# 1. Подготовка

### Цели

* Полная готовность к работе с Git.

## Установка имени и электронной почты

Если вы никогда ранее не использовали git, для начала вам необходимо осуществить установку. Выполните следующие команды, чтобы git узнал ваше имя и электронную почту. Если git уже установлен, можете переходить к разделу [окончания строк](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8).

#### ВЫПОЛНИТЬ:

git config --global user.name "Your Name"

git config --global user.email "your\_email@whatever.com"

## Параметры установки окончаний строк

Также, для пользователей Unix/Mac:

#### ВЫПОЛНИТЬ:

git config --global core.autocrlf input

git config --global core.safecrlf true

Для пользователей Windows:

#### ВЫПОЛНИТЬ:

git config --global core.autocrlf true

git config --global core.safecrlf true

# 2. Финальные приготовления

### Цели

* Установить материалы учебника и подготовить их к работе.

## 01Скачайте учебные материалы

Скачайте учебные материалы здесь:

* <http://githowto.com/git_tutorial.zip>

## 02Распакуйте учебные материалы

Пакет учебных материалов должен иметь главную папку «git\_tutorial» с двумя подпапками:

* work — пустой рабочий каталог. Здесь будут лежать ваши репозитории.
* files — заранее упакованные файлы для того, чтобы вы могли продолжить работать с учебными материалами на любом этапе. Если вы застрянете, просто скопируйте нужный урок в свою рабочую папку.

# 3. Создание проекта

### Цели

* Научиться создавать git репозиторий с нуля.

## 01Создайте страницу «Hello, World»

Начните работу в пустом рабочем каталоге (например Work, если вы скачали архив с предыдущего шага) с создания пустого каталога с именем «hello», затем войдите в него и создайте там файл с именем hello.html с таким содержанием.

#### ВЫПОЛНИТЕ:

mkdir hello

cd hello

touch hello.html

#### ФАЙЛ: hello.html

Hello, World

## 02Создайте репозиторий

Теперь у вас есть каталог с одним файлом. Чтобы создать git репозиторий из этого каталога, выполните команду git init.

#### ВЫПОЛНИТЕ:

git init

#### РЕЗУЛЬТАТ:

$ git init

Initialized empty Git repository in /Users/alex/Documents/Presentations/githowto/auto/hello/.git/

## 03Добавьте страницу в репозиторий

Теперь давайте добавим в репозиторий страницу «Hello, World».

#### ВЫПОЛНИТЕ:

git add hello.html

git commit -m "First Commit"

Вы увидите …

#### РЕЗУЛЬТАТ:

$ git add hello.html

$ git commit -m "First Commit"

[master (root-commit) 911e8c9] First Commit

1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 hello.html

# 4. Проверка состояния

### Цели

* Научиться проверять состояние репозитория

## 01Проверьте состояние репозитория

Используйте команду git status, чтобы проверить текущее состояние репозитория.

#### ВЫПОЛНИТЕ:

git status

Вы увидите

#### РЕЗУЛЬТАТ:

$ git status

# On branch master

nothing to commit (working directory clean)

Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

Мы будем использовать команду git status, чтобы продолжать отслеживать состояние репозитория и рабочего каталога.

# 5. Внесение изменений

### Цели

* Научиться отслеживать состояние рабочего каталога

## 01Измените страницу «Hello, World»

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

#### ФАЙЛ: hello.html

**<h1>**Hello, World!**</h1>**

## 02Проверьте состояние

Теперь проверьте состояние рабочего каталога.

#### ВЫПОЛНИТЕ:

git status

Вы увидите …

#### РЕЗУЛЬТАТ:

$ git status

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello.html

#

no changes added to commit (use "git add" and/or "git commit -a")

Первое, что нужно заметить, это то, что git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории.

Также обратите внимание на то, что сообщение о состоянии дает вам подсказку о том, что нужно делать дальше. Если вы хотите добавить эти изменения в репозиторий, используйте команду git add. В противном случае используйте команду git сheckoutдля отмены изменений.

# 6. Индексация изменений

### Цели

* Научиться индексировать изменения для последующих коммитов

## 01Добавьте изменения

Теперь дайте команду git проиндексировать изменения. Проверьте состояние

#### ВЫПОЛНИТЕ:

git add hello.html

git status

Вы увидите…

#### РЕЗУЛЬТАТ:

$ git add hello.html

$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello.html

#

Изменения файла hello.html были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не перманентно (читай, навсегда) записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.

Если вы решили, что не хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git reset можно снять индексацию этих изменений.

# 7. Индексация и коммит

Отдельный шаг индексации в git позволяет вам продолжать вносить изменения в рабочий каталог, а затем, в момент, когда вы захотите взаимодействовать с версионным контролем, git позволит записать изменения в малых коммитах, которые фиксируют то, что вы сделали.

Предположим, что вы отредактировали три файла (a.html, b.html, и c.html). Теперь вы хотите закоммитить все изменения, при этом чтобы изменения в a.html и b.html были одним коммитом, в то время как изменения в c.html логически не связаны с первыми двумя файлами и должны идти отдельным коммитом.

В теории, вы можете сделать следующее:

git add a.html

git add b.html

git commit -m "Changes for a and b"

git add c.html

git commit -m "Unrelated change to c"

Разделяя индексацию и коммит, вы имеете возможность с легкостью настроить, что идет в какой коммит.

# 8. Коммит изменений

### Цели

* Научиться коммитить изменения в репозиторий

## 01Закоммитьте изменения

Достаточно об индексации. Давайте сделаем коммит того, что мы проиндексировали, в репозиторий.

Когда вы ранее использовали git commit для коммита первоначальной версии файла hello.html в репозиторий, вы включили метку -m, которая делает комментарий в командной строке. Команда commit позволит вам интерактивно редактировать комментарии для коммита. Теперь давайте это проверим.

Если вы опустите метку -m из командной строки, git перенесет вас в редактор по вашему выбору. Редактор выбирается из следующего списка (в порядке приоритета):

* переменная среды GIT\_EDITOR
* параметр конфигурации core.editor
* переменная среды VISUAL
* переменная среды EDITOR

У меня переменная EDITOR установлена в emacsclient (доступен для Linux и Mac).

Сделайте коммит сейчас и проверьте состояние.

#### ВЫПОЛНИТЕ:

git commit

Вы увидите в вашем редакторе:

#### РЕЗУЛЬТАТ:

|

# Please enter the commit message for your changes. Lines starting

# with '#' will be ignored, and an empty message aborts the commit.

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello.html

#

В первой строке введите комментарий: «Added h1 tag». Сохраните файл и выйдите из редактора (для этого в редакторе по-умолчанию (Vim) вам нужно нажать клавишу ESC, ввести :wq и нажать Enter). Вы увидите…

#### РЕЗУЛЬТАТ:

git commit

Waiting for Emacs...

[master 569aa96] Added h1 tag

1 files changed, 1 insertions(+), 1 deletions(-)

Строка «Waiting for Emacs…» получена из программы emacsclient, которая посылает файл в запущенную программу emacs и ждет его закрытия. Остальные выходные данные – стандартные коммит-сообщения.

## 02Проверьте состояние

В конце давайте еще раз проверим состояние.

#### ВЫПОЛНИТЕ:

git status

Вы увидите…

#### РЕЗУЛЬТАТ:

$ git status

# On branch master

nothing to commit (working directory clean)

Рабочий каталог чистый, можете продолжить работу.

[9. Изменения, а не файлы](https://githowto.com/ru/changes_not_files)

[7. Индексация и коммит](https://githowto.com/ru/staging_and_committing)