CONTINUOUS INTEGRATION & DELIVERY

Matthieu Caneill, February 2015

GIT, WHAT DO YOU KNOW?

Raise hands if you know...

```
$ git clone
```

clones a repository from a remote URL

```
$ git pull
```

receives the new commits from the remote

```
$ git add
```

adds files or folders to the *index*

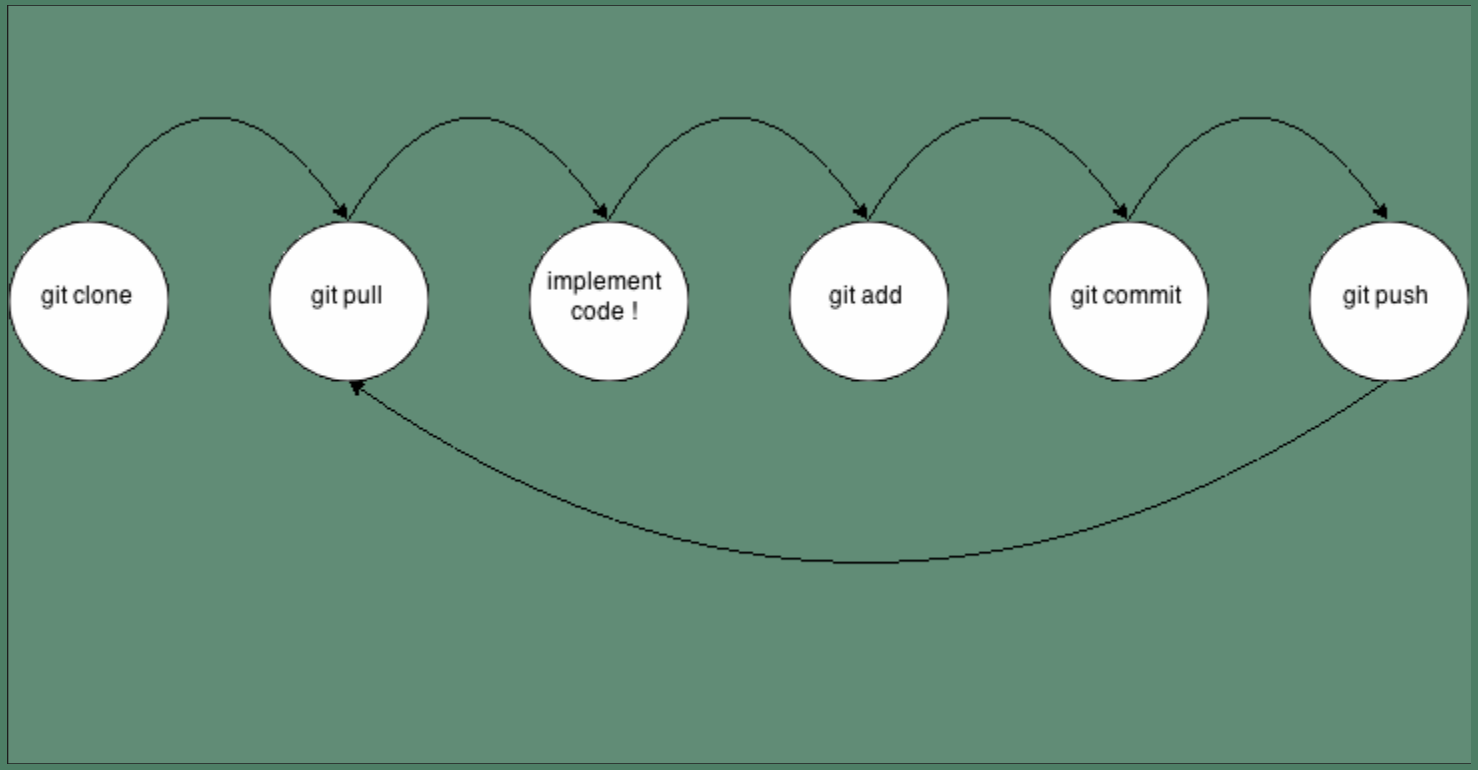
```
$ git commit
```

aggregates the changes in the index

```
$ git push
```

sends the new commits to the remote

GIT WORKFLOW



MANIPULATIONS

- Create or fork a repo on Github
- Clone it
- Add some content, commit it, push
- Watch the new file in Github

CONTINUOUS INTEGRATION

GOALS

- Have a stable software after each commit
- Simplify the release process
- Spot errors as soon as they appear

HOW?

- Run tests (unit tests, functional tests, integration tests, * tests...)
- If tests pass, the patch (commit) can be accepted and integrated
- CI servers do this automatically (for instance Jenkins at home or Travis-CI in the cloud)

MANIPULATIONS

With Travis-CI

- Follow instructions on http://air.imag.fr/index.php/TAGL/TP, section 2a

CONTINUOUS DELIVERY

GOALS

- Have up-to-date production servers
- Slow the "release-deploy" pain

HOW?

- git hooks
- use of production branches
- git pull/push from/to production servers

MANIPULATIONS

With Heroku

- Clone heroku/node-js-getting-started on Github
- Create an Heroku account, an app, and link it to the Github repo
- Deploy manually
- Activate automatic deployment
- Change content, commit, push