AIM: Case Study: GitHub for version Control

THEORY:

Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

<u>Why do we need Configuration management?</u>
The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating.
- It may be a case where multiple version, branches, authors are involved in a software config. project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders

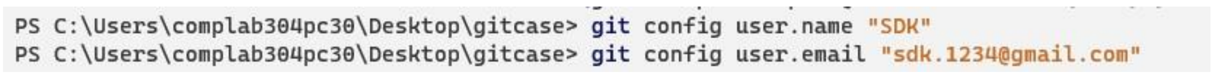
SCM process is also beneficial to control the costs involved in making changes to a system Version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.

<u>Benefits of the Version Control System</u>
The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.
Some key benefits of having a version control system are as follows.

- Complete change history of the file
- Simultaneously working
- Branching and merging
- Traceability

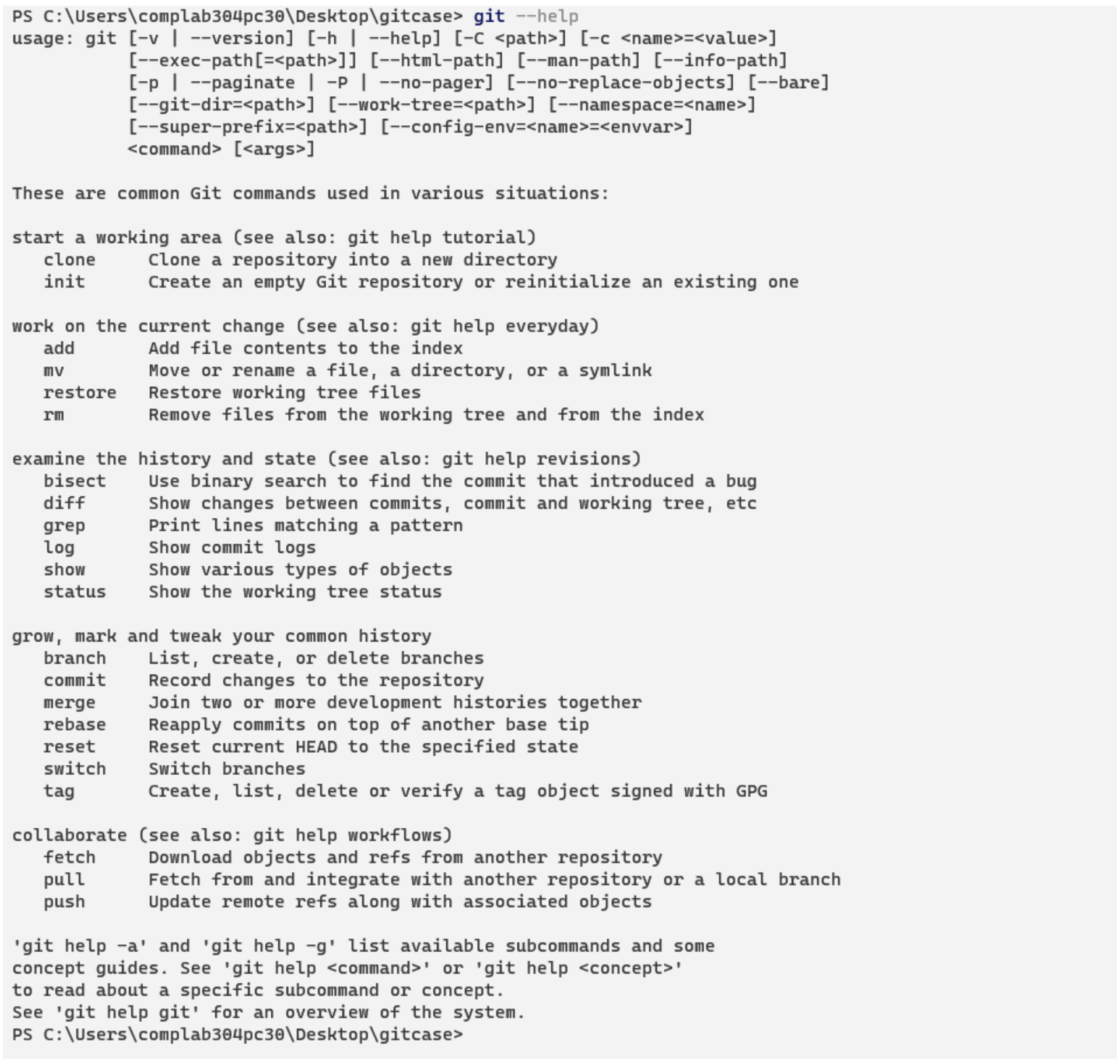
How does version control work?
What is a repository?
You can think of a repository (aka a repo) as a “main folder”, everything associated with a specific project should be kept in a repo for that project. Repos can have folders within them, or just be separate files.
You will have a local copy (on your computer) and an online copy (on GitHub) of all the files in the repository.

1. First of all configure the git username and git email id so that you can perform git commit operations



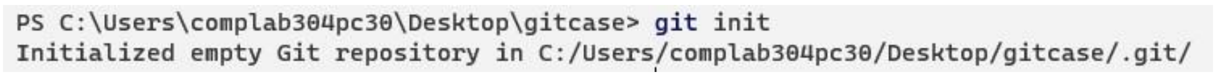
```
PS C:\Users\complab304pc30\Desktop\gitcase> git config user.name "SDK"
PS C:\Users\complab304pc30\Desktop\gitcase> git config user.email "sdk.1234@gmail.com"
```

2. You can view git commands by typing out below command



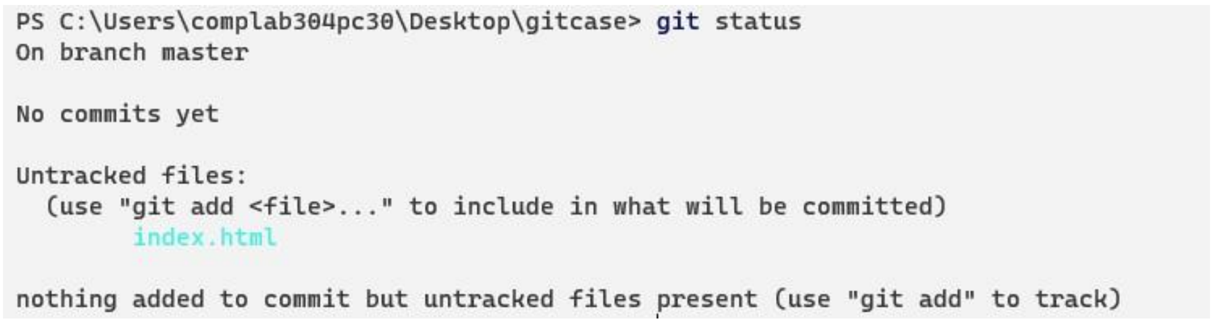
```
PS C:\Users\complab304pc30\Desktop\gitcase> git --help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
PS C:\Users\complab304pc30\Desktop\gitcase>
```

3. To initialise git repository use git init command



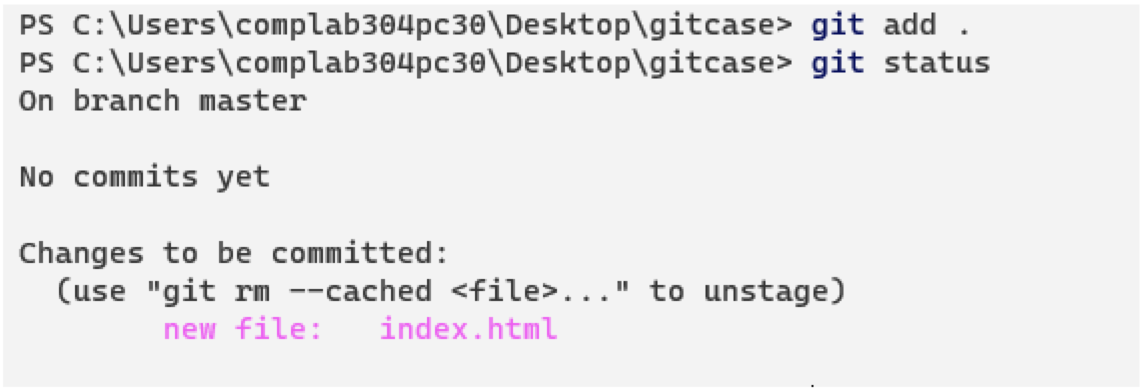
```
PS C:\Users\complab304pc30\Desktop\gitcase> git init
Initialized empty Git repository in C:/Users/complab304pc30/Desktop/gitcase/.git/
```

4. To check status of your local repository use following command



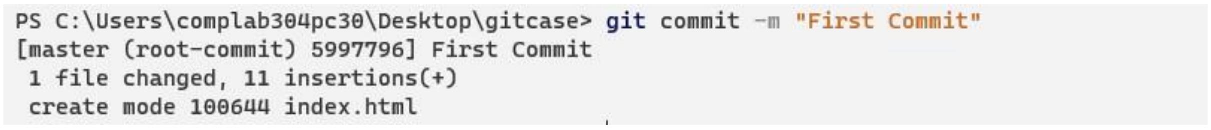
```
PS C:\Users\complab304pc30\Desktop\gitcase> git status
On branch master

No commits yet

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

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

5. Add file to staging area and again check the status



```
PS C:\Users\complab304pc30\Desktop\gitcase> git add .
PS C:\Users\complab304pc30\Desktop\gitcase> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

6. Now the file is in the staging area it is ready to be committed to save the history and you can use git log to see the history



```
PS C:\Users\complab304pc30\Desktop\gitcase> git commit -m "First Commit"
[master (root-commit) 5997796] First Commit
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
```