Лабораторная работа №2

Управление версиями

Латыпова Диана/НФИбд-02-21

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

Сначала я создала учетную запись на https://github.com.

При создании аккаунта в гит на моем компьютере произошла ошибка, поэтому я создала учетную запись на чужом компьютере (рис. **??**). Затем вошла в аккаунт со своего компьютера и заполнила основную информацию (рис. **??**).

Ошибка создания учетной записи в git

Вход в аккаунт dlatypova, заполнение основной информации.

Далее я установила вручную программное обеспечение git-flow в терминале с помощью команд:

1 **cd /tmp**

2 **wget–no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/con**
installer.sh

3 **chmod +x gitflow-installer.sh** 4 **sudo ./gitflow-installer.sh install stable**

И ввела пароль (рис. **??**).

Установка программного обеспечения git-flow

Далее установила gh (рис. **??**).

Установка gh

После чего занялась базовой настройкой git (рис. **??**). Задала имя и email владельца репозитория:

1 **git config –global user.name “Name Surname”**

2 **git config –global user.email “work@mail”**

Настроила utf-8 в выводе сообщений git:

git config –global core.quotepath false

Настроила верификацию и подписание коммитов git и задала имя начальной ветки: **git config –global init.defaultBranch master** - Параметр autocrlf:

git config –global core.autocrlf input - Параметр safecrlf:

git config –global core.safecrlf warn

Базовая настройка git

Создала ssh ключ :

– по алгоритму rsa с ключём размером 4096 бит(рис. **??**):

ssh-keygen -t rsa -b 4096

– по алгоритму ed25519(рис. **??**):

ssh-keygen -t ed25519

Создание ssh ключа по алгоритму rsa

Создание ssh ключа по алгоритму ed25519

Создала ключи pgp(рис. **??**)(рис. **??**)(рис. **??**):

Генерировала ключ **gpg –full-generate-key**

Из предложенных опций выбрала:

– тип RSA and RSA;

– размер 4096;

–срок действия; значение по умолчанию— 0 (срок действия не истекает никогда).

– GPG запросил личную информацию, которая сохранилась в ключе:

– Имя (Diana Latypova).

– Адрес моей электронной почты.

– Комментарий (оставила пустым).

Создание ключа pgp

Ввод фразы-пароля

Продолжение создания ключа pgp

Далее нужно было добавить pgp ключ в GitHub (рис. **??**). Вывела список ключей и скопировала отпечаток приватного ключа:

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

Скопировала сгенерированный PGP ключ в буфер обмена:

gpg –armor –export < PGP Fingerprint > | xclip -sel clip

Вывод списка ключей и копирование сгенерированного PGP ключа в буфер обмена

После чего перешла в настройки GitHub (https://github.com/settings/keys), нажала на кнопку New GPG key и вставила полученный ключ в поле ввода (рис. **??**).

Добавление полученного ключа pgp в поле ввода

Далее я настроила автоматические подписи коммитов git. С помощью следующих коммитов (рис. **??**):

1 **git config –global user.signingkey < PGP Fingerprint >**

2 **git config –global commit.gpgsign true**

3 **git config –global gpg.program $(which gpg2)**

Настройка автоматических подписей коммитов git

Настроила gh с помощью gh auth login (рис. **??**)

Настройка gh

Создала шаблон рабочего пространства (рис. **??**)(рис. **??**):

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 **cat ~/.ssh/id_rsa.pub | xclip -sel clip**

5 **git clone –recursive git@github.com:< owner >/study_2021-2022_os-intro.git os-intro**

Создание репозитория курса на основе шаблона 1

Создание репозитория курса на основе шаблона 2

И наконец, настроила каталог курса (рис. **??**) (рис. **??**) (рис. **??**):

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

– Удалила лишние файлы:

rm package.json

– Создайте необходимые каталоги:

1 **make COURSE=os-intro**

– Отправила файлы на сервер:

1 **git add .**

2 **git commit -am ‘feat(main): make course structure’**

3 **git push**

Настройка каталога курса { #fig:017 width=70% }

Ввод фразы-пароля

Продолжение настройки каталога курса

4 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.

   В свете усложнения сред разработки они помогают командам разработчиков работать быстрее и эффективнее. Системы контроля версий наиболее полезны командам DevOps, поскольку помогают сократить время разработки и увеличить количество успешных развертываний.

   Программное обеспечение контроля версий отслеживает все вносимые в код изменения в специальной базе данных. При обнаружении ошибки разработчики могут вернуться назад и выполнить сравнение с более ранними версиями кода для исправления ошибок, сводя к минимуму проблемы для всех участников команды.

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

   • Хранилище –репозиторий - место хранения всех версий и служебной информации.

   • Команда git commit делает для проекта снимок текущего состояния изменений, добавленных в раздел проиндексированных файлов

• 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 контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

   Децентрализованные VCS позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией.

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 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.

   Основные команды git: • Команда git add добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита.

   • Команда git status показывает состояния файлов в рабочем каталоге и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе.

   • Команда git diff используется для вычисления разницы между любыми двумя Git деревьями.

   • Команда git difftool просто запускает внешнюю утилиту сравнения для показа различий в двух деревьях, на случай если вы хотите использовать что-либо отличное от встроенного просмотрщика git diff.

   • Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

   • Команда git rm используется в Git для удаления файлов из индекса и рабочей копии.

   • Команда git mv — это всего лишь удобный способ переместить файл, а затем выполнить git add для нового файла и git rm для старого.

   • Команда git clean используется для удаления мусора из рабочего каталога.

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

«git push» означает мерж локальных изменений в удаленный репозиторий, а «git pull» — наоборот, мерж изменений из удаленного репозитория в локальный.

Команда git fetch связывается с удалённым репозиторием и забирает из него все изменения, которых у вас пока нет и сохраняет их локально.

Команда git remote служит для управления списком удалённых репозиториев.

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

   Ветвь разработки – это бифуркация состояния кода, что создает новый путь для его эволюции. Можно генерировать разные ветки Git, которые будут существовать параллельно к главной ветке. Таким образом, можно упорядоченно и точно включать новые функции в наш код.

   Использование Git Branches имеет несколько преимуществ. Из основных:

   - Можно разрабатывать новые функции нашего приложения, не мешая разработке в основной ветке.
   - С помощью веток Git можно создавать разные ветки разработки, которые могут сходиться в одном и том же хранилище, например, стабильную ветвь, тестовую ветвь и нестабильную ветвь.

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

    Файл .gitignore — это текстовый файл, помещенный в ваш репозиторий git, который сообщает git не отслеживать определенные файлы и папки, которые вы не хотите загружать в ваш главный репозиторий.

    Есть несколько задач, которые наиболее эффективно решаются с использованием файла .gitignore:

    - Не мучаться с выбором нужных файлов для индексации (которая git add).

- Сделать локальный конфиг, который не будет затронут pull-ом.
- Защитить чувствительную информацию от случайного раскрытия.
- Быстро очищать проект от временных файлов.

5 Выводы

Я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.