Отчёт по лабораторной работе №2

Управление версиями

Гайсина Алина Ринатовна

Содержание

0.1 Цель работы:

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

0.2 Задание

Выполнить отчёт к предыдущей лабораторной работе.

1 Ход работы

1. Создала учетную запись на GitHub.

2. Вручную установила программное обеспечение git-flow.(рис.1)

```
[argayjsina@fedora ~]$ cd /tmp
[argayjsina@fedora tmp]$ wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib
/gitflow-installer.sh
[argayjsina@fedora tmp]$ chmod +x gitflow-installer.sh
[argayjsina@fedora tmp]$ sudo ./gitflow-installer.sh install stable
```

рис. 1: Установка программного обеспечения

3. Установка gh. (рис.2)

```
[argayjsina@fedora tmp]$ sudo dnf install gh
Fedora 35 - x86_64                              50% [==============
```

рис. 2: Установка gh

4. Задала имя и email владельца репозитория. (рис.3)

```
[argayjsina@fedora tmp]$ git config --global user.name "Alina Gaysina"
[argayjsina@fedora tmp]$ git config --global user.email "gajsinaalina858@gmail.com"
[argayjsina@fedora tmp]$ git config --global core.quotepath false
```

рис. 3: Имя и email владельца репозитория

5. Настроила utf-8 в выводе сообщений git. (рис.4)

```
[argayjsina@fedora tmp]$ git config --global init.defaultBranch master
[argayjsina@fedora tmp]$ git config --global core.autocrlf input
[argayjsina@fedora tmp]$ git config --global core.safecrlf warn
```

рис. 4: Настройка utf-8 в выводе сообщений git

6. Настроила верификацию и подписание коммитов git и задала имя начальной ветки (master). (рис.5)

```
[argayjsina@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/argayjsina/.ssh/id_rsa):
Created directory '/home/argayjsina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/argayjsina/.ssh/id_rsa
Your public key has been saved in /home/argayjsina/.ssh/id_rsa.pub
The key fingerprint is:
```

рис. 5: Коммиты и начальная ветка

7. Создание ключа ssh. (рис.6, 7)

```
[argayjsina@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/argayjsina/.ssh/id_ed25519):
/home/argayjsina/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/argayjsina/.ssh/id_ed25519
Your public key has been saved in /home/argayjsina/.ssh/id_ed25519.pub
```

рис. 6: Создание ключа ssh.

```
[argayjsina@fedora tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; 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.

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
```

рис. 7: Создание ключа ssh.

8. Создала ключ pgp и добавила его в GitHub. (рис.8-11)

```
Ваше полное имя: Alina Gaysina
Адрес электронной почты: gajsinaalina858@gmail.com
```

рис. 8: Генерация ключа pgp

```
[argayjsina@fedora tmp]$ 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
/home/argayjsina/.gnupg/pubring.kbx
---------------------------------
sec   rsa4096/D57F652A8C59F102 2022-06-13 [SC]
      0CE55F3B740BEC91391C5E7CD57F652A8C59F102
uid                 [  абсолютно ] Alina Gaysina <gajsinaalina858@gmail.com>
ssb   rsa4096/83BAF140FF4518C8 2022-06-13 [E]
```

рис. 9: Вывод ключа pgp

```
[argayjsina@fedora tmp]$ gpg --armor --export D57F652A8C59F102 | xclip -sel clip
[argayjsina@fedora tmp]$
```

рис. 10: Копирование ключа pgp

рис. 11: Добавление ключа pgp на сайт GitHub

9. Настроила автоматические подписи коммитов git. (рис.12)

```
[argayjsina@fedora tmp]$ git config --global user.signingkey D57F652A8C59F102
[argayjsina@fedora tmp]$ git config --global commit.gpgsign true
[argayjsina@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

рис. 12: Автоматические подписи коммитов git

10. Настроила gh. (рис.13)

```
[argayjsina@fedora tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: B68B-D61A
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as aliinna
```

рис. 13: Настройка gh

11. Создала репозиторий курса на основе шаблона. (рис.14-16)

```
[argayjsina@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[argayjsina@fedora tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
```

рис. 14: Создание директории “Операционные системы”

```
[argayjsina@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directo
ry-student-template --public
✓ Created repository aliinna/study_2021-2022_os-intro on GitHub
```

рис. 15: Создание репозитория на GitHub

```
[argayjsina@fedora Операционные системы]$ git clone --recursive git@github.com:aliinna/study_2021-2022_os-intro.git os
-intro
Клонирование в «os-intro»…
```

рис. 16: Копирование шаблона в мой репозиторий

12. Настроила каталоги курса. (рис.17-19)

```
[argayjsina@fedora Операционные системы]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
[argayjsina@fedora os-intro]$ rm package.json
[argayjsina@fedora os-intro]$ make COURSE=os-intro
```

рис. 17: Настройка

```
[argayjsina@fedora os-intro]$ git add .
[argayjsina@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master 92bbe40] feat(main): make course structure
```

рис. 18: Настройка

```
[argayjsina@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.53 КиБ | 1.21 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:aliinna/study_2021-2022_os-intro.git
   89a505e..92bbe40  master -> master
```

рис. 19: Настройка

2 Выводы

Изучила идеологию и применение средств контроля версий и освоила умениярпо работе с git.

3 Контрольные вопросы

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

- Система контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

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

- Хранилище – репозиторий - место хранения всех версий и служебной информации.
- Commit - это команда для записи индексированных изменений в репозиторий.
- 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 системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример – Subversion.
- Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев. Пример – Git.

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

- В рабочей копии, которую исправляет человек, появляются правки, которые отправляются в хранилище на каждом из этапов. То есть в правки в рабочей копии появляются, только если человек делает их (отправляет их на сервер) и никак по-другому .

5. Опишите порядок работы с общим хранилищем VCS.

- Если хранилище общее, то в рабочую копию каждого, кто работает над проектом, приходят изменения, отправленные на сервер одним из команды. Рабочая правка каждого может изменяться вне зависимости от того, делает ли конкретный человек правки или нет.

6. Каковы основные задачи, решаемые инструментальным средством git?

- У Git две основных задачи: первая — хранить информацию обо 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.

- 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 изменения: 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
- удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов
- 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, она будет создана и связана с удалённой)
- отправка изменений конкретной ветки в центральный репозиторий: 1 git push origin имя_ветки
- слияние ветки с текущим деревом: 1 git merge —no-ff имя_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.

- Работа с удаленным репозиторием: git remote – просмотр списка настроенных удаленных репозиториев.

- Работа с локальным репозиторием: git status - выводит информацию обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки

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

- Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. 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. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл . Временно игнорировать изменения в файле можно командой git update-index-assumeunchanged