¿Qué es Git?

Git se ha convertido en el estándar mundial para el control de versiones. Entonces, ¿qué es exactamente?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

La flexibilidad y popularidad de Git hacen que sea una excelente opción para cualquier equipo. Muchos desarrolladores y graduados universitarios ya saben cómo usar Git. La comunidad de usuarios de Git ha creado recursos para entrenar a desarrolladores y la popularidad de Git facilita la ayuda cuando sea necesario. Casi todos los entornos de desarrollo tienen compatibilidad con Git y las herramientas de línea de comandos de Git implementadas en cada sistema operativo principal.

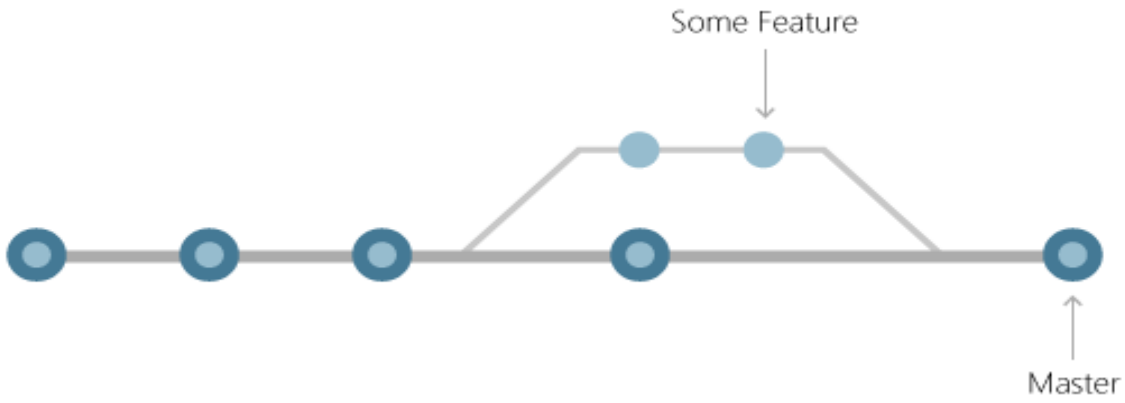
Conceptos básicos de Git

Cada vez que se guarda el trabajo, Git crea una confirmación. Una confirmación es una instantánea de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo almacenado anteriormente. Este diseño difiere de otros sistemas que almacenan una versión inicial de un archivo y mantienen un registro de deltas a lo largo del tiempo.

Las confirmaciones crean vínculos a otras confirmaciones, formando un gráfico del historial de desarrollo. Es posible revertir el código a una confirmación anterior, inspeccionar cómo cambiaron los archivos de una confirmación a la siguiente y revisar información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en Git mediante un hash criptográfico único del contenido de la confirmación. Dado que todo está hash, es imposible realizar cambios, perder información o archivos dañados sin que Git lo detecte.

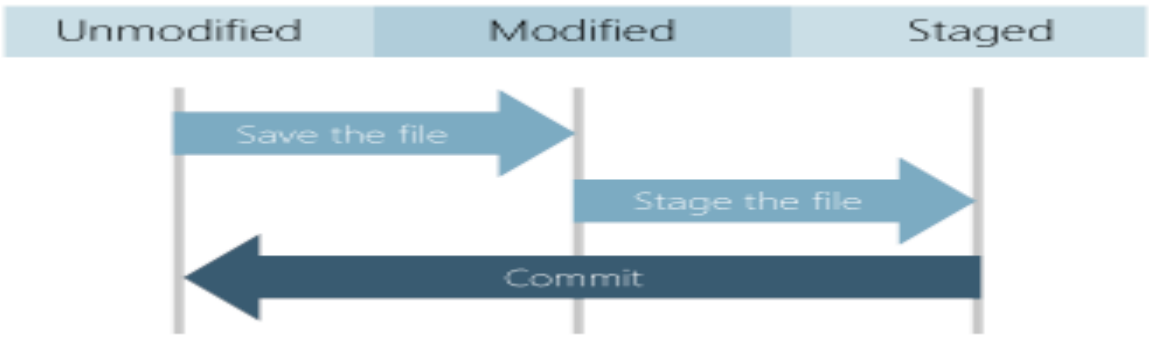
Ramas

Cada desarrollador guarda los cambios en su propio repositorio de código local. Como resultado, puede haber muchos cambios diferentes en función de la misma confirmación. Git proporciona herramientas para aislar los cambios y volver a combinarlos posteriormente. Las ramas, que son punteros ligeros para trabajar en curso, administran esta separación. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal (o troncal) del equipo.



Archivos y confirmaciones

Los archivos de Git se encuentran en uno de los tres estados: modificados, almacenados provisionalmente o confirmados. Cuando se modifica un archivo por primera vez, los cambios solo existen en el directorio de trabajo. Todavía no forman parte de una confirmación o del historial de desarrollo. El desarrollador debe *almacenar* provisionalmente los archivos modificados que se incluirán en la confirmación. El área de almacenamiento provisional contiene todos los cambios que se van a incluir en la siguiente confirmación. Una vez que el desarrollador esté satisfecho con los archivos almacenados provisionalmente, los archivos se empaquetan *como confirmación con* un mensaje que describe lo que ha cambiado. Esta confirmación forma parte del historial de desarrollo.



El almacenamiento provisional permite a los desarrolladores elegir qué cambios de archivo guardar en una confirmación para desglosar los cambios grandes en una serie de confirmaciones más pequeñas. Al reducir el ámbito de las confirmaciones, es más fácil revisar el historial de confirmaciones para buscar cambios de archivo específicos.

Ventajas de Git

Las ventajas de Git son muchas.

Desarrollo simultáneo

Todos tienen su propia copia local de código y pueden trabajar simultáneamente en sus propias ramas. Git funciona sin conexión, ya que casi todas las operaciones son locales.

Versiones más rápidas

Las ramas permiten el desarrollo flexible y simultáneo. La rama principal contiene código estable y de alta calidad desde el que se publica. Las ramas de características contienen trabajo en curso, que se combinan en la rama principal tras la finalización. Al separar la rama de versión del desarrollo en curso, es más fácil administrar código estable y enviar actualizaciones más rápidamente.

Integración integrada

Debido a su popularidad, Git se integra en la mayoría de las herramientas y productos. Cada IDE principal tiene compatibilidad integrada con Git y muchas herramientas admiten la integración continua, la implementación continua, las pruebas automatizadas, el seguimiento de elementos de trabajo, las métricas y la integración de características de informes con Git. Esta integración simplifica el flujo de trabajo diario.

Soporte técnico sólido de la comunidad

Git es de código abierto y se ha convertido en el estándar de facto para el control de versiones. No hay escasez de herramientas y recursos disponibles para que los equipos aprovechen. El volumen de compatibilidad de la comunidad con Git en comparación con otros sistemas de control de versiones facilita la ayuda cuando sea necesario.

Git funciona con cualquier equipo

El uso de Git con una herramienta de administración de código fuente aumenta la productividad de un equipo fomentando la colaboración, aplicando directivas, automatizando procesos y mejorando la visibilidad y la rastreabilidad del trabajo. El equipo puede establecerse en herramientas individuales para el control de versiones, el seguimiento de elementos de trabajo y la integración e implementación continuas. O bien, pueden elegir una solución como GitHub o Azure DevOps que admita todas estas tareas en un solo lugar.

Solicitudes de incorporación de cambios

Use solicitudes de incorporación de cambios para analizar los cambios de código con el equipo antes de combinarlos en la rama principal. Las discusiones en las solicitudes de incorporación de cambios son valiosas para garantizar la calidad del código y aumentar el conocimiento en todo el equipo. Las plataformas como GitHub y Azure DevOps ofrecen una experiencia enriquecida de solicitudes de incorporación de cambios donde los desarrolladores pueden examinar los cambios de archivos, dejar comentarios, inspeccionar confirmaciones, ver compilaciones y votar para aprobar el código.

Directivas de rama

Teams puede configurar GitHub y Azure DevOps para aplicar flujos de trabajo y procesos coherentes en todo el equipo. Pueden configurar directivas de rama para asegurarse de que las solicitudes de incorporación de cambios cumplen los requisitos antes de la finalización. Las directivas de rama protegen ramas importantes mediante la prevención de inserciones directas, la necesidad de revisores y la garantía de compilaciones limpias.

Contenido recomendado

- **¿Qué es el control de versiones? - Azure DevOps**

  Los sistemas de control de versiones son software que ayudan a realizar un seguimiento de los cambios realizados en el código a lo largo del tiempo y crean instantáneas de código a las que un equipo puede revertir si es necesario.

- **Guardado y uso compartido de código con Git - Azure DevOps**

Obtenga información sobre el uso de ramas en Git para guardar, almacenar provisionalmente, insertar, confirmar y compartir cambios de código con el equipo.

- **Configuración de un repositorio - Azure DevOps**

Obtenga información sobre cómo crear un nuevo repositorio de Git o clonar uno existente para que el equipo pueda realizar un seguimiento y administrar su código de desarrollo, correcciones de errores, nuevas características y configuraciones.

¿Cómo se utiliza Github pages?

GitHub es un sitio "social coding". Te permite subir repositorios de código para almacenarlo en el **sistema de control de versiones** Git. Tu puedes colaborar en proyectos de código, y el sistema es código abierto por defecto, lo que significa que cualquiera en el mundo puede encontrar tu código en GitHub, usarlo, aprender de el, y mejorarlo. ¡Tú puedes hacer eso con el código de otras personas tambien! Este artículo provee una guía básica para publicar contenido usando la característica gh-pages de Github.

Publicando contenido

Github es una comunidad muy importante y útil para involucrarse, y Git/GitHub es un sistema de control de versiones muy popular — la mayoría de las empresas de tecnología ahora lo utilizan en su flujo de trabajo. GitHub tiene una característica muy útil llamada GitHub pages, que te permite publicar el código del sitio en vivo en la Web.

Configuración básica de Github

1. Primero que todo, instala Git en tu máquina. Este es el software del sistema de control de versiones subyacente en el que GitHub funciona.
2. Seguido, Regístrate para una cuenta de GitHub. Es simple y fácil.
3. Una vez te hayas registrado, inicia sesión en github.com con tu nombre de usuario y contraseña.

Preparando tu código para subirlo

Tú puedes almacenar cualquier código que tu quieras en un repositorio de Github, pero para usar la característica páginas de Github con pleno efecto, tu código debe estar estructurado como un sitio web típico, por ejemplo que el punto de entrada primario sea un archivo HTML llamado index.html.

La otra cosa que necesitas hacer antes de seguir adelante es inicializar el directorio de código como un repositorio Git. para hacer esto:

1. Apunta la línea de comandos a tu directorio test-site (o como se llame el directorio que contiene tu sitio web). Para esto, usa el comando cd (Es decir "*cambio de directorio*"). Esto es lo que deberías digitar si has puesto tu sitio web en un directorio llamado test-site en tu escritorio:
2. cd Desktop/test-site

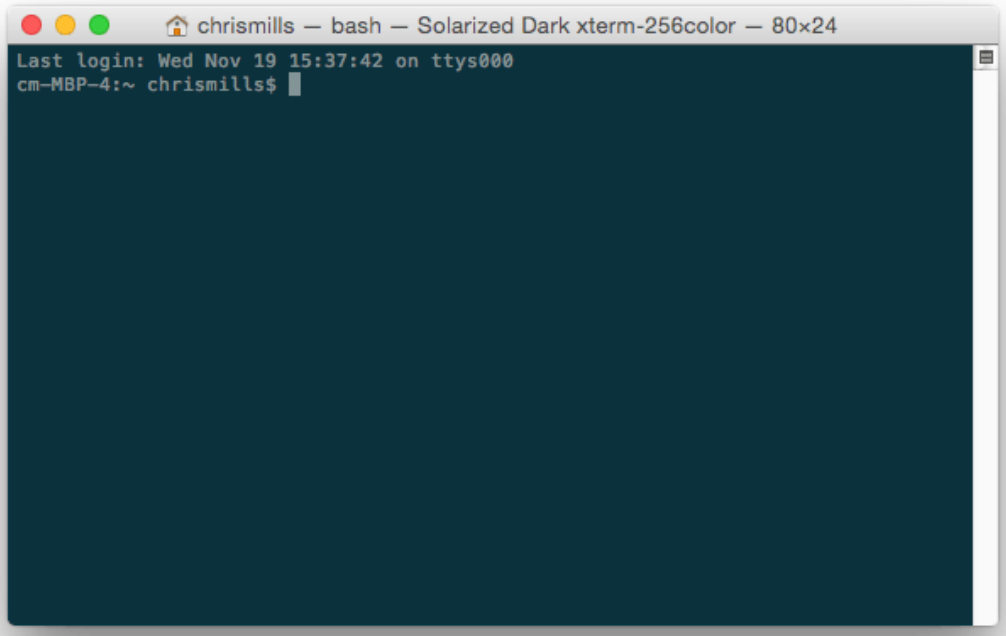
   Copy to Clipboard

3. Cuando la línea comandos está apuntando hacia el interior del directorio de tu sitio web, digita el siguiente comando, que le dice a la herramienta git para convertir el directorio en un repositorio git:
4. git init

   Copy to Clipboard

An aside on command line interfaces

La mejor manera de subir tu código a Github es mediante la linea de comandos — esta es una ventana donde tú escribe comandos para hacer cosas como crear archivos y ejecutar programas, en lugar de hacer clic dentro de una interfaz de usuario. Se verá algo como esto:



Nota: Tú también podrías considerar una interfaz gráfica de usuario de Git para hacer el mismo trabajo, si te sientes incómodo con la linea de comandos.

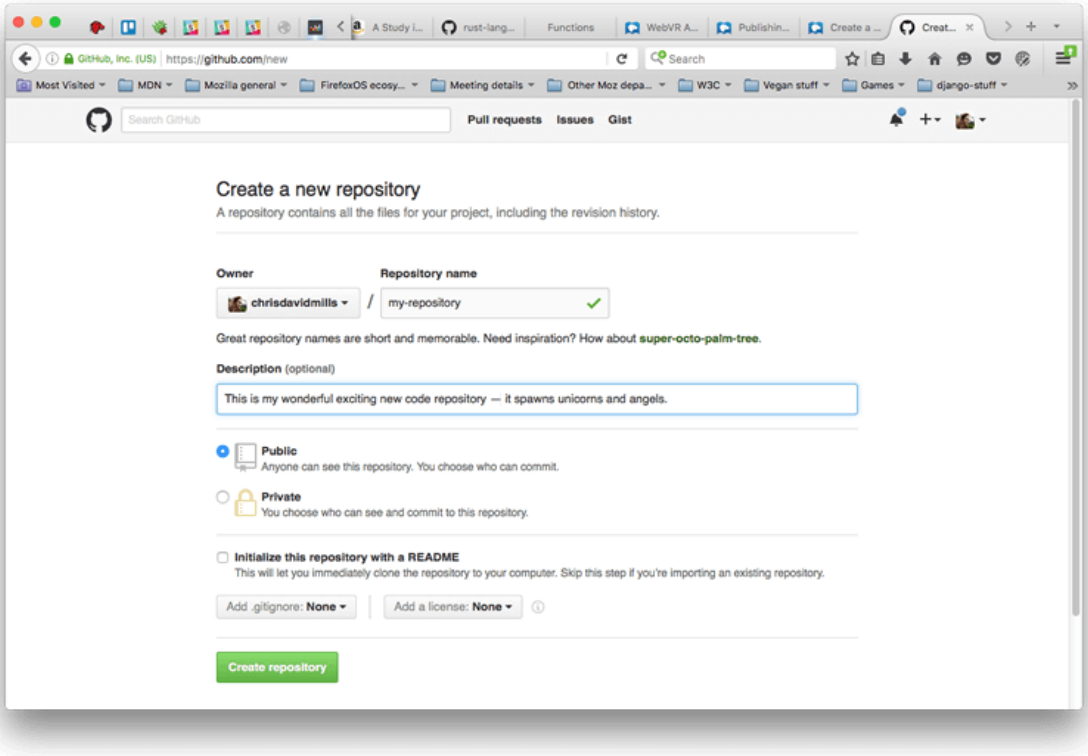
Cada sistema operativo viene con una herramienta de línea de comandos:

- **Windows**: **Command Prompt** se puede acceder pulsando la tecla Windows, tipeando *Command Prompt*, Y elegirlo de la lista que aparece. Nota que Windows tiene sus propias convenciones de comando diferentes de Linux y OS X, así que los comandos abajo pueden variar en su máquina.
- **OS X**: **Terminal** se puede encontrar en *Aplicaciones > Utilidades*.
- **Linux**: Por lo general, puede extraer una terminal con *Ctrl + Alt + T*. Si eso no funciona, busca **Terminal** en una barra de aplicaciones o menú.

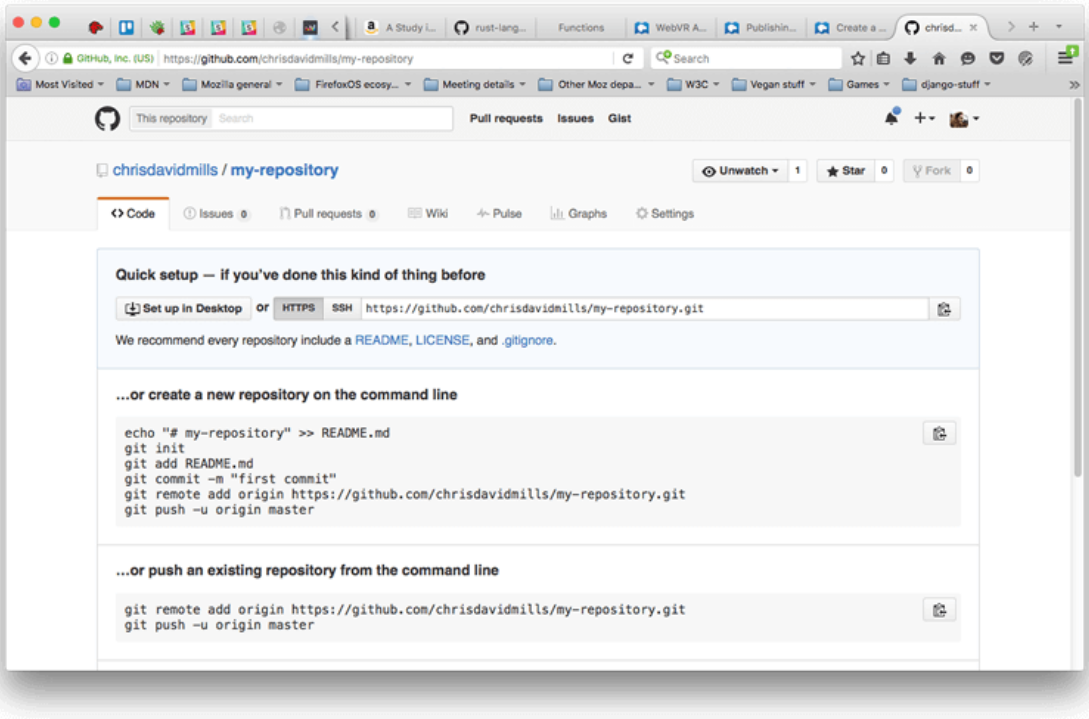
Esto puede parecer un poco espantoso al principio, pero no te preocupes — que pronto conseguiras la caída de los conceptos básicos. Tú le dices a la computadora que haga algo en la terminal, digitando un comando y oprimiendo la tecla Enter, como se ha visto anteriormente.

Creando un repositorio para tu código

1. A continuación, tu necesitas crear un nuevo repositorio para colocar tus archivos en el. Has clic en el signo más (+) en la parte superior derecha de la página de inicio de GitHub, luego escoge *Nuevo Repositorio*.
2. En esta página, en la caja *Nombre del Repositorio*, digita el nombre para tu repositorio de código, por ejemplo *my-repository*.
3. También llena una descripción para decir lo que tu repositorio va a contener. Tu pantalla debe mostrar algo como esto:



4. Has Clic en *Crear repositorio*; Esto debería llevarte a la siguiente página:



Subiendo tus archivos a GitHub

1. En la página actual, tú estás interesado en la sección *…o empujar un repositorio existente desde la línea de comandos*. Tú deberias ver dos lineas de código listado en esta sección. Copia la totalidad de la primera línea, pégala en la línea de comandos, presiona la tecla Enter. El comando debería mostrarte algo como esto:
2. git remote add origin https://github.com/chrisdavidmills/my-repository.git

   Copy to Clipboard

3. A continuación, digita los siguientes dos comandos, presionando Enter despues de cada uno. Estos preparán el código para subirlo a GitHub, y pida a GIt que administre estos archivos.
4. git add --all
5. git commit -m 'adding my files to my repository'

   Copy to Clipboard

6. Por último, empuja el código hasta GitHub yendo a la página de GitHub en la que estas e ingresando en la terminal el segundo de los dos comandos que vimos *…o empuje un repositorio existente desde la sección de línea de comandos*:
7. git push -u origin master

Copy to Clipboard

8. Ahora necesitas crear la rama gh-pages de tu repositorio; actualiza la página actual y verá una página del repositorio algo así como la de abajo. Tú necesitas presionar el boton que dice *Branch:* ***master***, digita *gh-pages* en el campo de texto, luego presiona el boton azul que dice *Create branch: gh-pages*. Esto crea una rama de código especial llamada gh-pages que es publicada en una ubicación especial. La URL toma la forma *username.github.io/my-repository-name*, asi en mi caso de ejemplo, la URL debería ser https://chrisdavidmills.github.io/my-repository. La página mostrada es la página index.html.
9. Navega tu dirección web de GitHub pages en un nuevo ta del navegador, y tu deberias ver tu sitio en linea! Mandalo por correo electrónico a tus amigos y muestra tu dominio.

Nota: Si te atascas, la página de inicio de GitHub Pages también es muy útil.

Un mayor conocimiento de GitHub

Si deseas realizar más cambios en su sitio de prueba y cargarlos en GitHub, simplemente tendrás que realizar el cambio en tus archivos como antes. A continuación, debes introducir los siguientes comandos (pulsando Intro después de cada uno) para empujar los cambios a GitHub:

```
git add --all
git commit -m 'another commit'
git push
```
Copy to Clipboard

Puedes reemplazar *otro commit* con un mensaje más adecuado para describir qué cambio acaba de hacer.

Cómo utilizar Markdown para escribir documentación técnica

Los artículos técnicos de Adobe están escritos con un lenguaje de marcado ligero llamado Markdown, que es fácil de leer y aprender.

Como el contenido Adobe Docs se almacena en GitHub, puede utilizar una versión de Markdown denominada GitHub Flavored Markdown (GFM), que proporciona funcionalidad adicional para satisfacer necesidades comunes de aplicación de formato. Además, el Adobe Markdown ampliado proporciona varias formas que admiten determinadas funciones relacionadas con la ayuda, como notas, sugerencias y vídeos incrustados.

Conceptos básicos de Markdown

Las secciones siguientes describen los conceptos básicos de la creación en Markdown.

Encabezados

Para crear un encabezado, utilice el símbolo de almohadilla (#) al principio de una línea:

```
# This is level 1 (article title)
## This is level 2
### This is level 3
#### This is level 4
##### This is level 5
```

Texto básico

En Markdown, un párrafo no requiere sintaxis especial.

Para aplicar **negrita** al texto, se escribe entre dos asteriscos. Para aplicar *cursiva* al texto, se escribe entre un solo asterisco:

```
This text is **bold**.
This text is *italic*.
This text is both ***bold and italic***.
```

Copy

Para ignorar los caracteres de formato de Markdown, ponga \ antes del carácter:

```
This is not \*italicized\* type.
```

Copy

Listas numeradas y listas con viñetas

Para crear listas numeradas, empiece una línea con `1.` or `1)`, pero no mezcle los formatos dentro de la misma lista. No es necesario especificar los números. GitHub lo hace por usted.

```
1. This is step 1.
1. This is the next step.
1. This is yet another step, the third.
```

Copy

Visualización:

1. This is step 1.
2. This is the next step.
3. This is yet another step, the third.

Para crear listas de viñetas, empiece una línea con *, - o +, pero no mezcle los formatos dentro de la misma lista. (No mezcle formatos de viñetas, como * y +, dentro del mismo documento).

```
* First item in an unordered list.
* Another item.
* Here we go again.
```

Copy

Visualización:

- First item in an unordered list.
- Another item.
- Here we go again.

También puede incrustar listas dentro de listas y añadir contenido entre elementos de la lista.

```
1. Set up your table and code blocks.
1. Perform this step.

   ![screen](/docs/contributor/assets/adobe_standard_logo.png?lang=es)
1. Make sure that your table looks like this:

   | Hello | World |
   |---|---|
   | How | are you? |
1. This is the fourth step.

   >[!NOTE]
   >
   >This is note text.

1. Do another step.
```

Copy

Visualización:

1. Set up your table and code blocks.
2. Perform this step.

3. Make sure that your table looks like this:

Hello	World
How	are you?

4. This is the fourth step.

   [!NOTE]

   This is note text.

5. Do another step.

Tablas

Las tablas no forman parte de la especificación principal de Markdown, pero Adobe las admite en cierta medida. Markdown no admite listas de líneas múltiples en celdas. Lo mejor es evitar el uso de varias líneas en las tablas. Puede crear tablas utilizando la barra vertical (|) para definir columnas y filas. Los guiones crean el encabezado de cada columna, mientras que las barras verticales separan las columnas. Incluya una línea en blanco antes de la tabla para que se muestre correctamente.

```
| Header | Another header | Yet another header |
|--- |--- |--- |
| row 1 | column 2 | column 3 |
| row 2 | row 2 column 2 | row 2 column 3 |
```

Copy

Visualización:

Header	Another header	Yet another header
row 1	column 2	column 3
row 2	row 2 column 2	row 2 column 3

Las tablas sencillas funcionan correctamente en Markdown. Sin embargo, es difícil trabajar con tablas que incluyen varios párrafos o listas dentro de una celda. Para dicho contenido, recomendamos utilizar un formato diferente, como encabezados y texto.

Para obtener más información sobre cómo crear tablas, consulte:

- Organización de información con tablas de GitHub
- La aplicación web Markdown Tables Generator
- Convertir tablas HTML a Markdown

Vínculos

La sintaxis Markdown para un vínculo en línea consiste en la parte `[link text]`, que es el texto que se va a hipervincular, seguido de la parte `(file-name.md)`, que es la URL o el nombre de archivo al que se va a vincular:

```
[link text](file-name.md)
```

```
[Adobe](https://www.adobe.com)
```
Copy

Visualización:

Adobe

Para vínculos a artículos (referencias cruzadas) dentro del repositorio, utilice vínculos relativos. Puede utilizar todos los operandos de vínculos relativos, como ./ (directorio actual), …/ (atrás un directorio) y …/…/ (atrás dos directorios).

```
See [Overview example article](../../overview.md)
```
Copy

Para obtener más información sobre la vinculación, consulte el artículo Vínculos de esta guía sobre las sintaxis de los vínculos.

Imágenes

```
![Adobe Logo](/docs/contributor/assets/adobe_standard_logo.png "Hover text")
```
Copy

Visualización:

NOTA: Para aquellas imágenes que no deban localizarse, cree una carpeta independiente `do-not-localize` en la carpeta de recursos. Por lo general, las imágenes sin texto o las imágenes que solo contienen contenido de muestra se colocan allí. Esto elimina el ruido de la carpeta de recursos y reduce la cantidad de preguntas.

Bloques de código

Markdown admite la colocación de bloques de código tanto en línea como en un bloque "delimitado" independiente entre frases. Para obtener más información, consulte Compatibilidad nativa de Markdown para bloques de código (en inglés).

Utilice comillas invertidas (`) para crear estilos de código en línea dentro de un párrafo. Para crear un bloque de código multilínea específico, agregue tres comillas invertidas (```) antes y después del bloque de código (denominado "bloque de código delimitado" en Markdown y "componente de bloque de código" en AEM). Para bloques de código delimitado, agregue el lenguaje del código después del primer conjunto de comillas invertidas para que Markdown resalte correctamente la sintaxis del código.
Ejemplo: ```` ```javascript ````

Ejemplos:

```
This is `inline code` within a paragraph of text.
```
Copy

Visualización:

This is `inline code` within a paragraph of text.

Este es un bloque de código delimitado:

```
function test() {
 console.log("notice the blank line before this function?");
```

Copy

Extensiones de Markdown personalizadas

En los artículos de Adobe se utiliza la puntuación estándar para la mayoría de los formatos de artículo, como párrafos, vínculos, listas y encabezados. Para obtener un formato enriquecido, los artículos pueden utilizar funciones ampliadas de Markdown, como:

- Bloques de notas
- Vídeos incrustados
- No localizar
- Propiedades del componente, como asignar un ID de encabezado diferente a un encabezado

Utilice el símbolo de citas de bloque de Markdown (>) al principio de cada línea para enlazar un componente ampliado, como una nota. Si necesita utilizar subcomponentes dentro de componentes, agregue un nivel adicional de citas de bloque (> >) para esa sección de subcomponentes. Por ejemplo, una NOTA dentro de una sección DONOTLOCALIZE debe comenzar por > >.

Algunos elementos comunes de Markdown, como encabezados y bloques de código, incluyen propiedades ampliadas. Si necesita cambiar las propiedades predeterminadas, agregue los parámetros entre llaves /{ /} después del componente. Las propiedades ampliadas se describen en contexto.

Bloques de notas

Puede elegir entre estos tipos de bloques de notas para llamar la atención sobre un contenido específico:

- `[!NOTE]`
- `[!TIP]`
- `[!IMPORTANT]`
- `[!CAUTION]`
- `[!WARNING]`
- `[!ADMINISTRATION]`
- `[!AVAILABILITY]`
- `[!PREREQUISITES]`

En general, los bloques de notas deben usarse con moderación porque pueden resultar molestos. Aunque también se admiten bloques de código, imágenes, listas y vínculos, intente que los bloques de notas sean simples y directos.

```
>[!NOTE]
>
>This is a standard NOTE block.
```
Copy

Visualización:

[!NOTE]

This is a standard NOTE block.

```
>[!TIP]
>
>This is a standard tip.
```
Copy

Visualización:

[!TIP]

This is a standard tip.

Vídeos

Los vídeos incrustados no se representan de forma nativa en Markdown, pero puede utilizar esta extensión de Markdown.

```
>[!VIDEO](https://video.tv.adobe.com/v/29770/?quality=12)
```
Copy

Visualización:

Más como esto

El componente “Más como esto” de AEM aparece al final de un artículo. Muestra vínculos relacionados. Cuando se representa el artículo, se le puede dar el mismo formato que a los encabezados de nivel 2 (##) sin que se agreguen al mini-TOC.

```
>[!Related Articles]
```

```
>* [Article 1](https://helpx.adobe.com/es/support/analytics.html)
>* [Article 2](https://helpx.adobe.com/es/support/audience-manager.html)
```
Copy

Visualización:

[!Related Articles]

- Article 1
- Article 2

UICONTROL y DNL

Todo el contenido de ayuda de Markdown se traduce inicialmente mediante traducción automática. Si la ayuda nunca se ha traducido antes, se conserva la traducción automática. Pero si el contenido de ayuda se había traducido anteriormente, el contenido derivado de la traducción automática actuará como referencia mientras el contenido esté en proceso de traducción humana.

``

Durante la traducción automática, los elementos etiquetados con `` se contrastan con el contenido de una base de datos de localización para garantizar su correcta interpretación. En caso de que la IU no esté traducida, esta etiqueta permite que el sistema deje la referencia de la IU en inglés para ese idioma en particular (por ejemplo, las referencias de Analytics en italiano).

Ejemplo:

1. Vaya a la pantalla **Run Process**.
2. Seleccione **File > Print > Print All** para imprimir todos los archivos del servidor.
3. Aparece el cuadro de diálogo Processing Rules.

Fuente:

```
1. Go to the **Run Process** screen.
1. Choose **File > Print > Print All** to print all the files on your server.
1. The Processing Rules dialog box appears.
```
Copy

NOTA: De las tres opciones de etiquetado, esta es la más importante para ofrecer alta calidad y es obligatoria.

`[!DNL]`

Como regla general, se utiliza la lista No traducir para indicar a los motores de traducción automática qué elementos deben conservarse en inglés. Por lo común, dichos elementos serían los nombres de soluciones largos como Adobe Analytics, Adobe Campaign y Adobe Target. Sin embargo, puede haber casos en los que sea necesario forzar a que el motor utilice un término determinado en inglés porque se emplea así de forma específica. Los ejemplos más obvios de este caso serían los nombres cortos de las soluciones, como Analytics, Campaign, Target, etc. Sería difícil para una máquina discernir si estos nombres hacen referencia a soluciones o son términos generales. La etiqueta también puede utilizarse para nombres o funciones de terceros que siempre permanecen en inglés, o para secciones de texto más cortas como una frase u oración que deban permanecer en inglés.

Ejemplo:

- Con Target, puede crear pruebas A/B para encontrar lo óptimo
- Adobe Analytics es una potente solución para recopilar análisis en el sitio. Analytics también puede ayudarle con los informes para asimilar fácilmente esos datos.

Fuente:

```
* With Target, you can create A/B tests to find the optimal

* Adobe Analytics is a powerful solution to collect analytics on your site. Analytics can also help you with reporting to easily digest that data.
```

Copy

Problemas y soluciones

Texto alternativo

El texto alternativo que contiene guiones bajos no se representa correctamente. Por ejemplo, en lugar de poner esto:

```
![Settings_Step_2](/assets/settings_step_2.png)
```

Copy

Lo mejor es utilizar guiones (-) en lugar de guiones bajos (_) en los nombres de archivo.

```
![Settings-Step-2](/assets/settings-step-2.png)
```

Copy

Apóstrofes y comillas

Si copia texto en un editor de Markdown, el texto puede contener apóstrofos “inteligentes” (curvos) o comillas. Deben codificarse o cambiarse por apóstrofos básicos o comillas. De lo contrario, aparecerán caracteres extraños como este cuando se publique el archivo: Itâ€™s

Estas son las codificaciones para las versiones “inteligentes” de estos signos de puntuación:

- Comilla tipográfica izquierda (apertura): `&#8220;`
- Comilla tipográfica derecha (cierre): `&#8221;`
- Comilla tipográfica derecha simple (cierre) a la derecha o apóstrofo: `&#8217;`
- Comilla tipográfica izquierda simple (apertura; poco usada): `&#8216;`

Antilambdas

Si utiliza antilambdas en un texto (que no sea código) del archivo, por ejemplo, para denotar un marcador de posición, debe codificarlos manualmente. De lo contrario, Markdown considera que se trata de una etiqueta HTML.

Por ejemplo, codifique `<script name>` como `&lt;script name&gt;`

El símbolo “et” en los títulos

El símbolo “et” (&) no se permite en los títulos. Utilice “y” o la codificación `&`.