

Лабораторная работа №2

НКАбд-03-22

Шубнякова Дарья Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	13
	Список литературы	18

Список иллюстраций

4.1	Базовая настройка	9
4.2	Скриншот рабочего ключа SSH	10
4.3	GPG-ключ	10
4.4	Команды	11
4.5	Аккаунт на Github	11
4.6	Репозиторий, созданный на основании локального рабочего каталога	12

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

2 Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

Настроить подписи git.

Зарегистрироваться на Github.

Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

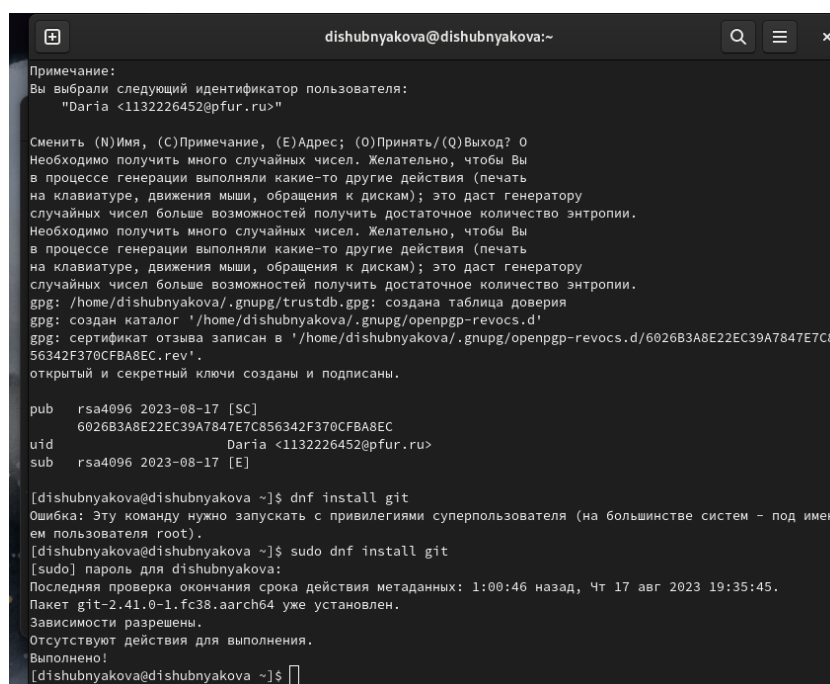
Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

Примеры использования git

Система контроля версий Git представляет собой набор программ командной строки. Благодаря тому, что Git является распределённой системой контроля версий, резервн

4 Выполнение лабораторной работы

Как можно увидеть на скриншоте, мною уже проведена базовая настройка git в предыдущем семестре(рис. 4.1).



```
dishubnyakova@dishubnyakova:~  
Примечание:  
Вы выбрали следующий идентификатор пользователя:  
"Daria <1132226452@pfur.ru>"  
  
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O  
Необходимо получить много случайных чисел. Желательно, чтобы Вы  
в процессе генерации выполняли какие-то другие действия (печать  
на клавиатуре, движения мыши, обращения к дискам); это даст генератору  
случайных чисел больше возможностей получить достаточное количество энтропии.  
Необходимо получить много случайных чисел. Желательно, чтобы Вы  
в процессе генерации выполняли какие-то другие действия (печать  
на клавиатуре, движения мыши, обращения к дискам); это даст генератору  
случайных чисел больше возможностей получить достаточное количество энтропии.  
gpg: /home/dishubnyakova/.gnupg/trustdb.gpg: создана таблица доверия  
gpg: создан каталог '/home/dishubnyakova/.gnupg/openpgp-revocs.d'  
gpg: сертификат отзыва записан в '/home/dishubnyakova/.gnupg/openpgp-revocs.d/6026B3A8E22EC39A7847E7C8  
56342F370CFBA8EC.rev'.  
открытый и секретный ключи созданы и подписаны.  
  
pub   rsa4096 2023-08-17 [SC]  
      6026B3A8E22EC39A7847E7C856342F370CFBA8EC  
uid           Daria <1132226452@pfur.ru>  
sub   rsa4096 2023-08-17 [E]  
  
[dishubnyakova@dishubnyakova ~]$ dnf install git  
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на большинстве систем - под имен  
ем пользователя root).  
[dishubnyakova@dishubnyakova ~]$ sudo dnf install git  
[sudo] пароль для dishubnyakova:  
Последняя проверка окончания срока действия метаданных: 1:00:46 назад, Чт 17 авг 2023 19:35:45.  
Пакет git-2.41.0-1.fc38.aarch64 уже установлен.  
Зависимости разрешены.  
Отсутствуют действия для выполнения.  
Выполнено!  
[dishubnyakova@dishubnyakova ~]$
```

Рис. 4.1: Базовая настройка

Ключ так же уже был мною создан(рис. 4.2).

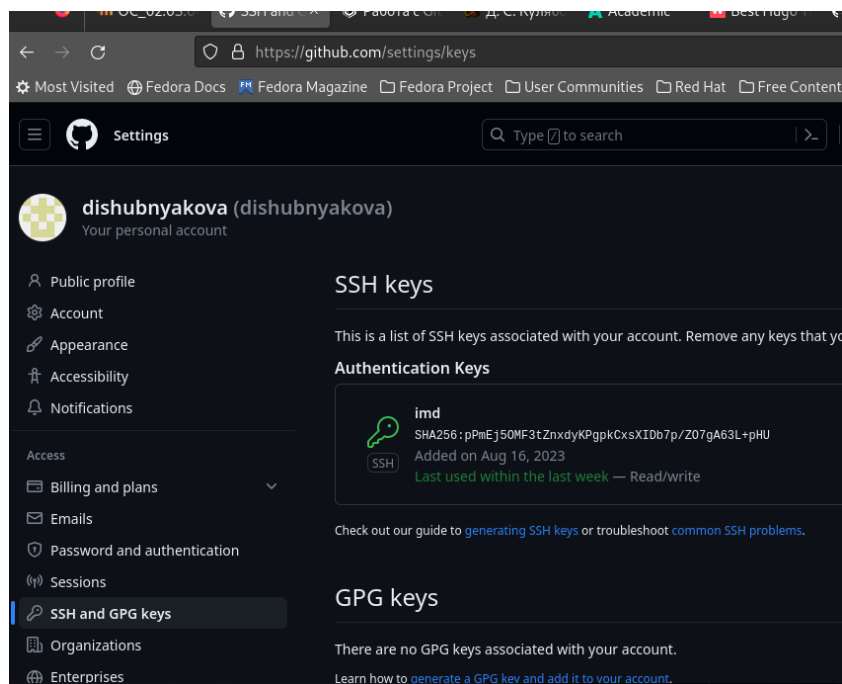


Рис. 4.2: Скриншот рабочего ключа SSH

Создаем ключ GPG с помощью команды `gpg --full-generate-key`(рис. 4.3).

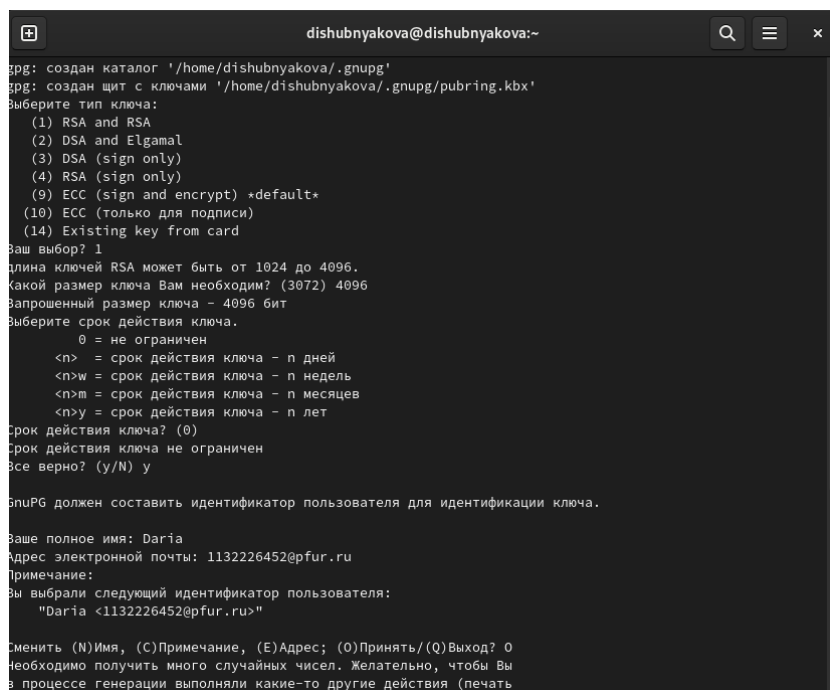


Рис. 4.3: GPG-ключ

Настройка подписей git (проведена)(рис. 4.4).

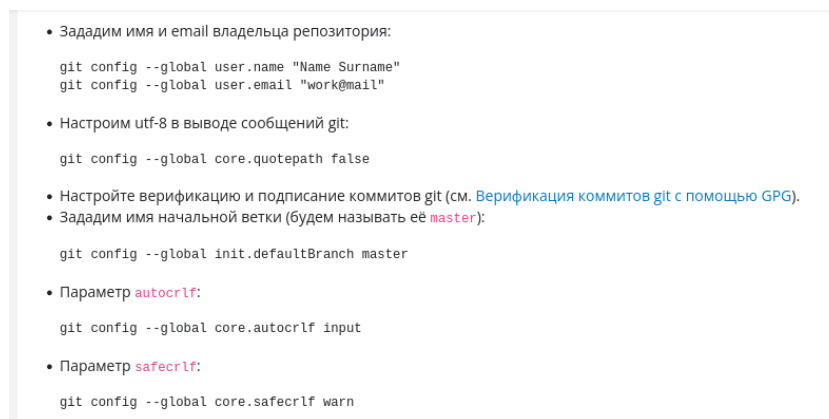


Рис. 4.4: Команды

Уже созданный мною аккаунт(рис. 4.5).

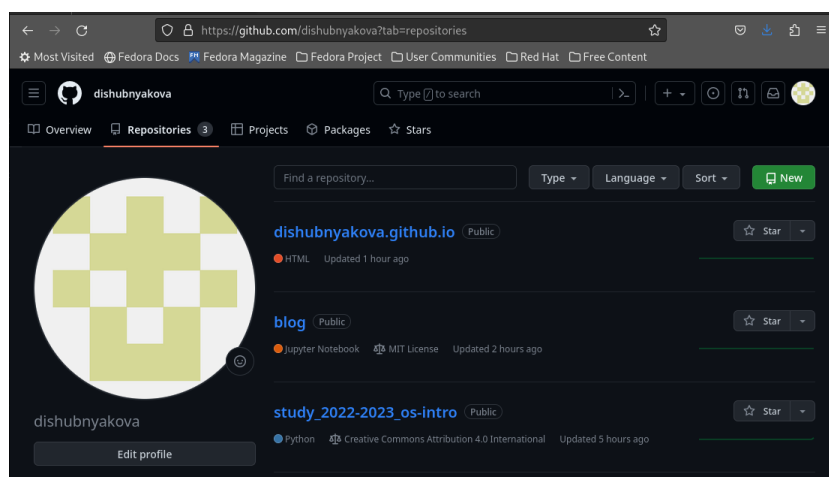


Рис. 4.5: Аккаунт на Github

Созданный при выполнении первой лабораторной работы каталог(рис. 4.6).

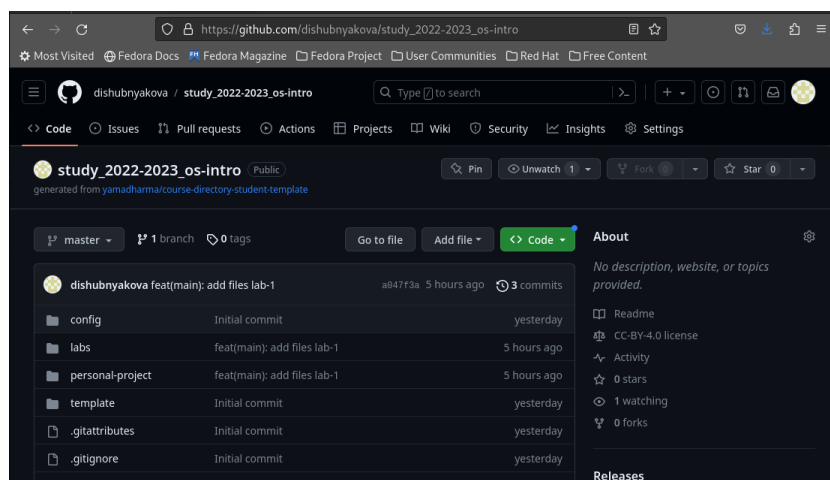


Рис. 4.6: Репозиторий, созданный на основании локального рабочего каталога

5 Выводы

Мы проделали большую часть работы на курсе “Архитектура компьютеров”, благодаря чему выполнение данной работы не заняло у нас слишком много времени.

- 1) VCS – система контроля версий. Хранит записи обо всех записях. Является резервным хранилищем. Можно работать совместно.
- 2) Репозиторий – это хранилище (сама VCS). Коммит – сохранение изменений в репозитории. Из коммитов состоит история. Рабочая копия – копия проекта, связанная с репозиторием.
- 3) Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev):
 - Одно основное хранилище всего проекта
 - Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратноДецентрализованные VCS (Git; Mercurial; Bazaar):
 - У каждого пользователя свой вариант (возможно не один) репозитория
 - Присутствует возможность добавлять и забирать изменения из любого репозиторияВ классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

- 4) Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
- 5) Участник проекта (пользователь) перед началом работы посредством определенных команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.
- 6) Первая—хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
- 7) Перечислим наиболее часто используемые команды git.

Создание основного дерева репозитория:

`git init`

Получение обновлений (изменений) текущего дерева из центрального репозитория:

`git pull`

Отправка всех произведённых изменений локального дерева в центральный репозиторий:

`git push`

Просмотр списка изменённых файлов в текущей директории:

`git status`

Просмотр текущих изменений:

`git diff`

Сохранение текущих изменений:

добавить все изменённые и/или созданные файлы и/или каталоги:

```
git add .
```

добавить конкретные изменённые и/или созданные файлы и/или каталоги:

```
git add имена_файлов
```

удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог удаляется из репозитория):

```
git rm имена_файлов
```

Сохранение добавленных изменений:

сохранить все добавленные изменения и все изменённые файлы:

```
git commit -am 'Описание коммита'
```

сохранить добавленные изменения с внесением комментария через встроенный редактор:

```
git commit
```

создание новой ветки, базирующейся на текущей:

```
git checkout -b имя_ветки
```

переключение на некоторую ветку:

```
git checkout имя_ветки
```

(при переключении на ветку, которой ещё нет в локальном репозитории, он будет создан)

отправка изменений конкретной ветки в центральный репозиторий:

```
git push origin имя_ветки
```

слияние ветки с текущим деревом:

```
git merge --no-ff имя_ветки
```

Удаление ветки:

удаление локальной уже слитой с основным деревом ветки:

```
git branch -d имя_ветки
```

принудительное удаление локальной ветки:

```
git branch -D имя_ветки
```

удаление ветки с центрального репозитория:

```
git push origin :имя_ветки
```

- 8) `git push -all` (push origin master/любой branch)
- 9) Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Обычно есть главная ветка (master), или ствол (trunk). Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.
- 10) Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например,

временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

Список литературы

1. Dash P. Getting started with oracle vm virtualbox. Packt Publishing Ltd, 2013. 86 p.
2. Colvin H. Virtualbox: An ultimate guide book on virtualization with virtualbox. CreateSpace Independent Publishing Platform, 2015. 70 p.
3. van Vugt S. Red hat rhcsa/rhce 7 cert guide : Red hat enterprise linux 7 (ex200 and ex300). Pearson IT Certification, 2016. 1008 p.
4. Робачевский А., Немнюгин С., Стесик О. Операционная система unix. 2-е изд. Санкт-Петербург: БХВ-Петербург, 2010. 656 p.
5. Немет Э. et al. Unix и Linux: руководство системного администратора. 4-е изд. Вильямс, 2014. 1312 p.
6. Колисниченко Д.Н. Самоучитель системного администратора Linux. СПб.: БХВ-Петербург, 2011. 544 p.
7. Robbins A. Bash pocket reference. O'Reilly Media, 2016. 156 p.