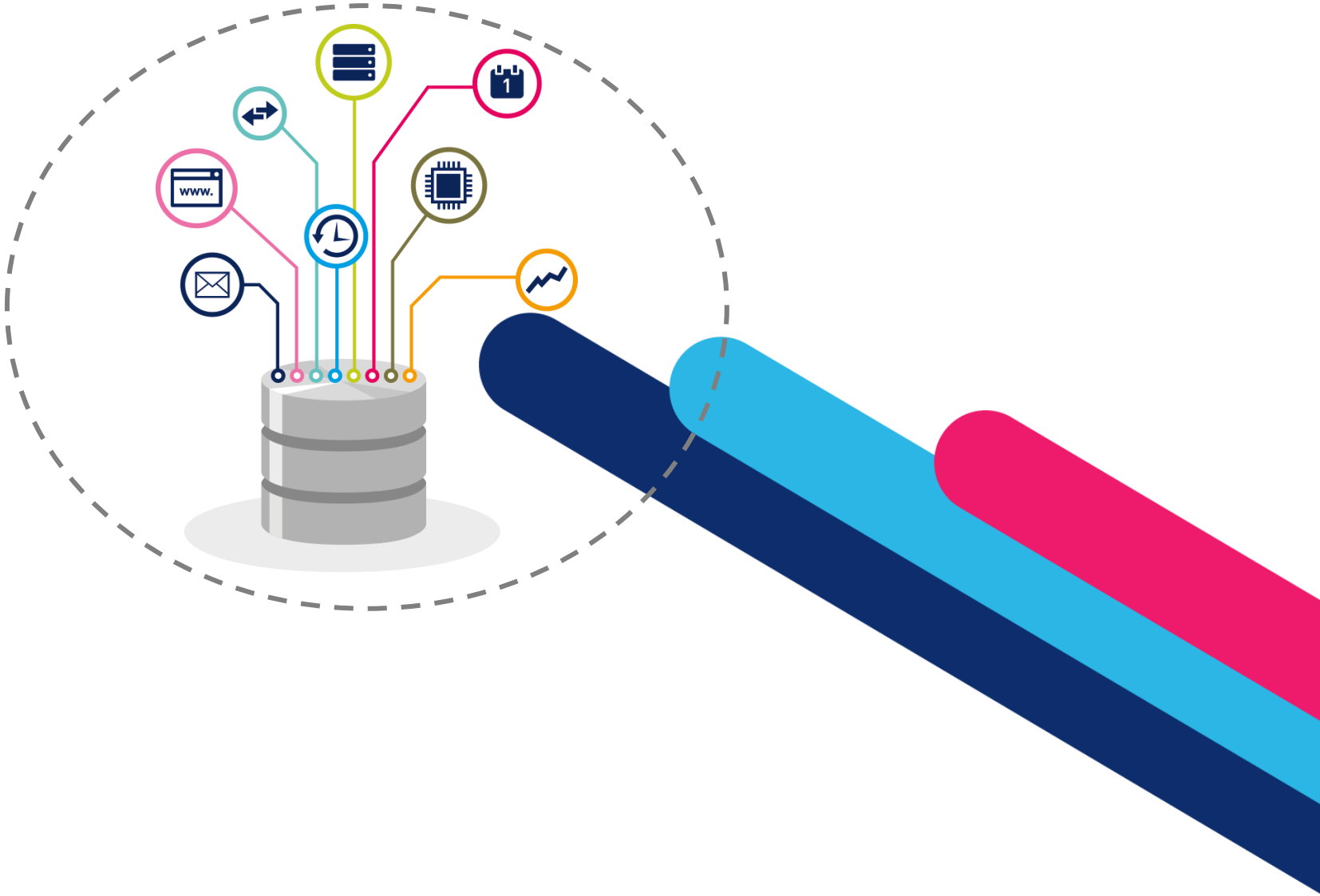
CERTUS



Curso: DataOps

Preguntas

- ¿Qué se entiende por control de versiones?

Contenidos que veremos el día de hoy

DataOps

1. Conceptos de DevOps y DataOps. ✓
2. CI/CD Jenkins. ✓
3. Control de Versiones con Git y GitHub. ✓
4. Microservicios e Infraestructura como código.

CERTUS

Nuestro objetivo

Al finalizar esta sesión, conocerá la práctica de control de versiones con las herramientas de git y github.

DataOps: Control de Versiones

- Git y GitHub

Prácticas DevOps / DataOps

1. **Integración y entrega continuas (CI/CD):**
   Uso de herramientas para administrar el número de cambios en el proceso de desarrollo, y así tener un mayor control evitando así errores que podrían modificar el sortware.

2. **Control de versiones:**
   Es la administración de las versiones de un software y el seguimiento de cada una de las revisiones en caso de recuperar un código.

3. **Microservicios:**
   Este es el enfoque de creación de las apps, el cual se logra a través de "microservicios", que después sin conectados por una intefaz.

4. **Infraestructura como código:**
   Permite interactuar con la infraestructura como si se tratara de un código. Esto permite hacerlo a escala sin la necesidad de realizar los ajustes de forma manual.

5. **Administración de configuración:**
   Sirve para administrar los recursos de los sistemas, además de seguir el estado de estos. Evita cualquier alteración en la configuración.

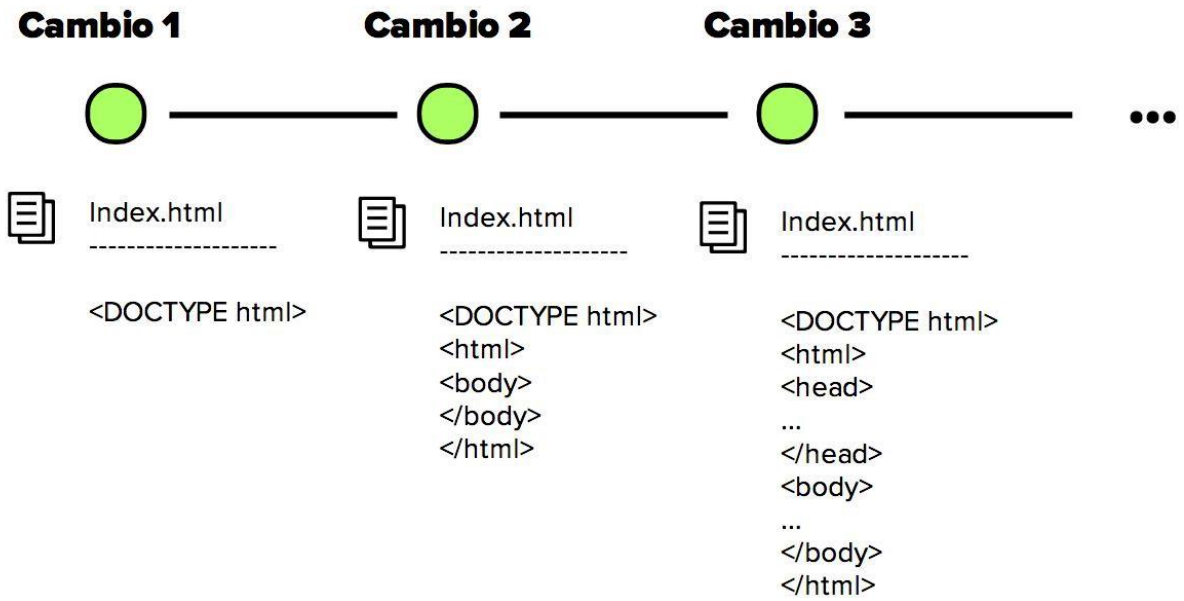
6. **Supervisión continua:**
   Es la identificación y recopilación de información sobre los problemas que puedan surgir en alguna versión específica del software.

CERTUS

Control de Versiones

Control de Versiones

En realidad, los cambios y diferencias entre las versiones de nuestros proyectos pueden tener similitudes, algunas veces los cambios pueden ser solo una palabra o una parte específica de un archivo o segmento de código específico.

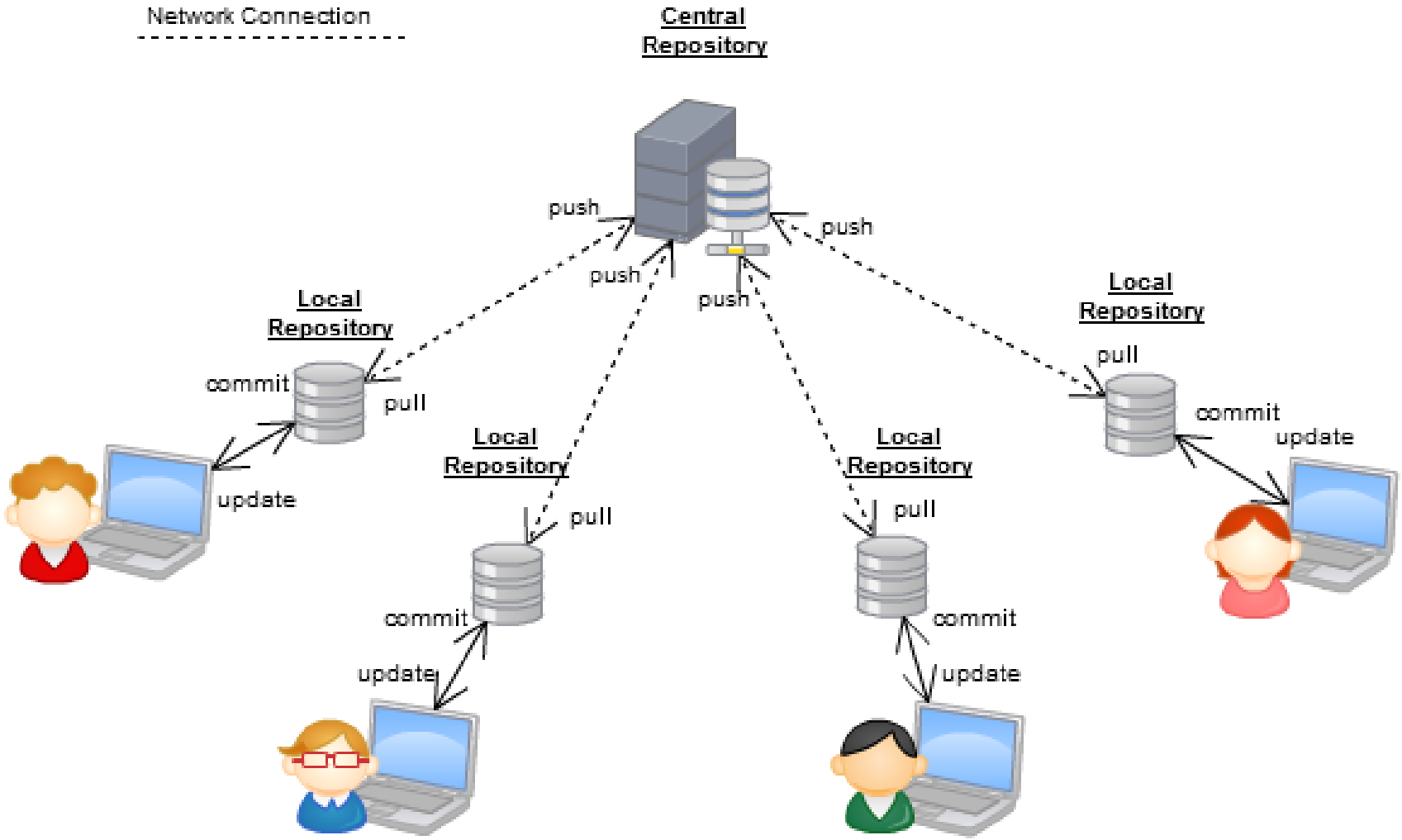


Ventajas del Control de Versiones

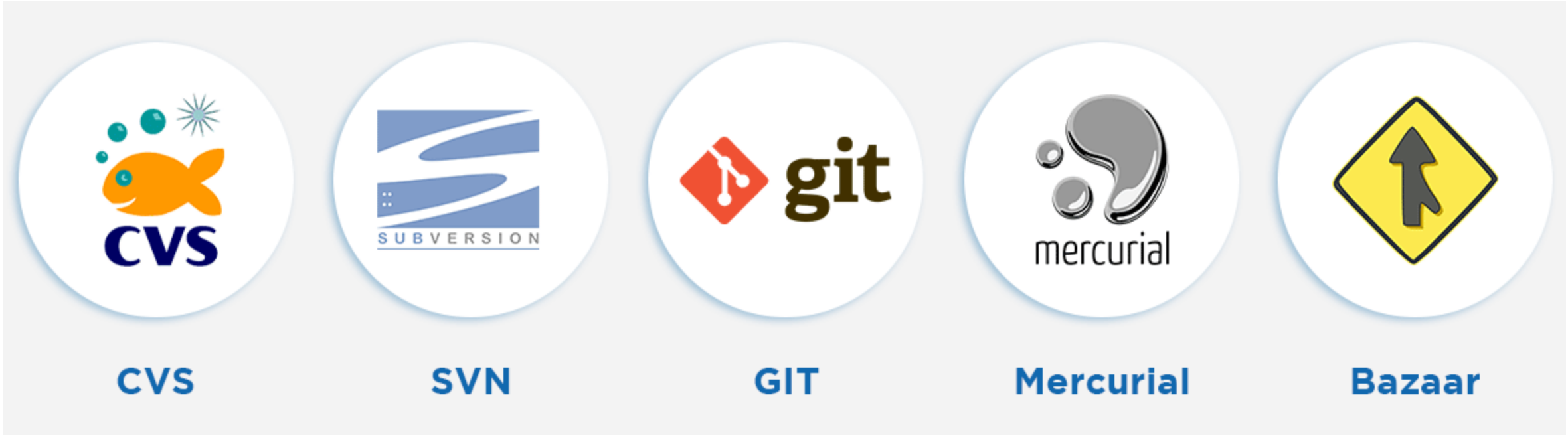
- Rastrear el desarrollo y los cambios de tus documentos.
- Registrar los cambios que has hecho de una manera que puedas entender posteriormente.
- Experimentar con versiones distintas de un documento al mismo tiempo que conservas la más antigua.
- Fusionar dos versiones de un documento y administrar los conflictos existentes entre distintas versiones.
- Revertir cambios y volver atrás gracias al historial de versiones anteriores de tu documento.

CERTUS

Sistema de Control de Versiones



Herramientas para el Control de Versiones



GIT para el Control de Versiones

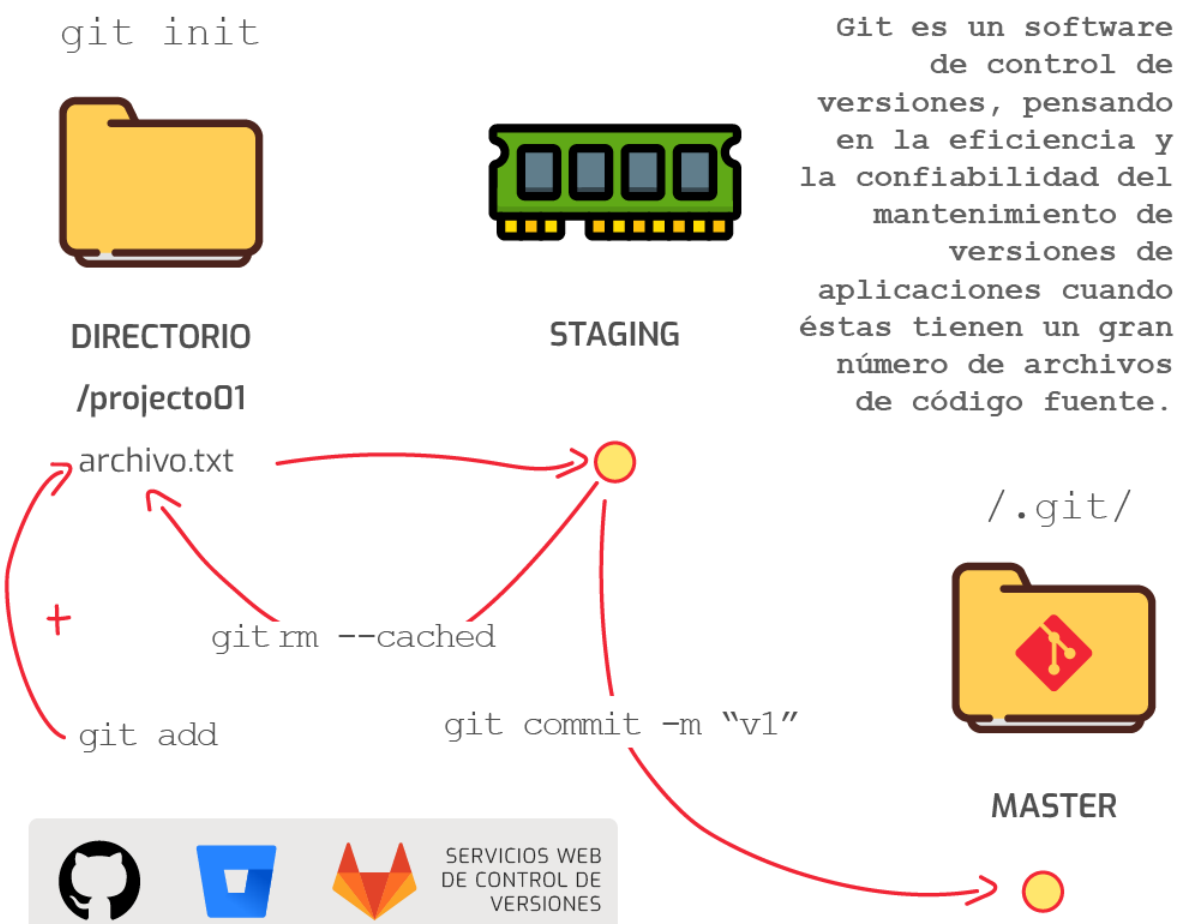
Git es un sistema de control de versiones distribuido, diseñado por Linus Torvalds. Está pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

- Git está optimizado para guardar cambios de forma incremental.
- Permite contar con un historial, regresar a una versión anterior y agregar funcionalidades.
- Lleva un registro de los cambios que otras personas realicen en los archivos.

CERTUS

Ciclo de trabajo en GIT

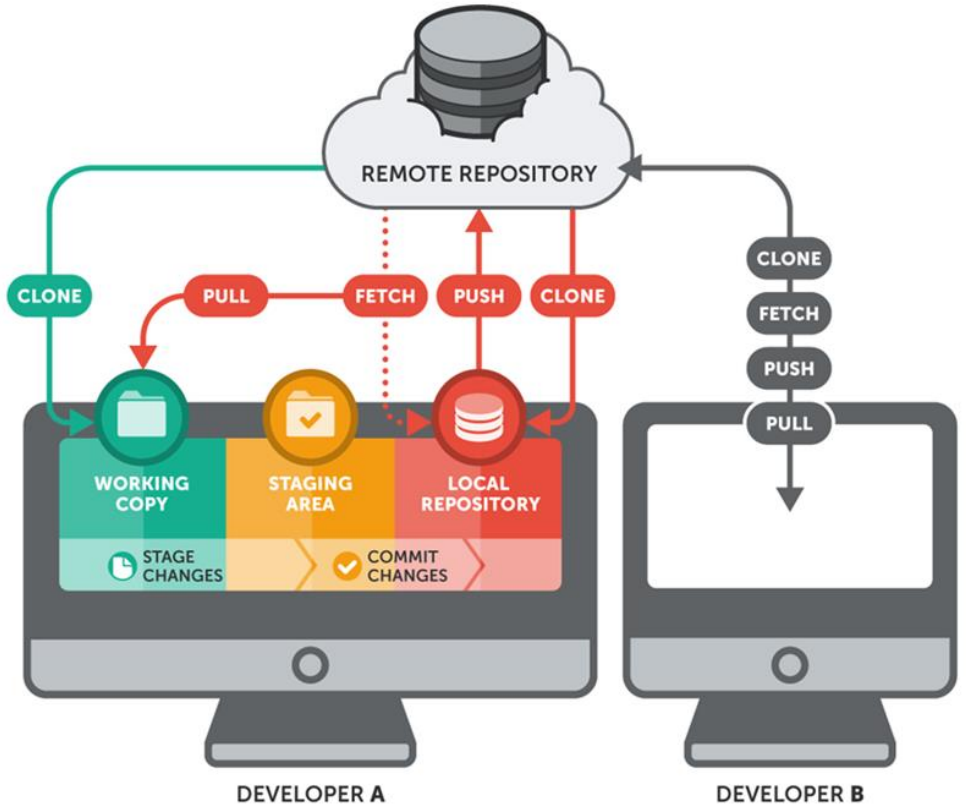
CICLO BÁSICO DE TRABAJO EN GIT



CERTUS

Conceptos GIT

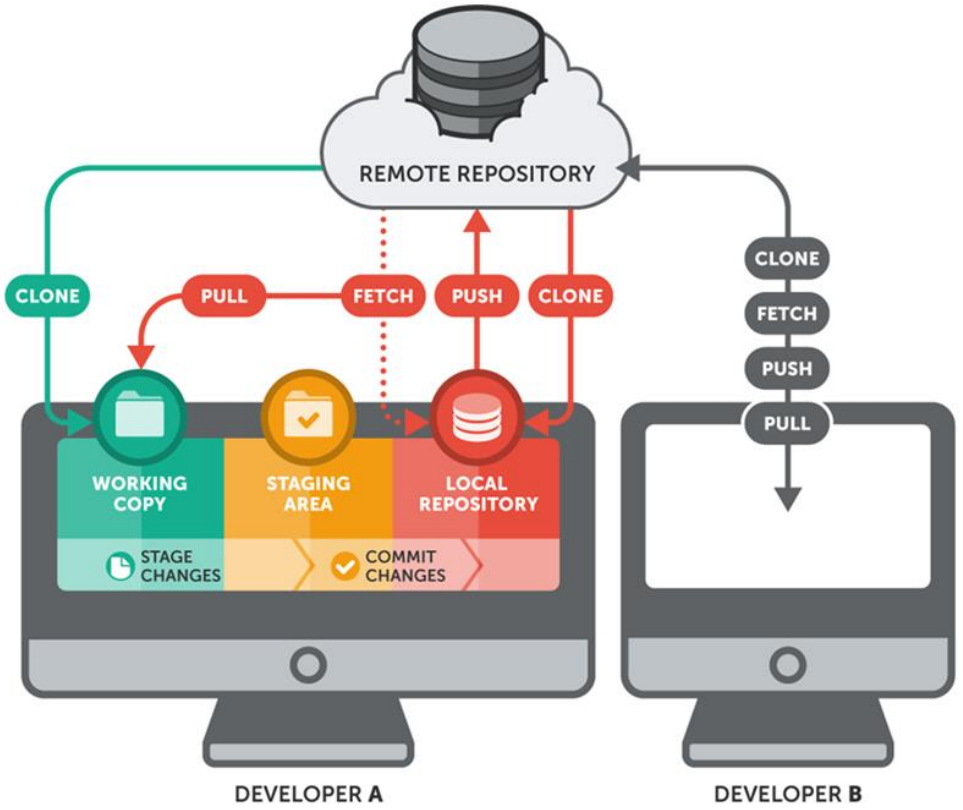
- **Bug:** Error en el código
- **Repository:** Donde se almacena todo el proyecto, el cual puede vivir tanto en local como en remoto. El repositorio guarda un historial de versiones y, más importante, de la relación de cada versión con la anterior para que pueda hacerse el árbol de versiones con las diferentes ramas.
- **Fork:** Si en algún momento queremos contribuir al proyecto de otra persona, o si queremos utilizar el proyecto de otro como el punto de partida del nuestro. Esto se conoce como “fork”.
- **Clone:** Una vez se decide hacer un fork , hasta ese momento sólo existe en GitHub. Para poder trabajar en el proyecto, toca clonar el repositorio elegido al computador personal.
- **Branch:** Es una bifurcación del proyecto que se está realizando para anexar una nueva funcionalidad o corregir un bug.



CERTUS

Conceptos GIT

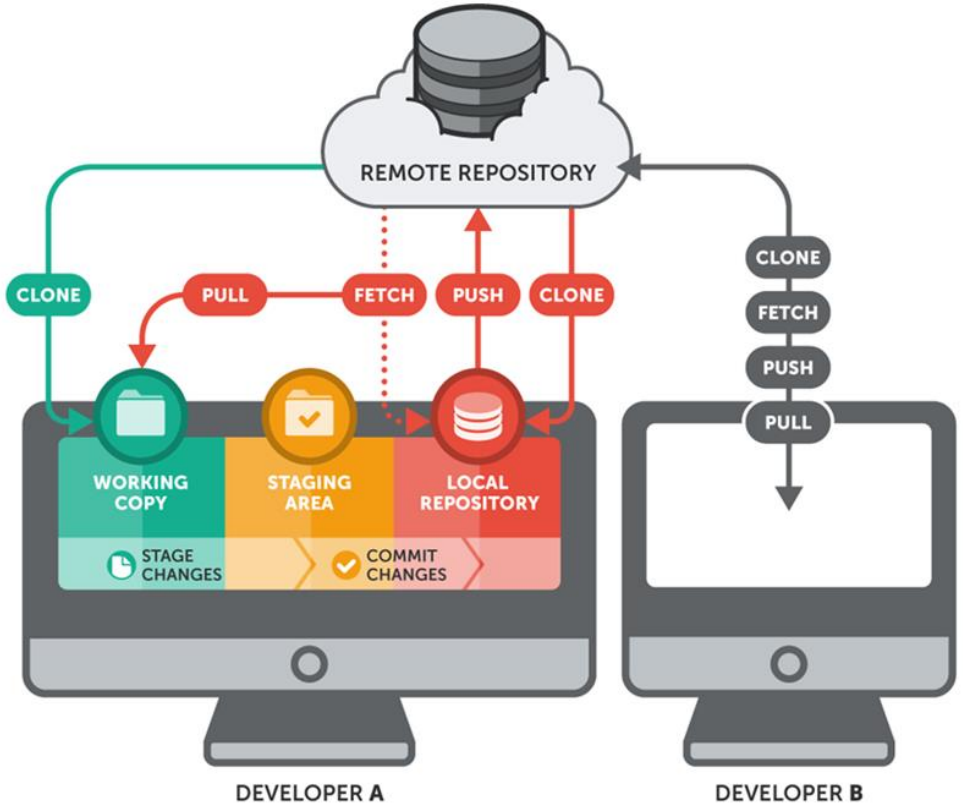
- **Master:** Rama donde se almacena la última versión estable del proyecto que se está realizando. La rama master es la que está en producción en cada momento (o casi) y debería estar libre de bugs. Así, si esta rama está en producción, sirve como referente para hacer nuevas funcionalidades y/o arreglar bugs de última hora.
- **Commit:** consiste en subir cosas a la versión local del repositorio. De esta manera se puede trabajar en la rama de forma local sin tener que modificar ninguna versión en remoto ni tener que tener la última versión remota, cosa muy útil en grandes desarrollos trabajados por varias personas.
- **Push:** Consiste en enviar todo lo que se ha confirmado con un commit al repositorio remoto. Aquí es donde se une nuestro trabajo con el de los demás.
- **Checkout:** Acción de descargarse una rama del repositorio GIT local (sí, GIT tiene su propio repositorio en local para poder ir haciendo commits) o remoto.



CERTUS

Conceptos GIT

- **Fetch:** Actualiza el repositorio local bajando datos del repositorio remoto al repositorio local sin actualizarlo, es decir, se guarda una copia del repositorio remoto en el local.
- **Merge:** La acción de merge es la continuación natural del fetch. El merge permite unir la copia del repositorio remoto con tu repositorio local, mezclando los diferentes códigos.
- **Pull:** Consiste en la unión del fetch y del merge, esto es, recoge la información del repositorio remoto y luego mezcla el trabajo en local con esta.
- **Diff:** Se utiliza para mostrar los cambios entre dos versiones del mismo archivo.



CERTUS

GitHub

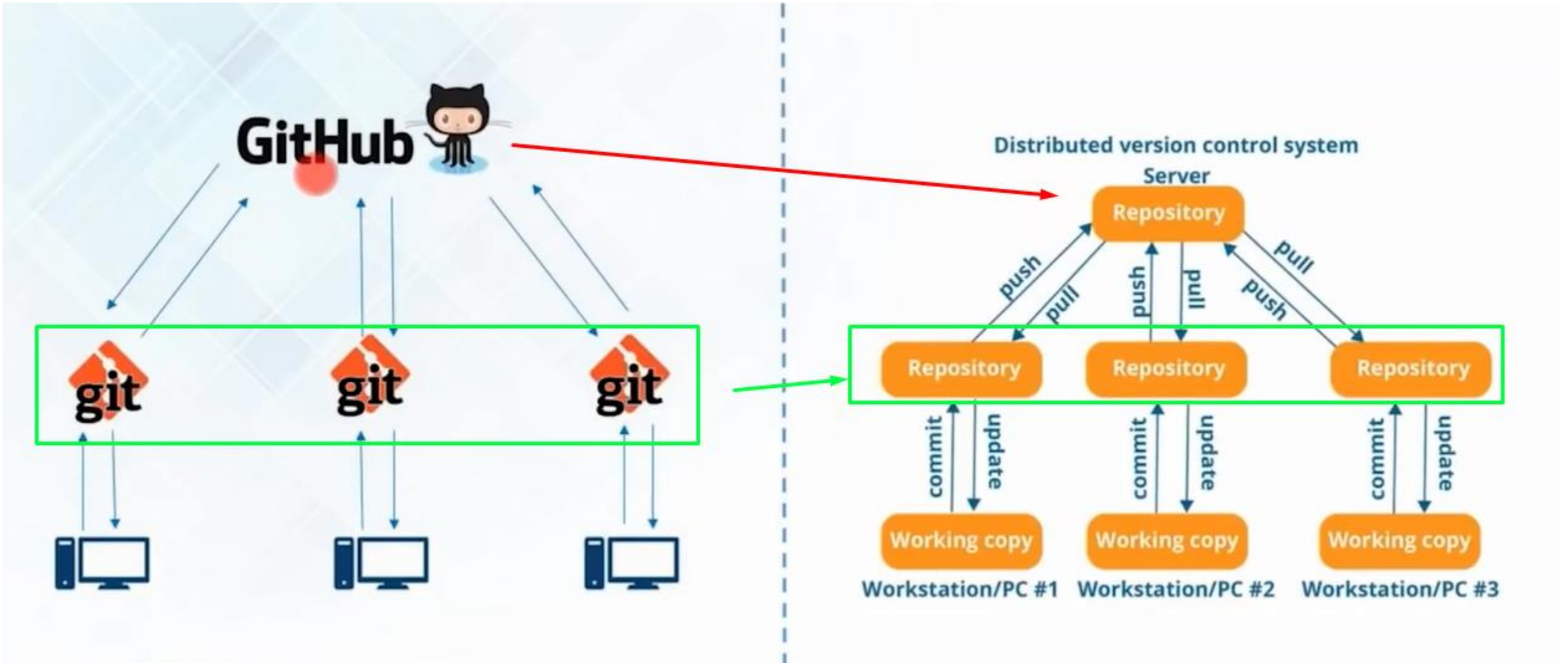
Github es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se emplea principalmente para la creación de código fuente de programas de computadora.

GitHub

- GitHub permite alojar proyectos en repositorios de forma gratuita y pública, pero tiene una forma de pago para privados.
- Puedes compartir fácilmente tus proyectos.
- Permite colaborar para mejorar los proyectos de otros y a otros mejorar o aportar a los tuyos.
- Ayuda a reducir significativamente los errores humanos, a tener un mejor mantenimiento de distintos entornos y a detectar fallos de una forma más rápida y eficiente.
- Es la opción perfecta para poder trabajar en equipo en un mismo proyecto.
- Ofrece todas las ventajas del sistema de control de versiones Git, pero también tiene otras herramientas que ayudan a tener un mejor control de los proyectos.

CERTUS

Git y GitHub



CERTUS

Comandos GIT

GIT INIT

Inicia un nuevo repositorio, esto creará el '**staging**' o área de ensayo y un repositorio local

GIT ADD

Añade un archivo al área de ensayo. Ejemplo: "**git add archivo**" o "**git add .**" para añadir todos los archivos.

GIT STATUS

Muestra el estado de nuestros archivos, es decir; los archivos que se han modificado.

GIT CLONE

Obtiene una copia de un proyecto que se encuentra en un repositorio público: "**git clone [url]**"

GIT COMMIT

Guarda los archivos en el repositorio local, Ejemplo: "**git commit -m 'Comentario sobre la actualización'**"

GIT CONFIG

Establece parámetros que GIT utiliza por defecto, ejemplo el autor; "**git config --global user.name "Tecsify**"

GIT RESET

Quita archivos añadidos al área de ensayo, borrará todos los cambios hechos despues del commit.

GIT DIFF

Muestra la diferencia entre una versión y otra, Ejemplo "**git diff commit1 commit2**"

CERTUS

Comandos GIT

Git Cheat Sheet

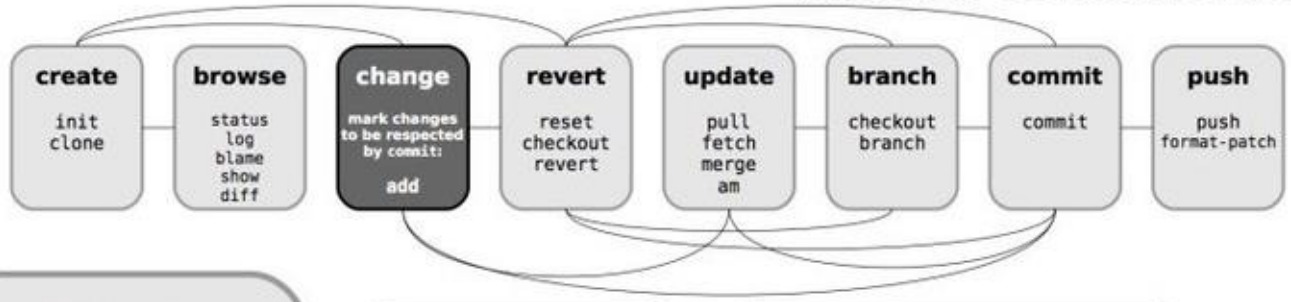
by Jan Krüger <jk@jk.gs>, http://jan-krueger.net/git/
Based on work by Zack Rusin

Basics

Use git help [*command*] if you're stuck.

`master`	default devel branch
`origin`	default upstream branch
`HEAD`	current branch
`HEAD^`	parent of HEAD
`HEAD~4`	great-great grandparent of HEAD
`foo..bar`	from branch *foo* to branch *bar*

(left to right) **Command Flow**



Create

From existing files
```
git init
git add .
```
From existing repository
```
git clone ~/old ~/new
git clone git://...
git clone ssh://...
```

Publish

In Git, `commit` only respects changes that have been marked explicitly with add.

```
git commit [-a]
```
(-a: add changed files automatically)
```
git format-patch origin
```
(create set of diffs)
```
git push remote
```
(push to origin or *remote*)
```
git tag foo
```
(mark current version)

Useful Tools

```
git archive
```
Create release tarball
```
git bisect
```
Binary search for defects
```
git cherry-pick
```
Take single commit from elsewhere
```
git fsck
```
Check tree
```
git gc
```
Compress metadata (performance)
```
git rebase
```
Forward-port local changes to remote branch
```
git remote add URL
```
Register a new remote repository for this tree
```
git stash
```
Temporarily set aside changes
```
git tag
```
(there's more to it)
```
gitk
```
Tk GUI for Git

Tracking Files

```
git add files
git mv old new
git rm files
git rm --cached files
```
(stop tracking but keep files in working dir)

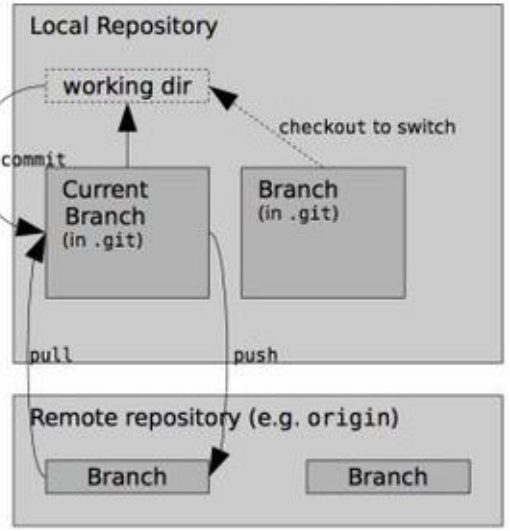
View

```
git status
git diff [oldid newid]
git log [-p] [file|dir]
git blame file
git show id (meta data + diff)
git show id:file
git branch (shows list, * = current)
git tag -l (shows list)
```

Update

```
git fetch (from def. upstream)
git fetch remote
git pull (= fetch & merge)
git am -3 patch.mbox
git apply patch.diff
```

Structure Overview



Revert

In Git, `revert` usually describes a new commit that undoes previous commits.

```
git reset --hard (NO UNDO)
```
(**reset** to last commit)
```
git revert branch
git commit -a --amend
```
(**replaces** prev. commit)
```
git checkout id file
```

Branch

```
git checkout branch
```
(switch working dir to *branch*)
```
git merge branch
```
(merge into current)
```
git branch branch
```
(branch current)
```
git checkout -b new other
```
(branch new from other and switch to it)

Conflicts

Use add to mark files as resolved.

```
git diff [--base]
git diff --ours
git diff --theirs
git log --merge
gitk --merge
```

CERTUS

Repasando lo aprendido

Git y GitHub

Git es un software de control de versiones diseñado por **Linus Torvalds**, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

Sistema operativo: Unix-like, Windows, Linux
Programado en: C, Bourne Shell, Perl
Modelo de desarrollo: Software libre
Escrito en: C, Perl, Tcl, Python

Importante

Directorios en Git
Es el lugar donde se almacenan los metadatos y las bases de datos para nuestros proyectos, y es justamente lo que se copia cuando clonamos de un ordenador a otro los archivos.

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en **Ruby on Rails**. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc.

☑ GIT

Git es un sistema de control de versiones que originalmente fue diseñado para operar en un entorno Linux. Actualmente Git es multiplataforma, es decir, es compatible con Linux, MacOS y Windows.

Características de Git

- Git almacena la información como un conjunto de archivos.
- No existen cambios, corrupción en archivos o cualquier alteración sin que Git lo sepa.
- Casi todo en Git es local. Es difícil que se necesiten recursos o información externos, basta con los recursos locales con los que cuenta.
- Git cuenta con 3 estados en los que podemos localizar nuestros archivos: Staged, Modified y Committed

☑ GITHUB

GitHub es un servicio de alojamiento que ofrece a los desarrolladores repositorios de software usando el sistema de control de versiones, Git.

Características de Github

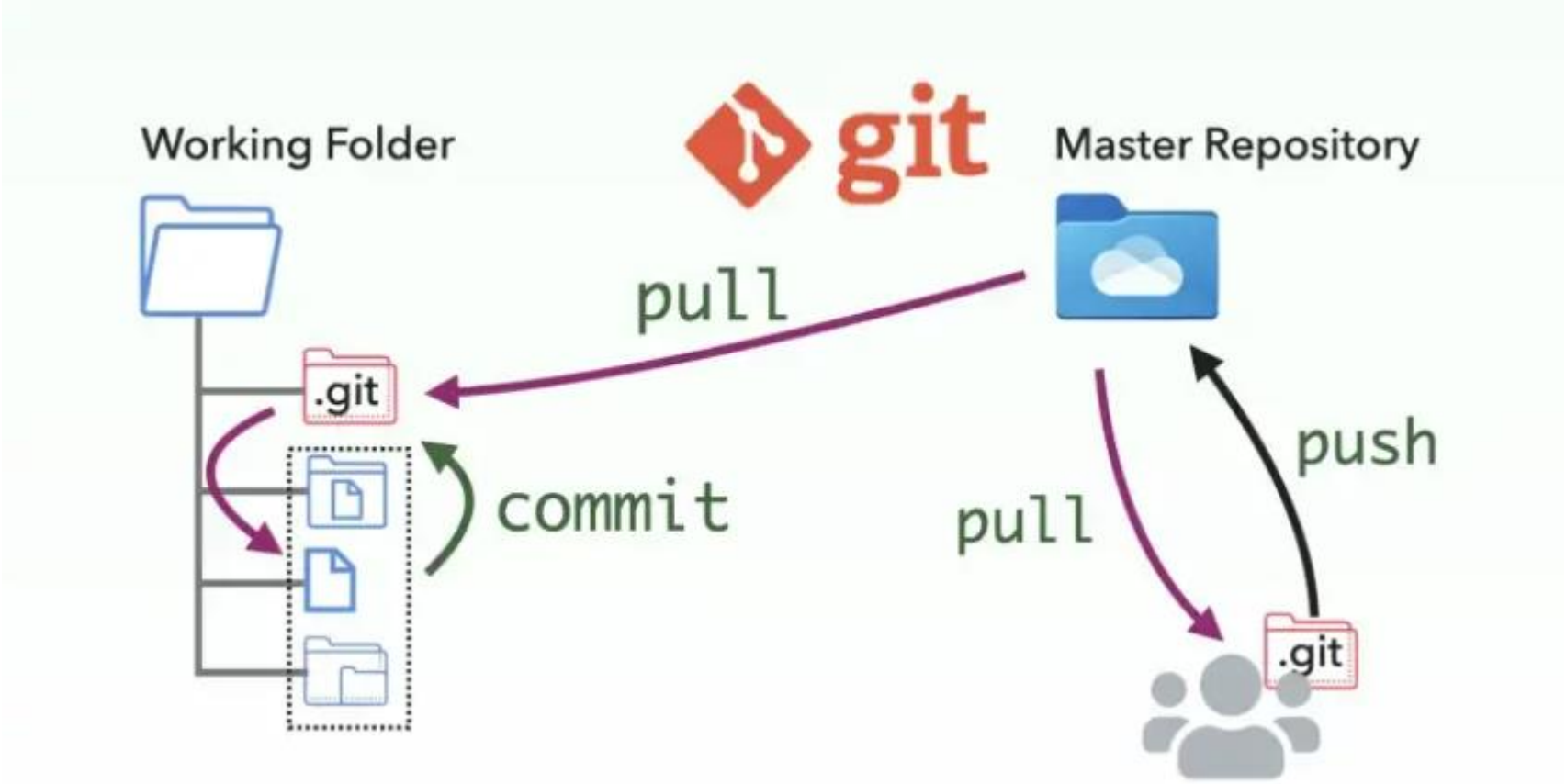
- GitHub permite que alojemos proyectos en repositorios de forma gratuita y publica, pero tiene una forma de pago para privados.
- Puedes compartir tus proyectos de una forma mucho más fácil.
- Te permite colaborar para mejorar los proyectos de otros y a otros mejorar o aportar a los tuyos.
- Ayuda reducir significativamente los errores humanos, a tener un mejor mantenimiento de distintos entornos y a detectar fallos de una forma más rápida y eficiente.
- Es la opción perfecta para poder trabajar en equipo en un mismo proyecto.
- Ofrece todas las ventajas del sistema de control de versiones, Git, pero también tiene otras herramientas que ayudan a tener un mejor control de nuestros proyectos.

En vez de guardar un mismo archivo varias veces. Git nos ayuda a guardar solo los cambios del mismo, además maneja los cambios que otras personas hagan sobre los mismos archivos, así múltiples personas pueden trabajar en un mismo proyecto sin conflictos. Git permite rastrear que miembro realiza los cambios, además de recuperar una versión antigua de manera precisa. Github nos permite publicar un repositorio para trabajarlo de forma remota y colaborar con otros miembros dentro y/o fuera de nuestra organización.

CERTUS

Actividad

Desarrollar el esquema local de Control de Versiones con GIT y recorrido sobre las herramientas Git y GitHub



Gracias