**Первоначальная настройка**

Настройка информации о пользователе для всех локальных репозиториев

$ git config --global user.name "[имя]"

Устанавливает имя, которое будет отображаться в поле автора у выполняемых вами коммитов

$ git config --global user.email "[адрес электронной почты]"

Устанавливает адрес электронной почты, который будет отображаться в информации о выполняемых вами коммитах.

Обычно начинают с [**познавательного чтения**](http://git-scm.com/book/ru/v1/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D1%83%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F%D0%BC%D0%B8)  
  
Коротко. Репозиторий живет на вашей машине. Вы привязывете его к github

git remote add origin git://github.com/paulboone/ticgit.git

И при необходимости, выливаете изменения:  
git push origin master

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду

$git remote.

Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование:

**Создание репозитория**

Создание нового репозитория или получение его по существующему URL-адресу

$ git init [название проекта]

Создаёт новый локальный репозиторий с заданным именем

$ git clone [url-адрес]

Скачивает репозиторий вместе со всей его историей изменений

**Внесение изменений**

Просмотр изменений и создание коммитов (фиксация изменений)

$ git status

Перечисляет все новые или изменённые файлы, которые нуждаются в фиксации

$ git diff

Показывает различия по внесённым изменениям в ещё не проиндексированных файлах

$ git add [файл]

Индексирует указанный файл для последующего коммита

$ git commit -a

Замеяет сразу ДВЕ команды (git add и git commit)

$ git commit -am “…”

Замеяет сразу ТРИ команды (git add, git commit и git commit -m)

$ git diff --staged

Показывает различия между проиндексированной и последней зафиксированной версиями файлов

$ git reset [файл]

Отменяет индексацию указанного файла, при этом сохраняет его содержимое

$ git commit -m "[сообщение с описанием]"

Фиксирует проиндексированные изменения и сохраняет их в историю версий

**Коллективная работа**

Именованные серии коммитов и соединение результатов работы

$ git branch

Список именованных веток коммитов с указанием выбранной ветки

$ git branch [имя ветки]

Создаёт новую ветку

$ git switch -c [имя ветки]

Переключается на выбранную ветку и обновляет рабочую директорию до её состояния

$ git merge [имя ветки]

Вносит изменения указанной ветки в текущую ветку

$ git branch -d [имя ветки]

Удаляет выбранную ветку

**Операции с файлами**

Перемещение и удаление версий файлов репозитория

$ git rm [файл]

Удаляет конкретный файл из рабочей директории и индексирует его удаление

$ git rm --cached [файл]

Убирает конкретный файл из контроля версий, но физически оставляет его на своём месте

$ git mv [оригинальный файл] [новое имя]

Перемещает и переименовывает указанный файл, сразу индексируя его для последующего коммита

**Игнорирование некоторых файлов**

Исключение временных и вторичных файлов и директорий

\*.log

build/

temp-\*

Git будет игнорировать файлы и директории, перечисленные в файле .gitignore с помощью wildcard синтаксиса

$ git ls-files --others --ignored --exclude-standard

Список всех игнорируемых файлов в текущем проекте

**Сохранение фрагментов**

Сохранение и восстановление незавершённых изменений

$ git stash

Временно сохраняет все незафиксированные изменения отслеживаемых файлов

$ git stash pop

Восстанавливает состояние ранее сохранённых версий файлов

$ git stash list

Выводит список всех временных сохранений

$ git stash drop

Сбрасывает последние временно сохранённыe изменения

**Просмотр истории**

Просмотр и изучение истории изменений файлов проекта

$ git log

История коммитов для текущей ветки

$ git log --follow [файл]

История изменений конкретного файла, включая его переименование

$ git diff [первая ветка]...[вторая ветка]

Показывает разницу между содержанием коммитов двух веток

$ git show [коммит]

Выводит информацию и показывает изменения в выбранном коммите

**Откат коммитов**

Удаление ошибок и корректировка созданной истории

$ git reset [коммит]

Отменяет все коммиты после заданного, оставляя все изменения в рабочей директории

$ git reset --hard [коммит]

Сбрасывает всю историю вместе с состоянием рабочей директории до указанного коммита.

**Синхронизация с удалённым репозиторием**

Регистрация удалённого репозитория и обмен изменениями

$ git fetch [удалённый репозиторий]

Скачивает всю историю из удалённого репозитория

$ git merge [удалённый репозиторий]/[ветка]

Вносит изменения из ветки удалённого репозитория в текущую ветку локального репозитория

$ git push [удалённый репозиторий] [ветка]

Загружает все изменения локальной ветки в удалённый репозиторий

$ git pull

Загружает историю из удалённого репозитория и объединяет её с локальной. pull = fetch + merge