Небольшой мануал по работе с удаленным репозиторием

До этого момента мы работали только с локальным репозиторием, который располагается только на вашей машине.

При командной разработке необходимо как-то синхронизировать изменения в проекте всей команде, т.е. если один разработчик внес правки, должна быть возможность сделать так, чтобы эта правка появилась у всех остальных членов команды.

Для этих целей существует множество сервисов. Мы будем в дальнейшем работать в GitVerse.

С регистрацией разберетесь сами, там нет ничего сложного. После регистрации важно создать ssh-ключ, который будет использоваться для идентификации вас, когда вы будете делать запросы к gitVerse.

Создание ssh ключа

В терминале вводим

```
$ ssh-keygen
```

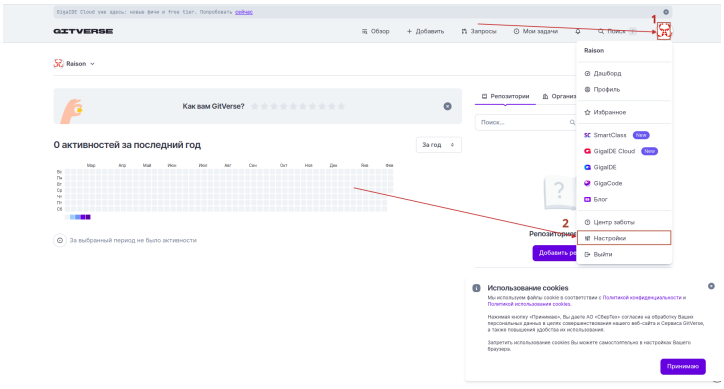
После будет несколько запросов на уточнение директории с ключами и задание пароля, можно просто везде нажимать Enter, в этом случае будут использованы директория по умолчанию и пустой пароль. В конце вы получите примерно вот такой вывод:

```
raison@raison-Nitro:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/raison/.ss
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./temp/key
Your public key has been saved in ./temp/key.pub
The key fingerprint is:
SHA256:WsvRO72e/3htxt3aD1QX71692JcDxkbKyFqTvw9cFAE ra
The key's randomart image is:
+--[ED25519 256]--+
|           E.o.. |
|              . o|
|             o  +|
|         ..+ =  oo|
|        S*.o *. +|
|       +oo+o+.+.+|
|      ..o o+...**|
|           .oo +@|
|           o=o+*=|
+----[SHA256]-----+
```

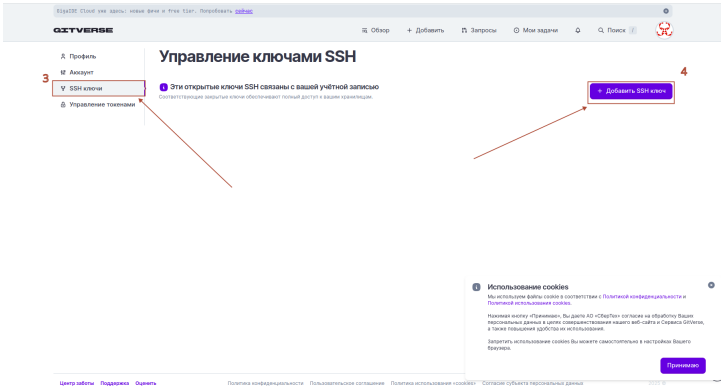
В директории ~/ssh(если вы использовали директорию по умолчанию) появяться 2 файла id_ed*.pub и id_ed*. Публичный и приватный ключи. Собственно приватный ключ всегда лежит у вас и вы его не трогаете, а публичный надо будет закинуть на сайт GitVerse.

Подключение ssh ключа

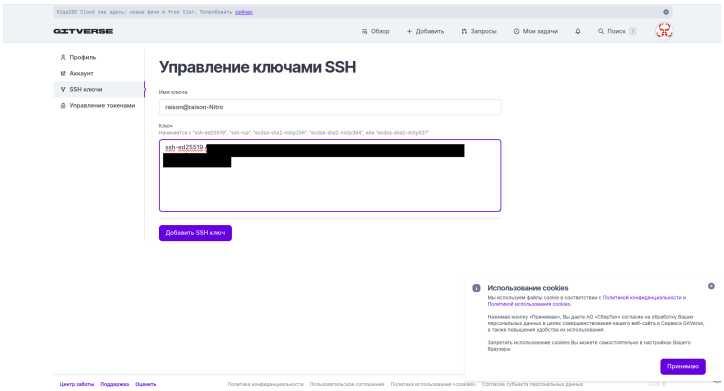
После регистрации заходим в настройки



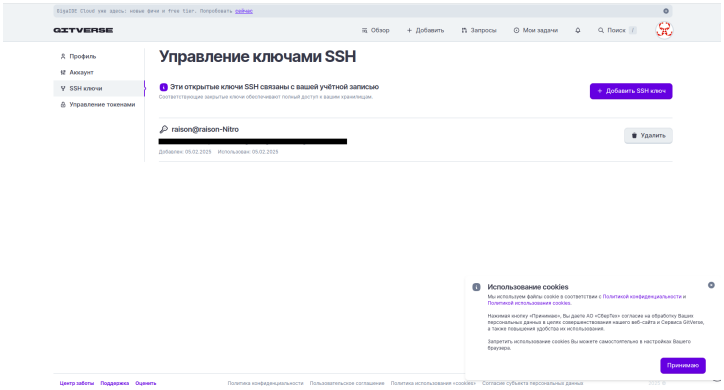
Дальше в раздел ssh ключей



Вводим название ключа и данные *публичного* ключа в соответствующие поля



Подтверждаем и готово. Если все прошло хорошо – у вас в списке должен появится добавленный ключ



Теперь все ваши запросы на удаленный репозиторий будут *подписаны* вашей цифровой подписью. Ну и соединение будет шифроваться.

Важно! Данный ключ привязан именно к операционной системе. Поэтому если у вас есть несколько машин/операционных систем, то для доступа с них к вашему аккаунту надо с каждой из них сделать свой ssh-ключ и подключить их к вашему аккаунту.

Теперь про наш с вами pipeline работы.

Задания и все с ними связанное

В общем-то gitVerse сделан как аналог github и gitlab, так что в нем можно делать все аналогичные действия по работе с удаленными репозиториями. Также в gitVerse есть удобный функционал для преподавателей и студиоузов. Им мы и будем пользоваться.

Я создам класс и вы добавитесь туда по ссылке, после этого вы сможете видеть задания выполнять их, я смогу проверять их, делать замечания и т.д. и т.п. Красота в общем.

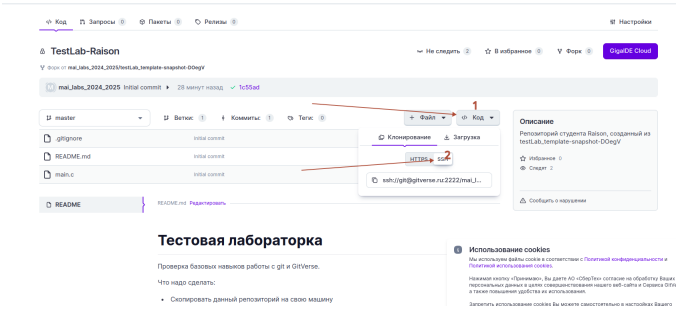
Я скину ссылку с заданием. Там в файле readme.md в общем описано, что надо сделать. Руководствуйтесь предыдущим мануалом по git'у. Единственное, добавлю про скачивание и запуливание коммитов с репозитория(в репозиторий).

Работа с удалённым репозиторием

Если вы делали создание локального репозитория через git init, то для подключения удаленного репозитория можно использовать команду

```
$ git remote add origin <link>
```

Ссылку можно вытащить отсюда:



Также можно напрямую скопировать удаленный репозиторий на локальную машину командой:

```
$ git clone <link>
```

Ссылка используется та же.

После копирования у вас появляется папочка с проектом, где уже можно делать задания лабы. Если вы делали через git remote add, то надо дополнительно скачать файлы из ветки master в ваш локальный репозиторий. Это делается командой:

```
$ git pull origin master
```

Дальше вы делаете задание в отдельной ветке, создаете коммит, и после этого его надо запушить обратно в удаленный репозиторий в вашу ветку:

```
$ git push origin <branch>
```