Version Control Tutorial using TortoiseSVN and TortoiseGit

Christopher J. Roy, Professor

Virginia Tech, <u>cjroy@vt.edu</u>

This tutorial can be found at:

https://www.aoe.vt.edu/content/dam/aoe_vt_edu/people/faculty/cjroy/Software-Resources/tortoise-svn-git-tutorial.pdf

Software engineering is critical for accurate and reliable simulation codes in scientific computing. However, most scientists and engineers receive little or no formal training in software engineering. Configuration management is an important component of software engineering which deals with the control and management of software products. The key aspects include using version (or revision) control for source code and other software artifacts, recording and tracking issues with the software, and ensuring backups are made. This tutorial will focus on the first of these: software version control. While most software engineering practices are critical only for large software development efforts, every software project, regardless of how large or small, should use a version control system for the source code.

Version Control

Version control tracks changes to source code or any other files. A good version control system can tell you what was changed, who changed it, and when it was changed.

It allows a software developer to undo any changes to the code, going back to any prior version, release, or date. This can be particularly helpful when a researcher is trying to reproduce results from an earlier paper or report and merely requires documentation of the version number. Version control also provides a mechanism for incorporating changes from multiple developers, an essential feature for large software projects or any projects with geographically remote developers.

Some key concepts pertaining to version control are discussed below. Note that the generic descriptor "file" is used and could represent not only source code, but also user's manuals, software tests, design documents, web pages, or any other item produced during software development.

- **repository** a location where the current and all prior versions of the files are stored; in distributed version control systems, there is a master repository which can be copied (or "cloned") locally for development
- **working copy** the local copy of a file from the repository which can be modified and then checked in or "committed" to the repository
- **check-out** the process of creating a working copy from a repository (either the current version or an earlier version)
- **check-in** a check-in or commit occurs when changes made to a working copy are merged into a repository
- **push** the merging of changes from a local repository to the master repository (for distributed version control systems)
- **diff** a summary of the differences between two versions of a file, often taking the form of the two files side-by-side with differences highlighted

- **conflict** a conflict occurs when two or more developers attempt to make changes to the same file and the system is unable to reconcile the changes (note: conflicts generally must be resolved by either choosing one version over the other or by integrating the changes from both into the repository by hand)
- **update** merges recent changes to a repository into a working copy (for centralized version control systems)
- **pull** merges recent changes to the master repository into a local repository (for distributed version control systems)

The basic steps that one would use to get started with a centralized version control system (such as Subversion/SVN) are as follows:

- 1. Create a repository
- 2. Import a directory structure and/or files into the repository
- 3. Check-out the repository version as a working copy
- 4. Edit/modify the files in the working copy and examine the differences between the working copy and the repository (i.e., diff)
- 5. Check-in (or commit) the changes to the repository

The basic steps that one would use to get started with a distributed version control system (such as Git) are as follows:

- 1. Create an original "master" repository
- 2. Clone the repository to allow development
- 3. Add a directory structure and/or files into the local repository

- 4. Check-in or "commit" the directory structure and/or files to the local repository
- 5. Push these modifications of the local repository into the master repository
- 6. Edit/modify the files in the working copy and examine the differences between the modified files and those in the repository (i.e., diff)
- 7. Check-in or "commit" the file modifications to the local repository
- 8. Pull any modifications from the original repository into the local repository (in case someone has modified some files and your files are out of date)
- 9. Push these modifications of the local repository into the master repository

1 Recommended Version Control Systems

There is a wide array of version (or revision) control systems available to the software developer. We will focus on free, open-source version control systems. One up front choice you will have to make is whether you want a centralized repository (e.g., Subversion, CVS, Vesta) or a distributed repository (e.g., Git, Mercurial, Bazaar). A detailed list of both open source and proprietary version control software can be found at: en.wikipedia.org/wiki/List_of_revision_control_software

The most popular, freely-available, centralized version control system is Subversion (SVN), which is generally seen as the successor to CVS. Subversion is also freelyavailable (at <u>subversion.tigris.org</u>), is open-source and actively maintained, and has a very useful, free book available online (Collins-Sussman et al., 2007). Subversion has a number of easy-to-use Graphical User Interfaces (GUIs) for all major computing platforms. Eddins (2006) recommends using TortoiseSVN (<u>tortoisesvn.tigris.org</u>) on Windows platforms (which integrates into Windows Explorer) and RapidSVN (<u>rapidsvn.tigris.org</u>) on Linux and Macintosh platforms.

The leading distributed version control systems are Git and Mercurial; however we will focus on Git. The main difference between Subversion and Git is that while SVN provides for a single, centralized repository, Git allows a number of distributed repositories that are all tied in to a master repository. Thus Git makes it easier for a number of developers to perform their work on their own projects then integrate these changes back into the master repository once it appears to be working correctly. This helps prevent developers from "breaking" the main version of the code during development. Another advantage of the distributed repository system is that one does not need network access to the master repository in order to do development. To access Git from a Unix/Linux command line, please see the main <u>Git web site</u> which contains documentation, <u>Git cheatsheets</u>, and a 15 minute <u>Git tutorial</u>. A useful Windows GUI for Git is <u>TortoiseGit</u>, which is based on TortoiseSVN discussed earlier. The examples given in the next two sections were developed using TortoiseSVN and TortoiseGit, respectively.

2 TortoiseSVN Tutorial

The following tutorial on the Subversion version control system was created using TortoiseSVN version 1.4.5 on a computer running Microsoft Windows.

2.1 Creating a Repository

Determine a location for the repository, ideally on a server which is automatically backed up. Create a folder with the name of the repository; in this example the repository is called "VVCS-Example." Right click on the folder name, choose "TortoiseSVN" (which is integrated into the Microsoft Windows Explorer menu), then "Create Repository Here." Choose the Native Filesystem, then you should see the message "Repository Successfully Created."

🔾 🔍 🔻 🐌 🕨 Computer 🕨 SW_Preload (C:) • SVN-Repository •		🕶 🐓 Search	
🖢 Organize 🔻 🏢 Views 🔻 🛄 Open	👷 Share	_		
lame	Date modified	Туре	Size	
📙 VVCS-Example	1/26/2008 9:40 AM	File Folder		
Open Explore Search Open as Notebook in OneNote				
Share	_			
R TortoiseSVN	Repo-browser			
Groove Folder Synchronization Restore previous versions	Export Create repository	here		
b Combine supported files in Acrobat	🔊 Import			
Scan for Viruses	? Help			
Send To	 Settings About 			
Cut Copy				
Create Shortcut Delete Rename				
Properties	_			

Creating a repository

2.2 Importing a File into the Repository

Right click on the directory containing the file(s) and/or directory structure you wish to import to the repository (note, the directory that you click on will not be imported). Here we will simply be importing the file "code1.f" from directory "Code1." This code creates a 17×17 two-dimensional Cartesian grid for *x* and *y* between 0 and 1. Browse until you find the location of the repository "VVCS-Example" and select that directory name. This version of the code will be Revision 1.

	Computer SW_Preload (C:)) ► SVN-Codes ►		 ✓ ✓ 	ch 🔎
🎍 Organize 👻	📗 Views 👻 🖬 Open 🖃	E-mail 👩 Share		_	0
Name	Date modified	Туре	Size	Tags	
📙 Code1	1/26/2008 9:06 AM	File Folder			
Share Share Syn CH Groovy Restorn & Combi Scan fr Send T Cut Copy Create Delete Renam	is Notebook in OneNote eckout eSVN special	 Repo-brows Export Create repos Import Help Settings About 			
Proper					

Compute	er → SW_Preload (C;) → SVN-Codes →	▼ 🍫 Search	
🐚 Organize 👻 🏢 Views	; 🔻 📑 Open 🖃 E-mail <u> 8</u> Share		0
Name	Date modified Type 1/26/2008 9:06 AM File Folder Import Repository URL of repository:	Size Tags	
	Import message Recent messages	Browse For Folder SVN-Repository SVN-Reposit	E Cancel

Importing a file into the repository: selecting Import in the TortoiseSVN menu (top) and selecting the repository directory name (bottom)

2.3 Checking the Code out from the Repository

You now have the code "code1.f" safely placed in the repository. To modify this code and create a new revision, you will need to check out a working copy of the code. Go to the directory where you will be modifying the code, in this example, the directory "Modifications." Right click in Windows Explorer, and select "SVN Checkout…" Select the name of the repository you just created, then click "OK." You will now get a window telling you that you are at Revision 1. Notice the green check mark on the "code1.f" icon. This indicates that this working copy is up to date with the version in the repository.



a) SVN Checkout

SW_Prelo	ad (C:) 🕨 SVN-Codes	▶ Code1 ▶ Modi	fications	✓ ✓ Search	٩
🌗 Organize 👻 🏢 Views	•	_	_		0
Name		Date modified	Туре	Size	
		This folde	er is empty.		
	🖋 Checkout			X	
	Repository URL of repository:				
		tepository/VVCS-Exam	ple	-	
	Checkout directory:				
	C:\SVN-Codes\Cod				
	Only check out the Only Check ou	ne top folder			
	HEAD revision				
	Revision			Show log	
			ок с	ancel Help	
					1
1					
	Omit externals Revision HEAD revision		OK (

b) Selecting the proper repository

	_Preload (C:) SVN-Cod	es ▶ Code1 ▶ Modifi	cations 👻	✓y Search		X •
🖣 Organize 👻 📗	Views 🔻				_	0
Name	*	Date modified	Туре	Size		
🛃 code1.f		1/26/2008 10:00 AM	Absoft Fortran File	1 KB		
	🐭 file:///C:/SVN-Repo	sitory/VVCS-Example - T	ortoiseSVN Checkout.	Finished!	- • •	
	Action Path		Mime type			
	Added C:\SVN-C Completed At revisio	odes\Code1\Modifications\	code1.f			
	Completed At revisio	1: 1				
	Added:1				Cancel	
				ОК		

c) Successful checkout of Revision 1

Checking out a working copy from the repository

2.4 Modify the Code and Compare to the Repository Version

The code "code1.f" can now be modified. Here we will change the code to allow the Cartesian grid to contain 33x17 points between the values of zero and ten. Once the code has been modified, you will notice that the green check mark has been replaced by a red exclamation point, indicating that the current working copy has been modified from the version in the repository. To examine these differences, right click on the "code1.f" file, select "TortoiseSVN," then "Diff." This opens the "TortoiseMerge" tool which clearly shows the modifications to the repository version (Working Base) that were made in the Working Copy.



Comparing the Working Copy to the repository version (Working Base)



TortoiseMerge tool which clearly highlights differences between the modified version (Working Copy) and the repository version (Working Base)

2.5 Check the Code back into the Repository

When the changes are complete, the repository can be updated with your modified Working Copy by performing a checkin. Just right click on the file (or directory) and select "SVN Commit..." Enter a message describing the changes that were made, then select "OK." You are now at Revision 2.

	N-Codes → Code1 → Modifie		(a) Count	
Organize - III Views - 💀 Op		ations 👻 🖣	Search	
me	Date modified	Туре	Size	
code1.f	1/26/2008 10:14 AM	Absoft Fortran File	1 KB	
Open Print Scan for Viruses Open With If SVN Update If SVN Update If SVN Commit If TortoiseSVN If TortoiseSVN E-mail with Yahoo! Restore previous versions Send To Cut Copy Create Shortcut Delete Rename Properties				

Check the modified version (Working Copy) into the repository

3 TortoiseGit Tutorial

The following tutorial on the Git version control system was created using TortoiseGit version 1.8.1.0 on a computer running Microsoft Windows 7. Another TortoiseGit tutorial is available at the <u>TortoiseGit homepage</u> under <u>screenshots</u>.

3.1 Creating a Master Repository

Determine a location for the master repository, ideally on a server which is automatically backed up. Create a folder with the name of the repository; in this example the repository is called "Git-Example.git". Right click on the folder name, choose "Git Create repository here...".



Check the box labeled "Make it Bare (no working directories)", which means that you cannot do your work in this location, but instead it will serve as the master repository that you can copy locally to do your development work, then select "OK". You should see a message like "Initialized empty Git repository in Z:/Git-Repository1/Git-Example.git/", then click on "OK".



3.2 Cloning the Repository

To create a local repository where you can work, you need to clone the master repository. Navigate to the local location where you want to work, then right click and select "Git Clone...".

<u> </u>	C:) CIROY Classes CFD-Advanced	2013 • Git-Tutonal	• •7	Search Git-Tutorial	
Favorites Name Downloads Recent Places	* Date mod	dified Type his folder is empty.	Size		
E Desktop Dropbox	View Sort by Group by Refresh	*			
 Bocuments Git Music 	Customize this folder Paste Paste shortcut Undo Rename	Ctrl+Z			
S Pictures Subversion Videos	Git Init Here Git Gui Git Bash				
Scomputer	Share with Git Clone				
DATAPART1 (D:) roycfd (\\alexandr	If Git Create repository hereIf TortoiseGit				
🫫 cjroy (\\alexandria	🛃 SVN Checkout 😁 TortoiseSVN				
0 items	New Properties				

Select the "Dir..." button and navigate to the location of the master repository that you

created, then select "OK". You have now successfully cloned the master repository.

🕈 Git clone - TortoiseGit	🕼 C:\CJROY\Classes\CFD-Advanced\2013\Git-Tutorial - Git Command P 📼 📼
Clone Existing Repository URL: Z:\Git-Repository2\Git-Example URL: Z:\Git-Repository2\Git-Example Directory: C:\CIROY\Classes\CFD-Advanced\2013\Git-Tutorial\Git-Exz Browse Gepth 0 Recursive Clone into Bare Repo	3 * 📁
	dome. varning: You appear to have cloned an empty repository. Success (249 ms 8 2/13/2013 3:58:24 FM)
Trunk Tags: branches Prom: 0 Username: OK Cancel Help	Cose Abort

3.3 Adding a File to the Local Repository

Copy any files you wish to add to the local repository into this "Git-Example" directory, then right-click and choose "TortoiseGit" then "Add…".



Place a check mark by the file(s) you want to add (here we are adding the single file LWE.f95) then select "OK". You can then select "Commit …" to commit this file to the local repository (note: it will not yet be added to the master repository).



You should now enter a message associated with this initial import of the file into your local repository then select "OK".

C:\CJROY\Classes\CFD-Advanced\2	013\Git-Tu	orial\	Git-E	xample - Commit 💶 🗉 🗾 🗾									
Commit to: master	new branc	า											
Initial import of Linear Way	ve Equatio	on co	le i	nto the local repository.									
Amend Last Commit				1/71									
Set commit date				Add Signed-off-by									
Check: All None Unversioned V Path													
				B									
Show Unversioned Files 1 files selected, 1 files total													
Do not autoselect submodules				View Patch>>									
Whole Project			(OK Cancel Help									
Message only													

At this point, the file should show up with a green check mark indicating that this version is up to date relative to your local repository.

Organize • Include in	library Share with Burn New feeture	older			- 198	
🚖 Favorites 🛛 🔺	Name	Date modified	Type	Size		
🐌 Downloads	📙 .git	2/13/2013 4:10 PM	File folder			
Skecent Places	LWE.f95	1/29/2009 9:14 AM	F95 File	15 KB		
Cesktop						
🗼 Dropbox						
🚆 Libraries						
Documents						
- Git						
Music						
S Pictures						
😹 Subversion						
JUI Videos						
Scomputer						
lows7 (C:)						
International (D:)						
🛫 roycfd (\\alexandı						
🛫 cjroy (\\alexandria						
💐 Network 🛛 👻						

3.4 Modify the Code in the Local Repository

You now have the code "LWE.f95" safely placed in the local repository. You can now modify the code by changing some of the input parameters. Notice that the green check mark has now changed to a red exclamation mark, indicating that the file is no longer at the same revision level as the file in the repository.

Organize 🔹 Include in	library • Share with • Burr	n New folder			H • 🗍	1 (
🚖 Favorites 🔶	Name	Date modified	Туре	Size		
🐌 Downloads	📜 .git	2/13/2013 4:10 PM	File folder			
🐉 Recent Places	UWE.f95	2/13/2013 4:20 PM	F95 File	15 KB		
Cesktop						
L Dropbox						
States in a						
Libraries Documents						
🥧 Git						
Music						
Pictures						
Subversion						
L Videos						
- The second second						
Somputer						
😂 Windows7 (C:)						
INTAPART1 (D:)						
🛫 roycfd (\\alexandı						
🛫 cjroy (\\alexandria						
💽 Network 🛛 🚽						

To check these changes in to the local repository, there are (at least) two options. First, one could right-click on the file name and select "Git Commit -> "master"...".



At this point, it is often useful to review the changes that were made to the file by rightclicking the file name and selecting "Compare with base".



This bring up a valuable tool called "TortoiseGitMerge", which highlights the changes that were made in the file by comparing the differences graphically (and with color) sideby-side: the original code on the left and the modified code (the "working tree") on the right. The left side bar show a graphical summary of where all code changes are located (in this case, they are all near the top of the file). In addition, placing the cursor over any line with bring up the old and new lines at the bottom of the tool so they can be compared in more detail. Once you have reviewed the changes, enter a message in the "Commit" dialog box and select "OK". If your commit was successful, then select "Close".

Open Save as mis The set in the	Open Note: Sum as well below Sum as we	* 8						LW	VE.f95: W	/orking T	ree - To	rtoiseGitMerg	e				
Save s Reload Undo Copy Paste Us Book Goto Line Dhaft is stocked Free Stock is the code of the stocked Free Stock is the code of the stocked Integer, Parameter ::: Code -:: Selected Real, Kind (p=17, r=2000) Integer, Parameter ::: Code -:: Selected Real, Kind (p=17, r=2000) If Integer, Parameter ::: Code -:: Selected Real, Kind (p=17, r=2000) 19	Save See a Relad Und Copy Past Use Mutata resolution of the path find filterace difference office resolution of the set of the second of th		ain														Style 🔹 🕡
Same as Description Data and R deference of difference conflict difference of dif	Backs and the fibre of the score path fibre difference di			20 X (L		16	4 4	合日	Ca	11/2				and the second		
IWE . f95: 406b6c5 16 Integer, Parameter :: Extended = Selected Real Kind(p=17, r=200)] 17 Integer, Parameter :: Extended =: Selected Real Kind(p=27, r=200)] 18 Integer, Parameter :: Prec := Double	INE. 495:406b6c5 16 Integer, Parameter::: Extended:Selected Real_Kind(p=17,r=200)] 17 Integer, Parameter::: Cuad:Selected Real_Kind(p=26,r=200)] 18 Integer, Parameter::: Prec:Singled Real_Kind(p=26,r=200)] 19	iave as	Reload Undo	Copy Paste		Mark as resolved									Inline diff word-wise	Ignore all whitespace changes	5 1
<pre>16 Integer, Parameter :: Extended Selected Real Kind(p=17, r=2000) 17 Integer, Parameter :: Pudd · Selected Real Kind(p=26, r=200) 17 Integer, Parameter :: Pudd · Selected Real Kind(p=26, r=200) 19 Integer, Parameter :: Prec -= Double ·· Prec -= Single gives ·double · 19 ·· · · · Prec -= Single gives ·double · 20 ·· · · · Prec -= Double ·· · · · Prec -= Double ·· · · · · · · · · · · · · · · · · ·</pre>	16 Integer, 'Parameter :: Extended Selected Real_Kind(p=17, r=2000) 17 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 18 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 19 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 10 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 11 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 12 End Module 21 End Module 22 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 14 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 15 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 22 End Module 23 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 24 Module-Set_Inputs] 25 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 26 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 27 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 28 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 29 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 20 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 21 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 21 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 22 Integer, 'Parameter :: Ouda Selected Real_Kind(p=26, r=200) 23 Integer, 'Parameter :: Ouda Se	iles			Edit					N	lavigate				Vie	W	r5
<pre>17 Integer, Parameter :: QuadSelected Real_Kind(p=26,r=200) ' 18 Integer, Parameter :: Prec = Double</pre>	17 Integer, Parameter::::Quadselected_Real_KInd(p=26,r=200)] 18 Integer, Parameter::::PrecSouble:Precision-for-the computation 19 Integer, Parameter:::PrecSouble:Precision-for-the computation 19 Integer, Parameter:::PrecSouble:PrecUble-gives double-p 20 Prec				LW	E.f95:406b6c5					^				LWE.f95: Workin	g Tree	
	45 46 End Module] 47 48 49 Module-Geometry 50 € m → € m → € m	19 211 222 231 24 25 260 27 28 29 301 31 322 331 322 331 332 331 335 331 335 331 335 335 335 335 335	End Module ************************************	rameter - :: rec) -:: rec) -: rec) -:	···ima: ··for·tl ··ite ···cfl ··tol ··cs ··rtin ··rtin ··rtin ···	<pre>x = 05 x = 05 x = 50000 rout = 2 </pre>	· Prec · = · Prec · = ave · Equ. rec · = Prec · =	<pre>Single - Double - ation - co </pre>	ode ode ax # of cof po ax # of cof ti mber ance fo r wave parame hitial m time 3.14150	oints i f time i speed time te	n st st (m	19 21 End -1 22 J : **** 24 Modu: 25 J : **** 26 Use -: 27 J 28 ! - Set 29 J 30 Impl: 31 J 32 Save 33 J 32 Save 34 Intes * 35 Inte * 36 Inte * 36 Inte * 36 Real 38 Real 39 Real 40 Real 41 Real 42 Real 43 Real	Moduled K****** le Set_Tr Select_Pr t the cod icit - None ger :: ger :: (kind=Prc (kind=Prc (kind=Prc (kind=Prc (kind=Prc	ameter :: c) ::	<pre>for the 1D Line. imax = 33 max = 50000 </pre>	<pre></pre>	<pre>points in time step rd .for conv ve sped. anterd al.time time. iso</pre>
		•									>	4		.111			,
Integer, Parameter :: imax = 33				ontally with Cti													



Alternatively, one could right-click on the file and select the "Git Commit Tool", which brings up the following window indicating that three lines were changed from the version in red to the version in green.

Demositeres Edite Committe To	ala Uala			_
Repository Edit Commit To Current Branch: master	ois Heip			
			an Ann	
Unstaged Changes	Modified, not staged	File: LW	E. 1 95	
LWE.f95	<pre>@@ -29,13 +29,13 @@ Use Implicit None Save -Integer, Parameter :: -Integer :: +Integer, Parameter :: +Integer :: Real(kind=Prec) :: Real(kind</pre>	<pre>imax = 65 nmax = 500000 iterout = 2 imax = 33 nmax = 50000 iterout = 20 cfl = 0.5_Prec toler = 5.0e-14_Prec</pre>	! number of time steps between o ! CFL number ! tolerance for convergence	ut .n
	Stage Changed Sign Off Commit			
+	Push			

CAUTION: At this point, the file import and changes have not yet been "pushed' into the master repository, they have only been added to the local repository.

3.5 Pushing Changes to the Master Repository

To push the changes from the local repository to the master repository, right-click in the window area (NOT on the LWE.f95 file), select "TortoiseGit", then "Push".

🕽 🔵 🗣 📙 « cjroy 🕨 clas	ses + cfd-advanced + 20	013 • git-tutorial • updated • git-examp	ole 🕨 👻 🔩	Search Git-Example	
Organize • Include in I	ibrary • Share with •	Burn New folder		II • 🔲	0
 Favorites Downloads Recent Places Desktop Dropbox Libraries Git Music Pictures Subversion Videos Computer 	ibrary • Share with • Name .git .git .git	Date modified 2/14/2013 12:54 View Sort by Group by Refresh Customize this folder Paste Paste shortcut Undo Copy Cit Add all files now Git History Git Branch Git Bash	Type Size File folder File F95 File 15 KB Image: Post of the state sta		
 Windows7 (C:) DATAPART1 (D:) roycfd (\\alexandria.a cjroy (\\alexandria.ao 		Share with Git Sync Git Commit -> "master" TortoiseGit Syn Checkout	 A Stash Save Bisect start A Resolve Revert Clean up 		
🕵 Network		TortoiseSVN New Properties	 Switch/Checkout Merge Create Branch Create Tag Export Add Submodule Add 		
2 items			 Create Patch Serial Apply Patch Serial Settings 		

This will pull up the Git Push dialog box, then click on "OK".

tef			
Push all brand	hes		
Local:	master		
Remote:	master		•
Destination			
Remote:	origin		▼ Manage
C Arbitrary URL:			*
Options			
		anch (May discard changes)	
Use Thin Pack	(For slow ne	twork connections)	
Autoload Putty	Kov		
Set upstream/	0 - 2 - 4	branch	
Always push to	o the selected	I remote archive for this local branch	
Recurse submodu		I remote branch for this local branch None	

If you get the message that the push was successful, then the changes to your local repository have been successfully pushed to the master repository.

C:\CJROY\Classes\CFD-Advanced\2013\Git-Example - Git Comm	nand Progress - Tor	- 🗆 🗙
*		
Writing objects		
git.exe pushprogress "origin" master:master		
Counting objects: 6, done. Compressing objects: 100% (4/4)		
Writing objects: 100% (6/6)		
Writing objects: 100% (6/6), 3.70 KiB, done. Total 6 (delta 1), reused 0 (delta 0)		
To Z:\Git-Repository1\Git-Example.git * [new branch] master -> master		
Success (468 ms @ 2/14/2013 12:58:55 PM)		
Create pull request	Close	Abort

If you are developing code with other developers, before you "push" your changes to the master repository, you should first "pull" any changes that the other developers have pushed to the master repository since you last synchronized with it. To pull any changes in the master repository to your local repository, right-click in the window area (NOT on the LWE.f95 file), select "TortoiseGit", then "Pull".

3.6 Other Useful Tools

After you have made additional changes to the code, you may want to examine the history of the changes to a file/project by right-clicking on the file and selecting "Git History". This will pull up the "gitk" tool that is also available in Unix/Linux.

🌍 🖌 « cjroy 🕨 clas	ses 🕨 cfd-adv	ance	d ▶ 2013 ▶ git-tutorial ▶	updated 🕨 git-exampl	e 🕨	• 49 S	Search Git-Example	
Organize 🔹 🔲 Open	▪ Burn	Ne	ew folder				III • 🔲	(
Favorites	Name		* 	Date modified	Туре	Size		
Recent Places	📕 .git			2/14/2013 1:03 PM	File folder			
Desktop	🥖 LWE.f95		Open	1:03 PM	F95 File	15 KB		
), Dropbox			Git Add all files now Git Commit Tool					
🟐 Libraries			Git History Git Blame	1				
Documents			Git Gui					
J. Git			Git Branch	۰.				
🕹 Music			Git Bash	12 March 11				
S Pictures			Secure within Identity Finder Shred with Identity Finder	Vault				
😹 Subversion 🎩 Videos			Search with Identity Finder					
Judeos			Scan for Viruses					
Computer			Open with	٠				
Windows7 (C:)		7	Git Commit -> "master"					
INTAPART1 (D:)		R	TortoiseGit	•				
🛫 roycfd (\\alexandria.a		16	TortoiseSVN	•				
🛫 cjroy (\\alexandria.ao			Restore previous versions					
			Send to	P.				
Network			Cut					
			Сору					
			Create shortcut Delete					
	modified: 2/14		Rename	2:50 PM				

III gitk: Git-Example	_ D <mark>_ X</mark>
File Edit View Help	
Imaster Imaster Modified additional inputs in LWE.195. Chris Roy <c roy@vt.edu=""> 2013-02-14 13:03:48 Modified some of the input values in LWE.95. Chris Roy <c roy@vt.edu=""> 2013-02-14 13:03:48 2013-02-14 13:03:48 Initial import of Linear Wave Equation code into the local repository. Chris Roy <c roy@vt.edu=""> 2013-02-14 12:51:35</c></c></c>	
SHA1 ID: c3a135cf01e831058bc41e40c3b9e4ada46b435b ← → Row 1/ 3	
Find next prev commit containing:	Exact 🔹 All fields 💌
Search Patch © Tree	
<pre>Diff Old version 0 New version Lines of context 3</pre>	
4 III	Ŧ

You may also want to see the log file for a given file by right-clicking on the file, selecting "TortoiseGit", then "Show log". This brings up a GUI which shows useful information about the file in question.



naster		From	2/14/201	3 💌	To:	2/14/2	2013	•			s, Emi	Aut	IUI LI	nall	•		
Graph	Actions	Message		Author	Date	1											
		Working dir o	nanges														
•	6 6	master origi			2/14	·											
I	• •	Modified som Initial import															
Modi.	fied add	cf01e831058 ditional in atus L L			46b43	5b											
* Modi Path	E St	ditional in atus L L od 2 2	puts in L	WE.195.													
Modi Path	E St	ditional in atus L L	puts in L	WE. <u>f</u> 95.			evision	(s) s	elected	1							
Modi Path	E St Mo 3 revision	ditional in atus L L od 2 2	puts in L	WE. <u>f</u> 95.	n c3a13					1					S	tatisti	8
Modi Path LW Showing	E St Mo 3 revision	ditional in atus L L od 2 2 u(s), from revis	puts in L	WE. <u>f</u> 95.	a c3a13	5c - 1 r	hole Pr	ojec		1						tatisti	0.92.

References

- Steve Eddins (2006), "Taking Control of Your Code: Essential Software Development Tools for Engineers," International Conference on Image Processing, Atlanta, GA, 2006 (see: <u>http://blogs.mathworks.com/images/steve/92/handout_final_icip2006.pdf</u>, last accessed February 3, 2019).
- Chacon, S., and Straub, B. (2019), Pro Git, 2nd Ed., Apress, Version 2.1.108, 2019-01-31 (see: <u>https://git-scm.com/book/en/v2</u>, last accessed February 3, 2019).
- Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato, Version Control with Subversion: For Subversion 1.7 (Compiled from r5868), (see <u>http://svnbook.red-bean.com/en/1.7/svn-book.pdf</u>, last accessed February 3, 2019).