Git Extensions Documentation

Release 4.1

Contributors

May 03, 2023

Contents

1	Git Extensions	1
	1.1 Features 1.2 Video tutorials	1
	13 Links	2
2	Installation 2.1 Windows installer 2.2 Portable 2.3 Settings	- 3 6 6
3	Dashboard 3.1 Create new repository 3.2 Open repository 3.3 Clone repository 3.4 Clone Github repository	9 11 11 11 12
4	Browse Repository 4.1 Main toolbar 4.2 Revision graph 4.2.1 Search 4.2.1.1 Quick search in history 4.2.2 Navigation 4.2.3 Filter 4.2.3.1 Filter history	13 13 15 15 16 16 16
	4.2.3.2 Reflog 4.3 Tabs	18 19 19 19 20 20 20
5	Commit 5.1 Commit changes	21 21

	5.2	5.1.1 5.1.2 5.1.3 5.1.4 Amend	Staging changes	
	D			20
0	Bran	Create h	aron ah	29
	0.1	6 1 1	Orphan branches	31
	62	Checkou	ut branch	31
	6.3	Merge h	pranches	
	0.0	6.3.1	Advanced Merge Options	
	6.4	Rebase I	branch	
	6.5	Interacti	ive rebase	
	6.6	Squash		
	6.7	Delete b	pranch	
_				
7	Tag	G		38
	7.1	Create ta	ag	
	1.2	Delete ta	ag	
8	Rem	otes		40
Ŭ	8.1	Manage	e remote repositories	
	8.2	Git Cred	dential Manager	42
	8.3	Create S	SSH key	
		8.3.1	PuTTY and github	
		8.3.2	OpenSSH and github	
	8.4	Pull cha	anges	
	8.5	Push cha	anges	
0	G 1			
9	Subn	nodules		49
9	Subn 9.1	nodules Manage	e submodules	49
9	Subn 9.1 9.2	nodules Manage Add sub	e submodules	49
9 10	Subn 9.1 9.2 Worl	nodules Manage Add sub	e submodules	49
9 10 11	Subn 9.1 9.2 Worl Stash	nodules Manage Add sub ktrees	e submodules	49
9 10 11	Subn 9.1 9.2 Worl Stash 11.1	nodules Manage Add sub ktrees Revisior	e submodules	49
9 10 11	Subn 9.1 9.2 Worl Stash 11.1 11.2	nodules Manage Add sub ktrees Revisior Left pan	e submodules	49
9 10 11	Subn 9.1 9.2 Worl Stash 11.1 11.2	nodules Manage Add sub ktrees Revisior Left pan	e submodules	49
9 10 11 12	Subm 9.1 9.2 Worl Stash 11.1 11.2 Modi	nodules Manage Add sub ctrees Revisior Left pan	e submodules	49
9 10 11 12	Subn 9.1 9.2 Work Stask 11.1 11.2 Modi 12.1	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p	e submodules	49
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modifu	e submodules	49 50 51 53 54 55 56 56 58 58 58
9 10 11 12	Subm 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modify	e submodules	49
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ctrees Revisior Left pan ify Git hi Cherry p Revert c Modify 12 4 1	e submodules	49
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git his Cherry p Revert c Modify 12.4.1	e submodules	49
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modify 12.4.1 12.4.2 12.4.3	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 56 56 56 56 57 56 57 58 59 59 60 61 62 63
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modify Modify 12.4.1 12.4.2 12.4.3 12.4.4	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 57 56 57 58 59 60 61 63 63
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modify Modify 12.4.1 12.4.2 12.4.3 12.4.4 Merge C	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 57 56 57 56 57 58 59 60 61 63 63 63 64
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git hi Cherry p Revert c Modify 12.4.1 12.4.2 12.4.3 12.4.4 Merge C 12.5.1	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 56 56 56 56 57 56 57 58 59 60 61 63 63 63 63 63 63 64 65 65 65 65 65
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ctrees Revisior Left pan ify Git hi Cherry p Revert c Modify 12.4.1 12.4.2 12.4.3 12.4.4 Merge C 12.5.1	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 57 56 57 56 57 58 59 60 61 62 63 63 63 63 64 65 65 65 65
9 10 11 12	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ktrees Revisior Left pan ify Git his Cherry p Revert c Modify 12.4.1 12.4.2 12.4.3 12.4.4 Merge C 12.5.1	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 57 56 57 56 57 56 57 56 57 56 57 56 57 56 57 58 59 59 50 51 52 53 54 56 57 58 59 50 51 52 53 54 55 56 57 58 59 51 52 53 54
9 10 11 12 13	Subn 9.1 9.2 Worl Stash 11.1 11.2 Modi 12.1 12.2 12.3 12.4	nodules Manage Add sub ctrees Revisior Left pan ify Git hi Cherry p Revert c Modify 12.4.1 12.4.2 12.4.3 12.4.4 Merge C 12.5.1	e submodules	49 50 51 53 54 55 56 56 56 56 56 56 56 56 57 56 57 56 57 56 57 56 57 58 59 59 60 61 63 63 63 63 64 65 65 65 66 67 68 69 69 69 69 61 62 63 64 65 67 68 69 69 69 69 69

14 INULCS	14	Notes
-----------	----	-------

14	Notes	5		71
15	File h	istory		73
	15.1	Commit		74
	15.2	Diff		74
	15.3	View .		75
	15.4	Blame.		76
16	Main	tenance		77
	16.1	Compre	ss Git database	77
	16.2	Recover	lost objects	78
	16.3	Fix user	names	80
	16.4	Ignore f	lles	81
17	Settin	igs		82
	17.1	Git Exte	ensions	82
	- /	17.1.1	General	84
		17.1.2	Appearance	85
		17.1.3	Sorting	87
		17.1.4	Colors	87
		17.1.5	Fonts	88
		17.1.6	Console style	88
		17.1.7	Revision Links	88
		17.1.8	Build server integration	91
		17.1.9	Scripts	92
		17.1.10	Hotkeys	93
		17.1.11	Shell extension	94
		17.1.12	Advanced	94
		17.1.13	Confirmations	95
		17.1.14	Detailed	97
		17.1.15	Browse repository window	97
		17.1.16	Commit dialog	97
		17.1.17	Diff viewer	98
		17.1.18	Blame viewer	103
		17.1.19	SSH	104
	17.2	Git		104
		17.2.1	Paths	104
		17.2.2	Config	105
		17.2.3	Advanced	106
	17.3	Plugins		106
10	Dhuai	20		107
19	Plugi	IIS Dundlad		107
	10.1	10 1 1	Auto compile submodules	107
		10.1.1	Rithucket Server	107
		10.1.2	Create local tracking branches	107
		10.1.3		108
		18.1.4	Find large files	108
		18.1.5	GitHub	108
		18 1 7	GitFlow	108
		1818	Gource	109
		1819	Impact Graph	109
		18.1.10	Periodic background fetch	109
		18.1.11	Plugin Manager	109
		18.1.12	Proxy Switcher	109
			•	

	18.1.13 Release Notes Generator 11 18.1.14 Statistics 11 18.2 Third party extensions 11 18.2.1 Gerrit Code Review 11 18.2.2 Jira Commit Hint 11	10 10 10 10
19	GitHub1119.1Clone Github repository1119.2View pull requests1119.3Create pull requests1119.4Add upstream remote1119.5Settings11	1 3 13 14 15 16
20 21	Translations 11 20.1 Change language 11 20.2 Translate Git Extensions 11 Windows Explorer 11	17 17 18
22	Other tools 12 22.1 Visual Studio Code 12 22.2 Visual Studio 12 22.2.1 Menu 12 22.2.2 Toolbar 12 22.2.3 Context menu 12	21 21 21 21 22 22 23
23	Command line 12 23.1 Git Extensions command line 12	24 24
24	Appendix 12 24.1 Git Cheat Sheet 12	27 28

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

- Feature rich user interface for Git
- 32bit and 64bit support
- Windows Explorer integration for Git
- Visual Studio extension (2015-2022)

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 (made for older Git Extensions versions).

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

5. Handle merge conflicts

1.3 Links

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

- Download page: https://github.com/gitextensions/gitextensions/releases
- 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

Installation

2.1 Windows installer

This section only covers Git Extensions installation, you will need Git for Windows.

The Git Extensions installer can be found on GitHub.



Start installation.



Destination folder.





Allow telemetry to allow the app to collect anonymous data to improve the application.



2.2 Portable

Git Extensions is also distributed as a portable .zip file, that only requires unpacking to a new directory (migrate settings files and theme manually). Some features like *Windows Explorer* is not available with this package.

2.3 Settings

Git must be installed prior to starting Git Extensions:

🚫 Git	Extensions.exe	\times
\otimes	The Git executable could not be located on your system.	
	\rightarrow Find git	
	\rightarrow Install git	
	Cancel	

X Choose language			_	\times
Choose your language You can change the language at any	time in the settings dialog			
English	Japanese	Russian		
★**				
Simplified Chinese	Spanish	Spanish (Argentina)		
*	C *			
Traditional Chinese	Turkish			

First selection is language (depends on the installed languages):

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.



For further information see Settings.

chapter $\mathbf{3}$

Dashboard

The dashboard contains the most common tasks, recently opened repositories and categories (favourites). Categories can be added, grouped in the repository context menu.

X Git Extension	5				-	×
Start Dashbo	ard GitHub Tools Help					
	GITXEXTENSIONS	Recent repositories				
	🐻 Create new repository	gitext			1	
	👩 Open repository	Recent repositories		Actions		
	Clone repository Clone GitHub repository	C\de-\gc\gitextensionsdac\ feature/prepare-41	C\dev\gc\gitextensions\ tmp/ge9-7		Γ.	
		C:\der\gc\gitextensions_4\ release/40	\\wsl\$\Ubuntu-20.04\home\ejgo\gc\gitextensions\ tmp/vsl-push1			
		\wsl\$\Ubuntu-20.04\home\ejgo\gc\gitextensions\Externals\conemu-inside\ (no branch)	\\wsl.localhost\Ubuntu-20.04\home\ejgo\gc\GitExtensionsDoc\			
		\wsl\$\Ubuntu-20.04\home\ejgo\gc\GitExtensionsDoc\ tmp/wsl	\\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\ master			
		\\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\conemu-inside\ (no branch)	(\vsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\EasyHook\ (no branch)			
		\\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\Github\ (no branch)	\\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\ICSharpCode.TextEditor\ (no branch)			
		C:\dev\gc\gitextensions_2.51\ NikolayXHD/toobar	C\dev\gc\gitextensions_2.51\Externals\conemu-inside\ (no branch)			
		C:\dev/gc\gitextensions_2.51\Externals\Github\ (no branch)	C\dev\gc\gitextensions_2.51\Externals\NBug\ (no branch)			
		C\dev\gc\gitextensions_3\ master	C:\dev\gc\gitextensions_3\Externals\conemu-inside\ (no branch)			
		C:\dev/gc\gitextensions_3\Externals\EasyHook\ (no branch)	C\dev\gc\gitextensions_3\Externals\Git.hub\ (no branch)			
		C:\dev\gc\gitextensions_3\Externals\\CSharpCode.TextEdito\ (no branch)	C\dev\gc\gitextensions_4\Externals\conemu-inside\ master			
		C\dev/gc\gitextensions_4\Externals\EasyHook\ (no branch)	C\dev\gc\gitextensions_4\Externals\Git.hub\ (no branch)			
		C:\dev\gc\gitextensions_4\Externals\\CSharpCode.TextEdito\ (no branch)	C\dev\gc\gitextensions\Externals\conemu-inside\ (no branch)			
		C:\dev\gc\gitextensions\Externals\EasyHook\ (no branch)	C\dev\gc\gitextensions\Externals\Git.hub\ master			
		C\dev\gc\gitextensions\Externals\lCSharpCode.TextEditor\ (no branch)	C\dev\gc\tmp_tesboxx\GitExtensionsDoc3\			
		C\dev\gc\tmp\gitextensionsdoc_bare\	F:\Downloads\ReproGitExtensionsBug10423\			
	Contribute	ge C:\dev\gc\gitextensions_A release/40	\\ws/S\Ubuntu-20.04\home\ejgo\gc\gitextensions\ tmp/ws1-push1	Actions 🔺		
	<> DevelopImage: Image DevelopImage Develop	(no branch)				
77	漢 Translate	test		Actions A		
	♣ Issues	C\dev\gc\tmp_testxxx\GitExtensionsDac3\	C:\dev\gc\tmp\gitextensionsdoc_bare\	~		

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 repositories			
Search repositories			
Recent repositories			
\\wsl\$\Ubuntu-20.0	Show in folder		×
	Categories •		(none)
C:\dev\gc\gitextens tmp/go9-7	Remove project from the list		ge
	Remove missing projects from the list		other
\\wsl\$\Ubuntu-20.04\nc	pme\ejgo\gc\gitextensions\		test
		•	Add new

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.

3.1 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		?	\times
Directory	\sim	Browse	
Repository type			
Personal repository			
 Central repository, no working directory (bareshared=all) 			
		Creat	e

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.

3.2 Open repository

Opens a Git repo already existing on the file system.

🗙 Open loca	l repository				×
Directory:	\\wsl\$\Ubuntu-20.04\home\ejgo\gc\GitExtensionsDoc\	~		Browse	
	\\wsl\$\Ubuntu-20.04\home\ejgo\gc\GitExtensionsDoc\ C:\dev\gc\gitextensionsdoc\	^	6	Open	
ons_2.51\Extern	C:\dev\gc\gitextensions\ C:\dev\gc\gitextensions_4\ \\wsl\$\Ubuntu-20.04\home\ejgo\gc\gitextensions\ \\wsl\$\Ubuntu-20.04\home\eigo\gc\gitextensions\Externals\cou		\gc\gitexte	nsions_2.51\E	.: xterna
ons_3\	\\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\ICShar \\wsl\$\Ubuntu-20.04\home\ejgo\gitextensions\Externals\ICShar		\gc\gitexte	nsions_3\Exte	ernals\c

3.3 Clone repository

You can clone an existing repository using this option.

🗙 Clone		? ×
Repository to clone:	https://github.com/gitextensions/GitExtensions.git ~	Browse
Destination:	~	Browse
Subdirectory to create:	GitExtensions	
Branch:	(default: remote HEAD) ~	
The repository will be cle [Destination:]\GitExtens	oned to a new directory located here: ions	
Repository type		
Personal reposito	ry .	
 Public repository, 	no working directory (bare)	
🗹 Initialize all submo	dules 🛛 Download full history	
		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 initialized 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.

3.4 Clone Github repository

See GitHub.

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 *Windows Explorer* and from *Visual Studio*.

The Browse window contains of several parts:

- Main toolbar
- Revision graph
- Tabs
- Left panel

The Left panel, Tabs and the toolbar can be hidden, as well as showing the Commit tab as a panel in the revision graph.

4.1 Main toolbar

The main toolbar in Browse contains contains menus for other commands like Commit, Stash and Maintenance.

4.2 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.

	•			
🗙 gitextensions_5 (tmp/reword1) - Git Extension	ons f0344e66 (tmp/go9-6)			- 🗆 X
Start Repository Navigate View Com	nmands GitHub Plugins Tools Help			
🔯 🔲 🖃 🕶 📥 🕶 🔂 C:\dev\gc\gitext	tensions_5\ 🔹 🖟 tmp/reword1 👻 😽 🗣 🐨 😵 Commit (1) 隨 👻 🚞	🔷 🕶 🌐 🍸 👻 All branches 💌	Branches: V 🕆 🛄	Filter: 🛛 🕎 👻 📄 Open in VS Code
8 📮 🔶 🕞 💼	 Working directory + 1 		Author:	Gerhard Olsson <6248932+gerhardol@users.noreply
Q	 Commit index 		Date:	1 month ago (2022-04-29 21:14:28)
v 🔄 Submodules	tmp/reword1 Show multi re Gerhard Olsson	2022-04-29 21:14:28 2516b60	Commit hash:	9f9d9cf27c1ee117771197631024c0abd17d0e00
gitextensions_5 (tmp/reword1)	master upstream/master Sho 🙀 Gerhard Olsson	2022-04-29 21:14:28 9f9d9cf	Parent:	<u>t3ebc63e</u>
conemu-inside	left panel: reverted menu icon scal 👔 Holger Schmidt	2022-04-29 13:06:57 f3e5c63	Show multi revision diff also with	no HEAD (#9947)
EasyHook	Show no changes in grid for artific Gerhard Olsson	2022-04-29 00:43:21 14471c3	Related links: <u>View on GitHub</u> , Iss	ue 9947
ICSharpCode.TextEditor	E Diff Bo tils ton 0 CBC E Counts			
	 2 Diff Pies File tree P GPG Console Filter files using a regular expression (1) Diff with A f3e5c63e: left panels reverted menu icon scale to matc GRU//UserControls/FileStatusDiffCalculator.cs 	<pre>diffgit a/Git index a03fada0i a/GitU/User +++ b/GitU/User 0@ -112,21 +112,2 113 113 114 114 115 - if 116 116 { 116 116 { 116 116 } 118 118 } 119 119 120 2// // 121 - var 122 - var 121 + var 123 + var 124 124 15 - if</pre>	JI/UserControls/FileStatusDiffC2 0929557 106644 Controls/FileStatusDiffC2lculat 21 @@ public ReadonJyListFileS 21 @@ public ReadonJyListFileS 31 (lapSettings.ShowDiffForAllPare (lapSettings.ShowDiffForAllPare return fileStatusDescs; Sat base commit, add as parent i firstRevHead = GetRevisionOrtes selectedRevHead = GetRevisionOrtes selectedRevHead = GetRevisionOrtes firstRevHead = GetRevisionOrtes selectedRevHead = GetRevisionOrtes selectedRevHead = headId is nul baseRevGuid = (firstRevHead is	<pre>alculator.cs b/GitUI/UserControls/FileStatu: ^ ir.cs ir.cs itatusWithDescription> Reload() WithParent + GetDescriptionForRevision(fir: ithSubmodulesStatus(firstRev.ObjectId, selec ints revisions.Count > maxMultiCompare ints revisions.Count > maxMultiCompare if unique dd(firstRev, headId); Wead(selectedRev, headId); RedselectedRev, headId); RedselectedRevHead(firstRev, headId); Roull selectedRevHead(s null) ? null : etRevisionOrtHead(firstRev, headId); Null selectedRevHead is null) ? null : % </pre>

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



4.2.1 Search

You can find text in the commit messages or jump to a specific commit in the current commit history shown in Git Extensions.

4.2.1.1 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 Navigation

4.2.2.1 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

You can also filter the commit history so that fewer commits are shown.

4.2.3.1 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.

🍸 🔻 🛄 😰 All branches 🔻 Branches: 🛛 🗸 🍸 🖌 Filter: 🔍	₹.	\$
ad Michael Seibt	~	Commit message
d # Conflicts: #GitUl/UserControls/RevisionGrid/Graph/Rev 🚥 🌑 Michael Seibt		Author
traightenLanes clearer # Conflicts: #GitUI/UserControls/Rev 🌑 Michael Seibt		Diff contains (SLOW)

In the toolbar or 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.

1	manter unstrange /		waster test. Undate translations at 75	0/		
T	master upstream/	A	Copy to clipboard	^{/0}		
1	Merge branch 'releas					
•	Working directory	8	Checkout branch	I 1		
1.4	Commit index	սոր	Merge into current branch			
	Commendex	Ł	Rebase current branch on			
	▶ release/4.0 ▷ u	€ ∂	Reset current branch to here			
	Replace sync Commi	J.	Create new branch here Ctrl+B	lves i	#10926	
	Some recent reposito	A	Rename branch	repos	itories popup: Being able to see the repository	selected in the lists Being able
1.0	RussKie/ asvnc ICor	14	Delete branch	Com	mitMessageManager.cs	
II	Undate CitCommand	-	Reset another branch to here		5 5	
I.I.	opdate offcommand	12	Create powrtag here Ctrl. T			
	Update GitUl/Comma					
•	Update GitUl/Comma			-by: I	Vichael Seibt <36601201+mstv@users.noreply.g	github.com>
. •	Update GitUl/Comma		Checkout this commit	Aicha	ael Seibt <36601201+mstv@users.noreply.githul	b.com>
	Update GitCommand	1	Revert this commit	Mic	hael Seibt <36601201+mstv@users.noreply.gith	ub.com>
	Undate translations a	•	Cherry pick this commit			
		e	Archive this commit			
\mathcal{C}	Merge branch 'releas	1	Advanced •			
1) 🛉	Replace sync Commi	22	Compare +	26 Tł	ne use of sync API leads to observable delays an	d locking the application whe
	Update System.IO.Ab	٩	Navigate •			
	Merge branch 'releas	17	View •		Branches	
		0	Open on GitHub	ع	Show all branches	Ctrl+Shift+A
Commi	it 🔡 Diff 😤 Fil	17	View build report in a browser	la.	Show current branch only	Ctrl+Shift+U
and the second second	Author:	Ruse	sKie < russkie@gmail.com>	2	Show filtered branches	Ctrl+Shift+T
	Date:	17 h	ours ago (2023-05-01 00:41:18)		Show reflog referencer	Ctrl+Shift+1
	Commit hash:	1bb	c2c007960cfb4916b6864478aa1140d607d1f		Show renog references	CUITOINITE
0	Parent:	77c4	41427	Y	Advanced filter	Ctrl+l
				~	Draw non relatives gray	
odate tran	stations at 75%				Highlight selected branch (until refresh) Ctr	I+Shift+B, Alt+LButton
lated links	s: <u>View on GitHub</u>				Commits	
untained in	n branches:			~	Show artificial commits	
aster	n branches.			~	Show stashes	
stream/m	naster			~	Show git notes	
<u>ip/gos-/</u>					Grid Johnste	
intained i	n tags:				Chautabers	Chill, Childs, D
aster test				×	Show remote branches	Ctrl+Shirt+K
rives fron	n tag: <u>master_test</u>			~	Show tags	Ctil+Ait+1
					Show superproject tags	
					Show superproject remote branches	
					snow superproject branches	
					Grid info	
				~	Show build status icon	
					Show build status text	
				~	Show commit message body	
				~	Show author date	
				~	Show relative date	
					Columns	
				~	Show revision graph column	
				~	Show author avatar column	
				~	Show author name column	
		Show date column				
				~	Show SHA-1 column	
					Casting	
					Sorung	
					sort commits by author date	<u>,</u>
42 F	Revision grant	 1			Arrange commits by topo order (ancestor ord	er)17

When a filter is set, the icon for the advanced filter is changed and you see the current filter as a tooltip for the button. To remove the filter either remove the filter in the toolbar and press enter or remove the filter in the advanced filter dialog.

🗙 Filter	×
Since	🗌 den 1 maj 2023 🗐 🔻
Until	🗌 den 1 maj 2023 🗐 🔻
Author	
Committer	
Message	
Diff contains	
	✓ Ignore case
Limit	100000
Path filter	
Branches	
	Show current branch only
	Show reflog
	Only first parent
	Merge commits
	Simplify by decoration
	Full history
	Simplify merges
	ОК

4.2.3.2 Reflog

By default, Git will not show any commits that are not reachable and do not have any references, such as a branch or a tag. See Git reflog. Such commits will be purged when Git runs maintenance. By enabling Reflog via the toolbar button or in the View menu these commits will be shown too.

4.3 Tabs

For settings and further description, see *Tabs*.

4.3.1 Commit

Commit information and message as well as branch and tag information for the selected commit. This tab can be moved to the revision grid.

4.3.2 Diff

See also Show file differences for all parents in browse dialog and Show all available difftools

The file viewer will by default show the diff, but may also show Blame for the selected file.

4.3.3 File tree

Show the file tree for the commit.

The file viewer will by default show the file contents, but may also show Blame for the selected file.

You could search a file in the file tree using the default keyboard shortcut Ctrl + Shift + F (configurable in *Hotkeys*).

X gitextensions (rebase_from_revision_grid) - Git Extensions × Repository Navigate View Solution Runner Start Commands GitHub Plugins Tools Help >> 🗞 🔲 🥅 🥧 🕶 📴 gitextensions 💌 👂 rebase_from_revision_grid 💌 😽 👻 😭 011 🕝 Commit (1) 👫 (3) 🔻 📄 🚸 🔻 🎲 🐺 🖛 Prioritize branches and remotes in LeftPanel... a 6cb5d74 Author: Gerhard Olsson < 6248932 + gerhardol@ Date: 1 week ago (09/04/2023 00:26:14) master upstream/master Prioritize bran... ... 58078a2 Committer: GitHub <noreply@github.com> Commit hash: 66867165463bb383cdf7036d152d56a6f 84623b6 Bump vNext to 4.2 Child: f04e6b81 Rename to "LeftPanel" in hotkeys (#10857) f04e6b8 Parent: 75c701et Filter toolbar: Show reflog first and defa Enter File Name Reset renamed index files (#10853) Ren Form Restore from minimized: Configurable BugReporter/B BugReporter/BugReportForm.cs IScriptHostControl.CurrentBranch() to BugReporter/UserEnvironmentInformation.cs GitCommands/ExternalLinks/ExternalLinkFormat.cs GitExtUtils/GitUl/Theming/BmpTransformation.cs 🖳 Diff 🛛 😤 File tree 🔑 GPG 🛛 📰 Conso GitExtensionsShellEx/Resources/information.ico github 🥛 GitUI/Avatars/TemplateFormatter.cs > 📄 GitUI/CommandsDialogs/AboutBoxDialog/FormContributors.cs .tools GitUI/CommandsDialogs/BrowseDialog/DashboardControl/FormDashboardCategoryTitle.Designer.cs > i vscode GitUl/CommandsDialogs/BrowseDialog/DashboardControl/FormDashboardCategoryTitle.cs > 📄 assets GitUl/CommandsDialogs/BrowseDialog/DashboardControl/FormDashboardCategoryTitle.resx > 📲 Bin GitUI/CommandsDialogs/BrowseDialog/FormBisect.Designer.cs GitUI/CommandsDialogs/BrowseDialog/FormBisect.cs > EugReporter GitUI/CommandsDialogs/BrowseDialog/FormBisect.resx Externals > GitUI/CommandsDialogs/BrowseDialog/FormBrowseMenus.cs > CitCommands GitUI/CommandsDialogs/BrowseDialog/FormBrowseUtil.cs > E GitExtensions GitUI/CommandsDialogs/BrowseDialog/FormChangeLog.Designer.cs > 🧾 GitExtensionsShellEx GitUI/CommandsDialogs/BrowseDialog/FormChangeLog.cs > 📄 GitExtSshAskPass GitUI/CommandsDialogs/BrowseDialog/FormChangeLog.resx GitUI/CommandsDialogs/BrowseDialog/FormDonate.Designer.cs > iii GitExtUtils > 📄 GitUl > 📄 IntegrationTests 🔉 📄 Logo

4.3.4 GPG

Show the GPG key for signed commits.

4.3.5 Console

Show Git terminal.

4.3.6 Build report

Optional, only shown if the build server plugin is configured for the repo and if it has found a build for the selected commit. *Show build result page*

4.4 Left panel

Show local and remote branches, git remotes, tags, submodules and stashes.

CHAPTER 5

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 40 long characters cryptographically strong hash.

5.1 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.

Commit to feature/prepare-4.1 (C:\dev\gc\gitextensionsdoc\)		
Commit to fasture/prepar-4.1 (Cstdev/gc/gitedetmonidack) Working directory changes = Ther file using a regular expression. Source/browse_repository.stt source/images/install/digt_missing.png source/images/commit_contextual_menu.png source/images/commit_digt_wiss.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png/lick.exepng (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commit/commit_dialog.png (source/images/commit_dialog_spellcheckes source/images/commi	<pre>diffgit a/source/comit.rst b/source/comit.rst mindex 077dd637abce558 308644 +** / source/comit.rst b/source/comit.rst mindex 0.00000000000000000000000000000000000</pre>	commit in a commit
Committer Gerhard Olsson <gerhardol@users.noreply.github.com></gerhardol@users.noreply.github.com>	▶ feature/prepare-4.1 → origin/feature/prepare-4.1 Staged 10/43 Ln 3	Col 24 .:

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.



🗙 Commit to feature/prepare-4.1 (C:\dev\gc\gitexter

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.

5.1.1 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.

5.1.2 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.

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.

5.1.3 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.



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.

5.1.4 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\GitExtensions[Doc)	
🍓 📑 Working directory changes 👻			
Filter files	•		
 source/commit.rst source/images/commit_dialog.png source/images/reset_changes.png source/images/commit_reset_changes.png () 			
😭 🔓 Unstage	🌷 Stage 🛛 🌷		
There are no staged changes		Commit Commit & push Stage in Superproject Amend Commit Reset all changes Reset unstaged changes	Commit message • E Enter commit messag

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.



5.2 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 remote repository already.

Amend also enables the following options:

• By checking the Reset Author checkbox the author and date of the commit will also be overwritten.

• Reset soft: Perform a soft reset to the previous commit; leaves working directory and index untouched



CHAPTER 6

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.



6.1 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.

•	▶ rele	ease/4.0 > upstream/release/4.0	Updat
1	Replace	<u>e svnc CommitMessageManager Al</u> <u>C</u> opy to clipboard	Pl with ►
	F M.	Merge into current branch	•
	U A	<u>R</u> ebase current branch on	-
	•	Reset current branch to here	
🔘 Com	nm 🧏	Create new branc <u>h</u> here Ctrl+E	3

I will create a new branch called feature/my_branch. 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			?	\times
<u>B</u> ranch name		feature/my_branch		
Create branch at this revision		84fb5d93		-1
		Checkout <u>a</u> fter create		
84fb5d93				
Replace sync C	ommitMess	ageManager API with async (#10930)		
Author:	lgor Veliko	rossov		
Commit date:	18 hours ag	o (2023-05-01 00:31:12)		
Branch(es):	n/a			
Tag(s):	n/a			
Orphan				
Create orpl	han 🖂 Cle	ar working directory and index		
		a working directory and index		
				_
		<u>िक</u> ्ष <u>C</u> re	ate bran	ch

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

	release/4.0 upstream/release/4.0 Update change log
	Working directory 📀
1	Commit index 📀
•	feature/my_branch Replace sync CommitMessageManager API with async (#10930
•	Some recent repositories settings improvements (#10903) Configure repositories popup: B

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.

6.1.1 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.

6.2 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.

Ý	re 🕞	<u>C</u> opy to clipboard	•	ge log
•	W 🔝	Chec <u>k</u> out branch	•	₽ release/4.0
	C n	Merge into current branch	+	unstream/release/4.0
F	•	<u>R</u> ebase current branch on	•	essagemanager Ar i with async (#10
6.3 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.



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

÷	feature/refactor dummy commit 2
•	dummy commit
•	Working directory 🥝
•	Commit index 🥝
•	▶ master ▷ upstream/master tmp/dummy3 Update transl
	Merge branch 'release/4.0'
	release/4.0 upstream/release/4.0 Update change log
•	Replace sync CommitMessageManager API with async (#10930) ${\ensuremath{\mathbb R}}$
•	Some recent repositories settings improvements (#10903) Configu

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.

	feature (afected at the				
.	reature/refactor di	P	Copy to clipboard	•	
	dummy commit				
		82	Checkout branch	•	
•	Working directory	M,	Merge into current branch	•	& feature/refactor
	Commit index	L	Rebase current branch on	•	
•	▶ master ▷ upst	F o	Reset current branch to here		ions at 75%
	Merge branch 'releas	<u>ie</u>	Create new branc <u>h</u> here Ctrl+B		
	release/4.0 upstre	A	R <u>e</u> name branch	•	

In the merge dialog you can verify which branch you are working on, as well as set advanced merge options (see *Advanced Merge Options*). Select the branch to merge with then click the Merge button.

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 📀
Q	master Merge branch 'feature/refactor'
•	feature/refactor dummy commit 2
•	dummy commit
•	upstream/master tmp/dummy3 Update translati
٩l	Merge branch 'release/4.0'
	release/4.0 upstream/release/4.0 Update change log

For more information about merge conflicts and further options, see Merge Conflicts.

6.3.1 Advanced Merge Options

The Show advanced options checkbox will show the following options when activated:

• Use non-default merge strategy, with a drop-down field for selecting the alternate merge strategy. The strategies are:

- resolve

* This can only resolve two heads (i.e. the current branch and another branch you pulled from) using a 3-way merge algorithm. It tries to carefully detect criss-cross merge ambiguities and is considered generally safe and fast.

- recursive

* This can only resolve two heads using a 3-way merge algorithm. When there is more than one common ancestor that can be used for 3-way merge, it creates a merged tree of the common ancestors and uses that as the reference tree for the 3-way merge. Additionally this can detect and handle merges involving renames. This is the default merge strategy when pulling or merging one branch.

octopus

* This resolves cases with more than two heads, but refuses to do a complex merge that needs manual resolution. It is primarily meant to be used for bundling topic branch heads together. This is the default merge strategy when pulling or merging more than one branch.

- ours

- * This resolves any number of heads, but the resulting tree of the merge is always that of the current branch head, effectively ignoring all changes from all other branches. It is meant to be used to super-sede old development history of side branches.
- subtree
 - * This is a modified recursive strategy. When merging trees A and B, if B corresponds to a subtree of A, B is first adjusted to match the tree structure of A, instead of reading the trees at the same level. This adjustment is also done to the common ancestor tree.
- Squash commits
- · Allow unrelated histories
- Add log messages, with an input field for specifying number of log messages to add
- Specify merge message, with an input field for specifying a custom merge message

6.4 Rebase branch

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 📀
¢.	feature/refactor dummy commit 2
•	dummy commit
1 T	master upstream/master Update translations at 75%
	Merge branch 'release/4.0'
	release/4.0 upstream/release/4.0 Update change log
•	Replace sync CommitMessageManager API with async (#10
1	C

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

Working director	ory 📀			
Commit index	0			
► feature/refa	ctor dur	nmy commit 2		
dummy commit				
master tmp/d	um	Copy to clipboard	•	75%
Merge branch 're	eleas estre	Checkout branch Merge into current branch	•	
Replace sync Co	mm 🔬	Rebase current branch on	•	Selected commit
 Some recent rep 	osite 🐔	Reset current branch to here		Selected commit interactively
 Update translation 	ons a 🙀	Create new branch here Ctrl+B		Selected commit with advanced options

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

- The branch feature/refactor will be moved to the master branch
- The commits in the previous feature/refactor branch will be recommitted in the moved 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
Hide help	Rebase current branch on top of another branch Current branch: feature/refactor Rebase on 1bbc2c007960cfb4916b6864478aa1140d607d1f Interactive Rebase Preserve Merges Autosquash Specific range From (exc.) Io	□ Co <u>m</u> mitter date is aut	hor date
b + BASE b	Status Action Commit hash Subject	Author	Date
square = green = current branch single for the single commit \$.AAA postfix for files if merge conflicts occur			
		🔔 R	ebase

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.



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.

6.5 Interactive rebase

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

See Modify Git history for more information.

6.6 Squash

Git has no native squash operation, it can be done with various combinations of rebase and reset. This is described in the Git Extensions wiki.

See Modify Git history and Git documentation for more information.

6.7 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.

•	Working directo	лу 📀			
•	Commit index	0			
ф —	feature/refa	ctor dummy commit 2			
•	dummy commit	:			
	master t	Copy to clipboard	•	slatio	ns at 75%
	Merge brar release/4.0	Checkout branch Merge into current branch	۲ ۲		
•	Replace syr 🔬	Rebase current branch on	•	0930)) Resolves #10926
•	Some recer 🐔	Reset current branch to here		Confi	gure repositories popup: E
•	Update trar 强	Create new branch here Ctrl+B			
	Merge brar 웝	Rename branch	•		
	Translate 'G 😼	Delete branch	•	ş	master
	Translate 'G	Reset another branch to here		ş	tmp/dummy3
•	Avoid "git :	Create new tag here Ctrl+T		Ŀ	upstream/master

CHAPTER 7

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].

Avoid "git : warning: in the working copy of '<file
 Merge tag 'v4.0.2' Servicing release for v4.0
 v4.0.2 [...] Ensure release builds are clean
 Update build scripts * Disable loc-related warning
 Update changelog

Tags are also available in the Left panel.

7.1 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.

7.2 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 t	ag ×
Select tag	v3.00.00 V Delete
This will de	lete the selected tag from the (local) repository.
Delete t	ag also from the following remote(s):
0gerhardo	
🔯 <u>Help</u>	(includes information about deleting tags which are already pushed)

CHAPTER 8

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.

8.1 Manage remote repositories

You can manage the remote repositories in the Remotes menu or in the Left panel.



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.

🗙 Remote repositorie	s			×
Remote repositories	Default pull be	ehavior (fetch & merge)		
Active	-	Edit Remote Details		
_go mstv		Name	_go	
origin	ø	Url	https://github.com/gerhardol/GitExtensior v Browse 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.

🗙 Remote repositories				×
Remote repositories D	efault pull behavio	or (fetch & merge)		
Local branch name		Remote repository	Default merge with	^
master				
renamed_a1				
renamed_a2				
tmp/a				
tmp/aa				
				~
Local branch nam	e master			
Remote repository	/		~	
Default merge wit	h		~	
			Save changes	

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.

8.2 Git Credential Manager

The Git Credential Manager can be used to authenticate https links. For more information and instructions, see https://github.com/git-ecosystem/git-credential-manager.

8.3 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 practice 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.

8.3.1 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

X GitExtensionsDoc (feature/i5693-doc-3.00) - Git Extensions Repository Navigate View Commands GitHub Start Tools Help Plugins 6 Git bash Ctrl+G 🍓 🔲 🚍 🖛 📴 ▼ F:\dev\gc\gitextensions_4\GitExtensionsDoc 💌 fea Commit (6) 🛛 🚺 🔹 📄 🚸 🎲 🛛 Bran to Git GUI feature/ui-refresh _go/feature/ui-refresh Search: GitK 🗸 💻 Branches (4) Working directory 🥖 4 🛛 🗕 2 P PuTTY ۲ ث Start authentication agent ✓ I feature Commit index 🛿 ui-refresh Generate or import key 7 Git command log F12 ₽ i5693-doc ▶ feature/i5693-doc-3.00 ▷ _go/feature 8 Settings Ctrl+, 2.00-versic stash الا master release commit, split to stash, modify history 20

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

Key Please generate some randomness by moving the mouse over the	blank area.	Key Public key for pasting ssh-rsa AAAAB3NzaC1yc2E Jo+m66n +5TBgKaVVc2CO 26k3QixAPkcxlsnhw	into OpenSSH authorized_keys file: AAAABJQAAAQEAhR4ujlBw8igTyWR5+g3Ssi zQSfrVKKNKQbuR5Kyws6WBm3lbJlvdkey0kq dfNHu0EzB2x2BP27vBHmBwnnnUsX3S	Unv2qKi3jT4uZjIRfO
		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/pri	vate key pair	Generate
Load an existing private key file	Load	Load an existing priva	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 ED255	i19 🔿 SSH-1 (RSA)	Type of key to generation RSA	ate: DSA OECDSA OED2551:	9 OSSH-1 (RSA)
Number of bits in a generated key:	2048	Number of bits in a ge	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.

Q Add new SSH keys	× +			
← → ♂ û	🕕 🔒 GitHub, Inc. (US) https://github.com/settings/ssh/new 🛛 💀 🏠	2 IIIN	⊡ »	≡
Search or jump to	Pull requests Issues Marketplace Explore		+• 😐	•
Personal settings	SSH keys / Add new			
Profile	Title			
Account	Demo			
Emails	Kau			
Notifications	ssh-rsa			
Billing	AAAAB3NzaC1yc2EAAAABJQAAAQEAhR4ujlBw8igTyWR5+g3SsUnv2qKi3jT4uZjlRfOJo+	+m6t6n+5TBgKaVW0	CyCeOzQ Bwpppl.ls	
SSH and GPG keys	X3S+1XMmE9EeOjsvhaiGrSCnVPmLZybKJmxQPmNGM/tijl8mNOK7q/TDQG5ucK1CCq	ZEoUkt98k	1TuD=Ples	
Security	26Q4+Zzwkw== rsa-key-20181210		тугдоку	
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 dialogue 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.

8.3.2 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.



Type the following command: ssh-keygen -t ed25519 -C "your@email.com" 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
🧑 Commit
<pre>ejgo@ejgo3 \$ ssh-keyge Generating Enter file Enter passp Enter same Your identi Your public The key fin SHA256:qaip The key's n +[RSA 20 oo o ooo. + .oo.o . o++. = . BB0.+ S *E0 =*==0 . +B0B. =+*. +[SHA25 ejgo@ejgo3 \$]</pre>

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.

8.4 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.

🛯 💽 🗔 💶 🕶 📥 🖛 🔂 C:\dev\gc\gitextensionsdoc\ 💌 🐉 feat	ure/prepare-4.1 🔻 😽 🔻 😭 1† 📀 Commit (2) 🎼 🕶
--	---

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.

¥ Fetch (C:\dev\gc\tmp\both_changed_xxx\)		×	
<u>Hide help</u>	Pull from	[All] V Anage remotes V	
remote repository e d d d d d d d d d d d c d d d d d d d	Branch Local branch Rem <u>o</u> te branch		
	Merge options M. Merge remote branch into current branch A. Rebase current branch on top of remote branch, creates linear history (use with caution) Do not merge, only fetch remote changes		
	Tag options Follow <u>t</u> agopt, if not s Fetch <u>n</u> o tag Fetch <u>all</u> tags	specified, fetch tags reachable from remote HEAD	
	Prune remote branch Prune remote branch	es es an <u>d</u> tags	
	Solve conflicts	Stash <u>c</u> hanges Auto stas <u>h</u>	

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.

¥ Pull (C:\dev\gc\gitextensions_4\)			\times
Hide help Hover to see scenario when fast forward is possible. ^{merge} commit	Pull from <u>R</u> emote <u>U</u> RL	upstream Image remotes https://github.com/gitextensions/gitextensions.gi Image remotes]
+ REMOTE other remote repository e	Branch <u>L</u> ocal branch Rem <u>o</u> te branch	master v	
b 9.BASE	Merge options (a) Merge remote br (b) A Rebase current b (c) Do not merge, only	anch into current branch ranch on top of remote branch, creates linear history (use with caution) fetch remote changes	
square = Image: Square = current branch Image: Square = y.AAA postfix for files if merge conflicts occur	Tag options Follow tagopt, if no Fetch <u>n</u> o tag Fetch <u>a</u> ll tags	: specified, fetch tags reachable from remote HEAD	
	<u>P</u> rune remote branc Prune remote branc <u>Solve conflicts</u>	hes an <u>d</u> tags Stash changes Auto stash	

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.

♀ Pull (C:\dev\gc\gitextensions_4\)			×
<u>Hide help</u>	Pull from <u>R</u> emote <u>U</u> RL	upstream https://github.com/gitextensions/gitextensions.gi	Manage remotes
+.LOCAL other remote repository	Branch Local branch Rem <u>o</u> te branch	master v	
b +.BASE b	Merge options Merge remote brar Rebase current bra Do not merge, only <u>f</u> e	nch into current branch nch on top of remote branch, creates linear history (use w tch remote changes	ith caution)
a square = current branch y.AAA postfix for files if merge conflicts occur	Tag options Follow <u>t</u> agopt, if not s Fetch <u>n</u> o tag Fetch <u>all</u> tags	pecified, fetch tags reachable from remote HEAD	
	Prune remote branche Prune remote branche	is an <u>d</u> tags	
	Solve conflicts	Stash <u>c</u> hanges Auto stas <u>h</u>	<mark>⊸ P</mark> ull

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	

8.5 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 one commit.

To push the changes press Push in the toolbar.



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 (C:∖dev∖gc∖git	extensionsdoc\)	— 🗆 X
Push to <u>R</u> emote	origin	→ 🛃 <u>M</u> anage remotes
<u>○ Ur</u> l	https://github.com/gitextensions/GitExtensionsDoc.git	×
Push branches Pus	h tags Push multiple branches	
<u>B</u> ranch to push	feature/prepare-4.1 V feature	e/prepare-4.1 v
Eorce with lease	Force push	Recursive <u>s</u> ubmodules None ~
Replace tracking	reference	
Create pull reque	st after push	
P <u>u</u> ll		😭 <u>P</u> ush

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.

CHAPTER 9

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.

The structure of the submodules can be seen in the submodule toolbar and the Left panel.

9.1 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			– 🗆 X
Name	Status	Details	
Externals/EasyHook	Up-to-date	Name	Externals/Git.hub
Externals/Git.hub	Up-to-date	Remote path	//gitextensions/Git.hub.git
Externals/ICSharpCode.TextEditor	Up-to-date	Local path	Externals/Git.hub
Externals/conemu-inside	Up-to-date	Commit	3312526b05db52f2c70cf10b37221e3f361df677
		Branch	remotes/origin/HEAD
		Status	Up-to-date
Add submodule		😂 Synchroniz	ze 💛 Update Remove

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

To change a submodule path, delete the existing submodule, move the filesystem directory and add it again in the new location.

9.2 Add submodule

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

🗙 Add submodule		×
Path to submodule	https://github.com/pmiossec/GitExten ~	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.

CHAPTER 10

Worktrees

Git Extensions support Git worktrees: Multiple checked out working directories can share local branches. For more information see the Git documentation.

X gitextensions_4 (master) - Git Extensions



>	Existing worktrees			×
	Path	Туре	Branch	SHA-1
	C:\dev\gc\gitextensions	Branch	tmp/go9-7	79f9e29fa10745f914a616f70ed25404ddf3cd60
	C:\dev\gc\gitextensions_2.51	Branch	NikolayXHD/_toolbar	f1561fad44e3e6a7bf46d18055a8715deb0666f8
	C:\dev\gc\gitextensions_3	Branch	tmp/dummy3	1bbc2c007960cfb4916b6864478aa1140d607dlf
	C:\dev\gc\gitextensions_4	Branch	master	1bbc2c007960cfb4916b6864478aa1140d607d1f
l				
				💥 Delete selected 📄 Open selected
	Prune deleted worktrees			- Create

Note for WSL

Note that Git creates worktrees with a full "native" path, the worktree is only usable with the Git executable creating the path.

The path in the worktree file must be changed to a relative path if the worktree is to be used in both Windows and WSL.

CHAPTER 11

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.

🗙 Sta	ish														_			\times	
Show:	@{2}: WIP	on tmp/go9-	7: 3702c fi»	•	\overline{O}		Θ		¶ (div	¢_	J+ J	.*	Unicod	e (UTF-	8)	•	÷	s0u [.]	^
Filter file	les using a i	regular expres	sion		-		í		/GitCo	mman	ids/Gi	1+/6	ietAll	Change	edFil	es0u	tout	tPar	
🥖 GitC	ommands/	/Git/GetAllCh	angedFiles	Outp	utPa		-	++•b/	/GitCo	mman	ds/Gi	it/G	ietAll	Change	edFil	.es0u	tput	tPar	
0							(018	31,11.	+181	,17.0	<u>)@</u> •р	orivat	e∙stat	tic·I	Read	IOn 1	yLis [.]	
						181	181				··if·	(en	ntryTy	pe·==	. 15.17	·e	ntry	уТур	I
						182	182				…{ ∖n	1							
						183	183 -					De	ebug.A	ssert	(line	.Len	gth	·>·2	
						184		••••	• • • • • •	• • • •	• • • • •	••st	ring.	fileNa	ame•=	• lin	ie.Si	ubsti	
						185	-	••••		• • • •	• • • • •	··Uр	dateI	temSta	atus(entr	уТу	pe,.	
							184 -		• • • • • •	• • • •	•••••	••st	ring.	untra	ckedF	ileN	lame	•=•1:	
							185 -			• • • •		··Up	dateI	temSta	atus(entr	уту	pe,.	
						4.00	186 -					··co	ontinu	e;\n					
						100	187				···}\/	1 10.1	f. (on	toutu		. 141	. 11.	. onto	
						107	188 -	\n				26.1	ri - (en	cryry	pe.==	·· 1	.11.	·enti	1
<					>		189 -				··if·	(en	ntrvTv	ne·!=	.'1'.	22.0	ntry	vTvn	
Messag	le:					188	190				··{\n	(PC .	-			7 ' J P'	
WIP on t	tmp/go9-/:	3/02c fixup!	adding par	sing t	est		191 -					//	·Unep	ected	\n				
							192 -					··co	ntinu	e;∖n					
							193 -				$\cdot \cdot \} \setminus n$	1							
							194 -	\n											
🗌 Keep	o index	🗹 In	clude untra	cked		189	195				//.	Par	serfr	om·gi	t-sta	tus	docu	umen [.]	
						190	196	• • • •	• • • • • •	• • • •	··//·	Ign	nore·o	ctal∙a	and∙t	reeG	iuid	\n	
	S	tash all chang	es			191	197				··//·	1 · X	(Y·sub	m·≺mH:	>· <mi< th=""><th>:>·<m< th=""><th>w>…</th><th><hh></hh></th><th></th></m<></th></mi<>	:>· <m< th=""><th>w>…</th><th><hh></hh></th><th></th></m<>	w>…	<hh></hh>	
	Stac	h selected ch	ndes				(@•-23	35,7.+	241,	6.00	pri	ivate•	stati	c∙IRe	adOn	lyL	ist<(
	Juda	ii selecteu chi	inges			235	241	\n											
	Dr	op Selected S	ash			236	242				···Upd	late	eItemS	tatus	(y,∙f	alse	,∙si	ubm,	
		•				237	243			···}/	n								v
	Ap	ply Selected S	tash			<					n							>	

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

11.1 Revision graph

You can create multiple stashes if needed. The 10 latest stashes are shown in the commit log with the text [stash], all stashes if reflog is visible (see *Maintenance*).

2	stash@{0	WIP on release/4.0: 6084cf Update change log
	index on	elease/4.0: 6084cf Update change log
16	untracked	files on release/4.0: 6084cf Update change log
	tmp/go9	-7 AnnVevor better build description Remove skinne
		Closed, git Lino Officiation
		stash@{3} WIP on feature/i10423-conflicted-in-worktree-only: 3129f fixup! adding parsing test
		The second secon

11.2 Left panel

Stashes are also available in the *Left panel*. Select the non-grey stash commits to select the commits in the revision grid. To see stashes hidden in the revision grid, double click the stash.



CHAPTER 12

Modify Git history

A Git commit cannot be changed, the sha for the commit will be replaced at all changes. However, the contents of a commit can be modified and committed again as a new commit with a new sha and the branch/tag can be moved to the modified (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.

12.1 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:



The confirm dialog opens:

X Cherry pick con	nmit	×
Cherry pick this con	nmit:	
3a40a20abd		Choose another
Review chang	les	revision:
Author	Hand Weather	
Author:	Henk Westhuls	
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

12.2 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.

•	Working directory 📀
•	Commit index 🧇
ф —	feature/refactor dummy commit 2
•	dummy commit
	master tmp/dumm Merge branch 'release Merge into current branch release/4.0 upstreat Replace sync Commit Reset current branch to here Some recent repositor Reset another branch to here
	Update translations at Image branch 'release Create new tag here Ctrl+T Checkout this commit Checkout this commit Ctrl+T
(Translate 'GitUI/Translate' Revert this 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 cr	eate a commit	Revert this commit

12.3 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.



12.4 Modify an older commit

It normally makes sense just to change the history for the current branch. To change the parents of the current branch you will have to make a rebase. Git Extensions has functionality that wraps the Git rebase commands and simplifies usage in some situations.

12.4.1 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		—	×
Hide help f other + LOCAL C	Rebase current branch on top of another branch Current branch: feature/refactor Rebase on 1bbc2c007960cfb4916b6864478aa1140d607d1f ~ Interactive Rebase Preserve Merges Autosguash Auto stash Ignore gate Specific range From (exc.) Image: To feature/refactor Commits to re-apply:] Co <u>m</u> mitter date is autho	or date
b + BASE square = corrective for first if	Status Action Commit hash Subject	Author	Date
4.AAA merge conflicts occur			
		🔒 Reb	ase

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
З
    # Rebase 3c2ac977b..4feed7716 onto 3c2ac977b (2 commands)
4
    #
5
    # Commands:
6
    # p, pick <commit> = use commit
    # r, reword <commit> = use commit, but edit the commit message
8
    # e, edit <commit> = use commit, but stop for amending
q
   # 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
16
    # t, reset <label> = reset HEAD to a label
17
    # m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
    #.
18
              create a merge commit using the original merge commit's
    # .
              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
    #
28
    # Note that empty commits are commented out
29
```

You could have a look to Git documentation to better understand all the possibilities offered.

The options offered are :

- reorder the lines to reorder the commits,
- remove 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.

12.4.2 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...

• Worki	ing directory				
Comr	mit index				
🗖 🕨 🕨 fea	ture/refactor dummy change 2		11	Gerhard Olsson	2018-12-15 1
dı 🕞	Copy to clipboard	•		Gerhard Olsson	2018-12-15 1
• • •	Merge into current branch	▶ nu ···	×	Gerhard Olsson	2018-12-15 0
🔶 ບເຼຼັ	Rebase current branch on		澎	Gerhard Olsson	2018-12-13 2
J - F -	Reset current branch to here	99.90 ->		Gerhard Olsson	2018-12-10 2
s 🔒	Create new branch			Henk Westhuis	2018-12-13 1
K M 📑	Compare	▶ 859 ···	6.0	RussKie	2018-12-13 0
M ₂	Create new tag			Henk Westhuis	2018-12-12 2
• R	Checkout revision	ay witho		Henk Westhuis	2018-12-12 1
J 5 🖝	Revert commit	tExa •••	蠮	Henk Westhuis	2018-12-11 1
6	Cherry pick commit	nduct	1	Drew Noakes	2018-12-12 1
	Archive revision		-		
😫 Diff 😤 💼	Advanced	 Edi 	t com	mit	- 1
> 🚞 .github	Navigate	► Rev	word o	commit	
> - Bin	View	Cre	eate a	fixup commit	Ctrl+X
> 🚞 Build 👩	Open on GitHub	Cre	eate a	squash commit	()
> External	supdate	Ger	t neip	on now to use these	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			_		Х
<u>Hide help</u>	Rebase current branch on top of another branch Current branch: feature/refactor				
	Rebase on $3a40a20abd8be37865635468ff965a9f1 \sim$		Re	base	
Gother Gurrent	🗹 Interactive Rebase 🗌 Preserve Merges 🗹 Autosquash 🗹 Auto stash				
4.LOCAL	Specific range From (exc.)		Solve	conflicts	
	Commits to re-apply:		A .1		
d current f other	Name Subject Author Date Sta	atus	Add		_
•			Con		
b +.BASE b			Contin	ue rebase	
			Skip thi	s commit	:
			A	port	
current branch					
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.

12.4.3 Edit/reword commit

These options are the same as starting an interactive rebase on the parent to the selected commit and doing an edit (allow to amend to the commit) or reword (editing the commit message) and then run an interactive rebase in the background.

Note especially that this functionality will fail if you try to edit/reword a commit that is not a parent to the current checkout.

12.4.4 Rebase onto

When you would like to rebase a branch, it could happen that you don't want to rebase all the commits of the branch that git will by default determine to rebase.

One possibility is to do an interactive rebase and when git open the editor to let you decide what actions you will do on commits, you keep only the lines corresponding to the commit(s) you want to rebase (See interactive rebase to throw away a commit!)

If that is the last commit(s) of the branch that you want to rebase, you could instead do a *Rebase onto* where you select the range of commits to rebase by defining from which commit (not included!) you will start the rebase.

🗙 Git	tExtensionsDoc (gerhardol/feature/prepare-4.1) -	it Extensions d29880ca (rebase_from_revision_grid)		
Start	t Repository Navigate View Command	GitHub Plugins Tools Help Solution Runner		
Q	🔲 🔲 🐟 🔻 🛜 GitExtensionsDoc 🔻 👂 gerh	ardol/feature/prepare-4.1 🔻 🗣 🗣 🕲 Commit 🛛 📔 (0) 👻 🚞 🥎 🕶 🗐 💱 🖓 🖬 Branch	nes 🔻 Branches:	
	upstream/tmp/prepare-4.1 linkcheck: https a	id permanent redirect Update links after running linkcheck.		
•	Describe 4.1 features			
•	Documentation refresh Features in 4.0 or	ebase	- 0	×
•	Rename side panel to left panel	Rebase current branch on top of another branch		
•	Markdown lint on readme	Current branch: gerhardol/feature/prepare-4.1		
•	translation further removal	Rebase on dc1bc2130e22fed799bdc795f22fe898bfbbabe5 \lor		
	Working directory	🗌 Interactive Rebase 📄 Preserve Merges 🕜 Autosquash 📄 Auto stash 📄 Ignore date 📄 Co	ommitter date is author	r date
	Commit index 💿	Specific range From (ave.) 5990fe53		
	gerhardol/feature/prepare-4.1 ge			
	Documentation refresh Features in 4.0 or	Commits to re-apply:		
9	Rename side panel to left panel	Status Action Commit Subject Aut	thor	Date
Í.	upstream/main Remove translations			
•	Updated translations from Transifex Mate			
+	Updated source files Transifex api v3			
•	Update documentation for new checkbox			
	Diff 😤 File tree 🔑 GPG 🔳 Cons			
Filte	r files using a regular expression			
(20)) Diff with A 58f9fe53: Rename side panel to			
ØB m Ø M	nake.cmd Aakefile			
	eadme.md		🚨 Reba	ise
10 51	ource/branches.rst	16 16 The Left panel_, Tabs _ and the toolbar can be hidden, as well	I as showing the	Commit'

To select the base commit from which the range selection will be made, you have to options in Git Extensions:

• You can use the commit selection popup-up:

, X	Rebase -	- 0		
G	Rebase current branch on top of another branch			
	Current branch: gerhardol/feature/prepare-4.1			
	Rebase on dc1bc2130e22fed799bdc795f22fe898bfbbabe5 🗸			
	🗌 Interactive Rebase 📄 Preserve Merges 💿 Autosquash 📄 Auto stash 📄 Ignore date 📄 Committer d	ate is au	thor date	
	Specific range From (exc.)			
	X Choose Commit	-		>
	gerhardol/feature/prepare-4.1 gerhardol/feature/prepare-4.1 upstream/feature/prepare-4.1 Describe	4.1	0f4e42	2f
	Documentation refresh Features in 4.0 or earlier not described or description tuned.	🧃	e92499	5d
	Rename side panel to left panel	-	58f9f e	e5
	upstream/main Remove translations	1	629d78	83
	 Updated translations from Transifex Matching old source files 		0c0050	d6
	Updated source files Transifex api v3		471b10	ЭЬ
	 Update documentation for new checkbox 'Reset Author' (#138) 	J	P d5960d	1 4
	upstream/latest Add base documentation for advanced merge options (#144) - this is to support fixing https:		H 9e9f72	27
	upstream/release/4.0 bump version (#143)		979106	5f
	Update documentation for 4.0 (#140)		19101a	a1
	Find specific commit: Parent(s): 58f9fe53		OK	

• Or you could do it by selecting two commits from the revision grid. The first one selected will fill the *from* field.

🔀 GitExtensionsDoc (gerhardol/feature/prepare-4.1) - Git Extensions d29880ca (rebase_from_revision_grid) <u>Repository</u> <u>Navigate</u> <u>View</u> <u>Commands</u> GitHub <u>Plugins</u> <u>Tools</u> <u>Help</u> Solution Runner <u>S</u>tart 🔲 🧫 🖛 🚰 GitExtensionsDoc 🔻 🖗 gerhardol/feature/prepare-4.1 💌 😽 🖛 🎓 📀 Commit (3) 👫 (0) 👻 🚞 🥎 👻 🌐 🍞 👻 🛄 🕑 upstream/tmp/prepare-4.1 linkcheck: D. Copy to clipboard ۲ Describe 4.1 features 8 Checkout branch... ۲ Documentation refresh Features in 4.0 o m. Merge into current branch... ۲ Rename side panel to left panel Rebase current branch on ۲ Selected commit Markdown lint on readme Reset current branch to here ... Selected commit interactively. translation further removal Create new branch here... Ctrl+B je. Selected commit with advanced options... Working directory / 1 + 2 ĵę, Delete branch... 6 Reset another branch to here Commit index gerhardol/feature/prepare-4.1 Create new tag here... Ctrl+T /prepare-4.1 Describe 4.1 features ge 12 Documentation refresh Features in 4.0 or 2 Checkout this commit... c Revert this commit... Rename side panel to left panel Cherry pick this commit... upstream/main Remove translations Archive this commit...

The second will be the target commit when the branch will be rebased *onto*:

Warning: When doing a rebase onto, the *from* commit defining the range of commit(s) that will be rebased *must* be an ancestor of the current branch checked out that will be rebased.

12.5 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.

<u>P</u> lu	gins	<u>T</u> ools	<u>H</u> elp				
₽ ĵ	, tmp	/aa-rebas	se 🕶 😽 🕶 🕯	👔 🥝 Commit (3)	📔 🕈 🔶 🚞 🔻 🎒	P - 🛄	»»
	6	There are	e unresolved n	nerge conflicts.			Resolve
0			rena 🚴	Gerhard Olsson	24 days ago	891	.805d ^

12.5.1 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	- 🗆 ×	
Unresolved merge conflicts	Open in p4merge	
Filename		
a2	Start mergetool	
	Rescan merge conflicts	
A file with the same name has been created locally (ours) and remotely (theirs). Choose the file you want to keep or merge the files.	<u>R</u> eset	
Local a2		
Base no base		
Remote a2		
🔘 <u>Help</u>		

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

🗙 Resolve merge conflic	ts		– 🗆 ×
Unresolved merge conflic	Open in p4merge		
Filename	Chat manufact		
a2	Open in p4merge		Start mergetool
	Open in <u>m</u> ergetool		Rescan merge conflicts
A file with the san and remotely (the	Mark conflict as solved	Merge	<u>R</u> eset
or merge the files.	Choose local (ours) Ctrl+1		
lacel 2	Choose remote (theirs) Ctrl+2		
Base no base	Choose base Ctrl+3		
Remote a2	Open local with		
× 11-1-	Open remote with		
	Open base with		
	Save local as		
	Save remote as	1	
	Save base as		
	Open		
	Open With	1	
	Show in folder		
	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.

	-				
1 1 .					
s: Windows)					
Differences from base: 2					
Differences from base: 0					
Conflicts: 1					
./GitExtSshAskPass/SshAskPass_BASE_2604.rc2	./GitExtSshAskPass/SshAskPass_REMOTE_2604.rc2				
9 ////////////////////////////////////	<pre>9 ////////////////////////////////////</pre>				
25 FILETYPE 0x2L	25 FILETYPE 0x2L				
ShAdPass.c2 /////////////////////////////////					
VS VERSION INFO VERSIONINFO					
	<pre>:Whows) Differences from base: 2 Differences from base: 0 Confict: 1 .GitExtSchaldPase_BAEE_3604.rc2 .GitExtSchaldPase_BAEE_3604.rc2</pre>				
CHAPTER 13

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
```

13.1 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										
 Mail patches from 										
То				~						
Subject										
Body				_						
Feature/refactor	dummy commit 2	Gerhard Olsson	1 hour ago	e82659d	^					
 dummy commit 		Gerhard Olsson	1 hour ago	82cb062						
tmp/go9-7 AppVey	ror: better build description Remove skipped dis 🚥 🤱	Philippe Miossec	13 days ago	79f9e29						
Cl: more meaningful	icon instead of displaying a circle	Philippe Miossec	13 days ago	9850c51						
 (I hope) more pleasa 	nt default colors	Philippe Miossec	14 days ago	321ce46						
Same tooltip for Date	e, Author and Avatar column 🤱 🤱	Philippe Miossec	18 days ago	eale83f						
EditNetSpell: Close the second sec	ne completion when pressing some special char 🚥 🧣	Philippe Miossec	8 months ago	77e05a9						
🌒 📑 fixup! Form Reba	se: Add rebase dependent branches `update-refs` r 👔	Gerhard Olsson	10 days ago	56adcc6						
Form Rebase: Add re	base dependent branches `update-refs` rebas 🚥 🧣	Philippe Miossec	7 months ago	8c3c31c						
 Settings: refactor get 	ting true/false settings	Philippe Miossec	7 months ago	5938683	~					
Current branch: feature/refa	actor			Create patch(es)						

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

Patch result	×
F:/temp/0001-branches.patch	
ОК	

13.2 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.

Use 'Sign-Off' checkbox to sign off commits of applying patch. Git Extensions will remember your choice.

🗙 Apply patch	(C:\dev\gc\g	jitextensions_4\)			- 🗆 ×
Patch <u>file</u> Patch <u>direct</u>	ory	F:\temp\0001-status-monitor-tmp.patch		B <u>r</u> owse Bro <u>w</u> se	Apply patch ✓ Ignore Wh.spc. ✓ Sign- <u>O</u> ff
Status Name	Subject		Author	Dat	<u>Solve conflicts</u>
					Conflicts resolved Skip patch Abort patch

CHAPTER 14

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
  #
         Avoid blocking UI via WebClient.OpenReadTaskAsync
  #
  18 # GitUI/UserControls/AvatarControl.cs | 2 ++
19 # Plugins/Gource/GourceStart.cs
                                      3 +++
   # 3 files changed, 24 insertions(+)
```

CHAPTER 15

File history

File history is a separate form to view the history of a file or folder. Since Git Extensions 4.0 this functionality is included in *Browse Repository* but can be activated by *Show file history in the main window*. This form is deprecated and may be removed in future releases.

To display the single file history, right click on a file name in *File tree* or *Diff* tab and select File history or Blame.

The single file history viewer shows all revisions of a single file or submodules.



15.1 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: <u>GitHub < n</u> oreply@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 / GitUI/CommandsDialogs/FormBrowse.cs +++ b/GitUI/CommandsDi GitUI/CommitInfo/CommitInfo.Designer.cs @@ -1214,11 +1214,6 @@ // GitUI/CommitInfo/CommitInfo.cs var child GitUI/CommitInfo/CommitInfoHeader.Designer.cs RevisionI // GitUI/CommitInfo/CommitInfoHeader.cs 1217 if (Revis -// GitUl/MouseWheelRedirector.cs { Diff with: e558ea50 _ paren / .github/ISSUE_TEMPLATE.md paren 🛶 Externals/NBug (+1) } // GitCommands/GitCommands.csproj 1223 1218 + GitExtUtils/BinarySearch.cs private async GitExtUtils/GitExtUtils.csproj

15.2 Diff

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

Branches:			-	▼ + Filter:							
	Add	RefreshTree									
	Rework RepoObjectsTree so that each sub-Tree (Branches, Remotes, et										
	Issue 5616: Provide a way to remove all invalid recent repositories.										
ŧ١	Merg	je pull requ	est #5641 fro	om NikolayXHD/	fix_commit_info_	scroll					
•	Force	e commit in	fo to clear o	n module chang	je						
1.1.1	F:		01 1-	C							
🔵 Com	nmit	😫 Diff	🔎 View	🐞 Blame							
	di	ffgit	a/GitUI/0	CommandsDialo	gs/FormBrowse	.cs b/					
	in 	dex 7306† - a/GitUI	ece0759	0d88737 10064 Dialogs/Form	4 Browse cs						
	++	+ b/GitUI	/Commands	Dialogs/Form	Browse.cs						
	60	-1214,11	+1214,6	@@ private v	oid FillCommi	tInfo(
1214 12	14										
1215 12	15		var chi	ildren = Revi	sionGrid.GetR	evisio					
1216 12	16		Revisio	onInfo.SetRev	isionWithChil	dren(r					
1217	-		if (Rev	/isionInfo.Pa	rent is Panel	paren					
1218	-		{								
1219	-		par	ent.AutoScro	ll = true;						
1220	-		par	rent.AutoScro	llMinSize = R	evisio					
1221	-		}								
1222 12	17	}									
1223 12	18										
1224 12	19	pr	ivate asj	nc Task Fill	GpgIntoAsync()					

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

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

15.3 View

You can view the content of the file in after each commit in the View tab.

Branches:			•	₹.	Filter:		\mathbb{T}		
	Add F	RefreshTree							
	Rewo	Rework RepoObjectsTree so that each sub-Tree (Branches, Remotes, ϵ							
	lssue	5616: Provi	de a way to	remo	ve all inv	valid recent repositori	es.		
\	Merg	e pull requ	est #5641 fr	om Ni	kolayXH	D/fix_commit_info_s	croll		
•	Force	commit in	fo to clear o	on mo	dule cha	ange			
	г			· · · · ·					
🔵 Con	nmit	📱 Diff	🔎 View	*	Blame				
1 U	sing	System;							
2 U	sing	System.C	ollectio	ns.Ge	neric;				
3 U	ising	System.C	:omponent/	Model	;				
4 U	ising	System.D)iagnosti	cs;					
5 U	ising	System.D)rawing;						
6 L	ising	System.D)rawing.D	rawin	g2D;				
7 U	ising	System.]	:0;						
8 L	ising	System.L	inq;						

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

15.4 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: ▼ + Q + 🗇 巻 +											
	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											
🧑 Со	mmit 😫 Diff 🔎 View 🔻 Blame											
N. N. N.	Author:Henk Westhuis < henk westhuis@hotmail.com>Date:10 years ago (2008-11-27 20:17:44)Commit hash:bfcbc832fd9aa409f67fe9a7f3452a16739fe1feParent:a6a4eab406											
addeo	l files											
Notes												
1 2 3 4 5 6 7	Tal952 - 2012-06-15 00:03:18 - GitUI/FormBrowse.cs1using System;Henk Westhuis - 2008-11-27 20:17:44 - GitUI/Browse.cs2using System.Collections.Generic;Drew Noakes - 2018-06-19 11:37:16 - GitUI/CommandsDialogs/FormBrowse.cs3using System.Collections.Generic;Steffen Forkmann - 2010-07-28 16:44:38 - GitUI/FormBrowse.cs4Author: Henk WesthuisArkadiy Shapkin - 2011-10-28 01:53:22 - GitUI/FormBrowse.cs5AuthorTime: 2008-11-27 20:17:44Drew Noakes - 2018-07-19 16:22:56 - GitUI/CommandsDialogs/FormBrowse.cs6Committer: Henk WesthuisSteffen Forkmann - 2010-07-28 16:44:38 - GitUI/FormBrowse.cs7Summary: added files											

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

CHAPTER 16

Maintenance

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

16.1 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	tensions	_4 (fea	ture/versior	n-update)) - Git Exten	sions	;			
Start	Reposit	tory	Navigate	View	Comman	ds	GitHub	Plugir	ns Tools	н
3 65	🚱 Re	efresh			F5	-	feature/ve	ersion-u	pdate 👻 🌡	F →
	📄 Fi	le Expl	orer	Ctrl+Sk	tift+0	12				
Search:	A Re	emote	repositories			g dire	ectory 🤤	1	¥ 1	
Bra			I. I			t ind	lex			
	ຼຸລ ນ	Jomac	ules			Jre/\	version-u	pdate	Øgerhard	ol/f
	🥹 U	pdate	all submodu	lles		/feat	ture/lockf	reelist	Review char	nges
	i Sy	nchro	nize all subr	modules		sion	c/master	Marga	null request	#50
	🧟 W	/orktre	es		•	SION	is/indster	werge	puirrequest	.#JC
	D Ec	lit .giti	ignore			ptim	alizations			
	Ec	lit .git	/info/exclud	e		Sync	hronized	Collectio	on with chea	per
	Ec	lit .git	attributes			/feat	ture/tryge	etexacth	path Delete	: Try
	Ec	dit .ma	ilmap			akes	/code-of-	conduc	t Add code	of
	Ec	lit .git	review			akes	/fix-5859-	blockin	g-avatar-dov	wnle
	Sp	oarse \	Vorking Cop	у		aker	/macter	Margar		#500
	🥢 Gi	it mair	tenance		•	akes,	Compre	wierge p	atabase	+ 365
	All Re	enosito	ory settings			•	Recover	r lost ob	iects	-
	000		ing seconds				Delete	ndev loc	-b	
	M Cl	lose (g	o to Dashbo	oard) C	trl+W		Edit ait	loopfin		
	P :5206	throt	+1c		Cat Satha	2	East .git	/config	I Justicities Theorem	

16.2 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. The easiest way to view the commits is to show *Reflog*.

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

Start Repository Navigate	Vie	W Commands GitHub	Plugins	Tools	Help	
🖏 🔲 🖃 🕶 🔚 🕶 F:\de	· ~	Show all branches		Ctrl+	Skift+A	te
Search:		Show current branch only		Ctrl+	Skift+U	
Branches (112)		Show filtered branches		Ctrl+	Skift+T	
✓	~	Show remote branches		Ctrl+	Skift+R	
🕼 version-upd;	~	Show reflog references				
🖬 3.00-version 🗖						-

The reflog commits are listed as gray:

🗙 gitex	tensions_4 (master) - Git Extensions
Start	Repository Navigate View Commands GitHub Plu
@	🗆 🗔 🗢 🕇 🔂 🔻 F:\dev\gc\gitextensions_4 💌 master 💌 😽
•	Working directory 🚕 1 🛛 😓 1
•	Commit index
Þ	▶ master ▷ gitextensions/master Update build version
—	feature/version-update Ogerhardol/feature/version-updat
C	Update build version 2.99.90 -> 3.01.00.a0
9	Update build version 2.99.90 -> 3.01.00.a0
	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	tensio	ons_4 (m	aster) - Git E	xtensions	5					
Start	Rep	ository	Navigate	View	Comm	ands	GitHub	Plugins	Tools	Hel
B 5	<u>6</u> 2	Refresh	i		F5	• r	master 🔻	🛛 😽 🗕 😭	🕝 Cor	nmit
		File Exp	lorer	Ctrl+S	cift+O			1 -		
Ι.	\$	Remote	e repositories	i						
Ă.	4	Submo	dules							1.00
Ψ.	4	Update	all submodu	ules		date	build ver	rsion 3.00.	00 -> 3.0	1.00.
	\$	Synchro	onize all subi	modules		ture/	version-u	pdate Up	date build	l vers
2		Worktre	ees							
2		Edit .git	tignore							
		Edit .git	/info/exclud	le		nges				
		Edit .git	tattributes			es/fix	-5859-blo	cking-avat	ar-downl	oad
1))		Edit .ma	ailmap							
		Edit .git	treview			aper a	array with	out locking	9	
		Sparse	Working Cop	у		e Try	GetExactP	ath, to rem	ove perfo	orma
	ø?	Git mai	ntenance			2	Compre	ess git data	base	
	÷	Reposit	ory settings				Recover	r lost objec	ts	llie
1	N	Close (g	go to Dashbo	oard) C	trl+W	_	Delete i	ndex.lock		
		aitexten	sions/release	⊳/3.00 F	ives 5880	Reorde	Edit .git	/config		

>	Verify database				— 🗆 X		
B 2 d C C D	y default only unre weeks are removed ther object are only angling objects" heck commits you ontext menu for ac ouble-click on a ro	ferenced objects that d when cleaning up v deleted when you want to recover and Iditional operations w for quick view	at are older than the database. All run "Remove all d press Recover button	 Show commits and tags Show of Do not consider commits that are referreflog to be reachable. Print out objects that exist but that are nodes. Check not just objects in GIT_OBJECT_but also the ones found in alternate of 	her objects renced only by an entry in a n't readable from any of the reference DIRECTORY (\$GIT_DIR/objects), bject pools.		
	Date	Type	Author	Hash	Parent(s) hashs		
	2018-12-13 21:58	dangling commit	Gerhard Olsson	cc75167da4417ba3a1cf2bfcc6eaf0de45bd03cc	a 177c46a2f0e7a42481d21e2f5638d9		
	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		
	Remove all o	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.

16.3 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"</henk_westhuis@hotmail.com></henk@.(none)>
	Save

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

16.4 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.

🗙 Edit .gitignore	- 🗆 ×
<pre>1 build 2 *.orig 3 tx.exe 4 GitExtensions.settings.backup 5 source/locale/.doctrees/ 6 /source/_pycache/ 7 /source/extensions/_pycache/ 8 *.pyc 9</pre>	*.suo*.bak*.cache*.ilk*.log[Bb]in[Dd]ebug*/*.sbrobj/[Rr]elease*/_ReSharper*/
Add default ignores Add pattern	Example ignore patterns Generate a custom ignore file for git
	Cancel 📰 Save

A short overview of the syntax:

#	Lines started with # are handled as comments	
!	Lines started with ! are exclude patterns	
[Dd]	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	
	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.

CHAPTER 17

Settings

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

17.1 Git Extensions

The top level page has a checklist for settings in Git and Git Extensions.



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 of		
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.

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.

17.1.1 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 (and next) 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 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 are not monitored).

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 is turned off by default.

Show ahead and behind information on status bar in browse window

If the current local checkout branch is tracking a remote branch, show the number of commits the branch is ahead (changed locally) and behind (changed on the remote) on the status bar in *Main toolbar* and for branches on the *Left panel*.

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 histogram diff algorithm

Use the Git 'histogram 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 autostash

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.

Update submodules on checkout

Update the commits for submodules when updating the commit for the current repository.

Follow renames in file history

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.

Default pull action

The default action for *Pull* in *Main toolbar*, see the dropdown list.

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.

Telemetry

Yes, I allow telemetry!

Allow that Git Extensions collect anonymous information about usage.

17.1.2 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 names in the dashboard and the recent repositories dropdown menu Also show the branch in Left panel.

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

Show author's avatar column in the commit graph

If checked, avatar images are downloaded for commit authors and shown in the revision grid.

Show author's avatar in the commit info view

If checked, avatar images are downloaded for commit authors and shown in the commit info view.

Cache images (days)

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

Avatar provider

The avatar provider setting determines the source from which avatar images are requested.

- Default The default avatar provider loads a user defined avatar images, depending on the email address, from GitHub or Gravatar. If no user defined image could be found, a fallback images is used.
- None If selected, no user-defined images are loaded and the fallback is evaluated immediately.
- Custom An advanced mode that allows you to set one or more custom avatar provider services (e.g. Libravatar) by providing URL templates.

URL Template Syntax

The URL template syntax consists of regular URLs to avatar images, that can be enriched with variables, which are substituted before evaluation. Those variables are encoded using curly brackets {} and can be used like this: https://example.avatar.service/u/{email}/avatar.png. If a request fails (http 400 and 500 errors) or does not provide a valid image, the next URL is used. More URLs can be specified by chaining them together with semicolons (";") like so: https://provider1.com/ {shal}.png; https://provider2.com/{shal}.png. If all custom URLs fail to provide an avatar image, the applications internal fallback mechanism will provide one for that user. The variable names are case insensitive. If a variable is not found (for example because of typo or it does not exist), it is substituted with an empty string, so the resulting URL never contains the curly brackets.

The following variables are currently supported:

- name The name of the commit author (git config user.name). Special characters are URL encoded.
- email The email address of the commit author (git config user.email). Special characters are URL encoded.
- md5 A lowercase hex representation of the MD5 hash of the normalized (all characters lowercase) email address (without URL encoding). This hash is compatible with Gravatar and thus compatible with a lot of similar services.
- sha1 Like the md5 variable but with SHA1 as hash algorithm.
- sha256 Like the md5 variable but with SHA256 as hash algorithm.
- imagesize Represents the requested avatar size in pixels.

A complete working configuration might look something like this: https://www.libravatar. org/avatar/{md5}?s={imageSize}&default=404;https://avatar.tobi.sh/ {md5}?size={imageSize}

Fallback generated avatar style

The configured fallback determines how authors without a user-defined avatar are presented. Besides Author Initials all other options are provided by Gravatar. Details about their fallback modes can be found here https://en.gravatar.com/site/implement/images/ in the section "Default Image". Author Initials are generated by the application internally and require no network connection to be displayed.

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.

17.1.3 Sorting

Sort revisions by

This setting causes commits in the revision grid to be sorted by Git default (commit date), author date or topology. Sorting by other than Git default may delay rendering of the revision graph.

Sort branches by

The sort order for branches in Main toolbar and Left panel in a dropdown.

Order branches

Order the branches within the sorting in Sort branches by.

Prioritized branches

Regex to prioritize branch names in the left panel and commit info. The branches matching the pattern will be shown before the others. Separate the priorities with ';'.

Prioritized remotes

Regex to prioritize branch names in the left panel and commit info. The branches matching the pattern will be shown before the others. Separate the priorities with ';'.

17.1.4 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.

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.

Highlight authored revisions

Highlight revisions committed by the same author as the selected revision.

Fill Git ref labels

Fill labels in the revision grid.

Theme

Git Extensions allows that some application colors are changed. A few themes are included.

For more information see the README in the themes folder or GitHub.

Open Theme folder

Open the folder with the themes in Windows Explorer.

Colorblind

Adjust the theme colors for colorblind users (if specified in the theme).

Use system-defined visual style

Use a the system wide visual style (will not look good with all themes).

17.1.5 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.

17.1.6 Console style

Settings for the ConEmu console tab.

Console settings

Console style

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

Font

Console font size.

17.1.7 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		#5853: Use s
· · · ·		//	and a final a fac		1.111
🍥 Commit	😫 Diff	😤 File tree	🔑 GPG	Console	
	Author: Date: Commit Child: Parent:	Henk W 3 days a hash: c046f57 <u>1b59094</u> 9bc432	/ <u>esthuis <he< u=""> igo (2018-12 c939be7ddal <u>14a3</u> 1707</he<></u>	nk.westhuis@u -05 13:41:22) b2da5b7d21c26	ultimo.com> 5a043e0792d
#5853: Use	stack i	nstead of re	ecursion i	in EnsureSco	oreIsAbove

Related links: View on GitHub, Issue 5853 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. You could add this quite generic GitExtensions.settings file to the root of your repository.

🗙 Settings - Revision links					×
X Settings - Revision links Type to find Y Git Extensions Appearance Revision links P Ruid server integration P Scripts P Hotkeys SH Shell extension P Detailed P SH P Git P Plugins	Settings source: Effective Categories 1 GitHub - issues 2 GitHub - PR 3 GitHub - commit	C Local for current r	repository ame 1 GitHub - Remote data Use remotes	Distributed with current repository issues upstream origin	 Global for all repositories ☑ Enabled <a>S Help ☑ Only use the first match
		2 2 9 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Search in Search pattern Revision data Search in Search pattern Nested pattern	✓ URL Push URL (?i)^(git@ http(s?)://)([^:/]+)[:/](.*)).git\$ ✓ Message ✓ Local branch name (?i)(? pull request pr[_]?)(# (((feat(ure))).git))</td \d+	Remote branch name ?)[fix:][/]]))(d+
			Taption sue {4}	Links URI http{1}://{2}//3}/issues/{4}	
	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.

Note that the branch name is only searchable in the branch heads.

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%

17.1.8 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.

Show build result page

Show a page with build information in *Tabs*.

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. Required 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

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.

17.1.9 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.

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).

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.

Execute on event

Select when this command will be executed, either before/after certain Git commands, or displayed on the User Menu bar. Since the git pull command includes a fetch, before/after fetch events are triggered on pure fetches as well as on pulls. For the pull command the script execution order ist BeforePull - BeforeFetch - git pull - AfterFetch - AfterPull.

Icon

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

Script behavior

Ask 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.

Is PowerShell script

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.

Script context

Show in RevisionGrid

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

17.1.10 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. LeftPanel: The left panel for Browse with branches etc.
- 4. RevisionGrid: The list of commits in Browse and other forms.
- 5. FileViewer: The page displayed when viewing the contents of a file.
- 6. FormMergeConflicts: The page displayed when merge conflicts are detected that need correcting.
- 7. BrowseDiff: Diff tab in Browse.
- 8. RevisionFileTree: The FileTree tab in Browse.
- 9. Stash
- 10. 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).

17.1.11 Shell extension

Windows explorer integration

Enable Git Extensions to add 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.

Cascaded context menu

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.

17.1.12 Advanced

This page allows advanced settings to be modified. Refer Confirm actions.

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 normalized as per git branch naming rules. If checked, 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 checked 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 checked, 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.

17.1.13 Confirmations

This page allows you to turn off certain confirmation popup windows by unchecking the checkboxes.

Confirm actions

Commits

Amend last commit

Display the popup warning about the rewriting of history when you have elected to amend the last committed change.

Undo last commit

Display the warning when undoing (resetting) the commit for the current branch in Main toolbar.

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.

Rebase on top of selected commit

Rebase context menu command popup in revision graph.

Branches

Fetch and prune all

Browse fetch/prune popup.

Push a new branch for the remote

Warning when pushing a new branch that does not exist on the remote repository.

Add a tracking reference for newly pushed branch

Warning 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 unchecked, a tracking reference will always be added if it does not exist.

Delete unmerged branches

Display the warning when deleting a branch that has not been merged to the current branch (use -force).

Stashes

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 unchecked, 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 unchecked, 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.

Rebase / conflict resolution

Resolve conflicts

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

Commit changes after conflicts have been resolved

Uncheck 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 any other operation that caused conflicts to be resolved, an user is warned about the consequences of aborting and is 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. Uncheck this option to skip this second confirmation.

Submodules

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 not checked, the submodules will be updated without asking.

Worktrees

Switch Worktree

Switch worktree popup.

17.1.14 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 received 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 *Git merge <https://gitscm.com/docs/git-merge#Documentation/git-merge.txt*—logltngt>.

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.

17.1.15 Browse repository window

General

Default shell

Choose one of the predefined terminals in Console tab and browser popup.

Show file history in the main window

Open file history in Browse Repository window instead of the deprecated File history window.

Show blame in diff view

Show blame in the diff view tab *Diff* instead of switching to *File tree* tab.

Tabs

Show the Console tab Show the Console tab in the *Browse Repository* window.

Show GPG information

Show tab for GPG information if available.

17.1.16 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 Auto-CompleteRegexes.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.

17.1.17 Diff viewer

Settings for *Diff*.

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.

Remember the 'Show syntax highlighting' 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 *git diff-tree <https://git-scm.com/docs/git-diff-tree#Documentation/git-diff-tree.txt*—*cc>*

Enable automatic continuous scroll (without ALT button)

For file status lists like in *Diff* and *Commit* it is possible to scroll continuously to the next (or previous) file with the mouse wheel and *ALT* button. This setting allows scrolling to the next file with only the mouse wheel.

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

The *Diff* can show more than one diff, depending on the selections in *Revision graph*.

- For a single selected commit, show the difference with its parent commit.
- For a single selected merge commit, show the difference with all parents.
- For two selected commits with a common ancestor (BASE) or two *ranges* described below, show the difference between the commits as well as the difference from BASE to the commits. See below for more details about icons and range diffs.
- For multiple selected commits (up to four), show the difference for all the first selected with the last selected commit.
- For more than four selected commits, show the difference from the first to the last selected commit.

Common BASE icons

If the selected commits have a common BASE, the icons in the file list has an overlay on the icon with information where the file has been changed.

- A Change done in first (A) commit.
- B Change done in selected (B) commit. (Last selected commit.)
- = Same change in both commits.
- ! Unequal changes are done in the commits.



Range diff

git range-diff <https://git-scm.com/docs/git-range-diff> shows the difference between two versions of a patch series with a common BASE. The command can require a lot of resources and it is possible to define the ranges for Git .

• If two commits are selected, all commits from BASE to selected (B) and first (A) are included. With Git this is written as A...B, BASE A B or BASE..A BASE..B.

Example where one commit differs for two branches (but the branches have identical information).

¢.	master PostRepository change: Shared G	itRefs() (#9735) Share the derived references instead of calling GetRefs() r		
•	Mergetool and difftool commands are now populated if paths are manually configured (#9780) Fix #8918 Command is alte			
 Load app colors from CSS themes (#9784) 				
	upstream/master PostRepository change: Sha	red GitRefs() (#9735) Share the derived references instead of calling GetRefs() n		
•	Mergetool and difftool commands are now pop	ulated if paths are manually configured (#9780) Fix #8918 Command is altered		
•	Load app colors from CSS themes (#9784)			
	signed CLA (#9781)			
	upstream/translations_plugins-3_5_ko upstre	am/translations_ui-3_5_ko WSL: force-push was inserted incorrectly (#9760)		
	WSL: Git internal paths were incorrectly resolved			
T .				
55 L	Off 😤 File tree 🔤 Console			
Filter	files using a regular expression 🔹 1:	ad3c8aa9e < -: signed CLA (#9781)		
Diff v	vith A 07d87634: upstream/master — 🗧	771h327eb = 2: c59a39918 Mergetool and difftool commands		
🕎 Rai	nge diff 4↓ 3↑ BASE 1097a210: upstream/trans 4.	07d87634f ! 3: 186b93115 PostRepository change: Shared G		
(21) [Diff BASE with B 186b9311: master	00 Commit message		
(21)	Diff BASE with A 07d87634: upstream/master			
(21)1	on base with a order to a apstream, master	Share the derived references instead of calling GetR		
		Amondod the file in the chinned commit		
		+ Amended the file in the skipped committee		
		<pre>## GitUI/BranchTreePanel/RepoObjectsTree.BranchTree.cs ;</pre>		
		00		
		-using System.Collections.Generic;		
		@@ UnitTests/GitCommands.Tests/Git/FilteredGitRefsProvide		
		+ }		
		+ 1		
		+		
		+ ## contributors.txt ##		
		+@@ contributors.txt: YYYY/MM/DD, github id, Full name, e		
		+ 2021/10/22, calebnhay, Caleb N. Hay, caleb(at)calebnhay		
		+ 2021/10/28, matthiaslischka, Matthias Lischka, matthias		
	_	+ 2021/11/13, h0lg, Holger Schmidt, leroyz.mailaccount(at		
	-	+ 2021/12/06, Tyrrr2, Alexey Golub, tyrrr2(at)gmail.com		
	-	++2021/12/06, Tyrrrz, Alexey Golub, tyrrrz(at)gmail.com		
		++2021/12/29, blazejszuca, BÅažej Szuca, blazej.szuca+gi		
		+ \ No newline at end of file		

• If two ranges are selected with four selected commits (where the number indicates the click order) A1..A2 B3..B4 where BASE is parent to A1 and B3 as well as A1 is a parent to A2 and B3 is a parent to B4. Note that A2 is considered as first selected commit in the diff.

Example where only two of the commits are compared.



Show all available difftools

Git Extensions uses the default Git GUI diff and merge tool in *Config*. This setting enables a submenu for many diff and merge tool menus with all tools known by Git. This enables use of specific tools in certain situations like using *TortoiseGitIDiff* specifically for images.



Note for WSL Git

For *Notes for WSL Git* the Windows Git version is always used for diff and merge tools so the same tools is available in WSL as in Windows.

Vertical ruler position

Position for ruler in TextEditor controls. Set to 0 to disable. (This should be moved to the TextEditor context menu.)

17.1.18 Blame viewer

Settings for blame in File tree and Diff.

Blame settings

Ignore whitespace

See git blame -w.

Detect move and copy in this file

See Git blame -M.

Detect move and copy in all files

See Git blame -C.

Display result settings

Various settings for the blame viewer.
17.1.19 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

For PuTTY, the paths to the executable must be specified.

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.

17.2 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.

17.2.1 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.

Notes for WSL Git

For Git repos stored in \\wsl\$ or \\wsl.localhost directories, Git Extensions executes the WSL Git executable where possible to improve performance. WSL Git is several times faster than Windows Git (native) application.

The paths internal to Git Extensions are always in Windows format. Therefore, paths in both inputs and outputs for WSL Git commands must be translated. For instance \\wsl\$\Ubuntu\repo to /repo and c:\repo to /mnt/c/repo.

The Git Extensions Windows (native) Git executable is still used for the following:

- All handling and settings related to Git in Settings. This includes display of Git version as well. However, if the WSL Git version is too old to be supported, Git Extensions will report this in a popup.
- Custom merge implementation in FormResolveConflicts.
- Custom difftool/mergetool list, see Note for WSL Git.
- ScriptRunner and some built-in plugins like FindLargeFiles always use Windows Git.

Some notes: - Git repos accessed in \\wsl.localhost will be displayed as \\wsl\$ (so only one occurrence in recent lists etc). - Git repos mapped to a drive letter will not use the special WSL handling but Windows Git. - Files modified in WSL are not reported by Windows FileSystemWatcher, so the GitStatusMonitor will only report issues at explicit refresh and every minute. - The WSL executable occasionally fail (for instance when the WSL machine is busy) which will be seen as a Git failure that will result in a popup. You may have to ignore the popup, refresh or even reopen the application to recover from these failures.

See also Worktrees for Git limitations.

17.2.2 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).

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://docs.github.com/en/get-started/getting-started-with-git/configuring-git-to-handle-line-endings

Files content encoding

The default encoding for files content.

17.2.3 Advanced

Various settings for Git.

17.3 Plugins

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

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.

18.1 Bundled

18.1.1 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.

18.1.2 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/enterprise/data-center

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.

18.1.3 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.

18.1.4 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.

18.1.5 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.
```

speenij wina size is considered a mag

18.1.6 GitHub

This plugin adds a GitHub menu item in the main toolbar.

- Fork/Clone repository
- View pull requests
- · Create pull requests
- Add upstream remote

Personal Access Token

The plugin adds configuration for the token generated and retrieved from GitHub. For more information see: https://github.com/ or the links in the plugin settings.

18.1.7 GitFlow

This plugin permit to manage your _branching model: https://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

18.1.8 Gource

Gource is a software version control visualization tool.

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

Path to "gource"

Enter the path to the gource software.

Arguments

Enter any arguments to gource.

18.1.9 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.

18.1.10 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.

18.1.11 Plugin Manager

Plugin to manage third party plugins.

18.1.12 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.

18.1.13 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.

18.1.14 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).

18.2 Third party extensions

18.2.1 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/

18.2.2 Jira Commit Hint

🗙 Settings - Plugin: Jira Commit Hin	t	×
Type to find	Settings source:	ocal for current repository << 🔿 Distributed with current repository << 🔿 Global for all repositories
General Appearance Appearance Sevision links Sild server integration Scripts Hotkeys Kolle extension Min Advanced Soll SSH SGit SSH SGit Sild Server integration Sild Server	Jira hint plugin enabled Jira URL Jira user Jira password JQL Query	Image: State Stat
Bitbucket Server Delete obsolete branche Find large files GitHub Gource Jira Commit Hint Periodic background fet Proxy Switcher Statistics	Jira fields Message Template	{HasUserVoted} (Item) {Jira} {Jiraldentifier} {Key} {Labels} {ParentIssueKey} {Priority} {Project} {Reporter} {Resolution} {ResolutionDate} {SecurityLevel} {Status} {Summary} {Type} {Updated} {Votes} {Key} {Summary} Preview
< >>		OK Cancel Discard Apply

Provides hints for Atlassian Jira issues in the commit form. For example, you can configure Key - Summary message for all your in progress tasks.

Jira hint plugin enabled

Whether plugin enabled or not.

Jira URL

Link to your Jira server.

Jira user

Your username.

Jira password

Your password.

JQL Query

Query to Jira, results of which you want to show in "Commit Templates" in Commit Form. For more information see: https://confluence.atlassian.com/jiracoreserver073/advanced-searching-861257209.html

Jira fields

Key words that you can use in Message Template.

Message Template

Result format to insert into message text box after some line from "Commit Templates" selected.

GitHub

Git Extensions has specific integration with GitHub that adds a GitHub menu item in the main toolbar.



19.1 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.

ny repositories	Search for repositories							
Name		ls fork	# Forks	Priv	^			
Git.hub		Yes	0	No				
Git.hub		Yes	29	No				
gitextensions		Yes	1	No				
gitextensions		No	2025	No				
GitExtensions.A	AzureDevOpsCommitMessage	No	1	No		If you want to fork a repository owne		
gitextensions.e	xtensibility	No	4	No				
GitExtensions.G	GerritPlugin	No	8	No		repositories tab.		
gitextensions.g	ithub.io	Yes	0	No				
gitextensions.github.io gitextensions.pluginmanager gitextensions.plugintemplate gitextensions.vsix		No	13	No				
		No	10	No				
		No	8	No				
		Yes	0	No				
gitextensions.v	six	No	10	No	\mathbf{v}			
lone estination fold C:\dev\gc	er:		Bro	owse				
reate directory:	Add up	stream re	mote as:	~				
imit Depth:)	<u>A</u> •							

19.2 View pull requests

View current pull requests for the active remotes hosted on GitHub, including the diff and comments.

X View	w Pull Reque	sts			_		\times
Choose	repository:	gitextensions/gitextensions \checkmark					
#	Heading		Ву	Createc ^	Fetch to pr	/ branc	:h
10915	Avoid doub	ole graph drawing	mstv	2023-04			
10906	Tweak repo	s dropdown	RussKie	2023-04	Add remote	and fet	tch
10893	Cl: more m	eaningful icon instead of displaying a circle	pmiossec	2023-04 🗸			
<				>	Close pull	reques	t
Diffs	Comment	is					
Filter f	iles using a n	egular expression					-
🥖 Gitl	JI/Command	sDialogs/BrowseDialog/DashboardControl/UserRepo	sitoriesList.cs				~
/ Gitl	JI/Command	IsDialogs/FormBrowse.cs					
GitUI/CommandsDialogs/FormBrowse.Designer.cs							
🥖 Gitl	GitUI/CommandsDialogs/Menus/ToolStripMenultemEx.cs				\checkmark		
<							>
	index f57cc0e7693c562bda8f09.100644\n					~	
a/GitUI/CommandsDialogs/BrowseDialog/DashboardControl/UserRepositoriesList.cs\n							
	+++∙b/Git	UI/CommandsDialogs/BrowseDialog/Dashboar	dControl/UserRepo	ositoriesLis	t.cs\n		
	@@15,7.	+15,7·@@·namespace·GitUI.CommandsDialogs	.BrowseDialog.Das	shboardContr	ol \n		
15 15	····publ	<pre>ic partial class UserRepositoriesList :</pre>	GitExtensionsCont	trol\n			
16 16	••••• { \n						
17 17		private readonly TranslationString grou	pRecentRepositor	ies = new("R	ecent reposi	tories	s")
18		private readonly TranslationString _ repo	sitorySearchPlace	eholder.=.ne	w("Search•re	posito	ori.
10 10	+	private readonly TranslationStringrepo	sitorysearchPlace	enoider·=·ne	w(Search•re	posito	ori
19 19		private readonly TranslationString _grou	teCategoryCartic	Actions);\h			
20 20		privace readonly mansfactorschingdele	cecaregory caption	1 new(//i	1		>
-							-

19.3 Create pull requests

Create a pull request to an active remote.

Note: Many GitHub repos has templates that should be used when creating PR, if so this form should not be used.

🗙 Create	Pull Request		_		\times
Target rep	ository:	gitextensions/gitextensions			~
Your bran	ch:	feature/suppress-VSTHRD110			\sim
Target bra	nch:	master			\sim
Pull requ	uest data				
Title:	ConEmu su	ppress build warning VSTHRD110			
Body:	Body of PR				
				Create	

19.4 Add upstream remote

Add the repo the current repo was forked from as *upstream*.

19.5 Settings

See *Plugins* to configure access.

Translations

20.1 Change language

In the settings dialog the language can be chosen.



20.2 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 and configured in *Shell extension*.

	GitEx Browse			
\bigcirc	GitEx Commit			
志	Git Extensions		₽	Pull
		1	a	Push
		Į	1	View stash
		Ŧ	+++	View changes
		Q	8	Checkout branch
		Ę	N	Checkout revision
		Q	۹	Create branch
		Ŧ	-+-+	Open with difftool
		ų	C)	File history
		3	2	Reset file changes
		-	+	Add files
				Apply patch
		-	Ô	Settings

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

8	GitEx Clone		
8	GitEx Create new repository		
ㅎ	Git Extensions	Ô	Settings

Other tools

Git Extensions can be started from Windows Explorer and other tools.

22.1 Visual Studio Code

3rd party extensions are available from the Marketplace pmiossec and ForEvolve.

22.2 Visual Studio

The Visual Studio extension is available from Visual Studio application or Marketplace.

22.2.1 Menu

Most functions can be started from the Extensions / GitExt menu in Visual Studio.



22.2.2 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	
✓ 5 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	Taxaa ka ka a Charana a
🖾 Hotkeys	Truncate long filenames None V
Chall antipation	

22.2.3 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

23.1 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.

🗙 Commandline usage	—		\times
Supported commandline arguments for gitex.cmd / gitex (located in the same folder as	s GitExte	ensions.ex	(e):
[path] browse [path] [-filter=] [pathFilter= <filepath [-commit=<selectedsha>[,<firstsha>]]</firstsha></selectedsha></filepath 	>]		
about add [filename] addfiles [filename]			
apply [filename] applypatch [filename] blame filename			
branch checkout checkout			
checkoutrevision cherry			
cleanup clone [path] commit [quiet] [message commitmessage]		
difftool filename filehistory filename fileeditor filename			
formatpatch gitignore help (shows this dialog)			
init [path] merge [branch name] mergeconflicts [quiet]			
mergetool [quiet] openrepo [path] [-filter=] pull [rebase] [merge] [fetch] [quiet] [remoteb	oranch na	mel
push [quiet] rebase [branch name] remotes			
reset revert filename			
seatchine settings stash			
synchronize [rebase] [merge] [fetch] [q tag viewdiff	uietj		
viewpatch [filename]			



Appendix

24.1 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	<i>\$ git checkout <name></name></i>
Create branch	\$ git branch <name></name>
Delete branch	<i>\$ git branch -d <name></name></i>
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)	git format-patch –M –C origin
Apply patch without merging	<i>\$ git apply c:/patch/01-emp.patch</i>
Merge patch	\$ git am3waysignoff c:/patch/01-emp.patch
Solve conflicts (add -tool=kdiff3 if no mergetool is	
specified)	\$ oit mergetool
	\$ git an 3way resolved
	\$ gu um Sway resolvea
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)	
	<i>\$ git mergetool</i>
	\$ git commit
24.1. Git Cheat Sheet	128
Push changes (in branch \$ git push c:/demol master	\$ git push c:/demo1
master: <new>)</new>	ϕ , 11 M c
Blame	\$ git blame $-M - w < filename >$
Hein	$1 \times 0 \times c \circ m m a n a \times - n \rho n$

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		