

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Тураева Оиша

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Контрольные вопросы	14
5	Выводы	16

Список иллюстраций

3.1	Базовая настройка git	9
3.2	Создание ключа SSH	10
3.3	Ключ SSH создан	10
3.4	Ключ GPG создан	11
3.5	Отпечаток приватного ключа	11
3.6	Настройка подписей	12

Список таблиц

1 Цель работы

1. Изучить идеологию и применение средств контроля версий.
2. Освоить умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. 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. # Выполнение лабораторной работы

Базовая настройка git:

1. Задаём имя и email владельца репозитория (1 и 2 строка на рисунке)
2. Настраиваем utf-8 в выводе сообщений git (3 строка на рисунке)
3. Настраиваем верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master) (4 строка на рисунке)


```
omturaeva@dk1n22 ~$ git config --global user.name "b325353"
omturaeva@dk1n22 ~$ 
omturaeva@dk1n22 ~$ git config --global user.email "aishaturaeva13@gmail.com"

git: «config» не является командой git. Смотрите «git --help».

Самые похожие команды:
  config
omturaeva@dk1n22 ~$ 
omturaeva@dk1n22 ~$ git config --global user.email "aishaturaeva13@gmail.com"
omturaeva@dk1n22 ~$ git config --global core.quotepath false
omturaeva@dk1n22 ~$ git config --global init.defaultBranch master
omturaeva@dk1n22 ~$ git config --global core.autocrlf input
omturaeva@dk1n22 ~$ git config --global core.safecrlf warn
omturaeva@dk1n22 ~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/o/m/omturaeva/.ssh/id_rsa):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/o/m/omturaeva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/o/m/omturaeva/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/o/m/omturaeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:mOp8PySeS4/50nbp4SBC1UAB6/QAvdVJW80slwbimw omturaeva@dk1n22
The key's randomart image is:
+---[RSA 4096]-----+
|  ..o+...o+=      |
|  ....+o=  =o+    |
|  +o. E +  .      |
|  o.+ +          |
|  o + S          |
|  . . .          |
|  o.o=.  ..      |
|  o o==+oo.      |
|  o.++++o        |
+---[SHA256]-----+
omturaeva@dk1n22 ~$ gpg --full-generate-key
gpg (GnuPG) 2.2.42; Copyright (C) 2023 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
(1) RSA и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
```

Рис. 3.1: Базовая настройка git

Создаём ключ SSH. В терминале вводим данную команду:

`ssh-keygen -t rsa -b 4096`

Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ.

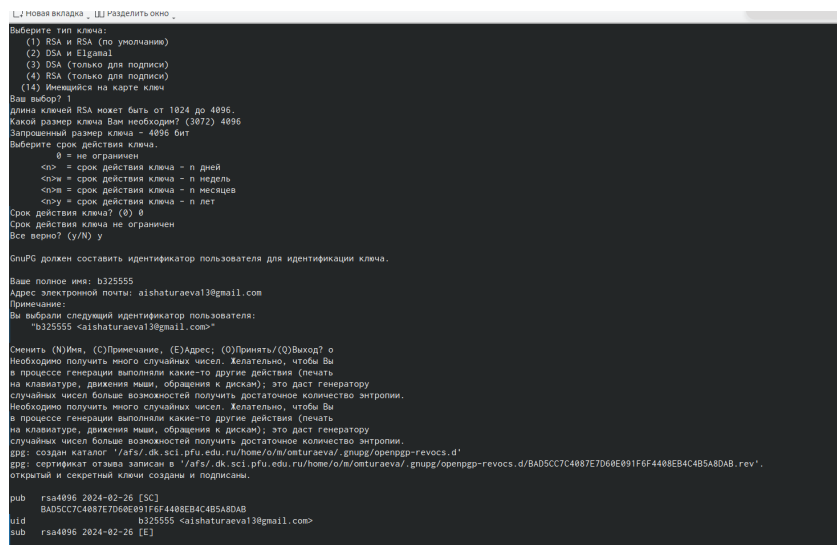


Рис. 3.2: Создание ключа SSH

Ключ нужно добавить на github. Для этого переходим на сайте в раздел “Settings” и выбираем “SSH and GPG keys”.

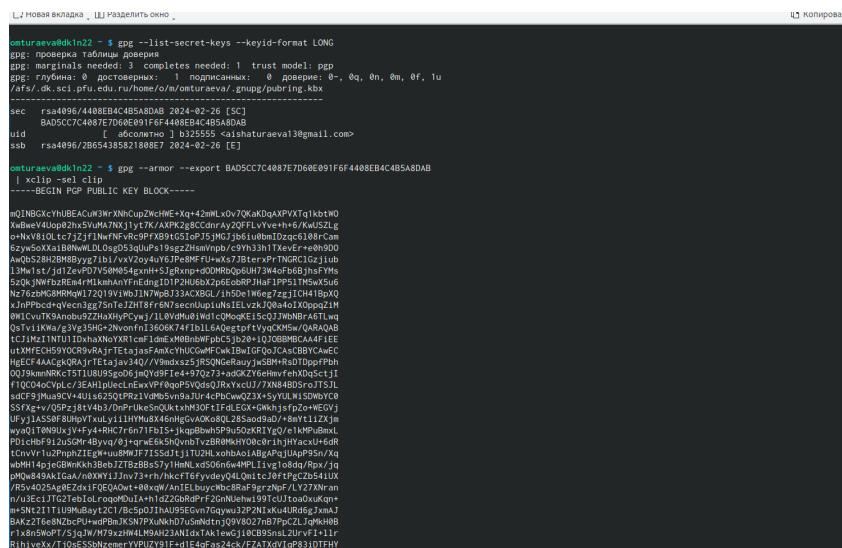


Рис. 3.3: Ключ SSH создан


```

remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 97 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 KiB | 97.00 KiB/c, готово.
Определение изменений: 100% (14/14), готово.
Копирование в «/afs/06.scl.yu.edu.ru/home/o/s/osturaeva/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro/template/report...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 315.08 KiB | 1.29 MiB/c, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/report': checked out '40c761813e101d0e843ff1c72c68a204f24c'
Submodule path 'template/report': checked out '7c31ab8e5d8ac0b2d07cae8a19ef8028ced88e'
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы $ cd ~/work/study/2022-2023/Операционные системы/os-intro
bash: cd: /afs/06.scl.yu.edu.ru/home/o/s/osturaeva/work/study/2022-2023/Операционные системы/os-intro: Нет такого файла или каталога
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы $ ls
study_2022-2023_os-intro
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы $ cd study_2022-2023_os-intro
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ rm package.json
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ echo os-intro > COURSE
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ make
Usage:
  make *target*

Targets:
  list          List of courses
  prepare      Generate directories structure
  submodule    Update submodules
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ git add
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ git commit -m 'feat(main): make course structure'
[master e03d983] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $ git push
Пересылание объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сканировании используются до 6 потоков
Сканирование объектов: 100% (2/2), готово.
Запись объектов: 100% (1/1), 953 байта | 953.00 KiB/c, готово.
Всего 3 изменения: 1, повторно использовано 0 изменений 0; повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:0025955/study_2022-2023_os-intro.git
ec0a0ef..e03d980 master -> master
osturaeva@dkln22: ~/work/study/2022-2023/Операционные системы/study_2022-2023_os-intro $

```

Рис. 3.6: Настройка подписей

Возвращаемся в наш терминал и настраиваем gh командой:

gh auth login.

Во всех пунктах выбираем у(yes).

По полученной ссылке переходим в браузер на виртуальной машине и вводим код из терминала (находится перед ссылкой). Создаём репозиторий курса на основе шаблона. Все нужные команды для создания были в указаниях к лабораторной работе. В 4 команде, вместо , указываем своё имя профиля на github.

1. mkdir -p ~/work/study/2021-2022/“Операционные системы”
2. cd ~/work/study/2021-2022/“Операционные системы”
3. gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
4. git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro

Настраиваем каталог курса. Для этого переходим в него командой:

cd ~/work/study/2021-2022/“Операционные системы”/os-intro

Далее командой ls проверяем, что мы в него перешли. В каталоге “os-intro” нам потребуется удалить файл “package.json”. Выполняем данную задачу командой:

rm package.json

Снова командой ls проверяем успешное выполнение удаления файла.

Создаём необходимые каталоги и отправляем наши файлы на сервер
make COURSE=os-intro

1. `git add .`
2. `git commit -am 'feat(main): make course structure'`
3. `git push`

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?

Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам `git --version` (Проверка версии Git) `git init` (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) `git clone https://www.github.com/username/repo-name` (Скопировать существующий удаленный Git-репозиторий) `git remote` (Просмотреть список текущих удалённых репозиториях Git) `git remote -v` (Для более подробного вывода) `git add my_script.py` (Можете указать в команде конкретный файл). `git add .` (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) `git commit -am "Commit message"` (Вы можете сжать все индексированные файлы и отправить коммит). `git branch` (Просмотреть список текущих веток можно с помощью команды `branch`) `git --help` (Чтобы узнать больше обо всех доступных параметрах и командах) `git push origin master` (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветки (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Как и зачем можно игнорировать некоторые файлы при `commit`? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.