W4995 Applied Machine Learning

Git, Github and Testing

01/28/19

Andreas C. Müller

Why do I care?

Version Control Flowchart

Git and Github

https://guides.github.com/

http://rogerdudler.github.io/git-guide/

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL

6 / 58

Configuration

```
git config --global user.name "Andreas Mueller"
git config --global user.email "acm2248@columbia.edu"
git config --global color.ui "auto"
git config --global core.editor "vim"
```

Show your configuration:

```
git config --list
```

Creating a repository

```
$ mkdir homework1
$ cd homework1
```

```
$ git init
```

```
> Initialized empty Git repository in /tmp/homework1/.git/
```

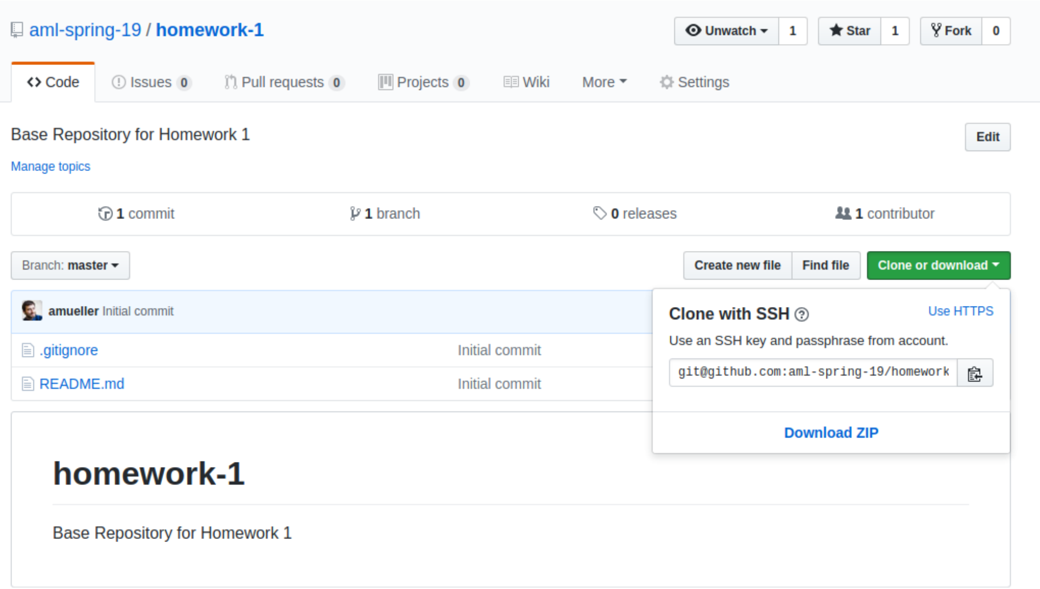
```
$ ls -a
```

```
.  ..  .git
```

```
$ git status
```

```
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Cloning a repository



aml-spring-19 / homework-1

Base Repository for Homework 1

Manage topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master

amueller Initial commit

.gitignore Initial commit

README.md Initial commit

Clone with SSH

Use an SSH key and passphrase from account.

git@github.com:aml-spring-19/homework

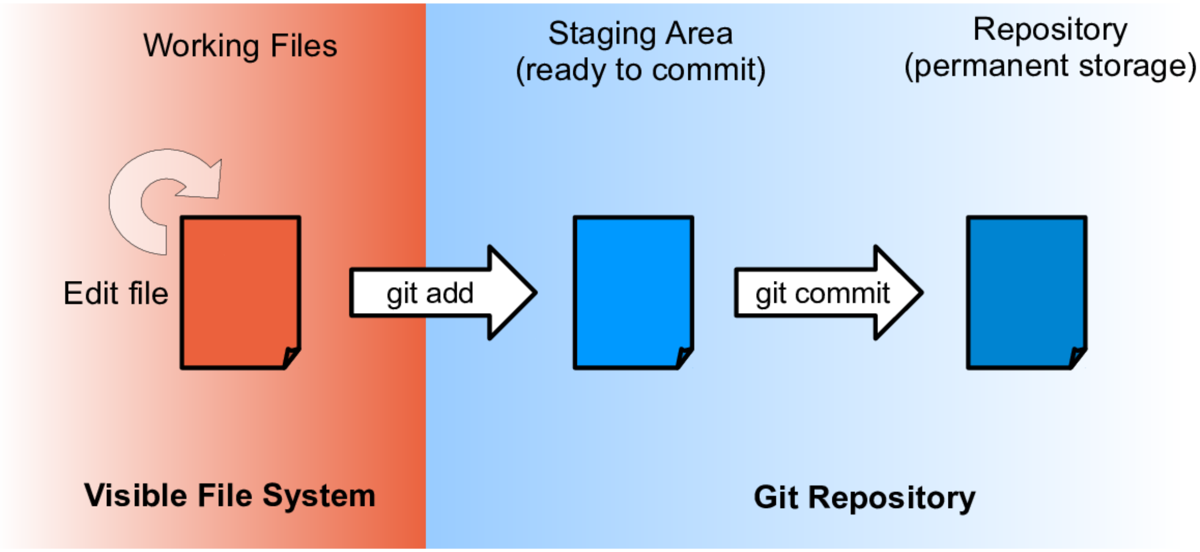
Download ZIP

homework-1

Base Repository for Homework 1

```
$ git clone git@github.com:aml-sprint-19/homework-1
```

Workflow?


Working Files
Staging Area
(ready to commit)
Repository
(permanent storage)
Edit file
git add
git commit
Visible File System
Git Repository

Changing a File

```
$ echo "print('Hello world!')" >> task1.py
```

```
$ git status
```

```
On branch master

No commits yet

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

    task1.py

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

```
$ git add task1.py
```

Viewing your history

```
$ git commit -m "say hello"
```

```
[master (root-commit) 7d139fb] say hello
 1 file changed, 1 insertion(+)
 create mode 100644 task1.py
```

```
$ git log
```

```
commit 7d139fb317ecfa7d629654b709747e91ececa444 (HEAD -> master)
Author: Andreas Mueller <andreas.mueller@columbia.edu>
Date:   Mon Jan 28 12:37:29 2019 -0500

    say hello
```

A note on viewing changes

Changes between working directory and what was last staged

```
git diff
```

Changes between staging area and last commit

```
git diff --staged
```

Referencing different versions

- Shorthand for different versions of a repository (refers to commits)
  - Current Version (most recent commit): `HEAD`
  - Version before current: `HEAD~1`
  - Version before that: `HEAD~2`
- Each of these also has a commit hash
  - use `git log` to get appropriate hash

Exploring History

Changes made in the last commit

```
git diff HEAD~1
```

Changes made in the last 2 commits

```
git diff HEAD~2
```

Changes made since commit hash...

```
git diff 0b0d55e
```

Recovering Older Versions of Files

- Overwrite task1.py:

```
$ echo "print('goodbye, cruel world!')" >  task1.py
$ cat task1.py
```

```
print('goodbye, cruel world!')
```

Recover last recorded version:

```
$ git checkout HEAD task1.py
$ cat task1.py
```

```
print('Hello world!')
```

Recovering Older Versions of whole repo

```
$ echo "print('goodbye, cruel world!')" >  task1.py
$ git add task1.py
$ git commit -m " saying goodbye"
```

```
[master 95f3b39]  saying goodbye
 1 file changed, 1 insertion(+), 1 deletion(-)
```

$ git log

```
commit 95f3b39402451c474e3887be94d2cc37a11be511 (HEAD -> master)
Author: Andreas Mueller <andreas.mueller@columbia.edu>
Date:   Mon Jan 28 12:44:52 2019 -0500

     saying goodbye
```

Resetting:

```
$ git reset --hard HEAD~1
```

```
HEAD is now at 7d139fb say hello
```

Branches

```
$ git checkout -b "new_feature"
```

```
Switched to a new branch 'new_feature'
```

make some changes, add, commit...

Moving between branches:

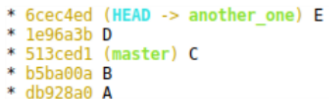
```
$ git checkout master
```

```
Switched to branch 'master'
```

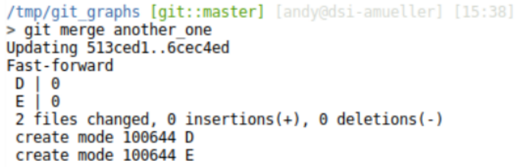
changes are not present in master..

Merge

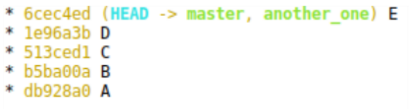
- Fast-forward merge:



```
* 6cec4ed (HEAD -> another_one) E
* 1e96a3b D
* 513ced1 (master) C
* b5ba00a B
* db928a0 A
```

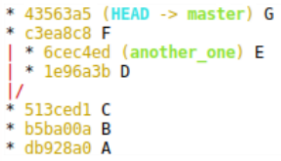


```
/tmp/git_graphs [git::master] [andy@dsi-amueller] [15:38]
> git merge another_one
Updating 513ced1..6cec4ed
Fast-forward
 D | 0
 E | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 D
 create mode 100644 E
```

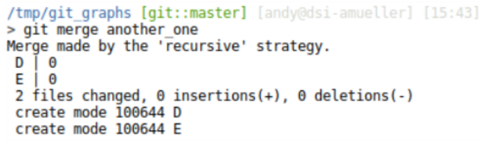


```
* 6cec4ed (HEAD -> master, another_one) E
* 1e96a3b D
* 513ced1 C
* b5ba00a B
* db928a0 A
```

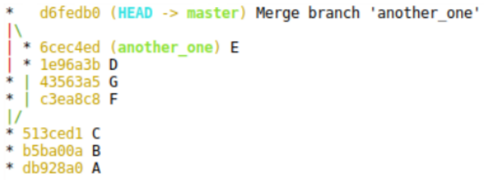
- Merge-commits:



```
* 43563a5 (HEAD -> master) G
* c3ea8c8 F
| * 6cec4ed (another_one) E
| * 1e96a3b D
|/
* 513ced1 C
* b5ba00a B
* db928a0 A
```



```
/tmp/git_graphs [git::master] [andy@dsi-amueller] [15:43]
> git merge another_one
Merge made by the 'recursive' strategy.
 D | 0
 E | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 D
 create mode 100644 E
```



```
*   d6fedb0 (HEAD -> master) Merge branch 'another_one'
|\
| * 6cec4ed (another_one) E
| * 1e96a3b D
* | 43563a5 G
* | c3ea8c8 F
|/
* 513ced1 C
* b5ba00a B
* db928a0 A
```

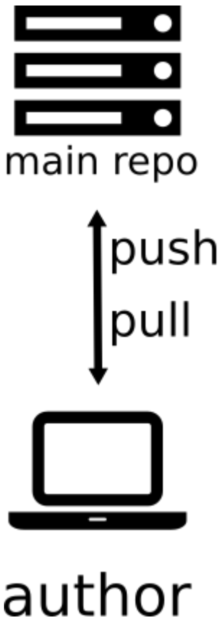
Collaborating using git & github

Remote repository

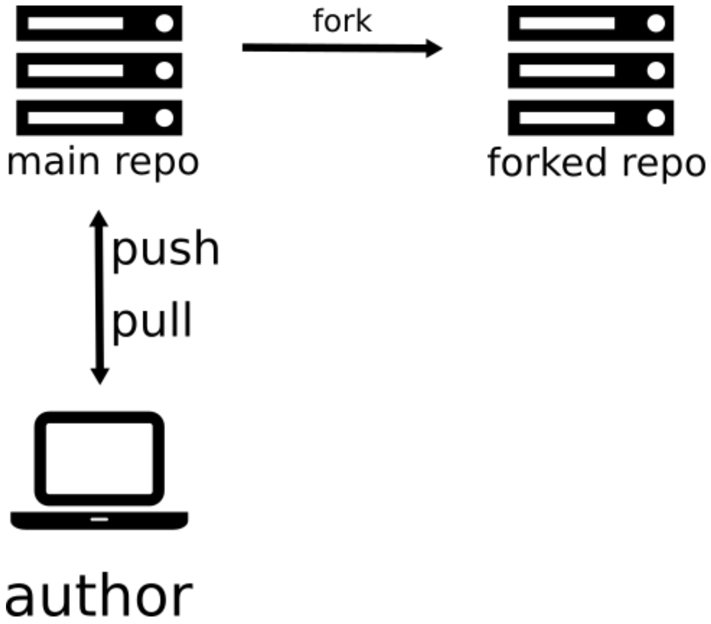
- central location everyone can see
- requires network ?
- github, bitbucket
- public vs private

Github

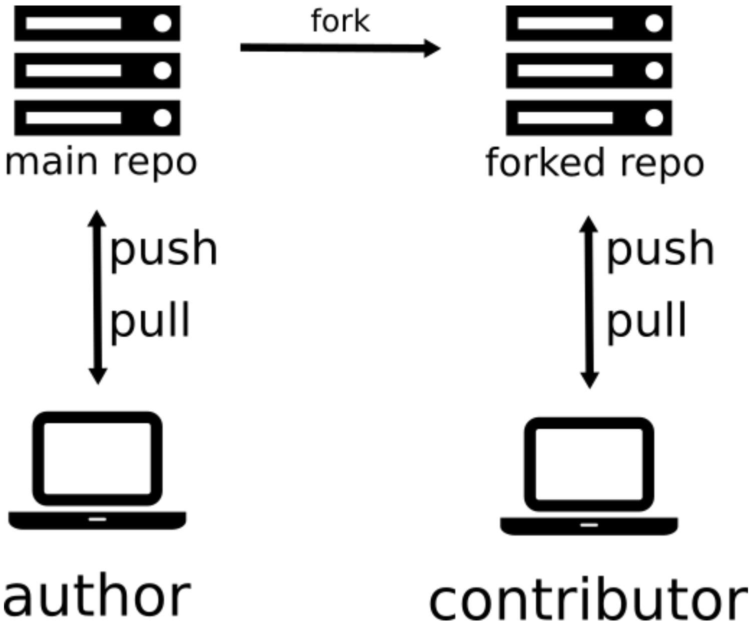
Github pull request workflow



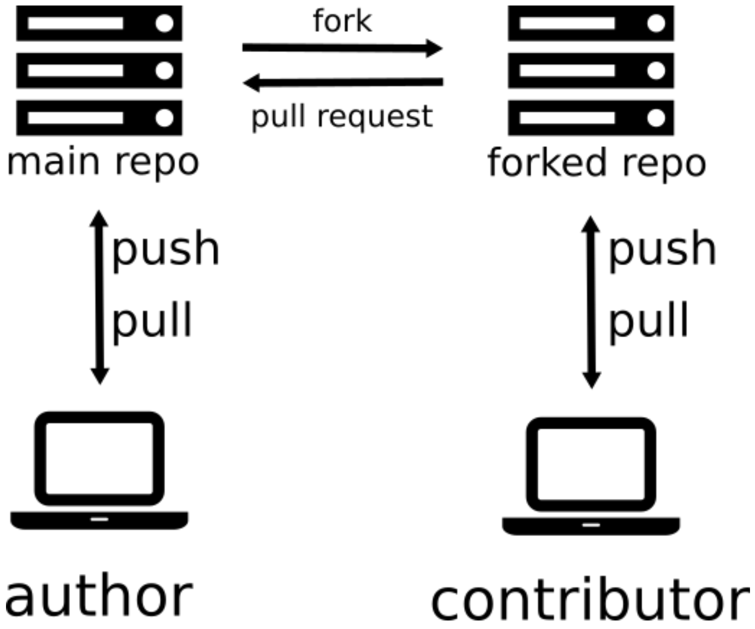
Github pull request workflow



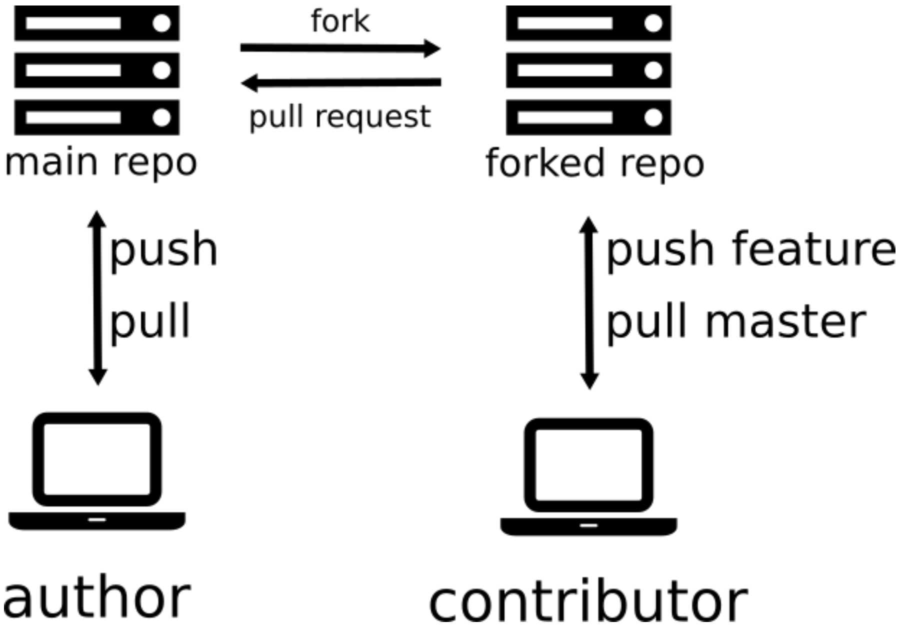
Github pull request workflow



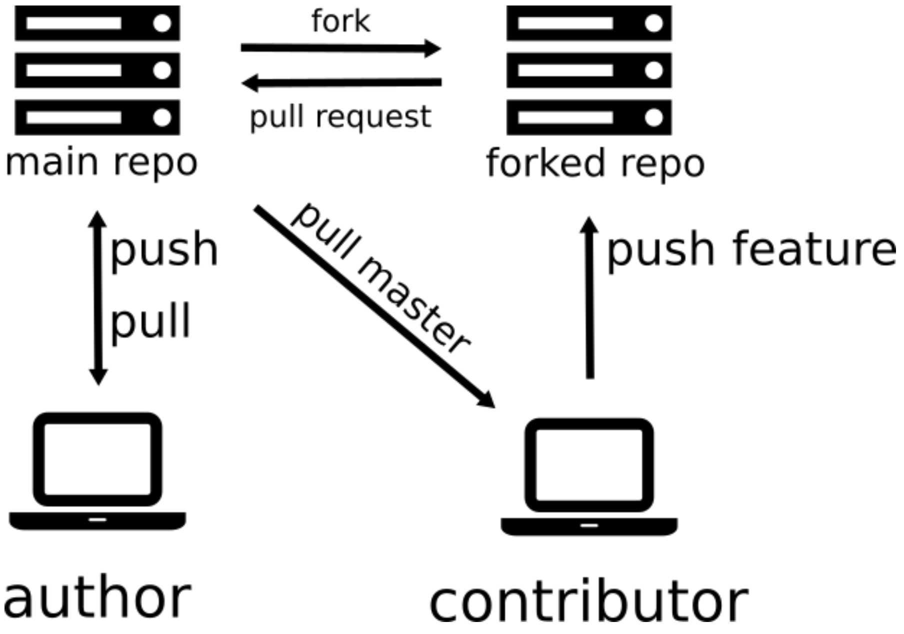
Github pull request workflow



Github pull request workflow



Github pull request workflow

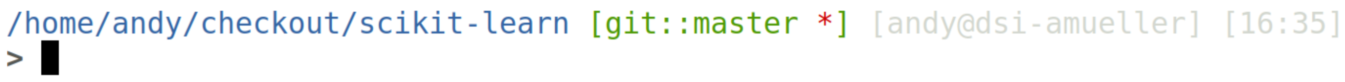


Typical Workflow

- Clone
- Branch
- Add, commit, add, commit, add, commit, ...
- Merge / rebase
- Push

Some tips

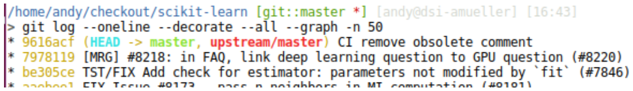
- `$ git status`
- Install shell plugins for status and branch (`oh-my-zsh`)



```
/home/andy/checkout/scikit-learn [git::master *] [andy@dsi-amueller] [16:35]
> █
```

- Set editor, pager and diff-tool (check out meld!)
- Use `.gitignore`

Git log

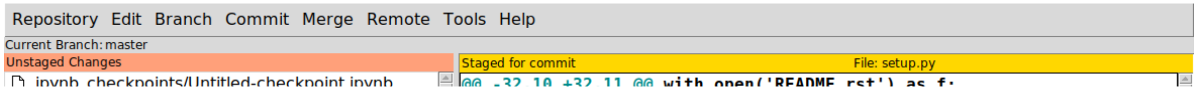


```
/home/andy/checkout/scikit-learn [git::master *] [andy@dsi-amueller] [16:43]
> git log --oneline --decorate --all --graph -n 50
* 9616acf (HEAD -> master, upstream/master) CI remove obsolete comment
* 7978119 [MRG] #8218: in FAQ, link deep learning question to GPU question (#8220)
* be305ce TST/FIX Add check for estimator: parameters not modified by `fit` (#7846)
* aaebee1 FIX Issue #8173 - pass n_neighbors in MI computation (#8181)
```

32 / 58

I DON'T DO GIT LOG ALL THE TIME, BUT WHEN I DO JUST REMEMBER
(A DOG) --ALL --DECORATE --ONELINE --GRAPH
memegenerator.net

git gui



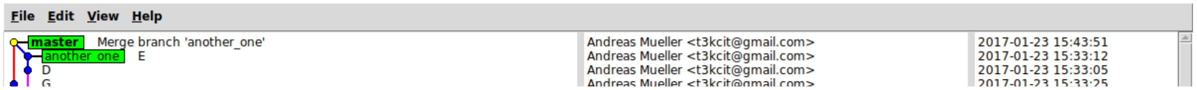
Current Branch: master

File: setup.py

ipynb_checkpoints/Untitled-checkpoint.ipynb

```
@@ -32,10 +32,11 @@ with open('README.rst') as f:
```

gitk



```
master  Merge branch 'another_one'     Andreas Mueller <t3kcit@gmail.com>   2017-01-23 15:43:51
another one  E                          Andreas Mueller <t3kcit@gmail.com>   2017-01-23 15:33:12
D                                       Andreas Mueller <t3kcit@gmail.com>   2017-01-23 15:33:05
G                                       Andreas Mueller <t3kcit@gmail.com>   2017-01-23 15:33:25
```

Understanding Git

- Working directory
- Repository (Commit graph, history)
- Index (Staging Area)
- Branches
- Head

git Commands

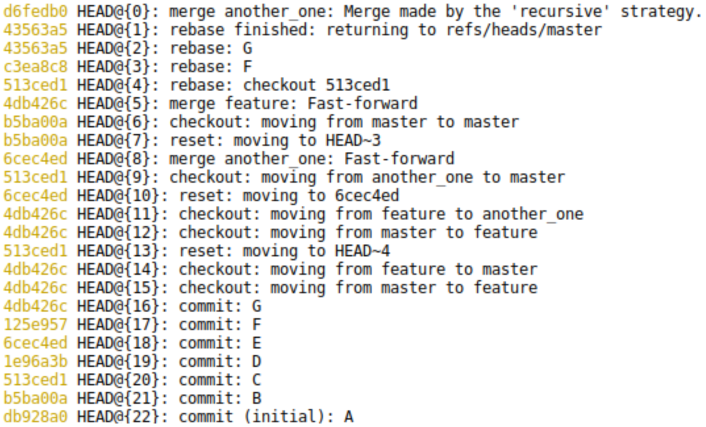
- `git add`
  puts files from working director into staging area (index) If not tracked so far, adds them to tracked files.
- `git commit`
  commits files from staging area (index) to repository, moves current branch with HEAD
- `git checkout [<commit>] [<file>]`
  Set `<file>` in working directory to state at `<commit>` and stages it.
- `git checkout [-b] <branch>`
  moves HEAD to `<branch>` (-b creates it), changes content of working dir
- `git reset --soft <commit>`
  moves HEAD to `<commit>` (takes the current branch with it)
- `git reset --mixed <commit>`
  moves HEAD to `<commit>`, changes index to be at `<commit>` (but not working directory)
- `git reset --hard <commit>`
  moved HEAD to `<commit>`, changes index and working tree to `<commit>`.

Git Data Transport Commands

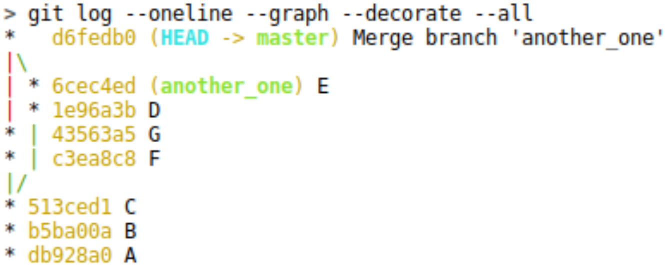
http://osteele.com

reflog

```
$ git reflog
```



```
d6fedb0 HEAD@{0}: merge another_one: Merge made by the 'recursive' strategy.
43563a5 HEAD@{1}: rebase finished: returning to refs/heads/master
43563a5 HEAD@{2}: rebase: G
c3ea8c8 HEAD@{3}: rebase: F
513ced1 HEAD@{4}: rebase: checkout 513ced1
4db426c HEAD@{5}: merge feature: Fast-forward
b5ba00a HEAD@{6}: checkout: moving from master to master
b5ba00a HEAD@{7}: reset: moving to HEAD~3
6cec4ed HEAD@{8}: merge another_one: Fast-forward
513ced1 HEAD@{9}: checkout: moving from another_one to master
6cec4ed HEAD@{10}: reset: moving to 6cec4ed
4db426c HEAD@{11}: checkout: moving from feature to another_one
4db426c HEAD@{12}: checkout: moving from master to feature
513ced1 HEAD@{13}: reset: moving to HEAD~4
4db426c HEAD@{14}: checkout: moving from feature to master
4db426c HEAD@{15}: checkout: moving from master to feature
4db426c HEAD@{16}: commit: G
125e957 HEAD@{17}: commit: F
6cec4ed HEAD@{18}: commit: E
1e96a3b HEAD@{19}: commit: D
513ced1 HEAD@{20}: commit: C
b5ba00a HEAD@{21}: commit: B
db928a0 HEAD@{22}: commit (initial): A
```



```
> git log --oneline --graph --decorate --all
*   d6fedb0 (HEAD -> master) Merge branch 'another_one'
|\
| * 6cec4ed (another_one) E
| * 1e96a3b D
* | 43563a5 G
* | c3ea8c8 F
|/
* 513ced1 C
* b5ba00a B
* db928a0 A
```

Git for ages 4 and up:

https://www.youtube.com/watch?v=1ffBJ4sVUb4

(with play-doh!)

Unit Tests and integration tests

Why test?

- Ensure that code works correctly.
- Ensure that changes don’t break anything.
- Ensure that bugs are not reintroduced.
- Ensure robustness to user errors.
- Ensure code is reachable.

Test-driven development?

Types of tests

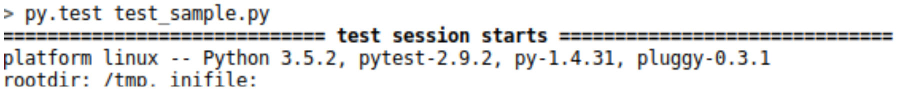
- Unit tests – function does the right thing.
- Integration tests – system / process does the right thing.
- Non-regression tests – bug got removed (and will not be reintroduced).

How to test?

- pytest – http://doc.pytest.org
- Searches for all test*_.py files, runs all test_* methods.
- Reports nice errors!
- Dig deeper: http://pybites.blogspot.com/2011/07/behind-scenes-of-pytests-new-assertion.html

Example

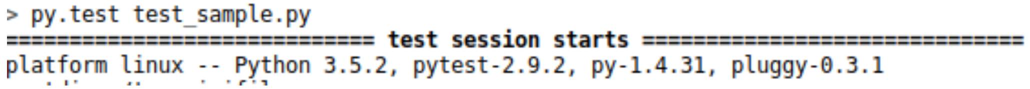
```python
# content of inc.py

def inc(x):
    return x + 2

# content of test_sample.py
from inc import inc

def test_answer():
    assert inc(3) == 4
```



```
> py.test test_sample.py
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /tmp, inifile:
```

Example

```
# content of inc.py

def inc(x):
    return x + 1

# content of test_sample.py
from inc import inc


def test_answer():
    assert inc(3) == 4
```



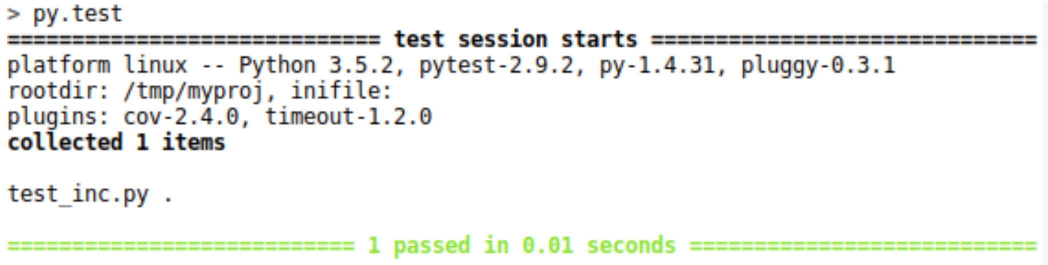
```
> py.test test_sample.py
============================== test session starts ===============================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
```

48 / 58

Test coverage

```
# inc.py
def inc(x):
    if x < 0:
        return 0
    return x + 1


def dec(x):
     return x - 1
```

```
# test_inc.py
from inc import inc

def test_inc():
     assert inc(3) == 4
```

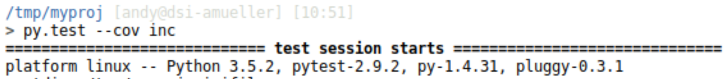


```
> py.test
============================== test session starts ===============================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /tmp/myproj, inifile:
plugins: cov-2.4.0, timeout-1.2.0
collected 1 items

test_inc.py .

=========================== 1 passed in 0.01 seconds ============================
```

Test coverage

```python
# inc.py
def inc(x):
    if x < 0:
        return 0
    return x + 1


def dec(x):
     return x - 1
```

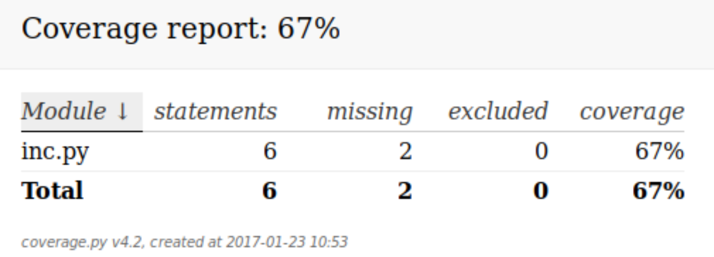
```python
# test_inc.py
from inc import inc

def test_inc():
     assert inc(3) == 4
```



```
/tmp/myproj [andy@dsi-amueller] [10:51]
> py.test --cov inc
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
```

HTML report

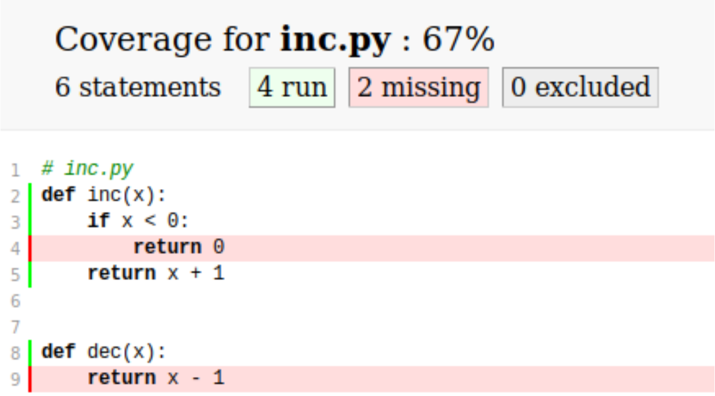
```
$ pytest --cov inc --cov-report=html
```



Coverage report: 67%

Module ↓	statements	missing	excluded	coverage
inc.py	6	2	0	67%
Total	**6**	**2**	**0**	**67%**

coverage.py v4.2, created at 2017-01-23 10:53



Coverage for **inc.py** : 67%

6 statements 4 run 2 missing 0 excluded

```
# inc.py
def inc(x):
    if x < 0:
        return 0
    return x + 1


def dec(x):
    return x - 1
```

Continuous integration (with GitHub)

What is Continuous integration?

- Run command on each commit (or each PR).
- Unit testing and integration testing.
- Can act as a build-farm (for binaries or documentation).
- requires clear declaration of dependencies.
- Build matrix: Can run on many environments.
- Standard serviced: TravisCI, Appveyor, Azure Pipelines and CircleCI

Benefits of CI

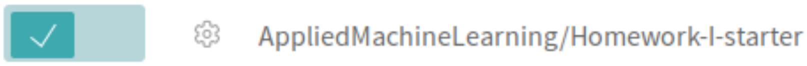
- Can run on many systems
- Can’t forget to run it
- Contributor doesn’t need to know details
- Can enforce style
- Can provide immediate feedback
- Protects the master branch (if run on PR)

What does it do?

- Triggered at each commit / push
- Sets up a virtual machine with your configuration.
- Pulls the current branch.
- Runs command. Usually: install, then test.
- Reports success / Failure to github.

Setting up TravisCI

- Create account linked to your github account: http://travis-ci.org



☑ ⚙ AppliedMachineLearning/Homework-I-starter

- Check out docs at https://docs.travis-ci.com

```
language: python
python:
  - "2.7"
  - "3.4"
  - "3.5"
  - "3.6"
  - "3.7"
  - "nightly"
# command to install dependencies
install:
  - pip install -r requirements.txt
# command to run tests
script:
  - pytest # or pytest for Python versions 3.5 and below
```

Using Travis

- Triggered any time you push a change
- Integrated with Pull requests
- Try a pull request on your own repository!

Questions ?