РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Операционные системы

Студент: Маметкадыров Ы. Т.

Группа: НПМбд-02-20

МОСКВА

2021 г.

Лабораторная работа № 2. Управление версиями

Цель работы. Изучить идеологию и применение средств контроля версий.

Ход работы. 1. Создали учётную запись на https://github.com. (рис. 1)

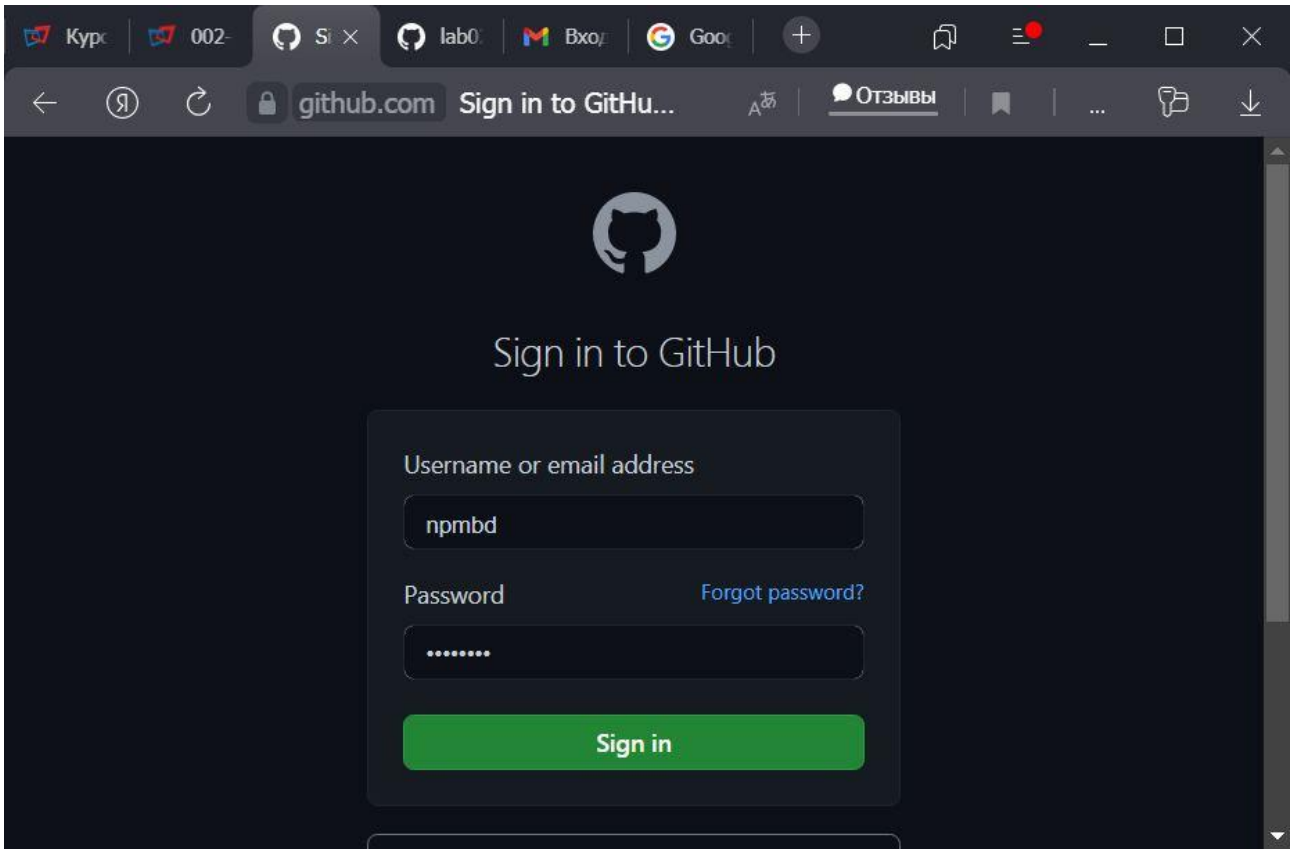


Рис. 1. Учетная запись GitHub

2. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

```
[itmametkadihrov@itmametkadihrov ~]$ git config --global user.name "npmbd"
[itmametkadihrov@itmametkadihrov ~]$ git config --global user.email "mametkadyrov.y@gmail.com"
[itmametkadihrov@itmametkadihrov ~]$ git config --global quotepath false
```

3. Создали репозиторий с именем os-intro. (рис. 2)

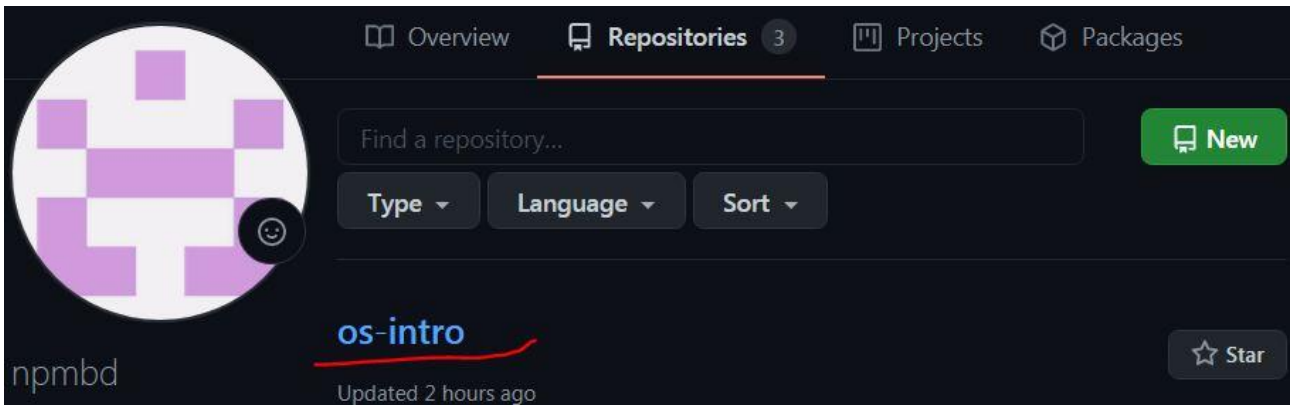


Рис. 2. Репозиторий os-intro

4. Инициализировали системы git командой get init.

5. Создали заготовку для файла README.md (рис. 3)

```
[itmametkadihrov@itmametkadihrov laboratory]$ echo "# Лабораторные работы" >> README.md[itmame
tkadihrov@itmametkadihrov laboratory]$ git add README.md
[itmametkadihrov@itmametkadihrov laboratory]$ git commit -m "first commit"
[master (root-commit) 0752a7c] first commit
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
[itmametkadihrov@itmametkadihrov laboratory]$ git remote add origin
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags                import all tags and associated objects when fetching
                          or do not fetch any tag at all (--no-tags)
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=<push|fetch>]
                          set up remote as a mirror to push to or fetch from

[itmametkadihrov@itmametkadihrov laboratory]$ git@github.com:<npmbd>/sciproc-intro.git
bash: npmbd: Нет такого файла или каталога
[itmametkadihrov@itmametkadihrov laboratory]$ git push -u origin master
```

Рис. 3. Создание заготовки

6. Добавили файл лицензии:

```
[itmametkadihrov@itmametkadihrov laboratory]$ wget https://creativecommons.org/licenses/by/4.0
/legalcode.txt -O LICENSE
--2021-05-01 14:57:18--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.150.16, 172.67.34.140, 104.20
.151.16, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.150.16|:443... соединение устан
овлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

    [ <=>                                       ] 18 657      --.-K/s   за 0,006s

2021-05-01 14:57:19 (2,95 MB/s) - «LICENSE» сохранён [18657]
```

Рис. 4. Лицензия

7. Добавили шаблон игнорируемых файлов. Просмотрели список имеющихся шаблонов (рис. 5)

```
[itmametkadihrov@itmametkadihrov laboratory]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratortitanium,appcode,appcode+all,appcode+iml
appengine,aptanastudio,arcanist,archive,archives
archlinuxpackages,aspnetcore,assembler,ate,atmelstudio
ats,audio,automationstudio,autotools,autotools+strict
awr,azurefunctions,backup,ballerina,basercms
basic,batch,bazaar,bazel,bitrise
bitrix,bittorrent,blackbox,bloop,bluej
bookdown,bower,bricxcc,buck,c
c++,cake,cakephp,cakephp2,cakephp3
calabash,carthage,certificates,ceylon,cfwheels
chefcookbook,chocolatey,clean,clion,clion+all
clion+iml,clojure,cloud9,cmake,cocoapods
cocos2dx,cocoscreator,code,code-java,codeblocks
codecomposerstudio,codeigniter,codeio,codekit,codesniffer
coffeescript,commonlisp,compodoc,composer,compressed
compressedarchive,compression,conan,concrete5,coq
cordova,craftcms,crashlytics,crbasic,crossbar
crystal,cs-cart,csharp,cuda,cvs
cypressio,d,dart,darteditor,data
database,datarecovery,dbeaver,defold,delphi
dframe,diff,direnv,diskimage,django
dm,docfx,docpress,docz,dotenv
dotfilessh,dotnetcore,dotsettings,dreamweaver,dropbox
drupal,drupal7,drupal8,e2studio,eagle
easybook,eclipse,eiffelstudio,elasticbeanstalk,elisp
elixir,elm,emacs,ember,ensime
episerver,erlang,espresso,executable,exercism
expressionengine,extjs,fancy,fastlane,finale
```

Рис. 5. Добавление шаблона

8. Затем скачали шаблон для C (рис. 6)

```
zsh,zukencr8000[itmametkadihrov@itmametkadihrocurl -L -s https://www.gitignore.io/api/c >> .gi
tignore
```

Рис. 6. Скачивание шаблона

9. Добавили новые файлы: git add .

10. Выполнили коммит: git commit -a

11. Отправили на github: git oush

12. Инициализировали git-flow: git flow init

13. Проверили что мы на ветке develop: git branch

14. Создали релиз с версией 1.0.0, записали версию, добавили индекс (рис. 7).

```
[itmametkadihrov@itmametkadihrov ~]$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '1.0.0'

[itmametkadihrov@itmametkadihrov ~]$ echo "1.0.0" >> VERSION
[itmametkadihrov@itmametkadihrov ~]$ git add .
[itmametkadihrov@itmametkadihrov ~]$ git commit -am 'chore(main): add version'
[release/1.0.0 5bb6f66] chore(main): add version
 1156 files changed, 7445 insertions(+)
 create mode 100644 .ICEauthority
 create mode 100644 .bash_history
 create mode 100644 .bash_logout
 create mode 100644 .bash_profile
 create mode 100644 .bashrc
 create mode 100644 .cache/abrt/applet_dirlist
 create mode 100644 .cache/abrt/lastnotification
 create mode 100644 .cache/event-sound-cache.tdb.e4a3d6250d32433b9e4d6a9fae0f70c5.x86_64-redha
t-linux-gnu
 create mode 100644 .cache/fontconfig/477ff6b974c3c1b81af411ebecb34280-le64.cache-7
 create mode 100644 .cache/fontconfig/CACHEDIR.TAG
 create mode 100644 .cache/gdm/session.log
 create mode 100644 .cache/gdm/session.log.old
 create mode 100644 .cache/gnome-shell/update-check-3.28
 create mode 100644 .cache/gstreamer-1.0/registry.x86_64.bin
```

Рис. 7. Создание релиза

15. Залили релизную ветку в основную ветку командой: git flow release finish 1.0.0.

16. Отправили данные на github (рис. 8)

```
[itmametkadihrov@itmametkadihrov ~]$ git push --all
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>

[itmametkadihrov@itmametkadihrov ~]$ git push --tags
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>
```

Рис. 8. Отправка на github

Вывод. Изучили идеологию и применение средств контроля версий.

Контрольные вопросы

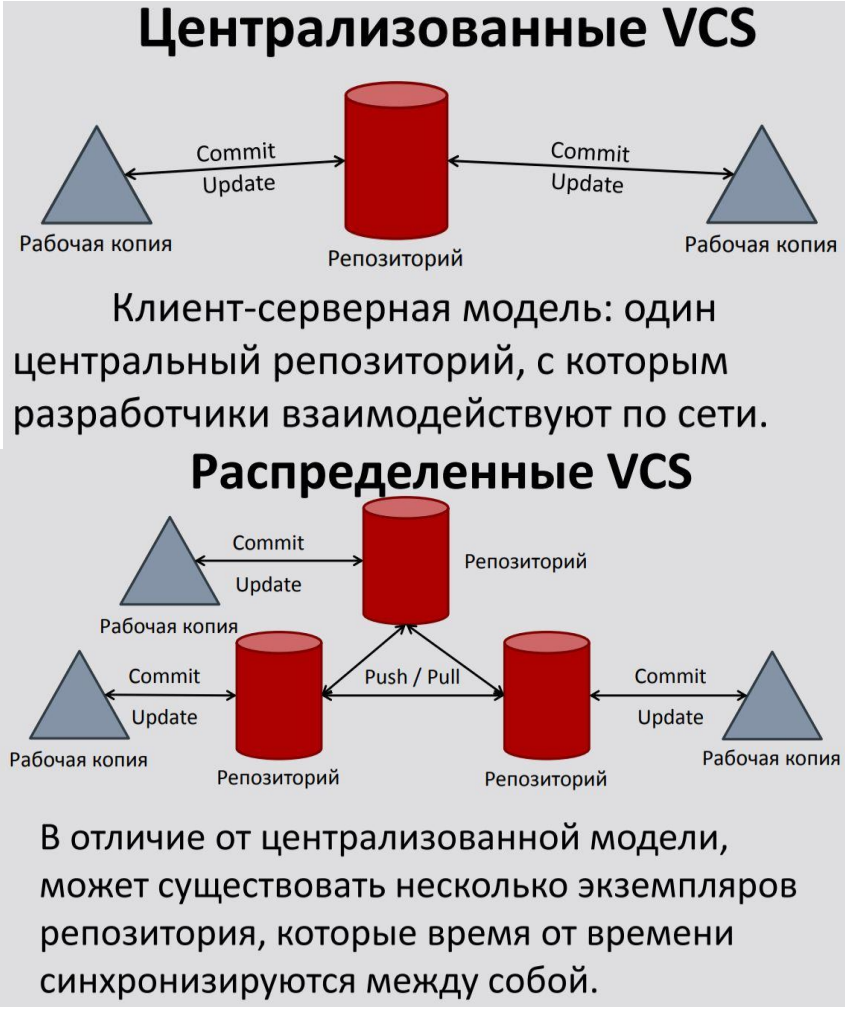
1. Системы контроля версий - это программы которые позволяют отслеживать изменения вашего файла, и хранить их. Системы контроля версий применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ 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. Хранилище — это содержимое скрытой папки .git. В этой папке хранятся все версии рабочей области и служебная информация.

   Коммит – сохранение, фиксация изменений в программном коде.

   Рабочая копия — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.
3. Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет бóльшую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, так называемая «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. копия - копия проекта, связанная с репозиторием.

   Децентрализованная (DVCS) система означает, что у каждого разработчика есть личный репозиторий проекта с полным набором всех версий. А все необходимые для работы файлы находятся на компьютере. При этом постоянное подключение к сети не требуется, поэтому система работает

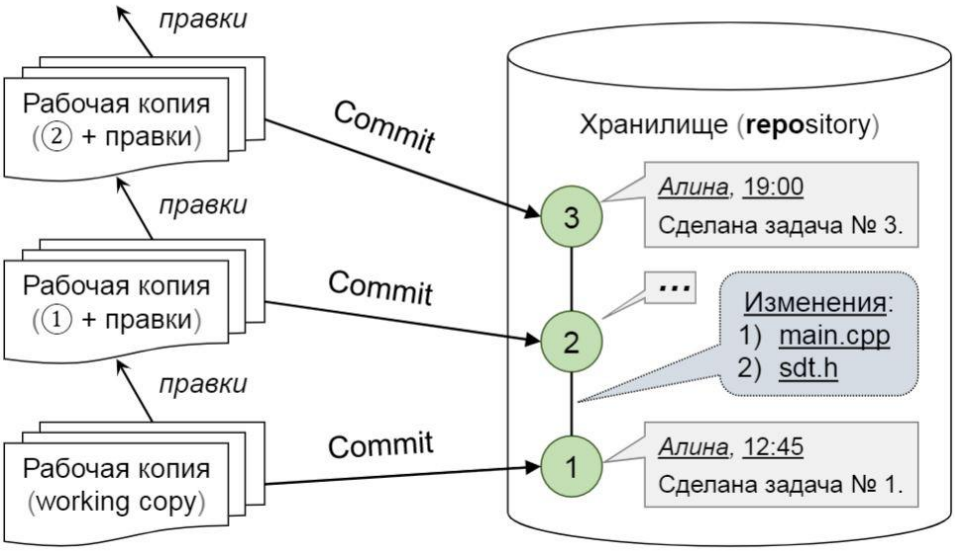
быстро. При командной разработке нужна синхронизация репозиториев, так как проект — один и его состояние должно быть у всех одинаковым.



Две наиболее известные DVCS – это Git и Mercurial.

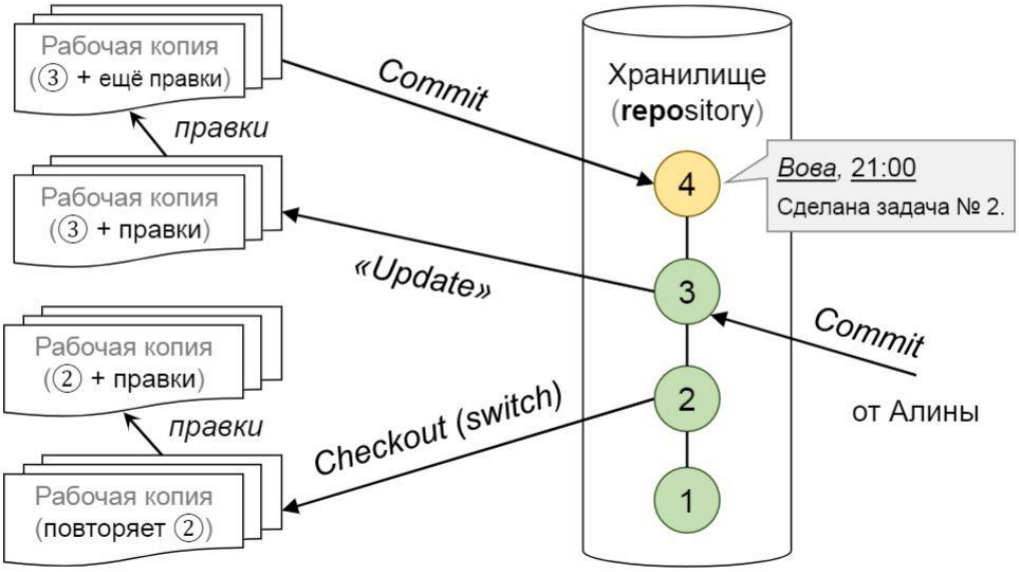
В качестве примеров централизованных VCS можно привести CVS, Subversion.

4. Первым действием, которое должен выполнить разработчик, является извлечение рабочей копии проекта или той его части, с которой предстоит работать. Это действие выполняется с помощью команды извлечения версии (обычно checkout или clone). Разработчик задаёт версию, которая должна быть скопирована, по умолчанию обычно копируется последняя (или выбранная администратором в качестве основной) версия.



5. По команде извлечения устанавливается соединение с сервером, и проект (или его часть — один из каталогов с подкаталогами) в виде дерева каталогов и файлов копируется на компьютер разработчика. Обычной практикой является дублирование рабочей копии: помимо основного каталога с проектом на локальный диск (либо в отдельный, специально выбранный каталог, либо в системные подкаталоги основного дерева проекта) дополнительно записывается ещё одна его копия. Работая с проектом, разработчик изменяет только файлы основной рабочей копии. Вторая локальная копия хранится в качестве эталона, позволяя в любой момент без обращения к серверу определить, какие изменения внесены в конкретный файл или проект в целом и от какой версии была «отпочкована» рабочая копия; как правило, любая попытка ручного изменения этой копии приводит к ошибкам в работе программного обеспечения VCS.

Работа с общим хранилищем



6. Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки.

   Удалённый доступ к репозиториям Git обеспечивается git-демоном, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

7. Наиболее часто используемые команды git:

   – создание основного дерева репозитория:

   git init

   – получение обновлений (изменений) текущего дерева из центрального репозитория:

   git pull

   – отправка всех произведённых изменений локального дерева в центральный репозиторий:

   git push

   – просмотр списка изменённых файлов в текущей директории:

   git status

– просмотр текущих изменения:

git diff

– сохранение текущих изменений:

– добавить все изменённые и/или созданные файлы и/или каталоги:

git add .

– добавить конкретные изменённые и/или созданные файлы и/или каталоги:

git add имена_файлов

– удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог

остаётся в локальной директории):

git rm имена_файлов

– сохранение добавленных изменений:

– сохранить все добавленные изменения и все изменённые файлы:

git commit -am 'Описание коммита'

– сохранить добавленные изменения с внесением комментария через встроенный редактор:

git commit

– создание новой ветки, базирующейся на текущей:

git checkout -b имя_ветки

– 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)

– отправка изменений конкретной ветки в центральный репозиторий:

git push origin имя_ветки

– слияние ветки с текущим деревом:

git merge --no-ff имя_ветки

– удаление ветки:

– удаление локальной уже слитой с основным деревом ветки:

git branch -d имя_ветки

– принудительное удаление локальной ветки:

git branch -D имя_ветки

– удаление ветки с центрального репозитория:

git push origin :имя_ветки

8. Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование:

```
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
$ git remote
origin
```

Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит добавить их в индекс и осуществить первый коммит изменений. Добиться этого вы сможете запустив команду git add несколько раз, указав индексируемые файлы, а затем выполнив git commit:

```
$ git add *.c
$ git add LICENSE
$ git commit -m 'Initial project version'
```

9. Ветвь - направление разработки, независимое от других. Ветвь представляет собой копию части хранилища, в которую можно вносить изменения, не влияющие на другие ветви. Документы в разных ветвях имеют одинаковую историю до точки ветвления и разные - после неё. Системы управления версиями предоставляют инструменты для манипуляции ветвями, прежде всего создание ветви и слияние изменений с другой ветвью.
10. .gitignore - это простой текстовый файл, в каждой строке которого содержится шаблон файла или каталога, который необходимо проигнорировать.

    .gitignore использует glob шаблоны для сопоставления имен файлов с символами подстановки. Если у вас есть файлы или каталоги, содержащие шаблон подстановки (например *), вы можете использовать один обратный слеш (*), чтобы экранировать такой символ.