LEARNING version-control

#version-control

Table of Contents

About

Chapter 1: Getting started with version-control

Remarks

This section provides an overview of what version-control is, and why a developer might want to use it.

It should also mention any large subjects within version-control, and link out to the related topics. Since the Documentation for version-control is new, you may need to create initial versions of those related topics.

Examples

Selection, Installation & Setup

Selection of a Version Control System

Of course many organisations or projects already have a preferred or selected version control system if this is the case for you just skip to client installation.

There are a number of things to consider when selecting the version control system to use, *given unordered as everybody will have different priorities*:

- **Costs** ranges to nearly nothing to huge sums *breaks down into*
    - **Initial Costs** *(non-recurrent)*
        - Server Software Initial Purchase price - *ranges from zero to many thousands*
        - Server hardware & operating provision costs
    - **Ongoing Usage Fees (Server)** *(recurrent)*
        - Any Server Software annual or monthly licence &/or maintenance fees - *ranges from zero to many thousands*
        - Possible server storage fees *hosted services often have some sort of storage fees*
        - For hosted solutions you may also be looking at CPU time charges, bandwidth charges, etc.
        - Administration & Maintenance costs *some high end version control systems require multiple full time administrators*
    - **Per developer/seat setup charges** *only when adding new team member or platform*
        - Client software purchase price - *ranges from zero to thousands*
        - Developer (re-)training
    - **Per developer/seat ongoing charges**
        - Client software annual renewal/maintainace fees - *ranges from zero to thousands*
        - Any client usage fees

    - Developer re-training after any major updates to the VCS or the procedures
- **Available platforms (Server)** - which platform(s) can act as your server if one is required *personally when I am informed that the server only runs on only **one** specific OS, sometimes to a specific service pack, or hardware platform it rings alarm bells*
- **Available platforms (Client(s))** - which platform or platforms can your developers use? Are you limited to a single client? *Again a personal preference is for multiple clients on multiple platforms*
- **Availability**
- Possible Offline Use
-

Setup of VCS Server

This will be specific to the VCS selected above.

Setup of VCS Client(s)

This will be specific to the VCS clients that match the server above but has some common points exist:

What is version control and why use it?

A version control system allows a developer or development team access to essentially I time machine. If the source code, settings, etc., that were used to build a program or system are under version control then the developers can step back in time to recover lost functionality, trace how errors were introduced, support users of older versions of the software who, for one reason or another are not *yet* ready or able to upgrade to the latest version.

The oldest instances of version control were taking a snapshot of the source code at a specific point in time and putting them in a safe place with a label on, in some cases these were literally draws full of punch cards.

Currently there are numerous version control programs which simplify this task which fall into a number of distinct categories:

- Simple backup solutions *not really version control but sometimes claimed as such*
- Local change traking inside files *such as in MS-Word*
- Locking Centralised Version control systems *such as SCCS, CVS, Perforce, etc.*
- Permissive Centralised Version control systems *such as SVN*
- Distributed Version Control, *such as git & mercurial*

All version control systems should let you:

- Revert the source code to a given point in time == infinite undo
- Track changes that have been posted to the system
- Upgrade to a later version of the tool without loosing any information

Most version control systems also include:

- Comparison of changes
- Trace who made a given change
- Permissions on who can make changes
- Integration with bug tracking
- Off-machine, Off-site or online backup of all of your committed work *which gives you a warm feeling* and in the case of Distributed VCS tools every developer has a complete back-up of the entire history of the project.

Many version control tools include, *or provide mechanisms for,* additional features such as:

- Blame *who last changed which line at which revision*
- Create, Distribute & Apply patches *just send out your changes*
- Collaboration via email
- Automatic checking of incoming files/changes to ensure that they meet given rules
- Code/Change review &/or approval
- Automatic insertion of version status into files *for embedding in your program*
- Experimental, Feature Specific or Customer specific versions of the code *usually called branches*
- Working with Continuous Integration &/or Continuous Test tools
- Release &/or Deployment mechanisms
- Release Note generation
- Temporary storage of changes you have not quite finished while you switch to another version/branch to do something else.
- Binary search to allow location of which version introduced a specific bug

To conclude this section most professional or serious code **developers** working on anything but the most trivial code fall in love with version control once they have got into the habit of using it. Being able to say, even years later, what **exactly** changed between version 1.3.1 and version 1.3.1.1 can be a great help. People such as **auditors** love good version control systems & practices as it gives trace-ability when integrated with an issue tracking system - being able to say "**this** issue was fixed by **these** changes produced by **this** person" or "*this* change was made on **this** date to address **this** issue" makes them a lot happier. Having, possibly multiple, off machine &/or off site backups of all of your work means that if your machine breaks or your office is destroyed you can resume work in short order. If you are working on open source projects the VCS is likely to be your main **collaboration** & **review** tool, trying to work without one will get you not taken seriously by your **peers**, (github, bitbucket, etc. are built around version control tools). The **users** of your final program will probably not notice that you are using a version control tool but they will love it when you are able to say "that problem was fixed in version X.Y.Z if you upgrade to that you should see the problem go away".

Read Getting started with version-control online: https://riptutorial.com/version-control/topic/9061/getting-started-with-version-control

Credits

S. No	Chapters	Contributors
1	Getting started with version-control	Community, Steve Barnes