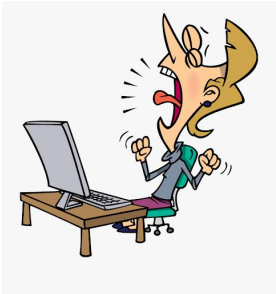
Introduction to git

Shahab Fatemi (shahab.fatemi@umu.se)
Department of Physics, Umeå University, Umeå, Sweden

2020-12-18, Umeå

Introduction

- My code was running just fine yesterday, but not today! What did I change?



- I would like to keep track of my code (thesis) development and see how it has evolved over time.

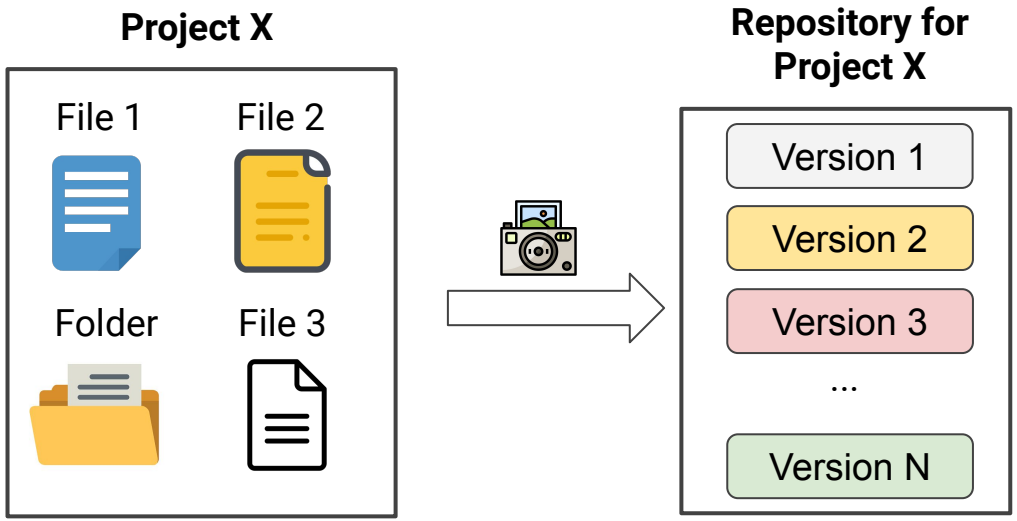
- We are working within a team and we like to work together on the same piece of code.

You should use a Version Control System

What do we learn today?

- Introduction to Version Control Systems
- We learn about git and its web-based tools
- We learn how to use git and its essential commands by working on an example

Version Control System (VCS)

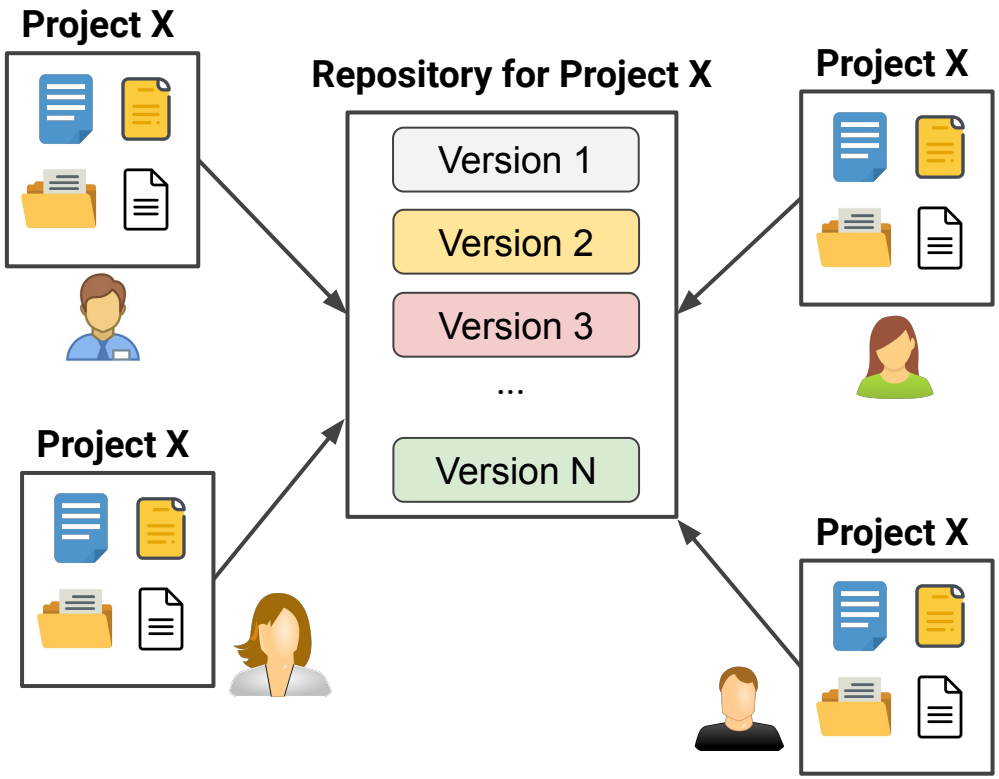


- VCS tracks the history of changes (it takes a snapshot)
- History of changes can be viewed
- Any version can be recovered at any time
- VCS are software tools that help developers to track and manage changes to their source code over time [ref: www.atlassian.com]

Version Control System (VCS) -> Part II

Developers can find out about

- **Which** changes were made
- **Who** made the changes
- **When** were the changes made
- **Why** were changes needed



An example!

Imagine that you are working on a Python-based project and the example below is one of the functions you have developed last year.

```
def verify(self):
    if self.title not in self.Titles:
        raise ValueError("%s is not a valid title." % self.title)

    if( self.age() > 130 ):
        raise ValueError("Person's age is higher than pre-defined value.")

    # pass the regular expression
    # and the string in search() method
    if(re.search(self.RegEx, self.email) == False):
        raise ValueError("Invalid email")
```

You know VCS and you are using it for your project.

An example!

By early this year, you found out that you need to work more on your function and you made some changes in it. By the end of the year, you may wonder which changes you made in your code.

An example of log report you can take from git(hub).

Old version

```
44          if self.title not in self.Titles:
45              raise ValueError("%s is not a valid title." % self.title)
46
-           if( self.age() > 130 ):
-               raise ValueError("Person's age is higher than pre-defined value.")
47  +       if( self.age() > 130 and
48  +           self.age() < 0 ):
49  +           raise ValueError("Person's age (%d) is not defined correctly."%(self.age()) )
50
51          # pass the regular expression
52          # and the string in search() method
```

New version

Version Control System Tools

Many tools are available.

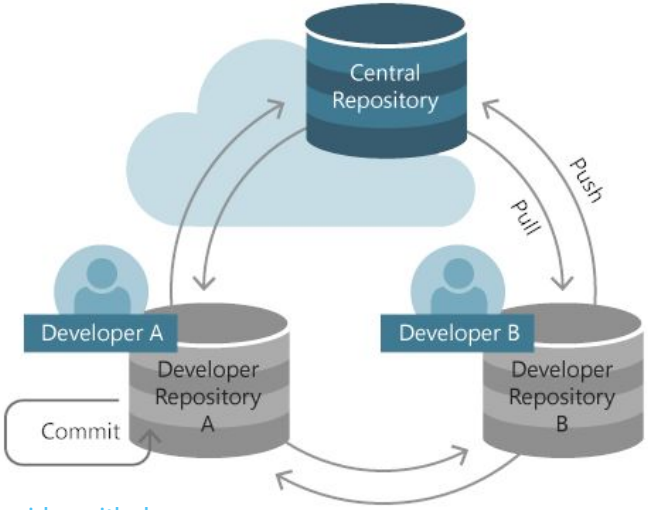
- Subversion is an open source VCS founded in 2000
- It was incredibly successful by then
- It is a centralized VCS (uses a central server to store files)

- Git is an open source VCS founded in 2005
- It is a distributed VCS (uses a central server + takes a local copy of everything)
- It works asynchronously from any time zone

git

- **Git** is the most-used VCS in the world (according to Developer Survey Results 2018)
- It allows viewing the entire timeline of the project

Some fundamental concepts:



www.guides.github.com

- A **"repository"** keeps the entire files and folders associated with a project, along with each file's revision history.
- **local repository** vs. **central repository**
- The developers **"pull"** the latest changes from the central repository into their local repository.
- And **"push"** all the codes from their local repository into the central repository.
- A **"branch"** represents an independent line of development, and is similar to the branch of a tree.

Where to begin?

- You need to install “git” on your machine (supports Windows, Linux, & Mac OS) https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

- You can use git and work on your local repository

- There are also plenty of web-based version control repository hosting services that support git. The most well-knowns & popular hosting services are

Atlassian Bitbucket
www.bitbucket.org

GitHub
www.github.com

GitLab
www.gitlab.com

- You can run your own hosting service on your own server as well

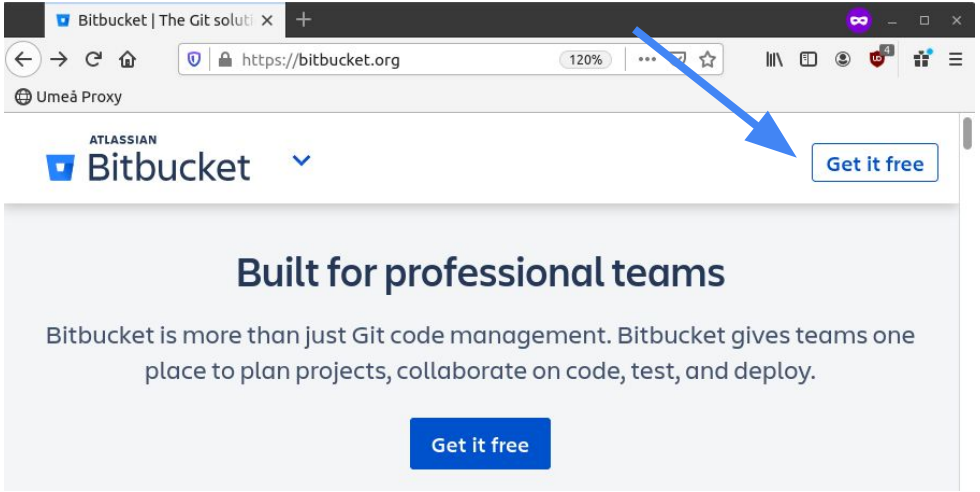
Let's get started

- Make sure git is installed

  (The latest git version is 2.29.2)

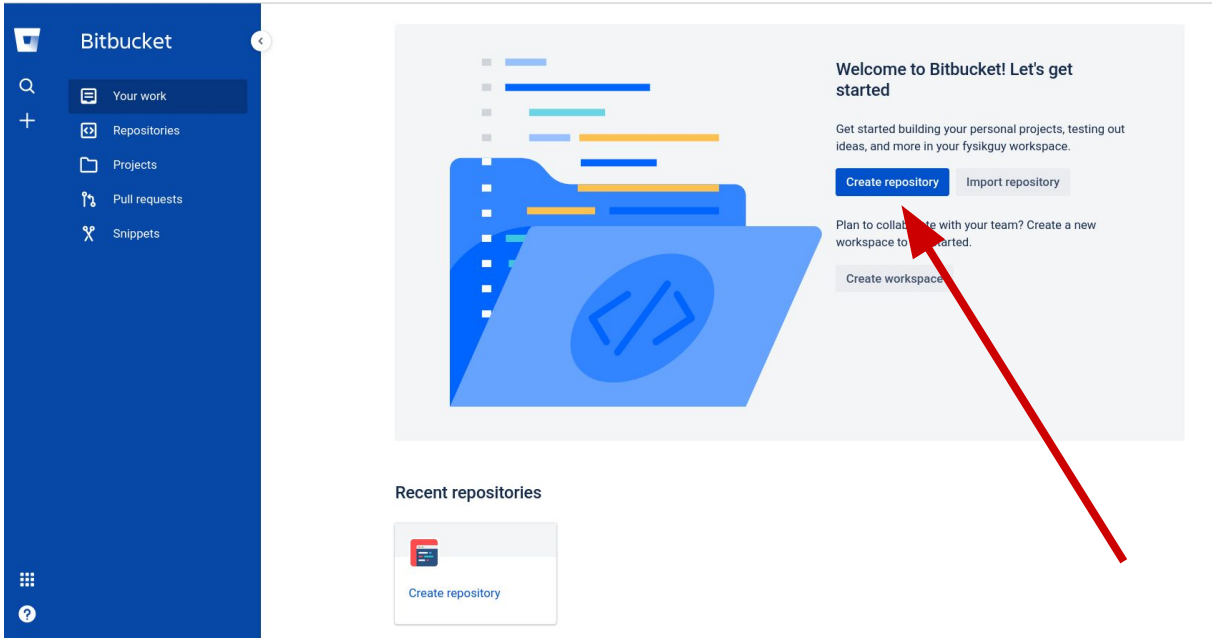
```
(base) shahab@desktop:~$ git --version
git version 2.29.2
(base) shahab@desktop:~$
```

- Customize your git environment (e.g., your name, email, etc): https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
- Get a free account on your favorite version control hosting service



Login to Bitbucket (www.bitbucket.org)

- You're ready to set up your first repository.
- The repository is remote (central).



Create your remote (central) repository

Create a new repository

Import repository

Workspace

Project name* projectx

Repository name* testrepo

Access level ☑ Private repository

Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.

Include a README? Yes, with a tutorial (for beginne...

Include .gitignore? Yes (recommended)

› Advanced settings

Create repository Cancel

testrepo

- Source
- Commits
- Branches
- Pull requests
- Pipelines
- Jira issues
- Downloads
- Repository settings

Shahab Fatemi / projectx

testrepo

Clone •••

Take the next steps for this new repository and its freshly added files

Copy and connect the repository locally so that you can push updates you make and pull changes others make. Enter **git clone** and the repository URL at your command line:

```
git clone https://fysikguy@bitbucket.org/fysikguy/testrepo.git
```

Learn more or clone in Sourcetree to avoid the command line. Sourcetree is a free Git client.

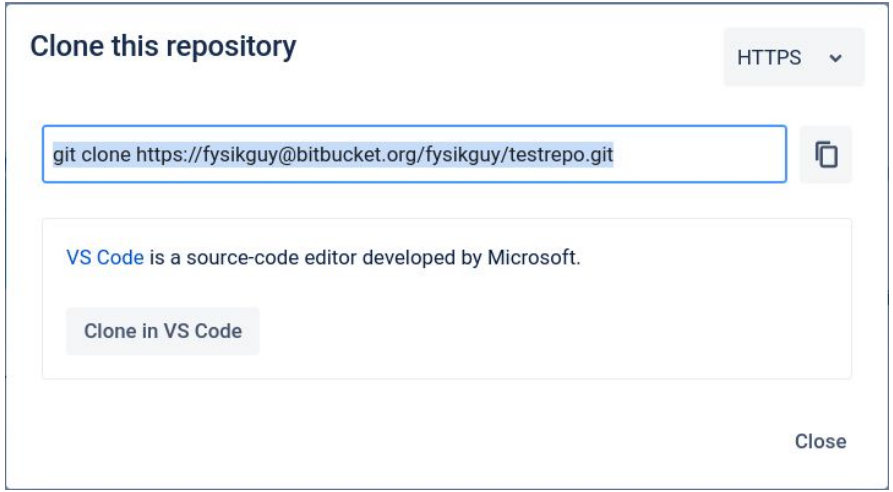
Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.

master | Files | Filter files

/

Name	Size	Last commit	Message
.gitignore	624 B	14 seconds ago	Initial commit
README.md	2.56 KB	14 seconds ago	Initial commit

Clone your project




```
(base) shahab@desktop:~/Documents$ git clone https://fysikguy@bitbucket.org/fysikguy/testrepo.git
Cloning into 'testrepo'...
Password for 'https://fysikguy@bitbucket.org': 
```

Git “clone” creates a local copy of a project that already exists remotely.

Local repository

Now your local repository is all set up and you are ready to work on the local version of your project (the local repo. is also linked to the remote repo.)

```
shahab@desktop: ~/Documents/testrepo
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$ git clone https://fysikguy@bitbucket.org/fysikguy/testrepo.git
Cloning into 'testrepo'...
Password for 'https://fysikguy@bitbucket.org':
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), 1.71 KiB | 875.00 KiB/s, done.
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$
(base) shahab@desktop:~/Documents$ cd testrepo/
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$
```

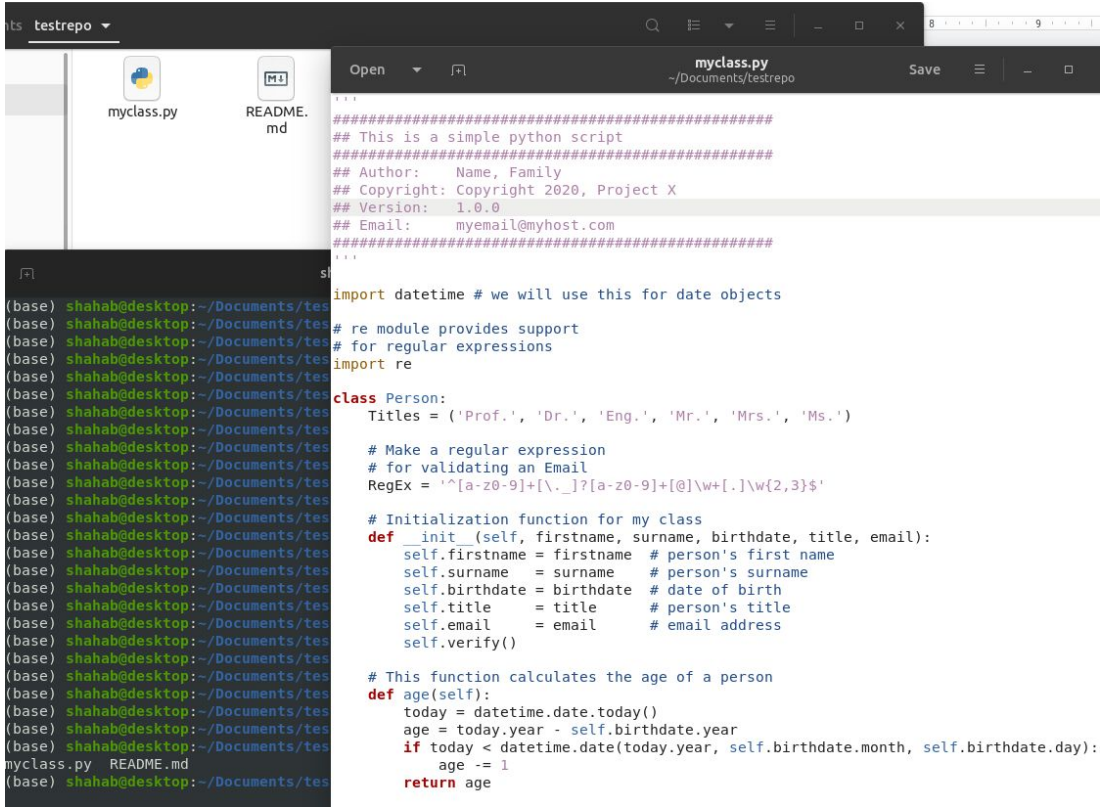
My project

Moved to your project directory:
~/Documents/testrepo

Work on your program in that directory.

My sample program is in Python and I have made only one file. You can test it with your favorite programming language:
C, C++, Matlab, Java, Fortran, etc.

At this moment, I am happy with my development and now is the time to take a snapshot of my development on my local machine and then store it on my remote repository.



Adding and Committing to your git repository

Here is a chain of commands we need to execute one after another:

- **git status**

  This command will show you what files have not been added or changed in your local repository.

- **git add .**

  Note, there is a dot at the end of this command.

  by adding a . at the end of the command, you're telling git to include everything in the directory. If you want to add a single file, just use it's filename. For example, *myclass.py*

- **git commit -m "short explanation of what you have done"**

  When you commit these files, you should also leave a quick message to let your teammates (or yourself in future) know exactly what was in that commit. Standard git conventions say that you should be as concise and specific as possible, and start the message using the present tense of the verb.

Adding and Committing to your git repository

```
(base) shahab@desktop:~/Documents/testrepo$ ls
myclass.py  README.md
(base) shahab@desktop:~/Documents/testrepo$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$ git add .
(base) shahab@desktop:~/Documents/testrepo$ git commit -m "Version 1"
[master 6c48936] Version 1
 1 file changed, 66 insertions(+)
 create mode 100644 myclass.py
(base) shahab@desktop:~/Documents/testrepo$
```

- Now you have taken a snapshot of your project locally.
- You should then add it to your remote repository, if you want.

Push to the remote repository

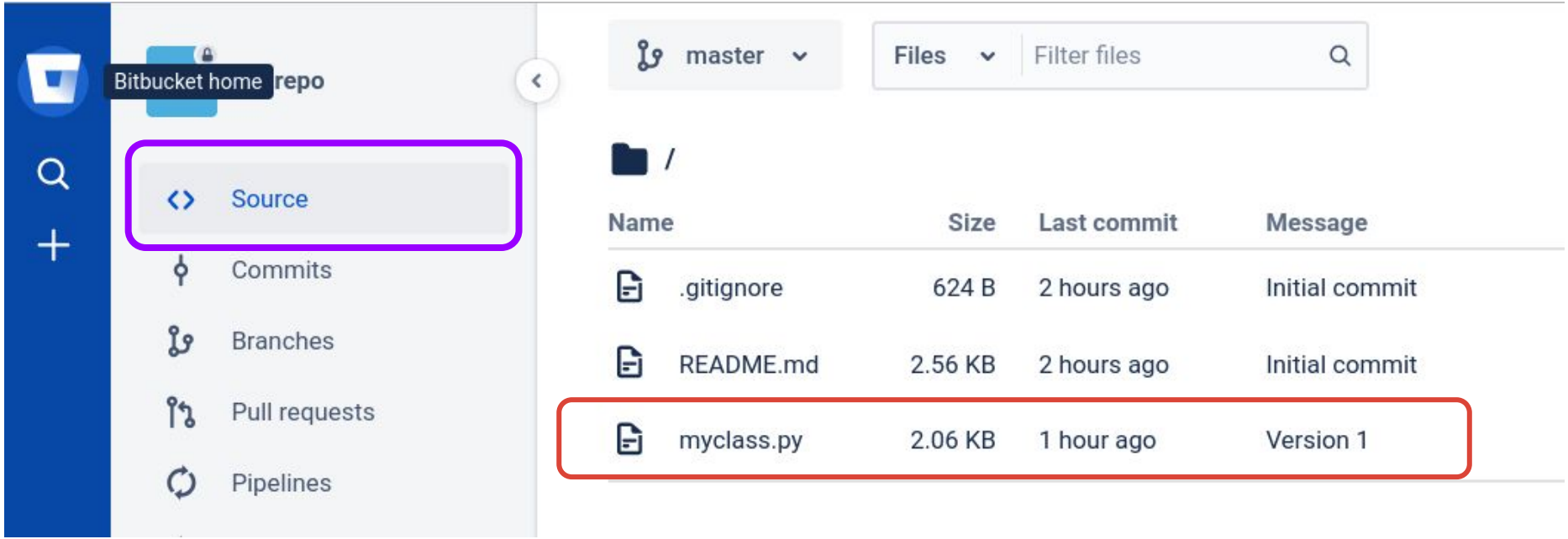
The main command is:

- **git push origin master**

  This command uploads the content of the local repository to a remote repository.

```
(base) shahab@desktop:~/Documents/testrepo$ git push origin master
Password for 'https://fysikguy@bitbucket.org':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.16 KiB | 1.16 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://bitbucket.org/fysikguy/testrepo.git
   fab9980..6c48936  master -> master
```

Remote repository (source)



Remote repository (commits)

Bitbucket home

repo

Source

Commits

Branches

Pull requests

Pipelines

Jira issues

Downloads

Repository settings

Shahab Fatemi / projectx / testrepo

Commits

Clone

Search commits

All branches

Author	Commit	Message	Date
SF Shahab Fatemi	6c48936	Version 1	1 hour ago
SF Shahab Fatemi	fab9980	Initial commit	2 hours ago

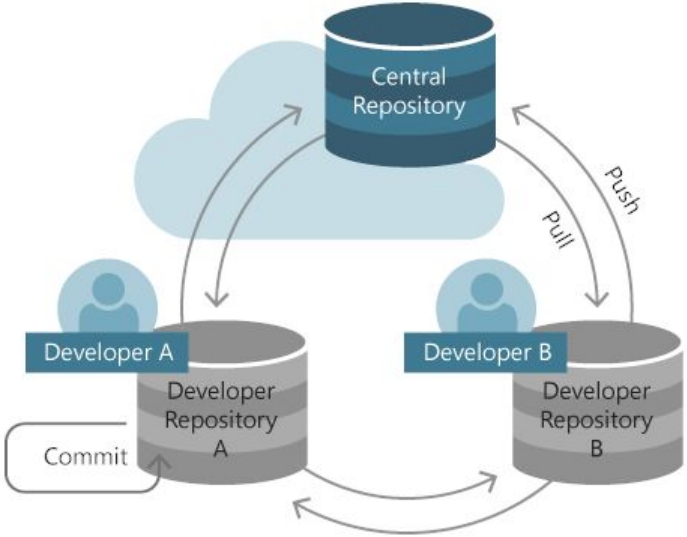
Pull from the remote repository

The main command is:

- **git pull origin master**

  This command downloads the content from a remote repository and immediately updates the local repository.

If the repository is shared, you need to “pull” the latest changes from your remote repo. to your local repo. before making any changes to your program/code.



Version 2 with some updates

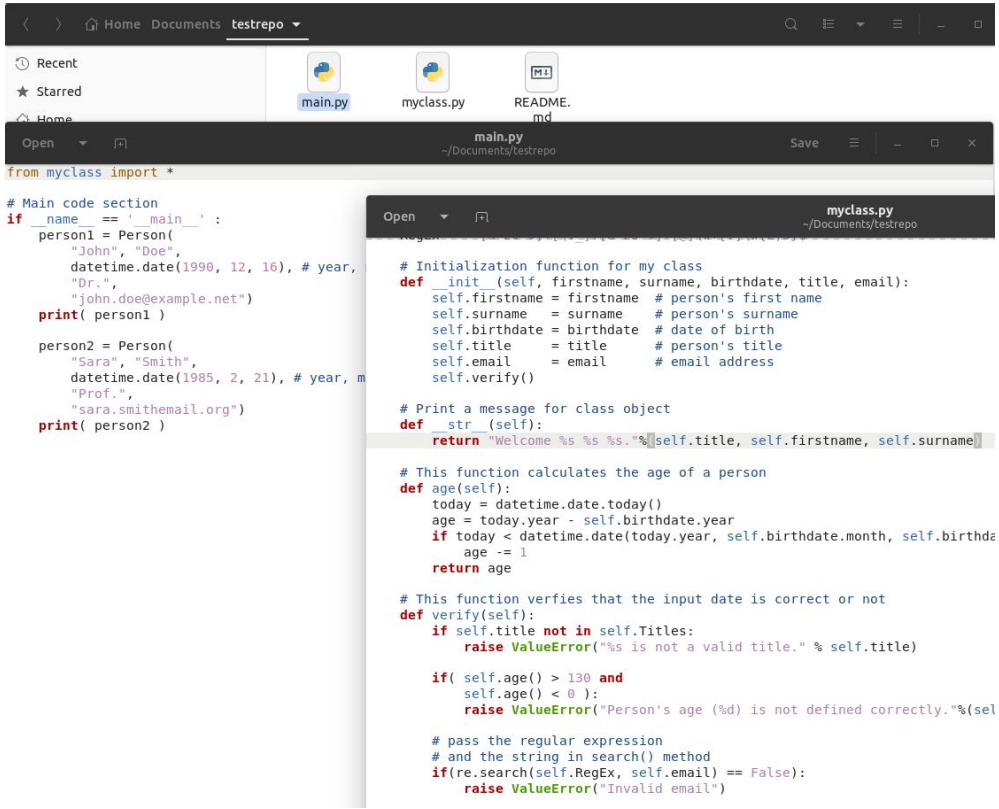
Me, or one of my teammates, made some changes into our project (updated *myclass.py* and added *main.py* to my project).

The changes are on the **local machine**.

The changes are neither “pushed” to the local repository nor to the remote repository.

What should we do now after making some changes?

- We “add”,
- Then “commit”,
- And finally “push”



“Push” it!

```
(base) shahab@desktop:~/Documents/testrepo$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

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

no changes added to commit (use "git add" and/or "git commit -a")
(base) shahab@desktop:~/Documents/testrepo$ git add .
(base) shahab@desktop:~/Documents/testrepo$ git commit -m "Version 2"
[master 80bd47a] Version 2
 2 files changed, 24 insertions(+), 15 deletions(-)
 create mode 100644 main.py
(base) shahab@desktop:~/Documents/testrepo$ git push origin master
Password for 'https://fysikguy@bitbucket.org':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 705 bytes | 705.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
To https://bitbucket.org/fysikguy/testrepo.git
   6c48936..80bd47a  master -> master
(base) shahab@desktop:~/Documents/testrepo$
```

Commits on remote repository

testrepo

Source
Commits
Branches
Pull requests
Pipelines
Jira issues
Downloads
Repository settings

Shahab Fatemi / projectx / testrepo

Commits

Clone

Search commits

All branches

Author	Commit	Message	Date
SF Shahab Fatemi	80bd47a	Version 2	23 minutes ago
SF Shahab Fatemi	6c48936	Version 1	2 hours ago
SF Shahab Fatemi	fab9980	Initial commit	3 hours ago

List all commits

```
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$ git log --pretty=format:"%h %s" --graph
* 80bd47a Version 2
* 6c48936 Version 1
* fab9980 Initial commit
(base) shahab@desktop:~/Documents/testrepo$
```

Note!

You can have many local commits that are not pushed to the remote repository. Using the command above, you can list all the **local commits**.

Differences between two commits on remote repo.

testrepo

- Source
- Commits
- Branches
- Pull requests
- Pipelines
- Jira issues
- Downloads
- Repository settings

myclass.py MODIFIED

Side-by-side diff | View file | Comment

```diff
31 31         self.email     = email      # email address
32 32         self.verify()
33 33
   34 +    # Print a message for class object
   35 +    def __str__(self):
   36 +        return "Welcome %s %s %s."%(self.title, self.firstname, self.surname)
   37 +
34 38     # This function calculates the age of a person
35 39     def age(self):
36 40         today = datetime.date.today()
...
44 48         if self.title not in self.Titles:
45 49             raise ValueError("%s is not a valid title." % self.title)
46 50
47    -        if( self.age() > 130 ):
48    -            raise ValueError("Person's age is higher than pre-defined value.")
   51 +        if( self.age() > 130 and
   52 +            self.age() < 0 ):
   53 +            raise ValueError("Person's age (%d) is not defined correctly."%(self.age()) )
49 54
50 55         # pass the regular expression
51 56         # and the string in search() method
52 57         if(re.search(self.RegEx, self.email) == False):
53 58             raise ValueError("Invalid email")
54    -
55    -# Main code section
56    -if __name__ == '__main__' :
57    -    # A sample object
58    -    person1 = Person(
59    -        "John", "Doe",
60    -        datetime.date(1990, 12, 16), # year, month, day
61    -        "Dr.",
62    -        "john.doe@example.com")
63    -
64    -    # prinst the age of the sample object
65    -    print( person1.age() )
```

Differences between two commits on local repo.

- **git diff commit_id1 commit_id2**

Version1 **Version2**

```
(base) shahab@desktop:~/Documents/testrepo$ git diff 6c48936 80bd47a
```

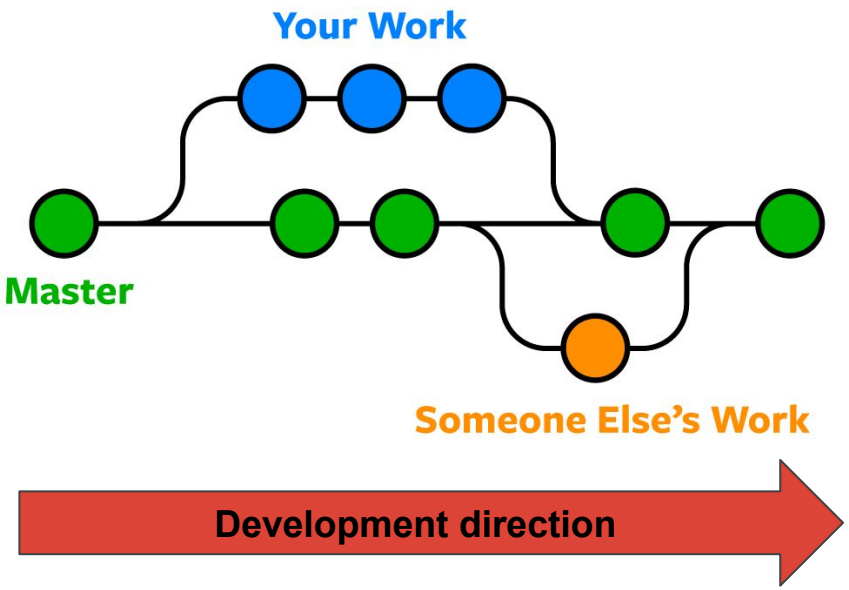
```
diff --git a/myclass.py b/myclass.py
index 26ac8ab..3923962 100644
--- a/myclass.py
+++ b/myclass.py
@@ -31,6 +31,10 @@ class Person:
         self.email     = email      # email address
         self.verify()

+    # Print a message for class object
+    def __str__(self):
+        return "Welcome %s %s %s."%(self.title, self.firstname, self.surname)
+
     # This function calculates the age of a person
     def age(self):
         today = datetime.date.today()
@@ -44,23 +48,11 @@ class Person:
         if self.title not in self.Titles:
             raise ValueError("%s is not a valid title." % self.title)

-        if( self.age() > 130 ):
-            raise ValueError("Person's age is higher than pre-defined value.")
+        if( self.age() > 130 and
+            self.age() < 0 ):
+            raise ValueError("Person's age (%d) is not defined correctly."%(self.age()) )
```

Branching

A **“branch”** represents an independent line of development, and is similar to the branch of a tree.

You can take a branch of a project, work on that branch separately from the main (“master”) branch.

Working on a branch keeps the main codebase safe, while allows you to take snapshots of your changes on your own branch.



Each node indicates a “commit”.

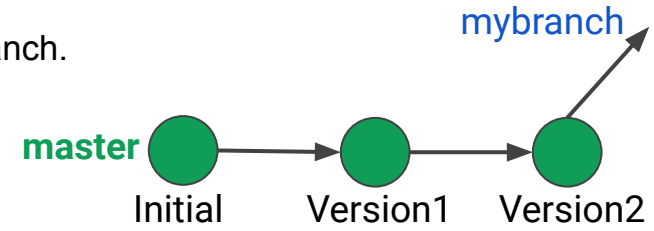
Branching

- **git branch "branch_name"**

  This command will create a branch from your master branch.

- **git checkout branch_name**

  This command will move you to "mybranch".



- **git status**

  It verifies in which branch you are

- **git branch**

  Lists all branches

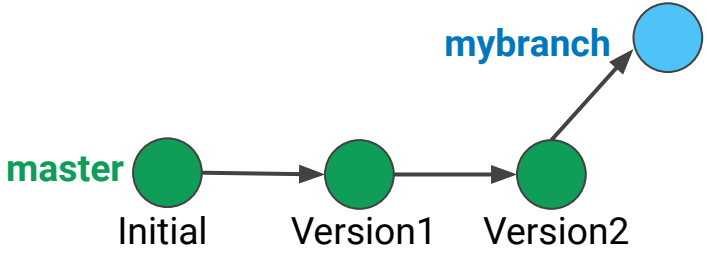
```
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$ git branch mybranch
(base) shahab@desktop:~/Documents/testrepo$ git checkout mybranch
Switched to branch 'mybranch'
(base) shahab@desktop:~/Documents/testrepo$ git status
On branch mybranch
nothing to commit, working tree clean
(base) shahab@desktop:~/Documents/testrepo$ git branch
  master
* mybranch
(base) shahab@desktop:~/Documents/testrepo$
```

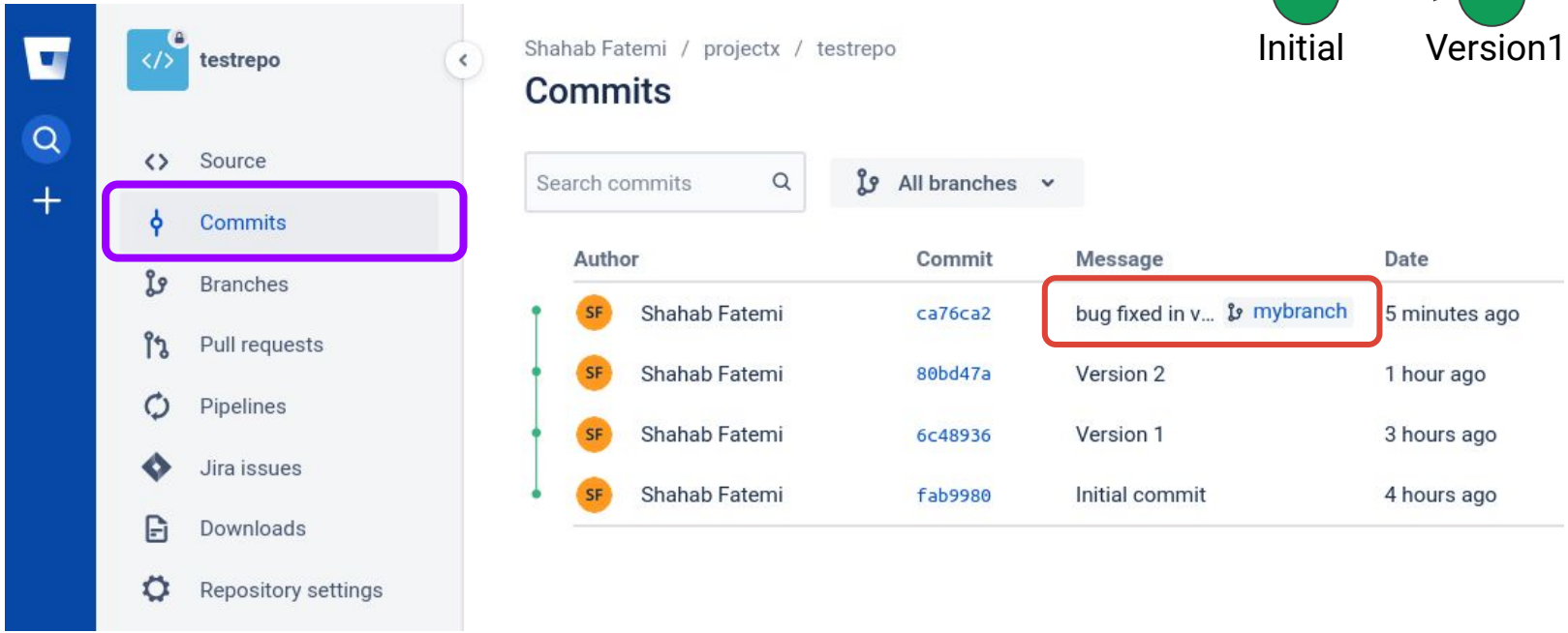
Work on my own branch

I noticed some errors at “myclass.py” and fixed them on “mybranch”. Now, commit them to my local repository as well as on the remote repository.

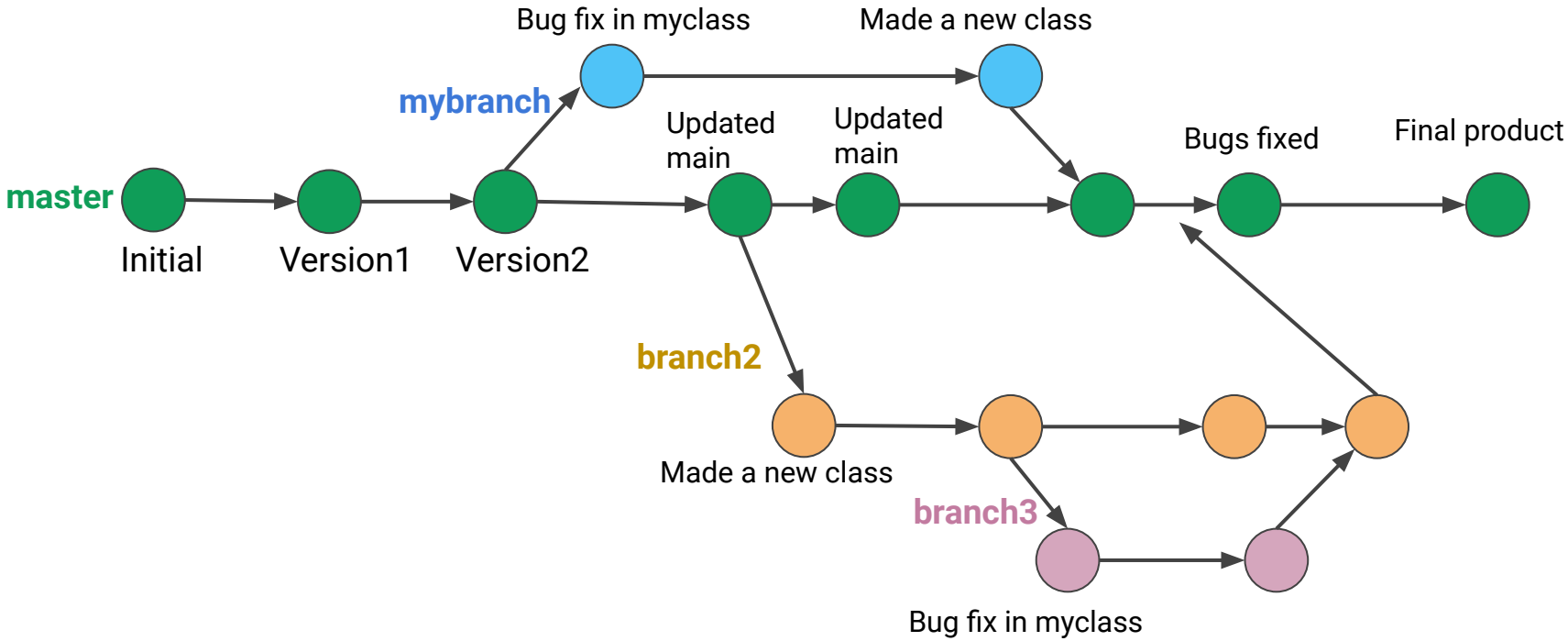
```
(base) shahab@desktop:~/Documents/testrepo$ git status
On branch mybranch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   myclass.py

no changes added to commit (use "git add" and/or "git commit -a")
(base) shahab@desktop:~/Documents/testrepo$ git add .
(base) shahab@desktop:~/Documents/testrepo$ git commit -m "bug fixed in verify function"
[mybranch ca76ca2] bug fixed in verify function
 1 file changed, 5 insertions(+), 3 deletions(-)
(base) shahab@desktop:~/Documents/testrepo$ git push origin mybranch
Password for 'https://fysikguy@bitbucket.org':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 424 bytes | 424.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote:
remote: Create pull request for mybranch:
remote:   https://bitbucket.org/fysikguy/testrepo/pull-requests/new?source=mybranch&t=1
remote:
To https://bitbucket.org/fysikguy/testrepo.git
 * [new branch]      mybranch -> mybranch
(base) shahab@desktop:~/Documents/testrepo$
```

Commits on the remote repository





Branching and Merging



Merging to “master” branch

```
(base) shahab@desktop:~/Documents/testrepo$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
(base) shahab@desktop:~/Documents/testrepo$ git  pull  origin  master
(base) shahab@desktop:~/Documents/testrepo$
(base) shahab@desktop:~/Documents/testrepo$ git merge mybranch
Updating 80bd47a..ca76ca2
Fast-forward
 myclass.py | 8 +++++---
 1 file changed, 5 insertions(+), 3 deletions(-)
(base) shahab@desktop:~/Documents/testrepo$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
(base) shahab@desktop:~/Documents/testrepo$ git push origin master
Password for 'https://fysikguy@bitbucket.org':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://bitbucket.org/fysikguy/testrepo.git
   80bd47a..ca76ca2  master -> master
(base) shahab@desktop:~/Documents/testrepo$
```

Typical questions and misconceptions

- A typical question by students: **Can I put my thesis on git?**
  Yes, but if you use e.g., MS Excel files, you cannot *easily* view the differences.
  If you use a central server, make sure your materials in your repository are not copyrighted and approved by the supervisors.
  When git can't recognise the type of a file, it just treats it as binary data for versioning purposes; so **diff** will just state that the files are different by a number of bytes.
- Another typical question: **How often should we commit, push and pull?**

Misconceptions:

- Git is GitHub **(WRONG!)**
- Backups are equivalent to version control **(WRONG!)**
- Git is only suitable for teams **(WRONG!)**
- To effectively use Git, you need to learn every command to work **(WRONG!)**

Thank you & happy holidays!

Questions?

https://www.pinterest.com

My email:
shahab.fatemi@umu.se

Read more…!

- Git website: https://git-scm.com/
- **Download and installation guide: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git**
- Git handbook: https://guides.github.com/introduction/git-handbook/
- Git cheat sheet: https://training.github.com/downloads/github-git-cheat-sheet/
- Play with https://git-school.github.io/visualizing-git/

Online youtube lectures:

- Short versions
  - https://www.youtube.com/watch?v=SWYqp7iY_Tc
  - https://www.youtube.com/watch?v=N_bMCff8q6A
- Long versions
  - https://www.youtube.com/watch?v=xuB1Id2Wxak
  - https://www.youtube.com/watch?v=8JJ101D3knE

and a lot more…!