



# CI/CD на основе GitLab

Создание и настройка проектов. Тюнинг  
GitLab Runner



Проверить, идет ли запись

# Меня хорошо видно & слышно?



Ставим "+", если все хорошо  
"-", если есть проблемы

Тема вебинара

# Создание и настройка проектов. Тюнинг GitLab Runner



**Эрик Арайс**

преподаватель курса администрирования Linux и DevOps

7 лет опыта работы системным администратором Linux  
DevOps-инженер в компании "Сигма"

telegram: @arais\_erick



# Правила вебинара



Активно  
участвуем



Off-topic обсуждаем  
в учебной группе  
**#канал группы**



Задаем вопрос  
в чат или ГОЛОСОМ



Вопросы вижу в чате,  
могу ответить не сразу

## Условные обозначения



Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# Карта курса



**СТАРТ**

CI/CD - системы,  
подходы и workflow

Gitlab CI

Безопасность

Проектная работа

**ФИНИШ**



# Маршрут вебинара

Вводная часть

Группы, подгруппы и проекты в Gitlab

Создание и настройка проектов в Gitlab

Настройка и тюнинг GitLab Runner

Рефлексия



# Цели вебинара

К концу занятия вы сможете

1. Создать проект, группу и подгруппу в Gitlab

2. Ориентироваться в основной части интерфейса Gitlab

3. Выполнить базовую настройку Gitlab Runner

4. Запустить простой пайплайн CI/CD



# СМЫСЛ

## Это нужно уметь чтобы:

1. Понимать, как организована командная работа посредством Gitlab

2. Уверенно ориентироваться в самом часто используемом функционале Gitlab

3. Предлагать командам более эффективную организацию совместной работы за счет создания оптимальной структуры групп, подгрупп и проектов

4. Приобрести навык развертывания и настройки Gitlab Runner, который востребован у современного разработчика или devops-инженера

# Создание и настройка проектов. Тюнинг GitLab Runner

# Краткое напоминание

Для успешного прохождения этого занятия ранее нужно было сделать:

1. Создать и запустить **Managed Service for Gitlab** в облаке Yandex.Cloud
2. Создать и запустить отдельную **Virtual Machine Ubuntu** для Gitlab Runner там же  
*(в той же виртуальной подсети и зоне, чтобы они видели друг друга)*
  - a. На VM развернуть Docker (для работы Gitlab Runner с Docker Executor)  
по [этой инструкции](#), плюс [дополнительная настройка](#)
  - b. На VM развернуть Gitlab Runner по [этой инструкции](#)

# Вопросы для проверки

**Получилось ли у вас развернуть Gitlab и Gitlab Runner?**

Напишите в чат одну цифру с ответом:

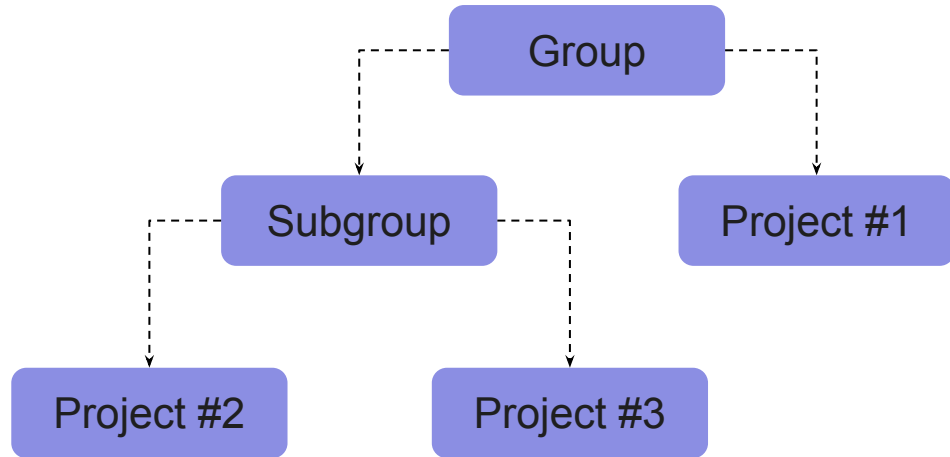
0. - еще не сделал / не было времени
1. - начал делать / в процессе
2. - частично сделал / возникли сложности
3. - да, задеплоил и то и другое, все готово
4. - уже запустил CI/CD пайплайн и он успешно выполнен!



# Создание и настройка проектов

# Основные сущности проектов в Gitlab

1. Namespace
2. Group
3. Subgroup
4. Project



# Пространства имен

Пространства имен (namespaces) используются для организации и группировки проектов. Типы namespaces:

## 1. Пользовательские пространства имен (User namespaces):

- Пользователь GitLab имеет свое личное пространство имен
- URL вида: [gitlab.com/username/project-name](https://gitlab.com/username/project-name)

## 1. Групповые пространства имен (Group namespaces):

- Для организации проектов, связанных с группой или командой
- URL вида: [gitlab.com/group-name/project-name](https://gitlab.com/group-name/project-name)

## 1. Подгрупповые пространства имен (Subgroup namespaces):

- Подгруппы внутри групп для дополнительной иерархической организации (необязательный элемент)
- URL вида: [gitlab.com/parent-group/subgroup/project-name](https://gitlab.com/parent-group/subgroup/project-name)

# Группы

Инструмент для организации проектов и управления доступом. Основные функции:

1. **Упорядочивание:** организация проектов в иерархию
2. **Управление доступом:** права доступа для всех проектов в группе и назначение ролей пользователям на уровне группы
3. **Совместное использование ресурсов:** Общие CI/CD переменные, общие Docker-реестры и пакеты
4. **Совместная работа:** Групповые доски задач (issue boards) для управления задачами по нескольким проектам, групповые вики для общей документации.
5. **Отчетность и аналитика:** Групповые панели мониторинга для отслеживания активности в проектах, статистика и метрики по группе
6. **Безопасность:** Централизованное управление настройками безопасности
7. **Администрирование:** Единое управление большим числом проектов через группы
8. **Интеграции:** Настройка интеграций на уровне группы, которые автоматически применяются ко всем проектам

# Подгруппы

Инструмент для организации гибкой информационной структуры.

Функции подгрупп:

1. Дополнительная иерархическая организация проектов
2. Детальное разделение доступа на уровне подгруппы
3. Улучшенная навигация и поиск по проектам
4. Изоляция проектов
5. Наследование настроек от родительских групп
6. Гибкость в управлении ресурсами, назначение квот и лимитов для подгрупп
7. Отдельные рабочие пространства для разных команд или проектов
8. Улучшенная аналитика на уровне подгруппы
9. Упрощение миграции проектов между группами

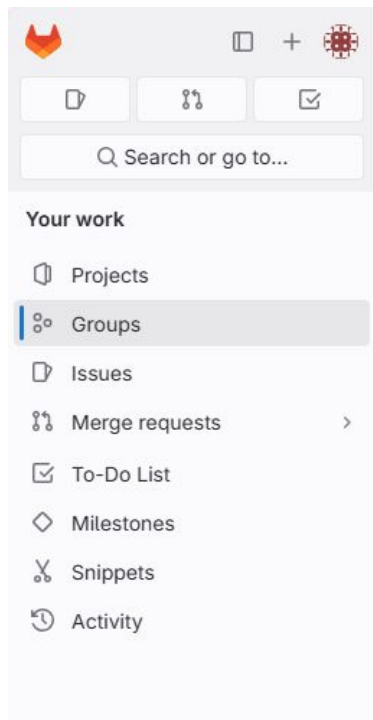
# Проекты

Проекты – это основные структурные элементы Gitlab, включающие комплексную среду для управления жизненным циклом разработки, организации кода и совместной работы:

1. Хранение кода в репозитории Git
2. Управление задачами (Issues)
3. Слияние кода (Merge Requests)
4. CI/CD пайплайны
5. Вики для документации проекта
6. Snippets для совместного использования фрагментов кода
7. Безопасность
8. Мониторинг и аналитика
9. Управление доступом
10. Интеграции с внешними инструментами
11. Артефакты и пакеты
12. Статичные сайты (Gitlab Pages)
13. Планирование спринтов
14. Контейнеризация Docker
15. Совместная работа



# Пример создания групп и подгрупп



The sidebar navigation menu includes the following items:

- Your work
- Projects
- Groups
- Issues
- Merge requests
- To-Do List
- Milestones
- Snippets
- Activity

Your work / Groups

## Groups

Explore groups

New group

	Search or filter results...	Q	Created date	↓		
▼	 Otus Group  Owner		2	0	1	⋮
	 Example Subgroup  Owner		0	0	1	⋮
	 Gitlab CI  Owner		0	0	1	⋮
	 Sample group  Owner		0	1	1	⋮



# Уровни доступа в GitLab

- **Instance administrator** — доступен только для отдельных инсталляций, может всё.
- **Owner** — владелец группы проектов, может всё кроме чисто технических штук типа включения-отключения фич и интеграции с другими сервисами.
- **Maintainer** — может всё кроме некоторых действий в отношении всего проекта типа изменения его названия или степени видимости, а также деструктивных действий типа удаления задач
- **Developer** — может то же что и Maintainer кроме некоторых администраторских и деструктивных функций внутри проекта типа настройки защищенности ветвей и редактирования комментариев.



# Уровни доступа в GitLab

- **Reporter** — может редактировать задачи, но не может вносить изменения в репозиторий.
- **Guest** — доступ только на чтение issue кроме конфиденциальных, может создавать новые задачи.





# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет

# Практика

# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет

# Тюнинг GitLab Runner

# Как работает Gitlab Runner

Gitlab Runner предназначен для выполнения заданий, запущенных в пайплайне Gitlab CI/CD. Задания описываются в файле [.gitlab-ci.yml](#)

## Как работает Gitlab Runner:

1. Runner запрашивает новые задания из экземпляра Gitlab Server
2. Runner компилирует и отправляет задачу в Executor
3. Executor клонирует исходные файлы, загружает (если нужно) требуемые заданию артефакты из Gitlab Registry и выполняет задачу
4. Executor возвращает результат работы задания обратно в Runner
5. Runner обновляет выходные данные и статус задания в Gitlab Server



# Типы Runners

## 1. Shared runner (Instance)

*Где найти: Admin Area > CI/CD > Runners*

## 2. Group runner

*Где найти: CI/CD > Runners*

## 3. Project runner (Specific)

*Где найти: Settings > CI/CD*



# Типы Gitlab Runner Executors

- SSH
- Shell
- Parallels
- VirtualBox
- Docker
- Docker Autoscaler
- Docker Machine (auto-scaling)
- Kubernetes
- Instance
- Custom

*Docker Executor -- рекомендуемый вариант для большинства случаев  
(или если не знаете, что выбрать)*



# Вопросы для проверки

**Понимаете ли вы, как работает Gitlab Runner?**

**А чем отличается Runner от Executor?**

*Напишите в чат одну цифру с ответом:*

0. - не уверен или не понимаю до конца
1. - в целом понятно, но остались неясные моменты
2. - да, все понятно
3. - вчера деплоил раннеры в кубер, элементарно



# 1/3 Создаем Project Runner

The screenshot shows the GitLab interface for configuring CI/CD settings. The left sidebar contains a navigation menu with items like Project, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings, General, Integrations, Webhooks, Access tokens, Repository, Merge requests, CI/CD, Packages and registries, and Monitor. The 'Runners' section is highlighted in the top navigation bar. The main content area is titled 'Runners' and includes a 'Collapse' button. It explains that runners are processes that pick up and execute CI/CD jobs for GitLab. It provides instructions on how to register runners and lists two states: 'active' (available to run jobs) and 'paused' (not available to run jobs). The page is divided into three main sections: 'Project runners', 'Instance runners', and 'Group runners'. The 'Project runners' section shows a 'New project runner' button. The 'Instance runners' section has a toggle for 'Enable instance runners for this project' which is currently turned on. The 'Group runners' section has a 'Disable group runners' button for this project. At the bottom, it states that the group does not have any group runners yet.

Sample group / demo / CI/CD Settings

## Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.

### How do runners pick up jobs?

Runners are either:

- active** - Available to run jobs.
- paused** - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make sure runners only handle the jobs they are equipped to run. [Learn more.](#)

### Project runners

These runners are assigned to this project.

[New project runner](#)

### Instance runners

These runners are available to all groups and projects.

**Enable instance runners for this project**

This GitLab instance does not provide any instance runners yet. Administrators can register instance runners in the admin area.

### Group runners

These runners are shared across projects in this group.

Group runners can be managed with the [Runner API](#).

[Disable group runners](#) for this project

This group does not have any group runners yet. To register them, go to the

### Assigned project runners

- #6 (9t92b-PWy) [Remove runner](#)  
docker-runner

# 2/3 Создаем Project Runner

Sample group / demo **CI/CD Settings / New runner**

## New project runner

Create a project runner to generate a command that registers the runner with all its configurations.

### Tags

Add tags to specify jobs that the runner can run. [Learn more.](#)

Separate multiple tags with a comma. For example, `macos, shared`.

Run untagged jobs  
Use the runner for jobs without tags in addition to tagged jobs.

### Configuration (optional)

#### Runner description

Paused  
Stop the runner from accepting new jobs.

Protected  
Use the runner on pipelines for protected branches only.

Lock to current projects   
Use the runner for the currently assigned projects only. Only administrators can change the assigned projects.

#### Maximum job timeout

Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.

Enter the job timeout in seconds. Must be a minimum of 600 seconds.

[Help](#) [Admin](#) [Create runner](#)



# 3/3 Создаем Project Runner

Sample group / demo / CI/CD Settings **Register runner**

Runner created. X

GitLab Runner must be installed before you can register a runner. [How do I install GitLab Runner?](#)

## Step 1

Copy and paste the following command into your command line to register the runner.

```
$ gitlab-runner register  
--url https://dpt55.gitlab.yandexcloud.net  
--token glrt-46F6
```

**i** The runner authentication token `glrt-46F6` displays here for a short time only. After you register the runner, this token is stored in the `config.toml` and cannot be accessed again from the UI.

## Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. [Not sure which one to select?](#)

## Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a [system](#) or [user service](#).

[View runners](#)

# Регистрируем Runner

1. Токен показывается **один раз**. Сохраните команду с токеном, если необходимо
2. Откройте консоль VM с Ubuntu, где вы ранее установили Runner, и отправьте предложенную команду регистрации с URL вашего инстанса Gitlab и токеном
3. На вопрос установщика о выборе Executor введите “`docker`”, затем укажите любой подходящий Docker образ (можно предлагаемый по умолчанию)  
*NB: Не забудьте предварительно установить Docker на VM.*
4. Проверьте, что Runner появился: `gitlab-runner list`  
(добавьте `sudo` перед командой, если Runner был установлен с админ правами)
5. Перезапустите демон, если Runner был установлен как служба:  
`sudo systemctl restart gitlab-runner`
6. Вернитесь в Gitlab и проверьте статус Runner

# Настройка Group runner

Group

- G Gittabb CICD
- Pinned
- Issues
- Merge requests
- Manage
- Plan
- Code
- Build
- Deploy
- Operate
- Settings**
  - General
  - Integrations
  - Access tokens
  - Projects
  - Repository
  - CI/CD**
  - Applications

Help Admin

Otus Group / Gittabb CICD / CI/CD Settings

Search page

## General pipelines

Customize your pipeline configuration. Expand

## Variables

Variables store information that you can use in job scripts. Each group can define a maximum of 30000 variables. [Learn more.](#) Expand

## **Runners**

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#) Collapse

**Enable instance runners for this group**  
Enable instance runners for all projects and subgroups in this group.

Allow projects and subgroups to override the group setting  
Allows projects or subgroups in this group to override the global setting.

## Auto DevOps

Automate [building](#), [testing](#), and [deploying](#) your applications based on your continuous integration and delivery configuration. [How do I get started?](#) Expand

# Настройка Shared Runner

Admin area / CI/CD

## Continuous Integration and Deployment Expand

Customize CI/CD settings, including Auto DevOps, instance runners, and job artifacts.

## Package Registry Expand

Configure package forwarding and package file size limits.

## Container Registry Expand

Various container registry settings.

## Runners Collapse

Configure runner version management and registration settings.

### Runner version management

- Fetch GitLab Runner release version data from GitLab.com  
Official runner version data is periodically fetched from GitLab.com to determine whether the runners need upgrades. [Learn more.](#)

### Runner registration

- Allow runner registration token  
When disabled, runner registration tokens are disabled from runner pages, and maintainers and owners cannot use registration tokens to register runners. They can use runner authentication tokens instead as the more secure runner registration method.
- Members of the project can create runners
- Members of the group can create runners

[Save changes](#)

Admin area / CI/CD

Monitoring

Messages

System hooks

Applications

Abuse reports

Deploy keys

Labels

Settings

General

Integrations

Repository

CI/CD

Reporting

Metrics and profiling

Network

Appearance

Preferences

Help

Admin

# Создаем простой пайплайн

1. Создайте файл `.gitlab-ci.yml` в корневой папке репозитория вашего проекта Gitlab
2. Добавьте в него следующее содержимое:

```
echo job:  
  script:  
  - echo "Hello world!"
```

1. Сохраните и отправьте коммит с изменениями



# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет

# Практика

# Вопросы?



Ставим "+",  
если вопросы есть



Ставим "-",  
если вопросов нет

# Подведём итоги занятия

# Теперь вы сможете:

Отправьте в чат номера достигнутых целей

1. Создать проект, группу и подгруппу в Gitlab

2. Ориентироваться в основной части интерфейса Gitlab

3. Выполнить базовую настройку Gitlab Runner

4. Запустить простой пайплайн CI/CD



# Домашнее задание

К этому уроку на платформе есть ДЗ, в котором вам нужно:

1. Создать и настроить проект
2. Привязать к нему GitLab Runner
3. Запустить простой пайплайн из одной задачи
4. Проверить успешность выполнения задачи в пайплайне



# Полезные ссылки

1. [Установка Gitlab Runner](#)
2. [Зарезервированные слова, которые нельзя использовать при именовании проектов и групп](#)
3. [Что не забыть сделать после установки Docker на Ubuntu](#) (для запуска Gitlab Runner с Docker Executor)



**Заполните, пожалуйста,  
опрос о занятии  
по ссылке в чате**



# ЭТОТ КОНТЕНТ КУПЛЕН НА САЙТЕ **SKLADCHIK.ORG**



**ТЫ ЕЩЕ  
НЕ В КЛУБЕ?**  
ПРИСОЕДИНЯЙСЯ  
И НАЧИНАЙ ЭКОНОМИТЬ!



## ЧТО ТАКОЕ КЛУБ «СКЛАДЧИК»?



### Платформа

Это платформа, где каждый день тысячи людей собираются вместе, чтобы общими усилиями находить, приобретать и изучать курсы интересные именно им.



### Сообщество

Это сообщество из 500 000 людей, открывших для себя выгодный способ быть в тренде самых актуальных и получать ценные знания по минимальной цене.



### Библиотека

Это крупнейшая библиотека инфопродуктов в которой можно найти практически любой курс или тренинг продававшийся за последние 10 лет, а также уникальные авторские инфопродукты, которые не получится найти больше нигде.

