

Отчёт по лабораторной работе №2

Управление версиями

Терёхин Александр

Содержание

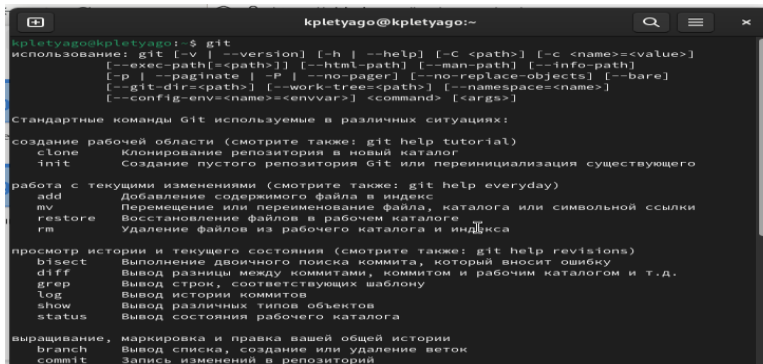
1	Цель работы	1
2	Выполнение лабораторной работы.....	1
3	Вывод.....	3
4	Контрольные вопросы.....	3

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

2 Выполнение лабораторной работы

Устанавливаем git, git-flow и gh.



```
kpletyago@kpletyago:~$ git
использование: 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>]

Стандартные команды Git используемые в различных ситуациях:

создание рабочей области (смотрите также: git help tutorial)
clone      Клонирование репозитория в новый каталог
init       Создание пустого репозитория git или переинициализация существующего

работа с текущими изменениями (смотрите также: git help everyday)
add        Добавление содержимого файла в индекс
mv         Перемещение или переименование файла, каталога или символической ссылки
restore    Восстановление файлов в рабочем каталоге
rm         Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: git help revisions)
bisect     Выполнение двоичного поиска коммита, который вносит ошибку
diff       Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
grep       Вывод строк, соответствующих шаблону
log        Вывод истории коммитов
show       Вывод различных типов объектов
status     Вывод состояния рабочего каталога

выращивание, маркировка и правка вашей общей истории
branch     Вывод списка, создание или удаление веток
commit     Запись изменений в репозиторий
```

Figure 1: Загрузка пакетов

Зададим имя и email владельца репозитория, кодировку и прочие параметры.

Создаем SSH ключи

```
kpletyago@kpletyago:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kpletyago/.ssh/id_rsa):
Created directory /home/kpletyago/.ssh.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kpletyago/.ssh/id_rsa
Your public key has been saved in /home/kpletyago/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:g3Gt20Af6IG0bavfQ5Sxkv0szTtp-pTGDbhJ3N/LEDU kpletyago@kpletyago
The key's randomart image is:
+----[RSA 4096]-----+
|  .o |
| ..+o |
| .B+.. E |
| .o X+.. . |
| ..oo 0 B+ |
| ..ooB + = o |
| O...+ + + |
| ... . o . |
| O.. o.. |
+----[SHA256]-----+
kpletyago@kpletyago:~$
```

Figure 3: rsa-4096

```
kpletyago@kpletyago:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kpletyago/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kpletyago/.ssh/id_ed25519
Your public key has been saved in /home/kpletyago/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:vhwkru4ndadLfaLrplmRYkCSTIgwBMZxt24t/l90 kpletyago@kpletyago
The key's randomart image is:
+----[ED25519 256]-----+
|X+.o |
|+..+ |
|=oo. |
|..+. |
|..+o. S |
|O... |
|..+..+ O . |
|O+...+o . E |
|O+...+.. |
|O+...+.. |
+----[SHA256]-----+
kpletyago@kpletyago:~$
```

Figure 4: ed25519

Создаем GPG ключ

Figure 5: GPG ключ

Добавляем GPG ключ в аккаунт

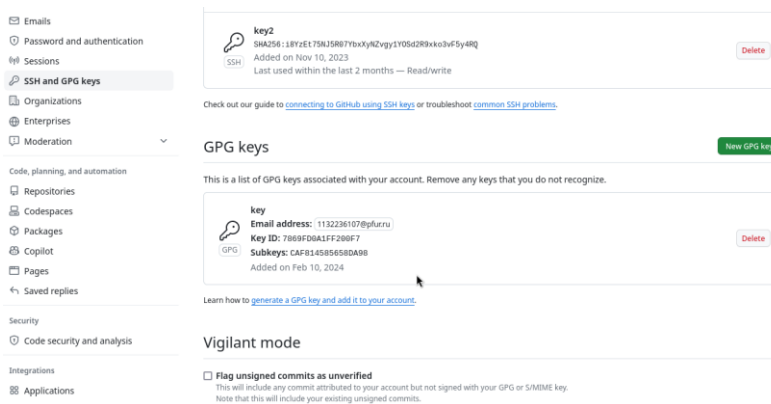


Figure 6: GPG ключ

Настройка автоматических подписей коммитов git

Figure 7: Параметры репозитория

Настройка gh

```
kpletyago@kpletyago:~$ 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? /home/kpletyago/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

i First copy your one-time code: 25C9-2EF1
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/kpletyago/.ssh/id_rsa.pub
✓ Logged in as kpletyago

kpletyago@kpletyago:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
kpletyago@kpletyago:~$ cd ~/work/study/2023-2024/"Операционные системы"
kpletyago@kpletyago:~/work/study/2023-2024/Операционные системы$ gh repo create os-intro
--template=yamadharma/course-directory-student-template --public
✓ Created repository kpletyago/os-intro on GitHub
kpletyago@kpletyago:~/work/study/2023-2024/Операционные системы$
```

Figure 8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
Определение изменений: 100% (52/52), готово.  
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-2024/Операционные системы/os-intro  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.json  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$ make COURSE=os-intro prepare  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$ ls  
CHANGELOG.md  tabs          prepare      README.en.md  template  
config         LICENSE       presentation README.git-flow.md  
COURSE        Makefile      project-personal  README.md  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$
```

Figure 9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py  
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py  
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattr1  
buses.py  
create mode 100644 project-personal/stage6/report/report.md  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$ git push  
Перечисление объектов: 38, готово.  
Подсчет объектов: 100% (38/38), готово.  
Пакетирование объектов: 100% (38/38), готово.  
Сжатие объектов: 100% (38/38), готово.  
Запись объектов: 100% (37/37), 342.06 КБ | 3.08 МБ/с, готово.  
Всего 37 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пак  
етов 0  
remote: Resolving deltas: 100% (4/4), completed with 1 local object.  
To github.com:kpletiyago/os-intro.git  
f0cf688..88e0543 master -> master  
kpletiyago@kpletiyago:~/work/study/2023-2024/Операционные системы/os-intro$
```

Figure 10: Первый коммит

3 Вывод

Мы приобрели практические навыки работы с сервисом github.

4 Контрольные вопросы

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

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. 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. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
 - хранилище - пространство на накопителе где расположен репозиторий
 - commit - сохранение состояния хранилища
 - история - список изменений хранилища (коммитов)
 - рабочая копия - локальная копия сетевого репозитория, в которой работает программист. Текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней)
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.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) 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. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

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

Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы.

5. Опишите порядок работы с общим хранилищем VCS.

Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя сливается с основной веткой проекта.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches).
- Работа с изменениями: выявление (diff), слияние (patch, merge).
- Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер.

7. Назовите и дайте краткую характеристику командам git.

- git config - установка параметров
- git status - полный список изменений файлов, ожидающих коммита
- git add . - сделать все измененные файлы готовыми для коммита.
- git commit -m “[descriptive message]” - записать изменения с заданным сообщением.
- git branch - список всех локальных веток в текущей директории.
- git checkout [branch-name] - переключиться на указанную ветку и обновить рабочую директорию.
- git merge [branch] — соединить изменения в текущей ветке с изменениями из заданной.
- git push - запустить текущую ветку в удаленную ветку.
- git pull - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
 - `git remote add [имя] [url]` — добавляет удалённый репозиторий с заданным именем;
 - `git remote remove [имя]` — удаляет удалённый репозиторий с заданным именем;
 - `git remote rename [старое имя] [новое имя]` — переименовывает удалённый репозиторий;
 - `git remote set-url [имя] [url]` — присваивает репозиторию с именем новый адрес;
 - `git remote show [имя]` — показывает информацию о репозитории.
9. Что такое и зачем могут быть нужны ветви (branches)?

Ветвление — это возможность работать над разными версиями проекта: вместо одного списка с упорядоченными коммитами история будет расходиться в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется master, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью.

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

Зачастую нам не нужно, чтобы Git отслеживал все файлы в репозитории, потому что в их число могут входить: