SAS® GLOBAL FORUM 2021

**Paper 1021-2021**

# Git and SAS®: A Match Made in (SAS®) Studio

Joe Matise, NORC at the University of Chicago

## ABSTRACT

Once upon a time, version control was something for programmers who lived on the command line, and required remembering esoteric commands and complex sequences. Thankfully, this is no longer the case - with the latest version of SAS Studio, version control using Git is now easily accessible for all!

We will cover the basics of using Git integration in SAS Studio, from how to set up a repository, how to connect it to your company's server, how to commit changes to your code, and how to view past versions of your code.

For those of you who are seasoned Git pros, we will show how Git integration in SAS Studio can save you time, making for a more concise and smooth workflow while maintaining your ability to use Git outside of SAS Studio for those more advanced operations you need it for.

This presentation is aimed at programmers at any level who either know nothing at all about Git, or who would like to learn how to use the built-in Git functionality in SAS Studio. The presentation will use SAS Studio 3.8 with Base SAS 9.4 TS1m7, and will also mention the differences with SAS Studio for SAS Viya version 5.2. No prior knowledge of SAS, SAS Studio, or Git is required.

## INTRODUCTION

Are you a programmer, who is looking for an easier way to use Git to manage the code for your SAS applications, either alone or as part of a team? Or are you an analyst who's looking for an easy way to back up your SAS programs and easily see what's changed in them? Or perhaps you are a SAS administrator, looking for a way to get your users using Git without too much trouble.

In the following pages, we'll cover how a user can connect SAS Studio to your local Git repository as well as your centralized Git server. This simple, step by step method will get you running in Git in minutes, as well as showing you some simple Git workflows for turbocharging your SAS Studio programming experience!

## WHAT IS GIT?

Git is a "free and open source distributed version control system" (https://git-scm.com), developed in 2005 by the creator of Linux, Linus Torvalds, initially to manage the source code for the Linux kernel. Today, it is used for two main purposes: archival/backup, and collaboration.

### ARCHIVAL/BACKUP PURPOSES

Do you have lots of versions of your programs sitting in a folder somewhere (maybe called "archive" or "old"), each with a timestamp after the program's name? Are you worried that you might make a change that breaks something, and not be able to find out what happened? Git simplifies this process, allowing you to essentially save copies of your code however often you want – all with a single filename, and with an easy way to see the differences between versions of a program.

### COLLABORATION

Whenever you have multiple people work on the same project – no matter how big or small the project is – you inevitably run into conflicts. You have to call down the hall – or, now, grab a Zoom room – and make sure nobody is working on the program you're working on. Or worse, you make changes to something, save it, and then someone else makes changes to the original and saves that – overwriting your code! Just as bad, you might have an outdated version of a program in your local copy, if you use that route; then you have to go in and figure out what you changed from the original, when you update the new copy.

Git fixes that through the use of branches, which allow users to each have a separate set of files to work on; when your changes are ready to be combined, you can merge your changes to the master files, with Git automatically telling you anywhere that you changed a file someone else also changed. Git also simplifies the process of ensuring you have the most up to date files possible; before starting work on a new part of the project, you can query the server and ask for the most up-to-date files be pulled down to your local folder, and be confident you have the right code to work on.

### DISTRIBUTED, YET CENTRAL

Git differs from older source control methods in that you store your changes on your local folder, in a *local repository*. For some users, that local repository might be literally on their own machine; for other users, the local repository might be in a network file share (for example, if you are running SAS Studio, your server most likely cannot see your local machine). Either way, the storage is local to the code – usually in a file that resides in the same folder as the root folder of the project – and usually accessible only to that user. The local repository keeps track of all of the changes to the code, and should be updated frequently – ideally several times a day, any time a meaningful change is made.

How do users collaborate, then? Git uses a central server to store a copy of the repository. This server, called a *remote repository*, hosts a copy of those versions that its users have decided to share with each other. This remote repository might be updated every few days, or less or more often depending on your collaboration schedule. This process requires authenticating to the server, sending it a copy of your changes, and asking the server to verify your changes do not conflict with anyone else's code changes that have happened since you pulled the data down last.

## HOW CAN I USE GIT WITH SAS?

The SAS system supports Git in many different ways. SAS Studio, SAS Enterprise Guide, and many other tools support Git functionality, as well as the SAS language itself!

### USING GIT IN THE SAS LANGUAGE

Git functionality was added to the SAS language officially in SAS 9.4 TS1M6, and further enhanced in Viya 3.5. 9.4m6 introduced the GITFN family of functions, which allow a user to perform most of the typical Git operations in the SAS programming language. Using functions like GITFN_COMMIT, GITFN_PUSH, and GITFN_PULL, a user can directly manipulate their Git repository from within their SAS code.

These functions are documented in the SAS Programming Documentation for SAS 9.4/Viya 3.5, in the "Functions and CALL Routines" section. There is a page titled "Using Git Functions in SAS," which gives several examples of using the functions for typical Git operations, as well as pages for each individual function.

Users of SAS Viya 3.5 will note that there are a new set of functions, prefixed with "GIT", that replace the "GITFN" functions. SAS 9.4 users still are asked to use the "GITFN" functions, and at the present time it is unknown if the GIT functions will be ported to SAS 9.

### USING GIT IN SAS STUDIO

As of SAS Studio 3.8 (SAS 9), or SAS Studio 5.2 (SAS Viya), this functionality is also built in to SAS Studio's user interface. Using the GITFN functions behind the scenes, you are able to perform most typical Git operations with a few clicks of your mouse. Users are free to mix and match using the Git functions in their code and interactively – they work in the same manner.

The remainder of this paper will cover setting up the point-and-click Git functionality within SAS Studio, and performing basic operations with it. Examples and screenshots in this paper are from SAS Studio 3.8 (SAS 9.4 TS1m7); SAS Studio 5.2 (SAS Viya 3.5) will be similar, but not exactly identical, and users should be able to follow the same basic steps – sometimes with slightly different screens.

## PREREQUISITES FOR SETTING UP GIT IN SAS STUDIO

Before you use Git in SAS Studio, you will need some information.

### YOUR GIT SERVER

You will first need the address of the Git server you will be pushing to and pulling from. For some users this will be Github, Gitlab, Bitbucket, or another public hosting site; for others it will be an on premises solution (often still using one of those major providers).

### HOW TO CONNECT TO YOUR GIT SERVER

You have two choices for how to connect to your Git server: SSH or HTTPS. Which you choose will depend in part on how your Git server is set up, and in part on your preference. HTTPS is a bit easier to set up; some people prefer SSH, particularly in production setups, due to the ease of authentication with an SSH key.

### YOUR CREDENTIALS

If you are using HTTPS, you likely already have your credentials – your username and password. If you are using SSH, you will need to generate an SSH key if you have not done so already. How to do this will be addressed in the section on setting up Git with SSH.

## SETTING UP GIT IN SAS STUDIO WITH HTTPS

To set up your Git profile in SAS Studio using HTTPS, you first need to create a Git profile. You can do this in the Preferences dialog within SAS Studio. Select the Git Profile option from the left-hand menu, and select "Switch to HTTPS". This will bring up a window like the picture below.

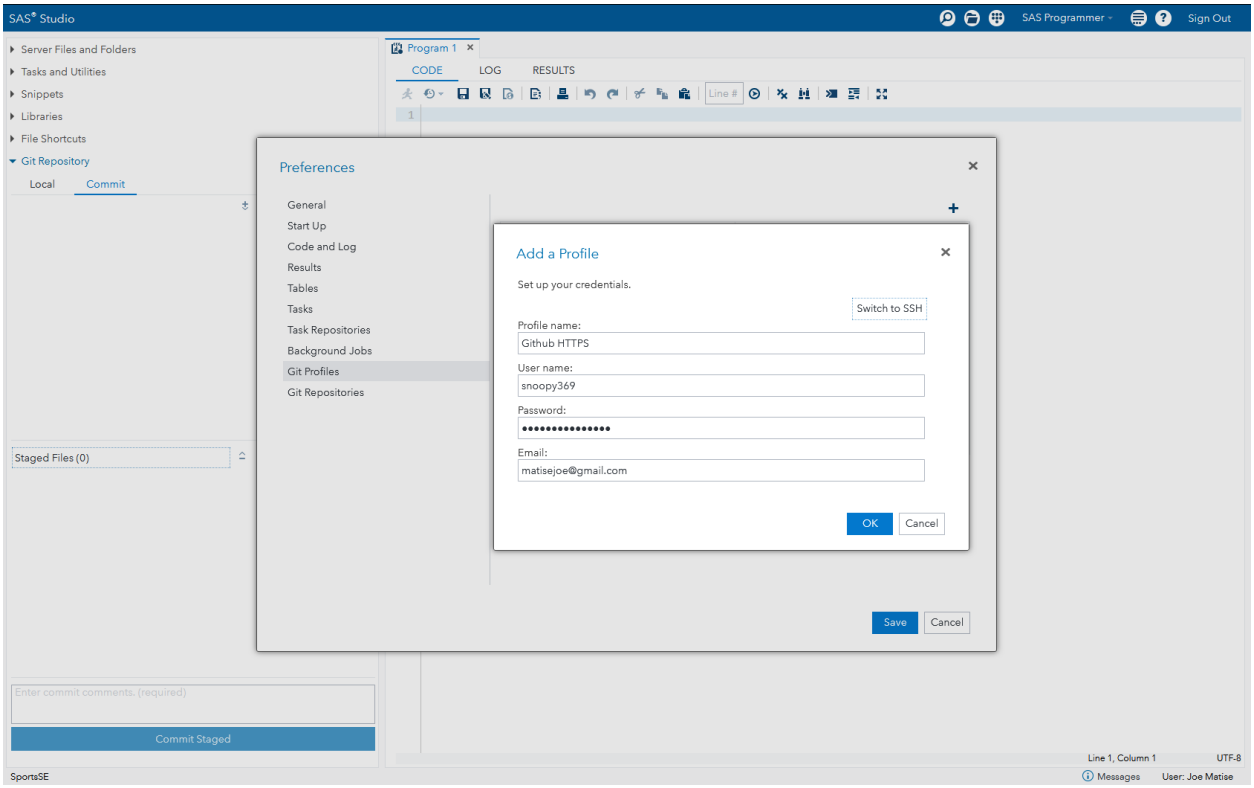


**Figure 1. Add a Profile window: HTTPS**

Fill in the information requested – your username, password, email address, and give the profile a name. You will note that you do not enter the server name here – you will enter that information later.

Once you do this, you are ready to get going. You may need to re-enter your password later if it changes, depending on how your system is set up, so be aware of that if you are using your corporate account and have forced password changes – don't lock yourself out!

One note: if you do not see the "Switch to HTTPS" option here, you should contact your SAS administrator. Early versions of Studio 3.8 did not have that option enabled, and may require a hotfix as well.

## SETTING UP GIT IN SAS STUDIO WITH SSH

If you are using SSH, your first step is to generate an SSH key. To do that, you will need a program that can generate an RSA key, called ssh-keygen. Most Linux/Unix distributions include ssh-keygen; if you are using Windows, you can get that a few different ways, but the simplest is probably to download Git for Windows, which includes ssh-keygen in its distribution.

When you have that program located, you need to run it with the following parameters:

```
ssh-keygen -t rsa -b 4096 -C your@email.address -f "output_file"
```

The program should then ask you to provide a passphrase. In the current version of SAS Studio, you must press enter to generate the key without a passphrase. This is not ideal security-wise, and SAS is aware of this issue; hopefully it will be addressed in a future release.

Your screen should look something like the below figure:


**Figure 2. Generating an SSH Key**

Don't worry, the key above is not my actual key. When you generate yours, treat it like you would treat your password – do not share it or place it anywhere that another user could get to it, as they could impersonate you with it. Place the two files it produces in a safe location that SAS can see, but preferably no other user can.

Once you have your SSH key, head over to SAS Studio, and open the Preferences dialog. Select the Git Profile option from the left-hand menu. This will bring up a window like the picture below.

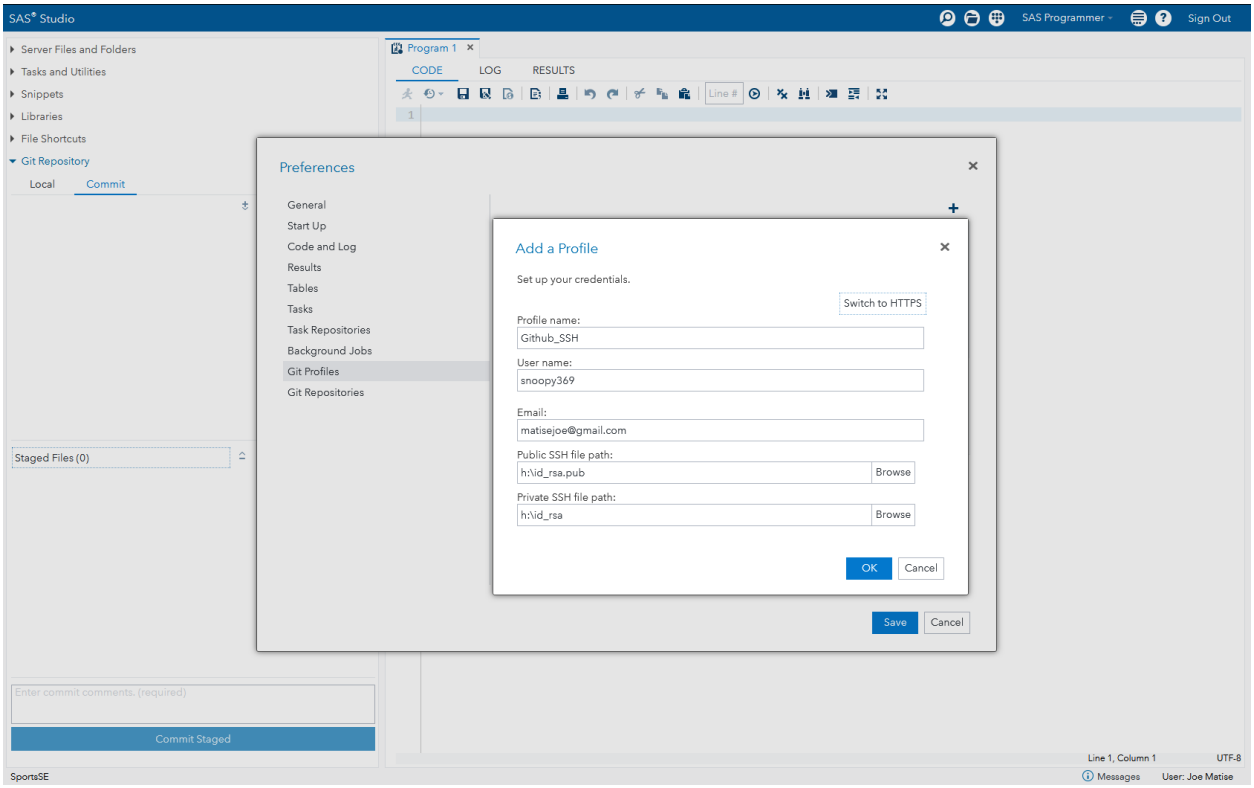


**Figure 3. Add a Profile window: SSH**

Fill in the information requested – your username, email address, and the location of the two keys (the Public key will have a .pub extension by default, the Private key will have no extension by default), and then give the profile a name. You will note that you do not enter the server name here – you will enter that information later.

Once you do this, you are ready to get going.

# BASIC GIT OPERATIONS IN SAS STUDIO

Once you have your Git profile set up, it's time to hook up your first repository! Most commonly, you will start within your Git server interface (your web browser, or a Git client program like Sourcetree). From here, you will want to browse to the repository you want to use in SAS. Then, you should look for the option to clone that repository.

## CLONING A REPOSITORY

Cloning is the process by which you take an existing repository from your Git server and make a local copy of it. You'll be looking for a link of the type you set up your profile for – SSH or HTTPS – most likely looking something like this:

`https://github.com/snoopy369/presentations` (HTTPS)

`matisejoe@github.com/snoopy369/presentations` (SSH)

You will then come back to SAS Studio with that link in your clipboard. In SAS Studio, there should now be a "Git Repository" option on the left side, below File Shortcuts; if you select that, it will show "Clone a Repository" as an option. Select that, and it will open a box that allows you to paste your URL under "Remote Repository". Then, select the server folder that you want to clone the repository to (this will be a location that your SAS Studio server can access, likely on your network share or server disk), and select your profile. When you

hit OK, it should download the repository and place it in that folder, and should also appear in your Server folders. See the example below:

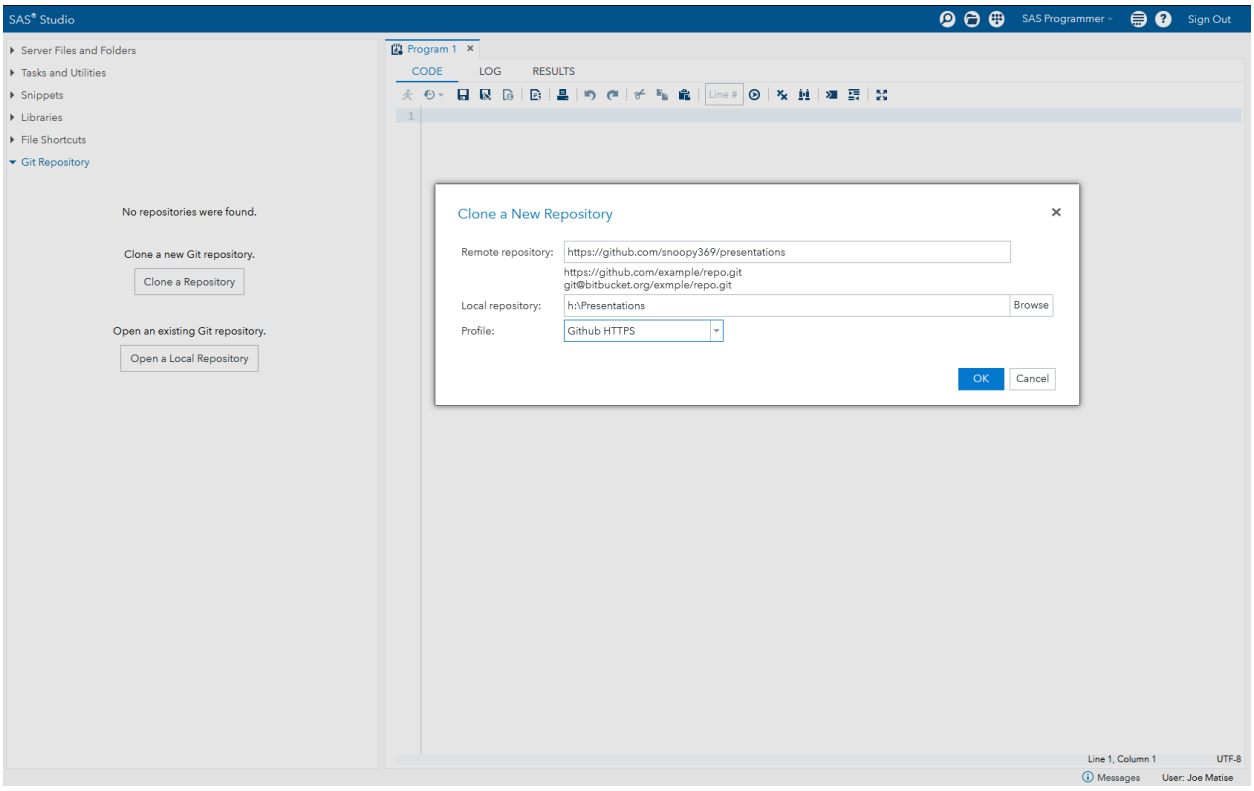


**Figure 4. Clone a Repository**

## OPENING A LOCAL REPOSITORY

You can also open a repository that already exists, if you already use Git outside of SAS Studio. Select the other option under the Git Repository menu, "Open a Local Repository", and identify which folder has that repository. The folder will have a different icon than most folders, indicating SAS already knows it is a Git repository – even if you haven't told it! Simply select the folder and it will add it to your repositories that Studio tracks.

## VIEWING THE HISTORY OF A REPO

Once you have the repository tracked in SAS Studio, either via cloning or via opening a local repository, you can view the commit history inside SAS Studio. The commit history is the list of every time a user committed a change to a file or group of files. It shows who committed the changes, what date, and the commit message they left.

To view the commit history, first select the repo from the dropdown in the Git Repository menu item. Below that is a button "History". Click that, and you will see the history for that repo. This history will include all of the commits for that repo, whether they occurred in your local repo or somewhere else – anything prior to the point at which you cloned (or updated via a pull) will be in the list. That history will look something like this:

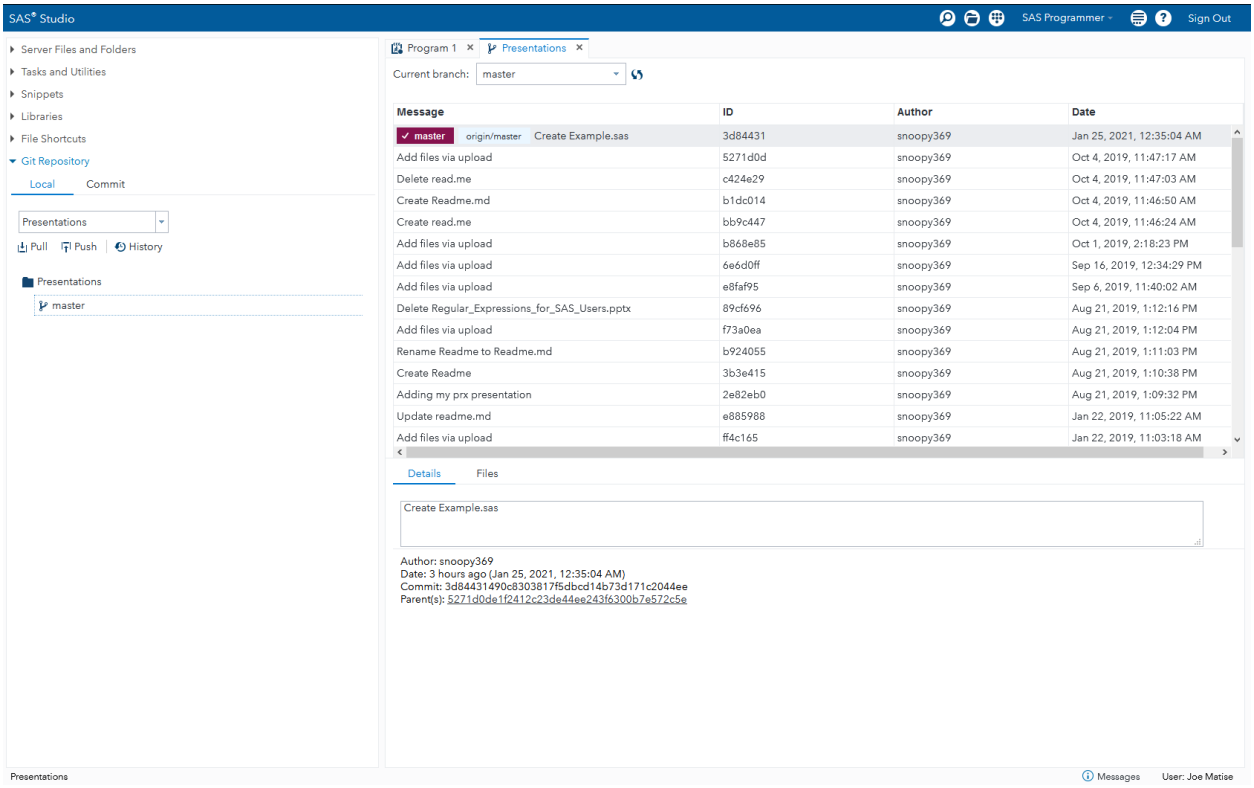


**Figure 5. History of a Git repo**

Notice the commits and dates here – this goes back years, even though I only cloned the repo locally for this paper. Also notice the top entry – it has two marks on it, "Master" and "Origin/Master". Master means it is the current version of the local repository's Master branch – this is the main branch, and if you do not have any branches it is the only branch. "Origin/Master" is the remote server's Master branch that we cloned this from (Github, in this case). Those two are currently the same – but they do not have to be.

## COMMITTING YOUR CHNAGES

If you make changes to a file, and wish to commit them, you can do that inside SAS Studio. Once you have made your changes, under the Git Repository menu item, select the repository you want to update from the dropdown, and then select the Commit tab. Under that tab, you have two lists; the top list is a list of files with changes. You can select each file, and view the "diff", or differences between the last version the Git repository has stored and the current version of the file. It will show you adds, deletes, and modifications.

If you are satisfied with the changes, you can then stage the file. Staging the file says you are ready to commit your changes for that file. You can stage all of your files, or stage them one at a time, using the buttons at the top of the box.

Once you have staged all of the files you wish to commit, enter a *descriptive* commit message – often mentioning the ticket(s) related to the change, or other details that will help you identify why you made the change – and press the "Commit Staged" button. This will commit the changes to your *local* Git repository, for all of the files that were staged.

If you then view the history of the files, you will see that now the Master tag is on a new line, but Origin/Master is still on the old line. See the figure for an example:

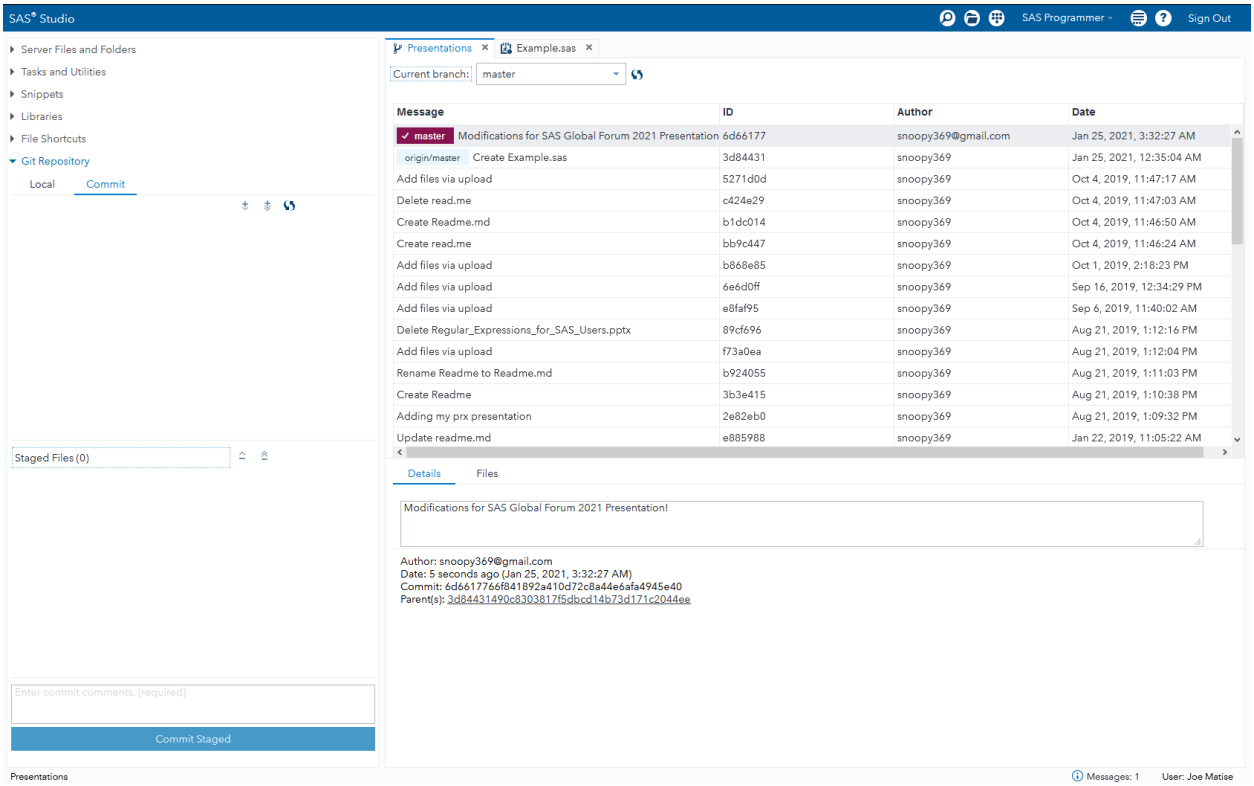


**Figure 6. History after committing changes**

This shows how the local version and the remote version now track different versions of the code. If another programmer were to clone the project, they would not have your changes – only the version tracked by Origin/Master. In order to make your changes available for other users, you need to use the *push* command.

## PUSHING YOUR CHANGES TO THE SERVER

Pushing your changes is very easy, as long as you haven't changed a file another user also changed. You do this from the Git Repository menu, under Local. Select the Git repository you want to work with from the dropdown, and then select "Push" right below. The screen should show a brief spinning circle, and then it should show the History list again – but now, with the Origin/Master tag on the current version. See the figure below:

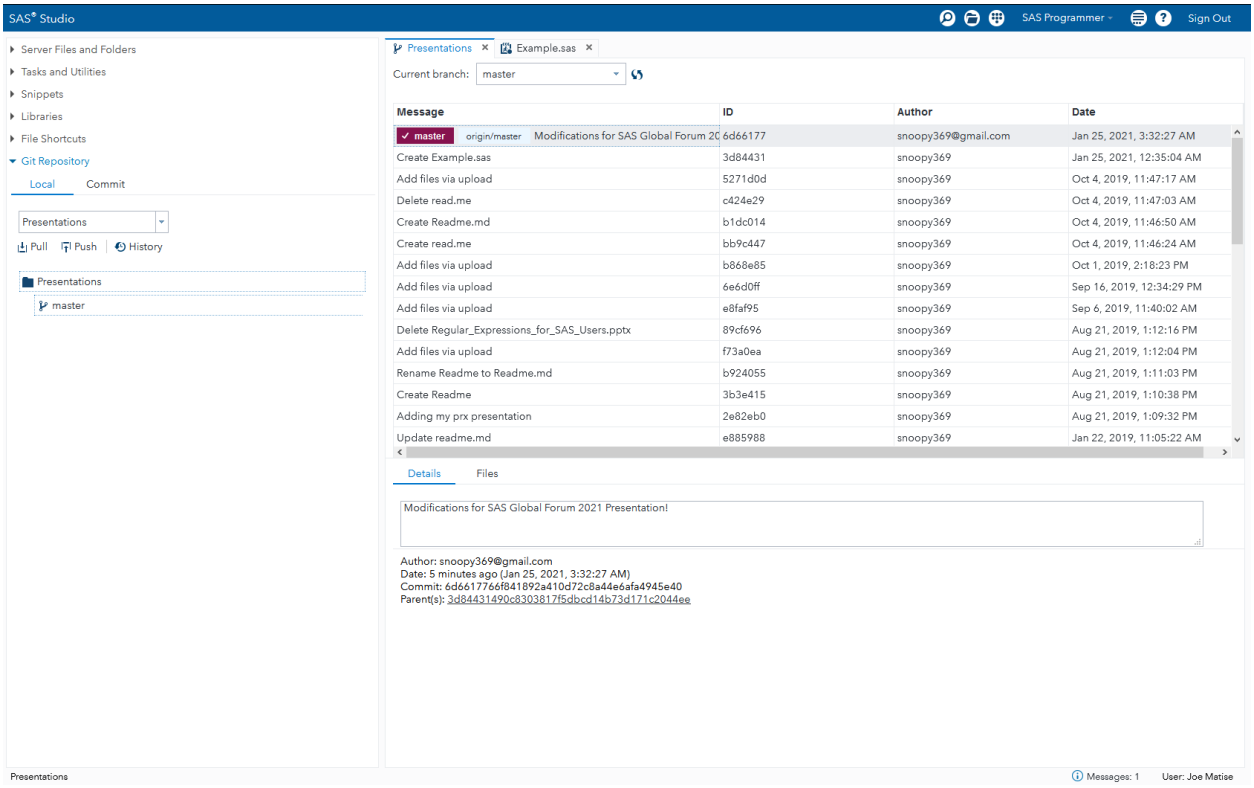


**Figure 7. History after pushing changes**

Now, any programmer who clones or pulls from the repository will get my latest changes.

If you did have a conflict, meaning someone else changed a file you also changed, SAS will let you know, and will unstage the affected file(s).

To resolve the conflict, you will need to first do a pull (described in the next section), and then resolve the conflicts before committing and pushing again.

### PULLING A NEW VERSION FROM THE SERVER

Pulling is the complement of pushing; it is how you receive changes made by others from the server and update your repository with those changes. It is done in the same manner as pushing; you select "Pull" from underneath the dropdown where you selected your repository under Git Repository. When you do that, you will see a brief spinning circle, and then a new history window – now with any changes made on the server.

If you do a pull while you have outstanding changes, you might have a conflict if a file you have made changes to was also changed on the server. SAS will list those conflicted files in the Unstaged box under Commit. You will need to resolve those conflicts (by editing the programs), and then commit and push the changes.

## OTHER SUPPORTED GIT OPERATIONS

Beyond the basics (clone, commit, push, pull), SAS supports several other operations inside SAS studio. In particular, SAS allows you to work with branches, which can be very helpful either to permit multiple users to work on the same codebase at the same time, or to allow one programmer to work on several different things at one time – such as code for different rounds, or testing out an idea in development without affecting production.

Branches are managed in the History window. By right clicking on the particular commit that you want to work with, you can create a new branch, checkout a branch, merge a branch, or delete a branch.

See the SAS documentation for more details on how to use these features!

## CONCLUSION

Git integration in SAS Studio makes it easier than ever to enable users to add Git to their workflow without missing a beat. With only a few setup steps, SAS Studio is now a first-class Git client, as well as the primary interface for SAS programming for many SAS users. This enables SAS programmers to collaborate more efficiently and take full advantage of all that Git has to offer, without leaving their (SAS) Studio!

## REFERENCES

SAS Institute, 2020. "Understanding Git Integration in SAS Studio." SAS Studio 3.8 User's Guide. Accessed on 4/15/2021. Available at https://documentation.sas.com/doc/en/webeditorcdc/3.8/webeditorug/p0hlfsbbbdnco4n1vep8z6k2p6us.htm

## ACKNOWLEDGMENTS

Thanks to José Centeno, Matt Kastin, Zeke Torres, David Trevarthen, and Jeff Vose for their contributions and assistance with this paper. Thanks as well to the SAS Global Forum staff for inviting me to speak on this topic.

## RECOMMENDED READING

- *The SAS Dummy: Using built-in Git operations in SAS, https://blogs.sas.com/content/sasdummy/2019/01/17/git-in-sas/*
- *Understanding Git Integration in SAS Enterprise Guide, Enterprise Guide 8.2 documentation, https://documentation.sas.com/doc/en/egdoccdc/8.2/egug/p078cl08fol7hkn11a8nf1f9izq7.htm*
- *The SAS Dummy: How to Organize your SAS projects in Git, https://blogs.sas.com/content/sasdummy/2020/11/10/sas-projects-git/*

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Joe Matise
NORC at the University of Chicago
matisejoe@gmail.com