

B216A Développement informatique

## **Cours 1 & 2**

# **Gestion de projet, versioning, package et intégration continue**

*Sébastien Combéfis, Quentin Lurkin*



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

# Objectifs

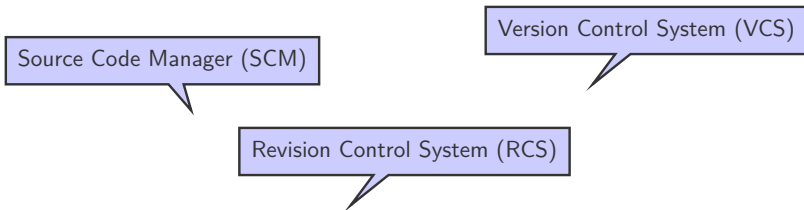
- Gestion du **code source**
  - Versioning de code et exemple avec GitHub
  - Package et gestion des dépendances
- **Test** du code source
  - Unit testing et TDD
  - Intégration continue avec Travis



Versioning de code

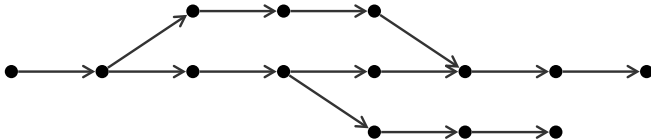
# Source Code Management

- Pour tout projet informatique, il faut une **stratégie de backup**
- On ajoute souvent une **gestion des versions**
- Un développeur peut proposer plusieurs **révisions** par jour



# Buts d'un gestionnaire de versions

- Gestion d'un projet de programmation
- Garder l'**historique** de toutes les modifications
- Travail en **équipe**
- Support de **branches** de développement



# Git

- Système inventé par **Linus Torvalds** pour le kernel Linux

- Git a vu le jour en avril 2005

*Premier commit le 8 avril*





- Logiciel de gestion de versions **décentralisé**

*Connexion internet uniquement pour les pull et push*

**Initial revision of "git", the information manager from hell**

[Browse code](#)

 master  v2.1.2 ... v0.99



Linus Torvalds authored on 8 Apr 2005

0 parents

commit e83c5163316f89bfbd7d9ab23ca2e25604af290

# Pronunciation

[ gít ]

[ jít ]



# Pronunciation

[ gít ]

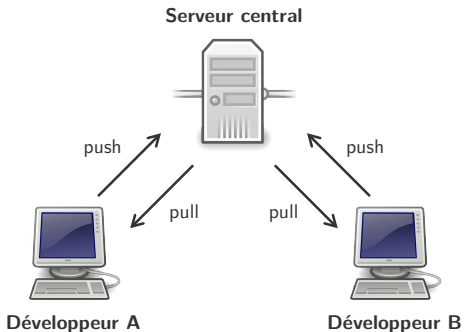


[ jít ]



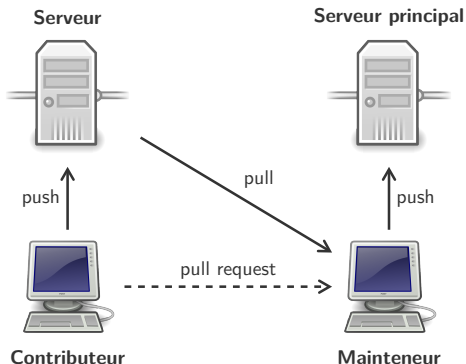
# Git avec un serveur central

- Accès en écriture pour **tous les développeurs**



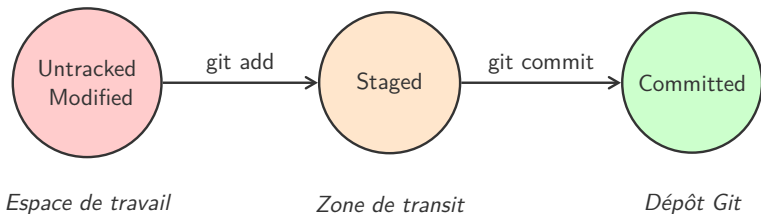
# Git décentralisé

- Accès en écriture seulement pour **les mainteneurs**
- **Les contributeurs** font des *pull requests*



# États des fichiers (1)

- Un fichier doit être **explicitement ajouté** au dépôt Git



# États des fichiers (2)

## ■ **Untracked/Modified**

- Nouveaux fichiers ou fichiers modifiés
- Pas pris en compte pour le prochain commit

## ■ **Staged**

- Fichiers ajoutés, modifiés, supprimés ou déplacés
- Pris en compte pour le prochain commit

## ■ **Unmodified/Committed**

- Aucune modification pour le prochain commit

# Commandes de base

- Ajouter un fichier dans la zone de transit

*git **add** <fichier>*

- Obtenir l'état des fichiers

*git **status***

- Valider les modifications en créant un commit

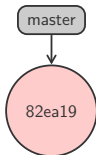
*git **commit** -m "Titre du commit"*

- Obtenir l'historique des commits

*git **log***

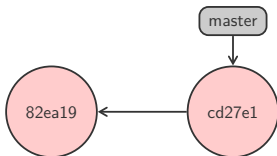
# Le concept de branche

- Une **branche** pointe vers un commit
- À chaque nouveau commit, le **pointeur de branche** avance
- Un commit pointe vers le commit parent



# Le concept de branche

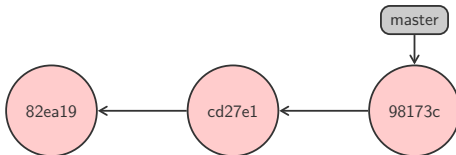
- Une **branche** pointe vers un commit
- À chaque nouveau commit, le **pointeur de branche** avance
- Un commit pointe vers le commit parent





# Le concept de branche

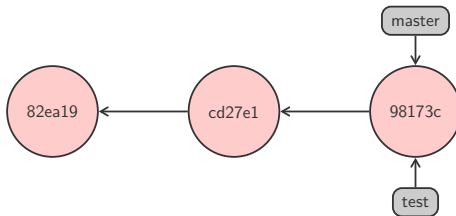
- Une **branche** pointe vers un commit
- À chaque nouveau commit, le **pointeur de branche** avance
- Un commit pointe vers le commit parent



# Création d'une nouvelle branche

- Une nouvelle **branche** est créée avec « **git branch** <name> »

```
1 $ git branch test
```

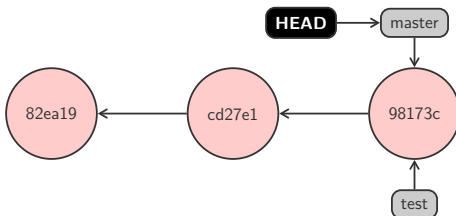


# Branche courante

- La commande « **git branch** » liste les branches existantes

```
1 $ git branch
2 * master
3   test
```

- La **branche courante** est identifiée par **HEAD**

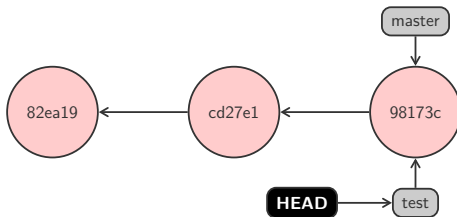


# Changer de branche

- La commande « **git checkout** <name> » **change de branche**

```
1 $ git checkout test
2 Switched to branch 'test'
```

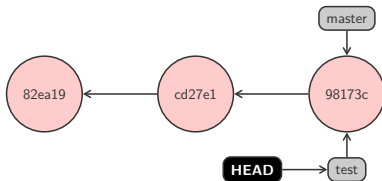
- La branche courante est identifiée par **HEAD**



# Commit sur une branche

- Un commit va toujours se faire sur la branche courante

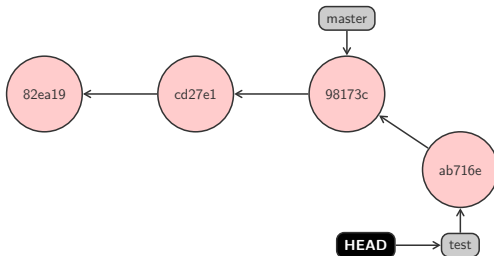
```
1 ...  
2 $ git commit ...  
3 $ git checkout master  
4 ...  
5 $ git commit ...
```



# Commit sur une branche

- Un commit va toujours se faire sur la branche courante

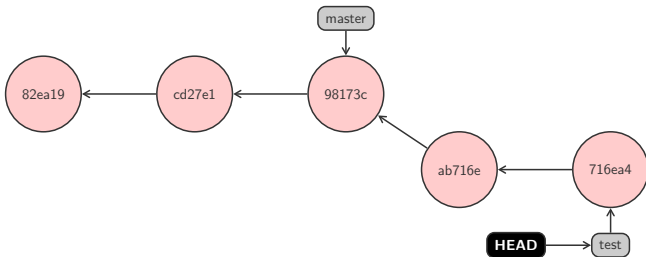
```
1 ...  
2 $ git commit ...  
3 $ git checkout master  
4 ...  
5 $ git commit ...
```



# Commit sur une branche

- Un commit va toujours se faire sur la branche courante

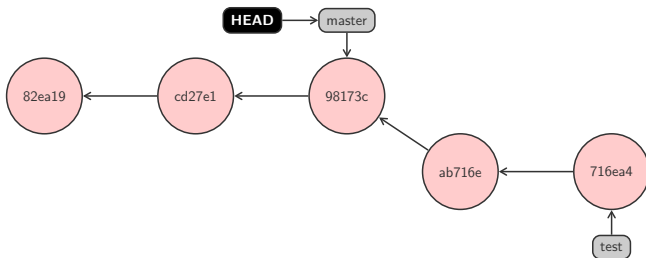
```
1 ...  
2 $ git commit ...  
3 $ git checkout master  
4 ...  
5 $ git commit ...
```



# Commit sur une branche

- Un commit va toujours se faire sur la branche courante

```
1 ...  
2 $ git commit ...  
3 $ git checkout master  
4 ...  
5 $ git commit ...
```

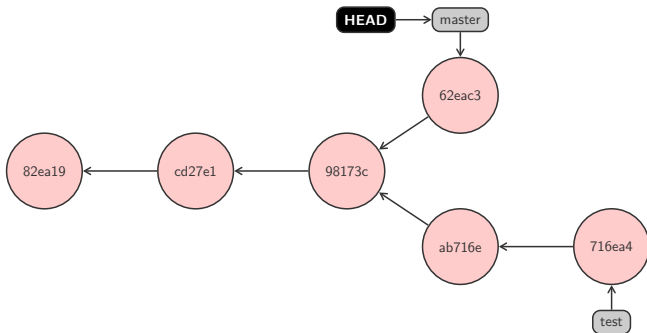




# Commit sur une branche

- Un commit va toujours se faire sur la branche courante

```
1 ...  
2 $ git commit ...  
3 $ git checkout master  
4 ...  
5 $ git commit ...
```



# Opérations de base sur une branche

- On peut **supprimer** une branche avec l'option `-d`

```
1 $ git branch -d test
2 Deleted branch test (was 617a041).
```

- On peut **renommer** une branche avec l'option `-m`

```
1 $ git branch
2 * master
3   test
4 $ git branch -m test alternative
5 $ git branch
6 alternative
7 * master
```

# Plateforme GitHub

- Plateforme d'hébergement de dépôts Git

*Serveur public permettant le partage de code*

- Création gratuite d'un compte pour dépôts publics

*<https://github.com/>*



for this packaging: [www.amazon.com/packaging](http://www.amazon.com/packaging)

amazon.com®

**Packages**

# Pip

- Pip est le gestionnaire de package de Python

*Pip Installs Packages*

- Installation d'un package

*pip **install** <package>*

- Suppression d'un package

*pip **uninstall** <package>*

- Lister les packages

*pip **list***

- Si l'exécutable n'est pas disponible

*python -m **pip** <commande>*

# requirements.txt

- Liste les packages nécessaires pour un projet

*Un nom de package par ligne*

- Installation de tous les packages

*pip **install -r** requirements.txt*

- L'installation est toujours globale

*Les packages sont ajoutés à l'interpréteur Python*



Test unitaire

# Testing

- Nécessité de **tester** qu'un programme fait bien ce qu'il faut
  - Définir ce que le programme doit faire
  - Écrire un jeu de tests pour vérifier le programme

- Impossible de garantir l'**exactitude d'un programme**

*On ne peut pas tester tous les scénarios possibles*

- Amélioration de la **qualité de code**

*Un jeu de tests bien choisi diminue le nombre de bugs potentiels*



# Types de test

- **Test utilisateur** (*usability testing*)

*Évaluer un programme par des tests utilisateurs (ergonomie...)*

- **Test fonctionnel** (*functional testing*)

*Assurance qualité (QA) et test black-box sur les spécifications*

- **Test d'intégration** (*integration testing*)

*Vérification des performances et de la fiabilité du programme*

# Test unitaire

- Test individuel d'une **unité** dans le code

*Une fonction, une classe ou une méthode*

- Définition du test sur base d'une **spécification** du code

*Étant donné les préconditions, vérifier les postconditions*

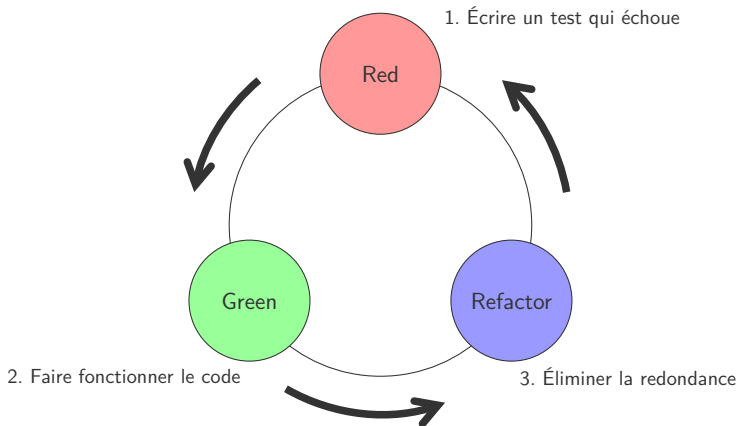
- Utilisé notamment en **Test-Driven Development** (TDD)

*Technique de développement de logiciel piloté par les tests*

# Cycle TDD

- Cycle en **trois phases** principales

*Red-Green-Refactor*



# Module unittest

- **Séparation claire** entre les tests et le code à tester

*Utile lorsque code et tests pas rédigés par les mêmes personnes*

- Quatre concepts clés

**1 Test fixture** *pour initialiser et nettoyer un test*

**2 Test suite** *est un ensemble de test cases*

**3 Test case** *est l'unité de base des tests*

**4 Test runner** *permet d'exécuter des suites de tests*

# Classe de test (1)

- Une suite de tests se définit à partir d'une **classe de test**

*Classe « spéciale » construite sur base de `unittest.TestCase`*

- Utilisation de méthodes prédéfinies pour **exprimer les tests**

*Expression de la valeur attendue d'exécution de code*

Méthode	Description	Test
<code>assertTrue(x)</code>	Affirme que x est vrai	<code>bool(x) is True</code>
<code>assertEqual(a, b)</code>	Affirme que a et b sont égaux	<code>a == b</code>
<code>assertIs(a, b)</code>	Affirme que a et b sont identiques	<code>a is b</code>
<code>assertIsNone(x)</code>	Affirme que x est None	<code>x is None</code>
<code>assertIn(a, b)</code>	Affirme que a se trouve dans b	<code>a in b</code>
<code>assertIsInstance(a, b)</code>	Affirme que a est une instance de b	<code>isinstance(a, b)</code>

Et aussi `assertFalse`, `assertNotEqual`, `assertIsNot`, `assertIsNotNone`, `assertNotIn` et `assertNotIsInstance`...

# Classe de test (2)

```
1 import unittest
2 import program
3
4 class Test(unittest.TestCase):
5     def test_compute(self):
6         self.assertEqual(program.compute(0), 0)
7         self.assertEqual(program.compute(-2), -1)
8
9 suite = unittest.TestLoader().loadTestsFromTestCase(Test)
10 runner = unittest.TextTestRunner()
11 print(runner.run(suite))
```

```
$ python3 test_program.py
F
=====
FAIL: test_compute (__main__.Test)
-----
Traceback (most recent call last):
  File "test_program.py", line 7, in test_compute
    self.assertEqual(testE.compute(-2), -1)
AssertionError: -2 != -1

-----

Ran 1 test in 0.000s

FAILED (failures=1)
<unittest.runner.TextTestResult run=1 errors=0 failures=1>
```

# Initialisation et nettoyage

- **Initialisation** avant et **nettoyage** après exécution de chaque test

*Via les méthodes `setUp` et `tearDown`*

```
1 class Test(unittest.TestCase):  
2     def setUp(self):  
3         # Code exécuté avant chaque test  
4  
5     def tearDown(self):  
6         # Code exécuté après chaque test
```

# Plateforme Travis

- Plateforme d'**exécution automatique** de tests

*Code automatiquement rapatrié depuis GitHub par exemple*

- Création gratuite d'un compte pour tester des dépôts publics

*<https://travis-ci.org/>*



Travis CI



# Configuration de Travis

- Création d'un **fichier .travis.yml** pour la configuration
  - Language de programmation
  - Version spécifique
  - Script à exécuter

```
1 language: python
2 python:
3   - "3.5"
4 script: python3 test.py
```

# Crédits

- <https://www.flickr.com/photos/jwhitesmith/7363049912>
- <https://openclipart.org/detail/36565/tango-network-server-by-warszawianka>
- <https://openclipart.org/detail/34531/tango-computer-by-warszawianka>
- [https://www.flickr.com/photos/faisal\\_akram/8107449789](https://www.flickr.com/photos/faisal_akram/8107449789)
- [https://www.flickr.com/photos/rachel\\_s/9243714784](https://www.flickr.com/photos/rachel_s/9243714784)
- <https://www.flickr.com/photos/110777427@N06/15632985383>
- <https://www.flickr.com/photos/nasamarshall/21064480196>
- <https://www.flickr.com/photos/aidanrissa/6287801918>