

# TortoiseGIT / GIT Tutorial:

## Hosting a dedicated server with auto commit periodically on Windows 7 and Windows 8

---

### **Abstract**

This is a tutorial on how to host a dedicated gaming server on Windows 7 and Windows 8 using TortoiseGIT and/or GIT, whichever you prefer.

The intended audience is for those who runs a dedicated server that has an ever changing world map, for example, a Minecraft server, and for those who runs a dedicated server with constant changes to some files, such as leaderboards, player save data, etc.

---

---

## Table of Contents

1. Setup and Installation
  2. Instructions
  3. Credits
- 

## Setup and Installation

The first thing you will need to do is to install the following softwares if you do not have them at this point of time.

- TortoiseGIT or GIT (Required)
- msysgit (Required)

I recommend with starting off with TortoiseGIT, as it has an easier learning curve than GIT. For advanced users, feel free to use GIT, as they both have common functionalities.

First, open up your preferred web browser and navigate to the TortoiseGIT project site (Figure 1). The URL is given below:

<http://code.google.com/p/tortoisegit/>

Click on the Downloads tab near the top of the project home page, and we should see two versions of TortoiseGIT and a list of language packs (Figure 2). If you are using an x86 operating system, please use the 32-bit version; otherwise, if you are using an x64 operating system, please use the 64-bit version. Both of these versions are not entirely the same. For more information, you may look it up on Wikipedia. In my case, I am using Windows 8 64-bit, so there will be minor differences from your operating system with mine in this tutorial.

If your native language is not English, you may select the correct version with your preferred language.

Once the setup files have finished downloading, use the File Explorer to navigate to the setup files and proceed to install them.

When TortoiseGIT has completed its installation, you may be prompted to install msysgit, which is the core part of GIT and provides GIT functions for TortoiseGIT and GIT on the Windows operating system. The URL for msysgit is given below, which will navigate you to its project site (Figure 3):

<http://code.google.com/p/msysgit/>

(Figure 1): TortoiseGIT project home page.

Country	Code	Completeness	32 BIT	64 BIT
Bulgarian, Bulgaria	bg_BG	65%	<a href="#">Setup</a>	<a href="#">Setup</a>
Chinese, simplified	zh_CN	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
Chinese, traditional	zh_TW	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
Czech	cs	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
Dutch, Netherlands	nl_NL	60%	<a href="#">Setup</a>	<a href="#">Setup</a>
French	fr	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
German	de	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
Italian, Italy	it_IT	69%	<a href="#">Setup</a>	<a href="#">Setup</a>
Japanese	ja	100%	<a href="#">Setup</a>	<a href="#">Setup</a>
Korean, Korea	ko_KR	61%	<a href="#">Setup</a>	<a href="#">Setup</a>
Dutch, Dutch	nl_NL	60%	<a href="#">Setup</a>	<a href="#">Setup</a>

(Figure 2): The Downloads tab for TortoiseGIT.

(Figure 3): msysgit project home page.

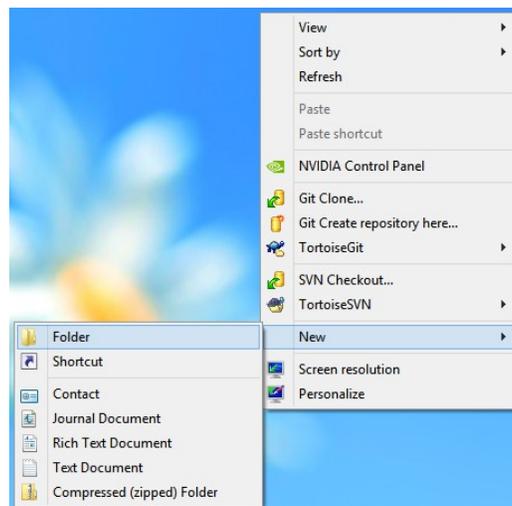
Once all TortoiseGIT and msygit installations have been completed, the next thing to do is to determine where to create a repository folder for your dedicated server data. For more information about using GIT, you may visit the following URL:

<http://www.vogella.com/articles/Git/article.html>

---

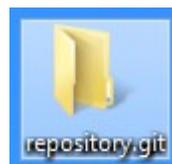
## Instructions

I will use the Desktop as my location for creating my repository for the purposes of this tutorial. To start, right click anywhere on the Desktop, in the context menu, hover over “New”, and click on “Folder” (Figure 4). Note that I have TortoiseSVN installed. Please ignore that for the time being.



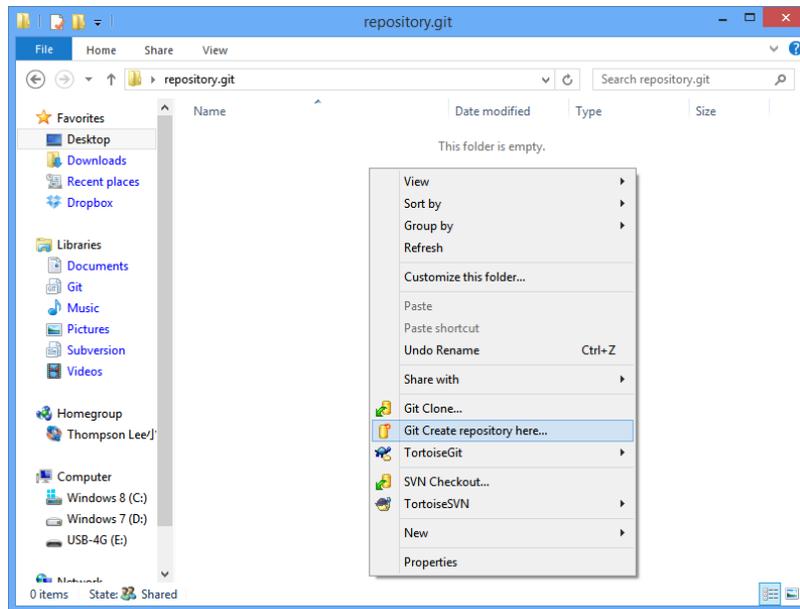
(Figure 4): The context menu.

After a new folder has been created, rename the folder so that there is “.git” suffix at the end of the name. This is the GIT convention, and it’s recommended to use this naming scheme. I have named mine, “repository.git”, for example (Figure 5). This will be your repository folder.

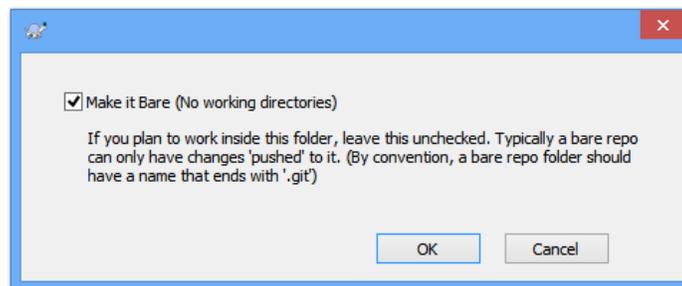


(Figure 5): repository.git folder.

Open the “repository.git” folder, and right click anywhere in the empty space to open up the context menu. Click on “GIT Create repository here...”, and a prompt will display (Figure 6). Enable “Make it bare (No working directories)”, and finally click on OK (Figure 7). The “repository.git” folder should then be populated with files and folders. You may ignore the files and folders.



(Figure 6): The context menu (again).

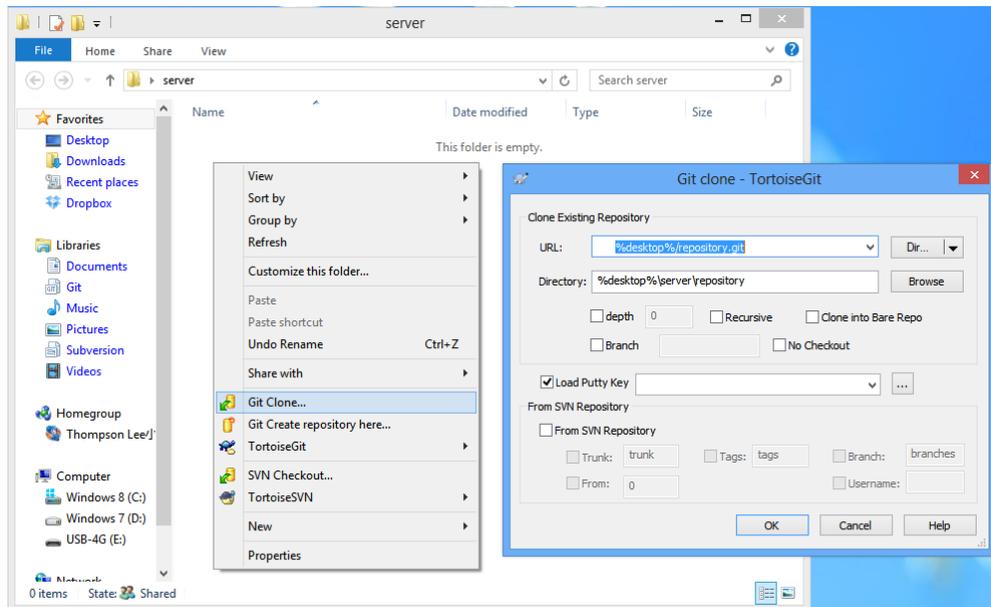


(Figure 7): The dialog prompt.

Going back to Desktop, create another new folder and name this folder to anything you want. I named mine to “server”. This folder shall then become the directory that you are to put your server data files in and commit to your repository folder. This will be your commit folder.

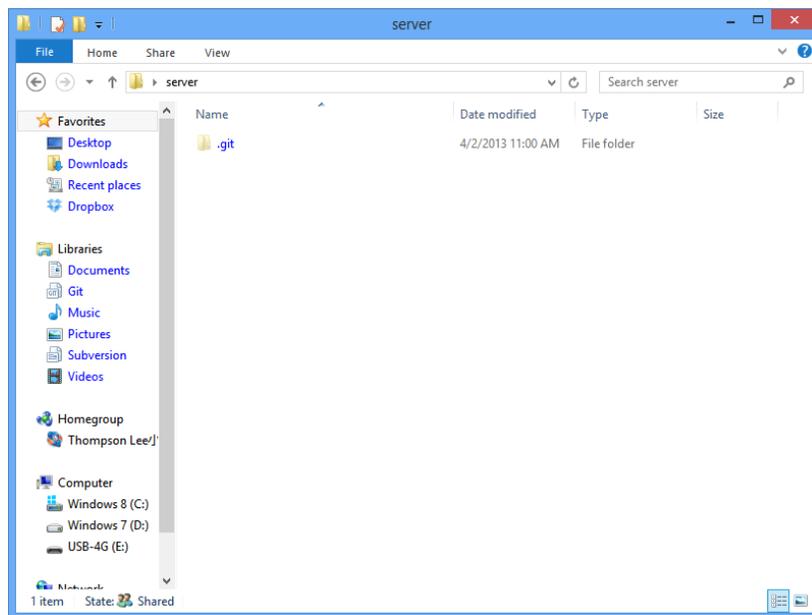
Open up the newly created folder you had just made, and right click anywhere again to bring up the context menu. Click on “GIT Clone...” to bring up the Clone dialog (Figure 8). For the URL, click on the “Dir...” button and find the folder, either the “repository.git” or the folder ending with the “.git” suffix that you had just created, and press OK. For the Directory, make sure that it is pointing towards the directory folder that you had just right clicked on “GIT Clone”, by browsing to the location. You may ignore the rest of the options given, and press OK once you’re done.

If you happened to spot a new folder directory inside your commit folder, and that new folder’s name is the same as your repository folder, but without the “.git” suffix, then what you did was you created a new commit folder nested in your “originally planned commit folder”. Your “originally planned commit folder” is no longer your commit folder, but rather it is just a file directory.



(Figure 8): The context menu and the GIT clone dialog.

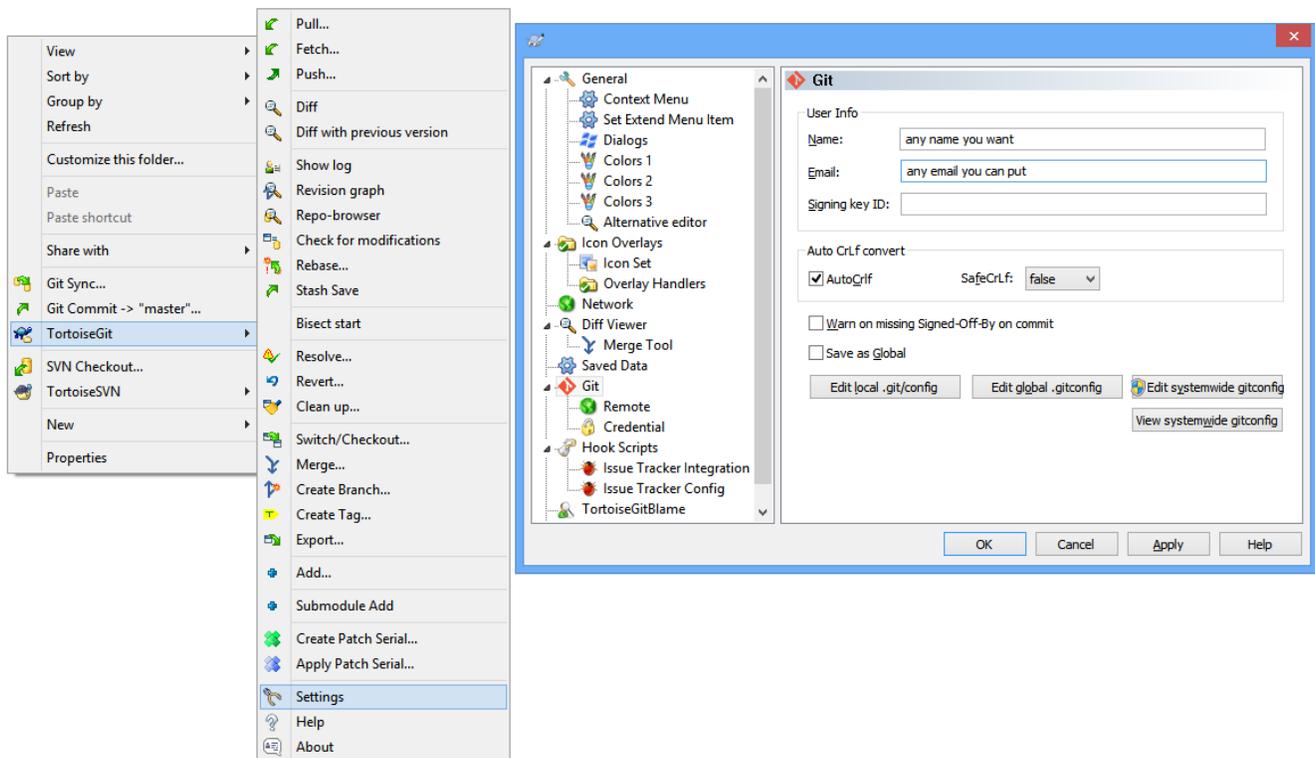
If you have “Show hidden folders” option enabled, when you browse your commit folder, you should see a “.git” folder (Figure 9). This “.git” folder serves as a bookmarker for your local changes. You may ignore the “.git” folder, and continue with the tutorial.



(Figure 9): The hidden “.git” folder.

The next important step is to set your GIT account. This GIT account is a local account used when you are committing local changes and pushing changes to your local repository folder.

Right click anywhere in your commit folder, hover over TortoiseGIT, and select “Settings” near the bottom of the context menu. In the left navigation page, click on “Git”. In the right navigation pane, put in any name and any email account (Figure 10). Since we are doing a local repository, the name and



(Figure 10): The context menu and the Settings dialog, with “Git” selected.

email account is not important for our purposes, therefore, you may ignore their importances for now.

From here, you can start placing your server data files inside your commit folder. For the purposes of this tutorial, I will be using a Minecraft server to serve as a placeholder for your dedicated server files.

After you have placed your server data files inside your commit folder, you should see a navy blue question mark icon overlay on all of your data files and folders (Figure 11). This icon overlay tells you that GIT has detected new files in your commit folder. We shall come back to this in a moment.

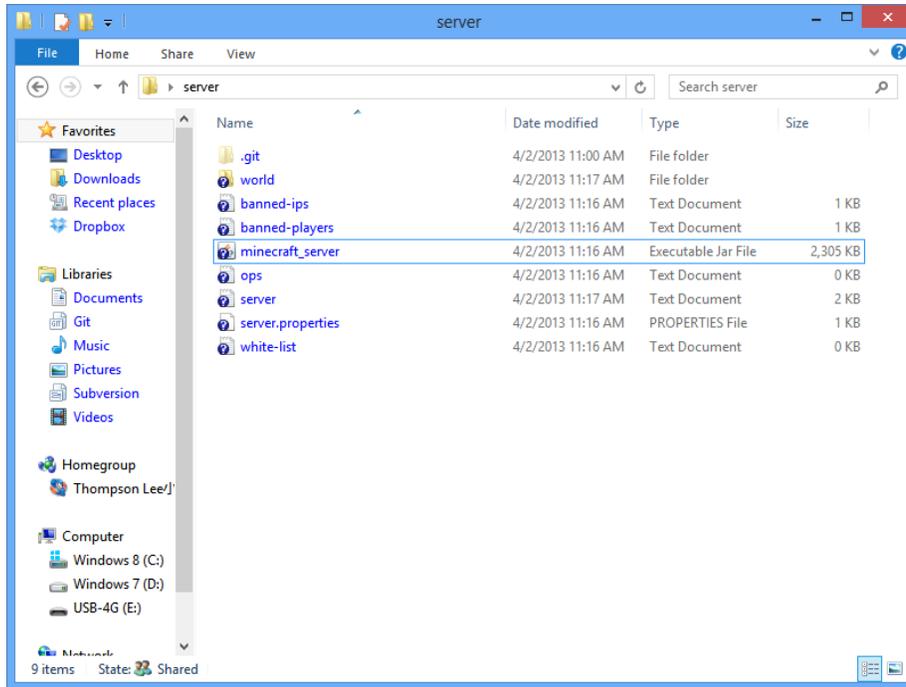
This next step involves a little bit of batch programming. For those who are not familiar with batch programming, you may refer to the following URL for more information:

<http://www.robvanderwoude.com/batchfiles.php>

I can assure you, it is not programming. Rather, it is just to change the GIT installation directory if you did not install GIT at the default installation location.

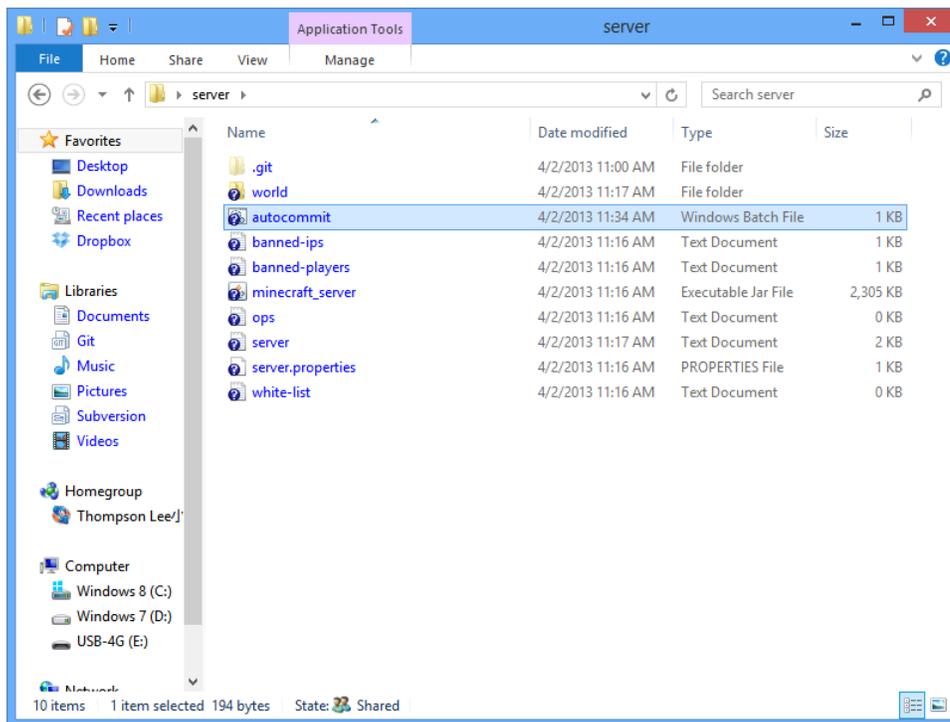
Now, run Notepad from your Start menu. In my case, I used Notepad++ for its syntax highlighting features. You may choose to install Notepad++ from the following URL:

<http://notepad-plus-plus.org/>

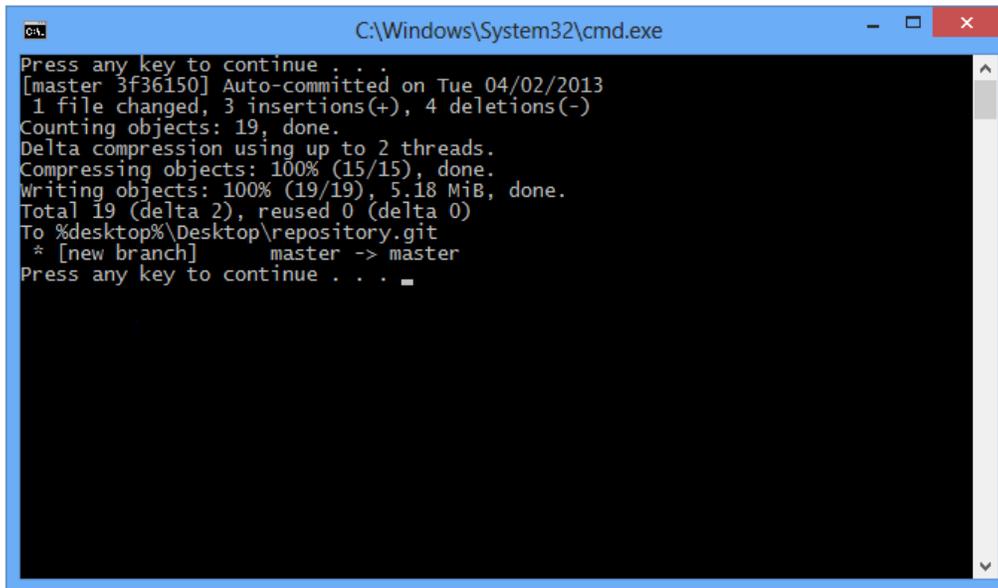


(Figure 11): GIT detects new files.

Run the BATCH file, and it *should* automatically change the icon overlays to a green checkmark, telling you that all of your files and folders have been committed successfully to your repository folder (Figure 12).



(Figure 12a): The “autocommit.bat” file. Before execution.



```
C:\Windows\System32\cmd.exe
Press any key to continue . . .
[master 3f36150] Auto-committed on Tue 04/02/2013
1 file changed, 3 insertions(+), 4 deletions(-)
Counting objects: 19, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (15/15), done.
Writing objects: 100% (19/19), 5.18 MiB, done.
Total 19 (delta 2), reused 0 (delta 0)
To %desktop%\Desktop\repository.git
* [new branch] master -> master
Press any key to continue . . . =
```

(Figure 12b): The output of the GIT

When I say “*should*”, I meant that due to TortoiseGIT having the issue of not refreshing the icon overlays not completely fixed, you couldn’t really tell if the files and folders have been committed and/or pushed to the repository. For more information, you may consult the following URL:

<http://stackoverflow.com/questions/8137929/git-modified-sign-never-leaves>

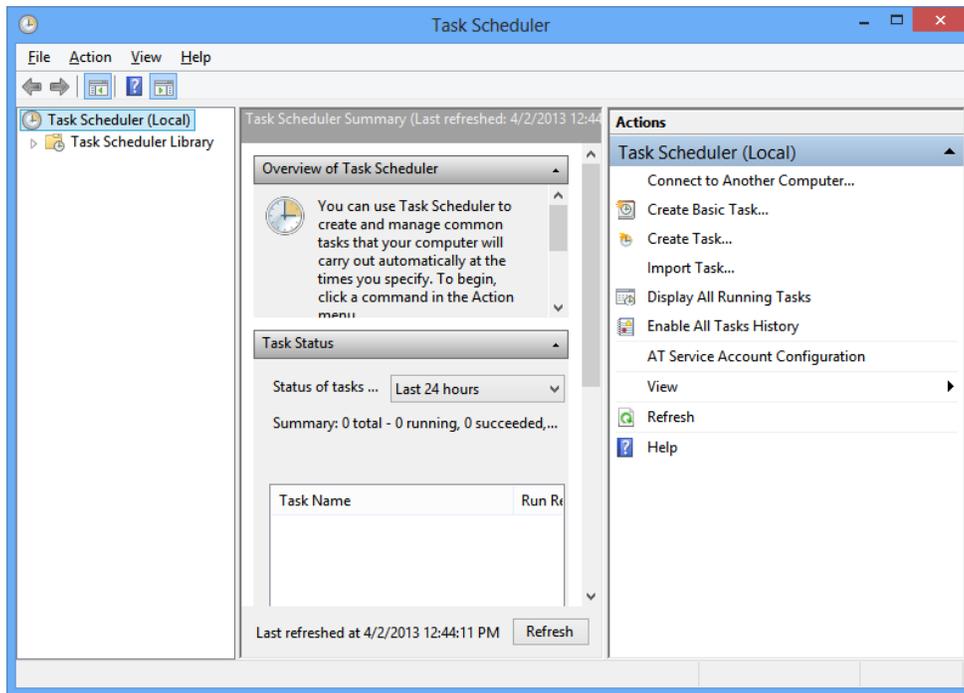
This ends the batch programming session.

Now, on towards automation.

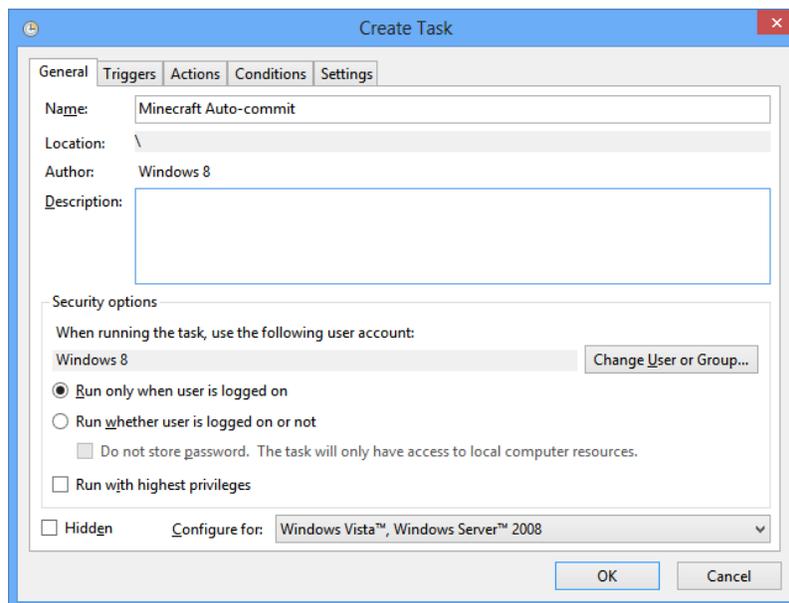
In Windows 7, navigate to the Control Panel via Start Menu → Control Panel. In Windows 8, press the Start button, type “Control”, and then press Enter. In the Control Panel, go to System and Security → Administrative Tools → Task Scheduler (Figure 13).

On the right pane, find and click on the option, “Create Task...”. It is not to be confused with “Create Basic Task...”, as it takes an additional step to setting it up. It should bring up the Task Wizard (Figure 14).

In the General tab, fill in the name and description of your task. In the Triggers tab, click on “New” to bring up a schedule wizard that aids you in setting up automation. Once this is set, in the Actions tab, press “New” to bring up a dialog asking you for the program/script you want to run. In this case, we want to run the BATCH file we have just copied/pasted/created in the commit folder. Browse to the BATCH file, select the file, and press OK. You may continue to set additional options in the Conditions and Settings tabs, if you liked. Once all of the options have been set, go ahead and click on OK to set the task in the Task Scheduler Library folder.



(Figure 13): Task Scheduler.



(Figure 14): Task Wizard.

Once all of this is done, your task should be in the Task Scheduler Library folder, by click on the “Task Scheduler Library” folder on the left pane in Task Scheduler. From this point on, you will have to test and see if the BATCH file is working for GIT, if all modified files have been committed and pushed to the repository, check to see if your Task is running and working properly, and **check to see if it causes substantial lag when it’s committing and pushing while your dedicated server is running.**

Other than all of those things, you have successfully created an automated backup system. Refer to the TortoiseGIT tutorials to learn how to revert your modified files to the last pushed version.

---

## **Credits**

Author: Thompson Lee  
Illustrator: Thompson Lee

All images are screenshots from Windows 8 64-bit.  
All URL links provided are for public consumption.

April 2, 2013

Non-commercial Creative-Commons license.

There is no April Fool's joke in this tutorial. Feel free to add one.

---