Git Extensions Documentation

Release 3.00.00.4433

Contributors

Dec 23, 2018

Contents

1	Git E	Extensions	1
	1.1		1
	1.2		1
	1.3	Links	2
2	Getti 2.1	ing Started Installation	3 3
	2.1	Portable	7
	2.2	Settings	7
	2.4	Dashboard	8
	2.5	Create new repository	9
	2.6	Open repository	9
	2.7	Clone repository	9
	2.8		10
	2.0		. 0
3	Setti	ngs 1	12
	3.1	Git Extensions	13
			14
		11	15
			16
			17
			17
			20
		1	21
			22
			23
			23
			24
			25
		1 5	26
			26
			27
			27
	3.2	Git 2	28
			28
			29
		3.2.3 Advanced	80

	3.3	Plugins	30			
4	4 Browse Repository 31					
	4.1 4.2	View revision graph	31 33 33 34 34			
5	File h 5.1 5.2 5.3 5.4	Commit	37 38 38 39 40			
6	Com 6.1	Commit changes 6.1.1 Staging changes 6.1.2 6.1.2 Staging selected lines 6.1.3 0.1.3 Undoing or resetting changes 6.1.4	41 43 43 44 45 47			
7	Stash 7.1		48 49			
8	Tag 8.1 8.2	Create tag	50 50 51			
9	Bran 9.1 9.2 9.3 9.4 9.5 9.6	Create branch 9.1.1 Orphan branches 9.1.1 Orphan branches 9.1.1 Orphan branches 9.1.1 Checkout branch 9.1.1 9.1.1 9.1.1 Merge branches 9.1.1 9.1.1 9.1.1 Merge branches 9.1.1 9.1.1 9.1.1 Merge branches 9.1.1 9.1.1 9.1.1 Interactive rebase 9.1.1 9.1.1 9.1.1	53 54 55 55 55 58 59 60			
10		Create patch	61 62 62			
11	11.4	Manage remote repositoriesGit Credential ManagerCreate SSH key11.3.1PuTTY and github11.3.2OpenSSH and githubPull changes	64 66 66 66 68 69 71			
12			73 73			

10	Modify Git history 13.1 Cherry pick commit	77 77
	13.2 Revert commit	79
	13.3 Modify the last commit	81
	13.4 Modify an older commit	81
	13.4.1 Interactive rebase	82
	13.4.2 Using autosquash rebase feature	83
14	Notes	86
15	Submodules	88
	15.1 Manage submodules	88 89
16	Worktrees	90
17	Maintenance	91
	17.1 Compress Git database	91
	17.2 Recover lost objects	92
	17.3 Fix user names	94 95
10	17.4 Ignore files	
18	Franslations	96
	18.1 Change language 18.2 Translate Git Extensions	96 97
19	Windows Explorer	98
20	Visual Studio	100
		100
	20.1 Menu	
	20.1 Menu	100
		100 101
21	20.2 Toolbar Toolbar 20.3 Context menu Context menu	100 101 102 103
21	20.2 Toolbar Context menu 20.3 Context menu Command line	100 101 102
	20.2 Toolbar Context menu 20.3 Context menu Command line Command line Command line 21.1 Git Extensions command line Command line Appendix Command line	100 101 102 103 103 106
	20.2 Toolbar Context menu 20.3 Context menu Command line Command line Command line 21.1 Git Extensions command line Command line Appendix	100 101 102 103 103
22	20.2 Toolbar Context menu 20.3 Context menu Command line Command line Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Command line	100 101 102 103 103 106 107 109
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command Sheet 22.1 Git Cheat Sheet Command Sheet Plugins Compile SubModules	100 101 102 103 103 106 107 109 109
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command Structure 22.1 Git Cheat Sheet Command line Plugins Command Structure 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line	1000 1011 1022 103 103 106 107 109 109
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line 22.1 Git Cheat Sheet Command line Plugins Compile SubModules 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Compile SubModules 23.3 Create local tracking branches Command line	1000 1011 1022 103 103 106 107 109 109 109 110
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Compile SubModules 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line	1000 1011 1022 103 103 106 107 109 109
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Compile SubModules 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line 23.5 Find large files Command line	1000 1011 1022 1033 103 1066 1077 1099 1099 1100 1100
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Compile SubModules 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line 23.5 Find large files Command line	1000 1011 1022 1033 1033 1066 1077 1099 1099 1099 1100 1100 1100
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Command line 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line 23.5 Find large files Command line 23.6 Gerrit Code Review Command line	1000 1011 1022 1033 1033 1066 1077 1099 1099 1099 1100 1100 1100 1100
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Command line 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line 23.5 Find large files Code Review 23.6 Gerrit Code Review Code Review 23.7 GitHub Code Review 23.8 GitFlow Code Review 23.9 Gource Code Review	1000 1011 1022 1033 1033 1066 1077 1099 1099 1100 1100 1100 1100 1100
22	20.2 Toolbar Context menu 20.3 Context menu Command line 21.1 Git Extensions command line Command line Appendix Command line 22.1 Git Cheat Sheet Command line Plugins Context menu 23.1 Auto compile SubModules Command line 23.2 Bitbucket Server Command line 23.3 Create local tracking branches Command line 23.4 Delete obsolete branches Command line 23.5 Find large files Command line 23.6 Gerrit Code Review Code Review 23.7 GitHub Code Review 23.8 GitFlow Code Review 23.9 Gource Code Reple 23.10 Impact Graph Code Reple	1000 1011 1022 1033 1033 1066 1077 1099 1099 1099 1100 1100 1100 1100
22	20.2 Toolbar	1000 1011 1022 1033 1033 1033 1060 107 1099 1099 1099 1099 1100 1100 1100
22	20.2 Toolbar	100 101 102 103 103 103 106 107 109 109 109 109 110 110 110 110 110 111 111
22	20.2 Toolbar	1000 1011 1022 1033 1033 1033 1060 107 1099 1099 1099 1099 1100 1100 1100

CHAPTER 1

Git Extensions

Git Extensions is a toolkit aimed at making working with Git under Windows more intuitive. The shell extension will integrate in Windows Explorer and presents a context menu on files and directories. There is also a Visual Studio extension to use Git from the Visual Studio IDE.

1.1 Features

- Windows Explorer integration for Git
- · Feature rich user interface for Git
- 32bit and 64bit support
- Visual Studio extension (2015-2017)

Specific in 2.5x releases:

- Visual Studio (2010 2015) add-in
- Runs under Linux or Mac OS X using Mono
- Basic SVN functionality

For description of 2.x specific features, see the 2.x documentation

1.2 Video tutorials

There are video tutorials for some basic functions on YouTube, for older Git Extensions versions.

- 1. Clone
- 2. Commit changes
- 3. Push changes
- 4. Pull changes

- 5. Handle merge conflicts
- 6. Install Git Extensions on Ubuntu 11.04

1.3 Links

See the following links for the Git Extensions download page, source code and documentation.

- Download page: https://sourceforge.net/projects/gitextensions/
- Source Code: https://github.com/gitextensions/gitextensions
- Source Code Issue tracker: https://github.com/gitextensions/gitextensions/issues
- Documentation: https://github.com/gitextensions/GitExtensionsDoc
- Documentation Issue tracker: https://github.com/gitextensions/GitExtensionsDoc/issues
- Wiki: https://github.com/gitextensions/gitextensions/wiki

Please feel free to raise any issues with Git Extensions or its documentation at the appropriate Issue tracker link as shown above.

CHAPTER 2

Getting Started

2.1 Installation

This section only covers Git Extensions installation, you will need to have Git for Windows installed: https://git-scm. com/download/win

The Git Extensions installer can be found on GitHub.

😥 Git Extensions 3.00.00.4409 Setup	_		×
Ready to install Git Extensions 3.00.00.4409		2	50
Click Install to begin the installation. Click Back to review or o your installation settings. Click Cancel to exit the wizard.	change	any of	
Back	tall	Cano	el

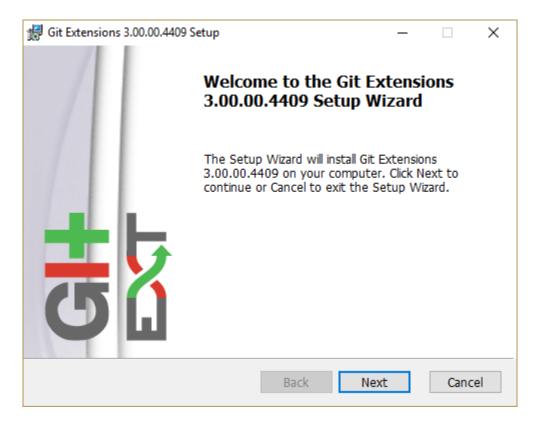


Fig. 1: Start installation.

🖟 Git Extensions 3.00.00.4409 Setup	-		×		
Installation Scope			L P		
Choose the installation scope and folder		C	5 ()		
 Install just for you (ejgo) Git Extensions 3.00.00.4409 will be installed in a per-user folder and be available just for your user account. You do not need local Administrator privileges. Install for all users of this machine Git Extensions 3.00.00.4409 will be installed in a per-machine folder by 					
default and be available for all users. You can change the d installation folder. You must have local Administrator priviled	lefault				
Back Next		Can	cel		

Fig. 2: Installation scope.

🕼 Git Extensions 3.00.00.4409 Setup	_		×
Destination Folder Click Next to install to the default folder or click Change	to choos	2	50
Install Git Extensions 3.00.00.4409 to: C:\Program Files (x86)\GitExtensions\			
Change			
Back	Next	Cano	cel

Fig. 3: Destination folder.

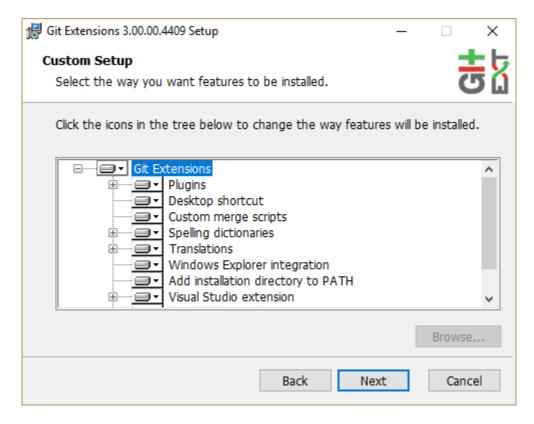


Fig. 4: Choose the options to install.

🖟 Git Extensions 3.00.00.4409 Setup	_	
Select SSH Client		± Ե
Select the SSH client that will be used by Git Extensions		50
○ OpenSSH		
OpenSSH is the Git default.		
PuTTY (plink.exe)		
PuTTY has better integration with Windows		
Back Next	t	Cancel

Fig. 5: Choose the SSH client to use. PuTTY is the default because it has better Windows integration, but Pageant must be running.

2.2 Portable

Git Extensions is also distributed as a portable .zip file, that only require unpacking. Some features like Windows shell integration and Visual Studio integration is not available with this package.

2.3 Settings

Git must be installed prior to starting Git Extensions: .. image:: /images/install/git_missing.png

First selection is language (depends on the installed languages): .. image:: /images/install/language.png

All settings will be verified when Git Extensions is started for the first time. If Git Extensions requires any settings to be changed, the Settings dialog will be shown. All incorrect settings will be marked in red (for instance if the Git version is unsupported) and orange for not recommended setting (like that Git version is older than recommended). You can ask Git Extensions to try to fix the setting for you by clicking on it. When installing Git Extensions for the first time, you will normally be required to configure your username and email address.

The settings dialog can be invoked at any time by selecting Settings from the Tools menu option.

🗙 Settings - Checklist		×			
Type to find	Settings source: Global for all repositories 				
✓ X Git Extensions					
Appearance	The checklist below validates the basic settings needed for GitExtensions to work properly.				
Aa Fonts	Git 2.19.1 is found on your computer.				
Build server integration Scripts	A username and an email address are configured.				
	There is a custom mergetool configured: p4merge				
✓ III Advanced Confirmations	There is a difftool configured: tmerge				
 Detailed Browse repository with 	Shell extensions registered properly. Linux tools (sh) found on your computer.				
Commit dialog	GitExtensions is properly registered.				
SSH ✓ ♠ Git	Default SSH client, OpenSSH, will be used. (commandline window will appear on pull, push and clone operations)				
Paths	The configured language is English.				
Advanced	Check settings at startup (disables automatically if all settings are correct)				
	Save and i	rescan			
Bitbucket Server					
GitHub					
Gource Jira Commit Hint					
Periodic background fet					
Statistics					
< >>	Changes on the selected page will be saved instantly. OK Cancel	Apply			
	Therefore the Cancel button does NOT revert any changes made.	Арріу			

For further information see Settings.

2.4 Dashboard

The dashboard contains the most common tasks, recently opened repositories and favourites. Favourite repositories can be added, grouped under Category headings in the right panel.

🗙 Git Extensions				-	\times
Start Dashboard Tools Help					
	Recent Repositories				
Create new repository Open repository Clone repository Clone GitHub repository	Recent repositories F:\de\\gc\gitextensions master F:\de\\gc\gitextensions_4 spdr870_feature5653 ge F:\de\\gc\gitextensions_2.51 release/2.51 other F:\de\\gc f:\de\\gc	F:\dev\gc\roslyn F:\dev\gc\gitextensions F:\dev\gc\roslyn F:\dev\gc\roslyn F:\dev\gc\roslyn F:\dev\gc\roslyn F:\dev\gc\roslyn	F:\dev\gc\gitextensions_2.51 release/2.51 F:\dev\gc\gitextensions_4 spdr870/_feature5853	· · · ·	
Contribute ↔ Develop © Donate 漢 Translate ∦ Issues					

Recent Repositories can be moved to favourites using the repository context menu. Choose Categories / Add new to create a new category and add the repository to it, or you can add the repository to an existing category (e.g. 'Currents' as shown below).

Recent Repo	ositories		
Recent repositorie		ktens	ions
spdr870/	Show in folder		
Ev da v a	Categories •		(none)
F:\dev\g release/2	Remove project from the list		ge
			other
ge		0	Add new
release/2.51	textensions_2.51 F:\dev\gc\gite master	extens	ions

To open an existing repository, simply click the link to the repository, or select Open repository (from where you can select a repository to open from your local file system).

To create a new repository, one of the following options under Common Actions can be selected.

2.5 Create new repository

When you do not want to work on an existing project, you can create your own repository using this option.

X Create new repository				
Directory		Browse		
Repository typ	e			
Personal re	pository			
 Central repository, no working directory (bareshared=all) 		Create		

Select a directory where the repository is to be created. You can choose to create a Personal repository or a Central repository.

A personal repository looks the same as a normal working directory but has a directory named .git at the root level containing the version history. This is the most common repository.

Central repositories only contain the version history. Because a central repository has no working directory you cannot checkout a revision in a central repository. It is also impossible to merge or pull changes in a central repository. This repository type can be used as a public repository where developers can push changes to or pull changes from.

2.6 Open repository

Opens a Git repo already existing on the file system.

🗙 Open local	repository			×
Directory 0	F:\dev\gc\gitextension F:\dev\gc\gitextensions F:\dev\gc\gitextensions_2.51 F:\dev\gc\gitextensions_4 F:\dev\gc\gitextensions_9	✓ ◎ □	Browse Open	
F:\dev\gc master	F:\dev\gc\gitextensions_jbialobr F:\dev\gc\gitextensions_wt2 F:\dev\gc\gitextensionsxxx			

2.7 Clone repository

You can clone an existing repository using this option.

🗙 Clone			×				
Repository to clone:	https://github.com/gerhardol/NBug.git	~	Browse				
Destination:	F:\dev\gc\gitextensions_4	~	Browse				
Subdirectory to create:	NBug						
Branch:	(default: remote HEAD)	~					
	loned to a new directory located here: s_4\NBug (New directory)						
Repository type Personal repository	,						
O Public repository, no working directory (bare)							
✓ Initialize all submod	dules 🖸 Download full history 🗌 Use LFS extension						
P Load SSH key			Clone				

The repository you want to clone could be on a network share or could be a repository that is accessed through an internet or intranet connection. Depending on the protocol (http or ssh) you might need to load a SSH key into PuTTY. You also need to specify where the cloned repository will be created and the initial branch that is checked out. If the cloned repository contains submodules, then these can be initialised using their default settings if required.

There are two different types of repositories you can create when making a clone. A personal repository contains the complete history and also contains a working copy of the source tree. A central (bare) repository is used as a public repository where developers push the changes they want to share with others to. A central repository contains the complete history but does not have a working directory like personal repositories.

2.8 Clone Github repository

This option allows you to

- 1. Fork a repository on GitHub so it is created in your personal space on GitHub.
- 2. Clone any repositories on your personal space on GitHub so that it becomes a local repository on your machine.

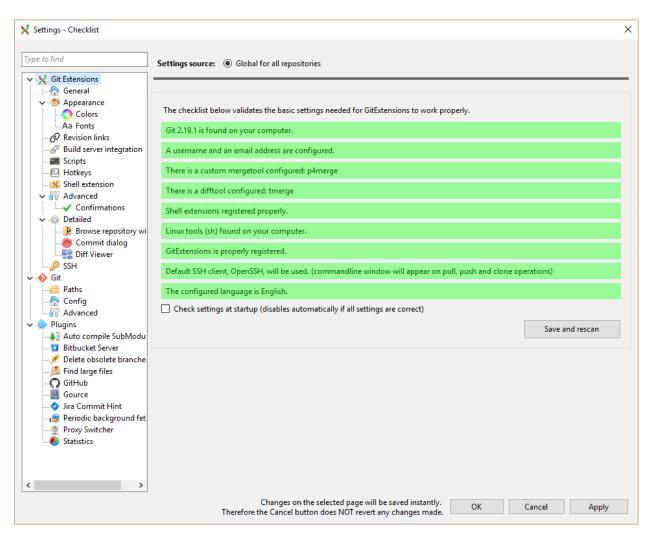
You can see your own personal repositories on GitHub, and also search for repositories using the Search for repositories tab.

🗙 GitHub: Remo	te repository for	k and clo	ne				– 🗆 X
My repositories	Search for repo	sitories					
Name conemu-inside Git.hub gitextensions.gi GitExtensions.Do ICSharpCode.Te libgit2sharp NBug runnerup st2funbeat	thub.io	ls fork Yes Yes No No Yes Yes Yes Yes No No	# Forks 9 15 1108 4 25 4 1 16 187 0	Priv No No No No No No No No No		^	If you want to fork a repository owned by somebody else, go to the Search for repositories tab.
Clone Destination folde F:\dev\gc Create directory: gitextensions	ithub.com:gitext	Add	remote a	s:	Browse nto F:\dev\gc\gitextensions.	•	Clone Close

chapter $\mathbf{3}$

Settings

The settings dialog can be invoked at any time by selecting Settings from the Tools menu option.



The following buttons are always available on any page of the Settings dialog. Sometimes the Cancel button has no effect for the page - this will be noted on the page in the area next to the buttons.

Button	Description
OK	Save any entered changes made in <i>any</i> settings page and close the Settings dialog.
Cancel	Any entered changes in <i>any</i> settings page are <i>not</i> saved. The Settings dialog is closed.
Apply	Any entered changes in <i>any</i> settings page are saved.

Settings that are specific to Git Extensions and apply globally will be stored in a file called GitExtensions. settings either in the user's application data path or with the program. The location is dependent on the Is-Portable setting in the GitExtensions.exe.config file that is with the program. Settings that are specific to Git Extensions but apply to only the current repository will be stored in a file of the same name, GitExtensions. settings, but in either the root folder of the repository or the .git folder of the repository, depending on whether or not they are distributed with that repository.

3.1 Git Extensions

This page is a visual overview of the minimal settings that Git Extensions requires to work properly. Any items highlighted in red should be configured by clicking on the highlighted item.

This page contains the following settings and buttons.

Check settings at startup

Forces Git Extensions to re-check the minimal set of required settings the next time Git Extensions is started. If all settings are 'green' this will be automatically unchecked.

Save and rescan

Saves any setting changes made and re-checks the settings to see if the minimal requirements are now met.

3.1.1 General

This page contains general settings for Git Extensions.

Performance

Show number of changed files on commit button

When enabled, the number of pending commits are shown on the toolbar as a figure in parentheses on the Commit button. Git Extensions must be stopped and restarted to activate changes to this option. Turn this off if you experience slowdowns.

Show number of changed files for artificial commits

If artificial commits are enabled in the revision graph, show the pending commits as well as a tool tip with a summary of changes.

Show current working directory changes as an artificial commit.

When enabled, two artificial revisions are added to the revision graph. The first shows the worktree (current working directory) status. The second shows the commit index (staged).

Use FileSystemWatcher to check if index is changed

Monitor if the Git index is changed due to changes outside of Git Extensions, if so show this in the Refresh button in Browse.

Show stash count on status bar in browse window

When you use the stash a lot, it can be useful to show the number of stashed items on the toolbar. This option causes serious slowdowns in large repositories and is turned off by default.

Show submodule status in browse window

Show the status for submodules (as well as supermodules) in the dropdown menu in Browse. The status is updated if *Show number of changed files for artificial commits* is enabled and the number of artificial commits is updated. (Changes in supermodules is not monitored). This option causes serious slowdowns in large repositories and is turned off by default.

Check for uncommitted changes in checkout branch dialog

Git Extensions will not allow you to checkout a branch if you have uncommitted changes on the current branch. If you select this option, Git Extensions will display a dialog where you can decide what to do with uncommitted changes before swapping branches.

Limit number of commits that will be loaded at start-up

This number specifies the maximum number of commits that Git Extensions will load when it is started. These commits are shown in the Revision Graph window. To see more commits, then this setting will need to be adjusted and Git Extensions restarted.

Behaviour

Close Process dialog when process succeeds

When a process is finished, close the process dialog automatically. Leave this option off if you want to see the result of processes. When a process has failed, the dialog will automatically remain open.

Show console window when executing git process

Git Extensions uses command line tools to access the git repository. In some environments it might be useful to see the command line dialog when a process is executed. An option on the command line dialog window displayed allows this setting to be turned off.

Use patience diff algorithm

Use the Git 'patience diff' algorithm instead of the default. This algorithm is useful in situations where two files have diverged significantly and the default algorithm may become 'misaligned', resulting in a totally unusable conflict file.

Include untracked files in stash

If checked, when a stash is performed as a result of any action except a manual stash request, e.g. checking out a new branch and requesting a stash then any files not tracked by git will also be saved to the stash.

Follow renames in file history (experimental)

Try to follow file renames in the file history.

Follow exact renames and copies only

Follow file renames and copies for which similarity index is 100%. That is when a file is renamed or copied and is committed with no changes made to its content.

Open last working dir on startup

When starting Git Extensions, open the last used repository (bypassing the Dashboard).

Default clone destination

Git Extensions will pre-fill destination directory input with value of this setting on any form used to perform repository clone.

Revision grid quick search timeout [ms]

The timeout (milliseconds) used for the quick search feature in the revision graph. The quick search will be enabled when you start typing and the revision graph has the focus.

Email settings for sending patches

SMTP server name

SMTP server to use for sending patches.

Port

SMTP port number to use.

Use SSL/TLS

Check this box if the SMTP server uses SSL or TLS.

3.1.2 Appearance

This page contains settings that affect the appearance of the application.

General

Show relative date instead of full date

Show relative date, e.g. 2 weeks ago, instead of full date. Displayed on the commit tab on the main Revision Graph window.

Show current branch in Visual Studio

Determines whether or not the currently checked out branch is displayed on the Git Extensions toolbar within Visual Studio.

Auto scale user interface when high DPI is used

Automatically resize controls and their contents according to the current system resolution of the display, measured in dots per inch (DPI).

Truncate long filenames

This setting affects the display of filenames in a component of a window e.g. in the Diff tab of the Revision Graph window. The options that can be selected are:

- None no truncation occurs; a horizontal scroll bar is used to see the whole filename.
- Compact no horizontal scroll bar. Filenames are truncated at both start and end to fit into the width of the display component.
- Trimstart no horizontal scroll bar. Filenames are truncated at the start only.
- FileNameOnly the path is always removed, leaving only the name of the file, even if there is space for the path.

Author images

Get author image from gravatar.com

If checked, gravatar will be accessed to retrieve an image for the author of commits. This image is displayed on the commit tab on the main Revision Graph window.

Image size

The display size of the user image.

Cache images

The number of days to elapse before gravatar is checked for any changes to an authors image.

No image service

If the author has not set up their own image, then gravatar can return an image based on one of these services.

Clear image cache

Clear the cached avatars.

Language

Language (restart required)

Choose the language for the Git Extensions interface.

Dictionary for spelling checker

Choose the dictionary to use for the spelling checker in the Commit dialog.

3.1.2.1 Colors

This page contains settings to define the colors used in the application.

Revision graph

Multicolor branches

Displays branch commits in different colors if checked. If unchecked, all branches are shown in the same color. This color can be selected.

Striped branch change

(Unused) .. todo Any effect?

Draw alternate background

Alternate background colour for revision rows.

Draw non relatives graph gray

Show commit history in gray for branches not related to the current branch.

Draw non relatives text gray

Show commit text in gray for branches not related to the current branch.

Color tag

Color to show tags in.

Color branch

Color to show branch names in.

Color remote branch

Color to show remote branch names in.

Color other label

Color to show other labels in.

Difference View

Color removed line Highlight color for lines that have been removed.

Color added line

Highlight color for lines that have been added.

Color removed line highlighting

Highlight color for characters that have been removed in lines.

Color added line highlighting

Highlight color for characters that have been added in lines.

Color section

Highlight color for a section.

3.1.2.2 Fonts

Fonts

Code font

The font used for the display of file contents.

Application font

The font used on Git Extensions windows and dialogs.

Commit font

The font used for entering a commit message in the Commit dialog.

Monospace font

The font used for the commit id in the revision graph.

3.1.3 Revision Links

You can configure here how to convert parts of a revision data into clickable links. These links will be located under the commit message on the Commit tab in the Related links section.

	spdr870/_feature/5853			> spdr870/feature/5853 #5		
/		//	Haal - Is.		1.111	
🍥 Commit	😫 Diff	😤 File tree	🔑 GPG	Console		
#5853: Use	Author: Date: Commit Child: Parent:	3 days a	go (2018-12 939be7ddal <u>4a3</u> 707	nk.westhuis@u -05 13:41:22) b2da5b7d21c26 in EnsureSco	a043e0792d	

Related links: <u>View on GitHub, Issue 5853</u> Contained in branches: <u>spdr870/ feature/5853</u> Contained in tags: <u>v3.00.00</u>

Derives from tag: v3.00.00-rc2 + 53 commits

The most common case is to convert an issue number given as a part of commit message into a link to the coresponding issue-tracker page. The screenshot below shows an example configuration for GitHub issues.

X Settings - Revision links			\times
Type to find You will be a constructed of the second of the sec	Settings source: (e) Effective << (c) Local for cur Categories GitHub - commit GitHub - issues GitHub - PR gh feature	urrent repository << O Distributed with current repository << O Global for all repositories <table> Name GitHub - issues Enabled Sentence Remote data Use remotes upstream origin gitextensions Only use the first match Search in URL Push URL Search pattern github.com[/](.+)\.git Revision data Search in Message Local branch name Remote branch name Search pattern [?](?<1pull request [pr[_]?)#?\d+ Nested pattern (d+)+ Links Caption URI Links</table>	
	Add Remove	OK Cancel Apply	

Categories

Lists all the currently defined Categories. Click the Add button to add a new empty Category. The default name is 'new'. To remove a Category select it and click the Remove button.

Name

This is the Category name used to match the same categories defined on different levels of the Settings.

Enabled

Indicates whether the Category is enabled or not. Disabled categories are skipped while creating links.

Remote data

It is possible to use data from remote's URL to build a link. This way, links can be defined globally for all repositories sharing the same URL schema.

Use remotes

Regex to filter which remotes to use. Leave blank to create links not depending on remotes. If full names of remotes are given then matching remotes are sorted by its position in the given Regex.

Only use the first match

Check if you want to create links only for the first matching remote.

Search in

Define whether to search in URL, Push URL or both.

Revision data

Search in

Define which parts of the revision should be searched for matches.

id search-pattern

Search pattern

Regular expression used for matching text in the chosen revision parts. Each matched fragment will be used to create a new link. More than one fragment can be used in a single link by using a capturing group. Matches from the Remote data group go before matches from the Revision data group. A capturing group value can be passed to a link by using zero-based indexed placeholders in a link format definition e.g. {0}.

Nested pattern

Nested pattern can be used when only a part of the text matched by the *Search pattern* should be used to format a link. When the Nested pattern is empty, matches found by the *Search pattern* are used to create links.

Links: Caption/URI

List of links to be created from a single match. Each link consists of the Caption to be displayed and the URI to be opened when the link is clicked on. In addition to the standard zero-based indexed placeholders, the %COMMIT_HASH% placeholder can be used to put the commit's hash into the link. For example: https://github.com/gitextensions/gitextensions/commit/%COMMIT_HASH%

3.1.4 Build server integration

This page allows you to configure the integration with build servers. This allows the build status of each commit to be displayed directly in the revision log, as well as providing a tab for direct access to the Build Server build report for the selected commit.

Enable build server integration

Check to globally enable/disable the integration functionality.

Build server type

Select an integration target.

AppVeyor

Account name

AppVeyor account name. You don't have to enter it if the projects you want to query for build status are public.

API token

AppVeyor API token. Requiered if the Account name is entered. See https://ci.appveyor.com/api-token

Project(s) name(s)

Projects names separated with 'l', e.g. gitextensions/gitextensions/jbialobr/gitextensions

Display tests results in build status summary for every build result Include tests results in the build status summary for every build result.

Azure DevOps and Team Foundation Server (since TFS2015)

Project URL

Enter the URL of the server (and port, if applicable).

Build definition name

Limit the builds if desired.

Rest API token

Read token for the build server.

Jenkins

Jenkins server URL

Enter the URL of the server (and port, if applicable).

Project name

Enter the name of the project which tracks this repository in Jenkins. Separate project names with "|". Multi-branch pipeline projects are supported by adding "?m" to the project name.

Ignore build for branch

The plugin will normally display the last build for a certain commit. If Jenkins starts several builds for one commit, it is possible to ignore the non interesting builds if all builds are not interesting.

TeamCity

TeamCity server URL

Enter the URL of the server (and port, if applicable).

Project name

Enter the name of the project which tracks this repository in TeamCity. Multiple project names can be entered separated by the | character.

Build Id Filter

Enter a regexp filter for which build results you want to retrieve in the case that your build project creates multiple builds. For example, if your project includes both devBuild and docBuild you may wish to apply a filter of "devBuild" to retrieve the results from only the program build.

Team Foundation Server

For TFS prior to 2015.

Tfs server (Name or URL)

Enter the URL of the server (and port, if applicable).

Team collection name

Project name

Enter the name of the project which tracks this repository in Tfs.

Build definition name

Use first found if left empty.

3.1.5 Scripts

This page allows you to configure specific commands to run before/after Git actions or to add a new command to the User Menu. The top half of the page summarises all of the scripts currently defined. If a script is selected from the summary, the bottom half of the page will allow modifications to the script definition.

A hotkey can also be assigned to execute a specific script. See Hotkeys.

Add

Adds a new script. Complete the details in the bottom half of the screen.

Remove

Removes a script.

Up/Down Arrows

Changes order of scripts.

Name

The name of the script.

Enabled

If checked, the script is active and will be performed at the appropriate time (as determined by the On Event setting).

Ask for confirmation

If checked, then a popup window is displayed just before the script is run to confirm whether or not the script is to be run. Note that this popup is *not* displayed when the script is added as a command to the User Menu (On Event setting is ShowInUserMenuBar).

Run in background

If checked, the script will run in the background and Git Extensions will return to your control without waiting for the script to finish.

Add to revision grid context menu

If checked, the script is added to the context menu that is displayed when right-clicking on a line in the Revision Graph page.

Is PowerShell

If checked, the command is started through a powershell.exe process. If the *Run in background* is checked, the powershell console is closed after finishing. If not, the powershell console is left for the user to close it manually.

Command

Enter the command to be run. This can be any command that your system can run e.g. an executable program, a .bat script, a Python command, etc. Use the Browse button to find the command to run.

There are some special prefixes which change the way the script is executed:

- plugin:<plugin-name>: Where <plugin-name> is the name of a *plugin* (refer *Plugins*). If a plugin with that name is found, it is run.
- navigateTo:<script-path>: Where <script-path> is the path to a file containing the script to run. That script is expected to return a commit hash as the first line of its output. The UI will navigate to that commit once the script completes.

Arguments

Enter any arguments to be passed to the command that is run. The Help button displays items that will be resolved by Git Extensions before executing the command e.g. {cBranch} will resolve to the currently checked out branch, {UserInput} will display a popup where you can enter data to be passed to the command when it is run.

On Event

Select when this command will be executed, either before/after certain Git commands, or displayed on the User Menu bar.

Icon

Select an icon to be displayed in a menu item when the script is marked to be shown in the user menu bar.

3.1.6 Hotkeys

This page allows you to define keyboard shortcuts to actions when specific pages of Git Extensions are displayed. The HotKeyable Items identifies a page within Git Extensions. Selecting a Hotkeyable Item displays the list of commands on that page that can have a hotkey associated with them.

The Hotkeyable Items consist of the following pages

- 1. Commit: the page displayed when a Commit is requested via the Commit User Menu button or the Commands/ Commit menu option.
- 2. Browse: the Revision Graph page (the page displayed after a repository is selected from the dashboard (Start Page)).
- 3. RevisionGrid: the list of commits in Browse and other forms.
- 4. FileViewer: the page displayed when viewing the contents of a file.
- 5. FormMergeConflicts: the page displayed when merge conflicts are detected that need correcting.
- 6. BrowseDiff: Diff tab in Browse.
- 7. RevisionFileTree: The FileTree tab in Browse.
- 8. Scripts: shows scripts defined in Git Extensions and allows shortcuts to be assigned. Refer Scripts.

Hotkey

After selecting a Hotkeyable Item and the Command, the current keyboard shortcut associated with the command is displayed here. To alter this shortcut, click in the box where the current hotkey is shown and press the new keyboard combination.

Apply

Click to apply the new keyboard combination to the currently selected Command.

Clear

Sets the keyboard shortcut for the currently selected Command to 'None'.

Reset all Hotkeys to defaults

Resets all keyboard shortcuts to the defaults (i.e. the values when Git Extensions was first installed).

3.1.7 Shell Extension

When installed, Git Extensions adds items to the context menu when a file/folder is right-clicked within Windows Explorer. One of these items is Git Extensions from which a further (cascaded) menu can be opened. This settings page determines which items will appear on that cascaded menu and which will appear in the main context menu. Items that are checked will appear in the cascaded menu.

To the right side of the list of check boxes is a preview that shows you how the Git Extensions menu items will be arranged with your current choices.

By default, what is displayed in the context menu also depends on what item is right-clicked in Windows Explorer; a file or a folder (and whether the folder is a Git repository or not). If you want Git Extensions to always include all of its context menu items, check the box Always show all commands.

3.1.8 Advanced

This page allows advanced settings to be modified. Refer Confirmations.

Checkout

Always show checkout dialog

Always show the Checkout Branch dialog when swapping branches. This dialog is normally only shown when uncommitted changes exist on the current branch

Use last chosen "local changes" action as default action.

This setting works in conjunction with the 'Git Extensions/Check for uncommitted changes in checkout branch dialog' setting. If the 'Check for uncommitted changes' setting is checked, then the Checkout

Branch dialog is shown only if this setting is unchecked. If this setting is checked, then no dialog is shown and the last chosen action is used.

General

Don't show help images

In the Pull, Merge and Rebase dialogs, images are displayed by default to explain what happens with the branches and their commits and the meaning of LOCAL, BASE and REMOTE (for resolving merge conflicts) in different merge or rebase scenarios. If checked, these Help images will not be displayed.

Always show advanced options

In the Push, Merge and Rebase dialogs, advanced options are hidden by default and shown only after you click a link or checkbox. If this setting is checked then these options are always shown on those dialogs.

Use Console Emulator for console output in command dialogs

Using Console Emulator for console output in command dialogs may be useful the running command requires an user input, e.g. push, pull using ssh, confirming gc.

Auto normalise branch name

Controls whether branch name should be automatically normalised as per git branch naming rules. If enabled, any illegal symbols will be replaced with the replacement symbol of your choice.

Commit

Push forced with lease when Commit & Push action is performed with Amend option checke In the Commit dialog, users can commit and push changes with one click. However, if changes are meant to amend an already pushed commit, a standard push action will be rejected by the remote server. If this option is enabled, a push action with --force-with-lease switch will be performed instead. The --force-with-lease switch will be added only when the Amend option is checked.

Updates

Check for updates weekly

Check for newer version every week.

Check for release candidate versions

Include release candidate versions when checking for a newer version.

3.1.8.1 Confirmations

This page allows you to turn off certain confirmation popup windows.

Don't ask to confirm to

Amend last commit

If checked, do not display the popup warning about the rewriting of history when you have elected to amend the last committed change.

Commit when no branch is currently checked out

When committing changes and there is no branch currently being checked out, then GitExtensions warns you and proposes to checkout or create a branch. Enable this option to continue working with no warning.

Apply stashed changes after successful pull

In the Pull dialog, if Auto stash is checked, then any changes will be stashed before the pull is performed. Any stashed changes are then re-applied after the pull is complete. If this setting is checked, the stashed changes are applied with no confirmation popup.

Apply stashed changes after successful checkout

In the Checkout Branch dialog, if Stash is checked, then any changes will be stashed before the branch is checked out. If this setting is checked, then the stashed changes will be automatically re-applied after successful checkout of the branch with no confirmation popup.

Drop stash

Popup when dropping a stash.

Add a tracking reference for newly pushed branch

When you push a local branch to a remote and it doesn't have a tracking reference, you are asked to confirm whether you want to add such a reference. If this setting is checked, a tracking reference will always be added if it does not exist.

Push a new branch for the remote

When pushing a new branch that does not exist on the remote repository, a confirmation popup will normally be displayed. If this setting is checked, then the new branch will be pushed with no confirmation popup.

Update submodules on checkout

When you check out a branch from a repository that has submodules, you will be asked to update the submodules. If this setting is checked, the submodules will be updated without asking.

Resolve conflicts

If enabled, then when conflicts are detected GitExtensions will start the Resolve conflicts dialog automatically without any prompt.

Commit changes after conflicts have been resolved

Enable this option to start the Commit dialog automatically after all conflicts have been resolved.

Confirm for the second time to abort a merge

When aborting a merge, rebase or other operation that caused conflicts to be resolved, an user is warned about the consequences of aborting and asked if he/she wants to continue. If the user chooses to continue the aborting operation, then he/she is asked for the second time if he/she is sure that he/she wants to abort. Enable this option to skip this second confirmation.

Rebase on top of selected commit

Rebase context menu command popup in revision graph.

Undo last commit

Browse Command popup.

Fetch and prune all

Browse fetch/prune popup.

3.1.9 Detailed

This page allows detailed settings to be modified.

Push window

Get remote branches directly from the remote

Git caches locally remote data. This data is updated each time a fetch operation is performed. For a better performance GitExtensions uses the locally cached remote data to fill out controls on the Push dialog. Enable this option if you want GitExtensions to use remote data recieved directly from the remote server.

Merge window

Add log messages

If enabled, then in addition to branch names, git will populate the log message with one-line descriptions from at most the given number actual commits that are being merged. See https://git-scm.com/docs/git-merge#git-merge_logltngt

3.1.9.1 Browse repository window

Console emulator

Show the Console tab

Show the Console tab in the Browse Repository window.

Console style

Choose one of the predefined ConEmu schemes. See http://conemu.github.io/en/SettingsColors.html.

Shell to run

Choose one of the predefined terminals.

Font size

Console font size.

Show GPG information

Show tab for GPG information if available.

3.1.9.2 Commit dialog

This page contains settings for the Git Extensions *Commit* dialog. Note that the dialog itself has further options.

Behaviour

Provide auto-completion in commit dialog

Enables auto-completion in commit dialog message box. Auto-completion words are taken from the changed files shown by the commit dialog. For each file type there can be configured a regular expression that decides which words should be considered as candidates for auto-completion. The default regular expressions included with Git Extensions can be found here: https://github.com/gitextensions/gitextensions/blob/master/GitExtensions/AutoCompleteRegexes.txt You can override the default regular expressions by creating an AutoCompleteRegexes.txt file in the Git Extensions installation directory.

Show errors when staging files

If an error occurs when files are staged (in the Commit dialog), then the process dialog showing the results of the git command is shown if this setting is checked.

Ensure the second line of commit message is empty

Enforces the second line of a commit message to be blank.

Compose commit messages in Commit dialog

If this is unchecked, then commit messages cannot be entered in the commit dialog. When the Commit button is clicked, a new editor window is opened where the commit message can be entered.

Number of previous messages in commit dialog

The number of commit messages, from the top of the current branch, that will be made available from the Commit message combo box on the Commit dialog.

Remember 'Amend commit' checkbox on commit form close

Remembers the state of the 'Amend commit' checkbox when the 'Commit dialog' is being closed. The remembered state will be restored on the next 'Commit dialog' creation. The 'Amend commit' checkbox is being unchecked after each commit. So, when the 'Commit dialog' is being closed automatically after

commiting changes, the 'Amend commit' checkbox is going to be unchecked first and its state will be saved after that. Therefore the checked state is remembered only if the 'Commit dialog' is being closed by an user without commiting changes.

Show additional buttons in commit button area

Tick the boxes in this sub-group for any of the additional buttons that you wish to have available below the commit button. These buttons are considered additional to basic functionality and have consequences if you should click them accidentally, including resetting unrecorded work.

3.1.9.3 Diff Viewer

Remember the 'Ignore whitespaces' preference

Remember in the GitExtensions settings the latest chosen value of the 'Ignore whitespaces' preference. Use the remembered value the next time GitExtensions is opened.

Remember the 'Show nonprinting characters' preference

Remember in the GitExtensions settings the latest chosen value of the 'Show nonprinting characters' preference. Use the remembered value the next time GitExtensions is opened.

Remember the 'Show entire file' preference

Remember in the GitExtensions settings the latest chosen value of the 'Show entire file' preference. Use the remembered value the next time GitExtensions is opened.

Remember the 'Number of context lines' preference

Remember in the GitExtensions settings the latest chosen value of the 'Number of context lines' preference. Use the remembered value the next time GitExtensions is opened.

Omit uninteresting changes from combined diff Includes git -cc switch when generating a diff. See https://git-scm.com/docs/git-diff-tree#git-diff-tree=-cc

Open Submodule Diff in separate window

If enabled then double clicking on a submodule in the Diff file list opens a new instance of GitExtensions with the submodule as the selected repository. If disabled, the File history window is opened for the double clicked submodule.

Show file differences for all parents in browse dialog

Enable this option to see diff against each of the revision parents, combined diff including.

Vertical ruler position

Position for ruler. .. todo Any effect?

3.1.10 SSH

This page allows you to configure the SSH client you want Git to use. Git Extensions is optimized for PuTTY. Git Extensions will show command line dialogs if you do not use PuTTY and user input is required (unless you have configured SSH to use authentication with key instead of password). Git Extensions can load SSH keys for PuTTY when needed.

Specify which ssh client to use

PuTTY

Use PuTTY as SSH client.

OpenSSH

Use OpenSSH as SSH client.

Other ssh client Use another SSH client. Enter the path to the SSH client you wish to use.

Configure PuTTY

Path to plink.exe Enter the path to the plink.exe executable.

Path to puttygen Enter the path to the puttygen.exe executable.

Path to pageant

Enter the path to the pageant.exe executable.

Automatically start authentication

If an SSH key has been configured, then when accessing a remote repository the key will automatically be used by the SSH client if this is checked.

3.2 Git

The settings that are used by Git are stored in the configuration files of Git. The global settings are stored in the file called .gitconfig in the user directory. The local settings are stored in the .git\config file of the repository.

3.2.1 Paths

This page contains the settings needed to access git repositories. The repositories will be accessed using external tools. For Windows usually "Git for Windows" is used. Git Extensions will try to configure these settings automatically.

Git

Command used to run git (git.cmd or git.exe)

Needed for Git Extensions to run Git commands. Set the full command used to run git ("Git for Windows"). Use the Browse button to find the executable on your file system. (Cygwin Git may work but is not officially supported.)

Path to Linux tools (sh).

A few Linux tools are used by Git Extensions. When Git for Windows is installed, these tools are located in the bin directory of Git for Windows. Use the Browse button to find the directory on your file system. Leave empty when it is in the path.

Environment

Change HOME

This button opens a dialog where the HOME directory can be changed.

The global configuration file used by git will be put in the HOME directory. On some systems the home directory is not set or is pointed to a network drive. Git Extensions will try to detect the optimal setting for your environment. When there is already a global git configuration file, this location will be used. If you need to relocate the home directory for git, click the Change HOME button to change this setting. Otherwise leave this setting as the default.

3.2.2 Config

This page contains some of the settings of Git that are used by and therefore can be changed from within Git Extensions.

If you change a Git setting from the Git command line using git config then the same change in setting can be seen inside Git Extensions. If you change a Git setting from inside Git Extensions then that change can be seen using git config --get.

Git configuration can be global or local configuration. Global configuration applies to all repositories. Local configuration overrides the global configuration for the current repository.

User name

User name shown in commits and patches.

User email

User email shown in commits and patches.

Editor

Editor that git.exe opens (e.g. for editing commit message). This is not used by Git Extensions, only when you call git.exe from the command line. By default Git will use the built in editor.

Mergetool

Merge tool used to solve merge conflicts. Git Extensions will search for common merge tools on your system.

Path to mergetool

Path to merge tool. Git Extensions will search for common merge tools on your system.

Mergetool command

Command that Git uses to start the merge tool. Git Extensions will try to set this automatically when a merge tool is chosen. This setting can be left empty when Git supports the mergetool (e.g. kdiff3).

Keep backup (.orig) after merge

Check to save the state of the original file before modifying to solve merge conflicts. Refer to Git configuration setting `mergetool.keepBackup`.

Difftool

Diff tool that is used to show differences between source files. Git Extensions will search for common diff tools on your system.

Path to difftool

The path to the diff tool. Git Extensions will search for common diff tools on your system.

DiffTool command

Command that Git uses to start the diff tool. This setting should only be filled in when Git doesn't support the diff tool.

Path to commit template

A path to a file whose contents are used to pre-populate the commit message in the commit dialog.

Line endings

Checkout/commit radio buttons

Choose how git should handle line endings when checking out and checking in files. Refer to https://help.github.com/articles/dealing-with-line-endings/#platform-all

Files content encoding

The default encoding for files content.

3.2.3 Advanced

Various settings for Git.

3.3 Plugins

Plugins provide extra functionality for Git Extensions. Please refer to *Plugins*.

CHAPTER 4

Browse Repository

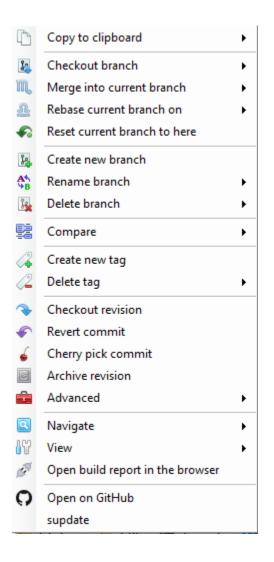
You can browse a repository by starting Git Extensions and select the repository to open. The main window contains the revision graph (commit log). You could also open the 'Browse' window from the shell extensions and from Visual Studio.

4.1 View revision graph

The full commit history can be browsed. There is a graph that shows branches and merges. You can show the difference between any two revisions by selecting them using ctrl-click.

X gitextensions_4 (release	se/3.00) - Git Extensions				_	□ ×
		'lugins Tools Help ▼ 🗣 ▼ 🍙 🥝 Commit (5) 👫 (97) ▼ 🚞	🕎 🌐 Branches:	∽ Ţ • Filter:	₹.	¢
Search: Search: Searches (111) Searches (111) Searches (111) Searches (111) Searches (111) Searches (111) Searches (111) Searches (111)	gitextensions/master Update RE Working directory 2 - 1 Commit index 1		RussKie	2018-12-07 11:54:40	7a9a18c 🔸	3.00.00.44
	Merge pull request #5860 from sp spdr870/feature/guardensuresco spdr870/release/5781 Fix build s 5781: Attempt to fix disappearing	e Added guard for EnsureScorelsAbove. rript scrollbar	RussKie RussKie Henk Westhuis RussKie Henk Westhuis	2018-12-07 11:04:24 2018-12-07 11:05:18 2018-12-07 08:57:23 2018-12-06 11:47:15 2018-12-06 10:43:42	c15bad2 • fae7b02	3.00.00.44 3.00.00.44
> - b mstv bialobr - b drewnoakes - vbjay - b local - b tmp	Filter files Diff with: Merge pull request #5860 from sp	elease GPG Console 🧿 Build Report diffgit a index heafs7	/GitUI/GitUI.csproj b/GitUI/ Bbef51f32c2d 100644 GitUI.csproj	2018-12-05 22:21:04 2018-12-05 22:20:39 'GitUI.csproj	50cbc4c 1ffb097	
> - 🦻 amaiorano > - 🍺 AbhiAgarwal1 > - 🍺 russkie 🗗 qqqqqqqqqq > - 🍺 crypto-rsa	Image: Properties water / GitUl/GitUl.csproj Image: Properties water /					asource",
 NikolayXHD MikolayXHD manaiorano_an monatas monavigate_com misosec monavigate_com test Nikolay 			nclude="Translation\Czech.P] ToOutputDirectory>Preserve№	•	rectory>	
s - 🖡 or		<				

The context menu for a commit can both execute Git commands and change the appearance for the form.



4.2 Search or filter the commit history

You can find text in the commit messages or jump to a specific commit in the current commit history shown in Git Extensions. You can also filter the commit history so that fewer commits are shown.

4.2.1 Quick search in history

You can find a commit in the commit history that is shown in Git Extensions by searching for text in the commit message, branch label or tag. This is a quick search function. Simply click into the commit history to give that pane focus and start typing. Git Extensions will show your search term in the top left corner and will immediately jump to the next commit with matching text. You can search for the next or previous commit with matching text using Alt-Down Arrow or Alt-Up Arrow.

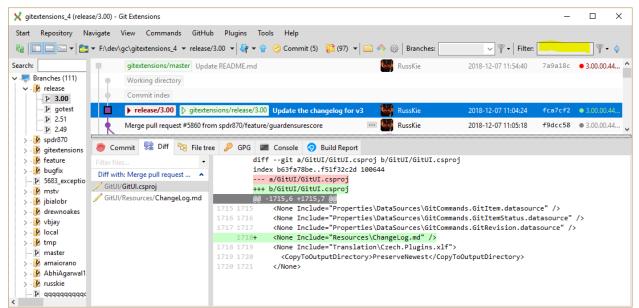
In Settings, Git Extensions you can change the timeout for typing the text for the quick search.

4.2.2 Go to a specific commit

You can jump to a particular commit in the commit history if you know the SHA, tag or branch. In fact you can use any expression valid for git-rev-parse. Select Navigate, Go to commit or press Ctrl-Shift-G to open the Go to commit window. Enter an SHA or other term to be passed to git-rev-parse into the box at the top and click Go, or select a branch or tag from one of the two combo boxes below.

4.2.3 Filter history

The history can be filtered using regular expressions and basic filter terms. Filtering will reduce the number of commits that are shown in the Git Extensions commit history. The quick filter in the toolbar filters by the commit message, the author and/or the committer.



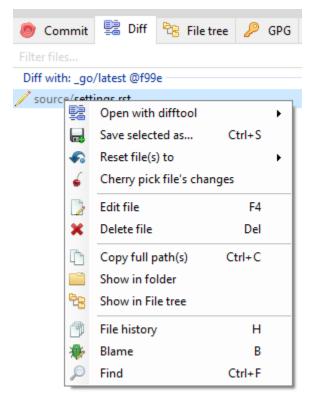
In the context menu of the commit log you can open the advanced filter dialog. The advanced filter dialog allows you to filter for more specific commits. To remove the filter either remove the filter in the toolbar and press enter or remove the filter in the advanced filter dialog.

	(2.00) N	2.00	Upd	ate th	ne changelog for v3	(RussKie
♦ M	Copy to clipboard		juard	lensur	escore	(RussKie
s 📓	Checkout branch	•	rd for	Ensu	reScorelsAbove.	😫	Henk Westhuis
	Merge into current branch Rebase current branch on					ĺ	RussKie
57 🔊	Reset current branch to here					Ę	Henk Westhuis
	Create new branch					1	RussKie
Ri 🛟	Create new branch Rename branch	•					RussKie
1	Delete branch				-	•	Kusskie
Second Second			Con	sole	O Build Report		
Filter files	Compare	•	-				
Diff with: N 🔏 if GitUl/Git 🖉	Create new tag		<u>م</u>	Sho	w all branches		Ctrl+Skift+A
GitUI/Re	Delete tag	•	_		w current branch only	v	Ctrl+Skift+U
~	Checkout revision				w filtered branches	·	Ctrl+Skift+T
Sec. 1	Revert commit		~	Sho	w remote branches		Ctrl+Skift+R
•	Cherry pick commit Archive revision				w reflog references		
	Advanced				w superproject tags		
			-		w superproject branc	hes	
	Navigate View	•			w superproject remot		
a di la califacta di la califa	Open build report in the browser	•	~	Sho	w revision graph colu	umn	
			~		w author avatar colur		
Ģ	Open on GitHub supdate		~	Sho	w author name colun	nn	
	xxx		~	Sho	w date column		
-	, , , , , , , , , , , , , , , , , , ,		~	Sho	w SHA-1 column		
			~	Sho	w build status icon		
			~	Sho	w build status text		
			~	Drav	w non relatives gray		
			~	Sho	w author date		
					w relative date		
			~		w merge commits		Ctrl+Skift+M
					w tags		Ctrl+Alt+T
					w git notes		
				Higł	hlight selected branch	h (until refresh)	Ctrl+Skift+B
			¢		w first parents		Ctrl+Skift+S
			T	Set a	advanced filter		Ctrl+F

🗙 Filter		Х
Since	den 9 december 2018	
Until	den 9 december 2018	
Author		
Committer		
Message		
Ignore case		
Limit	100000	
File filter		
Branches		
	Show current branch only	
	Simplify by decoration	
	OK	

File history

To display the single file history, right click on a file name in the *Browse Repository* File tree or in the Diff tab and select File history or Blame. The single file history viewer shows all revisions of a single file. (This is available for submodules too, but the information is mostly not interesting.)



5.1 Commit

The Commit tab contains the information about the commit, including the other files in the commit. ¥ File History - GitUl/CommandsDialogs/FormBrowse.cs - F:\dev\gc\gitextensions_4 Branches: →
▼ → Filter: 🍸 • 🖕 | 🦓 • 🗊 🚸 • Add RefreshTree Rework RepoObjectsTree so that each sub-Tree (Branches, Remotes, etc.) registers for callbacks and handles its own update, instead of being told to reload Issue 5616: Provide a way to remove all invalid recent repositories. Merge pull request #5641 from NikolayXHD/fix_commit_info_scroll Force commit info to clear on module change 🎯 Commit 📑 Diff 🔎 View 🚸 Blame RussKie <RussKie@users.noreply.github.com> Author: 1 month ago (2018-10-31 06:54:11) Date: Committer: GitHub < noreply@githu Commit hash: 66380983451680c816d9a6c46797c37ed2507449 0e3d4bd55f e558ea50b8 Parents: Merge pull request #5641 from NikolayXHD/fix_commit_info_scroll Fix commitInfo scroll on mouse wheel Notes: Related links: View on GitHub, PR 5641 Contained in branches: vbjay/_AboutLayout, tmp/master, spdr870/_feature/gitcmdmissing, spdr870/_feature/5853, spdr870/_feature/5782, spdr870/_feature/5678, russkie/_fix_5644_unde diff --git a/GitUI/Com Diff with: 0e3d4bd5 index 7306fece0..759d8 GitUI/CommandsDialogs/FormBrowse.Designer.cs --- a/GitUI/CommandsDi / GitUl/CommandsDialogs/FormBrowse.cs +++ b/GitUI/CommandsDi // GitUl/CommitInfo/CommitInfo.Designer.cs @@ -1214,11 +1214,6 @@ // GitUI/CommitInfo/CommitInfo.cs GitUI/CommitInfo/CommitInfoHeader.Designer.cs // GitUI/CommitInfo/CommitInfoHeader.cs 1217 -// GitUl/MouseWheelRedirector.cs -Diff with: e558ea50 _ / .github/ISSUE_TEMPLATE.md _ 🛶 Externals/NBug (+1) } // GitCommands/GitCommands.csproj 1223 1218 + GitExtUtils/BinarySearch.cs private async GitExtUtils/GitExtUtils.csproj

5.2 Diff

You can view the difference report from the commit in the Diff tab.

Note: Added lines are marked with a +, removed lines are marked with a -.

var child

RevisionI

if (Revis

paren

paren

{

Branches:			• 🕆 • Filter:		
	Add Refres	hTree			
	Rework Rep	poObjectsTr	ee so that each sub-	Tree (Branches, R	emotes, et
	lssue 5616:	Provide a w	ay to remove all inv	alid recent reposit	tories.
ŧ١	Merge pull	request #56	41 from NikolayXHI)/fix_commit_info	o_scroll
•	Force com	mit info to c	lear on module cha	nge	
1.1.1	r:				
💩 Com	nmit 📱	Diff 🔎 🗤	/iew 🐞 Blame		
	index 7 a/0 +++ b/0	7306fece0. iitUI/Comn iitUI/Comn	UI/CommandsDial .759d88737 1006 andsDialogs/For andsDialogs/For 4,6 @@ private	544 mBrowse.cs mBrowse.cs	
1214 12 1215 12 1216 12	15		• children = Rev visionInfo.SetRe		
1217 1218	-	if {	(RevisionInfo.F	Parent is Pane	el paren
1219 1220	-		parent.AutoScr parent.AutoScr	-	Revisio
1221	-	}	· · · · · · · · · · · · · · · · · · ·		
1222 12 1223 12		}			
1224 12	19	private	async Task Fil	llGpgInfoAsyno	=()

¥ File History - GitUl/CommandsDialogs/FormBrowse.cs - F:\dev\gc\gitextensi

5.3 View

You can view the content of the file in after each commit in the ${\tt View}$ tab.

Branches:	▼ ▼ Filter: ▼
	Add RefreshTree
	Rework RepoObjectsTree so that each sub-Tree (Branches, Remotes, $\boldsymbol{\varepsilon}$
	Issue 5616: Provide a way to remove all invalid recent repositories.
\	Merge pull request #5641 from NikolayXHD/fix_commit_info_scroll
•	Force commit info to clear on module change
	Financial and the state of the
💩 Con	nmit 😫 Diff 🔎 View 🚸 Blame
	using System;
2 u	sing System.Collections.Generic;
3 u	ising System.ComponentModel;
4 u	ising System.Diagnostics;
5 u	ising System.Drawing;
6 u	ising System.Drawing.Drawing2D;
7 u	using System.IO;
8 u	using System.Linq;

X File History - GitUI/CommandsDialogs/FormBrowse.cs - F:\dev\gc\gitextens

5.4 Blame

There is a blame function in the file history browser. The commit for the selected line is displayed.

```
      X
      File History - GitUl/CommandsDialogs/FormBrowse.cs - F:\dev\gc\gitextensions_4

      Branches:
      ▼ ▼ < | Filter:</td>
      ▼ ▼ < 0 | №  < ∩</td>
```

Dianches								
	Add RefreshTree							
	Rework RepoObjectsTree so that each sub-Tree (Branches, Remotes, etc.) registers for callbacks and handles its own update, instead of being told							
	Issue 5616: Provide a way to remove all invalid recent repositories.							
\	Merge pull request #5641 from NikolayXHD/fix_commit_info_scroll							
•	Force commit info to clear on module change							
	Financial and the financial control of the financial states							
🧶 Co	nmit 😫 Diff 🔎 View 🔻 Blame							
STAL V	Author: Henk Westhuis < henk westhuis@hotmail.com> Date: 10 years ago (2008-11-27 20:17:44) Commit hash: bfcbc832fd9aa409f67fe9a7f3452a16739fe1fe Parent: a6a4eab406							
addeo	files							
Notes	:							
	Tal952 - 2012-06-15 00:03:18 - GitUI/FormBrowse.cs	1 using System;						
-	Henk Westhuis - 2008-11-27 20:17:44 - GitUI/Browse.cs	<pre>using System; using System.Collections.Generic;</pre>						
-	Drew Noakes - 2018-06-19 11:37:16 - GitUI/CommandsDialogs/FormBrowse.cs	<pre>using System.ComponentModel:</pre>						
-	Steffen Forkmann - 2010-07-28 16:44:38 - GitUI/FormBrowse.cs	Author: Henk Westhuis						
	Arkadiy Shapkin - 2011-10-28 01:53:22 - GitUI/FormBrowse.cs	AuthorTime: 2008-11-27 20:17:44						
	Drew Noakes - 2018-07-19 16:22:56 - GitUI/CommandsDialogs/FormBrowse.cs	Committer: Henk Westhuis						
	Steffen Forkmann - 2010-07-28 16:44:38 - GitUI/FormBrowse.cs	CommitterTime: 2008-11-27 20:17:44						

Double clicking on a code line shows the full commit introducing the change.

Commit

A commit is a set of changes with some extra information. Every commit contains the following information:

- Changes
- Committer name and email
- Commit date
- Commit message
- Cryptographically strong SHA1 hash

Each commit creates a new revision of the source. Revisions are not tracked per file; each change creates a new revision of the complete source. Unlike most traditional source control management systems, revisions are not named using a revision number. Each revision is named using a SHA1, a 41 long characters cryptographically strong hash.

6.1 Commit changes

Changes can be committed to the local repository. Unlike centralised source control management systems you do not need to checkout files before you start editing. You can just start editing files, and review all the changes you made in the commit dialog later. When you open the commit dialog, all changes are listed in the top-left.

diffgit a /acorce/comit.rst b/source/comit.rst Incer fac	Commit to feature/i5693-doc-3.00 (F:\dev\gc\gitextensions_4\GitExtensionsD	oc)			- 0	×
There are no staged changes Commit Commit message Commit genplates Up Create branch Options Image: Commit Republic Commit Republ	 Working directory changes • Filter files< • 	diffgit a/sour index b545bbb5 a/source/com +++ b/source/com @@ -13.7 +18.7 @ 20 21Changes can be c 21 4Changes can be c 22 22 checkout files b 23 23 dialog later. Wh	25c2 100644 it.rst iv.rst using a revision number. Each revision is named using a S ommitted to the local repository. Unlike most other source ommitted to the local repository. Unlike centralised source efore you start editing. You can just start editing files,	e control management systems you do not ce control management systems you do no and review all the changes you made i	need to t need to	
Committer Gerhard Olsson <gerhardol@users.noreply.github.com> Feature/5693-doc-3.00 Staged 0/1 Ln 0 Col 0</gerhardol@users.noreply.github.com>	There are no staged changes	Commit & push Stage in Superproject Amend Commit Reset all changes		1. 6 autor 65502 day 200 - Strand 0/1 - 1		

There are three kinds of changes:

Un-	This file is not yet tracked by Git. This is probably a new file, or a file that has not been committed to
tracked	Git before.
Modified	This file is modified since the last commit.
Deleted	This file has been deleted.

When you rename or move a file Git will notice that this file has been moved and notice in index pane (not in working directory).

During your initial commit there are probably lots of files you do not want to be tracked. You can ignore these files by not staging them, but they will show every time. You can instead add them to the .gitignore file of your repository. Files that are in the .gitignore file will not show up in the commit dialog again. You can open the .gitignore editor from the menu Working dir changes by selecting Edit ignored files.

0		in to reactine, is oss doe shoo (in (der (ge (
6	🔁 W	orking directory changes 👻	
Filter		Show ignored files	•
s 🖉		Show skip-worktree files	
so 🖉		Show assumed-unchanged files	
— so	~	Show untracked files	
🕇 50		Delete selected files	
		Reset selected files	
	÷.	Reset unstaged changes	
	÷.	Reset all (tracked) changes	
		Edit ignored files	
		Edit locally ignored files	
		Delete all untracked files	
		Selection filter	🌷 Stage 🛛 🌷
Then	e are i	no staged changes	

X Commit to feature/i5693-doc-3.00 (F:\dev\gc\gitextensions_4\GitExtensionsDoc)

Making a commit is a two step procedure:

- Adding to index (staging) the changes to be committed, which saves a snapshot of the changes into the Git "index".
- Committing those staged changes, which records the staged changes and other information into the repository.

You do not have to commit immediately after staging changes. You can close the commit dialog, make further changes to the files in the working dir, then re-open the commit dialog to stage further changes and commit. Changes that you have staged previously will still be staged when you re-open the dialog.

6.1.1 Staging changes

The changes that you have made to your working directory are not automatically included in a commit. You must choose which of the changed files, or individual changes from within those files, will be included in the commit by "staging" the changes in Git Extensions. Staging changes in Git Extensions is the same as using git add on the Git command line.

You can stage the changes you want to commit by selecting the files in the top-left or "Unstaged changes" pane and pressing the Stage button or pressing the [S] key. The file entries will move to the lower left or "Staged changes" pane. You need to stage deleted files because you stage the change and not the file. If you have staged changes from a file and you wish to exclude those changes from the commit, select the entry in the staged changes pane and press the Unstage button or press the [U] key.

If the file that is selected in either the unstaged or staged changes pane is text format, Git Extensions will show a Git "diff" view in the right side pane of the window.

6.1.2 Staging selected lines

You do not have to commit all of the changes in a text format file in one commit. You can select and stage individual lines from within a file such that only the chosen lines will be included in your next commit; the remaining changes in

the file will appear as unstaged changes for the next commit.

In the diff view on the right, select the line or lines that you want to stage then right-click and choose Stage selected line(s) or press the [S] key. The file will now appear in both the staged changes and unstaged changes panes on the left since now there are both staged and unstaged changes in the same file. The change that was selected will disappear from the diff view on the right because the diff view is showing only the unstaged changes.

To see the line changes that have been staged select the entry for the file in the staged changes pane. To unstage selected changed lines from a file, select that file in the staged changes pane, then select the line or lines in the diff view, right -click, and choose Unstage selected line(s) or press the [U] key.

Note: If you select an entire line including the end-of-line character then staging or unstaging that line will include both the selected line and the next line. To select a single line to stage or unstage you may simply click onto the line without selecting any particular characters.

Note: Staging and unstaging individual lines from a file does not change the file itself. It is simply choosing which changes from within that file will be included in the next commit.

6.1.3 Undoing or resetting changes

You can undo or reset changes to files from the commit dialog. You can only do this from the top-left or "Unstaged changes" pane. If you have already staged the changes then you must first unstage them as described above. To reset the changes in a file, select the file in the unstaged changes pane, right-click and choose Reset file or directory changes or press the [R] key.

X	Comr	nit to feature/i5693-doc-3.00 (F:\dev\gc\	gitextensions_4\Git	tExtens	ionsD	oc)	
0	🔁 V	Vorking directory changes 👻					
Filter		Show ignored files	1		-		
🥖 so		Show skip-worktree files					
🥖 so		Show assumed-unchanged files					
- so	~	Show untracked files					
🕇 so		Delete selected files					
		Reset selected files					
	\$ }	Reset unstaged changes					
	\$	Reset all (tracked) changes					
		Edit ignored files					
		Edit locally ignored files					
		Delete all untracked files					
		Selection filter	e 🦊	Stage	\$		
Ther	e are	no staged changes				Commit	G C
						☆ Commit & push	Ente
						Stage in Superproject	
						Amend Commit	
						Reset all changes	
						Reset unstaged changes	

You can reset individual changed lines in a similar way to staging and unstaging individual lines, which are described above. To reset an individual line, select the line or lines in the diff view on the right then right-click and choose Reset selected lines or press the [R] key.

Warning: Resetting changes modifies the file, discarding either all of the changes or the changes on the selected lines.

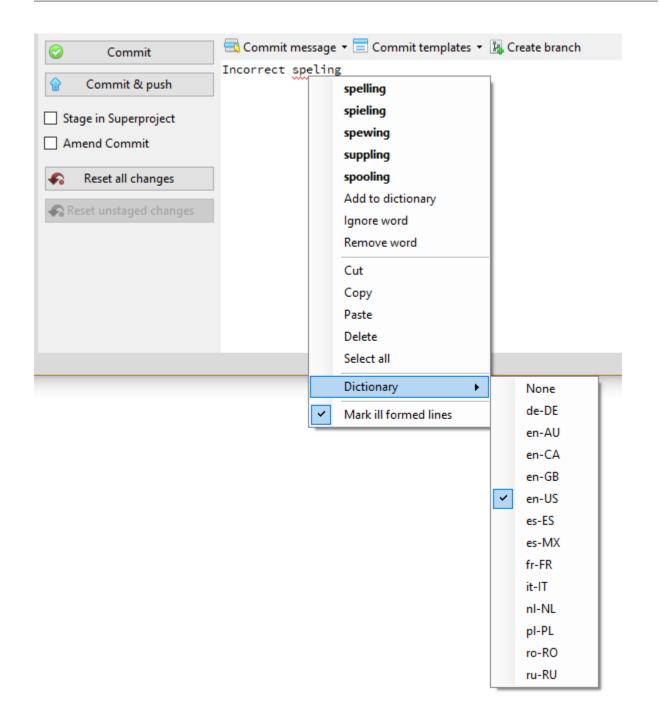
6.1.4 Making the commit

When all the changes you want to commit are staged, enter a commit message into the lower-right pane and press the commit button.

X Commit to feature/i5693-doc-3.00 (F:\dev\gc\gitextensions_4\GitExtensionsDoc) 🌄 🔄 Working directory changes 🝷 Ŧ source/commit.rst source/images/commit_dialog.png source/images/reset_changes.png + source/images/commit_reset_changes.png () 🌷 Stage 🛛 🌷 👚 Unstage 🔜 Commit message 🔻 Commit Commit & push Stage in Superproject Amend Commit Reset all changes Reset unstaged changes

There is a built-in spelling checker that checks the commit message. Incorrectly spelled words are underlined with a wavey red line. Right-click on the misspelled word to choose the correct spelling or choose one of the other options.

Git Extensions installs a number of dictionaries by default. You can choose another language in the context menu of the spelling checker or in the settings dialog. To add a new spelling dictionary add the dictionary file to the Dictionaries folder inside the Git Extensions installation folder.



6.2 Amend commit

It is also possible to add changes to your last commit by checking the Amend Commit checkbox. This can be very useful when you forgot some changes. This function rewrites history; it deletes the last commit and commits it again including the added changes.

See also *Modify Git history*, especially if you have published the changes to a repote repository already.

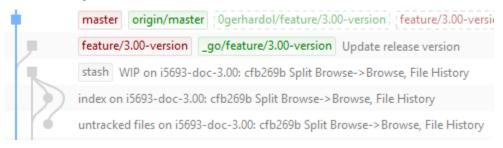
Stash

If there are local changes that you do not want to commit yet and not want to throw away either, you can temporarily stash them. This is useful when working on a feature and you need to start working on something else for a few hours. You can stash changes away and then reapply them to your working dir again later. Stashes are typically used for very short periods.

🗙 Stash				– 🗆 ×
Show: WIP on i5693	-doc-3.00: cfb26! 🔻 🖉 🦓	1	3	👔 🔎 🔎 📓 🏾 🗮 분분 US-ASCII 🔹 🏶
🧪 make.cmd				a/readme.md
🧪 readme.md				+++ b/readme.md
				@@ -18,6 +18,11 @@ up to date).
		18	18	Build
		19	19	
		20	20	
				+To generate the documentation, you need to have Sph +If you have
			23	+ pip install -U sphinx
			24	+ sphinx-build -b html -d build/doctrees source buil
			25	+
		21	26	
		22	27	
		23	28	file. The `make_and_start_Browser.cmd` is an alias
WIP on i5693-doc-3.0 Browse, File History	0: cfb269b Split Browse->			
Keep index	✓ Include untracked			
Stas	sh all changes			
Stash s	elected changes			
Drop	Selected Stash			
Apply	/ Selected Stash	<		>

7.1 Revision graph

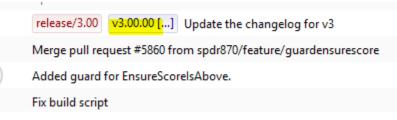
You can create multiple stashes if needed. The latest stash is shown in the commit log with the text [stash], all stashes if reflog is visible (see *Maintenance*).



The stash is especially useful when pulling remote changes into a dirty working directory. If you want a more permanent stash, you should create a branch.

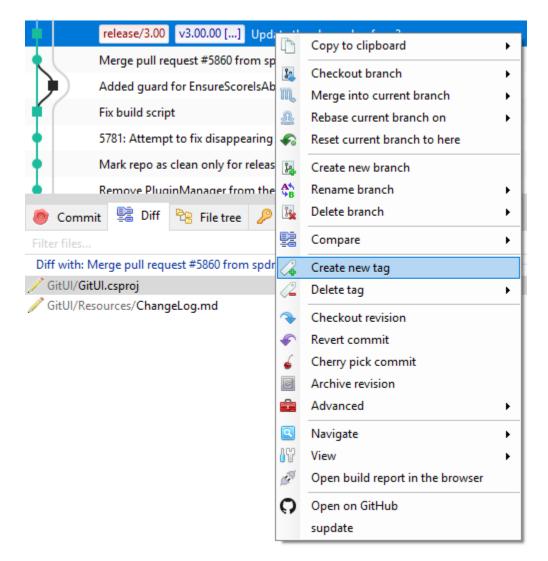
Tag

Tags are used to mark a specific version. Usually a tag will not be moved anymore. The image below shows the commit log of Git Extensions with a tag indicating version [3.00.00].



8.1 Create tag

In Git Extensions you can tag a revision by choosing Create new tag in the commit log context menu. A dialog will prompt for the name of the tag. You can also choose Create tag from the Commands menu, which will show a dialog to choose the revision and enter the tag name.



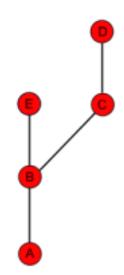
Once a tag is created, it cannot be moved again. You need to delete the tag and create it again to move it.

8.2 Delete tag

Tags can be deleted, read about "What should you do when you tag a wrong commit and you would want to re-tag?" here: https://www.kernel.org/pub/software/scm/git/docs/git-tag.html#_on_re_tagging

🗙 Delete tag	×
Select tag	v3.00.00 V Delete
This will delete	e the selected tag from the (local) repository.
🗌 Delete tag	also from the following remote(s):
0gerhardol	~
🔯 <u>Help</u> (i	includes information about deleting tags which are already pushed)

Branches



Branches are used to commit changes separate from other commits. It is very common to create a new branch when you start working on a feature to keep the work done on that feature separate from other work. When the feature is complete the branch can be merged or rebased as you choose such that the commits for the feature either remain as a parallel branch or appear as a continuous single line of development as if the branch had never existed in the first place. The image on the right illustrates a branch created on top of commit B.

You can see the name of your current branch in a combo box in the toolbar. You can switch to another branch by choosing from the combo box list. In the commit log the current branch has an arrow head to the left of its name. If you are not currently on a branch because you have checked out a specific commit but not any particular branch then Git Extensions will show (no branch) in place of a branch name in the toolbar. This is called "Detached HEAD mode". In Git you can refer to your current branch or commit by the special reference HEAD in place of the branch name or commit reference.



9.1 Create branch

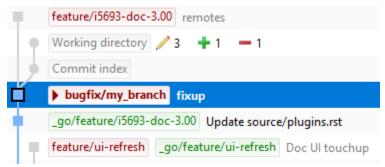
In Git Extensions there are multiple ways to create a new branch. In the image below I create a new branch from the context menu in the commit log. This will create a new branch on the revision that is selected.

•	Working	directory 🥖 1 📫 1 💻 1	
•	Commit	index	
¢.	▶ featur	e/i5693-doc-3.00 remotes	
÷	fixup D_C feat	Copy to clipboard Merge into current branch Rebase current branch on	jins.rst touchuj
•	tag 🎧	Reset current branch to here	
•	stash 🙀	Create new branch	
1	com	Compare	•
T 1	file_ł 🔊	Croate new tag	

I will create a new branch called feature/refactor. In this branch I can do whatever I want without affecting others. The default in Git Extensions is to check out a new branch after it is created. If you want to create a new branch but remain on your current branch, uncheck the Checkout after create checkbox in the Create branch dialog.

🗙 Create branch	×
Branch name	bugfix/my_branch
Create branch at this revision	a83867c8b3 3
Checkout after create	$\overline{\checkmark}$
Orphan	
Create orphan 🗹 Cle	ar working directory and index
🔯 <u>Help</u>	📴 Create branch

When the branch is created you will see the new branch feature/refactor in the commit log. If you chose to checkout this branch the next commit will be committed to the new branch.



Creating branches in Git requires only 41 bytes of space in the repository. Creating a new branch is very easy and fast. The complete work flow of Git is optimized for branching and merging.

9.1.1 Orphan branches

In special cases it is helpful to have orphan branches (see for example https://www.google.com/search?q=why+use+ orphan+branches+in+git). Check the "Create orphan" checkbox to create an orphan branch (--orphan option in git).

The newly created branch will have no parent commits.

The option "Clear working dir and index" (git rm -rf) is active by default. So the working dir and index will be cleared. If you uncheck the last option then the working dir and index will not be touched.

9.2 Checkout branch

You can switch from the current branch to another branch using the checkout command. Checking out a branch sets the current branch and updates all of the source files in the working directory. Uncommitted changes in the working directory can be overwritten so it is best practice to make sure your working directory is clean by either committing or stashing any current changes before checking out a branch. If you do not clean your working directory then, in the Checkout branch dialog, you can choose between four options for your local uncommitted changes:

Don't	Local changes will be retained if there are not conflicting changes from the branch you are checking
change	out.
Merge	Performs a three-way merge between your current branch, your local changes and the branch you are
	checking out.
Stash	Your local changes are stashed and the new branch is checked out. You can retrieve your changes on
	the new branch with stash-pop.
Reset	Your local changes are discarded and the new branch is checked out. Use caution with this option as
	Git has no record of uncommitted changes so they cannot be retrieved.

•	Working directory 🥒 4 🛛 🛨 1 👘 1		
•	Commit index		
¢.	Feature/i5693-doc-3.00 remotes		
	bugfix/my_branch fixup		
	▷ _go/feature/i5693-doc-3.00 Update source/plugins.rs	st	
	feature/ui-refresh _go/feature/ui-refresh Doc	Copy to clipboard	
۰.	tag	Checkout branch	feature/ui-refresh
۰.	stash	Merge into current branch	_go/feature/ui-refresh
•	commit. split to stash. modify history	Rebase current branch on	

9.3 Merge branches

In the image below there are two branches, [feature/refactor] and [master]. We can merge the commits from the master branch into the feature/refactor branch. If we do this, the feature/refactor branch will be up to date with the master branch, but not the other way around. As long as we are working on the feature/refactor branch we cannot touch the master branch itself. We can merge the sources of master into our branch, but cannot make any change to the master branch.

•	Working directory	
•	Commit index	
Þ	▶ feature/refactor dummy change 2	Gerha
•	dummy change 1	Gerha
1.1	master gitextensions/master Correct version nu 🎇	Gerhai
•	Update build version 3.00.00 -> 3.01.00.0 (#5881)	Gerhai
	feature/version-update Update build version 2.99.90 ->	Gerha
	spdr870/feature/lockfreelist Review changes	Henk
	Merge pull request #5878 from drewnoakes/fix-5859 🚥 🌉	RussKi

To merge the feature/refactor branch into the master branch, we first need to switch to the master branch.

	feature/refactor dummy change 2
•	dummy change 1
•	Working directory
•	Commit index
	▶ master ▷ gitextensions/master Correct versi 🚥 🎇
•	Update build version 3.00.00 -> 3.01.00.0 (#5881)
	feature/version-update Update build version 2.99.90 ->
	spdr870/feature/lockfreelist Review changes
	Merge pull request #5878 from drewnoakes/fix-5859 🚥 🔛

Once we are on the master branch, select the feature/refactor branch and select merge. Alternatively choose Merge branches from the Commands menu and select the feature/refactor branch.

9	Å	Copy to clipboard	•) 😐	Gerhard C
•	1	Checkout branch	•	<u> </u>	Gerhard C
•	M,	Merge into current branch	•	feature/ref	actor
•	£	Rebase current branch on	+		
- P -	\$	Reset current branch to here		t versi 😐 🎽	🕺 Gerhard O
•	J.	Create new branch		81) 🌋	🕺 Gerhard O
	A ⁴s SB	Rename branch	+	n 2.99.90 -> 빌	Gerhard C
	I.	Delete branch	•	5	Henk West
		Compare	+	ix-5859 🚥 🚺	RussKie
	12	Crosto pourtad			

In the merge dialog you can verify which branch you are working on. Select the branch to merge with then click the Merge button.

X Merge branches	×
Hide help Hover to see scenario when fast forward is possible. merge commit	Merge Merge branch <u>feature/refactor</u> Into current branch master
f other + .REMOTE f ther + .LOCAL + .LOCAL + .LOCAL f ther f th	 Keep a single branch line if possible (fast forward) Always create a new merge commit Do not commit Show advanced options Use non-default merge strategy Squash commits Allow unrelated histories Add log messages 20 +
Current branch Postfix for files if merge conflicts occur	Merge

After the merge the commit log will show the new commit containing the merge. Notice that the feature/refactor branch is not changed by this merge. If you want to continue working on the feature/refactor branch you can merge the feature/refactor branch with master. You can instead delete the feature/refactor branch if it is not used anymore.

•	Working directory
•	Commit index
L	master Merge branch 'feature/refactor'
•	feature/refactor dummy change 2
•	dummy change 1
	j gitextensions/master Correct version number fo
I	Update build version 3.00.00 -> 3.01.00.0 (#5881)
	feature/version-update Update build version 2.99.90 ->
	spdr870/feature/lockfreelist Review changes
	Merge pull request #5878 from drewnoakes/fix-5859

Note: When you need to merge with an unnamed branch you can use a tag to give it a temporary name.

Note: During a merge conflicts can occur. See *Merge Conflicts* for more information.

9.4 Rebase branch

The rebase command is the most complex command in Git. The rebase command is very similar to the merge command. Both rebase and merge are used to get a branch up-to-date. The main difference is that rebase can be used to keep the history linear contrary to merges.

•	Working directory
•	Commit index
Þ	feature/refactor dummy change 2
•	dummy change 1
1.7	master gitextensions/master Correct version nu
•	Update build version 3.00.00 -> 3.01.00.0 (#5881)
	feature/version-update Update build version 2.99.90 ->
	spdr870/feature/lockfreelist Review changes
	Merge pull request #5878 from drewnoakes/fix-5859 🚥 🎚

Select the commit where you want to to rebase the current branch.

	_			
- 1	ß	Copy to clipboard	•	📕 Gerhard C
•	8	Checkout branch	•	Gerhard C
•	M,	Merge into current branch	•	feature/refactor
	£	Rebase current branch on	×	
L 🗭 🗌	\$ 0	Reset current branch to here		: t versi 😐 🎇 Gerhard O
•	j.	Create new branch		81) 💥 Gerhard O
	A 4 98	Rename branch	+	n 2.99.90 -> 芷 Gerhard C
	J.	Delete branch	•	s Henk West
		Compare	•	ix-5859 🚥 🌆 RussKie
	12	Create new tag		

A rebase of feature/refactor on top of master will perform the following actions:

- All commits specific to the feature/refactor branch will be stashed in a temporary location
- The branch feature/refactor will be removed
- The branch feature/refactor will be recreated on the master branch
- All commits will be recommitted in the new feature/refactor branch

Note: During a rebase merge conflicts can occur. You need to solve the merge conflicts for each commit that is rebased. The rebase function in Git Extensions will guide you through all steps needed for a successful rebase. See *Merge Conflicts* for more information.

🗙 Rebase		– 🗆 X
<u>Hide help</u>	Rebase current branch on top of another branch Current branch: feature/refactor	
	Rebase on 7b342f925f740a74957a5570afc58a20d 🗸	Rebase
f other d current	🗌 Interactive Rebase 📄 Preserve Merges 🗹 Autosquash 🗹 Auto stash	
4.LOCAL	Specific range From (exc.)	Solve conflicts
	Commits to re-apply:	Add files
d current 9.REMOTE e	Name Subject Author Date Status	Commit
b f.BASE b		Continue rebase
		Skip this commit
• •		Abort
square = green = current branch new commit		
4.AAA postfix for files if merge conflicts occur		

The image below shows the commit log after the rebase. Notice that the history is changed and it seems like the commits on the feature/refactor branch are created after the commits on the master branch.

•	Working directory
•	Commit index
	feature/refactor dummy change 2
•	dummy change 1
	master gitextensions/master Correct version nu 🚥 🐉
•	Update build version 3.00.00 -> 3.01.00.0 (#5881)
_	feature/version-update Update build version 2.99.90 ->
	spdr870/feature/lockfreelist Review changes
	Merge pull request #5878 from drewnoakes/fix-5859 🚥 🌆

Warning: Because this function rewrites history you should only use this on branches that are not published to other repositories yet. When you rebase a branch that is already pushed it will be harder to pull or push to that remote. If you want to get a branch up-to-date that is already published you should merge.

9.5 Interactive rebase

It is possible to modify the order, merge commits etc when committing.

See Modify Git history for more information.

9.6 Delete branch

Since it is common to create many branches, it is often necessary to delete branches. Most commonly you will need to delete branches on which work has finished and their contents are merged into master or your main branch. You can also delete unmerged branches when they are not needed anymore and you do not want to keep the work done in that branch.

When you delete a branch that is not yet merged, all of the commits that are in only the deleted branch will be lost. When you delete a branch that is already merged with another branch, the merged commits will not be lost because they are also part of another branch.

You can delete a branch using Delete branch from the Commands menu. If you want to delete a branch that is not merged into your current branch (HEAD in Git), you need to check the Force delete checkbox.

Þ.	▶ feature/i5693-doo	c- 3.0	0 remotes		
•	bugfix/my_branch f	ß	Copy to clipboard		
۰.	_go/feature/i5693-	8	Checkout branch	•	
1	feature/ui-refresh	m,	Merge into current branch	•	
• 1	tag	Ł	Rebase current branch on	•	
	stash	\$	Reset current branch to here		
•	commit, split to stash,	Ŀ,	Create new branch		
•	file_history	A4 9B	Rename branch	•	
	fixup explorer	i	Delete branch	•	bugfix/my_branch
		D2	Compare		

Patches

Every commit contains a change-set, a commit date, the committer name, the commit message and a cryptograph SHA1 hash. Local commits can be published by pushing it to a remote repository. To be able to push you need to have sufficient rights and you need to have access to the remote repository. When you cannot push directly you can create patches. Patches can be e-mailed to someone with access to the repository. Each patch contains an entire commit including the commit message and the SHA1.

```
1 From 58c02ec4701c94c671a41e1e5d50c582e859851f Mon Sep 17 00:00:00 2001
 2 From: Russell King <rmk@dyn-67.arm.linux.org.uk>
 3 Date: Sun, 17 Apr 2005 15:40:46 +0100
 4 Subject: [PATCH 000213/123824] [PATCH] ARM: h3600 irda set speed arguments
 5
 6 h3600 irda set speed() had the wrong type for the "speed" argument.
 7 Fix this.
 8
9 Signed-off-by: Russell King <rmk@arm.linux.org.uk>
10 ----
11 arch/arm/mach-sal100/h3600.c |
                                      2 +-
12 1 files changed, 1 insertions(+), 1 deletions(-)
13
14 diff --git a/arch/arm/mach-sa1100/h3600.c b/arch/arm/mach-sa1100/h3600.c
15 index 9788d3a..84c8654 100644
16 --- a/arch/arm/mach-sal100/h3600.c
17 +++ b/arch/arm/mach-sal100/h3600.c
18 @@ -130,7 +130,7 @@ static int h3600 irda set power(struct device *dev, unsigned int state)
19
      return 0;
20 }
21
22 -static void h3600 irda set speed(struct device *dev, int speed)
23 + static void h3600 irda set speed(struct device *dev, unsigned int speed)
24 {
      if (speed < 4000000) {
25
26
          clr_h3600_egpio(IPAQ_EGPIO_IR_FSEL);
27 --
28 1.6.1.9.g97c34
```

10.1 Create patch

Format a single patch or patch series using the format patch dialog. You need to select the newest commit first and then select the oldest commit using ctrl-click. You can also select an interrupted patch series, but this is not recommended because the files will not be numbered.

🗙 Format patch						×
Save patches in directory	F:\temp				Browse	
 Mail patches from 						
То				~		
Subject						
Body						
Feature/i5693-do	c-3.00 branches		Gerhard Olsson	2018-12-11 23:41:59	3bc70ac	^
 remotes 		11	Gerhard Olsson	2018-12-11 00:06:33	a813a63	
 fixup 		12	Gerhard Olsson	2018-12-11 00:05:52	a83867c	
_go/feature/i5693-	doc-3.00 Update source/plugins.rst		Drew Noakes	2018-12-11 15:29:52	8191893	
feature/ui-refresh	_go/feature/ui-refresh Doc UI touchup	··· 🔛	Gerhard Olsson	2018-12-10 22:15:01	e63e13c	
🔶 tag		11	Gerhard Olsson	2018-12-10 00:53:18	9f69221	
 stash 		11	Gerhard Olsson	2018-12-10 00:33:08	ff2e237	
commit, split to stash	, modify history	11	Gerhard Olsson	2018-12-10 00:30:10	4efba20	
 file_history 		×	Gerhard Olsson	2018-12-09 23:32:58	228c2aa	
fixup explorer		×	Gerhard Olsson	2018-12-09 23:10:09	5767560	~
Current branch: feature/i569	13-doc-3.00			Cre	ate patch(es)	

When the patches are created successfully the following dialog will appear.

Patch result	×
F:/temp/0001-branch	hes.patch
E	ОК

10.2 Apply patches

It is possible to apply a single patch file or all patches in a directory. When there are merge conflicts applying the patch you need to resolve them before you can continue. Git Extensions will help you applying all patches by marking the next recommended step.

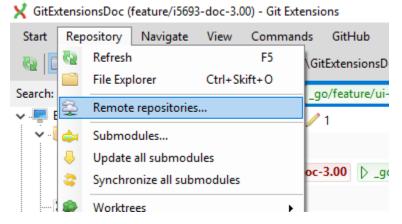
🗙 Арр	ly patch (F:\dev\		- 🗆 ×				
● Pato ○ Pato	:h file :h directory	F:\temp\0001-branch	es.patch	Browse	Apply patch Ignore Wh.spc. 		
Name	Subject	Author	Date	Status	Solve conflicts		
					Add files		
					Conflicts resolved		
					Skip patch Abort patch		

Remotes

Git is a distributed source control management system. This means that all changes you make are local. When you commit changes, you only commit them to your local repository. To publish your local changes you need to push. In order to get changes committed by others, you need to fetch/pull.

11.1 Manage remote repositories

You can manage the remote repositories in the Remotes menu.



When you cloned your repository from a public repository, this remote is already configured. You can rename each remote for easy recognition. The default name after cloning a remote is origin. If you use PuTTY as SSH client you can also enter the private key file for each remote. Git Extensions will load the key when needed. How to create a private key file is described in the next paragraph.

X Remote repositor	ies			×
Remote repositories	Default pull be	ehavior (fetch & merge)		
Active	+	Edit Remote Details		
_go mstv		Name	_go	
origin	Ø	Url	https://github.com/gerhardol/GitExtensior ~	
		Separate P	ush Url	
			🝙 Save changes	

In the Default pull behaviour tab you can configure the branches that need to be pulled and merged by default. If you configure this correctly you will not need to choose a branch when you pull or push. There are two buttons on this dialog:

Prune remote branches	Throw away remote branches that do not exist on the remote anymore.
Update all remote branch info	Fetch all remote branch information.

Teature/ui-refresh _go Teature/ui-refresh _go feature/i5693-doc-3 _go feature/i5693-doc-3 Default merge with master origin master feature/3.00-version _go feature/3.00-version release/3.00	🗙 Remote repositorie	s				Х
name repository with Local branch name reature/ui-refresh feature/ui-refresh _go feature/ui-refresh Remote repository feature/i5693-doc-3 _go feature/i5693-doc-3 Default merge with master origin master feature/3.00-version _go feature/3.00-version release/3.00 Image: Color branch name _go	Remote repositories	Default pull beha	vior (fetch & merge)			
reactive/unrenesition go reactive/unrenesition feature/i5693-doc-3 _go feature/i5693-doc-3 master origin master feature/3.00-version _go release/3.00			-	Local branch name	feature/ui-refresh	
master origin master feature/3.00-version	feature/ui-refresh	_go	feature/ui-refresh	Remote repository	_go	~
feature/3.00-version _go feature/3.00-version release/3.00	feature/i5693-doc-3	_go	feature/i5693-doc-3	Default merge with	feature/ui-refresh	~
feature/3.00-version _go feature/3.00-version release/3.00	master	origin	master		Save changes	
	feature/3.00-version	_go	feature/3.00-version		all our changes	
latest origin latest	release/3.00					
latest latest	latest	origin	latest			
			Prune remote bra	nches	Update all remote branch info	

After cloning a repository you do not need to configure all remote branches manually. Instead you can checkout the remote branch and choose to create a local tracking branch.

11.2 Git Credential Manager

The Git Credential Manager can be used to authenticate http links. For more information and instructions, see https://github.com/Microsoft/Git-Credential-Manager-for-Windows

11.3 Create SSH key

Git uses SSH for accessing private repositories. SSH uses a public/private key pair for authentication. This means you need to generate a private key and a public key. The private key is stored on your computer locally and the public key can be given to anyone. SSH will encrypt whatever you send using your secret private key. The receiver will then use the public key you send to decrypt the data.

This encryption will not protect the data itself but it protects the authenticity. Because the private key is only available to the sender, the receiver can be sure about the origin of the data. In practise the key pair is only used for the authentication process. The data itself will be encrypted using a key that is exchanged during this initial phase.

11.3.1 PuTTY and github

PuTTY is SSH client that for Windows that is a bit more user friendly then OpenSSH. Unfortunately PuTTY does not work with all servers. In this paragraph I will show how to generate a key for github using putty.

First make sure GitExtensions is configured to use PuTTY and all paths are correct, see SSH

🗙 GitExtensionsDoc (feature/i5693-doc-3.00) - Git Extensions

Start Repository Navigate	e View Commands GitHub Plugins	Too	ols Help				
🗞 🔽 🗁 🕶 🔁 🕶 F:\.c	dev\gc\gitextensions_4\GitExtensionsDoc 🔻 fe	* *	Git bash C Git GUI	Ctrl+G	Ӯ Commit (6) 🛛 👔 (1) 👻 📄 🚸 🌐 🛛 Bra		
Search:	feature/ui-refreshgo/feature/ui-refresh		GitK				
✓ · Image: Vertical and the second secon	Working directory / 4 - 2						
🗸 🥑 feature		₽	PuTTY	•	1	Start authentication agent	
🦾 🖗 ui-refresh	Commit index		Git command log	F12	2	Generate or import key	
🦾 👂 i5693-doc	▶ feature/i5693-doc-3.00 ▷ _go/feature		on command log	,g 112			
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>		-	Settings	Ctrl+,			
🔤 🖟 master	stash	_					
> 🦻 release	commit, split to stash, modify history						

can choose Generate or import key to start the key generator.

PuTTY Key Generator	×	😴 PuTTY Key Genera		
le Key Conversions Help		File Key Conversi	ons Help	
Key Please generate some randomness by moving the mouse over the bl	ank area	Key Public key for posting	g into OpenSSH authorized keys file:	
Prease generale some randomness by moving the mouse over the bi	anik area.	ssh-rsa AAAAB3NzaC1yc2E Jo+m6t6n +5TBgKaVWCyCeC	g mto OpenSSH addinized_keys me. EAAAABJQAAAQEAhR4ujlBw&gTyWR5+g3Ssl IzQSfrxKKNKQbuR5Kyws6WBm3lbJlvdkey0kqr ofNHu0EzBXzEbP27vBHmBwnnnUsX3S	
		Key fingerprint:	ssh-rsa 2048 b6:83:7d:f1:b9:6a:df:c3:75:b5:	b4:ca:ca:4d:8c:03
		Key comment:	rsa-key-20181210	
		Key passphrase:		
		Confirm passphrase:		
Actions		Actions		
Generate a public/private key pair	Generate	Generate a public/pr	ivate key pair	Generate
Load an existing private key file	Load	Load an existing priv	ate key file	Load
Save the generated key Save public key	Save private key	Save the generated	Save public key	Save private key
Parameters		Parameters		
Type of key to generate: RSA DSA ECDSA ED25515	9 OSSH-1 (RSA)	Type of key to gener RSA	ate:) DSA O ECDSA O ED25515	SSH-1 (RSA)
Number of bits in a generated key:	2048	Number of bits in a g	enerated key:	2048

PuTTY will ask you to move the mouse around to generate a more random key. When the key is generated you can save the public and the private key in a file. You can choose to protect the private key with a password but this is not necessary.

Now you have a key pair you need to give github the public key. This can be done in Account Settings in the tab SSH Public Keys. You can add multiple keys here, but you only need one key for all repositories.

O Add new SSH keys	× +											
← → ♂ ☆	①	· 🗵 🕁	lii\	⊡ »	≡							
Search or jump to	Pull requests Issues Marketplace Explore		<u>ب</u>	+- 😐	•							
Personal settings	SSH keys / Add new											
Profile	Title											
Account	Demo											
Emails	Key											
Notifications	ssh-rsa											
Billing	AAAAB3NzaC1yc2EAAAABJQAAAQEAhR4ujlBw8igTyWR5+g3SsUnv2qKi3jT4u	•	-	-								
SSH and GPG keys	X3S+1XMmE9EeOjsvhaiGrSCnVPmLZybKJmxQPmNGM/tijl8mNOK7q/TDQG5	SlfvKKNKQbuR5Kyws6WBm3lbJlvdkey0kqmYreaYQXqiBoh5u626k3QixAFkcxlsnhwofNHu0Ez8XzEbP27vBHmBwnnnUs X35+1XMmE9EeQisvhaiGr5CnVPmLZybKJmxQPmNGMttjildmNOKTq/TDQG5ucK1CCqZE0Ukt98k										
Security	26Q4+ZzwKw== rsa-key-20181210	/H+lj+bcJlaHdPltOP8iAsBftlbcVmeMzrSqJjJPjY6T+oLMu05WyS5Rb1iT7WazxtdTT7HijXuEPfNfcVRxh+cToQfE1TyPzBkg 26Q4+ZzwKw== rsa-key-20181210										
Sessions												
Blocked users				.::								
Repositories	Add SSH key											
Organizations					~							

After telling github what public key to use to decrypt, you need to tell GitExtensions what private key to use to encrypt. Load the private key into the PuTTY authentication agent in Clone dialoge or by starting the PuTTY authentication agent and choose add key in the context menu in the system tray.

GitExtensions can load the private keys automatically for you when communicating with a remote. You need to

configure the private key for the remote.

This is done in the Manage remote repositories dialog.

11.3.2 OpenSSH and github

To configure GitExtensions to use OpenSSH, see SSH.

OpenSSH is the best SSH client there is but it lacks Windows support. Therefore it is slightly more complex to use. Another drawback is that GitExtensions cannot control OpenSSH and needs to show the command line dialogs when OpenSSH might be used. GitExtensions will show the command line window for every command that might require a SSH connection. For this reason PuTTY is the preferred SSH client in GitExtensions.

To generate a key pair in OpenSSH you need to go to the command line. I recommend to use the git bash because the path to OpenSSH is already set. Open the separate Git bash or the console tab.

🗙 GitE	xtensionsDoc (feature/i5693	3-doc-3.(00) - Git Extensio	ons								
Start	Repository	Navigate	View	Commands	GitHub	Plugins	Tools	Help					
6 2		🛅 🔻 F:\de	v\gc\git	extensions_4\Git	tExtensions	Doc 🔻 fea	ature/i569)3-doc-3.00	• • 😽 • 😭	🕝 Commit	[(1) 👻	1	6

Type the following command: ssh-keygen -C "your@email.com" -t rsa Use the same email address as the email address used in git. You will be asked where if you want to protect the private key with a password. This is not necessary. By default the public and private keys are stored in c:\Documents and Settings\[User]\.ssh\ or c:\Users\[user]\.ssh\.

• Update to GE icons		
💩 Commit 😫 Diff 🗞 File tree 👂	GPG Console	
<pre>ejgo@ejgo3 MINGW64 /f/dev/gc/gitexte \$ ssh-keygen -C "your@email.com" -t Generating public/private rsa key pa Enter file in which to save the key Enter passphrase (empty for no passp Enter same passphrase again: Your identification has been saved in Your public key has been saved in /c The key fingerprint is: SHA256:qaipsWhHZZHpBsIrF4ahtpPue4ncw The key's randomart image is: +[RSA 2048]+ loo o looo. + .oo.o. o++. = . BBo.+ S *E0 =*==o. +BoB. =+*. +[SHA256]+</pre>	ensions_4/GitExtensionsDoc (feature/i5693-doc-3.00) rsa air. (/c/Users/ejgo/.ssh/id_rsa): /c/Users/ejgo/.ssh/github_test_ phrase): in /c/Users/ejgo/.ssh/github_test_rsa. c/Users/ejgo/.ssh/github_test_rsa.pub.	rsa

You do not need to tell GitExtensions about the private key because OpenSSH will load it for you. Now open the public key using notepad and copy the key to github. This can be done in Account Settings in the tab SSH

Public Keys on GitHub.

11.4 Pull changes

You can get remote changes using the pull function. Before you can pull remote changes you need to make sure there are no uncommitted changes in your local repository. If you have uncommitted changes you should commit them or stash them during the pull. You can read about how to use the stash in the Stash chapter.

In order to get your personal repository up-to-date, you need to fetch changes from a remote repository. You can do this using the Pull dialog. When the dialog starts the default remote for the current branch is set. You can choose another remote or enter a custom url if you like. When the remote branches configured correctly, you do not need to choose a remote branch.

If you just fetch the commits from the remote repository and you already committed some changes to your local repository, the commits will be in a different branch. In the pull dialog this is illustrated in the image on the left. This can be useful when you want to review the changes before you want to merge them with your own changes.

\mathbf{X} Fetch (F:\dev\gc\gitextensions_4\GitExtensionsDoc)			×
<u>Hide help</u>	Pull from Remote Url http://www.apacharaka.com/apacharakaa.com/apacharakaaa.com/apacharakaa.com/apacharakaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaaaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaaaaa.com/apacharakaaa.com/apacharakaaa.com/apacharakaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	ps://github.com/gerhardol/GitExtensionsDoc.git	✓ Anage remotes
remote repository	Branch Local branch Remote branch		 ▼
d d c d d c d d c d c d c c d d c c d d c c d d c c d d c c d d c c d d c c d d c c d d c c d d c d d c d d d d c d	Merge options () 10, Merge remote branch in () 2, Rebase current branch o () Do not merge, only fetch re	n top of remote branch, creates linear history (use	with caution)
	Tag options Follow tagopt, if not specif Fetch no tag Fetch all tags	ied, fetch tags reachable from remote HEAD	
	Prune remote branches Solve conflicts	Stash changes 🗌 Auto stash	😽 Pull

When you choose to merge the remote branch after fetching the changes a branch will be created, and will be merged into your commit. Doing this creates a lot of branches and merges, making the history harder to read.

Fetch (F:\dev\gc\gitextensions_4\GitExtensionsDoc)		×
Hide help Hover to see scenario when fast forward is possible. merge commit	Pull from Remote Url	_go V Anage remotes https://github.com/gerhardol/GitExtensionsDoc.git V
f REMOTE other remote repository e	Branch Local branch Remote branch	feature/i5693-doc-3.00
d current +.LOCAL	Merge options (a) M, Merge remote bra (b) M, Rebase current brack (c) Do not merge, only f	anch on top of remote branch, creates linear history (use with caution)
b +.BASE b a a square = current branch green = new commit	Tag options Follow tagopt, if not Fetch no tag Fetch all tags	specified, fetch tags reachable from remote HEAD
9.AAA postfix for files if merge conflicts occur	Prune remote branch Solve conflicts	Stash changes Auto stash 🕹 Pull

Instead of merging the fetched commits with your local commits, you can also choose to rebase your commits on top of the fetched commits. This is illustrated on the left in the image below. A rebase will first undo your local commits (c and d), then fetch the remote commits (e) and finally recommit your local commits. When there is a merge conflict during the rebase, the rebase dialog will show.

Fetch (F:\dev\gc\gitextensions_4\GitExtensionsDoc)		×		
<u>Hide help</u>	Pull from Remote Url	_go Manage remotes https://github.com/gerhardol/GitExtensionsDoc.git		
6 LOCAL other remote repository	Branch Local branch Remote branch	feature/i5693-doc-3.00		
d current ^{4.REMOTE} e	Merge options O M, Merge remote branch into current branch @ A Rebase current branch on top of remote branch, creates linear history (use with caution) O Do not merge, only fetch remote changes			
b f.BASE a a a a a a a a a a a a a a a a a a a	Tag options Follow tagopt, if not s Fetch no tag Fetch all tags	pecified, fetch tags reachable from remote HEAD		
	Solve conflicts	S Stash changes 🗌 Auto stash 😽 Pull		

Next to the pull button there are some buttons that can be useful:

Solve	When there are merge conflicts, you can solve them by pressing this button.
con-	
flicts	
Stash	When the working dir contains uncommitted changes, you need to stash them before pulling.
changes	
Auto	Check this checkbox if you want to stash before pulling. The stash will be reapplied after pulling.
stash	
Load	This button is only available when you use PuTTY as SSH client. You can press this button to load the
SSH	key configured for the remote. If no key is set, a dialog will prompt for the key.
key	

11.5 Push changes

In the browse window you can check if there are local commits that are not pushed to a remote repository yet. In the image below the green labels mark the position of the master branch on the remote repository. The red label marks the position of the master branch on the local repository. The local repository is ahead three commits.

	3.3	
•	Working directory 🥖 5 🕂 3 🗕 4	
•	Commit index	
þ	▶ feature/i5693-doc-3.00 wip remotes	
•	fixup	
•	_go/feature/i5693-doc-3.00 Update source/plugins.rst	
Го pu	sh the changes press Push in the toolbar.	
-		
2	🔲 🚍 🖛 🖾 🔻 F:\dev\gc\gitextensions_4\GitExtensionsDoc 💌 feature/i5693	doc-3.00 🔻 😽 🔻 😭 📀 Commit

The push dialog allows you to choose the remote repository to push to. The remote repository is set to the remote of the current branch. You can choose another remote or choose a url to push to. You can also specify a branch to push.

¥ Push (F:\dev\gc\gite	extensions_4\GitExtensionsDoc\)		_		\times
Push to Remote Url	_go https://github.com/gerhardol/G		lanage rei	motes	
Push branches Push	h tags Push multiple branches				
Branch to push	feature/i5693-doc-3.00	✓ to feature/i5693-de	oc-3.00		\sim
Force With Lease	Force Push	Recursive submodule	s On-de	mand	\sim
Replace tracking	reference				
Create pull reque	st after push				
Pull			Ŷ	Push	

Tags are not pushed to the remote repository. If you want to push a tag you need to open the Tags tab in the dialog. You can choose to push a singe tag or all tags. No commits will be pushed when the Tags tab is selected, only tags.

You can not merge your changes in the remote repository. Merging must be done locally. This means that you cannot push your changes before the commits are merged locally. In practice you need to pull before you can push most of the times.

Merge Conflicts

When merging or rebasing branches or commits you can get conflicts. Git will try to resolve these, but some conflicts need to be resolved manually. Git Extensions will show warnings when there is a merge conflict in the status bar in the bottom right corner.

!There are unresolved merge conflicts 💥

12.1 Handle merge conflicts

To solve merge conflicts just click on a warning or open the Solve merge conflicts... dialog from the Commands menu. A dialog will prompt showing all conflicts.

X Resolve merge conflicts	– 🗆 X
Unresolved merge conflicts	Open in p4merge
Filename	opennipinierge
GitExtSshAskPass/SshAskPass.rc2	Start mergetool
GitExtensionsShellEx/GitExtensionsShellEx.rc	Rescan merge conflicts
	Abort merge
 The file has been changed both locally(ours) and remotely(theirs). Merge the changes.]
Local GitExtSshAskPass/SshAskPass.rc2	
Base GitExtSshAskPass/SshAskPass.rc2	
Remote GitExtSshAskPass/SshAskPass.rc2	
🔯 <u>Help</u>	

The context menu shows the actions to resolve the conflicts. Double-click on a filename will start the mergetool.

🗙 Resolve merge conflicts			_		\times
Unresolved merge conflicts			Open i	n p4mer	ne
Filename			-		
GitExtSshAskPass/SshAskPass.rc	Open in p4merge		Start r	nergetod	bl
GitExtensionsShellEx/GitExtensio	Mark conflict as solved	-1	Rescan m	erge con	flicts
	Choose local (ours) Ctrl+1		Abo	rt merge	
	Choose remote (theirs) Ctrl+2				
	Choose base Ctrl+3				
	Open local with				
	Open remote with				
	Open base with				
The file has been changed bo	Save local as				
remotely(theirs). Merge the c	Save remote as				
Local GitExtSshAskPass/Ssh/	Save base as				
Base GitExtSshAskPass/Ssh/	Open				
Remote GitExtSshAskPass/Ssh/	Open With				
🔯 <u>Help</u>	File history				

There are three kinds of conflicts:

File deleted and changed	Use modified or deleted file?
File deleted and created	Use created or deleted file?
File changed both locally and remotely	Start merge tool.

If the file is deleted in one commit and changed in another commit, a dialog will ask to keep the modified file or delete the file. When there is a conflicting change the merge tool will be started. You can configure the tool you want to use for merge conflicts. The image below shows Perforce P4Merge, a merge tool free to use for small teams.

In the merge tool you will see four versions of the same file:

Base	The latest version of the file that exist in both repositories
Local	The latest local version of the file
Remote	The latest remote version of the file
Merged	The result of the merge

Caution: When you are in the middle of a merge the file named local represents your file. When you are in the middle of a rebase the file named remote represents your file. This can be confusing, so double check if you are in doubt.

Edit View Search Help P P P P P P P P P P P P P P P P P P P	E 4 .	
fs (Ignore line ending differences) [Tab spacing: 4 [File format (Encoding: System Line en		
Base: SshAskPass_BASE_2604.rc2	dings: Windows)	
Left: SshAskPass_LOCAL_2604.rc2	Differences from base: 2	
Right: SshAskPass_REMOTE_2604.rc2	Differences from base: 0	
🔶 Merge: SshAskPass.rc2	Conflicts: 1	
/GitExtSshAskPass/SshAskPass_LOCAL_2604.rc2	./GitExtSshAskPass/SshAskPass_BASE_2604.rc2	./GitExtSshAskPass/SshAskPass_REMOTE_2604.rc2
//////////////////////////////////////	9 ////////////////////////////////////	<pre>9 ////////////////////////////////////</pre>
FILETYPE 0x2L	25 FILETYPE 0x2L	25 FILETYPE 0x2L
Schadgesr.c2 ////// Add manually edited resources here /// Version // Version // Version INFO VERSIONINFO FROUCTRESION 3,00,00,4433 FROUCTRESION 3,00,00,4433 FROUCTRESION 3,00,00,4433 FROUCTRESION 3,00,00,4433 FILEVERSION 3,01,00,0 FROUCTRESION 3,00,00,4433 FILEVERSION 4,00,00,4433 FILEVERSION 4,00,00,4434 FILEVERSION 4,00,00,4434 FILEVERSION 4,00,00,00,4434 FILEVERSION 4,00,00,00,00,00,4434 FILEVERSION 4,00,00,00,4434 FILEVERSION 4,00,00,00,00,00,00,00,00,00,00,00,00,00		

Modify Git history

A Git commit cannot be changed, the sha for the commit will change at all changes. All changes will be seen as a new commit.

- A commit can be reverted, the changes of a certain commit can be reverted and added as a new commit. Similar, a commit can be applied again (possibly to a new branch), known as cherry picking.
- The commit can be added again (and all commits that are children to the commit) as new commits and git branches can be made to point to the new commit instead.

There are 2 different cases, and consequently 2 ways to do it with git when we want to modify the history:

- · Modify the last commit of the current branch with doing an amend
- Modify an older commit with doing an interactive rebase

Note: There are 2 things to understand when working with the history with git:

- As git only creates immutable commits (sealed by the sha1), "modifying" a commit is in fact creating a new more or less similar commit.
- Consequently, the entire history of children following the changed commit will be different.

So, except if the history has not been already pushed, or if you have good reasons, it is a bad practice to change the history because you will mess the history of other developers.

13.1 Cherry pick commit

A commit can be recommitted by using the cherry pick function. This can be very useful when you want to make the same change on multiple branches. Select the commit (or range of commits) you want to cheery pick:

Working directory
Commit index
Feature/refactor dummy change 2
dummy change 1
master gitextensions/master Correct version nu
 Update build version 3.00.00 -> 3.01.00.0 (#5881)
feature/version-update Update build version 2.99.
Spd=070 (forsture (forst forst int D in the Copy to clipboard +
Mei Checkout branch
Mir Merge into current branch
Rebase current branch on
spo 🚓 Reset current branch to here
dre 🙀 Create new branch
Delete branch
📱 Diff 😤 📴 Compare 🔸
Filter files Diff with: Minor Create new tag
/ GitUI/GitUI.cs 📀 Checkout revision
🖉 GitUI/UserCo 🛷 Revert commit [,
GitUI/UserCol 🧉 Cherry pick commit
GitUl/UserCo Archive revision
- GitUI/UserCo
🔍 Navigate 🕨
₩ View ►
Open on GitHub
supdate

The confirm dialog opens:

🗙 Cherry pick con	nmit	×
Cherry pick this con	nmit:	
3a40a20abd		Choose another revision:
Review chang	les	
Author:	Henk Westhuis	
Commit date:	2 days ago (2018-12-13 19:40:34)	
Branch(es):	n/a	
Tag(s):	n/a	
Automatically cr	eate a commit	
Add commit refe	erence to commit message	Cherry pick

13.2 Revert commit

A commit cannot be deleted once it is published. If you need to undo the changes made in a commit, you need to create a new commit that undoes the changes. This is called a revert commit. A revert commit is similar to a cherry pick, but the cherry pick tries to apply the same changes as the original commit, a revert will try to reverse the changes.

• Worki	ng directory
Comn	nit index
📘 🗼 fea	ture/refactor dummy change 2
dumm	y change 1
maste	r gitextensions/master Correct version nu
 Update 	e build version 3.00.00 -> 3.01.00.0 (#5881)
featur	e/version-update Update build version 2.99.9
Me Mii Rep Sp dr P	Merge into current branch Rebase current branch on Reset current branch to here Create new branch Delete branch
Filter files	
Diff with: Mino	Create new tag
GitUl/GitUl.ce	Checkout revision
GitHU/UserCo	Revert commit

The confirm dialog opens:

🗙 Revert commit		×
Revert this commit: 3a40a20abd		
Review chang	es	
Author:	Henk Westhuis	
Commit date:	2 days ago (2018-12-13 19:40:34)	
Branch(es):	n/a	
Tag(s):	n/a	
Automatically c	reate a commit	Revert this commit

13.3 Modify the last commit

The easiest way to modify the last commit is to do an amend commit. To do that, open the commit windows and check the option "Amend commit". If the commit message text area was empty, it is now filled with the message of the last commit. You could now just update the commit message and commit or also add some more changes in the staging area to add them to the commit.

Commit	🚭 Commit message 🝷 言 Commit templates 🝷 🧏 Create branch
1 Commit & push	Fixes 5880: Reorder nodes correct
Amend Commit	
Reset all changes	
Reset unstaged changes	

13.4 Modify an older commit

To modify an older commit than the last one of the current branch, use interactive rebase.

13.4.1 Interactive rebase

First, you should create a commit containing the changes you want to add to a previous commit (or know an existing commit that contains this changes).

Then use the *rebase* feature in interactive mode on a base commit older than the one that you want to modify. See *Branches* for how to start a rebase, start an interactive rebase from the context menu or by selecting the checkbox in the rebase dialog.

🗙 Rebase		- 0	×
Hide help	Rebase current branch on top of another branch Current branch: feature/refactor		
	Rebase on 7b342f925f740a74957a5570afc58a20d 🗸	Rebase	
f other f.LOCAL C	□ Interactive Repase □ Preserve Merges ○ Autosquash ○ Autosquash □ Specific range From (exc.) □ □ feature/refactor	Solve conflicts	
d current f other	Commits to re-apply:	Add files	
• REMOTE	Name Subject Author Date Status	Commit	
b +.BASE b		Continue rebase	
		Skip this commit	
•		Abort	
square = green = new commit			
+.AAA merge conflicts occur			

You will be prompted by a text editor displaying all the commits that will be rebased

```
🗙 F:/dev/gc/gitextensions/.git/rebase-merge/git-rebase-todo
                                                                                               \times
pick 9d7a081f5 dummy change 1
1
    pick 4feed7716 dummy change 2
2
З
    # Rebase 3c2ac977b..4feed7716 onto 3c2ac977b (2 commands)
4
    #
5
    # Commands:
6
    # p, pick <commit> = use commit
8
    # r, reword <commit> = use commit, but edit the commit message
    # e, edit <commit> = use commit, but stop for amending
    # s, squash <commit> = use commit, but meld into previous commit
10
    # f, fixup <commit> = like "squash", but discard this commit's log message
11
    # x, exec <command> = run command (the rest of the line) using shell
12
    # b, break = stop here (continue rebase later with 'git rebase --continue')
13
    # d, drop <commit> = remove commit
14
    # 1, label <label> = label current HEAD with a name
15
    # t, reset <label> = reset HEAD to a label
16
17
    # m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
              create a merge commit using the original merge commit's
   # .
18
              message (or the oneline, if no original merge commit was
    # .
19
    # .
              specified). Use -c <commit> to reword the commit message.
20
    #
21
    # These lines can be re-ordered; they are executed from top to bottom.
22
    #
23
    # If you remove a line here THAT COMMIT WILL BE LOST.
24
25
    #
    # However, if you remove everything, the rebase will be aborted.
26
27
    # Note that empty commits are commented out
28
29
```

You could have a look to this _documentation: https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History to better understand all the possibilities offered.

The options offered are :

- reorder the lines to reorder the commits,
- delete a line to throw away a commit and the changes introduced by the commit,
- write r or reword in front of a commit to rewrite the commit message,
- write *f* or *fixup* in front of a commit to meld the commit with the previous commit and with keeping the commit message of the first commit,
- write *s* or *squash* in front of a commit to meld the commit with the previous commit and with rewriting the commit message.

Often, we will use interactive rebase to move the line and squash or fixup commits to modify the history.

Once we did the changes, save and close the editor to let git do the rebase.

13.4.2 Using autosquash rebase feature

There is an option to facilitate the use of the interactive rebase when you know, at the moment of doing a commit that the changes introduced by this commit should have been made in an older commit (the case of a *fixup* or *squash*).

In this case, you should create a commit containing the changes you want to add to a previous commit and use the *Advanced* menu to:

- create a *fixup* commit
- create a *squash* commit

Right click on the commit in the history, you know that you want to "modify".

And choose the suitable option...

	ting directory			
	mit index ature/refactor dummy change 2		Gerhard Olsson	2018-12-1
dt 🕞	Copy to clipboard	•	Gerhard Olsson	2018-12-15
n 10	Merge into current branch	nu	Gerhard Olsson	2018-12-1
U	-		🮉 Gerhard Olsson	2018-12-1
📕 🖣 🗲		99.90 -> .	🔛 Gerhard Olsson	2018-12-1
s 🖁	Create new branch		Henk Westhuis	2018-12-1
M	Compare	▶ 859	·	2018-12-1
		_	Henk Westhuis	2018-12-1
• Re	Checkout revision	ay witho.	📃 Henk Westhuis	2018-12-1
📕 S 🦨	Revert commit	tExa •••	• 🏭 Henk Westhuis	2018-12-1
d 6	Cherry pick commit	nduct	Drew Noakes	2018-12-1
	Archive revision			
🗟 Diff 😤 💼	Advanced	Ed	lit commit	
igithub 🧧	Navigate	▶ Re	word commit	
📲 Indget 👫	View	·	eate a fixup commit	Ctrl+X
- Build G	Open on GitHub		reate a squash commit	
External GitCom	supdate	Ge	et help on how to use thes	e reatures

If you have not the changes prior to open the dialog, do them now.

GitExtensions will open the commit window with an already filled commit message containing the needed information to find the commit to "modify". Do not change the commit message and commit all the changes needed.

Then process to the interactive rebase, like describe in the previous paragraph but with enabling the option Autosquash.

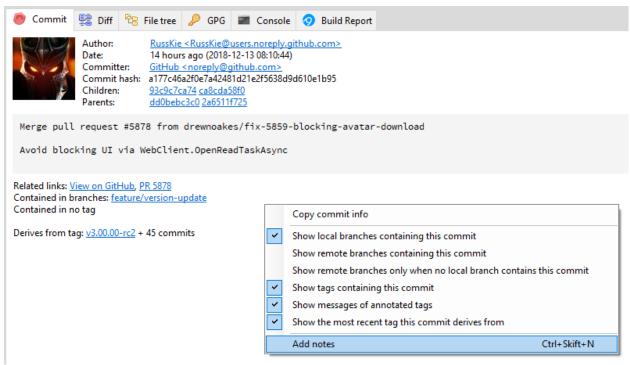
X Rebase			_		×
<u>Hide help</u>	Rebase current branch on top of another branch Current branch: feature/refactor				
	Rebase on 3a40a20abd8be37865635468ff965a9f1 ~		R	ebase	
f other d current	🗹 Interactive Rebase 📋 Preserve Merges 🗹 Autosquash 🗹 Auto stash				
F.LOCAL	Specific range From (exc.)		Solve	conflicts	
	Commits to re-apply:		Ad		
e e	Name Subject Author Date Status	15	Co	mmit	
b +.BASE b			Conti	nue rebase	2
			Skip tł		t
• •			1	Abort	
square = green = new commit					
 AAA postfix for files if merge conflicts occur 					

Launch the rebase by clicking on Rebase.

The interactive rebase will process the same way but with a major difference! When enabling the *Autosquash* option, git will automatically reorder the commits lines and write the good actions in front of the commits when it will open the text editor. You normally have just to close the editor (except if you want to do additional changes). And let git do the rebase.

Notes

Notes can be added to a commit. Notes will be stored separately and will not be pushed. To add a new note choose add notes in the context menu of the commit information box.



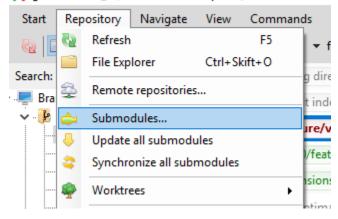
The editor that has been configured in the settings dialog will be used to enter or edit the notes. The Git Extensions editor is advised.

```
Х
K F:/dev/gc/gitextensions/.git/worktrees/gitextensions_4/NOTES_EDITMSG
                                                                             _
I
    #
   # Write/edit the notes for the following object:
   # commit a177c46a2f0e7a42481d21e2f5638d9d610e1b95
   # gpg: Signature made Thu Dec 13 08:10:44 2018
                      using RSA key 4AEE18F83AFDEB23
   # gpg:
   # gpg: Can't check signature: No public key
   # Merge: dd0bebc3c 2a6511f72
   # Author: RussKie <RussKie@users.noreply.github.com>
   # Date: Thu Dec 13 18:10:44 2018 +1100
   #
   #
         Merge pull request #5878 from drewnoakes/fix-5859-blocking-avatar-download
   #
14
15 #
         Avoid blocking UI via WebClient.OpenReadTaskAsync
16 #
  18 # GitUI/UserControls/AvatarControl.cs | 2 ++
19 # Plugins/Gource/GourceStart.cs
                                      3 +++
   # 3 files changed, 24 insertions(+)
```

Submodules

Large projects can be split into smaller parts using submodules. A submodule contains the name, url and revision of another repository. To create a submodule in an existing git repository you need to add a link to another repository containing the files of the submodule.

X gitextensions_4 (feature/version-update) - Git Extensions



15.1 Manage submodules

The current state of the submodules can be viewed with the Manage submodules function. All submodules are shown in the list on the left.

🗙 Submodules					_		×
Name	Status	Details					
Externals/Git.hub	Up-to-date	Name	Externals/G	it.hub			
Externals/ICSharpC	Up-to-date	Remote path	https://gith	ub.com/gitexten	sions/Git.hu	b.git	
Externals/NBug	Up-to-date	Local path	Externals/G	-			
Externals/conemu-i	Up-to-date						
GitExtensionsDoc	Modified	Commit	7df924a6a2	0fb608ff38ffbf95	d41b2c9e665	alc	
		Branch	remotes/or	igin/HEAD			4
		Status	Up-to-date	:			
Add submodul	e	😂 Sync	hronize	😓 Update		Remov	e

Add sub-	Add a new submodule to the repository
module	
Synchro-	Synchronizes the remote URL configuration setting to the value specified in .gitmodules for the
nize	selected submodule.
Initialize	Initialize the selected submodules, i.e. register each submodule name and url found in .gitmodules
	into .git/config. The submodule will also be updated.
Update	Update the registered submodules, i.e. clone missing submodules and checkout the commit specified
	in the index of the containing repository.
Remove	Remove the submodule from the repository

15.2 Add submodule

To add a new submodule choose Add submodule in the Manage submodules dialog.

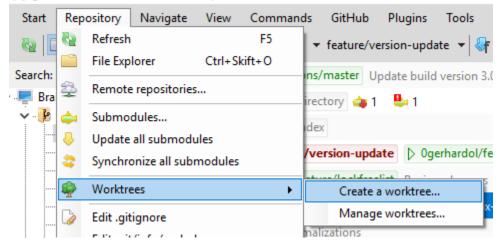
Add submodule		x
Path to submodule	lttps://github.com/gerhardol/NBug.git 🗸	Browse
Local path		
Branch	~	
Force		Add

Path to submodule	Path to the remote repository to use as submodule.
Local path	Local path to this submodule, relative to the root of the current repository.
Branch	Branch to track.

Worktrees

Git Extensions support Git worktrees: Multiple checked out working directories can share local branches. For more information see the Git documentation: https://git-scm.com/docs/git-worktree

X gitextensions_4 (feature/version-update) - Git Extensions



Maintenance

In this chapter some of the functions to maintain a repository are discussed.

17.1 Compress Git database

Git will create a lot of files. You can run the Compress git database to pack all small files building up a repository into one big file. Git will also garbage collect all unused objects that are older then 15 days. When a database is fragmented into a many small files compressing the database can increase performance.

🗙 gitex	tensi	ons_4 (fe	ature/versior	n-update) - Git Ext	tensi	ons				
Start	Rep	ository	Navigate	View	Comm	ands	; Gi	tHub	Plugi	ns Too	ls H
@	25	Refresh File Exp		Ctrl+Sk	F5		▼ fea	ture/v	ersion-u	update 🔻	🦣 🕶
Search:						9	direct	ory 🤞	a 1 📲	1	
📲 Bra	2	Remote	e repositories			t	index				
🖌 - 🏓	4	Submo	dules				re/ver	sion-u	pdate	▷ 0gerh	ardol/f
		Update all submodules Synchronize all submodules							-	Review c	
	-	Worktre				۱s ۱	ions/r	naster	Merge	e pull requ	uest #58
	۵	Edit .git	ignore				ptimalizations SynchronizedCollection with cheaper				
		Edit .git	/info/exclud	e		- 13	-				
		Edit .git	attributes							npath De	-
		Edit .ma				а	kes/co	de-of	-condu	ct Add c	ode of
		Edit .git				а	akes/fix-5859-blocking-avatar-downl			downlo	
		Sparse	Working Cop	у		a	kes/m	aster	Merge	pull reque	est #589
	ø	Git mai	ntenance			•	¢ (Compr	ess git o	database	le
		Reposit	ory settings			1	M F	lecove	er lost ol	bjects	
	2	Close (g	go to Dashbo	oard) C	trl+W				index.lo		
		286-thro			Cat 's	thy	🧼 E	ait .gr	t/config	d labe to	1000

17.2 Recover lost objects

Normally Git will not delete files right away when you remove something from your repository. The reason for this is that you can restore deleted items if you need to. Git will delete removed items when they are older then 15 days and you run Compress git database.

Commits without branches or tags can be shown with Git reflog https://git-scm.com/docs/git-reflog The easiest way to view the commits is to show Git reflog in the revision graph:

X	gitextensions_4	(feature/version-u	pdate) - Git Extensions
---	-----------------	--------------------	-------------------------

Start Repository Navigate	Vie	w Commands GitHub	Plugins	Tools	Help	
🍓 🔲 🖃 🖛 🔂 🔻 F:\de	~	Show all branches		Ctrl+S	Skift+A	Łe
		Show current branch only		Ctrl+S	Skift+U	- 81
Search:		Show filtered branches		Ctrl+S	Skift+T).0
📲 Branches (112) 👘 👘						_
🗸 🥑 feature	~	Show remote branches		Ctrl+S	Skift+R	
····· 🐉 version-updi	~	Show reflog references				
2.00-version						

The reflog commits are listed as gray:

🗙 gitextens	sions_4 (master) - Git Extensions
Start Re	pository Navigate View Commands GitHub Plu
R 🗖	📄 📼 🔻 🖾 🔻 F:\dev\gc\gitextensions_4 💌 master 👻 😽
•	Working directory 🛶 1 🛛 😓 1
	Commit index
¢	▶ master ▷ gitextensions/master Update build version
	feature/version-update Ogerhardol/feature/version-update
	Update build version 2.99.90 -> 3.01.00.a0
9	Update build version 2.99.90 -> 3.01.00.a0
	and r 970 / fasture / lockfreelist Deview shares
T	spdr870/feature/lockfreelist Review changes

GE also supports the previous way to show you all dangling objects and will allow you to review and recover them. If you accidentally deleted a commit you can try to recover it using the Recover lost objects function.

🗙 gitex	tensi	ons_4 (m	aster) - Git E	xtensions	5					
Start	Rep	ository	Navigate	View	Comman	ds	GitHub	Plugins	Tools	Hel
ି କରୁ 🛛	6	Refresh	n		F5	- 1	master 🔻	🛛 😽 🗝 🏠	🕝 Con	nmit
		File Exp	olorer	Ctrl+S	cift+O					
I.	\$	Remote	e repositorie	s						
<u>–</u>	4	Submo	dules			data	build you	sion 3.00.0	0 . 2 0	1.00
ц н	₽.	Update	all submod	ules						
	\$	Synchronize all submodules Worktrees				ture/	version-u	pdate Upo	late build	vers
2	\$									
1		Edit .git	tignore							
		Edit .git	t/info/exclud	de		nges				
		Edit .git	tattributes			es/fix	-5859-blo	cking-avat	ar-downl	oad
\mathbf{D}		Edit .m	ailmap							
		Edit .git	treview			aper a	array with	out locking		
		Sparse	Working Co	ру		e Try	GetExactP	ath, to rem	ove perfo	rmai
	ø?	Git mai	intenance		•	\$	Compre	ess git datal	oase	
	£639	Reposit	tory settings				Recover	lost object	s	llie
17	25	Close (go to Dashb	oard) C	trl+W	۵.	Delete i	ndex.lock		
\mathbf{N}	_	aitexten	isions/releas	e/3.00 F	ives 5880+ R	eorde	Edit .git			

X	Verify database				$ \square$ X			
2 v ot	weeks are removed	ferenced objects tha d when cleaning up v deleted when you	the database. All	 Show commits and tags Show other objects Do not consider commits that are referenced only by an entry in a reflog to be reachable. 				
Co		Iditional operations	l press Recover button	 Print out objects that exist but that are nodes. Check not just objects in GIT_OBJECT, but also the ones found in alternate of 	en't readable from any of the reference _DIRECTORY (\$GIT_DIR/objects), bject pools.			
	Date	Туре	Author	Hash	Parent(s) hashs			
	2018-12-13 21:58	dangling commit	Gerhard Olsson	cc75167da4417ba3a1cf2bfcc6eaf0de45bd03cc	a177c46a2f0e7a42481d21e2f5638d9			
	2018-12-13 21:49	dangling commit	Gerhard Olsson	661b2fb7e5f2f93cf88354a11246dfb287a5ae1b	a177c46a2f0e7a42481d21e2f5638d9			
	2018-12-11 16:27	dangling commit	Henk Westhuis	6b283c5999b802e56c05c8d6c88255da588e18df	022e8e1643316e535b9de32b6a62e9			
	2018-12-11 00:09	dangling commit	Gerhard Olsson	47377b365317afbbfa907adc19a3f154f0b23fba	034fc2528ac8d1e18be476dafe74117			
<			1	1	>			
	Remove all o	dangling objects			Save objects to .git/lost-found			
	Delete all LOST	_AND_FOUND tags		Recover selected objects Cancel				

Git Extensions also is able to tag all lost objects. Doing this will make all lost objects visible again making it very easy to locate the commit(s) you would like to recover.

17.3 Fix user names

When someone accidentally committed using a wrong username this can be fixed using the Edit .mailmap function. Git will use the username for an email address when it is set in the .mailmap file.

	🗙 Edit .mailmap	- 🗆 ×
1	Proper Name <proper <proper@email.xx="" name=""> Commit Name <commit@email.xx></commit@email.xx></proper>	Edit the mailmap. This file is meant to correct usernames. Example: Henk Westhuis < Henk@.(none)> Henk Westhuis < henk_westhuis@hotmail.com> For more information run command "git help shortlog"
		Save

For more information, see https://git-scm.com/docs/git-check-mailmap.

17.4 Ignore files

Git will track all files that are in the working directory. Normally you do not want to exclude all files that are created by the compiler. You can add files that should be ignored to the .gitignore file. You can use wildcards and regular expressions. All entries are case sensitive. The button Add default ignores will add files that should be ignored when using Visual Studio.

X	Edit .gitignore				_		×
1 2 3 4 5 6 7 8 9	<pre>build *.orig tx.exe GitExtensions.settin source/locale/.doctr /source/_pycache_/ /source/extensions/_ *.pyc</pre>	ees/	*.suo*.bak*.cache*.ilk*.log[Bb]in[Dd]e	.bug*/*.sbrobj/[Rr]elease	₽*/_ReSh	arper*/
	Add default ignores	Add pattern	Example ignore patterns	Generate a cus		<u>ore file fo</u> Save	or git

A short overview of the syntax:

#	Lines started with # are handled as comments
!	Lines started with ! are exclude patterns
[Dd]	Characters inside [] means that 1 of the characters must match
*	Wildcard
/	A leading slash matches the beginning of the pathname; for example, /*.c matches cat-file.c but not
	mozilla-sha1/sha1.c
/	If the pattern ends with a slash, it is removed for the purpose of the following description, but it would only
	find a match with a directory. In other words, foo/ will match a directory foo and paths underneath it, but
	will not match a regular file or a symbolic link foo (this is consistent with the way how pathspec works in
	general in git).

For more detailed information.

Translations

18.1 Change language

Choose language
You can change the language at any time in the settings dialog
French
French
German
Polish
Russian
Nutrie Chinese

In the settings dialog the language can be chosen.

18.2 Translate Git Extensions

More information in the Git Extensions wiki: https://github.com/gitextensions/gitextensions/wiki/Translations Translations are done on Transifex: https://www.transifex.com/git-extensions/git-extensions/

Windows Explorer

The common commands can be started from Windows Explorer using the shell extensions. This option is only available when Shell Extensions are installed.

	GitEx Browse		
0	GitEx Commit		
58	Git Extensions	₽.	Pull
			Push
		1	View stash
		++++	View changes
		-	Checkout branch
		۲	Checkout revision
		9	Create branch
		++++	Open with difftool
		•	File history
		2	Reset file changes
		+	Add files
			Apply patch
		0	Settings

If the folder do not have a Git repository, you can clone.

8	GitEx Clone		
8	GitEx Create new repository		
<u>58</u>	Git Extensions	0	Settings

Visual Studio

20.1 Menu

Almost all function can be started from the GitExt menu in Visual Studio.

GitExtensions - Microsoft Visual Studio								
File	Edit	View	GitE	xt	Project	Build	Debug	
	- 0	わ-	X	Bro	owse		i i	
허	Calution	F	.	Clo	one reposit	ory		
olbo	Solution		★	Cre	eate new re	pository		
×	GO	û 🛱	0	Co	mmit			
Data	Search S	olution	₽	Pu	II			
Toolbox Data Sources		ution 'G		Pu	sh			
Irces		.nuget External		Sta	sh			
		Plugins	÷.	Re	set File Cha	inges		
	Þ 💼				anage remo	-		
	۵ 🛑	UnitTes	0		it .gitignore			
		VS	-		ply patch			
		GitCom GitExte			rmat patch			
	- ⊵	> Prop			e history			
	Þ	∎•∎ Refe	12		w changes			
		🔁 app.			me	,		
		🖶 app.	-		ine d file			
	⊳	C# Proc						
		GitExtU			eckout bra			
	⊳	🏓 Prop	_		eate branch	1		
	⊳	∎•∎ Refe	-		erge			
		GitU			base			
	₽	🛑 Linq 🖓 app.			ve merge (conflicts		
	⊳	C# Argi	-	Ch	erry pick			
	⊳	C# Argi		Git	bash			
	Þ	C# Arra	_	Set	tings			
		C# Bina C# Clin	•		out Git Ext	ensions		
		CIDI	Juaru	Uui.				

20.2 Toolbar

A Git Extensions toolbar allows you to perform the most common actions. The buttons can be customized, same functions as in the menu.



The current branch name can be shown in the commit button.

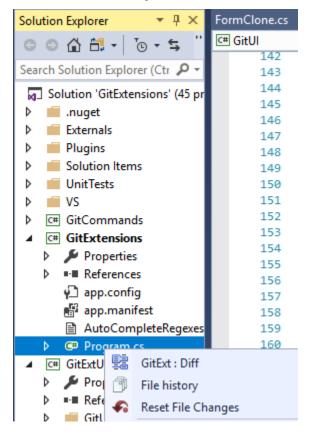
🗙 Settings - Appearance

Type to find	Settings source: Global for all repositories
✓ X Git Extensions	
✓ 🤧 Appearance	General
	Show relative date instead of full date
Aa Fonts	
🔗 Revision links	Show current branch in Visual Studio
Build server integration	Auto scale user interface when high DPI is used
Scripts	Transitation (Transitation)
🖾 Hotkeys	Truncate long filenames None ~
Shall advantage	

20.3 Context menu

Options in the context menu on files and in Solution Explorer:

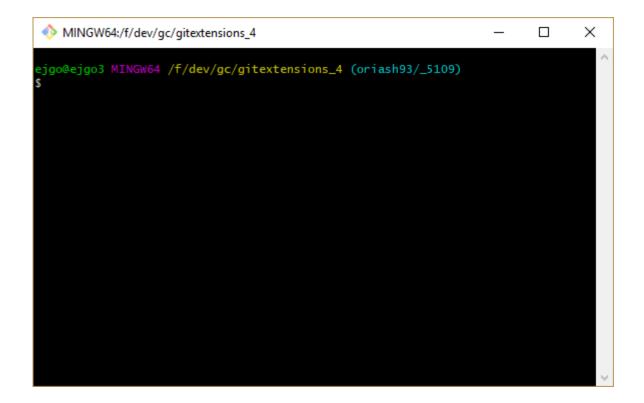
- Diff changes to the commit index
- View the file history by choosing the 'File history' option.
- Reset the file changes to the last committed revision.



Command line

21.1 Git Extensions command line

Most features can be started from the command line. It is recommended to add gitex.cmd to the path when using from the command line. It is typically stored in the C:\Program Files (x86)\GitExtensions folder.



Appendix

22.1 Git Cheat Sheet

Action	Command
Create new repository	\$ git init
Create shared repository	\$ git initbareshared=all
Clone repository	\$ git clone c:/demo1 c:/demo2
Checkout branch	\$ git checkout <name></name>
Create branch	\$ git branch <name></name>
Delete branch	\$ git branch -d <name></name>
Merge branch (from the branch to merge into):	\$ git merge PDC
Solve conflicts (add -tool=kdiff3 if no mergetool is specified)	<i>\$ git mergetool \$ git commit</i>
Create tag	\$ git tag <name></name>
Add files/changes (. for all files)	\$ git add .
Commit added files/changes (-amend to amend to last	\$ git commit –m "Enter commit message"
commit)	
Discard changes	\$ git reset –hard
Create patch (-M = detect renames $-C$ = detect copies)	format-patch -M -C origin
Apply patch without merging	<pre>\$ git apply c:/patch/01-emp.patch</pre>
Merge patch	\$ git am3waysignoff c:/patch/01-emp.patch
Solve conflicts (add -tool=kdiff3 if no mergetool is specified)	\$ git mergetool \$ git am3wayresolved
Stash changes	\$ git stash
Apply stashed changes	\$ git stash apply
Pull changes (add –rebase to rebase instead of merge)	\$ git pull c:/demo1 master
Solve conflicts (add -tool=kdiff3 if no mergetool is specified)	
- r /	\$ git mergetool
	\$ git commit
22.1. Git Cheat Sheet Push changes (in branch \$ git push c:/demo1 master master: <new>)</new>	\$ git push c:/demo1
Blame	\$ git blame –M –w <filename></filename>
Help	\$ ait < command>_help

Here are some default names used by Git.

Default names	
master	default branch
origin	default upstream repository
HEAD	current branch
HEAD^	parent of HEAD
HEAD~4	the great-great grandparent of HEAD

Plugins

Git Extensions has a possibility to add functionality in external plugins. Some are distributed with the main program.

Most plugins has settings in *Settings*. Most plugins also have UI forms accessible from the main menu in *Browse Repository*.

This list is incomplete.

23.1 Auto compile SubModules

This plugin proposes (confirmation required) that you automatically build submodules after they are updated via the GitExtensions Update submodules command.

Enabled

Enter true to enable the plugin, or false to disable.

Path to msbuild.exe

Enter the path to the msbuild.exe executable.

msbuild.exe arguments

Enter any arguments to msbuild.

23.2 Bitbucket Server

For repositories is hosted on Atlassian Bitbucket Server, the plugin cannot be used for bitbucket.org. For more information see: https://www.atlassian.com/software/bitbucket/server

This plugin will enable you to view and create pull requests for Bitbucket.

Bitbucket Username

The username required to access Bitbucket.

Bitbucket Password

The password required to access Bitbucket.

Specify the base URL to Bitbucket

The URL from which you will access Bitbucket.

Disable SSL verification

Check this option if you do not require SSL verification to access Bitbucket Server.

23.3 Create local tracking branches

This plugin will create local tracking branches for all branches on a remote repository. The remote repository is specified when the plugin is run.

23.4 Delete obsolete branches

This plugin allows you to delete obsolete branches i.e. those branches that are fully merged to another branch. It will display a list of obsolete branches for review before deletion.

```
Delete obsolete branches older than (days)
Select branches created greater than the specified number of days ago.
```

```
Branch where all branches should be merged
The name of the branch where a branch must have been merged into to be considered obsolete.
```

23.5 Find large files

Finds large files in the repository and allows you to delete them.

```
Find large files bigger than (Mb)
Specify what size is considered a 'large' file.
```

23.6 Gerrit Code Review

The Gerrit plugin provides integration with Gerrit for GitExtensions. This plugin has been based on the git-review tool.

For more information see: https://www.gerritcodereview.com/

23.7 GitHub

This plugin will create an OAuth token so that some common GitHub actions can be integrated with Git Extensions.

For more information see: https://github.com/

OAuth Token

The token generated and retrieved from GitHub.

23.8 GitFlow

This plugin permit to manage your _branching model: http://nvie.com/posts/a-successful-git-branching-model/ with _GitFlow: https://github.com/nvie/gitflow in GitExtension

You should have GitFlow installed to use this plugin.

The GitFlow plugin permit to : - init gitflow in your git repository - create your feature, hotfix, release or support branch - manage (pull, publish or finish) your existing gitflow branches

23.9 Gource

Gource is a software version control visualization tool.

For more information see: http://gource.io/

```
Path to "gource"
```

Enter the path to the gource software.

Arguments

Enter any arguments to gource.

23.10 Impact Graph

This plugin shows in a graphical format the number of commits and counts of changed lines in the repository performed by each person who has committed a change.

23.11 Jira Commit Hint

Provides hints for Atlassian Jira issues in the commit form.

23.12 Periodic background fetch

This plugin keeps your remote tracking branches up-to-date automatically by fetching periodically.

Arguments of git command to run

Enter the git command and its arguments into the edit box. The default command is fetch --all, which will fetch all branches from all remotes. You can modify the command if you would prefer, for example, to fetch only a specific remote, e.g. fetch upstream.

Fetch every (seconds)

Enter the number of seconds to wait between each fetch. Enter 0 to disable this plugin.

Refresh view after fetch

If checked, the commit log and branch labels will be refreshed after the fetch. If you are browsing the commit log and comparing revisions you may wish to disable the refresh to avoid unexpected changes to the commit log.

Fetch all submodules

If checked, also perform git fetch --all recursively on all configured submodules as part of the periodic background fetch.

23.13 Proxy Switcher

This plugin can set/unset the value for the http.proxy git config file key as per the settings entered here.

Username

The user name needed to access the proxy.

Password The password attached to the username.

HttpProxy Proxy Server URL.

HttpProxyPort

Proxy Server port number.

23.14 Release Notes Generator

This plugin will generate 'release notes'. This involves summarising all commits between the specified from and to commit expressions when the plugin is started. This output can be copied to the clipboard in various formats.

23.15 Statistics

This plugin provides various statistics (and a pie chart) about the current Git repository. For example, number of commits by author, lines of code per language.

Code files

Specifies extensions of files that are considered code files.

Directories to ignore (EndsWith)

Ignore these directories when calculating statistics.

Ignore submodules

Ignore submodules when calculating statistics (true/false).