JAN 1, 2018

GitHub / git – Eclipse overview and integration for FRC teams.

GitHub is an invaluable source control and collaboration tool for FRC teams. Below are some explanations on what GitHub and git are, why they should be used by all teams and how to use them. The how portion of this document will be broken in multiple topics each building on top of each other, starting with the basics.

What is git and Github?

- Git is a version control system created by Linus Torvolds along with other Linux kernel developers in 2005. It was created to provide an open source alternative to the proprietary system BitKeeper and to address issues with other open source version control systems. Key benefits of git, distributed workflow, corruption resilient, fast commits and self-contained repositories in every directory under git control.
- GitHub – A commercial provider of online git repositories. Free services are provided by GitHub that meet most non-commercial users' needs. Commercial version are also available if desired.

What is a version control system, AKA VCS?

- A system that provides services useful for the storage, distributed development and versioning of source code.
- Version control systems, track changes made to source code files and provide methods for storing those changes either locally, remotely or both. This tracking allows for changes to be backed out if needed, provide details on what the change contains or allow multiple people to change the same source files and have the system be aware of this and provide for automated ways to "merge" the changes or provide feedback to the users notifying them that issues need to be resolved manually. Along with tracking comes other useful features such as tagging, which allows all code at a certain point and time to be labeled and recalled later in the state it was at the time of tagging. This is useful for "releasing" code to be used for general consumption. There are numerous other benefits to a version control system, more of which we will cover later in this document.
- Version control systems typically store code in what's called a "repository". Think of it as a safe deposit vault that includes accounting and ledgers for your source code.

Why use git?

- Git provides a well tested, well documented, very capable and widely used version control system.
- Although any version control system has a learning curve, git has many similarities with other version control systems and as such, if you are familiar with CVS, BitKeeper or SVN, learning git will be relatively easy.
- Change tracking and commenting. The primary reason for using a version control system is typically the ability of the system to track changes, along with comments made when those changes are added to the system. This tracking allows a user to revert, share and compare changes between branches, tags, forks, etc... The act of tracking changes occurs when a user "commits" his changes into the repository. Good commit practice also includes a high level overview of the changes.
- Multiple code branches. Branching code is a common practice that allows multiple independent changes to be made to a common code base without interfering with each other. The analogy for this is a tree, with a common code base making up the "trunk" with branches being made off the trunk which have a common ancestor in the trunk and so share that starting code base, but each branch has its own unique changes that are stored only in the branch and thus allow the trunk code to remain static while allowing branches to test out new fixes, features, optimizations, etc. Changes in a branch can be merged back into the trunk or the branch can be destroyed leaving the trunk in its original state. Development can continue in trunk after branching, but changes in the trunk do not automatically make it into the branches and vice versa, changes must be merged between the branches and trunk. Changes can be merged between branches as well.
- Tagging code releases. Tagging code is the practice of labelling a code base, typically in the trunk, with a label. This label can then be used to see all the code as it was at the time of tagging. Tagging is typically done when releases are made. But the practice of tagging can be done at any time on any branch for any reason.
- Forking code bases is a GitHub feature and is documented later, it's noted here for reference since it was mentioned above.

Why use GitHub?

- Centralized git repository with a very capable web interface and access controls.
- GitHub repositories can be “forked” meaning one person can make a their own copy of any GitHub repository they have access to. This fork can then be modified, branched, tagged, etc… like any other git repository. Changes between forks are typically done via “Pull requests”. Though changes between forks can also be done via merges, just like changes between branches can be done. A pull request, aka PR, is a request from someone to have the changes they have made to their local fork of a repository, incorporated into another repository, typically the source repository for the initial fork. This is a very powerful feature of GitHub as it allows anyone to make a copy of a repository, make changes to it, and then have those changes pulled back into the source repository. This is the typical method of development for community open source projects. Eg; Linux kernel
- Teams are another way to have multiple individuals work on a common code base. A team in GitHub is a set of individuals with direct access to the source code repository. These individuals will “clone” the source repository to a local location. The individuals in a team can also directly modify the original GitHub repository.
- The main primary difference between Teams and forks are; access to the code, what code is modified when changes are committed and the act of getting their changes inserted into the original GitHub repository.

Useful terms not described above.

- Cloning is the process that copies the contents of the source repository to a local location. Changes are then made to the local code base and save to the local repository via a “commit”. The commit saves the changes to the repository directly responsible for the source, in this case that repository is the local repository. You can also make changes directly via GitHub and those commits would then be made to the GitHub repository. To have local changes become part of the GitHub repository they must be “pushed”. The act of pushing, moves all the change made in a local repository to the remote repository, in this case, GitHub.
- Pulling is the opposite of a push, it “pulls” all changes made in the remote repository, in this case GitHub, to the local repository.
- Commit, push, pull and PRs are all intelligent processes and will only act on deltas between the various repositories.

Year Round CSA Contact info: firstmn.csa@gmail.com Slack: firstmncsa.slack.com

First MN Website: http://mnfirst.org/first-community-resources/local-assistance/

First MN CSA Github: https://github.com/firstmncsa

First MN Google Drive: http://goo.gl/STtiAg