

Отчет по лабораторной работе №2

Дисциплина: Операционные системы

Астраханцева Анастасия Александровна

Содержание

1	Цель работы	4
2	Задания	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	имя и email	7
4.2	имя начальной ветки и параметры autocrlf, autocrlf	7
4.3	Генерация ключа pgr	8
4.4	Вывод списка ключей	8
4.5	Копирование pgr ключа	9
4.6	Настройка автоматических подписей коммитов git	9
4.7	Настройка автоматических подписей коммитов git	9
4.8	Создаем каталог “Операционные системы”	10
4.9	Создание репозитория	10
4.10	Удаление лишних файлов, создание каталогов	10
4.11	Отправка файлов на сервер	10
4.12	Отправка файлов на сервер	11

1 Цель работы

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

2 Задания

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ PGP.
3. Настроить подписи git.
4. Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Системы контроля версий

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.

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.

Мы будем работать с системой контроля версий Git, далее рассмотрим подробнее данную систему.

Примеры использования git

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
2. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

4 Выполнение лабораторной работы

Поскольку git уже был установлен, начинаем выполнение ЛР с того, что задаем имя пользователя и email. После этого настроим utf-8 в выводе сообщений git (рис. fig. 4.1).

```
Выполнено:
[root@aaastrakhantseva ~]# git config --global user.name "Anastasiia Astrakhantc
eva"
[root@aaastrakhantseva ~]# git config --global user.email "nastyamolot04@gmail.c
om"
[root@aaastrakhantseva ~]# git config --global core.quotepath false
[root@aaastrakhantseva ~]#
```

Рис. 4.1: имя и email

Далее задаем имя начальной ветки, будем называть ее master. Задаем параметры autocrlf, autocrlf (рис. fig. 4.2).

```
[root@aaastrakhantseva ~]# git config --global core.quotepath false
[root@aaastrakhantseva ~]# git config --global init.defaultBranch master
[root@aaastrakhantseva ~]# git config --global core.autocrlf input
[root@aaastrakhantseva ~]# git config --global core.safecrlf warn
[root@aaastrakhantseva ~]#
```

Рис. 4.2: имя начальной ветки и параметры autocrlf, autocrlf

Создаем ключ ргр. Из предложенных опций выберем: тип RSA and RSA, размер 4096, выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда) (рис. fig. 4.3).

```
[root@aaastrakhantseva ~]# gpg --full-generate-key
gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/root/.gnupg'
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
  (10) ECC (только для подписи)
  (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Anastasiia
Адрес электронной почты: nastyamolot04@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Anastasiia <nastyamolot04@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
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.
```

Рис. 4.3: Генерация ключа gpg

После этого выводим список ключей и копируем отпечаток приватного ключа. Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа. (рис. fig. 4.4)

Формат строки:

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]

ID_ключа:

```
[root@aaastrakhantseva ~]# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
-----
sec  rsa4096/5F7EC278C5FFB26C 2023-02-11 [SC]
      7E0DF74B4A0D819A40CB7DC45F7EC278C5FFB26C
uid          [ абсолютно ] Anastasiia <nastyamolot04@gmail.com>
ssb  rsa4096/99F1C76EF2185438 2023-02-11 [E]

[root@aaastrakhantseva ~]#
```

Рис. 4.4: Вывод списка ключей

После этого копируем ключ в буфер обмена (рис. fig. 4.5).

```
[root@aaastrakhantseva ~]# gpg --armor --export 5F7EC278C5FFB26C | xclip -sel cl  
ip  
[root@aaastrakhantseva ~]#
```

Рис. 4.5: Копирование ргр ключа

Используя введенный email, указываем Git применять его при подписи коммитов:(рис. fig. 4.6).

```
[root@aaastrakhantseva ~]# git config --global user.signingkey 5F7EC278C5FFB26C  
[root@aaastrakhantseva ~]# git config --global commit.gpgsign true  
[root@aaastrakhantseva ~]# git config --global gpg.program $(which gpg2)  
[root@aaastrakhantseva ~]# gh auth login
```

Рис. 4.6: Настройка автоматических подписей коммитов git

Для настройки авторизируемся (рис. fig. 4.7).

```
[aaastrakhantseva@aaastrakhantseva ~]$ gh auth login  
? What account do you want to log into? GitHub.com  
? What is your preferred protocol for Git operations? SSH  
? Upload your SSH public key to your GitHub account? /home/aaastrakhantseva/.ssh  
/id_rsa.pub  
? Title for your SSH key: home  
? How would you like to authenticate GitHub CLI? Login with a web browser  
  
! First copy your one-time code: 5TA0-1D7A  
Press Enter to open github.com in your browser...  
✓ Authentication complete.  
- gh config set -h github.com git_protocol ssh  
✓ Configured git protocol  
HTTP 422: Validation Failed (https://api.github.com/user/keys)  
key is already in use  
[aaastrakhantseva@aaastrakhantseva ~]$
```

Рис. 4.7: Настройка автоматических подписей коммитов git

Создаем каталог “Операционные системы” в каталоге ~/work/study/2022-2023 (рис. fig. 4.8).

```
[aaastrakhantseva@aaastrakhantseva ~]$ ls
work  Документы  Изображения  Общедоступные  Шаблоны
Видео  Загрузки  Музыка  'Рабочий стол'
[aaastrakhantseva@aaastrakhantseva ~]$ cd work
[aaastrakhantseva@aaastrakhantseva work]$ ls
arch-pc  study
[aaastrakhantseva@aaastrakhantseva work]$ cd study
[aaastrakhantseva@aaastrakhantseva study]$ ls
2022-2023
[aaastrakhantseva@aaastrakhantseva study]$ cd 2022-2023
[aaastrakhantseva@aaastrakhantseva 2022-2023]$ ls
'Архитектура компьютера'
[aaastrakhantseva@aaastrakhantseva 2022-2023]$ mkdir "Операционные системы"
[aaastrakhantseva@aaastrakhantseva 2022-2023]$
```

Рис. 4.8: Создаем каталог “Операционные системы”

Создаем репозиторий (рис. fig. 4.9).

```
[aaastrakhantseva@aaastrakhantseva ~]$ gh repo create study_2022-2023_os-intro -
-template=yamadharma/course-directory-student-template --public
✓ Created repository anastasiia7205/study_2022-2023_os-intro on GitHub
[aaastrakhantseva@aaastrakhantseva ~]$ git clone --recursive git@github.com:<own
er>/study_2022-2023_os-intro.git os-intro
bash: owner: Нет такого файла или каталога
[aaastrakhantseva@aaastrakhantseva ~]$ git clone --recursive git@github.com:
anastasiia7205/study_2022-2023_os-intro.git os-intro
```

Рис. 4.9: Создание репозитория

Далее переходим в каталог курса os-intro. Удаляем лишние файлы, создаем необходимые каталоги, отправим файл на сервер (рис. fig. 4.10 - fig. 4.12).

```
os-intro
[aaastrakhantseva@aaastrakhantseva Операционные системы]$ cd os-intro
[aaastrakhantseva@aaastrakhantseva os-intro]$ rm package.json
[aaastrakhantseva@aaastrakhantseva os-intro]$ echo os-intro > COURSE
[aaastrakhantseva@aaastrakhantseva os-intro]$ make
[aaastrakhantseva@aaastrakhantseva os-intro]$
```

Рис. 4.10: Удаление лишних файлов, создание каталогов

```
[aaastrakhantseva@aaastrakhantseva os-intro]$ git commit -am 'feat(main):make co
urse structure'
[master 0abac07] feat(main):make course structure
361 files changed, 100327 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
```

Рис. 4.11: Отправка файлов на сервер

```
[aaastrakhantseva@aaastrakhantseva os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.44 КиБ | 2.34 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использо-
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:anastasiia7205/study_2022-2023_os-intro.git
 1588cf7..0abac07 master -> master
[aaastrakhantseva@aaastrakhantseva os-intro]$
```

Рис. 4.12: Отправка файлов на сервер

Контрольные вопросы:

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий (Version Control System, VCS)- программное обеспечение для облегчения работы с изменяющейся информацией.

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.

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. 2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.

Хранилище — это содержимое скрытой папки .git. В этой папке хранятся все версии рабочей области и служебная информация. Этим версиям система автоматически даёт название, состоящее из букв и цифр.

Рабочая копия - копия проекта, связанная с репозиторием

commit - это команда, которая делает так, что новая версия проекта сохраняется и добавляется в хранилище. В файле с сохранением отображаются: все изменения, которые происходили в рабочей области, автор изменений и краткий комментарий, описывающий суть изменений. Каждый коммит хранит полное состояние рабочей области, её папок и файлов проекта.

История хранит все изменения проекта и позволяет при необходимости обратиться к ним.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные VCS: Одно основное хранилище всего проекта, каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Пример централизованных VCS: Subversion, CVS, TFS, VAULT.

Децентрализованные VCS: У каждого пользователя свой вариант (возможно не один) репозитория, присутствует возможность добавлять и забирать изменения из любого репозитория. Примеры децентрализованных VCS: Git, Mercurial, Bazaar.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Для начала создаем и подключаем удаленный репозиторий. По мере изменения проекта отправляем изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Пользователь получает определенные файлы, производит различные изменения, после этого загружает измененные файлы обратно на сервер. При этом старые версии не удаляются, а хранятся в хранилище, к ним всегда можно вернуться.

6. Каковы основные задачи, решаемые инструментальным средством git?

Хранение информации о всех внесенных изменениях, обеспечение удобства командной работы над проектом.

7. Назовите и дайте краткую характеристику командам 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 <имя_файла>``

удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в директории): ``git rm <имя_файла>``

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>``

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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

`git push -all` (push origin master/ любой branch)

9. Что такое и зачем могут быть нужны ветви (branches)?

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 ветвление. 10. Как и зачем можно игнорировать некоторые файлы при commit?

Во время создания проекта могут создаваться файлы, которые не нужно добавлять в репозиторий. К примеру, временные файлы, создаваемые редакторами и др. Для того, чтобы эти файлы не отправлялись в репозиторий можно прописать шаблон игнорируемых файлов в файл .gitignore с помощью сервисов.

5 Выводы

1. В ходе выполнения лабораторной работы мною были изучены идеология и применение средств контроля версий и освоены умения по работе с git.

Список литературы

1. О системе контроля версий [электронный ресурс] – Режим доступа:
<https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий>
2. Введение в системы контроля версий [электронный ресурс] - Режим доступа:
<https://htmlacademy.ru/blog/git/version-control-system>
3. Системы контроля версий Выполнил Горвиц Евгений, ВМИ-301 [электронный ресурс] - Режим доступа: <https://glebradchenko.susu.ru/courses/bachelor/engineering/20>