

**Топ-12 наиболее часто используемых команд Git для разработчиков**

**(https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp)**

Git — это **распределенная система контроля версий** и программное обеспечение с открытым исходным кодом. Git позволяет разработчикам легко **управлять многими версиями исходного кода**. Вы можете использовать его, чтобы выяснить, **кто, что** сделал, **когда и почему**. В настоящее время Git стал обязательным инструментом для любого разработчика. Существуют сотни команд Git, но регулярно используются лишь некоторые из них.

1. **git help**

git <command> help

**1) git config**

Перед использованием Git, вам необходимо его настроить. Эта команда позволяет указать имя пользователя и адрес электронной почты, которые будут использоваться с вашими коммитами.

# sets up Git with your name (# - строка комментарий)

**$ git config --global user.name "<Your-Full-Name>"**

# sets up Git with your email

**$ git config --global user.email "<your-email-address>"**

### 2) git init

Сначала необходимо создать репозиторий git, прежде чем вы сможете делать коммиты или делать с ним что-либо еще. Для этого используется команда **git init** - создание нового репозитория **Git**. Команда **git init** выполняет всю первоначальную настройку репозитория, создает необходимые файлы и каталоги для отслеживания всех изменений. Все эти файлы хранятся в каталоге с именем **.git** (обратите внимание на . в начале; это означает, что он будет скрыт на Mac/Linux/windows). «Репо» — это каталог .git (repository).

**$ git init**

### 3) git clone

<https://github.com/bogevg/delphi.git> URL Евгении

Команда, которую мы будем выполнять на терминале, — это **git clone**, и именно так клонирование связано с Git. Команда git clone получает путь (обычно **URL-адрес**) к репозиторию Git, который вы хотите клонировать. Команда — git clone, а в качестве аргумента передается путь к репозиторию Git, который вы хотите клонировать. Это URL-адрес проекта, над которым мы будем работать на протяжении всего курса. git clone создает локальную рабочую копию исходного кода удаленного репозитория. Когда вы клонируете репозиторий, код автоматически загружается на ваш компьютер. Если у вас есть разрешение, эта команда добавит исходное местоположение в качестве удаленного местоположения, что позволит вам получать из него изменения и отправлять в него изменения.

**$ git clone https://github.com/<repo-url>**

### 4) git status

### Команда git status — это наш ключ к пониманию Git. Она будет информировать нас информировать о состоянии нашего репозитория, как его видит Git. Когда вы только начинаете, вам всегда следует использовать команду git status! Серьезно. Рекомендуется запускать его после любой другой команды. Это поможет вам изучить Git и избежать (потенциально) неточных предположений о состоянии ваших файлов/репозитория.

**$ git status**

### В зависимости от состояния ваших файлов, рабочего каталога и репозитория команда git status отобразит соответствующую информацию.

### 5) git add

### Чтобы переместить файлы из рабочего каталога в промежуточный индекс, используйте команду git add. Команда git add сохраняет изменения в файле в промежуточной области, что позволяет вам сравнить локальную версию с версией удаленного репозитория. Прежде чем зафиксировать новый или измененный файл, используйте команду git add, чтобы добавить его в промежуточную область. Чтобы добавить определенные файлы:

### $ git add <файл1> <файл2> … <файлN>

### Чтобы добавить все файлы:

### $ git add .

### 6) git commit

### Эта команда сохраняет сообщение в логе вместе с идентификатором коммита (контрольной суммой) зафиксированных изменений. Вы должны включать краткое описание изменений, вносимых каждый раз, когда вы фиксируете изменения в коде. Это сообщение о фиксации помогает тебе и другим понять внесенные изменения.

**$ git commit –m "<Type your commit message here>"**

### 7) git push

### Эта команда передает содержимое вашего локального репозитория в добавленный вами удаленный репозиторий. Это переместит коммиты вашей основной ветки в недавно добавленный удаленный репозиторий. Если именованная ветка не существует в удаленном репозитории, она будет создана.

**$ git push**

### 8) git branch

Добавьте новую ветку к существующей ветке, перечислите все существующие ветки и удалите ветку с помощью git Branch. Эта команда используется для выполнения операций над указанной веткой. Когда вы запустите эту команду, Git удалит все файлы и папки из рабочего каталога, который он отслеживает (файлы, которые отслеживает Git, хранятся в репозитории, поэтому ничего не потеряно) и извлечет все файлы и каталоги из коммита, который содержит ветка. ссылки из репозитория. Создайте новую ветку локально:

**$ git branch <branch-name>**

Взгляните на ветки и определите, над какой веткой вы сейчас работаете:

**$ git branch or $ git branch --list**

### 9) git checkout

Мы можем использовать команду git checkout для переключения на существующую ветку или для создания и переключения на новую. Для этого ветка, на которую вы хотите переключиться, должна присутствовать в вашей локальной системе, а любые изменения, внесенные в вашу текущую ветку, должны быть зафиксированы или сохранены перед переключением. Эту команду также можно использовать для извлечения файлов. Когда вы запустите эту команду, Git удалит все файлы и папки из рабочего каталога, который он отслеживает (файлы, которые отслеживает Git, хранятся в репозитории, поэтому ничего не потеряно) и извлечет все файлы и каталоги из коммита, который содержит ветка. ссылки из репозитория.

**$ git checkout <branch-name>**

### 10) git merge

С помощью этой команды история указанной ветки объединяется с текущей веткой. Команда git merge объединяет вашу ветку с родительской веткой. В зависимости от вашего процесса родительская ветка может быть либо веткой разработки, либо основной веткой. Если конфликтов нет, новая фиксация будет сделана автоматически. Прежде чем запускать команду git merge, вы должны находиться в ветке, которую хотите объединить с родительской веткой. С помощью этой команды история указанной ветки объединяется с текущей веткой.

**$ git merge <name-of-branch-to-merge-in>**

### 11) git pull

Содержимое удаленного репозитория извлекается и интегрируется в ваш локальный репозиторий с помощью этой команды. git pull переносит самые последние изменения с удаленного сервера в локальный репозиторий, гарантируя, что у вас будет самая актуальная информация от ваших коллег.

**$ git pull**

### 12) git log

Команда git log используется для отображения всех коммитов репозитория. Эта команда отображает журнал всех коммитов, сделанных на данный момент в **текущей** ветке.

**$ git log**