Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКИЕ РАБОТЫ

ПО МДК 04.02 «Обеспечение качества функционирования компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент  группы П50-4-20  Недвецкая М.А. | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ К.С. Образцова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 года |

Москва 2023

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА №1. Работа с Git 3](#_Toc147533933)

[практическая работа №2. Работа Git-хостингами 14](#_Toc147533934)

# ПРАКТИЧЕСКАЯ РАБОТА №1. Работа с Git

Цель работы: Написать конспект лекции, продемонстрировать конфигурацию Git, создать локальный репозиторий, где показать работу основных команд для работы с файлами и описать работу с указателями HEAD

Ход работы:

1. Конспект лекции

Жизненный цикл ПО

**Каскадные модели разработки**

Waterfall

- Каждый этап один за другим, нельзя вернуться назад.

- Этапы циркулируют только после завершения последнего этапа.



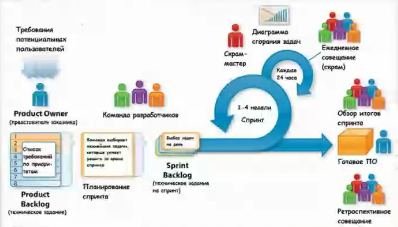
SCRUM

- В конце спринта имеется готовый продукт.

- Гибкое изменение требований.

- Быстрое реагирование на изменение трендов.

- Бюджет не фиксирован.



Основные этапы разработки ПО

Анализ

Разработка

Сборка

Тестирование

Развертывание

Релиз

Сопровождение

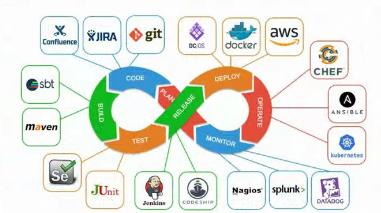
Мониторинг

DevOps

-Бесперебойная поставка ПО с помощью непрерывной интеграции рабочих процессов.

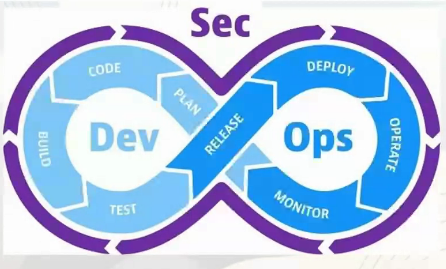
- Ускорение этапов разработки.

- Снижение ошибок, связанных с человеческим фактором.



DevSecOps

- Интегрирует безопасность в процессы разработки и автоматизации.



Концепции составляющие DevOps

Инфраструктура как код

Методология, которая предлагает использовать архитектурный подход к управлению инфраструктурными ресурсами, а именно автоматизировать создание и управление инфраструктурными компонентами с помощью программного кода.

Основная идея заключается в том, что вся инфраструктура, такая как виртуальные машины, сетевые настройки, хранилища данных и другие ресурсы, описываются в виде кода, который может быть управляемы исходными файлами, хранящимся в системе контроля версий.

CI/CD

Методология разработки программного обеспечения целью которой является автоматизация и ускорение процесса разработки, тестирования и доставки программного продукта.

Цель CI/CD состоит в том, чтобы обеспечить быстрое время от идеи до развёртывания новых функций или исправлений ошибок, с учётом меньшей вероятности ошибок и конфликтов, а также поддержания высокого уровня качества кода и процесса разработки.

Состав

- Непрерывная интеграция

- Непрерывная доставка

- непрерывное развёртывание

Преимущества внедрения DevOps в разработку

Ускорение процесса разработки

DevOps позволяет автоматизировать и упростить различные этапы разработки, такие как сборка, тестирование и развёртывание. Это позволяет сократить время, затрачиваемое на каждый из этих этапов, и значительно увеличить производительность команды разработки.

Улучшение качества ПО

DevOps способствует автоматизации тестирования и контроля качества, что позволяет быстрее выявлять и устранять ошибки и повышать стабильность и надёжность разрабатываемого продукта.

Снижение рисков

DevOps позволяет осуществлять постоянный мониторинг и контроль процесса разработки Это позволяет оперативно реагировать на возникающие проблемы и снижает риски, связанные с качеством, безопасностью.

Более быстрая доставка нового функционала

DevOps способствует автоматизации процесса развёртывания и доставки нового функционала пользователям. Благодаря этому, команда разработки может выпускать обновления и новые версии приложения значительно быстрее и с меньшими затратами.

Более гибкий и масштабируемый процесс разработки

DevOps позволяет легко внедрять изменения и масштабировать процесс разработки в соответствии с потребностями пользователей или рынка. Это позволяет более гибко реагировать на изменения условий и быстро адаптировать к новым требованиям.

Снижение затрат на разработку и эксплуатацию

DevOps позволяет оптимизировать процессы разработки и эксплуатацию что сокращает затраты на труд и ресурсы. Это позволяет сделать разработку ПО более эффективной и экономичной.

Более высокая удовлетворённость пользователей

Благодаря улучшенному качеству ПОЮ более быстрой доставке нового функционала и более оперативной поддержке пользователи получают более удовлетворительный опыт использования приложения. Это повышает удовлетворённость пользователей и способствует удержанию и привлечению новых клиентов.

Система контроля версий Git. Работа с GitLab и GitHub часть 1

Что такое система контроля версий и для чего она нужна

Система контроля версий - это специальное программное обеспечение, которое используется для управления изменениями в файловой системе, отслеживания и контроля версий документов или кода программы. Она позволяет разработчикам и команде проекта работать над файлами вместе, отслеживать изменения, управлять их версиями и восстанавливаться к предыдущим состояниям проекта.

Преимущества применения системы контроля версий

1. История изменений

2. Контроль версий

3. Коллективная работа

4. Ветвление и слияние

5. Отслеживание ошибок

Распределенная VS Централизованная системы контроля версий

Архитектура: Централизованная система контроля версий имеет единую центральную базу данных, где хранятся все файлы и изменения. В распределенной системе каждый пользователь имеет собственную копию репозитория, включая всю историю изменений.

Работа в офлайн-режиме: Распределенная система позволяет работать в офлайн-режиме, так как каждый пользователь имеет полную копию репозитория. В централизованной системе требуется подключение к центральному серверу для выполнения большинства операций.

Распределенная VS Централизованная системы контроля версий

Управление правами доступа: В централизованной системе управление правами доступа к репозиторию обычно происходит на уровне центрального сервера. В распределенной системе каждый пользователь может выполнять операции с собственной копией репозитория, что позволяет более гибко управлять правами доступа.

Скорость работы: Распределенная система обычно быстрее выполняет операции, так как все данные хранятся локально у каждого пользователя. В централизованной системе операции происходят через удаленное соединение, что может быть медленнее при большом количестве пользователей.

snapshot-based

Система контроля версий с механизмом снимков (например, Git, Mercurial) хранят все версии файлов и директорий в форме полных копий (снимков) состояния проекта в определенный момент времени. Когда вы делаете коммит, система фиксирует все измененные файлы и создает новый снимок, который включает в себя полные копии всех измененных файлов. Это позволяет быстро переключаться между разными версиями проекта и осуществлять обход старых коммитов.

delta-based

Система контроля версий со списком изменений (например, Subversion) хранят только разницу (delta) между последующими версиями файлов. Когда вы делаете коммит, система сохраняет только изменения, сделанные в файлах, относительно предыдущего коммита, в виде списка изменений. На самом деле, система часто хранит несколько версий файла, чтобы быстрее обрабатывать запросы, основанные на списке изменений. При переключении на предыдущие версии проекта, СКВ применяет Bсe изменения последовательно, чтобы восстановить запрошенную версию проекта.

Коммит

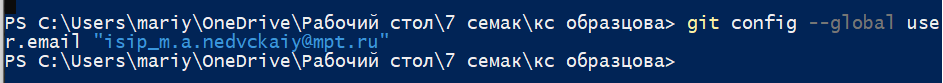
Коммит (commit) представляет собой операцию, при которой изменения в файловой системе, внесенные программистом, сохраняются в репозитории проекта. Коммит фиксирует изменения в файле или наборе файлов, а также добавляет комментарии или описания к этим изменениям.

Каждый коммит имеет уникальный идентификатор, который позволяет отслеживать историю изменений и восстанавливать предыдущие версии файлов. Коммиты также позволяют работать с несколькими программистами над одним проектом, объединяя изменения, внесенные каждым из них.

1. Продемонстрировать конфигурацию Git

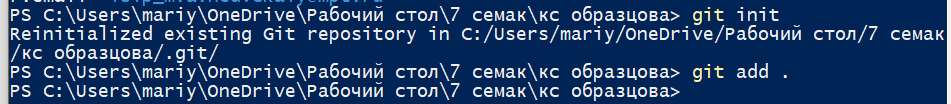
Запускаем PowerShell в нашей папке. Инициализируем свои данные в гите командами git config –global user.name/email





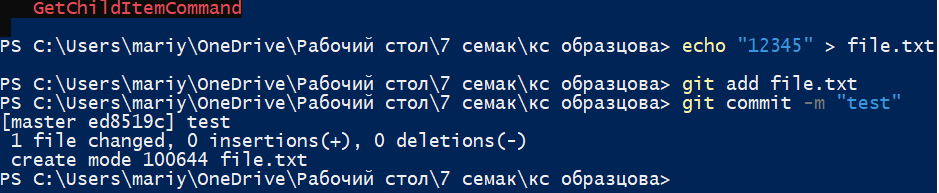
1. Создание локального репозитория

Создаём гит репозиторий в нашей папке командой get init. Чтобы сохранить файлы из рабочего каталога в индексе, перенесём сначала все файлы командой git add .



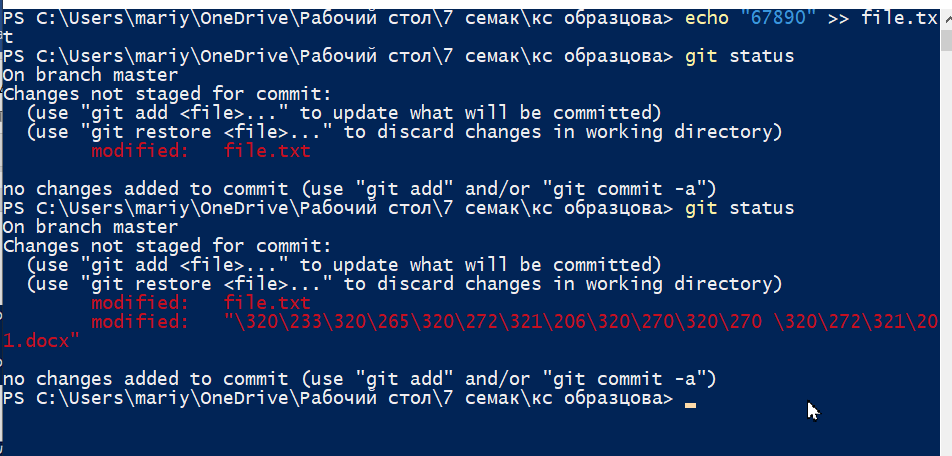
Используем git commit, чтобы создать копию индекса, те коммит 

Создадим ещё один файл и перенесём его в индекс, после создания второго коммита он будет содержать новую копию

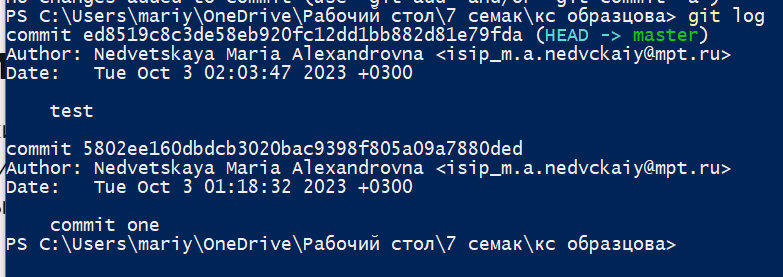


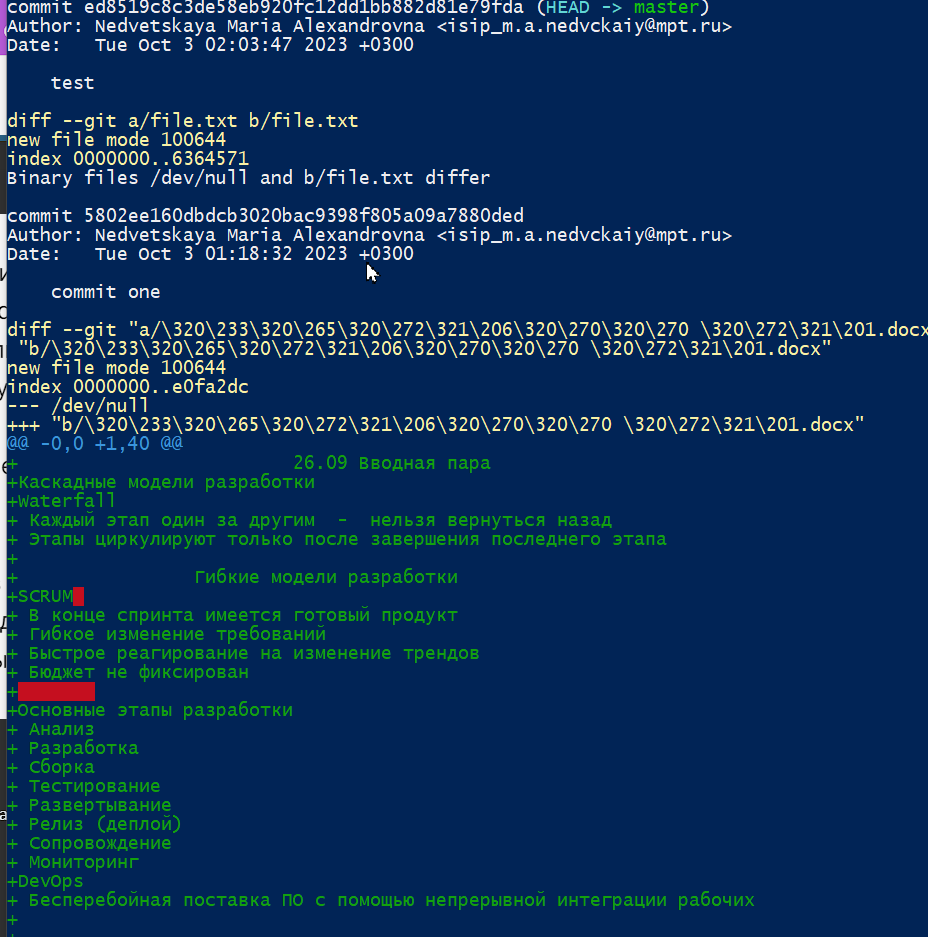
1. Продемонстрировать работу основных команд для работы с файлами в репозитории

Изменим содержание файла и пропишем команду git status для просмотра изменений и отслеживания всех файлов в индекче и рабочем каталоге

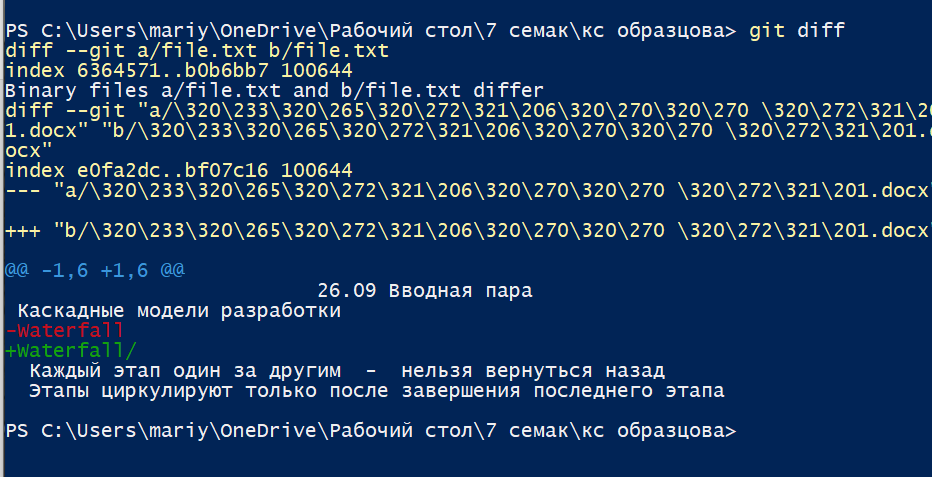


Чтобы просмотреть историю репозитория с авторами, датой, временем и идентификатором, пропишем команду git log

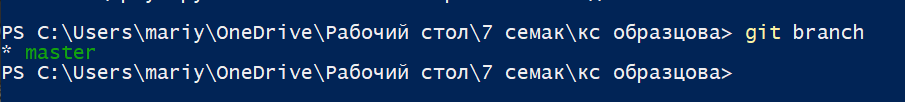




Командой git diff можем увидеть разницу рабочего каталога и коммита

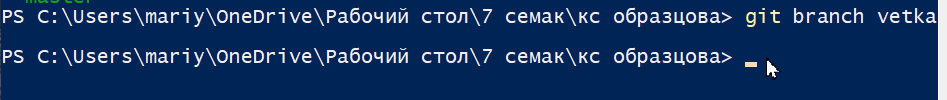


Для вывода всех веток используем git branch

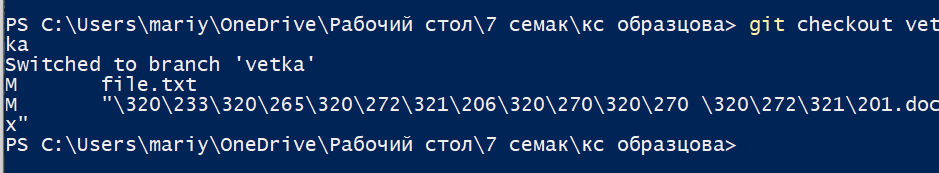


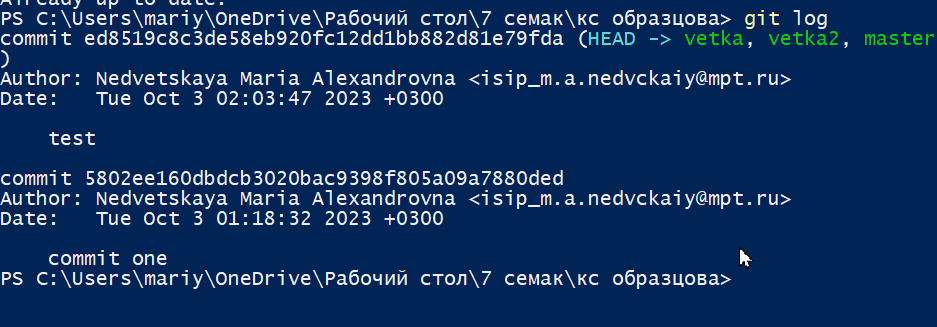
1. Создание нескольких различных веток

Для создания ветки используем команду выше но уже с названием ветки



Для переключения между ветками используем git checkout





1. Описать работу с указателями HEAD

HEAD - это условное имя текущего коммита, указывающее обычно на имя ветки/хэш. Если пропишем git log, стрелочка HEAD укажет значение нашей ветки, которая сама по себе будет содержать коммиты. При переключении между ветками стрелочка будет указывать на выбранную ветку и соответственно её коммиты.

Можно использовать checkout и хэш коммитов для отслеживания изменений на определённой копии. В таком состоянии лог HEAD будет указывать на тот коммит, к которому мы перешли

Вывод: В ходе работы написали конспект лекции, продемонстрировали конфигурацию Git, создали локальный репозиторий, где показали работу основных команд для работы с файлами и описали работу с указателями HEAD.

# Практическая работа №2. Работа Git-хостингами

Цель работы: Написать конспект лекции, описать создание аккаунта на любом Git-хостинге и общий функционал вкладок, продемонстрировать основные команды для работы с удаленным репозиторием, сделать ограничение на определенную ветку и попробовать залить сразу несколько веток

Ход работы:

1. Конспект лекции

Работа с GitLab и GitHub

Система контроля версий – это специальной программное обеспечение, которое используется для управления изменениями в файловой системе, отслеживания и контроля версий документов или кода программы. Она позволяет разработчикам и команде проекта работать над файлами вместе, отслеживать изменения, управлять их версиями и восстанавливаться к предыдущим состояниям проекта.

Преимущества применения системы контроля версий

1. История изменений

2. Контроль версий

3. Коллективная работа

4. Ветвление и слияние

5. Отслеживание ошибок

Распределенная VS Централизованная системы контроля версий

Архитектура: Централизованная система контроля версий имеет единую центральную базу данных, где хранятся все файлы и изменения. В распределенной системе каждый пользователь имеет собственную копию репозитория, включая всю историю изменений.

Работа в офлайн-режиме: Распределенная система позволяет работать в офлайн-режиме, так как каждый пользователь имеет полную копию репозитория. В централизованной системе требуется подключение к центральному серверу для выполнения большинства операций.

Управление правами доступа: В централизованной системе управление правами доступа к репозиторию обычно происходит на уровне центрального сервера. В распределенной системе каждый пользователь может выполнять операции с собственной копией репозитория, что позволяет более гибко управлять правами доступа.

Скорость работы: Распределенная система обычно быстрее выполняет операции, так как все данные хранятся локально у каждого пользователя. В централизованной системе операции происходят через удаленное соединение, что может быть медленнее при большом количестве пользователей.

Snapshot-based

Система контроля версий с механизмом снимков (например, Git, Mercurial) хранят все версии файлов и директорий в форме полных копий (снимков) состояния проекта в определенный момент времени. Когда вы делаете коммит, система фиксирует все измененные файлы и создает новый снимок, который включает в себя полные копии всех измененных файлов. Это позволяет быстро переключаться между разными версиями проекта и осуществлять обход старых коммитов.

Delta-based

Система контроля версий со списком изменений (например, Subversion) хранят только разницу (delta) между последующими версиями файлов. Когда вы делаете коммит, система сохраняет только изменения, сделанные в файлах, относительно предыдущего коммита, в виде списка изменений. На самом деле, система часто хранит несколько версий файла, чтобы быстрее обрабатывать запросы, основанные на списке изменений. При переключении на предыдущие версии проекта, СКВ применяет все изменения последовательно, чтобы восстановить запрошенную версию проекта.

Коммит

Коммит (commit) представляет собой операцию, при которой изменения в файловой системе, внесенные программистом, сохраняются в репозитории проекта. Коммит фиксирует изменения в файле или наборе файлов, а также добавляет комментарии или описания к этим изменениям.

Каждый коммит имеет уникальный идентификатор, который позволяет отслеживать историю изменений и восстанавливать предыдущие версии файлов. Коммиты также позволяют работать с несколькими программистами над одним проектом, объединяя изменения, внесенные каждым из них.

Удаленные Git-хостинги

Git-хостинги - это платформы, которые предоставляют инфраструктуру для хранения и управления репозиториями Git. Они позволяют разработчикам работать с Git репозиториями удаленно, управлять версиями, отслеживать изменения и сотрудничать с другими разработчиками.

GitLab - это хостинг-платформа для хранения и управления кодом. Она предоставляет функциональность, аналогичную GitHub, но с открытым исходным кодом. GitLab может быть развернут на собственных серверах, что позволяет организациям иметь полный контроль над своими репозиториями. Он также предлагает широкий набор инструментов для управления проектами.

Работа с удаленным репозиторием

git remote add <name> <address> - добавление нового репозитория. Обращение через название

git remote -v - просмотр списка добавленных удаленных репозиториев

git remote remove <name> - удаление удаленного репозитория

Отправка локального репозитория на удаленный

git push <name> <branch> - отправка локального репозитория на удаленный

git push --all <name> - отправка всех локальных веток на удаленный репозиторий

git push -f <name> <branch> - отправка локального репозитория на удаленный с переписыванием всех коммитов на сервере в этой ветке

Клонирование репозитория

git clone <url> - клонирует репозиторий с удаленного на локальный. Отображаемая версия - основная ветка. Все ветки также скачиваются.

git clone <url> -b <branch> - клонирует репозиторий с удаленного на локальный. Отображаемая версия - указанная ветка. Все ветки также скачиваются.

git clone <url> --single-branch -b <branch> - клонирует репозиторий с удаленного на локальный. Отображаемая версия - указанная ветка. Скачивает только указанную ветку.

1. Описать создание аккаунта на любом Git-хостинге. Кратко описать функционал различных вкладок.

Откроем официальный сайт и заполним данные ФИО, юзернейма, почты и пароль

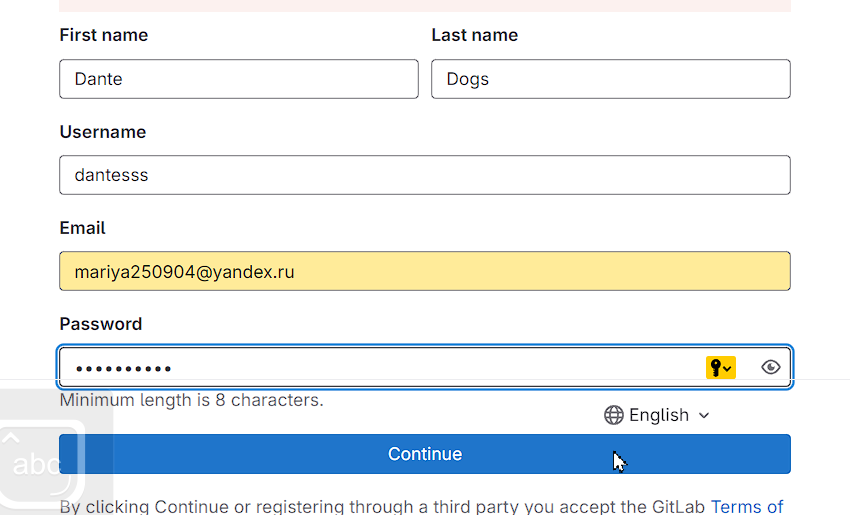


Рисунок – Ввод данных

Необходимо ввести код с указанной почты

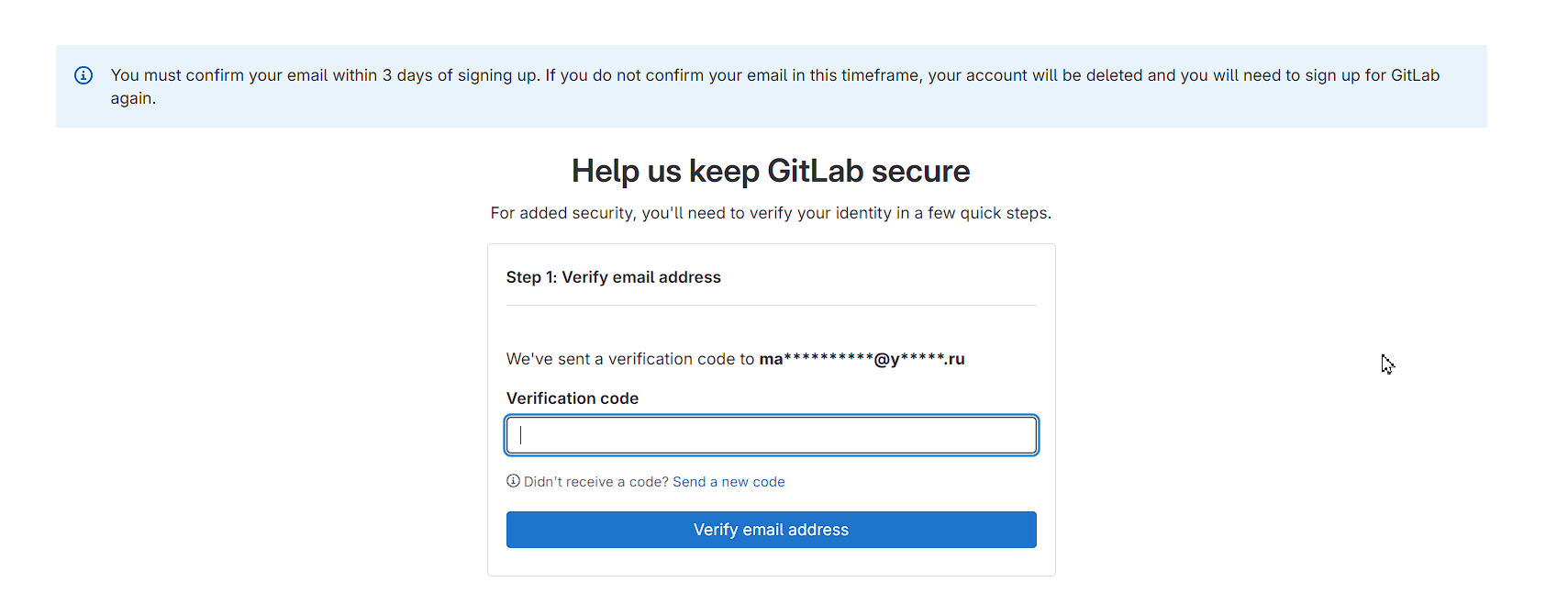


Рисунок – Поле ввода кода

Открываем письмо на почте и вводим код

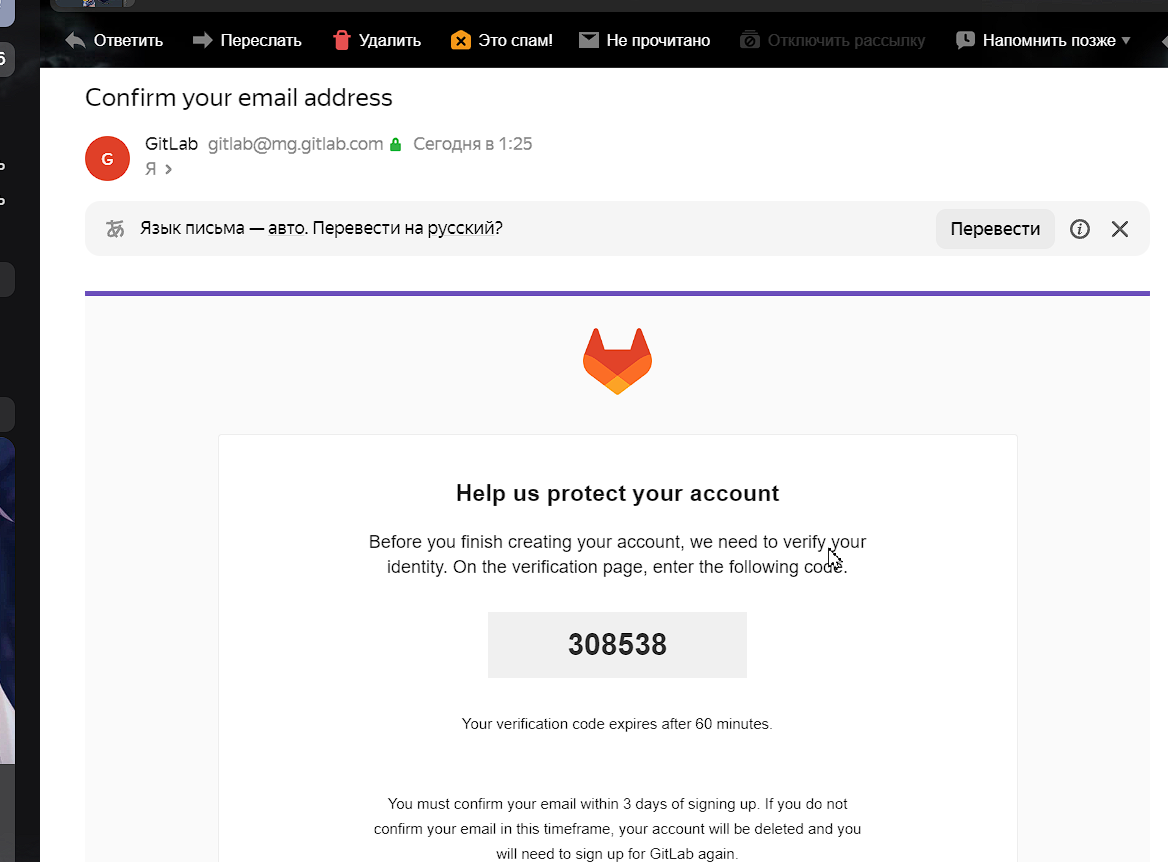


Рисунок – Письмо на почте

После подтверждения почты нажимаем Next

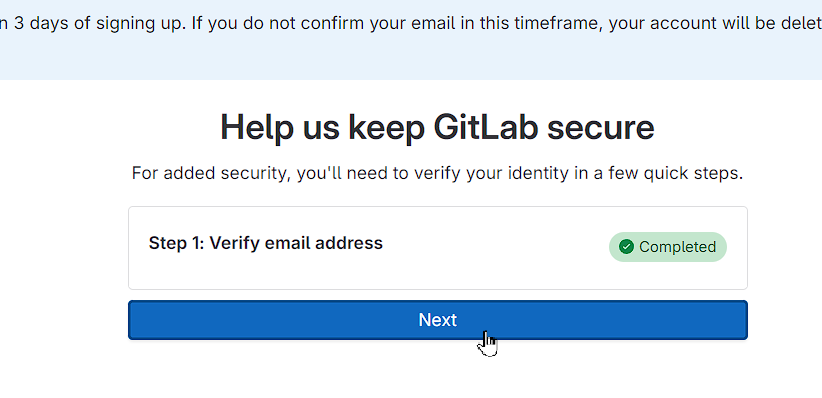


Рисунок – Следующий шаг

Выбираем роль и цели использования лаба

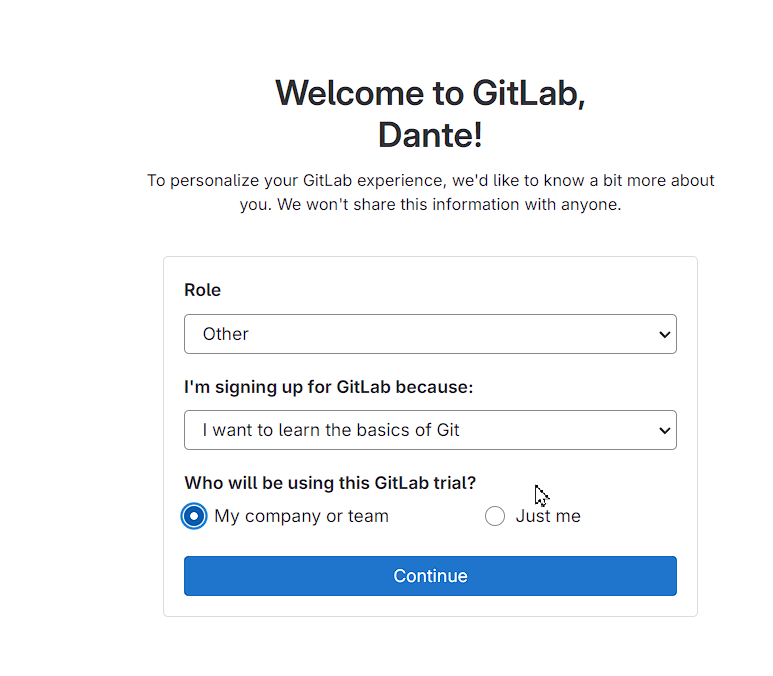


Рисунок – Использование лаба

Так как по ошибке была выбрана компания, заполняем название, количество пользователей и страну

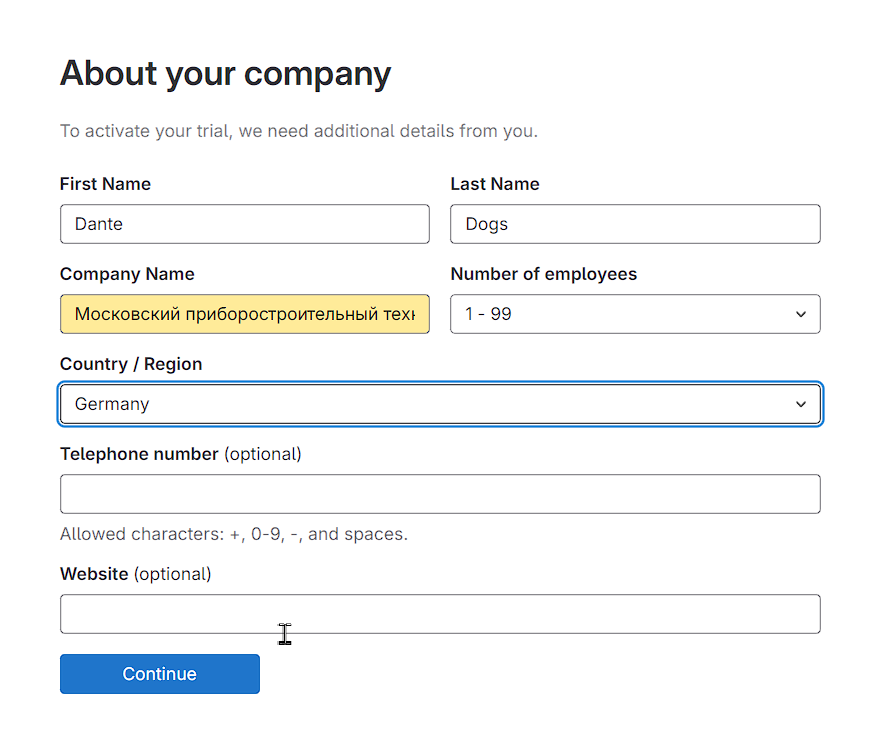


Рисунок – Данные компании

Создаём первый проект

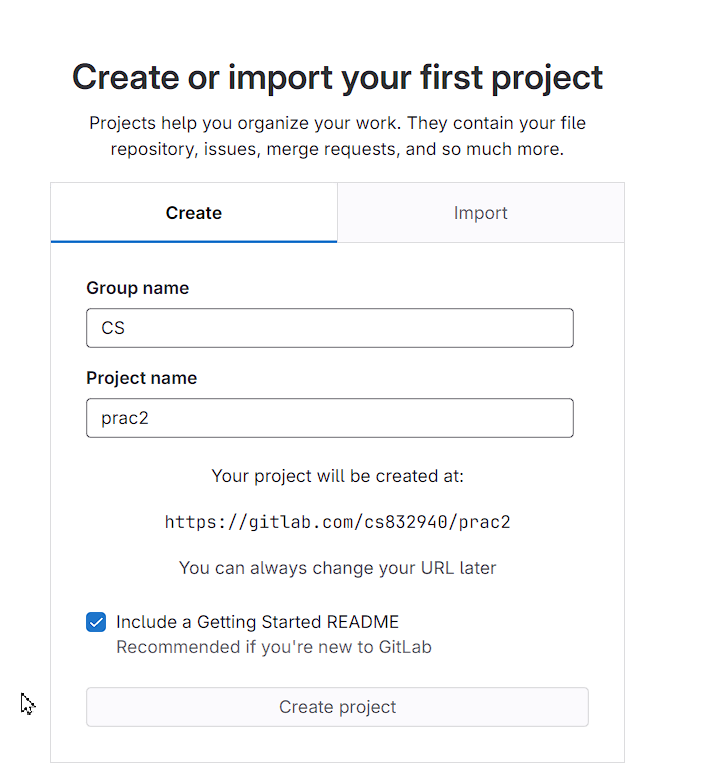


Рисунок – Создание первого проекта

В приветственном сообщении нажимаем начать работу

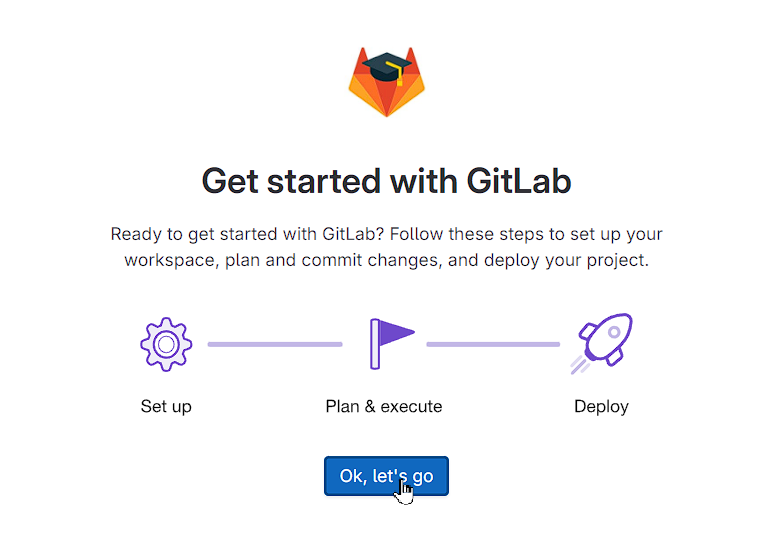


Рисунок – Приветственное сообщение

Во вкладке Projects отображаются проекты

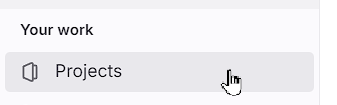


Рисунок – Проекты

Во вкладке Group будут отображаться сформированные группы

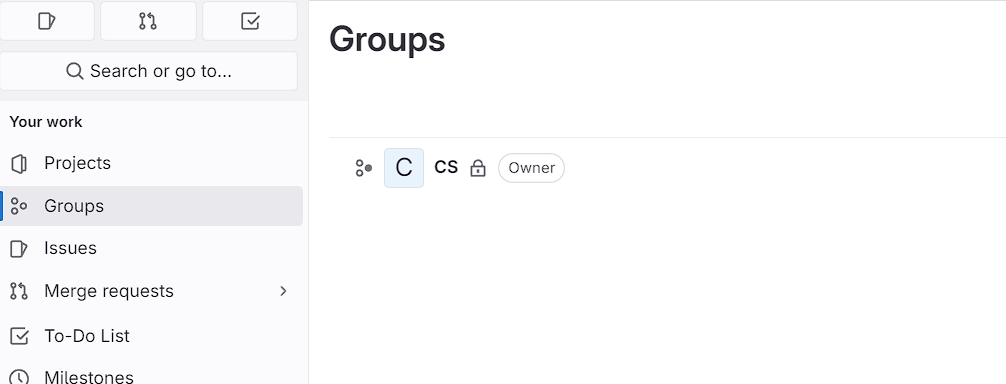


Рисунок – Группы

Во вкладке issues можно просмотреть проблемы с фильтрами

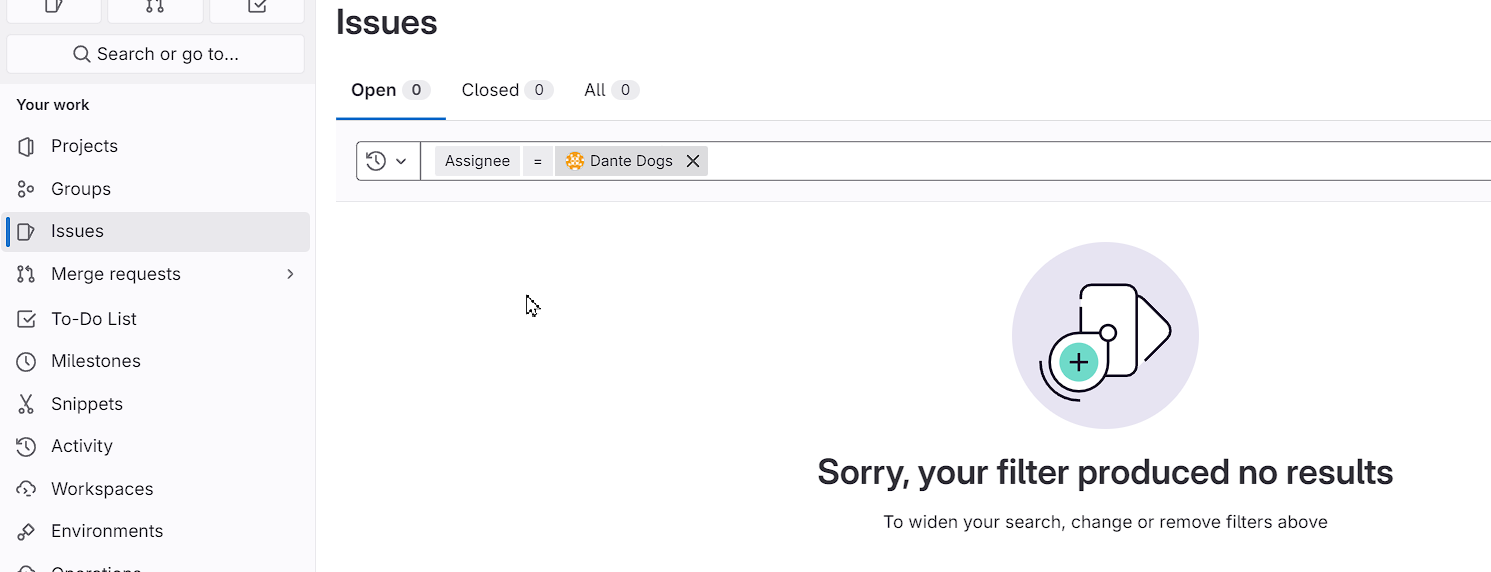


Рисунок – Проблемы

Во вкладке Merge будут запросы за слияние проектов

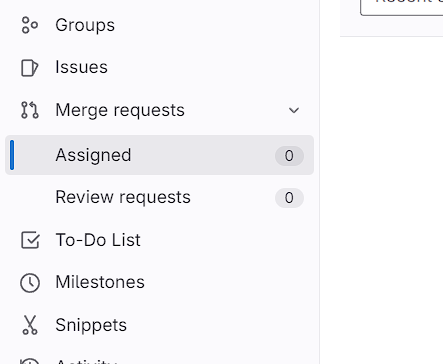


Рисунок – Запросы

Во вкладке To-DO-List можно отобразить список задач

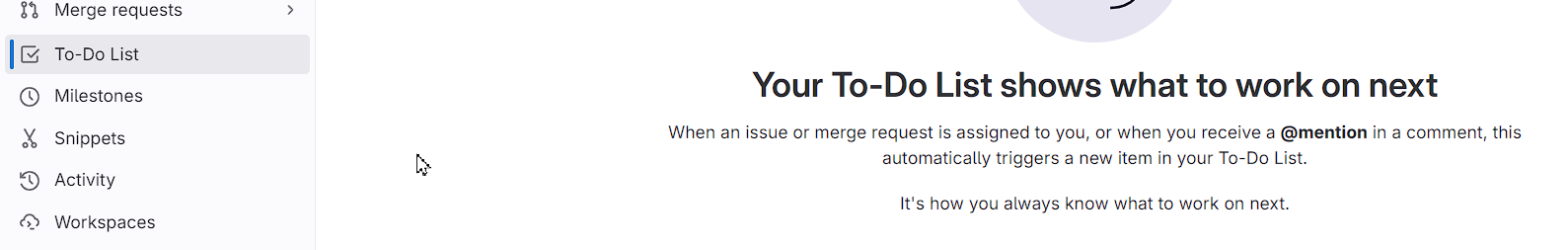


Рисунок – Список задач

Во вкладке Milestones можно сгруппировать проблемы и запросы

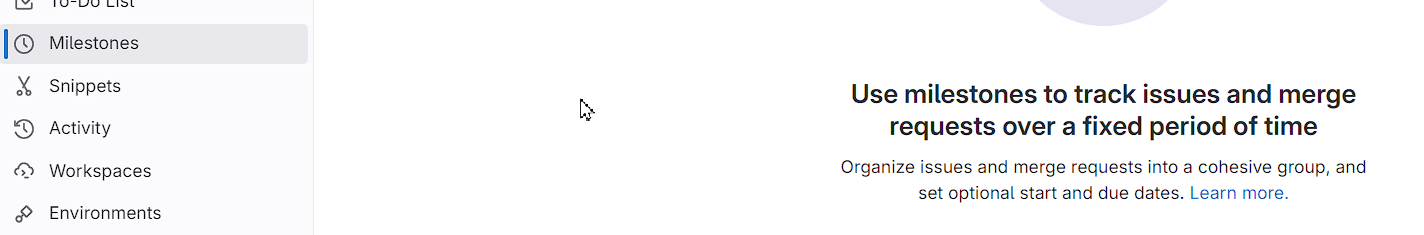


Рисунок – Группировка

Во вкладке Snippets можно вставлять фрагменты дока, текста и документацию



Рисунок – Вставка

Во вкладке Activity отображается активность пользователей в разных вкладках

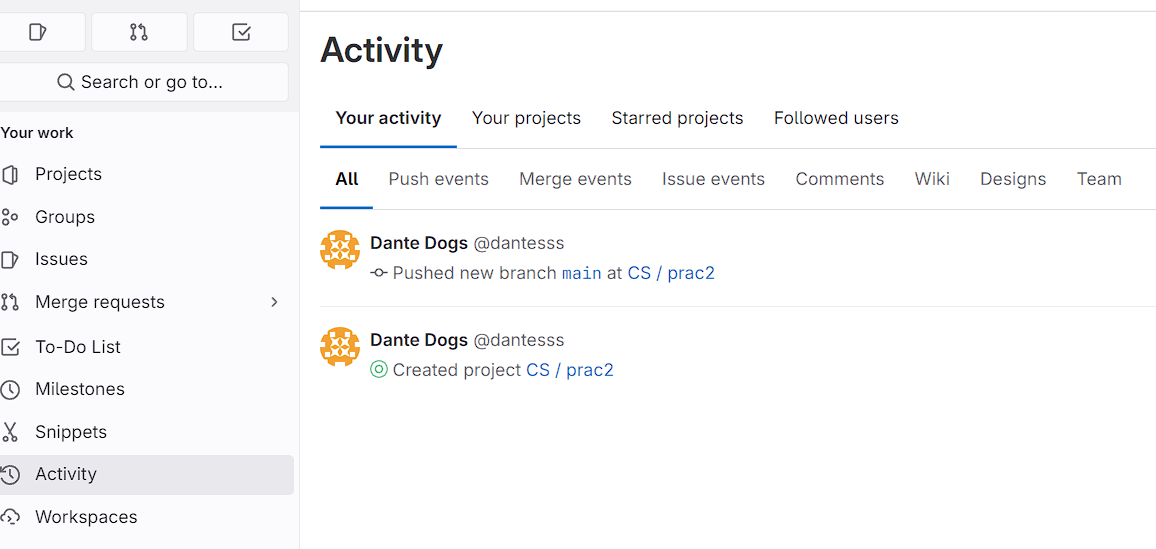


Рисунок – Активность

Во вкладке Workspaces можно организовать рабочее пространство для удобства разработки

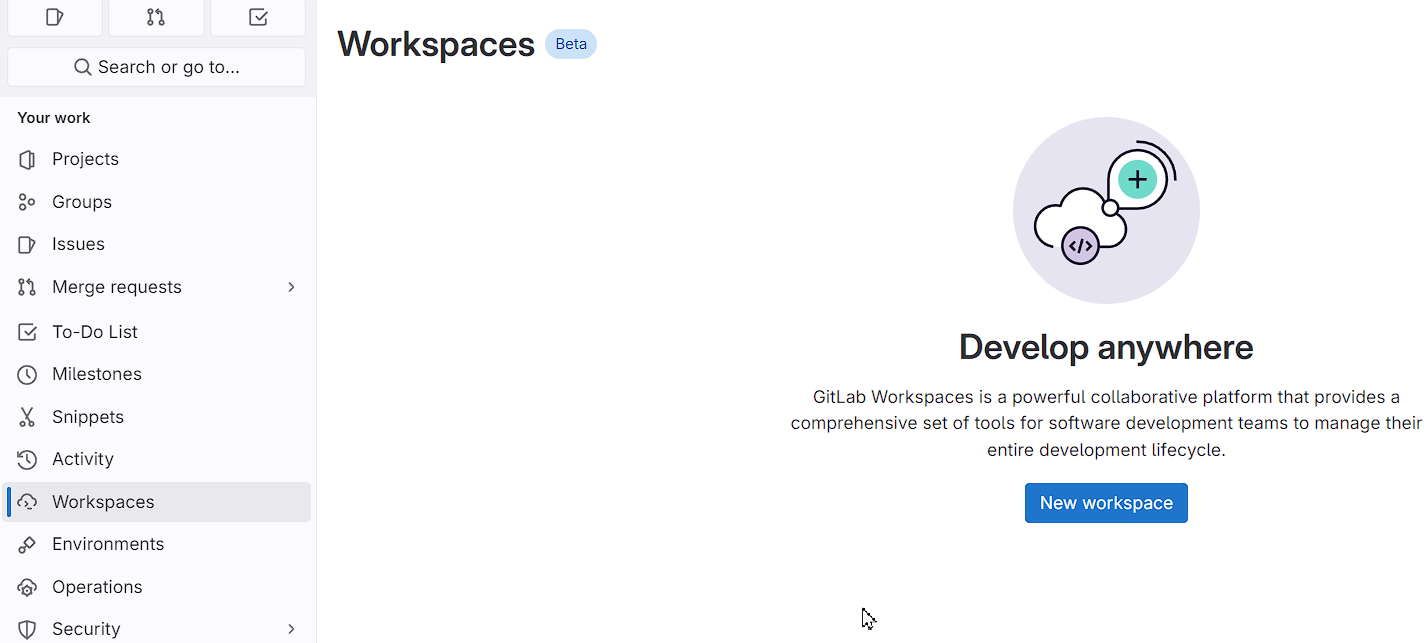


Рисунок – Рабочее пространство

Во вкладке Environments можно отслеживать состояние сред каждого проекта

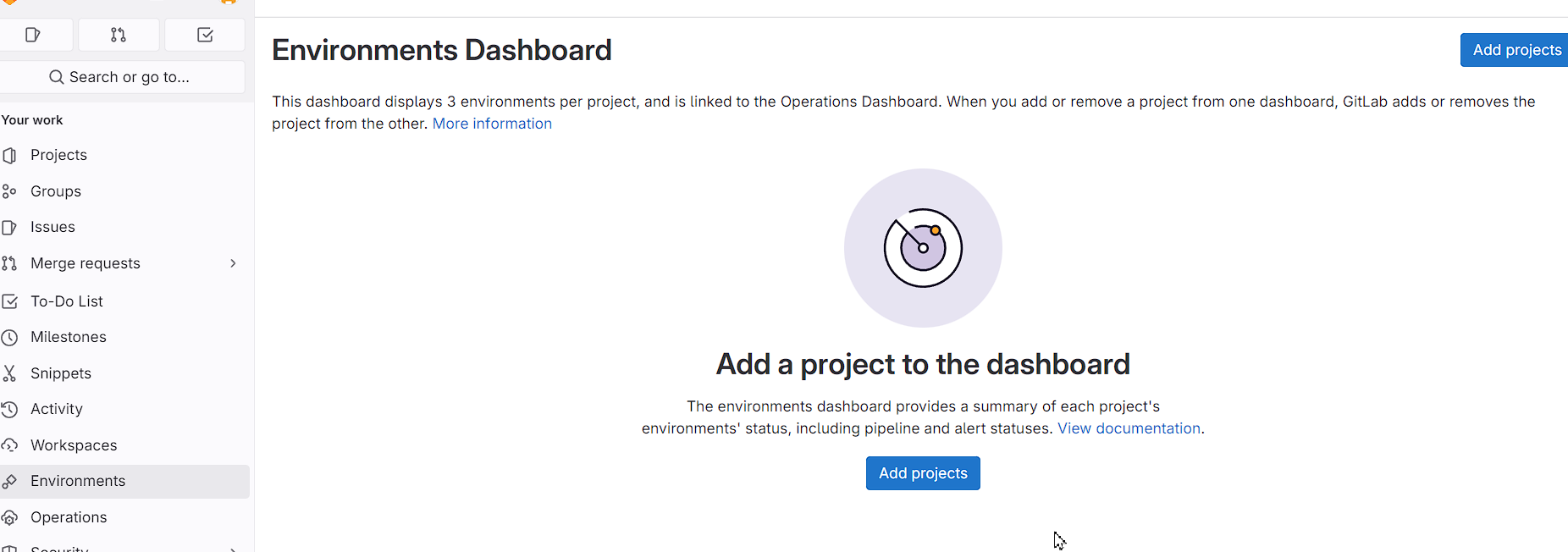


Рисунок – Состояние

1. Работа с удаленным репозиторием. Основные команды для работы с удаленным репозиторием ( + обязательно сделать ограничение на определенную ветку и попробовать залить сразу несколько веток, описать результат)



