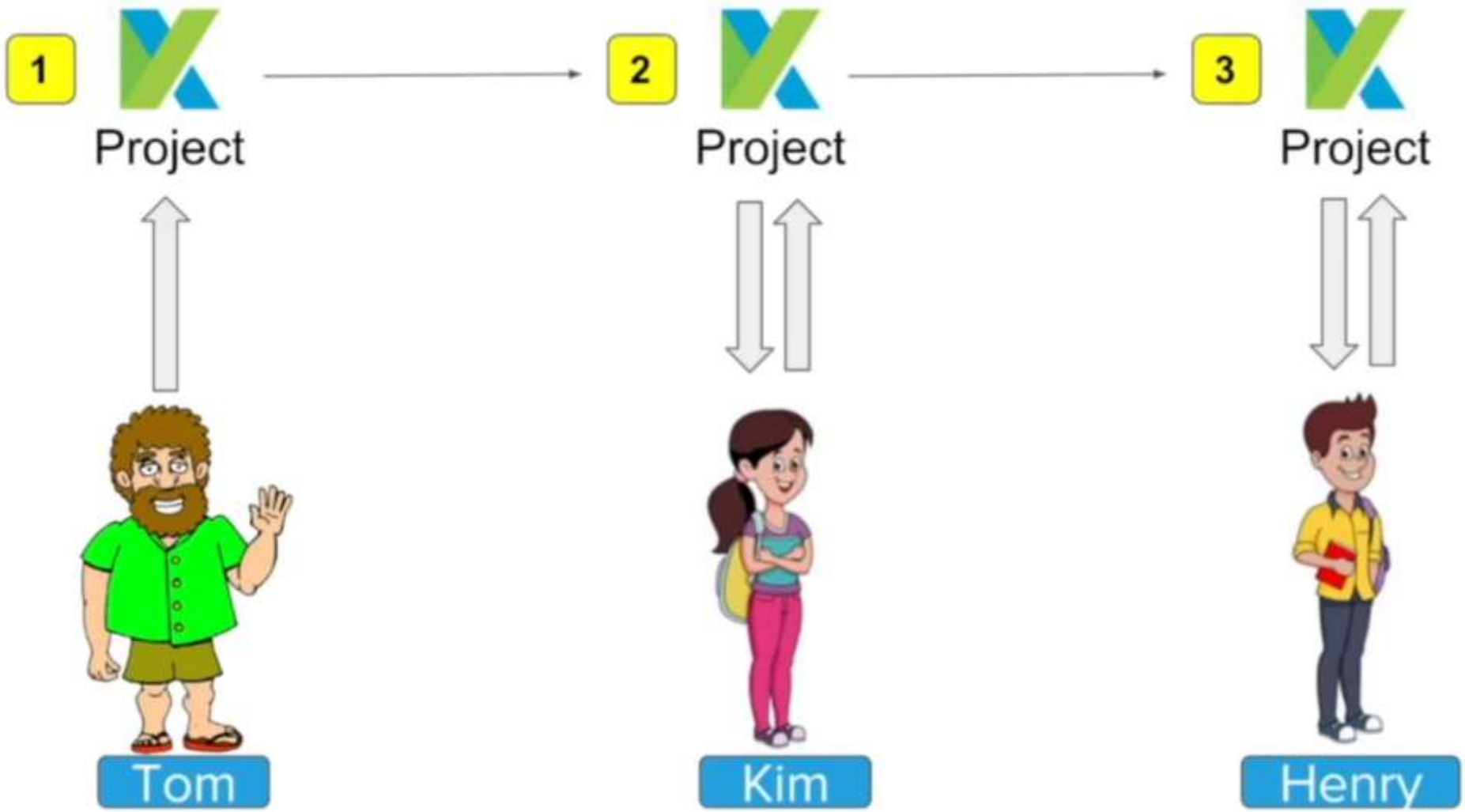
What is Git?

Version Control System

Tom

Kim

Henry


1
Project
2
Project
3
Project
Tom
Kim
Henry

Henry makes his changes and sends his project to Kim

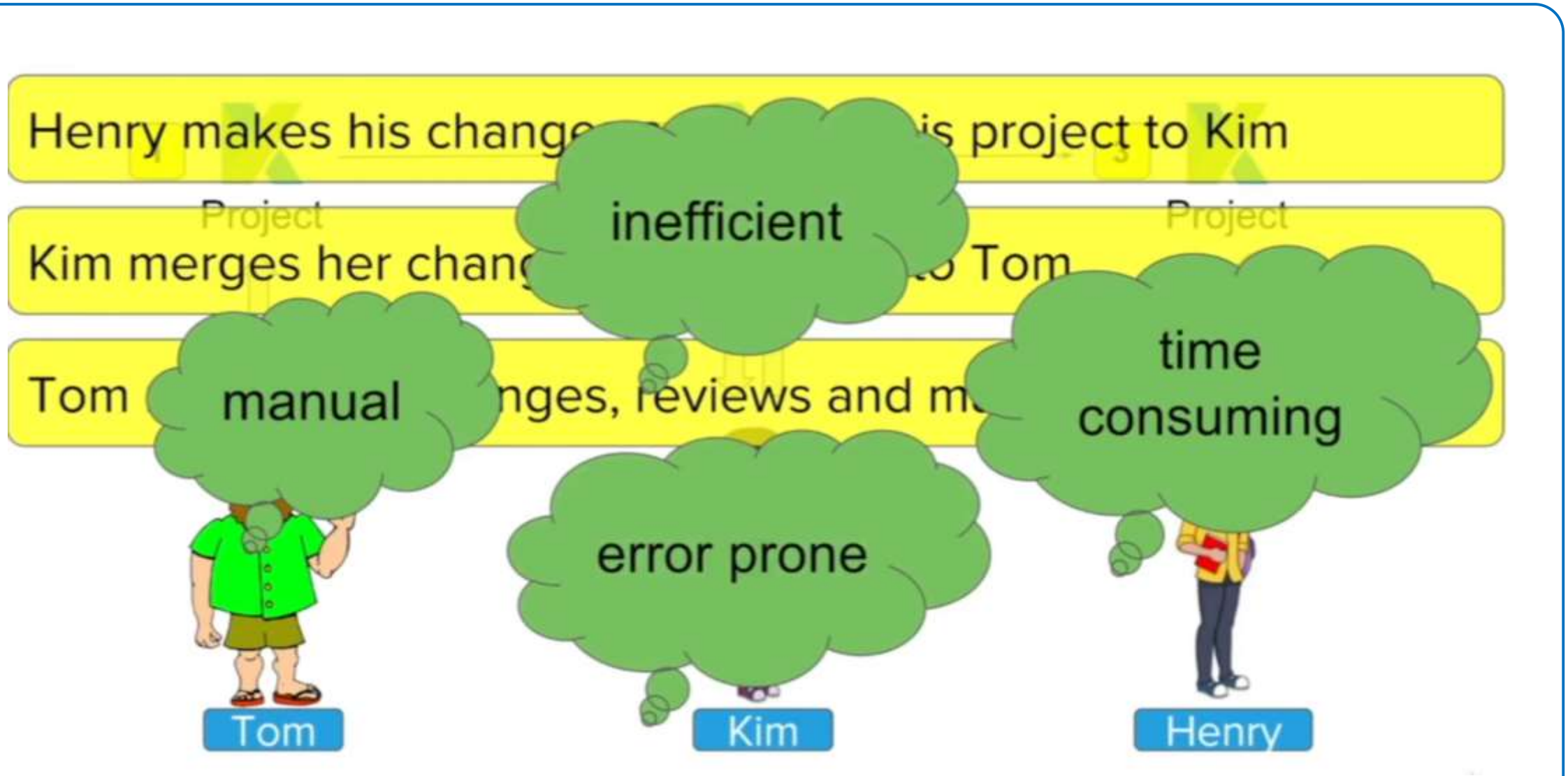
Kim merges her changes and sends it to Tom

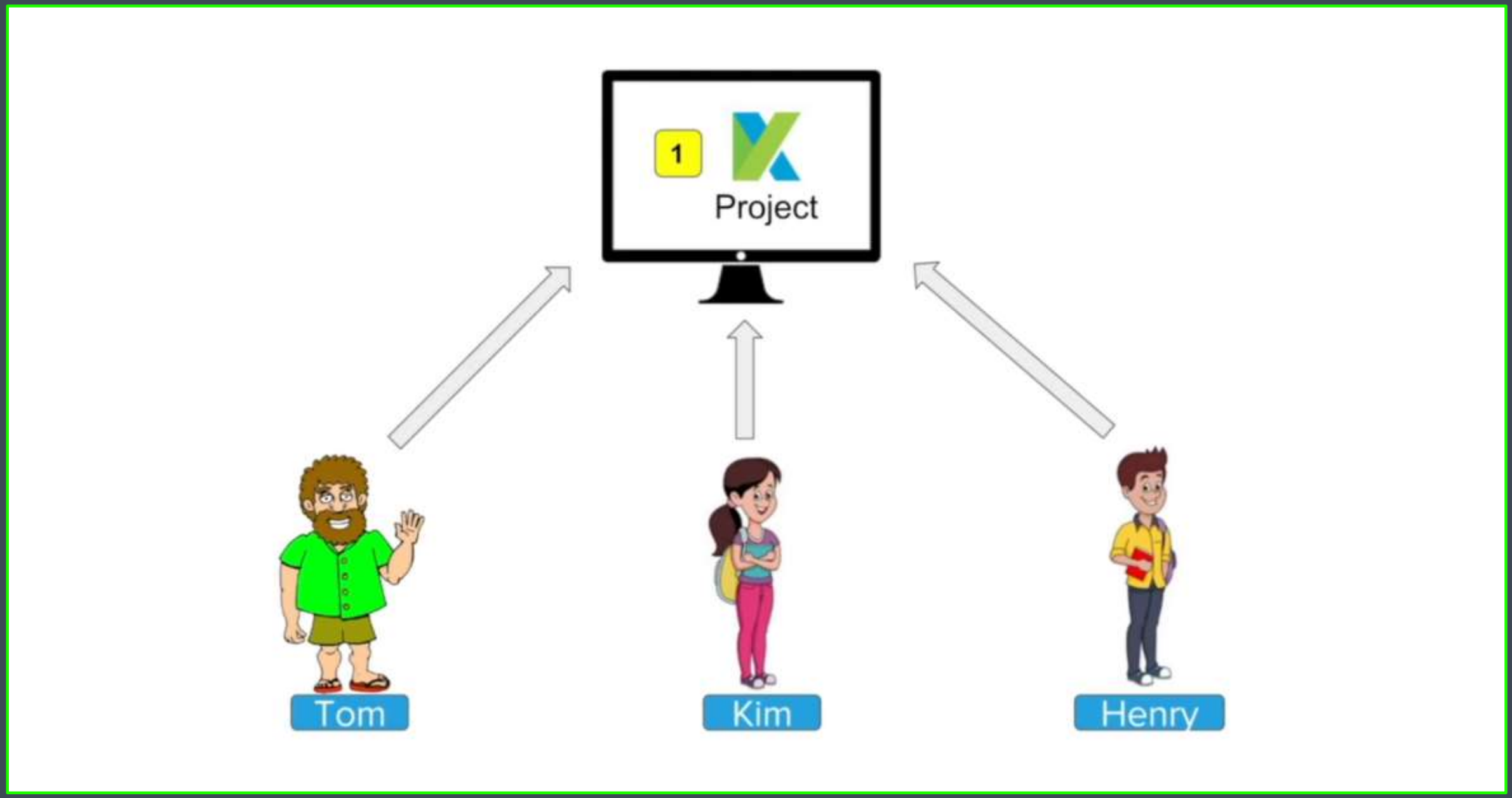
Tom merges his changes, reviews and maintains the final copy

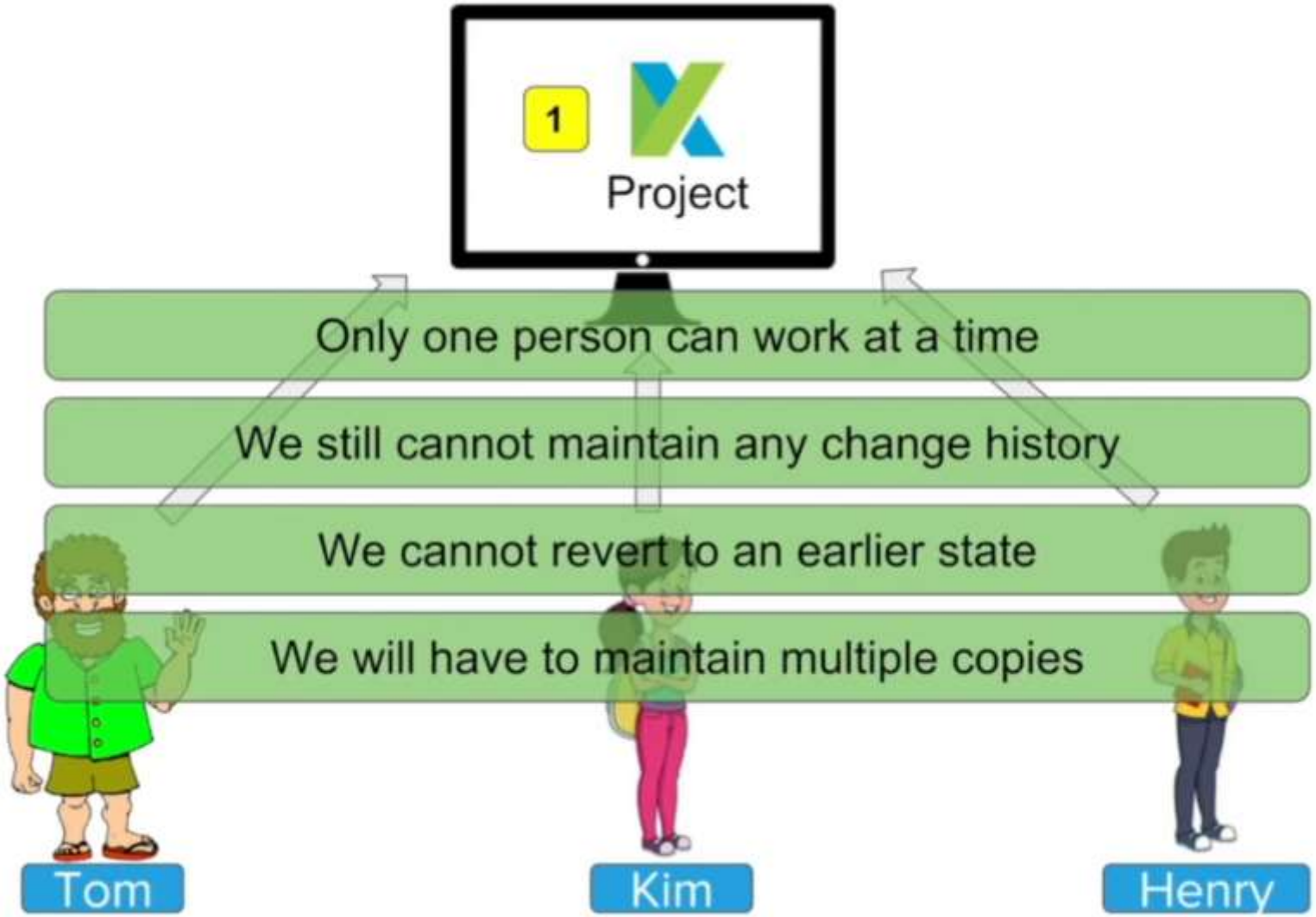
Tom

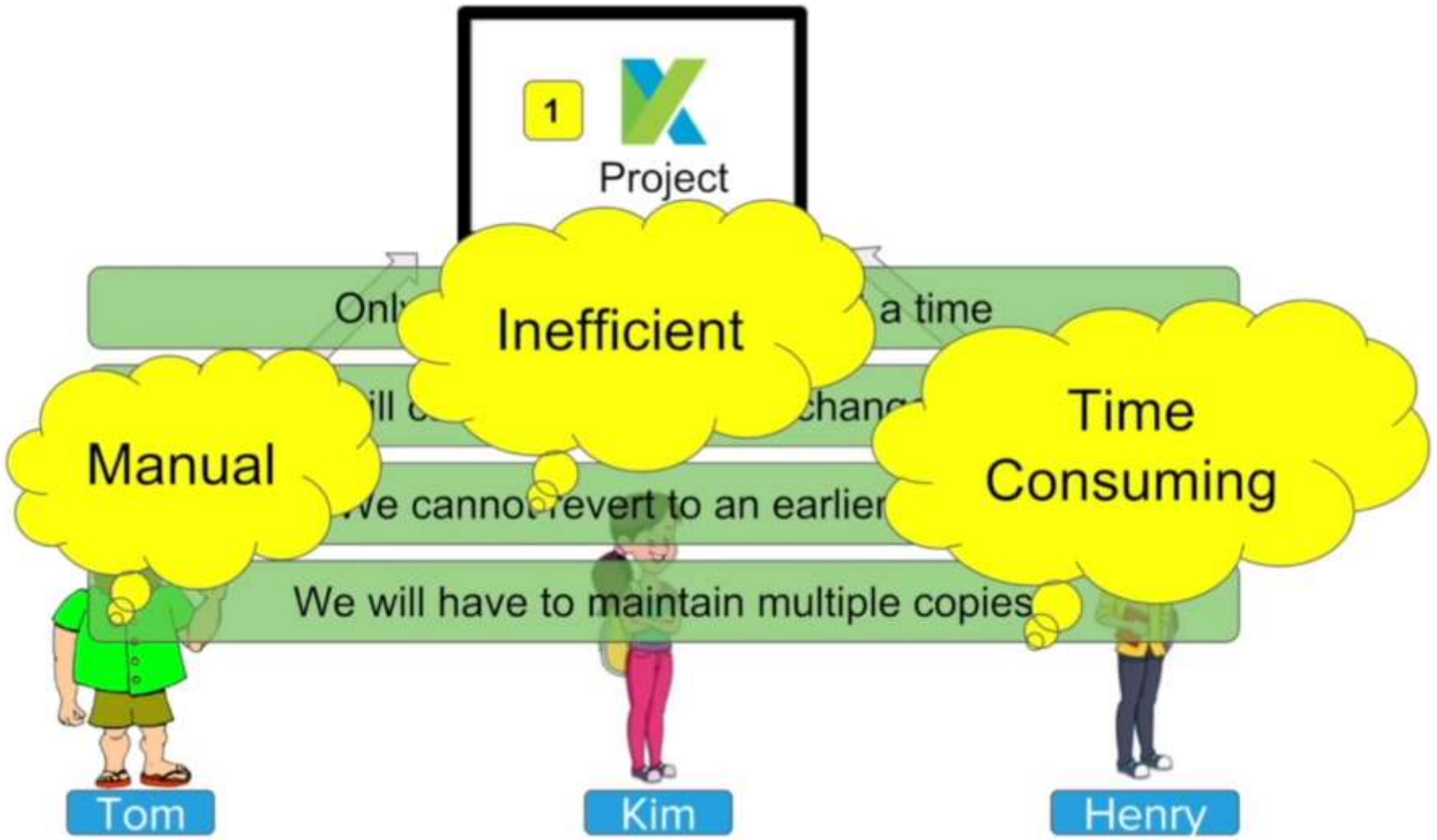
Kim

Henry


Henry makes his change
is project to Kim
Kim merges her chang
to Tom
Tom
nges, reviews and m
inefficient
manual
time
consuming
error prone
Tom
Kim
Henry


1
Project
Tom
Kim
Henry

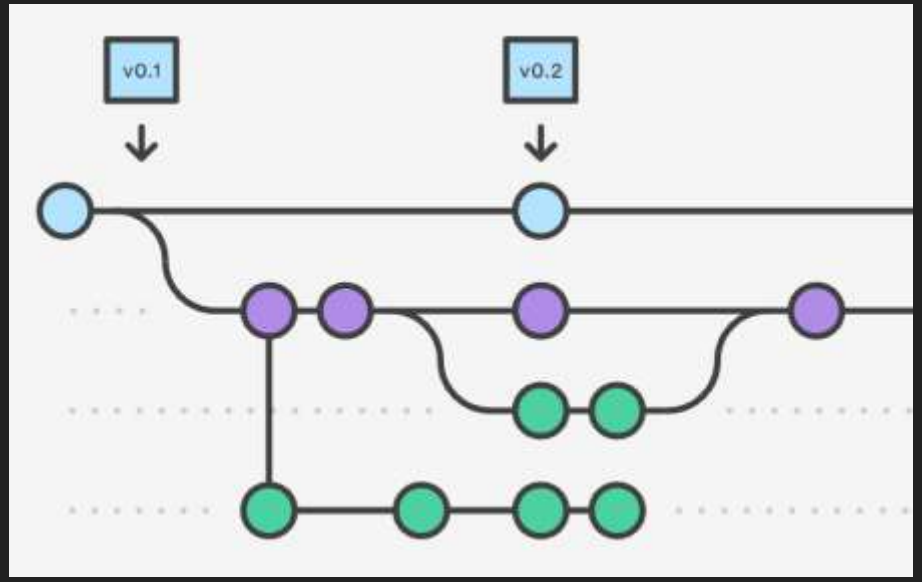

1
Project
Only one person can work at a time
We still cannot maintain any change history
We cannot revert to an earlier state
We will have to maintain multiple copies
Tom
Kim
Henry


1
Project
Inefficient
Manual
Time Consuming
a time
We will have to maintain multiple copies
Tom
Kim
Henry

Version Control System

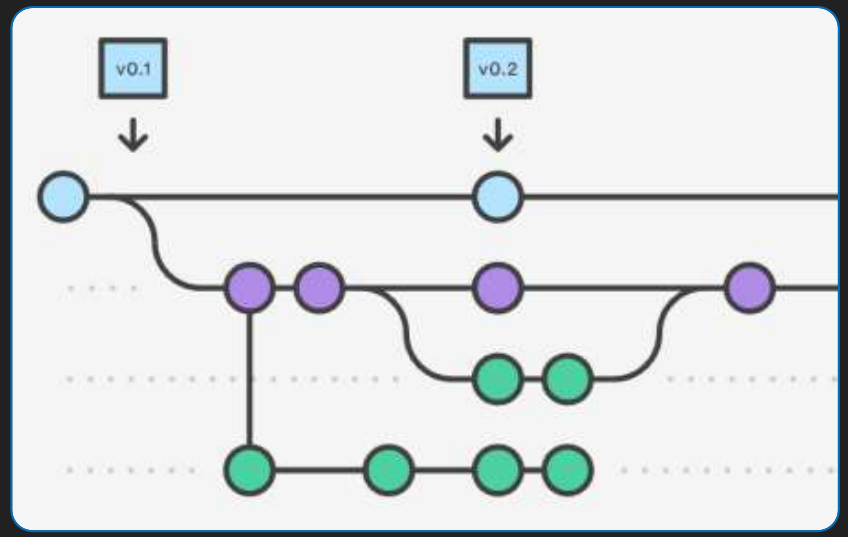
Version control or Source Control

- Source control, or version control, is a way of tracking your files progress over time.
- It is usually saved in a series of snapshots and branches, which you can move back and forth between.



What's a version control system?

- A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together.
- As the project evolves, teams can run tests, fix bugs, and contribute new code with the confidence that any version can be recovered anytime.
- Developers can review project history to find out:
  - What changes were made?
  - Who made the changes?
  - When were the changes made?
  - Why were changes needed?

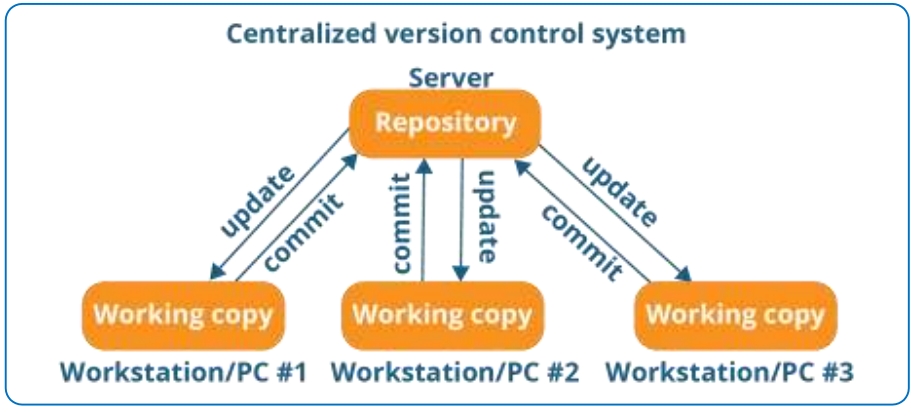


What you can do

- Version control allows you to:
  - Distribute your file changes over time
  - Prevent against data loss/damage by creating backup snapshots
  - Manage complex project structures

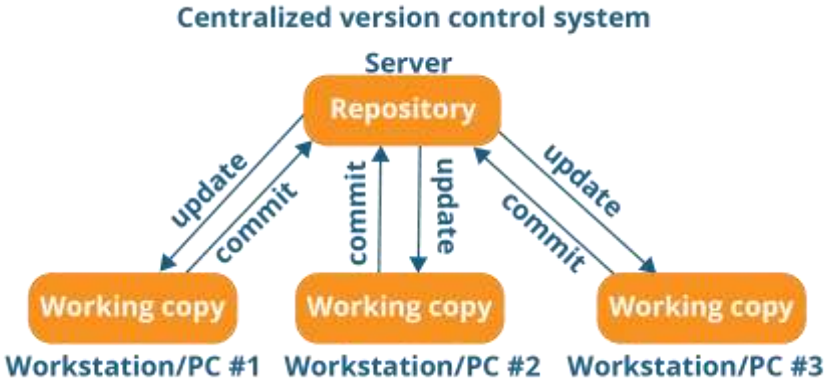
Centralized vs Distributed Version Control Systems

- There are two general varieties of version control: centralized and distributed.
- Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy.
- "Committing" a change simply means recording the change in the central system. Other programmers can then see this change.



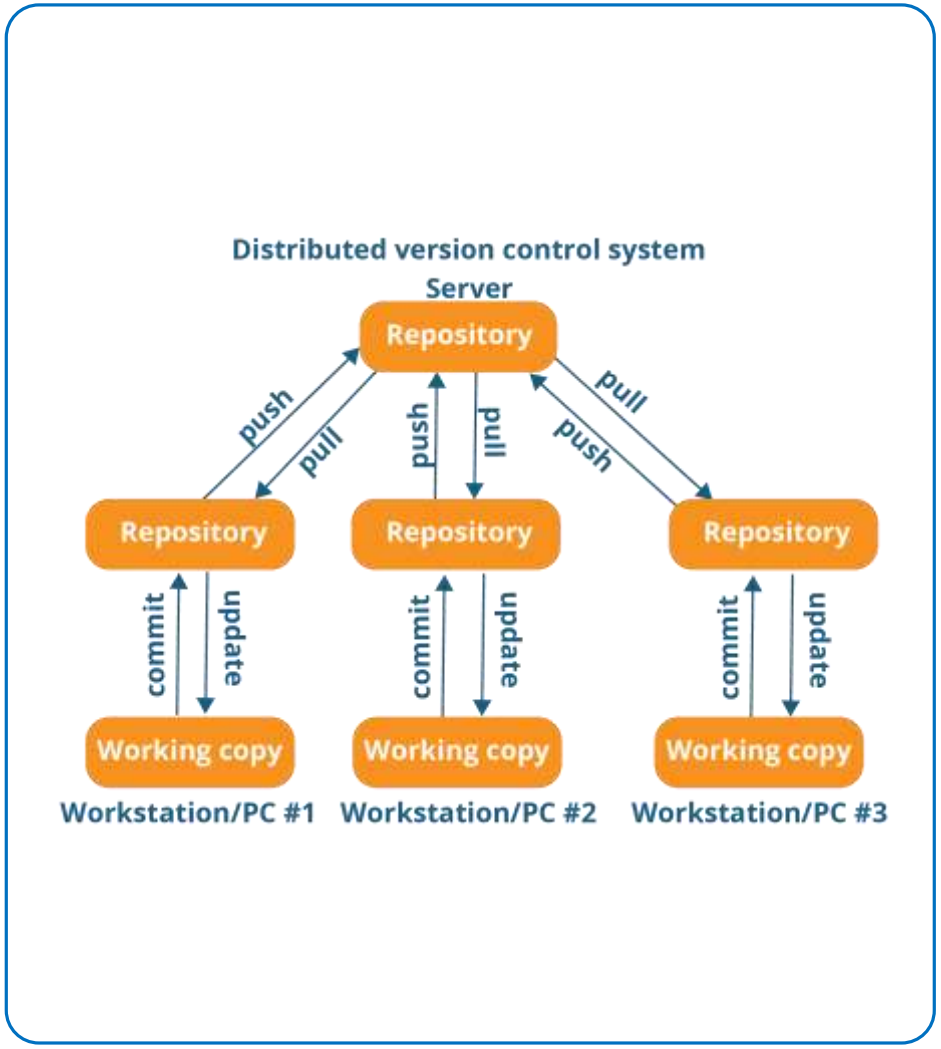
Centralized Version Control System

- They can also pull down the change, and the version control tool will automatically update the contents of any files that were changed.
- Some of the most common centralized version control systems are Subversion (or SVN) and Perforce.
- Main benefits:
  - Centralized systems are typically easier to understand and use
  - You can grant access level control on directory level
  - performs better with binary files



Distributed Version Control System

- In distributed version control, every developer "clones" a copy of a repository and has the full history of the project on their own hard drive. This copy (or "clone") has all of the metadata of the original.
- The act of getting new changes from a repository is usually called "pulling," and the act of moving your own changes to a repository is called "pushing".
- The three most popular distributed version control systems are Git, Mercurial, and Bazaar.
- Main benefits:
  - Performance of distributed systems is better
  - Branching and merging is much easier
  - With a distributed system, you don't need to be connected to the network all the time (complete code repository is stored locally on PC)



About Git

- Git is a Version control software similar to many others out there.
- It follows all the same rules and concepts that any version control follows.

Why use Git?

- The most popular version control software in the world.
- According to the latest Stack Overflow developer survey, more than 70 percent of developers use Git, making it the most-used VCS in the world.
- Lots of documentation and support.
- Lots of integration with other applications (SourceTree, Heroku, GitHub).
- Git is commonly used for both open source and commercial software development, with significant benefits for individuals, teams and businesses.

Git vs GitHub

What GitHub Is

- GitHub is an application that allows you to store remote repositories. (it's in the name!)
- You can interact with your GitHub repository through your local machine's push/pull system.
- GitHub is used primarily to allow other people to add to the project (ex. Open source projects)
- GitHub allows more people than yourself to see and interact with the repository.

The Difference

- Git is a version control software allowing you to take snapshots and distribute your creations & modifications over time.
- GitHub is an application that allows you to store and interact with your repository on a remote server, as well as add more features (e.g., Publicity, licensing, collaborators)
- Git is the bones and flesh of version control, while GitHub gives you the platform to work with your repository more easily.

	
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

Git Workflow

Introduction

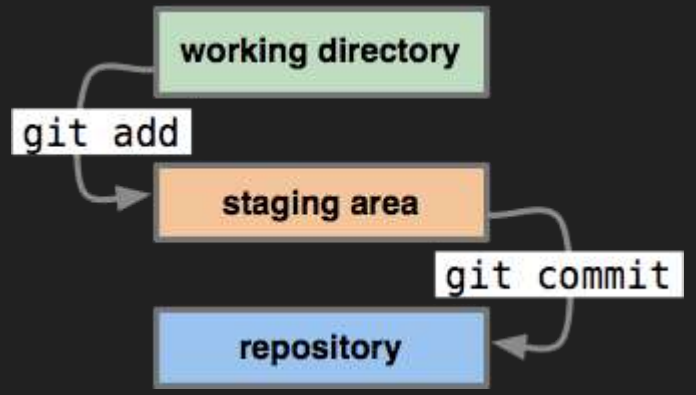
- Before Git tracks a change, it goes through a long chain of operations and tasks.
- Many of these tasks are user controlled, and are required for changes to be tracked correctly.

Repositories

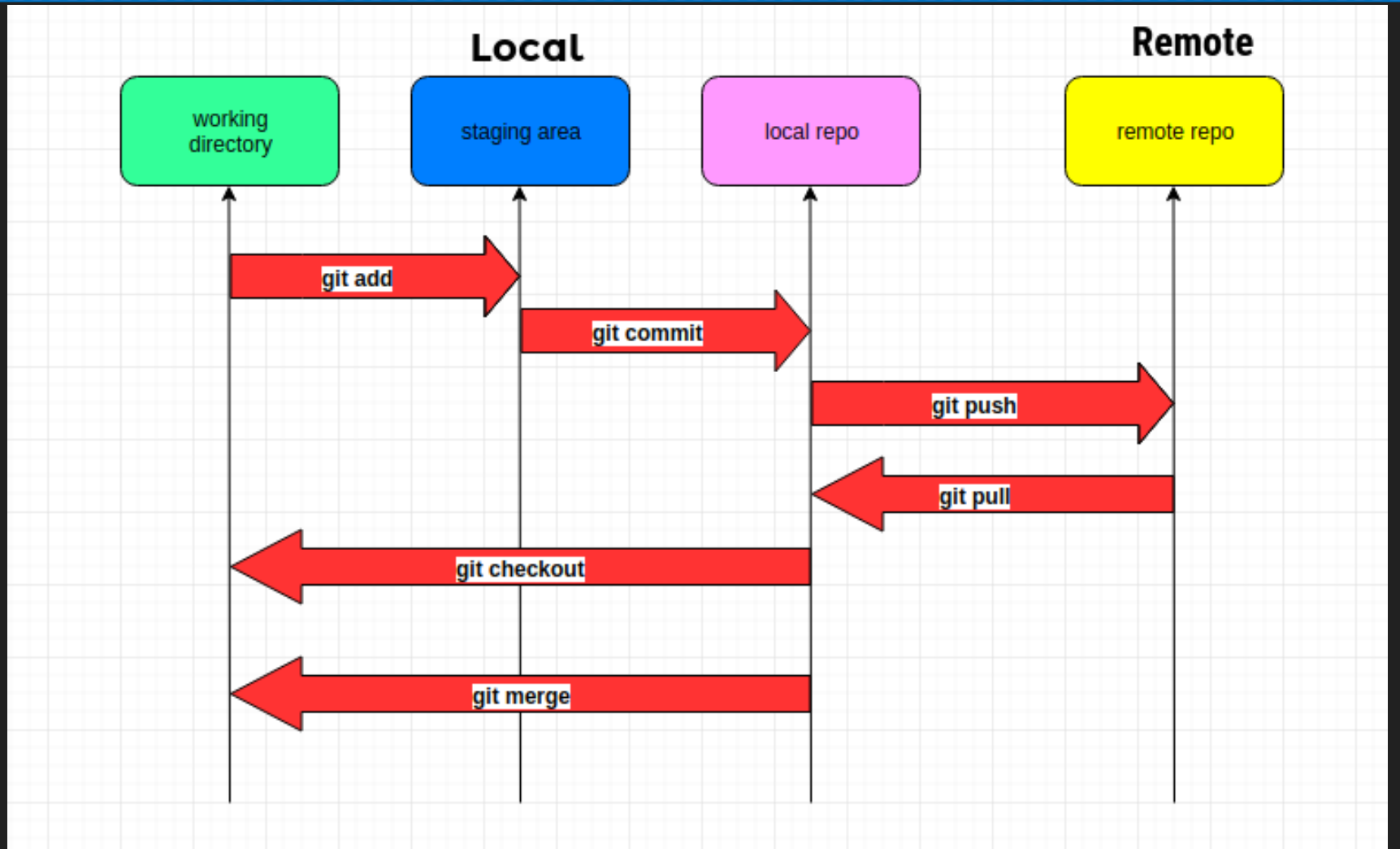
- Repositories, usually called ‘repos,’ store the entire history and source control of a project.
- They can either be hosted locally or on a shared server, such as GitHub.
- Most repositories are stored on GitHub, while core contributors make copies of the repository on their machine and update it using the push/pull system.
- Any repository stored somewhere other than locally is called a ‘remote repository’.

Repos vs Directories

- Repositories are timelines of the entire project, including all.
- Directories, or 'working directories,' are projects in their current state.
- Any local directory interacting with a repository is technically a repository itself. However, it is better to call these directories 'local repositories' as they are instances of a remote repository.

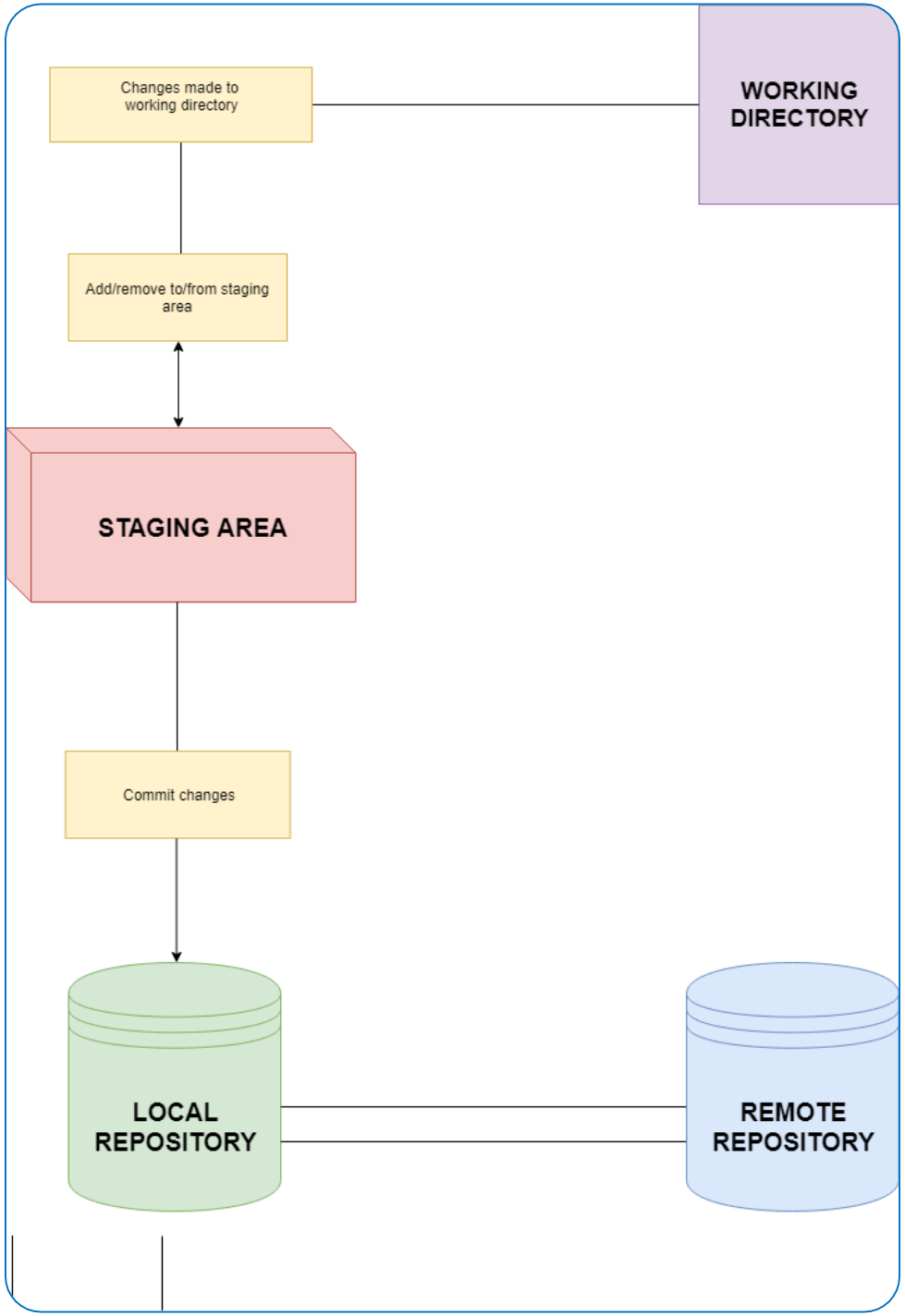


Workflow Diagram



Workflow Diagram

- This diagram shows a little bit about how the basic Git workflow process works
- The staging area is the bundle of all the modifications to the project that will be committed.
- A ‘commit’ is similar to taking a snapshot of the project’s current state and storing it on a timeline.



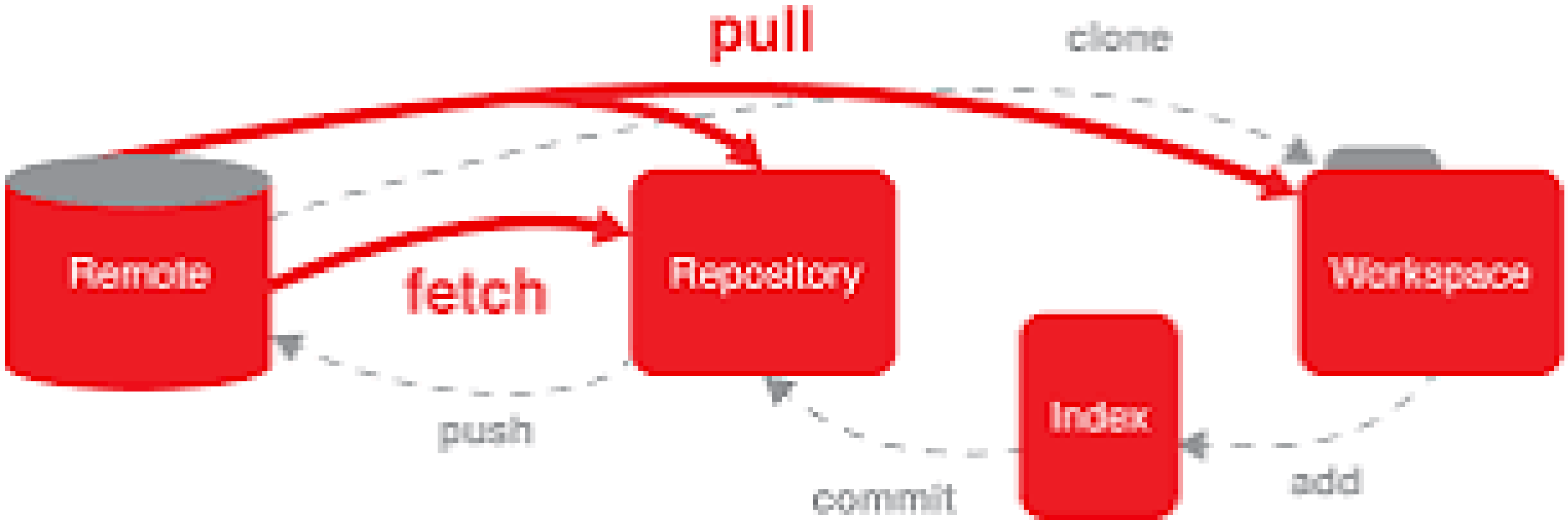
Git Commands

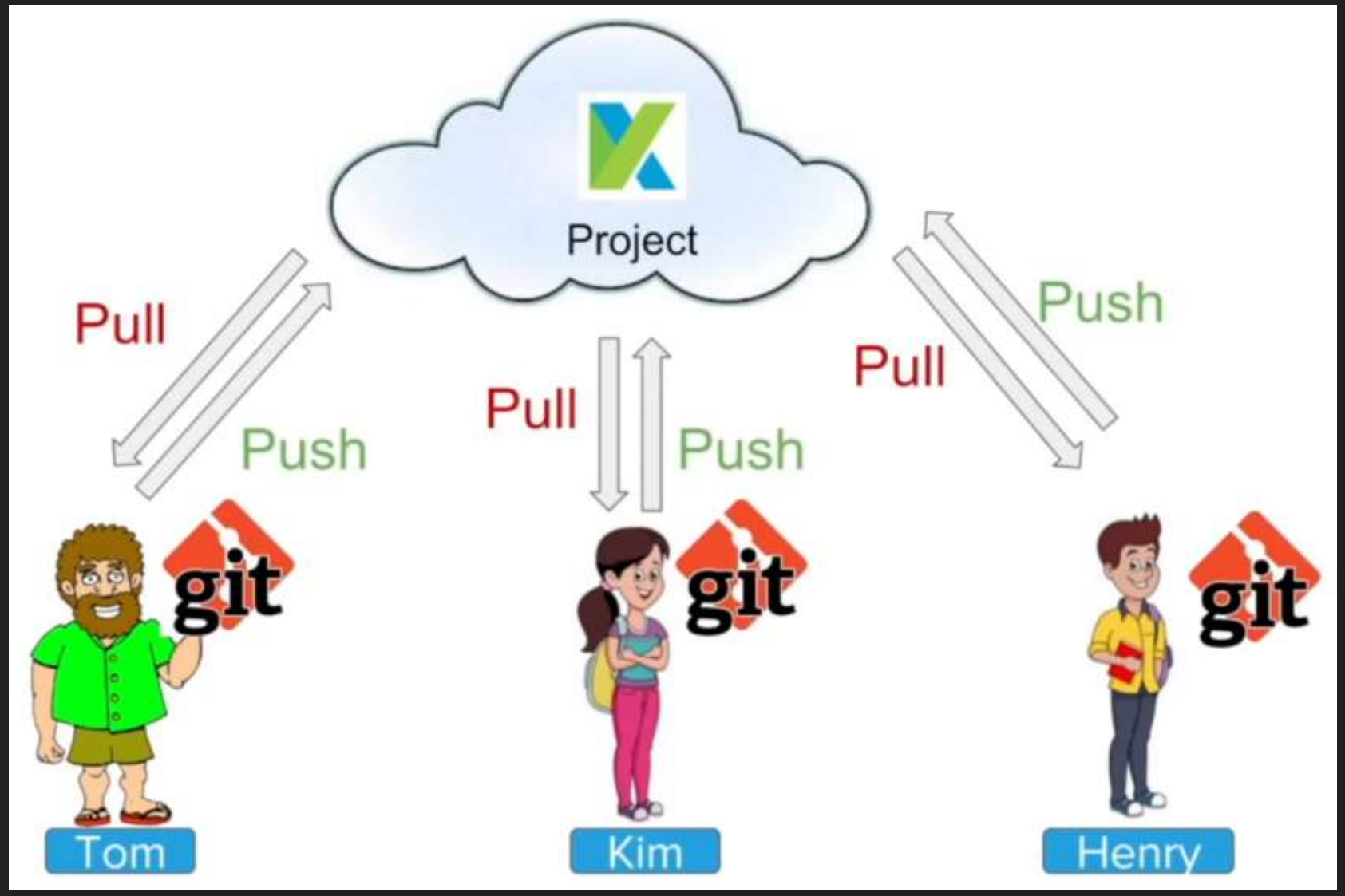
- **git init**: initializes a new Git repository and begins tracking an existing directory.
- **git clone**: creates a local copy of a project that already exists remotely.
- **git add**: performs staging, the first part of that two-step process.
- **git commit**: saves the snapshot to the project history and completes the change-tracking process.

Git Commands

- **git status**: shows the status of changes as untracked, modified, or staged.
- **git branch**: shows the branches being worked on locally.
- **git merge**: merges lines of development together. This command is typically used to combine changes made on two distinct branches.
- **git pull:** updates the local line of development with updates from its remote counterpart.
- **git push:** updates the remote repository with any local commits to a branch.

Command	Description
git fetch	Downloads the latest changes from the remote repository, but does not merge them into your local branch.
git pull	Downloads the latest changes from the remote repository and merges them into your local branch.


pull
clone
Remote
fetch
Repository
Workspace
Index
push
commit
add


Project
Pull
Push
Pull
Push
Pull
Push
git
git
git
Tom
Kim
Henry

Step 1 All team members will install git on their systems

Tom

Kim

Henry

Step 2

Anyone can put a copy of project on remote repository

(like GitHub or BitBucket)

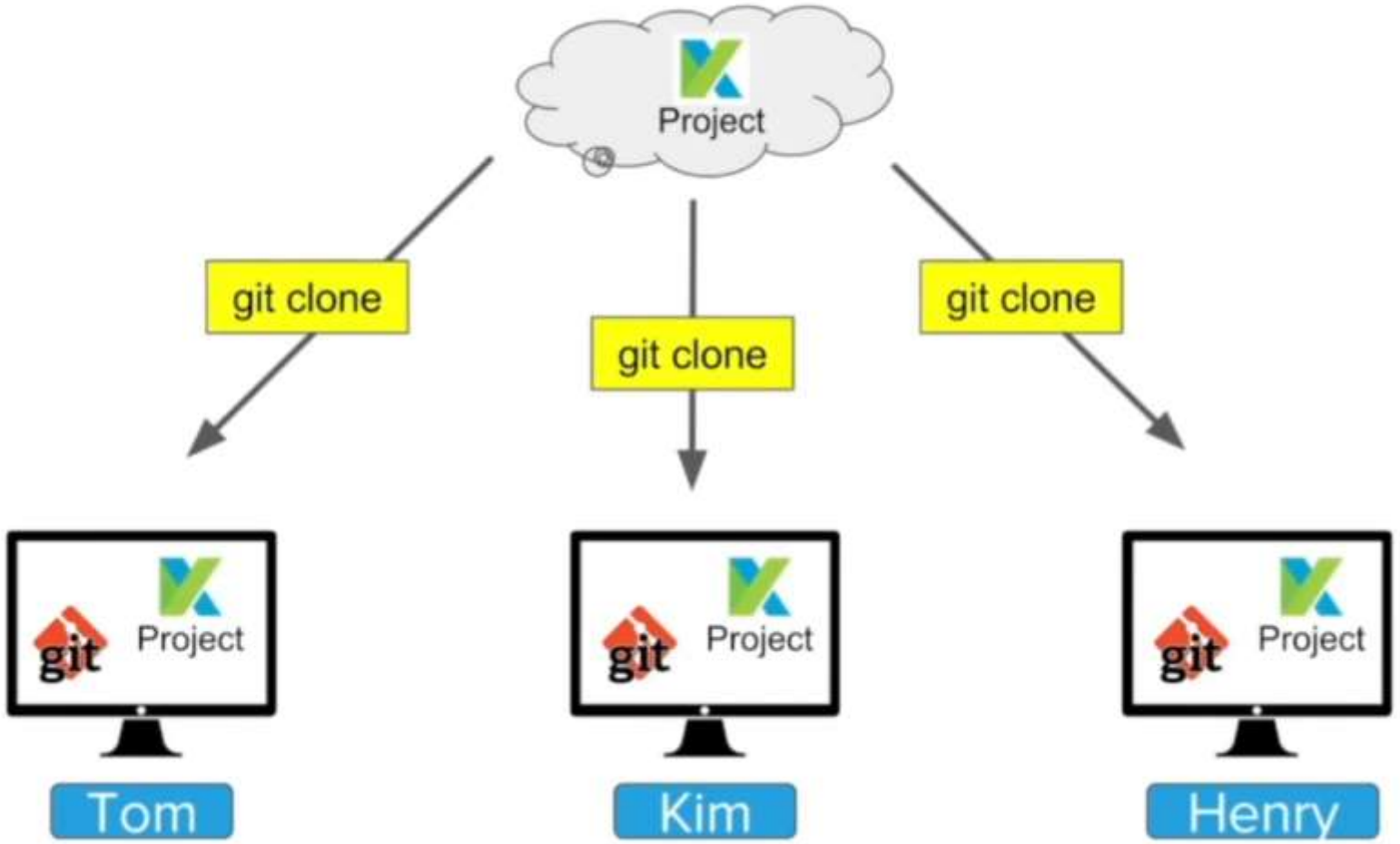
Tom

Kim

Henry

Step 3

Everyone will clone the project from remote to their local



Step 4

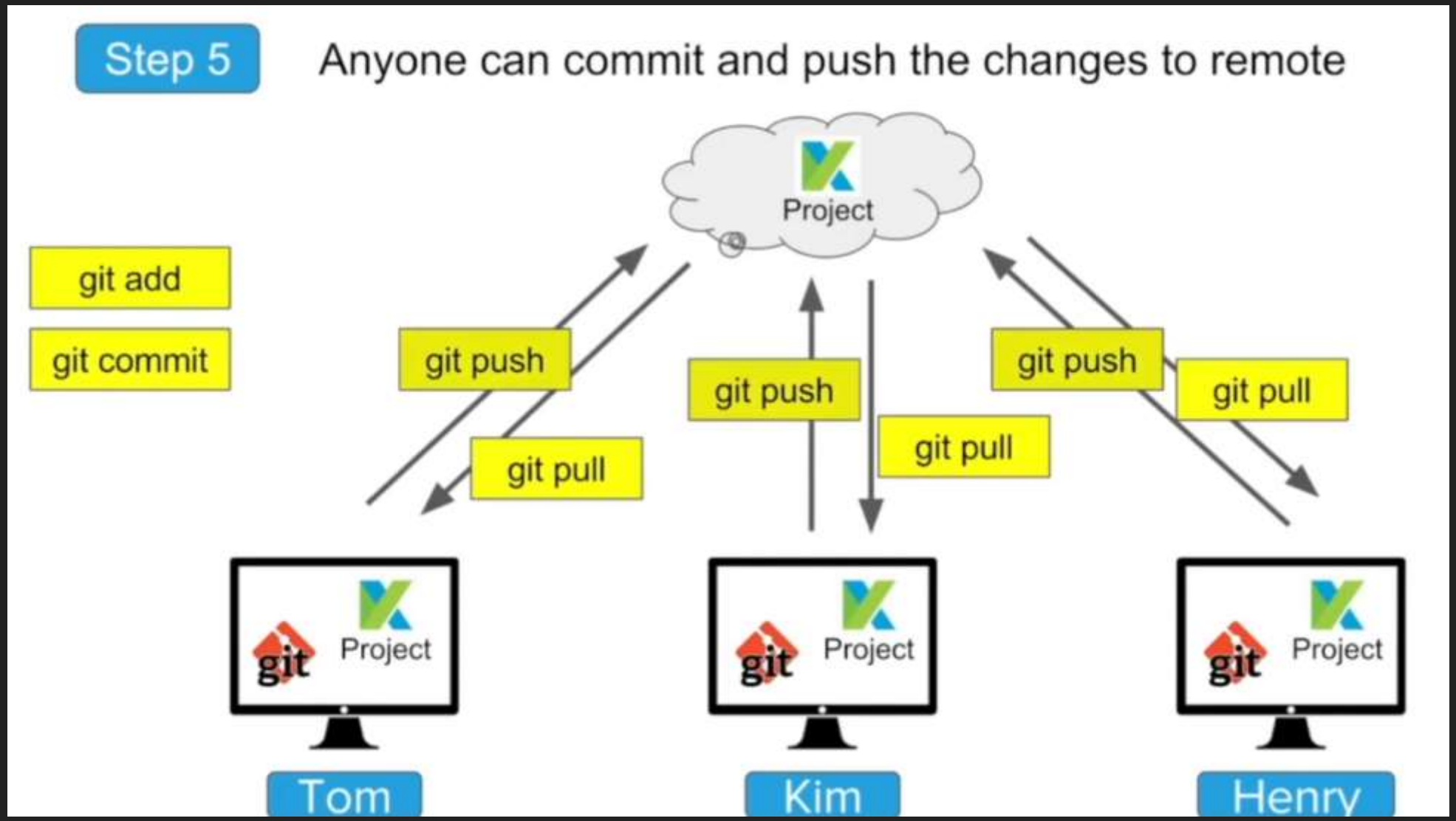
Everyone can now work on their local copies

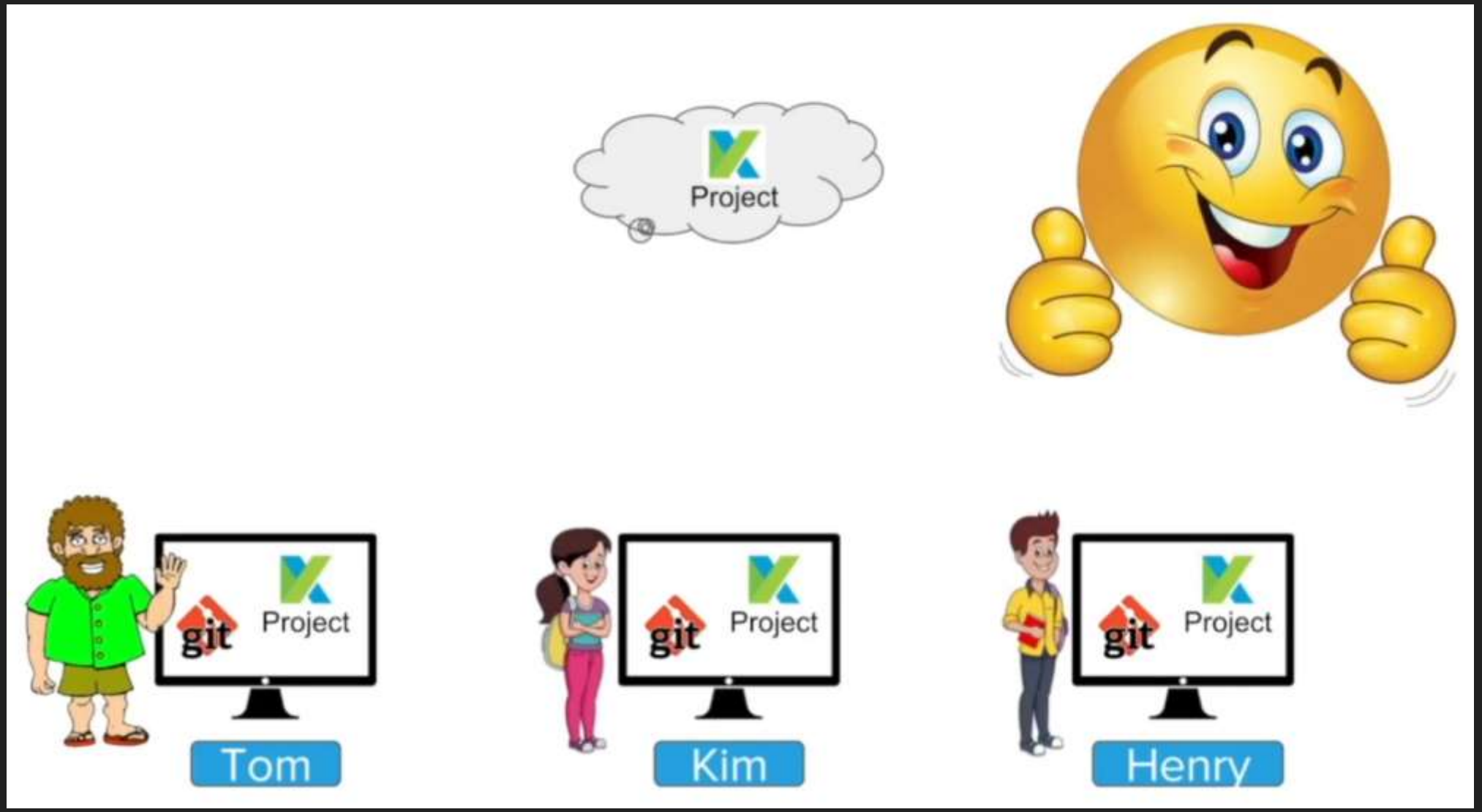
Connection to Remote not required

Tom

Kim

Henry


Step 5
Anyone can commit and push the changes to remote
Project
git add
git commit
git push
git pull
git push
git pull
git push
git pull
git
Project
git
Project
git
Project
Tom
Kim
Henry


Project
git
Project
Tom
git
Project
Kim
git
Project
Henry

Lets get Started!