

СТРУКТУРА GIT

Git – распределенная система контроля версий. Git представляет собой серверную часть, которая отвечает за версионное хранение документов и обработку запросов, клиентскую часть, позволяющую формировать запросы к серверу Git и набора конфигурационных файлов, хранящих настройки пользователя и позволяющих адаптировать Git для самых разных задач.

Git состоит из набора консольных утилит и файла конфигурации. За счет такой архитектуры достигается переносимость на разные платформы и позволяет создавать графические клиенты для работы с проектами.

Структура репозитория (отдельного проекта) Git представляет собой каталог файловой системы, в котором хранятся конфигурационные файлы, журналы — файлы изменений репозитория, файлы индексов, ускоряющие поиск по репозиторию и файлы проекта. Дерево файлов в репозитории не отражает реальное хранение файлов, так как Git изменяет расположение файлов для ускорения доступа. Если при работе с проектом файл изменяется, то в репозитории Git создается новый файл. Таким образом, при изменении Git не изменяет старые файлы, а создает новые. Пользователь же видит «снимок» файловой системы на определенный момент времени (под снимком здесь понимается информация на определенный период времени, т. е. информация созданная до запрашиваемого периода и измененная после этого периода). Из репозитория файлы не удаляются. Они могут быть лишь помечены как удаленные.

По умолчанию конфигурация репозитория хранится в каталоге “.git”, корневого каталога проекта. Таким образом, достаточно легко превратить любой каталог в репозиторий или импортировать существующий репозиторий вызвав соответствующую утилиту, которая создаст подкаталог с необходимыми файлами или исправит существующие файлы в случае импорта.

Для каждого объекта в репозитории считается SHA-1 хеш, и именно он становится именем файла, содержащего данный объект в директории .git/objects. Для оптимизации работы с файловыми системами, не использующими деревья для директорий, первый байт хэша становится именем поддиректории, а остальные — именем файла в ней, что снижает количество файлов в одной директории

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и коммит. Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных поддиректорий, коммит — дерево + некая дополнительная информация

(например, родительский(е) коммит(ы), а также комментарий).

В репозитории иногда производится сборка мусора, во время которой устаревшие файлы заменяются на «дельты» между ними и файлами современными, после чего данные «дельты» складываются в один большой файл, к которому строится индекс. Это снижает требования по месту на диске.

Репозиторий git бывает локальный и удаленный. Локальный репозиторий — это поддиректория .git, создается (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удаленного репозитория и простановкой ссылки на родителя) командой `git clone`.

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удаленный репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

Работа с Git из командной строки

Для работы с Git-репозиторием необходимо установить утилиту git. Данная утилита находится в сети интернет в свободном доступе. Для настройки конфигурации Git используются следующие команды:

Команда	Комментарий
<code>git config --global user.name Имя</code>	Задаёт имя пользователя
<code>git config --global user.email email@example.com</code>	Задаёт email пользователя
<code>git init</code>	Если проекта нет, его можно инициализировать командой
<code>git status</code>	Состояние: что было отредактировано, что добавлено в индекс для коммита
<code>git add .</code>	добавить в индекс все изменения
<code>git add file.txt</code>	добавить содержимое файла в индекс
<code>git commit</code>	Коммит проекта
<code>git commit -am</code>	Коммит без добавления новых файлов

Команда	Комментарий
"comment" - add + commit	
git branch	список веток
git checkout имя_ветки	Переход в другую ветку
git checkout -f	отметить незакоммиченные изменения
git diff ветка1 ветка2	Сравнить ветка1 и ветка2
git merge новая_ветка	Объединение текущей ветки и новая_ветка
git reset --hard HEAD	Вернуть ветку в состояние до слияния
git push pb master	Отправить код в ветку master удаленного репозитория pb
git fetch pb	Извлечь информацию из репозитория pb

Задание

1 Если система контроля версий Git не установлена, то установить ее (параметры оставить по умолчанию).

2 Запустить Git GUI (или консоль). Создать новый репозиторий. Добавить в папку репозитория файлы. Зафиксировать состояние репозитория (выполнить commit).

3 Внести изменения в файлы. Зафиксировать новое состояние репозитория.

4 Создать новую ветку. Внести в нее изменения и зафиксировать их.

5 Переключиться на ветку мастера. Внести в нее изменения и зафиксировать их.

6 Продемонстрировать слияние веток.

7 Просмотреть дерево изменений веток (историю).

8 Продемонстрировать откат изменений в ветке.

В отчете показать выполняемую команду и результат выполнения (снимок GUI или вывод консоли).

Для защиты знать основные команды Git (см. таблицу выше).

Материалы для выполнения работы

- Система управления версиями Git:
- Дистрибутив выдается преподавателем или скачивается из интернета.
- Дополнительно - Краткое руководство по командам Git

<http://www.calculate-linux.ru/main/ru/git>

- Дополнительно - Подробное руководство по командам Git

<http://marklodato.github.io/visual-git-guide/index-ru.html>