Revision Control

Dr. Hyrum Carroll

September 13, 2016

(Updated September 22, 2016)

Revision Control: Motivation

The typical problem

`http://www.phdcomics.com/comics.php?n=1531`

Does this look familiar?

```
-rw-r--r-- 1 usr1 Sep 19 16:53 simulation.f
-rw-r--r-- 1 usr1 Sep 19 16:53 #simulation.f#
-rw-r--r-- 1 usr1 Sep 19 14:38 simulation.f~
-rw-r--r-- 1 usr1 Sep 17 12:01 simulation.f.old
-rw-r--r-- 1 usr1 Feb 21  2014 simulation.f.bak
-rw-r--r-- 1 usr1 Sep 16  2014 simulation.f.orig
-rw-r--r-- 1 usr1 Dec 13  2010 simulation.f.from-BYU
```

Or maybe this?

```
-rw-r--r-- 1 usr1 Sep 19 16:53 simulation.f
-rw-r--r-- 1 usr1 Sep 17 12:01 simulation.f.2014.09.17
-rw-r--r-- 1 usr1 Feb 21  2010 simulation.f.2010.02.21
```

Possible Uses

- Source code projects (Python, Perl, Fortran, C, C++, bash, MATLAB, etc.)
- Configuration files
  - For code (e.g., Makefiles)
  - Systems (e.g., .bashrc)
  - security
  - Web server (e.g., httpd.conf)
- Research projects
  - Digital lab notebook
  - Scripts
  - Manuscripts
- Web documents (HTML, CSS, Javascript, etc)

Revision Control: History

Version Control System (VCS)

- Good for storing changes
  - Stores notes about changes
  - Allows for displaying differences of previous versions
  - Allows for checking out previous versions
- E.g., RCS (1985)

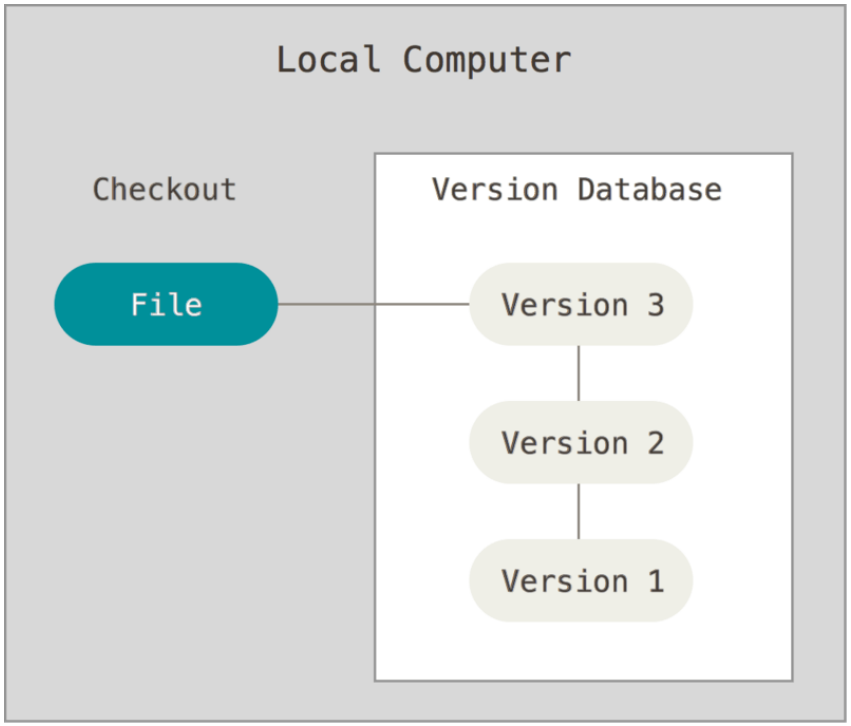


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Revision Control: History

Centralized Version Control Systems (CVCS)

- Good for storing changes
- Good for collaborating
  - Allows for "branches" for subprojects
  - Automatically handles simultaneous changes (unless they conflict)
- E.g., CVS, Subversion (SVN), and Perforce

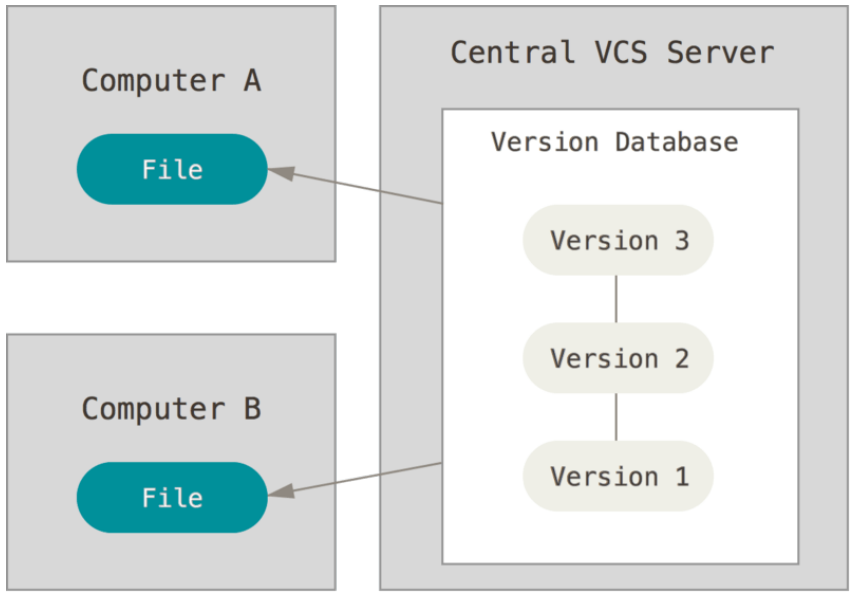


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Revision Control: History

Distributed Version Control Systems (DVCS)

- Good for storing changes
- Good for collaborating
- Mirror of repository
  - Built to support querying changes locally
  - Every clone (local copy) is a full version of all the changes
- E.g., Git (, Mercurial, Bazaar & Darcs)

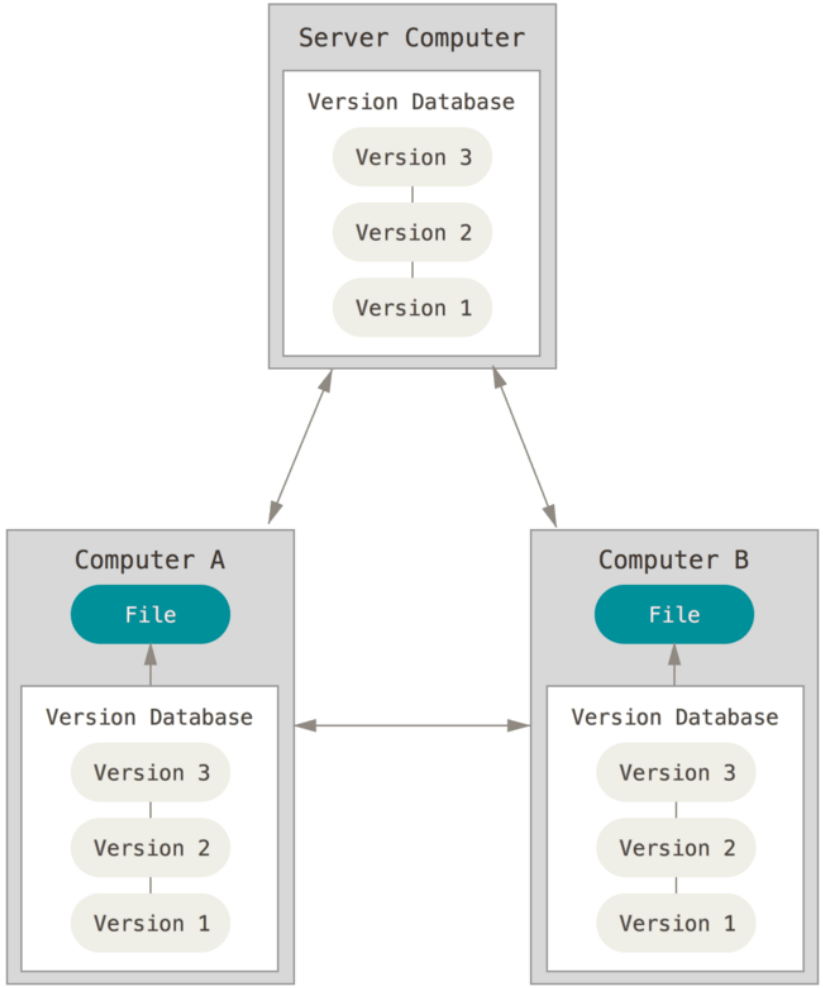


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Revision Control: History

Git

Git:

- Created by Linus Torvalds (2005), the creator of Linux
- Goals:
  - Speed
  - Simple design
  - Strong support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Handle large projects (e.g., the Linux kernel) efficiently (speed and data size)

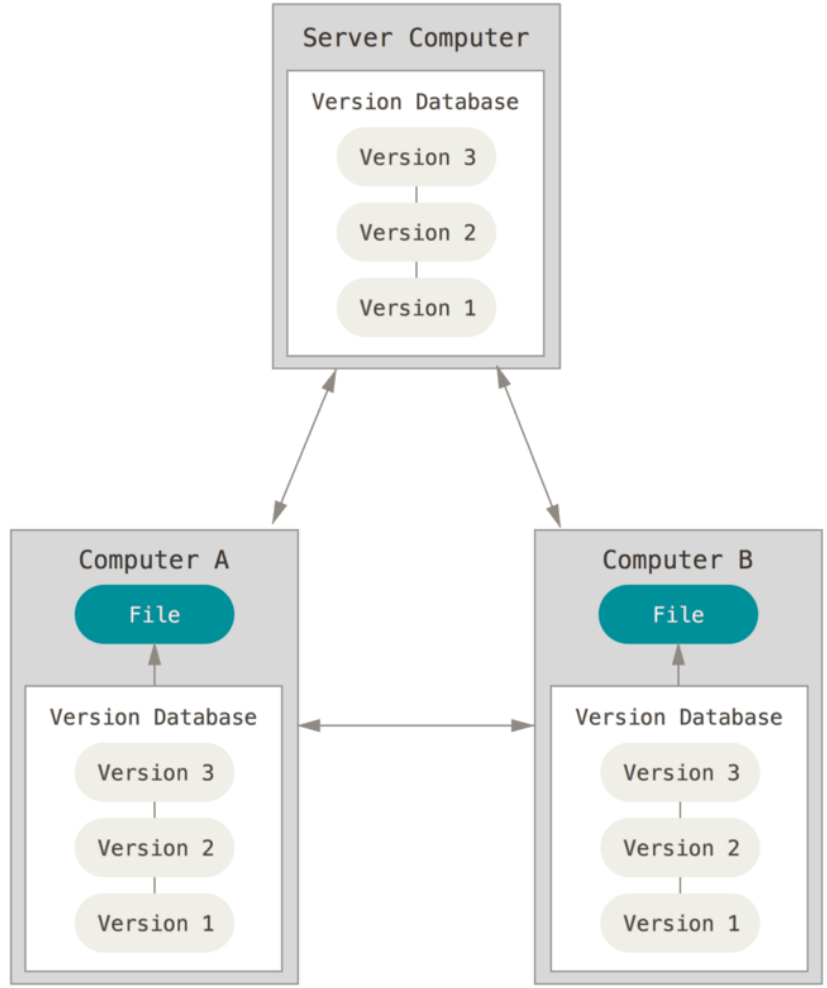


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Git

Git

Stream of snapshots:

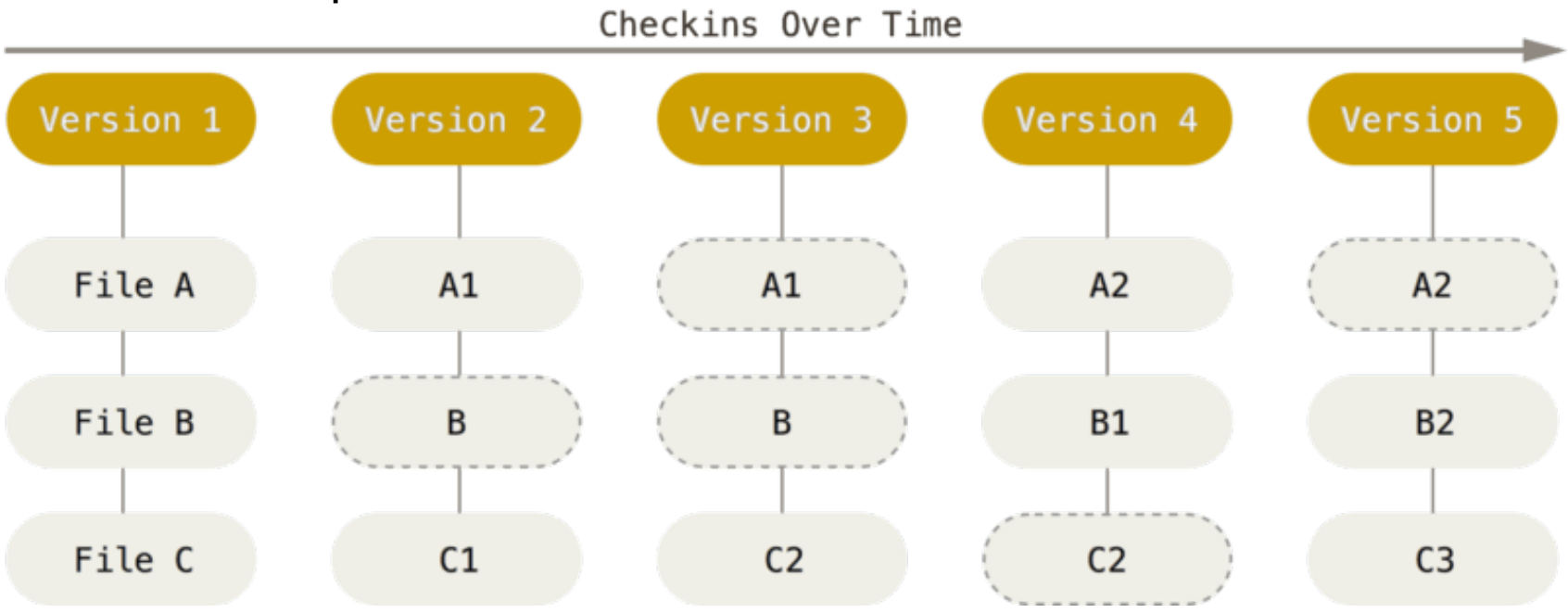


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Git

Three main sections:

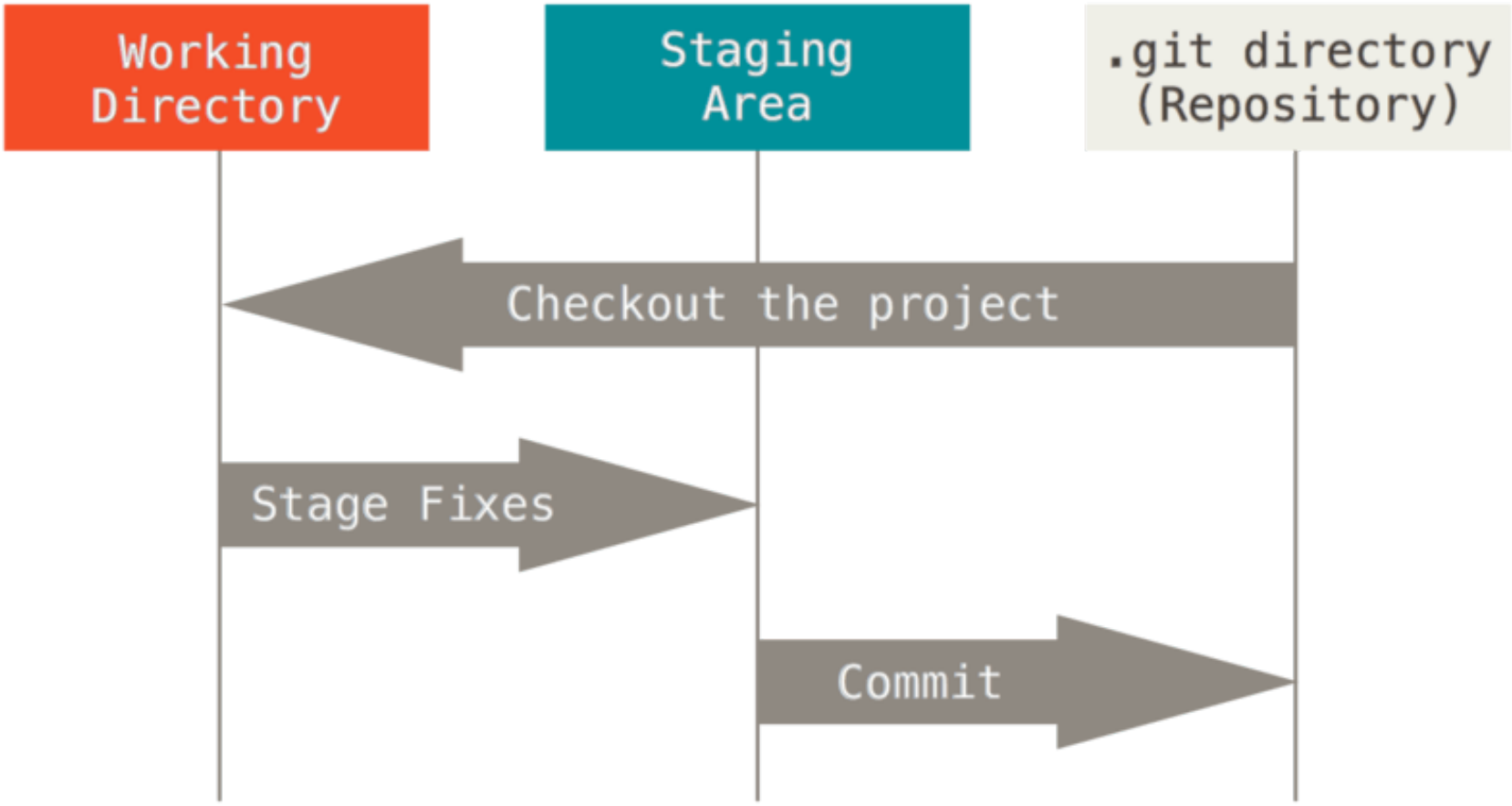


Image source: Pro Git by Scott Chacon and Ben Straub (Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

Git

Initial Steps:

- `git clone URL`

or

- `git init projectName`
- `cd projectName`

Git

Basic Git workflow:

- Modify files (in working directory)
- Stage the files (which adds "snapshots" to your staging area)
- Commit (copies the "snapshot" from the staging area to your Git directory (`.git`))

Basic Git workflow:

- `git add .` # add modified files to the staging area (from the working directory)
- `git commit -m "Message about changes"` # add modified files to local repository
- `git pull` # sync local repository with changes from the remote repository
- `git push` # sync local repository with your changes to the remote repository

Subversion (SVN)

Revision Control: Vocab

Basic Setup

- Repository (repo): The database storing the files.
- Server: The computer storing the repo.
- Client: The computer connecting to the repo.
- Working Copy: Your local directory of files, where you make changes.
- Trunk/Main: The primary location for code in the repo. Think of code as a family tree — the trunk is the main line.

(Source: `betterexplained.com`)

Revision Control: Vocab

Basic Actions

- Add: Put a file into the repo for the first time, i.e. begin tracking it with Version Control.
- Revision: What version a file is on (v1, v2, v3, etc.).
- Check out: Download a file from the repo.
- Check in: Upload a file to the repository (if it has changed). The file gets a new revision number, and people can "check out" the latest one.
- Checkin Message: A short message describing what was changed.
- Changelog/History: A list of changes made to a file since it was created.
- Head: The latest revision in the repo.
- Update/Sync: Synchronize your files with the latest from the repository. This lets you grab the latest revisions of all files.
- Revert: Throw away your local changes and reload the latest version from the repository.

(Source: `betterexplained.com`)

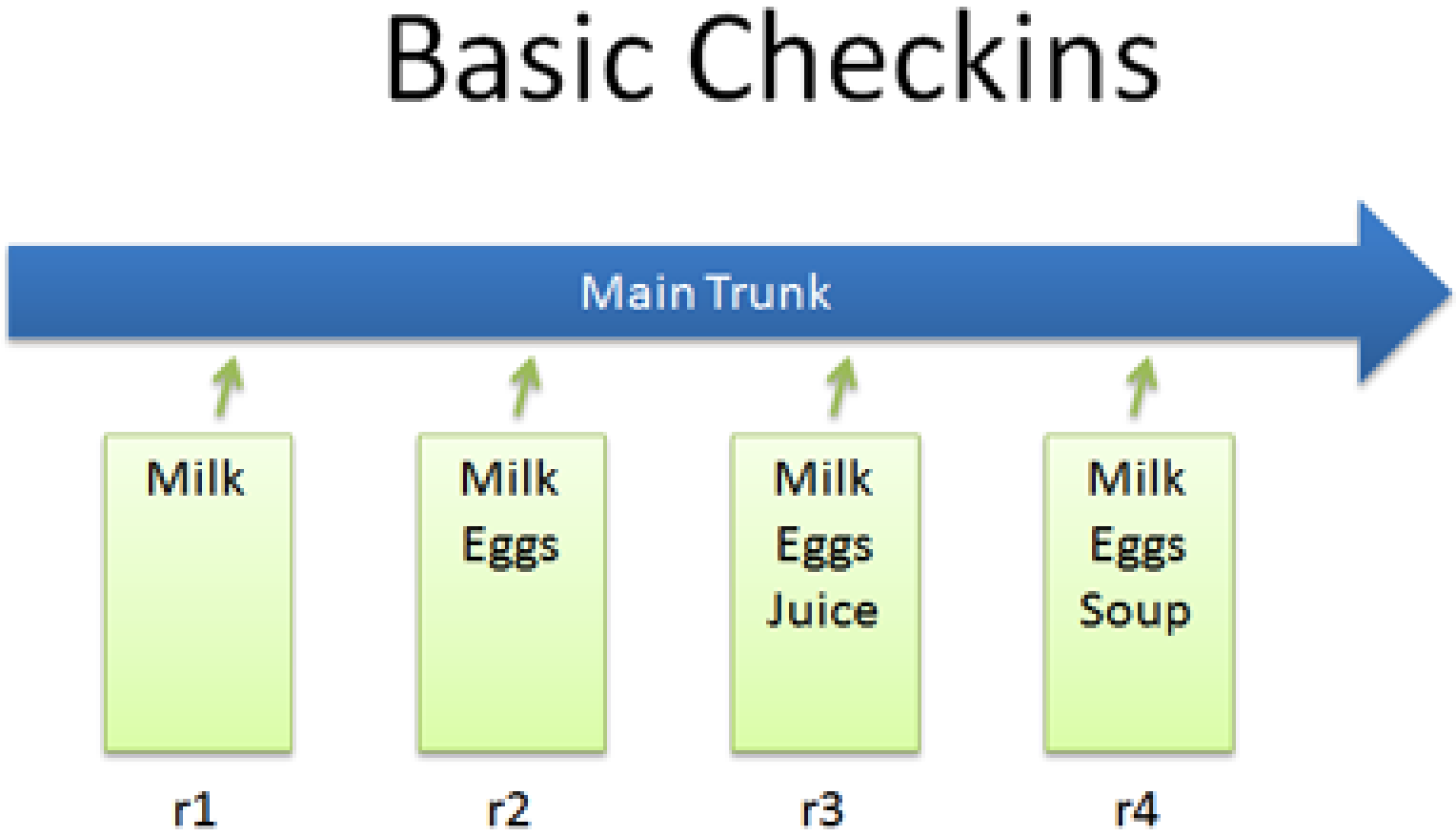
Revision Control: Vocab

Advanced Actions

- Branch: Create a separate copy of a file/folder for private use (bug fixing, testing, etc). (both a verb and a noun).
- Diff: Finding the differences between two files. Useful for seeing what changed between revisions.
- Merge (or patch): Apply the changes from one file to another, to bring it up-to-date.
- Conflict: When pending changes to a file contradict each other (both changes cannot be applied).
- Resolve: Fixing the changes that contradict each other and checking in the correct version.
- Locking: Taking control of a file so nobody else can edit it until you unlock it. Some version control systems use this to avoid conflicts.
- Breaking the lock: Forcibly unlocking a file so you can edit it.

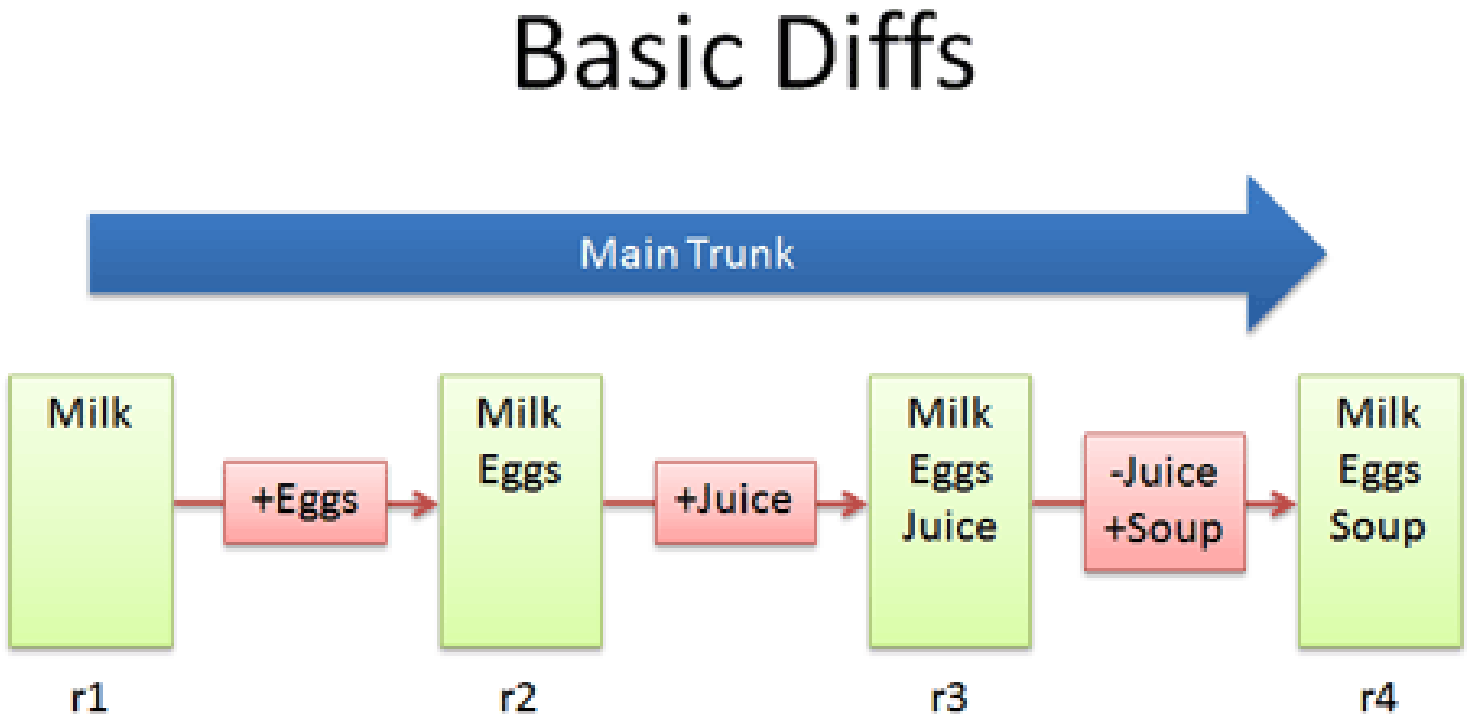
(Source: `betterexplained.com`)

Revision Control



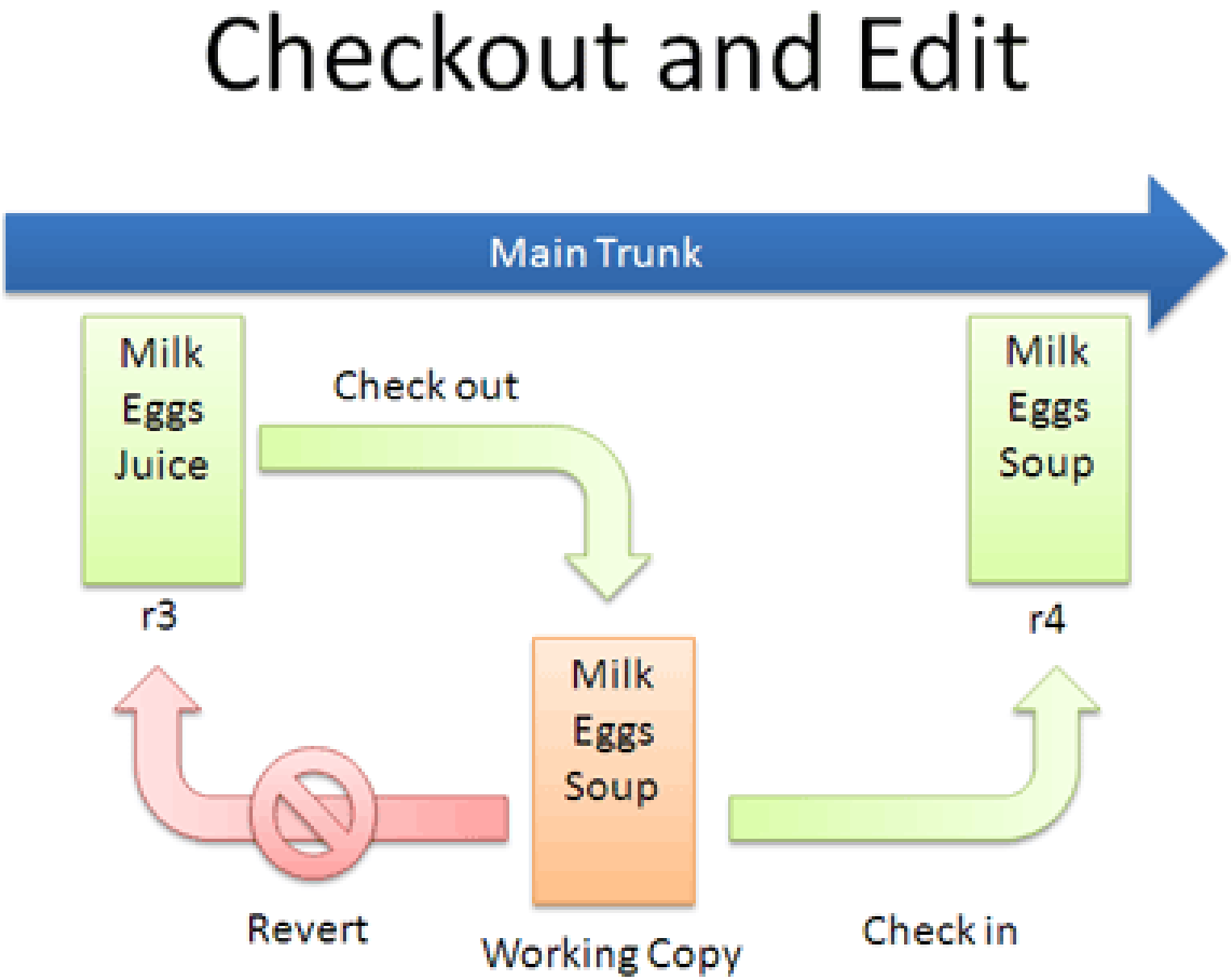
(Source: `betterexplained.com`)

Revision Control



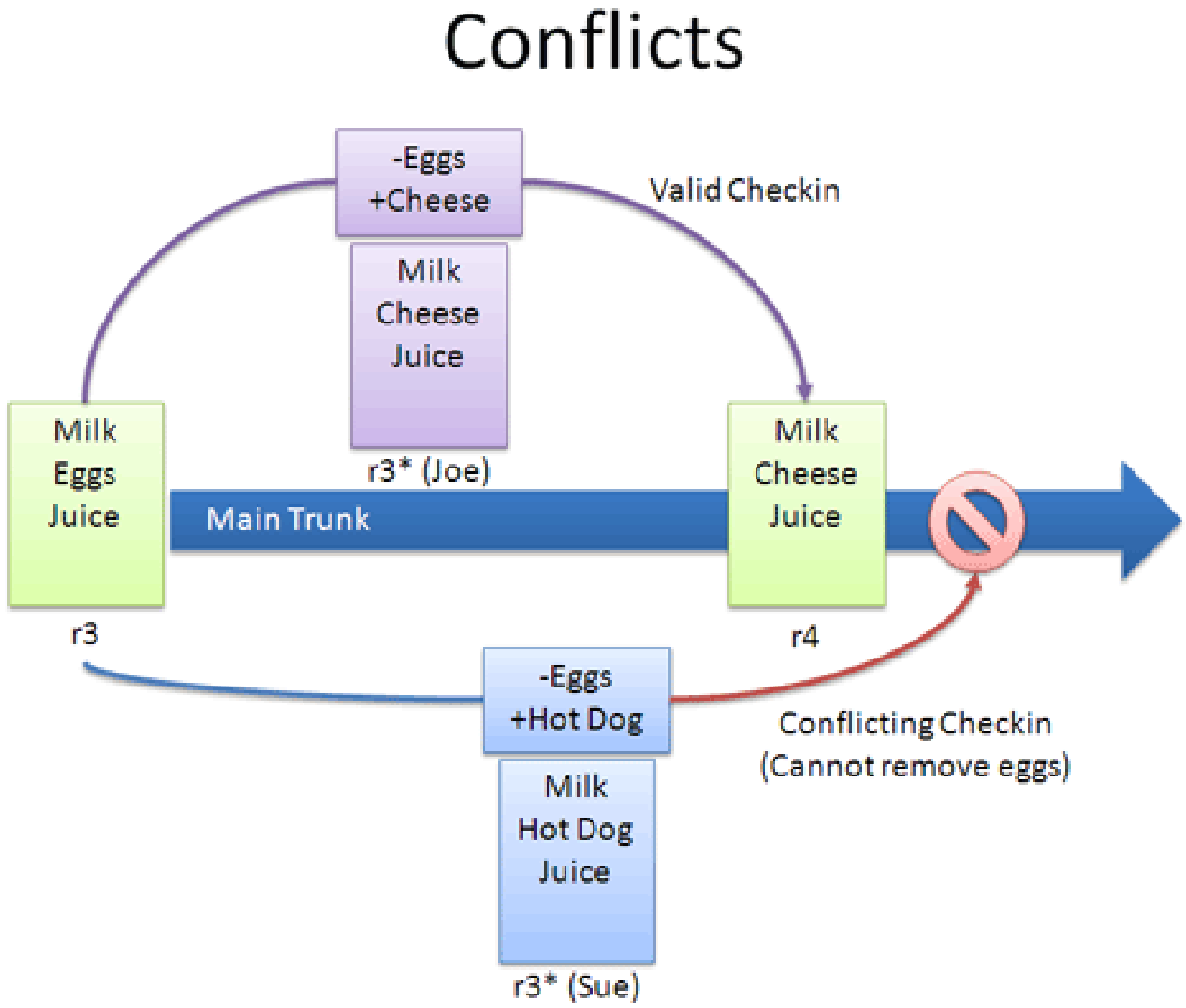
(Source: `betterexplained.com`)

Revision Control



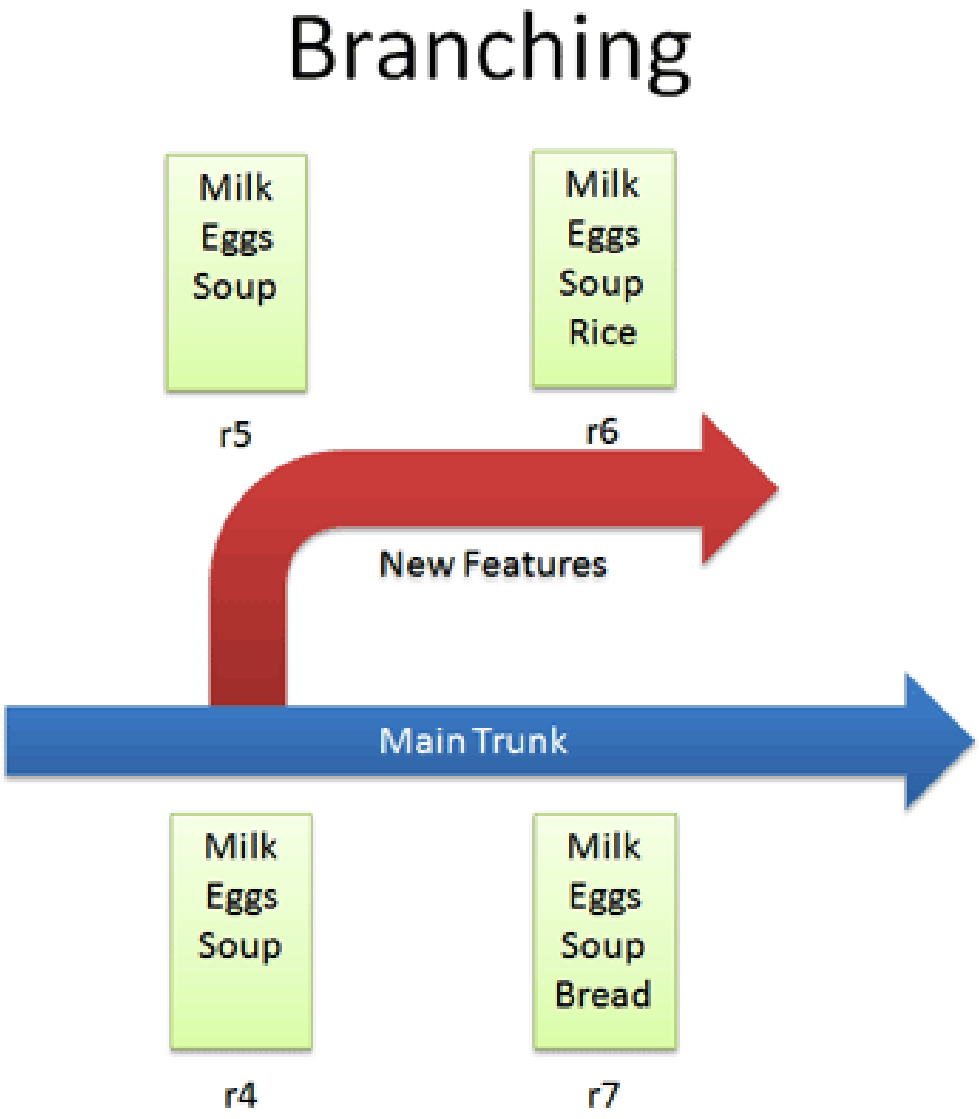
(Source: betterexplained.com)

Revision Control



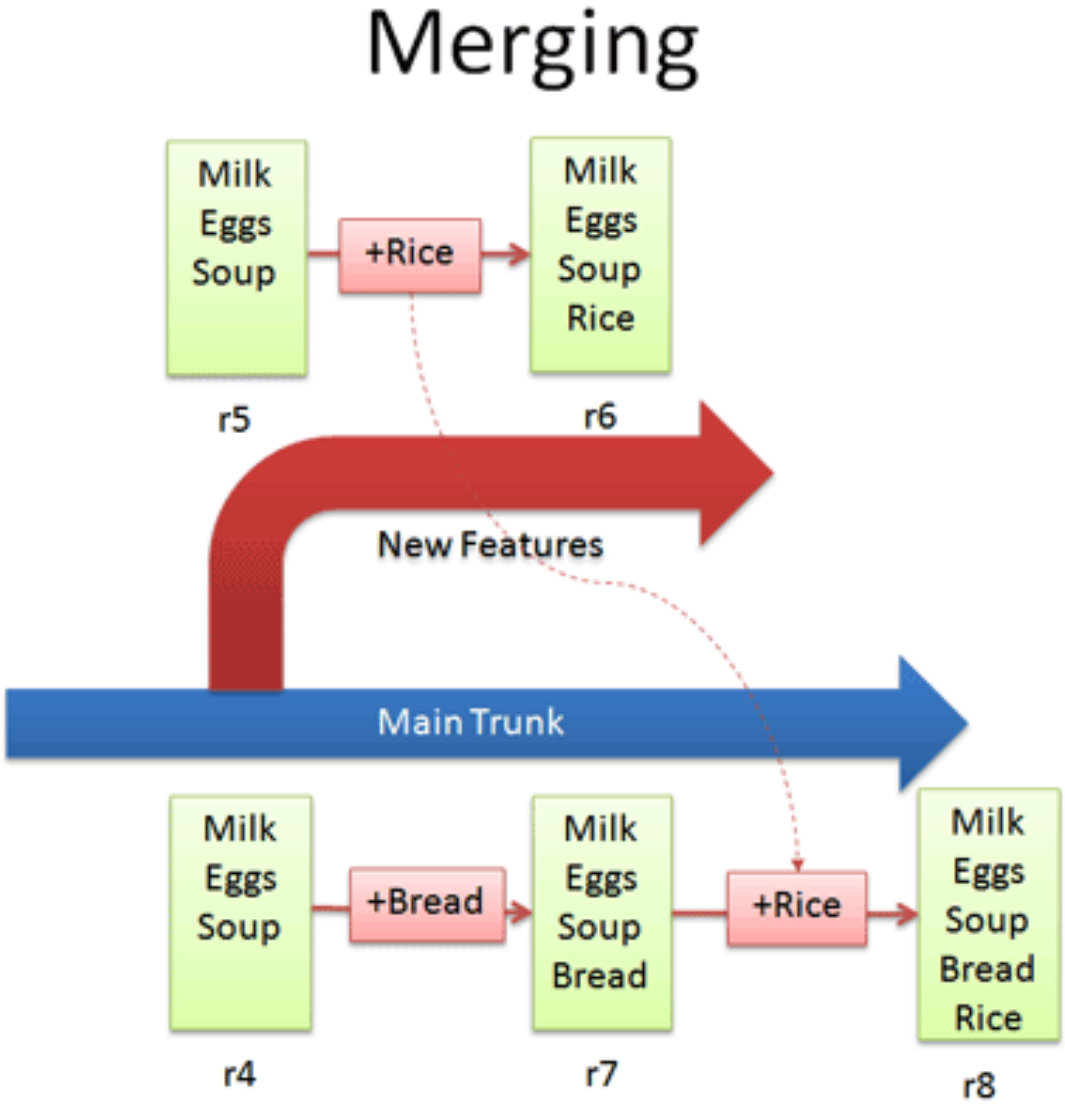
(Source: `betterexplained.com`)

Revision Control



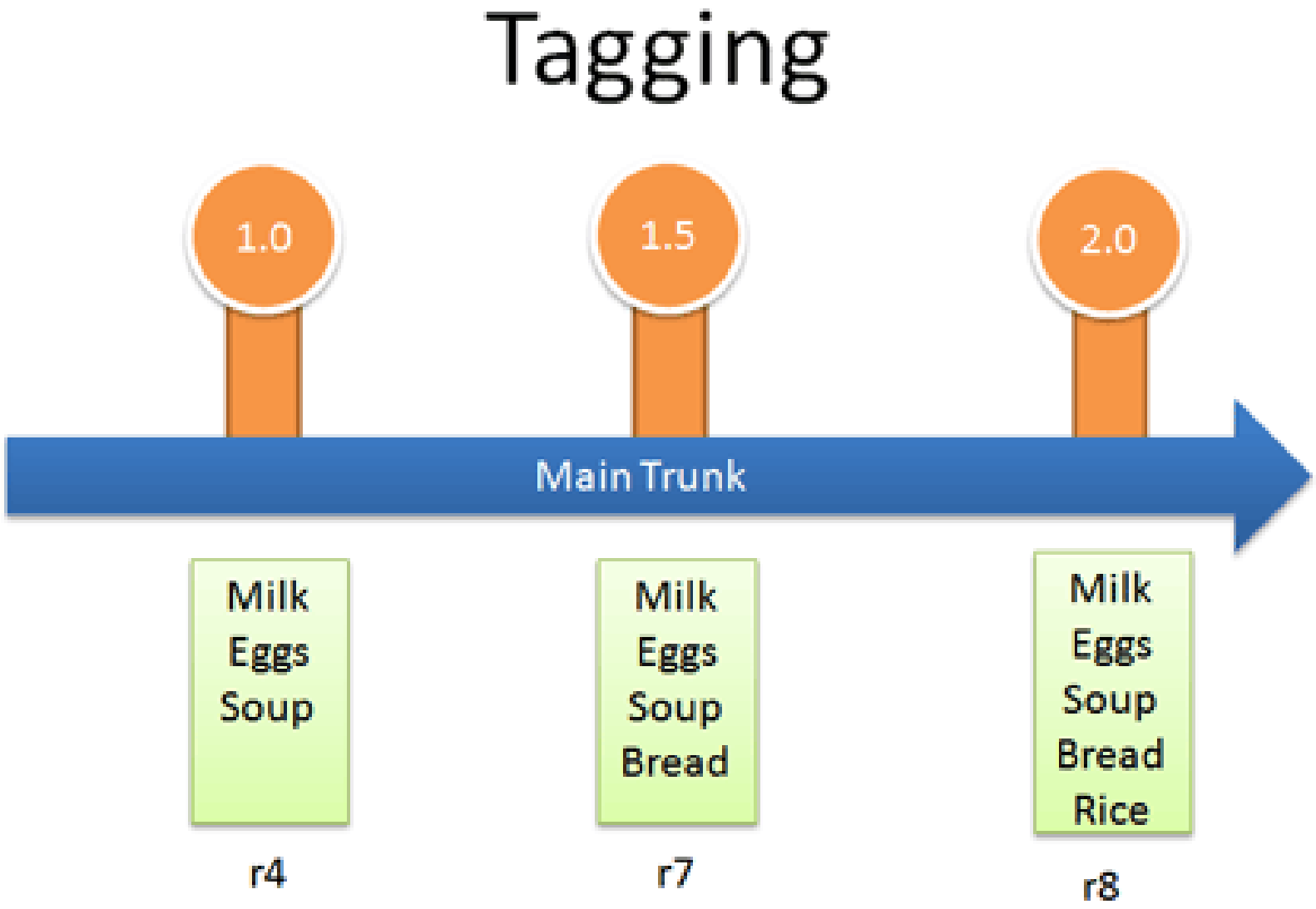
(Source: `betterexplained.com`)

Revision Control



(Source: `betterexplained.com`)

Revision Control



(Source: `betterexplained.com`)

SVN Setup Instructions

On the web server (*i.e.*, `ranger`), in `~/public_html/`:

1. `svnadmin create newProj`
2. `chmod 700 newProj/`
3. `cd newProj/conf/`
4. `echo '[users]' > passwd`
5. `echo 'user1' = ToPsEcReT >> passwd`
6. `chmod go-rwx passwd`
7. `cp ~cs/public_html/share/svnserve.conf .`
   either the above command or the next four
8. `echo "[general]" > svnserve.conf`
9. `echo "anon-access = none" >> svnserve.conf`
10. `echo "auth-access = write" >> svnserve.conf`
11. `echo "password-db = passwd" >> svnserve.conf`

SVN New Repository Instructions

On YOUR machine, (*e.g.*, a laptop):

1. `mkdir mortgageCalculator`
2. `mkdir mortgageCalculator/trunk/`
3. `emacs mortgageCalculator/trunk/assignment5.f90`
4. `emacs mortgageCalculator/trunk/Makefile`
5. `svn import --username user1 mortgageCalculator/ \`
   `svn://svn.cs.mtsu.edu/$USER/public_html/newProj \`
   `-m "Initial import"`

```
Authentication realm: <svn://svn.cs.mtsu.edu:3690> 22d56e8
Password for 'user1':
Adding         mortgageCalculator/trunk
Adding         mortgageCalculator/trunk/assignment5.f90
Adding         mortgageCalculator/trunk/Makefile
```

SVN Check Out Repository Instructions

On the YOUR machine, (*e.g.*, a laptop):

1. `svn checkout --username user1`
   `svn://svn.cs.mtsu.edu/$USER/public_html/newProj`

```
A     newProj/trunk
A     newProj/trunk/assignment5.f90
A     newProj/trunk/Makefile
Checked out revision 1.
```