Лабораторная работа №2

Управление версиями

НКНбд-01-21

Смирнов-Мальцев Е.Д.

Преподаватель: Кулябов Д.С.

План

1) Цель работы
2) Ход работы
3) Контрольные вопросы
4) Вывод

Цель работы

– Изучить идеологию и применение средств контроля версий.

– Освоить умения по работе с git.

Ход работы

1. Создал учетную запись на Github и заполнил основные данные
2. Установил Git-flow и gh на виртуальной машине
3. Настроил git
4. Создал 2 ssh ключа: по алгоритму rsa размером 4096 бит и по алгоритму ed25519.
5. Сгенерировал pgp-ключ
6. Скопировал pgp-ключ и ввел его на сайте Github.
7. Скопировал ssh-ключ и ввел его на Github
8. Указал Git применять ранее введенный email в качестве подписи коммитов.
9. Авторизировался на Github.
10. Создал репозиторий на основе шаблона рабочего пространства.
11. Настроил репозиторий.

Ответы на контрольные вопросы

1) **VCS** – это системы контроля версий. Они используются при работе в команде. 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.

2) **Хранилище** – место, в котором хранятся все версии проекта. Для уменьшения объема используемой памяти можно хранить только изменения проекта. **Commit** – добавленные и измененные файлы по сравнению с предшествующей версией проекта. **История** – последовательность изменений проекта. **Рабочая копия** – копия над которой сейчас идет работа.

Ответы на контрольные вопросы

1) Централизованная система – система, в которой существует центральное хранилище, которое доступно всем участникам проекта. В децентрализованной системе у каждого участника есть свой репозиторий, что позволяет работать, не подключаясь к сети. Пример централизованной системы: CVS. Децентрализованной: Git.

2) При единоличной работе с VCS берешь нужную версию, вносишь правки и добавляешь коммит.

3) При совместной работе в централизованном хранилище, добавляется работа по устранению конфликтов. Также необходимо сливать версии, и вообще следить за структурированностью истории. Нельзя забывать проверять последнюю версию программы, потому что напарник мог ее изменить.

4) Git помогает работать одновременно над одним и тем же проектом независимо, а потом совмещать достижения разработчиков. Также он помогает хранить историю версий, что позволяет быстро откатиться в случае необходимости.

Ответы на контрольные вопросы

5) Основные команды Git:

1. Создание основного дерева репозитория: git init
2. Получение обновлений (изменений) текущего дерева из центрального репозитория: git pull
3. Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push
4. Просмотр списка изменённых файлов в текущей директории: git status
5. Просмотр текущих изменений: git diff
6. Добавить все изменённые и/или созданные файлы и/или каталоги: git add .
7. Добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена_файлов
8. Удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов
9. Сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита'
10. Сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit

Ответы на контрольные вопросы

5) Основные команды Git:

1. Создание новой ветки, базирующейся на текущей: checkout -b имя_ветки
2. 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)
3. Отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки
4. Слияние ветки с текущим деревом: git merge --no-ff имя_ветки
5. Удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки
6. Принудительное удаление локальной ветки: git branch -D имя_ветки
7. Удаление ветки с центрального репозитория: git push origin :имя_ветки

Ответы на контрольные вопросы

8) Пример использования локального репозитория: В Git создать репозиторий привязанный к имени пользователя и email и работать в нем не подключаясь к сети. Пример использования удаленного репозитория: создать репозиторий на Github, настроить авторизацию через ssh-ключ и pgp-ключ, авторизоваться и работать с ним через командную строку.

9) Ветви необходимы в случае, если есть несколько путей развития программы, поэтому из одной версии получаются сразу несколько.

10) Игнорировать файлы можно с помощью команды gitignore для того, чтобы не забивать хранилище мусором.

Вывод

1) Я научился создавать репозиторий на Github с помощью командной строки.
2) Я ознакомился с Git структурой и понял ее преимущества при работе в команде, а именно удобство отслеживания изменений.