Using Git to Manage I&T Projects and for Code & Design File CMVC

"Like a jigsaw puzzle: you have to make the pieces fit without getting out the scissors." – Dr. Karl Maurer – On translating Greek sentences [1]

http://www.stsc.hill.af.mil/resources/tech_docs/gsam4.html

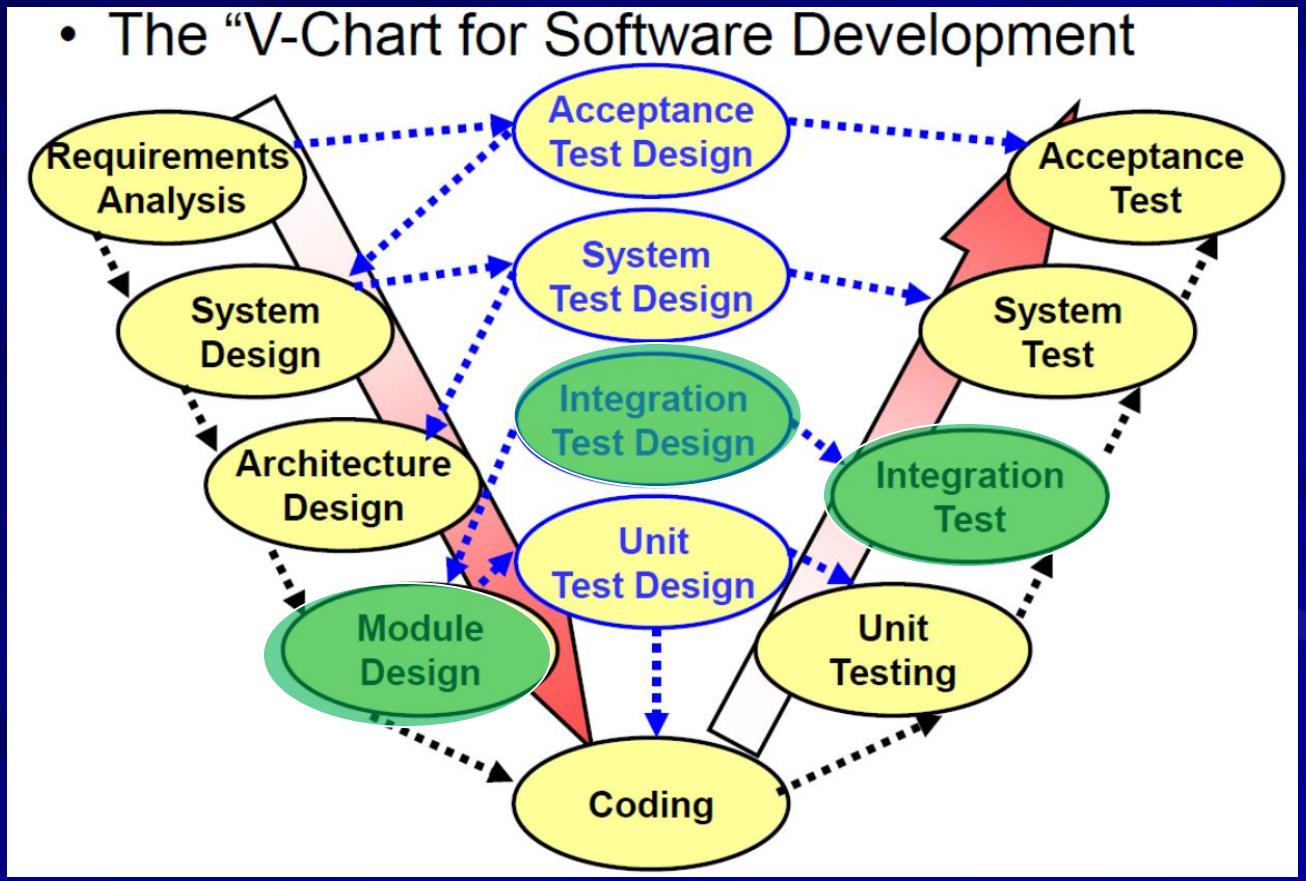
GitHub Use

October 28, 2014

© Sam Siewert

Integration and Test

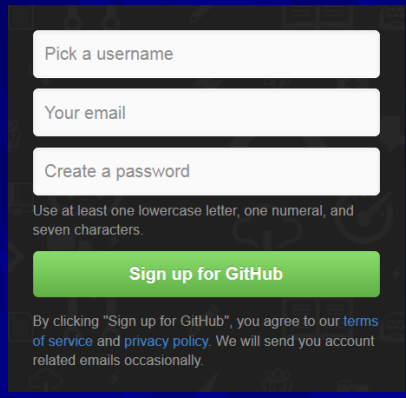
- Integrate Software Modules [units] and Hardware Components into Sub-systems – CMVC is Critical at all Phases, but Can't Live without it During I&T!!
- Test Focus on Interfaces [Function, Message, Shared Memory, Hardware], Protocols, and Interoperability of Modules

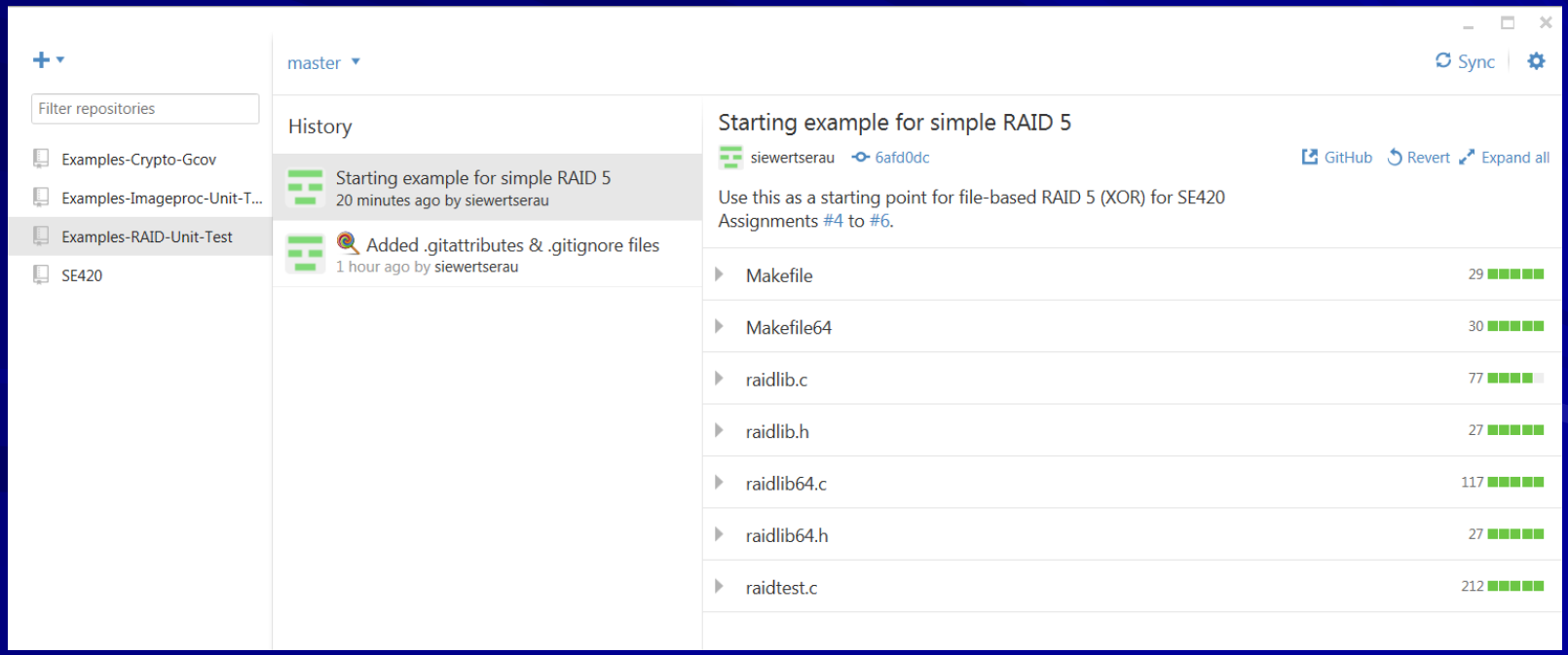


© Sam Siewert

GitHub and GitHub Desktop Primer

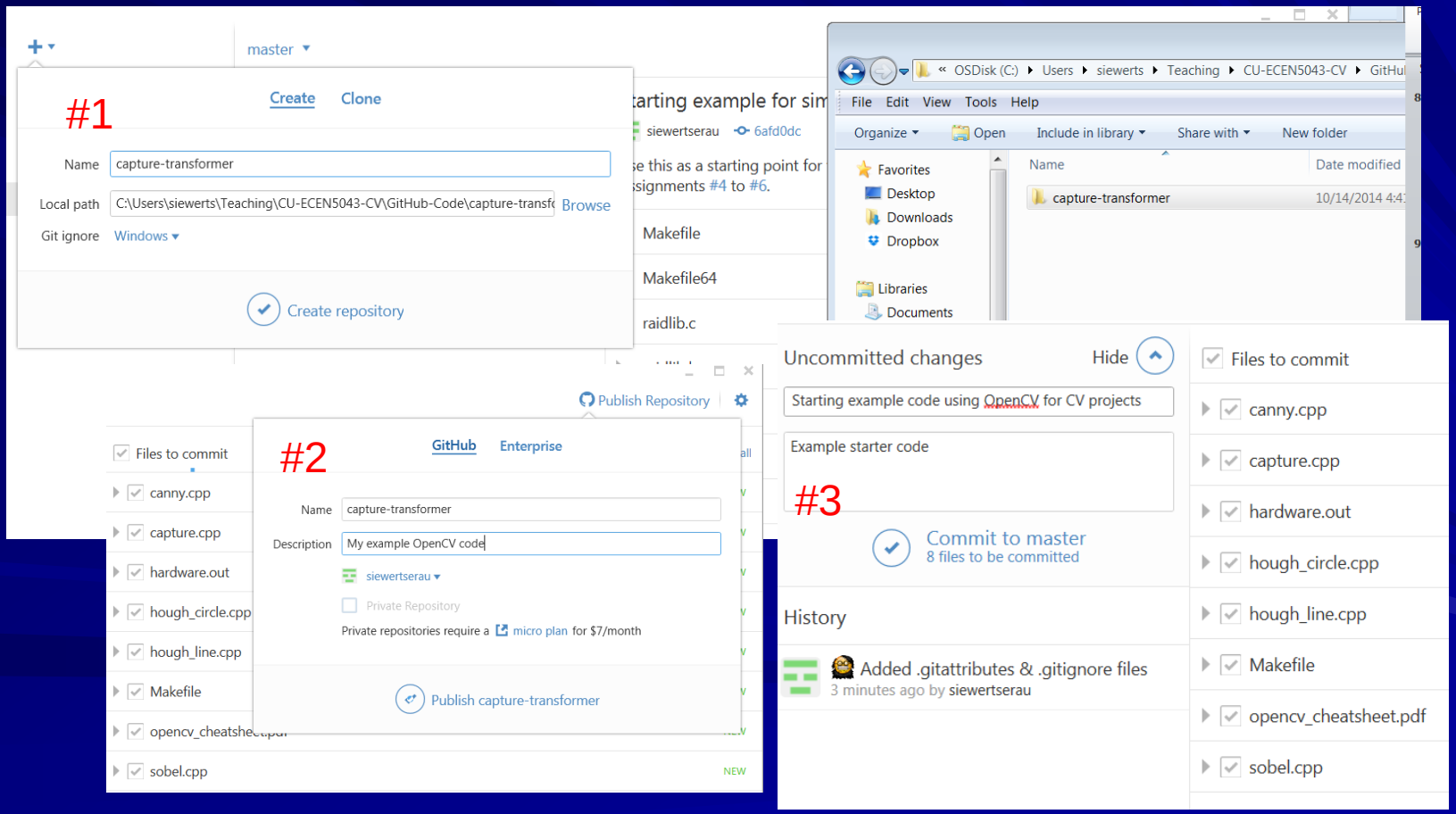
- Use CMVC from Now On
  - Install GitHub Desktop on Windows or Mac
  - Create GitHub Account - https://github.com/
  - Code is Public for FREE Use
  - Clone from siewertserau or PRClab or Desktop
    - git clone https://github.com/siewertserau/Examples-RAID-Unit-Test.git
    - You need my permission to push changes back, but anyone can pull





GitHub New Project

- Create Your Own SE420 Student Project
- Start GitHub Desktop, Download Code from Web, Drag Folder over to Create new repository
- Publish for Use by Others [on your team, public]
- Push to GitHub so Others can Clone on Your Team [Pull]

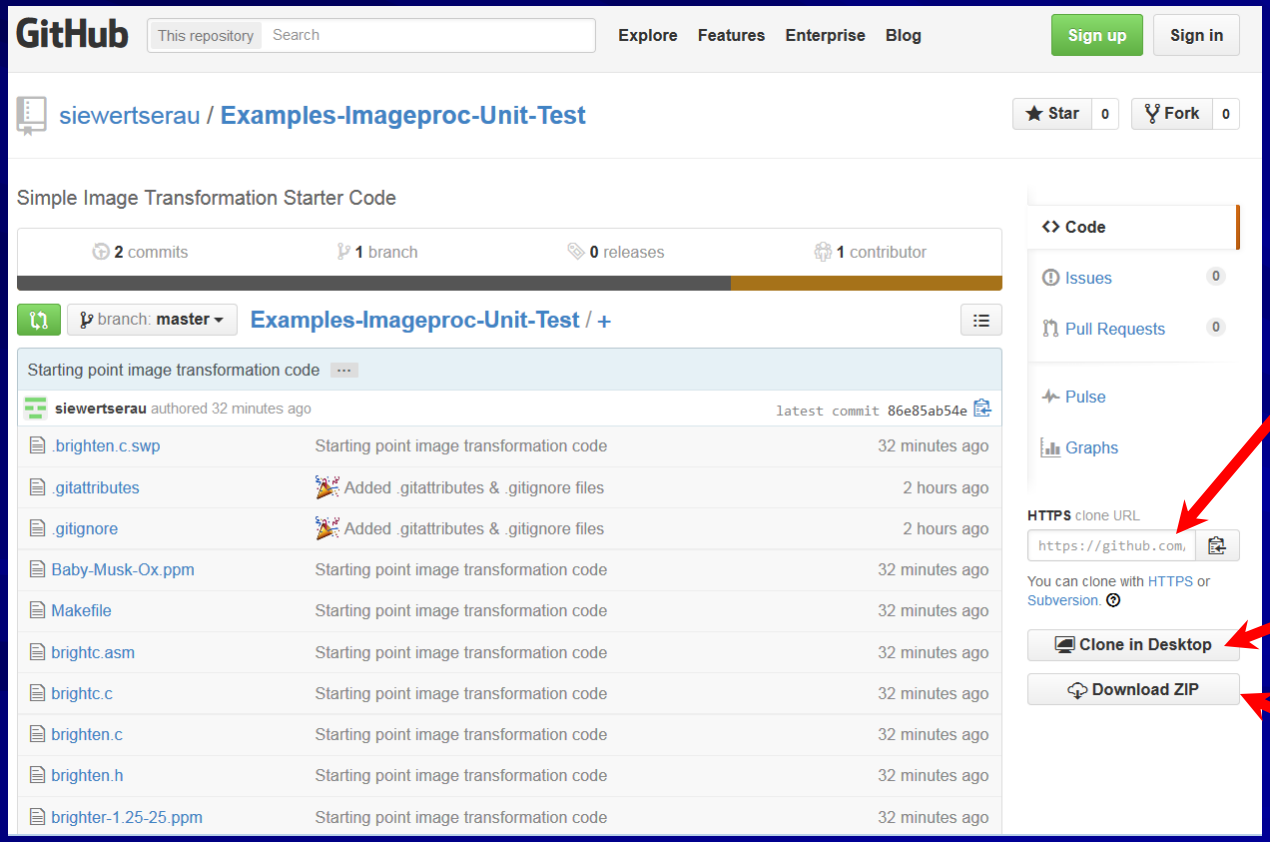


© Sam Siewert

University Machines

- I Requested Git client for Cloning and CMVC on EE Servers

- See Cloning Path on GitHub Desktop / Web – View on GitHub – Use SSH or HTTPS to Clone [SSH is Preferred]



Grab clone path here

Clone to Desktop here

Get Archive here

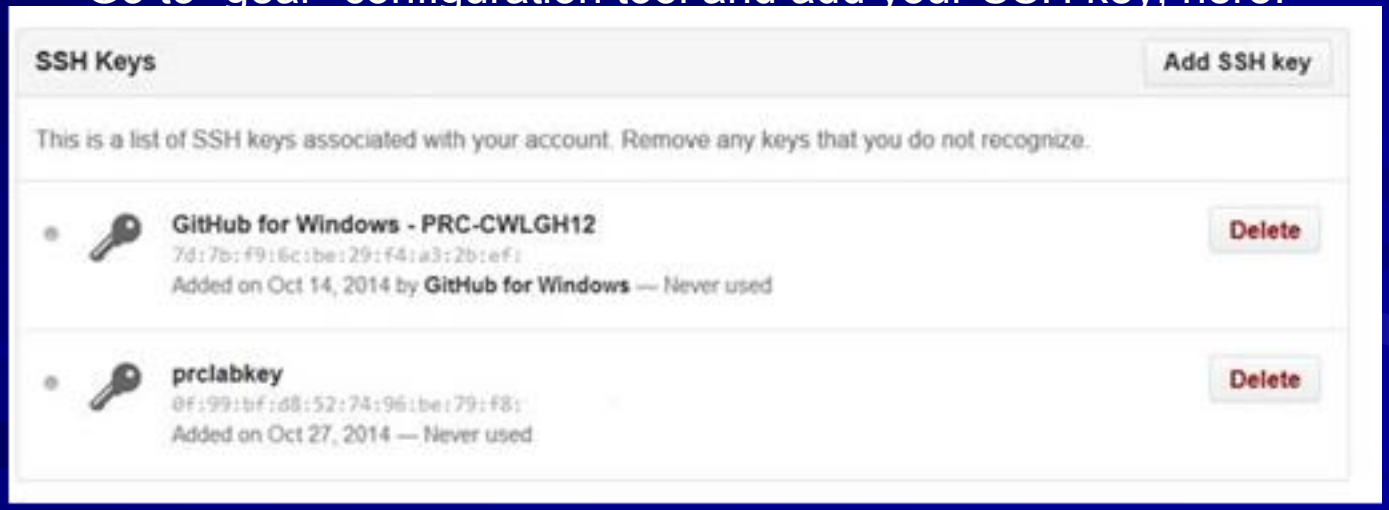
Example Use of Git for CMVC

Simple RAID Code Example at ERAU

© Sam Siewert

Git Example – Use SSH Tunnel

- Set up your SSH key on PRClab and copy it to your key-ring on GitHub (described here too - https://help.github.com/articles/generating-ssh-keys/):
  - ssh-keygen -t rsa -C siewerts@erau.edu [follow prompts as described on web page above]
  - ssh-agent –s [start ssh agent for your current login – do this on each login]
  - cat ~/.ssh/id_rsa.pub [copy the output that should be something like "ssh-rsa ********** siewerts@erau.edu", where ****'s are my encrypted key, so your's will be unique and different
  - got to github.com/siewertserau [replace with your github username]
  - Go to "gear" configuration tool and add your SSH key, here:



note that I have erased portions of my keys for security

Git Example – SSH Cloning

- Now that you have your SSH key generated on PRClab on your GitHub keyring, you can clone with SSH instead of the HTTPS clone
  - Now, for me, I just copied from the web page and then issued the clone command on PRClab "git clone git@github.com:siewertserau/Examples-Imageproc-Unit-Test.git" and I'm prompted for my SSH passphrase now, which is great, and if I enter it correctly, I will get an SSH clone sandbox.
  - It should look like this:

Use SSH PATH

```
%git clone git@github.com:siewertserau/Examples-Imageproc-Unit-Test.git
Initialized empty Git repository in /home/facstaff/siewerts/se420/clones/Examples-Ima
geproc-Unit-Test/.git/
Enter passphrase for key '/facstaff/siewerts/.ssh/id_rsa':
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 19 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (19/19), 1.61 MiB | 1.35 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Git Example – Sandbox Work

- Now edit your files, add files, build, test and when your happy, do “git add *” to add any new files you copied into your sandbox, and do a pull to make sure you are current.

```
%git pull
Enter passphrase for key '/facstaff/siewerts/.ssh/id_rsa':
Already up-to-date.
%
```

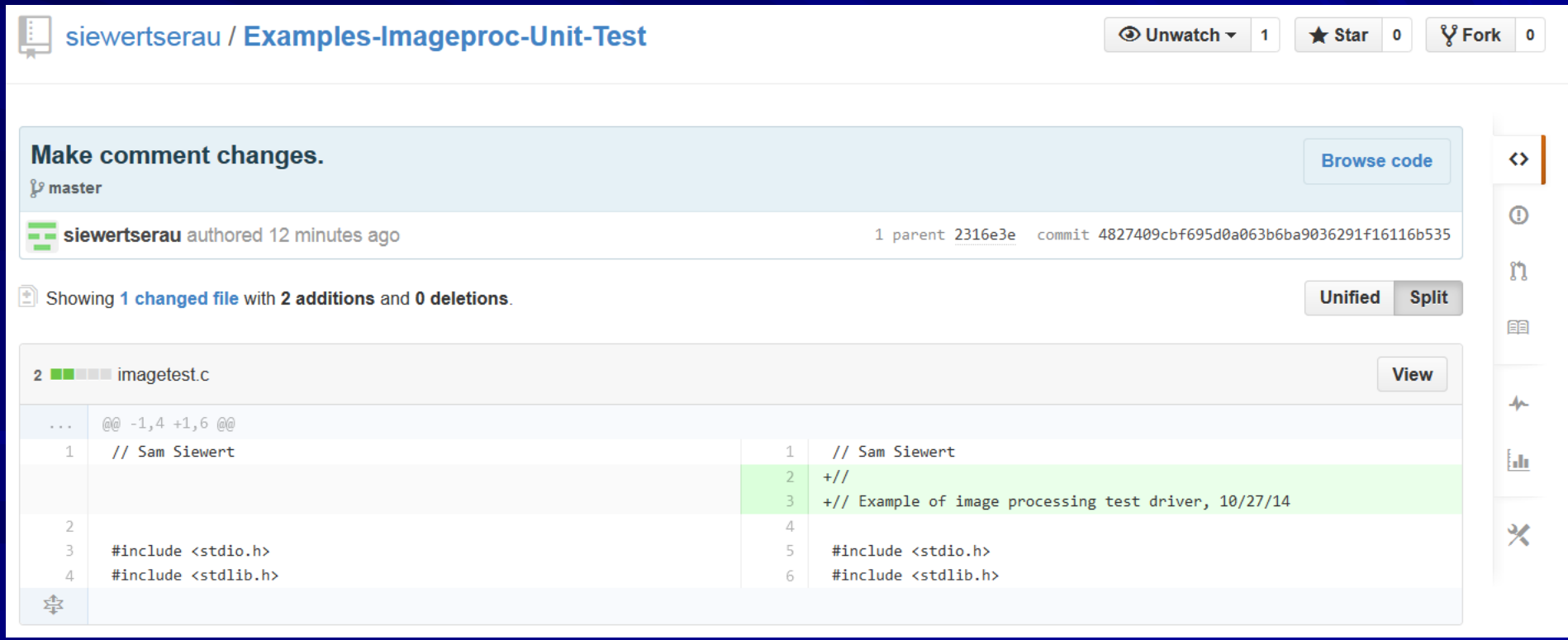
- Now do “git commit –a” to commit the change set to your clone sandbox and add a meaningful comment and after do “git push”, which should look like:

```
%git commit
[master 2316e3e] Clean up of swapfile and add comments.
 1 files changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 .brighten.c.swp
%git push
Warning: Permanently added the RSA host key for IP address '192.30.252.128' to the li
st of known hosts.
Enter passphrase for key '/facstaff/siewerts/.ssh/id_rsa':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 235 bytes, done.
Total 2 (delta 1), reused 0 (delta 0)
To git@github.com:siewertserau/Examples-Imageproc-Unit-Test.git
   9cbfca3..2316e3e  master -> master
```

© Sam Siewert

Git Example – Browse Change History

- Use Browse code and Side-by-side “Split” view to Compare Old and New Versions of a file
- Browsing Code Change History is One of the MOST Powerful Features of CMVC



Binary File Example

- Make changes, build, test and then commit and push

```
%git commit -a
[master f782300] Updates to add binary file example to use in class.
 1 files changed, 36 insertions(+), 0 deletions(-)
%git push
Enter passphrase for key '/facstaff/siewerts/.ssh/id_rsa':
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 800 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
To git@github.com:siewertserau/Examples-RAID-Unit-Test
   6afd0dc..f782300  master -> master
%
```

- Now Run to Generate Chunk files

```
RAID Operations Performance Test
Test Done in 13951 microsecs for 1000 iterations
71679.449502 RAID ops computed per second

%ls
Chunk1.bin  Chunk4.bin          Makefile    raidlib.h    raidlib64.h  raidtest.o
Chunk2.bin  Chunk4_Rebuilt.bin  Makefile64  raidlib.o    raidtest
Chunk3.bin  ChunkXOR.bin        raidlib.c   raidlib64.c  raidtest.c
%
```

© Sam Siewert

Git RAID Example - Browse

- To See What I Changed, Browse
- First, 4 New “+” Lines for Includes for Simple file I/O
- Next, 2 New Lines to Declare Simple file Descriptors

36 raidtest.c — View

```
@@ -5,6 +5,10 @@
 #include <assert.h>                          #include <assert.h>
 #include <string.h>                          #include <string.h>
                                              +// recommended includes for use with simple file I/O
                                              +#include <sys/stat.h>
                                              +#include <fcntl.h>
                                              +
 #ifdef RAID64                               #ifdef RAID64
 #include "raidlib64.h"                      #include "raidlib64.h"
 #define PTR_CAST (unsigned long long *)     #define PTR_CAST (unsigned long long *)
@@ -64,6 +68,8 @@ void dumpBuffer(unsigned char *bufferToDump)
 int main(int argc, char *argv[])            int main(int argc, char *argv[])
 {                                           {
         int idx, LBAidx, numTestIterations, rc;     int idx, LBAidx, numTestIterations, rc;
                                             +        int written=0, fd[5];
                                             +        int fdrebuild;
         double rate=0.0;                            double rate=0.0;
         double totalRate=0.0, aveRate=0.0;          double totalRate=0.0, aveRate=0.0;
         struct timeval StartTime, StopTime;         struct timeval StartTime, StopTime;
@@ -120,6 +126,36 @@ int main(int argc, char *argv[])
          assert(memcmp(testRebuild, testLBA4, SECTOR_SIZE) ==0);     assert(memcmp(testRebuild, testLBA4, SECTOR_SIZE) ==0);
```

Git RAID Example – Test Case Update

- New Block of Binary Output for External Verification
- Can “diff” and use “od –t x1” on files to Examine

Line 129 to 158 is New
Line 124 and 160 Sync Up Again

```
123
124         // TEST CASE #1
```

```
129 +       // Adding TEST CASE #0 Feature to dump binary files with each 512byte LBA
130 +       // so I can do a "diff" on them, dump with od -x 2, etc.
131 +       //
132 +       fd[0] = open("Chunk1.bin", O_RDWR | O_CREAT, 00644);
133 +       fd[1] = open("Chunk2.bin", O_RDWR | O_CREAT, 00644);
134 +       fd[2] = open("Chunk3.bin", O_RDWR | O_CREAT, 00644);
135 +       fd[3] = open("Chunk4.bin", O_RDWR | O_CREAT, 00644);
136 +       fd[4] = open("ChunkXOR.bin", O_RDWR | O_CREAT, 00644);
137 +
138 +       written=write(fd[0], &testLBA1[0], SECTOR_SIZE);
139 +       assert(written == SECTOR_SIZE);
140 +       written=write(fd[1], &testLBA2[0], SECTOR_SIZE);
141 +       assert(written == SECTOR_SIZE);
142 +       written=write(fd[2], &testLBA3[0], SECTOR_SIZE);
143 +       assert(written == SECTOR_SIZE);
144 +       written=write(fd[3], &testLBA4[0], SECTOR_SIZE);
145 +       assert(written == SECTOR_SIZE);
146 +       written=write(fd[4], &testPLBA[0], SECTOR_SIZE);
147 +       assert(written == SECTOR_SIZE);
148 +
149 +       for(idx=0; idx < 5; idx++) close(fd[idx]);
150 +
151 +
152 +       // Now, do the same for the rebult 4th chunk
153 +       fdrebuild = open("Chunk4_Rebuilt.bin", O_RDWR | O_CREAT, 00644);
154 +       written=write(fdrebuild, &testRebuild[0], SECTOR_SIZE);
155 +       assert(written == SECTOR_SIZE);
156 +       close(fdrebuild);
157 +
158 +
159
160         // TEST CASE #1
```

© Sam Siewert

Basic Binary File Dump and Compare

- Chunk1.bin, …, Chunk4.bin are Data Files in 4+1 RAID-5 Set
- ChunkXOR.bin is the XOR Parity File in the RAID-5 4+1
- Chunk4_Rebuilt.bin is the Recovered Data in Test Case #0
- Identical Chunks, so XOR=0

```
%od -t x1 ChunkXOR.bin
0000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0001000
%od -t x1 Chunk1.bin
0000000 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000020 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000040 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000060 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000100 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000120 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000140 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000160 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000200 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000220 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000240 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000260 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000300 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000320 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000340 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000360 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000400 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000420 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000440 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000460 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000500 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000520 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000540 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000560 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000600 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000620 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000640 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000660 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000700 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000720 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000740 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000760 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0001000
%od -t x1 Chunk2.bin
0000000 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000020 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000040 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000060 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000100 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000120 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000140 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000160 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000200 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000220 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000240 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
0000260 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 23
0000300 23 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
0000320 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75
0000340 76 77 78 79 7a 41 42 43 44 45 46 47 48 49 4a 4b
```

© Sam Siewert