Introducción a los Repositorios de Código Distribuido

Manejo Básico de Repositorios Git

Diego Madariaga

Contenidos de la clase

1. Seguimiento de contenido
2. Manejo de archivos
3. Vista de modelo de objetos

1.
Seguimiento de contenido

Repositorio Git

- Dentro de un repositorio se almacenan 2 estructuras:
  - Object store
    - Blobs
    - Trees
    - Commits
    - Tags
  - Index: Separación entre el desarrollo incremental y la confirmación de esos cambios (commits)

Visualización de Object Store

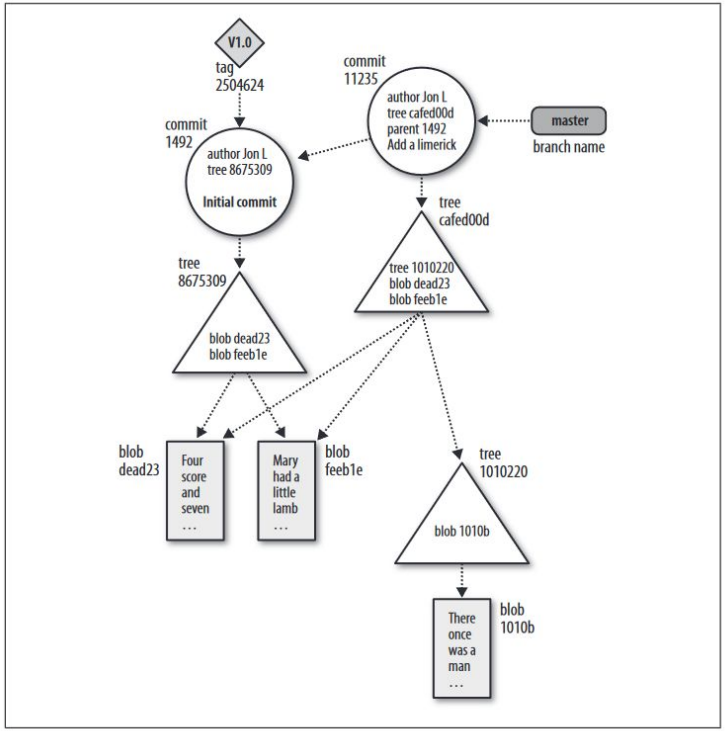


Figure 4-2. Git objects after second commit

Identificadores de objetos

- Object store: Organizado e implementado como un sistema de almacenamiento “direccionable por contenido”

- Nombres únicos para cada objetos:
  - Ej: 770ef6f29c5d298a8933c7c53aa20034000ab6c6

Identificadores de objetos

- Los identificadores de objetos son producidos al aplicar el algoritmo de hash seguro SHA-1 al contenido de los objetos
- SHA-1(objeto Git) -> 160 bits
  -> 2^{160} números distintos (~10^{48})
- Prácticamente irreversible
- Teóricamente pueden haber colisiones

Identificadores de objetos

- 160 bits son representados en números hexadecimales de 40 dígitos
- Usualmente, se utiliza un prefijo de estos 40 dígitos para identificar a los objetos de forma más “amigable”
- Object ID <-> SHA-1 hash code

Identificadores de objetos

- Algoritmo SHA-1 siempre retorna el mismo valor para contenidos idénticos
- Los identificadores son únicos globalmente
- Se pueden comparar 2 blobs en cualquier parte del Internet para chequear si representan al mismo archivo comparando sus identificadores SHA-1

Git traza contenido

- Más que un sistema de control de versiones
  - Git es un sistema de seguimiento de contenido

- Diferencia importante frente a otros VCS

Git traza contenido

- Object store está basado en el hash del contenido de los archivos
  - NO en los archivos ni en los directorios originales

- Cuando Git guarda un archivo en el Object store, lo hace basado en el hash de su contenido
  - NO en su nombre

- Git no hace seguimiento de archivos. Hace seguimiento de contenidos (asociados a archivos)

Git traza contenido

- Si un proyecto tiene dos archivos en distintos directorios pero con el mismo contenido
  - Git guarda solo una copia de su contenido (blob)
  - Sin importar su ubicación
  - Sin importar sus nombres de archivo

Git traza contenido

- Git guarda todas las versiones de los archivos del proyecto (como blobs en el Object store)
  - No guarda las “diferencias” entre versiones de archivos
  - Estas son calculadas dinámicamente para mostrarlas al usuario
- El historial de Git se compone de muchos blobs con distintos identificadores SHA-1

2.

Manejo de archivos en Git

Índice

- Separación entre el desarrollo incremental y la confirmación de esos cambios (commits)
- Los cambios se acumulan en el Índice para luego ser confirmados como un cambio único

Índice

- Puede interpretarse como un conjunto modificaciones previstas
  - Ediciones
  - Creación de archivos
  - Eliminación de archivos
- Culmina en un commit

Clasificación de archivos

- **Tracked**: Archivo perteneciente al repositorio o que se encuentra ya en el área de preparación (Index)
  - git add filename

- **Ignored**: Archivo declarado explícitamente como ignorado. Ej: archivos temporales, archivos generados al compilar, notas personales, etc.

Clasificación de archivos

- **Untracked**: Archivo que no pertenece a ninguna de las 2 categorías previas

Uso de git add

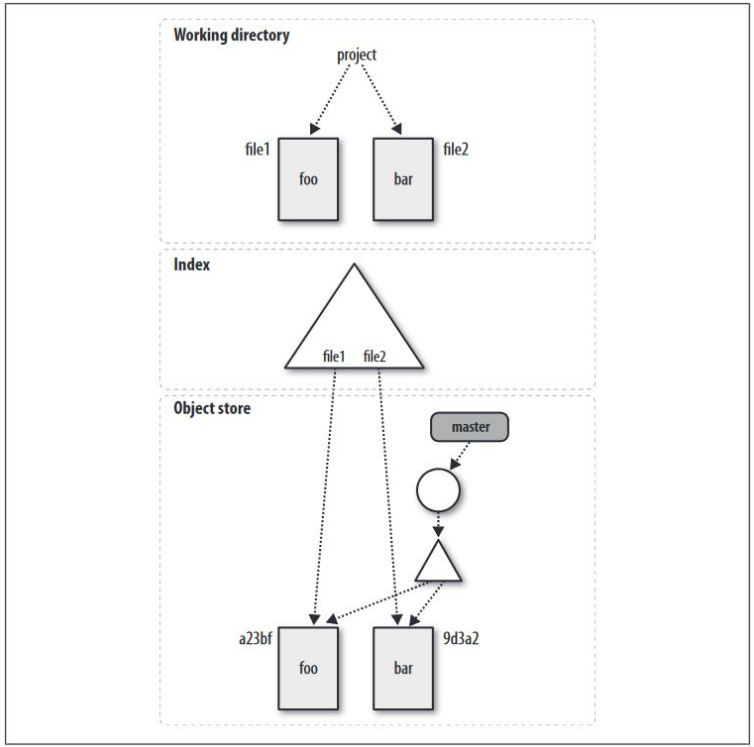
- Prepara uno o más archivos (cambian su estado a Tracked)
- Se copia su contenido al object store (identificado con su hash SHA-1)

Uso de git rm

- Remueve un archivo del repositorio y **también del directorio de trabajo**
- No se elimina el historial del archivo eliminado (continuará en el object store)
- Para eliminar solamente del Index: git rm --cached

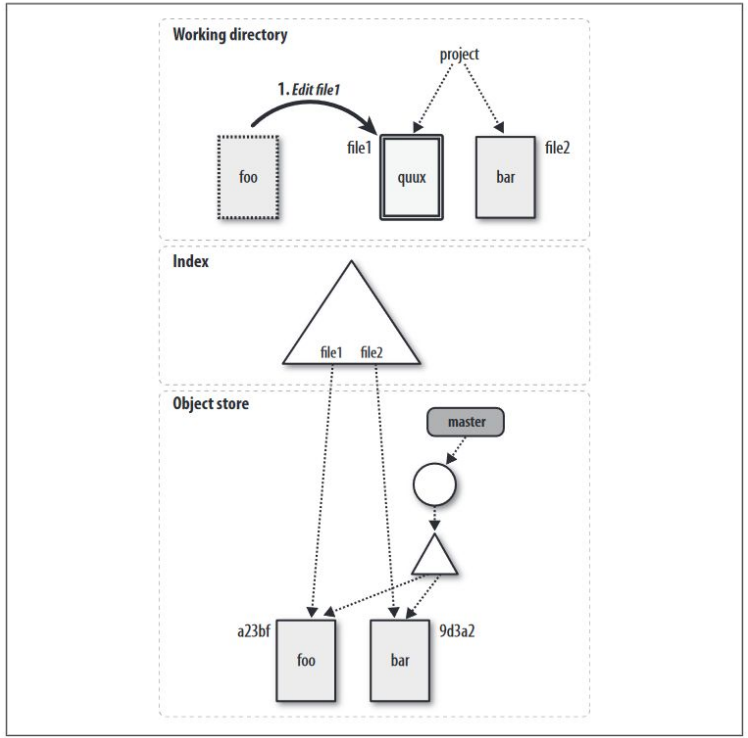
3.
Vista de modelo de objetos

Vista del modelo de objetos de Git



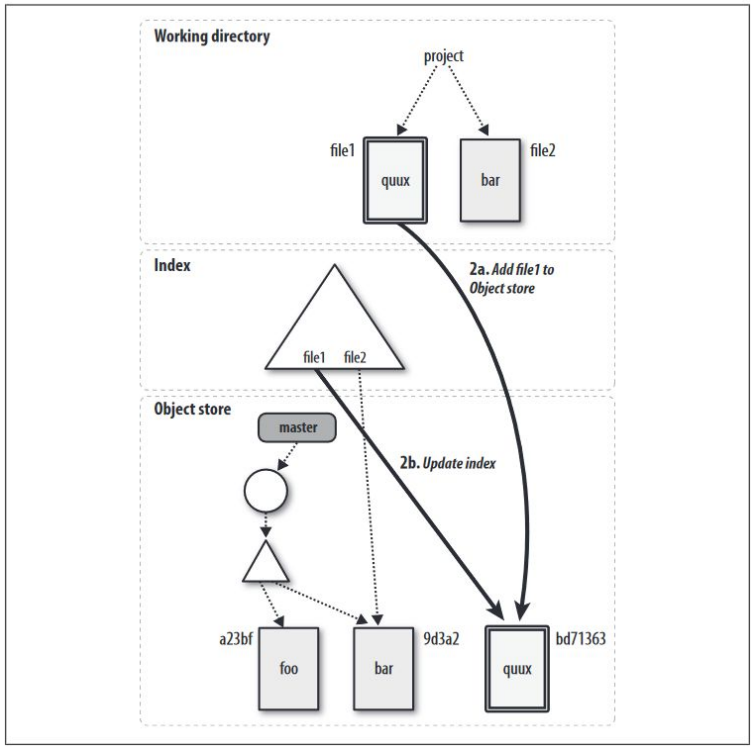
- Directorio de trabajo contiene 2 archivos con contenidos “foo” y “bar”
- Rama principal contiene un commit que referencia a un tree que apunta a 2 blobs (mismos 2 archivos)
- El índice contiene los identificadores SHA-1 de ambos archivos, lo cual muestra concordancia y sincronización entre los 3 ambientes

Vista del modelo de objetos de Git



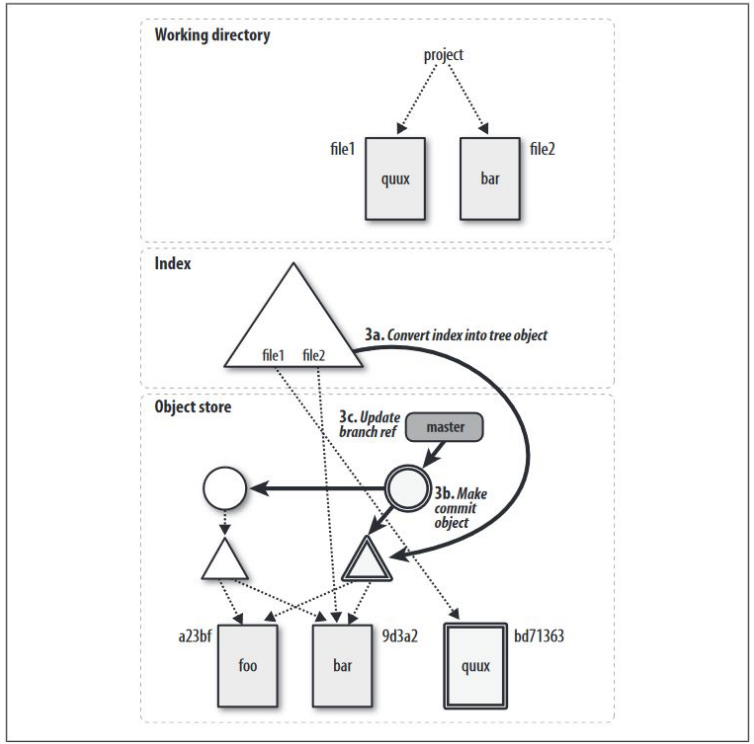
- Cambios luego de editar el archivo 1
- No hay cambios en Index ni Object store

Vista del modelo de objetos de Git



- Cambios luego de ejecutar
  - git add file1
- Git toma el nuevo contenido de file1, computa su SHA-1 y lo guarda en el Object store
- Git registra en el Index el cambio del contenido de file1, por lo que cambia su ruta hacia el nuevo blob

Vista del modelo de objetos de Git



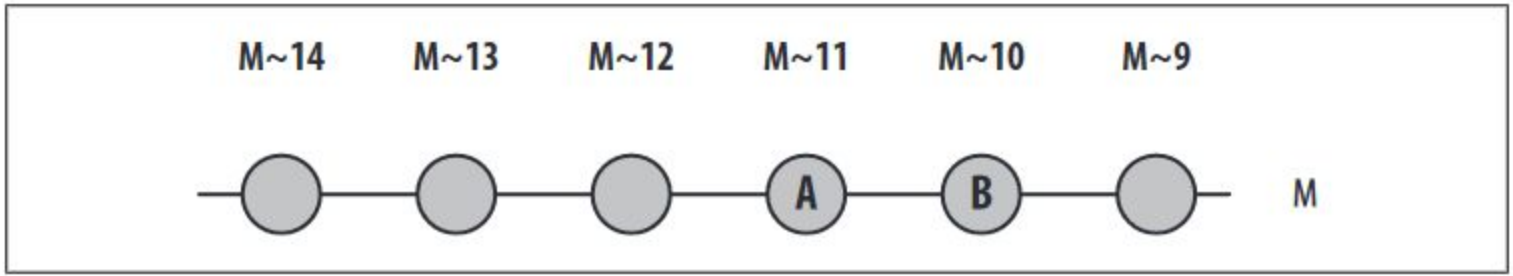
- Después de ejecutar git commit
- El contenido del Index se transforma en un objeto Tree dentro del Object Store
- Se crea un objeto commit en el Object Store (nuevo último commit)

4.
Manejo de commits

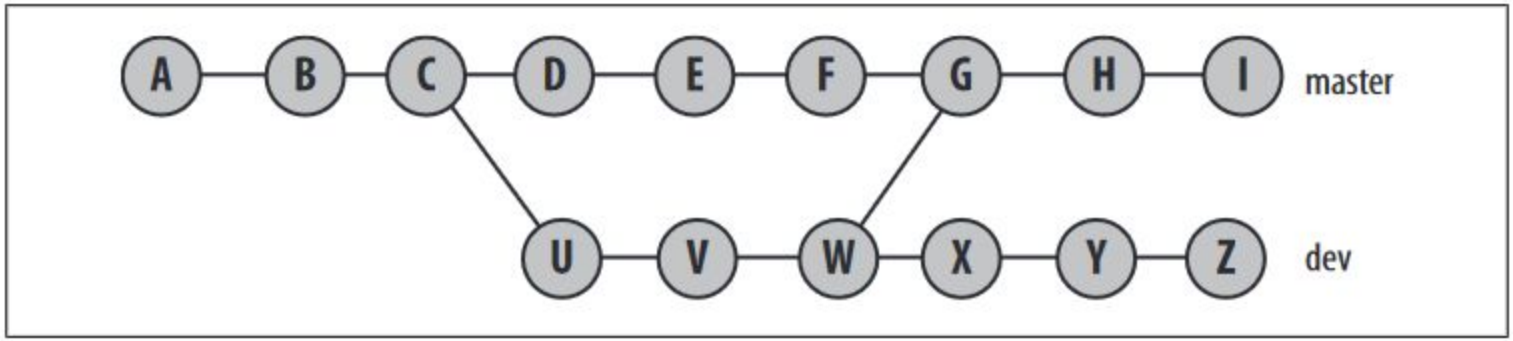
Commits

- Dependen del estilo de programación de cada persona
- Idea: Conjunto de cambios atómico
  - Tienen un propósito bien definido (el “por qué”)
  - No hay problema en realizar gran cantidad de commits
- Historial de Git se compone como una sucesión de commits

Historial de commits lineal



Historial de commits no lineal



Buscando entre commits

- Git provee una serie de herramientas para inspeccionar el historial de commits y buscar puntos específicos de él

Buscando entre commits

- git bisect: Ayuda a identificar el commit en donde se introdujo un error o bug
  - Se debe especificar un commit anterior donde se sabe que no está el error
  - Se debe especificar un commit en donde se sabe que está el error
  - git bisect usa búsqueda binaria para encontrar el commit

Buscando entre commits

- git blame: Ayuda a identificar el autor de ciertos cambios específicos (por ejemplo, una línea de código)
  - Información acerca del autor y del commit en donde se introdujo el cambio

Buscando entre commits

- git log: Historial de cambios en el repositorio
  - Opción -S (piqueta) para realizar búsqueda por fuerza bruta de un string determinado
  - Muestra todos los commits en donde se añadió o se eliminó alguna línea de código que incluyera cierto string

Introducción a los Repositorios de Código Distribuido

Manejo Básico de Repositorios Git

Diego Madariaga