

Летняя школа DevOps и CI/CD

GitLab CI/CD basics
































































GitLab это не только сервер для git

- Контроль версий (исходников, конфигурации, скриптов развертывания)
- Хранилище кода
- Поддержка запросов на слияние (merge requests)
 - Поддержка gitflow
- Поддержка code reviews
- Разрешение/предупреждение конфликтов
- Управление доступом
- И много другое

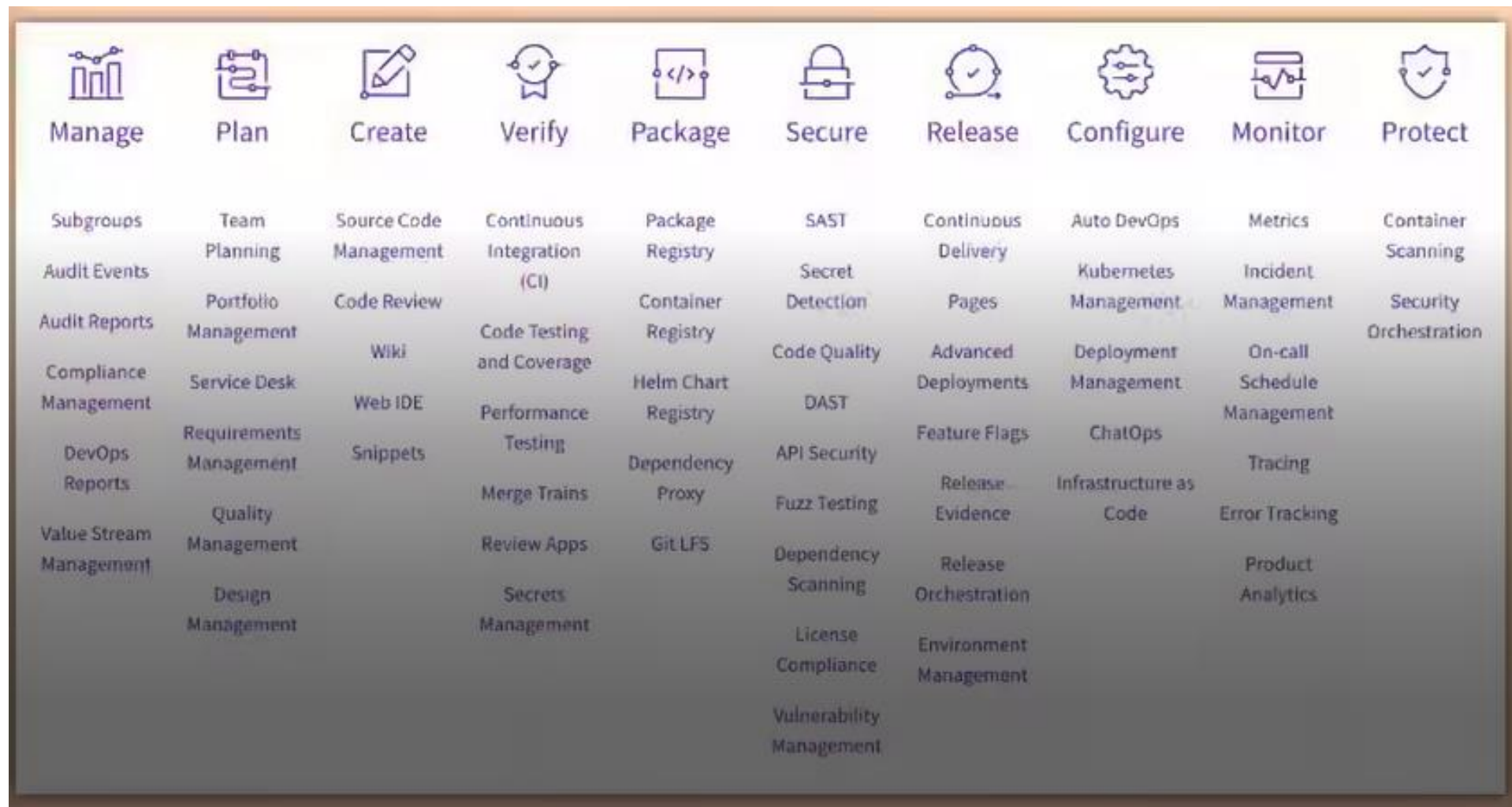
GitLab это не только сервер для git

- Инструменты, позволяющие проводить тестирование, сборку и т.п.
- Встроенный реестр пакетов
- Встроенные средства безопасности (такие как SAST, DAST)
- Интегрированное решение – DevOps platform

DevOps platform

 Manage	 Plan	 Create	 Verify	 Package	 Release	 Configure	 Monitor	 Secure
								
								
								
								
								
								

DevOps platform



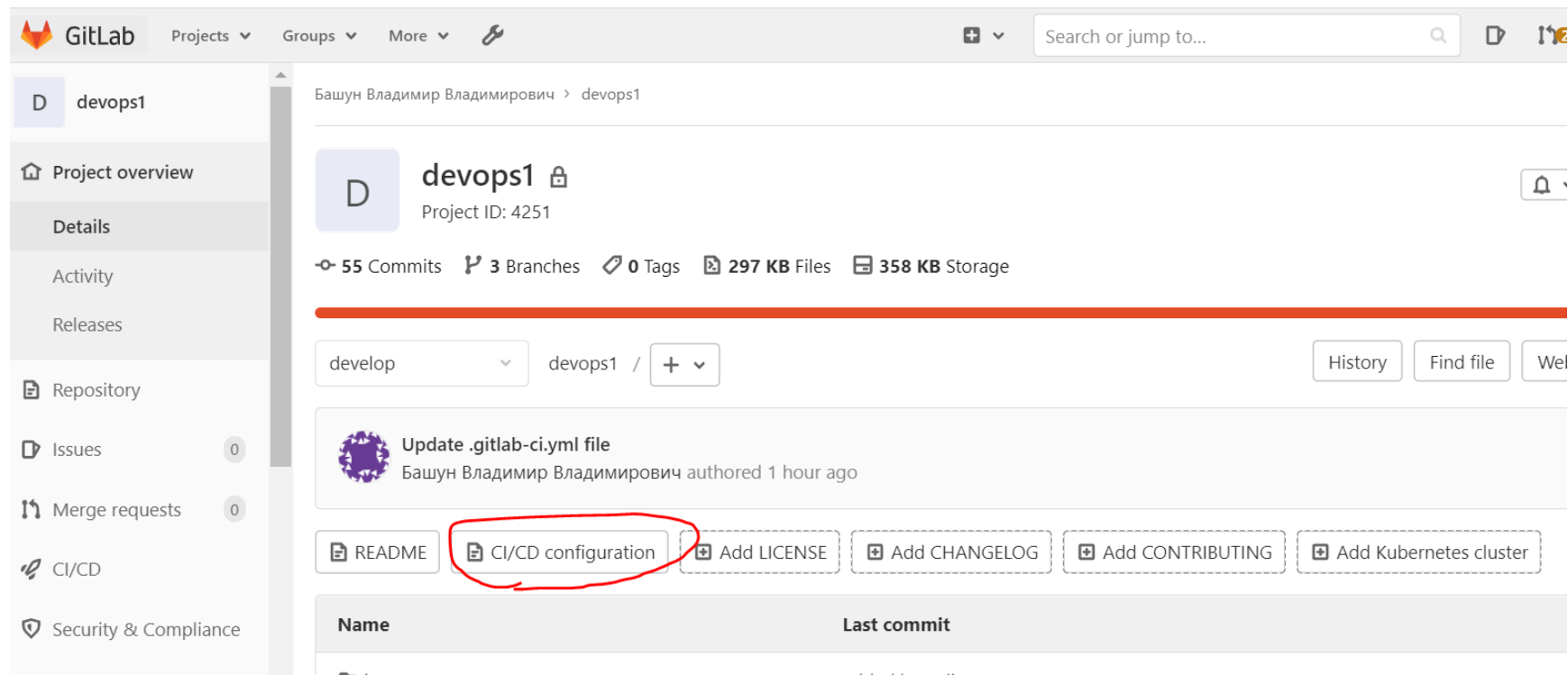
GitLab

- Варианты развертывания
 - SaaS – gitlab.com
 - Self-managed (как git.miem.hse.ru)

type of difference	GitLab SaaS	GitLab self-managed
Infrastructure	GitLab manages HA Architecture, and instance-level backups, recovery, and upgrades	manage your own, anywhere
Instance wide settings	same for all users	custom
Access controls	customer is group owner	customer is admin
Features availability, such as:	SAML SSO is Premium	SAML or LDAP is Core
Log information and auditing *	no access, but Support or Security can answer questions	unrestricted access
Reporting, DevOps adoption	Group and project level DevOps adoption reports	Usage trends, instance-level DevOps adoption reports

Настройка CI/CD в GitLab

- Как именно настроить CI/CD процессы в GitLab
 - Нужно создать конфигурационный yaml файл
 - По умолчанию это будет .gitlab-ci.yml



Настройка CI/CD в GitLab

- Как и все с концепции IaC, все должно быть в форме скриптов / кода
- Используется стандартный формат файла – YAML
- Настройки действий по автоматизации процессов лежат рядом с самим проектом, в том же репозитории
 - Хотя можно настроить и по другому
- Это элемент авто-документации
 - Сразу видно, как нужно собирать/тестировать/разворачивать продукт

Создание заданий (jobs)

- Необходимо определить следующие сущности
 - Задания (jobs) – определяют, что именно надо делать
 - Этапы (stages) – определяют, когда именно надо выполнять работы
 - Конвейеры (pipelines) – собирают все вместе, определяют поэтапное выполнение заданий

Настройка pipelines

- Это компонент верхнего уровня
- Pipelines могут быть достаточно сложными
 - объединять несколько проектов,
 - завязаны на merge request в отличие от комитов
 - родитель-потомок и т.п.
- Мы рассмотрим простые pipeline (базовые)
- Pipeline формируется из отдельных заданий (jobs)
 - Т.е. надо сначала сформировать задание и добавить его в конвейер

Создаем задание

- Это самый базовый элемент, из которого строятся конвейеры работ при непрерывной интеграции или доставке
- Минимально у задания есть название и сценарий (script)

```
7   job:  
8   |   script:  
9   |       # provide a shell script as argument for this keyword.  
10  |       - echo "Hello World"
```

- В script можно писать bash команды
 - Можно набор команд, можно запустить bash скрипт, лежащий в репозитории
 - Дальше увидим, какие вообще команды допустимы

Создаем задание

- Задаются прямо в gitlab-ci файле
 - Создадим работы
run_unit_test, run_lint_test,
build_image, push_image

```
run_tests:
  script:
    - echo "Running tests..."

build_image:
  script:
    - echo "Building the docker image..."
    - echo "Tagging the docker image"

push_image:
  script:
    - echo "Logging into docker registry..."
    - echo "Pushing docker image to registry"
```

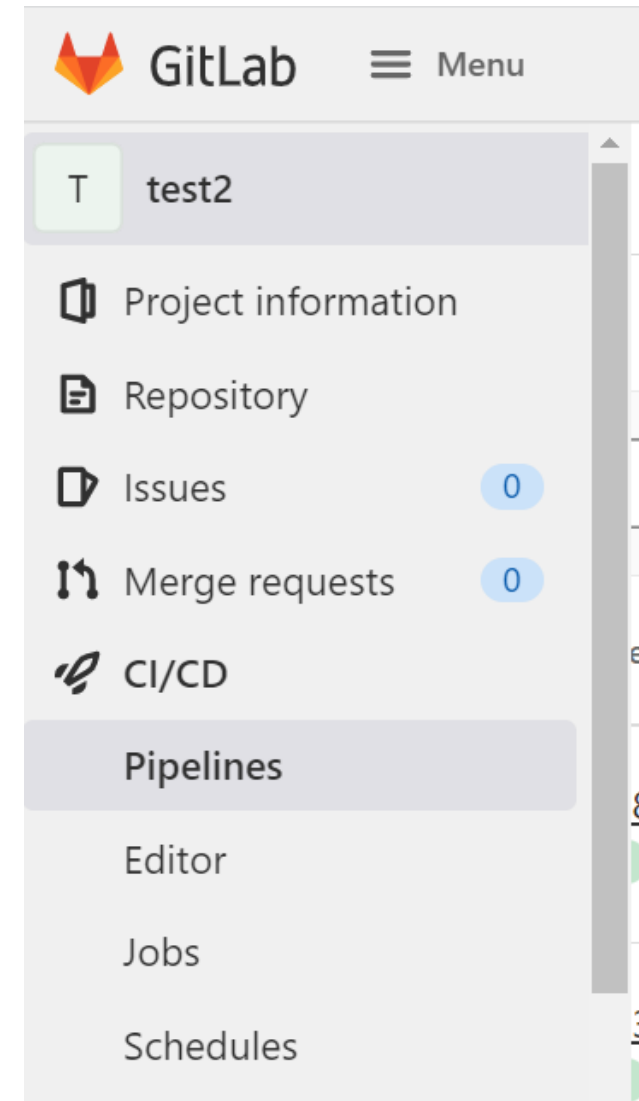
Создаем задание

- Задаются прямо в gitlab-ci файле
 - Создадим работы run_unit_test, run_lint_test, build_image, push_image
- Кроме раздела script, можно так же добавить before_script, after_script

```
1  run_tests:
2    before_script:
3      - echo "Preparing test data..."
4    script:
5      - echo "Running tests..."
6    after_script:
7      - echo "Cleaning up temporary files..."
8
9  build_image:
10   script:
11     - echo "Building the docker image..."
12     - echo "Tagging the docker image"
```

Запуск задания

- Создадим несколько заданий, и автоматически создастся конвейер
- Результат выполнения можно увидеть в разделе pipelines



Запуск задания

- Результат выполнения можно увидеть в разделе pipelines
- Видим, что работа не выполняется, т.к. нет подходящего раннера (runner)



🕒 3 jobs for **master**



🚩 **latest** **stuck**

🔑 **b467fe6d** 🗄️

🔗 No related merge requests found.

Pipeline Needs Jobs 3 Tests 0

Build	Test
 build-job	 lint-test-job

 pending **Job build-job** created 1 minute ago by  Башун Владимир Владимирович

⚠️ This job is stuck because the project doesn't have any runners online assigned to it.
Go to project [CI settings](#)

Запуск задания

- Результат выполнения можно увидеть в разделе pipelines
- Видим, что работа не выполняется, т.к. нет подходящего раннера (runner)
- Позже разберемся что это, а пока добавим еще раздел tags: - docker

```
✓ deploy-job:      # This job runs in the deploy s
| stage: deploy    # It only runs when *both* jobs
| tags:
|   - docker
| script:
|   - echo "Deploying application..."
|   - echo "Application successfully deployed."
```


Запуск задания

- Возвращаемся в раздел pipelines, видим, что работы запущены

devops > test2 > Pipelines

All14FinishedBranchesTags

Clear runner cachesCI lintRun pipeline

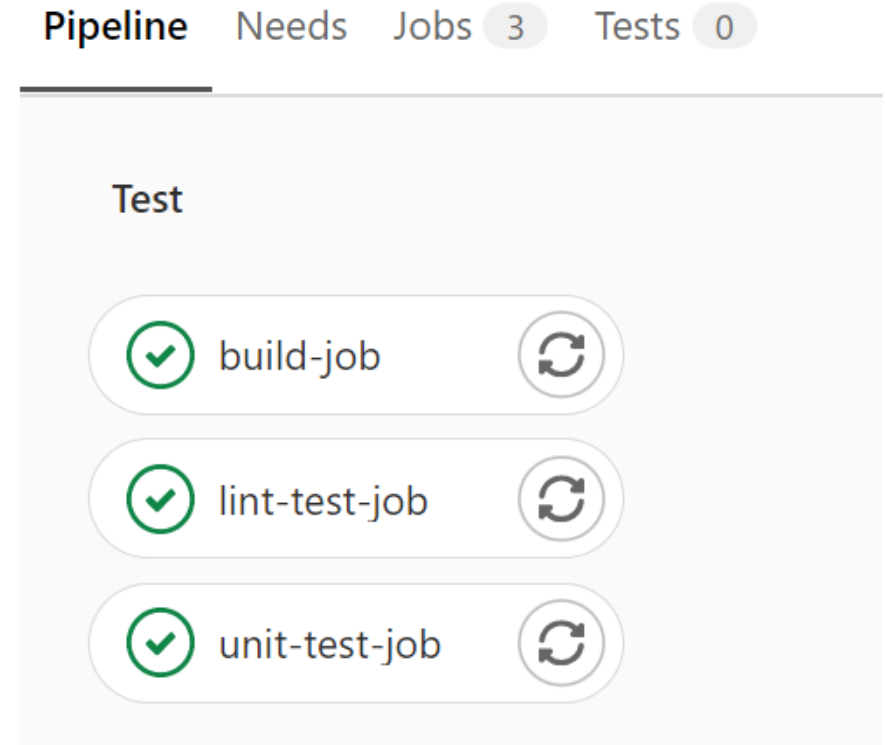
Filter pipelines

Show Pipeline ID

Status	Pipeline ID	Triggerer	Commit	Stages	Duration
<div>running</div>	<div>#71572</div> <div>latest</div>	<div></div>	<div>master 7c747622</div> <div>Update .gitlab-ci.yml ...</div>	<div></div>	<div>In progress</div>

Запуск задания

- Возвращаемся в раздел pipelines, видим, что работы запущены
- Можно зайти внутрь конвейера и посмотреть сами работы







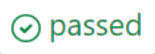





Запуск задания

- Так же их видно в разделе Jobs

devops > test2 > Jobs

All 31 Pending 0 Running 0 Finished 29

Status	Name	Job	Pipeline	Stage	Duration	Coverage
 passed	lint-test-job	#153533  master  7c747622 docker-linux	#71572 by 	test	 00:00:16  4 minutes ago	
 passed	unit-test-job	#153532  master  7c747622 docker-run	#71572 by 	test	 00:00:24  4 minutes ago	
 passed	build-job	#153531  master  7c747622 shell-runner	#71572 by 	test	 00:00:05  3 minutes ago	

Запуск задания

- Логи выполнения самого задания

```
✓ 8  Preparing environment 00:01
   9  Running on runner-kppnacs-w-project-7914-concurrent-0 via student-VirtualBox...
✓ 11  Getting source from Git repository 00:03
   12  Fetching changes with git depth set to 50...
   13  Reinitialized existing Git repository in /builds/devops/test2/.git/
   14  Checking out 7c747622 as master...
   15  Skipping Git submodules setup
✓ 17  Executing "step_script" stage of the job script 00:05
   18  Using docker image sha256:9c6f0724472873bb50a2ae67a9e7adcb57673a183cea8b06eb778dca85918
      1b5 for alpine:3.16 with digest alpine@sha256:bc41182d7ef5ffc53a40b044e725193bc10142a124
      3f395ee852a8d9730fc2ad ...
   19  $ echo "Linting code... This will take about 10 seconds."
   20  Linting code... This will take about 10 seconds.
   21  $ sleep 4
   22  $ echo "No lint issues found."
   23  No lint issues found.
✓ 25  Cleaning up project directory and file based variables 00:01
   27  Job succeeded
```

Duration: 16 seconds

Timeout: 1h (from project) ?

Runner: #94 (KPPnACsW) vbox-ubuntu-docker

Tags: docker-linux

Commit [7c747622](#) ?

Update .gitlab-ci.yml file

✓ Pipeline #71572 for master ?

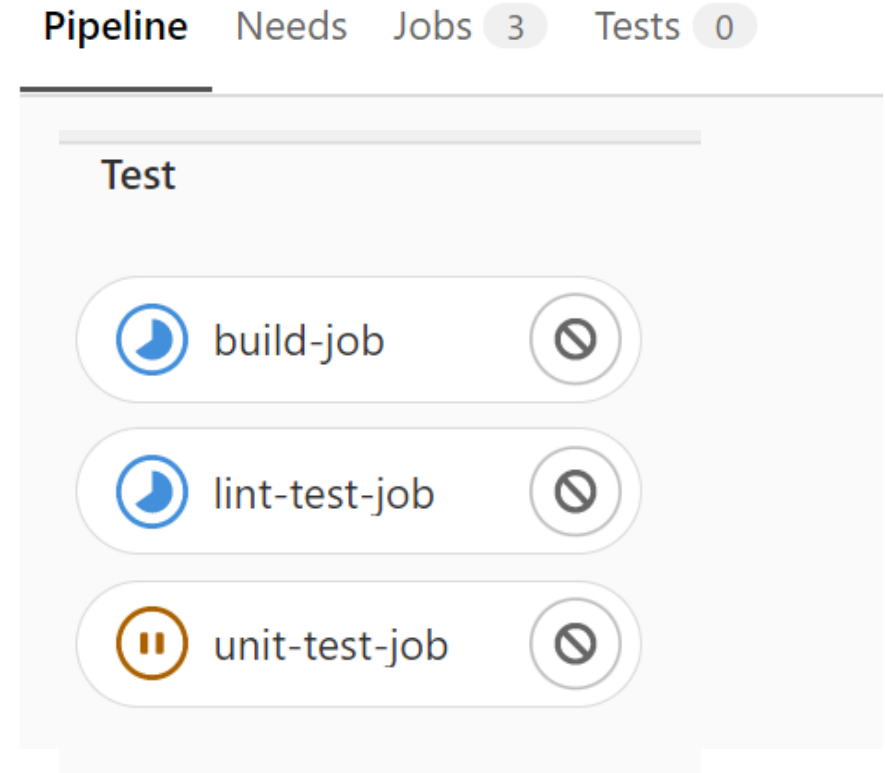
test

✓ build-job

→ ✓ lint-test-job

Запуск задания

- Можно заметить, что работы запустились одновременно
- Это явно не то, что мы хотели, т.к. нельзя протестировать до сборки и т.п.
- Хочется выстроить последовательную цепочку
 - Сборка => Тестирование => Упаковка



Настройка этапов

- Чтобы настроить порядок выполнения работ, нужно задать этапы
- Вспоминаем основные стадии
 - Сборка, тестирование, упаковка, заливка на staging сервер, заливка на production сервер
- Можно определять их столько, сколько нужно по необходимости
 - В файле `.gitlab-ci.yml` это будут stages

```
.gitlab-ci.yml 1.41 KB
1 # This file is a template, and might need editing
2 # This file is a template demonstrating the `scri
3 # Learn more about this keyword here: https://doc
4
5 # After committing this template, visit CI/CD > J
6 stages:
7   - build
8   - testing
9   - staging
10  - production
11
```

Настройка этапов

- Определяем некоторые обязательные параметры
 - К какому этапу относится задание

```
7  ∨ job:
8    stage: build
9  ∨ script:
10     # provide a shell script as argument for this keyword.
11     - echo "Hello World"
12
```

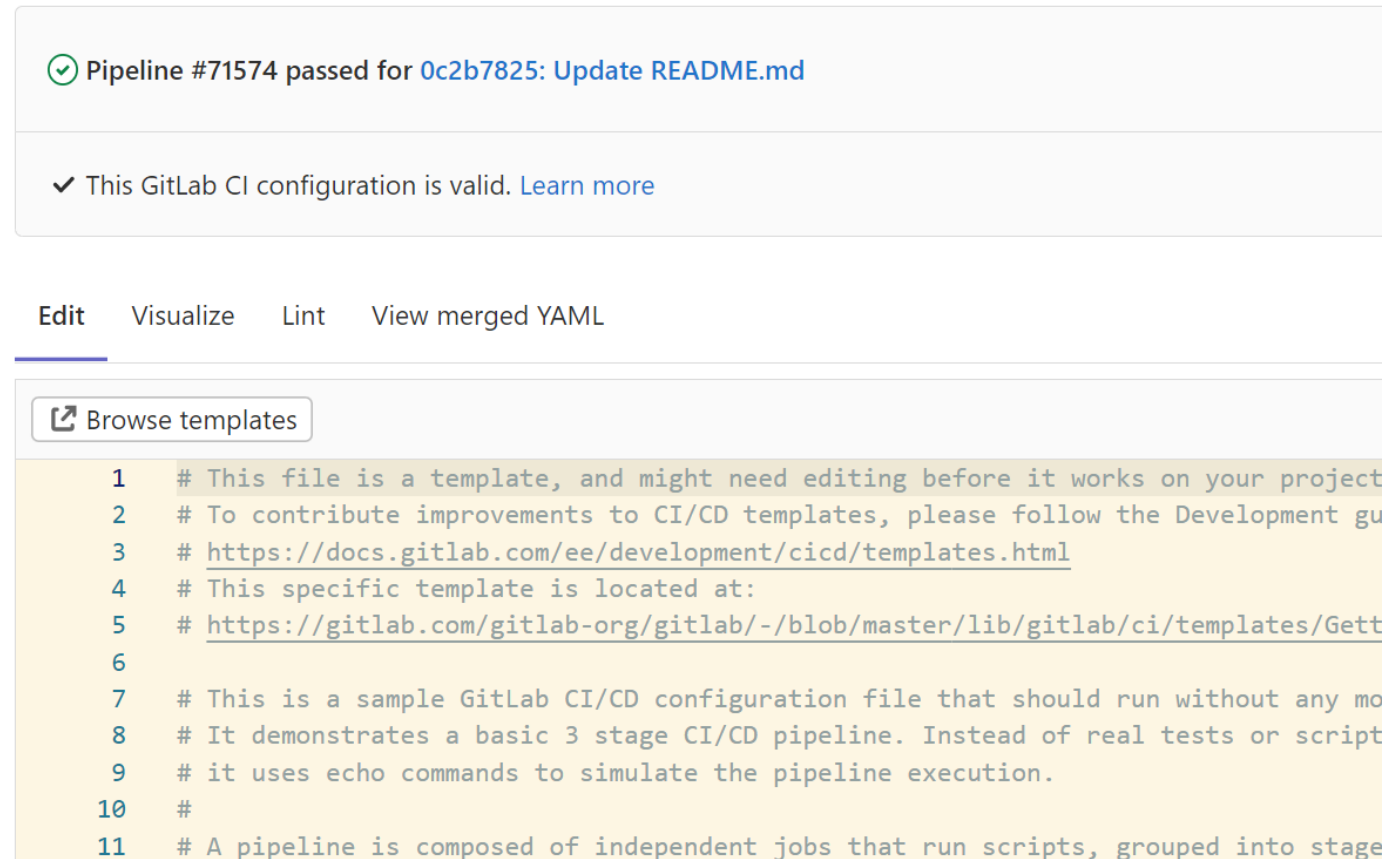
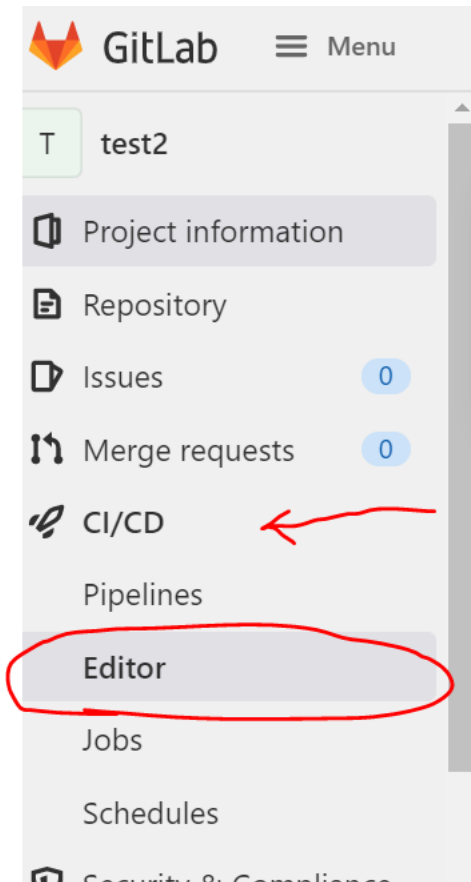
- Задания будут выполняться по этапам

Настройка этапов

- Позволяют настроить последовательность запуска заданий
 - Т.е. например сначала произойдет сборка, потом тестирование, потом сборка образа, а потом уже отправка образа в репозиторий
- Позволяют контролировать ход выполнения
 - Если на одном этапе задание не выполнится, конвейер остановится (остальные выполняться не будут)
- Позволяет группировать задания одного этапа
 - Разные виды тестирования


Pipeline editor


- Отдельный интерфейс для настройки pipeline через веб-приложение




Pipeline editor

- Кроме собственно редактирования ci файла дает дополнительные возможности

 Pipeline #71574 passed for [0c2b7825: Update README.md](#)

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

Edit Visualize Lint View merged YAML

 Browse templates

```
1 # This file is a template, and might need editing before it works on your project
2 # To contribute improvements to CI/CD templates, please follow the Development gu
3 # https://docs.gitlab.com/ee/development/cicd/templates.html
4 # This specific template is located at:
5 # https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Gett
6
7 # This is a sample GitLab CI/CD configuration file that should run without any mo
8 # It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or script
9 # it uses echo commands to simulate the pipeline execution.
10 #
11 # A pipeline is composed of independent jobs that run scripts, grouped into stage
```

Pipeline editor

- Кроме собственно редактирования ci файла дает дополнительные возможности

Быстрый доступ и информация о последнем запуске конвейера

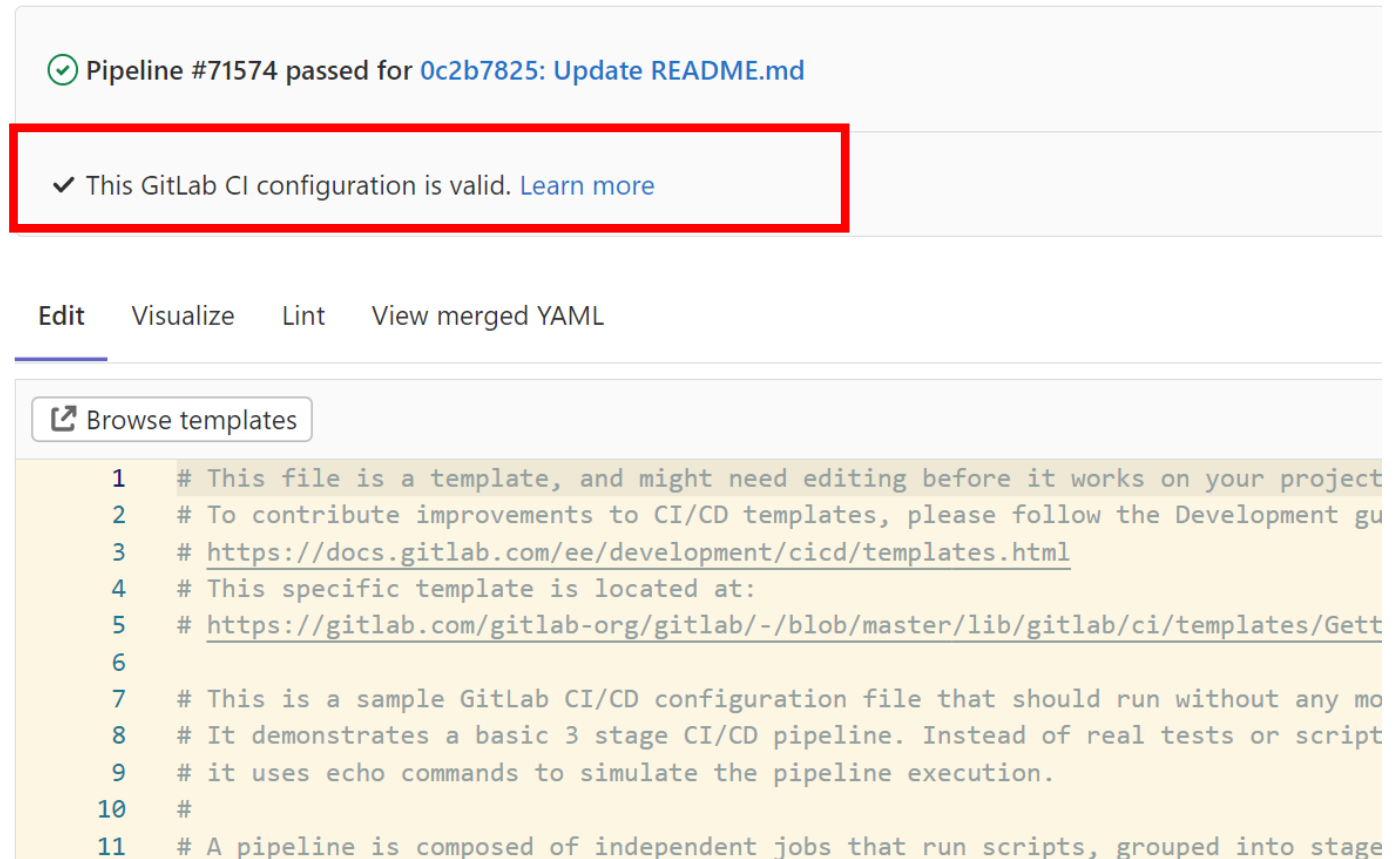
The screenshot displays the GitLab Pipeline Editor interface. At the top, a green checkmark icon and the text "Pipeline #71574 passed for 0c2b7825: Update README.md" are highlighted with a red rectangle. Below this, another green checkmark icon and the text "This GitLab CI configuration is valid. [Learn more](#)" are visible. The interface includes tabs for "Edit", "Visualize", "Lint", and "View merged YAML", with "Edit" being the active tab. A "Browse templates" button is located above a code editor. The code editor contains a sample GitLab CI/CD configuration file with the following content:

```
1 # This file is a template, and might need editing before it works on your project
2 # To contribute improvements to CI/CD templates, please follow the Development gu
3 # https://docs.gitlab.com/ee/development/cicd/templates.html
4 # This specific template is located at:
5 # https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Gett
6
7 # This is a sample GitLab CI/CD configuration file that should run without any mo
8 # It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or script
9 # it uses echo commands to simulate the pipeline execution.
10 #
11 # A pipeline is composed of independent jobs that run scripts, grouped into stage
```

Pipeline editor

- Кроме собственно редактирования ci файла дает дополнительные возможности

Быстрая проверка
конфигурации и
синтаксиса ci yaml
файла



✓ Pipeline #71574 passed for 0c2b7825: Update README.md

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

Edit Visualize Lint View merged YAML

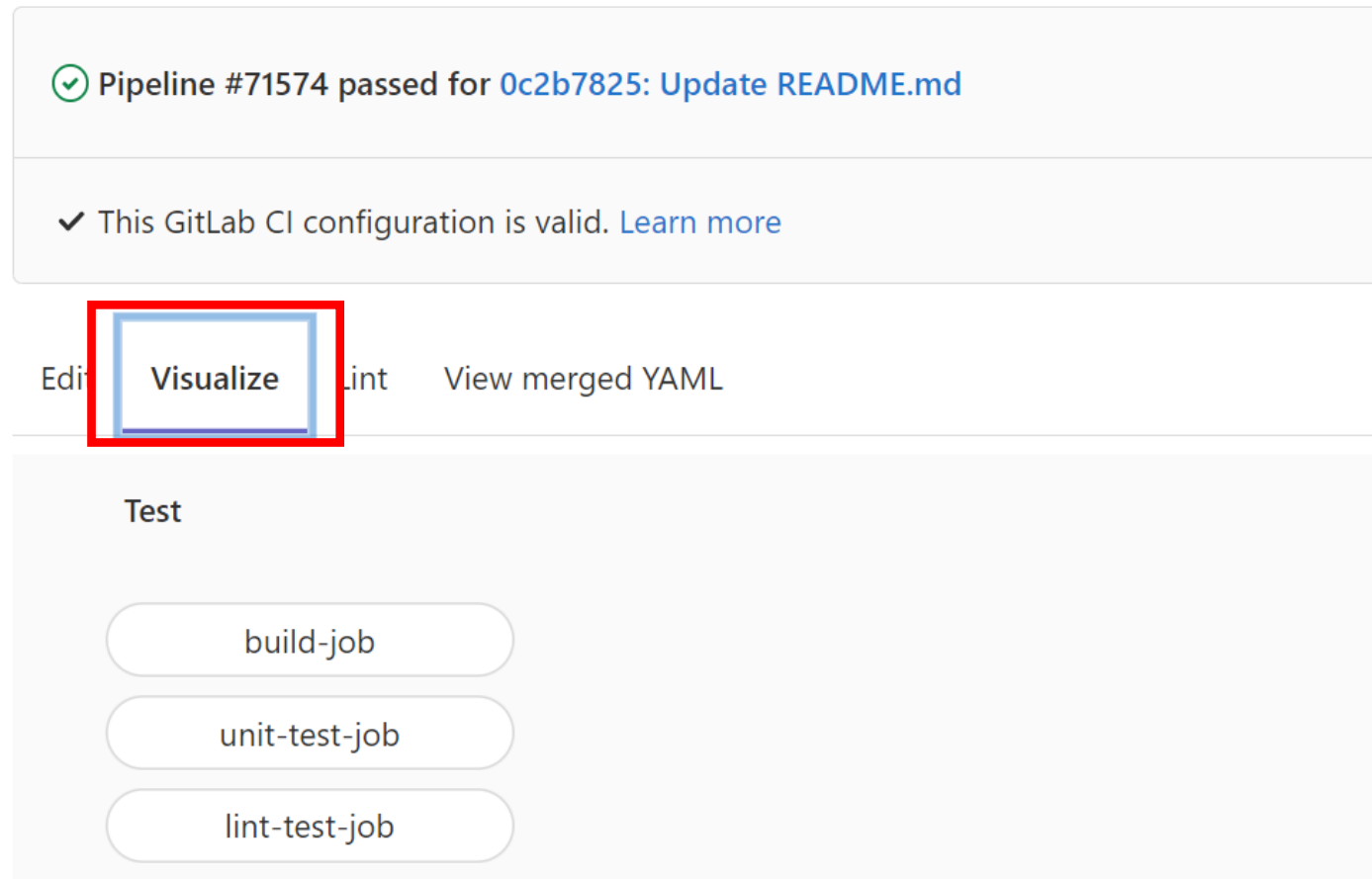
[Browse templates](#)

```
1 # This file is a template, and might need editing before it works on your project
2 # To contribute improvements to CI/CD templates, please follow the Development gu
3 # https://docs.gitlab.com/ee/development/cicd/templates.html
4 # This specific template is located at:
5 # https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Gett
6
7 # This is a sample GitLab CI/CD configuration file that should run without any mo
8 # It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or script
9 # it uses echo commands to simulate the pipeline execution.
10 #
11 # A pipeline is composed of independent jobs that run scripts, grouped into stage
```

Pipeline editor

- Кроме собственно редактирования ci файла дает дополнительные возможности

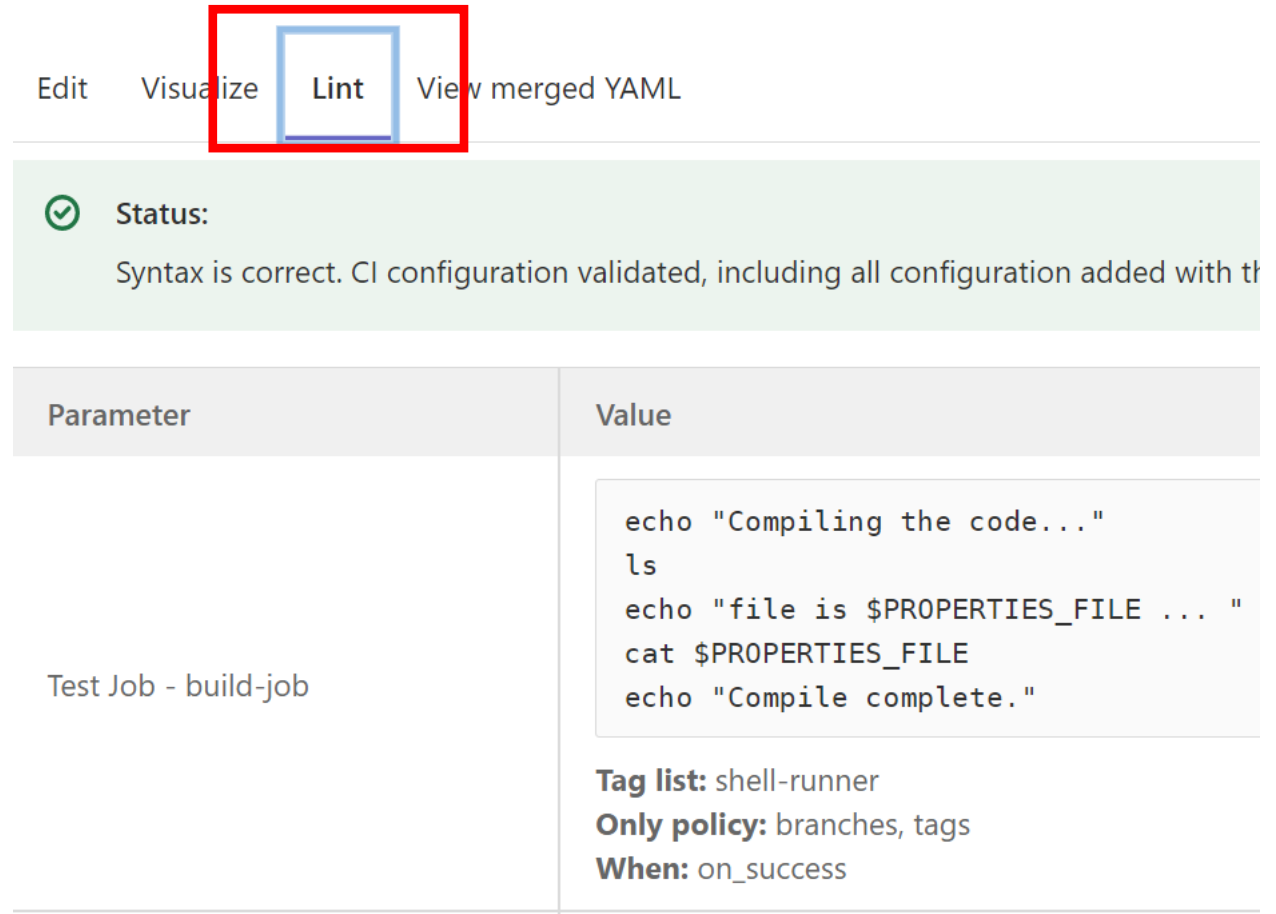
Визуализация
конвейера



Pipeline editor

- Кроме собственно редактирования ci файла дает дополнительные возможности

Альтернативный вариант просмотра с дополнительной информацией



The screenshot shows the 'Lint' tab selected in the Pipeline editor. The 'Lint' button is highlighted with a red rectangle. Below the tabs, a green status bar indicates that the syntax is correct. A table below shows the configuration for a job named 'Test Job - build-job'.

Parameter	Value
Test Job - build-job	<pre>echo "Compiling the code..." ls echo "file is \$PROPERTIES_FILE ... " cat \$PROPERTIES_FILE echo "Compile complete."</pre> <p>Tag list: shell-runner Only policy: branches, tags When: on_success</p>

Настройка этапов

- Создадим стадии build, test, package, deploy
- Распределим задания по стадиям
 - На каждой стадии может быть несколько заданий
 - Build
 - build-job
 - Test
 - Unit-test-job
 - Lint-test-job
 - Package
 - build-image
 - push-image

Pipeline editor

- Получится вариант, разбитый на этапы выполнения

✓ Pipeline #71574 passed for 0c2b7825: Update README.md

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








Edit **Visualize** Lint View merged YAML

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

Pipeline editor

- Сохраним изменения и посмотрим, как изменится запуск
 - Видно, что добавились стадии

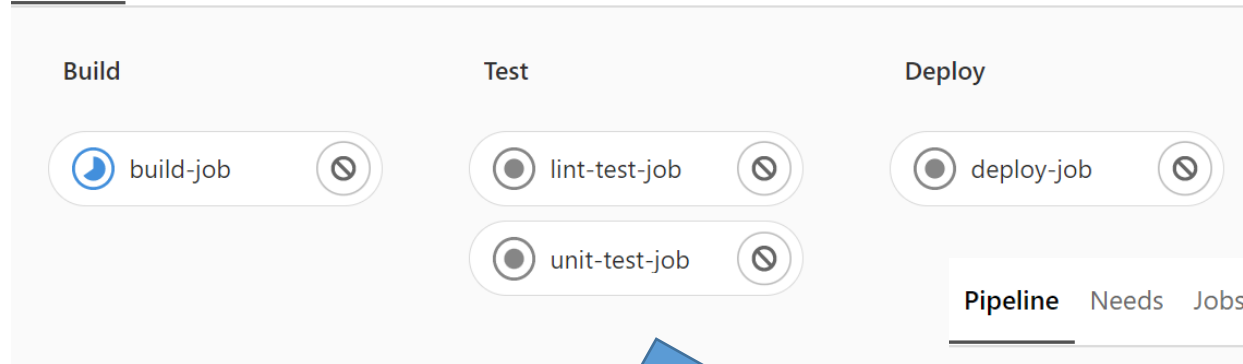
devops > test2 > Pipelines

All 18 Finished Branches Tags				Clear runner caches	CI lint	Run pipeline
Filter pipelines				Q	Show Pipeline ID v	
Status	Pipeline ID	Triggerer	Commit	Stages	Duration	
 pending	#71597 latest		 master  b3f60c97  Update .gitlab-ci.yml ...		 In progress	

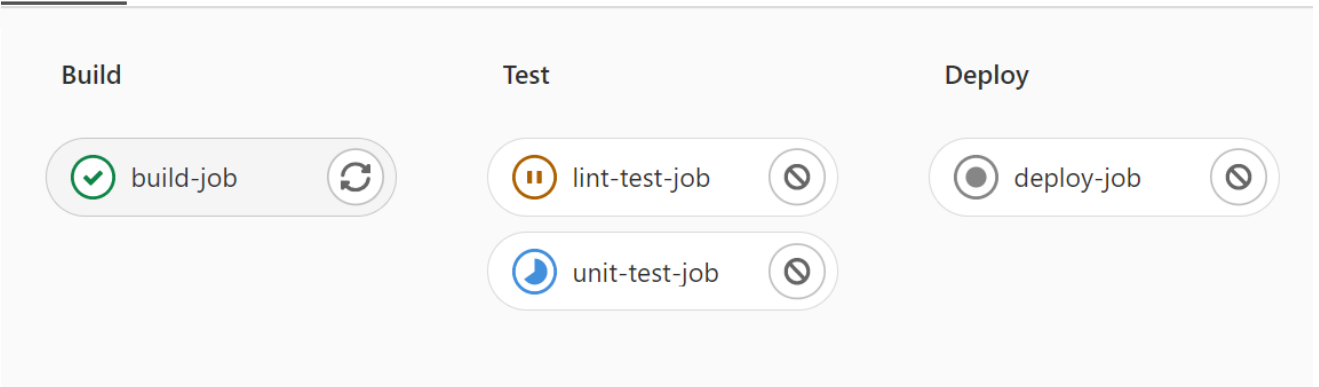
Pipeline editor

- Если открыть информацию о pipeline, видно, что работы запускаются по этапам

Pipeline Needs Jobs 4 Tests 0



Pipeline Needs Jobs 4 Tests 0



Что дальше?

- Кстати, а где именно они запускаются?
- Кто выполняет наши задания?
- Какие команды можно указывать в заданиях?
- Чтобы ответить, необходимо немного разобраться с архитектурой gitlab