GIT: From Beginner to Fearless

GIT Ignore Activity: Excluding files from our repository

Brian Gorman, Author/Instructor/Trainer

Introduction

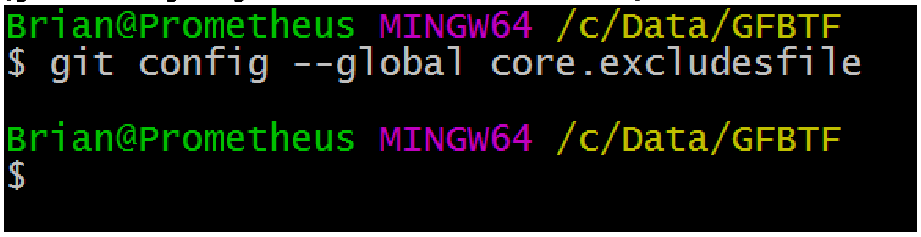
In almost every repository, there will be many files we are wanting to track changes on. Additionally, there will be a few files that we never want to track for various reasons. Perhaps the files are auto-generated by a build process. Perhaps the files are unique to a user. Perhaps the files contain sensitive information. In any event, we are able to exclude files from our repository very quickly and easily using a .gitIgnore file.

In this activity, we are going to take a look at how we can add a .gitIgnore file, and then we'll look into how we can modify that file to make sure that specific files and folders are eliminated from tracking.

Let's gets started!

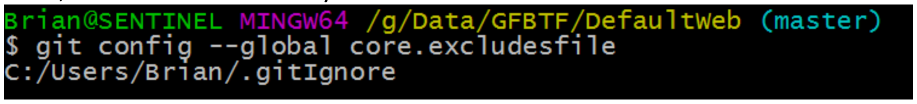
Step 1: Creating a global .gitIgnore file

a) Check to see if there is an excludes file directive in your global config.
First we need to tell our system where to find our global gitIgnore [excludes] file if one doesn't already exist.
[**git config --global core.excludesfile**]



```
Brian@Prometheus MINGW64 /c/Data/GFBTF
$ git config --global core.excludesfile

Brian@Prometheus MINGW64 /c/Data/GFBTF
$
```

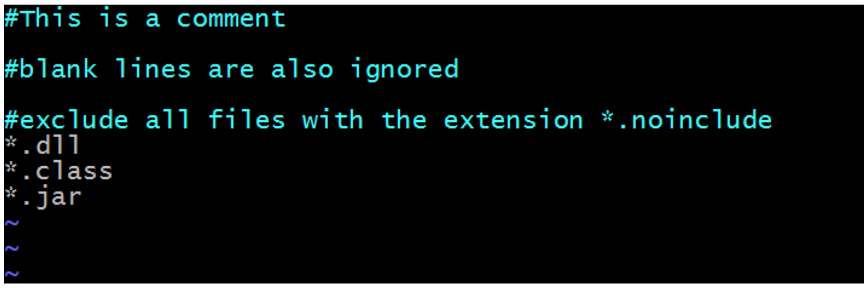
If nothing is set – nothing is shown. If you see a file listed, then you already have a global core excludes file, and you should be able to edit that file.
Here, I have a file set already:



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global core.excludesfile
C:/Users/Brian/.gitIgnore
```

If no file is present, create it:
[**git config --global core.excludesfile ~/.gitIgnore**]

b) Open the .gitIgnore File and add exclusions
Make sure you have the path to your gitIgnore file handy [as created or listed in a above]
[**vim <path-to-your-global-ignore>**]
[**i**]
[***.dll**]
[***.class**]
[***.jar**]
[**{esc}**]
[**:wq**]



```
#This is a comment

#blank lines are also ignored

#exclude all files with the extension *.noinclude
*.dll
*.class
*.jar
~
~
~
```

Notes

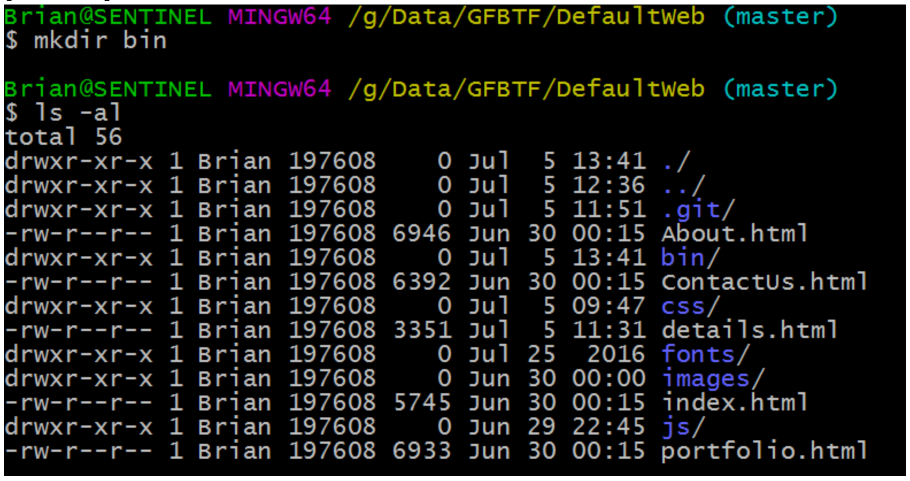
Step 2: Check that the .gitIgnore file is working:

a) Get the resources; extract them; copy them; or just create new ones

Under the root of the project, create a new 'bin' folder

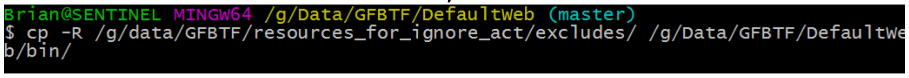
[**`mkdir bin`**]

[**ls `-al`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ mkdir bin

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ ls -al
total 56
drwxr-xr-x 1 Brian 197608    0 Jul  5 13:41 ./
drwxr-xr-x 1 Brian 197608    0 Jul  5 12:36 ../
drwxr-xr-x 1 Brian 197608    0 Jul  5 11:51 .git/
-rw-r--r-- 1 Brian 197608 6946 Jun 30 00:15 About.html
drwxr-xr-x 1 Brian 197608    0 Jul  5 13:41 bin/
-rw-r--r-- 1 Brian 197608 6392 Jun 30 00:15 ContactUs.html
drwxr-xr-x 1 Brian 197608    0 Jul  5 09:47 css/
-rw-r--r-- 1 Brian 197608 3351 Jul  5 11:31 details.html
drwxr-xr-x 1 Brian 197608    0 Jul 25  2016 fonts/
drwxr-xr-x 1 Brian 197608    0 Jun 30 00:00 images/
-rw-r--r-- 1 Brian 197608 5745 Jun 30 00:15 index.html
drwxr-xr-x 1 Brian 197608    0 Jun 29 22:45 js/
-rw-r--r-- 1 Brian 197608 6933 Jun 30 00:15 portfolio.html
```

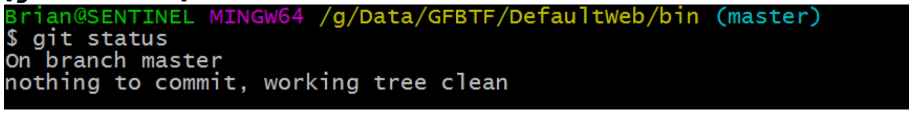
Place the two files for exclusion into the bin folder. Note that you will need to move the files from the location they are in to the location we created. You don't have to do this via BASH if you don't want to.



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ cp -R /g/data/GFBTF/resources_for_ignore_act/excludes/ /g/Data/GFBTF/DefaultWe
b/bin/
```

Make sure the repo does not recognize that anything has changed

[**`git status`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Note that even though we've added files, they are excluded by the .gitIgnore

b) Prove the folder is monitored

To prove the folder is being monitored, let's add something that would be found:

[If necessary, cd to the directory with the excluded files]

[**`touch info.txt`**]

[**`vim info.txt`**]

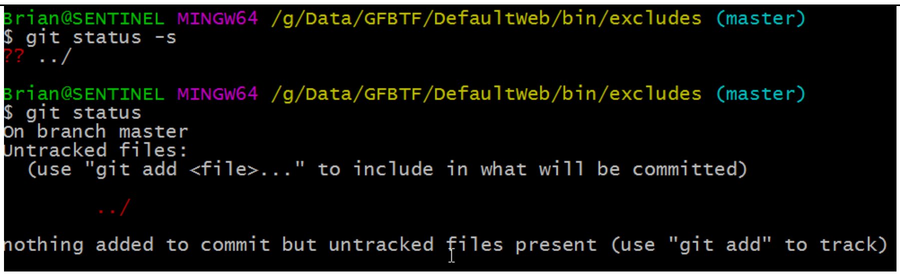
[**`i`**]

[**Working with .gitIgnore, this file should be included**]

[**`{esc}`**]

[**`:wq`**]

[**`git status -s`**]



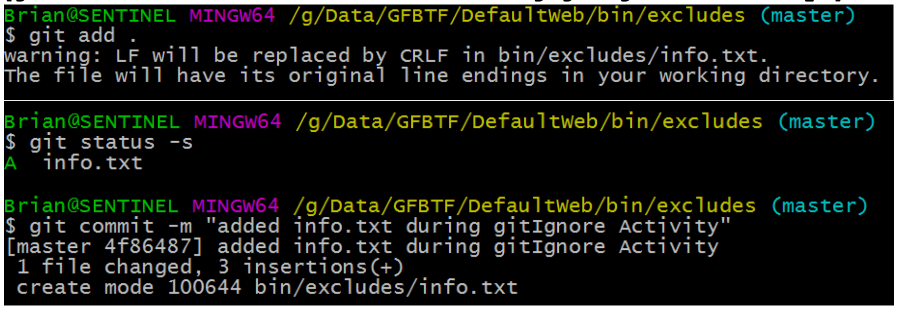
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status -s
?? ../

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ../

nothing added to commit but untracked files present (use "git add" to track)
```

c) Add and commit the info.txt file – further proving the others are ignored.

[**git add .**]

[**git status -s**]

[**git commit –am "Added info.txt during gitIgnore Activity"**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git add .
warning: LF will be replaced by CRLF in bin/excludes/info.txt.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status -s
A  info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git commit -m "added info.txt during gitIgnore Activity"
[master 4f86487] added info.txt during gitIgnore Activity
 1 file changed, 3 insertions(+)
 create mode 100644 bin/excludes/info.txt
```

Step 3: Create a local .gitIgnore just for this repo:

a) Navigate to the root directory of your repo (the folder that has the .git folder in it).

[**touch .gitIgnore**]

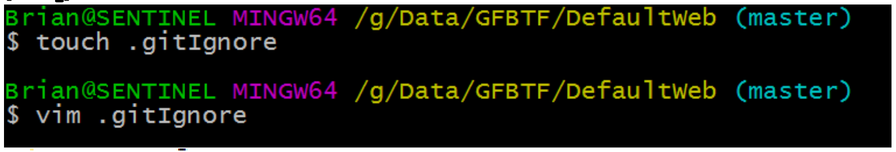
[**vim .gitIgnore**]

[**i**]

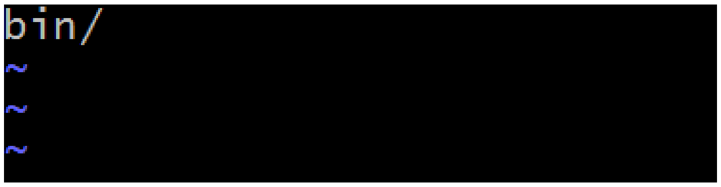
[**bin/**]

[**{esc}**]

[**:wq**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ touch .gitIgnore

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ vim .gitIgnore
```



```
bin/
~
~
~
```

Note that files we are already tracking (info.txt) will still be tracked. However, new files created in bin/ will be ignored.

b) Add another file to prove local .gitIgnore is working

Note that this points out the fact that our global .gitIgnore is NOT being tracked…

[navigate back to the bin folder or use full paths]

[**`touch anotherFile.txt`**]

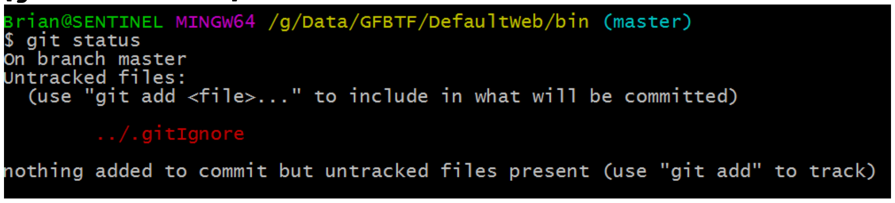
[**`vim anotherFile.txt`**]

[**`i`**]

[**`this is another file and it should be ignored`**]

[**`{esc}:wq`**]

[**`git status -s`**]



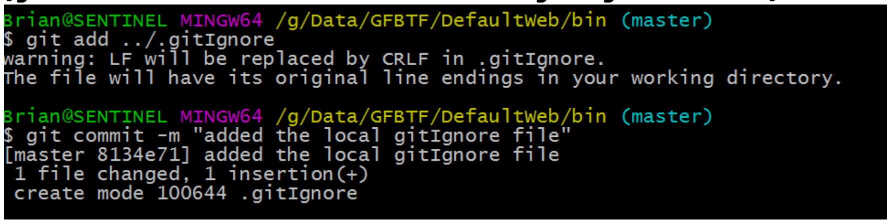
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ../.gitIgnore

nothing added to commit but untracked files present (use "git add" to track)
```

[we're able to add the new .gitIgnore but no sign of anotherFile.txt]

Add and commit the gitIgnore file [must be at the correct level or reference the path]

[**`git add ../.gitIgnore`**]

[**`git commit –m "added the local gitIgnoreFile"`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin (master)
$ git add ../.gitIgnore
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin (master)
$ git commit -m "added the local gitIgnore file"
[master 8134e71] added the local gitIgnore file
 1 file changed, 1 insertion(+)
 create mode 100644 .gitIgnore
```

Step 4: More changes to show tracked/untracked behaviors

a) Make changes to the info.txt file to prove it is still tracked even though it is in a folder that is ignored:

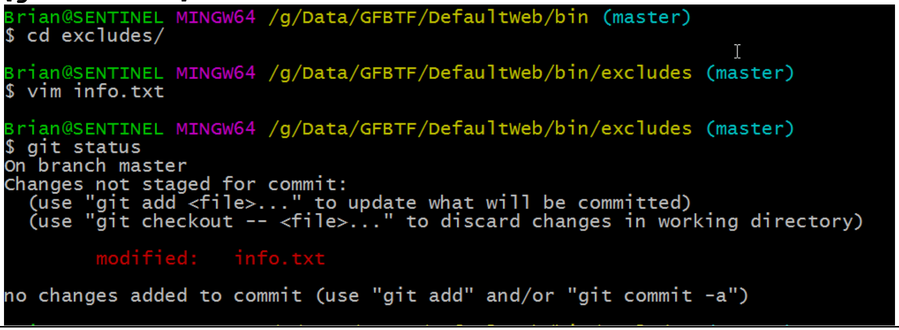
[**`vim info.txt`**]

[**`i`**]

[**`Still being tracked`**…]

[**`{esc}:wq`**]

[**`git status`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin (master)
$ cd excludes/

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ vim info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   info.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

So now we know the file is modified and tracked even though the folder is excluded.

b) Create another file in the subfolder of bin/excludes to prove the ignore is recursive for any subfolder under root of bin/:

[navigate to …/bin/excludes]

[**touch yetAnotherFile.txt**]

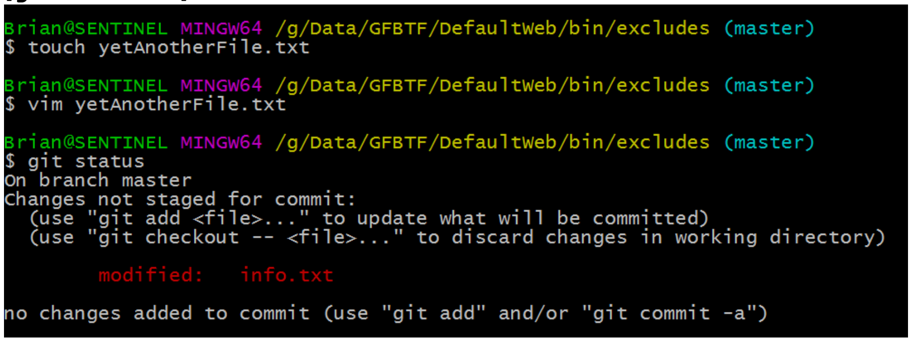
[**vim yetAnotherFile.txt**]

[**i**]

[**This is yet another file that will now be ignored**]

[**{esc}:wq**]

[**git status**]

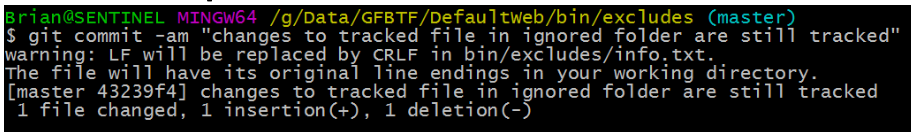


```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ touch yetAnotherFile.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ vim yetAnotherFile.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   info.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

So this proves the folder and subfolders are ignored now, but previously tracked files are still tracked.

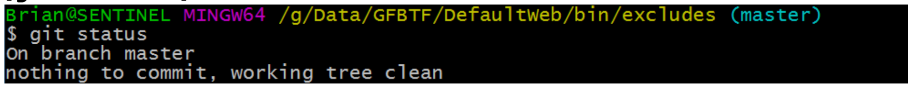
Add and commit the changes

[**git commit -am "changes to tracked file in ignored folder are still tracked"**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git commit -am "changes to tracked file in ignored folder are still tracked"
warning: LF will be replaced by CRLF in bin/excludes/info.txt.
The file will have its original line endings in your working directory.
[master 43239f4] changes to tracked file in ignored folder are still tracked
 1 file changed, 1 insertion(+), 1 deletion(-)
```

[**git status**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/bin/excludes (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Step 5: Adding Local resources that would ordinarily be ignored:

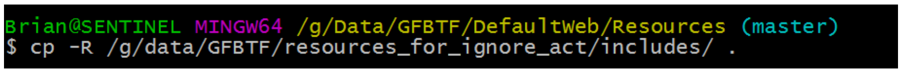
a) Sometimes we have files that we need to keep in the repo that would ordinarily be excluded

In practical situations, sometimes we need to keep a valuable resource around in order to have a valid version of it, or a reference to it in our project. Here we are going to simulate this practical situation

[navigate back to root]

[**mkdir Resources**]

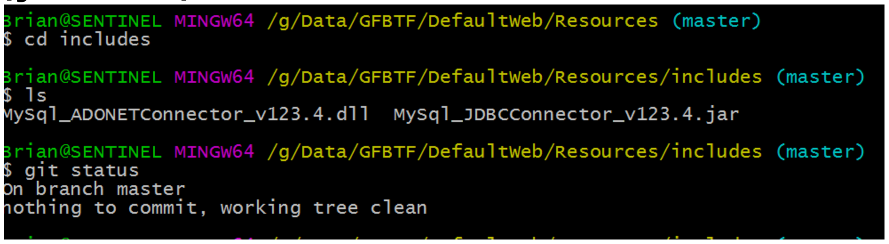
[copy the resource files into that folder]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources (master)
$ cp -R /g/data/GFBTF/resources_for_ignore_act/includes/ .
```

[navigate to the includes folder]

[**git status**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources (master)
$ cd includes

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ ls
MySql_ADONETConnector_v123.4.dll  MySql_JDBCConnector_v123.4.jar

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Note that the files are initially ignored, as expected for these types of files.

b) Include the resources

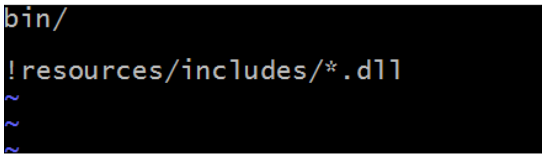
Next we need to re-include the files that are globally ignored. This will prove that our local .gitIgnore is able to supercede the global .gitIgnore

[navigate back to the root where the local .gitIgnore file is]

[**vim .gitIgnore**]

[**i**]

[**!resources/includes/*.dll**]



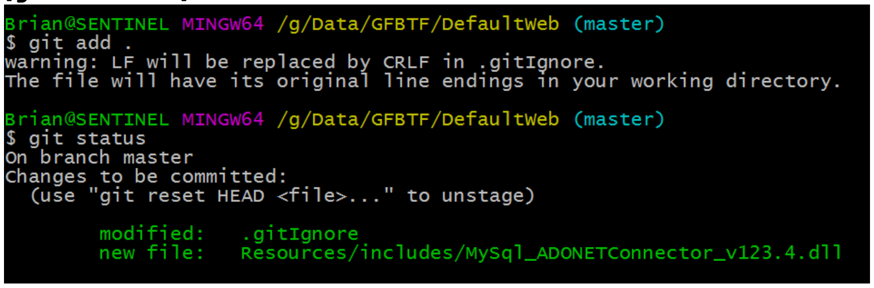
```
bin/

!resources/includes/*.dll
~
~
~
```

[**{esc}:wq**]

We are now including the includes folder dll [we could have included files directly by name, too, but this gets ALL dlls in the includes folder]

[**git add .**] //add the included dlls

[**git status**]



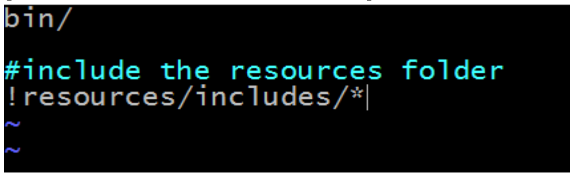
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git add .
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitIgnore
        new file:   Resources/includes/MySql_ADONETConnector_v123.4.dll
```

c) Include the whole folder, rather than just one file type

[**vim .gitIgnore**]

[**i**]

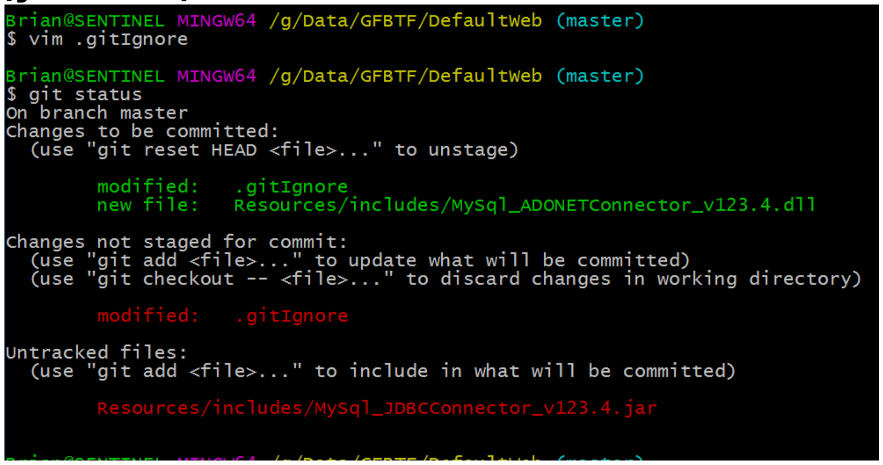
[**!resources/includes/***]



```
bin/

#include the resources folder
!resources/includes/*
~
~
```

[**{esc}:wq**]

[**git status**]



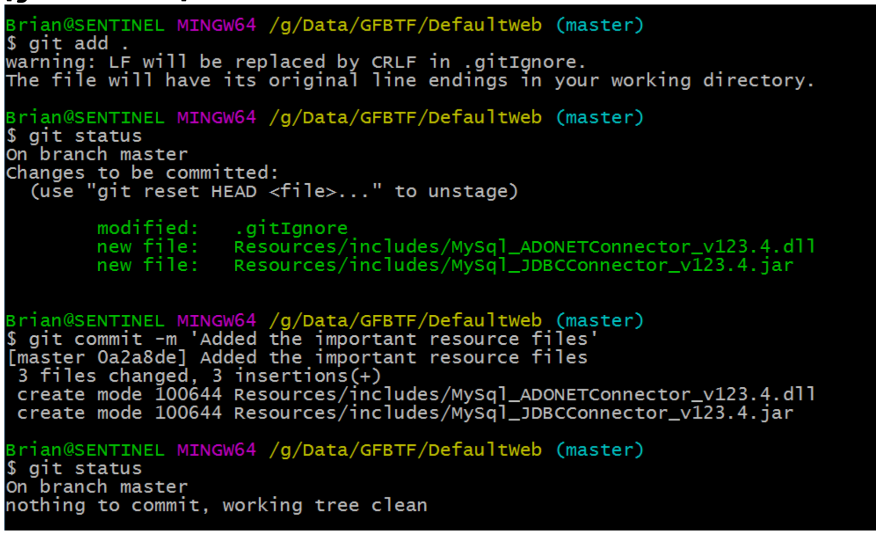
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ vim .gitIgnore

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitIgnore
        new file:   Resources/includes/MySql_ADONETConnector_v123.4.dll

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitIgnore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Resources/includes/MySql_JDBCConnector_v123.4.jar
```

Now all the files are found, we need to add and commit to track and keep them.

[**git add .**]

[**git commit -m "added the important resource files"**]

[**git status**]



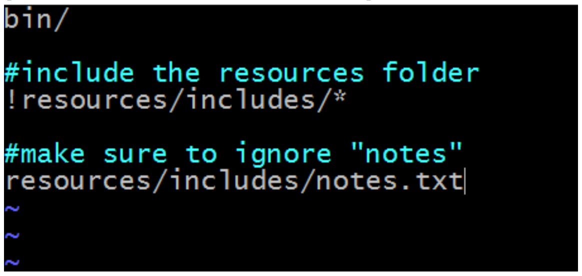
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git add .
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitIgnore
        new file:   Resources/includes/MySql_ADONETConnector_v123.4.dll
        new file:   Resources/includes/MySql_JDBCConnector_v123.4.jar


Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git commit -m 'Added the important resource files'
[master 0a2a8de] Added the important resource files
 3 files changed, 3 insertions(+)
 create mode 100644 Resources/includes/MySql_ADONETConnector_v123.4.dll
 create mode 100644 Resources/includes/MySql_JDBCConnector_v123.4.jar

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

d) Exclude a file after including the whole folder

[**vim .gitIgnore**]

[**i**]

[**!resources/includes/***]



```
bin/

#include the resources folder
!resources/includes/*

#make sure to ignore "notes"
resources/includes/notes.txt
~
~
~
```

[**{esc}:wq**]

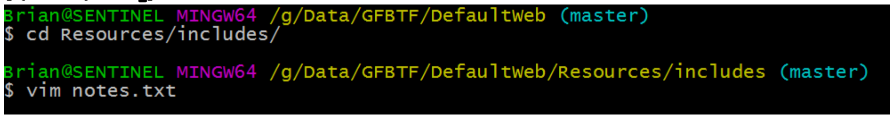
[navigate to the includes directory]

[**vim notes.txt**]

[**i**]

[**these are important notes for myself that we are not tracking**]

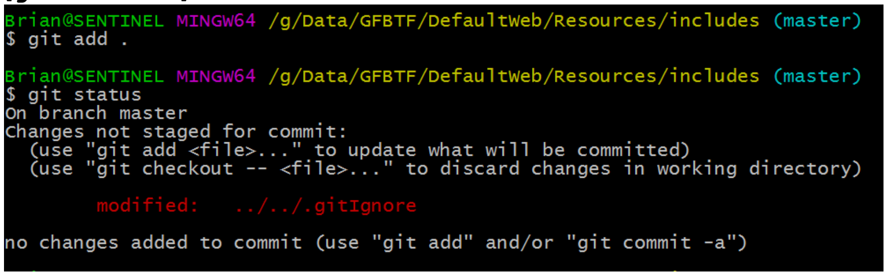
[**{esc}:wq**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ cd Resources/includes/

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ vim notes.txt
```

[**git add .**]

[**git status**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ git add .

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ../../.gitIgnore

no changes added to commit (use "git add" and/or "git commit -a")
```

e) Make sure can add another text file that would be tracked

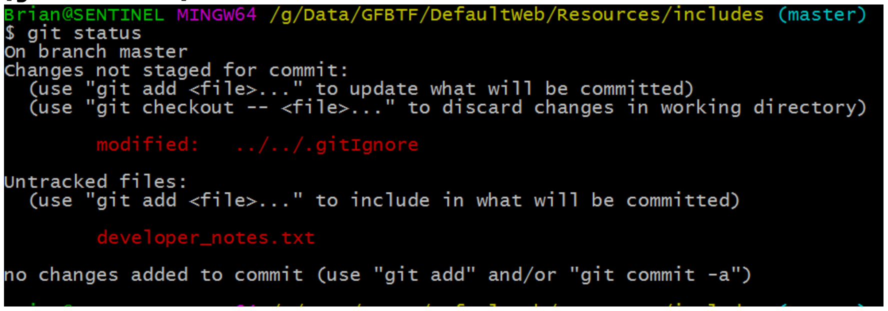
[**vim developer_notes.txt**]

[**i**]

[**these are important developer notes that everyone needs**]

[**{esc}:wq**]

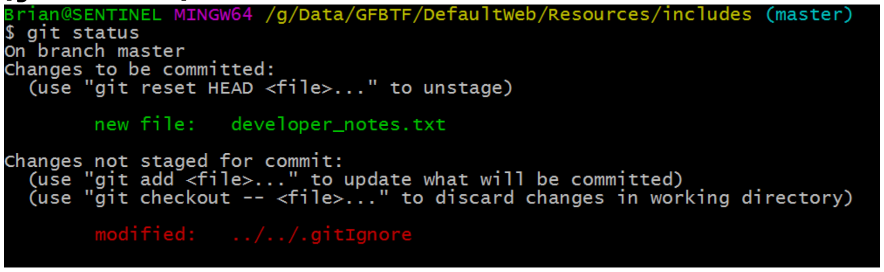
[**git status**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ../../.gitIgnore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        developer_notes.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

[**git add .**]

[**git status**]



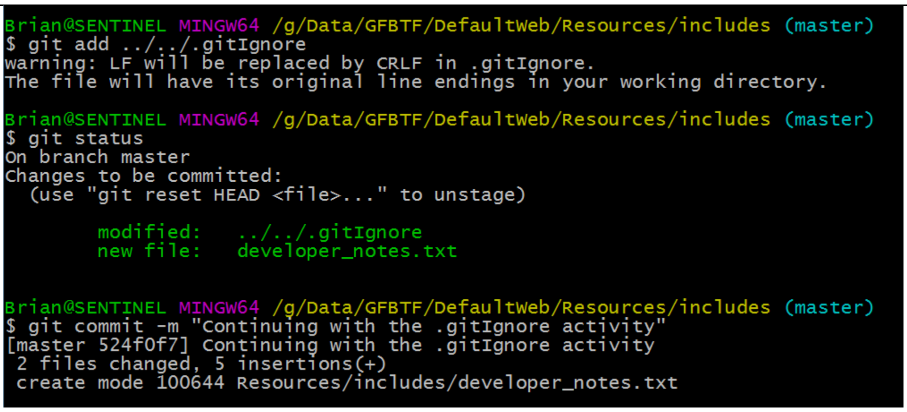
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Resources/includes (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   developer_notes.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ../../.gitIgnore
```

Need to add the .gitIgnore as well:

[**git add ../../.gitIgnore**]

[**git status**]

[**git commit -m "Continuing with the .gitIgnore Activity…"**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb/Resources/includes (master)
$ git add ../../.gitIgnore
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb/Resources/includes (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   ../../.gitIgnore
        new file:   developer_notes.txt


Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb/Resources/includes (master)
$ git commit -m "Continuing with the .gitIgnore activity"
[master 524f0f7] Continuing with the .gitIgnore activity
 2 files changed, 5 insertions(+)
 create mode 100644 Resources/includes/developer_notes.txt
```

So now we have seen that we can include entire folders, exclude entire folders, include specific files within excluded folders and exclude specific files using different statements in the .gitIgnore.

Also, due to the fact that the local .gitIgnore .dll works when named, we can see the hierarchy – that a local .gitIgnore can override a global gitIgnore.

Step 6: Using patterns in the .gitIgnore file:

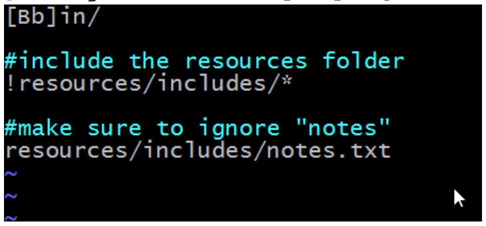
a) Make some changes

[navigate to the root folder]

[**vim .gitIgnore**]

[**i**]

[change bin/ to **[Bb]in**]



```
[Bb]in/

#include the resources folder
!resources/includes/*

#make sure to ignore "notes"
resources/includes/notes.txt
~
~
~
```

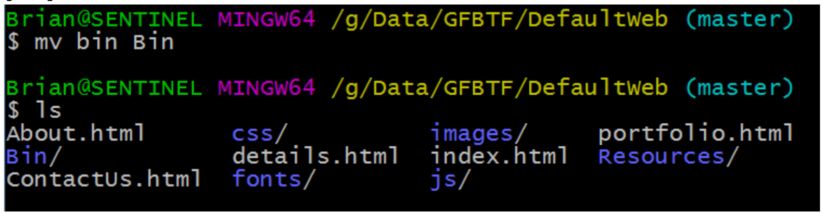
[**{esc}:wq**]

[**git status**]

[Rename the bin folder to Bin]

[**mv bin Bin**]

[**ls**]



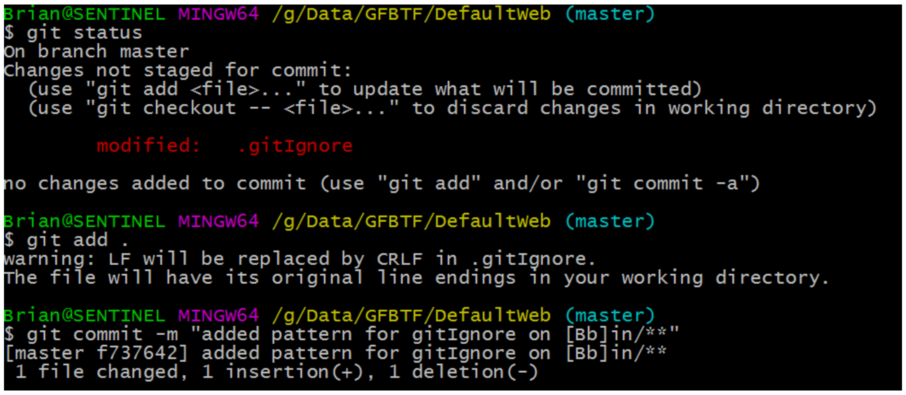
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb (master)
$ mv bin Bin

Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb (master)
$ ls
About.html      css/          images/     portfolio.html
Bin/            details.html  index.html  Resources/
ContactUs.html  fonts/        js/
```

b) Commit the changes

[**git status**]

[**git add .**]

[**git commit -m "added pattern for gitIgnore on [Bb]in/****]

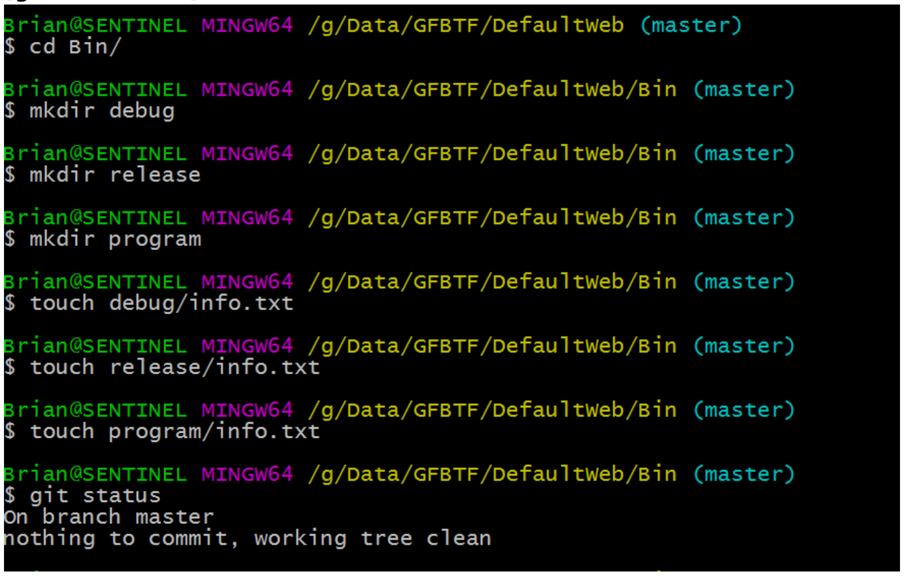


```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitIgnore

no changes added to commit (use "git add" and/or "git commit -a")

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git add .
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git commit -m "added pattern for gitIgnore on [Bb]in/**"
[master f737642] added pattern for gitIgnore on [Bb]in/**
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Note that even though we changed the directory from bin/ to [Bb]in/, excluded files are still excluded as would be expected.

c) Make sure subfolders are ignored as expected

Make directories 'debug', 'release', and 'program' under 'bin'
Add a simple text file to each.
Validate the files are ignored due to our new settings:
[**navigate to Bin**]
[**mkdir debug**]
[**mkdir release**]
[**mkdir program**]
[**touch debug/info.txt**]
[**touch release/info.txt**]
[**touch program/info.txt**]
[**git status**]



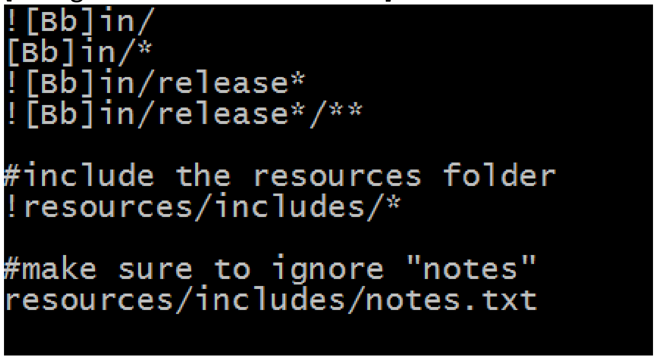
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ cd Bin/

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ mkdir debug

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ mkdir release

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ mkdir program

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ touch debug/info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ touch release/info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ touch program/info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb/Bin (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Notice that none of the files in the subfolders are showing up as untracked, they are simply ignored.

d) Track all files in a subfolder of an excluded folder (tricky)

[navigate to the root folder]

[**vim .gitIngore**]
[**i**]
[change the file to look like this:]



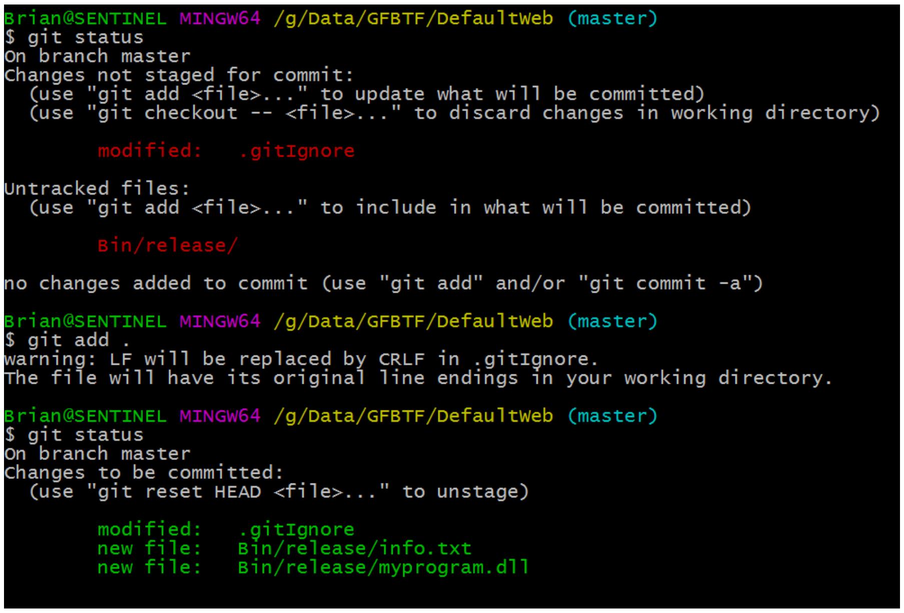
```
![Bb]in/
[Bb]in/*
![Bb]in/release*
![Bb]in/release*/**

#include the resources folder
!resources/includes/*

#make sure to ignore "notes"
resources/includes/notes.txt
```

[**{esc}:wq**]

The first line says 'don't exclude the bin folder at the top level – if it's excluded, no subfolders can show

The second line hides everything in the bin folder by default

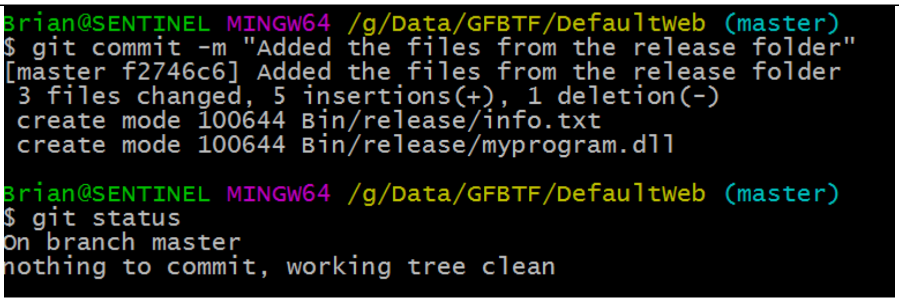
The third line unhides the release folder
The fourth line unhides all the files in release folder.



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitIgnore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Bin/release/

no changes added to commit (use "git add" and/or "git commit -a")

Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb (master)
$ git add .
warning: LF will be replaced by CRLF in .gitIgnore.
The file will have its original line endings in your working directory.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/Defaultweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitIgnore
        new file:   Bin/release/info.txt
        new file:   Bin/release/myprogram.dll
```

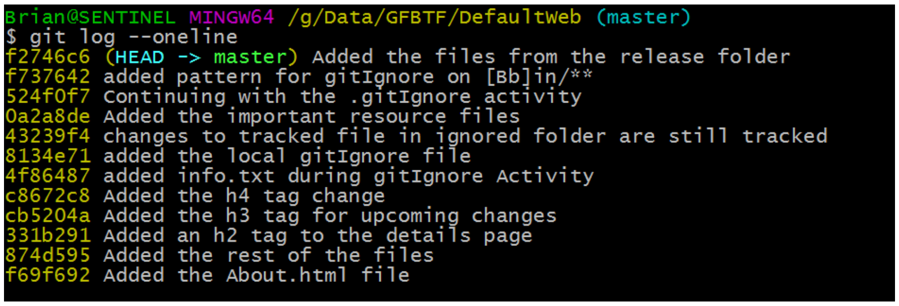
e) Commit and review the log

[**git add .**]
[**git status**]
[**git commit -m 'Added the files from the release folder'**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git commit -m "Added the files from the release folder"
[master f2746c6] Added the files from the release folder
 3 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 Bin/release/info.txt
 create mode 100644 Bin/release/myprogram.dll

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

[**git log -oneline**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git log --oneline
f2746c6 (HEAD -> master) Added the files from the release folder
f737642 added pattern for gitIgnore on [Bb]in/**
524f0f7 Continuing with the .gitIgnore activity
0a2a8de Added the important resource files
43239f4 changes to tracked file in ignored folder are still tracked
8134e71 added the local gitIgnore file
4f86487 added info.txt during gitIgnore Activity
c8672c8 Added the h4 tag change
cb5204a Added the h3 tag for upcoming changes
331b291 Added an h2 tag to the details page
874d595 Added the rest of the files
f69f692 Added the About.html file
```

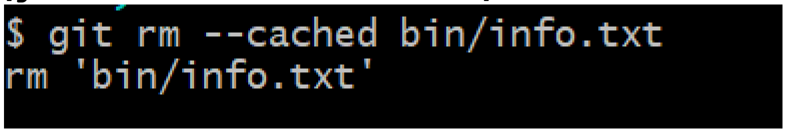
Should look something like this after working through both the status and the .gitIgnore activities.

Step 7: Remove a tracked but now ignored file:

a) Bin/info.txt needs to be untracked

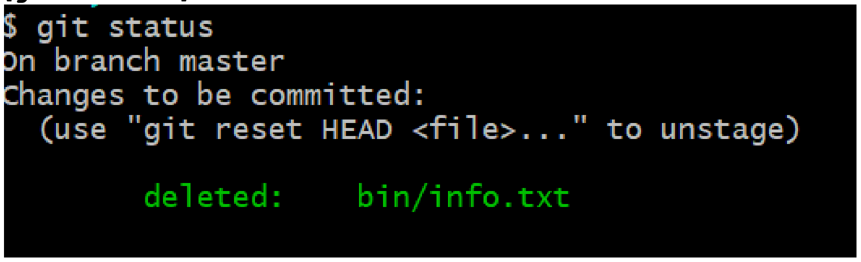
We no longer want to track this file but it was tracked before we ignored it. First, we need to remove it from the repository. Navigate to the root folder, then run the command:

[**git rm --cached bin/info.txt**]



```
$ git rm --cached bin/info.txt
rm 'bin/info.txt'
```

[**git status**]



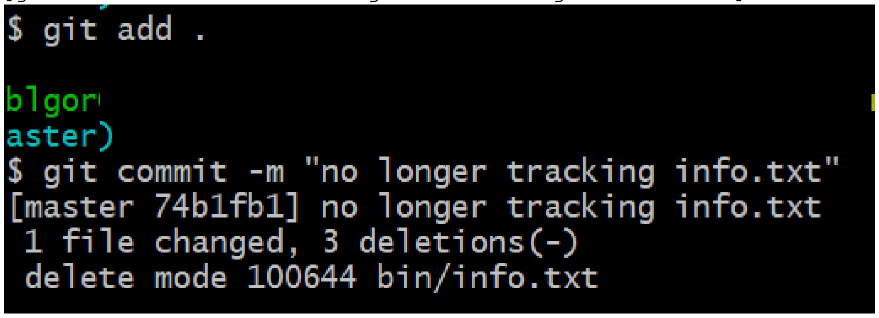
```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    bin/info.txt
```

b) Add and commit to make the changes final

Add and commit to make the deletion final

[git add .]

[git commit -m “no longer tracking info.txt”]



```
$ git add .

blgor
aster)
$ git commit -m "no longer tracking info.txt"
[master 74b1fb1] no longer tracking info.txt
 1 file changed, 3 deletions(-)
 delete mode 100644 bin/info.txt
```

c) Validate changes no longer tracked

Make another change to the info.txt file to validate it is no longer tracked

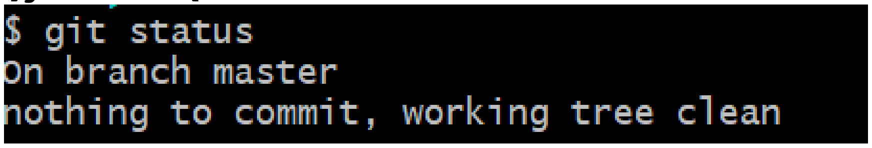
[**vim bin/info.txt**]

[**i**]

[**enter a line: “No longer tracked”**]

{esc}:wq]

[git status]



```
$ git status
On branch master
nothing to commit, working tree clean
```

The file is no longer tracked.

This concludes the GIT Ignore Activity

Closing Thoughts

In this activity, we looked at creating both a global and a local .gitIgnore file. We were able to prove that the local .gitIgnore file supersedes the global .gitIgnore file when we were able to keep .dll files that would have otherwise been excluded.

In our examination, we saw that it is very easy to exclude a folder and its subfolders, as well as include/exclude files by name. We also saw that it is possible to set patterns in the .gitIgnore file that allow for pattern matching to apply the rules that are appropriate for the repo or user.

We also learned how we can keep tracking a file that would otherwise be ignored, as well as how to remove files that are tracked prior to having been added to the .gitIgnore file for exclusion.

Take a few minutes to make some notes about the various commands we've learned about in this activity, and practice using them.

Notes

____

____

____

____

____

____

____

____

____

____

____

____