

GIT WORKSHOP

GIT WORKSHOP



Manuela Salvucci

manuelasalvucci@rcsi.ie

2019-11-06



OUTLINE

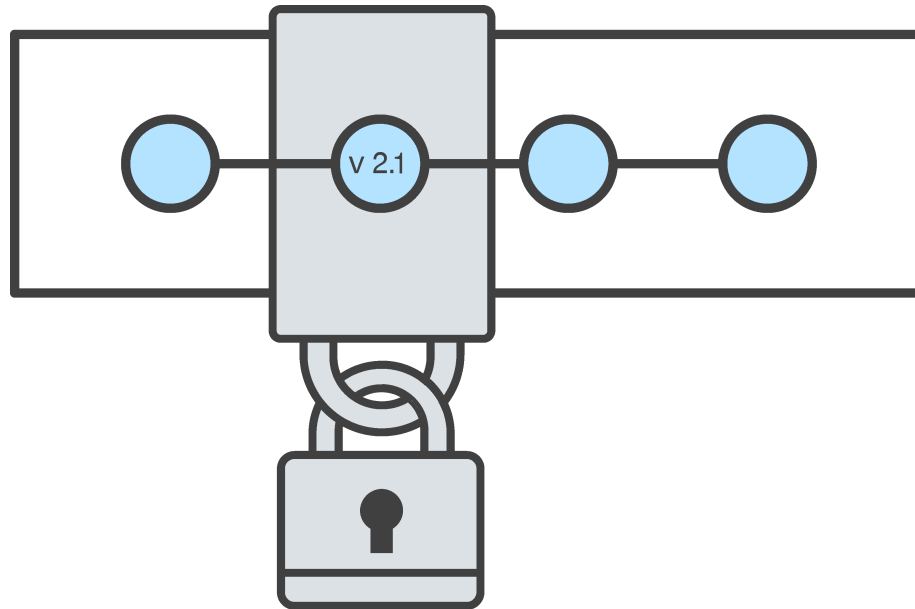
- What is version control?
- Why bother with “formal” version control?
- How to install and get started with GIT
- Use GIT core features
- Review files history, revert/amend changes
- Collaborate online with others with BitBucket, GitHub or GitLab
- Hands-on examples

VERSION CONTROL

WHAT IS VERSION CONTROL?

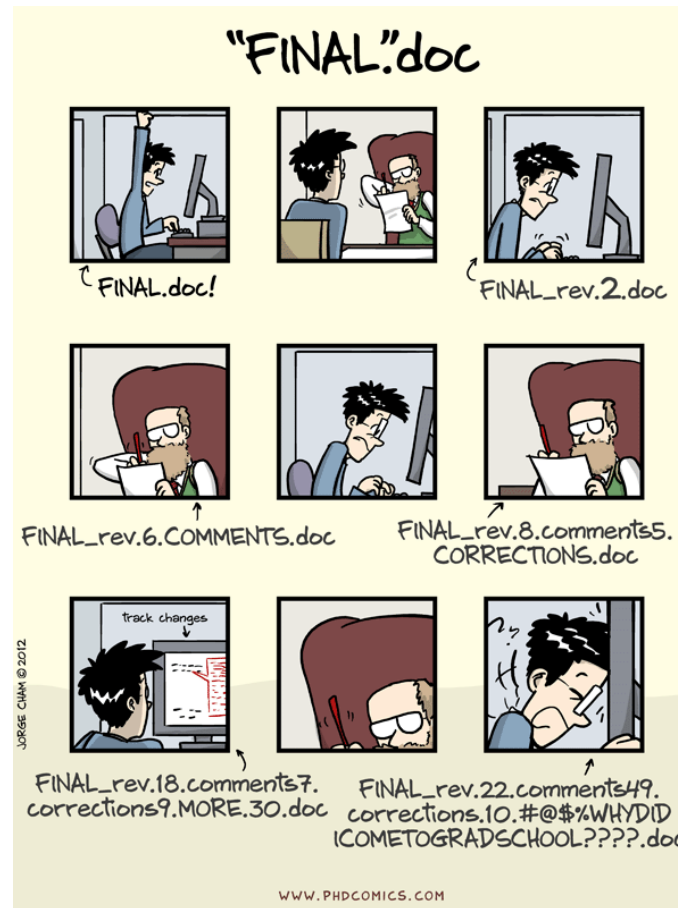
“Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.”

(Pro Git, Scott Chacon and Ben Straub, 2014)



From <https://wac-cdn.atlassian.com/dam/jcr:34e935dd-3108-40ef-bb3d-9ed01d977d6d/hero.svg?cdnVersion=659>

WITHOUT VERSION CONTROL... A WAY TOO FAMILIAR PICTURE



From <http://phdcomics.com/comics/archive.php?comid=1531>

WITHOUT VERSION CONTROL... “INFORMAL” VERSIONING

- None
- Named files:
 - OK:
 - manuscript_my_draft.docx
 - manuscript_my_draft_with_coauthor_comments.docx
 - ...
 - Better:
 - manuscript_draft_v01.docx
 - manuscript_draft_v02.docx
 - ...
- Named zip-files:
 - manuscript_drafts.zip
 - manuscript_cell_submission.zip
 - manuscript_pnas_submission.zip
 - manuscript_pnas_revisions.zip
 - manuscript_pnas_proofs.zip
- Sync online services (Microsoft/Dropbox/Google/Overleaf/Sharelatex)

WITHOUT VERSION CONTROL... CHALLENGES

- Time consuming
- Error prone
- Requires self-discipline (save everything, good file names, sticking to a routine, ...)
- Relationship between changes in multiple files is lost
- Information about what, when and why something changed is lost?
 - How would you go about finding out when the p-value for Figure 2.A got set to the (wrong) value?
- Non-linear history (parallel versions)
- Disk space



From <https://dynamicbusiness.com.au/wp-content/uploads/2012/09/>

WHY BOTHER WITH “FORMAL” VERSION CONTROL?

- We are too busy to use inefficient, manual, error-prone versioning
- Research is increasingly collaborative:
 - we need a better way to document the rationale behind data cleaning, analysis steps, generation of figures, write-ups...
 - we need a better way to “merge” inputs and feedback to the project from collaborators
 - often your future self is the collaborator (and you don’t reply to emails...)
- We do research anywhere:
 - on our workstation at work
 - on the laptop at home/bus/conference
 - on a dedicated facility workstation
 - ...
- Projects are always evolving and never “really” finished

WHAT CAN VERSION CONTROL SYSTEMS DO FOR MY RESEARCH?

- Version Control Systems are software that keep track of your files and their full history
- Project files and “history” in the form of “snapshots”/“checkpoints” are organized in a folder
- Explicitly indicate what file(s) and what change(s) to store with a named snapshot (include why the changes were made)
- Can “go back in time” and see/use files how they look at a specific snapshot
- Can see what changed between snapshots, and in what snapshot content was first introduced
- Can “experiment” by having “organized parallel versions” of files
- Synchronize different copies of the project between different computers/collaborators

VERSION CONTROL SYSTEMS - VOCABULARY

- Version Control Systems are software that keep track of your files and their full history
- Project files and “history” in the form of “snapshots”/“checkpoints” are organized in a folder -> **repository**
- Explicitly indicate what file(s) and what change(s) to store with a named snapshot (include why the changes were made) -> **commit** or **revision**
- Can “go back in time” or “jump forward” and see/use files how they look at a specific snapshot -> **checkout** or **revert**
- Can see what changed between snapshots, and in what snapshot content was first introduced -> **diff**, **annotate** and **blame**
- Can “experiment” by having “organized parallel versions” of files -> **branch**
- Synchronize different copies of the project between different computers/collaborators -> **push** & **pull**

WHAT TYPE OF FILES CAN I TRACK WITH VERSION CONTROL?

- All types of files can be tracked with version control (but big files may require special care)
- **Version control is most useful for plain “text”-files** (txt, md, tex, csv, .py, .R, .m, html,) where differences between versions can be “easily” visualized and multiple changes can be merged/combined automatically
- Version control works also for binary files (docx, xlsx, etc.), but it would only tell us if there is a change, but not visualize the change and the version control system will not be able to merge changes automatically

WHAT VERSION CONTROL SYSTEMS ARE AVAILABLE?

- GIT, PerForce, Mercurial, Subversion (SVN), Bazaar, Concurrent Versions System (CVS), Monotone,
- We will focus on GIT in this workshop



RhodeCode
@rhodecode

Follow

What is your [#versioncontrol](#) of choice in 2016?

Vote and share, then check the Insights:
bit.ly/vcs-popularity...

87% #Git

6% #SVN #Subversion

5% #Mercurial #Hg

2% #Perforce

881 votes • Final results

4:50 AM - 18 Jul 2016

21 Retweets 12 Likes



2

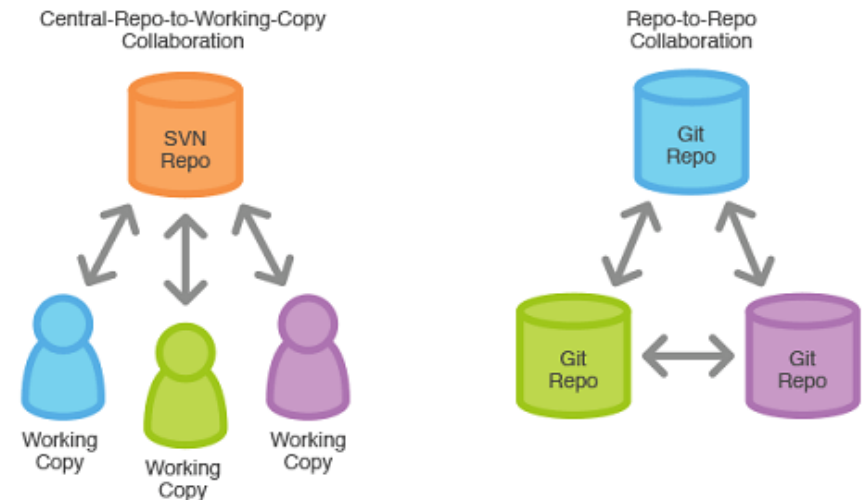
21

12

From <https://twitter.com/rhodecode>

CENTRALISED VS. DISTRIBUTED VERSION CONTROL SYSTEM

- In the centralized setup, there is a single (central) copy of the project and each user will apply changes to the central copy
- In the distributed setup, each user has their own (full) copy of the project (a clone)
- SVN, PerForce, CVS are examples of centralised version control system
- **GIT** and Mercurial are examples of **distributed version control system**



From <https://github.com/AnnieCannons/ac-terminal-and-git/blob/gh-pages/images/versioncontrol>

GIT

GIT



From <https://git-scm.com/>

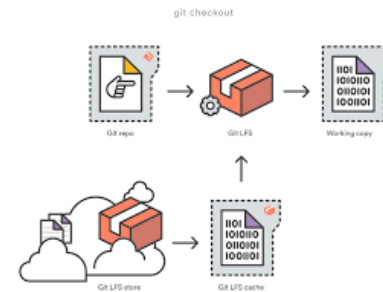
- Popular version control software:
 - Distributed system
 - Free and Open Source
 - Available for Windows, Linux and Mac
 - A lot of support, infrastructure and tools available to interface with GIT:
 - graphical user interfaces (GUIs)
 - seamless integration with Integrated Development Environments (IDEs) for R, MATLAB, Python, ...
 - cloud services (BitBucket, GitHub, GitLab)
- Developed by the Linus Torvalds in 2005 to manage the development of Linux and maintained by Junio Hamano

TRACKING LARGE FILES WITH GIT

“Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise”

<https://git-lfs.github.com/>

- Drop-in replacement for “normal” GIT -> **git lfs add** vs. **git add**
- Files are stored “externally”, so that GIT operations can run seamlessly and fast
- Good solution for “large” files (100 MB - 2 GB)



From <https://git-lfs.github.com/>

- Alternatives:
 - [git-annex](#)
 - Do not version control large file
 - set permission to Read Only
 - version control metadata instead
 - ...

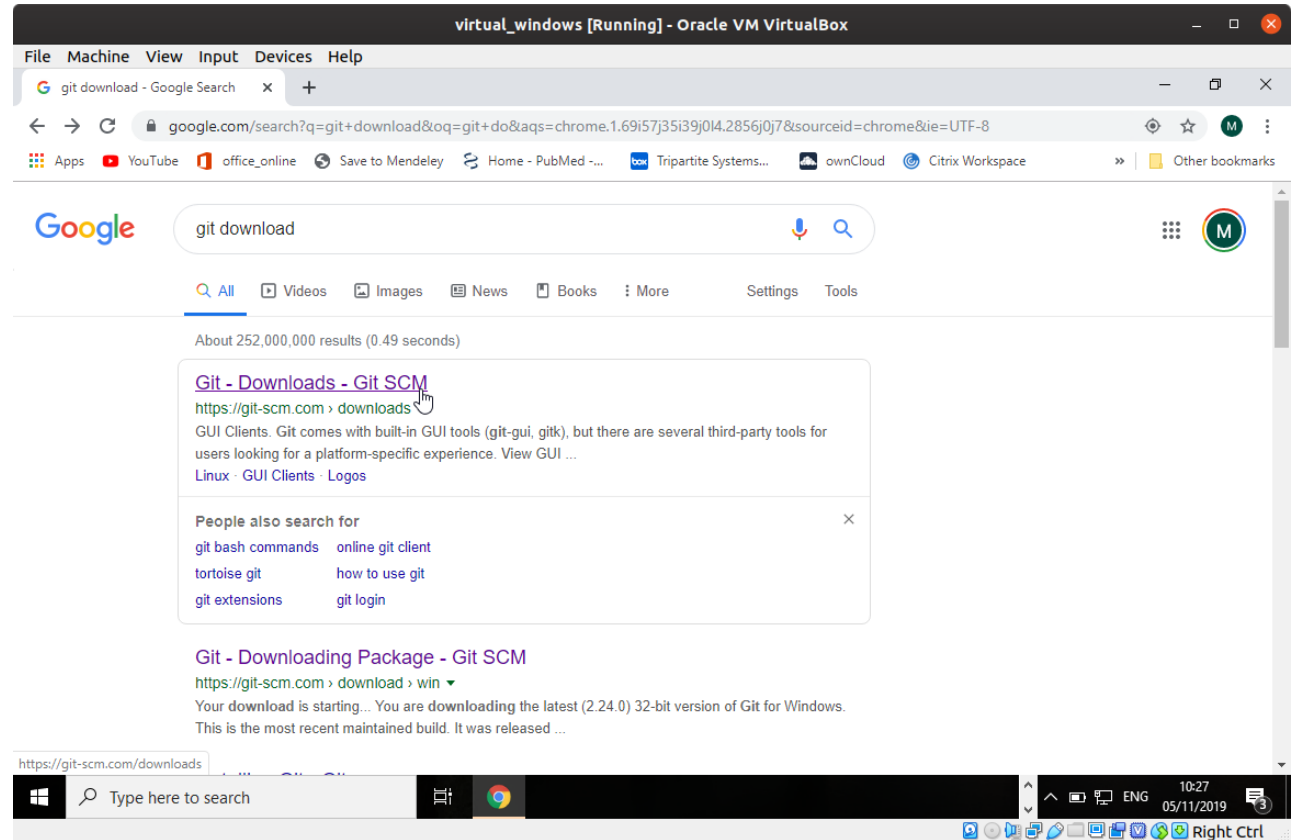
EXCLUDE FILES FROM TRACKING

- **.gitignore:** “special” file to list files and folders to intentionally not track
- Prevents files/folders from showing when running *git status* -> less clutter
- Can also be added by running *git add -f* (short for force)
- **Rationale:** not all files need to be version controlled
 - figures, tables, manuscript pdf generated by running code -> version control the raw data and the code to generate outputs instead
 - temporary files, compiled outputs, ...
- Checkout <https://www.gitignore.io/> to help identify files to ignore

INSTALLATION

GIT INSTALLATION

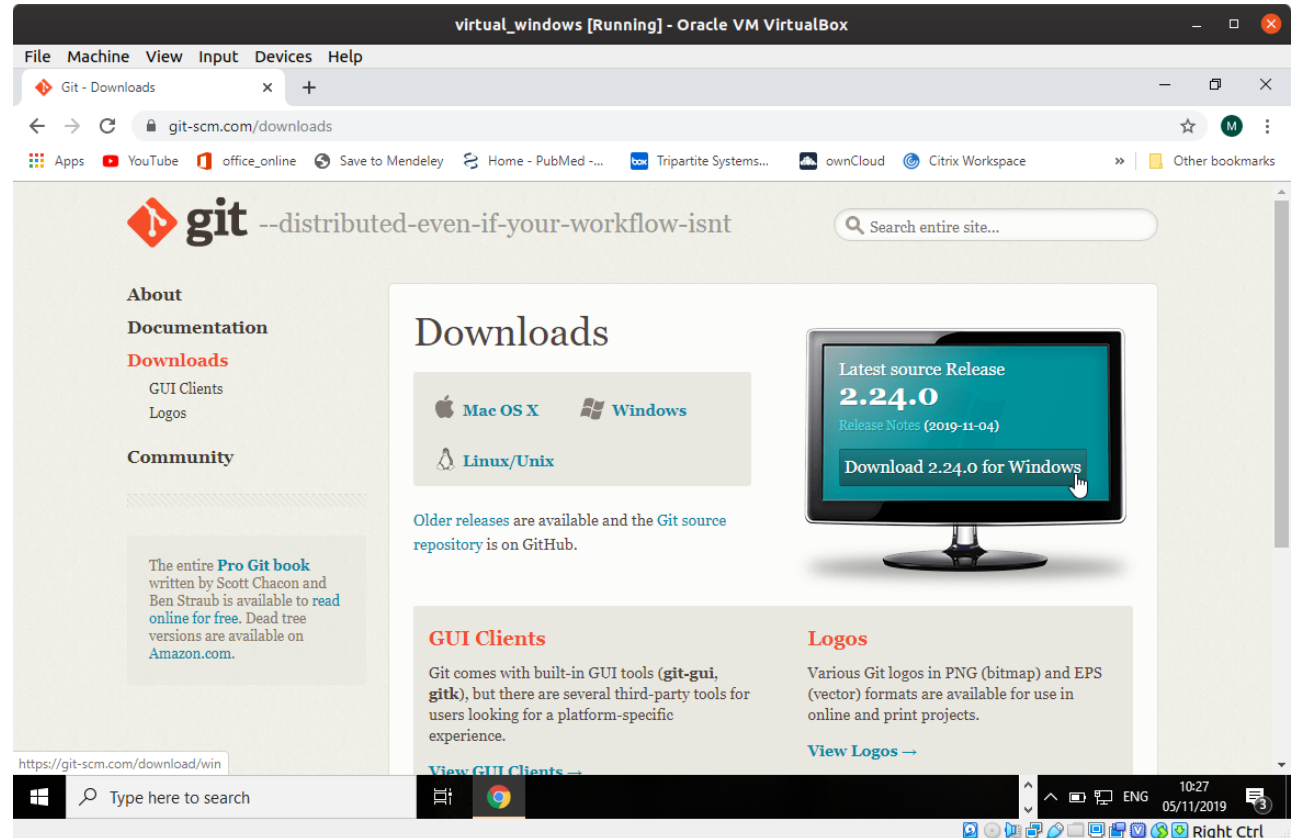
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Go to <https://git-scm.com/>

GIT INSTALLATION

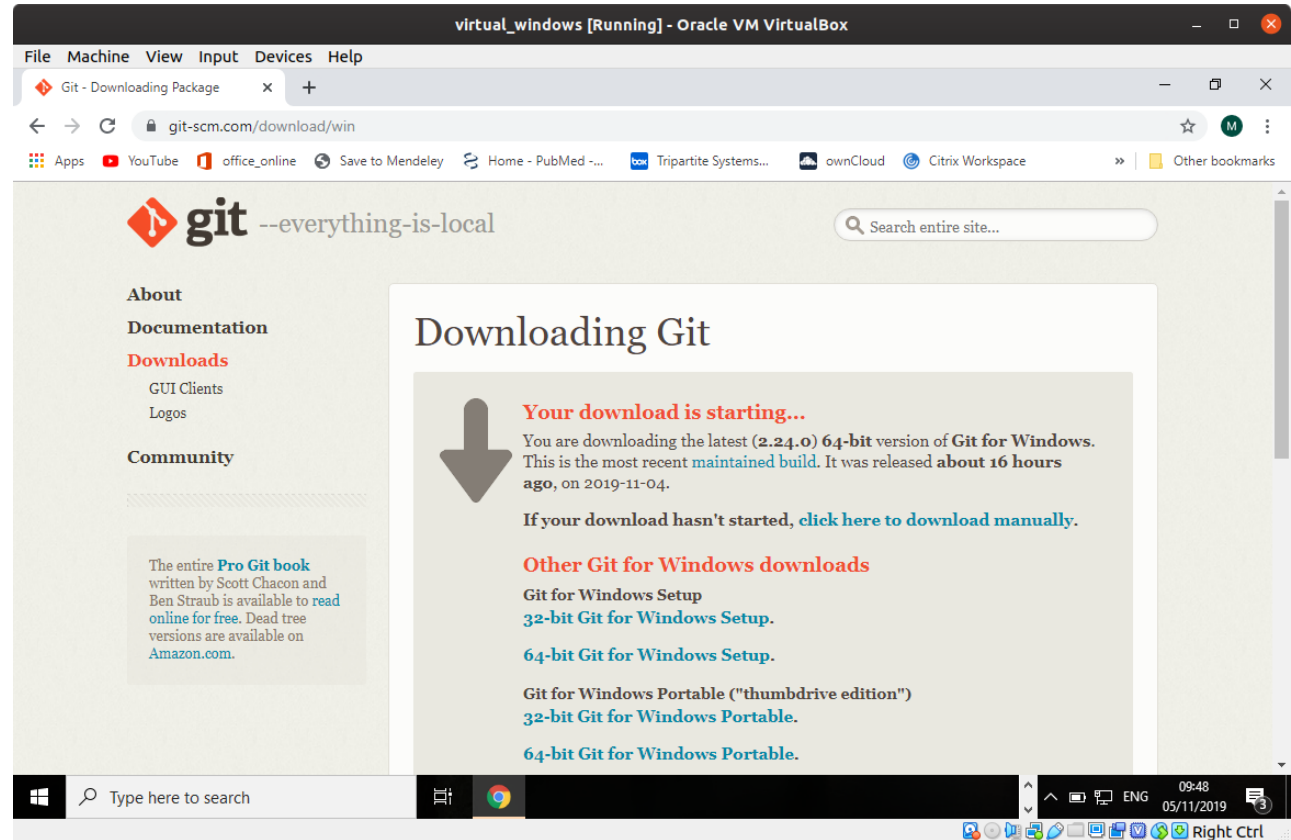
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Select to download latest stable GIT release for Windows

GIT INSTALLATION

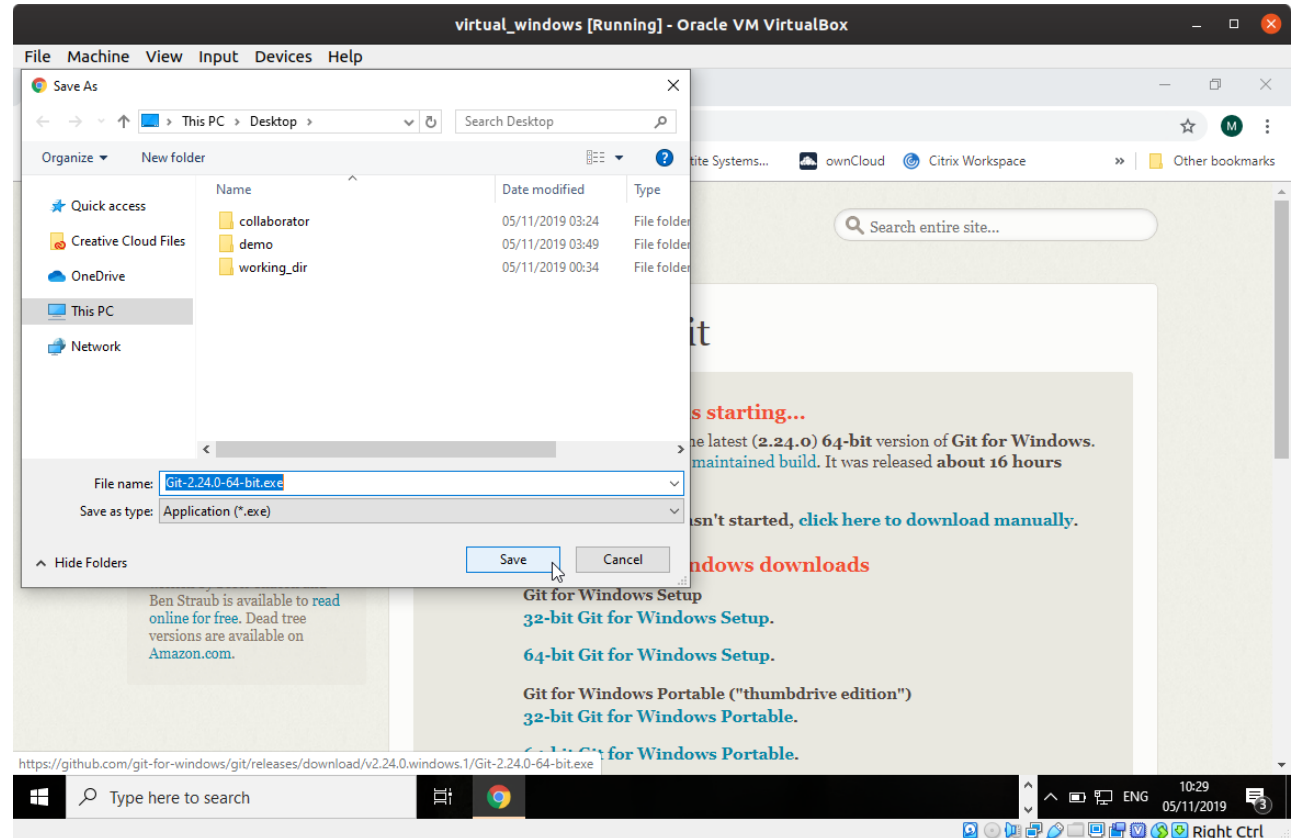
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Wait for executable to download

GIT INSTALLATION

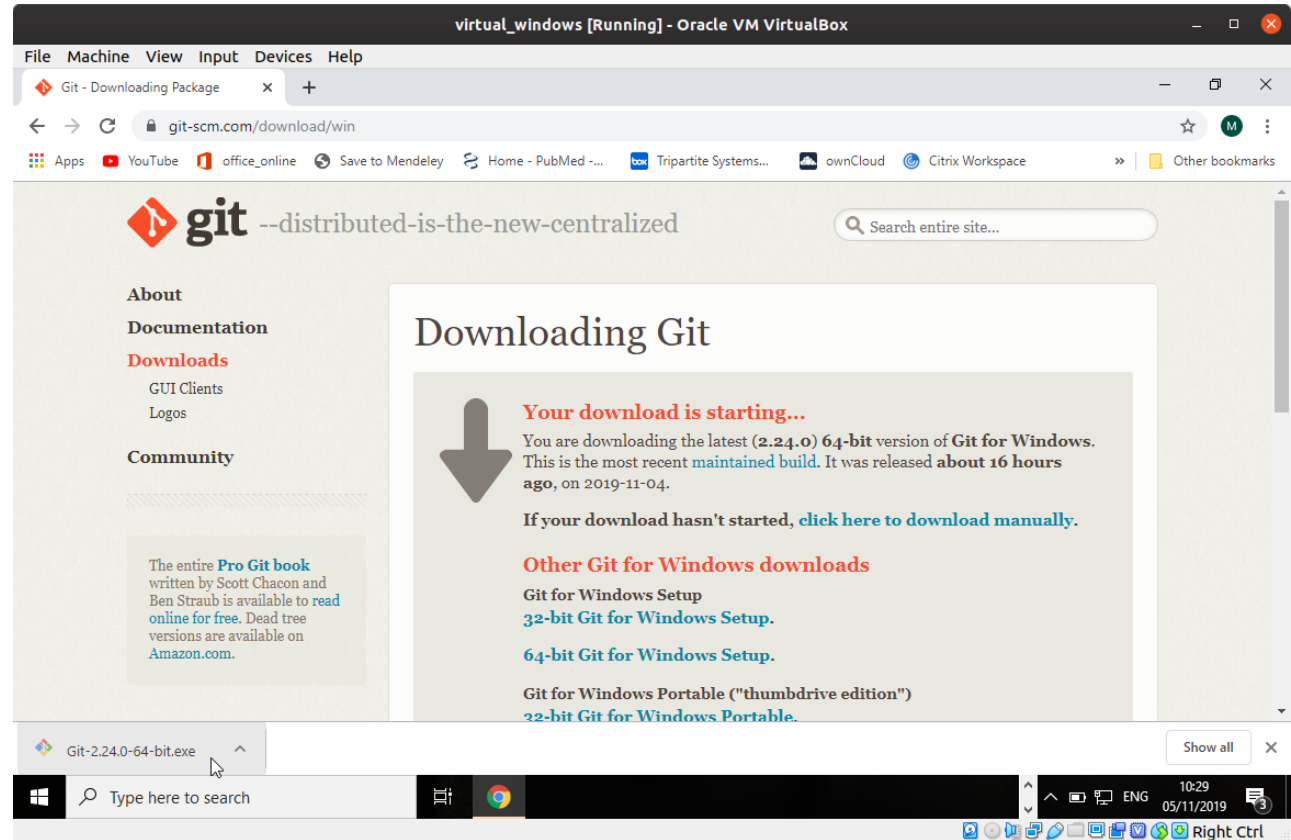
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Save executable in suggested folder

GIT INSTALLATION

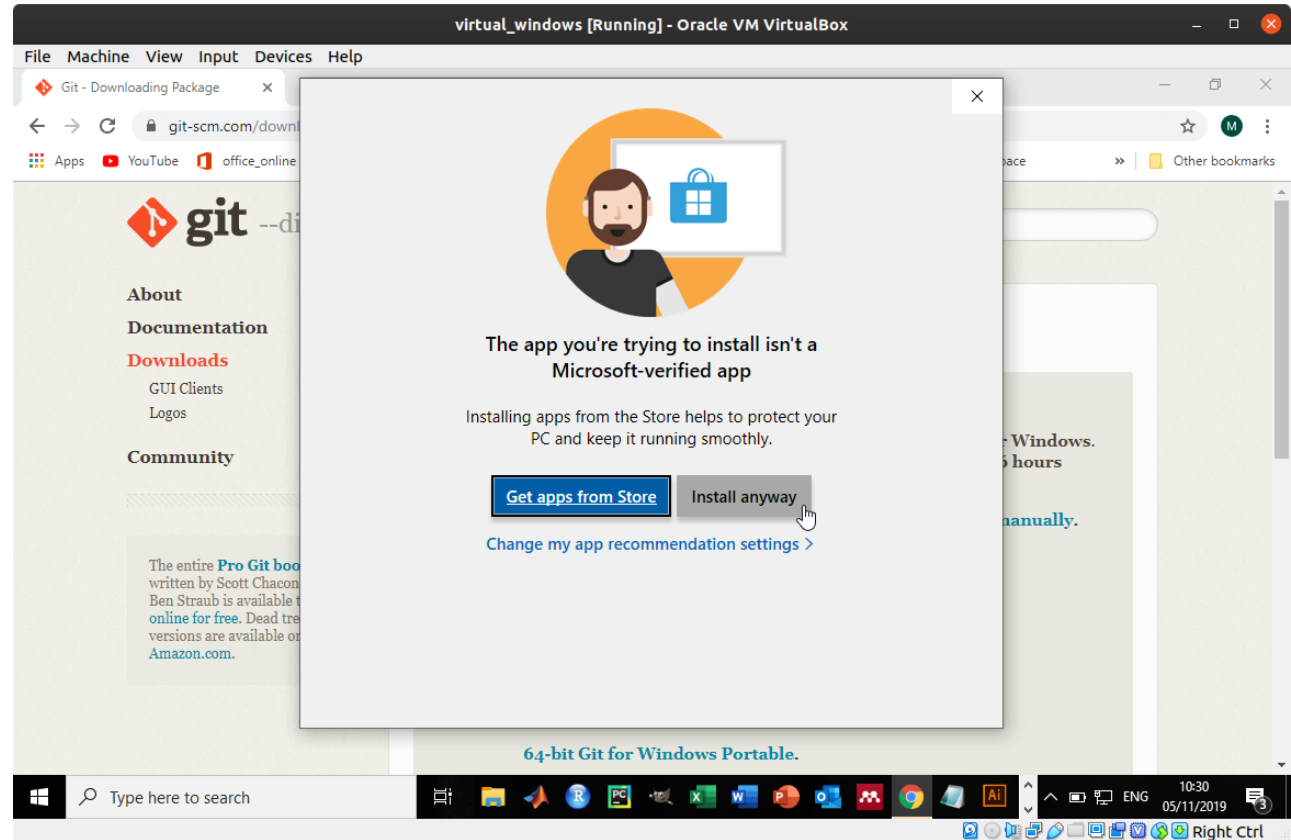
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Click on executable to start installation process

GIT INSTALLATION

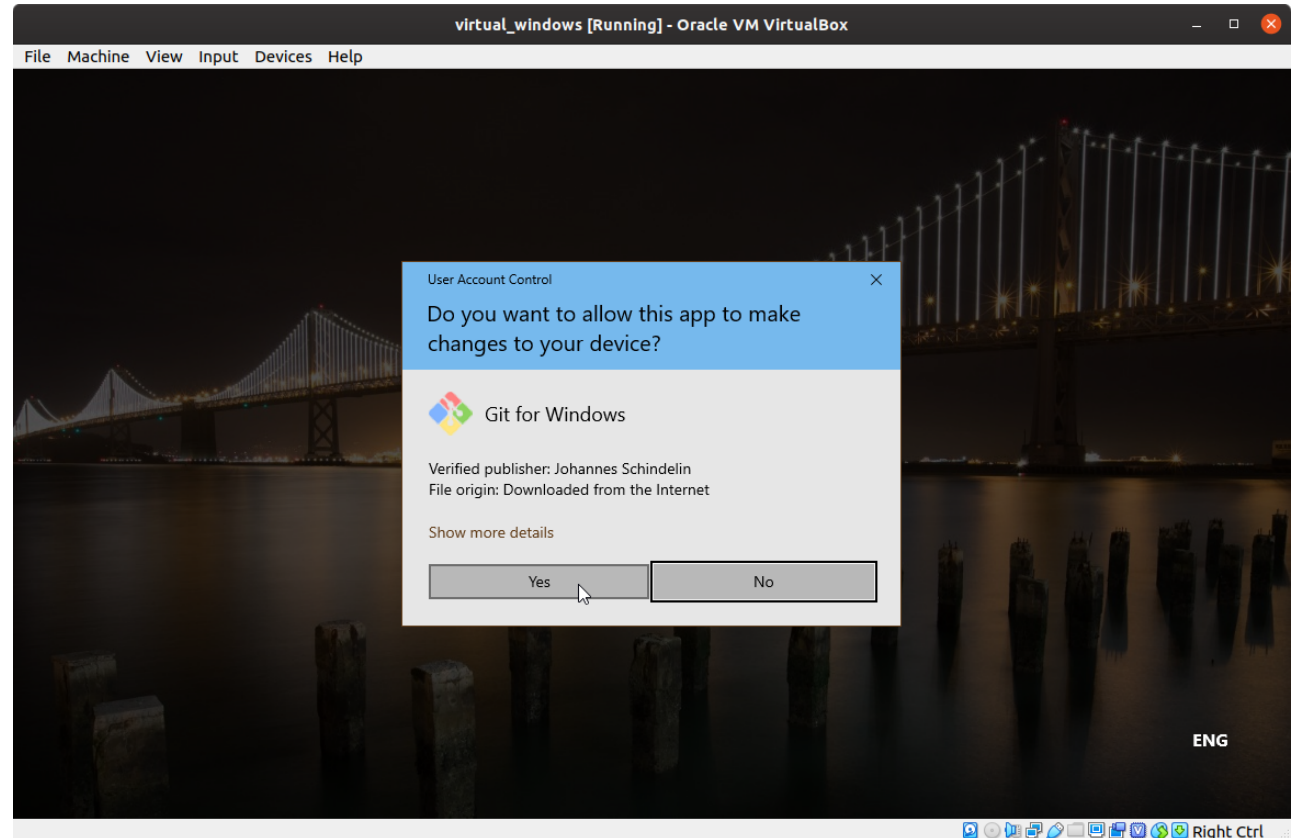
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Select Install anyway

GIT INSTALLATION

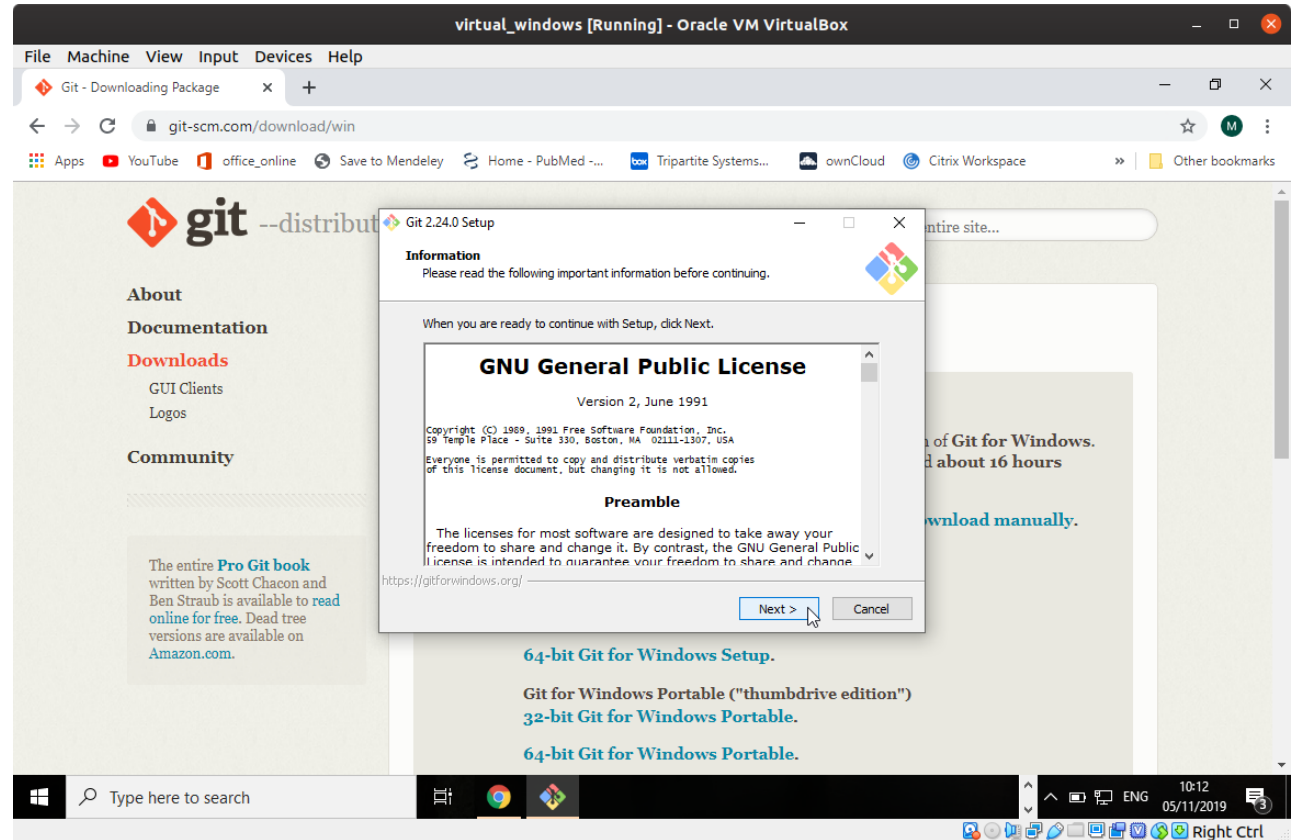
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Select Yes

GIT INSTALLATION

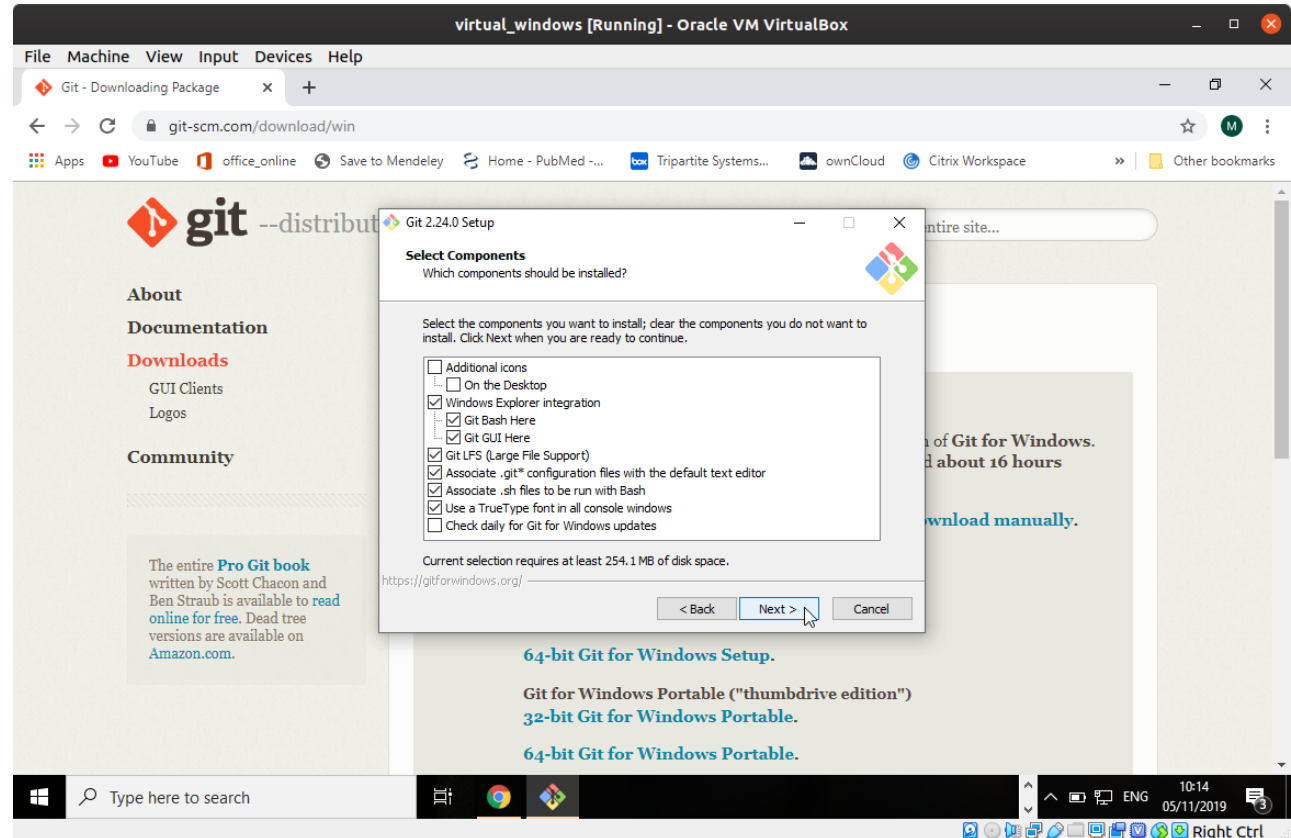
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

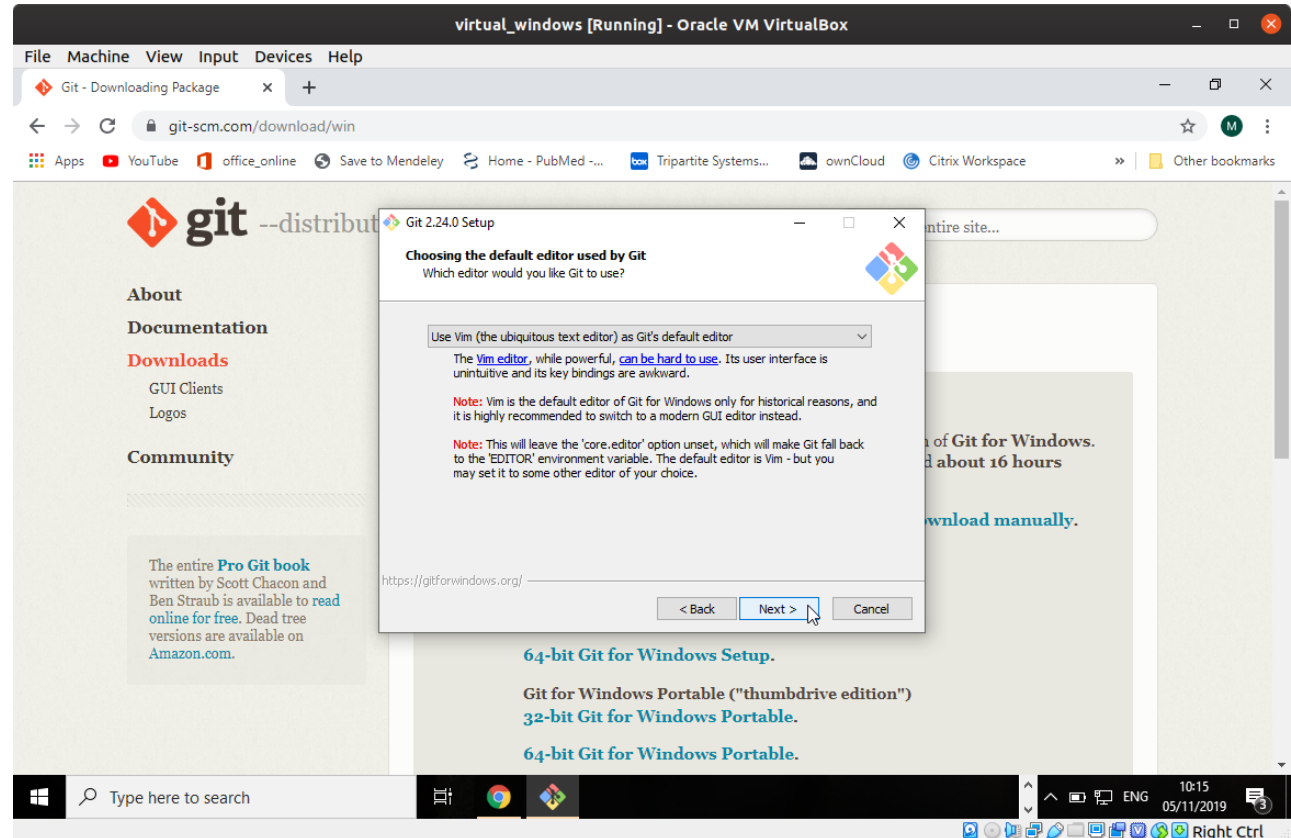
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

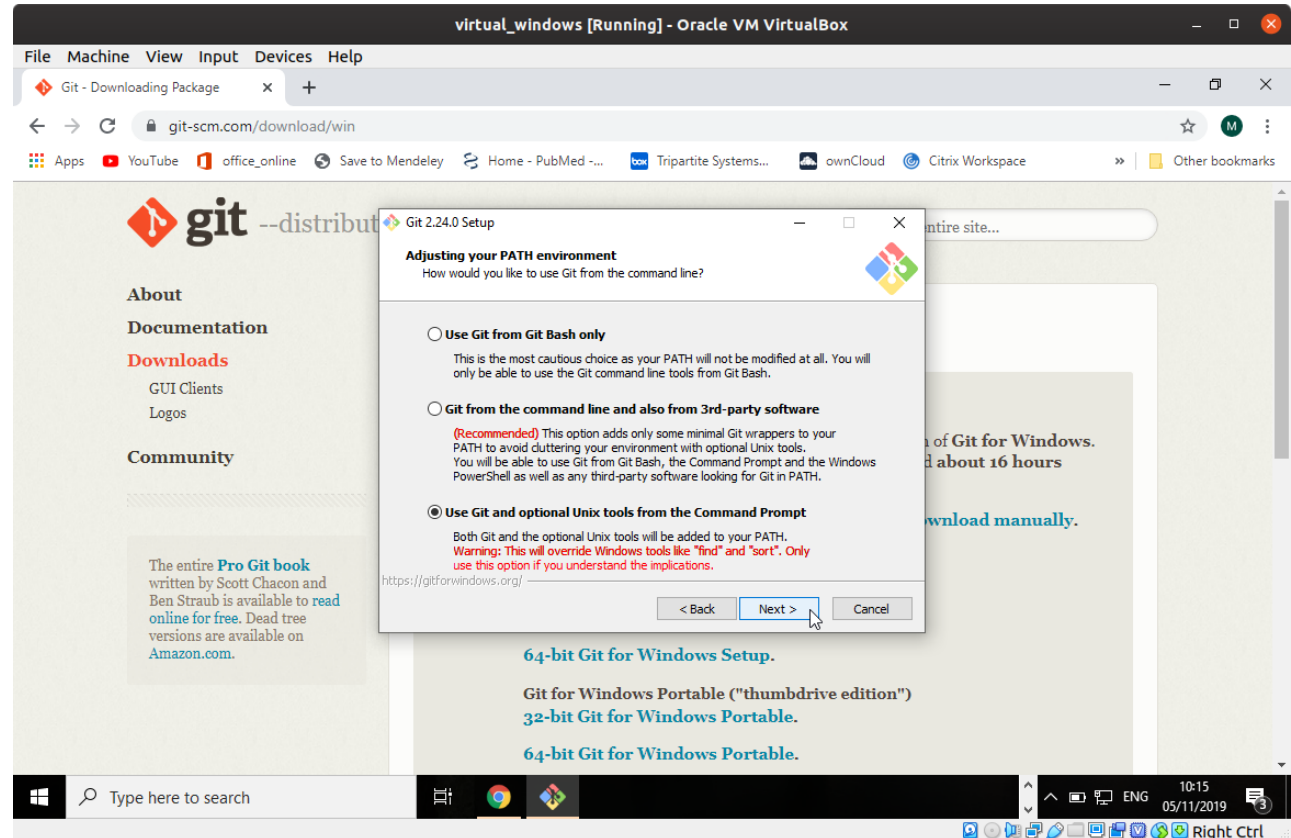
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

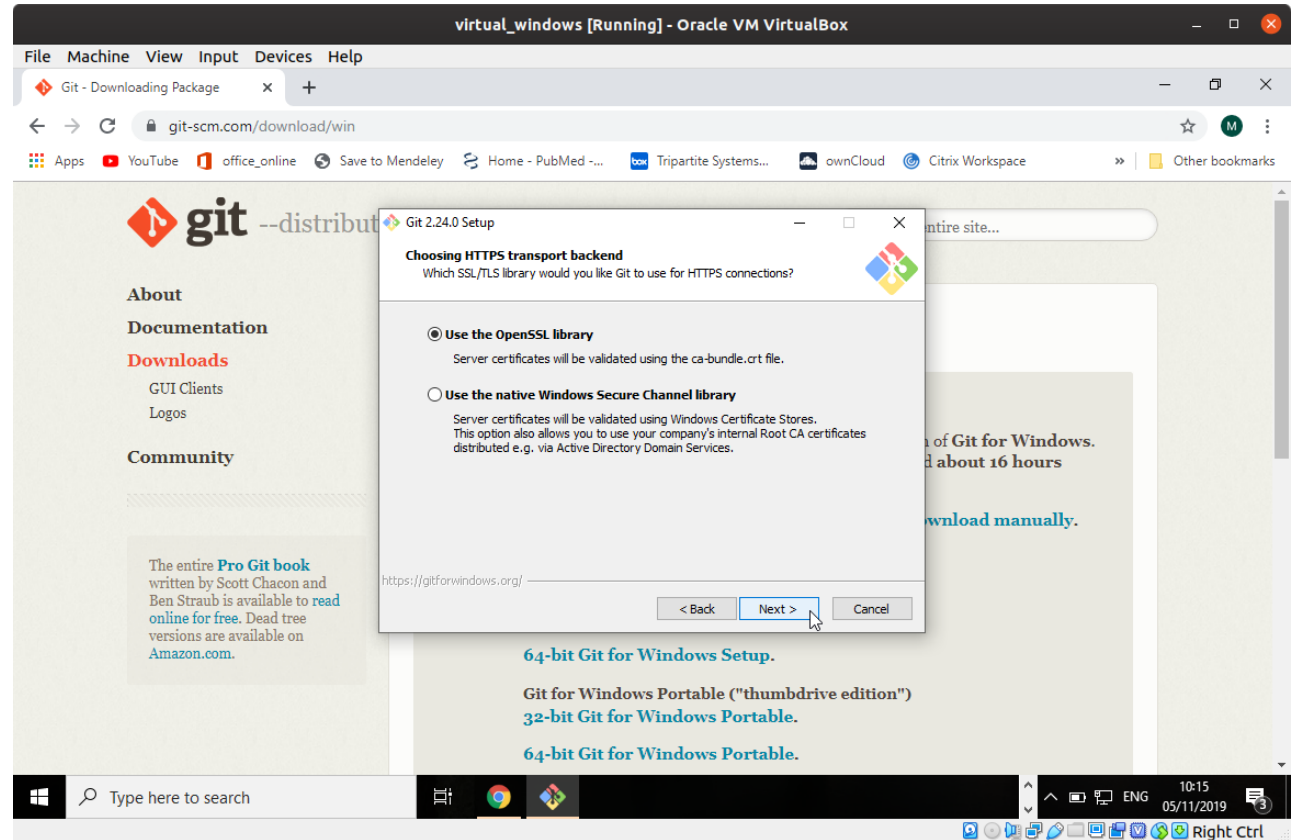
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

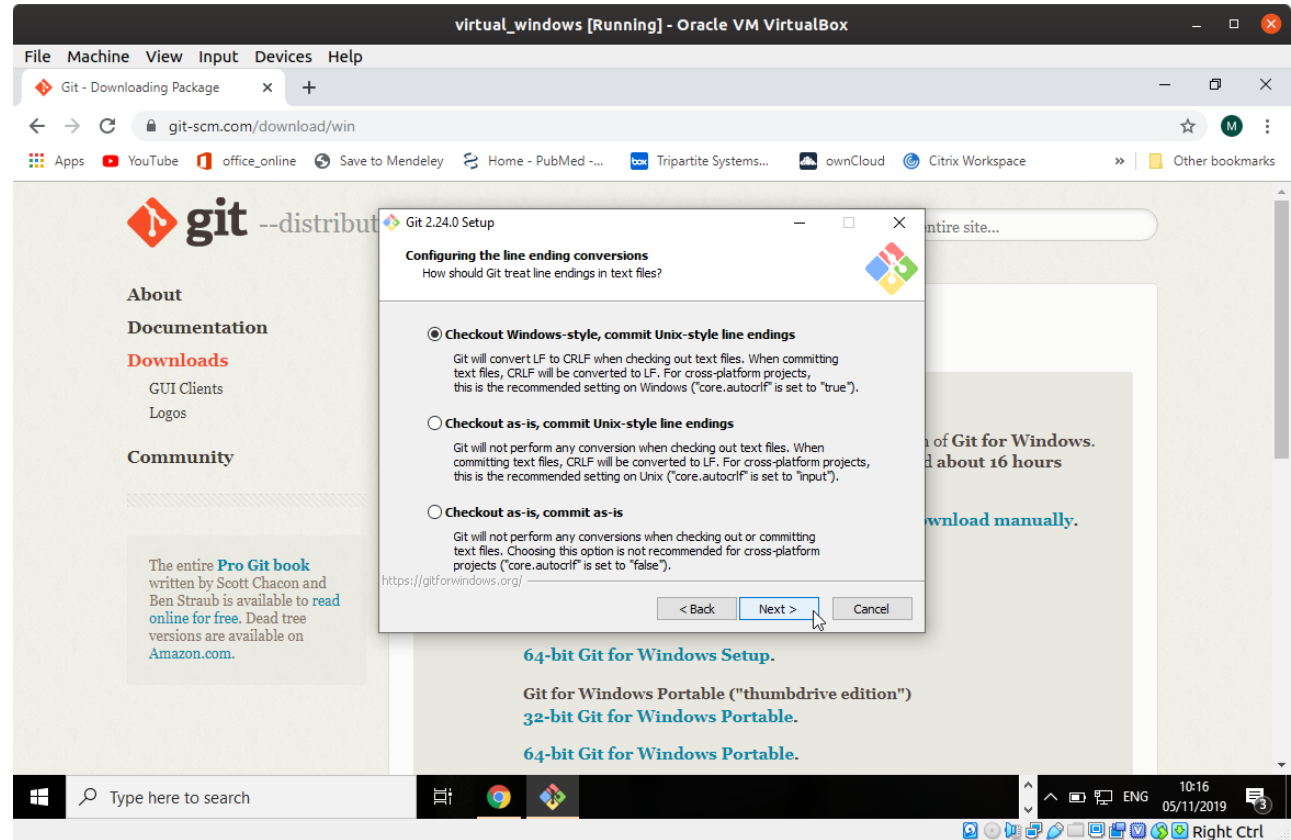
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

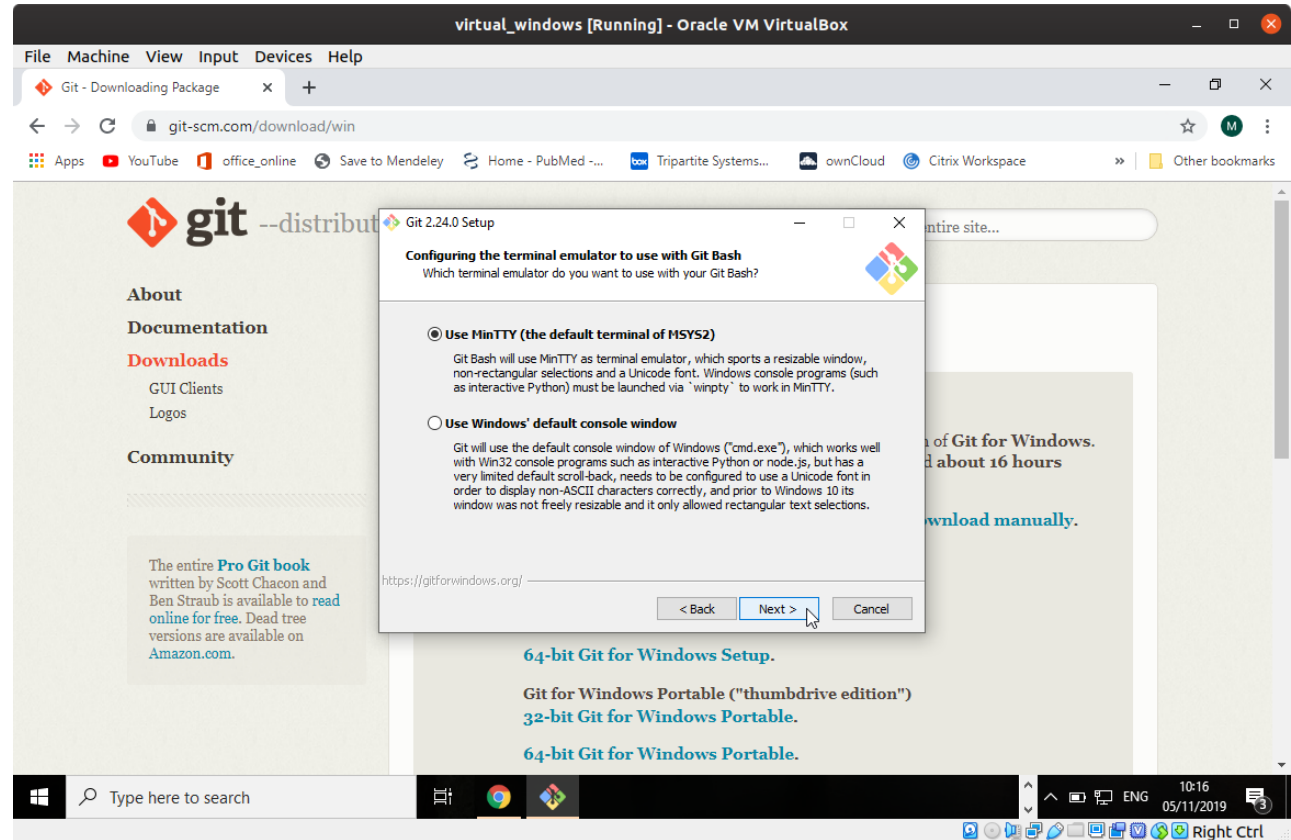
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

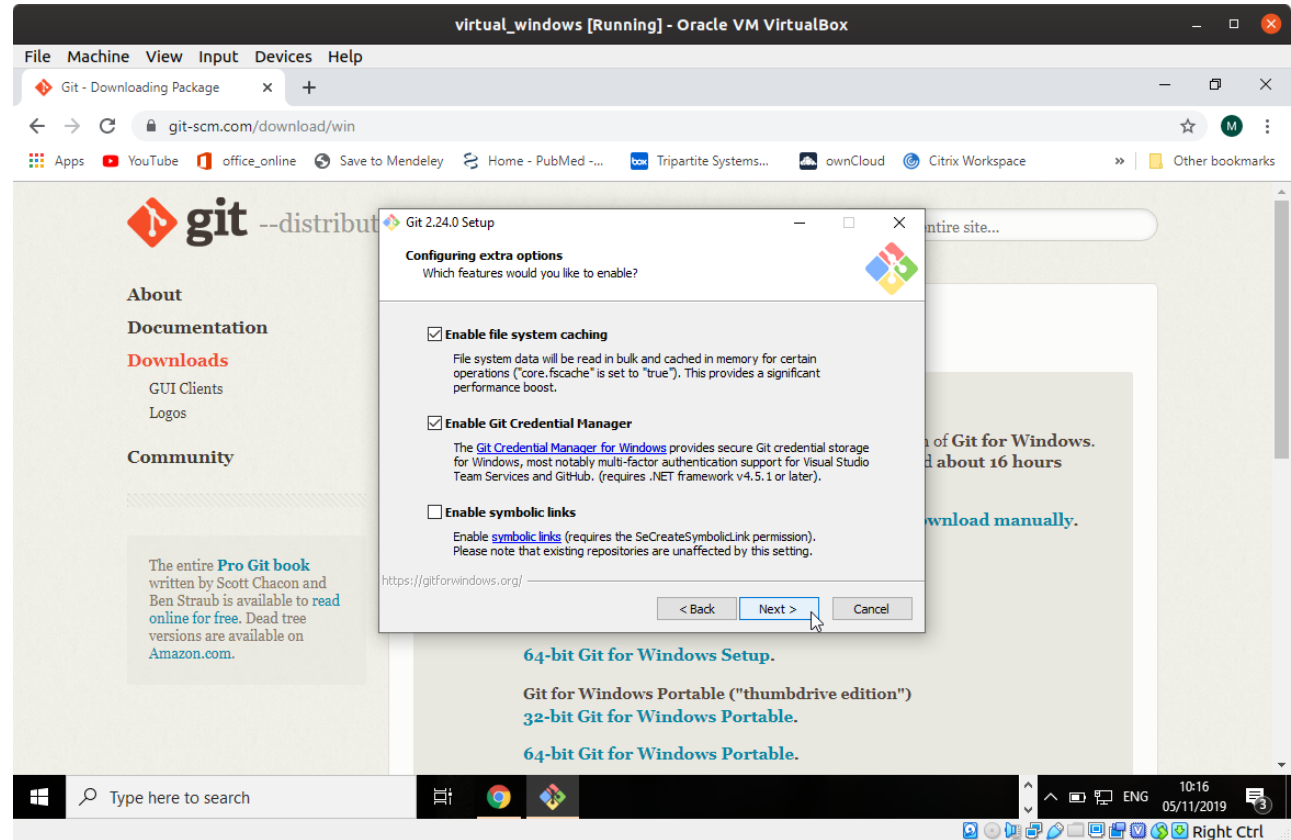
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

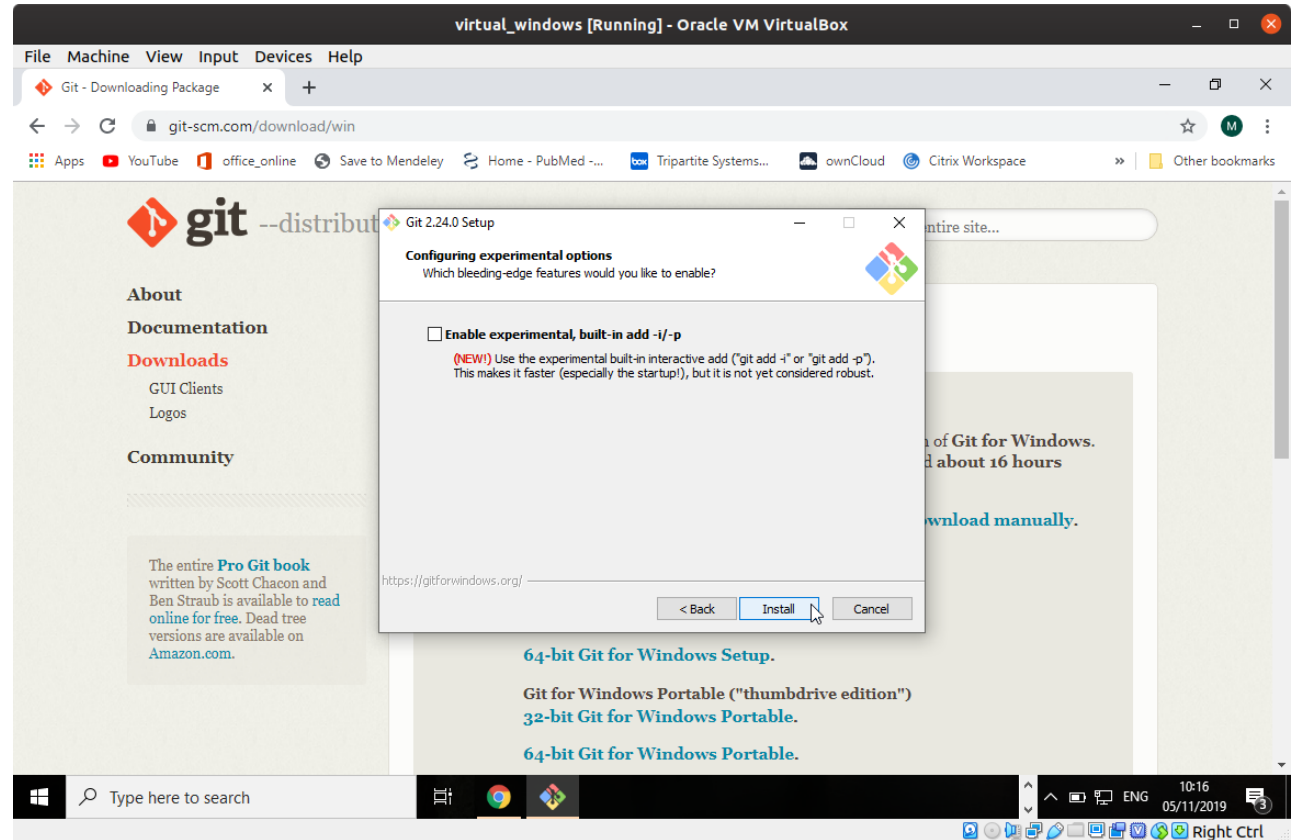
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Next

GIT INSTALLATION

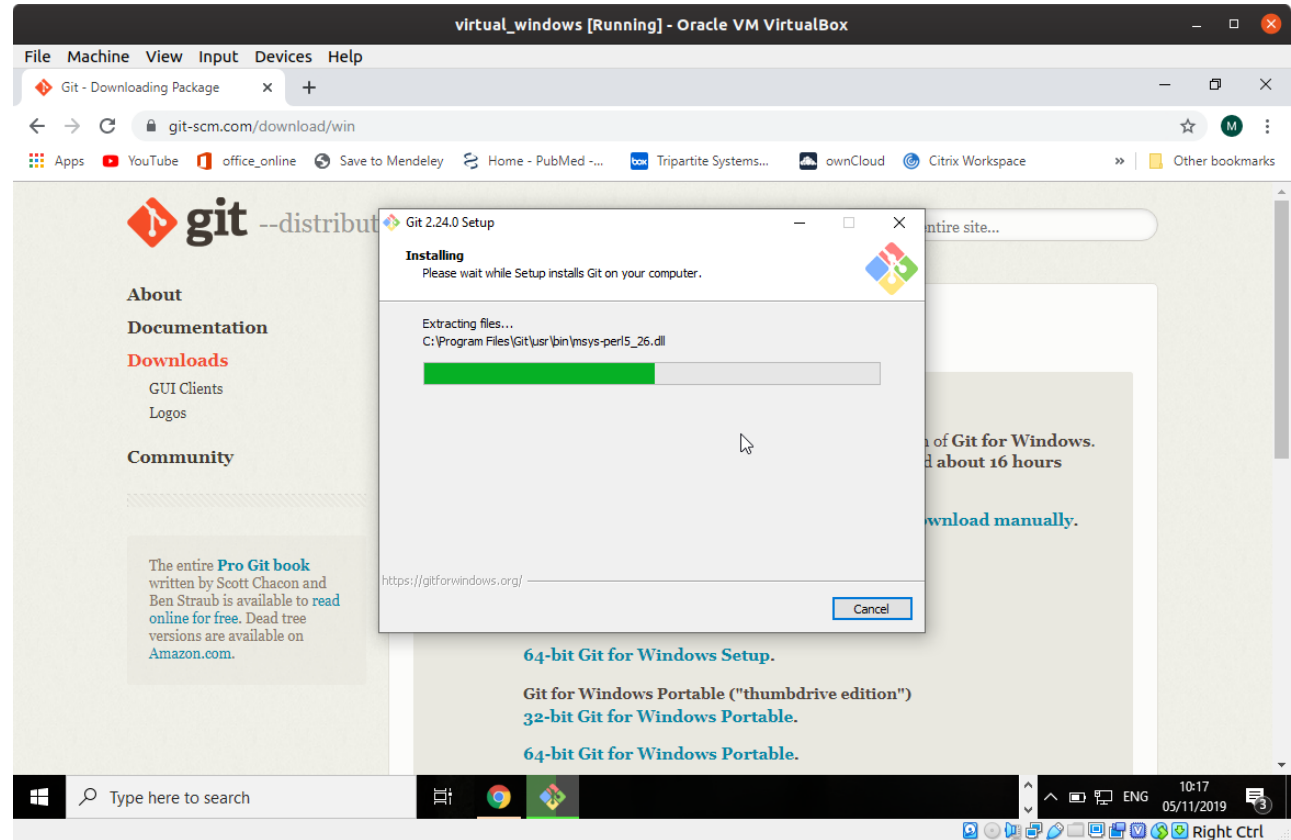
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Accept default settings by clicking on Install

GIT INSTALLATION

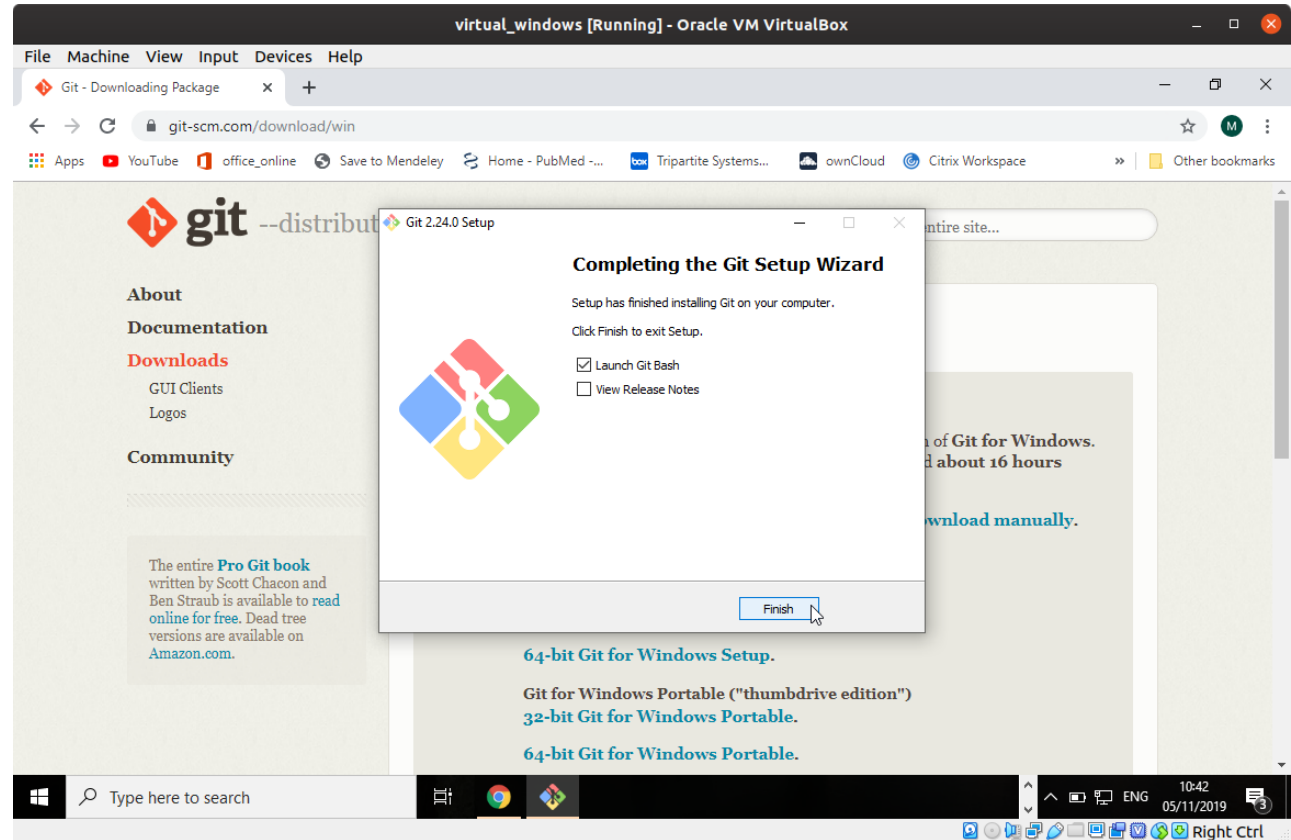
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Monitor installation progress

GIT INSTALLATION

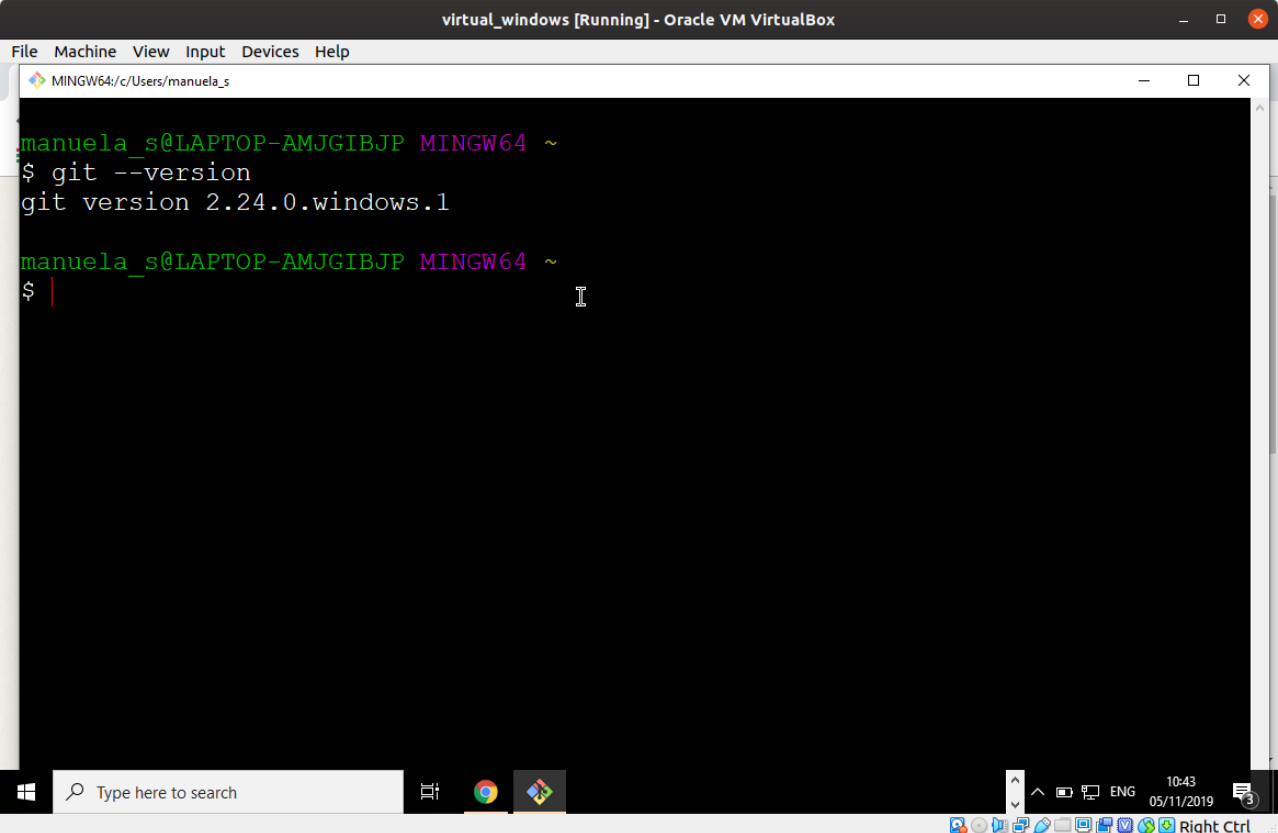
1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully



Select Launch Git Bash, Unselect View Release Notes and click on Finish

GIT INSTALLATION

1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings
4. Verify installation completed successfully

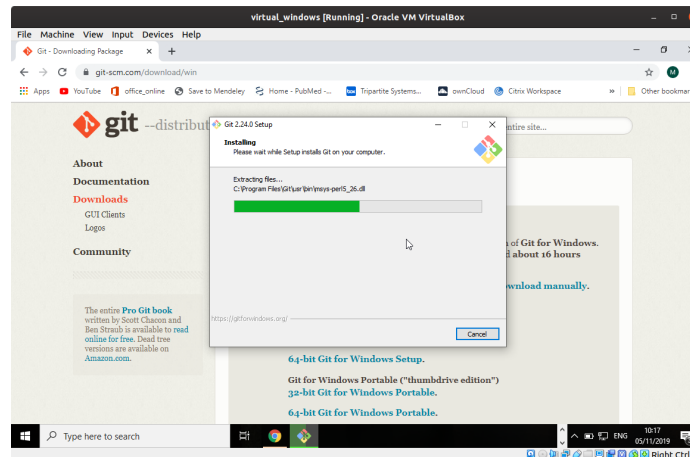


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64:/c:/Users/manuela_s
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~
$ git --version
git version 2.24.0.windows.1
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~
$ |
```

Verify installation completed successfully

NOW YOU INSTALL GIT ON YOUR COMPUTER (5-10 MIN)

1. Go to <https://git-scm.com/>
2. Download executable in suggested directory
3. Install by following step-by-step instructions and accepts default settings



Signal once installation progress has started

DEMO

DEMO

To demonstrate we are going to go through an example for writing a manuscript.

- We will track the history of our manuscript and accompanying files in git
- We will use git to see the history of our files and to undo a mistake
- We will use git to synchronize the files between multiple computers and to collaborate with other authors

WRITING PAPERS WITH MARKDOWN

“Markdown is a lightweight markup language with plain text formatting syntax”

(Wikipedia)

- Markdown text (.md extension) can be converted to other formats (.docx, .pdf, .html) with [Pandoc](#)
- References can also be stored in plain text files (.bib)
- Learn more about markdown [here](#)
- [Try it online](#)

WRITING PAPERS WITH MARKDOWN - EXAMPLE

Markdown manuscript

```
---
title: 'HCP: A Matlab package to create beautiful
  heatmaps with richly annotated covariates'
authors:
  - name: Manuela Salvucci
    orcid: 0000-0001-9941-4307
    affiliation: 1
  - name: Jochen H. M. Prehn
    orcid: 0000-0003-3479-7794
    affiliation: 1
affiliations:
  - name: Centre for Systems Medicine, Department of
    Physiology and Medical Physics, Royal College of
    Surgeons in Ireland, Dublin, Ireland
    index: 1
date: 20 January 2019
bibliography: paper.bib
---

# Summary
A heatmap is a graphical technique that maps 2-
dimensional matrices of numerical values to colors
to provide an immediate
and intuitive visualization of the underlying patterns
[@Eisen1998]. Heatmaps are often used in conjunction
```

Markdown pdf



HCP: A Matlab package to create beautiful heatmaps
with richly annotated covariates

Manuela Salvucci¹ and Jochen H. M. Prehn¹

¹ Centre for Systems Medicine, Department of Physiology and Medical Physics, Royal College of Surgeons in Ireland, Dublin, Ireland

DOI: [10.21105/joss.01291](https://doi.org/10.21105/joss.01291)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Submitted: 26 January 2019

Published: 12 June 2019

License

Authors of papers retain
copyright and release the work
under a Creative Commons
Attribution 4.0 International
License ([CC-BY](#)).

Summary

A heatmap is a graphical technique that maps 2-dimensional matrices of numerical values to colors to provide an immediate and intuitive visualization of the underlying patterns (Eisen, Spellman, Brown, & Botstein, 1998). Heatmaps are often used in conjunction with cluster analysis to re-order observations and/or features by similarity and thus, rendering common and distinct patterns more apparent. When generating these visualizations, it is often of interest to interpret the underlying patterns in the context of other data sources. In the field of bioinformatics, heatmaps are frequently used to visualize high-throughput and high-dimensional datasets, such as those derived from profiling biological samples with *-omic* technologies (whole genome sequencing, transcriptomics and proteomics). Often, biological samples (for example, patient tumour samples) are characterized at multiple *-omic* level and it is of interest to contrast and compare patterns captured at the different molecular layers along with their

GETTING STARTED

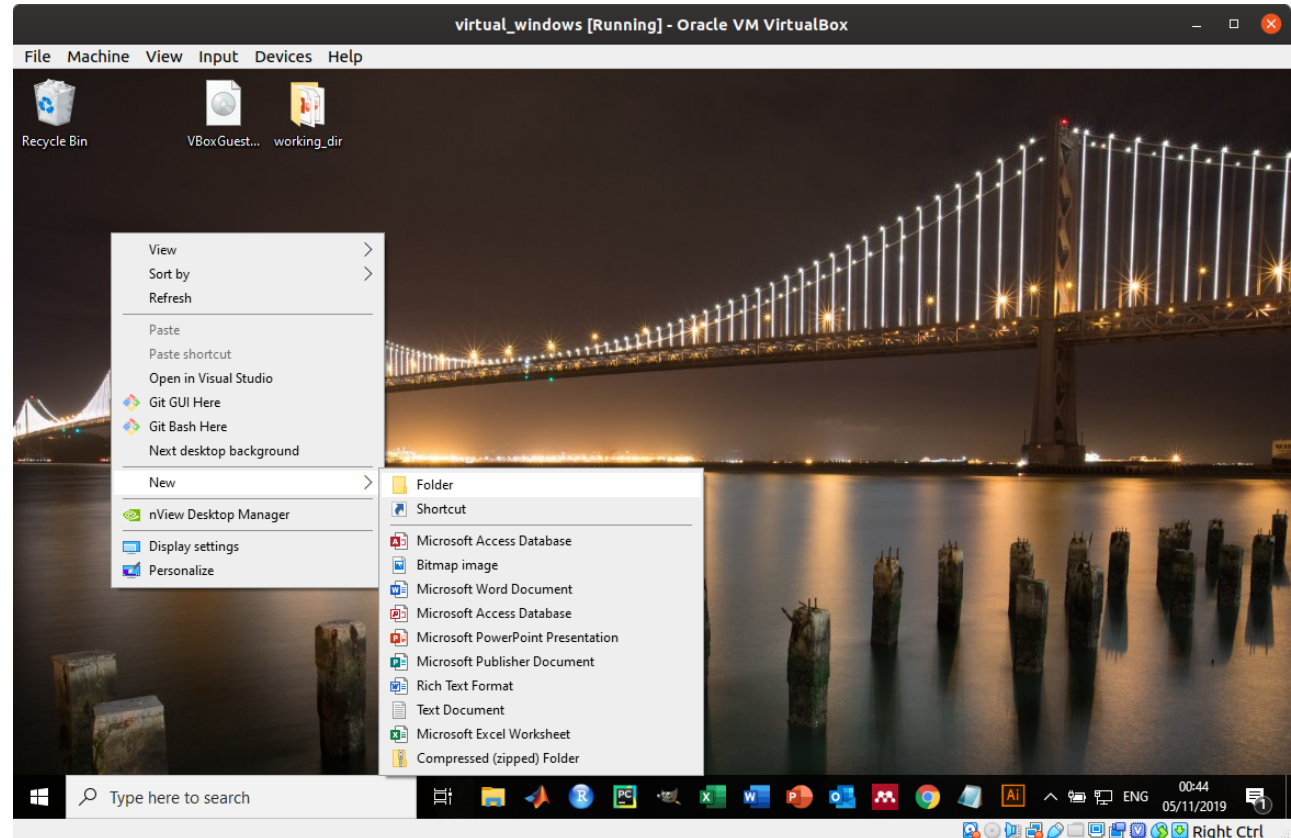
Two main approaches to get a git repository:

- start a repository from scratch -> **git init**
- start by cloning an existing repository -> **git clone**

PRACTICAL EXAMPLE

PRACTICAL EXAMPLE

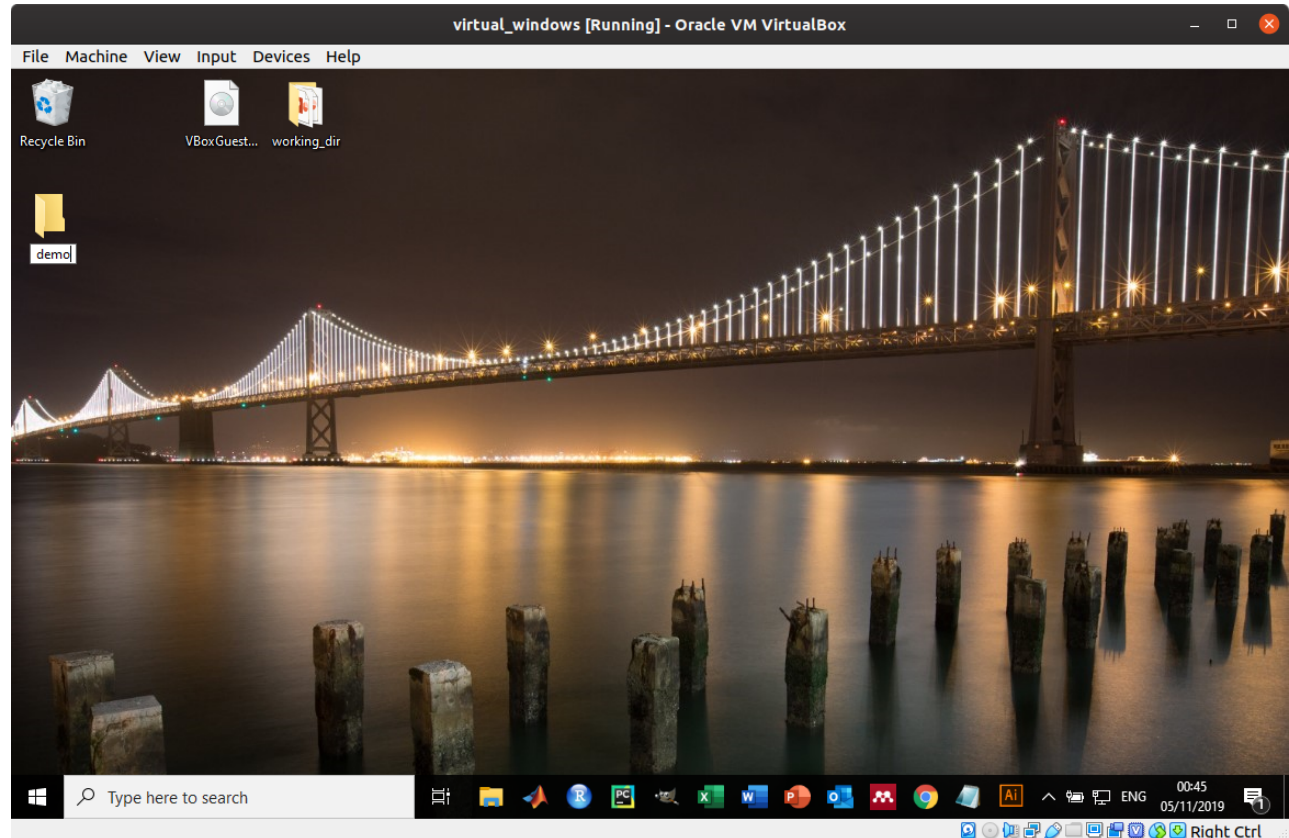
1. **Make a project folder**
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



Make a project folder

PRACTICAL EXAMPLE

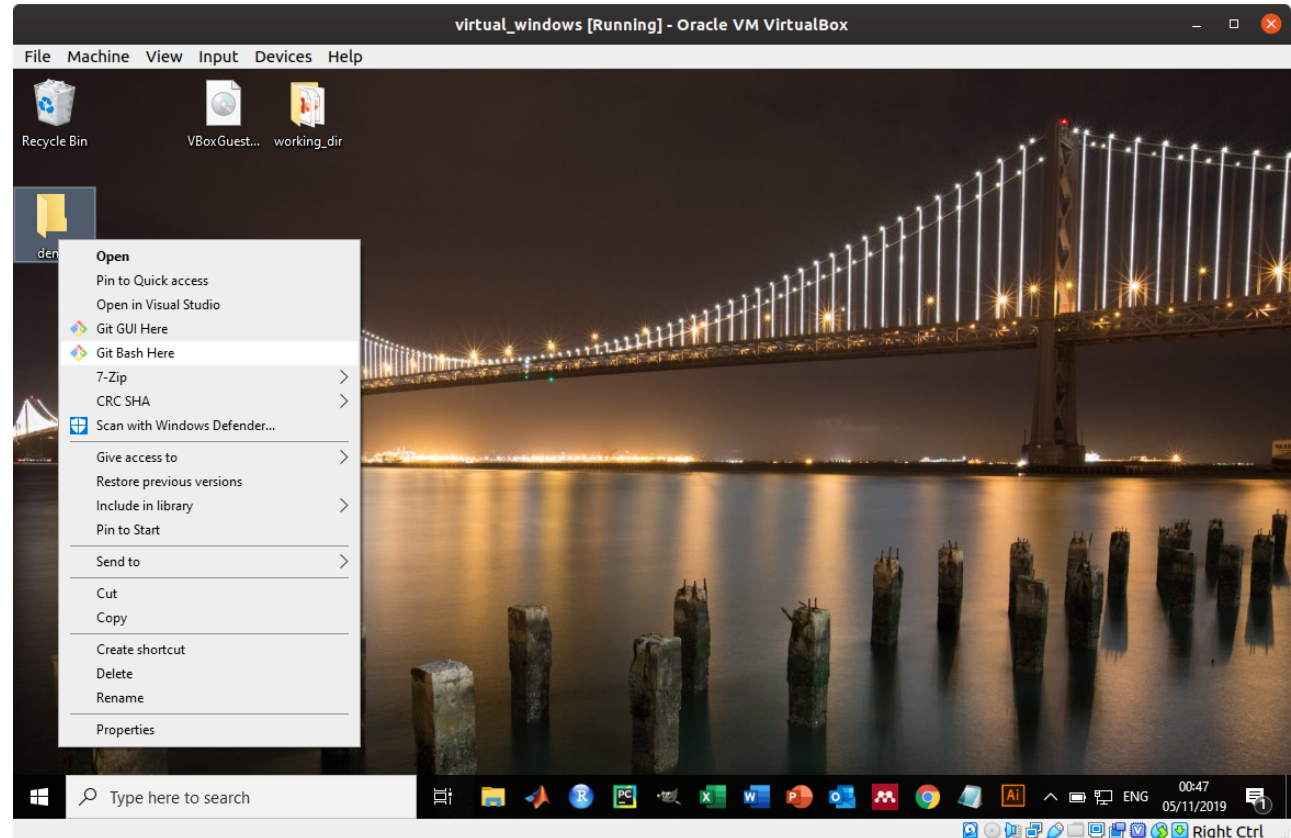
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



Name it demo

PRACTICAL EXAMPLE

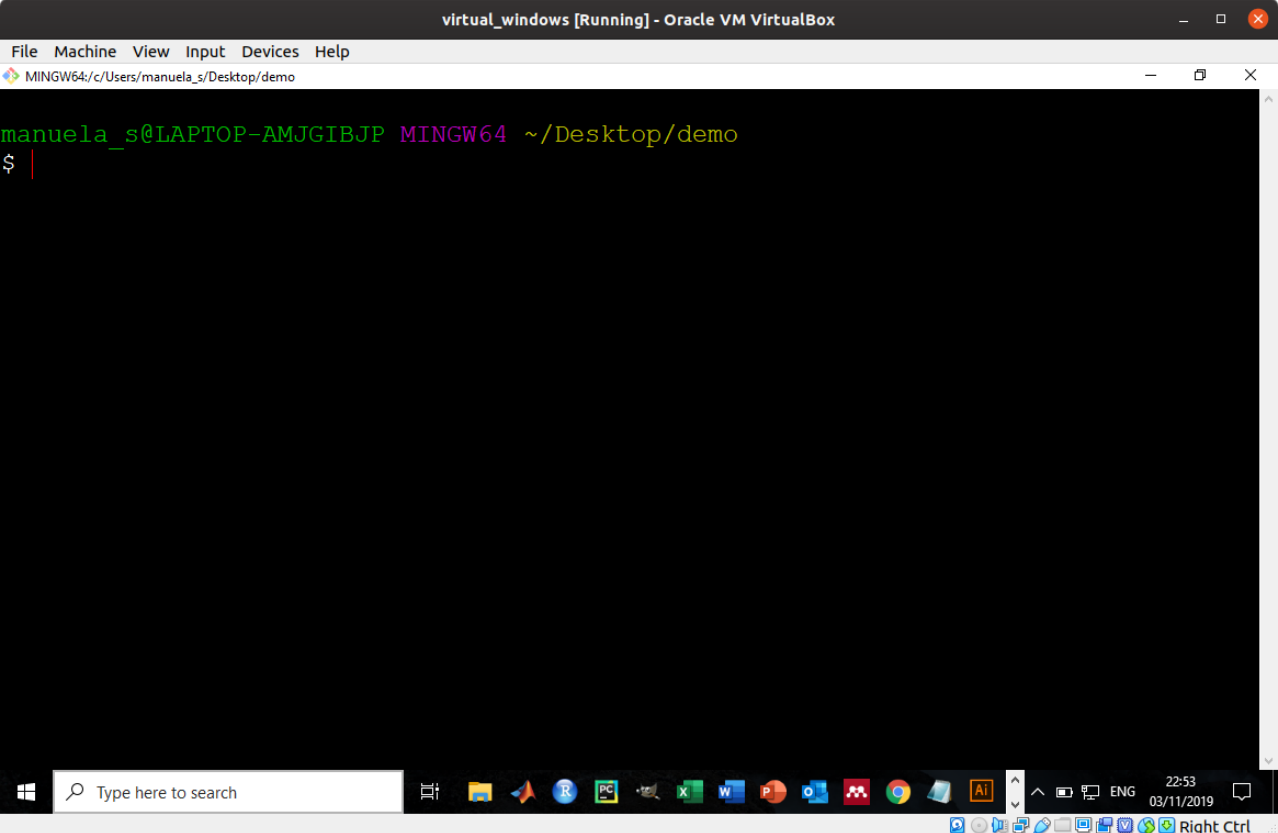
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



Open GIT bash

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

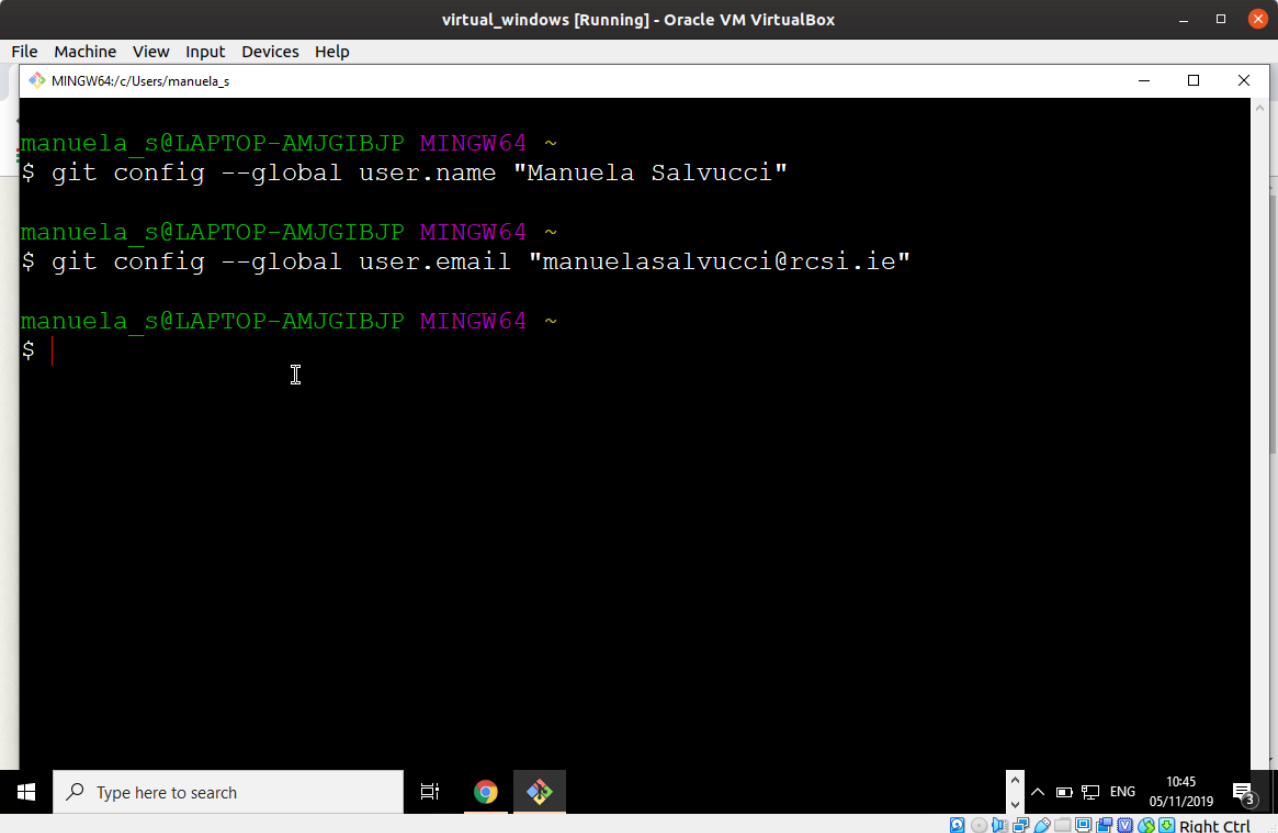


The screenshot shows a virtual machine window titled "virtual_windows [Running] - Oracle VM VirtualBox". Inside the VM, there is a terminal window titled "MINGW64/c/Users/manuela_s/Desktop/demo". The terminal prompt is "manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo" followed by a dollar sign "\$" and a cursor. The Windows taskbar is visible at the bottom, showing the search bar and various application icons. The system tray on the right shows the time as 22:53 and the date as 03/11/2019.

GIT bash

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

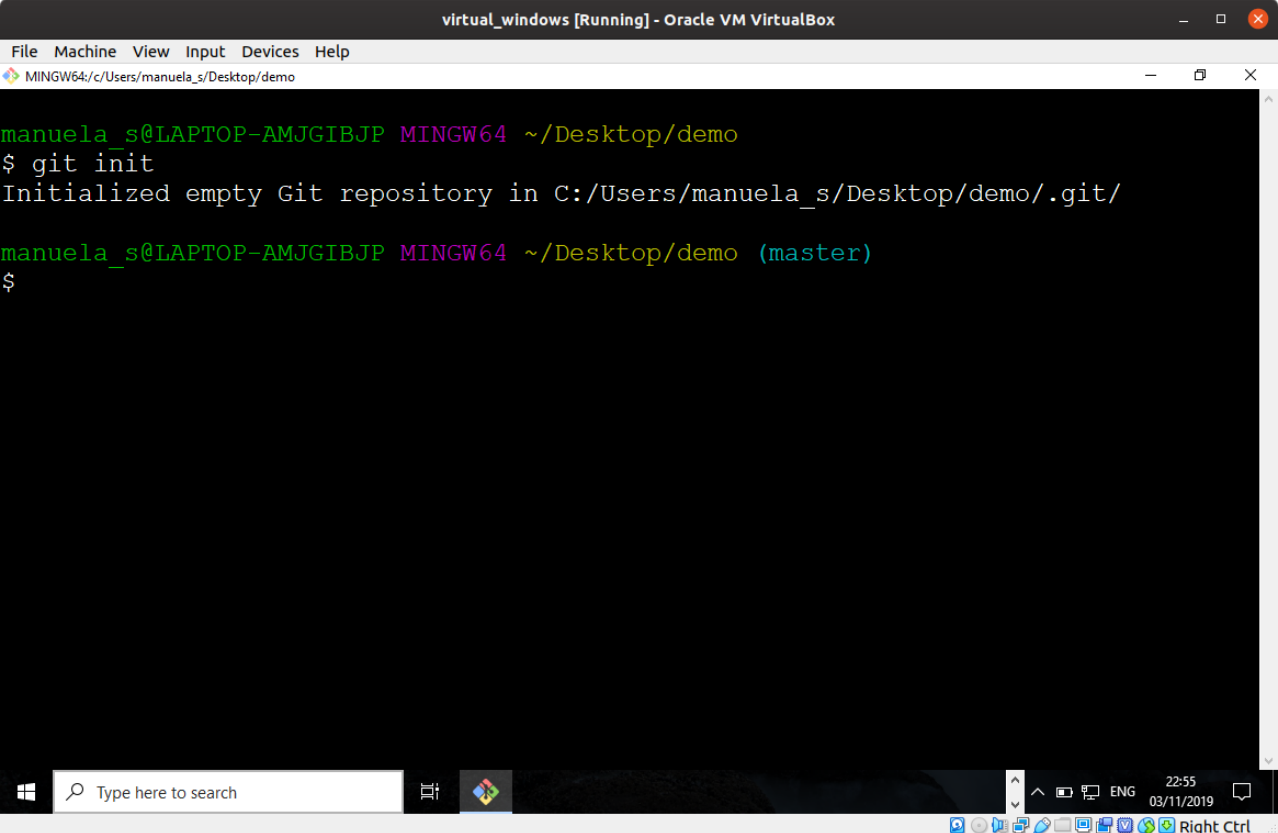


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64:/c:/Users/manuela_s
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~
$ git config --global user.name "Manuela Salvucci"
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~
$ git config --global user.email "manuelasalvucci@rcsi.ie"
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~
$ |
```

Configure GIT

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

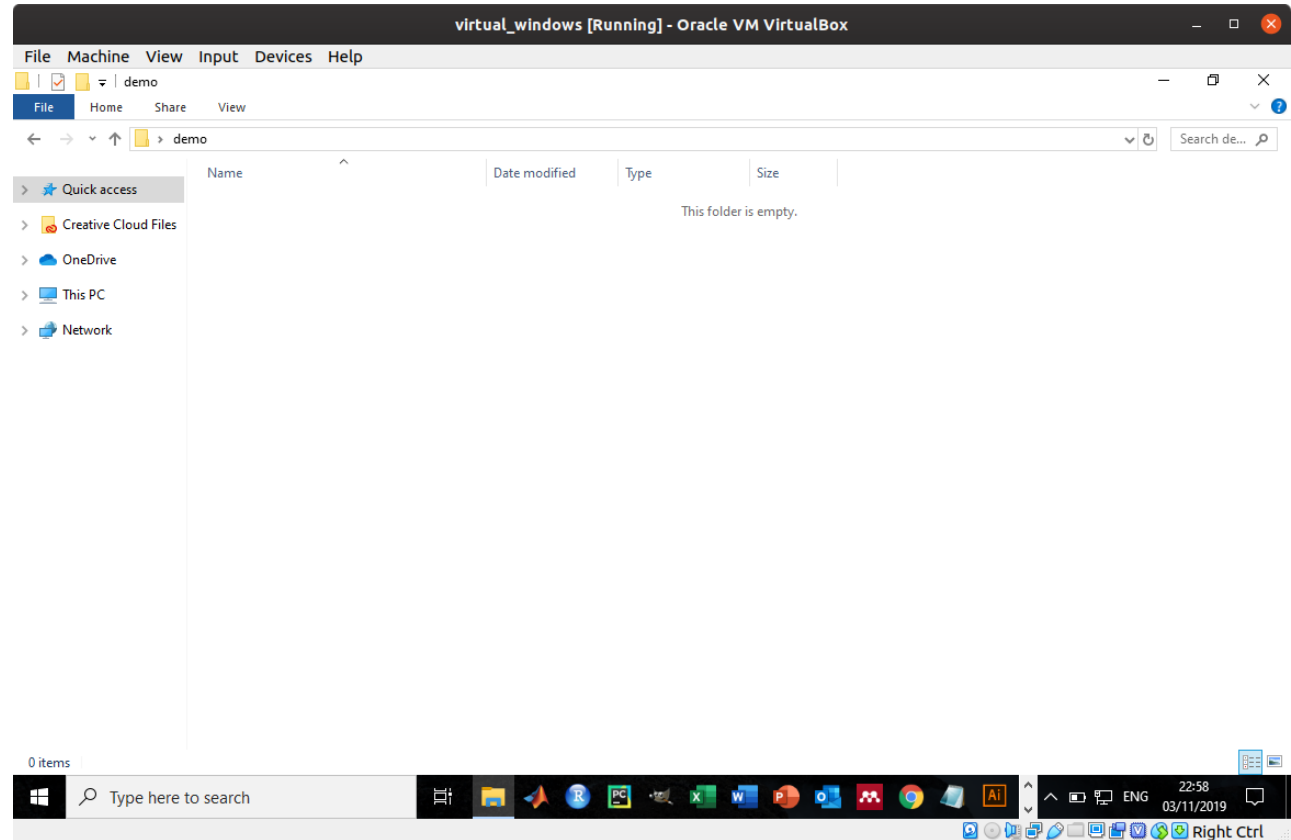


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo
$ git init
Initialized empty Git repository in C:/Users/manuela_s/Desktop/demo/.git/
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

Initialize repository

PRACTICAL EXAMPLE

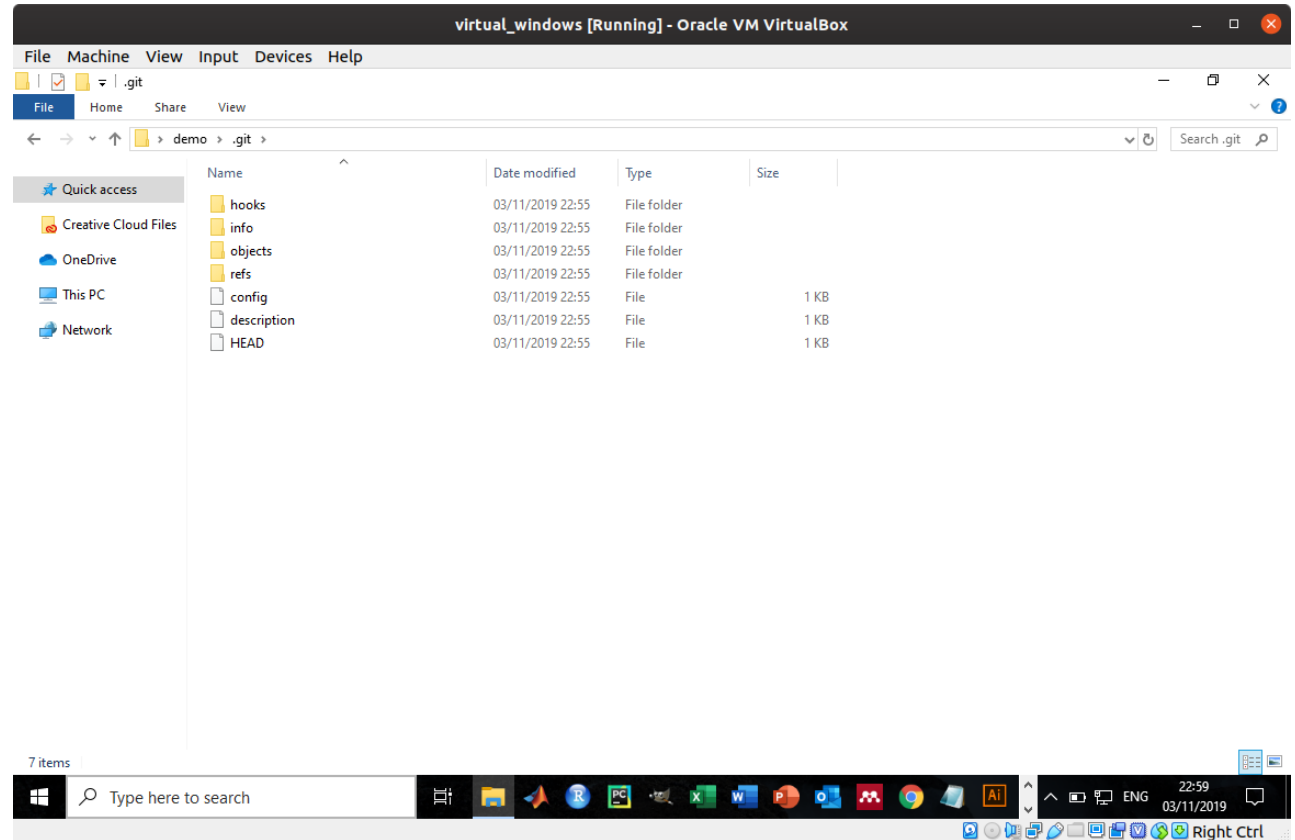
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



The folder still looks empty after git init. There is a hidden .git directory that you can normally not see

PRACTICAL EXAMPLE

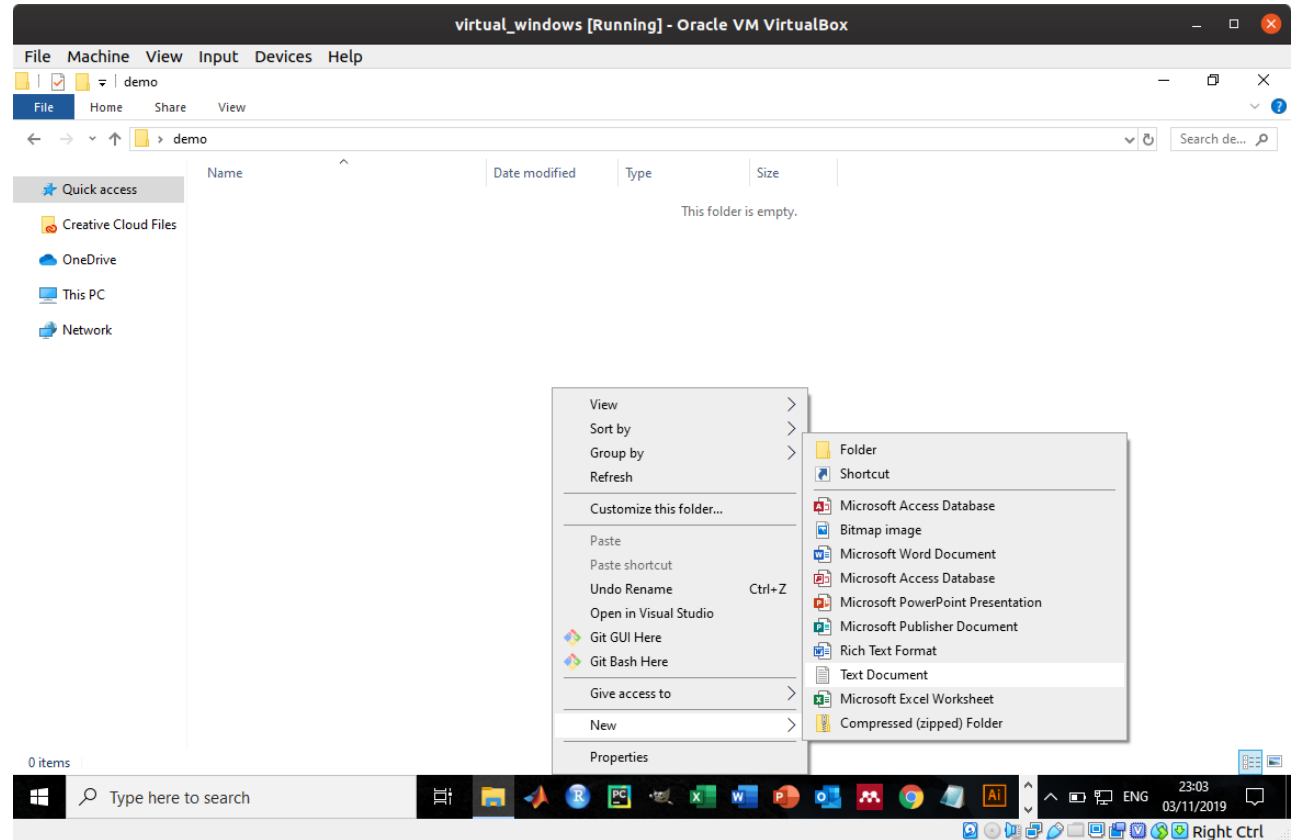
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



If you explicitly open the .git subdirectory, you can see a lot of files internal to GIT. You do not need to directly interact with these files (and do not delete them)

PRACTICAL EXAMPLE

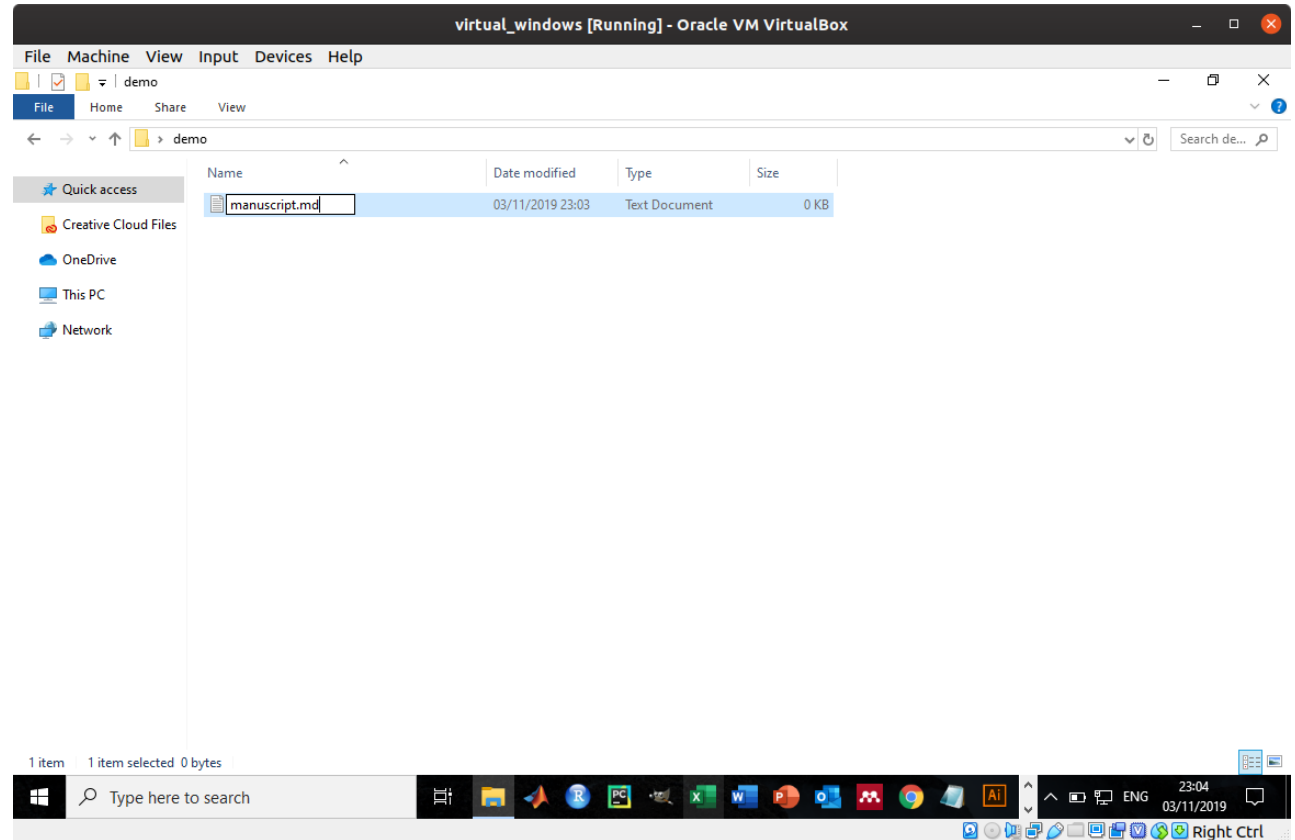
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



Create a new text document for a manuscript we are writing

PRACTICAL EXAMPLE

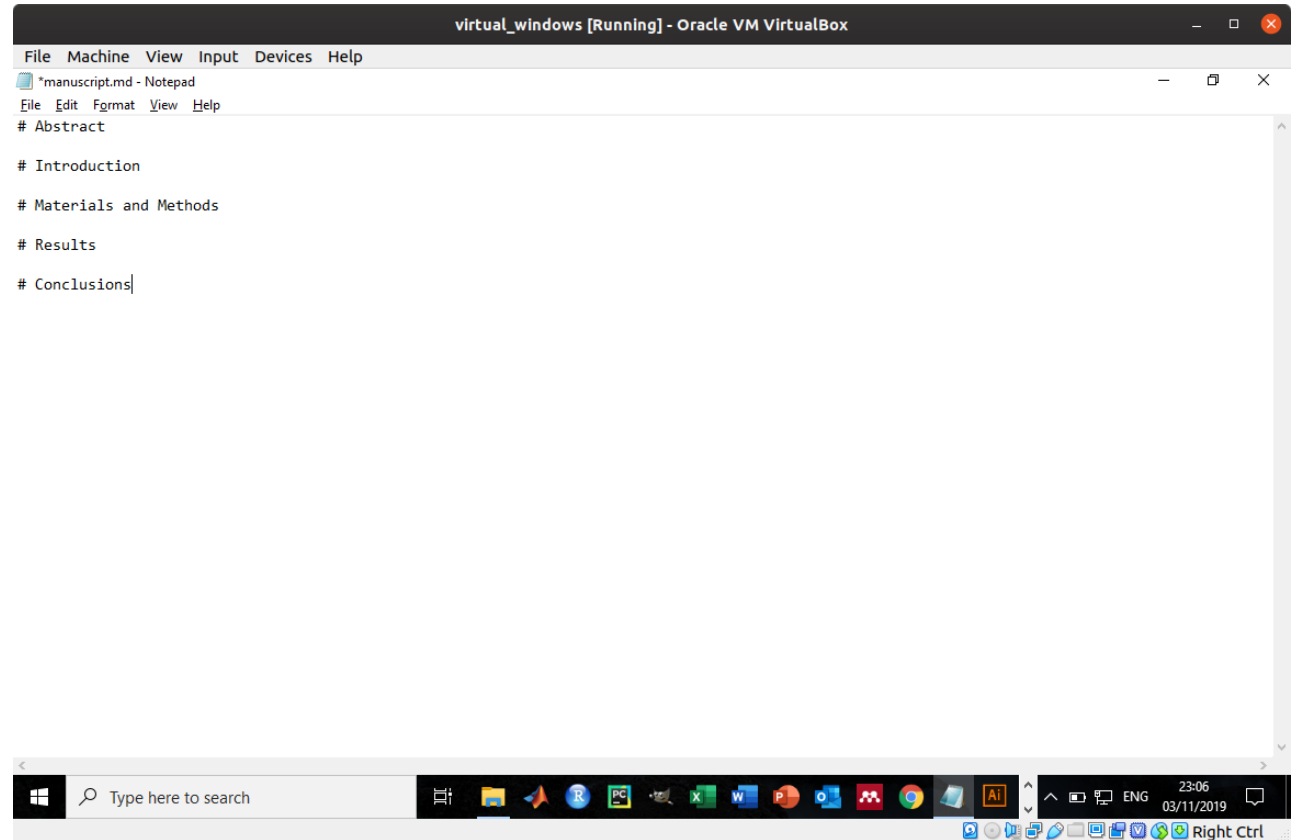
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



Rename the file to manuscript.md to indicate that the file is formatted with markdown

PRACTICAL EXAMPLE

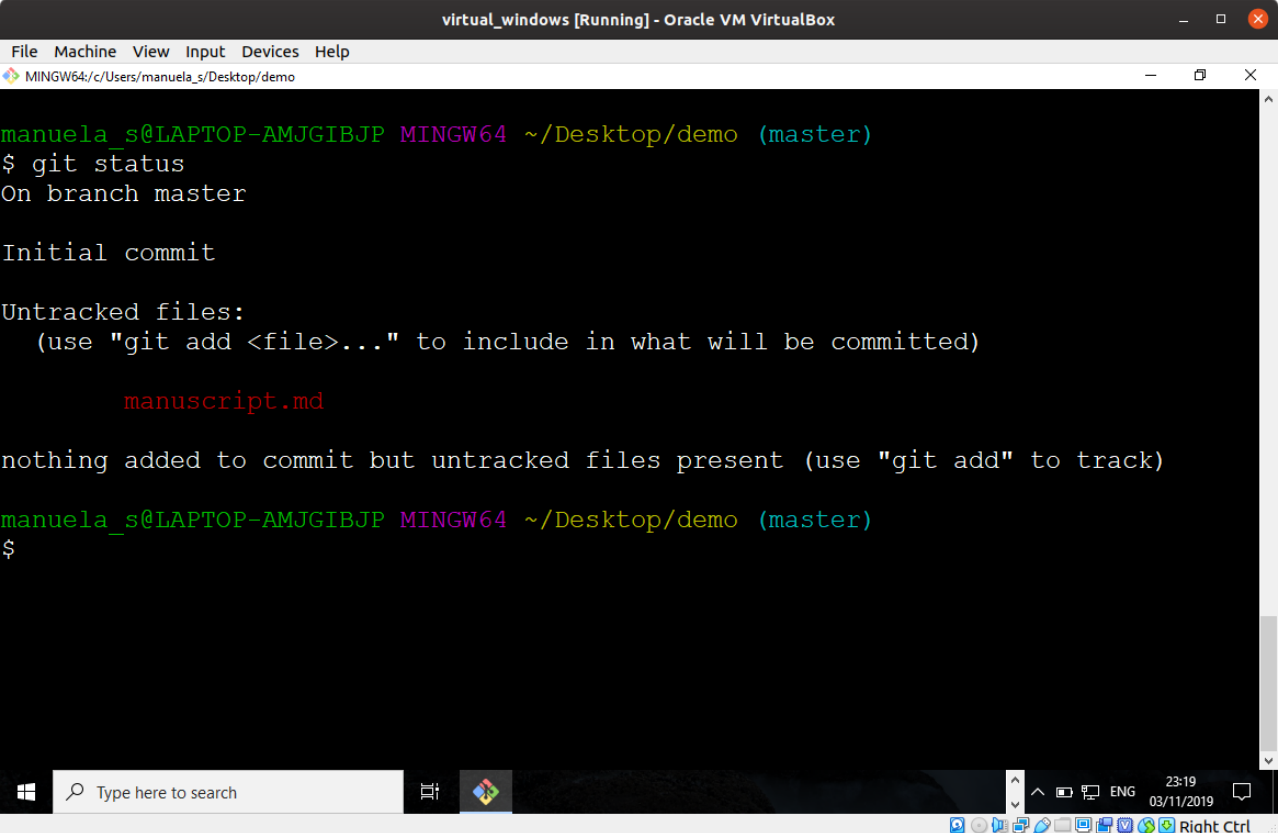
1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



First manuscript draft

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git status
On branch master

Initial commit

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

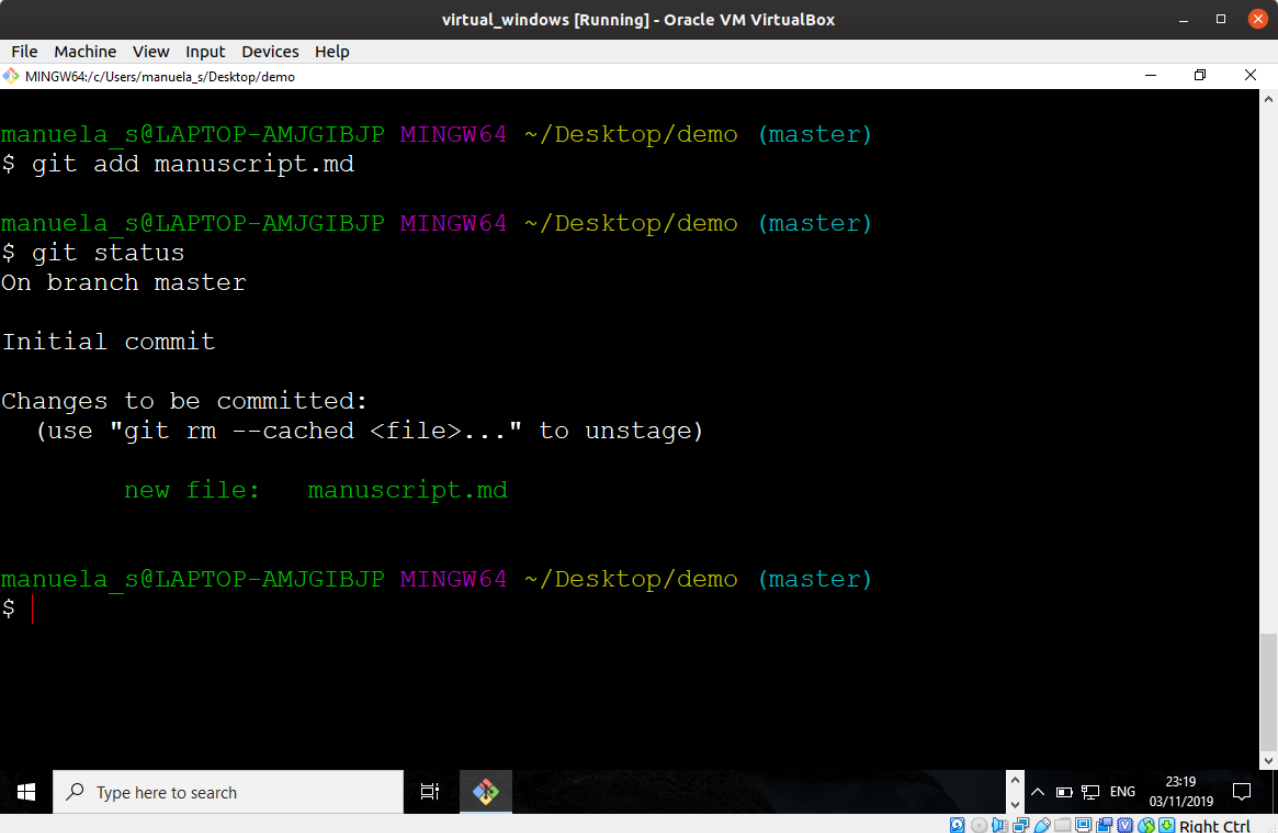
       manuscript.md

nothing added to commit but untracked files present (use "git add" to track)
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

We can use git status command to see what the repository status is

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git add manuscript.md
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git status
On branch master

Initial commit

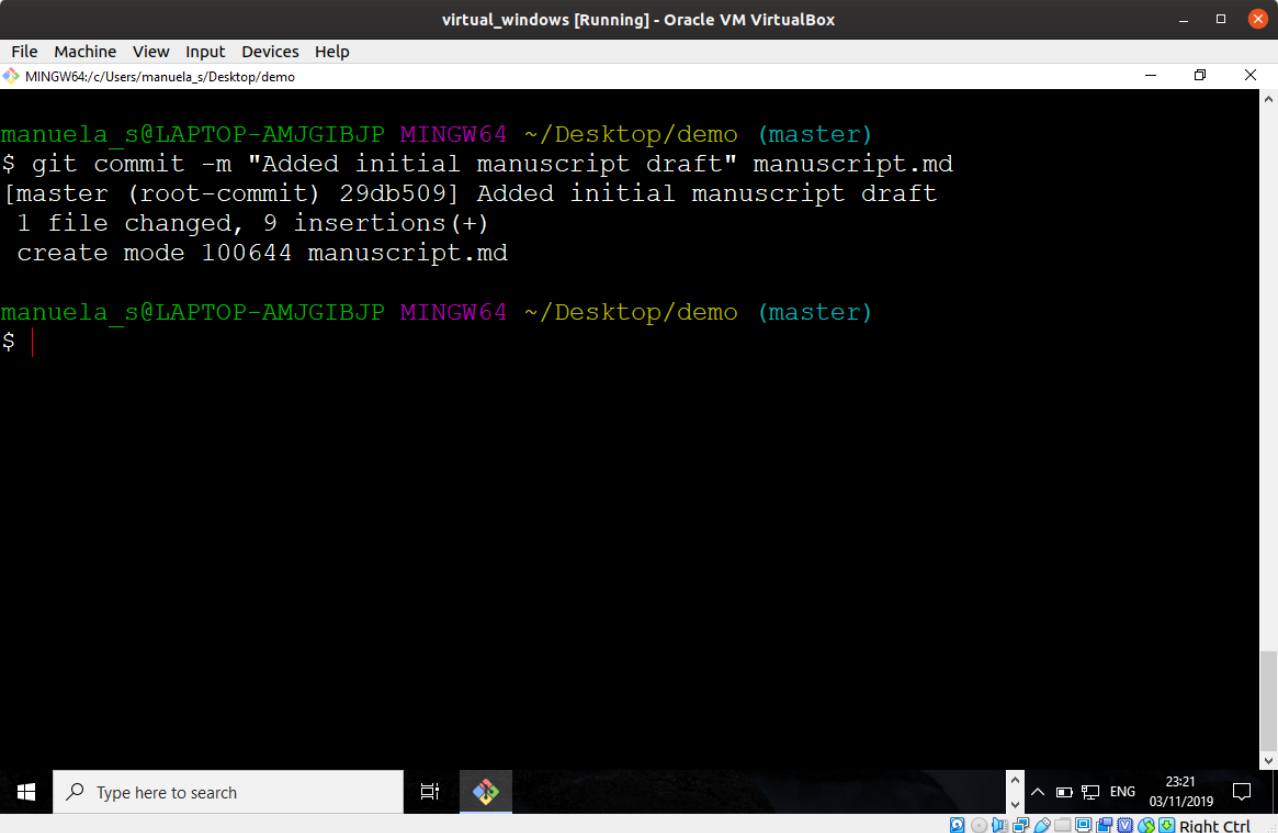
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

       new file:   manuscript.md
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

To prepare a new file to be added to the repository, we use git add. If we re-run git status we now see that the file is staged

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git commit -m "Added initial manuscript draft" manuscript.md
[master (root-commit) 29db509] Added initial manuscript draft
1 file changed, 9 insertions(+)
 create mode 100644 manuscript.md

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ |
```

To store changes in the repository, we use git commit. We specify a commit message after -m to record what we did

ANATOMY OF A COMMIT

- Includes:
 - what changed compared to the previous commit (**snapshot**)
 - which files are affected by changes and how
 - rationale for the change (**commit message**)
 - timestamp
 - “**name**”: unique identifier represented by [SHA-1 hashes](#)
 - for example: 4fc82ba7bb3f3a3de8ac57f16b6a926a7e60a21e
 - first 6 digits are *typically* sufficient to describe a commit -> shorthand version 4fc82ba
 - “**parent**” commit (reference to previous snapshot)
 - first commit is special (has no parent)
 - last commit is special (it is called HEAD)
- The full series of commits makes up the whole project

GUIDELINES ON COMMITS... SIZE MATTER

- Commit small units of changes and commit often
- A good unit of change is a small, self-contained, working change
 - **GOOD:** data.csv, process_data_figure1.py, make_figure1.py
 - **BAD:** 1 commit with a day worth of work (on multiple fronts)
- **Rule of thumb:** commit together what you would need to undo if you later want to disregard this change

GUIDELINES ON COMMITS... MESSAGE

- Write good commit messages:
 - **GOOD:** Update ReadMe to include 'how-to-install' section. Fixes issue ##1
 - **BAD:** Major fixup
 - which of the 2 messages above would you rather read the evening before a deadline?
- A perfect commit message summarises the what and why of the change, not the how (can be seen from the diffs)
- Other advice include:
 - keep the message subject concise (<50 words) -> log looks cleaner
 - add additional details (if needed) after a blank line and wrap at 72 characters -> readability
 - use imperative verb (*Add* vs. *Added*) -> if change get reverted, message reads better (*Revert Add ...*)
 - use commit.template
- [Examples of how \(not to\) write commit messages](#)
- [More tips on writing good commit messages](#)



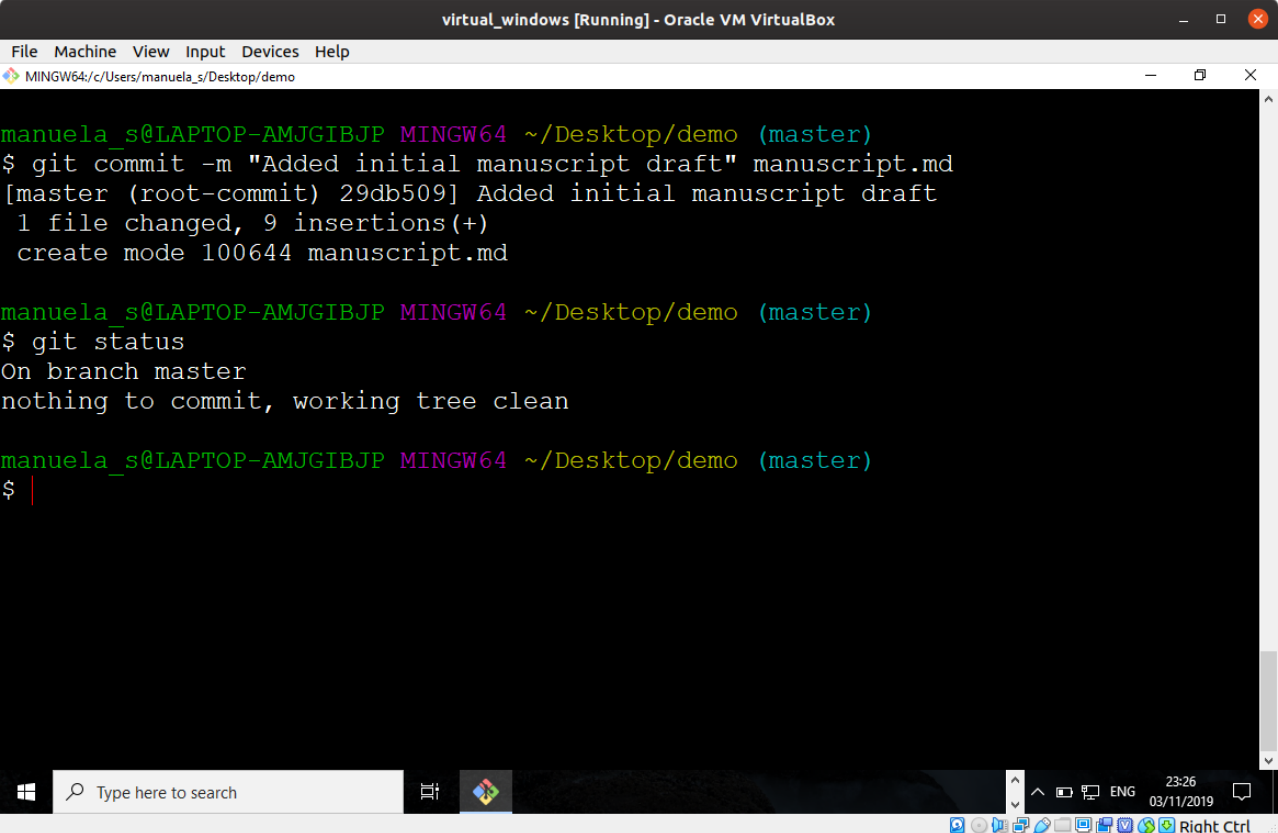
	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

From <https://xkcd.com/1296/>

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git commit -m "Added initial manuscript draft" manuscript.md
[master (root-commit) 29db509] Added initial manuscript draft
1 file changed, 9 insertions(+)
 create mode 100644 manuscript.md

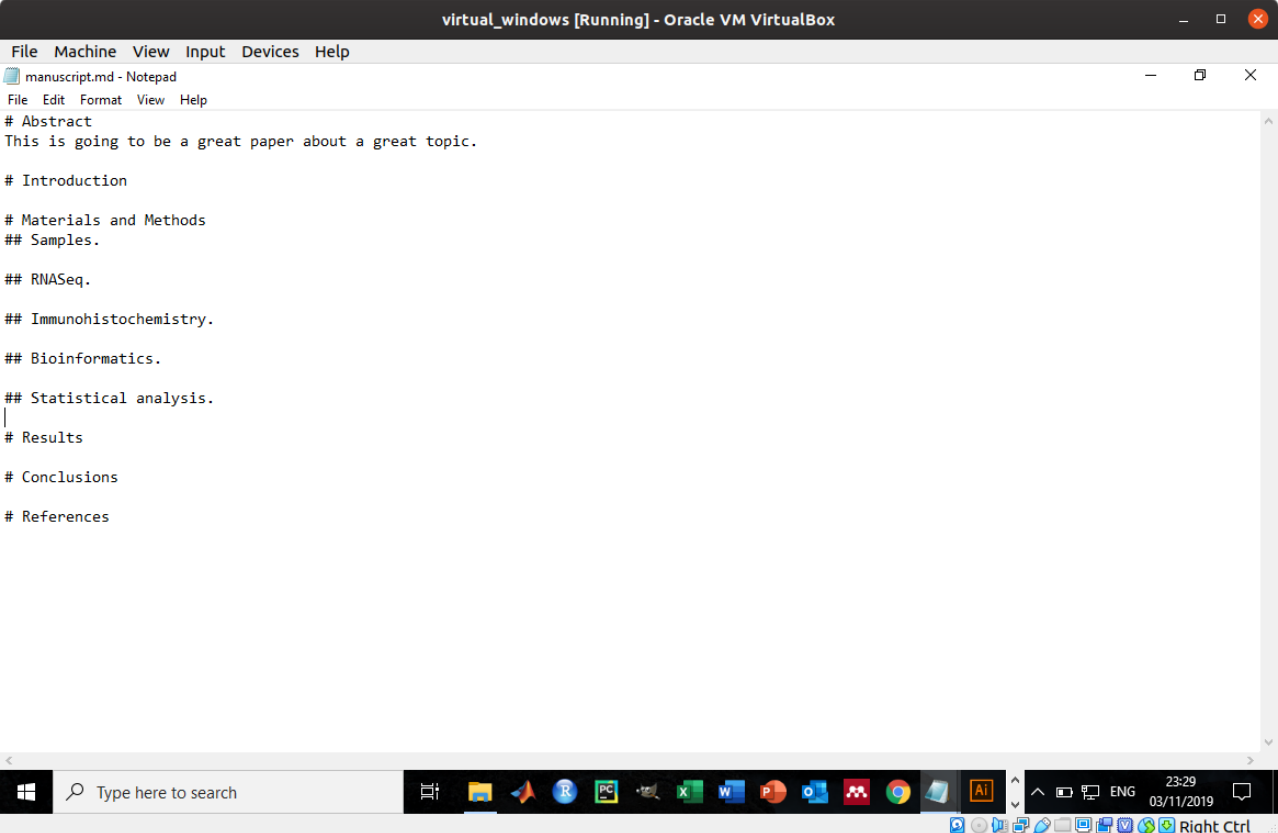
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git status
On branch master
nothing to commit, working tree clean

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

We use GIT status to check that there are no outstanding changes

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuscript.md - Notepad
File Edit Format View Help
# Abstract
This is going to be a great paper about a great topic.

# Introduction

# Materials and Methods
## Samples.

## RNASeq.

## Immunohistochemistry.

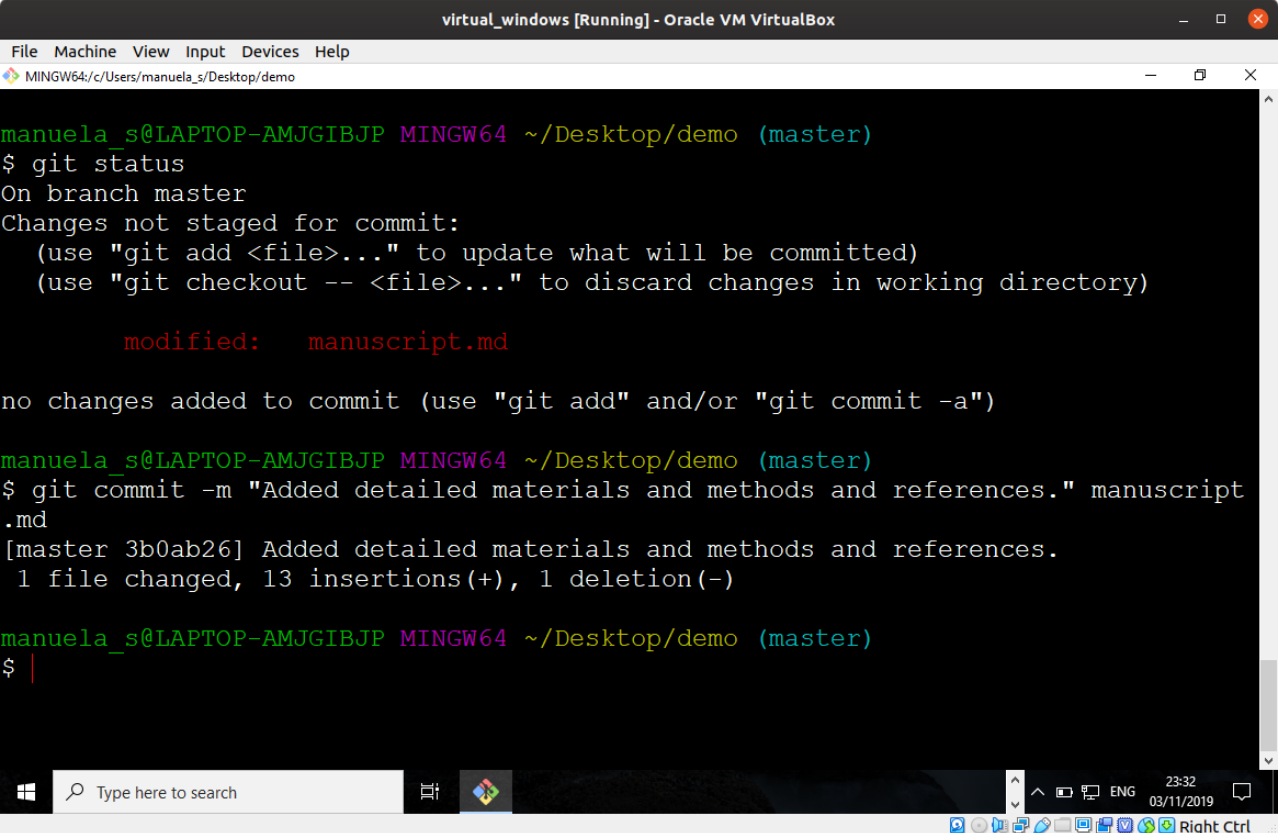
## Bioinformatics.

## Statistical analysis.
|
# Results
# Conclusions
# References
```

Let us do some more work on the manuscript. We need to add more details for materials and methods and add a section for references

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (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:   manuscript.md

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

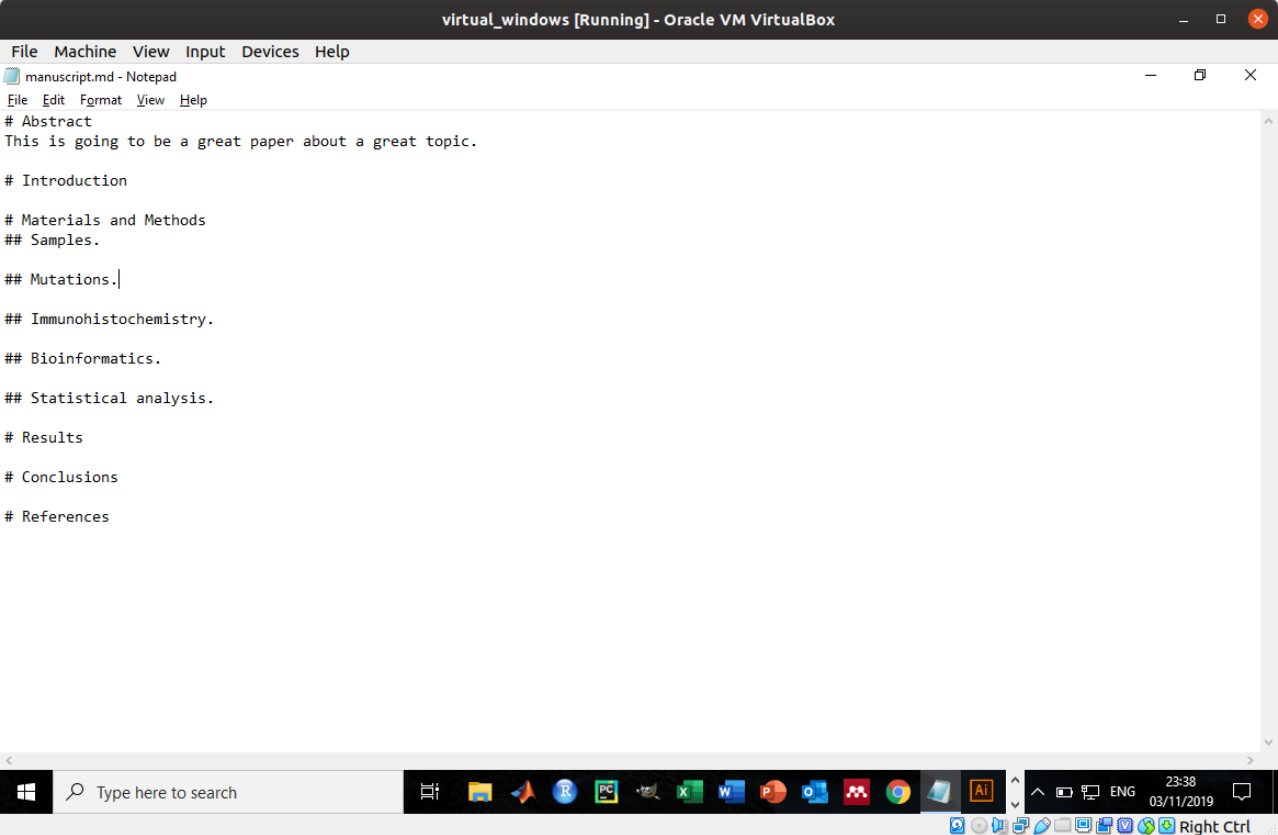
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git commit -m "Added detailed materials and methods and references." manuscript.md
[master 3b0ab26] Added detailed materials and methods and references.
 1 file changed, 13 insertions(+), 1 deletion(-)

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ |
```

Once we have finished with our change, we use git commit to add the new version of the file to the GIT repository

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



The screenshot shows a virtual machine window titled "virtual_windows [Running] - Oracle VM VirtualBox". Inside, a Notepad window titled "manuscript.md - Notepad" is open. The text in the Notepad window is as follows:

```
File Machine View Input Devices Help
manuscript.md - Notepad
File Edit Format View Help
# Abstract
This is going to be a great paper about a great topic.

# Introduction

# Materials and Methods
## Samples.

## Mutations.|

## Immunohistochemistry.

## Bioinformatics.

## Statistical analysis.

# Results

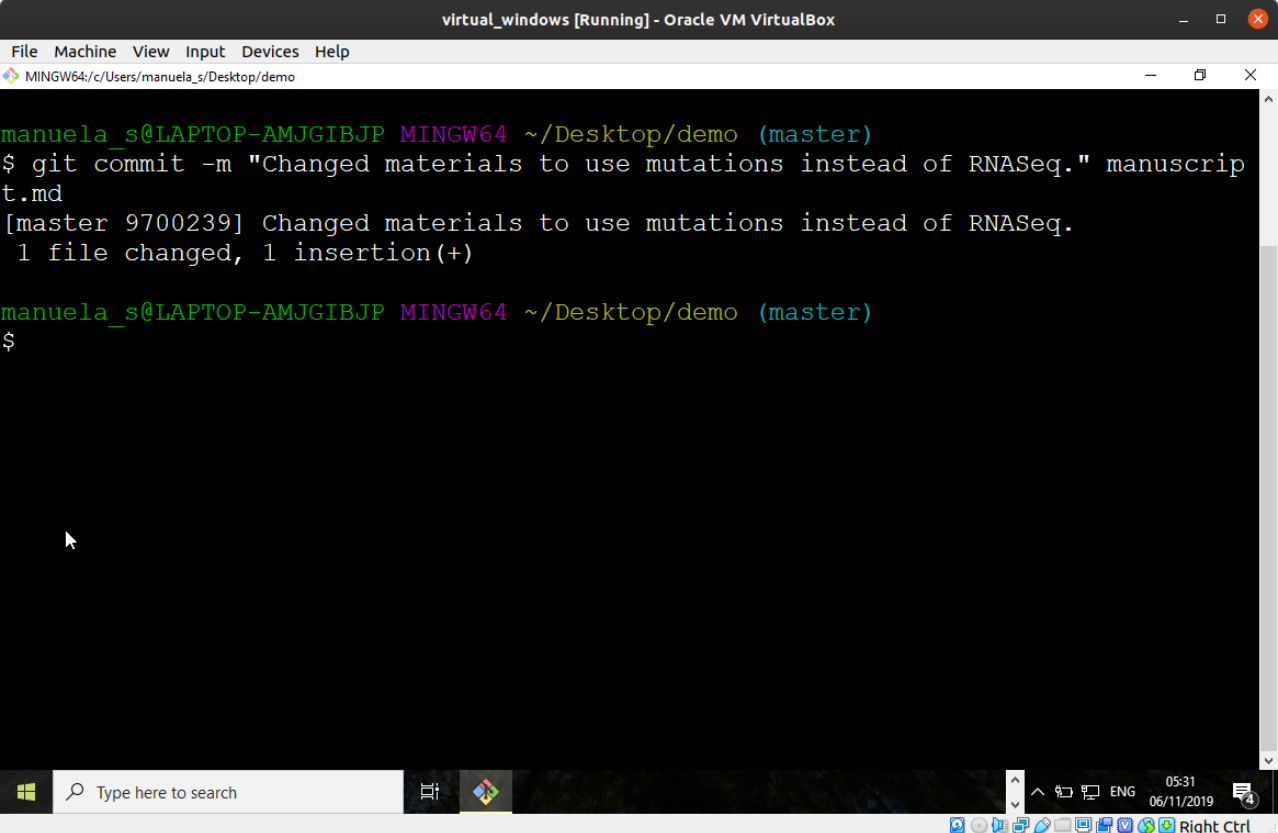
# Conclusions

# References
```

We changed our mind, and we will use mutation data instead of RNASeq. Let us update the materials and methods

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

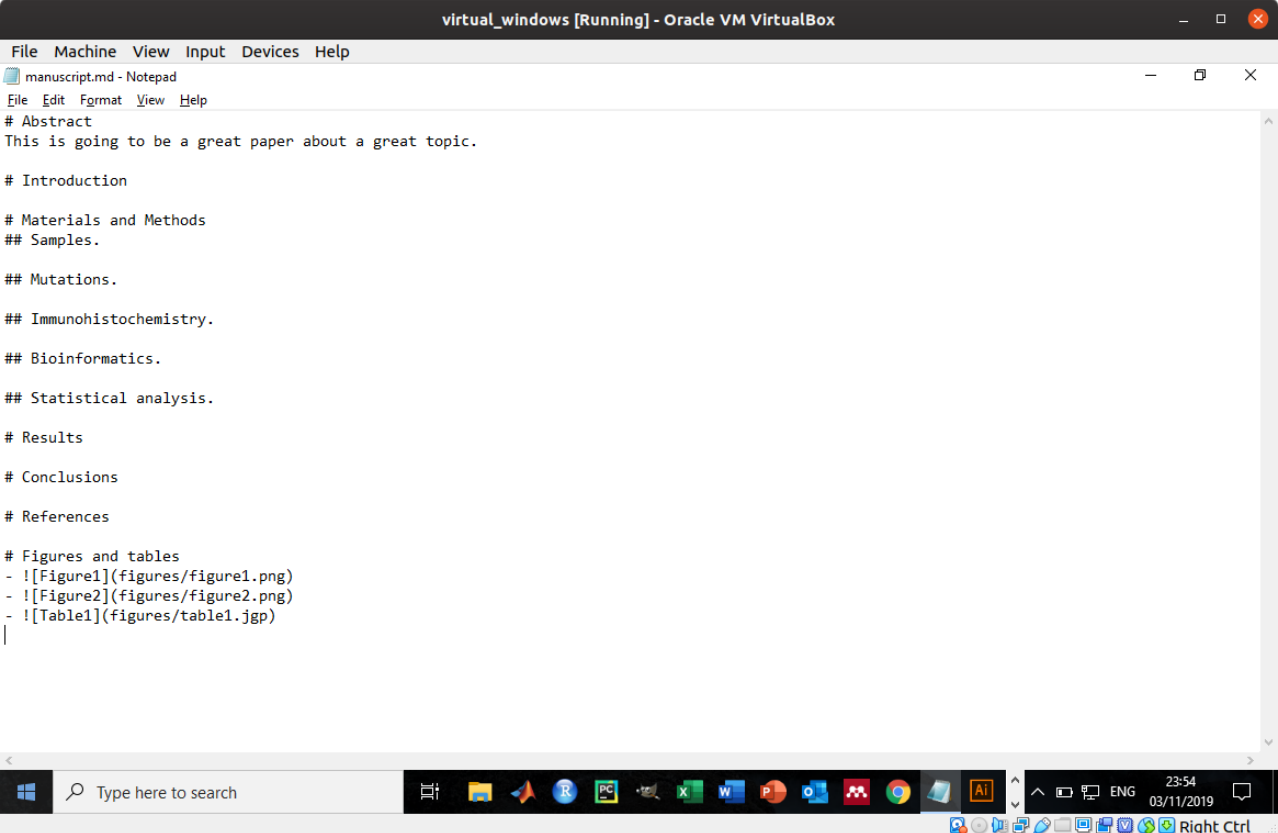


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git commit -m "Changed materials to use mutations instead of RNASeq." manuscript.md
[master 9700239] Changed materials to use mutations instead of RNASeq.
1 file changed, 1 insertion(+)
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

Commit the change as before

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuscript.md - Notepad
File Edit Format View Help
# Abstract
This is going to be a great paper about a great topic.

# Introduction

# Materials and Methods
## Samples.

## Mutations.

## Immunohistochemistry.

## Bioinformatics.

## Statistical analysis.

# Results

# Conclusions

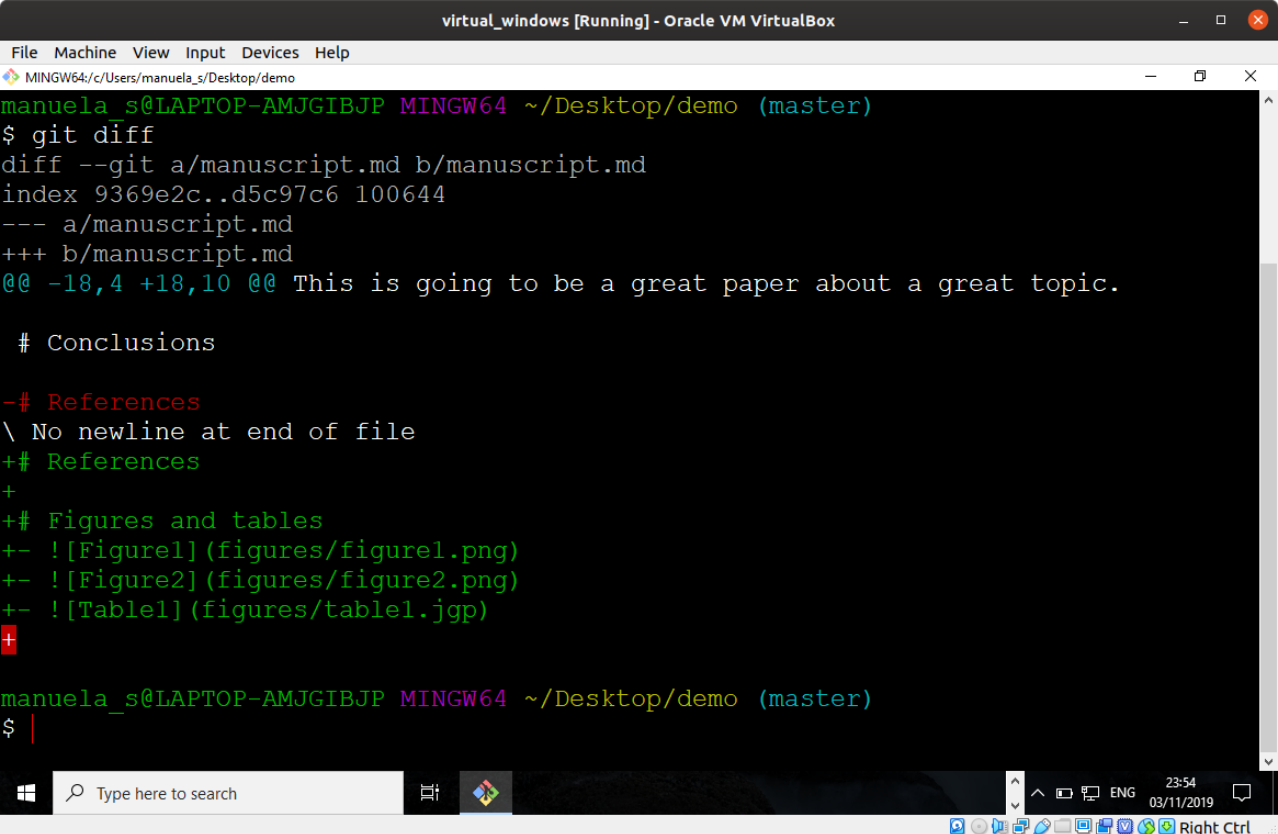
# References

# Figures and tables
- ![Figure1](figures/figure1.png)
- ![Figure2](figures/figure2.png)
- ![Table1](figures/table1.jpg)
|
```

Add figures and tables to our manuscript

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git diff
diff --git a/manuscript.md b/manuscript.md
index 9369e2c..d5c97c6 100644
--- a/manuscript.md
+++ b/manuscript.md
@@ -18,4 +18,10 @@ This is going to be a great paper about a great topic.

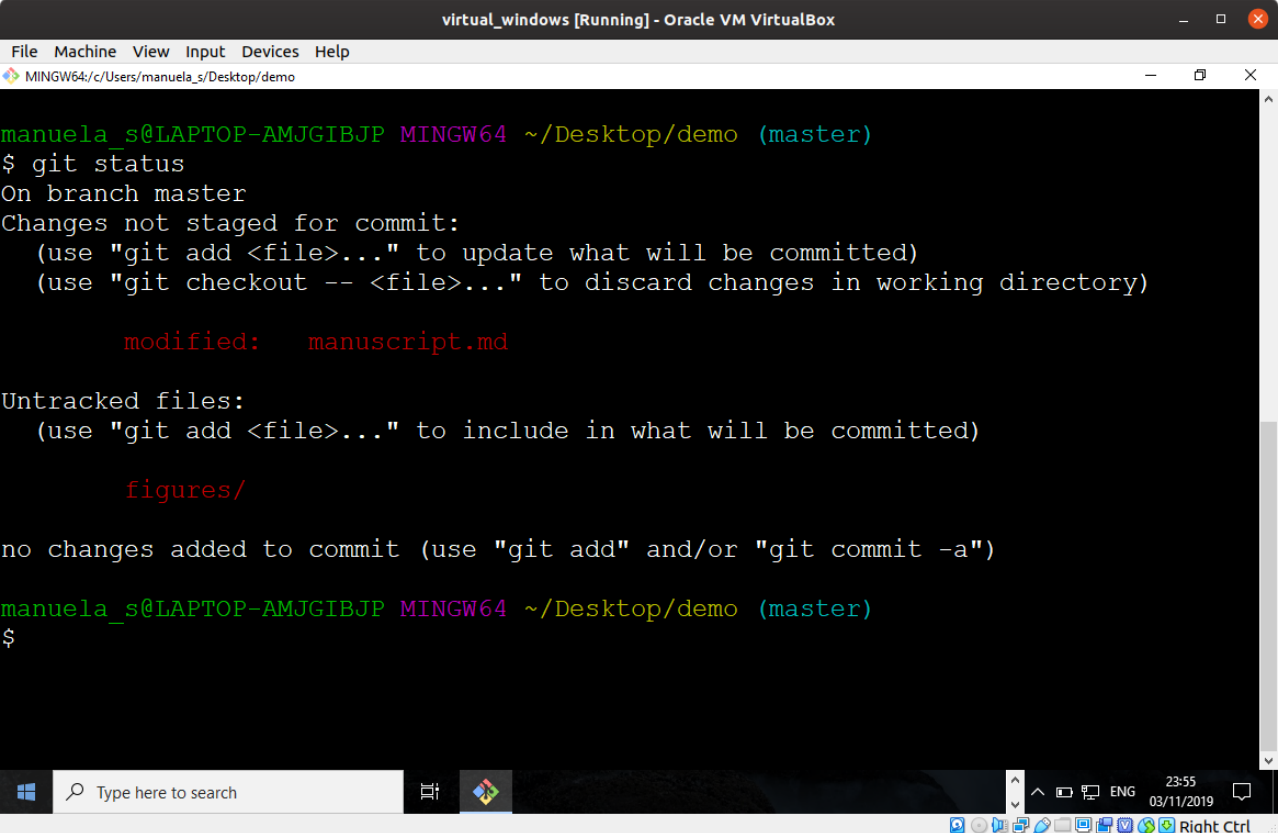
 # Conclusions

-# References
\ No newline at end of file
+# References
+
+# Figures and tables
+- ![Figure1] (figures/figure1.png)
+- ![Figure2] (figures/figure2.png)
+- ![Table1] (figures/table1.jpg)
+
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

We can use the git diff command to see how our current files are different from the last one checked into the repository

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (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:   manuscript.md

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

        figures/

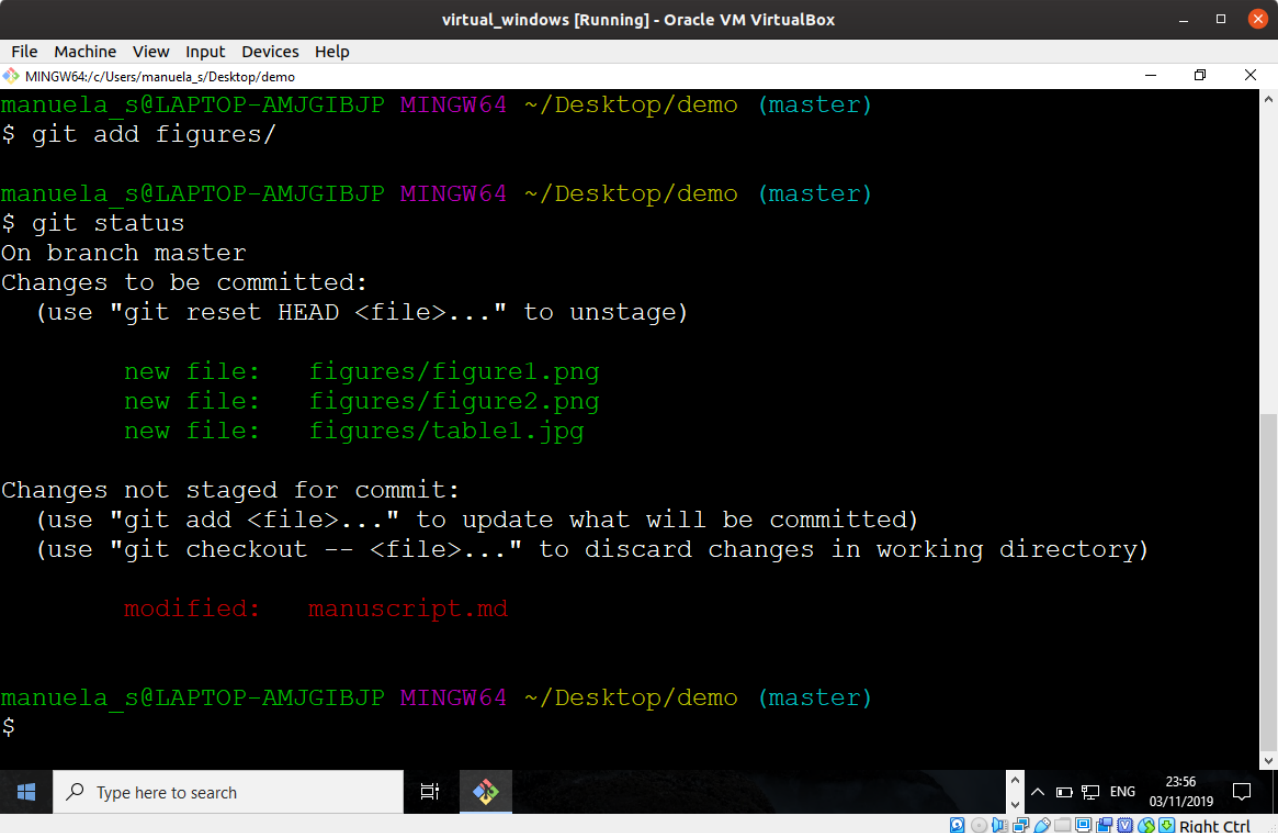
no changes added to commit (use "git add" and/or "git commit -a")

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

If we create a figures directory with some image files and run git diff we see that this directory is untracked by GIT

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git add figures/

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

       new file:   figures/figure1.png
       new file:   figures/figure2.png
       new file:   figures/table1.jpg

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)

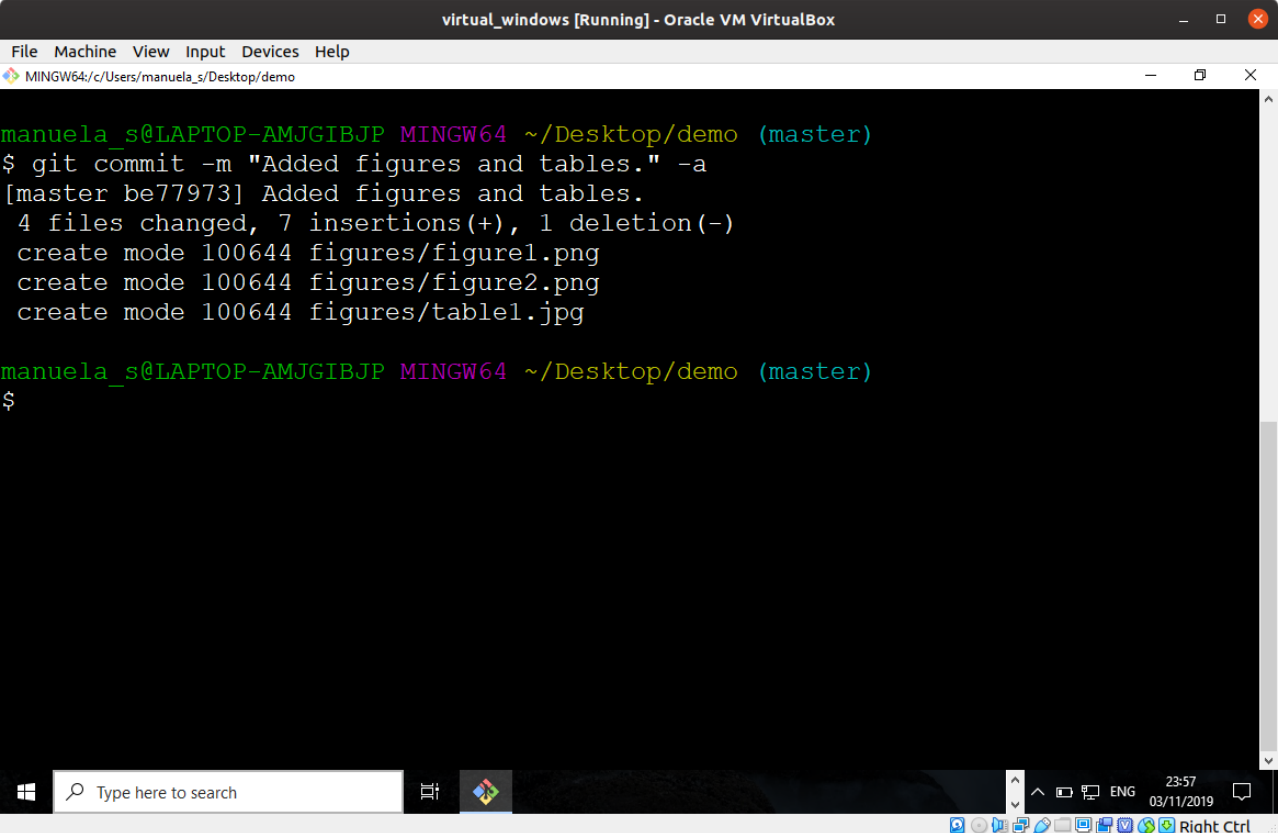
       modified:   manuscript.md

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

We add the whole directory with git add and rerun git status.
Now it lists the files as new instead

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

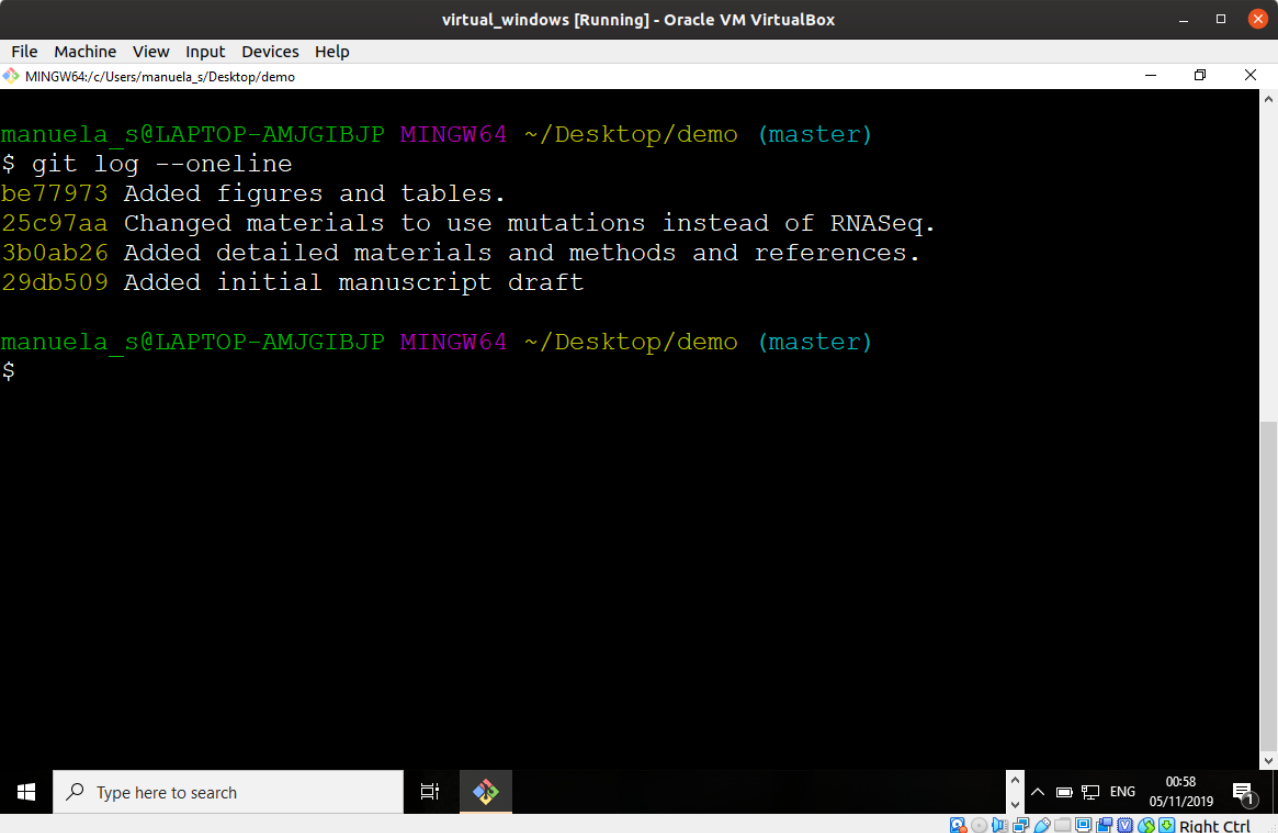
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git commit -m "Added figures and tables." -a
[master be77973] Added figures and tables.
4 files changed, 7 insertions(+), 1 deletion(-)
create mode 100644 figures/figure1.png
create mode 100644 figures/figure2.png
create mode 100644 figures/table1.jpg

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

To commit the files, we use `git commit -a` instead of listing them. `-a` means all, and will commit all files that we have added or modified

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

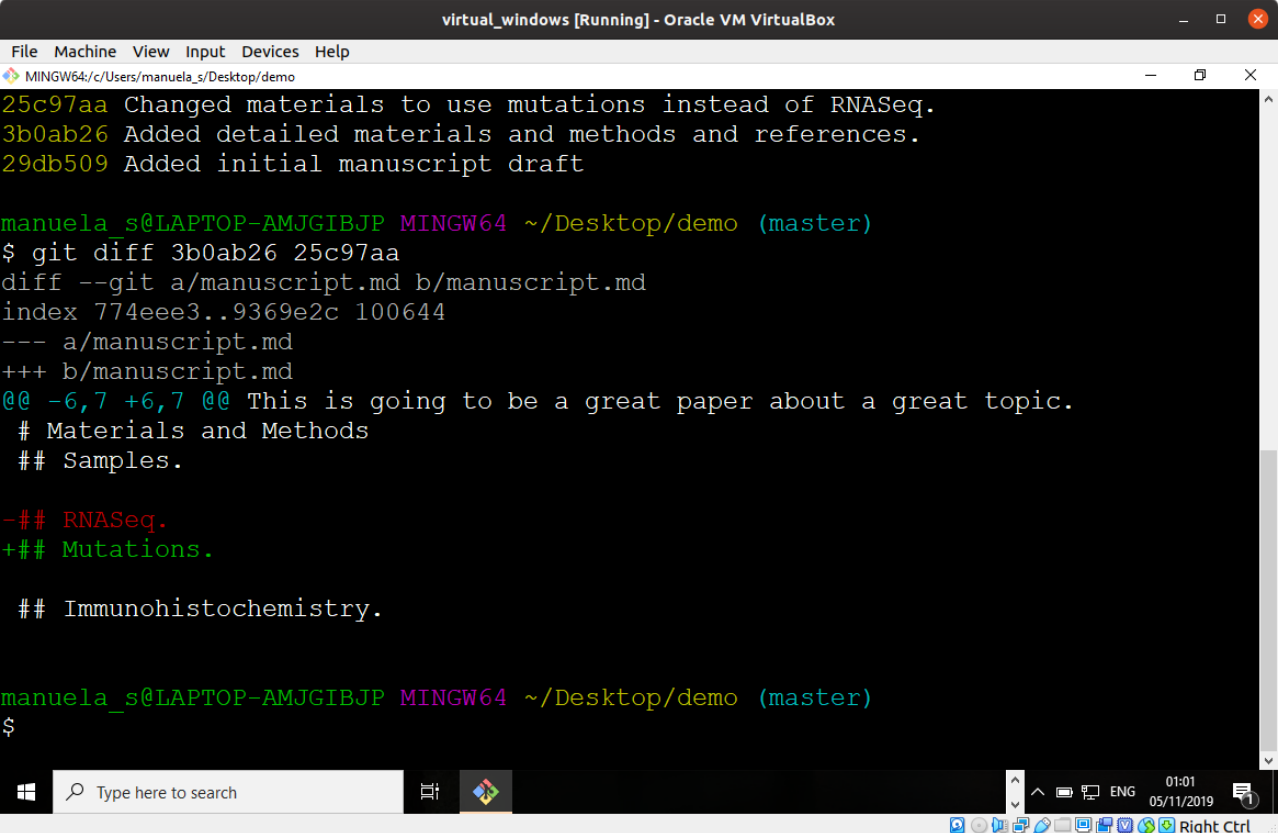


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git log --oneline
be77973 Added figures and tables.
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

The git log command can show a history from the repository. Last change on top. --oneline gives a more compact representation

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

A screenshot of a terminal window titled 'virtual_windows [Running] - Oracle VM VirtualBox'. The terminal shows the output of a 'git diff' command between two commits. The first commit (25c97aa) changed materials to use mutations instead of RNASeq. The second commit (3b0ab26) added detailed materials and methods and references. The third commit (29db509) added an initial manuscript draft. The diff output shows the changes in the 'manuscript.md' file, highlighting the addition of 'Mutations' and the removal of 'RNASeq'.

```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git diff 3b0ab26 25c97aa
diff --git a/manuscript.md b/manuscript.md
index 774eee3..9369e2c 100644
--- a/manuscript.md
+++ b/manuscript.md
@@ -6,7 +6,7 @@ This is going to be a great paper about a great topic.
 # Materials and Methods
 ## Samples.

-## RNASeq.
+## Mutations.

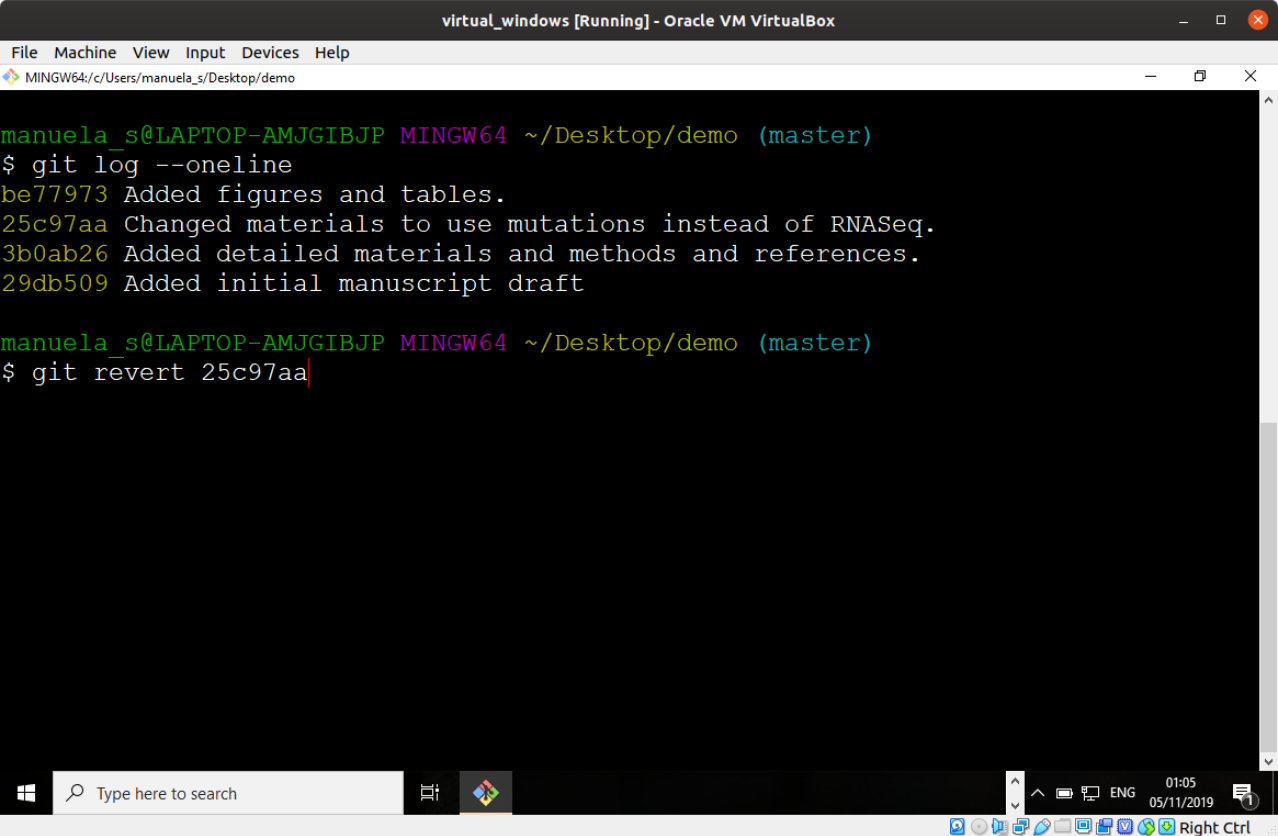
 ## Immunohistochemistry.

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

Git diff can also be used to show the difference between two revisions in the history. We need to specify the two commit identifiers

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo

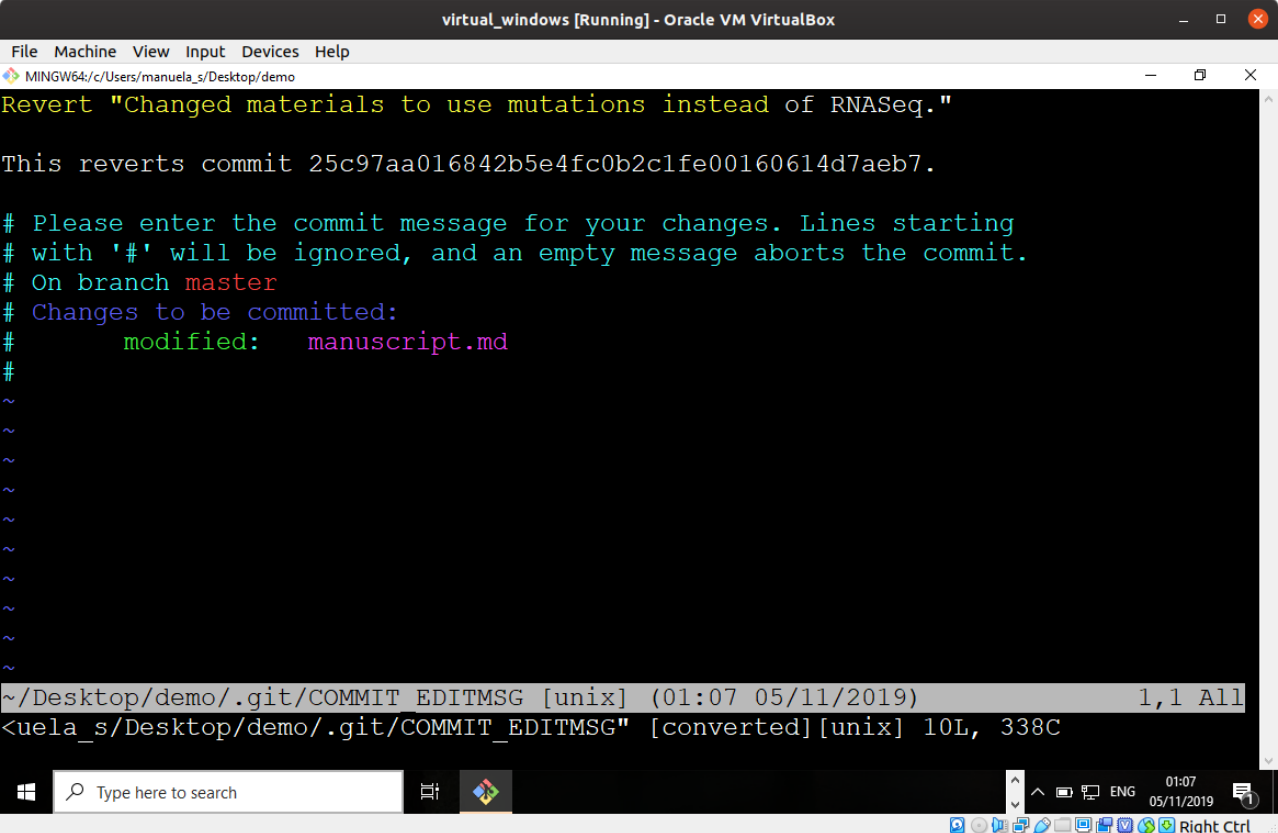
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git log --oneline
be77973 Added figures and tables.
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git revert 25c97aa
```

Actually, we changed our mind again, and want to use RNASeq.
Let us revert the previous change

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert

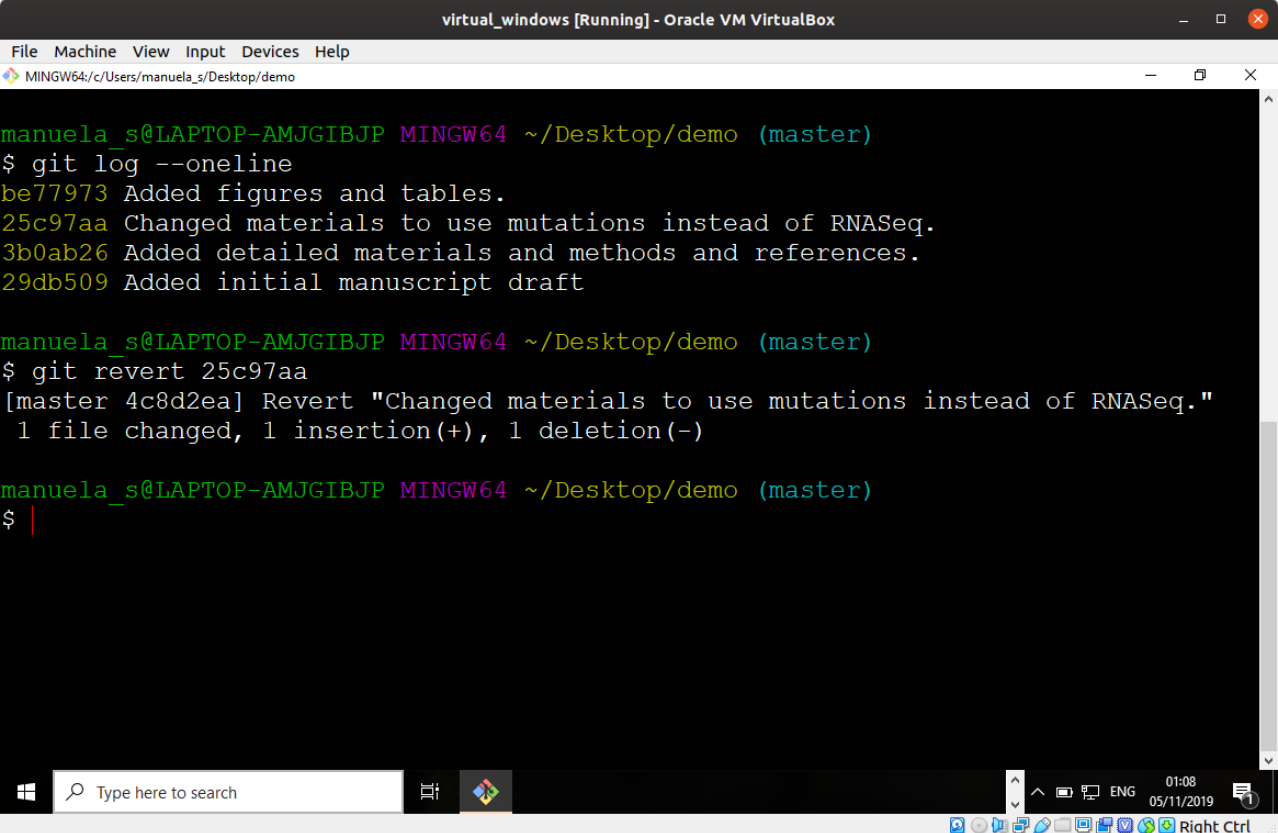


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
Revert "Changed materials to use mutations instead of RNASeq."
This reverts commit 25c97aa016842b5e4fc0b2c1fe00160614d7aeb7.
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   modified:   manuscript.md
#
~/Desktop/demo/.git/COMMIT_EDITMSG [unix] (01:07 05/11/2019) 1,1 All
<uela_s/Desktop/demo/.git/COMMIT_EDITMSG" [converted][unix] 10L, 338C
```

GIT will ask us for a commit message for the revert. The default message is fine

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git log --oneline
be77973 Added figures and tables.
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft

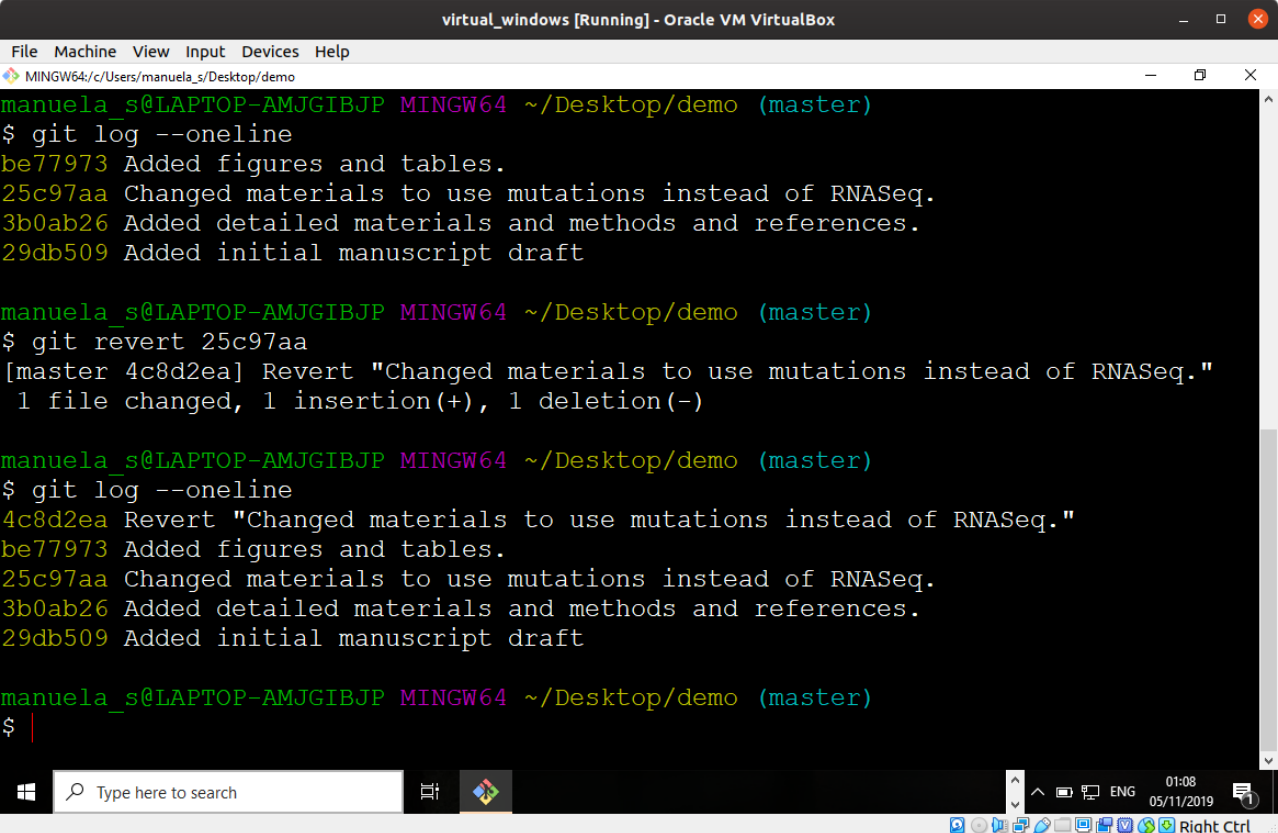
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git revert 25c97aa
[master 4c8d2ea] Revert "Changed materials to use mutations instead of RNASeq."
1 file changed, 1 insertion(+), 1 deletion(-)

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ |
```

GIT confirms the change, like a normal commit

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
5. Standard workflow
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git log --oneline
be77973 Added figures and tables.
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git revert 25c97aa
[master 4c8d2ea] Revert "Changed materials to use mutations instead of RNASeq."
1 file changed, 1 insertion(+), 1 deletion(-)

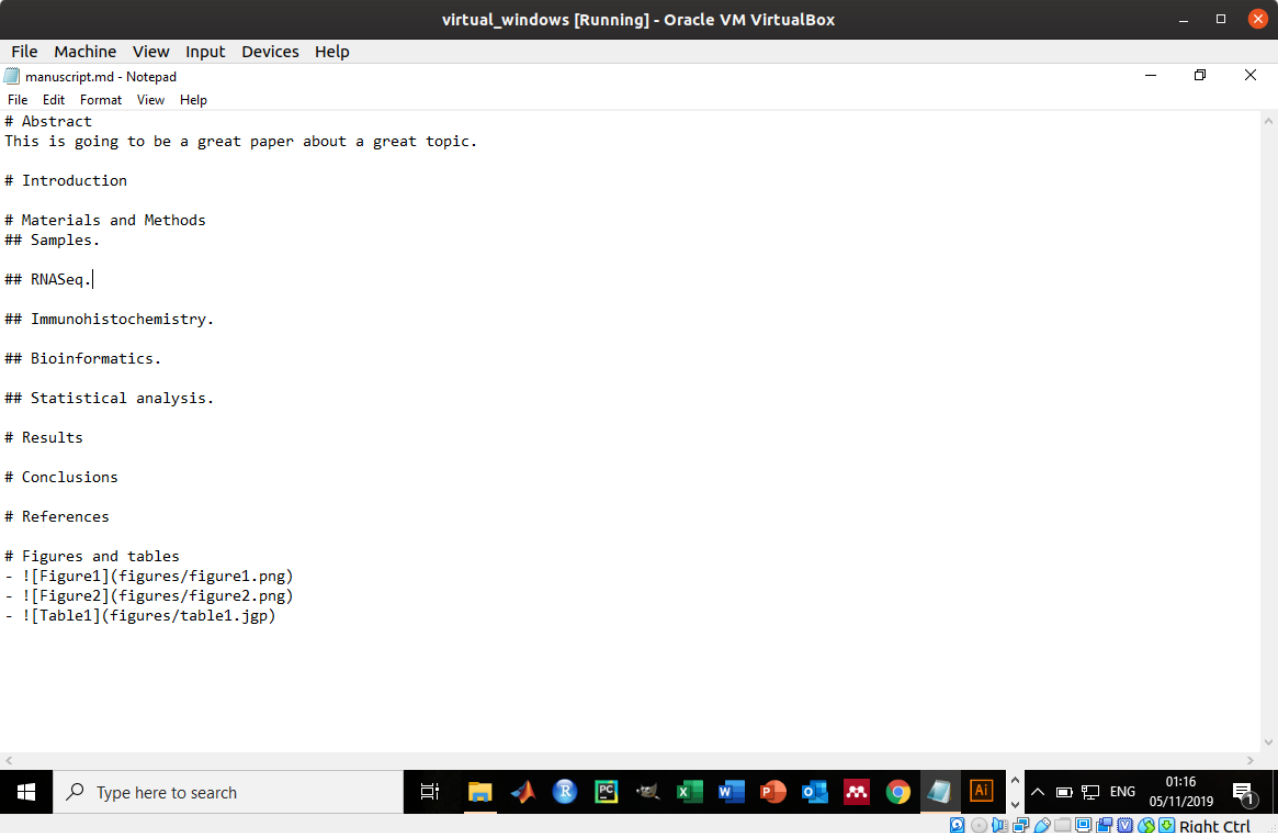
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git log --oneline
4c8d2ea Revert "Changed materials to use mutations instead of RNASeq."
be77973 Added figures and tables.
25c97aa Changed materials to use mutations instead of RNASeq.
3b0ab26 Added detailed materials and methods and references.
29db509 Added initial manuscript draft

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

The history captures the revert

PRACTICAL EXAMPLE

1. Make a project folder
2. Start a GIT bash in the project folder
3. Configure GIT
4. Initialize repository
 1. Make edits
 2. (git add)
 3. git commit
 4. Repeat
6. git status
7. git diff
8. git log
9. git revert



The screenshot shows a virtual machine window titled 'virtual_windows [Running] - Oracle VM VirtualBox'. Inside the VM, a Notepad window titled 'manuscript.md - Notepad' is open, displaying the following text:

```
File Machine View Input Devices Help
manuscript.md - Notepad
File Edit Format View Help
# Abstract
This is going to be a great paper about a great topic.

# Introduction

# Materials and Methods
## Samples.

## RNASeq.

## Immunohistochemistry.

## Bioinformatics.

## Statistical analysis.

# Results

# Conclusions

# References

# Figures and tables
- ![Figure1](figures/figure1.png)
- ![Figure2](figures/figure2.png)
- ![Table1](figures/table1.jpg)
```

The Windows taskbar at the bottom shows the search bar, taskbar icons for various applications, and system tray information including the date (05/11/2019) and time (01:16).

Note that we did not just go back to a previous revision. We selectively undid the RNASeq->mutation change, but we still have the figures and tables, which was added afterwards. GIT has automatically merged our changes together

OTHER USEFUL GIT COMMANDS

- `git rm FILENAME`: delete tracked file
- `git mv FILENAME1 FILENAME2`: rename file from FILENAME1 to FILENAME2
- `git log -follow`: inspect log (even with renaming)

I AM WORKING ON IT...

- **git add -p FILENAME:** add portions of changes you made to a file
 - preserve other changes, but they will not be captured in this commit
 - useful when you set out to make some changes, but you could not help fixing (other) unrelated stuff
- **git squash:** pool related commits in a meta-commit
- **git stash:** stash away *work in progress* which is in a state that is too preliminary to be committed and get back to it later
 - **git stash list**
 - **git stash pop**
 - **git stash drop**

OPS, I DID NOT MEAN TO DO THAT.... LET'S PRETEND IT NEVER HAPPENED

- **git commit -amend:** by far the most used command
 - useful when you forgot to add a file before committing or you would like to change commit message
- **git revert SHA:** revert changes applied by SHA by creating a new commit
- **git checkout FILENAME:** undo (uncommitted) changes to FILENAME
- **git checkout SHA:** checkout a snapshot where all was good
- **git reset:** undo changes, degree of annihilation depends on flags (*-soft* vs. *-hard*), be careful

OPS, SOMETHING WENT TERRIBLY TERRIBLY WRONG, AT SOME POINT IN THE PAST










- **git show** : inspect (suspicious) commit
- **git blame**: when and who changed/broke this?
- **git bisect**: run binary search to identify when problem was introduced
 - extremely useful command, a life-saver
 - requires *knowing* what *right* vs. *wrong* means (unit tests, ground truth, ...)

GIT SUBMODULES... RUSSIAN DOLLS REPOSITORIES

“It often happens that while working on one project, you need to use another project from within it. Perhaps it’s a library that a third party developed or that you’re developing separately and using in multiple parent projects. A common issue arises in these scenarios: you want to be able to treat the two projects as separate yet still be able to use one from within the other.”

<https://git-scm.com/book/en/v2/Git-Tools-Submodules>

Name

-  data
 -  deps
 -  outputs
 -  paper_draft
 -  +bioenergetics_patients [33a2bf510c47]
 -  +cohort_utilities [6005a6bedc06]
 -  +in_house_cohorts [37c42332b6fa]
 -  +matlab_utilities [22029def9a1]
 -  +tcga_pancancer [e5e8ce28db82]
 -  py_utils [aa7d5b3cf31d]
 -  transcriptomics [bf129dd11966]
-

EXERCISE 1

EXERCISE 1 (20 MIN)

1. Create a folder named “christmas_repo”
2. Open GIT bash, verify the installation and configure GIT
3. Initialize a GIT repository in the folder
4. Create a text file (“wish_list.md”) with 3 gifts you wish to receive for Christmas
5. Add and commit the wish list file to GIT
6. Edit the wish list file, and add 2 more presents
7. Use GIT to check the difference between the current and previous version
8. Commit the updated file
9. Create a file (“recipients.md”) with a list of people you plan to buy gifts for
10. Check the status of the GIT repository
11. Add and commit the new file to GIT
12. Create a file (“past_gifts.md”) with a list of what gifts you gave last year
13. Maybe you remembered a few more people you would like to give gifts to. Add them to “recipients.md”
14. Add the new file and commit “past_gifts.md” and “recipients.md” to GIT
15. Look at the GIT history
16. Revert the change that added more presents to the wish list in step 5
17. Play around with doing more changes and commits

GIT SUPPORT TOOLS

GIT SUPPORT TOOLS

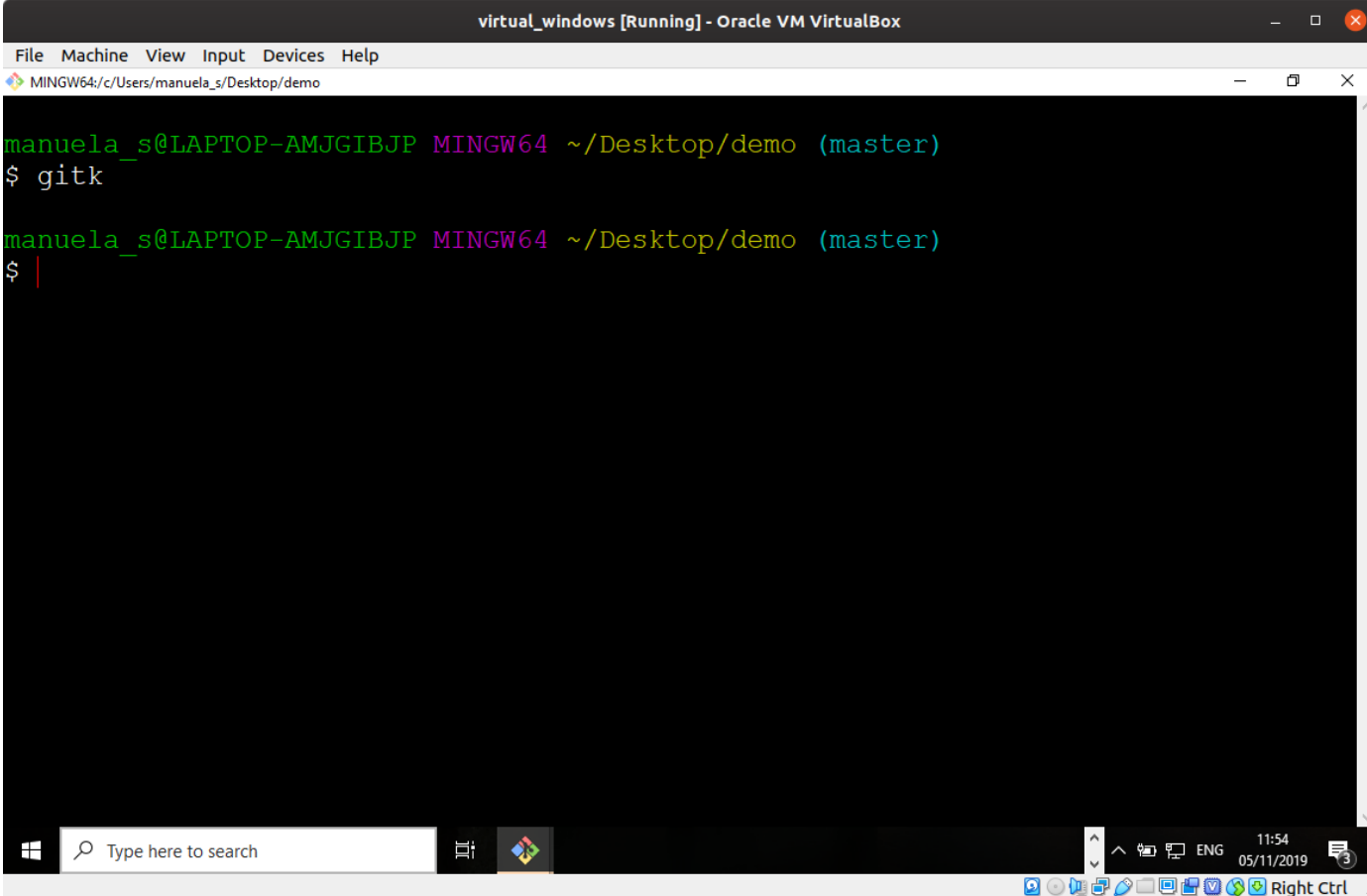
- Graphical user interface (GUIs)
- Integration with software Integrated Development Environments (IDEs) for R, MATLAB, Python, ...
- Cloud services (BitBucket, GitHub, GitLab)

GIT GRAPHICAL USER INTERFACE (GUI)

The screenshot shows a web browser window titled "virtual_windows [Running] - Oracle VM VirtualBox". The browser address bar shows "git-scm.com/downloads/guis/". The page content includes the Git logo and the text "--fast-version-control". A search bar is present with the text "Search entire site...". The main heading is "GUI Clients". Below the heading, there is a paragraph: "Git comes with built-in GUI tools for committing (`git-gui`) and browsing (`gitk`), but there are several third-party tools for users looking for platform-specific experience." Below this, there is a link: "If you want to add another GUI tool to this list, just [follow the instructions](#)." There are several filter buttons: "All", "Windows", "Mac", "Linux", "Android", and "iOS". Below the filters, there are two preview images. The left one is for "SourceTree" and the right one is for "GitHub Desktop". Both preview images show their respective graphical interfaces. At the bottom of the browser window, there is a Windows taskbar with a search bar and several application icons. The system tray shows the date and time as "11:38 05/11/2019".

From <https://git-scm.com/downloads/guis>

GIT GRAPHICAL USER INTERFACE (GUI)

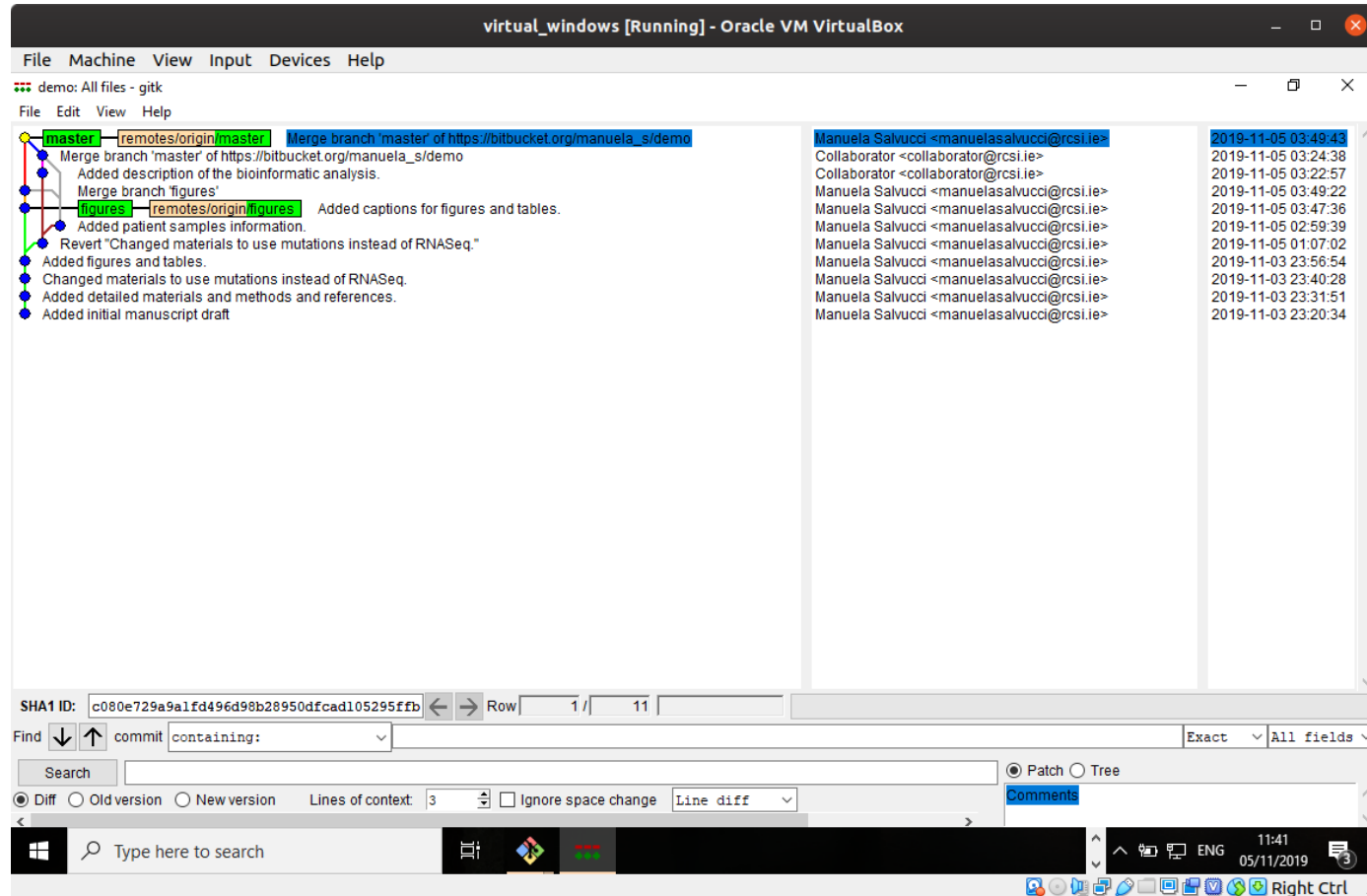


The image shows a screenshot of a Windows command prompt window running inside a virtual machine. The window title is "virtual_windows [Running] - Oracle VM VirtualBox". The menu bar includes "File", "Machine", "View", "Input", "Devices", and "Help". The address bar shows the path "MINGW64:/c/Users/manuela_s/Desktop/demo". The command prompt shows the user "manuela_s@LAPTOP-AMJGIBJP" in the "MINGW64" environment, currently on the "master" branch in the directory "~/Desktop/demo". The user has entered the command "gitk" and the prompt is waiting for input.

```
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ gitk

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ |
```

GIT GRAPHICAL USER INTERFACE (GUI)

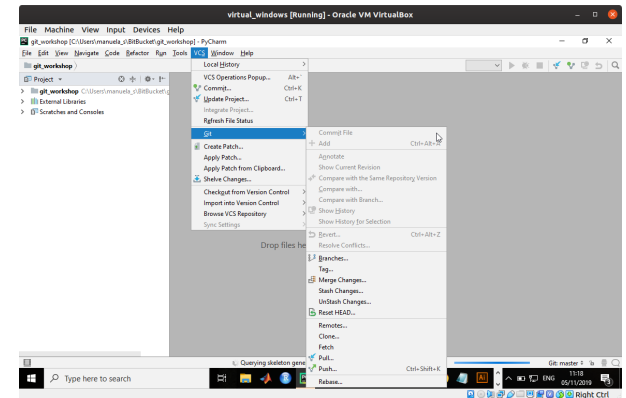
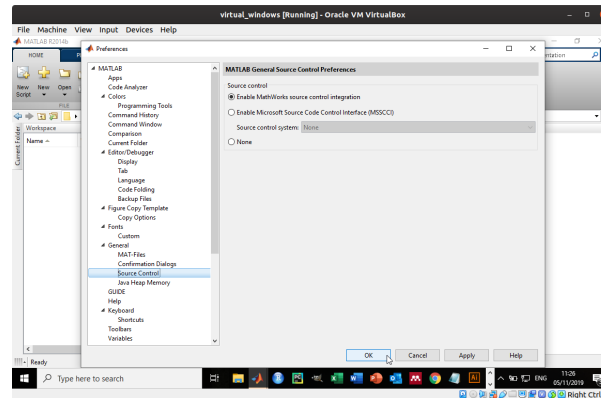
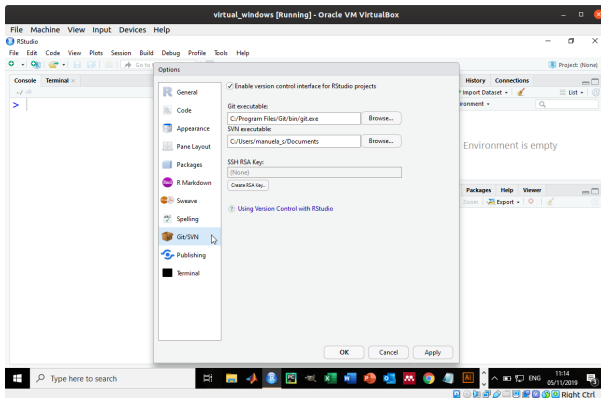


GIT INTEGRATION WITH SOFTWARE INTEGRATED DEVELOPMENT ENVIRONMENTS (IDE)

RStudio (R IDE)

MATLAB (MATLAB IDE)

PyCharm (Python IDE)



CLOUD SERVICES THAT SUPPORT GIT... “SOCIAL” CODING

- Servers that can host a copy of your repository
- Useful as a backup
- Can make synchronization and collaboration easier
- Free plans available
- Most popular alternatives:

GitHub

 **Bitbucket**

 **GitLab**

- Other alternatives include Crucible, AWS CodeCommit, CodeCommit,

CLOUD SERVICES THAT SUPPORT GIT... “SOCIAL” CODING

Comparison of key features in free plans from GitHub, BitBucket and GitLab

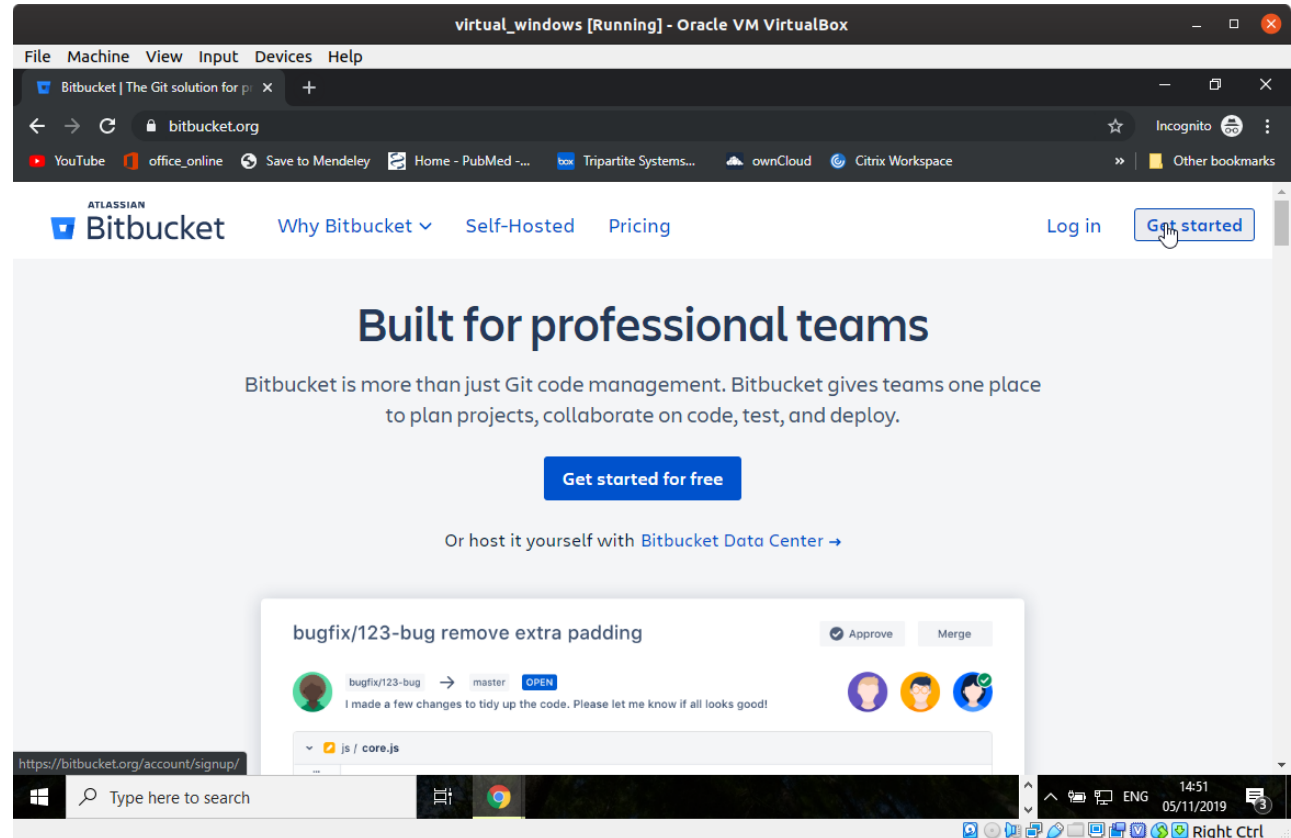
	GitHub	BitBucket	GitLab
Max # users	3	5	unlimited
# private repositories	unlimited	unlimited	unlimited
Size (per repository)	1-2 GB	1 GB	10 GB
Large file storage support	yes	yes	yes
Continuous integration support	yes	yes	yes

- Similar products, select the one that suits best your needs

BITBUCKET

BITBUCKET EXAMPLE

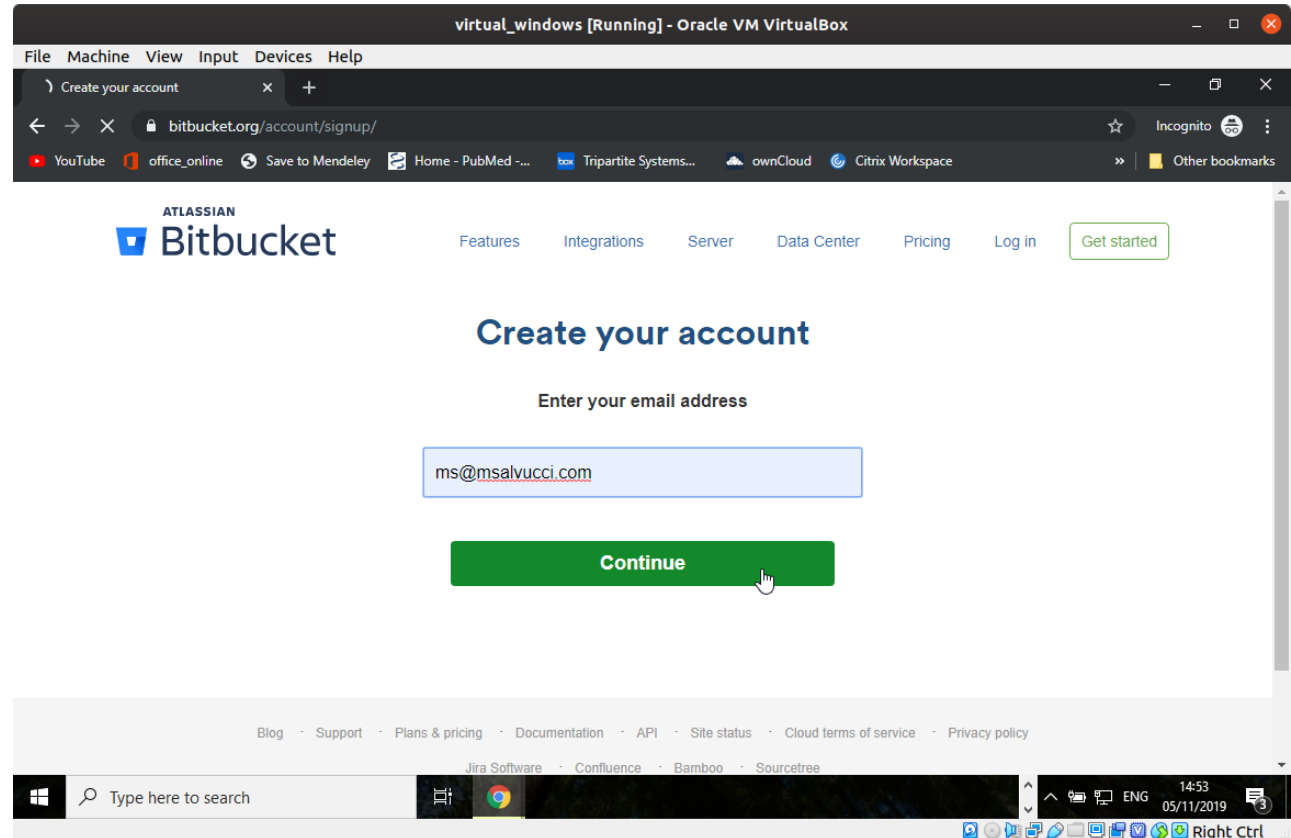
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Go to the BitBucket website and click Get Started

BITBUCKET EXAMPLE

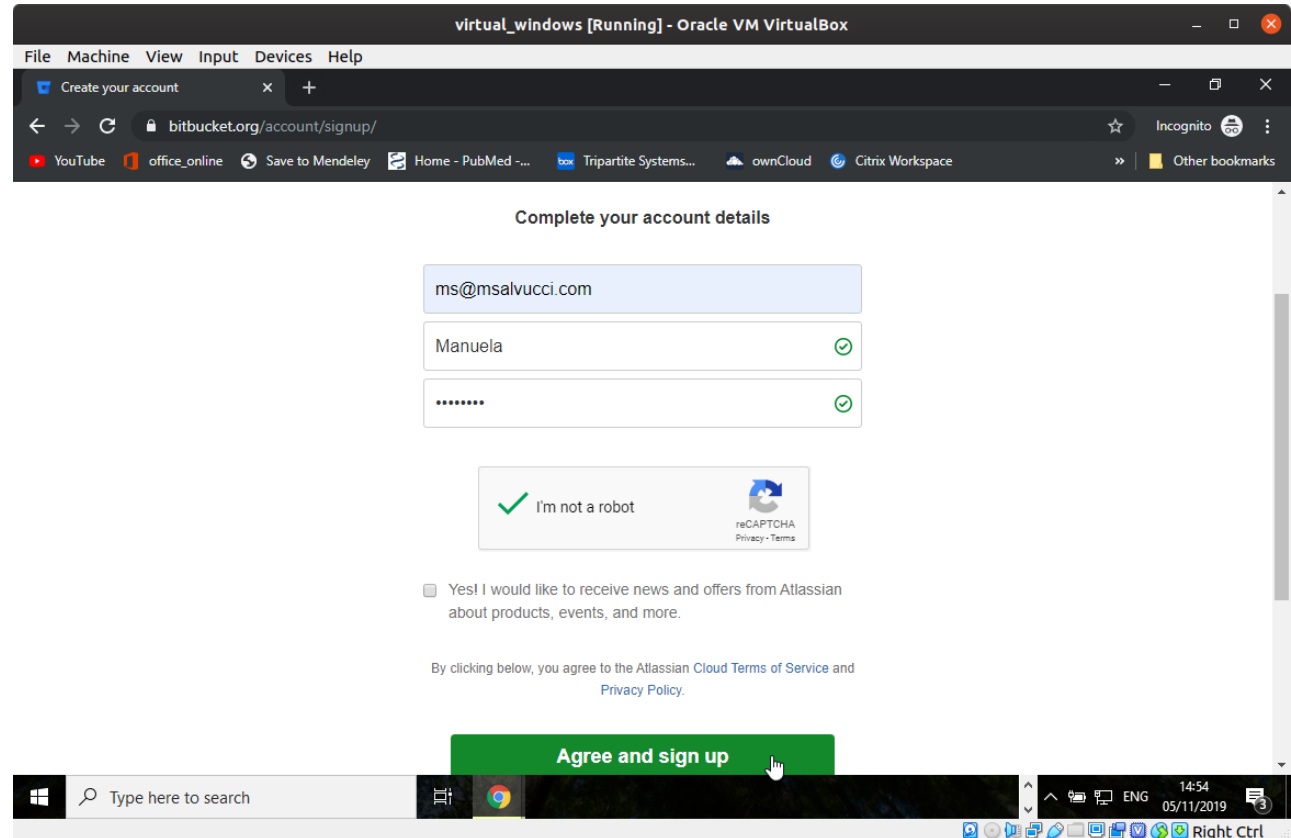
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Follow instruction by filling in required info

BITBUCKET EXAMPLE

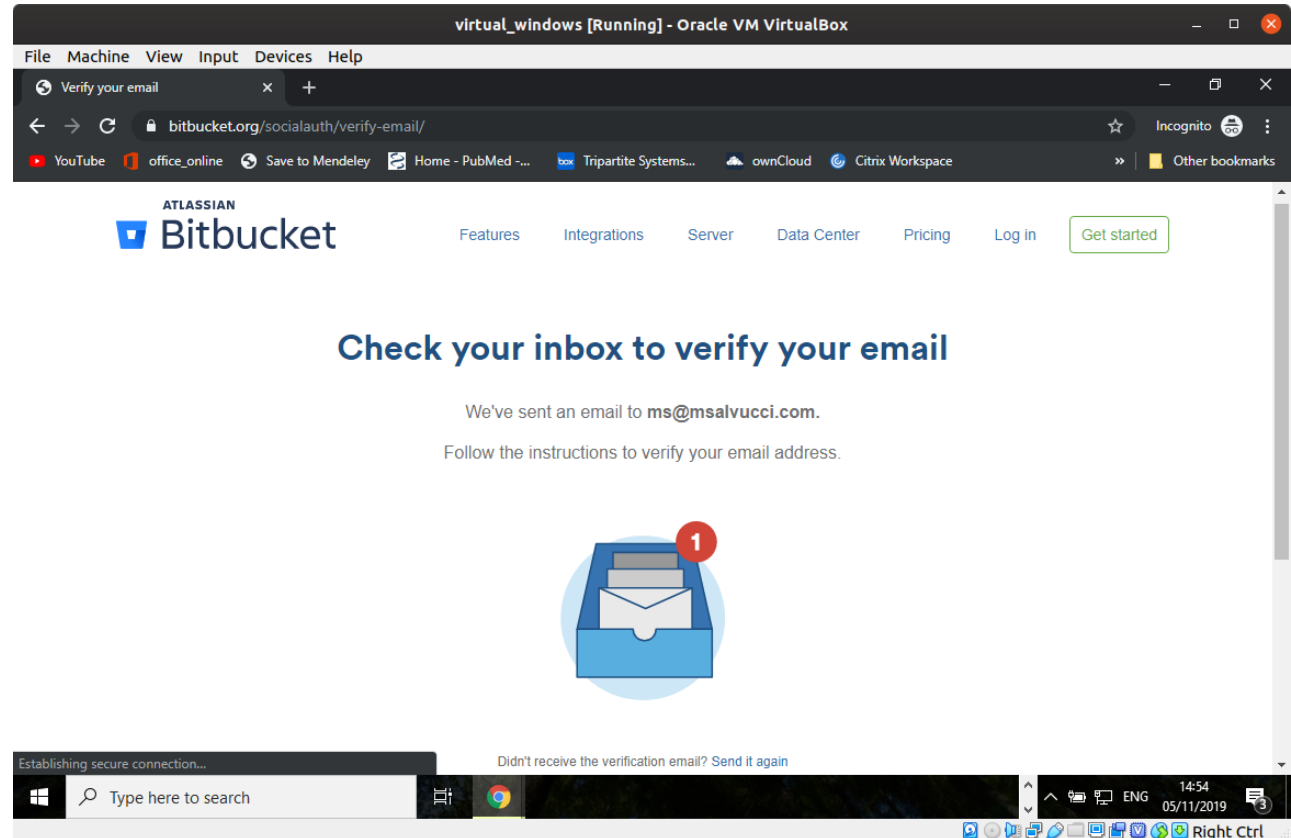
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Follow instruction by filling in required info

BITBUCKET EXAMPLE

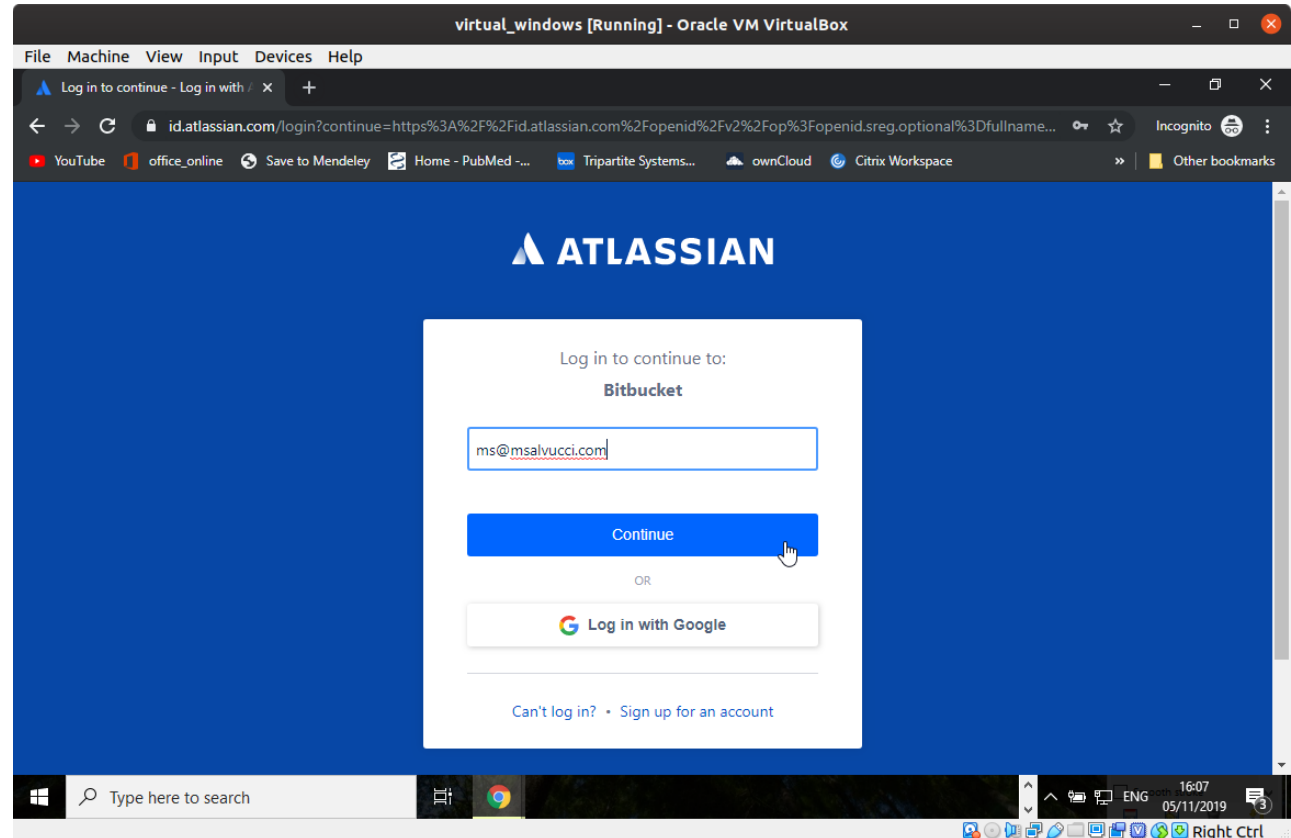
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Verify email

BITBUCKET EXAMPLE

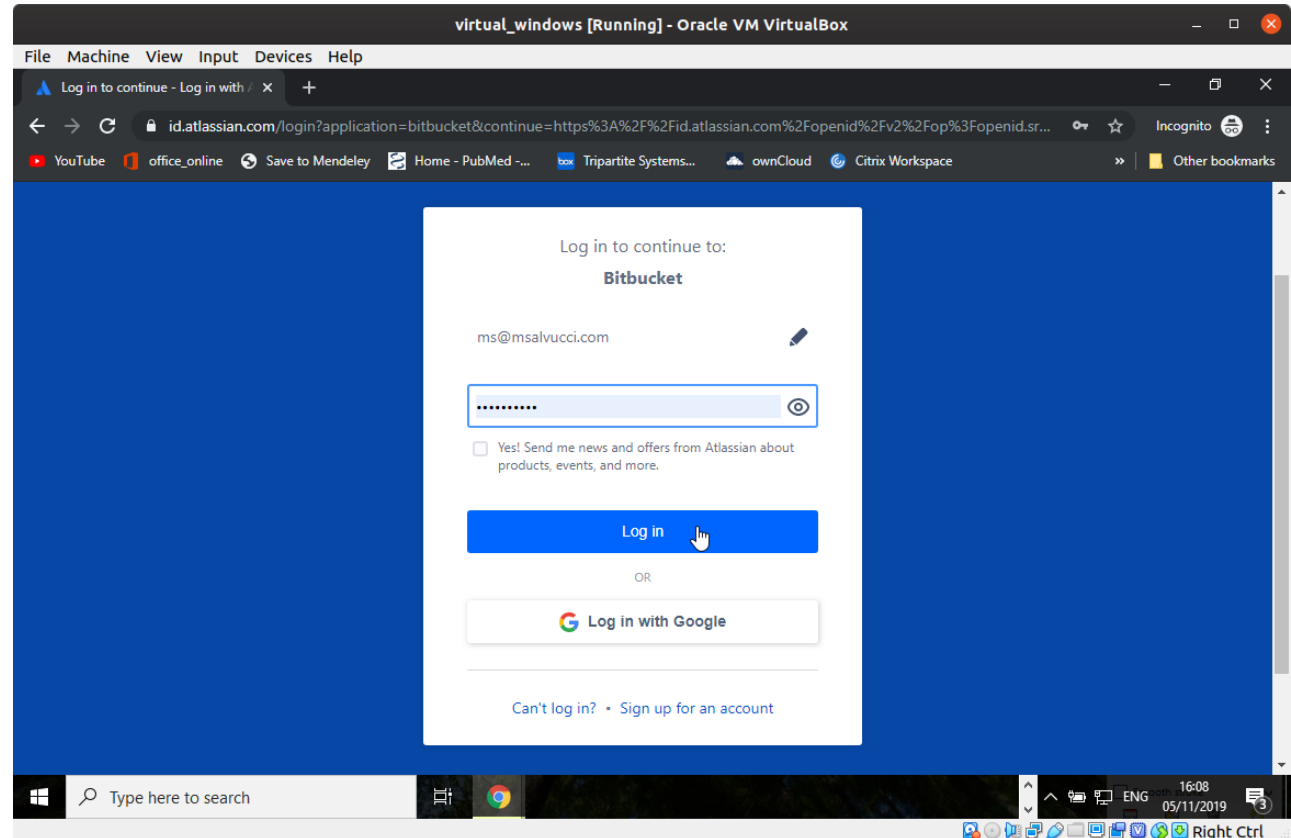
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Log in with your credential

BITBUCKET EXAMPLE

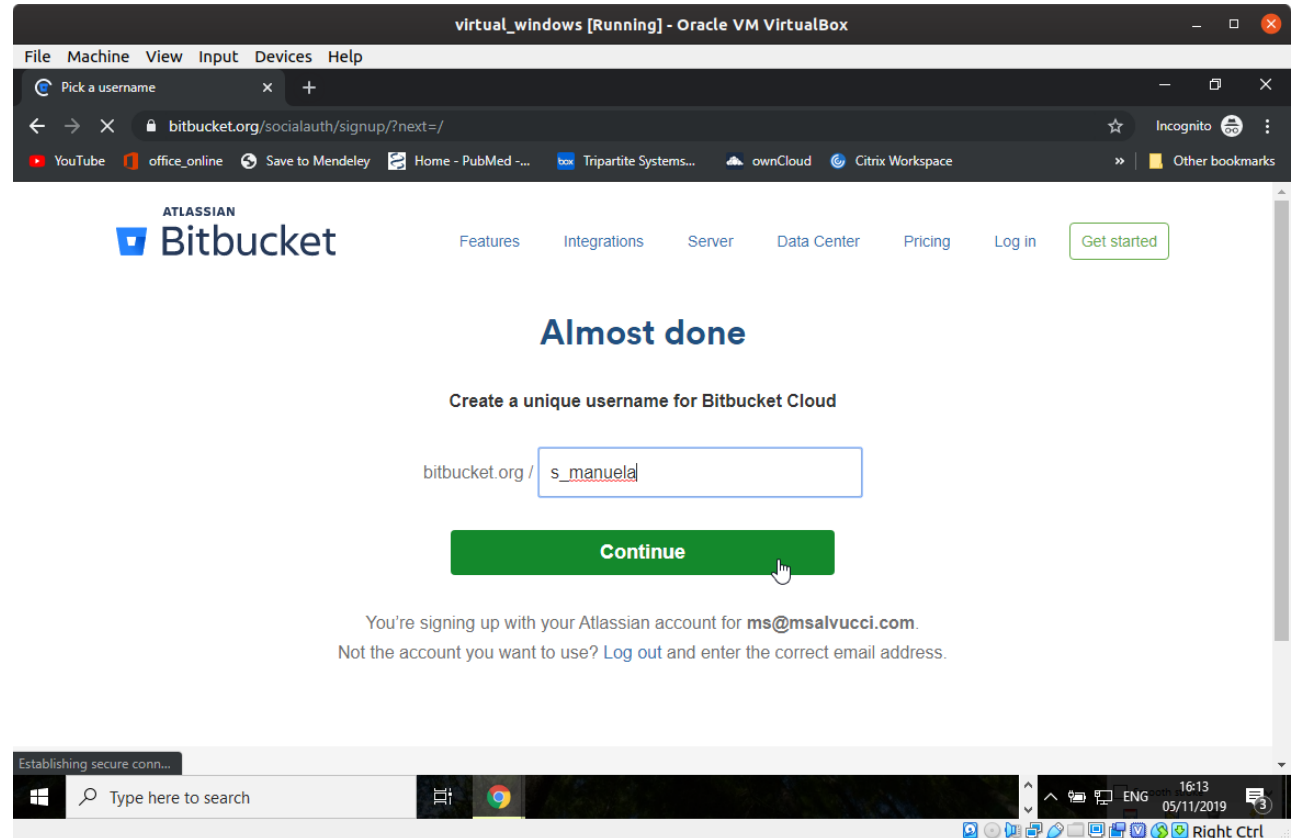
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Log in with your credential

BITBUCKET EXAMPLE

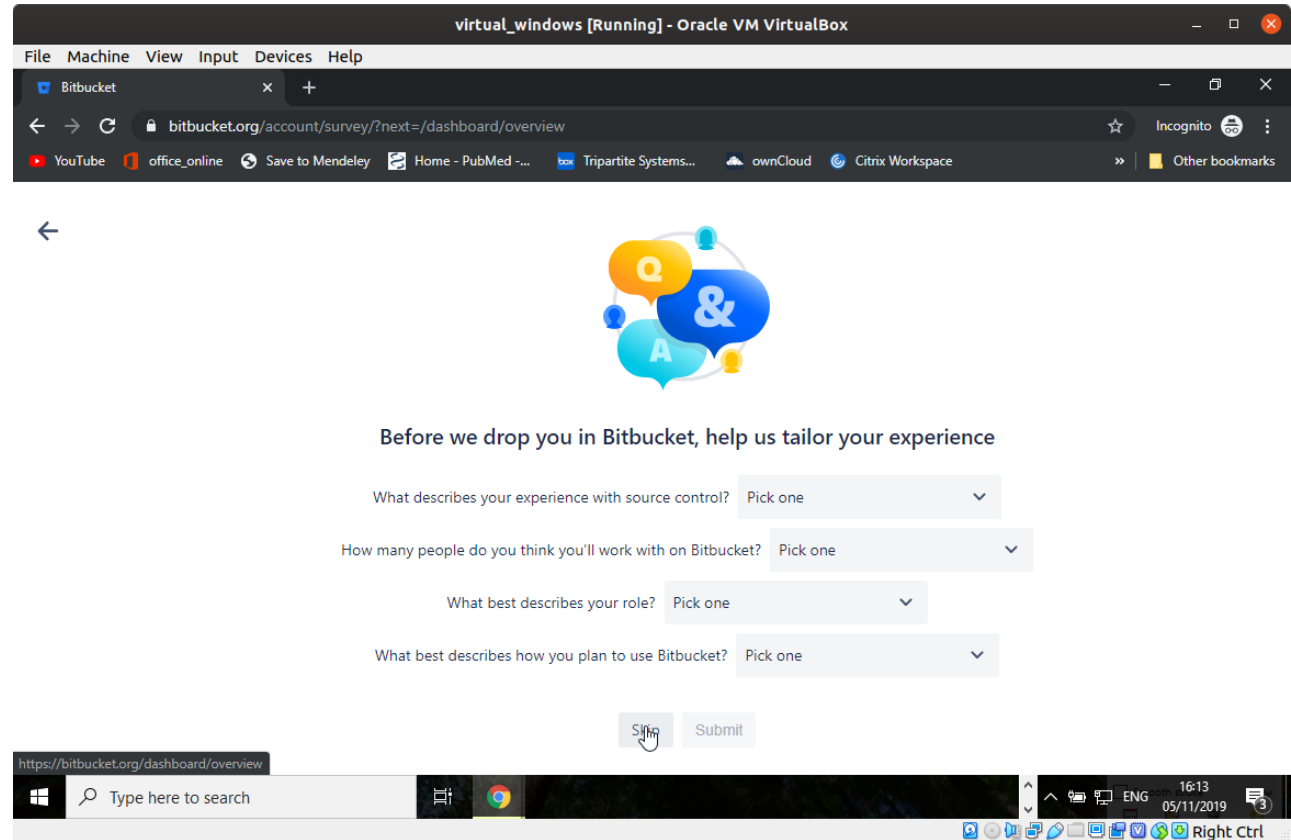
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Choose your username

BITBUCKET EXAMPLE

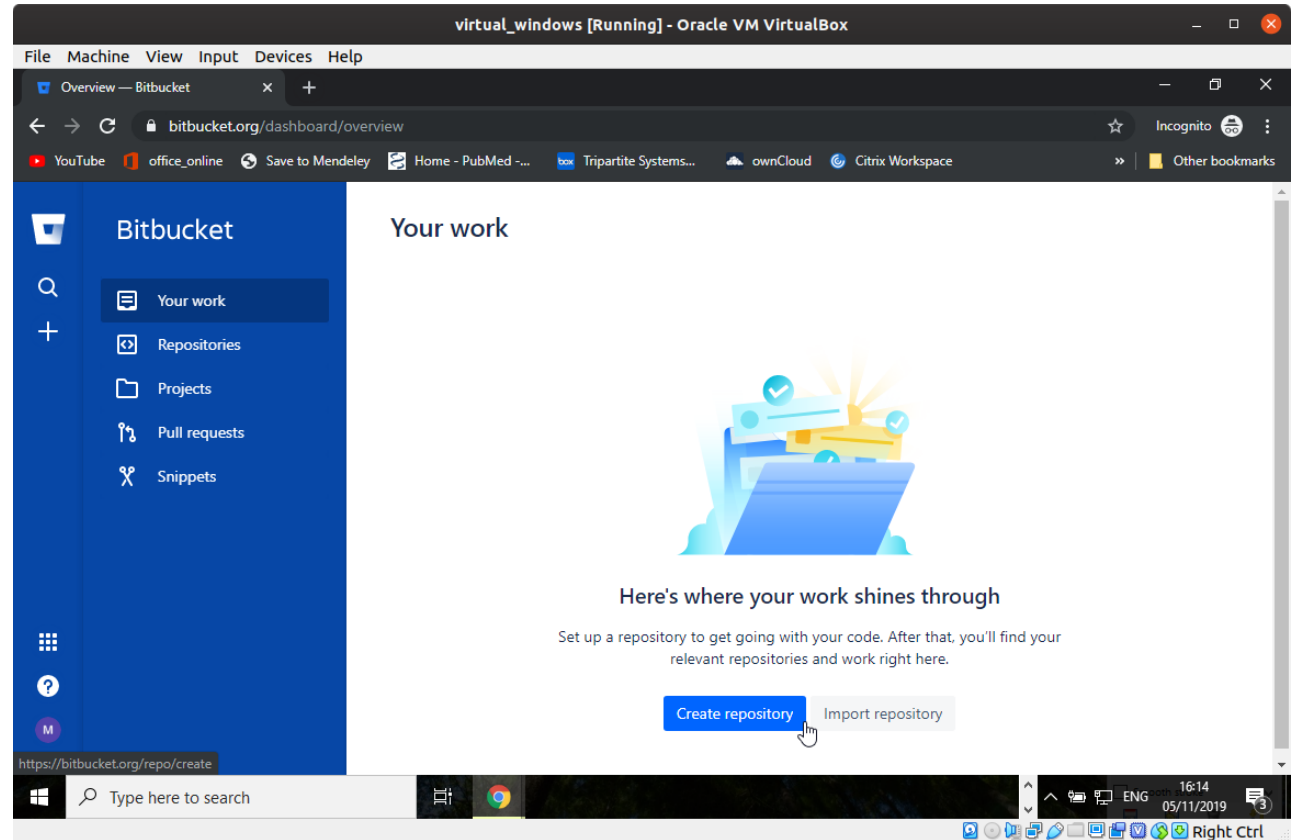
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Finalize setup

BITBUCKET EXAMPLE

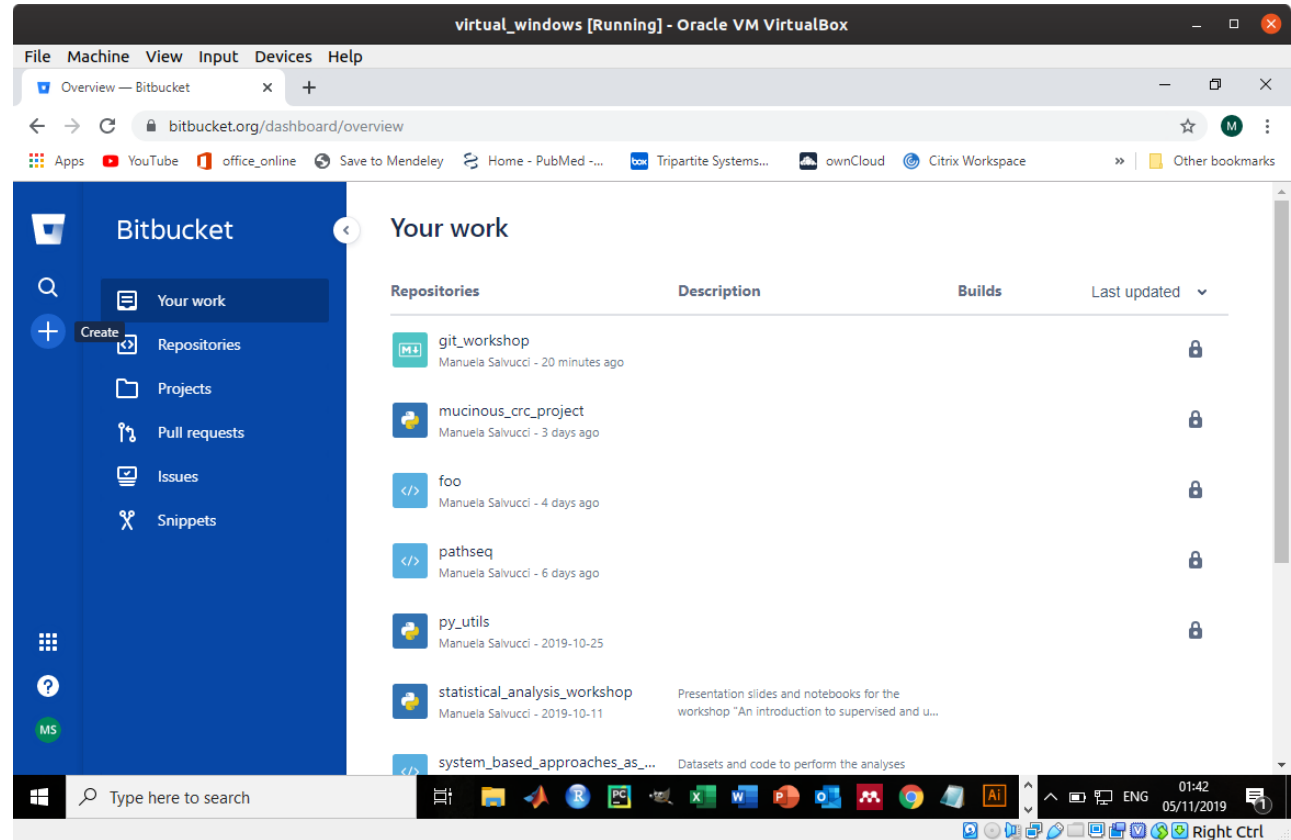
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Complete account creation

BITBUCKET EXAMPLE

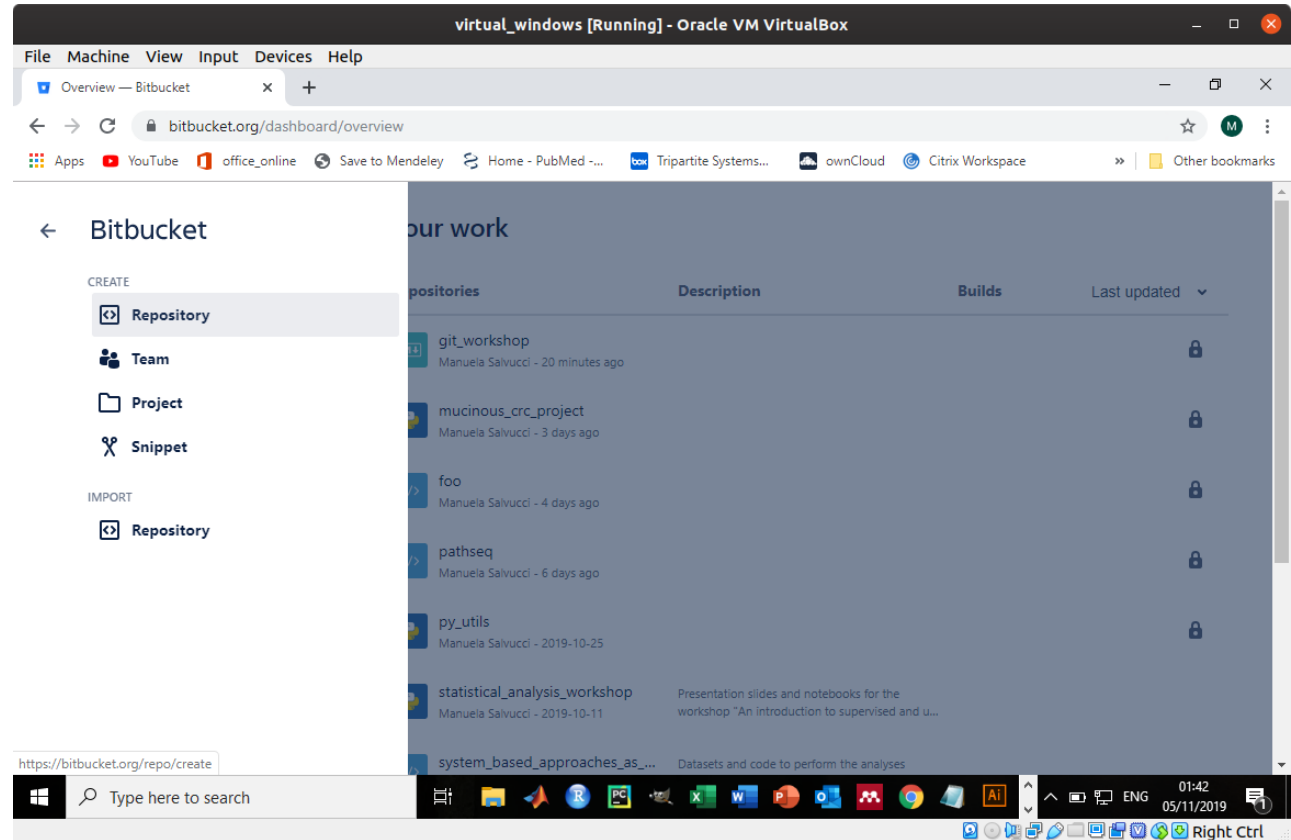
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Create a repository for the demo

BITBUCKET EXAMPLE

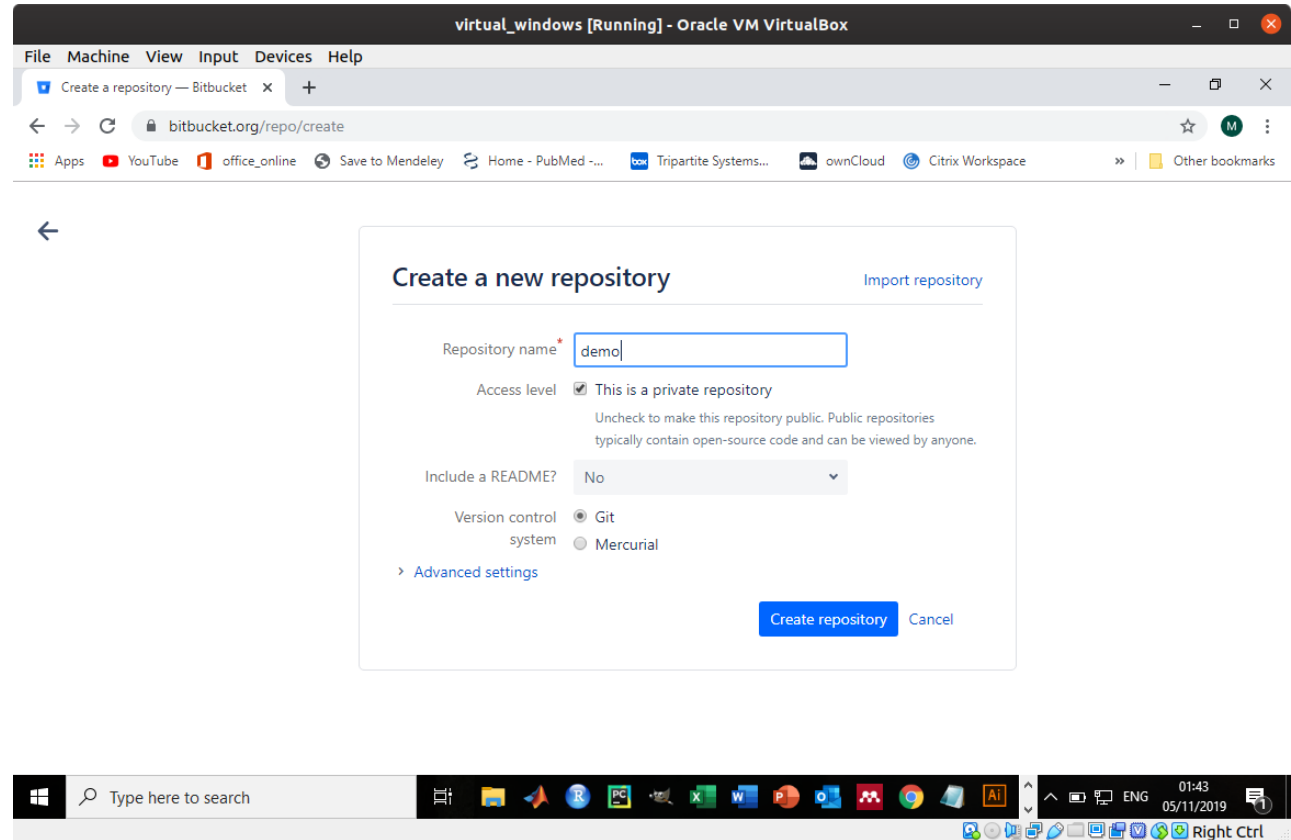
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Create a repository for the demo

BITBUCKET EXAMPLE

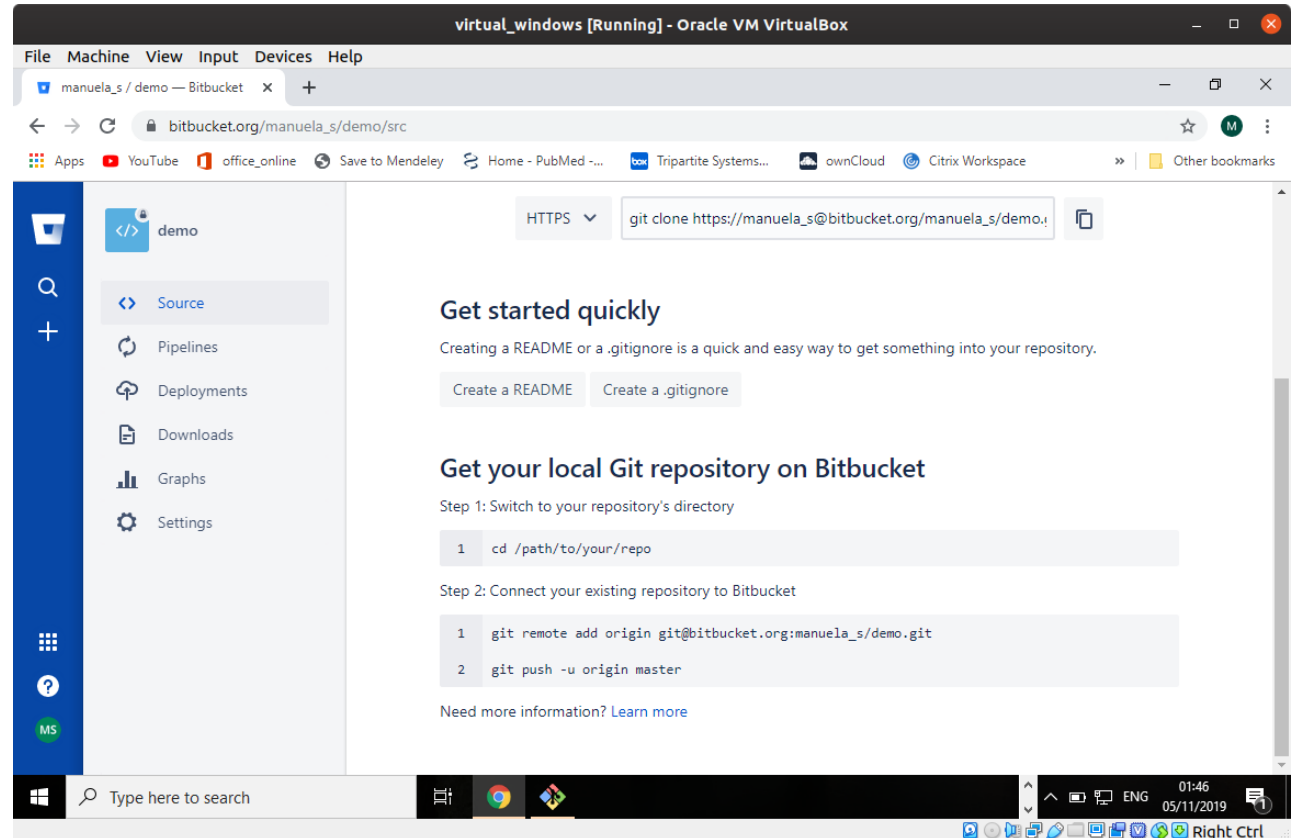
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Create a repository for the demo

BITBUCKET EXAMPLE

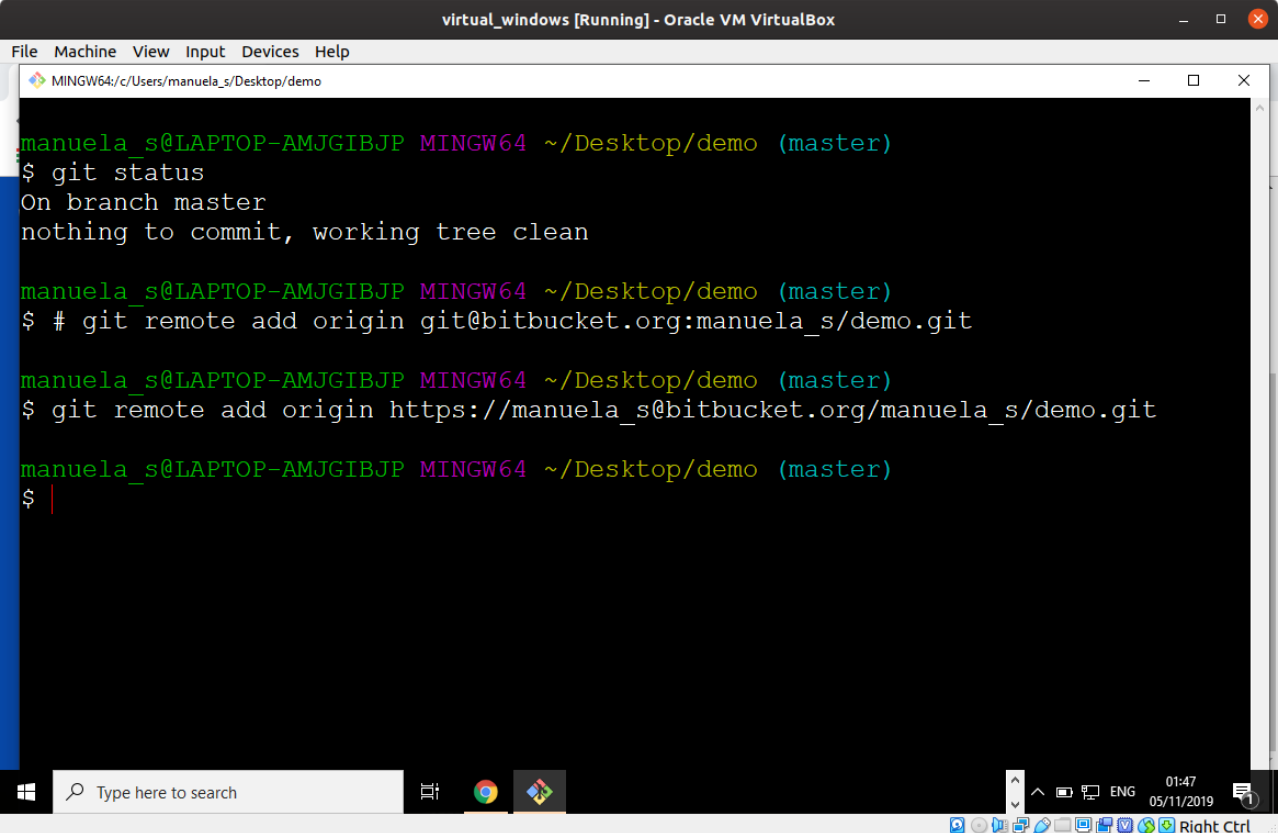
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Since we have an existing repository to upload, we follow the instructions for Get your local repository on BitBucket

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64:/c:/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git status
On branch master
nothing to commit, working tree clean

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ # git remote add origin git@bitbucket.org:manuela_s/demo.git

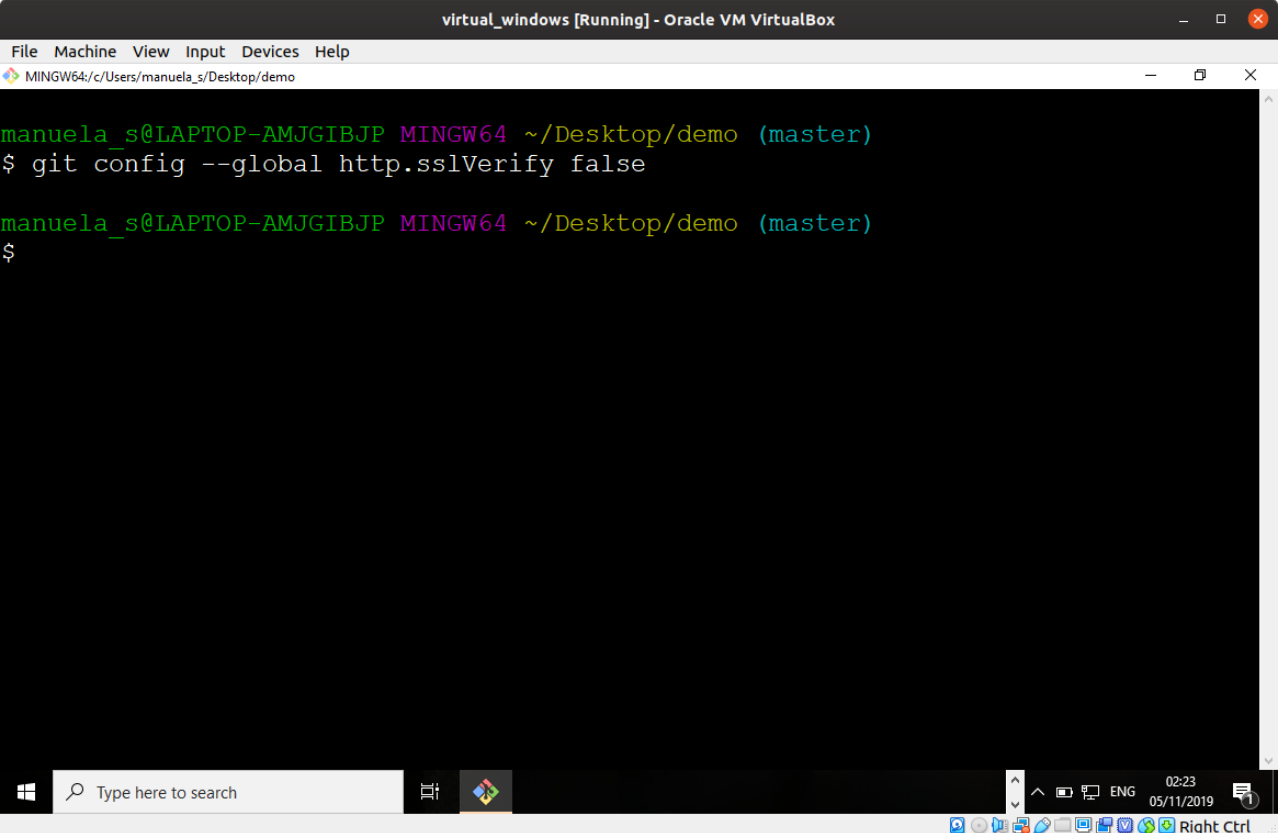
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git remote add origin https://manuela_s@bitbucket.org/manuela_s/demo.git

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ |
```

We go to the GIT bash to upload. We need to use the https protocol (instead of ssh) on the RCSI network

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history

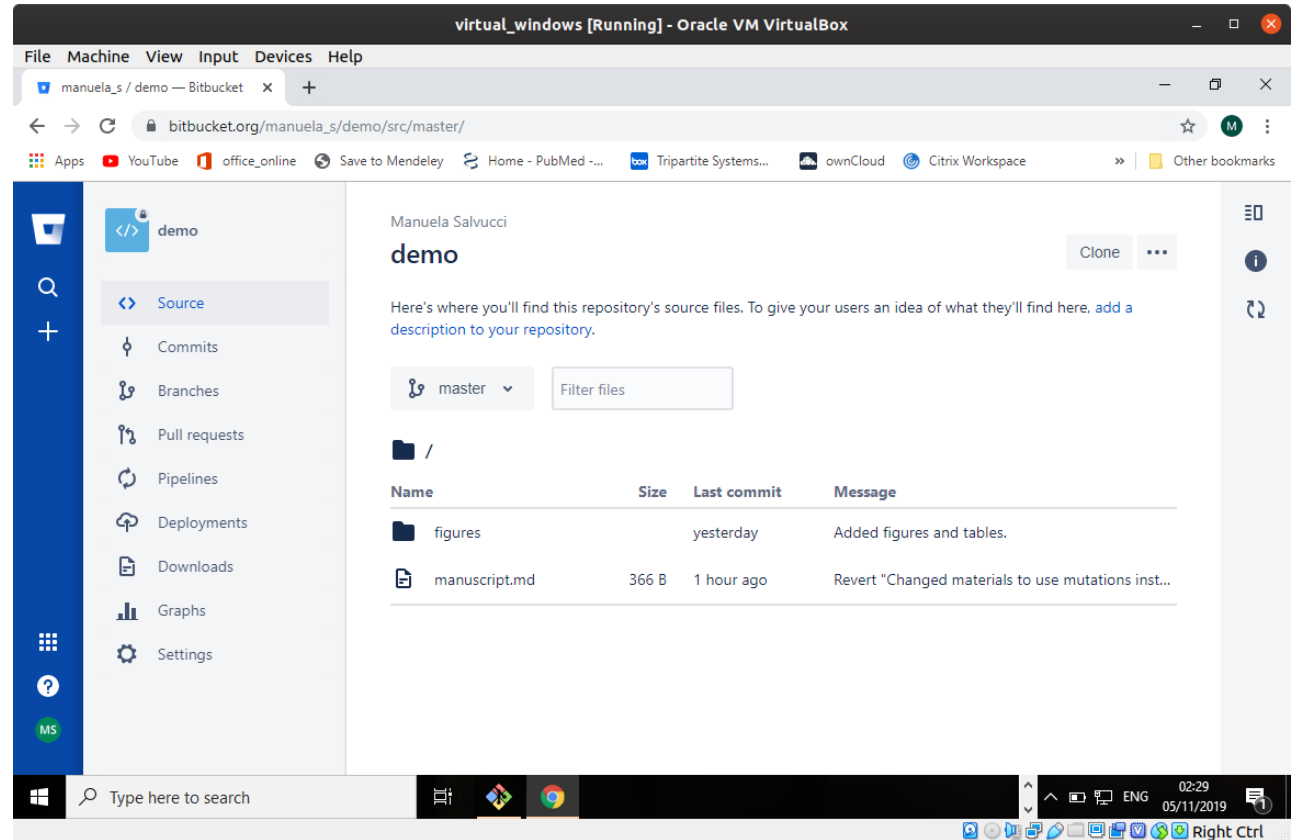


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/demo
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$ git config --global http.sslVerify false
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/demo (master)
$
```

Also to use git from the RCSI network, we need a workaround for ssl verification

BITBUCKET EXAMPLE

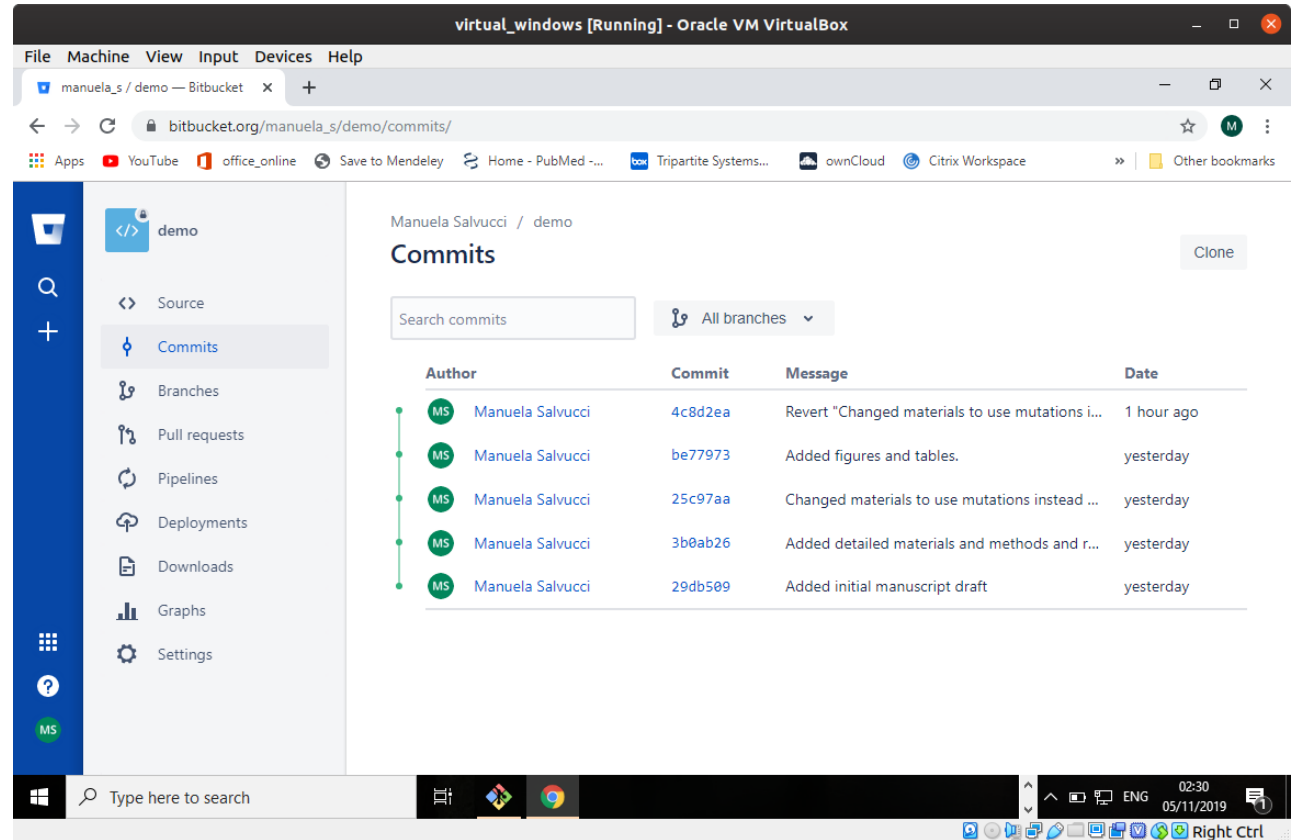
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



The BitBucket landing page for the repository shows the list of files and when they were last changed

BITBUCKET EXAMPLE

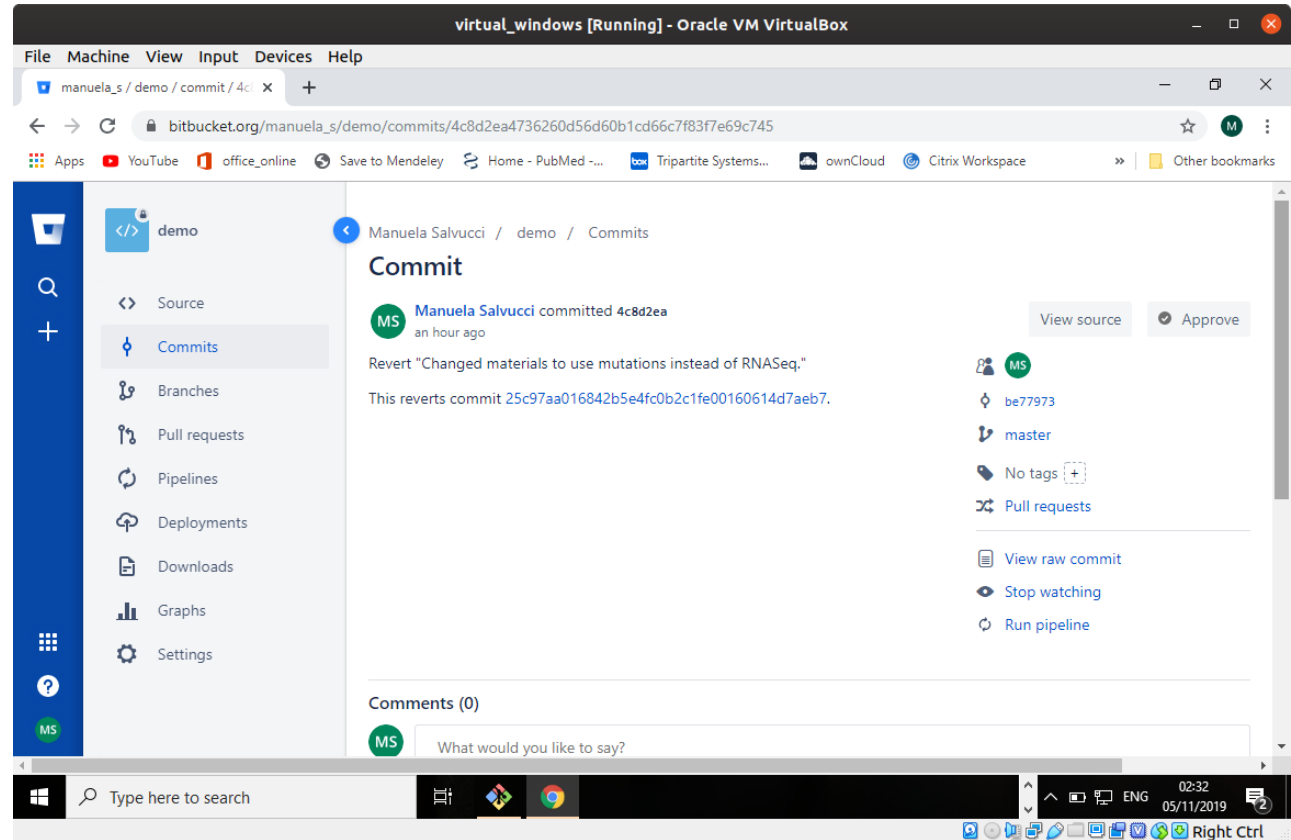
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



We can see the history of commits

BITBUCKET EXAMPLE

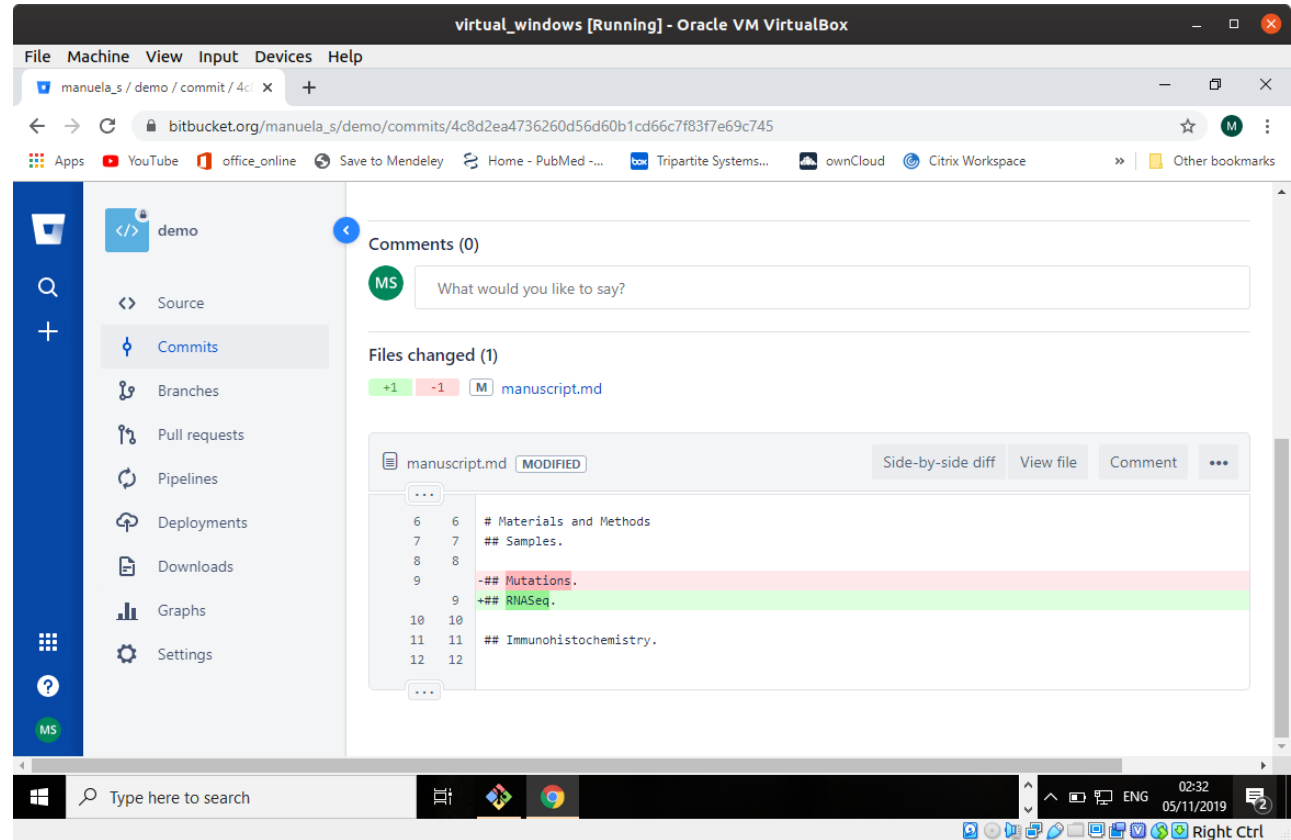
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



The content of the last commit

BITBUCKET EXAMPLE

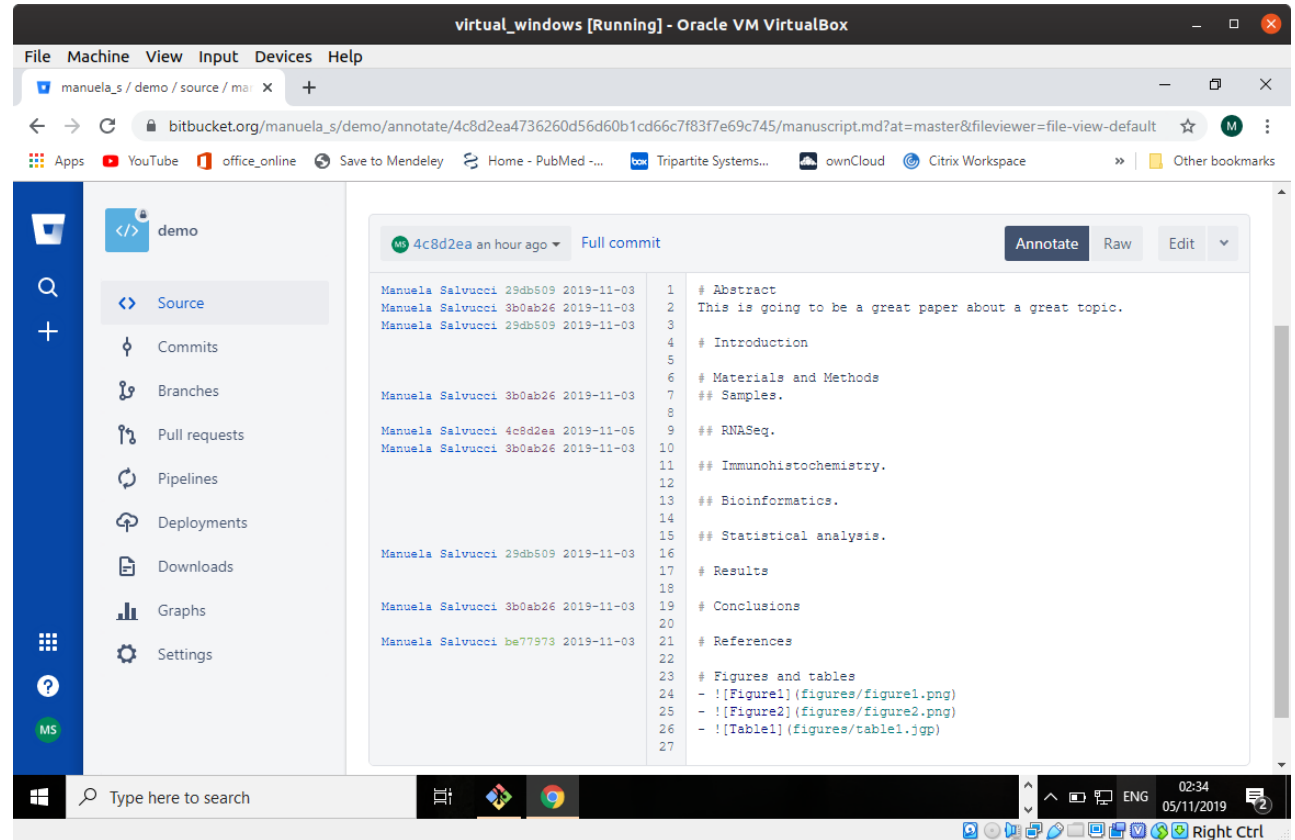
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



The content of the last commit

BITBUCKET EXAMPLE

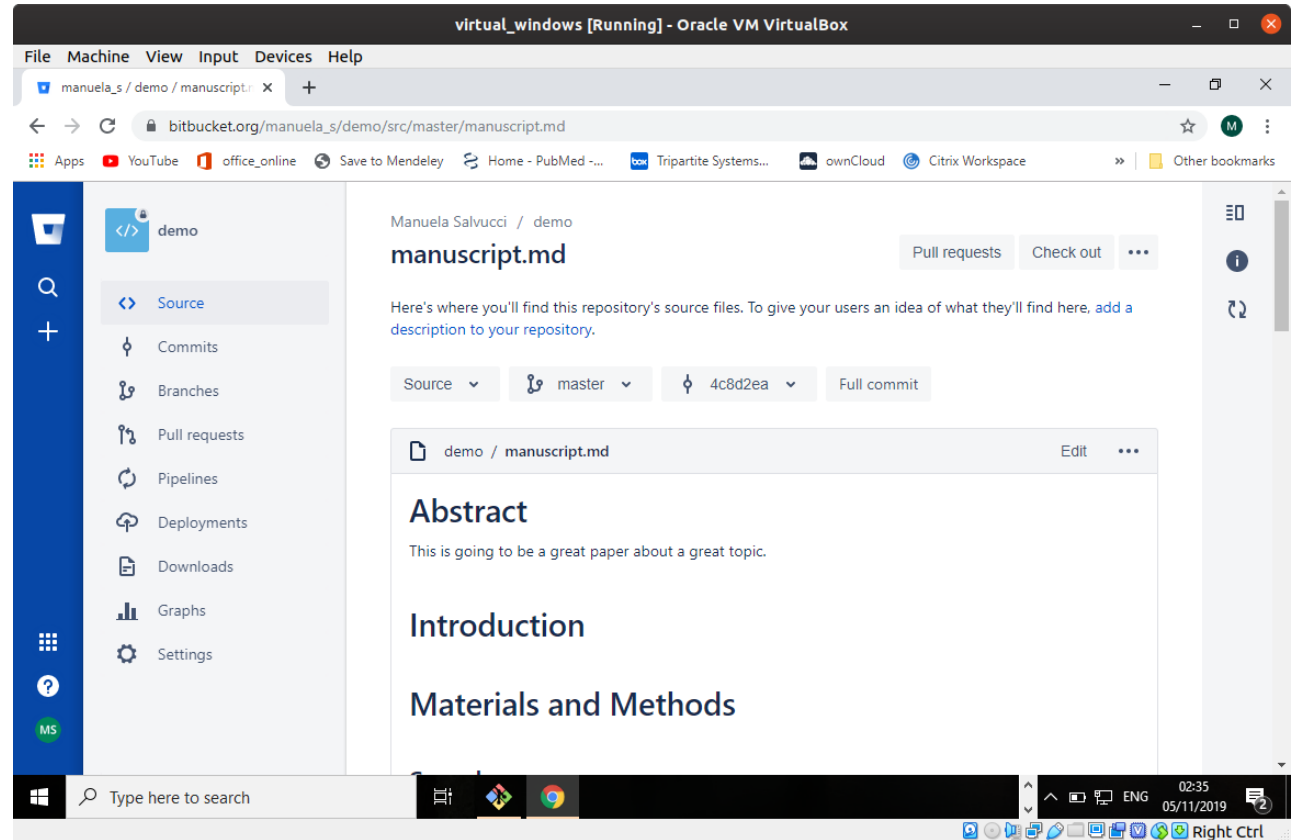
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



BitBucket has an annotate feature which highlights when each line in the file was last changed

BITBUCKET EXAMPLE

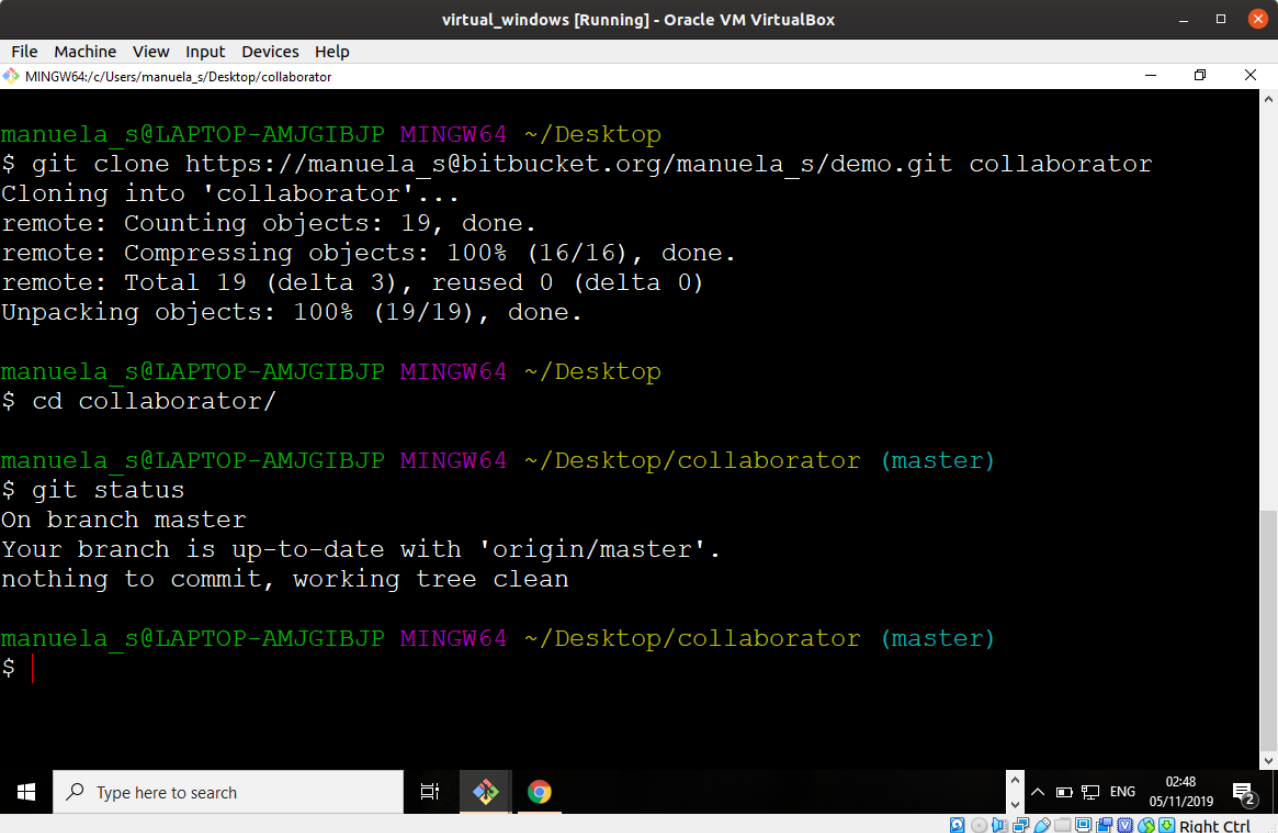
1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



Markdown rendering of the manuscript file

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop
$ git clone https://manuela_s@bitbucket.org/manuela_s/demo.git collaborator
Cloning into 'collaborator'...
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 19 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (19/19), done.

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop
$ cd collaborator/

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ |
```

Collaborator clones repository

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)

2. Making changes

3. Pushing and pulling

4. Inspecting history



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuscript.md - Notepad
File Edit Format View Help
# Abstract
This is going to be a great paper about a great topic.

# Introduction

# Materials and Methods
## Samples.

## RNASeq.

## Immunohistochemistry.

## Bioinformatics.
Some very complicated analysis was performed.

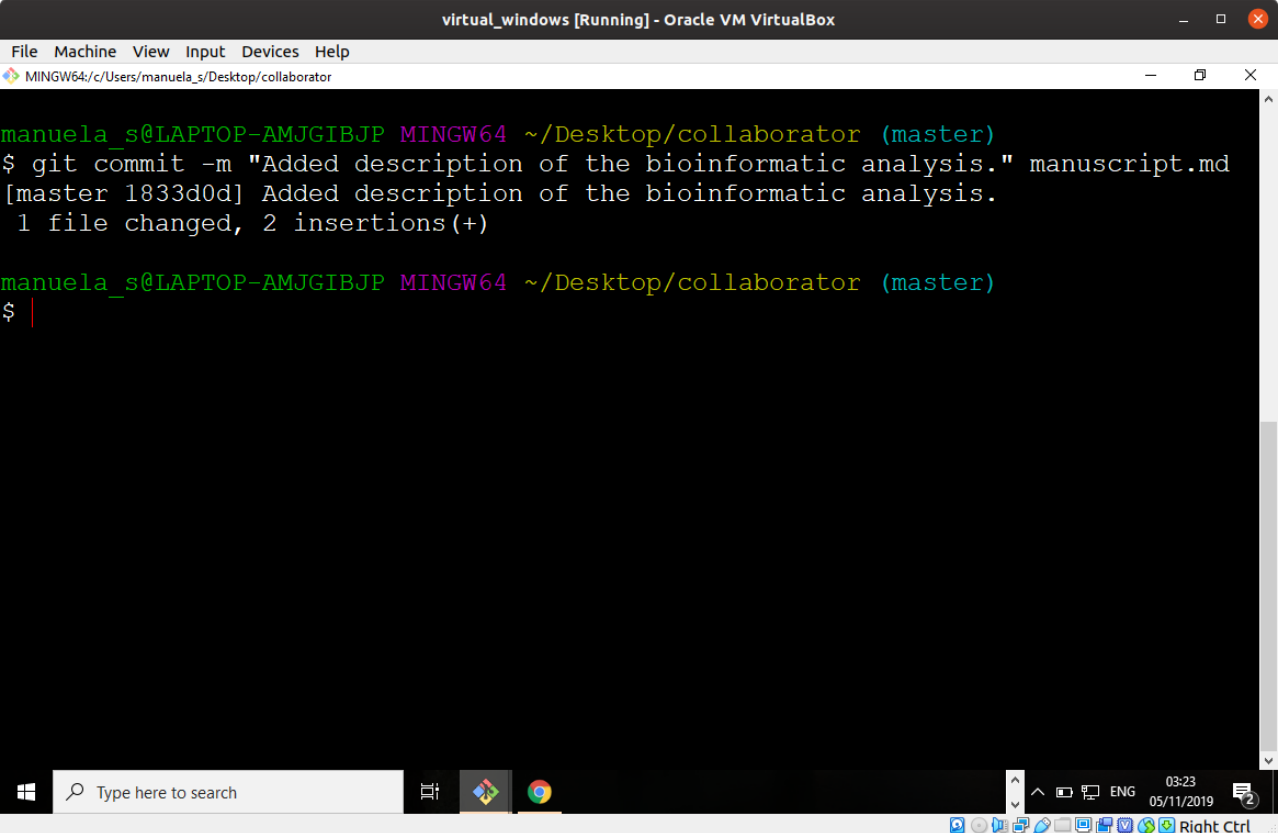
## Statistical analysis.
# Results
# Conclusions
# References

# Figures and tables
- ![Figure1](figures/figure1.png)
- ![Figure2](figures/figure2.png)
- ![Table1](figures/table1.jpg)
```

Collaborator adds text on bioinformatic analysis

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



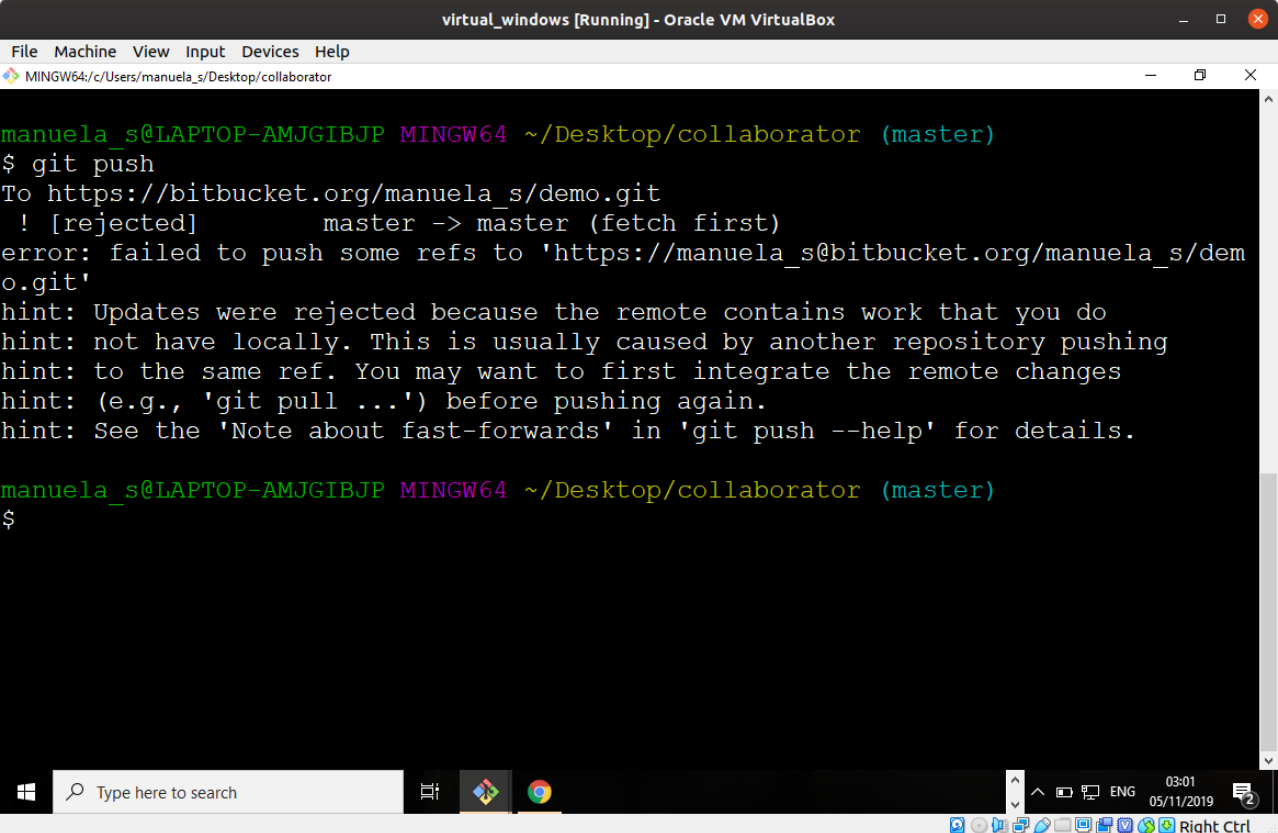
```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ git commit -m "Added description of the bioinformatic analysis." manuscript.md
[master 1833d0d] Added description of the bioinformatic analysis.
1 file changed, 2 insertions(+)

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ |
```

Collaborator commits their change

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



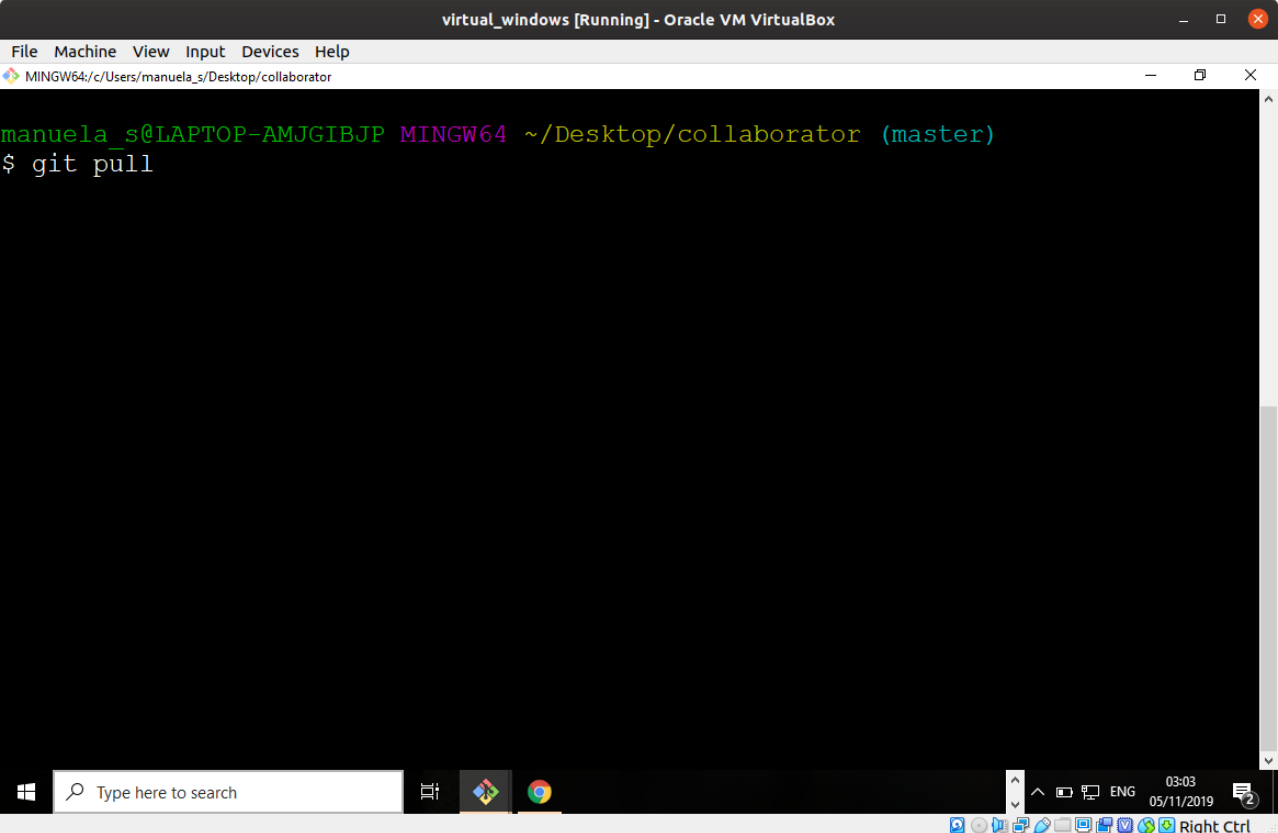
```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ git push
To https://bitbucket.org/manuela_s/demo.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://manuela_s@bitbucket.org/manuela_s/dem
o.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$
```

**Collaborator tries to push their change to the bitbucket server.
This fails, because another change has been made after they
clone**

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history

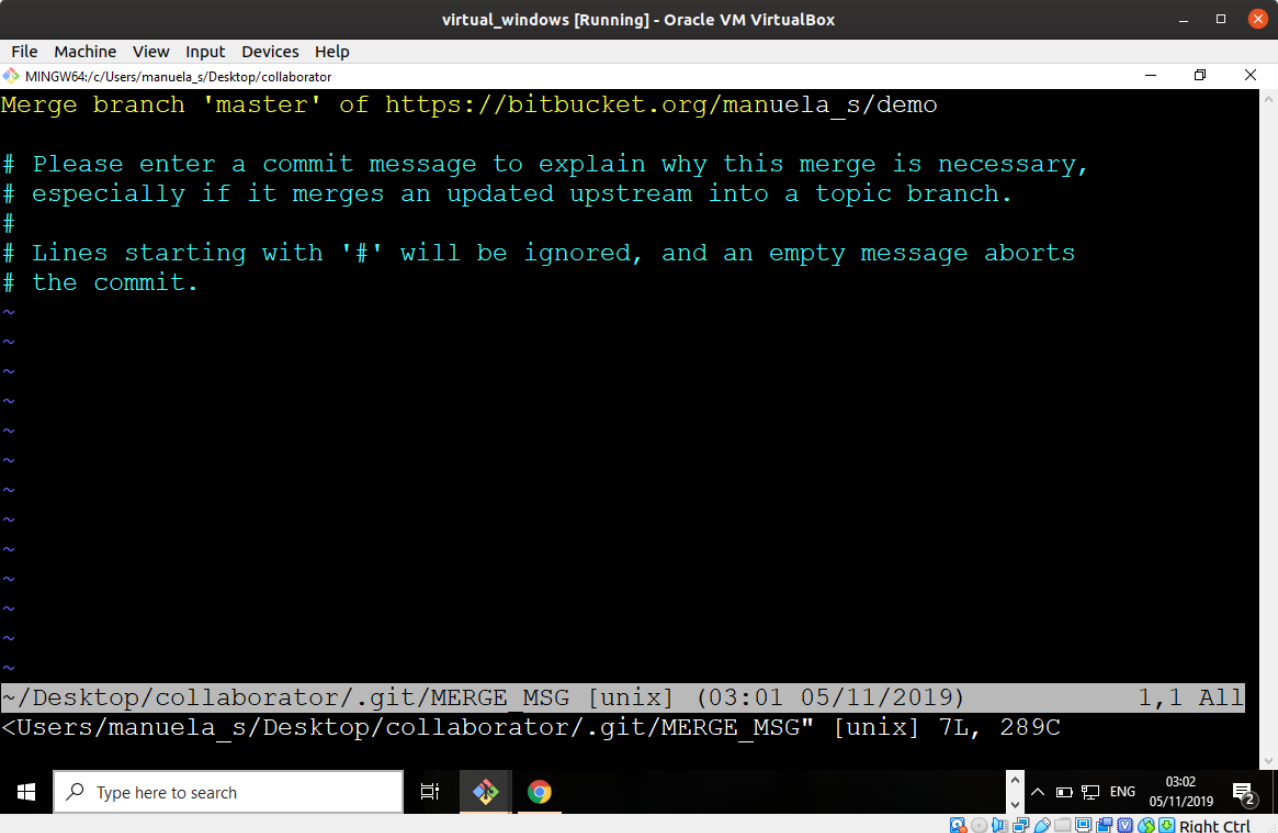


The screenshot shows a terminal window titled "virtual_windows [Running] - Oracle VM VirtualBox". The terminal prompt is "manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)". The user has entered the command "\$ git pull". The terminal output is currently empty, indicating the command has been executed but the results are not yet visible. The terminal window is running on a Windows operating system, as indicated by the taskbar at the bottom.

Collaborator needs to first pull from the server

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history

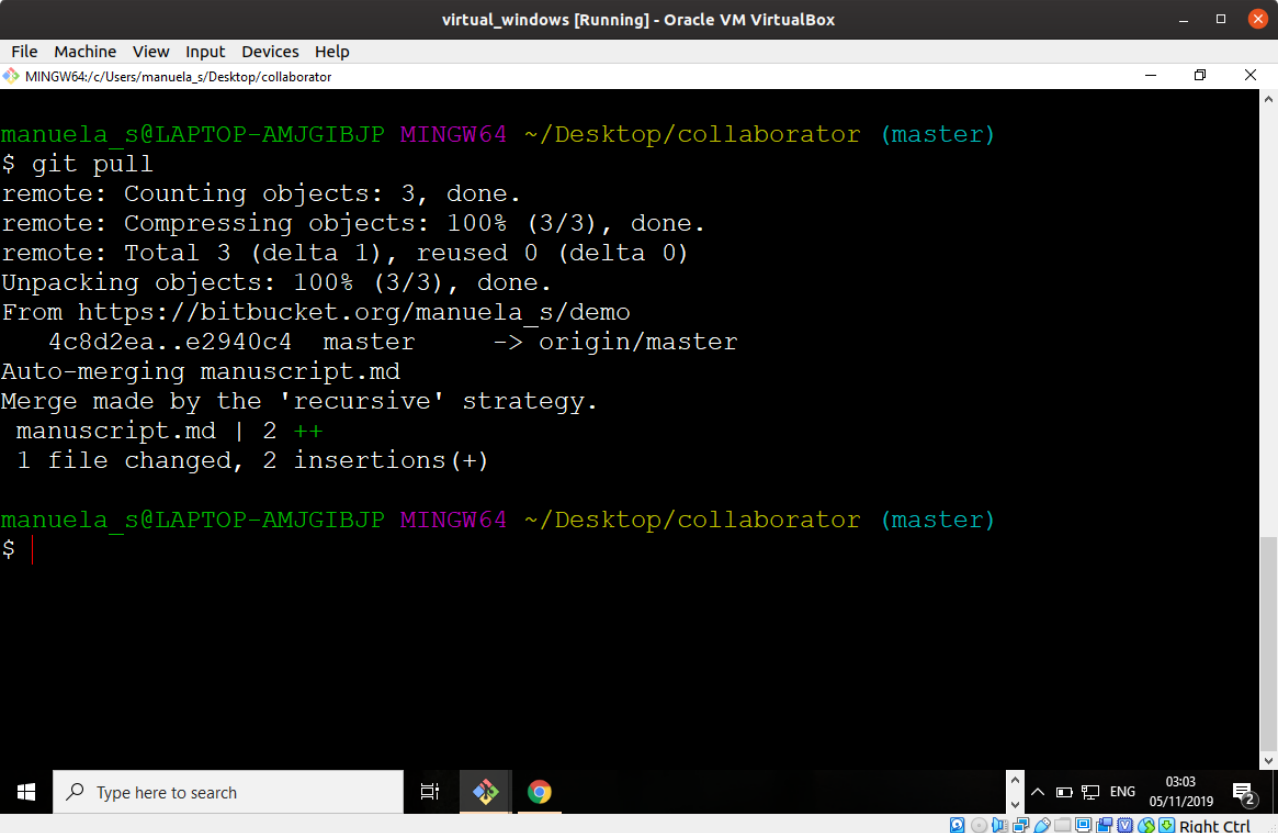


```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator
Merge branch 'master' of https://bitbucket.org/manuela_s/demo
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
~
~
~
~
~
~/Desktop/collaborator/.git/MERGE_MSG [unix] (03:01 05/11/2019) 1,1 All
<Users/manuela_s/Desktop/collaborator/.git/MERGE_MSG" [unix] 7L, 289C
```

The pull results in a merge between the two changes. They accept the default commit message for the merge

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



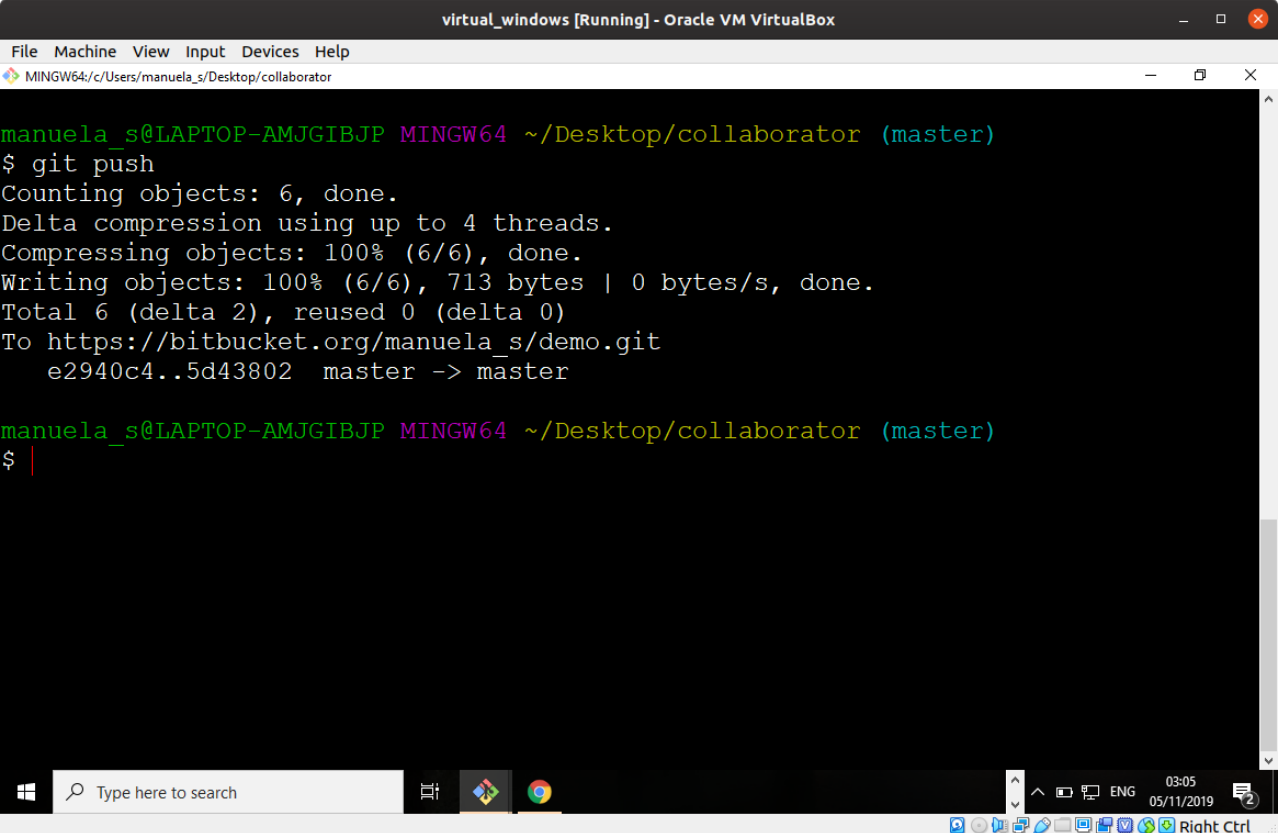
```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://bitbucket.org/manuela_s/demo
 4c8d2ea..e2940c4  master    -> origin/master
Auto-merging manuscript.md
Merge made by the 'recursive' strategy.
 manuscript.md | 2 ++
 1 file changed, 2 insertions(+)

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ |
```

The pull is successful

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario
 1. Cloning repository (before last commit)
 2. Making changes
 3. Pushing and pulling
 4. Inspecting history



```
virtual_windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINGW64/c/Users/manuela_s/Desktop/collaborator
manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ git push
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 713 bytes | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
To https://bitbucket.org/manuela_s/demo.git
 e2940c4..5d43802  master -> master

manuela_s@LAPTOP-AMJGIBJP MINGW64 ~/Desktop/collaborator (master)
$ |
```

Now they can push their change to the server

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)
2. Making changes
3. Pushing and pulling
4. Inspecting history

The screenshot shows a web browser window displaying the BitBucket interface for a repository named 'demo'. The 'Commits' page is active, showing a list of commits with columns for Author, Commit ID, Message, and Date. A commit history graph on the left shows a merge of two branches. The commit list includes:

Author	Commit	Message	Date
Collaborator	ded5b13	Merge branch 'master' of https://bitbuck...	30 seconds ago
Collaborator	1833d0d	Added description of the bioinformatic a...	2 minutes ago
Manuela Salvucci	e2940c4	Added patient samples information.	25 minutes ago
Manuela Salvucci	4c8d2ea	Revert "Changed materials to use mutatio...	2 hours ago
Manuela Salvucci	be77973	Added figures and tables.	yesterday
Manuela Salvucci	25c97aa	Changed materials to use mutations inste...	yesterday
Manuela Salvucci	3b0ab26	Added detailed materials and methods a...	yesterday
Manuela Salvucci	29db509	Added initial manuscript draft	yesterday

The server now shows history for the file that includes both the collaborators changes and my other simultaneous change, and show that they have been merged together

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)
2. Making changes
3. Pushing and pulling
4. Inspecting history

```
Manuela Salvucci 29db509 2019-11-03 1 # Abstract
Manuela Salvucci 3b0ab26 2019-11-03 2 This is going to be a great paper about a great topic.
Manuela Salvucci 29db509 2019-11-03 3
Manuela Salvucci 3b0ab26 2019-11-03 4 # Introduction
Manuela Salvucci 3b0ab26 2019-11-03 5
Manuela Salvucci 3b0ab26 2019-11-03 6 # Materials and Methods
Manuela Salvucci 3b0ab26 2019-11-03 7 ## Samples.
Manuela Salvucci e2940c4 2019-11-05 8
Manuela Salvucci e2940c4 2019-11-05 9 We used N=100 patient samples.
Manuela Salvucci 4c8d2ea 2019-11-05 10
Manuela Salvucci 4c8d2ea 2019-11-05 11 ## RNASeq.
Manuela Salvucci 3b0ab26 2019-11-03 12
Manuela Salvucci 3b0ab26 2019-11-03 13 ## Immunohistochemistry.
Manuela Salvucci 3b0ab26 2019-11-03 14
Manuela Salvucci 3b0ab26 2019-11-03 15 ## Bioinformatics.
Manuela Salvucci 3b0ab26 2019-11-03 16
Collaborator 1833d0d 2019-11-05 17 Some very complicated analysis was performed.
Manuela Salvucci 3b0ab26 2019-11-03 18
Manuela Salvucci 29db509 2019-11-03 19 ## Statistical analysis.
Manuela Salvucci 29db509 2019-11-03 20
Manuela Salvucci 3b0ab26 2019-11-03 21 # Results
Manuela Salvucci 3b0ab26 2019-11-03 22
Manuela Salvucci 3b0ab26 2019-11-03 23 # Conclusions
Manuela Salvucci 3b0ab26 2019-11-03 24
Manuela Salvucci be77973 2019-11-03 25 # References
Manuela Salvucci be77973 2019-11-03 26
Manuela Salvucci be77973 2019-11-03 27 # Figures and tables
Manuela Salvucci be77973 2019-11-03 28 - ![Figure1] (figures/figure1.png)
Manuela Salvucci be77973 2019-11-03 29 - ![Figure2] (figures/figure2.png)
Manuela Salvucci be77973 2019-11-03 30 - ![Table1] (figures/table1.jpg)
Manuela Salvucci be77973 2019-11-03 31
```

When we annotate the file we see the bioinformatic analysis text from the collaborator and the samples information from our change

USING BRANCHES AND TAGS

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)
2. Making changes
3. Pushing and pulling
4. Inspecting history

The screenshot shows the BitBucket web interface for a repository named 'demo'. The 'Commits' page is displayed, showing a list of commits with columns for Author, Commit ID, Message, and Date. A history graph on the left shows a main branch (green) and a 'figures' branch (red) that branches off from the main branch. The commit list includes:

Author	Commit	Message	Date
Manuela Salvucci	ff58bc6	Added captions for figures a... figures	1 minute ago
Collaborator	ded5b13	Merge branch 'master' of https://bitbuc...	24 minutes ago
Collaborator	1833d0d	Added description of the bioinformatic ...	26 minutes ago
Manuela Salvucci	e2940c4	Added patient samples information.	49 minutes ago
Manuela Salvucci	4c8d2ea	Revert "Changed materials to use mutati..."	3 hours ago
Manuela Salvucci	be77973	Added figures and tables.	yesterday
Manuela Salvucci	25c97aa	Changed materials to use mutations inst...	yesterday
Manuela Salvucci	3b0ab26	Added detailed materials and methods ...	yesterday
Manuela Salvucci	29db509	Added initial manuscript draft	yesterday

The history graph shows a figures branch for work on the figures that is kept separate from the rest

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)
2. Making changes
3. Pushing and pulling
4. Inspecting history

The screenshot shows a BitBucket web interface for a repository named 'demo'. The 'Commits' page is displayed, showing a list of commits with columns for Author, Commit ID, Message, and Date. A commit graph on the left shows the sequence of commits and branches. The most recent commit is a merge of the 'figures' branch into the 'master' branch.

Author	Commit	Message	Date
Manuela Salvucci	c080e72	Merge branch 'master' of https://bitbu...	46 seconds ago
Manuela Salvucci	5b4caed	Merge branch 'figures'	1 minute ago
Manuela Salvucci	ff58bc6	Added captions for figures and tables.	2 minutes ago
Collaborator	ded5b13	Merge branch 'master' of https://bitbu...	25 minutes ago
Collaborator	1833d0d	Added description of the bioinformatic...	27 minutes ago
Manuela Salvucci	e2940c4	Added patient samples information.	50 minutes ago
Manuela Salvucci	4c8d2ea	Revert "Changed materials to use muta...	3 hours ago
Manuela Salvucci	be77973	Added figures and tables.	yesterday
Manuela Salvucci	25c97aa	Changed materials to use mutations in...	yesterday

The figures branch was merged with the rest of the work

BITBUCKET EXAMPLE

1. Create account and log in
2. Push demo repository to BitBucket
3. Show history and diffs
4. Collaboration scenario

1. Cloning repository (before last commit)
2. Making changes
3. Pushing and pulling
4. Inspecting history

The screenshot shows a BitBucket web interface for a repository named 'demo'. The 'Commits' page is displayed, showing a list of commits with columns for Author, Commit ID, Message, and Date. A commit history graph is visible on the left side of the commit list, showing the sequence of commits and their relationships. The commit list includes several commits by 'Manuela Salvucci' and two by 'Collaborator'. The commit with ID 'ded5b13' is highlighted, indicating it is the current revision being shared with co-authors.

Author	Commit	Message	Date
Manuela Salvucci	c080e72	Merge branch 'master' of https://bitbu...	9 minutes ago
Manuela Salvucci	5b4caed	Merge branch 'figures'	10 minutes ago
Manuela Salvucci	ff58bc6	Added captions for figures and tables.	11 minutes ago
Collaborator	ded5b13	Merge branch ... sent_to_co-authors	34 minutes ago
Collaborator	1833d0d	Added description of the bioinformatic...	36 minutes ago
Manuela Salvucci	e2940c4	Added patient samples information.	59 minutes ago
Manuela Salvucci	4c8d2ea	Revert "Changed materials to use muta...	3 hours ago
Manuela Salvucci	be77973	Added figures and tables.	yesterday
Manuela Salvucci	25c97aa	Changed materials to use mutations in...	yesterday

We tagged the revision we shared with the other co-authors

EXERCISE 2

EXERCISE 2 (20 MIN)

1. Create a BitBucket account
2. Push your “christmas_repo” from exercise 1 to BitBucket
3. Add more past_gifts and push to BitBucket
4. Update recipients through the BitBucket web-interface, and pull to your local machine
5. Collaborate in pairs

WRAP-UP

TAKE HOME MESSAGES




- Version control with GIT helps keep your file history organized
- Light weight: minimum effort required
- If you are not comfortable with using the command line, download a GUI or use GIT from your IDE
- Commit early and often
- Write good commit messages - future you will appreciate it
- Useful for backups and for collaboration

THANKS & QUESTIONS

- Get in touch: manuelasalvucci@rcsi.ie
- Presentation, CheatSheet, Handout and Solution:
https://bitbucket.org/manuela_s/git_workshop/downloads/
- Workshop repo:
https://bitbucket.org/manuela_s/git_workshop
- Useful resources
 - <https://git-scm.com/>
 - <https://git-scm.com/book/en/v2>
 - <https://try.github.io/>
 - <https://stackoverflow.com/questions/tagged/git>
 - <https://sethrobertson.github.io/GitBestPractices/>

In case of fire



1.  **git commit**
2.  **git push**
3.  **leave building**

From
<https://raw.githubusercontent.com/hendrixroa/in-case-of-fire-1/master/>