Введение

Привет! С этого модуля вы переходите к более серьёзной и профессиональной работе. Вам предстоит с нуля разработать проект со всеми его нюансами. Здесь не обойтись без правильной организации кода.

С этого задания вы будете использовать систему контроля версий Git.

Git — это система управления версиями, с помощью которой можно удобно организовывать работу над проектом. Git хорошо подходит для команд, когда каждый участник работает над кодом на своём компьютере, а затем передаёт его в общий главный проект, который идёт в продакшн.

Вы будете работать в GitLab, то есть в графическом представлении Git, в котором показано, где хранятся файлы, как с ними взаимодействовать, какие проверки они проходят и так далее.

В этом и следующих проектах вы примерите на себя роль разработчика, а ваш куратор станет тимлидом, который проследит за тем, чтобы ваша работа была правильной с точки зрения функционала и код-стайла, выбранного на курсе.

Ваша задачи в проекте:

- соблюдать общепринятый подход к разработке, когда вы делаете задачи в отдельных «ветках»;
- отдавать куратору на ревью ваши изменения.

Если вы пройдёте ревью, то после него будете «вливать» свои изменения в общую ветку разработки.

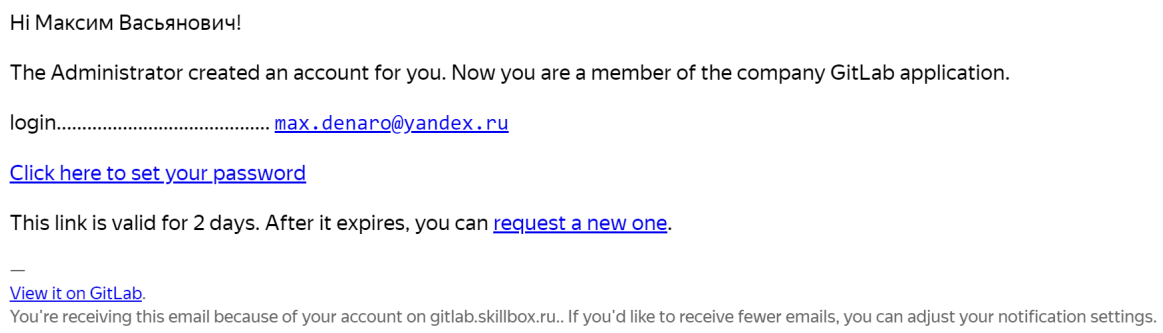
По этому принципу работают практически все компании: так можно поддерживать качество кодовой базы и управлять изменениями в проекте. Это правило хорошего тона в проекте с командой из двух-трёх человек и просто жизненная необходимость, когда в ней появляется больше пяти участников.

У нас есть бесплатный курс «Система контроля версий Git», доступ к которому уже открыт для вас. Если доступа нет, запросите его в поддержке — hello@skillbox.ru. Рекомендуем изучить курс, чтобы познакомиться со всеми нюансами Git, но всё самое важное мы расскажем здесь.

GitLab и структура проекта

Прежде чем разбираться непосредственно с загрузкой кода, посмотрим, как устроен Gitlab.

Переходим на сайт gitlab.skillbox.ru, на котором вас сразу встретит окно входа (или регистрации). Мы уже зарегистрировали вас в GitLab. В самом начале прохождения курса вам на почту пришло письмо с логином для входа. Используйте его, чтобы зайти на сайт. Если письма нет — напишите в поддержку hello@skillbox.ru, и вам предоставят данные снова.



Hi Максим Васьянович!

The Administrator created an account for you. Now you are a member of the company GitLab application.

login..... max.denaro@yandex.ru

Click here to set your password

This link is valid for 2 days. After it expires, you can request a new one.

—

View it on GitLab.

You're receiving this email because of your account on gitlab.skillbox.ru.. If you'd like to receive fewer emails, you can adjust your notification settings.

Нажмите на ссылку click here to set your password, чтобы установить пароль учётной записи. После ввода данных вы зайдёте в ваш профиль GitLab.

Ваш профиль может быть пустым (без репозиториев): они создаются, когда вы начинаете практическую работу. Например, когда вы начнёте модуль 6, создастся репозиторий, связанный с макетом lagoona, где можно работать с кодом.

Структура проекта максимально простая: в папке лежит файл .editorconfig, который устанавливает одни и те же настройки редактора у всех участников проекта. Также в ней есть папка src с файлами вашего проекта.

Теперь воспользуемся Git для загрузки кода.

Как загрузить проект

Что понадобится для работы с Git

1. Git, установленный в систему.
    a. Как установить Git на Windows.
    b. Как установить Git на Mac OS.
2. VS Code, а точнее его терминал (командная строка).

После установки Git в систему можно использовать его команды, которые легко запоминаются.

Шаг 1: Клонируем файлы

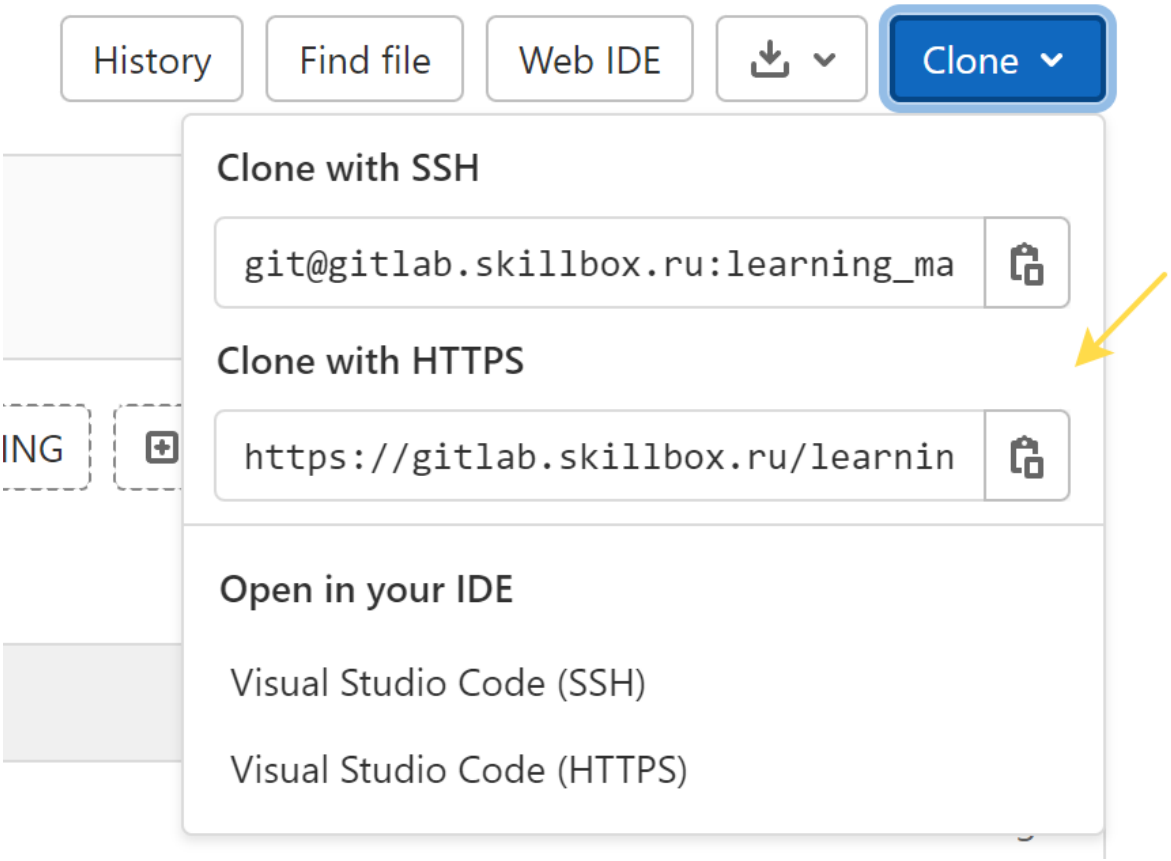
Чтобы начать работу с проектом, нужно получить на свой компьютер все файлы, которые уже лежат в GitLab. Данные хранятся в репозитории.

Процесс, когда вы впервые загружаете файлы проекта к себе локально, называется «клонированием». Вы клонируете репозиторий на свой компьютер, то есть создаёте его копию.

Зайдём в любой диск в системе, например в диск D, и создадим в нём папку Skillbox Projects. После этого нужно открыть её внутри VS Code. Вы можете просто перетащить папку в открытое окно VS Code. Это необходимо, чтобы работать с Git именно в этой папке.

Теперь открываем терминал сочетанием клавиш Ctrl + Shift +’ на Windows или Control + Shift + ‘ на Mac OS.

После открытия нужно использовать говорящую команду — git clone, которой необходимо передать адрес, с которого делать клонирование. Его можно получить внутри репозитория в GitLab. Нажмите на кнопку Clone и скопируйте путь по названию Clone with HTTPS.



Мы скопировали путь, теперь нужно ввести команду git clone https://gitlab.skillbox.ru//weblayout_1.git

Так мы говорим: «Склонируй репозиторий, который находится по адресу...».

При успешном клонировании вы увидите:

```
Cloning into 'weblayout_06_layout-html'...
warning: redirecting to https://gitlab.skillbox.ru/learning_r
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Unpacking objects: 100% (3/3), 354 bytes | 2.00 KiB/s, done.
PS E:\>
```

Шаг 2: Переходим в проект

Чтобы дальше работать с репозиторием, нужно перейти в него с помощью команды change directory (cd).

Поскольку мы производили клонирование, папка репозитория weblayout_Lagoona появилась в папке, из которой и запускалось клонирование. Чтобы перейти в клонированную папку, достаточно написать cd weblayout_Lagoona.

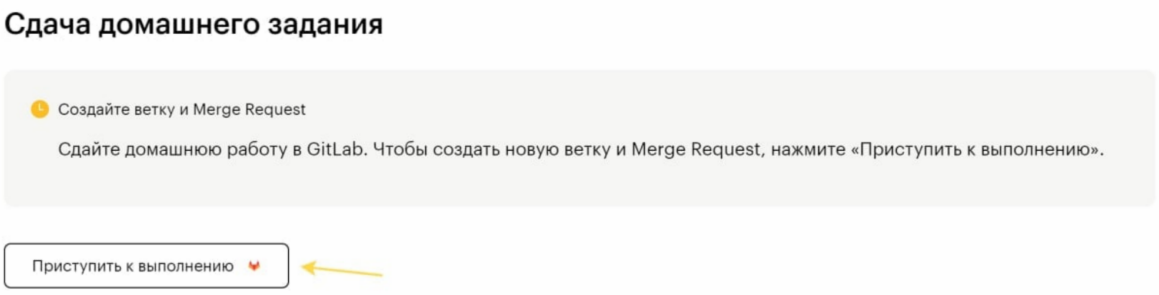
Шаг 3: Создаём ветку в проекте
Теперь мы находимся в проекте. Это место, в котором вы будете выполнять несколько практических работ, объединённых общим смыслом. Вы создадите один сайт, разработка которого делится на несколько задач. Каждая задача — это отдельная практическая работа, для которой у вас будет отдельная ветка.

Ветка — специальное место, в которое отклоняется проект при разработке. С веткой Master будет работать только ваш куратор (он решает, принимать или нет проект), а вы будете работать с другими ветками.

Например, проект состоит из пяти практических работ. Значит, вы будете работать в пяти ветках.

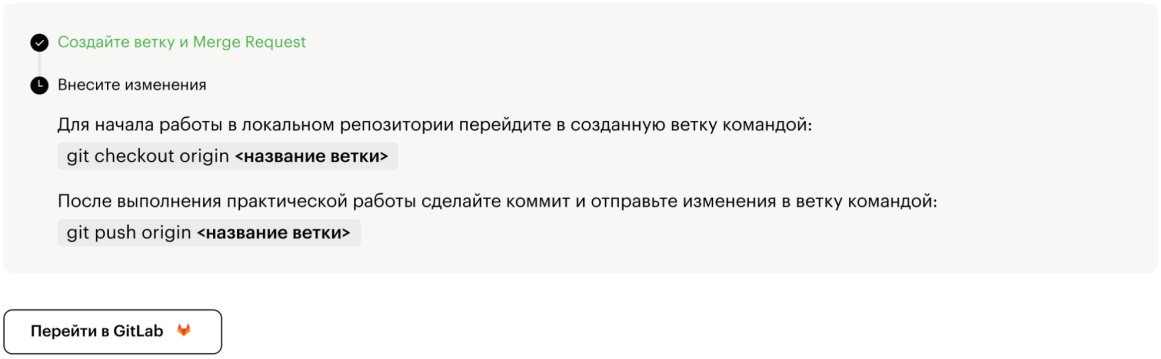
Подробнее ознакомиться с идеей ветвления можно в модуле 3 курса «Система контроля версий Git».

Чтобы создать ветку, нужно нажать на кнопку «Приступить к выполнению» внутри практической работы. Ветка создастся автоматически, и вы сможете в ней работать.



После создания ветки нужно перейти в неё, чтобы загрузить практическую работу. Название ветки появится у вас автоматически.

Сдача практической работы



Если вы напишете ветку неверно, то не сможете сдать работу.

Напишите команду для ветки: git checkout origin.

Шаг 4: Работаем в проекте
Когда вы перейдёте в нужную ветку — можно начать практическую работу. Для примера создадим небольшую разметку проекта (у вас будет своя) и загрузим её в Git пошагово.

Зайдём в папку weblayout_Lagoona. В ней уже лежит файл .editorconfig, который управляет настройками редактора. Ваша задача — сделать разметку, поэтому создайте файл index.html прямо в этой папке и напишите разметку проекта.

Когда вы написали код, уверены, что сделали всё, проверили работу и готовы её отправить, откройте терминал в папке weblayout_Lagoona и напишите несколько команд.

Шаг 5: Отправляем результаты работы через Git
Любые изменения внутри репозитория Git фиксируются и отмечаются специальным образом. Они представляют из себя коммит, то есть небольшую ячейку временной шкалы проекта в Git. Вы можете переходить к коммиту, откатываться назад, если допустили ошибку, и так далее. Это позволит разделить проект на маленькие части.

Чтобы создать коммит, нужно сперва добавить файлы к нему.

Чтобы лучше понимать, что происходит, полезно знать статус файлов в Git. Для этого существует команда git status. Если вы введёте её, то увидите статусы файлов: какой изменён, какой готов к коммиту, какой удалён и так далее.

Введём git status и посмотрим на ситуацию.

```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Появился добавленный файл index.html, Git отреагировал на него и сразу показал.

Когда вы уверены, что что-то поменялось, можете добавить файл к коммиту, используя команду git add (как раз её подсказывает консоль). В эту команду можно передавать нужный файл через параметр, но это неудобно, если их несколько. Для передачи нескольких файлов используется параметр точка.

Вводим команду git add . и после неё сразу — git status. Вы увидите, что файлы из красных стали зелёными — они подготовлены к коммиту.

Следующий шаг — сделать коммит на основе текущих изменений. Коммит должен содержать небольшое описание, которое чётко говорит, что именно вы сделали. Это описание указывается в кавычках. Напишем коммит.

git commit -m «добавил разметку сайта»

В кавычках указано то, что сделано, а параметр -m позволяет дать описание коммиту. В описании принято указывать, что конкретно добавляет этот коммит. Так удобнее смотреть историю изменений.

Например:

- «добавлен футер»;
- «исправлена обработка номера телефона в форме регистрации».

После коммита остаётся пуш (отправка) файлов, который доставляет локальные файлы в удалённый репозиторий.

git push -u origin work_html

Важные команды

1. Origin — это сокращённое название удалённого репозитория в GitLab. По умолчанию используется название origin, которое переводится как оригинал, источник. Если вы введёте команду git remote list, то увидите, что origin — это синоним репозитория в GitLab.
2. -U — технический флаг, который объясняет, что ветка на вашем компьютере связана с веткой в репозитории на сервере в GitLab. Так в будущем вы сможете писать git push, а Git сам поймёт, куда пушить из этой ветки.

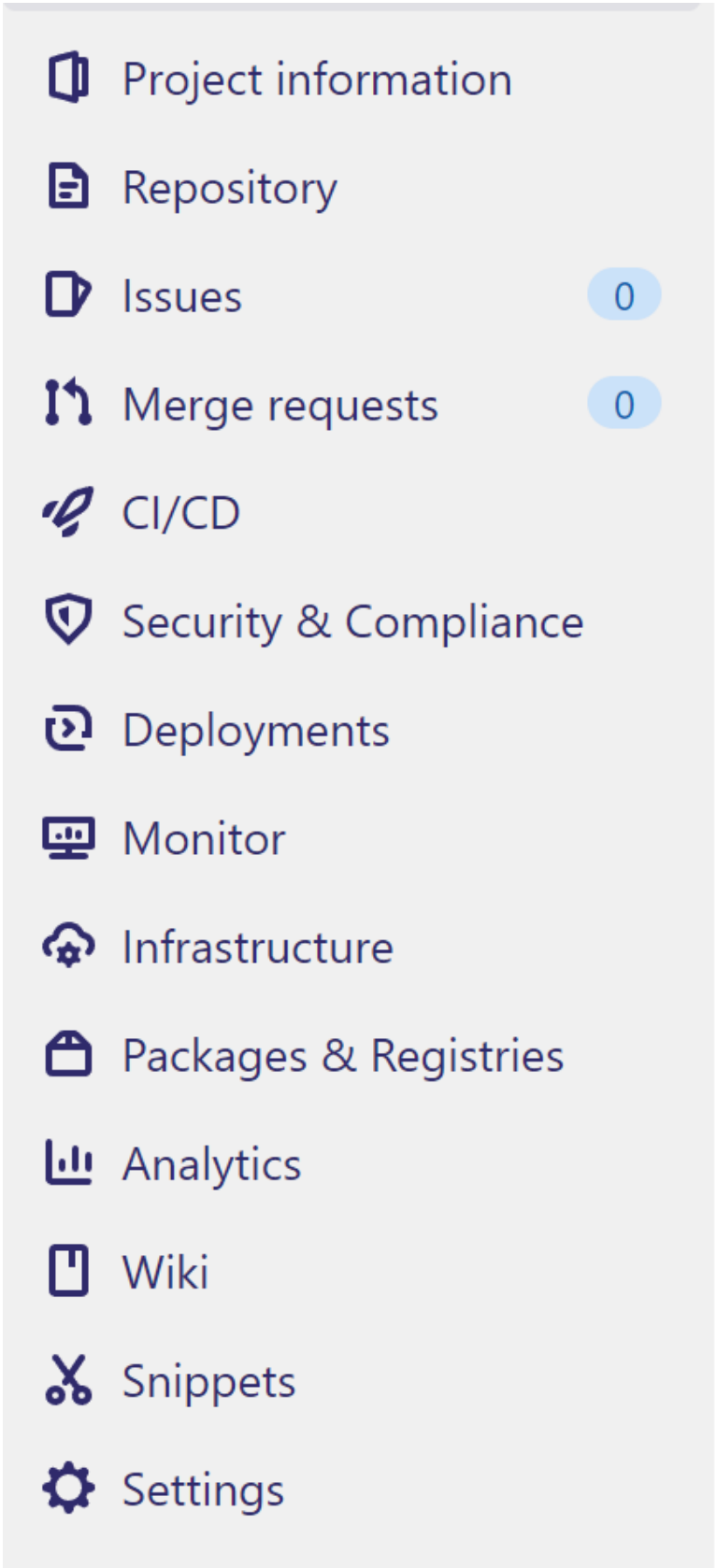
Шаг 6: Отправляем запрос на слияние

После пуша проекта нужно сделать Merge Request (запрос на слияние). Это позволит оформить запрос на влитие кода в основную ветку разработки.

В рамках MR ваш тимлид будет проводить Core Review, то есть комментировать ваш код, просить внести изменения, доработать.

Когда вы доработаете код по всем комментариям — работа будет принята. После этого её можно отправить в основную ветку разработки нажатием одной кнопки. В итоге код попадёт в ветку master как абсолютно правильный и рабочий.

Сделать merge request можно на сайте gitlab.skillbox.ru в вашем репозитории.



Нажмите на кнопку New merge request, выберите ветку, изменения из которой вы хотите интегрировать в мастер, укажите описание запроса и отправьте его.

Внимание. Несмотря на то, что вы можете принять запрос самостоятельно, делать этого не нужно. Это может сделать только ваш куратор, если работа выполнена корректно.

Как загрузить правки по работе

Представьте, что ваша работа выполнена неверно, некоторые детали нужно доработать. Преподаватель вернул работу на платформе и отклонил ваш запрос на слияние. Что делать дальше?

Внесите изменения по комментариям, сделайте правильно всю вёрстку и вернитесь к пятому шагу.

Как работать с новой практической работой

Представьте, что вы выполнили шестую практическую работу и приступили к седьмой. Перед началом работы зайдите на сайт GitLab через шестую практическую работу, а затем переходите к третьему шагу. В нём нужно указать новую ветку из практической работы.

Заключение

Сначала работа с Git может быть непростой, но со временем вы привыкнете и будете чувствовать себя в нём как рыба в воде. Это важно для будущего трудоустройства.