

Experiment 8: Change specification and use any SCM Tool to make different versions

Learning Objective: Students will be able to use

Tools: Git Bash, GitHub

Theory:

Software configuration management: The traditional software configuration management (SCM) process is looked upon by practitioners as the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

The SCM process further defines the need to trace changes, and the ability to verify that the final delivered software has all of the planned enhancements that are supposed to be included in the release. It identifies four procedures that must be defined for each software project to ensure that a sound SCM process is implemented. They are:

1. Configuration identification
2. Configuration control
3. Configuration status accounting
4. Configuration audits

These terms and definitions change from standard to standard, but are essentially the same.

- Configuration identification is the process of identifying the attributes that define every aspect of a configuration item. A configuration item is a product (hardware and/or software) that has an end-user purpose. These attributes are recorded in configuration documentation and baselined. Baselining an attribute forces formal configuration change control processes to be affected in the event that these attributes are changed.
- Configuration change control is a set of processes and approval stages required to change a configuration item's attributes and to re-baseline them.
- Configuration status accounting is the ability to record and report on the configuration baselines associated with each configuration item at any moment of time.
- Configuration audits are broken into functional and physical configuration audits. They occur either at delivery or at the moment of effecting the change. A functional configuration audit ensures that functional and performance attributes of a configuration item are achieved, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation.

GitHub offers all of the distribution revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management for every project.

Implementation:

```
MINGW64:/c/Users/sharm/Ashutosh/SE-exp8

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8
$ git init
Initialized empty Git repository in C:/Users/sharm/Ashutosh/SE-exp8/.git/

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git remote add origin https://github.com/Ashutosh-Sharma-14/SE-SCM.git

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$
```

```
MINGW64:/c/Users/sharm/Ashutosh/SE-exp8

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    trial.txt

nothing added to commit but untracked files present (use "git add" to track)

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git add .

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   trial.txt

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ |
```

```
MINGW64:/c/Users/sharm/Ashutosh/SE-exp8

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git commit -m "first commit"
[master (root-commit) 5b96073] first commit
1 file changed, 1 insertion(+)
create mode 100644 trial.txt
```

```
MINGW64:/c/Users/sharm/Ashutosh/SE-exp8

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (master)
$ git branch -M main

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ashutosh-Sharma-14/SE-SCM.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

```
MINGW64:/c/Users/sharm/Ashutosh/SE-exp8

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ vim trial.txt

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ git add .


sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ git commit -m "second commit"
[main 6221945] second commit
 1 file changed, 2 insertions(+), 1 deletion(-)


sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

sharm@LAPTOP-LST66D23 MINGW64 ~/Ashutosh/SE-exp8 (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ashutosh-Sharma-14/SE-SCM.git
 5b96073..6221945  main -> main
branch 'main' set up to track 'origin/main'.
```


first commit

 main

 Ashutosh-Sharma-14 committed 5 minutes ago

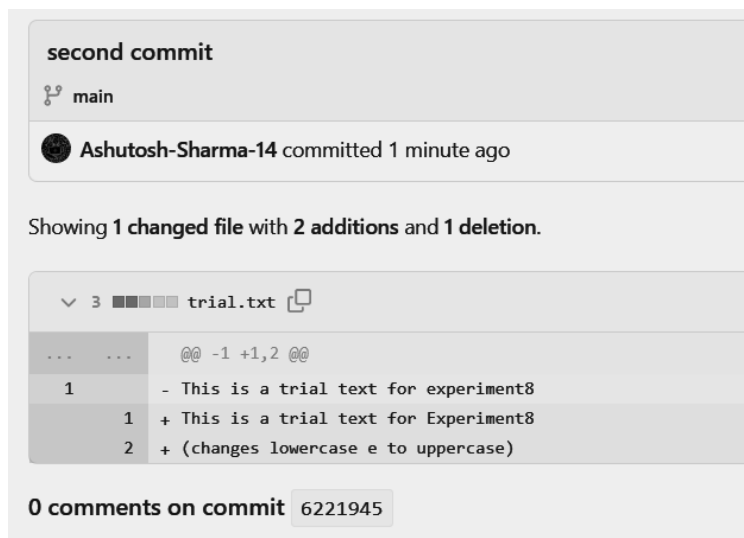
Showing 1 changed file with 1 addition and 0 deletions.

1

 trial.txt
 

...
 ...
 @@ -0,0 +1 @@

1
 + This is a trial text for experiment8



Learning Outcomes: Students should have the ability to

LO1: Understanding of the importance of version control in software development.

LO2: Familiarity with a version control system and how it can be used to manage software versions.

LO3: Understanding use of a version control system in a collaborative development environment.

Outcomes: Upon completion of the course students will be able to use tools like GitHub and git Bash for software configuration management.

Conclusion: - In conclusion, learning how to change specifications and use SCM tools to create different versions is an important skill for any software developer or IT professional. You can effectively manage changes and updates to your projects, keep track of previous versions, and collaborate with other team members more efficiently.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				