

DevOps in the clouds - the technological blog of Patrycjusz Czerniga



docker
Compose

HOW TO INSTALL GITLAB USING DOCKER COMPOSE?

How to install GitLab using Docker Compose?

Published by [Patrycjusz Czerniga](#) on [November 14, 2021](#)

Hi. Today I will show you how you can install the GitLab server within an hour and run your first CI/CD process in it.

This article is part of a series on how to get started with the popular CI/CD tools. In this article, I will show you how to install the CI/CD tool and how to prepare the process of building and testing a simple project based on Maven.

What is GitLab?

Gitlab is a tool supporting software development using the Continuous Integration and Continuous Delivery processes. Its main component is the Git version control system. In addition, it provides many functions that support the work of programmers in the continuous building, testing and automatic installation of projects for various environments.

What is Docker Compose?

To find out what Docker Compose is, go to the article: [How to install Jenkins using Docker Compose?](#)

Tools required

Before starting work, prepare the required tools. You can find a description of these tools on the website: [How to install Jenkins using Docker Compose?](#)

Configuration for Docker Compose

We will start work on the installation by creating a dedicated directory in which we will store data and Gitlab configuration.

```
> mkdir gitlab
```

For convenience, we will also set an environment variable that will contain the path to our Gitlab directory:

```
> export GITLAB_HOME=$(pwd)/gitlab
```

In the next step, we create the docker-compose.yml file with the following content:

```
# docker-compose.yml
version: '3.7'
services:
  web:
    image: 'gitlab/gitlab-ce:latest'
    restart: always
    hostname: 'localhost'
    container_name: gitlab-ce
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://localhost'
    ports:
      - '8080:80'
      - '8443:443'
    volumes:
      - '$GITLAB_HOME/config:/etc/gitlab'
      - '$GITLAB_HOME/logs:/var/log/gitlab'
```

```
- '$GITLAB_HOME/data:/var/opt/gitlab'
networks:
  - gitlab
gitlab-runner:
  image: gitlab/gitlab-runner:alpine
  container_name: gitlab-runner
  restart: always
  depends_on:
    - web
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - '$GITLAB_HOME/gitlab-runner:/etc/gitlab-runner'
  networks:
    - gitlab

networks:
  gitlab:
    name: gitlab-network
```

This configuration defines what containers we want to run. In our case, it will be the GitLab service with one GitLab runner (a separate module for running CI / CD tasks). The most important configuration parameters are:

- **image** – docker image that we want to use in our server
- **ports** – a list of ports that we make available outside the container. In our configuration, we provide ports 80, 443 (website)
- **container_name** – the name of our container
- **volumes** – specifies the volumes that are used by the container. In our configuration, we have directories shared with our system (subdirectories in \$ GITLAB_HOME) and an additional volume that allows access to the Docker environment from the GitLab runner.
- **networks** – defines the virtual network in which the containers will operate. In our case, the www portal and the runner operate in one “gitlab-network”

Gitlab installation

Containers are started using the command:

```
> docker-compose up -d
```

Once launched, Docker will download GitLab and GitLab Runner images from the servers. On my computer it looked like this:

```
[+] Running 17/17
# gitlab-runner Pulled 43.4s
# df20fa9351a1 Pull complete 3.3s
# 619db56e2f1b Pull complete 3.6s
# 39b9b6c9cb77 Pull complete 8.5s
# e76f8528690c Pull complete 23.0s
# 6a738eb478ee Pull complete 23.4s
# 4a518abb6253 Pull complete 30.1s
# 806889e90566 Pull complete 30.7s
# web Pulled 103.0s
# 7b1a6ab2e44d Pull complete 3.7s
# f1684ebc6541 Pull complete 5.5s
# 02468abe4769 Pull complete 5.8s
# 67434d82c6fc Pull complete 5.9s
# f3b378955ed9 Pull complete 6.1s
# 9161b9f516d4 Pull complete 6.4s
# 7b8b542d764e Pull complete 6.7s
# e297c8f137fb Pull complete 90.9s
[+] Running 3/3
# Network gitlab-network Created 0.0s
# Container gitlab-ce Started 1.4s
# Container gitlab-runner Started 1.7s
```

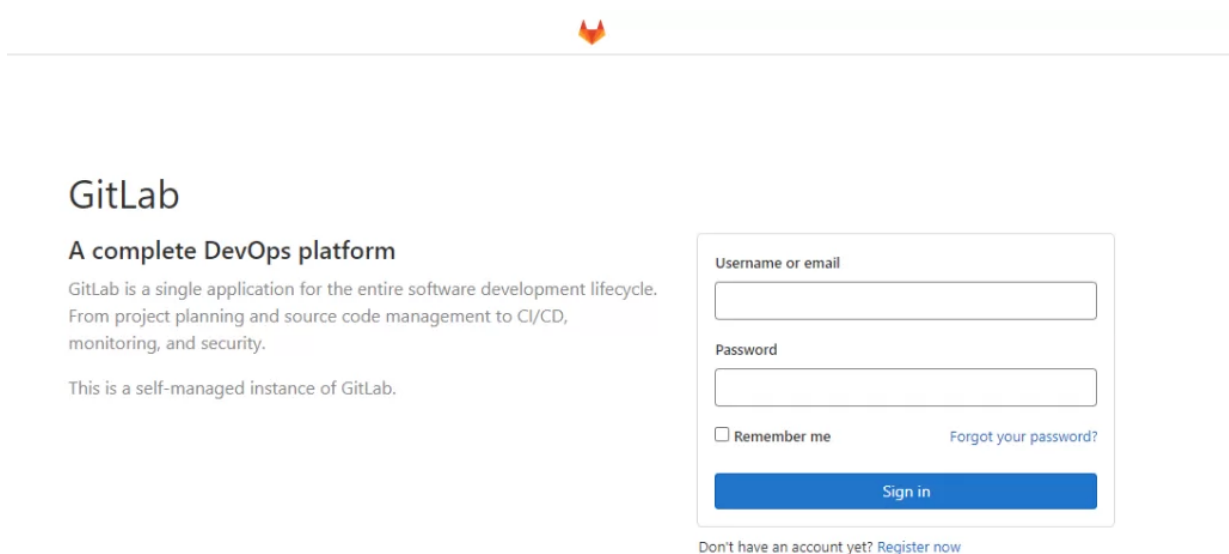
Docker compose

To log in to GitLab for the first time, you need a temporary password, which is generated automatically during installation. We get the password using the command:

```
> docker exec -it gitlab-ce grep 'Password:'
/etc/gitlab/initial_root_password
```

GitLab launching

Our GitLab is available at: <http://localhost:8080>. After going to this address, the following screen should appear:



GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

Username or email

Password

☐ Remember me [Forgot your password?](#)

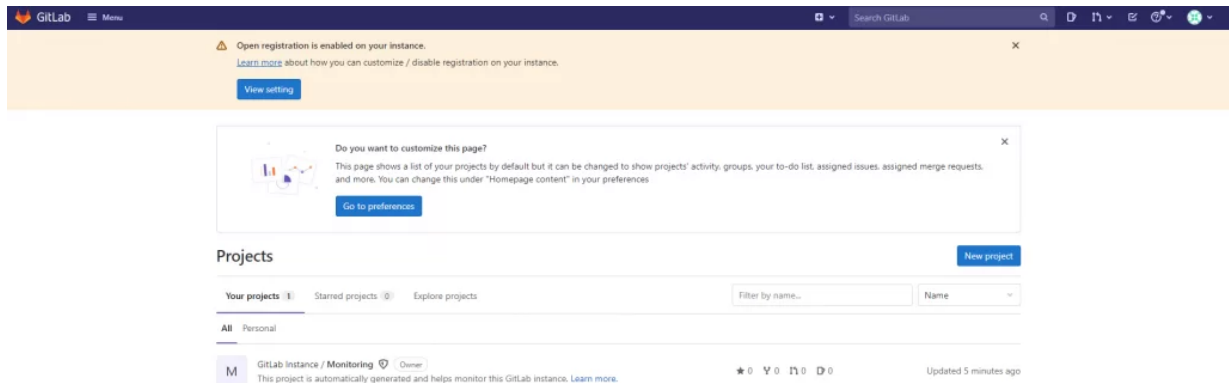
[Sign in](#)

Don't have an account yet? [Register now](#)

Note: The first launch of the portal may take several minutes.

To log in to the portal, we must enter the “root” in **Username** field and the temporary password that we obtained earlier in the **Password** field.

After logging in, the following screen should appear:

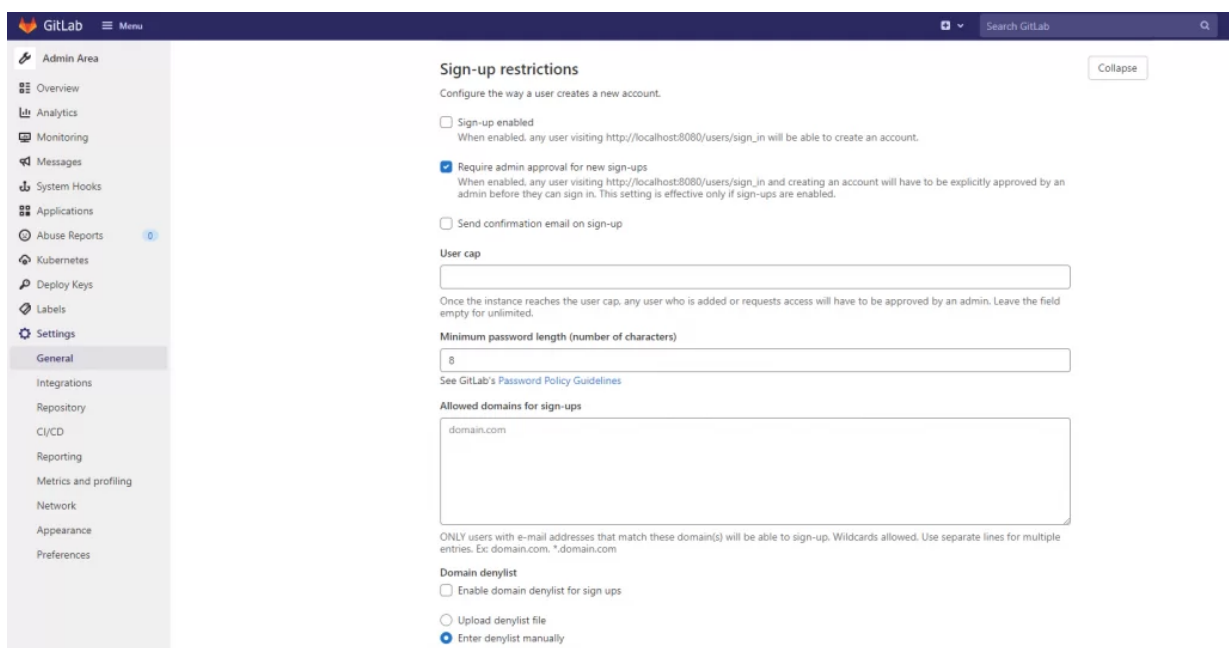


Gitlab

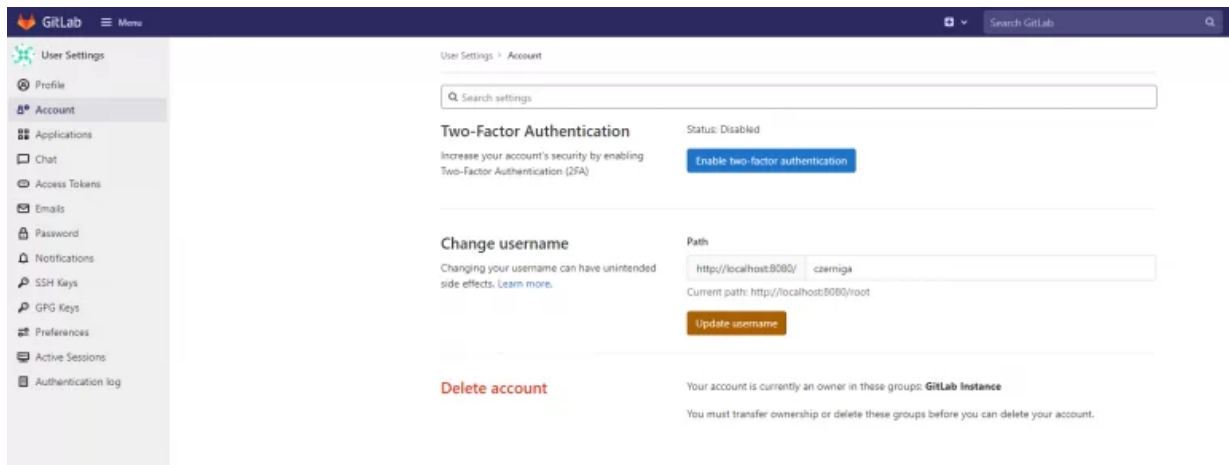
Congratulations, GitLab has been successfully launched!

Initial configuration of the portal

Before we proceed, it is worth changing a few portal settings. First, we'll turn off open registration for everyone. To do this, click the View Setting button available on the upper bar with a warning (address to the panel; http://localhost:8080/admin/application_settings/general#js-signup-settings). On the new page, uncheck Sign-up enabled and save the changes.



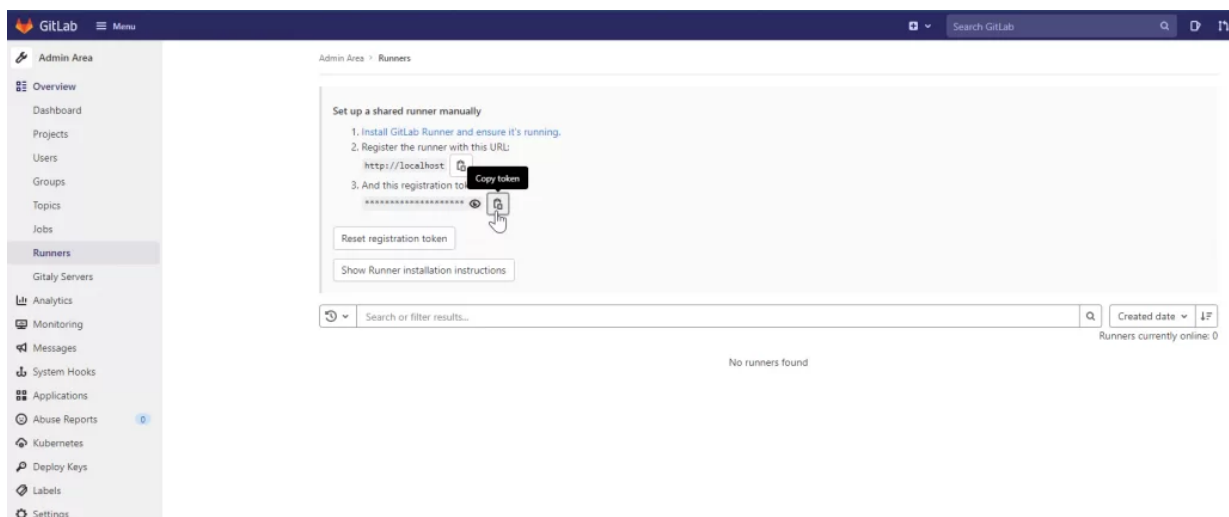
The next step should be to change the root user. To do this, go to the website: <http://localhost:8080/-/profile/account> and enter the name in the **Change username** field. We approve by clicking on **Update username**.



The last change we will make is to change the password. To do this, go to the page: <http://localhost:8080/-/profile/password/edit> and enter a temporary password and a new password. We approve the change by clicking **Save password**.

GitLab runner configuration

To use the GitLab runner in GitLab, you need to configure it. For correct configuration, we will need a token copied from the portal. To do this, go to the address: <http://localhost:8080/admin/runners> and click the **Copy token** button.



In the next step, it goes to the console and run the following command:

```
> docker exec -it gitlab-runner gitlab-runner register --url
"http://gitlab-ce" --clone-url "http://gitlab-ce"
```

After launching, a configuration module will appear. The module provides the following information:

- **Enter the GitLab instance URL:** confirm the entered value (click enter)
- **Enter the registration token:** enter the token copied before.
- **Enter a description for the runner:** enter the name of the runner, e.g. docker-runner
- **Enter tags for the runner:** leave the field blank here
- **Enter an executor:** enter docker here
- **Enter the default Docker image:** here we provide the default docker image, e.g. maven:latest

After proper configuration, we should see confirmation **Runner registered successfully:**

```
Runtime platform arch=amd64 os=linux pid=16 revision=4b9e985a version=14.4.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
[http://gitlab-ce]:
Enter the registration token:
[299c5246fdd6]:
Enter a description for the runner:
[299c5246fdd6]: docker-runner
Enter tags for the runner (comma-separated):
docker
Registering runner... succeeded runner=UYWWSVra
Enter an executor: custom, parallels, shell, kubernetes, docker, docker-ssh, ssh, virtualbox, docker+machine, docker-ssh+machine:
docker
Enter the default Docker image (for example, ruby:2.6):
maven:latest
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

In addition to the basic configuration, we also need to allow access for containers launched from the runner to the virtual network in which GitLab operates. To do this, we run the editor (e.g. vi)

```
> sudo vi gitlab/gitlab-runner/config.toml
```

Then we add new line to the end of the runner configuration: `network_mode = "gitlab-network"`

```

concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "docker-runner"
  url = "http://gitlab-ce"
  token = "XXXXXXXXXX"
  executor = "docker"
  clone_url = "http://gitlab-ce"
  [runners.custom_build_dir]
  [runners.cache]
  [runners.cache.s3]
  [runners.cache.gcs]
  [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "maven:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0
    network_mode = "gitlab-network"

```

To check if the runner is available from the GitLab level, go to the following address: <http://localhost:8080/admin/runners>

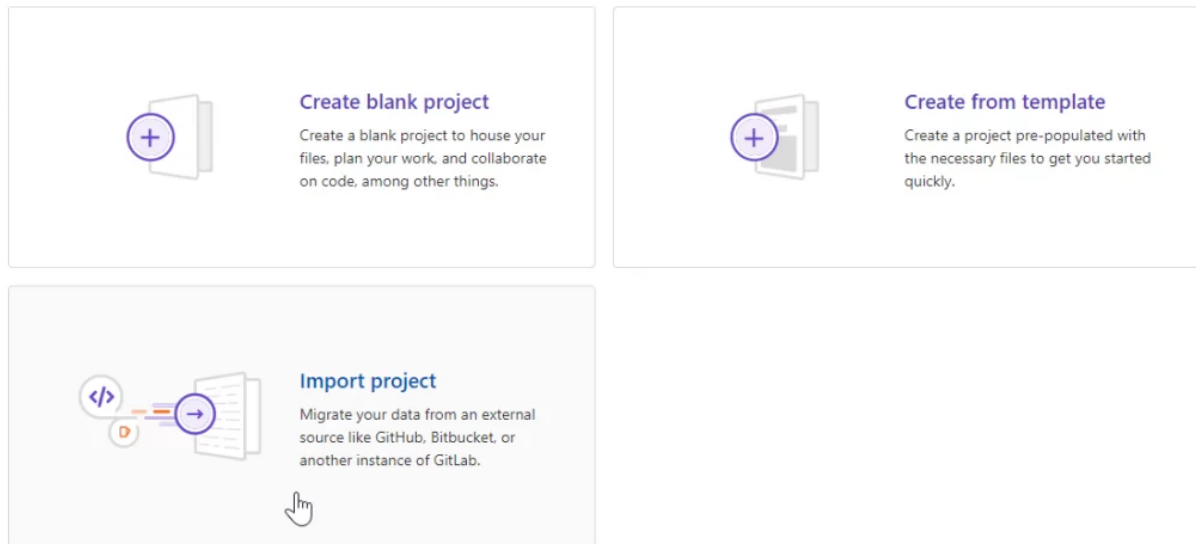
The screenshot shows the GitLab Admin Area - Runners page. The left sidebar contains navigation links: Admin Area, Overview, Dashboard, Projects, Users, Groups, Topics, Jobs, Runners, GitLab Servers, Analytics, Monitoring, Messages, System Hooks, Applications, Abuse Reports, Kubernetes, Deploy Keys, Labels, and Settings. The main content area is titled 'Admin Area > Runners' and includes a 'Set up a shared runner manually' section with three steps: 1. Install GitLab Runner and ensure it's running, 2. Register the runner with this URL: <http://localhost>, and 3. And this registration token: [XXXXXXXXXX](#). Below this are buttons for 'Reset registration token' and 'Show Runner installation instructions'. A search bar is present with the text 'Search or filter results...'. Below the search bar is a table of runners. A red box highlights the first runner in the table.

Type/State	Runner	Version	IP Address	Projects	Jobs	Tags	Last contact
shared	#1 (vF564hg) docker-runner	14.4.0	172.31.0.3	n/a	0	docker	1 minute ago

We create our first repository

After setting up the runner, we can create our first repository. To do this, go to the page: <http://localhost:8080/projects/new> and click **Import project**.

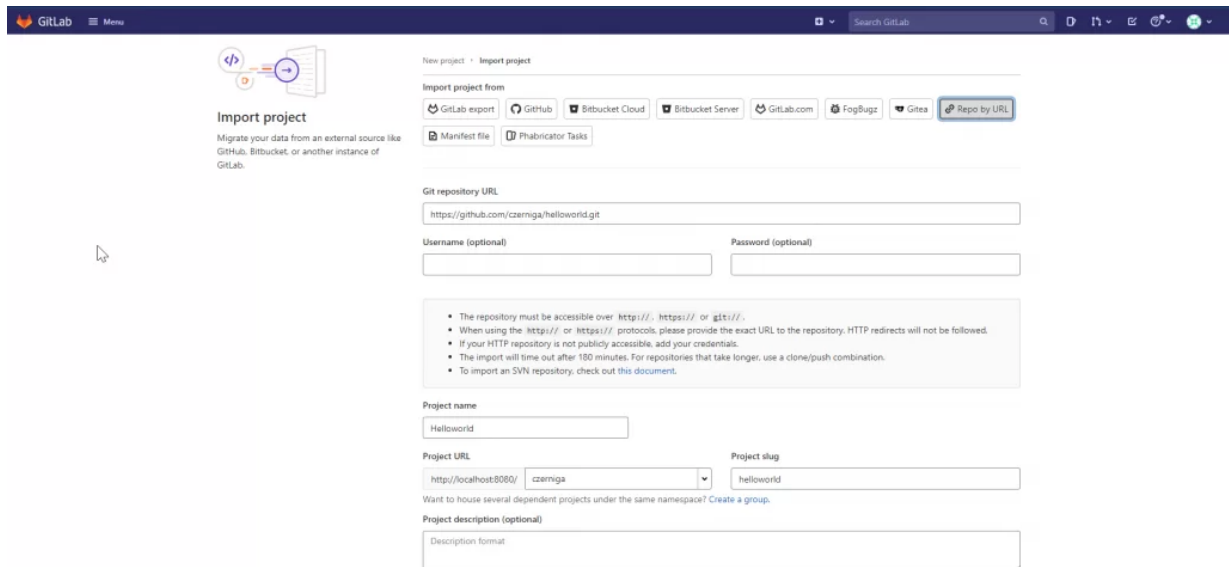
Create new project



You can also create a project from the command line. [Show command](#)


Na następnym ekranie wybieramy **Import project from: Repo from URL**. Następnie podajemy w **Git repository URL** address: <https://github.com/czerniga/helloworld.git>. Na końcu zatwierdzamy klikając **Create project**.

On the next screen, select **Import project from: Repo from URL**. Then we provide the Git repository URL address: <https://github.com/czerniga/helloworld.git>. Finally, confirm by clicking Create project.







After a while you should have the first repository copied to your GitLab.

Administrator > Helloworld




Helloworld
Project ID: 2




Star 0

Fork 0

4 Commits
1 Branch
0 Tags
133 KB Files
246 KB Storage

master
helloworld /
+
History
Find file
Web IDE
Clone


Update README.md
Patrycjusz Czerniga authored 1 day ago
Unverified
d1e12ebb

README
Auto DevOps enabled
Add LICENSE
Add CHANGELOG
Add CONTRIBUTING
Add Kubernetes cluster
Configure Integrations

Name	Last commit	Last update
src/main/java/com/coveros/dem...	Added time stamp to make each run unique ...	2 years ago
.gitignore	Simple Java application	3 years ago
README.md	Update README.md	1 day ago
pom.xml	Added time stamp to make each run unique ...	2 years ago

README.md


HelloWorld

A simple Java application that can be compiled into a .jar file using Maven.

To build


We create the CI/CD pipeline

To create a CI/CD pipeline for the project, click the main menu on the left, **CI/CD**, and then **Editor**. An option to create a `.gitlab-ci.yml` file, which will contain our pipeline definitions, will appear on the screen. This file will be created in the Git repository.


GitLab
Menu

Administrator > Helloworld > Pipeline Editor

master



Optimize your workflow with CI/CD Pipelines

Create a new `.gitlab-ci.yml` file at the root of the repository to get started.

[Create new CI/CD pipeline](#)

- Helloworld
- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
 - Pipelines
 - Editor**
 - Jobs
 - Schedules
- Security & Compliance
- Deployments
- Monitor
- Infrastructure
- Packages & Registries
- Analytics
- Wiki
- Snippets
- Settings

On the new screen we can see our pipeline editor. In the editor, paste the following content:

```
image: maven:latest

stages:
  - build
  - test

build-job:
  stage: build
  script:
    - echo "Compiling the code..."
    - mvn clean package
    - echo "Compile complete."
  artifacts:
    paths:
      - target

test-job:
  stage: test
  dependencies:
    - build-job
  script:
    - ls -al
    - echo "Running tests"
    - java -cp target/helloworld-1.1.jar
      com.coveros.demo.helloworld.HelloWorld
```

The above definition describes how the CI / CD process should work. The most important elements are:

- **image:** docker image that we will use to build our project
- **stages:** a list of our process steps
- **build-job:** the first step in our process to build our project. Additionally, we save the artifacts for use in the next step
- **test-job:** the second step to run our project

After pasting our file, confirm the changes by clicking **Commit changes**.

✓ This GitLab CI configuration is valid. [Learn more](#)

Edit Visualize Lint View merged YAML

Browse templates

```

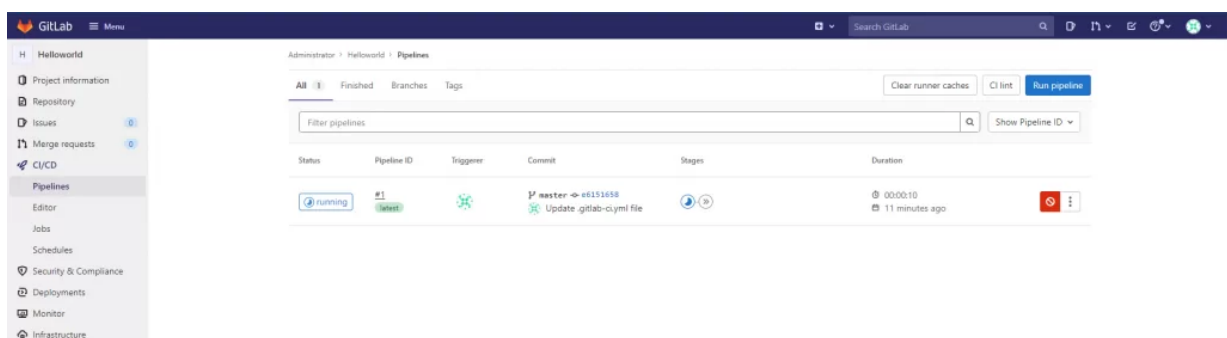
1 image: maven:latest
2
3 stages:
4   - build
5   - test
6
7 build-job:
8   stage: build
9   script:
10    - echo "Compiling the code..."
11    - mvn clean package
12    - echo "Compile complete."
13   artifacts:
14     paths:
15       - target
16
17 test-job:
18   stage: test
19   dependencies:
20     - build-job
21   script:
22     - ls -al
23     - echo "Running tests"
24     - java -cp target/helloworld-1.1.jar com.coveros.demo.helloworld.HelloWorld

```

Commit message:

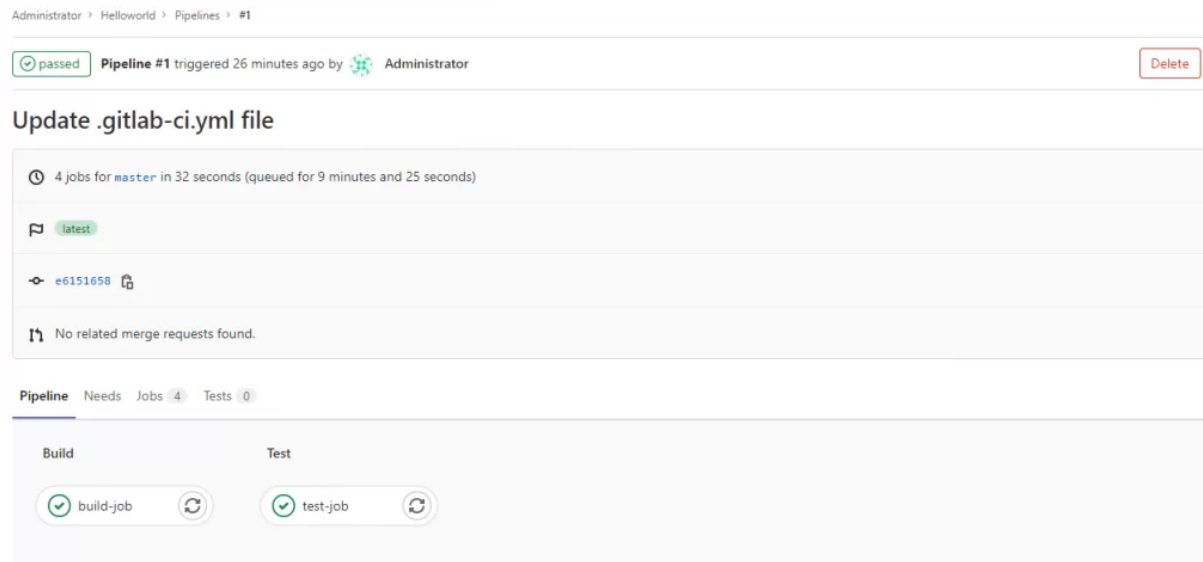
Target Branch:

Once approved, GitLab will launch the process. To check its results, go to **CI/CD** -> **Pipelines** in the menu on the left. On the screen we should see that our first task has already been started.



We can go to the details of this task by clicking on the **pending** button or the build number **#1**

After a while, the task should be built and tested.



How to install GitLab using Docker Compose

Congratulations, you have just created your first CI/CD job in GitLab!

Stopping the container

The containers containing our service were launched with the switch causing the work in the background. If you want to stop the portal, execute this command:

```
docker-compose down
```

Summary

In this tutorial, I showed how you can run GitLab with one GitLab runner. The configuration given here causes that all website data is saved in the directory on your computer / server. For this reason, the data will not be deleted when the containers are stopped or removed.

Published in [CI/CD](#) [Docker](#) [Tutorial](#)

[Docker compose](#)[GitLab](#)

Previous Post

[How to install Jenkins using Docker Compose?](#)

Next Post

[How to prepare your first CI/CD project in CircleCI?](#)

8 Comments



Mahesh

Dear Patrycjusz Czerniga

This is very useful article, thanks for sharing

If you are planning to update this further, I would recommend adding Gitlab-docker registry and NGINX reverse proxy to make it a complete suite

Thanks & Regards

Mahesh

January 15, 2022 | [Reply](#)



Agustin

this is amazing, I've been looking for a long time how to do it, thank you so much!

I consult you and I hope you can help me, when I run some job is giving me the following error:

fatal: unable to access 'http://gitlab-ce/gitlab-instance-bd6478cd/Monitoring.git/':
Could not resolve host: gitlab-ce

could you tell me how to fix it?

February 22, 2022 | [Reply](#)



Patrycjusz Czerniga (author)

Please double check if you gitlab-runner is in gitlab network and if you add new line to the end of the runner configuration: `network_mode = "gitlab-network"`.

February 23, 2022 | [Reply](#)



bahram mahmoudi

Hi, patrycjusz
Thank You so much
cool & perfect.

September 11, 2022 | [Reply](#)



Takis Pournaras

Awesome post, thanks for detailing the process.

There's a typo in the following paragraph:

GitLab launching
Our GitLab is available at: `http: //localhost:8080`

(space after http:)

September 26, 2022 | [Reply](#)



Patrycjusz Czerniga (author)

You're right, thanks for pointing it out!

September 26, 2022 | [Reply](#)



Eva

Thank you very much. Just one caveat for those who try:
Be careful if you copy&paste
`network_mode = "gitlab-network"`

quotation marks can be tricky on Vim editor

September 29, 2022 | [Reply](#)



Arda

Thank you its amazing and very helpful!

October 8, 2022 | [Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment