

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

Git Workflow from installation to repository	1
Inhoud	1
1. What is GIT, and why GIT?	2
2. Installation of GIT.	3
3. GIT initialize.	4
4. Your first (local) repository.	6
5. Create files.	7
6. Change and track changes.	10
7. Create a new branch.	14
8. Merge branches.	17
9. Fork a repository.	21
10. The .gitignore file	22
11. SSH key	23
12. Remote repositories.	28
13. Remote repository on GitHub.	29
14. Remote repository on a local server.	35
15. VS Code Git Status labels.	36

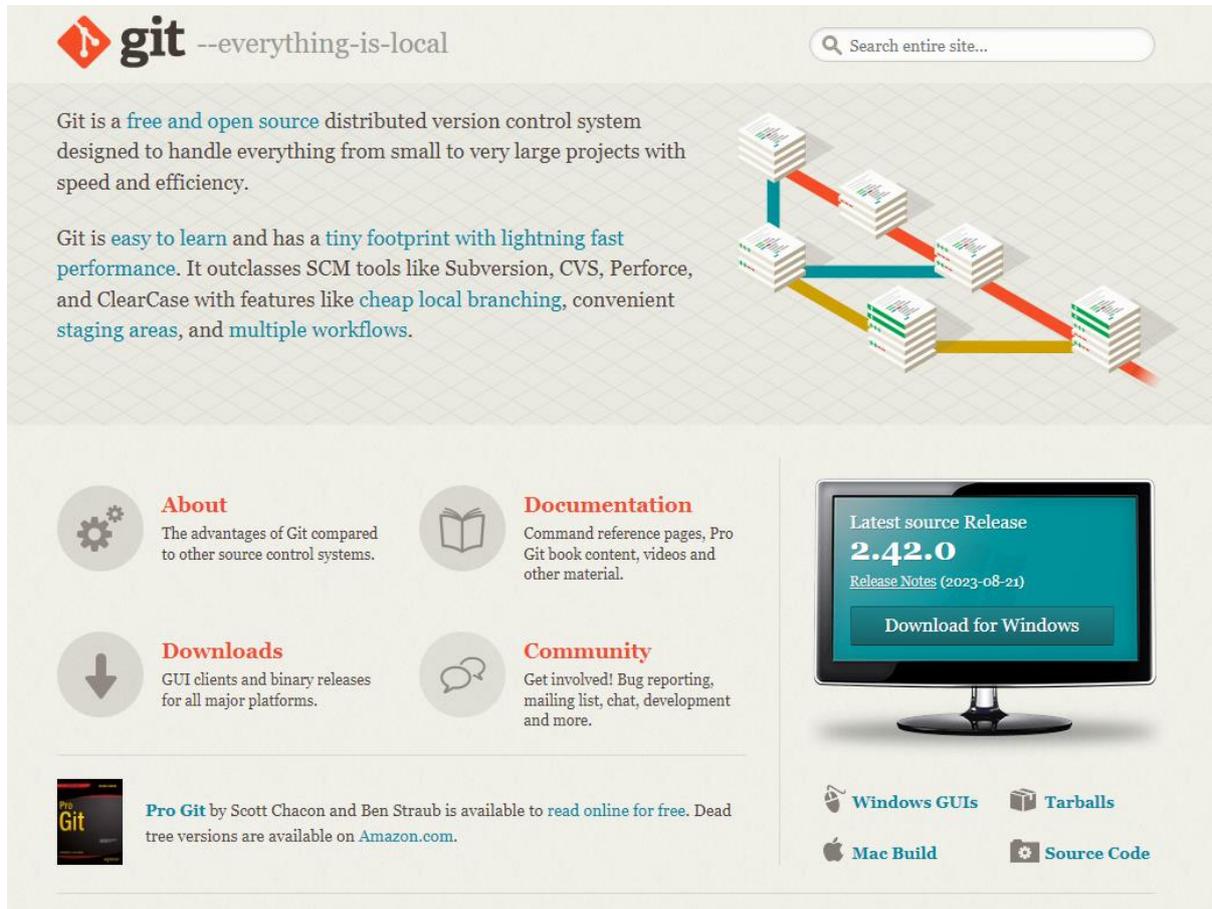
1. What is GIT, and why GIT?

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.



The screenshot shows the Git website homepage. At the top left is the Git logo with the tagline "--everything-is-local". A search bar is located at the top right. The main content area features a description of Git as a free and open source distributed version control system, followed by a diagram illustrating a branching model with stacks of code and colored lines representing branches. Below this, there are four sections: "About" (advantages compared to other systems), "Documentation" (command reference, Pro Git book, videos), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list, chat). On the right side, a monitor displays the latest source release "2.42.0" with a "Download for Windows" button. At the bottom, there are links for "Windows GUIs", "Tarballs", "Mac Build", and "Source Code".

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).

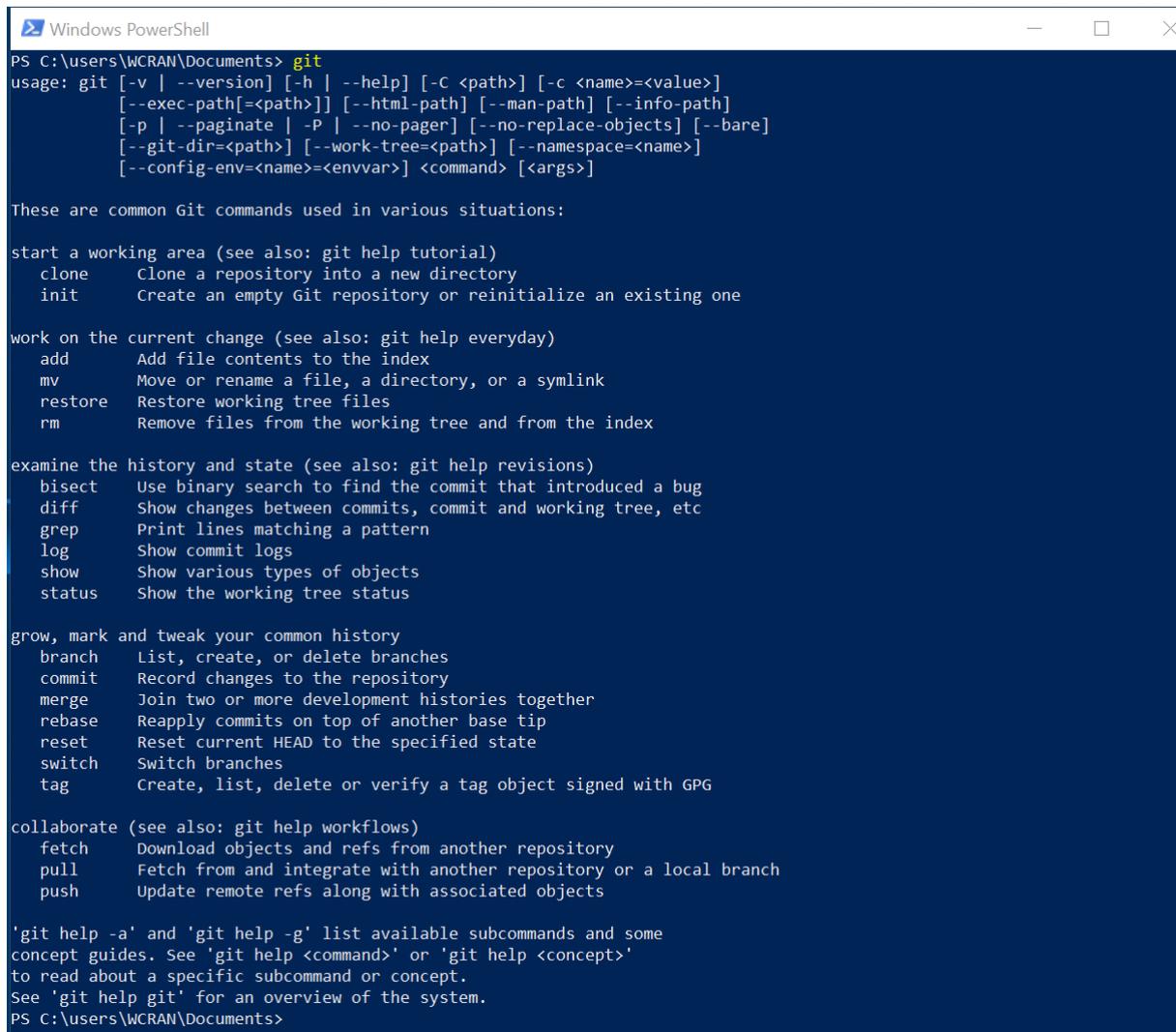
3. GIT initialize.

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.



```
Windows PowerShell
PS C:\users\WCRAN\Documents> git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
  [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
  [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
  [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
  [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

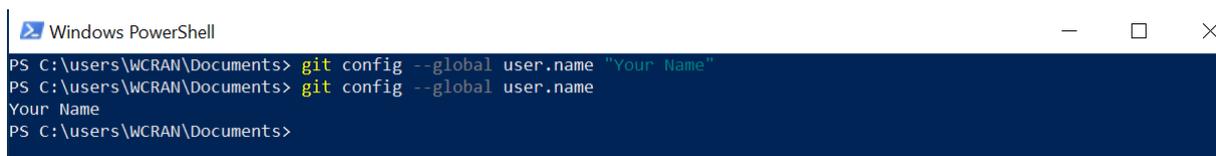
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
PS C:\users\WCRAN\Documents>
```

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
```



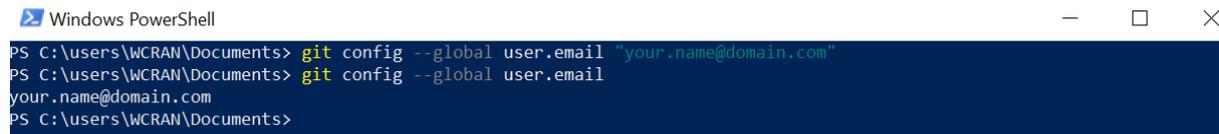
```
Windows PowerShell
PS C:\users\WCRAN\Documents> git config --global user.name "Your Name"
PS C:\users\WCRAN\Documents> git config --global user.name
Your Name
PS C:\users\WCRAN\Documents>
```

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:

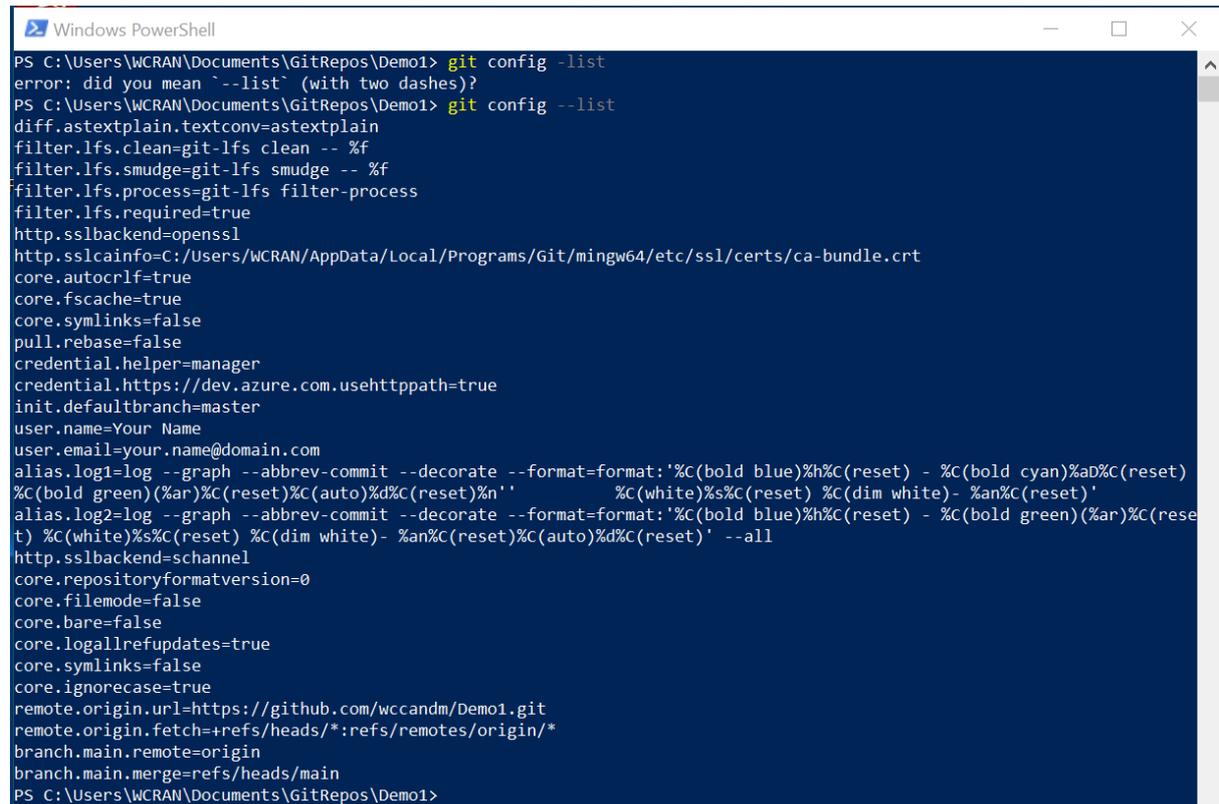
```
git config --global user.email
```



```
Windows PowerShell
PS C:\users\WCRAN\Documents> git config --global user.email "your.name@domain.com"
PS C:\users\WCRAN\Documents> git config --global user.email
your.name@domain.com
PS C:\users\WCRAN\Documents>
```

Now your git environment is ready for operation.

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



```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git config -list
error: did you mean `--list` (with two dashes)?
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Users/WCRAN/AppData/Local/Programs/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fsckcache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Your Name
user.email=your.name@domain.com
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)'
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
http.sslbackend=schannel
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/wccandm/Demo1.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

4. Your first (local) repository.

In the documents folder create a new folder with the name "GitRepos" change to that folder and in the GitRepos folder create a new folder with the name "Demo1" and change into that folder with power shell.

```
Windows PowerShell
PS C:\users\WCRAN\Documents> cd .\GitRepos\
PS C:\users\WCRAN\Documents\GitRepos> cd .\Demo1\
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

Enter the command: `git status`.

```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
fatal: not a git repository (or any of the parent directories): .git
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

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.

```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
fatal: not a git repository (or any of the parent directories): .git
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git init
Initialized empty Git repository in C:/Users/WCRAN/Documents/GitRepos/Demo1/.git/
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

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".

```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> ls
PS C:\users\WCRAN\Documents\GitRepos\Demo1> ls -h

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

Mode                LastWriteTime         Length Name
----                -
d--h--             21-9-2023   09:13         .git

PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

This .git folder contains all information that git saves.

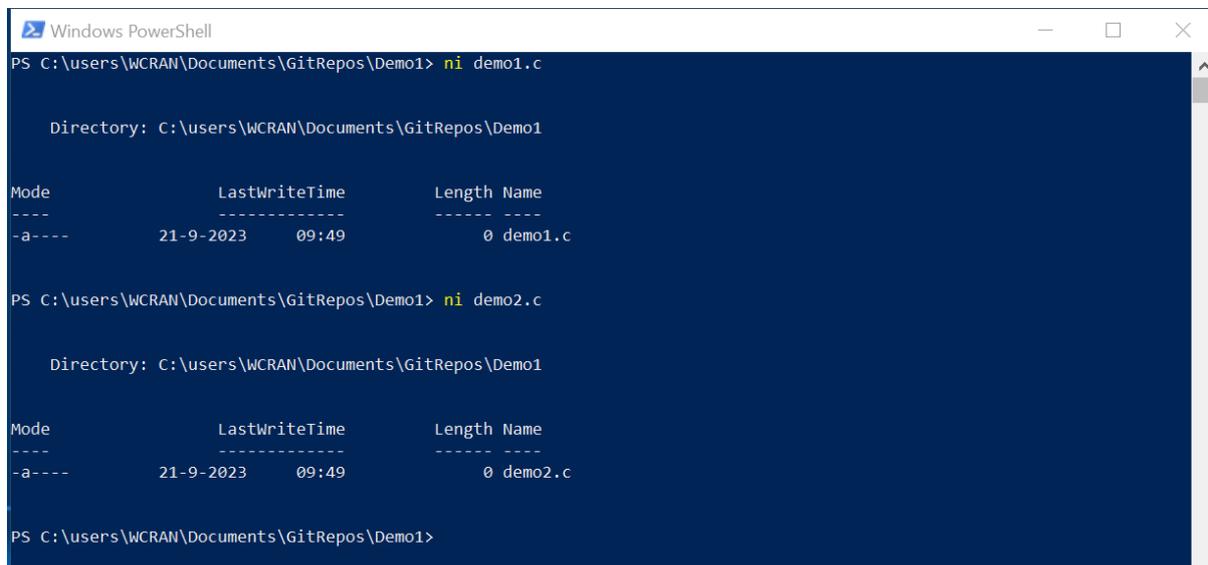
If you want to get rid of the git information, just give the command `rm .git` and answer with "Y".

5. Create files.

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

```
ni demo1.c
```

```
ni demo2.c
```



```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> ni demo1.c

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

Mode                LastWriteTime         Length Name
----                -
-a----             21-9-2023    09:49             0 demo1.c

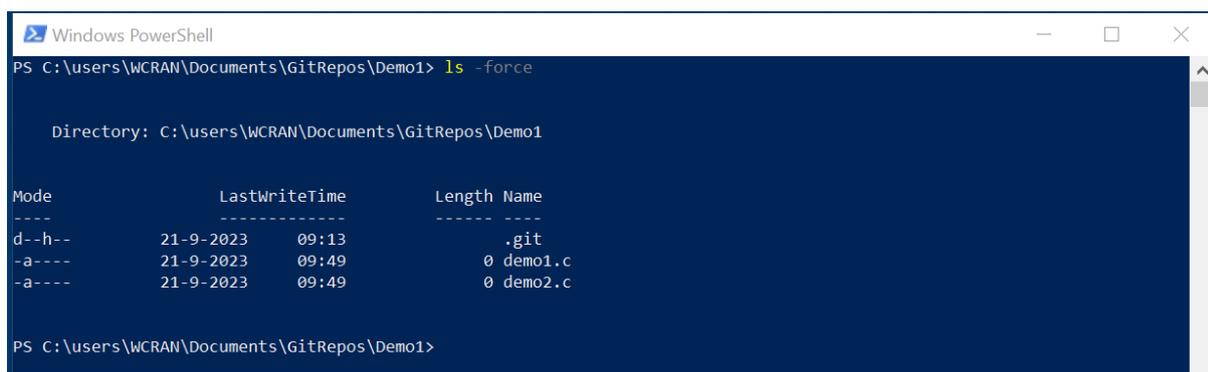
PS C:\users\WCRAN\Documents\GitRepos\Demo1> ni demo2.c

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

Mode                LastWriteTime         Length Name
----                -
-a----             21-9-2023    09:49             0 demo2.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

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



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

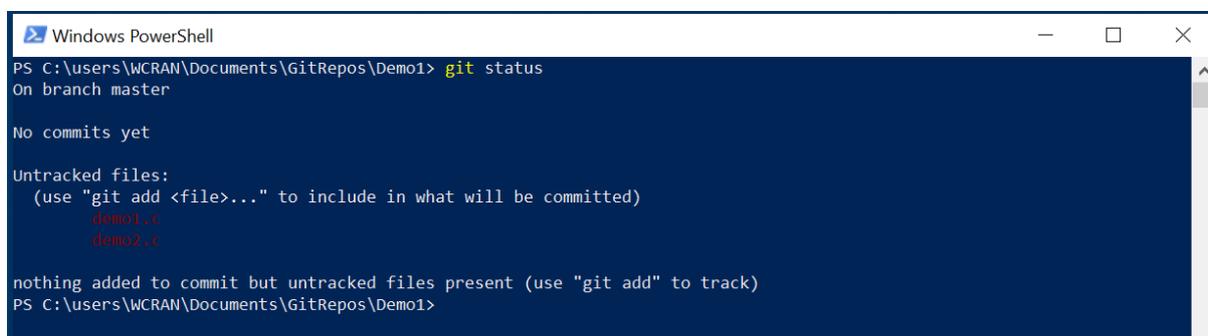
Directory: C:\users\WCRAN\Documents\GitRepos\Demo1

Mode                LastWriteTime         Length Name
----                -
d--h--             21-9-2023    09:13             .git
-a----             21-9-2023    09:49             0 demo1.c
-a----             21-9-2023    09:49             0 demo2.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

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`.



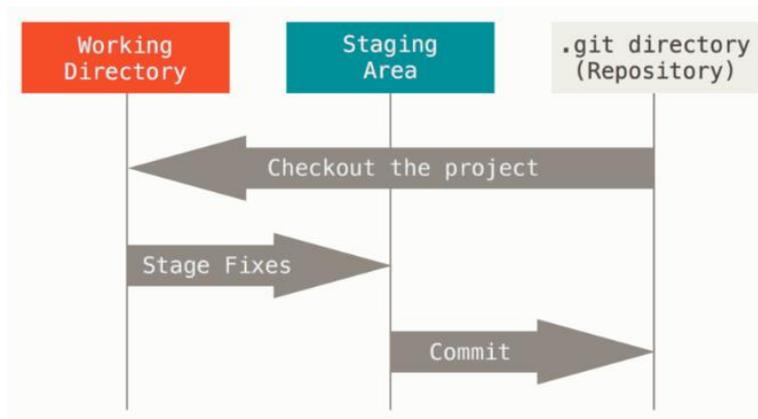
```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    demo1.c
    demo2.c

nothing added to commit but untracked files present (use "git add" to track)
PS C:\users\WCRAN\Documents\GitRepos\Demo1>
```

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 they 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.

```

Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git add .
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   demo1.c
        new file:   demo2.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1>

```

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.

```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git commit -m "v0.0 - Initial commit"
[master (root-commit) af12b45] v0.0 - Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo1.c
 create mode 100644 demo2.c
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 af12b458e50f0200938c3f3ed9225bdd017db248 (HEAD -> master)
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>
```

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 x
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  // the setup function runs once when you press reset or power the board
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34    delay(1000); // wait for a second
35    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36    delay(1000); // wait for a second
37  }
```

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.

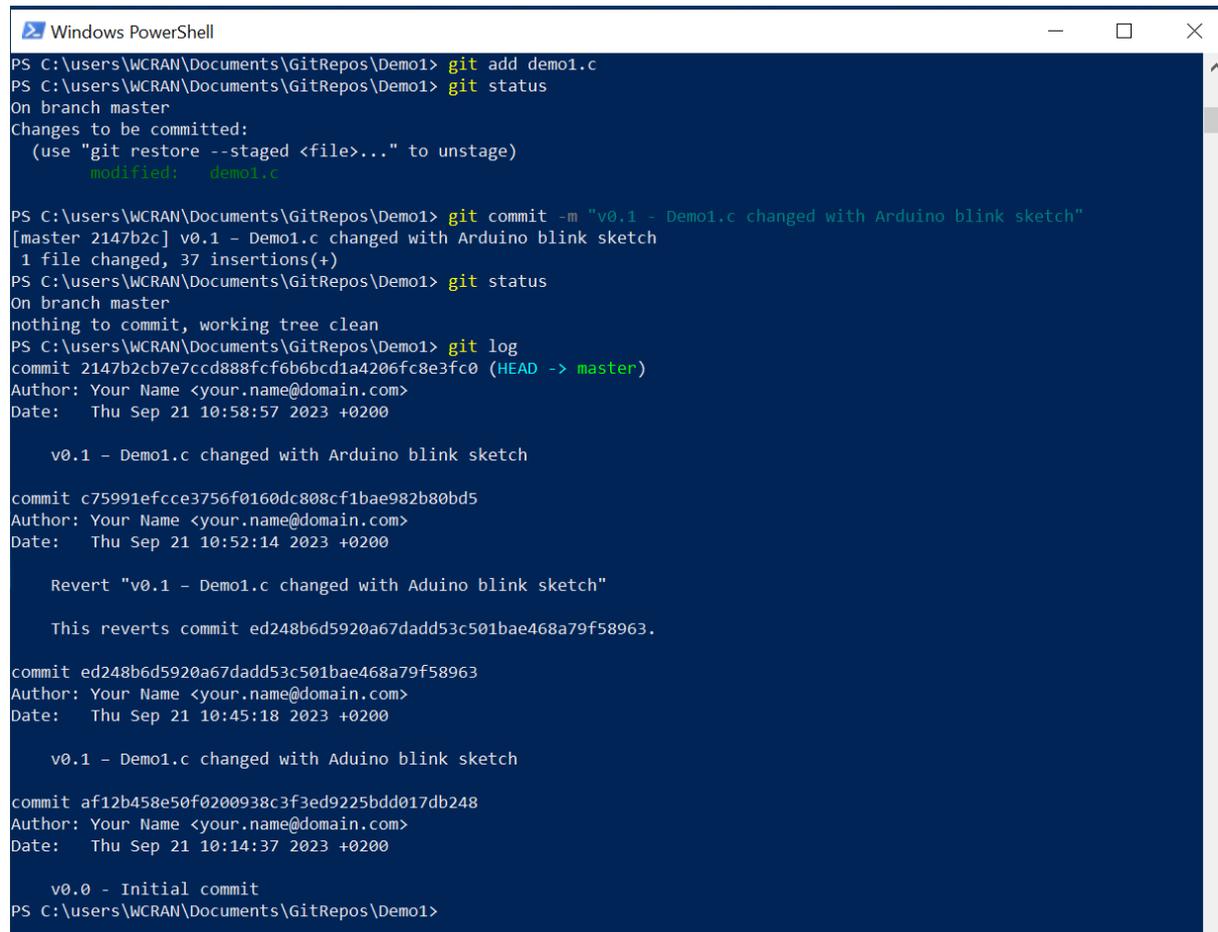
```
git add demo1.c
```

```
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`.

```

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    10:59             .git
-a----             21-9-2023    11:03           1299 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 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

```

```

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 "v0.2 - Demo1.c added #include <Arduino.h>"
[master a76b4bc] v0.2 - Demo1.c added #include <Arduino.h>
 1 file changed, 2 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 a76b4bc091d7198055fcadd96e2958250b3aa114 (HEAD -> 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 c75991efc3756f0160dc808cf1bae982b80bd5
Author: Your Name <your.name@domain.com>
Date: Thu Sep 21 10:52:14 2023 +0200

```

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"
```

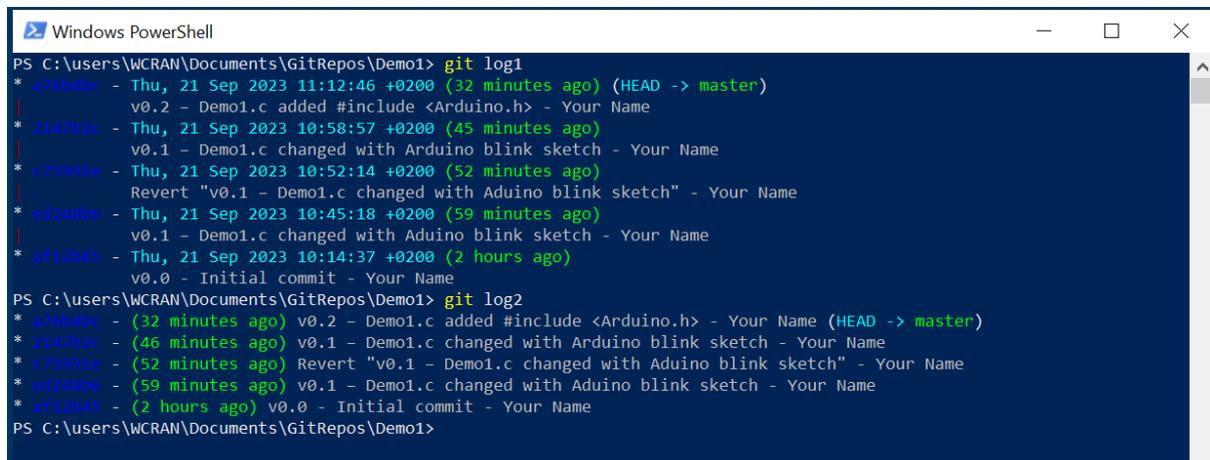
The you can give the commands:

```
git log1
```

Or

```
git log2
```

And that gives the following results:



```
Windows PowerShell
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log1
* a76b4bc - Thu, 21 Sep 2023 11:12:46 +0200 (32 minutes ago) (HEAD -> master)
  v0.2 - Demo1.c added #include <Arduino.h> - Your Name
* 2147b2c - Thu, 21 Sep 2023 10:58:57 +0200 (45 minutes ago)
  v0.1 - Demo1.c changed with Arduino blink sketch - Your Name
* c75991e - Thu, 21 Sep 2023 10:52:14 +0200 (52 minutes ago)
  Revert "v0.1 - Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - Thu, 21 Sep 2023 10:45:18 +0200 (59 minutes ago)
  v0.1 - Demo1.c changed with Aduino blink sketch - Your Name
* af12b45 - Thu, 21 Sep 2023 10:14:37 +0200 (2 hours ago)
  v0.0 - Initial commit - Your Name
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log2
* a76b4bc - (32 minutes ago) v0.2 - Demo1.c added #include <Arduino.h> - Your Name (HEAD -> master)
* 2147b2c - (46 minutes ago) v0.1 - Demo1.c changed with Arduino blink sketch - Your Name
* c75991e - (52 minutes ago) Revert "v0.1 - Demo1.c changed with Aduino blink sketch" - Your Name
* ed248b6 - (59 minutes 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>
```

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.

```
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   11:51             .git
-a----             21-9-2023   11:56          1388 demo1.c
-a----             21-9-2023    09:49           0 demo2.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch feature1
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`

`git status`

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

`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 feature1
Changes to be committed:
  (use "git restore --staged <file>.." to unstage)
        modified:   demo1.c

PS C:\users\WCRAN\Documents\GitRepos\Demo1> git commit -m "v0.5 - Feature1 - blinkspeed as a variable"
[feature1 625f1ff] v0.5 - Feature1 - blinkspeed as a variable
 1 file changed, 4 insertions(+), 2 deletions(-)
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git status
On branch feature1
nothing to commit, working tree clean
PS C:\users\WCRAN\Documents\GitRepos\Demo1> git log
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
```

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>
```

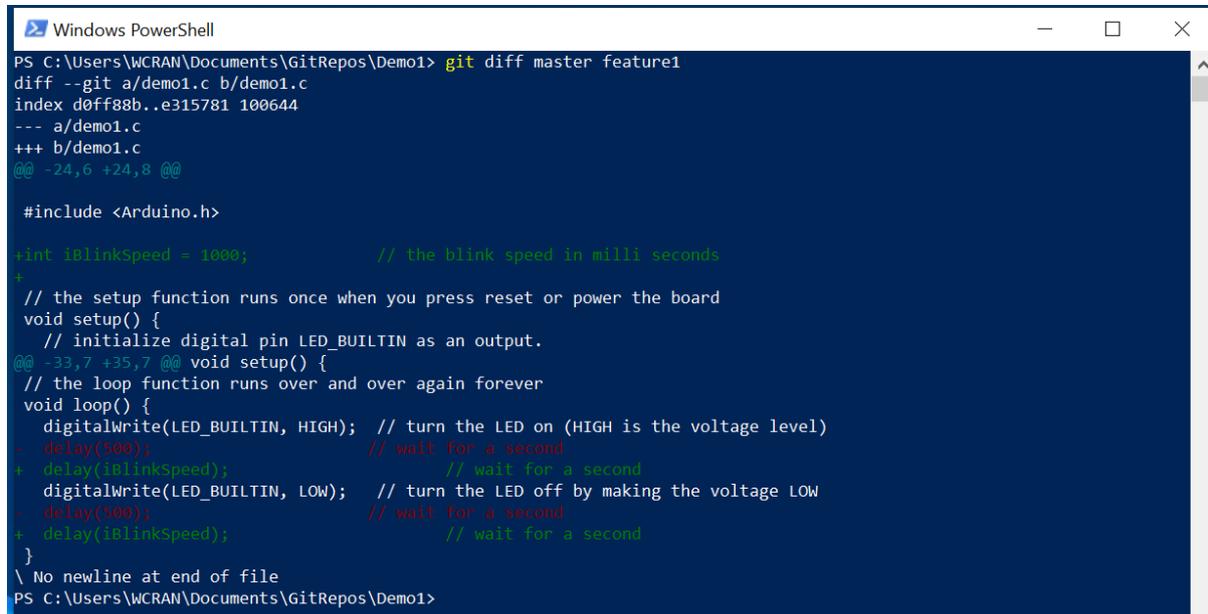
Now you can see the branch feature1 going away from the master branch.

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
```

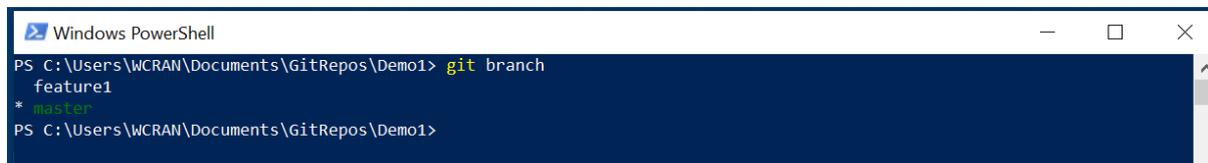


```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git diff master feature1
diff --git a/demo1.c b/demo1.c
index d0ff88b..e315781 100644
--- a/demo1.c
+++ b/demo1.c
@@ -24,6 +24,8 @@

#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.
@@ -33,7 +35,7 @@ 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)
-  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(500);                     // wait for a second
+  delay(iBlinkSpeed);            // wait for a second
}
\ No newline at end of file
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

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

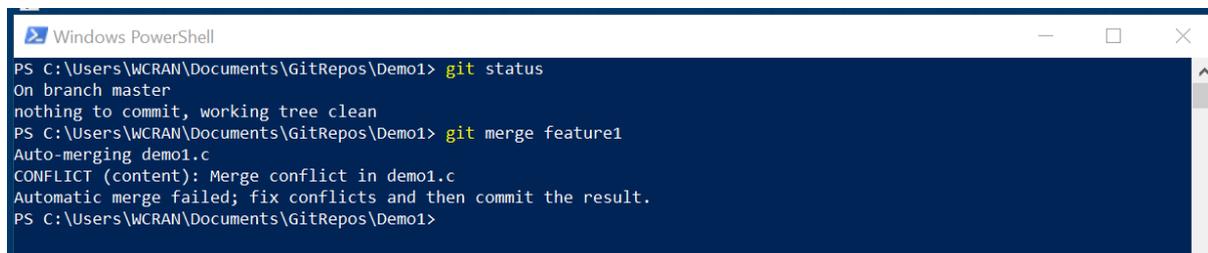


```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git branch
  feature1
* master
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

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
```



```
Windows PowerShell
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 merge feature1
Auto-merging demo1.c
CONFLICT (content): Merge conflict in demo1.c
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

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

Now the `git diff` command can help us.

This would be ok:

```
demo1.c x
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  }
42
```

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
| * c75991e - (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.

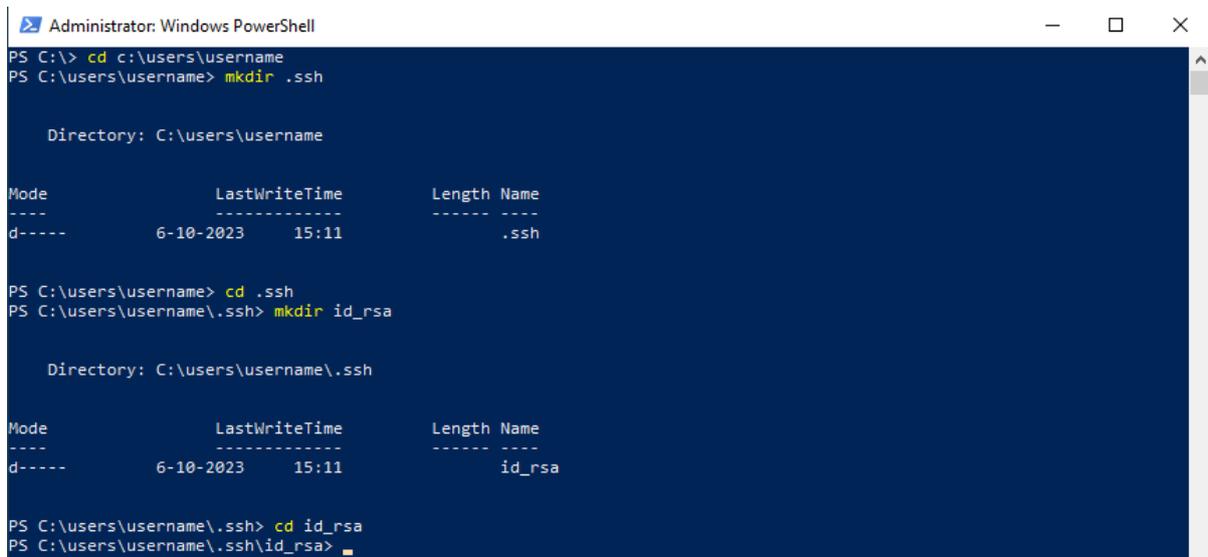
Why? Think of Wi-Fi or other credentials, but also other files that we do not want to be tracked.

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_rsa
```



```
Administrator: Windows PowerShell
PS C:\> cd c:\users\username
PS C:\users\username> mkdir .ssh

Directory: C:\users\username

Mode                LastWriteTime         Length Name
----                -
d-----            6-10-2023   15:11         .ssh

PS C:\users\username> cd .ssh
PS C:\users\username\.ssh> mkdir id_rsa

Directory: C:\users\username\.ssh

Mode                LastWriteTime         Length Name
----                -
d-----            6-10-2023   15:11         id_rsa

PS C:\users\username\.ssh> cd id_rsa
PS C:\users\username\.ssh\id_rsa>
```

If the folders `.ssh` and `id_rsa` exists continue with:

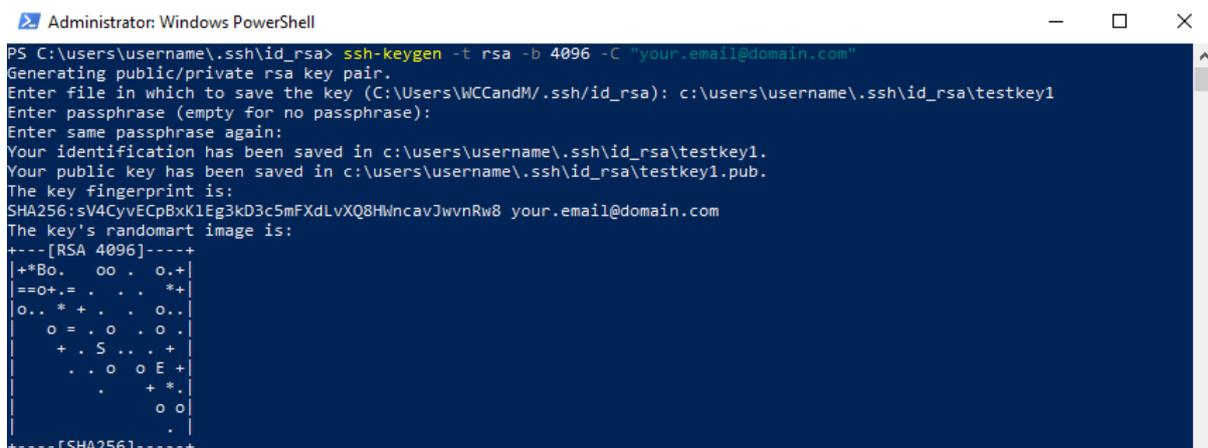
```
> ssh-keygen -t rsa -b 4096 -C "your.email@domain.com"
```

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.



```
Administrator: Windows PowerShell
PS C:\users\username\.ssh\id_rsa> ssh-keygen -t rsa -b 4096 -C "your.email@domain.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\WCCandM\.ssh\id_rsa): c:\users\username\.ssh\id_rsa\testkey1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in c:\users\username\.ssh\id_rsa\testkey1.
Your public key has been saved in c:\users\username\.ssh\id_rsa\testkey1.pub.
The key fingerprint is:
SHA256:sV4CyvECpBxK1Eg3kD3c5mFXdLvXQ8HwncavJwvnRw8 your.email@domain.com
The key's randomart image is:
+---[RSA 4096]-----+
|+Bo.  oo . o.+|
|==O+= . . . *+|
|O.. * + . . o..|
| o = . o . o .|
| + . S . . . +|
| . . o o E +|
| . + *|
| o o|
| .|
+----[SHA256]-----+
```

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

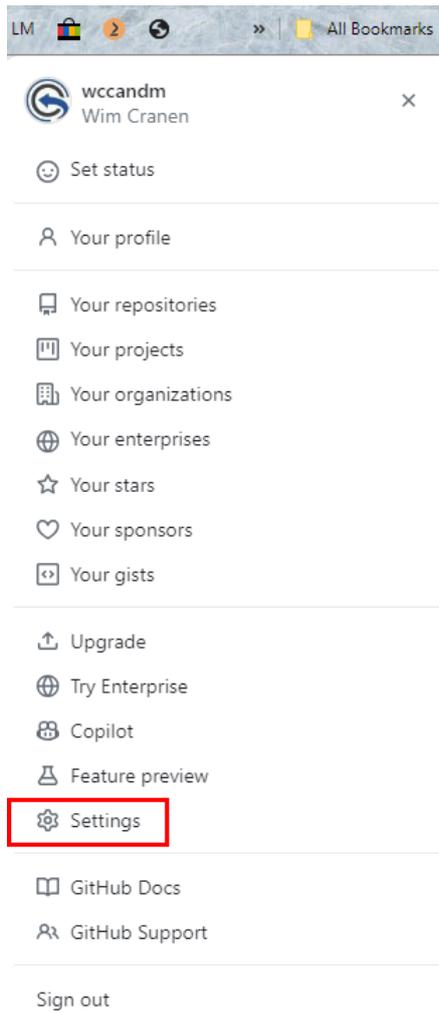
Directory: C:\users\username\.ssh\id_rsa

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

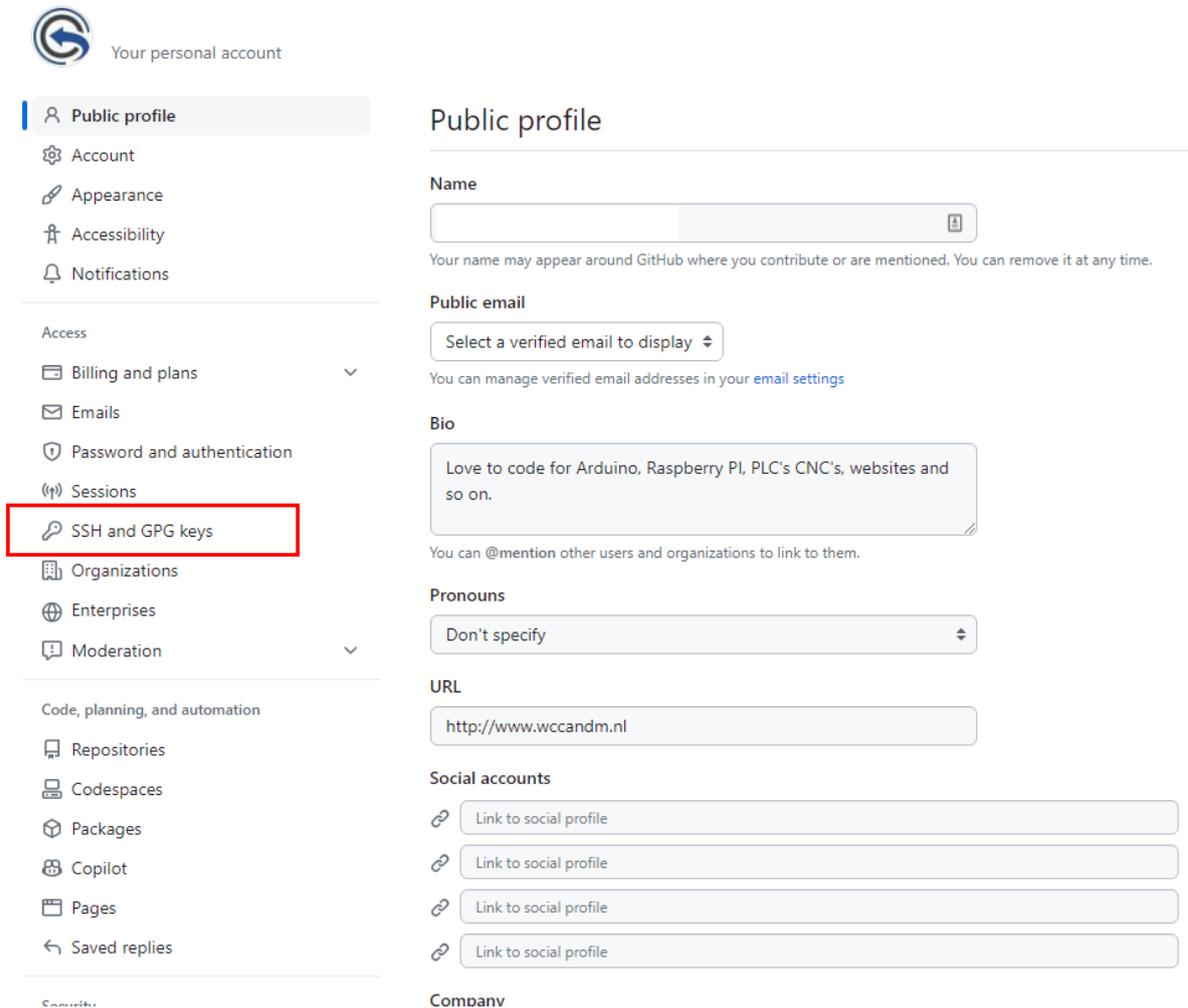
Open testkey1.pub in notepad and switch word wrap on.
You can see that the content of the file starts with "ssh-rsa" and ends with your.email@domain.com

```
testkey1.pub - Notepad
File Edit Format View Help
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACj+a0peFJYwgo6m4Gw16kE
+a1l6hn6Rfz7C13GHuDHs8VzEd0xP8XA0MeLWqKFK6MT13rPet0TrnNZGaxrWw5G/RQFu8+DzDxwCxsqJ2FAI1P
+0BmPaNaWKBGGJeoNt7X5y01jr5oPgK1rWC8m+5K30Fmv467gLyZQdtVQab3/z8E9p20xVUAhSYeCrcYJ
+T6fCgQd7B1PZ4w6woo59S5LdZcsK/WlNe5ztRP+HHaP7PjxNMiZRwILqeoZapHWginT
+070cEvoqMJWdzdXt1c0sXqWkaE3zHKdHfDVRJqVzZUK0be1n4eA6p0DyCjmAzTTB9obNKabmM9d60Gx88LNqVjbycUP7RtVUAXyywtTZpheNb
+xz8h9ymPtXNZC6s0DEMm9y5jK7TXL2xU1A9aJrzjbByjWbhSrJM3QY
+GGbbuSKGkvRq/04X2p7vmwP5TSvHnFWQsQeJzCmc4bp1Z5f4sWsvMwf1bUynve9RBA0x8AyZe5wtnLMK6f3T
+KFBRHgX41uUucojHr0rCbPksQboROtYaHpJdTWLBgbc4zH4BCRKIsktQmudYiBJvkfjBYMpvJvtqIXbE1fjgu7YmK7apI1A7WrPvmuFVuyXYRKJm/Lm
0/W2zvLr5yw9/mogsba+ooh5RySR+D6PGqLWTOuG9c6yVwCZA3tSP4LqUw== 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.



The screenshot shows the GitHub account settings page. On the left, a sidebar contains various settings categories. The 'SSH and GPG keys' option is highlighted with a red rectangle. The main content area is titled 'Public profile' and includes fields for Name, Public email, Bio, Pronouns, URL, Social accounts, and Company.

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

SSH keys

New SSH key

There are no SSH keys associated with your account.
Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.
Learn how to [generate a GPG key and add it to your account](#).

Vigilant mode

Flag unsigned commits as unverified
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.
[Learn about vigilant mode.](#)

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

Add new SSH Key

Title

testkey1

Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQCj+a0peFJYwgo6m4Gwl6kE+aN6hn6Rfz7C13GHuDHs8VzEdOxP8XA0MeLWqKFK6MTI3rPet0TrnNZGaxrW
w5G/RQFu8+DzDxwCxsqJ2FAIIP+0BmPaNaWKBGGJeoNt7X5y0Jr5oPgKlrWC8m+5K30Fmv467gLyZQdtVQab3/z8E9p20xVUAhSYeCrcYJ+T6fCgQd7
BIPZ4w6wo059S5LdZcsK/WINe5ztRP+HHaP7PjxNMIzRwLlqeoZapHWginT+07OcEvoqMJWzdzt1cOsXqWkaE3zHKdHFdVRJqVzZUK0beln4eA6p0
DyCjmAzTTB9obNKabmM9d6OGx88LNqVjbycUP7RtvUAXyywutTZpheNb+xz8h9ymPtxNZC6s0DEMm9y5jK7TXL2xUIA9aJrzibByjWbhSrJM3QY+GG
bbuSKGkvRg/O4X2p7vmwP5TSvHnfWQsQejzCmc4bplZ5f4sWsvMwFibUynve9RBAAOxBayZe5wtNLMK6f3T+KfBRHgX41wUucojHr0rCbPksQboROT
YaHpJdTWLBgbc4zH4BCRKlsktQmudYiBJvkFjBYMjVutqlXbElfjgu7YmK7apllA7WrPvumuFvuyXYRkjm/Lm0/W2zyLr5yw9/moagsba+ooh5RySR+D6PG
qLWTOuG9c6yVwCZAJ3tSP4LqUw== your.email@domain.com
```

Add SSH key

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

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys



testkey1

SHA256: sV4CyvECpBxK1Eg3kD3c5mFXdLvXQ8HwncavJwvnrw8

SSH

Added on Oct 6, 2023

Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH problems](#).

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. These 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).

MINGW64:/c/Users/username/.ssh/id_rsa

```
wccandM@DESKTOP-NTS6QHT MINGW64 /c/Users/username/.ssh/id_rsa
$ eval "$(ssh-agent -s)"
Agent pid 1883

wccandM@DESKTOP-NTS6QHT MINGW64 /c/Users/username/.ssh/id_rsa
$ ssh-add c:/users/username/.ssh/id_rsa/testkey1
Identity added: c:/users/username/.ssh/id_rsa/testkey1 (your.email@domain.com)

wccandM@DESKTOP-NTS6QHT MINGW64 /c/Users/username/.ssh/id_rsa
$ |
```

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?

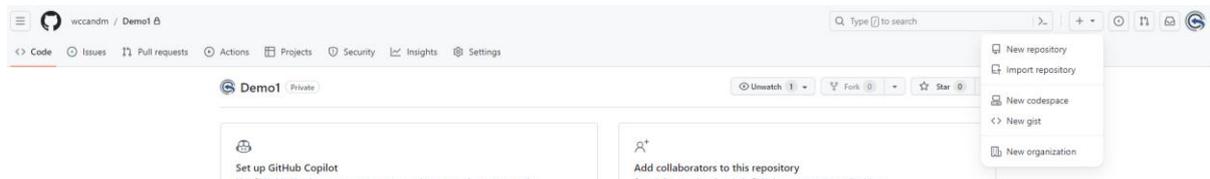
There is more than one solution:

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

13. Remote repository on GitHub.

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

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



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

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

/

✔ Demo1 is available.

Great repository names are short and memorable. Need inspiration? How about [vigilant-octo-bassoon](#) ?

Description (optional)

- Public**
Anyone on the internet can see this repository. You choose who can commit.
- Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

i You are creating a public repository in your personal account.

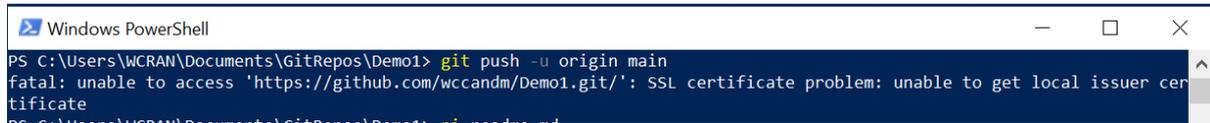
Create repository

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:

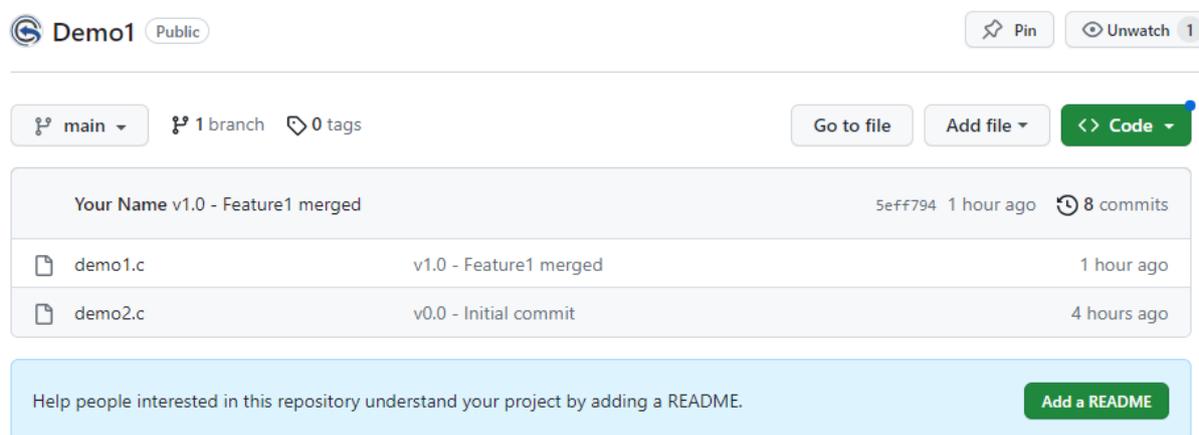


```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git push -u origin main
fatal: unable to access 'https://github.com/wccandm/Demo1.git/': SSL certificate problem: unable to get local issuer certificate
```

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...

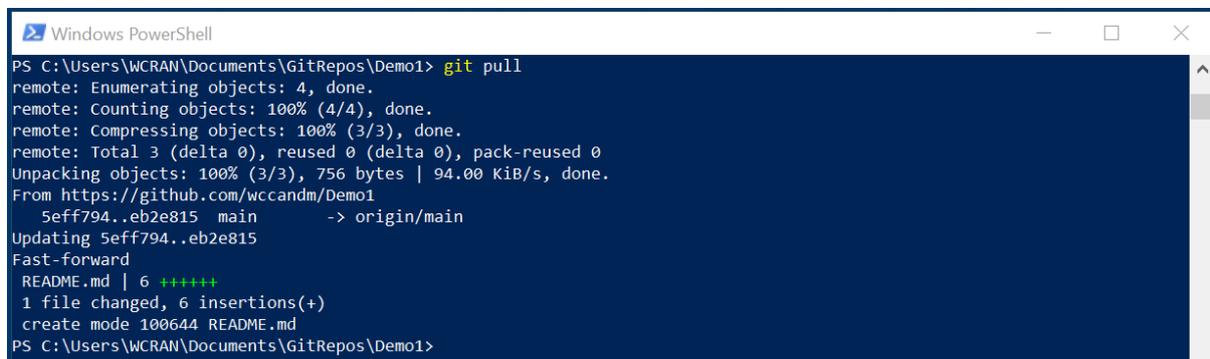


Your repository exists on GitHub.

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
```



```
Windows PowerShell
PS C:\Users\WCRAN\Documents\GitRepos\Demo1> git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 756 bytes | 94.00 KiB/s, done.
From https://github.com/wccandm/Demo1
 5eff794..eb2e815  main    -> origin/main
Updating 5eff794..eb2e815
Fast-forward
 README.md | 6 ++++++
 1 file changed, 6 insertions(+)
 create mode 100644 README.md
PS C:\Users\WCRAN\Documents\GitRepos\Demo1>
```

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

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
42 }
```

```
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:

 **Demo1** Public

 Pin
  Unwatch 1

1 2 3 4 5 6

main 1 branch 0 tags

 Go to file
 Add file
 Code

7 8 9 10 11

Your Name v1.01 - Changed blink speed to 500 ms

 961720e 26 minutes ago
  10 commits

1. The existing branches
2. The default branch
3. The releases 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

Commits

main

Commits on Sep 21, 2023

v1.01 - Changed blink speed to 500 ms Your Name committed 33 minutes ago	 961720e 
Created README.md file ...  wccandm committed 45 minutes ago	Verified  eb2e815 
v1.0 - Feature1 merged Your Name committed 2 hours ago	 5eFF794 
v0.3 - Made blink faster Your Name committed 3 hours ago	 a4b3db0 
v0.5 - Feature1 - blinkspeed as a variable Your Name committed 3 hours ago	 625f1ff 
v0.2 - Demo1.c added #include <Arduino.h> Your Name committed 4 hours ago	 a76b4bc 
v0.1 - Demo1.c changed with Arduino blink sketch Your Name committed 4 hours ago	 2147b2c 
Revert "v0.1 - Demo1.c changed with Aduino blink sketch" ... Your Name committed 4 hours ago	 c75991e 
v0.1 - Demo1.c changed with Aduino blink sketch Your Name committed 4 hours ago	 ed248b6 
v0.0 - Initial commit Your Name committed 5 hours ago	 af12b45 

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