Introducción a los Repositorios de Código Distribuido

Introducción a Repositorios Git

Diego Madariaga

Contenidos de la clase

1. Introducción a Git
2. Conceptos básicos de Git

1.
Introducción a Git

Software en constante evolución

- Cambios en los requerimientos
- Implementación de nuevas funcionalidades
- Cambios en los desarrolladores
- Cambios en el ambiente

Historial de cambios

- Sistema de control de versiones:
  - Guarda registro de cada cambio (quién, qué, cuándo y por qué)

- Git: Historial de commits

Git

- Uno de muchos sistemas de control de versiones
- Logra diferenciarse del resto debido a su implementación
- La seguridad es una prioridad principal
- Alto índice de adopción

Git: Diferencias del resto

- Tiene una arquitectura distribuida
  - Cada copia del trabajo de cada desarrollador es un repositorio que puede tener el historial completo de cambios

Git: Diferencias del resto

Ejemplo:

- Un desarrollador añade nuevas funciones en la versión 2.0 del proyecto (crea commits con mensajes descriptivos)
- Luego, se cambia a la rama de la versión 1.3 para arreglar un error y lanzar la versión 1.3.1
- Finalmente, vuelve a la rama de trabajo de la versión 2.0 para seguir trabajando

Podría hacer todo esto sin siquiera tener conexión a la red, y enviarlos después al repositorio remoto para confirmar los cambios

Git: Diferencias del resto

- Proyecto de código abierto de calidad
  - Respaldo y fiabilidad por años de desarrollo
  - Gran apoyo de la comunidad
  - Documentación: libros, tutoriales, podcasts, etc.
  - Gratis (para proyectos de todas magnitudes)

Git: Seguridad

- Todos los objetos del repositorio (contenido de archivos, etiquetas, commits, etc) están protegidos criptográficamente
- Se protege el código de cambios accidentales o maliciosos
- Garantiza que el historial completo sea totalmente trazable

Git: Alto índice de adopción

- Comunidad muy grande
- Estándar en muchas compañías de desarrollo
- Muchas herramientas de 3ros se encuentran integradas con git

Git: Principal crítica

- Es difícil de aprender y de usar correctamente

2.
Conceptos básicos de Git

Repositorio Git

- Base de datos con todo lo necesario para manejar la historia de un proyecto
- Provee una copia de todos los archivos y una copia del repositorio en sí
- Junto a un repositorio se guardan valores de configuración propios de cada usuario (ej: nombre y email)

Repositorio Git

- Dentro de un repositorio se almacenan 2 estructuras:
  - Object store
  - Index

Object Store

- Contiene los archivos originales, mensajes de log, información de autor, fechas y toda la información requerida para restablecer cualquier versión o rama del proyecto

- Existen 4 tipos de objetos

Object Store: Blobs

- Cada versión de un archivo se representa como un blob (binary large object).
- Un blob es tratado de forma transparente (no importa su contenido o estructura interna)
- Un blob contiene los datos de un archivo pero no contiene ningún tipo de metadata (ni nombre del archivo)

Object Store: Trees

- Representa un directorio. Puede estar vacío o no
- Contiene identificadores de blob, rutas, y metadatos relacionados a cada archivo en el directorio
- Permiten construir la jerarquía de archivos y subdirectorios en un proyecto

Object Store: Commits

- Contiene información de cada cambio hecho en el repositorio (autor, mensaje, fecha de commit, etc)
- Apunta a un objeto Tree que captura el estado del repositorio en el momento que se realizó el commit
- Todo commit tiene (al menos) un padre, a exepción del commit inicial

Object Store: Tags

- Asigna un nombre arbitrario a un objeto específico (usualmente un commit)
  - 9da581d910c9c4ac93557ca4859e767f5caf5169
  - Ver-1.0-Alpha

Object Store

- Durante el desarrollo de un proyecto, este conjunto de objetos cambiará y crecerá, permitiendo trazar y modelar todas las modificaciones realizadas
- Para mayor eficiencia, Git comprime los objetos en *pack files*, que también se almacenan en el Object store

Index

- Archivo binario dinámico que describe la estructura del directorio del repositorio completo
- Captura la estructura general de una versión del proyecto en un determinado instante
- El estado del proyecto puede representarse por
  - Commit + Tree de cualquier punto en la historia del proyecto
  - Estado futuro hacia el que se está desarrollando activamente

Index

- Permite una separación entre el desarrollo incremental y la confirmación de esos cambios (commits)

Index: ¿Cómo funciona?

- Desarrollador: ejecuta comandos Git para “preparar” cambios en el índice. (stage)
  - ¿Qué cambios?: añadir, eliminar o editar uno o varios archivos
- El índice graba esos cambios, manteniéndolos hasta el momento que estén listos para ser confirmados (commit)
- Una vez en el índice, se dice que los archivos están en el área de preparación (staging area)

Index: ¿Cómo funciona?

- Permite una transición gradual (guiada por el programador) entre un estado del repositorio y otro (presumiblemente mejor)

Visualización de Object Store

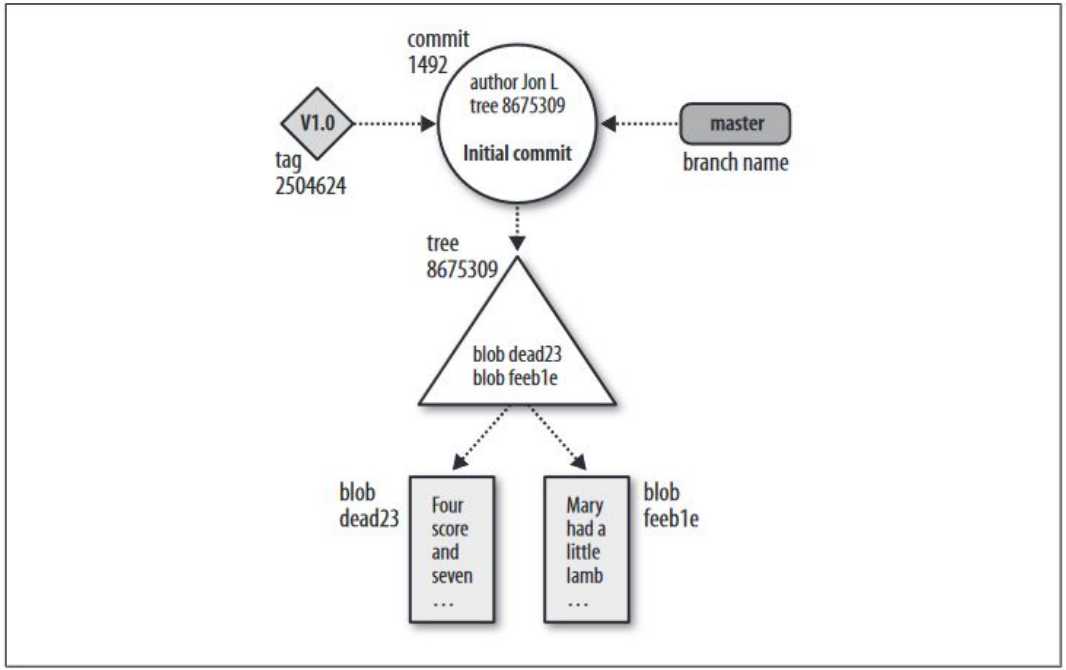

Figure 4-1. Git objects

- **Blobs**: rectángulos
- **Trees**: apuntan a Blobs u otros Trees (pueden ser apuntados por distintos commits)
- **Commits**: apuntan a un Tree particular, introducido por el commit
- **Branch**: no es un objeto Git, pero es crucial para nombrar commits

Visualización de Object Store

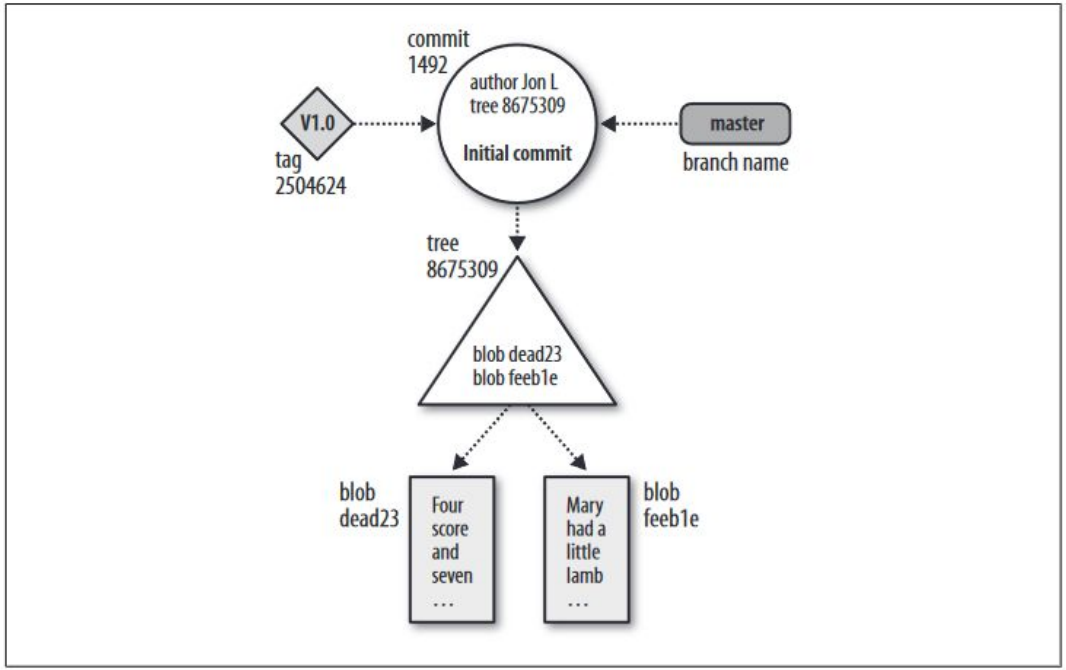

Figure 4-1. Git objects

- Estado de un repositorio después de un único commit inicial que añadió dos archivos
- Ambos archivos están ubicados en el directorio de más alto nivel

Visualización de Object Store

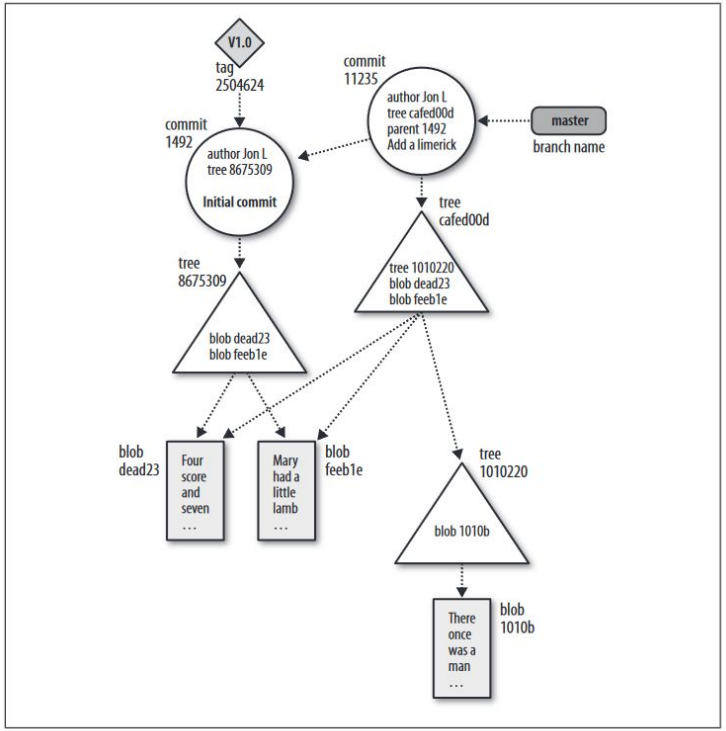

Figure 4-2. Git objects after second commit

El nuevo commit añade un objeto Tree, que representa la adición de un nuevo subdirectorio

Introducción a los Repositorios de Código Distribuido

Introducción a Repositorios Git

Diego Madariaga