Using Local Web-based Version Control System in Botball Teams
Frank Blackburn(Censored)
Censored

Using Local Web-based Version Control System in Botball Teams

Abstract: This paper discusses an effective way to collaborate within botball teams by using web-based version control application, including factors such as certain requirement analysis, workspace building and workflow training.

Keywords: Version Control System, Git, Botball, Web-based Application.

1. Introduction

In software development, using Version Control System (VCS) to improve the efficiency of collaborative coding and debugging is a common practice[1]. The basic principle of these systems is to record every change of each file and using these records to save all versions of the codes.

The usage of VCS can bring a lot of helps to a botball team, and the main advantages of them will be discussed as following.

(1) Tracking Code Changes

It is very common in a Botball team that somebody, sometimes intentionally or accidentally alters the source code of the robot. It will not be a big deal unless, which usually happens, everybody forgets the original structure or arguments (such as servo positions) of the code after the changes are saved. It will become a very complicated mess if no backups are made, however without version control system, backing up 10 - 15 source codes will be a very annoying job to do.

(2) Making Collaboration Much Easier

In our team, we usually do not need two members response for one file at the same time. The real issue VCS is solving is that when members who do not program is practicing only and problems related to programs occurred, this member can connect the VCS-hosting computer to the internet, and programmers in the team can alter the code directly and remotely. All the non-programming member need to do is copy the code, paste it into the KIPR IDE and run the program.

(3) Permission controls

Some naughty programmers often want to modify all unsatisfactory codes, and sometimes it will causes chaotic situations. Although our team is not as complicated as in a company, it

is still a very good idea to let some important code "read only" to particular team members.

2. The Selection of VCS

When we refer to "Version Control System", most of us will have git, a version control system created in 2005 by Linus Torvalds, in mind. However, there are actually a lot of free version control systems available. We looked through all of them, and we will perform a very quick analysis below.

(1) Apache Subversion(SVN)

Apache Subversion is a VCS developed by Apache Software Foundation in 2000, which is the oldest among all VCSs listed. Because of its client-server architecture[2], it is not easy to sync among different servers. However, this feature is needed since our computer can only connect to the internet for a limited time.
When tracking binary files (such as pictures and videos), SVN will do a better job than most other VCSs since it only record changes to the files instead of record the whole new file.

(2) Git

Created in 2005 by Linus Torvalds, git is probably the most famous VCS in the whole programming community. But since it's developed for open source projects, a lot of plugins or changed versions of it may be needed for permission control and other required features.

(3) Mercurial(Hg)

Mercurial is a VCS developed in 2005. Its workflow is somehow similar to git, and it's much easier to learn than git. It’s commands, in contrast, are very similar to SVN. However, it does not have much extensions, features and web-based control panels while we do need these.

(4) GNU/Bazaar

GNU/Bazaar is a VCS developed and mainly used by Canonical Ltd. Helpful for big programming teams, it unfortunately combines all the drawbacks git and mercurial have: it's both lack of extension and not very easy to learn.

After comparing all VCSs, we decided to use git as ours mainly for its extendibility, for example, we will need git lfs to store our multimedia, especially pictures (which often updates) used in our documents. Also, because most of our members does not have experience of using command-based VCS, we decided to stay on a web-based system.

3. Using the VCS

In previous GCER conference papers, there are practices of using online version control service providers, which is, as these papers all used, GitHub[3].

3.1 Analysis of the Situation of Last Year

Our team used GitHub as version control system last year (team 17-0553), but the result is not satisfactory. Therefore, our new team analyzed the reasons why GitHub did not work well in our team last year.

1. GitHub needs internet access

GitHub needs internet to commit changes, and if changes are not committed on time, it will often be forgotten and become chaotic when the process is finally remembered to do.

Unfortunately, our lab does not have direct internet access. We either have to use a very long ethernet cable to connect our router in the lab on third floor to the ethernet switch on the first floor, or have to open use our cellular network, which can be considered expensive.

2. GitHub does not have localization

Although our team does have fluent English speakers, there are still members, and mostly programmers, have limited English skill. Since GitHub only supports English, learning curves become steeper for them, making them unwilling to use git.

3. GitHub cannot be accessed sometimes

Because of some serious and mysterious reasons, GitHub sometimes cannot be accessed in China. This event seldom happens nowadays, but our work may be discontinued if it happens.

Also, although GitHub offers free unlimited private repository plan to students (yes, even for middle school students), it cannot be applied to organization, which raises the difficulty of permission control.

3.2 Our New Solution

Since we do not have internet access, a local server with web-based VCS will be a better solution. Furthermore, it will be much better if it has localizations.

After several days of investigation and comparison, we found that GITLAB is our best choice. Since GITLAB is fully customizable with partial official Chinese translation and full community translation, it is very suitable for non-native language programmers like us.

The main usages and steps of GITLAB will be introduced as following:

3.2.1 Environment Setting

Since the computers in our lab are prohibited from running virtual machines or install multi operating system for some management reason, we have to build our own servers with Intel® Core™ i3-2130 Processor. Fortunately, our lab has 7x24 hours of power supply so that our server can provide uninterrupted services.

As for the system, we choose Ubuntu 16.04 LTS Xenial Xerus for its reliable stability and active community. When this paper is written, it has been updated to Ubuntu 18.04 LTS Bionic Beaver.

Installation process of GitLab is easy - just follow the guide on the project website. Remember to set the EXTERNAL_URL to 172.0.0.1, which, practically, means localhost. Access GitLab at 127.0.0.1using browser, then do all the set-ups: admin password, language, login pages, etc.

Finally, add accounts for all members in the team. Since registration requires e-mail confirmation while this server does not need send e-mails.

Fig3.0 Sign in page of the System

3.2.2 Basic Workflow

3.2.2.1 Groups

In the course of the competition, each team has 3 repositories, each for Create, Wallaby robot and documents contained in one group, and members are told to update all files using internal editor after every major change of the code or after end of every day's work if codes are changed. We can clearly see the changelog using git diff tools.

Fig 3.1 Updating Function Library

3.2.2.2 Permission Control

In each repository, editing permissions are granted only to the members who are

responsible to certain repository. If some changes in repository that somebody who's not in charge needs to be proposed, giving advice to those who is in charge or creating pull request is encouraged.

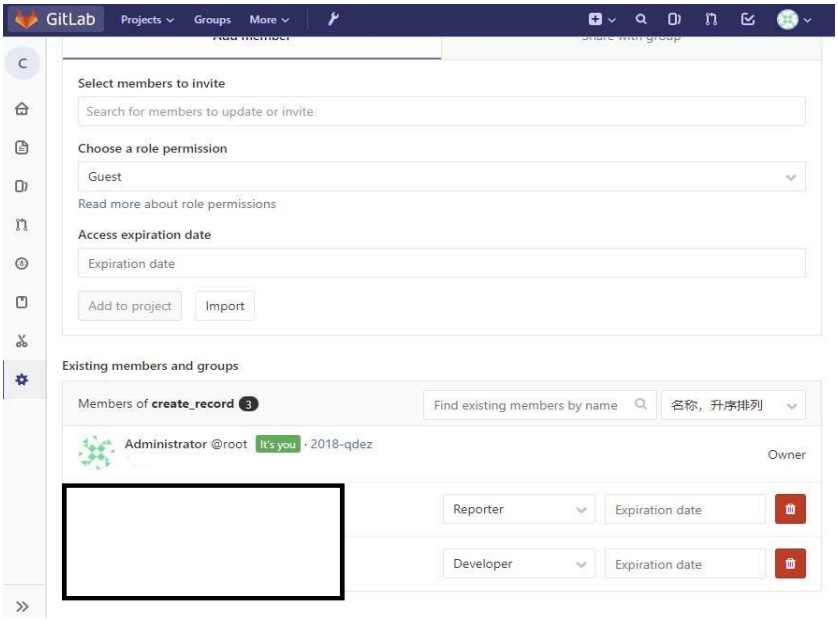

Fig 3.2 Option of Permission Control

3.2.2.3 Documents

GCER documents are told to write in GitHub Flavored Markdown, a lightweight markup language that can be rendered very easily using online renderer and also very easy to learn. It is also simple to convert them to PDF format that GCER requires.

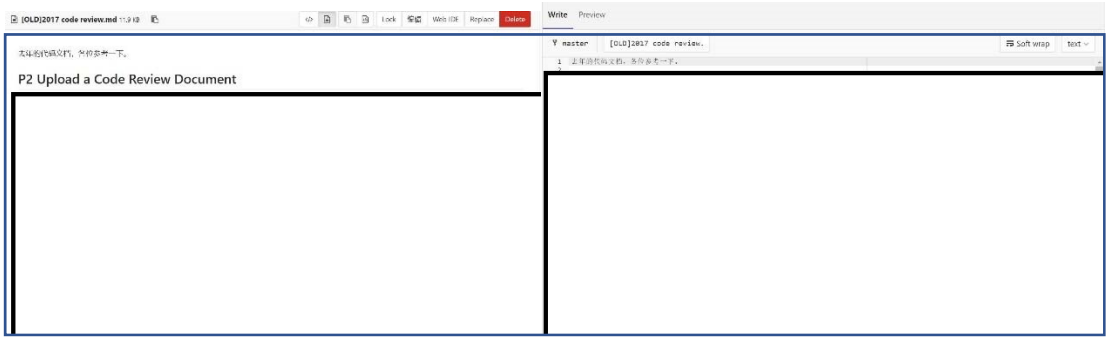

Fig 3.3, 3.4. Code review document last year

3.2.2.4 Multimedia

As for binary files and multimedia, we use FTP to save them and use an automated task to add them to our repository using git lfs so that they will not waste spaces.

3.2.2.5 Cooperation

When in lab, cooperation is not very common since usually only one member is programming one robot. However, it is very important when cooperation is done remotely,

which means, some members may edit codes when they are not in lab.

The underlying concept is that this router has configured DDNS feature, so that every time it connects to the network, a certain web domain can allow us access this subnet via Internet. This way, we can access GitLab anywhere in the world when the Internet is connected.

We also connect an Ethernet cable to the server at the start and the end of every weekend and national holiday. The server will automatically push all repositories to a remote one (currently hosted on Visual Studio Team Services). Changes can be made during holidays on the remote servers, and after ending of holidays, the server will pull the changed (or sometimes unchanged) repositories so that they're up to date[4].

Here's a simple demonstration of the workflow:

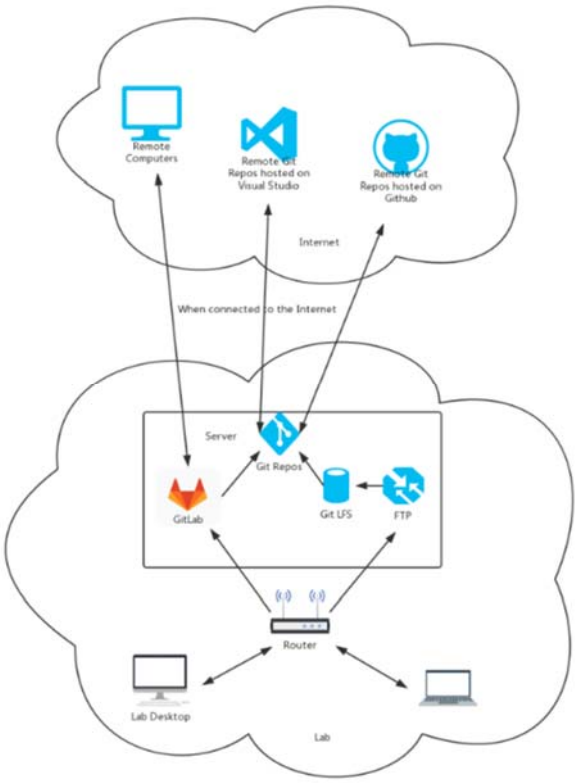

Fig 3.5 Structure of Workflow

4. Future Works

Wallaby is a customized Linux system, so it is possible to install git or nginx on it. We are currently trying to do so on one of our wallaby controller to see if integrated version control works. We are also wondering whether KIPR IDE can integrate git into it.

In a nutshell, using version control system makes tracking code changes simpler, makes cooperation within a team easier and makes the whole project much more organized if members are willing to learn some version control knowledges.

Reference

[1] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, "Collaboration Tools for Global Software Engineering," *IEEE Software,* vol. 27, no. 2, pp. 52-55, 2010.

[2] B. Collins-Sussman, B. Fitzpatrick, and M. Pilato, *Version Control with Subversion*. O'Reilly Media, 2004.

[3] L. Butler and L. Dubishar, "SourceControl," presented at the Global Conference on Educational Robotics, Georgetown University, Washington, DC, 2015.

[4] J. M. Hethey, *GitLab Repository Management*. Packt Publishing, 2013.