

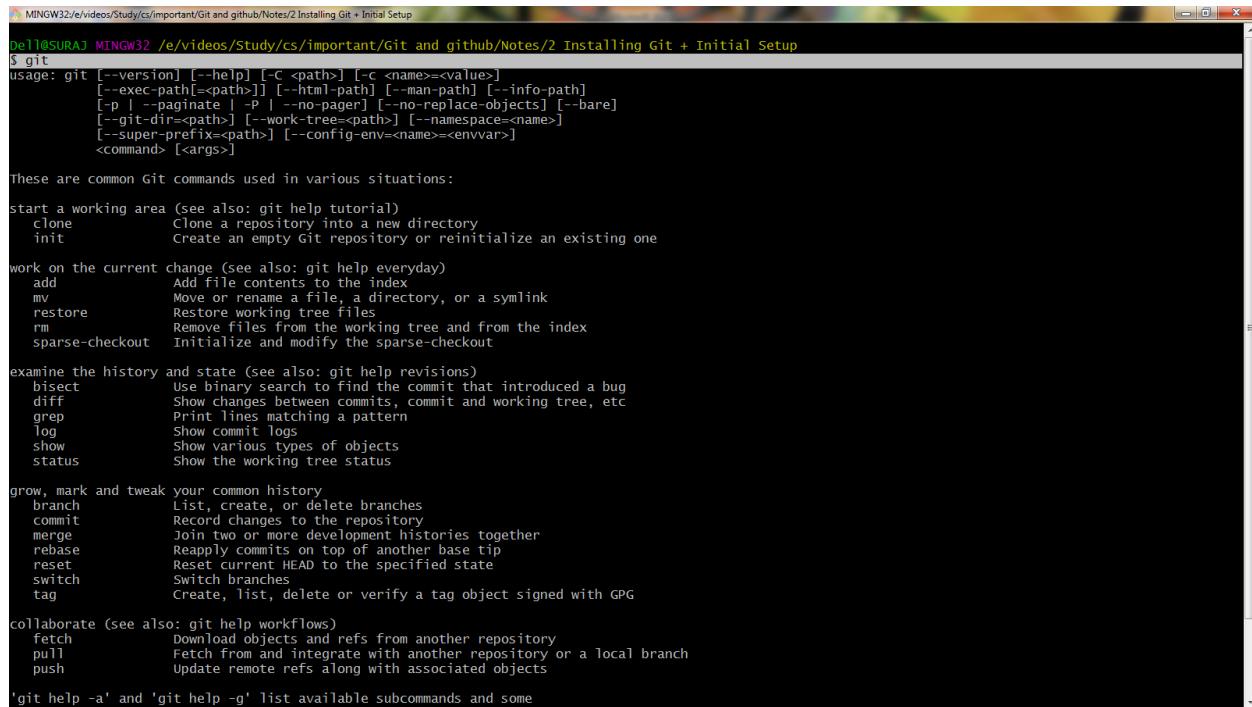
Download git from its official website → while installing no need to change any setting

After installing it, open “git bash” terminal

Right click mouse button and open it

Note: Ctrl + Mouse wheel button → zoom in and out

Write “git” and press enter → it will show all commands



```
Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github/Notes/2 Installing Git + Initial Setup
$ git
usage: git [--version] [--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>]
   [--super-prefix=<path>] [--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
  sparse-checkout Initialize and modify the sparse-checkout

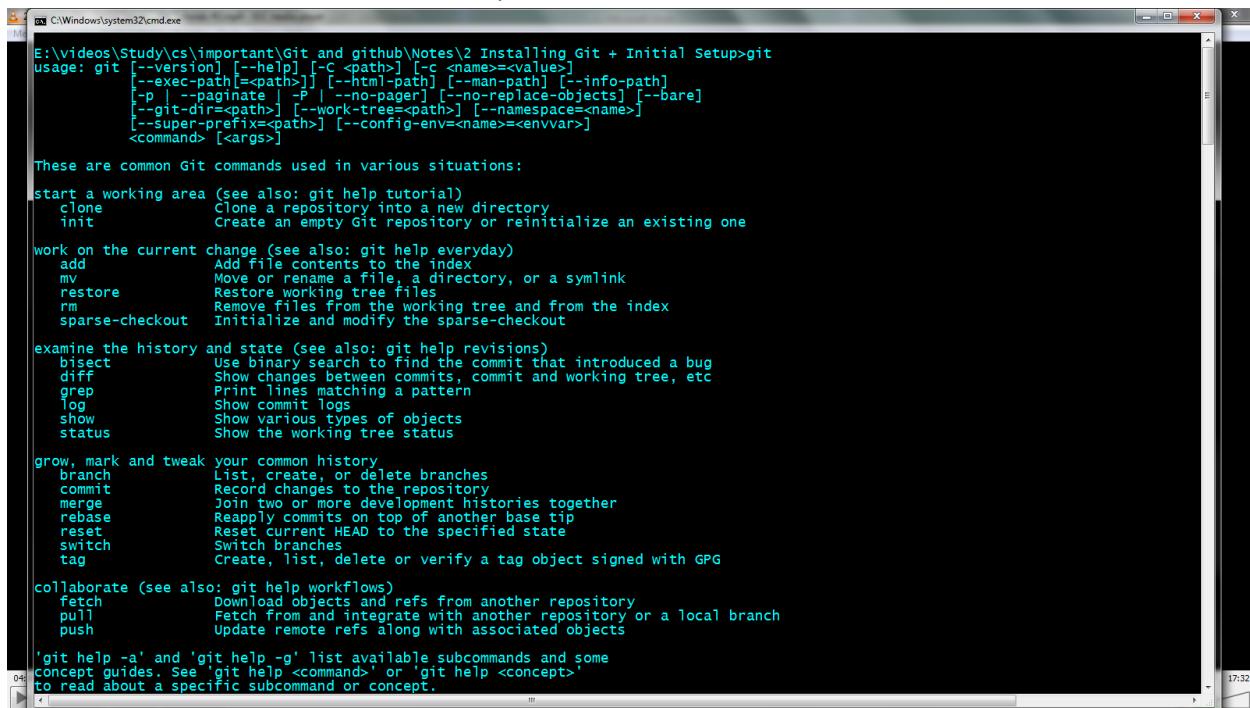
examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  diff      Show changes between commits, commit and working tree, etc
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types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

These commands can work on “cmd” or “powershell”



```
E:\videos\Study\cs\important\Git and github\Notes\2 Installing Git + Initial Setup>git
usage: git [--version] [-h] [-C <path>] [-c <name>=<value>]
           [--exec-path=<path>] [-C <path>] [-c <name>=<value>]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-s <super-prefix=<path>] [--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
  sparse-checkout Initialize and modify the sparse-checkout

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

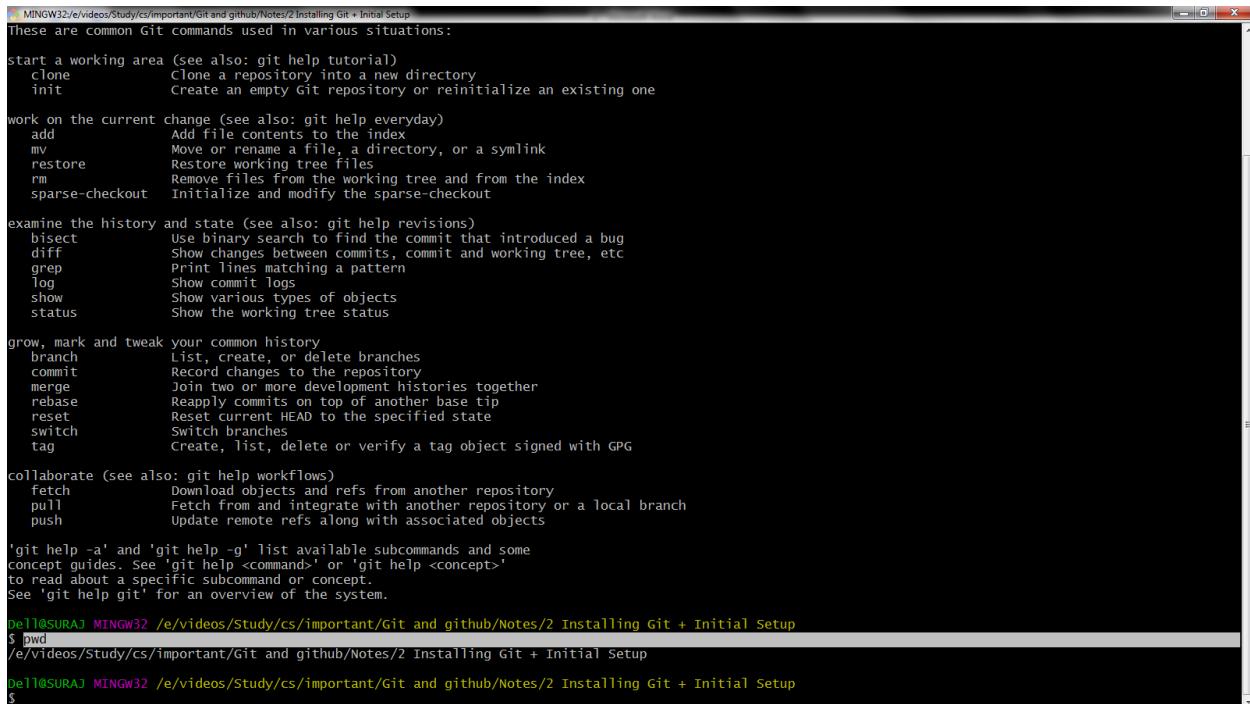
grow, mark and tweak your common history
  branch   List, create, or delete branches
  commit   Record changes to the repository
  merge    Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  reset   Reset current HEAD to the specified state
  switch   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.
```

Git bash is similar to linux

“pwd” → present working directory



```
MINGW32/e/videos/Study/cs/important/Git and github/Notes/2 Installing Git + Initial Setup
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
  sparse-checkout Initialize and modify the sparse-checkout

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.

Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github/Notes/2 Installing Git + Initial Setup
$ pwd
/e/videos/Study/cs/important/Git and github/Notes/2 Installing Git + Initial Setup
Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github/Notes/2 Installing Git + Initial Setup
$
```

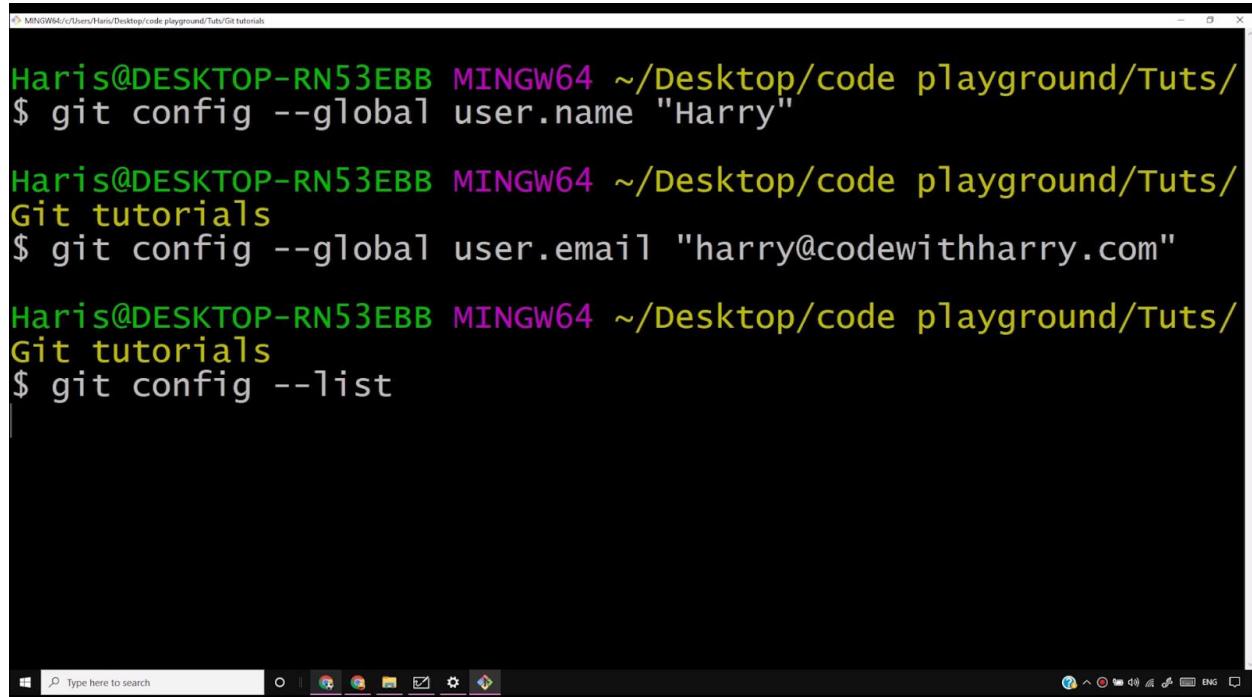
We can use “cd folderName” → change drive to go other folder (if it is present)

“cd ..” → go back to previous folder

“dir” → list all files and folders in current directory

“cd /c” → go to C drive. Here we can't use “cd c:” b/c it is windows command and “cd /c” is linux based command and git works on similar system to linux

Do the following for first time run to tell get about yourself

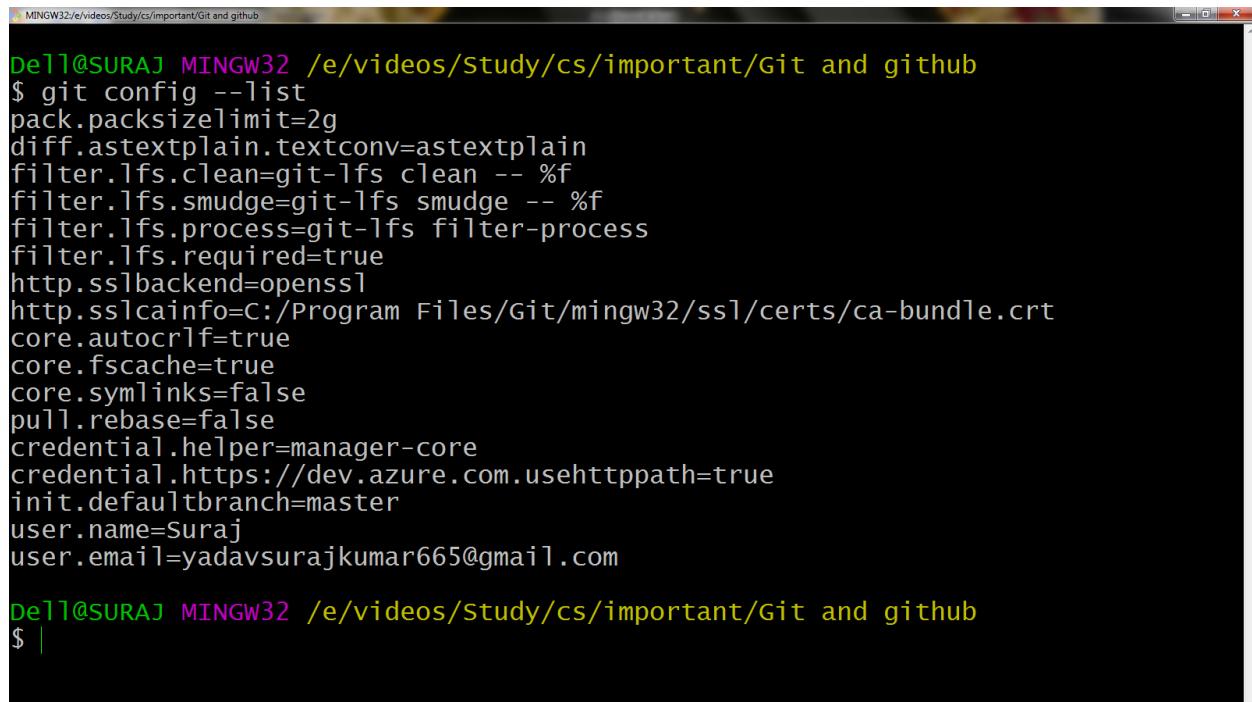


```
MINGW64:/c/Users/Haris/Desktop/code playground/Tuts/Git tutorials
$ git config --global user.name "Harry"

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/code playground/Tuts/
Git tutorials
$ git config --global user.email "harry@codewithharry.com"

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/code playground/Tuts/
Git tutorials
$ git config --list
```

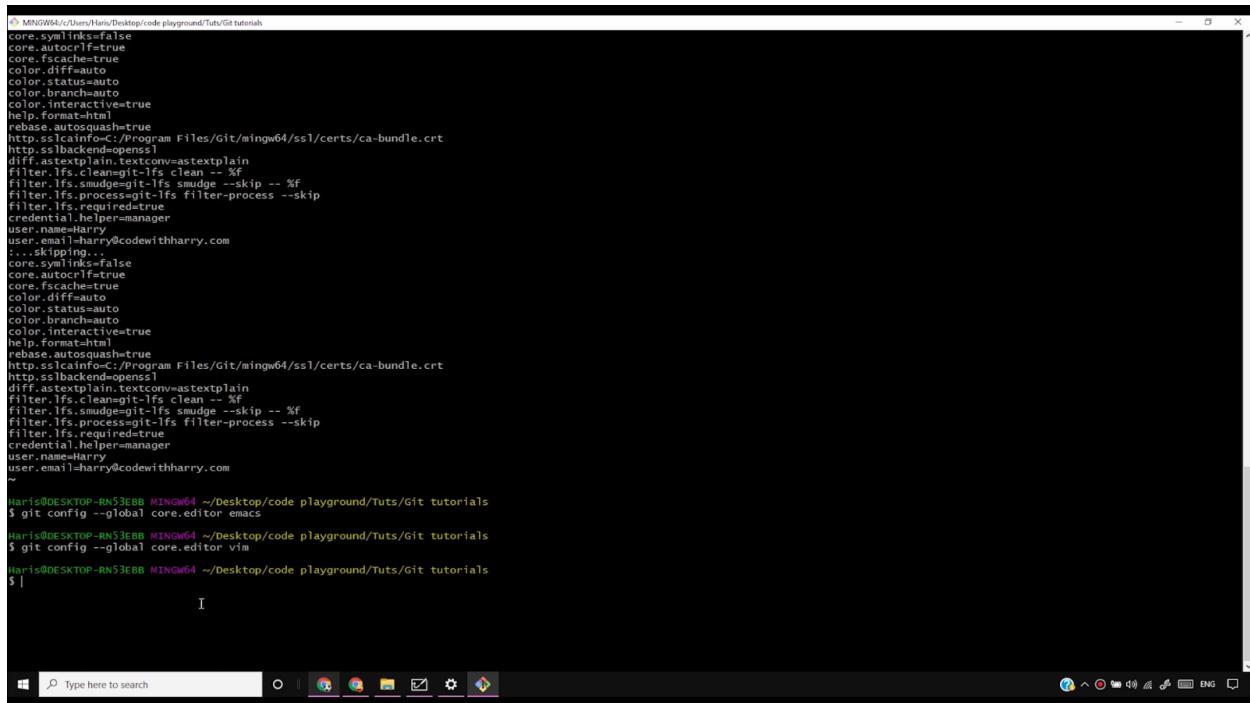
and last command list the data



```
MINGW32:/e/videos/Study/cs/important/Git and github
$ git config --list
pack.packsize=limit=2g
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:/Program Files/Git/mingw32/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Suraj
user.email=yadavsurajkumar665@gmail.com

Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github
$ |
```

Following are some editors of git

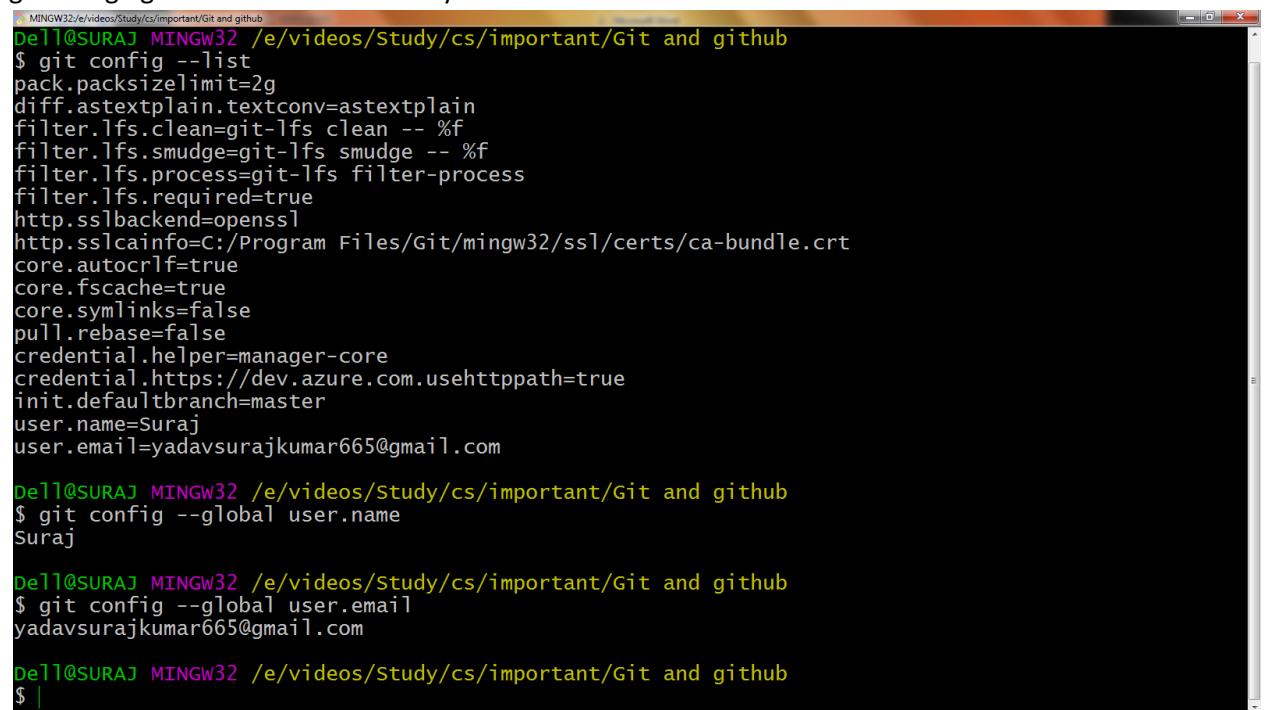


```
MINGW64:/Users/Harry/Desktop/code playground/Tuts/Git tutorials
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge --skip
filter.lfs.process=git-lfs filter-process --skip
filter.lfs.required=true
credential.helper=manager
user.name=Harry
user.email=harry@codewithharry.com
user.url=https://codewithharry.com
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge --skip
filter.lfs.process=git-lfs filter-process --skip
filter.lfs.required=true
credential.helper=manager
user.name=Harry
user.email=harry@codewithharry.com
~

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/code playground/Tuts/Git tutorials
$ git config --global core.editor emacs
Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/code playground/Tuts/Git tutorials
$ git config --global core.editor vim
Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/code playground/Tuts/Git tutorials
$ |
```

How to check name and email

- `git config --global user.name` → tells your name
- `git config --global user.email` → tells your email id



```
Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github
$ git config --list
pack.packsizelimit=2g
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:/Program Files/Git/mingw32/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Suraj
user.email=yadavsurajkumar665@gmail.com

Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github
$ git config --global user.name
Suraj

Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github
$ git config --global user.email
yadavsurajkumar665@gmail.com

Dell@SURAJ MINGW32 /e/videos/Study/cs/important/Git and github
$ |
```