0. ПОДГОТОВКА

1. Регистрация на GitHub . GitHub - это сервис для хранения репозиториев. Репозиторий научного проекта будет находиться именно там поэтому если у вас еще нет аккаунта на GitHub, вам необходимо пройти стандартную процедуру регистрации.

2. Проверти установлен ли Git на вашем компьютере, зайдя в командную строку и введя команду:

```
git --version
```

3. Установить Git (если его нет, или обновите до последней минорной версии) с официального сайта по инструкции: http://git-scm.com/downloads

4. Все команды git выполняются в терминале. Пользователям Linux/MacOS для этого достаточно запустить Терминал. Пользователям Windows следует использовать утилиту Git Bash

Часть I: Основы Git

Git — это **система контроля версий**, которая позволяет отслеживать изменения в вашем коде с течением времени. Это очень мощный инструмент, который сейчас используется большинством разработчиков программного обеспечения. Это также отличный способ сотрудничать с другими разработчиками над одним проектом.

Система контроля версий GIT предназначена для контролируемого внесения изменений в процессе разработки программного обеспечения, а также при написании статей и другом роде деятельности, связанном с редактированием текстовых файлов (для удобства в дальнейшем будем обсуждать только тексты программ). Система позволяет **отследить** происходящие изменения в исходных текстах программ, что делает удобной совместную разработку.

Механизм ветвей позволяет с легкостью проводить эксперименты над проектом, не затрагивая основного кода, либо вести раздельную разработку модулей, которые впоследствии могут быть включены в основной проект выполнением операции **слияния**.

Внесенные изменения можно контролировать с помощью протокола, который ведет система. Кроме того, система контроля версий в автоматическом режиме отслеживает корректность вносимых изменений при одновременной разработке и предотвращает ситуации противоречащих друг другу исправлений в коде.

Легкая в использовании и одновременно богатая функционально, система контроля версий Git призвана стать незаменимым инструментом любого разработчика!

Git — это программа с текстовым интерфейсом, с которой надо работать в командной строке. Если вы никогда не работали с текстовыми программами или командной строкой, то сначала это все может выглядеть пугающе. Но не волнуйтесь, в этом курсе мы также рассмотрим, как пользоваться командной строкой. К тому же, большинство современных редакторов кода имеют встроенный Git-клиент, который позволяет легко взаимодействовать с Git с помощью графического интерфейса пользователя, минуя командную строку. Однако, все же полезно научиться работать с Git с помощью командной строки:

- Это даст вам лучшее понимание того, как работает Git.
- Это позволит вам использовать Git на любом компьютере, даже если на нем не установлены ваши любимые инструменты разработчика.
- Это позволяет вам использовать Git на удаленном сервере, который не имеет графического пользовательского интерфейса.

Возможности Git:
1. Контроль версий программ
- автоматическое резервное копирование текущей версии программы и соответствующих исходных кодов;
- удобная сводка внесенных изменений;
- интеллектуальное отслеживание противоречащих друг другу изменений;
2. Совместная разработка программы группой программистов:
- контроль ответственности: кто, когда и какие изменения внес в исходный код;
- статистика вклада участников в разработку проекта;
3. Создание версий разработки
- проведение экспериментов, не затрагивая основного кода программы;
- совместная модульная разработка: отдельные модули, которые в дальнейшем включаются в общий проект;
- легкое управление версиями: слияние версий в основной проект, перемещение версий, удаление версий;
4. Использование системы контроля версий для написания статей
- совместное написание с контролем ответственности и разделением обязанностей;
- удобство отслеживания измен
Прежде чем мы начнем, давайте рассмотрим некоторые термины, которые мы будем использовать в этом курсе.

ТЕРМИНОЛОГИЯ

Репозиторий

Репозиторий Git — это хранилище, в котором расположен ваш проект и его история. Это может быть локальное хранилище где-то на вашем компьютере или удаленное хранилище на сервисе типа GitHub или другом хостинге в Интернете. Репозиторий служит для отслеживания изменений в проекте, координации работы между несколькими людьми и отслеживания истории проекта.

Скажем, у вас на компьютере есть директория со всеми файлами вашего проекта. Когда вы инициализируете репозиторий Git в этой директории, Git создает скрытую поддиректорию под названием .git, в которой хранится вся информация о репозитории. Эта информация включает историю всех изменений, внесенных в репозиторий, а также его текущее состояние.

Скучный факт №1: По умолчанию директория .git скрыта. Если вы хотите ее увидеть, убедитесь, что в вашем файловом менеджере включена опция показа скрытых файлов.

Коммит

Вы можете думать о коммите как о снимке вашего проекта в определенный момент времени. Правда, коммит содержит только информацию об изменениях, которые были внесены в репозиторий с момента последнего коммита. Он не содержит все файлы репозитория (если только это не первый коммит). Таким образом, каждый коммит — это небольшой кусочек истории репозитория, основанный на предыдущем коммите. Все они связаны между собой в цепочку, формируя историю изменений вашего проекта.

Ветка

Ветка — это параллельная версия репозитория. Ветки позволяют вам работать над отдельными функциями вашего проекта, не влияя на основную версию. Закончив работу над новой фичей, вы можете объединить эту ветку с основной версией проекта.

***Скучный факт №2*:** В репозитории всегда есть по крайней мере одна ветка, даже если вы сами ее не создавали. Обычно ее называют веткой main (или master).

1. ФИНАЛЬНЫЕ ПРИГОТОВЛЕНИЯ

*01*Установка имени и электронной почты

Зарегистрируйтесь на GitHub.
Если вы никогда ранее не использовали Git, для начала вам необходимо осуществить установку. Выполните следующие команды, чтобы Git узнал ваше имя и электронную почту. Эти данные используются для подписи изменений сделанных вами, что позволит отслеживать, кто и когда сделал изменения в файле.

Выполните

git config --global user.name "Your Name"

git config --global user.email "your_email@whatever.com"

*02*Имя ветки по умолчанию

Мы будем использовать main в качестве имени ветки по умолчанию. Чтобы установить его, выполните следующую команду:

Выполните

git config --global init.defaultBranch main

*03*Корректная обработка окончаний строк

Для пользователей Unix/Mac:

Выполните

git config --global core.autocrlf input

git config --global core.safecrlf warn

Для пользователей Windows:

Выполните

git config --global core.autocrlf true

git config --global core.safecrlf warn

2. СОЗДАНИЕ ПРОЕКТА

*01*Создайте страницу «Hello, World»

Начните работу в пустой директории (например, respositories, если вы скачали архив с предыдущего шага) с создания пустой поддиректории work, затем войдите в неё и создайте там файл hello.html с таким содержанием:

Выполните

mkdir work

cd work

touch hello.html

Файл: hello.html

Hello, World

***02*Создайте репозиторий**

Теперь у вас есть директория с одним файлом. Чтобы создать Git-репозиторий из этой директории, выполните команду git init.

Выполните

git init

Результат

Initialized empty Git repository in /home/alex/githowto/repositories/work/.git/

***03*Добавьте страницу в репозиторий**

Теперь давайте добавим в репозиторий страницу «Hello, World».

Выполните

git add hello.html

git commit -m "Initial Commit"

Вы увидите...

Результат

$ git add hello.html

$ git commit -m "Initial commit"

[main (root-commit) 5836970] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 hello.html

3. ПРОВЕРКА СОСТОЯНИЯ

***01*Проверьте состояние репозитория**

Используйте команду git status, чтобы проверить текущее состояние репозитория.

Выполните

git status

Вы увидите:

Результат

$ git status

On branch main

nothing to commit, working tree clean

Если после выполнения предыдущей команды вы видите On branch master вместо On branch main, это означает, что у вас немного устаревшая версия Git'a, которая не поняла нас, когда мы попросили установить имя ветки по умолчанию на main. В этом случае вы можете переименовать ветку в main с помощью следующей команды:

git branch -m master main

Команда проверки состояния сообщила, что коммитить нечего. Это означает, что в репозитории уже хранится текущее состояние рабочих файлов, и нет никаких изменений, которые могли бы ожидать записи.

Мы будем использовать команду git status, чтобы продолжать отслеживать состояние репозитория и рабочей директории.

4. ВНЕСЕНИЕ ИЗМЕНЕНИЙ

***01*Измените страницу «Hello, World»**

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

Файл: hello.html

<h1>Hello, World!</h1>

***02*Проверьте состояние**

Теперь проверьте состояние рабочей директории.

Выполните

git status

Вы увидите...

Результат

$ git status

On branch main

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: hello.html

no changes added to commit (use "git add" and/or "git commit -a")

Первое, что нужно заметить, это то, что Git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории.

Также обратите внимание на то, что сообщение о состоянии дает вам подсказку о том, что нужно делать дальше. Если вы хотите добавить эти изменения в репозиторий, используйте команду git add. В противном случае используйте команду git checkout для отмены изменений.

Полезные ссылки:

- Основные команды https://training.github.com/downloads/ru/github-git-cheat-sheet/
- «Методичка» по работе с Git https://andron13.de/pdf/github-basic.pdf
- Документация GitHub https://docs.github.com/ru/get-started/getting-started-with-git/set-up-git
- Документация Githttps://git-scm.com/doc