

Git, GitHub y Markdown

Adolfo Sanz De Diego

Mayo 2016

1 Acerca de

1.1 Autor

- **Adolfo Sanz De Diego**
 - Blog: asanzdiego.blogspot.com.es
 - Correo: asanzdiego@gmail.com
 - GitHub: github.com/asanzdiego
 - Twitter: twitter.com/asanzdiego
 - LinkedIn: in/asanzdiego
 - SlideShare: slideshare.net/asanzdiego

1.2 Licencia

- **Este obra está bajo una licencia:**
 - Creative Commons Reconocimiento-CompartirIgual 3.0

1.3 Fuente

- Las slides y sus fuentes las podéis encontrar en:
 - <https://github.com/asanzdiego/curso-git-github-markdown-2016>

2 Introducción

2.1 Objetivos

1. Conocer las **características de Git** y ser capaz de instalarlo y configurarlo.
2. Conocer y ser capaz de usar los **comandos de Git**.
3. Conocer las **características de GitHub** y ser capaz de crear una cuenta y configurarla.
4. Ser capaz de **crear y clonar repositorios** en GitHub.
5. Conocer y ser capaz de **usar las principales características de GitHub**.
6. Conocer la **sintaxis del lenguaje Markdown**.

2.2 Indice

- **Bloque 1**
 - Uso básico de Git y GitHub
- **Bloque 2**
 - Uso avanzado de Git y GitHub
- **Bloque 3**
 - Markdown

2.3 Enlaces imprescindibles

- Pro GIT (sobre todo temas 1, 2, 3 y 6):
 - <https://git-scm.com/book/es/v2>
- Página oficial de Git:
 - <https://git-scm.com/>
- Página oficial de GitHub:
 - <https://github.com/>
- Chuleta de la sintaxis de Markdown:
 - <http://warpedvisions.org/projects/markdown-cheat-sheet>

2.4 Otros enlaces de interés

- Aprender GIT... y de camino GitHub:
 - <https://github.com/oslugr/curso-git>
- Minitutorial de GIT:
 - <https://try.github.io/>
- Tutorial de GIT de codecademy;
 - <https://www.codecademy.com/learn/learn-git>
- How GitHub Uses GitHub to Build GitHub:
 - <http://zachholman.com/talk/how-github-uses-github-to-build-github/>

3 Uso básico de Git

3.1 Sistema Control de Versiones

"Sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante."

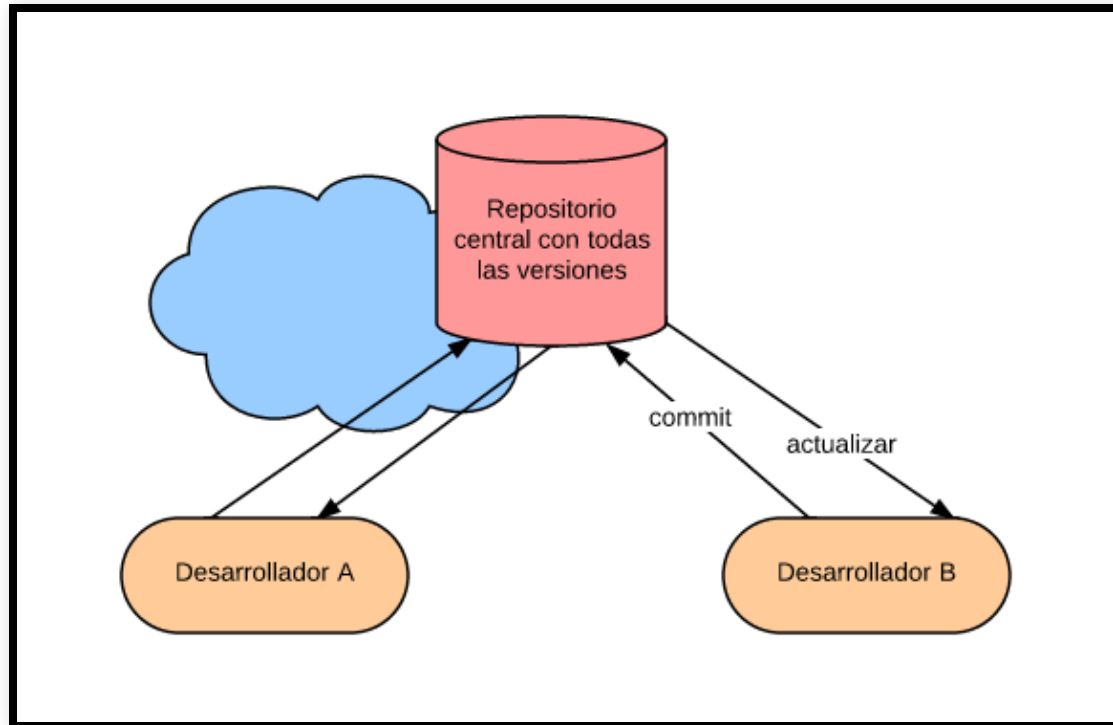
- <https://git-scm.com/book/es/v2/Empezando-Acerca-del-control-de-versiones>

3.2 VCS Locales

- **Lo más simple:** hacer copias de directorios.
- Aparecieron **BD en local** que guardan el registro de los cambios realizados a los archivos.

3.3 VCS Centralizados

- Un **servidor central** que guarda los cambios.



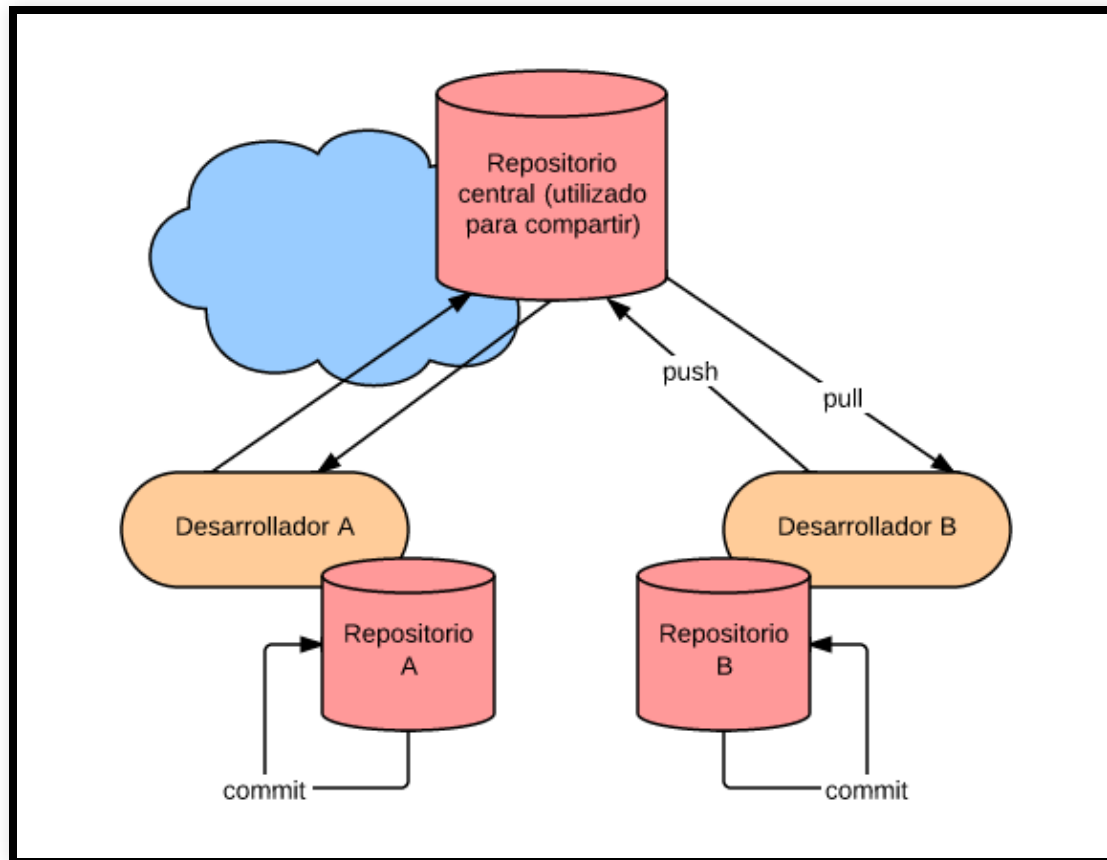
VCS Centralizado

3.4 Pros y Contras VCS Centralizados

- **Pros:** más colaborativo que el local.
- **Contras:** dependes de un servidor central.

3.5 VCS Distribuidos

- Cada cliente **no solo descarga la última copia, sino todo el repositorio.**



VCS Distribuido

3.6 Ventajas VCS Distribuidos

- Puedes seguir trabajando aunque el repositorio remoto esté caído.
 - **más autonomía**
- La información está más replicada.
 - **menos vulnerable**
- Permite pruebas en local y subir solo lo relevante.
 - **más limpieza**

3.7 Características de Git

- Creado por **Linux Torvalds**, líder del equipo del kernel Linux.
- Objetivos cuando se creó:
 - **Rápido**
 - **Sencillo**
 - **Multi rama**
 - **Distribuido**
 - **Grandes proyectos**

3.8 Instalación

- Windows: <https://git-scm.com/download/win>
- Mac: <https://git-scm.com/download/mac>
- Linux: <https://git-scm.com/download/linux>

3.9 Configuración inicial

```
git config --global user.name "Nombre que quieras mostrar"
```

```
git config --global user.email "correo@electronico.es"
```

3.10 GUIs

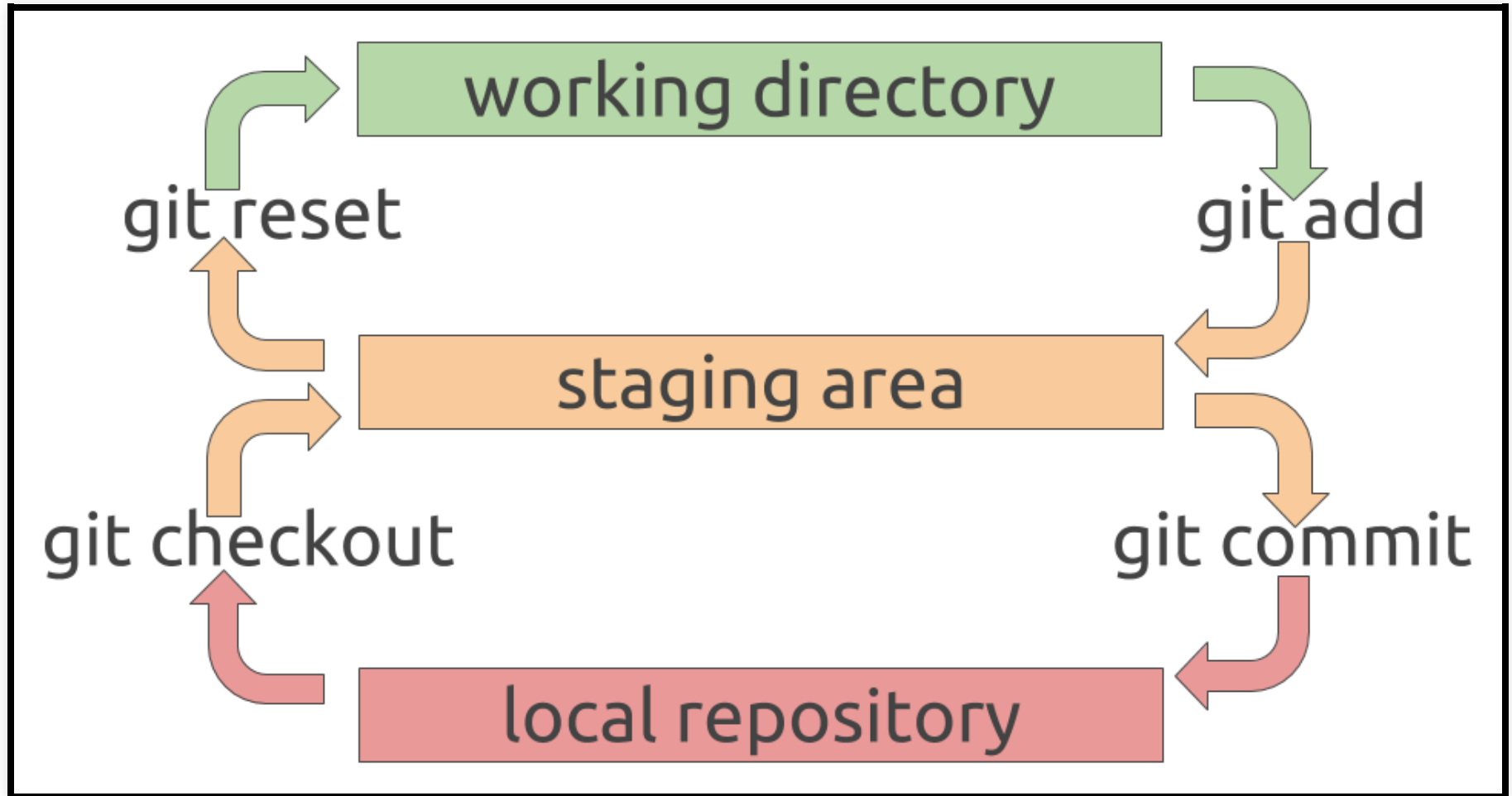
- <https://git-scm.com/downloads/guis>

3.11 Inicializar un repositorio

- Crea el **subdirectorio .git** con archivos de git para gestionar el repositorio.

```
git init
```

3.12 El área de staging



Staging Area

3.13 Ver el estado de los archivos

- Importante saber el **estado** de los archivos.

```
git status
```


3.14 Ver las diferencias

- Podemos ver las **diferencias** entre el área de staging y el área de trabajo.

```
git diff
```

3.15 Añadir archivos

- Podemos **añadir** los cambios de un fichero (o varios) al área de staging (desde el área de trabajo).

```
git add nombre-del-fichero
```

```
git add *.extension
```

```
git add -A
```

3.16 Borrar archivos

- Podemos **borrar archivos** del área de staging (también lo borrará del área de trabajo)

```
git rm nombre-del-fichero
```

3.17 Mover/renombrar archivos

- Podemos **mover/renombrar archivos** en el área de staging (también lo hará en el área de trabajo)

```
git mv antiguo-nombre-del-fichero nuevo-nombre-del-fichero
```

3.18 Resetar archivos

- Para **resetear** los cambios de un fichero (o varios) al area de trabajo (desde el area de staging).

```
git reset nombre-del-fichero
```

3.19 Grabar los cambios

- Para **grabar** los cambios realizados al repositorio (desde el área de staging).

```
git commit -m "mensaje corto descriptivo con los cambios"
```

3.20 Deshacer los cambios

- Para **deshacer** los cambios de un fichero (o varios) al area de staging (desde el repositorio).

```
git checkout nombre-del-fichero
```

3.21 Listado de cambios

- Para ver el **listado de cambios** realizados en el repositorio.

```
git log
```


3.22 Alias

- Podemos crear **alias**.

```
git config --global alias.list 'log --oneline --decorate --graph --all'
```

3.23 Ignorar archivos

- Podemos ignorar archivos añadiéndolos al fichero **.gitignore**.

3.24 Creando etiquetas

- Existen etiquetas **ligeras**, y etiquetas **anotadas** (iguales pero estas con más información)

```
git tag nombre-etiqueta-lijera
```

```
git tag -a nombre-etiqueta-anotada -m "mensaje que acompaña a la etiquet
```

3.25 Etiquetas tardías

- Se puede crear una etiqueta **conociendo el hash del commit** (verlo con git log).

```
git tag -a nombre-etiqueta-anotada -m "mensaje que acompaña a la etiquet
```

3.26 Ver una etiqueta

- Podemos **ver información concreta de una etiqueta.**

```
git show nombre-etiqueta
```

3.27 Sacar una etiqueta

- No podemos sacar una etiqueta, pero podemos **colocar en nuestro directorio de trabajo una versión que coincida con alguna etiqueta, creando una rama nueva:**

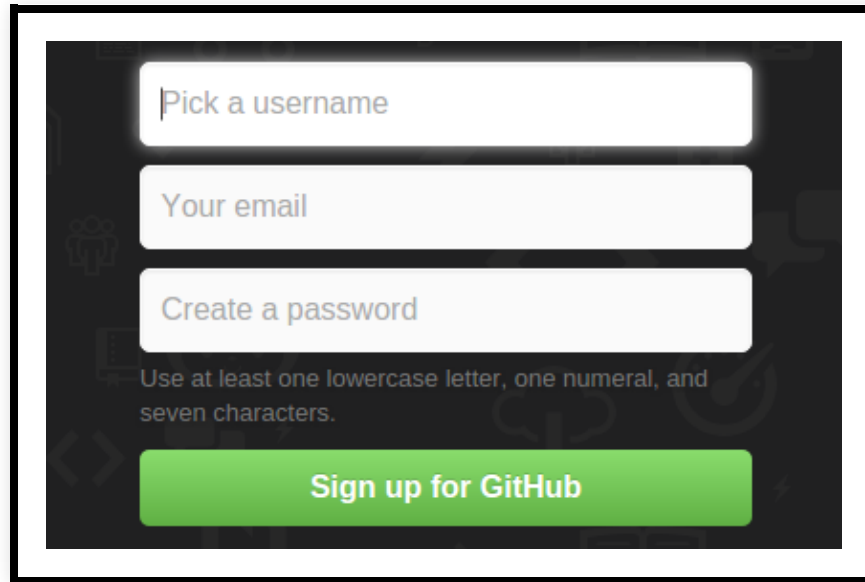
```
git checkout -b nombre-rama nombre-etiqueta
```

4 Uso básico de GitHub

4.1 Características de GitHub

- **Plataforma de desarrollo colaborativo**, que utiliza Git.
- Los **repositorios son públicos**, salvo con cuenta de pago.
- Tiene facetas de **red social** (perfil público, seguidores, estrellas, etc.)
- Nos permite **gestionar organizaciones y equipos**.
- **Gestión de proyectos** (wiki, releases, incidencias, gráficos, etc.)
- **Servidor web**.

4.2 Crear cuenta

A screenshot of the GitHub sign-up form. It features three white input fields on a dark background. The first field is labeled 'Pick a username', the second 'Your email', and the third 'Create a password'. Below the password field, there is a text requirement: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form is a prominent green button with the text 'Sign up for GitHub' in white.

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

Sign up for GitHub

Crear cuenta en GitHub

4.3 Crear repositorio

The screenshot shows the GitHub profile of Adolfo Sanz De Diego. The profile includes a profile picture, the name 'Adolfo Sanz De Diego', the username 'asanzdiego', and the location 'Madrid'. The page is divided into sections for 'Popular repositories' and 'Repositories contributed to'. A dropdown menu is open, showing options to 'New repository' and 'New organization'.

Popular repositories

Repository	Stars
markdownslides	47 ★
curso-interfaces-web-2014 Curso de Diseño de Interfaces Responsive We...	4 ★
casperjs-examples	3 ★
deck.js-pandoc-slides	3 ★
curso-javascript-avanzado-2015	3 ★

Repositories contributed to

Repository	Stars
HackathonLovers/HackathonLov...	0 ★
LogLock/LogLock.github.io .	0 ★
pelitweets/pelitweets-backend Backend of pelitweets project	0 ★
HackathonLovers/snaphack-nod...	0 ★
pronoide/pronoide.github.io pronoide web site	0 ★

Crear un repositorio

4.4 Configurar claves (I)

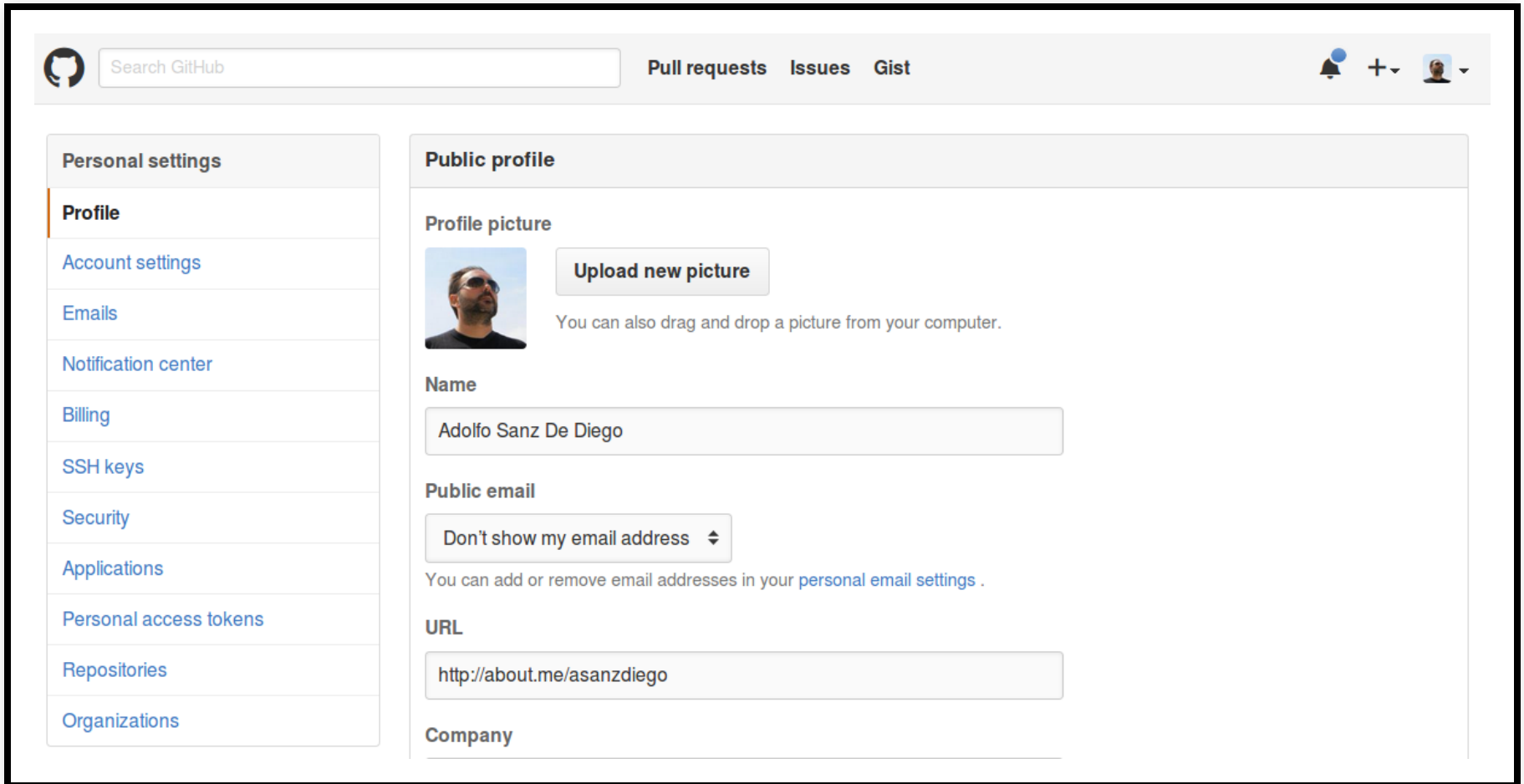
- Nos permite gestionar repositorios **mediante SSH** sin tener que estar poniendo siempre nuestra contraseña.
- Se genera una **clave privada** que se guarde en nuestro ordenador y una **clave pública** que es la que tenemos que guardar en nuestra cuenta.

4.5 Configurar claves (II)

- La podemos usar pues **solo con un ordenador**.
- Instrucciones:
 - <https://help.github.com/articles/generating-ssh-keys/>

4.6 Cambiar avatar

- View profile and more > Settings > Profile



The screenshot shows the GitHub 'Public profile' settings page. On the left is a sidebar with 'Personal settings' and a list of options: Profile (highlighted), Account settings, Emails, Notification center, Billing, SSH keys, Security, Applications, Personal access tokens, Repositories, and Organizations. The main content area is titled 'Public profile' and contains several sections: 'Profile picture' with a current profile picture of a man and an 'Upload new picture' button; 'Name' with a text input field containing 'Adolfo Sanz De Diego'; 'Public email' with a dropdown menu set to 'Don't show my email address'; 'URL' with a text input field containing 'http://about.me/asanzdiego'; and 'Company' with an empty text input field. The top of the page features the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', and 'Gist'.

Search GitHub


Pull requests Issues Gist

Personal settings

- Profile**
- Account settings
- Emails
- Notification center
- Billing
- SSH keys
- Security
- Applications
- Personal access tokens
- Repositories
- Organizations

Public profile

Profile picture

 **Upload new picture**

You can also drag and drop a picture from your computer.

Name

Adolfo Sanz De Diego

Public email

Don't show my email address

You can add or remove email addresses in your [personal email settings](#).

URL

http://about.me/asanzdiego

Company

Cambiar avatar en GitHub

4.7 Doble factor de autenticación

- View profile and more > Settings > Security

GitHub interface showing the **Security** settings page.

Personal settings

- Profile
- Account settings
- Emails
- Notification center
- Billing
- SSH keys
- Security**
- Applications
- Personal access tokens
- Repositories
- Organizations

Two-factor authentication

Status: **On** ✓ [Edit](#)

🔔 [Save your recovery codes](#) in a safe place. They will allow you to access your account if you ever lose your phone.

Sessions

This is a list of devices that have logged into your account. Revoke any sessions that you do not recognize.

- Madrid**
Your current session [...](#)
Firefox on Ubuntu
Location:
Madrid, Madrid, Spain
Signed in:
October 18, 2015

Activar el doble factor de autenticación en GitHub

4.8 Uso social

- Características sociales:
 - Seguir a gente.
 - Seguir proyectos (watch).
 - Premiar proyectos (start).
 - Forquear proyectos (fork).
 - Crear organizaciones.

5 Uso avanzado de Git

5.1 Conectar un repositorio remoto

- Podemos **conectar uno o varios repositorios remotos** a nuestro repositorio.

```
git remote add alias-repositorio-remoto url-repositorio-remoto
```

5.2 Renombrar un repositorio remoto

- Podemos **renombrar el alias de un repositorio remoto**.

```
git remote rename antiguo-alias nuevo-alias
```

5.3 Desconectar un repositorio remoto

- Podemos **desconectar un repositorio remoto**.

```
git remote remove alias-repositorio-remoto
```

5.4 Ver los repositorios remotos

- Podemos **ver los repositorios remotos conectados y los permisos que tenemos.**

```
git remote -v
```

5.5 Descargar cambios remotos

- Podemos **descargar los cambios remotos sin modificar nuestro repositorio local.**

```
git fetch alias-repositorio-remoto
```

5.6 Descargar y combinar

- Podemos **descargar y combinar los cambios remotos** con los de tu repositorio local.

```
git pull alias-repositorio-remoto nombre-rama-repositorio-remoto
```


5.7 Enviar datos (I)

- Podemos **enviar datos al repositorio remoto** (solo si está up-to-date).

```
git push alias-repositorio-remoto nombre-rama-repositorio-remoto
```

5.8 Enviar datos (II)

- Normalmente:

```
git push origin master
```

5.9 Enviar datos (III)

- Si queremos subir los tags:

```
git push --tag origin master
```

5.10 Clonar repositorios

- Clonar es como:
 - hacer un init
 - luego un remote add
 - luego un fetch con alias=origin
 - dejando las ramas remota y local en master

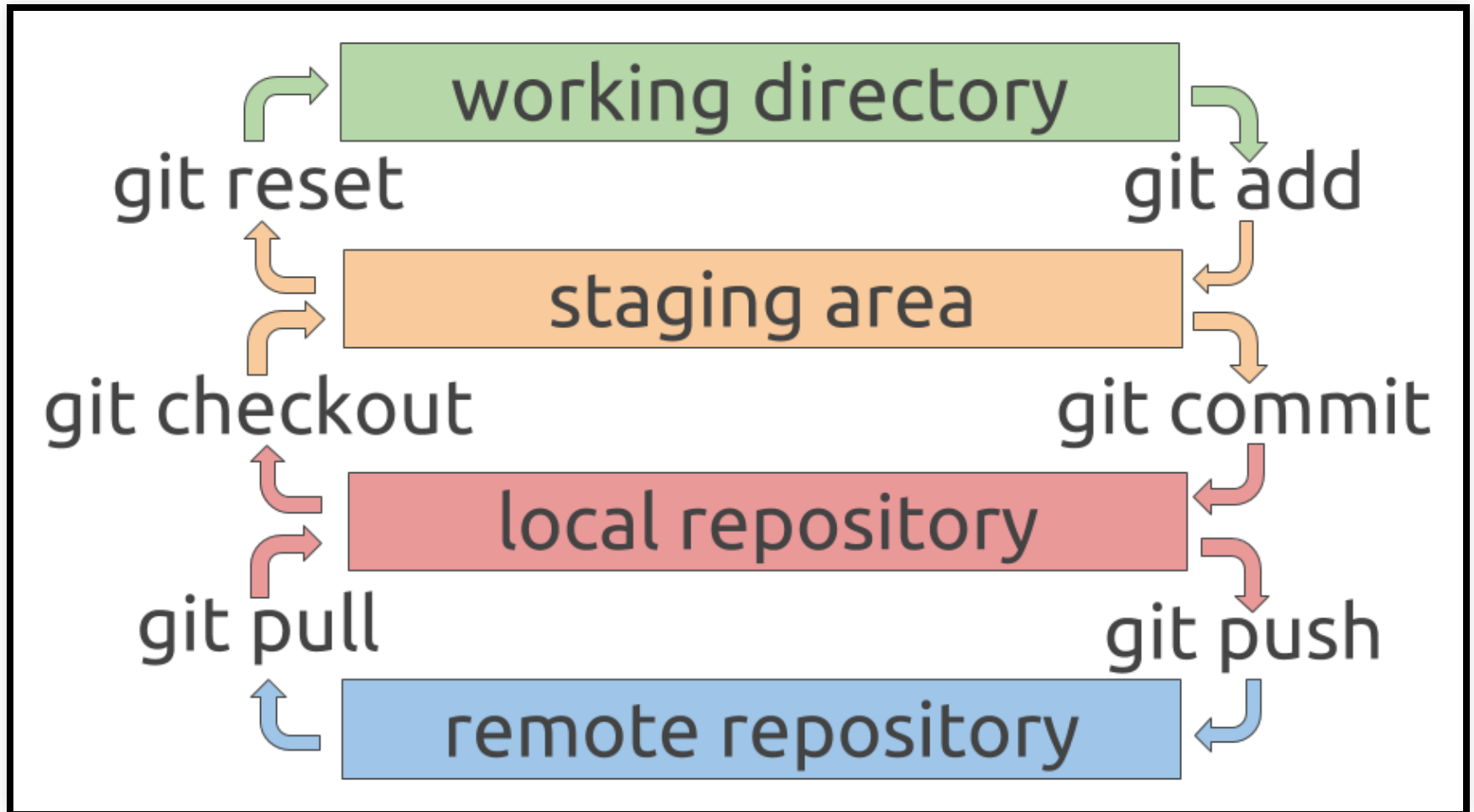
```
git clone url-repositorio-remoto
```

5.11 Inspeccionar repositorio remoto

- Podemos ver **información de un remoto particular, y como están configurados pull y push.**

```
git remote show alias-repositorio-remoto
```

5.12 Resumen áreas



Resumen áreas GIT

5.13 Crear una rama

- Podemos crear ramas que son **apuntadores que podemos mover por los distintos snapshots.**
- Solo la creamos, no nos situamos en ella.

```
git branch nombre-rama
```

5.14 Cambiar de rama

- El HEAD es el apuntador que usa GIT para saber en que rama estás.
- Cuando cambiamos de rama GIT **cambia el HEAD y los ficheros de tu área de trabajo.**

```
git checkout nombre-rama
```


5.15 Crear y cambiar de rama

- Podemos **crear y cambiar de rama** con un mismo comando.

```
git checkout -b nombre-rama
```

5.16 Ver las ramas y el HEAD

- Podemos **ver las ramas y donde apunta el HEAD**.

```
git log --oneline --decorate --graph --all
```

```
git branch -v
```

5.17 Fusionar ramas

- GIT es **muy potente** con la fusión de ramas.

```
git merge nombre-rama
```

5.18 Solucionar conflictos

- Si al hacer un merge existan conflictos **GIT los apunta en los propios ficheros.**

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">please contact us at support@github.com</div>
>>>>>> issue:index.html
```

5.19 Borrar ramas

- Una vez fusionado la rama en el master, **conviene borrarla** (solo nos deja si está fusionada).

```
git branch -d nombre-rama
```

5.20 Listado de ramas por estado

- Podemos saber **que ramas están fusionada y cuales no.**

```
git branch --merged
```

```
git branch --no-merged
```

5.21 Sincronizar rama remota

- Igual que sincronizamos la rama master remota, podemos **sincronizar otras ramas remotas**.

```
git checkout -b nombre-rama-local alias-repositorio-remoto/nombre-rama-r
```

```
git checkout --track alias-repositorio-remoto/nombre-rama-remota
```

5.22 Asignar rama remota

- Podemos **asignar el área de trabajo a una rama remota.**

```
git checkout -u alias-repositorio-remoto/nombre-rama-remota
```


5.23 Listado de todas las ramas

- Podemos listar no solo las ramas locales, sino **también las remotas.**

```
git branch -vv
```

5.24 Eliminar rama remota

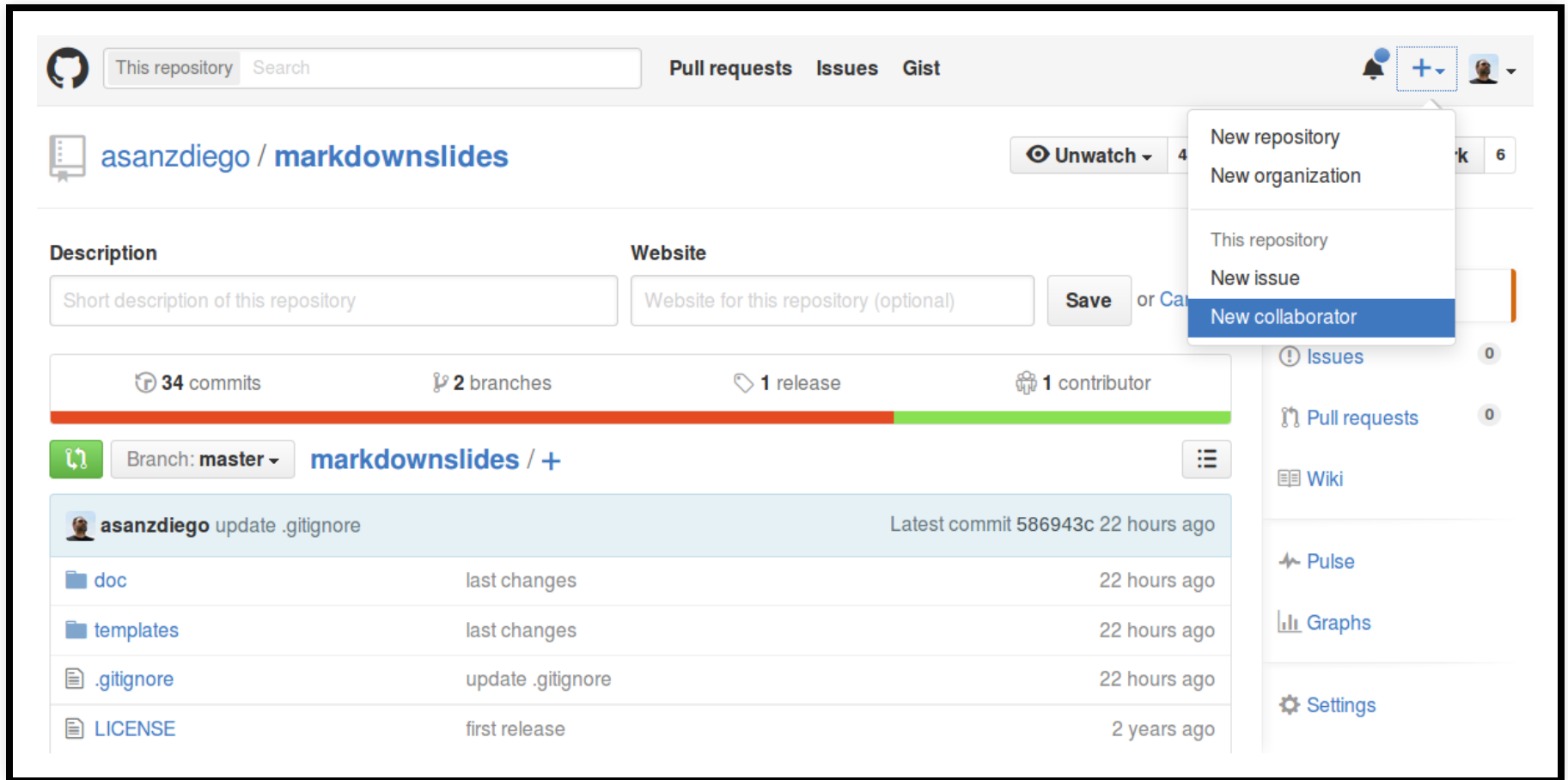
- Podemos **eliminar las ramas remotas**.

```
git push alias-repositorio-remoto --delete nombre-rama-remota
```

6 Uso avanzado de GitHub

6.1 Añadir colaboradores

- Podemos **dar permisos de push** a quien queramos.



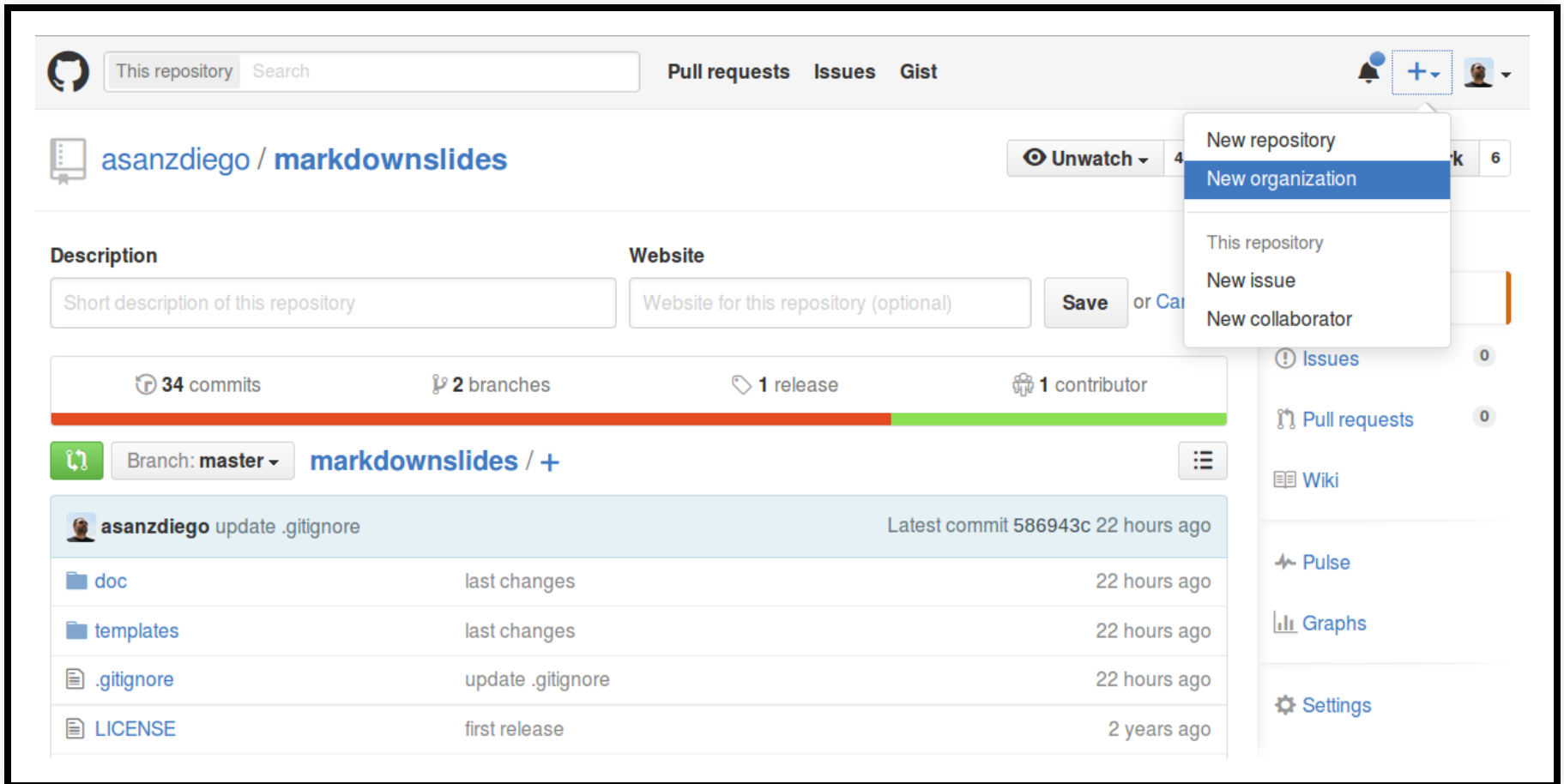
The screenshot shows the GitHub interface for the repository 'asanzdiego / markdownslides'. The top navigation bar includes 'Pull requests', 'Issues', and 'Gist'. A dropdown menu is open, showing options: 'New repository', 'New organization', 'This repository', 'New issue', and 'New collaborator' (highlighted in blue). Below the repository name, there are fields for 'Description' and 'Website', and a 'Save' button. The repository statistics show 34 commits, 2 branches, 1 release, and 1 contributor. The file list includes 'doc', 'templates', '.gitignore', and 'LICENSE'. The right sidebar contains links to 'Issues', 'Pull requests', 'Wiki', 'Pulse', 'Graphs', and 'Settings'.

File	Changes	Time
doc	last changes	22 hours ago
templates	last changes	22 hours ago
.gitignore	update .gitignore	22 hours ago
LICENSE	first release	2 years ago

GitHub New Collaborator

6.2 Crear organizaciones

- Podemos **crear organizaciones**.



The screenshot shows the GitHub interface for the repository 'asanzdiego / markdownslides'. The top navigation bar includes 'Pull requests', 'Issues', and 'Gist'. A dropdown menu is open, showing options: 'New repository', 'New organization' (highlighted), 'This repository', 'New issue', and 'New collaborator'. Below the repository name, there are input fields for 'Description' and 'Website', and a 'Save' button. The repository statistics show 34 commits, 2 branches, 1 release, and 1 contributor. The file list includes 'doc', 'templates', '.gitignore', and 'LICENSE'.

File	Changes	Time
doc	last changes	22 hours ago
templates	last changes	22 hours ago
.gitignore	update .gitignore	22 hours ago
LICENSE	first release	2 years ago

GitHub New Organization

6.3 Gestionar organizaciones

- Dentro de las organizaciones podemos **crear equipos** y/o trabajar con colaboradores externos.
- El **nivel de permisos se gestiona a nivel de equipo**.
- Las personas tendrán los permisos de los equipos a los que pertenezca.
- Los permisos se otorgan a cada repositorio.

6.4 Forkear proyectos

- Para **participar en un proyecto sin permisos de escritura**, puedes forkearlo.
- Consiste en crear una copia completa del repositorio bajo tu control: se encontrará **en tu cuenta** y podrás escribir en él sin limitaciones.

6.5 Pull-requests (I)

- Para **enviar propuestas de mejora**.
- Se usa mucho para proyectos que no son tuyos y en donde te gustaría colaborar.
- También se usa dentro de equipos para gestionar proyectos grandes.

6.6 Pull-requests (II)

1. Crear un fork de proyecto.
2. Clonar nuestro fork en nuestro equipo.
3. Crear una rama que sea descriptiva.
4. Realizar nuestros cambios.
5. Comprobar los cambios.
6. Enviar nuestra nueva rama de vuelta a nuestro fork.

6.7 Pull-requests (III)

1. Abrir un Pull Request en GitHub.
2. Participa en la discusión asociada.
3. Opcionalmente, se realizan nuevos commits.
4. El propietario del proyecto original cierra el Pull Request
 - bien fusionando la rama con tus cambios
 - o bien rechazándolos.

6.8 Issues y Wikis

- Todos los repositorios de GitHub tienen asociados:
 - un gestor de incidencias (issues)
 - una wiki para documentar

6.9 GitHub pages (I)

- Podemos tener **servidor web en los repositorios simplemente nombrandolos así:**

```
usuario.github.io
```

```
organizacion.github.io
```

6.10 GitHub pages (II)

- También podemos hacer lo mismo con un determinado proyecto **creando una rama gh-pages**.
- Ver : <https://pages.github.com/>

6.11 Fichero README.md

- Nos **lo muestra renderizado** en la página del repositorio.

6.12 Webhooks & services

- Para que GitHub pueda **interactuar con sistemas externos**.
- Los servicios están ya medio configurados.
- Si necesitas algo más específico lo tienes que hacer con webhooks, que lo que hace GitHub es hacer un POST a la URL que indiques cuando se lance algún evento (push, pull request, fork, etc.)

7 Markdown

7.1 ¿Qué es Markdown?

"Es un lenguaje de marcado ligero que trata de conseguir la máxima legibilidad y 'publicabilidad' usando texto plano."

- <https://es.wikipedia.org/wiki/Markdown>

7.2 Características principales

- Texto plano
- Sintaxis sencilla
- Legibilidad
- Publicabilidad
- Exportabiliad

7.3 Markdownslides

- <https://github.com/asanzdiego/markdownslides>

7.4 Chuleta de Markdown:

- <http://warpedvisions.org/projects/markdown-cheat-sheet>

7.5 Editor online

- <https://jbt.github.io/markdown-editor/>

7.6 Encabezados (I)

- <h1>, <h2>, <h3>

```
# Encabezado de primer nivel
```

```
## Encabezado de segundo nivel
```

```
### Encabezado de tercer nivel
```

7.7 Encabezados (II)

- Equivalente a lo anterior.

```
Encabezado de primer nivel
```

```
=====
```

```
Encabezado de segundo nivel
```

```
-----
```

```
### Encabezado de tercer nivel ###
```

7.8 Listas no numeradas

- No enumeradas:
 - se puede usar el menos
 - se puede usar el asterísico
 - se puede usar el más

```
- se puede usar el menos  
* se puede usar el asterísico  
+ se puede usar el más
```


7.9 Listas numeradas

- Enumeradas:
 1. Primer elemento
 2. Segundo elemento
 3. Tercer elemento

```
1. Primer elemento  
1. Segundo elemento  
1. Tercer elemento
```

7.10 Formato (negrita, cursiva, tachado)

- Texto en cursiva con *un asterisco* o con *un guión bajo*.
- Texto en negrita con **dos asteriscos** o con **dos guiones bajos**.
- Texto tachado con ~~dos virgulillas~~.

```
- Texto negrita con **dos asteriscos** o con __dos guiones bajos__.  
- Texto cursiva con *un asterisco* o con _un guión bajo_.  
- Texto tachado con ~~dos virgulillas~~.
```

7.11 tablas

Header	Header	Right
Cell	Cell	\$10
Cell	Cell	\$20

```
| Header | Header | Right |
| -----|-----|-----:|
| Cell   | Cell   | $10    |
| Cell   | Cell   | $20    |
```

7.12 Citas

"No hay camino hacia el Software Libre, el Software Libre es el camino"

```
> "No hay camino hacia el Software Libre,  
el Software Libre es el camino"
```

7.13 Código

```
require(maps) # activación de librería
require(mapproj) # se usará para projection="polyconic"
  # Cargar los datos
  # unemp incluye datos para condados de los Estados Unidos continentales
data(unemp) # Datos de desempleo
data(county.fips) # mapa de los condados
```

```
require(maps) # activación de librería
require(mapproj) # se usará para projection="polyconic"
  # Cargar los datos
  # unemp incluye datos para condados de los Estados Unidos continentales
data(unemp) # Datos de desempleo
data(county.fips) # mapa de los condados
```

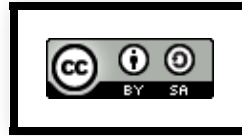
7.14 Enlaces

- Enlace con texto
- Enlace sencillo:
- <https://github.com/asanzdiego/curso-git-github-markdown-2015>

```
- [Enlace con texto](https://github.com/asanzdiego/curso-git-github-markdown-2015)
- Enlace sencillo:
  -<https://github.com/asanzdiego/curso-git-github-markdown-2015>
```

7.15 Imágenes

- Este obra está bajo una licencia:



Creative Commons BY SA

- Este obra está bajo una licencia:

```
![Creative Commons BY SA](../img/cc-by-sa.png)
```