Git

Getting started with Version Control

You should learn to understand how to use git locally and remotely and get some familiarity working with the basic git commands.

Git version control

- Free, open source, version control system
  - Most likely the most widely used version control system today
- Distributed
  - Can be used only locally
  - But local repository may also be pushed to remote
- Not just version control, tool for teamwork also
  - And part of deployment chain (CD, continuous delivery)
- What is a git repository
  - Basically, just a directory containing the files you work with
  - And .git –directory where git places the versions you commit

- We are not going to go through all possible commands
  - Just enough to get you started (kind of walkthrough)
    - After all you will only need couple of commands you repeat
  - Check the git reference for more

Getting started locally

- Create a working directory for your project
  - C:\course\gittest
- Open command prompt and cd into the project directory
- Initialize the git repository
  - git init
  - Creates a hidden directory .git

- Create a file to your directory
  - echo First step > first.txt
- Make sure git tracks the changes
  - git add –A
- And commit changes
  - git commit –m “This is the first commit”
- You may need to run git config commands to make yourself known to git
  - Email and username

This is what you do repeatedly:’
- git add –A
- git commit –m “Some comment”

Discard changes to a file after last commit

- Open your working directory with Visual Studio Code
  - Make a change to first.txt
- On command prompt
  - git checkout -- first.txt

Discard all changes after last commit permanently

- With Visual Studio Code
  - Make a small change to first.txt
  - Create also a file called second.txt
- On command prompt run
  - git reset –hard
  - What happens to your changes?

- Again, modify the first.txt
- On command prompt run
  - git add –A
  - git reset --hard

git add is necessary for the git to track any changes to the file.

There is also git rm.

Discard changes but save them for later use

- Again, modify an existing file and add a new file
- On command prompt run
  - git add -A
  - git stash
- What happens in the directory
- Run
  - git stash list
- Create a file third.txt, and then run
  - git stash pop

Create a branch

- First make certain your master is committed
  - git add –A
  - git commit –m “Default for Branch”
- Create and checkout a new branch
  - git checkout –b second
- Show branches
  - git branch
- Create a new file branchfile.txt
  - git add –A
  - git commit –m “Committed to branch”
- Checkout different branches
  - git checkout master
  - dir
  - git checkout second
  - dir

You may create a new branch for a new feature in your solution and keep the master in the same state as the version of your application in production.

When the feature is finished you merge it back to master,

Merging branches

- First checkout master
  - git checkout master
- Then merge with second branch
  - git merge second
  - dir
- Create masterfile.txt
  - git add –A
  - git commit –m “Master with branchfile”
- Checkout second and merge with master
  - git checkout second
  - git merge master
  - dir

Resolving conflicts

- Two branches contain committed changes to the same file
  - git checkout second
  - Modify masterfile.txt
  - git add –A
  - git commit –m “Second modified masterfile”
  - git checkout master
  - Modify masterfile.txt
  - git add –A
  - git commit –m “Master modified masterfile”
  - git merge second
- Check what happens in Visual Studio Code
  - You can now select which changes to apply

Time for Quiz

- After you have completed the Quiz, you can continue to work with these files.

Getting started with GitHub

- Github gives you remote git repositories
- Sign in to github and create a new repository
  - Check to option to create README.md –file
- On the command prompt of your earlier “project” (gittest)
  - git remote add origin https://github.com/[your/repository]
  - git push origin master

- Study your repository at GitHub, it now has two branches
- On the command prompt
  - git pull origin main
  - git checkout main

Some extra work with branches

- Github by default creates a main-branch
  - Git locally creates master-branch
- Let's merge these into main
  - git checkout main
  - git merge master --allow-unrelated-histories
  - git add –A
  - git commit –m "Merged from master"
  - git push origin main
- Now we can delete master (local and remote)
  - git branch –d master
  - git push origin –delete master

Let's experiment with teamwork

- Clone your github repository to a second directory
  - cd ..
  - git clone https://github.com/[your/repository] gittest2
  - Open the gittest2 in second Visual Studio Code instance

- You you should have two Visual Studio Codes running
  - Let's say that the first instance in which you edit the original local repository is user1
  - And the new instance where you have opened the cloned repository is user2
- In both editors (or directories) create a branch for the user
  - Use the names user1 and user2
  - Also, in both editors push the newly created branch to the remote repository

Team is working on features

- Both users create a new file
  - User1 creates user1file.txt
  - User2 creates user2file.txt
- Both commit their work and push to remote repository
- User2 needs work done by user1
  - git pull origin user1
  - git add –A
  - git commit –m “Combined work”
  - git push origin user2

git fetch ?

Fetch fetches the remote branch so you can start working with it locally

Both users modify same file

- In both editors make a modification to user1file
  - Commit and push
- And now user1 wants to access changes of user2
  - git pull origin user2
  - Again, Visual Studio Code displays the differences

- In cases like this you of course need to be careful which changes you accept.

Merging the work from different users may become tedious, let's try to avoid that…..

So, for now

- You should have some familiarity working with basic git commands
  - git clone
  - git add
  - git commit
  - git branch and git checkout
  - git push, git pull
  - git merge