Шаблон отчёта по лабораторной работе

Простейший вариант

Василий Юрьевич Кузнецов

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Указания к лабораторной работе

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды gitc различными опциями.

3 Последовательность выполнения работы.

Настройка github 1. Создайте учётную запись на https://github.com. 2. Заполните основные данные на https://github.com.

(рис. 3.1)

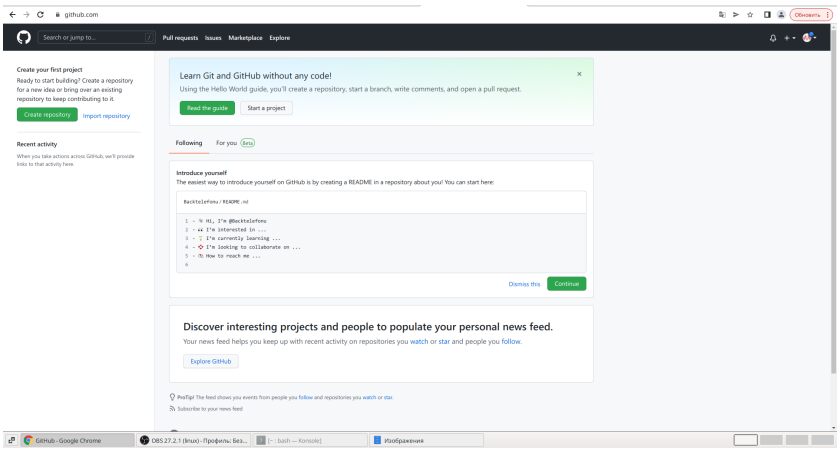

Рис. 3.1: Регистрация на Гитхаб

4 Базовая настройка git

–Зададим имя и email владельца репозитория: git config–global user.name “Name Surname” git config –global user.email “work@mail”

(рис. 4.1)

```
vykuznecov@dk3n31 ~ $ git config --global user.name "Vasiliy Kuznetsov"
vykuznecov@dk3n31 ~ $ git config --global user.email "vasya-kuznetsov-1997@mail.ru"
vykuznecov@dk3n31 ~ $
```

Рис. 4.1: Имя и мейл репозитория

Настроим верификацию и подписание коммитов git. –Зададим имя начальной ветки (будем называть её master):

(рис. 4.2)

```
vykuznecov@dk3n31 ~ $ git config --global core.quotpeth false
vykuznecov@dk3n31 ~ $ git config --global init.defaultBranch master
git config --global core.autocrlf inpvykuznecov@dk3n31 ~ $ git config --global core.autocrlf i
nput
vykuznecov@dk3n31 ~ $ git config --global core.autocrlf input
vykuznecov@dk3n31 ~ $ git config --global core.safecrlf warn
vykuznecov@dk3n31 ~ $
```

Рис. 4.2: Задаем имя ветки и параметры

5 Создание ключей ssh

Генерируем ключ

(рис. 5.1)

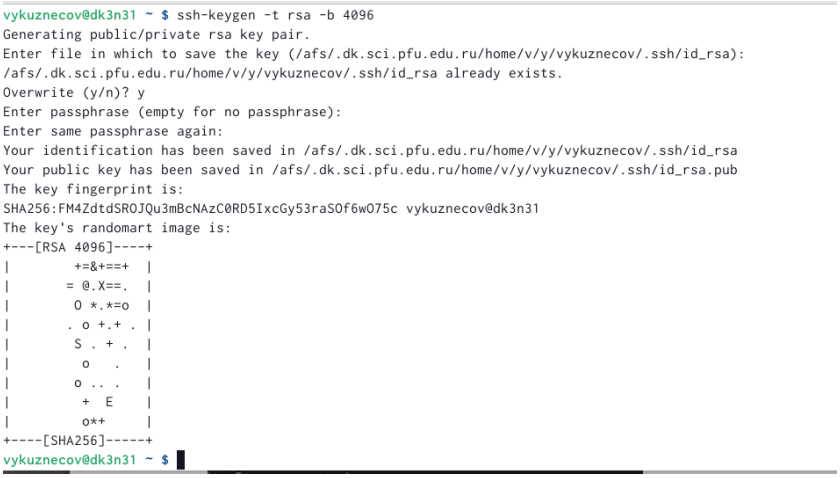

Рис. 5.1: Ключ ssh

Из предложенных опций выбираем: –тип RSAandRSA; –размер 4096; –выберите срок действия; значение по умолчанию—0 (срок действия не истекает никогда). –GPGзапросит личную информацию, которая сохранится в ключе: –Имя (не менее 5 символов). –Адрес электронной почты. –При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. –Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

(рис. 5.2)

```
vykuznecov@dk3n31 ~ $ 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
/afs/.dk.sci.pfu.edu.ru/home/v/y/vykuznecov/.gnupg/pubring.kbx
------------------------------------------------------------------
sec   rsa4096/69E0CC67B4FF5EC7 2022-04-27 [SC]
      235975E8A23DBB3B7432A90669E0CC67B4FF5EC7
uid                 [  абсолютно ] Vasiliy <vasya-kuznetsov-1997@mail.ru>
ssb   rsa4096/08390371D8790EF2 2022-04-27 [E]
```

Рис. 5.2: Опции ключа

6 Добавление PGPключа в GitHub

–Выводим список ключей и копируем отпечаток приватного ключа:

gpg–list-secret-keys–keyid-format LONG

рис. 6.1)

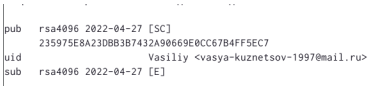

Рис. 6.1: Вывод ключа

Перейдём в настройки GitHub(https://github.com/settings/keys), нажмём на кнопку New GPG key и вставим полученный ключ в поле ввода.

(рис. 6.2)

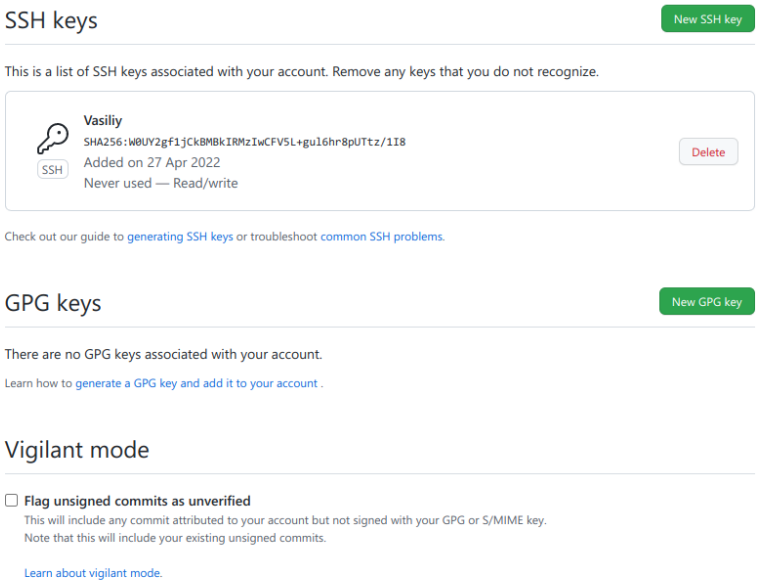

Рис. 6.2: Добавление ключа GPG

7 Создание репозитория

Шаблон для рабочего пространства –Репозиторий: https://github.com/yamadharma/course-directory-student-template.

Создание репозитория примет следующий вид: mkdir-p~/work/study/2021-2022/“Операционные системы” cd~/work/study/2021-2022/“Операционные системы” gh repo create study_2021-2022_os-intro –template=yamadharma/course-directory-student-template –public git clone –recursive git@github.com:/study_2021-2022_os-intro.git os-introКоманда gh оказалась недоступной для использования, поэтому мы пошли другим путём:Клонируем репозиторий, представленный в виде шаблона.

Команда gh оказалась недоступной для использования, поэтому мы пошли другим путём: Клонируем репозиторий, представленный в виде шаблона

Далее, аналогично ключамGPG впишем ключи SSH:

(рис. 7.1)

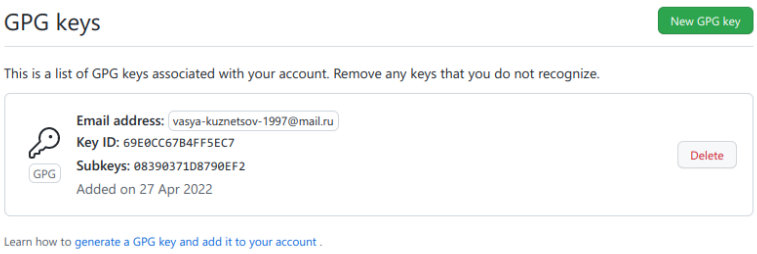

Рис. 7.1: Добавление первых ключей

(рис. 7.2)

```
vykuznecov@dk3n31 ~/work/study/2021-2022/Операционные системы $ git clone --recursive git@github.com:Backtelefonu/study_2021-2022_os-intro.git os-intro
Клонирование в «os-intro»…
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (20/20), 12.49 КиБ | 6.24 МиБ/с, готово.
Определение изменений: 100% (2/2), 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 «/afs/.dk.sci.pfu.edu.ru/home/v/y/vykuznecov/work/study/2021-2022/Операционные системы/os-intro/template/presentation»…
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 КиБ | 1.04 МиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/v/y/vykuznecov/work/study/2021-2022/Операционные системы/os-intro/template/report»…
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КиБ | 456.00 КиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Submodule path 'template/presentation': checked out '3eaebb7586f8a9aded2b506cd1018e625b228b93'
Submodule path 'template/report': checked out 'df7b2ef80f8def3b9a496f869527469a1a7842a'
```

Рис. 7.2: Клонирование файлов в репозитория

8 Настройка автоматических подписей коммитов

git–Используя введёный email, укажемGitприменять его при подписи коммитов: git config–global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2)

(рис. 8.1)

Рис. 8.1: Нахождение новых файлов в папках

(рис. 8.2)

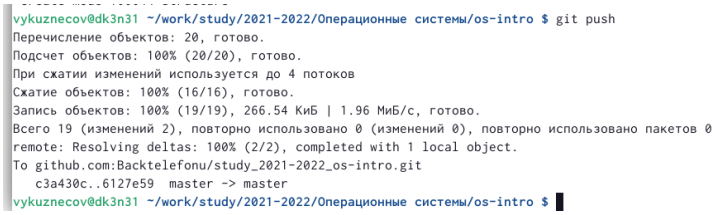

Рис. 8.2: Отправка файлов в репозиторий

9 Вывод

–Изучилиидеологию и применение средств контроля версий. –Освоилиумения по работе с git.

10 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.
2. Объясните следующие понятия VCSи их отношения: хранилище, commit, история, рабочая копия. Репозиторий -хранилище версий -в нем хранятся все документы вместе с историей их изменения и другой служебной информацией Commit(«[трудовой] вклад», не переводится) —процесс создания новой версииРабочая копия (workingcopy) —текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).Версия (revision), или ревизия, —состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы —это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. (Пример —Wikipedia.) В децентрализованных системах каждый узел принимает свое собственное решение. Конечное

поведение системы является совокупностью решений отдельных узлов. (Пример —Bitcoin)

4. Опишите действия с VCSпри единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git? У Gitесть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам git. –создание основного дерева репозитория: gitinit–получение обновлений (изменений) текущего дерева из центрального репозитория: gitpull–отправка всех произведённых изменений локального дерева в центральный репозиторий: gitpush–просмотр списка изменённых файлов в текущей директории: gitstatus–просмотр текущих изменения: gitdiff–сохранение текущих изменений: –добавить все изменённые и/или созданные файлы и/или каталоги: gitadd. –добавить конкретные изменённые и/или созданные файлы и/или каталоги: gitaddимена_файлов –удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): gitrmимена_файлов.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями. Локальный репозиторий—она же директория ".git". В ней хранятся коммиты и другие объекты.Удаленный репозиторий–тот самый репозиторий который считается общим, в который вы можете передать свои коммитыиз локального репозитория, что бы остальные программисты могли их увидеть. Удаленных репозиториев может быть несколько, но обычно он бывает один.
9. Что такое и зачем могут быть нужны ветви (branches)? 'Gitbranch' –это команда для управления ветками в репозитории Git. Ветка –это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты,

указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта.Ветвление («ветка», branch) —один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.

10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы обычно представляют собойфайлы, специфичные для платформы, или автоматически созданные из сборочных систем. Временно игнорировать изменения в файле можно командой: gitupdate-index—assume-unchanged