Software Development on Linux Systems

4002-XXX-XX

By

Cody Van De Mark

Today

- Publishing Process
- Code Hosting
- Bug Tracking
- Forking
- Merging

Publishing Process

- Publishing code as open source is slightly more involved than most would think
- In order to have an open source project, you need only a license, but also the availability of code
- Your code is not truly open source unless the source code itself is available

Publishing Process

- When you publish your project, you do not just give out the packages you built;

  You need to provide your source code, as well

- You also need to provide the licenses of other open source projects you used

- Depending on the license, you may need to actually deliver the source code of those projects too;

  As a rule of thumb, it is good to do this out of habit

Publishing Process

- Distributed packages usually come in a couple forms
  - Installable Package
  - Source Package
- Both of these packages include all documents, licenses and administrative material
- The installable package, however, does not usually have the source code itself packaged in it

Code Hosting

- After you have packaged all of the source code and licenses necessary in your package, you can begin to publish your work
- In order for your project to be open source, you need to distribute the code and packages
- In order to do this you need somewhere to host your code

Code Hosting

- There are a number of sites to choose from depending on how you want to host your code
- You may take the approach of hosting the code yourself on your own website
- There are many cost hosting sites that offer and encourage free open source code hosting

Code Hosting

- Open source code hosting sites support most of the main open source version control systems
- You may want to choose a site that supports your preferred version control software
- You may also want to choose a site that is geared towards where you want your code to go
- You are not limited to only site though, as you can host on as many code hosting sites as you would like

Code Hosting

- Popular Code Repositories
  - Github - Git powered
  - Launchpad - Bzr powered
  - BitBucket - Git, Mercurial powered
  - Gitorious - Git powered
  - Google Code - Git, Mercurial, SVN powered
  - Source Forge - Bzr, Git, Mercurial, SVN powered
  - Tigris - SVN powered

Code Hosting

- Github is very popular due to its social aspect, great collaboration tools and well designed UI

- Launchpad is very popular because of its strong business aspect and its incredible integration into Ubuntu

Code Hosting

- Code hosting is mostly similar to the version control software itself
- You interact with most hosting sites as if they were a remote version control repository
- After you create an account on your site of choice, you need to add your ssh key of your machine to the site
- This allows your machine and the code hosting site to communicate securely through your version control software

Code Hosting

- After you make a project on code hosting site, you will be given a url for version control that you can use on your machine
- This combined with your SSH key will allow you to connect and push projects
- Many code hosting sites will allow you to choose your license on the site so that users can see it without downloading your code
- Many code hosting sites have many other features, such as bug tracking, communications, forking, merging, social networks, etc

Bug Tracking

- As most code hosting sites have a bug tracker where you can accept and track bugs from users
- Any registered user can post a bug on your bug tracker
- Bugs have a commenting section where you and users can communicate details
- After you have received a bug, you can mark it as open or closed:

  You can also label the issue for tracking purposes

Bug Tracking

- Some code hosts, such as Launchpad offer you the ability to label bugs by how important they are, how close to completion they are and how popular the bug is
- Most major code hosts allow you to assign a bug to a particular user in your project group
- Many code hosts also allow you to link your bug to a particular milestone

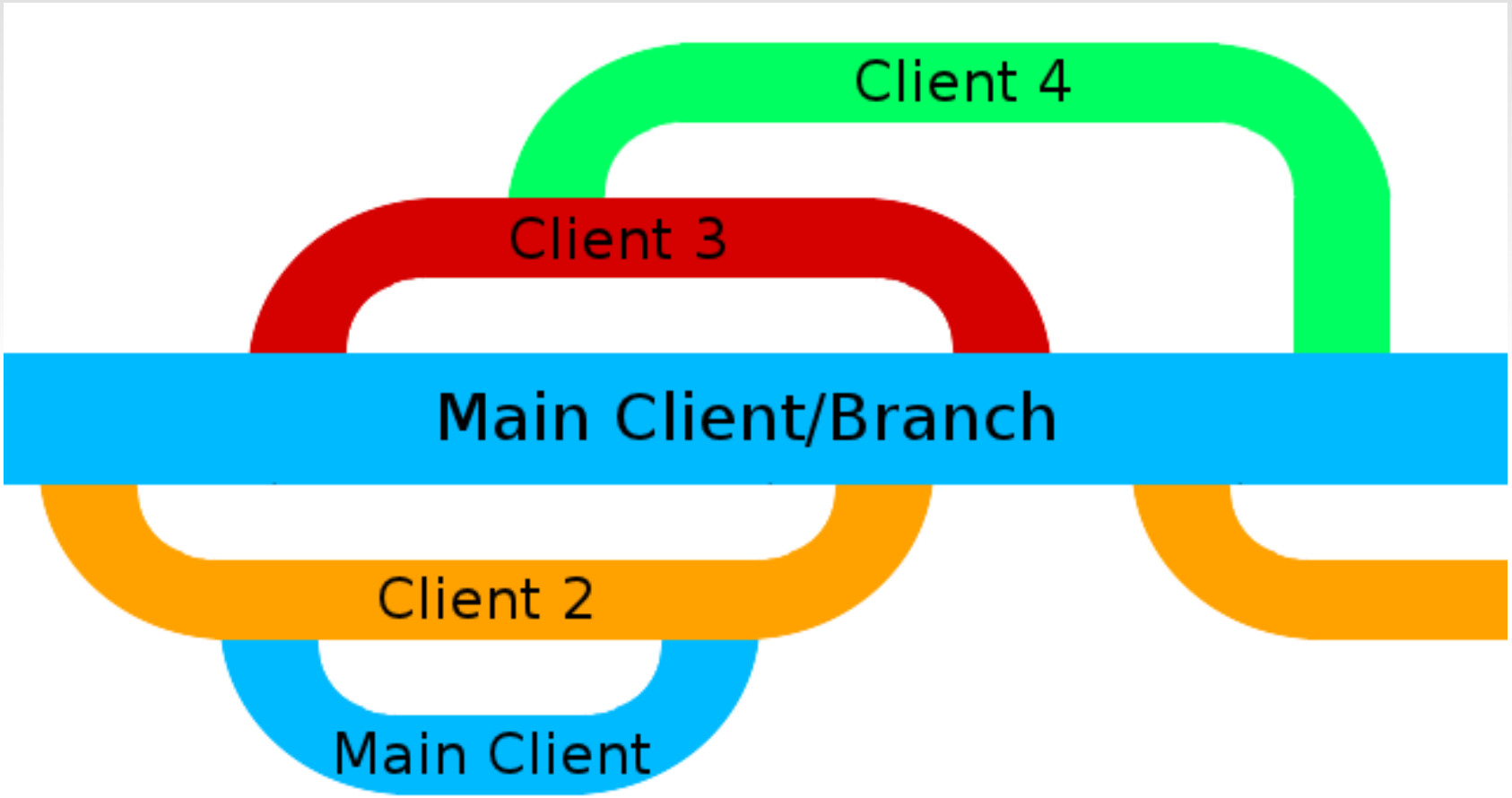
Forking

- When you fork a project, you are making a copy of the code base that you own and control
- Forking projects is extremely common in open source projects
- You may fork a project:
  - In attempt to fix a bug
  - To experiment with a new design/component
  - To take the project in a different direction
  - To integrate it into your own project

Forking

- Earlier in the course, we discussed distributed version control
- Distributed version control essentially relies on forking/merging
- When you “pull” a project in a distributed system, you are making your own branch and consequentially forked the code
- In this fork, you can modify the code however you want without affecting the main code base as it is a private branch

Forking



Distributed Version Control Process

Forking

- Forking code on code hosting sites:
  - Some sites offer a forking option on the site
  - Others require you to fork in version control

- Some sites, such as Github, have an option on the site to automatically fork a project into your own branch;

  This is done using a web interface that does the version control fork for you

- Other sites, such as Launchpad, require you to fork the project yourself using version control commands

Forking

- You may manually fork a project on Github and other services, as well
- Manually forking is the done by branching a project from a repository as your own private branch
- This is done with the “branch” or “clone” commands
- A repository you are branching from may be local, on a remote machine or on a code hosting site

Forking

- A branch is a fork within your own repository as it is still your code base;

  It is just a branch of your code base

- To fork, you are branching from another code base creating a “fork” as it is no longer attached to that code base, but your own new one

- A fork is essentially a new code base, and it may not ever merge back in with the original, but has the ability to

Forking

- Branching:
  - **Launchpad:** **bzr branch lp:projectName**
  - **Github will do this for you, but it can be done manually with**

git clone git://github.com/userName/projectName.git

Forking

- After you branch a project from a remote code base that is not your own, you have created a fork
- You may want to put it back up onto a code host for you or your team to access it
- To do this, you need to upload the code back to the code host under your own name
- This is done by pushing the code to your repository

Forking

- After you have a branch of the project you need to create a repository under your own name on the code host

- This is done with the push command

  Launchpad:
  bzr push --use-existing lp:~username/projectName/branchName

  Github:
  git push --mirror git@github.com:username/projectName.git

Merging

- When you merge a branch with another branch, it takes changes between the two branches and applies them to both
- This leaves you with one branch that has the code from both
- Because of this, there may be conflicts that version control detects between versions
- If there are conflicts, these will be noted telling you where they are;

  You need to manually look at the code and resolve the conflicts, then save the code

Merging

- After you resolve your conflicts and save the code, the code will be merged
- You do not need to merge again after you resolve conflicts, as your code has already been merged into one branch
- Merging is also extremely common in open source and is done as part of distributed version control
- Merging allows you combine code in two branches, whether they are in the same code base, or if they between a fork and the original

Merging

- Most open source development is done by forking code and merging the code back together

- Developers of a project, will fork a project from a code hosting site, make their changes and merge it back in

- Merging is also done to combine two projects that took different directions back together, though this takes more time;

  For example: If someone forked Firefox and made massive improvements, they could merge it back in with the real Firefox even though theirs might be out of date by the time they merge

Merging

- Merging is done with the “merge” command
- Branches you are merging may be local, on a remote machine or on a code hosting site
- After you merge two branches, you will likely want to push it back to your own repository or send a merge request to the original repository

Merging

- Bzr keeps track of the original repository for you, allowing you to merge in one of two ways

- Merge with original
  - **bzr merge**

- Merge with another bzr directory (you may not always have the original to merge with)
  - **Create another folder to branch the latest code from a project**
  - **bzr branch lp:do**
  - **In your branch**

    bzr merge ../path/to/latest/pulled/branch/

Merging

- Launchpad Merging:
  - Upload your merged code back to your Launchpad repository
  - To merge with another project, send a “merge request” through the site to the project you want to merge with
  - The person receiving a merge request will be given your project url
  - They can merge with your code with
    bzr merge lp:~yourUsername/projectName/branchName
  - They can then upload your merged code back to their original repo

Merging

- Git is similar to bzr in merging, but requires you to point back to the original code base

- Merge with original
  - **git remote add upstream git://github.com/*theirUsername/projectName*.git**

Merging

- Merge with another Git branch (you may not have the original to merge with)
  - **Create another folder to branch the latest code from a project**
  - **git clone git://github.com/userName/projectName.git**
  - **In your branch**
    - **git remote add project2 ../path/to/latest/pulled/branch/**
    - **git fetch project2**
    - **git merge project2/master**

Merging

- Github Merging:
  - Upload your merged code back to your Github repository
  - To merge with another project, send a “pull request” through the site to the project you want to merge with
  - The person receiving a merge request will be given your project url

Merging

- Github Merging:
  - They can merge with your code with
    - **git remote add username git://github.com/username/projectName.git**
    - **git fetch username**
    - **git merge username/master**
  - They can then upload your merged code back to their original repo

Merging

- For a user to push merged code back to a code host, they just need to use “push” as normal

- After the code has been merged:
  - **Launchpad:**
    - **bzr add ***
    - **bzr commit -m 'merged with username'**
    - **bzr push lp:~userName/projectName/branchName**
  - **Github:**
    - **git add ***
    - **git commit -m 'merged with username'**
    - **git push git@github.com:username/projectName.git**