

Formation CI/CD



Présenté par Tom
27/11/23



Bon déjà, c'est quoi ?

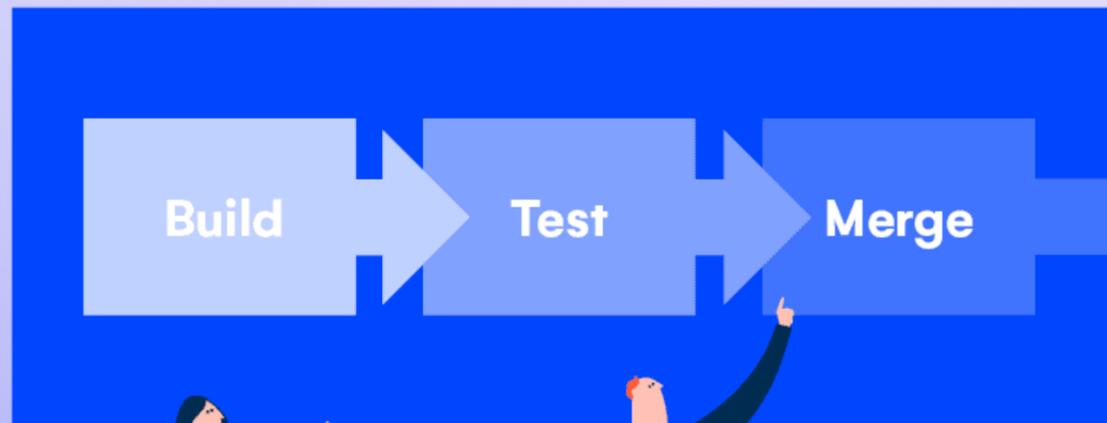


Le **CI/CD** ou **intégration continue / livraison continue**, est une pratique de développement logiciel rendue possible par l'**automatisation**. Des mises à jour fréquentes et fiables accélèrent les cycles de publication grâce à la livraison continue du code.

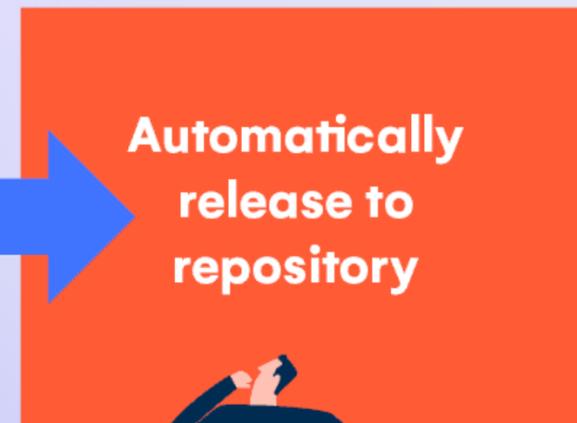


Bon déjà, c'est quoi ?

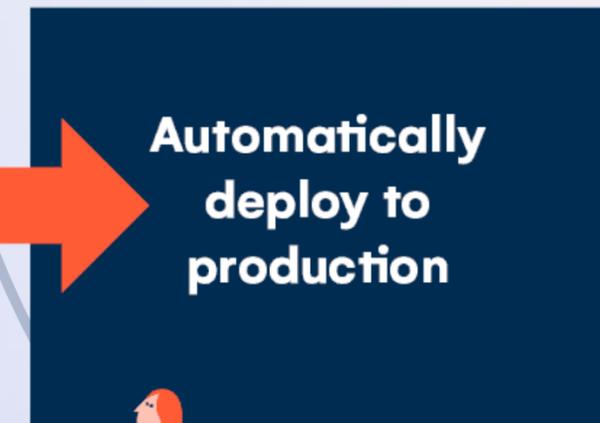
Continuous Integration



Continuous Delivery



Continuous Deployment



Comment on s'en sert ?



GitLab/GitHub

Tout se passe sur git, lorsque l'on travaille sur des projets **sensibles, complexes, en équipe ou simplement pour gagner du temps**, il va être important de mettre en place un CI/CD dans so projet gitlab via l'utilisation d'un fichier **.gitlab-ci.yml**.



Runners

Mais ou est-ce qu'on **exécute** tout notre code ? Il va falloir mettre en place des **runners** qui serviront d'environnement d'exécution de code pour **construire, tester, déployer** notre application.

Et en pratique ?



minet Search GitLab

Formation CI-CD

- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security and Compliance
- Deployments
- Packages and registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings

Pipelines
Editor
Jobs
Artifacts
Schedules

Tom Burellier > Formation CI-CD

Formation CI-CD Project ID: 66

21 Commits 1 Branch 0 Tags 72 KB Project Storage 1 Environment

Update file README.md Tom Burellier authored 1 day ago

main formation-ci-cd / +

Find file Web IDE Clone

README CI/CD configuration Add LICENSE Add CHANGELOG Add CONTRIBUTING Auto DevOps enabled Add Kubernetes cluster Add Wiki

Configure Integrations

Name	Last commit	Last update
.gitlab-ci.yml	Update file .gitlab-ci.yml	1 day ago
README.md	Update file README.md	1 day ago
data.txt	Update file data.txt	1 day ago
script.py	Add python script	1 day ago

README.md

Formation CI-CD

Un peu de documentation sur le CI/CD si vous voulez aller plus loin :)

bfabc213



Focus sur gitlab-ci.yml

```
👉 .gitlab-ci.yml
1
2 stages:
3   - build
4   - test
5   - deploy
6
7 build-job:
8   stage: build
9   script:
10    - echo "Compiling the code..."
11    - echo "Compile complete."
12
13 unit-test-job:
14   stage: test
15   script:
16    - echo "Running unit tests... This will take about 60 seconds."
17    - sleep 60
18    - echo "Code coverage is 90%"
19
20 lint-test-job:
21   stage: test
22   script:
23    - echo "Linting code... This will take about 10 seconds."
24    - sleep 10
25    - echo "No lint issues found."
26
27 deploy-job:
28   stage: deploy
29   environment: production
30   script:
31    - echo "Deploying application..."
32    - echo "Application successfully deployed."
33
```



Focus sur une pipeline



minet Search GitLab

Tom Burellier > My awesome project > Pipelines > #166

passed Pipeline #166 triggered in 10 seconds by Tom Burellier Delete

Update .gitlab-ci.yml file

🕒 4 jobs for `main`
in 1 minute and 13 seconds and was queued for 2 seconds

📄 latest

🔗 `03783cac`

🔗 No related merge requests found.

Pipeline Needs Jobs Tests

build	test	deploy
build-job	lint-test-job	deploy-job
	unit-test-job	



Focus sur un job



minet Search GitLab

My awesome project

Tom Burellier > My awesome project > Jobs > #895

passed Job **unit-test-job** triggered 1 minute ago by Tom Burellier

Search job log

```
Running with gitlab-runner 16.6.0 (3046fee8)
on gitlab-runner-dev JhVVn3aKq, system ID: s_68ba1c4ad5c1
3 Preparing "shell" executor
4 Using Shell (bash) executor...
6 Preparing environment
7 Running on gitlab-runner-dev...
9 Getting source from Git repository
10 Fetching changes with git depth set to 20...
11 Reinitialized existing Git repository in /home/gitlab-runner/builds/JhVVn3aKq/0/Balmine/my-awesome-project/.git/
12 Checking out 03783cac as detached HEAD (ref is main)...
13 Skipping Git submodules setup
15 Executing "step_script" stage of the job script
16 $ echo "Running unit tests... This will take about 60 seconds."
17 Running unit tests... This will take about 60 seconds.
18 $ sleep 60
19 $ echo "Code coverage is 90%"
20 Code coverage is 90%
22 Cleaning up project directory and file based variables
24 Job succeeded
```

unit-test-job

Duration: 1 minute 0 seconds
Finished: just now
Queued: 1 second
Timeout: 10m (from runner)
Runner: #4 (JhVVn3aKq) Shared-runner

Commit 03783cac
Update .gitlab-ci.yml file

Pipeline #166 for main

test

lint-test-job

→ unit-test-job



Focus sur les runners

The screenshot shows the GitLab interface for a project named "My awesome project". The left sidebar contains navigation options, with "Settings" and "CI/CD" highlighted by red arrows. The main content area is titled "Runners" and includes a "Collapse" button. It explains that runners are processes that execute CI/CD jobs and lists their states: "active" (available) and "paused" (not available). It also mentions tags for job filtering. The page is divided into three sections: "Project runners" (circled in red), "Shared runners" (circled in red), and "Group runners". The "Project runners" section shows a "New project runner" button and a table of "Other available runners" with columns for runner ID, tags, and an "Enable for this project" button. The "Shared runners" section shows a toggle for enabling shared runners and a list of available shared runners, including runner #4 with tags "shared-runner" and "shared-runner". The "Group runners" section notes that the project does not belong to a group and cannot use group runners. A code block on the right shows a snippet of a .gitlab-ci.yml file:

```
txt-test-job: # Ce j
  stage: tests # IT
  tags: [production]
  script:
    - if ! grep -q "pr
```

Exemples d'utilisation à

```
57
58 deploy: # This job runs in the test stage.
59   stage: deploy # It only starts when the job in the build stage completes successfully.
60   needs:
61     - unit-test
62   script:
63     - sudo rm -r /etc/nginx/sites-available/
64     - cd /etc/nginx/
65     - sudo mkdir sites-available/
66     - cd -
67     - sudo cp -r sites-available/* /etc/nginx/sites-available/
68     - cd /etc/nginx/
69     - sudo rm /etc/nginx/sites-enabled/*
70     - sudo ln -s /etc/nginx/sites-available/* sites-enabled/
71     - sudo nginx -s reload
72   tags:
73     - runner-rp1
74
75
76
77 deploy2: # This job runs in the test stage.
78   stage: deploy # It only starts when the job in the build stage completes successfully.
79   needs:
80     - unit-test
81   script:
82     - sudo rm -r /etc/nginx/sites-available/
83     - cd /etc/nginx/
84     - sudo mkdir sites-available/
85     - cd -
86     - sudo cp -r sites-available/* /etc/nginx/sites-available/
87     - cd /etc/nginx/
88     - sudo rm /etc/nginx/sites-enabled/*
89     - sudo ln -s /etc/nginx/sites-available/* sites-enabled/
90     - sudo nginx -s reload
91   tags:
92     - runner-rp2
93
```

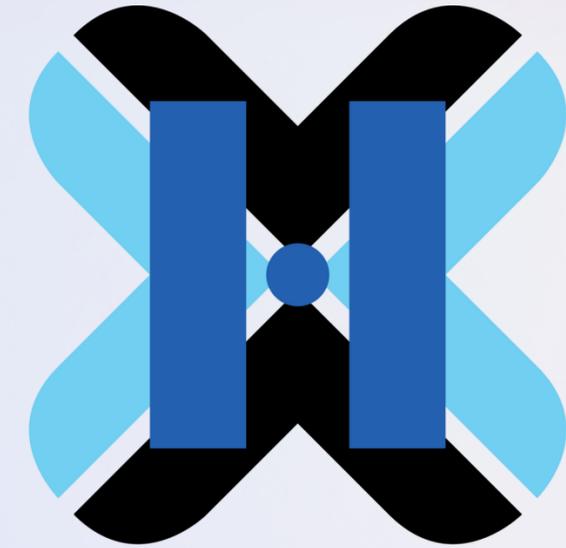
reverse proxy

NGINX

100 lignes de CI/CD

Exemples d'utilisation à

```
48 test_backend:
49   stage: test
50   image: python:3.9
51   needs :
52   - python_code_quality
53   before_script:
54   - export http_proxy="$HTTP_PROXY"
55   - export https_proxy="$HTTPS_PROXY"
56   - export no_proxy="$NO_PROXY"
57   - apt update
58   - apt install sqlite3
59   - export ENVIRONMENT='TEST'
60   - export KEYRING_DNS_SECRET=$KEYRING_DNS_SECRET
61   - export PROXMOX_API_KEY_NAME=$PROXMOX_API_KEY_NAME
62   - export PROXMOX_API_KEY=$PROXMOX_API_KEY
63   - export PROXMOX_BACK_DB_DEV=$PROXMOX_BACK_DB_DEV
64   - export ADH6_API_KEY=$ADH6_API_KEY
65   - cd backend/
66   - export PYTHONPATH=$(pwd)
67   - pip3 install -r requirements.txt
68   - pip3 install -r test-requirements.txt
69   script :
70   - pytest --cov=proxmox_api --cov-report term-missing --cov-report xml:./coverage_out_report.xml --junitxml=report.xml
71   coverage: '/TOTAL.*\s+(\d+)%$/'
72   artifacts:
73     when: always
74     paths:
75     - backend/report.xml
76     reports:
77     junit: backend/report.xml
78
79
```



HOSTING

200 lignes de CI/CD

Exemples d'utilisation à

```
182 #####
183 ##### Deploy in staged #####
184 #####
185 staged:frontend:
186   stage: deploy
187   rules:
188     - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
189   when: manual
190   needs:
191     - lint:frontend
192     - generation:frontend_module
193     - generation:frontend_assets
194     - build:frontend
195   script:
196     - cp -r frontend_angular/dist/adh6/* /var/www/
197     - cp -r frontend_angular/src/assets /var/www/assets
198   environment:
199     name: staged
200     url: https://adh6-staged.minet.net/
201   tags:
202     - adh6-staged
203
204 staged:backend:
205   stage: deploy
206   rules:
207     - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
208   when: manual
209   needs:
210     - lint:backend
211     - test:backend
212   before_script:
213     - sudo systemctl stop emperor.uwsgi.service
214     - export http_proxy=$HTTP_PROXY
215     - export https_proxy=$HTTPS_PROXY
216     - cd api_server/
217     - sudo -E pip3 install -r ./requirements.txt
218   script:
219     - /usr/local/bin/mibdump.py --mib-source=mibs.zip CISCO-VLAN-MEMBERSHIP-MIB IF-MIB CISCO-MAC-AUTH-BYPASS-MIB
220     - wget -O - -o /dev/null http://standards-oui.ieee.org/oui.txt | grep '(hex)' | sed -E 's/\/s*?(hex)\s+\/t/' > OUIs.txt
221     - cp ../openapi/spec.yaml ../openapi/spec.yaml
222     - export FLASK_APP=manage:manager
223     - ENVIRONMENT=production flask db upgrade
224     - cat uwsgi-prod.template | envsubst | tee uwsgi-prod.ini
225     - sudo rm /etc/uwsgi/vassals/adh6_api.ini
226     - sudo ln -s $(pwd)/uwsgi-prod.ini /etc/uwsgi/vassals/adh6_api.ini
227     - sudo systemctl start emperor.uwsgi.service
228   environment:
229     name: staged
230     url: https://adh6-staged.minet.net/
231   tags:
232     - adh6-staged
233
```



300 lignes de CI/CD

Maintenant, TP