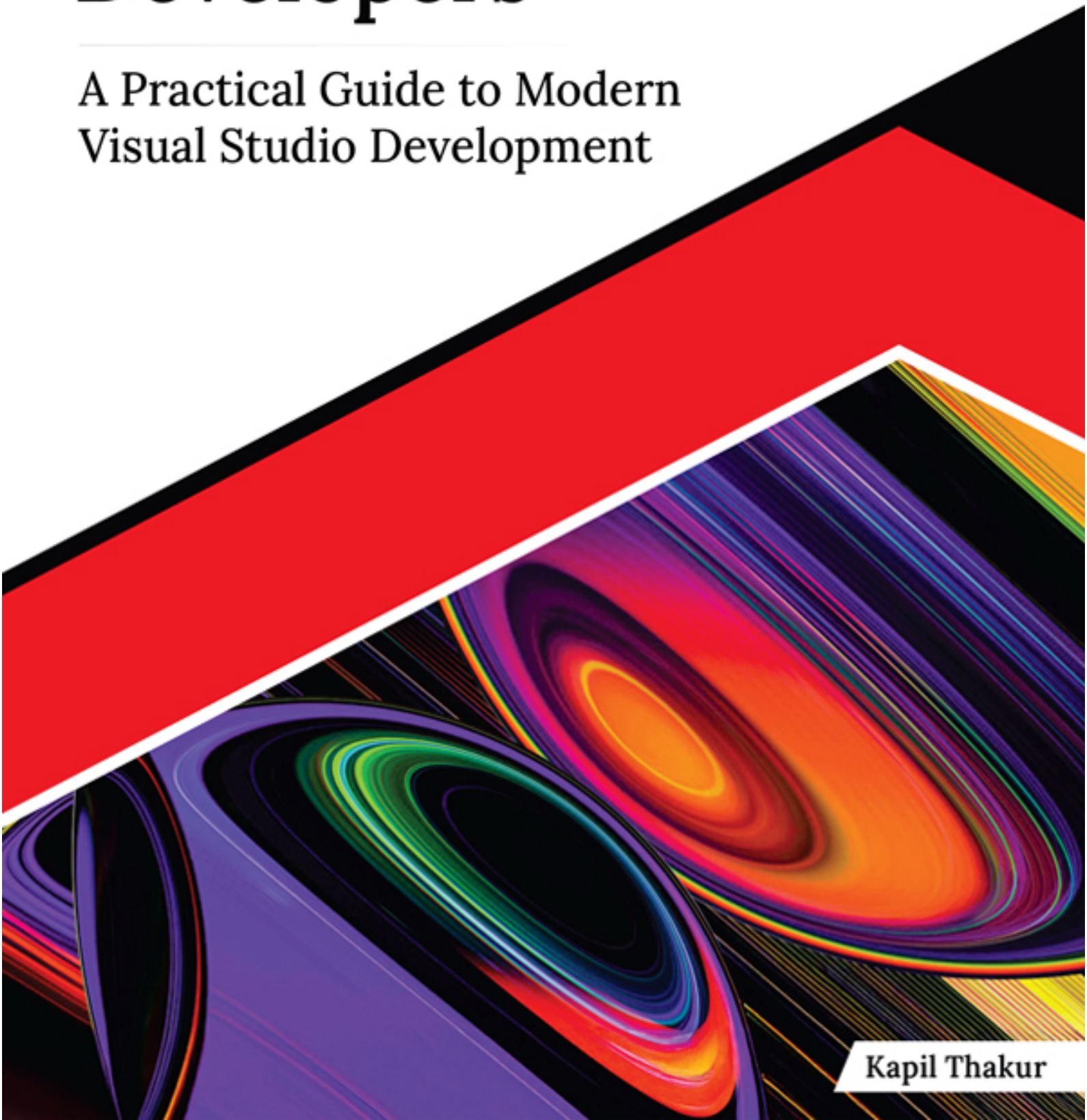
ULTIMATE

Visual Studio for Developers

A Practical Guide to Modern Visual Studio Development



Kapil Thakur

First Published: February 2026

Published by: Orange Education Pvt Ltd, AVA®

Address: 9, Daryaganj, Delhi, 110002, India

275 New North Road Islington Suite 1314 London,
N1 7AA, United Kingdom

ISBN (PBK): 978-93-49887-51-0

ISBN (E-BOOK): 978-93-49887-44-2

Scan the QR code to explore our entire catalogue

www.orangeava.com

Table of Contents

CHAPTER 1

Getting Started with Visual Studio

Introduction

We will begin by exploring the significance of Visual Studio in the realm of software development, while discussing why it stands out as the go-to choice for millions of developers worldwide. From its intuitive interface to its extensive array of functionalities, Visual Studio provides a versatile platform to tackle projects of all sizes and complexities.

As we progress through the chapter, we will navigate through the landscape of Visual Studio versions, and shed light on the different editions available, while helping you choose the one that best suits your needs. Additionally, we will walk you through the step-by-step installation process for a seamless setup experience for both newcomers and experienced users alike.

Once Visual Studio is up and running on your system, we will embark on an exploration of its core components. You will be introduced to the IDE and gain familiarity with its layout, navigation tools, and essential features. From creating and opening projects to customizing your development environment, this chapter lays the groundwork for your journey with Visual Studio.

Furthermore, we will dive into the intricacies of the menu bar and its sub-menu items, which will act as a treasure trove of tools and options at your disposal. Hence, whether you are looking to configure project settings, access debugging tools, or explore advanced features, the menu bar will be the gateway to a wealth of functionalities within Visual Studio.

At the end of this chapter, you will be well-equipped with the knowledge and skills necessary to harness the full potential of Visual Studio 2022. Thus, whether you are aiming to build simple applications or tackle complex software projects, this chapter provides the foundation upon which you can build your expertise, and embark on a rewarding journey in software development with Visual Studio.

Structure

In this chapter we will cover the following topics:

- Introduction to Visual Studio
- Choosing Your Visual Studio Edition
- Visual Studio 2022 System Requirements for Installation
- Visual Studio 2022 Installation Guide
- Customizing Panels
- Working with Documents
- Synchronizing Accounts and Settings
- Customizing the Menu bar and Toolbars
- Understanding the VS Menu bar

Introduction to Visual Studio

Visual Studio is a cornerstone of software development and offers a comprehensive **Integrated Development Environment (IDE)** that covers the entire development cycle in one cohesive platform. It serves as a complete application development tool with features for designing, developing, debugging, deploying and finally maintaining and enhancing any .Net application. It supports several technologies apart from .Net.

Visual Studio is designed to streamline software development across various platforms and languages. Unlike other IDEs, Visual Studio offers incredible versatility and supports many programming languages such as C#, C++, Python, and JavaScript, along with robust tools and features tailored to each language's ecosystem.

With its rich array of tools and features, Visual Studio provides an unparalleled environment for writing, editing, debugging, building, and deploying code across several languages and platforms, which makes it a preferred choice for developers worldwide. It is the most comprehensive IDE for .NET and C++ developers on Windows for building web, cloud, desktop, mobile apps, services and games.

Let us take a closer look at the key components, features, and benefits that make Visual Studio the preferred choice for developers worldwide.

One of the key advantages of Visual Studio lies in its seamless integration with Microsoft Azure and other cloud platforms. This ability facilitates cloud development and direct deployment from the IDE. Additionally, Visual Studio stands out for its rich ecosystem of extensions and plug-ins, which allows developers to customize their development environment according to their specific needs and preferences.

As a developer, understanding the significance of Visual Studio is important to navigate the vast landscape of software development tools. What sets Visual Studio apart is its unwavering commitment to providing developers with a holistic experience. Through an intuitive user interface and a plethora of productivity-enhancing features, Visual Studio streamlines the entire software development lifecycle, from coding and debugging, to testing and deployment. Its extensibility is another hallmark, as it allows developers to augment its capabilities with several extensions and plug-ins tailored to meet their specific needs.

The advantages of Visual Studio over other IDEs are manifold. Firstly, its unparalleled integration with Microsoft's ecosystem, including Azure cloud services, enables seamless development and deployment of cloud-native applications. Additionally, Visual Studio's support for many programming languages, frameworks, and platforms ensures that developers can work with their preferred tools and technologies without constraints. Moreover, its robust debugging tools like IntelliTrace and Live Share facilitate collaborative debugging and troubleshooting, which makes the development process more efficient.

At the heart of Visual Studio lie its main features that empower developers to streamline their workflows. These include:

- **Code Editor:** A powerful and feature-rich code editor with syntax highlighting, code completion, and refactoring capabilities
- **Debugger:** Advanced debugging tools for identifying and resolving software issues quickly and efficiently
- **Integrated Git Support:** Refers to the built-in Git integration for version control, which enables seamless collaboration and code management
- **Testing Tools:** Comprehensive testing tools like unit testing and code coverage analysis ensure the quality and reliability of applications

- **Extensibility:** A vibrant ecosystem of extensions and plug-ins that extend Visual Studio's capabilities and cater to diverse development needs
- **Cloud Integration:** Provides seamless integration with Microsoft Azure for cloud development, deployment, and management
- **Cross-Platform Development:** Grants support for cross-platform development across Windows, macOS, and Linux, while ensuring that applications can reach a wider audience

In essence, Visual Studio transcends the boundaries of a conventional IDE and is a comprehensive platform that empowers developers to bring their ideas to life. Its unparalleled versatility, robust feature set, and seamless integration with various services have set the standard for modern software development environments.

Reasons to Use Visual Studio

Visual Studio is not just another IDE; it is a complete ecosystem designed to enhance developer productivity and efficiency. Here is why developers choose Visual Studio:

- **Workload-Based Installer**: Install only what you need as per your specific development requirements to ensure a streamlined and efficient setup process.
- **Powerful Coding Tools and Features**: From code completion tools and compilers, to source control integration and AI-assisted development, Visual Studio provides everything developers need to build high-quality applications efficiently.
- **Multiple Language Support**: With support for languages such as C++, C#, JavaScript, TypeScript, Python, and more, Visual Studio accommodates the requirements of developers working across diverse tech stacks and projects.
- **Cross-Platform Development**: Whether building for Windows, web, mobile, or the cloud, Visual Studio offers tools and frameworks targeting a wide range of platforms, which enables developers to reach a broader audience with their applications.

- **Version Control Integration**: Collaborate seamlessly with team members using integrated version control systems like Git, which allows for efficient code management and collaboration.
- **AI-Assisted Development**: Leverage AI-powered tools like GitHub Copilot and IntelliCode to write code faster with greater accuracy and gain deeper insights into your codebase.

Choosing Your Visual Studio Edition

There are 3 versions of Visual Studio as provided by Microsoft:

- Community Edition
- Professional Edition
- Enterprise Edition

Let us understand the features and differences between these versions.

Community Edition

The VS Community edition is a free complimentary offering from Microsoft which encompasses the essential functionalities for .NET application development such as creation, building, debugging, and deployment, alongside the collaborative features of Visual Studio. Designed for non-enterprise entities, it can only accommodate a maximum of five users and caters primarily to students, freelancers, independent developers, and small businesses. It can only be used for open source, academic research, and classroom learning environments. It cannot be used for creating enterprise level applications.

Here are the key features of VS Community:

- **Complete Debugging Tools**: Facilitate code inspection during debugging processes
- **Performance and Diagnostics Hub**: Assesses application performance and memory utilization
- **Refactoring Tools**: Streamline code maintenance and adherence to coding best practices
- **Unit Testing Capabilities**: Navigate, execute, and aggregate results from unit tests

- **Peek Definition Functionality**: Provides expedited navigation to method or function definitions
- **VS Live Share**: Facilitates real-time collaborative development

While VS Community encompasses essential tools for daily use, certain scenarios like unit testing, memory management, or code inspection may necessitate additional functionalities. For instance, Live Unit Testing, which provides real-time feedback from continuously running unit tests, is not available in the VS Community edition.

Professional Edition

Visual Studio Professional is a licensed edition of Visual Studio that is available through a subscription model. This version is typically recommended for enterprise-level applications and development teams with more than five members.

The Visual Studio Professional edition includes all the core tools and features found in the Community edition, but also comes with some additional capabilities. For instance, it integrates CodeLens, a feature that helps developers quickly identify references, changes, and unit test information directly within the code.

In terms of pricing, the current subscription cost for an individual user of Visual Studio Professional is $45 per month. This subscription provides several additional benefits, including:

- Access to training resources
- Technical support services
- Integration with the basic Azure DevOps plan

The higher cost of the Professional edition is justified by the expanded feature set and the enterprise-grade tools it provides. Organizations with more complex development requirements and larger team sizes are likely to find the Professional edition better suited to their needs.

While the Community edition covers the essential needs of many individual developers and small teams, the Professional subscription unlocks a more robust set of capabilities for larger and more sophisticated software projects.

Enterprise Edition

Visual Studio Enterprise is the highest-tier subscription offering within the Visual Studio product lineup. Building upon the capabilities of the Professional edition, the Enterprise subscription includes all the features available in the Community version, along with additional advanced tools and capabilities.

One of the exclusive features of the Enterprise edition is IntelliTrace, which is a powerful debugging tool that records the state of the application. It allows developers to inspect the system in detail before an exception occurs. This provides valuable information about events, exceptions, and function calls, which can be helpful during the debugging process.

Other notable features of the Visual Studio Enterprise edition include:

- **Live Unit Testing:** Automatically re-runs unit tests after code changes and provides immediate feedback
- **Snapshot Debugger:** Saves snapshots of the application during debugging, thereby enabling in-depth analysis of errors and issues
- **Performance Analysis Tools:** Advanced tools for profiling and optimizing the performance of mobile applications
- **Architectural Layer Diagrams:** Visualize the logical architecture of your application for designing and planning

The Visual Studio Enterprise subscription comes with a monthly fee of $250 per user. However, this subscription includes several additional benefits, such as:

- $150 in monthly Azure credits
- Access to Power BI Pro
- Enhanced Azure DevOps features, including test plans
- All the features and capabilities that are available in the lower-tier Visual Studio editions

To compare the pricing and features of the different Visual Studio offerings, you can visit the official Visual Studio pricing page at https://visualstudio.microsoft.com/vs/pricing/

Here is the official link to download Visual Studio: https://visualstudio.microsoft.com/downloads/

**This book uses the Visual Studio 2022 Professional version as base edition for all its examples*.

Visual Studio 2022 is a 64-bit application. Please check system requirements before installing it for your development machine.

Visual Studio Code

It is a lighter standalone source code editor that runs on Windows, macOS, and Linux. It is the top pick for JavaScript and Java and web developers as it provides several extensions to support just about any programming language.

Features of Visual Code:

- **Cross-Platform Compatibility**: Visual Studio Code is a lightweight, open-source code editor that runs on Windows, macOS, and Linux. This makes it a versatile choice for developers operating different platforms.
- **Extensive Language Support**: VS Code supports a wide range of programming languages, including C++, C#, Java, JavaScript, TypeScript, Python, and many more, with excellent syntax highlighting and code completion.
- **Integrated Terminal**: VS Code includes a built-in terminal that allows developers to run and interact with command-line tools and shells directly within the editor.
- **Intelligent Code Editing**: VS Code provides advanced code editing features like IntelliSense, code folding, bracket matching, and code formatting to enhance productivity.
- **Debugging Tools**: VS Code includes robust debugging tools that allow developers to set breakpoints, step through code, and inspect variables, making it easier to identify and fix issues.
- **Extensions and Customization**: VS Code has a vast ecosystem of extensions and plugins that can be installed to add support for additional languages, frameworks, and development tools, as well as customize the editor's appearance and behavior.
- **Git and Source Control Integration**: VS Code integrates seamlessly with Git and other version control systems, which allow developers to

manage their code repositories, view diffs, and perform common Git operations directly within the editor.

- **Live Share**: This feature enables real-time collaboration, allowing developers to share their workspace, code, and debugging sessions with others, thereby fostering remote collaboration.
- **Lightweight and Fast**: Compared to the full-featured Visual Studio IDE, VS Code is a lightweight, fast, and efficient code editor that can run on lower-powered machines.
- **Open-Source and Community-Driven**: Visual Studio Code is an open-source project, which allows developers to make contributions and improvements.

In conclusion, Visual Studio empowers developers with the tools and resources they need to succeed. With its robust features, comprehensive support, and seamless integration, Visual Studio remains the IDE of choice for developers worldwide, as it enables them to build with confidence and innovate without being hindered by limits.

Visual Studio 2022 System Requirements for Installation

Visual Studio 2022 is supported on the following 64-bit operating systems (as it is not supported on 32 bit operating systems):

Windows 11

- Home, Pro, Pro Education, Pro for Workstations, Enterprise, and Education
- Minimum supported OS version or higher

Windows 10

- Home, Professional, Education, and Enterprise
- Minimum supported OS version or higher

Windows Server

- Windows Server 2022: Standard and Datacenter
- Windows Server 2019: Standard and Datacenter

- Windows Server 2016: Standard and Datacenter

Hardware Requirements

- ARM64 or x64 processor
- Minimum 4GB RAM
- Minimum 20-50GB of available space
- Administrator rights are required for installing or updating Visual Studio.

Visual Studio 2022: Harnessing the Power of 64-Bit Architecture

Visual Studio 2022 brings a significant yet subtle enhancement under the hood: a transition to a 64-bit architecture. While this change may not be immediately visible to users, its impact on performance is profound. By leveraging the capabilities of modern 64-bit CPUs found in today's laptops and PCs, Visual Studio 2022 enhances efficiency and reduces latency, particularly while you are executing multiple tasks concurrently.

One of the most notable advantages of moving to a 64-bit architecture is the removal of the previous 4 GB memory access limitation which is inherent in 32-bit systems. With this upgrade, Visual Studio can tap into a significantly larger pool of memory, which is limited only by the physical hardware's capabilities. This expansion in memory access translates to smoother operation, while mitigating time constraints and preventing frustrating IDE freezes or crashes.

To illustrate the difference, one needs to compare the performance of Visual Studio 2022 when it is running on a 64-bit platform alongside its predecessor, Visual Studio 2019, which operates on a 32-bit architecture. Task Manager in Windows provides a tangible demonstration of the enhanced efficiency, which highlights the distinct advantages of the 64-bit environment.

Moreover, this shift to a 64-bit architecture represents a forward-looking investment in performance optimization. The Visual Studio development team has diligently fine-tuned the software to deliver improved performance across various scenarios, which is particularly noticeable when working on large-scale projects. As Visual Studio evolves in future iterations, users can

expect continued enhancements in performance, which provides for a more seamless and efficient development experience.

In essence, the adoption of a 64-bit architecture in Visual Studio 2022 marks a significant milestone in optimizing performance, as it lays a robust foundation for the software's evolution and ensures a smoother, more productive development workflow for users.

Introducing .NET 8: A Powerful Upgrade for Developers

.NET, the popular open-source software framework, has just released its latest version, which is called .NET 8. This new release brings a host of exciting improvements and features that benefit developers operating across various domains like web, desktop, mobile, and more.

New Features in .NET 8

- **Native AOT for Apps**: .NET 8 introduces a new feature called Native AOT (Ahead-of-Time) compilation, which allows developers to create native, standalone applications. This advancement offers improved performance and smaller application sizes, thereby making it especially beneficial for resource-constrained environments.
- **C# 12 Compatibility**: The .NET 8 framework is fully compatible with the latest version of the C# programming language, C# 12. This means that developers can leverage the latest language features and enhancements to write more efficient and expressive code.
- **Continuous Improvements**: As with each annual release of .NET, .NET 8 builds upon the foundation of its predecessors and introduces various refinements and optimizations to improve overall performance, stability, and developer productivity.

Seamless Integration with Visual Studio 2022

The release of .NET 8 is well-timed, as it coincides with the availability of Visual Studio 2022. Developers can now seamlessly create, compile, and publish their projects using the latest version of the .NET framework directly within the Visual Studio 2022 IDE.

Staying Up-to-Date

To learn more about the specific improvements and features in .NET 8, we encourage you to visit the official Microsoft documentation at https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8.

Whether you are a seasoned .NET developer or just starting out, the release of .NET 8 presents an exciting opportunity to leverage the latest advancements in the framework and take your applications to new heights.

Visual Studio 2022 Installation Guide

Improving the Visual Studio Installation Experience:

With each iteration, the installation process for Visual Studio (VS) has seen enhancements, which offer users greater convenience and flexibility. Notably, the latest versions allow for seamless upgrades and customization of workloads to cater to various project requirements.

- **Obtaining the Installer**: Acquiring the VS installer is a straightforward process, which is devoid of complexities or the need for an account. Users can easily access the installer by visiting the official Visual Studio website at visualstudio.microsoft.com/downloads. From there, navigate to the Visual Studio section and select the desired option such as the Community edition to initiate the download process.

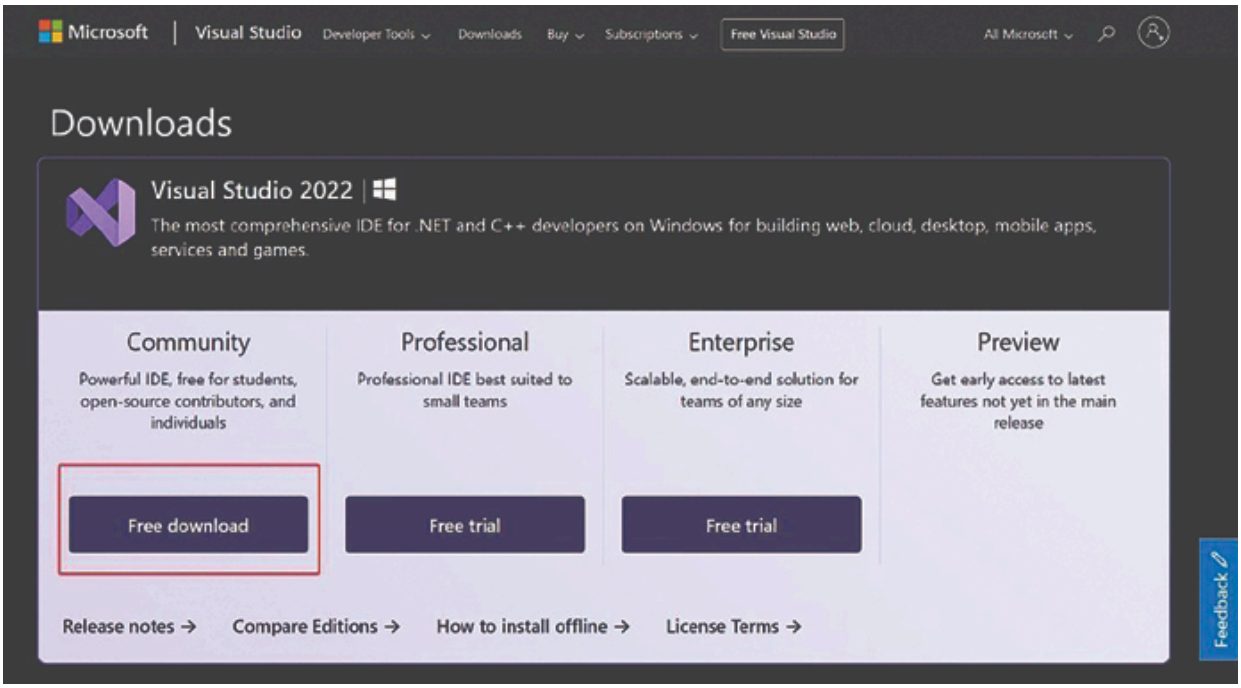


Figure 1.1: *Visual Studio Versions*

Should users prefer older versions, links to download VS 2022 and previous iterations are available at https://visualstudio.microsoft.com/vs/older-downloads.

- **Installation Procedure**: Upon downloading the VS installer, running it triggers an update process for the installer itself, which ensures that it is up-to-date with the latest improvements. The installer is designed to seek out and incorporate the latest updates from the development team seamlessly.

Updating the VS Installer

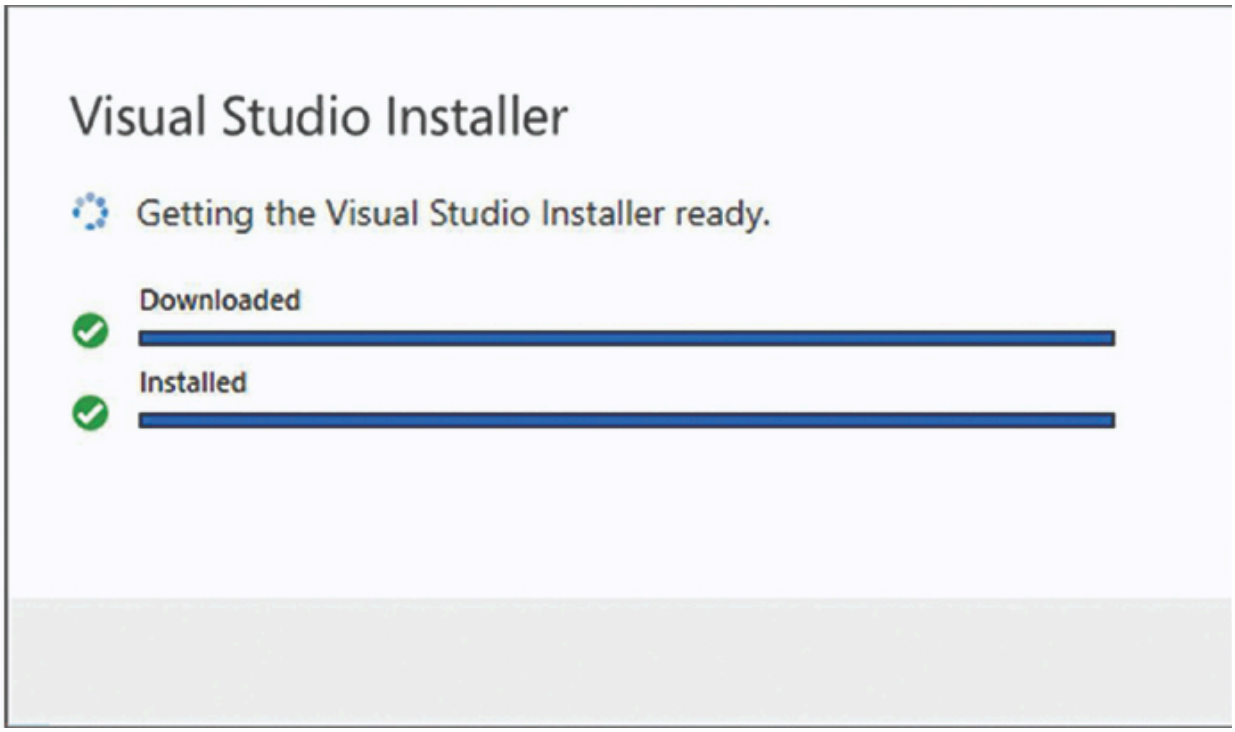


Figure 1.2: *VS Installer*

Upon completion of the update, users are presented with the initial screen of the installer, which features four primary configuration options that are accessible from the top menu.

The VS Workloads Tab

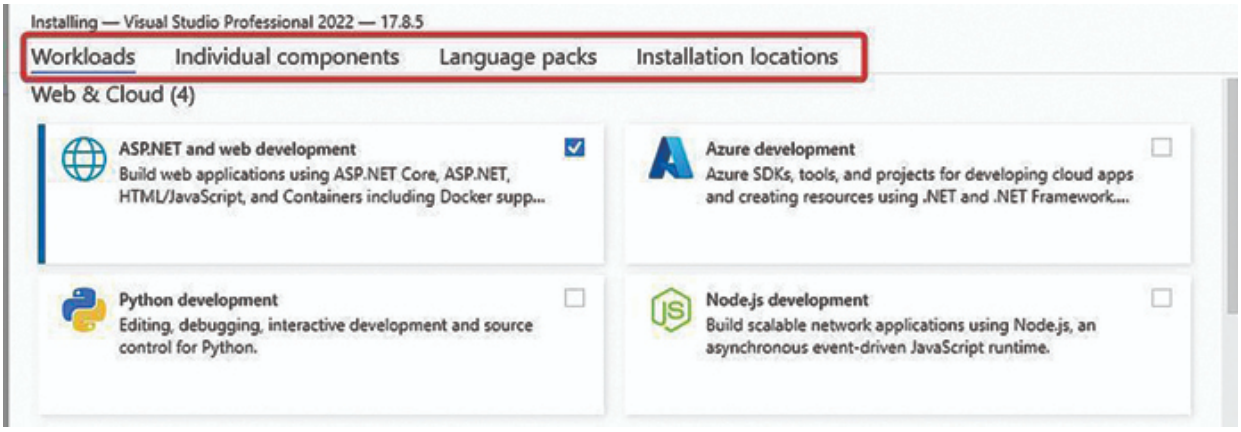


Figure 1.3: *VS Workloads Tab*

- **`Workloads`**: This tab offers a selection of pre-configured workloads tailored to different project types which encompass mobile, desktop,

Python, and other technologies. Each workload includes a curated set of tools and components specific to the chosen technology stack.

- **`Individual Components`**: Here, users can peruse and select individual components to install, even if they are not part of the selected workloads, which allows for a more granular customization of the installation.
- **`Language Packs`:** Users can choose from a list of 14 languages for the VS interface for seamless development in their preferred language(s). This flexibility extends to the ability to switch between languages during the development process.
- **`Installation Locations`:** This tab provides insights into the space requirements for each installation path, which facilitates informed decision-making.

Once satisfied with the configurations, users can proceed with the installation process by clicking the Install button. Notably, users have the flexibility to modify workloads, individual components, and language preferences by re-running the downloaded installer.

VS Installation in Progress

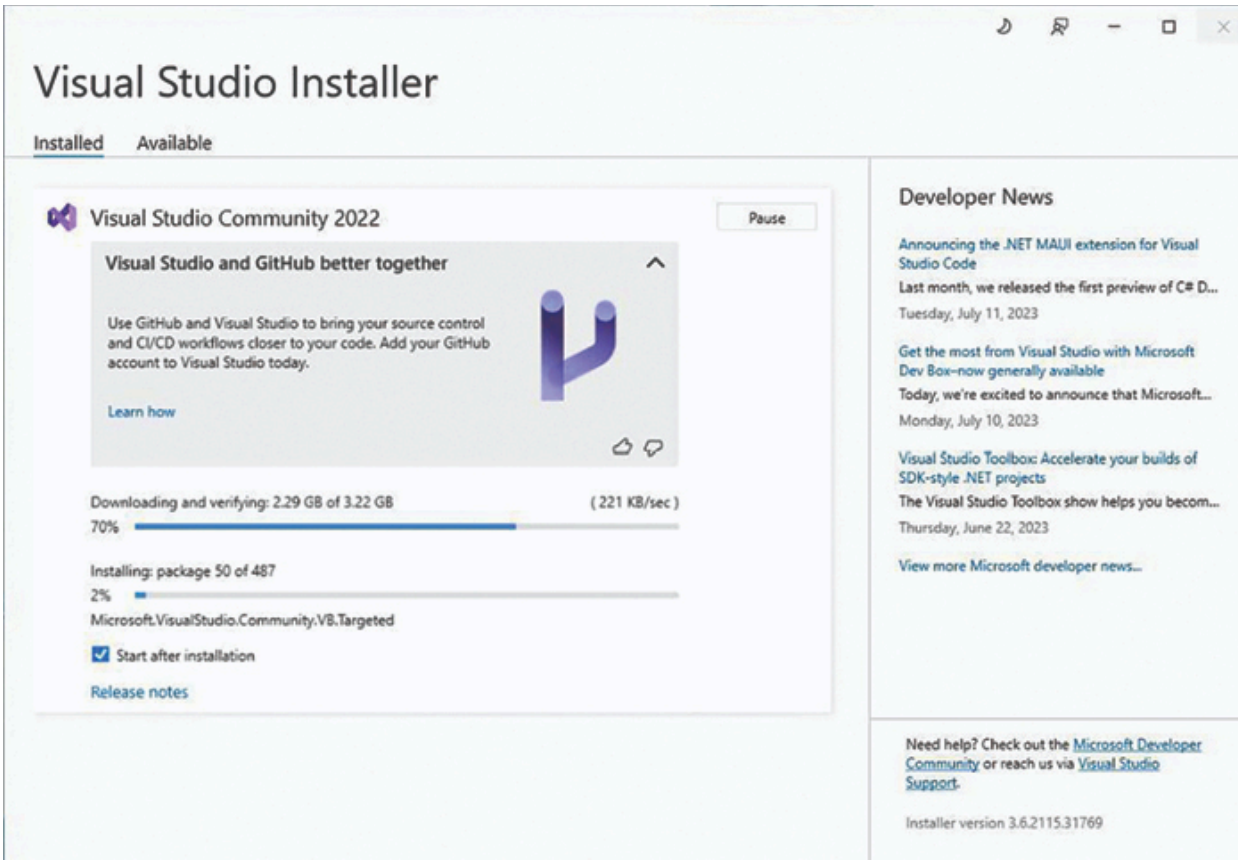


Figure 1.4: *Visual Studio Installer Progress Page*

Upon initiating the installation, users are presented with a detailed overview of the download and installation progress, while ensuring transparency throughout the process.

With the installation complete, users are now ready to explore the Visual Studio interface, and embark on their development journey.

Introduction to Visual Studio IDE

Now that Visual Studio 2022 is successfully installed, you will encounter the startup screen presenting options for creating, cloning, or opening projects.

Tip: *Always run the Visual Studio in "`run-as Administrator`" mode.*

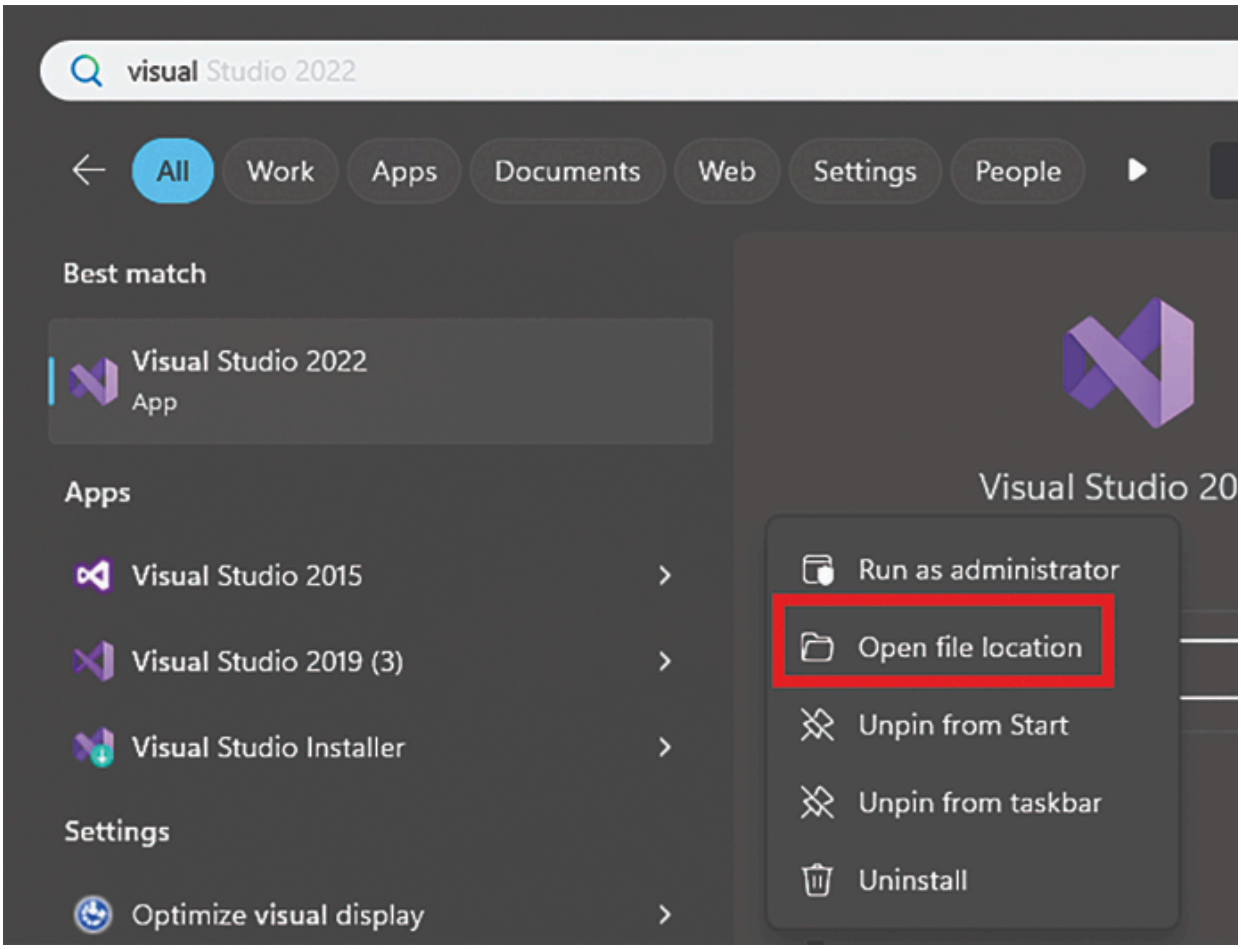


Figure 1.5: *Open File Location*

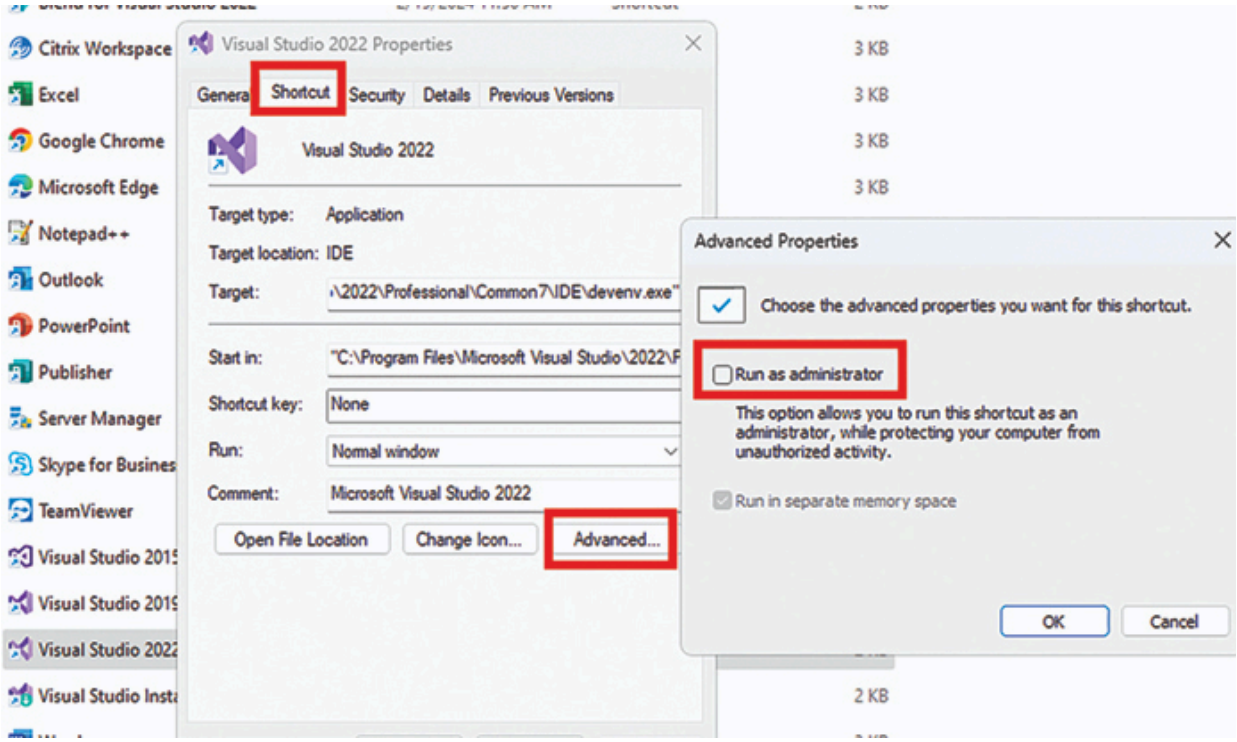


Figure 1.6: *Run as administrator*

Check "**`Run as administrator`**" and click on **`Ok`**. Now every time you open the visual studio it will always run in Admin mode by default. So, you will not encounter any hassle of remembering to manually run VS in admin mode.

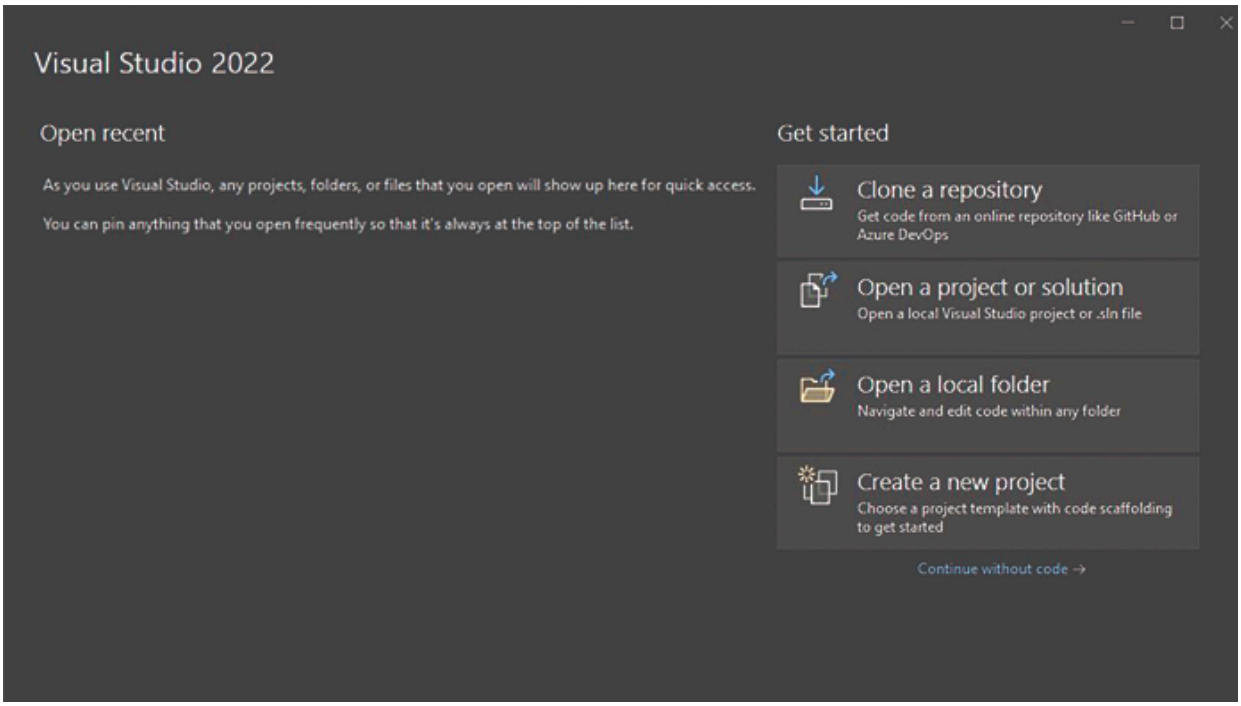


Figure 1.7: *VS Getting Started Screen*

It is noteworthy that multiple versions of Visual Studio can coexist on the same machine. This flexibility is particularly useful for maintaining compatibility with older technologies like Visual Studio 2010 for Silverlight development.

Let us delve into the interface without creating a project yet by selecting the "**`Continue without code`**" option. Upon entering the main interface, you will notice the "**`Options`**" menu at the top, which offers a plethora of tools for various purposes, alongside buttons for common actions like starting application execution and debugging.

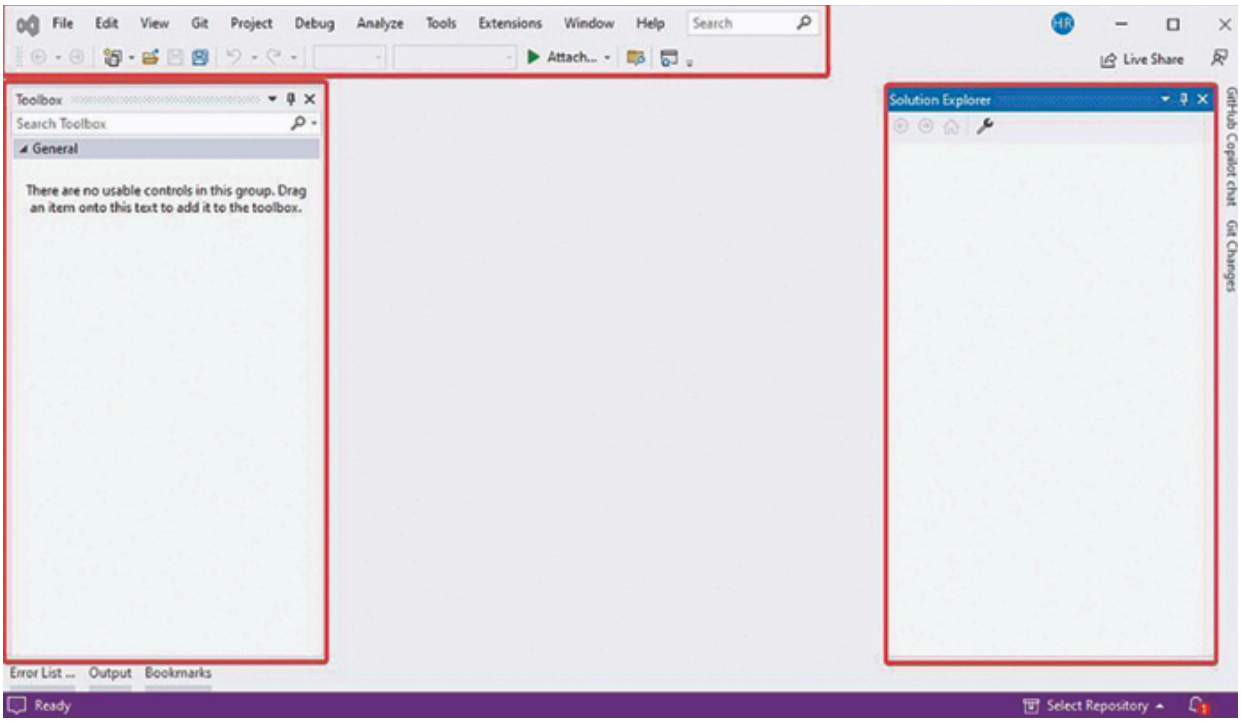


Figure 1.8: *VS General Interface*

The available tools may vary depending on the open project. For instance, since no project is created yet, the toolbar displays minimal options.

In the left and right sections, you will find panels containing tools specific to the type of project you are working on. These panels typically include commonly used tools like Solution Explorer, Toolbox, Properties, and Code Editor.

Customizing Panels

Visual Studio panels provide access to project-specific tools. These panels consist of tools and document editors that are essential for development tasks.

Visual Studio 2022 offers a wide array of tools that you can show or hide according to your requirements. Access these tools from the "**`View`**" menu, where commonly used ones are listed upfront, with additional options available under "**`Other Windows`**."

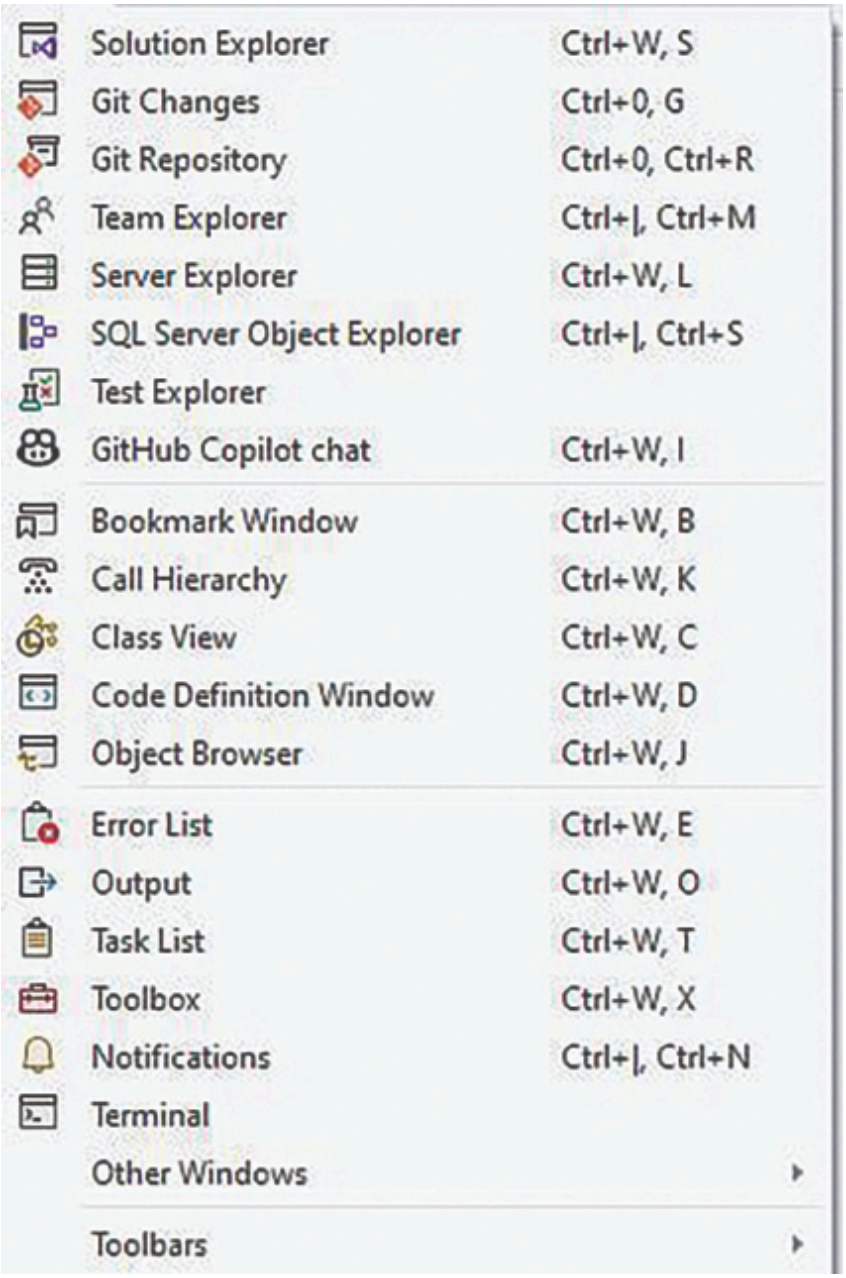


Figure 1.9: *View Menu Options*

To add a tool to a panel, simply select it. For instance, adding the Server Explorer tool places it in the left panel, while adding the Output tool places it at the bottom pane.

One of the advantages of Visual Studio is the ability to arrange tool panels according to your preferences. Each panel comprises five sections where tools can be placed. The IDE simplifies this process by indicating possible

locations for tool placement. Additionally, you can detach panels from the main window for standalone use.

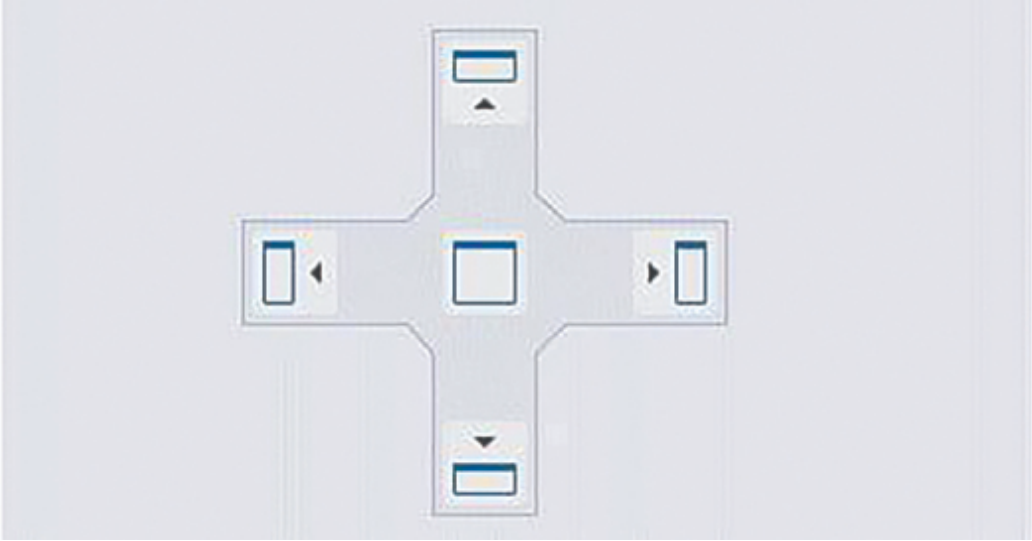


Figure 1.10: *Showing Panel Locations*

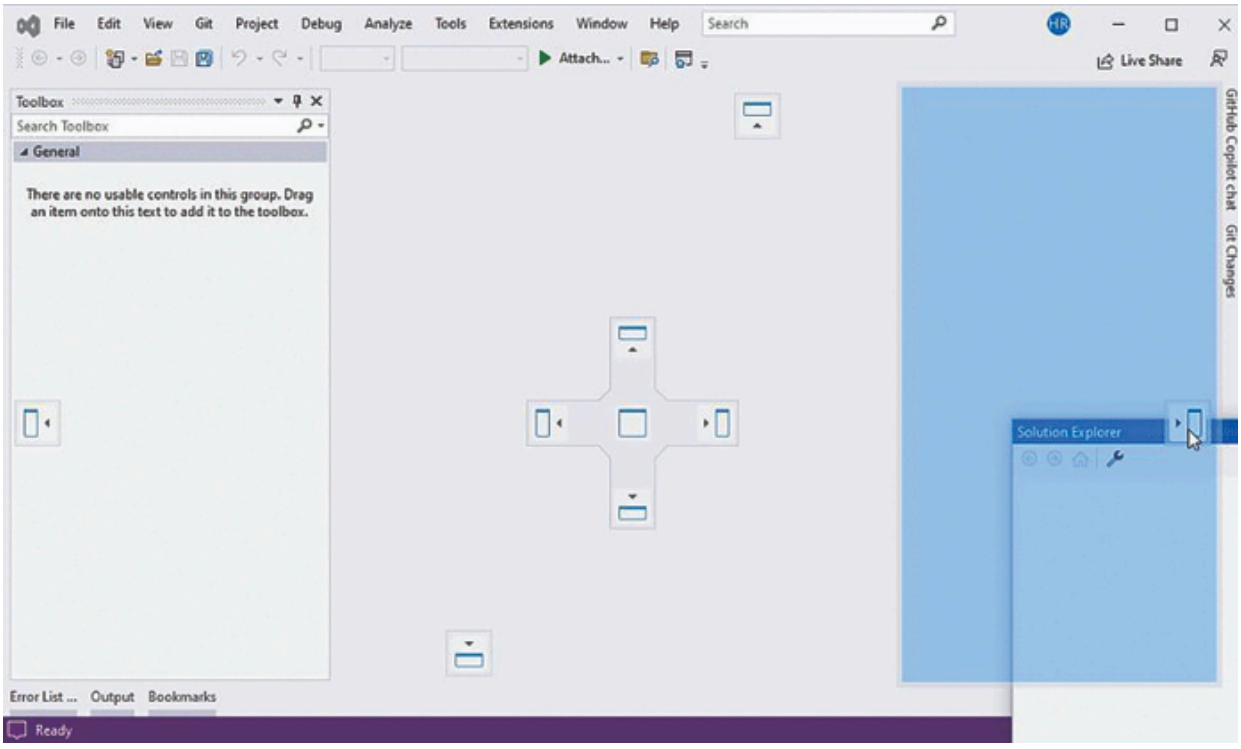


Figure 1.11: *Available Panel Locations*

Working with Documents

Document editors, like Code Editor, offer special options to enhance productivity. Right-clicking on an open document tab reveals options like floating the editor into a separate window, pinning the tab, or adjusting the tab layout to suit your workflow.

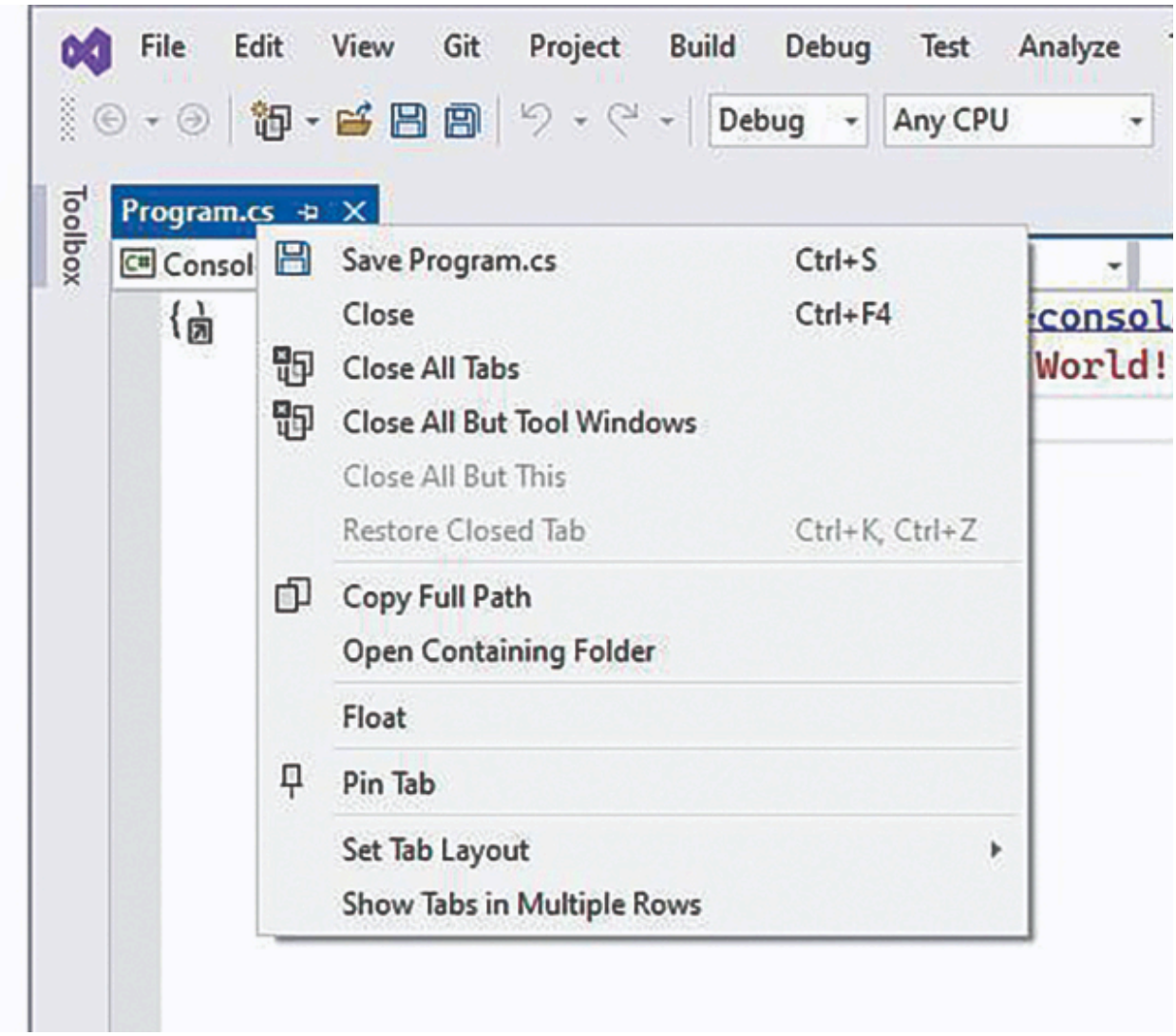


Figure 1.12: *Context Menu for Code Editor*

Utilize the *Ctrl + F1* shortcut to access Visual Studio documentation for guidance and updates.

- **Customizing the IDE:** Software developers often have different preferences when it comes to coding environments. Visual Studio 2022 offers various customization options, including theme selection and font customization.

- **Selecting a VS Theme:** Upon startup, Visual Studio defaults to a dark theme, though several pre-installed themes are available. These themes cater to different preferences and environmental conditions, with the dark theme offering reduced eye strain in low-light conditions.

To change the theme, navigate to the "`Tools`" menu, select "`Themes`", and choose from the available options. Additional themes can be obtained from the Visual Studio Marketplace.

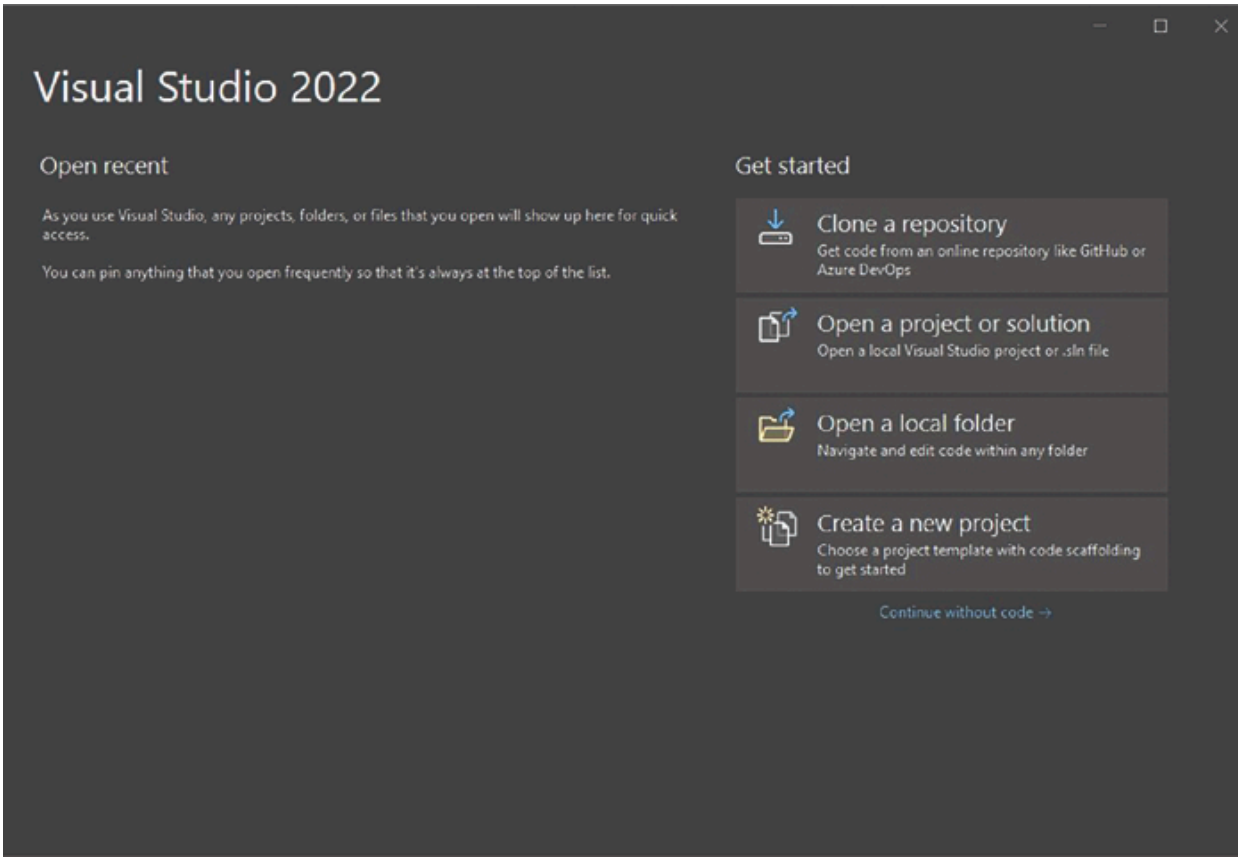


Figure 1.13: *Visual Studio Marketplace*

- **Customizing Fonts and Colors:** Visual Studio allows for comprehensive customization of fonts and colors to suit individual preferences. You can access these settings from the "`Tools`" menu, where you can adjust the environment font and the font settings for the code editor.

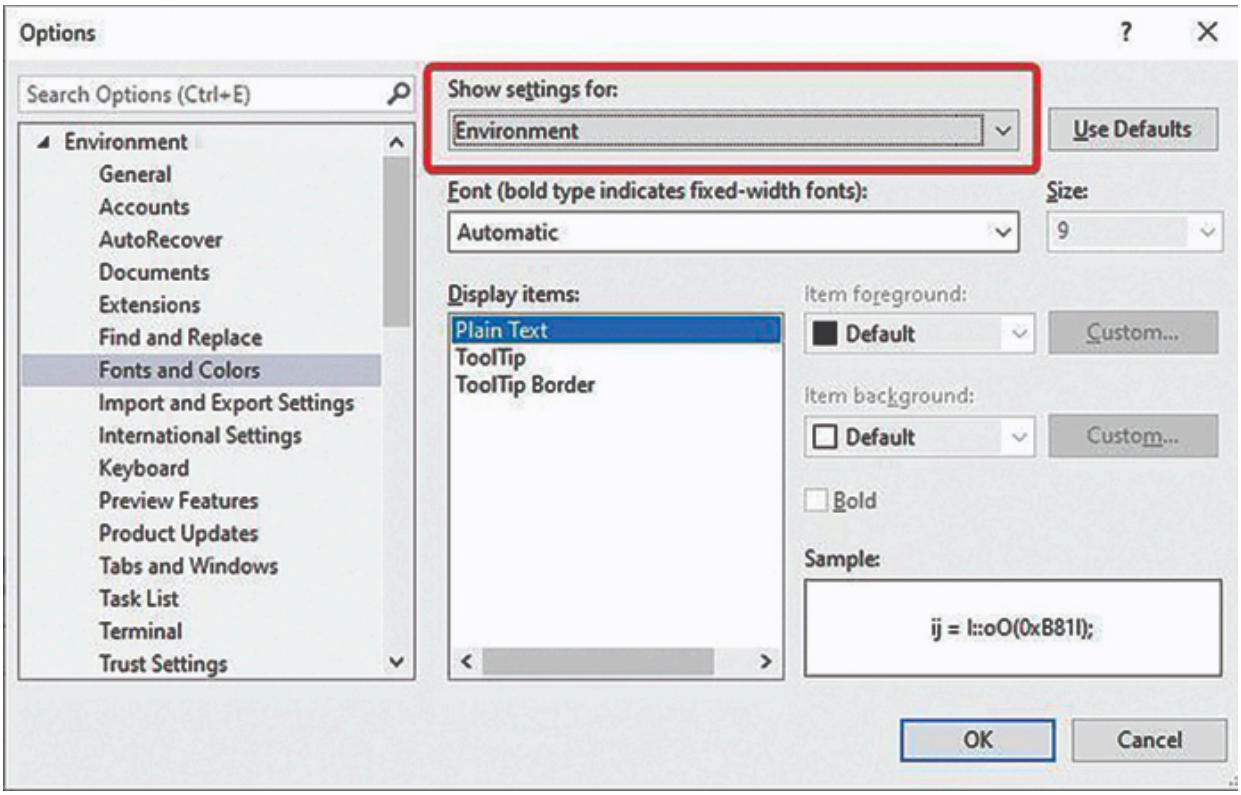


Figure 1.14: *Changing the Environment Font Settings*

Moreover, Visual Studio introduces the Cascadia font as the default font, which is known for its improved code readability.

Synchronizing Accounts and Settings

Visual Studio facilitates synchronization of configurations across different devices through a Microsoft account. Key settings such as user-defined window layouts, themes, fonts, keyboard shortcuts, and text editor settings can be synchronized seamlessly.

To enable or disable synchronization, access the "**`Tools`**" menu, navigate to "**`Options`**," and select the "**`Environment`**" and "**`Accounts`**" sections.

[VS – Sign in (at the top right of the main screen)](link-to-image)

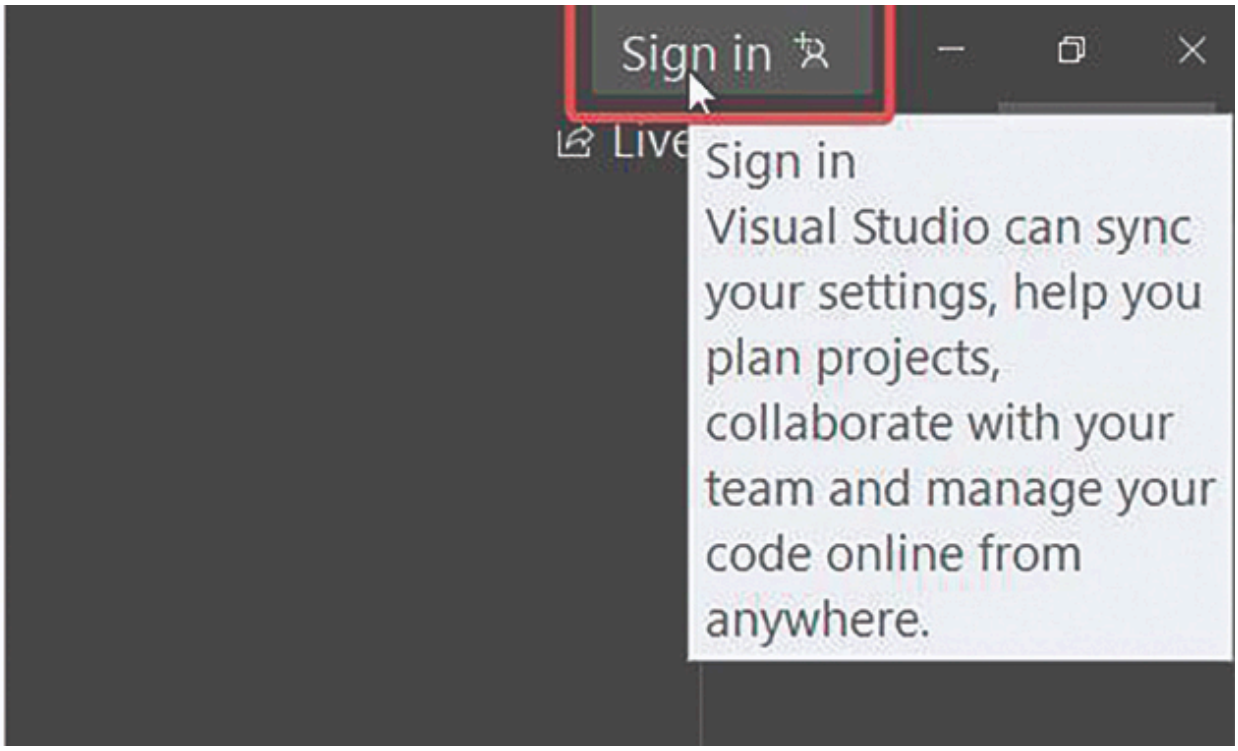


Figure 1.15: *VS Account Sign In*

Customizing the Menu Bar and Toolbars

Menus and toolbars provide quick access to frequently used commands. Visual Studio allows users to customize these elements to streamline their development workflow.

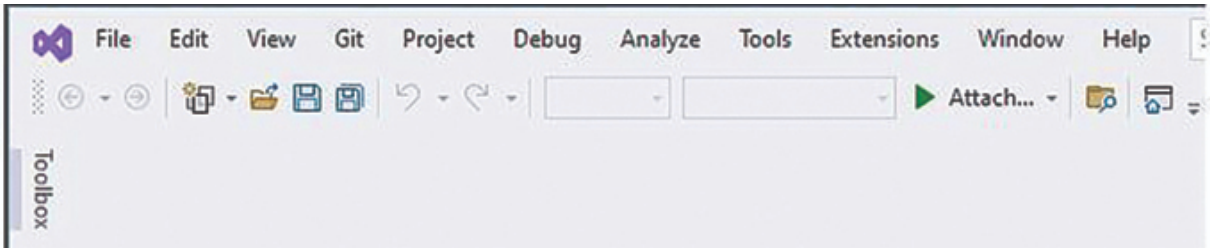


Figure 1.16: *VS Menu Bar*

Customizing the Menu Bar

The menu bar, located at the top of the IDE, offers dropdown menus for executing tasks and accessing tools. Customize the menu bar by selecting the "`Tools`" menu, followed by "`Customize`." From here, you can modify the

main and secondary menu bars by adding, deleting, or rearranging menu items.

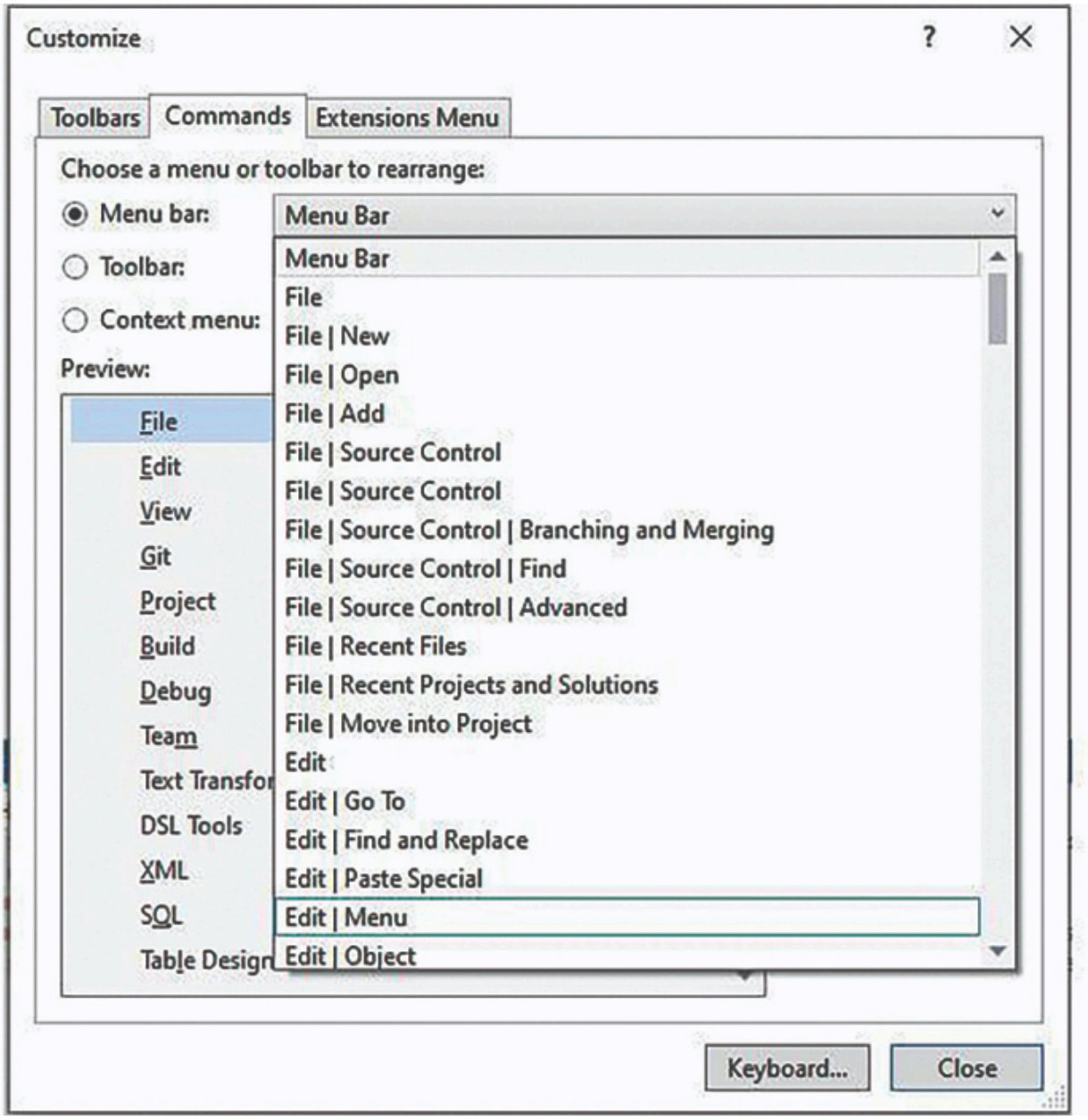


Figure 1.17: *Customizing Menu Bar*

Select the command you want to add in menu bar and click **OK** (as shown in image).

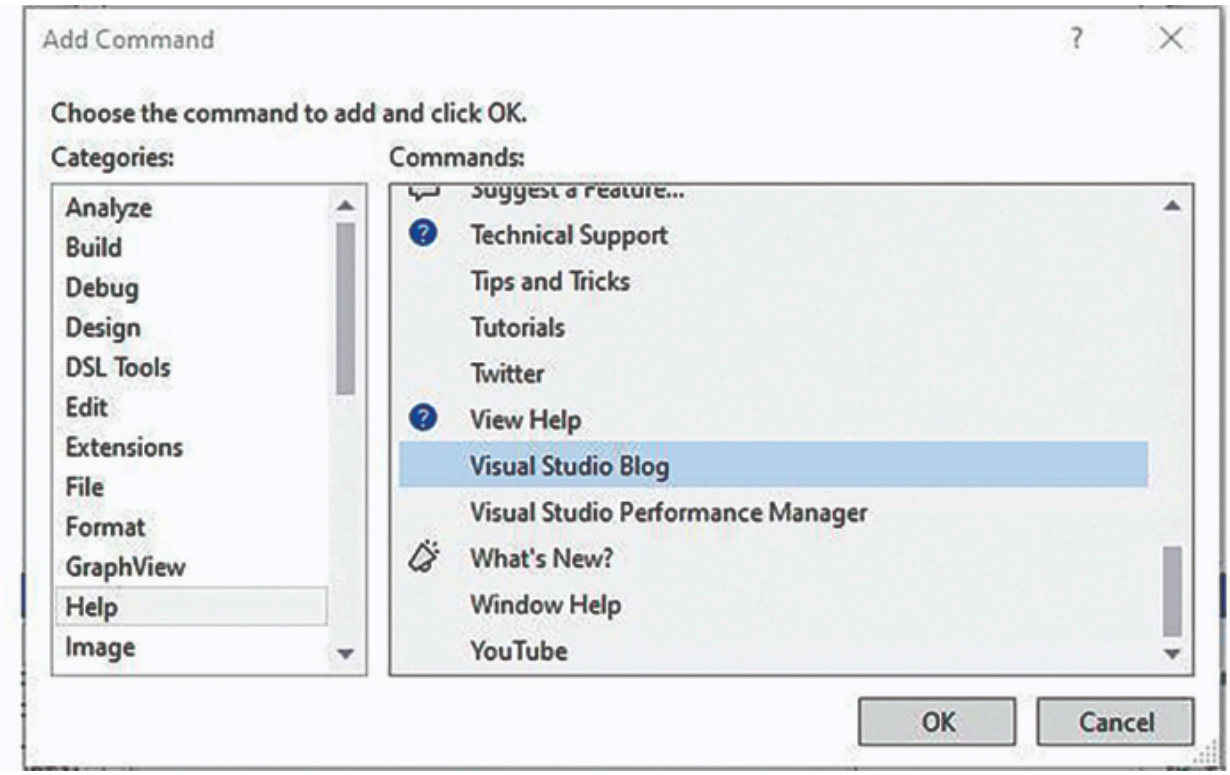


Figure 1.18: *Add Command*

You can delete the menu item if you think you will not be using it often during development.

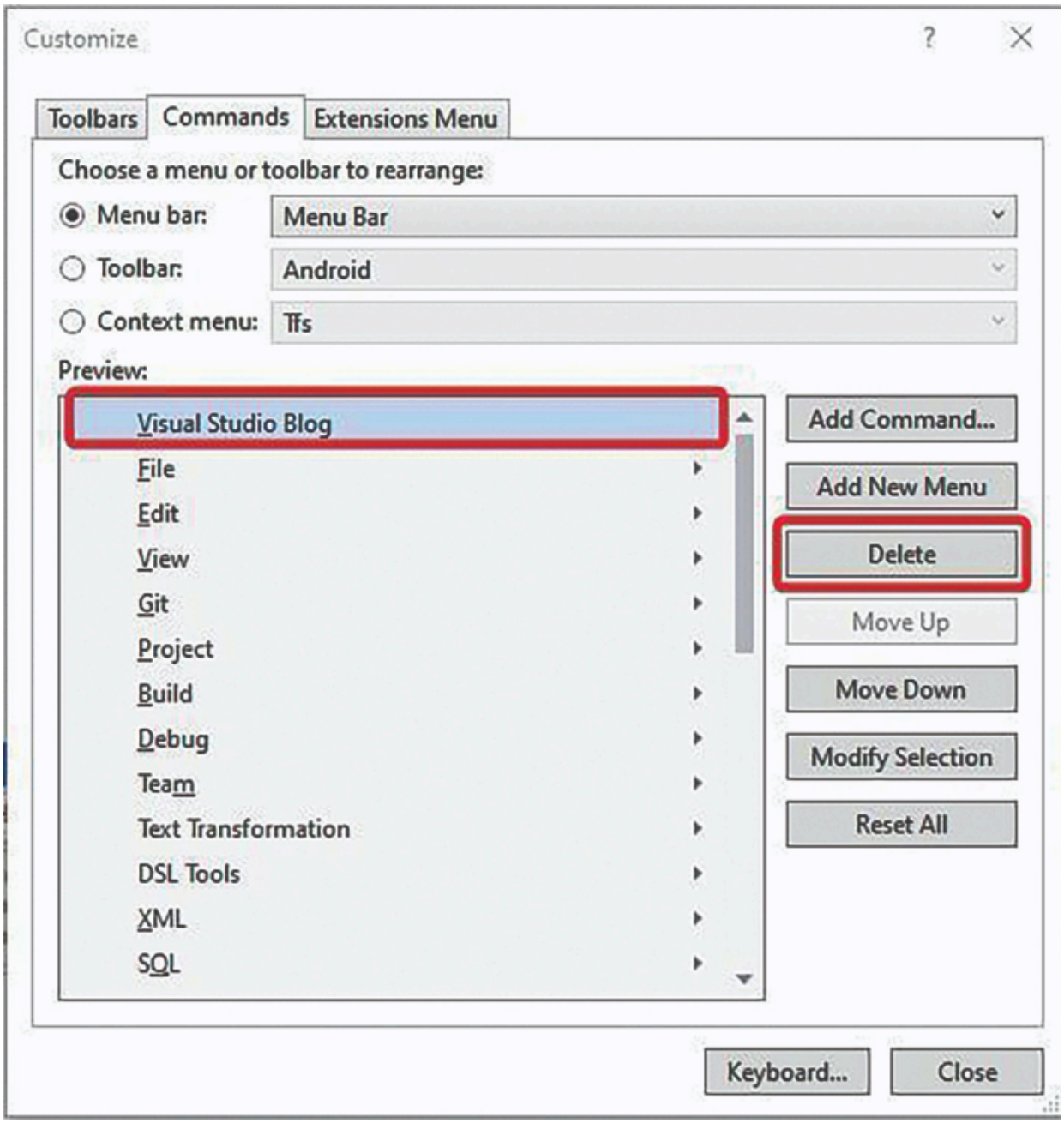


Figure 1.19: *Customizing Menu Bar Items*

Customizing Toolbars

Toolbars offer direct access to commands without the need for dropdown menus. You can customize toolbars by selecting the desired options from the "`Tools`" menu and "`Customize`" submenu. You can add or remove toolbars and commands to tailor the IDE to your preferences.

Toolbar can also be modified to add or remove which options to be displayed in IDE.

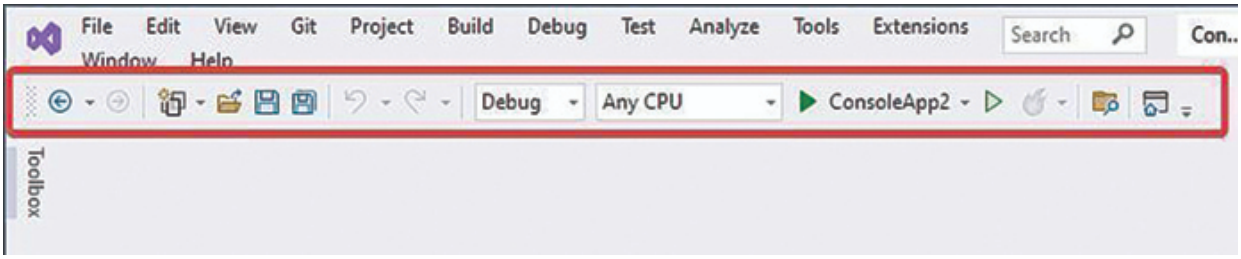


Figure 1.20: *Toolbar Icons*

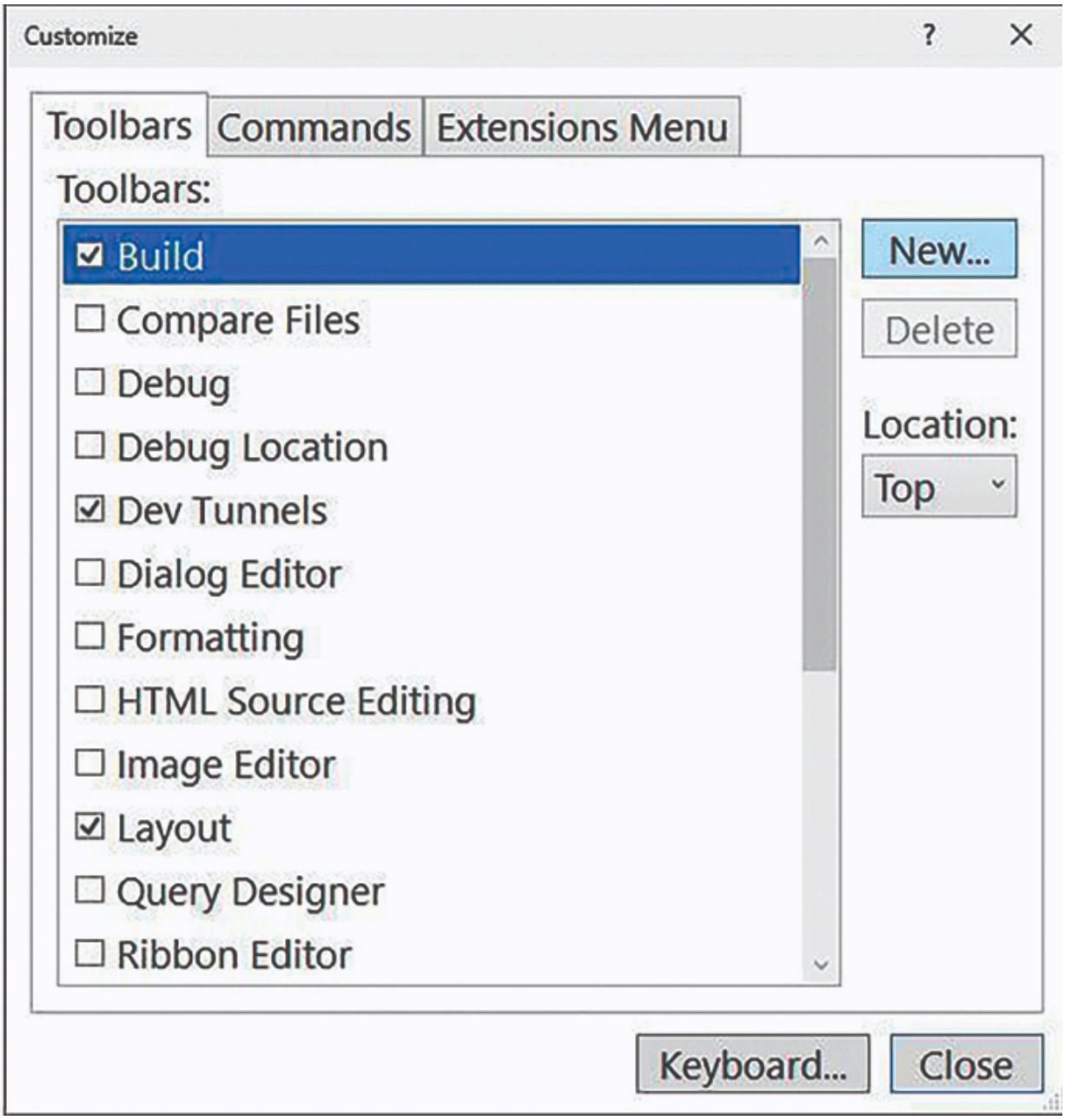


Figure 1.21: *Customizing ToolBar Items*

Figure 1.22: *Additional Menu Items under View→Toolbars*

Accessibility Features

Visual Studio includes accessibility features to accommodate users with visual impairments. These features include a high-contrast blue theme and the ability to add sounds for various IDE events. Users can also utilize keyboard shortcuts for common tasks, which enhances accessibility and productivity.

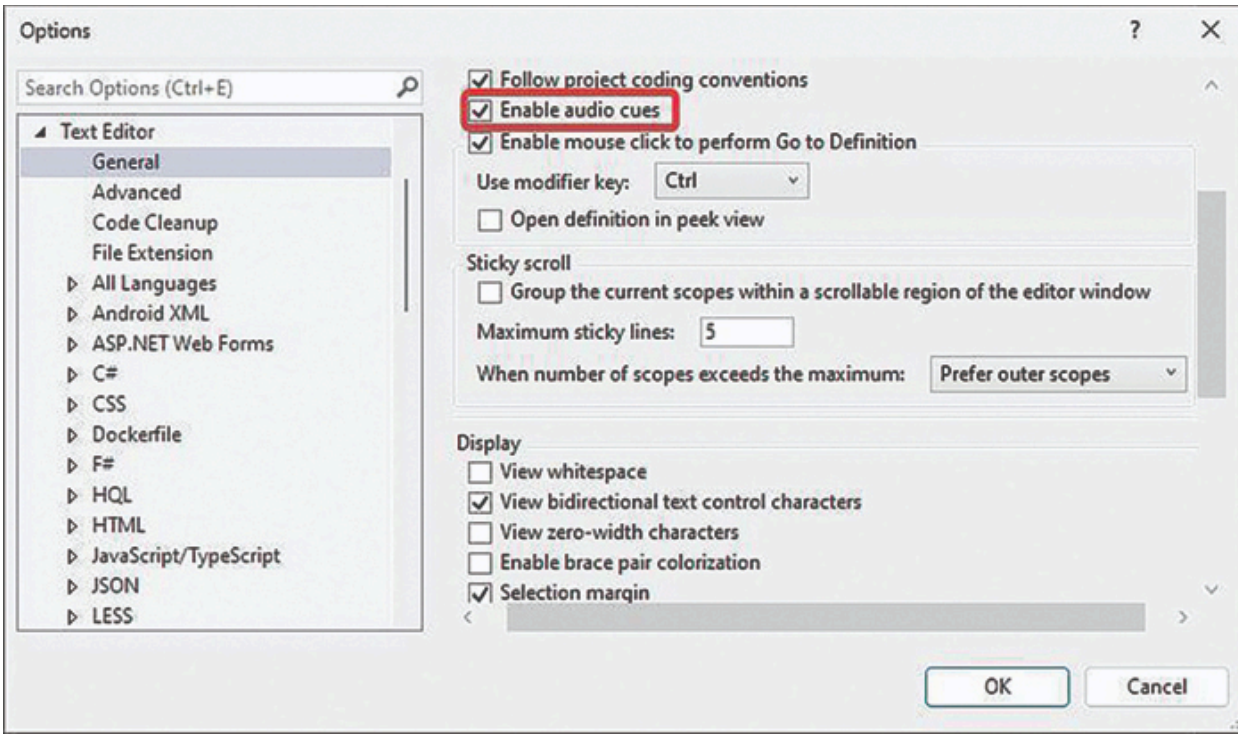


Figure 1.23: *Accessibility Features*

In summary, Visual Studio offers a highly customizable environment to meet the diverse needs of developers. From theme selection to accessibility features, users can tailor the IDE to optimize their development experience.

Understanding the VS Menu Bar

The Visual Studio menu bar is the primary navigation tool in the IDE, and it provides access to a wide range of features and options that are essential for the software development process. Let us go through each menu and explain its purpose and the options it offers to a new software engineer:

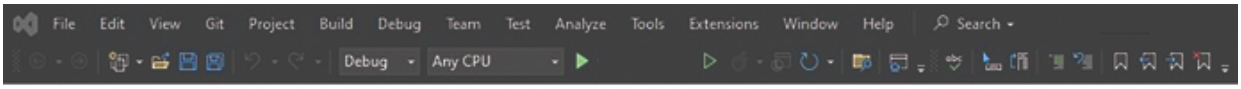


Figure 1.24: *VS Menu Bar*

These options may vary slightly depending on the specific version of Visual Studio 2022 you are using and any installed extensions or customizations.

`File` Menu

The "**`File`**" menu typically contains various options and features related to file operations, project management, and other general functionalities. Here is a comprehensive list of options you might find under the "**`File`**" menu along with their typical uses:

`New`

- Create a new project, file, or item within your project.
- Select from various project templates and file types depending on your development needs.

`Open`

- Open an existing project, solution, file, or folder in Visual Studio.
- Navigate to the location of the desired file or project and open it directly from the file system.

`Close`

- Close the currently active document or project without closing the entire Visual Studio instance.
- Useful for managing multiple open files or projects.

`Save`

- Save any changes made to the active document or project.
- If the document or project has not been saved previously, this option will prompt you to specify a location and filename.

`Save As`

- Save a copy of the active document or project with a different filename or in a different location.
- This option is helpful for creating new versions of a file without overwriting the original.

`Save All`

- Save all open documents and projects in Visual Studio.
- It is convenient when you have multiple unsaved changes across several files and want to save them all at once.

Print

- Print the contents of the active document or active page.
- Configure print settings such as printer selection, page layout, and number of copies.

Recent Files

- Display a list of recently opened files
- Quickly access documents you have been working on without navigating through your file system.

Recent Projects and Solutions

- Display a list of recently opened projects and solutions.
- Quickly resume work on previous projects.

Exit

- Close Visual Studio and exit the application.
- Used when you are done with your work and want to close Visual Studio entirely.

New Repository

- Create a new Git repository directly within Visual Studio.
- Start version controlling a new project from the beginning using Git.

Create a Git repository (**File** → **New** → **Repository**) :

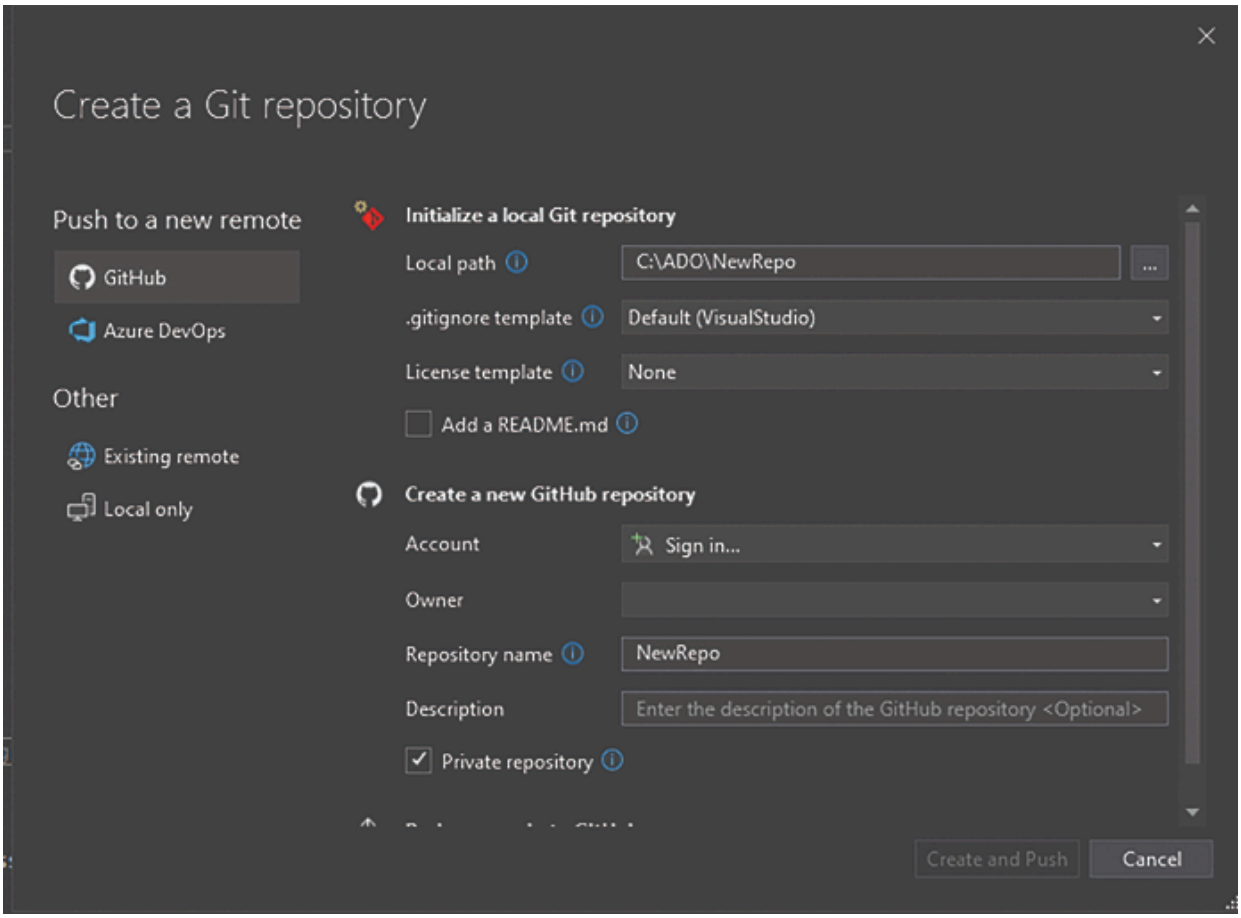


Figure 1.25: *Create Git Repository*

In Visual Studio 2022, you can create a new repository by navigating to "**`File`** > **`New`** > **`New Repository`**" option under the "**`File`**" menu allows you to create a new Git repository directly within Visual Studio. This feature is particularly useful when you want to start a new project and version control it from the beginning using Git.

When you select the "**`New Repository`**" option, Visual Studio will guide you through the process of creating a new Git repository. This typically involves specifying a location for the repository on your local filesystem, providing a name for the repository, and optionally choosing a **`.gitignore`** file template to exclude certain files from version control.

Once the repository is created, Visual Studio will initialize it with an initial commit containing any files you have added to the project. From there, you can continue to work on your project, staging and committing changes as needed using the built-in Git integration in Visual Studio.

Creating a new repository directly within Visual Studio streamlines the process of setting up version control for your projects, making it easier to manage changes and collaborate with others on your code.

`File > Clone a Repository`

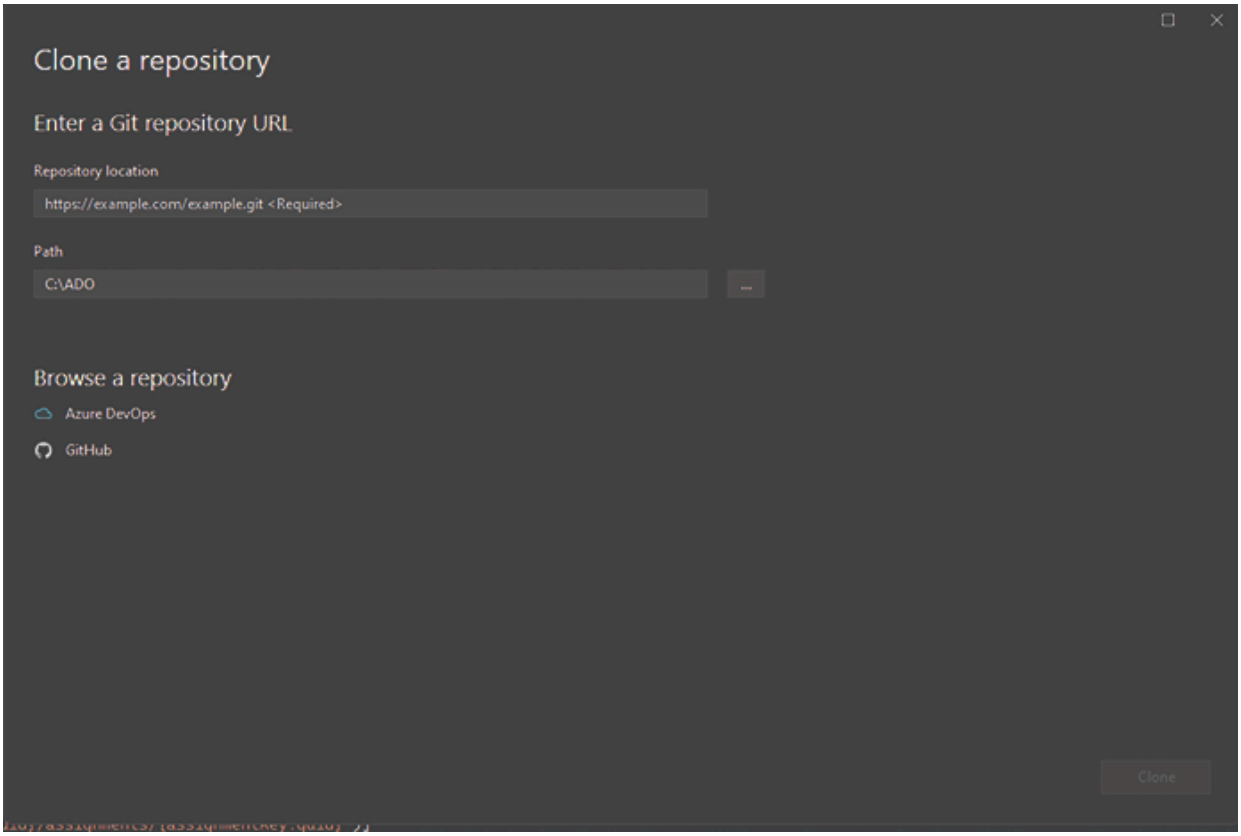


Figure 1.26: *Clone Repository*

In Visual Studio 2022, the "**`Clone Repository`**" option under the "**`File`**" menu allows you to clone an existing Git repository from a remote location, such as GitHub, Azure DevOps, or a self-hosted Git server, directly within Visual Studio.

When you select the "**`Clone Repository`**" option, Visual Studio will prompt you to provide the URL of the Git repository you want to clone and specify a local directory where the repository will be cloned to. Additionally, you can choose the branch to clone or leave it to the default branch.

Once you provide the necessary information and confirm, Visual Studio will initiate the cloning process. It will download all the files and commit history from the remote repository to your local machine, while creating a new local Git repository in the specified directory.

After the cloning process is complete, you can start working with the cloned repository in Visual Studio. You will have access to all the files and commit history, and you can use Visual Studio's built-in Git tools to manage changes, commit new code, and push/pull changes to and from the remote repository.

The "**`Clone Repository`**" option in Visual Studio simplifies the process of getting started with an existing project hosted on a remote Git repository, which will enable you to quickly bring the codebase onto your local development environment and start contributing.

`Edit` Menu

This menu provides options for editing and modifying the code in your project.

Its constituent options include:

- **`Undo`:** Reverts the most recent action performed in the active document
- **`Redo`:** Reapplies the most recently undone action in the active document
- **`Cut`:** Removes the selected text or content from the active document and places it in the clipboard
- **`Copy`:** Copies the selected text or content from the active document to the clipboard
- **`Paste`:** Inserts the content from the clipboard into the active document at the current cursor position
- **`Duplicate`:** If no selection is made, it typically duplicates the entire line where the cursor is positioned. If a selection is active, it duplicates the selected text.
- **`Delete`:** Deletes the selected text or content from the active document
- **`Find and Replace`:** Opens the Find and Replace dialog, allowing you to search for specific text within the active document and optionally replace it with different text
- **`Go To`:** Opens the Go To dialog, which allows you to navigate to a specific line number or declaration within the active document
- **`Advanced`:** This submenu often contains additional editing options such as toggling word wrap, indenting or unindenting text, and formatting

selections.

- **`Select All`:** Selects all text or content within the active document
- **`Format`:** This submenu usually contains options for formatting code, such as applying code styling rules, organizing using directives, and removing and sorting using directives.
- **`Comment/Uncomment`:** These options allow you to comment or uncomment selected lines of code, depending on the language you are working with.
- **`IntelliSense`:** This submenu provides options related to IntelliSense such as refreshing IntelliSense suggestions or displaying parameter info.
- **`Bookmarks`:** Allows you to set, navigate to, and manage bookmarks within the active document
- **`Snippets`:** Provides access to code snippets that can be inserted into the active document to speed up coding tasks
- **`Outlining`:** This submenu offers options for collapsing and expanding code blocks, which makes it easier to navigate and focus on specific sections of code.
- **`Spelling`:** Contains options for spell checking the active document

`View` Menu

In Visual Studio 2022, the "**`View`**" menu contains various options and features to customize the layout, appearance, and functionality of the IDE. Here is a detailed list of the options along with their usage and specific examples of how they can be helpful during software development:

`Solution Explorer`: This option toggles the visibility of the Solution Explorer window, which displays the structure of your solution and its projects, folders, and files. You can use it to navigate through your project hierarchy, add or remove files, and manage project settings.

In Visual Studio 2022, the "**`View`**" menu provides options related to various aspects of the development environment, including **`Git Changes`** and **`Git Repository`**. Here is a breakdown of these options and their purposes:

`Git Changes`: This option opens the Git Changes tool window, which displays changes made to files in your Git repository.

- **Purpose**: It allows you to review and manage changes to your code, including additions, modifications, and deletions, before committing them to the repository.
- **Example**: You can use Git Changes to review the modifications you've made to your codebase, stage specific changes for commit, and discard changes if needed.

`Git Repository`: This option opens the Git Repository tool window, which provides a summary of the current Git repository and its status.

- **Purpose**: It allows you to view information about the repository, such as the current branch, recent commits, untracked files, and branch history.
- **Example**: You can use the Git Repository tool window to switch between branches, view the commit history, perform Git operations like branching, merging, and pushing changes, and manage repository settings.

These options are particularly useful for developers working with Git version control within Visual Studio. They provide easy access to Git-related features and help streamline the process of managing changes and collaborating on code with a team.

`Properties Window`: The Properties window displays properties for the currently selected item in the IDE, such as a control on a form or a file in Solution Explorer. You can use it to view and modify properties, which can be particularly useful when designing user interfaces or configuring project settings.

`Output`: This window displays diagnostic messages, build output, debugging information, and other output generated by Visual Studio and your projects. It is helpful for troubleshooting errors, monitoring build progress, and viewing runtime messages during debugging sessions.

`Error List`: The Error List window displays compiler errors, warnings, and other issues detected in your code. You can use it to quickly navigate to errors, fix syntax issues, and address other problems that may prevent your code from building or running correctly.

`Task List`: The Task List window displays tasks, comments, and other annotations within your code. You can use it to keep track of TODO

comments, code review feedback, and other items that require attention during development.

Code Definition Window: This window displays the definition of the symbol at the current cursor position in your code. You can use it to quickly view the definition of a class, method, variable, or other code element without navigating away from your current location.

Document Outline: The Document Outline window provides a hierarchical view of the elements within the currently active document, such as classes, methods, and controls. You can use it to navigate through the structure of your code and quickly locate specific elements.

Toolbox: The Toolbox window contains controls and components that you can drag and drop onto your forms or design surfaces when building user interfaces. You can use it to quickly access and add common UI elements like buttons, text boxes, and labels to your projects.

Code Snippet Manager: This option opens the Code Snippet Manager, where you can manage and customize code snippets for common programming tasks. You can use it to create, edit, import, and export snippets to streamline your coding workflow and promote code reuse.

Server Explorer: Server Explorer provides access to data connections, servers, and services within your development environment. You can use it to browse databases, interact with Azure resources, and manage connected services during application development.

SQL Server Object Explorer: This window allows you to interact with SQL Server databases directly within Visual Studio. You can use it to browse database objects, execute queries, manage table data, and perform other database-related tasks without leaving the IDE.

Command Window: The Command Window in Visual Studio provides a command-line interface for executing commands, especially those related to debugging, editing, and navigation.

- It allows developers to execute commands directly without navigating through menus or toolbars.
- Developers can use the **Command Window** to perform various tasks such as executing debugging commands, setting breakpoints, navigating to specific files or lines, and interacting with the Visual Studio IDE programmatically through commands.

- Example usage of some common scenarios:
    - Executing a debugging command to set a breakpoint at a specific line of code
    - Navigating to a specific file by entering its path in the Command Window

`Immediate Window`: The **`Immediate Window`** is primarily used during debugging sessions in Visual Studio.

- It provides a way to interactively execute code statements and evaluate expressions while debugging.
- Developers can use the **`Immediate Window`** to inspect variables, evaluate expressions, call methods, and perform calculations during debugging sessions.
- It can help in debugging by allowing developers to quickly test and evaluate code snippets without modifying the source code.
- Example usage of some common scenarios:
    - Inspecting the value of a variable by typing its name and pressing Enter in the Immediate Window
    - Evaluating an expression to understand its result during debugging
    - Calling a method with specific arguments to observe its behavior at runtime

***Tip**: Both the `Command Window` and `Immediate Window` are powerful tools in Visual Studio that aid developers in debugging and interacting with their code efficiently during development sessions. They provide a command-line interface and an interactive environment for executing commands and evaluating expressions during debugging sessions, respectively.*

`Project` Menu

This menu is focused on managing and configuring your project settings and build options.

The options in this menu are:

`Add New Item`: Add a new file or item to your project.

`Add Existing Item`: Add an existing file to your project.

`Properties`: Access the project properties, where you can configure build settings, target frameworks, and other project-specific options.

`Build` Menu

This menu is used to control the build process for your project.

In Visual Studio, both "**`Build Solution`**" and "**`Rebuild Solution`**" are options used to compile the code and produce the output, but they differ in their approach and behavior.

`Build Solution:`

- When you select "**`Build Solution`**," Visual Studio compiles only those project files that have changed since the last build, along with their dependencies.
- It is a faster operation compared to "**`Rebuild Solution`**" because it only rebuilds what is necessary.
- If the project has already been built and there are no changes, selecting "**`Build Solution`**" will not rebuild anything, which results in no changes to the output files.

`Rebuild` Solution:

- When you select "**`Rebuild Solution`**," Visual Studio performs a clean build by first cleaning all output files (that is, removing all previously built files) and then compiling all project files from scratch, regardless of whether they have changed or not.
- It ensures that the entire solution is built from a clean state, which can be useful in situations where the build might be corrupted or inconsistent due to various reasons.
- "**`Rebuild Solution`**" takes more time compared to "**`Build Solution`**" because it compiles all files, regardless of whether they have changed or not.

In summary, "**`Build Solution`**" is generally faster and is used for incremental builds, compiling only what has changed since the last build. "**`Rebuild Solution`**" is slower but ensures a clean and consistent build by compiling all project files from scratch, regardless of changes. The choice

between them depends on the specific needs and circumstances of your project.

`Clean Solution`: This option is used to remove all intermediate and output files generated during the build process. When you clean a solution, Visual Studio deletes all the compiled binaries, temporary files, and other artifacts associated with the projects in the solution. Here is what the "**`Clean Solution`**" option does:

Remove Output Files:

- Deletes all compiled output files, such as executable files, DLLs, and object files, generated by the build process for each project in the solution.
- Clears the output directories where the build artifacts are stored, ensuring a clean slate for the next build.

Clears Temporary Files:

- Removes any temporary files generated during the build process, including intermediate files and compiler-generated files.
- Clears the temporary directories used by the compiler and build tools, freeing up disk space and reducing clutter.

Cleans Project Dependencies:

- Cleans any build outputs for projects within the solution.
- Removes any dll(s) and other files ensuring a clean and consistent build environment.

Resets Build State:

- Resets the build state of the solution, so the next build operation starts from scratch.
- Ensures that the next build operation recompiles all source files and regenerates all output files, regardless of whether they have changed or not.

Using the "**`Clean Solution`**" option can be helpful in various scenarios, such as:

- Resolving build errors or inconsistencies that may arise due to outdated or corrupted build artifacts
- Ensuring a clean and consistent build environment before performing a fresh build or rebuilding the solution
- Cleaning up unnecessary files and freeing up disk space by removing temporary and intermediate build artifacts

It is important to note that "**`Clean Solution`**" does not remove source code files or project settings. It only removes the output files and temporary build artifacts generated during the build process.

`Cancel`: Cancel the current build process (Keyboard shortcut: *Ctrl* + *Break*).

`Debug` Menu

The "**`Debug`**" menu provides various options and features to assist developers in debugging their applications effectively. Below is a detailed list of the options, features, and sub-menu items available in the "**`Debug`**" menu, along with specific examples of how to use them during software development:

`Start Debugging`: Launches the application in debug mode, allowing breakpoints and other debugging features to be utilized. **Example**: Launching a .NET application to debug and step through code execution

`Start Without Debugging`: Launches the application without enabling debugging features. **Example**: Launching a .NET application to run it normally without any debugging intervention

`Attach to Process`: Allows you to attach the debugger to a running process for debugging. **Example**: Attaching the debugger to an already running web server process to debug an issue

`Step Over`: It steps over the current line of code during debugging, without stepping into any function calls. **Example**: Stepping over a method call to quickly move to the next line of code in the current method.

`Step Into`: It steps into and inside the currently selected function or method during debugging. **Example**: Stepping into a method call to debug the code within that method

`Step Out`: It steps out of the current function or method during debugging, returning to the calling function. **Example**: Stepping out of a method to return to the caller and continue debugging from there

`Continue`: It resumes execution of the application until the next breakpoint is reached. **Example**: Continuing execution after fixing an issue to see if it is resolved

`Stop Debugging`: It stops the debugging session and terminates the application. **Example**: Stopping debugging after identifying and fixing all the issues

`Restart`: It terminates the current debugging session and immediately starts a new one with the debugger attached. It does not preserve the state of the previous session. **Example**: Restarting the application to test changes made during debugging and automatically reattaching the debugging session

`Break All`: It Pauses the execution of the application, allowing you to inspect the current state. **Example**: Using "**`Break All`**" to pause the application and investigate unexpected behavior

`Windows`: It provides various debugging-related windows such as "**`Breakpoints`**," "**`Call Stack`**," "**`Locals`**," and so on, to inspect the state of the application during debugging

These options and features in the "**`Debug`**" menu facilitate efficient debugging workflows, which allows developers to diagnose and fix issues in their applications effectively during software development.

`Git` Menu

This menu is focused on source control and collaboration features.

The options under this menu include:

`Clone Repository`: Connect your project to a team collaboration platform, such as Azure DevOps or GitHub.

`Commit`: Commit your changes to the source control repository

`Pull`: Pull the latest changes from the source control repository

`Push`: Push your local changes to the source control repository.

`Tools` Menu

This menu provides access to various tools and utilities within Visual Studio.

The options under this menu include:

`NuGet Package Manager`: Manage NuGet packages for your project.

`Extensions and Updates:` Explore and install additional extensions and tools for Visual Studio.

`Options:` Access the Visual Studio options and settings.

`Help` Menu

This menu offers access to the documentation, resources, and support for Visual Studio.

The options under this menu include:

`Visual Studio Help:` Access the Visual Studio documentation and user guide.

`Send Feedback:` Submit feedback or report issues with Visual Studio.

`Check for Updates:` It will check for any latest versions or updates available for currently installed VS.

`About Microsoft Visual Studio:` View information about the current version of Visual Studio.

`Search` Menu

The **`Search`** option in the Visual Studio menu bar is a powerful tool that can greatly improve your productivity and workflow. Let us dive into the details of this feature:

The **`Search`** option is located in the top-right corner of the Visual Studio IDE, next to the Help menu. This search bar provides a centralized way to quickly find and access various elements within your codebase, the Visual Studio interface, and the development tools.

Here are the key features and use cases of the Visual Studio Search option:

`Code` **Search:**

- You can use the search bar to quickly find and navigate to specific classes, methods, variables, or other code elements within your project.
- As you type in the search bar, Visual Studio will provide live suggestions and preview the matching code elements.
- This helps you quickly jump to the desired code location, which will save you time and improve your overall productivity.

`Command` Search:

- The search bar can also be used to find and execute Visual Studio commands like build, debug, or project management actions.
- By typing in a command or its description, Visual Studio will suggest you the matching commands, which will allow you to quickly access and execute them.
- This is particularly useful when you are trying to find a specific command or feature within the Visual Studio interface, but you are unsure of its exact location.

`Settings` Search:

- The search bar can be used to search for and modify Visual Studio settings and options.
- You can type in keywords related to the setting you are looking for, and Visual Studio will provide suggestions and direct you to the relevant settings page.
- This can be helpful when you need to customize the IDE's behavior or appearance to suit your preferences or project requirements.

Extension Manager:

- It can be found in **`Extensions`** menu.
- The search pr browse option can also be used to find and install Visual Studio extensions and tools from the Visual Studio Marketplace.
- By typing in keywords related to the extension you are looking for, Visual Studio will display relevant search results, which allows you to quickly discover and install the desired extensions.

Documentation Search:

- The search bar can be used to search the Visual Studio documentation and online resources.
- This can be particularly useful when you are trying to find information about a specific feature, an API, or a development best practice.

Thus, by mastering the Visual Studio Search option, you can significantly improve your development workflow, quickly navigate your codebase,

access important features and commands, and stay informed about the various tools and resources available within the Visual Studio ecosystem.

Also, by familiarizing yourself with the Visual Studio menu bar and the options available in each menu, you will be able to efficiently navigate the IDE, access the necessary tools and features, and streamline your software development process.

Conclusion

In this inaugural chapter, we explored a powerful development environment called Visual Studio 2022.

We began by delving into the rationale behind utilizing Visual Studio, while emphasizing on its significance as a comprehensive integrated development environment (IDE) that caters to a wide array of programming languages and platforms. We examined the various iterations of Visual Studio, while highlighting the evolution of this indispensable toolset over the years.

Our journey continued with a step-by-step guide to the installation process, which ensured a seamless setup experience for readers who are eager to harness the capabilities of Visual Studio 2022. Through clear instructions and visual aids, we navigated the installation procedure and empowered users to embark on their development journey with confidence.

Upon successful installation, we then ventured into the heart of Visual Studio—the IDE itself. After introducing the readers to the interface, we provided insights into the myriad options available within the menu bar and its submenus. From essential tools for project management, to customizable settings for personalization, we explored the diverse functionalities available.

As readers acquaint themselves with the intuitive layout and extensive feature set of Visual Studio 2022, they will be able to embark on a journey of exploration and innovation in the realm of software development. Armed with a foundational understanding of Visual Studio's capabilities and customization options, readers will be poised to delve deeper into the intricacies of this indispensable toolset in the chapters that follow.

With the groundwork laid in this chapter, readers will be equipped to harness the full potential of Visual Studio 2022 and embark on a transformative journey of creativity, productivity, and innovation in software development.

As we bid adieu to this introductory chapter, we look forward to accompanying readers on their journey through the Ultimate Guide to Visual Studio 2022.

The upcoming chapter will delve into project creation within VS, and will cover popular templates and project types.