

■ ■ ■ Push up your code – next generation version control with (E)Git



Dominik Schadow
Senior Consultant
Application Development

dominik.schadow@trivadis.com

Java Forum Stuttgart, 07.07.2011

trivadis
makes **IT** easier. ■ ■ ■

Agenda



- Almost all about Git and EGit
- Push and pull, a typical day with Git
- The ultimate question of version control

A screenshot of a Git log showing commit history. The log includes commit hashes, dates, times, and the names of the committers. The committer is consistently 'Dominik Schadow'. The log shows a sequence of commits, including one that merges a branch named 'bugfix_1000' back into the 'master' branch. The commit messages are partially visible, showing actions like 'Added missing field to Data bean' and 'Filled missing information'. The log is displayed in a terminal-like window with a light background and dark text.

```
commit e47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
Merge branch 'bugfix_1000'
Added missing field to Data bean

commit d1ec4946988fb8e47464d20a8b9a6d155b73087a
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:23:14
Filled missing information

commit 6957a53029201d4b8c91319b9c0457f62a304904
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:22:03
Added initial project files

commit 6957a53029201d4b8c91319b9c0457f62a304904
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:20:46
Added initial project
```

Agenda



- Almost all about Git and EGit
- Push and pull, a typical day with Git
- The ultimate question of version control

A screenshot of a Git commit log and file diff. The commit log shows a series of commits by Dominik Schadow, including 'Merge branch 'bugfix_1000'', 'Added missing field to Data bean', 'Filled missing information', 'Added initial project files', and 'Added initial project'. The file diff shows changes to a file named 'Data bean', with lines added and removed. The commit hash is c47b558465376301ebb689b0f9f1e35dc1f42ef5.

```
commit c47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
Merge branch 'bugfix_1000'

Added missing field to Data bean
Filled missing information
Added initial project files
Added initial project

c47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
Merge branch 'bugfix_1000'

Added missing field to Data bean
Filled missing information
Added initial project files
Added initial project

c47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
Merge branch 'bugfix_1000'

Added missing field to Data bean
Filled missing information
Added initial project files
Added initial project
```

Subversion and CVS have many disadvantages



- ⊕ Creating a branch is easy and fast
 - ⊖ Merging is a pain (most of the time)
 - ⊖ All branches are shared, no local (private) branches

- ⊕ Central repository server makes backups easy
 - ⊖ Everybody is working against the same repository
 - ⊖ Clients require server connection for most operations

- ⊕ Performance is good for certain operations
 - ⊖ Slow merge, diff or switch operations
 - ⊖ Slows down as the project (history) grows larger

Git was created in the Linux community



2005	Git development starts in the Linux (kernel) community by Linus Torvalds
2006	JGit development starts, a 100% pure Java reimplementation of the Git version control system
2009	EGit/ JGit move to Eclipse, first projects migrate to Git
2010 (Sep)	EGit/ JGit 0.9.1
2011 (Feb)	EGit/ JGit 0.11.1
2011 (Jun)	EGit/ JGit 1.0 with Eclipse 3.7
2012 (Jun)	EGit/ JGit 2.2 with Eclipse 3.8

JGit and EGit are official Eclipse projects



- The original **Git**
 - Original version developed by the Linux community
 - Distributed under the GNU General Public License (GPL)
- **JGit** is a lightweight Java library implementing Git
 - JGit library can be found in many Java based products
 - Plug-ins for Eclipse and NetBeans IDE, Hudson CI server, Apache Maven, and Gerrit Code Review
 - Distributed under the Eclipse Distribution License (EDL)
- **EGit** is the Eclipse team provider and uses JGit
 - No team provider trouble as with Subversion
 - Normally no command line required
 - Distributed under the Eclipse Public License (EPL)

Git is a **D**istributed **V**ersion **C**ontrol **S**ystem (DVCS)



- Git clients fully mirror the repository
 - ▣ Every clone is a complete backup
 - Git always clones the entire repository
 - No partial checkout possible
 - ▣ The whole repository is available locally
 - Entire development history
 - Complete repository with all branches, not only the latest snapshot
- No network connection required
 - ▣ Most operations, except *push/pull* and *fetch*, work offline
 - Much better performance
 - ▣ No central server is required
 - Local repository for private development
 - Clients can directly communicate with each other

Branching and merging is easy and fast



„In Git it's common to create, work on, merge, and delete branches several times a day.“

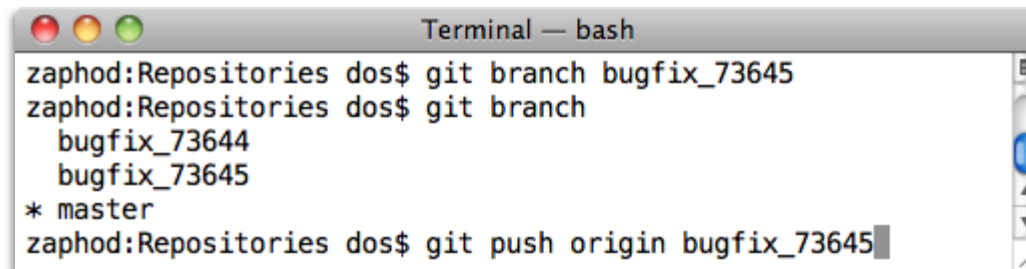
<http://progit.org/book>

- Branching and merging are an essential Git concept
 - ▣ Create local branch for each feature/ bug fix you work on
 - ▣ You can have many feature branches at any time
 - Easy to switch between them
 - No mix up of changes in the same branch
 - ▣ History-aware merging capability
 - Auditing of branch and merge events

The default 'trunk' is called 'master' in Git



- All branches are local after creation
 - Extremely fast, no network communication required
 - Every developer's working copy is a private branch
- Easy to share a branch (or tags) with other developers
 - But most branches live only for a short time locally
 - Push to share
 - **git push (remote) (branch)**

A screenshot of a macOS Terminal window titled "Terminal — bash". The window shows a series of Git commands and their output. The prompt is "zaphod:Repositories dos\$". The commands and output are: "git branch bugfix_73645", "git branch", "bugfix_73644", "bugfix_73645", "* master", and "git push origin bugfix_73645".

```
zaphod:Repositories dos$ git branch bugfix_73645
zaphod:Repositories dos$ git branch
  bugfix_73644
  bugfix_73645
* master
zaphod:Repositories dos$ git push origin bugfix_73645
```

Store your working directory and revert

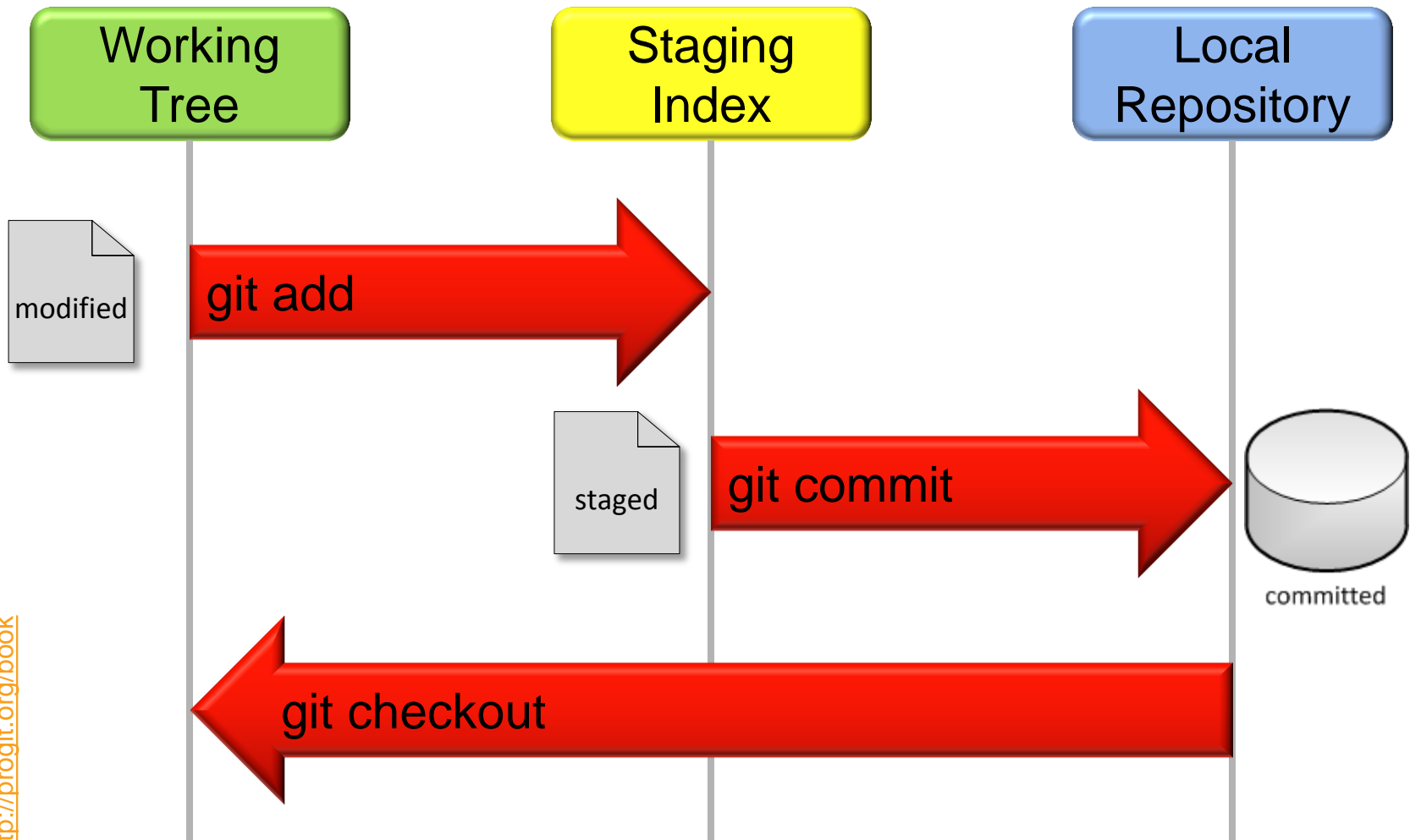


- Use **git stash** to record current working directory state
 - ▣ Saves current state of work
 - ▣ Resets working tree/ index to match latest version of current branch (a clean workspace)
 - ▣ Re-apply it at later to continue your work

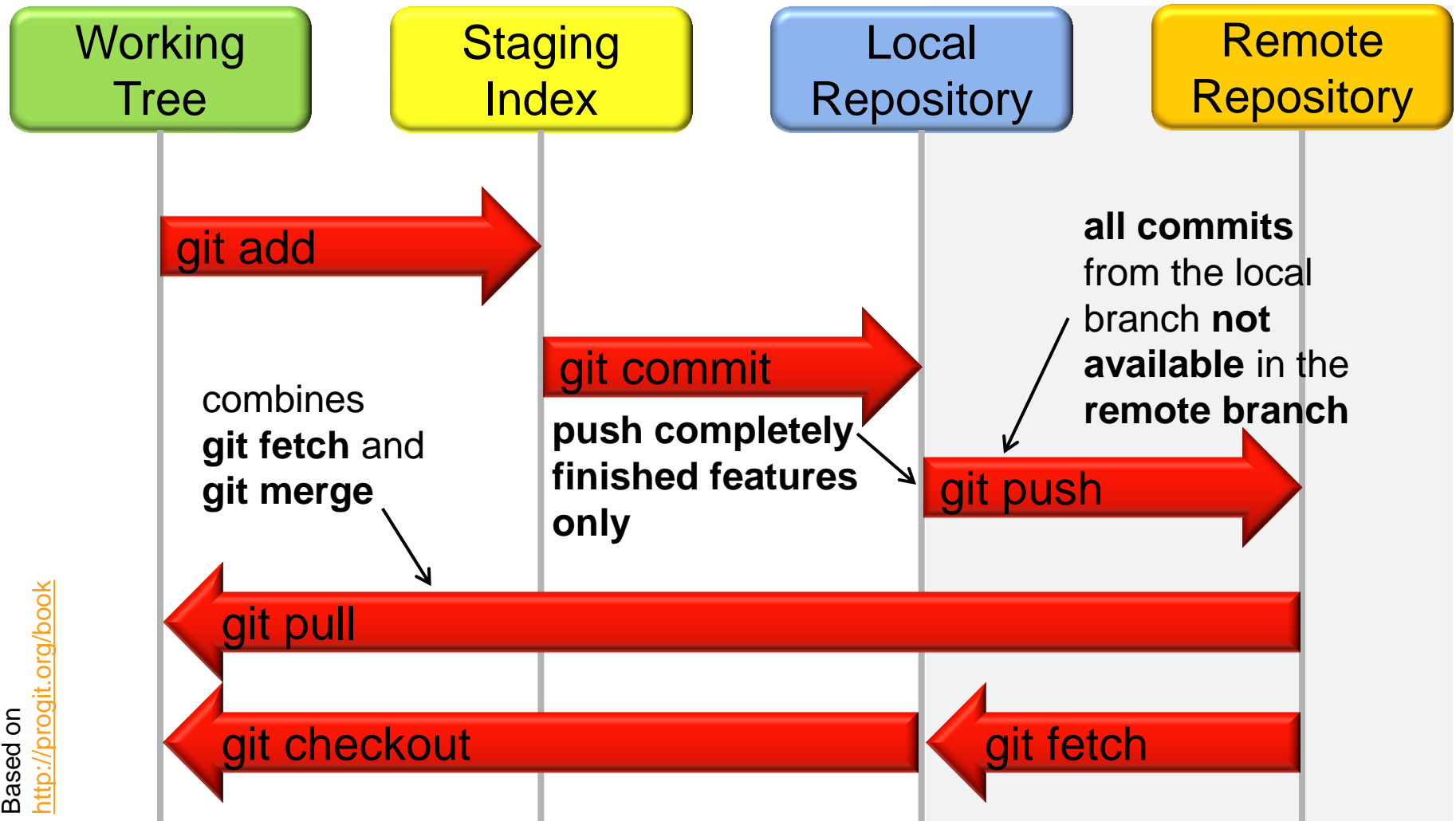
```
Terminal — bash
zaphod:Repositories dos$ git stash save "work in progress for bug 73648"
Saved working directory and index state On bugfix_73648: work in progress for bug 73648
HEAD is now at 0caab06 Changed comment
zaphod:Repositories dos$
zaphod:Repositories dos$ git stash apply
# On branch bugfix_73648
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   test.xml
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       readme
no changes added to commit (use "git add" and/or "git commit -a")
```

**Not supported
in EGit yet**

There are three main states/ sections in a Git project



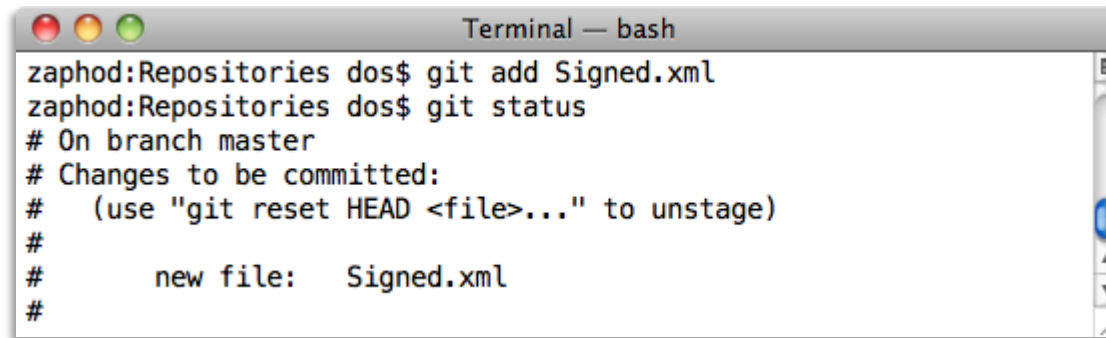
Changes flow between repositories by push and pull



The index is a staging area for the next commit



- Index is changed via **git add**

A screenshot of a macOS Terminal window titled "Terminal — bash". The window shows the following text:

```
zaphod:Repositories dos$ git add Signed.xml
zaphod:Repositories dos$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   Signed.xml
#
```

- State of the index becomes the tree of the next commit
 - ▣ Index provides an extra layer of control
 - ▣ Index is like an active changeset

Git tracks objects by their hash value



- Each blob is identified/ named by a SHA-1 hash
 - ▣ Git automatically computes the hash
 - Hash input is the objects content
 - Tamper-proof signature as a bonus
 - ▣ Blob does not contain any metadata
- Path and filename information is not considered
 - ▣ A renamed file is still linked with the original version
 - ▣ Sometimes problems with binary files
 - Even a small change might create a whole different hash
 - Relationship between new and original file might be lost

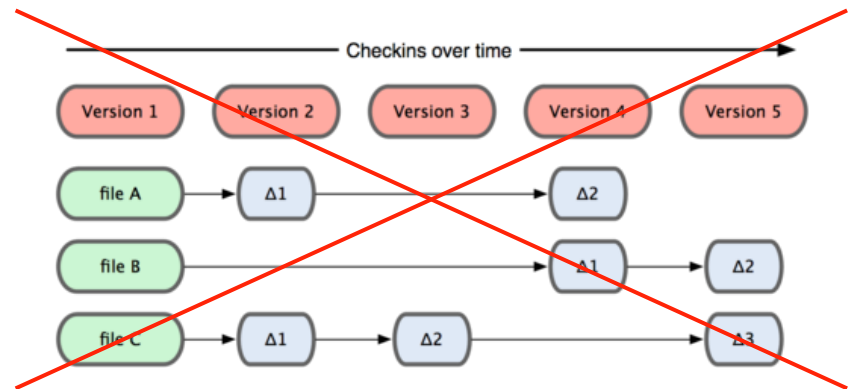
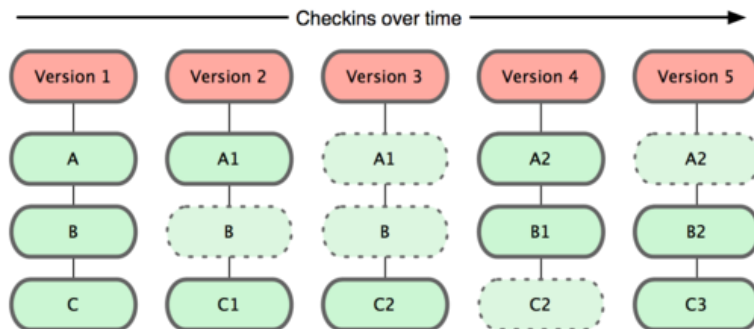
```
Terminal — bash
zaphod:Repositories dos$
zaphod:Repositories dos$
zaphod:Repositories dos$ git commit -m "Modified test file"
[master 633bc28] Modified test file
1 files changed, 1 insertions(+), 1 deletions(-)
```

A screenshot of a macOS Terminal window titled "Terminal — bash". The window shows a user at the "zaphod:Repositories dos\$" prompt. They enter the command "git commit -m 'Modified test file'". The output shows the commit is successful on the master branch with hash 633bc28, and that 1 file was changed with 1 insertion and 1 deletion.

The append-only object database



- Git stores each revision of a file as a unique blob object
 - Relationships between the blobs
 - Can be found through examining the tree and commit objects
 - Newly added objects are stored in their entirety
 - Git saves states, not deltas as Subversion
 - Uses zlib compression



Agenda



- Almost all about Git and EGit
- Push and pull, a typical day with Git
- The ultimate question of version control

A screenshot of a terminal window showing Git commit history and a diff. The commit log lists several commits by Dominik Schadow, including 'Merge branch 'bugfix_1000'', 'Added missing field to Data bean', 'Filled missing information', and 'Added initial project files'. The diff shows changes to a file, with lines added and removed. The commit hashes and dates are visible for each entry.

```
commit c47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
    Merge branch 'bugfix_1000'

commit d1ec4946988fb8e47464d20a8b9a6d155b73087a
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:23:14
    Added missing field to Data bean

commit 6957a53029201d4b8c91319b9c0457f62a304904
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:22:03
    Filled missing information

commit 6957a53029201d4b8c91319b9c0457f62a304904
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:20:46
    Added initial project files

commit 6957a53029201d4b8c91319b9c0457f62a304904
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:20:46
    Added initial project
```


Git command line

- ■ ■
- Git is available for Linux, Mac OS X and Windows
 - Windows command line is a little bit slower
- Clients/ command lines are in different development stages
 - Generally better and tighter integration on Linux and Mac OS X
- Some initial configuration required
 - Creates the *.gitconfig* file in your home directory
 - Via command line or Eclipse preferences



```
Terminal — bash
zaphod:~ dos$ git config --global user.name "Dominik Schadow"
zaphod:~ dos$ git config --global user.email "dominik.schadow@trivadis.com"
zaphod:~ dos$ git config --list
user.name=Dominik Schadow
user.email=dominik.schadow@trivadis.com
```

Git commands



```
Terminal — bash
usage: git [--version] [--exec-path[=<path>]] [--html-path]
        [-p|--paginate|--no-pager] [--no-replace-objects]
        [--bare] [--git-dir=<path>] [--work-tree=<path>]
        [-c name=value] [--help]
        <command> [<args>]

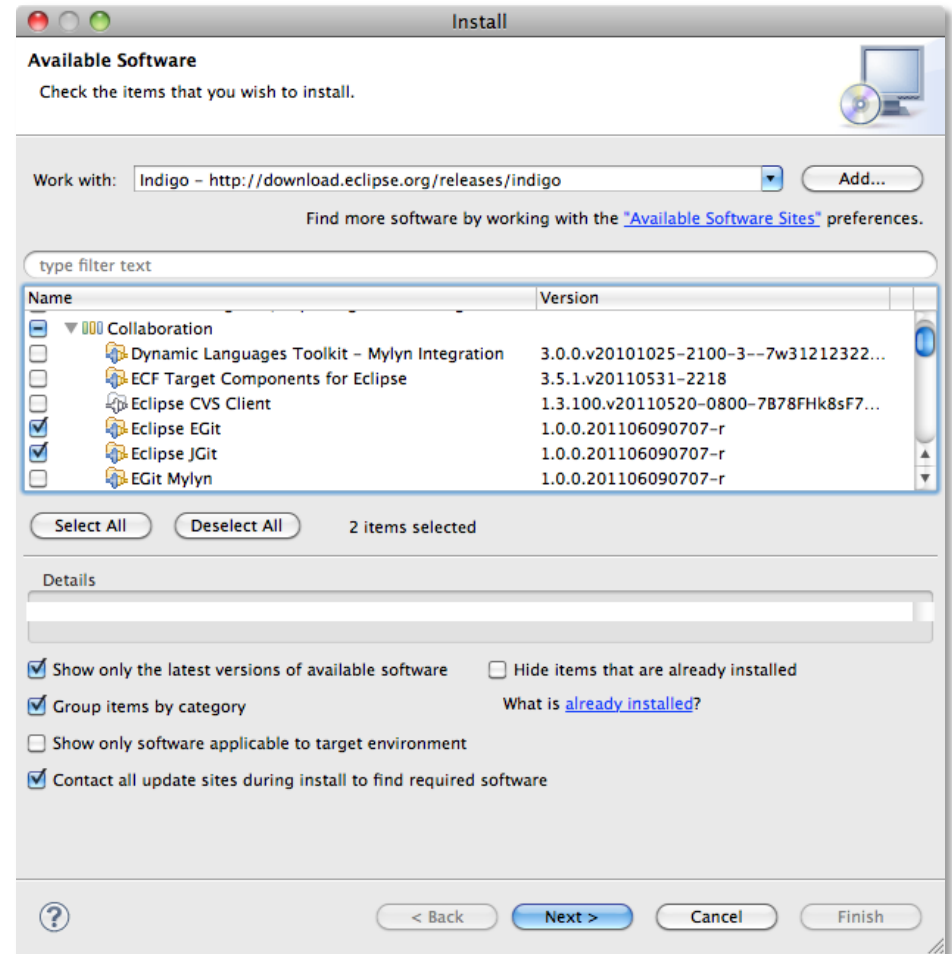
The most commonly used git commands are:
  add          Add file contents to the index
  bisect       Find by binary search the change that introduced a bug
  branch       List, create, or delete branches
  checkout     Checkout a branch or paths to the working tree
  clone        Clone a repository into a new directory
  commit       Record changes to the repository
  diff         Show changes between commits, commit and working tree, etc
  fetch        Download objects and refs from another repository
  grep         Print lines matching a pattern
  init         Create an empty git repository or reinitialize an existing one
  log          Show commit logs
  merge        Join two or more development histories together
  mv           Move or rename a file, a directory, or a symlink
  pull         Fetch from and merge with another repository or a local branch
  push         Update remote refs along with associated objects
  rebase       Forward-port local commits to the updated upstream head
  reset        Reset current HEAD to the specified state
  rm           Remove files from the working tree and from the index
  show         Show various types of objects
  status       Show the working tree status
  tag          Create, list, delete or verify a tag object signed with GPG

See 'git help <command>' for more information on a specific command.
```

EGit/ JGit installation via Eclipse update site

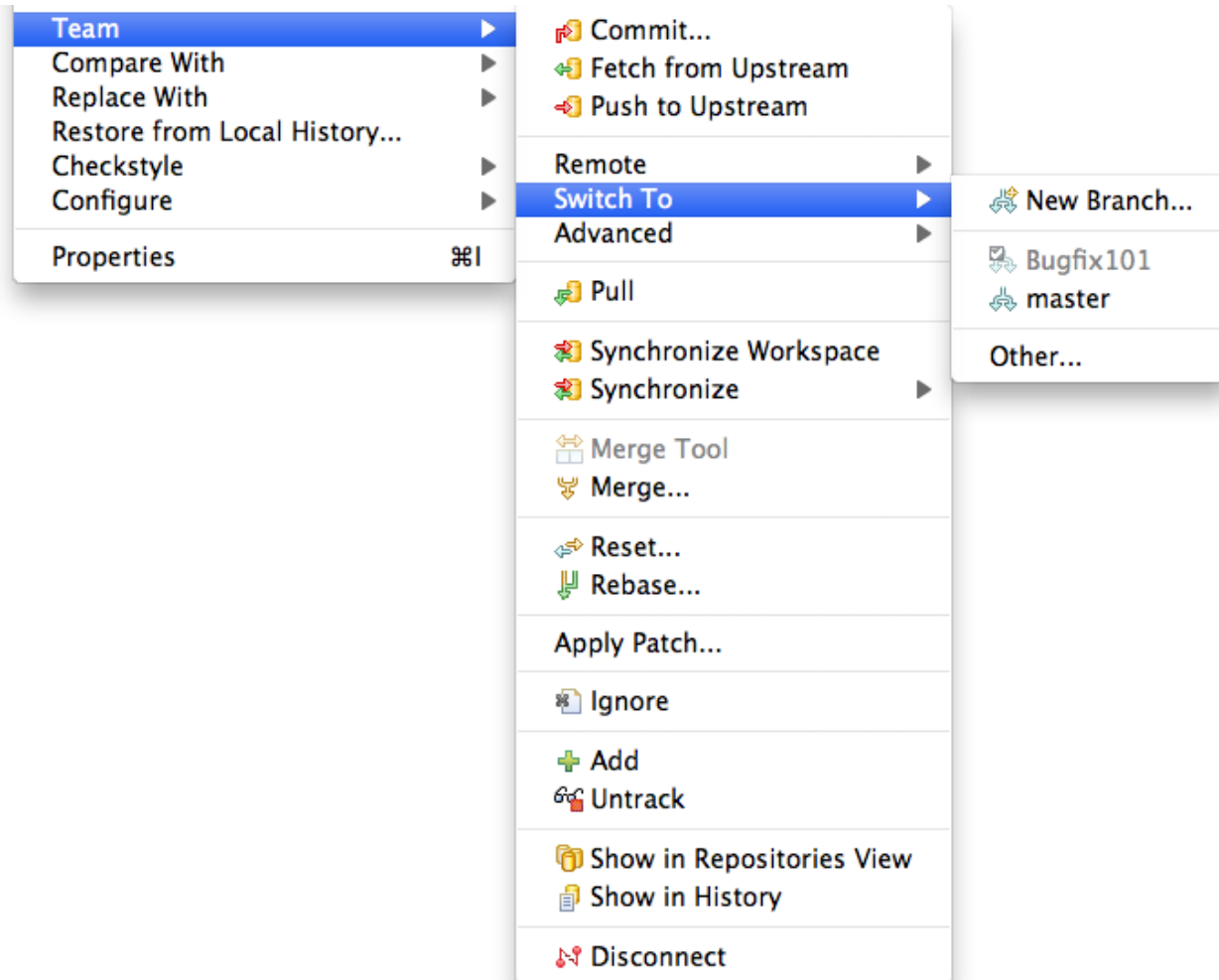


- Git command line is not required
 - But Plug-ins do not provide command line interface
- Install via update site
 - Eclipse EGit
 - Eclipse JGit



Before Indigo <http://download.eclipse.org/egit/updates>

EGit provides almost everything you need



Cloning an existing repository

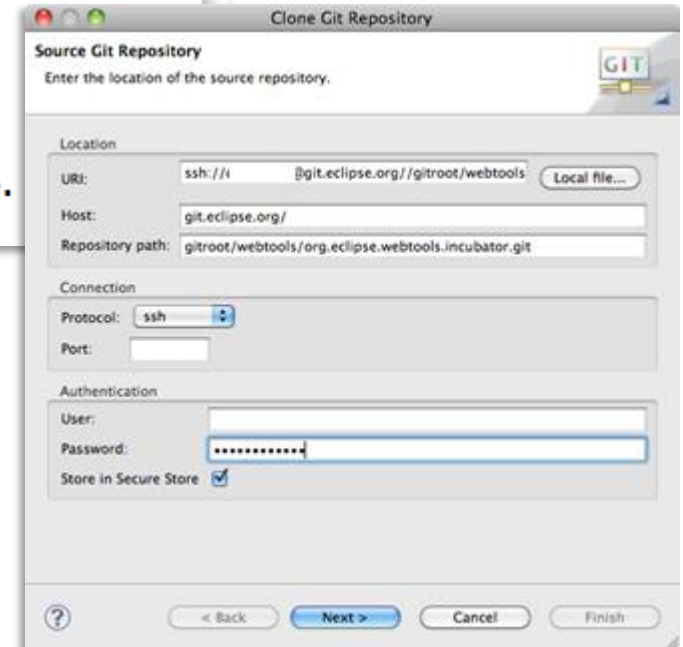


- Git clone automatically names the clone **master**
 - ▣ **master** is based on the remote **origin** branch
 - ▣ Creates a new directory
 - Using the Git repository name as directory name
 - Use optional *directory* parameter to specify a different name
- All its data is pulled to the local repository
 - ▣ A pointer to its master is created
 - ▣ Never modify the single **(one and only one)** .git directory
 - That is the Git repository
 - Exists only once in your repository root
 - Files/ directories under the parent of .git are the working tree

Initialize a new repository or clone an existing one



```
Terminal — bash
zaphod:eclipse dos$ git clone ssh://      @git.eclipse.org/gitroot/webtools/
org.eclipse.webtools.incubator.git source
Cloning into source...
Password:
remote: Counting objects: 38227, done.
remote: Compressing objects: 100% (12559/12559), done.
remote: Total 38227 (delta 18699), reused 31072 (delta 15256)
Receiving objects: 100% (38227/38227), 15.96 MiB | 377 KiB/s, done.
Resolving deltas: 100% (18699/18699), done.
```



Git supports many different protocols:
file, ftp, **git**, http, https, sftp, ssh

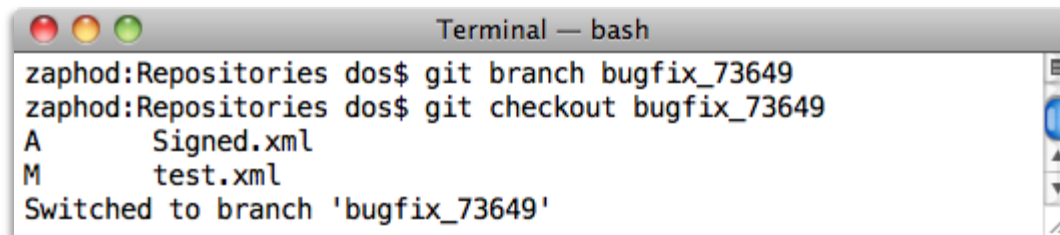
faster, more efficient, but read-only

```
Terminal — bash
zaphod:~ dos$ cd Repositories/
zaphod:Repositories dos$ git init
Initialized empty Git repository in /Users/dos/Repositories/.git/
zaphod:Repositories dos$
zaphod:Repositories dos$
```

Creating new branches

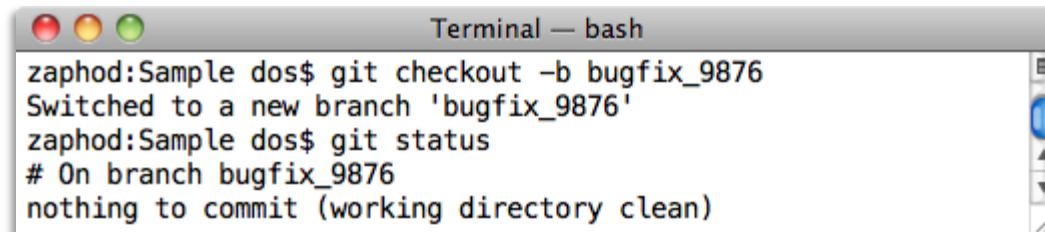


- Creating a new branch creates a new pointer (fast!)
 - ▣ Points to the same commit currently working on
 - ▣ Switch to the new branch with checkout



```
Terminal — bash
zaphod:Repositories dos$ git branch bugfix_73649
zaphod:Repositories dos$ git checkout bugfix_73649
A       Signed.xml
M       test.xml
Switched to branch 'bugfix_73649'
```

- Or create and switch with a single command



```
Terminal — bash
zaphod:Sample dos$ git checkout -b bugfix_9876
Switched to a new branch 'bugfix_9876'
zaphod:Sample dos$ git status
# On branch bugfix_9876
nothing to commit (working directory clean)
```

Merging is trivial in Git



- Each changeset tree node
 - Contains a pointer to its previous node
 - Back to the first commit
- Git knows what changes need to be made
 - And at what point in history they need to be applied
 - Automatically merges the given branch into the active one
- Listing the merged and unmerged branches

A screenshot of a terminal window titled "Terminal — bash". The prompt is "zaphod:Repositories dos\$". The first command is "git branch --merged", which outputs "bugfix_73644", "bugfix_73645", "bugfix_73648", and "* master". The second command is "git branch --no-merged", which outputs "bugfix_73649".

```
Terminal — bash
zaphod:Repositories dos$ git branch --merged
bugfix_73644
bugfix_73645
bugfix_73648
* master
zaphod:Repositories dos$ git branch --no-merged
bugfix_73649
```

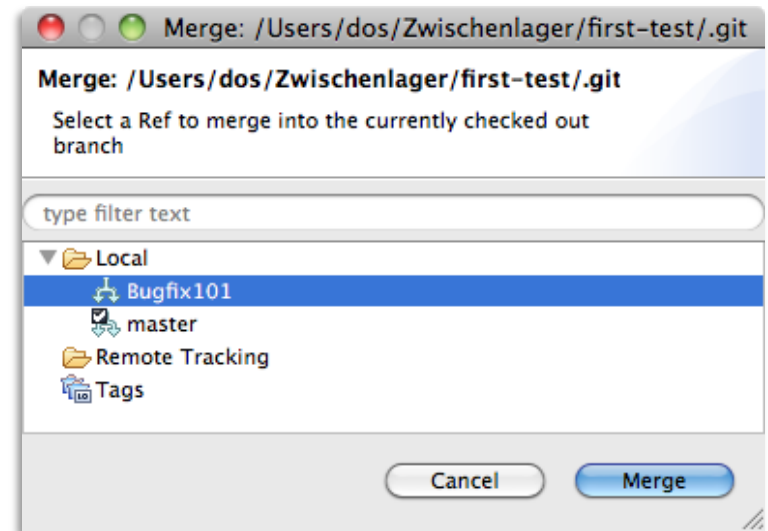
Not supported
in EGit

Switch to the branch to merge the changes in



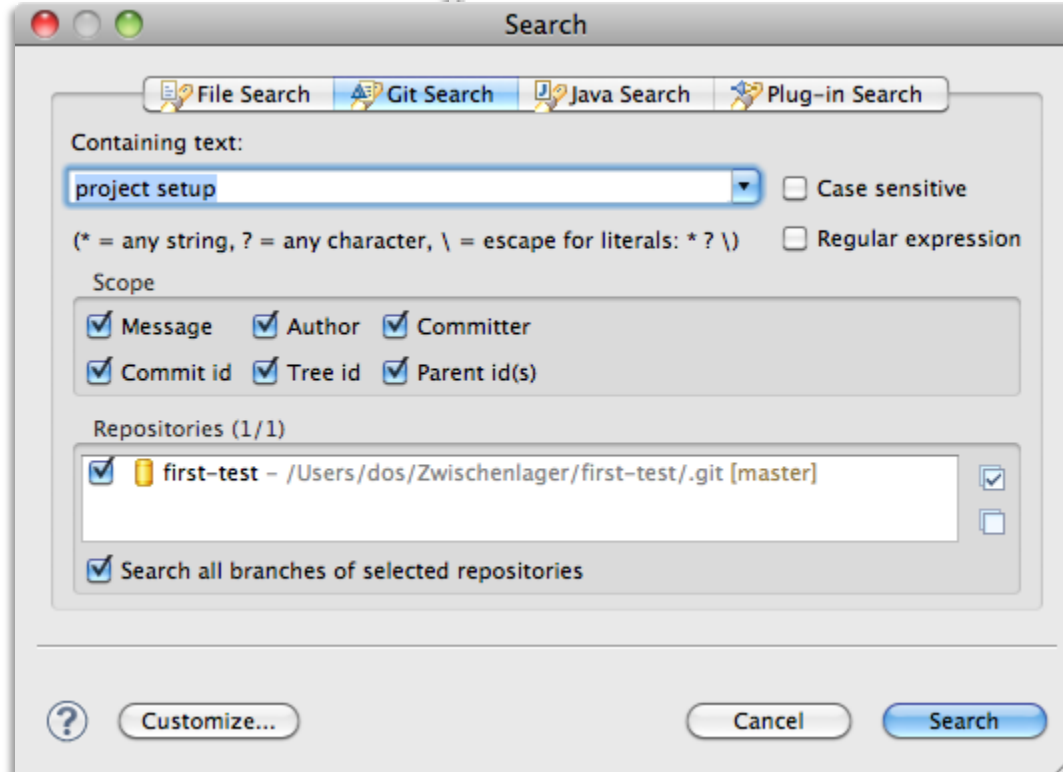
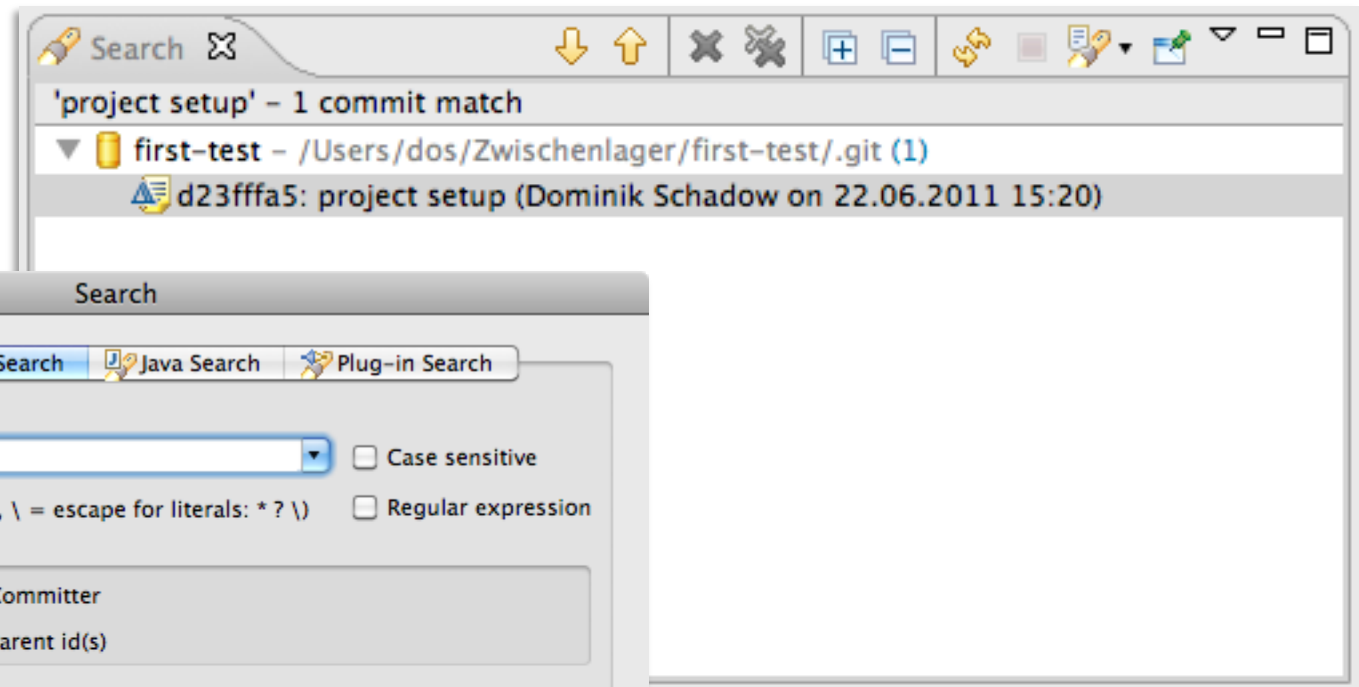
■ Use **git merge** and select the branch to integrate

- **Fast-forward merge**
 - Only the other branch changed
 - No merge operation required
- **Three-way merge**
 - Both branches changed
 - Don't expect miracles, conflicts happen: Resolve with merge tool or manually

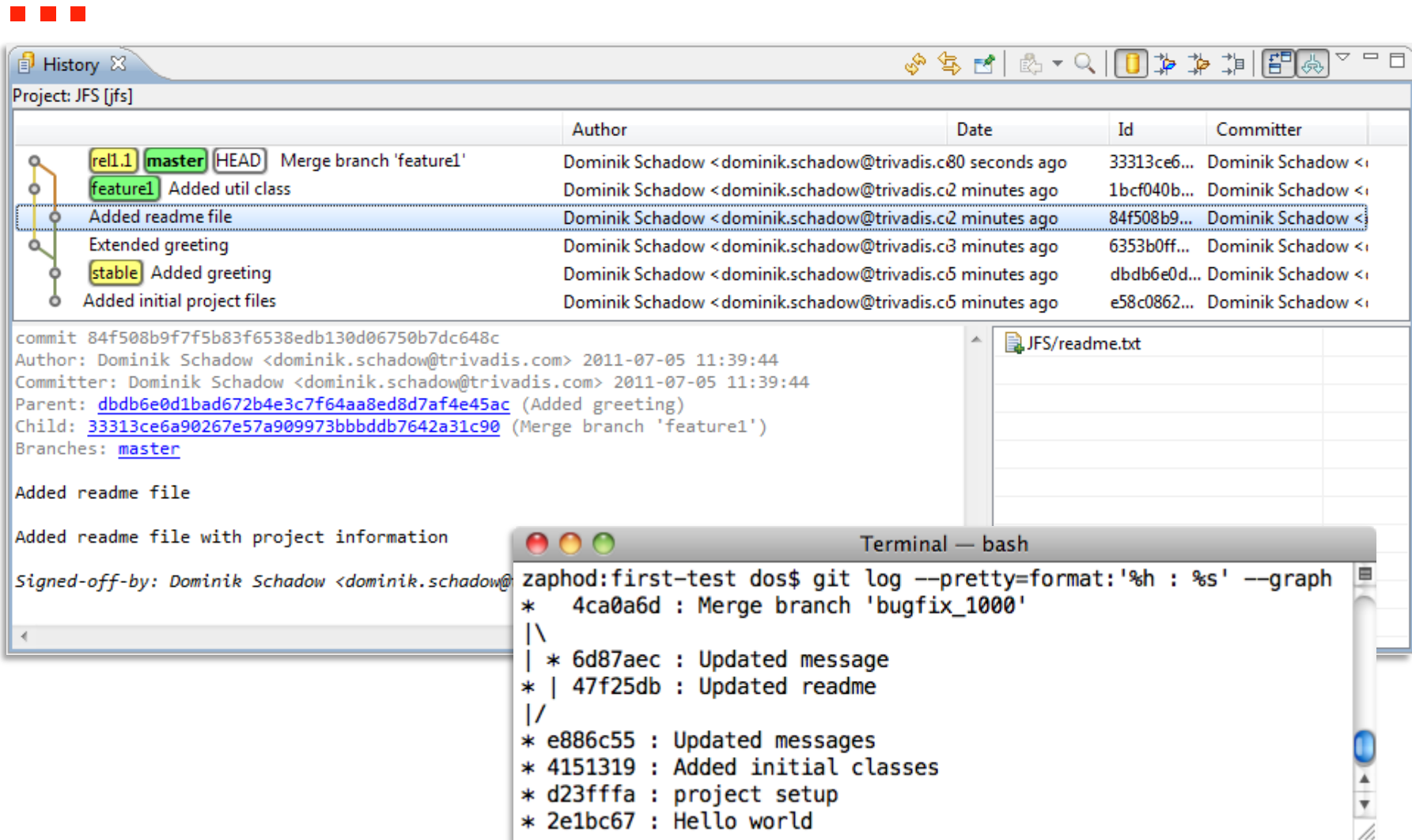


```
Terminal — bash
zaphod:Sample dos$ git checkout -b bugfix_1000
Switched to a new branch 'bugfix_1000'
zaphod:Sample dos$ git checkout master
Switched to branch 'master'
zaphod:Sample dos$ git merge bugfix_1000
Merge made by recursive.
Files/Readme | 4 +++-
src/com/trivadis/jfs/Main.java | 2 +-
2 files changed, 4 insertions(+), 2 deletions(-)
```

Searching for Git commits



EGit History view and the Git log



The screenshot displays the EGit History view for a project named 'JFS [jfs]'. The view shows a commit graph on the left and a table of commit details on the right. The selected commit is 84f508b9f7f5b83f6538edb130d06750b7dc648c, titled 'Added readme file', by Dominik Schadow. Below the table, the commit details are shown, including the author, committer, parent, and child commits. A terminal window in the foreground shows the output of the command `git log --pretty=format:'%h : %s' --graph`.

	Author	Date	Id	Committer
rel1.1 master HEAD Merge branch 'feature1'	Dominik Schadow <dominik.schadow@trivadis.c>	80 seconds ago	33313ce6...	Dominik Schadow <...
feature1 Added util class	Dominik Schadow <dominik.schadow@trivadis.c>	2 minutes ago	1bcf040b...	Dominik Schadow <...
Added readme file	Dominik Schadow <dominik.schadow@trivadis.c>	2 minutes ago	84f508b9...	Dominik Schadow <...
Extended greeting	Dominik Schadow <dominik.schadow@trivadis.c>	3 minutes ago	6353b0ff...	Dominik Schadow <...
stable Added greeting	Dominik Schadow <dominik.schadow@trivadis.c>	5 minutes ago	dbdb6e0d...	Dominik Schadow <...
Added initial project files	Dominik Schadow <dominik.schadow@trivadis.c>	5 minutes ago	e58c0862...	Dominik Schadow <...

commit 84f508b9f7f5b83f6538edb130d06750b7dc648c
Author: Dominik Schadow <dominik.schadow@trivadis.com> 2011-07-05 11:39:44
Committer: Dominik Schadow <dominik.schadow@trivadis.com> 2011-07-05 11:39:44
Parent: dbdb6e0d1bad672b4e3c7f64aa8ed8d7af4e45ac (Added greeting)
Child: 33313ce6a90267e57a9099973bbbddb7642a31c90 (Merge branch 'feature1')
Branches: master

Added readme file

Added readme file with project information

Signed-off-by: Dominik Schadow <dominik.schadow@trivadis.com>

```
zaphod:first-test dos$ git log --pretty=format:'%h : %s' --graph
* 4ca0a6d : Merge branch 'bugfix_1000'
|\
| * 6d87aec : Updated message
* | 47f25db : Updated readme
|/
* e886c55 : Updated messages
* 4151319 : Added initial classes
* d23fffa : project setup
* 2e1bc67 : Hello world
```

Agenda



- Almost all about Git and EGit
- Push and pull, a typical day with Git
- The ultimate question of version control

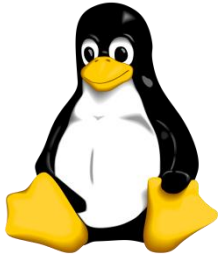
A screenshot of a Git commit log and file diff. The commit log shows a series of commits by Dominik Schadow, including 'Merge branch 'bugfix_1000'', 'Added missing field to Data bean', 'Filled missing information', 'Added initial project files', and 'Added initial project'. The file diff shows changes to a file named 'Data bean', with lines added and removed. The commit hash is 'c47b558465376301ebb689b0f9f1e35dc1f42ef5'.

```
commit c47b558465376301ebb689b0f9f1e35dc1f42ef5
Author: Dominik Schadow <dominik.schadow@trivadis.com>
Date: 2011-06-13 12:24:32
Merge branch 'bugfix_1000'

Added missing field to Data bean
Filled missing information
Added initial project files
Added initial project

c47b558465376301ebb689b0f9f1e35dc1f42ef5
Dominik Schadow <dominik.schadow@trivadis.com> 2011-06-13 12:24:32
Merge: Dominik Schadow <dominik.schadow@trivadis.com> 2011-06-13 12:24:32
d1ec4946988fb8e47464d20a8b9a6d155b73087a (Added missing field to Data bean)
6957a53029201d4b8c91319b9c0457f62a304904 (Filled missing information)
Files: master
branch 'bugfix_1000'
```

Git command line interfaces and tools



- gitg <http://trac.novowork.com/gitg>
- giggle <http://live.gnome.org/giggle>



- Git for OS X <http://code.google.com/p/git-osx-installer>
- GitX <http://gitx.frim.nl>



- cygwin <http://www.cygwin.com>
- msysGit <http://code.google.com/p/msysgit>
- TortoiseGit <http://code.google.com/p/tortoisegit>

Git IDE integration



- Useable version since EGit 0.11
- Stable version available with Eclipse Indigo
- Updates 1.1/2.0/2.1/2.2 already scheduled up to Eclipse 3.8



- Stable version available since version 10.x



- Usable version with some features available since version 7.0

Always keep in mind



- IDE integration
 - Sometimes still in an early stage
 - More updates in the future
- Usage concept
 - (totally) different from CVS/ SVN
- Build server integration
 - A plug-in for Git is required
 - Available for Hudson and Jenkins
- No central server
 - Makes backup of latest version more difficult

The first step is always the hardest

- ■ ■
- Create a new **branch** for every feature item, bug fix, ...
- **commit** as often as you like
 - **push** once when the feature, bug fix, ... is complete
- **reset** (revert) depends on where the changes are
 - Command line
 - **git checkout** *file* for not staged (not added) files
 - **git reset HEAD** *file* for staged files
 - EGit requires simple selection of reset type (soft, mixed, hard)
- SHA-1 hash value instead of a revision number
 - Usually the first six or seven characters are enough

(E)Git Pros and Cons



Performance: extremely fast even in large projects

Offline mode: no server connection required

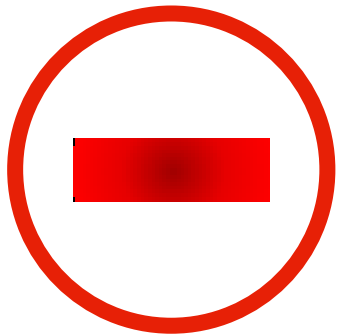
Branching/ merging: fast merging is done all the time

Fully distributed: no central server required

Repository size: requires less space as SVN

Search view: search for commits in Eclipse

Creativity: experimental branches for new ideas



Revisions: Hash value required for distributed versioning

No partial checkout: clones the entire repository

And the winner is...



- **EGit is ready**

- Use it for your next new project
- Faster and much more fun
- Some commands are not available in EGit yet
 - Install command line as well
 - Updates already scheduled until Eclipse 3.8

- As a temporary alternative

- Connect your existing repositories via
 - **git svn**
 - **git cvsimport**

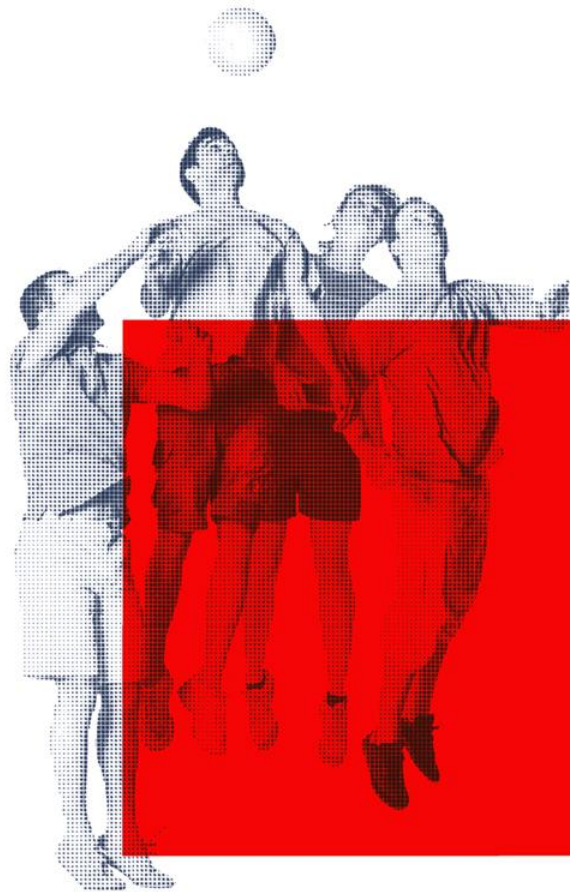
Keep in mind

Once started, it is very difficult to go back...

More information

- ■ ■
- Git <http://git-scm.com>
- Git Community Book <http://book.git-scm.com/>
- ProGit <http://progit.org>
- Git Cheat Sheet <http://ktown.kde.org/~zrusin/git/>
- GitHub www.github.com
- Eclipse JGit www.eclipse.org/jgit
- Eclipse EGit www.eclipse.org/egit
- Linus Torvalds on Git <http://www.youtube.com/watch?v=4XpnKHJAok8>
- It's time to stop using Subversion <http://altdevblogaday.org/2011/03/09/its-time-to-stop-using-subversion>

■ ■ ■ Thank you!



www.trivadis.com

trivadis
makes IT easier. ■ ■ ■



Basel

Bern

Lausanne

Zurich

Düsseldorf

Frankfurt/M.

Freiburg i. Br.

Hamburg

Munich

Stuttgart

Vienna