Git

Description

Git is the de-facto standard source control system for the tech industry and is one of the most flexible software tools to be found. Any developer or devops person probably needs at least a basic understanding of Git to get going and this course is intended for that purpose. This course covers all of the fundamental operations an experienced coder would use on a daily basis. The course begins with an introduction to Git and a comparison of Git to other version control systems. It then transitions into the nuts-and-bolts of working with Git, including everything from setting up a repository to advanced topics like branching and merging. Because the industry sometimes misuses git this course is also focuses on clearing up muddled understanding of git: the staging area, merge vs rebase, history management, branching and more

Duration

24 hours / 3 days

Intended Audience

- Any software developer or devops who needs to work with Git or understand Git better.
- Any software developer who has worked with Git but wants a deeper understanding of it.
- System administrators who are moving to devops in general or Git specifically.
- Any manager who needs to understand what is possible and how to manage git using projects.

Prerequisites

- Tech affinity.

Objectives

- Setup and use git
- Understand and use Git’s branching features correctly and effectively
- Decide on which workflows to use when using Git

Outline

- Introduction to Git
  - History of Git

  - Who is using Git
  - Adopting Git
- Git basics
  - Setting up a local repository
  - Setting up a client to a repository
    - Local
    - Remote
  - The staging area
    - git status
    - git add
    - git stage
    - git rm
    - git mv
  - Undoing things in the staging area
    - git restore --staged <file>
    - git reset
    - git reset <ref> <file>
    - git reset --hard
  - git diff
  - git commit
  - git log
  - git grep
- Configuring Git
  - Local and global config files
  - Configuring Git commands
  - Configuring signing
  - Adding aliases
  - .gitignore
    - ignoring patterns
    - whitelisting patterns
- Undoing things
  - Why you should not rewrite history, especially of published changes
  - History rewriting commands
  - git commit --amend
  - git reset --hard
  - git revert
  - git rebase
  - git restore
  - git cherry-pick and git cherry
  - using git rebase to split past changes
  - extreme undoing

        - Cutting the history tail
        - Rewriting many commits
- Remote repositories
    - Working with a remote repository
    - Setting up / publishing a repository
    - Understanding the repository structure
    - Working with Multiple remotes
    - Working with GitHub
- Branches
    - Theory: the many reasons we want local branches
    - Theory: the many reasons we want branches
    - Creating branches: git branch
    - Describing branches: git branch --edit-description
    - Renaming branches: git branch -m
    - Working on branches
    - Committing on branches
    - Moving between branches
        - git checkout
        - git switch
    - Visualizing branching activity
    - Pruning local/remote branches git branch -d git branch -D
    - git reflog
- Merging changes
    - git fetch
    - git pull
    - git rebase
    - Fast forwarding?
    - Cherry picking
    - Handling conflicts
        - basics
        - using merge tools
- Merge vs Rebase
    - Which should you choose? (Rebase)
    - Why?
- Workflows
    - Git does not force a workflow
    - Feature branches
    - Development vs production
    - Back porting changes
    - Examples of workflows
        - Working with your own workflow

        - Jenkins
        - Working with pull requests
        - Gerrit
- Showing Git data
    - git log
    - git ls-files
    - git show
    - git diff
    - git show-branch
    - git blame and git annotate
    - git whatchanged
    - Visual tools
        - Gitk
        - Source Tree
        - Git Karen
        - Git Cola
        - Many more
- Under the hood
    - Digital signatures overview
    - Core ideas
        - Always on a branch
        - SHA includes all history
        - SHA is unique in the world
    - Core concepts
        - The three core structures: commit, tree, blob
        - Commits point to previous commit
        - Commits point to trees
        - Trees point to blobs (never store anything twice)
    - .git:
        - The Git object store and how it works
        - What branch am I on?
        - What commit am I on?
        - Where are the tags?
        - Where are the heads?
        - Where are the remotes?
    - What happens when you
        - Add to staging area (the index)
        - Commit
        - Create a branch
        - Create an annotated tag
- Work trees

  - Why are they needed?
  - Creating a worktree
  - Working with worktrees
  - Pruning worktrees
- Tagging
  - Why tag?
  - Difference between annotated and non annotated tags
  - Pushing and pulling tags
  - Using tags in other Git commands
- Understanding revisions (gitrevisions)
  - Various things git understands
    - \<branch\>
    - \<object\>
    - \<commit\>
    - \<commit-ish\>
    - \<tree\>
    - \<tree-ish\>
    - \<pathspec\>
  - Commit-ish vs Tree-ish
  - git rev-parse
- Searching by content
  - git grep with git rev-list
  - git log
- git sub-modules
  - How to create them?
  - How to pull them?
- Stashing
  - Why would you want stashing?
  - Creating and naming stashes
  - Apply a specific stash
  - Delete stashes
- Git hooks
  - How to set up hooks?
  - What guarantees do you get?
- Built in tools
  - git instaweb
  - git daemon
  - git http-backend
  - git shell
  - git export
  - git bisect

    - git describe
    - git archive
    - git bundle
    - git submodule
    - git notes
- Git tools
    - Git and programming languages: GitPython
    - Git and development platforms: GitHub, BitBucket, Gitlab
    - Git and IDEs: PyCharm, Eclipse, Spyder
    - Git and CICD tools: Jenkins, Bamboo

Installations

- Any recent distribution of Linux with sudo/root privilege.
- WSL installation on Windows with git installed.
- Git BASH installed on Windows.
- In any case the installations are not a must and the instructor can guide the students how to perform the installations on the first day of the course.