# version control

What is “version control”, and why should you care? Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

**Types of Version Control**

1. **LVCSs (Local Version Control Systems)**

**Tools**: RCS

**Drawbacks**: Developers can not collaborate with on other systems. single point of failure

1. **CVCSs (Centralized Version Control Systems)**

**Tools**: CVS,Subversion, and Perforce

**Advantages**: Everyone knows to a certain degree what everyone else on the project is doing.

Administrators have fine-grained control over who can do what.

it’s far easier to administer a CVCS than it is to deal with local databases on every client.

**Drawbacks**: Single point of failure

If the hard disk the central database is on becomes corrupted, and proper backups haven’t been kept, you lose absolutely everything – the entire history of the project except whatever single snapshots people happen to have on their local machines.

1. **DVCSs (Distributed Version Control Systems)**

**Tools**: Git, Mercurial, Bazaar or Darcs

**Advantages**: Clients don’t just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

**Drawbacks**:

Deleting and renaming files

$ rm file2.txt

$ git status

$ git add –u

$ git status

$ vim file3.txt

Add data and save.

$ git status

$ git add file3.txt

$ git status

$ mv file1.txt new\_file\_name.txt

$ git status

$ git add –A

$ git status

$ git commit –m “Reorganized the feature”

$ git status

$ git log

Undoing changes to the working copy

$ vim README.TXT

Modify data and save

$ git status

TO undo the changes to previous head

$ git checkout README.TXT

$ git status

$ cat README.txt

Previous details are there.

To reset the data in working copy to HEAD.

$ rm file1.txt README.TXT

$ git status

$ git reset –hard

$ git status

Undoing and redoing changes in the repository

$ git status

$ git log

$ git reset --soft HEAD~1

$ git status

$ git log

Cleaning the working copy

$ touch temp1.txt temp2.txt

$ git status

$ git clean

$ git clean –n (shows What it would do)

$ git clean –f

$ git status

Ignoring files with .gitignore

If you have temp files and logs that don’t need to be commited to git. You can use the following commands.

$ mkdir logs

$ touch logs/log.tx

$ git status

Add .gitignore file in root of repository.

$ vim .gitignore

Add path to ignore files inthat directory.

/logs/\*.txt

Or

/logs/\*.log

or

/logs

$ git status

$ git add .gitignore

$ git commit –m “Added .git ignore”

$ git status

$ git log

## Working remotely with GIT

Cloning a remote repository

Listing remote repositories

Fetching changes from a remote

Merging changes

Pulling from a remote

Pushing changes remotely

Working with tags

Cloning a Remote Repository

$ git clone https://github.com/jquery/jquery.git

3. Basic Repository Statistics

To check line count

$ git log –online | wc –l

$ git log –online –graph

Showing different merges and