From plumber to porcelain

Introduction

- Its going to be quite more than an hour.
- Need participation, ask for everything and propose, will answer, try or guess.

1. What is Git.
    - Git ∈ { *D*CVS }
    - `git init`
    - Show `.git` file contents. Contains an individual *git* repository.
    - Everything outside `.git` is the working directory.

2. What is also Git.
    - Git ∈ { K,V stores }
    - Different type of objects with a value from a *sha* function.
    - *Draw empty repository with working directory and staging area*.

Repository content and management

1. Types of objects:
    - Stored in `.git/objects` as files.
    - *Draw objects section*.
    - *Draw object types and used while using them as a mindmap*.
    - *Draw examples of each object type examples*.
    - `blob`: Contain data.
        - Examples:
            - Get *hash* from `echo`.
            - Get *hash* from file and compare with `echo`.
            - Write an object.
            - Show object content.
            - Write object to another file name.

    - Commands:
        - hash-object [--stdin]
        - hash-object -w
        - cat-file -p
        - cat-file -t
    - No name, only content.
    - Different content -> Different key -> Different objec.
    - Only stored once.
- tree: Gives names and permissions to blob or other tree objects.
    - Examples:
        - Add files to index.
        - Show index content.
        - Create a one level tree.
        - Show tree content.
        - Add directory with files to index moving an object.
        - Create two level tree.
    - Commands:
        - update-index [--add|--remove]
        - ls-files -s
        - write-tree
        - cat-file (-t|-p)
        - ls-tree
    - Can be only be created through index.
    - Tree only one level, index multiple.
    - Only privileges, *pointer* and name. No metadata.
    - Talk about metadata in index.
    - Review, same file in both trees.
    - Talk about and pointers.
- commit: Pointer to tree with content.
    - Examples:
        - Create a commit from first tree.
        - Create a commit from second tree.
        - Show revision list.

        - Repeat full process with a third commit.
        - git cat-file -p <commit> | get hash-object -t commit --stdin.
        - git cat-file -p <commit> | get mktree.

    - Commands:
        - write-tree
        - commit-tree <tree> [-p <parent>]

    - Content:
        - Author (name, e-email, time).
        - Commiter (name, e-email, time).
        - Text (header, description).
        - tree and [parent commit]*.

    - First commit no parent.
    - Talk about and pointers.
    - By design, changes in history changes everything. Is a different history.
    - Never two commits equal - time.
    - Hash function = <type><size>\0<content>.

- tag - Ignored in this talk. Pointer to commit with text and author. Can be signed.

2. Refs: Objects alone are impossible to maintain.

- Stored in .git/refs as files.
- Pointers to commits.
- *Draw refs section*.
- *Draw commads while using them*.
- *Draw examples of each reference movement*.
- Talk about HEAD and symbolic refs.
- *Draw HEAD near staging area and working directory*.
- *Different arrow type for symbolic refs*
- Examples - *Use commit scripts*.
    - Clean everything and repeat using HEAD and master.
    - Create branch devel after commit 2.
    - Apply commit 3 and show log.

- Commands:
    - rev-parse
    - `rev-list [--all]
    - update-ref [--no-deref]
    - symbolic-ref

- Talk about how commit is done.
- Show references in history.
- Talk about detached head. *Draw example*.

3. Repository, staging area and working directory.

    - Only way to interchange files between working directory and repository.
    - *Draw commands while examples with arrows*
    - Examples:
        - Already seen write to staging area from working directory.
        - Join read from index with refs.
            - checkout
            - reset

    - Commands:
        - update-index
        - write-tree
        - read-tree
        - checkout-tree

Plumber commands

1. Already seen:

    - add
    - commit
    - checkout
    - branch
    - reset

2. Pending:

    - log:

    - Rev list with cat-files.

- merge:
    - Guess if merge is needed - merge origin branch commit already in rev-list.
    - Check if fast forward - current HEAD commit in merge origin branch rev-list.
    - Do fast forward - change HEAD to merge origin branch.
    - Explain no fast forward.
        - Flag in index when multiple copies stored.

Conclusions

- We have not talk about of pull, push or clone from other repositories.
- All scripts and examples in my GitHub account[1].
- Its not necessary but helps know this.
- Impact of design in operations: checkout, *diffs*, rewriting history.

Talk about

- Revert full working directory only to previous commit... ¿different index file?
- Bamboo branch failure causes master failure.
- symbolic-ref previously implemented as ln -s and deprecated due to multi-platform compatibility.
- Why is a distributed CVS and what means a checkout.

1. https://github.com/pablerass/talk-from-plumber-to-porcelain