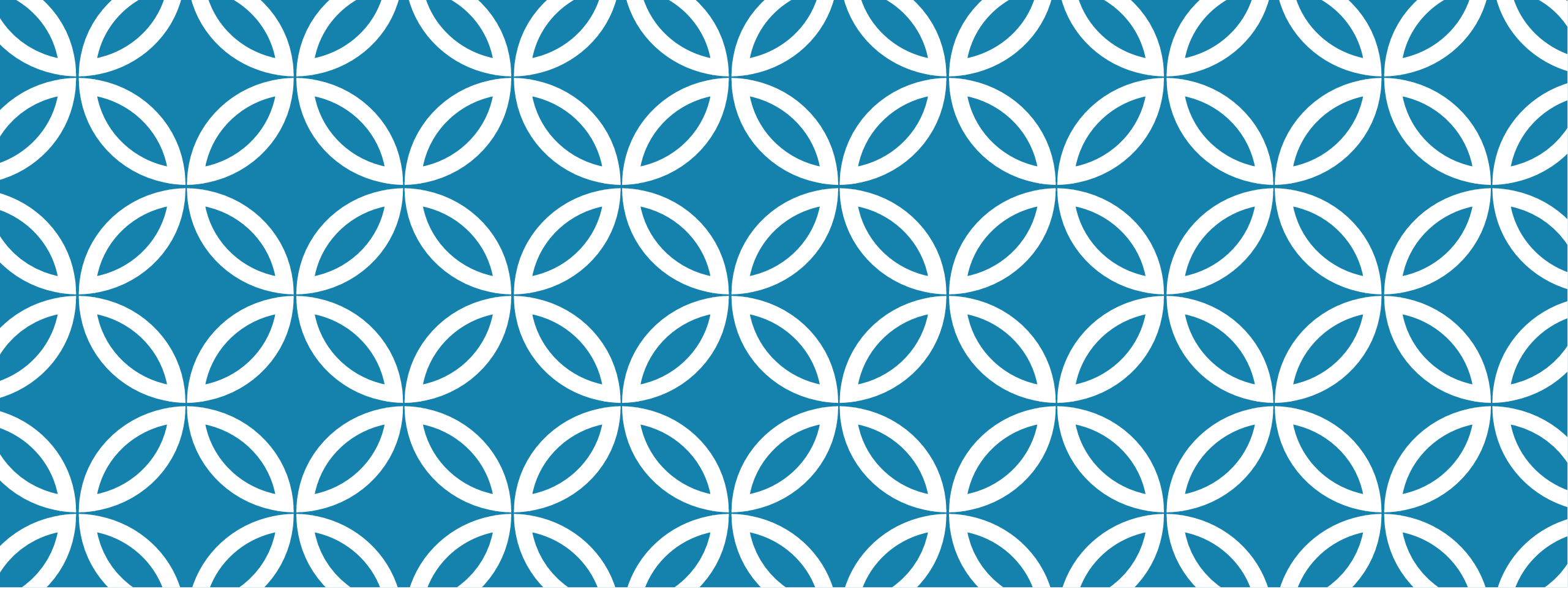


MÓDULO PROFESIONAL ENTORNOS DE DESARROLLO

UD 4.1 – Control de versiones

MODUS OPERANDI CLÁSICO

MODUS OPERANDI CLÁSICO

Trabajando en grupo

- Enviar cambios por mail, o
- Sincronizar cambios por Dropbox, o
- Sincronizar cambios por Google Docs.

SISTEMA DE CONTROL DE VERSIONES

Los **Sistemas de Control de Versiones** son programas que permiten **manejar los cambios** en el código fuente de un proyecto a lo largo del tiempo.

SISTEMA DE CONTROL DE VERSIONES

Llevan un **seguimiento** de las modificaciones que hacemos, y en caso de que nos equivoquemos, es posible volver atrás y comparar el código actual con versiones anteriores para ayudar a arreglar el error.

SISTEMA DE CONTROL DE VERSIONES

También permiten que distintas personas modifiquen el código a la vez y **compartan los cambios**, tratando de prevenir conflictos, y en caso de que los hubiera, ayudando a identificarlos y resolverlos.

SISTEMA DE CONTROL DE VERSIONES

Es decir, permiten...

- Arreglar *accidentes* y volver a versiones anteriores del código.
- Compartir código con otras personas.

SISTEMA DE CONTROL DE VERSIONES

¿Porqué usar un control de versiones nos hará felices?

- Proporciona **copias de seguridad** automáticas de los ficheros.
- Permite **volver a un estado anterior** de nuestros ficheros.
- Ayuda a trabajar de una forma **más organizada.**
- Permite **trabajar de forma local,** sin conexión con servidor. (en distribuídos).
- Permite que **varias personas** trabajen en los **mismos ficheros.**
- Permiten trabajar en **varias funcionalidades en paralelo separado** (ramas).

CÓMO EMPEZAR CON GIT Y TRABAJAR CON UN REPOSITORIO REMOTO GIT

GIT es un sistema de control de versiones (Version Control System)

Un **VCS** sirve como repositorio de códigos de programa, incluidas todas las revisiones históricas. Registra los cambios en los archivos cuando hacemos confirmaciones (commit) y los registra en un log para que se pueda recuperar cualquier archivo en cualquier punto de confirmación (commit)

CÓMO EMPEZAR CON GIT Y TRABAJAR CON UN REPOSITORIO REMOTO GIT

Git fue diseñado y desarrollado inicialmente por **Linus Torvalds**, en 2005, para apoyar el desarrollo del kernel de Linux. GIT es un sistema de control de versiones **distribuido** (**DVCS**).

Otros VCS populares son:

- Sistemas centralizados de control de versiones cliente-servidor (**CVCS**):
  - **CVS** (sistema de versiones concurrentes),
  - **SVN** (Subversion)
  - Perforce.
- VCS distribuidos (**DVCS**):
  - GIT,
  - Merurial,
  - Bazar,
  - Darcs.

El sitio de Git es http://git-scm.com

SISTEMAS DE CONTROL DE VERSIONES DISTRIBUIDOS VS CENTRALIZADOS

Distribuido vs. Centralizado

El control de versiones **distribuido** toma un enfoque entre iguales (peer-to-peer), opuesto al enfoque de **cliente-servidor de los sistemas centralizados**.

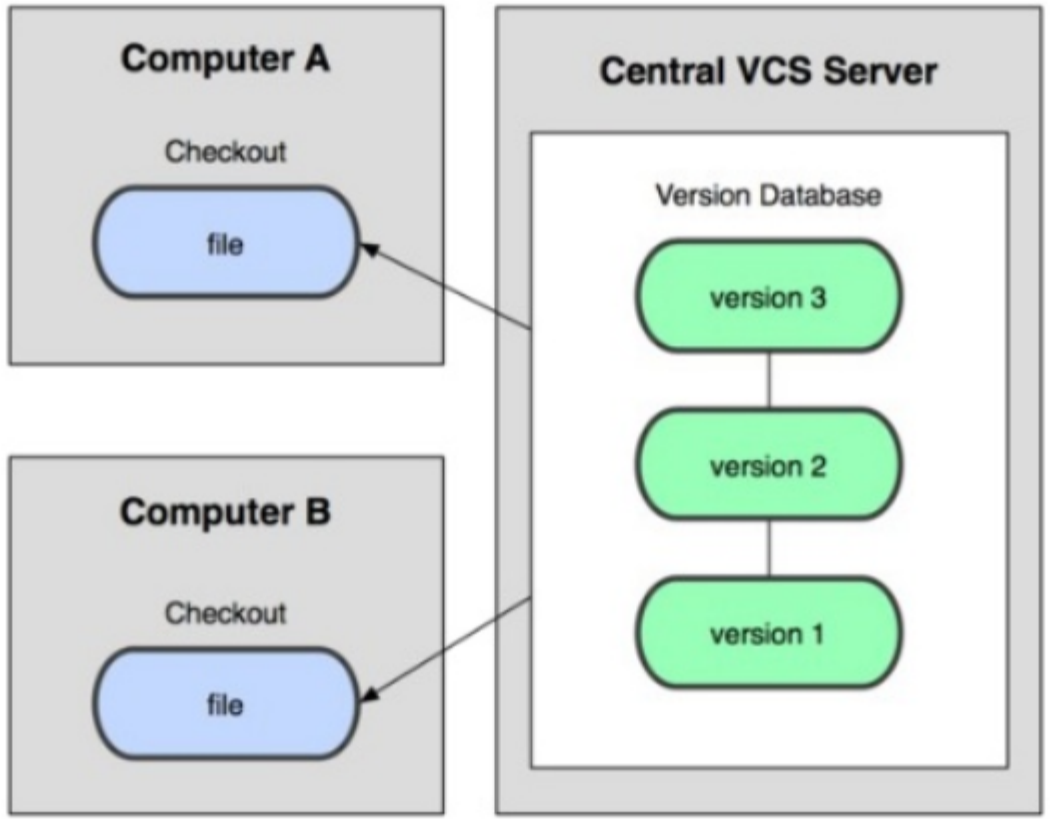
En lugar de un único repositorio central en el cual los clientes se sincronizan, la copia local del código base de cada igual es un repositorio completo. El control de versiones distribuido sincroniza los repositorios intercambiando ajustes (conjuntos de cambios) entre iguales.

Distribuidos: cada usuario tiene una copia completa del repositorio (pueden ser utilizados como copias de respaldo de emergencia). Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos.

Centralizados: existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos)

Sistemas de Control de Versiones Centralizados (CVCS)

Ejemplos: CVS, Subversion, Perforce, SourceSafe, ...

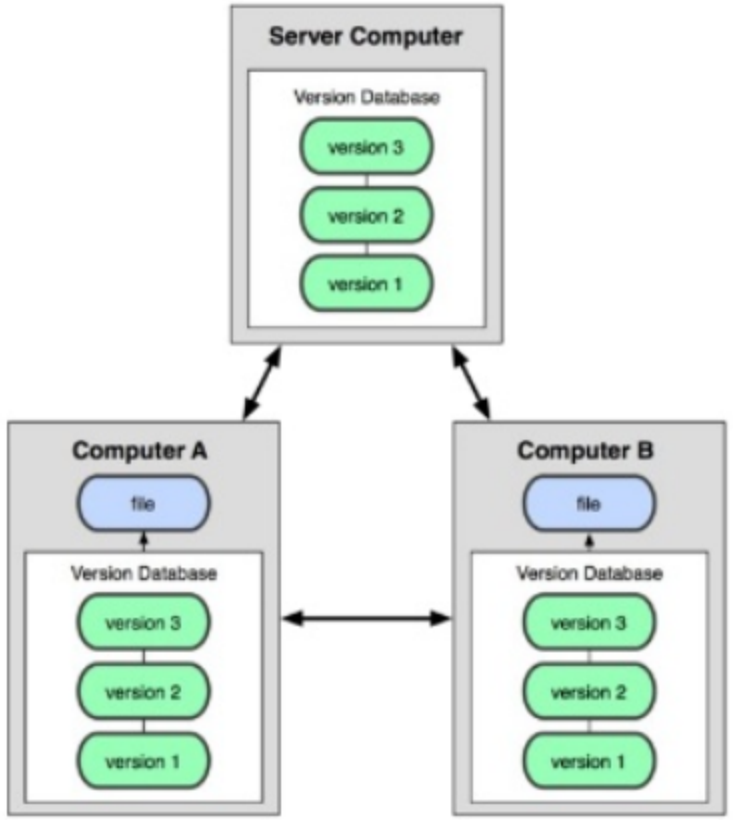


Sistemas de Control de Versiones Distribuidos (DVCS)

Ejemplos: git, Mercurial, Bazaar, BitKeeper,...

Distribuidos: cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos.

Centralizados: existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos)



CONCEPTOS GENERALES

Antes de nada, vamos explicar conceptos generales de los sistemas de control de versiones:

Repositorio

El **repositorio** es el lugar en el que se almacenan los datos actualizados e históricos de cambios.

CONCEPTOS GENERALES

Ramas

Branches o ramas, son **bifurcaciones** en el desarrollo del proyecto.

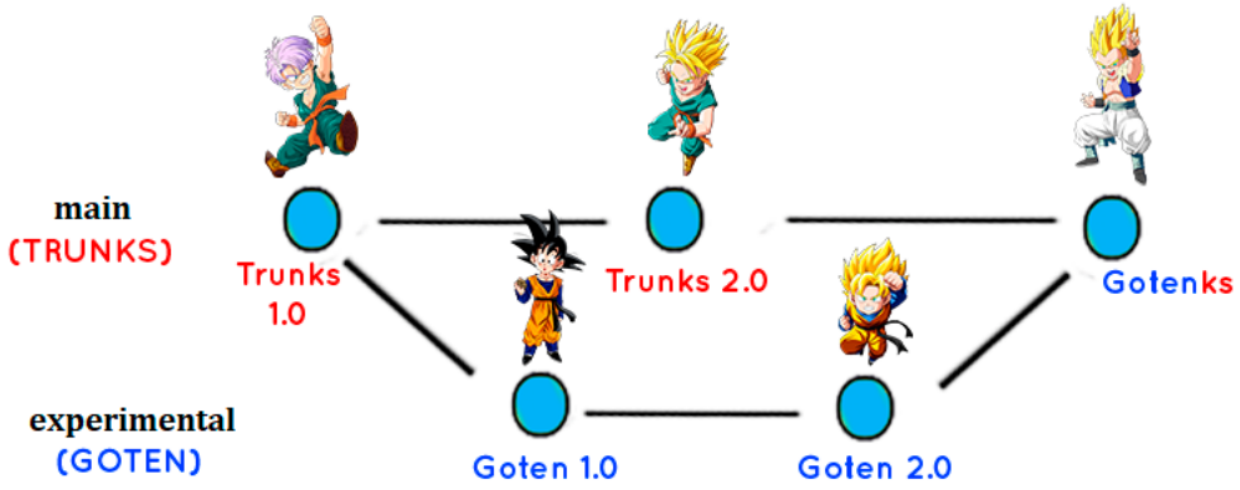
Supongamos que estamos trabajando en un proyecto y queremos añadir nueva funcionalidad al mismo. La forma **adecuada** de hacerlo es crear una **nueva rama** con el nombre de la nueva funcionalidad donde añadiremos nuestros cambios.

Una vez que hayamos testeado profundamente nuestra nueva funcionalidad, moveremos el código de esta nueva rama a la rama principal main. Dicha acción requiere que **mezclemos** el contenido de una de las ramas dentro de la otra.

CONCEPTOS GENERALES

Ramas

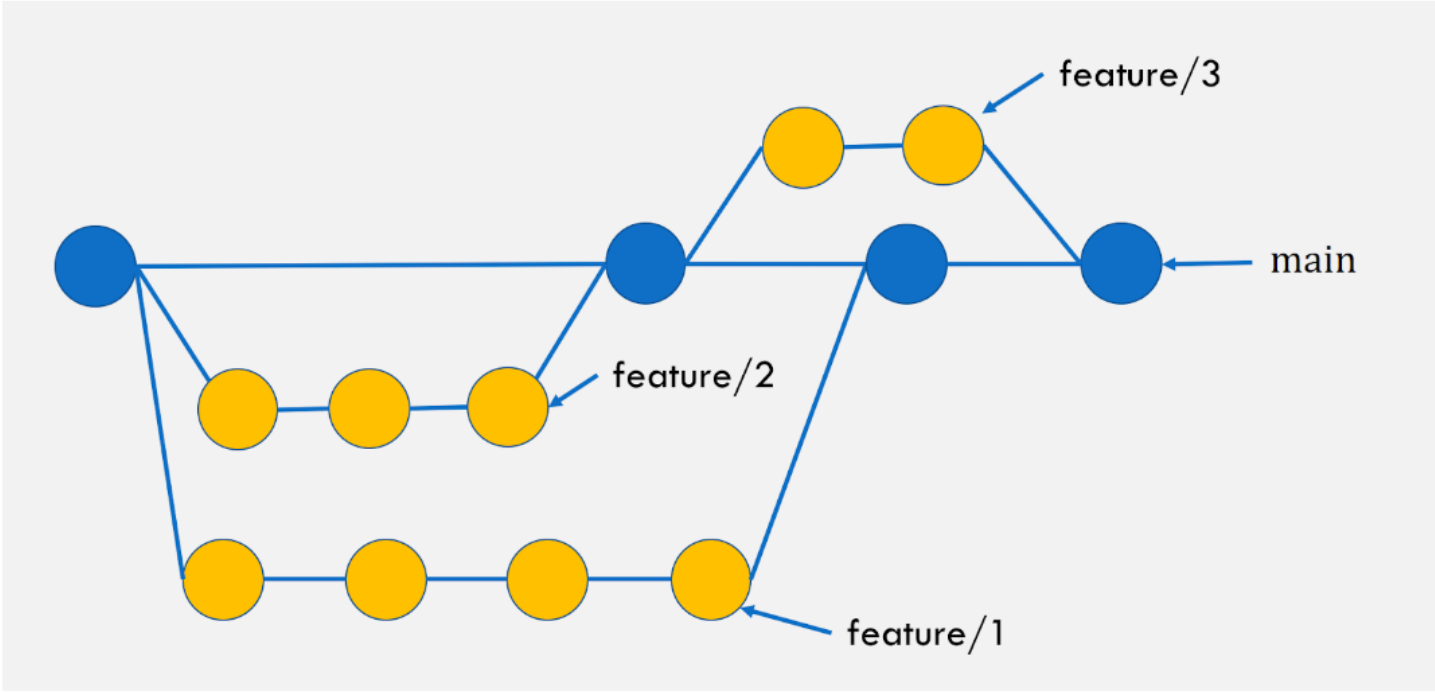
Cuando inicializamos un proyecto con Git automáticamente nos encontramos en una rama a la que se denomina ~~"master"~~ "main".



Ejemplo de rama 1

CONCEPTOS GENERALES

Ramas



Ejemplo de rama 2

CONCEPTOS GENERALES

Commits

Los commits son los puntos de guardado que se van realizando sobre las ramas en los que fijamos los cambios realizados en el código. Podremos volver a ellos para recuperar el estado del código en un determinado momento.

Merge

Un merge es la acción de juntar una rama sobre otra y unificar los cambios. Por ejemplo para añadir una funcionalidad sobre el proyecto principal.

Sincronización (Pull/Push)

Son los procesos para sincronizar el estado de nuestros sistemas locales con el servidor de código principal, tanto para subir los cambios locales como para descargar las últimas actualizaciones.