Enseigner avec GitLab

Aurélien Esnard

Enseigner avec GitLab

GT Innovation Pédagogique
UF Info. @ Univ. de Bordeaux
~
aurelien.esnard@u-bordeaux.fr
30/11/2021

Motivations

Dans le contexte de l'enseignement en Licence Informatique…

- Difficultés à coordonner une UE à fort effectifs et/ou une équipe pédagogique importante !
  - Comment faciliter, fluidifier et inciter le travail de l'équipe pédagogique ?
- Problématique de l'écriture des supports de cours + ressources
  - cycle "dev" : écrire, relire, publier, corriger, publier, …
  - ops : automatiser la publication des supports, …
- Problématique de la gestion des devoirs / projets étudiants
  - rendu des devoirs → lourdeur de la gestion à la main (mails, …)
  - mieux suivre les étudiants en ayant accès au travail en cours… → feedback accéléré
  - automatiser la gestion des devoirs → correction automatique

Objectifs : Utiliser des outils modernes pour enseigner, et surtout être plus efficace pour la coordination d'une UE et l'interaction pédagogique avec nos étudiants. Partager les bonnes pratiques.

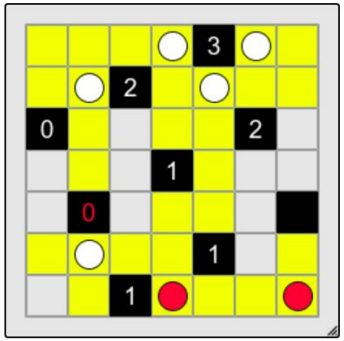
Question : Quels outils peuvent nous aider ? → Moodle, VPL et GitLab.

Contexte

UE Projet Techno en L2 Info & L3 Math-Info

- Effectifs : 180 étudiants, 10 groupes de TD, 54 équipes de 3 étudiants
- Moodle : https://moodle1.u-bordeaux.fr/course/view.php?id=9142

Objectifs



Jeu Lightup en 2021

- Développer en langage C99 un petit jeu de type *puzzle 2D*
  - spécification imposée : https://pt2.gitlabpages.inria.fr/puzzle/lightup/html/
- Travailler sur un projet logiciel en équipe avec Git
- Suivi précis des projets étudiant par les enseignants
  - lister tous les projets de son TD, consulter (ou modifier) le code étudiant, revue de code (issues, bugs), statistiques, …
- Projets découpés en plusieurs jalons tout au long de l'année
  - make, cmake, interface texte, tests, implémentation du jeu, solveur, interface graphique, portage Android, …
- Rendu avec Moodle de chaque jalon sous forme d'un <commit> Git
  - correction automatique avec VPL (scripts run & eval) → notation

Plan

1. Introduction Rapide à Git
2. Introduction à GitLab
3. Enseigner avec GitLab
   a. Contexte de l'UE "Projet Techno"
   b. Ecrire et Publier son Cours avec GitLab
   c. Gérer des Projets Étudiants avec GitLab
4. Bilan

Introduction Rapide à Git

Introduction à Git

Un logiciel de gestion des versions d'un projet (code, doc, …)

- Développer un logiciel à plusieurs, c'est difficile !
  - Sauvegarder l'historique des versions du code, release, …
  - Fusionner les différentes contributions, sans rien perdre…
- Développé en 2005 par Linus Torvalds (le créateur de Linux), sous licence GPL
- Décentralisé
  - Chaque machine possède une copie locale du dépôt distant (autonaume)
  - Possibilité de travailler sur le dépôt local, même sans le réseau
  - Généralement il y a un dépôt central (*origin*) et des dépôts secondaires…
- Le gestionnaire de version actuellement le plus populaire !
  - Successeur de CVS, SVN, …
- Un outil très puissant, mais complexe !

Introduction Git

1) Créer son projet sur Git, puis récupérer une copie locale

```
$ git clone git@gitlab.emi.u-bordeaux.fr/auesnard/test
```

2) Créer ou modifier un fichier puis faire un commit (ou plusieurs)

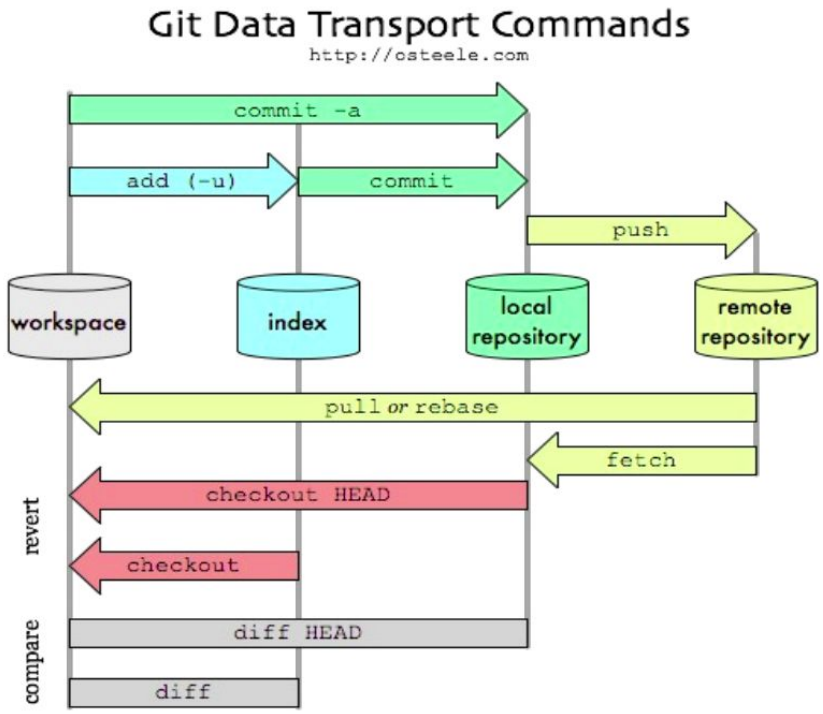
```
$ git add file1 [file2 ...]
$ git commit -m "my message"
```

3) Synchronisation avec le dépôt distant (*origin*)

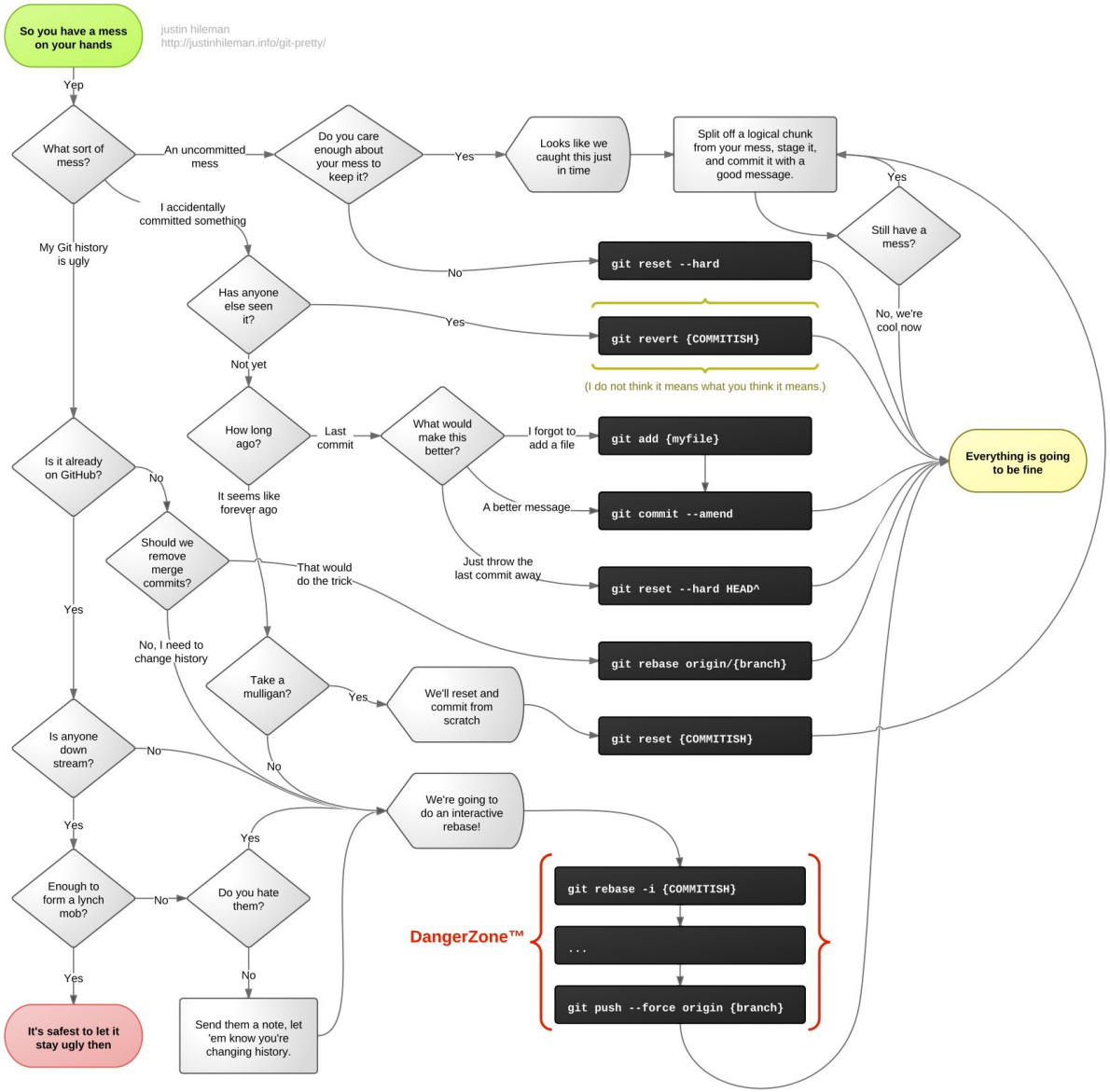
```
# récupération des autres commits, fusion...
git pull

# envoi de mes derniers commits
git push
```

4) History & Log

```
# afficher tous les commits & revisions
git log
```



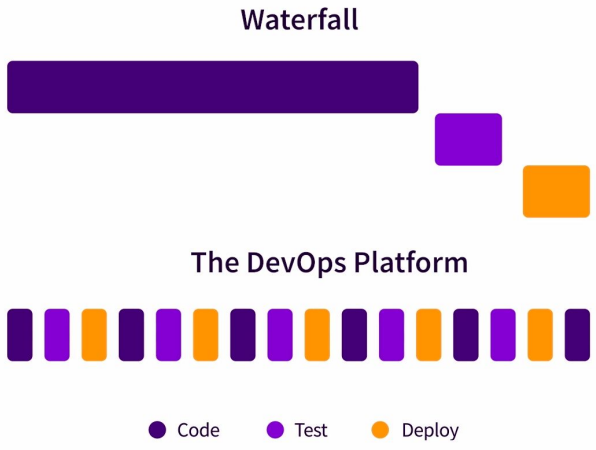
Git Mess !

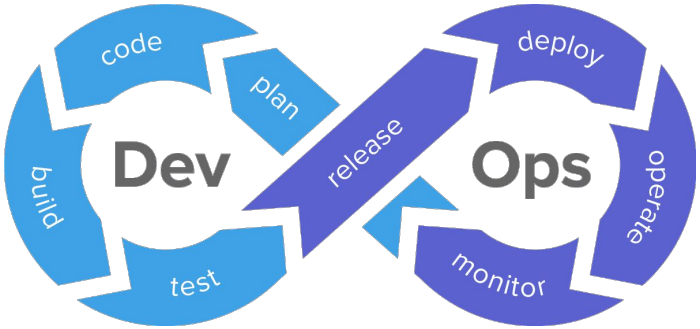

So you have a mess on your hands
justin hileman
http://justinhileman.info/git-pretty/
Yep
What sort of mess?
An uncommitted mess
Do you care enough about your mess to keep it?
Yes
Looks like we caught this just in time
Split off a logical chunk from your mess, stage it, and commit it with a good message.
Yes
Still have a mess?
No, we're cool now
No
git reset --hard
I accidentally committed something
My Git history is ugly
Has anyone else seen it?
Yes
git revert {COMMITISH}
(I do not think it means what you think it means.)
Not yet
How long ago?
Last commit
What would make this better?
I forgot to add a file
git add {myfile}
A better message
git commit --amend
Just throw the last commit away
git reset --hard HEAD^
It seems like forever ago
Is it already on GitHub?
No
Should we remove merge commits?
That would do the trick
git rebase origin/{branch}
Yes
No, I need to change history
Take a mulligan?
Yes
We'll reset and commit from scratch
git reset {COMMITISH}
Is anyone down stream?
No
No
We're going to do an interactive rebase!
Yes
Yes
Enough to form a lynch mob?
No
Do you hate them?
Yes
No
DangerZone™
git rebase -i {COMMITISH}
...
git push --force origin {branch}
It's safest to let it stay ugly then
Send them a note, let 'em know you're changing history.
Everything is going to be fine

Introduction à GitLab

GitLab

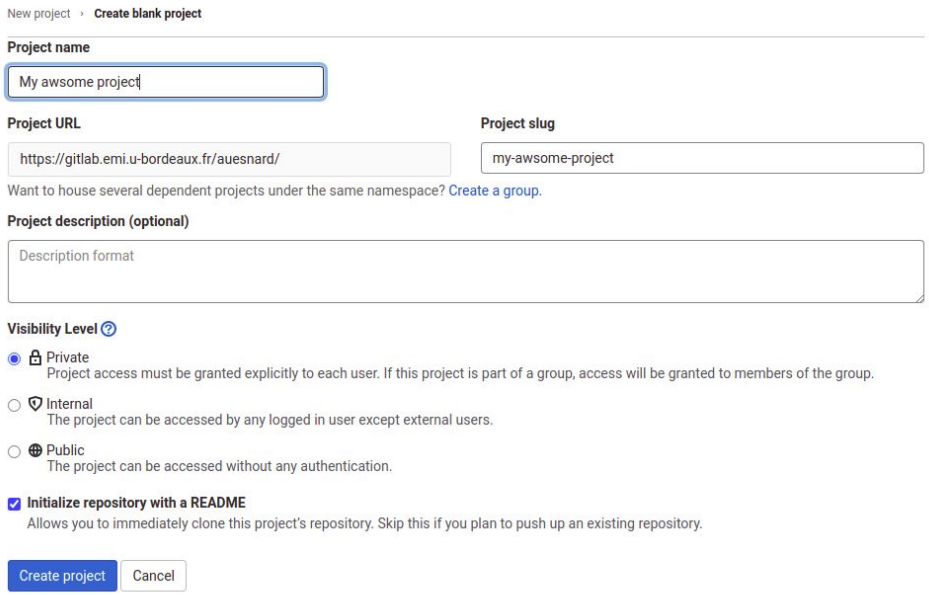
- Une forge logicielle basée sur Git (depuis 2011)
  - GitLab CE : distribution libre et open-source (licence MIT)
  - GitLab EE : distribution propriétaire avec plus de fonctionnalités
  - Version en ligne https://gitlab.com/
- Une plateforme DevOps unifiée
  - DevOps = *software development* (Dev) + *operations* (Ops, exploitation en français)
  - Un ensemble de pratiques & techniques qui visent à réduire le temps entre la modification du logiciel et son exploitation (déploiement / production)
- Les fonctionnalités principales
  - Source Code Management (SCM) basé sur Git
  - Continuous Integration, Delivery and Deployment (CI/CD)
  - Divers : issues, wiki, pages, releases, API Restful, …
- Plateforme très populaire
  - Utilisé par IBM, NVIDIA, Siemens, NASA, …





GitLab

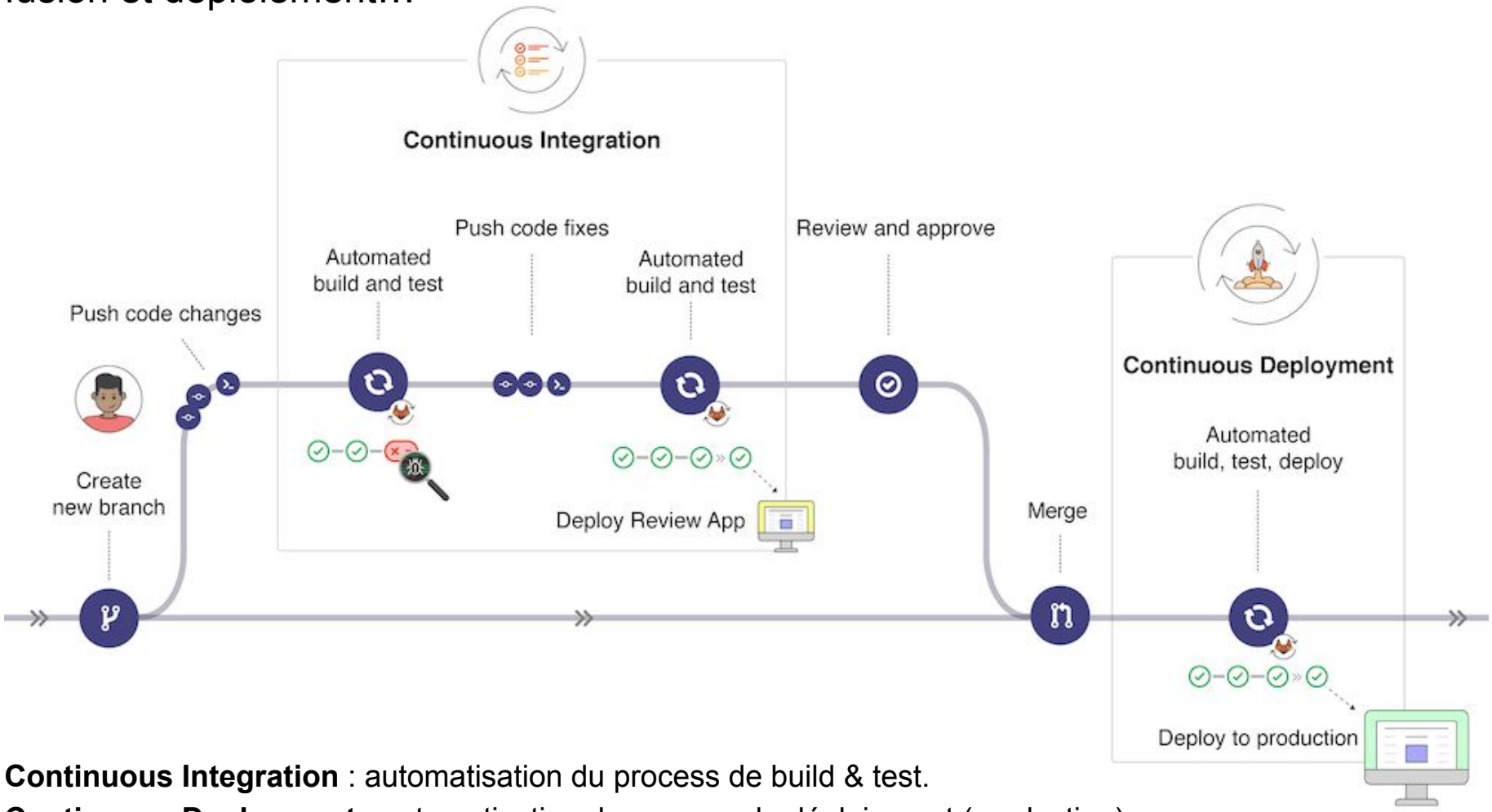
- Créer son projet dans GitLab en quelques clics…
  - Visibilité : *private, internal, public*
- Ajouter de membres à son projet
  - Rôle : *guest < developper < maintainer < owner*



- Quelques GitLab académiques
  - Inria : https://gitlab.inria.fr (ouvert aux équipes Inria, sponsoring)
  - UBx : https://gitub.u-bordeaux.fr (ouvert aux enseignants-chercheurs ubx)
  - CREMI : https://gitlab.emi.u-bordeaux.fr (ouvert aux enseignants & étudiants du CREMI)

GitLab CI/CD Workflow

Développement d'une nouvelle fonctionnalité dans une branche, vérification, fusion et déploiement…

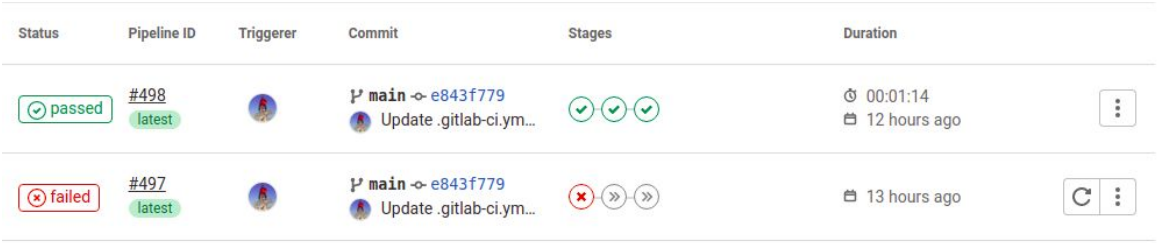


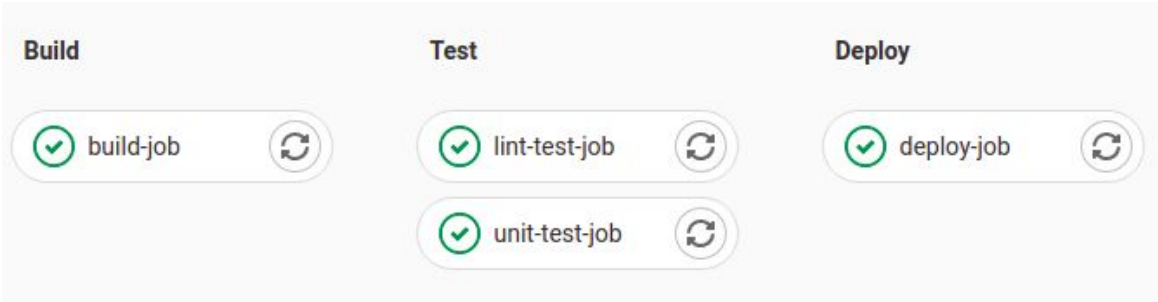
Continuous Integration : automatisation du process de build & test.
Continuous Deployment : automatisation du process de déploiement (production).

GitLab Runner

Principe : Après chaque *push*, le *runner* est en charge de récupérer le projet Git et d'exécuter différentes tâches (scripts) spécifiques à ce projet : *build / test / deploy / …*

- Description des tâches dans un fichier *.gitlab-ci.yml* dans son dépôt Git
- Visualisation de l'état du pipeline dans GitLab






```yaml
# fichier .gitlab-ci.yml

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  script:
    - echo "Compiling the code..."

unit-test-job:
  stage: test
  script:
    - echo "Running unit tests..."

lint-test-job:
  stage: test
  script:
    - echo "Linting code..."

deploy-job:
  stage: deploy
  script:
    - echo "Deploying application..."
```

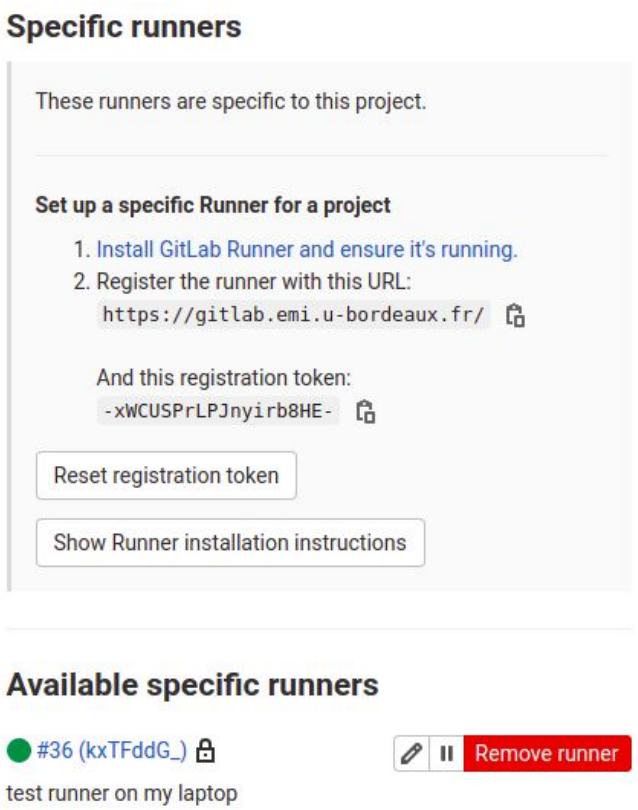
GitLab Runner

- Configuration d'un *runner* sur une machine (*laptop, serveur, vm@cloud*)

```
$ sudo apt install gitlab-runner
```

- Association du *runner* dans les *Settings* de son projet GitLab via un token…

```
$ sudo gitlab-runner register
Runtime platform (arch=amd64 os=linux pid=53111 version=13.7.0)
Running in system-mode.
Enter the GitLab instance URL :https://gitlab.emi.u-bordeaux.fr/
Enter the registration token:-xWCUSPrLPJnyirb8HE-
Registering runner... succeeded (runner=-xWCUSPr)
Enter an executor : shell # or docker
Runner registered successfully.
```



- Utilisation d'une image *Docker* pour contrôler l'environnement d'exécution (les dépendances)

```
# fichier Dockerfile
FROM ubuntu:20.10

RUN apt update
RUN apt install -yq build-essential bash gcc make cmake git
RUN apt install -yq python3 ...

$ docker build -t "orel33/myubuntu:latest" .
$ docker push "orel33/myubuntu:latest"
```

Enseigner avec GitLab

Enseigner avec GitLab

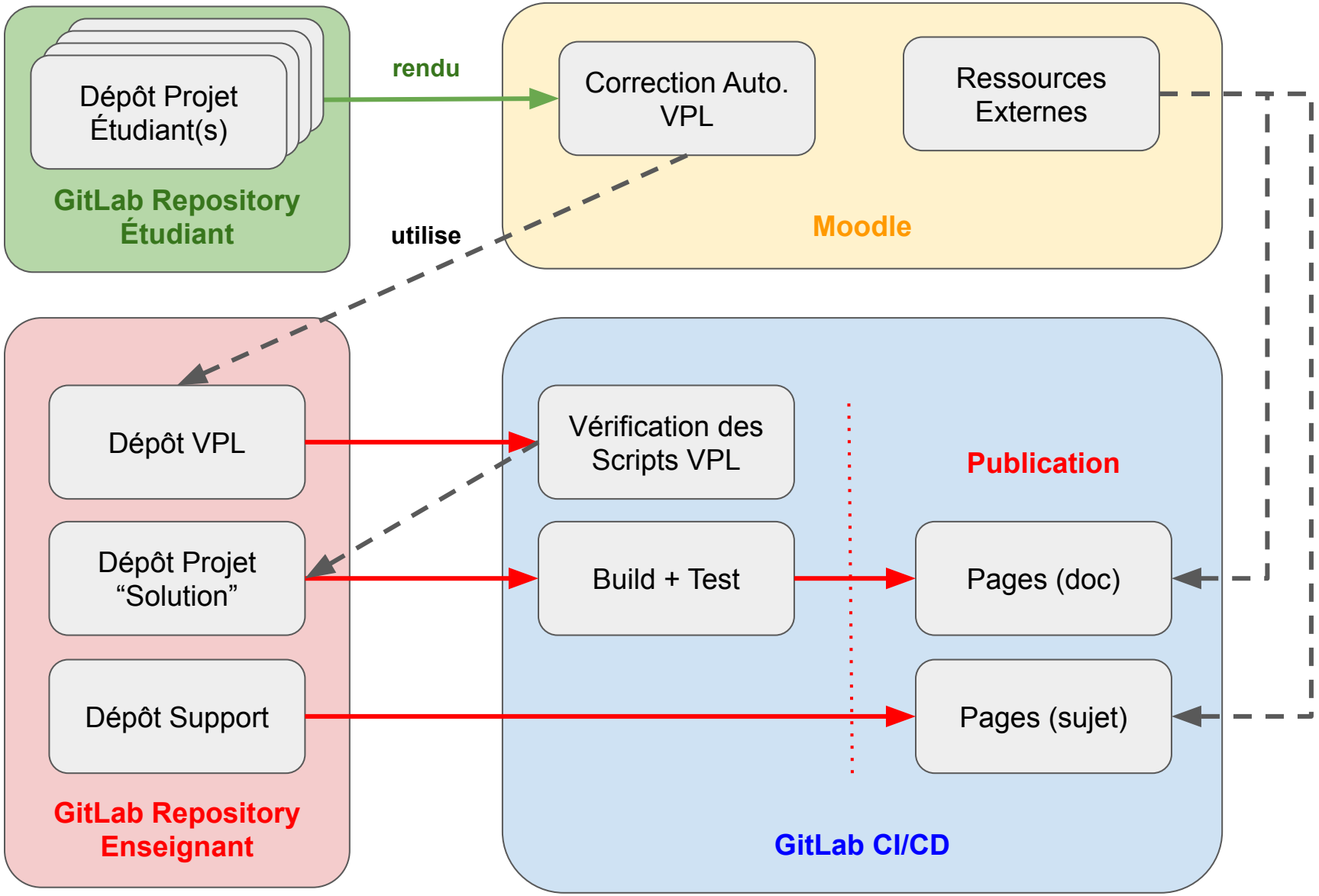
Problématiques

- Comment coordonner efficacement son équipe pédagogique et favoriser le travail en équipe ?
- Comment automatiser la publication des supports de cours (dans des formats accessibles) ?
- Comment partager les ressources publiquement en dehors de Moodle ?
- Comment automatiser la création/gestion/correction des devoirs étudiants ?
- Comment vérifier les scripts de correction automatique ?

Utilisation de GitLab à plusieurs niveaux

- Amélioration continue des supports par l'équipe enseignante
- Déploiement automatique des supports sur GitLab Pages
- Externaliser le plus possible les ressources en dehors de Moodle…
- Créer automatiquement tous les projets étudiants…

Solution Moodle + GitLab



Mise en oeuvre avec Moodle

TD05 : Tests

Durée : 3 semaines (dont 1 semaine de vacances)

Rendu Initial : coeff 1, peu de travail supplémentaire hors TDM (individuel)

Rendu Final : coeff 5, +3h00 de travail supplémentaire hors TDM (en équipe)

- Introduction aux tests unitaires (vidéo de N. Bonichon)
- Sujet du TD5
- 05 - Rendu Test - Initial (TDM)
- 05 - Rendu Test - Final (TEAMS)
- 05 - Rendu Test - Final (deuxième passe) (TEAMS)

rendu.txt

```
1  REPOSITORY: https://gitlab.emi.u-bordeaux.fr/pt2/teams/a11/lightup-a11a
2  COMMIT: 6f990760bb1e9a8f0c8923739ec2613731ba5c3e
3
```

Rendu dans VPL du projet étudiant (dépôt GitLab)

TD5 : Tests

Ce TD va vous apprendre à programmer une batterie de tests pour tester le bon fonctionnement de la bibliothèque *game*.

Exercice 1 : activité préliminaire

Considérez l'exemple d'une structure de données "file" (ou *queue* en anglais), tel que les éléments les premiers entrés sont aussi les premiers sortis (First In, First Out) : https://github.com/orel33/queue.

- Faites un *clone* de ce projet Git.

Ce petit projet se compose de plusieurs fichiers, dont voici une brève description :

- le module *queue* (`queue.c` + `queue.h`)
- un exemple d'utilisation de la queue (`sample.c`)
- un fichier de tests du module *queue* (`test_queue.c`)
- le fichier `CMakeLists.txt` pour compiler ce projet

La compilation du projet et l'exécution des tests se fait de la manière suivante :

```
$ mkdir build ; cd build
$ cmake .. ; make          # compilation
$ make test                # lancement des tests
```

- Analysez en particulier le code des fichiers `queue.h` , `test_queue.c` .
- Dans le fichier `CMakeLists.txt` , comprenez le rôle des commandes `add_test()` ainsi que de la commande `enable_testing()` en préambule.

Sujet sur GitLab Pages (lien web)

GitLab Enseignant

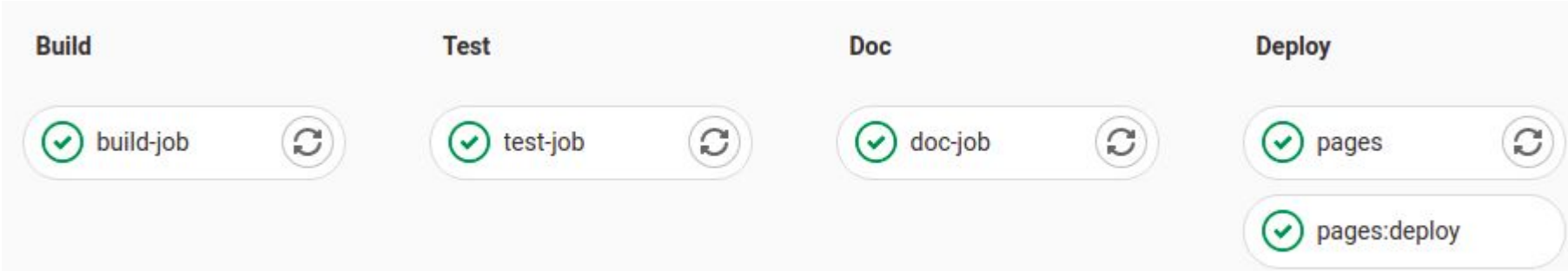
Les ressources pour les enseignants : https://gitlab.inria.fr/pt2

GitLab Enseignant

Version Enseignant du Projet : la “solution”

- Dépôt : https://gitlab.inria.fr/pt2/puzzle/lightup
- Coder le projet comme les étudiant
  - évaluer la difficulté/faisabilité au début de l’année...
- Effectuer de la CI/CD pour vérifier sa propre solution… → test unitaires
- Autres intérêts
  - validation des scripts de correction automatique
  - préparer des ressources : spécification, documentation
  - proposer une correction à la fin du semestre…

```
13/20 Test #13: testv1_get_flags .................   Passed    0.00 sec
      Start 14: testv1_is_state
14/20 Test #14: testv1_is_state ..................   Passed    0.00 sec
      Start 15: testv1_has_flag
15/20 Test #15: testv1_has_flag ..................   Passed    0.00 sec
      Start 16: testv1_play_move
16/20 Test #16: testv1_play_move .................   Passed    0.00 sec
      Start 17: testv1_check_move
17/20 Test #17: testv1_check_move ................   Passed    0.00 sec
      Start 18: testv1_restart
18/20 Test #18: testv1_restart ...................   Passed    0.00 sec
      Start 19: testv1_update_flags
19/20 Test #19: testv1_update_flags ..............   Passed    0.00 sec
      Start 20: testv1_is_over
20/20 Test #20: testv1_is_over ...................   Passed    0.00 sec
100% tests passed, 0 tests failed out of 20
Total Test time (real) =   0.05 sec
Built target ExperimentalTest
Uploading artifacts for successful job
Uploading artifacts...
build/: found 236 matching files and directories
test.log: found 1 matching files and directories
Uploading artifacts as "archive" to coordinator... ok  id=1564531 respon
Cleaning up project directory and file based variables
Job succeeded
```



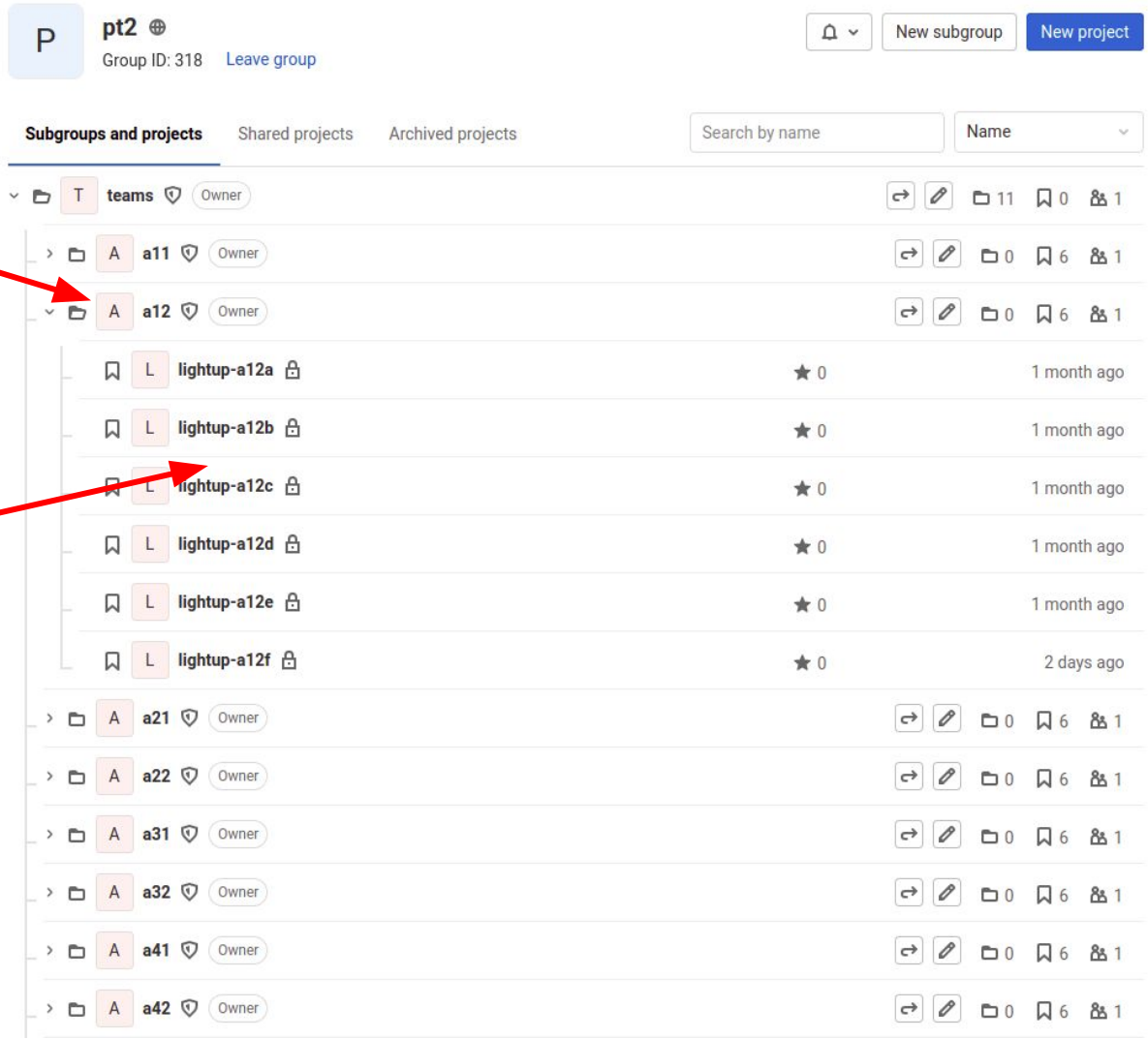
Génération Documentation + API : https://pt2.gitlabpages.inria.fr/puzzle/lightup/ (Doxygen)

GitLab Étudiant

Organisation des projets étudiants : https://gitlab.emi.u-bordeaux.fr/pt2

Groupe de TM A12

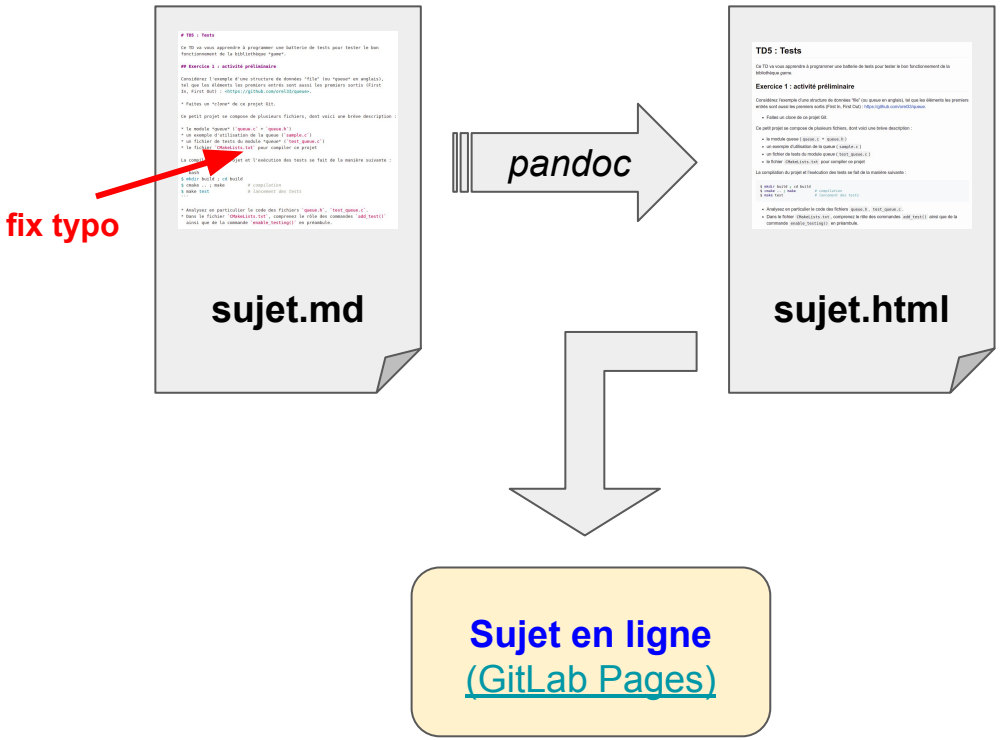
Projet Git de l'équipe A12B



Écrire et Publier son Cours avec GitLab

Dépôt : https://gitlab.inria.fr/pt2/support

- Versionner dans Git les supports de cours et les diverses ressources
- Travail collaboratif de l'équipe pédagogique avec Git
- Utilisation de GitLab CI/CD (.gitlab-ci.yml) pour produire les supports et les déployer automatiquement sur le web (GitLab Pages)




```yaml
# fichier .gitlab-ci.yml

image: orel33/mydebian

pages:
  stage: deploy
  script:
    - mkdir -p public
    - cp style.css sujet.md public/
    - cd public/td
    - pandoc -s -t html4 --css style.css sujet.md \
             -o sujet.html )
    # ...
  artifacts:
    paths:
      - public  # export web
  only:
    - master
```

Écrire et Publier son Cours avec GitLab

Problématique de l'accessibilité

- Promouvoir des formats accessibles comme MarkDown
- Conversion avec *pandoc* du Markdown vers HTML (style CSS) & PDF
- Quelques limites du Markdown (images, include, conditionnel)

```
# TD5 : Tests

Ce TD va vous apprendre à programmer une batterie de tests pour tester le bon
fonctionnement de la bibliothèque *game*.

## Exercice 1 : activité préliminaire

Considérez l'exemple d'une structure de données "file" (ou *queue* en anglais),
tel que les éléments les premiers entrés sont aussi les premiers sortis (First
In, First Out) : <https://github.com/orel33/queue>.

* Faites un *clone* de ce projet Git.

Ce petit projet se compose de plusieurs fichiers, dont voici une brève description :

* le module *queue* (`queue.c` + `queue.h`)
* un exemple d'utilisation de la queue (`sample.c`)
* un fichier de tests du module *queue* (`test_queue.c`)
* le fichier `CMakeLists.txt` pour compiler ce projet

La compilation du projet et l'exécution des tests se fait de la manière suivante :

```bash
$ mkdir build ; cd build
$ cmake .. ; make         # compilation
$ make test               # lancement des tests
```

* Analysez en particulier le code des fichiers `queue.h`, `test_queue.c`.
* Dans le fichier `CMakeLists.txt`, comprenez le rôle des commandes `add_test()`
  ainsi que de la commande `enable_testing()` en préambule.
```

pandoc

TD5 : Tests

Ce TD va vous apprendre à programmer une batterie de tests pour tester le bon fonctionnement de la bibliothèque *game*.

Exercice 1 : activité préliminaire

Considérez l'exemple d'une structure de données "file" (ou *queue* en anglais), tel que les éléments les premiers entrés sont aussi les premiers sortis (First In, First Out) : https://github.com/orel33/queue.

- Faites un *clone* de ce projet Git.

Ce petit projet se compose de plusieurs fichiers, dont voici une brève description :

- le module *queue* (`queue.c` + `queue.h`)
- un exemple d'utilisation de la queue (`sample.c`)
- un fichier de tests du module *queue* (`test_queue.c`)
- le fichier `CMakeLists.txt` pour compiler ce projet

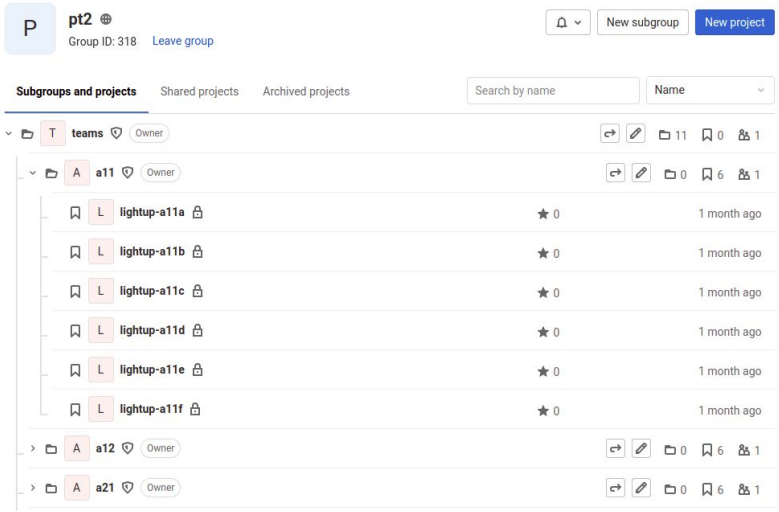
La compilation du projet et l'exécution des tests se fait de la manière suivante :

```
$ mkdir build ; cd build
$ cmake .. ; make         # compilation
$ make test               # lancement des tests
```

- Analysez en particulier le code des fichiers `queue.h` , `test_queue.c` .
- Dans le fichier `CMakeLists.txt` , comprenez le rôle des commandes `add_test()` ainsi que de la commande `enable_testing()` en préambule.

Gérer des Projets Étudiants avec GitLab

- Organisation des projets étudiants
  - Groupe racine : https://gitlab.emi.u-bordeaux.fr/pt2
  - Organisation hiérarchique en sous-groupes : <ue>/<year>/<td>/<team>
- Création automatique de tous les projets étudiants (privés)
  - Projet de l'équipe A du TD A11 : https://gitlab.emi.u-bordeaux.fr/pt2/teams/a11/lightup-a11a
- Les enseignants sont membres du groupe racine
  - avec le rôle *owner* → héritage des permissions
  - accès à l'ensemble des projets étudiants
- Ajout (automatique) des étudiants comme membre de chaque projet
  - avec le rôle *maintainer* dans leur projet uniquement
  - les étudiants administrent entièrement leur projet, mais ne voient pas les autres projets !



Gérer des Projets Étudiants avec GitLab

Création automatique des groupes et des projets avec l'API RESTful

- création d'un *personnal access token* dans GitLab
- effectuer de simples requêtes web avec *curl*
- écrire un script pour automatiser la création de tous les projets

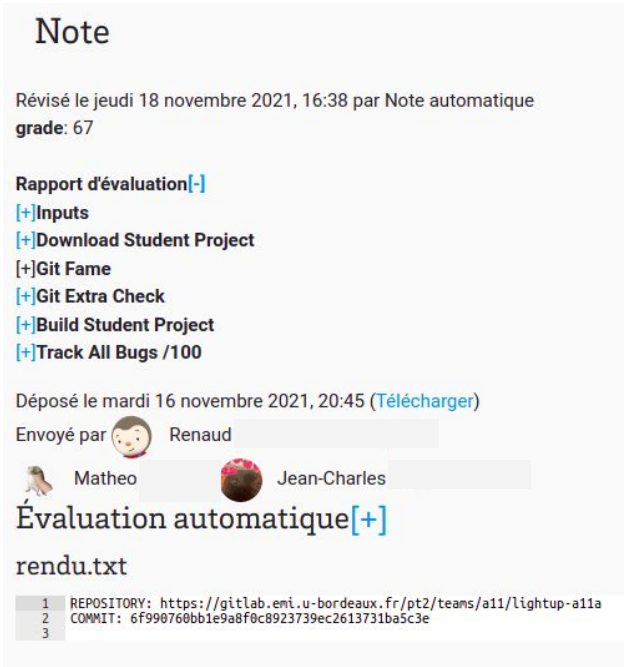
```
URL="https://gitlab.emi.u-bordeaux.fr"
TOKEN="XXXXXXXXXXXXX"       # personnal access token
PARENT_ID="pt2/teams/a11"   # parent group id
PROJECT_NAME="lightup-a11a"
VISIBILITY="private"

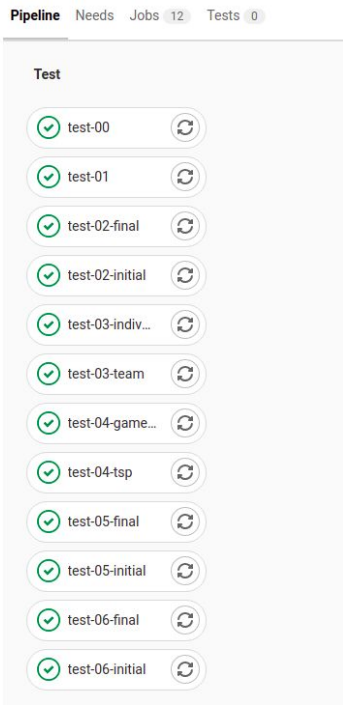
curl -s --request POST --url "$URL/api/v4/projects"                    \
        --header "content-type: application/json"                      \
        --header "PRIVATE-TOKEN: $TOKEN"                               \
        --data "{ \"name\": \"${PROJECT_NAME}\",                       \
                  \"namespace_id\": \"${PARENT_ID}\",                  \
                  \"visibility\": \"$VISIBILITY\",                     \
                  \"initialize_with_readme\": \"true\"                 \
                }"
```

Correction Automatique avec VPL

Dépôts : https://gitlab.inria.fr/pt2/vpl

- Juste rendre un *commit* dans VPL → ne pas coder dans VPL !
- Externaliser les scripts de correction automatique en dehors de Moodle / VPL
- Mise à jour / correction des scripts par un simple commit !
- Vérification automatique des scripts VPL en utilisant le Projet Enseignant
  - Le projet enseignant doit avoir 100% à toutes les évaluations !!!
  - Garantir la non-regression de la notation des étudiants !!!

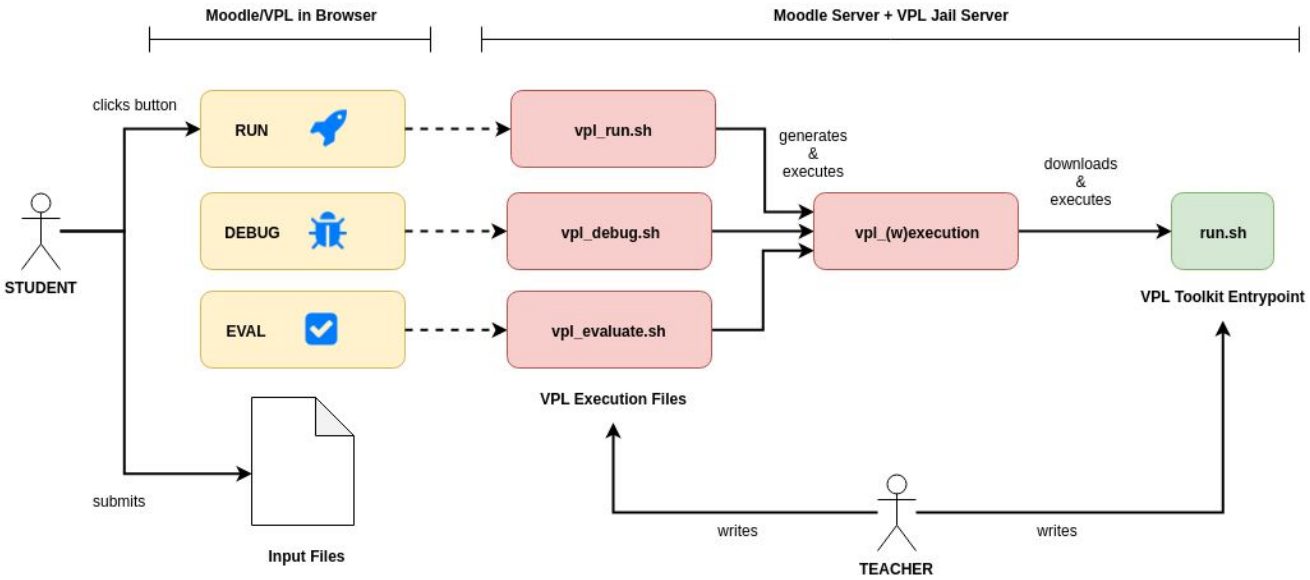




Correction Automatique avec VPL

Au moment du Run & Debug & Eval

- Récupération du projet étudiant (rendu.txt) → GitLab étudiant
- Récupération du script de correction automatique → GitLab VPL
- Exécution du script → feedback (commentaires, note)



Talk : https://github.com/orel33/vpltoolkit/blob/master/misc/talk-grenoble-2019.md

Bilan

Expérience en cours, pas encore le recul suffisant pour faire un bilan !

Tout n’est pas réutilisable facilement clé en main…

Attention au compromis sur le temps passer à automatiser certaines tâches, qui peut se révéler pas toujours gagnant…

⇒ Analyser la réutilsabilité d’une année sur l’autre…

Diverses Pistes à Explorer

Explorer les différentes possibilités de GitLab

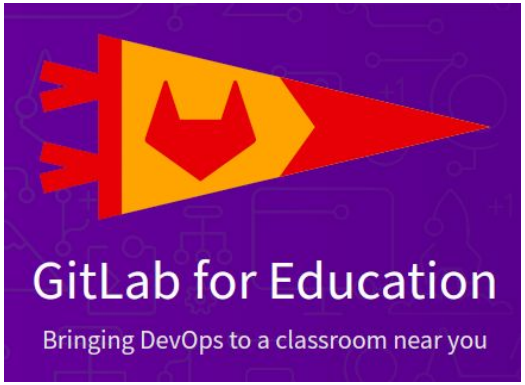
- Utilisation de GitLab CI/CD dans les projets étudiants
- Utilisation des GitLab Releases → Livraison du Projet (v1, v2)
- Utilisation des GitLab Issues + Board
- Utilisation de GitLab Code Review sur une *Merge Request*
- Installation de GitLab Pages et Docker au CREMI
- Aspect sécurité : utilisateur *Moodle Manager*, …

Appoche alternative : étudier les possibilités de *GitHub & Google Classroom*

- https://classroom.github.com
- https://classroom.google.com

Quid d'une solution "GitLab Classroom" ?

- https://about.gitlab.com/solutions/education



Annexes

Le Gitlab du CREMI

Comment récupérer son projet ?

Faire un clone en HTTPS ✘ (non supporté au CREMI)

```
$ git clone https://gitlab.emi.u-bordeaux.fr/auesnard/my-awsome-project.git
Cloning into 'bis'...
Username for 'https://gitlab.emi.u-bordeaux.fr': auesnard
Password for 'https://auesnard@gitlab.emi.u-bordeaux.fr': xxxxxxxxxx
remote: HTTP Basic: Access denied. Fatal: Authentication failed.
```

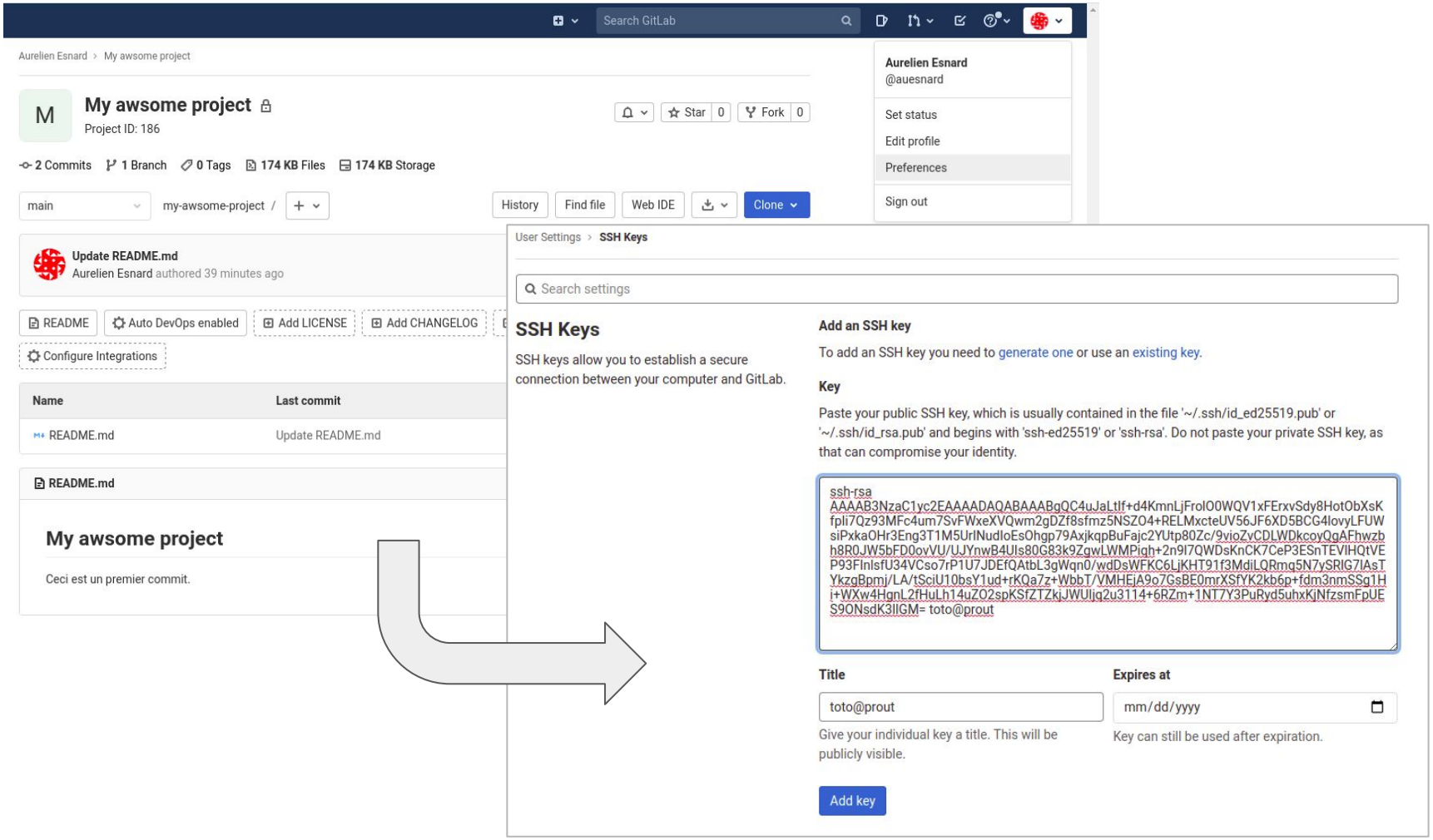
Faire un clone en SSH ✓

```
$ git clone git@gitlab.emi.u-bordeaux.fr:auesnard/my-awsome-project.git
Clonage dans 'my-awsome-project'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Réception d'objets: 100% (6/6), fait.
```

Prérequis : Il faut ajouter sa clé publique SSH ~/.ssh/id_rsa.pub dans les Préférences de son compte Gitlab.

Le Gitlab du CREMI

Ajoutez sa clé publique à son compte sur Gitlab



Le Gitlab du CREMI

Générer sa clé SSH, si ce n'est pas déjà fait...

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (~/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ~/.ssh/id_rsa
Your public key has been saved in ~/.ssh/id_rsa.pub # <-- la clé publique

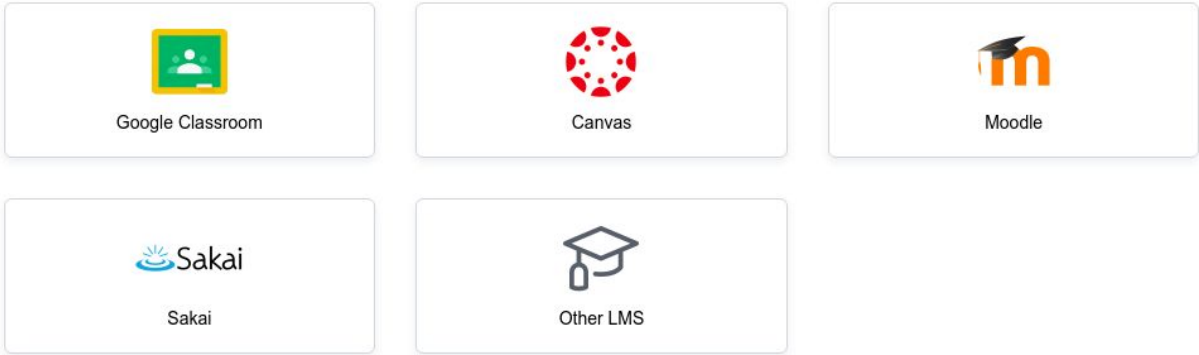
$ cat .ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC4uJaLtIf+d4KmnLjFrolO0WQV1xFErxvSdy8HotObXsKfpIi7Qz93MFc4um7
SvFWxeXVQwm2gDZf8sfmz5NSZO4+RELMxcteUV56JF6XD5BCG4lovyLFUWsiPxkaOHr3Eng3T1M5UrINudIoEsOhgp7
9AxjkqpBuFajc2YUtp80Zc/9vioZvCDLWDkcoyQgAFhwzbh8R0JW5bFD0ovVU/UJYnwB4UIs80G83k9ZgwLWMPiqh+2
n9l7QWDsKnCK7CeP3ESnTEVlHQtVEP93FInlsfU34VCso7rP1U7JDEfQAtbL3gWqn0/wdDsWFKC6LjKHT91f3MdiLQR
mq5N7ySRlG7lAsTYkzgBpmj/LA/tSciU10bsY1ud+rKQa7z+WbbT/VMHEjA9o7GsBE0mrXSfYK2kb6p+fdm3nmSSg1H
i+WXw4HgnL2fHuLh14uZO2spKSfZTZkjJWUIjq2u3114+6RZm+1NT7Y3PuRyd5uhxKjNfzsmFpUES9ONsdK3lIGM=
toto@prout
```

→ Faire un copier / coller du texte de la clé...

GitHub Classroom

https://classroom.github.com/

Connect a learning management system to GitHub Classroom. *You can configure an LTI-compliant Learning Management System (LMS) to connect to GitHub Classroom so that you can import a roster for your classroom.*



Dans Moodle > Ajout Activité “Outil Externe”... Faire le lien avec GitHub Classroom…

TODO : à compléter…