IT-Academy

Git

Рассматриваемые вопросы

IT-Academy

- Системе контроля версий
- Git
- GitHub

Система контроля версий

IT-Academy

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

Вы, как программист, хотите сохранить каждую версию кода в файле, в этом нам и помогает **система контроля версий** (далее **СКВ**). Она позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему, кто поставил задачу и когда и многое другое. Использование СКВ также значит в целом, что, если вы сломали что-то или потеряли файлы, вы спокойно можете всё исправить.

Типы СКВ

IT-Academy

Различают 3 типа СКВ:

1) Локальные системы контроля версий
2) Централизованные системы контроля версий
3) Распределённые системы контроля версий

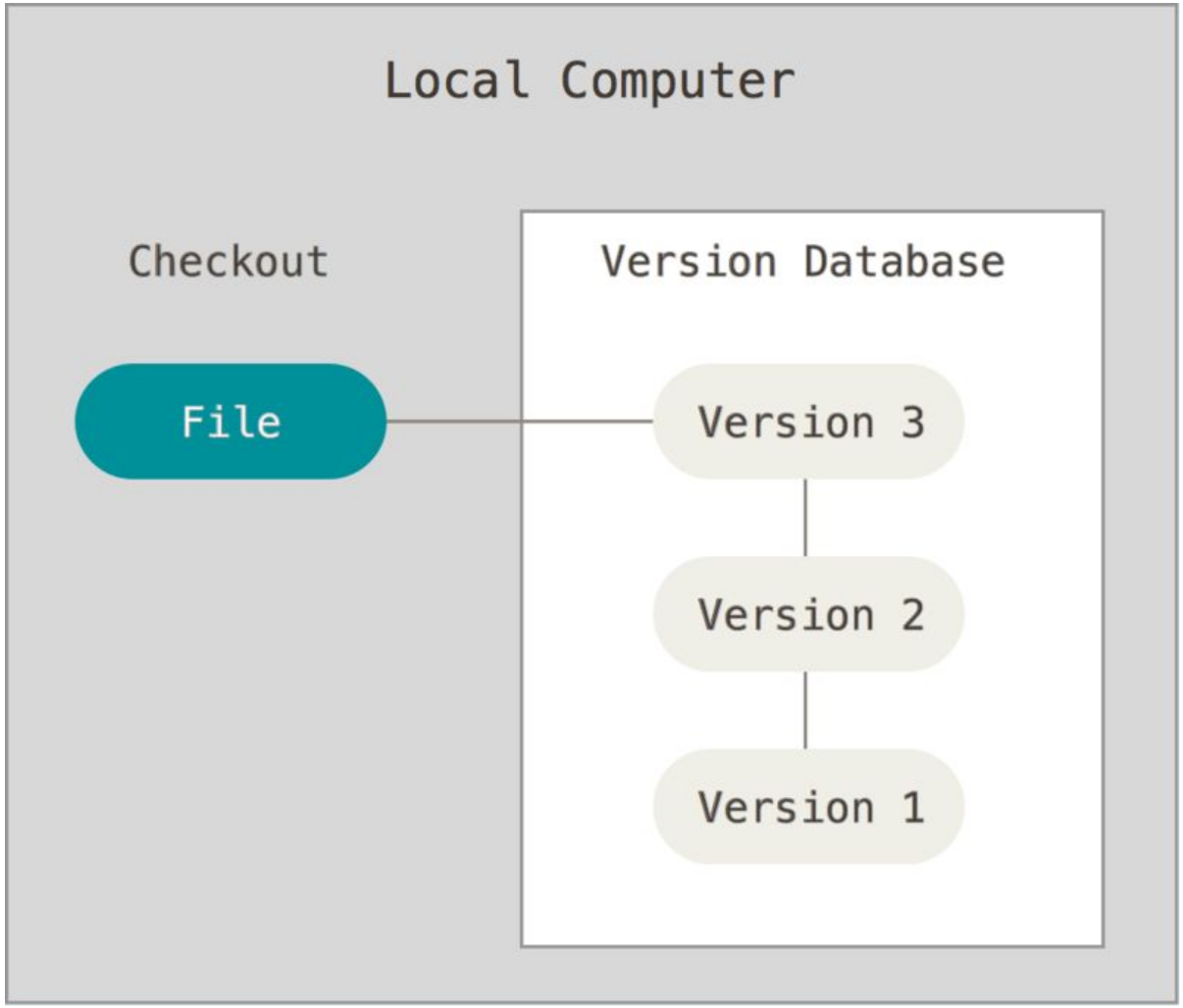
Локальные СКВ

IT-Academy

Многие копируют файлов в отдельную директорию. Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Для того, чтобы решить эту проблему, программисты давным-давно разработали локальные СКВ с простой базой данных, которая хранит записи о всех изменениях в файлах, осуществляя тем самым контроль ревизий.

Локальные СКВ



Централизованные СКВ

IT-Academy

Следующая серьёзная проблема — необходимость взаимодействовать с другими разработчиками. Для ее решения были разработаны централизованные системы контроля версий (ЦСКВ). Такие системы используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Применение ЦСКВ являлось стандартом на протяжении многих лет.

Плюсы очевидны, особенно перед локальными СКВ. Например, все разработчики проекта знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте.

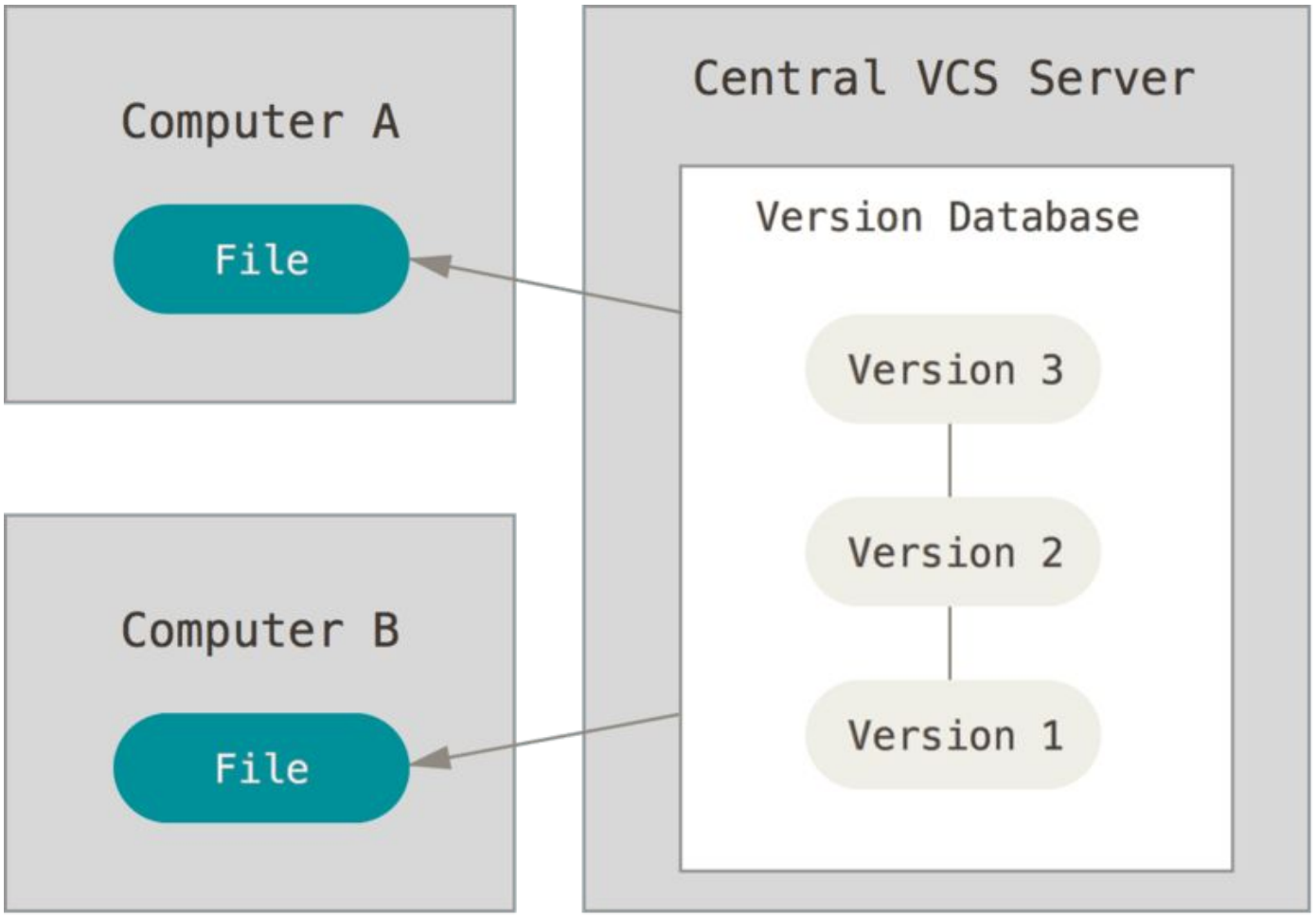
Централизованные СКВ

IT-Academy

Главный минус — это единая точка отказа, представленная централизованным сервером. Если сервер выйдет из строя на час, то в это времен никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

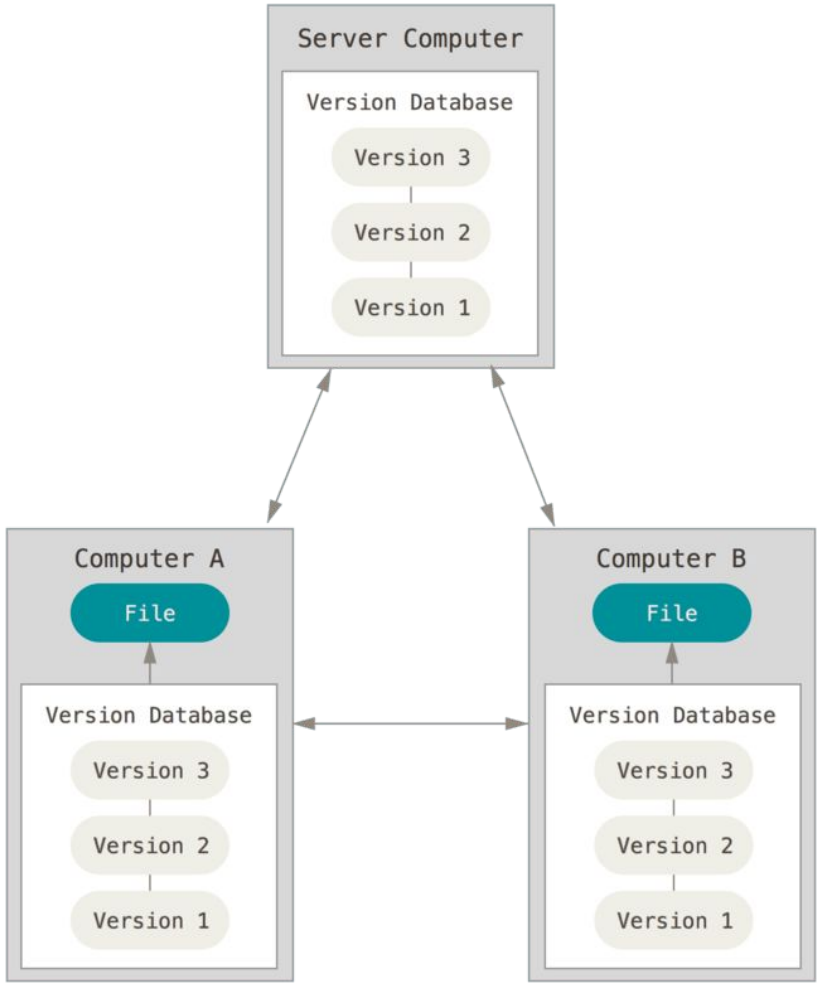
Централизованные СКВ



Распределённые СКВ

IT-Academy

В РСКВ клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) — они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.



Git

IT-Academy

Git — распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать совместно с другими разработчиками. Git известен своей скоростью, простым дизайном, поддержкой нелинейной разработки, полной децентрализацией и возможностью эффективно работать с большими проектами.

Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Преимущества Git

IT-Academy

- **Бесплатный и open-source**. Это значит, что его можно бесплатно скачать и вносить любые изменения в исходный код;
- **Небольшой и быстрый**. Он выполняет все операции локально, что увеличивает его скорость. Кроме того, Git локально сохраняет весь репозиторий в небольшой файл без потери качества данных;
- **Резервное копирование**. Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при использовании Git;
- **Простое ветвление**. В других СКВ создание веток— утомительная и трудоёмкая задача, так как весь код копируется в новую ветку. В Git управление ветками реализовано гораздо проще и эффективнее.

Git. Три состояния

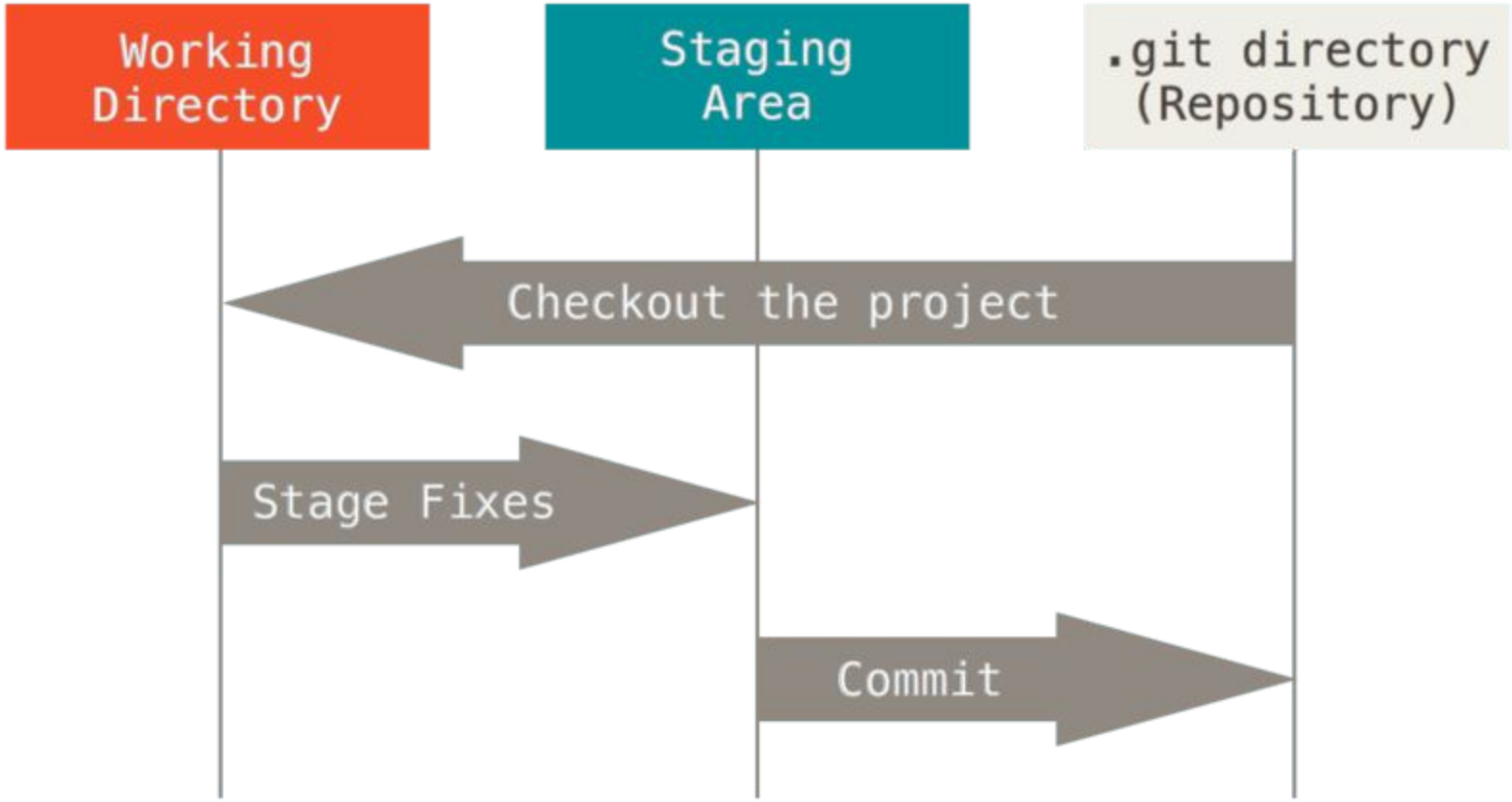
IT-Academy

У Git есть три основных состояния, в которых могут находиться ваши файлы:

- **изменённые (modified)** - файлы, которые поменялись, но ещё не были зафиксированы
- **подготовленные (staged)—** это изменённые файлы, отмеченные для включения в следующий коммит
- **зафиксированные (committed)** - файл уже сохранён в вашей локальной базе

Мы подошли к трём основным секциям проекта Git: **Git-директория** (Git directory), **рабочая директория** (working directory) и **область подготовленных файлов** (staging area).

Git. Три состояния



Git. Три состояния

IT-Academy

Git-директория — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git, и это та часть, которая копируется при клонировании репозитория с другого компьютера.

Рабочая директория является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать.

Область подготовленных файлов — это файл, обычно располагающийся в вашей Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, однако называть её stage-область также общепринято.

Базовый подход в работе с Git

IT-Academy

1. Вы изменяете файлы в вашей рабочей директории.
2. Вы выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит, добавляя тем самым снимки только этих изменений в область подготовленных файлов.
3. Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в вашу Git-директорию.

Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной.

Если версия файла изменена и добавлена в индекс, значит, она подготовлена.

И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

Первоначальная настройка Git

IT-Academy

Теперь, когда Git установлен в вашей системе, самое время настроить среду для работы с Git под себя. Это нужно сделать только один раз. Но, при необходимости, вы можете поменять их в любой момент.

Чтобы посмотреть все установленные настройки и узнать где именно они заданы, используйте команду:

```
git config --list --show-origin
```

Первое, что вам следует сделать после установки Git — указать ваше имя и адрес электронной почты. Это важно, потому что каждый коммит в Git содержит эту информацию, и она включена в коммиты, передаваемые вами, и не может быть далее изменена:

```
git config --global user.name "John Doe"

git config --global user.email johndoe@example.com
```

Проверка настроек. Git

IT-Academy

Если вы хотите проверить используемую конфигурацию, можете использовать команду **git config --list**, чтобы показать все настройки, которые Git найдёт:

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Также вы можете проверить значение конкретного ключа, выполнив git config <key>:

git config user.name

Задание

IT-Academy

Установить Git.

Выполнить команды:

git config --list --show-origin

git config --global user.name <name>

git config --global user.email <email>

GitHub

IT-Academy

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше. Обычно он используется вместе с Git и даёт разработчикам возможность сохранять их код онлайн, а затем взаимодействовать с другими разработчиками в разных проектах.

Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта. Цель GitHub — содействовать

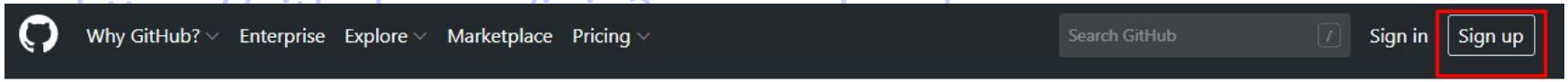
GitHub

IT-Academy

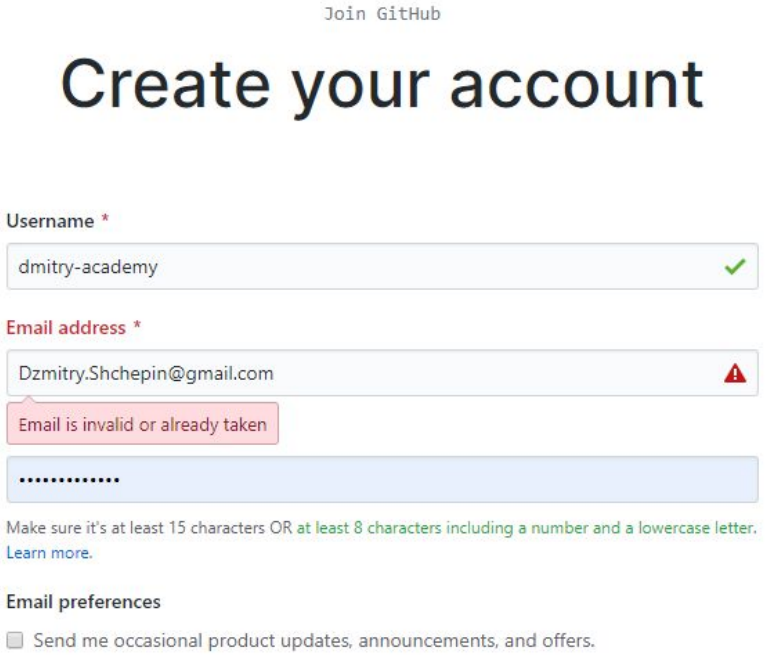
https://github.com/

Регистрация на GitHub

- Заходим по ссылке:

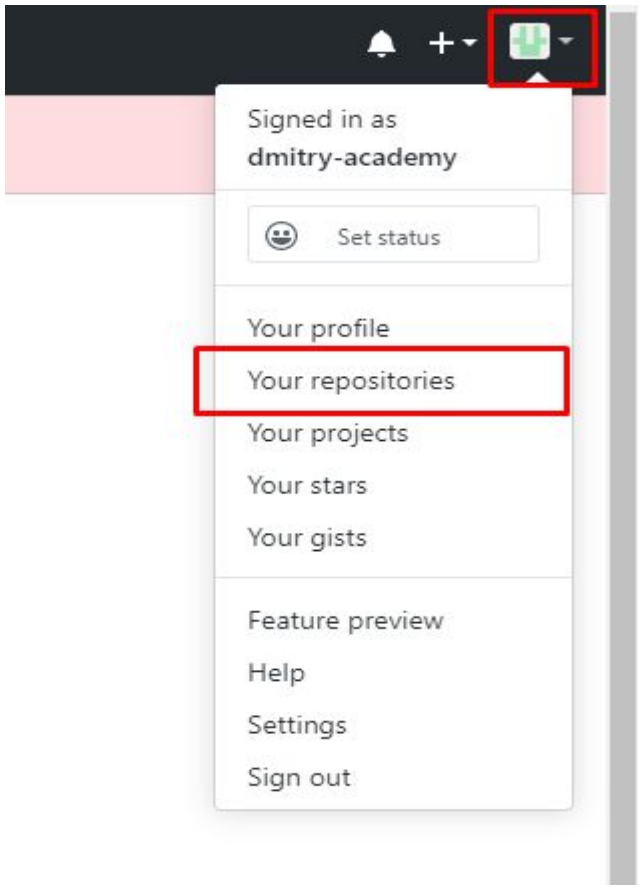


- Заполняем все поля.
- Подтверждаем.
- Выбираем Free план.



Регистрация на GitHub

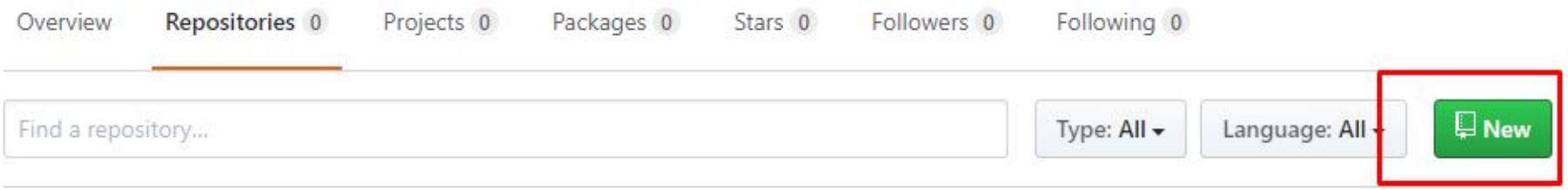
- Верифицируем аккаунт (на почту должно прийти письмо с подтверждением)
- Нажимаем на кнопку аккаунта в правом верхнем углу
- Нажимаем кнопку “Your repositories”



Создание репозитория

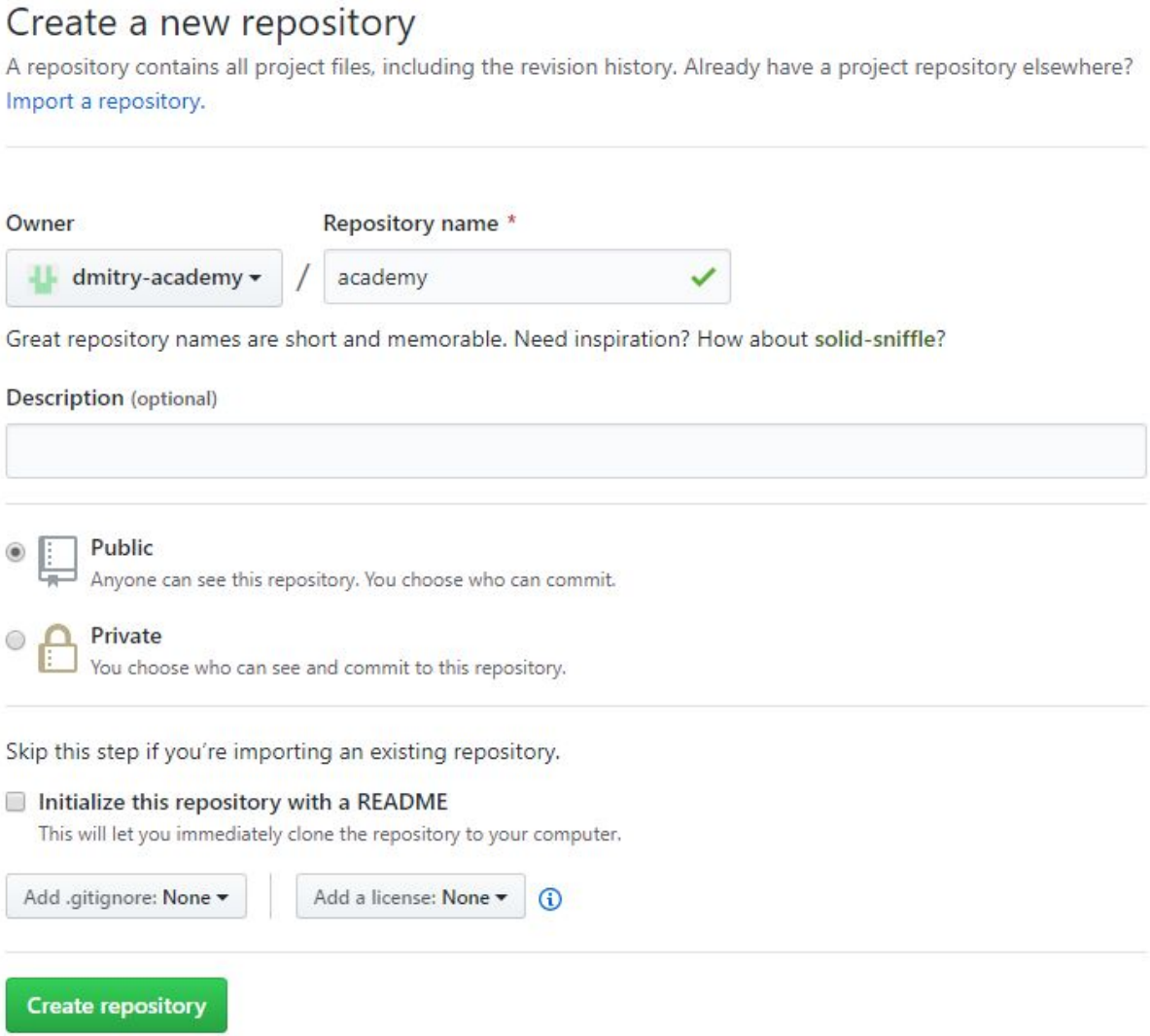
IT-Academy

Нажимаем New



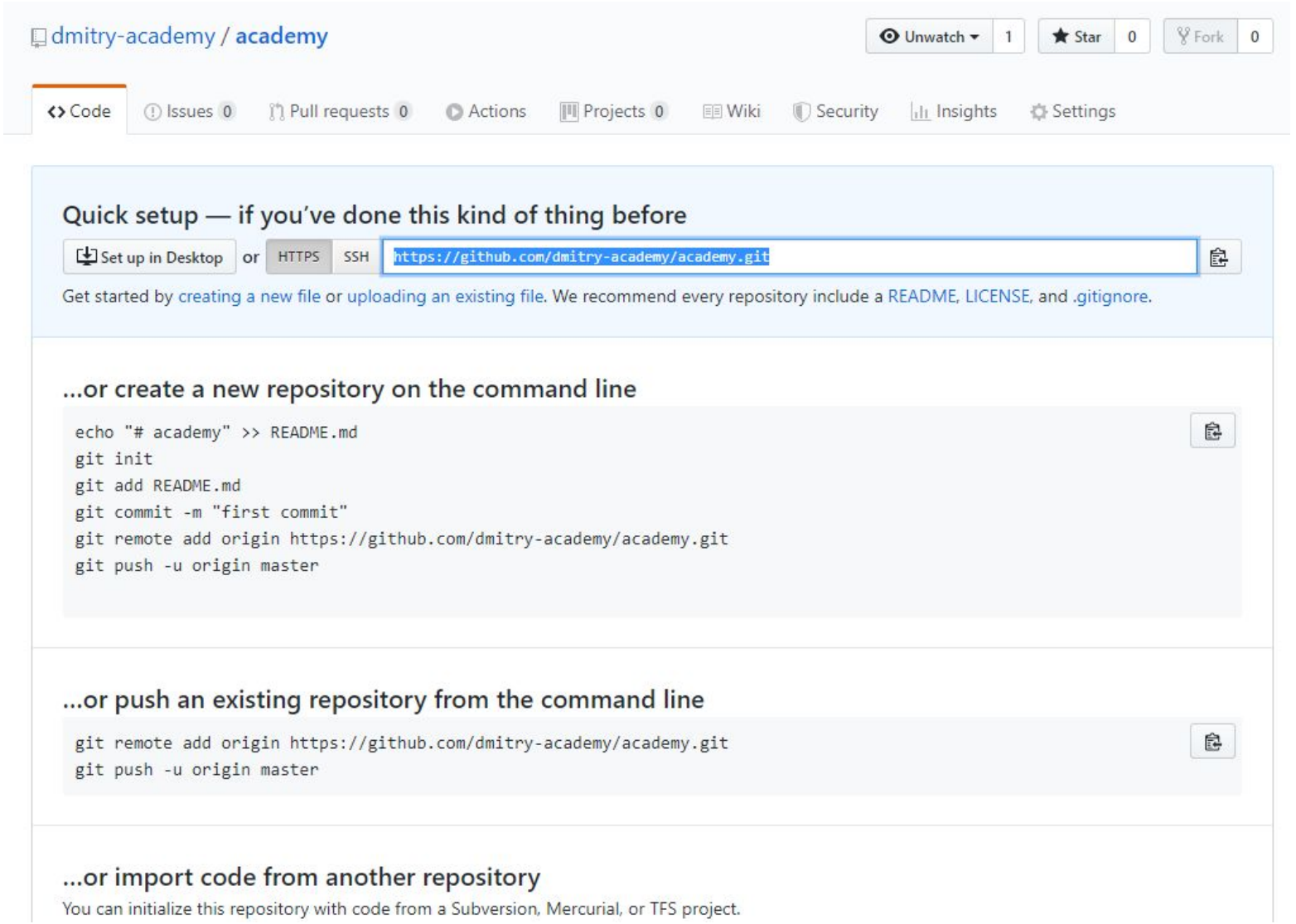
Заполняем форму создание репозитория

IT-Academy



Результат создания репозитория

IT-Academy



Создание проекта в Eclipse

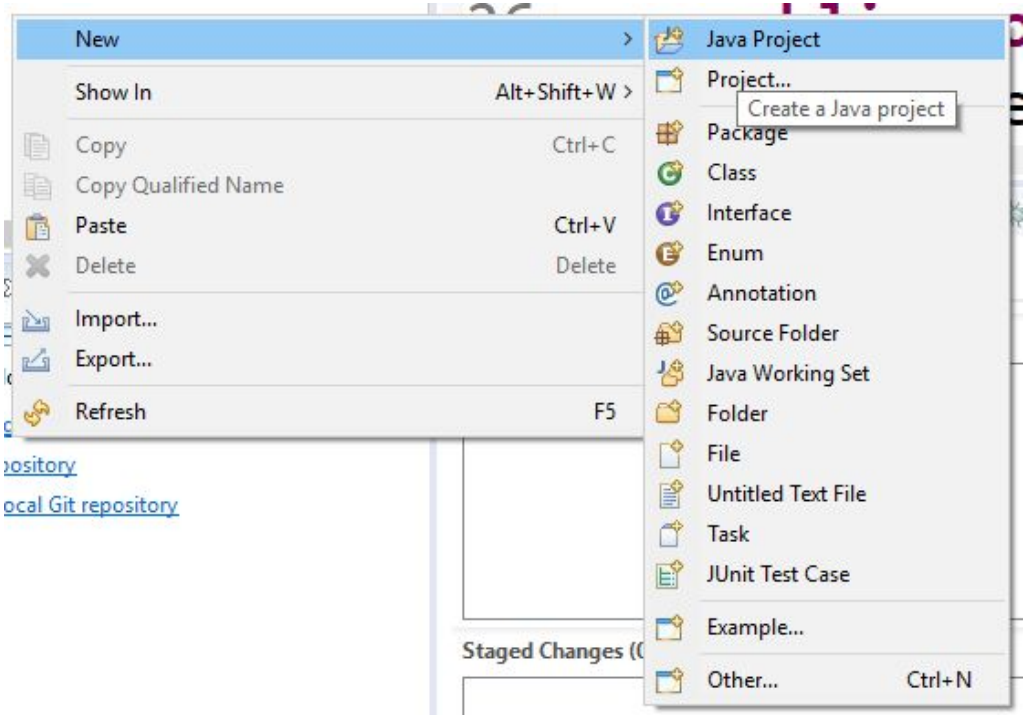
IT-Academy

Создаем проект в Eclipse.

Справа, в Package Explorer нажимаем правую кнопку мыши, выбираем **New -> Java Project**

Называем проект **academy.**

Жмем **Finish -> Don’t create Module**

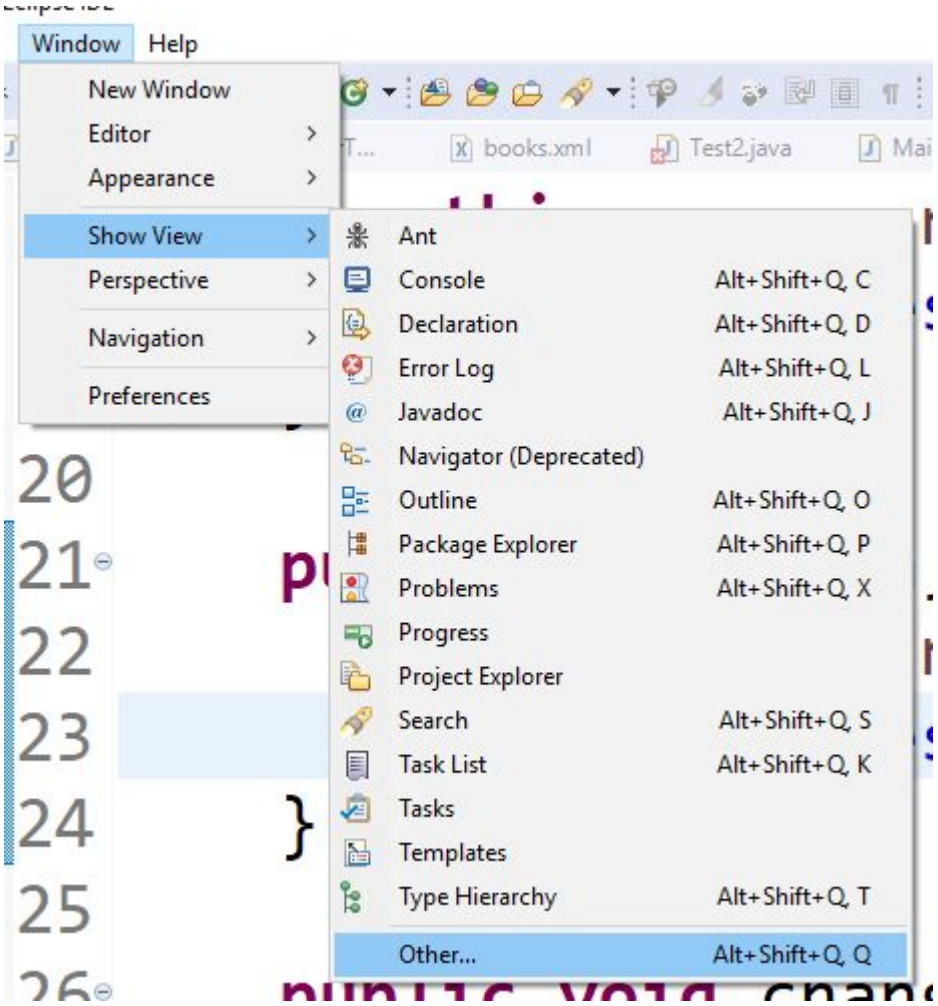


Добавление инструментов для Git

IT-Academy

Добавляем 2 View:

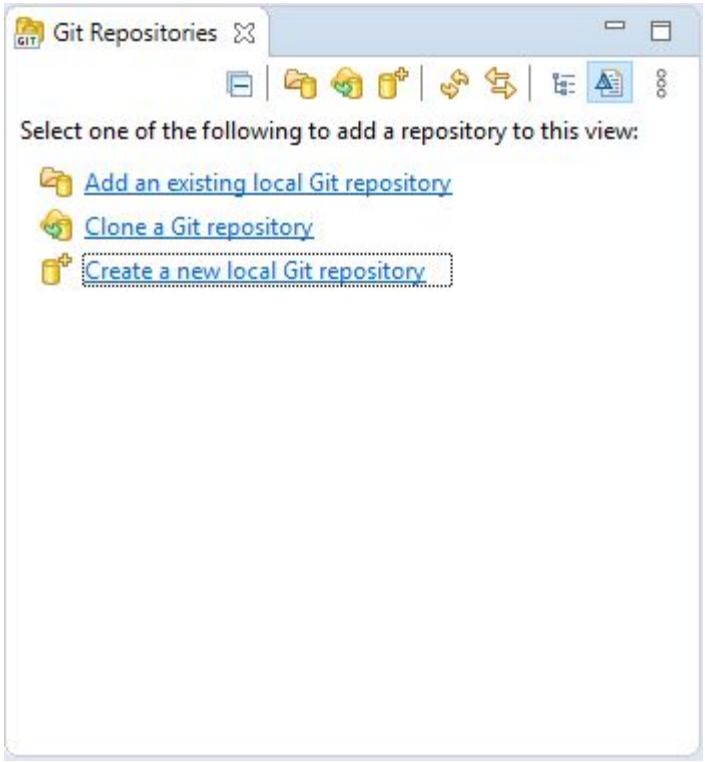
1) Git Repositories
2) Git Staging



Создание локального репозитория

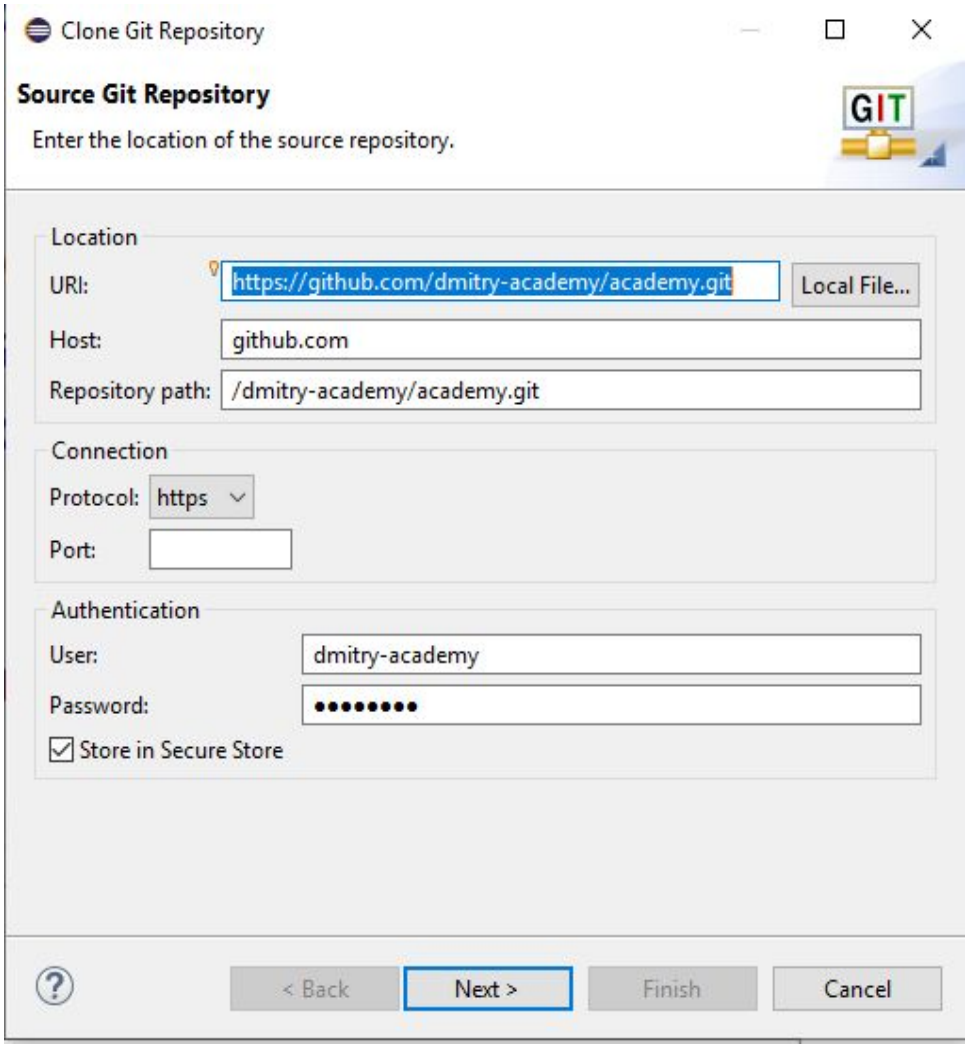
IT-Academy

Нажимаем Clone a new local Git Repository



Создание локального репозитория

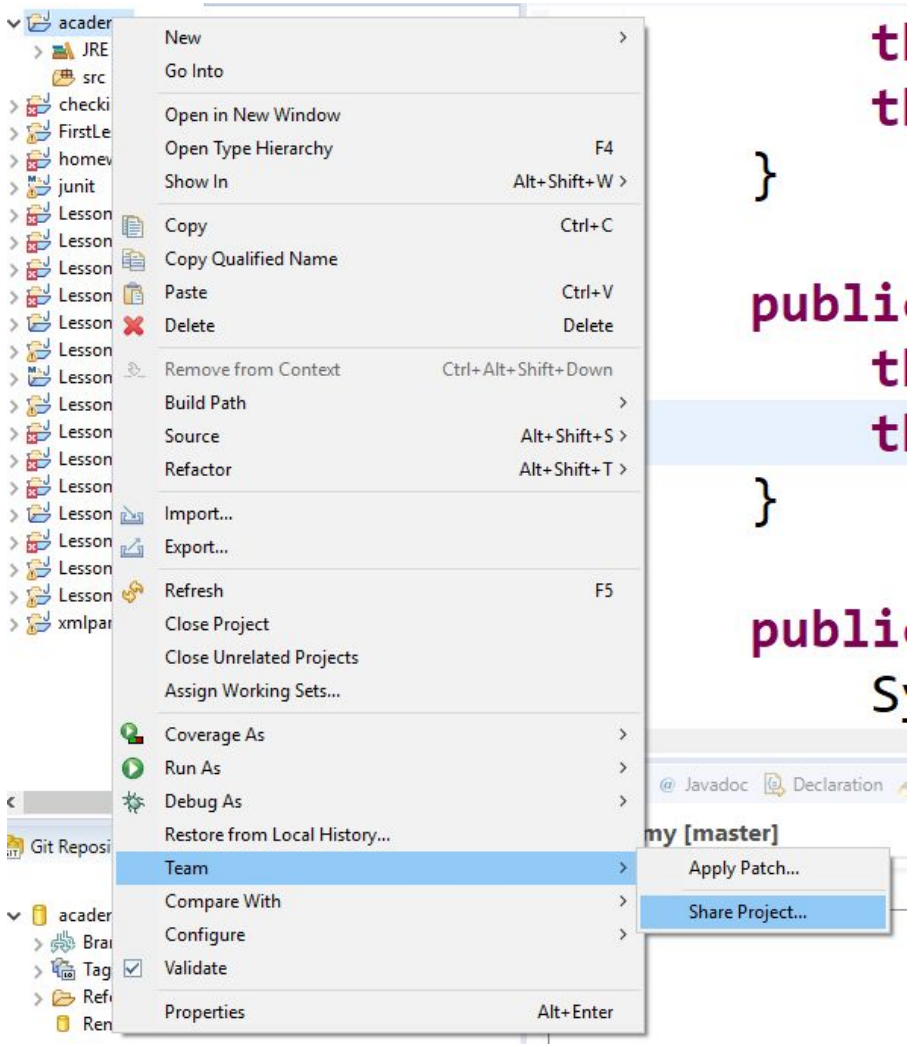
Копируем линку с Git и вставляем ее в поле URI. Заполняем поля User и Password вашими логином и паролем с GitHub.
Next -> Next -> Finish



Добавляем проект в локальный репозиторий

IT-Academy

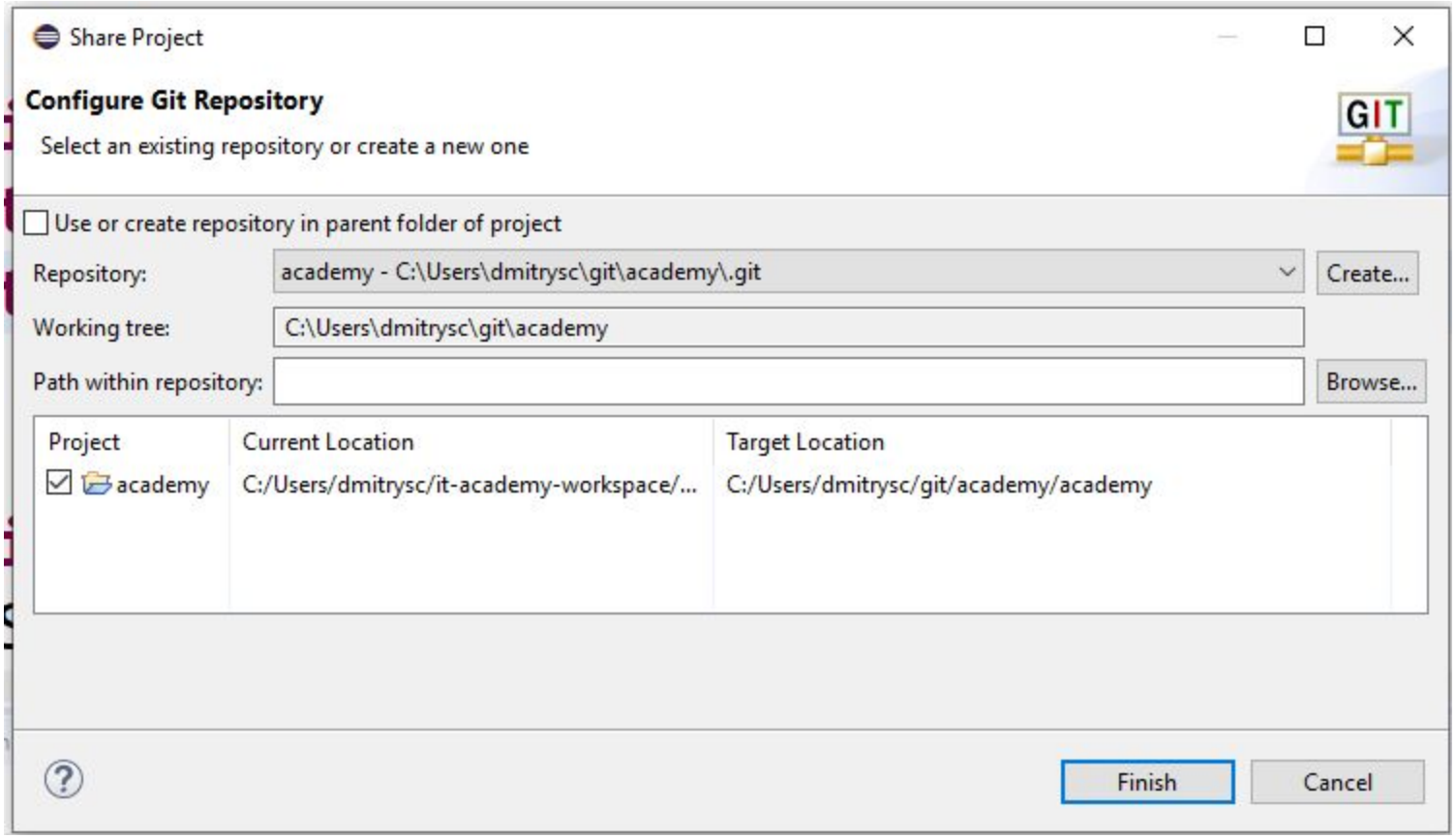
Кликаем правой кнопкой по проекту, выбираем:
Team -> Share Project…



Добавляем проект в локальный репозиторий

IT-Academy

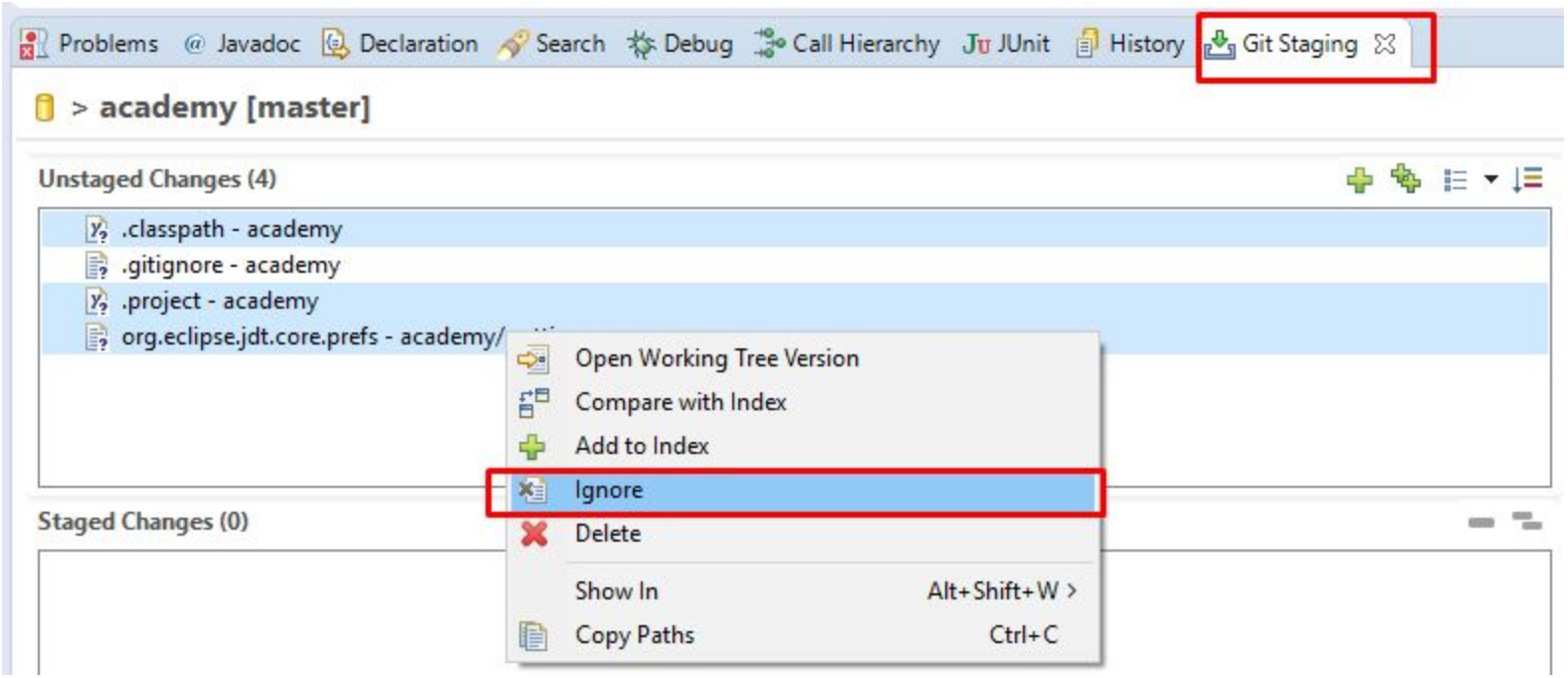
Выбираем созданный локальный репозиторий -> Finish



Добавляем файлы в gitignore

IT-Academy

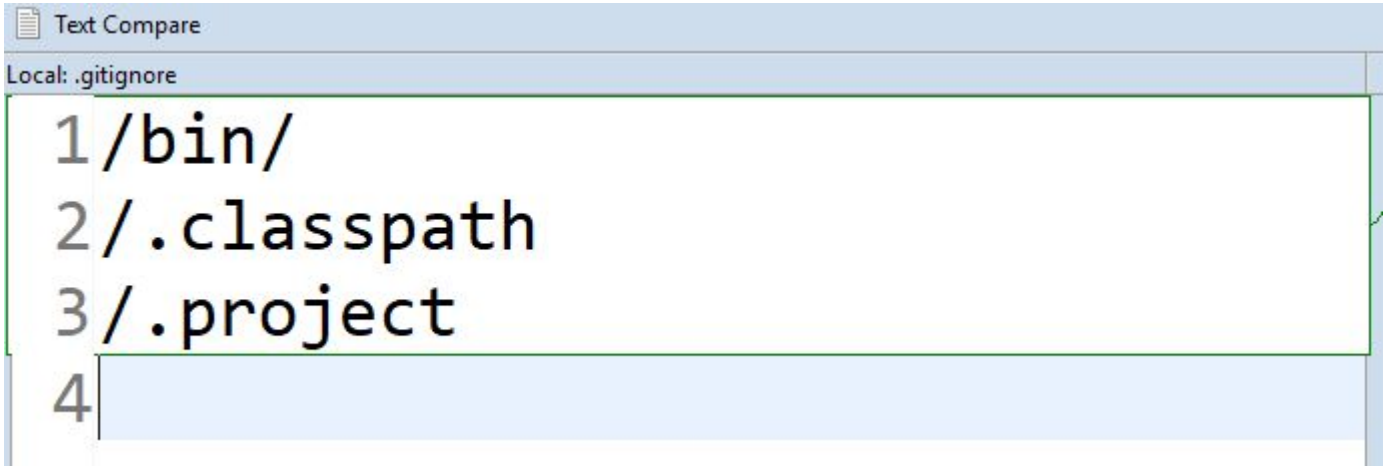
Теперь, при нажатии на проект у вас появился трекинг файлов в Git Staging tab.
Давайте добавим в ignore file следующие файлы:



Добавляем файлы в gitignore

IT-Academy

Если вы кликните 2 раза на файл .gitignore в Unstaged files списке, вы увидите:

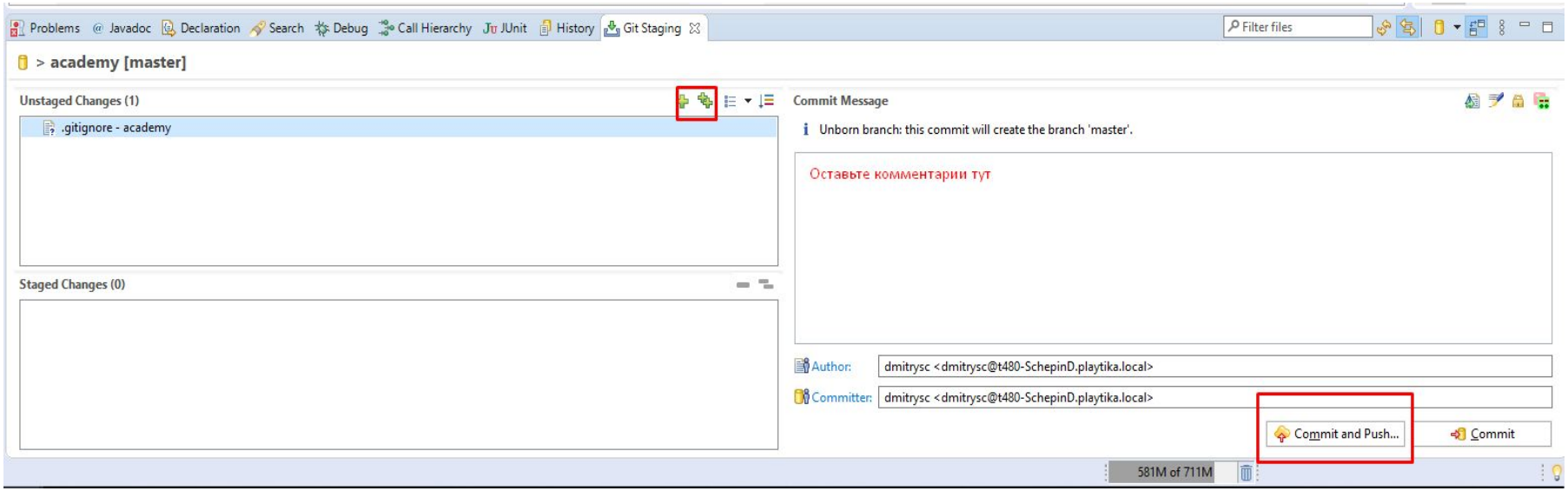


Эта конфигурация файла .gitignore – эти файлы будут игнорироваться гитом при изменении.

Добавляем файлы на GitHub

IT-Academy

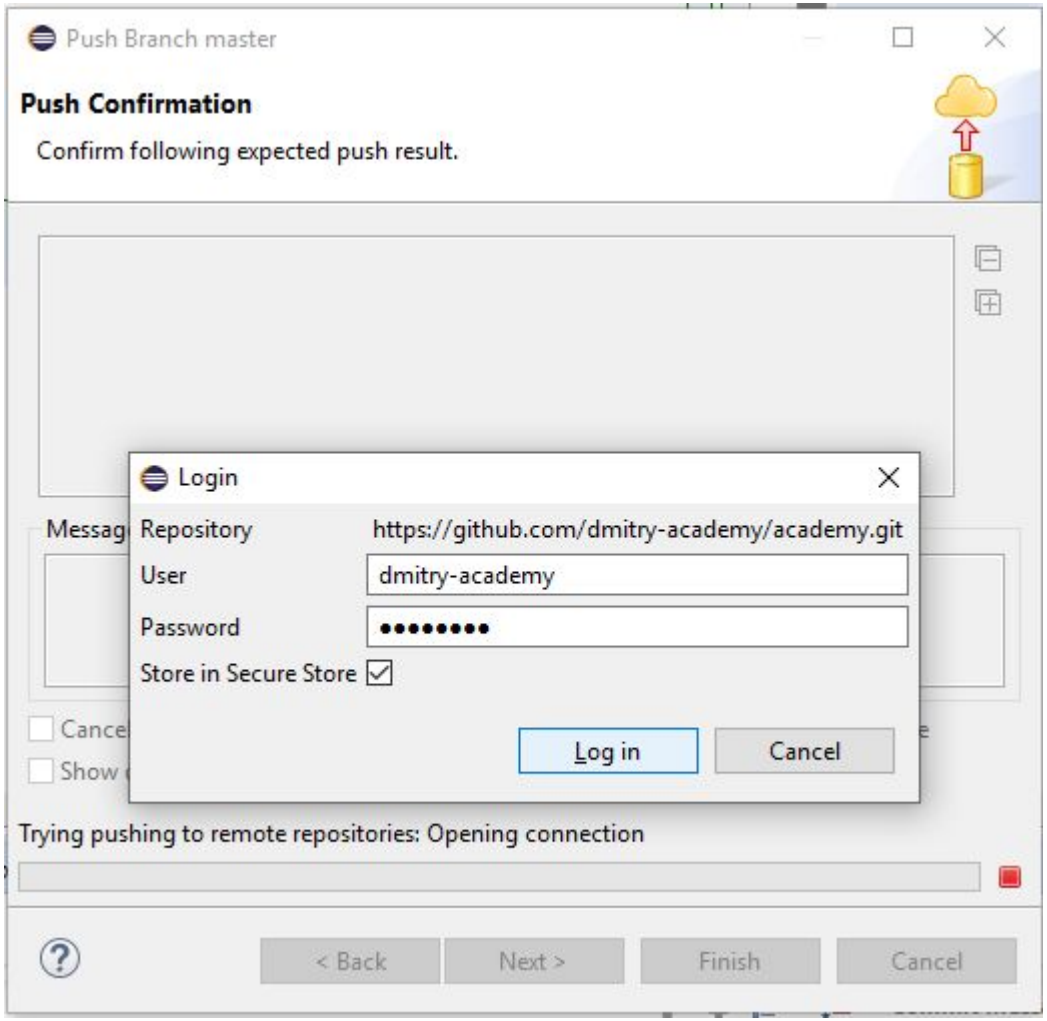
Жмем кнопку Добавить файлы, они переместятся в Staged Changes лист. Добавим комментарий “First commit” справа и нажмем **Commit and Push…**



Добавляем файлы на GitHub

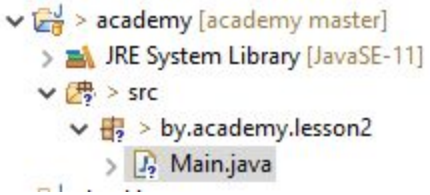
IT-Academy

Жмем Next -> вводим логин/пароль с GitHub. Store in Secure Store -> checked -> Log In -> Finish



Задание

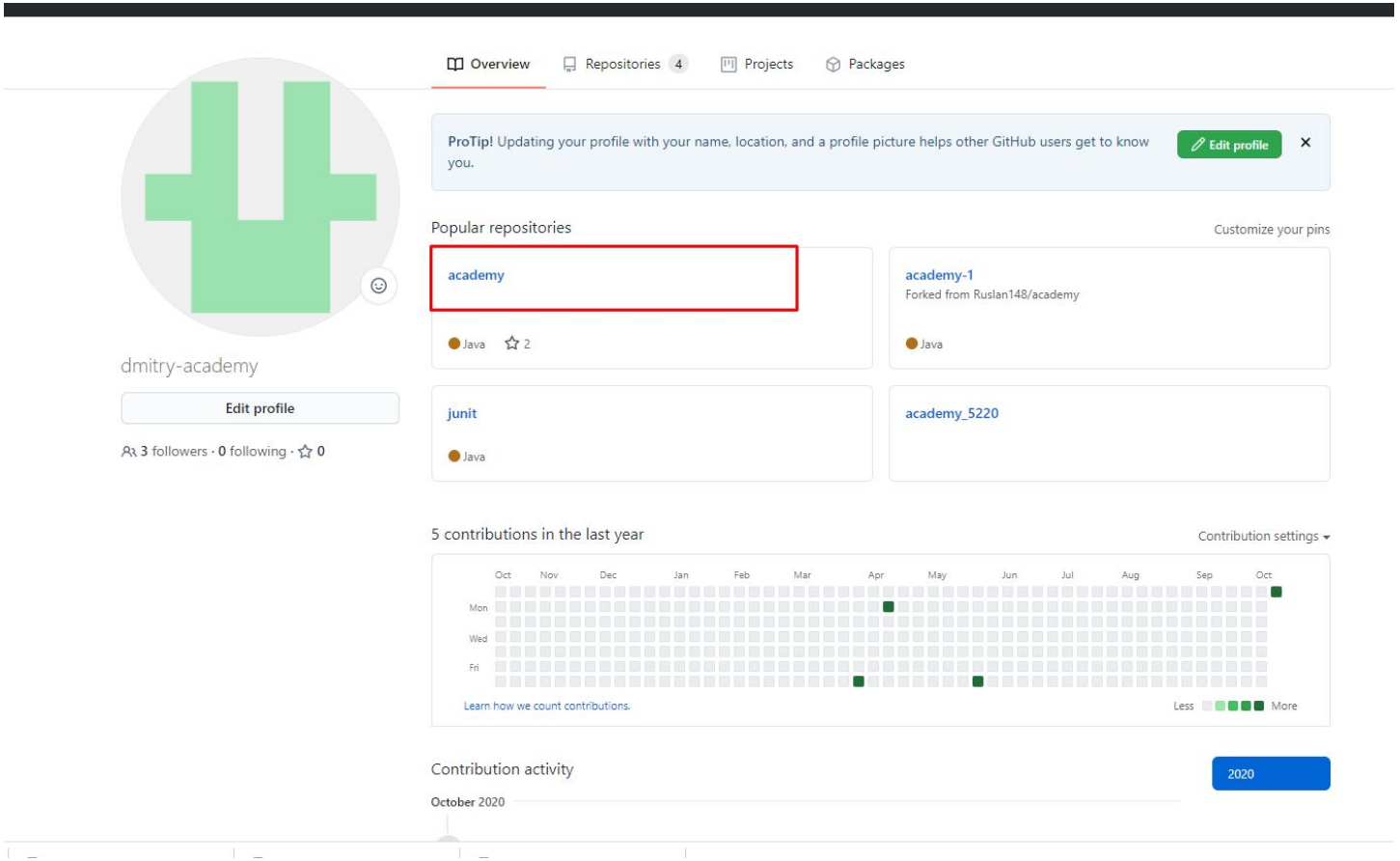
1) Создать пакет by.academy.lesson2 в проекте academy
2) Добавить класс Main
3) В классе Main добавить вывод на консоль: “Second commit!”
4) Отправить все изменения на GitHub.



Скопировать репозиторий в eclipse

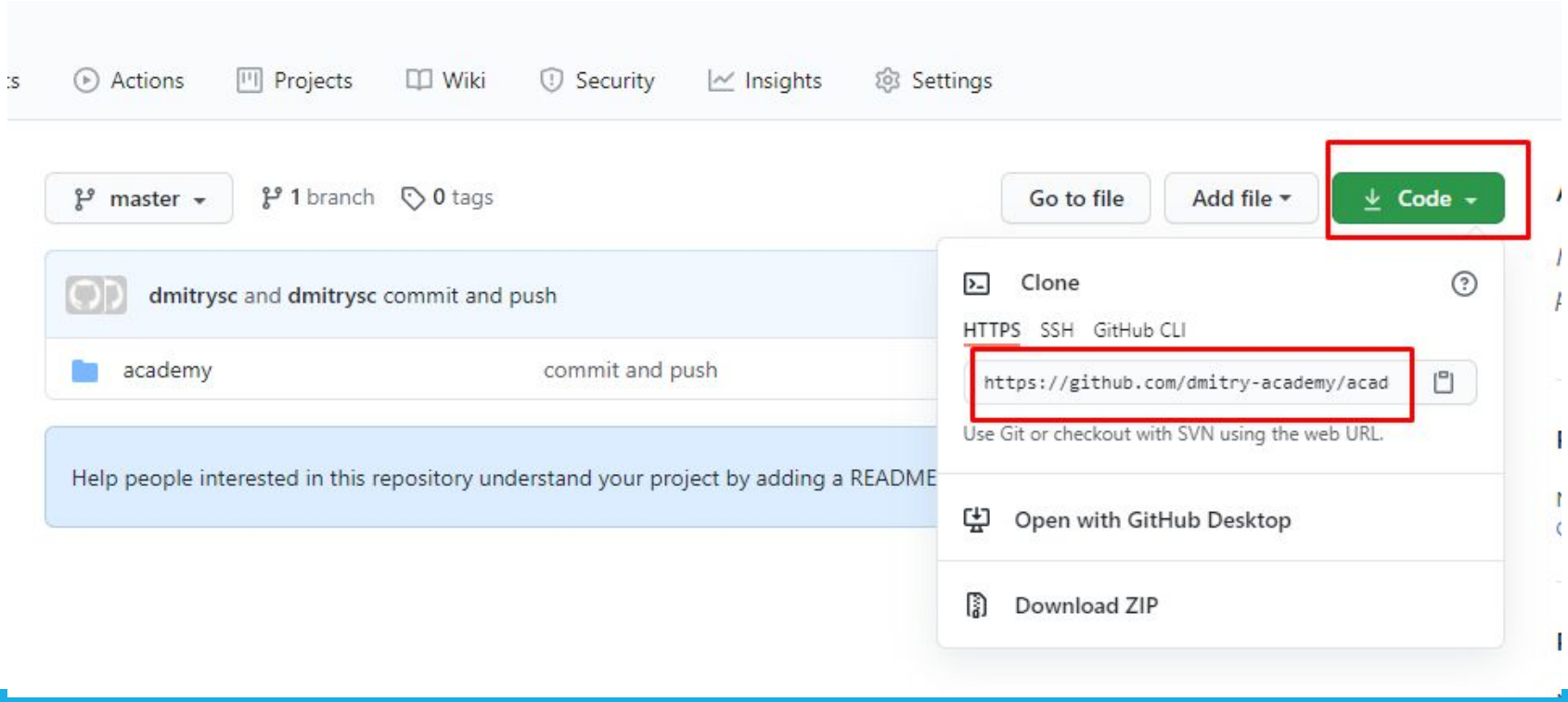
IT-Academy

Выбираем репозиторий



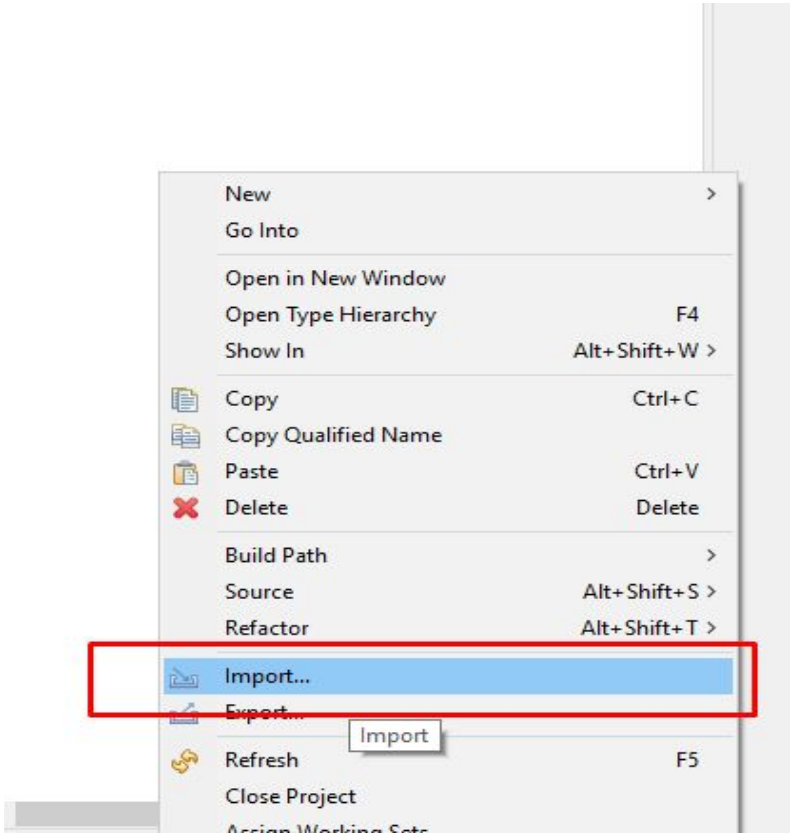
Скопировать репозиторий в eclipse

1) Нажимаем Code
2) Копируем ссылку (HTTPS)



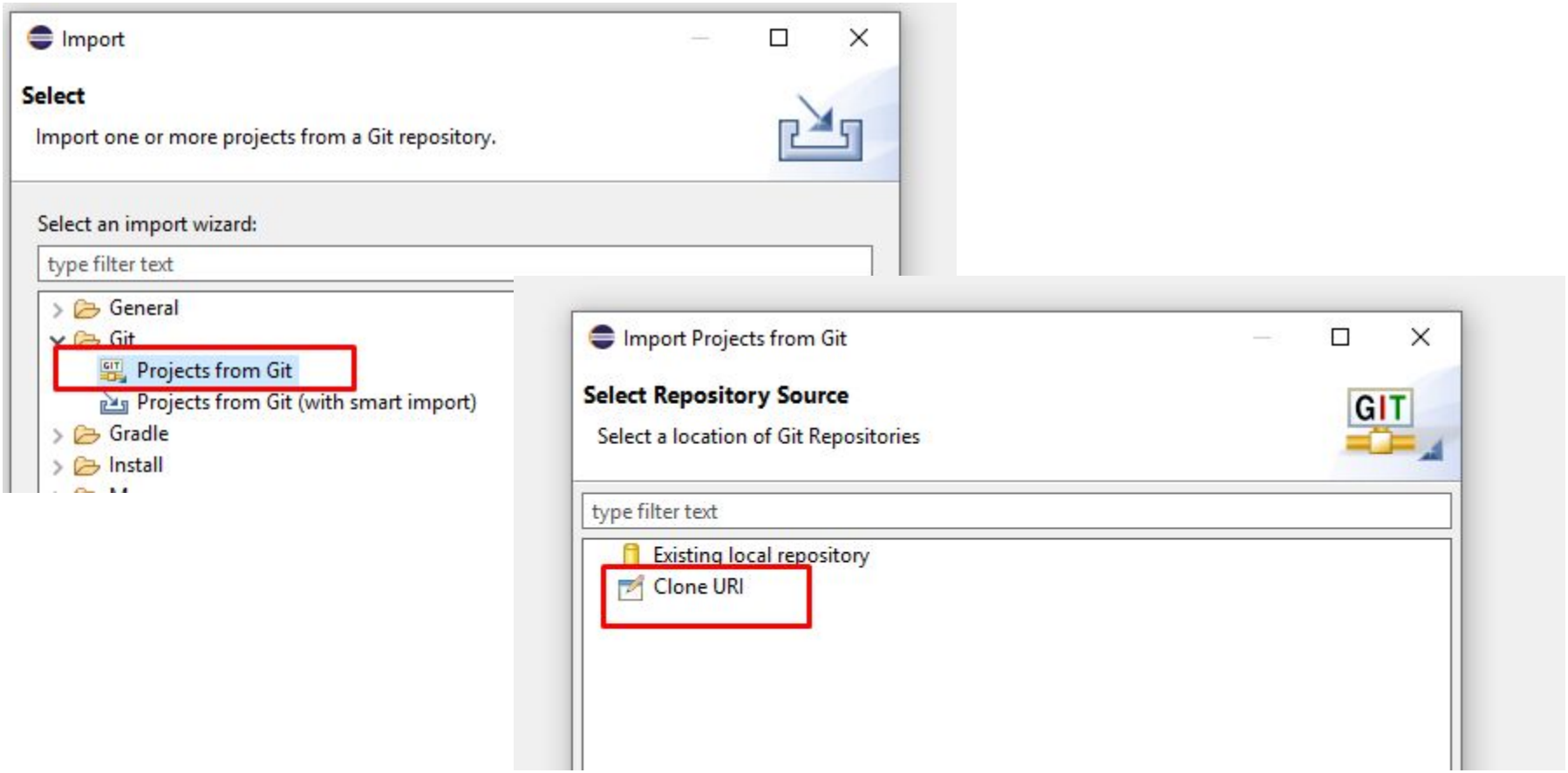
Скопировать репозиторий в eclipse

1) Открываем eclipse
2) Кликаем правую кнопку мыши в Package Explorer (слева)
3) Выбираем import



Скопировать репозиторий в eclipse

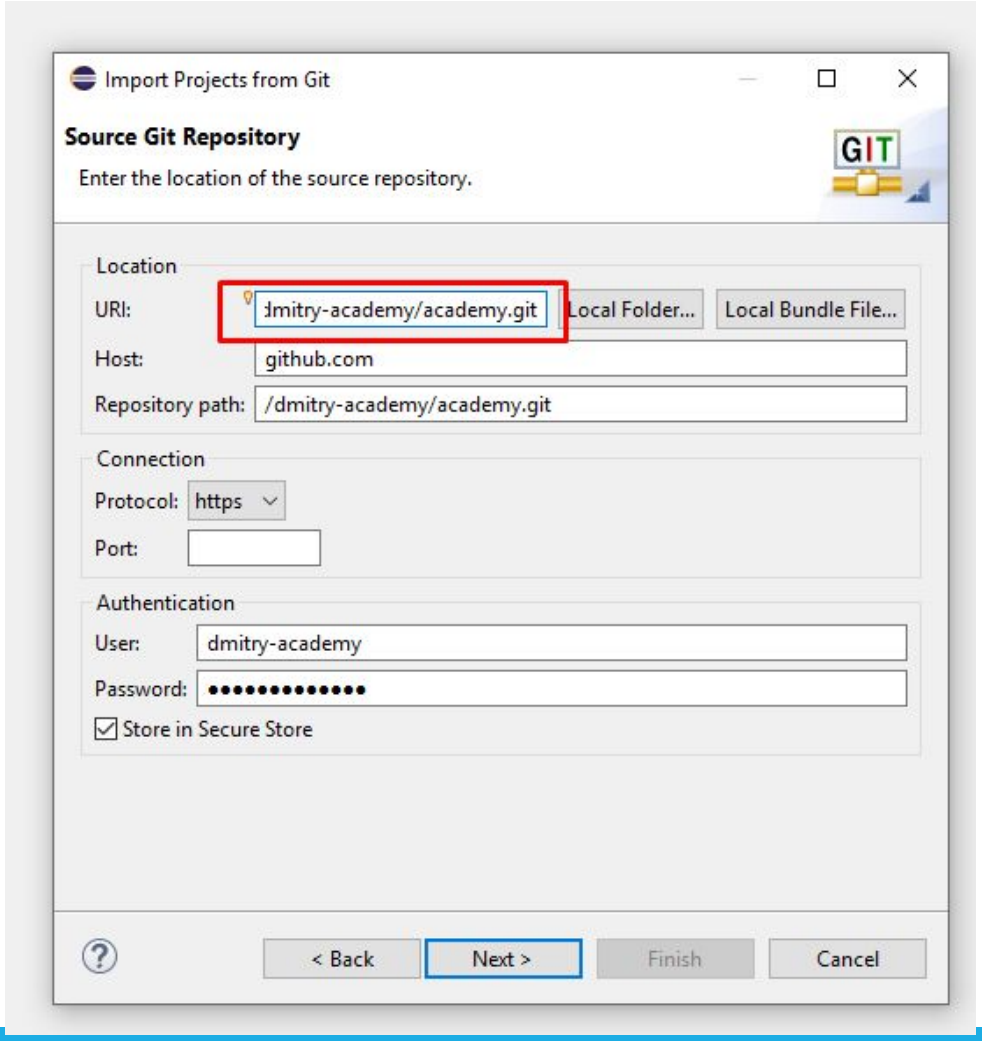
1) Выбираем Projects from Git
2) Clone URI



Скопировать репозиторий в eclipse

IT-Academy

1) Вставляем ссылку с Git
2) Жмем NEXT
3) Жмем NEXT
4) Жмем NEXT
5) Жмем FINISH



Вопрос

IT-Academy

IT-Academy

Спасибо за внимание