MANEJO DE REPOSITORIOS: GITHUB

Programación de Computadoras II

AGENDA

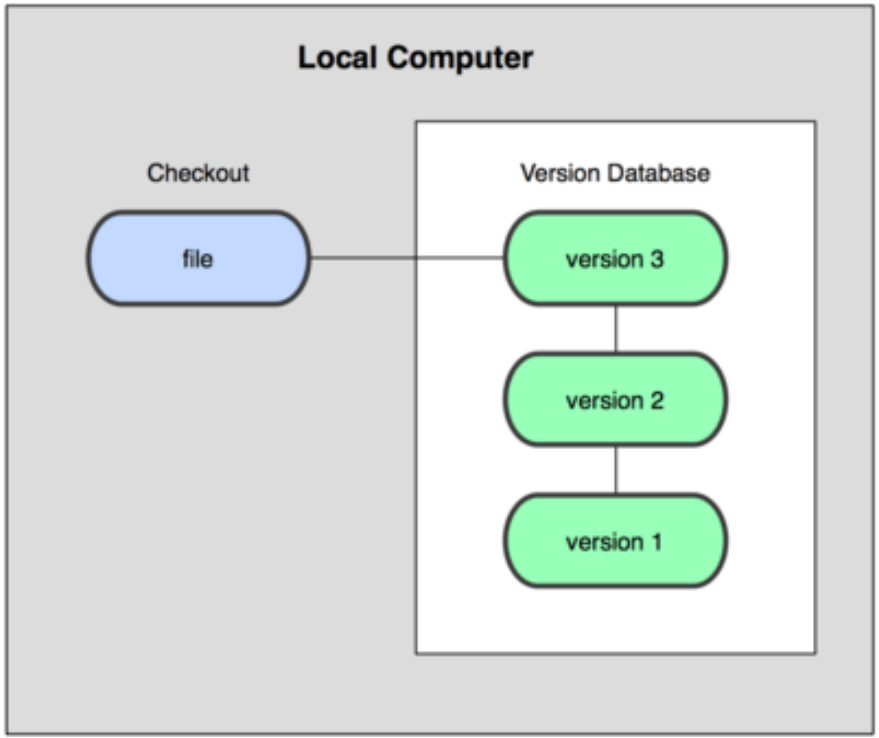
Manejo de repositorios: Github

- El problema de los cambios de versión
- Distintos puntos de vista
- Su origen, historia de amor y odio
- Representación gráfica
- Git es local, íntegro y seguro
- Flujo de trabajo
- Instalación
- Comandos básicos
- Las mejores prácticas
- Demostración
- Otras referencias

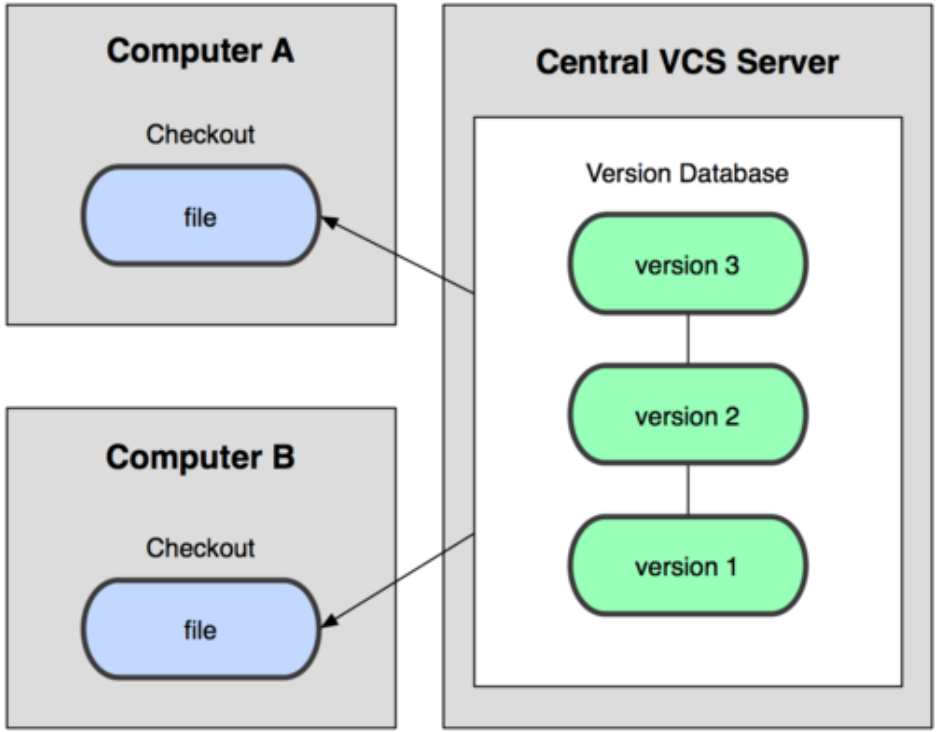
EL PROBLEMA DE LOS CAMBIOS DE VERSIÓN

- Sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo
- Es útil para cualquier profesión, no únicamente programadores
- Permite revertir archivos específicos a su estado anterior, o bien todo el proyecto entero
- Compara cambios a lo largo del tiempo, permitiendo ver quién modificó por última vez un archivo
- Su coste de implementación es muy bajo

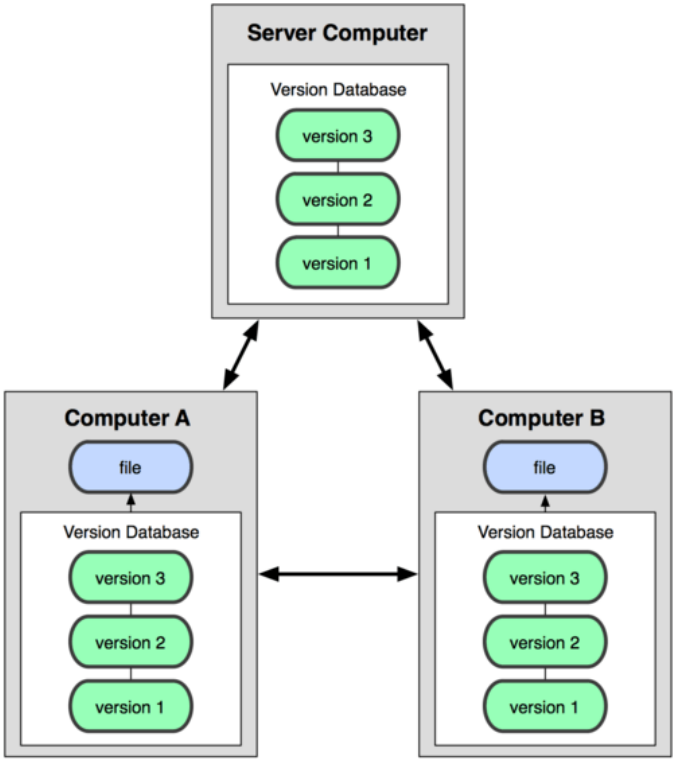
DISTINTOS PUNTOS DE VISTA



DISTINTOS PUNTOS DE VISTA



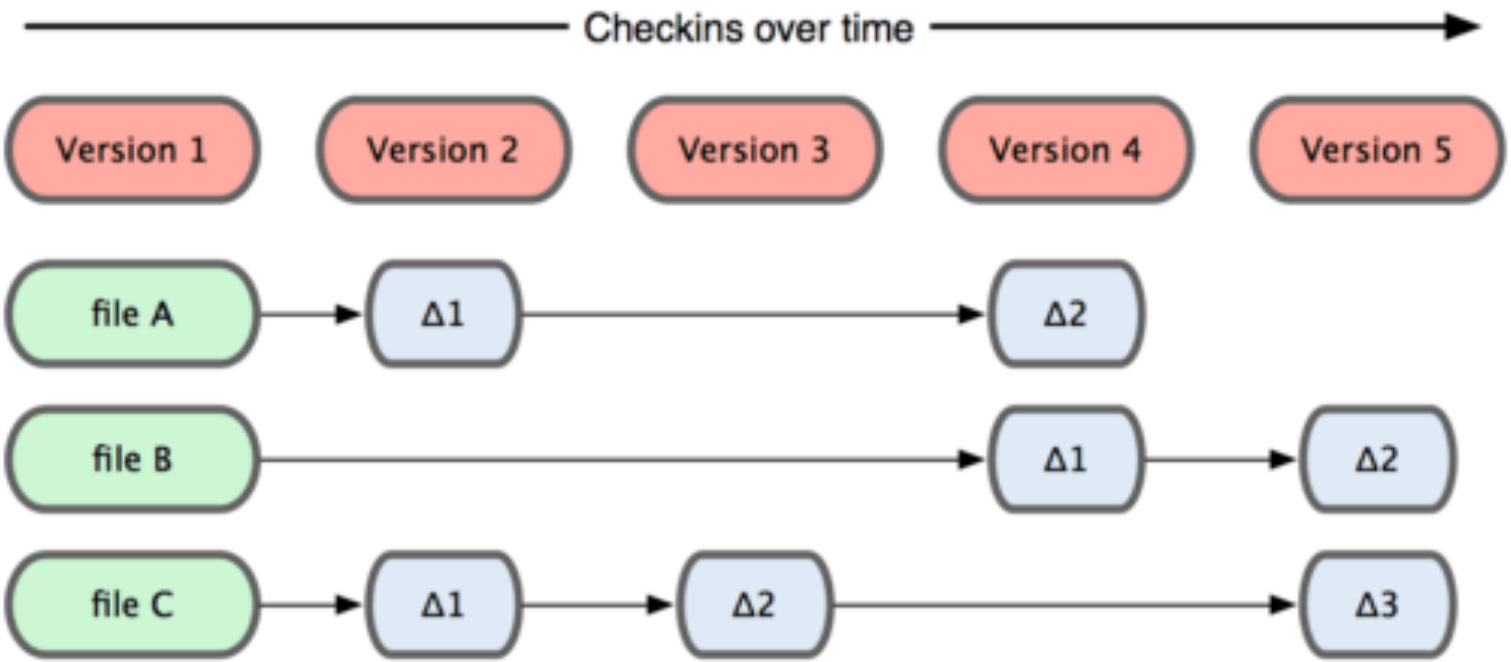
DISTINTOS PUNTOS DE VISTA



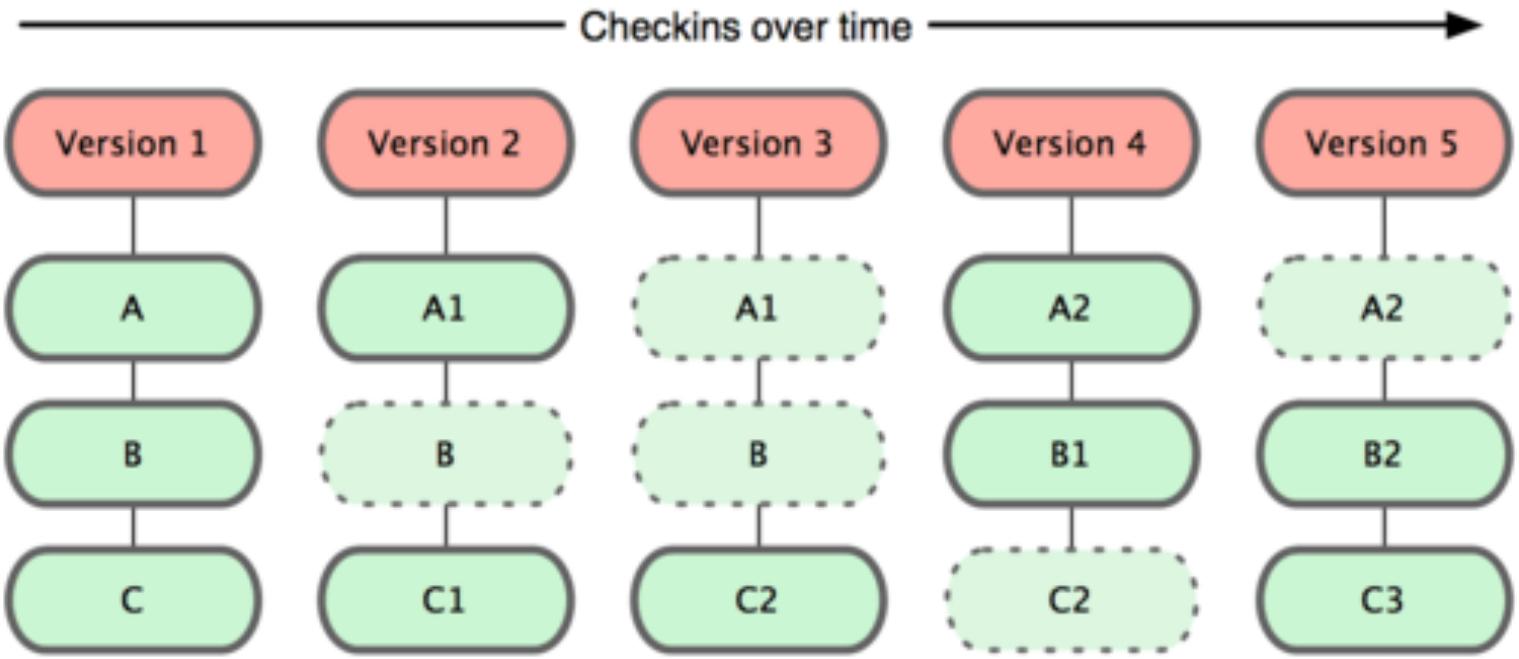
SU ORIGEN, HISTORIA DE AMOR Y ODIO

- Comenzó de la mala relación entre la comunidad desarrolladora del núcleo de Linux y la compañía que desarrollaba BitKeeper
- Linus Torvalds, impulsó a desarrollar su propia herramienta basada en la experiencia de uso de BitKeeper
- Git evolucionó y maduró, desde 2005, para ser fácil de usar, ser tremendamente rápido, eficiente con grandes proyectos y tener un sistema de ramificación increíble para el desarrollo no lineal

REPRESENTACIÓN GRÁFICA



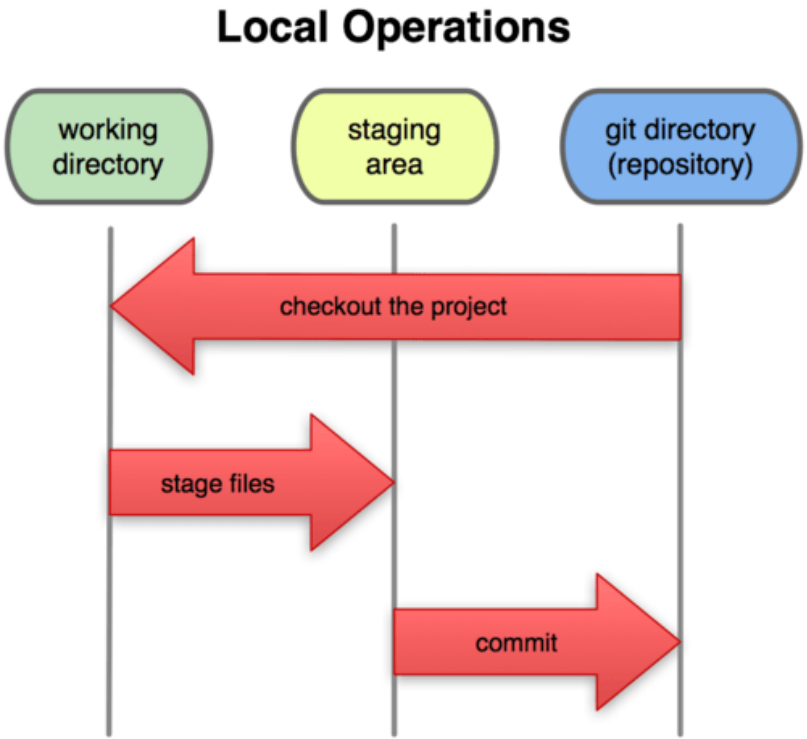
REPRESENTACIÓN GRÁFICA



GIT ES LOCAL, ÍNTEGRO Y SEGURO

- Las operaciones necesitan archivos y recursos locales
- No necesita información de ninguna otra máquina de la red
- Se usa una suma de comprobación (*checksum*) como verificación
- Git guarda todo la información por el valor hash de su contenido
- El mecanismo es SHA-1 (40 caracteres hexadecimales)
- Las acciones realizadas añaden información a la base de datos
- Es difícil conseguir que el sistema borre información ya registrada

FLUJO DE TRABAJO



INSTALACIÓN

- **Windows**

  msysGit (http://msysgit.github.com/)

- **Mac OS X**

  Git Installer (http://sourceforge.net/projects/git-osx-installer)

  Usando el comando `brew install git`

- **GNU/Linux**

  En Fedora se instala usando `dnf install git-core`

  En Ubuntu se instala usando `apt-get install git`

COMANDOS BÁSICOS

- Clonar repositorio existente

```
git clone https://domain.com/user/repo.git
```

- Crear nuevo repositorio local

```
git init
```

- Establecer usuario y correo electrónico

```
git config --global user.name “Analida Martínez”
git config --global user.mail analida@martinez.org
```

COMANDOS BÁSICOS

- Comprobar la configuración actual

```
git config --list
```

- Ver archivos cambiados

```
git status
```

- Ver cambios en los archivos

```
git diff
```

- Agregar cambios al siguiente commit

```
git add <archivo>
```

COMANDOS BÁSICOS

- Entregar todos los cambios

```
git commit -a
```

- Corregir último commit

```
git commit --amend
```

- Mostrar bitácora de commits

```
git log
```

- Ver quién cambió qué y cuándo

```
git blame <archivo>
```

COMANDOS BÁSICOS

- Listar ramas existentes

```
git branch -av
```

- Cambiar rama en el HEAD

```
git checkout <rama>
```

- Crear rama

```
git branch <nueva-rama>
```

- Eliminar una rama

```
git branch -d <rama>
```

COMANDOS BÁSICOS

- Listar repositorios remotos

```
git remote -v
```

- Mostrar repositorios remotos

```
git remote show <remoto>
```

- Agregar nuevo repositorio

```
git remote add <nombre-corto> <url>
```

- Descarga todos los cambios de <remoto>, sin integrar el HEAD

```
git pull <remoto> <rama>
```

COMANDOS BÁSICOS

- Publica todos los cambios locales en un repositorio remoto

```
git push <remoto> <rama>
```

- Descartar cambios locales en un archivo específico

```
git checkout HEAD <file>
```

- Revertir un *commit*

```
git revert <commit>
```

- Obtener ayuda desde la línea de comandos

```
git help <comando>
```

LAS MEJORES PRÁCTICAS

- Hacer *commit* de cambios relacionados
- Hacer *commit* más seguidos
- No hacer *commit* de trabajo medio-terminado
- Probar el código antes de hacer *commit*
- Escribir un buen mensaje en los *commit*
- Un control de versiones no es sinónimo de sistema de respaldos
- Utilizar ramas
- Seguir un flujo de trabajo

OTRAS REFERENCIAS

- **Git – la guía sencilla**

  http://rogerdudler.github.io/git-guide/index.es.html

- **Git Documentation**

  https://git-scm.com/documentation

- **Try Git**

  https://try.github.io/