CVCS vs. DVCS In a Nutshell

CVCS vs DVCS in a Nutshell

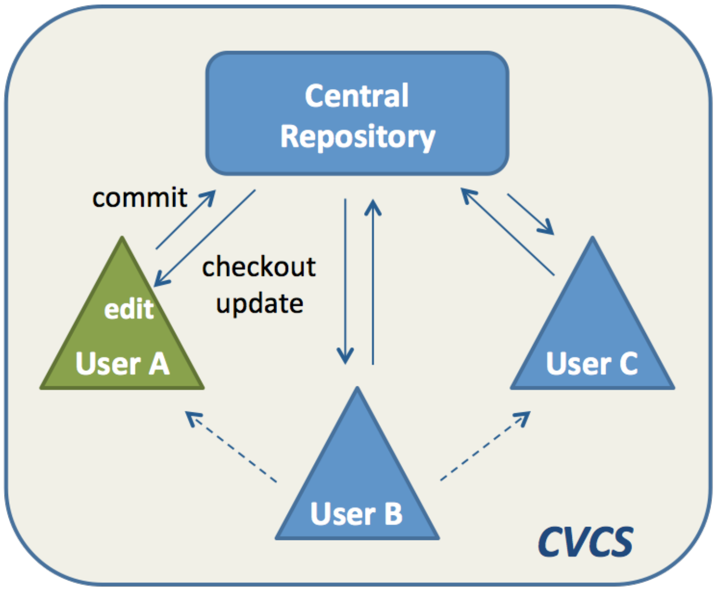
Traditional version control (CVCS) helps you backup, track and synchronize files. Distributed version control (DVCS) makes it easy to share changes.

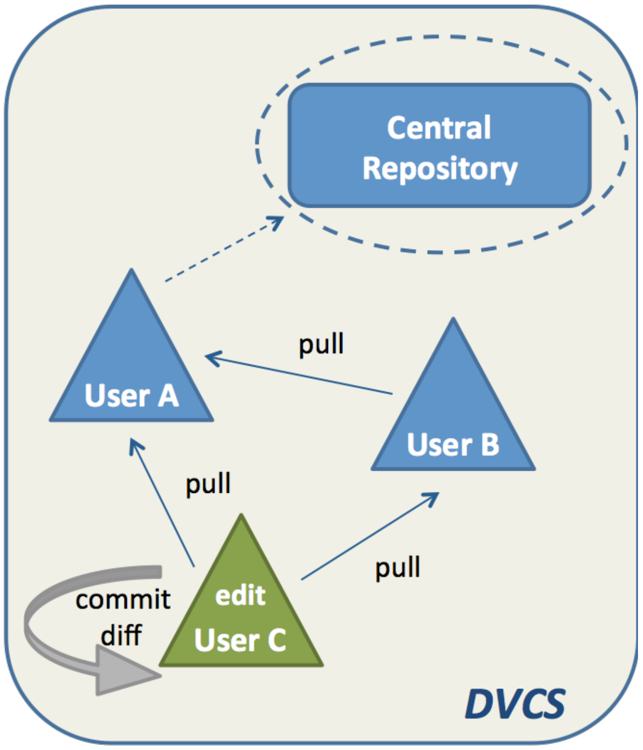
With DVCS git, you can get the best of both worlds: simple merging and centralized releases.

Questions? Ask us!

Key Concepts

- Centralized version control systems (CVCS) focuses on synchronizing, tracking, and backing up files.
- Distributed version control systems (DVCS) focuses on sharing changes; every change has a guid or unique id.
- Recording/downloading and applying a change are separate steps (in a centralized system, they happen together).
- Distributed systems have no forced structure. You can create “centrally administered” locations or keep everyone as peers.
- DVCS new terminology
    - Pushes refer to sending a change to another repository (permissions may be required)
    - Pulls refer to grabbing a change from a repository





DVCS Core Advantages

- ***Everyone has their own local sandbox.***
    - You can make changes and roll back, all on your local machine.

    - No more giant checkins; your incremental history is in your repo.
- ***DVCS git works offline.***
    - You only need to be online to share changes.
    - Otherwise, you can happily stay on your local machine, checking in and undoing, no matter if the “server” is down or you’re on an airplane.
- ***DVCS git is fast.***
    - Since diffs, commits and reverts are all done locally.
    - There’s no sketchy network or server to ask for old revisions from a year ago.
- ***DVCS handles changes very well.***
    - Distributed version control systems were built around sharing changes.
    - Every change has a guid which makes it easy to track.
- ***Branching and merging is easy.***
    - Because every developer “has their own branch”, every shared change is like reverse integration.
    - The guids make it easy to automatically combine changes and avoid duplicates.
- ***With DVCS, there is less management.***
    - DVCS systems are easy to get running since there is no “always-running” server software to install.
- DVCS systems may not require you to “add” new users since you can just pick what URLs to pull from.

DVCS Core Disadvantages

- ***You still need a backup.***
    - Some claim your “backup” is the other machines that have your changes, but what if they didn’t accept them all? ** What if they’re offline and you have new changes?
- ***You still want a machine to push changes*** to “just in case”.
    - In Subversion, you usually dedicate a machine to store the main repo; do the same for a DVCS.
- ***There’s not really a “latest version”.***
    - If there’s no central location, you don’t immediately know whether to see others for the latest version.
    - A central location helps clarify what the latest “stable” release is.
- ***There aren’t really revision numbers.***
    - Every repo has its own revision numbers depending on the changes.
    - Instead, people refer to change numbers which are not intuitive. But, you can tag releases with meaningful names.

Questions? Ask us!

—

1 http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/
2 http://www.infoq.com/articles/dvcs-guide