1. **Установка Git**

Откройте у себя в браузере сайт с последней версией Git’а:

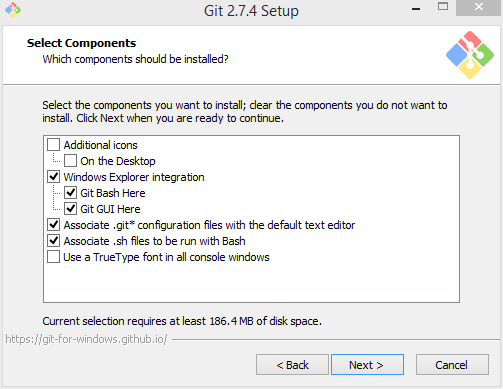
<https://git-scm.com/downloads>

Скачайте версию Git’а в зависимости от вашей операционной системы.

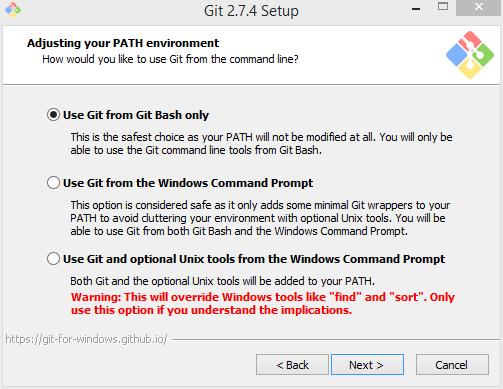
Продолжите инсталяцию программы как показано на следующих картинках.

Нажмайте “Next” пока не нужно будет что-то выбирать.

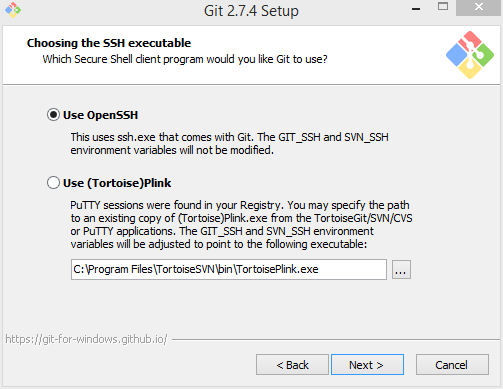
Не убирайте “Windows Explorer Integration” на этом шаге:



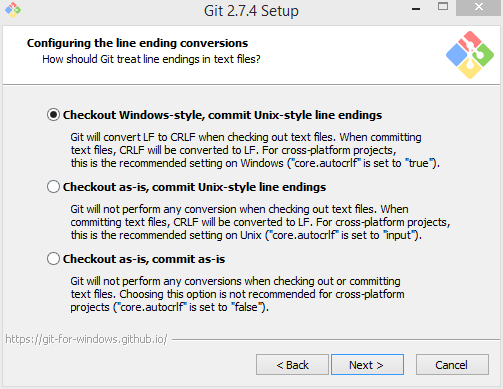
Также оставьте пункт “Use Git from Git Bash only”:



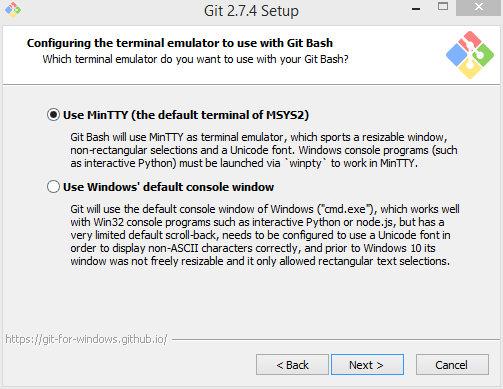
В следующем окне выберите “Use OpenSSH”:



Выберите “Checkout Windows-style, commit Unix-style line endings”, если вы работаете в Windows:



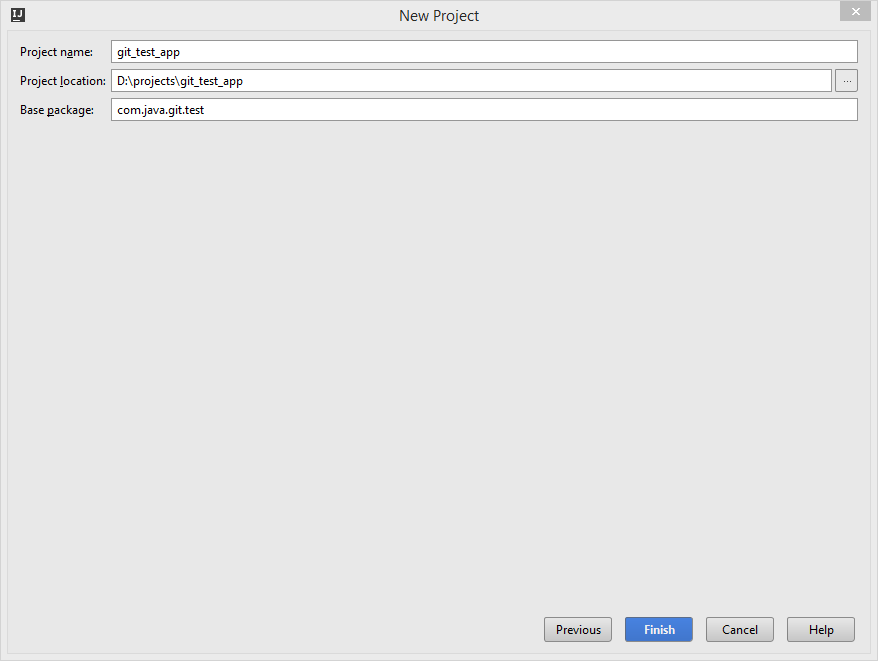
В следующем окне выберите “Use MinTTY” так как это лучше стандартной Windows консоли:



Оставьте всё как есть в последнем окне.

1. **Инициализация локального репозитория**

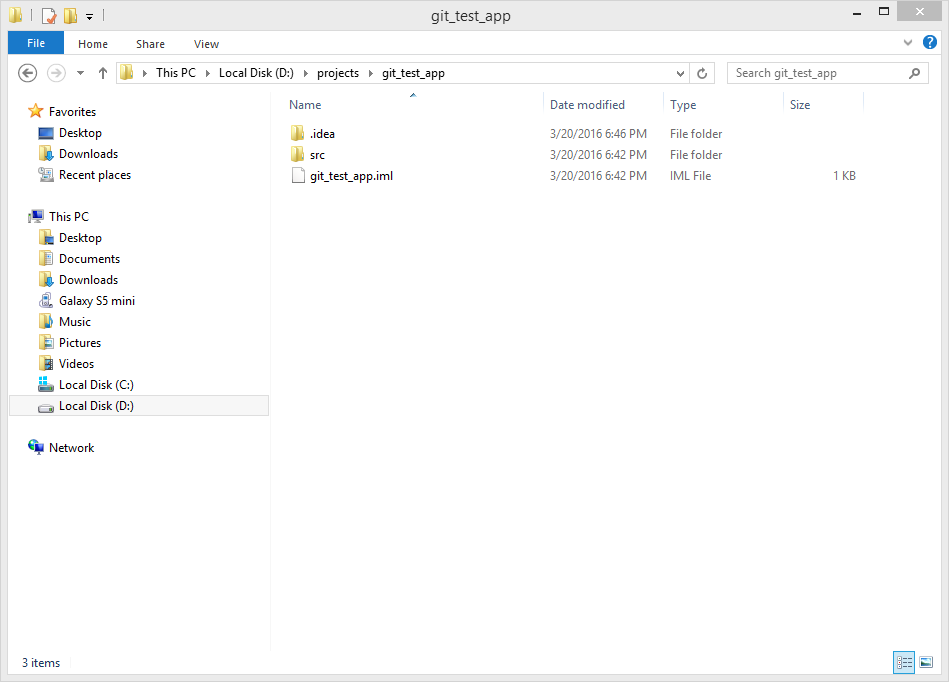
Создайте проект в IntelliJ IDEA. Это может быть простой консольный проект. Я создал консольный проект и назвал его “git\_test\_app”.



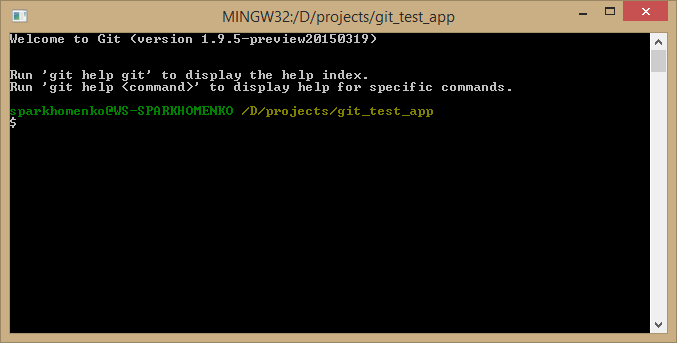
Всего один файл “Main.java” внутри проекта со следующим содержимым:

**package** com.java.git.test;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 System.***out***.println(**"Hello Git"**);  
 }  
}

Перейдите в папку проекта на файловой системе. Вы можете это сделать, кликнув на самую первую папку (“git\_test\_app”) в окне проекта в IDE и выбрав меню “Show in Explorer”. Войдите в эту папку.



Выполните клик правой кнопкой мыши на свободном месте в окне и выберите “Git Bash” из контекстного меню. Вы увидите следующее окно:



Теперь мы готовы инициализировать репозиторий.

Введите команду:

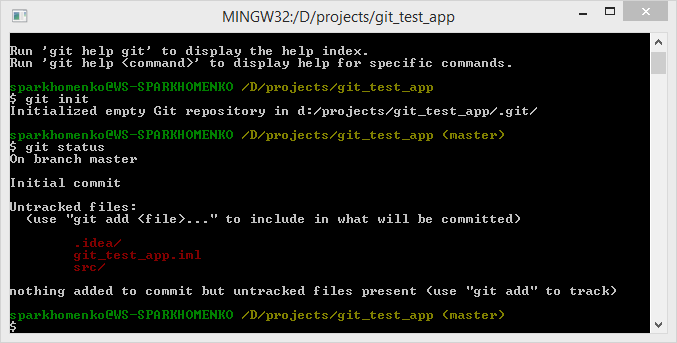
git init

После этого Git создаст папку “.git” (это скрытая папка, включите “показывать скрытые файлы” если вы не видите ее). Это значит что весь контент папки git\_test\_app стал локальным репозиторием Git.

Чтобы увидеть текущее состояние вашего репозитория введите команду:

git status

Вы увидите несколько файлов красного цвета:



Красный цвет значит, что все эти файлы находятся в неотслеживаемом состоянии, т.е. файлы, которые не войдут в следующих коммит (в базу данных Git).

Также мы не собираемся отслеживать папку “.idea/” и файл “git\_test\_app.iml” потому что они никак не относятся к нашей программе и они нам не нужны. Давайте создадим файл “.gitignore” и запишем все ненужные файлы и папки сюда.

Введите команду:

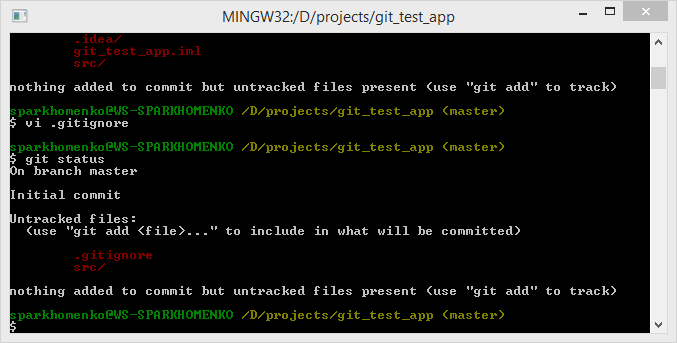
vi .gitignore

Это переместит вас в текстовый редактор “vi”, чтобы начать вводить текст нажмите кнопку “i”. Добавьте две строки в этот файл:

.idea/  
\*.iml

Нажмите кнопку на клавиатуре “Esc” и введите “wq” что значит write and quit(сохранить изменения и выйти). Нажмите “Enter” после этого. Файл “.gitignore” будет создан.

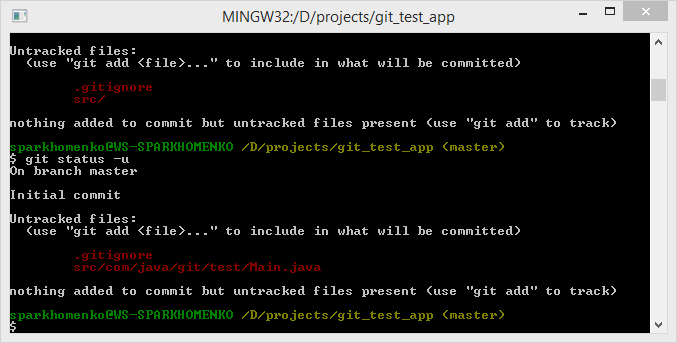
Выполните команду “git status” опять. Вы должны увидеть, что все ненужные файлы больше не отслеживаются Git’ом:



Попробуйте ввести:

git status -u

Вы увидите полные пути ко всем файлам, а не только папки, выполнив эту команду:



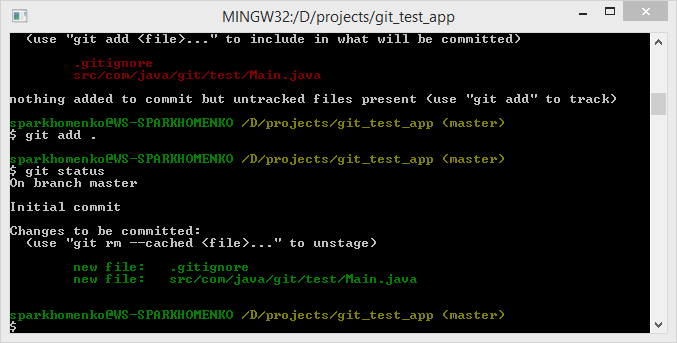
1. **Сохранение изменений в локальном репозитории**

Для добавления файлов или изменений в отслеживаемую область (такие файлы будут добавлены в базу данных Git’а при следующем коммите) наберите:

git add .

“.” значит, что все файлы будут добавлены, вы можете указывать конкретные имена файлов, если хотите.

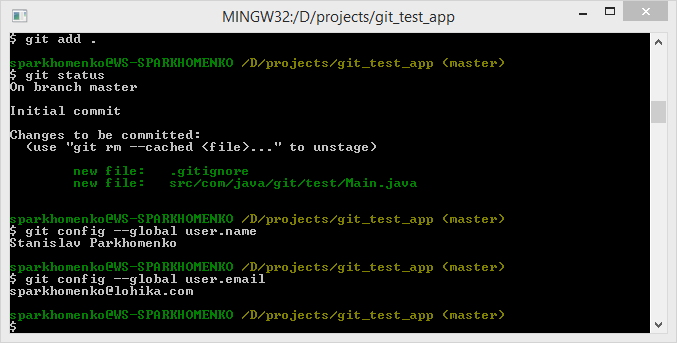
Выполните команду “git status”. Все файлы теперь в зеленом цвете (войдут в следующий коммит):



Перед тем как сделать первый коммит, мы должны добавить свои имя и почту, эта информация будет использована в описании коммита:

git config --global user.name “Stanislav Parkhomenko”  
git config --global user.email “stanislav.parkhomenko.gmail.com”

Вы можете запустить такую же команду без указания имени или почты, чтобы проверить всё ли вы ввели корректно:

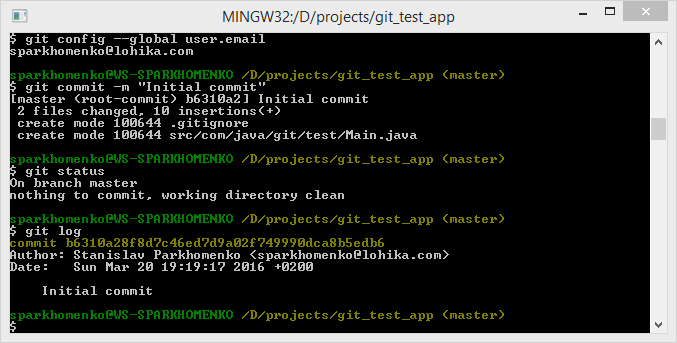


Параметр “--global” значит, что эти параметры будут добавлены для всех последующих проектов, в которых вы проинициализируете репозиторий Git.

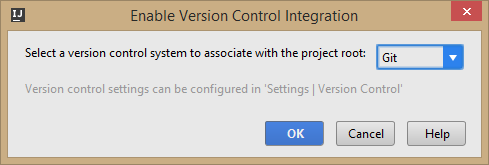
Введите следующую команду:

git commit -m “Initial commit”

После этого наши изменения будут добавлены в репозиторий (базу данных) Git. Введите команду “git log”, чтобы увидеть, что у вас появилась одна запись в истории коммитов:



Войдите в IDE. В меню “VCS - Enable Version Control integration” выберите “Git” and press “OK”.



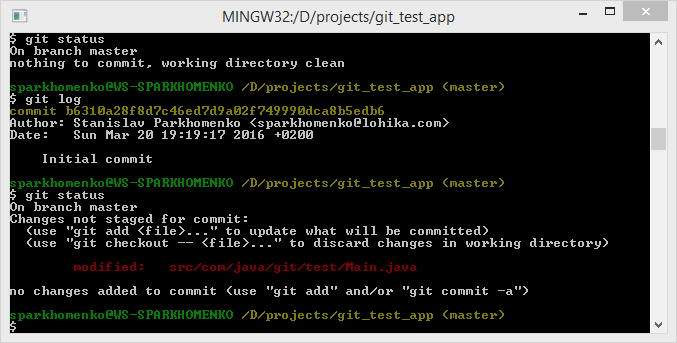
Вы должны увидеть вашу текущую и единственную ветку в нижнем правом углу:



Давайте добавим еще немного кода в наш проект. Я добавлю цикл, который выводит на экран несколько чисел (конкретный код неважен, можно добавить всё что угодно на ваш выбор):

**package** com.java.git.test;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 System.***out***.println(**"Hello Git"**);  
  
 **for** (**int** i = 1; i <= 10; i++) {  
 System.***out***.println(i);  
 }  
 }  
}

Выполните команду “git status”. Вы должны увидеть, что файл “Main.java” опять стал красным:



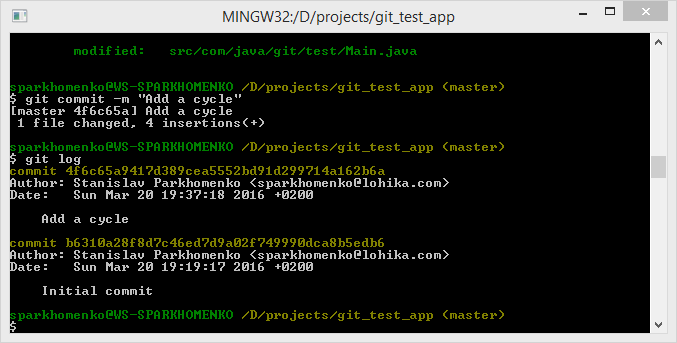
Добавьте ваши изменения для следующего коммита

git add .

Выполните коммит

git commit -m “Add a cycle”

Теперь вы должны видеть 2 коммита в истории (в базе данных Git):



1. **Сохранение изменений в удаленный репозиторий**

Откройте ваш github аккаунт и создайте новый репозиторий. Я создал “git\_app\_test”.

Мой репозиторий имеет следующую ссылку:

<https://github.com/parkhomenko/git_app_test.git>

У вас должо быть тоже что-то вроде этого.

Теперь давайте добавим информацию об удаленном репозитории в нашем локальном:

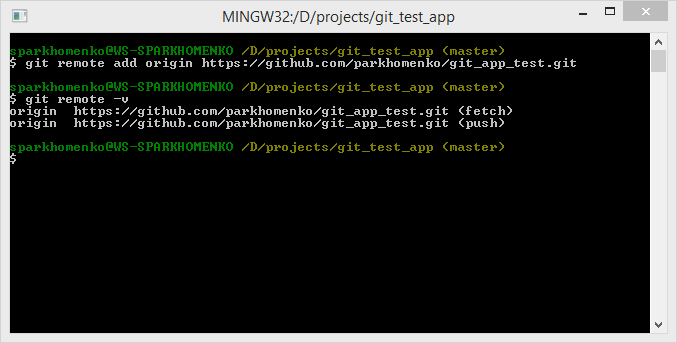
git remote add origin <https://github.com/parkhomenko/git_app_test.git>

“origin” алиас для ссылки к нашему удаленному репозиторию: вы ведь не хотите писать каждый раз всю ссылку, когда она вам будет нужна

Проверьте, что ссылка была добавлена успешно набрав команду:

git remote -v

Вы должны получить список ваших удаленных репозиториев:



Теперь всё готово для того, чтобы загрузить изменения из локального в удаленный репозиторий:

git push origin master

“origin” имя вашего удаленного репозитория

“master” ветка, которую вы собираетесь сохранить в удаленном репозитории

Git попросит вас ввести имя пользователя и пароль для github’а, введите их и вы сможете увидеть ваши изменения на github:

