GitHub Essentials

Exercise Guide

Learn. To Change.

Contents

1. Course Introduction

There are no exercises for this section.

2. Introduction to Git

2.1. Quick Lab – Git User Configuration

Overview

Install Git Bash and use the command line to configure Git with your username and email.

- Check the settings to ensure these are set.

Steps

Getting Started with Git	
1	Download and install Git Bash using the following link. https://git-scm.com/downloads Click the **Download** button to get the appropriate version for your operating system (Windows, macOS, or Linux).
2	Run the Installer: • Open the downloaded file and follow the installation wizard. • During installation, ensure the option to install Git Bash is selected.
3	Verify the Installation: • Open **Git Bash** from your Start Menu (Windows) or Applications (macOS/Linux). • Run the following command to check the version: `git --version` • If the version displays, Git Bash is installed successfully.
Set Up Git with Your User Information	

4	configure your Git username and email by running the following 2 commands:
	`git config --global user.name "Your Name"`
	`git config --global user.email "your.email@example.com"`

2.2. Quick Lab – Git Help

Overview

Use your git bash command line to access git help to find out about some Git commands.

Steps

Using Git Help	
1	display the Git glossary by typing the following into Git Bash: `git help glossary`
2	List all Git commands: `git help –a`
3	Get information about Git configuration: Git help config
4	Search git documentation for common concepts: `git help -g`

2.3. Quick Lab – Using Git Bash to create a first repository

Overview

Use your git bash command line to access git help to find out about some Git commands.

Steps

Create Your First Repository

1 Open Git Bash and create and navigate to the folder where you want to create the repository using the cd command:

```
cd path/to/your/folder
```

For example:

```
cd C:/Labs/GitTest
```

2 Initialize a new Git repository by running the following command:

```
git init
```

This will create a hidden .git folder that tracks your project.

3 Add files to your repository by either creating new ones or copying existing ones using any of the following options:

- Use touch:

```
touch myFile.txt
```

- Use redirection to create a file (called myFile.txt) and add the text "Some Content" to it:

```
echo "Some Content" > myFile.txt
```

- Or use a test editor such as Notepad.exe:

```
notepad myFile.txt
```

- To copy a file from one location to another, use the git cp command:

```
git cp source_filepath destination_filepath
```

4 Add all the files to the repository's staging area:

```
git add -A
```

5 Commit the changes:

```
git commit -m "Initial commit"
```

6	Verify the repositor's status: `git status`
7	View the commit history: `git log`

3. Branching and Merging

3.1. Quick Lab – Create and Work on a Branch

Overview

Create a new branch, add a new file to it, commit the changes, return to the master, ensure new file is **not** in the folder and check the repository's history.

Steps

Create a new Branch	
1	Return to the repository you worked on in the previous lab

2	Create and checkout a new branch called new-feature:
	`git branch new-feature`
	`git checkout new-feature`
	Or combine the two commands into one using the -b option:
	`git checkout -b new-feature`

3	Make some changes to the new branch by adding a new file:
	`touch myNewFeaturesFile.txt`
	`echo "Some New Features Content" > myNewFeaturesFile.txt`
	Feel free to add and edit some additional files if you want.

4	Commit the changes:
	`git add .`
	`git commit -m "Added a file on new branch"`

5	Check the content of the repository's folder and ensure the new features file **is** present. `ls`
Checkout the main branch	

6	Return to the main branch: `git checkout master` Note: the main branch may be called `main`
7	Check the content of the repository's folder and ensure the new features file is **not** present. `ls`
8	Check the log history: `git log`

3.2. Quick Lab – Merge Branches

Overview

Merge the new-feature branch back into main.

Steps

Switch to the target branch	
1	Return to the repository you worked on in the previous lab
2	Switch to the branch you want to merge into (e.g., main). `git checkout main` Note: the main branch may be called `main`
Merge the Source Branch into the Target Branch	
3	Use the git merge command to merge changes from the branch you were working on (new-feature) into the current branch. `git merge new-feature` This will merge the changes from new-feature into main. If there are no conflicts, Git will perform the merge automatically. If there were to be any conflicts, Git would prompt you to resolve them manually before completing the merge (see more about how to do this in session 08).
4	Check the contents of the folder and note it contains the myNewFeaturesFile.txt file: `ls`

3.3. Quick Lab – Deleting Branches

Overview

Delete the new-feature branch.

Steps

Deleting a Branch	
1	Use the branch command to see all current branches: `git branch`
2	Make sure the current branch is main: `git checkout main` Note: the main branch may be called `main`
2	Once you've successfully merged a branch, you can delete it if you no longer need it. `git branch -d new-feature`
3	Try to delete the main branch (the attempt should fail): `git branch -d main`
4	Check the log history: `git log`

4. Introduction to GitHub

4.1. Quick Lab – Setup a GitHub Account

Overview

Follow these simple steps to create and configure your GitHub account.

Steps

Sign Up to GitHub	
1	Open a web browser and browse to https://github.com
2	Click the "Sign up" button in the top-right corner of the GitHub homepage.
3	Fill in the required information: • **Username**: Choose a unique username that represents you. • **Email Address**: Provide a valid email address (used for account recovery and notifications). It's best to use the same address as the one you used to configure Git. However, GitHub allows you to add as many email addresses to your account as you like. If your Git email address is different, you will need to add it to your GitHub account settings in order to connect your commits to your account. For more information see Adding an email address to your GitHub account - GitHub Docs. • **Password**: Create a strong password.
4	Click Create Account
5	Verify your email address by checking your email inbox for a verification email from GitHub. When it arrives, open the email and click the verification link to confirm your email address.
6	Choose a plan (GitHub offers both **free** and **paid** plans). The free plan is more than adequate for you to fully get to grips with GitHub and complete the course.

7	Optionally complete the onboarding process by specifying why you are using GitHub (personal, professional, educational…). There is no need to enable any of the recommended features.

5. Repository Creation and Setup

5.1. Quick Lab – Create a GitHub Repository

Overview

Follow these simple steps to create and configure a new GitHub repository.

Steps

To Create a GitHub Repository	
1	Open a web browser and browse to https://github.com and sign in
2	Navigate to the "New Repository" page by clicking the "+" icon in the top-right menu and select "New repository". Or Click on your profile picture in the top-right corner of the page. From the dropdown menu, select "Your repositories". Click the green "New" button on the right side of the repositories page.
3	Complete the repository details by specifying: **Repository Name**: myFirstGitHubRepository **Description** (Optional): Leave this empty **Visibility**: Public **Initialize with a README**: Check this box **Add .gitignore**: Select a language template (if in doubt select Python)
4	Create the repository by clicking the "Create Repository" button. GitHub will create the repository and redirect you to its main page.

5.2. Quick Lab – Clone the Repository

Overview

Follow these simple steps to clone the GitHub repository.

Steps

Configure Git to use the GitHub repository and clone its content	
5	In GitHub click on the green **Code** dropdown button on the repository page and copy the HTTPS link.
6	Create a new Git repository on your local machine by running Git Bash and changing to a relevant folder using the `cd` (change directory) and `mkdir` (make directory) functions as appropriate. Then run: `git init`
7	Clone the repository to your local machine by entering: `git clone <repository-url>` Where `<repository-url>` is the URL you copied in step 5.
8	Ensure the README and .gitignore files have been downloaded to the folder. `Ls -a` The "a" switch shows all files and folders.

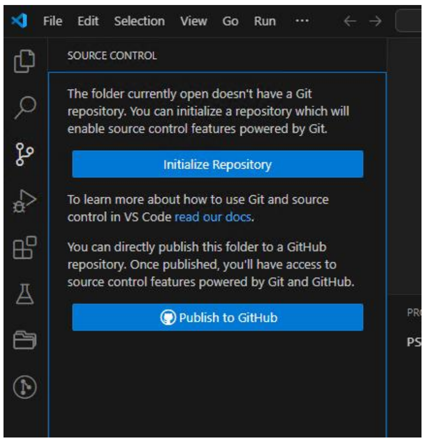
6. Linking to Visual Studio Code

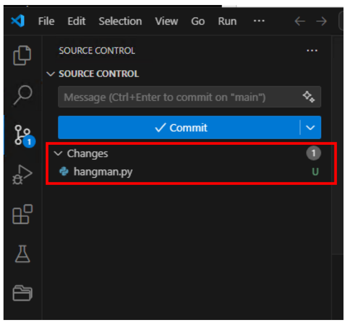
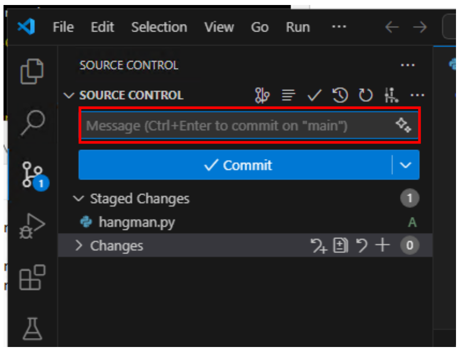
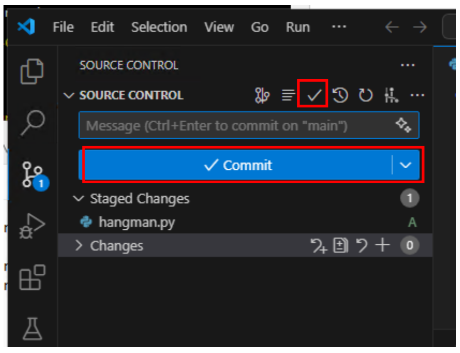
6.1. Quick Lab – Create and Manage Repos from VSC

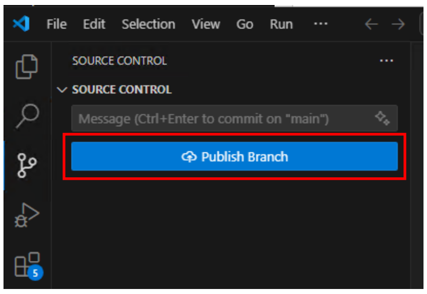
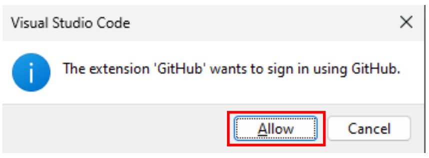
Overview

Create a new Git repository from a local Python project using Visual Studio Code (VS Code) and push it to GitHub as a new repository.

Steps

Initialize a Git Repository in Your Local Python Project	
1	Copy the Hangman.py file located in C:/Labs/Games to a folder of your choice.
2	Launch VS Code and open the folder containing your Python code.:
3	Open the Source Control panel in the Activity Bar on the left-hand side of the screen (it looks like a branch with nodes).
4	If your folder is not yet a Git repository, you can initialize it by clicking "Initialize Repository" so it runs `git init` in your project folder.  SOURCE CONTROL The folder currently open doesn't have a Git repository. You can initialize a repository which will enable source control features powered by Git. Initialize Repository To learn more about how to use Git and source control in VS Code read our docs. You can directly publish this folder to a GitHub repository. Once published, you'll have access to source control features powered by Git and GitHub. Publish to GitHub
5	View changes by looking at the Source Control panel, where you'll see a list of files with changes.

	 File Edit Selection View Go Run ··· SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") ✓ Commit Changes hangman.py U
6	Stage the changes by hovering over the file you want to stage and clicking the + icon that appears next to the file name. Or, right-click the file and select "Stage Changes". To stage all changes at once, click the + icon next to the Changes heading.
7	Commit the changes by firstly writing "First commit" in the Message text box located above the Commit button.  File Edit Selection View Go Run ··· SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") ✓ Commit Staged Changes hangman.py A Changes And then, press the Commit button or click the checkmark icon (a tick symbol) at the top of the Source Control panel.  File Edit Selection View Go Run ··· SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") ✓ Commit Staged Changes hangman.py A Changes
8	Verify the commit by clicking the period ellipses (...) in the Source Control panel and selecting "View Commit History" (Note, this requires an extension like GitLens).

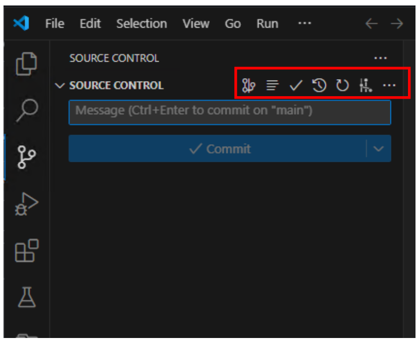
	Alternatively, use VSCode's terminal window (Shown via the View menu's Terminal option) and running: `git log`
Push the repository to GitHub	
9	Make sure there is an appropriate GitHub repository for the code to be pushed to (you could use the one created in step 5 or create a new one).
10	Click the Publish Branch button located in the Source Control panel.  File Edit Selection View Go Run ··· SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") Publish Branch Click the "Allow" button on the message box that appears:  Visual Studio Code The extension 'GitHub' wants to sign in using GitHub. If prompted select a user to authorise VSCode's request and grant any additional permissions.
11	Select the option that will publish to a public GitHub repository. This option should create a new repository on your GitHub account.
12	Navigate to your GitHub home page. Click on your profile picture in the top-right corner of the page and select the "Your Repositories" option. You should see your newly created repository on the top of the list. Click on the repository name and make sure the hangman.py file has been uploaded.

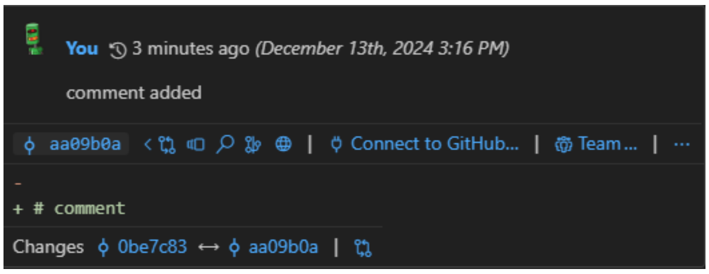
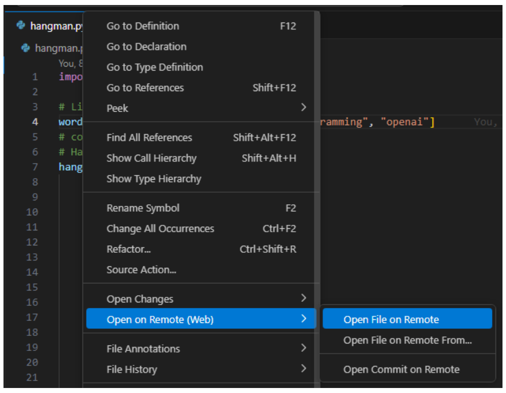
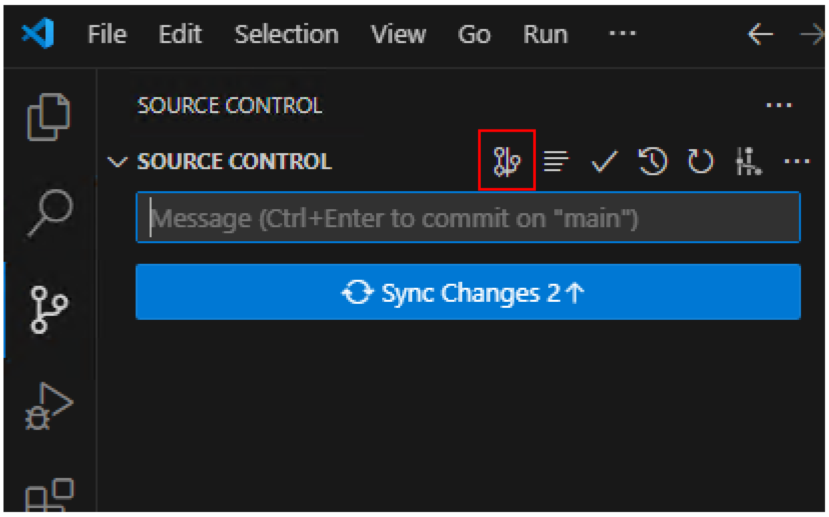
6.2. Quick Lab – Add extensions to VSC

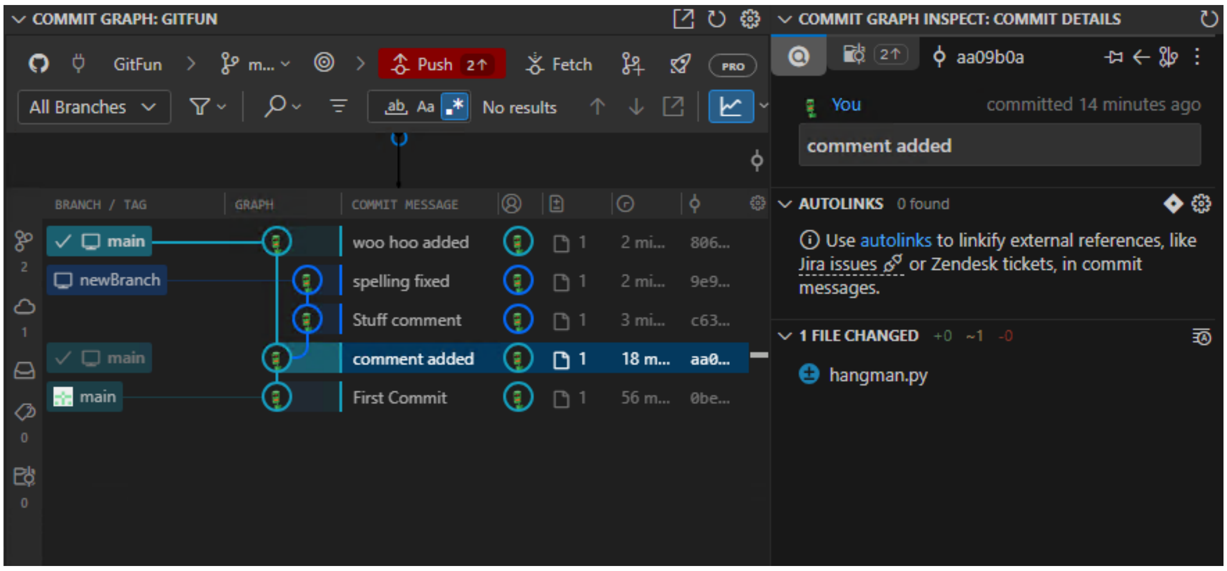
Overview

Add some of the recommended Git extensions to VSCode to enhance the ways you can interact with repositories.

Steps

Add Some Extensions to VSCode	
1	Open the Extensions panel in the Activity Bar on the left-hand side of the screen (it looks like a branch with stacked boxes).
2	Type Git Blame into the search box and click on its install button
3	Repeat step 2 for Git Extension Pack (which will install "Open in GitHub, Bitbucket and GitLab", "GitLens" and "Git History")
4	Repeat step 2 for Git Graph.
5	Return to viewing VSCode's Source Control panel. Notice the additional options that have appeared on its top menu.  File Edit Selection View Go Run ··· SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") ✓ Commit Hover your mouse over the various options to see some text that describes what they do.
6	Add a line of comment to the hangman.py file: `# comment` Stage and commit the change
7	Hover the cursor over the comment and notice a popup window appears that tells you who authored the line and when the change was made.

	 You 3 minutes ago (December 13th, 2024 3:16 PM) comment added + # comment Changes 0be7c83 ⟷ aa09b0a
8	Right click on a piece of code and select "Open on Remote (Web) > Open File on remote. Notice the code for the specified page opens up in GitHub.  hangman.py Go to Definition F12 Go to Declaration Go to Type Definition Go to References Shift+F12 Peek Find All References Shift+Alt+F12 Show Call Hierarchy Shift+Alt+H Show Type Hierarchy Rename Symbol F2 Change All Occurrences Ctrl+F2 Refactor... Ctrl+Shift+R Source Action... Open Changes Open on Remote (Web) File Annotations File History Open File on Remote Open File on Remote From... Open Commit on Remote
9	Create a new branch called "newbranch" and edit the hangman.py code a few times, staging and committing changes as you go. Return to the main branch and do the same things.
10	Click on the Show Commit Graph button on the Source Control Panel's menu.  File Edit Selection View Go Run ... SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "main") Sync Changes 2
11	Look at the Commit Graph that gets displayed (something like the following):



Branch / Tag	Commit message	Changes	Date	SHA
main	woo hoo added	1	2 mi...	806...
newBranch	spelling fixed	1	2 mi...	9e9...
	Stuff comment	1	3 mi...	c63...
main	comment added	1	18 m...	aa0...
main	First Commit	1	56 m...	0be...

aa09b0a — You committed 14 minutes ago

comment added

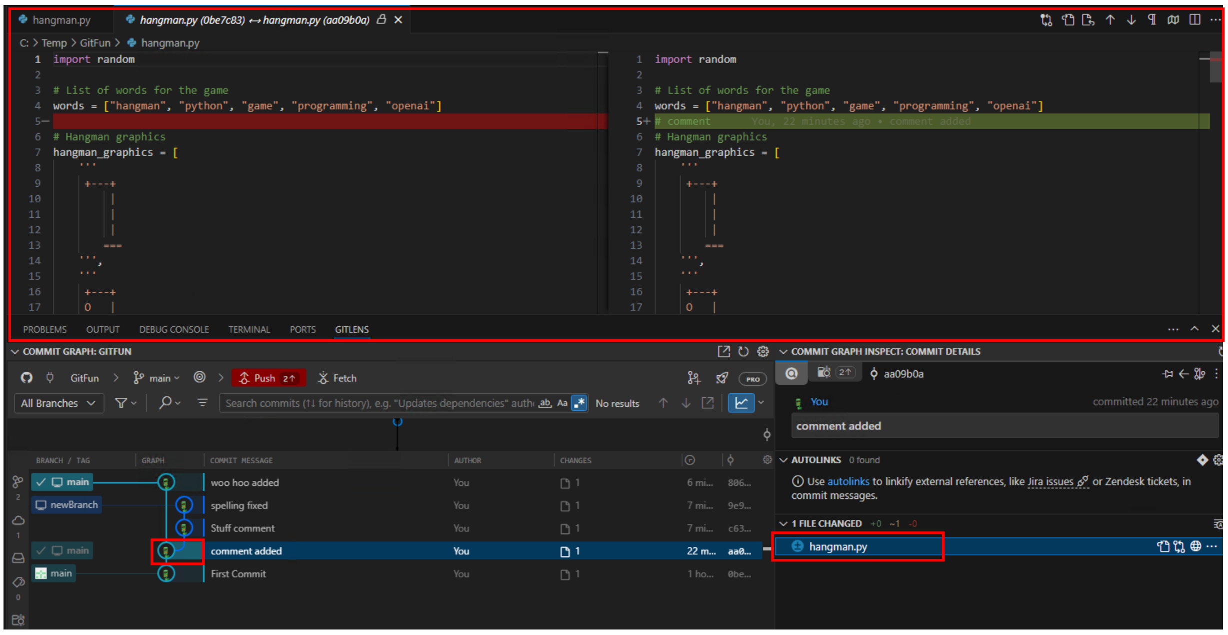
AUTOLINKS 0 found

Use autolinks to linkify external references, like Jira issues or Zendesk tickets, in commit messages.

1 FILE CHANGED +0 ~1 -0

hangman.py

Notice it visually shows the branches and commits with details of the author, dates and times the changes were made. Clicking on a commit reveals information of the change that was made in the right-hand pane. Clicking on this opens up a before and after pair of windows that provide a before and after snapshot of what happened:



hangman.py (0be7c83) ↔ hangman.py (aa09b0a)

```
import random

# List of words for the game
words = ["hangman", "python", "game", "programming", "openai"]
- 
# Hangman graphics
hangman_graphics = [
```

```
import random

# List of words for the game
words = ["hangman", "python", "game", "programming", "openai"]
+ # comment      You, 22 minutes ago • comment added
# Hangman graphics
hangman_graphics = [
```

12	Click on the other buttons on the Source Control panels' menu and see if you can work out what they are showing and doing.
13	**DO <u>NOT</u>** push the changes to GitHub. Wait until the next Quick Lab.

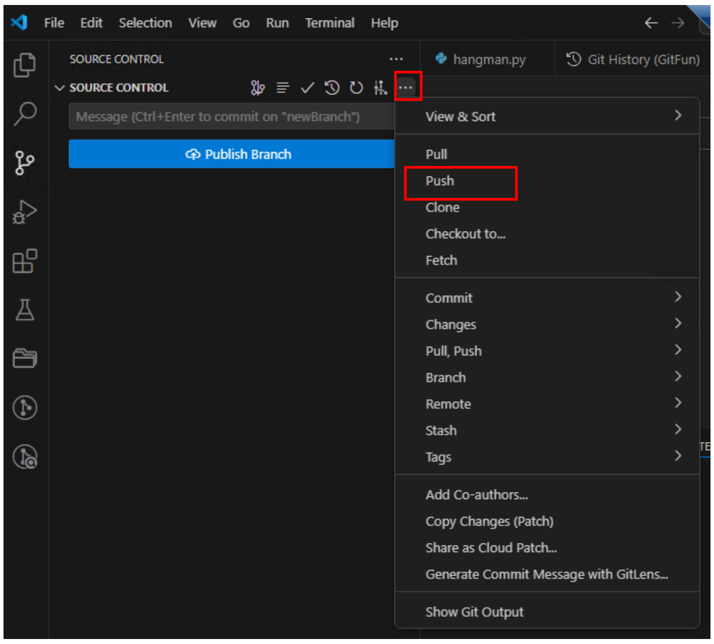
7. Pull Requests

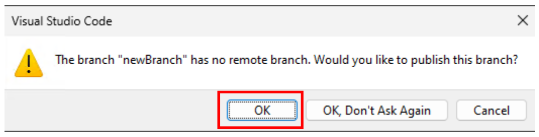
7.1. Quick Lab – Carry Out a Pull Request

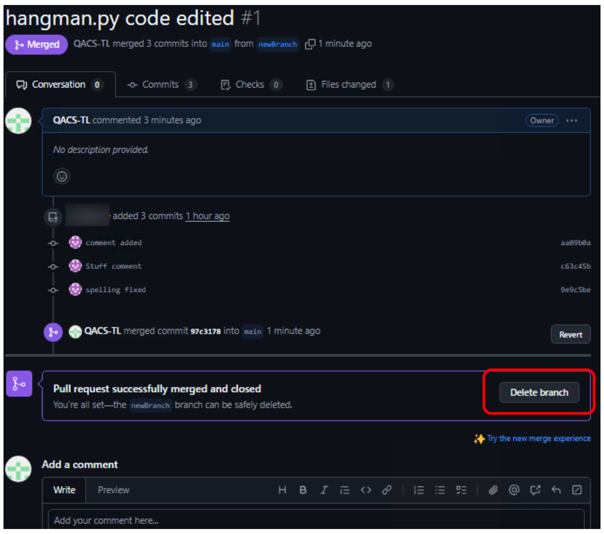
Overview

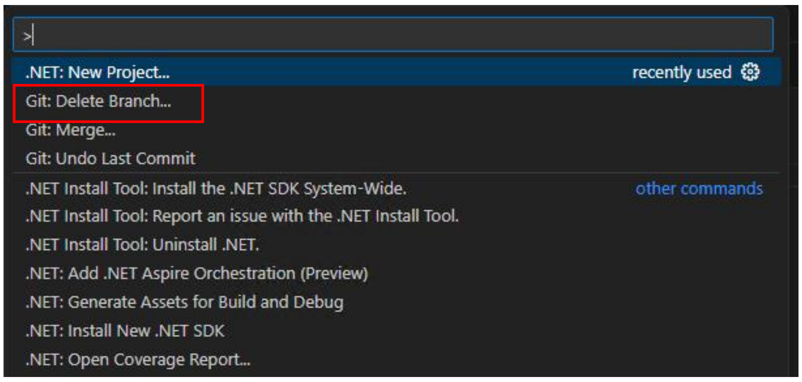
Push a changed and committed branch to GitHub and carry out a "clean" pull request with no merge issues.

Steps

Push the latest version of the Committed changes made to the Hangman.py File	
1	Return the VSCode and the project that contains the Hangman.py file. Make sure there are a number of committed changes which won't cause any merge issues and the changes are ready to be pushed to GitHub.
2	Make sure the current branch is "newbranch" (the new one you created in step 9 of the previous quick lab).
3	Push newbranch to the GitHub repository by clicking the period ellipses (...) in the Source Control panel and selecting Push:  File Edit Selection View Go Run Terminal Help SOURCE CONTROL SOURCE CONTROL Message (Ctrl+Enter to commit on "newBranch") Publish Branch hangman.py Git History (GitFun) View & Sort Pull Push Clone Checkout to... Fetch Commit Changes Pull, Push Branch Remote Stash Tags Add Co-authors... Copy Changes (Patch) Share as Cloud Patch... Generate Commit Message with GitLens... Show Git Output
4	Answer OK to the pop-up message that says 'The branch "newBranch" has no remote branch. Would you like to publish this branch?'

	 Visual Studio Code — The branch "newBranch" has no remote branch. Would you like to publish this branch?
Create a Pull Request on GitHub	
5	Navigate to the GitHub repository page in your browser.
6	Open the Pull Request Page (GitHub will usually show a banner suggesting you create a pull request after pushing a new branch). Click "Compare & pull request". If no banner appears, go to the Pull Requests tab and click "New Pull Request".
7	Fill in the Pull Request Details: Ensure the base branch (Something main) and compare branch (new branch) are correct. Add a title and description explaining the changes (something like "hangman.py code edited")
8	Click the "Create pull request" button.
9	Review and Merge the Pull Request. In the real world, Team members or collaborators can review the code and suggest changes. If changes are requested, implement them locally, commit them, and push them to the same branch. The Push Request (PR) will automatically update with the new commits.
10	Merge the Pull Request by clicking the "Merge pull request" button on the GitHub Pull Request page. Confirm the merge by clicking "Confirm merge".
11	After merging, GitHub will offer an option to delete the branch. Click "Delete branch".

	 hangman.py code edited #1
	Merged QACS-TL merged 3 commits into `main` from `newBranch` 1 minute ago
	Conversation 0 · Commits 3 · Checks 0 · Files changed 1
	QACS-TL commented 3 minutes ago · Owner
	No description provided.
	[redacted] added 3 commits 1 hour ago
	`comment added` `aa09b0a`
	`Stuff comment` `c63c45b`
	`spelling fixed` `9e9c5be`
	QACS-TL merged commit **97c3178** into `main` 1 minute ago · Revert
	Pull request successfully merged and closed
	You're all set—the `newBranch` branch can be safely deleted. · Delete branch
	Try the new merge experience
	Add a comment
	Write · Preview
	Add your comment here...
	
	If not prompted, you can manually delete it under the "Branches" tab.

12	Delete the branch locally by pressing the "F1" key whilst viewing VSCode and selecting the "Git: Delete Branch..." option from the drop down.  >| .NET: New Project... · recently used Git: Delete Branch... Git: Merge... Git: Undo Last Commit .NET Install Tool: Install the .NET SDK System-Wide. · other commands .NET Install Tool: Report an issue with the .NET Install Tool. .NET Install Tool: Uninstall .NET. .NET: Add .NET Aspire Orchestration (Preview) .NET: Generate Assets for Build and Debug .NET: Install New .NET SDK .NET: Open Coverage Report... And then selecting newBranch.
13	Run `git log` to confirm all is well.

8. Managing Merge Conflicts

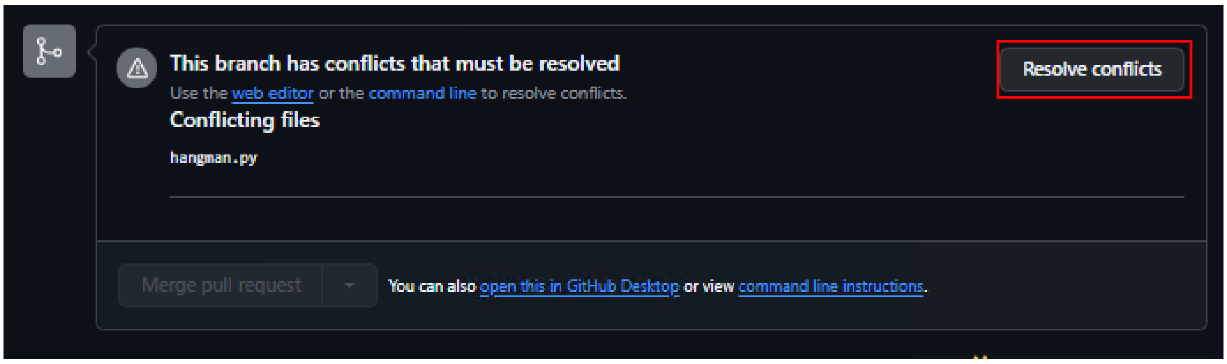
8.1. Quick Lab – Managing Merge Conflicts

Overview

Handling a Pull Request (PR) with conflicts involves identifying the issues, resolving them, and ensuring the codebase is stable.

Steps

Make changes to and push the latest version of the committed versions of the Hangman.py file to GitHub	
1	In the VSCode project create a new branch (called "newbranch") and edit a line in the Hangman.py file and remember the line number. Save, stage and commit the changes.
2	Make sure you are on the main branch in VSCode. Make some changes to the same line of code in Hangman.py file. Save, stage and commit the changes.
3	push the changes made to the main branch to GitHub.
4	push the changes made to the branch called "newbranch" to GitHub. Answer OK to the pop-up message that says 'The branch "newBranch" has no remote branch. Would you like to publish this branch?'
5	Navigate to the GitHub repository page in your browser.
6	Open the Pull Request Page by clicking the "Compare & pull request" button. If no banner appears, go to the Pull Requests tab and click "New Pull Request".
7	Fill in the Pull Request Details: Ensure the base branch (Something main) and compare branch (new branch) are correct. You will probably get a warning that the branches can't be automatically merged but note that you can still create a Pull Request by clicking the "Create Pull request button"

8	Give the request a Title and press the "Create Pull Request" button
9	Click the "Create pull request" button.
10	GitHub will display a message like *"This branch has conflicts that must be resolved"*. Click **Resolve Conflicts** to view the files with conflicts.  This branch has conflicts that must be resolved Use the web editor or the command line to resolve conflicts. Conflicting files hangman.py You can also open this in GitHub Desktop or view command line instructions.

11	Understand the Conflict:
	
	Conflicts occur when changes from the feature branch and the main branch overlap.
	
	Git marks the conflicts in the file with:
	
	`<<<<<<< newbranch`
	`(Code from the newbranch branch)`
	`=======`
	`(Code from the main branch)`
	`>>>>>>> main`
	
	You'll notice the "Mark as resolved" button is disabled.

12	**Note**: We are going to resolve the issue directly in the current GitHub window, but it is often better for control to handle them locally. You could do this by running `git fetch origin` and `git checkout newbranch` commands in VSCode's terminal window. And then running `git merge newbranch` from the main branch and use the IDE's tools to resolve the conflicts (in VSC this is the Resolve Tools Merge Editor).
12	Edit the code in the GitHub window by removing all the conflict markers (`<<<<<<<`, `=======`, `>>>>>>>` and branch names) and changing what is left to reflect the "correct" version. You'll notice the "Mark as resolved" button has now been enabled.

13	When you are happy with your changes click the " Mark as resolved" button. And then the "Commit Merge" button
14	You should see a message that says, "This branch has no conflicts with the base branch" and a green Merge pull request button which you should press. You can finally click the "Confirm Merge" button
15	All should be well, and you are invited to press the Delete branch button which will be a safe thing for you to do.
16	Delete the branch locally by pressing the "F1" key whilst viewing VSCode and selecting the "Git: Delete Branch..." option from the drop down.
17	Run `git log` to confirm all is well.

9. GitHub Challenge

9.1. Quick Lab – Managing Merge Conflicts

Challenge

Your tutor will have paired you up and sent you to a breakout room. You are tasked to collaboratively create some code (using a programming language of your choice). Each one of you needs to create a unique code file and you should also share a file. You don't need to write any valid code (lines of comment, plain English or gibberish will be fine) but both of you should edit your unique file and the shared file. The repository should also include a README file which both of you should edit and a .gitignore file.

You should start by creating a single GitHub repository. You should both clone the repository down to your local machines and create your own branches which you should work with to add your unique files and edit the shared ones.

The principal aim is to ensure the main branch on GitHub is **never** compromised. If a Pull request identifies a conflict, then it is generally better to fetch and review the issues locally and having resolved the conflicts to commit and push them back to GitHub.

QA

Learn. To Change.

QA.com