

Лабораторная работа №2: отчет.

работа с git.

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.

Содержание

Цель работы	4
Задание	5
Выполнение лабораторной работы	6
Установка программного обеспечения	6
Базовая настройка git	6
Создайте ключи pgr	8
Добавление PGP ключа в GitHub	9
Создайте ключи ssh	9
Настройка автоматических подписей коммитов git и Настройка gh . .	11
Создание репозитория курса на основе шаблона	12
Настройка каталога курса	13
Контрольные вопросы	14
Выводы	18
Список литературы	19

Список иллюстраций

1	установка git	6
2	Установка gh	6
3	задание базовых параметров	7
4	настройка генератора gpg	7
5	Полученные значения ключа	7
6	Смена хот клавиши	8
7	Получение значение отпечатка и его применения	8
8	Успешно применили gpg-ключ в github	9
9	генератор ssh	10
10	второй генератор ssh	10
11	указание параметров для авто подписи	11
12	авторизация gh	11
13	Запущенный ключ ssh	12
14	Создание локального репозитория	13
15	Настройка локального репозитория	13

Цель работы

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

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.

Выполнение лабораторной работы

Установка программного обеспечения

Установим git командой “dnf install git” и Установка gh “dnf install gh” на нашу систему Fedora:

```
root@evdokimov: ~$ sudo -i
[sudo] пароль для max:
[root@evdokimov ~]# dnf install git
Fedora 39 - x86_64 - Updates                                65 kB/s | 20 kB  00:00
Fedora 39 - x86_64 - Updates                                1.2 MB/s | 1.8 MB 00:01
Последняя проверка окончания срока действия метаданных: 0:00:14 назад, Сб 23 дек 2023 17:13:19.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
-----
Установка:
git                x86_64      2.43.0-1.fc39 updates      54 k
Установка зависимостей:
git-core          x86_64      2.43.0-1.fc39 updates      4.5 M
git-core-doc      noarch     2.43.0-1.fc39 updates      2.9 M
perl-Error        noarch     1.0.17029-13.fc39 fedora       48 k
perl-git          noarch     2.42.0-1.fc39 updates      40 k
perl-TestReadKey  x86_64     2.38-18.fc39 fedora       35 k
perl-lib          x86_64     0.65-502.fc39 updates      15 k
=====
Результат транзакции
=====
Установка 7 Пакетов
```

Рис. 1: установка git

```
root@evdokimov: ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:01:02 назад, Сб 23 дек 2023 17:13:19.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
-----
Установка:
gh                x86_64      2.40.1-1.fc39 updates      9.1 M
=====
Результат транзакции
=====
Установка 1 Пакет
```

Рис. 2: Установка gh

Базовая настройка git

1. Зададим имя и email владельца репозитория используя команды “git config –global user.name”имя”” и “git config –global user.email”почта”“, а также

настроим utf-8 в выводе сообщений git”git config –global core.quotePath false”:

```
[root@evdokimov ~]# git config --global user.name "Kerreduen"
[root@evdokimov ~]# git config --global user.email "sam33848@gmail.com"
[root@evdokimov ~]# git config --global core.quotePath false
```

Рис. 3: задание базовых параметров

2. Настройте верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG).

```
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: Kerreduen
Адрес электронной почты: sam33848@gmail.com
Примечание: Python is convenient
Вы выбрали следующий идентификатор пользователя:
"Kerreduen (Python is convenient) <sam33848@gmail.com>"
```

Рис. 4: настройка генератора gpg

```
gpg: /root/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/root/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/root/.gnupg/openpgp-revocs.d/ACF6A58F25200D4CA7EBF66AD081DA1B775BB615.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-12-23 [SC]
      ACF6A58F25200D4CA7EBF66AD081DA1B775BB615
uid           Kerreduen (Python is convenient) <sam33848@gmail.com>
sub   rsa4096 2023-12-23 [E]
```

Рис. 5: Полученные значения ключа

3. Зададим имя начальной ветки на master командой “git config –global init.defaultBranch master” и настроим другие параметры autocrlf - “git config –global core.autocrlf input” и safecrlf - “git config –global core.safecrlf warn”

```
Установлен:
  xclip-0.13-20.git11cba61.fc39.x86_64

Выполнено!
[root@evdokimov ~]# gpg --armor --export ACF6A58F25200D4CA7EBF66AD081DA1B775BB615 | xclip -sel clip
[root@evdokimov ~]# git commit -a -S -m "Python forever! Max"
fatal: не найден git репозиторий (или один из родительских каталогов): .git
[root@evdokimov ~]# git --global init.defaultBranch master
unknown option: --global
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        [--config-env=<name>=<envvar>] <command> [<args>]
[root@evdokimov ~]# git config --global init.defaultBranch master
[root@evdokimov ~]# git config --global core.autocrlf input
[root@evdokimov ~]# git config --global core.safecrlf warn
[root@evdokimov ~]#
```

Рис. 6: Смена хот клавиши

Создайте ключи ргр

Генерируем ключ командой “gpg –full-generate-key” в опциях указываем такие значения: тип RSA and RSA; размер 4096; выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда); указываем личную информацию которую запрашивает GPG, которая сохранится в ключе: Имя (не менее 5 символов) и Адрес электронной почты (соответствующий GitHub). В комментарий можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
[root@evdokimov ~]# 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
[keyboard]
-----
sec   rsa4096/D081DA1B775BB615 2023-12-23 [SC]
      ACF6A58F25200D4CA7EBF66AD081DA1B775BB615
uid   [ абсолютно ] Kerreduen (Python is convenient) <sam33848@gmail.com>
ssb   rsa4096/4F833F5A21B4658C 2023-12-23 [E]
```

Рис. 7: Получение значение отпечатка и его применения

Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа с помощью команды “`gpg –list-secret-keys –keyid-format LONG`” имеющий такой формат:

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до] ID_ключа
Скопируйте ваш сгенерированный PGP ключ в буфер обмена:

После чего копируем ключ и вводим в github “`gpg –armor –export ACF6A58F25200D4CA7EBF66AD081DA1B775BB615 | xclip -sel clip`”

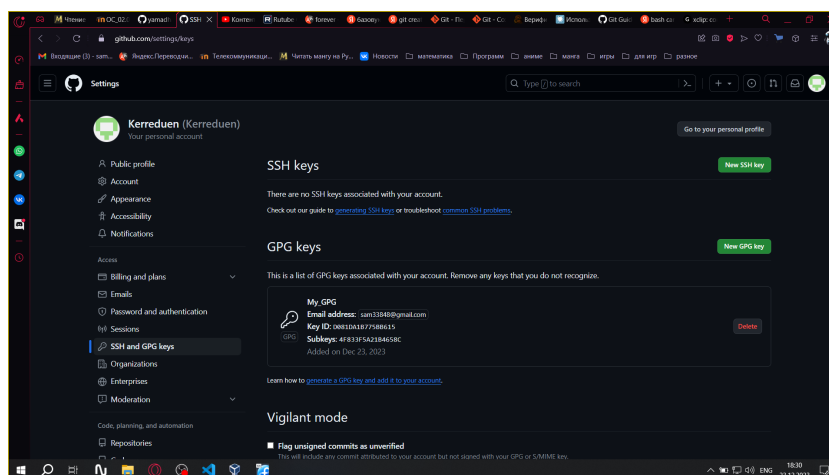


Рис. 8: Успешно применили gpg-ключ в github

Создайте ключи ssh

1. По алгоритму rsa с ключём размером 4096 бит используем команду “`ssh-keygen -t rsa -b 4096`” и следуем указанным инструкциям получаем ssh-ключ:

```

[root@evdokinov ~]# gpg --armor --export AC6A58F252004CA7EBF66AD081DA1B775BB615 | xclip -sel clip
[root@evdokinov ~]# git commit -a -S -m "Python forever! Max"
fatal: не найден git репозиторий (или один из родительских каталогов): .git
[root@evdokinov ~]# git --global init.defaultBranch master
unknown option: --global
usage: git [-v] [--version] [-h] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path<path>] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir<path>] [--work-tree<path>] [--namespace<name>]
        [--config-env=<name>=<envvar>] <command> [<args>]
[root@evdokinov ~]# git config --global init.defaultBranch master
[root@evdokinov ~]# git config --global core.autocrlf input
[root@evdokinov ~]# git config --global core.safecrlf warn
[root@evdokinov ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): 335728forGit
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in 335728forGit
Your public key has been saved in 335728forGit.pub
The key fingerprint is:
SHA256:p5L7YH7Ho/AdTURicNoQn+LA1RKR2hZ890TP01JL7m root@evdokinov
The key's randomart image is:
+--[RSA 4096]--+
|  .+ .+ + o |
|  + . o * B.. |
| o o . O.+ |
| .+ . o + |
| . . o S o o E |
| . . +oo+ o |
| .+ o+oo. |
| o +.o. |
| .+. |
+-----[SHA256]-----+
[root@evdokinov ~]#

```

Рис. 9: генератор ssh

- Повторяем генерацию, но по алгоритму ed25519 командой “ssh-keygen -t ed25519”:

```

[root@evdokinov ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): 335728forGit
335728forGit already exists.
Overwrite (y/n)? n
[root@evdokinov ~]# ssh-copy-id -i ~/.ssh/mykey user@host~ | xclip -sel clip
[root@evdokinov ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): saymyname
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in saymyname
Your public key has been saved in saymyname.pub
The key fingerprint is:
SHA256:B7aPi0Lt2KJuu9bJvtke1Q8+M52mxvXjW0+x5c+oohA root@evdokinov
The key's randomart image is:
+--[ED25519 256]--+
|
|      o
|      . +
|    .E S +
|    . O = +..
|  + =O . . * . = . . .
| o Boo+ . + * oo = +
| += += + + oo . oo o + B
+-----[SHA256]-----+

```

Рис. 10: второй генератор ssh

Настройка автоматических подписей коммитов git и

Настройка gh

1. Используя введённый email, укажите Git применять его при подписи коммитов:

```
[root@evdokimov ~]# git config --global user.signinkey ACF6A58F2520D4CA7EBF66AD081DA1B775BB615
[root@evdokimov ~]# git config --global commit.gpgsign true
[root@evdokimov ~]# git config --global gpg.program $(which gpg2)
[root@evdokimov ~]#
```

Рис. 11: указание параметров для авто подписи

2. Для начала необходимо авторизоваться “gh auth login” Утилита задаст несколько наводящих вопросов на которые мы отвечаем, после чего я авторизуюсь через браузер, так как в консоли не так удобно.

```
[root@evdokimov ~]# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /root/.ssh/id_ed25519.pub
? Title for your SSH key? Operations Notes
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 10D9-458D
Press enter to open github.com in your browser.
restorecon: SELinux: Could not get canonical path for /root/.mozilla/firefox/* restorecon: No such file or directory
Running Firefox as root in a regular user's session is not supported. ($XAUTHORITY is /run/lightdm/max/xauthority which is owned by max.)
/usr/bin/xdg-open: строка 881: x-www-browser: команда не найдена
restorecon: SELinux: Could not get canonical path for /root/.mozilla/firefox/* restorecon: No such file or directory
Running Firefox as root in a regular user's session is not supported. ($XAUTHORITY is /run/lightdm/max/xauthority which is owned by max.)
/usr/bin/xdg-open: строка 881: iceweasel: команда не найдена
/usr/bin/xdg-open: строка 881: seamonkey: команда не найдена
/usr/bin/xdg-open: строка 881: mozilla: команда не найдена
/usr/bin/xdg-open: строка 881: epiphany: команда не найдена
/usr/bin/xdg-open: строка 881: konqueror: команда не найдена
/usr/bin/xdg-open: строка 881: chromium: команда не найдена
[3938.3938:1223/190853.438849:ERROR:zygote_host_impl_linux.cc(100)] Running as root without --no-sandbox is not supported. See https://crbug.com/638180
[3954.3954:1223/190853.521516:ERROR:zygote_host_impl_linux.cc(100)] Running as root without --no-sandbox is not supported. See https://crbug.com/638180
/usr/bin/xdg-open: строка 881: www-browser: команда не найдена
/usr/bin/xdg-open: строка 881: links2: команда не найдена
/usr/bin/xdg-open: строка 881: elinks: команда не найдена
/usr/bin/xdg-open: строка 881: lynx: команда не найдена
```

Рис. 12: авторизация gh

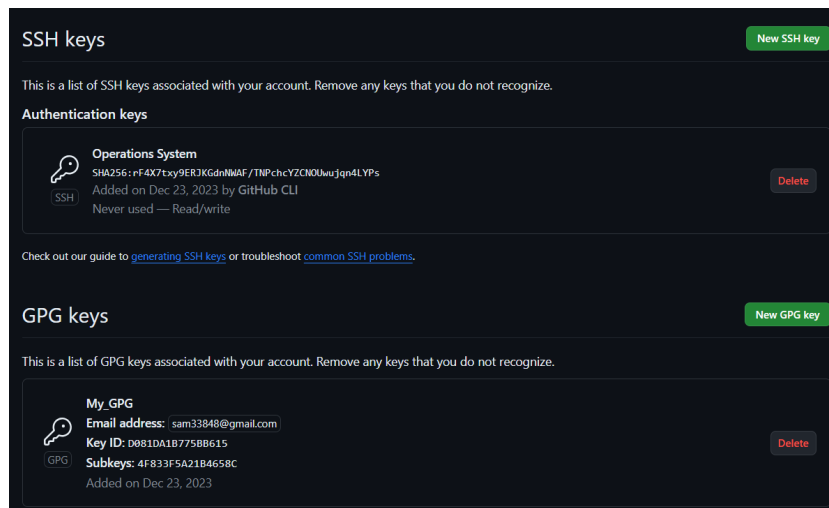


Рис. 13: Запушенный ключ ssh

Создание репозитория курса на основе шаблона

Создаём шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы) для 2022–2023 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид (выполняем все команды):

```
mkdir -p ~/work/study/2022-2023/"Операционные системы"
cd ~/work/study/2022-2023/"Операционные системы"
gh repo create study_2022-2023_os-intro --template=yamadharma/course-
directory-student-template --public
git clone --recursive git@github.com:<owner>/study_2022-2023_os-
intro.git os-intro
```

```

Authentication credentials saved in plain text
Uploaded the SSH key to your GitHub account: /root/.ssh/id_ed25519.pub
Logged in as Kerreduen
[root@evdokinov ~]# mkdir -p ~/work/study/2022-2023/"Операционные системы"
[root@evdokinov ~]# cd -p ~/work/study/2022-2023/"Операционные системы"
-bash: cd: -p: недопустимый параметр
cd: использование: cd [-L][-P][-e]] [-O]] [каталог]
[root@evdokinov ~]# cd ~/work/study/2022-2023/"Операционные системы"
[root@evdokinov Операционные системы]# gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository Kerreduen/study_2022-2023_os-intro on GitHub
[root@evdokinov Операционные системы]# git clone git@github.com:Kerreduen/study_2022-2023_os-intro.git os-intro
-bash: owner: Нет такого файла или каталога
[root@evdokinov Операционные системы]# git clone --recursive git@github.com:Kerreduen/study_2022-2023_os-intro.git os-intro
Клонирование в os-intro...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DIY3wvvV6TujHbp7isF/zLDAbzPMSvhdKx4UvcOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/root/.ssh/id_ed25519':
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 38 (delta 1), reused 17 (delta 0), pack-reused 0
Получение объектов: 100% (38/38), 17.75 Кб | 599.00 Кб/с, готово.
Определение изменений: 100% (11), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в ~/root/work/study/2022-2023/Операционные системы/os-intro/template/presentation...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 Кб | 632.00 Кб/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в ~/root/work/study/2022-2023/Операционные системы/os-intro/template/report...
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 112 (delta 45), reused 98 (delta 31), pack-reused 0
Получение объектов: 100% (112/112), 331.19 Кб | 865.00 Кб/с, готово.
Определение изменений: 100% (45/45), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '25e169d367953f60c76c251db299ed52852b401f'
[root@evdokinov Операционные системы]# cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[root@evdokinov os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? y
[root@evdokinov os-intro]# echo os-intro > COURSE
[root@evdokinov os-intro]# make
[root@evdokinov os-intro]# git add
[root@evdokinov os-intro]# git commit -am 'feat(main): make course structure'
error: gpg failed to sign the data:
gpg: получено "Kerreduen <sam33848@gmail.com>". Нет секретного ключа
[GNUPG:] INY_SIGN 9 Kerreduen <sam33848@gmail.com>
[GNUPG:] FAILURE sign 17
gpg: signing failed: Нет секретного ключа
fatal: сбоя записи объекта коммита
[root@evdokinov os-intro]# git push
Enter passphrase for key '/root/.ssh/id_ed25519':
Everything up-to-date

```

Рис. 14: Создание локального репозитория

Настройка каталога курса

Перейдя в каталог курса командой “`cd ~/work/study/2022-2023/"Операционные системы"/os-intro`” удаляем лишние файлы командой “`rm package.json`” и Создаём необходимые каталоги “`echo os-intro > COURSE`” после применяем изменения “`make`”, и отправляем файлы на сервер последовательность команд “`git add .`”, “`git commit -am 'feat(main): make course structure'`”, “`git push`”:

```

Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в ~/root/work/study/2022-2023/Операционные системы/os-intro/template/presentation...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 Кб | 632.00 Кб/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в ~/root/work/study/2022-2023/Операционные системы/os-intro/template/report...
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 112 (delta 45), reused 98 (delta 31), pack-reused 0
Получение объектов: 100% (112/112), 331.19 Кб | 865.00 Кб/с, готово.
Определение изменений: 100% (45/45), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '25e169d367953f60c76c251db299ed52852b401f'
[root@evdokinov Операционные системы]# cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[root@evdokinov os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? y
[root@evdokinov os-intro]# echo os-intro > COURSE
[root@evdokinov os-intro]# make
[root@evdokinov os-intro]# git add
[root@evdokinov os-intro]# git commit -am 'feat(main): make course structure'
error: gpg failed to sign the data:
gpg: получено "Kerreduen <sam33848@gmail.com>". Нет секретного ключа
[GNUPG:] INY_SIGN 9 Kerreduen <sam33848@gmail.com>
[GNUPG:] FAILURE sign 17
gpg: signing failed: Нет секретного ключа
fatal: сбоя записи объекта коммита
[root@evdokinov os-intro]# git push
Enter passphrase for key '/root/.ssh/id_ed25519':
Everything up-to-date

```

Рис. 15: Настройка локального репозитория

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Version Control System - создано для совместной работы нескольких человек над одним проектом находящемся в локальном или удаленном репозитории.

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

Хранилище - это особая система хранения данных которая после обновления данных сохранять предыдущие состояния файла или директории для просмотра изменений и возможности возвращения к предыдущему состоянию.

Commit - это команда для идентификации изменений файлов, так он храним как и новое состояние так и все предыдущие сохранённые.

История - это система сохраняющая все внесенные в файл или ветвь изменения которая позволяет просмотреть кем и когда было совершено то или иное изменения, а также сохраняя общую историю изменений до точки ветвления версий.

Рабочая копия - это отдельно сохранённая в системе версия ветви или всего проекта и в зависимости от настроек которая не позволяет другим участникам вносить изменения в выбранный сегмент или версию.

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

Централизованный характеризуется тем что есть только один общий репозиторий (проект) в котором в зависимости от настроек и уровня прав каждый может вносить изменения и структура или файлы. Пример: Subversion и Perforce - Централизованные СКВ. А детерминированный выражается в том что у каждого пользователя есть своя копия исходного репозитория которым они могут пользоваться независимо (даже офлайн) и после уже вносить изменения в серверный (исходный) из которого другие участники могут получить обновление своих локальных копий. Пример: RCS - локальное СКВ (1985).

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

Когда пользователь является единоличным пользователем хранилища (владельцем например) он в целом самостоятельно регулировать процесс работы с ним, и всё что ему требуется это соблюдать общие правила и соблюдать базовые принципы работы с VCS: Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).

5. Опишите порядок работы с общим хранилищем VCS.

Он схож с единоличным за исключением некоторых отличий: нужно проверить и объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику, предостережительно обсудив или предупредив других пользователей.

6. Каковы основные задачи, решаемые инструментальным средством git?

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями что позволяет нам избегать различные различные ошибки за счёт сохранения предыдущих версий, а также совместно работать над

одним проектом не боясь испортить чужую работу посредством создания новых ветвей и версии.

7. Назовите и дайте краткую характеристику командам git.

1)config - позволяет изменять базовые настройки git а так её взаимодействия с системой. 2)init - отвечает за создание новых репозиториях. 3)add - добавление указанных или всех файлов в актуальный репозиторий. 4)status - позволяет узнать и просмотреть статус репозитория. 5)commit - внесение изменений через однострочные сообщения или через редактор. 6)log - позволяет просматривать историю коммита. 7)show - просмотр указанного коммита. 8)diff - просмотр не подготовленных для фиксации коммитов. 9)rm - удаление указанных файлов или ветвей. 10)mv - переименовывание файлов, ветвей. 11)checkout - отмена актуальных, последних или указанных изменений. 12)reset - восстановление подготовленных файлов. 13)commit –amend - изменение последнего коммита. 14)revert - откат последнего коммита. 15)branch - создание новой ветви. 16)merge - слияние ветвей. 17)remote - восстановление удалённого репозитория, файла, ветви. 18)push - отправка изменений в удаленный (общий) репозиторий. 19)pull - получение последних версии, изменений из удалённого репозитория. 20) rebase - переназначает словно изменяет коммиты с одной ветви на другую (копирует в новую директорию).

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

Допустим, нужно добавить в проект новый файл “file.txt”. Если мы работаем с локальным репозиторием то достаточно Добавим файл в локальный репозиторий “git add file.txt” (файл лежит в том же каталоге, что и репозиторий) Сохранив изменения. Если с удалённым то загрузим нужную версию из репозитория “git checkout last” (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним

изменения: `git commit -am "file.txt was added"` Отправим изменения в удалённый репозиторий: `git push`.

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

Ветка в Git — это набор коммитов, расположенных в хронологическом порядке. У каждой ветки есть свое название. Основная ветка чаще всего называется `master`, она появляется при инициализации репозитория и считается главной веткой проекта. Другим веткам вы даете имена самостоятельно. Дополнительные ветки используются для создания нового функционала и исправления ошибок. То есть ветки это “перестраховка” для того чтобы не изменять раньше времени или не вызывать конфликты с изменениями других в системе.

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

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять впоследствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, а также некоторые тестовые и личные данные.

Выводы

В ходе работы произведена установка и настройка всех необходимых программ и утилит для работы с git и github, в особенности по настройке системы ssh и pgr.

Список литературы

1. Лабораторная работа №2
2. Справочник по настройке git
3. Введение git
4. Список стандартных команд git