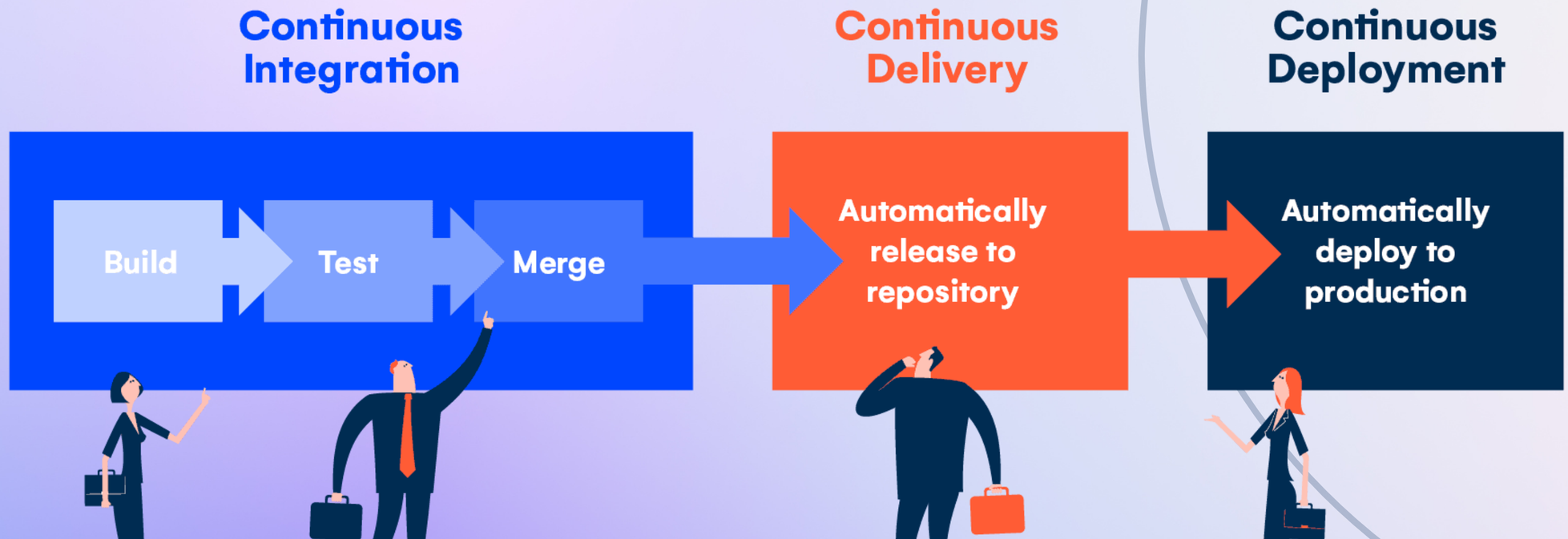
Formation CI/CD

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 ?



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’éxécution de code pour **construire, tester, déployer** notre application.

Et en pratique ?

Search GitLab /

F 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

CI/CD
- Pipelines
- Editor
- Jobs
- Artifacts
- Schedules

Tom Burellier › Formation CI-CD

F **Formation CI-CD**
Project ID: 66

Star 0 Fork 0

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

Update file README.md
Tom Burellier authored 1 day ago
bfabc213

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

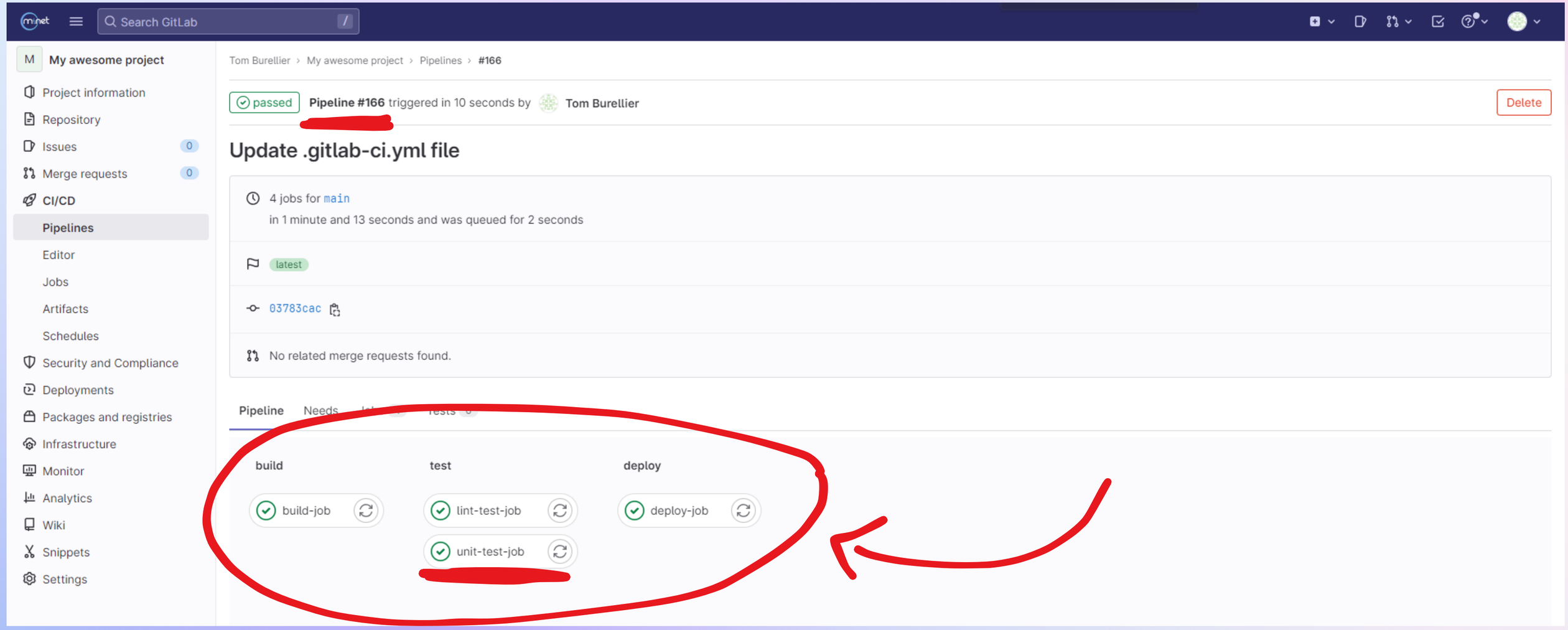
Un peu de documentation sur le CI/CD si vous voulez aller plus loin :)

Focus sur gitlab-ci.yml

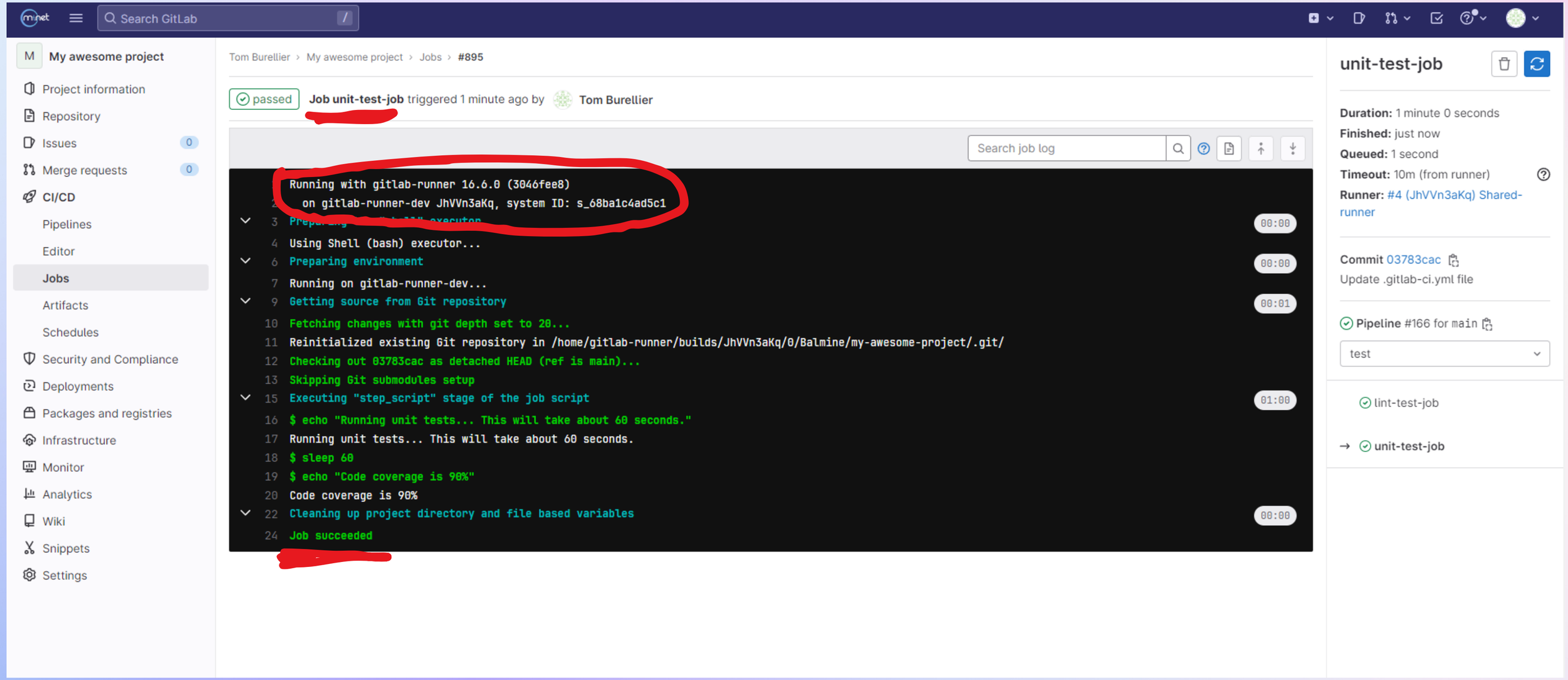
.gitlab-ci.yml

```yaml

stages:
  - build
  - test
  - deploy

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

unit-test-job:
  stage: test
  script:
    - echo "Running unit tests... This will take about 60 seconds."
    - sleep 60
    - echo "Code coverage is 90%"

lint-test-job:
  stage: test
  script:
    - echo "Linting code... This will take about 10 seconds."
    - sleep 10
    - echo "No lint issues found."

deploy-job:
  stage: deploy
  environment: production
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
```

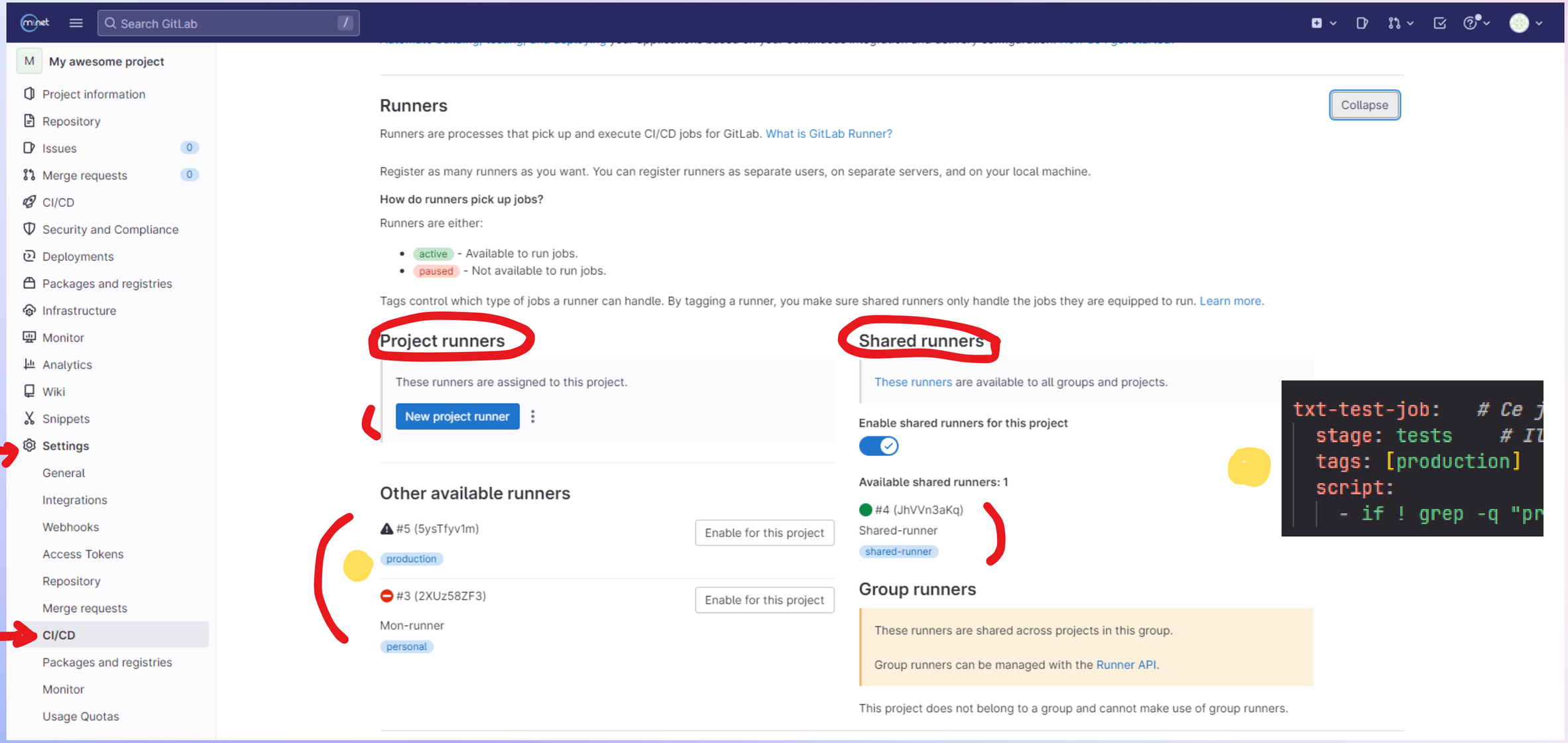
Focus sur une pipeline



Focus sur un job



Focus sur les runners



Exemples d'utilisation à

```
deploy:   # This job runs in the test stage.
  stage: deploy   # It only starts when the job in the build stage completes successfully.
  needs:
    - unit-test
  script:
    - sudo rm -r /etc/nginx/sites-available/
    - cd /etc/nginx/
    - sudo mkdir sites-available/
    - cd -
    - sudo cp -r sites-available/* /etc/nginx/sites-available/
    - cd /etc/nginx/
    - sudo rm  /etc/nginx/sites-enabled/*
    - sudo ln -s /etc/nginx/sites-available/* sites-enabled/
    - sudo nginx -s reload
  tags:
    - runner-rp1


deploy2:   # This job runs in the test stage.
  stage: deploy   # It only starts when the job in the build stage completes successfully.
  needs:
    - unit-test
  script:
    - sudo rm -r /etc/nginx/sites-available/
    - cd /etc/nginx/
    - sudo mkdir sites-available/
    - cd -
    - sudo cp -r sites-available/* /etc/nginx/sites-available/
    - cd /etc/nginx/
    - sudo rm  /etc/nginx/sites-enabled/*
    - sudo ln -s /etc/nginx/sites-available/* sites-enabled/
    - sudo nginx -s reload
  tags:
    - runner-rp2
```

reverse proxy

NGINX

100 lignes de CI/CD

Exemples d’utilisation à

```yaml
test_backend:
  stage: test
  image: python:3.9
  needs :
   - python_code_quality
  before_script:
    - export http_proxy="$HTTP_PROXY"
    - export https_proxy="$HTTPS_PROXY"
    - export no_proxy="$NO_PROXY"
    - apt update
    - apt install sqlite3
    - export ENVIRONMENT='TEST'
    - export KEYRING_DNS_SECRET=$KEYRING_DNS_SECRET
    - export PROXMOX_API_KEY_NAME=$PROXMOX_API_KEY_NAME
    - export PROXMOX_API_KEY=$PROXMOX_API_KEY
    - export PROXMOX_BACK_DB_DEV=$PROXMOX_BACK_DB_DEV
    - export ADH6_API_KEY=$ADH6_API_KEY
    - cd backend/
    - export PYTHONPATH=$(pwd)
    - pip3 install -r requirements.txt
    - pip3 install -r test-requirements.txt
  script :
    - pytest --cov=proxmox_api --cov-report term-missing --cov-report xml:./coverage_out_report.xml --junitxml=report.xml
  coverage: '/TOTAL.*\s+(\d+%)$/'
  artifacts:
    when: always
    paths:
      - backend/report.xml
    reports:
      junit: backend/report.xml

```

HOSTING

200 lignes de CI/CD

Exemples d'utilisation à


```yaml
########################################
######### Deploy in stagged ############
########################################
stagged:frontend:
  stage: deploy
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
  when: manual
  needs:
    - lint:frontend
    - generation:frontend_module
    - generation:frontend_assets
    - build:frontend
  script:
    - cp -r frontend_angular/dist/adh6/* /var/www/
    - cp -r frontend_angular/src/assets /var/www/assets
  environment:
    name: stagged
    url: https://adh6-stagged.minet.net/
  tags:
    - adh6-stagged

stagged:backend:
  stage: deploy
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
  when: manual
  needs:
    - lint:backend
    - test:backend
  before_script:
    - sudo systemctl stop emperor.uwsgi.service
    - export http_proxy=$HTTP_PROXY
    - export https_proxy=$HTTPS_PROXY
    - cd api_server/
    - sudo -E pip3 install -r ./requirements.txt
  script:
    - /usr/local/bin/mibdump.py --mib-source=mibs.zip CISCO-VLAN-MEMBERSHIP-MIB IF-MIB CISCO-MAC-AUTH-BYPASS-MIB
    - wget -O - -o /dev/null http://standards-oui.ieee.org/oui.txt | grep '(hex)' | sed -E 's/\s*?\(hex\)\s+/\t/' > OUIs.txt
    - cp ../openapi/spec.yaml ./openapi/spec.yaml
    - export FLASK_APP=manage:manager
    - ENVIRONMENT=production flask db upgrade
    - cat uwsgi-prod.template | envsubst | tee uwsgi-prod.ini
    - sudo rm /etc/uwsgi/vassals/adh6_api.ini
    - sudo ln -s $(pwd)/uwsgi-prod.ini /etc/uwsgi/vassals/adh6_api.ini
    - sudo systemctl start emperor.uwsgi.service
  environment:
    name: stagged
    url: https://adh6-stagged.minet.net/
  tags:
    - adh6-stagged

```

300 lignes de CI/CD

Maitenant, TP