

Documentación sobre el control de versiones en Visual Studio

Comparta el código desde Visual Studio mediante tecnologías de control de código fuente como Git y GitHub.

Introducción

INFORMACIÓN GENERAL

[Git y Visual Studio para el control de código fuente](#)

GUÍA PASO A PASO

[Creación de un repositorio existente](#)

[Creación de un repositorio nuevo](#)

REFERENCIA

[Comparación entre Git y Team Explorer](#)

[Cómo dominar Git y el código abierto \(blog\) ↗](#)

Flujo de trabajo de desarrollo de Innerloop

GUÍA PASO A PASO

[Creación de una rama](#)

[Realización de una confirmación \("commit"\)](#)

[Agregar al "stage" líneas de código](#)

[Envío de cambios \("push"\) a un repositorio remoto](#)

[Recuperación \("fetch"\) e incorporación \("pull"\) de cambios y sincronización \("sync"\)](#)

Administración de repositorios

GUÍA PASO A PASO

[Examinar repositorios y comparar ramas](#)

[Administrar repositorios](#)

[Trabajo con varios repositorios](#)

[Resolución de conflictos de combinación](#)

[Configuración y preferencias de Git](#)

Cómo Visual Studio facilita el control de versiones con Git

Artículo • 10/11/2022 • Tiempo de lectura: 15 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

¿Alguna vez ha deseado volver a una versión anterior de su código que funcionara? ¿Tiene que almacenar manualmente copias del código en diferentes ubicaciones como medida de seguridad? El control de versiones es la respuesta.

Git es el sistema de control de versiones moderno más usado. Con Git, puede realizar un seguimiento de los cambios de código que realice con el tiempo y revertir a versiones específicas. Así que, tanto si es un desarrollador profesional como si está aprendiendo a codificar, la experiencia de Git de Visual Studio le puede resultar muy útil.

Sugerencia

Para obtener información sobre el uso de Git y GitHub en Visual Studio, regístrese en la [serie de aprendizaje de Git](#).

Ahora Git es la experiencia de control de versiones predeterminada en **Visual Studio 2019**. Desde la [versión 16.6](#), hemos trabajado en la creación del conjunto de características y la iteración en él en función de vuestros comentarios. En la [versión 16.8](#), se convirtió en la experiencia de control de versiones predeterminada para todos los usuarios.

Nota

También seguimos compilando e iterando en la característica de Git establecida en **Visual Studio 2022**. Para obtener más información sobre las actualizaciones recientes de la característica, vea la entrada de blog [Compatibilidad de varios repositorios en Visual Studio](#).

Más información sobre Git

Git es el sistema de control de versiones moderno más usado, por lo que tanto si es un desarrollador profesional como si está aprendiendo a codificar, Git puede resultarle muy útil. Si ha comenzado a usar Git recientemente, el sitio web <https://git-scm.com/> es un

buen punto de partida. Allí encontrará hojas de referencia rápida, un libro en línea conocido y vídeos de conceptos básicos de Git.

Primeros pasos con Git en Visual Studio 2019

Le guiaremos por el uso de la nueva experiencia de Git en Visual Studio, pero si quiere realizar un recorrido rápido primero, eche un vistazo al vídeo siguiente:

Duración del vídeo: 5,27 minutos

<https://www.youtube-nocookie.com/embed/UHrAg3iKoe0> ↗

Hay tres formas de empezar a usar Git con Visual Studio para ser más productivo:

- **Creación de un repositorio Git.** Si ya tiene código que no está asociado a Git, puede empezar por crear un repositorio de Git.
- **Clonación de un repositorio de Git existente.** Si el código en el que quiere trabajar no está en la máquina, puede clonar repositorios remotos existentes.
- **Apertura de un repositorio de Git existente.** Si el código ya está en la máquina, puede abrirlo mediante **Archivo > Abrir > Proyecto o solución (o Carpeta)** y Visual Studio detectará automáticamente si tiene un repositorio de Git inicializado.

ⓘ Nota

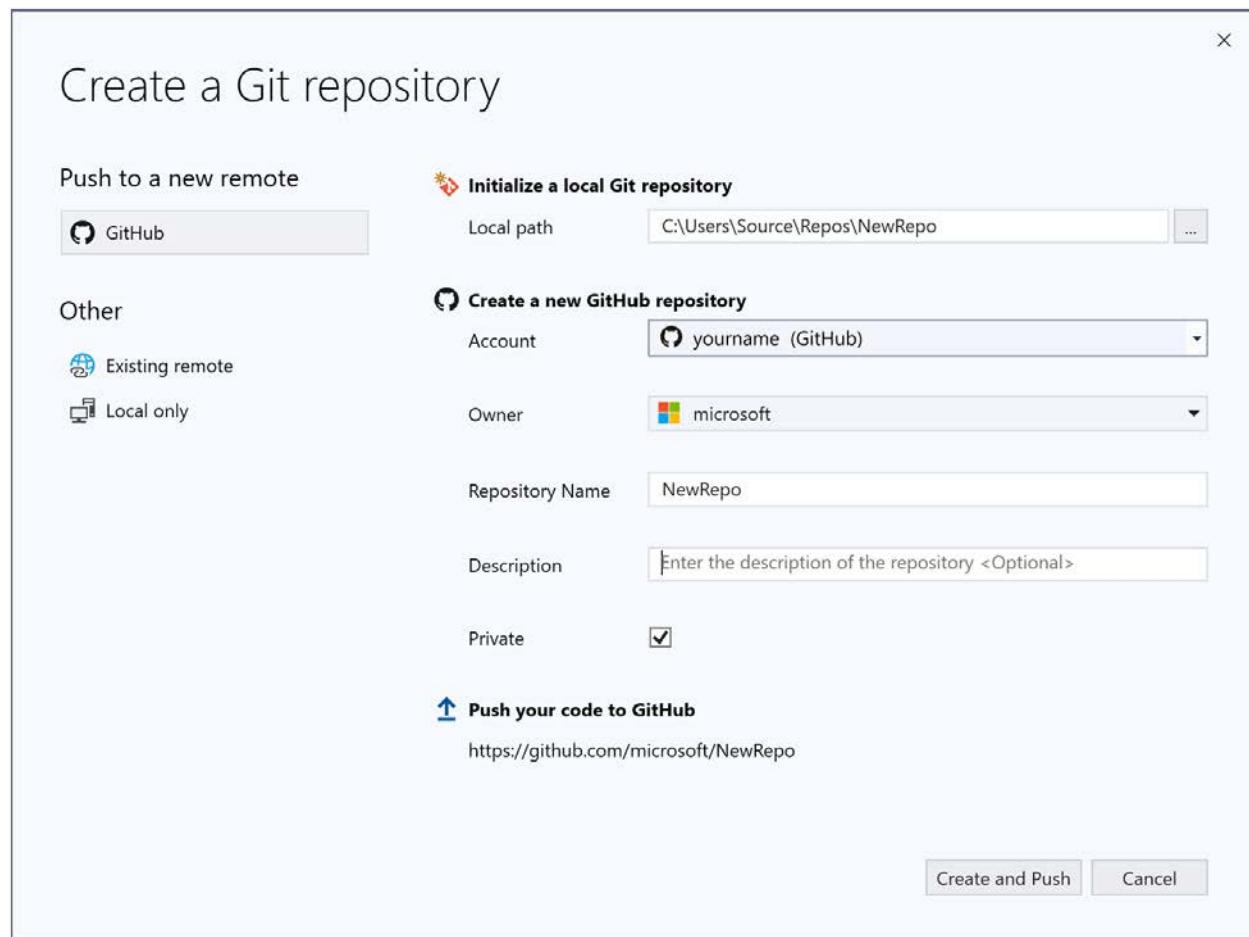
A partir de la **versión 16.8** de Visual Studio 2019, incluimos una experiencia de cuenta de GitHub totalmente integrada. Ahora, puede agregar cuentas de GitHub y GitHub Enterprise a la cadena de claves. Puede agregar estas cuentas y aprovecharlas igual que hace con las cuentas de Microsoft, lo que significa que tendrá una mayor facilidad para acceder a los recursos de GitHub en Visual Studio. Para más información, consulte la página **Trabajar con cuentas de GitHub en Visual Studio**.

💡 Sugerencia

Si no tiene una cuenta de GitHub, puede empezar siguiendo los pasos descritos en la página **Creación de una cuenta de GitHub para usarla con Visual Studio**.

Creación de un repositorio Git en Visual Studio 2019

Si el código no está asociado a Git, puede empezar por crear un repositorio de Git. Para ello, seleccione **GIT>Crear repositorio GIT** en la barra de menús. A continuación, en el cuadro de diálogo **Crear un repositorio GIT**, escriba su información.



El cuadro de diálogo **Crear un repositorio GIT** facilita la inserción del nuevo repositorio en GitHub. De forma predeterminada, el nuevo repositorio es privado, lo que significa que usted es el único que puede acceder a él. Si desactiva la casilla, el repositorio será público, lo que significa que cualquier persona en GitHub podrá verlo.

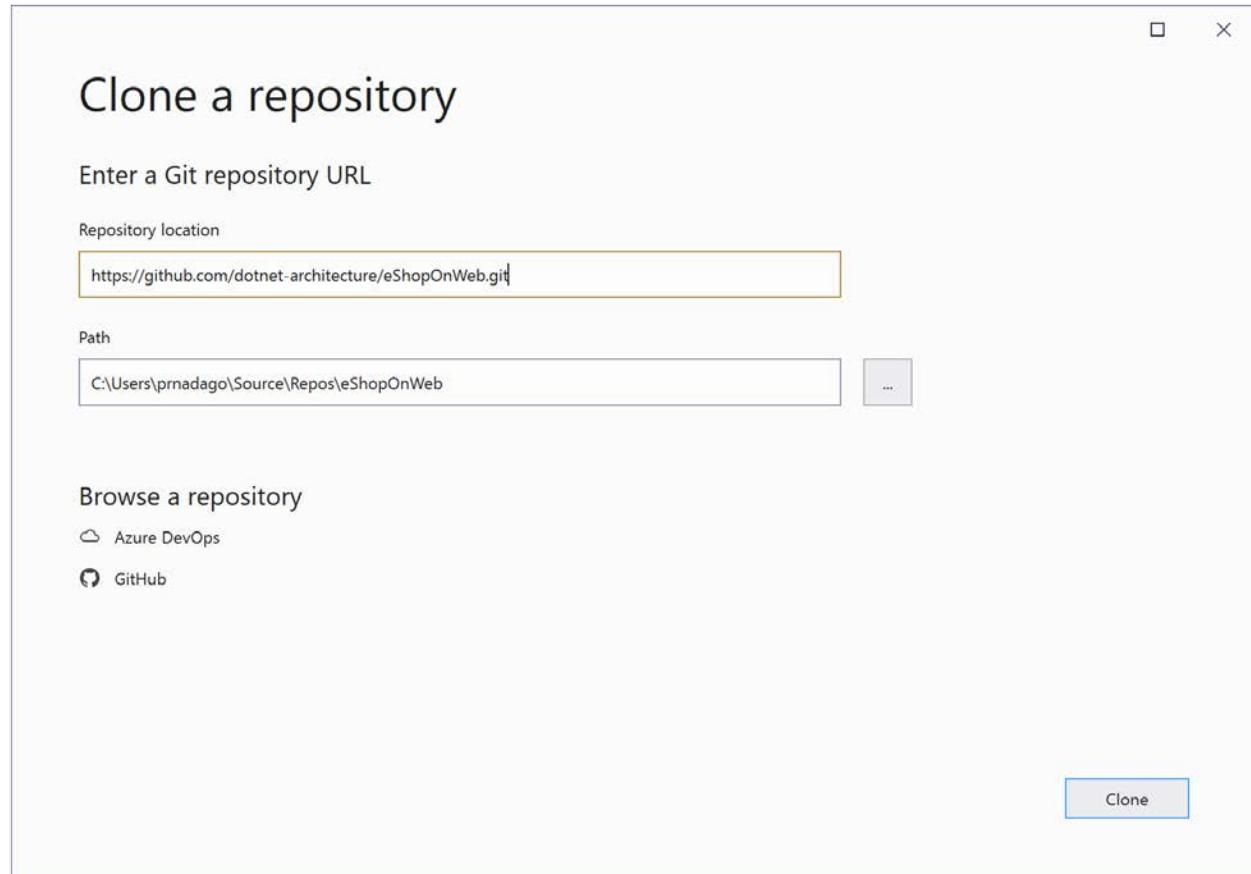
💡 Sugerencia

Tanto si el repositorio es público como privado, es mejor tener una copia de seguridad remota del código almacenada de forma segura en GitHub, aunque no esté trabajando con un equipo. Así también tendrá el código a su disposición independientemente de la máquina que use.

Puede optar por crear un repositorio de Git solo local mediante la opción **Solo locales**. También puede vincular el proyecto local a un repositorio remoto vacío existente en Azure DevOps o cualquier otro proveedor de Git mediante la opción **Repositorio remoto existente**.

Clonación de un repositorio de Git existente en Visual Studio 2019

Visual Studio incluye una experiencia de clonación sencilla. Si conoce la dirección URL del repositorio que quiere clonar, puede pegar la dirección URL en la sección **Ubicación del repositorio** y luego elegir la ubicación del disco en la que quiere que Visual Studio se clone.

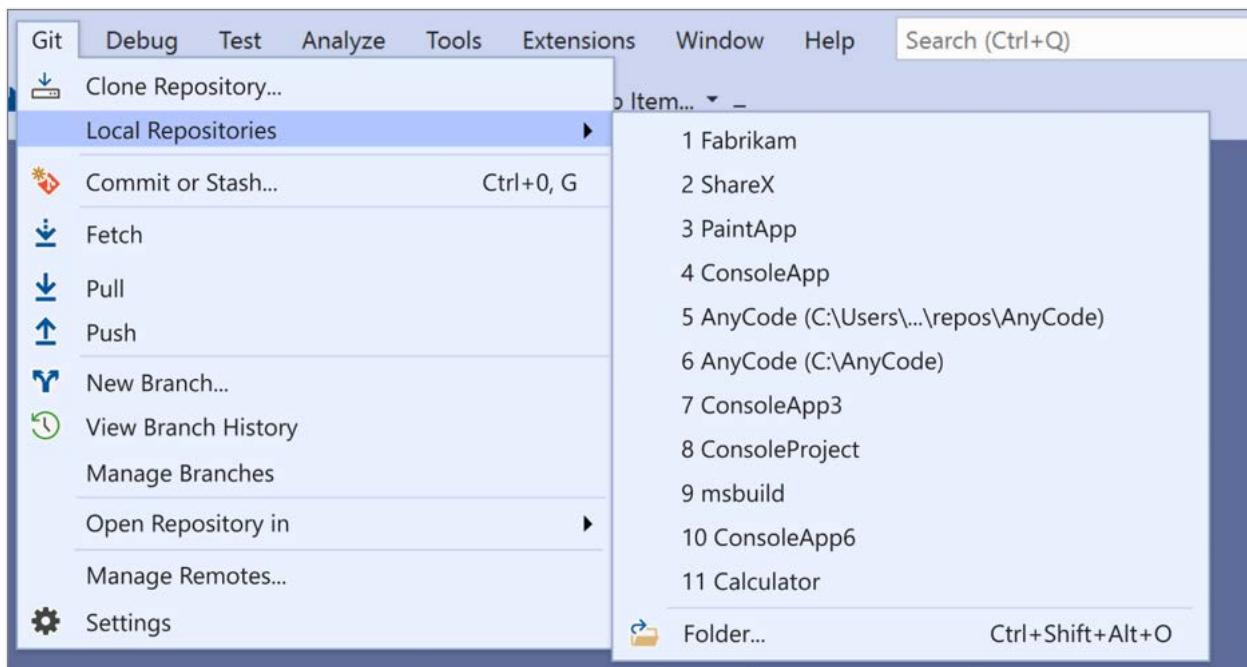


Si no conoce la dirección URL del repositorio, Visual Studio facilita la búsqueda y clonación del repositorio existente de GitHub o Azure DevOps.

Apertura de un repositorio local existente en Visual Studio 2019

Después de clonar un repositorio o de crear uno, Visual Studio detecta el repositorio de Git y lo agrega a la lista de **repositorios locales** en el menú de Git.

Desde aquí, puede acceder rápidamente a los repositorios de Git y cambiar de uno a otro.



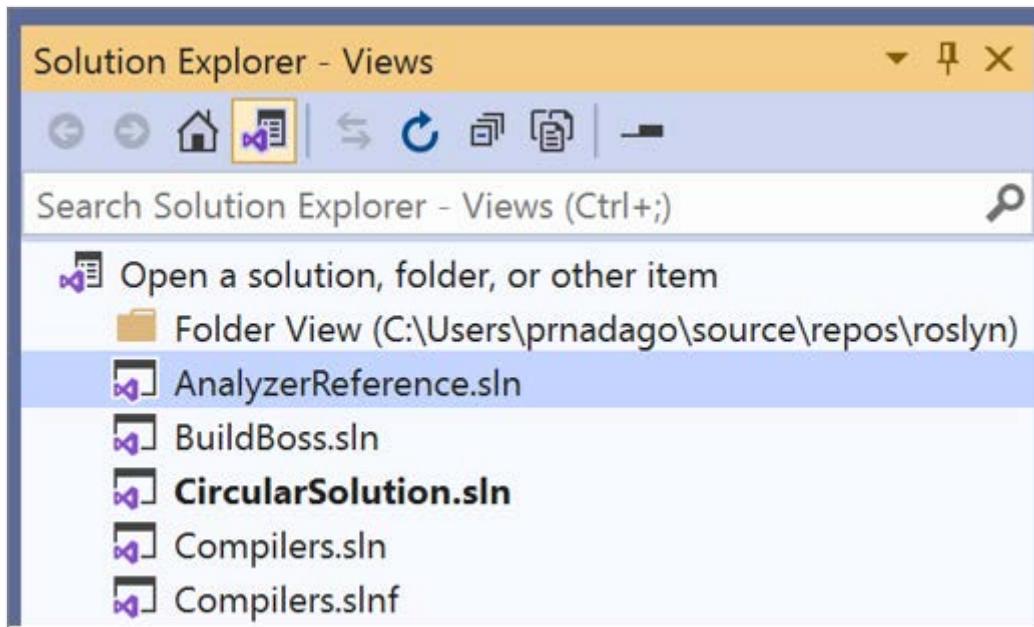
Visualización de archivos en el Explorador de soluciones de Visual Studio 2019

Al clonar un repositorio o abrir un repositorio local, Visual Studio lo lleva a ese contexto de Git al guardar y cerrar cualquier solución y proyecto abiertos previamente. El Explorador de soluciones carga la carpeta en la raíz del repositorio de Git y examina el árbol de directorios en busca de cualquier archivo visible. Estos incluyen archivos tales como CMakeLists.txt o aquellos que tienen la extensión de archivo .sln.

Visual Studio ajusta su Vista en función del archivo que se cargue en el Explorador de soluciones:

- Si clona un repositorio que contiene un solo archivo .sln, el Explorador de soluciones carga automáticamente esa solución.
- Si el Explorador de soluciones no detecta ningún archivo .sln en el repositorio, de forma predeterminada carga la Vista de carpetas.
- Si el repositorio tiene más de un archivo .sln, el Explorador de soluciones muestra la lista de Vistas disponibles para que pueda elegir una.

Puede alternar entre la Vista abierta actualmente y la lista de Vistas mediante el botón **Cambiar de vista** de la barra de herramientas del Explorador de soluciones.



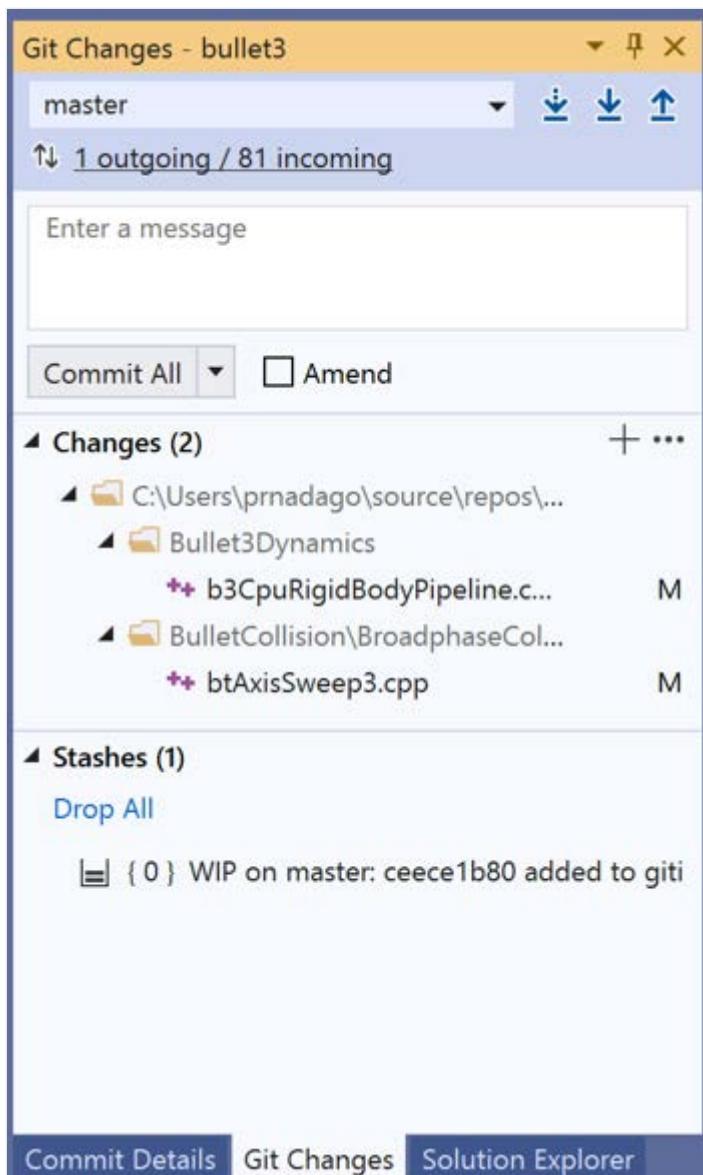
Para obtener más información, vea la sección [Vista de archivos en el Explorador de soluciones](#) del tutorial [Abrir un proyecto desde un repositorio](#).

Ventana Cambios de Git en Visual Studio 2019

Git realiza un seguimiento de los cambios de archivo en el repositorio mientras usted trabaja y separa los archivos del repositorio en tres categorías. Estos cambios son equivalentes a lo que se vería al escribir el comando `git status` en la línea de comandos:

- **Archivos sin modificar:** Estos archivos no han cambiado desde la última confirmación.
- **Archivos modificados:** Estos archivos incluyen cambios realizados desde la última confirmación, pero aún no se han almacenado provisionalmente para la siguiente confirmación.
- **Archivos almacenados provisionalmente:** Estos archivos tienen cambios que se agregarán a la siguiente confirmación.

Mientras usted realiza su trabajo, Visual Studio realiza un seguimiento de los cambios de archivo en el proyecto en la sección **Cambios** de la ventana **Cambios de Git**.



Cuando esté listo para almacenar provisionalmente los cambios, haga clic en el botón + (más) en cada archivo que quiera almacenar provisionalmente o haga clic con el botón derecho en un archivo y seleccione **Agregar al "stage"**. También puede almacenar provisionalmente todos los archivos modificados con un solo clic mediante el botón + (más) de Almacenar todo provisionalmente situado en la parte superior de la sección **Cambios**.

Al almacenar provisionalmente un cambio, Visual Studio crea una sección **Cambios almacenados provisionalmente**. Solo se agregan en la siguiente confirmación los cambios de la sección **Cambios almacenados provisionalmente**, lo que puede hacer seleccionando **Confirmar almacenados provisionalmente**. El comando equivalente para esta acción es `git commit -m "Your commit message"`. También se puede cambiar el almacenamiento provisional de los cambios haciendo clic en el botón – (menos). El comando equivalente para esta acción es `git reset <file_path>` para cambiar el almacenamiento provisional de los cambios de un único archivo o `git reset <directory_path>` para cambiar el de todos los archivos de un directorio.

También puede optar por no almacenar provisionalmente los archivos modificados omitiendo el almacenamiento provisional. En este caso, Visual Studio le permite confirmar los cambios directamente sin tener que almacenarlos provisionalmente. Solo tiene que escribir el mensaje de confirmación y luego seleccionar **Confirmar todo**. El comando equivalente para esta acción es `git commit -a`.

Visual Studio también facilita la confirmación y sincronización con un solo clic mediante los métodos abreviados **Confirmar todo e insertar** y **Confirmar todo y sincronizar**. Al hacer doble clic en cualquier archivo de las secciones sección **Cambios y Cambios almacenados provisionalmente**, puede ver una comparación línea a línea con la versión no modificada del archivo.

```
b3CpuRigidBodyPipeline.cpp | Git Repository - bullet3
eglRendererVisualShapeConverter.cpp;391cf6bd
916    };
917
918    int·EGLRendererVisualShapeConverter::registerSh
919    {
920        → int·uniqueId =·orgGraphicsUniqueId;
921        → float·rgbaColor[4] = {·visualShape.m_rgbaCo
922
923        → EGLRendererObjectArray**·visualsPtr =·m_da
924        → if·(visualsPtr == 0)
925        {
926            →     m_data->m_swRenderInstances.insert(unic
927        }
928        → visualsPtr =·m_data->m_swRenderInstances[ur
929        → if·(visualsPtr)
930        {
931            →     EGLRendererObjectArray*·visuals =·*visu
932
933            →     ///////////////
934
935            →     B3_PROFILE("m_instancingRenderer·regis
936
937            →     //·register·mesh·to·m_instancingRender
```

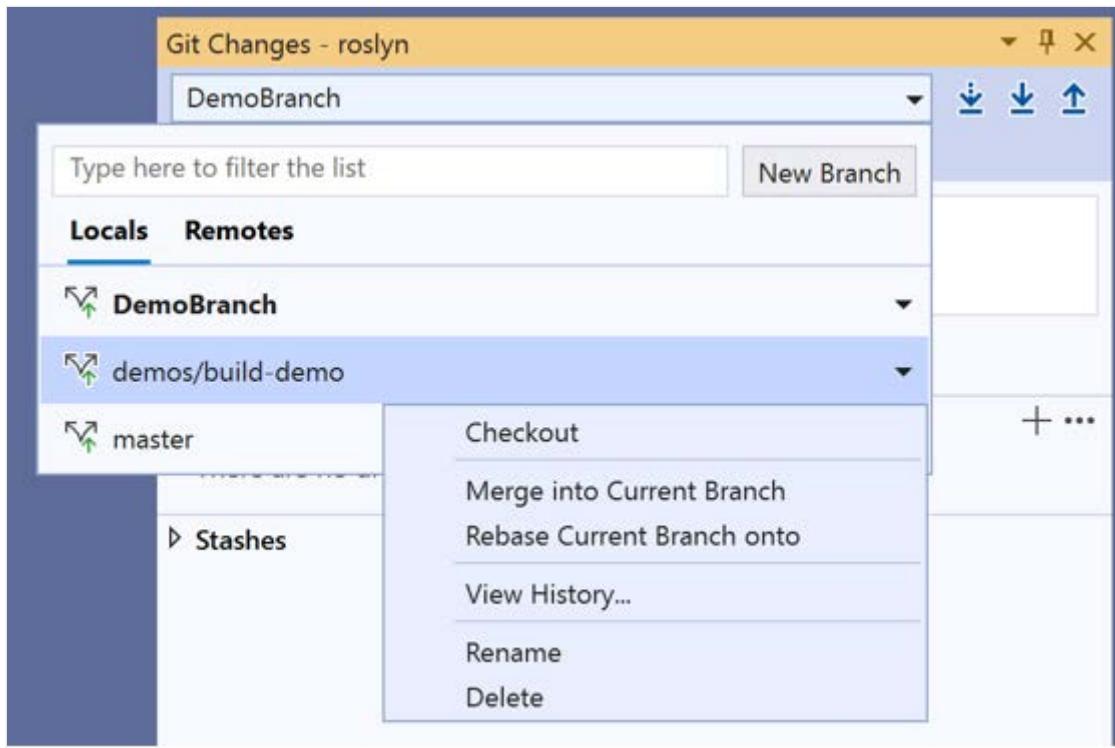
```
Diff - eglRendererVisualShapeConverter.cpp;e9f486ee
eglRendererVisualShapeConverter.cpp;e9f486ee
916    };
917
918    int·EGLRendererVisualShapeConverter::registerSh
919    {
920        → int·uniqueId =·orgGraphicsUniqueId;
921        → float·rgbaColor[4] = {·(float)visualShape
922
923        → EGLRendererObjectArray**·visualsPtr =·m_da
924        → if·(visualsPtr == 0)
925        {
926            →     m_data->m_swRenderInstances.insert(unic
927        }
928        → visualsPtr =·m_data->m_swRenderInstances[ur
929        → if·(visualsPtr)
930        {
931            →     EGLRendererObjectArray*·visuals =·*visu
932
933            →     ///////////////
934
935            →     B3_PROFILE("m_instancingRenderer·regis
936
937            →     //·register·mesh·to·m_instancingRender
```

💡 Sugerencia

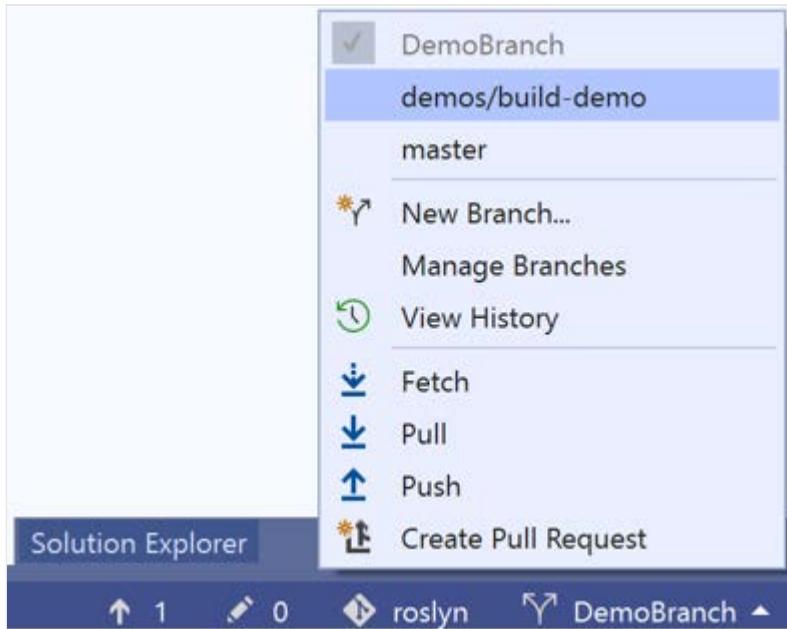
Puede asociar un elemento de trabajo de Azure DevOps con una confirmación mediante el carácter "#" si está conectado al repositorio de Azure DevOps. Puede conectar el repositorio de Azure DevOps mediante **Team Explorer>Administrar conexiones**.

Selección de una rama existente en Visual Studio 2019

Visual Studio muestra la rama actual en el selector situado en la parte superior de la ventana **Cambios de Git**.



La rama actual también está disponible en la barra de estado en la esquina inferior derecha del IDE de Visual Studio.

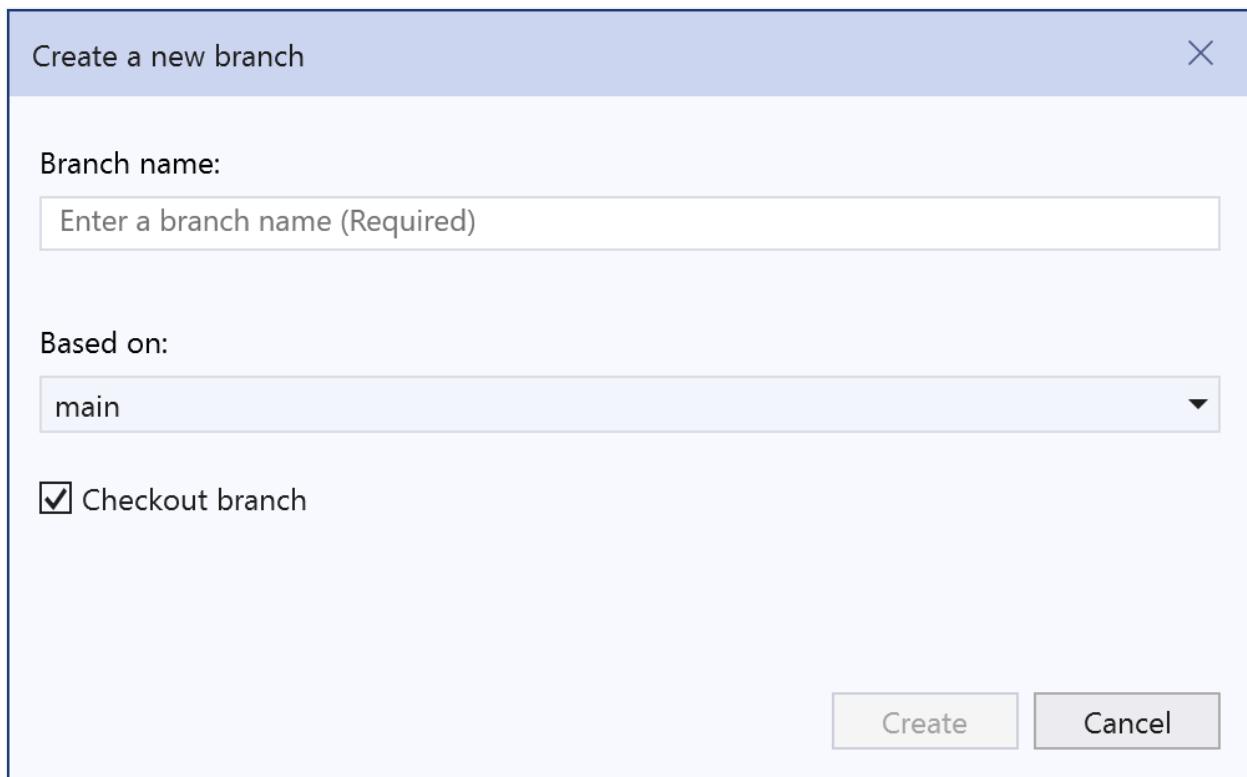


En ambas ubicaciones, puede alternar entre las ramas existentes.

Creación de una rama en Visual Studio 2019

También puede crear una rama. El comando equivalente para esta acción es `git checkout -b <branchname>`.

La creación de una rama es tan sencillo como escribir el nombre de la rama y basarla en una rama existente.



Puede elegir una rama local o remota existente como base. La casilla **Desproteger rama** le lleva automáticamente a la rama recién creada. El comando equivalente para esta acción es `git checkout -b <new-branch><existing-branch>`.

Ventana Repositorio de GIT en Visual Studio 2019

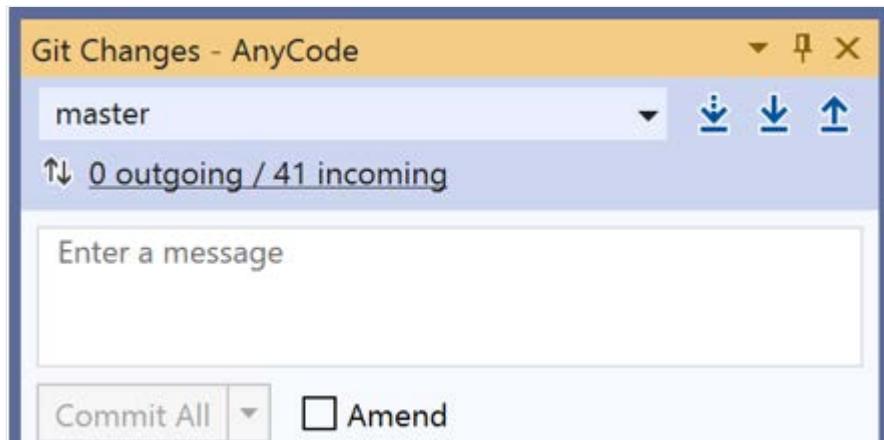
Visual Studio tiene una nueva ventana **Repositorio de GIT**, que es una vista consolidada de todos los detalles del repositorio, incluidas todas las ramas, los elementos remotos y los históricos de confirmación. Puede acceder a esta ventana directamente desde **Git** o **Vista** en la barra de menús o desde la barra de estado.

Administración de ramas en Visual Studio 2019

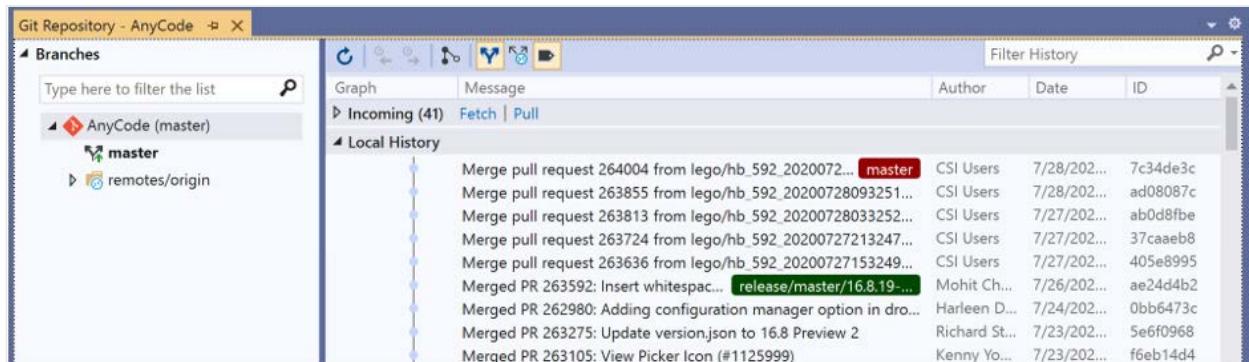
Al seleccionar **Administrar ramas** en el menú **Git**, verá la vista de árbol Ramas en la ventana **Repositorio de GIT**. En el panel izquierdo, puede usar el menú contextual para desproteger ramas, crear ramas, combinar, fusionar mediante cambio de base, selección exclusiva, etc. Al hacer clic en la rama, puede ver una vista previa del histórico de confirmaciones en el panel derecho.

Confirmaciones entrantes y salientes en Visual Studio 2019

Al capturar una rama, la ventana **Cambios de Git** presenta un indicador en la lista desplegable Rama, que muestra el número de confirmaciones no extraídas de la rama remota. Este indicador también muestra el número de confirmaciones locales sin insertar.



El indicador también funciona como un vínculo que le lleva al historial de confirmaciones de esa rama en la ventana **Repositorio de GIT**. En la parte superior del historial se muestran los detalles de estas confirmaciones entrantes y salientes. Desde aquí, también puede optar por extraer o insertar las confirmaciones.



Detalles de confirmación en Visual Studio 2019

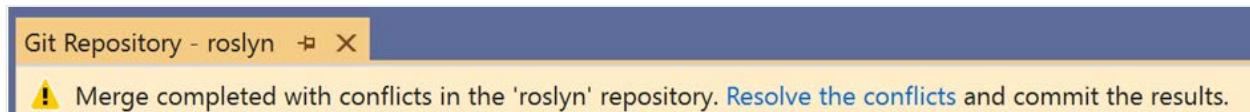
Al hacer doble clic en una **Confirmación**, Visual Studio abre los detalles en una ventana de herramientas independiente. Aquí puede revertir la confirmación, restablecer la confirmación, modificar el mensaje de confirmación o crear una etiqueta en la confirmación. Al hacer clic en un archivo cambiado en la confirmación, Visual Studio abre la vista en paralelo **Diferencias** de la confirmación y su elemento primario.

The screenshot shows the 'Commit Details' window for a commit named 'd05f43ee'. The commit was made on 7/23/2020 at 10:44:31 AM by a user represented by a green 'SG' icon. The parent commit is 'fb25acd2'. A note indicates that the commit merged PR 262895: 'Add event notification of refresh command execution.' Related work items include '#1160780'. Action buttons at the top right include 'Revert', 'Reset', 'Create Tag', and 'Actions'. Below these, sections for 'Related Work Items' (listing '#1160780') and 'Changes (10)' (listing a folder 'src') are shown.

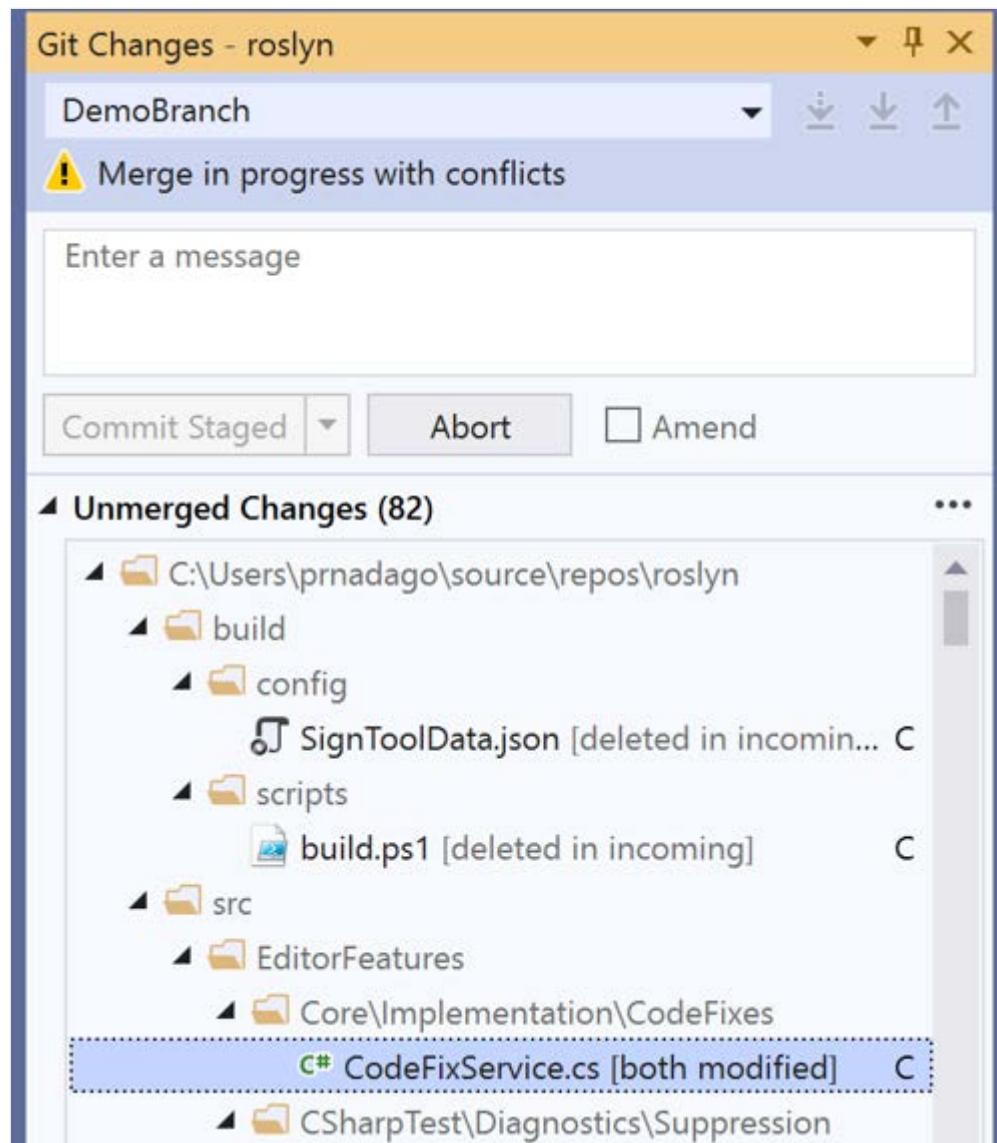
Control de conflictos de combinación en Visual Studio 2019

Pueden producirse conflictos durante una combinación si dos desarrolladores modifican las mismas líneas en un archivo y Git no sabe automáticamente cuál es la correcta. Git detiene la fusión mediante combinación e informa de que está en un estado Con conflictos.

Visual Studio hace que sea fácil identificar y resolver un conflicto de fusión mediante combinación. En primer lugar, la ventana **Repository de GIT** muestra una barra de información dorada en la parte superior de la ventana.



La ventana **Cambios de Git** también muestra el mensaje "*Fusión mediante combinación en curso con conflictos*", con los archivos sin combinar en su sección independiente debajo.

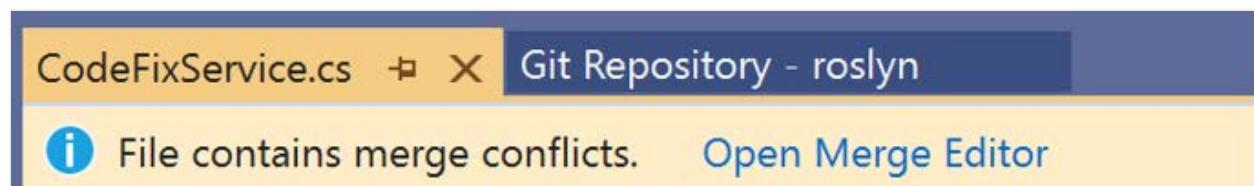


Pero si ninguna de estas ventanas se abre y, en su lugar, se va al archivo que tiene conflictos de combinación, no tendrá que buscar el texto siguiente:

```
Bash

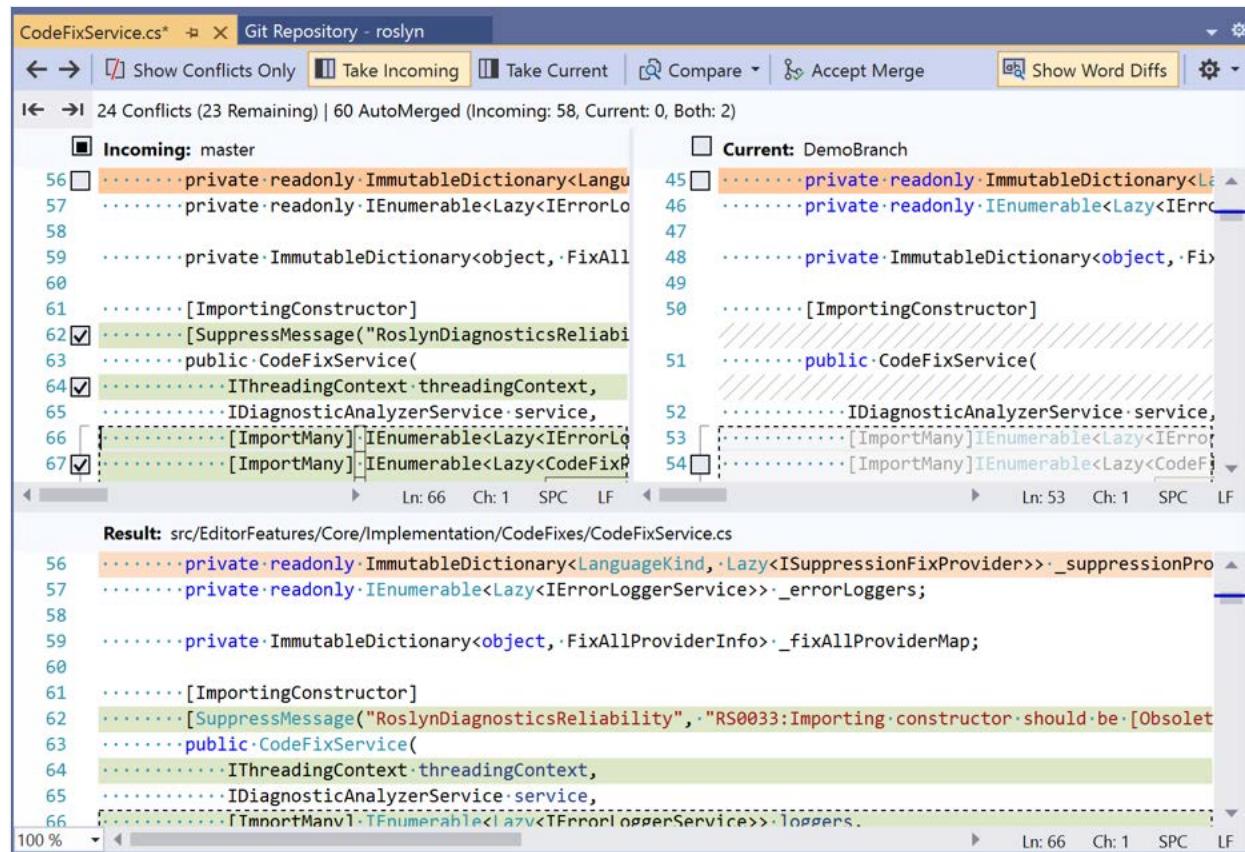
<<<<< HEAD
=====
>>>>> main
```

En su lugar, Visual Studio muestra una barra de información dorada en la parte superior de la página que indica que el archivo abierto tiene conflictos. A continuación, puede hacer clic en el vínculo para abrir el **editor de combinación**.



Editor de combinación en Visual Studio 2019

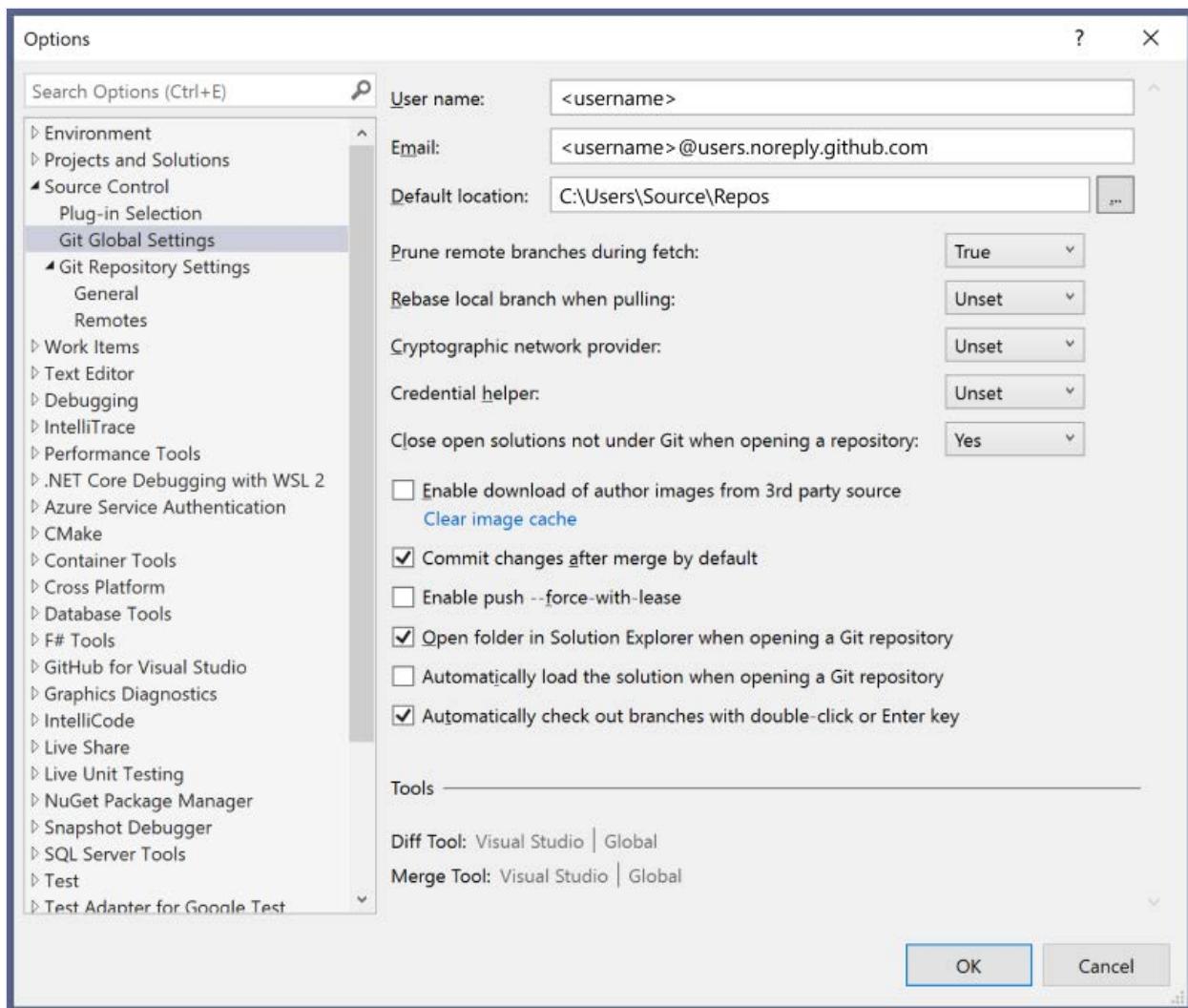
El editor de combinación de Visual Studio es una herramienta de combinación tridireccional que muestra los cambios entrantes, los cambios actuales y el resultado de la combinación. Puede usar la barra de herramientas situada en el nivel superior del **editor de combinación** para desplazarse entre los conflictos y las diferencias de combinación automática en el archivo.



También puede usar los controles de alternancia para mostrar u ocultar las diferencias, mostrar u ocultar las diferencias de palabras y personalizar el diseño. Hay casillas en la parte superior de cada lado que puede usar para realizar todos los cambios de un lado o del otro. No obstante, para realizar cambios individuales, puede hacer clic en las casillas situadas a la izquierda de las líneas en conflicto en cualquier lado. Por último, cuando termine de resolver los conflictos, puede seleccionar el botón **Aceptar combinación** del editor de combinación. Escriba luego un mensaje de confirmación y confirme los cambios para completar la resolución.

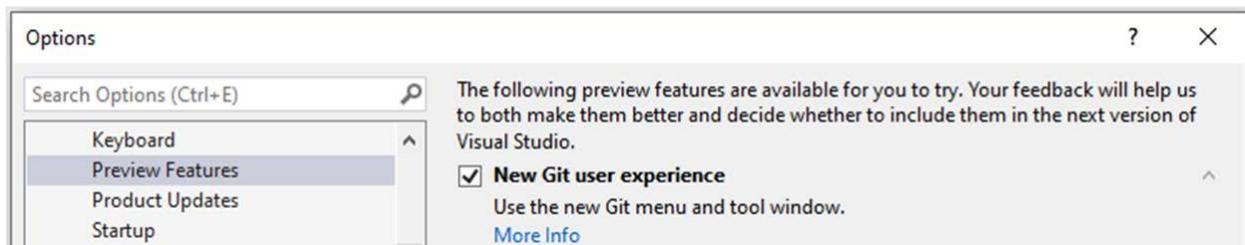
Personalización de la configuración de Git en Visual Studio 2019

Para personalizar la configuración de Git en un nivel de repositorio, así como en un nivel global, vaya a **Git>Configuración** o a **Herramientas>Opciones>Control de código fuente** en la barra de menús. Después, elija las **opciones** que quiera.



Cómo usar la experiencia de Team Explorer heredada en Visual Studio 2019

La nueva experiencia de Git es el sistema de control de versiones predeterminado en Visual Studio 2019 a partir de la [versión 16.8](#). Sin embargo, si desea desactivarla, puede hacerlo. Vaya a **Herramientas > Opciones > Entorno > Características en versión preliminar** y active la casilla **New Git User Experience** (Nueva experiencia de usuario de Git), que le cambiará a la experiencia de Team Explorer para Git.



Clonación de un repositorio de Git en Visual Studio

Artículo • 27/09/2022 • Tiempo de lectura: 2 minutos

Se aplica a: Visual Studio para Mac Visual Studio Code

Visual Studio facilita la clonación de un repositorio directamente desde el IDE. Puede trabajar de forma remota con el proveedor de Git que prefiera, como GitHub o Azure DevOps.

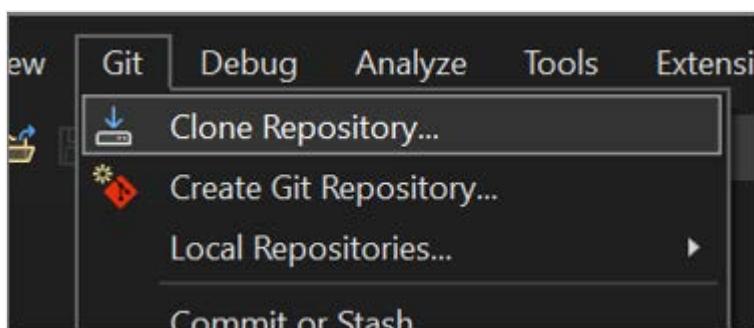
Prerrequisitos

Para seguir este artículo, necesitará lo siguiente:

- [Visual Studio instalado](#)
- [Una cuenta de usuario de GitHub](#)

Clonación de un repositorio de GitHub e inicio de sesión

1. Abra Visual Studio.
2. En el menú **git**, seleccione **Clonar repositorio**.



Nota

Si no ha interactuado con el menú **de Git** antes, es posible que vea **Clonar** en lugar de **Clonar repositorio**. Si es así, seleccione **Clonar**.

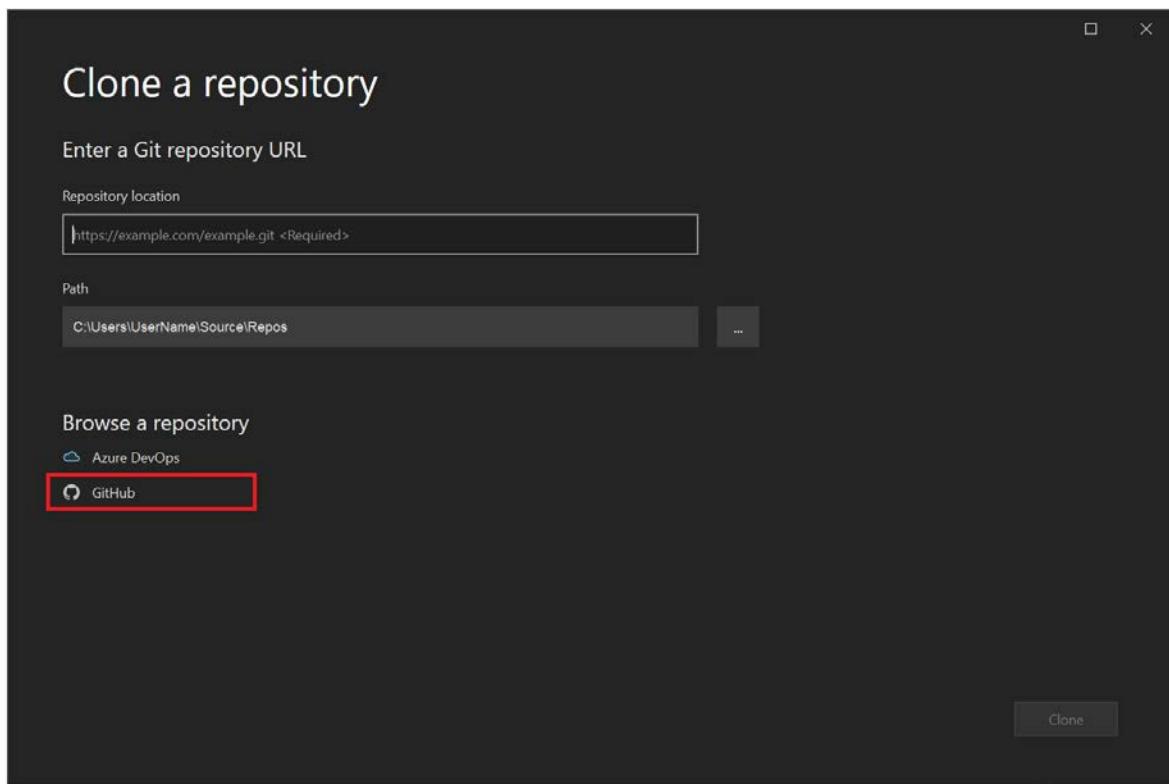
Y, si **Git** no está en la barra de menús, vaya a **Herramientas > Opciones > Selección del complemento de control >** de código fuente y, a continuación,

seleccione **Git** en la lista desplegable Complemento de control de código fuente actual.

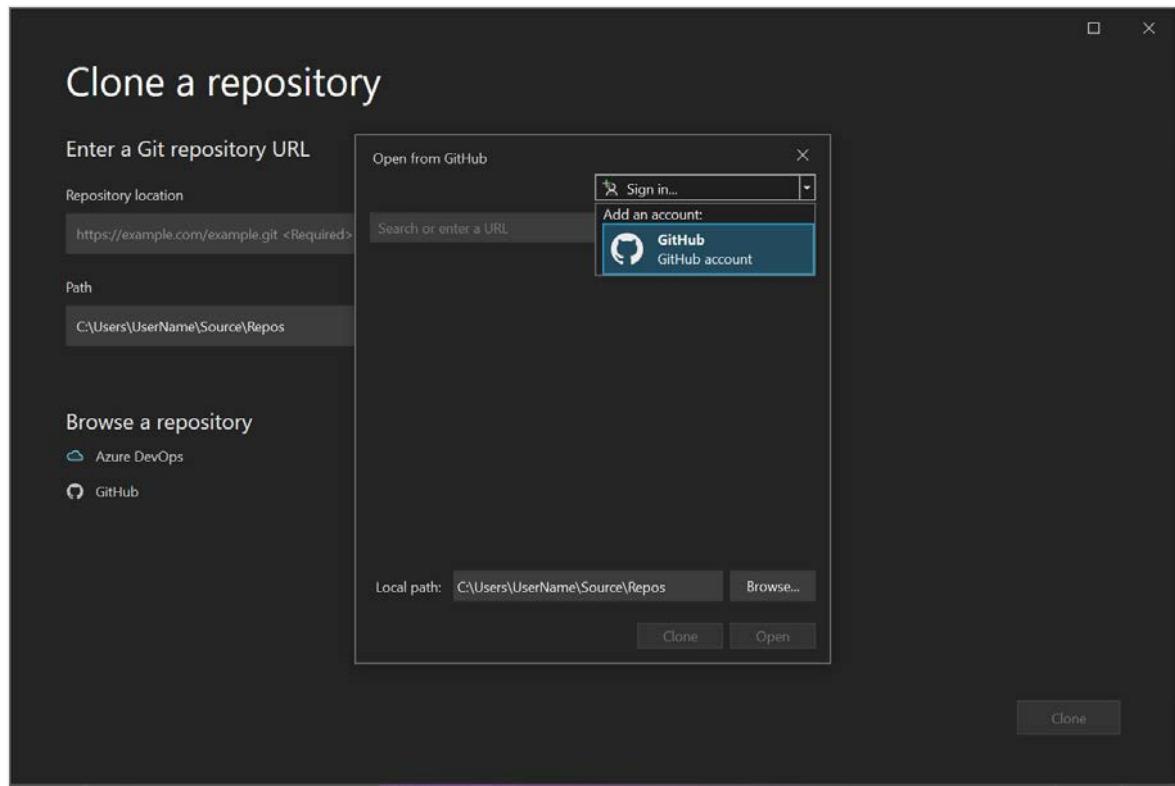
3. En la ventana **Clonar un repositorio** , en la sección **Escriba una dirección URL del repositorio de Git** , agregue la información del repositorio en el cuadro **Ubicación del repositorio** .

A continuación, en la sección **Ruta** de acceso, puede elegir aceptar la ruta de acceso predeterminada a los archivos de origen locales, o bien puede ir a otra ubicación.

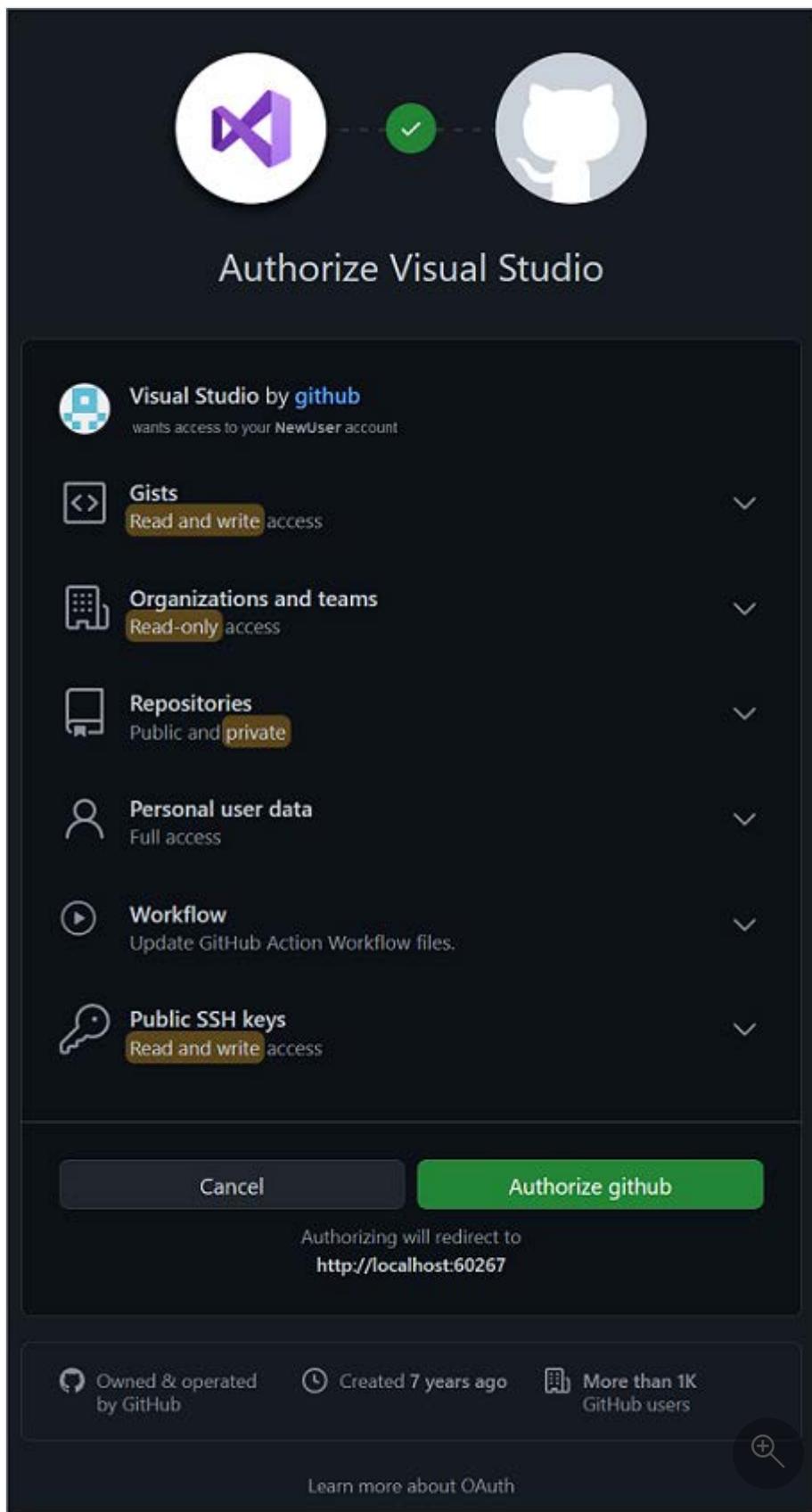
A continuación, en la sección **Examinar un repositorio** , seleccione **GitHub**.



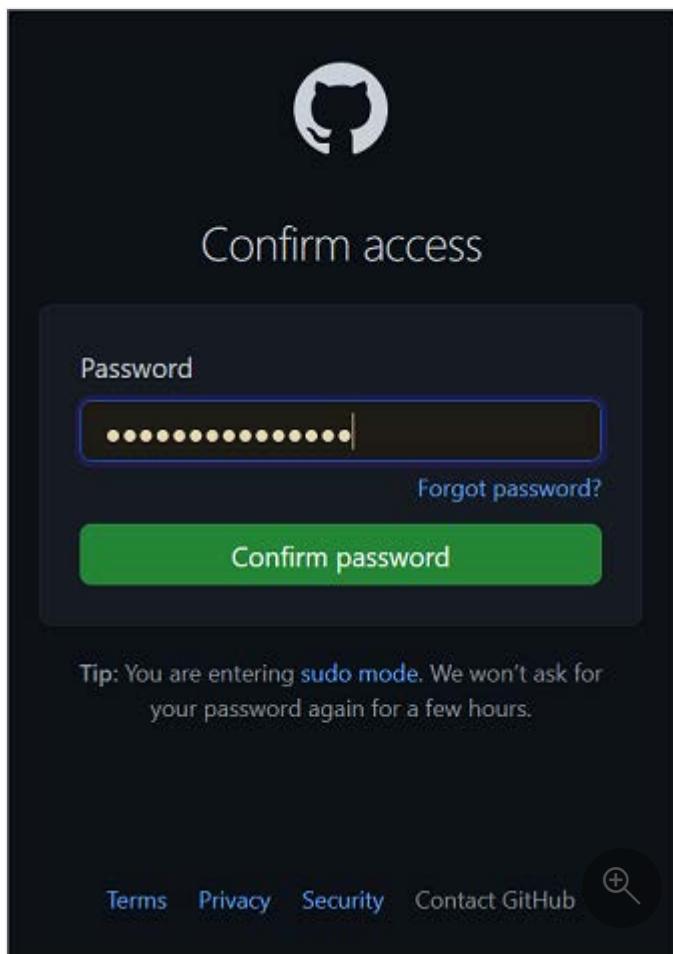
4. En la ventana **Abrir desde GitHub** , puede comprobar la información de la cuenta de GitHub o agregarla. Para ello, seleccione **Iniciar sesión** en el menú desplegable.



Si inicia sesión en GitHub desde Visual Studio por primera vez, aparece un aviso **Autorizar Visual Studio**. Elija las opciones que quiera y, después, seleccione **Autorizar github**.



A continuación, verá una ventana de confirmación de autorización. Escriba la contraseña y, a continuación, seleccione **Confirmar contraseña**.

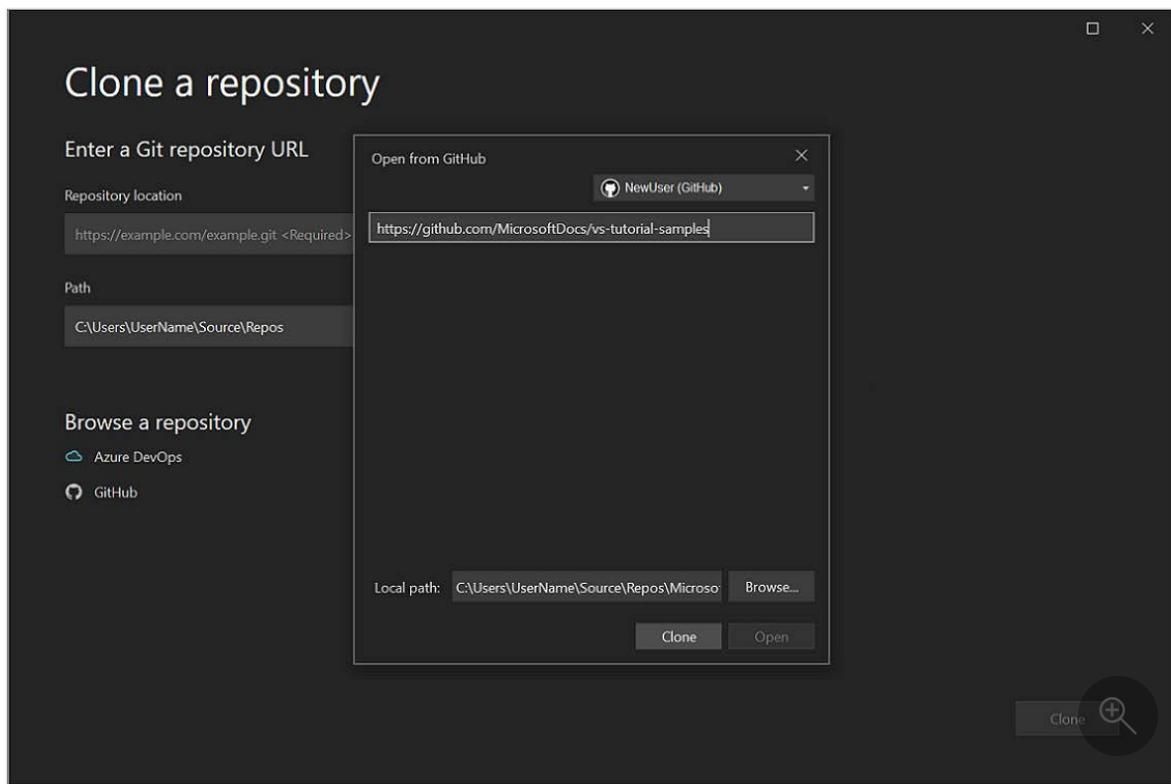


Después de vincular la cuenta de GitHub con Visual Studio, aparece una notificación Correcto.

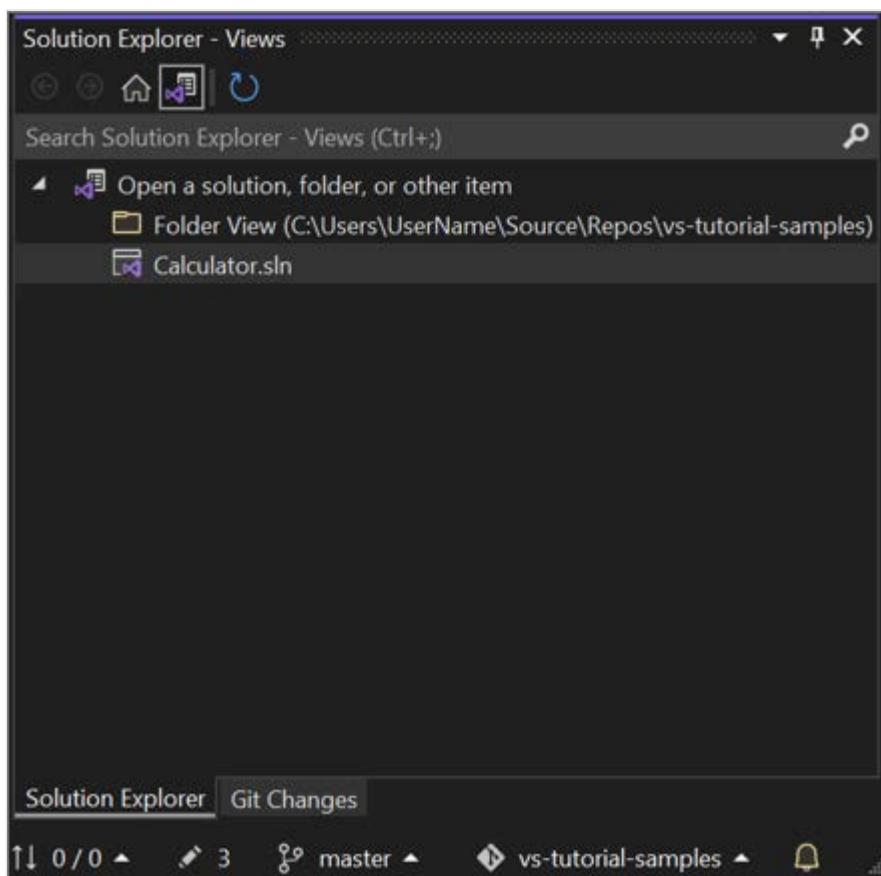


5. Despues de iniciar sesión, Visual Studio vuelve al cuadro de diálogo **Clonar un repositorio** , donde la ventana **Abrir desde GitHub** muestra todos los repositorios a los que tiene acceso. Seleccione la que desee y, a continuación, **seleccione Clonar**.

Si no aparece una lista de repositorios, escriba la ubicación del repositorio y seleccione **Clonar**.



6. A continuación, Visual Studio presenta una lista de soluciones en el repositorio. Elija la solución que desea cargar o abrir la **Vista de carpetas** en el [Explorador de soluciones](#).



Sugerencia

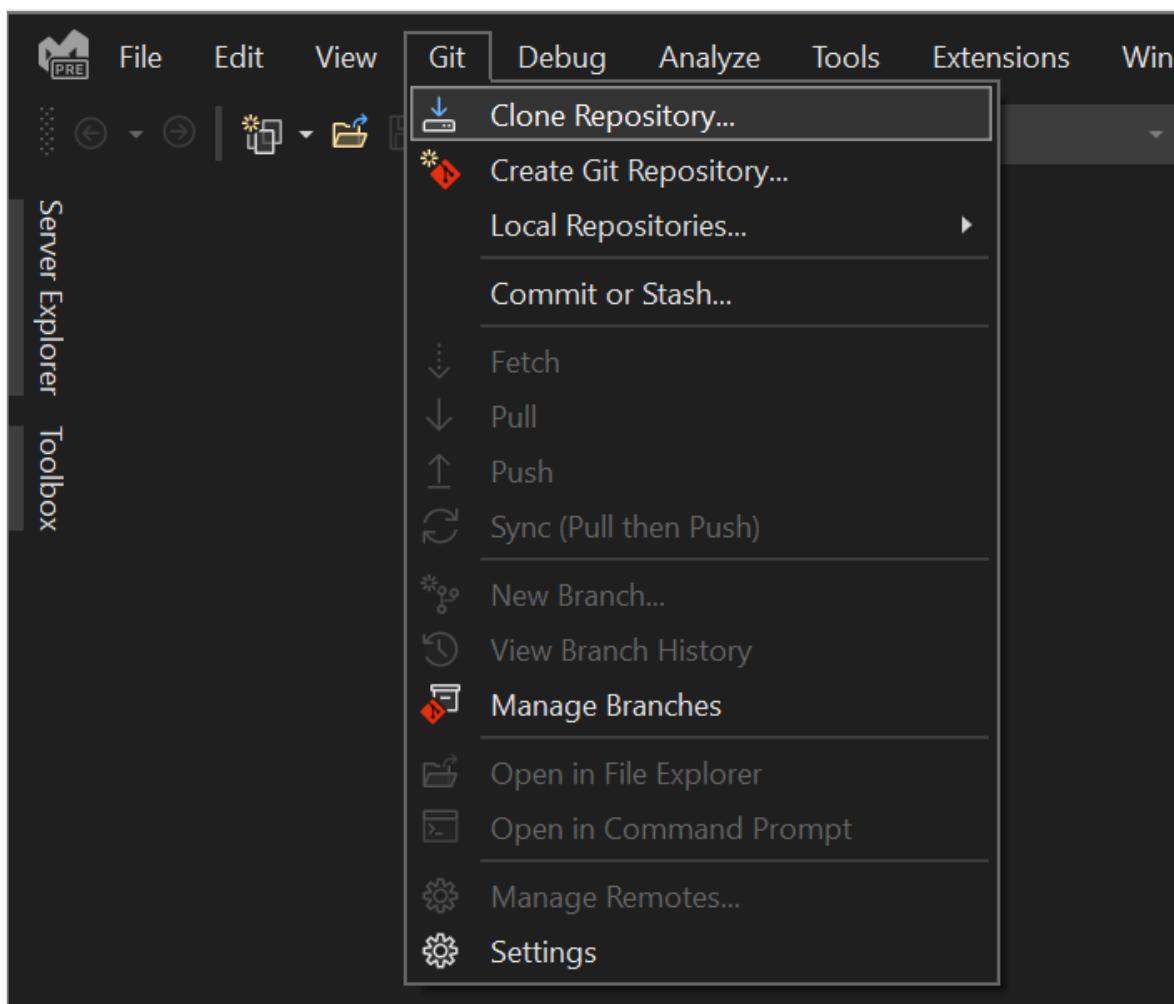
Puede cambiar la vista de carpetas predeterminada a vista de solución en el menú Git . Para ello, seleccione Configuración>Control de código fuente>Configuración global de Git>Cargar automáticamente la solución al abrir un repositorio GIT.

Apertura de un repositorio de Git existente

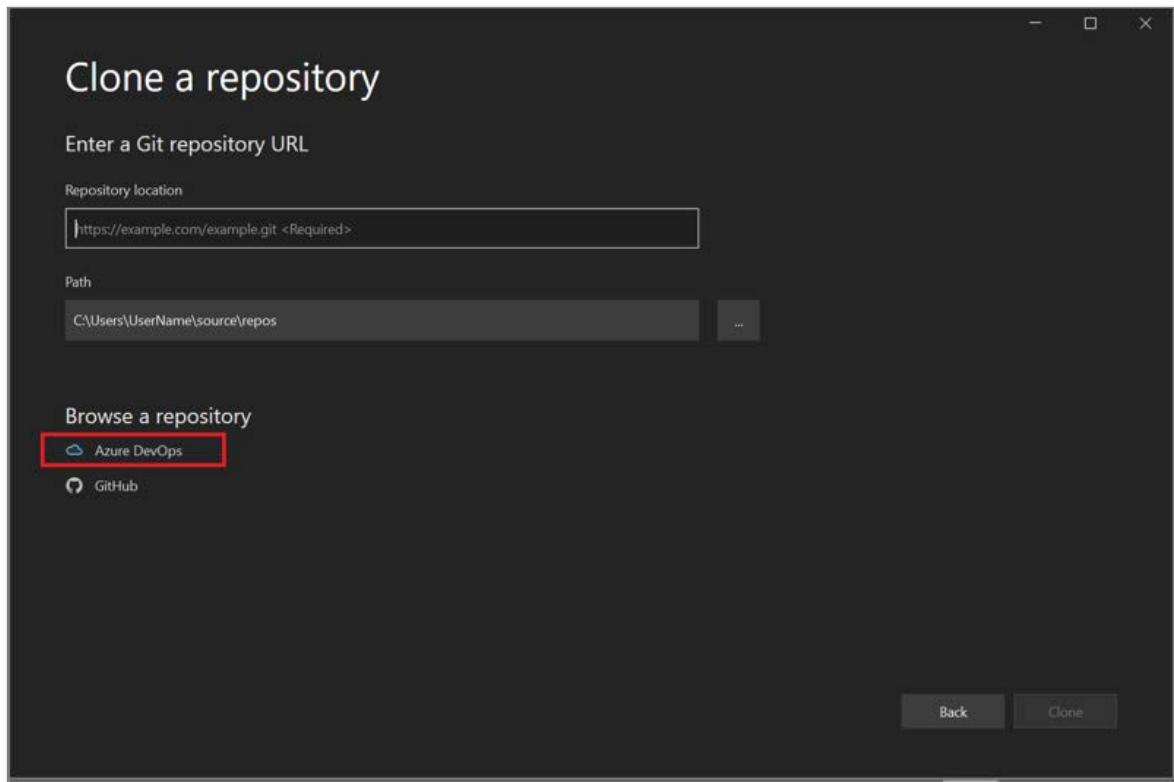
Después de clonar un repositorio o [crear uno](#), Visual Studio detecta el repositorio de Git y lo agrega a la lista de **repositorios locales** en el menú Git. Desde allí, puede acceder y cambiar rápidamente entre los repositorios de Git.

Vaya a y clone un repositorio de Azure DevOps.

1. Abra Visual Studio.
2. En el menú git , seleccione Clonar repositorio.



3. En la sección Examinar un repositorio del cuadro de diálogo Clonar un repositorio , seleccione Azure DevOps.



4. Aparece un cuadro de diálogo **Conectar a un proyecto**. Siga las indicaciones para iniciar sesión en su cuenta de Azure y, después, vaya a Azure DevOps Server que hospeda los archivos que busca.

Pasos siguientes

Para continuar con el recorrido, visite la página [Crear un repositorio](#).

Creación de un repositorio Git en Visual Studio

Artículo • 21/03/2023 • Tiempo de lectura: 2 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

Visual Studio facilita la creación de un repositorio directamente desde el IDE. La creación de repositorios desde Visual Studio está optimizada para GitHub, pero puede trabajar de forma remota con el proveedor de Git que prefiera. Esta es la manera de hacerlo.

Requisitos previos

Para seguir este artículo, necesitará lo siguiente:

- [Visual Studio instalado](#)
- [Cuenta de usuario de GitHub](#)

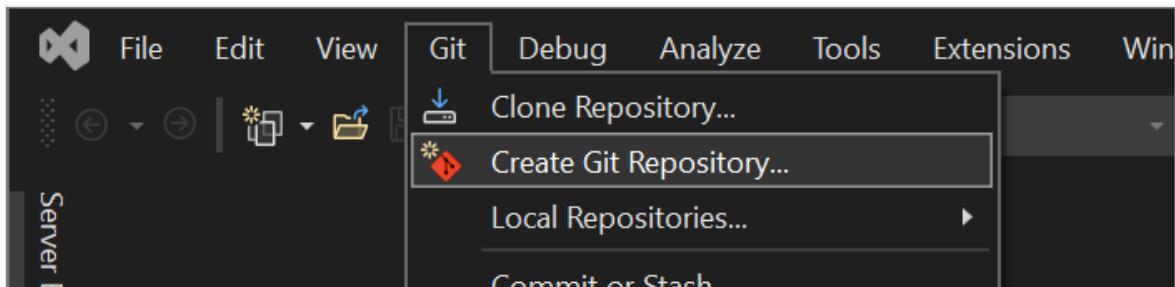
Creación de un repositorio de GitHub

1. Abra Visual Studio y seleccione **Crear un proyecto**.

Sugerencia

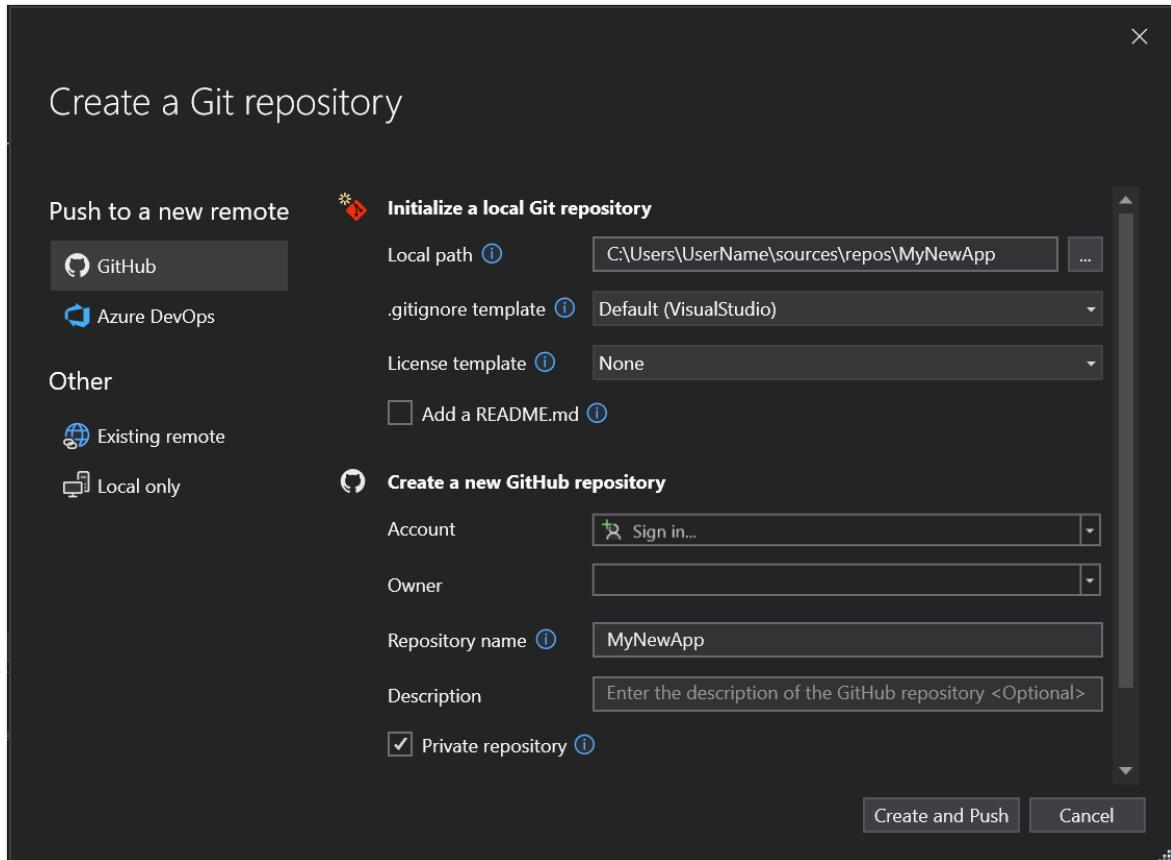
Si aún no tiene un proyecto en Visual Studio para agregarlo a un repositorio, puede **crear rápidamente una nueva aplicación de consola de C#** y asignarle el nombre **MyNewApp**. Visual Studio rellena la nueva aplicación con el código predeterminado "*Hello, World!*".

2. En el menú **Git**, seleccione **Crear un repositorio GIT**.



3. En el cuadro de diálogo **Crear un repositorio GIT**, en la sección **Enviar cambios a un nuevo repositorio remoto**, elija **GitHub**.

4. En la sección **Crear un nuevo repositorio de GitHub** del cuadro de diálogo **Crear un repositorio GIT**, escriba el nombre del repositorio que desea crear. (Si aún no ha iniciado sesión en su cuenta de GitHub, también puede hacerlo desde esta pantalla).



En **Iniciar un repositorio de Git local**, puede usar la opción de **plantilla .gitignore** para especificar los archivos no seguidos intencionadamente que quiera que Git omita. Para obtener más información sobre .gitignore, consulte [Omisión de archivos](#). Y para obtener más información sobre las licencias, consulte [Licencias de un repositorio](#).

Sugerencia

Puede actualizar y cambiar esta configuración siempre que quiera. Para obtener instrucciones detalladas, consulte [Configuración de Git en Visual Studio](#).

5. Después de iniciar sesión y escribir la información del repositorio, seleccione el botón **Crear y enviar los cambios** para crear el repositorio y agregar la aplicación.

Apertura de un repositorio de Git existente

Después de crear un repositorio o [clonar uno](#), Visual Studio detecta el repositorio de Git y lo agrega a la lista de **repositorios locales** en el menú de Git. Desde aquí, puede acceder rápidamente a los repositorios de Git y cambiar de uno a otro.

Creación de un repositorio de Azure DevOps

1. Abra Visual Studio y seleccione **Crear un proyecto**.

Sugerencia

Si aún no tiene un proyecto en Visual Studio para agregarlo a un repositorio, puede **crear rápidamente una nueva aplicación de consola de C#** y asignarle el nombre **MyNewApp**. Visual Studio rellena la nueva aplicación con el código predeterminado "*Hello, World!*".

2. En el menú **Git**, seleccione **Crear un repositorio GIT**.
3. En el cuadro de diálogo **Crear un repositorio GIT**, en la sección **Enviar cambios a un nuevo repositorio remoto**, elija **Azure DevOps**.
4. En la sección **Crear un nuevo repositorio de Azure DevOps**, inicie sesión en su cuenta de Azure y seleccione un proyecto en la lista desplegable **Proyecto**.
5. Seleccione el botón **Crear y enviar los cambios** para crear el repositorio y agregar la aplicación.

Pasos siguientes

Para continuar con el recorrido, visite la página [Crear una bifurcación](#).

Vea también

- [Tutorial: Abrir un proyecto desde un repositorio](#)
- [Trabajar con cuentas de GitHub en Visual Studio](#)

Opciones y preferencias de Git en Visual Studio

Artículo • 28/02/2023 • Tiempo de lectura: 12 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

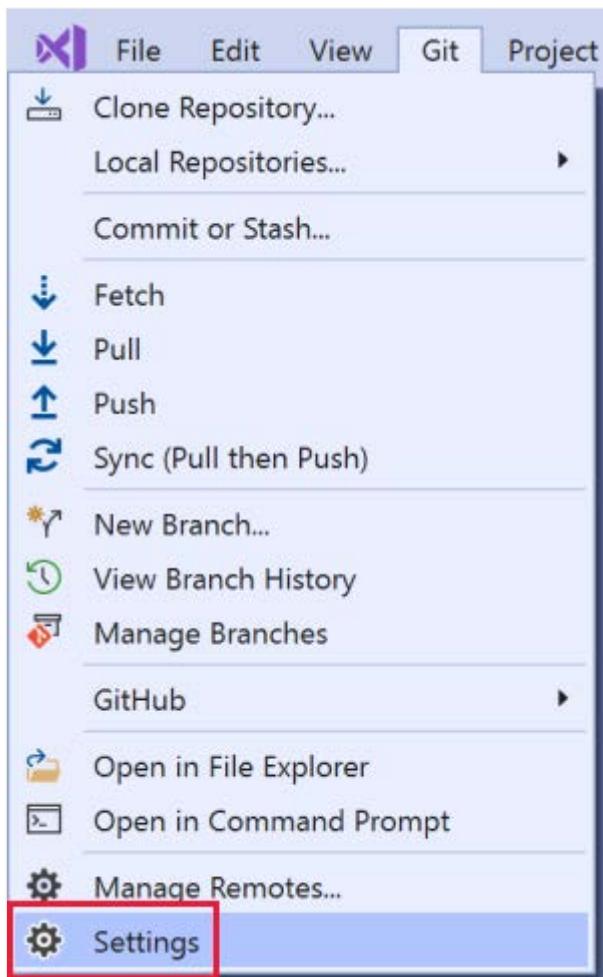
En Visual Studio, puede configurar y visualizar las opciones y preferencias comunes de Git, como su nombre y dirección de correo electrónico, las herramientas preferidas de comparación y fusión mediante combinación, etc. Estas opciones y preferencias se pueden ver y configurar en el **cuadro de diálogo Opciones** de la página de **configuración global de Git** (se aplica a todos los repositorios) o en la página de **configuración de repositorios de Git** (se aplica al repositorio actual).

Puede configurar dos tipos de opciones:

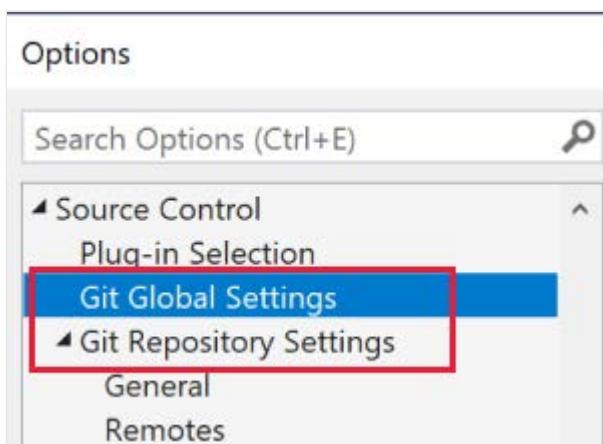
- [Opciones de Git](#): las opciones de esta sección se corresponden con las opciones de Git que están guardadas en los archivos de configuración de Git. Estas opciones puede verse y modificarse en Visual Studio, pero se administran mediante archivos de configuración de Git.
- [Opciones de Visual Studio](#): las opciones de esta sección configuran las preferencias y opciones relacionadas con Git que administra Visual Studio.

Cómo configurar los valores

1. Para configurar las opciones de Git en Visual Studio, elija **Configuración** en el menú de Git de nivel superior.



2. Elija **Git Global Settings** (Configuración global de Git) o **Configuración de repositorios de Git** para ver y configurar la configuración global o de nivel de repositorio.



3. Puede configurar varias opciones comunes de Git, como se describe en las secciones siguientes de este artículo. Despu s de configurar las opciones deseadas, seleccione **Aceptar** para guardar la configuraci n actualizada.



Configuración de Git

También puede configurar y comprobar algunas de las opciones de configuración de Git más comunes. Puede ver y modificar las opciones siguientes en Visual Studio, aunque se administran mediante archivos de configuración de Git.

- [Nombre y correo electrónico](#)
- [Eliminación de ramas remotas durante la captura](#)
- [Fusión de la rama local mediante cambio de base al enviar cambios](#)
- [Proveedor de la red criptográfica](#)
- [Aplicación auxiliar de credenciales](#)
- [Herramientas de comparación y fusión mediante combinación](#)
- [Archivos de Git](#)
- [Remotas](#)
- [Otras configuraciones](#)

ⓘ Nota

Las opciones de Git establecidas en la [configuración global](#) de Visual Studio se corresponden con las opciones del archivo de configuración específico del usuario de Git, mientras que las opciones de [configuración de repositorios](#) se corresponden con las opciones del archivo de configuración específico del repositorio. Para obtener más información sobre la configuración de Git, consulte el [capítulo de Git para profesionales sobre la personalización de Git](#), la [documentación de git-config](#) y la [referencia de Git para profesionales sobre archivos de configuración](#). Para configurar las opciones de Git que no están expuestas en Visual Studio, use el comando `git config` para escribir un valor en los archivos de configuración: `git config [--local|--global|--system] section.key value`.

Nombre y correo electrónico

El nombre y el correo electrónico que proporcione se usarán como información del confirmador para cualquier confirmación que realice. Esta opción está disponible en ámbitos globales y de repositorio, y se corresponde con las opciones [user.name](#) y [user.email](#) de `git config`.

1. En el menú de Git, vaya a **Configuración**. Para establecer el nombre de usuario y el correo electrónico en el nivel global, vaya a **Git Global Settings** (Configuración

global de Git); para establecer el nombre de usuario y el correo electrónico en el nivel de repositorio, vaya a **Configuración de repositorios de Git**.

2. Proporcione el nombre de usuario y el correo electrónico y seleccione **Aceptar** para guardar.



Eliminación de ramas remotas durante la captura

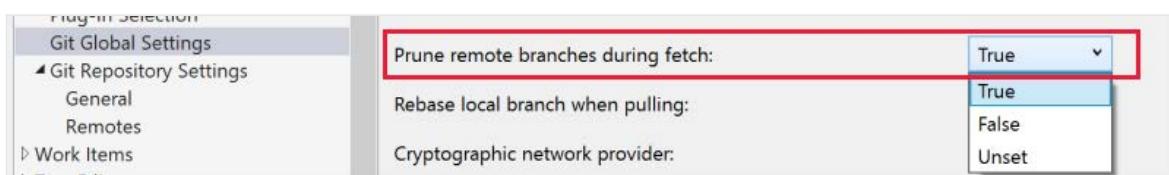
La eliminación quita ramas de seguimiento remotas que ya no existen en el origen remoto y permite mantener la lista de ramas limpia y actualizada. Esta opción está disponible en ámbitos globales y de repositorio, y se corresponde con la opción [fetch.prune](#) de `git config`.

Se recomienda establecer esta opción en **True** en el nivel global. Los valores válidos son los siguientes:

- **True** (recomendado)
- **False**
- **Sin establecer** (predeterminado)

A continuación se indica cómo cambiar los valores:

1. En el menú de Git, vaya a **Configuración**. Vaya a **Git Global Settings** (Configuración global de Git) para configurar esta opción en el nivel global; vaya a **Configuración de repositorios de Git** para configurar esta opción en el nivel de repositorio.
2. Establezca **Eliminar ramas remotas durante la captura** en **True** (recomendado). Seleccione **Aceptar** para guardar.



Fusión de la rama local mediante cambio de base al enviar cambios

Con la fusión mediante cambio de base, se separan los cambios realizados mediante confirmaciones en la rama actual que no se encuentran en la rama ascendente, se

restablece la rama actual en la rama ascendente y, después, se aplican los cambios que se separaron. Esta opción está disponible en ámbitos globales y de repositorio, y se corresponde con la opción `fetch.prune` ↗ de `git config`. Los valores válidos son los siguientes:

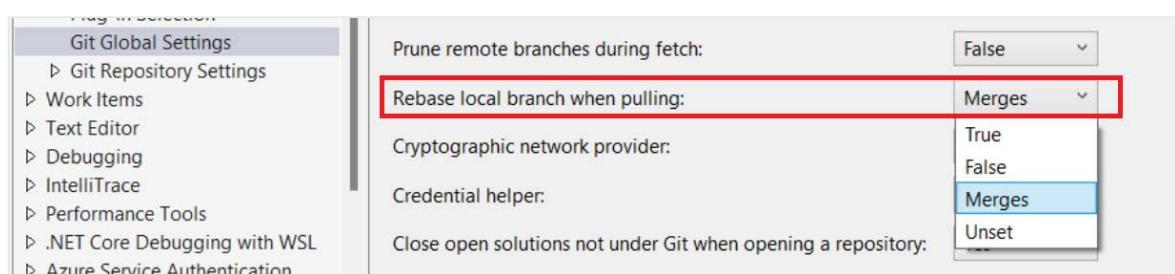
- **True**: fusione la rama actual mediante cambio de base en la parte superior de la rama ascendente después de la captura.
- **False**: fusione la rama actual mediante combinación en la rama ascendente.
- **Sin establecer** (valor predeterminado): a menos que se especifique en otros archivos de configuración, fusione la rama actual mediante combinación en la rama ascendente.
- **Interactivo**: fusione mediante cambio de base en modo interactivo.
- **Combinaciones**: fusione mediante cambio de base sin acoplar confirmaciones de combinación creadas localmente.

⚠ Nota

En la **versión 17.2** de Visual Studio 2022, hemos cambiado la opción "Conservar" a "Combinaciones" para que coincida con una actualización reciente de Git. Por lo tanto, si usa una versión anterior de Visual Studio con herramientas de Git, la interfaz de usuario podría mostrar "Conservar" en lugar de "Combinaciones". Aun así, la funcionalidad sigue siendo la misma.

A continuación se indica cómo cambiar los valores:

1. En el menú de Git, vaya a **Configuración**. Vaya a **Git Global Settings** (Configuración global de Git) para configurar esta opción en el nivel global; vaya a **Configuración de repositorios de Git** para configurar esta opción en el nivel de repositorio.
2. Establezca **Fusionar la rama local mediante cambio de base al enviar cambios** en la opción deseada y seleccione **Aceptar** para guardar.



No es posible configurar `pull.rebase` en **Interactivo** en Visual Studio. Visual Studio no tiene compatibilidad con la fusión mediante cambio de base interactiva. Para configurar `pull.rebase` de modo que use el modo interactivo, use la línea de comandos.

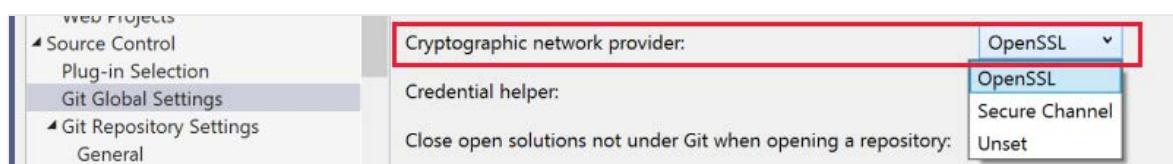
Proveedor de la red criptográfica

El proveedor de la red criptográfica es una opción de configuración de Git de ámbito global que configura qué back-end TLS/SSL se debe usar en tiempo de ejecución, y se corresponde con la opción [http.sslBackend](#) de `git config`. Los valores son los siguientes:

- **OpenSSL**: use [OpenSSL](#) para protocolos TLS y SSL.
- **Canal seguro**: use [Secure Channel \(SChannel\)](#) (Canal seguro [SChannel]) para protocolos TLS y SSL. SChannel es la solución nativa de Windows, que accede al Almacén de credenciales de Windows, lo que permite la administración de certificados en toda la empresa.
- **Sin establecer** (valor predeterminado): si esta opción no está establecida, el valor predeterminado es OpenSSL.

A continuación se indica cómo cambiar los valores:

1. En el menú de Git, vaya a **Configuración**. Vaya a **Git Global Settings** (Configuración global de Git) para configurar esta opción.
2. Establezca **Proveedor de la red criptográfica** en el valor deseado y seleccione **Aceptar** para guardar.



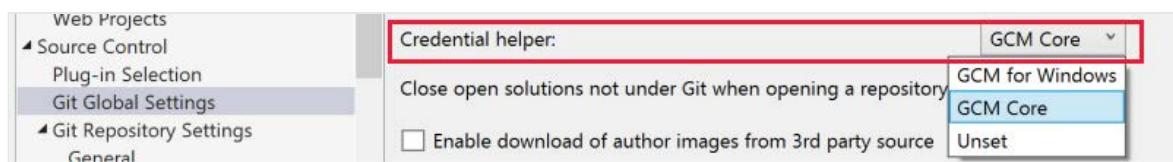
Aplicación auxiliar de credenciales

Cuando Visual Studio realiza una operación de Git remota, el punto de conexión remoto podría rechazar la solicitud porque requiere que se proporcionen credenciales con la solicitud. En ese momento, Git invoca una aplicación auxiliar de credenciales, que devuelve las credenciales necesarias para realizar la operación y, a continuación, volverá a intentar la solicitud. La aplicación auxiliar de credenciales usada se corresponde con la opción [credential.helper](#) de `git config`. Está disponible en el ámbito global con los valores siguientes:

- **GCM para Windows**: use el [Administrador de credenciales de Git para Windows](#) como aplicación auxiliar.
- **GCM Core**: use el [Administrador de credenciales de Git Core](#) como aplicación auxiliar.
- **Sin establecer** (valor predeterminado): si esta opción no está establecida, se usa la aplicación auxiliar de credenciales establecida en la configuración del sistema. A partir de Git para Windows 2.29, la aplicación auxiliar de credenciales predeterminada es GCM Core.

A continuación se indica cómo cambiar los valores:

1. En el menú de Git, vaya a **Configuración**. Vaya a **Git Global Settings** (Configuración global de Git) para configurar esta opción.
2. Establezca **Credential helper** (Aplicación auxiliar de credenciales) en el valor deseado y seleccione **Aceptar** para guardar.



Herramientas de comparación y fusión mediante combinación

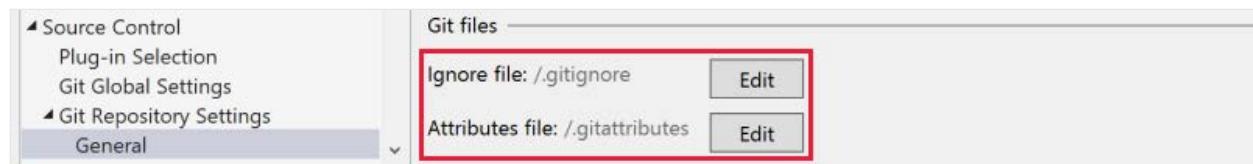
Git mostrará las comparaciones y los conflictos de combinación en las herramientas que usted prefiera. Las opciones de esta sección se corresponden con las opciones `diff.tool` y `merge.tool` de `git config`. Puede configurar Git para usar Visual Studio como herramienta de comparación o fusión mediante combinación en **Git Global Settings** (Configuración global de Git) y **Configuración de repositorios de Git**. Para ello, seleccione **Usar Visual Studio**. Para configurar otras herramientas de comparación y fusión mediante combinación, use `git config` con el modificador `diff.tool` o `merge.tool`.



Archivos de Git

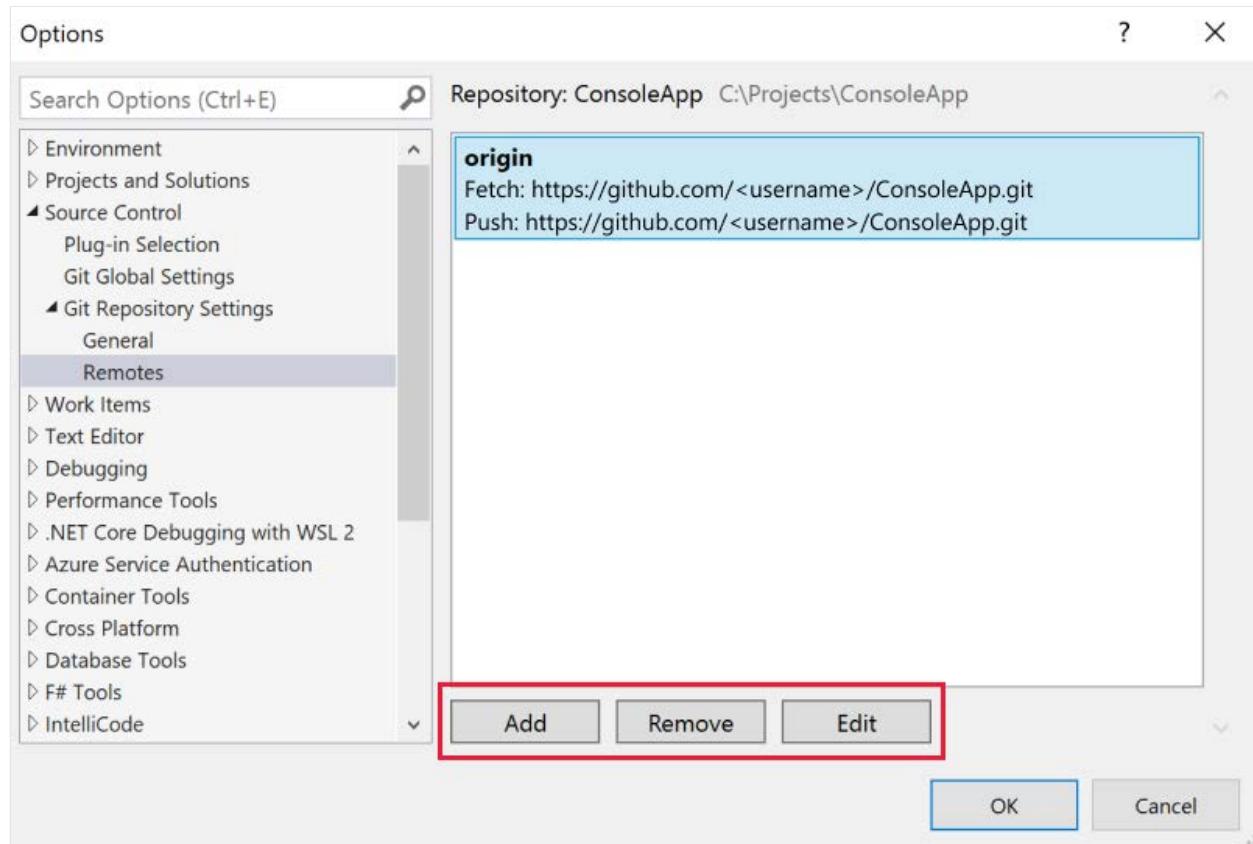
Puede usar la sección **Git files** (Archivos de Git) en el ámbito **Configuración de repositorios de Git** para ver y editar los archivos [gitignore](#) y [gitattributes](#) del

repositorio.



Remotas

Puede usar el panel Remotos en **Configuración de repositorios de Git** para configurar los orígenes remotos para el repositorio. Esta opción se corresponde con el comando [git remote](#) y permite agregar, editar o quitar orígenes remotos.



Otra configuración

Para ver todas las demás opciones de configuración de Git, abra y consulte los archivos de configuración, o bien ejecute `git config --list` para mostrar la configuración.

Configuración de Visual Studio

Las opciones siguientes administran las preferencias relacionadas con Git en Visual Studio y se administran mediante Visual Studio, en lugar de archivos de configuración de Git. Todas las opciones de esta sección se configuran en la página de configuración global de Git.

- Ubicación predeterminada
- Cierre de las soluciones abiertas que no estén en Git al abrir un repositorio
- Habilitación de la descarga de imágenes de autor de otros orígenes
- Confirmación de cambios tras la fusión mediante combinación de forma predeterminada
- Habilitación de la inserción de --force
- Apertura de una carpeta en el Explorador de soluciones al abrir un repositorio de Git
- Carga automática de la solución al abrir un repositorio de Git
- Extracción de las ramas automáticamente del repositorio con un doble clic o con la tecla ENTRAR

Ubicación predeterminada

La opción **Ubicación predeterminada** permite configurar la carpeta predeterminada en la que se clonian los repositorios.

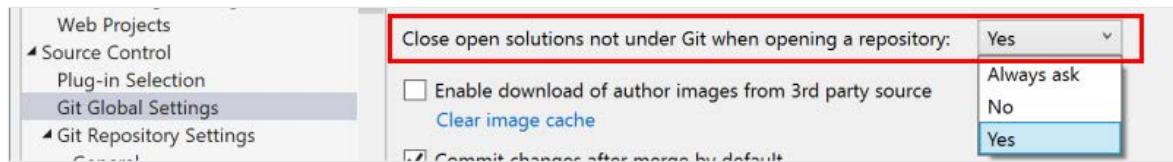


Cierre de las soluciones abiertas que no estén en Git al abrir un repositorio

De forma predeterminada, Visual Studio cierra cualquier solución o carpeta abierta al cambiar a otro repositorio. Cuando lo hace, también podría cargar la solución o carpeta del nuevo repositorio en función de si decide [abrir una carpeta en el Explorador de soluciones al abrir un repositorio de Git](#) y [cargar automáticamente la solución al abrir un repositorio de Git](#). Esto mantiene la coherencia entre el código abierto y el repositorio abierto. Aun así, si la solución no se encuentra en la misma raíz de carpeta que el repositorio, es posible que le interese mantener abierta la solución al cambiar al repositorio. Puede hacerlo con esta opción. Los valores son los siguientes:

- **Sí**: cuando se abre un repositorio, siempre se cierra la solución abierta actualmente.
- **No**: cuando se abre un repositorio, Visual Studio realiza una comprobación para saber si la solución actual está en Git. Si no es así, la solución permanece abierta.
- **Preguntar siempre** (valor predeterminado): cuando se establece este valor, puede elegir en un cuadro de diálogo por cada repositorio abierto si quiere mantener

abierta la solución actual o cerrarla.



Habilitación de la descarga de imágenes de autor de otros orígenes

La habilitación de la descarga de imágenes de autor de otros orígenes es una configuración específica de Visual Studio en el ámbito global. Cuando se activa, las imágenes de autor se descargan del [servicio de imágenes de Gravatar](#), si están disponibles, y se muestran en las vistas Confirmación e Historial.

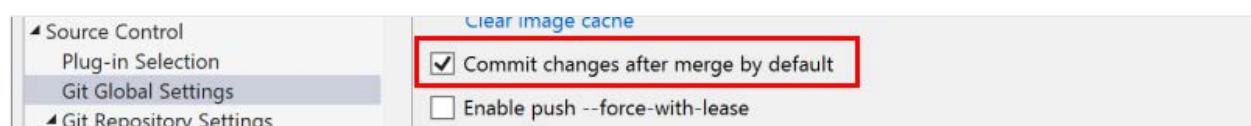


ⓘ Importante

Para proporcionar imágenes de autor en las vistas Confirmación e Historial, la herramienta crea un hash MD5 para las direcciones de correo electrónico del autor almacenadas en el repositorio activo. A continuación, este hash se envía a Gravatar para buscar un valor hash coincidente para los usuarios que se registraron previamente en el servicio. Si se encuentra una coincidencia, la imagen de usuario se recuperará del servicio y se mostrará en Visual Studio. Los usuarios que no hayan configurado el servicio devolverán una imagen generada aleatoriamente. Tenga en cuenta que las direcciones de correo electrónico no se registran en Visual Studio y no se comparten nunca con Gravatar ni con ningún otro tercero.

Confirmación de cambios tras la fusión mediante combinación de forma predeterminada

Cuando la opción **Confirmar cambios tras la fusión mediante combinación de forma predeterminada** está habilitada, Git crea automáticamente una confirmación cuando una rama se fusiona mediante combinación con la rama actual.

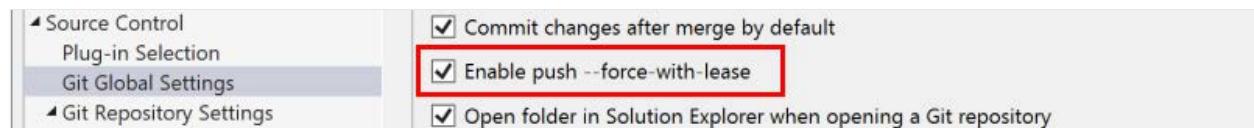


- Cuando está activada, los comandos `git merge` emitidos por Visual Studio se ejecutan con la opción `--commit`.
- Cuando está desactivada, los comandos `git merge` emitidos por Visual Studio se ejecutan con las opciones `--no-commit --no-ff`.

Para obtener más información sobre estas opciones, consulte [--commit](#) y [--no-commit](#) y [--no-ff](#).

Habilitación de la inserción de `--force-with-lease`

Cuando está habilitada, esta opción permite realizar la acción `push --force-with-lease` desde Visual Studio. De forma predeterminada, la opción **Enable push --force-with-lease** (Habilitar la inserción de `--force-with-lease`) está deshabilitada.

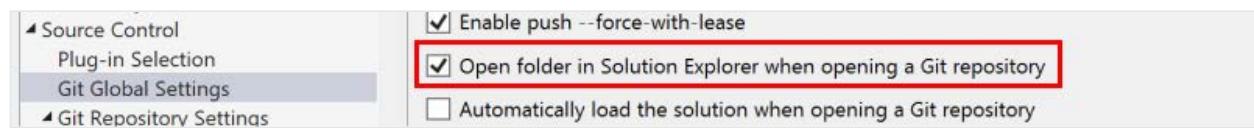


Para obtener más información, consulte [Inserción de `--force-with-lease`](#).

Apertura de una carpeta en el Explorador de soluciones al abrir un repositorio de Git

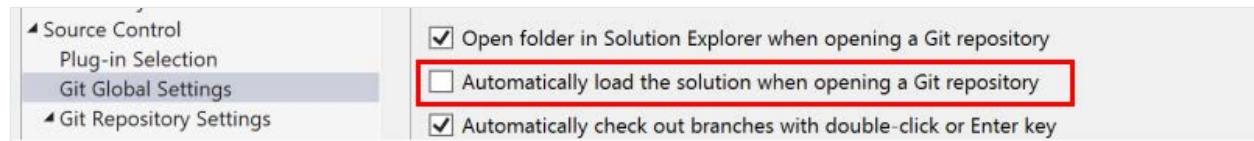
Cuando se usa Visual Studio para abrir o cambiar a un repositorio de Git, Visual Studio carga el contenido de Git para que pueda ver los cambios, confirmaciones y ramas, así como administrar el repositorio desde el IDE. Además, Visual Studio también cargará el código del repositorio en el Explorador de soluciones. Visual Studio examinará la carpeta del repositorio en busca de soluciones, CMakeLists.txt o cualquier otro archivo de vista que reconozca y los mostrará como una lista en el Explorador de soluciones.

Desde allí, puede seleccionar una solución para cargarla o la carpeta para ver el contenido del directorio. Si se desactiva esta casilla, Visual Studio no abrirá la carpeta del repositorio en el Explorador de soluciones. Esto le permitirá abrir Visual Studio solo como administrador de repositorios de Git. Esta configuración está activada de manera predeterminada.



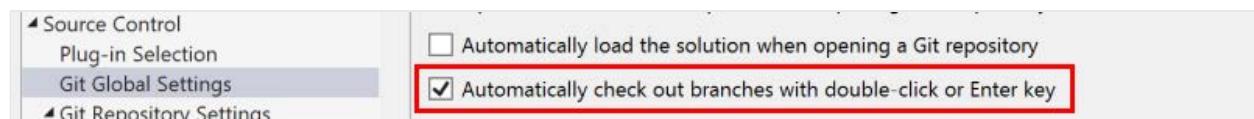
Carga automática de la solución al abrir un repositorio de Git

Esta opción solo se aplica cuando está activada la opción [Open folder in Solution Explorer when opening a Git repository](#) (Abrir una carpeta en el Explorador de soluciones al abrir un repositorio de Git). Cuando se abre un repositorio de Git en Visual Studio y el examen de carpetas posterior detecta que solo hay una solución en el repositorio, Visual Studio carga automáticamente esa solución. Si desactiva la opción, el Explorador de soluciones mostrará la única solución presente en el repositorio en la lista de vistas, pero no la cargará. De forma predeterminada, esta opción está desactivada.



Extracción de las ramas automáticamente del repositorio con un doble clic o con la tecla ENTRAR

La ventana Repositorio de Git tiene una lista de ramas que se muestran en una estructura de árbol. Al seleccionar una rama, el panel del historial de confirmaciones cambiará y mostrará las confirmaciones de la rama seleccionada. Para extraer del repositorio una rama, haga clic con el botón derecho para abrir el menú contextual y seleccione **Extraer del repositorio**. Si activa esta opción, al hacer doble clic o presionar la tecla ENTRAR se extraerá la rama del repositorio y se mostrarán sus confirmaciones.



Consulte también

ⓘ Importante

Si tiene alguna sugerencia para nosotros, háganoslo saber. Agradecemos la oportunidad de participar en sus decisiones de diseño a través del portal [Developer Community](#).

- Vídeo [Introducción a Git en Visual Studio](#) en YouTube
- [Productividad mejorada con Git en Visual Studio](#) (entrada de blog)
- [Opciones \(cuadro de diálogo\)](#)

Creación de una rama de Git en Visual Studio

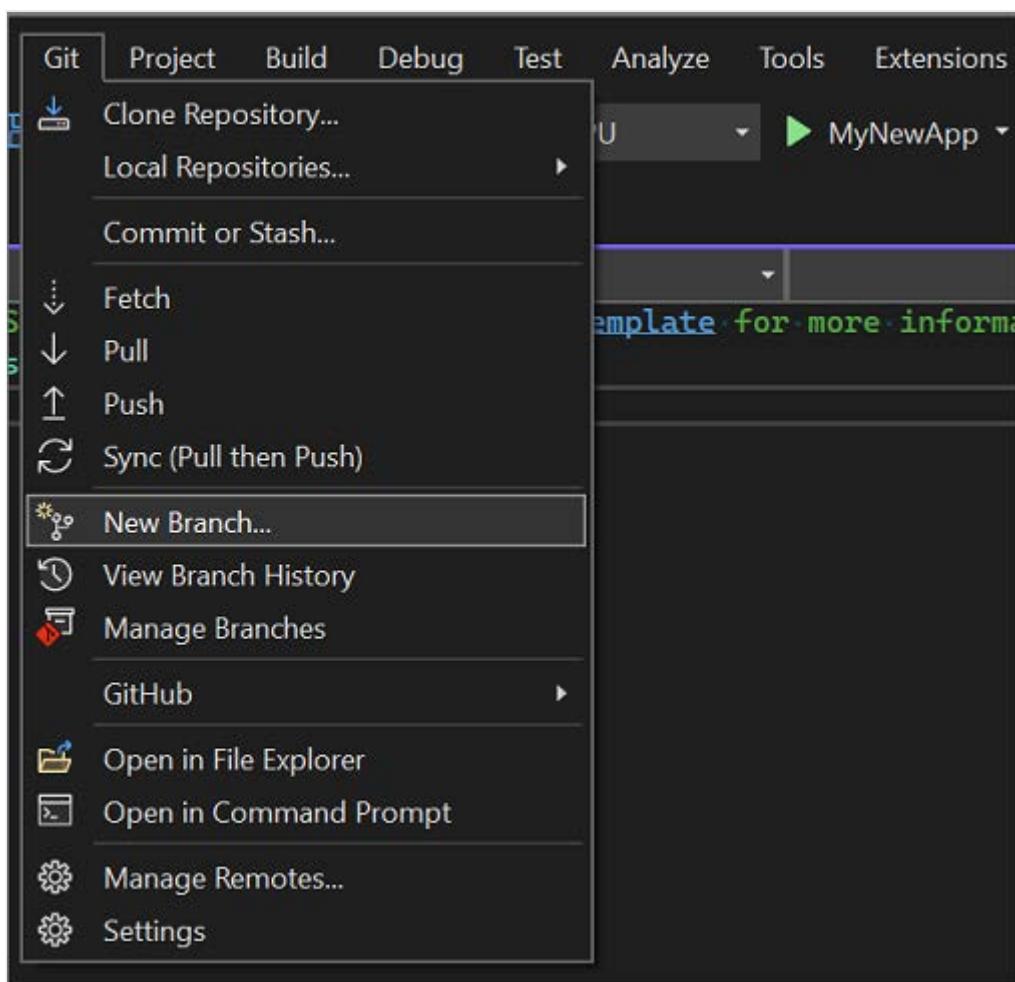
Artículo • 18/03/2023 • Tiempo de lectura: 2 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

Es fácil crear una nueva rama en Visual Studio; lo único que tiene que hacer es basarla en una rama existente.

Esta es la manera de hacerlo.

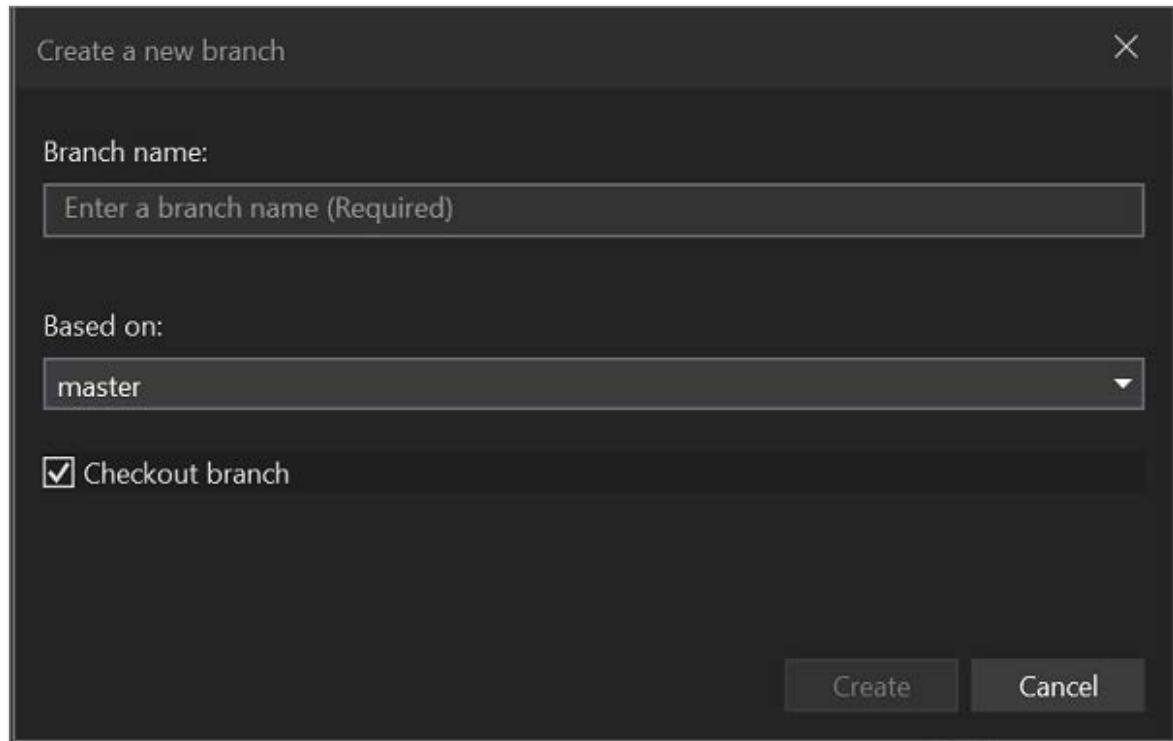
1. Para empezar, asegúrese de que tiene abierto un repositorio [creado](#) o [clonado](#) previamente.
2. En el menú **Git**, seleccione **Nueva rama**.



3. En el cuadro de diálogo **Crear una nueva rama**, escriba un nombre de rama.

Sugerencia

Para obtener detalles de nomenclatura de ramas, consulte [Caracteres especiales en nombres de rama y etiqueta](#).



4. En la sección **Basada en**, use la lista desplegable para elegir si desea basar la nueva rama en una rama local existente o en una rama remota.
5. La casilla **Desproteger rama**, que está activada de forma predeterminada, cambia automáticamente a la rama recién creada. Cambie esta opción si desea permanecer en la rama actual.

Ya ha creado una nueva rama.

Sugerencia

El comando equivalente para esta acción es `git checkout -b <new-branch> <existing-branch>`.

Nota

Para obtener más información sobre las actualizaciones más recientes que mejoran el cambio de rama, consulte la entrada de blog [Mejoras de rendimiento de Visual Studio 2022: Cambio de rama de Git](#).

Pasos siguientes

Para continuar su recorrido, visite la página [Realizar confirmación](#). Y para obtener más información sobre cómo administrar ramas en Visual Studio, consulte [Combinación y rebase de ramas](#).

Realización de una confirmación de Git en Visual Studio

Artículo • 27/09/2022 • Tiempo de lectura: 2 minutos

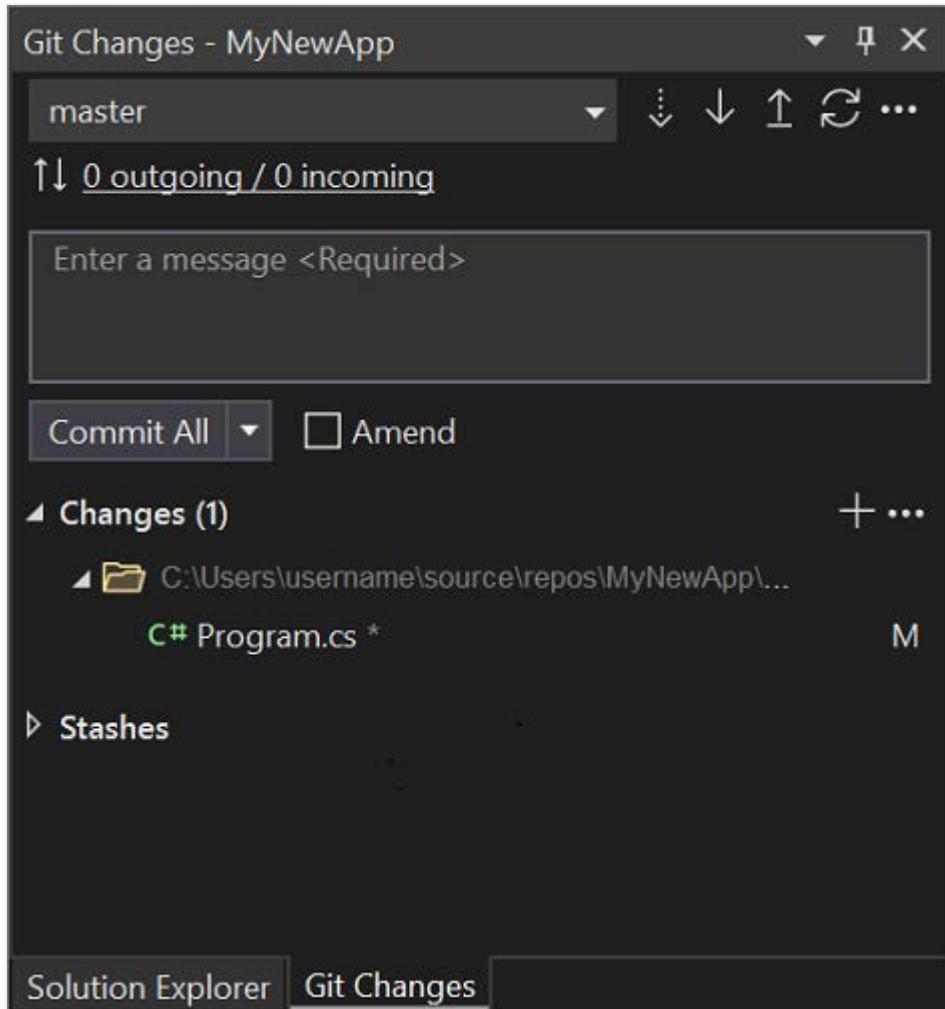
Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

El elemento principal de cualquier flujo de trabajo de Git es la modificación de archivos y la confirmación de los cambios en esos archivos. Aunque en este artículo se hace referencia a repositorios de GitHub, puede trabajar de forma remota con el proveedor de Git que prefiera, como GitHub o Azure DevOps. También puede trabajar en modo local sin ningún proveedor.

Git realiza un seguimiento de los cambios de archivo en el repositorio mientras usted trabaja y separa los archivos del repositorio en tres categorías. Estos cambios son equivalentes a lo que se vería al escribir el comando `git status` en la línea de comandos:

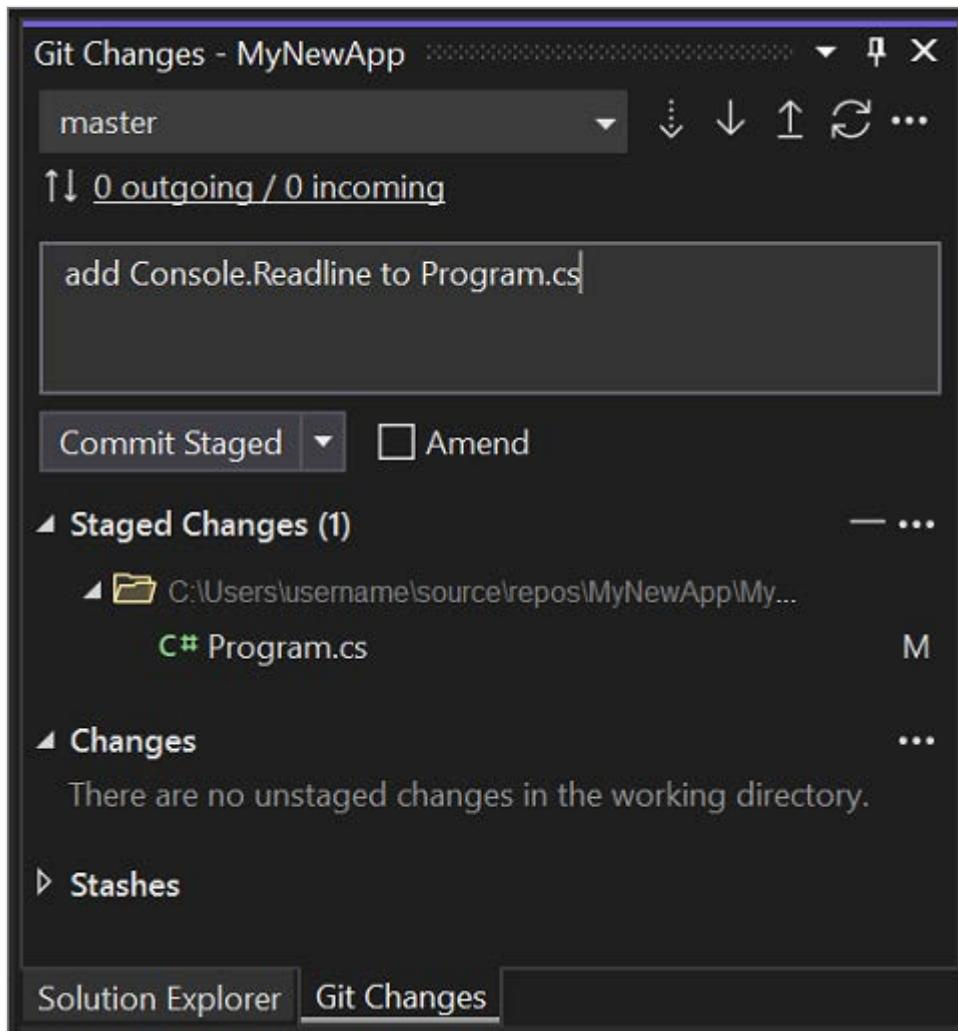
- **Archivos sin modificar:** Estos archivos no han cambiado desde la última confirmación.
- **Archivos modificados:** Estos archivos incluyen cambios realizados desde la última confirmación, pero aún no se han almacenado provisionalmente para la siguiente confirmación.
- **Archivos almacenados provisionalmente:** Estos archivos tienen cambios que se agregarán a la siguiente confirmación.

Mientras usted realiza su trabajo, Visual Studio realiza un seguimiento de los cambios de archivo en el proyecto en la sección **Cambios** de la ventana **Cambios de Git**.



Para almacenar provisionalmente los cambios cuando esté listo, seleccione el botón (más) en cada archivo que quiera almacenar provisionalmente, o haga clic con el + botón derecho en un archivo y, a continuación, seleccione **Fase**. También puede almacenar provisionalmente todos los archivos modificados con un solo clic mediante el botón + (más) de Almacenar todo provisionalmente situado en la parte superior de la sección **Cambios**.

Al almacenar provisionalmente un cambio, Visual Studio crea una sección **Cambios almacenados provisionalmente**. Solo se agregan en la siguiente confirmación los cambios de la sección **Cambios almacenados provisionalmente**, lo que puede hacer seleccionando **Confirmar almacenados provisionalmente**. El comando equivalente para esta acción es `git commit -m "Your commit message"`.



También se puede cambiar el almacenamiento provisional de los cambios haciendo clic en el botón – (menos). El comando equivalente para esta acción es `git reset <file_path>` para cambiar el almacenamiento provisional de los cambios de un único archivo o `git reset <directory_path>` para cambiar el de todos los archivos de un directorio.

También puede optar por no almacenar provisionalmente los archivos modificados omitiendo el almacenamiento provisional. En este caso, Visual Studio le permite confirmar los cambios directamente sin tener que almacenarlos provisionalmente. Solo tiene que escribir el mensaje de confirmación y luego seleccionar **Confirmar todo**. El comando equivalente para esta acción es `git commit -a`.

Visual Studio también facilita la confirmación y sincronización con un solo clic mediante los métodos abreviados **Confirmar todo e insertar** y **Confirmar todo y sincronizar**. Al hacer doble clic en cualquier archivo de las secciones sección **Cambios y Cambios almacenados provisionalmente**, puede ver una comparación línea a línea con la versión no modificada del archivo.

Git Repository - Files

INavigationToolbar.cs

NavToolbarViewModel.cs

ItemViewModel.cs

Diff - ItemV...ViewModel.cs

↑ ↓ 4 changes -8 +7

ItemViewModel.cs;HEAD

Miscellaneous Files

```

2163     Connection.RequestReceived -= ConnectionRequestReceived;
2164 }
2165     AppServiceConnectionHelper.ConnectionChanged -= AppServiceConnectionHelper_ConnectionChanged;
2166     AppSettings.PropertyChanged -= AppSettings_PropertyChanged;
2167 }
2168 }
2169
2170     public class PageTypeUpdatedEventArgs
2171 {
2172     public bool IsTypeCloudDrive { get; set; }
2173     public bool IsTypeRecycleBin { get; set; }
2174 }
2175
2176     public class WorkingDirectoryModifiedEventArgs : EventArgs
2177 {
2178     public string Path { get; set; }
2179
2180     public string Name { get; set; }
2181
2182     public bool IsLibrary { get; set; }
2183 }
2184
2185     public class ItemLoadStatusChangedEventArgs : EventArgs
2186 {
2187     public enum ItemLoadStatus
2188     {
2189         Starting,
2190         InProgress,
2191         Complete
2192     }
2193 }
2194
2195     public class PreviewPaneEventArgs : EventArgs
2196 {
2197     public enum PreviewPaneType
2198     {
2199         Preview,
2200         PreviewLarge
2201     }
2202 }
2203
2204     public class PreviewPanelEventArgs : EventArgs
2205 {
2206     public enum PreviewPanelType
2207     {
2208         Preview,
2209         PreviewLarge
2210     }
2211 }
2212
2213     public class PreviewPanelGridSplitterEventArgs : EventArgs
2214 {
2215     public enum PreviewPanelGridSplitterType
2216     {
2217         Preview,
2218         PreviewLarge
2219     }
2220 }
2221
2222     public class PreviewPanelGridEventArgs : EventArgs
2223 {
2224     public enum PreviewPanelGridType
2225     {
2226         Preview,
2227         PreviewLarge
2228     }
2229 }
2230
2231     public class PreviewPanelEventArgs : EventArgs
2232 {
2233     public enum PreviewPanelType
2234     {
2235         Preview,
2236         PreviewLarge
2237     }
2238 }
2239
2240     public class PreviewPanelEventArgs : EventArgs
2241 {
2242     public enum PreviewPanelType
2243     {
2244         Preview,
2245         PreviewLarge
2246     }
2247 }
2248
2249     public class PreviewPanelEventArgs : EventArgs
2250 {
2251     public enum PreviewPanelType
2252     {
2253         Preview,
2254         PreviewLarge
2255     }
2256 }
2257
2258     public class PreviewPanelEventArgs : EventArgs
2259 {
2260     public enum PreviewPanelType
2261     {
2262         Preview,
2263         PreviewLarge
2264     }
2265 }
2266
2267     public class PreviewPanelEventArgs : EventArgs
2268 {
2269     public enum PreviewPanelType
2270     {
2271         Preview,
2272         PreviewLarge
2273     }
2274 }
2275
2276     public class PreviewPanelEventArgs : EventArgs
2277 {
2278     public enum PreviewPanelType
2279     {
2280         Preview,
2281         PreviewLarge
2282     }
2283 }
2284
2285     public class PreviewPanelEventArgs : EventArgs
2286 {
2287     public enum PreviewPanelType
2288     {
2289         Preview,
2290         PreviewLarge
2291     }
2292 }
2293
2294     public class PreviewPanelEventArgs : EventArgs
2295 {
2296     public enum PreviewPanelType
2297     {
2298         Preview,
2299         PreviewLarge
2300     }
2301 }
2302
2303     public class PreviewPanelEventArgs : EventArgs
2304 {
2305     public enum PreviewPanelType
2306     {
2307         Preview,
2308         PreviewLarge
2309     }
2310 }
2311
2312     public class PreviewPanelEventArgs : EventArgs
2313 {
2314     public enum PreviewPanelType
2315     {
2316         Preview,
2317         PreviewLarge
2318     }
2319 }
2320
2321     public class PreviewPanelEventArgs : EventArgs
2322 {
2323     public enum PreviewPanelType
2324     {
2325         Preview,
2326         PreviewLarge
2327     }
2328 }
2329
2330     public class PreviewPanelEventArgs : EventArgs
2331 {
2332     public enum PreviewPanelType
2333     {
2334         Preview,
2335         PreviewLarge
2336     }
2337 }
2338
2339     public class PreviewPanelEventArgs : EventArgs
2340 {
2341     public enum PreviewPanelType
2342     {
2343         Preview,
2344         PreviewLarge
2345     }
2346 }
2347
2348     public class PreviewPanelEventArgs : EventArgs
2349 {
2350     public enum PreviewPanelType
2351     {
2352         Preview,
2353         PreviewLarge
2354     }
2355 }
2356
2357     public class PreviewPanelEventArgs : EventArgs
2358 {
2359     public enum PreviewPanelType
2360     {
2361         Preview,
2362         PreviewLarge
2363     }
2364 }
2365
2366     public class PreviewPanelEventArgs : EventArgs
2367 {
2368     public enum PreviewPanelType
2369     {
2370         Preview,
2371         PreviewLarge
2372     }
2373 }
2374
2375     public class PreviewPanelEventArgs : EventArgs
2376 {
2377     public enum PreviewPanelType
2378     {
2379         Preview,
2380         PreviewLarge
2381     }
2382 }
2383
2384     public class PreviewPanelEventArgs : EventArgs
2385 {
2386     public enum PreviewPanelType
2387     {
2388         Preview,
2389         PreviewLarge
2390     }
2391 }
2392
2393     public class PreviewPanelEventArgs : EventArgs
2394 {
2395     public enum PreviewPanelType
2396     {
2397         Preview,
2398         PreviewLarge
2399     }
2400 }
2401
2402     public class PreviewPanelEventArgs : EventArgs
2403 {
2404     public enum PreviewPanelType
2405     {
2406         Preview,
2407         PreviewLarge
2408     }
2409 }
2410
2411     public class PreviewPanelEventArgs : EventArgs
2412 {
2413     public enum PreviewPanelType
2414     {
2415         Preview,
2416         PreviewLarge
2417     }
2418 }
2419
2420     public class PreviewPanelEventArgs : EventArgs
2421 {
2422     public enum PreviewPanelType
2423     {
2424         Preview,
2425         PreviewLarge
2426     }
2427 }
2428
2429     public class PreviewPanelEventArgs : EventArgs
2430 {
2431     public enum PreviewPanelType
2432     {
2433         Preview,
2434         PreviewLarge
2435     }
2436 }
2437
2438     public class PreviewPanelEventArgs : EventArgs
2439 {
2440     public enum PreviewPanelType
2441     {
2442         Preview,
2443         PreviewLarge
2444     }
2445 }
2446
2447     public class PreviewPanelEventArgs : EventArgs
2448 {
2449     public enum PreviewPanelType
2450     {
2451         Preview,
2452         PreviewLarge
2453     }
2454 }
2455
2456     public class PreviewPanelEventArgs : EventArgs
2457 {
2458     public enum PreviewPanelType
2459     {
2460         Preview,
2461         PreviewLarge
2462     }
2463 }
2464
2465     public class PreviewPanelEventArgs : EventArgs
2466 {
2467     public enum PreviewPanelType
2468     {
2469         Preview,
2470         PreviewLarge
2471     }
2472 }
2473
2474     public class PreviewPanelEventArgs : EventArgs
2475 {
2476     public enum PreviewPanelType
2477     {
2478         Preview,
2479         PreviewLarge
2480     }
2481 }
2482
2483     public class PreviewPanelEventArgs : EventArgs
2484 {
2485     public enum PreviewPanelType
2486     {
2487         Preview,
2488         PreviewLarge
2489     }
2490 }
2491
2492     public class PreviewPanelEventArgs : EventArgs
2493 {
2494     public enum PreviewPanelType
2495     {
2496         Preview,
2497         PreviewLarge
2498     }
2499 }
2500
2501     public class PreviewPanelEventArgs : EventArgs
2502 {
2503     public enum PreviewPanelType
2504     {
2505         Preview,
2506         PreviewLarge
2507     }
2508 }
2509
2510     public class PreviewPanelEventArgs : EventArgs
2511 {
2512     public enum PreviewPanelType
2513     {
2514         Preview,
2515         PreviewLarge
2516     }
2517 }
2518
2519     public class PreviewPanelEventArgs : EventArgs
2520 {
2521     public enum PreviewPanelType
2522     {
2523         Preview,
2524         PreviewLarge
2525     }
2526 }
2527
2528     public class PreviewPanelEventArgs : EventArgs
2529 {
2530     public enum PreviewPanelType
2531     {
2532         Preview,
2533         PreviewLarge
2534     }
2535 }
2536
2537     public class PreviewPanelEventArgs : EventArgs
2538 {
2539     public enum PreviewPanelType
2540     {
2541         Preview,
2542         PreviewLarge
2543     }
2544 }
2545
2546     public class PreviewPanelEventArgs : EventArgs
2547 {
2548     public enum PreviewPanelType
2549     {
2550         Preview,
2551         PreviewLarge
2552     }
2553 }
2554
2555     public class PreviewPanelEventArgs : EventArgs
2556 {
2557     public enum PreviewPanelType
2558     {
2559         Preview,
2560         PreviewLarge
2561     }
2562 }
2563
2564     public class PreviewPanelEventArgs : EventArgs
2565 {
2566     public enum PreviewPanelType
2567     {
2568         Preview,
2569         PreviewLarge
2570     }
2571 }
2572
2573     public class PreviewPanelEventArgs : EventArgs
2574 {
2575     public enum PreviewPanelType
2576     {
2577         Preview,
2578         PreviewLarge
2579     }
2580 }
2581
2582     public class PreviewPanelEventArgs : EventArgs
2583 {
2584     public enum PreviewPanelType
2585     {
2586         Preview,
2587         PreviewLarge
2588     }
2589 }
2590
2591     public class PreviewPanelEventArgs : EventArgs
2592 {
2593     public enum PreviewPanelType
2594     {
2595         Preview,
2596         PreviewLarge
2597     }
2598 }
2599
2600     public class PreviewPanelEventArgs : EventArgs
2601 {
2602     public enum PreviewPanelType
2603     {
2604         Preview,
2605         PreviewLarge
2606     }
2607 }
2608
2609     public class PreviewPanelEventArgs : EventArgs
2610 {
2611     public enum PreviewPanelType
2612     {
2613         Preview,
2614         PreviewLarge
2615     }
2616 }
2617
2618     public class PreviewPanelEventArgs : EventArgs
2619 {
2620     public enum PreviewPanelType
2621     {
2622         Preview,
2623         PreviewLarge
2624     }
2625 }
2626
2627     public class PreviewPanelEventArgs : EventArgs
2628 {
2629     public enum PreviewPanelType
2630     {
2631         Preview,
2632         PreviewLarge
2633     }
2634 }
2635
2636     public class PreviewPanelEventArgs : EventArgs
2637 {
2638     public enum PreviewPanelType
2639     {
2640         Preview,
2641         PreviewLarge
2642     }
2643 }
2644
2645     public class PreviewPanelEventArgs : EventArgs
2646 {
2647     public enum PreviewPanelType
2648     {
2649         Preview,
2650         PreviewLarge
2651     }
2652 }
2653
2654     public class PreviewPanelEventArgs : EventArgs
2655 {
2656     public enum PreviewPanelType
2657     {
2658         Preview,
2659         PreviewLarge
2660     }
2661 }
2662
2663     public class PreviewPanelEventArgs : EventArgs
2664 {
2665     public enum PreviewPanelType
2666     {
2667         Preview,
2668         PreviewLarge
2669     }
2670 }
2671
2672     public class PreviewPanelEventArgs : EventArgs
2673 {
2674     public enum PreviewPanelType
2675     {
2676         Preview,
2677         PreviewLarge
2678     }
2679 }
2680
2681     public class PreviewPanelEventArgs : EventArgs
2682 {
2683     public enum PreviewPanelType
2684     {
2685         Preview,
2686         PreviewLarge
2687     }
2688 }
2689
2690     public class PreviewPanelEventArgs : EventArgs
2691 {
2692     public enum PreviewPanelType
2693     {
2694         Preview,
2695         PreviewLarge
2696     }
2697 }
2698
2699     public class PreviewPanelEventArgs : EventArgs
2700 {
2701     public enum PreviewPanelType
2702     {
2703         Preview,
2704         PreviewLarge
2705     }
2706 }
2707
2708     public class PreviewPanelEventArgs : EventArgs
2709 {
2710     public enum PreviewPanelType
2711     {
2712         Preview,
2713         PreviewLarge
2714     }
2715 }
2716
2717     public class PreviewPanelEventArgs : EventArgs
2718 {
2719     public enum PreviewPanelType
2720     {
2721         Preview,
2722         PreviewLarge
2723     }
2724 }
2725
2726     public class PreviewPanelEventArgs : EventArgs
2727 {
2728     public enum PreviewPanelType
2729     {
2730         Preview,
2731         PreviewLarge
2732     }
2733 }
2734
2735     public class PreviewPanelEventArgs : EventArgs
2736 {
2737     public enum PreviewPanelType
2738     {
2739         Preview,
2740         PreviewLarge
2741     }
2742 }
2743
2744     public class PreviewPanelEventArgs : EventArgs
2745 {
2746     public enum PreviewPanelType
2747     {
2748         Preview,
2749         PreviewLarge
2750     }
2751 }
2752
2753     public class PreviewPanelEventArgs : EventArgs
2754 {
2755     public enum PreviewPanelType
2756     {
2757         Preview,
2758         PreviewLarge
2759     }
2760 }
2761
2762     public class PreviewPanelEventArgs : EventArgs
2763 {
2764     public enum PreviewPanelType
2765     {
2766         Preview,
2767         PreviewLarge
2768     }
2769 }
2770
2771     public class PreviewPanelEventArgs : EventArgs
2772 {
2773     public enum PreviewPanelType
2774     {
2775         Preview,
2776         PreviewLarge
2777     }
2778 }
2779
2780     public class PreviewPanelEventArgs : EventArgs
2781 {
2782     public enum PreviewPanelType
2783     {
2784         Preview,
2785         PreviewLarge
2786     }
2787 }
2788
2789     public class PreviewPanelEventArgs : EventArgs
2790 {
2791     public enum PreviewPanelType
2792     {
2793         Preview,
2794         PreviewLarge
2795     }
2796 }
2797
2798     public class PreviewPanelEventArgs : EventArgs
2799 {
2800     public enum PreviewPanelType
2801     {
2802         Preview,
2803         PreviewLarge
2804     }
2805 }
2806
2807     public class PreviewPanelEventArgs : EventArgs
2808 {
2809     public enum PreviewPanelType
2810     {
2811         Preview,
2812         PreviewLarge
2813     }
2814 }
2815
2816     public class PreviewPanelEventArgs : EventArgs
2817 {
2818     public enum PreviewPanelType
2819     {
2820         Preview,
2821         PreviewLarge
2822     }
2823 }
2824
2825     public class PreviewPanelEventArgs : EventArgs
2826 {
2827     public enum PreviewPanelType
2828     {
2829         Preview,
2830         PreviewLarge
2831     }
2832 }
2833
2834     public class PreviewPanelEventArgs : EventArgs
2835 {
2836     public enum PreviewPanelType
2837     {
2838         Preview,
2839         PreviewLarge
2840     }
2841 }
2842
2843     public class PreviewPanelEventArgs : EventArgs
2844 {
2845     public enum PreviewPanelType
2846     {
2847         Preview,
2848         PreviewLarge
2849     }
2850 }
2851
2852     public class PreviewPanelEventArgs : EventArgs
2853 {
2854     public enum PreviewPanelType
2855     {
2856         Preview,
2857         PreviewLarge
2858     }
2859 }
2860
2861     public class PreviewPanelEventArgs : EventArgs
2862 {
2863     public enum PreviewPanelType
2864     {
2865         Preview,
2866         PreviewLarge
2867     }
2868 }
2869
2870     public class PreviewPanelEventArgs : EventArgs
2871 {
2872     public enum PreviewPanelType
2873     {
2874         Preview,
2875         PreviewLarge
2876     }
2877 }
2878
2879     public class PreviewPanelEventArgs : EventArgs
2880 {
2881     public enum PreviewPanelType
2882     {
2883         Preview,
2884         PreviewLarge
2885     }
2886 }
2887
2888     public class PreviewPanelEventArgs : EventArgs
2889 {
2890     public enum PreviewPanelType
2891     {
2892         Preview,
2893         PreviewLarge
2894     }
2895 }
2896
2897     public class PreviewPanelEventArgs : EventArgs
2898 {
2899     public enum PreviewPanelType
2900     {
2901         Preview,
2902         PreviewLarge
2903     }
2904 }
2905
2906     public class PreviewPanelEventArgs : EventArgs
2907 {
2908     public enum PreviewPanelType
2909     {
2910         Preview,
2911         PreviewLarge
2912     }
2913 }
2914
2915     public class PreviewPanelEventArgs : EventArgs
2916 {
2917     public enum PreviewPanelType
2918     {
2919         Preview,
2920         PreviewLarge
2921     }
2922 }
2923
2924     public class PreviewPanelEventArgs : EventArgs
2925 {
2926     public enum PreviewPanelType
2927     {
2928         Preview,
2929         PreviewLarge
2930     }
2931 }
2932
2933     public class PreviewPanelEventArgs : EventArgs
2934 {
2935     public enum PreviewPanelType
2936     {
2937         Preview,
2938         PreviewLarge
2939     }
2940 }
2941
2942     public class PreviewPanelEventArgs : EventArgs
2943 {
2944     public enum PreviewPanelType
2945     {
2946         Preview,
2947         PreviewLarge
2948     }
2949 }
2950
2951     public class PreviewPanelEventArgs : EventArgs
2952 {
2953     public enum PreviewPanelType
2954     {
2955         Preview,
2956         PreviewLarge
2957     }
2958 }
2959
2960     public class PreviewPanelEventArgs : EventArgs
2961 {
2962     public enum PreviewPanelType
2963     {
2964         Preview,
2965         PreviewLarge
2966     }
2967 }
2968
2969     public class PreviewPanelEventArgs : EventArgs
2970 {
2971     public enum PreviewPanelType
2972     {
2973         Preview,
2974         PreviewLarge
2975     }
2976 }
2977
2978     public class PreviewPanelEventArgs : EventArgs
2979 {
2980     public enum PreviewPanelType
2981     {
2982         Preview,
2983         PreviewLarge
2984     }
2985 }
2986
2987     public class PreviewPanelEventArgs : EventArgs
2988 {
2989     public enum PreviewPanelType
2990     {
2991         Preview,
2992         PreviewLarge
2993     }
2994 }
2995
2996     public class PreviewPanelEventArgs : EventArgs
2997 {
2998     public enum PreviewPanelType
2999     {
2999         Preview,
3000         PreviewLarge
3000     }
3001 }
3002
3003     public class PreviewPanelEventArgs : EventArgs
3004 {
3005     public enum PreviewPanelType
3006     {
3007         Preview,
3008         PreviewLarge
3009     }
3010 }
3011
3012     public class PreviewPanelEventArgs : EventArgs
3013 {
3014     public enum PreviewPanelType
3015     {
3016         Preview,
3017         PreviewLarge
3018     }
3019 }
3020
3021     public class PreviewPanelEventArgs : EventArgs
3022 {
3023     public enum PreviewPanelType
3024     {
3025         Preview,
3026         PreviewLarge
3027     }
3028 }
3029
3030     public class PreviewPanelEventArgs : EventArgs
3031 {
3032     public enum PreviewPanelType
3033     {
3034         Preview,
3035         PreviewLarge
3036     }
3037 }
3038
3039     public class PreviewPanelEventArgs : EventArgs
3040 {
3041     public enum PreviewPanelType
3042     {
3043         Preview,
3044         PreviewLarge
3045     }
3046 }
3047
3048     public class PreviewPanelEventArgs : EventArgs
3049 {
3050     public enum PreviewPanelType
3051     {
3052         Preview,
3053         PreviewLarge
3054     }
3055 }
3056
3057     public class PreviewPanelEventArgs : EventArgs
3058 {
3059     public enum PreviewPanelType
3060     {
3061         Preview,
3062         PreviewLarge
3063     }
3064 }
3065
3066     public class PreviewPanelEventArgs : EventArgs
3067 {
3068     public enum PreviewPanelType
3069     {
3070         Preview,
3071         PreviewLarge
3072     }
3073 }
3074
3075     public class PreviewPanelEventArgs : EventArgs
3076 {
3077     public enum PreviewPanelType
3078     {
3079         Preview,
3080         PreviewLarge
3081     }
3082 }
3083
3084     public class PreviewPanelEventArgs : EventArgs
3085 {
3086     public enum PreviewPanelType
3087     {
3088         Preview,
3089         PreviewLarge
3090     }
3091 }
3092
3093     public class PreviewPanelEventArgs : EventArgs
3094 {
3095     public enum PreviewPanelType
3096     {
3097         Preview,
3098         PreviewLarge
3099     }
3100 }
3101
3102     public class PreviewPanelEventArgs : EventArgs
3103 {
3104     public enum PreviewPanelType
3105     {
3106         Preview,
3107         PreviewLarge
3108     }
3109 }
3110
3111     public class PreviewPanelEventArgs : EventArgs
3112 {
3113     public enum PreviewPanelType
3114     {
3115         Preview,
3116         PreviewLarge
3117     }
3118 }
3119
3120     public class PreviewPanelEventArgs : EventArgs
3121 {
3122     public enum PreviewPanelType
3123     {
3124         Preview,
3125         PreviewLarge
3126     }
3127 }
3128
3129     public class PreviewPanelEventArgs : EventArgs
3130 {
3131     public enum PreviewPanelType
3132     {
3133         Preview,
3134         PreviewLarge
3135     }
3136 }
3137
3138     public class PreviewPanelEventArgs : EventArgs
3139 {
3140     public enum PreviewPanelType
3141     {
3142         Preview,
3143         PreviewLarge
3144     }
3145 }
3146
3147     public class PreviewPanelEventArgs : EventArgs
3148 {
3149     public enum PreviewPanelType
3150     {
3151         Preview,
3152         PreviewLarge
3153     }
3154 }
3155
3156     public class PreviewPanelEventArgs : EventArgs
3157 {
3158     public enum PreviewPanelType
3159     {
3160         Preview,
3161         PreviewLarge
3162     }
3163 }
3164
3165     public class PreviewPanelEventArgs : EventArgs
3166 {
3167     public enum PreviewPanelType
3168     {
3169         Preview,
3170         PreviewLarge
3171     }
3172 }
3173
3174     public class PreviewPanelEventArgs : EventArgs
3175 {
3176     public enum PreviewPanelType
3177     {
3178         Preview,
3179         PreviewLarge
3180     }
3181 }
3182
3183     public class PreviewPanelEventArgs : EventArgs
3184 {
3185     public enum PreviewPanelType
3186     {
3187         Preview,
3188         PreviewLarge
3189     }
3190 }
3191
3192     public class PreviewPanelEventArgs : EventArgs
3193 {
3194     public enum PreviewPanelType
3195     {
3196         Preview,
3197         PreviewLarge
3198     }
3199 }
3200
3201     public class PreviewPanelEventArgs : EventArgs
3202 {
3203     public enum PreviewPanelType
3204     {
3205         Preview,
3206         PreviewLarge
3207     }
3208 }
3209
3210     public class PreviewPanelEventArgs : EventArgs
3211 {
3212     public enum PreviewPanelType
3213     {
3214         Preview,
3215         PreviewLarge
3216     }
3217 }
3218
3219     public class PreviewPanelEventArgs : EventArgs
3220 {
3221     public enum PreviewPanelType
3222     {
3223         Preview,
3224         PreviewLarge
3225     }
3226 }
3227
3228     public class PreviewPanelEventArgs : EventArgs
3229 {
3230     public enum PreviewPanelType
3231     {
3232         Preview,
3233         PreviewLarge
3234     }
3235 }
3236
3237     public class PreviewPanelEventArgs : EventArgs
3238 {
3239     public enum PreviewPanelType
3240     {
3241         Preview,
3242         PreviewLarge
3243     }
3244 }
3245
3246     public class PreviewPanelEventArgs : EventArgs
3247 {
3248     public enum PreviewPanelType
3249     {
3250         Preview,
3251         PreviewLarge
3252     }
3253 }
3254
3255     public class PreviewPanelEventArgs : EventArgs
3256 {
3257     public enum PreviewPanelType
3258     {
3259         Preview,
3260         PreviewLarge
3261     }
3262 }
3263
3264     public class PreviewPanelEventArgs : EventArgs
3265 {
3266     public enum PreviewPanelType
3267     {
3268         Preview,
3269         PreviewLarge
3270     }
3271 }
3272
3273     public class PreviewPanelEventArgs : EventArgs
3274 {
3275     public enum PreviewPanelType
3276     {
3277         Preview,
3278         PreviewLarge
3279     }
3280 }
3281
3282     public class PreviewPanelEventArgs : EventArgs
3283 {
3284     public enum PreviewPanelType
3285     {
3286         Preview,
3287         PreviewLarge
3288     }
3289 }
3290
3291     public class PreviewPanelEventArgs : EventArgs
3292 {
3293     public enum PreviewPanelType
3294     {
3295         Preview,
3296         PreviewLarge
3297     }
3298 }
3299
3300     public class PreviewPanelEventArgs : EventArgs
3301 {
3302     public enum PreviewPanelType
3303     {
3304         Preview,
3305         PreviewLarge
3306     }
3307 }
3308
3309     public class PreviewPanelEventArgs : EventArgs
3310 {
3311     public enum PreviewPanelType
3312     {
3313         Preview,
3314         PreviewLarge
3315     }
3316 }
3317
3318     public class PreviewPanelEventArgs : EventArgs
3319 {
3320     public enum PreviewPanelType
3321     {
3322         Preview,
3323         PreviewLarge
3324     }
3325 }
3326
3327     public class PreviewPanelEventArgs : EventArgs
3328 {
3329     public enum PreviewPanelType
3330     {
3331         Preview,
3332         PreviewLarge
3333     }
3334 }
3335
3336     public class PreviewPanelEventArgs : EventArgs
3337 {
3338     public enum PreviewPanelType
3339     {
3340         Preview,
3341         PreviewLarge
3342     }
3343 }
3344
3345     public class PreviewPanelEventArgs : EventArgs
3346 {
3347     public enum PreviewPanelType
3348     {
3349         Preview,
3350         PreviewLarge
3351     }
3352 }
3353
3354     public class PreviewPanelEventArgs : EventArgs
3355 {
3356     public enum PreviewPanelType
3357     {
3358         Preview,
3359         PreviewLarge
3360     }
3361 }
3362
3363     public class PreviewPanelEventArgs : EventArgs
3364 {
3365     public enum PreviewPanelType
3366     {
3367         Preview,
3368         PreviewLarge
3369     }
3370 }
3371
3372     public class PreviewPanelEventArgs : EventArgs
3373 {
3374     public enum PreviewPanelType
3375     {
3376         Preview,
3377         PreviewLarge
3378     }
3379 }
3380
3381     public class PreviewPanelEventArgs : EventArgs
3382 {
3383     public enum PreviewPanelType
3384     {
3385         Preview,
3386         PreviewLarge
3387     }
3388 }
3389
3390     public class PreviewPanelEventArgs : EventArgs
3391 {
3392     public enum PreviewPanelType
3393     {
3394         Preview,
3395         PreviewLarge
3396     }
3397 }
3398
3399     public class PreviewPanelEventArgs : EventArgs
3400 {
3401     public enum PreviewPanelType
3402     {
3403         Preview,
3404         PreviewLarge
3405     }
3406 }
3407
3408     public class PreviewPanelEventArgs : EventArgs
3409 {
3410     public enum PreviewPanelType
3411     {
3412         Preview,
3413         PreviewLarge
3414     }
3415 }
3416
3417     public class PreviewPanelEventArgs : EventArgs
3418 {
3419     public enum PreviewPanelType
3420     {
3421         Preview,
3422         PreviewLarge
3423     }
3424 }
3425
3426     public class PreviewPanelEventArgs : EventArgs
3427 {
3428     public enum PreviewPanelType
3429     {
3430         Preview,
3431         PreviewLarge
3432     }
3433 }
3434
3435     public class PreviewPanelEventArgs : EventArgs
3436 {
3437     public enum PreviewPanelType
3438     {
3439         Preview,
3440         PreviewLarge
3441     }
3442 }
3443
3444     public class PreviewPanelEventArgs : EventArgs
3445 {
3446     public enum PreviewPanelType
3447     {
3448         Preview,
3449         PreviewLarge
3450     }
3451 }
3452
3453     public class PreviewPanelEventArgs : EventArgs
3454 {
3455     public enum PreviewPanelType
3456     {
3457         Preview,
3458         PreviewLarge
3459     }
3460 }
3461
3462     public class PreviewPanelEventArgs : EventArgs
3463 {
3464     public enum PreviewPanelType
3465     {
3466         Preview,
3467         PreviewLarge
3468     }
3469 }
3470
3471     public class PreviewPanelEventArgs : EventArgs
3472 {
3473     public enum PreviewPanelType
3474     {
3475         Preview,
3476         PreviewLarge
3477     }
3478 }
3479
3480     public class PreviewPanelEventArgs : EventArgs
3481 {
3482     public enum PreviewPanelType
3483     {
3484         Preview,
3485         PreviewLarge
3486     }
3487 }
3488
3489     public class PreviewPanelEventArgs : EventArgs
3490 {
3491     public enum PreviewPanelType
3492     {
3493         Preview,
3494         PreviewLarge
3495     }
3496 }
3497
3498     public class PreviewPanelEventArgs : EventArgs
3499 {
3500     public enum PreviewPanelType
3501     {
3502         Preview,
3503         PreviewLarge
3504     }
3505 }
3506
3507     public class PreviewPanelEventArgs : EventArgs
3508 {
3509     public enum PreviewPanelType
3510     {
3511         Preview,
3512         PreviewLarge
3513     }
3514 }
3515
3516     public class PreviewPanelEventArgs : EventArgs
3517 {
3518     public enum PreviewPanelType
3519     {
3520         Preview,
3521         PreviewLarge
3522     }
3523 }
3524
3525     public class PreviewPanelEventArgs : EventArgs
3526 {
3527     public enum PreviewPanelType
3528     {
3529         Preview,
3530         PreviewLarge
3531     }
3532 }
3533
3534     public class PreviewPanelEventArgs : EventArgs
3535 {
3536     public enum PreviewPanelType
3537     {
3538         Preview,
3539         PreviewLarge
3540     }
3541 }
3542
3543     public class PreviewPanelEventArgs : EventArgs
3544 {
3545     public enum PreviewPanelType
3546     {
3547         Preview,
3548         PreviewLarge
3549     }
3550 }
3551
3552     public class PreviewPanelEventArgs : EventArgs
3553 {
3554     public enum PreviewPanelType
3555     {
3556         Preview,
3557         PreviewLarge
3558     }
3559 }
3560
3561     public class PreviewPanelEventArgs : EventArgs
3562 {
3563     public enum PreviewPanelType
3564     {
3565         Preview,
3566         PreviewLarge
3567     }
3568 }
3569
3570     public class PreviewPanelEventArgs : EventArgs
3571 {
3572     public enum PreviewPanelType
3573     {
3574         Preview,
3575         PreviewLarge
3576     }
3577 }
3578
3579     public class PreviewPanelEventArgs : EventArgs
3580 {
3581     public enum PreviewPanelType
3582     {
3583         Preview,
3584         PreviewLarge
3585     }
3586 }
3587
3588     public class PreviewPanelEventArgs : EventArgs
3589 {
3590     public enum PreviewPanelType
3591     {
3592         Preview,
3593         PreviewLarge
3594     }
3595 }
3596
3597     public class PreviewPanelEventArgs : EventArgs
3598 {
3599     public enum PreviewPanelType
3600     {
3601         Preview,
3602         PreviewLarge
3603     }
3604 }
3605
3606     public class PreviewPanelEventArgs : EventArgs
3607 {
3608     public enum PreviewPanelType
3609     {
3610         Preview,
3611         PreviewLarge
3612     }
3613 }
3614
3615     public class PreviewPanelEventArgs : EventArgs
3616 {
3617     public enum PreviewPanelType
3618     {
3619         Preview,
3620         PreviewLarge
3621     }
3622 }
3623
3624     public class PreviewPanelEventArgs : EventArgs
3625 {
3626     public enum PreviewPanelType
3627     {
3628         Preview,
3629         PreviewLarge
3630     }
3631 }
3632
3633     public class PreviewPanelEventArgs : EventArgs
3634 {
3635     public enum PreviewPanelType
3636     {
3637         Preview,
3638         PreviewLarge
3639     }
3640 }
3641
3642     public class PreviewPanelEventArgs : EventArgs
3643 {
3644     public enum PreviewPanelType
3645     {
3646         Preview,
3647         PreviewLarge
3648     }
3649 }
3650
3651     public class PreviewPanelEventArgs : EventArgs
3652 {
3653     public enum PreviewPanelType
3654     {
3655         Preview,
3
```

Pasos siguientes

Para continuar su recorrido, visite las [líneas fases de la página de códigos](#).

Vea también

- [Experiencia de Git en Visual Studio](#)
- [GitHub de Visual Studio & : mejor juntos](#) ↗

Inserción desde Visual Studio en una rama remota

Artículo • 28/02/2023 • Tiempo de lectura: 2 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

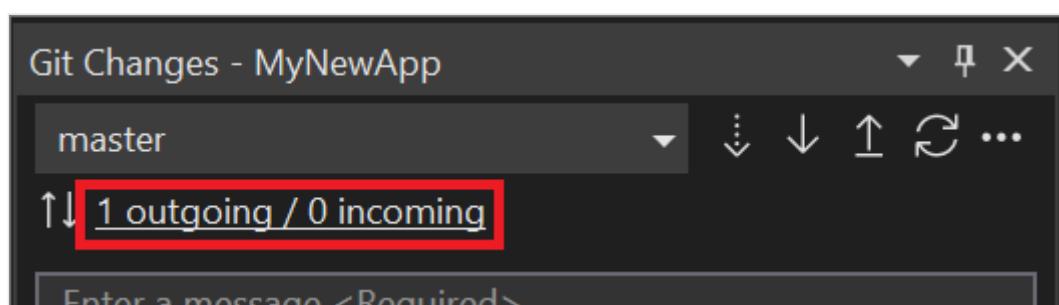
Una vez autenticado en GitHub, Visual Studio puede mejorar el flujo de trabajo de GitHub. Una de estas mejoras consiste en la posibilidad de insertar (también denominado publicar) un proyecto local directamente en GitHub con un solo clic. La fase final de un flujo de trabajo de Git simple consiste en enviar los cambios al repositorio remoto.

Un repositorio remoto es un lugar seguro para almacenar el código en la nube. Normalmente se conoce como **origin/main** (u **origin/master**), donde "origin" es el nombre predeterminado de un repositorio remoto. Para obtener más información sobre esta terminología, consulte la página [Creación de ramas en Git: ramas remotas](#) en el sitio web de Git.

Aunque este artículo hace referencia a repositorios de GitHub, puede trabajar de forma remota con el proveedor de Git que prefiera, como GitHub, GitLab o Azure DevOps.

A continuación se describe cómo insertar en un repositorio remoto en Visual Studio.

1. Asegúrese de que tiene un archivo abierto para trabajar en él en un repositorio previamente [creado](#) o [clonado](#).
2. Realice un cambio en el archivo, guárdelo, seleccione la pestaña **Git Changes** (Cambios de Git) y [confirme](#) el cambio.
3. En la ventana **Git Changes** (Cambios de Git), observe el texto del vínculo que incluye el número de confirmaciones entrantes y salientes. En el ejemplo siguiente, el texto del vínculo indica **1 saliente/0 entrante**.



El texto "saliente" representa el número de confirmaciones que aún no se han insertado en el repositorio remoto, mientras que "entrante" representa las

confirmaciones que se han capturado, pero que aún no se han extraído del repositorio remoto.

4. Para insertar en el repositorio remoto, seleccione el botón **Insertar** o seleccione **Insertar** en el menú de **Git**.

Pasos siguientes

Para continuar el recorrido, visite la página [Captura, extracción y sincronización en Visual Studio](#).

Consulte también

- [Experiencia de Git en Visual Studio](#)
- [Visual Studio y GitHub mejor juntos ↗](#)

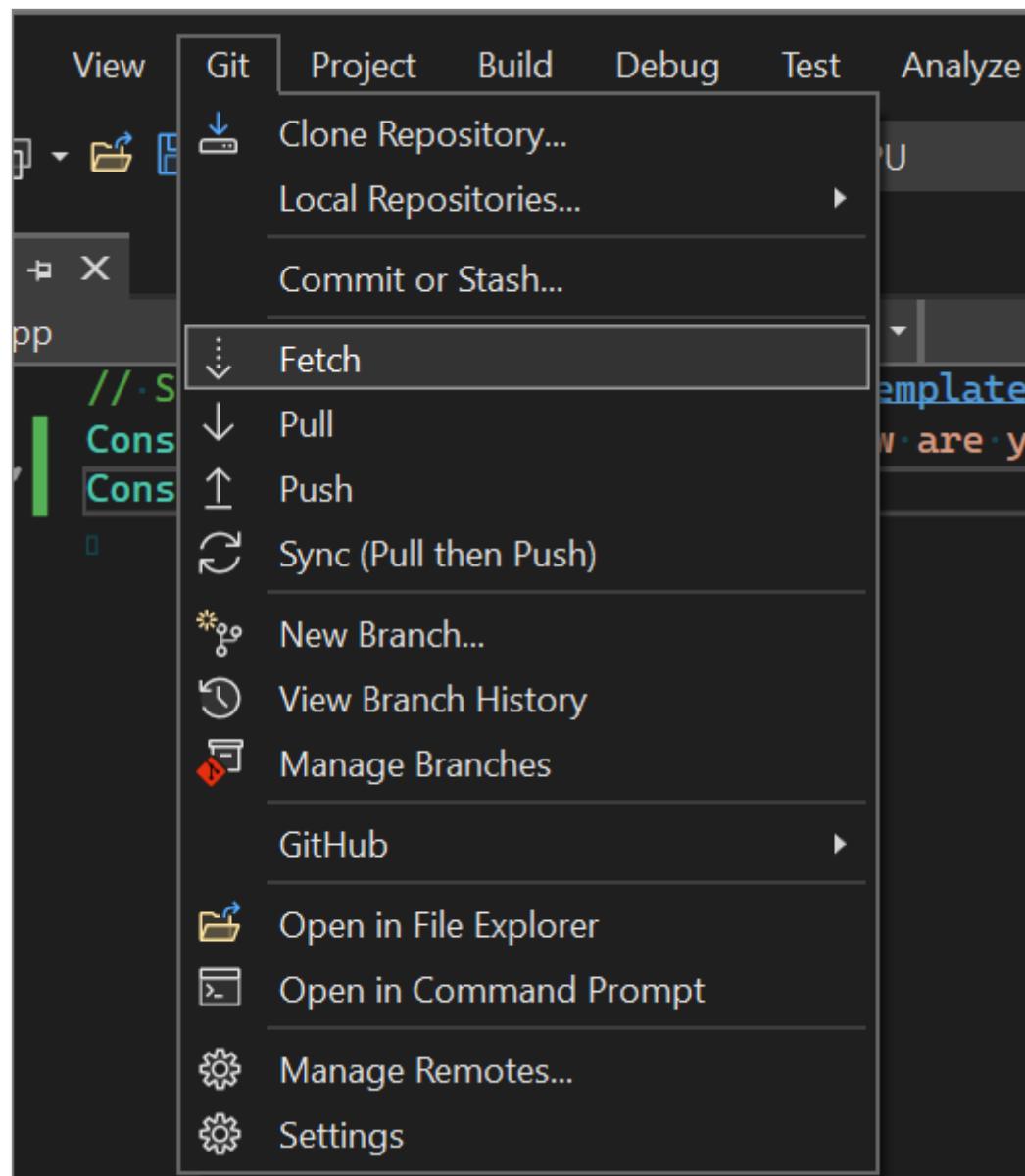
Uso de la captura, extracción, inserción y sincronización de Git para el control de versiones en Visual Studio

Artículo • 09/03/2023 • Tiempo de lectura: 2 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

Visual Studio le ayuda a mantener la rama local sincronizada con la rama remota por medio de las operaciones de descarga (recuperación y extracción) y de carga (inserción).

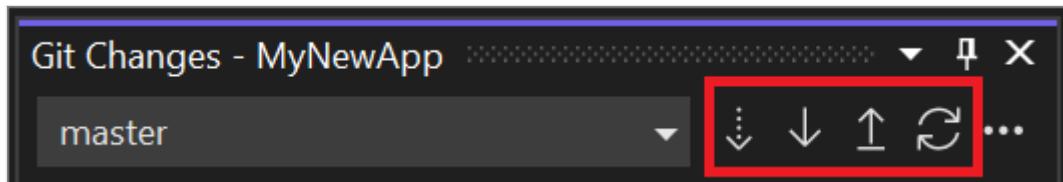
Puede capturar, extraer y sincronizar en Visual Studio 2022 mediante el menú **Git**.



En la captura de pantalla anterior, la opción **Capturar** está resaltada. El menú **Git** también incluye las siguientes opciones adicionales:

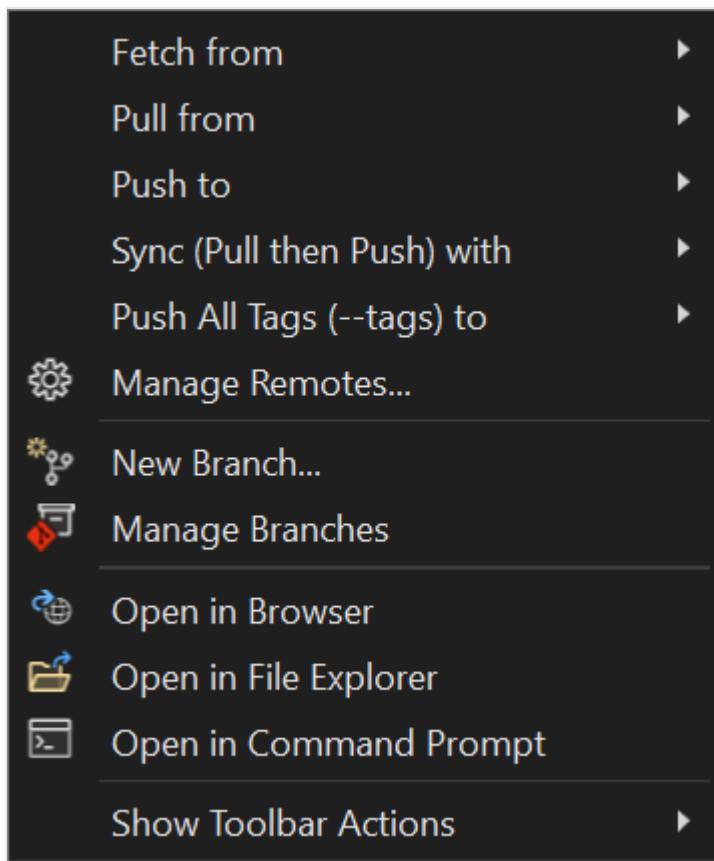
- Extracción
- Push
- Sincronización (extracción y después inserción)

También puede usar los controles de botón en la ventana **Cambios de Git** para realizar estas operaciones.



De izquierda a derecha, los controles de botón incluyen **Capturar**, **Extraer**, **Insertar** y **Sincronizar**.

Además, también hay un control de botón de **puntos suspensivos (...)** para operaciones adicionales. Al seleccionarlo, aparece un menú contextual. Puede usarlo para ajustar las operaciones de captura, extracción, inserción y sincronización.



Capturar

Es importante capturar y extraer antes de insertar. La recuperación comprueba si hay confirmaciones remotas que deban incorporarse a los cambios locales. Si ve alguna, realice primero la extracción para evitar conflictos de combinación ascendentes.

Al capturar una rama, la ventana **Cambios de Git** presenta un indicador en la lista desplegable Rama, que muestra el número de confirmaciones no extraídas de la rama remota. Este indicador también muestra el número de confirmaciones locales sin insertar.

El indicador también funciona como un vínculo que le lleva al historial de confirmaciones de esa rama en la ventana **Repositorio de GIT**. En la parte superior del historial se muestran los detalles de estas confirmaciones entrantes y salientes. Desde aquí, también puede optar por extraer o insertar las confirmaciones.

Extracción

Extraiga siempre antes de insertar. Al extraer primero, puede evitar [conflictos de combinación](#) ascendentes.

Inserción

Al crear confirmaciones, ha guardado intrínsecamente instantáneas locales del código. Use **Insertar** para insertar las confirmaciones en GitHub, donde puede almacenarlas como copias de seguridad o compartir el código con otros usuarios.

Pero, como se mencionó anteriormente, extraiga siempre antes de insertar. Como medida de seguridad, Visual Studio no permite insertar confirmaciones si la rama local está detrás de la rama remota. Si intenta insertar, un cuadro de diálogo le pide que extraiga antes de hacerlo.

Sync

Use esta operación para extraer y, después, insertar, secuencialmente.

Pasos siguientes

Para continuar el recorrido, visite la página [Examinar repositorios de Git](#).

Vea también

- [Tutorial: Abrir un proyecto desde un repositorio](#)
- [Visual Studio y GitHub: mejor juntos ↗](#)

Examinar repositorios de Git y comparar ramas en Visual Studio

Artículo • 27/09/2022 • Tiempo de lectura: 5 minutos

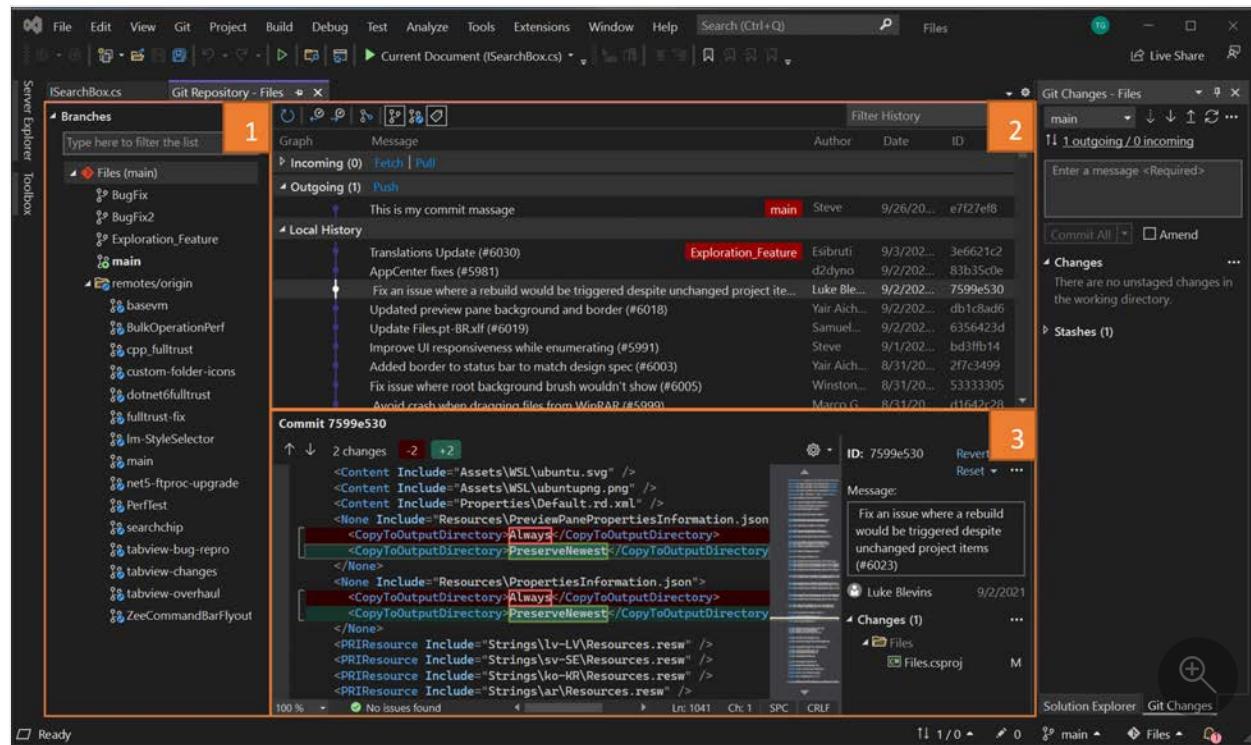
Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

La ventana **Cambios de Git** proporciona una manera sin problemas de interactuar con Git mientras se codifica sin tener que cambiar del código. Pero hay ocasiones en las que tiene más sentido centrarse en el repositorio de Git. Por ejemplo, es posible que tenga que obtener una buena imagen de lo que su equipo ha estado trabajando o quizás [comparar dos confirmaciones](#) para investigar un error.

Puede trabajar de forma remota con el proveedor de Git que prefiera, como GitHub o Azure DevOps.

Examinar ramas locales y remotas

Para empezar, abra la ventana **Repositorio de Git**; para ello, seleccione **Repositorio de Git** en el menú **Ver**. También puede acceder a la ventana **Repositorio de Git** seleccionando los vínculos **salientes** o **entrantes** en la ventana **Cambios de Git** y en la barra de estado.



La ventana **Repositorio de Git** contiene tres secciones principales, como se numera en la captura de pantalla anterior:

1. **Ramas**: Git permite a los usuarios realizar varias tareas y experimentar con su código a través de ramas. Si está trabajando en varias características al mismo tiempo o si desea explorar ideas sin afectar al código de trabajo, la bifurcación puede resultar útil.

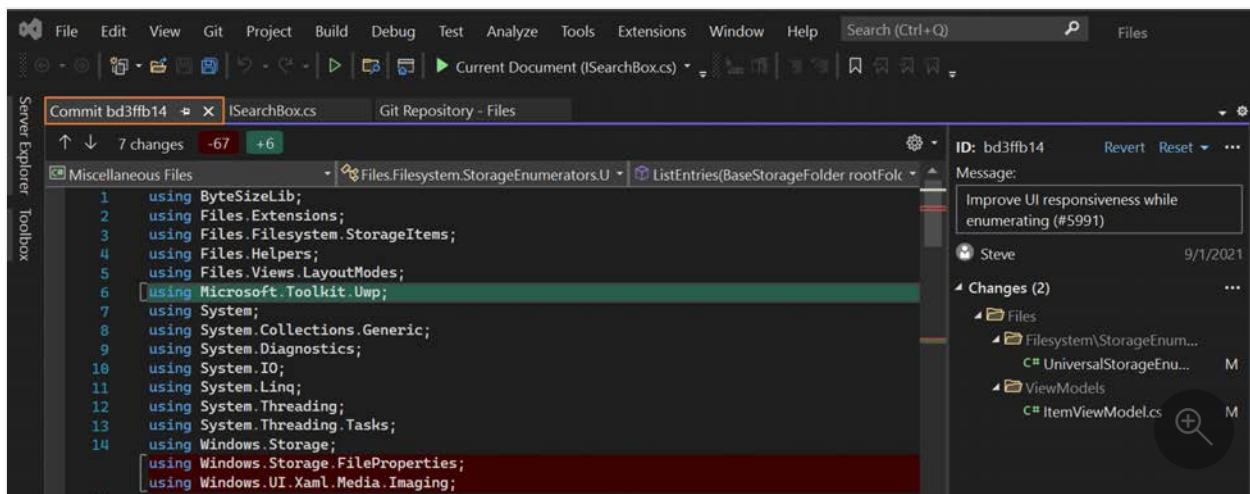
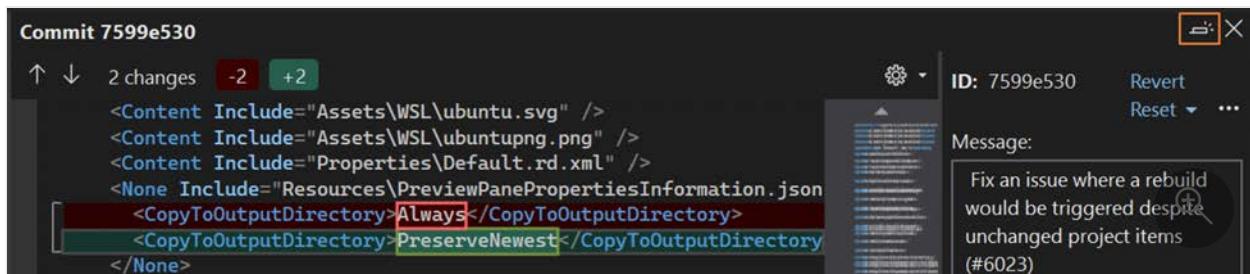
2. **Graph**: en esta sección se visualiza el estado de la rama. Tiene tres subsecciones:

- **Entrante** muestra confirmaciones entrantes que el equipo ha contribuido.
- **Saliente** muestra las confirmaciones locales que todavía no ha insertado.
- **El historial local** muestra el resto de confirmaciones de las que realiza el seguimiento del repositorio local.

3. **Confirmar**: al seleccionar cualquier confirmación en la sección **Graph** se abren sus detalles. Para comprobar los cambios que ha introducido una confirmación, selecciónelos, lo que muestra una diferencia. Por ejemplo, en la captura de pantalla anterior se muestran los cambios que se han introducido en el archivo *Files.csproj*.

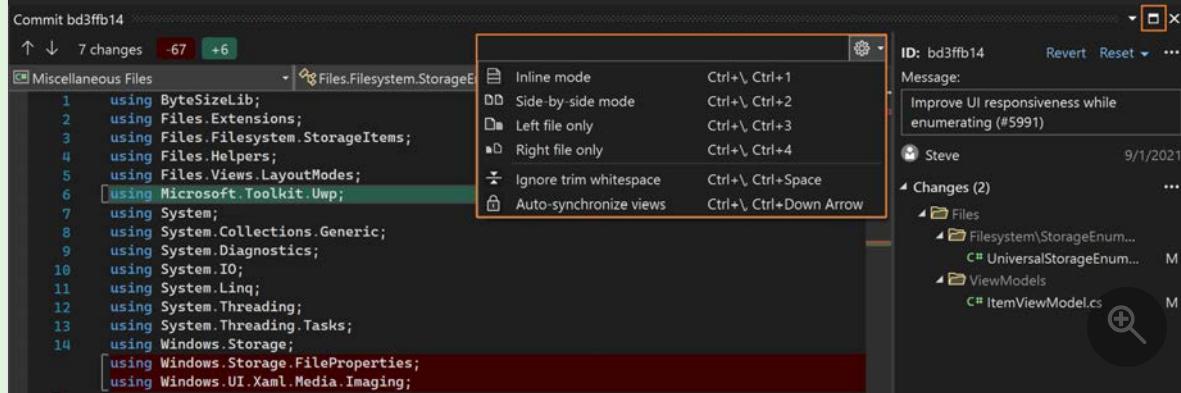
Los métodos abreviados de teclado Alt+Flecha arriba o Alt+Flecha abajo le permiten saltar entre estas secciones.

Puede examinar cualquier rama local o remota sin tener que cambiar la rama. Cuando encuentre una confirmación en la que desea centrarse, seleccione el botón **Abrir en nueva pestaña** para abrir la confirmación en otra pestaña.



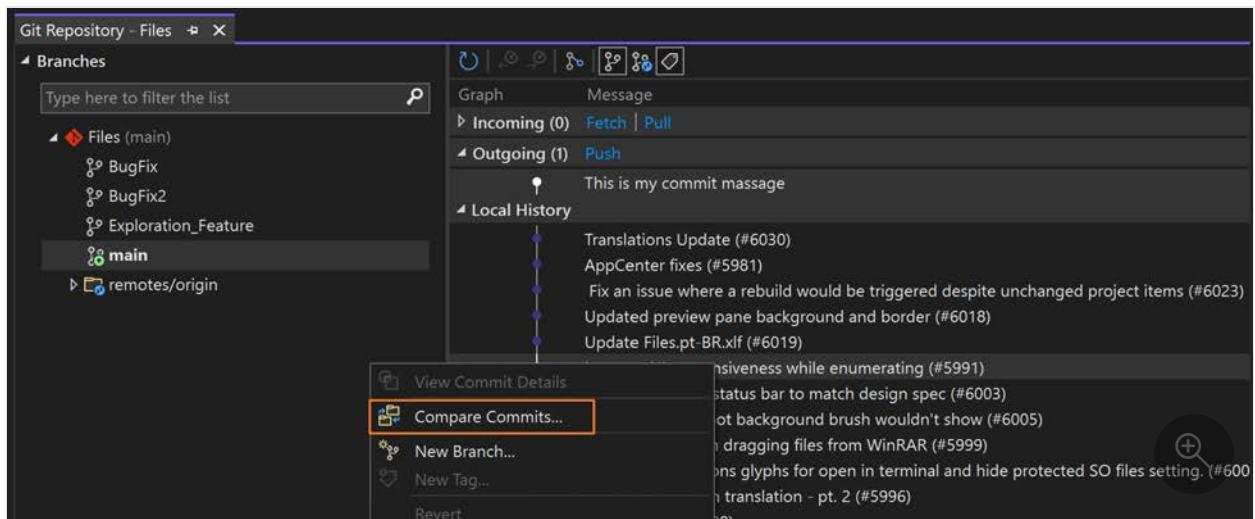
💡 Sugerencia

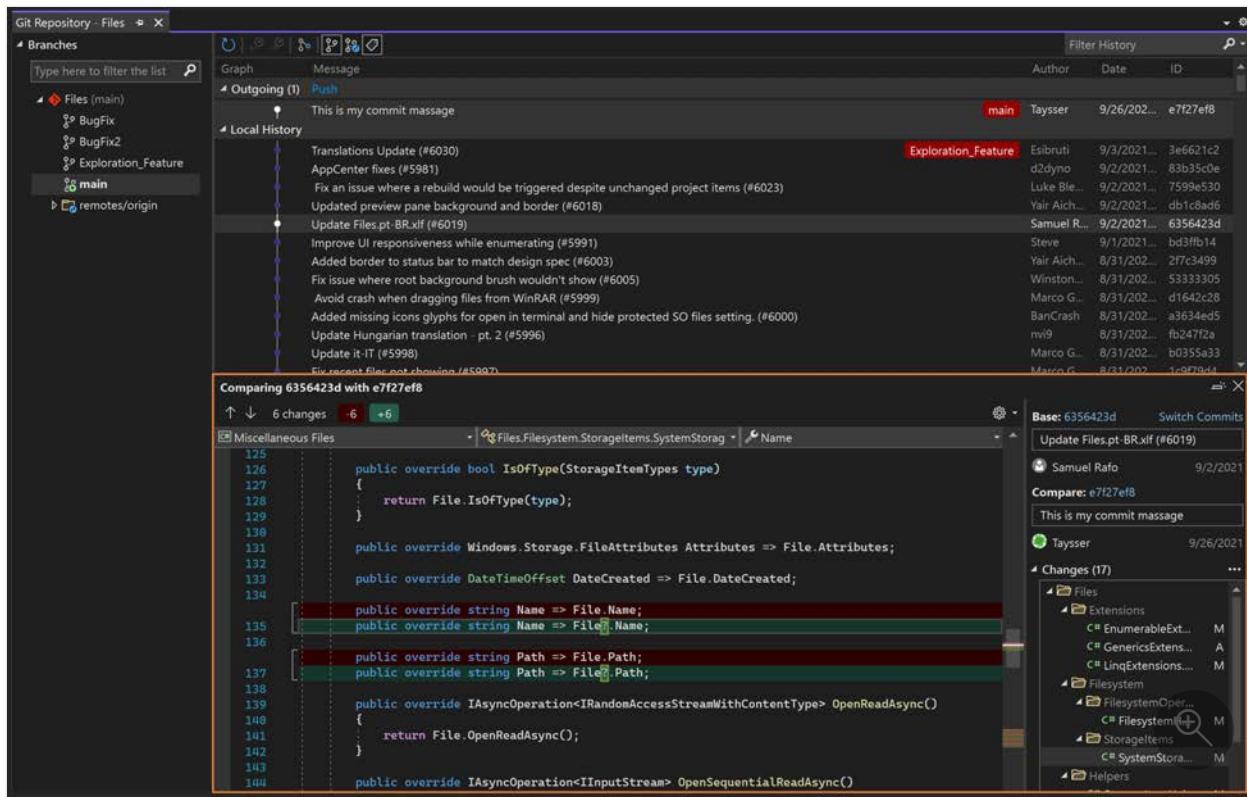
Para mostrar la confirmación en pantalla completa, desasocie la pestaña **Confirmar** y maximice la ventana **Confirmar** mediante el botón **Maximizar**. También puede seleccionar su configuración de diferencias favorita seleccionando **Configuración de diferencias** (el ícono de engranaje).



Comparación de confirmaciones

Para comparar dos confirmaciones en la rama, use la tecla **Ctrl** para seleccionar las dos confirmaciones que desea comparar. A continuación, haga clic con el botón derecho en uno de ellos y seleccione **Comparar confirmaciones**.



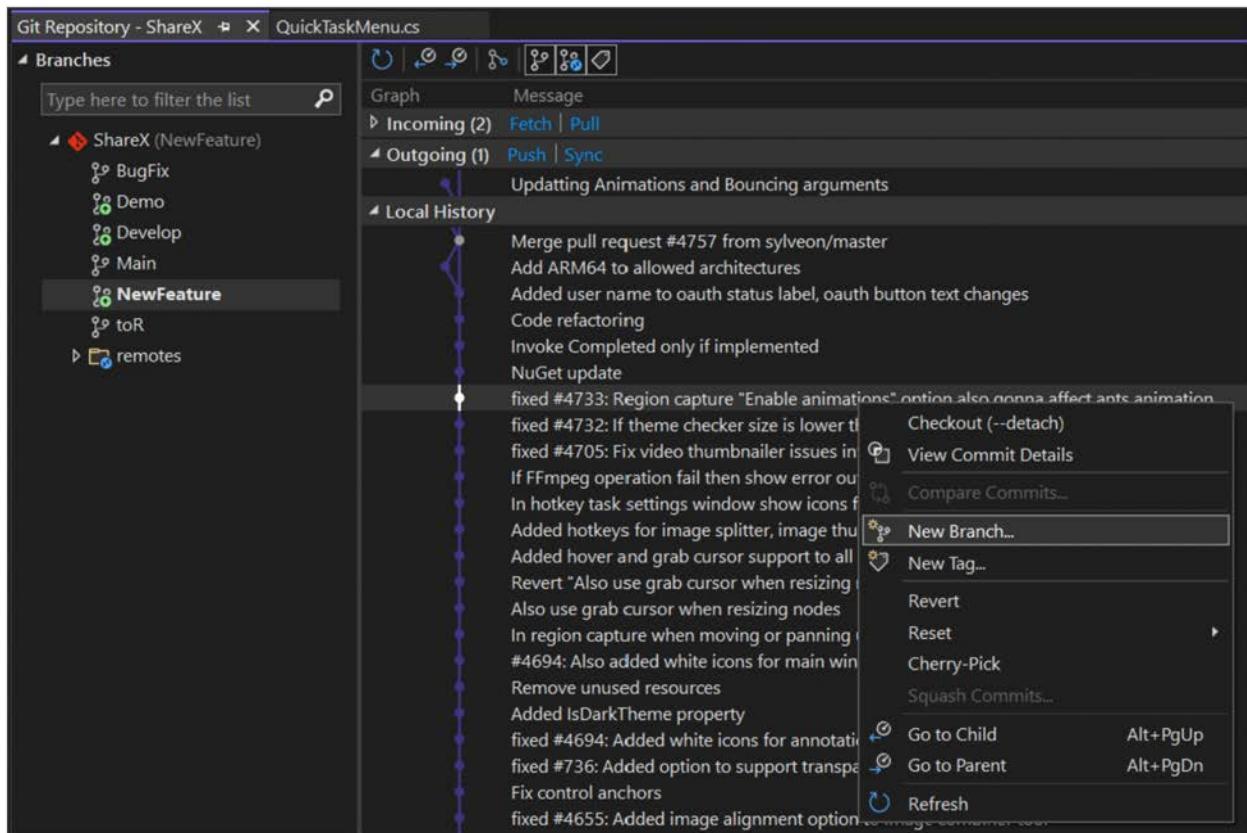


Sugerencia

De forma similar a **Los detalles** de confirmación, puede usar el botón **Abrir en nueva pestaña** para abrir la comparación en otra pestaña o maximizarla en la pantalla.

Creación de una rama a partir de una confirmación

En Visual Studio, puede usar el panel **git Graph** en la ventana **Repositorio de Git** para crear ramas a partir de confirmaciones anteriores. Para ello, haga clic con el botón derecho en la confirmación desde la que desea crear una nueva rama y, a continuación, seleccione **Nueva rama**.

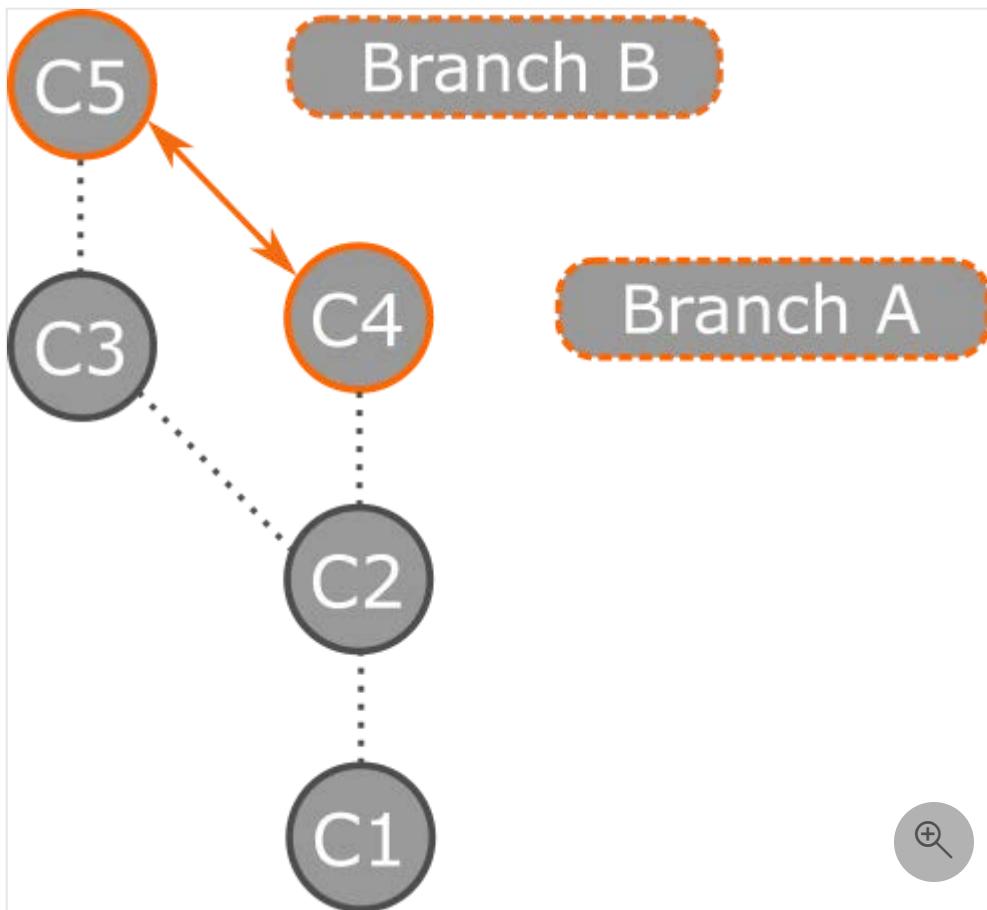


ⓘ Nota

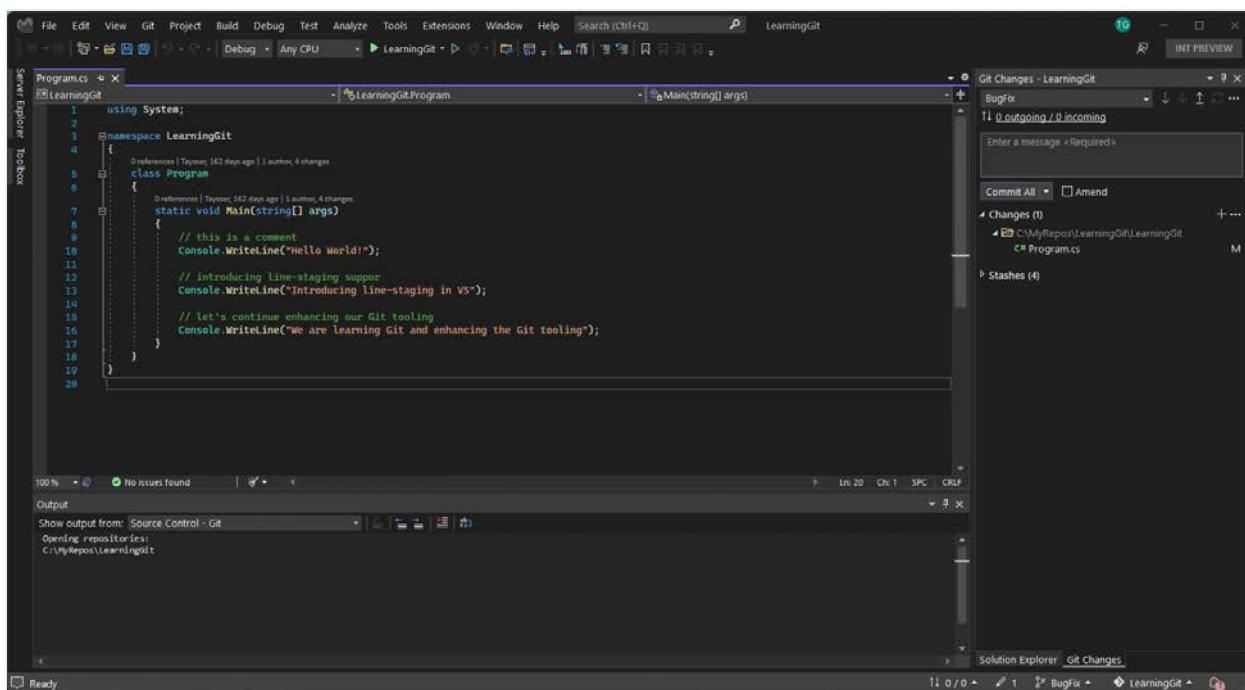
El comando equivalente para esta acción es `git branch <branchname> [<commit-id>]`.

Comparar ramas

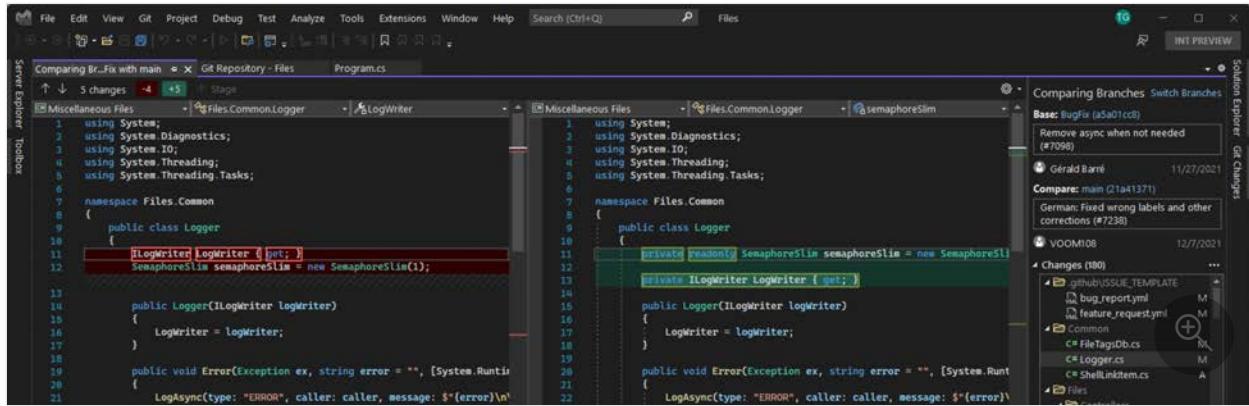
La comparación de ramas proporciona información general sobre las diferencias entre dos ramas que pueden resultar muy útiles antes de crear una solicitud de incorporación de cambios, combinar o incluso eliminar una rama.



Para comparar la rama desprotegida actualmente con otras ramas mediante Visual Studio, puede usar el selector de ramas hospedado en la barra de estado y la ventana de herramientas cambios de Git para elegir cualquier rama local o remota con la que comparar. Haga clic con el botón derecho en la rama de destino y seleccione **Comparar con la rama actual**. Como alternativa, puede usar la lista de ramas en la ventana Repositorio de Git para acceder al mismo comando.

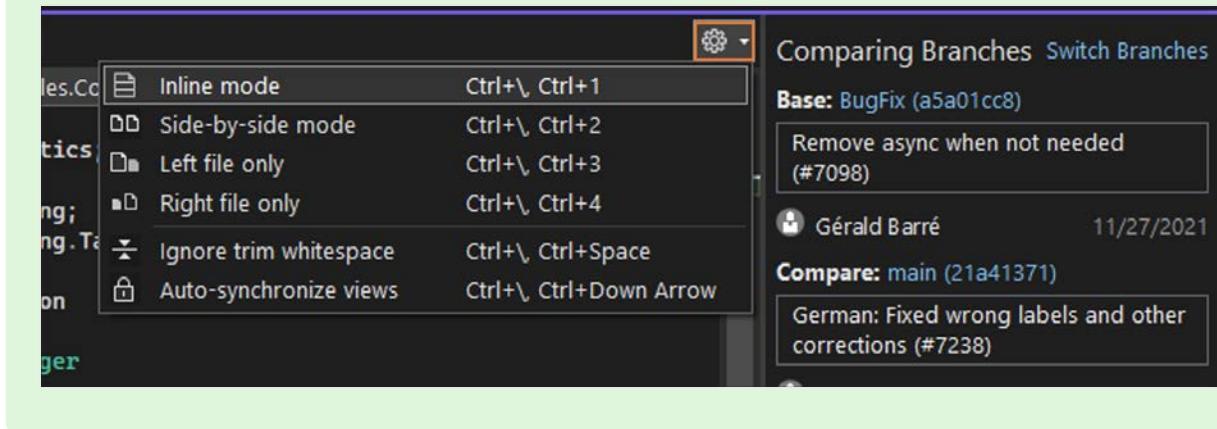


Al seleccionar **Comparar con la rama actual**, se abre la experiencia de comparación de ramas, donde puede navegar por la lista **Cambios** y seleccionar el archivo que desea comparar.



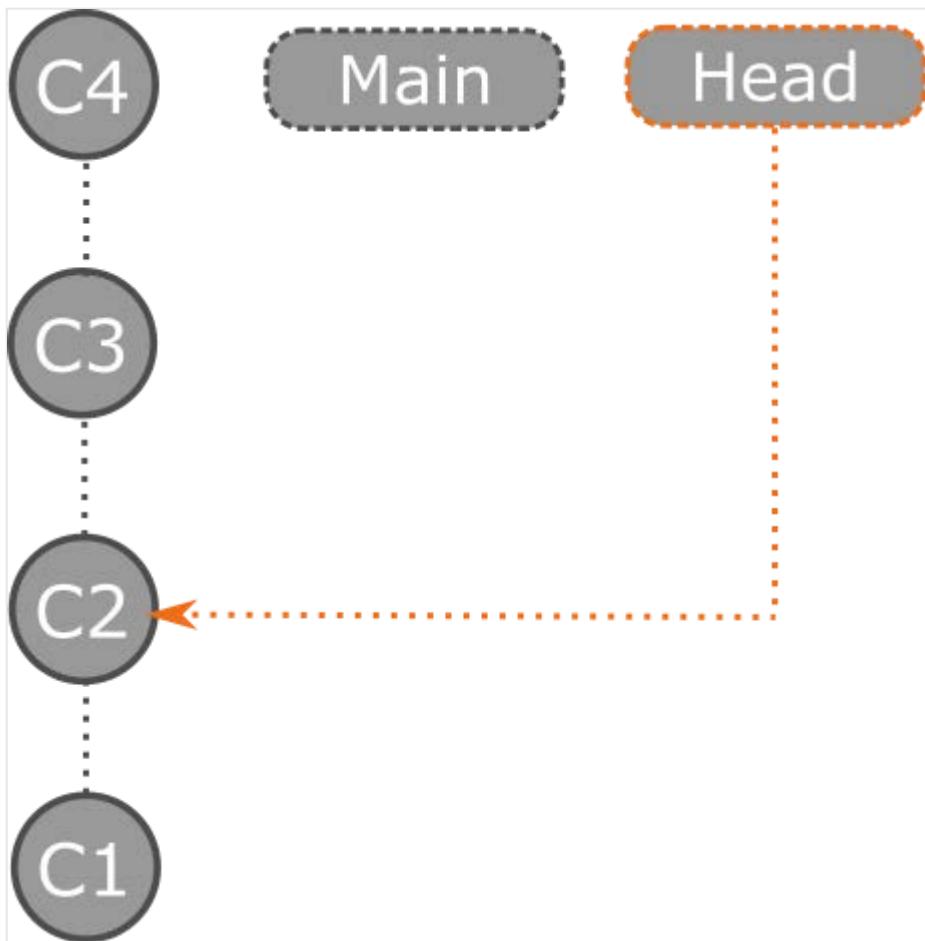
💡 Sugerencia

Si prefiere una diferencia entre líneas, puede usar el engranaje de opciones de configuración de diferencias y cambiar a una vista de diferencias en línea.

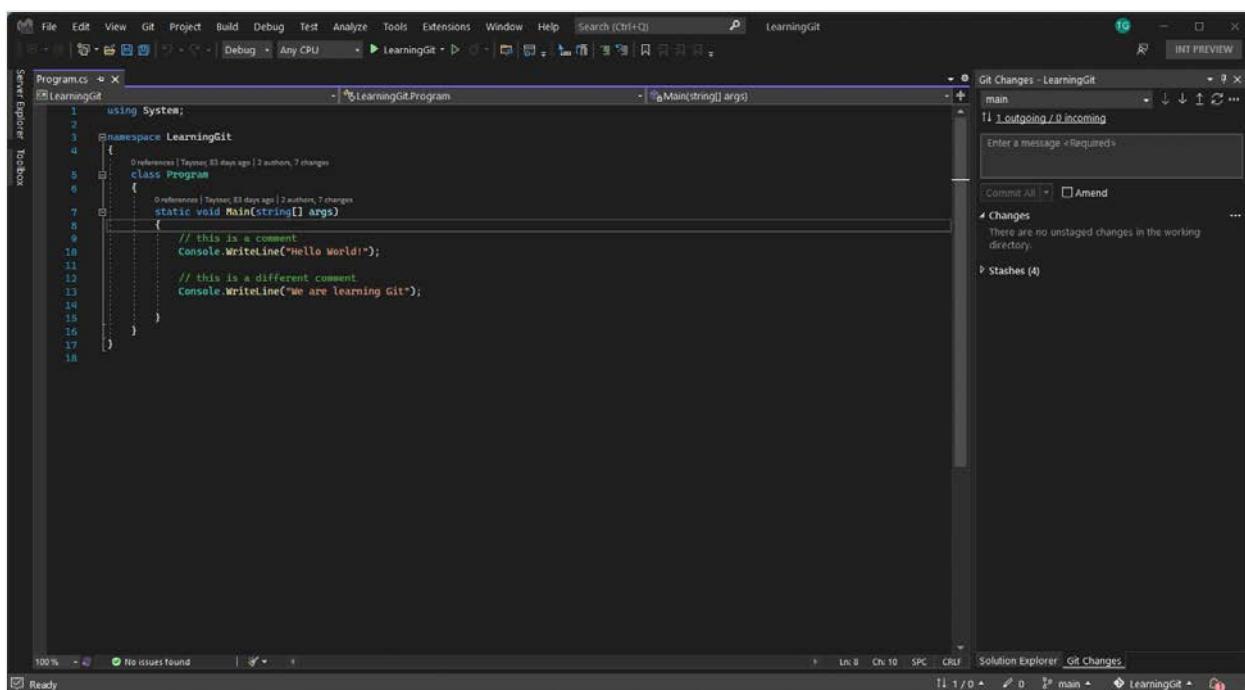


Confirmaciones de desprotección

La comprobación de una confirmación puede ser beneficiosa de varias maneras. Por ejemplo, permite volver a un punto anterior en el historial del repositorio, donde puede ejecutar o probar el código. También puede resultar útil si desea revisar el código de una rama remota (por ejemplo, la rama de un compañero). De este modo, no es necesario crear una rama local si no planea contribuir a ella. En este caso, puede simplemente desproteger la propina de la rama remota que le gustaría revisar.



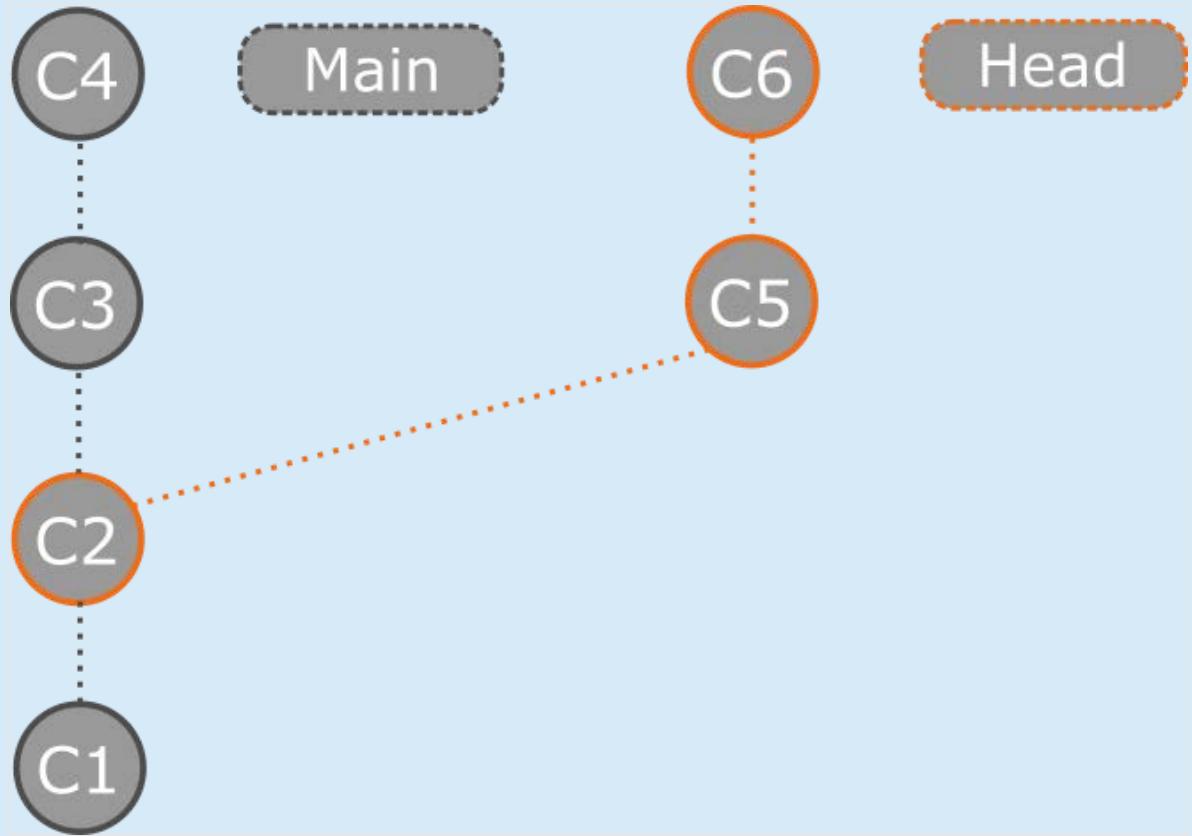
Para desproteger una confirmación anterior en Visual Studio, abra la ventana **Repositorio de Git**, haga clic con el botón derecho en la confirmación a la que desea volver y seleccione **Desproteger (-desasociar)**. Visual Studio muestra un cuadro de diálogo de confirmación que explica que, al desproteger una confirmación, estará en un estado HEAD desasociado. Lo que significa que head del repositorio apunta directamente a una confirmación en lugar de a una rama.



Ahora que está en un estado principal desasociado, no dude en ejecutar y probar el código, o incluso explorar y confirmar los cambios. Cuando haya terminado de explorar y desea volver a la rama, puede optar por descartar los cambios comprobando una rama existente o seleccionando conservar los cambios [creando primero una nueva rama](#).

Importante

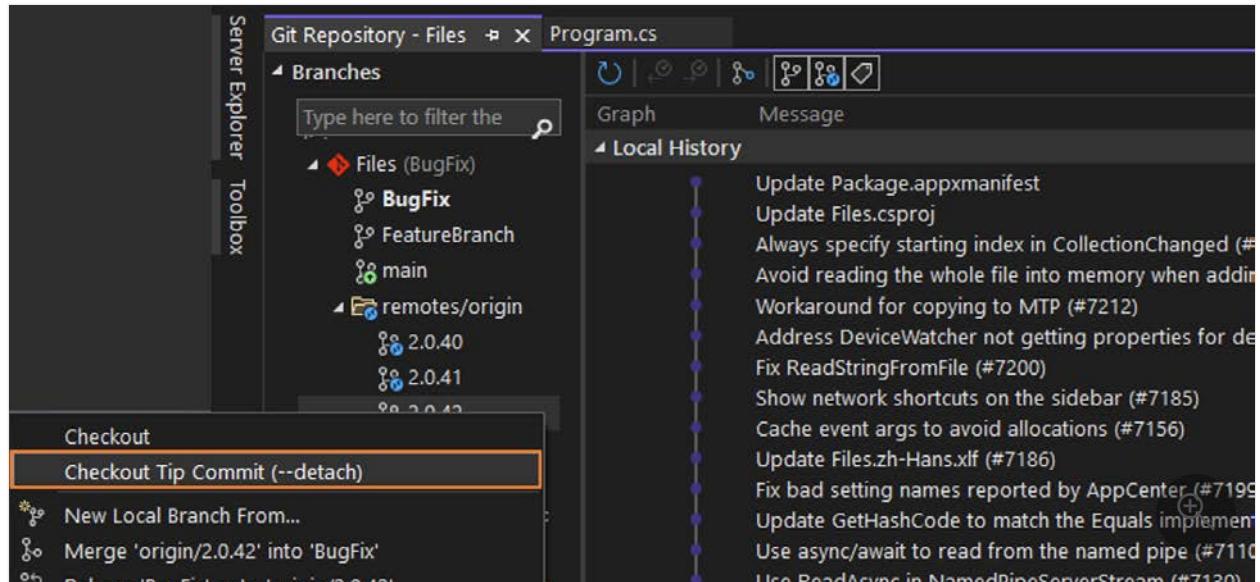
Las confirmaciones creadas en un estado principal desasociado no están asociadas a ninguna rama y Git puede recopilar elementos no utilizados después de desproteger una rama. Es por eso que se conservan los cambios, se recomienda crear una rama antes de extraer una rama. Por ejemplo, las confirmaciones C5 y C6 se recopilarán como elementos no utilizados si desprotegeremos Main sin crear una nueva rama.



Para obtener más información sobre la vista de estado principal desasociado, consulte la siguiente documentación de Git: [Head desasociado](#)

La sugerencia de una rama remota puede resultar útil si desea revisar rápidamente una solicitud de incorporación de cambios y evaluar las actualizaciones más recientes. Para hacerlo en Visual Studio, primero asegúrese de capturar y obtener las actualizaciones más recientes del repositorio remoto. A continuación, haga clic con el botón derecho en la rama remota que desea revisar y seleccione **Confirmar sugerencia de finalización de**

la compra.



Pasos siguientes

Para continuar con el recorrido, consulte [Administración de repositorios de Git en Visual Studio](#).

Consulte también

- [La experiencia Git en Visual Studio](#)
- [Visual Studio y GitHub: Mejor juntos ↗](#)

Administración de repositorios de Git en Visual Studio

Artículo • 28/02/2023 • Tiempo de lectura: 6 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

La ventana **Repositorio de GIT** proporciona una experiencia de Git de pantalla completa que ayuda a administrar el repositorio de Git y mantenerse al día con los proyectos del equipo. Por ejemplo, es posible que tenga que restablecer, revertir o realizar una selección exclusiva de las confirmaciones, o simplemente limpiar el historial de confirmaciones. La ventana **Repositorio de GIT** también es un lugar excelente para visualizar y administrar las ramas.

El control de versiones con Visual Studio es sencillo con Git. Además, puede trabajar de forma remota con el proveedor de Git que prefiera, como GitHub o Azure DevOps. También puede trabajar en modo local sin ningún proveedor.

Cambio en la última confirmación (rectificación)

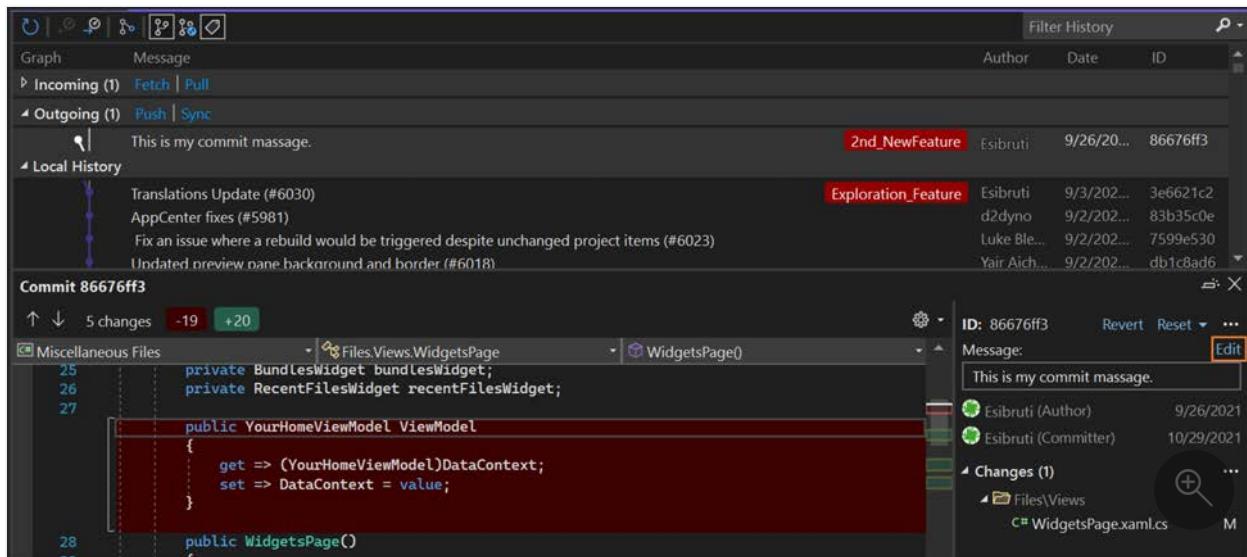
En Git, un caso de uso habitual consiste en actualizar la última confirmación, lo que se conoce como *rectificar*. A veces, simplemente necesita actualizar el mensaje de confirmación o incluir un cambio de última hora.

Puede rectificar una confirmación en la línea de comandos mediante el comando siguiente:

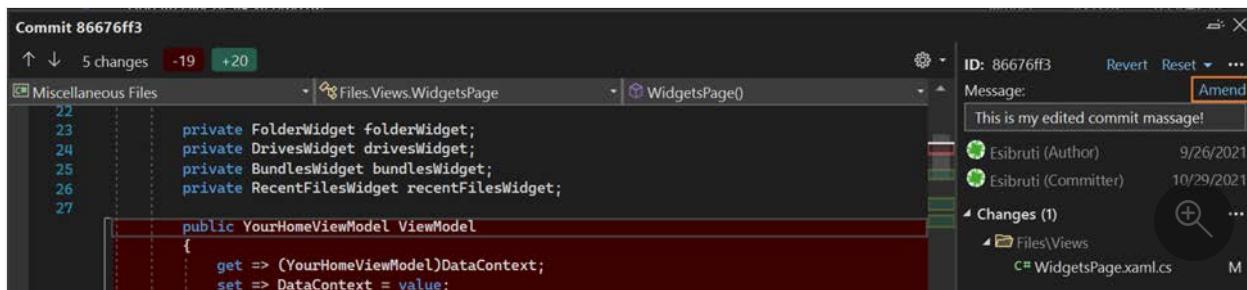
Bash

```
git commit --amend
```

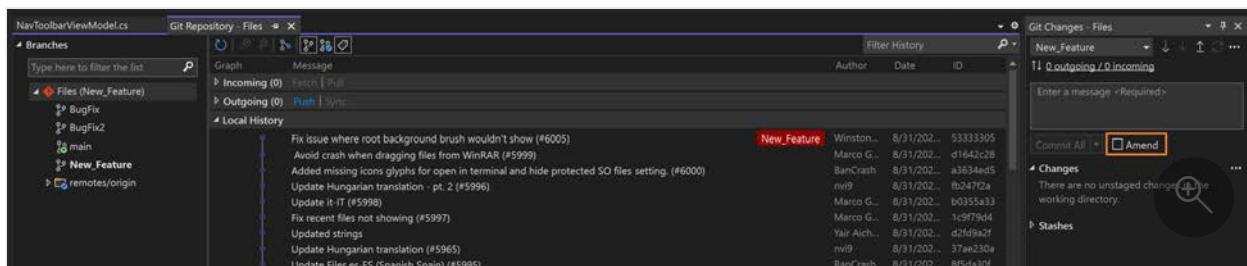
La ventana **Repositorio de GIT** facilita el proceso de actualizar el mensaje de confirmación. Haga doble clic en la última confirmación para abrir los detalles y seleccione la opción **Editar** que aparece junto al mensaje de confirmación.



Cuando acabe de editar el mensaje de confirmación, seleccione **Amend** (Rectificar).



Si necesita incluir cambios de código en la última confirmación, puede hacerlo en la ventana **Git Changes** (Cambios de Git). Active la casilla **Amend** (Rectificar) y confirme los cambios.



Para obtener más información sobre el proceso de rectificación, consulte [Herramientas de Git: reescritura del historial](#) en el sitio web de Git.

Confirmaciones de combinación (fusión mediante combinación con "squash")

Para combinar una serie de confirmaciones, Git proporciona una opción para fusionar mediante combinación con "squash" en una sola confirmación. Esta opción puede resultar útil si realiza confirmaciones frecuentes y acaba con una larga lista de

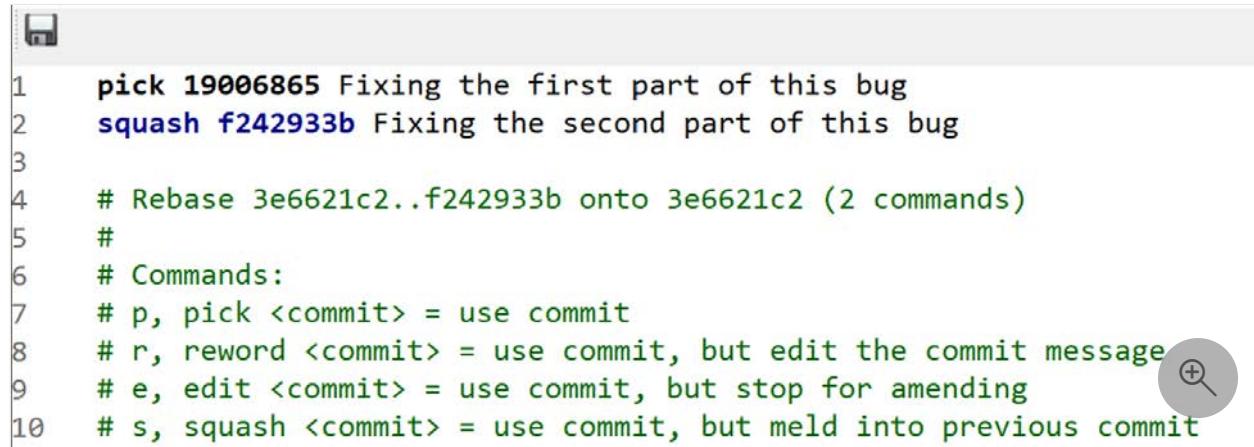
confirmaciones que quiere limpiar antes de realizar una inserción en un repositorio remoto.

Puede fusionar mediante combinación con "squash" dos confirmaciones en la línea de comandos mediante el comando siguiente:

Bash

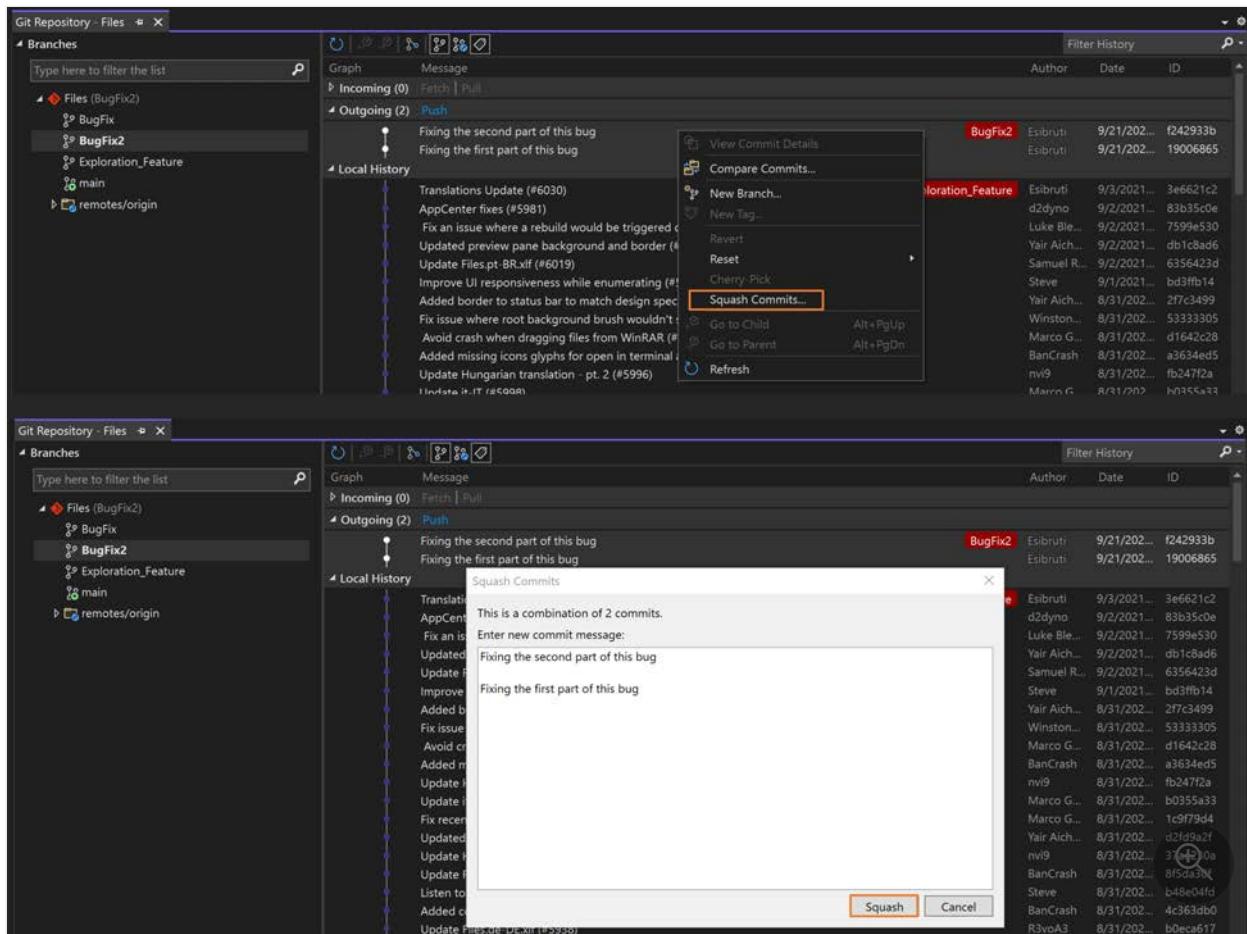
```
git rebase -i HEAD~2
```

Después, actualice `pick` a `squash`, guárdelo y actualice el mensaje de confirmación.



```
1 pick 19006865 Fixing the first part of this bug
2 squash f242933b Fixing the second part of this bug
3
4 # Rebase 3e6621c2..f242933b onto 3e6621c2 (2 commands)
5 #
6 # Commands:
7 # p, pick <commit> = use commit
8 # r, reword <commit> = use commit, but edit the commit message
9 # e, edit <commit> = use commit, but stop for amending
10 # s, squash <commit> = use commit, but meld into previous commit
```

Para combinar confirmaciones en Visual Studio, use la tecla **CTRL** para seleccionar varias confirmaciones que quiera combinar. Después, haga clic con el botón derecho y seleccione **Confirmaciones de squash**. Visual Studio combina automáticamente los mensajes de confirmación, pero a veces es mejor proporcionar un mensaje actualizado. Después de revisar y actualizar el mensaje de confirmación, seleccione el botón **Squash**.



Para obtener más información sobre la fusión mediante combinación con "squash", consulte [Herramientas de Git: reescritura del historial](#) en el sitio web de Git.

Combinación y fusión mediante cambio de base de las ramas

Si usa ramas de Git para trabajar en diferentes características, en algún momento tendrá que incluir actualizaciones introducidas en otras ramas. Esto puede ocurrir mientras todavía está trabajando en la rama de características. También podría suceder cuando ha acabado de trabajar en la rama de características y necesita conservar los cambios agregándolos a otra rama. En Git, puede incluir estas actualizaciones gracias a la combinación o la fusión mediante cambio de base de las ramas.

(!) Nota

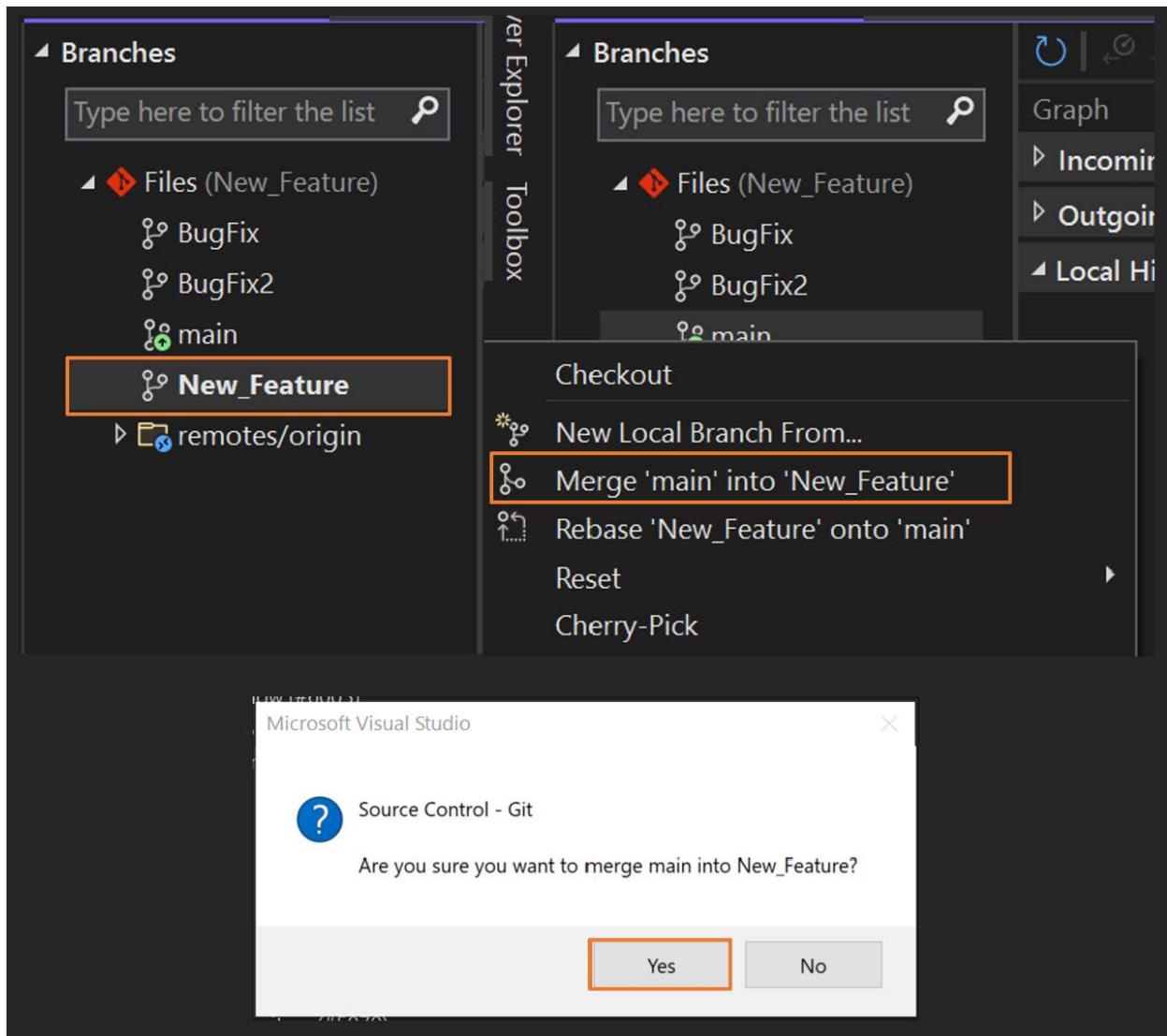
En las instrucciones siguientes se usa *New_Feature* como nombre de ejemplo para una rama de características. Reemplácelo por el nombre de su rama.

Para combinar la rama principal (`main`) con la rama de características en la línea de comandos, use los comandos siguientes:

Bash

```
git checkout New_Feature  
git merge main
```

Para hacer lo mismo en Visual Studio, en la lista de ramas, haga doble clic en la rama de características para extraerla del repositorio. Después, haga clic con el botón derecho en **main** y seleccione **Merge 'main' into 'New_Feature'** (Combinar "main" con "New_Feature").

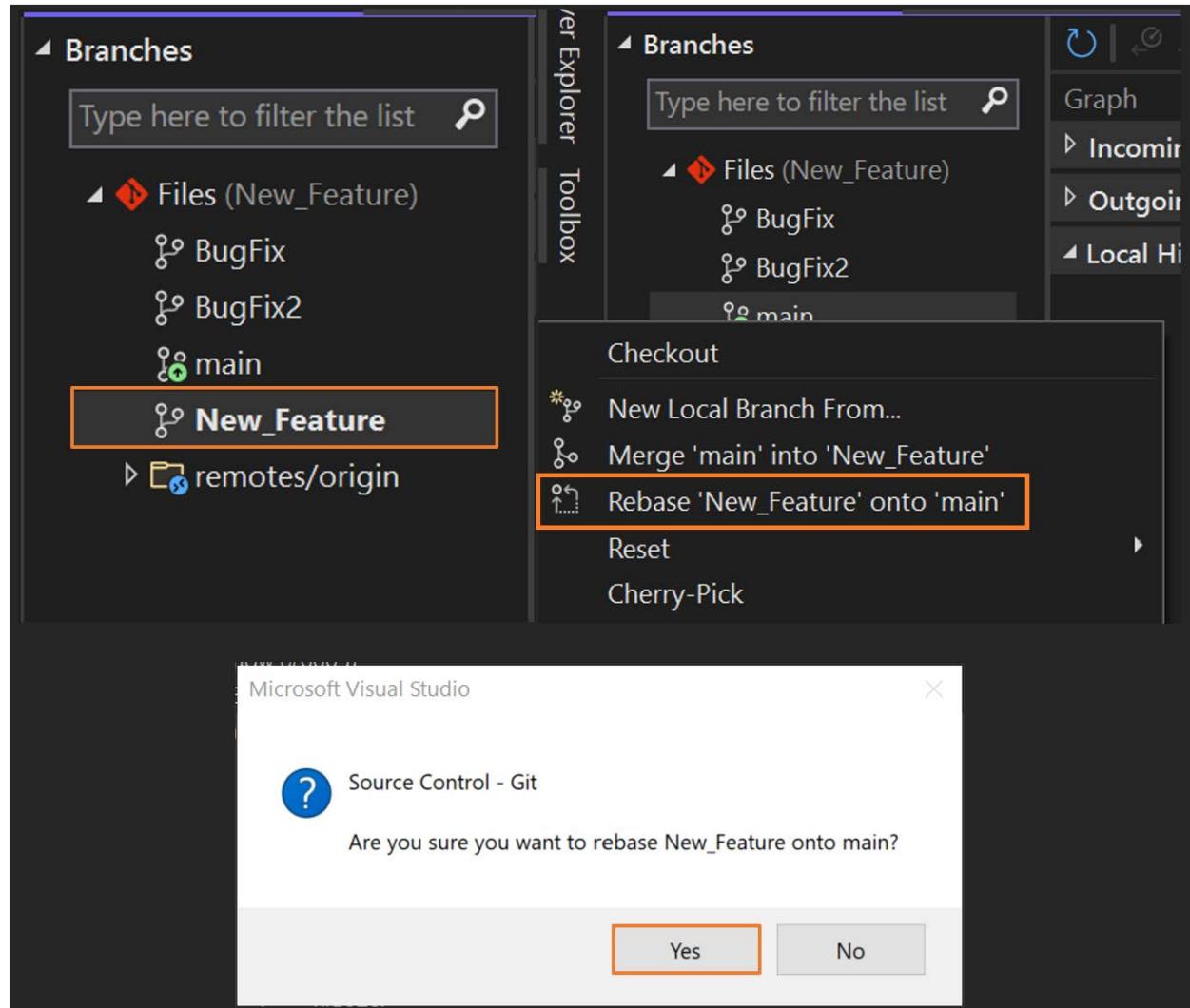


Para fusionar mediante cambio de base la rama principal (main) con la rama de características en la línea de comandos, use los comandos siguientes:

Bash

```
git checkout New_Feature  
git rebase main
```

Para hacer lo mismo en Visual Studio, en la lista de ramas, haga doble clic en la rama de características para extraerla del repositorio. Después, haga clic con el botón derecho en **main** y seleccione **Rebase 'New_Feature' onto 'main'** (Fusionar mediante cambio de base "New_Feature" en "main").



Para obtener más información sobre la combinación, la fusión mediante cambio de base y la creación de ramas en general, consulte [Creación de ramas en Git](#) en el sitio web de Git.

Copia de confirmaciones (selección exclusiva)

Copie confirmaciones de una rama a otra mediante la opción de selección exclusiva. A diferencia de una combinación o una fusión mediante cambio de base, la selección exclusiva solo aporta los cambios de las confirmaciones que seleccione, en lugar de todos los cambios de una rama. La selección exclusiva es una manera excelente de abordar estos problemas comunes:

- Realizar accidentalmente una confirmación en la rama incorrecta. Lleve a cabo una selección exclusiva de los cambios en la rama correcta y, luego, restablezca la rama

original a la confirmación anterior.

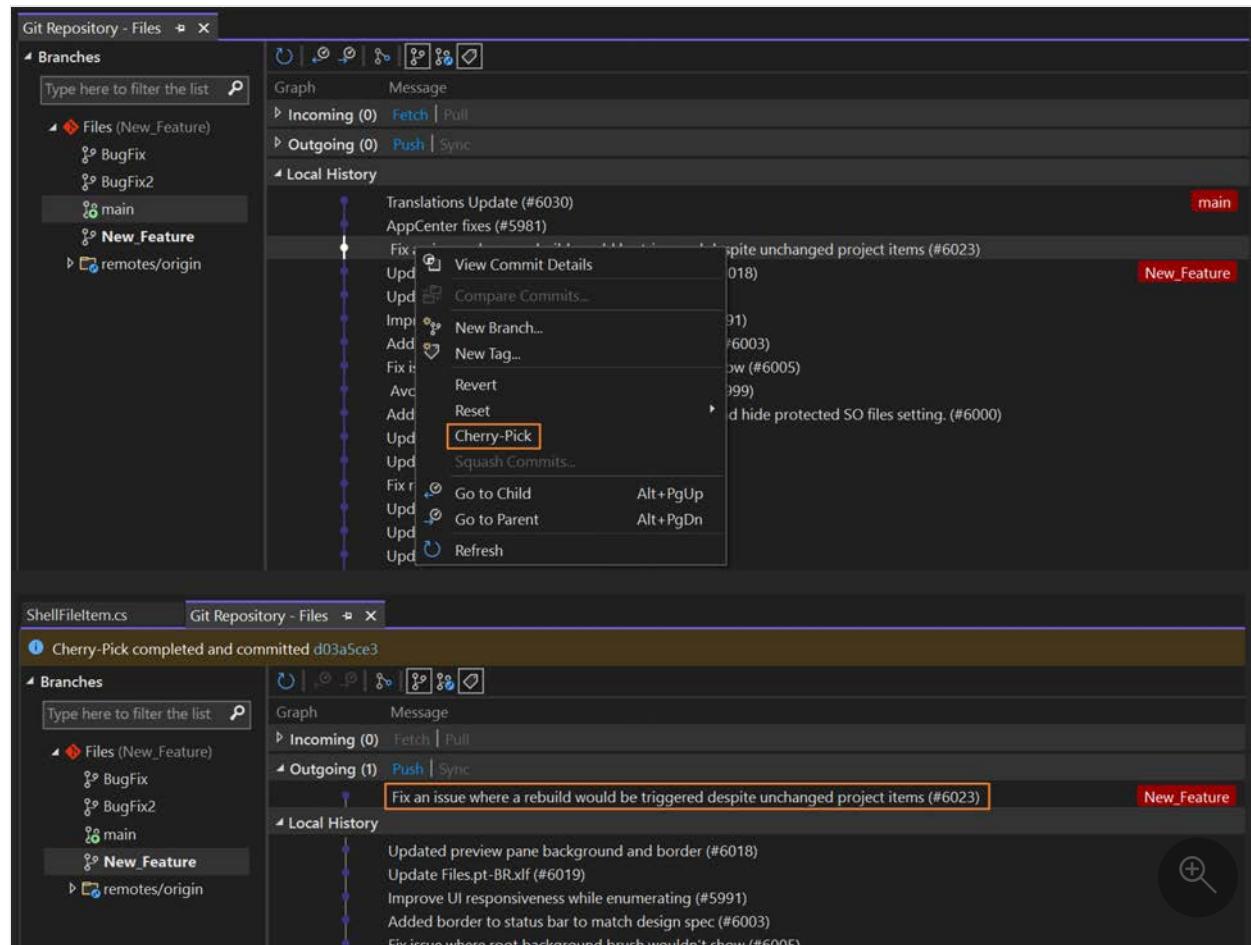
- Extraer un conjunto de confirmaciones realizadas en una rama de características, para volver a combinarlas antes en la rama principal.
- Portar confirmaciones específicas desde la rama principal sin fusionar la rama mediante cambio de base.

Para copiar los cambios de una confirmación en la rama actual mediante la línea de comandos, use el comando siguiente:

Bash

```
git cherry-pick 7599e530
```

Para hacer lo mismo en Visual Studio, seleccione con un solo clic la rama en la que quiere realizar una selección exclusiva de una confirmación para obtener una vista previa. Después, haga clic con el botón derecho en la confirmación que le interesa y elija **Selección exclusiva**.



Cuando se complete la operación, verá en Visual Studio un mensaje que indica que la operación es correcta. La confirmación de la que haya realizado una selección exclusiva aparecerá en la sección **Saliente**.

Para obtener más información sobre la selección exclusiva de confirmaciones, consulte la [página web de Git sobre el comando de selección exclusiva](#).

Reversión de cambios

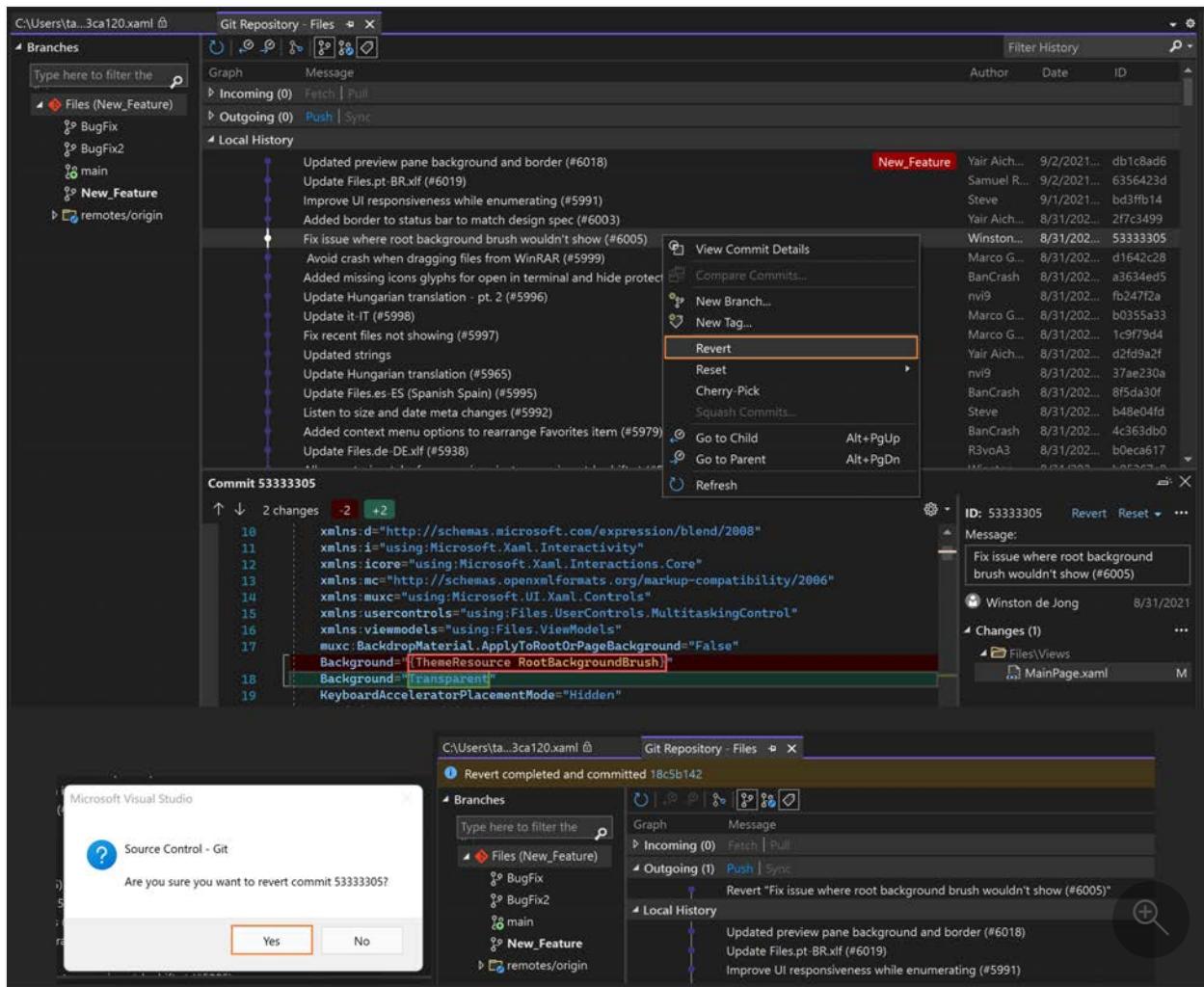
Use el comando "revert" para deshacer los cambios que se hayan realizado en confirmaciones insertadas en ramas compartidas. El comando "revert" crea una confirmación que deshace los cambios realizados en una confirmación anterior. El comando "revert" no reescribe el historial del repositorio, lo que hace que sea seguro usarlo al trabajar con otras personas.

Para revertir los cambios realizados en una confirmación mediante la línea de comandos, use los comandos siguientes. Reemplace el identificador de ejemplo por el identificador de una confirmación real en la rama.

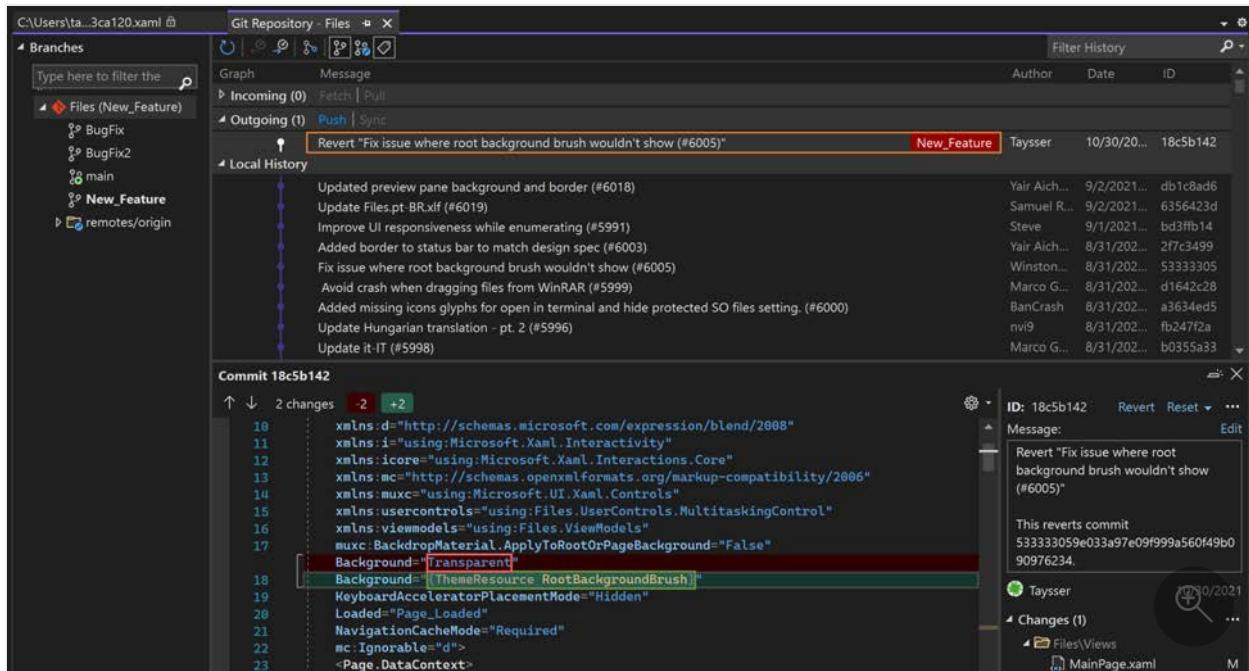
Bash

```
git revert 53333305  
git commit
```

En el ejemplo anterior, los comandos desharán los cambios realizados en la confirmación 53333305 y crearán una confirmación en la rama. La confirmación original todavía está en el historial de Git. Para hacer lo mismo en Visual Studio, haga clic con el botón derecho en la confirmación que quiere revertir y seleccione **Revertir**. Una vez que haya confirmado la acción y que la operación se haya completado, verá en Visual Studio un mensaje que indica que la operación es correcta y aparecerá una nueva confirmación en la sección **Saliente**.



Seleccione la nueva confirmación para confirmar que deshace los cambios de la confirmación revertida.



Para obtener más información sobre cómo revertir los cambios, consulte la página web de Git sobre el comando "revert" ↗.

Restablecimiento de una rama a un estado anterior

Use el comando "reset" para devolver una rama del repositorio local al contenido de una confirmación anterior. Esta acción descarta todos los cambios que se han producido desde la confirmación en la que está restableciendo la rama.

⚠️ Advertencia

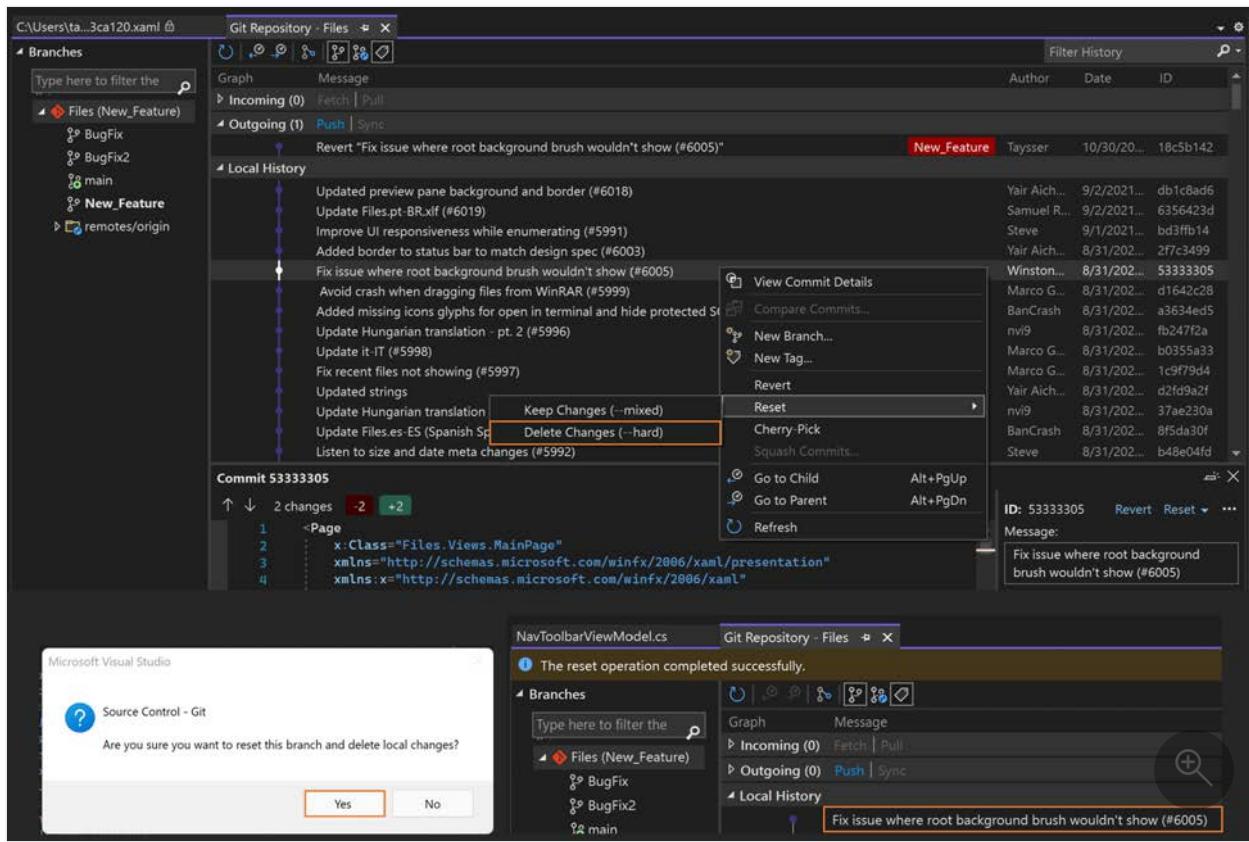
No restablezca las ramas compartidas, ya que podría eliminar el trabajo de otras personas. Use el comando "revert" en su lugar.

Para restablecer una rama a un estado anterior mediante la línea de comandos, use el comando siguiente. Reemplace el identificador de ejemplo por el identificador de una confirmación real en la rama.

Bash

```
git reset --hard 53333305
```

La parte `--hard` del comando le indica a Git que restablezca los archivos al estado de la confirmación anterior y que descarte los cambios "staged". Para hacer lo mismo en Visual Studio, haga clic con el botón derecho en la confirmación en la que quiere restablecer la rama y seleccione **Restablecer> Eliminar cambios (--hard)**.



Para obtener más información sobre el restablecimiento de ramas, consulte la [página web de Git sobre el comando "reset"](#).

Pasos siguientes

Para continuar el recorrido, consulte [Trabajo con varios repositorios](#).

Consulte también

- [La experiencia Git en Visual Studio](#)
- [Visual Studio y GitHub mejor juntos](#)

Resolución de conflictos de combinación en Visual Studio

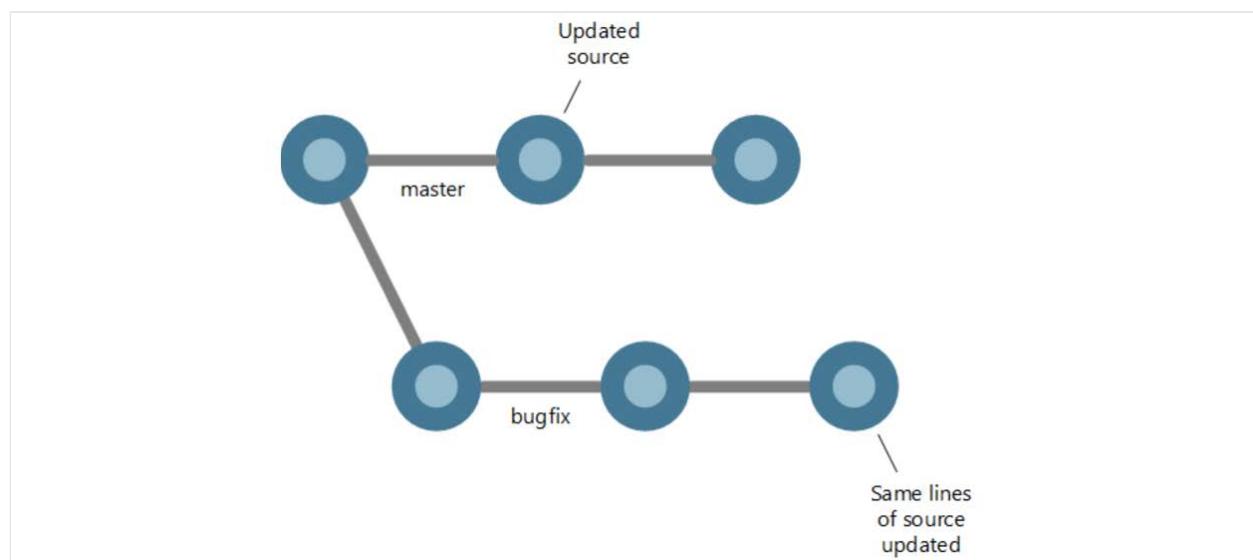
Artículo • 18/03/2023 • Tiempo de lectura: 3 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

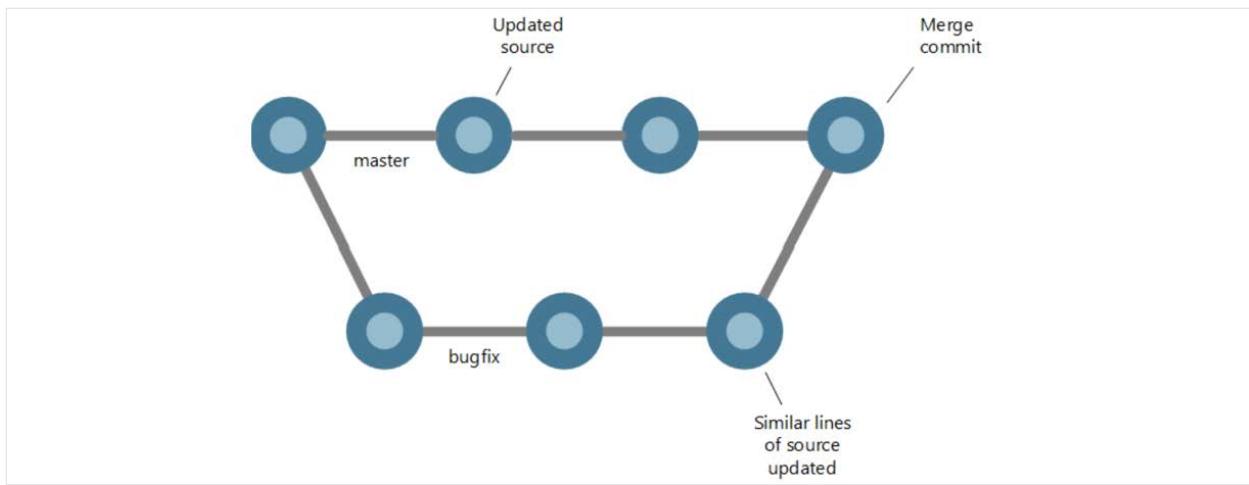
Al fusionar una rama con otra mediante combinación ("merge"), los cambios de archivo de las confirmaciones ("commit") de una rama pueden entrar en conflicto con los cambios de la otra rama. Git intenta resolver estos cambios mediante el historial del repositorio para determinar el aspecto que deben tener los archivos combinados. Cuando no está claro cómo combinar los cambios, Git detiene la combinación e indica qué archivos entran en conflicto.

Descripción de los conflictos de combinación

En la imagen siguiente se muestra un ejemplo básico de cómo los conflictos entran en conflicto en Git. En este ejemplo, la rama principal y la rama bugfix realizan actualizaciones en las mismas líneas del código fuente.



Si intenta fusionar mediante combinación la rama bugfix con la principal, Git no puede determinar qué cambios se deben usar en la versión combinada. Puede que quiera conservar los cambios en la rama principal, en la rama bugfix o en una combinación de ambas. Resuelva este conflicto con un commit de fusión mediante combinación ("merge commit") en la rama principal que concilie los cambios en conflicto entre ambas ramas.



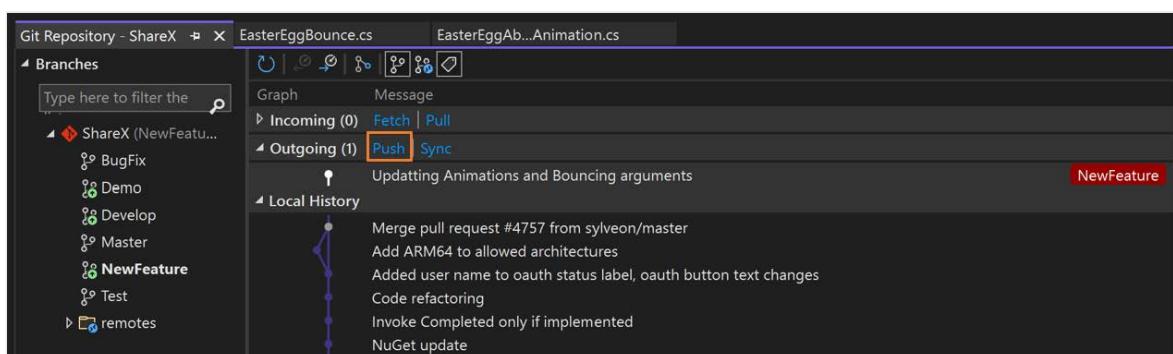
El escenario de conflicto de fusión mediante combinación más común se produce cuando se incorporan cambios ("pull") de actualizaciones de una rama remota a la rama local (por ejemplo, desde la rama de origen o bugfix a la rama bugfix local). Estos conflictos se pueden resolver de la misma forma: cree un commit en la rama local para conciliar los cambios y, a continuación, complete la fusión mediante combinación.

Evitar conflictos de fusión mediante combinación

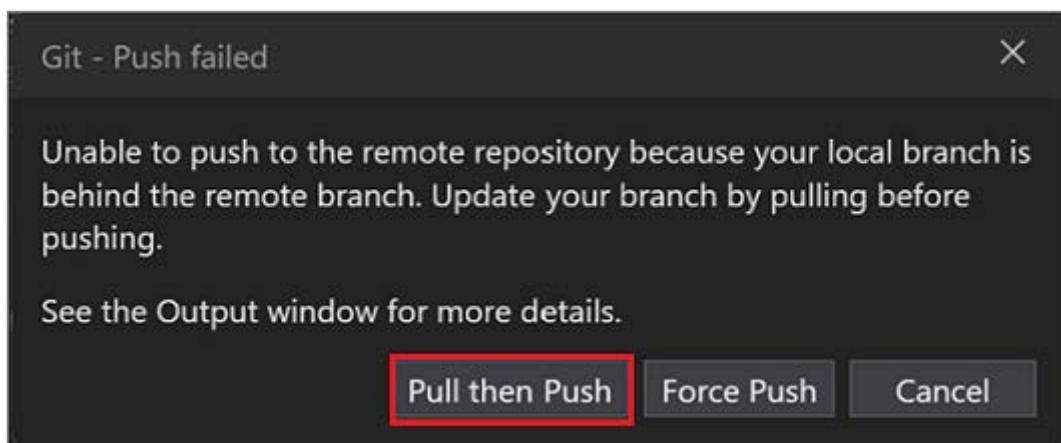
Git resulta óptimo para fusionar mediante combinación los cambios de archivo automáticamente en la mayoría de los casos, siempre y cuando el contenido del archivo no cambie drásticamente entre los commits. Si su rama está muy por detrás de la rama principal, considere la posibilidad de fusionar las ramas mediante cambio de base ("rebase") antes de abrir una solicitud de incorporación de cambios ("pull request"). Las ramas fusionadas mediante cambio de base se fusionarán mediante combinación en la rama principal sin conflictos.

Resolución de conflictos de combinación

- Si colabora con otros usuarios en la misma rama, puede que observe conflictos de fusión mediante combinación al enviar los cambios ("push").



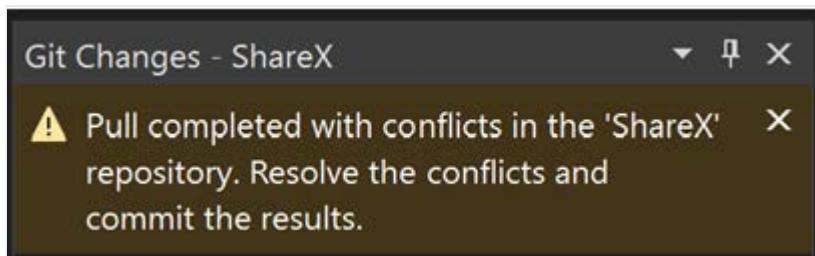
- Visual Studio detecta si la rama local en la que ha estado trabajando está detrás de la rama de seguimiento remoto y, a continuación, le ofrece opciones para elegir.



ⓘ Nota

Si el repositorio remoto admite la opción **Forzar envío de cambios**, puede habilitarlo mediante **Git>Configuración**.

En este ejemplo, seleccione **Incorporar y enviar cambios** para incluir los cambios introducidos en el repositorio remoto. Si hay conflictos de fusión mediante combinación al incorporar cambios o al intentar fusionar mediante combinación dos ramas, Visual Studio le permite saberlo en la ventana **Cambios de Git**, en la ventana **Repositorio de Git** y en los archivos que presenten conflictos.



- La ventana **Cambios de GIT** muestra una lista de archivos con conflictos en **Cambios sin fusionar**. Para empezar a resolver conflictos, haga doble clic en uno de los archivos. Si tiene un archivo con conflictos abiertos en el editor, también puede seleccionar **Abrir el Editor de combinación**.

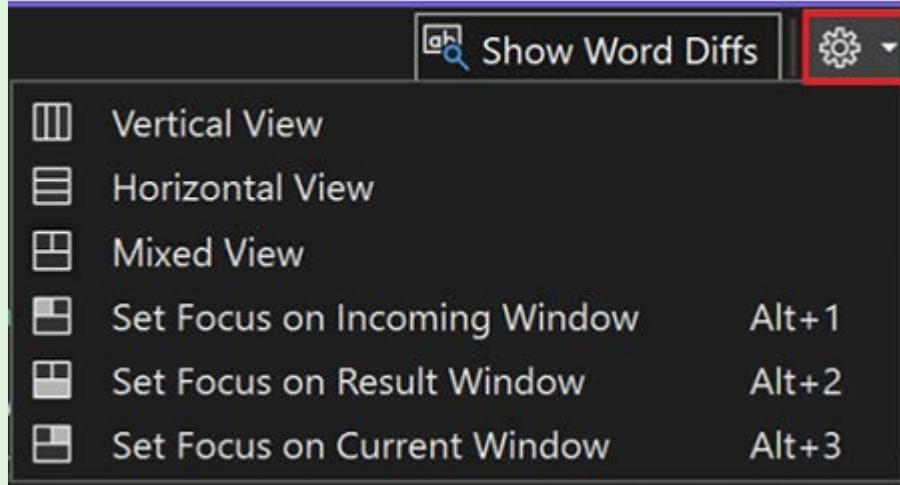
The screenshot shows the Visual Studio interface with the 'Git Changes' tool window open. The 'Unmerged Changes' section highlights two files: 'EasterEggAboutAnimation.cs' and 'EasterEggBounce.cs'. The code editor displays a conflict in the 'EasterEggBounce.cs' file, specifically in the constructor where there is a分歧 between the incoming code (left) and the local code (right).

- En el Editor de combinación, empiece a resolver el conflicto mediante cualquiera de los métodos siguientes (como se muestra en la captura de pantalla numerada):
 - Vaya a los conflictos, línea por línea, y elija entre mantener la parte derecha o la parte izquierda con la selección de las casillas correspondientes.
 - Conserve o ignore todos los cambios en conflicto.
 - Edite el código manualmente en la ventana Resultado.

This screenshot illustrates the 'Merge Conflict' editor in Visual Studio. It features three main panes: 'Incoming: Remote' (top left), 'Current: Local' (top right), and 'Result' (bottom). The 'Current: Local' pane contains a large orange box highlighting the selected changes. The 'Result' pane at the bottom shows the final merged code, which includes the selected local changes from the 'Current: Local' pane. The status bar at the bottom indicates the result of the merge.

Sugerencia

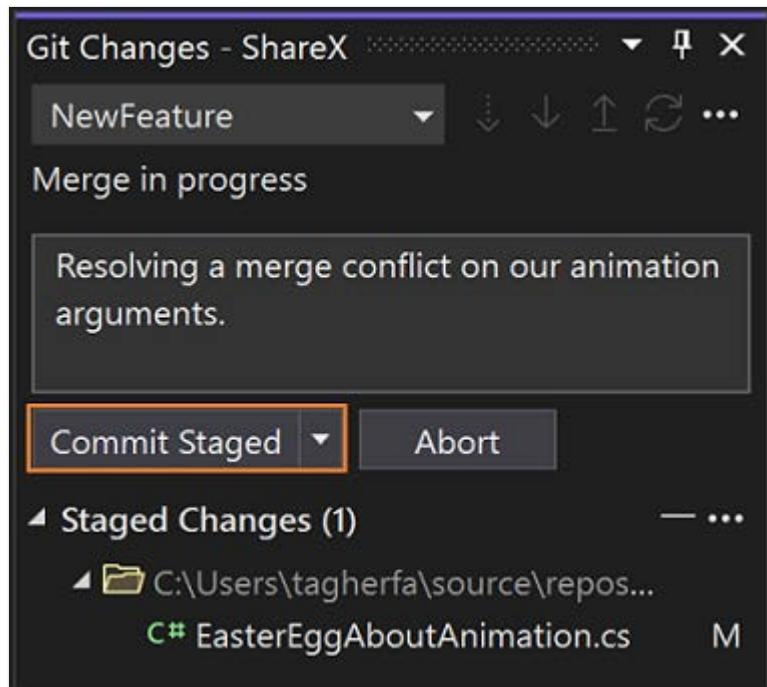
Si no le gusta el diseño predeterminado en el Editor de combinación, no dude en cambiarlo mediante el menú desplegable del engranaje.



Por ejemplo, en la captura de pantalla siguiente se muestra el aspecto de la vista vertical:

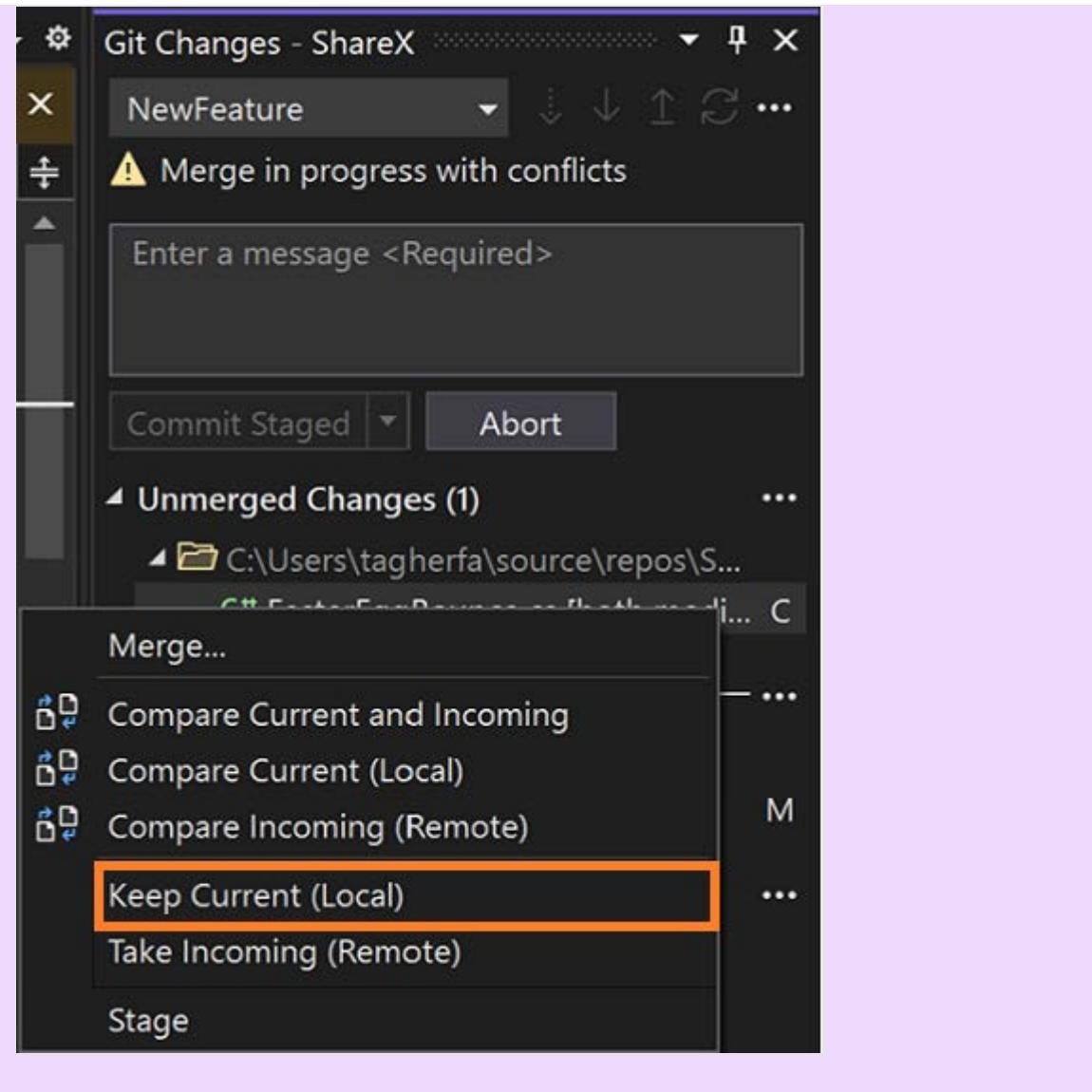
```

Incoming: Remote
33 { public class EasterEggAboutAnimation : IDisposable
34 {
35     public Canvas Canvas { get; private set; }
36     public bool IsPaused { get; set; }
37     public Size Size { get; set; } = new Size(0, 0);
38     public int Step { get; set; } = 10;
39     public int MinStep { get; set; } = 3;
40     public int MaxStep { get; set; } = 35;
41     public Speed { get; set; } = 1;
42     public Color Color { get; set; } = new HSB(0, 1, 1);
43     public int ClickCount { get; private set; }
44
45     private EasterEggBounce easterEggBounce;
46     private int direction;
47
48     public EasterEggAboutAnimation(Canvas canv,
49     {
50         Canvas = canv;
51         Canvas.MouseDown += Canvas_MouseDown;
52         Canvas.Draw += Canvas_Draw;
53     }
54
55     easterEggBounce = new EasterEggBounce();
56 }
57
58 public void Start()
59 {
60     direction = Speed;
61     Canvas.Start(50);
62 }
63
64 private void Canvas_MouseDown(object sender,
65 {
66     IsPaused = !IsPaused;
67
68     if (!easterEggBounce.IsWorking)
69     {
70         ClickCount++;
71
72         if (ClickCount >= 12)
73         {
74             easterEggBounce.ApplyGravity =
75             easterEggBounce.Start();
76         }
77     }
78 }
79
80 private void Canvas_Draw(object sender,
81 {
82     easterEggBounce.Update();
83
84     if (!easterEggBounce.IsWorking)
85     {
86         ClickCount++;
87
88         if (ClickCount >= 10)
89         {
90             easterEggBounce.ApplyGravity =
91             easterEggBounce.Start();
92         }
93     }
94 }
95
96 private void Canvas_Start(object sender,
97 {
98     easterEggBounce.Start();
99 }
100
101 private void Canvas_MouseUp(object sender,
102 {
103     IsPaused = false;
104 }
105
106 private void Canvas_MouseMove(object sender,
107 {
108     easterEggBounce.Move();
109 }
110
111 private void Canvas_KeyDown(object sender,
112 {
113     easterEggBounce.KeyDown();
114 }
115
116 private void Canvas_KeyUp(object sender,
117 {
118     easterEggBounce.KeyUp();
119 }
120
121 private void Canvas_DragOver(object sender,
122 {
123     easterEggBounce.DragOver();
124 }
125
126 private void Canvas_DragLeave(object sender,
127 {
128     easterEggBounce.DragLeave();
129 }
130
131 private void Canvas_DragDrop(object sender,
132 {
133     easterEggBounce.DragDrop();
134 }
135
136 private void Canvas_DragEnter(object sender,
137 {
138     easterEggBounce.DragEnter();
139 }
140
141 private void Canvas_DragLeave(object sender,
142 {
143     easterEggBounce.DragLeave();
144 }
145
146 private void Canvas_DragDrop(object sender,
147 {
148     easterEggBounce.DragDrop();
149 }
150
151 private void Canvas_DragOver(object sender,
152 {
153     easterEggBounce.DragOver();
154 }
155
156 private void Canvas_DragLeave(object sender,
157 {
158     easterEggBounce.DragLeave();
159 }
160
161 private void Canvas_DragDrop(object sender,
162 {
163     easterEggBounce.DragDrop();
164 }
165
166 private void Canvas_DragEnter(object sender,
167 {
168     easterEggBounce.DragEnter();
169 }
170
171 private void Canvas_DragLeave(object sender,
172 {
173     easterEggBounce.DragLeave();
174 }
175
176 private void Canvas_DragDrop(object sender,
177 {
178     easterEggBounce.DragDrop();
179 }
180
181 private void Canvas_DragOver(object sender,
182 {
183     easterEggBounce.DragOver();
184 }
185
186 private void Canvas_DragLeave(object sender,
187 {
188     easterEggBounce.DragLeave();
189 }
190
191 private void Canvas_DragDrop(object sender,
192 {
193     easterEggBounce.DragDrop();
194 }
195
196 private void Canvas_DragEnter(object sender,
197 {
198     easterEggBounce.DragEnter();
199 }
199
200 private void Canvas_DragLeave(object sender,
201 {
202     easterEggBounce.DragLeave();
203 }
203
204 private void Canvas_DragDrop(object sender,
205 {
206     easterEggBounce.DragDrop();
207 }
207
208 private void Canvas_DragOver(object sender,
209 {
210     easterEggBounce.DragOver();
211 }
211
212 private void Canvas_DragLeave(object sender,
213 {
214     easterEggBounce.DragLeave();
215 }
215
216 private void Canvas_DragDrop(object sender,
217 {
218     easterEggBounce.DragDrop();
219 }
219
220 private void Canvas_DragEnter(object sender,
221 {
222     easterEggBounce.DragEnter();
223 }
223
224 private void Canvas_DragLeave(object sender,
225 {
226     easterEggBounce.DragLeave();
227 }
227
228 private void Canvas_DragDrop(object sender,
229 {
230     easterEggBounce.DragDrop();
231 }
231
232 private void Canvas_DragOver(object sender,
233 {
234     easterEggBounce.DragOver();
235 }
235
236 private void Canvas_DragLeave(object sender,
237 {
238     easterEggBounce.DragLeave();
239 }
239
240 private void Canvas_DragDrop(object sender,
241 {
242     easterEggBounce.DragDrop();
243 }
243
244 private void Canvas_DragEnter(object sender,
245 {
246     easterEggBounce.DragEnter();
247 }
247
248 private void Canvas_DragLeave(object sender,
249 {
250     easterEggBounce.DragLeave();
251 }
251
252 private void Canvas_DragDrop(object sender,
253 {
254     easterEggBounce.DragDrop();
255 }
255
256 private void Canvas_DragOver(object sender,
257 {
258     easterEggBounce.DragOver();
259 }
259
260 private void Canvas_DragLeave(object sender,
261 {
262     easterEggBounce.DragLeave();
263 }
263
264 private void Canvas_DragDrop(object sender,
265 {
266     easterEggBounce.DragDrop();
267 }
267
268 private void Canvas_DragEnter(object sender,
269 {
270     easterEggBounce.DragEnter();
271 }
271
272 private void Canvas_DragLeave(object sender,
273 {
274     easterEggBounce.DragLeave();
275 }
275
276 private void Canvas_DragDrop(object sender,
277 {
278     easterEggBounce.DragDrop();
279 }
279
280 private void Canvas_DragOver(object sender,
281 {
282     easterEggBounce.DragOver();
283 }
283
284 private void Canvas_DragLeave(object sender,
285 {
286     easterEggBounce.DragLeave();
287 }
287
288 private void Canvas_DragDrop(object sender,
289 {
290     easterEggBounce.DragDrop();
291 }
291
292 private void Canvas_DragEnter(object sender,
293 {
294     easterEggBounce.DragEnter();
295 }
295
296 private void Canvas_DragLeave(object sender,
297 {
298     easterEggBounce.DragLeave();
299 }
299
298 private void Canvas_DragDrop(object sender,
299 {
300     easterEggBounce.DragDrop();
301 }
301
302 private void Canvas_DragOver(object sender,
303 {
304     easterEggBounce.DragOver();
305 }
305
306 private void Canvas_DragLeave(object sender,
307 {
308     easterEggBounce.DragLeave();
309 }
309
310 private void Canvas_DragDrop(object sender,
311 {
312     easterEggBounce.DragDrop();
313 }
313
314 private void Canvas_DragEnter(object sender,
315 {
316     easterEggBounce.DragEnter();
317 }
317
318 private void Canvas_DragLeave(object sender,
319 {
320     easterEggBounce.DragLeave();
321 }
321
322 private void Canvas_DragDrop(object sender,
323 {
324     easterEggBounce.DragDrop();
325 }
325
326 private void Canvas_DragOver(object sender,
327 {
328     easterEggBounce.DragOver();
329 }
329
330 private void Canvas_DragLeave(object sender,
331 {
332     easterEggBounce.DragLeave();
333 }
333
334 private void Canvas_DragDrop(object sender,
335 {
336     easterEggBounce.DragDrop();
337 }
337
338 private void Canvas_DragEnter(object sender,
339 {
340     easterEggBounce.DragEnter();
341 }
341
342 private void Canvas_DragLeave(object sender,
343 {
344     easterEggBounce.DragLeave();
345 }
345
346 private void Canvas_DragDrop(object sender,
347 {
348     easterEggBounce.DragDrop();
349 }
349
350 private void Canvas_DragOver(object sender,
351 {
352     easterEggBounce.DragOver();
353 }
353
354 private void Canvas_DragLeave(object sender,
355 {
356     easterEggBounce.DragLeave();
357 }
357
358 private void Canvas_DragDrop(object sender,
359 {
360     easterEggBounce.DragDrop();
361 }
361
362 private void Canvas_DragEnter(object sender,
363 {
364     easterEggBounce.DragEnter();
365 }
365
366 private void Canvas_DragLeave(object sender,
367 {
368     easterEggBounce.DragLeave();
369 }
369
370 private void Canvas_DragDrop(object sender,
371 {
372     easterEggBounce.DragDrop();
373 }
373
374 private void Canvas_DragOver(object sender,
375 {
376     easterEggBounce.DragOver();
377 }
377
378 private void Canvas_DragLeave(object sender,
379 {
380     easterEggBounce.DragLeave();
381 }
381
382 private void Canvas_DragDrop(object sender,
383 {
384     easterEggBounce.DragDrop();
385 }
385
386 private void Canvas_DragEnter(object sender,
387 {
388     easterEggBounce.DragEnter();
389 }
389
390 private void Canvas_DragLeave(object sender,
391 {
392     easterEggBounce.DragLeave();
393 }
393
394 private void Canvas_DragDrop(object sender,
395 {
396     easterEggBounce.DragDrop();
397 }
397
398 private void Canvas_DragOver(object sender,
399 {
398     easterEggBounce.DragOver();
399 }
399
400 private void Canvas_DragLeave(object sender,
401 {
402     easterEggBounce.DragLeave();
403 }
403
404 private void Canvas_DragDrop(object sender,
405 {
406     easterEggBounce.DragDrop();
407 }
407
408 private void Canvas_DragEnter(object sender,
409 {
410     easterEggBounce.DragEnter();
411 }
411
412 private void Canvas_DragLeave(object sender,
413 {
414     easterEggBounce.DragLeave();
415 }
415
416 private void Canvas_DragDrop(object sender,
417 {
418     easterEggBounce.DragDrop();
419 }
419
420 private void Canvas_DragOver(object sender,
421 {
422     easterEggBounce.DragOver();
423 }
423
424 private void Canvas_DragLeave(object sender,
425 {
426     easterEggBounce.DragLeave();
427 }
427
428 private void Canvas_DragDrop(object sender,
429 {
430     easterEggBounce.DragDrop();
431 }
431
432 private void Canvas_DragEnter(object sender,
433 {
434     easterEggBounce.DragEnter();
435 }
435
436 private void Canvas_DragLeave(object sender,
437 {
438     easterEggBounce.DragLeave();
439 }
439
440 private void Canvas_DragDrop(object sender,
441 {
442     easterEggBounce.DragDrop();
443 }
443
444 private void Canvas_DragOver(object sender,
445 {
446     easterEggBounce.DragOver();
447 }
447
448 private void Canvas_DragLeave(object sender,
449 {
450     easterEggBounce.DragLeave();
451 }
451
452 private void Canvas_DragDrop(object sender,
453 {
454     easterEggBounce.DragDrop();
455 }
455
456 private void Canvas_DragEnter(object sender,
457 {
458     easterEggBounce.DragEnter();
459 }
459
460 private void Canvas_DragLeave(object sender,
461 {
462     easterEggBounce.DragLeave();
463 }
463
464 private void Canvas_DragDrop(object sender,
465 {
466     easterEggBounce.DragDrop();
467 }
467
468 private void Canvas_DragOver(object sender,
469 {
470     easterEggBounce.DragOver();
471 }
471
472 private void Canvas_DragLeave(object sender,
473 {
474     easterEggBounce.DragLeave();
475 }
475
476 private void Canvas_DragDrop(object sender,
477 {
478     easterEggBounce.DragDrop();
479 }
479
480 private void Canvas_DragEnter(object sender,
481 {
482     easterEggBounce.DragEnter();
483 }
483
484 private void Canvas_DragLeave(object sender,
485 {
486     easterEggBounce.DragLeave();
487 }
487
488 private void Canvas_DragDrop(object sender,
489 {
490     easterEggBounce.DragDrop();
491 }
491
492 private void Canvas_DragOver(object sender,
493 {
494     easterEggBounce.DragOver();
495 }
495
496 private void Canvas_DragLeave(object sender,
497 {
498     easterEggBounce.DragLeave();
499 }
499
500 private void Canvas_DragDrop(object sender,
501 {
502     easterEggBounce.DragDrop();
503 }
503
504 private void Canvas_DragEnter(object sender,
505 {
506     easterEggBounce.DragEnter();
507 }
507
508 private void Canvas_DragLeave(object sender,
509 {
510     easterEggBounce.DragLeave();
511 }
511
512 private void Canvas_DragDrop(object sender,
513 {
514     easterEggBounce.DragDrop();
515 }
515
516 private void Canvas_DragOver(object sender,
517 {
518     easterEggBounce.DragOver();
519 }
519
520 private void Canvas_DragLeave(object sender,
521 {
522     easterEggBounce.DragLeave();
523 }
523
524 private void Canvas_DragDrop(object sender,
525 {
526     easterEggBounce.DragDrop();
527 }
527
528 private void Canvas_DragEnter(object sender,
529 {
530     easterEggBounce.DragEnter();
531 }
531
532 private void Canvas_DragLeave(object sender,
533 {
534     easterEggBounce.DragLeave();
535 }
535
536 private void Canvas_DragDrop(object sender,
537 {
538     easterEggBounce.DragDrop();
539 }
539
540 private void Canvas_DragOver(object sender,
541 {
542     easterEggBounce.DragOver();
543 }
543
544 private void Canvas_DragLeave(object sender,
545 {
546     easterEggBounce.DragLeave();
547 }
547
548 private void Canvas_DragDrop(object sender,
549 {
550     easterEggBounce.DragDrop();
551 }
551
552 private void Canvas_DragEnter(object sender,
553 {
554     easterEggBounce.DragEnter();
555 }
555
556 private void Canvas_DragLeave(object sender,
557 {
558     easterEggBounce.DragLeave();
559 }
559
560 private void Canvas_DragDrop(object sender,
561 {
562     easterEggBounce.DragDrop();
563 }
563
564 private void Canvas_DragOver(object sender,
565 {
566     easterEggBounce.DragOver();
567 }
567
568 private void Canvas_DragLeave(object sender,
569 {
570     easterEggBounce.DragLeave();
571 }
571
572 private void Canvas_DragDrop(object sender,
573 {
574     easterEggBounce.DragDrop();
575 }
575
576 private void Canvas_DragEnter(object sender,
577 {
578     easterEggBounce.DragEnter();
579 }
579
580 private void Canvas_DragLeave(object sender,
581 {
582     easterEggBounce.DragLeave();
583 }
583
584 private void Canvas_DragDrop(object sender,
585 {
586     easterEggBounce.DragDrop();
587 }
587
588 private void Canvas_DragOver(object sender,
589 {
590     easterEggBounce.DragOver();
591 }
591
592 private void Canvas_DragLeave(object sender,
593 {
594     easterEggBounce.DragLeave();
595 }
595
596 private void Canvas_DragDrop(object sender,
597 {
598     easterEggBounce.DragDrop();
599 }
599
598 private void Canvas_DragEnter(object sender,
599 {
600     easterEggBounce.DragEnter();
601 }
601
602 private void Canvas_DragLeave(object sender,
603 {
604     easterEggBounce.DragLeave();
605 }
605
606 private void Canvas_DragDrop(object sender,
607 {
608     easterEggBounce.DragDrop();
609 }
609
610 private void Canvas_DragOver(object sender,
611 {
612     easterEggBounce.DragOver();
613 }
613
614 private void Canvas_DragLeave(object sender,
615 {
616     easterEggBounce.DragLeave();
617 }
617
618 private void Canvas_DragDrop(object sender,
619 {
620     easterEggBounce.DragDrop();
621 }
621
622 private void Canvas_DragEnter(object sender,
623 {
624     easterEggBounce.DragEnter();
625 }
625
626 private void Canvas_DragLeave(object sender,
627 {
628     easterEggBounce.DragLeave();
629 }
629
630 private void Canvas_DragDrop(object sender,
631 {
632     easterEggBounce.DragDrop();
633 }
633
634 private void Canvas_DragOver(object sender,
635 {
636     easterEggBounce.DragOver();
637 }
637
638 private void Canvas_DragLeave(object sender,
639 {
640     easterEggBounce.DragLeave();
641 }
641
642 private void Canvas_DragDrop(object sender,
643 {
644     easterEggBounce.DragDrop();
645 }
645
646 private void Canvas_DragEnter(object sender,
647 {
648     easterEggBounce.DragEnter();
649 }
649
650 private void Canvas_DragLeave(object sender,
651 {
652     easterEggBounce.DragLeave();
653 }
653
654 private void Canvas_DragDrop(object sender,
655 {
656     easterEggBounce.DragDrop();
657 }
657
658 private void Canvas_DragOver(object sender,
659 {
660     easterEggBounce.DragOver();
661 }
661
662 private void Canvas_DragLeave(object sender,
663 {
664     easterEggBounce.DragLeave();
665 }
665
666 private void Canvas_DragDrop(object sender,
667 {
668     easterEggBounce.DragDrop();
669 }
669
670 private void Canvas_DragEnter(object sender,
671 {
672     easterEggBounce.DragEnter();
673 }
673
674 private void Canvas_DragLeave(object sender,
675 {
676     easterEggBounce.DragLeave();
677 }
677
678 private void Canvas_DragDrop(object sender,
679 {
680     easterEggBounce.DragDrop();
681 }
681
682 private void Canvas_DragOver(object sender,
683 {
684     easterEggBounce.DragOver();
685 }
685
686 private void Canvas_DragLeave(object sender,
687 {
688     easterEggBounce.DragLeave();
689 }
689
690 private void Canvas_DragDrop(object sender,
691 {
692     easterEggBounce.DragDrop();
693 }
693
694 private void Canvas_DragEnter(object sender,
695 {
696     easterEggBounce.DragEnter();
697 }
697
698 private void Canvas_DragLeave(object sender,
699 {
700     easterEggBounce.DragLeave();
701 }
701
702 private void Canvas_DragDrop(object sender,
703 {
704     easterEggBounce.DragDrop();
705 }
705
706 private void Canvas_DragOver(object sender,
707 {
708     easterEggBounce.DragOver();
709 }
709
710 private void Canvas_DragLeave(object sender,
711 {
712     easterEggBounce.DragLeave();
713 }
713
714 private void Canvas_DragDrop(object sender,
715 {
716     easterEggBounce.DragDrop();
717 }
717
718 private void Canvas_DragEnter(object sender,
719 {
720     easterEggBounce.DragEnter();
721 }
721
722 private void Canvas_DragLeave(object sender,
723 {
724     easterEggBounce.DragLeave();
725 }
725
726 private void Canvas_DragDrop(object sender,
727 {
728     easterEggBounce.DragDrop();
729 }
729
730 private void Canvas_DragOver(object sender,
731 {
732     easterEggBounce.DragOver();
733 }
733
734 private void Canvas_DragLeave(object sender,
735 {
736     easterEggBounce.DragLeave();
737 }
737
738 private void Canvas_DragDrop(object sender,
739 {
740     easterEggBounce.DragDrop();
741 }
741
742 private void Canvas_DragEnter(object sender,
743 {
744     easterEggBounce.DragEnter();
745 }
745
746 private void Canvas_DragLeave(object sender,
747 {
748     easterEggBounce.DragLeave();
749 }
749
750 private void Canvas_DragDrop(object sender,
751 {
752     easterEggBounce.DragDrop();
753 }
753
754 private void Canvas_DragOver(object sender,
755 {
756     easterEggBounce.DragOver();
757 }
757
758 private void Canvas_DragLeave(object sender,
759 {
760     easterEggBounce.DragLeave();
761 }
761
762 private void Canvas_DragDrop(object sender,
763 {
764     easterEggBounce.DragDrop();
765 }
765
766 private void Canvas_DragEnter(object sender,
767 {
768     easterEggBounce.DragEnter();
769 }
769
770 private void Canvas_DragLeave(object sender,
771 {
772     easterEggBounce.DragLeave();
773 }
773
774 private void Canvas_DragDrop(object sender,
775 {
776     easterEggBounce.DragDrop();
777 }
777
778 private void Canvas_DragOver(object sender,
779 {
780     easterEggBounce.DragOver();
781 }
781
782 private void Canvas_DragLeave(object sender,
783 {
784     easterEggBounce.DragLeave();
785 }
785
786 private void Canvas_DragDrop(object sender,
787 {
788     easterEggBounce.DragDrop();
789 }
789
790 private void Canvas_DragEnter(object sender,
791 {
792     easterEggBounce.DragEnter();
793 }
793
794 private void Canvas_DragLeave(object sender,
795 {
796     easterEggBounce.DragLeave();
797 }
797
798 private void Canvas_DragDrop(object sender,
799 {
800     easterEggBounce.DragDrop();
801 }
801
802 private void Canvas_DragOver(object sender,
803 {
804     easterEggBounce.DragOver();
805 }
805
806 private void Canvas_DragLeave(object sender,
807 {
808     easterEggBounce.DragLeave();
809 }
809
810 private void Canvas_DragDrop(object sender,
811 {
812     easterEggBounce.DragDrop();
813 }
813
814 private void Canvas_DragEnter(object sender,
815 {
816     easterEggBounce.DragEnter();
817 }
817
818 private void Canvas_DragLeave(object sender,
819 {
820     easterEggBounce.DragLeave();
821 }
821
822 private void Canvas_DragDrop(object sender,
823 {
824     easterEggBounce.DragDrop();
825 }
825
826 private void Canvas_DragOver(object sender,
827 {
828     easterEggBounce.DragOver();
829 }
829
830 private void Canvas_DragLeave(object sender,
831 {
832     easterEggBounce.DragLeave();
833 }
833
834 private void Canvas_DragDrop(object sender,
835 {
836     easterEggBounce.DragDrop();
837 }
837
838 private void Canvas_DragEnter(object sender,
839 {
840     easterEggBounce.DragEnter();
841 }
841
842 private void Canvas_DragLeave(object sender,
843 {
844     easterEggBounce.DragLeave();
845 }
845
846 private void Canvas_DragDrop(object sender,
847 {
848     easterEggBounce.DragDrop();
849 }
849
850 private void Canvas_DragOver(object sender,
851 {
852     easterEggBounce.DragOver();
853 }
853
854 private void Canvas_DragLeave(object sender,
855 {
856     easterEggBounce.DragLeave();
857 }
857
858 private void Canvas_DragDrop(object sender,
859 {
860     easterEggBounce.DragDrop();
861 }
861
862 private void Canvas_DragEnter(object sender,
863 {
864     easterEggBounce.DragEnter();
865 }
865
866 private void Canvas_DragLeave(object sender,
867 {
868     easterEggBounce.DragLeave();
869 }
869
870 private void Canvas_DragDrop(object sender,
871 {
872     easterEggBounce.DragDrop();
873 }
873
874 private void Canvas_DragOver(object sender,
875 {
876     easterEggBounce.DragOver();
877 }
877
878 private void Canvas_DragLeave(object sender,
879 {
880     easterEggBounce.DragLeave();
881 }
881
882 private void Canvas_DragDrop(object sender,
883 {
884     easterEggBounce.DragDrop();
885 }
885
886 private void Canvas_DragEnter(object sender,
887 {
888     easterEggBounce.DragEnter();
889 }
889
890 private void Canvas_DragLeave(object sender,
891 {
892     easterEggBounce.DragLeave();
893 }
893
894 private void Canvas_DragDrop(object sender,
895 {
896     easterEggBounce.DragDrop();
897 }
897
898 private void Canvas_DragOver(object sender,
899 {
900     easterEggBounce.DragOver();
901 }
901
902 private void Canvas_DragLeave(object sender,
903 {
904     easterEggBounce.DragLeave();
905 }
905
906 private void Canvas_DragDrop(object sender,
907 {
908     easterEggBounce.DragDrop();
909 }
909
910 private void Canvas_DragEnter(object sender,
911 {
912     easterEggBounce.DragEnter();
913 }
913
914 private void Canvas_DragLeave(object sender,
915 {
916     easterEggBounce.DragLeave();
917 }
917
918 private void Canvas_DragDrop(object sender,
919 {
920     easterEggBounce.DragDrop();
921 }
921
922 private void Canvas_DragOver(object sender,
923 {
924     easterEggBounce.DragOver();
925 }
925
926 private void Canvas_DragLeave(object sender,
927 {
928     easterEggBounce.DragLeave();
929 }
929
930 private void Canvas_DragDrop(object sender,
931 {
932     easterEggBounce.DragDrop();
933 }
933
934 private void Canvas_DragEnter(object sender,
935 {
936     easterEggBounce.DragEnter();
937 }
937
938 private void Canvas_DragLeave(object sender,
939 {
940     easterEggBounce.DragLeave();
941 }
941
942 private void Canvas_DragDrop(object sender,
943 {
944     easterEggBounce.DragDrop();
945 }
945
946 private void Canvas_DragOver(object sender,
947 {
948     easterEggBounce.DragOver();
949 }
949
950 private void Canvas_DragLeave(object sender,
951 {
952     easterEggBounce.DragLeave();
953 }
953
954 private void Canvas_DragDrop(object sender,
955 {
956     easterEggBounce.DragDrop();
957 }
957
958 private void Canvas_DragEnter(object sender,
959 {
960     easterEggBounce.DragEnter();
961 }
961
962 private void Canvas_DragLeave(object sender,
963 {
964     easterEggBounce.DragLeave();
965 }
965
966 private void Canvas_DragDrop(object sender,
967 {
968     easterEggBounce.DragDrop();
969 }
969
970 private void Canvas_DragOver(object sender,
971 {
972     easterEggBounce.DragOver();
973 }
973
974 private void Canvas_DragLeave(object sender,
975 {
976     easterEggBounce.DragLeave();
977 }
977
978 private void Canvas_DragDrop(object sender,
979 {
980     easterEggBounce.DragDrop();
981 }
981
982 private void Canvas_DragEnter(object sender,
983 {
984     easterEggBounce.DragEnter();
985 }
985
986 private void Canvas_DragLeave(object sender,
987 {
988     easterEggBounce.DragLeave();
989 }
989
990 private void Canvas_DragDrop(object sender,
991 {
992     easterEggBounce.DragDrop();
993 }
993
994 private void Canvas_DragOver(object sender,
995 {
996     easterEggBounce.DragOver();
997 }
997
998 private void Canvas_DragLeave(object sender,
999 {
1000    easterEggBounce.DragLeave();
1001 }
1001
1002 private void Canvas_DragDrop(object sender,
1003 {
1004     easterEggBounce.DragDrop();
1005 }
1005
1006 private void Canvas_DragEnter(object sender,
1007 {
1008     easterEggBounce.DragEnter();
1009 }
1009
1010 private void Canvas_DragLeave(object sender,
1011 {
1012     easterEggBounce.DragLeave();
1013 }
1013
1014 private void Canvas_DragDrop(object sender,
1015 {
1016     easterEggBounce.DragDrop();
1017 }
1017
1018 private void Canvas_DragOver(object sender,
1019 {
1020     easterEggBounce.DragOver();
1021 }
1021
1022 private void Canvas_DragLeave(object sender,
1023 {
1024     easterEggBounce.DragLeave();
1025 }
1025
1026 private void Canvas_DragDrop(object sender,
1027 {
1028     easterEggBounce.DragDrop();
1029 }
1029
1030 private void Canvas_DragEnter(object sender,
1031 {
1032     easterEggBounce.DragEnter();
1033 }
1033
1034 private void Canvas_DragLeave(object sender,
1035 {
1036     easterEggBounce.DragLeave();
1037 }
1037
1038 private void Canvas_DragDrop(object sender,
1039 {
1040     easterEggBounce.DragDrop();
1041 }
1041
1042 private void Canvas_DragOver(object sender,
1043 {
1044     easterEggBounce.DragOver();
1045 }
1045
1046 private void Canvas_DragLeave(object sender,
1047 {
1048     easterEggBounce.DragLeave();
1049 }
1049
1050 private void Canvas_DragDrop(object sender,
1051 {
1052     easterEggBounce.DragDrop();
1053 }
1053
1054 private void Canvas_DragEnter(object sender,
1055 {
1056     easterEggBounce.DragEnter();
1057 }
1057
1058 private void Canvas_DragLeave(object sender,
1059 {
1060     easterEggBounce.DragLeave();
1061 }
1061
1062 private void Canvas_DragDrop(object sender,
1063 {
1064     easterEggBounce.DragDrop();
1065 }
1065
1066 private void Canvas_DragOver(object sender,
1067 {
1068     easterEggBounce.DragOver();
1069 }
1069
1070 private void Canvas_DragLeave(object sender,
1071 {
1072     easterEggBounce.DragLeave();
1073 }
1073
1074 private void Canvas_DragDrop(object sender,
1075 {
1076     easterEggBounce.DragDrop();
1077 }
1077
1078 private void Canvas_DragEnter(object sender,
1079 {
1080     easterEggBounce.DragEnter();
1081 }
1081
1082 private void Canvas_DragLeave(object sender,
1083 {
1084     easterEggBounce.DragLeave();
1085 }
1085
1086 private void Canvas_DragDrop(object sender,
1087 {
1088     easterEggBounce.DragDrop();
1089 }
1089
1090 private void Canvas_DragOver(object sender,
1091 {
1092     easterEggBounce.DragOver();
1093 }
1093
1094 private void Canvas_DragLeave(object sender,
1095 {
1096     easterEggBounce.DragLeave();
1097 }
1097
1098 private void Canvas_DragDrop(object sender,
1099 {
1100     easterEggBounce.DragDrop();
1101 }
1101
1102 private void Canvas_DragEnter(object sender,
1103 {
1104     easterEggBounce.DragEnter();
1105 }
1105
1106 private void Canvas_DragLeave(object sender,
1107 {
1108     easterEggBounce.DragLeave();
1109 }
1109
1110 private void Canvas_DragDrop(object sender,
1111 {
1112     easterEggBounce.DragDrop();
1113 }
1113
1114 private void Canvas_DragOver(object sender,
1115 {
1116     easterEggBounce.DragOver();
1117 }
1117
1118 private void Canvas_DragLeave(object sender,
1119 {
1120     easterEggBounce.DragLeave();
1121 }
1121
1122 private void Canvas_DragDrop(object sender,
1123 {
1124     easterEggBounce.DragDrop();
1125 }
1125
1126 private void Canvas_DragEnter(object sender,
1127 {
1128     easterEggBounce.DragEnter();
1129 }
1129
1130 private void Canvas_DragLeave(object sender,
1131 {
1132     easterEggBounce.DragLeave();
1133 }
1133
1134 private void Canvas_DragDrop(object sender,
1135 {
1136     easterEggBounce.DragDrop();
1137 }
1137
1138 private void Canvas_DragOver(object sender,
1139 {
1140     easterEggBounce.DragOver();
1141 }
1141
1142 private void Canvas_DragLeave(object sender,
1143 {
1144     easterEggBounce.DragLeave();
1145 }
1145
1146 private void Canvas_DragDrop(object sender,
1147 {
1148     easterEggBounce.DragDrop();
1149 }
1149
1150 private void Canvas_DragEnter(object sender,
1151 {
1152     easterEggBounce.DragEnter();
1153 }
1153
1154 private void Canvas_DragLeave(object sender,
1155 {
1156     easterEggBounce.DragLeave();
1157 }
1157
1158 private void Canvas_DragDrop(object sender,
1159 {
1160     easterEggBounce.DragDrop();
1161 }
1161
1162 private void Canvas_DragOver(object sender,
1163 {
1164     easterEggBounce.DragOver();
1165 }
1165
1166 private void Canvas_DragLeave(object sender,
1167 {
1168     easterEggBounce.DragLeave();
1169 }
1169
1170 private void Canvas_DragDrop(object sender,
1171 {
1172     easterEggBounce.DragDrop();
1173 }
1173
1174 private
```



⚠ Nota

Si necesita mantener todos los cambios en un archivo, puede hacer clic con el botón derecho en este en la sección **Cambios sin combinar** y seleccionar **Mantener actual (local)** sin tener que abrir el Editor de combinación.



💡 Sugerencia

Para obtener más información sobre las opciones de accesibilidad disponibles, consulte la sección **Métodos abreviados de teclado de Git** de la página [Sugerencias y trucos de accesibilidad de Visual Studio](#).

Pasos siguientes

Para continuar el recorrido y obtener más información sobre la resolución de conflictos, consulte la [página web de Git para el comando merge](#).

Consulte también

- [La experiencia Git en Visual Studio](#)
- [Visual Studio y GitHub mejor juntos](#)

Creación de una cuenta de GitHub para usarla con Visual Studio

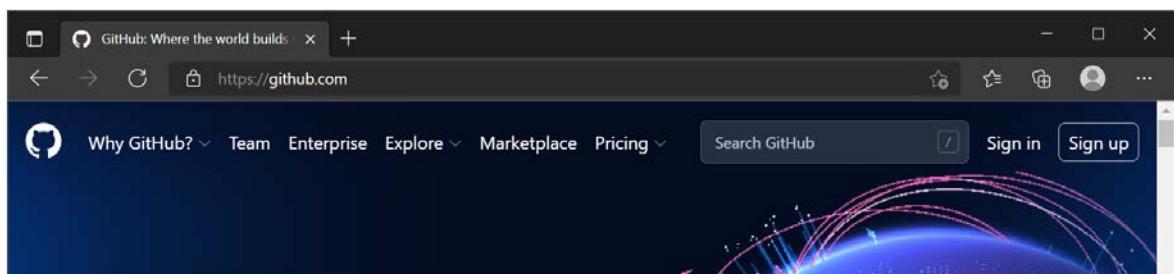
Artículo • 27/09/2022 • Tiempo de lectura: 2 minutos

Se aplica a:  Visual Studio  Visual Studio  para Mac  Visual Studio Code

Con una cuenta de GitHub, obtendrá [compatibilidad completa de GitHub](#) desde el [IDE de Visual Studio](#) para administrar el código y colaborar con otros usuarios en proyectos de desarrollo.

Si aún no tiene una cuenta de GitHub, aquí se muestra cómo crear una.

1. Abra <https://github.com> en un explorador web y seleccione **Registrarse**.



2. Escriba su dirección de correo electrónico .



3. Cree una contraseña para la nueva cuenta de GitHub y escriba también un nombre de usuario. A continuación, elija si desea recibir actualizaciones y anuncios por correo electrónico y, a continuación, seleccione **Continuar**.

Welcome to GitHub!
Let's begin the adventure

Enter your email

✓ username@contoso.com

Create a password

✓ ••••••••••••••

Enter a username

✓ NewUser

Would you like to receive product updates and announcements via email?

Type "y" for yes or "n" for no

→ |

Continue

4. Compruebe su cuenta resolviendo un rompecabezas. Seleccione el botón **Iniciar rompecabezas** para hacerlo y, a continuación, siga las indicaciones.
5. Después de comprobar la cuenta, seleccione el botón **Crear cuenta**.
6. A continuación, GitHub envía un código de inicio a la dirección de correo electrónico. Escriba ese código de inicio en el cuadro de diálogo **Escribir código** y presione **Entrar**.

You're almost done!
We sent a launch code to username@contoso.com

→ Enter code



Didn't get your email? Resend the code or update your email address.

7. GitHub le formula algunas preguntas para ayudar a adaptar su experiencia. Elija las respuestas que se le aplican en los cuadros de diálogo siguientes:

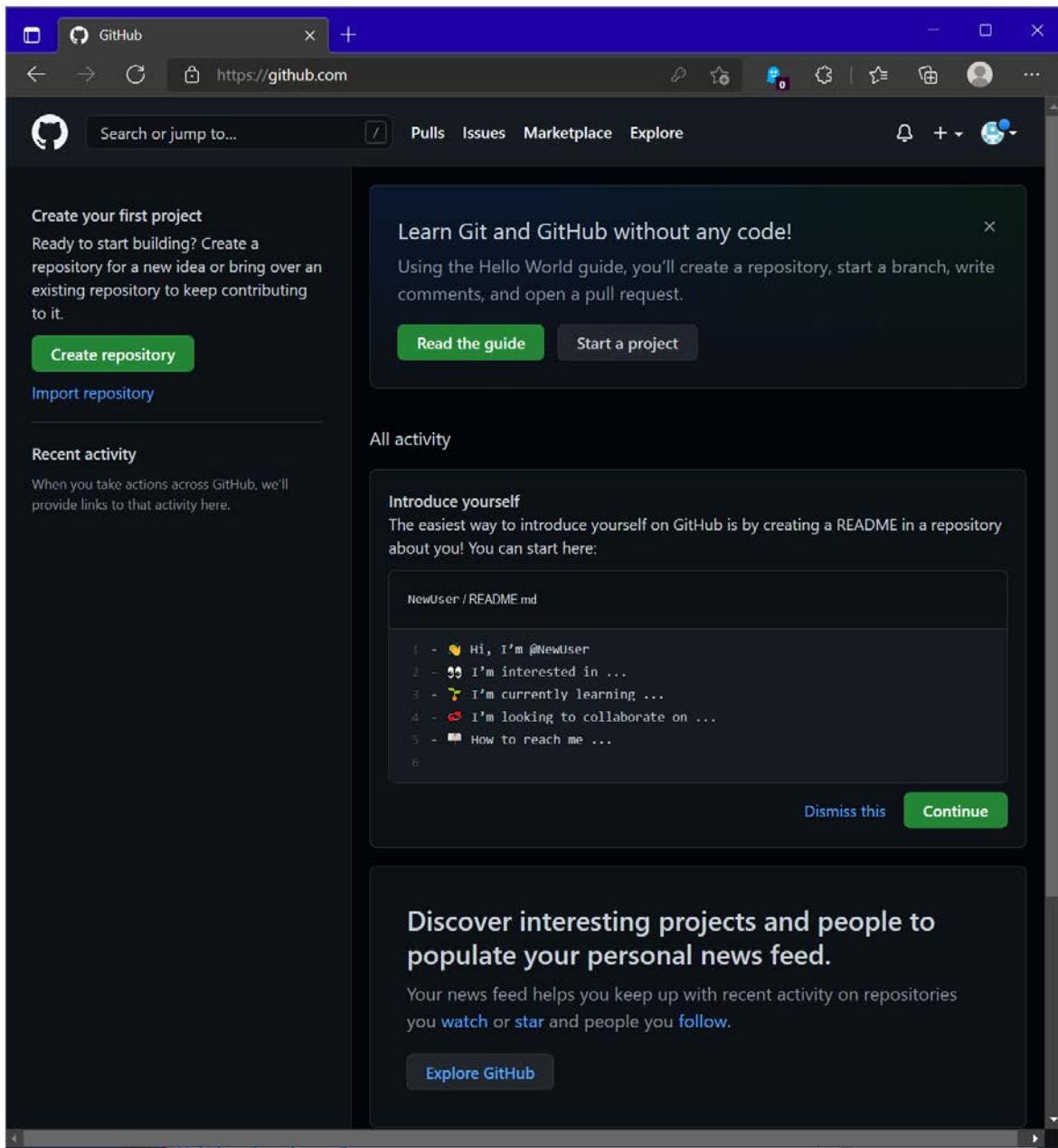
- ¿Cuántos miembros del equipo trabajarán con usted?
- ¿Qué características específicas le interesan usar?

8. En la pantalla **Dónde colaboran y envían los equipos**, puede elegir si desea usar la cuenta gratuita o la cuenta de equipo. Para elegir la cuenta **gratuita**, seleccione el botón **Omitir personalización**.

 **Sugerencia**

Siempre puede actualizar su cuenta más adelante. Consulte la página [Tipos de cuentas de GitHub](#) para obtener más información.

GitHub abre una página personalizada en el explorador.



¡Enhorabuena! Ha creado correctamente la cuenta de GitHub.

Pasos siguientes

Para continuar con el recorrido, visite la página [Clonar un repositorio](#).

Vea también

- [Tutorial: Abrir un proyecto desde un repositorio](#)
- [GitHub de Visual Studio & : mejor juntos](#)

Comparación de la experiencia de Git con Team Explorer en Visual Studio 2019

Artículo • 18/03/2023 • Tiempo de lectura: 5 minutos

Se aplica a: Visual Studio Visual Studio para Mac Visual Studio Code

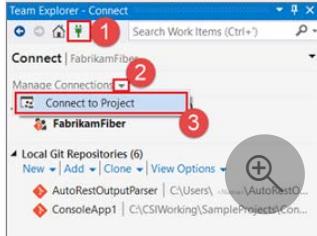
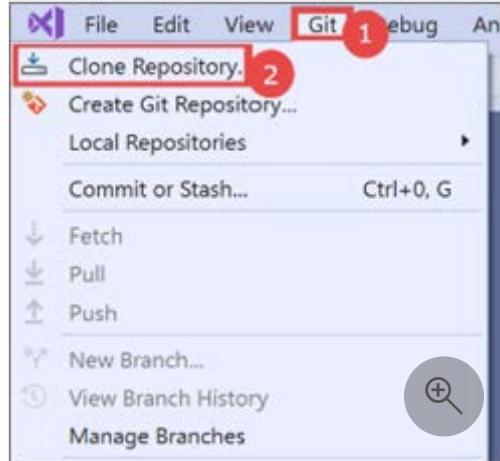
Lanzamos la primera versión de una experiencia de Git en Visual Studio 2019, [versión 16.8](#). Esta experiencia ayuda a reducir el cambio de contexto con una sencilla ventana **Cambios de Git** que incluye las tareas comunes de Git. También incluye una ventana amplia del **repositorio de Git** para las operaciones de Git más avanzadas, como la administración de ramas y la exploración de repositorios.

Si ha usado Team Explorer, esta es una guía paso a paso que explica cómo usar la experiencia de Git.

ⓘ Nota

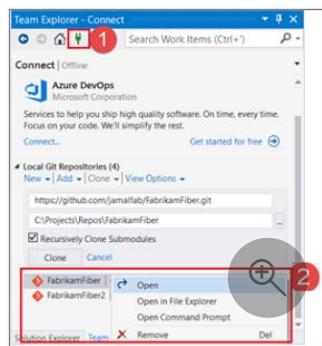
En la siguiente sección se incluyen capturas de pantallas con el tamaño adecuado para ajustarse a las columnas de la tabla. Haga clic en cada captura de pantalla para ver una versión más grande. (Si usa un dispositivo de pantalla táctil, pulse en cada captura de pantalla para ver una versión más grande de esta).

Introducción

Team Explorer	Experiencia de Git
<p>Clonación de un repositorio</p>  <ol style="list-style-type: none">1. Abra la página Conectar.2. Expanda Administrar conexiones.3. Seleccione Conectar con el proyecto.	 <ol style="list-style-type: none">1. Abra el menú Git.2. Seleccione Clonar repositorio.

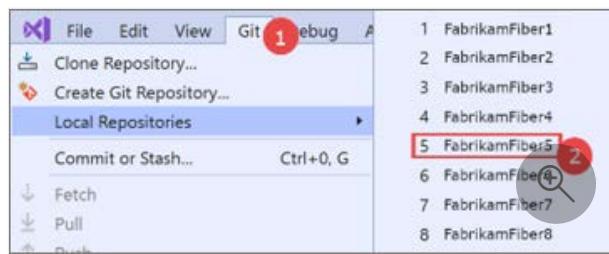
Team Explorer

Cambio entre repositorios



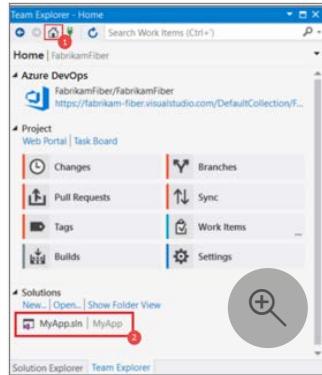
1. Abra la página **Conectar**.
2. Seleccione un repositorio de la lista **Repositorios locales**.

Experiencia de Git

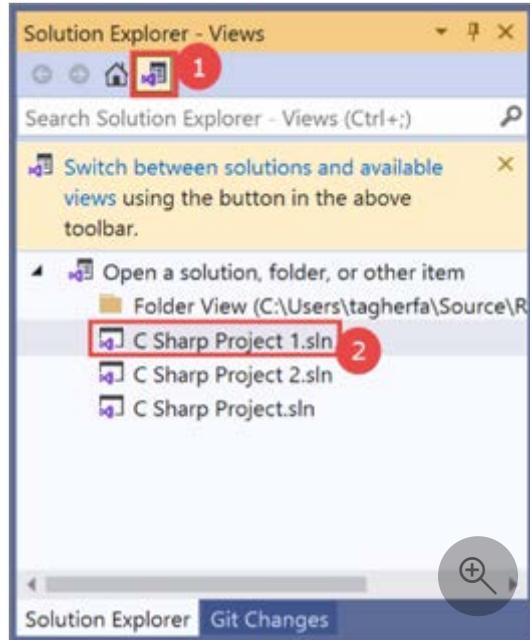


1. Abra el menú **Git**.
2. Seleccione un repositorio de la lista **Repositorios locales**.

Apertura de una solución



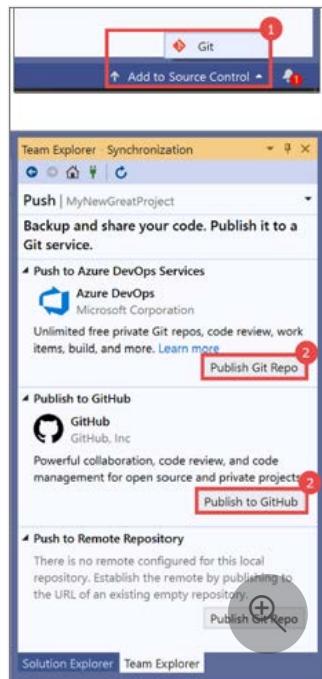
1. Abra la página **Inicio** en **Team Explorer**.
2. Seleccione una solución de la lista de soluciones.



1. Abra la página **Cambiar vistas** en el **Explorador de soluciones**.
2. Seleccione una solución de la lista de soluciones.

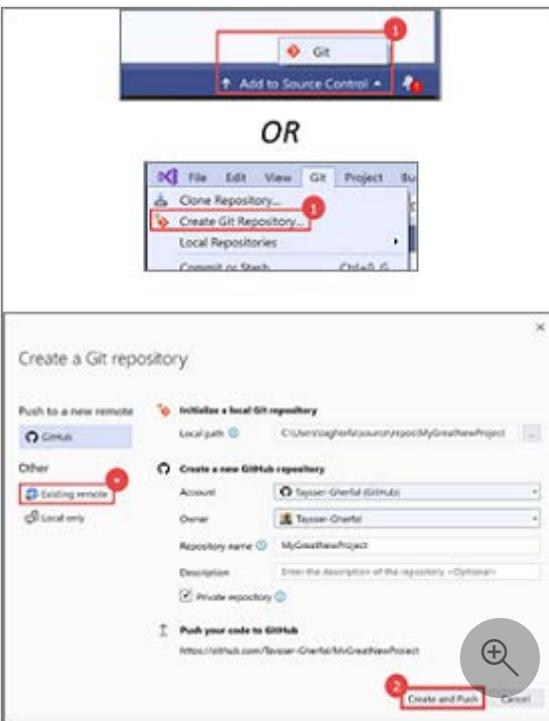
Team Explorer

Adición de una solución al control de origen y creación de un nuevo repositorio



1. Seleccione **Git** en el menú de la barra de estado **Agregar a control de origen**.
2. Seleccione **Publicar**.

Experiencia de Git



1. Seleccione **Git** en el menú de la barra de estado **Agregar a control de origen** o seleccione **Git>Crear repositorio de GIT** en la barra de menús superior de Visual Studio.
2. Seleccione **Crear y enviar los cambios**. Nota: Use la opción **Repositorio remoto existente** si quiere agregar el código a Azure DevOps. En este caso, debe crear primero un repositorio de Azure DevOps.

Sugerencia

La **experiencia de Git** debería conectarse automáticamente al repositorio de Azure DevOps correcto en función del repositorio o la solución que haya abierto. Sin embargo, si necesita conectarse manualmente al repositorio, puede hacerlo mediante Team Explorer. En la barra de menús de Visual Studio, seleccione **Ver>Team Explorer>Administrar conexiones>Conectar**.

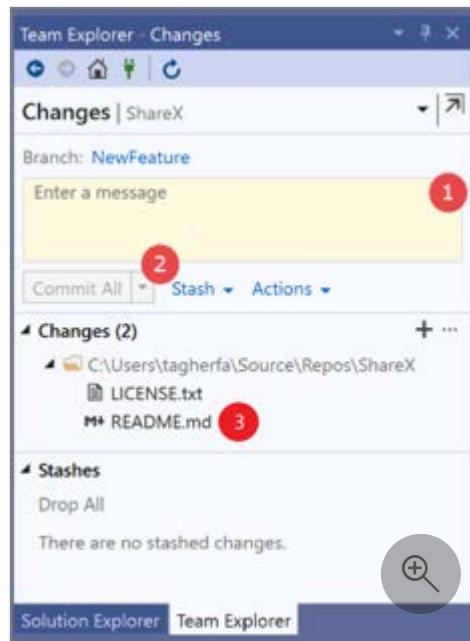
Cambios de Git

Team Explorer

Experiencia de Git

Team Explorer

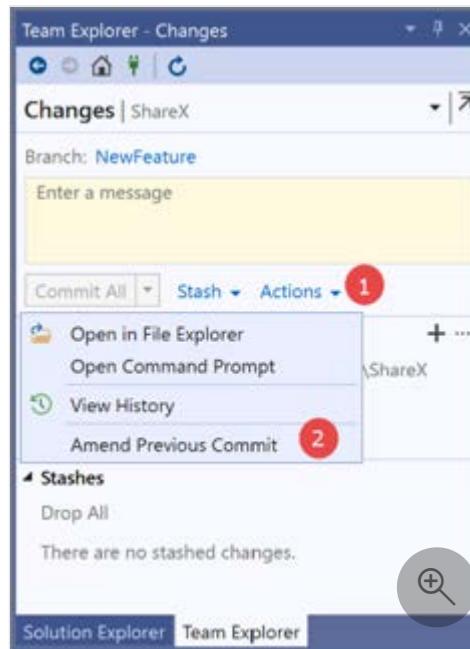
Confirmación y adición al "stage"



1.

1. Escriba un mensaje de confirmación.
2. Seleccione **Confirmar todo**.
3. Para agregar al "stage" archivos específicos, haga clic en ellos con el cursor sobre ellos y haga clic en el icono "+".

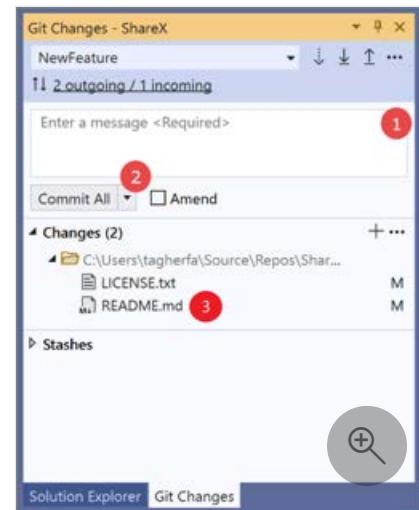
Rectificación de una confirmación



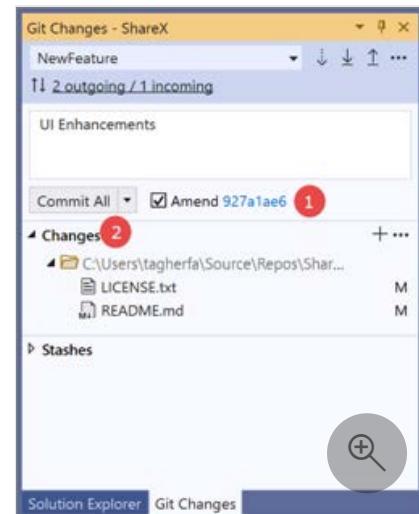
1.

1. Haga clic en el menú desplegable **Acciones**.
2. Seleccione **Rectificar confirmación** anterior.

Experiencia de Git



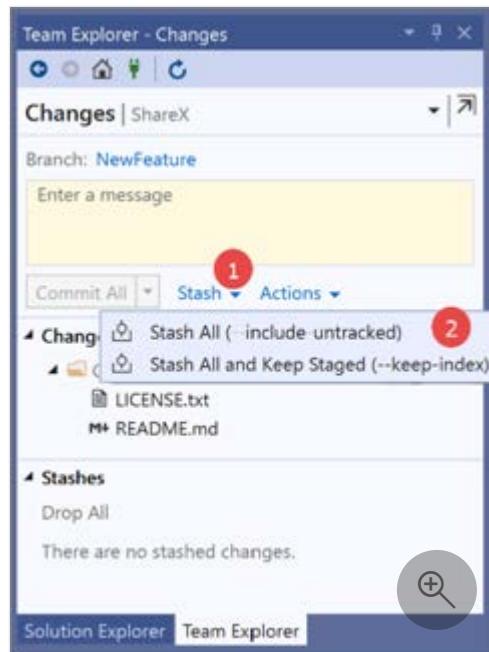
1. Escriba un mensaje de confirmación.
2. Seleccione **Confirmar todo**.
3. Para agregar al "stage" archivos específicos, pase el cursor sobre ellos y haga clic en el icono "+".



1. Haga clic en la casilla **Rectificar**.
2. Haga clic en **Confirmar todo** para confirmar las actualizaciones.

Team Explorer

Guardado provisional de un cambio

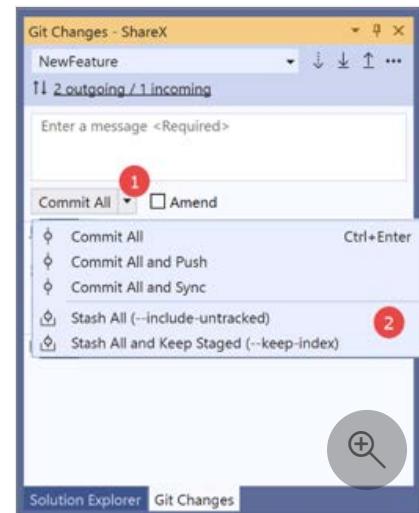


1.

Haga clic en el menú desplegable **Guardar provisionalmente**.

2. Seleccione la opción pertinente para **Guardar provisionalmente**.

Experiencia de Git

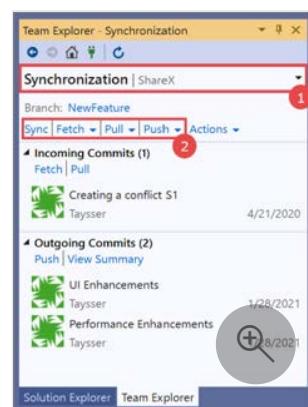


1. Haga clic en el menú desplegable **Confirmar todo**.
2. Seleccione la opción pertinente para **Guardar provisionalmente**.

Synchronization

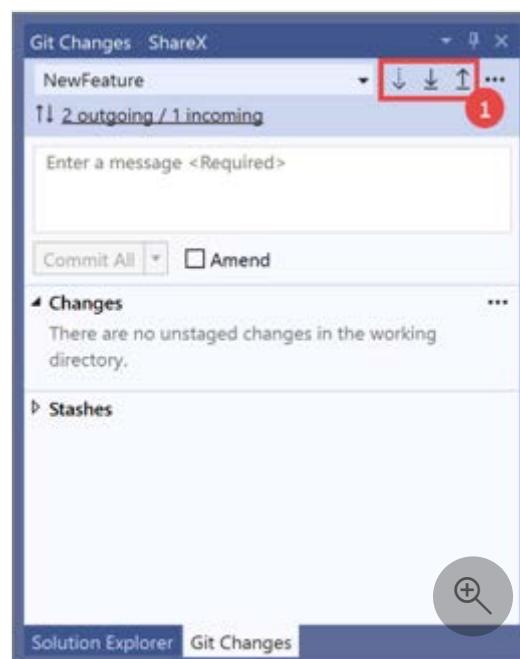
Team Explorer

Captura, extracción e inserción de cambios

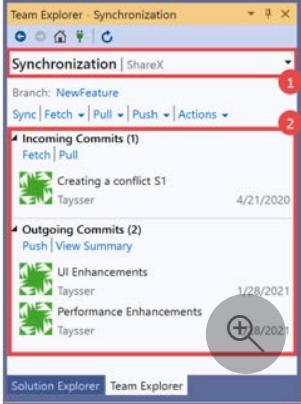


1. Vaya a la página **Sincronización**.
2. Haga clic en la operación de red que prefiera.

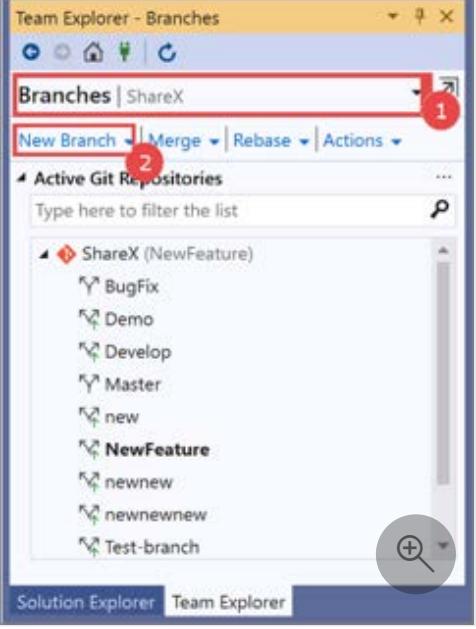
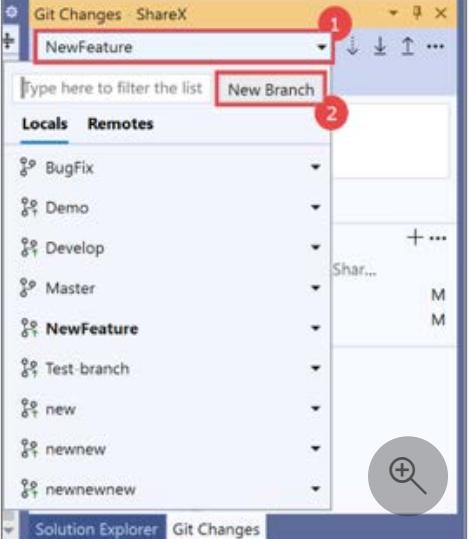
Experiencia de Git



1. Busque los botones de captura, extracción e inserción en la ventana **Cambios de Git**.
2. Haga clic en la operación de red que prefiera.

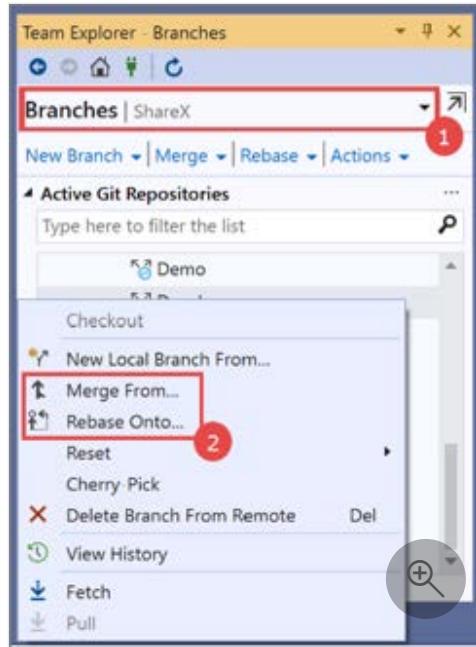
Team Explorer	Experiencia de Git
<p>Visualización de confirmaciones entrantes y salientes</p>  <ol style="list-style-type: none"> 1. Vaya a la página Sincronización. 2. Consulte las listas de entrada y de salida. 	 <ol style="list-style-type: none"> 1. Haga clic en el vínculo saliente/entrante en la ventana Cambios de Git. 2. Para ver las confirmaciones entrantes y salientes, use los iconos de la tabla del gráfico en la parte superior de la ventana del repositorio de Git.

Ramas

Team Explorer	Experiencia de Git
<p>Crear una rama</p>  <ol style="list-style-type: none"> 1. Vaya a la ventana Ramas. 2. Haga clic en Nueva rama. 	 <ol style="list-style-type: none"> 1. En la ventana de Cambios de Git, haga clic en la lista desplegable de la rama. 2. Haga clic en Nueva rama.

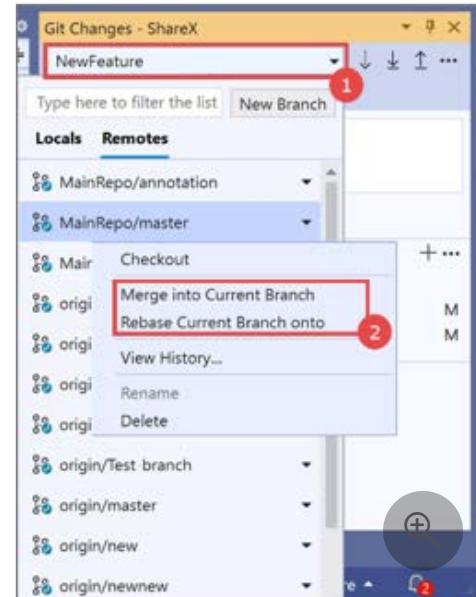
Team Explorer

Obtención de los últimos cambios de una rama remota



1. Vaya a la página **Ramas**.
2. Haga clic con el botón derecho en la rama remota y seleccione **Fusionar mediante combinación de o Fusionar mediante cambio de base en**.

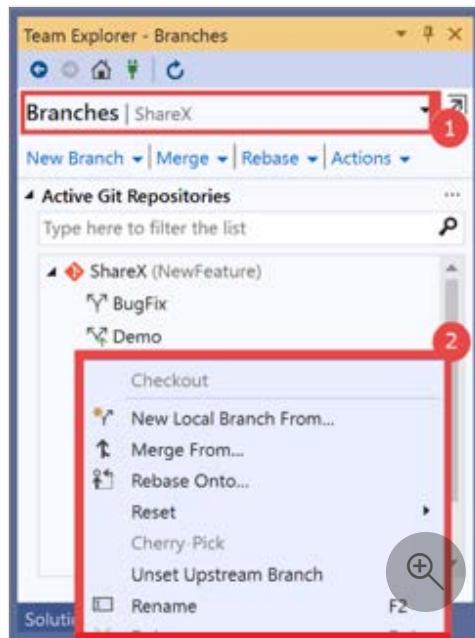
Experiencia de Git



1. Haga clic en la lista desplegable de la rama.
2. En la pestaña **Repositorios remotos**, haga clic en la rama remota y seleccione "Merge" en la rama actual o "Rebase" en la rama actual en.

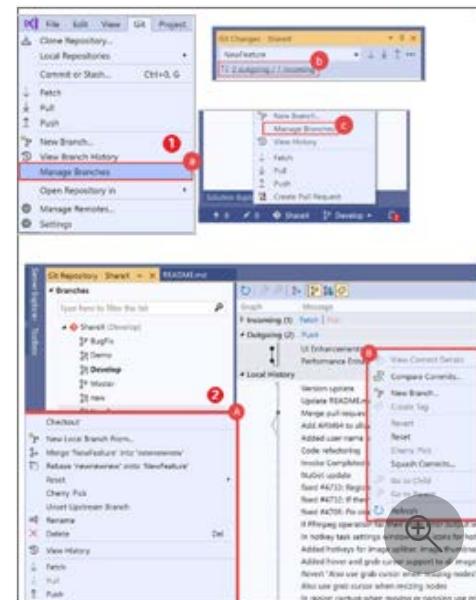
Team Explorer

Administración de ramas



1. Vaya a la ventana **Ramas**.
2. Haga clic con el botón derecho en las ramas que quiera administrar.
3. Consulte el **Historial** de las ramas para administrar las confirmaciones.

Experiencia de Git



1. Vaya a la ventana del repositorio de Git mediante uno de los siguientes puntos de entrada:
 - a. En el menú de Visual Studio de nivel superior, seleccione **Git>Administrar ramas**.
 - b. Seleccione **Cambios de Git>entrantes/salientes**.
 - c. En el menú de la barra de estado de la parte inferior derecha, seleccione **Administrar ramas**.
2. En el menú de nivel superior **Git>Administrar ramas**, realice una de las acciones siguientes:
 - A. Haga clic con el botón derecho en las ramas.
 - B. Seleccione de forma múltiple las confirmaciones que quiera administrar.

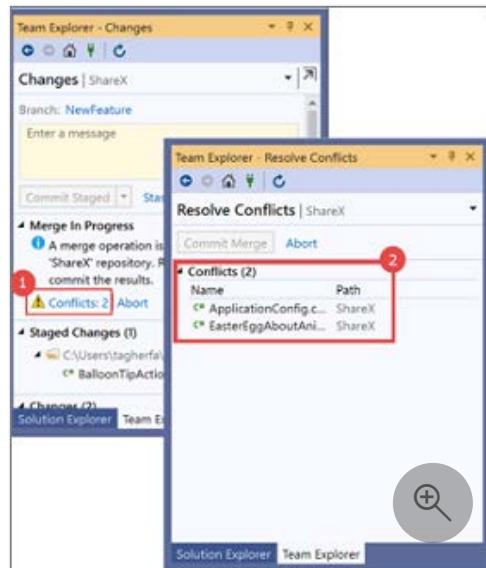
Resolución de conflictos

Team Explorer

Experiencia de Git

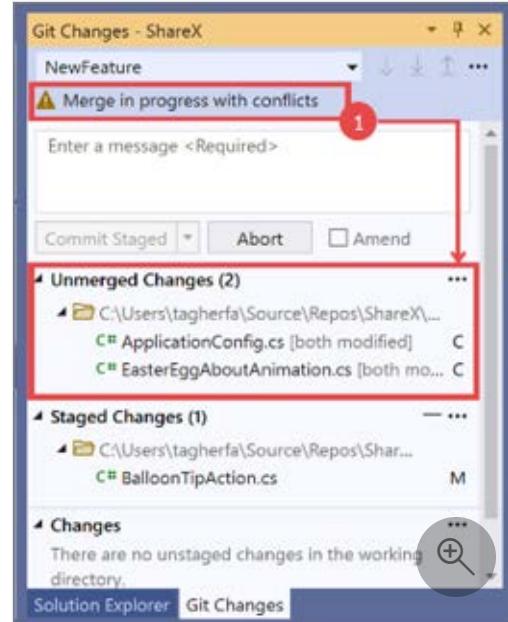
Team Explorer

Acceso a la lista de archivos con conflictos



1. Vaya a la ventana **Resolver conflictos** haciendo clic en el vínculo **Conflictos**.
2. Utilice la lista de **Conflictos** para resolver los conflictos de combinación.

Experiencia de Git



1. Compruebe que aparece **Fusión mediante combinación en curso**.
2. La lista de archivos con conflictos de combinación aparece en la sección **Cambios sin combinar** de la ventana **Cambios de Git**.
Resuelva los conflictos.

Pasos siguientes

Para más información sobre la experiencia de Git, consulte el vídeo más reciente, [Introducción a Git en Visual Studio 2019](#), en YouTube.

Consulte también

- [La experiencia de Git en Visual Studio 2019](#)
- [Trabajar con cuentas de GitHub en Visual Studio](#)