Тестирование ПО

Системы контроля версий

Кулаков Кирилл Александрович

Проблема обмена данными

- Сложный проект -> большая команда -> много кусочков кода
- Использование традиционных способов обмена данными:
  - Сетевые папки
  - Дискеты / Компакт диски / Флешки
  - Использование соц. сетей для передачи файлов
  - Облачные папки (Яндекс.диск)
  - Архивы на файлообменниках
  - Электронная почта

Системы контроля версий

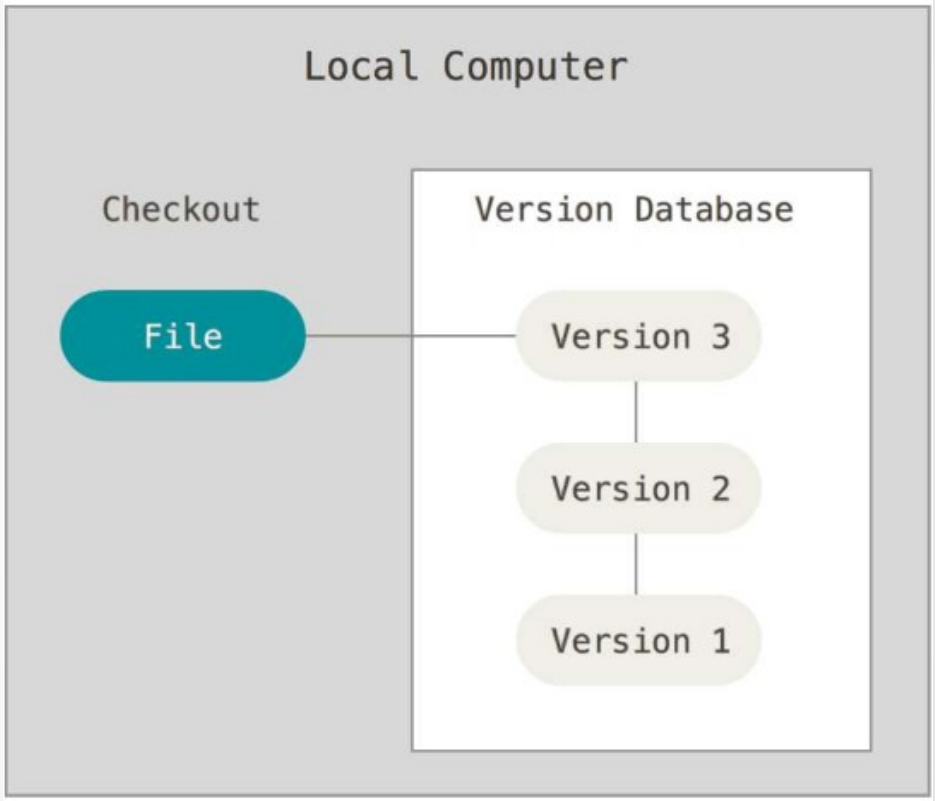
- **Контроль версий** (англ. Version control), также известный как контроль исходного кода (англ. Source control) - это практика отслеживания и управления изменениями в программном коде.
- **Системы контроля версий** (англ. version control systems или VCS,часто встречается название Revision Control System или RCS) - это программные инструменты, которые помогают группам разработчиков управлять изменениями исходного кода с учетом хронологии.
- Главные возможности системы контроля версий:
  - Полная история изменений каждого файла за длительный период
  - Возможность отслеживать каждое изменение, внесенное в файлы
  - Описание причин всех производимых изменений
  - Откат изменений, если что-то пошло не так
  - Совместная работа группы над одним проектом

Системы контроля версий

- **Репозиторий** — место, где хранятся и поддерживаются какие-либо данные.
- Также под репозиторием понимается каталог файловой системы, в котором могут находиться:
  - сами контролируемые файлы
  - журналы конфигураций сборки проекта
  - журнал операций, выполняемых над репозиторием
- **Версией** или **ревизией** (revision) называется конкретное зафиксированное состояние репозитория
- **Управление исходным кодом** (source control management, SCM) используется для отслеживания изменений в репозитории исходного кода

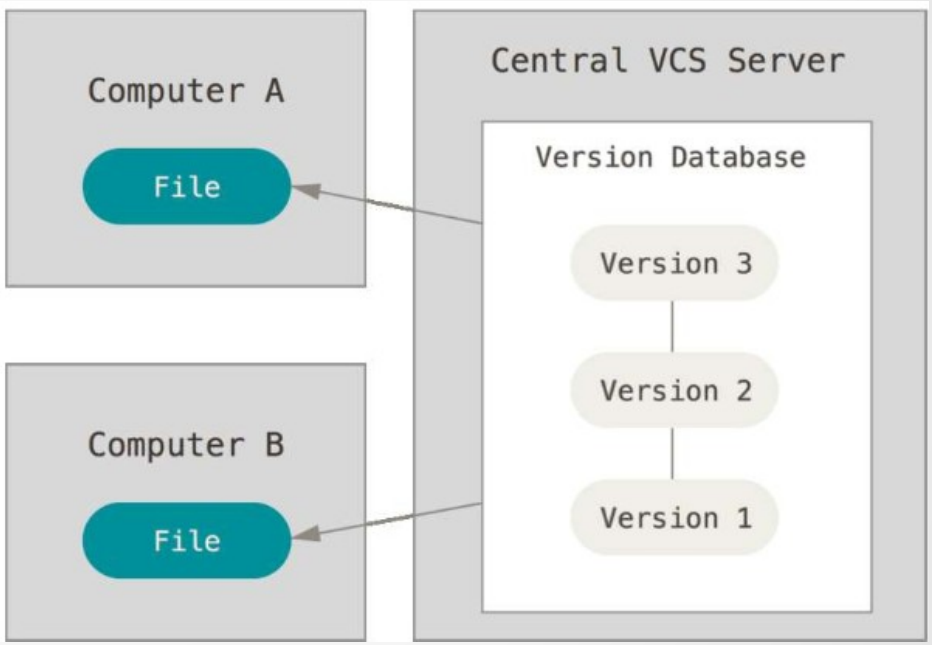
Виды систем контроля версий

- **Локальные системы контроля версий**
- Одной из наиболее известных VCS такого типа является rcs (Revision Control System).
- Утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске.
- Она позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.



Виды систем контроля версий

- **Централизованные системы контроля версий**
- Одно основное хранилище всего проекта
- Каждый пользователь копирует себе необходимые файлы из репозитория, изменяет и добавляет свои изменения обратно

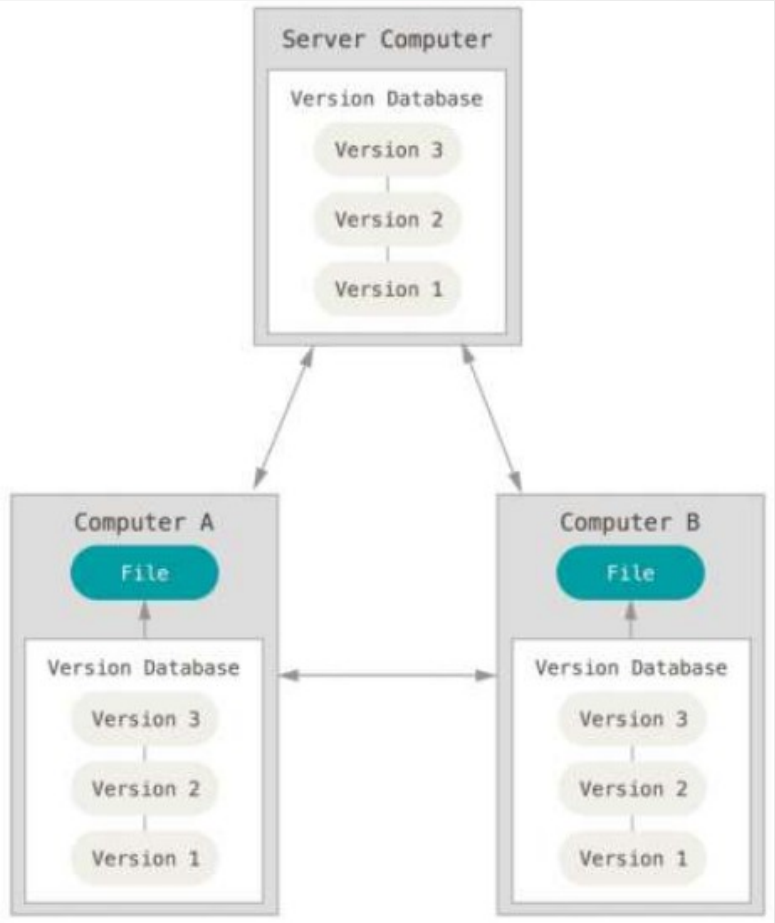


Виды систем контроля версий

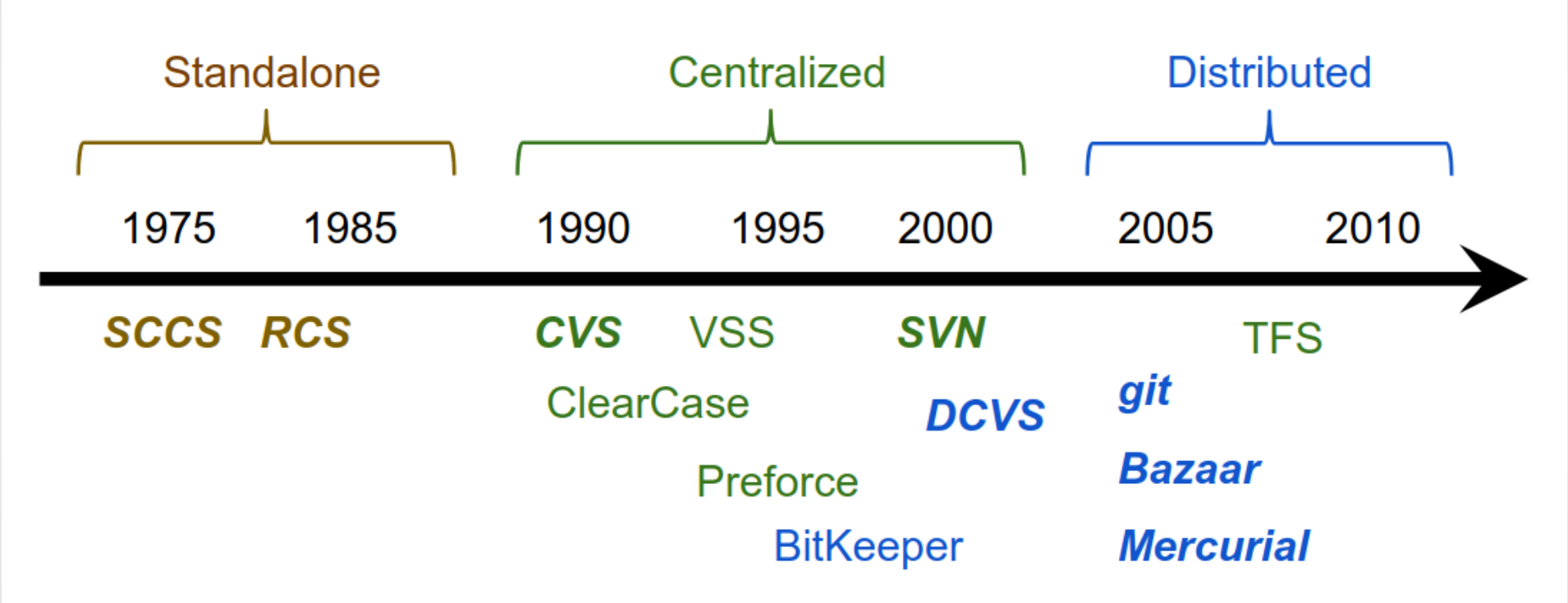
- **Централизованные системы контроля версий**
- Достоинства:
  - Всегда есть выделенная "эталонная" версия файлов
  - Централизованное администрирование
  - Привычный workflow
- Недостатки:
  - Единая точка отказа — сервер
  - Любые изменения влияют на всех пользователей
  - Требуется достаточно гибкая система прав
  - В очень больших или распределенных проектах централизованные механизмы работают плохо.
  - Репозитории уровня корпорации трудно администрировать

Виды систем контроля версий

- **Распределённые системы контроля версий**
- В таких системах как Git, Mercurial клиенты не просто выгружают последние версии файлов, а полностью копируют репозиторий.
- У каждого пользователя свой вариант (возможно не один) репозитория
- При этом можно выделить центральный репозиторий, в который будут отправляться изменения из локальных и с ним же эти локальные репозитории будут синхронизироваться
- Присутствует возможность добавлять и забирать изменения из любого репозитория



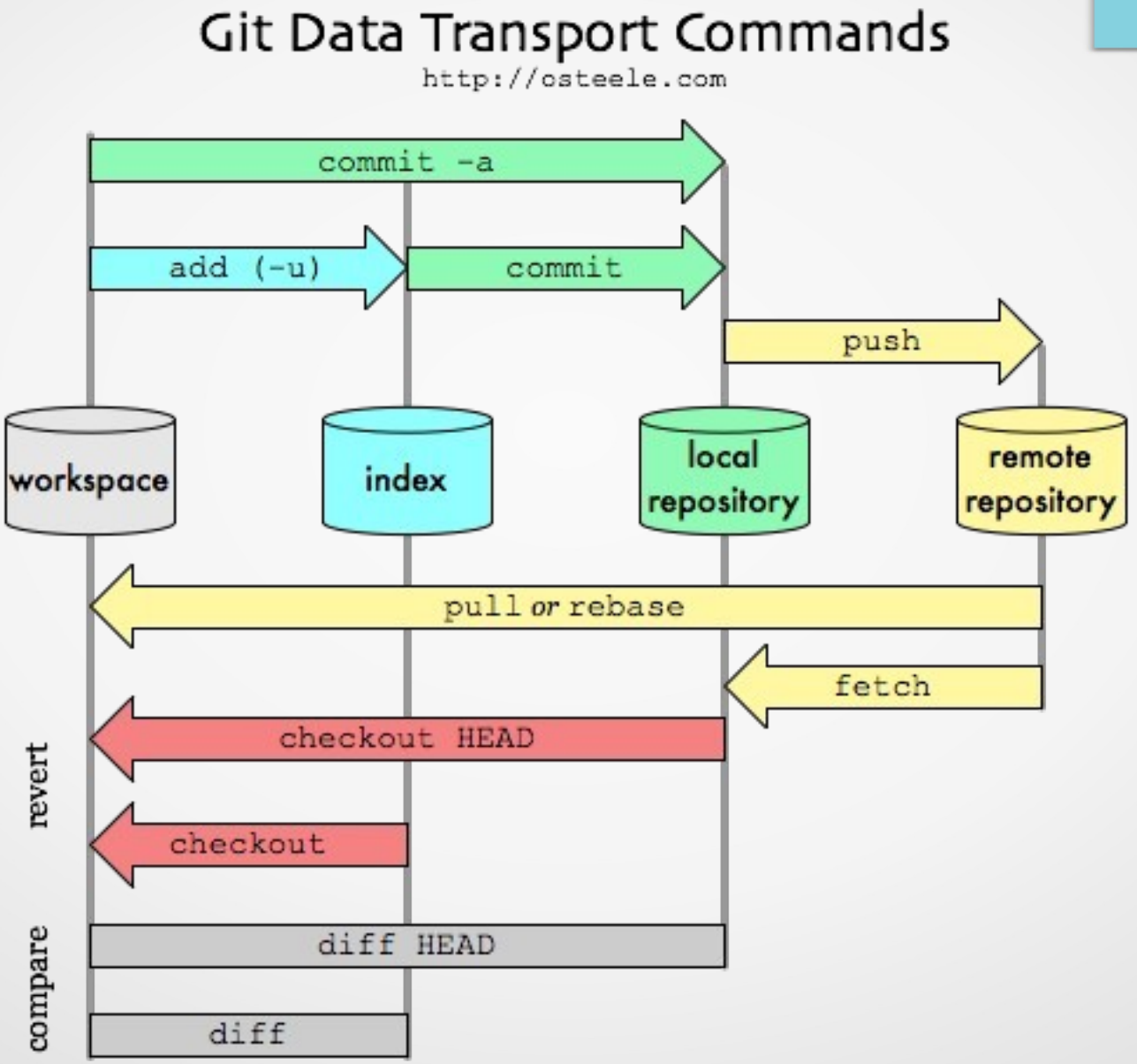
История систем контроля версий



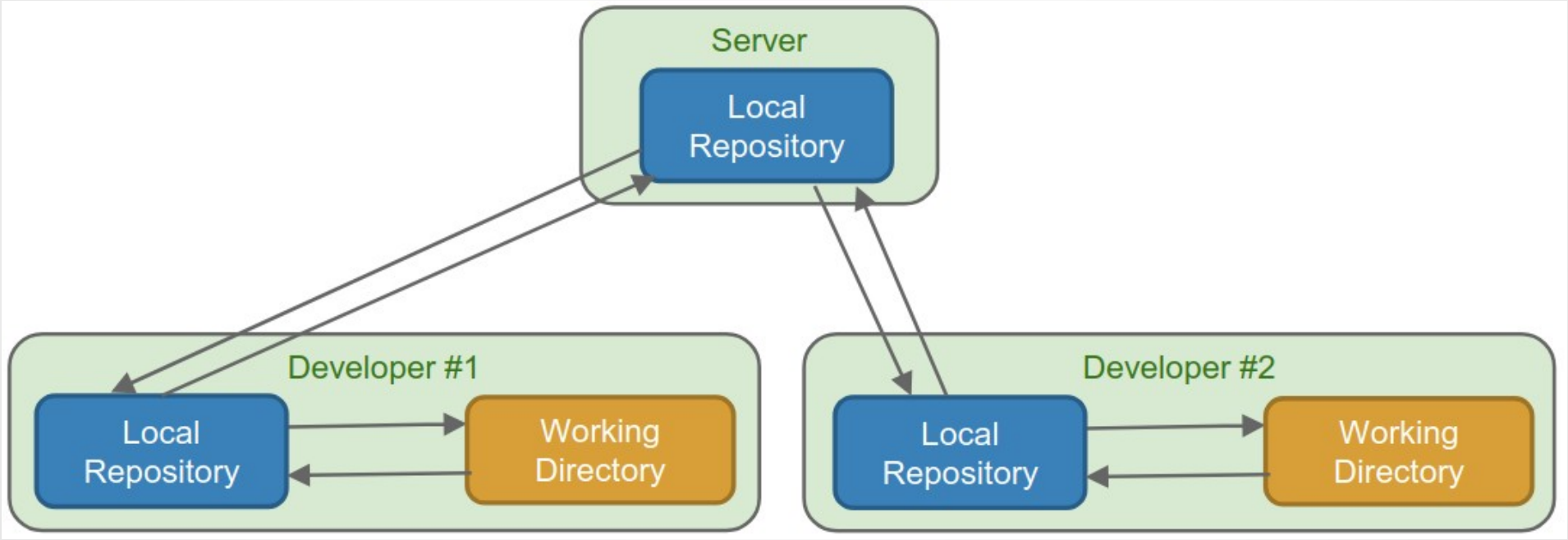
Терминология

- **tag** (Метка) - именованная версия проекта
- **branch** (Ветка) - метка, имеющая историю развития
- **commit** (Коммит, фиксация) - сохранение изменений в репозитории
- **checkout** (Чекаут, извлечение) - получение рабочей копии (без истории)
- **push/pull** — отправка/получение изменений в/из другого репозитория
- **main branch/master branch** - основная ветка
- **Working directory/working copy** (рабочая копия) - снимок одной из версий проекта

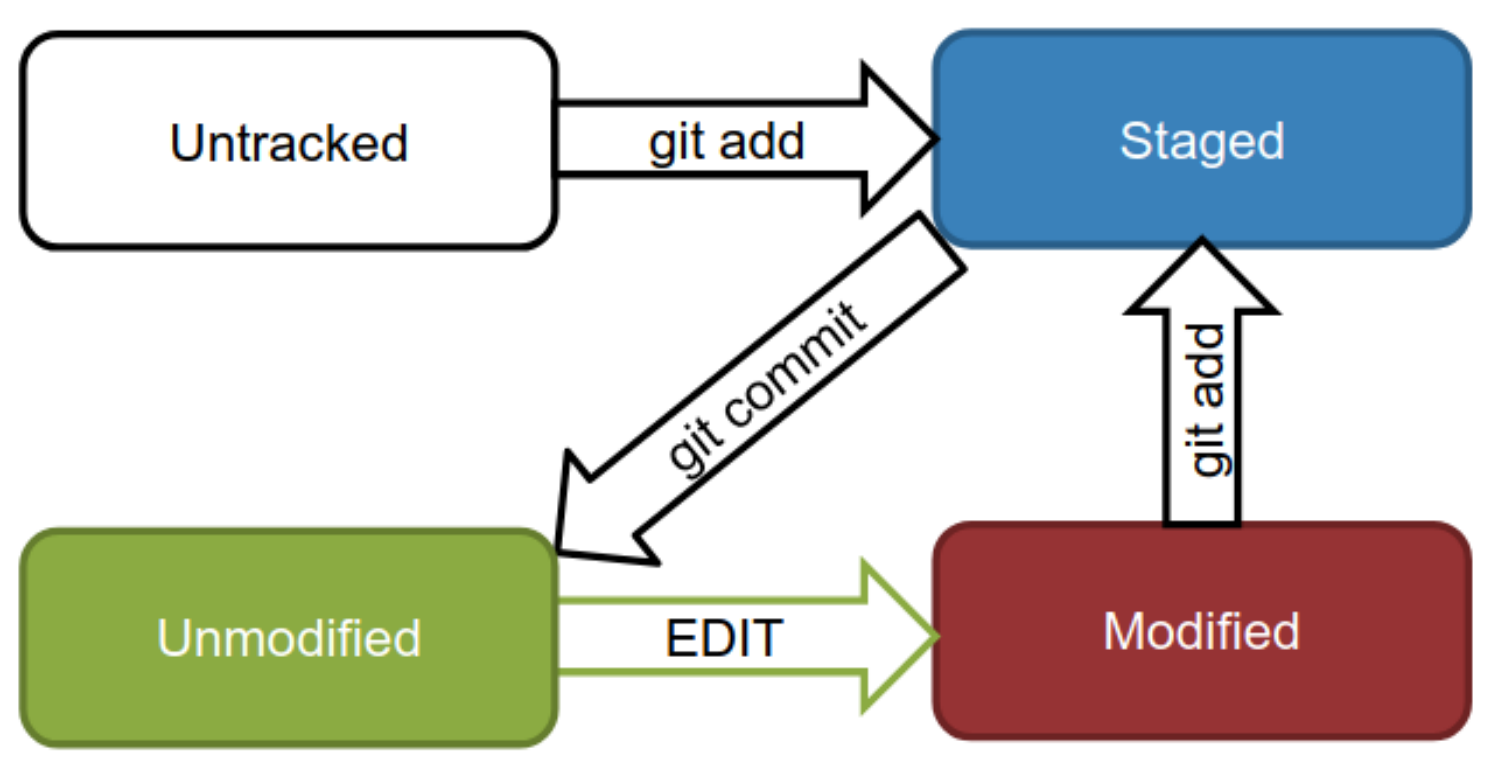
Git: Общая схема работы



Git: работа с сервером



Git: Статус файла



Git: Работа с индексом (stage)

- Команда **add** добавляет измененные файлы в stage.

  $git add filename

- Команда **rm** помечает файл в stage как удаленный.

  $git rm filename

- Команда **mv** перемещает/переименовывает файл

  $git mv old_filename new_filename

- Команда **reset** сбрасывает текущий stage.

  $git reset HEAD filename

- Команда **status** отображает текущий статус индекса (измененные и неотслеживаемые файлы)
- $git status

Git: Работа с локальным репозиторием

- Команда **commit** сохраняет текущий stage в локальный репозиторий.

  $git commit

  $git commit --amend

- Команда **branch** создает/удаляет новую ветку.

  $git branch

  * main

  $git branch test

  $git branch

  * main

  test

- Команда **checkout** переключает рабочую копию на другую ветку.

  $git checkout test

  Switched to branch 'test'

Git: Commit message

- Комментарий не должен быть пустым.
- Комментарий должен отвечать на вопрос что и зачем было сделано.
- Комментарий должен быть лаконичным (twitter style)
- Комментарий должен быть на английском языке
- В комментарии указывают ссылку на задачу в багтрекере (+ операцию над ней)

  Пример: «Closes #4, #6, Related to #5»

Git: Работа с удаленным репозиторием

- Команда **clone** клонирует репозиторий и создает рабочую копию.

  $git clone git@github.com:seekerk/gtest.git

  Cloning into gtest...

  ...
- Команда **push** отправляет изменения в удаленный репозиторий.

  $git push *origin main*
- Команда **pull** забирает изменения указанной ветки из удаленного репозитория и сливает их в текущую ветку.

  $git pull *origin master*
- Команда **fetch** забирает все изменения из удаленного репозитория.

  $git fetch

Git: Временное хранилище (stash)

- Команда **stash** (или продвинутый вариант **stash push**) помещает изменения из stage во временное хранилище и сбрасывает рабочую копию.

  $git stash

- Команда **stash pop** получает изменения из временного хранилища

  $git stash pop

- Получение изменений без удаления из хранилища с помощью команды **stash apply**

  $git stash apply

- Список спрятанных изменений: **stash list**
- Просмотр спрятанного изменения: **stash show**
- Удаление спрятанного изменения: **stash drop**

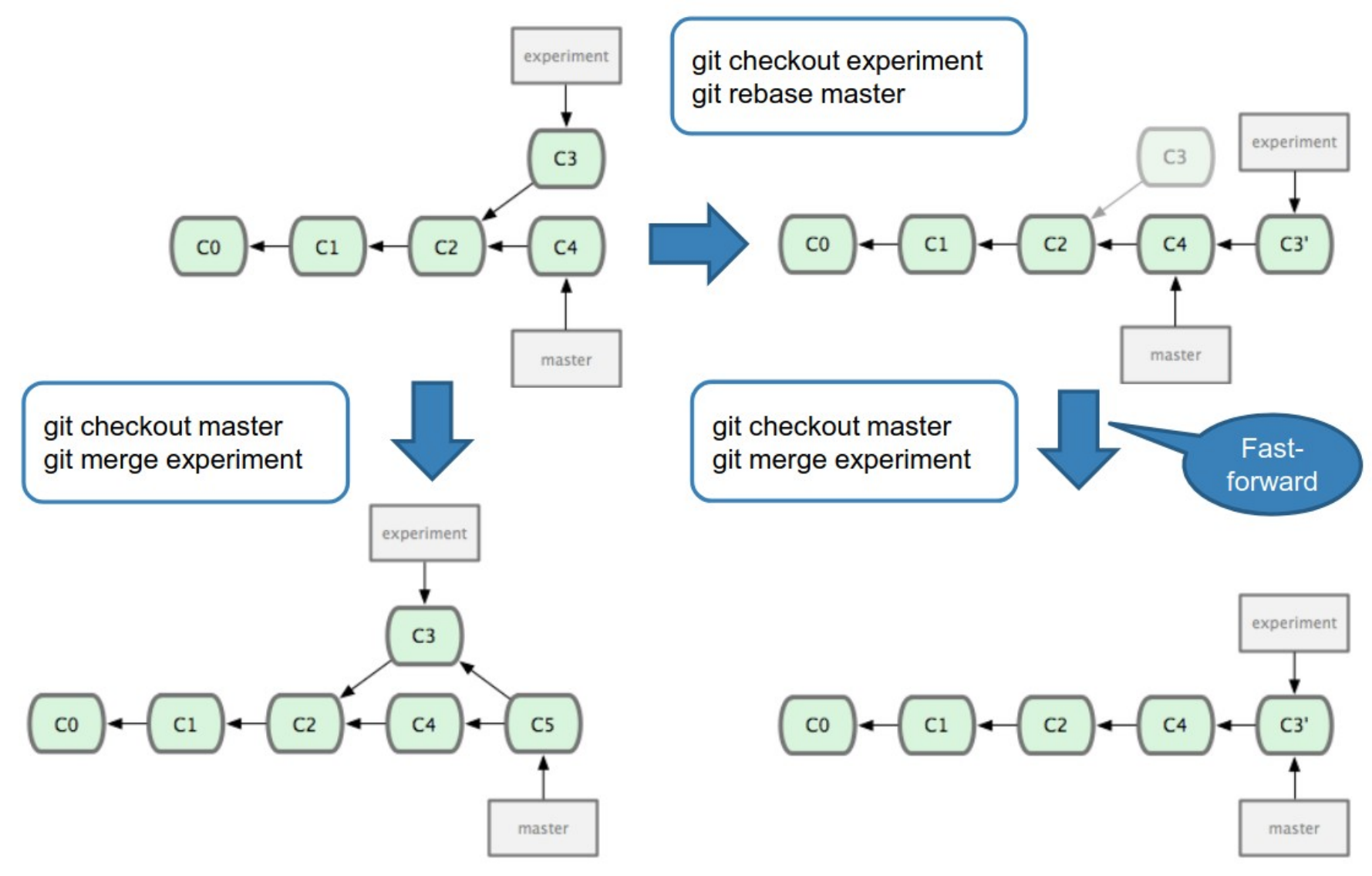
Git: Ключи

- Контентно-адресуемое key-value хранилище (.git)
- В качестве ключа используется SHA-1 хэш от содержимого
- Указатели на выделенные точки в репозитории
- В качестве идентификаторов ревизий используются хэши
  - Общепринято использовать первые 6-8 символов хэша, но это не однозначно

Git: Ветки

- Ветки это указатели на определенные коммиты
- Release branch. Параллельная разработка и поддержка старых версий.
- Feature branch. Разработка нового функционала без риска сломать работающий код.
- Частые коммиты в собственную ветку
- **git branch** + **git checkout**: создание ветки и переключение на нее
- **git merge**: слияние в текущую ветку
- **git rebase**: перестройка дерева коммитов в текущей ветке

Git: Merge vs. Rebase



Git: Конфликты слияний

- Совместное редактирование одних и тех же строк
- Редактирование рядом стоящих строк
- Изменение бинарных файлов

```
      <div id="navigation">
        <ul>
<<<<<<< HEAD
          <li><a href="index.html">Home</a></li>
          <li><a href="about.html">About Us</a></li>
          <li><a href="product.html">Product</a></li>
          <li><a href="imprint.html">Imprint</a></li>
=======
          <li><a href="returns.html">Returns</a></li>
          <li><a href="faq.html">FAQ</a></li>
>>>>>>> develop
        </ul>
      </div>
```

Git: Конфликты слияний

Git: предупреждение добавления «мусора»

- Часто появляются временные и вторичные файлы в директории
- Файл .gitignore используется для игнорирования таких файлов (wildcard синтаксис)
- Пример:

```
# игнорим логи запусков
*.log
build/
temp-*
```

Git: полезные команды

- **git init** — инициализация репозитория
- **git diff** — просмотр изменений в рабочей копии
- **git log** – показывает лог commits начиная с HEAD
- **git remote** – показывает информацию об удаленных репозиториях, а также позволяет удалять и добавлять их
- **git pull request** — запрос на изменение
- **git reset** — откат изменений рабочей копии
- **gitk** / **git gui** – визуальная утилита для работы с репозиторием

- Шпаргалка: https://training.github.com/downloads/ru/github-git-cheat-sheet/

Git: Установка и настройка

- ОС Linux: стандартные репозитории

  apt-get install git

  zypper in git
- MacOS: через Homebrew

  brew install git
- Windows: https://git-scm.com/

Git: Первоначальная настройка

- Настройка информации о пользователе для всех локальных репозиториев
  - Устанавливает имя, которое будет отображаться в поле автора у выполняемых вами коммитов

    $ git config --global user.name "[имя]"
  - Устанавливает адрес электронной почты, который будет отображаться в информации о выполняемых вами коммитах

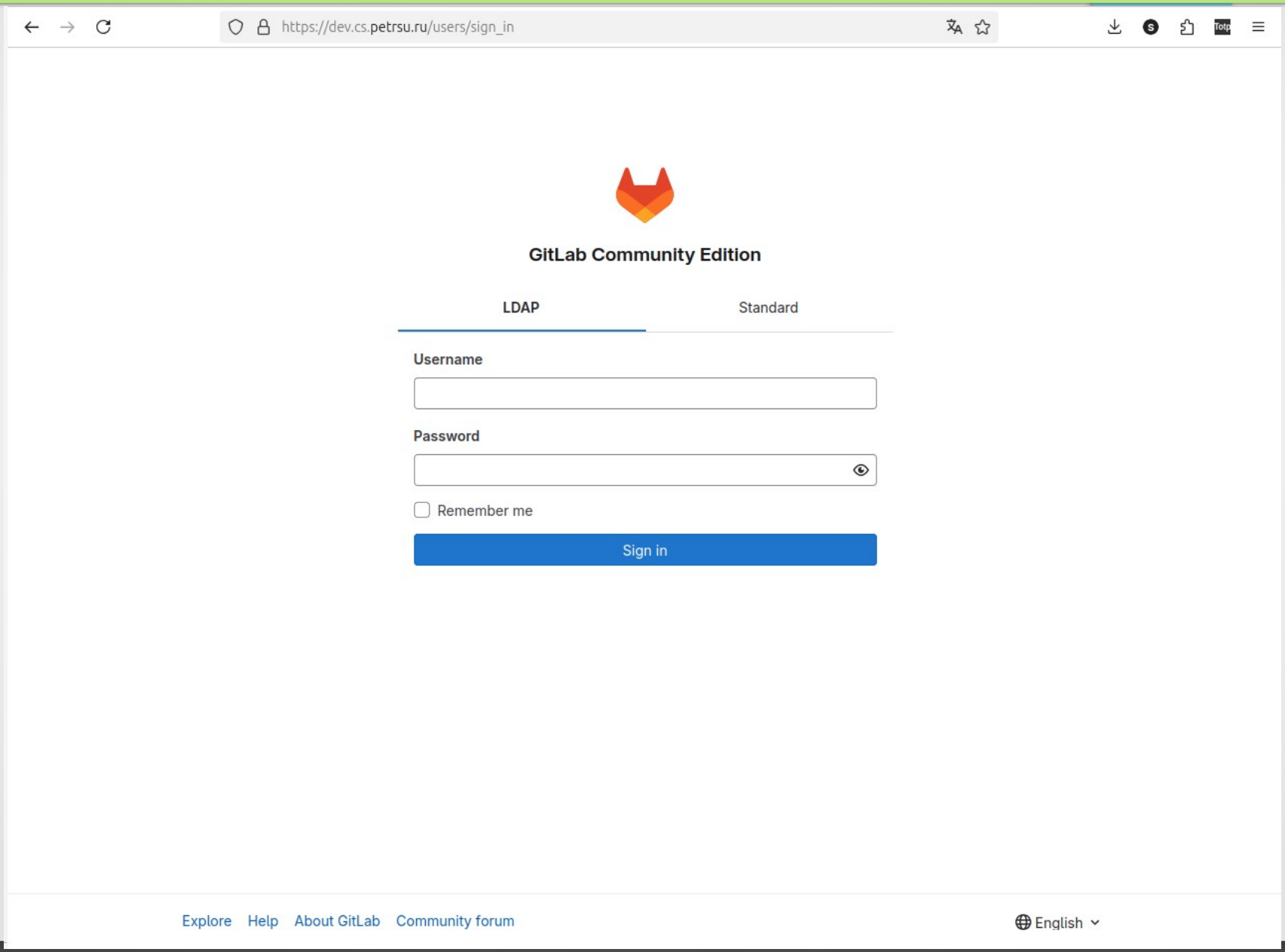
    $ git config --global user.email "[адрес электронной почты]"

Облачный репозиторий

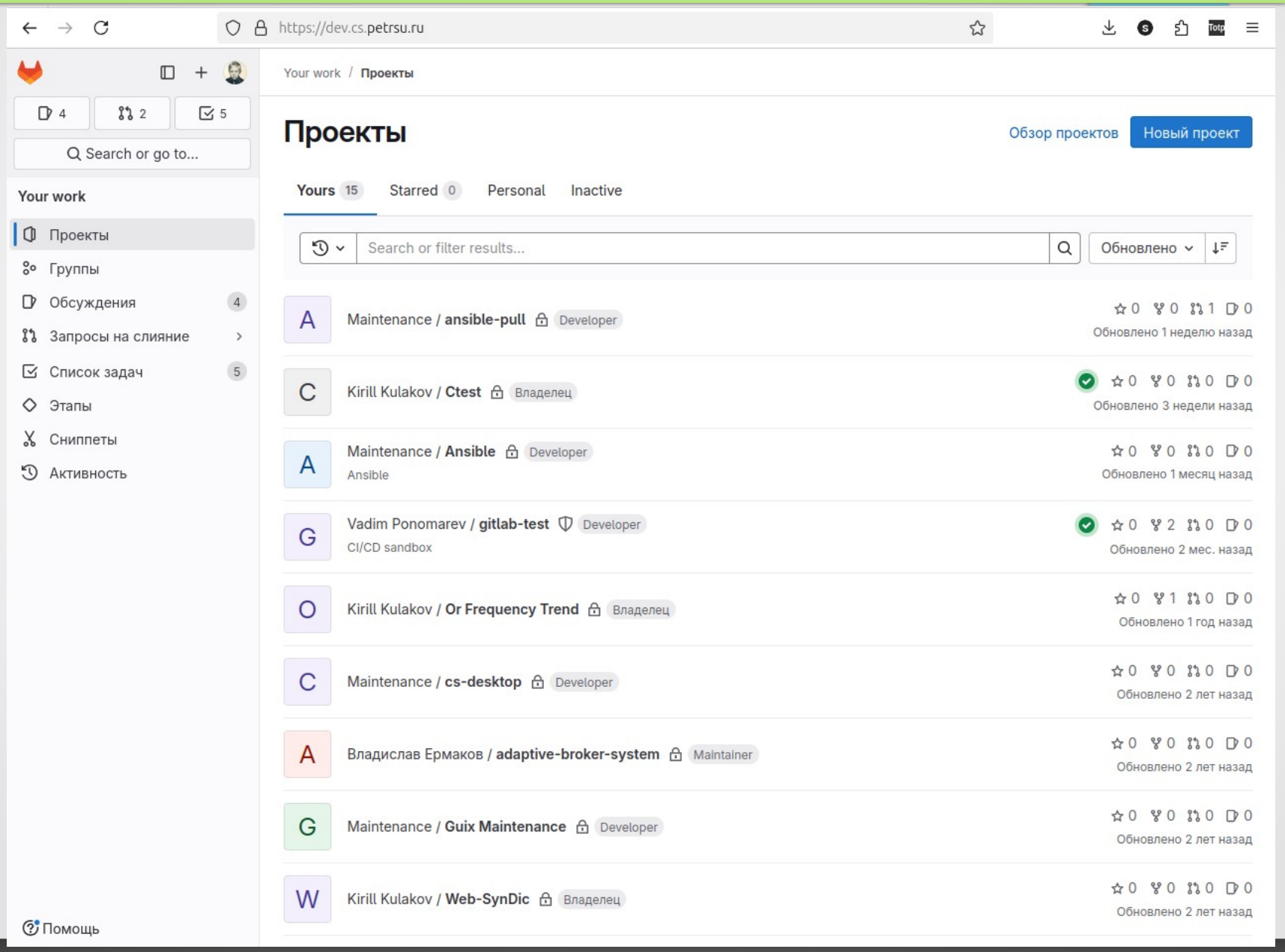
- Специальный сервер — удаленный репозиторий кода
  - Предоставление интерфейса Git репозитория
  - Управление правами и ролями
  - Web интерфейс доступа в т.ч. к содержимому репозитория
  - Вики система
  - Система отслеживания проблем / Bug tracking system (Issues)
  - Continuous Integration/ Continuous Delivery (CI/CD)
  - Интеграция с экосистемой разработки

- Из самых известных это GitHub, GitLab, GitFlic, Bitbucket

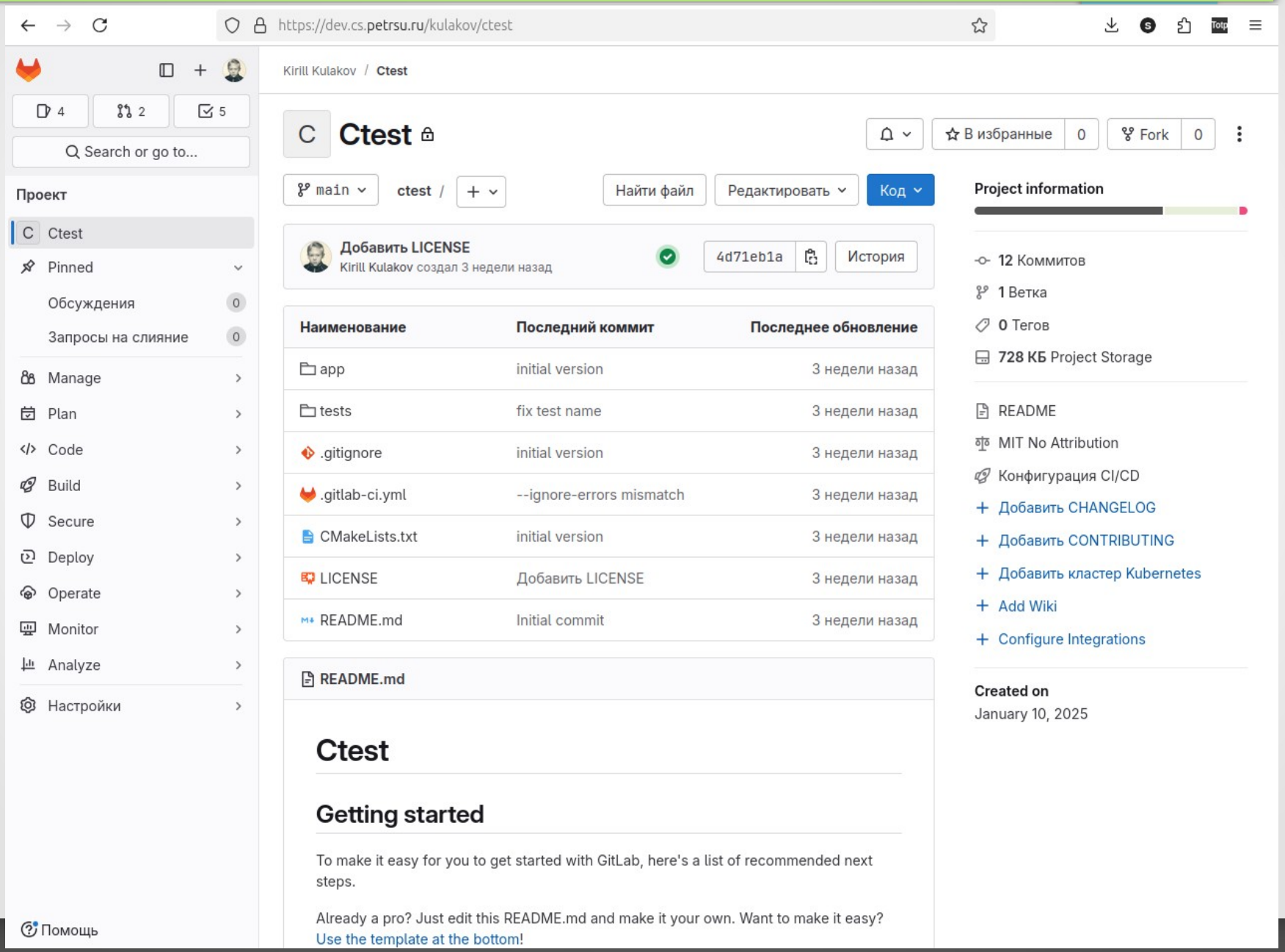
Gitlab: окно входа



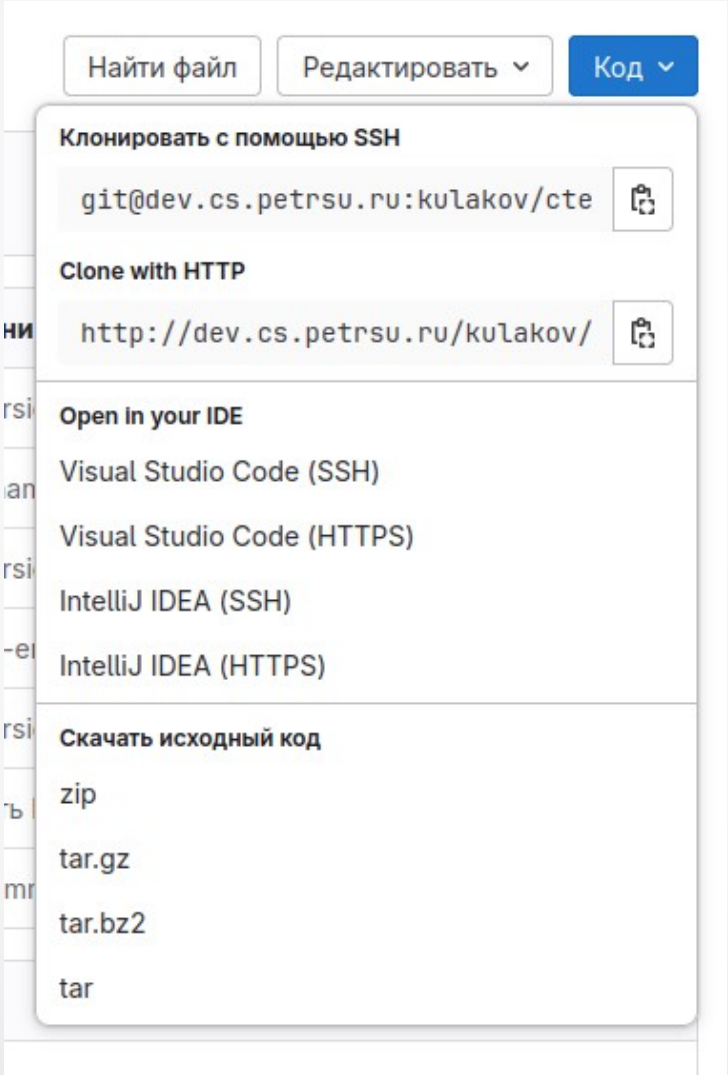
Gitlab: проекты



Gitlab: просмотр проекта

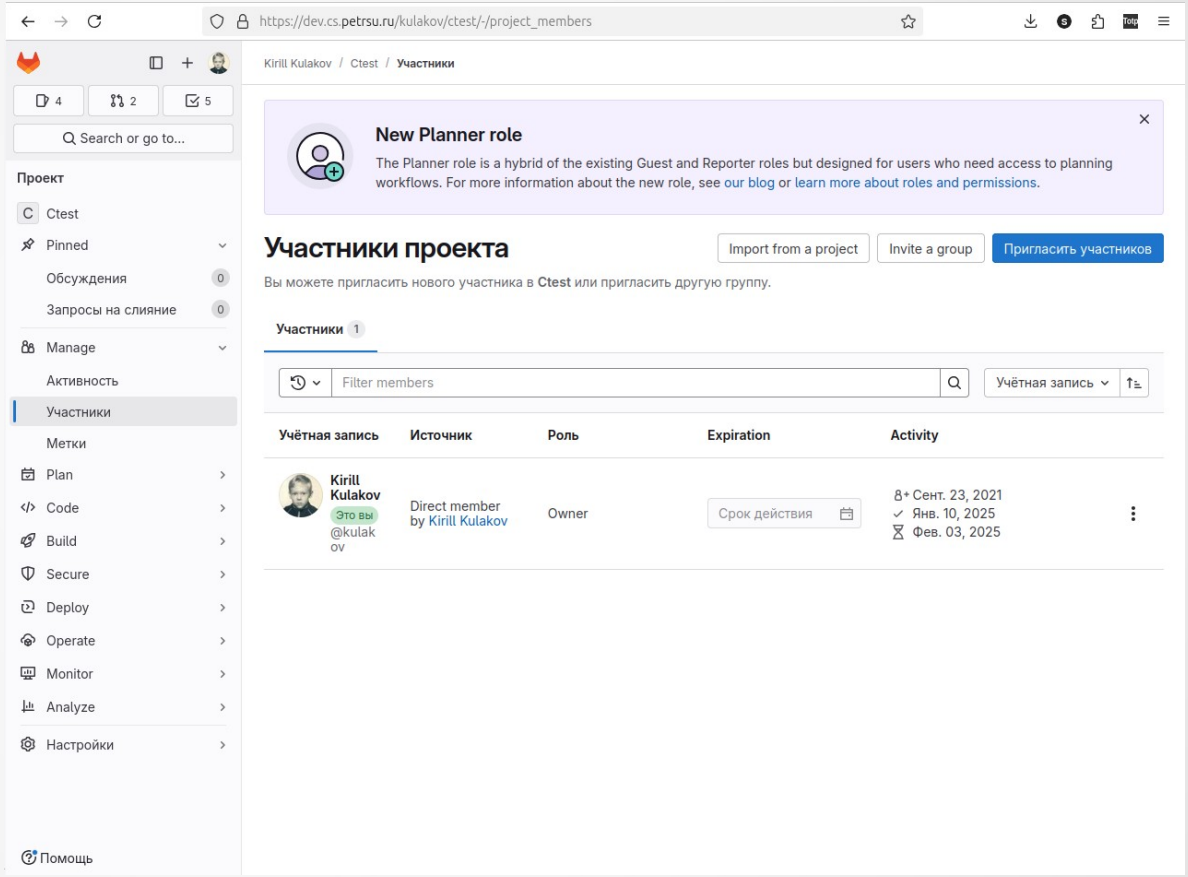


Gitlab: получение доступа к репозиторию

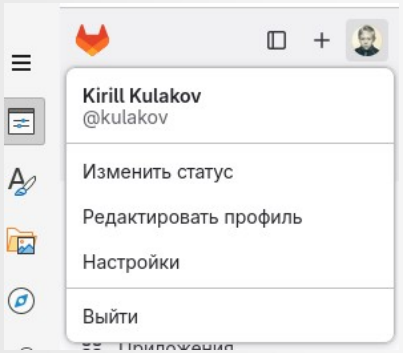


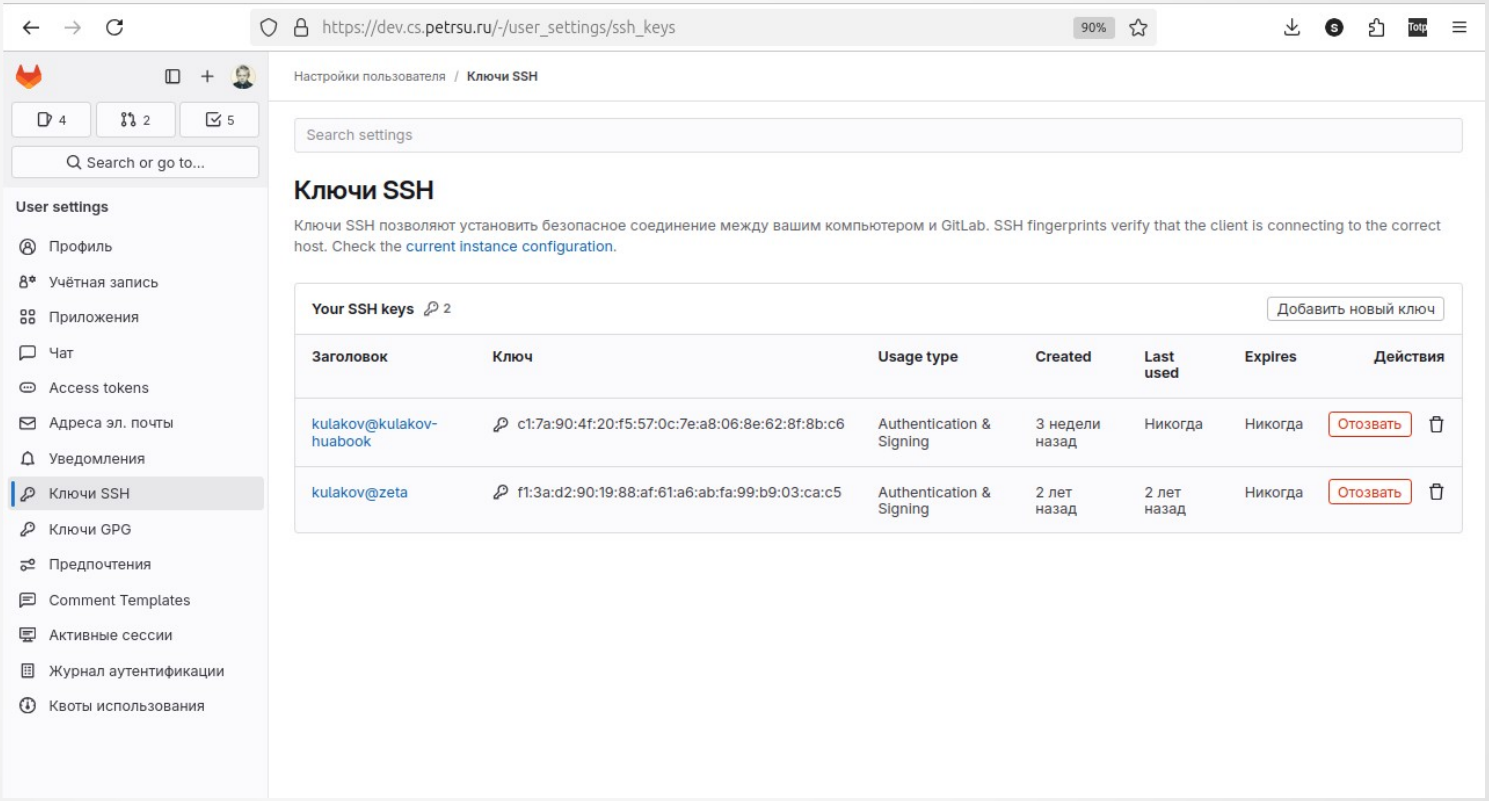
Gitlab: Пользователи и роли

- **Guest** - Гость (доступ к приватным репозиториям)
- **Reporter** - Репортер
- **Developer** - Разработчик
- **Maintainer** - Сопровождающий
- **Owner** - Владелец



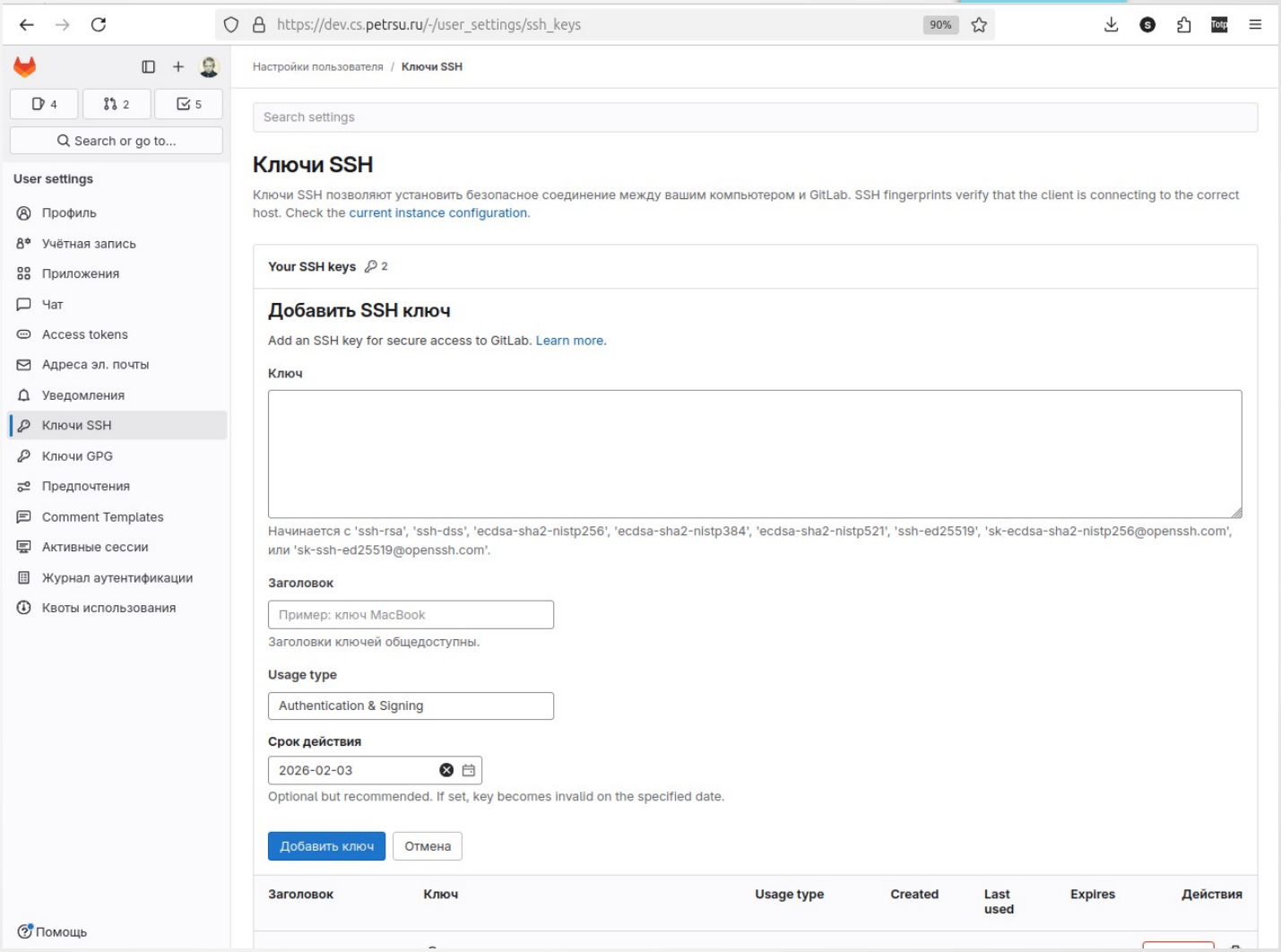
Доступ через ssh



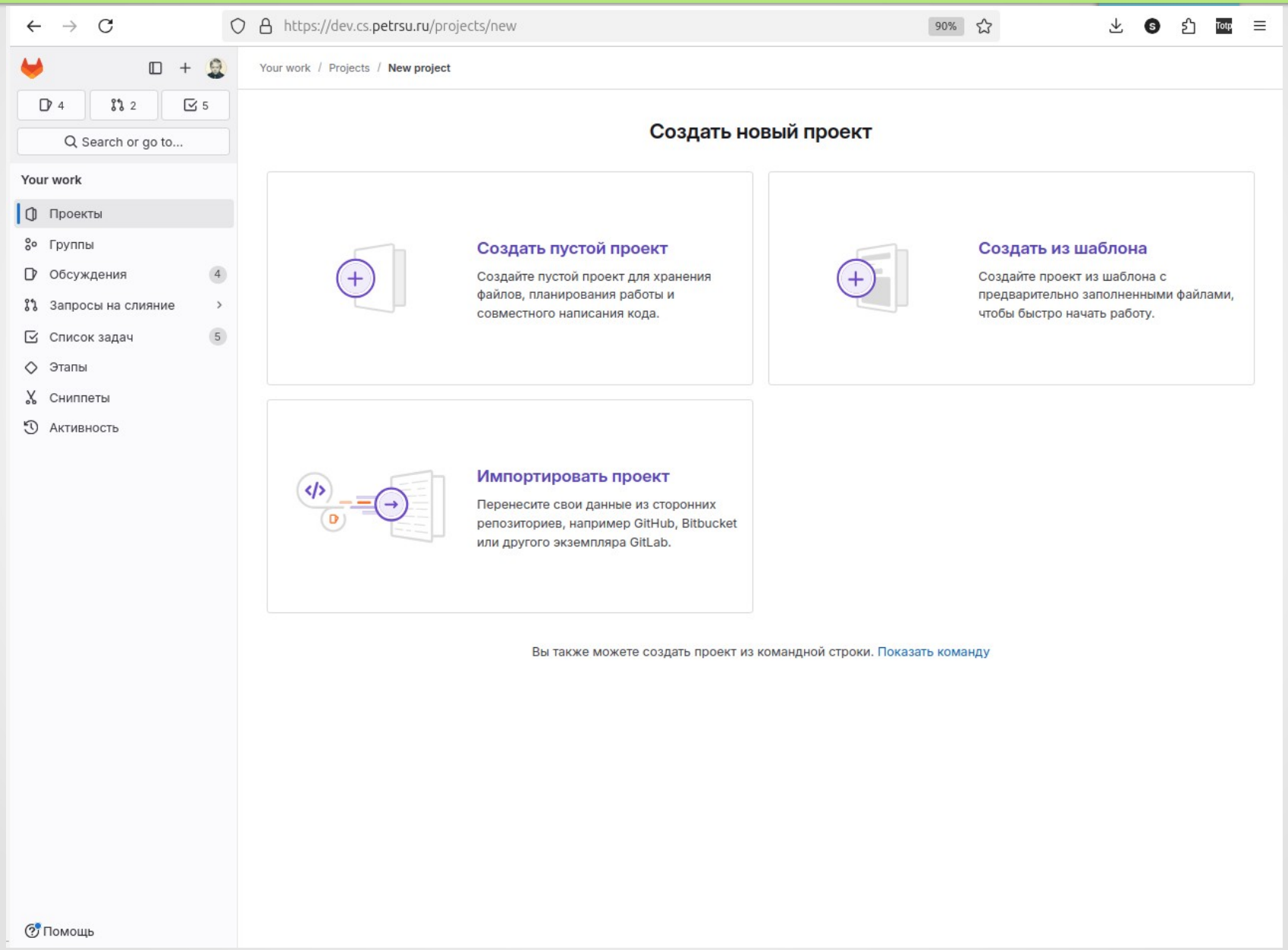


Доступ через ssh

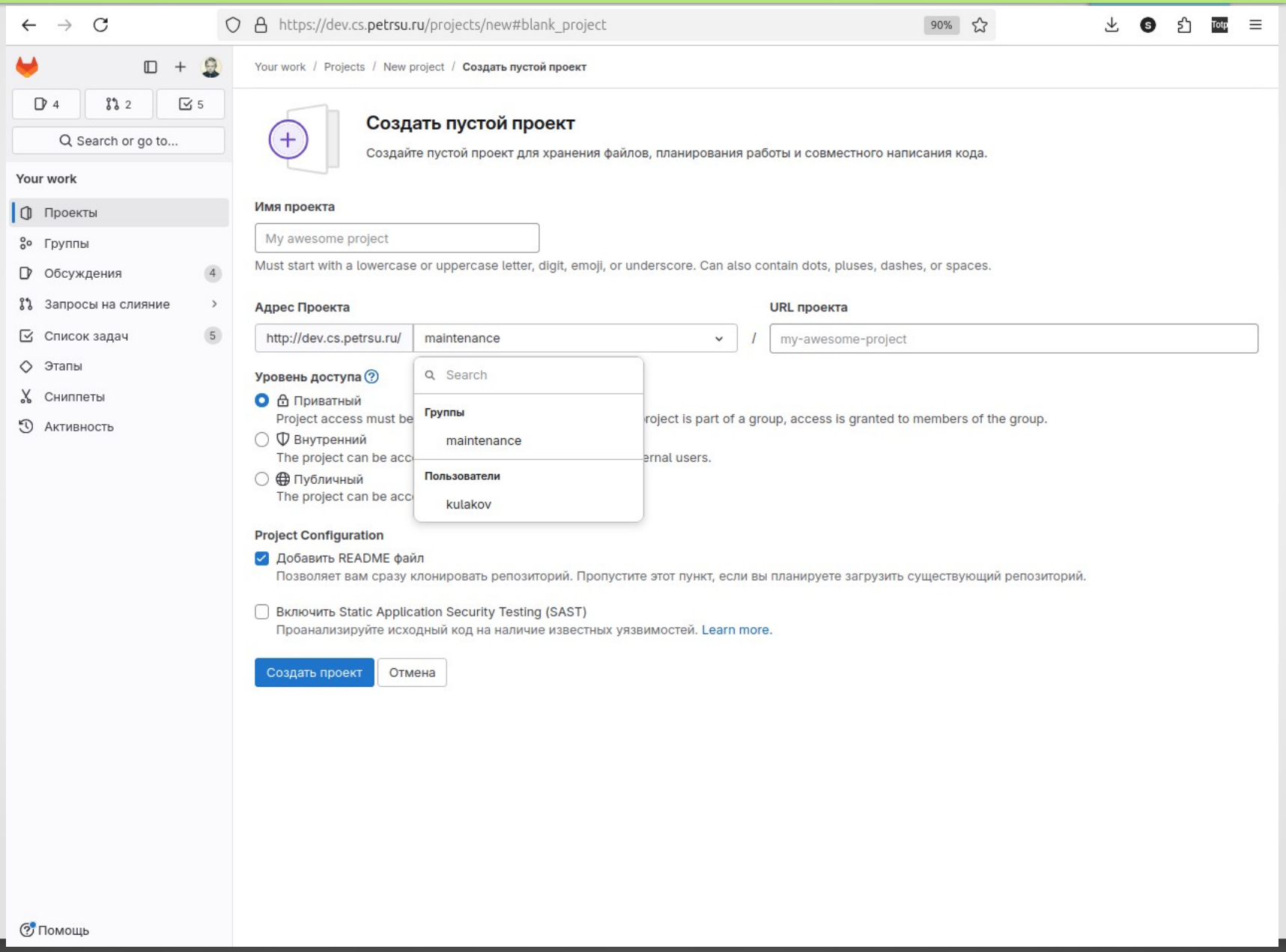
- ssh-keygen
- ~/.ssh/ id_rsa.pub



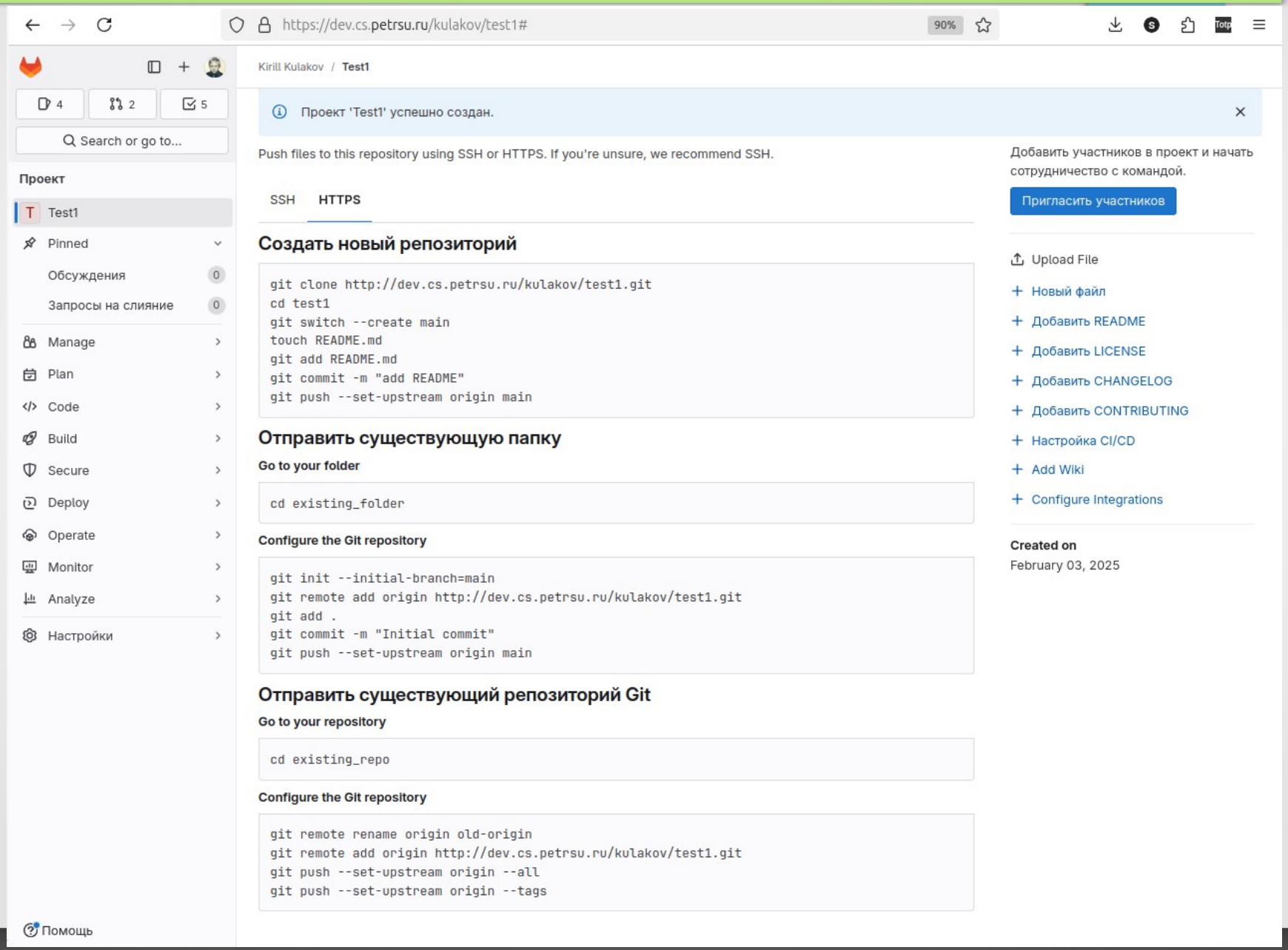
Создание проекта



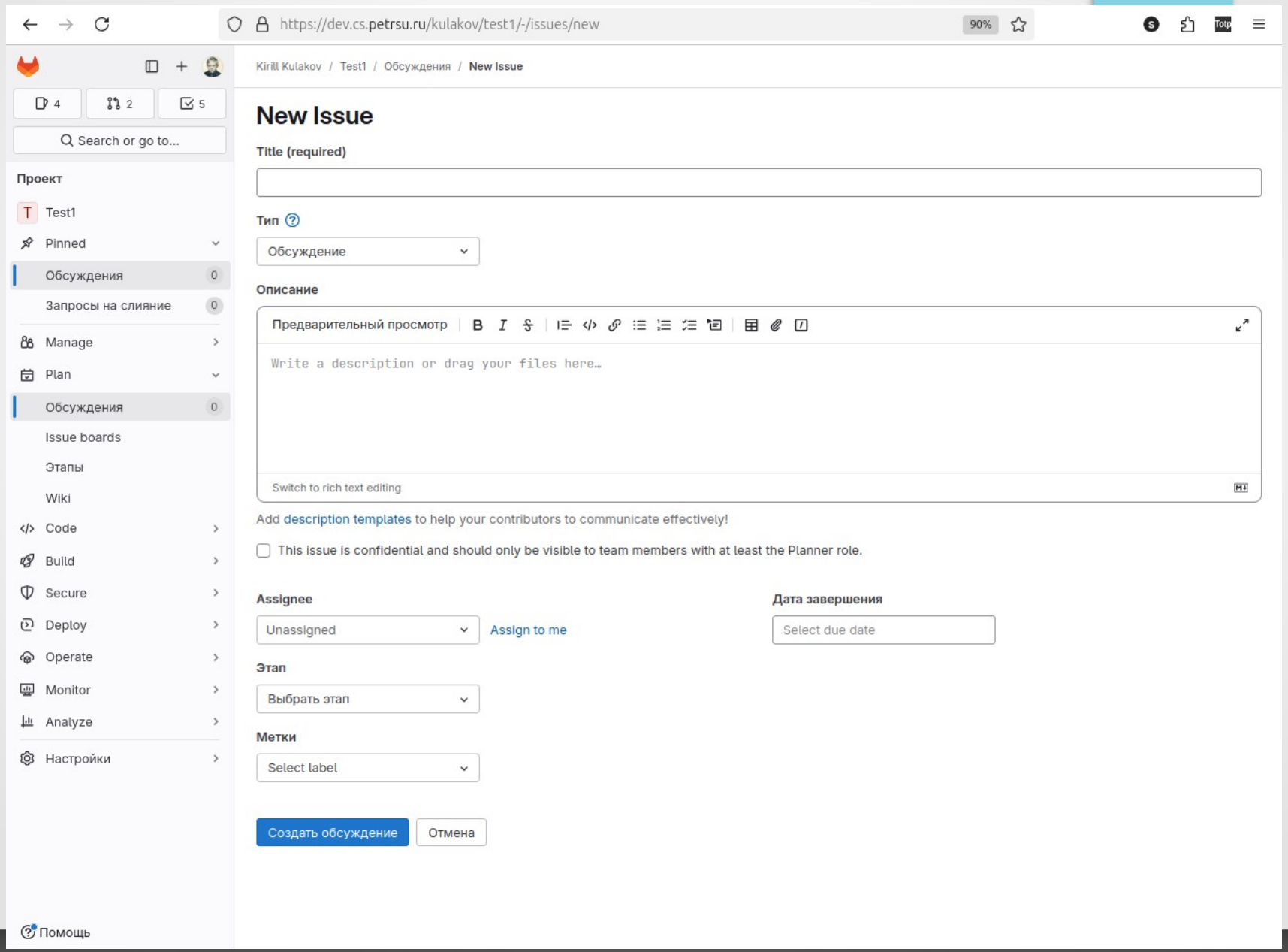
Создание проекта



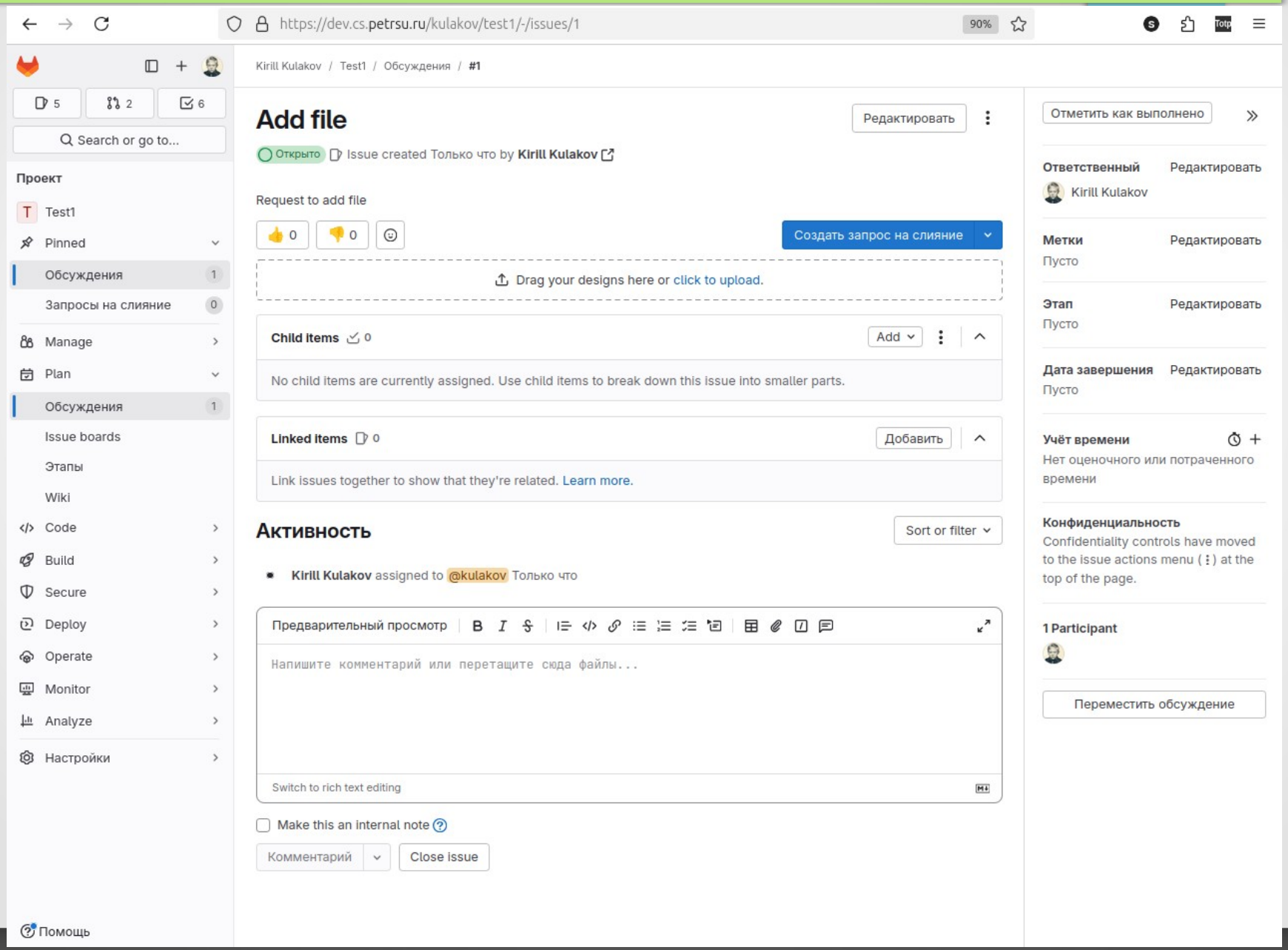
Создание проекта



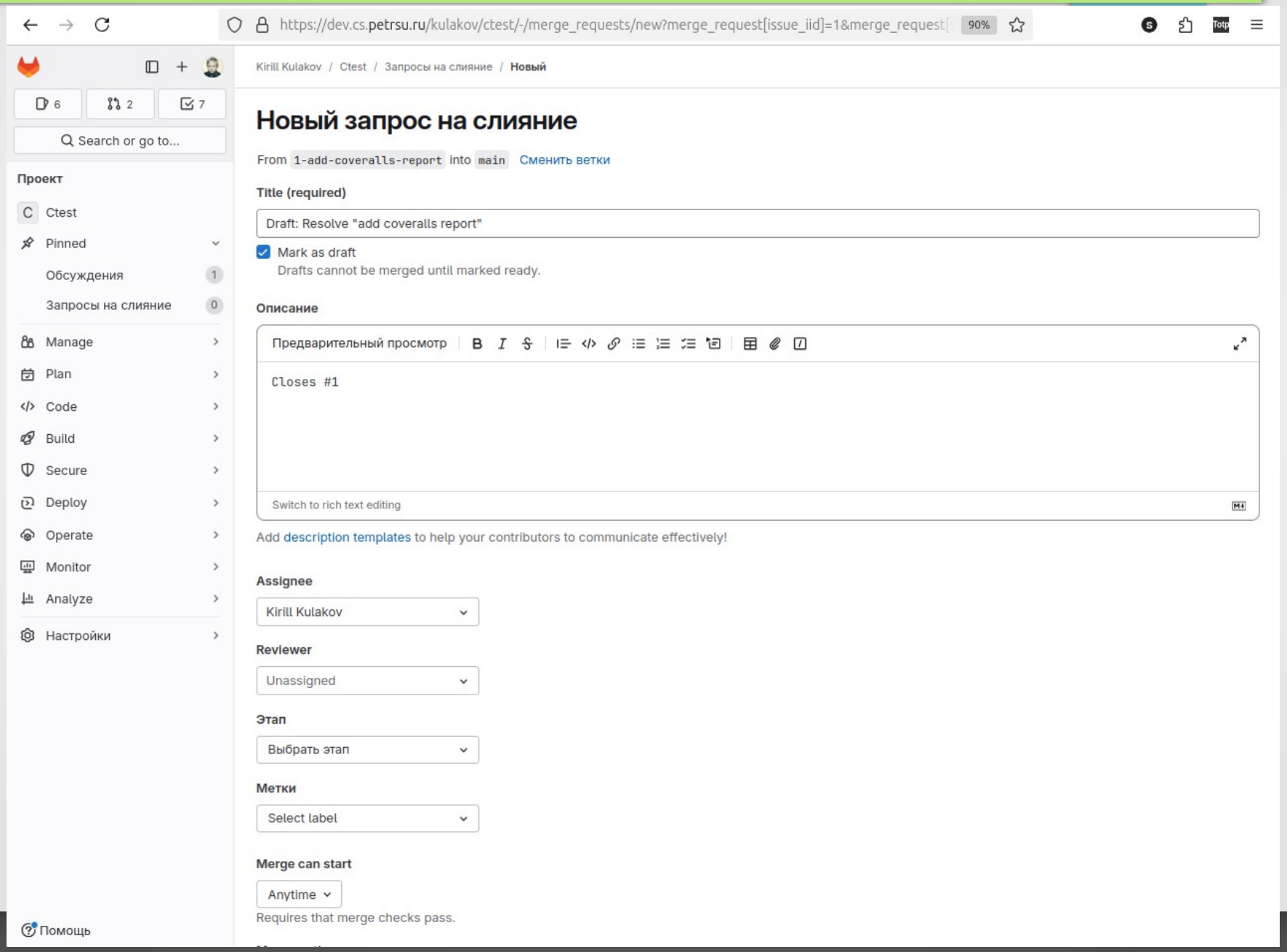
Git Workflow: Создание обсуждения



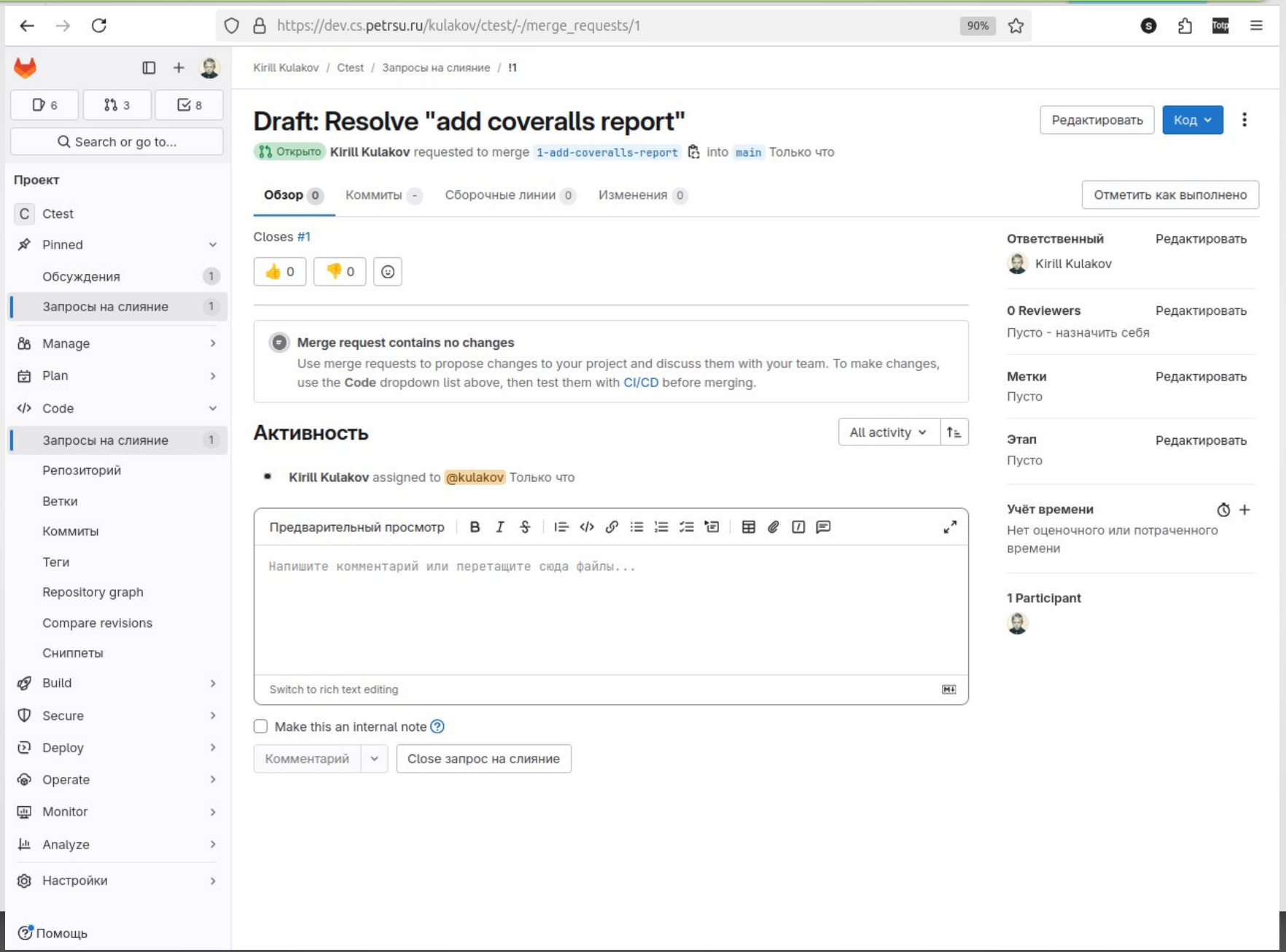
Git Workflow: просмотр обсуждения



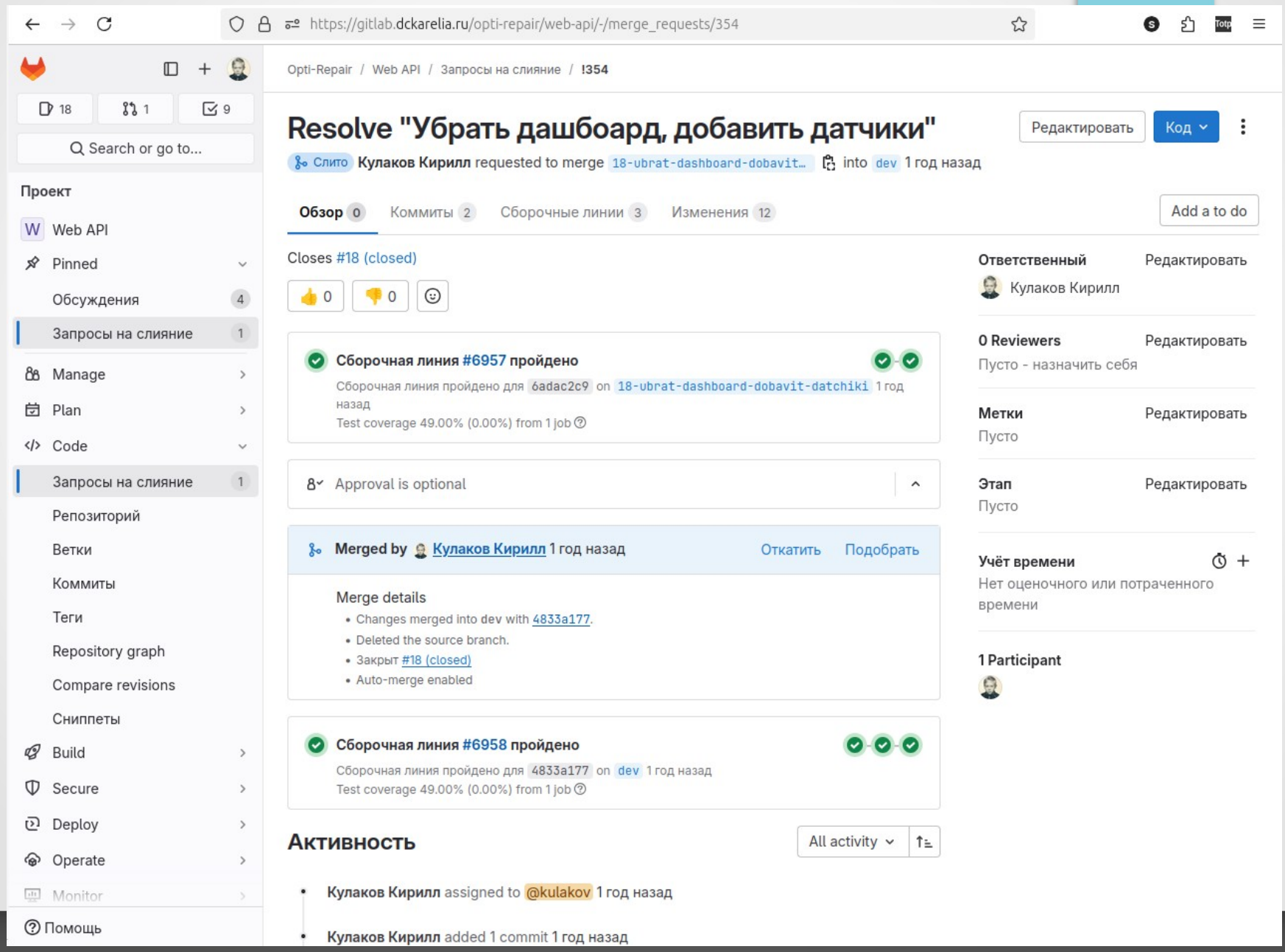
Git Workflow: запрос на слияние



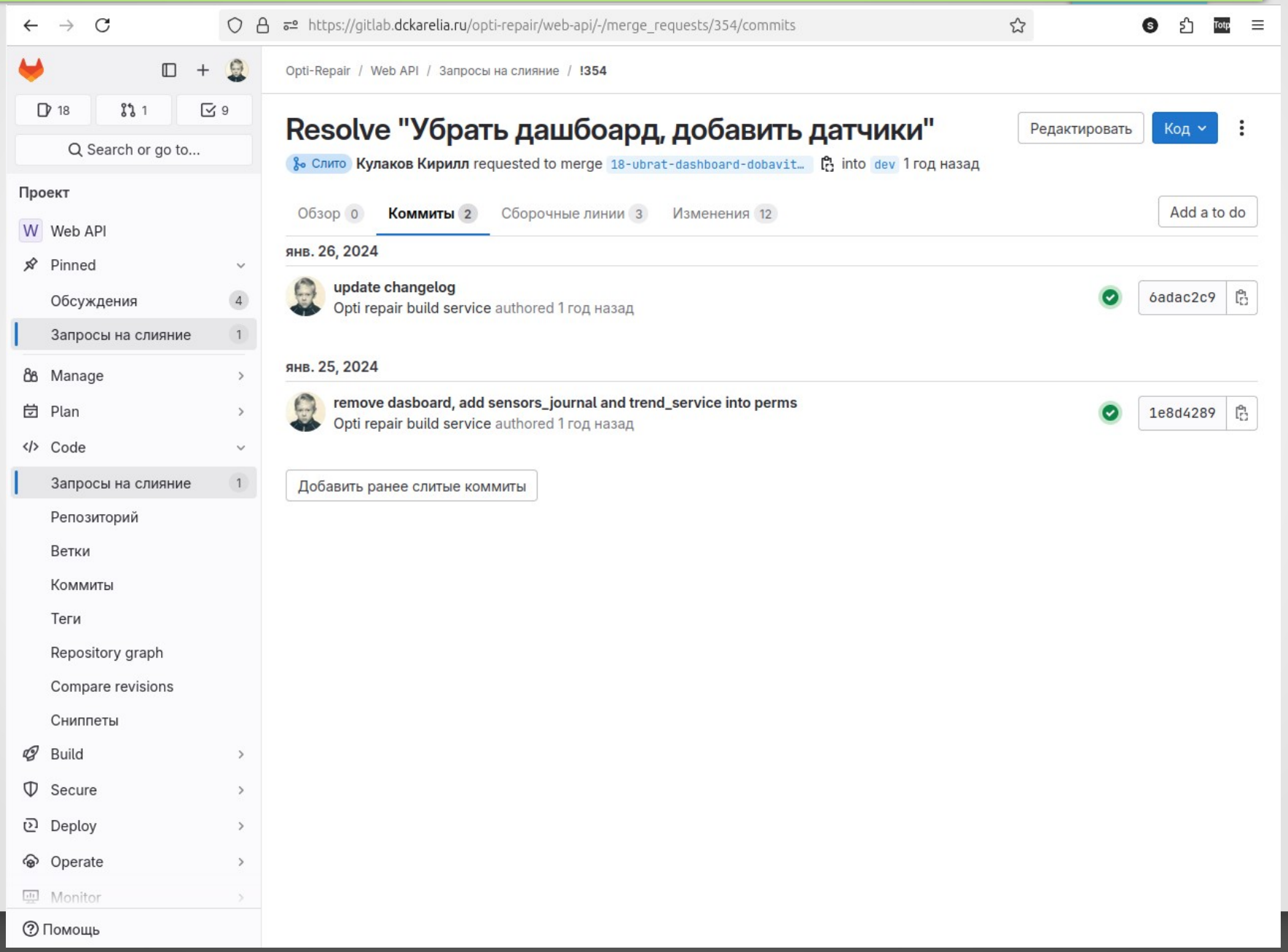
Git Workflow: статус пустого запроса на слияние



Git Workflow: после коммитов в ветку + CI/CD



Git Workflow: просмотр коммитов запроса



Git Workflow: просмотр изменений запроса