Module Checklist

Version Control with GIT

By Techworld with Nana

Video Overview

Check your progress... 1/3

Introduction to Version Control

❏ Watched video

Basic concepts of Git

❏ Watched video

Setup Git Repository

❏ Watched video

❏ **Demo executed**

  ❏ Created a user account for GitLab or GitHub (if you haven't already)

  ❏ Created your own git repository/project on GitLab or GitHub

  ❏ Installed a Git client (GUI or Command Line Tool)

  ❏ Created a SSH key pair (in case you don't have one already)

  ❏ Added public ssh key to GitLab or GitHub

  ❏ Configured your git command line tool with your username and email address using "git config .." command

  ❏ Cloned remote repository to your local machine

Useful Links:

- Git GUI - Installation Guides: https://www.git-scm.com/downloads/guis
- Install Git Command Line: https://git-scm.com/downloads
- Steps to create SSH Key Pair: https://docs.gitlab.com/ee/ssh/#common-steps-for-generating-an-ssh-key-pair

Working with Git

❏ Watched video

❏ **Demo executed** - make your first commit and push to your remote repository

Check your progress... 2/3

Initialize a Git Repository locally

❏ Watched video

❏ **Demo executed**

  ❏ Created another project locally (which is NOT a git repository yet)

  ❏ Transformed project into a git repository with *git init*

  ❏ Configured a remote repository for that local repository (*git remote add origin ...*) and pushed to it

Concept of Branches

❏ Watched video

❏ **Demo executed** - create a branch locally and remotely

Merge Requests

❏ Watched video

❏ **Demo executed** - create a Merge Request

Deleting Branches

❏ Watched video

❏ **Demo executed** - delete a branch remotely and locally

Avoiding Merge Commits (rebase)

❏ Watched video

❏ **Demo executed**

Resolving Merge Conflicts

❏ Watched video

❏ **Demo executed**

Check your progress… 3/3

Don’t track certain files/folders (.gitignore)

- ❏ Watched video
- ❏ **Demo executed** - create a .gitignore file to exclude editor specific files/folder

Save work-in-progress local changes (stash)

- ❏ Watched video
- ❏ **Demo executed**

Going back in history (git checkout, git log)

- ❏ Watched video
- ❏ **Demo executed**

Undoing and changing commits (reset, revert, amend)

- ❏ Watched video
- ❏ **Demo executed**
  - ❏ Used git reset
  - ❏ Used git commit --amend
  - ❏ Used git revert

Merging Branches

- ❏ Watched video
- ❏ **Demo executed** - merge bugfix branch into master

Git for DevOps

- ❏ Watched video

More Resources...

Best practices

Commit related best practices:

- Use descriptive and meaningful commit messages
- Commit in relatively small chunks
- Commit only related work
- Adequately configure the commit authorship (name and email address) with git config

Avoiding very large deviations between local and remote repository:

- Keep your feature/bugfix branch up-to-date with remote master and/or develop branch. So pull often from remote git repository
- Branches shouldn't be open for too long or master branch should be merged into your feature/bugfix branch often

Others:

- Don't git push straight to master branch
- Use -force push carefully! Do NOT force push into master or develop branches or better only when working alone in a branch
- Create a separate branch for each feature or bugfix and name the branch with prefix "feature/xx" and "bugfix/xxx" respectively
- Doing Code Reviews via Merge Requests
- Use .gitignore file to ignore e.g. editor specific files, build folders

More Resources...

More useful commands

- git diff (show difference between working changes and last commit)
- git diff --cached (show difference between staged changes and last commit)
- git cherry-pick (apply the changes introduced by some existing commits)

Cheatsheet

- Handy git cheatsheet: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet