Lancaster University

Problem Set 1: Version control and working in a team (Part 1)

Aims

This week's lab aims to:

- Introduce you to version control using Git and GitHub.
- Give you initial experience of working collaboratively within a larger programming project.

Tasks for Week 11

This week's tasks aim to familiarise you with version control principles.

Please watch the online demo on version control by David and Joe to understand some key concepts on version control before you attempt the tasks.

You are strongly encouraged to use the Internet to find additional examples that will help you understand version control and how to resolve conflicts when changes in files are merged. Ask one of the TAs if something is not clear or you need assistance.

SCC.110: Software Development Term 2. Java Programming.

Lancaster University

Task 1: Creating a GitHub account and linking it to a local directory

Sign up for a GitHub account and create a remote repository:

- Open a web browser.
- Sign up for a GitHub Student Developer Pack (https://education.github.com/pack). **Use your Lancaster University email address!**
- Go to 'Repositories' and create a new repository (hit the green button labelled new).
- Name the new repository 'SCC110'.
- Although the description field is optional, please add a description, e.g., *Repository for SCC.110 practical sessions*.
- Make your repository private.
- Do not initialize the repository (e.g., do not add a README file).
- Choose the MIT license.
- Hit the green button labelled Create repository

Now that you have created a remote repository on GitHub, clone it on your local machine using Git.

- Use the command-line shell (e.g., Terminal), go to the h-drive, create directory Year1, i.e., `mkdir Year1`, and go to this directory, i.e., `cd Year1`.
- Type `git --version` to ensure that Git has already been installed.
    - If you receive an error message, Git is not installed. Please go to: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git and follow the instructions to install Git.
    - If `git --version` returns a version number, proceed to the next step.
- Clone the remote repository to your local machine:
    - Go to your web browser and make a note of the URL of your GitHub account. It should have the format `https://github.com/<Your_account_name>/SCC110`.
    - Go to the command-line shell and type:
      `git clone https://github.com/<Your_account_name>/SCC110`
      (Replace <Your_account_name> above with your actual account name)
- If you are now asked for a username and password, when you provide your password you should see an error message. Due to passwords not being supported anymore you need an access token:
    - Go to your GitHub account, click on the dropdown menu at the top-right corner and select 'Settings'.
    - At the bottom of the left-hand side menu, select 'Developer settings' and then 'Personal access tokens' -> 'Tokens (classic)'.
    - Choose 'Generate a personal access token'.
    - Type a reason for the need of the token in the 'Note' field, e.g., 'Access to the SCC110 repo'.
    - Click 'repo' in the list, ignore all other fields and click on 'Generate token'.
    - Make sure you save this token on your h-drive as you can only see it once.

    - Now run the clone command again.
    - Provide your username and copy the long string of characters (access token) that has been generated and paste it into the password field (at the command-line shell).
- If you were not asked for a password, or if you obtained and provided an access token, the SCC110 repository should now appear on your machine.
- Type `ls` and `cd` into the new directory that contains your GitHub repo.
- Type `git config credential.helper 'cache --timeout=300'` to avoid typing your username and password every time you use Git.

Task 2: Storing versions of a file locally and pushing versions of this file to the remote repository.

- Create a text file in the SCC110 directory on your machine (call it README.txt) and write a few lines in it (write a poem!).
- Go back to the command-line shell and type `git status`. You will see that your text file has not been added to Git yet, i.e., it is untracked, so Git does not know about this file.
- Type `git add README.txt` followed by `git status` again. Git now knows about the text file, but the file has not been backed up (i.e., committed) yet.
- Type `git commit -m 'created new README file'` to commit files that have not been backed up yet and include a message (in this case 'created new README file') to explain what changes have taken place.
    - You might be asked to provide an identity, in this case follow the command provided in the terminal `git config --global user.email "youremail@yourdomain.com"` without the "" but with the user.email.
- If you type `git status` again, you will be informed that the working tree is clean and there are no files to commit.
- Edit README.txt and add or modify a few lines. Then, repeat the previous steps to add and commit the new changes.
- Type `git log` to see a history of the changes that you have made.
- Type `git push` to push all changes to the remote GitHub repository.
- The file README.txt should now appear in the SCC110 repository on GitHub (use your web browser to check).

Task 3: Writing a simple text file collaboratively

FOR THIS EXERCISE YOU WILL NEED TO WORK IN PAIRS. Your objective is to create a new repository and collaboratively develop a text file.

Working as a team:

- One of the two team members should host a GitHub repository. That user should go to their homepage on GitHub and log in (if not already logged in).
- Create a **private**, **MIT licensed** repository named 'Teamwork'. You can do this by selecting the **Repositories** tab and clicking **New**.

- Invite the other member of your team as a collaborator to the repository (use their GitHub username). You will find this under the **Settings -> Collaborators** page of your repository on GitHub. The collaborator will most likely receive an invitation and has to accept it before they are allowed to clone your repository.
- Make sure that the collaborator can **clone** the repository on their computer.
- One of the members should create a file called README.txt that contains a single paragraph and push it to the remote repository (using `git add`, `git commit` and `git push`, as explained before).
- The other member of the team should then obtain the most recent version of the README.txt file from the remote repository using `git pull`, add a second paragraph to the text file and then add, commit and push the changes to GitHub.
- The README.txt file in the Teamwork repository on GitHub should now contain two paragraphs.
- Continue editing the file concurrently and commit the changes. Resolve any conflicts that may arise.

Hacker Edition

Now that we have created a GitHub repository for our code and have experimented with it, there are a lot of interesting things we can do with it. One of them is to create 'branches'. As you might have noticed, we have been working on the 'main' which usually contains the current working version of the code. Branches are, like trees, deviations from the stem or main in this case. In big projects there can be many branches within a project, each working on new aspects of the project. Working in branches helps with reducing the risk of corrupting the main as new features are first developed in a separate branch, if things go wrong this branch can be reversed or deleted without impacting other work. It also helps with collaboration as everyone works on their own branch until their feature is merged with the main once it is done and tested.

Make sure your files are up to date by using `git status`, if not pull or push all changes before continuing.

- Type `git branch NewFeature` this will create a new branch in the project called 'newFeature'
- Once we have created the branch we use `git checkout NewFeature` to move to the new branch
- Now that we are on a new branch we can create new files let's make a new file called 'FeatureOverview.txt' by using the following command `echo "This is a new feature" > FeatureOverview.txt`
- We now need to add this to the branch by using the `git add FeatureOverview.txt` command
- Similar to before we now need to commit our changes with the following command `git commit -m "Added a file containing an overview of the new features"`

- Now that we have committed the changes to the branch we can push them to GitHub using the `git push origin NewFeature` command. This will push the changes to GitHub on the NewFeature branch.
- Now go to github.com and check if you can see a new branch.
- If you can see this we can now go back to our main branch with `git checkout main`
- Now that we are back on the main branch we can merge our NewFeature with our main with the following command `git merge NewFeature`. Check if you can see this happened through the website.
- Then we push the changes with `git commit -m " merge new feature"` and then we can do `git push origin main`.
- Now that we have completed the work on the NewFeature branch we can delete it by typing `git branch -d NewFeature` to delete it locally
- Now we can also delete it on the server by doing `git push origin -d NewFeature`

Now we have successfully created a new branch, pushed code to it, merged it with the main and deleted the branch. Make sure to use this whenever you are writing new code to make sure you do not break any of your working code.