МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа №2**

**По дисциплине «Инструментальные средства разработки ПО»**

**Изучение системы управления версиями Git**

**Выполнил студент группы №М32101**

**Аль Даббагх Харит Хуссейн**

**Проверил**

**Дядюшкин Александр Александрович**

САНКТ-ПЕТЕРБУРГ

2021

Содержание

[Система управления версиями Git 2](#_Toc65401476)

[Установка Git 3](#_Toc65401477)

[Базовое использование Git 3](#_Toc65401478)

[Конвенции 4](#_Toc65401479)

[Добавления 4](#_Toc65401480)

[Другие способности 4](#_Toc65401481)

# Система управления версиями Git

Система управления версиями записывает изменения, внесенные в наш код с течением времени, в специальную базу данных, называемую репозиторием. Она позволяет увидеть историю проекта, кто, когда и почему сделал изменения. Если что-то пойдет не так, мы сможем легко вернуться в раннее состояние.

Без нее нам пришлось бы сделать несколько копий проекта, а отследить работу каждого из них очень сложно. А передача кода должна была бы осуществляться с помощью такого механизма, как переписка по электронной почте и передача файлов.

Главной идеей такой системы является:

* Отслеживание истории.
* Совместная работа.

Существует два типа контроля версий:

* Централизованный.
* Распределенный.

Проблема с централизованным типом является единственной точкой аварии, если сервер выйдет из сети, мы не сможем сотрудничать или сохранить снимки нашего проекта. В распределенной системе такой проблемы нет. Во втором типе у каждого члена команды есть копия проекта с его историей на машине. Git относится ко второму типу.

Git также является наиболее популярным типом таких систем, потому что он бесплатный, с открытым исходным кодом, быстрый и масштабируемый.

Чтобы использовать Git, у нас есть несколько вариантов:

* Интерфейс командной строки (CLI).
* Текстовые редакторы или IDE.
* Графический интерфейс пользователя.

CLI - самый используемый и самый важный для того, чтобы научиться быть профессиональным в Git, современные IDE добавляют поддержку Git, например VS Code. А также git может быть использован, например, в GUI: GitKraken и Sourcetree.

## Установка Git

[Отсюда](http://git-scm.com/downloads) Git может быть установлен в соответствии с той операционной системой, которую мы используем.

Для своей ситуации я использую Windows OS. Во время установки Git'а мы также получим некоторые дополнения, такие как Git Bash и Git Gui.

Git Bash - это CLI, который добавляет команды linux в Windows, и его рекомендуется использовать вместо CMD.

Git Gui - это графический пользовательский интерфейс, выполняющий те же функции и команды.

## Базовое использование Git

Основная идея Git'а - создание репозитория, и на частых основаниях добавление в него снэпшотов нашей работы, по некоторым базовым правилам.

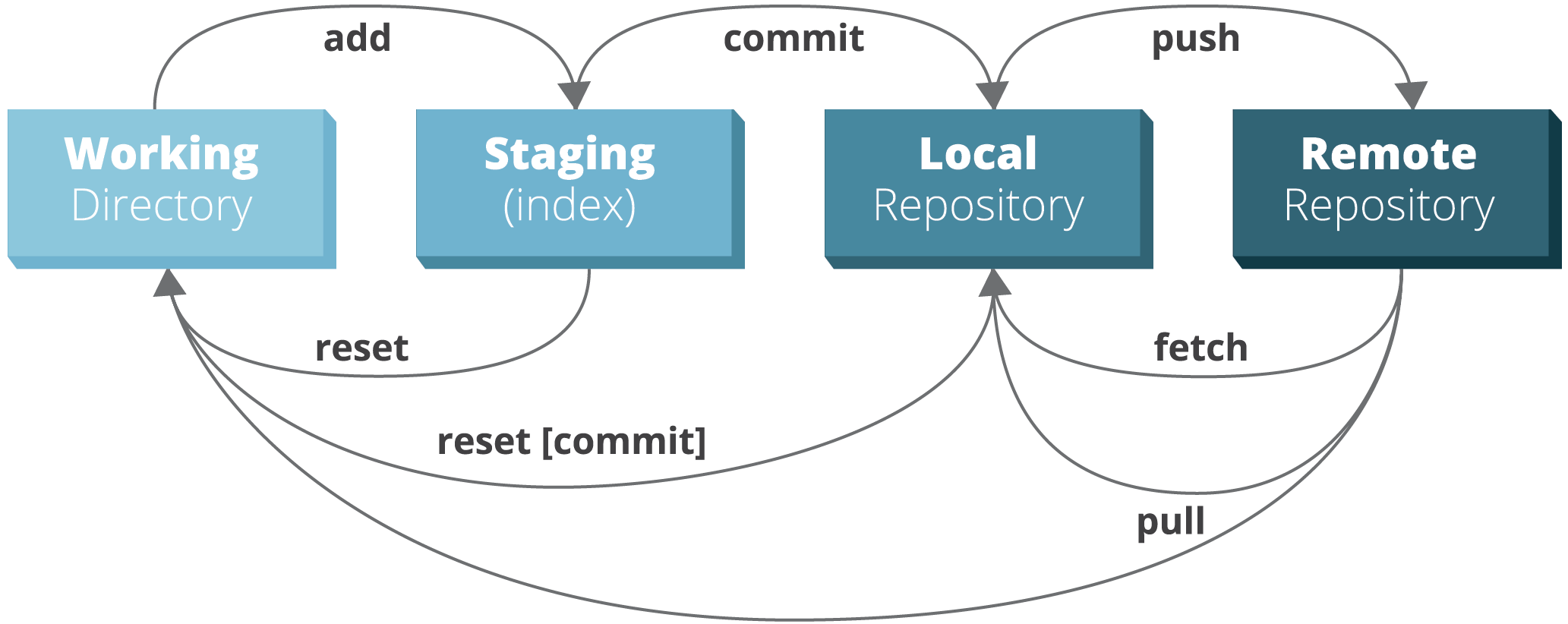
Не рекомендуется слишком часто вносить изменения в репозиторий, а также не рекомендуется делать это в больших объёмах (большие изменения/дополнения).

Самая основная часть Git'а - это то, что у него есть специальный атрибут, который отличает его от других систем управления версиями, называемый "Staging area" или индексом.

Что происходит в индексе, так это то, что мы сначала добавляем наши файлы в эту зону, убеждаемся, что все правильно, а затем фиксируем эти изменения в репозитории в виде снапшота.

Каждый коммит в системе имеет ID, сообщение, дату/время, автора и сам снапшот.

Следующая картинка в основном иллюстрирует то, как Git работает.



## Конвенции

Некоторые конвенции приняты с целью организации истории, особенно те из них, которые требуют участия многих авторов.

Веточка "master" по умолчанию создаётся с помощью git init и часто используется в качестве ветки, в которую сливаются другие изменения.

Конвенции GitFlow и именах часто применяются для различения нестабильных историй (feature/\*), нестабильных общих историй (develop), историй готовности к производству (master), а также патчей к выпущенным продуктам (hotfix).

## Добавления

В Git'е также могут быть установлены некоторые расширения, облегчающие его использование. Например, GitFlow, который облегчает работу с ветвлением репозитория.

Другим добавлением может быть LFS, которая отслеживает большие файлы и добавляет к ним указатели вместо коммита в репозиторий. Это особенно полезно для разработчиков игр, поскольку они могут работать с большими файлами, такими как уровни или бинарные файлы.

## Другие способности

Git позволяет добавить файл readme, чтобы рассказать другим людям, почему проект полезен, что они могут сделать с проектом и как они могут его использовать.

Git также имеет возможность помечать теги определенных точек в истории репозитория, которые являются важными. Обычно люди используют эту функциональность для пометки точек релиза (v1.0, v2.5...).

Часто бывает так, что при работе над одним проектом необходимо использовать другой проект прямо изнутри. Возможно, это библиотека, которую разработала третья сторона или которую мы разрабатываем отдельно и используем в нескольких родительских проектах. В таких сценариях возникает общая проблема: мы хотим иметь возможность рассматривать два проекта как отдельные, но при этом иметь возможность использовать один из них изнутри другого. Git решает эту проблему с помощью субмодулей. Субмодули позволяют нам хранить Git-репозиторий в качестве подкаталога другого Git-репозитория. Это позволяет нам клонировать другой репозиторий в наш проект и держать коммиты отдельно.