**GitHub Lab**

• Follow the GitHub instructions below

**Introduction**

You will host your project on GitHub. GitHub also stores all versions of your project each time you upload. This allows you to revert back to a previous version if you run into problems.

You will create a repository (repo) on GitHub. Using Git Bash you will create an ISD2WT directory on your onedrive and link this directory to your repository on GitHub.

From then on you just have to put the latest code in this ISD2 WT directory and complete step 7 to push the code to GitHub directly from the ISD2WT directory

**Tasks**

**Task 0: index.html**

* • Make sure the home page for your project is called index.html

**Task 1: Create a GitHub Account**

* • Goto http://www.github.com and follow the process of creating a new account
* • Use your Knumber as your username and your student email address as the email
* • Click ‘Signup for GitHub’
* • Follow the various instructions e.g. verify your account, select a plan, choose free, fill in the experience and planned use of Git Hub and interests as you wish.
* • Click ‘Complete Setup’
* • Verify your email address through your student email
  1. • In your own time have a look at the following optional resources to find more about Git:
* https://backlogtool.com/git-guide/en/
* http://rogerdudler.github.io/git-guide/
* <https://www.lynda.com/Git-tutorials/Git-Essential-Training/100222-2.html>
* <https://www.youtube.com/watch?v=w3jLJU7DT5E>
* <https://www.youtube.com/watch?v=0fKg7e37bQE>

**Task 2: Install tools**

• **THIS STEP IS NOT NEEDED IN THE LAB AS GIT BASH IS ALREADY INSTALLED**

• This step creates a Linux working area to communicate with GitBash which uses Linux

• In OSX you may need to install Xcode command line tools to use Git in the terminal.

• In Windows you will need to install git software as it is not installed by default. A good option is available in http://ww.git-scm.com

**Task 3: Create Your GitHub Repository**

The files that make up your website will need to be stored within a GitHub repository.

• Ensure you have logged into Github.com. Check your location – if necessary go to https://github.com/new

• Create a repository called **(ISD2WT)**

• Type in a relevant description e.g. News Website

• Make sure your repository is **PUBLIC**

• Click initialize the repository

• Click ‘Create Repository’

• If a note pops up on Actions click dismiss

• This next step creates the folder on your K drive which will communicate with the GitHub repository.

• Now open up **terminal** in Mac or use the **git-bash** command prompt on Windows. Type GIT in the Windows search box and select Git Bash.

• Type the following commands to firstly change directory to the **Desktop** and then copy or clone this new Github repository (empty at the moment) to your computer. Once you got your local copy, move into the project folder using the **cd** command.

cd

cd Desktop

git clone https://github.com/YourGitHubUserName/ISD2WT.git

cd ISD2WT

Note: Make sure that you change the clone URL to the URL of your GitHub repo. This can be found on the main project page in Github. E.g. https://github.com/K00123456/test.git

**Task 4: Creating an Orphan Branch**

Now you need to create a new orphan branch within your repo that will hold all of your website files. This new branch should be called gh-pages.

git checkout --orphan gh-pages

**Task 5: Adding Your Website Files**

Now that your repo has been properly setup it’s time to files that make up your website. No need to copy documents to GitHub

Copy the contents of your project folder into the **(whatever you named your folder)** repository folder on your one drive. This Repository is now your local root folder and should contain all your files.

Once you have added the local root folder contents to your repo you need to commit the changes. To do this you can use the following commands.

git add --all

git commit -m "Adding pages"

**Task 6: Pushing Your Changes to GitHub**

Okay so you’ve got all your files where they need to be. The only thing left to do now is to push the new gh-pages branch up to GitHub. You do this using the git push command.

git push origin gh-pages

GitHub will require you to log in again to complete this task. You will get an email alerting you to code changes on Git Hub

**Task 7: Making changes and updating your site**

From now on you can make changes to your site as required by editing / updating the files in your new Local Root Folder and then executing the following commands in the terminal

cd foldername

git add --all

git commit -m "Add a message here that describes what you added or changed"

git push origin gh-pages

**Your website should not be available at http://username.github.io/repname**

**Extra Notes:**

You cannot create an empty folder *and then* add files to that folder, but rather creation of a folder must happen *together with* adding of at least a single file. On GitHub you can do it this way:

* Go to the folder inside which you want to create another folder
* Click on *New file*
* On the text field for the file name, first write the folder name you want to create
* **Then type /**. This creates a folder
* You can add more folders similarly
* Finally, give the new file a name (for example, .gitkeep which is [*conventionally*](https://stackoverflow.com/a/7229996) used to make Git track otherwise empty folders; it is not a Git feature though)
* Finally, click *Commit new file*.

[**Create a new local repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init)

git init

[**Check out a repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone)

Create a working copy of a local repository:

git clone /path/to/repository

For a remote server, use:

git clone username@host:/path/to/repository

[**Add files**](https://www.atlassian.com/git/tutorials/saving-changes#git-add)

Add one or more files to staging (index):

git add <filename>

git add \*

[**Commit**](https://www.atlassian.com/git/tutorials/saving-changes#git-commit)

Commit changes to head (but not yet to the remote repository):

git commit -m "Commit message"

Commit any files you've added with git add, and also commit any files you've changed since then:

git commit –a

[**Push**](https://www.atlassian.com/git/tutorials/syncing#git-push)

Send changes to the master branch of your remote repository:

git push origin master

[**Status**](https://www.atlassian.com/git/tutorials/inspecting-a-repository#git-status)

List the files you've changed and those you still need to add or commit:

git status

[**Connect to a remote repository**](https://www.atlassian.com/git/tutorials/syncing#git-remote)

If you haven't connected your local repository to a remote server, add the server to be able to push to it:

git remote add origin <server>

List all currently configured remote repositories:

git remote -v[**Branches**](https://www.atlassian.com/git/tutorials/using-branches)Create a new branch and switch to it:

git checkout -b <branchname>

Switch from one branch to another:

git checkout <branchname>

List all the branches in your repo, and also tell you what branch you're currently in:

git branch

Delete the feature branch:

git branch -d <branchname>

Push the branch to your remote repository, so others can use it:

git push origin <branchname>

Push all branches to your remote repository:

git push --all origin

Delete a branch on your remote repository:

git push origin :<branchname>

[**Update from the remote repository**](https://www.atlassian.com/git/tutorials/syncing)

Fetch and merge changes on the remote server to your working directory:

git pullTo merge a different branch into your active branch:

git merge <branchname>

View all the merge conflicts:

View the conflicts against the base file:

Preview changes, before merging:

git diff

git diff --base <filename>

git diff <sourcebranch> <targetbranch>

After you have manually resolved any conflicts, you mark the changed file:

git add <filename>

**Tags**

You can use tagging to mark a significant changeset, such as a release:

git tag 1.0.0 <commitID>

CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:

git log

Push all tags to remote repository:

git push --tags origin

[**Undo local changes**](https://www.atlassian.com/git/tutorials/undoing-changes)

If you mess up, you can replace the changes in your working tree with the last content in head:

Changes already added to the index, as well as new files, will be kept.

git checkout -- <filename>

Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:

git fetch origin

git reset --hard origin/master

**Search**

Search the working directory for foo():

git grep "foo()"