Git 101 for Mainframers

Sylva Bursikova, CA Endevor Modern Developer Experience – Product Owner
Michal Vahala, CA Endevor Bridge for Git – Architect
Broadcom

November 2020
Session 5BB



Abstract

In recent years, Git has risen dramatically in adoption for development of off-host applications, becoming an almost de-facto standard across enterprises. As companies look to provide a common developer experience from Mobile to Mainframe, mainframers are being asked to look at using Git for mainframe development projects. In this session we will review what Git is, how it is typically used and why it is so popular. We will also discuss the role of the Enterprise or Team server along with the added value they bring to the development process.

Agenda

- What is Git and how does it work
- Enterprise Git Servers
- Common Git operations and workflows
- Using Git for the Mainframe
- Demo

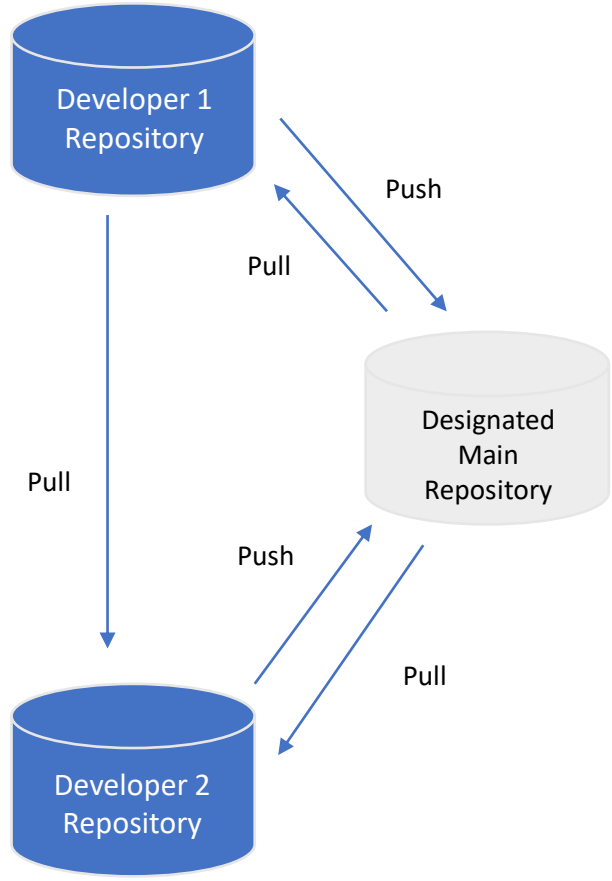
What is Git?

- Git is a (D)istributed (V)ersion (C)ontrol (S)ystem or DVCS
- Created in 2005 by Linus Torvalds for managing development of the Linux kernel
- Git is 100% open source and free
- Unlike Client-Server SCMs, every Git managed directory is a full functioning repository
- Why is it called Git?
  - When asked about the name git (which means unpleasant person in British slang), Linus Torvalds said: "I'm an egotistical bastard, and I name all my projects after myself."



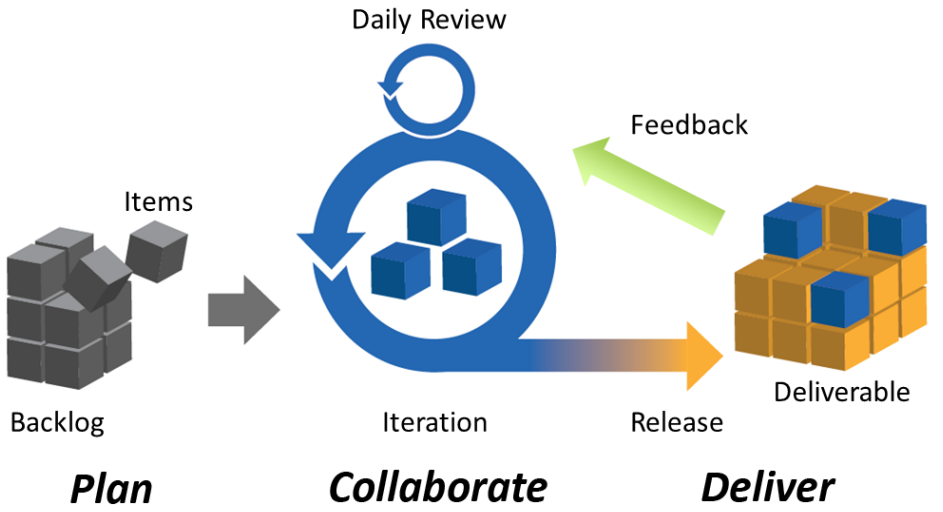
How does Git work?

- All Git instances or clones are equal
- Only by convention is any one designated the main instance
- Developers trade code back and forth between their clones and sync by using “push” and “pull” operations
- A pull operation integrates code from another developer into your repository and working directory
- A push operation sends your code to a remote repository
  - Pushes are generally used to send code to a designated main repository
- Between push & pull operations, developers can branch, switch branches, stash work and other operations



Why is Git so popular?

- Developers have full version control function without being connected to a server
- Git has excellent branching capability, developers can easily isolate work
- Changes are easily integrated – Git has excellent support for merging and tracking where changes came from and can auto-merge in many cases
  - Changes from the main branch can be integrated frequently
- All of this adds up to premiere support for Agile and parallel development

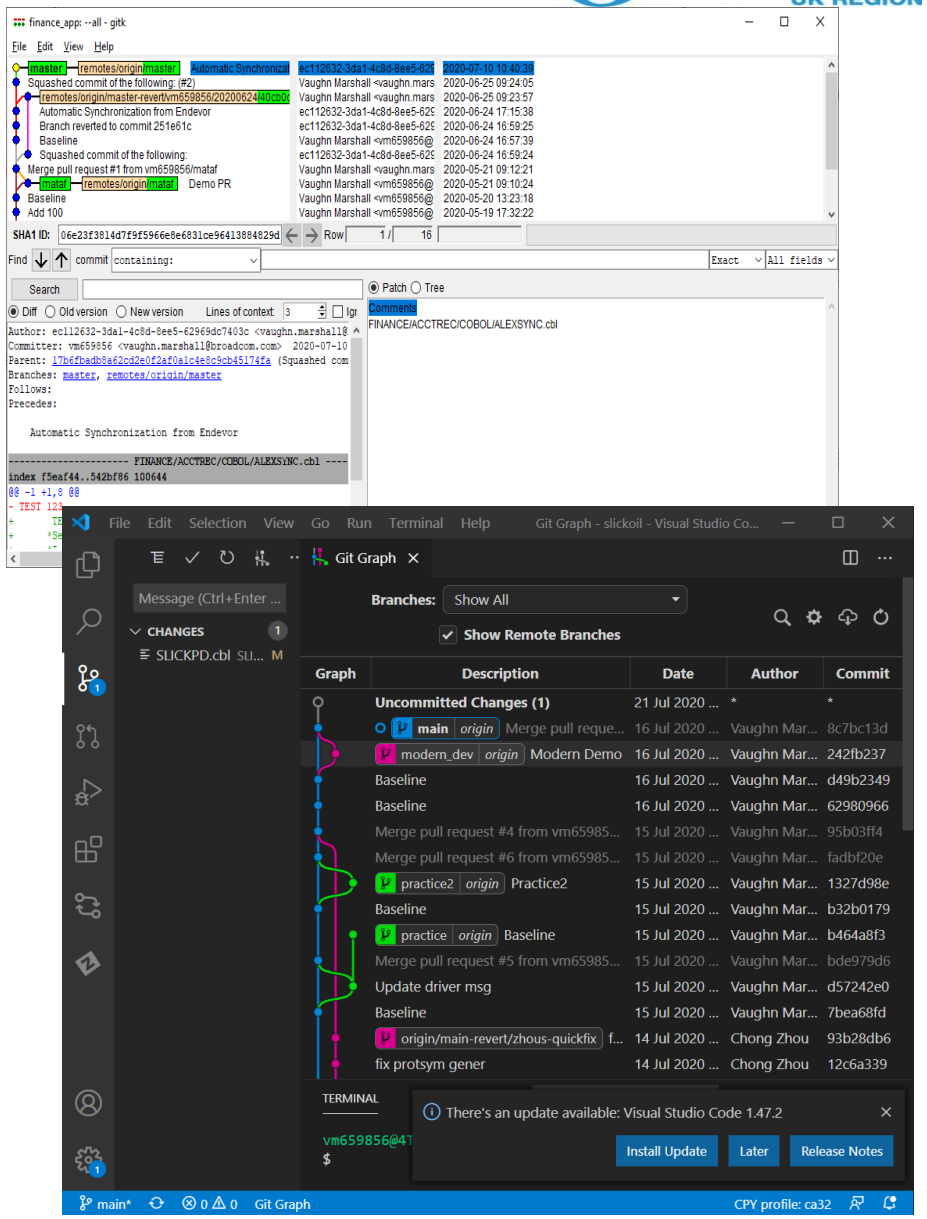


Agile Project Management: Iteration

Source: Wikimedia Commons

Git Interfaces

- By default installing Git installs a command line interface (CLI) and shell, as well as a very basic GUI
- Other interfaces can be downloaded, including:
  - Nicer GUI's
  - IDE extensions such as
    - VS Code Core Git Extensions as well as other add-ons such as Git-Graph
    - Eclipse's eGit which can be used with Topaz or IDz
    - Too many to name! Almost all IDE's have a Git extension available



Enterprise Git Servers

- Most Git users interface with an Enterprise Git repository of some sort. Prevalent Examples include:
  - Microsoft GitHub
  - Atlassian Bitbucket
  - GitLab
  - Azure DevOps
- Free and paid tiers are available



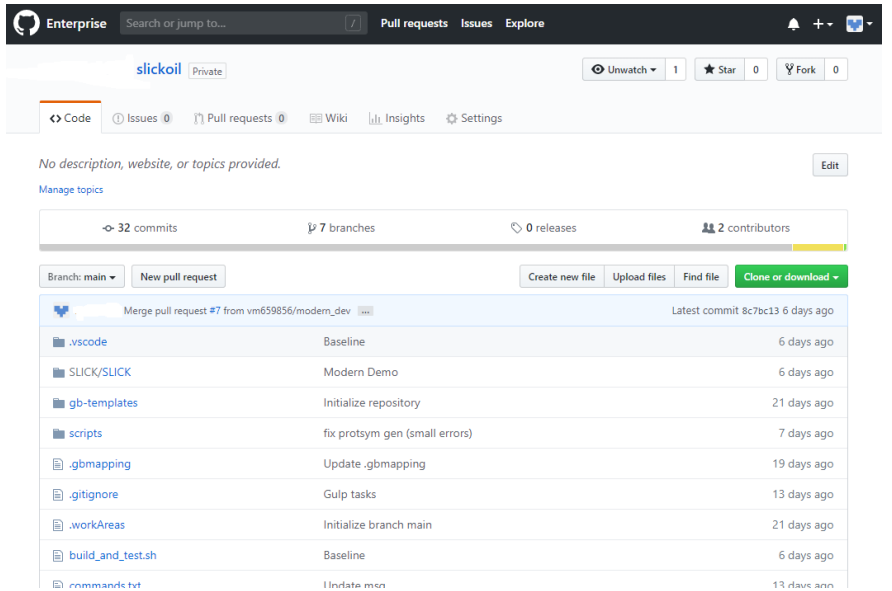






Enterprise Git Servers

- Enterprise Git servers provide
  - A central location for repositories
  - A web interface for accessing the repo
  - Centralized Security
  - Added collaboration tools – Pull Requests, Repo Forks, Code Reviews, Branch Protection Rules
  - Add-ons and plugins
  - Some include built in CI/CD or other planning or lifecycle capabilities



Common Git Workflows and Operations

- `git init` – initializes a directory as a git repository backed by a hidden .git folder and turns the directory into a working directory
- `git clone` – downloads a clone of a remote repository to a directory and initializes it as a repo / working directory
- `git fetch` – gets the latest changes from a remote repository into the current repo
- `git merge` – joins two or more development histories together
- `git stash` – saves your current changes and resets the directory to the latest HEAD commit

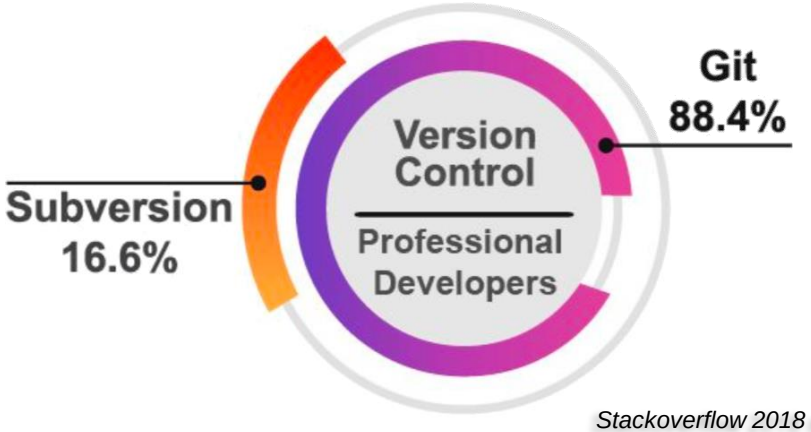
Common Git Workflows and Operations

- `git checkout` – changes the working directory branch or creates a new branch
- `git pull` – same as a fetch, then merge – essentially refreshes the repo and integrates the remote changes
- `git commit` – commits current working directory changes to the branch
- `git push` – sends latest commits to a remote repository (usually the clone origin)
- `git log` – shows the commit log history
- `git status` – shows the current state (delta) of the working directory vs the repo

Common Git Workflows and Operations

- A typical workflow for a change might go as follows:
  - `git clone` (for a brand new repo) or `git pull` (to get the latest changes) from an Enterprise Git repository
  - `git checkout -b <feature_branch_name>` to create a new branch to work on a change in isolation
  - Make your changes
  - `git commit` to commit your changes to the local repo
  - `git push` to create a remote copy of your feature branch in the Enterprise server
  - Create a pull or merge request from the branch
  - Code review, builds and automated checks such as static application analysis or unit tests
  - Merge the pull / merge request to the main branch

Using Git for Mainframe



Stackoverflow 2018 survey of 100,000 developers

- Git can be used with mainframe code just as it can with distributed code
- You will need a way to build the code on mainframe and therefore get the code onto the mainframe
- There are a few choices:
  - Use a mainframe Git client (Rocket's Git, ISPF Git Interface) + a build tool such as IBM DBB or custom JCL
  - CA Endevor provides an integration that lets you sync a Git repo in the background to an Endevor repository, modernizing in place using existing builds / lifecycle automation

Using Git for Mainframe

- Some considerations in your choice:
  - Mainframe monoliths may need to be refactored to move all code to Git in a “Git-native” solution
  - You may need to reinvent builds / lifecycle automation from your current SCM
  - Will you need to run your CI/CD tools on or off host (e.g. GitLab runners for z/OS or Jenkins agents or SonarQube scanning machines) or can you integrate remotely?
  - How will developers run builds?
  - Do you want to build on zD&T and are there performance or other considerations to that?

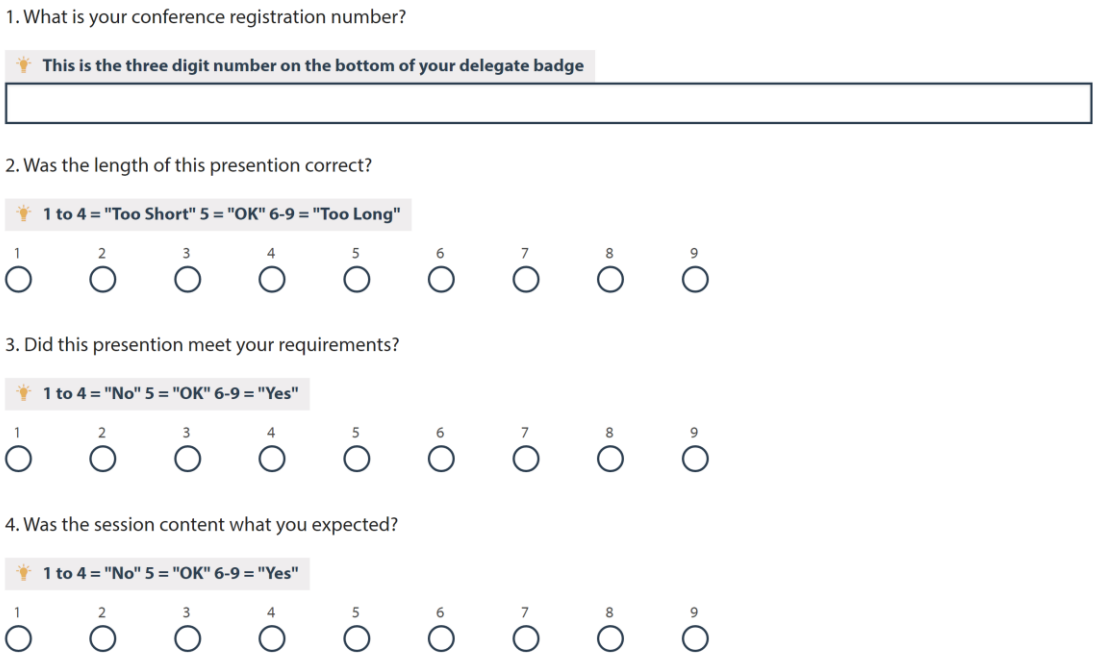
Demo

Git command line

Git plug-in in Visual Studio Code

Please submit your session feedback!

- Do it online at http://conferences.gse.org.uk/2020/feedback/5BB
- This session is 5BB



GSE UK Conference 2020 Charity

- The GSE UK Region team hope that you find this presentation and others that follow useful and help to expand your knowledge of z Systems.
- Please consider showing your appreciation by kindly donating a small sum to our charity this year, NHS Charities Together. Follow the link below or scan the QR Code:

http://uk.virginmoneygiving.com/GuideShareEuropeUKRegion

NHS CHARITIES TOGETHER