Practical Computing for Scientists

Armin Sobhani

CSCI 2000U

UOIT – Fall 2015

Checkpoint 6

- Review the *Self-Assessment* :
  - Blackboard > Course Content > Week 3 (Sept. 28- Oct. 2) > Monday Sept. 28 > Checkpoint 6

Checkpoint 6 – Question 1

1. In a directory I have the following files with Messier object (M) data and NGC data. Any data which has been altered is marked with an 'a'.
   - `M01.txt`
   - `M41.txt`
   - `M81.txt`
   - `M81a.txt`
   - `M101.tbl`
   - `M105.txt`
   - `M105a.txt`
   - `M107.txt`
   - `NGC4791.txt`
   - `NGC4791a.txt`
   - `NGC6371.txt`

Checkpoint 6 – Question 1

I want a list of data for only objects with messier numbers 100 or greater, without alterations. Which of the following commands will produce such a list?

Checkpoint 6 – Question 1

I want a list of data for only objects with messier numbers 100 or greater, without alterations. Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- `M101.tbl`
- `M105.txt`
- `M105a.txt`
- `M107.txt`
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- `M101.tbl`
- `M105.txt`
- `M105a.txt`
- `M107.txt`
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ←
- **`M105.txt`** ←
- `M105a.txt`
- **`M107.txt`** ←
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ⟵
- **`M105.txt`** ⟵
- `M105a.txt`
- **`M107.txt`** ⟵
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

1. ls *
2. ls M1*.txt
3. ls M1?.txt
4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`**
- **`M105.txt`**
- `M105a.txt`
- **`M107.txt`**
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

1. [✓ ls *]
2. ls M1*.txt
3. ls M1?.txt
4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- M01.txt
- M41.txt
- M81.txt
- M81a.txt
- **M101.tbl**
- **M105.txt**
- M105a.txt
- **M107.txt**
- NGC4791.txt
- NGC4791a.txt
- NGC6371.txt

1. ls *
2. ls M1*.txt
3. ls M1?.txt
4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ←
- **`M105.txt`** ←
- `M105a.txt`
- **`M107.txt`** ←
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

1. ✗ ls *
2. ls M1*.txt
3. ls M1?.txt
4. ls M1??.*

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`**
- **`M105.txt`**
- `M105a.txt`
- **`M107.txt`**
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

1. ✗ ls *
2. ls M1*.txt
3. ls M1?.txt
4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ⬅
- **`M105.txt`** ⬅
- `M105a.txt`
- **`M107.txt`** ⬅
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

✗ 1. ls *

✗ 2. ls M1*.txt

3. ls M1?.txt

4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- M01.txt
- M41.txt
- M81.txt
- M81a.txt
- **M101.tbl**
- **M105.txt**
- M105a.txt
- **M107.txt**
- NGC4791.txt
- NGC4791a.txt
- NGC6371.txt

1. ✗ ls *
2. ✗ ls M1*.txt
3. [✓ ls M1?.txt]
4. ls M1??.*

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ⬅
- **`M105.txt`** ⬅
- `M105a.txt`
- **`M107.txt`** ⬅
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

1. ✗ ls *
2. ✗ ls M1*.txt
3. ✗ ls M1?.txt
4. ls M1??.*

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 1

I want a list of data for only objects with **messier numbers 100 or greater, without alterations.** Which of the following commands will produce such a list?

- `M01.txt`
- `M41.txt`
- `M81.txt`
- `M81a.txt`
- **`M101.tbl`** ←
- **`M105.txt`** ←
- `M105a.txt`
- **`M107.txt`** ←
- `NGC4791.txt`
- `NGC4791a.txt`
- `NGC6371.txt`

✗ 1. ls *

✗ 2. ls M1*.txt

✗ 3. ls M1?.txt

✓ 4. ls M1??.*

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 1

M1*.txt

Checkpoint 6 – Question 1

$\boxed{\text{M1*txt}}$

M1*txt

* can be any character, one or more than one

Checkpoint 6 – Question 1

M1*txt

* can be any character, one or more than one

For example:

M10
M1d
M168
M1slkw28
M1sk3kdj438hskdn3
...

Checkpoint 6 – Question 1

M1*.txt

* can be any character, one or more than one

$\text{ls M1}\boxed{?}\text{.txt}$

? can be just one character!

Checkpoint 6 – Question 1

M1*.txt

* can be any character, one or more than one

ls M1?.txt

? can be just one character!

ls M1$\boxed{??}$.*

?? can be two characters!

Checkpoint 6 – Question 5

5. Greg needs a list of all of Bob's homework assignments; he needs the full paths for each homework file, like this:

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

Greg tried the four variations on the find command below; which one of these commands returned his desired result?

```
1. $ find *BOB*
2. $ find -name '*BOB*' homework_*
3. $ find . -name './homework*/BOB*'
4. $ find . -name 'BOB*'
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1. $ find *BOB*
2. $ find -name '*BOB*' homework_*
3. $ find . -name './homework*/BOB*'
4. $ find . -name 'BOB*'
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

```
$ find     (path)    -name    (pattern)
                     -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1. $ find *BOB*
2. $ find -name '*BOB*' homework_*
3. $ find . -name './homework*/BOB*'
4. $ find . -name 'BOB*'
```

```
$ find     (path)    -name    (pattern)
                     -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1. $ find *BOB*
2. $ find -name '*BOB*' homework_*
3. $ find . -name './homework*/BOB*'
4. $ find . -name 'BOB*'
```

```
$ find    (path)    -name    (pattern)
                    -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

```
./
+-- homework_1/
|    +-- BOB.hw1
|    +-- SAM.hw1
|    +-- ROB.hw1
               .
               .
               .
+-- homework_2/
|    +-- BOB.hw2
|    +-- SAM.hw2
|    +-- ROB.hw2
               .
               .
               .
```

```
$ find *BOB*
find: no such a file or directory
$
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*  ✗
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

```
$ find     (path)    -name    (pattern)
                     -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*  ✖
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

```
$ find    (path)   -name   (pattern)
                   -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*  ✖
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

Path

```
$ find     (path)    -name    (pattern)
                     -type
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

Path

```
$ find      (path)     -name     (pattern)
                       -type
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

```
$ find     (path)    -name    (pattern)
                     -type
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

It is not a pattern!

It is mixing of the path and the pattern!

```
$ find     (path)    -name    (pattern)
                     -type
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'
```

```
$ find     (path)    -name    (pattern)
                     -type
```

```
./
+-- homework_1/
|    +-- BOB.hw1
|    +-- SAM.hw1
|    +-- ROB.hw1
              .
              .
              .
+-- homework_2/
|    +-- BOB.hw2
|    +-- SAM.hw2
|    +-- ROB.hw2
              .
              .
              .
```

```
$ find . -name 'BOB*'
./homework_1/BOB.hw1
./homework_2/BOB.hw2
$
```

Checkpoint 6 – Question 5

```
./homework_1/BOB.hw1
./homework_2/BOB.hw2
```

```
1.  $ find *BOB*
2.  $ find -name '*BOB*' homework_*
3.  $ find . -name './homework*/BOB*'
4.  $ find . -name 'BOB*'   ✓
```

```
$ find     (path)    -name    (pattern)
                     -type
```

Version Control Systems

by Armin Sobhani

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Dealing with Change

Dealing with Change

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Dealing with Change

How do you manage your coursework?

Dealing with Change

How do you manage your coursework?

modifying existing code

Dealing with Change

How do you manage your coursework?

modifying existing code

backing up working code

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Dealing with Change

How do you manage your coursework?

modifying existing code

backing up working code

checking if an idea works

Dealing with Change

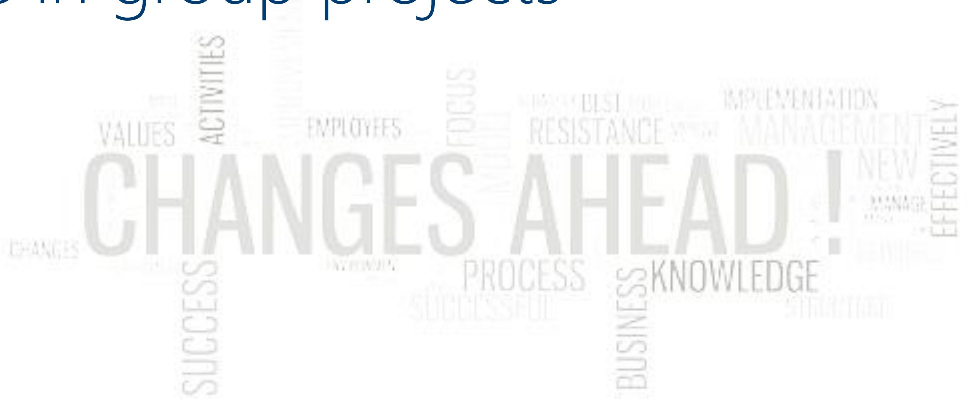
How do you manage your coursework?

modifying existing code

backing up working code

checking if an idea works

sharing code in group projects

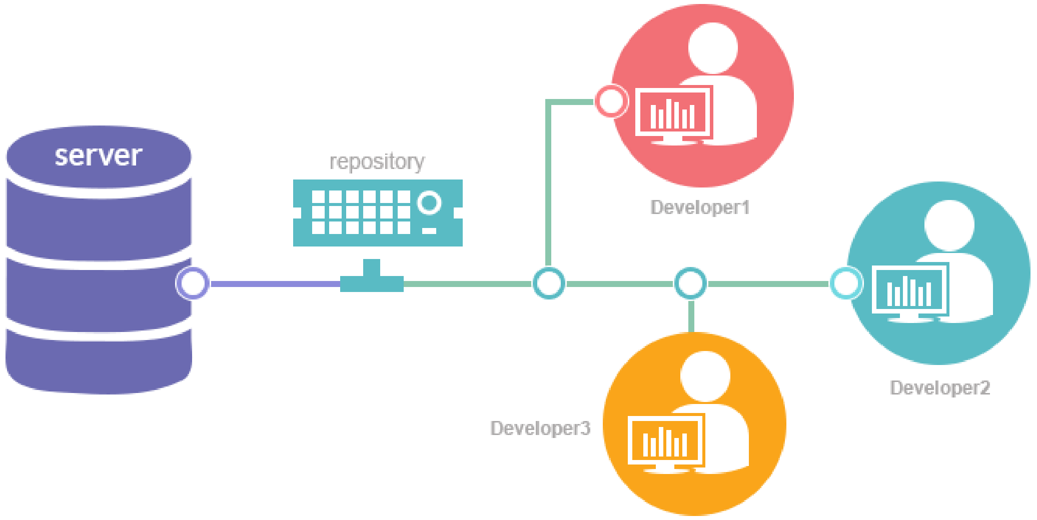


UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Control the Process Automatically

Control the Process Automatically

Manage these things using a Version Control System (VCS)



Control the Process Automatically

Manage these things using a Version Control System (VCS)

Version control is a system that records changes to a file or set of files over time

Why Version Control?

Why Version Control?

Working for your own…

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

Not in a team…

Acts as a "time machine" for going back to earlier versions

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

Not in a team…

Acts as a "time machine" for going back to earlier versions

Keeps the whole history of every file and a changelog

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

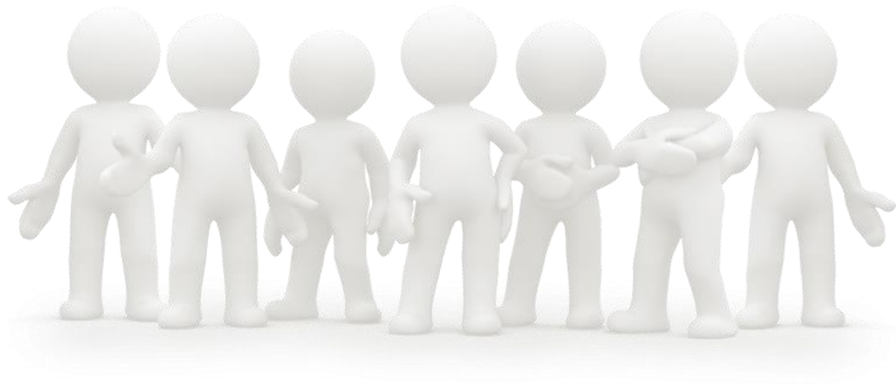
As part of a team…

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

As part of a team…

Greatly simplifies concurrent work, merging changes



UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

Other uses…

Why Version Control?

Other uses…

Helps you find an internship or job!

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

Other uses…

Helps you find an internship or job!

Can manage files when working across multiple computers

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Version Control?

Other uses…

Helps you find an internship or job!

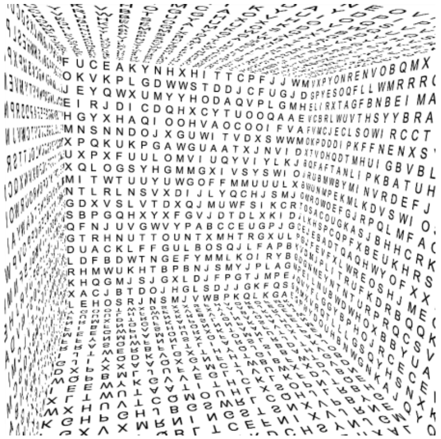
Can manage files when working across multiple computers

But there are better alternatives nowadays…

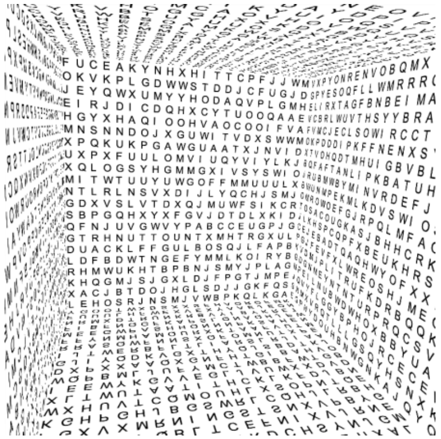
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Nothing's Perfekt

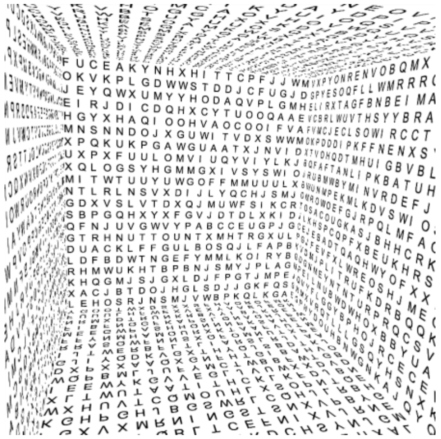
Nothing's Perfekt

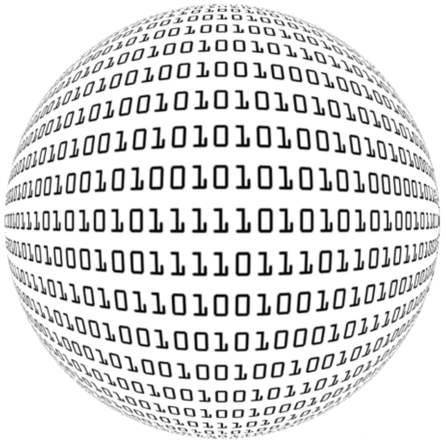


Nothing's Perfekt

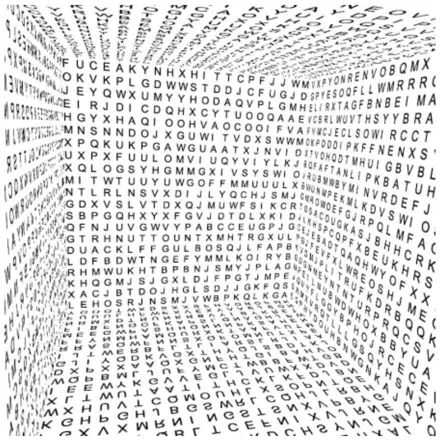


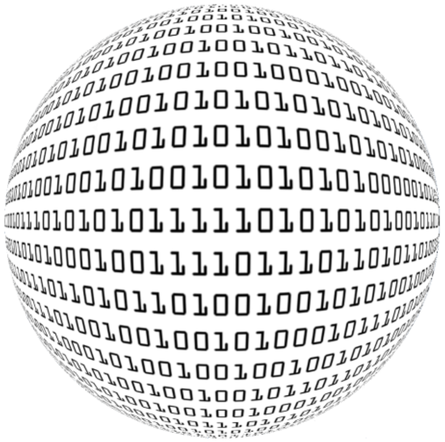
Nothing's Perfekt





Nothing's Perfekt





How it Works

How it Works

Files are kept in a repository

How it Works

Files are kept in a repository

Repositories can be local or remote to the user

How it Works

Files are kept in a repository

Repositories can be local or remote to the user

The user edits a copy called the working copy

How it Works

Files are kept in a repository

Repositories can be local or remote to the user

The user edits a copy called the working copy

Changes are committed to the repository when the user is finished making changes

How it Works

Files are kept in a repository

Repositories can be local or remote to the user

The user edits a copy called the working copy

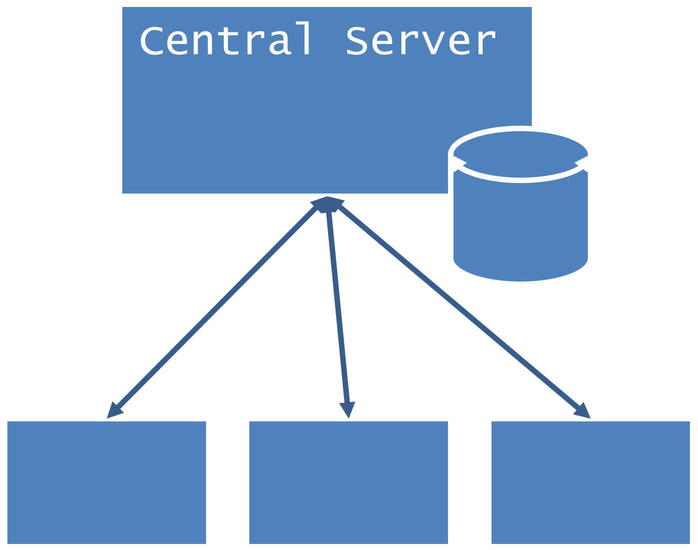
Changes are committed to the repository when the user is finished making changes

Other people can then access the repository to get the new code

Two Major Types of VCS

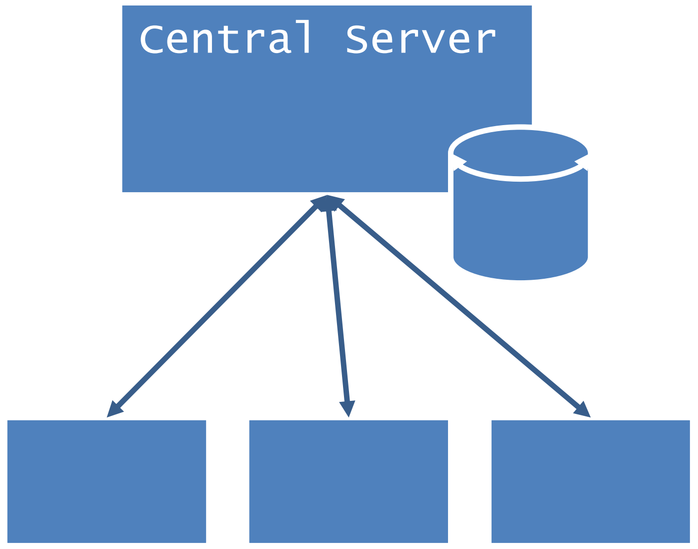
Two Major Types of VCS

Centralized

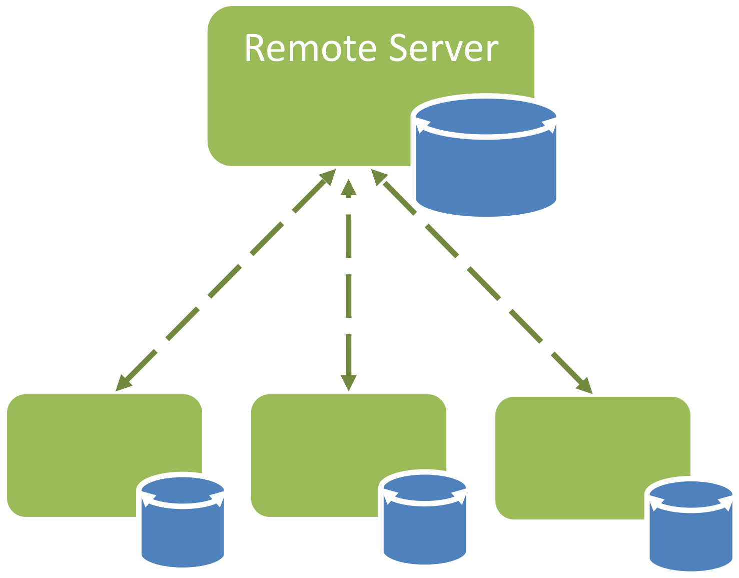


Two Major Types of VCS

Centralized



Distributed



Centralized Version Control

Centralized Version Control

A single server holds the code base

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs

Easier to maintain a single server

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs

Easier to maintain a single server

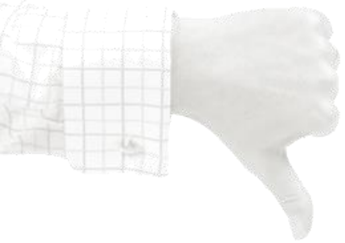
Single point of failure

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs



Easier to maintain a single server

Single point of failure

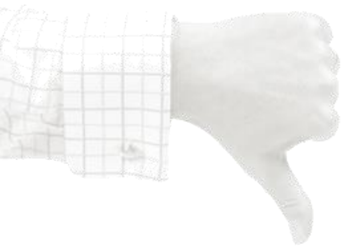
CVS

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs



Easier to maintain a single server

Single point of failure

CVS

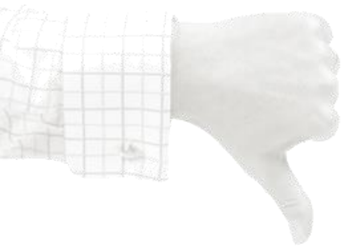
SVN

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Centralized Version Control

A single server holds the code base

Clients access the server by means of check-in / check-outs



Easier to maintain a single server

Single point of failure

CVS

SVN

Visual Source Safe

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed Version Control

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls

Many operations are cheaper

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls

Many operations are cheaper

No single point of failure

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls

Many operations are cheaper

No single point of failure

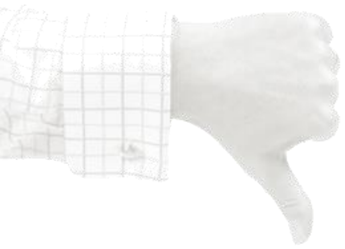
A bit more complicated!

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls



Many operations are cheaper

No single point of failure

A bit more complicated!

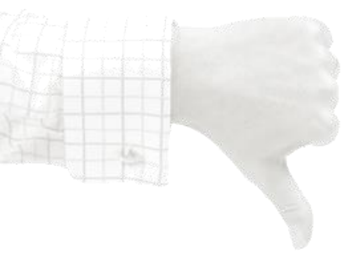
Git

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed Version Control

Each client (essentially) holds a complete copy of the code base

Code is shared between clients by push/pulls



Many operations are cheaper

No single point of failure

A bit more complicated!

Git

Mercurial

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

FAQTS – the Game

Blackboard

- Frequently Asked Questions with Tiny Sentences
- Both Q and A with least possible words
- The ideal word count for answers is *two*
- Our second round:

What is ERROR ?

Git 101

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Why Git?

Why Git?

Many advantages over earlier systems such as CVS and Subversion

Why Git?

Many advantages over earlier systems such as CVS and Subversion

More efficient, better workflow, etc.

Why Git?

Many advantages over earlier systems such as CVS and Subversion

More efficient, better workflow, etc.

Arguably the most popular version control system today

Why Git?

Many advantages over earlier systems such as CVS and Subversion

More efficient, better workflow, etc.

Arguably the most popular version control system today

Best competitor: Mercurial

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Git is a

Git is a

Distributed

Version Control System

Distributed

Everyone has the complete history

Distributed

Everyone has the complete history

Everything can be done offline

Distributed

Everyone has the complete history

Everything can be done offline

…except `push/pull`

Distributed

Everyone has the complete history

Everything can be done offline

No central authority

Distributed

Everyone has the complete history

Everything can be done offline

No central authority

…except by convention

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Distributed

Everyone has the complete history

Everything can be done offline

No central authority

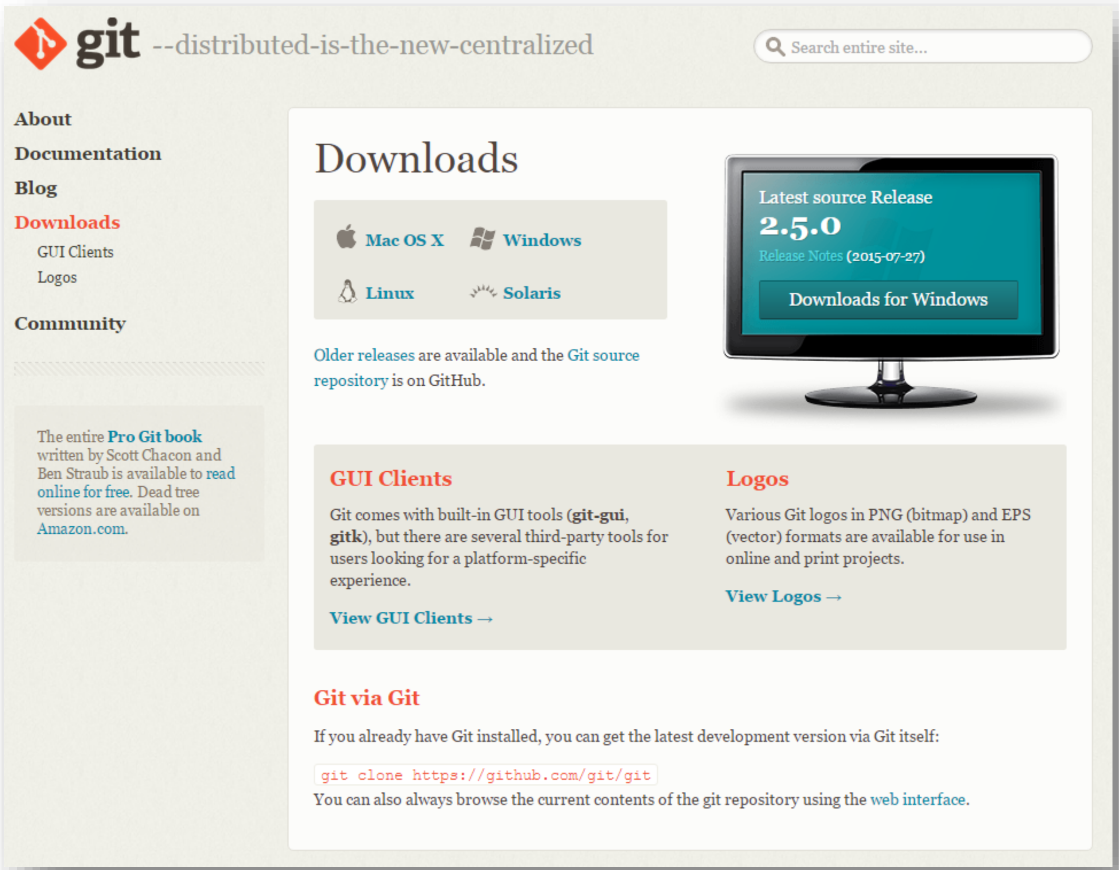
Changes can be shared without a server

Installing Git

Installing Git

- Windows
  - Download ->

http://git-scm.com/downloads



git --distributed-is-the-new-centralized

Search entire site...

About
Documentation
Blog
Downloads
GUI Clients
Logos
Community

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Downloads

Mac OS X Windows
Linux Solaris

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.5.0
Release Notes (2015-07-27)
Downloads for Windows

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

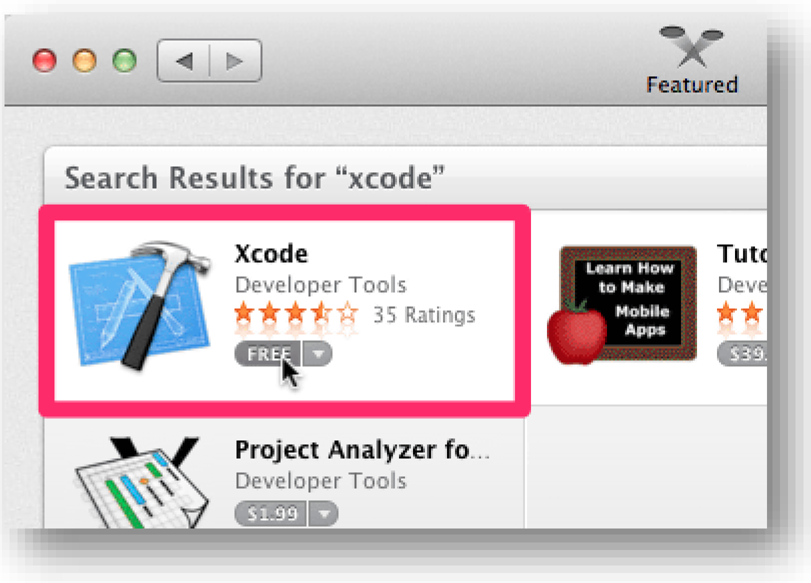
```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.

Installing Git

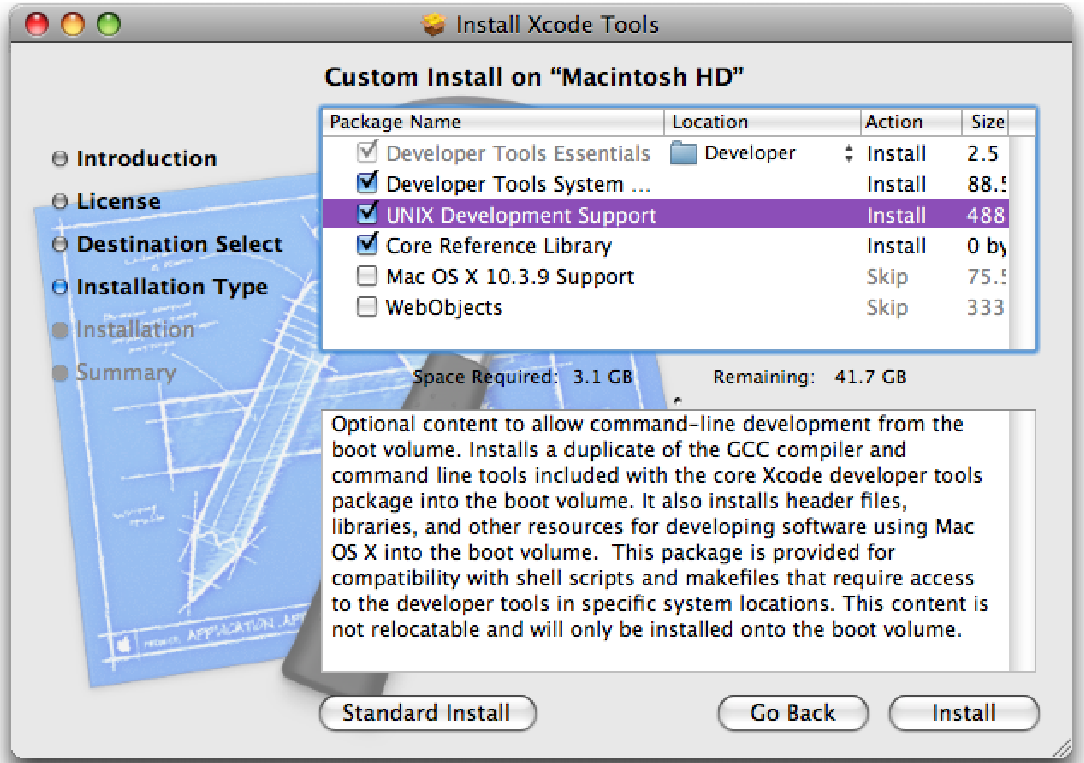
- Windows
  - Download

- Mac OS
  - Install Xcode ->



Installing Git

- Windows
  - Download
- Mac OS
  - Install Xcode ->



Install Xcode Tools

Custom Install on "Macintosh HD"

Package Name	Location	Action	Size
✓ Developer Tools Essentials	Developer	Install	2.5
✓ Developer Tools System ...		Install	88.
✓ UNIX Development Support		Install	488
✓ Core Reference Library		Install	0 by
Mac OS X 10.3.9 Support		Skip	75.
WebObjects		Skip	333

Space Required: 3.1 GB Remaining: 41.7 GB

Optional content to allow command-line development from the boot volume. Installs a duplicate of the GCC compiler and command line tools included with the core Xcode developer tools package into the boot volume. It also installs header files, libraries, and other resources for developing software using Mac OS X into the boot volume. This package is provided for compatibility with shell scripts and makefiles that require access to the developer tools in specific system locations. This content is not relocatable and will only be installed onto the boot volume.

Installing Git

- Windows
  - Download
- Mac OS
  - Install Xcode
- Linux
  - Use standard package manager

```
$ _
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Installing Git

- Windows
  - Download
- Mac OS
  - Install Xcode
- Linux
  - Use standard package manager

```
$ sudo apt-get update
$ _
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Installing Git

- Windows
  - Download
- Mac OS
  - Install Xcode
- Linux
  - Use standard package manager

```
$ sudo apt-get update
$ sudo apt-get install git
$ _
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Set Up Git – Introduce Yourself

```
$ _
```

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name“
$ _
```

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email‟
$ _
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ _
```

you only need to do this once…

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ _
```

omit to use a different name/email address for a particular project

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email‟
$ git config --list
```

Set Up Git – Introduce Yourself

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email‟
$ git config --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizelimit=2g
user.name=Your Name
user.email=Your Email
```

Set Up Git – Choose an Editor

```
$ _
```

Set Up Git – Choose an Editor

```
$ git config --global core.editor /usr/bin/nano
$ _
```

Set Up Git – Choose an Editor

```
$ git config --global core.editor /usr/bin/nano
$ _
```

the default is `vim`

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Set Up Git – Choose an Editor

```
$ git config --global core.editor /usr/bin/nano
$ git config --global color.ui auto
$ _
```

Set Up Git – Choose an Editor

```
$ git config --global core.editor /usr/bin/nano
$ git config --global color.ui auto
$ _
```

to turn on colors

Create and Fill a Repository

```
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ _
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
$ _
```

← this creates the repository which is a directory named *.git*

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
$ _
```

this creates the repository which is a directory named *.git*

you seldom (if ever) need to look inside this directory

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
$ _
```

this creates the repository which is a directory named *.git*
you seldom (if ever) need to look inside this directory

you do not work directly with the contents of *.git*; various git commands do that for you

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ _
```

this adds all your current files to the repository

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ _
```

this adds all your current files to the repository

if you create new files and/or folders, they are not tracked by Git unless you ask it to do so

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ _
```

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ _
```

committing makes a “snapshot” of everything being tracked into your repository

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ _
```

committing makes a "snapshot" of everything being tracked into your repository

commits are cheap. do them often.

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ _
```

you must provide a one-line message stating what you have done

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ git status
$ _
```

see what Git thinks is going on

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ git status
$ _
```

see what Git thinks is going on

use this frequently!

Create and Fill a Repository

```
$ cd Desktop
$ mkdir csci-2000
$ mkdir csci-2000/Assignments
$ mkdir csci-2000/Assignments/Assignment-1
$ nano csci-2000/Assignments/Assignment-1/sample.commands.txt
$ cd csci-2000
$ git init
Initialized empty Git repository in /home/vlad/csci-2000
$ git add .
$ git commit -a -m "Initial commit"
$ git status
$ git log
```

shows the log

Checkpoint 7

- Registering with GitHub :
  - Blackboard > Course Content > Week 3 (Sept. 28- Oct. 2) > Wednesday Sept. 30 > Checkpoint 7

QOTD

- "Imitation is the sincerest form of flattery"

Charles Colton

(1780–1832)

Cleric, writer and collector