Steven Grutman

7/27/17

Technical Writing: ENGL 393

# My Audience: Textron Systems

Currently, I am a software engineering intern at Textron Systems, a company that works closely with many branches of the military to create weapons testing products. My main job has been updating the graphical user interfaces of both the embedded software and windows applications for a particular disclosed project. Many of Textron's employees have strong backgrounds in electrical, mechanical, and software engineering. Textron also employs many professionals with varied backgrounds ranging from business experts to human resource specialists. Regardless of training or position, all employees are familiar with technological terminology simply due to the company's product line. My superiors have wanted to incorporate version control software into the company's infrastructure for firmware projects to replicate what software engineers do with their projects, making this a perfect audience for version control instructions for firmware.

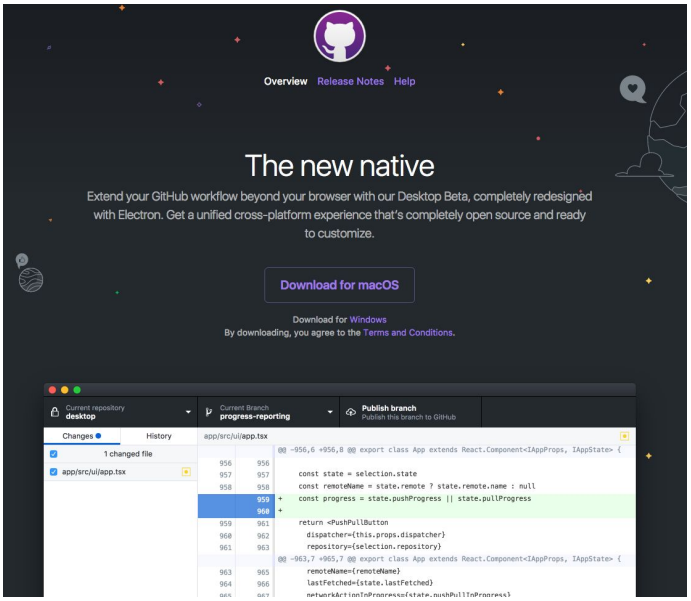
# Continuously Committing Firmware Projects to Github on Windows

## Introduction

- Firmware projects are hard to keep track of due to their ever changing nature, constant need to be rebuild, large size and scope of files in various locations, and a plethora of hardware choices, each with different languages and softwares. All firmware builds have superfluous intermediary files that take up significant storage space. These intermediary files can be removed without damaging the firmware project, because the next time the project is built, these files are recreated as the editor runs through the build process behind the scenes. Github is a version control software that allows the user(s) to keep track of version history of their projects and more easily collaborate with the same coding project.
- Target Audience: Firmware Engineers and Engineering Leadership at Textron Systems
- This document will explain how to create a new repository, utilize Github's version control software, remove said intermediary files, and maintain working solutions in your Github repository.

WARNING: Deleting files yourself after creating archive of project could have bad consequences. Ensure your project build properly before using on hardware.

## Hardware and Software Requirements



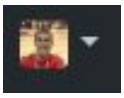

- Windows computer
    - Latest version of Windows
    - Access to the internet
- Latest version of Github GUI for Windows
    - A Github account
    - Accessible for download at:

      https://desktop.github.com/ (right)
- A functioning firmware solution
    - Respective firmware building software

## Instructions

### Create Repository

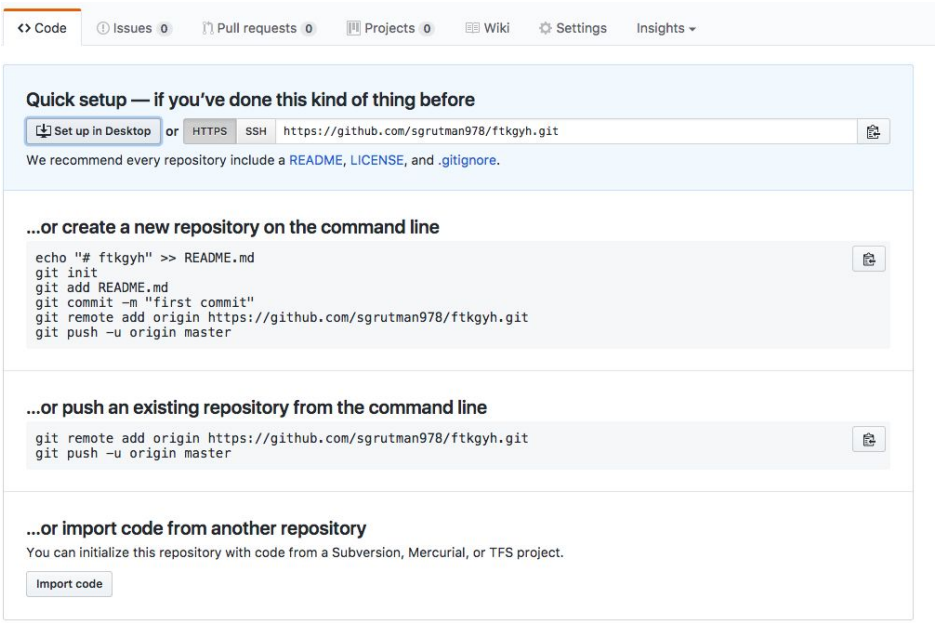
1. Go to github.com
2. Create an account
3. Navigate to your profile page
    a. Click on you picture icon in the top left corner with an arrow next to it



    b. Click “Your Profile”
4. Click “Repositories”
5. Click the green “New” button
6. Setup form
    a. Type in a name for your repository and an optional description
    b. Chose if the repository is public or private
    c. Click the checkbox labeled “Initialize this repository with a README”
    d. Leave “Add .gitgnore” and “Add a license” at “None

e. Click “Create Repository”

7. Open the Github program on your computer and Log in to your account
8. Navigate back to the website and click “Set up in Desktop”
    a. This will prompt you to open the Github Desktop Program
    b. Select “Open”




9. The Github app will open
    a. Click “Clone”
    b. This will begin the cloning process of extracting your new repository to be stored on your local machine
    c. Any changes you make to your Github repository in the cloud can be pulled to your local machine
    d. Similarly, any change made on your local machine can be pushed to the cloud as a commit, for which instruction is provided below

**New Commit**

1. Open your firmware solution in the building software you typically use

2. Create an archive of your software solution
    a. Location of archive button will vary based on building software utilized
3. Save the archive in the location of your choice
4. Open the archive files (.qar file) by double clicking (this will locally extract its contents)
    a. Extraction only contains non-intermediary files necessary to rebuild your solution
5. Ensure intermediary files are not present and necessary build files are present
6. Open the Github program and Log in to your account if you have not already done so
7. Ensure that your desired repository to edit is your “Current Repository” in the top toolbar
8. Chose the branch you want to push your changes to (master is ideal)
9. Chose the files you would like committed
    a. On the left hand side of the window, it will display the files being modified, added, or removes.
    b. You have the option to check and uncheck files you wish to be committed and not
    c. On the right, you can see the code changes made in each individual file
10. Add a commit summary and description
11. Click the blue commit button in the bottom left corner
12. To push the commit to the cloud to be viewed online, press “Publish Branch” on top