

O'REILLY®

Head First

C#

A Learner's Guide to
Real-World Programming
with C# and .NET

Andrew Stellman
& Jennifer Greene

Fifth Edition
Also covers .NET MAUI & Unity



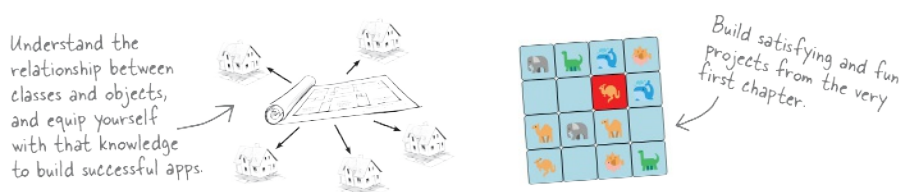
A Brain-Friendly Guide

Head First

C#

What will you learn from this book?

Create apps, games, and more using this engaging, highly visual introduction to C#, .NET, and software development. You'll learn how to use classes and object-oriented programming, create 3D games in Unity, and query data with LINQ. And you'll do it all by solving puzzles, completing hands-on exercises, and building real-world applications. Interested in a development career? You'll learn important development techniques and ideas—just like many others who've learned to code with this book and are now professional developers, team leads, coding streamers, and more. There's no experience required except the desire to learn. And this is the best place to start.



What's so special about this book?

If you've read a Head First book, you know what to expect: a visually rich format designed for the way your brain works. If you haven't, you're in for a treat. With this book, you'll learn C# through a multisensory experience that engages your mind—rather than a text-heavy approach that puts you to sleep.

"Thank you so much! Your books have helped me to launch my career."

—**Ryan White**
Game Developer

"In a sea of dry technical manuals, *Head First C#* stands out as a beacon of brilliance. Its unique teaching style not only imparts essential knowledge but also sparks curiosity and fuels passion for coding. An indispensable resource for beginners!"

—**Gerald Versluis**
Senior Software Engineer
at Microsoft

"Andrew and Jennifer have written a concise, authoritative, and, most of all, fun introduction to C# development."

—**Jon Galloway**
Senior Program Manager on
the .NET Community Team
at Microsoft

C#/.NET

US \$79.99

CAN \$99.99

ISBN: 978-1-098-14178-3



O'REILLY®

Praise for Head First C#

“In a sea of dry technical manuals, *Head First C#* stands out as a beacon of brilliance. Its unique teaching style not only imparts essential knowledge but also sparks curiosity and fuels passion for coding. An indispensable resource for beginners!”

—**Gerald Versluis, Senior Software Engineer at Microsoft**

“*Head First C#* started my career as a software engineer and backend developer. I am now leading a team in a tech company and an open source contributor.”

—**Zakaria Soleymani, Development Team Lead**

“Thank you so much! Your books have helped me to launch my career.”

—**Ryan White, Game Developer**

“If you’re a new C# developer (welcome to the party!), I highly recommend *Head First C#*. Andrew and Jennifer have written a concise, authoritative, and most of all, fun introduction to C# development. I wish I’d had this book when I was first learning C#!”

—**Jon Galloway, Senior Program Manager on the .NET Community Team, Microsoft**

“Not only does *Head First C#* cover all the nuances it took me a long time to understand, it has that Head First magic going on where it is just a super fun read.”

—**Jeff Counts, Senior C# Developer**

“*Head First C#* is a great book with fun examples that keep learning interesting.”

—**Lindsey Bieda, Lead Software Engineer**

“*Head First C#* is a great book, both for brand-new developers and developers like myself coming from a Java background. No assumptions are made as to the reader’s proficiency, yet the material builds up quickly enough for those who are not complete newbies—a hard balance to strike. This book got me up to speed in no time for my first large-scale C# development project at work—I highly recommend it.”

—**Shalewa Odusanya, Principal**

“*Head First C#* is an excellent, simple, and fun way of learning C#. It’s the best piece for C# beginners I’ve ever seen—the samples are clear, the topics are concise and well written. The mini-games that guide you through the different programming challenges will definitely stick the knowledge to your brain. A great learn-by-doing book!”

—**Johnny Halife, Partner**

“*Head First C#* is a comprehensive guide to learning C# that reads like a conversation with a friend. The many coding challenges keep it fun, even when the concepts are tough.”

—**Rebeca Dunn-Krahn, Founding Partner, Sempahore Solutions**

Praise for Head First C#

“I’ve never read a computer book cover to cover, but this one held my interest from the first page to the last. If you want to learn C# in depth and have fun doing it, this is THE book for you.”

—**Andy Parker, fledgling C# Programmer**

“It’s hard to really learn a programming language without good, engaging examples, and this book is full of them! *Head First C#* will guide beginners of all sorts to a long and productive relationship with C# and the .NET Framework.”

—**Chris Burrows, Software Engineer**

“With *Head First C#*, Andrew and Jenny have presented an excellent tutorial on learning C#. It is very approachable while covering a great amount of detail in a unique style. If you’ve been turned off by more conventional books on C#, you’ll love this one.”

—**Jay Hilyard, Director and Software Security Architect, and author of
*C# 6.0 Cookbook***

“I’d recommend this book to anyone looking for a great introduction into the world of programming and C#. From the first page onward, the authors walk the reader through some of the more challenging concepts of C# in a simple, easy-to-follow way. At the end of some of the larger projects/labs, the reader can look back at their programs and stand in awe of what they’ve accomplished.”

—**David Sterling, Principal Software Developer**

“*Head First C#* is a highly enjoyable tutorial, full of memorable examples and entertaining exercises. Its lively style is sure to captivate readers—from the humorously annotated examples to the Fireside Chats, where the abstract class and interface butt heads in a heated argument! For anyone new to programming, there’s no better way to dive in.”

—**Joseph Albahari, inventor of LINQPad, and coauthor of *C# 12 in a Nutshell* and
*C# 12 Pocket Reference***

“[*Head First C#*] was an easy book to read and understand. I will recommend this book to any developer wanting to jump into the C# waters. I will recommend it to the advanced developer that wants to understand better what is happening with their code. [I will recommend it to developers who] want to find a better way to explain how C# works to their less-seasoned developer friends.”

—**Giuseppe Turitto, Director of Engineering**

“Andrew and Jenny have crafted another stimulating Head First learning experience. Grab a pencil, a computer, and enjoy the ride as you engage your left brain, right brain, and funny bone.”

—**Bill Mietelski, Advanced Systems Analyst**

“Going through this *Head First C#* book was a great experience. I have not come across a book series which actually teaches you so well.... This is a book I would definitely recommend to people wanting to learn C#.”

—**Krishna Pala, MCP**

Praise for the Head First Approach

“I received the book yesterday and started to read it...and I couldn’t stop. This is definitely très ‘cool.’ It is fun, but they cover a lot of ground and they are right to the point. I’m really impressed.”

—**Erich Gamma, IBM Distinguished Engineer, and coauthor of *Design Patterns***

“One of the funniest and smartest books on software design I’ve ever read.”

—**Aaron LaBerge, SVP Technology & Product Development, ESPN**

“What used to be a long trial and error learning process has now been reduced neatly into an engaging paperback.”

—**Mike Davidson, former VP of Design, Twitter, and founder of Newsvine**

“Elegant design is at the core of every chapter here, each concept conveyed with equal doses of pragmatism and wit.”

—**Ken Goldstein, Executive VP & Managing Director, Disney Online**

“Usually when reading through a book or article on design patterns, I’d have to occasionally stick myself in the eye with something just to make sure I was paying attention. Not with this book. Odd as it may sound, this book makes learning about design patterns fun.

“While other books on design patterns are saying ‘Bueller...Bueller...Bueller...’ this book is on the float belting out ‘Shake it up, baby!’”

—**Eric Wuehler**

“I literally love this book. In fact, I kissed this book in front of my wife.”

—**Satish Kumar**

Head First C#

Wouldn't it be dreamy if there was a C# book that's more fun than memorizing a dictionary? It's probably nothing but a fantasy...



Andrew Stellman
Jennifer Greene

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Head First C# Guide to Git



Have you ever deleted a really important file?

We've all been there—one wrong keystroke and that crucial file you've been working on for hours disappears, and you frantically try to remember if you saved a backup or if there's any way to recover it. With Git, you **never have to worry about losing your code**. Git gives you **version control**, so you can manage changes to your code, collaborate seamlessly with others, and never worry about losing important work again. Luckily, there are great features **built into Visual Studio and Visual Studio Code** that let you use Git to **publish your work to a repository** and get a complete **history of every file** in your project, so recover from those “uh-oh” moments.



My project has a lot of code already! Wouldn't it be dreamy if there was an easy way for me to save everything I've done someplace where I can **save my code, share it, and always find it** any time I want?

You can use Git to save all of your code, and your IDE will help make it easy.

You're going to write a lot of code in this book! Wouldn't it be great if there was a convenient place to put that code so you can always go back to it?

We bet that you'll write some apps that you really like, and you'll want to share them with your friends so they can see the great things you've built.

Do you have a desktop and a laptop? A computer at home and at an office? Wouldn't it be great if you could start a project on one computer, then finish it on another one?

Imagine you're working on a project. You've spent hours getting the code right, and you're really happy with it. Then you make a few changes, and...oh no! Something went completely wrong, your code is broken, and you don't remember exactly what you changed. It would be great if you could see a history of all the changes you made, right?

Git can help you do all of those things!

Here are just a few things Git can do for you

- ★ It can save your files somewhere that you can access them from anywhere, any time.
- ★ It lets you save snapshots of your work so you can go back and see exactly what changed.
- ★ It lets you share your code with anyone (or keep it private!).
- ★ It lets a group of people collaborate on a project together—so if you're learning C# with your friends, you can all work on code together.

Visual Studio and Visual Studio Code make it easy to use Git

Git is a really powerful and flexible tool that can help you save, manage, and share the code and files for all of your projects. It can also be complex and confusing at times! Luckily, Visual Studio and Visual Studio Code both have **built-in Git support** that takes care of the complexity. It helps you with Git, so you can concentrate on your code.

Visual Studio can help you create a new Git repository on GitHub, the popular platform for source code hosting and collaboration.

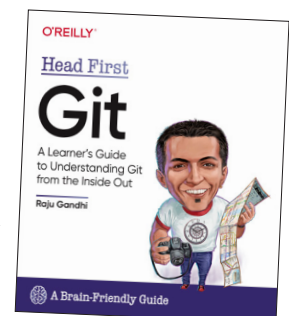
The image shows two overlapping windows from Visual Studio. The background window is titled 'Create a Git repository' and has three main sections: 'Push to a new remote' (with options for GitHub, Azure DevOps, Existing remote, and Local only), 'Initialize a local Git repository' (with fields for Local path, .gitignore template, License template, and a checkbox for 'Add a README.md'), and 'Create a new GitHub repository' (with fields for Account, Owner, Repository name, and Description, and a checkbox for 'Private repository'). The foreground window is titled 'Git Changes' and shows a commit message 'Finished the third part of the animal matching game'. It includes buttons for 'Pull' and 'Push', a 'Commit Staged' dropdown, and a list of files under 'Changes - 38' with a '+ Stage All' button. The file list includes .gitignore, AnimalMatchingGame, AnimalMatchingGame.sln, AnimalMatchingGame.csproj, App.xaml, App.xaml.cs, AppShell.xaml, AppShell.xaml.cs, and MainPage.xaml.

Visual Studio's Git features help you easily add your code to any Git and push changes as often as you want.

We recommend that you *create a GitHub account* and use it to save the code for each of the projects in this book. That will make it easy for you to go back and revisit past projects any time!

This free *Head First C# Guide to Git PDF* gives you a simple, step-by-step guide to saving your code in Git with Visual Studio.

We'll give you everything you need to use Visual Studio to save and share your projects. But there is a lot more that you can do with Git, especially if you're working with large teams! If you're fascinated by what you see and want to do a deep dive into Git, check out *Head First Git* by Raju Gandhi.



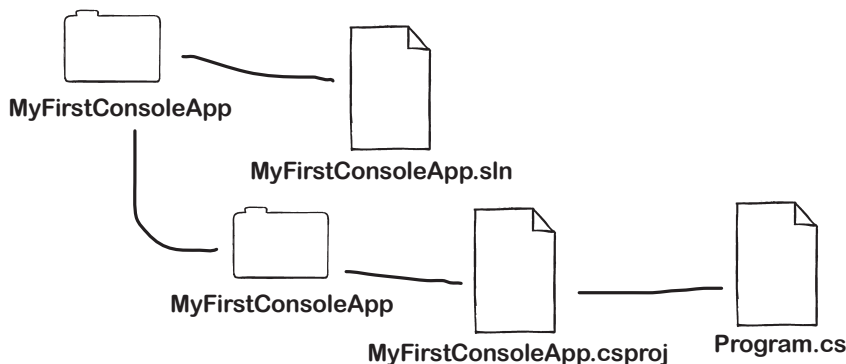
Git stores a history of your project files

Git is a powerful **version control system** that tracks changes to files and projects over time. It allows multiple people to collaborate on the same project, seamlessly integrating their contributions without overwriting each other's work. By keeping a detailed history of changes, Git enables you to easily revert to previous versions, compare differences, and understand the evolution of your code.

Git is used by professional software teams at companies all around the world. Using Git is an important developer skill that you'll use on the job.

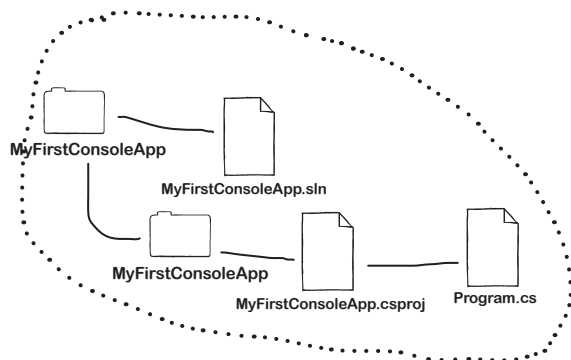
1 You'll start with the code for your project.

Your project's code lives in folders and files. These are the files you want to save in your **repository**, or the storage area Git uses for the history of your files. There are other folders that are created when you build your code—you don't want those in the Git repository,

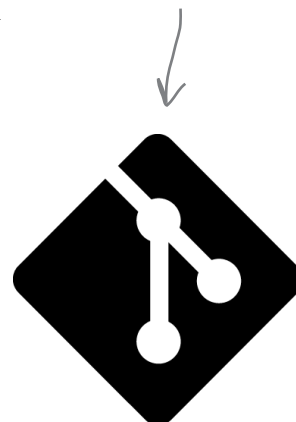


2 You'll create a new Git repository.

Visual Studio and Visual Studio Code both have built-in Git features that will create the new repository for you and publish your code into it.



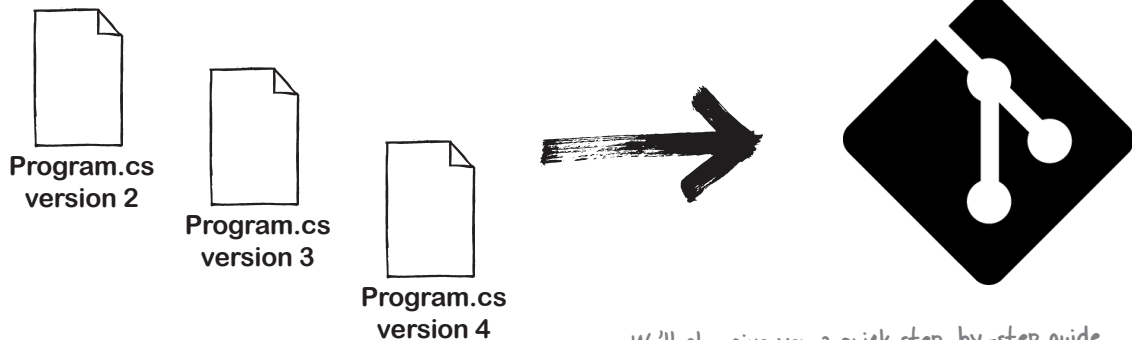
This is the Git logo. You'll see it a lot working with Git.



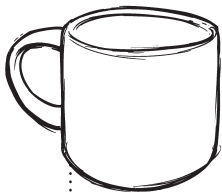
3

You'll make changes to your code changes.

When you make changes to your code, you can push new versions of the changed files to the Git **repo** (which is short for repository). Git will store the full history of each file, so you can see what changed and compare different versions.



We'll also give you a quick step-by-step guide to get started with Unity Version Control, and show you where to learn more about it.



Relax

This is a quick introduction to help you get started with Git.

Git is a really powerful tool that lets you do many things with your code repositories. Here are just some of the things that Git does:

- ★ **Version Tracking:** Keeps a detailed history of file changes.
- ★ **Branching and Merging:** Allows for isolated development and easy integration.
- ★ **Distributed System:** Each user has a complete copy of the repository.
- ★ **Collaboration:** Facilitates seamless collaboration among multiple developers.
- ★ **Staging Area:** Lets you prepare commits by selecting changes.
- ★ **Revert and Reset:** Easily undo mistakes and revert to previous states.
- ★ **Conflict Resolution:** Handles merge conflicts with clear guidance.
- ★ **Blame View:** Shows who last modified each line of a file.
- ★ **Commit History:** Provides a chronological log of all changes.
- ★ **Hooks and Scripts:** Automates tasks with custom scripts at various points in the Git workflow.

We're not going to cover most of those things in this PDF. We're just going to get you up and running so you can publish your code to Git and view it on GitHub.

Start by creating a free GitHub account

The first thing you'll do is **create a GitHub account** if you don't already have one. GitHub is a popular platform for hosting and managing Git repositories. It provides a web-based interface for working with your code repos, and has lots of great features for collaboration and code sharing.

There are *many Git providers*. We're using GitHub in this tutorial because Visual Studio has Git features that seamlessly integrate with it (probably because GitHub is owned by Microsoft). You can create your own Git repositories, or use many different Git providers, and still use the instructions in this guide.

This integration simplifies the process of committing, pushing, and pulling changes directly from your development environment. To get started, create a free GitHub account, which will allow you to store your code online, collaborate with others, and access a vast ecosystem of tools and resources.

A free GitHub account gives you unlimited public and private repositories, collaboration tools, and access to a community of developers. It has lots of features for managing and sharing your code.

Go to <https://github.com> and create a new account for yourself. We recommend that you turn on two-factor authentication to protect your account.

Sign in to GitHub

Username or email address

Password [Forgot password?](#)


[Sign in](#)


[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

Save your Unity projects to the Unity Cloud with Unity Version Control

Unity has its own version control system called **Unity Version Control** that's built right into the Unity editor. With a free Unity Version Control account, you get 5 GB of storage and support for up to three users.

When you have a project open in the Unity editor, click the Unity Version Control button  at the top of the window.

You'll see a Unity Version Control panel  at the bottom of the editor. Click the button to login or sign up. You'll be prompted to create an account if you don't have one. Then you'll be asked to join an organization—choose the default organization. Then you'll create a workspace:

Create a new workspace

Workspaces are used to store the local copies of your project files that you can check in to a repository.

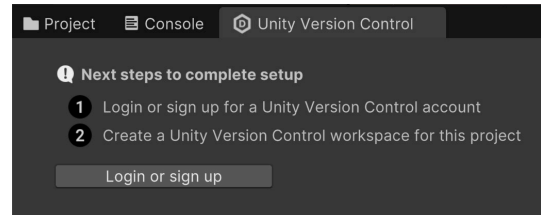
Repository Name

Workspace name

How do you prefer to work?

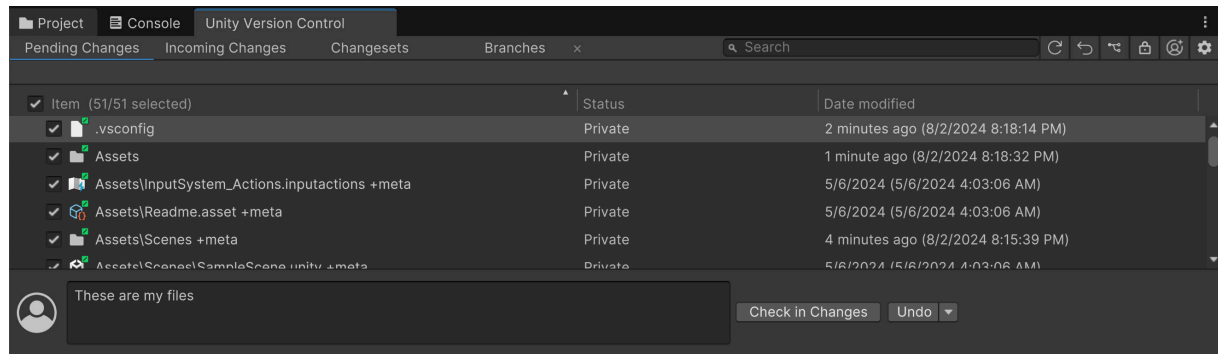
- I need branching, merging, and the ability to push/pull (Unity VCS workspace)
- I only need to check in my work (Gluon workspace)

Learn more about the differences between Unity VCS/Gluon workspaces <color="#0078DA">here.



Log into <https://cloud.unity.com/> to see all of your projects in Unity Cloud.

Any time you want to push your changes to the Unity Version Control cloud, open Unity Version Control, check the box to select all of the files, enter a message, and click the **Check in Changes** button.



Learn more about Unity Version Control here: <https://unity.com/how-to/redeem/version-control>



This guide covers Visual Studio, not Visual Studio Code.

But don't worry if you're using Visual Studio Code—it also has great out-of-the-box support for Git, and there's a great resource for you to use Git with VS Code.

*Have a look at the **Using Git source control in VS Code** page:*

<https://code.visualstudio.com/docs/sourcecontrol/overview> ←

This page does an amazing job showing you how to use Git with VS Code.

That page features a series of videos that show you how to use Git with VS Code and covers everything you need to get up and running with Git and VS Code.

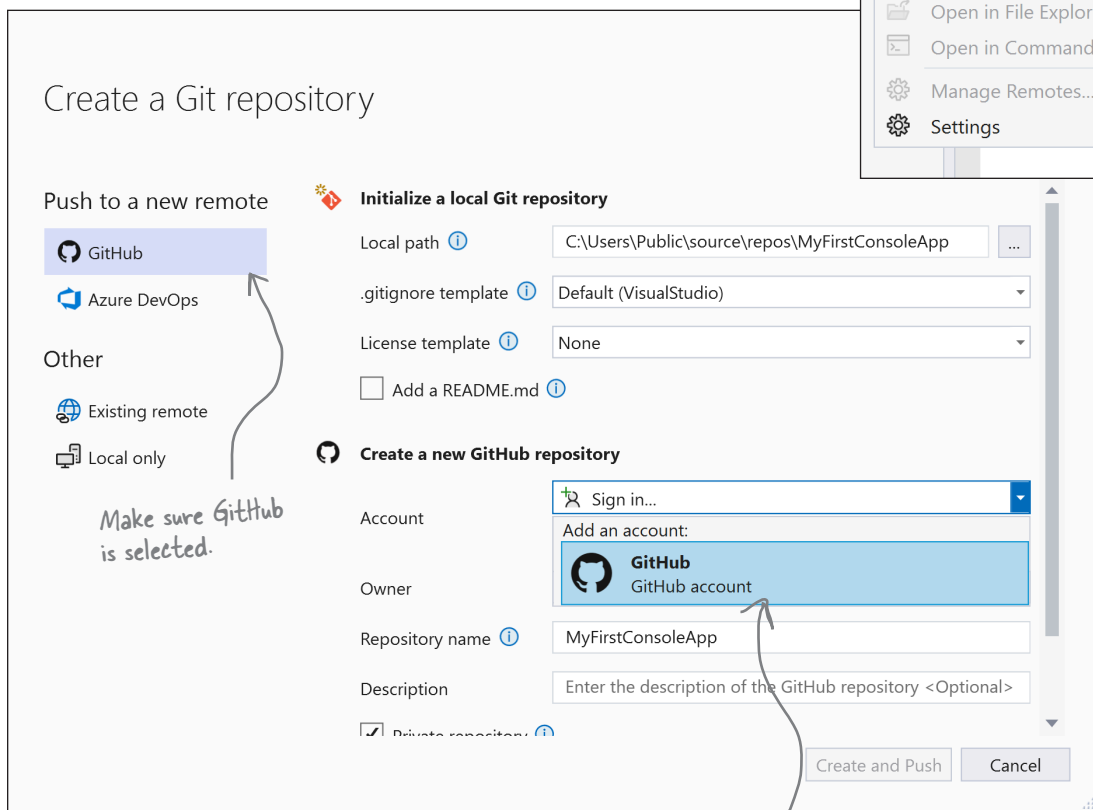
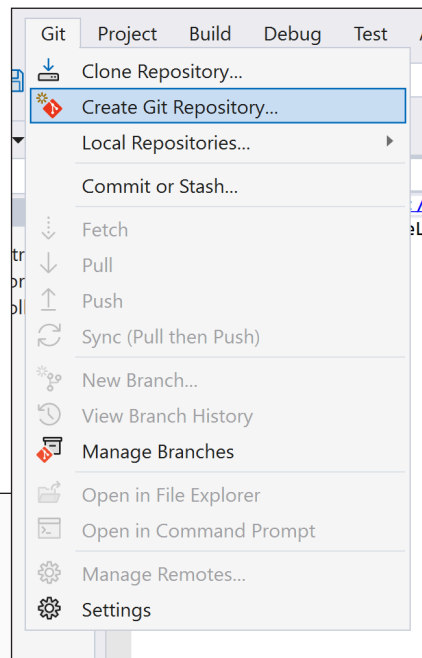
Create a repository for your project

You can create a new Git repository when you have a project open in Visual Studio. Select *Create Git Repository...* from the Git menu. This will cause Visual Studio to open the *Create a Git repository* dialog.

On the left side of the dialog is a menu listing remote Git sources that you can push to. Make sure GitHub is selected.

Initialize your Git repository

Find the Account section in the *Create a Git repository* dialog and sign into your account. Since you've selected GitHub, it will display an option for you to add your GitHub account. You'll only need to log in once, after that it will remember your Git credentials.



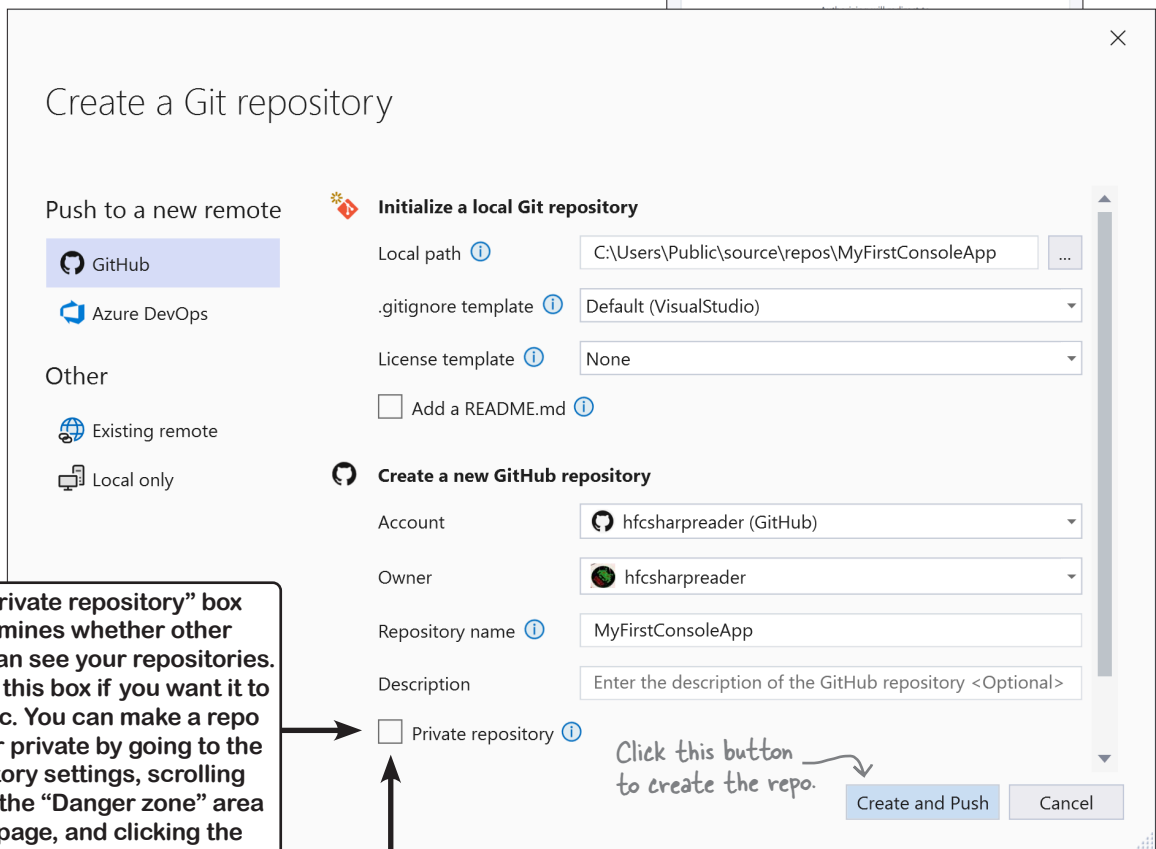
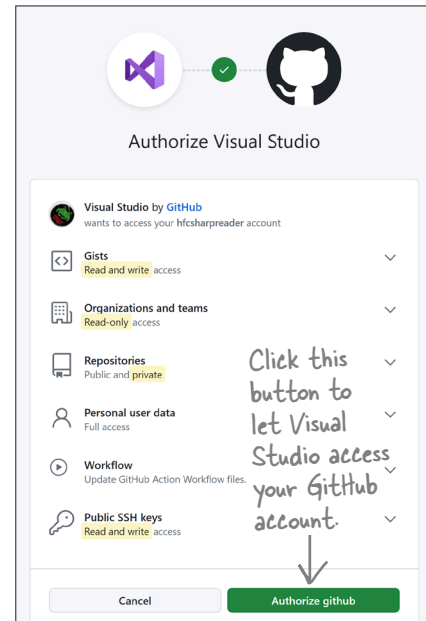
When you sign into your GitHub account, Visual Studio will pop up a browser window that brings you to the normal GitHub login page. If you turned on two-factor authentication (which you should!), you'll need to use it to log in.

Authorize Visual Studio

Visual Studio will connect directly to GitHub to send and receive files, so once you're logged in you'll be prompted to authorize Visual Studio to work with your GitHub account. Click the button to authorize it—once you do, you'll be able to close the window.

Create the repository

Once you're done logging in and authorizing Visual Studio, you'll be returned to the *Create a Git repository* dialog. You can accept all of the default settings to create a private repository with a name that matches your project name. Click the **Create and Push** button to create the new Git repository.

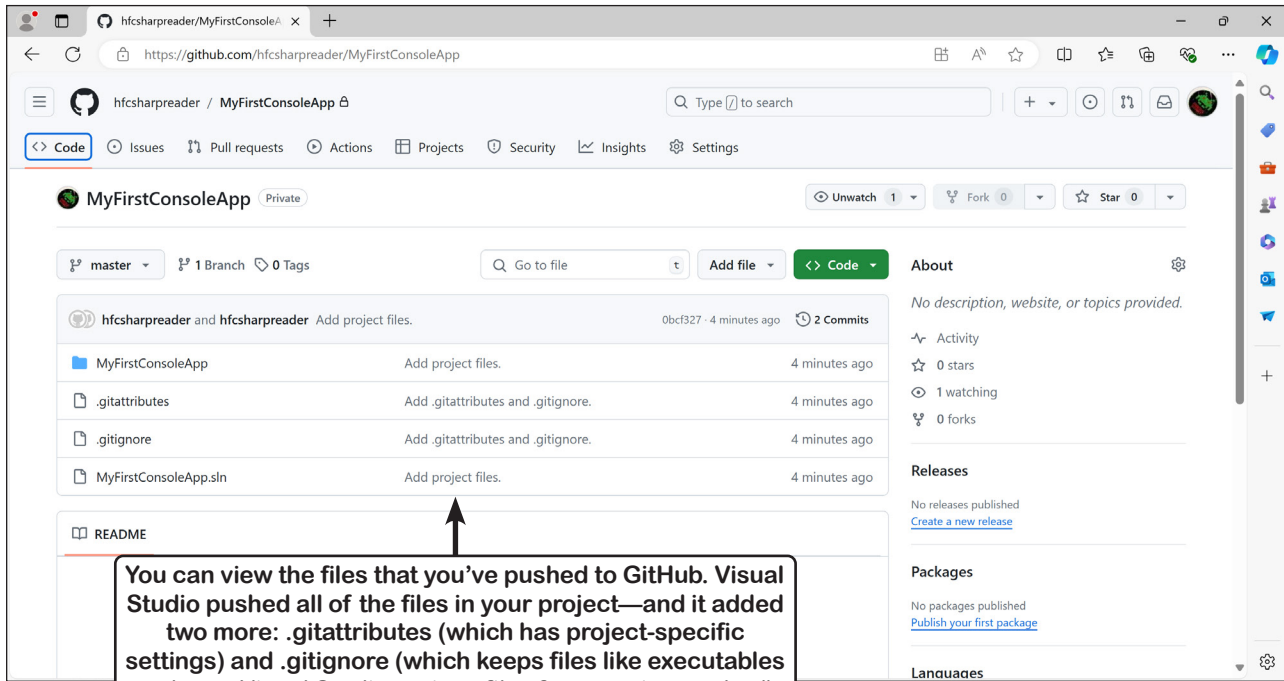
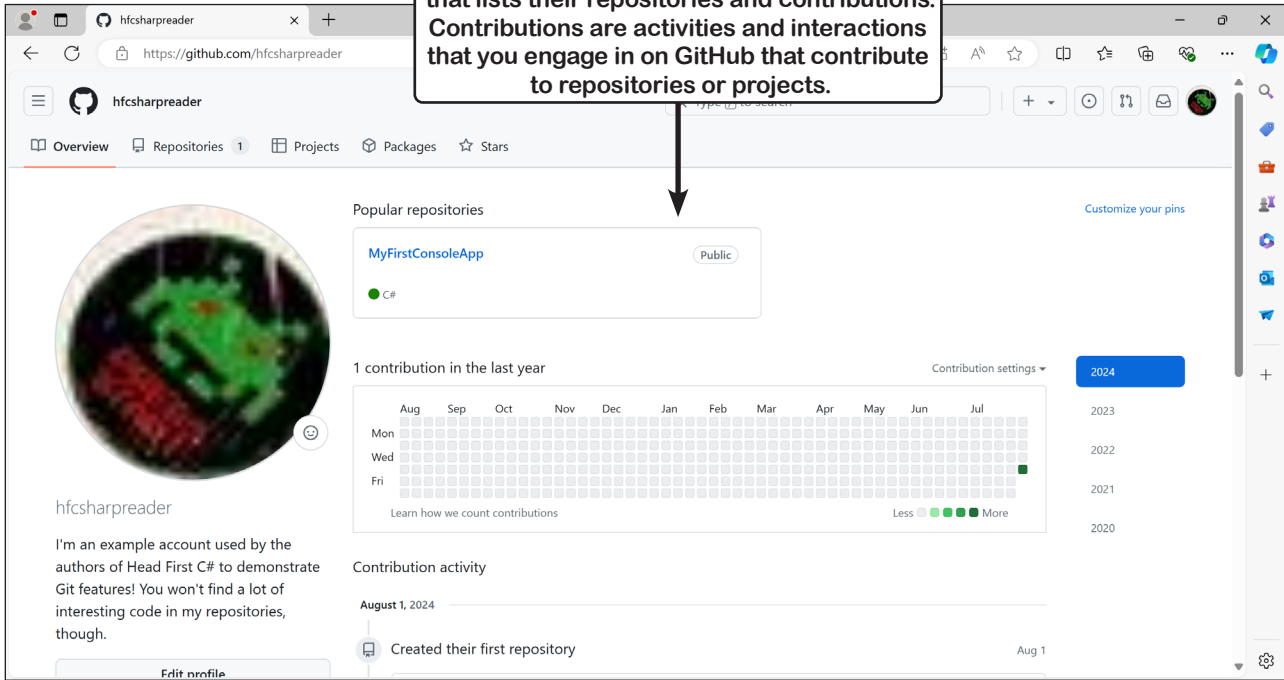


The "Private repository" box determines whether other people can see your repositories. Uncheck this box if you want it to be public. You can make a repo public or private by going to the repository settings, scrolling down to the "Danger zone" area of the page, and clicking the "Change visibility" button.

Want to show off your code to your friends? **Uncheck** the "Private repository" checkbox to make your repo public.

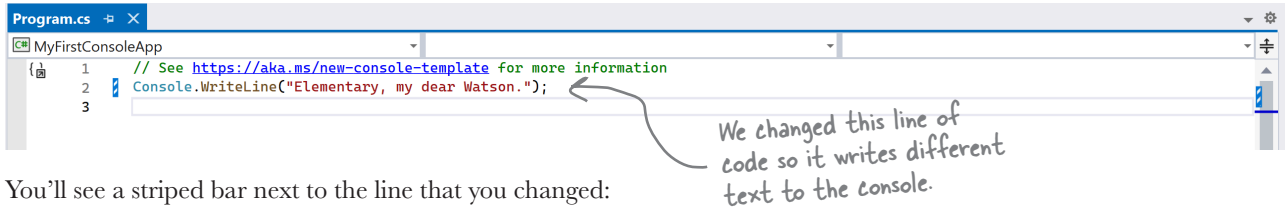
view the files in your repo

Every GitHub account has a profile page that lists their repositories and contributions. Contributions are activities and interactions that you engage in on GitHub that contribute to repositories or projects.

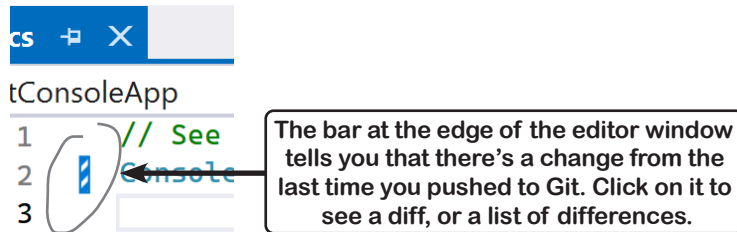


Use diff to see your modifications

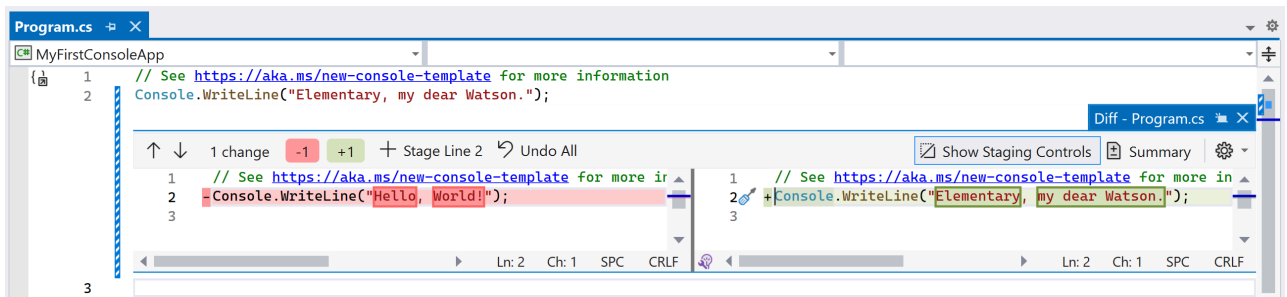
Once your code is pushed to Git, Visual Studio can track any changes since the most recent push. Make a change to the code:



You'll see a striped bar next to the line that you changed:



Click the bar to see a **diff**, or a list of differences between the code in your editor and the most recent code pushed to Git:



Visual Studio is displaying a side-by-side diff, which shows the old version in Git on the left and the current version in the editor on the right. The deleted text is highlighted in red, and the inserted or replaced text is highlighted in green.

A diff is a comparison that shows the differences between two versions of a file or set of files, highlighting the changes made such as additions, deletions, or modifications.

make changes and push them to your repo

Push the changes back to the Git repository

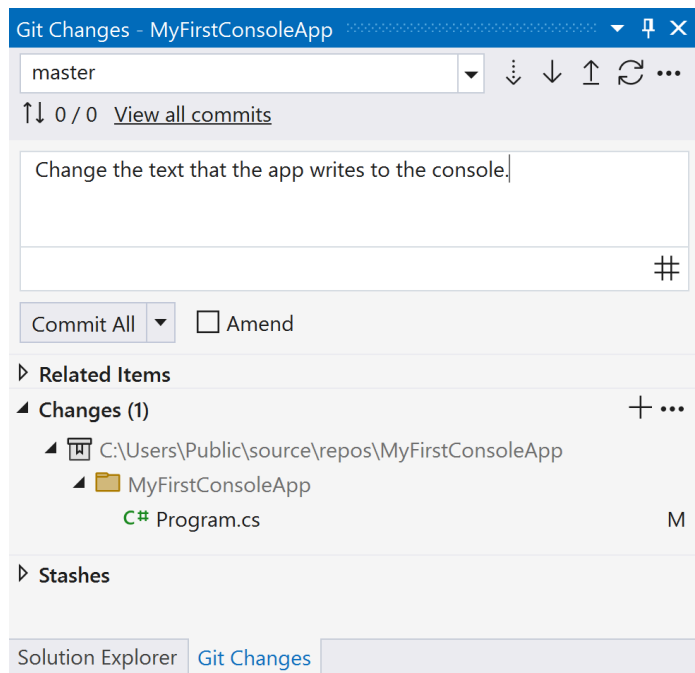
You've made a code change—in our case, we changed a line of code to write “Elementary, my dear Watson.” to the console. Now it's time to push the code back to the Git repo.

The first thing you need to do is create a new **commit**. A Git commit is a snapshot of changes in the repository. It has a record of the project **files** that changed, along with a **message** that you write to describe the changes.

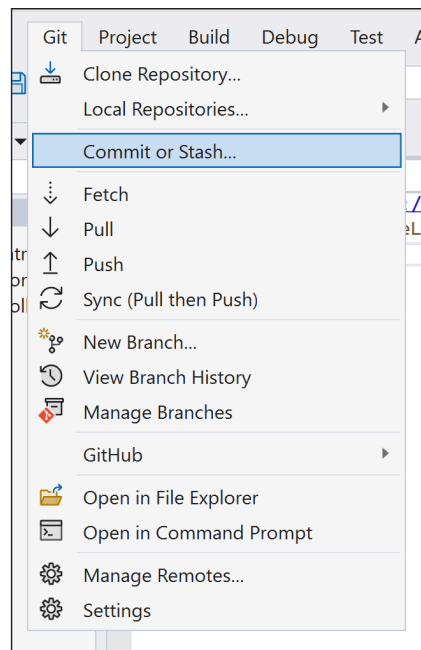
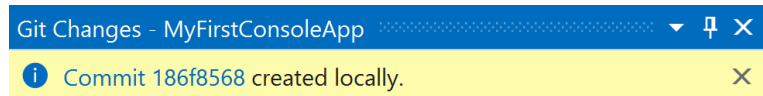
Create a new commit

Choose *Commit or Stash...* from the Git menu to open the Git Changes window, usually displayed in the same panel as the Solution Explorer.

There's a box near the top of the window for you to write a message. We added the message *Change the text that the app writes to the console*:

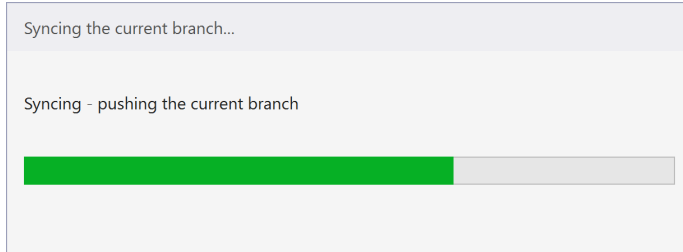


The Changes section of the window shows you all the files that changed. You can double-click on each file in the commit to view a diff. When you're satisfied with the message and changes, **click the Commit All button** to create the new commit. You'll see a message with the **commit hash**, or a unique identifier (in our screenshot it's 186f8568) created specifically for that commit.

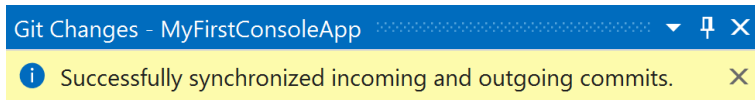


Sync your local repo with the remote

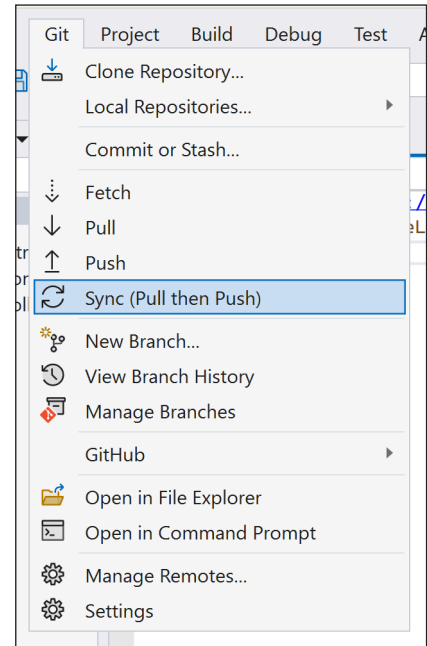
Choose *Sync (Pull then Push)* from the Git menu or click the  sync button in the Git Changes window. You'll see a dialog pop-up with a progress bar:



When it finishes, you'll see a message at the top of the Git Changes window:



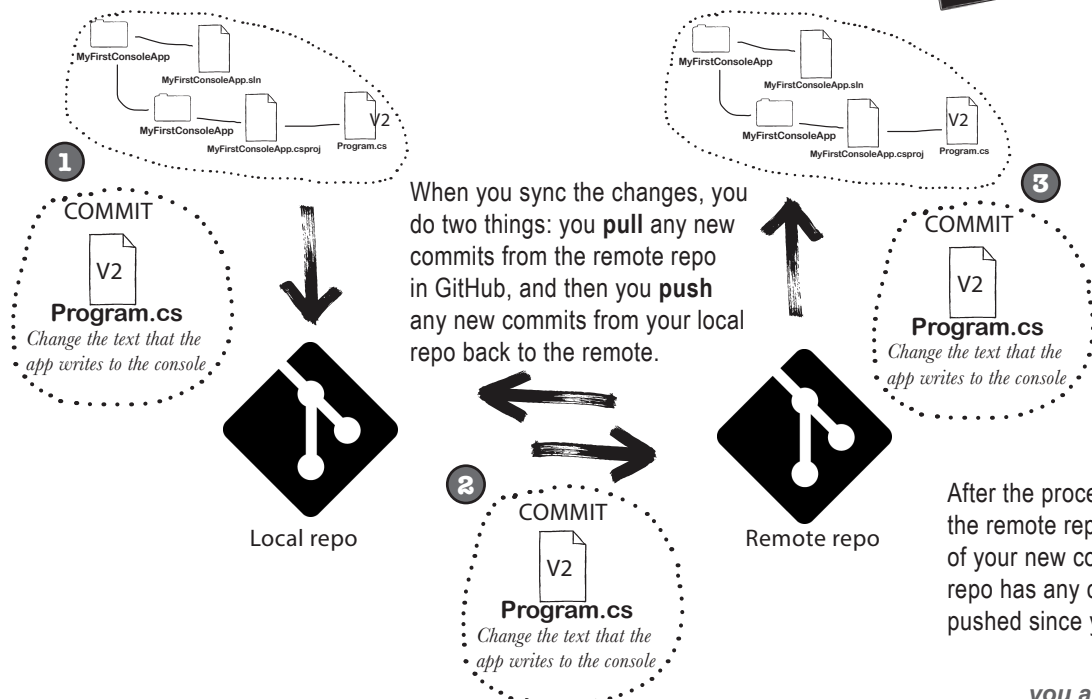
Congratulations, your changes are now synced to GitHub.



Git is a **distributed version control system**, which means that every user has a complete copy of the entire code history on their own computer, so they can work independently without relying on a central server. When you create a commit, you're actually updating a repository that lives on your computer.



Behind the Scenes




After the process is complete, the remote repo in GitHub has all of your new commits, and your repo has any commits that were pushed since you last synced.

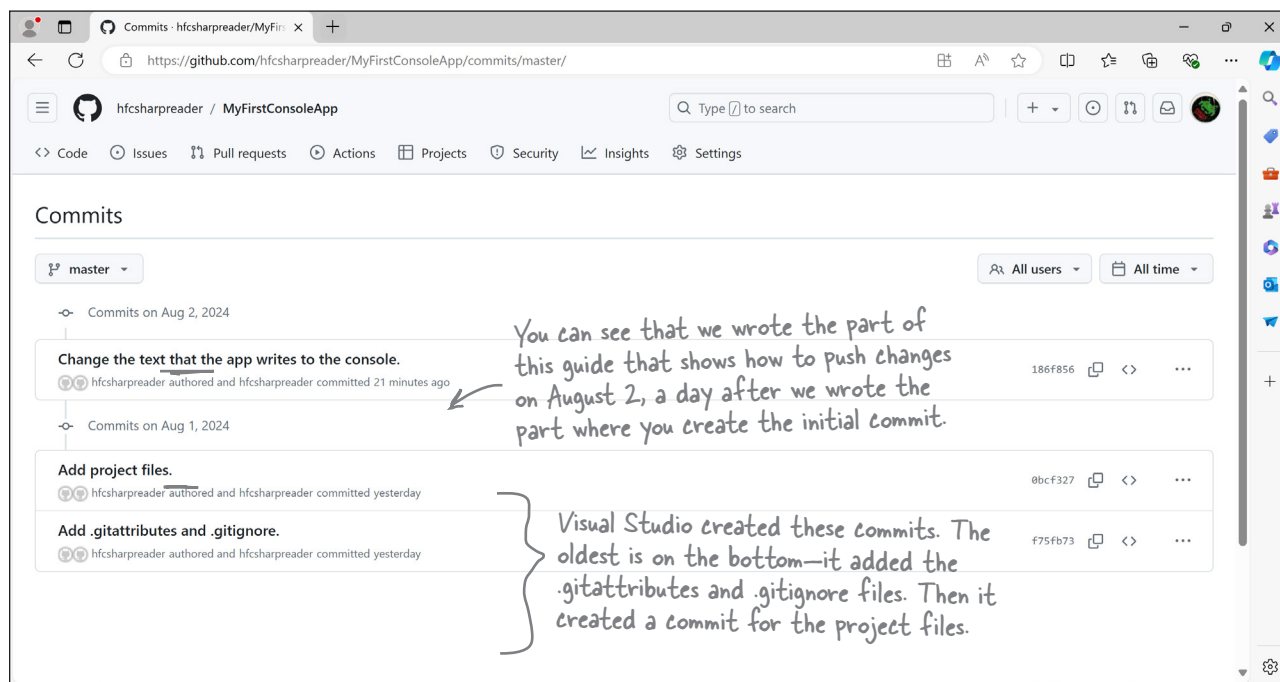
View the changes and commits in GitHub

You can go to any user's public repository by going to a URL like this:

$\text{https://github.com/}$
 github username Repository name
 $\text{https://github.com/hfcsharpreader/MyFirstConsoleApp}$

If you're logged in, you can use that same URL to view your private repo. If you try to view another user's private repo, it will give you a "404 Page Not Found" error—the same error you'll get if you give it an invalid URL (which helps keep people from figuring out the names of private repos).

When you go to the GitHub page for a repo, you'll see the list of files in the repo—that's the screenshot we showed you earlier. You can click into folders and view files in the browser. You can also view the individual commits. Find the link with the number of commits  in the corner of the page and click it to see all of the commits in the repo:



Click the commit message to see a diff of the files in the commit:

Showing 1 changed file with 1 addition and 1 deletion.

Whitespace Ignore whitespace Split Unified

```
MyFirstConsoleApp/Program.cs
@@ -1,2 +1,2 @@
1 // See https://aka.ms/new-console-template for more information
2 - Console.WriteLine("Hello, World!");
2 + Console.WriteLine("Elementary, my dear Watson.");
```


Some important Git concepts

That’s everything you need to know to get started with Git in Visual Studio. Any time you want to push your changes to Git, you can create a new commit and sync with the remote repo. Now you’ll never lose your code, because you’ll always have a backup in your Git repo.

But there is a lot more to Git—this guide just scratches the surface of everything you can do with Git. Here are a few important concepts, which make a great starting point for learning more:

- ★ A **branch** is like a parallel version of your project where you can work on new features or fixes without making changes to the main code. It’s like a safe sandbox where you can do work, make changes, and experiment.
- ★ When you’re happy with your work on a branch, you can **merge** it back into the default or main branch. Merging in Git means changes from one branch and combining them into another. It’s how you bring together different lines of development, like adding a new feature you’ve been working on into the main project.
- ★ A **merge conflict** happens when Git can’t automatically combine changes from different branches because the same parts of the code were changed in different ways. If there are any conflicts between changes, you’ll need to resolve them before everything gets blended smoothly.
- ★ If you have a public repository, other people can create a local copy of your repository and make changes to the code—but they can’t just merge it in. Instead, they create a **pull request**. That’s a way for them to ask you to review and consider adding their changes to your main project. You can review the changes, discuss them, and decide whether to merge them or suggest further modifications.
- ★ A **README.md file** is a text file that provides an overview and important details about a project, usually placed in the main folder of the repository, and displayed on the main web page for the repo. It often includes project descriptions, instructions, and links to more information, making it easy to understand what the project is about and how to get started.

You can change the name of the default branch

Some people don’t like using “master” for the default Git branch because it reminds them of slavery and power dynamics, and some people feel that the term has very negative connotations. That’s why lots of developers and companies are switching to more neutral names like “develop” or “main” for the default branch. You can change the name of the default branch in the repository settings.

Even if you’re okay with “master,” it’s good to know that some folks aren’t.



Default branch

The default branch is considered the “base” branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.



Bullet Points

- **Git** is a version control system. You can use it to avoid losing track of your code and easily recover from mistakes.
- Visual Studio and Visual Studio Code offer **built-in Git support**, simplifying the process of managing and sharing code while handling complex aspects of version control.
- To start using Git, you'll use Visual Studio's built-in Git features to **create a new repository** that stores your project's code and keeps a detailed history of changes.
- Create a **public repository** to share your code with others, or create a **private repository** that others cannot see.
- **Git saves your files** and allows access from anywhere, anytime. It provides snapshots of your work, letting you revisit and compare changes.
- After pushing your code to Git, Visual Studio allows you to track changes and view a **diff**, showing the differences between the current code and the last pushed version, with deleted text highlighted in red and new or modified text in green.
- After you make changes to your code, you can *push those changes to the Git repository* by creating a new commit, which captures a snapshot of the modifications along with a descriptive message.
- In Visual Studio, you can use the **Git Changes window** to write a commit message, review the files that have changed, and then click the Commit All button to finalize and record the changes, which will be stored in your local Git repository.
- To sync your local repository with the remote one, use the **Sync (Pull then Push) option** in the Git menu, which pulls any new commits from the remote repo on GitHub and pushes your local commits, ensuring both repositories are up to date.
- To **view changes and commits in GitHub**, navigate to the repository's URL, where you can browse the list of files, explore commit history, and click on specific commit messages to see a diff of the changes made.
- There's **a lot more to learn about Git** beyond the basics. It offers a wide range of powerful features and concepts that can greatly enhance your workflow and collaboration capabilities. But this guide gives you everything you need to get up and running with Git.



AI chatbots are a great way to learn more about Git

This guide will get you up and running with Git in Visual Studio. Want to learn more? Open your favorite AI chatbot and ask about some of the more advanced concepts that we talked about.

Here's a prompt to get you started—and be sure to ask lots of follow-up questions:

Can you tell me about some important Git concepts, like branching, merging, and pull requests?

Lots of people use Git from the command line. We didn't talk about it at all. Try giving the AI this prompt:

How do I use Git from the command line?

This is a great starting point to take your Git knowledge to the next level.