Key commands:

init

add

commit

status

diff

log

export

- Other commands:

remove

uncommit

revert

cat

tag

mv



Exercise 2: Browsing Changes

Time allowed: 15 minutes



Sharing with Peers

Getting by with a little help from your friends



Partner Workflow



① start project



Computer



③ record changes



② bzd branch



Computer



③ record changes

④ merge changes from peer



④ merge changes from peer

Creating a New Branch



- To clone an existing branch:
 - `bzr branch URL [directory]`
- By default, the directory name is taken from the last part of the URL
- Some examples:
 - `bzr branch /home/mary/cool-dev`
 - `bzr branch /home/mary/cool-dev cool`
 - `bzr branch http://mary-laptop/cool-dev`
 - `bzr branch sftp://bill@mary-laptop/cool-dev`

Merging Changes



- To grab changes from another branch, use the merge command:
 - `bzr merge [URL]`
- If the URL is omitted, the parent branch is assumed

Theory: 3 Way Merging



- To intelligently merge changes, the content of the “best common ancestor” is considered
- Given ancestor A and branches B and C :
 - $A == B == C$ *then* line unchanged
 - $A == B != C$ *then* take line from C
 - $A == C != B$ *then* takes line from B
 - $A != B != C$ *then* conflict

Conflicts



- Some merges can only be completed with the assistance of a human
- The merge command reports these files as having conflicts and creates:
 - a file with embedded markers showing the areas it couldn't resolve
 - three files to assist you:
 - foo.THIS
 - foo.OTHER
 - foo.BASE

Resolving Conflicts



Sample markers:

```
<<< TREE
import os, sys
===
import os, re
>>> MERGE-SOURCE
```

- After editing, to clear all conflicts:
 - `bzr resolve`
- To clear a given conflict:
 - `bzr resolve foo.py`

Recording a Merge



- After conflicts, if any, are resolved, the merge can be committed:
 - `bzr commit -m "merged Mary's changes"`
- Even if there are no conflicts, an **explicit** commit is still required!
- This is a **good** feature, not a bug:
 - might still want/need to tweak changes
 - regression tests can and do fail at times after a “clean” merge

Merge Tracking



- Bazaar **remembers** what has been merged already and doesn't try to merge those changes again
- You can safely merge from a branch as many times as you need or like
- It's very hard to explain just how much of a difference this makes vs development using less intelligent tools

Annotating a File



- When developing with 2 or more people, it can be very useful to see the origin of each line in a file
 - `bzr annotate foo.py`
- Aliases are `blame` and `praise`
- GUI included as part of the `bzr-gtk` plugin
 - `bzr gannotate foo.py`



Summary: Sharing With Peers



- Key commands:

branch

merge

resolve

(commit)

- Other commands:

(status)

(diff)

annotate



Exercise 3: Merging

Time allowed: 15 minutes



Team Development

There's more than one way to do it!



Pushing Branches



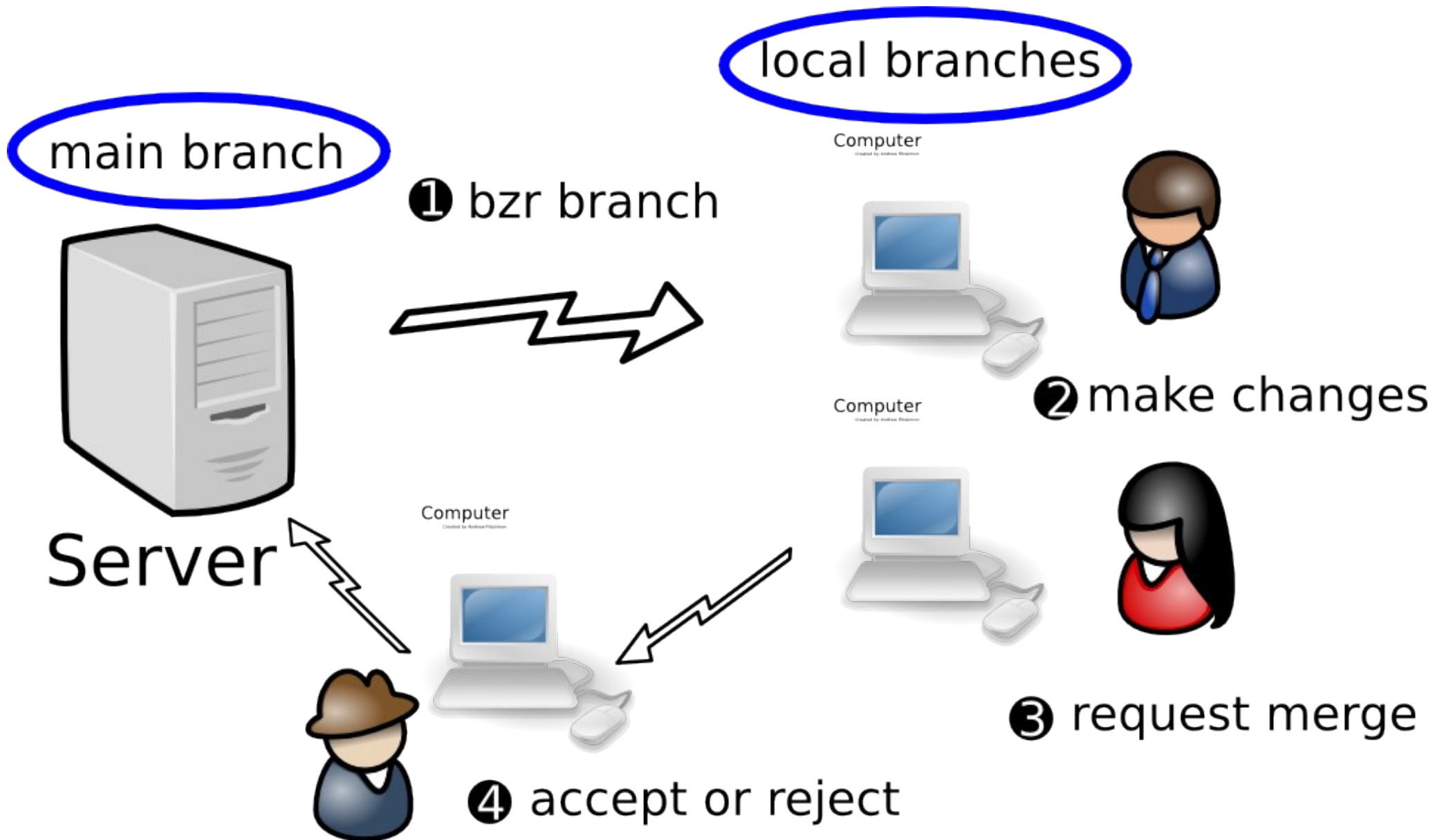
- To easiest way to share a branch with a team is to “push” it to a shared location
- Commonly used technologies include:
 - shared drives (*Windows/Samba, NFS*)
 - web servers (*write via sftp or ftp if remote*)
- Examples:
 - `bzr push x:\projects\cool`
 - `bzr push sftp://server/public_html/cool`

Best Practice: Feature Branches



- Each new feature or fix is developed in its own branch
- Advantages vs one workspace:
 - reduced coupling between changes
 - can work on multiple things in parallel
 - some changes need longer to cook, e.g. review comments can be applied and then changes resubmitted
- **Quality and stability of trunk** is higher

Distributed Workflow: Central GateKeeper



Mirroring Branches



- When developing using feature branches, it's a good idea to keep a local mirror of the trunk
- Setup:
 - `bzr branch URL my-mirror`
 - `bzr branch my-mirror my-change-X`
- Then later:
 - `cd my-mirror`
 - `bzr pull`

Getting Changes to the Gatekeeper



- Making changes:
 - *hack, hack, hack*
 - *bzr commit*
 - *notify gatekeeper*
- Might need to:
 - *cd my-change-X*
 - *bzr merge*
 - *bzr commit*
- Gatekeeper merges the changes when ready:
 - *bzr merge URL*
 - *bzr diff*
 - *test, test, test*
 - *commit or reject*
- Result: Quality++

Sending Deltas



- Sometimes, the gatekeeper can't pull from your branch, e.g.:
 - it's in a private location
 - you're likely to be offline
- So send them a “*merge directive*” - an **intelligent** patch including both content & metadata
 - `bzr send ...`
- This finds the delta from your branch to its origin and fires up your email tool

Managing Branches



- When developing using feature branches, a large number of branches can quickly accumulate
- To find out about a branch:
 - `bzr info`
- To give a branch a nick-name:
 - `bzr nick solves-world-peace`
- To check on the status of a mirror branch:
 - `bzr missing`

Best Practice: Shared Repositories



- As each branch has the full history of changes, lots of branches can mean lots of disk space
- A “shared repository” solves this nicely
 - `bzr init-repo cool-branches`
 - `cd cool-branches`
 - `bzr branch trunk-URL cool-dev`
 - `bzr branch cool-dev cool-fix`
 - *work, work, work*



Summary: Team Development 1



- Key commands:

push

pull

(merge, etc.)

send

- Other commands:

info

nick

missing

init-repo



Exercise 4: Push and Pull

Time allowed: 15 minutes



Centralised Workflow



Server

① bzip checkout



Computer



② bzip update



Computer



③ bzip commit



Checkouts



- A working tree which is **bound to a remote branch**
- Commits in a checkout:
 - verify that the working tree is up to date
 - store the revision in the remote branch (first)
- To create a checkout:
 - **bzr checkout URL**
- To create a checkout with no local branch:
 - **bzr checkout --lightweight URL**

Keeping up to Date



- To merge the latest changes from the master branch:
 - `bzr update`
- Just like a normal merge, some manual resolution of conflicts might be required
- Once your checkout is up to date (and the diff looks good), you can commit:
 - `bzr commit`

Disconnected Commits



- When disconnected from the master branch (e.g. on the road), commits can still be made locally like this:
 - `bzr commit --local`
- To remove or change the association with the master branch:
 - `bzr unbind`
 - *work, work, work*
 - `bzr bind URL`

Centralised Workflow: Disconnected at Times



Server

① bzd checkout



③ bzd commit

Computer



② bzd commit --local



Computer



② bzd unbind
bzd commit
bzd bind





Summary: Team Development 2



- Key commands:

checkout

update

(commit)

- Other commands:

bind

unbind



Exercise 5: Checkouts

Time allowed: 10 minutes



Empowering an Open Source Community

Power to the people



Teams vs Communities



- In both cases, adaptive planning and agile/lean practices are essential, **but** ...
- Best practices for developing software using an in-house team differ from those suitable for open source communities
- For most teams, the challenge is:
How fast/well can we plant?
- For most communities, the challenge is:
How fast/well can we harvest?

Changing the Community Rules ...



- Central VCS:
 - core people can commit to main trunk
 - these people get the power of a VCS
 - non-core people submit patches
- Distributed VCS:
 - core people can commit to main trunk
 - everyone gets the power of a VCS
 - non-core people submit branches (or merge directives)

Non-Core Developers Win



- DVCS removes the **technical** barrier between who can commit changes and who can't
- The issue becomes one of *where* they can commit to!
 - just their branch?
 - a team branch?
 - the main trunk?
- A barrier still exists but it's now a **social** one

Climbing the Social Ladder



- As contributors prove themselves capable, they gain:
 - community trust
 - rights & responsibilities accordingly
- In some projects, DVCS makes it easier to:
 - identify core candidates
 - transition them into the core

The Core Wins as Well



- Less administration:
 - sharing and merging is now possible without needing *yet another* central branch created
 - every branch is a full backup!
- More flexibility:
 - can share and merge changes with anyone, not just other core members
 - if using Bazaar, can change the workflow as the community evolves without needing to change tools

Survival of the Fittest



- A code base gets forked when:
 - similar but different needs arise
 - the trust model breaks down
- These are social issues!



Distributed VCS Tools Can Ease the Pain



- smart-merging is the bread-and-butter of DVCS tools
- so sharing and merging of patches across forked projects is easier
- so getting back together is easier

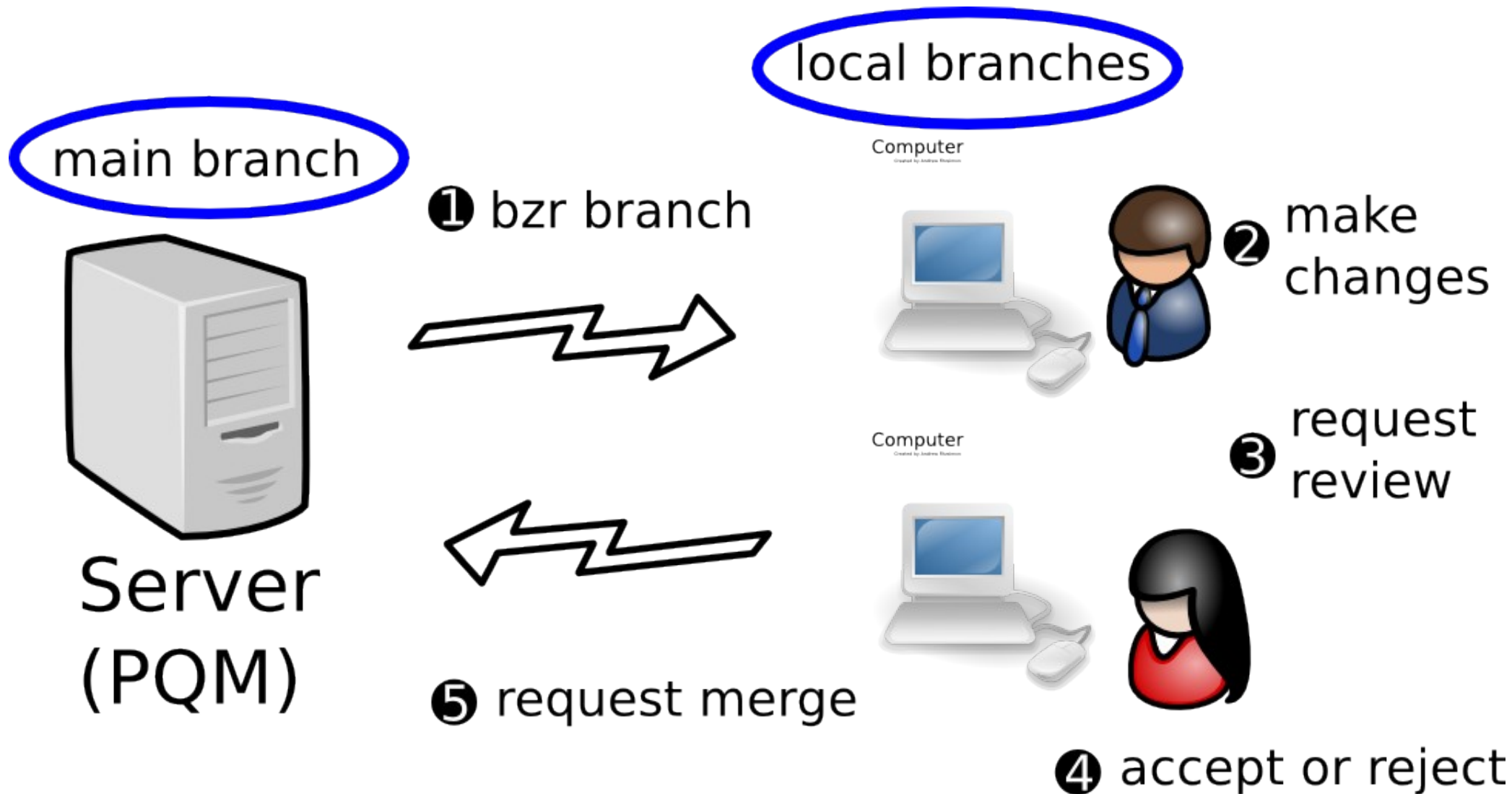


Community Workflow: The Hierarchy of Trust

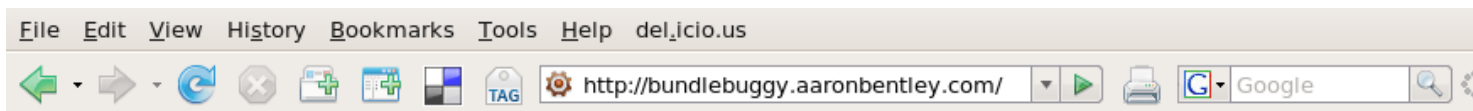


- The central gatekeeper model can be scaled to any number of levels
 - Linus trusts N people and merges changes from them
 - Each of them trusts N more and merges changes from them
 - and so on ...
- The **mechanical** aspect of the central gatekeeper can be automated

Distributed Workflow: Automated GateKeeper



Best Practice: Bundle Tracking


 Search

[Logout](#) | [Help](#)

[My Pending](#) |
 [Mine](#) |
 [Pending](#) |
 [Resubmit](#) |
 [Merged](#) |
 [Rejected](#)

Status	Votes	Tracking	Date	Submitter	Bug	Summary	Discussion
Semi-approved	tweak	Auto	Sep 26 2007	John Ferlito	—	[MERGE] bzd+https is not a supported protocol	Gmane
Semi-approved	approve	Auto	Sep 28 2007	Robert Collins	—	[MERGE] Require the root entry of a tree be supplied to CommitBuilder	Gmane
Semi-approved	approve	Auto	Sep 27 2007	Alexander Belchenko	—	[MERGE] check bzrlib version earlier (was: Re: Problems installing bzd from source on windows)	Gmane
Semi-approved	approve	Auto	Sep 24 2007	Vincent Ladeuil	140432	[MERGE][Bug 140432] Connection sharing for bound branches	Gmane
Semi-approved	approve	Auto	Sep 24 2007	Andrew Bennetts	—	Re: [MERGE] Implement KnitRepository. find inconsistent revision parents	Gmane
Semi-approved	approve, comment	Auto	Sep 21 2007	Alexander Belchenko	—	[MERGE] skip windows-unfriendly tests	Gmane
Semi-approved	approve	Auto	Sep 21 2007	Alexander Belchenko	—	[MERGE] test and fix for commit with bad non-ascii messages (non-ascii chars that cannot be decoded with current user encoding)	Gmane
Semi-approved	approve	Auto	Sep 21 2007	Martin Pool	—	Re: [merge] refactor pack repositories	Gmane
Semi-approved	approve, comment	Auto	Sep 04 2007	Andrew Bennetts	—	[MERGE] Implement chunked body encoding for the smart protocol.	Gmane
Semi-approved	approve, comment	Auto	Aug 28 2007	Aaron Bentley	—	Re: [MERGE] Reconcile can fix bad parent references	Gmane

1 - 10



Best Practice: Central Branch Tracking



File Edit View History Bookmarks Tools Help del.icio.us

https://code.launchpad.net/upstart/

Your location: Home upstart Logged in as Ian Clatworthy - Log Out

Help Upstart Search

Overview Code Bugs Blueprints Translations Answers

Actions
> Register branch

Bazaar branches

Show branches with status of

Name	Author	Status	Last Commit
main	Scott James Remnant	Development	six days ago
replacement-initscripts	Scott James Remnant	Development	32 weeks ago
watch-delayed	Johan Kiviniemi	Development	32 weeks ago
complex-event-config	Scott James Remnant	Experimental	32 weeks ago
profiles	Alex Smith	Experimental	15 weeks ago

Terms of Use | Help improve Launchpad | FAQ

Copyright 2004-2007 Canonical Ltd. | build 4920

D... code.launchpad.net DWL: 21.48% New Zealand: Fri 03:00 Sydney: Fri 01:00 UK: Thu 16:00 Chicago: Thu 10:00

Managing Branches Using Launchpad



- To get a copy of a project from LP:
 - `bzr branch lp:proj-name`
- Otherwise, browse the available branches and enter the URL registered
- To upload a branch, use either sftp or bzr +ssh, e.g.:
 - `bzr push sftp://...`
- Branches in other locations can be registered as well



Exercise 6: Launchpad

Time allowed: 10 minutes



Further Information

Learning more



But wait, there's more!



- Some things **not** covered in this course:
 - IDE and file manager integration
 - Smart servers
 - Hooks
 - Testaments (cryptographic signing)
 - Foreign branches (e.g. bzs-svn)
 - Nested trees
 - Scripting Bazaar

Bundled Documentation



The screenshot shows a Mozilla Firefox browser window titled "Bazaar Main Document Catalog - Mozilla Firefox". The address bar contains the URL "http://doc.bazaar-vcs.org/latest/". The page content includes a main heading "Bazaar Main Document Catalog", a paragraph stating that the latest documents are available from "http://doc.bazaar-vcs.org", and two sections: "Core Documentation" with links to Mini Tutorial, Quick Start Summary, User Guide, User Reference, Release Notes, and Developer Guide; and "Other Documentation" with links for switching and migration guides. A status bar at the bottom shows system information like "Done", "DWL: 21.48%", and various time zones.

Bazaar Main Document Catalog - Mozilla Firefox

File Edit View History Bookmarks Tools Help del.icio.us

http://doc.bazaar-vcs.org/latest/

Bazaar Main Document Catalog

The latest version of these documents are available from Bazaar's documentation site, <http://doc.bazaar-vcs.org>.

Core Documentation

- [Mini Tutorial](#)
- [Quick Start Summary](#)
- [User Guide](#)
- [User Reference](#)
- [Release Notes](#)
- [Developer Guide](#)

Other Documentation

Moving to Bazaar (Web links):

- [Switching Guides](#) - for users moving from another VCS tool
- [Migration Guide](#) - for teams migrating history from another VCS tool

Other Documents (Web links):

- [Glossary](#)
- [Frequently Asked Questions](#)
- [bzrlib API reference](#)

Done | DWL: 21.48% | New Zealand: Fri 18:41 | Sydney: Fri 16:41 | UK: Fri 07:41 | Chicago: Fri 01:41

Web Sites



- Main Web site:
 - <http://bazaar-vcs.org>
- Launchpad – Bug tracking, etc:
 - <https://launchpad.net/bazaar/> (*Bazaar related*)
 - <https://launchpad.net/bzr/> (*Bazaar core*)
- Official documentation:
 - <http://doc.bazaar-vcs.org>

The Bazaar Community



Internet Relay Chat:

- #bzd on irc.freenode.net

Mailing lists:

- See the main web site

New contributors always welcome!

Thank-you!

ian.clatworthy@internode.on.net

ianc@canonical.com

ianc@ubuntu.com

irc nick: igc

