**Aim:** Change specification and Configuration management with Git-Hub Tool/Bazaar Tool.

**Theory:**

**1.What is git?**

Git is a distributed version-control system for tracking changes in source code during software

development. It is designed for coordinating work among programmers, but it can be used to

track changes in any set of files. Its goals include speed, data integrity, and support for

distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel

developers contributing to its initial development

Characteristics

● Strong support for non-linear development

● Distributed development

● Compatibility with existing systems and protocols

● Efficient handling of large projects

● Cryptographic authentication of history

● Toolkit-based design

● Pluggable merge strategies

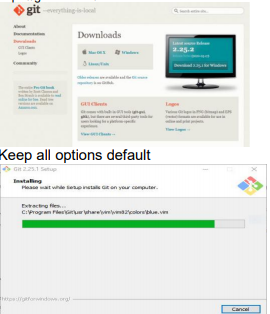
● Garbage accumulates until collected

● Periodic explicit object packing

Steps for using git

1) Installation

a) Download and install git from https://gitscm.com/downloads

****

**2.Define SCM**

System Configuration Management (SCM) is an arrangement of exercises which controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. In this way, SCM is a fundamental piece of all project management activities.

**Change control**

Change control is a systematic approach to managing all changes made to a product or system. The purpose is to ensure that no unnecessary changes are made, that all changes are documented, that services are not unnecessarily disrupted and that resources are used efficiently.

Here's an example of a six-step process for a software change request:

**1. Documenting the change request:**

When the client requests the change, that request is categorized and recorded, along with informal assessments of the importance of that change and the difficulty of implementing it.

**2. Formal assessment:**

The justification for the change and risks and benefits of making/not making the change are evaluated. If the change request is accepted, a development team will be assigned. If the change request is rejected, that fact is documented and communicated to the client.

**3.Planning:**

The team responsible for the change creates a detailed plan for its design and implementation, as well as a plan for rolling back the change should it be deemed unsuccessful.

**4.Designing and testing:**

The team designs the program for the software change and tests it. If the change is deemed successful, the team requests approval and a date for implementation.

**5.Implementation and Review:**

The team implements the program and stakeholders review the change.

**6.Final assessment:**

If the client is satisfied that the change was implemented satisfactorily, the change request is closed. If the client is not satisfied, the project is reassessed and steps may be repeated.

**Version control**

Software Version Control is a system or tool that captures the changes to a source code elements: files, folders, images or binaries. A version control system (also known as a Revision Control System) is a repository of files, often the files for the source code of computer programs, with monitored access. Every change made to the source is tracked, along with who made the change, why they made it, and references to problems fixed, or enhancements introduced, by the change.

Version control systems are essential for any form of distributed, collaborative development. Whether it is the history of a wiki page or large software development project, the ability to track each change as it was made, and to reverse changes when necessary can make all the difference between a well managed and controlled process and an uncontrolled ‘first come, first served’ system. It can also serve as a mechanism for due diligence for software projects.

Combines procedures and tools to manage the different versions of configuration objects created during the software process. Version control systems require the following capabilities.

Project repository – stores all relevant configuration objects.

o Version management capability – stores all versions of a configuration object (enables any version to be built from past versions)

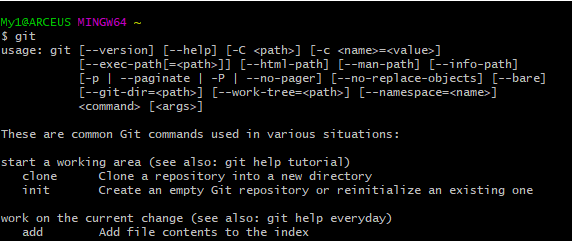
o Make facility – enables collection of all relevant configuration objects and constructs a specific software version.

o Issues (bug) tracking capability – enables team to record and track status of outstanding issues for each configuration object.

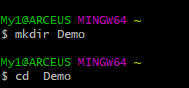
• Uses a system modeling approach (template – includes component hierarchy and component build order, construction rules, verification rules).

Output:

Verify the installation



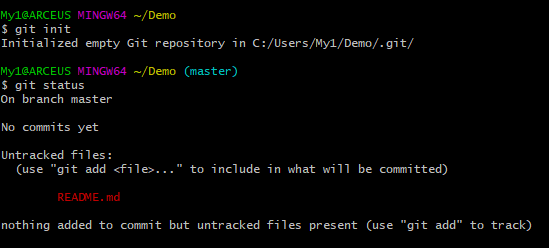
2) Creating an empty folder and initializing git a) Open your terminal and type the following commands mkdir Demo Creates a directory Demo and cd Demo Enter directory Demo



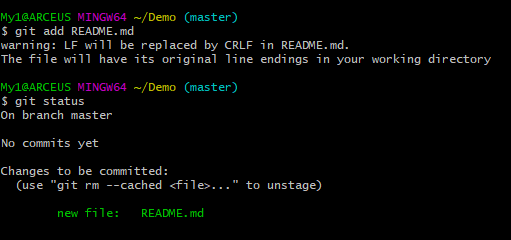
echo "#Demo" >> README.md Writes text “#Demo” to README.md

C:\Users\My1\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3.png

git init Initializes the current folder as a git repository

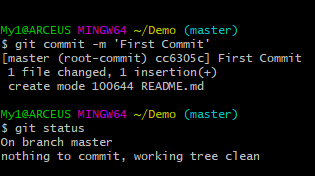


git add README.md Tells the Git program you care about this file and want to track any changes from this point forward also known as staging.



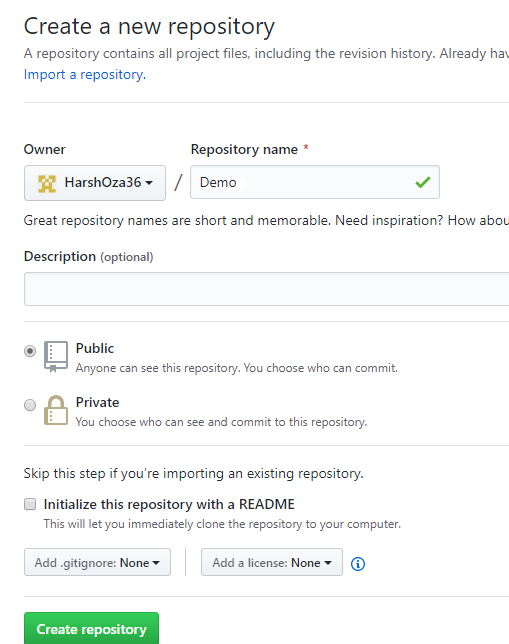
you have added multiple files you can replace README.md with the names of your files alternatively you can add all the files in the folder by using the command git add . You can also ignore certain files (server secrets, passwords, large files, build output files) by creating a . gitignore file and adding the names of the files or directories to it. More at https://guide.freecodecamp.org/git/gitignore/

3) Making a commit a) Commit can be thought of as a milestone. Every time you accomplish some work, you can write a Git commit to store that version of your file, so you can go back later and see what it looked like at that point in time. Whenever you make a change to your file, you create a new version of that file, different from the previous one. To make a commit enter: git commit -m "first commit"



b) By making commits you are now essentially tracking the changes made in your code and saving them as separate versions locally. c) As you go on making changes and commits you can use the g it log command to look at your past commits.

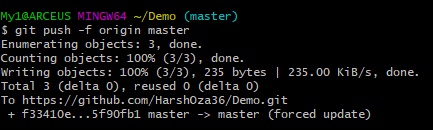
4) Pushing a commit a) Using a git repository hosting service or hosting your own git server (for advanced users), you can upload your repository and its subsequent changes to the cloud. b) Popular Git hosting services: GitHub, GitLab, Bitbucket, etc. c) We’ll be using GitHub: https://github.com d) Create an account on GitHub. e) Create a repository and give it a name (Demo)



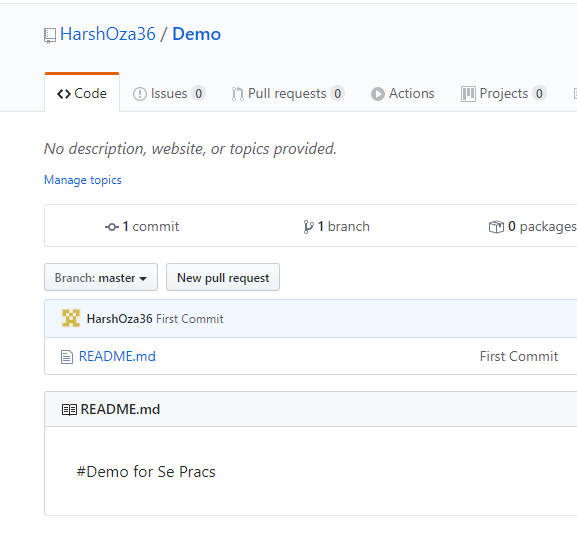
f) After creating the repository, you’ll have to tell the local version of your repository, the location of the remote repository. You can do this by running the command: git remote add origin https://github.com/username/repo.git Eg.

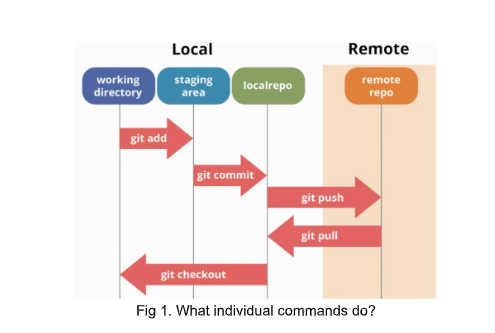
7

g) Now that your local repository knows where to push your date you can run the following command to push: git push -u origin master



h) Once this command is executed successfully you can refresh the page of the repository to find the same files as that of your local repository

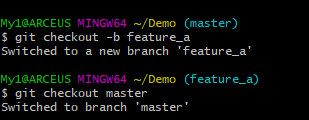




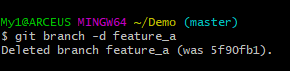
Pulling from remote If there are multiple users working on the same repository, there will be a situation when one user has made changes to his/her code and pushed them. Other users/collaborators of that repository will have to execute the following command to get obtain those changes in their local repository: git pull

11

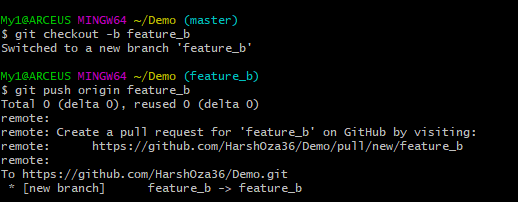
Branching Branches are used to develop features isolated from each other. The master branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion. ● To create a new branch named "feature\_a" and switch to it use: git checkout -b feature\_x and To switch back to master: git checkout master



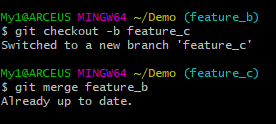
To delete the branch again git branch -d feature\_x



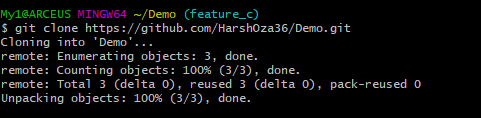
A branch is not available to others unless you push the branch to your remote repository git push origin



Merging To merge another branch into your active branch (e.g. master), use git merge



Using the above command (also while pulling) git tries to auto-merge changes. Unfortunately, this is not always possible and results in conflicts. You are responsible to merge those conflicts manually by editing the files shown by git. After changing, you need to mark them as merged with git add before merging changes, you can also preview them by using git diff More info on branching and merging at https://git-scm.com/book/en/v2/Git-BranchingBasic- B ranching-and-Merging git clone https://linktorepo.git creates a local copy of a project that already exists remotely. The clone includes all the project’s files, history, and branches.



git diff look at the difference between branches or commits All commands can be found at https://git-scm.com/docs

17

git tag version\_name tags the current commit (helpful for release versions).

C:\Users\My1\AppData\Local\Microsoft\Windows\INetCache\Content.Word\18.png

References and in-depth guides

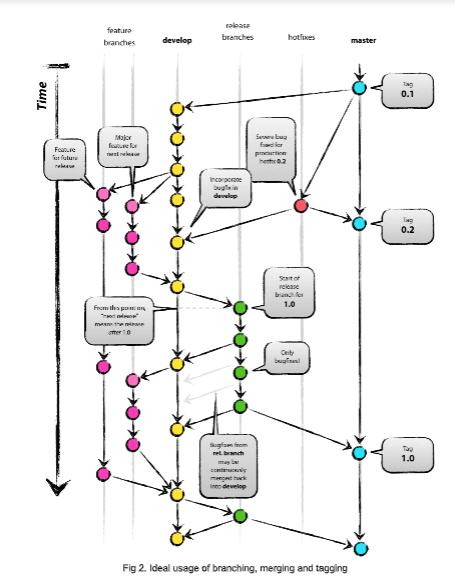
● Official git documentation: https://git-scm.com/docs

● All inclusive quick start guide https://rogerdudler.github.io/git-guide/ https://guides.github.com/introduction/git-handbook

Warnings

● Never commit files that contain secret information (server secrets, passwords,etc). Store them in a separate file and add it to .gitignore

● Some commands can be used with attributes like --force and --hard. By using them, git will not warn or ask for confirmation in case there is a discrepancy. Use these attributes with care. Adding large files to your repositories can slow the push and pull process. Avoid if possible.

****

**CONCLUSION:**

From this experiment we learnt about System Configuration Management or SCM and its tools.So there are various tools like Git, Bazaar but for this experiment we used Git tool. We understood how to install it and start it up. We implemented various git commands and created a demo repository on our Username of Github. So Scm can also be known as Source Code Management with respect to Git.