



502045

Software Engineering

Chapter 09

Lesson 15: Version Control with Git

Hello, Git!

- Installation in Windows
 - Configuration
- Basic Commands
 - GitHub

- Distributed SCM (source code management) system
 - Look at the word Distributed
 - Everyone will have a complete copy of all *history* of the project in their local computers.
 - That is, everyone has a **repository** in their own computers.
 - Repository, or “Repo” is where Git stores all necessary information. It's usually a *folder* with name “.git”

What's
Git?

- Store a complete history of your project
- Whenever you want to record a **snapshot** of your project, simply “**commit**”
 - ***Snapshot*** is nothing but the current situation of all files & folders of your project 😊
- Easily “**merge**” codes with your team-mates.
- Secured *backup* of your project. Since Git is *distributed*, every team-mate has complete record of the project, and it’s unlikely that everyone’s computer will crash at the same time!

Why Git?

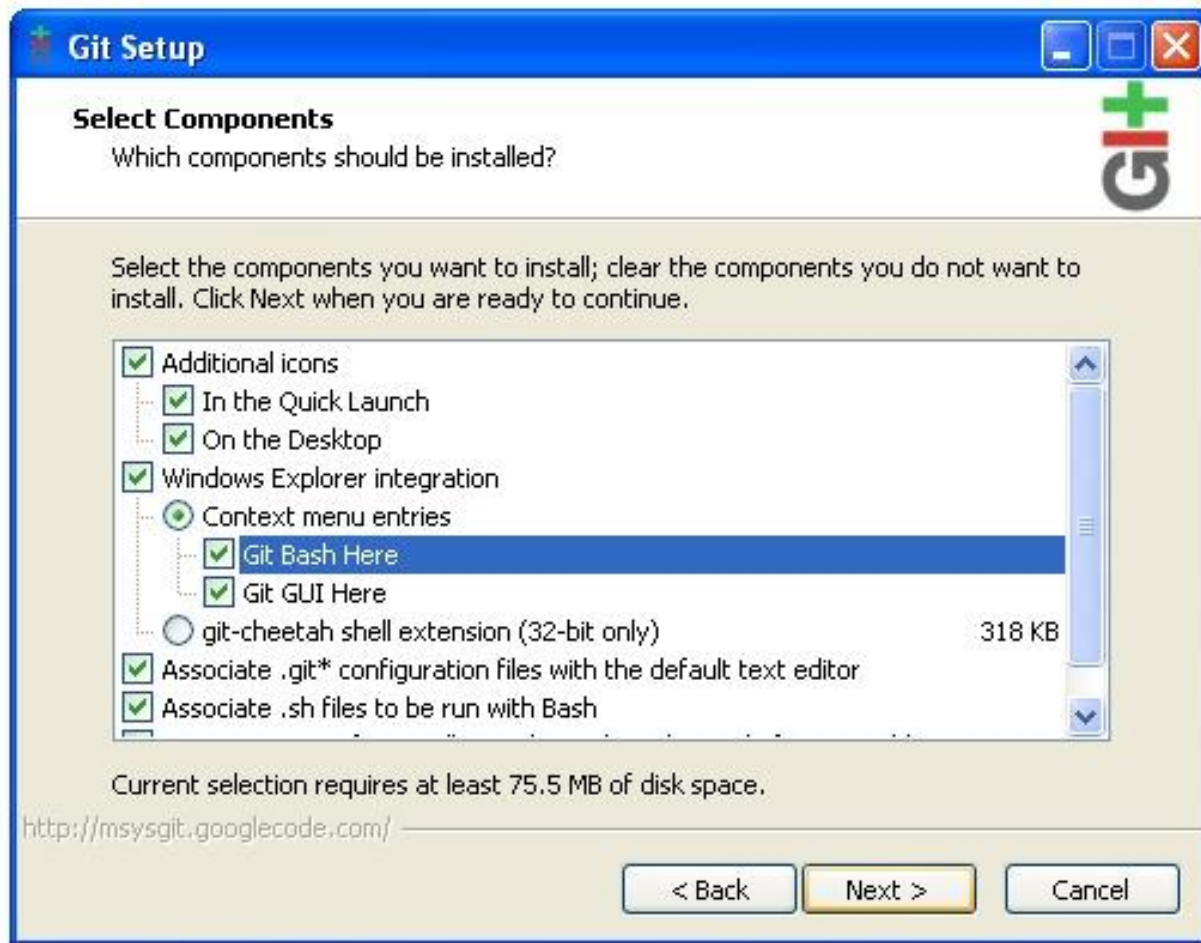
Windows setup

- Download ***msysgit*** from
 - <http://code.google.com/p/msysgit/downloads/list?q=full+installer+official+git>

Start Installation...



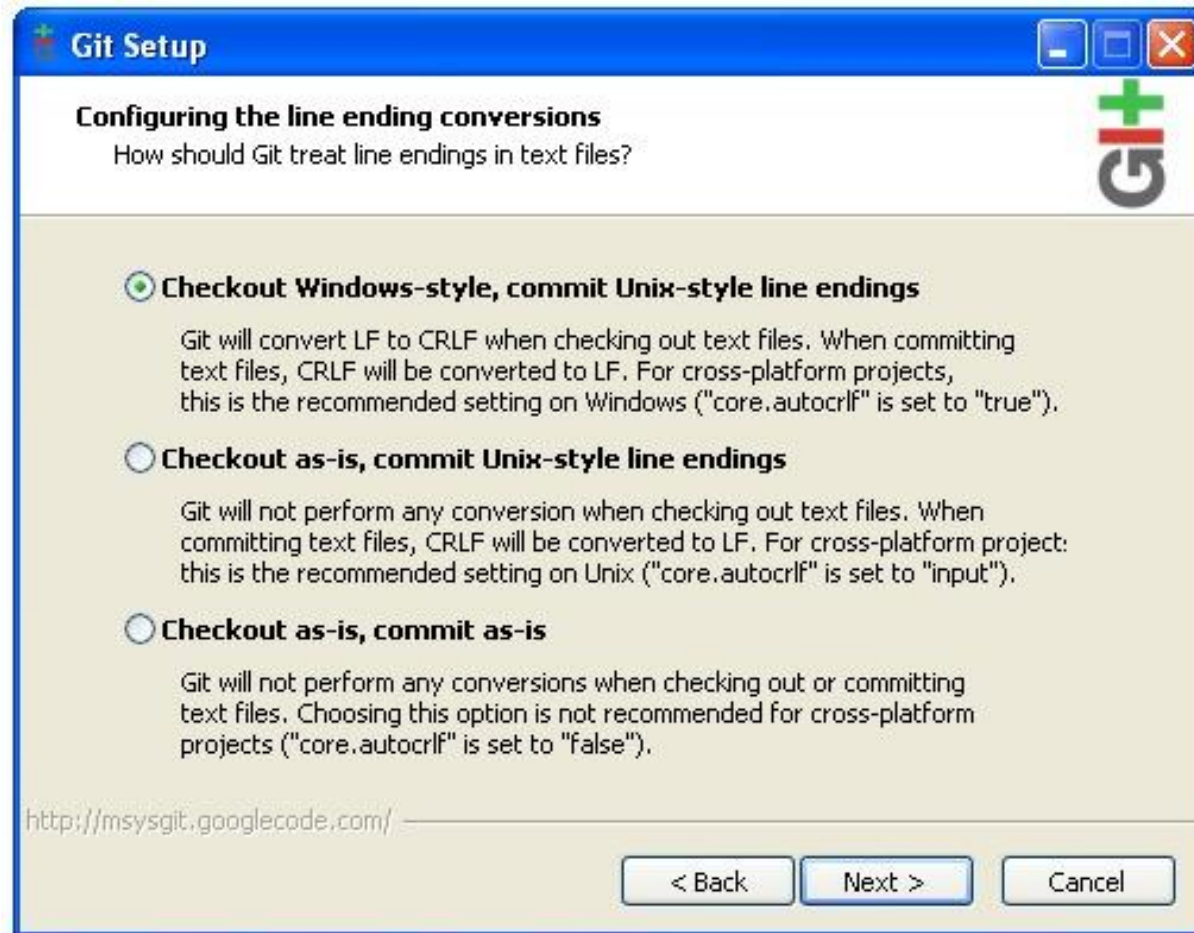
Make sure you tick these...



Go on...



Almost there...

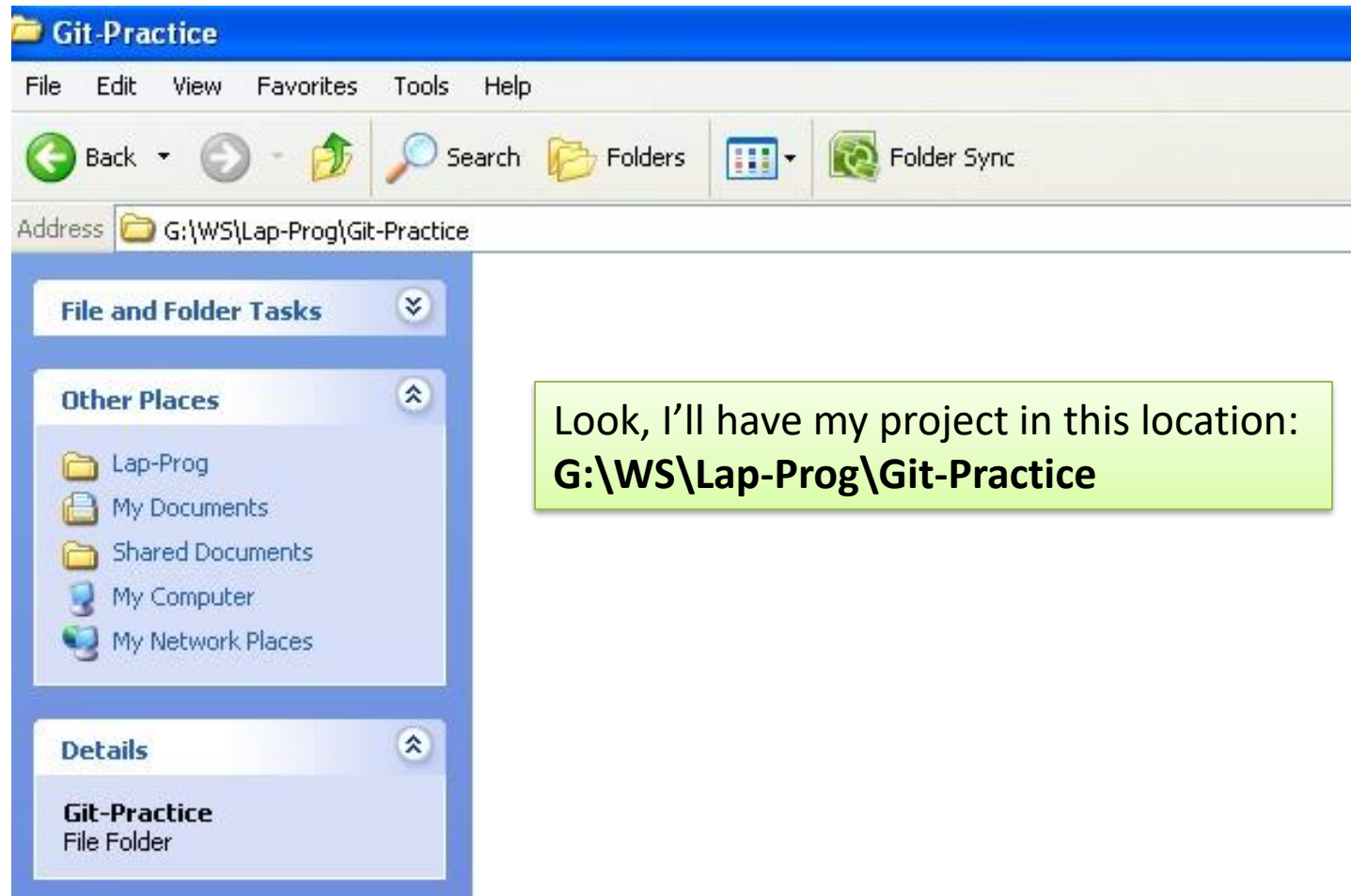


- You've successfully installed Git in your Windows.
- Now what?
- Create a new project?

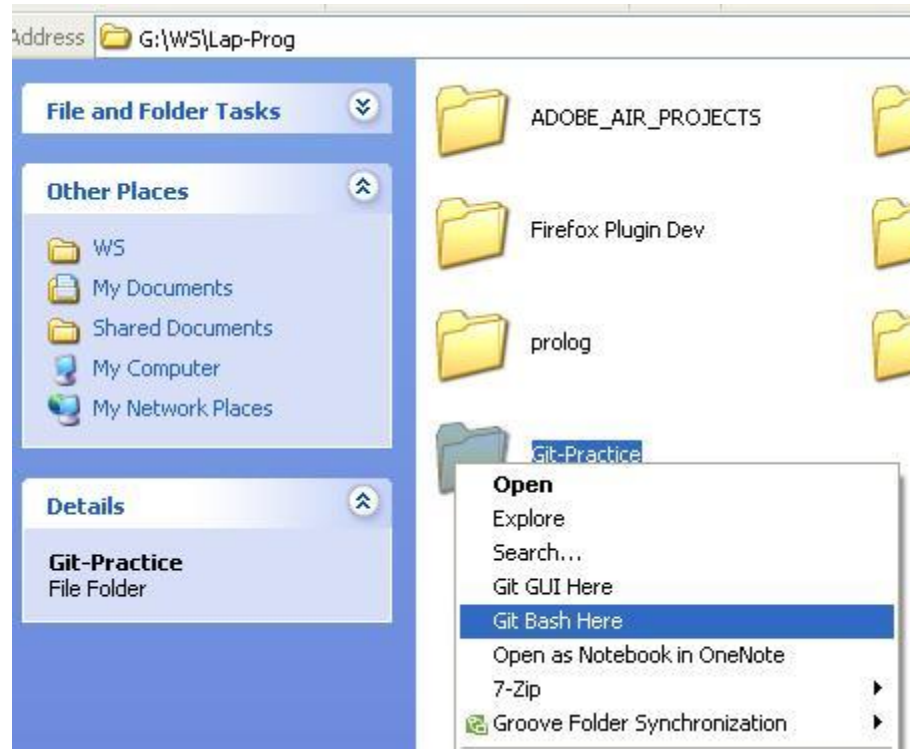


Done
Installing!

Go to the folder you want to start your project.

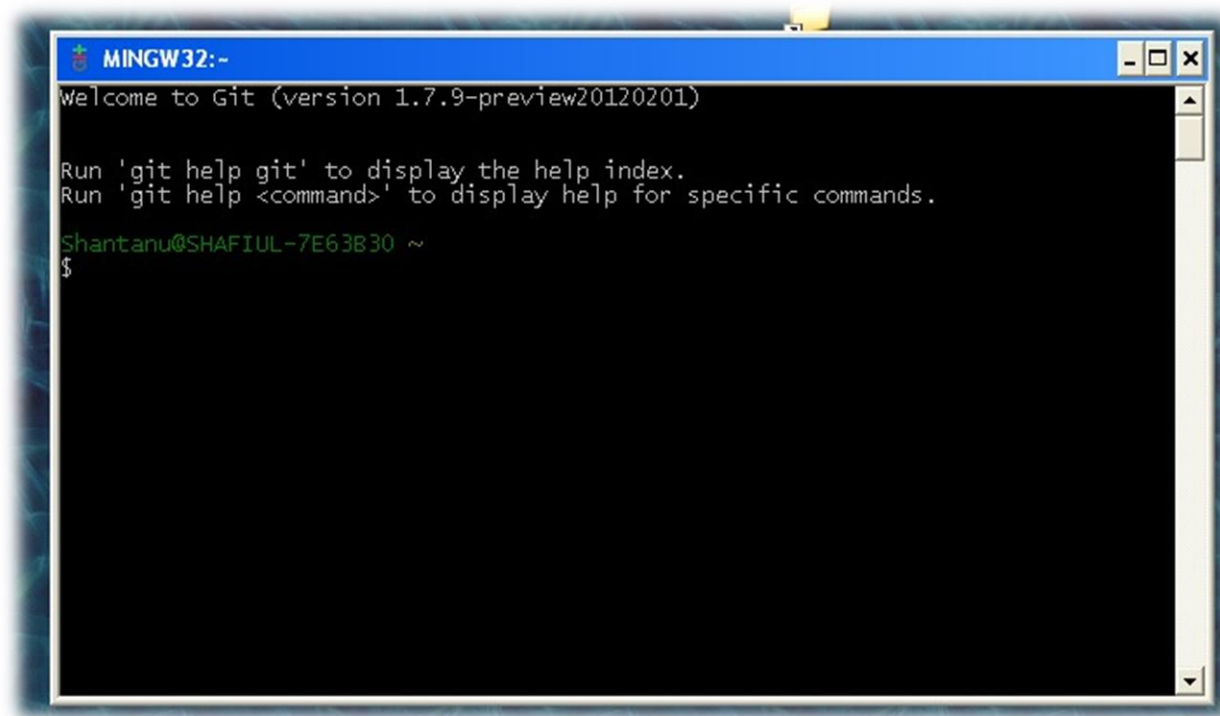


Right click on this folder and click “***Git Bash here***”



Every time you want to do something with Git, you will need to do this.

Say hello to the “Git Bash”

A screenshot of a Windows command prompt window titled "MINGW32:-". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). The terminal text is as follows:

```
Welcome to Git (version 1.7.9-preview20120201)

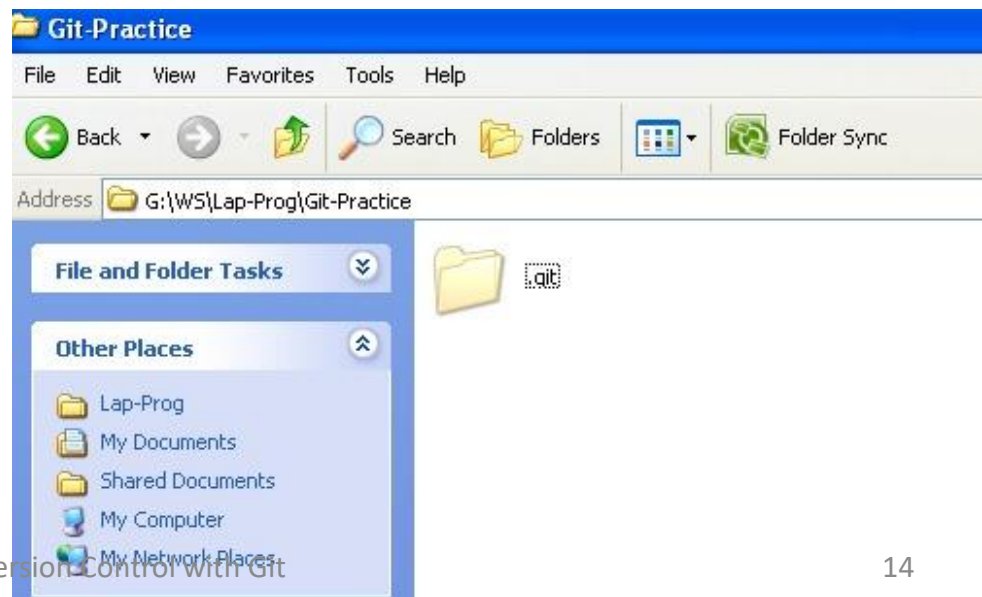
Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Shantanu@SHAFIUL-7E63B30 ~
$
```

Yes, it's a command-prompt interface. Don't get worried...

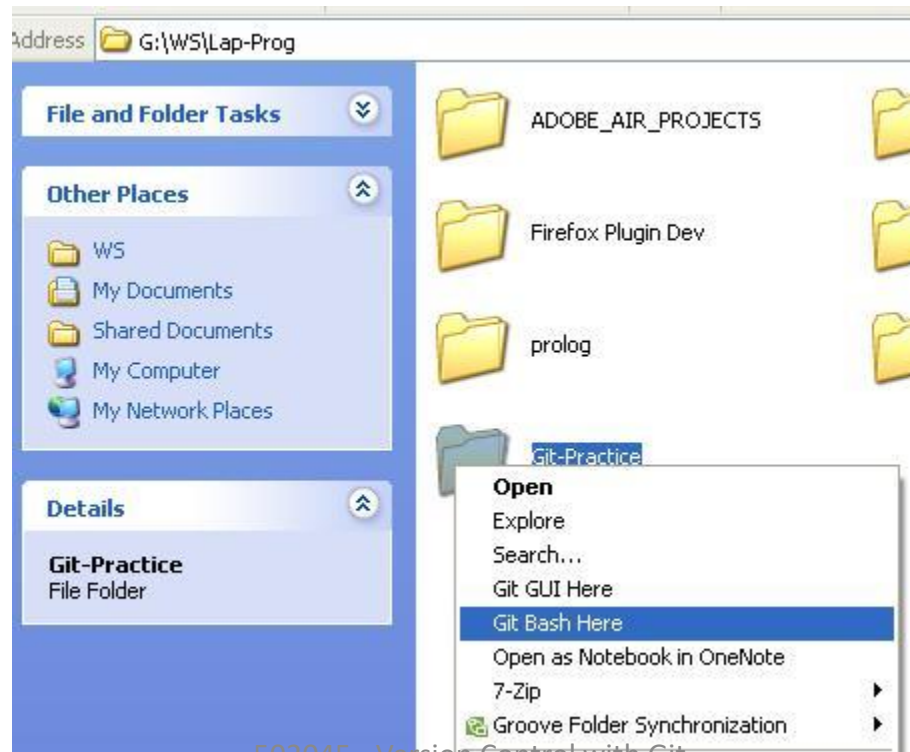
New Command: “*git init*”

- Type “git init” in ***Git Bash*** (the command-prompt window) to initiate your repository.
- You’ll see that a “.git” hidden folder is created. This folder is the Repository!!!



Congratulations!

- You've successfully created your Git Repo.
- Each time you want to use Git, you'll need to:



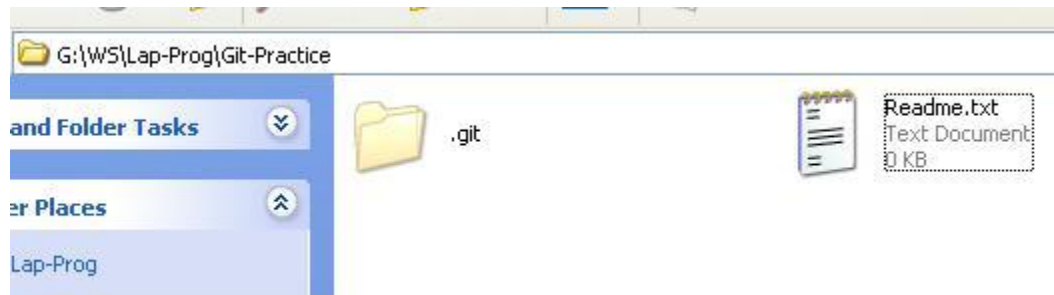
Tell Git who you are...

- Type in Git Bash following 2 commands:
 - `git config --global user.name "Your Name"`
 - `git config --global user.email your-email@yahoo.com`
- Congratulations, you've completed configuring your Git Bash.

Basic Commands

Create a New File in your project

- I've created a new *Readme.txt* file in my project's location...



New Command: “*git status*”

```
Shantanu@SHAFIUL-7E63B30 ~  
$ cd G:/WS/Lap-Prog/Git-Practice  
  
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)  
$ git status  
# On branch master  
#  
# Initial commit  
#  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
#       README.txt  
nothing added to commit but untracked files present (use "git add" to track)  
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)  
$
```

This command shows current status of your Repository, i.e. which files are Modified, which files are not yet tracked by Git etc...

It shows that, *Readme.txt* is an “**Untracked**” file. That is, Git does not know Anything about this file. You should type `git add -A` to start tracking All untracked files.

New Command: “*git commit*”

- Use `git add -A` to add start tracking all Untracked files.
- Then use `git commit -m “Message telling what you did”` command to actually *Commit* your changes.
 - Whenever you *commit*, you store a complete history/snapshot of your project. Later anytime, you may view the snapshots of your project wherever you made commit.

Add & Commit

- Whenever you wish to record snapshot of your project:
 - *git add -A* command in Git Bash will *add* all new & modified files for commit-list.
 - Then, *git commit -m “custom message”* command will actually do the commit.
 - Use a message telling what you’ve done, say, “removed database dependancy”

New Command: “*git log*”

- Displays a list of all commits you’ve made so far.
 - Tips: You want to go back to some previous commit? Hmm... you will need a list of all commits made in your project, and here *git log* command comes!
- Food for brain: how does Git uniquely identify each commit? It should assign each commit some number or Id...

Back to our example scenario

- After creating the Readme.txt file, we executed following commands in Git Bash, sequentially:
 - *git status*
 - *git add -A*
 - *git commit -m “My First Commit”*
 - *git log*
- Output window looks like the image of next slide...

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)

\$ git status

On branch master

#

Initial commit

#

Untracked files:

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

#

Readme.txt

nothing added to commit but untracked files present (use "git add" to track)

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)

\$ git add -A

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)

\$ git commit -m "My First Commit"

[master (root-commit) 7db40df] My First Commit

1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 Readme.txt

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)

\$ git log

commit 7db40dfe28a9c1fb829a628048dcfc9c80589eec

Author: Shafiul Azam <cppgcc@gmail.com>

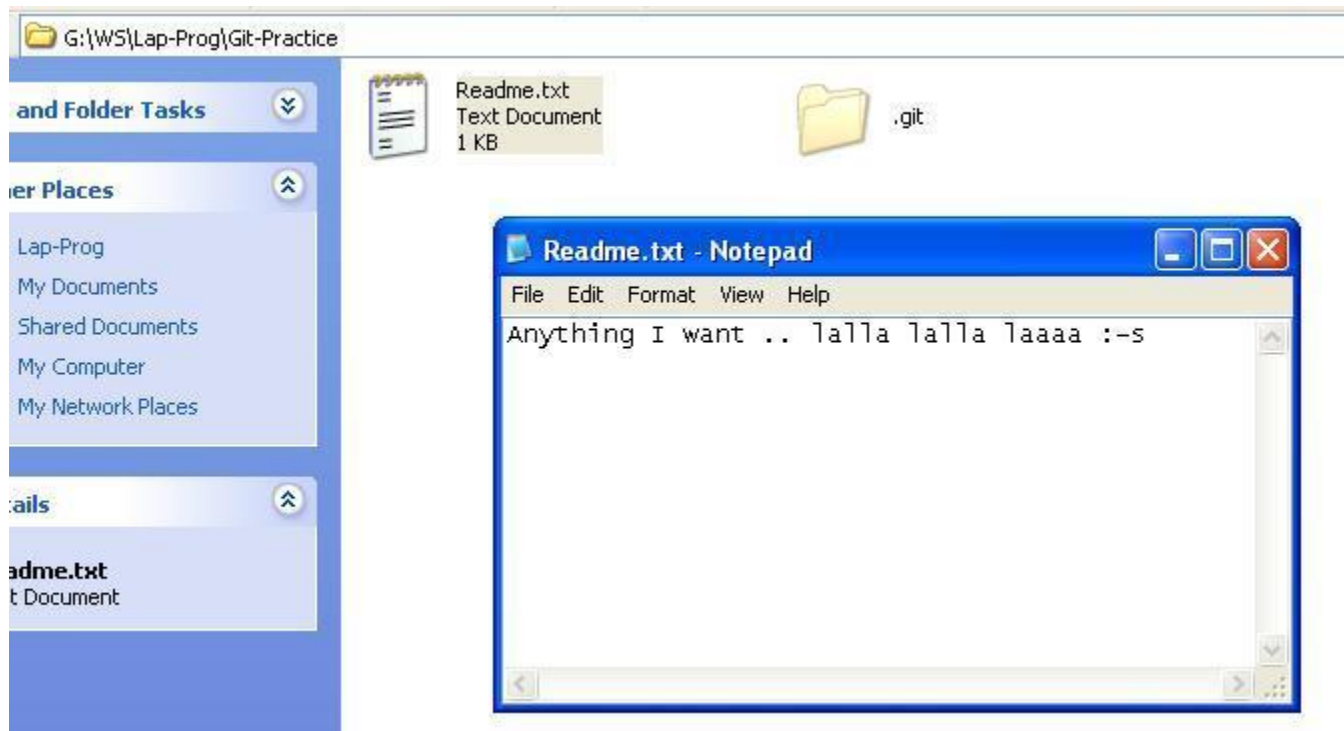
Date: Tue Jun 19 11:20:12 2012 -0700

My First Commit

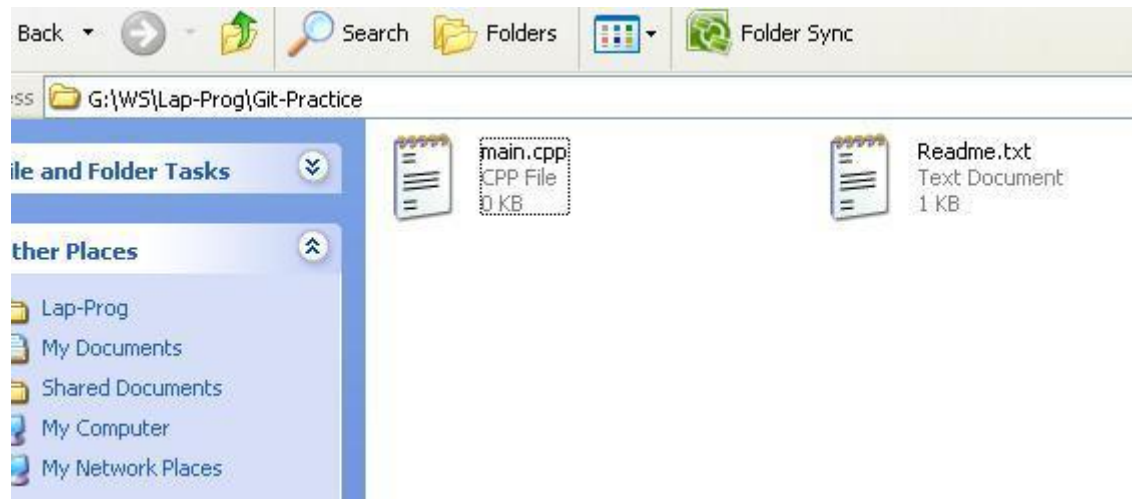
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)

\$

Now make some changes in Readme.txt



And create another file *main.cpp* or anything...



Now type *git status* in Git Bash

- Two types of status for files:
 - Changed (modified) file: *Readme.txt* (already tracked)
 - Totally Untracked file: *main.cpp* - because we've just created it and haven't told Git yet to start **tracking** it.

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (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:   Readme.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       main.cpp
no changes added to commit (use "git add" and/or "git commit -a")
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$
```

Add, Commit, and View log...

- Execute following commands in Git Bash, sequentially:
 - *git add -A*
 - *git commit -m “My second Commit”*
 - *git log*
- Output window is on next slide...

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ git add -A
```

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ git commit -m "My Second commit"
[master 7cb993a] My Second commit
 2 files changed, 2 insertions(+), 1 deletions(-)
 create mode 100644 main.cpp
```

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ git log
commit 7cb993a95d644a63c4a7be31b80ad6750b7c83c4
Author: Shafiul Azam <cppgcc@gmail.com>
Date:   Tue Jun 19 11:42:03 2012 -0700
```

My Second commit

```
commit 7db40dfe28a9c1fb829a628048dcfc9c80589eec
Author: Shafiul Azam <cppgcc@gmail.com>
Date:   Tue Jun 19 11:20:12 2012 -0700
```

My First Commit

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$
```

Commit IDs

- When we execute `git log` this time we see two commits!
- Each commit is identified by a string of 40 characters (say *“7db40dfe28a9c1fb829a628048dcfc9c80589eec” from our 1st commit example*)
 - The strings are underlined using red color in the image of the previous slide.
- We will use these strings to uniquely refer any particular commit of our project.

Teamwork

with
GitHub

Working in a team

- You and your team-mates.
- Everyone has their own *local* Git **repo** in their personal computers, and everyone is **committing** in their *local repos* (*Local repo is the Git repo in someone's own computer*).
- How will you get the **commits** made by your team-mates **merged** in your local Repo?
 - And vice versa, how will your team-mates get the *commits* made by you *merged* in their local repo?
- To get others' **commits merged** in your local repo, you **pull** from their repo.
 - And you **push** to their repo so that their local repo is *merged* with your *commits*! Making sense, right? 😊

You'll probably need a server...

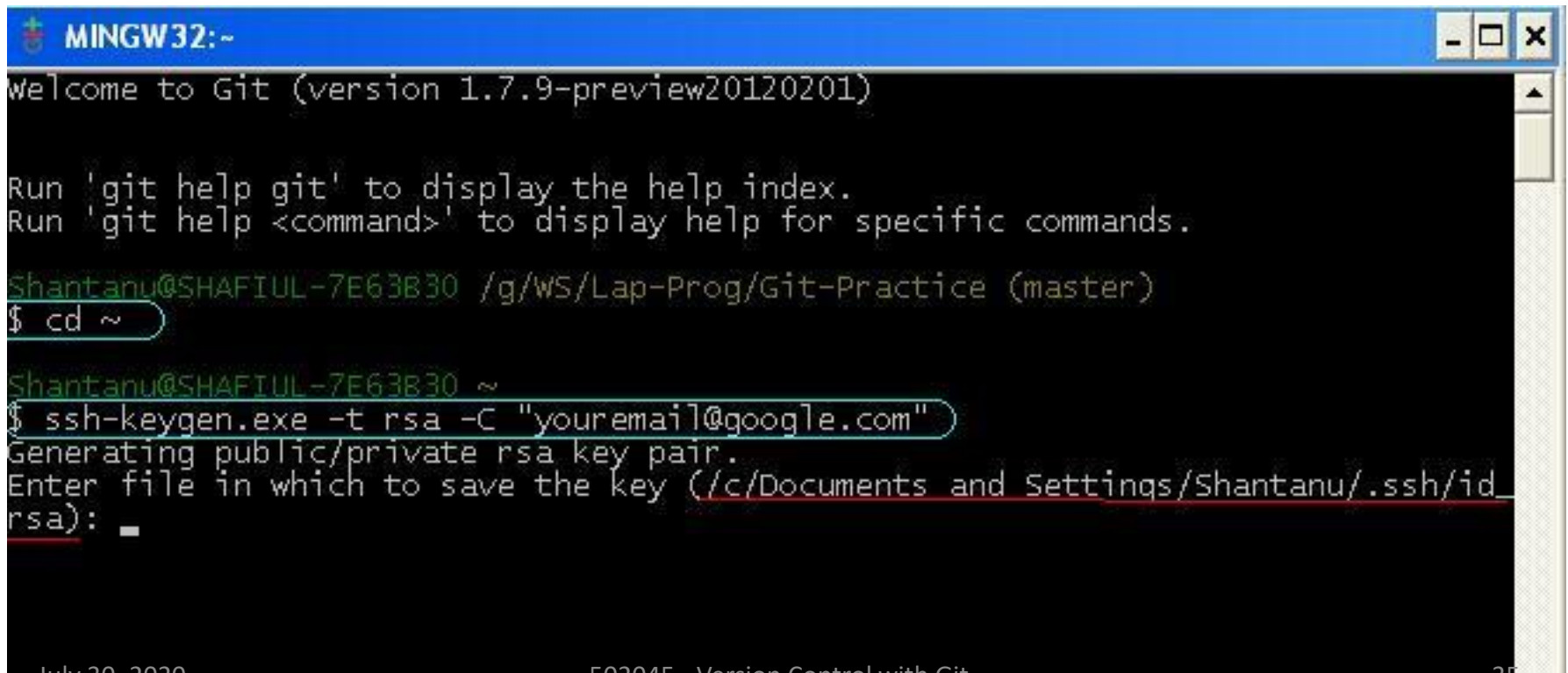
- So to get the *commits* made by others, you should **pull** from their local repos i.e. their computers.
- But wait... what if their computer has no Real IP? Then you can not send/receive network packets to/from it. Most of us don't have real IP.
- So we use a **server** which has a Real IP. Say, [GitHub.com](https://github.com) or [Assembla.com](https://assembla.com) – they provide us free Git hosting.
- They are nothing but Git repos like the repos in our local machine. But since they have real IP, we can directly push from & pull to it.
 - All team-mates including me **push** to the repo located in a server
 - All team-mates including me **pull** from the repo located in that server.
 - This repo is commonly addressed as “**origin**”

Another problem... how will the server ***authenticate*** the team-mates?

- Common approach: Username-Password system.
- But typically Git allows using ***Public & Private keys*** to authenticate someone.
 - Think of “public key” as your username. You can safely supply anyone your public key.
 - Think of “*private key*” as your password. You should NEVER provide anyone your private key!
 - Keys are nothing but some text/characters. LONG stream of characters!

Generate Public & Private keys

- Type following commands in Git Bash:
 - *cd ~*
 - *ssh-keygen.exe -t rsa -C "youremail@google.com"*



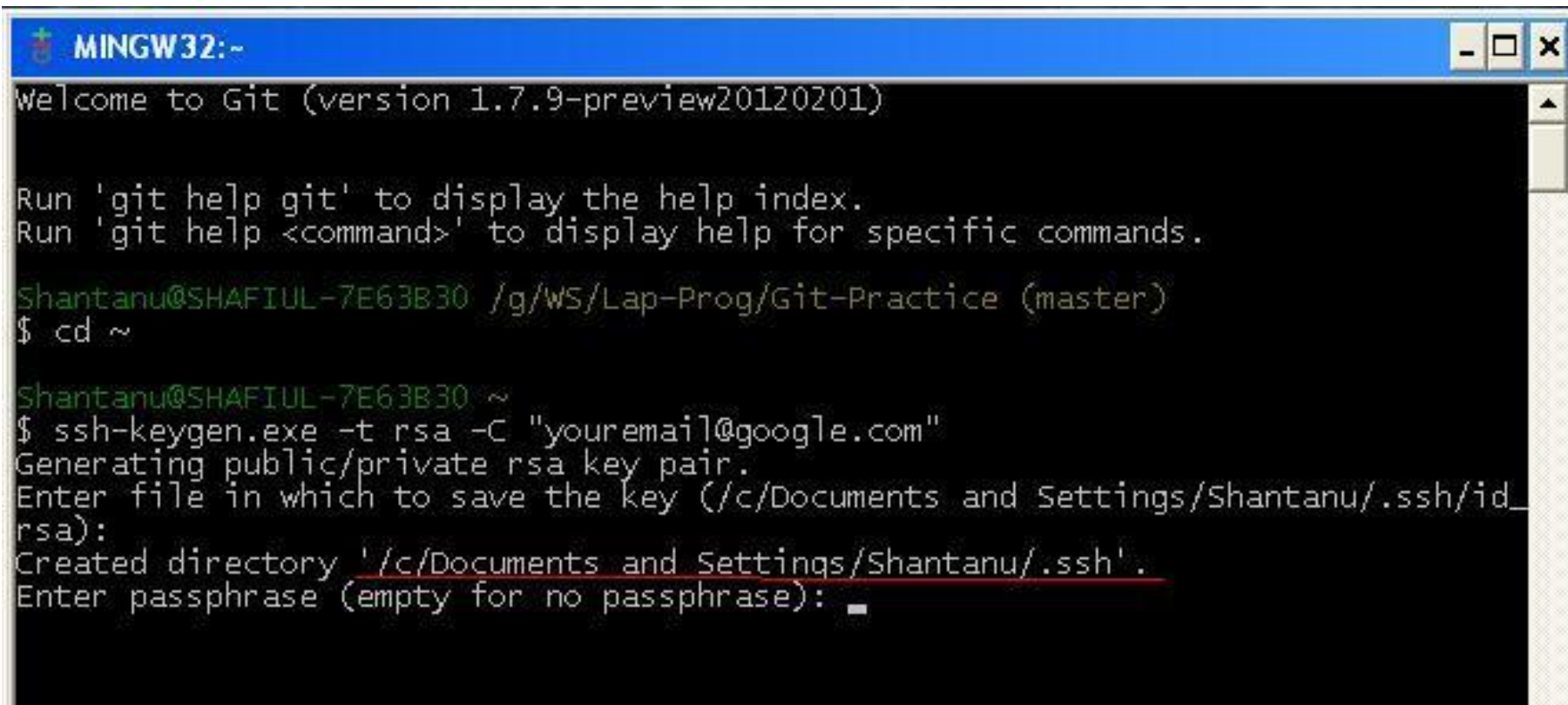
```
MINGW32:~
Welcome to Git (version 1.7.9-preview20120201)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ cd ~

Shantanu@SHAFIUL-7E63B30 ~
$ ssh-keygen.exe -t rsa -C "youremail@google.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Documents and Settings/Shantanu/.ssh/id
rsa): _
```

- Press **<enter>** in all prompts.
- Note the directory where the keys are created (**red colored underlined**)



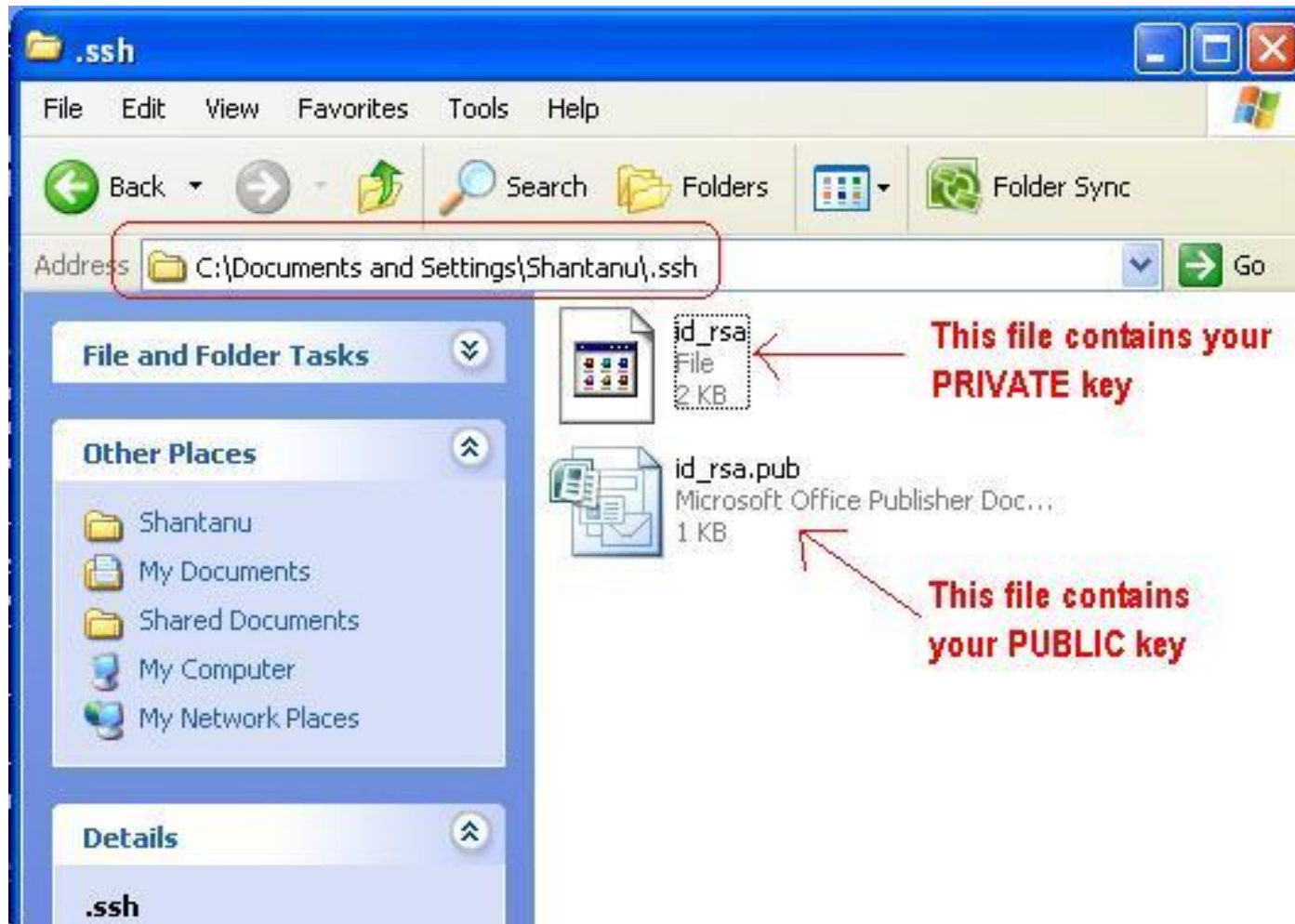
```
MINGW32:~
Welcome to Git (version 1.7.9-preview20120201)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ cd ~

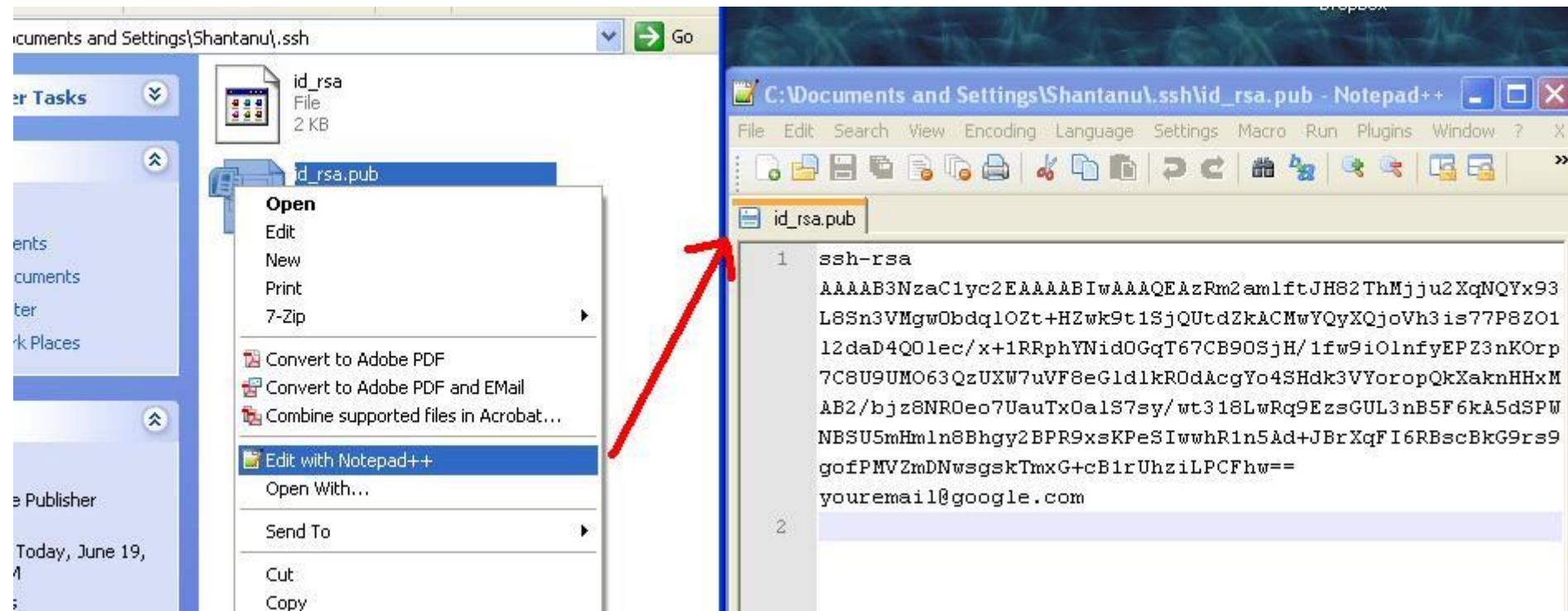
Shantanu@SHAFIUL-7E63B30 ~
$ ssh-keygen.exe -t rsa -C "youremail@google.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Documents and Settings/Shantanu/.ssh/id_
rsa):
Created directory '/c/Documents and Settings/Shantanu/.ssh/'.
Enter passphrase (empty for no passphrase): _
```

Open the directory (**red colored underlined** in previous slide's image)



You can open the files with Notepad++

- If you open them with Notepad++/Text Editor, you can find the keys.



Now you can open a free account at <http://github.com>

- After completing registration, add your PUBLIC key...

The screenshot shows the GitHub website interface. At the top, the browser address bar displays `https://github.com/settings/ssh`. A red arrow points to this address bar with the text "1. Go to this URL using your Browser". The GitHub logo and navigation links (Explore, Gist, Blog, Help) are visible. On the left sidebar, the user's profile "bsadd" is shown with a list of settings: Profile, Account Settings, Emails, Notification Center, Billing, Payment History, SSH Keys, Security History, and Applications. A red arrow points to the "SSH Keys" link with the text "2. Click this link". The main content area is titled "SSH Keys" and contains the text "There are no SSH keys with access to your account." A blue button labeled "Add SSH key" is in the top right corner of this section. A red arrow points to this button with the text "3. Click to add key". A help link is also visible: "Need help? Check out our guide to setting up Git and SSH keys or troubleshoot common SSH Problems".

Copy and Paste your PUBLIC key

Add an SSH Key

Title

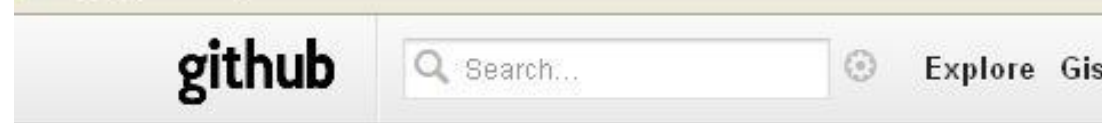
Key

Paste here your PUBLIC key (copy the text from the file with extension .pub)

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAAzRm2amltfJH82ThMjju2XqNQYx93L8Sn3VMgw0bdqIOZt+HZwk9t1SjQUtdZ
kACMwYQyXQjoVh3is77P8ZO1I2daD4Q0lec/x+1RRphYNid0GqT67CB90SjH
/1fw9iOlnfyEPZ3nKOrp7C8U9UMO63QzUXW7uVF8eGldIkR0dAcgYb4SHdk3VYoropQkXaknHHxMAB2
/bjz8NR0eo7UauTx0alS7sy
Aw1318LwRq9EzsGUL3nB5F6kA5dSPWNBSU5mHmln8Bhgy2BPR9xsKPeSlwwhR1n5Ad+JBrXqFI6RBscBkG9r
s9gofPMVZmDNwsgskTmxG+cB1rUhziLPCFhw== youremail@google.com
```

Add key **Then press this button to add your key**

- Cool! You've successfully provided your PUBLIC key at GitHub.com. Now you can *create your project*



1. Provide a Name

Owner: bsadd Repository name: My Cool Project

Great repository names are short and memorable.

Description (optional)

☒ Public Anyone can see this repository. You choose who can commit to it.
☐ Private You choose who can see and commit to this repository.

2. **Don't** check this!

☐ Initialize this repository with a README
This will allow you to git clone the repository immediately.
Add .gitignore: None

3. Click to *Create Repository*.

Create repository

You'll be given URL to your Repo.

b, Inc. (US) <https://github.com/bsadd/My-Cool-Project>

```
git add README
git commit -m 'first commit'
git remote add origin git@github.com:bsadd/My-Cool-Project.git
git push -u origin master
```



Existing Git Repo?

```
cd existing_git_repo
git remote add origin git@github.com:bsadd/My-Cool-Project.git
git push -u origin master
```

Remember this URL!

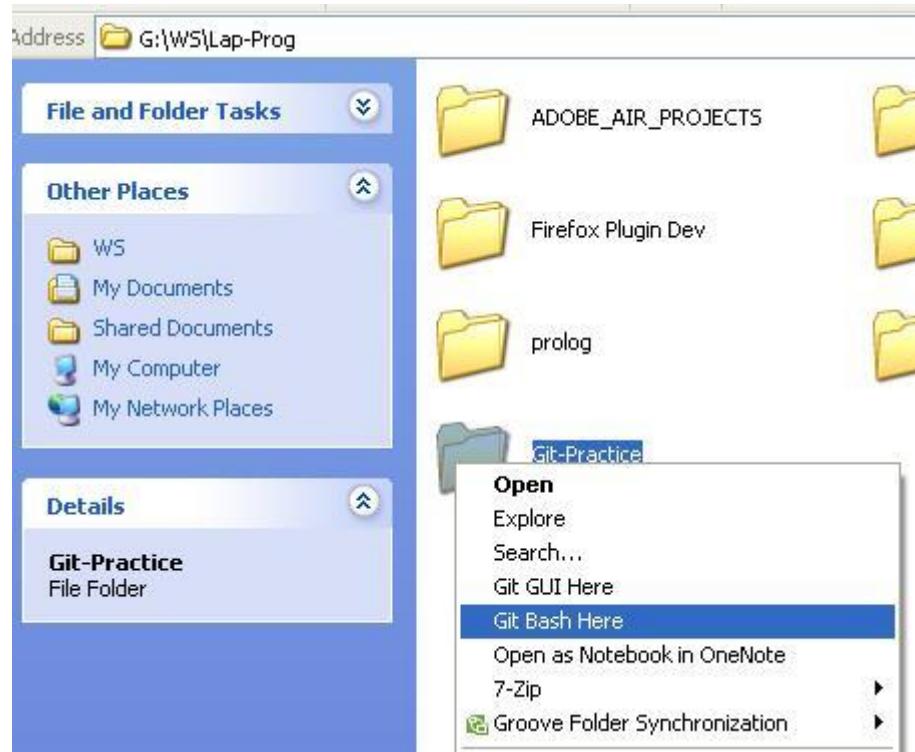
This is the URL to your git project at GitHub.com.

You can give this URL to your team-mates.

For our example, the URL is: [git@github.com:bsadd/My-Cool-Project.git](https://github.com/bsadd/My-Cool-Project)

Let's Push to GitHub's repo!

- Start Git Bash like usual...



Our First Push

- Type following command in Git Bash:
 - *git remote add origin git@github.com:bsadd/My-Cool-Project.git*
 - *git push origin master*

```
Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ git remote add origin git@github.com:bsadd/My-Cool-Project.git

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$ git push origin master
The authenticity of host 'github.com (207.97.227.239)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,207.97.227.239' (RSA) to the list of known hosts.
Counting objects: 7, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 537 bytes, done.
Total 7 (delta 0), reused 0 (delta 0)
To git@github.com:bsadd/My-Cool-Project.git
 * [new branch]      master -> master

Shantanu@SHAFIUL-7E63B30 /g/WS/Lap-Prog/Git-Practice (master)
$
```

July 30, 2020

Verify in GitHub's Website

<https://github.com/bsadd/My-Cool-Project>

Clone in windows ZIP HTTP SSH Git Read-Only [git@github.com:bsadd/My-Cool-Project.git](https://github.com/bsadd/My-Cool-Project.git)

branch: master Files Commits Branches 1 **URL to your Repo**

Latest commit to the master branch

My Second commit

Your Latest Commit Information

 **bsadd** authored 13 hours ago

My-Cool-Project /

name	age	message
 Readme.txt	13 hours ago	My Second commit [bsadd]
 main.cpp	13 hours ago	My Second commit [bsadd]

Files in your project

New Command: “*git clone*”

- Now you’ve pushed the initial repo to GitHub.
- Then, your team-mates should know about this repo and grab the code from it for the first time.
 - All of your team-mates should issue this command in their Git Bash:
 - *git clone git@github.com:bsadd/My-Cool-Project.git*
- Remember: they should not create Git repo in their computer. Only one person in the team should create the repo using *git init* command
 - Then he should push it to a server using *git push* command
 - And all other team mates should Clone the repo to their local computers using *git clone* command.
- Note: You should add your team-members in the Project Admin area. Otherwise they will not have access to push in the GitHub repo (see image in next slide)

Adding team members

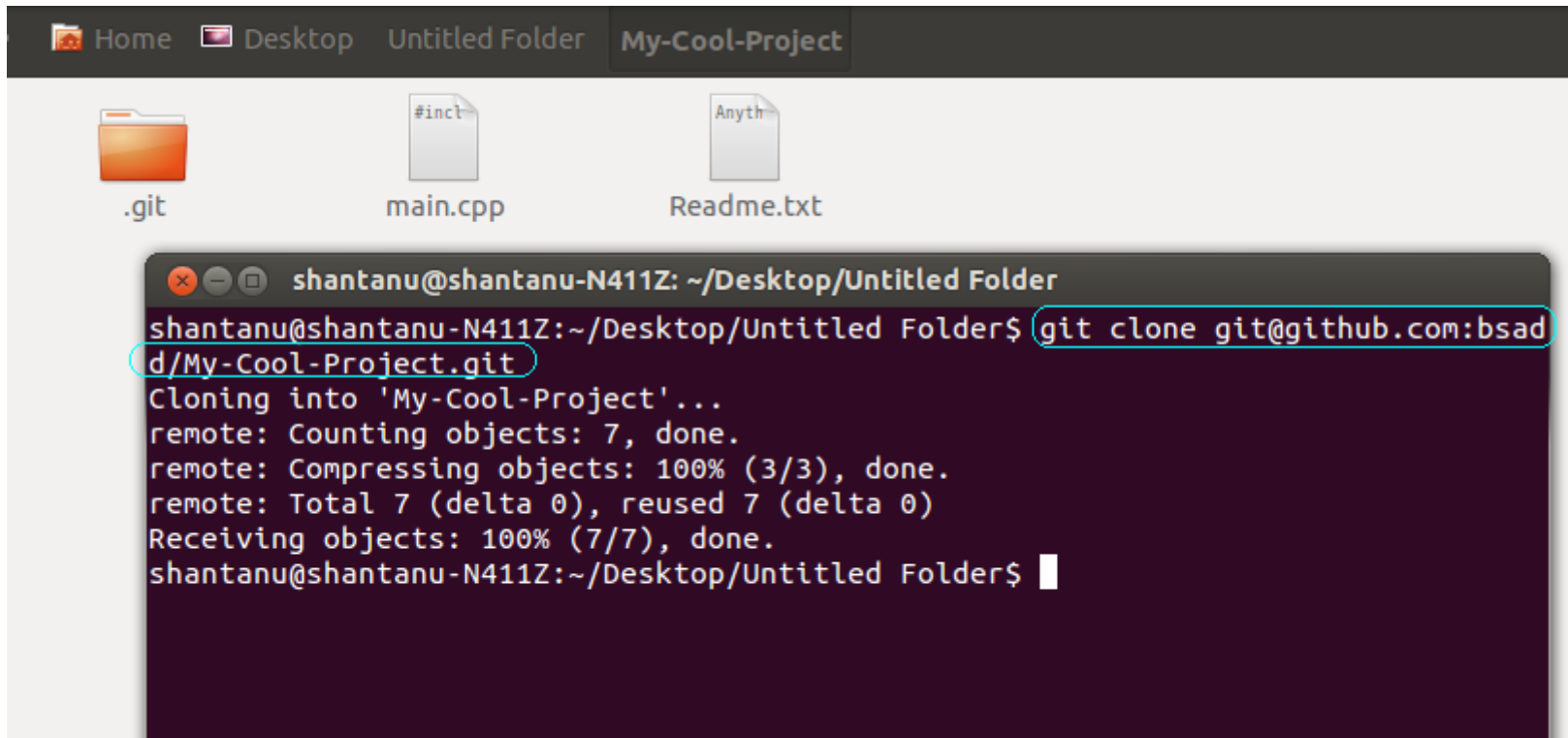


I've added "shafiul" in my project. This is the username of Shafiul Azam (my team-mate) in GitHub.com

Cloning

- When Shafiul will clone using `git clone` command, a new project will be created for him.
- Then he will be able to `push/pull/add/commit` whatever...

Shafiul, who is using Linux (another kind of operating system like Windows), cloned my Repo.



The screenshot shows a Linux desktop environment. At the top, a dark grey bar contains navigation buttons: 'Home', 'Desktop', 'Untitled Folder', and 'My-Cool-Project'. Below this, a file manager window displays three items: a folder icon labeled '.git', a file icon labeled 'main.cpp', and a file icon labeled 'Readme.txt'. In the foreground, a terminal window is open with the title 'shantanu@shantanu-N411Z: ~/Desktop/Untitled Folder'. The terminal shows the command `git clone git@github.com:bsad` being executed, followed by the output: `Cloning into 'My-Cool-Project'...`, `remote: Counting objects: 7, done.`, `remote: Compressing objects: 100% (3/3), done.`, `remote: Total 7 (delta 0), reused 7 (delta 0)`, `Receiving objects: 100% (7/7), done.`, and the prompt `shantanu@shantanu-N411Z:~/Desktop/Untitled Folder$`.

Wow! Git has created the repo for him with my codes “Readme.txt” & “main.cpp”
Note that this is his own local repo in his computer. He can do whatever he wants..

Shafiul will ***Push...***

- When Shafiul will finish editing, he will eventually type following commands in his Git Bash:
 - *git add -A*
 - *git commit -m “some message”*
 - *git push origin master*
- When he pushes, GitHub’s repo will be updated with his changes.

And I will ***Pull!***

- To get the changes made by shafiul automatically merged in my local computer, I will need to use command ***git pull***
 - *git pull origin master*
- Now my local repo will be updated with Shafiul's changes.
- Similarly I can push and Shafiul will need to pull...