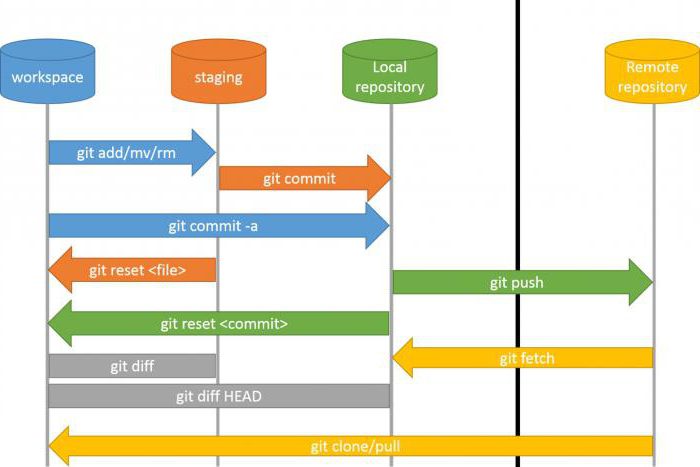
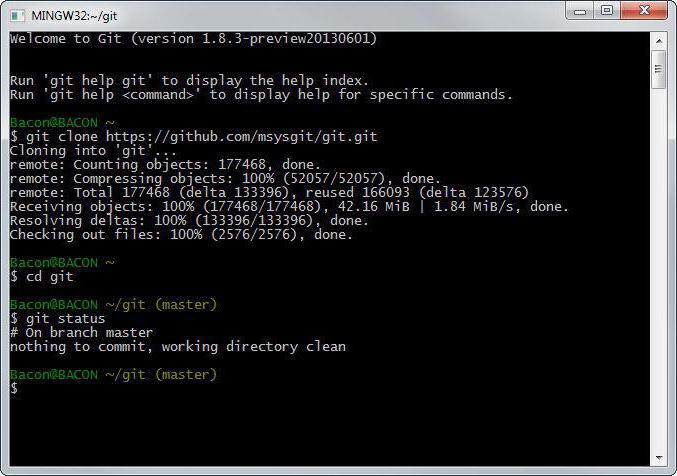
**Git: описание и особенности системы**

Система контроля версий Git обладает своими особенностями. Большинство систем хранит файлы, изменяя их в соответствии с инструкциями в проекте. То есть, к примеру, версия текущей разработки под номером 3 может содержать данные об изменениях в файле А и Б. А уже версия 4 будет иметь в себе А, Б и В. Git работает немного по-другому. Каждая версия проекта будет содержать в себе варианты всех файлов А, Б и В. Вне зависимости от того, в скольких из них будут проведены изменения. Конечно, Git не хранит каждый экземпляр файла, а использует всего лишь ссылку на него. Ещё одной важной особенностью является возможность непосредственной работы с локальным хранилищем. То есть если требуется делать правки, то их реализуют прямо на компьютере участника проекта. Это очень удобно, когда разработчик находится вдали от сети Интернет. Затем, при получении доступа к серверу, появится возможность отправить все данные в общее хранилище. Для сохранения целостности данных применяется метод хеширования каждого изменённого файла методом SHA-1. Это позволяет системе контроля версий точно знать, где, кто и когда изменил файл. Git: установка Для того чтобы начать работать с Git, нужно его установить. Система контроля версий доступна для использования в Windows, Mac OS, Linux. Версию для Windows можно скачать по адресу: git-for-windows.github.io. После загрузки программу нужно установить. Инсталлятор простой, так что эта процедура не должна вызвать проблем

Очень часто пользователи сталкиваются с такой проблемой, при которой консоль пишет, что Git не является внутренней или внешней командой при попытке ввести данные. Решением может послужить добавление полного пути к каталогу в переменную окружения PATH.

Первые команды После установки системы контроля версий на компьютер, ее необходимо настроить. Стоит отметить, что к этому моменту на сайте Github должен быть зарегистрирован свой аккаунт. Для того чтобы выполнить первоначальную настройку, нужно вызвать консольный клиент, так как придётся вводить команды. Сделать это можно, нажав правую кнопку мыши и выбрав пункт Git Bash. Должно открыться окно консольного клиента, которое пригласит ввести данные. В нем нужно выполнить последовательно Git Bash команды: git config --global user.name ''Имя''; git config --global user.mail ''Адрес электронной почты''. На этом же этапе необходимо настроить метод окончания строк с помощью двух команд: git config --global core.autocrlf true; git config --global core.safecrlf false. Для первичной настройки Git этого достаточно.

Далее будут использоваться только команды для управления проектами. 

**Основные команды Git**

Init: данная команда создаёт новый репозиторий. Пример использования: init имя проекта.

Clone. Производит копирование существующего репозитория с удалённого источника. Вот так используется команда git clone: clone git://github.com/путь к репозиторию.

Add. Наиболее часто используемая команда в системе контроля версий Git. Она выполняет простую задачу — добавляет указанные файлы в особую область, именуемую индексом или сценой. В неё можно перенести несколько файлов или папок, которые необходимо будет впоследствии добавить в репозиторий или, выражаясь на языке Git, «закоммитить». Пример использования этой Git команды выглядит так: add некий\_файл.txt.

Status. Позволяет просмотреть список файлов, которые присутствуют в индексе и рабочей папке. Служит для контроля и просмотра готовых к коммиту данных или их изменённых, но не внесённых версий в сцену.

Diff. Показывает разницу состояний. Например, с помощью этой Git команды можно определить, есть ли изменения между папкой с проектом и индексом. Commit. Выполняет сохранение слепка всего того, что находилось в индексе непосредственно в базу данных. В результате работы Git команды на экране отобразится текстовый файл, в котором можно указать, какие именно изменения были произведены. А также будет выведена информация о том, сколько файлов подверглись коммиту, и его контрольная сумма. Главное - не забывать о том, что после изменения в базу попадут только те данные, которые были занесены в индекс командой git add.

**Дополнительные команды**

Git Reset. О функциях этой команды говорит ее название. Она просто выбрасывает из специальной промежуточной области — индекса, указанный файл, помещённый туда по случайности. Стоит осторожно обращаться с reset при использовании команды с ключом - - hard, так как это затронет и файлы в рабочей папке, что может привести к непредвиденным последствиям. Rm. Наиболее точно эту команду можно описать как обратную git add, так как она удаляет файлы из индекса. Правда, при этом ещё и из рабочей папки.

Mv. Служит для перемещения файла.

Clean. Предназначена для очистки папки проекта от ненужных файлов. Представленные команды используются для общих нужд проекта.