Example CI/CD process with GitLab

Many developers use GitLab to store code and work together on one project (write and review code, make issues, user management, etc.). But it has many other built-in tools includes automate test and deploy process. With Gitlab you can easily build, test and deploy your code.

Official documentation

- GitLab CI/CD Documentation
- Getting started with GitLab CI/CD
- Configuration of your jobs with .gitlab-ci.yml

GitLab CI/CD. Main concepts and terms

If you want to use Continuous Integration (CI) and Continuous Delivery (CD) services you need to know basic terms using in GitLab. I selected the foremost ones and set links to the description and additional information about them.

- Configuration file:

For using a CI/CD service you need to create a .gitlab-ci.yml file to the root directory of your repository. This file is used by GitLab Runner to manage your project's jobs and stored in a YAML format.

- Runner:

GitLab Runner is a daemon in a host machine that is used to running your jobs and send the results back to GitLab. It permanently holds a connect with Gitlab. When a user runs job by pipeline Runner executes commands from .gitlab-ci.yml on the host.

Example CI/CD process with GitLab

- Job:

Jobs are set of commands that stored in section script in a config file. GitLab Runner runs jobs in a host machine. Each job is run independently of each other. The most popular jobs are a build_web_app1, build_web_app2, prepare, test, deploy, and etc.

- Stage:

A stage allows to group jobs, and jobs of the same stage are executed in parallel. Jobs of the next stage are run after the jobs from the previous stage complete successfully.

- Pipeline:

A pipeline is a group of jobs that get executed in stages (batches). All of the jobs in a stage are executed in parallel and if they all succeed the pipeline moves on to the next stage. If one of the jobs fails the next stage is not (usually) executed. You can access the pipelines page in your project's Pipelines tab.

- Artifact:

An artifact is a list of files and directories which are attached to a job after it completes successfully. The uploaded artifacts will be kept in GitLab during expiry period. You can download the artifacts archive or browse its contents in Job Info page.

- Dependency:

When defined jobs exist in a dependency block of your .gitlab-ci.yml the Runner should be download all artifacts before start the current job. It can be used to divide build and deploy jobs for example. The deploy job often uses generated artifacts on previous stages.

You find other terms using in Gitlab in the following URL: https://blog.eleven-labs.com/fr/introduction-gitlab-ci/

Example CI/CD process with GitLab

GitLab CI/CD. Create simple pipeline

With this example, i want to show you how to build and run a simple web application with pipelines. I wrote an example spring boot application to demonstrate a possibility of CI. The application code is placed here: https://gitlab.com/bileli/gitlab-cicd.git

For using it in GitLab just clone and add as new project.

GitLab CI/CD. Install and configure Runner

In GitLab Runners run the jobs that you define in .gitlab-ci.yml. A Runner can be a virtual machine, a VPS, a bare-metal machine, a docker container or even a cluster of containers. GitLab and the Runners communicate through an API, so the only requirement is that the Runner's machine has Internet access.

The official Runner supported by GitLab is written in Go and its documentation can be found at https://docs.gitlab.com/runner/.

In order to have a functional Runner you need to follow two steps:

1. Install it
2. Configure it

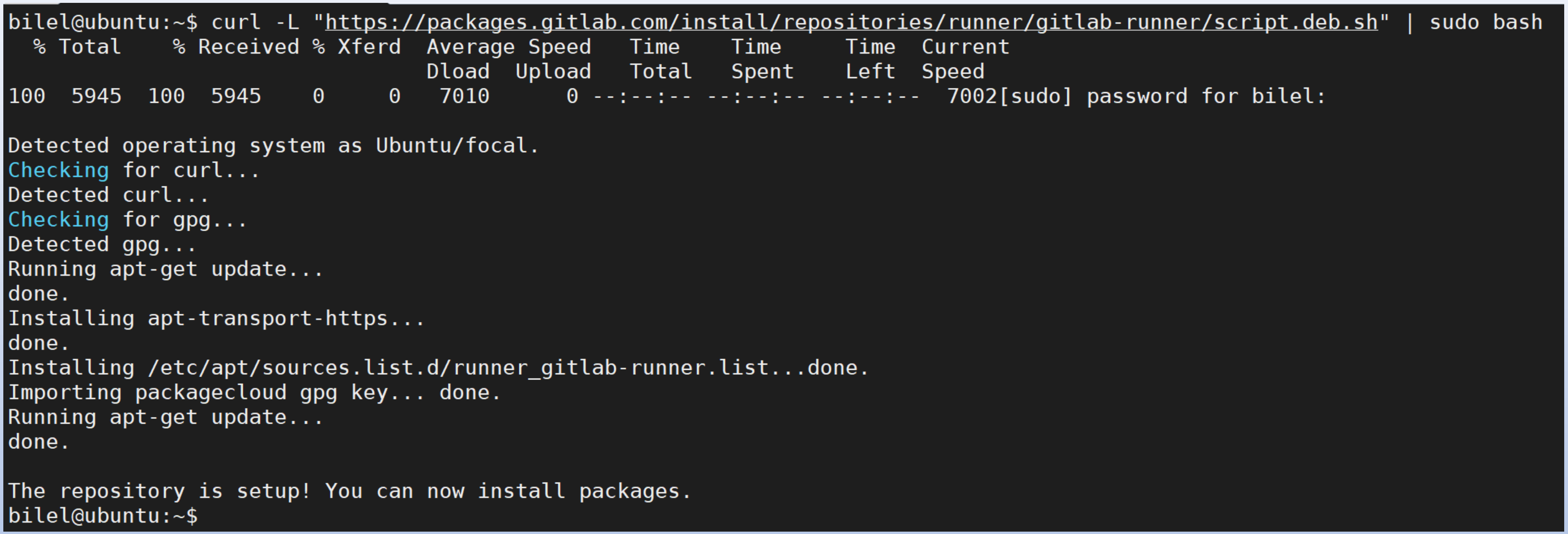
Next I'll show you how to install and configure the latest GitLab Runner with shell executor in Ubuntu system. Additional steps you can find in official docs.

Example CI/CD process with GitLab

1- Install Runner

- Add the official GitLab repository:

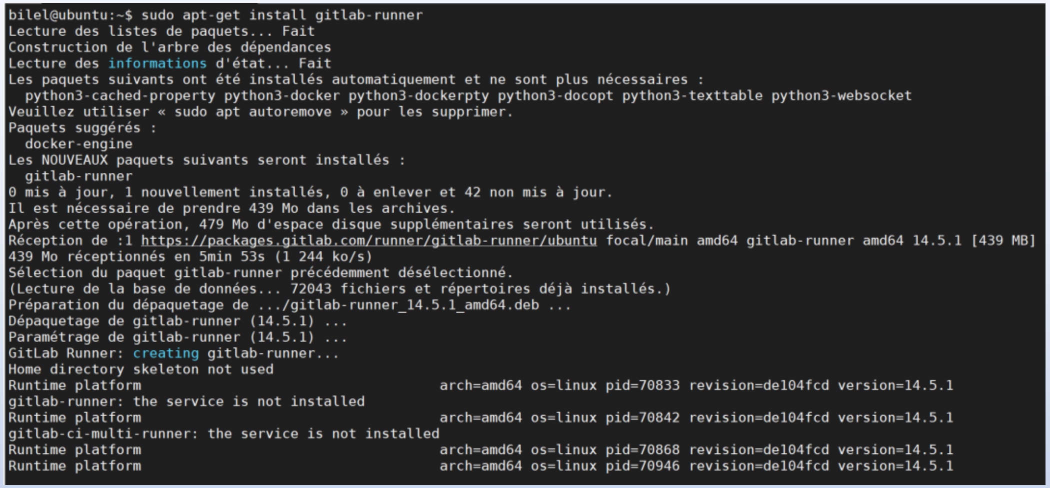
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash



```
bilel@ubuntu:~$ curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5945  100  5945    0     0  7010      0 --:--:-- --:--:-- --:--:--  7002[sudo] password for bilel:

Detected operating system as Ubuntu/focal.
Checking for curl...
Detected curl...
Checking for gpg...
Detected gpg...
Running apt-get update...
done.
Installing apt-transport-https...
done.
Installing /etc/apt/sources.list.d/runner_gitlab-runner.list...done.
Importing packagecloud gpg key... done.
Running apt-get update...
done.

The repository is setup! You can now install packages.
bilel@ubuntu:~$
```

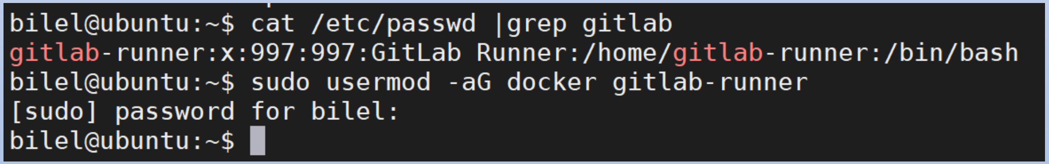
Example CI/CD process with GitLab

- Install the latest version of GitLab Runner:
sudo apt-get install gitlab-runner



```
bilel@ubuntu:~$ sudo apt-get install gitlab-runner
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  python3-cached-property python3-docker python3-dockerpty python3-docopt python3-texttable python3-websocket
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
Paquets suggérés :
  docker-engine
Les NOUVEAUX paquets suivants seront installés :
  gitlab-runner
0 mis à jour, 1 nouvellement installés, 0 à enlever et 42 non mis à jour.
Il est nécessaire de prendre 439 Mo dans les archives.
Après cette opération, 479 Mo d'espace disque supplémentaires seront utilisés.
Réception de :1 https://packages.gitlab.com/runner/gitlab-runner/ubuntu focal/main amd64 gitlab-runner amd64 14.5.1 [439 MB]
439 Mo réceptionnés en 5min 53s (1 244 ko/s)
Sélection du paquet gitlab-runner précédemment désélectionné.
(Lecture de la base de données... 72043 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../gitlab-runner_14.5.1_amd64.deb ...
Dépaquetage de gitlab-runner (14.5.1) ...
Paramétrage de gitlab-runner (14.5.1) ...
GitLab Runner: creating gitlab-runner...
Home directory skeleton not used
Runtime platform                                    arch=amd64 os=linux pid=70833 revision=de104fcd version=14.5.1
gitlab-runner: the service is not installed
Runtime platform                                    arch=amd64 os=linux pid=70842 revision=de104fcd version=14.5.1
gitlab-ci-multi-runner: the service is not installed
Runtime platform                                    arch=amd64 os=linux pid=70868 revision=de104fcd version=14.5.1
Runtime platform                                    arch=amd64 os=linux pid=70946 revision=de104fcd version=14.5.1
```

- Runner host preparing:
add a gitlab-runner user into docker group.
sudo usermod –aG docker gitlab-runner

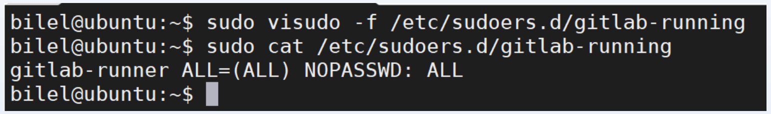


```
bilel@ubuntu:~$ cat /etc/passwd |grep gitlab
gitlab-runner:x:997:997:GitLab Runner:/home/gitlab-runner:/bin/bash
bilel@ubuntu:~$ sudo usermod -aG docker gitlab-runner
[sudo] password for bilel:
bilel@ubuntu:~$
```

Example CI/CD process with GitLab

In addition I setup gitlab-runner as sudo user to run protected linux commands without user passwords. Just add a next row into a file with a command

sudo visudo –f /etc/sudoers.d/gitlab-runner



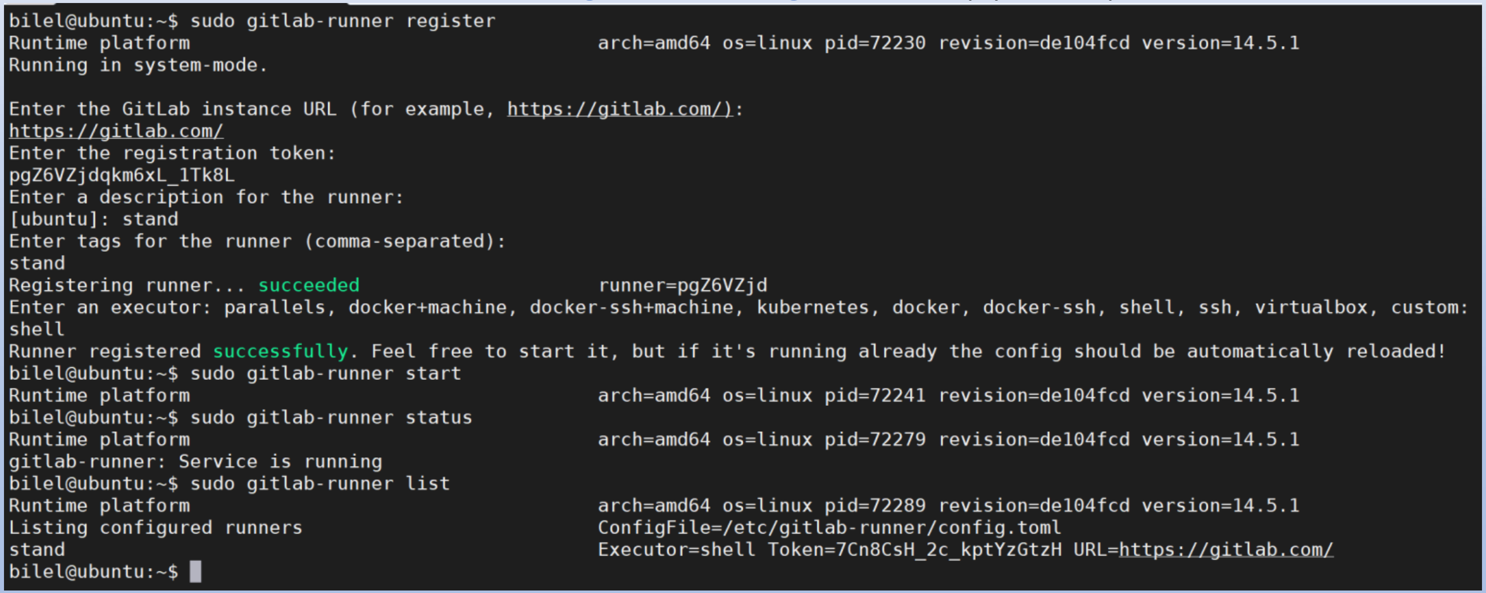
```
bilel@ubuntu:~$ sudo visudo -f /etc/sudoers.d/gitlab-running
bilel@ubuntu:~$ sudo cat /etc/sudoers.d/gitlab-running
gitlab-runner ALL=(ALL) NOPASSWD: ALL
bilel@ubuntu:~$ 
```

2- Configure Runner

- Register Runner:

Registering a Runner is the process that binds the Runner with a GitLab instance.

You can register a Runner in interactive mode with a command: $ “gitlab-runner register” And reply to the questions of the wizard.



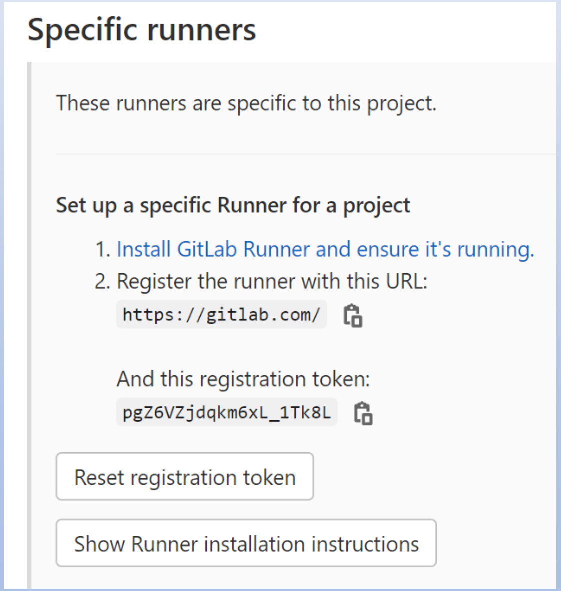
```
bilel@ubuntu:~$ sudo gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=72230 revision=de104fcd version=14.5.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com/
Enter the registration token:
pgZ6VZjdqkm6xL_1Tk8L
Enter a description for the runner:
[ubuntu]: stand
Enter tags for the runner (comma-separated):
stand
Registering runner... succeeded                     runner=pgZ6VZjd
Enter an executor: parallels, docker+machine, docker-ssh+machine, kubernetes, docker, docker-ssh, shell, ssh, virtualbox, custom:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
bilel@ubuntu:~$ sudo gitlab-runner start
Runtime platform                                    arch=amd64 os=linux pid=72241 revision=de104fcd version=14.5.1
bilel@ubuntu:~$ sudo gitlab-runner status
Runtime platform                                    arch=amd64 os=linux pid=72279 revision=de104fcd version=14.5.1
gitlab-runner: Service is running
bilel@ubuntu:~$ sudo gitlab-runner list
Runtime platform                                    arch=amd64 os=linux pid=72289 revision=de104fcd version=14.5.1
Listing configured runners                          ConfigFile=/etc/gitlab-runner/config.toml
stand                                               Executor=shell Token=7Cn8CsH_2c_kptYzGtzH URL=https://gitlab.com/
bilel@ubuntu:~$ 
```

Example CI/CD process with GitLab

You can also register a runner with inserting answers in parameters:

```
gitlab-runner register -n \
  --url "https://YOUR_GITLAB_URL/" \
  --registration-token "XXXXXXXXXXXXXXXXXXXXXXX" \
  --description "nginx" \
  --tag-list "nginx-shell" \
  --executor "shell" \
```

Don't forget place your token. It stored in: Setting ➔ CI/CD ➔ Runners



Specific runners

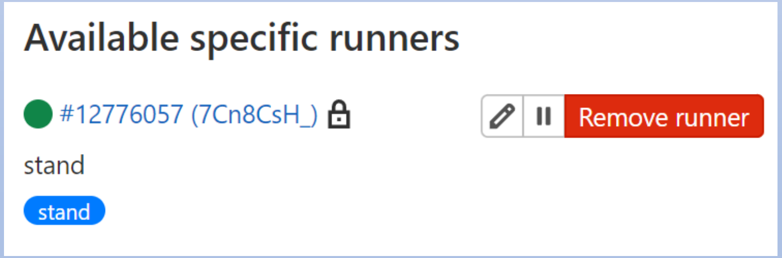
These runners are specific to this project.

Set up a specific Runner for a project

1. Install GitLab Runner and ensure it's running.
2. Register the runner with this URL:
   `https://gitlab.com/`

   And this registration token:
   `pgZ6VZjdqkm6xL_1Tk8L`

Once the Runner has been set up, you should see it on the Runners page of your project, following Settings ➔ CI/CD:



Available specific runners

● #12776057 (7Cn8CsH_)

stand

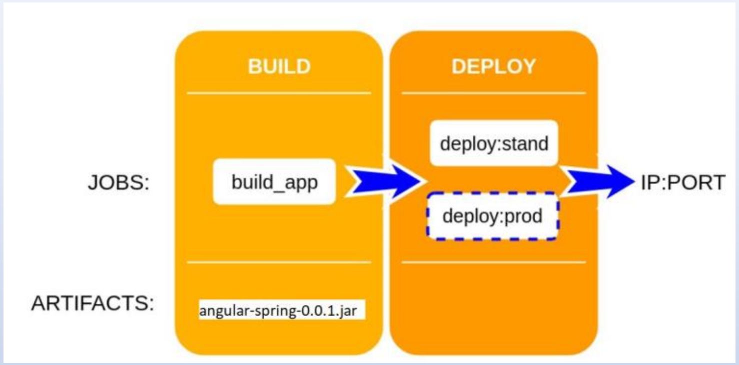
stand

Example CI/CD process with GitLab

After that you can configure your CI process in a .gitlab-ci.yml file with using tags to select specific Runners.

Writing pipeline

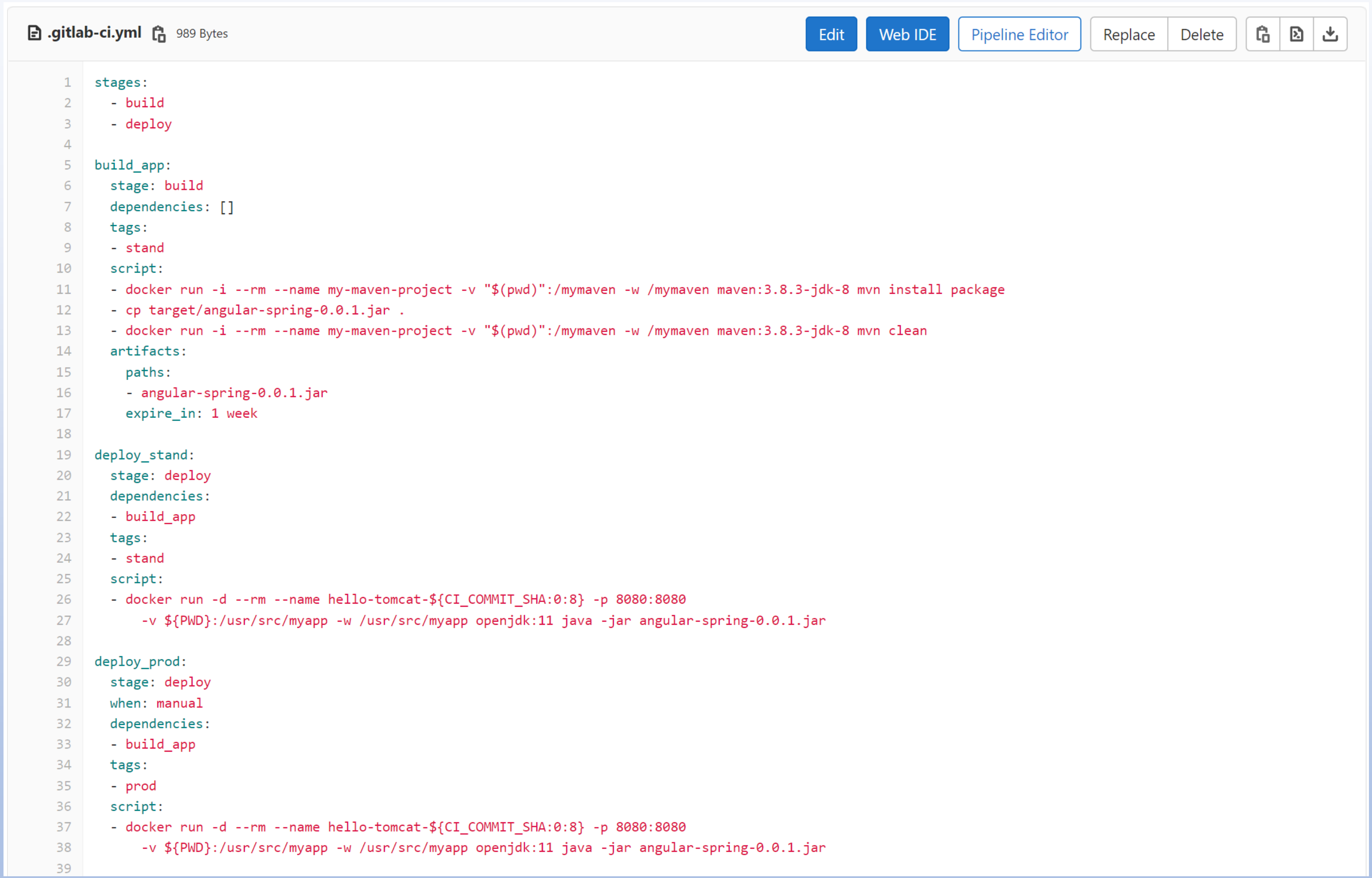
In this project I want to use two stages to build and deploy an application.



The stage deploy has two jobs to deploy the app into a stand and production servers. Runners on these servers have tags prod and stand. The deploy to the production server requires a manual action. To build an app are used a docker maven container. The app is deploying with openjdk container.

Thus the next code will configure the defined pipeline. It requires to be inputted in .gitlab-ci.yml

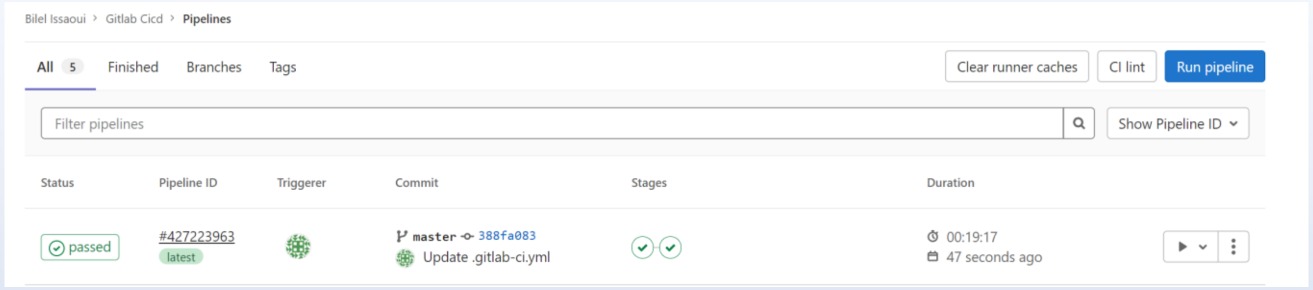
Example CI/CD process with GitLab



```yaml
stages:
  - build
  - deploy

build_app:
  stage: build
  dependencies: []
  tags:
  - stand
  script:
  - docker run -i --rm --name my-maven-project -v "$(pwd)":/mymaven -w /mymaven maven:3.8.3-jdk-8 mvn install package
  - cp target/angular-spring-0.0.1.jar .
  - docker run -i --rm --name my-maven-project -v "$(pwd)":/mymaven -w /mymaven maven:3.8.3-jdk-8 mvn clean
  artifacts:
    paths:
    - angular-spring-0.0.1.jar
    expire_in: 1 week

deploy_stand:
  stage: deploy
  dependencies:
  - build_app
  tags:
  - stand
  script:
  - docker run -d --rm --name hello-tomcat-${CI_COMMIT_SHA:0:8} -p 8080:8080
      -v ${PWD}:/usr/src/myapp -w /usr/src/myapp openjdk:11 java -jar angular-spring-0.0.1.jar

deploy_prod:
  stage: deploy
  when: manual
  dependencies:
  - build_app
  tags:
  - prod
  script:
  - docker run -d --rm --name hello-tomcat-${CI_COMMIT_SHA:0:8} -p 8080:8080
      -v ${PWD}:/usr/src/myapp -w /usr/src/myapp openjdk:11 java -jar angular-spring-0.0.1.jar

```

Example CI/CD process with GitLab

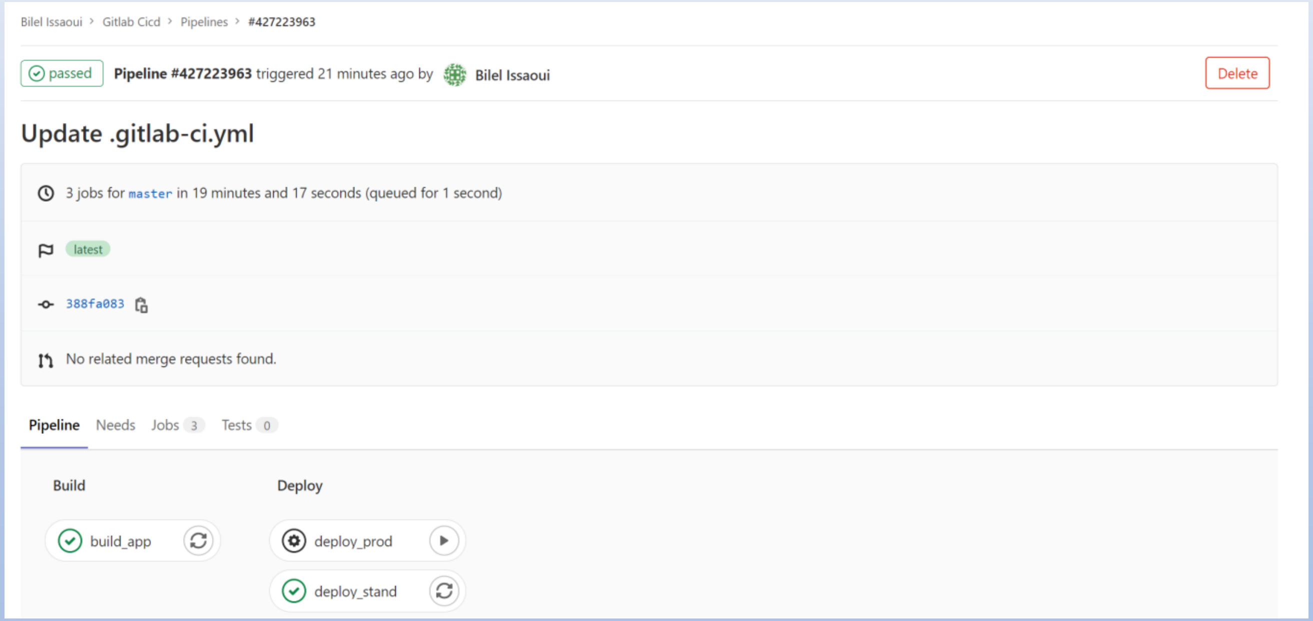
In the result we have an automation process that builds and deploy the web app. The app is deployed to stand host for test purposes after every commit into a repo.



Bilel Issaoui > Gitlab Cicd > Pipelines

Status	Pipeline ID	Triggerer	Commit	Stages	Duration
passed	#427223963 latest		master 388fa083 Update .gitlab-ci.yml	✓ ✓	00:19:17 47 seconds ago

The result pipeline is:



Bilel Issaoui > Gitlab Cicd > Pipelines > #427223963

passed **Pipeline #427223963** triggered 21 minutes ago by Bilel Issaoui

Update .gitlab-ci.yml

3 jobs for master in 19 minutes and 17 seconds (queued for 1 second)

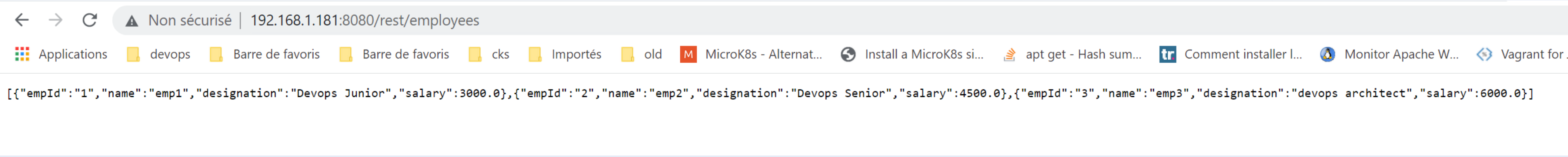
latest

388fa083

No related merge requests found.

Example CI/CD process with GitLab

the web app can be accessed by ip address of host stand and port 8080



```
[{"empId":"1","name":"emp1","designation":"Devops Junior","salary":3000.0},{"empId":"2","name":"emp2","designation":"Devops Senior","salary":4500.0},{"empId":"3","name":"emp3","designation":"devops architect","salary":6000.0}]
```

With this configuration you have tried to use CI/CD process with GitLab. Don’t forget this simple pipeline doesn’t provide jobs to stop stands. You should are stopping unused docker containers yourself.
Next you can change it to your complex configuration for own purposes.