Частное учреждение профессионального образования

«Высшая школа предпринимательства (колледж)»

(ЧУПО «ВШП»)

Кафедра информационных технологий

Инструментальные средства разработки программного обеспечения

Для разрядки и настройки

Требования:

— базовые знания JS, SCSS, HTML, CSS

— делал какие-то pet-проекты на React и, возможно, Node.js

— Писал на TypeScript

— проходил курсы для фронтенд-разработчиков по React'у (в сопроводительном письме указать какие) или обучался в профильном IT-вузе (также указать в каком)

Условия:

— Это **неоплачиваемая** стажировка, т.е. вы в течение 6 мес. не будете получать ЗП, т.к. для компании вы еще не можете давать результат.

— **Удаленная работа.** Возможность работы из дома.

— **Свободная атмосфера.** Минимум бюрократии и прочей мишуры.

— **Правда и уважение.** Это наши постулаты успешной работы.

Что такое «система контроля версий» и почему это важно?

Система контроля версий

Система контроля версий (от англ. Version Control System, VCS) - это система, записывающая изменения в файл или набор файлов в течение времени. Позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему. Использование VCS также значит в целом, что, если вы сломали что-то или потеряли файлы, вы спокойно можете всё исправить.

Какие виды VCS бывают?

Виды VCS

- Локальные системы контроля версий
- Централизованные системы контроля версий
- Распределённые системы контроля версий

Самая популярная VCS

Git — мощная и сложная распределенная система контроля версий. Понимание всех возможностей git открывает для разработчика новые горизонты в управлении исходным кодом.

Повторим термины

- **Git или Гит**
- **Репозиторий Git**
- **Локальный репозиторий**
- **Удалённый репозиторий**
- **Форк (Fork)**
- **Клонирование (Clone)**
- **Ветка (Branch)**
- **Мастер (Master) или Мейн (Main)**
- **Коммит (Commit)**
- **Пул (Pull)**
- **Пуш (Push)**
- **Мёрдж (Merge)**

Термины

Git или Гит — система контроля и управления версиями файлов.

GitHub или Гитхаб — веб-сервис для размещения репозиториев и совместной разработки проектов.

Репозиторий Git — каталог файловой системы, в котором находятся: файлы конфигурации, файлы журналов операций, выполняемых над репозиторием, индекс расположения файлов и хранилище, содержащее сами контролируемые файлы.

Локальный репозиторий — репозиторий, расположенный на локальном компьютере разработчика в каталоге. Именно в нём происходит разработка и фиксация изменений, которые отправляются на удалённый репозиторий.

Удалённый репозиторий — репозиторий, находящийся на удалённом сервере. Это общий репозиторий, в который приходят все изменения и из которого забираются все обновления.

Термины(1)

Форк (Fork) — копия репозитория. Его также можно рассматривать как внешнюю ветку для текущего репозитория. Копия вашего открытого репозитория на Гитхабе может быть сделана любым пользователем, после чего он может прислать изменения в ваш репозиторий через пулреквест.

Клонирование (Clone) — скачивание репозитория с удалённого сервера на локальный компьютер в определённый каталог для дальнейшей работы с этим каталогом как с репозиторием.

Ветка (Branch) — это параллельная версия репозитория. Она включена в этот репозиторий, но не влияет на главную версию, тем самым позволяя свободно работать в параллельной. Когда вы внесли нужные изменения, то вы можете объединить их с главной версией.

Мастер (Master) или Мейн (Main) — главная или основная ветка репозитория.

Термины(2)

Коммит (Commit) — фиксация изменений или запись изменений в репозиторий. Коммит происходит на локальной машине.

Пул (Pull) — получение последних изменений с удалённого сервера репозитория.

Пуш (Push) — отправка всех неотправленных коммитов на удалённый сервер репозитория.

Мёрдж (Merge) — слияние изменений из какой-либо ветки репозитория с любой веткой этого же репозитория. Чаще всего слияние изменений из ветки репозитория с основной веткой репозитория.

Разберем основные команды Git

git init

git status

git add

git commit

git log

Начало работы с Git

Обычно вы получаете репозиторий Git одним из двух способов:

1. Вы можете взять локальный каталог, который в настоящее время не находится под версионным контролем, и **превратить** его в репозиторий Git
2. Вы можете **клонировать** существующий репозиторий Git из любого места.

В обоих случаях вы получите готовый к работе Git репозиторий на вашем компьютере.

git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем **.git**, содержащий все необходимые файлы репозитория — структуру Git репозитория.

На этом этапе ваш проект ещё не находится под версионным контролем.

git add

Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит добавить их в индекс и осуществить первый коммит изменений.

Добавить файлы в индекс репозитория можно запустив команду **git add**, указав индексируемые файлы, а затем выполнив **git commit**

Разберем подробнее что мы должны делать в реальности

У вас имеется Git-репозиторий с рабочей копией файлов для некоторого проекта. Вам нужно делать некоторые изменения и фиксировать «снимки» состояния этих изменений в вашем репозитории каждый раз, когда проект достигает состояния, которое вам хотелось бы сохранить.

Каждый файл в вашем рабочем каталоге может находиться в одном из двух состояний: под версионным контролем (отслеживаемые) и нет (неотслеживаемые).

Состояния файлов

Отслеживаемые файлы — это те файлы, которые были в последнем снимке состояния проекта; они могут быть неизменёнными, изменёнными или подготовленными к коммиту. Если кратко, то отслеживаемые файлы — это те файлы, о которых знает Git.

Неотслеживаемые файлы — это всё остальное, любые файлы в вашем рабочем каталоге, которые не входили в ваш последний снимок состояния и не подготовлены к коммиту.

git status

```
git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
```

Это означает, что у вас чистый рабочий каталог, в нём нет отслеживаемых изменённых файлов.
Git также не обнаружил **неотслеживаемых** файлов, иначе они бы были перечислены здесь.
Команда сообщает вам на какой ветке вы находитесь и сообщает вам, что она не расходится с веткой на сервере

git commit

Коммит сохраняет снимок состояния вашего индекса.
Всё, что вы не проиндексировали, так и висит в рабочем каталоге как изменённое.
Можете сделать ещё один **коммит**, чтобы добавить эти изменения в **репозиторий**.
Каждый раз, когда вы делаете **коммит**, вы сохраняете снимок состояния вашего проекта, который позже вы можете восстановить или с которым можно сравнить текущее состояние

git log

Вы создали несколько **коммитов** или же **клонировали репозиторий** с уже существующей историей коммитов, вам понадобится возможность посмотреть что было сделано — историю **коммитов**.
Одним из основных и наиболее мощных инструментов для этого является команда **git log**

По умолчанию **git log** перечисляет **коммиты**, сделанные в **репозитории** в обратном к хронологическому порядке — последние **коммиты** находятся вверху

!!! Внимание !!!

Самый важный слайд, без комментариев

git --version

git config --global user.name "ваше имя"

git config --global user.email email@example.com

git add -A

git add .

git commit -m "ваше сообщение"

git checkout -b имя-новой-ветки

git branch -m старое-имя-ветки новое-имя-ветки

:wq

Как успехи?

learngitbranching.js.org

Для закрепления материала

githowto.com/ru