

Using GIT in IDEs and UI based tools

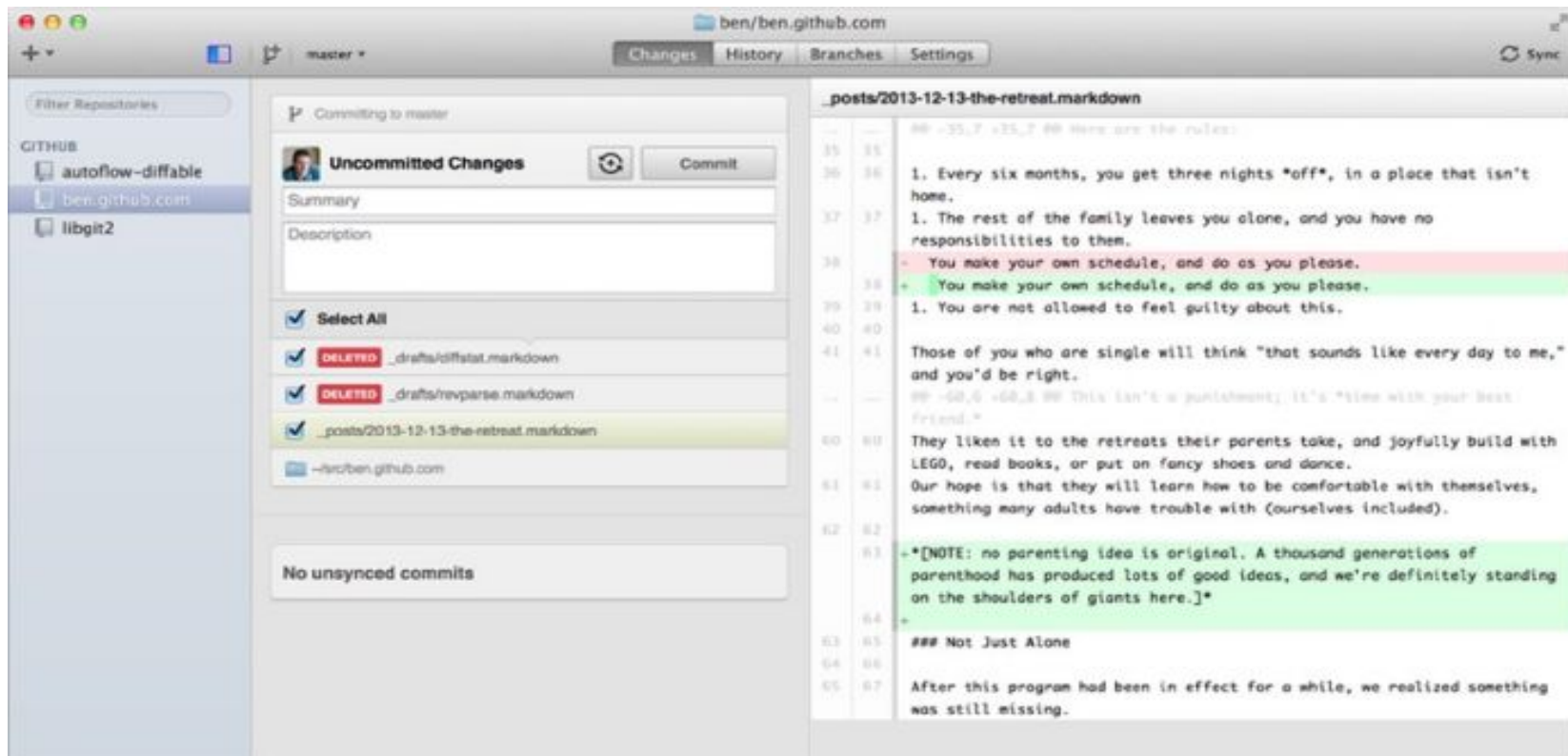
GITHUB

Github is a git repository hosting service, which helps every team members to work together on the project from anywhere and makes it easy for them to collaborate .

GitHub for Mac and Windows

GitHub has created two workflow-oriented Git clients: one for Windows and one for Mac. These clients are a good example of workflow-oriented tools.

Github for MAC :



GitHub for Windows :

The screenshot displays the GitHub for Windows application interface. On the left, a sidebar shows the repository list with a search filter and two repositories: `ben.github.com` and `libgit2`. The main area is divided into three sections:

- Uncommitted changes:** Includes a "Summary" field, a "Description" field, and a "Commit to master" button indicating 3 files to be committed.
- History:** A list of recent commits by Ben Straub, including "Verifying keybase", "Update about page", "Add twitter and github links to cv", "Add physical address", "Re-order and update content", "Port CV from old site", and "Add headshot for public consumption".
- Files to commit:** A list of files to be committed, including `_drafts\diffstat.markdown` (DELETED), `_drafts\revparse.markdown` (DELETED), and `_posts\2013-12-13-the-retreat.markdown`.

The right pane shows a diff view for the selected file, `_posts\2013-12-13-the-retreat.markdown`. The diff highlights changes in a green background, showing the addition of a new paragraph and a note. The diff content is as follows:

```
@@ -1,4 +1,4 @@
- ---
+ ---
+
+ layout: post-no-feature
+ title: "The Retreat"
+ comments: true
@@ -60,6 +60,8 @@ This isn't a punishment; it's "time with your best friend."
They liken it to the retreats their parents take, and joyfully build with LEGO,
read books, or put on fancy shoes and dance.
Our hope is that they will learn how to be comfortable with themselves, something
many adults have trouble with (ourselves included).
+
+ *[NOTE: This is hardly original. We're standing on the shoulders of a thousand
+ generations of parents that came before us.]
+
+ ### Not Just Alone
+
+ After this program had been in effect for a while, we realized something was
+ still missing.
```

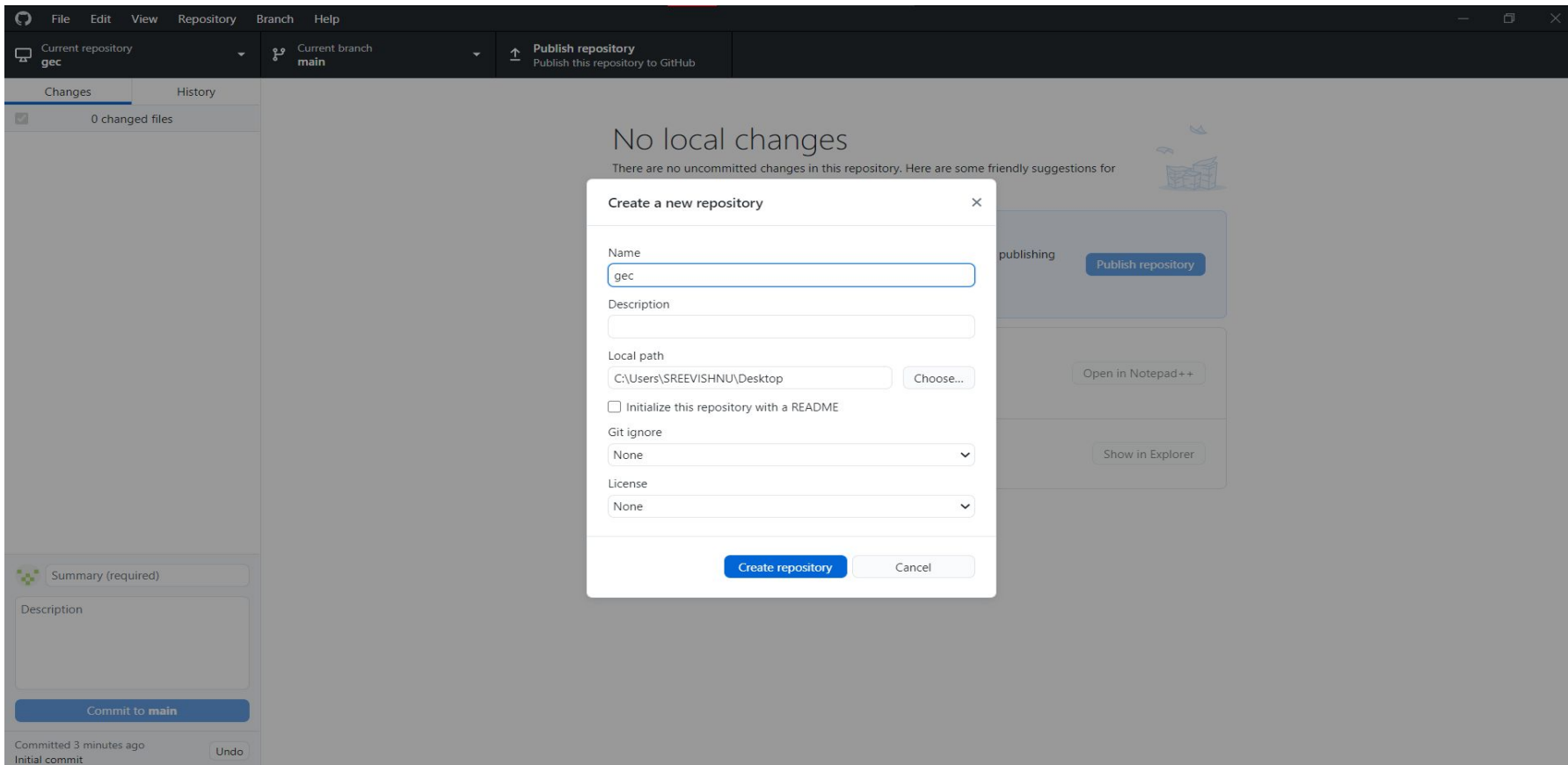
Installation

GitHub for Windows can be downloaded from <https://windows.github.com> .

GitHub for Mac can be downloaded from <https://mac.github.com> .

- Once installation is complete, the next step is to sign in with your GitHub credentials.
- If you don't already have a GitHub login, head over to the signup page and create a new account.
- Once you successfully sign in, you are ready to start using GitHub Desktop.
- Create a new repository on GitHub.
- The next step is to give the tool some repositories to work with. The client shows you a list of the repositories you have access to on GitHub, and can clone them in one step.
- If you already have a local repository, just drag its directory from the Finder or Windows Explorer into the GitHub client window, and it will be included in the list of repositories

- On the left is the list of repositories the client is tracking; you can add a repository (either by cloning or attaching locally) by clicking the “+” icon at the top of this area.
- In the center is a commit-input area, which lets you input a commit message, and select which files should be included. (On Windows, the commit history is displayed directly below this; on Mac, it’s on a separate tab.)
- On the right is a diff view(History), which shows what’s changed in your working directory, or which changes were included in the selected commit.
- The last thing to notice is the “Sync(Fetch Origin)” button at the top-right, which is the primary way you interact over the network.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Always available in the toolbar for local repositories or `Ctrl` `P`

[Publish repository](#)

Open the repository in your external editor

Select your editor in [Options](#)

Repository menu or `Ctrl` `Shift` `A`

[Open in Notepad++](#)


View the files of your repository in Explorer


Repository menu or `Ctrl` `Shift` `F`


[Show in Explorer](#)[Commit to main](#)


Committed 8 minutes ago
Initial commit




[Undo](#)

 File Edit View Repository Branch Help



 Current repository
gec

 Current branch
main


 Fetch origin
Last fetched a minute ago

Changes 1 History doc3.txt   


☐ 1 changed file

 doc3.txt 

		@@ -0,0 +1 @@
	1	+h1

 ,:

Description



Commit to main

Recommended Workflow

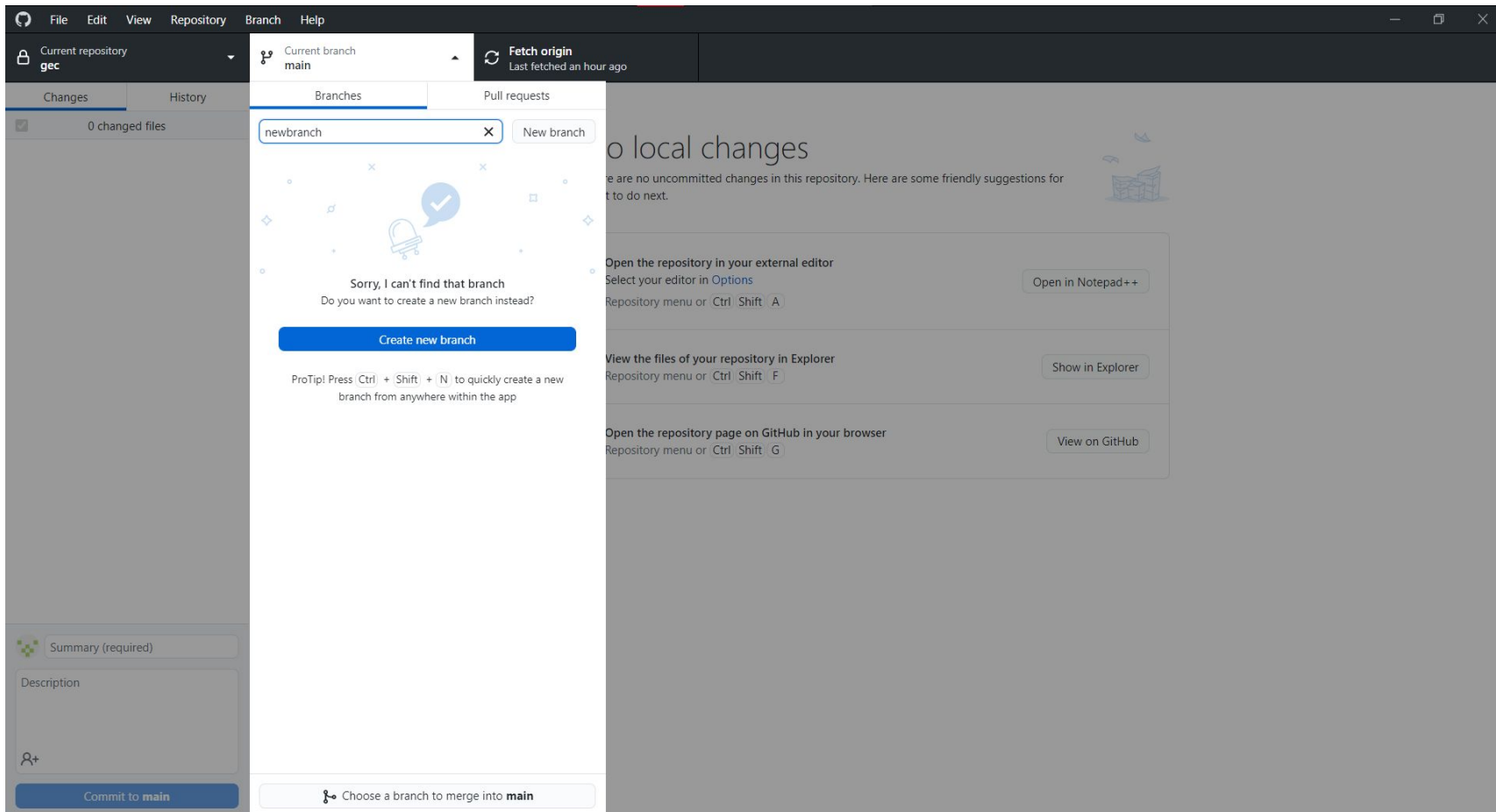
Once it's installed and configured, you can use the GitHub client for many common Git tasks. The intended workflow for this tool is sometimes called the “GitHub Flow”.

- **Branch :**

Suppose you are working on an application and you want to add a new feature to the app you can create a new branch and build the new feature on that branch.

By default you always work on the master(main) branch.

On Windows, this is done by typing the new branch's name in the branch-switching widget



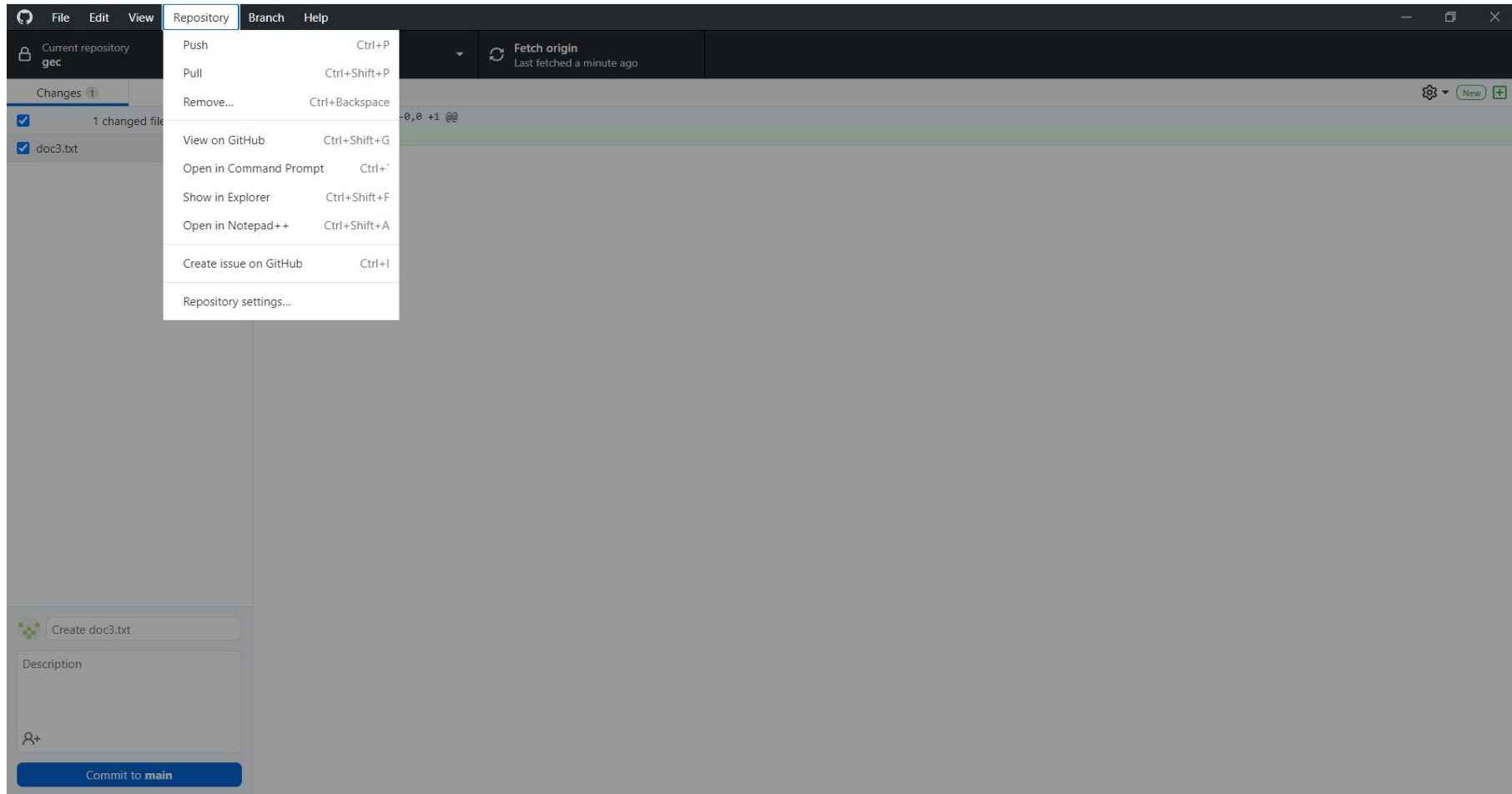
- Once your branch is created, making new commits is fairly straightforward.
- Make some changes in your working directory, and when you switch to the GitHub client window, it shows you which files changed. Enter a commit message, select the files you'd like to include, and click the “Commit” button
- The main way you interact with other repositories over the network is through the “Sync” (Fetch Origin) feature. Git internally has separate operations for pushing, fetching, merging, and rebasing, but the GitHub clients collapse all these into one multi-step feature.

- **Git pull :**

Git pull is used to fetch and merge changes from the remote repository to the local repository.

- **Git push:**

Git push command is used to upload local repository content to a remote repository.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



Open the repository in your external editor

Select your editor in [Options](#)

Repository menu or `Ctrl` `Shift` `A`

[Open in Notepad++](#)

View the files of your repository in Explorer


Repository menu or `Ctrl` `Shift` `F`

[Show in Explorer](#)


Open the repository page on GitHub in your browser

Repository menu or `Ctrl` `Shift` `G`

[View on GitHub](#)

 Summary (required)

Description



Commit to **main**

gitk

gitk

- ➡ When we install Git, we also get its visual tools gitk and git-gui.
- ➡ gitk is a graphical history viewer.
- ➡ It is the tool to use when you are trying to find something that happened in the past or visualize your project's history .

How to invoke gitk

- It is easiest to invoke gitk from command-line.

cd into a Git repository, then give the command :

\$ gitk [git log options]

- Git accepts many command-line options, most of them are passed through

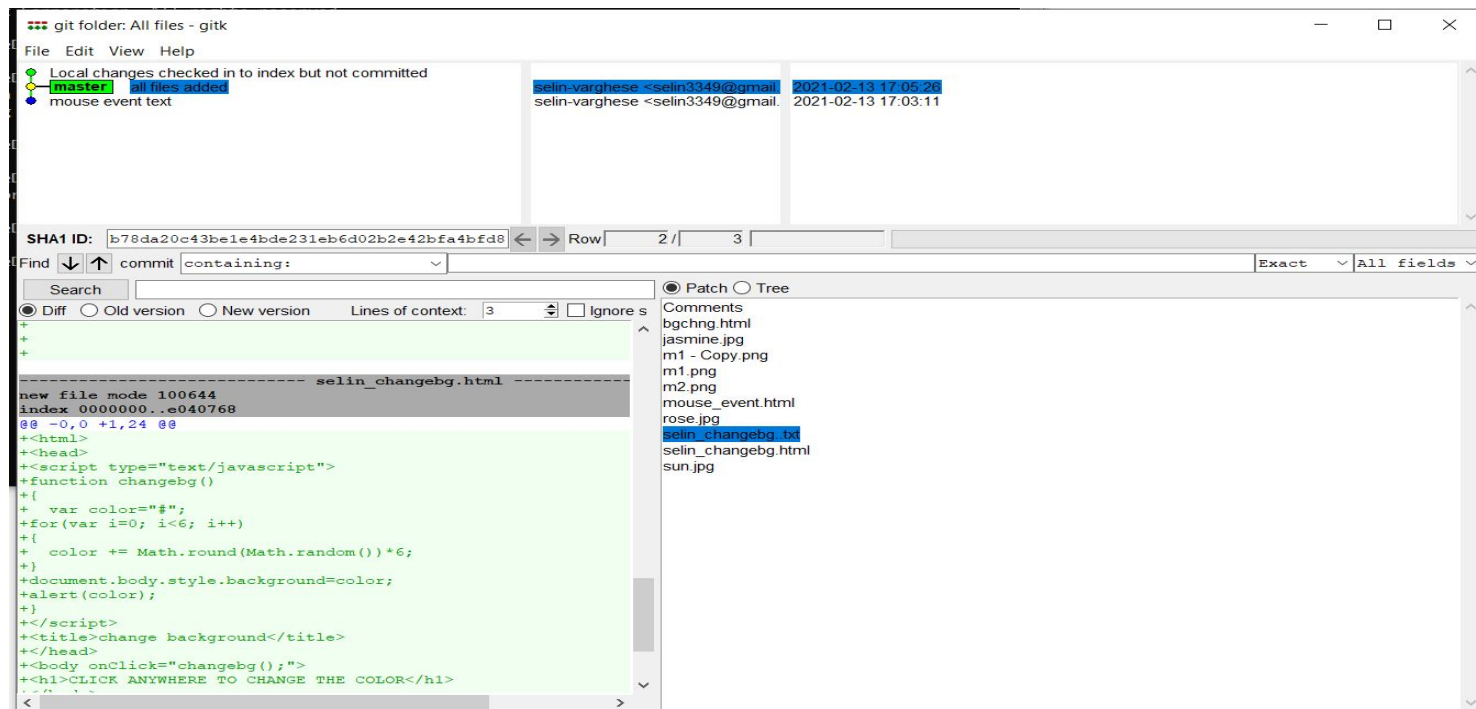
The git log action

- One of the most useful is :

--all flag

It tells gitk to show commits reachable from any ref, not just HEAD.

```
C:\Users\santo\OneDrive\Desktop\git folder>gitk
```



C:\Users\santo\OneDrive\Desktop\git folder>gitk --all

git folder: --all - gitk

File Edit View Help

branch2 main branch1 addition
Local changes checked in to index but not committed
master all files added
mouse event text
remotes/origin/main Initial commit

selin-varghese <selin3349@gmail.com> 2021-02-13 17:21:21
selin-varghese <selin3349@gmail.com> 2021-02-13 17:05:26
selin-varghese <selin3349@gmail.com> 2021-02-13 17:03:11
santeena-varghese <77263875+sante> 2021-02-12 22:31:21

SHA1 ID: b78da20c43be1e4bde231eb6d02b2e42bfa4bfd8 Row 3 / 5

Find commit containing: Exact All fields

Search

Diff Old version New version Lines of context: 3 Ignore space chan

Author: selin-varghese <selin3349@gmail.com> 2021-02-13 17:05:26
Committer: selin-varghese <selin3349@gmail.com> 2021-02-13 17:05:26
Parent: 50fd606c3a08975bd015faaf94460e7adc6b9266 (mouse event text)
Child: f39d707969639141ee7e4e321198464a98f0199 (branch1 addition)
Branches: branch2, main, master
Follows:
Precedes:

all files added

new file mode 100644
index 0000000..238db17
00 -0,0 +1,19 00
+<!DOCTYPE HTML>
+<html>
+<head>
+<title>
+changing the background color
+</title>
+</head>
+<body style = "text-align:center;" onclick="changeBg()">
+<h1 style = "color:blue;" > Welcome

Comments

bgchgng.html
jasmine.jpg
m1 - Copy.png
m1.png
m2.png
mouse_event.html
rose.jpg
selin_changebg.txt
selin_changebg.html
sun.jpg

C:\Users\santo\OneDrive\Desktop\git folder>gitk --graph

The screenshot shows the gitk graphical user interface for a repository named 'libgit2'. The top menu bar includes File, Edit, View, and Help. On the left, a commit graph shows the history of the repository, with branches 'development' and 'remotes/origin/development' highlighted. The main window displays commit details for the selected commit (SHA1 ID: b76b5d34275fe33192358d4ea1ae98e31efc2a1). The commit message is 'Improve test of submodule name sorting'. The right pane shows a list of commits, with the selected commit highlighted. The bottom pane shows a diff view of the file 'tests/diff/submodules.c', with the 'Patch' view selected. The diff shows changes to the 'test_diff_submodules__submod2_index_to_wd(void)' function.

File Edit View Help

Local uncommitted changes, not checked in to index
development remotes/origin/development refspec: git_refspec_parse() does not
Merge pull request #2208 from libgit2/vmg/mempack
In-memory packing backend
Merge pull request #2226 from libgit2/rb/submodule-sorting-fix
Improve test of submodule name sorting
Cleanups
Fix submodule sorting in workdir iterator
Add faster git_submodule_is_submodule check
Merge pull request #2229 from linquize/Wdeclaration-after-statement
Add CFLAGS -Wdeclaration-after-statement

SHA1 ID: b76b5d34275fe33192358d4ea1ae98e31efc2a1 Row 6 / 1359

Find commit containing: Search

Diff Old version New version Lines of context: 3 Ignore space change Line diff

Author: Russell Belfer <rb@github.com> 2014-03-31 13:33:11
Committer: Russell Belfer <rb@github.com> 2014-03-31 13:33:11
Parent: 7dc42a55f5fd61e8e8de472ec54ccc0613e23c (Cleanups)
Child: d67397d0c82fab82a1e6883107c97c4e133a911 (Merge pull request #2226 from libgit2/rb/submodule-sorting-fix)
Branches: development, remotes/origin/development
Follows: v0.20.0
Precedes:

Improve test of submodule name sorting

tests/diff/submodules.c

```
index ead5c71..2881f74 100644
@@ -182,6 +182,8 @@ void test_diff_submodules__submod2_index_to_wd(void)
    "<UNTRACKED>", /* not */
    "diff --git a/sm_changed_file b/sm_changed_file\nindex 4800958..4800958 :
    "diff --git a/sm_changed_head b/sm_changed_head\nindex 4800958..3d9386c :
    "<UNTRACKED>", /* sm_changed_head */
    "<UNTRACKED>", /* sm_changed_head */
    "diff --git a/sm_changed_index b/sm_changed_index\nindex 4800958..4800958 :
    "diff --git a/sm_changed_untracked_file b/sm_changed_untracked_file\nindex
    "diff --git a/sm_missing_commits b/sm_missing_commits\nindex 4800958..5e
@@ -190,6 +192,10 @@ void test_diff_submodules__submod2_index_to_wd(void)
```


GIT GUI

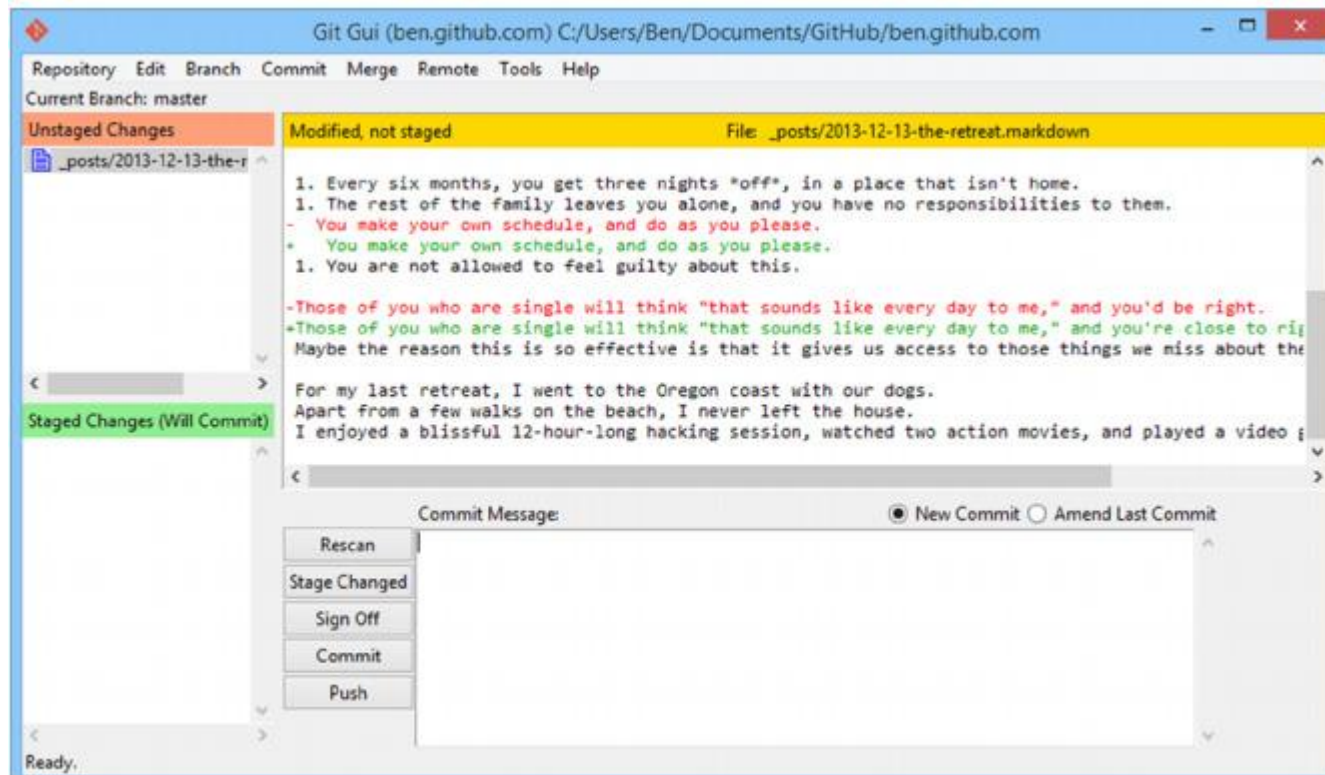
What is GIT GUI?

- Git Gui focuses on allowing users to make changes to their repository by making
 - * New commits,
 - * Amending existing ones,
 - * Creating branches,
 - * Performing local merges, and
 - * Fetching/Pushing to remote repositories.
- * Git-Gui, on the other hand, is primarily a tool for crafting commits. It, too, is easiest to invoke from the command.

```
$ git gui
```

And it looks something like this:

THE GIT COMMIT TOOL



- On the left is the index; unstaged changes are on top, staged changes on the bottom. You can move entire files between the two states by clicking on their icons, or you can select a file for viewing by clicking on its name.
- At top right is the diff view, which shows the changes for the currently-selected file. You can stage individual hunks (or individual lines) by right-clicking in this area.
- At the bottom right is the message and action area. Type your message into the text box and click “Commit” to do something similar to git commit.
- You can also choose to amend the last commit by choosing the “Amend” radio button, which updates the “Staged Changes” area with the contents of the last commit.
- Then you can simply stage or unstage some changes, alter the commit message, and click “Commit” again to replace the old commit with a new one.
- *Gitk and Git-Gui* are examples of task-oriented tools. Each of them is tailored for a specific purpose

Git in Visual Studio

- Starting with Visual Studio 2013 Update 1, Visual Studio users have a Git client built directly into their IDE.
- Visual Studio has had source-control integration features for quite some time.
- But they were oriented toward centralized, file-locking systems.
- Git was not a good match for this workflow.

- Visual Studio 2013's Git support has been separated from this older feature, and the result is a much better fit between Studio and Git.
- To locate the feature, open a project that's controlled by Git (or just `git init` an existing project), and select View ➤ Team Explorer from the menu.
- You'll see the "Connect" view, which looks a bit like this:

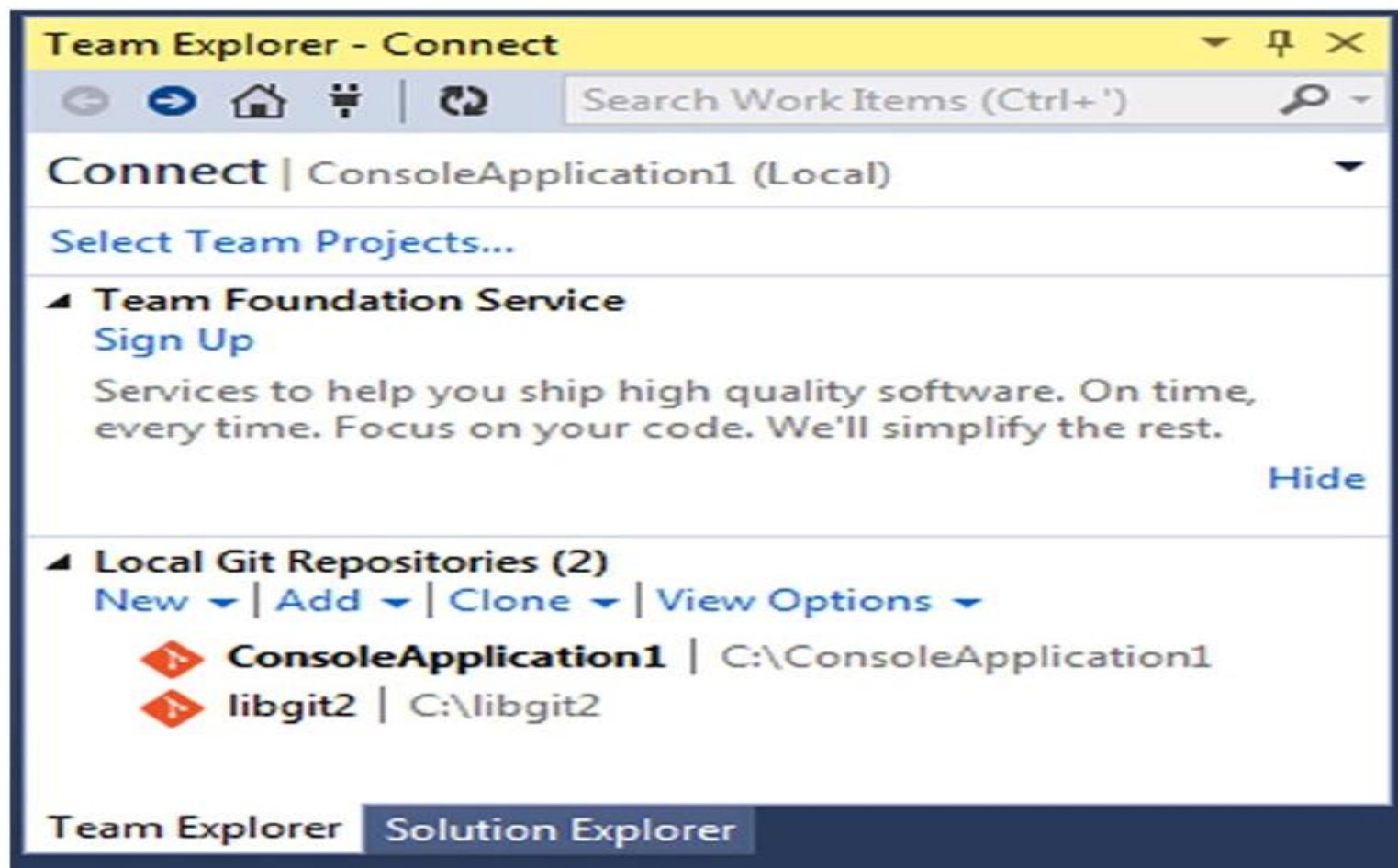


Figure A-7. Connecting to a Git repository from Team Explorer

- Visual Studio remembers all the projects you've opened that are Git-controlled, and they're available in the list at the bottom.
- If you don't see the one you want there, click the "Add" link and type in the path to the working directory.
- Double-clicking one of the local Git repositories leads you to the Home view
- This is a hub for performing Git actions; when you're writing code, you'll probably spend most of your time in the Changes view, but when it comes time to pull down changes made by your teammates, you'll use the Unsynced Commits and Branches views
- Visual Studio now has a powerful task-focused UI for Git. It includes a linear history view, remote commands and many other capabilities

GIT IN ECLIPSE

Eclipse

- ☐ It is an integrated development environment (IDE) used in computer programming.
- ☐ It contains a base workspace and an extensible plug-in system for customizing the environment.
- ☐ It is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Perl, Fortran, PHP, JavaScript, etc.

Git in Eclipse

- Eclipse ships with a plugin called Egit, which provides a fairly-complete interface to Git operations.
- It's accessed by switching to the Git Perspective (Window ➤ Open Perspective ➤ Other..., and select Git).
- EGit comes with plenty of great documentation, which you can find by going to Help ➤ Help Contents, and choosing the EGit Documentation node from the contents listing.

Egit

- Egit is the Git integration for Eclipse.
- The Egit project is implementing Eclipse tooling for the Jgit Java implementation of Git.
- You can create local repositories, even before you share them with other people.
- In this way you can version your work locally.

Eclipse's Egit Environment

The screenshot displays the Eclipse IDE interface with the Egit plugin. The top toolbar includes icons for Git operations like commit, push, pull, and fetch. The left sidebar shows the 'Git Repositories' view for the 'libgit2' repository, listing branches (Local, Remote Tracking), tags, references, and the working directory. The main editor area is divided into two panes. The top pane shows the 'Properties' view for the selected commit, displaying a table of commit history with columns for ID, Message, Author, Date, Committer, and Committer. The bottom pane shows the 'History' view, displaying a list of commits with their IDs and messages. The 'libgit2' repository is currently checked out to the 'master' branch, which is linked to the 'HEAD' pointer. The commit history table shows the following entries:

ID	Message	Author	Authored Date	Committer	Committer Date
4b0a36e	merge pull request #2255 from libgit2	Vicent Marti	5 months ago	Vicent Marti	5 months ago
43cb8b3	libgit2 0.20.0 "anmeldung"	Vicent Marti	5 months ago	Vicent Marti	5 months ago
1b3fe73	Formatting fix for ci/ci_acquire_lock	Carlos Martin Nieto	5 months ago	Carlos Martin Nieto	5 months ago
e479628	Merge pull request #1966 from nickh/patch_content_offsets	Vicent Marti	5 months ago	Vicent Marti	5 months ago
7146eff	util: NetBSD doesn't have posix, either	Alessandro Ghetti	5 months ago	Alessandro Ghetti	5 months ago
963edd9	Merge pull request #1969 from libgit2/ntk/fix/drop_nul_token	Vicent Marti	5 months ago	Vicent Marti	5 months ago
47a9a62	tests: Drop unrelated comment	nul token	5 months ago	nul token	5 months ago
65f6785	Merge pull request #1968 from libgit2/ntk/fix/bad_nul_token	Vicent Marti	5 months ago	Vicent Marti	5 months ago
e544a5b	index: free the index on git_index_open() failure	nul token	5 months ago	nul token	5 months ago
bd15b51	tree-cache: fix error message typo	nul token	5 months ago	nul token	5 months ago
a5d7318	tree-cache: Don't segfault upon corruption	nul token	5 months ago	nul token	5 months ago
3d52334	tree-cache: Don't segfault upon corruption	nul token	5 months ago	nul token	5 months ago
792d4e5	tree-cache: Don't segfault upon corruption	nul token	5 months ago	nul token	5 months ago

The commit message for the selected commit (43cb8b3) is: "libgit2 0.20.0 'anmeldung'". The commit history view shows the following commits:

- 43cb8b32478b1b29904874349ec22cb5372e152: (libgit2 0.20.0 "anmeldung")
- Parent: e479628a81eddaf357bd3b49526eb49998edcef (Merge pull request #1966 from nickh/patch_content_offsets)
- Child: 43cb8b32478b1b29904874349ec22cb5372e152: (libgit2 0.20.0 "anmeldung")

The 'libgit2' repository is currently checked out to the 'master' branch, which is linked to the 'HEAD' pointer. The commit history view shows the following commits:

- 43cb8b32478b1b29904874349ec22cb5372e152: (libgit2 0.20.0 "anmeldung")
- Parent: e479628a81eddaf357bd3b49526eb49998edcef (Merge pull request #1966 from nickh/patch_content_offsets)
- Child: 43cb8b32478b1b29904874349ec22cb5372e152: (libgit2 0.20.0 "anmeldung")

Git in Bash

INTRODUCTION

- Git Bash is an application for Microsoft Windows environments.
- If you are a bash user, you can tap into some of your shell's features to make your experience with Git a lot friendlier.
- Git actually ships with plugins for several shells, but its not turned on by default.

STEPS

- Get the copy of the, contrib/completion/git-completion.bash file out of the Git source code.
- Change your directory to a git repository, and type: `$ git chec<tab>` and Bash will auto-complete to Git checkout.
- This work with all git subcommands, command-line parameter, remote and ref frames.
- It helpful to customize your prompt to show the information about the current directories git repository.
- This can be simple or complex.

Generally a few key pieces of information that most people want,

The current branch

Status of working directory

To add these to your prompt just copy, `contrib/completion/git-prompt.sh` file

Add something to your .bashrc

```
. ~/git-prompt.sh
```

```
export GIT_PS1_SHOWDIRTYSTATE=1
```

```
export PS1=' \w$(__git_ps1"(%s)")\$'
```

\w - print the current working directory

\\$ - print the \$ part of the prompt

__git_ps1"(%s)" - calls the function by git prompt .sh

Now your bash prompt look like this

```
~/src/libgit2 (development *)$
```



GIT IN ZSH

Introduction

- Git also ships with a tab-completion library for Zsh.
- Just copy contrib/completion/git-completion.zsh to your home directory and source it from your .zshrc.
- Zsh interface is bit more powerful than bash due to it provides:
 - ❑ Spelling correction
 - ❑ Plugin supports
 - ❑ Better Themes

There are also some commands that helps Zsh interface more powerful than bashs:

\$ git che <tab>

check-attr -- display gitattributes information

check-ref-format -- ensure that a reference name is well formed

checkout -- checkout branch or paths to working tree

checkout-index -- copy files from index to working directory

cherry -- find commits not merged upstream

cherry-pick -- apply changes introduced by some existing commits

- **Tab Completion** help you speed up typing commands.
- Just hit **Tab** while typing a **command**, option, or file name and the shell environment will automatically **complete** what you're typing or suggest options to you.
- In zsh ambiguous tab-completions aren't just listed; they have helpful descriptions, and you can graphically navigate the list by repeatedly hitting Tab.
- This works with Git commands, their arguments, and names of things inside the repository (like refs and remotes), as well filenames and all the other things Zsh knows how to tab-complete.

- Zsh happens to be fairly compatible with Bash when it comes to prompt customization, but it allows you to have a right-side prompt as well.
- To include the branch name on the right side, add these lines to your ~/.zshrc file:

```
setopt prompt_subst
```

```
. ~/git-prompt.sh
```

```
export RPROMPT='${__git_ps1 "%s"}'
```

- This results in a display of the current branch on the right-hand side of the terminal window, whenever your shell is inside a Git repository

- Zsh is powerful enough that there are entire frameworks dedicated to making it better.
- One of them is called “oh-my-zsh”, and it can be found at <https://github.com/robbyrussell/oh-my-zsh>.
- oh-my-zsh’s plugin system comes with powerful git tab-completion, and it has a variety of prompt themes, many of which display version-control data



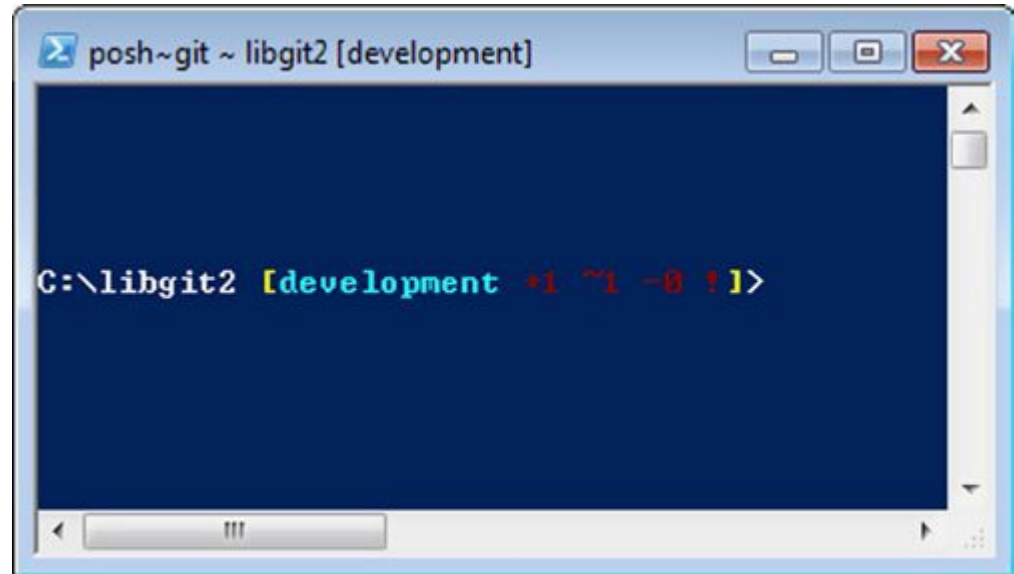
Figure A-11. Customized zsh prompt



Figure A-12. An example of an oh-my-zsh theme

GIT in Powershell

- ★ Capable of a customized Git experience
- ★ It can be accessed from a package called Posh-Git
(<https://github.com/dahlbyk/posh-git>)
- ★ Provides powerful tab-completion facilities
- ★ And also an enhanced prompt to help you stay on top of your repository status.



Powershell for Windows users

- ★ Posh-Git is included by default, if Git is installed in Windows (GitHub for Windows), and the following lines need to be added to your profile.ps1 (which is usually located in: C:\Users\<username>\Documents\WindowsPowerShell):
- ★ . (Resolve-Path "\$env:LOCALAPPDATA\GitHub\shell.ps1")
- ★ . \$env:github_posh_git\profile.example.ps1

Powershell for non -Windows users

If you're not a GitHub for Windows user, just download a Posh-Git release from (<https://github.com/dahlbyk/posh-git>), and uncompress it to the WindowsPowershell directory. Then open a

Powershell prompt as the administrator, and do this:

```
> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser -Confirm  
> cd ~\Documents\WindowsPowerShell\posh-git  
> .\install.ps1
```

This adds the proper line to your profile.ps1 file, and Posh-Git will be active the next time you open your prompt.

THANK YOU!