



Conceptos básicos de Git

Curso de Fundamentos

Introducción

En esta lección, cubriremos los comandos comunes de Git utilizados para administrar sus proyectos y para subir su trabajo a GitHub. Nos referimos a estos comandos como el **flujo de trabajo básico de Git**. Cuando estás usando Git, estos son los comandos que usarás entre el 70 y el 80% del tiempo. Si puedes bajarlos, ¡estarás más de la mitad del dominio de Git!

Resumen de la lección

Esta sección contiene una visión general de los temas que aprenderá en esta lección.

- Cómo crear un repositorio en GitHub.
- Cómo obtener archivos desde y hasta GitHub.
- Cómo tomar "instantáneas" de su código.

Asignación

¡Antes de empezar!

Contenido de la lección

[Introducción](#)[Resumen de la lección](#)[Asignación](#)[Comprobar de conocimientos](#)[Recursos adicionales](#)

- Github actualizó recientemente la forma en que nombra la rama predeterminada. Esto significa que debe asegurarse de que está utilizando una versión reciente de git (2.28 o posterior). Puedes comprobar tu versión ejecutando: `git --version`
- Si aún no lo ha hecho, establezca su rama predeterminada local de Git en `main`. Puedes hacerlo corriendo: `git config --global init.defaultBranch main`
- For more information on the change from `master` to `main` see [GitHub's Renaming Repository](#).

Crear el repositorio

1. Ya deberías haber creado una cuenta de GitHub en la lección [Configuración de Git](#).
2. Desde la página de inicio de GitHub, cree un nuevo repositorio haciendo clic en el botón "+" en la esquina superior derecha y seleccionando "Nuevo repositorio". *Si está utilizando una ventana gráfica más pequeña, ese botón puede estar oculto. En ese caso, haga clic en su foto de perfil en la esquina superior derecha y el botón aparecerá junto a su nombre de perfil.*
3. Asigne a su repositorio el nombre "git_test" en el campo de entrada del nombre del repositorio. Compruebe

"Añadir un archivo README". Luego cree el repositorio haciendo clic en el botón "Crear repositorio" en la parte inferior de la página.

4. Esto te redirigirá a tu nuevo repositorio en GitHub. Para prepararse para copiar (clonar) este repositorio en su máquina local, haga clic en el botón verde "Código", que debería estar a la derecha del botón que muestra la rama actual (normalmente mostrará la rama *principal*). Luego seleccione la opción SSH en la sección "Clonar" y copie la línea debajo de ella. **NOTA: DEBE hacer clic en la opción SSH para obtener la URL correcta.**
5. Let's use the command line on your local machine to create a new directory for all of your Odin projects. Create a directory called `repos` with the `mkdir` command in your home folder. Your home folder is represented by `~`. Note that depending on your OS, there may be some [home directory variation](#) - sometimes `~` stands for `/Users/your_username` and sometimes it stands for `/home/your_username`. If you're not sure if you're in your home folder, just type `cd ~`. Once it's made, move into it with the `cd` command.

```
1 | mkdir repos
2 | cd repos/
```

6. Now it's time to clone your repository from GitHub into your newly created `repos` directory with `git clone` followed by the URL you copied in the last step. The full command should look similar to `git clone git@github.com:USER-NAME/REPOSITORY-NAME.git`. If your URL looks like `https://github.com/USER-NAME/REPOSITORY-NAME.git`, you have selected the HTTPS option, not the required SSH option.

```
1 | git clone git@github.com:
```

7. That's it! You have successfully connected the repository you created on GitHub to your local machine. To test this, `cd` into the new **git_test** folder that was downloaded and then enter `git remote -v` on your command line. You should see an output similar to the following, where `USER-NAME` is your GitHub username:

```
1 | origin git@github.com:USE
2 | origin git@github.com:USE
```

Esto muestra la URL del repositorio que creaste en GitHub, que es el control remoto de tu copia local. Es posible que también haya notado el *origen* de la

palabra al comienzo de la salida `git remote -v`, que es el nombre de su conexión remota. El nombre "origen" es tanto el predeterminado como la convención para el repositorio remoto, pero podría haber sido llamado fácilmente "party-parrot" o "dancing-banana". (No te preocupes por los detalles del origen por ahora; volverá a surgir cerca del final de este tutorial).

Usa el flujo de trabajo de Git

1. Create a new file in the `git_test` folder called "hello_world.txt" with the command `touch hello_world.txt`.
2. Type `git status` in your terminal. In the output, your `hello_world.txt` file should be shown with a red text color as well as listed in a section titled "Untracked files". This means that the file is not yet staged.
3. Type `git add hello_world.txt`. This command adds your hello_world.txt file to the staging area in Git. The staging area is part of the two-step process for making a commit in Git. Think of the staging area as a "waiting room" for your changes until you commit them. Now, type `git status` again. In the output, notice that your file is now shown in green and in a section titled "Changes to be committed", which means that this file is now in the staging area.

4. Type `git commit -m "Add hello_world.txt"` and then type `git status` once more. The output should now say: *"nothing to commit, working tree clean"*, indicating your changes have been committed. Don't worry if you get a message that says *"upstream is gone"*. This is normal and only shows when your cloned repository currently has no branches. It will be resolved once you have followed the rest of the steps in this project.

El mensaje, *"Su rama está por delante de 'origen/principal' por 1 confirmación"* solo significa que ahora tiene instantáneas más nuevas que las que están en su repositorio remoto. Subirás tus instantáneas más abajo en esta lección.

5. Type `git log` and look at the output. You should see an entry for your *"Add hello_world.txt"* commit. You will also see other details which include the author who made the commit and the date and time of when the commit was made. If your terminal is stuck in a screen with (END) at the bottom, just press "q" to escape. You can configure settings for this later, but don't worry about it too much for now.

Modificar un archivo o dos

1. Open README.md in your text editor

of choice. In this example, we will open the directory in Visual Studio Code by using the command `code .` inside your repository.

Usuarios de MacOS: Si su terminal dice *"comando no encontrado: código"*, debe volver a [Conceptos básicos de la línea de comandos](#) y seguir las instrucciones proporcionadas para permitir que este comando funcione.

2. Add "Hello Odin!" to a new line in `README.md` and save the file with `Ctrl` + `S` (Mac: `Cmd` + `S`).
3. Go back to your terminal, or if you're using Visual Studio Code, open the built-in terminal by pressing `Ctrl` + ``` (backtick). Then type `git status`. Notice how the output is similar to when we created our `hello_world.txt` file before adding it to the staging area, except the `README.md` file is listed in a section titled "Changes not staged for commit". The meaning is similar to the "Untracked files" section in that the file is not yet added to the staging area.
4. Añade `README.md` al área de preparación con `git add README.md`.
5. Can you guess what `git status` will output now? `README.md` will be displayed in green text in the section

titled "Changes to be committed". That means `README.md` has been added to the staging area. The file `hello_world.txt` will not show up because it has not been modified since it was committed.

6. Open `hello_world.txt`, add some text to it, save it and stage it. You can use `git add .` to add all files in the current directory and all subsequent directories to the staging area. Then, type `git status` once more, and everything should now be in the staging area.
7. Finally, let's commit all of the files that are in the staging area and add a descriptive commit message. `git commit -m "Edit README.md and hello_world.txt"`. Then, type `git status` once again, which will output the same *"nothing to commit"* message as when we previously made a commit.
8. Eche un último vistazo a su historial de confirmación escribiendo `git log`. Ahora deberías ver tres entradas.

Envíe su trabajo a GitHub

Finalmente, subamos tu trabajo al repositorio de GitHub que creaste al comienzo de este tutorial.

1. Type `git push`. To be more specific, type `git push origin main`. Since

you are not dealing with another branch (other than *main*) or a different remote (as mentioned above), you can leave it as `git push` to save a few keystrokes. **NOTE: If at this point you receive a message that says "Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.", you have followed the steps incorrectly and cloned with HTTPS, not SSH. Please follow the steps for [switching remote URLs from HTTPS to SSH](#) to change your remote to SSH, then attempt to push to Github.**

2. Type `git status` one final time. It should output *"Your branch is up to date with 'origin/main'. nothing to commit, working tree clean"*.
3. Cuando actualice su página de repositorio en GitHub, debería ver los archivos README.md y hello_world.txt que acaba de enviar allí desde su máquina local.

Evite editar directamente en GitHub

Al intentar hacer cambios rápidos en los archivos de su repositorio, como intentar corregir un error tipográfico en su README.md, es posible que se sienta tentado a realizar este cambio directamente a través de Github. Sin embargo, es mejor evitar esto, ya que causará problemas que

requieren más conocimientos avanzados de Git de los que queremos repasar en esta etapa (se cubre en una lección futura). Por ahora, se recomienda realizar cualquier cambio a través de sus archivos locales, luego confirmarlos y enviarlos usando comandos de Git en su terminal una vez que estén listos.

Hoja de trucos

Esta es una lista de referencia de los comandos de Git más utilizados. (Podrías considerar marcar esta práctica página). Intenta familiarizarte con los comandos para que eventualmente puedas recordarlos todos:

- Comandos relacionados con un repositorio remoto:
 - `git clone git@github.com:USER-NAME/REPOSITORY-NAME.git`
 - `git push` or `git push origin main`
(Both accomplish the same goal in this context)
- Comandos relacionados con el flujo de trabajo:
 - `git add .`
 - `git commit -m "A message describing what you have done to make this snapshot different"`
- Comandos relacionados con la comprobación del estado o el historial de registro
 - `git status`

- `git log`

La sintaxis básica de Git es `program | action | destination`.

Por ejemplo,

- `git add .` se lee como `git | add | .`, donde el punto representa todo en el directorio actual;
- `git commit -m "message"` is read as `git | commit -m | "message"`; and
- `git status` se lee como `git | status | (no destination)`

Mejores prácticas de Git

Hay mucho que aprender sobre el uso de Git.

Pero vale la pena tomarse el tiempo para destacar algunas de las mejores prácticas para que pueda ser un mejor colaborador. Git no solo es útil cuando se colabora con otros. También es útil cuando se trabaja de forma independiente. Confiarás cada vez más en tu propio historial de confirmación en el futuro cuando vuelvas a visitar el código antiguo.

Dos mejores prácticas útiles a considerar son **las confirmaciones atómicas** y aprovechar esas confirmaciones atómicas para hacer que sus mensajes de confirmación sean más útiles para futuros colaboradores.

Una confirmación atómica es una confirmación que incluye cambios relacionados con una sola característica o tarea de su programa. Hay dos razones principales para hacer esto: primero, si algo que cambia resulta causar algunos


problemas, es fácil revertir el cambio específico sin perder otros cambios; y segundo, le permite escribir mejores mensajes de confirmación.

¡Aprenderás más sobre cómo es un buen mensaje de confirmación en una lección futura!

Cambiar el editor de mensajes de confirmación de Git

If you are using *Visual Studio Code* (and you should be if you're following this curriculum), there's a way to ensure that if you use `git commit` without the message flag (`-m`), you won't get stuck writing your commit message in [Vim](#).

Cambiar el editor de mensajes predeterminado es una buena idea en caso de que omita accidentalmente la bandera, a menos que prefiera usar Vim. No hay ningún inconveniente en cambiarlo, porque tendrá la opción de escribir sus mensajes de confirmación en el terminal o en la comodidad de VS Code.

El siguiente comando establecerá esta configuración. Escriba (o copie y pegue) este comando en su terminal y presione .

```
1 | git config --global core.editor "
```

No habrá confirmación ni salida en el terminal después de introducir este comando.

With that done, you can now choose to use either `git commit -m "your message here"` or `git commit` to type your message with Visual Studio Code!

To make a commit with Visual Studio Code as the text editor, just type `git commit`. After you hit `Enter` a new tab in VS Code will open for you to write your commit message. You may provide more details on multiple lines as part of your commit message. After typing your commit message, save it `Ctrl` + `S` (Mac: `Cmd` + `S`) and close the tab. If you return to the command line, you will see your commit message and a summary of your changes.

Comprobar de conocimientos

Las siguientes preguntas son una oportunidad para reflexionar sobre temas clave de esta lección. Si no puede responder a una pregunta, haga clic en ella para revisar el material, pero tenga en cuenta que no se espera que memorice o domine este conocimiento.

- [¿Cómo se crea un nuevo repositorio en GitHub?](#)
- [¿Cómo se copia un repositorio a su máquina local desde GitHub?](#)
- [¿Cuál es el nombre predeterminado de tu conexión remota?](#)
- [Explain what `origin` is in `git push origin main`.](#)
- [Explain what `main` is in `git push origin main`.](#)
- [Explica el sistema de dos etapas que utiliza Git para guardar archivos.](#)
- [¿Cómo compruebas el estado de tu repositorio actual?](#)


- [¿Cómo se añaden archivos al área de preparación en Git?](#)
- [¿Cómo se confirman los archivos en el área de preparación y se añade un mensaje descriptivo?](#)
- [¿Cómo envías tus cambios a tu repositorio en GitHub?](#)
- [¿Cómo ves el historial de tus compromisos anteriores?](#)

Recursos adicionales

Esta sección contiene enlaces útiles a contenido relacionado. No es obligatorio, así que considérelolo como suplemento.

- Parece que esta lección aún no tiene recursos adicionales. Ayúdanos a ampliar esta sección contribuyendo a nuestro plan de estudios.

 [Mejorar en GitHub](#)

 [Informar de un problema](#)

[Ver registro de cambios de la lección](#)



Ver
curso

Marcar completo



Próxima
lección

¡Apóyanos!

El Proyecto Odin está financiado por la comunidad. ¡Únete a nosotros para empoderar a los estudiantes de todo el mundo apoyando a The Odin Project!

[más información](#)

[Donar ahora](#)



THE ODIN PROJECT

Educación de codificación de alta calidad mantenida por una comunidad de código abierto.



Sobre nosotros

[acerca de](#)

[EQUIPO](#)

[blog](#)

[Casos de éxito](#)

soporte

[preguntas frecuentes](#)

[Contribuir](#)

[Contacta con nosotros](#)

Guías

[Guías de la comunidad](#)

[Guías de instalación](#)

LEGAL

[Términos](#)

[PRIVACIDAD](#)