

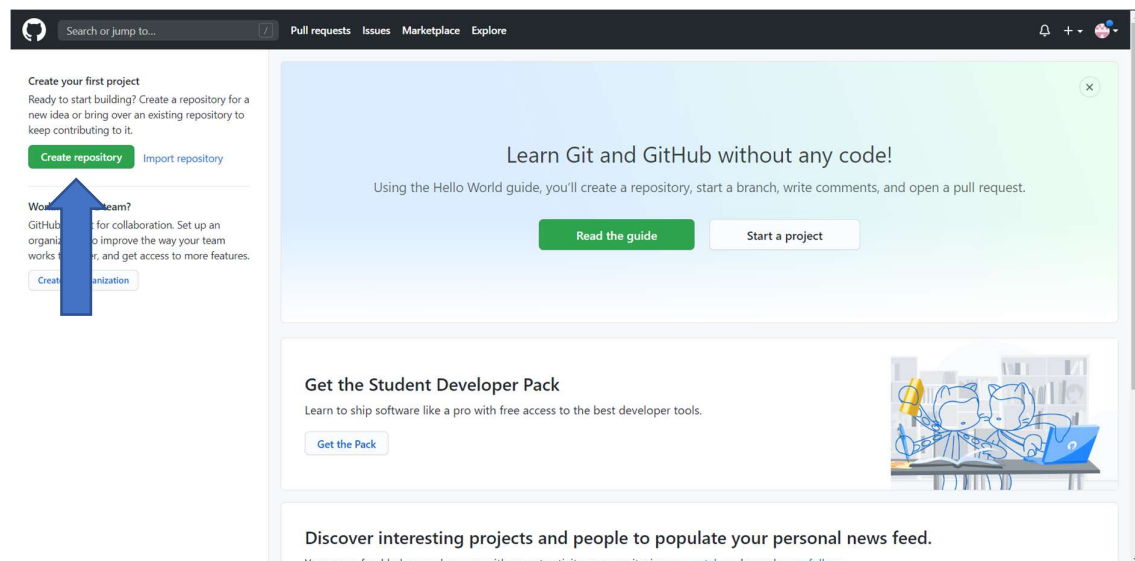
Git for COBOL Course

Last Update: 29 July 2020

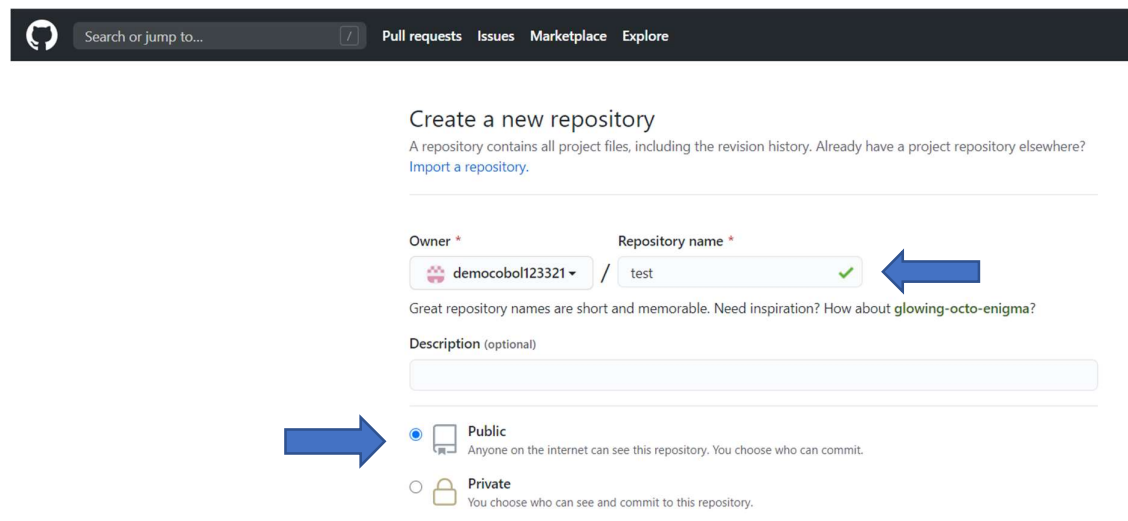
In this document, we will show how we can upload our files into GitHub for this COBOL course. Two pathways will be shown here, one for people using the IBM Developer for z/OS or the IBM Z Open Development and another one for those using TSO/ISPF.

For both pathways, you will need to head over to [GitHub.com](https://github.com) and create a free account.

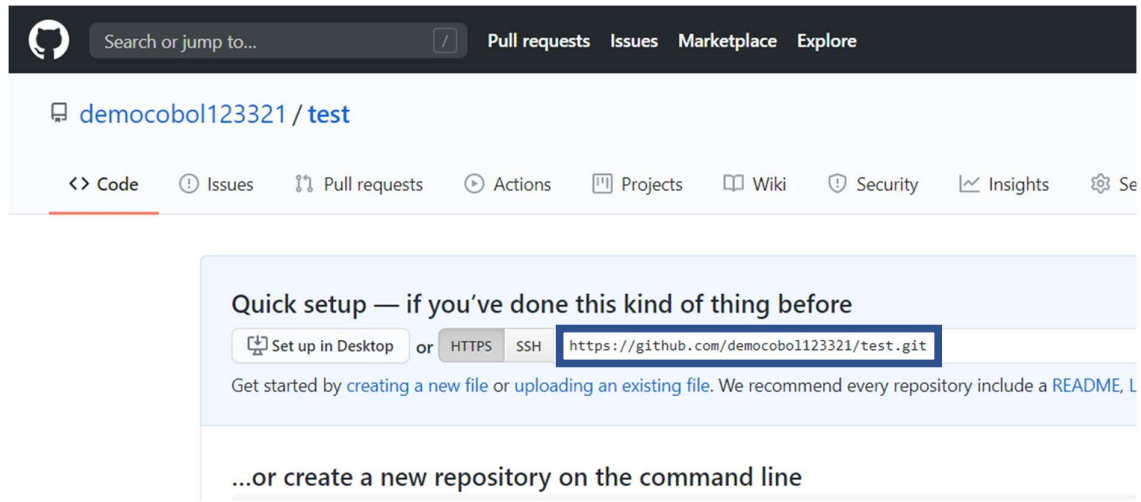
After creating the free account, you will need to create a repository using the Create Repository button.



Pick a short and memorable name for your repository, ensure that Public is selected and then continue.

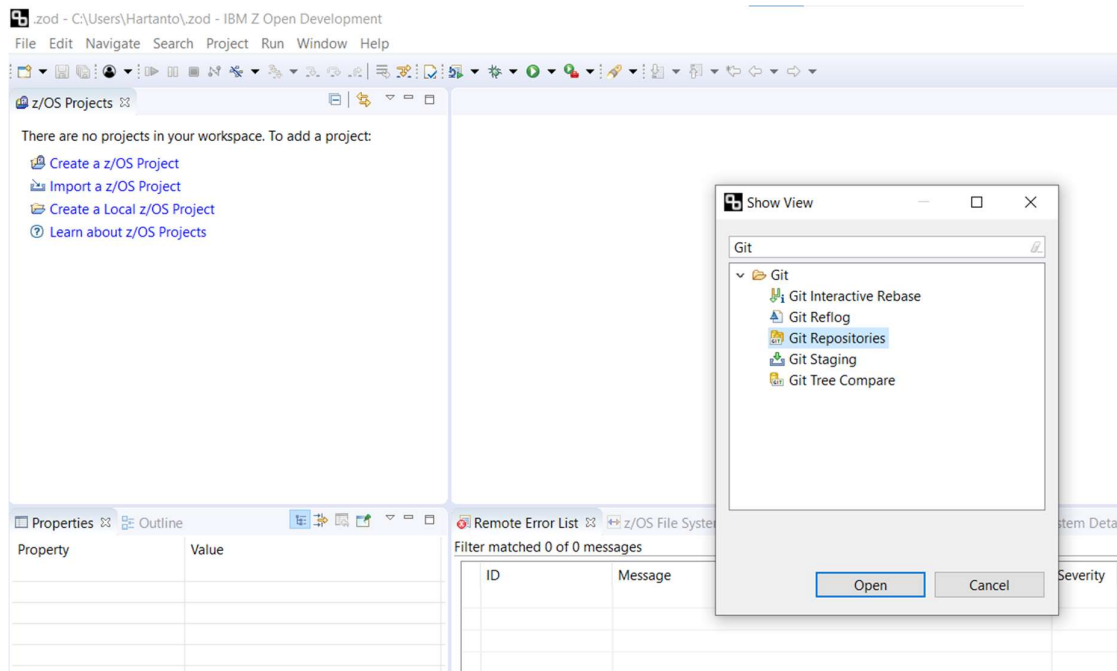


In the next page, copy the HTTPS git link shown below:

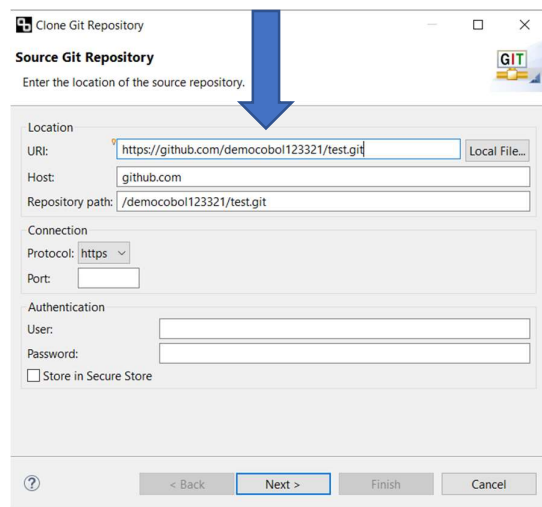
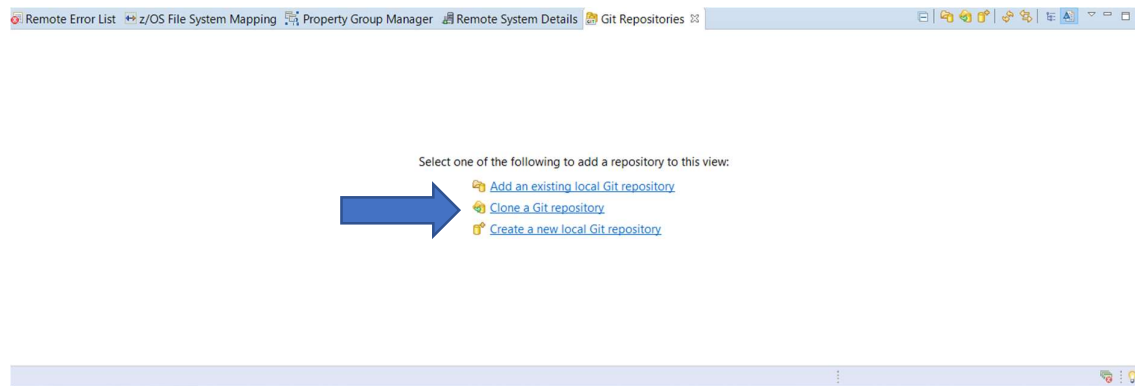


For IBM Developer for z/OS (IDz) or IBM Z Open Development (ZOD)

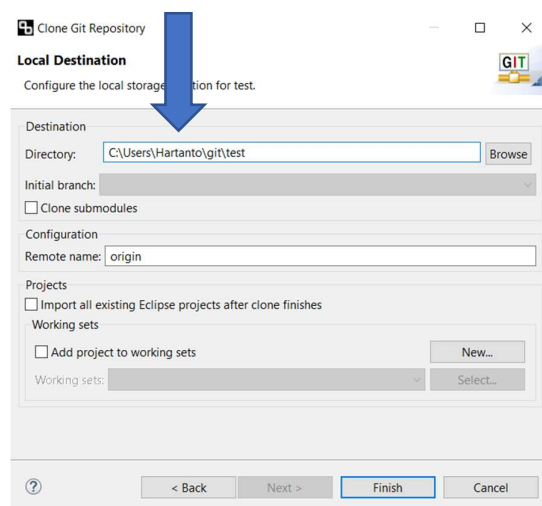
After opening ZOD, go to *Window > Show View > Other* and select the *Git Repositories* view.



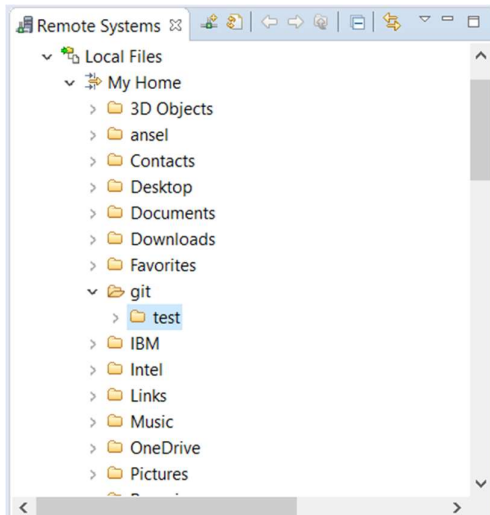
A view will open, then select *Clone a Git repository*



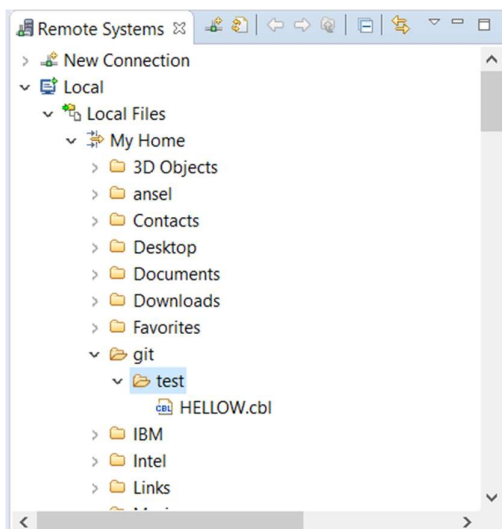
If the Git HTTPS linked you copied from GitHub is still on your clipboard, it will automatically be displayed in the window. Otherwise, copy and paste the link in the URI section.



Here, you will have the opportunity to specify the local file location for your Git repository. To do so, edit the Directory field or leave it as it is.

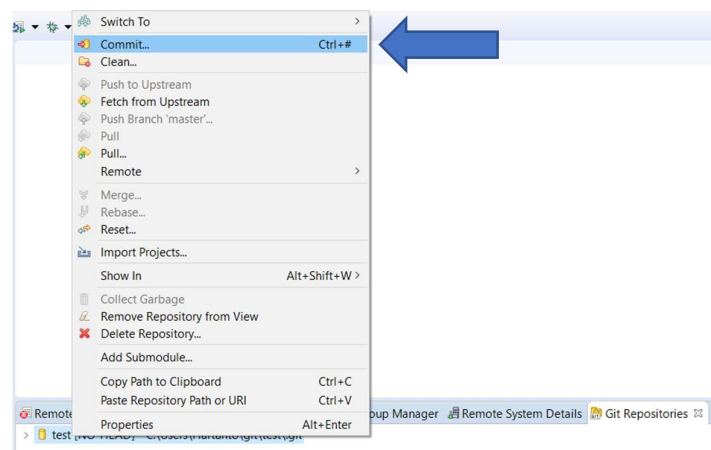


Under the *Remote Systems* view, open the directory you specified above. By default, the in-built *My Home* filter will point to your home directory. Any file you put and any changes to the file here will be considered as changes to your local repository.

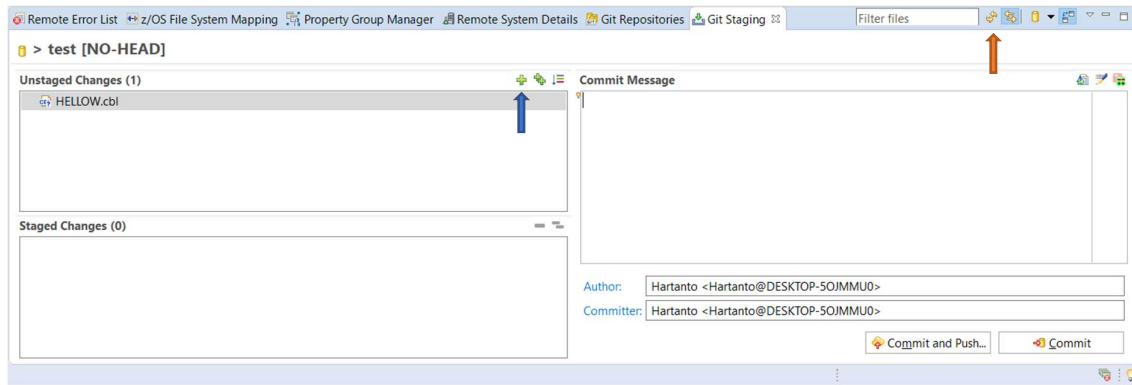


So, let us copy a file over from z/OS here, you can either drag or drop your file into this folder or CTRL+C and CTRL+V it.

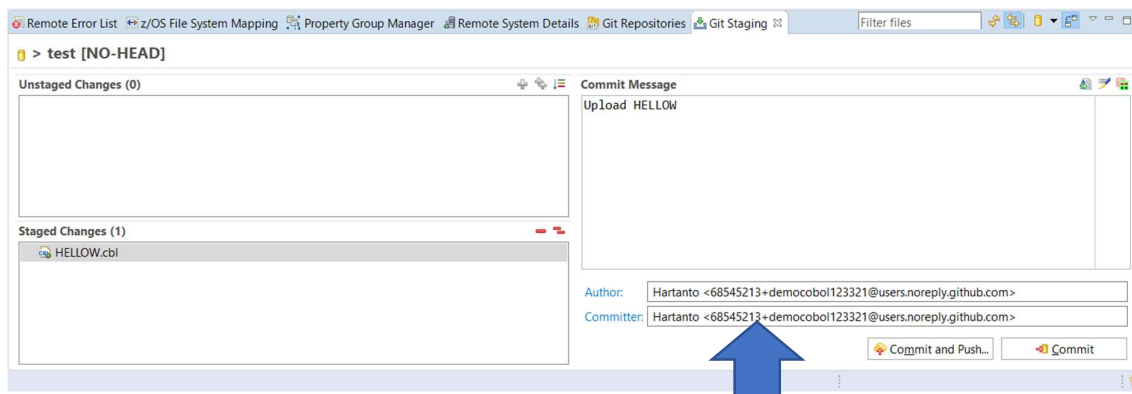
Then, on the *Git Repositories* view, right click on the specified repository and click *Commit*.



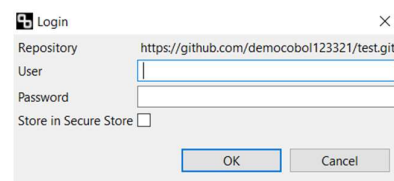
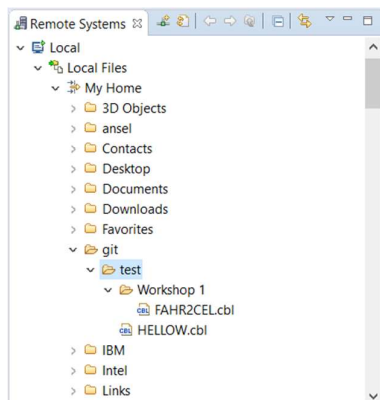
Next, the *Git Staging* view will appear, you should see the file you add or change under the *Unstaged Changes* section. If not, click the refresh button as indicated by the orange arrow below. You will then need to stage the changes by pressing the + button indicated with the blue arrow below. The + button will stage only the selected file. While the ++ button next to it will stage all files.



Now, you will need to add a meaningful Commit Message and add your email to the Author and Committer field. Note the difference between the image above and below.



When you are ready, press *Commit and Push*. When prompted, enter your GitHub.com username and password.

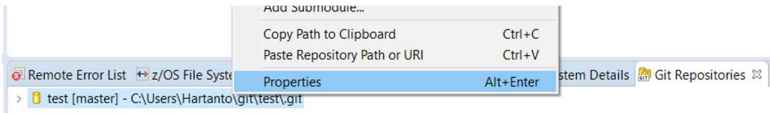


Now, head back to GitHub.com, you should see the changes you made there. You can also commit folders, to do so create a folder on the local repository folder and place a file there. The folder must not be empty. You can then follow the instructions above to commit.

Frequently Asked Questions

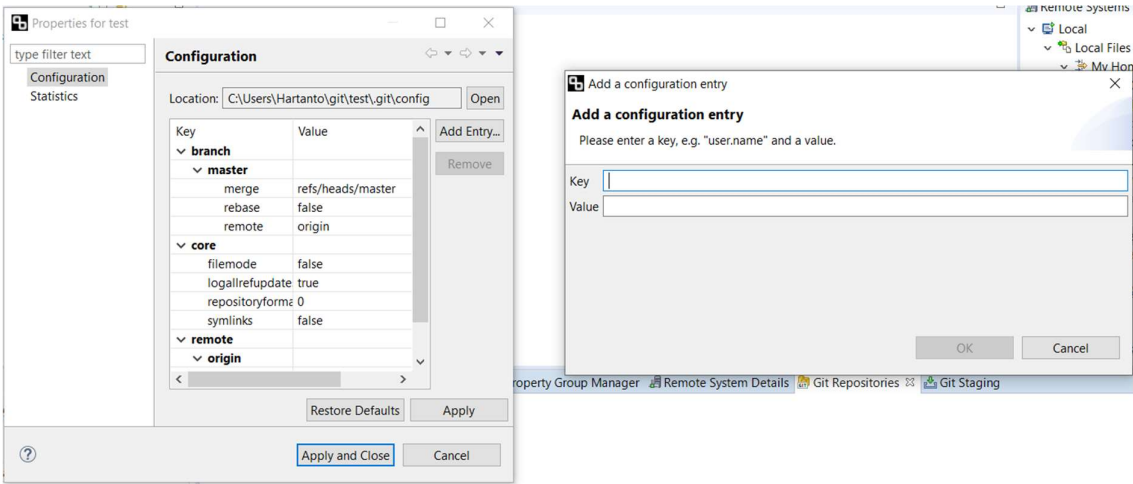
1. I am too lazy to re-type my email address every time. What should I do?

You are in luck! Right click the repository name on the *Git Repository* view and go to and select Properties.



Next you should see a Properties window. Click Add Entry and add the Key Value pair as shown on the table below:

Key	Value
user.name	<<Your Name>>
user.email	<<Your GitHub Email>>



Afterwards, click Apply and Close. Now, new commits will use the specified name and email address! You will need to do this for every repository though.

2. I do not want to reveal my email address. What should I do?

GitHub.com has kindly provided everyone with a private email! Go to GitHub.com, click your profile picture, and go to Settings. Under the Emails tab you should see this:

The screenshot shows the GitHub Settings page with the 'Emails' tab selected. On the left is a sidebar with links: Emails, Notifications, Billing, SSH and GPG keys, Blocked users, Repositories, Organizations, Saved replies, Applications, and Developer settings. The main content area has a header with 'Email address' and a 'Add' button. Below this, the 'Primary email address' section explains that due to email privacy, a private email (68545213+democobo1123321@users.noreply.github.com) will be used for notifications and password resets, while the user's email (@gmail.com) will be used for Git operations. There is a dropdown menu showing '@gmail.com' and a 'Save' button. The 'Backup email address' section explains it's used for security notifications and can also be used for password resets. It has a dropdown menu showing 'Allow all verified emails' and a 'Save' button, with a note to 'Please add a verified email, in addition to your primary email, in order to choose a backup email address.' At the bottom, there are two checkboxes: 'Keep my email addresses private' (checked) and 'Block command line pushes that expose my email' (unchecked). The 'Keep my email addresses private' checkbox has a detailed explanation: 'We'll remove your public profile email and use 68545213+democobo1123321@users.noreply.github.com when performing web-based Git operations (e.g. edits and merges) and sending email on your behalf. If you want command line Git operations to use your private email you must set your email in Git.'

Feel free to use that email address instead :)

3. I do not or cannot use IDz or ZOD. What should I do?

The next section will cover TSO/ISPF! Knowledge of basic command line commands and tools are assumed though.

4. I use Windows and when I commit there are this strange ^M on GitHub.

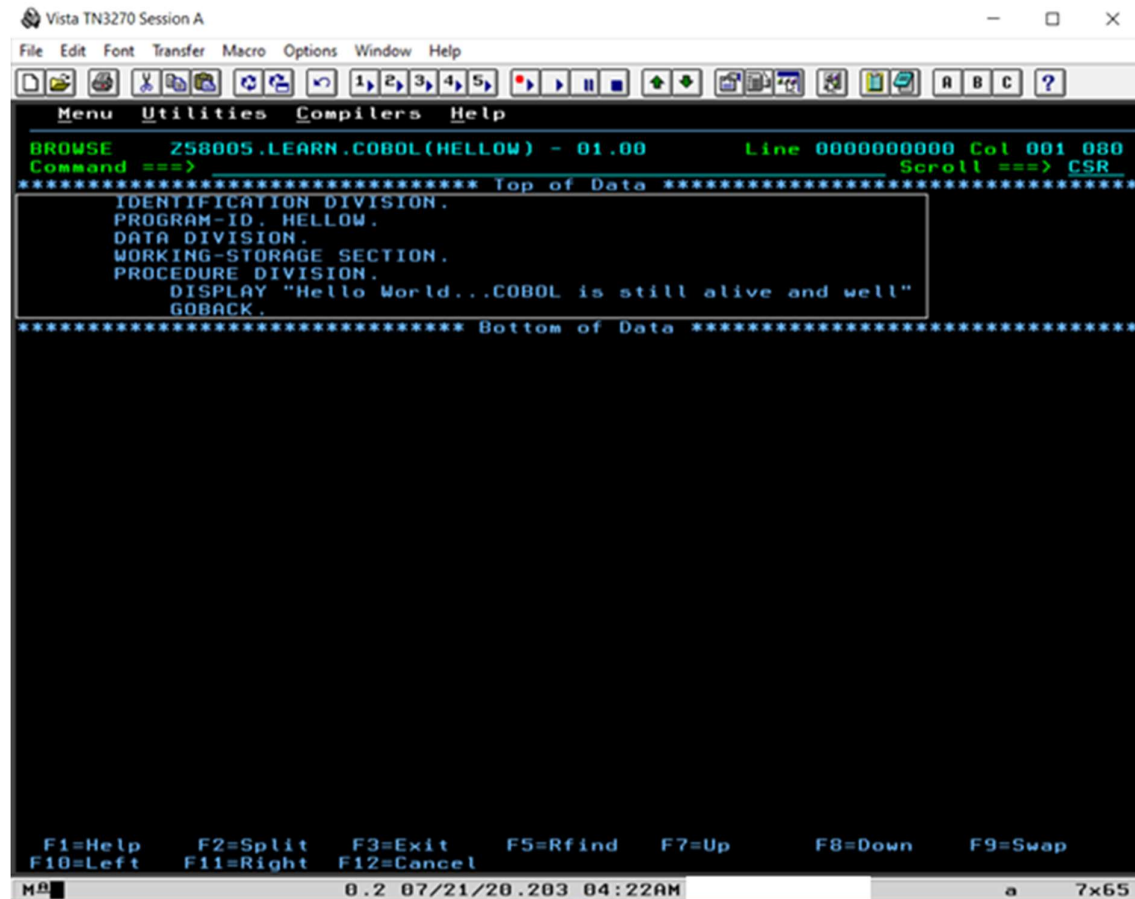
Those are Windows line-ending, which is annoyingly different then other OS. To change that, follow the instruction on FAQ 1 to access the Properties window of your repository, then add this Key Value pair.

Key	Value
core.autocrlf	true

There is this instruction to normalize existing file: <https://docs.github.com/en/github/using-git/configuring-git-to-handle-line-endings#refreshing-a-repository-after-changing-line-endings>, sadly it requires knowledge of command-line Git. Alternatively, you can delete the files, commit the changes, re-include the file and then do commit and push.

For TSO/ISPF – Advanced

With your preferred TN3270 emulator, do a `dslist` or use `=3.4` to browse the PDS member you want to copy over. Draw a rectangle over your code and copy it.



```
BROWSE      ZS8005.LEARN.COBOL(HELLOW) - 01.00      Line 0000000000 Col 001 080
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLOW.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
    DISPLAY "Hello World...COBOL is still alive and well"
    GOBACK.
***** Bottom of Data *****

F1=Help      F2=Split   F3=Exit     F5=Rfind    F7=Up       F8=Down     F9=Swap
F10=Left     F11=Right  F12=Cancel

M 0.2 07/21/20.203 04:22AM a 7x65
```

Paste the code to a text editor and save it. Next, you will have the option to use a Git terminal or the GitHub.com web application. To use the Git terminal, you will need to have Git installed on your workstation. By default, if you use Linux, Git should be pre-installed. If not, you can download it at git-scm.com/downloads.

Similar as with IDz/ZOD, do a `git clone` command followed with the Git HTTPS link obtained above (e.g `git clone https://github.com/democobol123321/test.git`).

By default, it will copy your remote directory to a folder called `test` located at the directory where you execute the `git clone` command. Then, move the code file over to the `test` folder and stage it using `git add -A`. The `-A` flag will add stage all new and changed file inside the folder. Alternatively, you can also specify the filename of the file you want to stage.

If you have not, you will need to specify the Git username and email. You can do so using the commands below:

- `git config --global user.email "<<Your GitHub email>>"`
- `git config --global user.name "<<Your Name>>"`

Omit the `--global` flag to only specify the name and email for that specific repository.

Next, commit the changes using `git commit -m "Enter Commit Message Here"`. Make sure you use a meaningful commit message.

Afterwards, do `git push origin master` to push your code to GitHub.com. Enter your username and password when prompted.

For documentation, refer to git-scm.com/docs.

Alternatively, there is an ISPF interface to Git being developed! Check it out at zigi.rocks.

I have created the GitHub repository and uploaded my code! Now what?

All that left for us to do is learn from each other code!

This tutorial is intended as a basic guide to Git. For more in-depth tutorial, refer to guides.github.com/introduction/git-handbook.

Questions regarding this document should be directed to Hartanto through email, Slack, or any other social media you deemed fit.

LinkedIn: <https://www.linkedin.com/in/hartantoariowidjaya>

IBM Z Slack: <https://ibm-systems-z.slack.com/team/W015K1UA16V>

Master the Mainframe Slack: <https://mtmcompetition.slack.com/team/U013M5YLF7V>

Disclaimer

The information provided in this document is for informational purposes only and is provided as is without warranty of any kind. The author shall not be responsible for any damages arising out of the use of these materials.