Software Configuration Management

Kent State University
Prof. Jonathan Maletic

SCM

- Software Configuration Management involves:
- Revision control (version control) and change management
- Build and release management (diff and patch)
- Process and environment management
- Facilitate teamwork and collaboration
- Issue/Defect tracking

Change Management

- Version Control Systems (VCS): git, mercurial, svn, perforce, MS Team Foundation, cvs, rcs

- VCS help developers manage changes made to files (source code, documentation, build files, etc.)

- A revision number is given to each new version (or revision) of a file or set of files

- A commit defines the set of files associated with a new version

- VCS manage the revisions for searching and roll back

Why use Version Control

- Software is really complex and constantly changing.
- Software that compiled and passed tests yesterday may fail today.
- Which change broke the system? Without VCS this is a guessing game.
- VCS records what changes are made, when the change was made, and who made the change - maintains a complete history of changes made to a system

Uses of VCS

- You are working on fixing a bug or adding a new feature
- Can not do this on the “production code” (master copy) - the code that compiles, runs, and passes all the tests
- VCS allows developers to make changes over multiple days on their local copy without modifying the master copy
- After the bug is fixed or feature completed the developer can commit their work to the master copy

Collaboration

- You modify the code in file foo, get it working
- Your team mate modifies the code in file foo and gets it working
- You combine both version and it doesn’t work
- Something in one of the versions broke the other
- Merging can also be supported by VCS
- Version control is essential for coordination of multiple developers

Other Benefits

- Provides a complete time machine for the project - able to roll back to any previous version or examine changes made in the past

- Allows for multiple different variant of the same project - stand alone app, web version, iPhone version, etc

- Greatly simplifies concurrent development and merging of different changes

- Helps define the development workflow and process

Critical Tool

- Version control systems are a critical component in the software engineering process

- Version control systems along with other components of configuration management and team management directly support the quality goal of the project

- By not adopting and correctly using a version control system your project is most likely doomed

How VCS Work

- Versioning at file level
- Text files - diff can be used to determine what changed. This works at the character level (line or word)
- Binary files - at the file level only
- No understanding of syntax or programming language

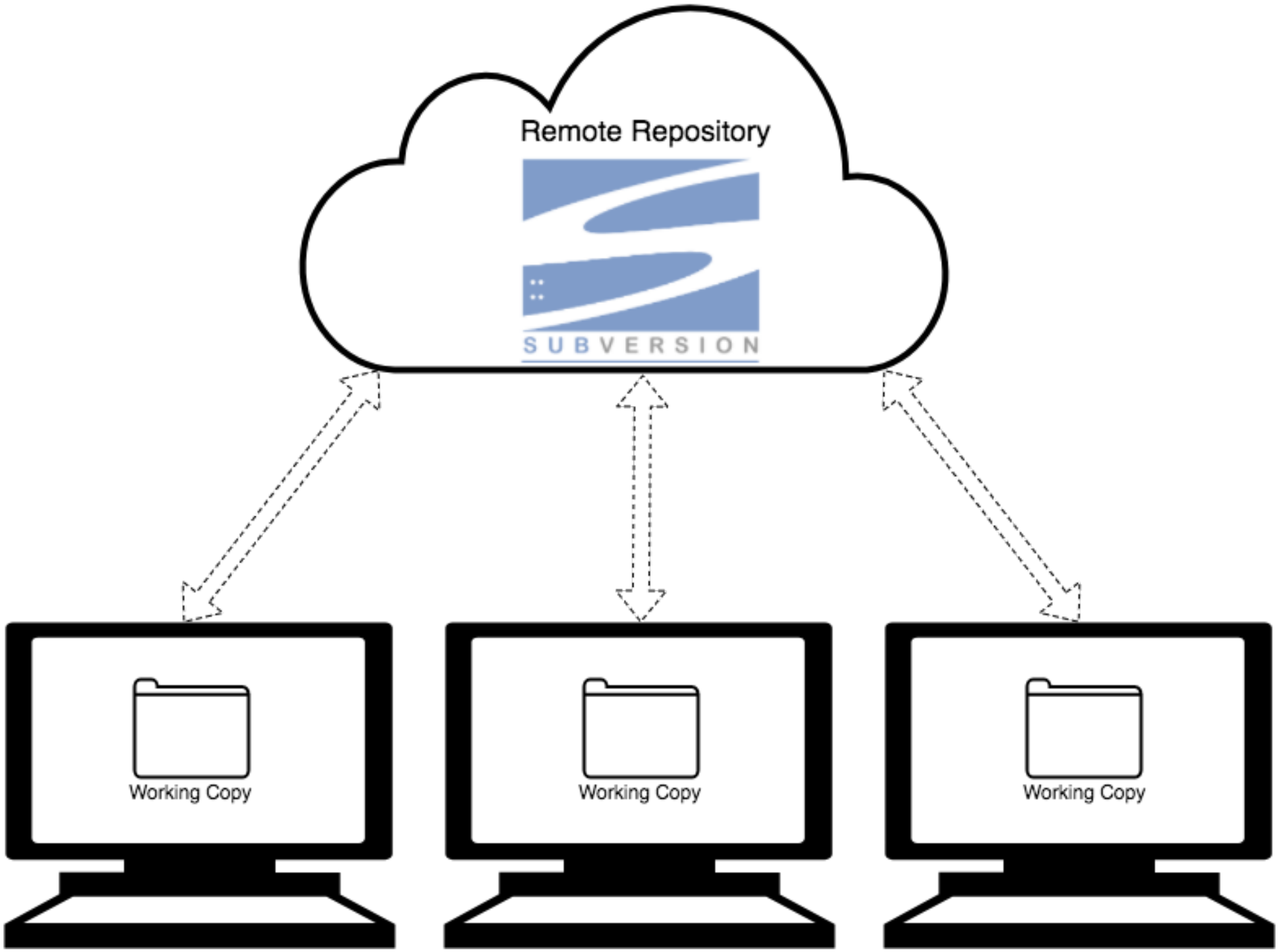
Types of VCS

- There are basically two types of version control systems
  - Centralized: svn, cvs, rcs
    - rcs use file locking
    - svn, cvs use version merging (with diff)
  - Distributed: git, mercurial
    - git uses version merging

Lock vs Merge

- Locking - one developer locks file and only they can modify the file (lock and modify)
    - Advantage is that there is no merging problems
    - Disadvantage is that it slows down development by not allowing others to work on the file (hence they are rarely used anymore)
- Merging has no restrictions on access (copy and modify)
    - Advantage is developers can work in parallel
    - Disadvantage is that there will be merging problems

Centralized Model



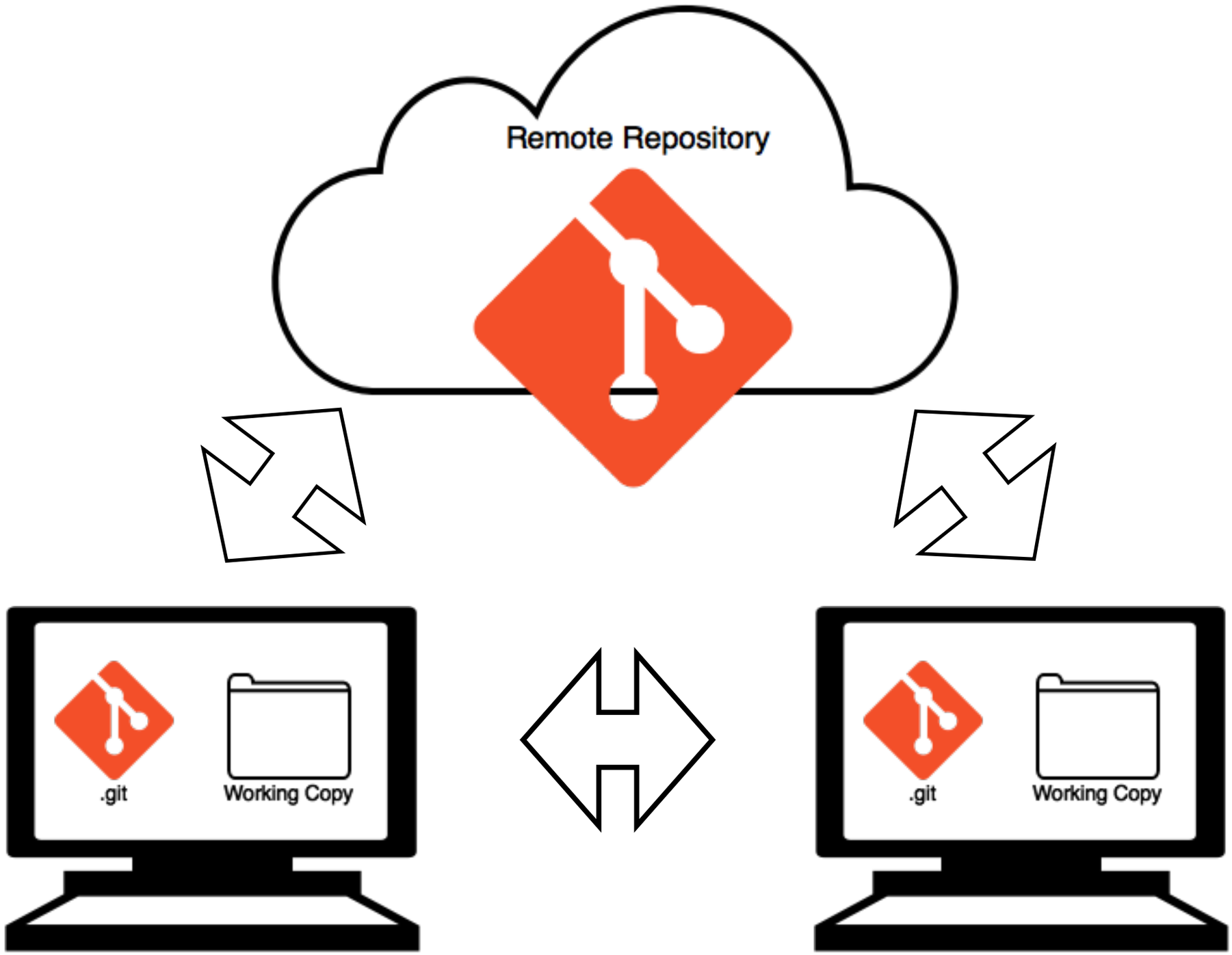
Subversion (svn)

- Subversion uses a centralized repository to store the main copy
- Developers check out a copy of this repo locally on their machine. Changes can be made to this local copy.
- Developers than commit their changes to the main copy
- Other developers must be informed of these changes and either do an update to their local copy or delay the merging process to a later time
- Merge conflicts become a difficult issue to resolve properly
- One sequential sequence of version numbers

svn Workflow

- Update a local working copy
- Perform changes to a local copy of the repo
- Commit changes to central repo
- Need network access to the central server
- Merging is difficult, branching tends to create problems

Distributed Model



Git 

- Git is a distributed version control system (peer-to-peer)
- Git supports the process of branching
- A master branch is maintained - this is the working production code branch
- Developers clone or branch the master as a local copy
- Changes can be pushed into the master. Changes can also be pulled into other developers local copy
- No one sequence of version numbers

Git Workflow

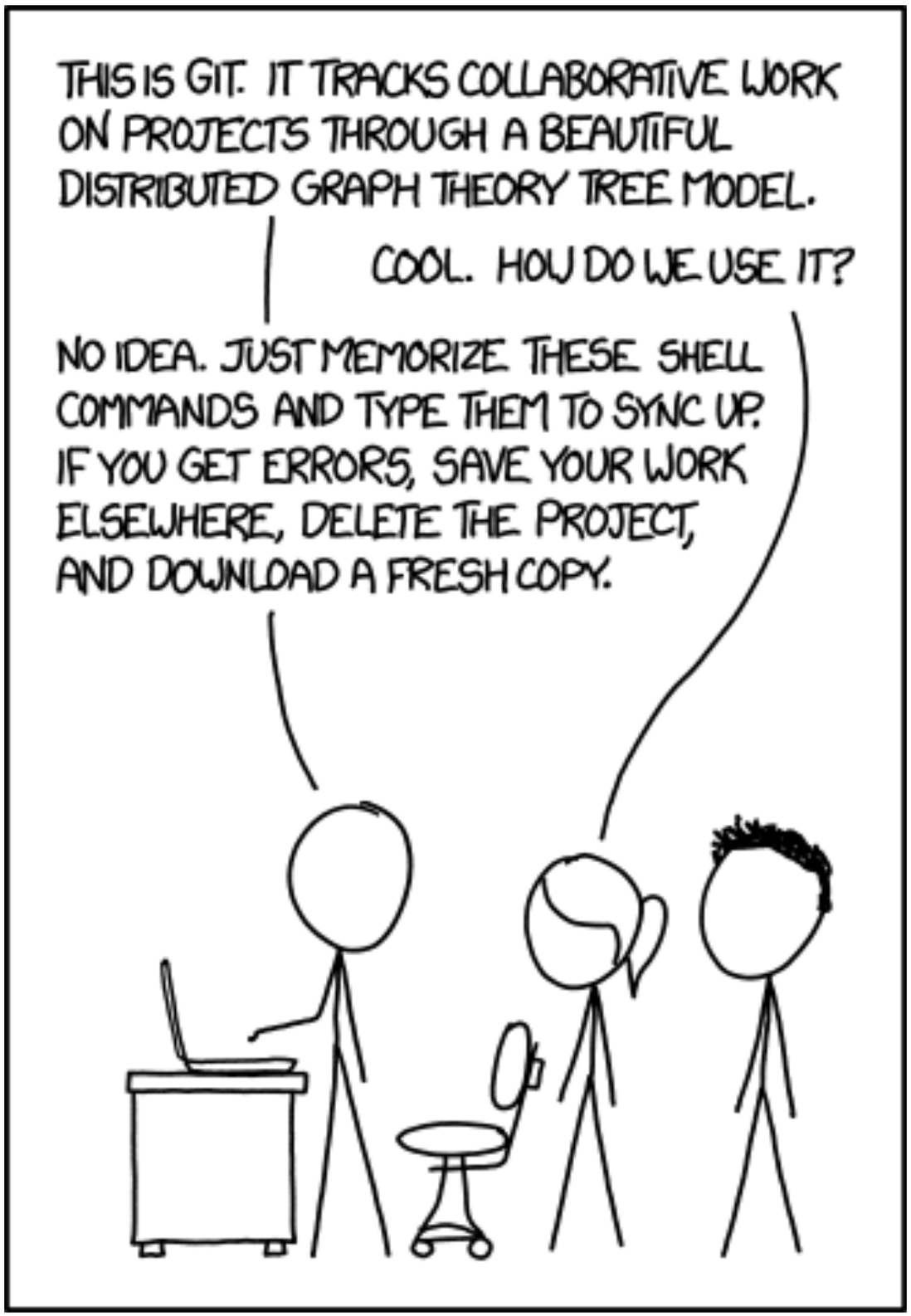
- Versions are identified by a SHA1 id (160 bit Hex decimal number) - Secure Hash Algorithm - gives a unique id. Records changes not versions

- Each local copy is a full fledged repo that can be worked on without a network connection

- Developers clones a repo and make changes, then pushes to others using the repo. Tracks merge data

- Advantage: fast, flexible, multiuser, very powerful

- Disadvantage: Complex and difficult to learn (over 160 commands)


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.