Software Engineering Assignment

Version Control Tools

As a system admin, the chances are you collaborate with multiple people across the company, therefore you will probably know the stress of constantly transferring files and version controlling the changes. Version control tools are a great way to enable collaboration, maintain versions, and track changes across the team.

Perhaps the greatest benefit of using version control tools is that you have the capacity to deal with an unlimited number of people, working on the same code base, without having to make sure that files are delivered back and forth.

Some of the most popular Version Control Tools:

SVN

- SVN, or Subversion as it is sometimes called, is generally the version control system that has the widest adoption.
- Most forms of open-source projects will use Subversion because many other large products such as Ruby, Python Apache, and more use it too. Google Code even uses SVN as a way of exclusively distributing code.
- Because it is so popular, many different clients for Subversion are available. If you use Windows, then Tortoise SVN may be a great browser for editing, viewing and modifying Subversion code bases. If you're using a MAC, however, then Versions could be your ideal client.
- In SVN When we "*check in*" or "*commit*" to svn, we are contacting a central repository, synchronizing (merging & resolving) with the centralized versions, creating a changelist, and then sending that changelist back to the centralized repository.
- All users use and share the same repository, possibly with branching. If two users want to share code, the only way they can do so is by checking into the repository and then each doing a sync.
- An SVN workflow looks like this:
    1. The trunk directory represents the latest stable release of a project.
    2. Active feature work is developed within subdirectories under branches.

3. When a feature is finished, the feature directory is merged into trunk and removed.

Mercurial

- Mercurial is a very fast and efficient application. The creators designed the software with performance as the core feature.
- It was designed initially as a source for larger development programs, often outside of the scope of most system admins, independent web developers and designers. However, this doesn't mean that smaller teams and individuals can't use it.
- Aside from being very scalable, and incredibly fast, Mercurial is a far simpler system to use than things such as Git, which one of the reasons why certain system admins and developers use it.
- Mercurial "*branches*" are more like labels, or tags, which is why we can't delete them, or rename them; they are stored forever in posterity just like the commit message. The only way to achieve this is to learn and enable the patch-queuing system and use the *hg strip* command, or install the local-branches extension. But again they are put to *.hg/strip-backup* directory but are never deleted by Mercurial.

Bazaar

- Bazaar is distributed version control system, which also provides a great, friendly user experience.
- Bazaar is unique that it can be deployed either with a central code base or as a distributed code base.
- It is the most versatile version control system that supports various different forms of workflow, from centralized to decentralized, and with a number of different variations acknowledged throughout.
- One of the greatest features of Bazaar is that you can access a very detailed level of control in its setup. Bazaar can be used to fit in with almost any scenario and this is incredibly useful for most projects and admins because it is so easy to adapt and deal with.

- It can also be easily embedded into projects that already exist. At the same time, Bazaar boasts a large community that helps with the maintenance of third-party tools and plugins.

Version Control Tool To be Used In the Project:

Git

- Git is considered to be a newer, and faster emerging star when it comes to version control systems.
- Here, there is no singular centralized code base that the code can be pulled from, and different branches are responsible for hosting different areas of the code. Other version control systems, such as CVS and SVN, use a centralized control, so that only one master copy of software is used.
- Many system administrators and open-source projects use Git to power their repositories. However it is worth noting that Git is not as easy to learn as SVN or CVS is, which means that beginners may need to steer clear if they're not willing to invest time to learn the tool.
- In Git when we check out a repository, we are getting a complete clone of the whole thing -- all revisions of all files, all the metadata, everything.
- We make changes locally, check in (*commit*) to our local copy of the repository, and then "*push*" those changes to another repository when we want to publish them or share them with other users.
- We can also "*pull*" to synchronize with a remote repository.
- A Git workflow looks like this:
    1. A Git repository stores the full history of all of its branches and tags within the *.git* directory.
    2. The latest stable release is contained within the *master* branch.
    3. Active feature work is developed in separate branches.
    4. When a feature is finished, the feature branch is merged into *master* and deleted.

Submitted By:

Aiman Abdullah Anees [15IT106]