GIT

GIT & GITHUB

Что такое GIT

Git — это распределенная система контроля версий, разработанная для управления изменениями в исходном коде и совместной работы над проектами.

Git решает несколько ключевых проблем, связанных с управлением исходным кодом:

Отслеживание изменений:

- Сохраняет историю всех изменений в проекте.
- Позволяет вернуться к любой предыдущей версии проекта.

Совместная работа:

- Обеспечивает удобную работу в команде.
- Позволяет нескольким разработчикам работать над одним проектом одновременно, не мешая друг другу.

Ветвление и слияние:

- Поддерживает создание отдельных веток для разработки новых функций или исправлений.
- Облегчает слияние веток с основной версией проекта.

Резервное копирование:

- Хранит копию проекта на каждом компьютере разработчика.
- Обеспечивает безопасность данных и возможность восстановления проекта.

Работа в оффлайн-режиме:

- Разработчики могут работать с репозиторием локально без подключения к сети.
- Синхронизация изменений происходит при подключении к удалённому репозиторию.

Как скачать GIT

ССЫЛКА НА GIT - ТЫК

ССЫЛКА НА ОФ.ИНСТРУКЦИЮ - ТЫК

GITHub аккаунт

GitHub — это платформа для хостинга репозиториев, созданных с использованием системы контроля версий Git.

Хостинг репозиториев: GitHub позволяет хранить и управлять кодом проекта в репозиториях. Репозиторий может содержать все файлы проекта и историю их изменений.

Сотрудничество: GitHub облегчает работу над проектами в команде. Разработчики могут вносить изменения, предлагать исправления и новые функции через систему pull request.

Контроль версий: С GitHub легко отслеживать изменения в коде, возвращаться к предыдущим версиям и управлять ветками разработки.

Процесс управления проектом: GitHub предлагает инструменты для управления проектами, такие как доски задач (issues), проекты и вики-страницы для документации.

Публикация кода: Разработчики могут делиться своими проектами с сообществом, делая репозитории публичными, или работать над закрытыми проектами в приватных репозиториях.

Интеграция с другими сервисами: GitHub интегрируется с множеством других инструментов и сервисов, таких как CI/CD системы, менеджеры задач, средства мониторинга и многое другое.

Альтернативы GITHUB:

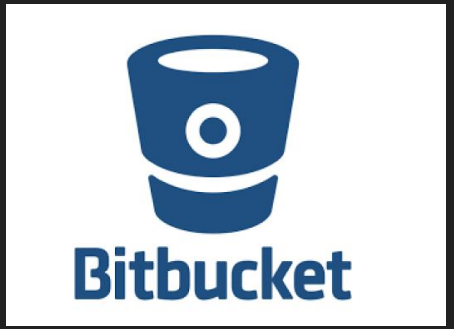
- GitLab - ТЫК

  Более официальный

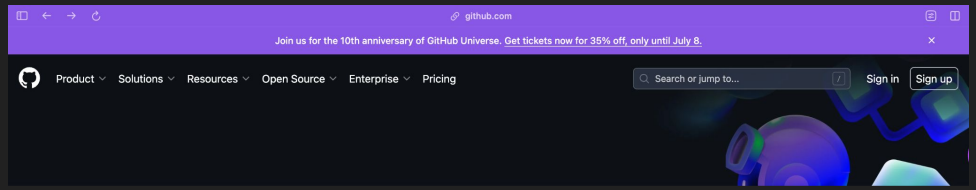
- BitBucket - ТЫК

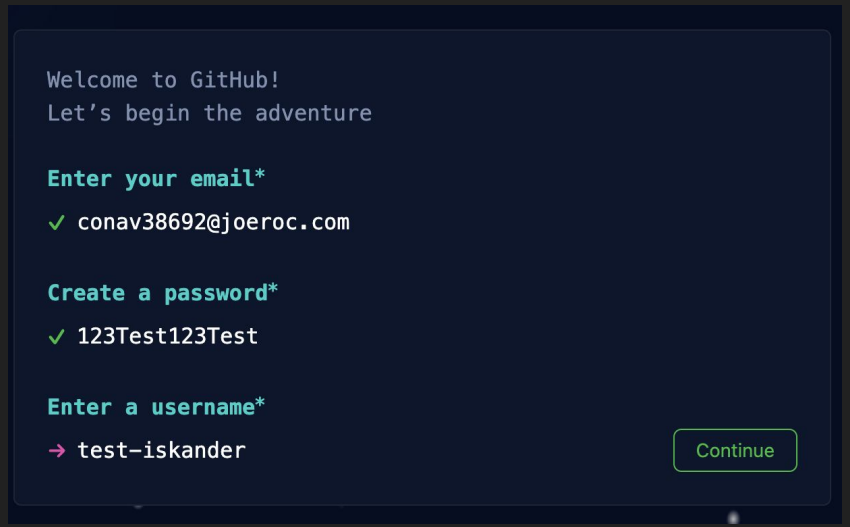
  Легкая интеграция с Atlassian продуктами

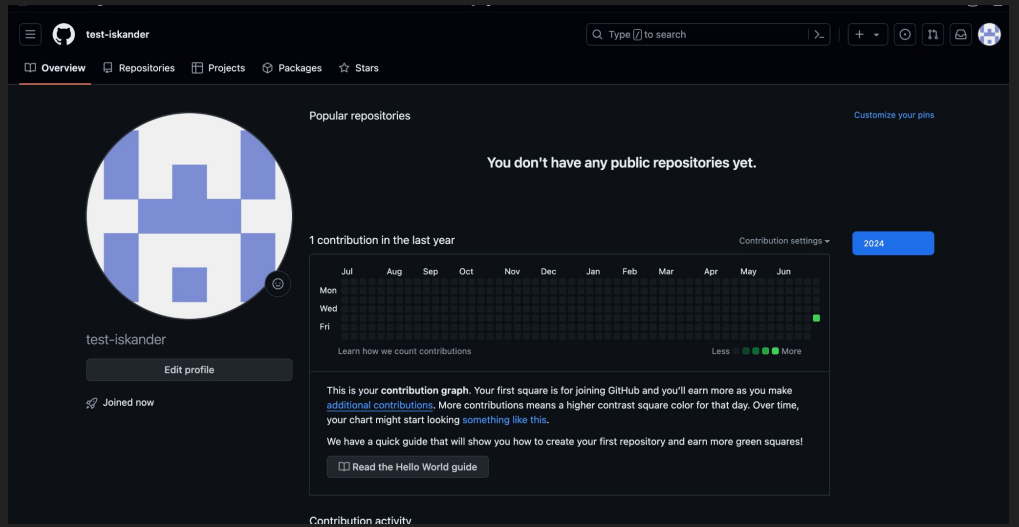
- AWS CodeCommit - ТЫК



Регистрация на GITHUB







Оформление github аккаунта

Нажмите + в правом верхнем углу и выберите New Repository.

Откроется страница создания нового репозитория. В названии укажите имя пользователя из вашего профиля. После того как вы введете имя пользователя, будет показана информация о том, что вы создаете особый репозиторий.

Под описанием репозитория не забудьте установить отметку Public, чтобы файл README.md был виден всем.

Обязательно установите отметку Add a README file. Это создаст файл README.md, в котором мы и будем работать.

Далее нажимаем кнопку Create repository. Репозиторий успешно создан. Зайдите в только что созданный репозиторий и увидите, что в нем уже есть файл README.md.

В следующих разделах мы добавим информацию в наш файл README.md. Мы будет работать с файлом через интерфейс GitHub, но вы можете использовать любой другой текстовый редактор.

Что такое репозиторий

Репозиторий — это хранилище, где хранятся файлы проекта и история их изменений. В контексте Git и GitHub репозиторий выполняет несколько ключевых функций:

- Хранение файлов проекта: Репозиторий содержит все файлы, необходимые для работы над проектом, включая исходный код, конфигурационные файлы, документацию и другие ресурсы.
- История изменений: В репозитории хранится полная история изменений всех файлов. Это позволяет отслеживать, кто и когда внёс изменения, а также восстанавливать предыдущие версии файлов.
- Ветвление и слияние: Репозиторий поддерживает работу с ветками (branches), что позволяет разработчикам работать над новыми функциями или исправлениями ошибок независимо от основной версии кода. После завершения работы изменения можно слить (merge) с основной веткой.
- Совместная работа: Репозиторий предоставляет инструменты для совместной работы, такие как pull requests, позволяя другим участникам команды предлагать изменения и обсуждать их до включения в основной код.
- Резервное копирование: Репозиторий служит в качестве резервной копии кода проекта, защищая его от потери данных.

Локальный репозиторий: Это репозиторий, который хранится на вашем компьютере и используется для локальной разработки и тестирования.

Удалённый репозиторий: Это репозиторий, который размещён на удалённом сервере, таком как GitHub, и доступен для команды разработчиков.

SSH ключ для github аккаунта

Применение SSH ключей

Аутентификация на сервере: SSH ключи позволяют безопасно подключаться к удалённым серверам без использования паролей. При настройке доступа к серверу, публичный ключ добавляется в файл authorized_keys на сервере. Когда пользователь пытается подключиться, сервер проверяет, что у пользователя есть соответствующий приватный ключ.

Защита данных: SSH ключи обеспечивают высокий уровень безопасности, так как для доступа к серверу требуется уникальный приватный ключ. Это снижает риск взлома по сравнению с использованием паролей.

Работа с репозиториями: При использовании GitHub или других систем контроля версий, SSH ключи позволяют безопасно взаимодействовать с удаленными репозиториями. Публичный ключ добавляется в настройки GitHub аккаунта, что позволяет клонировать, пушить и пулить репозитории без необходимости ввода пароля.

Что такое SSH

SSH (Secure Shell) ключ — это пара криптографических ключей, используемых для безопасного подключения к удалённым серверам и аутентификации пользователя. Пара ключей состоит из приватного и публичного ключа:

Приватный ключ: Хранится у пользователя и никогда не передается другим. Этот ключ используется для расшифровки данных, зашифрованных публичным ключом, и для подписи данных, чтобы подтвердить их происхождение.

Публичный ключ: Этот ключ можно свободно передавать и публиковать. Он используется для шифрования данных, которые может расшифровать только соответствующий приватный ключ.

Как настроить SSH

Установите Git for Windows

Создайте SSH ключ:
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

Добавьте SSH ключ в SSH-agent:
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa

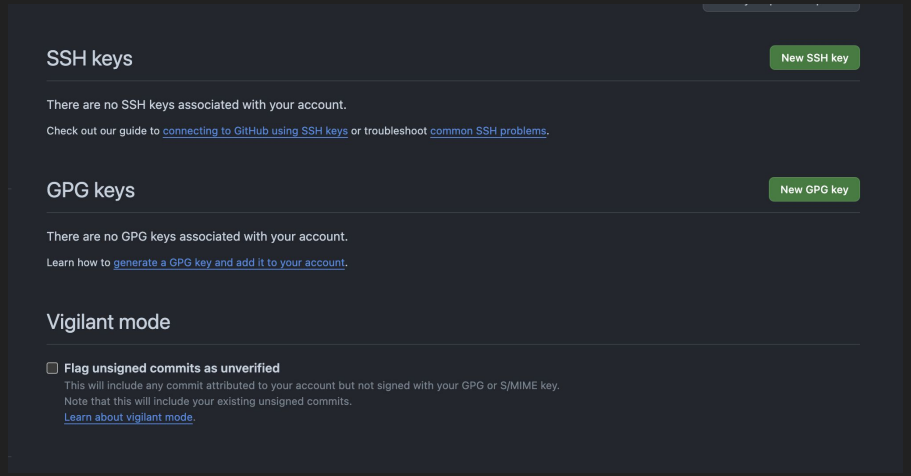
Добавьте SSH ключ на GitHub:
cat ~/.ssh/id_rsa.pub

Добавление public key в github аккаунт

Выделите и скопируйте весь текст публичного ключа. Затем зайдите на GitHub и выполните следующие шаги:

1. Перейдите в настройки аккаунта.
2. Выберите раздел "SSH and GPG keys".
3. Нажмите кнопку "New SSH key".
4. Вставьте скопированный ключ в поле "Key" и дайте ему название.
5. Нажмите "Add SSH key".

Теперь ваш SSH ключ добавлен на GitHub, и вы можете использовать его для аутентификации при клонировании, пуше и пуле репозиториев.



Основные git команды

- git init
- git pull
- git push
- git fetch
- git add
- git commit
- git checkout
- git branch
- git merge
- git clone

git init

Описание: Инициализирует новый пустой репозиторий Git или повторно инициализирует существующий репозиторий.

Эта команда создаёт новый подкаталог .git в вашем проекте, содержащий все необходимые файлы репозитория. После этого папка становится репозиторием Git.

git clone

git clone: Создаёт копию удалённого репозитория на вашем компьютере, включая все его файлы, историю коммитов, ветки и теги. Это наиболее часто используемая команда для начала работы с удалённым репозиторием.

Что происходит при клонировании репозитория

Создание нового каталога: git clone создаёт новый каталог с именем репозитория (если не указано другое имя) в текущем рабочем каталоге.

Копирование содержимого: Содержимое удалённого репозитория, включая все файлы и историю коммитов, копируется в этот новый каталог.

Настройка удалённого репозитория: Удалённый репозиторий, откуда был выполнен клон, автоматически добавляется под именем origin.

Переключение на основную ветку: После клонирования git clone автоматически переключается на основную ветку (обычно main или master) клонированного репозитория.

git pull

Описание: Скачивает содержимое с удаленного репозитория и сливает его с локальной веткой.

Эта команда скачает изменения с удаленного репозитория origin и объединит их с локальной веткой main.

git pull origin main

git push

Описание: Отправляет изменения из локального репозитория в удалённый репозиторий.

Эта команда отправляет коммиты из локальной ветки main в удалённый репозиторий origin.

git push origin main

git fetch

Описание: Скачивает изменения из удалённого репозитория, но не объединяет их с локальной веткой. Изменения просто загружаются в локальный репозиторий.

Эта команда скачает все новые изменения из удаленного репозитория origin.

git fetch origin

git add

Описание: Добавляет изменения в индекс (staging area), подготавливая их для следующего коммита.

Эта команда добавляет все изменения в текущем каталоге и подкаталогах в индекс.

git add .

git commit

Описание: Сохраняет изменения, добавленные в индекс, в репозиторий. Коммиты фиксируют состояние проекта на момент выполнения команды.

Эта команда создаёт новый коммит с сообщением "Add new feature".

git commit -m "Add new feature"

git checkout

Описание: Переключается между ветками или восстанавливает файлы рабочей копии.

Эта команда переключается на ветку main.

git checkout main

git checkout -b “название ветки”

Описание: Переключается между ветками и создает новую ветку.

Эта команда создает новую ветку на основе той в которой мы находимся.

git checkout -b test

git branch

Описание: Управляет ветками. Может создавать, перечислять или удалять ветки.

Эта команда создаёт новую ветку new-feature.

git branch new-feature

git merge

Описание: Объединяет изменения из одной ветки в другую.

Эта команда объединяет изменения из ветки new-feature в текущую ветку.

git merge new-feature

Процесс работы с git-ом

1. создаем связь с git-ом - git init
2. git clone <ссылка на репозиторий> клонируем удаленный репозиторий
3. git pull - получаем актуальную версию
4. git checkout -b <название ветки> - ветвимся для реализации нового функционала
5. вносим какие то изменения в проекте
6. git add . - подготавливаем свои изменения для коммита
7. git commit -m <название коммита> - оборачиваем свои изменения коммитом
8. git push - отправляем свои изменения на удаленный репозиторий

Файл gitignore

Файл .gitignore — это специальный файл в репозитории Git, который указывает, какие файлы и каталоги должны игнорироваться системой контроля версий. Это значит, что указанные файлы и каталоги не будут отслеживаться Git, не будут добавляться в индекс и не будут включаться в коммиты.

Зачем нужен .gitignore

- Исключение временных файлов: Такие файлы, как временные файлы редакторов, компилируемые файлы, файлы логов и другие временные данные, не должны включаться в репозиторий.
- Личные данные: Файлы с конфиденциальной информацией, такие как файлы конфигурации с паролями или ключами API, не должны попадать в репозиторий.
- Унификация команды: Убедиться, что все члены команды игнорируют одни и те же файлы и каталоги.

Что такое github pages

GitHub Pages — это бесплатный хостинг статических веб-сайтов, предоставляемый GitHub. Он позволяет пользователям публиковать веб-страницы прямо из репозиториев GitHub. Сайты на GitHub Pages могут быть использованы для личных блогов, портфолио, документации и других статических веб-ресурсов.

Перейдите на страницу настроек вашего репозитория. В разделе "GitHub Pages" выберите ветку, из которой будут браться файлы для публикации (обычно это main или master), и директорию (обычно корневую).

1. Перейдите в настройки (Settings) репозитория.
2. Прокрутите вниз до раздела "GitHub Pages".
3. В разделе "Source" выберите ветку (например, main) и директорию (/root для корневого каталога).
4. Нажмите "Save".

Какие есть альтернативы GIT-у?

Subversion (SVN):

- Централизованная система контроля версий.
- Имеет централизованный репозиторий, что делает её более простой в настройке, но менее гибкой для работы в оффлайн-режиме.

Mercurial:

- Распределённая система контроля версий, подобная Git.
- Известна своей простотой использования и стабильностью.

Perforce Helix Core:

- Централизованная система контроля версий.
- Подходит для крупных проектов и имеет высокую производительность.

Bazaar:

- Распределённая система контроля версий.
- Предоставляет удобные инструменты для совместной работы.

Ресурсы

- Документация GIT - ТЫК