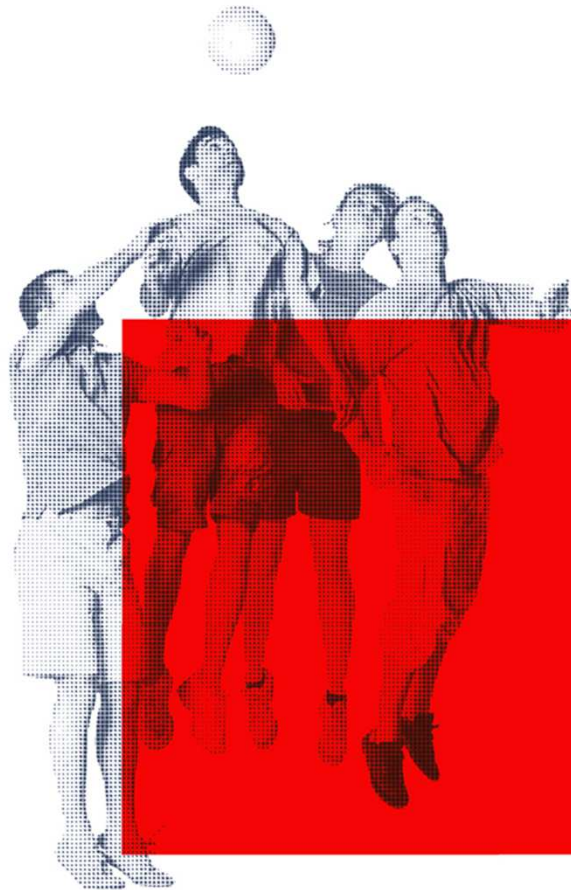


■ ■ ■ Push up your code – next generation version control with (E)Git



Dominik Schadow
Senior Consultant
Application Development

dominik.schadow@trivadis.com

Java Lounge, 05.04.2011

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

Agenda



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

Subversion and CVS have many disadvantages



- ⊕ Creating a branch is easy and fast
 - ⊖ Merging sucks (almost) all the time
 - ⊖ No local branches

- ⊕ Central repository server makes backups easy
 - ⊖ No distributed servers for distributed teams
 - ⊖ Clients require server connection for most operations

- ⊕ Performance is OK for small projects and some operations
 - ⊖ Slow merge, diff or switch operations
 - ⊖ Slows down as the project (history) grows larger

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



- Git clients fully mirror the repository
 - Not only the latest snapshot (revision)
 - Every clone is a complete backup
 - The whole repository is available locally
 - Copy of the entire development history
 - Complete repository with all branches and tags
- No network connection required
 - Most operations work offline
 - Commit/ merge/ diff/ branch and many more
 - Much faster
 - Most extreme, no central server is required
 - Local repository for private development

Branching and merging is easy and fast



- Branching and merging are an essential Git concept
 - Create local branch for each feature/ bugfix you work on
 - You can have many feature branches at any time
 - Easy to switch between them
 - No mix up of features in the same branch
- All branches are local after creation
 - Fast – no network communication required
 - Every developer's working copy is a private branch
- Branches can be shared with others
 - Most branches live only for a short time locally

Git is young, EGit and JGit even younger



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 Git migrations
2010 (May)	Eclipse projects start to migrate to Git
2010 (Sep)	EGit/ JGit 0.9
2011 (Feb)	EGit/ JGit 0.11.3
2011 (Jun)	EGit/ JGit 1.0 will be shipped with Eclipse Indigo

The story of Git, JGit and EGit



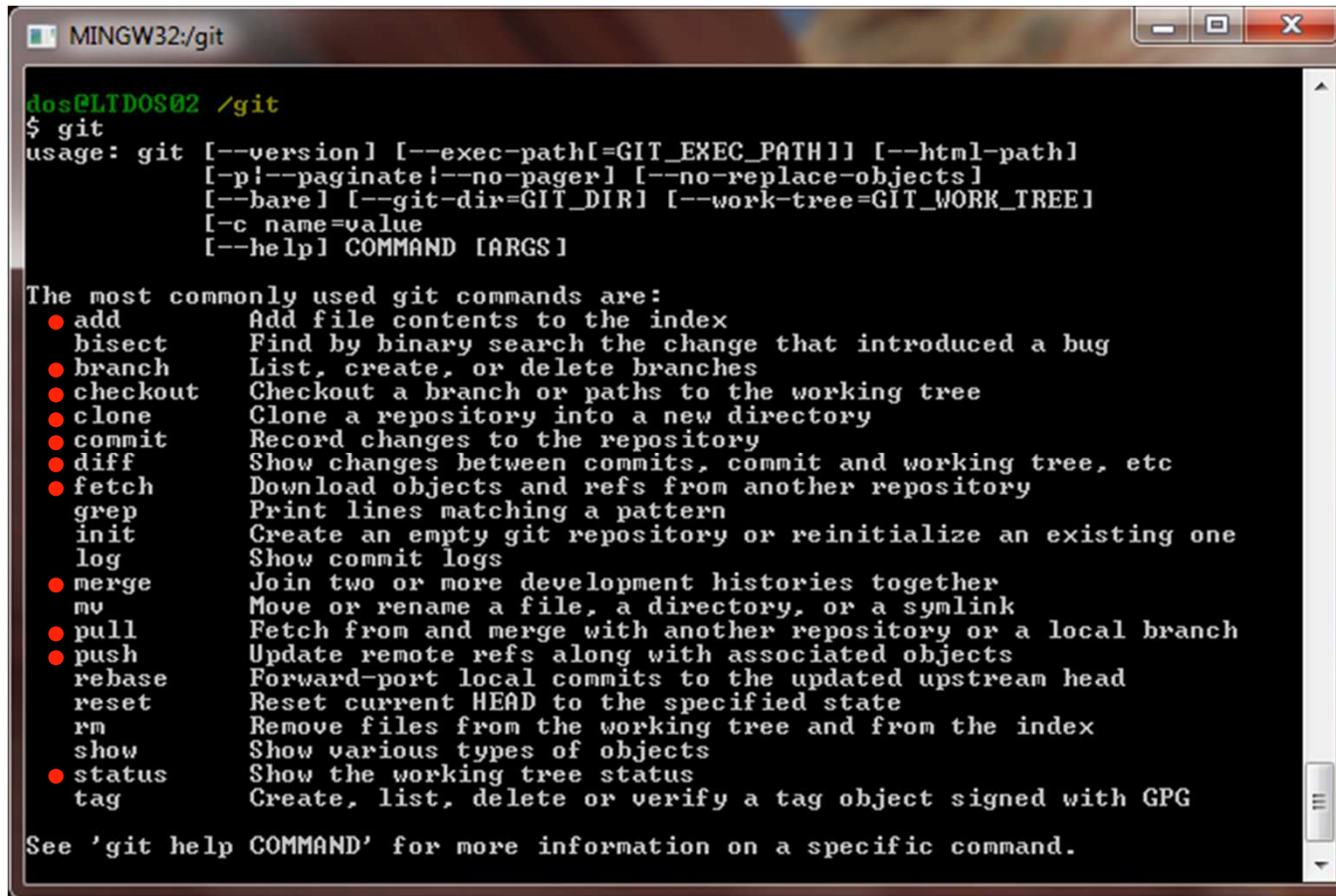
- The original **Git**
 - Original version developed by the Linux community
 - Distributed under the GNU General Public License (GPL)
- Official Eclipse projects must use the EPL
 - Eclipse Public License (EPL) and GPL are incompatible
 - Distribution chaos and installation trouble as with Subversion?
- JGit and EGit are official Eclipse projects
 - **JGit** is a lightweight Java library implementing Git
 - **EGit** is the Eclipse team provider and uses JGit

Closeup on JGit and EGit



- JGit and EGit are available in version 0.11.3
 - ▣ Plug-in (GUI) provides all features for *normal tasks*
 - Usable, but sometimes GUI does not offer all options
 - Git command line is sometimes required
 - Especially complicated operations may not be supported yet
 - ▣ 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
- Version 1.0 will be released mid 2011
 - ▣ Feature complete
 - No command line necessary any more
 - ▣ Full integration with Eclipse (Indigo)
 - Better than the Subversion provider integration

Git commands

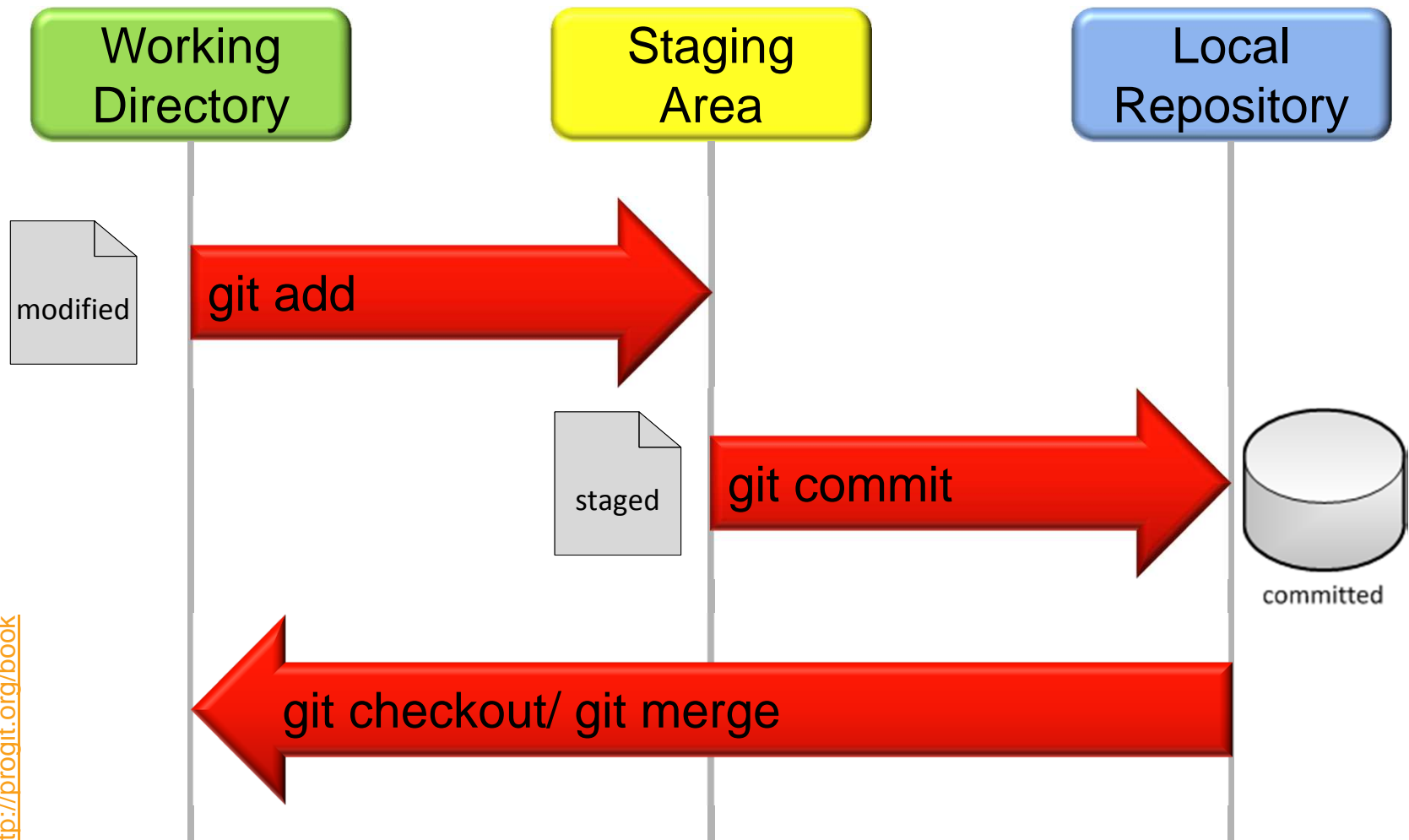


```
MINGW32/git
dos@LTD0S02 /git
$ git
usage: git [--version] [--exec-path=GIT_EXEC_PATH] [--html-path]
          [-p|--paginate|--no-pager] [--no-replace-objects]
          [--bare] [--git-dir=GIT_DIR] [--work-tree=GIT_WORK_TREE]
          [-c name=value]
          [--help] COMMAND [ARGS]

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

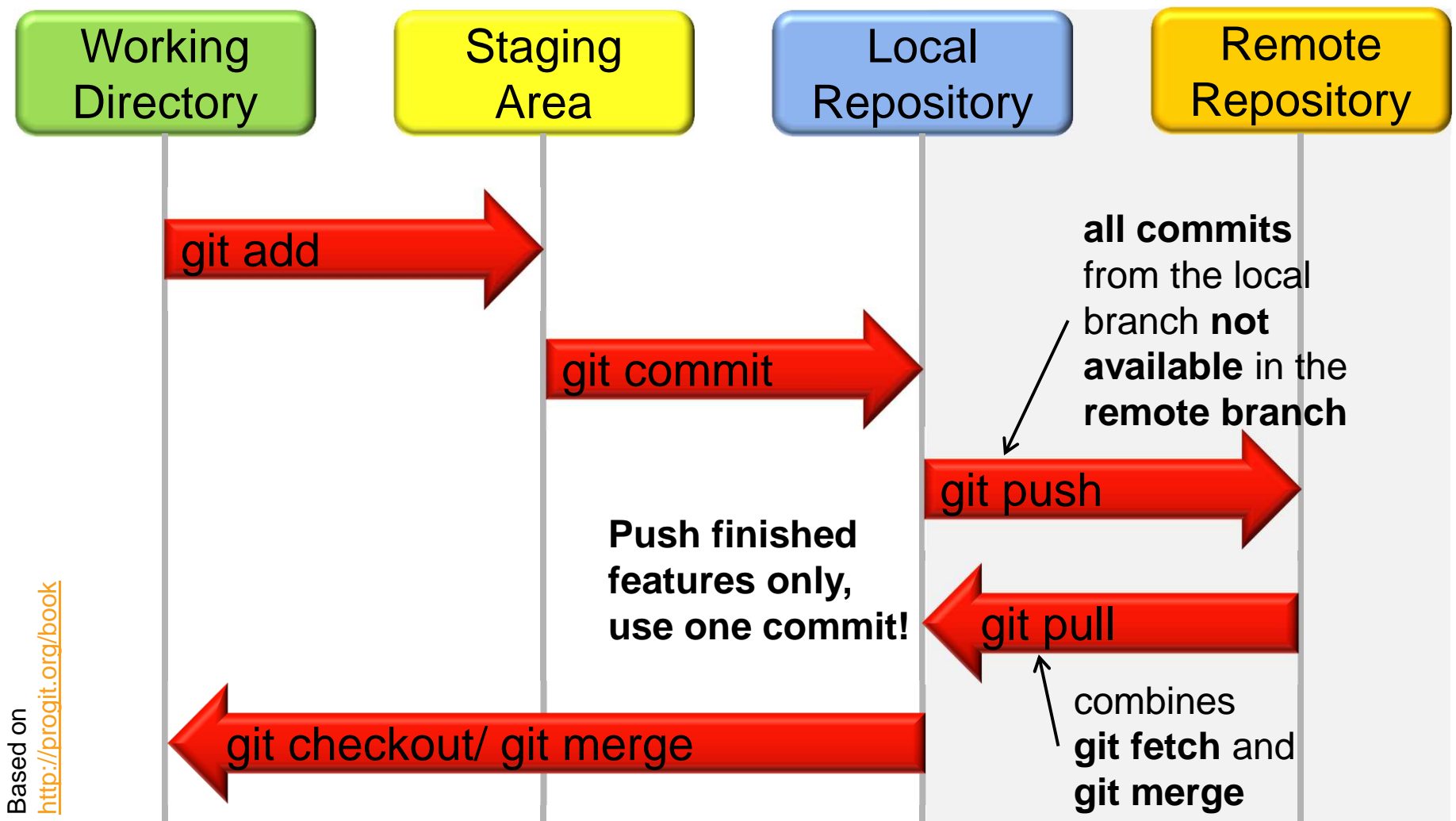
See 'git help COMMAND' for more information on a specific command.
```

There are three main states/ sections in a Git project



Based on
<http://progit.org/book>

Push and pull with a remote repository



Git tracks files by their content



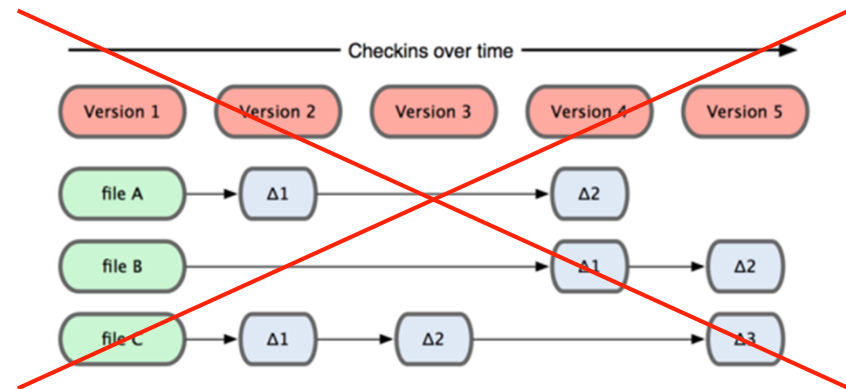
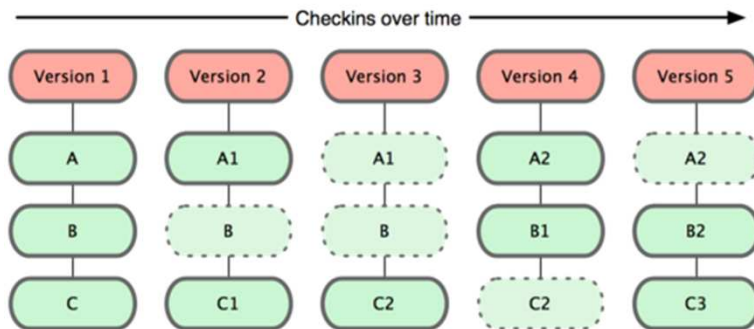
- Each object is identified by a SHA-1 hash of its contents
 - Value is used as the object's name
 - Git computes the hash
- Path and filename information is normally not considered
 - A renamed file is still linked with the original version
 - Sometimes problems with binary files
 - Even a small change might create a whole different hash
 - Relationship between new and original file might be lost

```
dos@LTDOS02 /c/DATA/Git/TechEvent (bug7483)
$ git commit -m "Modified Readme"
[bug7483 8d6d12e] Modified Readme
1 files changed, 1 insertions(+), 1 deletions(-)
```

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
 - Using zlib compression



Based on
<http://progit.org/book>

Start by cloning an existing repository



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

Clone, create and a wizard

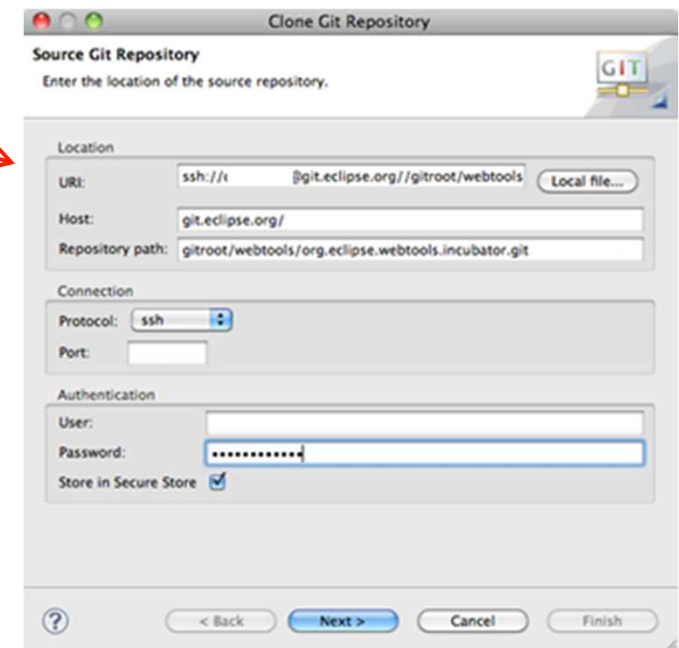


```
MINGW32:/c/DATA/WTP
dos@LTD0S02 /c/DATA/WTP
$ git clone ssh://[redacted]@git.eclipse.org/gitroot/webtools/org.eclipse.webtool
s.incubator.git source
Cloning into source...
Password:
remote: Counting objects: 32153, done.
remote: Compressing objects: 100% (9354/9354), done.
remote: Total 32153 (delta 15836), reused 29770 (delta 14599)
Receiving objects: 100% (32153/32153), 14.59 MiB | 41 KiB/s, done.
Resolving deltas: 100% (15836/15836), done.
```

init

```
MINGW32:/c/DATA/Dummy
dos@LTD0S02 /c/DATA/Dummy
$ git init
Initialized empty Git repository in c:/DATA/Dummy/.git/
dos@LTD0S02 /c/DATA/Dummy (master)
$
```

clone



Branching and merging is fast, easy and fun



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

<http://progit.org/book>

- Push to share branches
 - Branches are never automatically shared with remote repository
 - Simply type **git push (remote) (branch)**

The origin/master branch cannot be deleted



- Creating a new branch creates a new pointer
 - Points to the same commit currently working on
 - A manual switch to the new branch is required

```
MINGW32:/c/DATA/Git/TechEvent
dos@LTIDOS02 /c/DATA/Git/TechEvent <master>
$ git branch bug7483

dos@LTIDOS02 /c/DATA/Git/TechEvent <master>
$ git checkout bug7483
Switched to branch 'bug7483'

dos@LTIDOS02 /c/DATA/Git/TechEvent <bug7483>
$
```

- Listing the merged and unmerged branches

```
MINGW32:/c/DATA/Git/TechEvent
dos@LTIDOS02 /c/DATA/Git/TechEvent <master>
$ git branch --merged
dummy
* master
test

dos@LTIDOS02 /c/DATA/Git/TechEvent <master>
$ git branch --no-merged
```

Merging is normally done automatically



- Switch to the branch you intend to merge the changes in
 - Use **git merge** with the branch name you want to integrate
 - **Fast-forward merge** (only one branch changed) or **three-way merge** (both branches changed)
 - A commit is executed automatically (can be switched off)

A screenshot of a terminal window titled 'MINGW32:/c/DATA/Git/TechEvent'. The terminal shows the following sequence of commands and output:

```
dos@LTDOS02 /c/DATA/Git/TechEvent <master>
$ git checkout test
Switched to branch 'test'

dos@LTDOS02 /c/DATA/Git/TechEvent <test>
$ vim Readme.txt

dos@LTDOS02 /c/DATA/Git/TechEvent <test>
$ git commit -a -m 'Updated readme message'
[test d6396ae] Updated readme message
1 files changed, 1 insertions(+), 1 deletions(-)

dos@LTDOS02 /c/DATA/Git/TechEvent <test>
$ git checkout master
Switched to branch 'master'

dos@LTDOS02 /c/DATA/Git/TechEvent <master>
$ git merge test
Updating a7db580..d6396ae
Fast-forward
 README.txt | 2 +-
1 files changed, 1 insertions(+), 1 deletions(-)

dos@LTDOS02 /c/DATA/Git/TechEvent <master>
$
```

Agenda



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

Git command line installation



- 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
- Configuration file requires some work
 - *.gitconfig* in user home directory

Git command line interfaces and tools



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

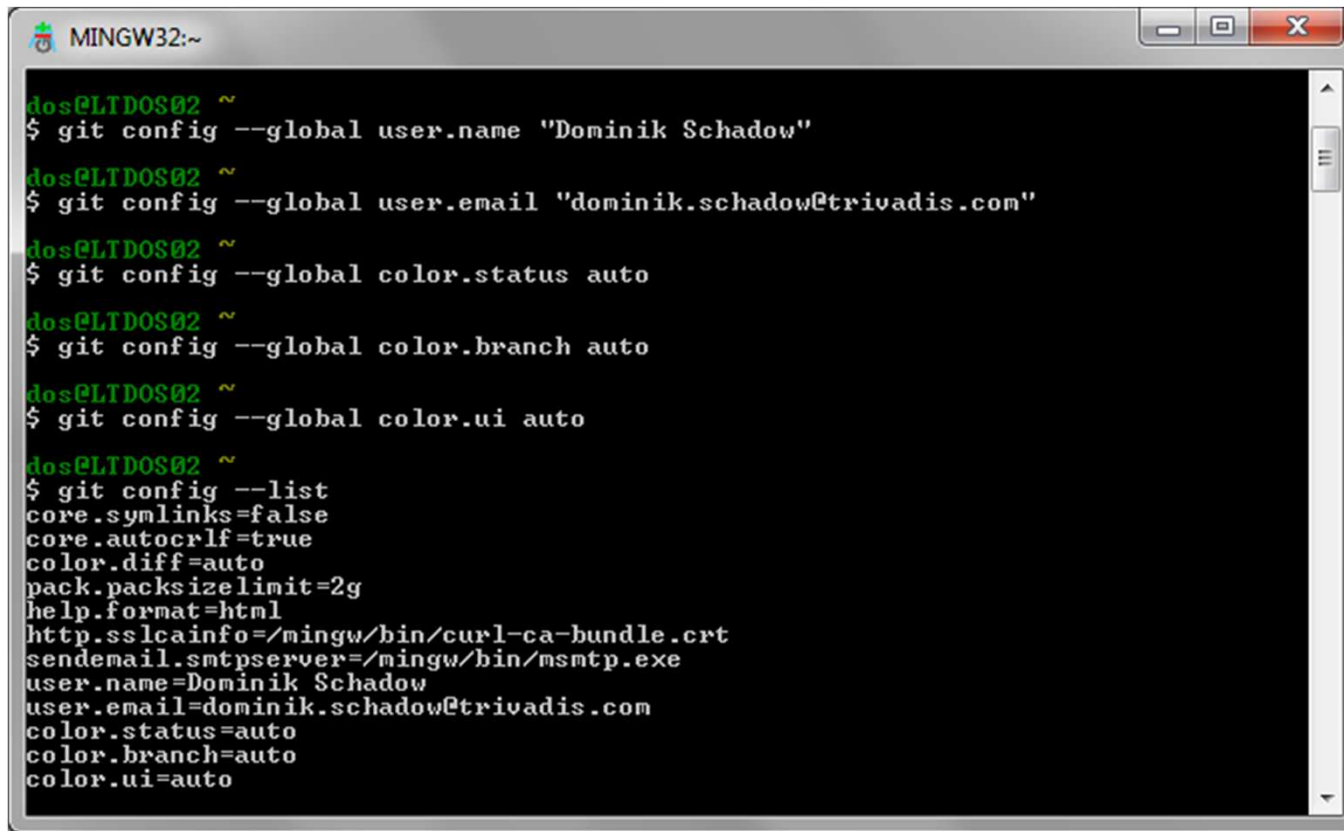


- Git for OS X <http://code.google.com/p/git-osx-installer>
- GitX <http://gitx.frim.nl>



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

Initial configuration requires some information



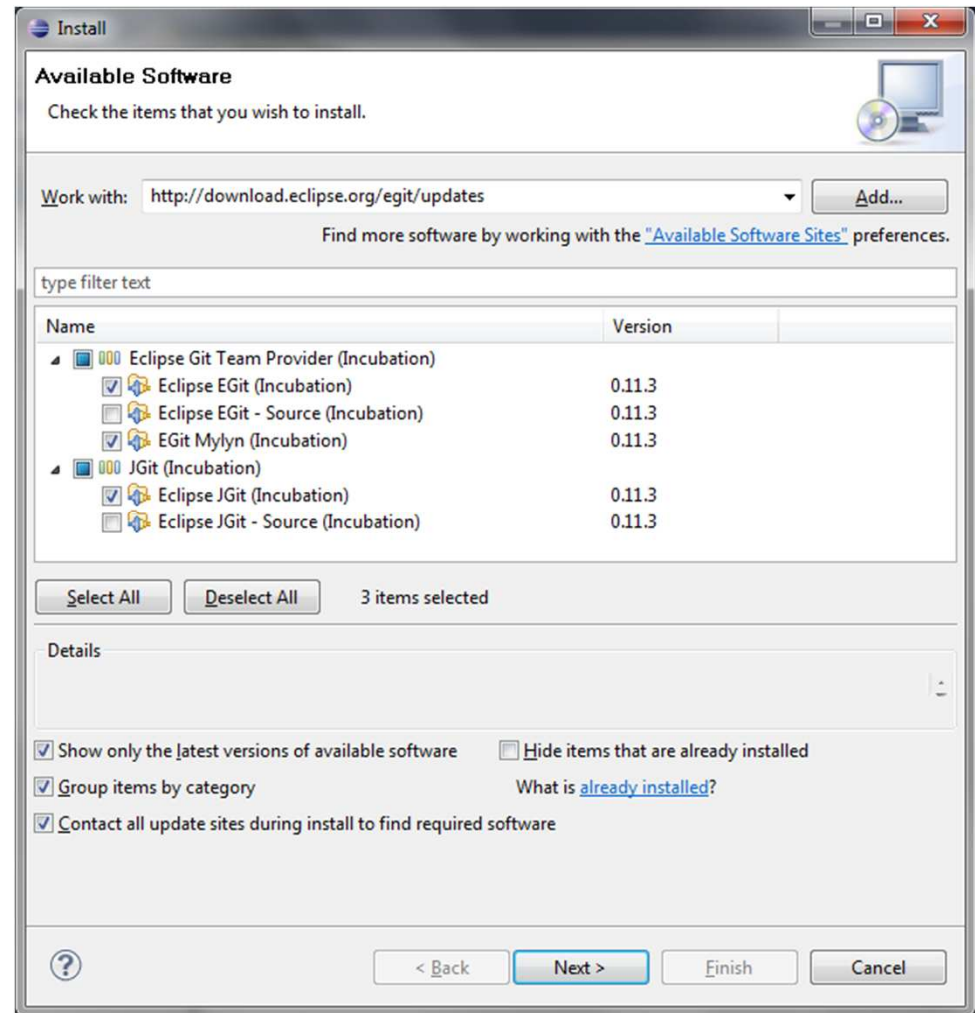
```
dos@LTDOS02 ~  
$ git config --global user.name "Dominik Schadow"  
dos@LTDOS02 ~  
$ git config --global user.email "dominik.schadow@trivadis.com"  
dos@LTDOS02 ~  
$ git config --global color.status auto  
dos@LTDOS02 ~  
$ git config --global color.branch auto  
dos@LTDOS02 ~  
$ git config --global color.ui auto  
dos@LTDOS02 ~  
$ git config --list  
core.symlinks=false  
core.autocrlf=true  
color.diff=auto  
pack.packsizelimit=2g  
help.format=html  
http.sslcainfo=/mingw/bin/curl-ca-bundle.crt  
sendemail.smtpserver=/mingw/bin/msmtp.exe  
user.name=Dominik Schadow  
user.email=dominik.schadow@trivadis.com  
color.status=auto  
color.branch=auto  
color.ui=auto
```

Type **git help config** for more information on parameters
On Windows, this can be done in the Eclipse EGit plug-in.

EGit/ JGit installation and configuration



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



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

A typical day with Git

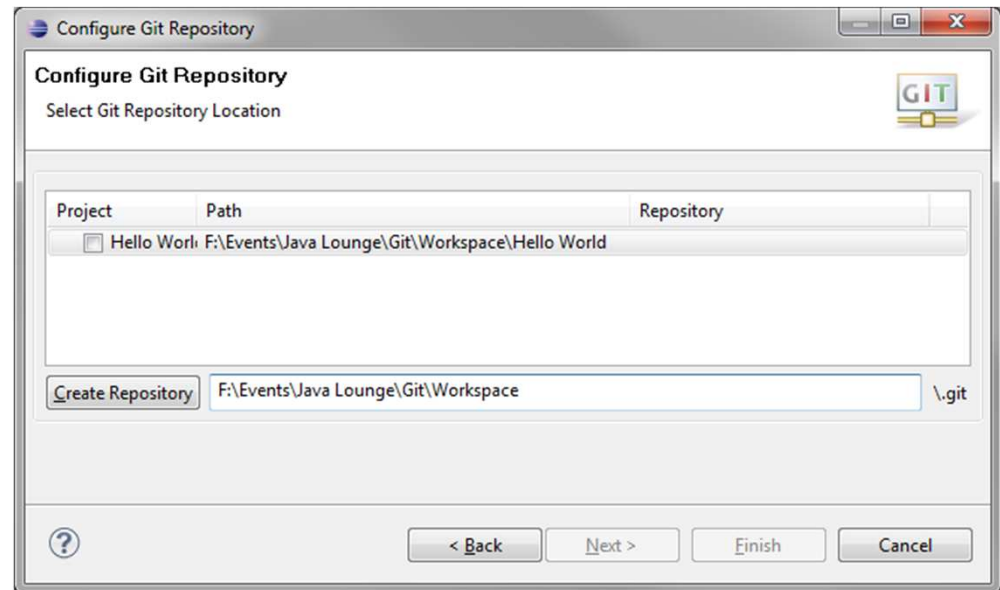
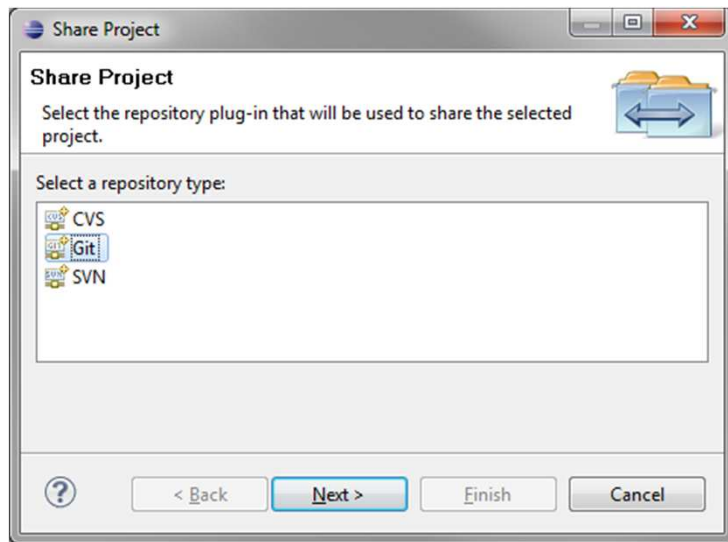


1. Share a new project and create a new Git repository
2. Add and commit all files
3. Modify a file and commit it
4. Create a new branch Bugfix and switch to it
5. Change a file and commit it
6. Switch back to the master branch
7. Merge it with the Bugfix branch
8. Show the changes in the History view

A typical day with Git (1)



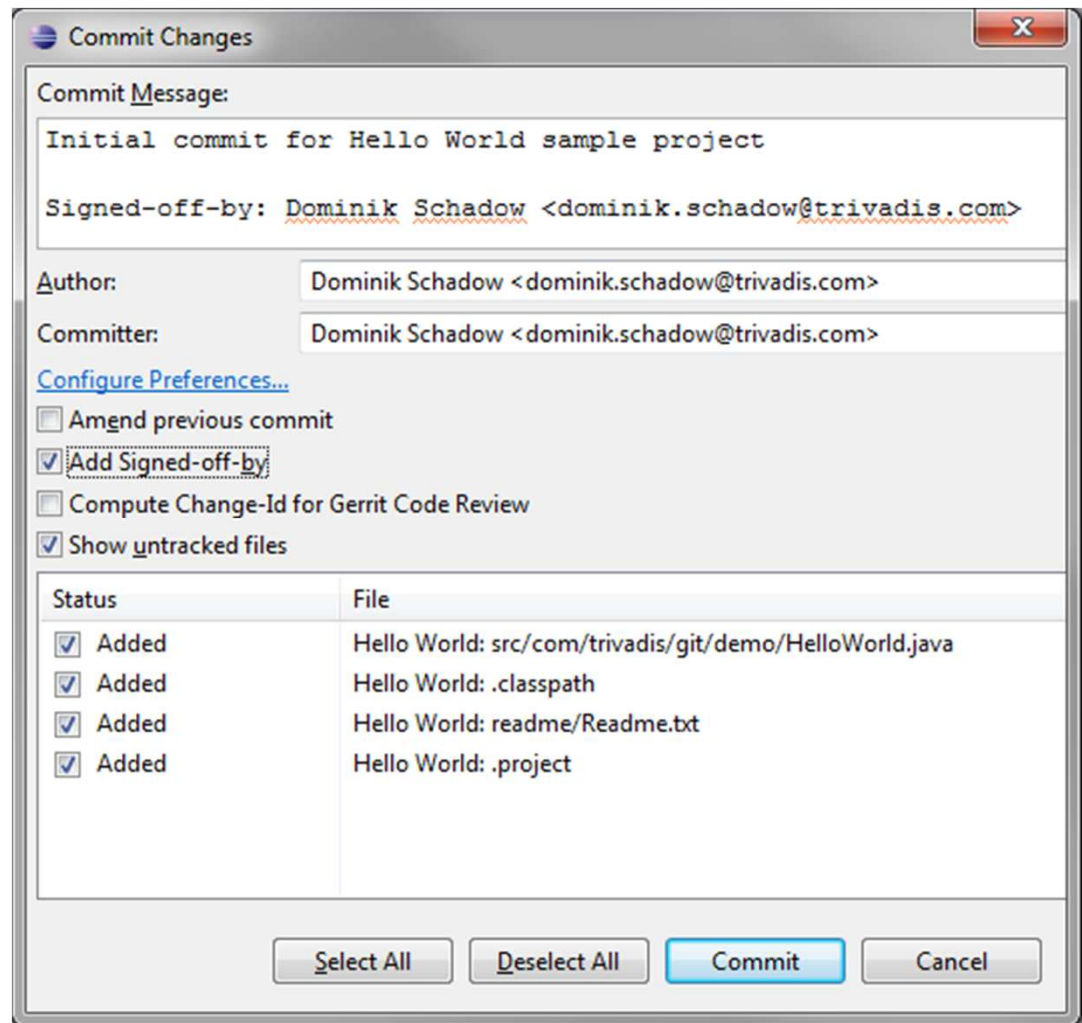
- Share a new project and create a new Git repository
 - Move repository folder up one level, do not create the repository inside the project
 - Click **Create Repository** and **Finish** when done



A typical day with Git (2)



- Add all files via **Team → Add**
- Commit them via **Team → Commit**
- Click the **Add Signed-off-by** checkbox



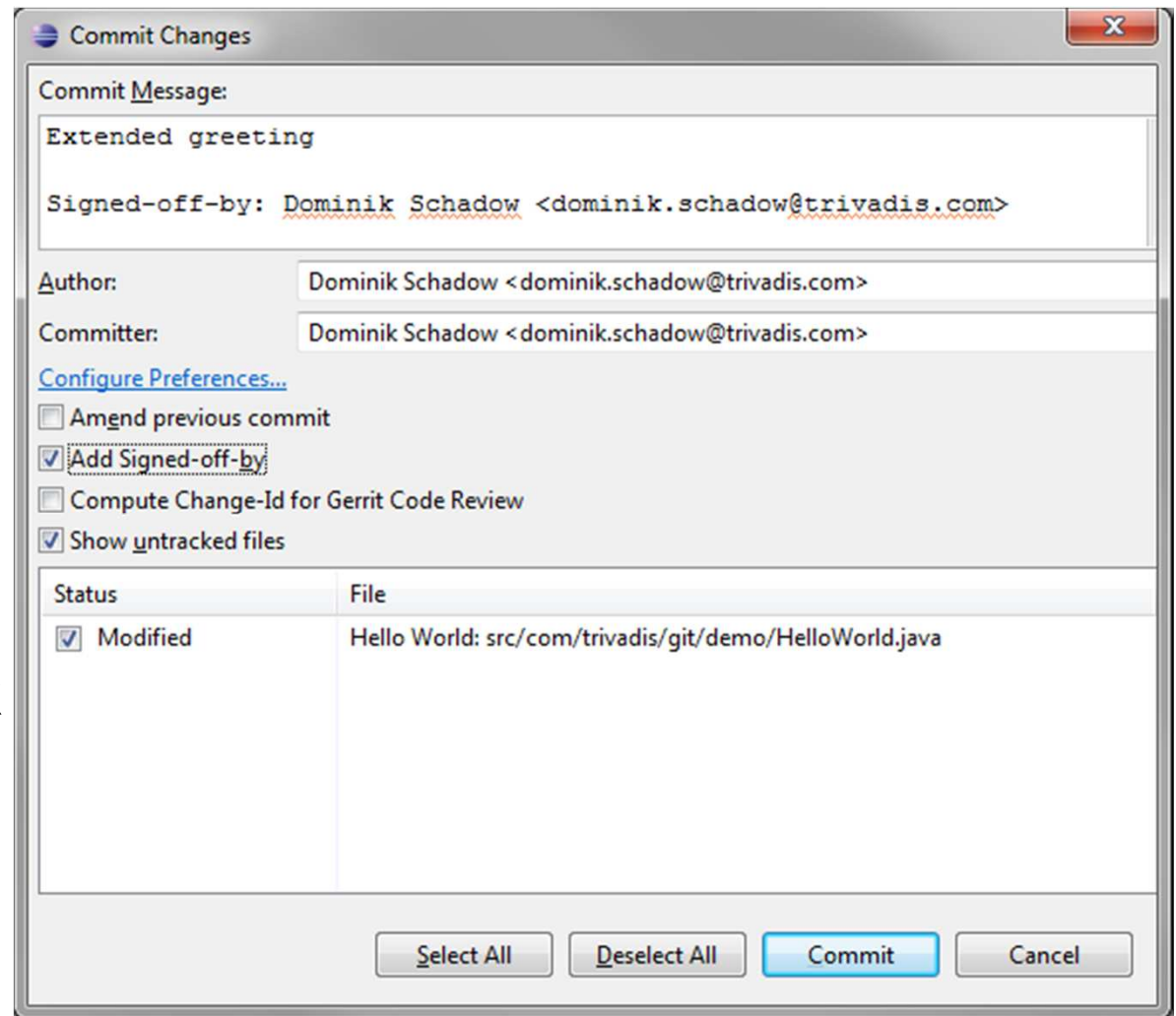
□ □ □



A typical day with Git (4)



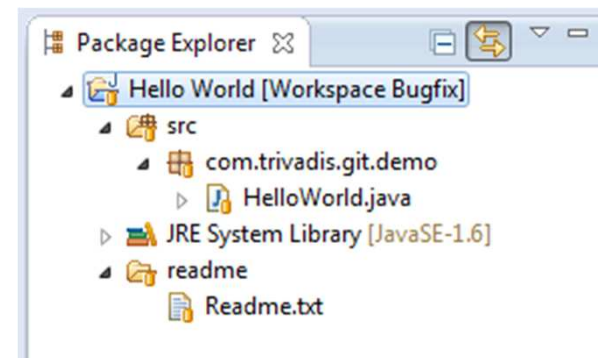
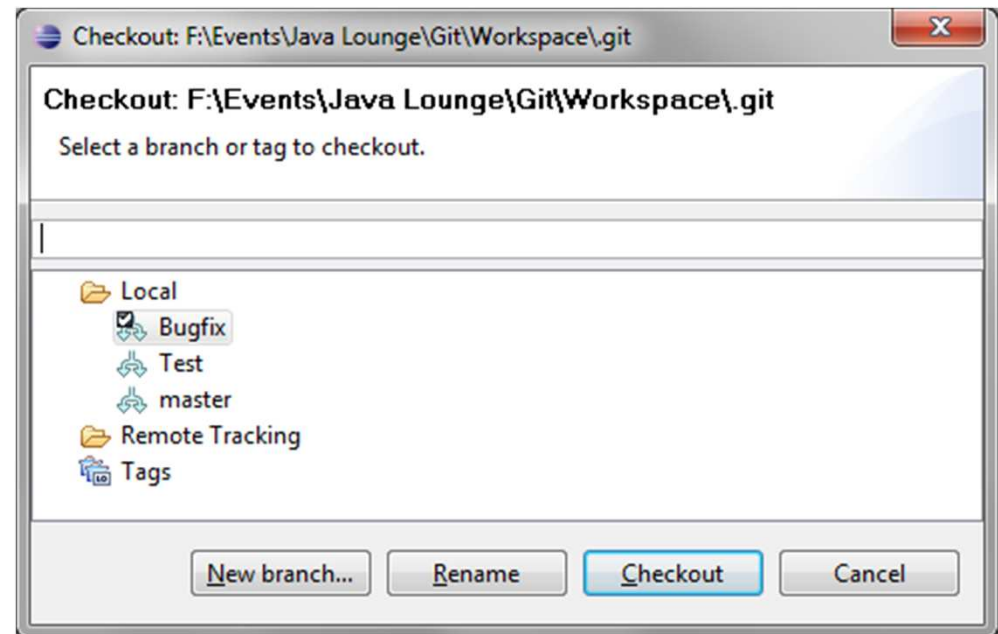
- Change the message in the Java class
- Click **Team** → **Add** on the file
- Commit the file via **Team** → **Commit**
- Don't forget to check the **Add Signed-off-by** checkbox



A typical day with Git (5)



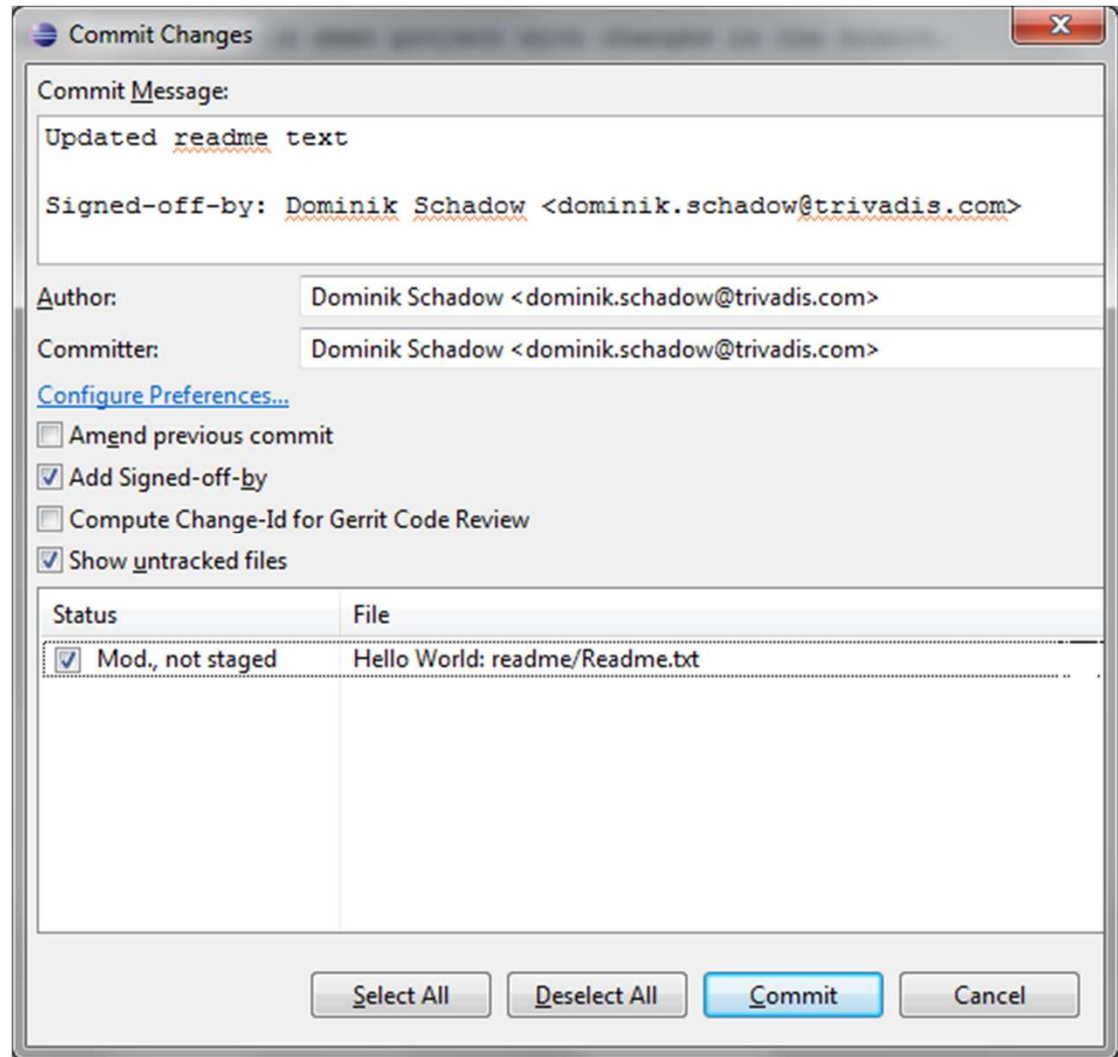
- Create a new branch named **Bugfix** via **Team** → **Branch** → **New branch...**
- The new branch is activated automatically
- See how the repository path in the Package Explorer changed



A typical day with Git (6)



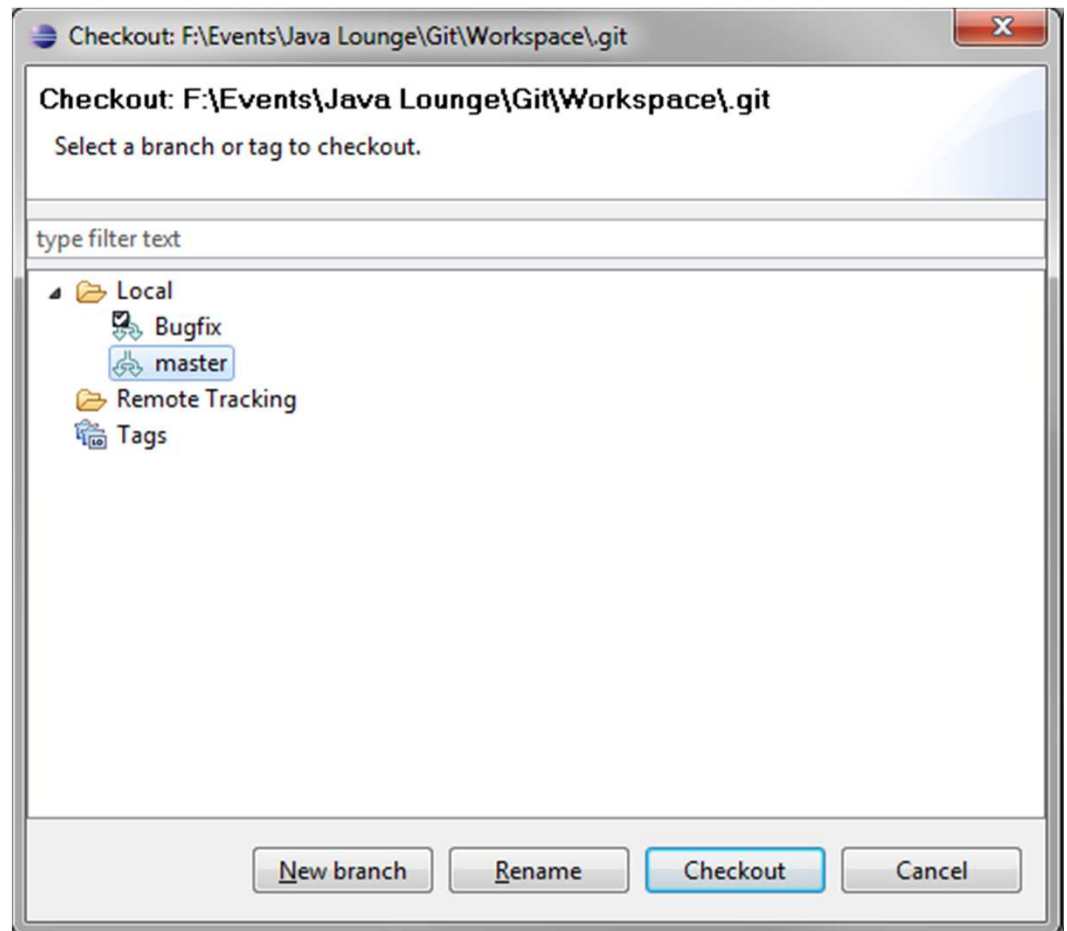
- Change the Readme.txt
- Use **Team** → **Commit** without **Team** → **Add** before
- Enter the message, select signed-off-by and commit the changes



A typical day with Git (7)



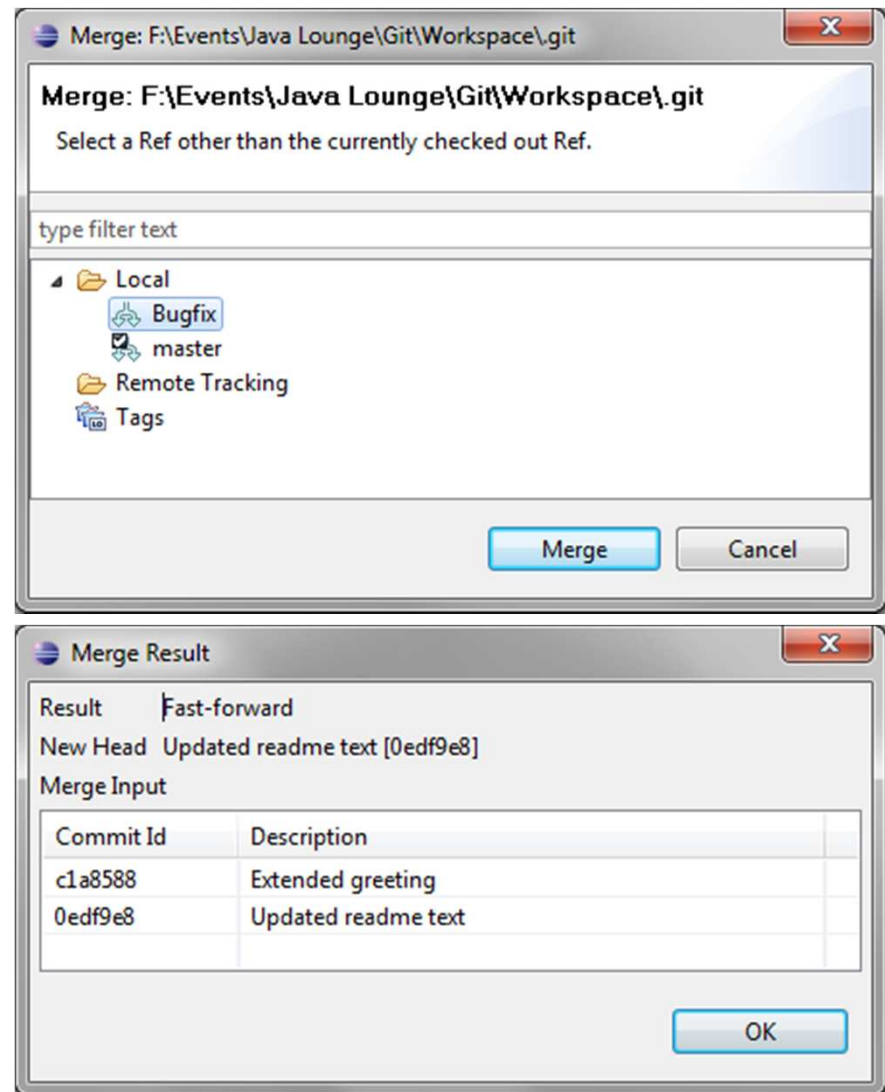
- Switch back to the **master** branch via **Team** → **Branch**
- Readme.txt is still the old one



A typical day with Git (8)



- Open the Merge dialog via **Team → Merge**
- Select the **Bugfix** branch
- Click the **Merge** button
- Confirm the **Merge Result** dialog
- The file(s) get merged and committed right away



A typical day with Git (9)



Project: Hello World [Workspace]

Branch	Author	Date	Id	Committer
Bugfix	Dominik Schadow <dominik.schadow@trivadis.com>	2011-03-01 15:20:21	0edf9e8d...	Dominik Schadow <dominik.sc
Test	Dominik Schadow <dominik.schadow@trivadis.com>	2011-03-01 15:14:57	c1a85887...	Dominik Schadow <dominik.s
HEAD	Dominik Schadow <dominik.schadow@trivadis.com>	2011-03-01 15:10:09	8b1eaf41...	Dominik Schadow <dominik.sc

commit c1a85887c6c416e1f74e98357ee64d3625dfe782
Author: Dominik Schadow <dominik.schadow@trivadis.com> 2011-03-01 15:14:57
Committer: Dominik Schadow <dominik.schadow@trivadis.com> 2011-03-01 15:14:57
Parent: 8b1eaf411c8eb4cd03f8497434dd0cf726112d37 (Initial commit for Hello World sample project)
Child: 0edf9e8d90f97775def4c41479a2326f592fd1b3 (Updated readme text)
Branches: Test, Bugfix, master

Extended greeting

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

Agenda



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

Git IDE integration



- Eclipse
 - Useable version available, supports most Git commands
 - Full featured version with Eclipse 3.7 in June 2011
- IntelliJ IDEA
 - Stable version available, supports subset of Git commands
- JDeveloper
 - Not available
- NetBeans
 - Full featured version with NetBeans 7.0 in April 2011

Git Pros and Cons



- ✓ Fast – extremely high performance even in large projects
- ✓ Offline mode – no server connection required
- ✓ Rapid branching and merging – merging is done all the time
- ✓ Fully distributed – no central server required
- ✓ Supports creativity – just hack something in a new branch

✗ Usage concept – (completely) different from CVS/ SVN

- ✗ IDE integration – still in an early stage
- ✗ Version numbers – a GUID is required for distributed versioning

And the winner is...



Git

More information



- Git <http://git-scm.com>
- ProGit <http://progit.org>
- Gerrit Code Review <http://code.google.com/p/gerrit/>
- GitHub www.github.com
- Eclipse JGit www.eclipse.org/jgit
- Eclipse EGit www.eclipse.org/egit
- NetBeans Git http://netbeans.org/projects/versioncontrol/pages/Git_main
- 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!



Basel

Bern

Lausanne

Zurich

Düsseldorf

Frankfurt/M.

Freiburg i. Br.

Hamburg

Munich

Stuttgart

Vienna