



Платформа для сборки и развертывания
приложений GitLabCI

Технологии DevOps

Ровнягин Михаил Михайлович

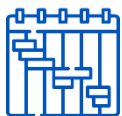


GitLab как платформа DevOps

- GitLabCI - это полноценная платформа для DevOps



Платформа DevOps, которая «умеет»:



Управлять



Планировать



Создавать



Проверять



Упаковывать



Защищать



Выпускать

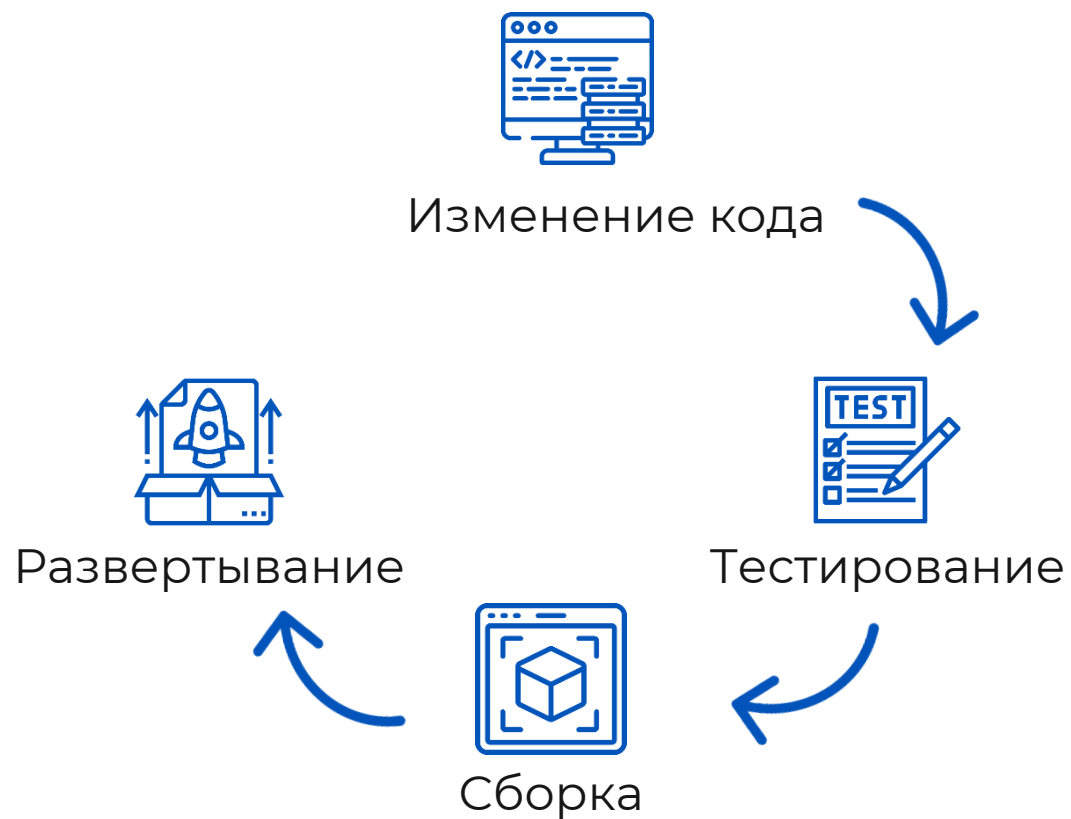
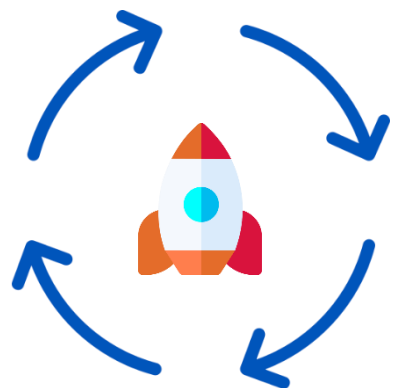


Конфигурировать



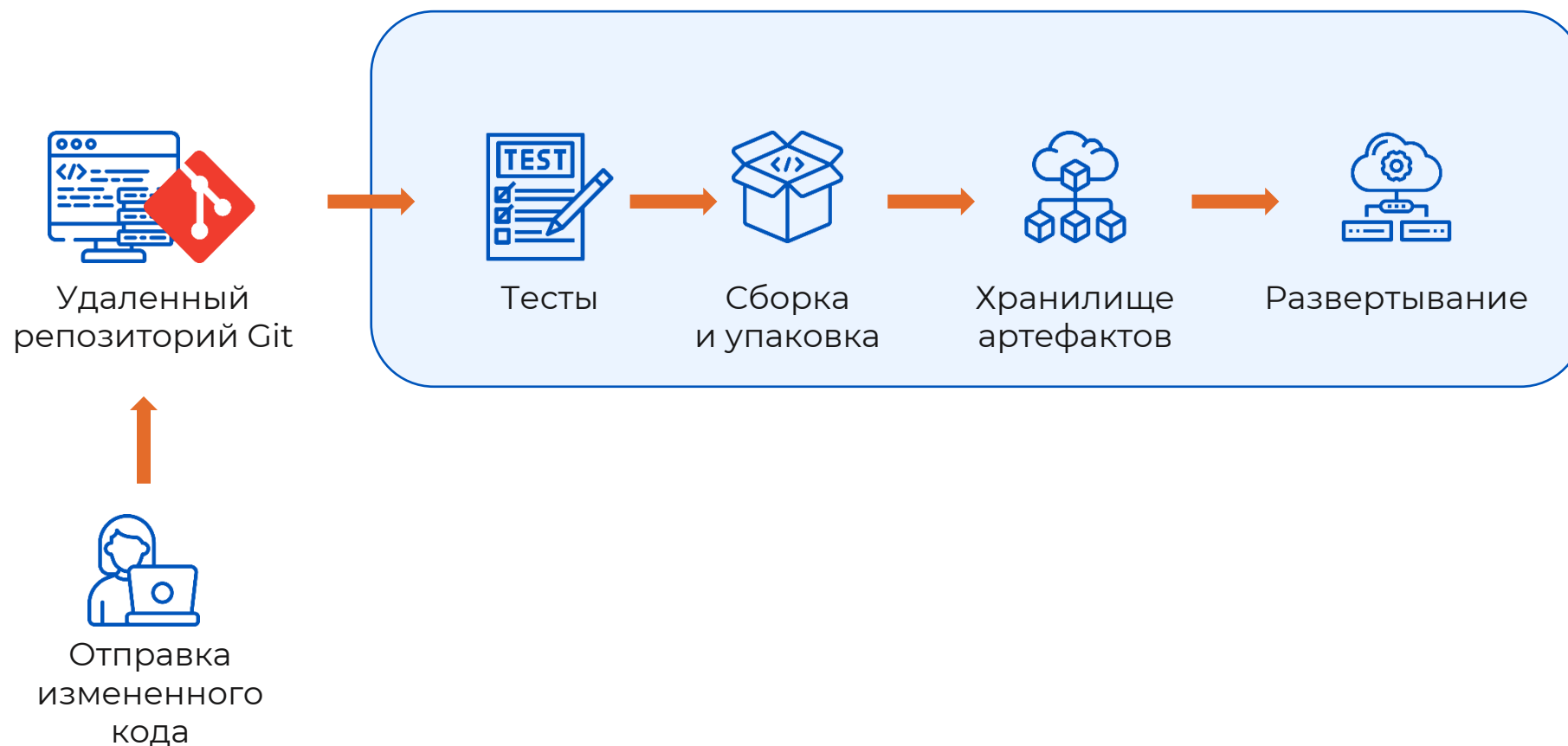
Мониторить

Автоматически и непрерывно

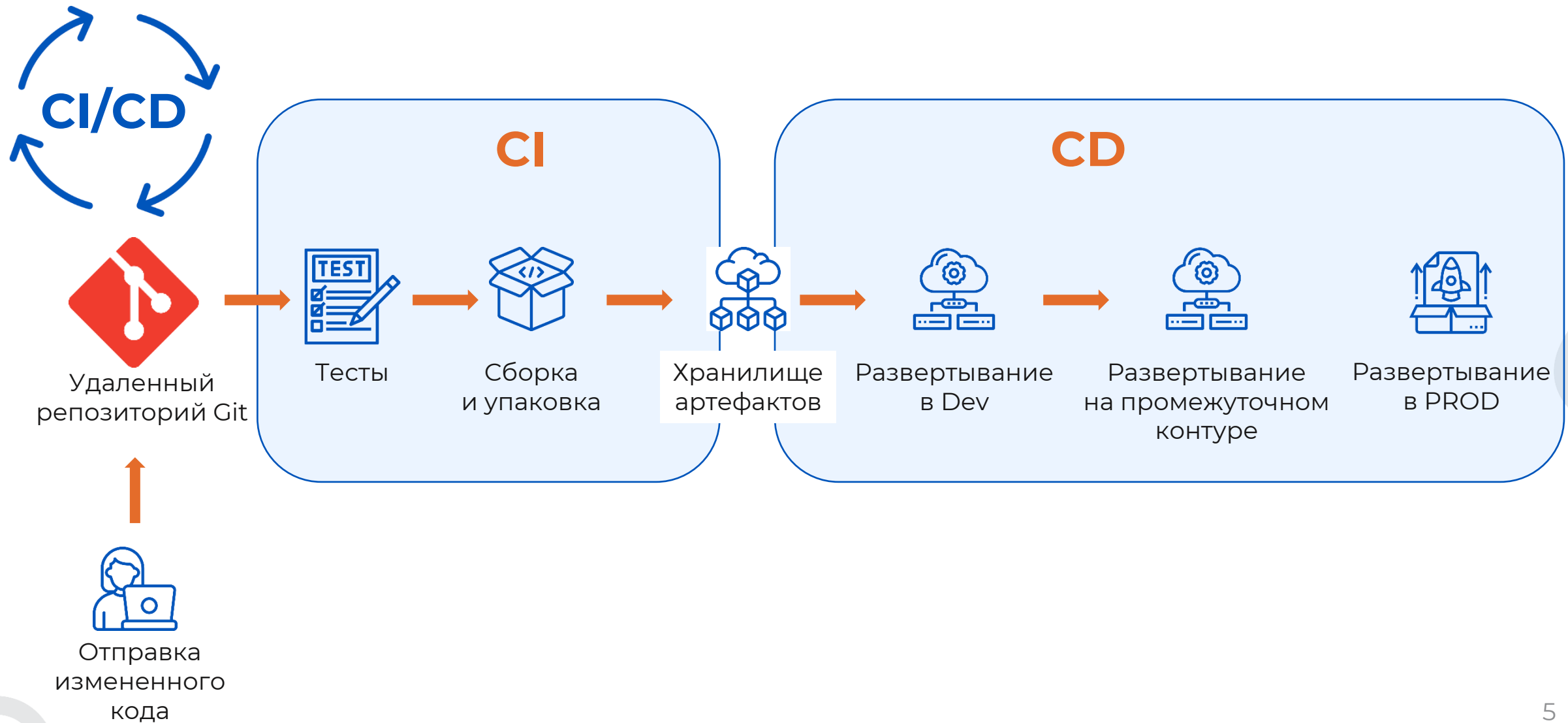




- GitLab выполняет конвейер (pipeline), который задал разработчик
- CI/CD процесс необходим для доставки изменений конечному пользователю



Непрерывная доставка изменений до среды эксплуатации



Архитектура GitLab

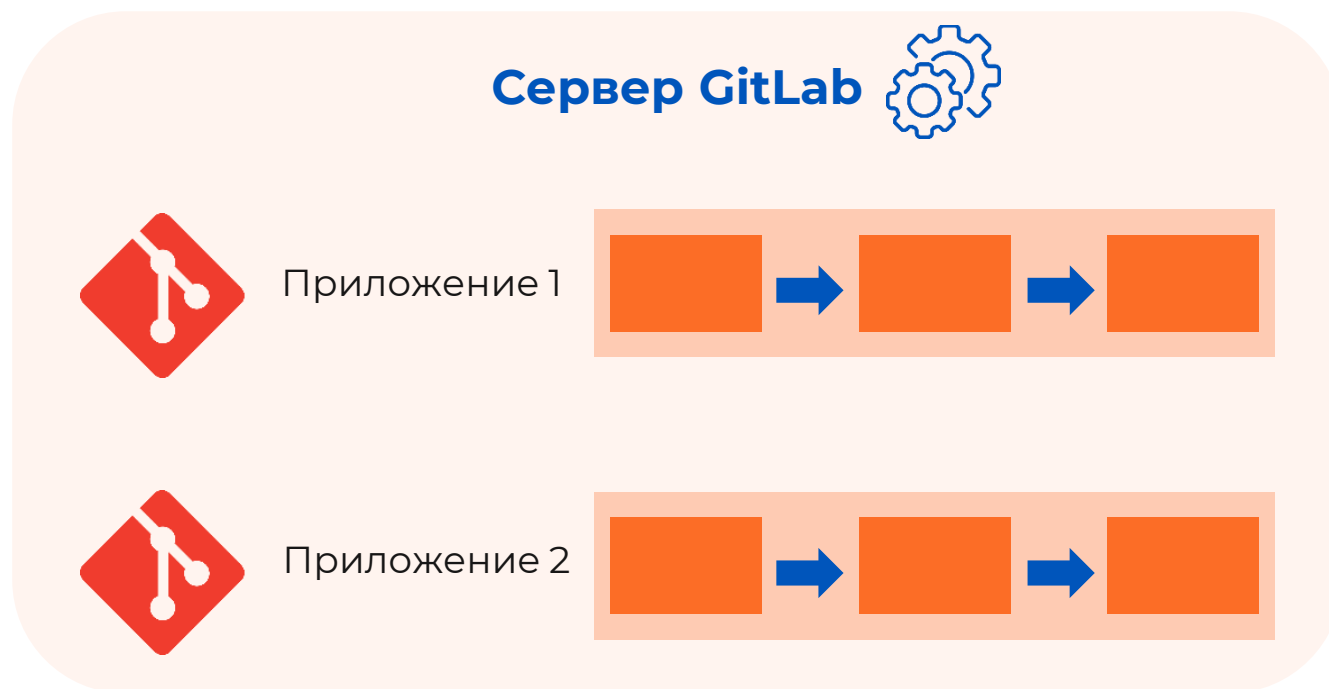
- Хранит код вашего приложения и конфигурацию конвейера
- Содержит внутренние конфигурации GitLab и т.д.
- Управляет выполнением конвейера



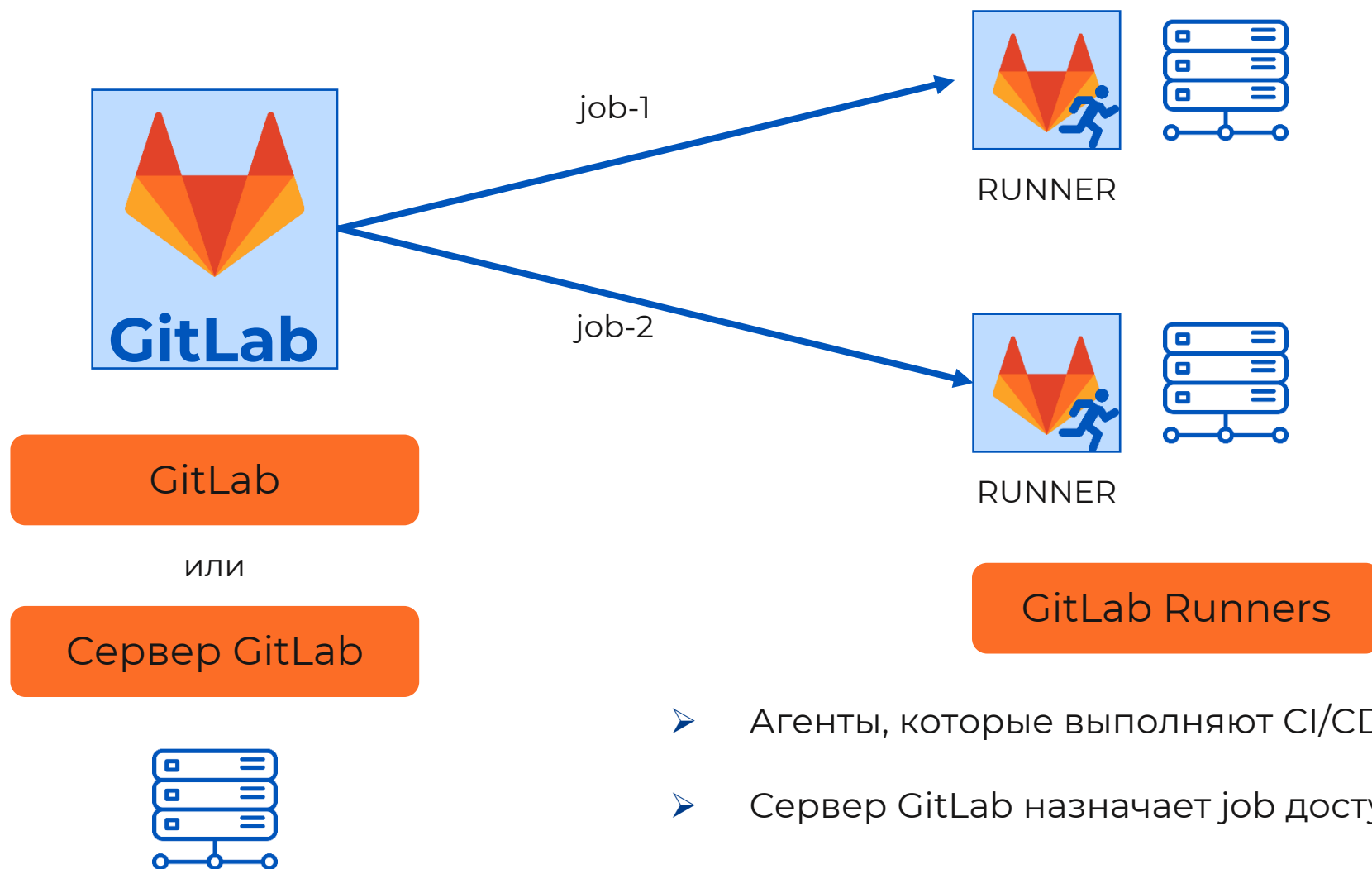
GitLab

или

Сервер GitLab

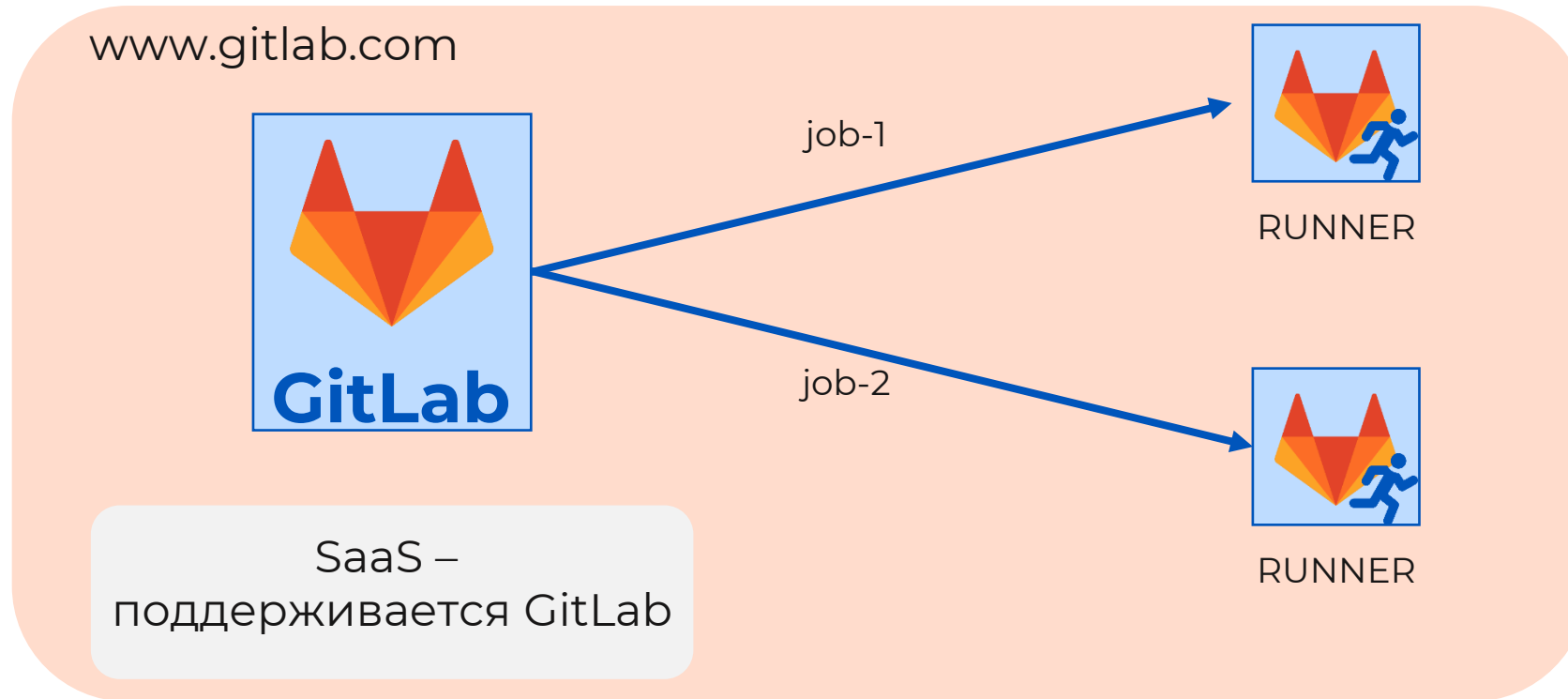


Архитектура GitLab



- Агенты, которые выполняют CI/CD job'ы
- Сервер GitLab назначает job доступным исполнителям

Архитектура GitLab

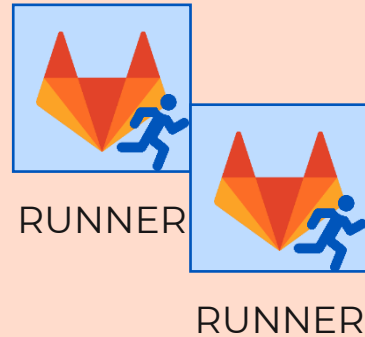


- GitLab предлагает также несколько runners, также поддерживаемых GitLab
- Это runners, доступные всем пользователям на `www.gitlab.com`

Архитектура GitLab

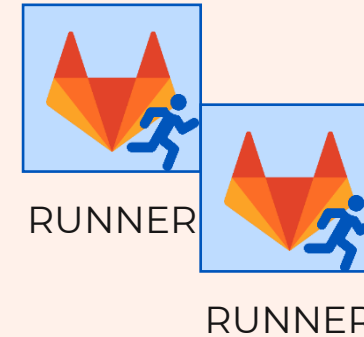


www.gitlab.com



SaaS –
поддерживается GitLab

www.gitlab.mycompany.com



Самоуправляемый



- Подключаются собственные runners GitLab
- Создается собственный экземпляр GitLab

Архитектура GitLab

Для выполнения соответствующего процесса необходимо установить менеджер пакетов или средство сборки для этого конкретного языка программирования

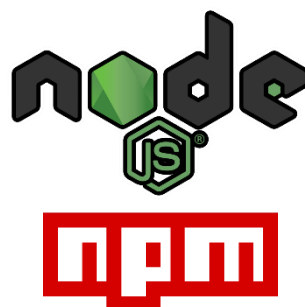
Run Python App

установлен pip



Run Node.js App

установлен npm или yarn



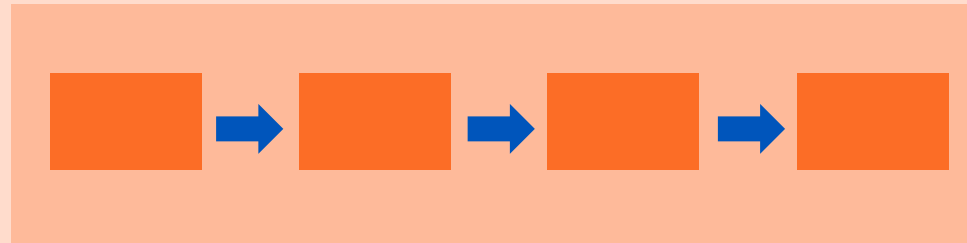
Run Java App

установлен maven или gradle



Представление конвейера

Pipeline



➤ Конвейер написан в виде кода

➤ Размещен внутри репозитория
git приложения

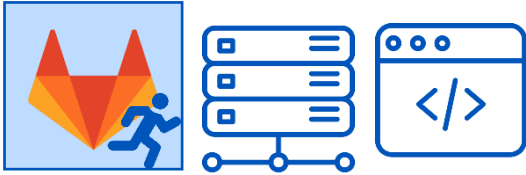
➔ Вся конфигурация CI/CD записана в YAML



gitlab-ci.yml

Различные типы исполнителей (Executor)

Исполнитель определяет среду, в которой выполняется каждое задание



RUNNER

Shell Executor

- **Shell** – самый простой исполнитель
- Команды, выполняемые в операционной системе
- В оболочке сервера, на котором установлен GitLab Runner

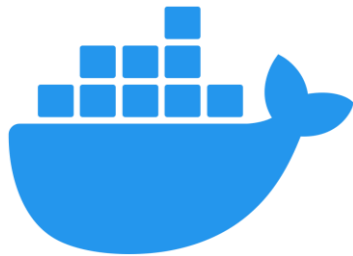
Различные типы исполнителей (Executor)



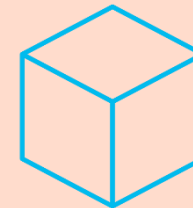
Исполнитель определяет среду, в которой выполняется каждое задание



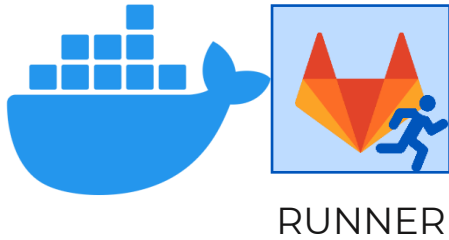
RUNNER



- Команды выполняются внутри контейнера
- Необходимо установить только сам **Docker**
- Каждый Job выполняется в отдельном изолированном контейнере

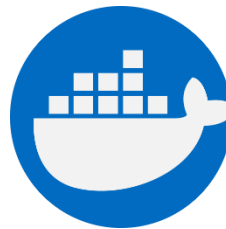


Какой образ Docker используется?

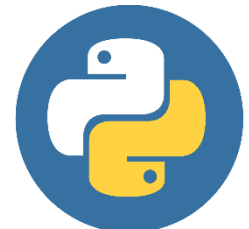


По умолчанию: GitLab Runners используют образ Ruby для запуска контейнера

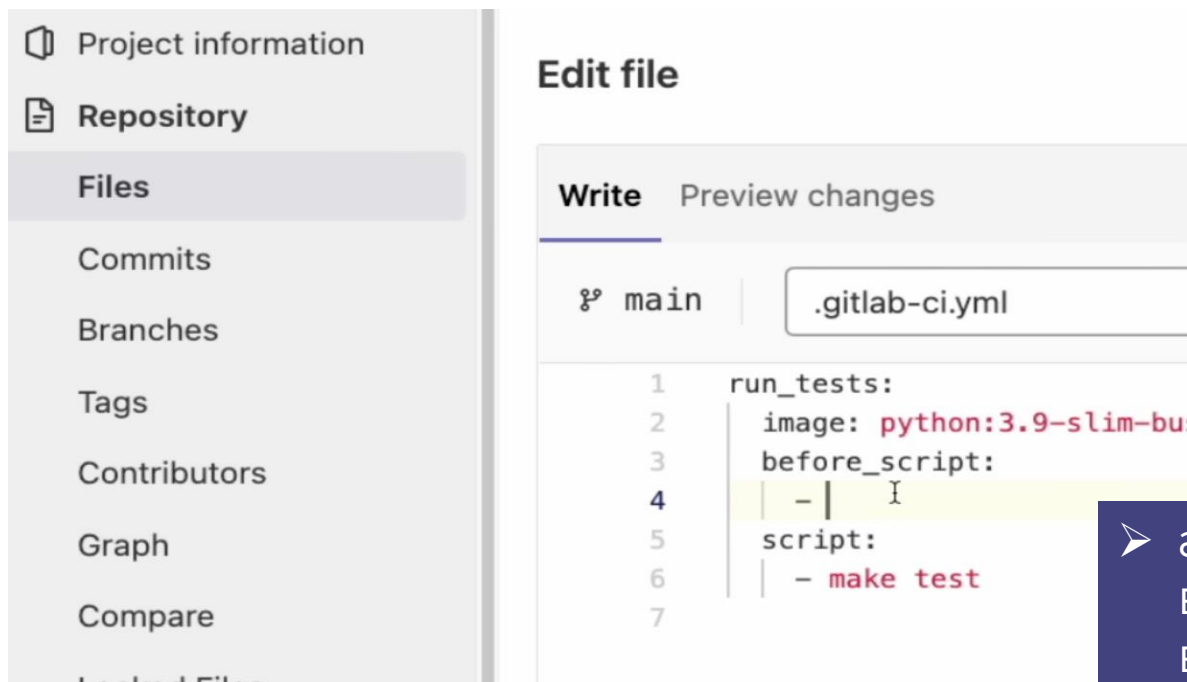
Образ можно настроить, выбрав, например, Python, а не Ruby



Запуск Python-тестов



GitLab – пример .gitlab-ci.yml файла

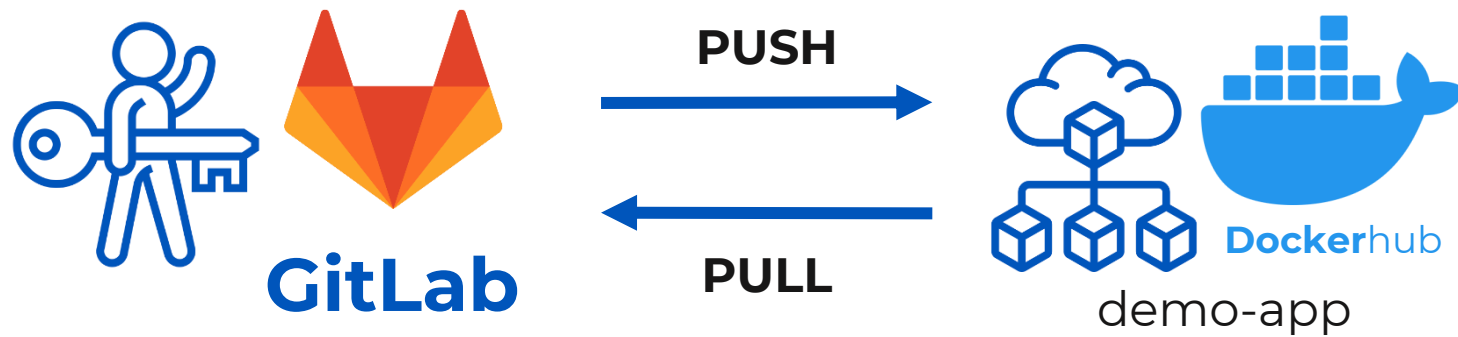


before_script - команды, которые должны выполняться перед командой script

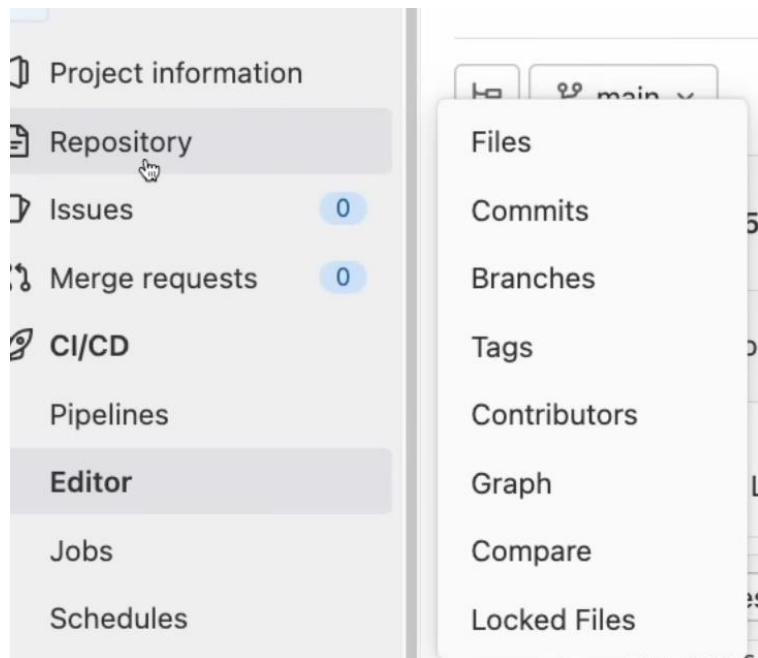
➤ after_script - команды, которые выполняются после каждого задания, включая неудачные задания

DockerHub/локальный registry

GitLab нужны учетные данные репозитория!



У вас могут быть разные роли



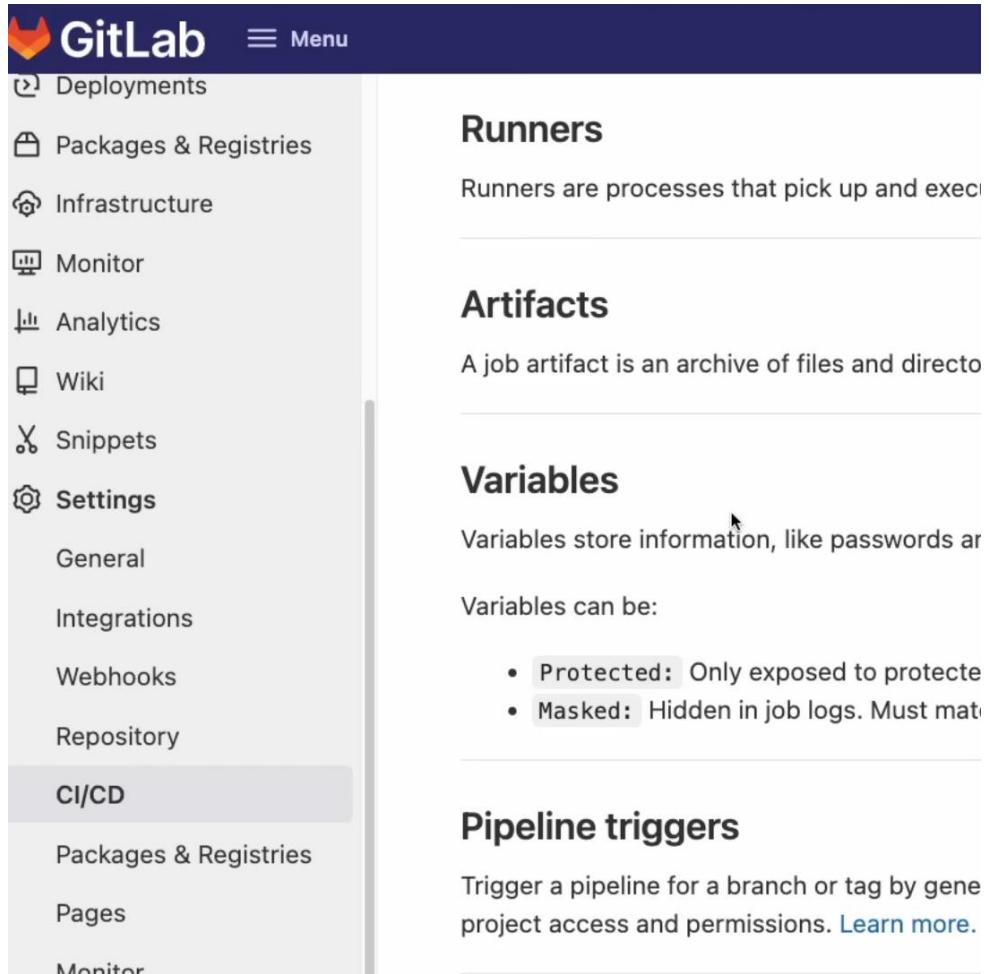
Администратор

- Администрирование и управление настройками
- Регистрирует runnerGitLab и т.д.

Пользователь

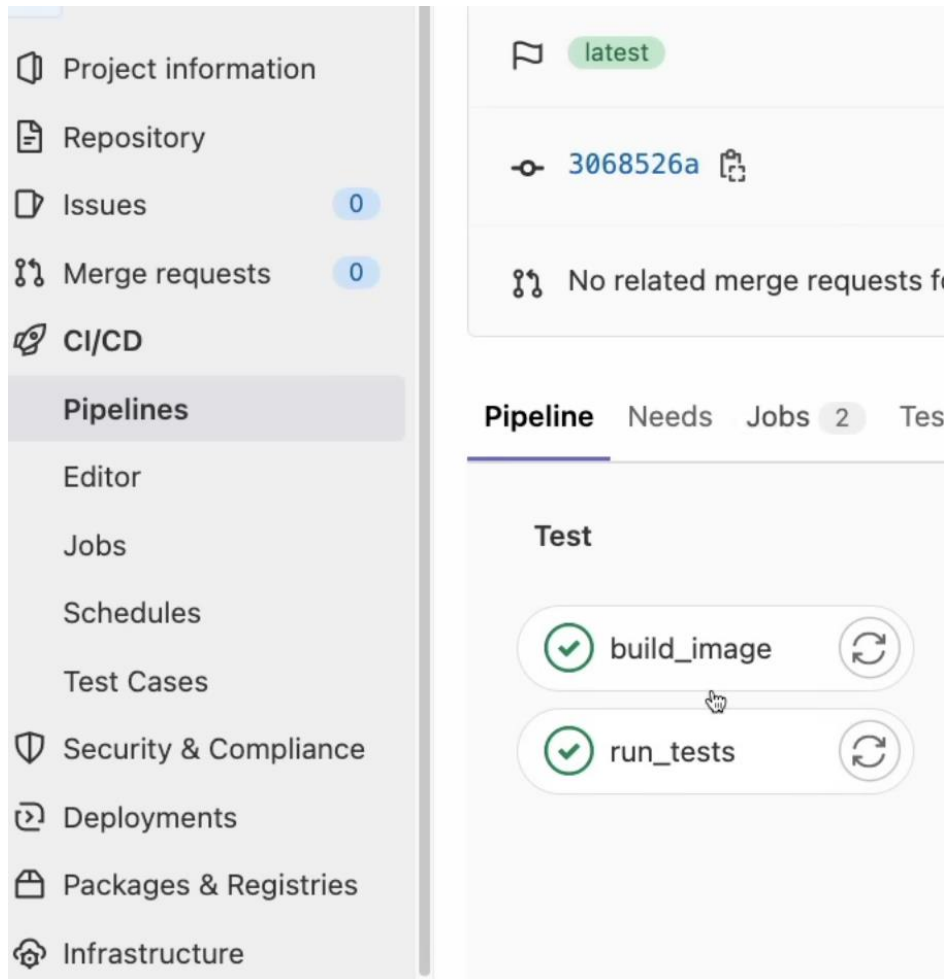
- Доступ к registry
- Написание CI/CD- сценария

Переменные проекта



The screenshot shows the GitLab web interface. The top navigation bar includes the GitLab logo and a 'Menu' button. The left sidebar contains a list of navigation items: Deployments, Packages & Registries, Infrastructure, Monitor, Analytics, Wiki, Snippets, Settings, General, Integrations, Webhooks, Repository, CI/CD (highlighted), Packages & Registries, Pages, and Monitor. The main content area is titled 'Runners' and 'Artifacts'. Below these, the 'Variables' section is active, showing a description: 'Variables store information, like passwords ar'. Below this, it states 'Variables can be:' followed by a list of two types: 'Protected: Only exposed to protecte' and 'Masked: Hidden in job logs. Must mat'. The 'Pipeline triggers' section is also visible at the bottom, with a description: 'Trigger a pipeline for a branch or tag by gene project access and permissions. [Learn more.](#)'

- Хранятся вне репозитория git (не в файле `.gitlab-ci.yml`)
- Идеально подходит для токенов и паролей, которые не должны быть включены в репозиторий по соображениям безопасности



stage

run_tests

build_image

deploy

➤ Несколько заданий на одном этапе выполняются одновременно. Конкретное исполнение не гарантировано



Stage 1

run_tests

Stage 2

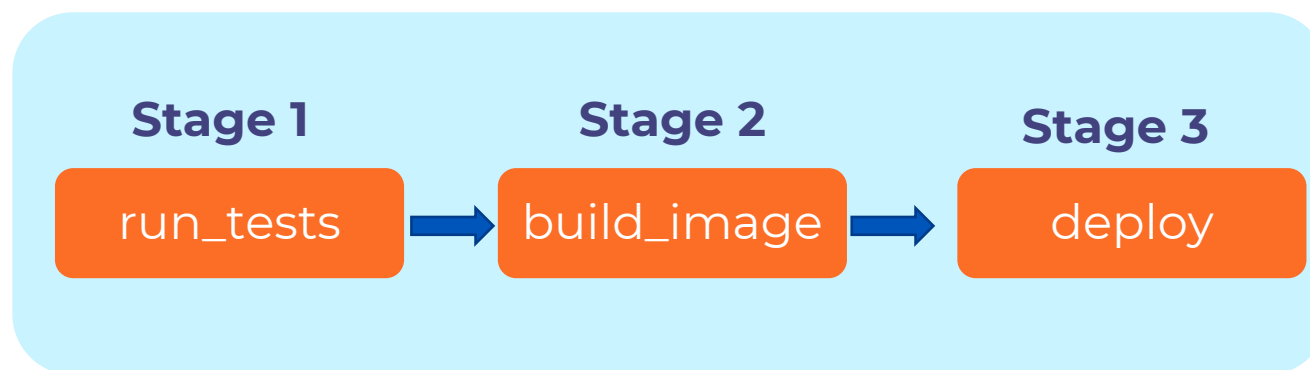
build_image

Stage 3

deploy



Порядок выполнения заданий



- Docker-образ создается только в том случае, если тесты пройдут успешно
- Развертывание проводится только в том случае, если сборка и публикация в registry прошли успешно

- Несколько Job можно сгруппировать в **stages**, которые выполняются **в определенном порядке**
 - Логическая группировка заданий, которые принадлежат друг другу

- Несколько Job (заданий) на одном **stages** выполняются **параллельно**
 - Только после того, как все Job (например, все тесты) будут выполнены успешно, запустится следующий **stage**

test

unit_tests

lint_tests

..._tests

- Различные этапы тестов могут выполняться параллельно, чтобы ускорить конвейер





МИФИ

Национальный
исследовательский
ядерный университет