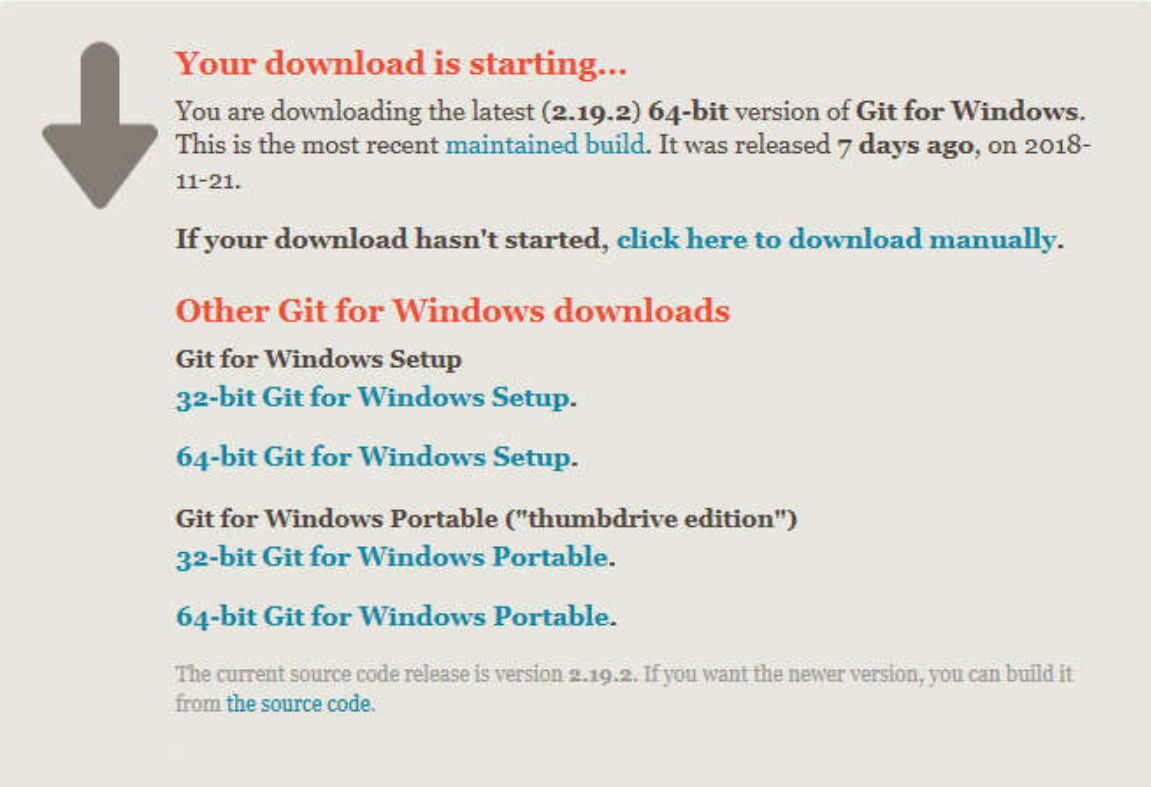
Version Control Systems - Git

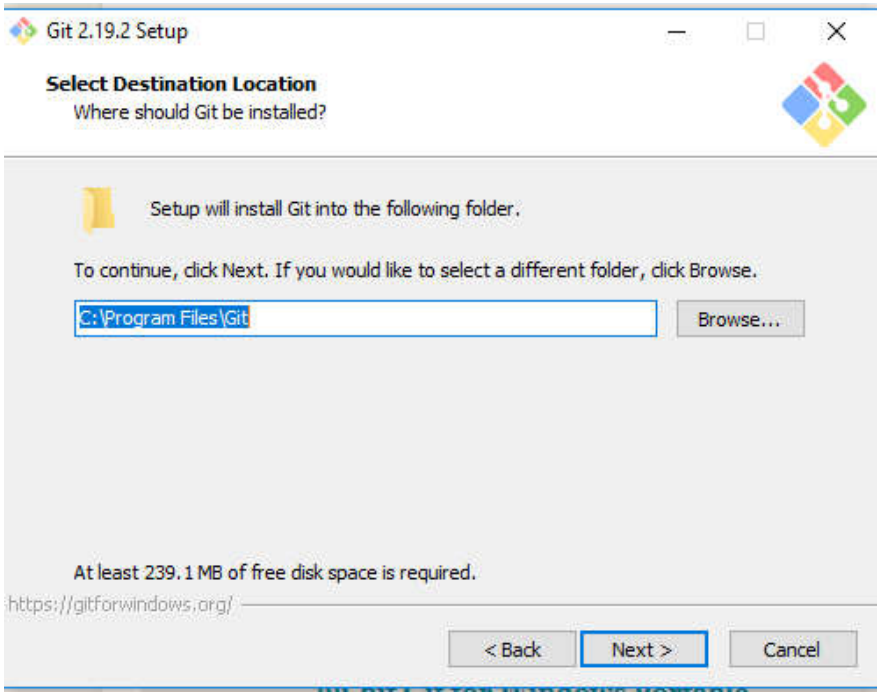
Murat Caliskan

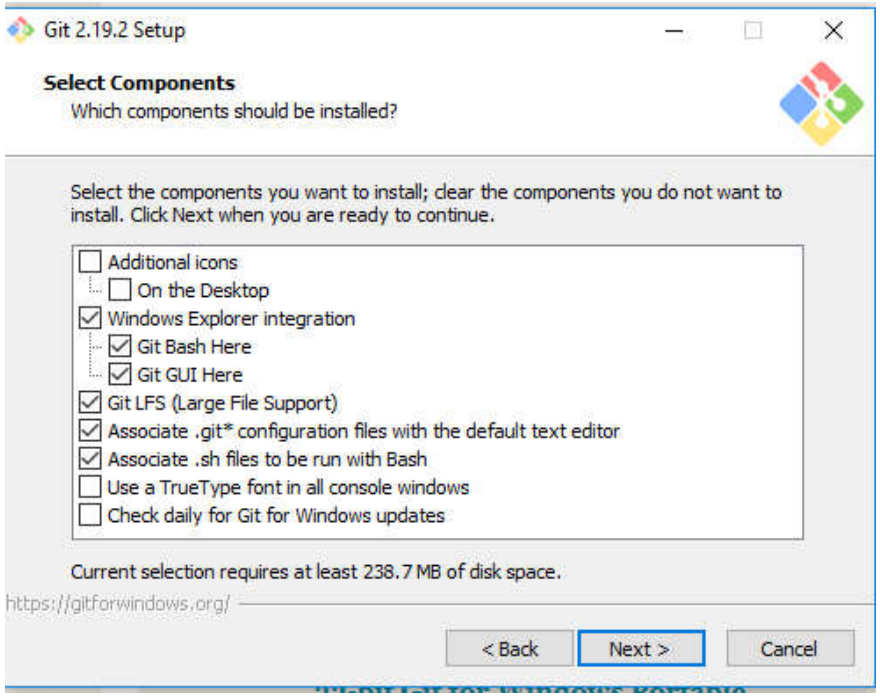
Git Installation for Windows

- https://git-scm.com/download/win

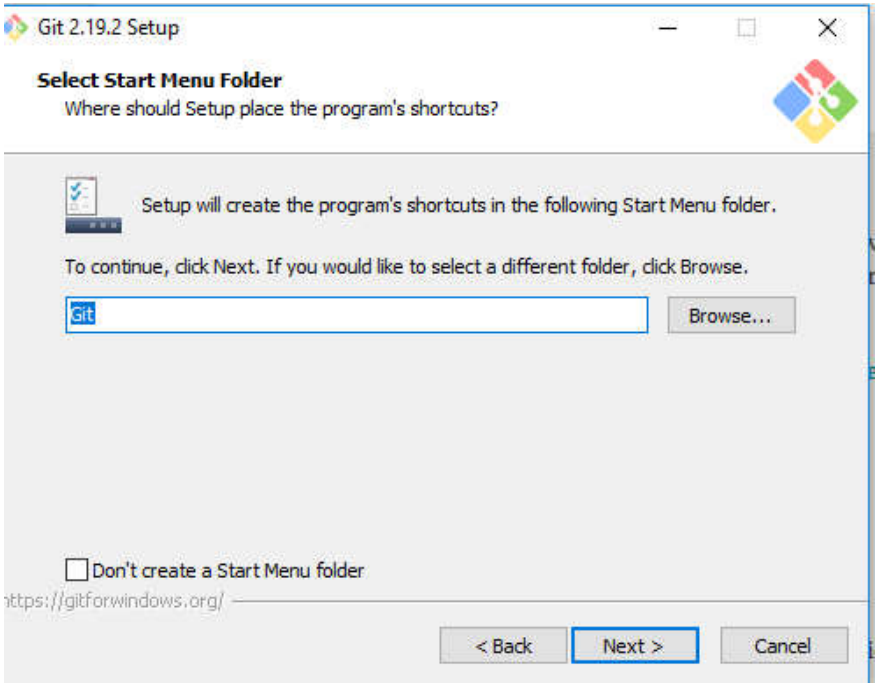


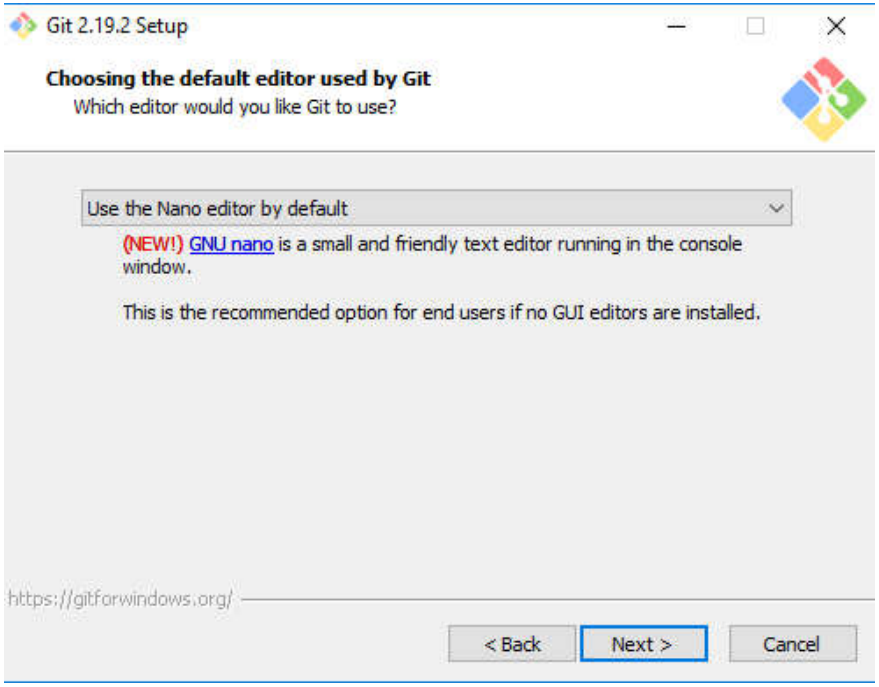
Git Installation for Windows



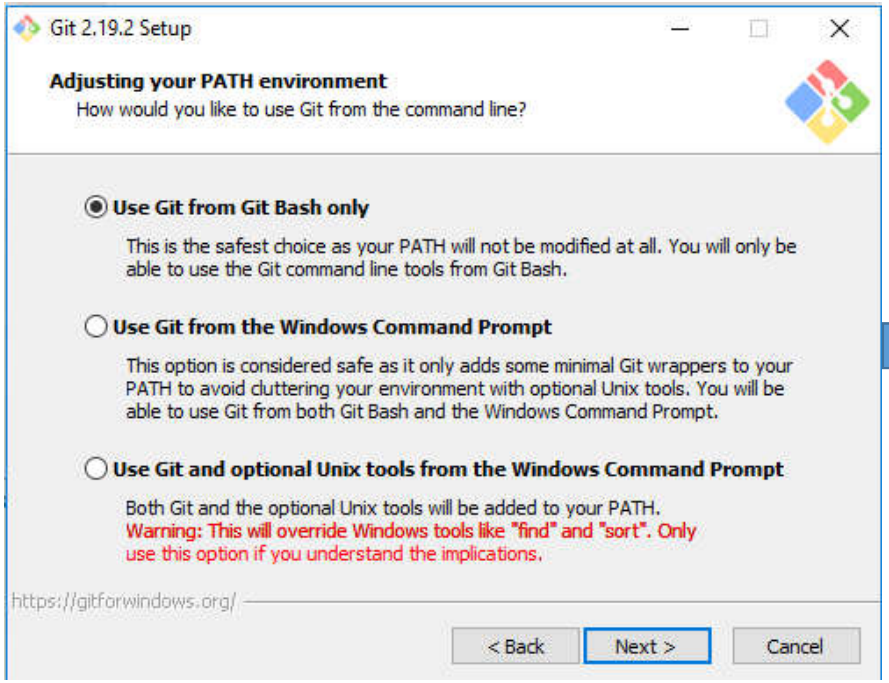


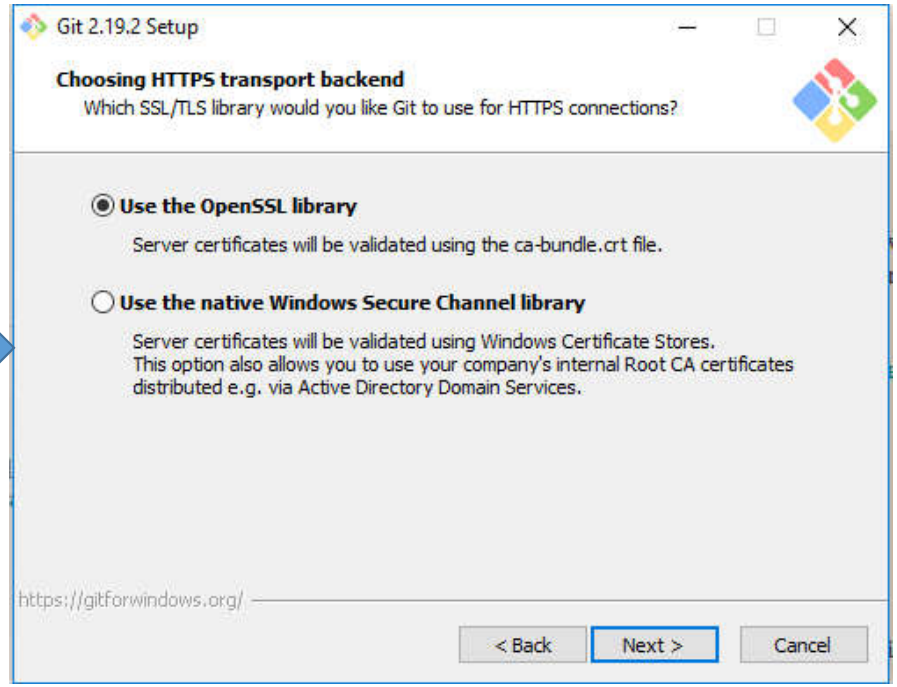
Git Installation for Windows



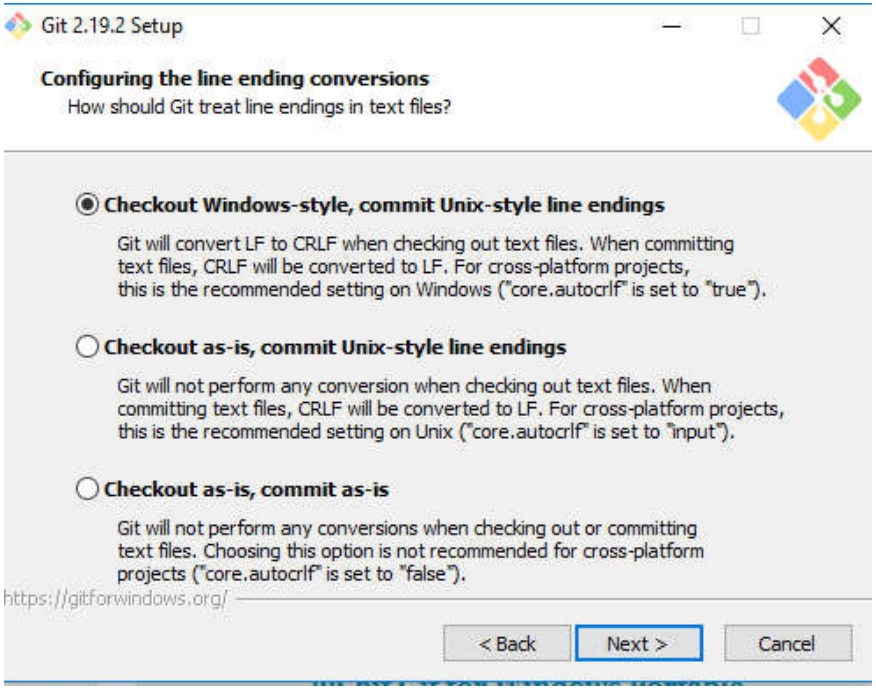


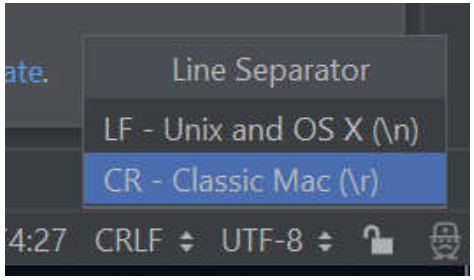
Git Installation for Windows



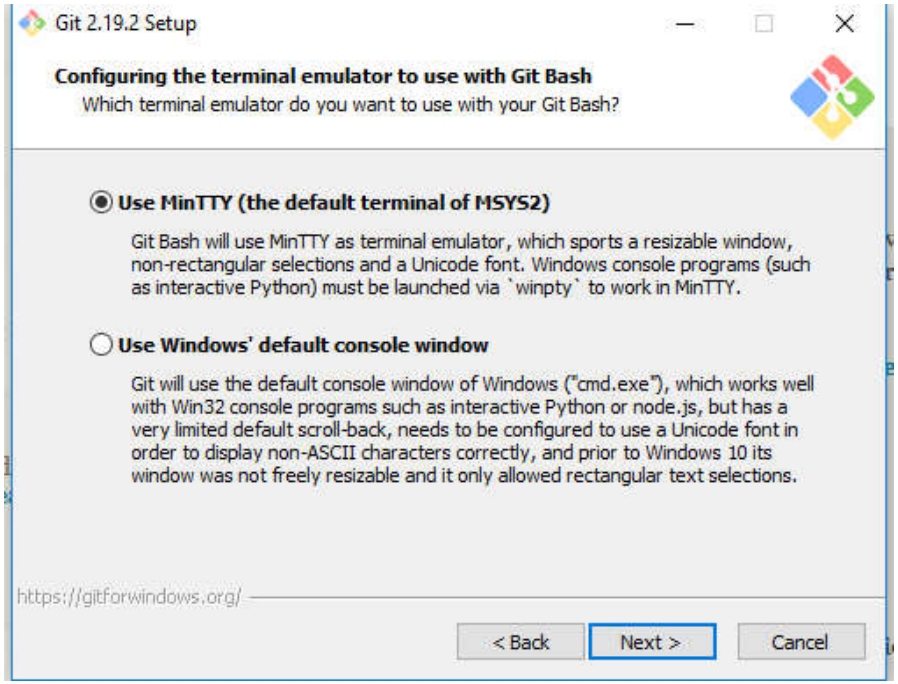


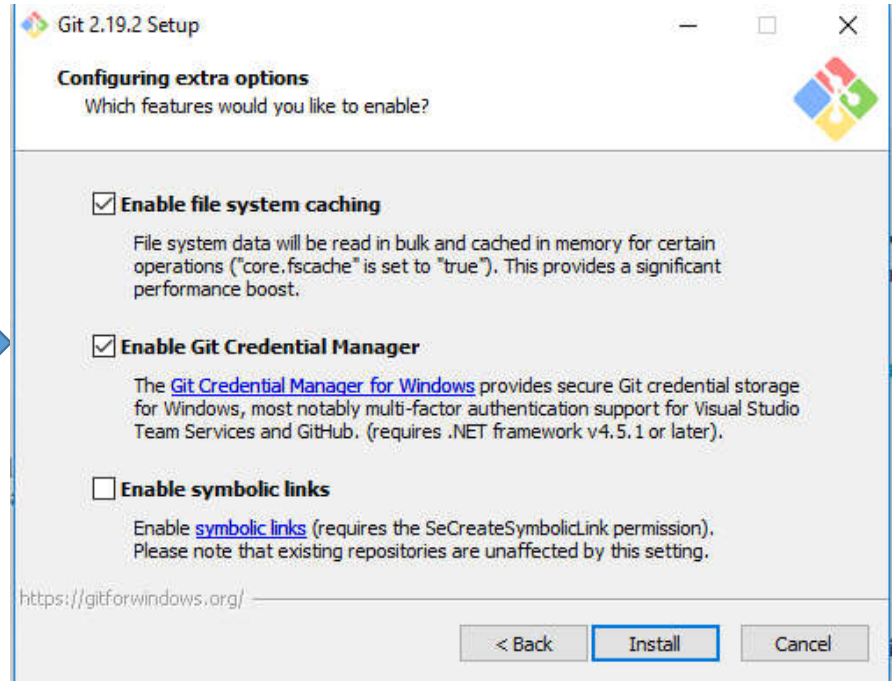
Git Installation for Windows



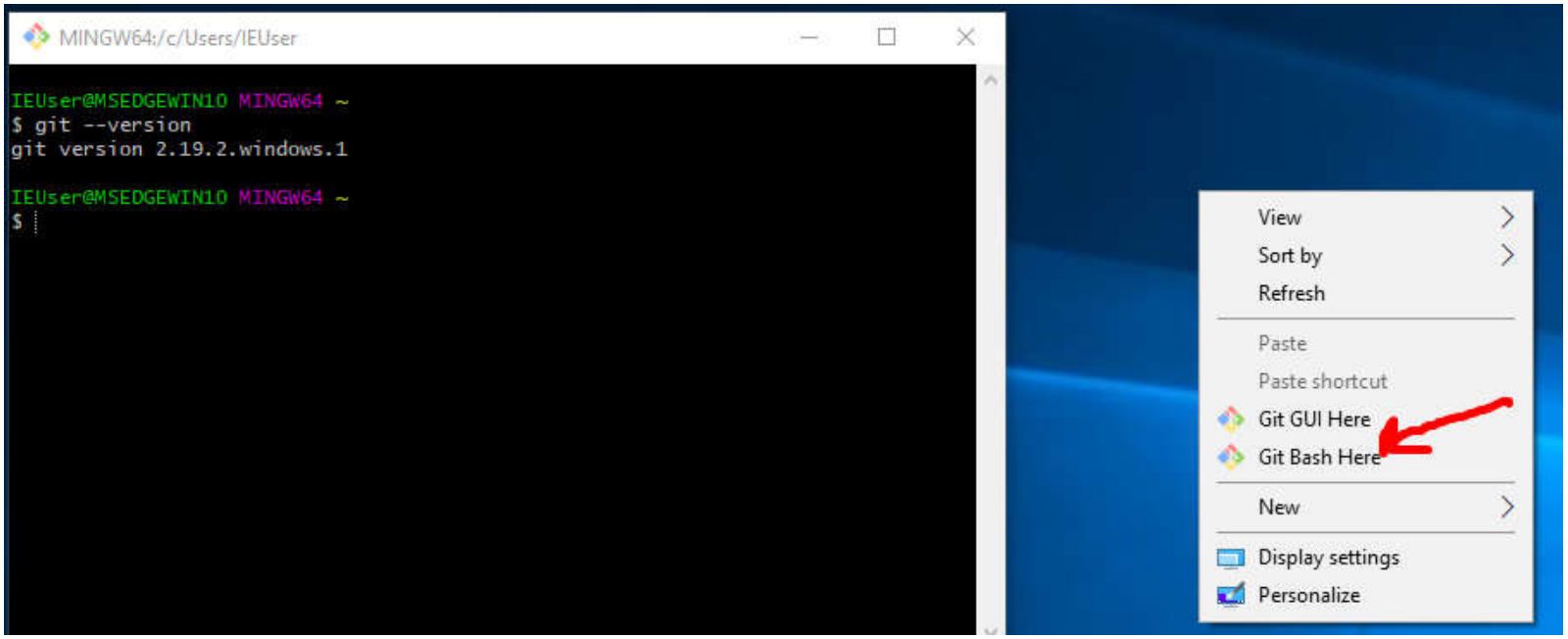


Git Installation for Windows





Git Installation for Windows



Git Installation for Mac

- Enter terminal
- Write **git** at terminal and verify if it was pre-installed
- If it writes “command not found ”
  - Download https://git-scm.com/download/mac
  - Install straight way.
  - Check with **git version** if installed

Git Installation for Linux (Ubuntu)

- Enter terminal
- Write **git version** at terminal and verify if it was install
- If it writes “program git is currently not installed”
  - Visit https://git-scm.com/download/linux
  - Check related installation guide for your linux operating system
  - For Ubuntu: **sudo apt-get install git**
    - **Sudo:** Gives administrative rights. It will ask your admin password.
  - Check with **git version** if installed

Step 2 = Learning Concept of VCS

- VCS = Version control system.
- Version control system: is a software that can store and maintain information about each and every change made to files and directories brought under it's control against specific version.

- **Question: What does it mean version?**
  - When you create a file which is monitored by the VCS we can say that it is version 1.
  - When you modify the file than it would be different from original so we can say it is version 1.1

Step 2 = Learning Concept of VCS

- VCS records history of all changes to a file against version numbers.
- This is also called commit history. You can change the file to a particular version in commit history at any time you want
- You can also revert entire projects not just files previous state.

- **Question: Why we may need to revert entire projects?**
  - If you are doing some recent changes (experimenting a big refactoring to improve performance etc) and it has created various bugs or some other problems then you can simply discard all changes and go back to beginning.

Step 2 = Learning Concept of VCS

- If you are working with many developers using VCS you can find:
  - Who did change?
  - What change was it?
  - When was that change?
- You can compare different versions of file or files.
- **Question: What its use to compare different versions?**
  - Can help for debugging purpose.
  - For solo developer it is good for memorizing if many changes were done.

Step 2 = Learning Concept of VCS

- There are 2 types of VCS
  - Centralized Version Control Systems
    - Example: CVS, Perforce, Subversion
  - Distributed Version Control Systems
    - Example: Git, Mercurial

- Centralized Version Control Systems
    - Single central server
    - Checkout only from that central server
- **Question: What is the Problem of Centralized Version Control Systems**
    - **When server goes down. No more check outs.**
    - **No Back up servers. Commit history is lost.**

Step 2 = Learning Concept of VCS

- Distributed Version Control Systems
  - When client checks out a repository (code base) from a server, it fully mirrors that repository.
  - **So if the server goes down.** No problem we have all history at local computer.

Step 2 = Learning Concept of VCS

- **Server:** Version control software running on a server grade machine that hosts or control & manages the repositories.
- **Client:** Software which communicates to the server. Mostly our local desktop software. Either command line or GUI interface (from Intellij)
- **Communication:** Accessing the repository for writing or reading files.
- **Repository:** Large collection of source code hosted on a remote server. Can be publicly or privately accessed.
  - Public access: Free of all.
  - Private access: Selected group of individuals with some kind of access control

Git

- Created in 2005 by Linus Torvalds. Father of Linux.
- Linux project was using Bitkeeper as version control.
  - Because it turned into a paid model, Linux community rioted.
- Now it is most popular DVCS in the world.

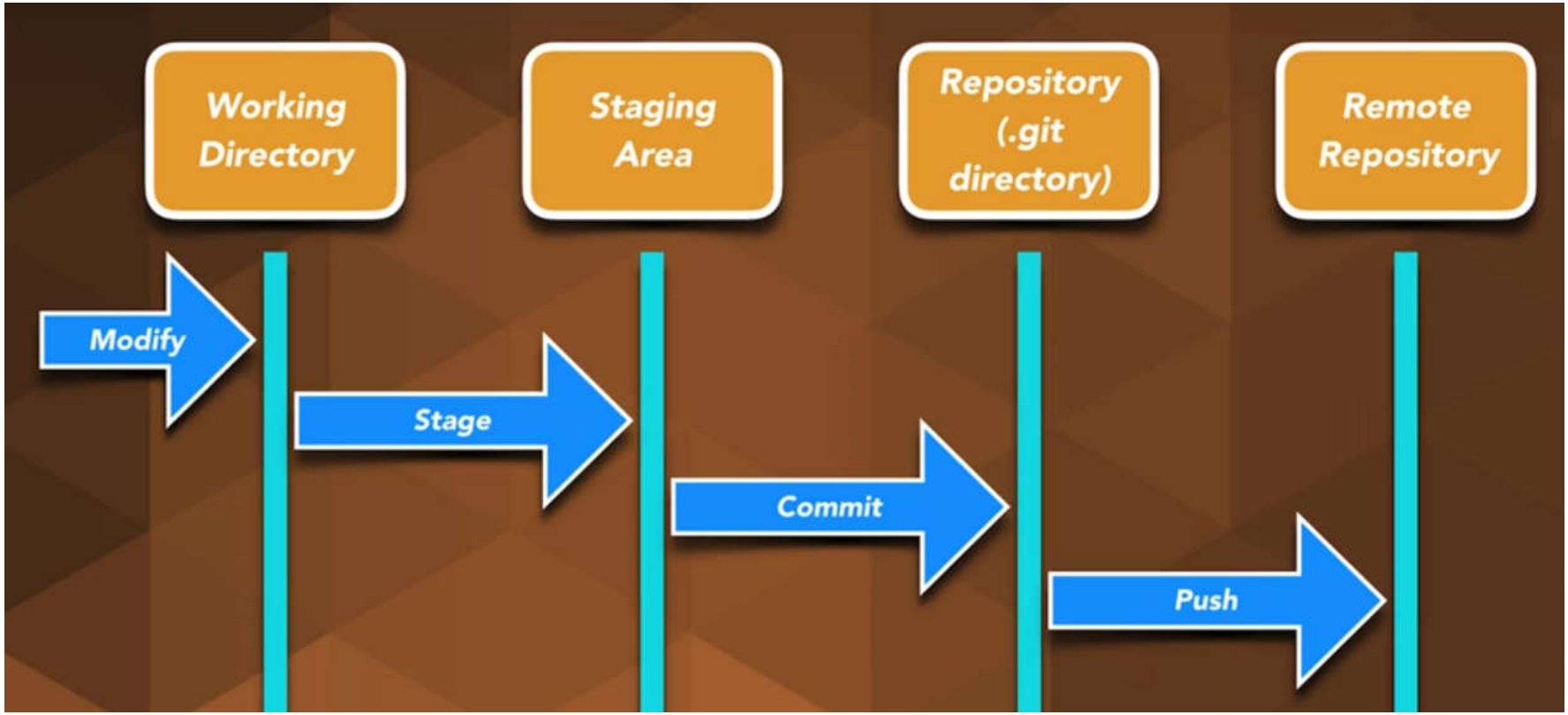
Git States

- Every files goes through **Three** states if you want to add to repository.
  - **Modified state:** In this state file gets modified and saved in the file system under the operating system that the user is using.
    - This is changes that are **NOT COMMITED** yet to git database.
  - **Staged state:** In this stage a modified file is marked in its
    - **Git add** command will be used.
  - **Commited state:** In the sate which the file is stored in the local Git Database.
    - **Git commit** command will be used.

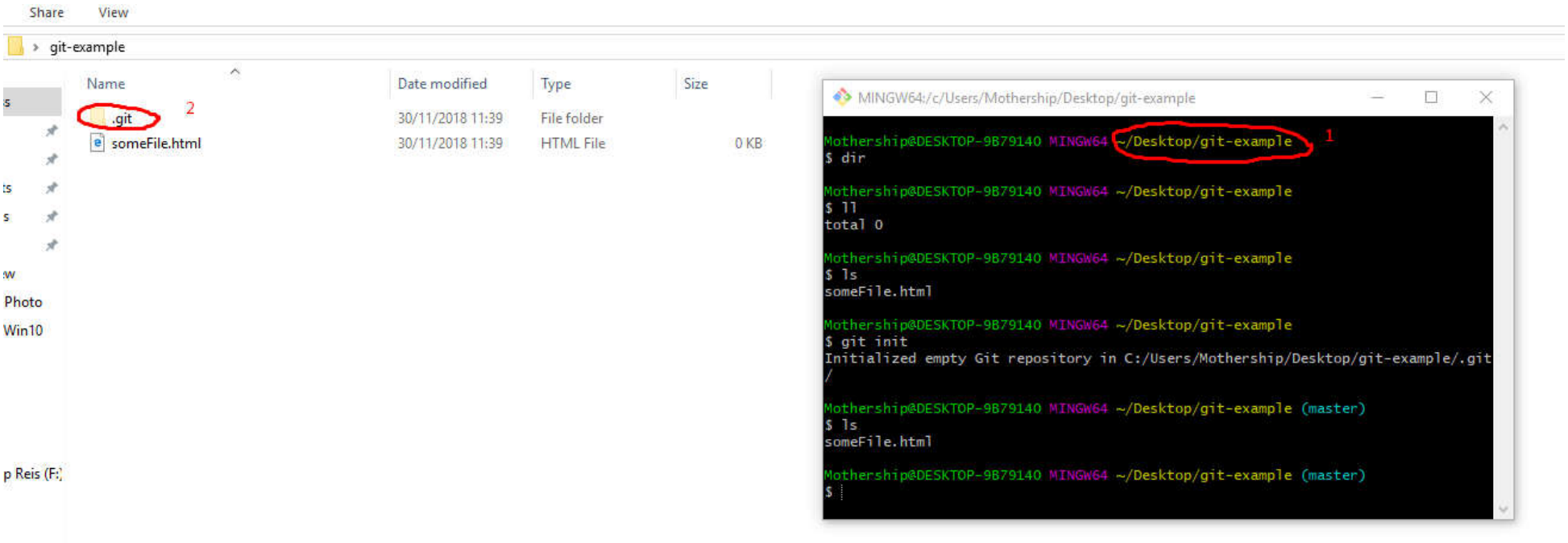
Git States

- Every files goes through **Three** states if you want to add to repository.
  - **Modified state:** In this state file gets modified and saved in the file system under the operating system that the user is using.
    - This is changes that are **NOT COMMITED** yet to git database.
  - **Staged state:** In this stage a modified file is marked in its
    - **Git add** command will be used.
  - **Commited state:** In the sate which the file is stored in the local Git Database.
    - **Git commit** command will be used.

Git States

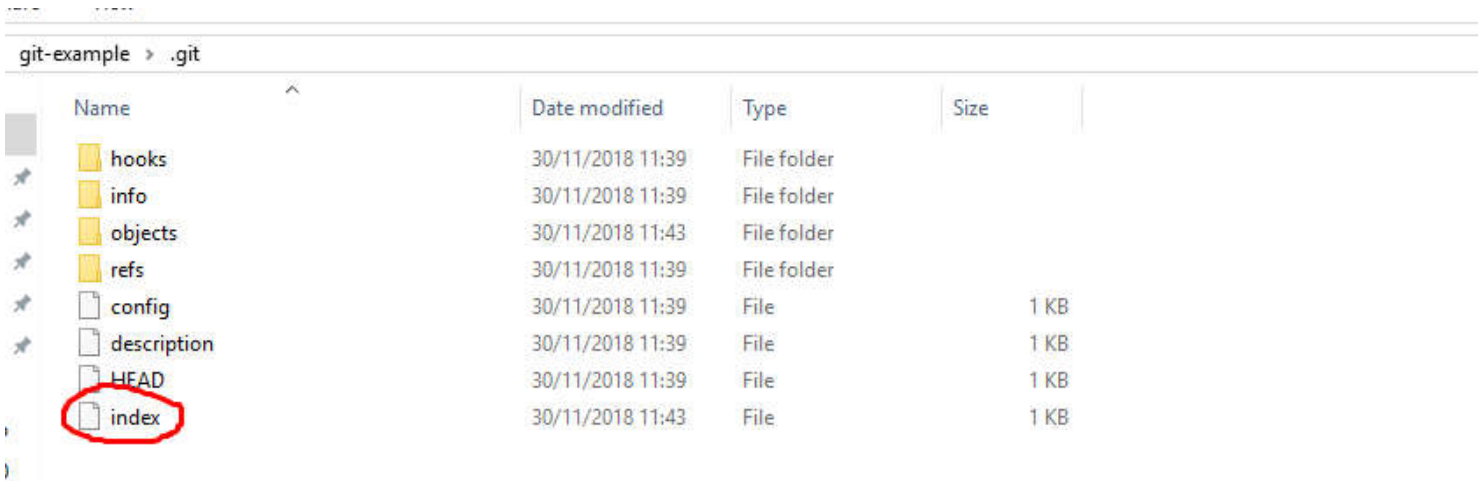

Working Directory
Staging Area
Repository (.git directory)
Remote Repository
Modify
Stage
Commit
Push

Git States



- 1 is the working directory where you create and modify files.

Git States



- The file “index” represents the index or staging area in Git. It contains meta-data such as timestamps, file names etc.

Git Repositories

- How to create?
  - From scratch by executing the “git init” command in an empty folder.
  - From existing project by executing “git init” command
  - Cloning from remote repository from a remote server. Bitbucket or Github.

Git Commands – git config

- Git help => Help section of how to use git
- Setup name and email address for Git

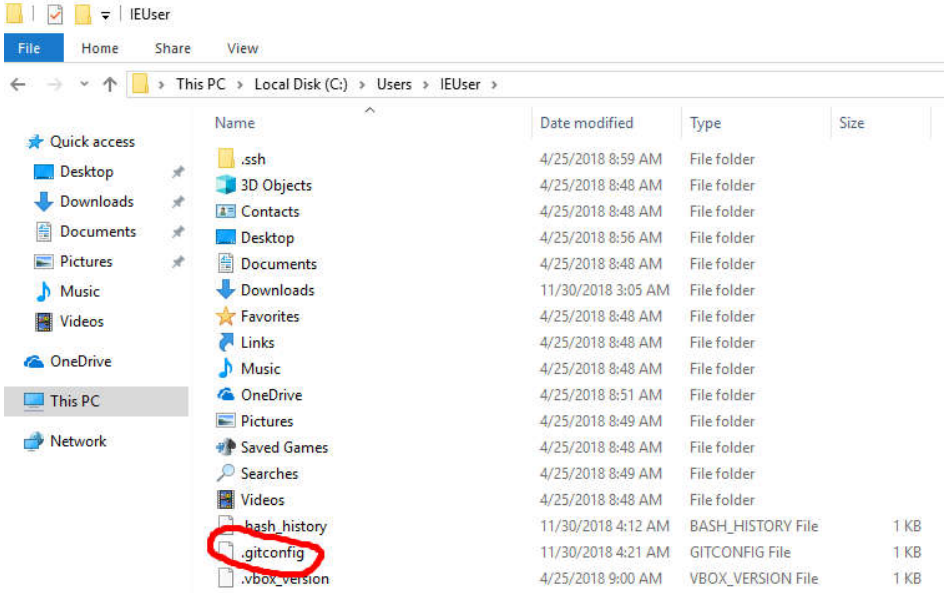
MINGW64:/c/Users/IEUser

```
IEUser@MSEDGEWIN10 MINGW64 ~
$ git config --global --list
fatal: unable to read config file 'C:/Users/IEUser/.gitconfig': No such file or
directory
```

```
IEUser@MSEDGEWIN10 MINGW64 ~/Desktop
$ git config --global user.name "Murat Caliskan"

IEUser@MSEDGEWIN10 MINGW64 ~/Desktop
$ git config --global user.email "myemail@yahoo.com"

IEUser@MSEDGEWIN10 MINGW64 ~/Desktop
$ git config --global --list
user.name=Murat Caliskan
user.email=myemail@yahoo.com
```

Git Commands – git config

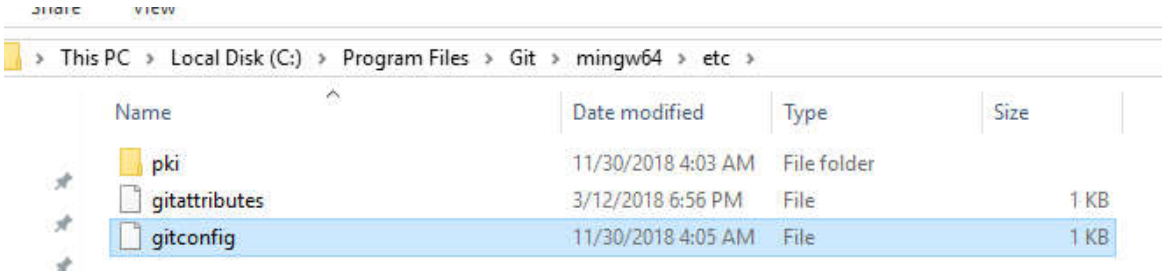
- Global Configuration file: Configuration values are applied to single user.



Git Commands – git config

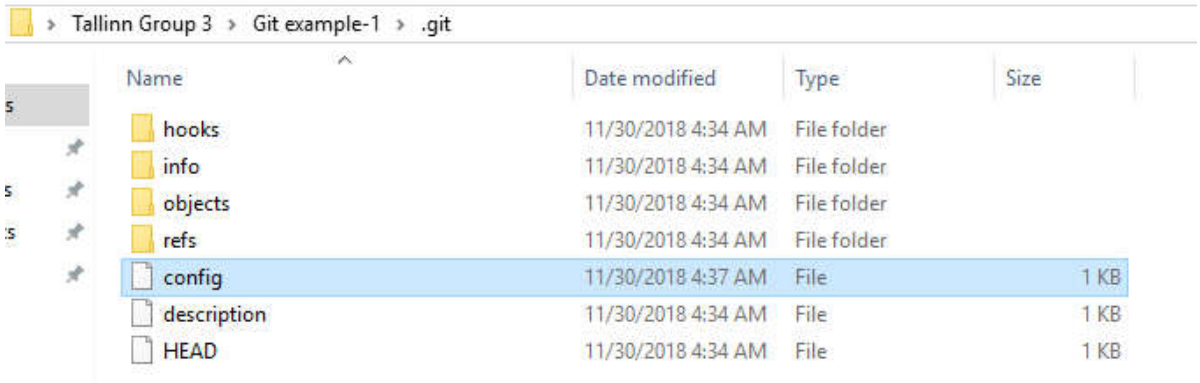
- System Configuration file: Configuration values are applied to all users.

```
IEUser@MSEDGEWIN10 MINGW64 ~/Desktop
$ git config --system --list
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
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
credential.helper=manager
core.editor=nano.exe
```

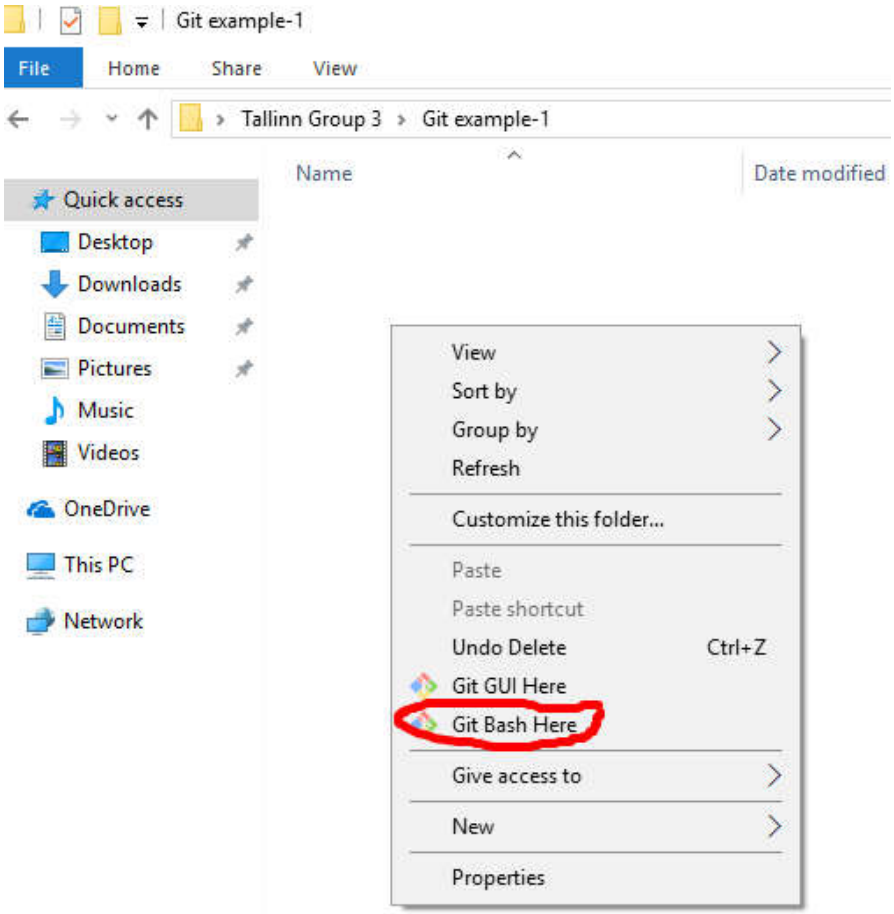


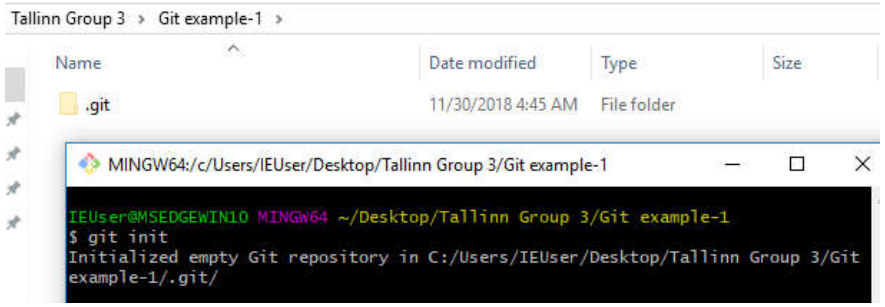
Git Commands – git config

- Local Configuration file: Configuration values are applied to a single repository. This file can overwrite configuration values set in Global and System

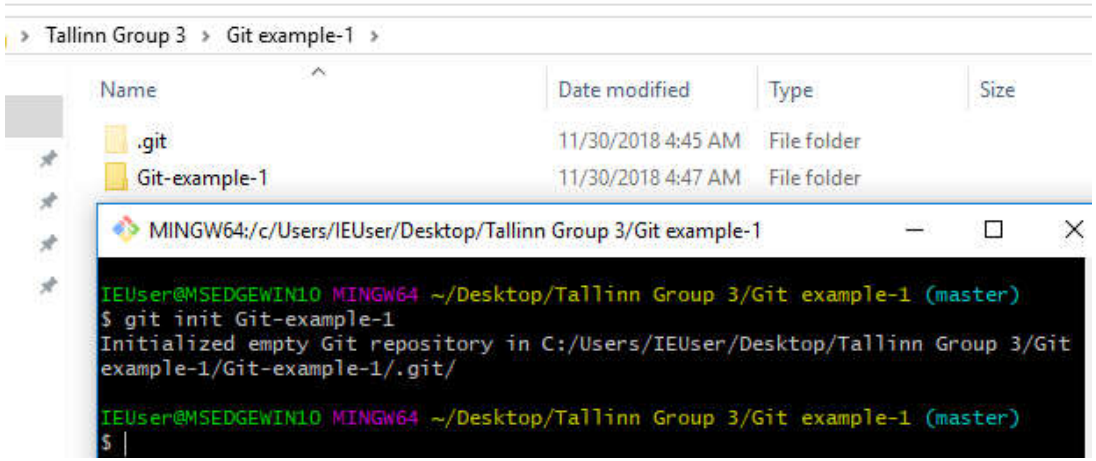


Git Commands – git init (for empty folder)



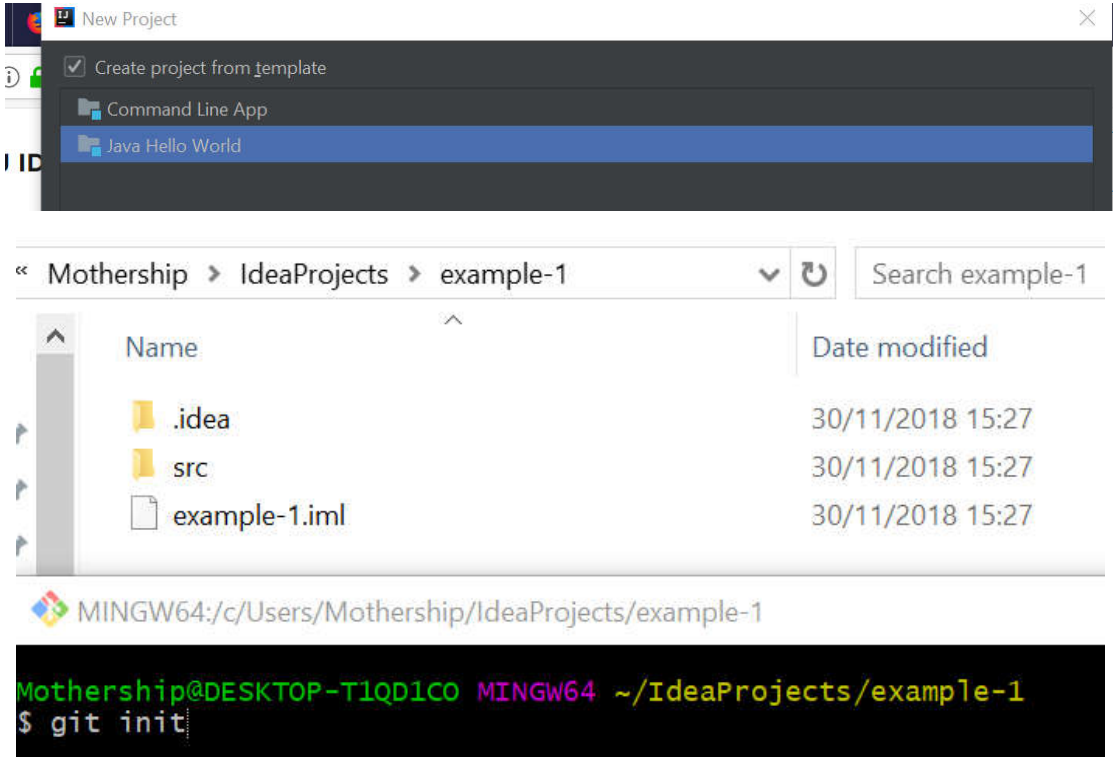


Git Commands – git init



- Git init <folder name>
  - Will create the git repository with the folder name

Git Commands – git init (for existing project)



- **Create a sample project**
- **Init Git to that project**

Git Commands – git add

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git add newFile.txt

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master

No commits yet

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

        new file:   newFile.txt

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

        .idea/
        example-1.iml
        src/
```

- Create a text file in the sample project
- Add the file for the git to control
- Check status if it is really ready to be committed.
- **Git status** command is used to check status.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git add src

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master

No commits yet

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

        new file:   newFile.txt
        new file:   src/Main.java
```

- You can also add folder and all its content.
- If you want all the files and folders:
  - “Git add .” command will work.

Git Commands – git commit

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git commit -m "Initial Commit"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Mothership@DESKTOP-T1QD1CO.(none)')
```

- Commit the files which we have added using the **git add** command
- If you did not do
  - git config --global user.email
  - git config --global user.name
- Left side error will happen.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git commit -m "Initial Commit"
[master (root-commit) f992746] Initial Commit
 12 files changed, 303 insertions(+)
 create mode 100644 .idea/compiler.xml
 create mode 100644 .idea/description.html
 create mode 100644 .idea/encodings.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .idea/workspace.xml
 create mode 100644 example-1.iml
 create mode 100644 newFile.txt
 create mode 100644 src/Main - Copy - Copy.java
 create mode 100644 src/Main - Copy.java
 create mode 100644 src/Main.java
```

- Successful commit
- 303 => is line amount
- F992746 => commit id. Identifies the commit.
- Initial Commit => our commit message
- 12 files => affected file amount

Git Commands – git status

```
othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
 git status
n branch master
hanges not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   .idea/workspace.xml
       modified:   newFile.txt

o changes added to commit (use "git add" and/or "git commit -a")
```

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git commit -m "another commit"
On branch master
Changes not staged for commit:
        modified:   .idea/workspace.xml
        modified:   newFile.txt

no changes added to commit
```

- Gives the status of your repository
- We changed newFile.txt
- Even though we did commit it does not insert to git repository as it is not staged.
- You must first do git add newFile.txt
- Then commit.
- In intellij it does both actions with one click.

Git Commands – Reverting changes.

```
othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
 git status
n branch master
hanges not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/workspace.xml
        modified:   newFile.txt

o changes added to commit (use "git add" and/or "git commit -a")

othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
 git checkout newFile.txt

othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
 git status
n branch master
hanges not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/workspace.xml

o changes added to commit (use "git add" and/or "git commit -a")

othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
```

- Git checkout : Gets the file from the git repository.

Git Commands – Reverting changes.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   newFile.txt

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

        modified:   .idea/workspace.xml


Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git reset head newFile.txt
Unstaged changes after reset:
M       .idea/workspace.xml
M       newFile.txt

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/workspace.xml
        modified:   newFile.txt

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

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$
```

- **Git reset Head command**: In case you want to revert file which you did “git add” command.
- It will take the file back to the previous state.

Git Commands – Reverting changes.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   anotherFile.txt

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

        modified:   .idea/workspace.xml


Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git reset Head anotherFile.txt
Unstaged changes after reset:
M       .idea/workspace.xml

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/workspace.xml

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

        anotherFile.txt

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

- **Git reset Head command**: In the case of newly added file same command can be used to take the file to previous state.
- **Warning:** If you just delete the file after you did “git add”; the file will still be added if you do **“git commit”**

Git Commands – Reverting changes.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   anotherFile.txt

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

        modified:   .idea/workspace.xml
        deleted:    anotherFile.txt


Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git add anotherFile.txt

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/workspace.xml

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

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$
```

- Another option is to remove the change
  - delete the file from file system
  - “git add <filename>”.

Git Commands – Git log

```
lothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git log
commit 492eecdeffe018c03bf84b3bc57a621a331c9695 (HEAD -> master)
Author: Murat <someEmail>
Date:   Fri Nov 30 16:31:14 2018 +0200

    some more commit

commit 32670b65d611d04e7b4596f0d78017c0f34f97d3
Author: Murat <someEmail>
Date:   Fri Nov 30 16:29:08 2018 +0200

    interesting commit

commit cc6885e5ac832b728b4731c770fd92c388bbeba5
Author: Murat <someEmail>
Date:   Fri Nov 30 16:10:52 2018 +0200

    another commit

commit 09c755d7c5790d2034fc365df6fc471345fa2d88
Author: Murat <someEmail>
Date:   Fri Nov 30 16:10:00 2018 +0200

    Commit

commit f99274668e68bec7e5c884f2d8672e129d16e949
Author: Murat <someEmail>
Date:   Fri Nov 30 15:50:43 2018 +0200

    Initial Commit
```

- This command shows the commit history
- **Git log –oneline** = for summary look
- **Git log –n 3**= last 3 commit history
- **Git log <filename>** = file commit history

Git Commands – Git stash

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

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

        modified:   .gitignore
        modified:   .idea/workspace.xml

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

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git stash
Saved working directory and index state WIP on master: ef343c5 new commit is here

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

- In scenario for emergency fixes we may need to stash.
- Or we need to rebase our branch we may be requested for stash or commit.
- Only tracked files.
- Git stash –u : will include untracked files
- Git stash pop : applies the stash and drop the stash.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git stash list
stash@{0}: WIP on master: ef343c5 new commit is here
stash@{1}: WIP on newBranchPlease: c7249df another add
stash@{2}: WIP on newBranchPlease: c7249df another add
stash@{3}: WIP on newBranchPlease: c7249df another add
```

Git Commands – Git stash

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git stash apply
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

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

        modified:   .gitignore
        modified:   .idea/workspace.xml

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

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git stash pop
Auto-merging asda
CONFLICT (content): Merge conflict in asda
The stash entry is kept in case you need it again.
```

- Git stash apply to get back the changes.
- Git stash drop to erase the applied stash.
- If you do change on same file after stash. When poping back it will create conflict.

Git Commands – Git tag

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (newBranchPlease)
$ git log --oneline
79fa2a4 (HEAD -> newBranchPlease, master) insert
ef343c5 new commit is here
c7249df another add
b572302 yes
d3671a9 (origin/master) Create README.md
d4e4fe2 new adding for new feature branch
492eecd some more commit
82670b6 interesting commit
cc6885e another commit
09c755d Commit
f992746 Initial Commit

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (newBranchPlease)
$ git tag v1.0 ef343c5

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (newBranchPlease)
$ git log --oneline
79fa2a4 (HEAD -> newBranchPlease, master) insert
ef343c5 (tag: v1.0) new commit is here
c7249df another add
b572302 yes
d3671a9 (origin/master) Create README.md
d4e4fe2 new adding for new feature branch
492eecd some more commit
82670b6 interesting commit
cc6885e another commit
09c755d Commit
f992746 Initial Commit
```

- Git tag <tagName> : it is just a marker on a commit.

Git ignore file

- Some folders and files are not meant for Git.
- https://www.atlassian.com/git/tutorials/saving-changes/gitignore
- .gitignore file is responsible for this need.

Git branching

- Whenever you create a repository in Git, a branch called master is created automatically.
- Branches are simply a lightweight movable pointer to a commit you have made in your repository.
- Different branches can have different versions of the files. You can add and remove from specific branches.

Git branching

Why it is good to have branches:

- Good for developers to update them anytime they like.
- Freedom to experiment without other developers.
- Companies policies for “one feature one branch”
- Master is considered production ready deployment.
- Fixing of the major bugs while other developers do new features.

Git branching

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git branch
* master

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git branch new-feature

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git branch
* master
  new-feature
```

- Git branch <branchName> : will create the branch.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git checkout new-feature
Switched to branch 'new-feature'
M       .idea/workspace.xml

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$
```

- Git checkout <branchName> : will check out the branch. All changes will be effecting this branch

Git branching

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$ git add newFile.txt

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$ git status
On branch new-feature
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   newFile.txt

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

        modified:   .idea/workspace.xml


Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$ git commit -m "new adding for new feature branch"
[new-feature d4e4fe2] new adding for new feature branch
 1 file changed, 1 insertion(+), 1 deletion(-)

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$ git log --oneline
d4e4fe2 (HEAD -> new-feature) new adding for new feature branch
492eecd (master) some more commit
32670b6 interesting commit
cc6885e another commit
09c755d Commit
f992746 Initial Commit
```

- We modify the newFile.txt
- We add the change then commit.
- Now we can see branch will contain this change.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (new-feature)
$ git checkout master
Switched to branch 'master'
M       .idea/workspace.xml

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git log --oneline
492eecd (HEAD -> master) some more commit
32670b6 interesting commit
cc6885e another commit
09c755d Commit
f992746 Initial Commit
```

- We checkout back to the master
- As seen our latest change is not in master yet.
- It is located at the branch.

Branch merging & deletion

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git merge new-feature
Updating 492eecd..d4e4fe2
Fast-forward
 newFile.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
Mothershipp@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git log --oneline
d4e4fe2 (HEAD -> master, new-feature) new adding for new feature branch
492eecd some more commit
B2670b6 interesting commit
cc6885e another commit
D9c755d Commit
f992746 Initial Commit
```

- Git merge <branchname>: With the merge latest changes appear in the master branch.

```
Mothershipp@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git branch -d new-feature
Deleted branch new-feature (was d4e4fe2).

Mothershipp@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git branch
* master
```

- Git branch –d <branchname>: After your merge you can delete your branch.

Git Rebase

```
othership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
it log --oneline
f343c5 (HEAD -> master) new commit is here
7249df (newBranchPlease) another add
572302 yes
3671a9 (origin/master) Create README.md
4e4fe2 new adding for new feature branch
92eecd some more commit
2670b6 interesting commit
c6885e another commit
9c755d Commit
992746 Initial Commit
```

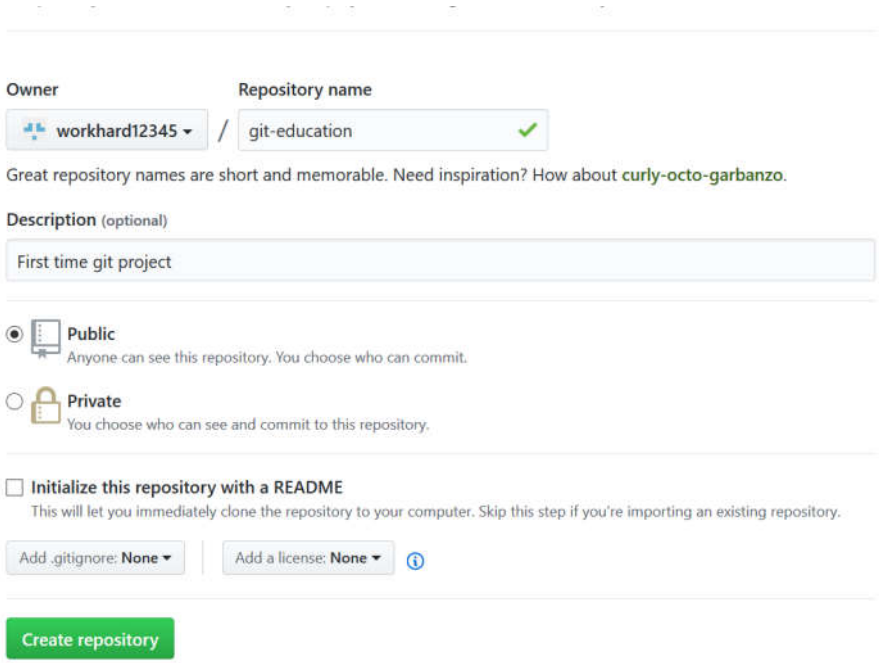
```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (newBranchPlease)
$ git rebase master
error: cannot rebase: You have unstaged changes.
error: Please commit or stash them.

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (newBranchPlease)
$ git status
On branch newBranchPlease
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore
        modified:   .idea/workspace.xml

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

- Lets suppose you have a branch
- A critical bug happened at master and it is fixed with the new commit
- Now you want to update your branch to the newest additions of the master
- **Note:** You have to stash or commit changes in your branch before rebasing.

- **Git stash:** temporary save changes

Github

- GitHub Inc. is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features.
- https://github.com/



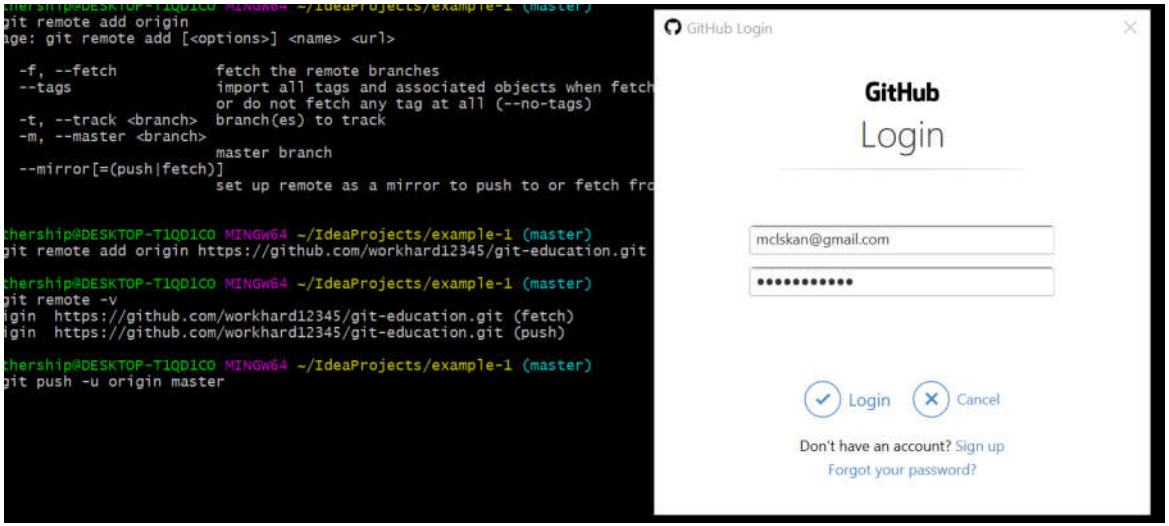
Git Remote

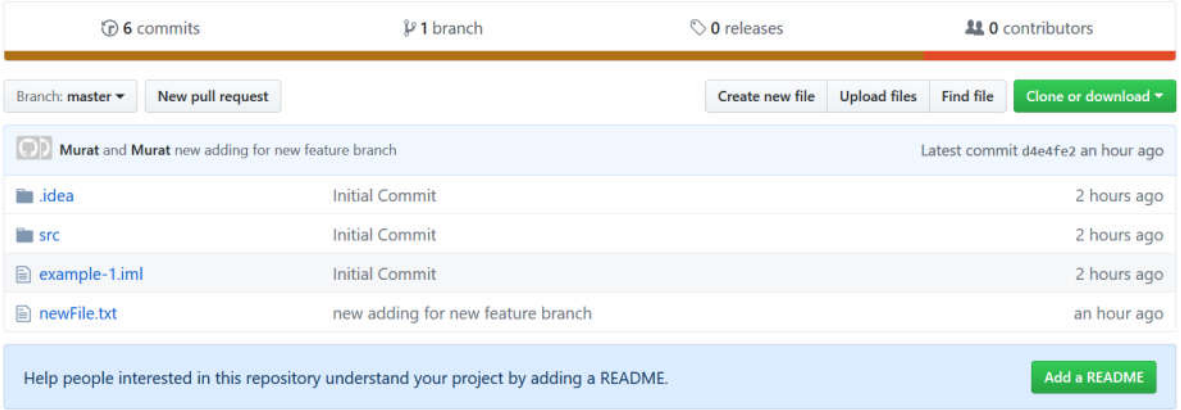
```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git remote -v

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$
```

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git remote add origin https://github.com/workhard12345/git-education.git

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git remote -v
origin  https://github.com/workhard12345/git-education.git (fetch)
origin  https://github.com/workhard12345/git-education.git (push)

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$
```

- Git remote: used to manage remote connections in git.
- Git remote add: adding new remote connection
- Origin: reference to the remote connection. It is tradition.

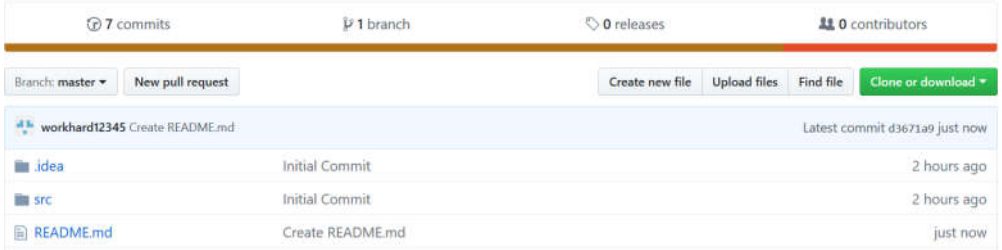
Git Push





- Git push –u origin master : Pushes the committed master at the remote repository.

Git Pull

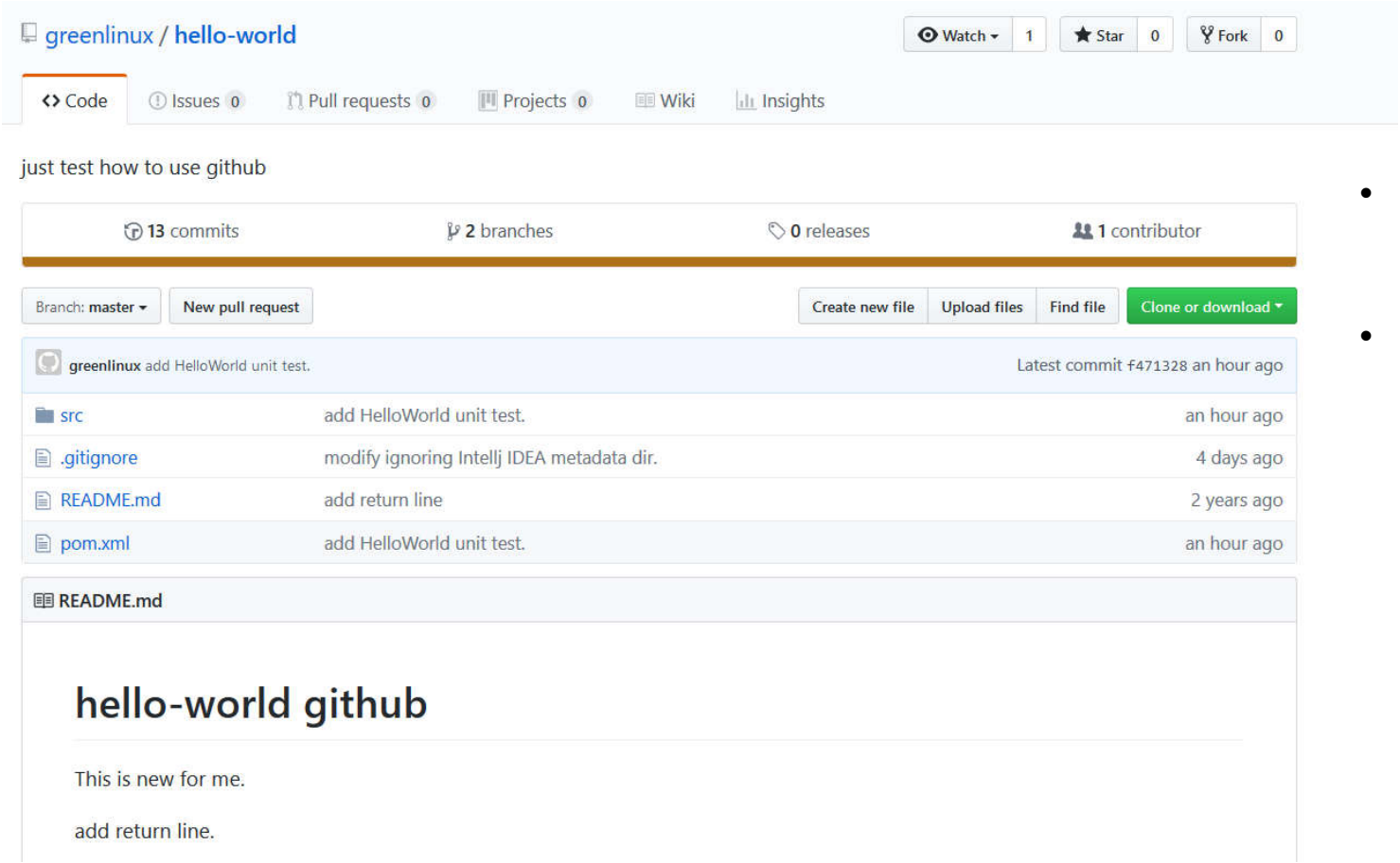


- Git pull: Get the latest changes from the remote repository
- **Warning:** First pull before a push. Maybe some developer pushed some changes which may break your changes.

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/workhard12345/git-education
   d4e4fe2..d3671a9  master     -> origin/master
Updating d4e4fe2..d3671a9
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$
```

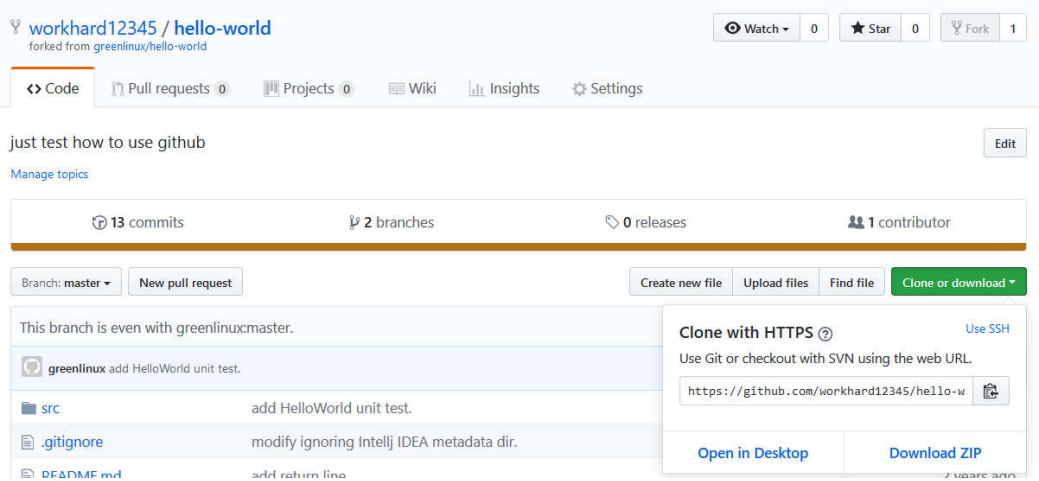
```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects/example-1 (master)
$ git log --oneline
d3671a9 (HEAD -> master, origin/master) Create README.md
d4e4fe2 new adding for new feature branch
492eecd some more commit
32670b6 interesting commit
cc6885e another commit
09c755d Commit
f992746 Initial Commit
```

Github Fork



- Creating personal repository from another users repository. A copy.
- Why we do forking:
  - Experimental purpose
  - Contribute to the original project

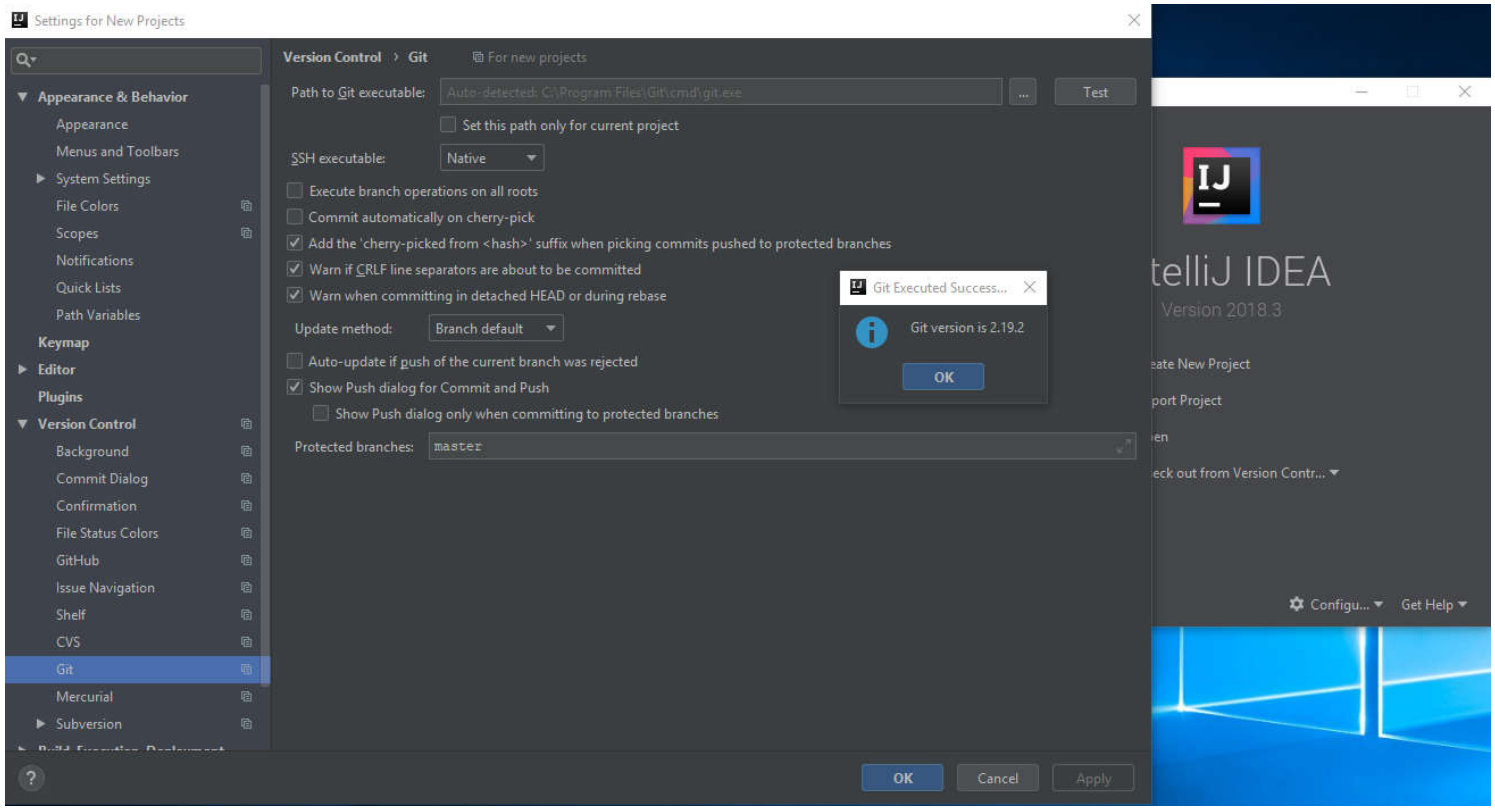
Git clone



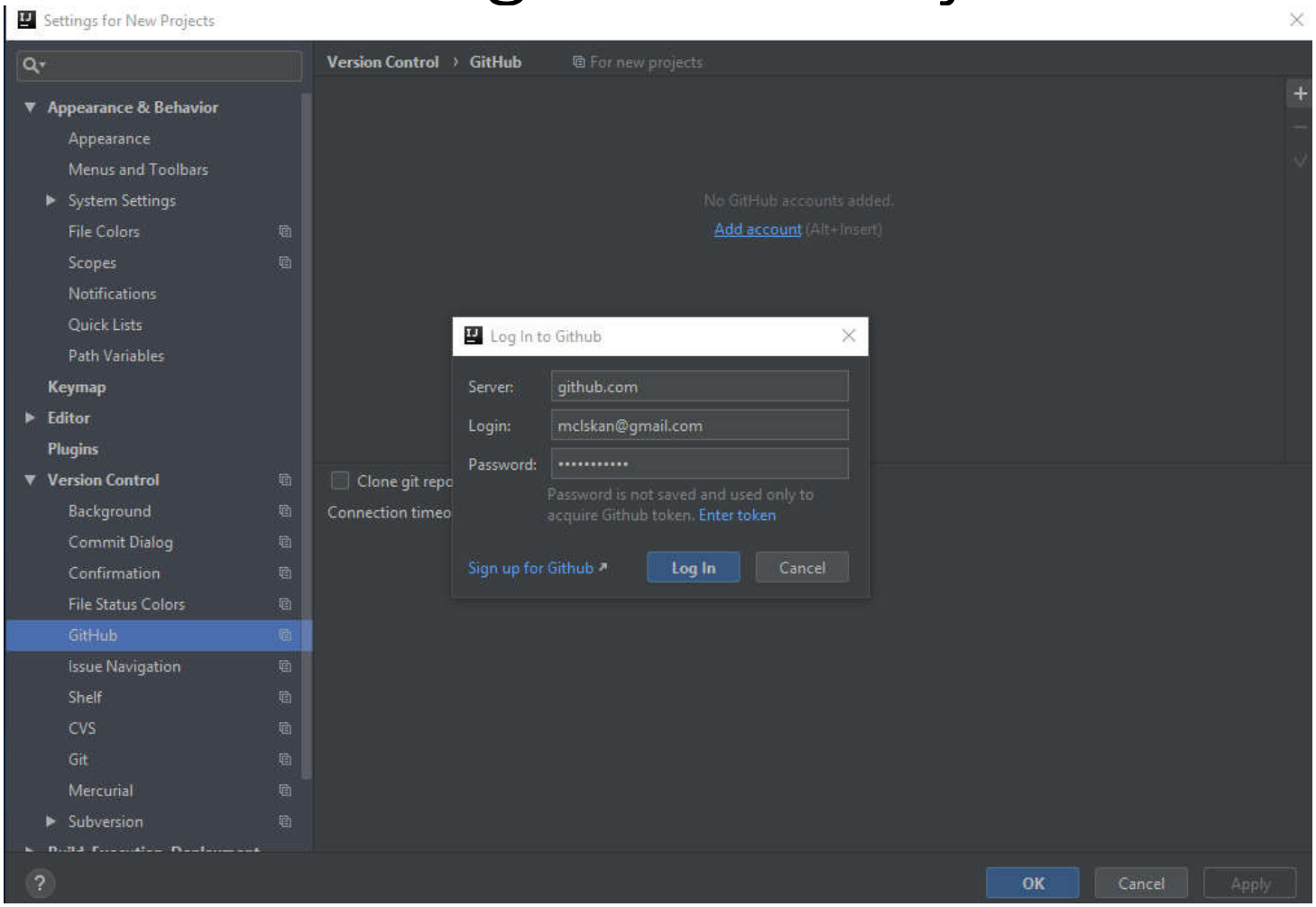

```
Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects
$ git clone https://github.com/workhard12345/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 64 (delta 6), reused 43 (delta 6), pack-reused 20
Unpacking objects: 100% (64/64), done.

Mothership@DESKTOP-T1QD1CO MINGW64 ~/IdeaProjects
```

- Cloning the project from the remote repository.
- **Warning:** Clone is not checkout .
- **Clone:** fetching repositories you do not have
- **Checkout:** switching between branches in a repository which you have.

Git – Working with Intellij



Git – Working with Intellij



Git – Working with Intellij

- **Task 1**
  - Create a new project
  - Create new repository at github
  - Push to github
  - Put in github a file from the github.com.
  - Pull it to your local repository

  - Delete that file and push it to github
  - Insert a class and commit.
  - Then revert this commit.

Git – Working with Intellij

- **Task 1**
  - Create a new project
  - Create new repository at github
  - Push to github
  - Put in github a file from the github.com.
  - Pull it to your local repository

  - Delete that file and push it to github
  - Insert a class and commit.
  - Then revert this commit.

Git – Working with Intellij

- **Task 2**
  - Clone the project from github
  - Add new text file
  - Push to github

- **Task 3**
  - Clone the project from github
  - Add new text file
  - Push to github

- **Task 4**
  - Clone the project from github
  - Create a branch
  - Change any file.
  - Push to branch.
  - Merge to master