Homework 8

Student Name: Anoosha Sagar

Student ID: 605028604

1. UNIX Tutorial

UNIX

- UNIX is an operating system which was first developed in the 1960s, and has been in development ever since.
- The UNIX operating system is made up of three parts: the kernel, the shell, and the programs.
- Versions of UNIX: Sun Solaris, GNU/Linux, and MacOS
- Everything in UNIX is either a file or process.
- All files are grouped together in a grouped directory structure. The top of the hierarchy is traditionally called *root*.

UNIX Commands

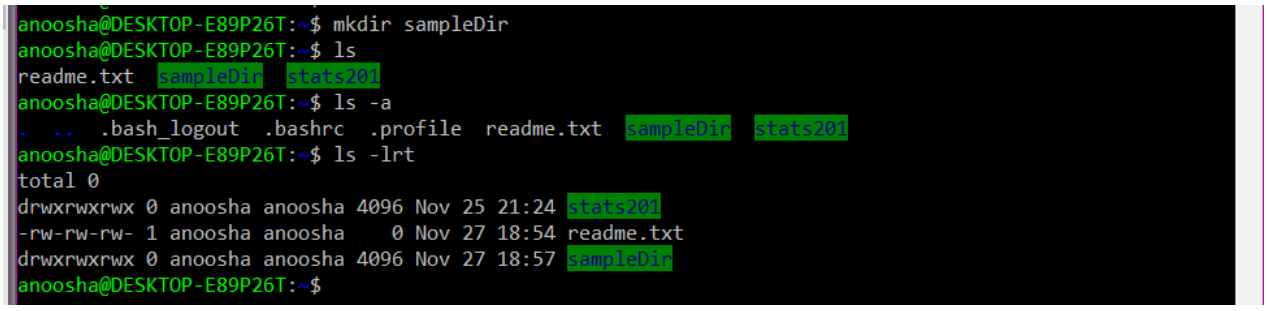
mkdir and ls

mkdir

- Used for creating a directory

ls

- It lists the contents of the current working directory
- Example usage:
  - ***ls***: list files and directories whose names do not start with a dot (.)
  - ***ls -a***: list all files in the directory
  - ***ls -lrt***: long listing of files, sorted by modification time in reverse



cd

- The ***cd*** command is used to change the directory
- Examples:
    - ***cd <<directory name>>***: change the directory to the directory listed in the command
    - ***cd*** .: stay where you are (dot(.) denotes the current directory)
    - ***cd*** **..:** move to the parent directory ((..) denotes parent of the current directory)

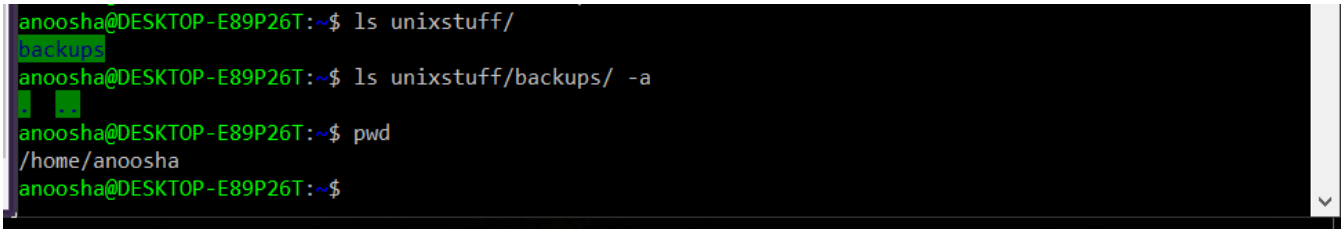
ls Using Pathnames and pwd

pwd

- displays the path of the current directory

Examples of ls using pathnames

- ***ls <<directory>>***: list contents of directory specified
- ***ls <<directory>> -a***: list all contents of the directory specified



cp, mv, rm, rmdir

cp

- ***cp file1 file2***: copy file1 in the current working directory and call it file2
- Examples:
    - ***cp filename .***: copy filename to the current directory keeping the name same

mv

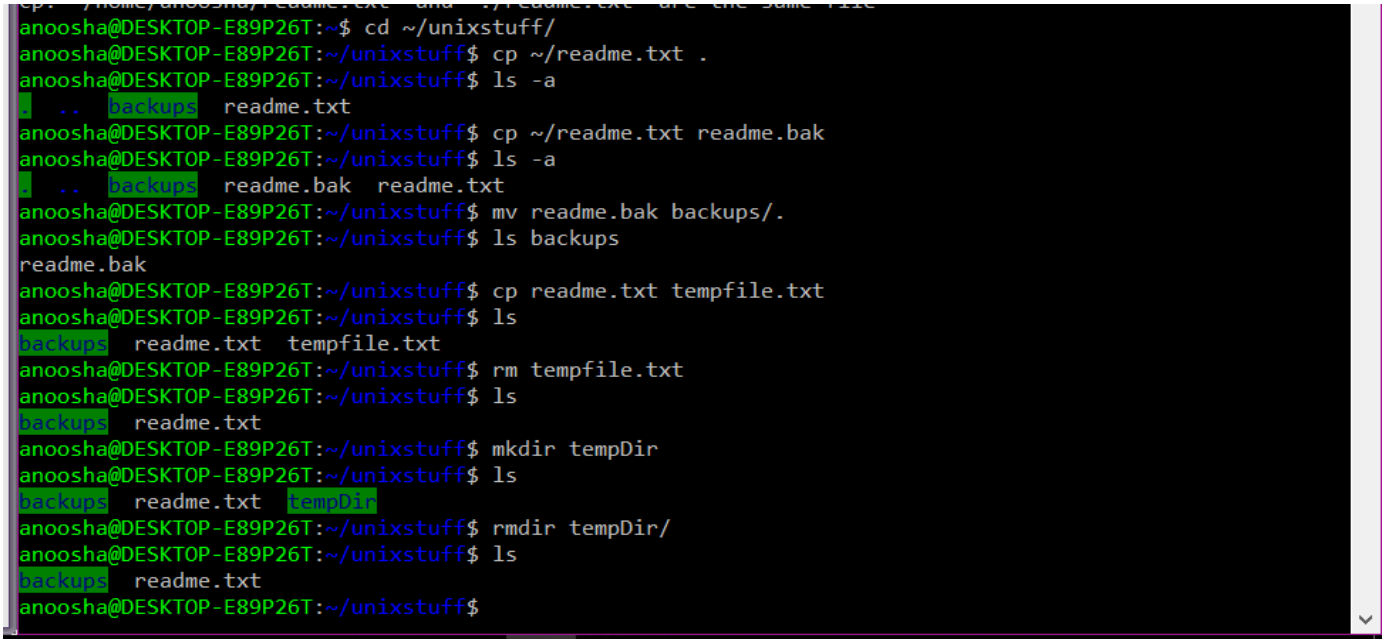
- ***mv file1 file2*** moves (or renames) file1 to file2
- Examples:
    - ***mv file directory/.***: move file to the specified directory and keep the name same

rm

- ***rm file*** deletes the specified file

rmdir

- ***rmdir directory*** deletes the specified directory



cat, less, head, and tail

cat

- ***cat filename*** displays the contents of the file on the terminal screen

less

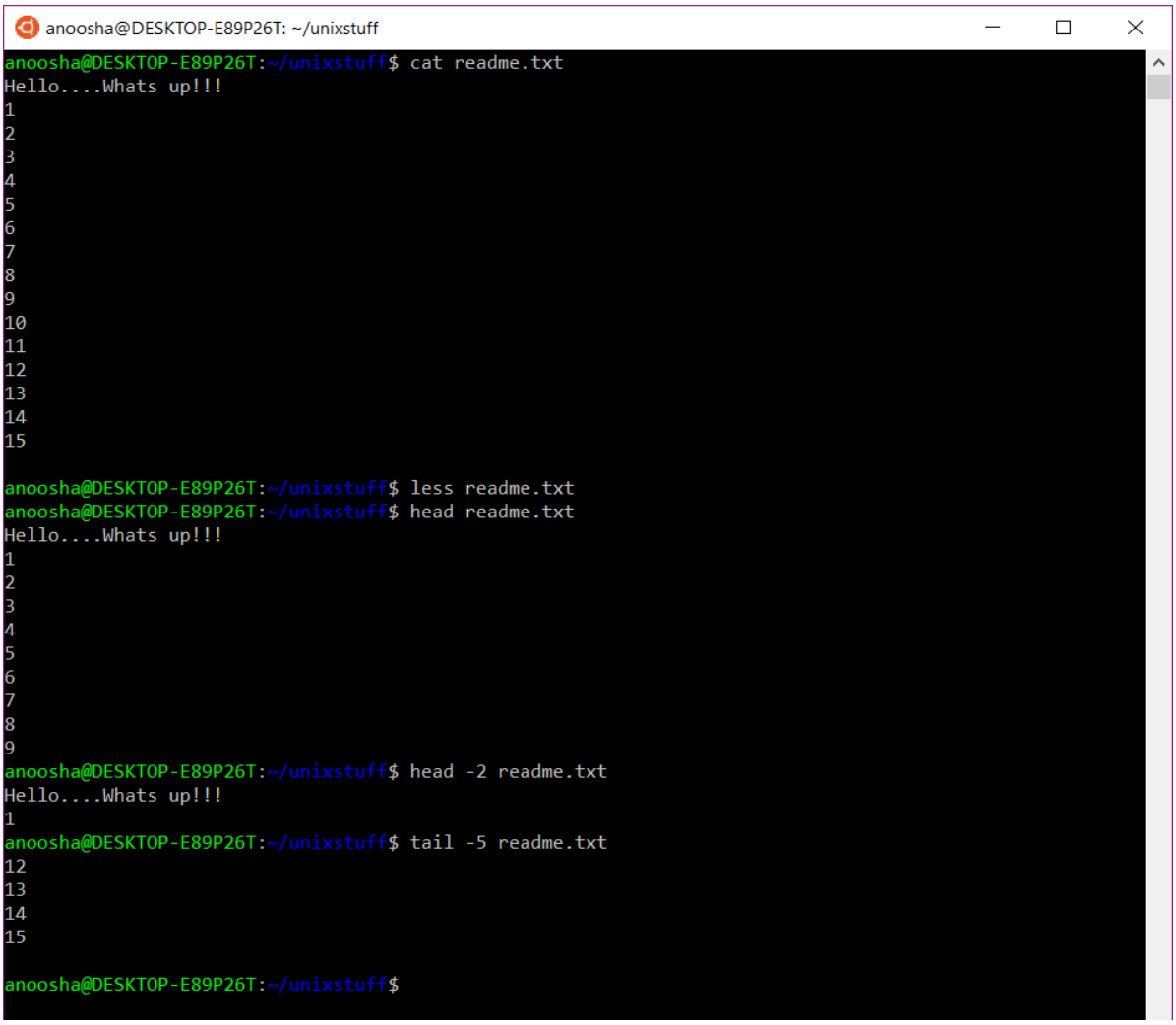
- ***less filename*** writes the contents of the file onto a separate screen one page at a time

head

- ***head filename*** writes the first 10 lines of the file to the terminal screen
- ***head -n filename*** writes the first n lines of the file to the terminal screen

tail

- ***tail filename*** writes the last 10 lines of the file to the terminal screen
- ***tail -n filename*** writes the last n lines of the file to the terminal screen



Simple searching using less

Steps

1. less filename
2. /[word to search]

less finds and highlights the keywords

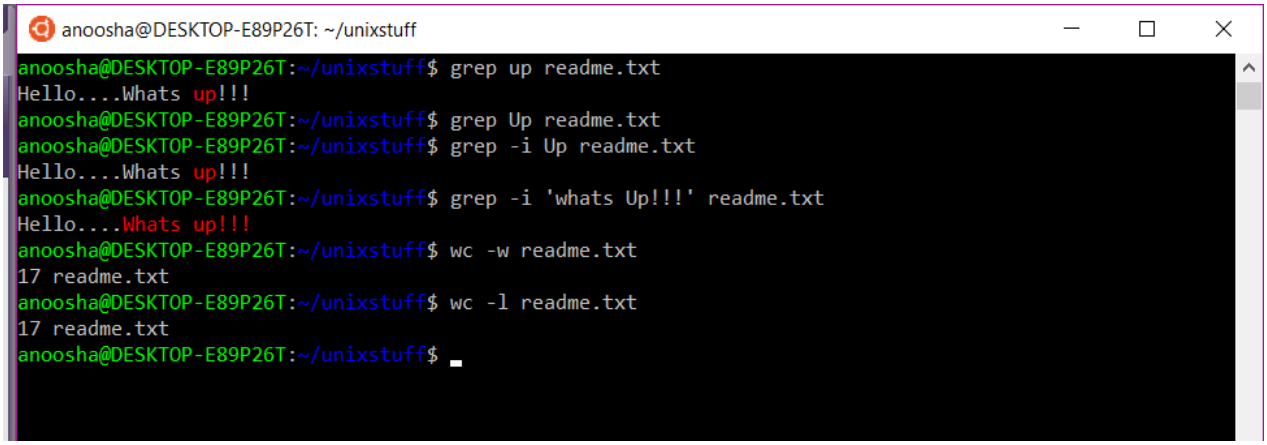
grep and wc

grep

- ***grep*** searches files for specified words or patterns
- Examples
    - ***grep keyword filename***: prints out each line in the file containing the keyword
    - ***grep -i keyword filename***: ignores case
    - ***grep 'phrase' filename***: searches for phrase in file

wc

- ***wc -w filename***: helps find number of words in the file
- ***wc -l filename***: helps find number of lines in file



Redirecting the Output

Redirect output of command

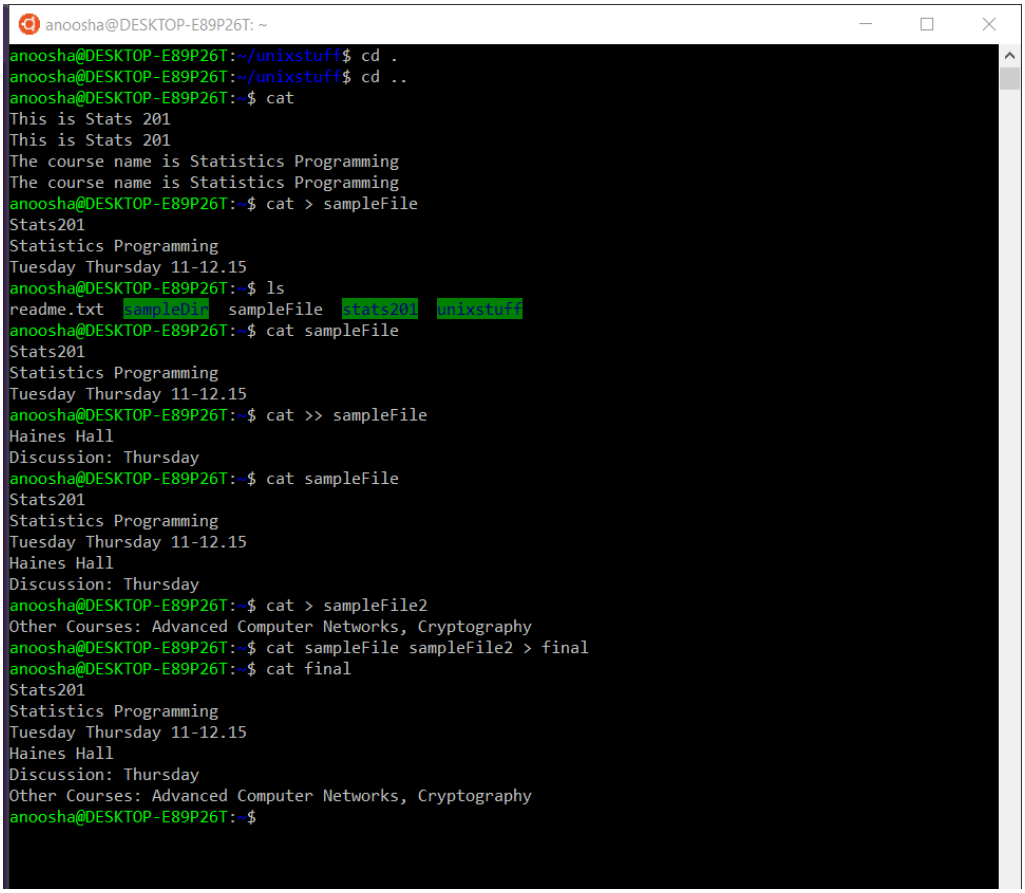
Example: cat > filename

Appending to a file

Example: cat >> filename

Concatenate files

Example: cat file1 file2 > file3



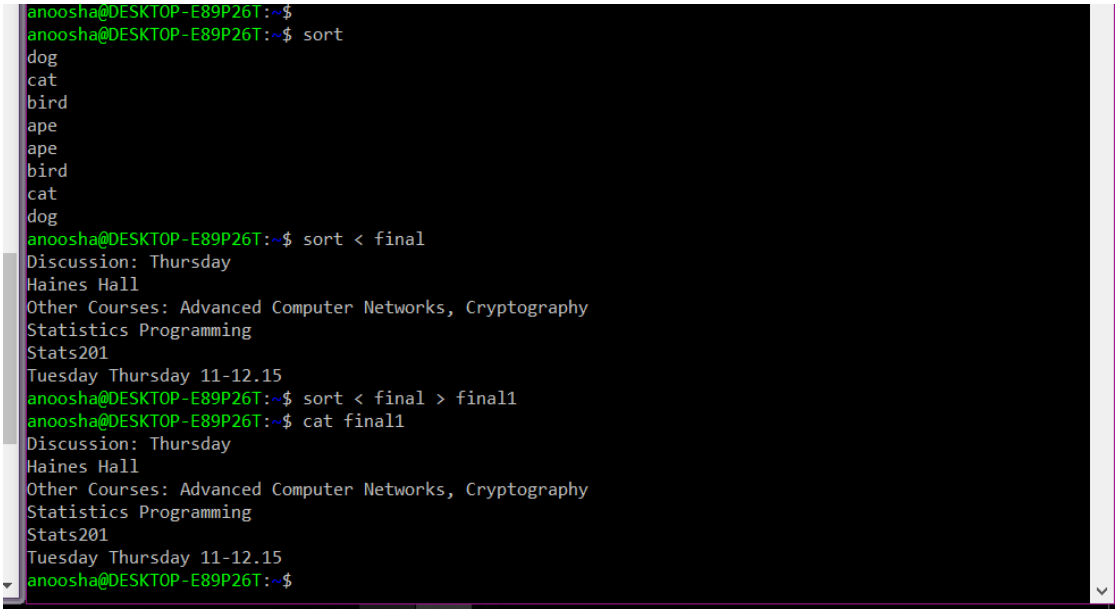
Redirecting the input

Example: sort < filename

Using < we can redirect the input to come from a file rather than keyboard

Example: sort <f file1 > file2

Outputs sorted file to another file



Pipes

Used when we want to connect the output of a command to the input of a second command. The symbol for pipe is a vertical bar (|)

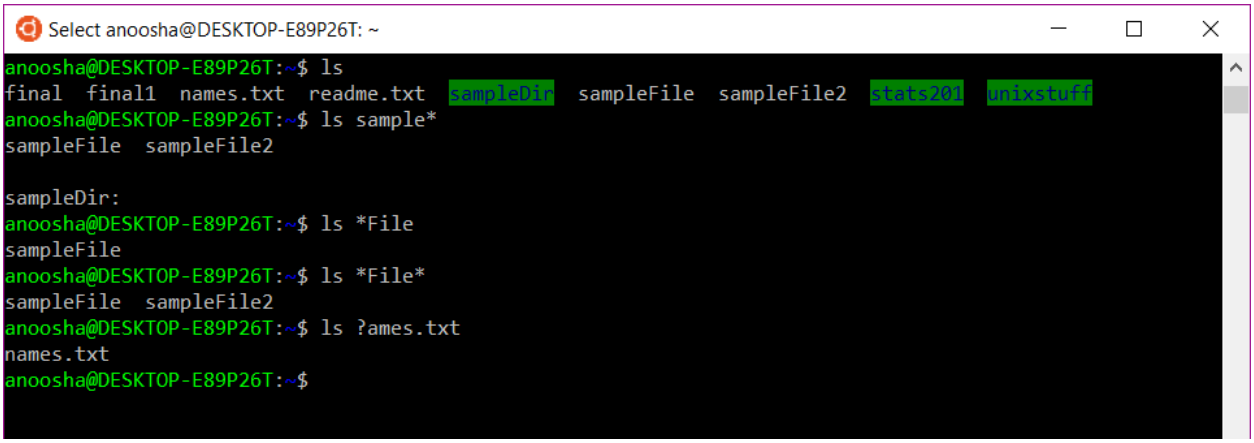
Wildcards

*

- The character * is called a wildcard, and will match against none or more character(s) in a file (or directory).
- Example:
    - ***ls list****: list all files in the current directory starting with list…
    - ***ls *list***: list all files in the current directory ending with …list

?

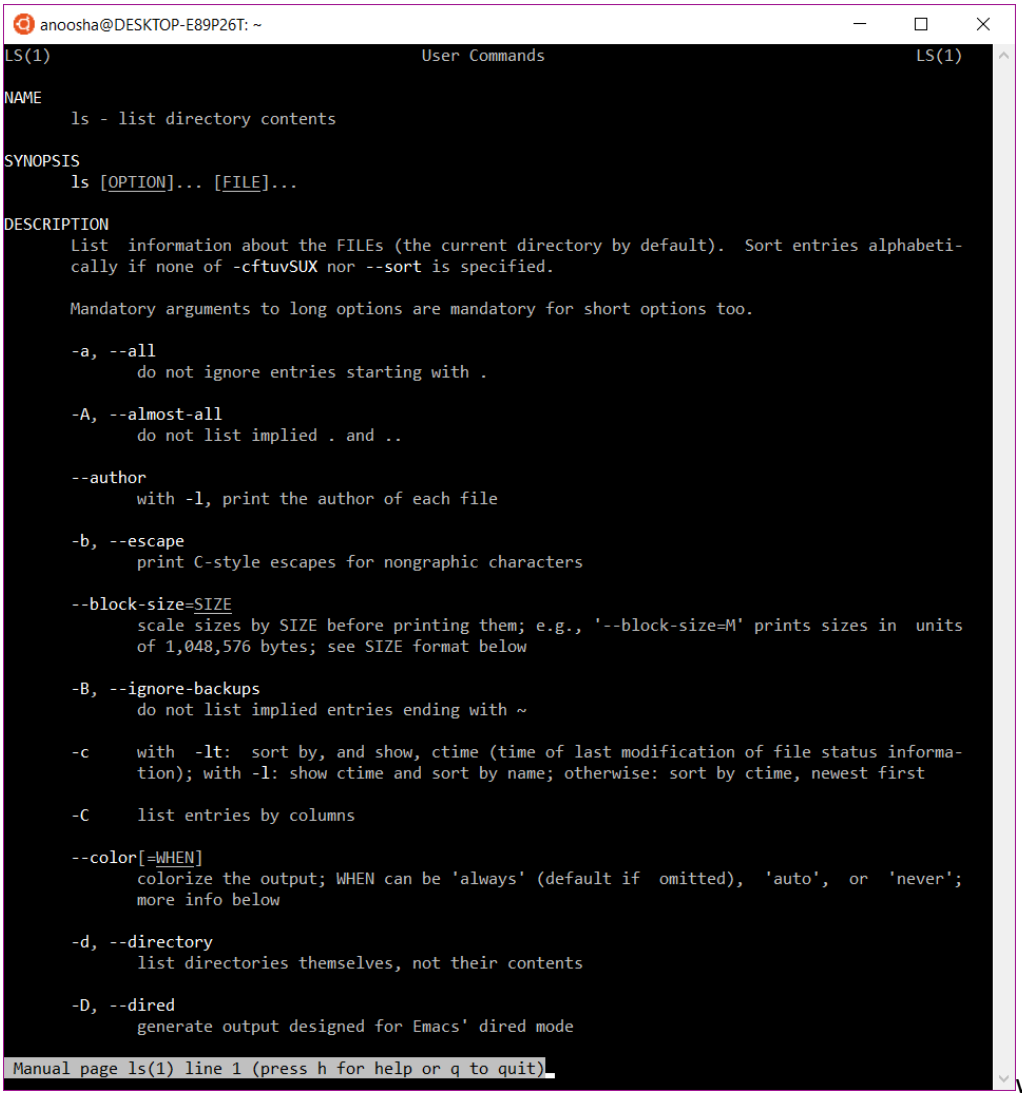
- The ? character will match exactly one character
- Example
    - ***ls ?list***



Online Manual

Example: man ls

Loads the manual page for the ls command



whatis

Example: whatis wc

Gives a one line description of the command

```
anoosha@DESKTOP-E89P26T:~$ whatis wc
wc (1)               - print newline, word, and byte counts for each file
anoosha@DESKTOP-E89P26T:~$
```

Apropos

apropos keyword *will give the commands with the keyword in their manual page header.*

```
anoosha@DESKTOP-E89P26T:~$ apropos copy
cp (1)               - copy files and directories
cpgr (8)             - copy with locking the given file to the password or group file
cpio (1)             - copy files to and from archives
cppw (8)             - copy with locking the given file to the password or group file
dd (1)               - convert and copy a file
debconf-copydb (1)   - copy a debconf database
git-checkout-index (1) - Copy files from the index to the working tree
install (1)          - copy files and set attributes
ntfscp (8)           - copy file to an NTFS volume.
rcp (1)              - secure copy (remote file copy program)
rsync (1)            - a fast, versatile, remote (and local) file-copying tool
scp (1)              - secure copy (remote file copy program)
ssh-copy-id (1)      - use locally available keys to authorise logins on a remote machine
xfs_copy (8)         - copy the contents of an XFS filesystem
xfs_metadump (8)     - copy XFS filesystem metadata to a file
xfs_rtcp (8)         - XFS realtime copy command
anoosha@DESKTOP-E89P26T:~$
```

chmod

- This command is used for changing the access rights of a file
- Only the owner of a file can use chmod to change the permissions of a file
- Example of access rights for a file

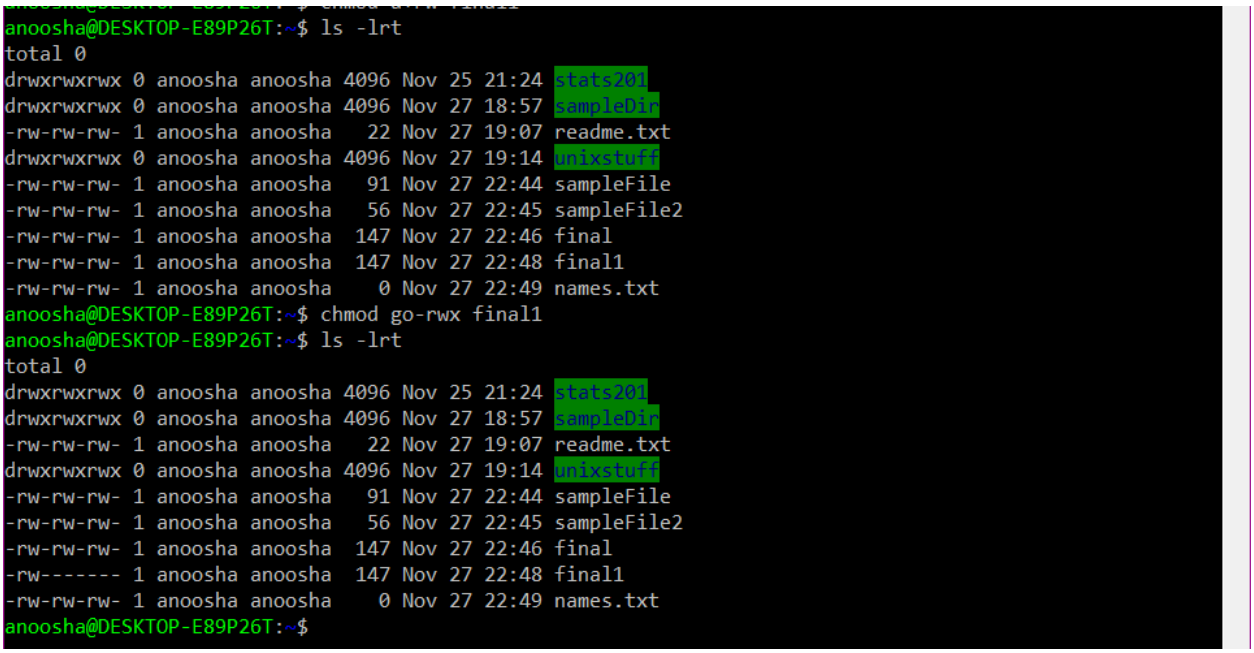
rwxrwxrwx	a file that everyone can read, write and execute (and delete).
-rw-------	a file that only the owner can read and write - no-one else can read or write and no-one has execution rights (e.g. your mailbox file).

- Options of chmod

Symbol	Meaning
u	user
g	group
o	other
a	all
r	read

w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

- Example usage
  - ***chmod go-rwx final1*** : remove read write and execute permissions for the file called final1 for group and others



Processes and Jobs

ps

- Used to see information about processes, with their associated PID and status
- Use ***&*** to run a process in background – e.g., ***sleep 10 &***
- To put a current foreground process in the background, use ***bg***

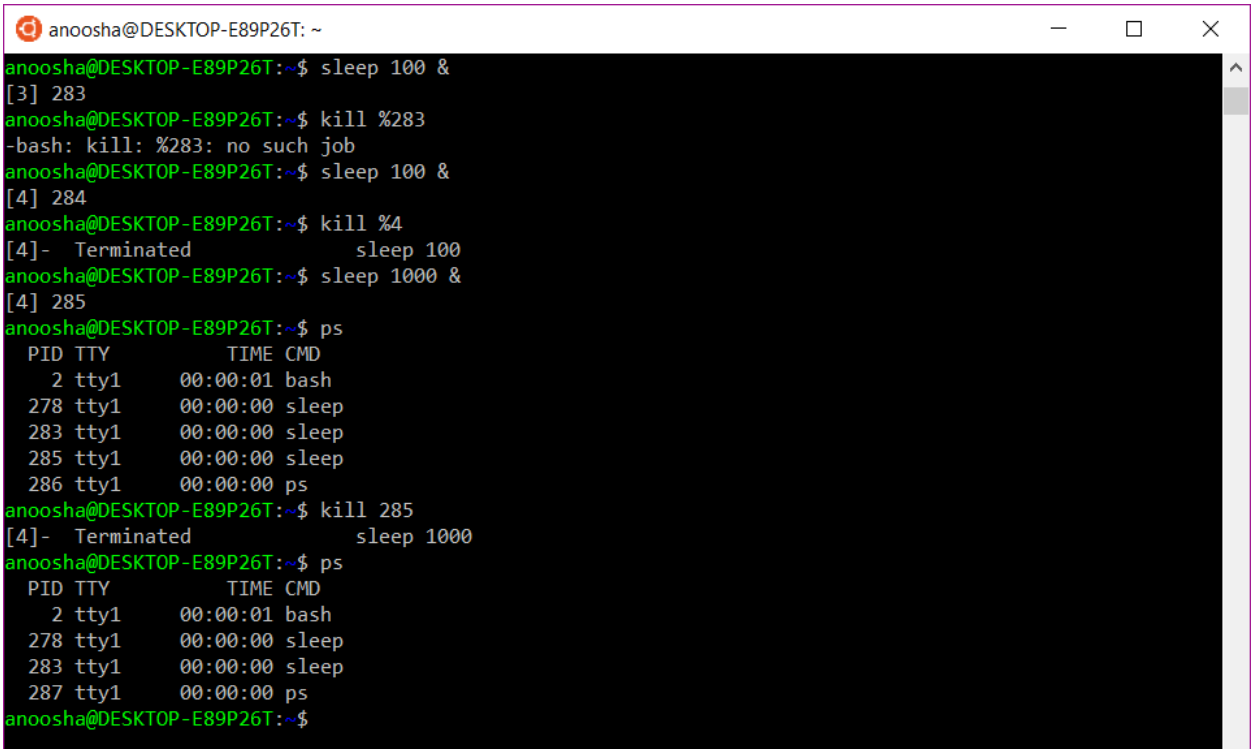
jobs

- It lists all running, backgrounded, and suspended processes
- To restart (foreground) a suspended process, use ***fg*** – e.g., ***fg %jobnumber***

kill

- Used for terminating processes
- Usage – ***kill %jobnumber*** or ***kill PID***

```
anoosha@DESKTOP-E89P26T:~$ ps
  PID TTY          TIME CMD
    2 tty1     00:00:01 bash
  275 tty1     00:00:00 ps
anoosha@DESKTOP-E89P26T:~$ sleep 10
anoosha@DESKTOP-E89P26T:~$ sleep 10 &
[1] 277
anoosha@DESKTOP-E89P26T:~$ sleep 1000
^Z[1]   Done                    sleep 10

[2]+  Stopped                 sleep 1000
anoosha@DESKTOP-E89P26T:~$ bg
[2]+ sleep 1000 &
anoosha@DESKTOP-E89P26T:~$ jobs
[2]+  Running                 sleep 1000 &
anoosha@DESKTOP-E89P26T:~$ fg %1
-bash: fg: %1: no such job
anoosha@DESKTOP-E89P26T:~$ fg %2
sleep 1000
^Z^X
[2]+  Stopped                 sleep 1000
anoosha@DESKTOP-E89P26T:~$ s
s: command not found
anoosha@DESKTOP-E89P26T:~$ ps
  PID TTY          TIME CMD
    2 tty1     00:00:01 bash
  278 tty1     00:00:00 sleep
  281 tty1     00:00:00 ps
anoosha@DESKTOP-E89P26T:~$ jobs
[2]+  Stopped                 sleep 1000
anoosha@DESKTOP-E89P26T:~$
```



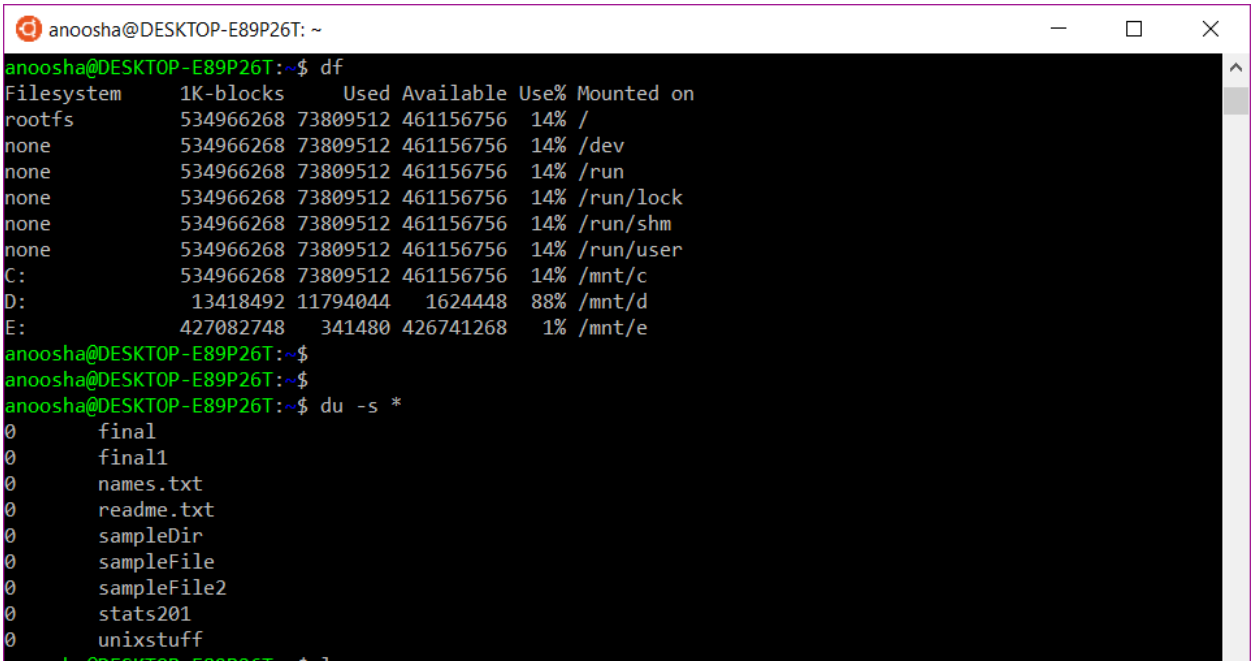
df

It reports the space left on the system

du

It outputs the number of kilobytes used by each subdirectory

***du -s ** *- will show a summary of all files and directories

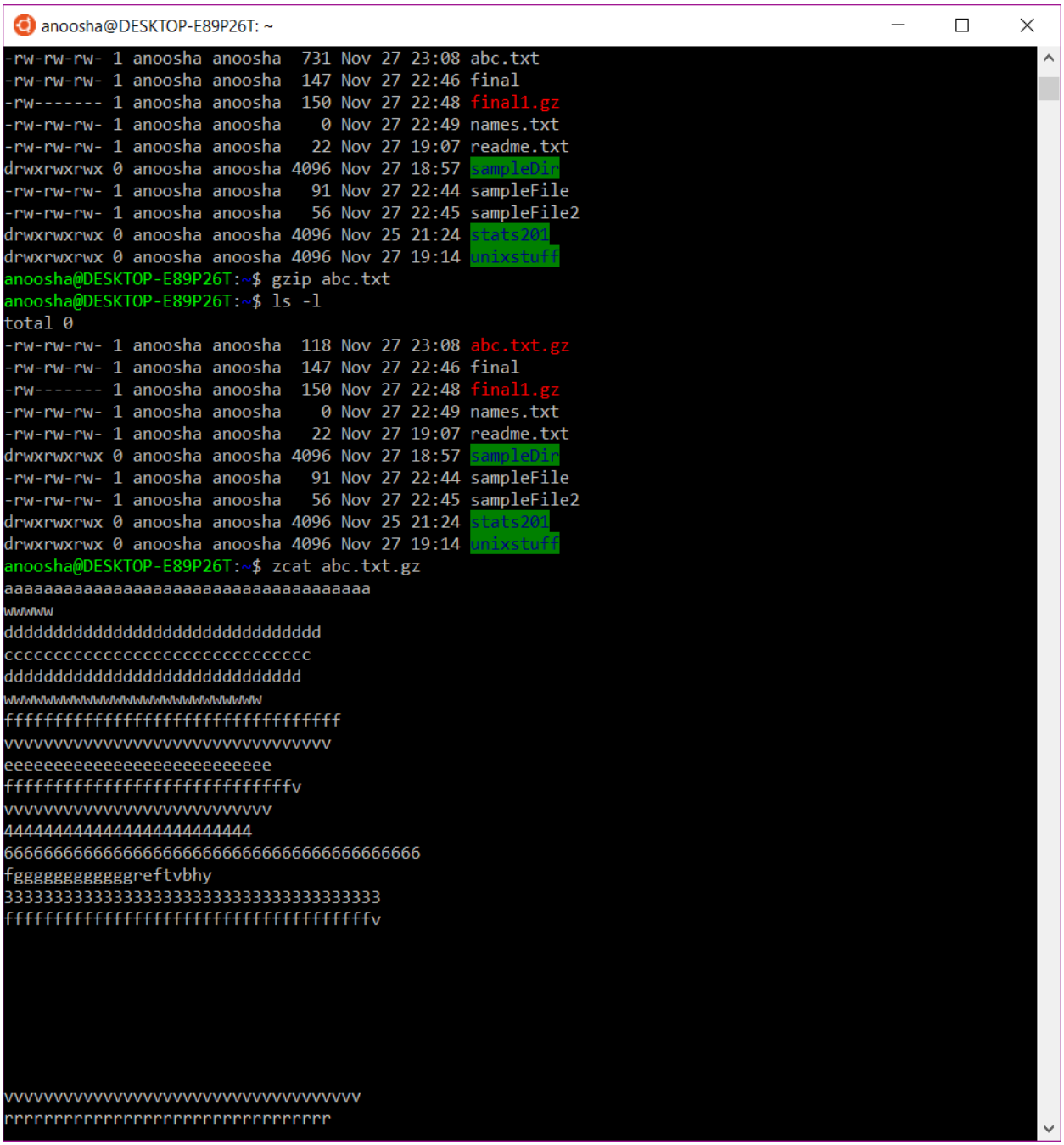


gzip

This command reduces the size of the file (compress the file)

zcat

This command reads the gzipped files without uncompressing them first



file

This command classifies the named files according to the type of data they contain.

diff

This command compares the contents of two files and displays the differences

find

This searches through the directories for files and directories with a given name, date, size, or any other attribute specified.

```
anoosha@DESKTOP-E89P26T:~$ file *
abc.txt.gz:  gzip compressed data, was "abc.txt", last modified: Tue Nov 28 07:08:42 2017, from Unix
final:       ASCII text
final1.gz:   gzip compressed data, was "final1", last modified: Tue Nov 28 06:48:00 2017, from Unix
names.txt:   empty
readme.txt:  ASCII text
sampleDir:   directory
sampleFile:  ASCII text
sampleFile2: ASCII text
stats201:    directory
unixstuff:   directory
anoosha@DESKTOP-E89P26T:~$ diff sampleFile sampleFile2
1,5c1
< Stats201
< Statistics Programming
< Tuesday Thursday 11-12.15
< Haines Hall
< Discussion: Thursday
---
> Other Courses: Advanced Computer Networks, Cryptography
anoosha@DESKTOP-E89P26T:~$ find . -name "*.txt" -print
./names.txt
./readme.txt
./unixstuff/readme.txt
```

history

This command shows the command history list

```
anoosha@DESKTOP-E89P26T: ~
./unixstuff/readme.txt
anoosha@DESKTOP-E89P26T:~$ find . -size +1M -ls
anoosha@DESKTOP-E89P26T:~$ history
    1  ls
    2  touch readme.txt
    3  ls
    4  cat readme.txt
    5  cls
    6  clr
    7  clear
    8  ls -lrt
    9  ls
   10  ls -a
   11  ls -lrt
   12  mkdir sampleDir
   13  ls
   14  ls -a
   15  ls -lrt
   16  mkdir unixstuff
```

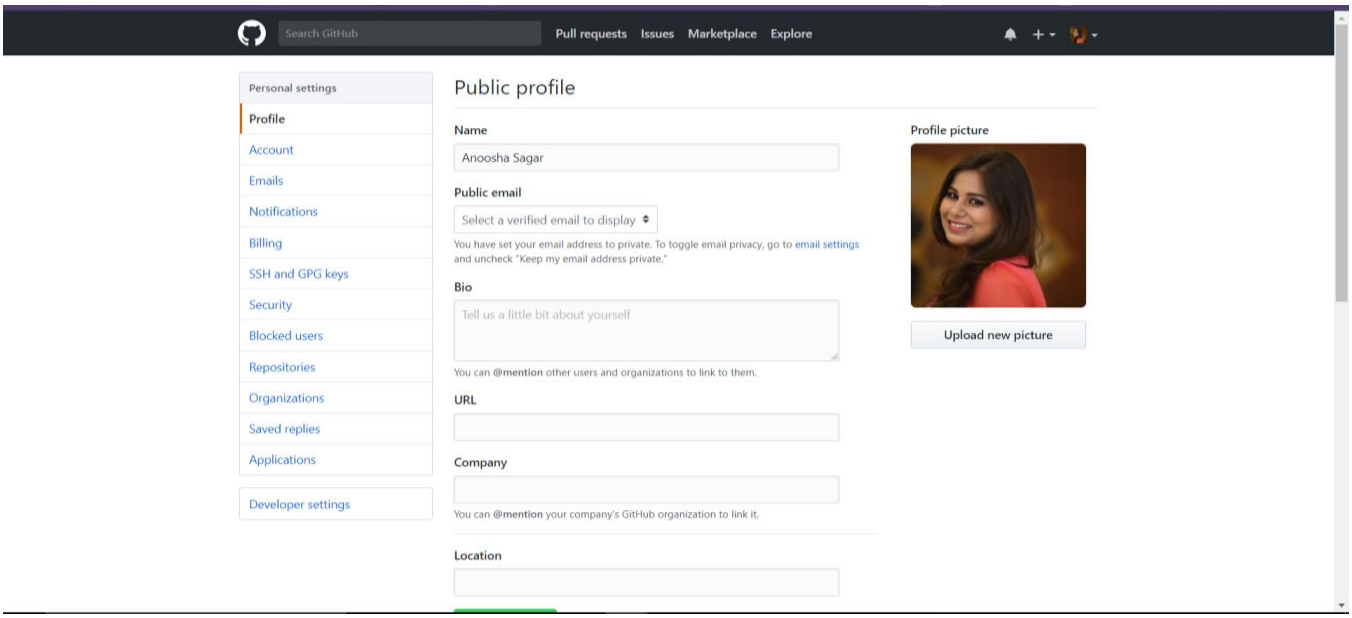
2. Git Tutorial

GitHub

GitHub is a Web-based Git version control repository hosting service. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

My GitHub Profile



Git Setup

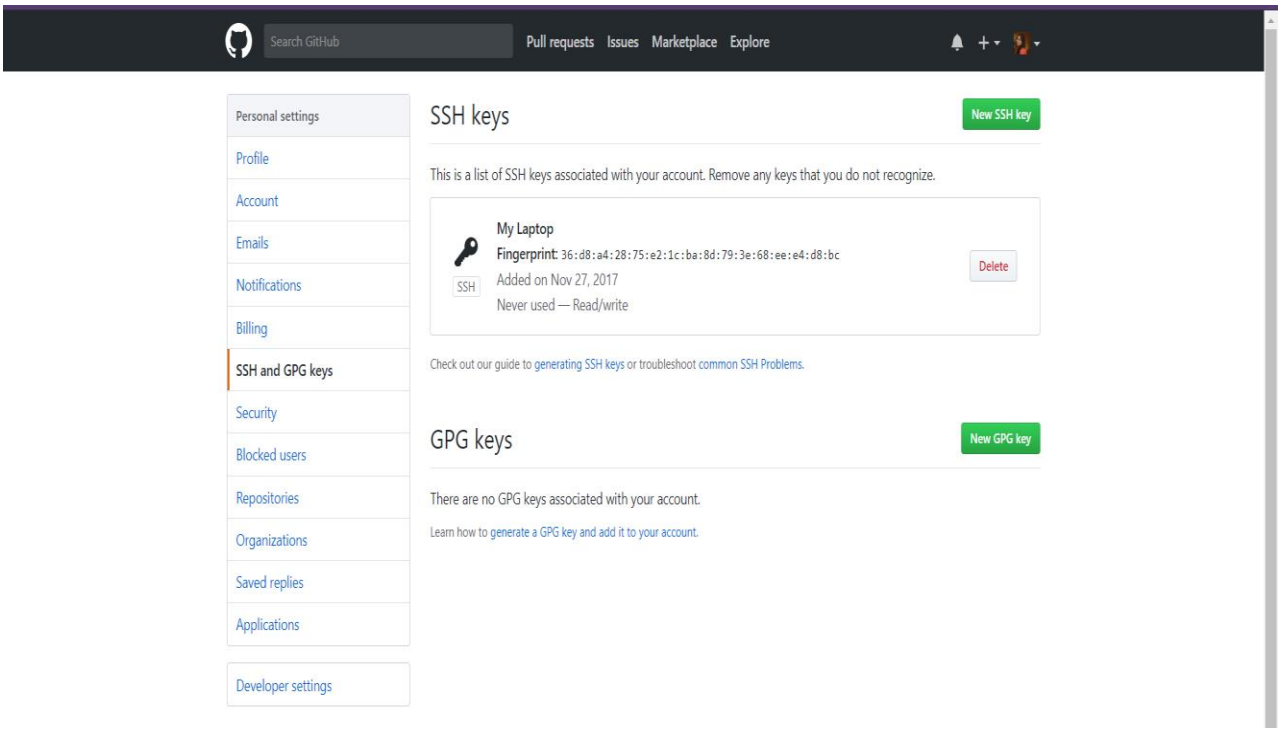
1. Get a GitHub account
2. Download and install git
3. Set up git with your username and email

```
anoos@DESKTOP-E89P26T MINGW64 ~
$ git config --global user.name "asagar06"

anoos@DESKTOP-E89P26T MINGW64 ~
$ git config --global user.email "anooshasagar@gmail.com"

anoos@DESKTOP-E89P26T MINGW64 ~
$ git config --global color.ui true

anoos@DESKTOP-E89P26T MINGW64 ~
$ git config --global core.editor emacs

anoos@DESKTOP-E89P26T MINGW64 ~
$
```

4. Set up ssh on the computer

```
anoos@DESKTOP-E89P26T MINGW64 ~
$ ssh-keygen -t rsa -C "anooshasagar@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/anoos/.ssh/id_rsa):
Created directory '/c/Users/anoos/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/anoos/.ssh/id_rsa.
Your public key has been saved in /c/Users/anoos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:gPhocZvevTepIPfyjKIfwfRIFuWkR5xfdlSWud3/984 anooshasagar@gmail.com
The key's randomart image is:
+---[RSA 2048]----+
|    .o+.    ...oo |
|   . Bo    o ..o  |
|  o B +. o .   o.|
|   X * ..      . o|
|  o B . S       .|
| . . o .        .|
|    + + .  .    .|
|    .+.= .+    .o|
|  .o...o*o .   .E|
+----[SHA256]-----+

anoos@DESKTOP-E89P26T MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.255.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.255.113' (RSA) to the list of known hosts.
Hi asagar06! You've successfully authenticated, but GitHub does not provide shell access.

anoos@DESKTOP-E89P26T MINGW64 ~
$
```

5. Copy paste the ssh public key into your github account settings

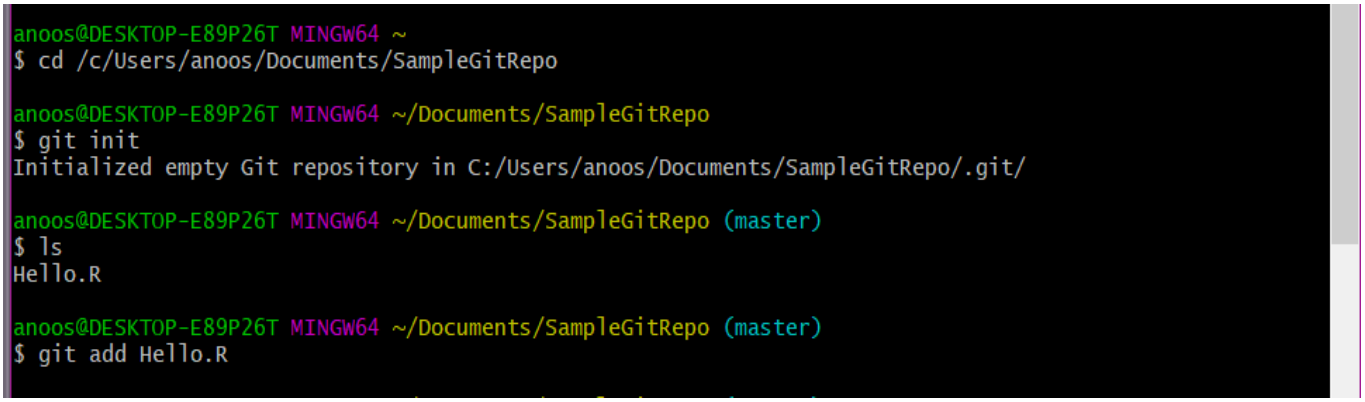


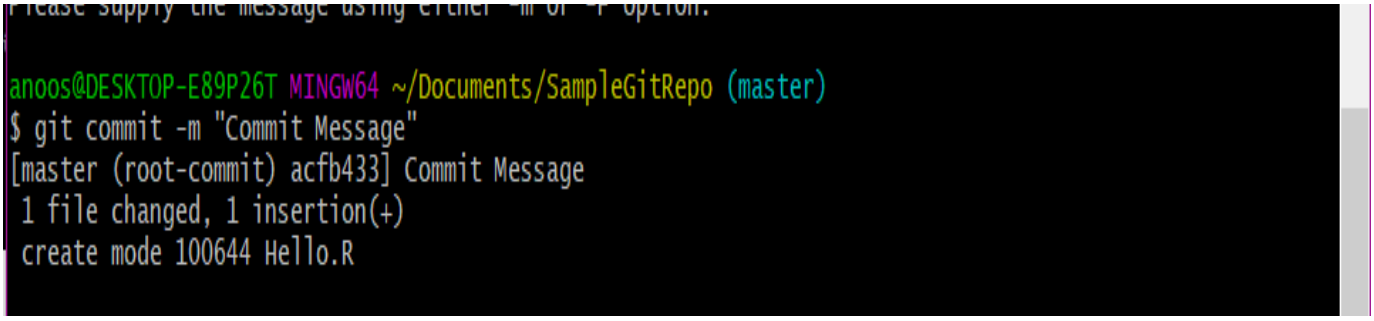
Setting Up a New Repository

1. Create a directory to contain the project.
2. Go into the new directory.
3. Type `git init`.
4. Write some code.
5. Type `git add` to add the file.

   git add is used to indicate that a modified file is ready to be added to the repository

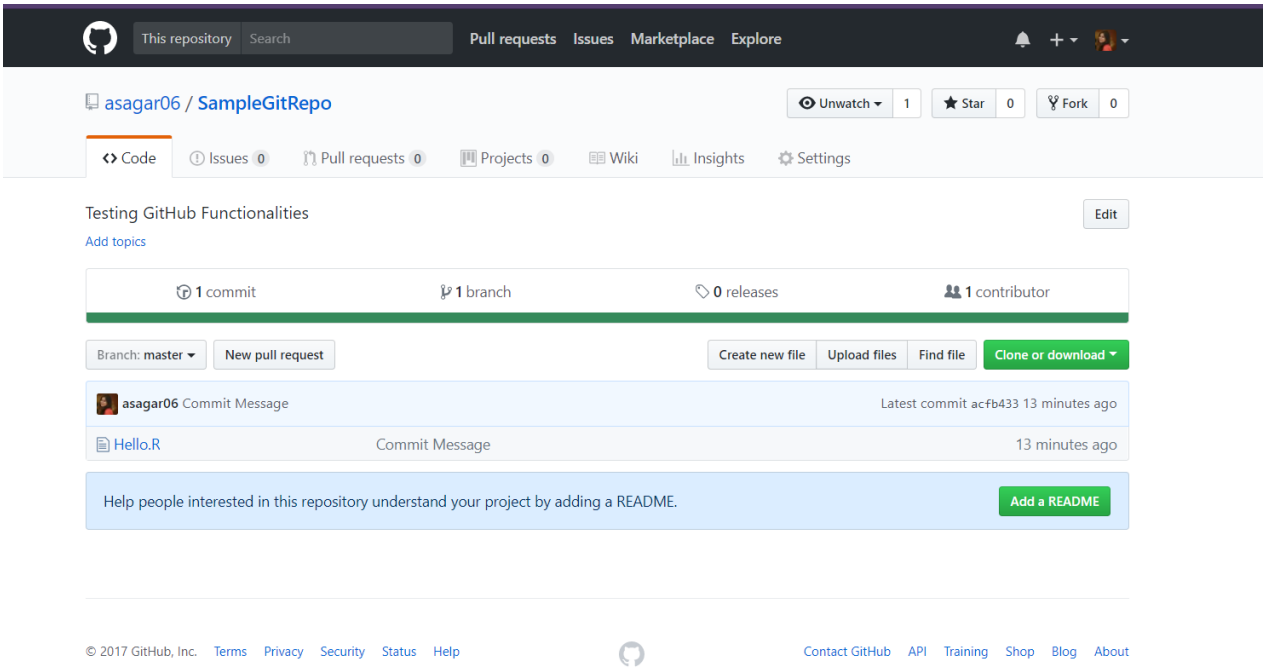
6. Type `git commit`.
   git commit adds the modifications to the repository
7. Type `git push`.
   git push pushes committed changes to github
8. Type `git status`.
   git status lists the files that have been changed and any other new additions
9. Type `git diff`.
   git diff shows the exact changes made

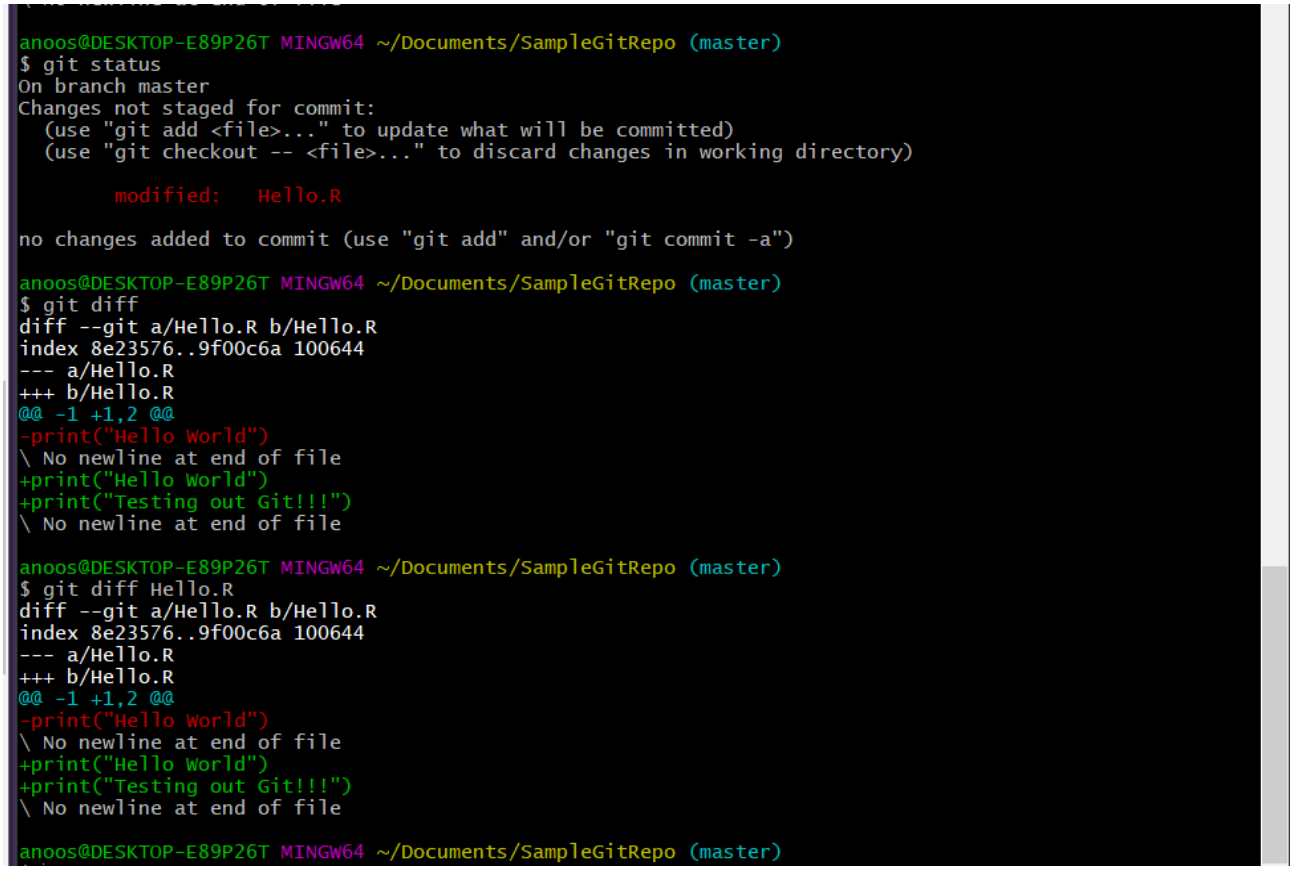




```
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleGitRepo (master)
$ git remote add origin https://github.com/asagar06/SampleGitRepo.git

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleGitRepo (master)
$ git remote -v
origin  https://github.com/asagar06/SampleGitRepo.git (fetch)
origin  https://github.com/asagar06/SampleGitRepo.git (push)

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleGitRepo (master)
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 230 bytes | 230.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/asagar06/SampleGitRepo.git
 * [new branch]      master -> master
```

Added Repository on GitHub



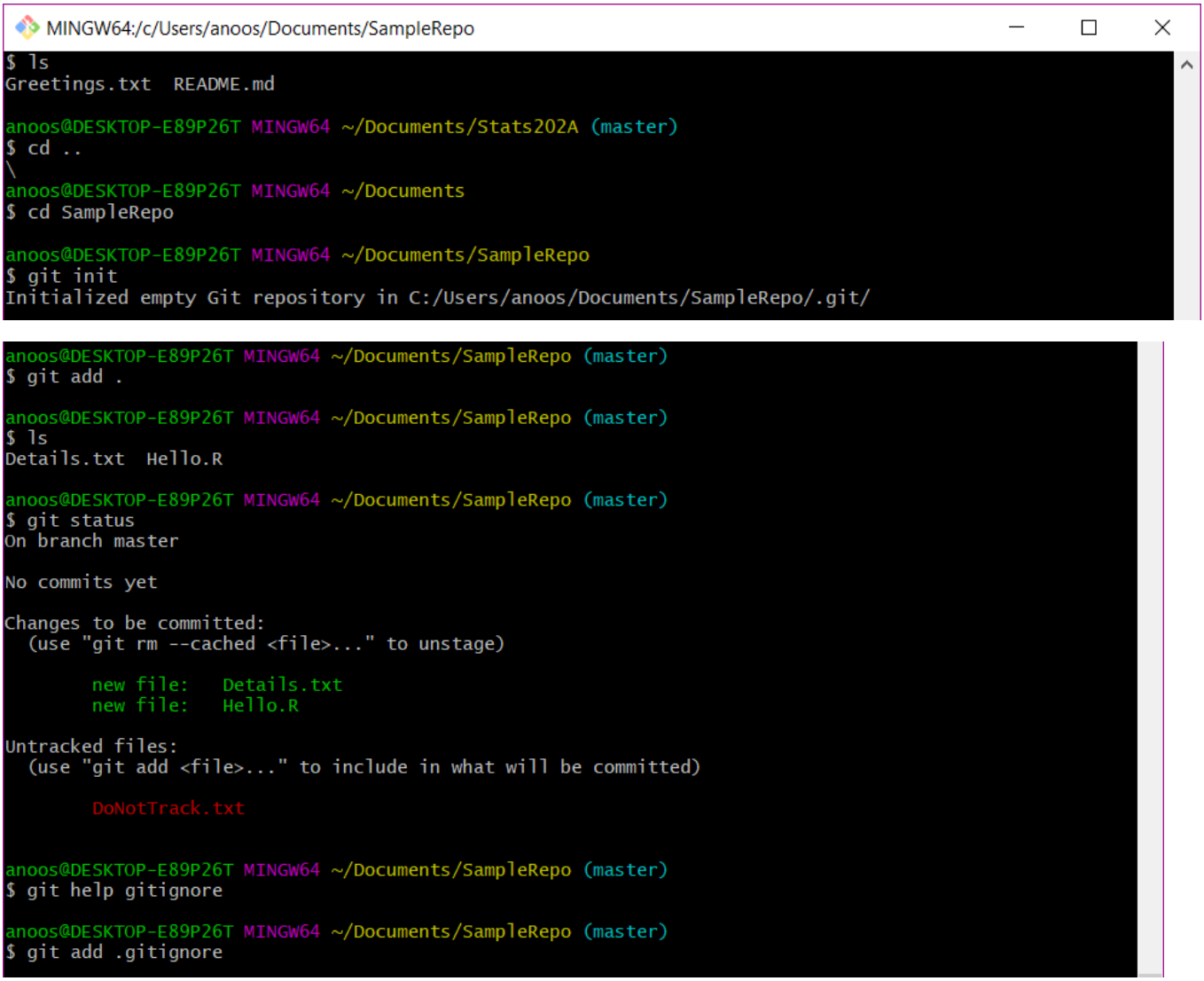


Create a New Repo from an Existing Project

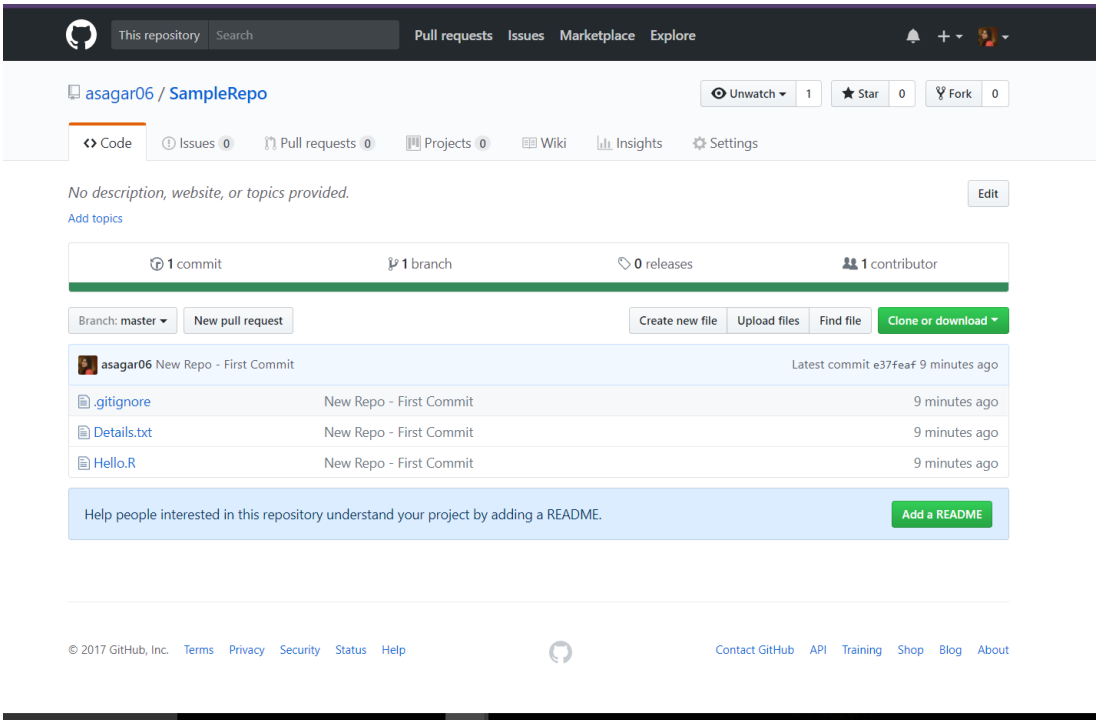
1. Go into the directory containing the project.
2. Type `git init`.
3. Type `git add` to add all of the relevant files.
4. If needed, create a `.gitignore` file right away, to indicate all of the files you don't want to track. Use `git add .gitignore`, too.
5. Type `git commit`.
6. Go to GitHub.
7. Log in to your account.
8. Click the new repository button in the top-right.

   ```
   $ git remote add origin https://github.com/username/new_repo
   ```

   ```
   $ git push -u origin master
   ```
9. Click the "Create repository" button.

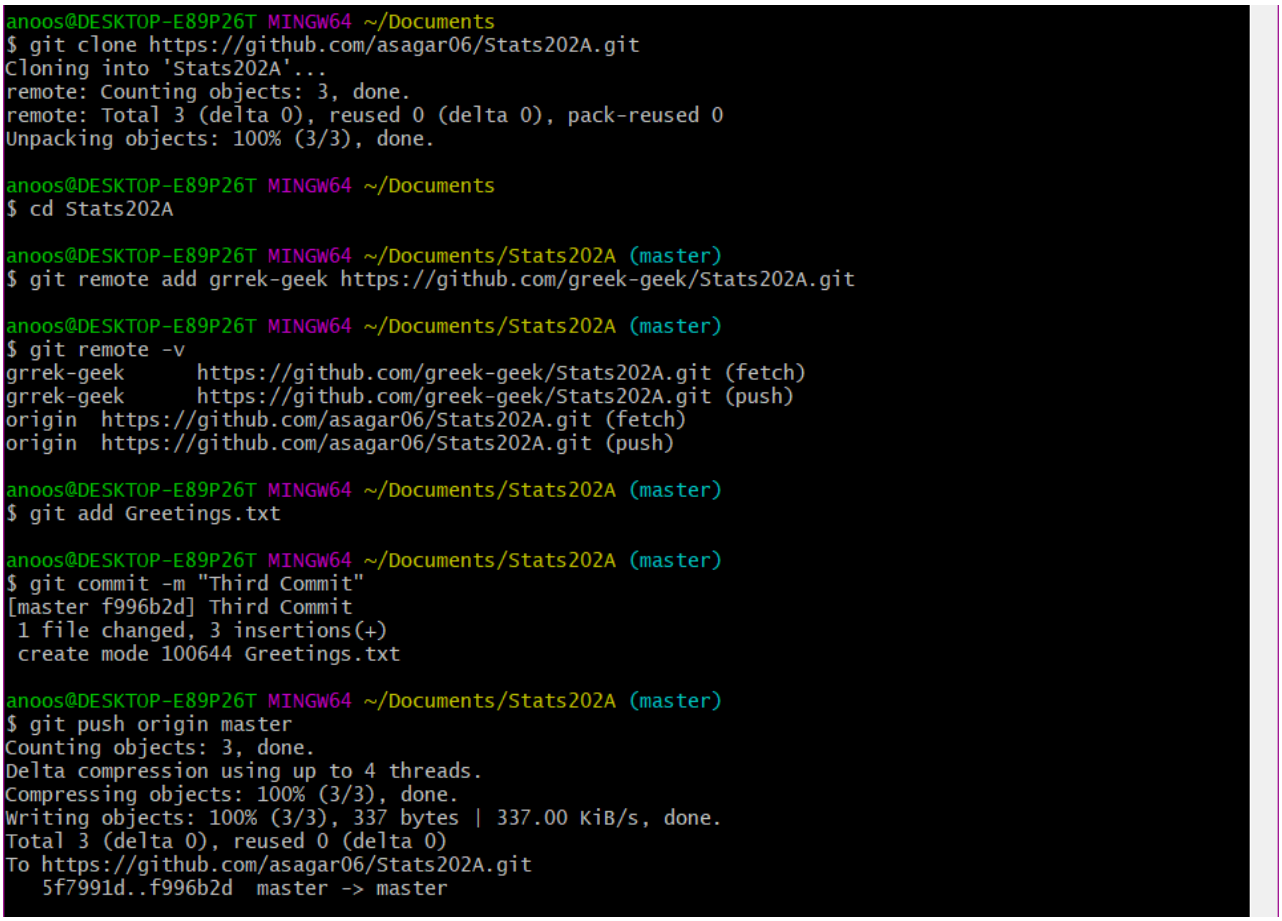

MINGW64:/c/Users/anoos/Documents/SampleRepo
$ ls
Greetings.txt README.md
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/Stats202A (master)
$ cd ..
\
anoos@DESKTOP-E89P26T MINGW64 ~/Documents
$ cd SampleRepo
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo
$ git init
Initialized empty Git repository in C:/Users/anoos/Documents/SampleRepo/.git/
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git add .
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ ls
Details.txt Hello.R
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git status
On branch master
No commits yet
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: Details.txt
new file: Hello.R
Untracked files:
(use "git add <file>..." to include in what will be committed)
DoNotTrack.txt
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git help gitignore
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git add .gitignore

```
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git status
On branch master

No commits yet

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

        new file:   .gitignore
        new file:   Details.txt
        new file:   Hello.R


anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git commit -m "New Repo - First Commit"
[master (root-commit) e37feaf] New Repo - First Commit
 3 files changed, 5 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Details.txt
 create mode 100644 Hello.R

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git remote add origin https://github.com/asagar06/SampleRepo.git

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$ git push -u origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 465 bytes | 232.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/asagar06/SampleRepo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/SampleRepo (master)
$
```

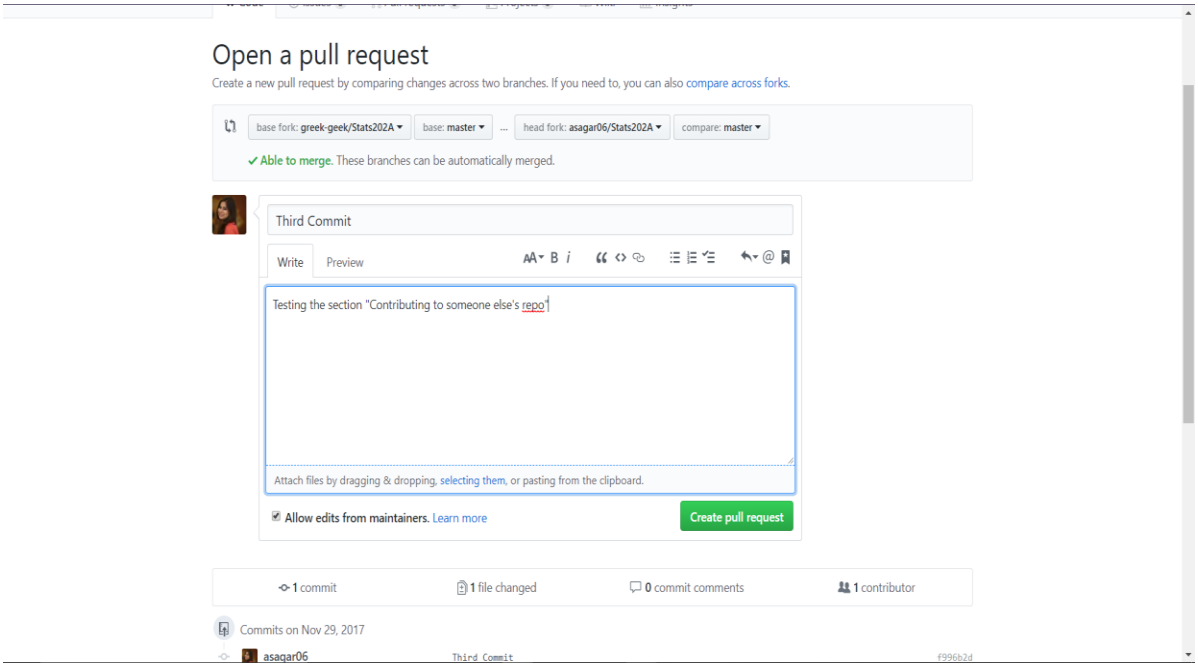


Contribute to Someone's Repository

1. Go to the repository on GitHub. (Say it's by `myfriend`, and is called `the_repo`, then you'll find it at `http://github.com/myfriend/the_repo`.)
2. Click the "Fork" button at the top right.
3. You'll now have your own copy of that repository in your GitHub account.
4. Change into that project directory (`the_repo`):
   `$ cd the_repo`
5. Add a connection to the original owner's repository.
   `$ git remote add myfriend git://github.com/myfriend/the_repo`
6. To check this `remote add` set up:
   `$ git remote -v`
7. Make changes to files.
8. `git add` and `git commit` those changes
9. `git push` them back to GitHub. These will go to *your version* of the repository.



Creating Pull Requests



Pulling Others' Changes

```
anoos@DESKTOP-E89P26T MINGW64 ~/Documents/Stats202A (master)
$ git pull grrek-geek master
From https://github.com/greek-geek/Stats202A
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> grrek-geek/master
Already up to date.

anoos@DESKTOP-E89P26T MINGW64 ~/Documents/Stats202A (master)
$ git push
Everything up-to-date
```

Handling Pull Requests

Using the GitHub website:

1. Go to your version of the repository.
2. Click on "Pull Requests" at the top.
3. Click on the request.

4. You'll see their comments on the pull request, and can click to see the exact changes.
5. If you want them to make further changes before you merge the changes into your repository, add a comment.
6. If you hate the whole idea, just click the "Close" button.
7. If you want to merge the changes into your repository, click the "Merge pull request" button.
8. Your GitHub repository will now be fixed, but you'll want to get them into your local repository, too.
9. Open a terminal/shell, and type

```
$ git pull
```