SAI CPD event - Version Control

last updated: 2020-12-04

Table of Contents

# Introduction to Version Control

## Introduction

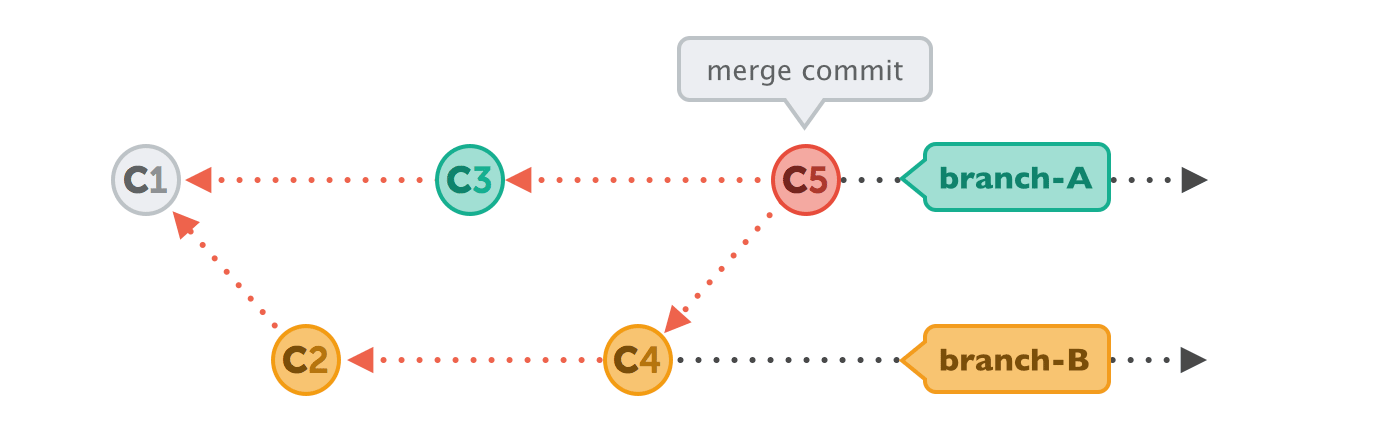
During the session we will provide an introduction to:

* What version control systems are and how they can be used by Actuaries
* examples of version control systems
* concepts of Local and Remote hosting
* user interface options
* commercial considerations

We will then provide an explanation of some key terminology before showing how a version control system can be used in practice (using Git, Github and RStudio for demonstration purposes)

## What is a Version Control System?

* Manages the evolution of a set of files (a repository) in a structured way
* Complete history of content is tracked and available
* This content may include model source code, validation testing code and documentation
* Facilitates collaboration as can manage many different workflows simultaneously
* Supports agile working as users can easily test, fix, undo or accept changes



Git is a Directed Acyclic Graph (DAG)

source: <https://i.stack.imgur.com/yE1kA.gif>

* Using the DAG data structure arrows point to the preceding commit, not the subsequent one
* Branches are labels on commits
* Commits can be merged to incorporate changes from more than one branch

## What is a Distributed Version Control System (DVCS)?

DVCS is a type of version control system with the following characteristics:

* Each user has a local project history called a repository
* A single remote repository exists which is the source of truth (master)
* Users can work offline since they have their own local copy of the project history
* DVCS allow users to ‘pull’ content from and ‘push’ content to the master copy

## Benefits of using a Version Control System

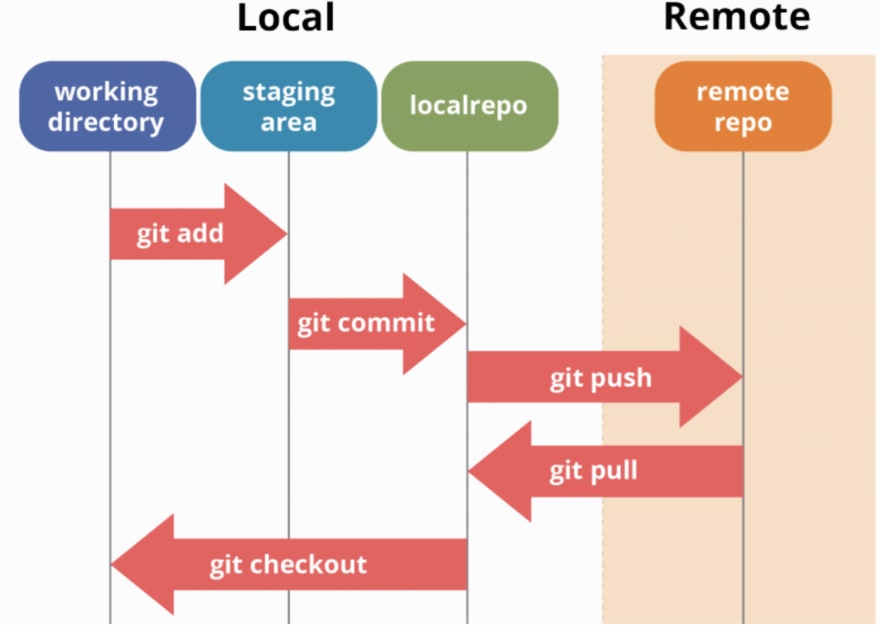
* Facilitates continuous improvements of development projects
* Allows professionals to share code with others and make changes which can be tracked
* Manages source code history
* Maintains a history of what changes have happened
* Supports collaboration and effective communication between professionals (e.g. Actuaries, Data Scientists, Software Developers)
* Facilitates multiple people working at the same time
* Helps code hygiene and review processes. Different users can have different permissions -e.g. only senior team members can merge code into main branch
* Git can help individual users improve their own work flows, even if not interacting with others

## Examples of Version Control systems

* Git (Global Information Tracker): Open source Distributed Version Control System Can be easily integrated with other technology (e.g. R studio)
* SVN (SubVersioN): Open source Centralized Version Control System
* CVS (Concurrent Versions System)
* Mercurial

## Local and remote hosting

The following diagram outlines a project structure

 source: <https://dev.to/mollynem/git-github--workflow-fundamentals-5496>

* Here we discuss Local and Remote hosting in context of the Git, a distributed version control system. The SVN version control system is different as it is centralized from the start

Key files include

* Working tree: Directory and files of a single version of the project (called a commit). Here you can view and edit files of a project
* Staging area: List of files ready for the next commit
* Local repository: Contains all of the commits that have been made for the project
* The working tree, staging area and local repository together form the project directory
* Remote Repository: Central copy of repository, located in a data center or in the cloud, contains the commits of the project and is the ‘source of truth’
* Workflow process includes the following steps:
  + copy down central copy of remote repository to your local machine (pull)
  + make changes/edit/save changes(commit) using git for this (iterative process), often several commits
  + when ready upload changes to central copy (push)
* Branch workflow practices
  + All work for a specific self-contained task in a branch so isolated from other changes
  + when ready for code to be moved into main branch then push and do a *pull request*
  + code reviewed -> merged into master
  + can implement software control practices here like testing, code review continuous integration - obviously depends on your job.
  + even for a single user, branch + pull requests + merge can be useful - easy to roll back a bad change, contain tasks

## Examples of Repository hosting services

Github:

* Web based hosting service for repositories (Git or SVN)
* Free option available. Private repositories have recently been made available to individuals and organizations

Other Repository hosting services include:

* Bitbucket (also offers free repositories)
* Gitlab
* Assembla
* Beanstalk
* Sourcetree

## Interacting with Version Control Systems

* Command line: requires command line skills
* User Interfaces: can be more user friendly
* User Interface examples for Git include Tortoise (also works with SVN), Github desktop (if using Github), Git-gui (Gits own built in user interface)
* A user can also interact with Git via RStudio, SAS Enterprise and Visual Studio

## Commercial considerations

* Security if using cloud based service like github
  + data security/privacy - need to ensure things that should not be uploaded do not get uploaded

# Some Key Terminology

## What is a Commit?

Git manages versions of a project. Each version of the project is called a Commit Each commit is a snapshot of the entire project Git is efficient at storing commits. For example, if a project contains 50 files and only one file is changed, the total number of files stored by git will be 51 i.e. each unique file is stored only once. Collection of commits contain the history of the project allowing a user to review the history and undo a change.

## What is a Branch?

All commits belong to a branch. Branches are independent lines of development of the project. The default is the master branch. Independence of branches allows teams to scale their work, with different people working on different parts of a project. For example, a project could be in production with steps A and B completed and in the master branch. At the same time one team member can independently be developing step C and another team member independently developing step D.

## What is a Pull Request and Merge?

Pull requests allow the user request that their development work is merged with the master branch. A merge combines the work of separate branches. During a pull request team members can review and approve your changes. If pull request is accepted then your changes are merged and become the latest changes in the master branch.

This facilitates strong governance of model updates.

# Demonstration

## Introduction

For demonstration purposes we will use the following:

Version Control System - Git Repository hosting service - Github User Interface - R Studio

Items to be covered in demonstration include:

* Getting set-up
* Cloning a Git Repository
* Create a script in R, then Commit and Push to Repository Master branch
* Pull from Repository Master branch, make edits and Push to new branch
* Review and accept or reject edits
* Rename and Delete files
* Recover files

## Getting set-up

* Downloading Git
* Obtaining a Github account
* Creating and Cloning a Repository in Github
* Interacting using R Studio

## Branches

* Demonstrate how to create a new Branch

## Commit

* Demonstrate how to make changes to code in R Studio within your branch, stage and then commit them
* This is done locally

## Push

* Demonstrate how to push the above changes into the remote repository

## Pull request and Merge

* Demonstrate how to create a Pull request

## Review and Accept or Reject

* Demonstrate how to Review and Accept or Reject updated prior to merging

# Appendix

## Some useful links

Different Version Control Systems

* <https://www.atlassian.com/git/tutorials/what-is-git>
* <https://www.perforce.com/blog/vcs/what-svn>
* <https://svn.apache.org/repos/asf/openoffice/ooo-site/trunk/content/docs/ddCVS.html.en>

Directed acyclic graph explained

* <https://medium.com/girl-writes-code/git-is-a-directed-acyclic-graph-and-what-the-heck-does-that-mean-b6c8dec65059>

Downloading git:

* <https://git-scm.com/downloads>

Overview of common user interfaces for git:

* <https://git-scm.com/downloads/guis>

Comparison of different Repository Hosting Services

* <https://www.git-tower.com/blog/git-hosting-services-compared/>

Creating a branch on Github

* <https://docs.github.com/en/desktop/contributing-to-projects/creating-a-branch-for-your-work>

Merging a branch on Github

* <https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/merging-a-pull-request>