Chris Hemedinger is the Director of SAS User Engagement. His talented team looks after SAS online communities, SAS user groups, developer experience and GitHub, tech newsletters, expert webinars and tutorials.

Chris is a recovering software developer who helped build popular SAS products such as SAS Enterprise Guide. Inexplicably, Chris is still coasting on the limited fame he earned as an author of *SAS For Dummies*. 
Workshop available on GitHub

github.com/sascommunities
git-workshop-pharmasug
Today’s topics

- What is Git™?
- Using Git with SAS tools
- Git functions in SAS programming language
- DevOps opportunities: collaboration and continuous integration
- Where to learn more
VERSION CONTROL

SAS Tutorial | How to use Git with SAS
What is Git?

Git is a **distributed version control system** that supports several collaborative and standalone workflows

- **Distributed** – each developer has a “clone” of the code repository
- **Workflows** – feature-based, experiment branches, prod/test/dev, patch/cherry-pick
- **Open source** – Git software is free and open source.

Several commercial systems (GitHub, GitLab, Bitbucket, and more) add features and enterprise-readiness.

Source: [https://git-scm.com](https://git-scm.com)
Benefits of using Git with SAS

**Version control**

**Collaboration**

**DevOps/automation**
Git has its own lingo

**Clone**
- Create a local copy of a repository

**Fork**
- Make a local copy of a repo where you don’t have push access

**Pull request**
- Open a discussion with proposed code changes to be merged to upstream repo

**Branch**
- A logical location to divert from the “main line” of code to stage changes

**Blame**
- Annotated code listing with revision/author for each line

**Commit**
- Record a set of changes to the local repo

**Commit History**
- Log that shows chronological list of changes to the repo

**Push / Pull**
- Update a remote repo with local changes, or sync/merge local repo with remote changes
SAS tools that integrate with Git

- SAS Enterprise Guide
- SAS Studio
- SAS programming
- SAS Data Integration Studio

And more! SAS Viya products add more Git integration
SAS Enterprise Guide and Git

Supports Git **internal** to project and/or with **external** repos

Project file (EGP)

- `.git repo`
  - `EmbeddedProgram1.sas`
    - Revisions / history
  - `EmbeddedProgram2.sas`
    - Revisions / history
  - `EmbeddedProgram3.sas`
    - Revisions / history

LinkedProgram1.sas
LinkedProgram2.sas
LinkedProgram3.sas

File system

- `.git repo`
  - `LinkedProgram1.sas`
    - Revisions / history
  - `LinkedProgram2.sas`
    - Revisions / history
  - `LinkedProgram3.sas`
    - Revisions / history
SAS Enterprise Guide with external repo

All history maintained within Git repo available to any tool that integrates with Git.
Git integration with SAS Studio

Available as of v3.8 (SAS 9.4) and SAS Viya 3.5

Complete support for local and remote repo management
New Git functions in SAS language

Functions added in SAS 9.4 Maint 6 and SAS Viya

New SAS functions that mirror most of the Git commands that users are familiar with.

Select functions - All documented by searching “Git functions” on support.sas.com.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GITFN.Clone</td>
<td>Clones a Git repository (for example, from GitHub) into a directory on the SAS server.</td>
</tr>
<tr>
<td>GITFN.Commit</td>
<td>Commits staged files to the local repository.</td>
</tr>
<tr>
<td>GITFN.Diff</td>
<td>Returns the number of diffs between two commits in the local repository and creates a diff record object for the local repository.</td>
</tr>
<tr>
<td>GITFN.Push</td>
<td>Pushes the committed files in the local repository to the remote repository.</td>
</tr>
<tr>
<td>GITFN.New.Branch</td>
<td>Creates a Git branch.</td>
</tr>
</tbody>
</table>

Name change: GITFN* → GIT* in SAS Viya
Demo
Connecting SAS to Git

**SSH or HTTPS**

- **SAS Studio**
  - Supports SSH or HTTPS

- **SAS Enterprise Guide 8.2**
  - Supports HTTPS

- **SAS GIT* functions**
  - Supports SSH or HTTPS
Generating an SSH key

SSH for passwordless access

• To generate a key for use with SAS Studio or SAS GIT functions:
  `ssh-keygen -t ecdsa -b 521 -C <email>`

• Will generate two files — a public and private key
  – Ex: `id_rsa.pub` and `id_rsa`

• Register the **public** key file with Git service

• **Protect** the **private** key file (readable by your account only)
Register SSH key

GitLab example
Generate an access token for HTTPS
Required for single-signon (SSO) or 2-factor auth

• HTTPS access requires username and password
• Instead of account password, use access token
• Access tokens convey specific permissions (api, read_repository, write_repository, etc.)
• Protect your access token!

From GitHub doc
Register access token

GitLab settings

GitLab example

SAS Studio settings
Use other tools alongside SAS

- Git Bash (CLI)
- Git GUI or GitHub Desktop
- VS Code
How to organize your SAS projects in Git

Read the blog article

- Don't create one huge repo
- Don't organize based on dept org structure
- Design with collaboration in mind
- Re-use 👍, Ownership 👎
- Work with Git features, not against them

https://blogs.sas.com/content/sasdummy/sas-projects-git/
Git integration is just the beginning
From there, trigger other operations

- Gerrit for code review
- Jenkins or Travis CI for build, deployment, and other continuous integration
- Trigger notifications/alerts in Slack, Microsoft Teams
Learn more

- **Using Git with SAS** *(SAS Users YouTube)*
- **Git functions in SAS 9.4 and SAS Viya** *(doc)*
- **Using Git in SAS Enterprise Guide** *(doc)*
- **Git with SAS Studio and SAS Enterprise Guide** *(video)*
- **SAS Software on GitHub**
- **Using built-in Git operations in SAS** *(blog)*
- **How to organize your SAS projects in Git** *(blog)*
- **DevOps with SAS 9: SAS code, GitLab, and Jenkins** *(community)*
- **developer.sas.com** for SAS app development
- **Pro Git** by Scott Chacon and Ben Straub, free online book about Git