

Инструкция по работе с GitLab CI

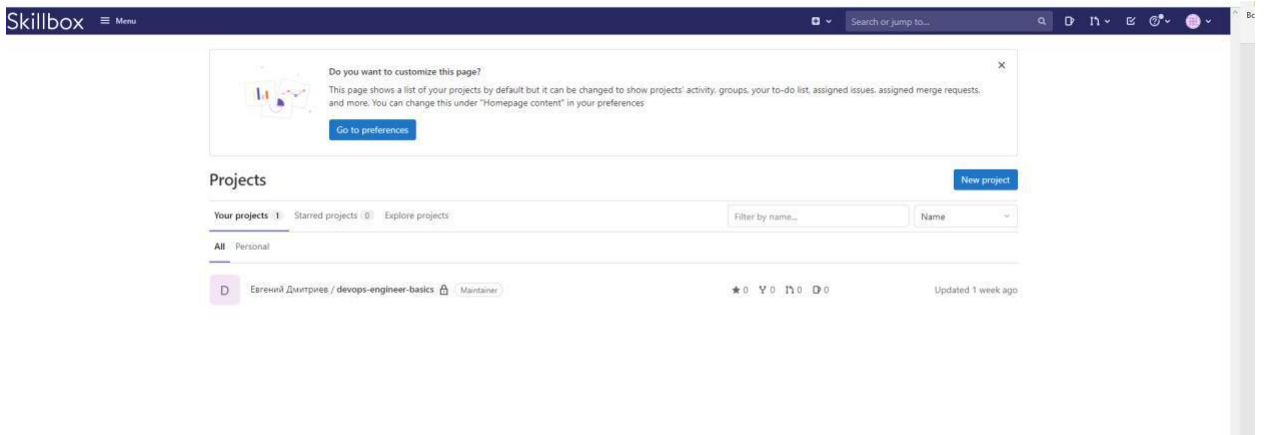
Эта инструкция о том, как правильно работать с GitLab. Информацию из инструкции следует рассматривать как отправную точку в изучении веб-инструмента. Мы не будем обсуждать тонкости, которые важны для опытных разработчиков, а рассмотрим только аспекты, необходимые для успешного прохождения курса.

Вы познакомитесь с базовыми темами, которые пригодятся вам при использовании GitLab CI, и узнаете:

- как работать с репозиториями;
- как подключать и настраивать раннеры;
- как создавать и запускать пайплайны.

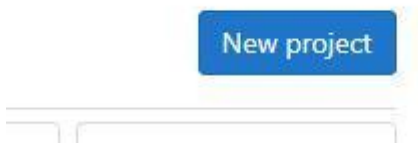
Работа с репозиториями

1. После получения доступа к GitLab Skillbox вы попадёте на начальную страницу. На ней будет только один проект:



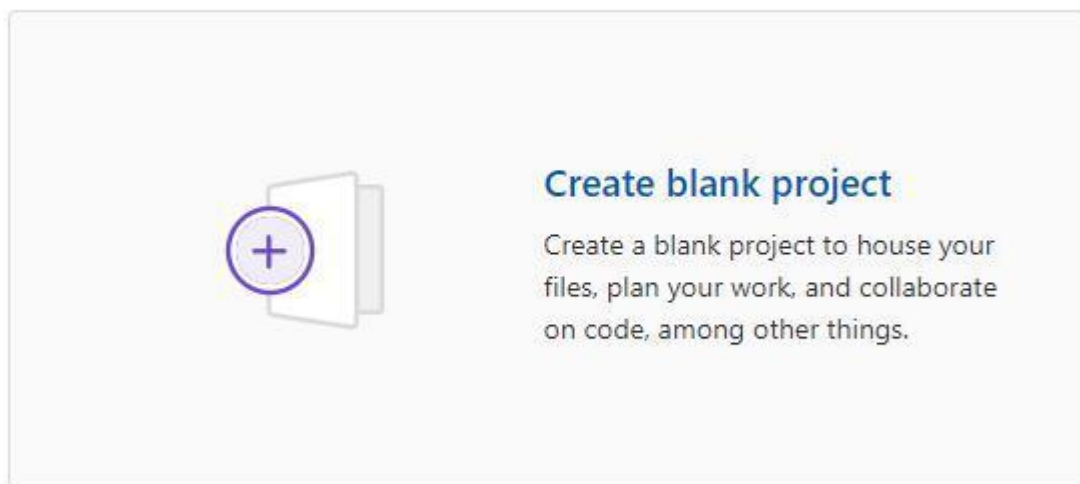
Здесь и далее — изображения Skillbox

2. Чтобы создать новый проект, нажмите на кнопку New project:



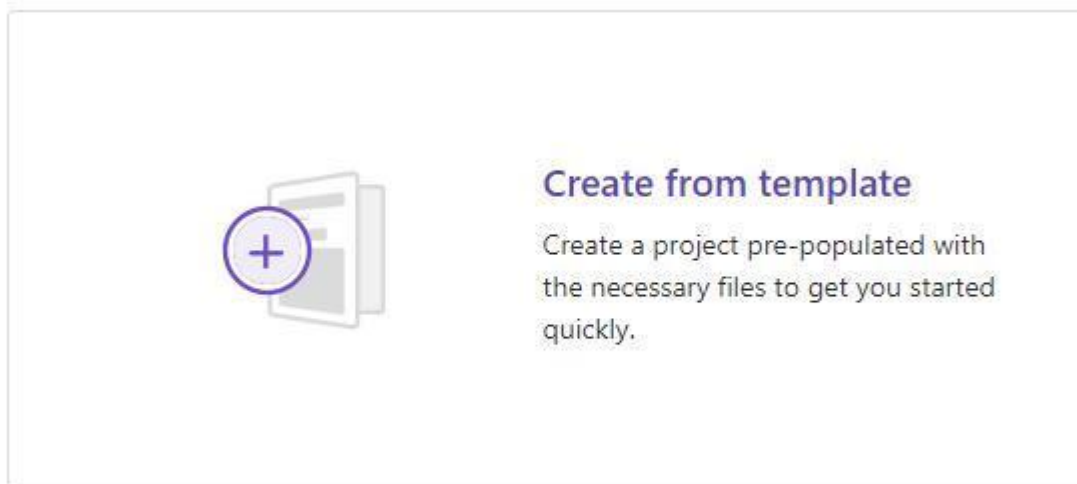
3. Вы окажетесь на новой странице с тремя вариантами.

- **Вариант 1** — создать пустой проект:



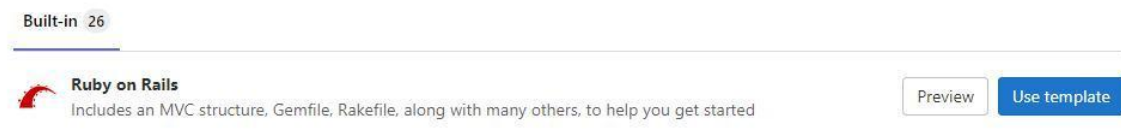
Этот вариант используется чаще всего, так как он позволяет создать проект с нуля и даёт полную свободу действий.

- **Вариант 2** — создать проект из шаблона:



Это продвинутый вариант. Выбрав его, вы будете использовать заранее определённые шаблоны репозитория в GitLab. Например, это могут быть шаблоны веб-приложений. Попробуйте использовать шаблон.

A. Кликните на шаблоне Ruby:



















B. Введите:

- имя,
- описание проекта.

Нажмите Create project («Создать проект»).

C. Вы попадёте в репозиторий с готовыми базовыми каталогами, деревом проекта и конфигурационными файлами:

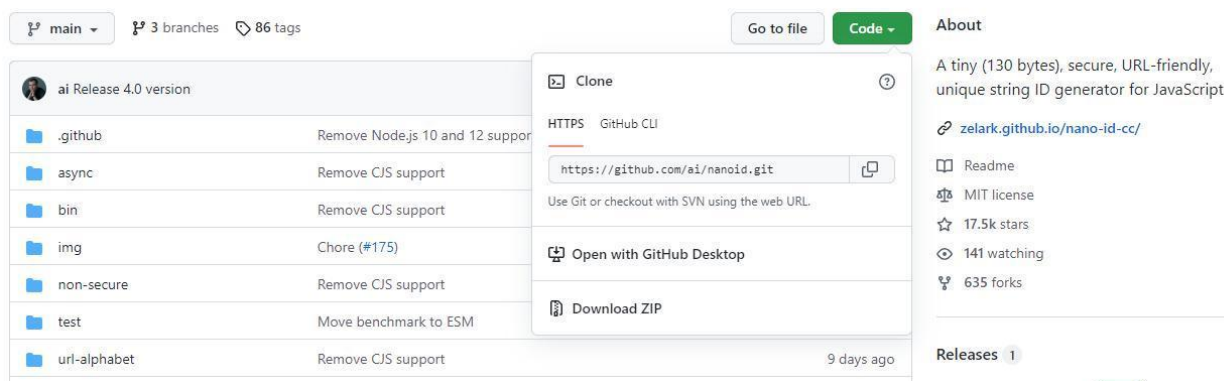
<div>  <div> Initialized from 'Ruby on Rails' project template ... </div> <div> GitLab authored 1 year ago </div> </div> <div> <div>07b712c4</div> <div></div> </div>		
Name	Last commit	Last update
 app	Initialized from 'Ruby on Rails' project templ...	1 year ago
 bin	Initialized from 'Ruby on Rails' project templ...	1 year ago
 config	Initialized from 'Ruby on Rails' project templ...	1 year ago
 db	Initialized from 'Ruby on Rails' project templ...	1 year ago
 lib	Initialized from 'Ruby on Rails' project templ...	1 year ago
 log	Initialized from 'Ruby on Rails' project templ...	1 year ago
 public	Initialized from 'Ruby on Rails' project templ...	1 year ago
 storage	Initialized from 'Ruby on Rails' project templ...	1 year ago
 test	Initialized from 'Ruby on Rails' project templ...	1 year ago
 tmp	Initialized from 'Ruby on Rails' project templ...	1 year ago
 vendor	Initialized from 'Ruby on Rails' project templ...	1 year ago
 .browserslistrc	Initialized from 'Ruby on Rails' project templ...	1 year ago
 .gitattributes	Initialized from 'Ruby on Rails' project templ...	1 year ago
 .gitignore	Initialized from 'Ruby on Rails' project templ...	1 year ago
 .gitpod.Dockerfile	Initialized from 'Ruby on Rails' project templ...	1 year ago

Если вы понимаете, что нужно делать, — такой вариант сэкономит ваше время. В GitLab немного шаблонов, но есть наиболее популярные языки и фреймворки.

- **Вариант 3** — импортировать проект.

Нажав на этот вариант, вы перейдёте на страницу с первоначальной настройкой. Импортируйте проект из GitHub, например лёгкий [репозиторий с Nanoid](#), чтобы не занимать слишком много места на серверах Skillbox.

A. Скопируйте ссылку для клонирования репозитория:



В. Добавьте её в GitLab:

Git repository URL

https://github.com/ai/nanoid.git

Username (optional)

Password (optional)

- The repository must be accessible over `http://`, `https://` or `git://`.
- When using the `http://` or `https://` protocols, please provide the exact URL to the repository. HTTP redirects will not be followed.
- If your HTTP repository is not publicly accessible, add your credentials.
- The import will time out after 180 minutes. For repositories that take longer, use a clone/push combination.
- To import an SVN repository, check out [this document](#).

Project name

Nanoid

Project URL

https://gitlab.skillbox.ru/evgenii_dmitriev_1/


Project slug

nanoid

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level 

☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

Other visibility settings have been disabled by the administrator.

Create project

Cancel

Поля Project name и Project slug подставляются автоматически.

С. Нажмите «Создать проект» и подождите:

Евгений Дмитриев > Nanoid > Import in progress

Import in progress

```
git clone --bare https://github.com/ai/nanoid.git
```

Please wait while we import the repository for you. Refresh at will.

Nanoid клонируется довольно быстро. После недолгого ожидания вы увидите синюю плашку. Это означает, что вы импортировали репозиторий в проект на GitLab.

Работа с репозиториями — первый шаг на пути освоения GitLab CI, понимание которого пригодится при прохождении курса. Чтобы закрепить эти знания, можете самостоятельно изучить разные варианты и попрактиковаться:

1. Импортируйте ещё пару репозиторииев на свой выбор.
2. Создайте пустые репозитории и добавьте в них что-нибудь. При этом в репозитории не стоит добавлять бинарные файлы, так как для них есть отдельные инструменты, например Artifactory и Nexus.
3. Создайте проект из шаблона, например Swift или Spring. Посмотрите, как он работает и что с ним можно делать.

Работа с GitLab Runner

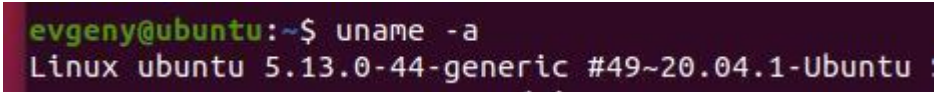
Вы импортировали репозиторий и можете приступить к дальнейшей работе. Например, можно собрать ранее импортированный код.

Даже если вы ни разу не сталкивались с понятием «компиляция» (процесс преобразования программного кода из одного языка программирования в другой, или сборка кода), вы сможете выполнить необходимые действия благодаря этой инструкции.

Установка софта

Чтобы полностью повторить шаги из гайда, желательно установить аналогичный софт:

- VMware Workstation Player 16;
- Ubuntu 20.04.



```
evgeny@ubuntu:~$ uname -a
Linux ubuntu 5.13.0-44-generic #49~20.04.1-Ubuntu
```

[На сайте GitLab](#) вы найдёте подробную информацию о том, как поставить раннер для GitLab через репозитории Linux.

Что такое GitLab Runner

Прежде чем приступить к работе с раннером, разберём, что такое раннер для GitLab и какие типы раннеров существуют.

GitLab Runner — компонент, который используется для запуска задач Continuous Integration (CI) и Continuous Delivery (CD) в GitLab. Он представляет собой отдельное приложение, которое работает на хост-машине или в контейнере и выполняет задачи, определённые в файле конфигурации CI/CD проекта.

- **Shell Executor** — наиболее простой тип Runner, который запускает задачи внутри командной оболочки хост-машин. Он позволяет выполнять команды и скрипты на хост-машине, подходит для простых проектов и маленьких сред.
- **Docker Executor** — использует Docker для запуска задач CI/CD. Он создаёт и запускает временные контейнеры для каждой задачи, обеспечивая изолированное и повторяемое окружение выполнения. Этот тип Runner особенно полезен для проектов, требующих специфических зависимостей и окружения.
- **Parallels Executor** — позволяет запускать задачи в виртуальных машинах с Parallels Desktop на macOS-хосте.

- **SSH Executor** — позволяет запускать задачи на удалённых хостах по протоколу SSH. Этот тип Runner особенно полезен, когда требуется выполнять задачи на удалённых хостах.
- **VirtualBox Executor** — позволяет запускать задачи в виртуальных машинах VirtualBox. Он создаёт и запускает изолированные виртуальные машины для выполнения задач и может быть полезен для тестирования и разработки на разных операционных системах.
- **Kubernetes Executor** — позволяет запускать задачи в кластере Kubernetes. Он создаёт и управляет подами Kubernetes для выполнения этих задач, что обеспечивает масштабируемость. Этот тип Runner полезен для проектов, использующих Kubernetes для развёртывания и управления контейнерами.
- **Docker Machine Executor** — позволяет использовать удалённые хосты, управляемые Docker Machine, для выполнения задач CI/CD. Он позволяет гибко масштабировать вычислительные ресурсы и может быть полезен, когда требуется доступ к удалённым ресурсам, таким как облачные провайдеры.

Отличительные особенности каждого типа раннера:

Executor	SSH	Shell	VirtualBox	Parallels	Docker	Kubernetes	Custom
Clean build environment for every build	×	×	✓	✓	✓	✓	conditional (4)
Reuse previous clone if it exists	✓	✓	×	×	✓	×	conditional (4)
Runner file system access protected (5)	✓	×	✓	✓	✓	✓	conditional
Migrate runner machine	×	×	partial	partial	✓	✓	✓
Zero-configuration support for concurrent builds	×	×	✓	✓	✓	✓	conditional (4)
Complicated build environments	×	×	✓ (3)	✓ (3)	✓	✓	✓
Debugging build problems	easy	easy	hard	hard	medium	medium	medium

Изображение: [GitLab](#)

Подключение GitLab Runner

1. Установите раннер. Для этого вбейте в консоль команду (можно скопировать её отсюда):

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

2. Установите пакет GitLab-runner:

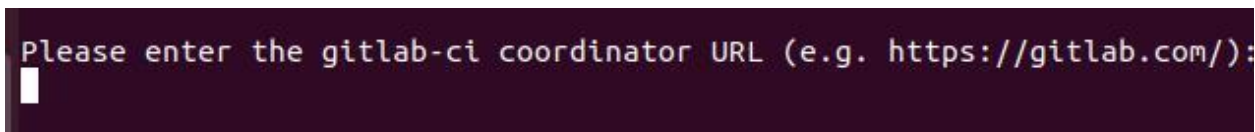
sudo apt-get install gitlab-runner

Настройка GitLab Runner

1. После окончания установки отрегулируйте настройки. Введите в консоль команду:

`sudo gitlab-runner register`

2. Первое, что вы увидите, — запрос ввести URL координатора GitLab:

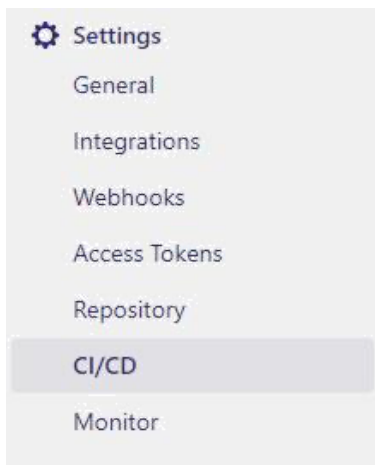


```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
```

Здесь и далее — изображения Skillbox

Для этого вернитесь в ваш проект в браузере.

3. В настройках перейдите в CI/CD и нажмите Expand («Развернуть») в разделе Runners:



4. В этой области есть URL и специальный токен, по которому можно зарегистрировать раннер:

Specific runners

These runners are specific to this project.

Set up a specific runner automatically

Register a runner on a Kubernetes cluster. [Learn more.](#)

1. Click the button below.
2. Select an existing Kubernetes cluster or create a new one.
3. From the Kubernetes cluster details view, applications list, install GitLab Runner.

[Install GitLab Runner on Kubernetes](#)

Set up a specific runner manually

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

And this registration token:
`wzSs6WfV5_QaCtnMurDn`

[Reset registration token](#)

[Show Runner installation instructions](#)

5. Введите URL:

```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):  
https://gitlab.skillbox.ru/
```

6. Введите токен:

```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):  
https://gitlab.skillbox.ru/  
Please enter the gitlab-ci token for this runner:
```

7. Вставьте токен из поля в браузере:

```
And this registration token:  
wzSs6WfV5_QaCtnMurDn
```

8. Введите описание:

```
Please enter the gitlab-ci description for this runner:  
[ubuntu]: Test
```

9. Укажите теги, которые позволяют запускать проект на конкретных раннерах:

```
Please enter the gitlab-ci tags for this runner (comma separated):  
study,skillbox  
Registering runner... succeeded runner=wzSs6WfV
```

10. Выберите экзекUTOR, в большинстве случаев — shell, и пропишите его в консоли:

```
Please enter the executor: docker, docker-ssh, parallels, ssh, docker+machine, shell, virtualbox, docker-ssh+machine, kubernetes:  
shell
```

11. Если вы всё сделаете правильно, в разделе раннеров появится только что зарегистрированный раннер:

Available specific runners

● #2402 (QTycgM_7) 🔒

  Remove runner

Test

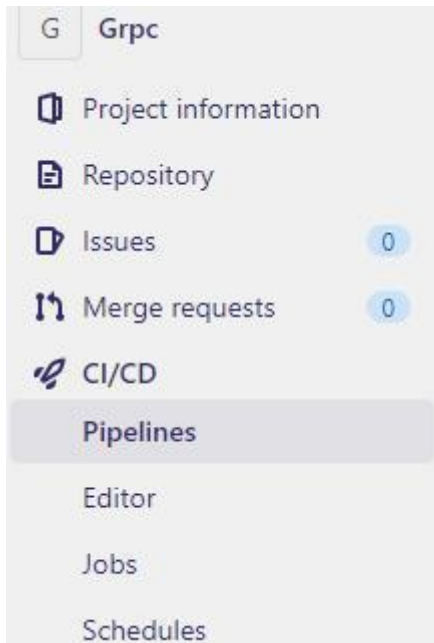
skillbox study

Теперь вы можете работать с репозиторием своего проекта и подключать или настраивать раннер. Перейдём к пайплайну.

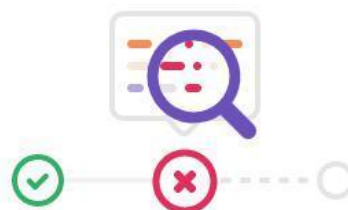
Создание и запуск пайплайнов

Пайплайны позволяют выполнять разные операции над кодом, который хранится в репозитории.

1. Чтобы попасть в меню редактирования пайплайнов, в левом меню перейдите в раздел CI/CD. Если у вас новый проект, страница будет пустой.



2. Создайте новый пайплайн:

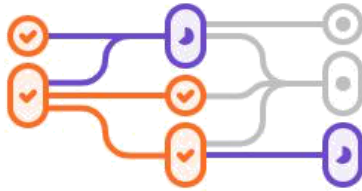


Build with confidence

GitLab CI/CD can automatically build, test, and deploy your code. Let GitLab take care of time consuming tasks, so you can spend more time creating.

[Get started with CI/CD](#)

3. Перейдите в раздел Editor, чуть ниже кнопки Pipelines.



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

4. Создайте новый пайплайн с тестовыми джобами. Добавьте к каждой джобе тег, который указывали при регистрации раннера:

```
5
6 build-job:      # This job runs in the build stage, which runs first.
7   stage: build
8   script:
9     - echo "Compiling the code..."
10    - echo "Compile complete."
11   tags:
12     - skillbox
```

5. Закоммитьте изменения. По умолчанию пайплайн запустится сразу после попадания изменений в репозиторий:

🕒 4 jobs for `master` in 2 minutes and 9 seconds (queued for 4 seconds)

🏷️ `latest`

🔗 `e2f547e5` 📄

🔗 No related merge requests found.

Pipeline Needs Jobs 4 Tests 0

Build	Test	Deploy
✅ build-job 🔄	✅ lint-test-job 🔄	✅ deploy-job 🔄
	✅ unit-test-job 🔄	

Пайплайн отработает через несколько минут.

Итог

Теперь вы можете применять полученные знания на практике.

Любая задача, с которой вы столкнётесь, сводится к трём базовым шагам:

1. Создать проект и репозиторий.
2. Зарегистрировать раннер и ассоциировать его.
3. Добавить, отредактировать, убрать джобы в пайплайне.

Вы можете возвращаться к этой инструкции каждый раз, когда нужно настроить сборку проекта. Также можно несколько раз перечитать её, чтобы запомнить все шаги.

Полезные ссылки

- [Organize work with projects](#) — информация, которая поможет при создании проекта и работе со встроенными шаблонами.
- [Tags](#) — информация о тегах, которые мы будем применять к джобам.
- [Install GitLab Runner](#) — детальная информация о том, как регистрировать и устанавливать раннеры на разных ОС в разных условиях.

Примечания

Если у вас ничего не работает, можете попробовать лайфхак — сделать так, чтобы ваши версии не отличались от использованных в инструкции:

```
Version:      11.2.0
Git revision: 11.2.0
Git branch:   HEAD
GO version:   go1.10.4
Built:        unknown
OS/Arch:      linux/amd64
```


Возможно, джобы не стартуют после коммита даже при подключённом раннере. В этом случае введите в консоли команду **gitlab-runner**, чтобы увидеть список доступных команд:

```
COMMANDS:
  exec          execute a build locally
  list          List all configured runners
  run           run multi runner service
  register      register a new runner
  install       install service
  uninstall     uninstall service
  start         start service
  stop          stop service
  restart       restart service
  status        get status of a service
  run-single    start single runner
  unregister    unregister specific runner
  verify        verify all registered runners
  artifacts-downloader download and extract build artifacts (internal)
  artifacts-uploader  create and upload build artifacts (internal)
  cache-archiver  create and upload cache artifacts (internal)
  cache-extractor  download and extract cache artifacts (internal)
  help, h       Shows a list of commands or help for one command

GLOBAL OPTIONS:
  --debug          debug mode [$DEBUG]
  --log-level value, -l value  Log level (options: debug, info, warn, error, fatal, panic)
  --cpuprofile value  write cpu profile to file [$CPU_PROFILE]
  --help, -h       show help
  --version, -v    print the version
evgeny@ubuntu:~$
```

Если билд не запускается, примените следующие команды по очереди и попробуйте запустить билд после этого:

- **gitlab-runner verify;**
- **gitlab-runner restart;**
- **gitlab-runner run.**