

Отчет по лабораторной работе 2

Первоначальна настройка git

Гисматуллин Артём Вадимович НПИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Системы контроля версий. Общие понятия	7
3.2	Примеры использования git	8
4	Выполнение лабораторной работы	9
5	Контрольные вопросы	16
6	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Командная строка. Базовая настройка git	9
4.2	Командная строка. Генерация ключа	10
4.3	Командная строка. Список ключей	11
4.4	Настройки GitHub. Добавление ключа	12
4.5	Командная строка. Настройка подписей	13
4.6	Настройки GitHub. Добавление SSH ключа	14
4.7	Командная строка. Настройка каталога курса	14
4.8	Командная строка. Завершение настройки	15

Список таблиц

1 Цель работы

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

2 Задание

1. Установить программное обеспечение (git, gh).
2. Провести базовую настройку git.
3. Создать SSH и pgr ключи, настроить github и добавить ключи в github.
4. Создание и настройка рабочего пространства на основе шаблона.

3 Теоретическое введение

3.1 Системы контроля версий. Общие понятия

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек

над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

3.2 Примеры использования git

- Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
- Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

4 Выполнение лабораторной работы

1. Сперва проведем базовую настройку git, задав имя и email владельца репозитория, настроим utf-8 и начнем настройку верификации и подписания коммитов git (рис. [4.1]):

```
avgismatullin@dk4n65 ~/work $ dnf install git
bash: dnf: команда не найдена
avgismatullin@dk4n65 ~/work $ git config --global user.name "Artem Gismatullin"
avgismatullin@dk4n65 ~/work $ git config --global user.email "DreamMagic147@yandex.ru"
avgismatullin@dk4n65 ~/work $ git config --global core.quotepath false
avgismatullin@dk4n65 ~/work $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
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 и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
(4) RSA (только для подписи)
(14) Имеющийся на карте ключ
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

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: Artem Gismatullin
```

Рис. 4.1: Командная строка. Базовая настройка git

После генерации ключа для верификации (рис. [4.2]) выводим список ключей и копируем отпечаток приватного ключа (рис. [4.3]):

```

Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Artem Gismatullin
Адрес электронной почты: DreamMagic147@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Artem Gismatullin <DreamMagic147@yandex.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/a/v/avgismatullin/.gnupg/openpgp-revocs
.d'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/a/v/avgismatullin/.gnupg/o
penpgp-revocs.d/0D560809EAB5A78A4C73D76A989D3815DF7EF3E0.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-02-16 [SC]
       0D560809EAB5A78A4C73D76A989D3815DF7EF3E0
uid           Artem Gismatullin <DreamMagic147@yandex.ru>
sub   rsa4096 2023-02-16 [E]

avgismatullin@dk4n65 ~/work $

```

Рис. 4.2: Командная строка. Генерация ключа

```
avgismatullin@dk4n65 ~/work $ 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/a/v/avgismatullin/.gnupg/pubring.kbx
-----
sec   rsa4096/989D3815DF7EF3E0 2023-02-16 [SC]
      0D560809EAB5A78A4C73D76A989D3815DF7EF3E0
uid           [ абсолютно ] Artem Gismatullin <DreamMagic147@yandex.ru>
ssb   rsa4096/841F35443D42E01C 2023-02-16 [E]

avgismatullin@dk4n65 ~/work $
```

Рис. 4.3: Командная строка. Список ключей

Добавляем полученный ключ в GitHub (рис. [4.4]):

GPG keys / Add new

Title

arch-pc

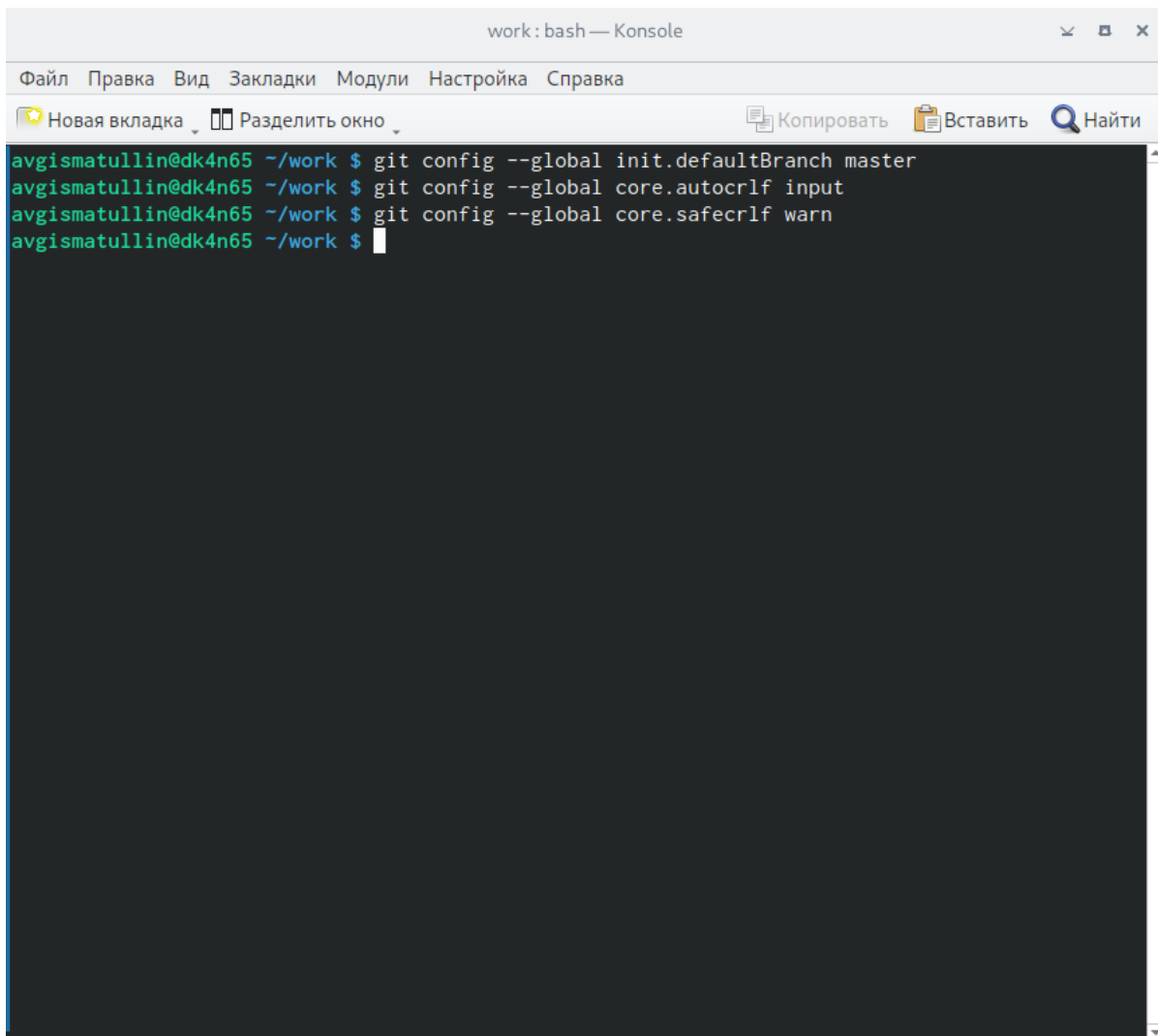
Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINB3CwW0tXpIaHfbiqYtHGB/K3304CK0QKAP91310TE3ZAXXWakT0
UW0na5uu+NI8JVgky8I9DHUHBXT+x81vjCvYlboErTW2KsVj/MuXeXanktLpVtK
QBWSFDhHzp0fE9idJbXxYEep81FL7Klc+hpIhrU6dPOiezly+Ejh7FC0N8rDYaVQ
Bllzt8tVcFIW8NckZ9Qu/qE8hbQ3oN47AwVk2QcOPRfPtK2Bqqc7qOoHcfghlgs
1O1dF46Y+p4llmdlYo/eM5vwHomEJku9poKP+USyZGAq6FHia10ag0qNbecevKFN
ztUorb7ef4ARXSrR8LJGF40As5DZv1qByyAigDWBYPMbQM65AYftfCQ18SdzjB20
xAfF8ftBvw==
=7mSA
-----END PGP PUBLIC KEY BLOCK-----
```

Add GPG key

Рис. 4.4: Настройки GitHub. Добавление ключа

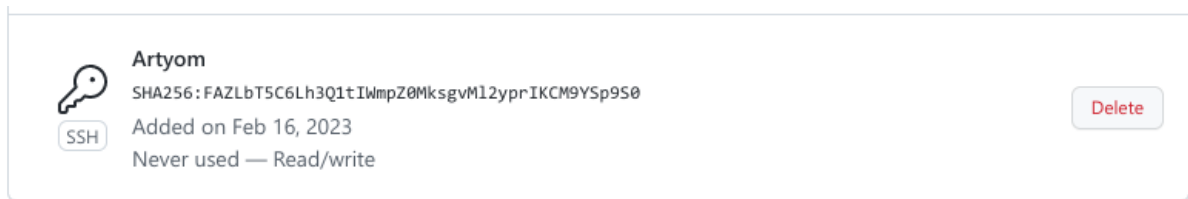
Перейдем к настройке автоматических подписей коммитов git. Укажем наш email для подписи (рис. [4.5]):



```
work: bash — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно  Копировать  Вставить  Найти
avgismatullin@dk4n65 ~/work $ git config --global init.defaultBranch master
avgismatullin@dk4n65 ~/work $ git config --global core.autocrlf input
avgismatullin@dk4n65 ~/work $ git config --global core.safecrlf warn
avgismatullin@dk4n65 ~/work $
```

Рис. 4.5: Командная строка. Настройка подписей

2. Для дальнейшей работы в дисплейном классе с GitHub необходимо создать SSH ключ. По алгоритму rsa (`ssh-keygen -t rsa -b 4096`) получаем его и, аналогично ключу pgr, добавляем на GitHub (рис. [4.6]):



Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Рис. 4.6: Настройки GitHub. Добавление SSH ключа

- Далее перейдем к созданию репозитория курса. Следуя иерархии, создадим соответствующие каталоги, настроим gh авторизацию и клонируем репозиторий на основе шаблона (рис. [4.7]):

```
avgismatullin@dk3n38 ~ $ mkdir -p ~/work/study/2022-2023/"Операционные системы"
avgismatullin@dk3n38 ~ $ cd ~/work/study/2022-2023/"Операционные системы"
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository ArtyomGismatullin/study_2022-2023_os-intro on GitHub
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:ArtyomGismatullin/course-directory-student-template.git
fatal: репозиторий «recursive» не существует
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:ArtyomGismatullin/course-directory-student-template.git
Клонирование в «course-directory-student-template»...
remote: Enumerating objects: 158, done.
remote: Counting objects: 100% (158/158), done.
remote: Compressing objects: 100% (118/118), done.
remote: Total 158 (delta 60), reused 136 (delta 38), pack-reused 0
Получение объектов: 100% (158/158), 51.65 КиБ | 534.00 КиБ/с, готово.
Определение изменений: 100% (60/60), 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/a/v/avgismatullin/work/study/2022-2023/Операционные системы/course-directory-student-template/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 1.02 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/a/v/avgismatullin/work/study/2022-2023/Операционные системы/course-directory-student-template/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 2.13 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы $
```

Рис. 4.7: Командная строка. Настройка каталога курса

Затем удалим лишние файлы, создадим необходимые каталоги и отправим файлы на сервер (рис. [4.8]).

```

avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы/os-intro $ rm package.json
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro > COURSE
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы/os-intro $ make
avgismatullin@dk3n38 ~/work/study/2022-2023/Операционные системы/os-intro $ git add .
git commit -am 'feat(main): make course structure'
git push
[master 2e5fb22] feat(main): make course structure
361 files changed, 100327 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab02/report/report.md

```

Рис. 4.8: Командная строка. Завершение настройки

5 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS 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.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. 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 VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?

Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о 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`. `git --version` (Проверка версии Git) `git init` (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) `git clone https://www.github.com/username/repo-name` (Скопировать существующий удаленный Git-репозиторий) `git remote` (Просмотреть список текущих удалённых репозиториях Git) `git remote -v` (Для более подробного вывода) `git add my_script.py` (Можете указать в команде конкретный файл). `git add .` (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) `git commit -am "Commit message"` (Вы можете сжать все индексированные файлы и отправить коммит). `git branch` (Просмотреть список текущих веток можно с помощью команды `branch`) `git --help` (Чтобы узнать больше обо всех доступных параметрах и командах) `git push origin master` (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветки (branches)? 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.

6 Выводы

В ходе работы я изучил идеологию и применение средств контроля версий, а также освоил умения по работе с git.

Список литературы