Deckhouse
ФЛАНТ

Enterprise CI/CD без компромиссов

Подход Deckhouse Code

Deckhouse Code

Спикер

Никита Борзых

Руководитель продукта Deckhouse Code

О чём поговорим

01 **CI/CD раньше и сейчас:** как изменилась роль платформы

02 **Ситуация на рынке:** что доступно российским компаниям

03 **Deckhouse Code:** наш подход к enterprise CI/CD

04 **Путь кода в продакшен:** что надо контролировать на каждом этапе

05 **Планы развития:** куда движется Deckhouse Code

06 **Живое демо:** от коммита до продакшена

Deckhouse
ФЛАНТ

ФЛАНТ

Синергия опыта вендора, интегратора, сервисной и консалтинговой компании

Deckhouse

Deckhouse — продуктовое подразделение, разработчик продуктов для построения надёжной enterprise-инфраструктуры

DaaS

DaaS — комплексное DevOps-сопровождение инфраструктуры в режиме 24/7 силами выделенной DevOps-команды

Express 42

«Экспресс 42» — DevOps-консалтинг. От анализа узких мест в ИТ-процессах до создания роадмапа изменения ИТ для реализации цифровой трансформации

CI/CD раньше и сейчас

Deckhouse
ФЛАНТ

2010-ые.
Git (SVN) и скрипты деплоя

Каждая команда работает по-своему — единых правил нет

Кто-то делает код-ревью, кто-то нет.
У каждой команды свои договорённости
про мерж, ветки и релизы. Общей картины
ни у кого нет.

CI/CD — у кого как настроено

Одни команды используют пайплайны,
другие собирают скриптами. Единого стандарта нет,
каждый проект — отдельный мир.

Доступы — раздали и забыли

Доступ к репозиториям выдаётся по запросу.
Люди выписывают вечные токены, ревизии
доступа не проводятся
или проводятся от случая к случаю.

Платформа не воспринимается как критическая инфраструктура

Пока работает — в неё не инвестируют.
Ответственной команды может не быть.
Риски копятся незаметно, пока
не становятся инцидентом.

2026.
Платформа управления кодом — фундамент бизнеса

От неё зависят скорость поставки и Time to Market

Чем быстрее код проходит путь от идеи до продакшена, тем быстрее бизнес реагирует на рынок. Платформа — это конвейер, который должен работать непрерывно.

Это источник правды: доступы, секреты, контроль изменений

Кто имеет доступ к коду, кто может вносить изменения, где хранятся ключи и токены — всё это в одном месте. Потеря контроля здесь = потеря контроля над разработкой.

Сбой платформы = остановка всех команд и релизов

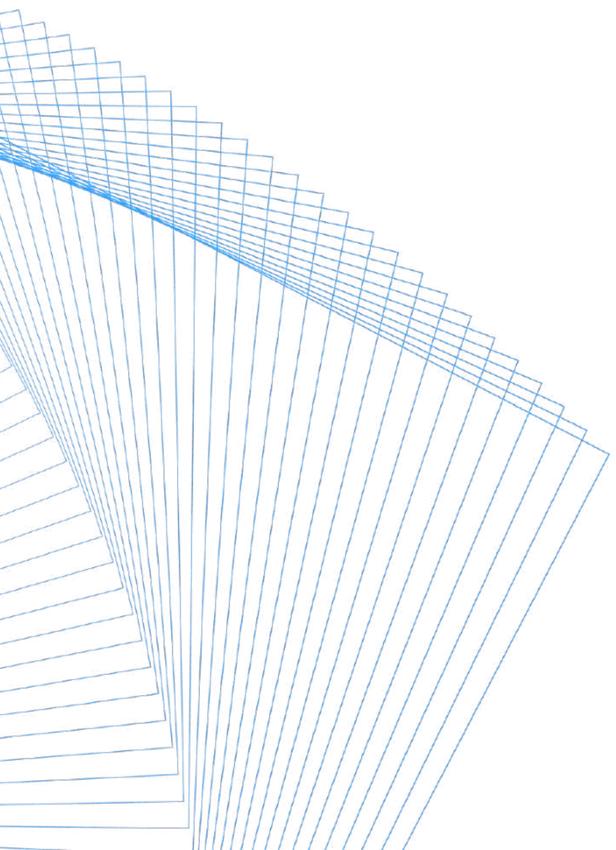
Не работает Git — разработчики не могут коммитить. Не работает CI/CD — ничего не собирается и не деплоится. Одна точка отказа останавливает всю компанию.

Надёжный фундамент ускоряет бизнес, ненадёжный — становится узким местом

Платформа либо помогает выпускать продукт быстрее конкурентов, либо тормозит каждый релиз.

Проблема российского рынка. Какие варианты есть сегодня?

После ухода Gitlab и Atlassian
с российского рынка у команд осталось три пути.
У каждого — свои ограничения.

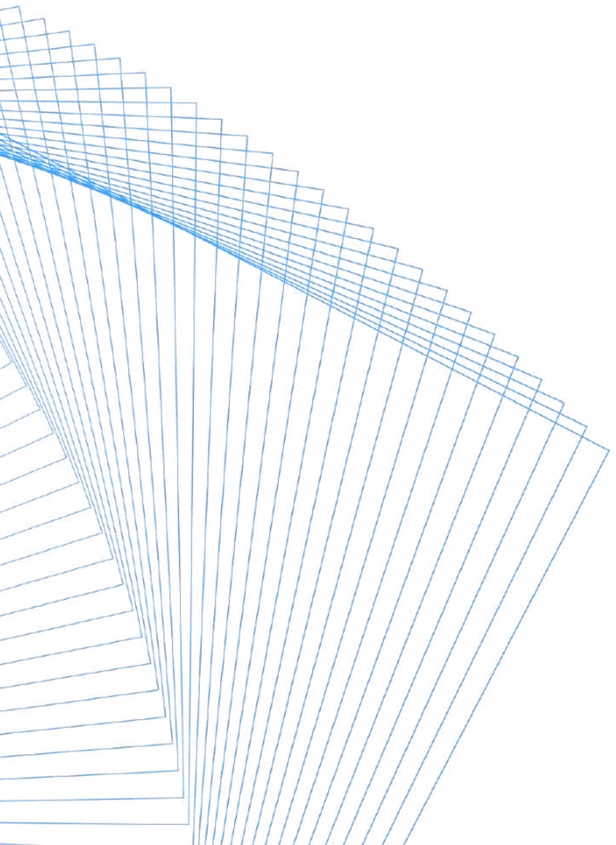


Использовать
Open Source-решения

Перейти на российские
платформы, написанные с нуля

Вести разработку
своими силами

Open-source решения



Бесплатные, знакомые, но с ограничениями

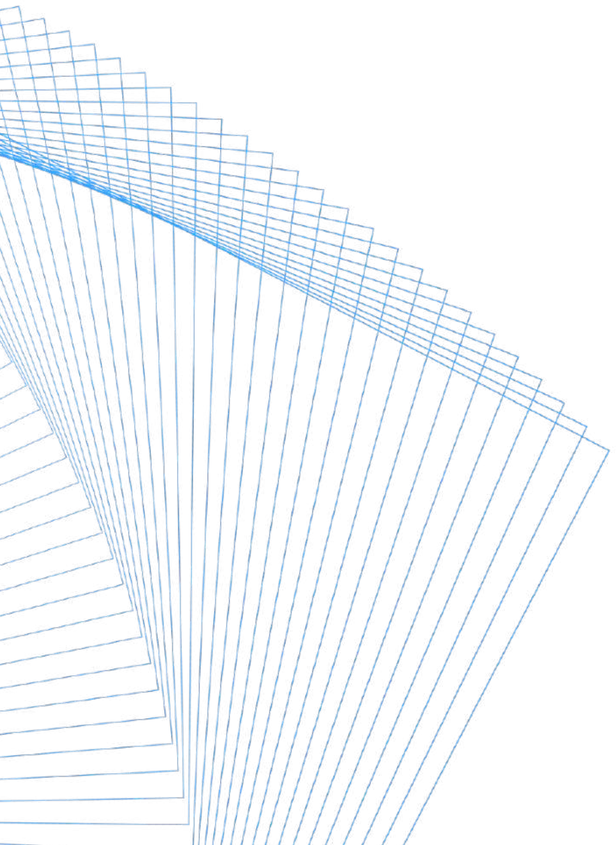
Open Source-платформы закрывают базовые сценарии, но в них нет enterprise-функций: управление доступами, безопасность, аудит, соответствие корпоративным политикам.

Нет вендорской поддержки

Патчи безопасности, совместимость, troubleshooting — всё на вашей команде.

Поддержки продукта нет.

Перейти на платформы, написанные с нуля



Нестабильны и бедны по функциональности

Молодые продукты ещё не прошли проверку нагрузкой и временем. Базовые сценарии работают, но до enterprise-уровня далеко.

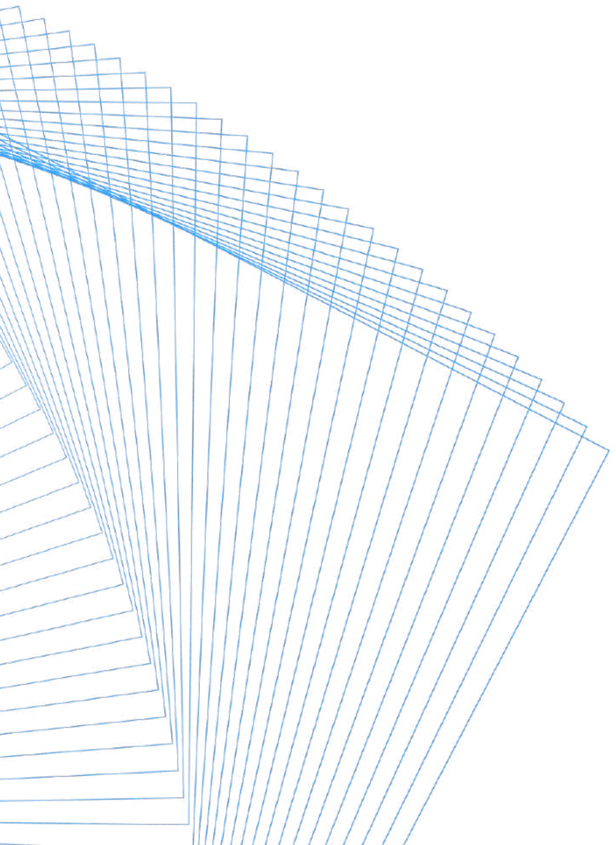
Переезд — отдельный проект

Миграция репозиториев, пайплайнов, интеграций, прав доступа — это месяцы работы и риски остановки сборки и потери данных.

Командам придётся переучиваться

Новый интерфейс, новая логика, новые утилиты. Накопленный опыт и наработки комьюнити становятся бесполезными.

Разработка своими силами



Дописать поверх Open Source или написать с нуля

В обоих случаях вы становитесь вендором для себя. Разработка, поддержка, безопасность, обновления, совместимость — всё на вашей команде.

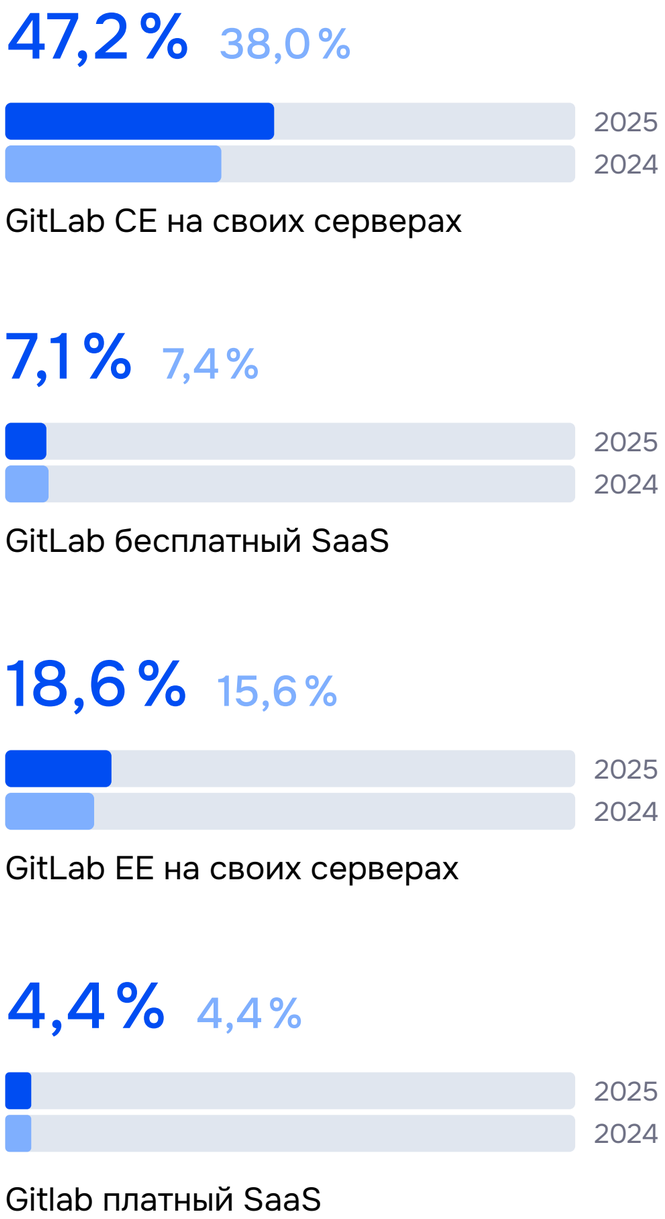
Ресурсы уходят на платформу, а не на продукт.

Повторение чужих ошибок

Проблемы, которые зрелые платформы уже решили за 15 лет, вам придётся решать заново — с нуля и за свой счёт.

Бóльшая часть рынка использует Gitlab

- Больше 70 % рынка использует GitLab для хранения кода и CI/CD
- Тренд продолжается — доля растёт год к году
- Команды знают инструмент, процессы настроены, CI/CD работает



По данным отчёта State of DevOps

Наш подход. Deckhouse Code

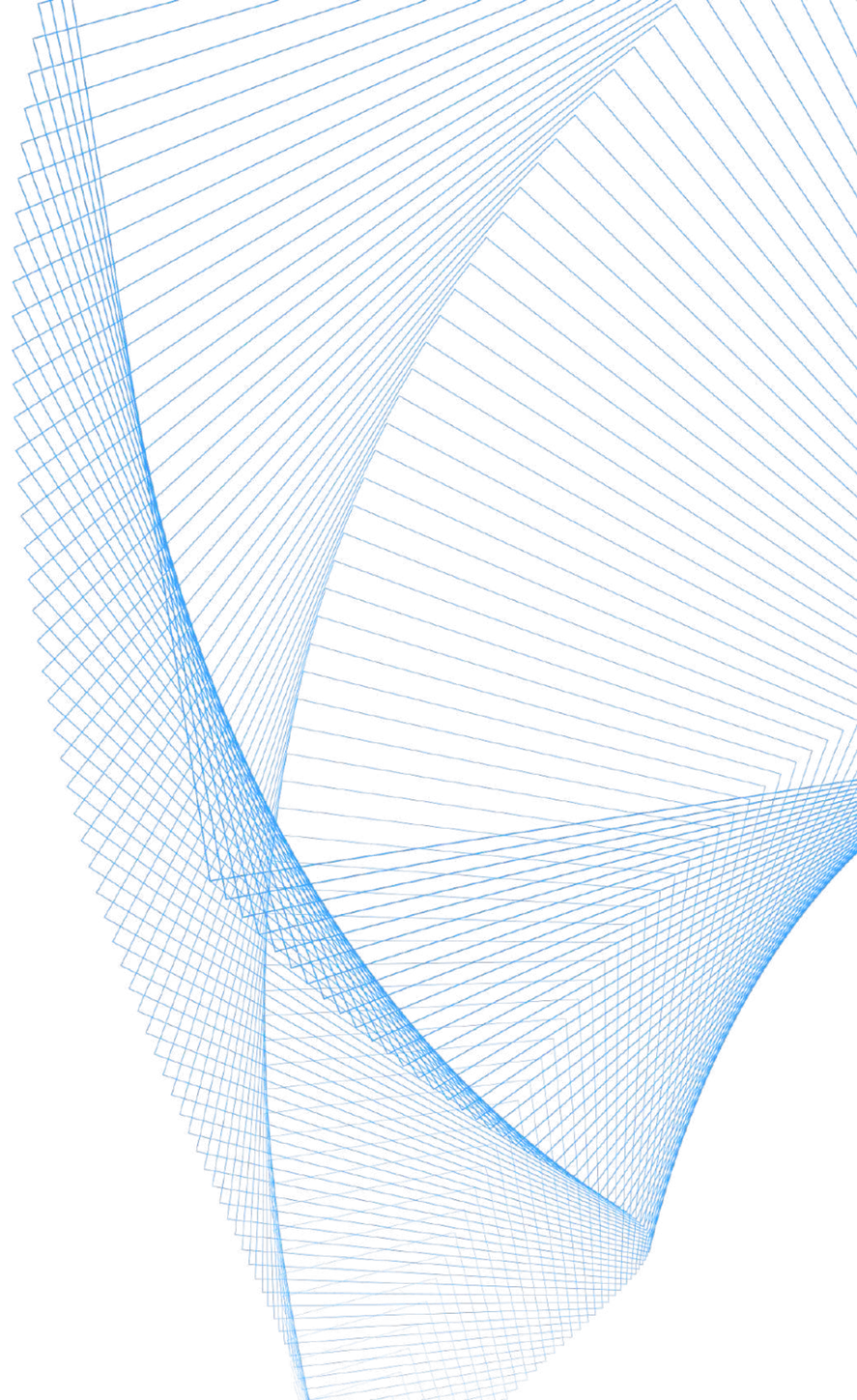
Deckhouse
ФЛАНТ

Deckhouse Code

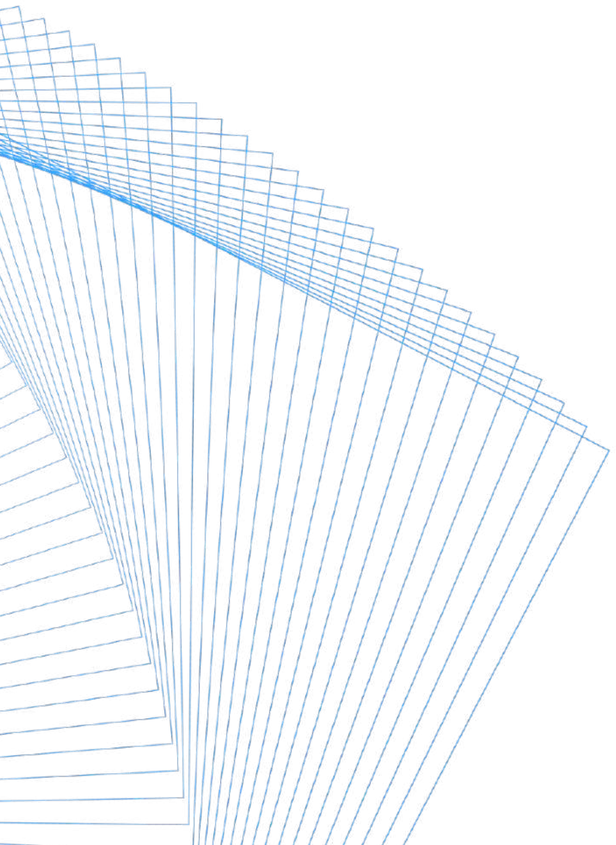
Самостоятельный продукт на базе GitLab CE

Мы взяли проверенную основу, на которой работают миллионы разработчиков, и построили поверх неё enterprise-платформу CI/CD.

GitLab CE — фундамент, Deckhouse Code — готовый продукт с поддержкой, автоматическими обновлениями, высокой доступностью и enterprise-функционалом.



Подход к продукту



01 Стабильная и обновляемая основа

Все обновления, патчи безопасности и исправления CVE из официального репозитория попадают к нам без задержек. Критические обновления безопасности доставляем с приоритетом.

02 Расширяем enterprise функционалом, а не заменяем

Наш код написан поверх базы, а не вместо неё.

Ядро и enterprise-функции развиваются независимо — обновления одного не затрагивают другое.

03 Собственная разработка

Enterprise-функции в Deckhouse Code написаны нами с нуля. Мы не используем код из GitLab EE — это полностью наша разработка.

Поддержка и экосистема

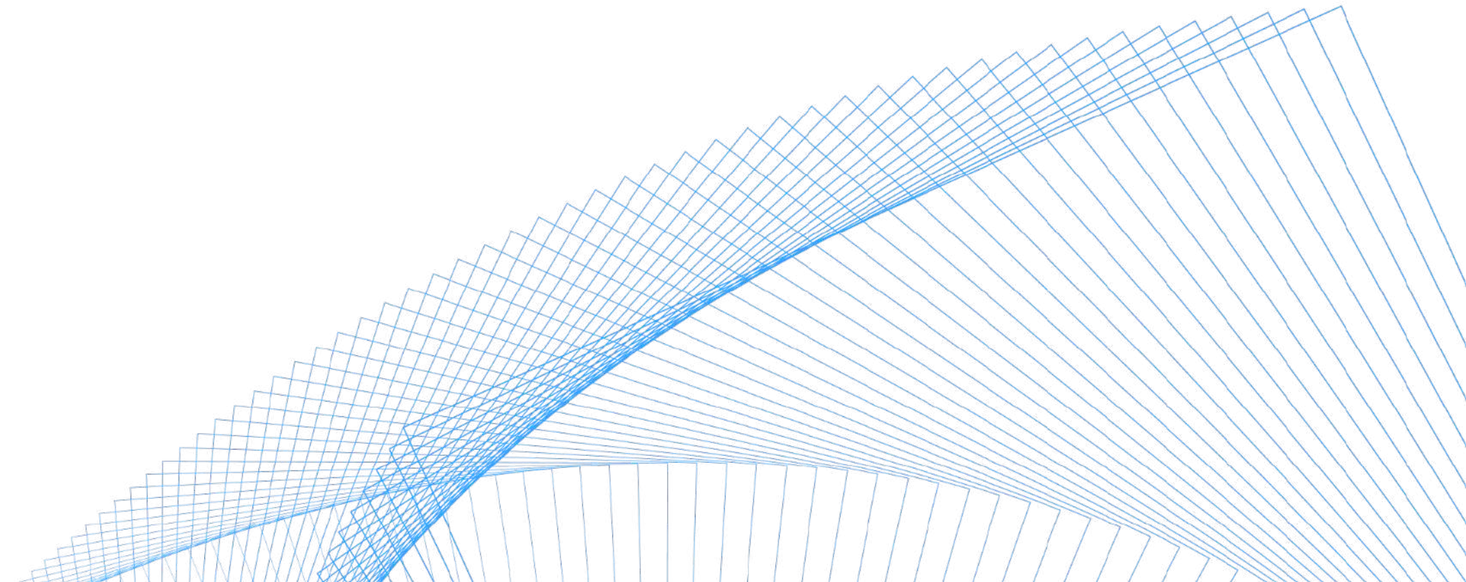
Вендорская поддержка

Open Source без поддержки — ваша ответственность.

В Deckhouse Code мы отвечаем за работу платформы, помогаем с настройкой и решаем проблемы.

Вся мощь комьюнити

Тысячи готовых шаблонов CI/CD, паттерны использования, интеграции с внешними системами — всё, что создано комьюнити за 15 лет, работает в Deckhouse Code без изменений. Не нужно переучиваться и искать решения с нуля.

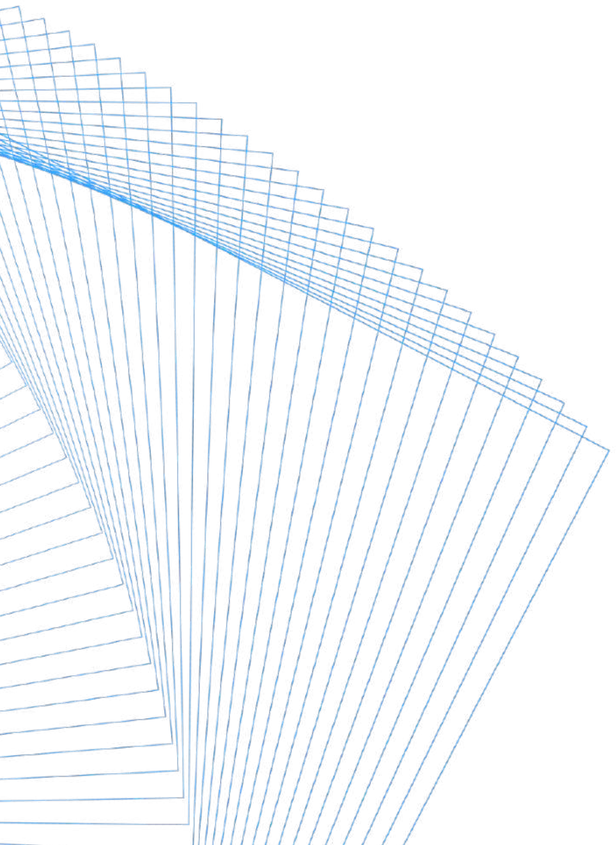


Enterprise CI/CD — какой он?

Требования к платформе
и как их закрывает Deckhouse Code

Deckhouse
ФЛАНТ

Что должно быть в enterprise CI/CD



01 Контроль изменений

- Ни одно изменение не попадает в продакшен без проверки
- Кто ревьювит, сколько подтверждений нужно, какие файлы требуют отдельного согласования — это должно быть контролируемо
- Контроль на правилах, а не на договорённостях

02 Автоматические проверки как обязательное условие

- Мерж невозможен без успешного пайплайна: сборка, тесты, проверка качества кода
- Пайплайн не прошёл — не мержим

03 Безопасность на каждом шаге

- Секреты не хранятся в коде и на платформе. Невалидные коммиты блокируются автоматически — до того, как попадут в общий репозиторий

04 Единый источник правды о доступах

- Пользователи и их права синхронизируются из корпоративного LDAP каталога
- Уволился сотрудник — доступ отозван автоматически
- Перешёл в другую команду — права доступа изменились

05 Прозрачность

- Знаем, кто имеет доступ, кому какие токены выданы, кто что менял
- Ответы на эти вопросы доступны в любой момент, а не по итогам расследования инцидента

Путь кода в продакшен

01
Push Rules

Push в ветку

Проверка автора, формата коммита и подписи коммита. Платформа отклоняет push, если правила нарушены

Коммиты провалидированы

02
CODEOWNERS + Approval Rules

Мерж-реквест

Правила апрува требуют подтверждений от нужных людей. CODEOWNERS автоматически назначает ревьюеров по изменённым файлам

Все апрувы получены

03
CI/CD

Пайплайн

Сборка, тесты, анализ качества.

Не прошёл конвейер — код не попадает в основную ветку

Пайплайн успешно пройден

04
Мерж

Мерж в main

Только после всех апрувов, успешного пайплайна и пройденного quality gates

Все проверки пройдены

05
Registry + Packages

Артефакты

Собранный образ попадает в container registry. Готов к деплою

Артефакт опубликован

06
Deploy

Деплой

Проверенный и собранный образ развёртывается в целевом окружении

Деплой прошёл успешно

На уровне всей платформы

Аудит событий

Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML

Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов

Все токены, SSH- и GPG-ключи — на виду в одном месте

Этап 1: push в ветку

Что попадает в репозиторий?

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита

Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров

Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault

Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна

Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою

Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении

Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

Push Rules
Проверка автора, формата коммита, подписи, запрет нежелательных файлов

Правила отправки изменений (Push Rules)

Что это:

правила, ограничивающие пуш коммитов в репозитории. Реализованы на уровне системы, группы и проектов с опциональной возможностью переопределения

Зачем:

дисциплина и безопасность разработки, а также предотвращение ошибок в коде

Сценарии использования:

Соблюдение внутренних стандартов разработки:

- обязательный префикс задачи в сообщении коммита (например, JIRA-123)
- унификация подхода к ветвлению (ограничение допустимых имён веток)

Гарантия корпоративной ответственности:

- запрет на коммиты без email в корпоративном домене. Исключает «анонимный» код и повышает прозрачность вклада каждого разработчика
- подпись коммитов GPG-ключом. Гарантирует, что конкретный коммит сделан конкретным автором

Повышение качества сборок:

- запрет пуша коммитов с определёнными типами файлов (например, .log, временные артефакты IDE)

Этап 2: мерж-реквест

Кто должен подтвердить изменение?

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита
Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров
Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault
Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна
Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою
Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении
Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

Approval Rules
Обязательные апрувы от нужных людей и команд

CODEOWNERS
Автоматическое назначение ревьюеров по изменённым файлам

Правила ревью запросов на слияние

Что это:

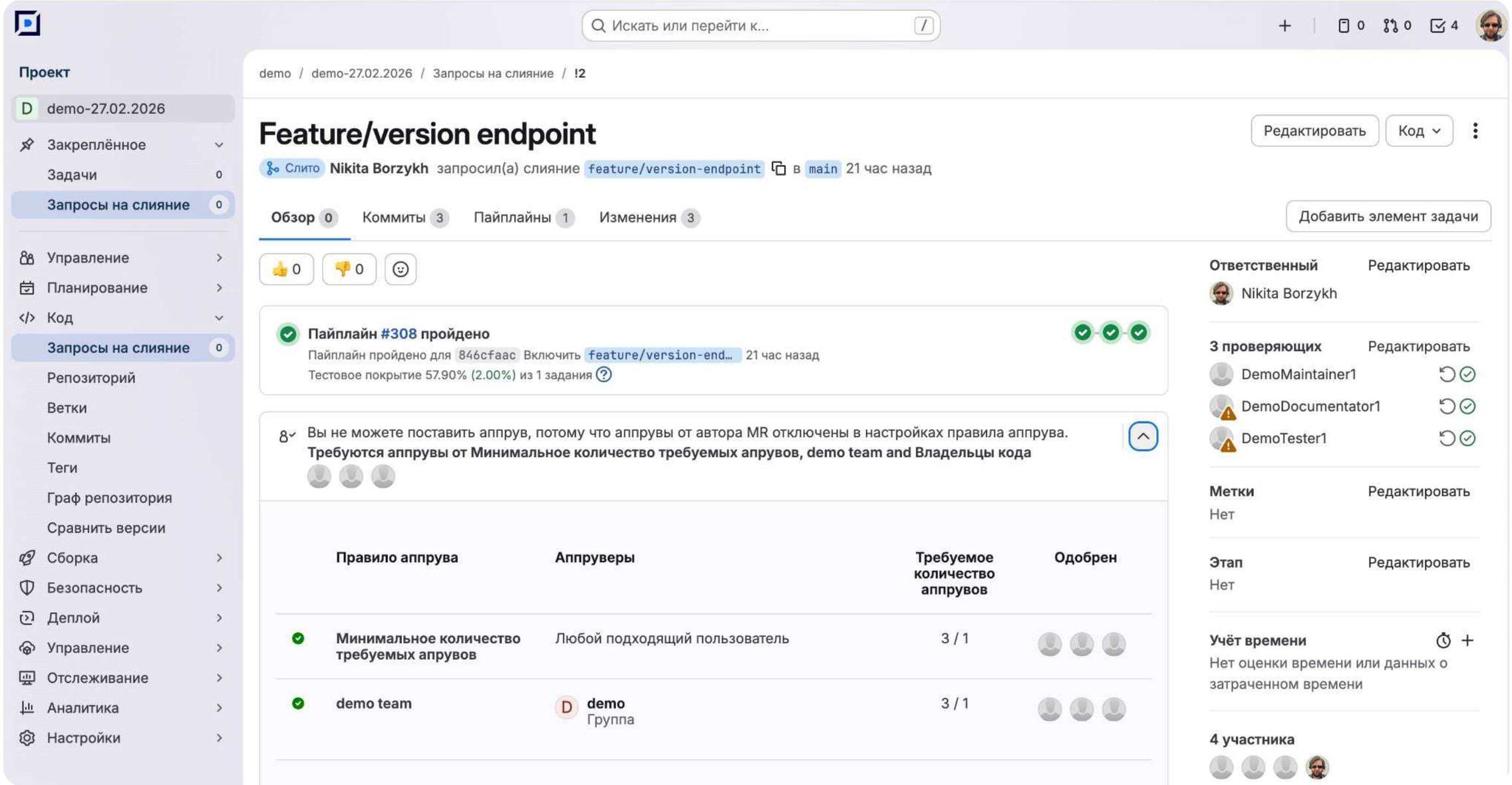
механизм настройки правил обязательного ревью перед слиянием кода

Зачем:

гарантирует соблюдение процессов качества и безопасности

Сценарии использования:

- Для критичных сервисов изменения не попадают в main, пока их не согласуют два ведущих разработчика и кто-то из команды тестирования или непосредственно тимлид
- В security-чувствительных проектах обязательным ревьюером назначается эксперт по ИБ
- Есть обязательные верхнеуровневые политики ревью на уровне группы, которые должны быть использованы в каждой команде


Искать или перейти к... /
0
0
4
Проект
D demo-27.02.2026
Закреплённое
Задачи 0
Запросы на слияние 0
Управление
Планирование
Код
Запросы на слияние 0
Репозиторий
Ветки
Коммиты
Теги
Граф репозитория
Сравнить версии
Сборка
Безопасность
Деплой
Управление
Отслеживание
Аналитика
Настройки
demo / demo-27.02.2026 / Запросы на слияние / !2
Feature/version endpoint
Редактировать
Код
Слито Nikita Borzykh запросил(а) слияние feature/version-endpoint в main 21 час назад
Обзор 0
Коммиты 3
Пайплайны 1
Изменения 3
Добавить элемент задачи
0
0
Пайплайн #308 пройдено
Пайплайн пройдено для 846cfaac Включить feature/version-end… 21 час назад
Тестовое покрытие 57.90% (2.00%) из 1 задания
Вы не можете поставить аппрув, потому что аппрувы от автора MR отключены в настройках правила аппрува.
Требуются аппрувы от Минимальное количество требуемых апрувов, demo team and Владельцы кода
Правило аппрува
Аппруверы
Требуемое количество аппрувов
Одобрен
Минимальное количество требуемых апрувов
Любой подходящий пользователь
3 / 1
demo team
D demo
Группа
3 / 1
Ответственный
Редактировать
Nikita Borzykh
3 проверяющих
Редактировать
DemoMaintainer1
DemoDocumentator1
DemoTester1
Метки
Редактировать
Нет
Этап
Редактировать
Нет
Учёт времени
Нет оценки времени или данных о затраченном времени
4 участника

Функциональность владения кодом (CODEOWNERS)

Что это:

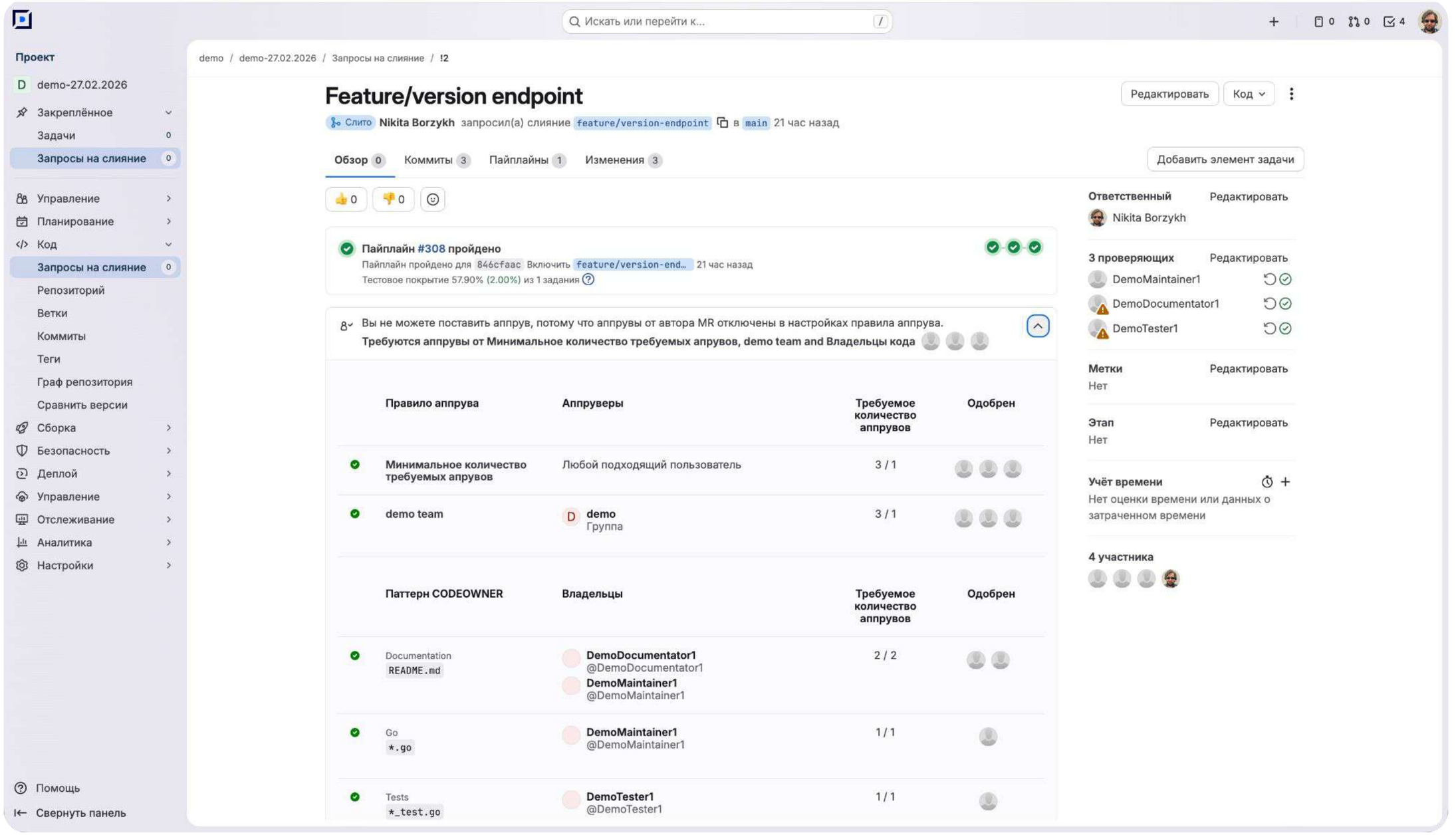
автоматическое назначение ревьюеров
в зависимости от изменяемых файлов
или директорий

Зачем:

ускоряет код-ревью и гарантирует,
что за каждый модуль отвечают эксперты

Сценарии использования:

- При изменении файлов директории **docs/** или любых Markdown-файлов автоматически назначается на команду «Документации»
- Владельцы кода контролируют качество и архитектуру именно своих компонентов
- В монорепозиториях: каждый сервис имеет своего владельца, MR не мержится без его апрува


Искать или перейти к...
Проект
demo-27.02.2026
Закреплённое
Задачи
Запросы на слияние
Управление
Планирование
Код
Запросы на слияние
Репозиторий
Ветки
Коммиты
Теги
Граф репозитория
Сравнить версии
Сборка
Безопасность
Деплой
Управление
Отслеживание
Аналитика
Настройки
Помощь
Свернуть панель
demo / demo-27.02.2026 / Запросы на слияние / !2
Feature/version endpoint
Редактировать
Код
Слито Nikita Borzykh запросил(а) слияние feature/version-endpoint в main 21 час назад
Обзор 0
Коммиты 3
Пайплайны 1
Изменения 3
Добавить элемент задачи
Пайплайн #308 пройдено
Пайплайн пройдено для 846cfaac Включить feature/version-end… 21 час назад
Тестовое покрытие 57.90% (2.00%) из 1 задания
Вы не можете поставить аппрув, потому что аппрувы от автора MR отключены в настройках правила аппрува.
Требуются аппрувы от Минимальное количество требуемых апрувов, demo team and Владельцы кода
Правило аппрува
Аппруверы
Требуемое количество аппрувов
Одобрен
Минимальное количество требуемых апрувов
Любой подходящий пользователь
3 / 1
demo team
demo
Группа
3 / 1
Паттерн CODEOWNER
Владельцы
Требуемое количество аппрувов
Одобрен
Documentation
README.md
DemoDocumentator1
@DemoDocumentator1
DemoMaintainer1
@DemoMaintainer1
2 / 2
Go
*.go
DemoMaintainer1
@DemoMaintainer1
1 / 1
Tests
*_test.go
DemoTester1
@DemoTester1
1 / 1
Ответственный
Редактировать
Nikita Borzykh
3 проверяющих
Редактировать
DemoMaintainer1
DemoDocumentator1
DemoTester1
Метки
Редактировать
Нет
Этап
Редактировать
Нет
Учёт времени
Нет оценки времени или данных о затраченном времени
4 участника

Этапы 3–4: пайплайн и мерж

Что проверяется перед попаданием в main?

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита
Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров
Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault
Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна
Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою
Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении
Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

CI/CD
Сборка, тесты, анализ качества, quality gates

Vault (Deckhouse Stronghold)
Секреты динамически подгружаются в пайплайн, не хранятся в коде

Блокировка мержа
Мерж в основную ветку невозможен без успешного пайплайна — код не попадает в main, пока сборка и тесты не пройдены

Интеграция с Vault

Что это:

поддержка подключения к любым Vault-совместимым системам управления секретами, включая Deckhouse Stronghold

Зачем:

секреты динамически подгружаются в пайплайны для безопасного хранения и централизованного управления ключами, паролями и токенами. Исключает хранение секретов в коде и повышает соответствие требованиям безопасности и аудита

Сценарии использования:

- **Безопасный деплой в продакшен** — пайплайн автоматически получает временные токены доступа и удаляет их по завершении
- **Единый контроль для всех команд** — секреты управляются централизованно, что исключает дублирование и «забытые» ключи
- **Независимость от конкретного вендора** — можно использовать Stronghold или любое другое Vault-совместимое решение, сохраняя независимость от конкретного вендора

Deckhouse Code

```yaml
deps:
  stage: deps
  variables:
    GOPROXY_BASE: "https://${NEXUS_USER}:${NEXUS_PASSWORD}@nb-stg1.ru-central1.internal/repository/golang-proxy/"
    GOSUMDB_PROXY_BASE: "https://${NEXUS_USER}:${NEXUS_PASSWORD}@nb-stg1.ru-central1.internal/repository/golang-sum-proxy/"
  id_tokens:
    VAULT_ID_TOKEN:
      aud: vault
  secrets:
    NEXUS_USER:
      vault: demo/NEXUS_USER@dcd-kv
      token: $VAULT_ID_TOKEN
      file: false
    NEXUS_PASSWORD:
      vault: demo/NEXUS_PASSWORD@dcd-kv
      token: $VAULT_ID_TOKEN
      file: false
  script:
    - go mod download
```

Этап 5: артефакты

Где хранятся результаты сборки?

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита
Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров
Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault
Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна
Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою
Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении
Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

Container Registry
Встроенный реестр образов, политики очистки

Package Registry
Хранилище пакетов и артефактов сборок (npm, PyPI, Maven, Helm, generic)

Container Registry

Что это:

встроенный реестр контейнерных образов — каждый проект получает собственный registry, доступный из пайплайнов без дополнительной настройки

Зачем:

образы остаются внутри контура, не нужны внешние registry. Пайплайн собирает, пушит и деплоит — всё в одной платформе

Сценарии использования:

- Пайплайн собирает Docker-образ и пушит в registry проекта — готов к деплою без внешних зависимостей
- Политики очистки (cleanup policies) автоматически удаляют устаревшие образы и теги без остановки registry
- Образы доступны между проектами внутри группы — удобно для общих базовых образов

Хранилище артефактов и пакетов

Что это:

встроенное хранилище для артефактов сборок и пакетов: бинарники, архивы, Helm-чарты, npm/PyPI/Maven/NuGet-пакеты, generic-файлы — всё доступно из пайплайнов без внешних менеджеров пакетов

Зачем:

не нужно поднимать отдельный Nexus или Artifactory. Артефакты и пакеты хранятся рядом с кодом, версионируются и доступны другим проектам и командам

Сценарии использования:

- Пайплайн публикует собранные .deb/.rpm-пакеты или Go-бинарники — другие команды забирают их как зависимости
- Maven/npm/PyPI-пакеты общих библиотек хранятся в package registry группы — команды подключают их стандартными менеджерами пакетов
- Helm-чарты публикуются в package registry проекта — деплой подключает его как стандартный Helm-репозиторий, без внешних chart-репозиториев

Этап 6: деплой

Проверенный образ из встроенного registry — в целевом окружении

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита
Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров
Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault
Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна
Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою
Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении
Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

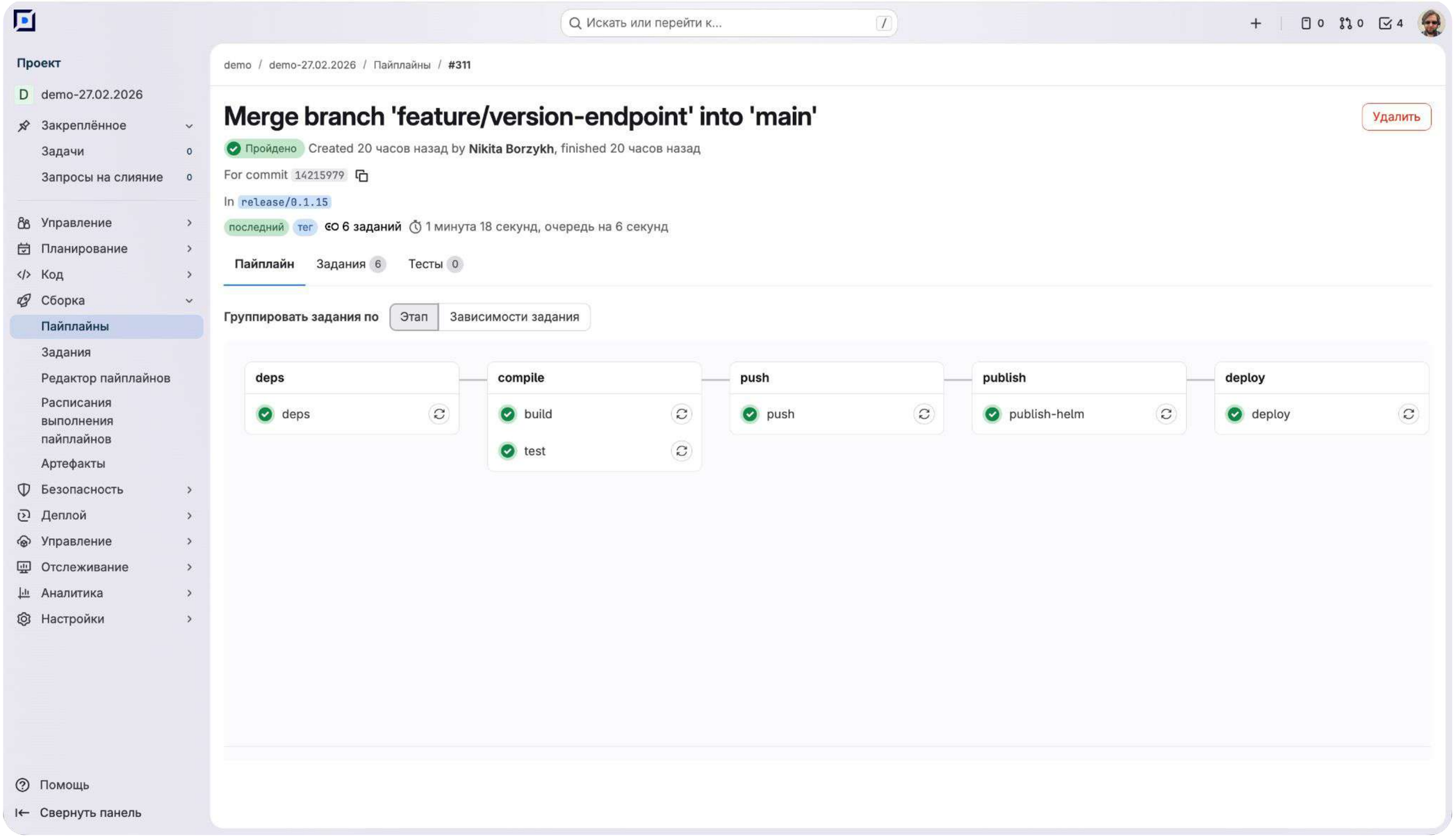
LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

Container Registry
Проверенный образ деплоится из встроенного registry — артефакты не покидают контур

Helm-чарты из Package Registry
Деплой подключает Helm-репозиторий проекта напрямую, без внешних chart-хранилищ


Искать или перейти к... /
0
0
4
Проект
D demo-27.02.2026
Закреплённое
Задачи 0
Запросы на слияние 0
Управление
Планирование
Код
Сборка
Пайплайны
Задания
Редактор пайплайнов
Расписания выполнения пайплайнов
Артефакты
Безопасность
Деплой
Управление
Отслеживание
Аналитика
Настройки
Помощь
Свернуть панель
demo / demo-27.02.2026 / Пайплайны / #311
Merge branch 'feature/version-endpoint' into 'main'
Удалить
Пройдено Created 20 часов назад by Nikita Borzykh, finished 20 часов назад
For commit 14215979
In release/0.1.15
последний тег 6 заданий 1 минута 18 секунд, очередь на 6 секунд
Пайплайн Задания 6 Тесты 0
Группировать задания по Этап Зависимости задания
deps
deps
compile
build
test
push
push
publish
publish-helm
deploy
deploy

На уровне всей платформы

Аудит, управление доступами и контроль токенов — сквозные функции на всех этапах

01
Push Rules
Push в ветку
Проверка автора, формата и подписи коммита
Коммиты провалидированы

02
CODEOWNERS + Approval Rules
Мерж-реквест
Обязательные апрувы и автоназначение ревьюеров
Все апрувы получены

03
CI/CD
Пайплайн
Сборка, тесты, quality gates. Секреты из Vault
Пайплайн успешно пройден

04
Мерж
Мерж в main
Только после всех апрувов и успешного пайплайна
Все проверки пройдены

05
Registry + Packages
Артефакты
Образы в container registry, пакеты в package registry. Готовы к деплою
Артефакт опубликован

06
Deploy
Деплой
Проверенный образ в целевом окружении
Деплой прошёл успешно

На уровне всей платформы

Аудит событий
Каждое действие зафиксировано: кто, когда, что изменил

LDAP/SAML
Доступы синхронизируются из корпоративного каталога автоматически

Дашборд доступов
Все токены, SSH- и GPG-ключи — на виду в одном месте

Функции Deckhouse Code на этом этапе

Аудит событий
Журнал всех действий, экспорт в SIEM

LDAP/SAML
Автоматическая синхронизация пользователей и групп

Дашборд токенов и ключей
Все способы доступа на виду

Роль Auditor
Полный read-only-доступ для ИБ и аудиторов

Аудит событий

Что это:

журнал всех значимых с точки зрения безопасности действий пользователей и администраторов

Зачем:

повышает прозрачность и позволяет быстро расследовать инциденты

Сценарии использования:

- Легко отследить, кто изменил права доступа к репозиторию
- Во время аудита или проверки можно подтвердить корректность процессов разработки
- Легко проверять, как, кем и когда были созданы группы, пользователи, проекты, ключи доступа
- Можно экспортировать в формате CEF в SIEM-систему для дальнейшего анализа

Аудит событий

Экспортировать в CSV

Все | Инстанс | Группа | Проект

Фильтровать по автору, типу эвента и цели

2025-11-01 — 2025-11-30

Автор	Событие	Обьект	Цель	Время события ↓
Administrator	Invited toolbox to the project app as maintainer	app	Toolbox	13 Ноябрь 2025 18:06
Administrator	Root granted access to new member reported_user_keena_franecki as a developer in the project 'app'.	app	reported_user_kee...	13 Ноябрь 2025 18:06
Administrator	Protected branch created	app	master	13 Ноябрь 2025 18:05
Administrator	Project created	app	app	13 Ноябрь 2025 18:03
Administrator	Group created	retail	retail	13 Ноябрь 2025 18:03
Administrator	User logged in	root	root	13 Ноябрь 2025 18:02
An unauthenticated user	User redirected to login page	Инстанс	(deleted)	13 Ноябрь 2025 18:02
Administrator	Password updated	root	root	13 Ноябрь 2025 18:02

Синхронизация групп и пользователей из LDAP

Что это:

автоматическая синхронизация пользователей и групп из LDAP/AD

Зачем:

снижает нагрузку на администраторов и повышает безопасность, позволяя сохранять единую систему управления пользовательским доступом

Сценарии использования:

- При увольнении сотрудника доступы к репозиториям автоматически закрываются
- Новые сотрудники сразу получают права в проектах согласно корпоративной группе
- При создании в LDAP группы, соответствующей заданным правилам, аналогичная создаётся и в Deckhouse Code

Единый реестр ключей и токенов доступа

Что это:

централизованный список всех способов доступа к Deckhouse Code: персональные токены, SSH- и GPG-ключи, групповые и проектные токены с указанием владельца, прав, области действия и срока жизни

Зачем:

прозрачность и контроль доступа, быстрый поиск и отзыв критичных токенов, снижение риска утечек и упрощение аудитов и расследования инцидентов

Сценарии использования:

- **Регулярный контроль прав** — администратор фильтрует список по уровню доступа и сроку действия, находит «вечные» и избыточно привилегированные токены и инициирует их отзыв
- **Наведение порядка** — поиск неиспользуемых токенов (давно не применялись) и устаревших ключей, их удаление и формирование прозрачной картины того, кто и через какие учётные данные имеет доступ к системе

Область администратора / **Ключи доступа**

Ключи доступа

Экспортировать в CSV

Персональные токены доступа | Ключи SSH | Ключи GPG | Токены доступа группы/проекта | Токены доступа ботов

Поиск или фильтрация токенов доступа... | Дата создания

Скрыть отозванные

Имя	Владелец	Область	Создан	Последний доступ	Истекает	Действия
test	**Administrator** gitlab_admin_738797...	read_user, api, write_repository, read_repository	28 Янв. 2026	28 Янв. 2026	⚠ 27 Фев. 2026	Отозвать Удалить
seeded-api-token	**Administrator** gitlab_admin_738797...	api, read_api, read_user, create_runner, manage_runner, k8s_proxy, self_rotate, mcp, granular, read_repository, write_repository, read_observability, write_observability, ai_features, sudo, admin_mode, read_service_ping	28 Янв. 2026	Никогда	28 Янв. 2027	Отозвать Удалить

Роль пользователя Auditor

Что это:

специальный тип пользователя с правами только на чтение всех проектов, групп и настроек инстанса — без возможности вносить изменения

Зачем:

даёт полную видимость для аудиторов и ИБ-специалистов без риска случайного или намеренного изменения данных

Сценарии использования:

- Служба ИБ получает доступ ко всем репозиториям и настройкам для проверки без риска что-то сломать
- Внешний аудитор видит процессы, права, историю изменений, но не может редактировать код или конфигурацию
- Руководитель разработки просматривает проекты всех команд для контроля стандартов, не вмешиваясь в процесс

Доступ

Тип пользователя

Определяет доступ пользователей к группам, проектам, ресурсам и области администратора.

- ○ Обычный
  Доступ к собственным группам и проектам.
- ◉ Аудитор
  Доступ только для чтения ко всем группам и проектам.
- ○ Администратор
  Полный доступ ко всем группам, проектам, пользователям, функциям и области администратора.

Сводка доступа для пользователя типа Аудитор

Область администратора
- Нет доступа.

Группы и проекты
- Доступ только для чтения ко всем группам и проектам.
- Также может быть добавлен в конкретные группы и проекты для получения дополнительных прав.

Настройки групп и проектов
- Требуется как минимум роль Мейнтейнера в указанных группах и проектах.

Что ещё есть?

Deckhouse
ФЛАНТ

Групповые и проектные хуки и Wiki

Что это:

Совместная и попроектная Wiki для команды
и система групповых вебхуков

Зачем:

Хранение знаний рядом с кодом
и автоматическая интеграция
с внешними системами

Сценарии использования:

- Wiki-пространство для команды
с документацией и схемами
- На базе групповой Wiki описан
team-API команды
- Вебхуки отправляют уведомления
в мессенджер команды при релизе

Зеркалирование репозиториев

Что это:

автоматическая синхронизация
кода с внешними репозиториями
по расписанию

Зачем:

упрощает миграции и работу
в гибридных сценариях

Сценарии использования:

- Репозитории синхронизируются с GitHub
при работе с внешними контрибьюторами
- Постепенный переезд на новую СКВ
без остановки разработки
- Получение последних обновлений из апстрима
- Отправка обновлений в апстрим

Что также доступно из GitLab CE

Deckhouse
ФЛАНТ

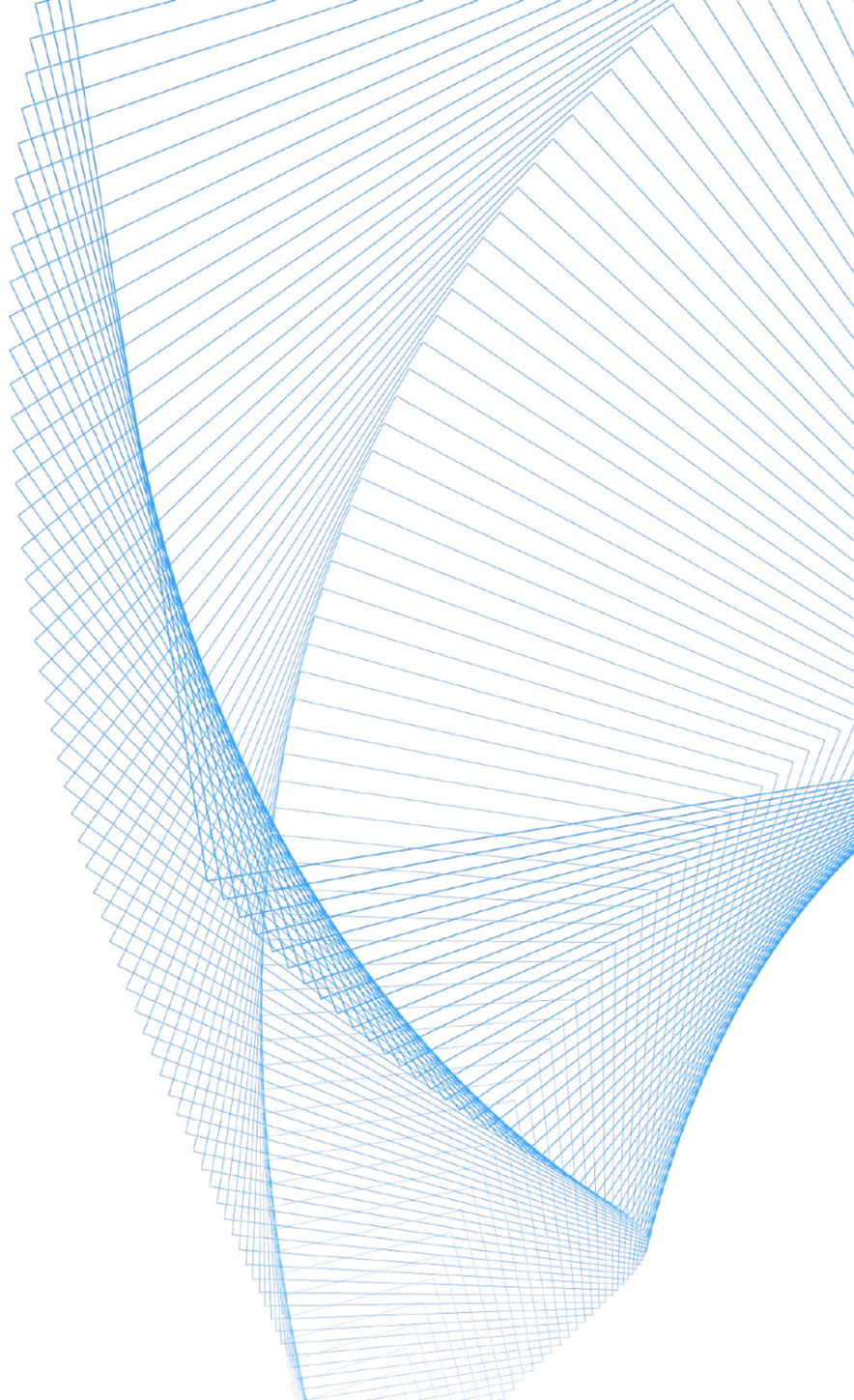
Deckhouse Code

Deckhouse Code сохраняет всю функциональность GitLab Community Edition

Например:

- Базовый Issue tracker — задачи, доски, майлстоуны работают «из коробки»
- Ваши существующие GitLab Runner'ы совместимы. Запускайте внутри DKP или на своей инфраструктуре
- CI/CD-скрипты и шаблоны — всё, написанное комьюнити GitLab, работает без изменений
- GitLab API — интеграции с IDE, CLI-утилитами и внешними системами сохраняются

Переезд на Deckhouse Code не ломает то, что уже работает



Установка, обновления и эксплуатация

Deckhouse
ФЛАНТ

Управляемая платформа

Что это:

Deckhouse Code развёртывается как модуль Deckhouse Kubernetes Platform. Оператор управляет полным жизненным циклом

Зачем:

инфраструктурная команда не тратит ресурсы на обслуживание — за работу платформы отвечает вендор

Три ключевых принципа:

- Установка за один CR — развёртывание через Custom Resource, оператор берёт на себя всю конфигурацию
- Автообновления без простоев — обновления, патчи безопасности и enterprise-функции доставляются автоматически. Критические CVE — с приоритетом
- HA и резервное копирование — отказоустойчивый режим, бэкапы по расписанию и перед обновлениями, восстановление после сбоев

Планы

Deckhouse
ФЛАНТ

Дорожная карта

01

Расширенный поиск

Полнотекстовый поиск
по артефактам, данным
и кодовой базе для ускорения
работы команд

02

Автоматизация CI/CD-инфраструктуры

Поставка собственного оператора
для Code Runner, упрощающая
управление сборочными
агентами в Kubernetes

03

Центр безопасности

Единая панель рисков
по проектам: уязвимости, тренды и
приоритеты в одном месте.
Быстрый обзор проблем
безопасности и их приоритизация
по уровню критичности

04

Гибкая модель доступа

Расширенные возможности
настройки ролей и прав
для разных сценариев
использования

05

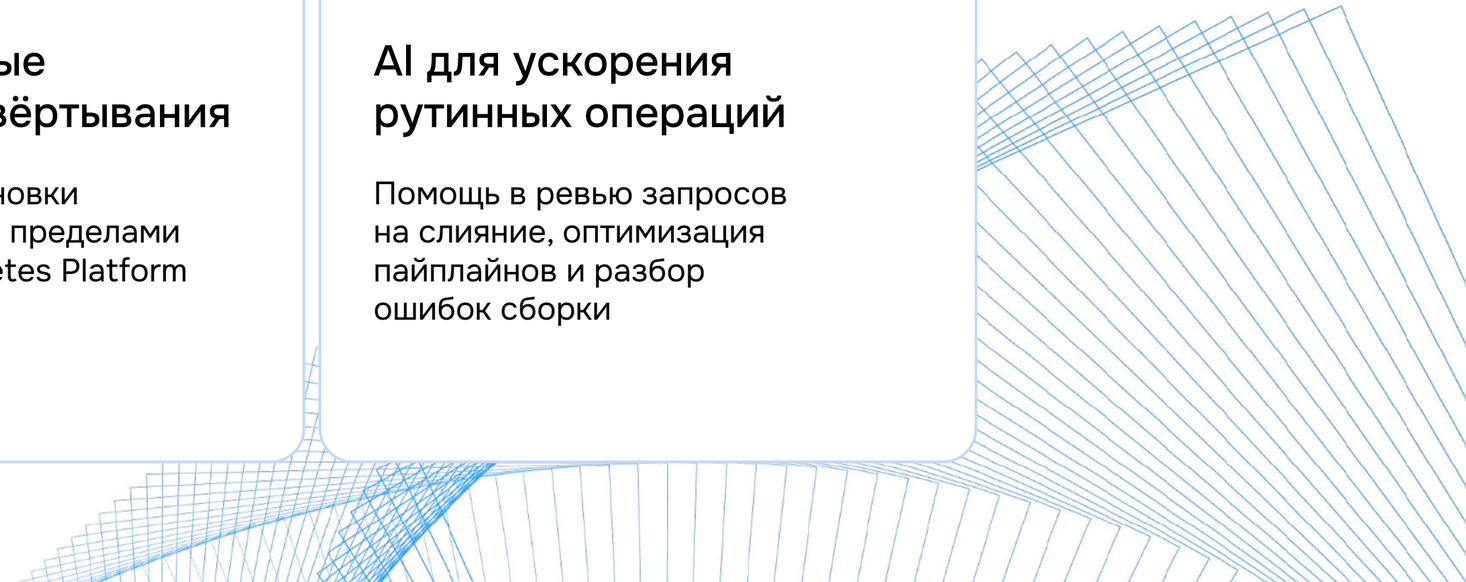
Альтернативные варианты развёртывания

Возможность установки
Deckhouse Code за пределами
Deckhouse Kubernetes Platform

06

AI для ускорения рутинных операций

Помощь в ревью запросов
на слияние, оптимизация
пайплайнов и разбор
ошибок сборки



Live demo

Анонсы мероприятий

12 марта 12:00 | Онлайн

Вебинар «Подтверждённая безопасность секретов: Deckhouse Stronghold получил сертификат ФСТЭК России»

Расскажем, как выстроить управление секретами в инфраструктуре, подпадающей под требования к использованию ПО на значимых объектах КИИ (ЗО КИИ), и избежать рисков несоответствия регуляторным требованиям

Зарегистрироваться на вебинар

12 марта 18:30 - 22:00 | Лофт «Авиатор»
г. Москва, Столешников пер., д. 6 стр. 3

Deckhouse Практикум, посвящённый продукту Deckhouse Code и лучшим DevOps-практикам.

Разберём ключевые возможности продукта и погрузимся в детали настройки процессов ревью MergeRequest’ов, а также расскажем, как интегрировать решение для безопасного хранения и управления секретами Deckhouse Stronghold в ваши CI/CD-процессы для максимальной безопасности чувствительных данных

Зарегистрироваться на Практикум

Deckhouse Code

В вашей компании есть запрос на комплексное, безопасное и удобное управление CI/CD для enterprise?

Приходите на личную консультацию с нашим экспертом. Подготовим демо под ваши задачи и обсудим все вопросы.

Deckhouse Code: ключевая информация (pdf)

Оставить заявку на консультацию

+7 (495) 721-10-27

deckhouse.ru