Software Development on Linux Systems

4002-XXX-XX

By

Cody Van De Mark

Today

- Version Control
- Project Planning

Version Control

- Version control is a very important part of development that is used in virtually every place in industry

- Version control is the processing of storing code by each update (version) so that all developers can work on the same code at the same time

- Historically, it was difficult for developers to work on similar files at the same time unless they would copy all of the code each developer changed into a single new file

Version Control

- Version control allows developers to commit their code to each other without overwriting other developers' code
- Version control branches back to the 70s/80s in its infancy
- Concurrent Version Control (CVS) was released as open source in 1990 and served as the leader in version control for the 1990s
- In 2000, another open source version control system launched, Subversion (SVN), which solved many problems that CVS faced

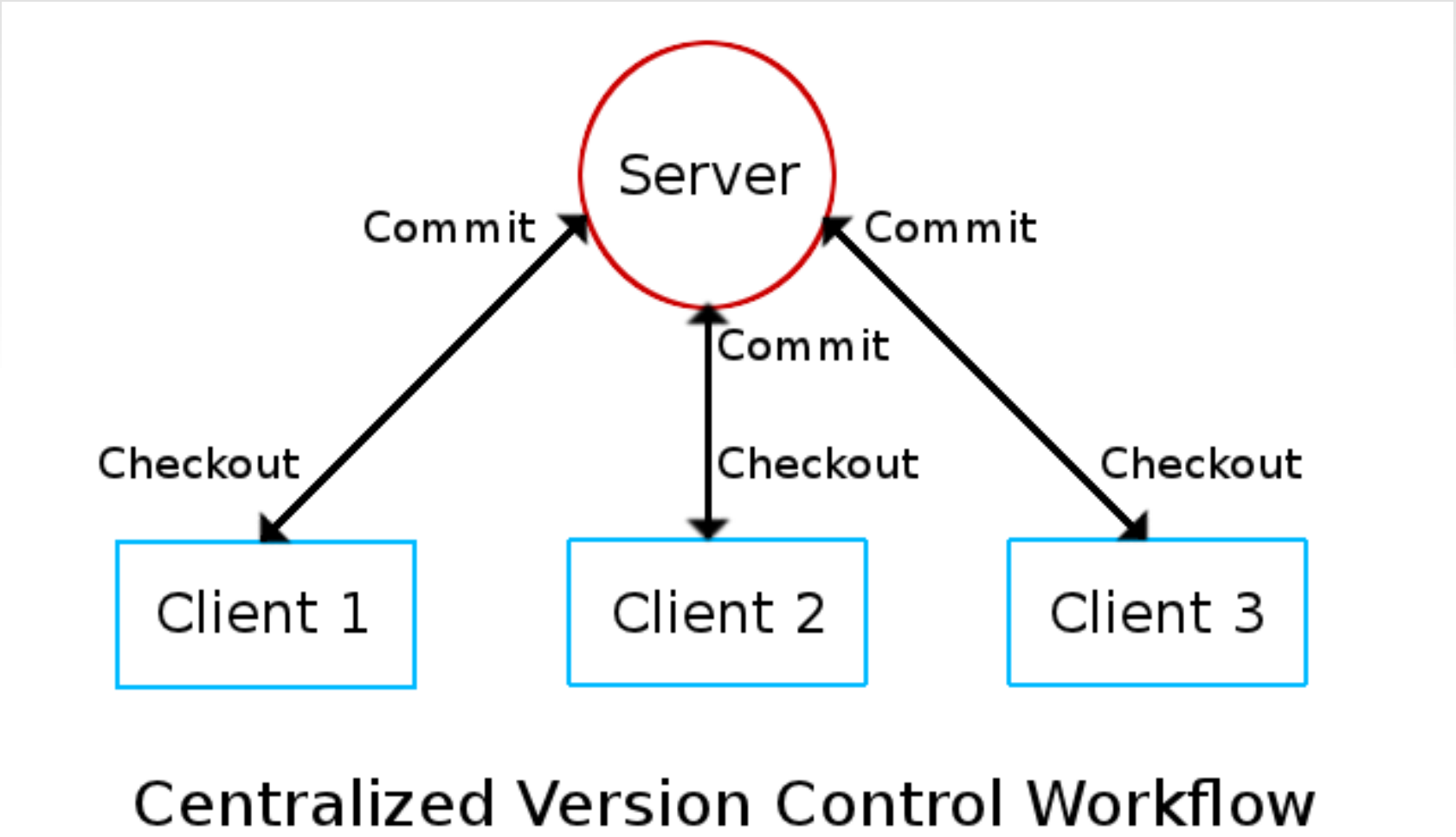
Version Control

- Subversion is still the standard of the industry, but like CVS required a central server for all of the code
- Version control in open source tends to be a decade ahead of industry
- The open source community thrives on new distributed version control systems, such as git, mercurial and bazaar
- While CVS and SVN hold a central version control server, distributed version control systems do not have any centralized location and can't fail unless every developer's machine died

Centralized Version Control

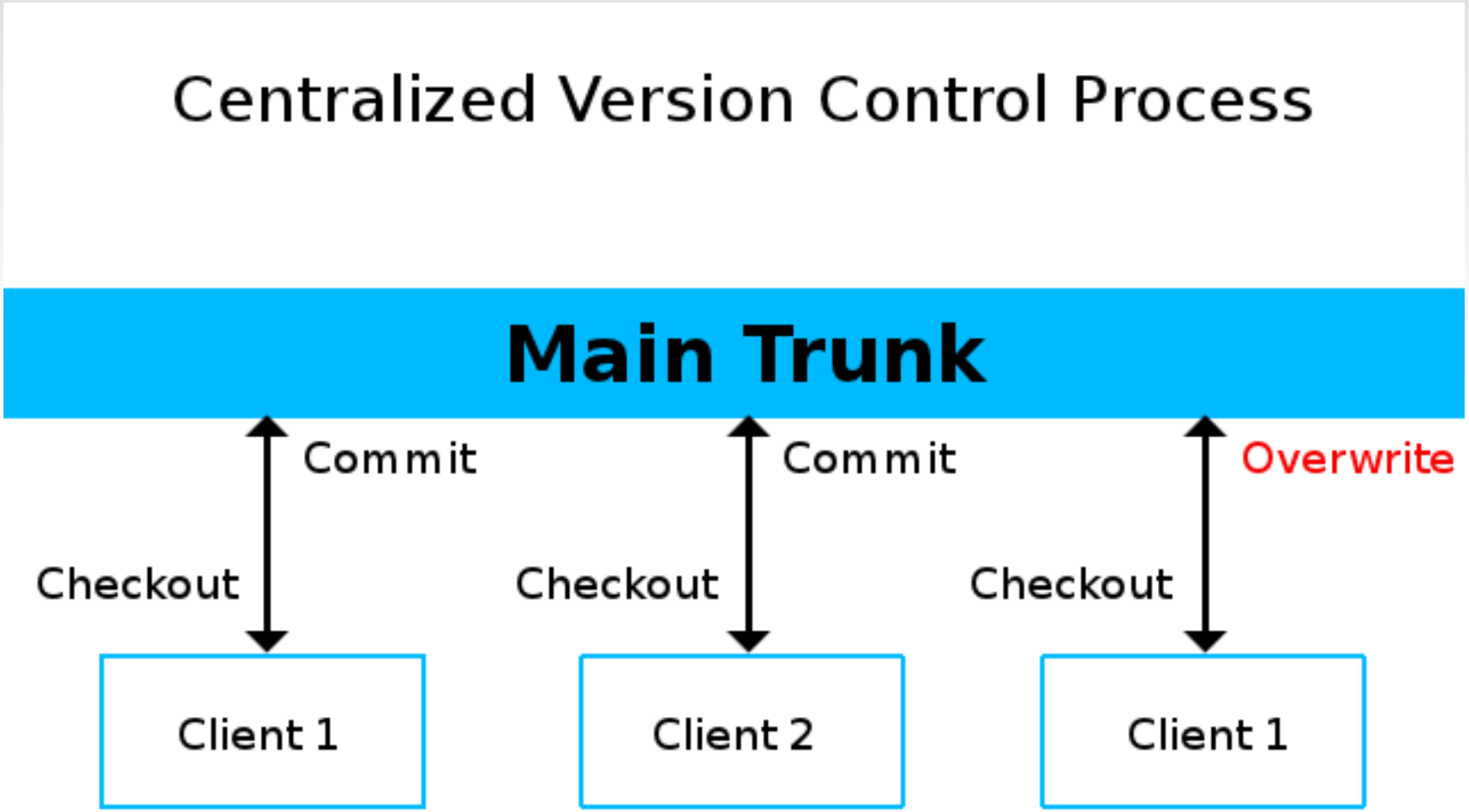
- Centralized version control has a main (central) server that everyone commits code to and pulls code from
- Server holds main trunk and people checkout the branches they need onto their local machine
- People usually do not have the entire trunk on their machine, unless the specifically pull everything
- Code needs to be up to date with the main server to commit or else it will prevent a commit or overwrite other users' code

Centralized Version Control



Centralized Version Control Workflow

Centralized Version Control



Distributed Version Control

- Decentralized version control can have many work flows, including a centralized approach
- Decentralized version control allows users to push and pull from any other user or machine
- Users may push and pull from one specific machine taking a centralized approach

Distributed Version Control

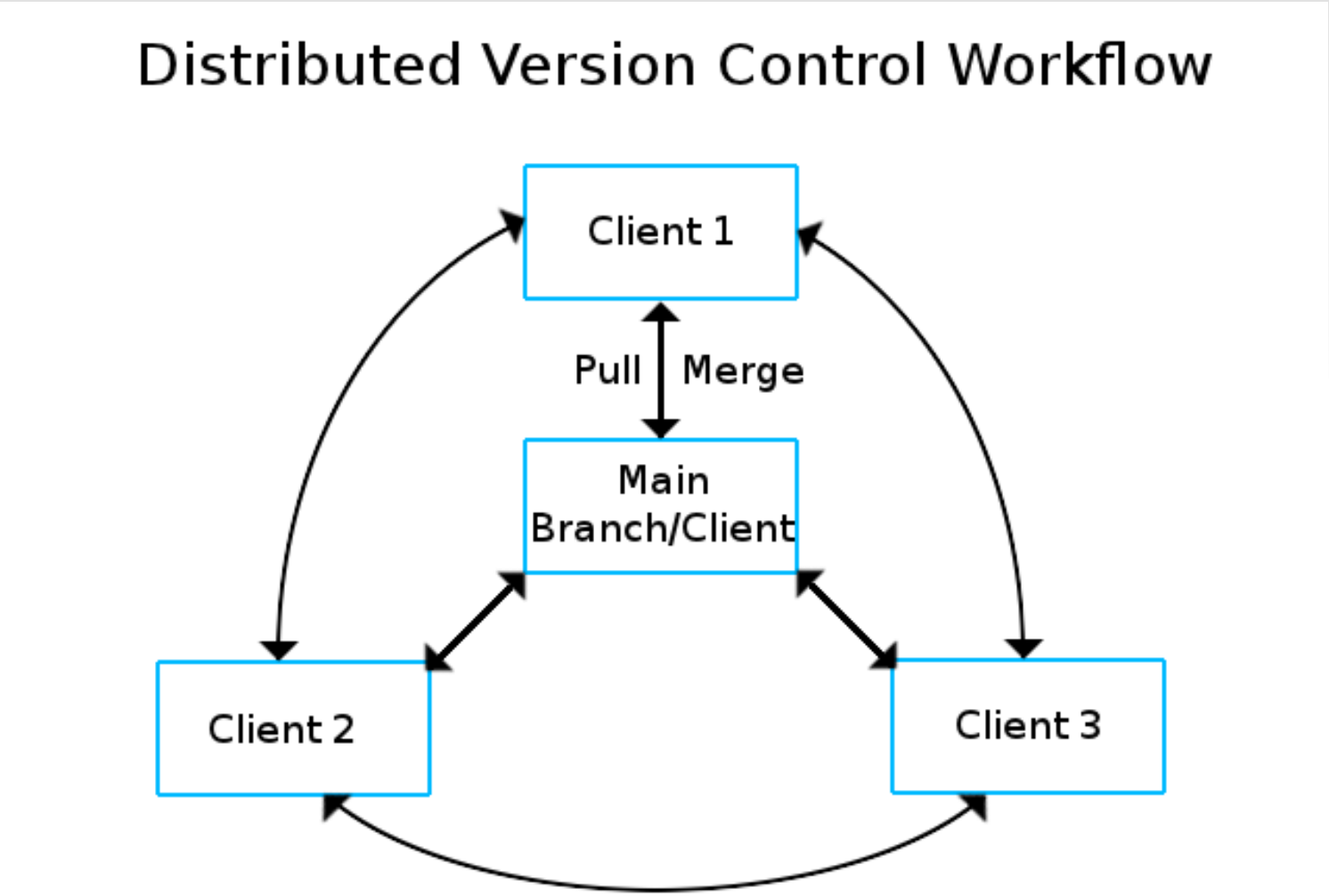
- In a decentralized system, your branch does not need a central server and keeps the entire code base locally unless specified not to

- Because there is no central or master in a decentralized system, your code does not need to be up to date with any other user to commit;

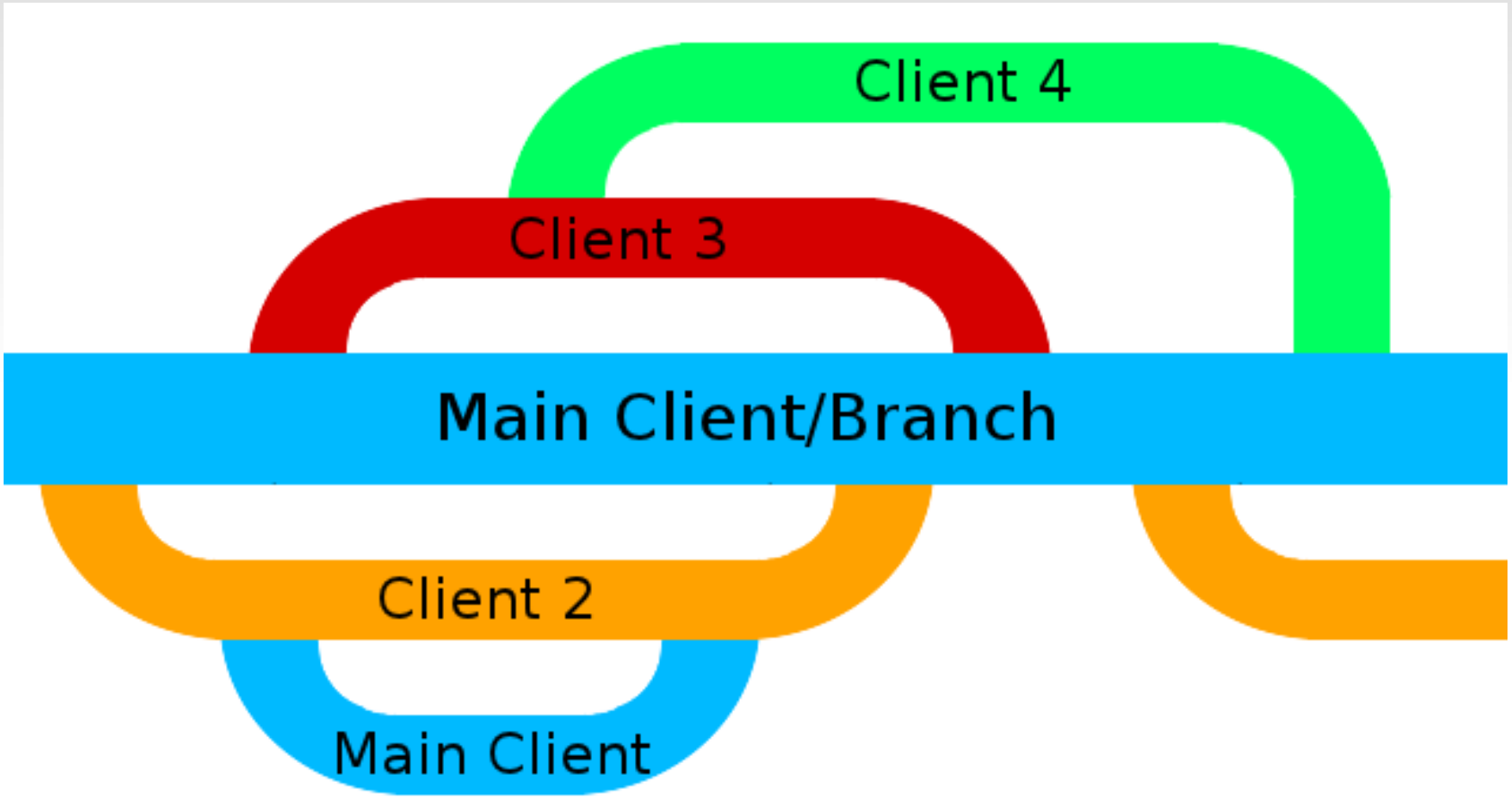
  Instead, decentralized systems use a merging technique that merges your code with another user's, and lets you modify any conflicts that may have existed

- It does not matter how out of date your code is, as the merging technique has a fast-forward option which lets you merge, push or pull from anyone with the same code base

Distributed Version Control



Distributed Version Control



Distributed Version Control Process

Distributed Version Control

- In a distributed version control, the maintainer(s) of the project usually have a master branch which they accept or reject code that was pushed to them;

- In an open source project, the master branch is publicly available and anyone can grab a copy to use for their own projects or the existing one

- They can code with others and merge together for the existing project or a new project;

  If modifying the existing code, the collective code can be pushed back to the master when ready

Distributed Version Control

- Advantages:
  - Group can set up virtually any kind of work flow model
  - No centralized system to fail
  - If using a centralized system or main branch, code can be restored by any user if the main fails (IE: hard drive failure)
  - Users can work with any other users directly and/or privately
  - Any number of branches can exist, including main development, testing, experimental, redesign or any other branch
  - Users may also have any number of branches on their local machines for the same purposes
  - Code can be updated/merged at any time, regardless of version

Distributed Version Control

- Disadvantages
  - The only real disadvantage to distributed version control is that it is more complex and harder to understand

Version Control

- Your projects requires you to use open source version control software
  - It is suggested you use distributed version control software (DVCS)
    DVCS:
    - git - http://git-scm.com/
    - bazaar (bzr) - http://bazaar.canonical.com/en/
    - mercurial (hg) - http://mercurial.selenic.com/
  - You may alternatively use subversion (svn) as a centralized system
    http://subversion.apache.org/

Project Planning

- Version control is only aspect of project planning, but it is a major aspect of the development process

- You also need to be able to plan ahead and manage the development process

  You need:

  - Planning
  - Tracking
  - Scheduling
  - Communication

Project Planning

- For planning and tracking you may want to use a free project organization website
  http://bettermeans.com
  http://trello.com
  http://corkboard.me
  http://asana.com

- Alternatively, you may want to just plan in more simplistic ways, such as Google docs or spreadsheets

- If you want to jump ahead in the course, you can use Launchpad or Github for tracking (bugs, issues, features, etc)
  *Note: You can plan with these also, but it is less clear

Project Planning

- For scheduling and progress tracking, you may want to use scheduling software like openproj
  http://sourceforge.net/projects/openproj/files/
  *Note: Look in binaries for Deb and RPM installers

- Again, you may want to take the classic approach and track scheduling and progress through Google calendar or some other system

Project Planning

- Finally, you will need to communicate very often for your project
- You are probably already using mailing lists to communicate, but it is highly recommended your use IRC as it is very fast and easy to setup for logging
- You may also want to consider Google Hangout, Ekiga (open source video/voip conferencing) or Skype
- Of course, it is suggested you meet in person as well, both for administrative and coding sessions

Project Planning

- Every one in the group should always blog about the results of their meetings, code sessions, developments, scheduling, etc
- Remember, just because another member in your group blogged about a topic does not mean you should not
- You need to blog your results regardless for this class
- Exposure is good; Employers will be impressed by your blog, and the more people blogging about your project, the more popular it gets