Script

{{{ Show the first slide containing title, name of the production team along with the logo of MHRD}}}

Hello friends and welcome to the tutorial on 'Version Control with Hg'

{{{Show the slide containing the objectives}}}

At the end of this tutorial you will be able to

1. Understand what is Version Control and the need for it.
2. Create and use repository on a daily basis

First let's understand what Version Control is

{{{Show the slide 'what is version control'}}}

Version control is just a way to track your files over time and share them. This allows you to go back to older versions when something goes wrong, see what changed when and why, collaborate on a single piece of work with a bunch of people.

Version control is just a way of backing up your files, before making changes to it. Most people would have cooked up their own version control system, without realizing there are tools built by others which takes the task much more organized and systematic.

{{{Show the slide 'Home-brewed'}}}

Let's look at an example of home-brew Version Control system

Version control is just a way of backing up your files, before making changes to it. Most people would have cooked up their own version control system, without realizing there are tools built by others which takes the task much more organized and systematic.

{{{Show the slide 'Problems'}}}

Let's look at the various problems associated with this setup.

Now let's move onto identifying the needs for a Version Control System.

{{{Show the slide 'The need for Version Control'}}}

1. To err is Human...
2. By tracking the history of the project, an outsider can see the evolution of a project.
3. Allows for effective collaboration on the project as everything is shared.
4. Helps to identify which additions have broken down the project and thus aids in efficient tracking down of the bugs.
5. It is good for a one man show as it is for a big group of people working on a project.

It is similar to playing an Video game.

1. We play games ins stages
2. Once we finish a stage or a task - we SAVE
3. We continue playing
4. But, if necessary, we could choose from one of the saved states and start from there
5. We could alter the course of the game

{{{Show the slide 'Mercurial or hg'}}}
Some of the Version Control tool available and used widely are:

1. cvs(Concurrent Version Systems)
2. svn(subversion)
3. hg(mercurial)
4. git

Each of these tools have their own merits and demerits. In this tutorial we will be learning to use mercurial or hg.

Let's now get into Installation
sudo apt-get install mercurial
For Windows,
http://mercurial.selenic.com/downloads/
Type 'hg' which lists out all the commands
$hg
and 'hg version' which gives the version number.
$hg version

Now why exactly is a repo? A repp/repository is a folder with all your files and a store of all the changes that were made to it. To save disk space, hg doesn't save all files, but only saves only a series of changes made to the files.

{{{Show the slide for 'We need a repo!'}}}
Let's now see how to initialize a repo
cd working-directory/
$hg init
ls -a

The .hg directory indicates that our book directory is now a hg repository. Mercurial keeps all the history of the changes made and a few other config files etc. in this directory.

$hg status
Gives the status of our repo. As a beginner, use it often.
$hg help 'status'

You can use 'hg help commandname' which gives the details about the command. For example.

hg help status
{{{Show the slides for 'Status Codes'}}}

Have a look at what various status codes associated with files means. By looking at the codes, it is clear that our files are not yet being tracked by hg. Now Let's move onto Adding Files.

$hg status

This shows that none of the files in the folder have not been added yet.

$hg add

This simply adds all the files in the (working) directory, to the repository, As expected, the status command shows an A has been appeneded to the filenames. We could also specify files individually, for example

$ hg add filename

We have added a set of files to the repository, but we haven't told mercurial to remember these changes. Now let's take a snapshot of this working directory. This can be done by using commit command.

$hg commit -u "Primal Papppachan <primal007@gmail.com>" -m "Initial Commit."

The -u parameter allows to specify the user details. The parameter -m is used to attach a commit message which gives a description of the changes committed to the repository. Check the status of repository by typing

$ hg st

To see the history of changes made to our repository, we use hg log. We can view the change that we just made to our repoistory.

{{{Show the slide 'Thumbnail views'}}}

hg log gives the log of the changes made in the form of changesets. A changeset is a set of changes made to the repository between two consecutive commits. It also shows the date at which the commit was made.

User information is set in the hgrc file. It can be either globally or locally to the project.

For linux systems

cat ~8.hgrc [ui] username = Primal Pappachan <primal007@gmail.com> editor = vim

We have now set the username details for mercurial to use.

{{{Show the slide 'Advice: commits, messages'}}}

1. Atomic changes; one change with one commit
2. Single line summary — 60 to 65 characters long
3. Followed by paragraphs of detailed description - Why the change? - What does it effect? - Known bugs/issues? - etc.

{{{Show the 'summary' slide'}}}

This brings us to the end of the tutorial. In this tutorial, we have learnt to,

{{{Show self assessment questions slide}}}

Here are some self assessment questions for you to solve

{{{Show the solutions slide to self assessment questions }}}

And the answers,

{{{Show the thank you slide}}}

Hope you have enjoyed this tutorial and found it useful. Thank you