Git Workflow from installation to repository

Autor : Wim Cranen (wim.cranen@wccandm.nl – https://www.wccandm.nl)
Date : October 06, 2023
Status : concept
Version : 0.6

We will cover local repositories and remote repositories on GitHub, on your own local server and on a remote server.

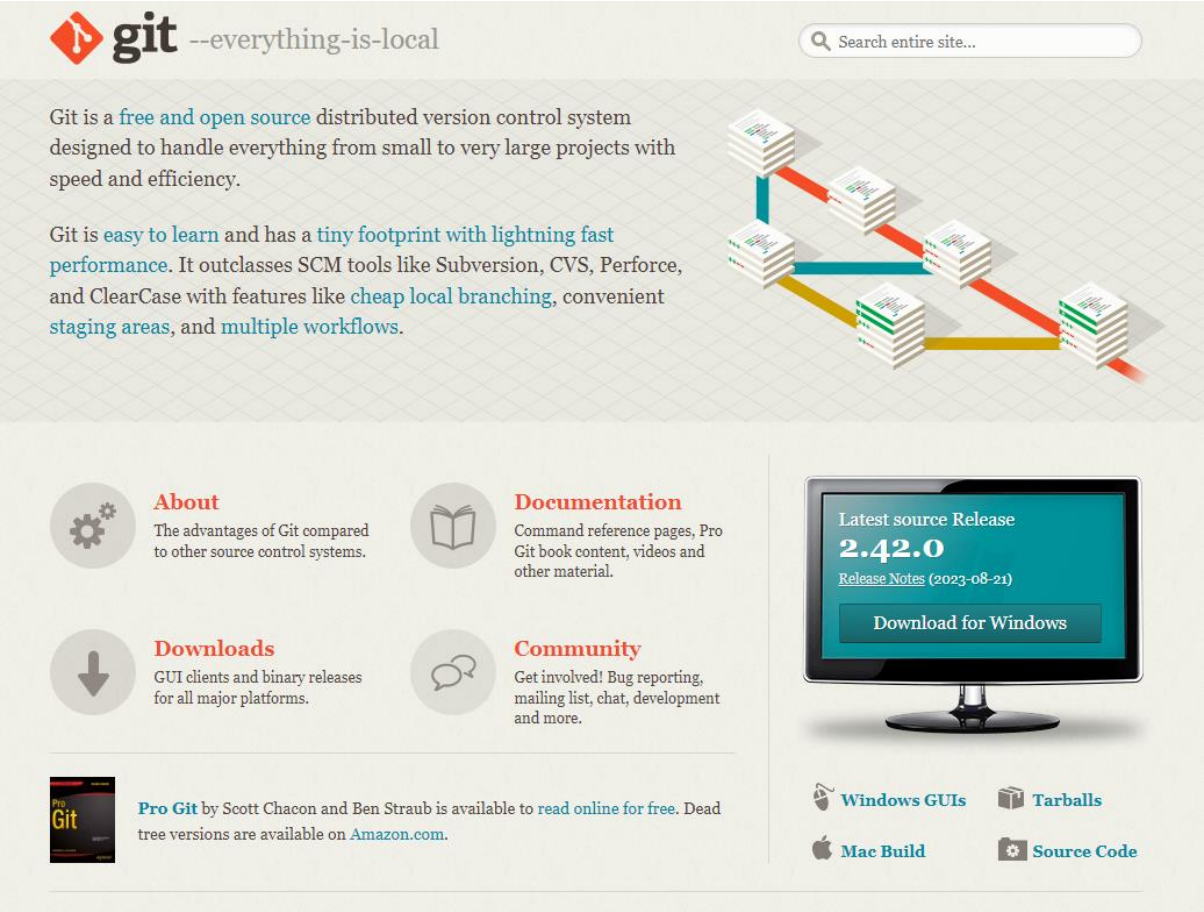
Inhoud

1. What is GIT, and why GIT?

Git is a version control system (vcs) originally developed for software development. But can be used for all text based change tracking. And that is what Git does. It tracks changes, not files.
Have close look at the following website. Especially the chapter 1.2 where the mechanism is explained and chapter 1.3 where versions, work area, staging area and repository are explained.

2. Installation of GIT.

Go to the GIT website and download the installer for your operating system. This example is for Windows.

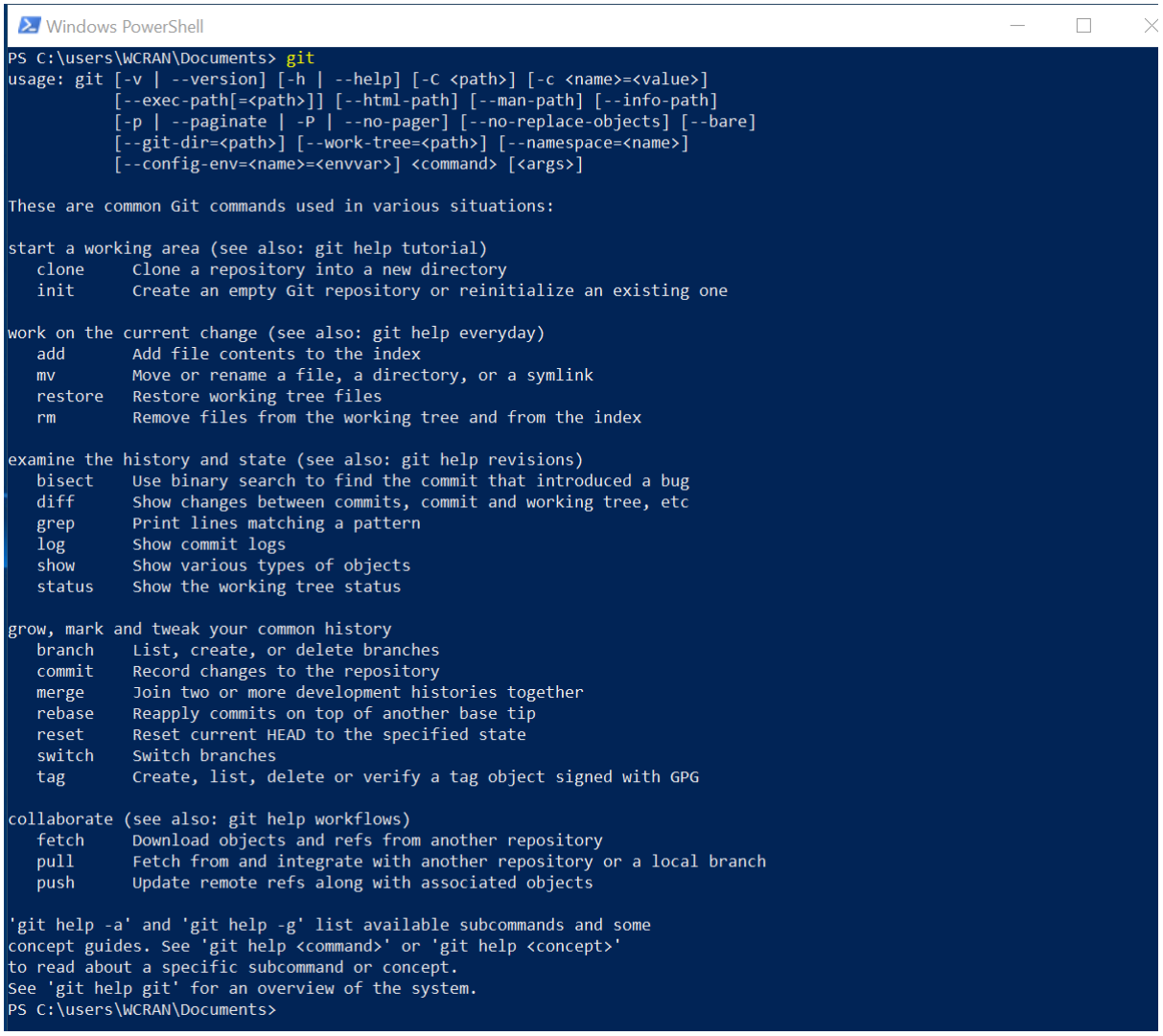


Start the installer. In this case the filename is: `Git-2.42.0.2-64-bit.exe`.
Change no options during installing, just click next until finish.

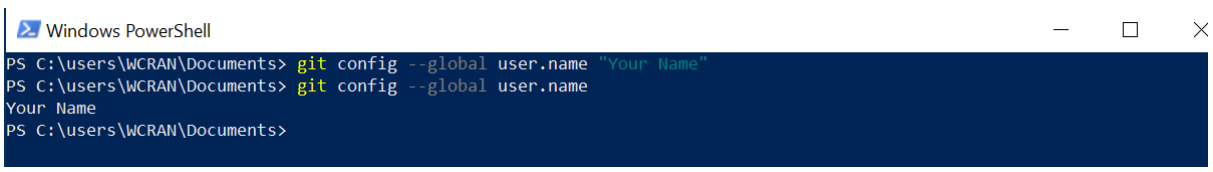
Git can be operated in the Windows power shell. Open power shell and navigate to your Documents folder (example).

3. GIT initialize.

In the power shell window type the following command:
`Git --version`
Or just `git`
This gives a whole lot of information about git and ensures that the installation is ok.

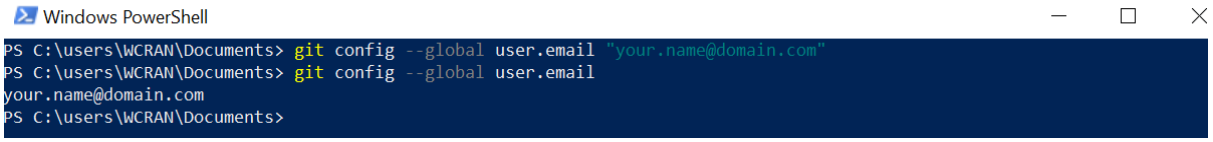


Then we have to initialize GIT, by giving the name and email address of the user. Insert the following command:
`git config --global user.name "Your Name"`
And verify with this command if the user name is set correctly:
`git config --global user.name`



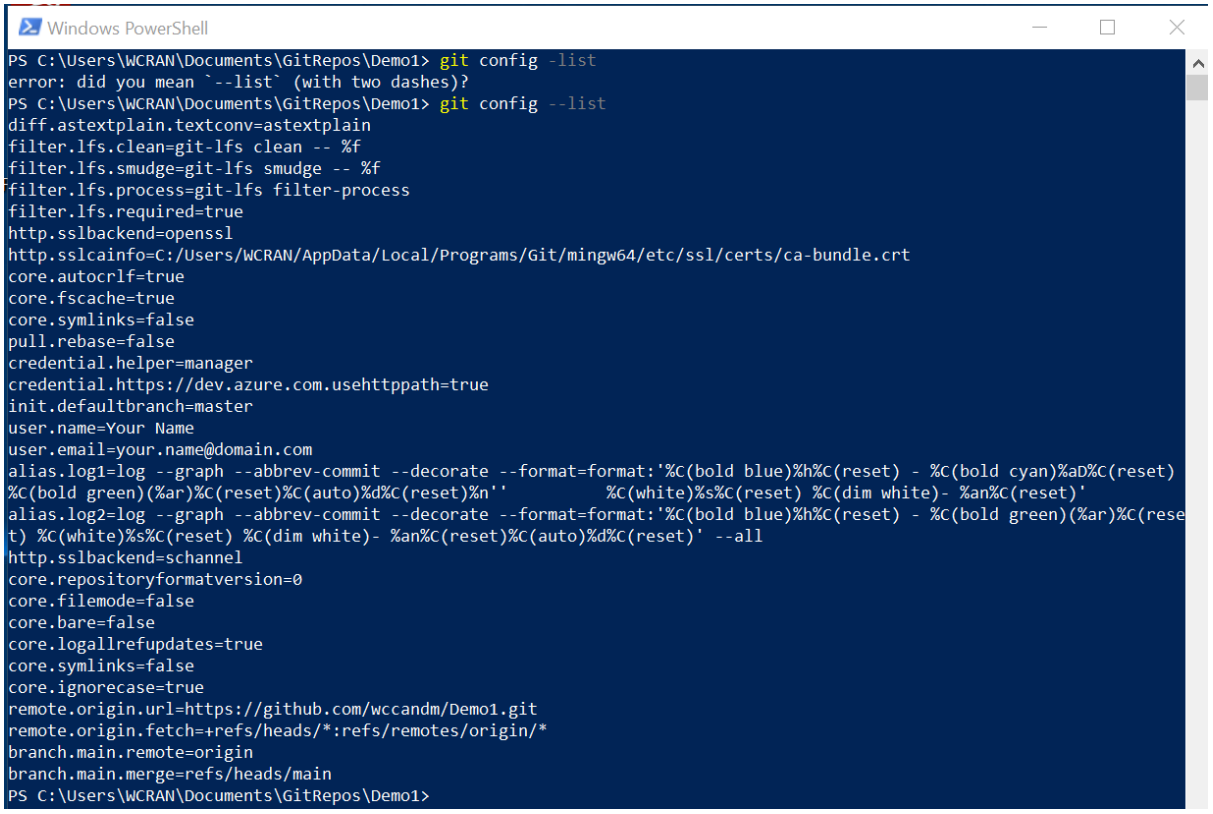
Then you have to set your email address with:
`git config --global user.email "your.name@domain.com"`
And verify with this command if your email address is set correctly:

`git config --global user.email`



Now your git environment is ready for operation.

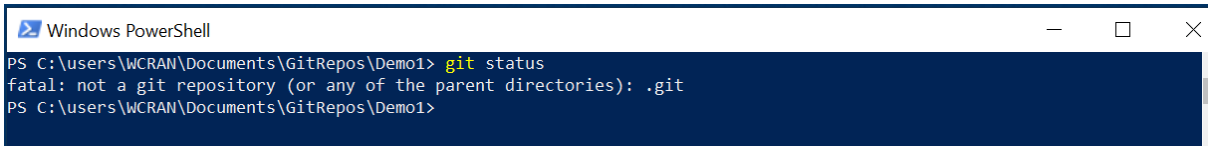
With the `git config --list` command, you can list all global variables for GIT.



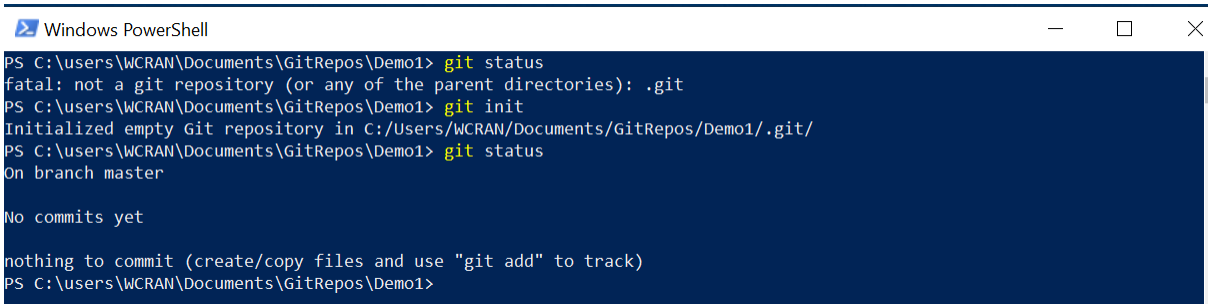
4. Your first (local) repository.

In the documents folder create an new folder with the name “GitRepos” change to that folder and in the GitRepos folder create an new folder with the name “Demo1” and change into that folder with power shell.

Enter the command: `git status`.

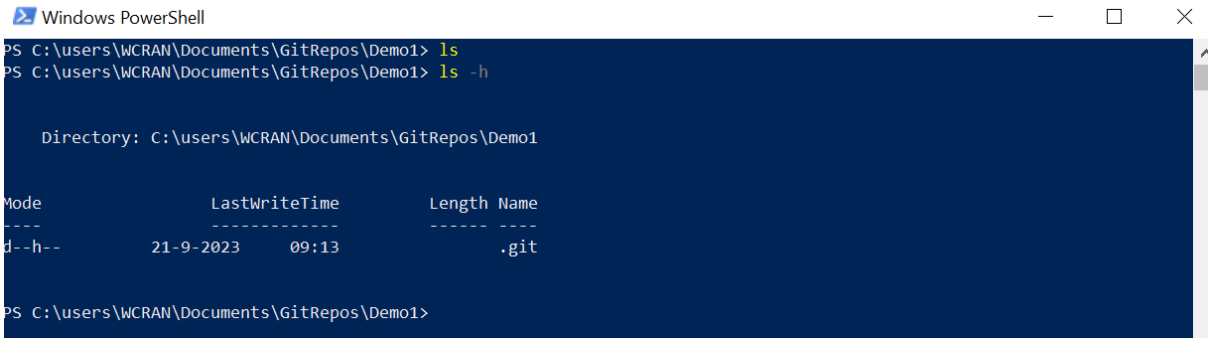


The answer is: this is not a git repository. So we are going to create it.
Give the command: `git init` and then `git status` again.



Now that GIT is initialized, and we see that we are on the master branch and no commits have been done yet and there is nothing to commit because the directory or folder is empty.

If we give the `ls` command, then we see that the folder is empty… but how can we track changes then and how does git know that there is nothing to commit? Give the `ls -h` command. This command also shows the hidden files. Now we see that there is a hidden folder called “.git”.

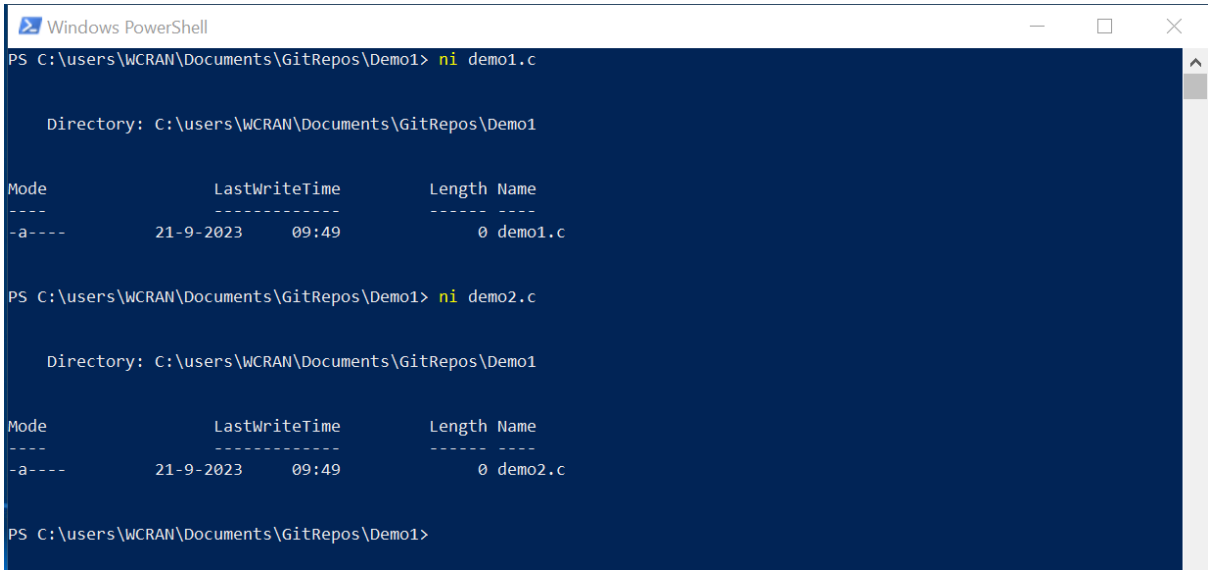


This .git folder contains all information that git saves.
If you want to get rid of the git information, just give the command `rm .git` and answer with “Y”.

5. Create files.

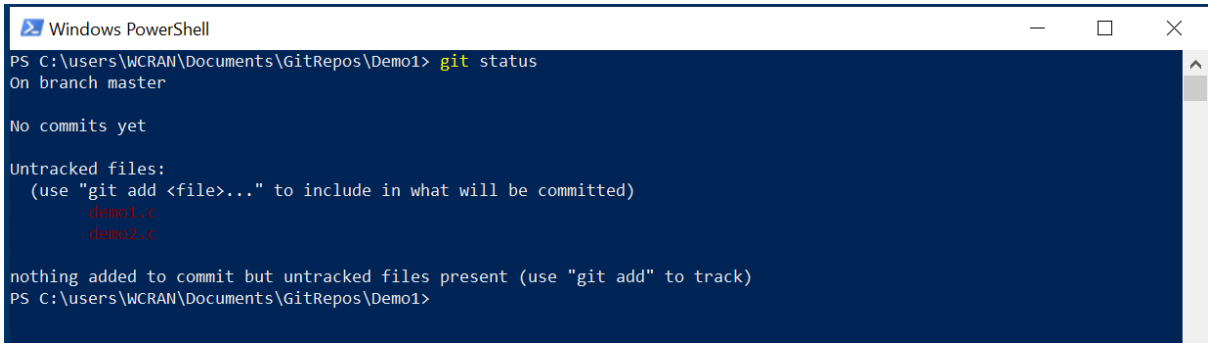
We are going to create two files with the following commands:

```
ni demo1.c
ni demo2.c
```

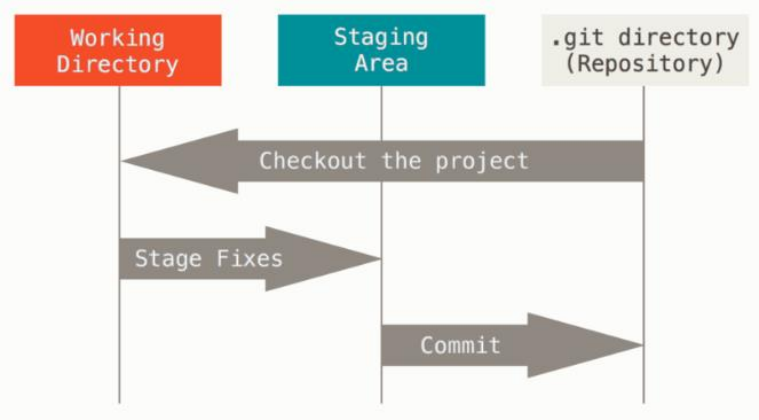


Then we give the `ls -force` command. This gives all files, also the hidden ones.

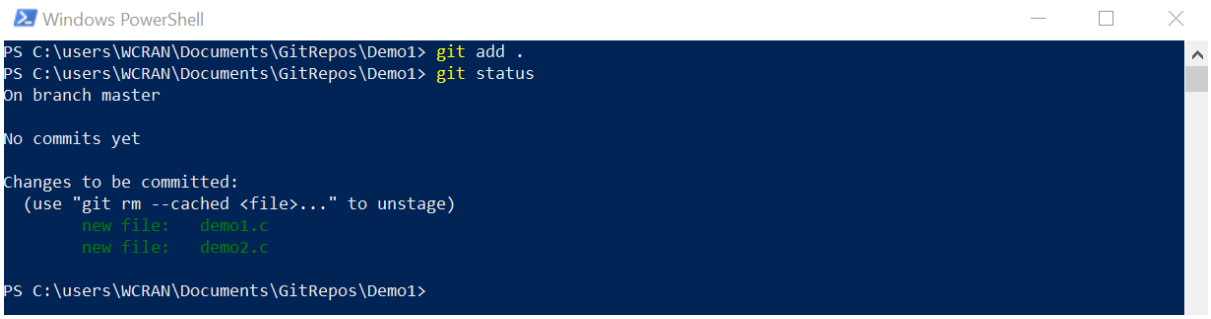
As we can see, the .git directory (d and h Mode) and two files are present (archive mode).
Then we want to know the status of git. Give the command `git status`.



We are still on the master branch, there are still no commits and we have two untracked files. Remember we are in the working directory. To have git trace the changes of these files the have to be moved to the staging area. If the files are in the staging area, we can do a commit to bring the files in the repository. Let's go.



By adding the files, we bring the files in the staging area. We can add files with the following commands:
`git add filename` (this adds only the specific filename)
`git add directory` (this adds only the specific directory)
`git add .` (this adds everything)
We are going to add everything and do a `git status` after that.



The we see that we are still on the master branch, no commits done, but there are changes that can be committed. The two files are tracked and are in the staging area.
Now we can do a commit with the following command:
`git commit -m "message"` and ask for the status and a log.
We give the command: `git commit -m "v0.0 - Initial commit"`
This commits the files to the repository with the message (-m): v0.0 – Initial Commit.
Then: `git status`
That gives us the information that we are on the master branch and that there is nothing to commit.
Then: `git log`
Gives us the information about the author, the date and the commit message.
Please be as specific as possible with this message, which can go over several lines.
Also see that this commit has got a code or a hash, in this case:
`af12b458e50f0200938c3f3ed9225bdd017db248`
Then there is information about HEAD -> master, where HEAD refers to the current branch.

All ready for now, the repository is clean and ready.

6. Change and track changes.

Now we are going to make changes in one of the files. Open your “Demo1” directory in the windows explorer and with a right mouse click start Notepad or Notepad++ with the “Demo1.c” file.
Add the following information to it and save and close the file.

demo1.c

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

The text can be copied from here.
Then we do a `ls -force` and a `git status`.

Windows PowerShell

```
PS C:\users\WCRAN\Documents\GitRepos\Demo1> ls -force


    Directory: C:\users\WCRAN\Documents\GitRepos\Demo1


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--         21-9-2023     10:15                .git
-a----         21-9-2023     10:36           1275 demo1.c
-a----         21-9-2023     09:49              0 demo2.c


PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo1.c

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

We see that the file “Demo1.c” has got some content (1275 bytes), we see that we are still in the master branch, there are changes but they are not staged and nothing to commit. So we are going to stage the Demo1.c file.

```
git add demo1.c
git status
git commit -m “v0.1 – Demo1.c changed with Arduino blink sketch”
git status
git log
```

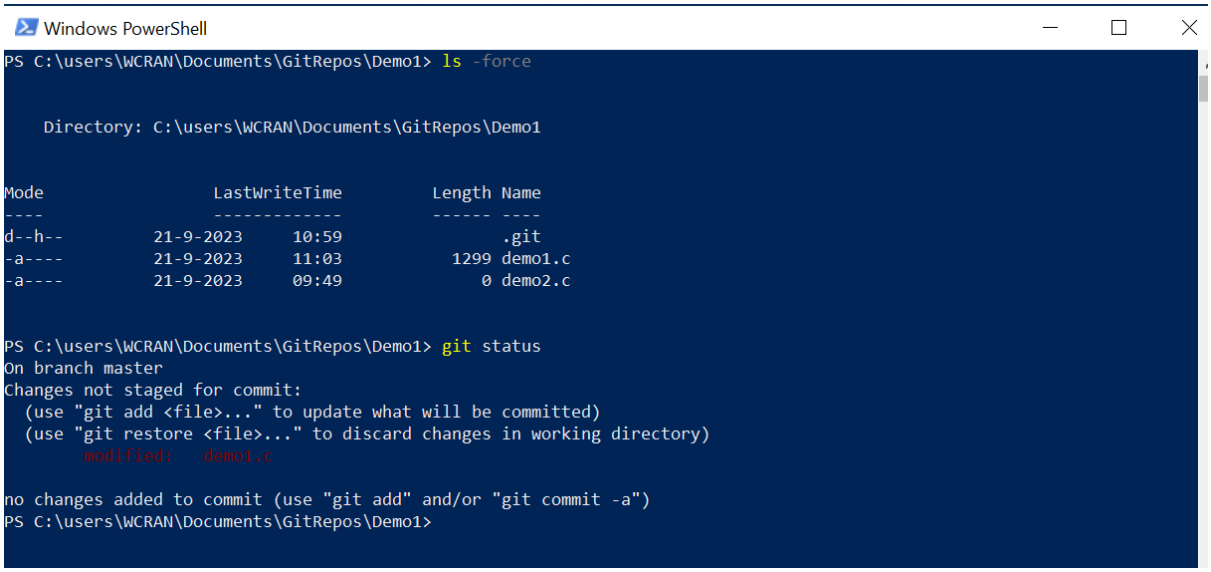
```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git add demo1.c
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demo1.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1> git commit -m "v0.1 - Demo1.c changed with Arduino blink sketch"
[master 2147b2c] v0.1 – Demo1.c changed with Arduino blink sketch
 1 file changed, 37 insertions(+)
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
nothing to commit, working tree clean
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log
commit 2147b2cb7e7ccd888fcf6b6bcd1a4206fc8e3fc0 (HEAD -> master)
Author: Your Name <your.name@domain.com>
Date:   Thu Sep 21 10:58:57 2023 +0200

    v0.1 – Demo1.c changed with Arduino blink sketch

commit c75991efcce3756f0160dc808cf1bae982b80bd5
Author: Your Name <your.name@domain.com>
Date:   Thu Sep 21 10:52:14 2023 +0200

    Revert "v0.1 – Demo1.c changed with Aduino blink sketch"

    This reverts commit ed248b6d5920a67dadd53c501bae468a79f58963.

commit ed248b6d5920a67dadd53c501bae468a79f58963
Author: Your Name <your.name@domain.com>
Date:   Thu Sep 21 10:45:18 2023 +0200

    v0.1 – Demo1.c changed with Aduino blink sketch

commit af12b458e50f0200938c3f3ed9225bdd017db248
Author: Your Name <your.name@domain.com>
Date:   Thu Sep 21 10:14:37 2023 +0200

    v0.0 - Initial commit
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

If you made a mistake in the commit, you can revert the commit with:
`git revert <commit_hash> --no-edit` This will also revert your changes.
if you only want to undo the last commit, give:
`git reset --soft HEAD~1` This will undo the commit but leaves your changes intact.
in the log, you can see that I made a revert, nothing keeps unseen, everything is tracked.

Now we are going to make a slight change in Demo1.c. Add the following:

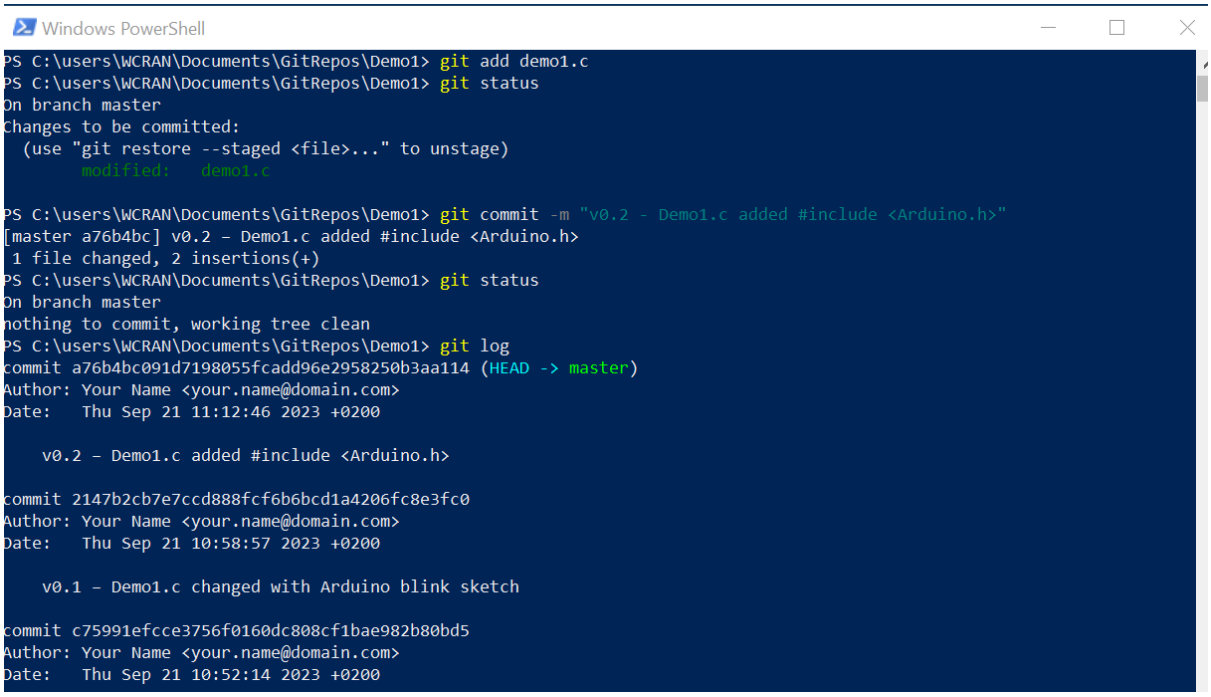
```
22      https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23  */
24
25  #include <Arduino.h>
26
27  // the setup function runs once when you press reset or power the board
```

See the yellow line.
Then we do a `ls -force` and a `git status`.



We see that the file “Demo1.c” has got more content (1299 bytes), we see that we are still in the master branch, there are changes but they are not staged and nothing to commit. So we are going to stage the Demo1.c file again.

```
git add demo1.c
git status
git commit -m "v0.2 - Demo1.c added #include <Arduino.h>"
git status
git log
```



You can see that the log is growing and changes are nicely tracked.
A nicer log can be called by:

```
git log --graph --abbrev-commit --decorate --format=format:'%C(bold
blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold
green)(%ar)%C(reset)%C(auto)%d%C(reset)%n''          %C(white)%s%C(reset)
%C(dim white)- %an%C(reset)'
```

Or:

```
git log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(auto)%d%C(reset)' --all
```

These can be kept in aliases.

```
git config --global alias.log1 "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(auto)%d%C(reset)%n''
%C(white)%s%C(reset) %C(dim white)- %an%C(reset)'"
```

Or:

```
git config --global alias.log2 "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(auto)%d%C(reset)' --all"
```

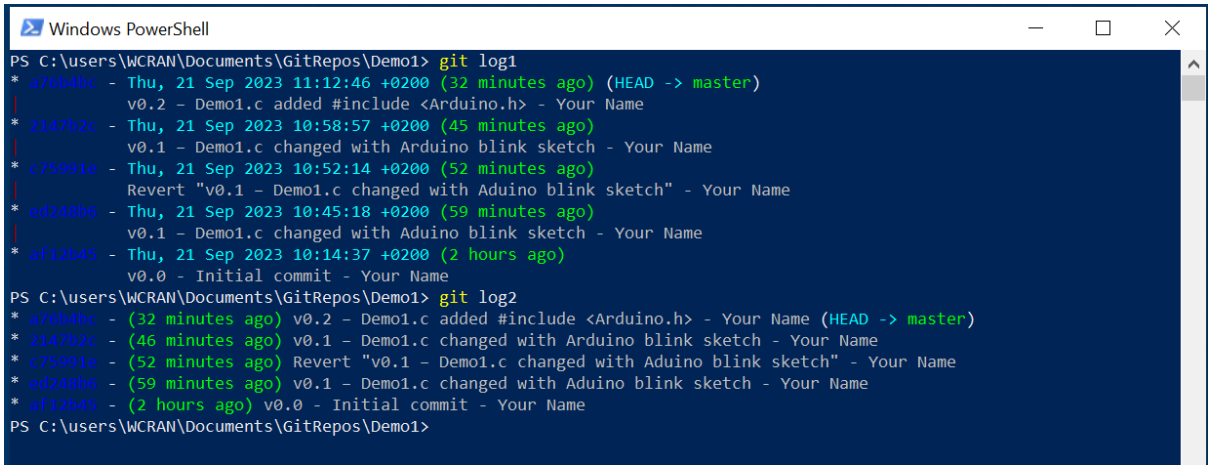
The you can give the commands:

```
git log1
```

Or

```
git log2
```

And that gives the following results:



7. Create a new branch.

Lets make a new branch to test a new feature. The existing branch stays intact. Issue the following commands:

```
git branch feature1
git checkout feature1
```

We see that we switched to feater1 branch

```
git status
```

We are on feature1 nothing to commit.
We are going to make changes lines 26/27, 38 and 40.

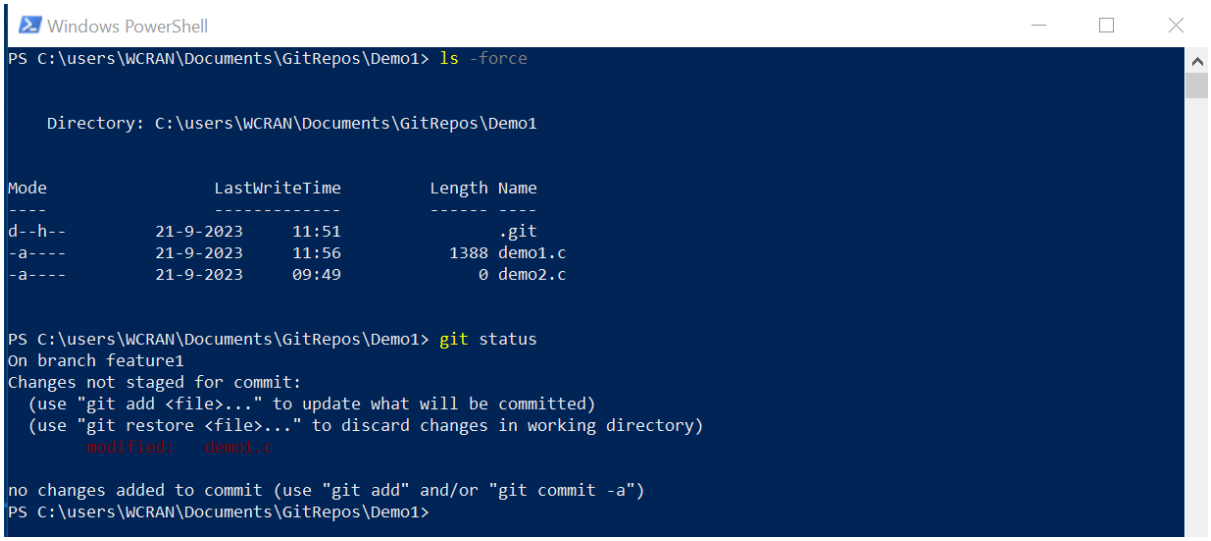
```
1  /*
2    Blink
3
4    Turns an LED on for one second, then off for one second, repeatedly.
5
6    Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7    it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8    the correct LED pin independent of which board is used.
9    If you want to know what pin the on-board LED is connected to on your Arduino
10   model, check the Technical Specs of your board at:
11   https://www.arduino.cc/en/Main/Products
12
13   modified 8 May 2014
14   by Scott Fitzgerald
15   modified 2 Sep 2016
16   by Arturo Guadalupi
17   modified 8 Sep 2016
18   by Colby Newman
19
20   This example code is in the public domain.
21
22   https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24
25 #include <Arduino.h>
26
27 int iBlinkSpeed = 1000;             // the blink speed in milli seconds
28
29 // the setup function runs once when you press reset or power the board
30 void setup() {
31   // initialize digital pin LED_BUILTIN as an output.
32   pinMode(LED_BUILTIN, OUTPUT);
33 }
34
35 // the loop function runs over and over again forever
36 void loop() {
37   digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
38   delay(iBlinkSpeed);                       // wait for a second
39   digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
40   delay(iBlinkSpeed);                       // wait for a second
41 }
```

Then we do a `ls -force` and a `git status`.

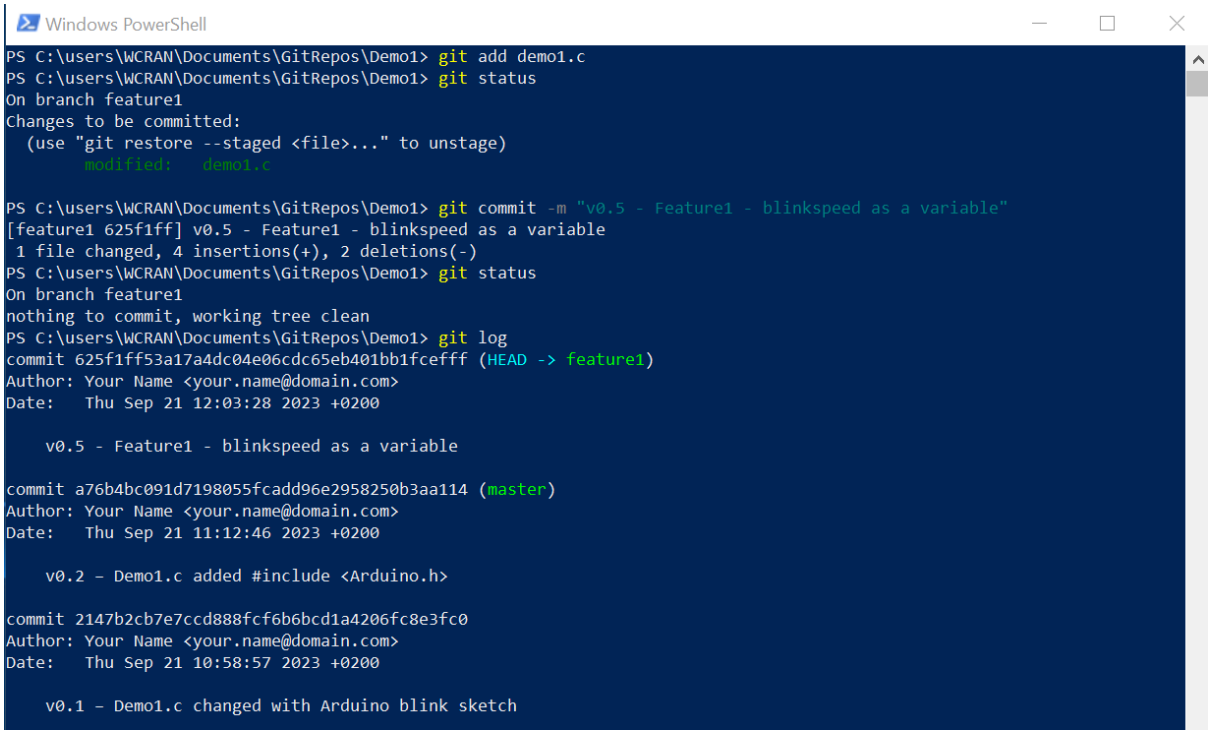
The file demo1.c has changed again and now counts 1388 bytes.

We are now on the feature branch, the file demo1.c is not staged, nothing to commit.

We are going to add demo1.c to the staging area.




```
git add demo1.c
git status
git commit -m "v0.2 - Demo1.c added #include <Arduino.h>"
git status
git log
```



We can see that we are on the feature1 branch (HEAD is pointing to this).

Lets do a `git log --graph`.

To see a graphical overview of the branches.

```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log --graph
* commit 625f1ff53a17a4dc04e06cdc65eb401bb1fcefff (HEAD -> feature1)
| Author: Your Name <your.name@domain.com>
| Date:   Thu Sep 21 12:03:28 2023 +0200
|
|     v0.5 - Feature1 - blinkspeed as a variable
|
* commit a76b4bc091d7198055fcadd96e2958250b3aa114 (master)
| Author: Your Name <your.name@domain.com>
| Date:   Thu Sep 21 11:12:46 2023 +0200
|
|     v0.2 – Demo1.c added #include <Arduino.h>
|
* commit 2147b2cb7e7ccd888fcf6b6bcd1a4206fc8e3fc0
| Author: Your Name <your.name@domain.com>
| Date:   Thu Sep 21 10:58:57 2023 +0200
|
|     v0.1 – Demo1.c changed with Arduino blink sketch
|
* commit c75991efcce3756f0160dc808cf1bae982b80bd5
| Author: Your Name <your.name@domain.com>
| Date:   Thu Sep 21 10:52:14 2023 +0200
|
|     Revert "v0.1 – Demo1.c changed with Aduino blink sketch"
|
|     This reverts commit ed248b6d5920a67dadd53c501bae468a79f58963.
|
* commit ed248b6d5920a67dadd53c501bae468a79f58963
| Author: Your Name <your.name@domain.com>
| Date:   Thu Sep 21 10:45:18 2023 +0200
|
|     v0.1 – Demo1.c changed with Aduino blink sketch
|
* commit af12b458e50f0200938c3f3ed9225bdd017db248
  Author: Your Name <your.name@domain.com>
  Date:   Thu Sep 21 10:14:37 2023 +0200

      v0.0 - Initial commit
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

Go back to the master branch.

```
git checkout master
```

Open the demo1.c file and see that the changes in lines 26/27, 38 and 40 are gone.

Now make some changes, change 1000 by 500 to make it faster.

Stage the demo1.c file and commit the change. Then perform a git log2.

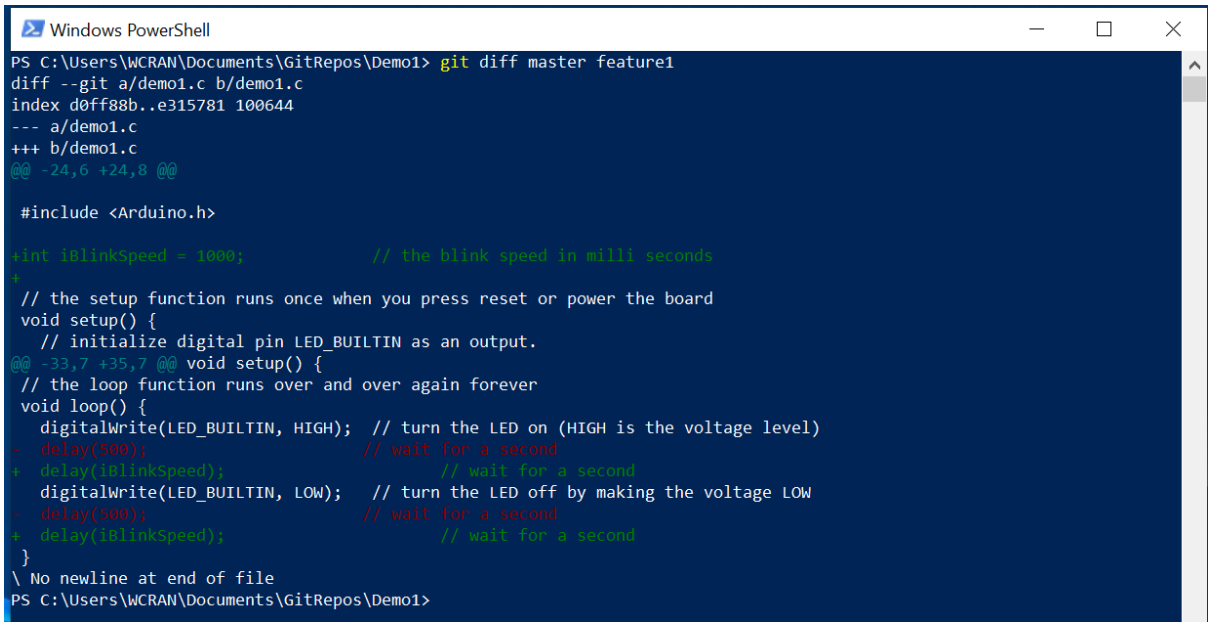
```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log2
* a4b3db0 - (2 minutes ago) v0.3 - Made blink faster - Your Name (HEAD -> master)
| * 625f1ff - (23 minutes ago) v0.5 - Feature1 - blinkspeed as a variable - Your Name (feature1)
|/
* a76b4bc - (73 minutes ago) v0.2 – Demo1.c added #include <Arduino.h> - Your Name
* 2147b2c - (87 minutes ago) v0.1 – Demo1.c changed with Arduino blink sketch - Your Name
* c75991e - (2 hours ago) Revert "v0.1 – Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - (2 hours ago) v0.1 – Demo1.c changed with Aduino blink sketch - Your Name
* af12b45 - (2 hours ago) v0.0 - Initial commit - Your Name
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

Now you can see the branch feature1 going away from the master branch.
HEAD is pointing to master again.

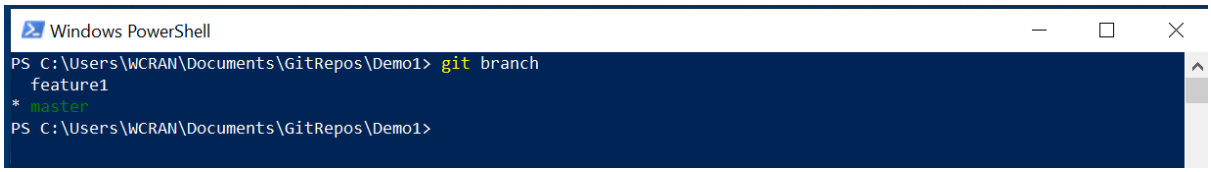
8. Merge branches.

Before we going to merge the branches, we want to show the differences from master to feature1 branch. Issue the following command:

```
git diff master..feature1
```



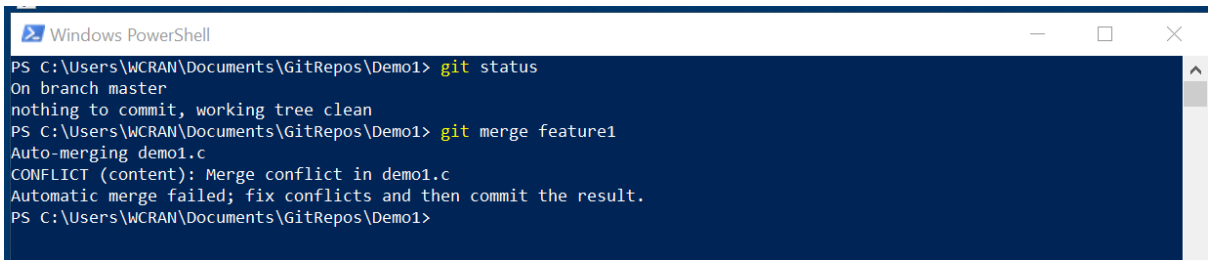
This gives us an overview of the changes. Merging these branches will give us some merge conflicts.



We see that there are two branches and master is the current branch.

Be sure to be on the master branch (`git status` or `git branch`) and issue the following command:

```
git merge feature1
```



We see that an automatic merge failed and we have to fix the conflicts.

Now the `git diff` command can help us.

```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git diff
diff --cc demo1.c
index d0ff88b,e315781..0000000
--- a/demo1.c
+++ b/demo1.c
@@@ -33,7 -35,7 +35,13 @@@ void setup()
  // the loop function runs over and over again forever
  void loop() {
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
++<<<<<<< HEAD
 +  delay(500);                      // wait for a second
 +  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
 +  delay(500);                      // wait for a second
++=======
+    delay(iBlinkSpeed);                      // wait for a second
+    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
+    delay(iBlinkSpeed);                      // wait for a second
++>>>>>>> feature1
  }
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

This shows us that in the master branch the delay has a number and in the feature branch the delay has a variable.

Let us open the demo1.c file in the master and fix the problem:

demo1.c

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

#include <Arduino.h>

int iBlinkSpeed = 1000;             // the blink speed in milli seconds

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
<<<<<<< HEAD
  delay(iBlinkSpeed);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(500);                      // wait for a second
=======
  delay(iBlinkSpeed);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(iBlinkSpeed);                       // wait for a second
>>>>>>> feature1
}
```

This would be ok:

demo1.c

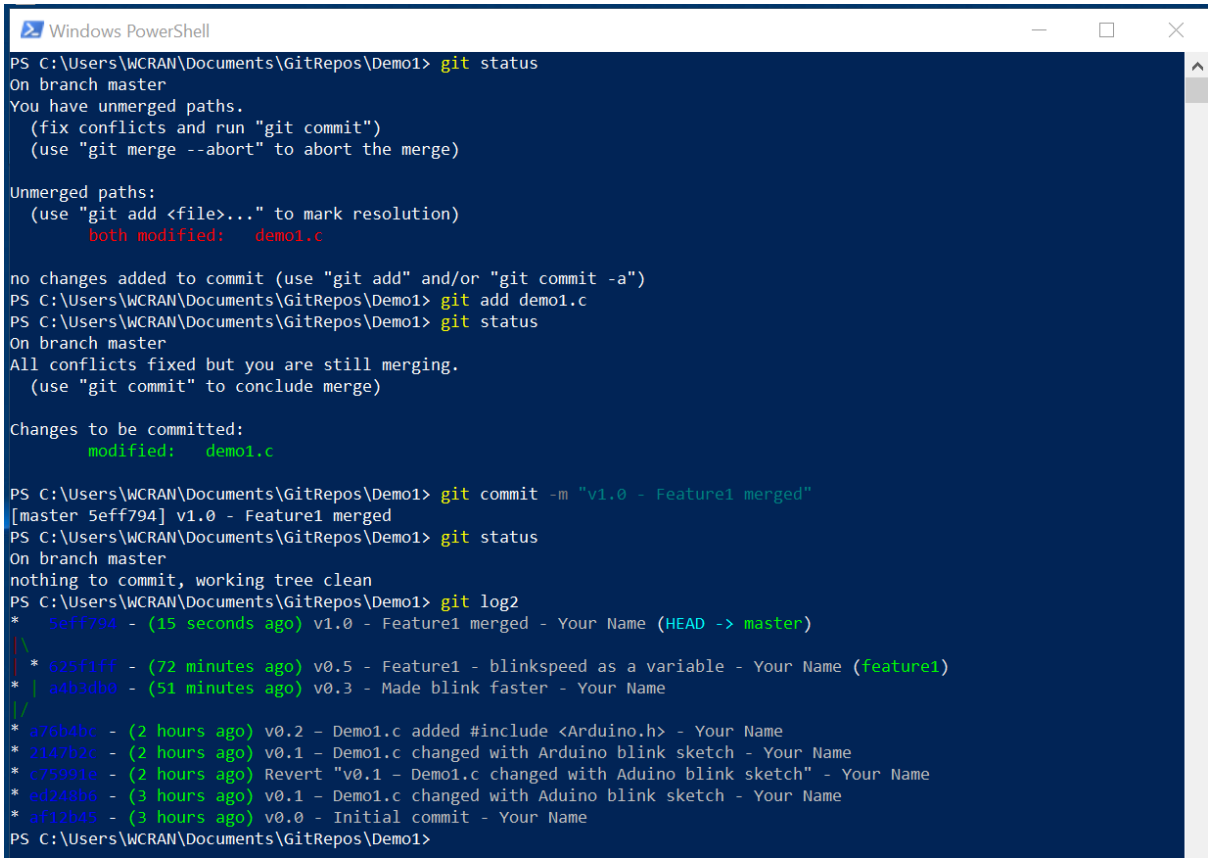
```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

#include <Arduino.h>

int iBlinkSpeed = 1000;             // the blink speed in milli seconds

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(iBlinkSpeed);               // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(iBlinkSpeed);               // wait for a second

}
```

Now lets us stage the demo1.c file, commit it, look at the status and perform a log2.

The result is here below.

We can see that the feature1 branch is closed and merged into the master branch.

Also HEAD is pointing to the master branch again.


Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)
Unmerged paths:
(use "git add <file>..." to mark resolution)
both modified: demo1.c
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git add demo1.c
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
All conflicts fixed but you are still merging.
(use "git commit" to conclude merge)
Changes to be committed:
modified: demo1.c
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git commit -m "v1.0 - Feature1 merged"
[master 5eff794] v1.0 - Feature1 merged
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git log2
* 5eff794 - (15 seconds ago) v1.0 - Feature1 merged - Your Name (HEAD -> master)
|\
| * 625f1ff - (72 minutes ago) v0.5 - Feature1 - blinkspeed as a variable - Your Name (feature1)
* | a4b3db0 - (51 minutes ago) v0.3 - Made blink faster - Your Name
|/
* a76b4bc - (2 hours ago) v0.2 – Demo1.c added #include <Arduino.h> - Your Name
* 2147b2c - (2 hours ago) v0.1 – Demo1.c changed with Arduino blink sketch - Your Name
* c75091e - (2 hours ago) Revert "v0.1 – Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - (3 hours ago) v0.1 – Demo1.c changed with Aduino blink sketch - Your Name
* af12b45 - (3 hours ago) v0.0 - Initial commit - Your Name
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>

9. Fork a repository.

10. The .gitignore file.

GIT has a file with a special purpose. This file is called .gitignore with no extensions.

The purpose of .gitignore files is to ensure that certain files not tracked by Git remain untracked.

To stop tracking a file that is currently tracked, use git rm --cached to remove the file from the index. The filename can then be added to the .gitignore file to stop the file from being reintroduced in later commits.

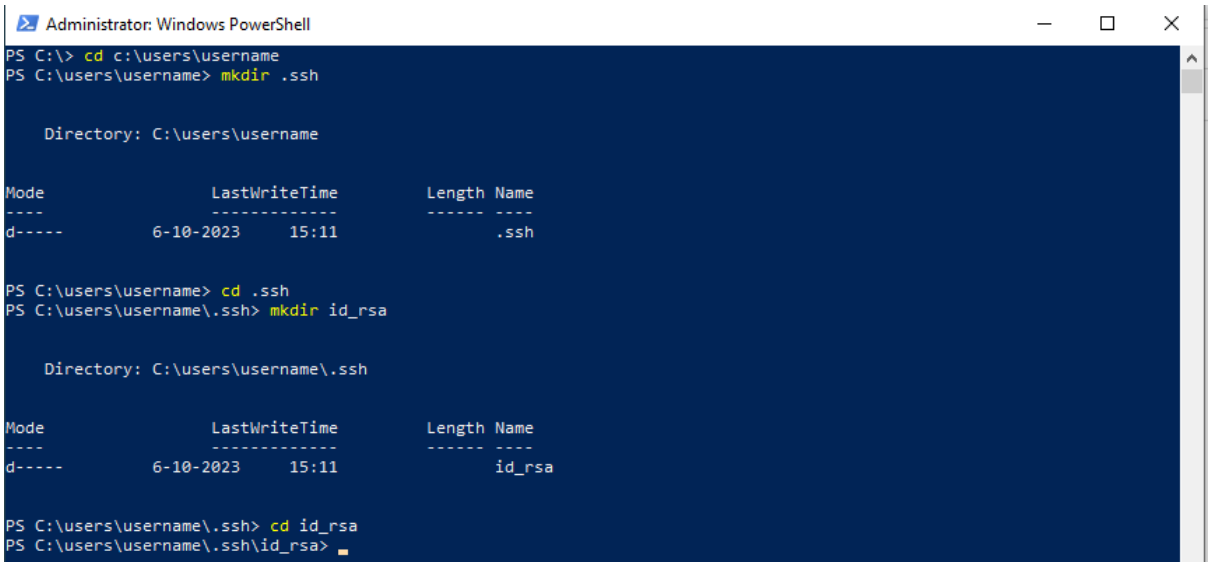
Git does not follow symbolic links when accessing a .gitignore file in the working tree. This keeps behavior consistent when the file is accessed from the index or a tree versus from the filesystem.

Why? Think of Wi-Fi or other credentials, but also other files that we do not want to be tracked.

11. SSH key.

Open PowerShell (PS) as an administrator and issue the following commands:
If no `.ssh/id_rsa` folder exists.

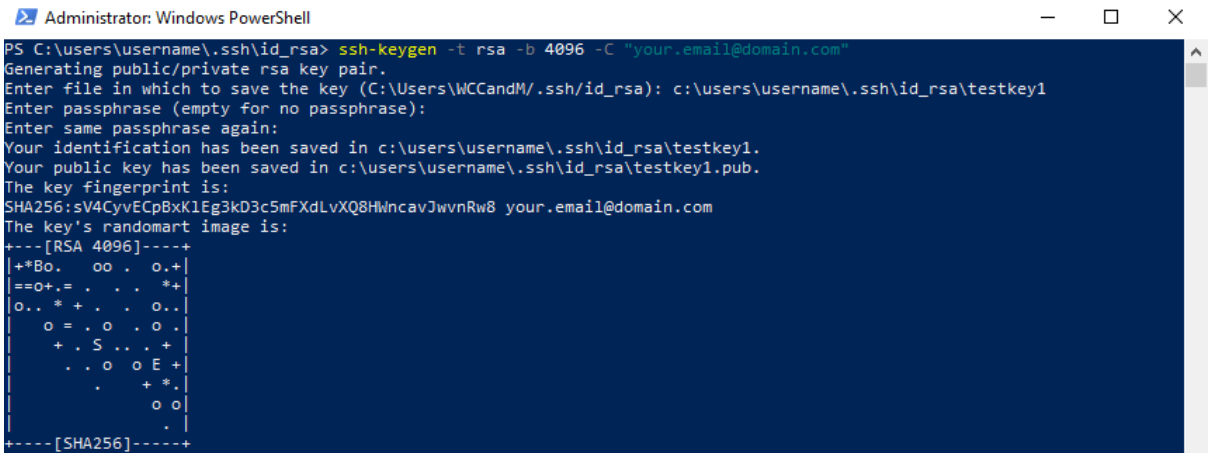
```
> cd c:\users\username
> mkdir .ssh
> cd .ssh
> mkdir id_rsh
```



If the folders .ssh and id_rsa exists continue with:

```
> ssh-keygen -t rsa -b 4096 -C "your.email@domain.com"
```

The email address to be used is the email address that you use to login to GitHub.
Enter the location: `c:\users\username\.ssh\id_rsa\testkey1`
Enter password: … (if any).
Wait for the message: key generated successfully.



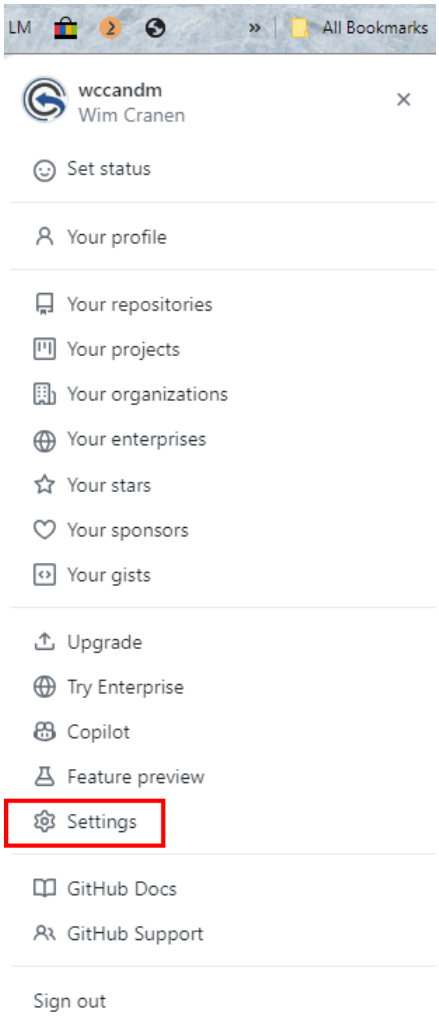
Open the folder you used to store the file in (location her above).
And you will find two files:

`testkey1` (your private key)
`testkey1.pub` (your public key)

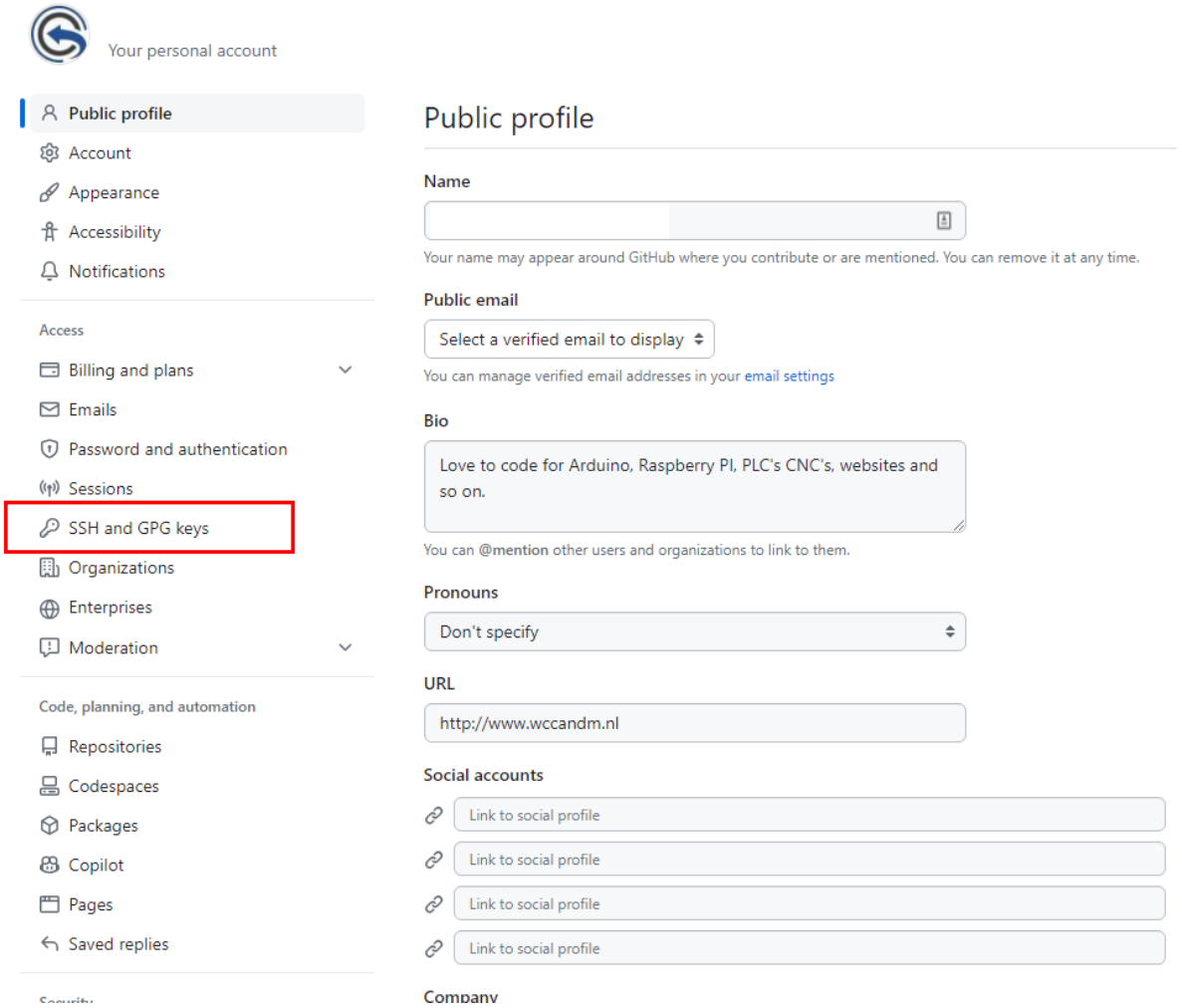
```
PS C:\users\username\.ssh\id_rsa> ls -force


    Directory: C:\users\username\.ssh\id_rsa


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6-10-2023     15:14           3389 testkey1
-a----         6-10-2023     15:14            748 testkey1.pub
```

Open testkey1.pub in notepad and switch word wrap on.
You can see that the content of the file starts with “ssh-rsa” and ends with your.email@domain.com

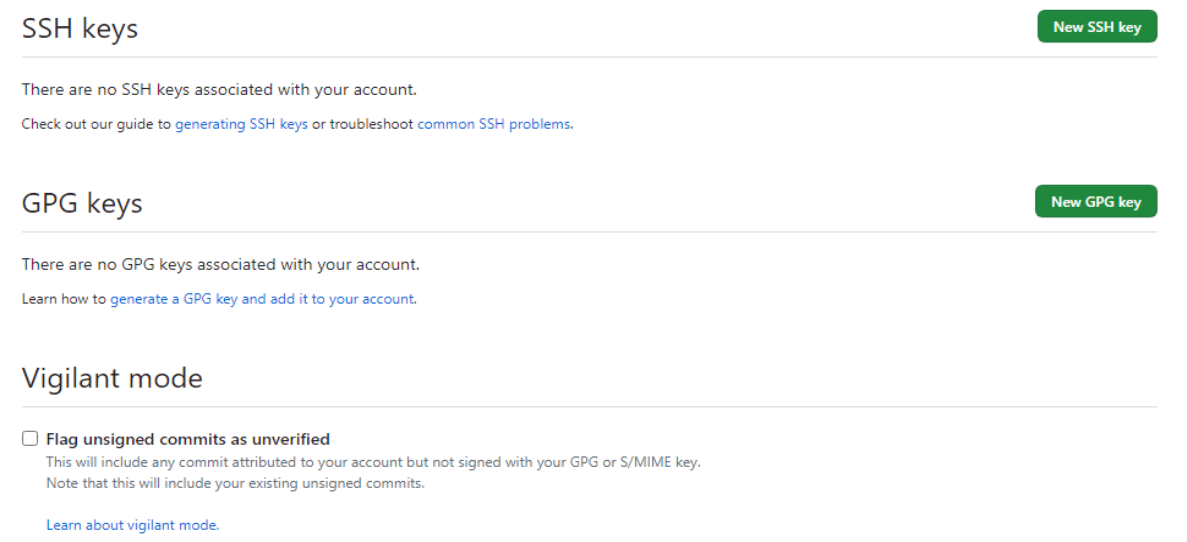
Copy this content to the clipboard.
Login to GitHub.
Go to your logo in the upper right corner of the opening screen.



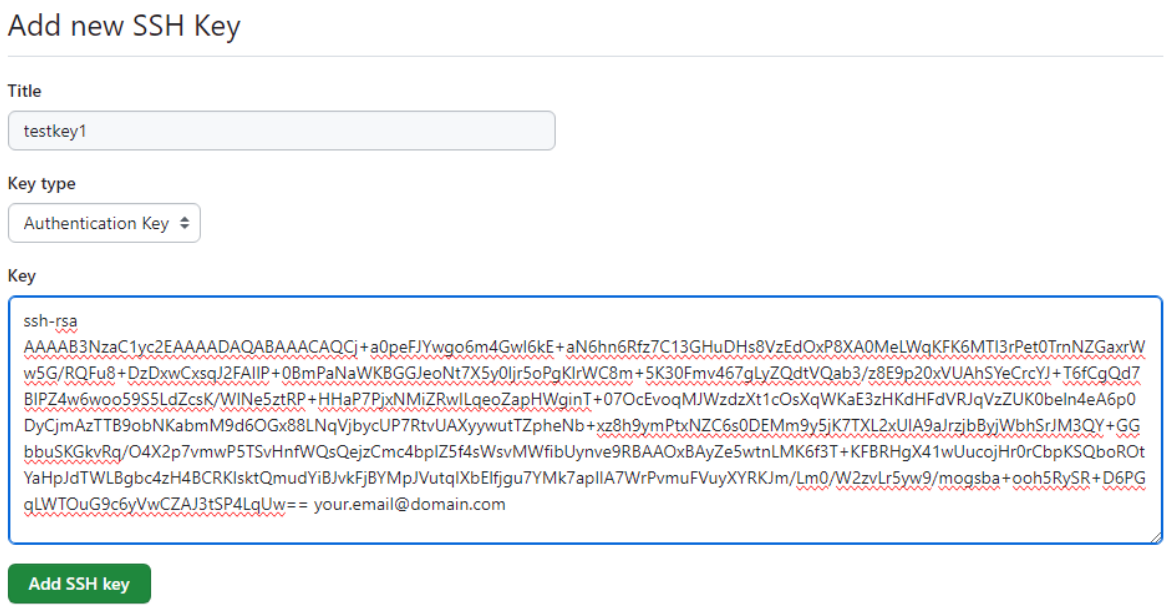
In the new window go to “SSH and GPG keys” in the left.



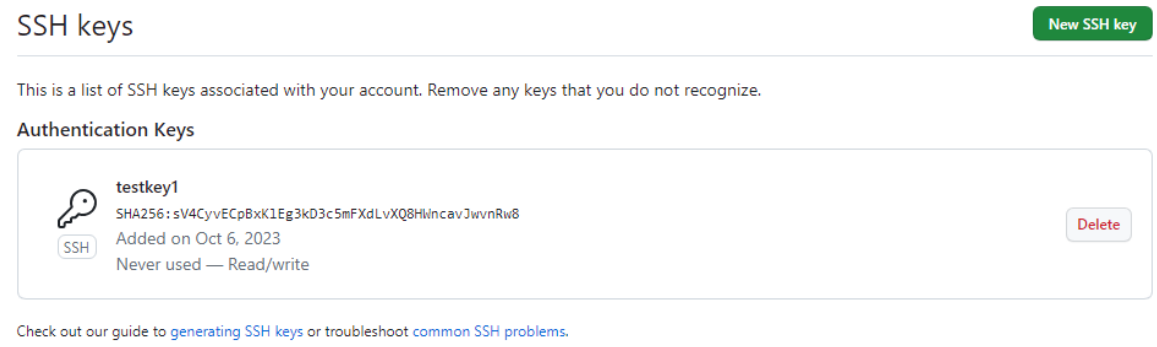
Here you can see your SSH keys, if any exists. Click “New Key”



Under title, give it an name (testkey1) and paste the content of your public key in the “key” windows. Remove the last space.



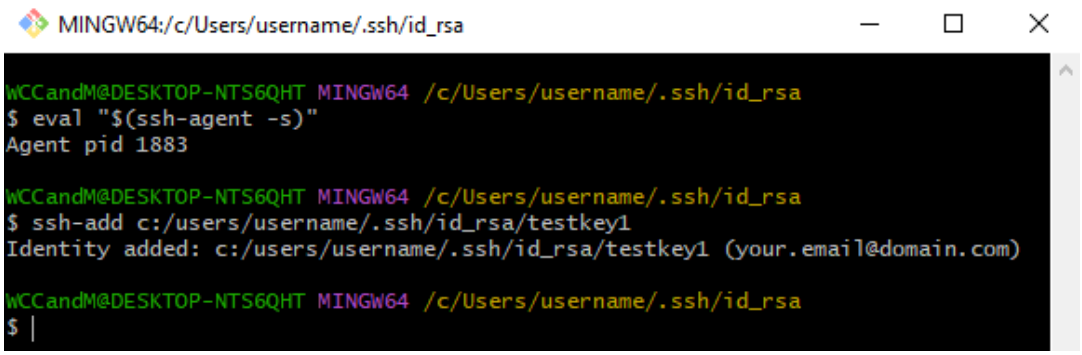
Click the “Add SSH key” button (you are required to enter your password and confirm). Done!!.



If you want to know more about the SSH keys, click the instruction to check out the guides.

Now you have to make sure that the local GIT knows about your private key. Thes keys match together and the pair is the mathematical prove that it is generated with the public key information.

Start the SSH-agent in the background (if you use Windows, do this in GIT bash).



GIT bash needs forward slashes!!

See also https://www.youtube.com/watch?v=RGOj5yH7evk&t=1230

12. Remote repositories.

Why would we want to have a remote repository?

1. To make it possible for a team to work on a project.
2. To have backups. Push regularly (at least every change) to the remote repository and pull the changes from your teammates.

This makes sure you are working with a clean project and maybe overcome merge conflicts.

Where do we want to create a remote repository?

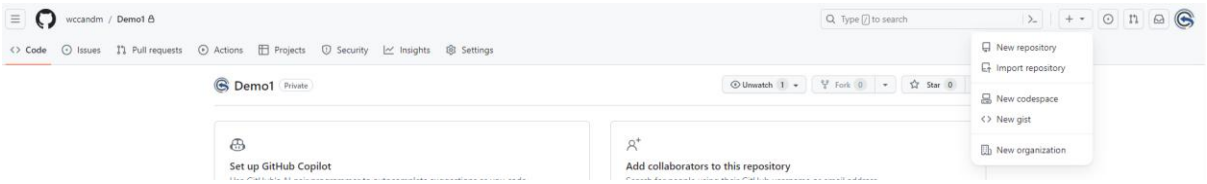
There is more than one solution:

1. On GitHub.
2. On a local server.
3. On a remote server

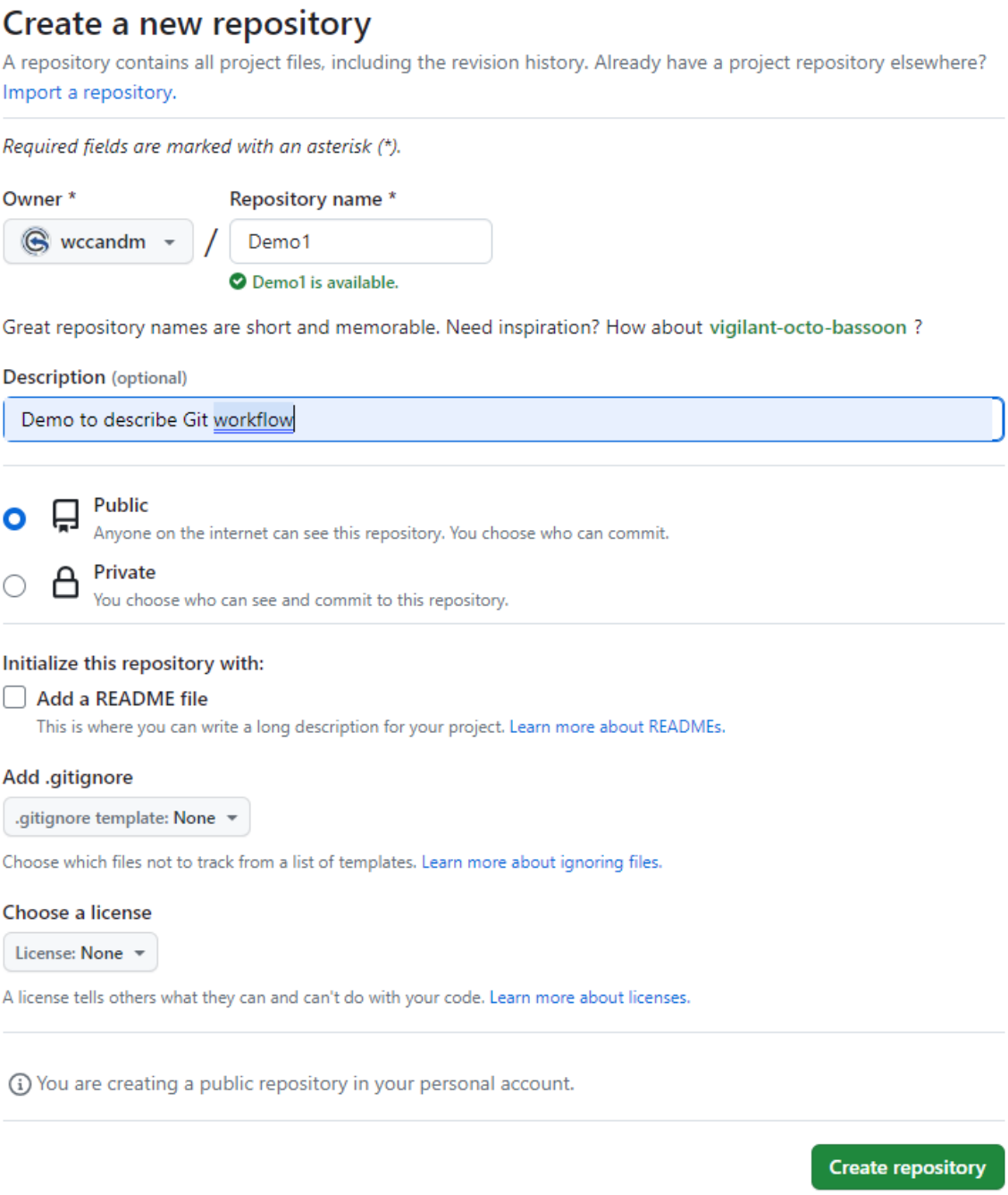
13. Remote repository on GitHub.

Open your GitHub account or create on if there is no account.

Create a new repository with the same name as your local repository.



Goto the “+” sign and click “New Repository”.

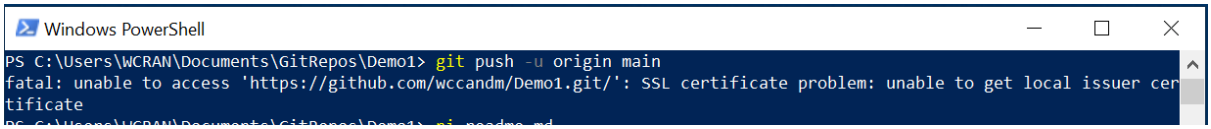


And click “Create Repository”.

If we issue the following sequence:

```
git remote add origin https://github.com/wccandm/Demo1.git
git branch -M main
git push -u origin main
```

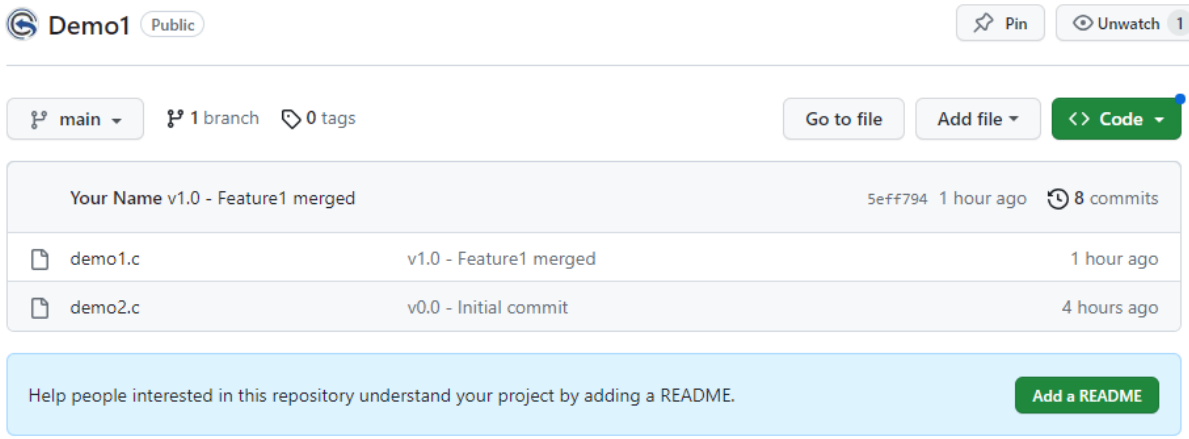
We get an error:



We have an issue with the SSL certificate. Issue the following command and try again:

```
git config --global http.sslbackend schannel
```

You have to authenticate in your browser. Do this and…

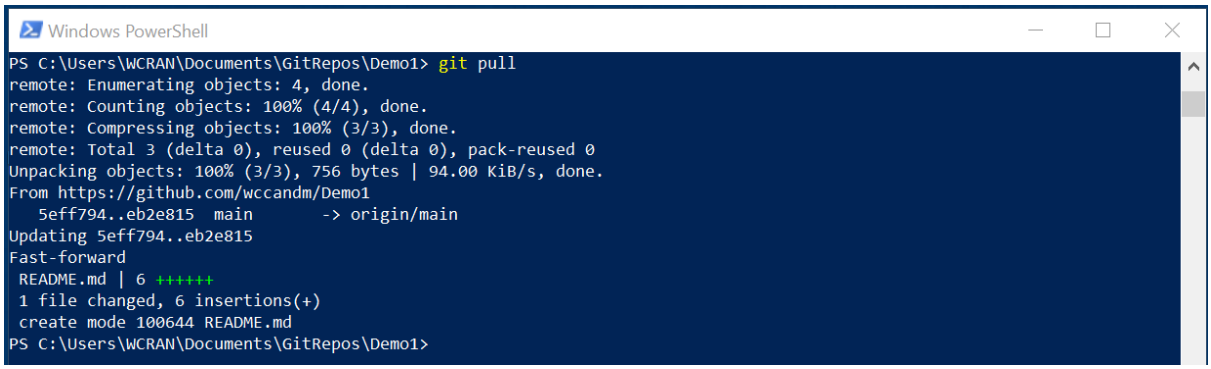


Your repository exists on GitHub.

Let us create the README.md file in GitHub and pull the project to the local repository.

While the remote repository is known, we just have to issue this command:

```
git pull
```



Perform `git log2` command:

You can see that a README.md file was created on the origin/main (=GitHub).
While the README.md file is in our local repository a pull must have been performed.

```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git log2
* eb2e815 - (6 minutes ago) Created README.md file - Wim Cranen (HEAD -> main, origin/main)
*   5eff794 - (2 hours ago) v1.0 - Feature1 merged - Your Name
|\
| * 625f1ff - (3 hours ago) v0.5 - Feature1 - blinkspeed as a variable - Your Name (feature1)
* | a4b3db0 - (2 hours ago) v0.3 - Made blink faster - Your Name
|/
* a76b4bc - (4 hours ago) v0.2 – Demo1.c added #include <Arduino.h> - Your Name
* 2147b2c - (4 hours ago) v0.1 – Demo1.c changed with Arduino blink sketch - Your Name
* c75991e - (4 hours ago) Revert "v0.1 – Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - (4 hours ago) v0.1 – Demo1.c changed with Aduino blink sketch - Your Name
* af12b45 - (5 hours ago) v0.0 - Initial commit - Your Name
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

Now let is change demo1.c in our local repository, change the blinking speed to 500.
Stage and commit the file.

```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo1.c

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git add demo1.c
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demo1.c

PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git commit -m "v1.01 - Changed blink speed to 500 ms"
[main 961720e] v1.01 - Changed blink speed to 500 ms
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

See that the master branch name changed to main.
Perform a `git log2`.

```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git log2
* 961720e - (2 minutes ago) v1.01 - Changed blink speed to 500 ms - Your Name (HEAD -> main)
* eb2e815 - (14 minutes ago) Created README.md file - Wim Cranen (origin/main)
*   5eff794 - (2 hours ago) v1.0 - Feature1 merged - Your Name
|\
| * 625f1ff - (3 hours ago) v0.5 - Feature1 - blinkspeed as a variable - Your Name (feature1)
* | a4b3db0 - (2 hours ago) v0.3 - Made blink faster - Your Name
|/
* a76b4bc - (4 hours ago) v0.2 – Demo1.c added #include <Arduino.h> - Your Name
* 2147b2c - (4 hours ago) v0.1 – Demo1.c changed with Arduino blink sketch - Your Name
* c75991e - (4 hours ago) Revert "v0.1 – Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - (4 hours ago) v0.1 – Demo1.c changed with Aduino blink sketch - Your Name
* af12b45 - (5 hours ago) v0.0 - Initial commit - Your Name
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

Let us open the demo1.c file on GitHub.

Code | Blame | 42 lines (32 loc) · 1.3 KB | Code 55% faster with GitHub Copilot

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

#include <Arduino.h>

int iBlinkSpeed = 1000;             // the blink speed in milli seconds

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(iBlinkSpeed);               // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(iBlinkSpeed);               // wait for a second

}
```

```
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 358 bytes | 358.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/wccandm/Demo1.git
   eb2e815..961720e  main -> main
```

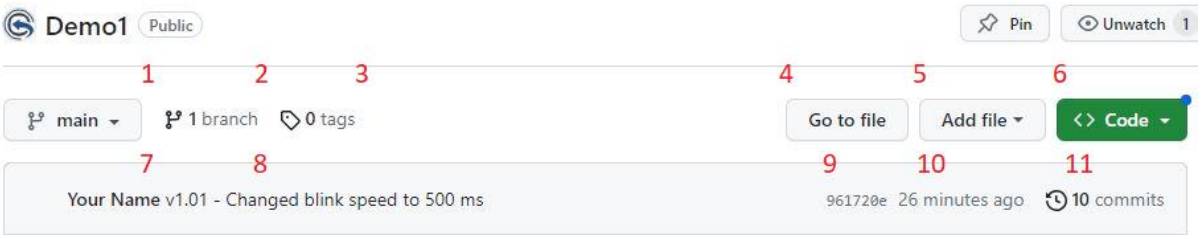
Then push the local repository to GitHub and again open the demo1.c file on GitHub.

Demo1 / demo1.c

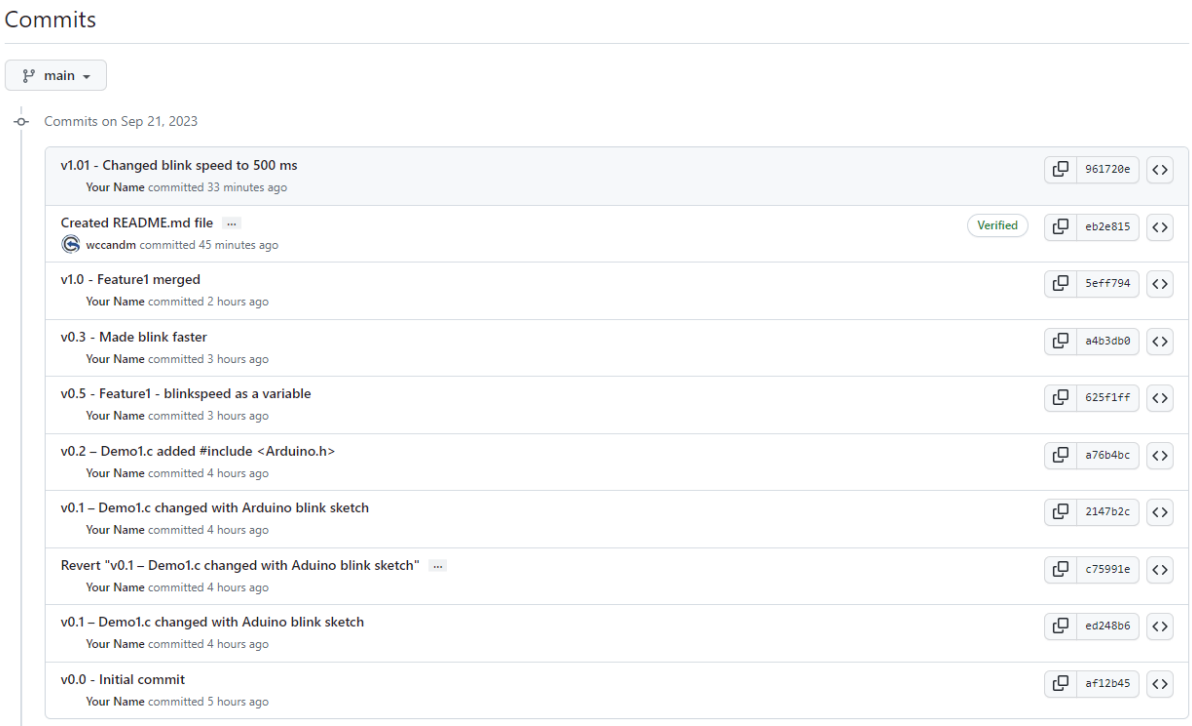
Code | Blame | 42 lines (32 loc) · 1.3 KB | Code 55% faster with GitHub Copilot

```
3
4       Turns an LED on for one second, then off for one second, repeatedly.
5
6       Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7       it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8       the correct LED pin independent of which board is used.
9       If you want to know what pin the on-board LED is connected to on your Arduino
10      model, check the Technical Specs of your board at:
11      https://www.arduino.cc/en/Main/Products
12
13      modified 8 May 2014
14      by Scott Fitzgerald
15      modified 2 Sep 2016
16      by Arturo Guadalupi
17      modified 8 Sep 2016
18      by Colby Newman
19
20      This example code is in the public domain.
21
22      https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23    */
24
25    #include <Arduino.h>
26
27    int iBlinkSpeed = 500;              // the blink speed in milli seconds
28
29    // the setup function runs once when you press reset or power the board
30    void setup() {
31      // initialize digital pin LED_BUILTIN as an output.
32      pinMode(LED_BUILTIN, OUTPUT);
33    }
34
35    // the loop function runs over and over again forever
36    void loop() {
37      digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
38      delay(iBlinkSpeed);               // wait for a second
39      digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
40      delay(iBlinkSpeed);               // wait for a second
41
42    }
```

And see, blink speed has changed.
What can we see more on GitHub? See the repository head line:



1. The existing branches
2. The default branch
3. The releasees and the tags
4. A got file button
5. An add file button
6. A code button for cloning
7. Your Name
8. Changes made in this commit
9. Changes made in this commit
10. Changes made in this commit
11. All commits



14. Remote repository on a local server.

Let us create a local Git Repository on a local server. Use this tutorial or this one.

Or install the Bonobo Git Server for Windows.

Or install GIT this way.

I will choose for the last one, and follow the instructions in the installation documentation.

15. VS Code Git Status labels.

A	Added	This is a new file that has been added to the repository
M	Modified	An existing file has been changed
D	Deleted	A file has been deleted
U	Untracked	The file is new or has been changed but has not been added to the staging area
C	Conflict	There is a conflict in the file
R	Renamed	The file has been renamed