

# Отчет по лабораторной работе №3

## Дисциплина: операционные системы

Королев Федор Константинович

### Содержание

Цель работы.....	1
Ход работы.....	1
Вывод.....	8
Контрольные вопросы.....	8

### Цель работы

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

### Ход работы

1. Создадим аккаунт на GitHub'е (Рис. 1):

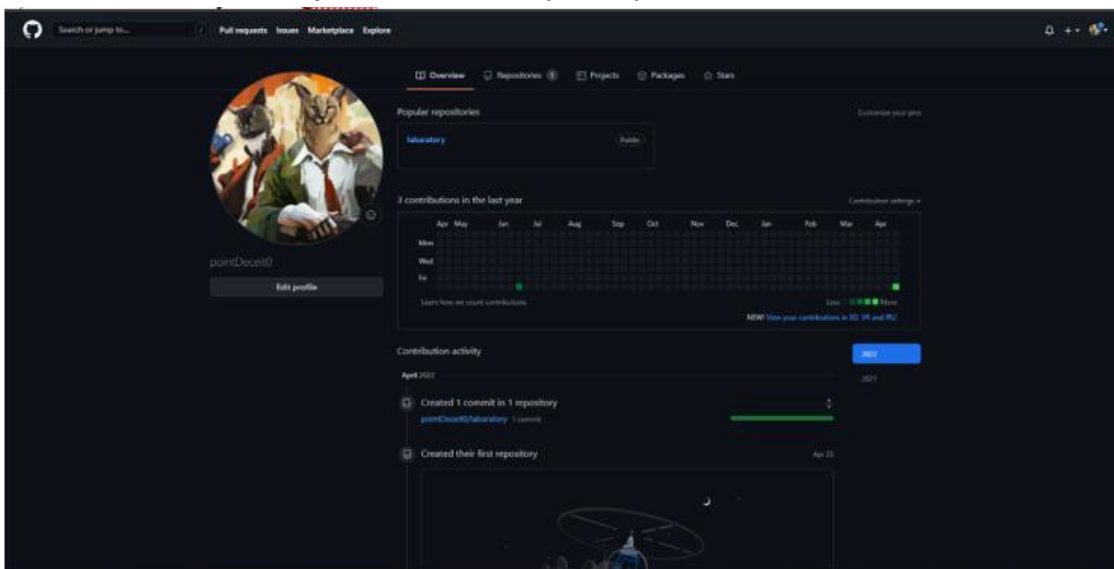


Рис. 1 создание учетной записи на GitHub

2. Настроим систему git. Синхронизируем учетную запись GitHub с компьютером, с помощью команд (Рис. 2):

```
git config --global user.name "user name"
```

```
git config --global user.email "user email"
```

```
fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$ git config --global user.name "pointDeceit0"

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$ git config --global user.name pointDeceit0

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$ git config --global user.email "fedor.korolev.87@mail.ru"

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$ git config --global user.email fedor.korolev.87@mail.ru
```

Рис. 2 настройка системы git, привязка к аккаунту

3. Сгенерируем ключ с помощью команды (Рис. 3):  
`ssh-keygen -C "Name Surname <work@mail>"`

```
fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ ssh-keygen -C "Fedor Korolev <fedor.korolev.87@mail.ru>"
generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\fkkor/.ssh/id_rsa): .ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh.
Your public key has been saved in .ssh.pub.
The key fingerprint is:
SHA256:yNRITmk2qncpiHzaZ/zqGgVeM4fUmbEEPcpiz08l8us Fedor Korolev <fedor.korolev.87@mail.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|          +*o+       |
|        .oo@.        |
|       ..=Xoo        |
|      .oo0++ .       |
|     ..=. S          |
|    .. o.o +         |
|   ..00+ = .         |
|  +..* o             |
| . o=o+E            |
+---[SHA256]-----+
```

Рис. 3 генерация ssh ключа

4. В GitHub'е создадим репозиторий с именем laboratory (Рис. 4):

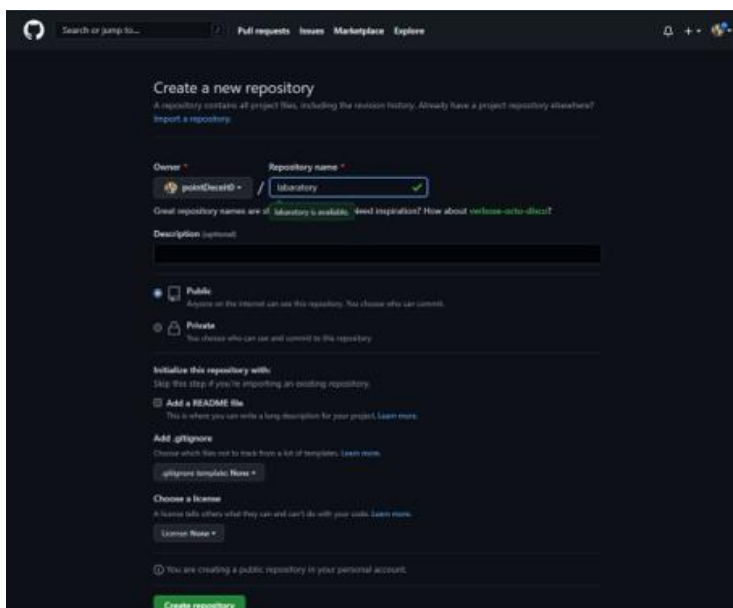


Рис. 4 создание репозитория с именем *laboratory*

5. Скачиваем пустой репозиторий, и в папке со скачанным репозиторием создаем 2 файла: README.md & LICENSE.md (Рис. 5):

```
fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2
$ git clone https://github.com/pointDeceit0/laboratory
Cloning into 'laboratory'...
warning: You appear to have cloned an empty repository.

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/laboratory
$ touch README.md

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/laboratory
$ touch LICENSE.md
```

Рис. 5 создание файлов *README.md* & *LICENSE.md*

6. Далее добавим их с помощью команды `git add -A`

примем изменения с помощью команды

```
git commit -m "Add -A"
```

и загрузим в репозиторий с помощью команды (Рис. 6)

```
git push
```

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ git add -A

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ git commit -m "Add -A"
[main (root-commit) 218d481] Add -A
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 LICENSE.md
 create mode 100644 README.md

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 224 bytes | 74.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/pointDeceit0/labaratory

```

Рис. 6 добавление файлов в репозиторий

7. Создадим директории lab01 & lab02 для первой и второй лабораторной с помощью команд mkdir, предварительно создав в обеих папках файлы .gitkeep, для того, чтобы github видел эти файлы. Далее добавим их, закомитим и загрузим(push)(Рис. 7):

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ mkdir lab01

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ mkdir lab02

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory
$ touch lab02/.gitkeep

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operation_systems/lab_2/labaratory/lab01
$ touch .gitkeep

```

Рис. 7 добавление папок lab01 & lab02 в репозиторий

8. Получим список имеющихся шаблонов с помощью команды (Рис. 8)  
`curl -L -s https://www.gitignore.io/api/list`

```

fkkor@dkz25 /cydrive/c/users/fkkor/desktop/study/operati
systems/lab_2/laboratory
$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,a1
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcleratotitanium,appcode,appco
de+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,azurite,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,cheffcookbook,chocolatey,circuitpython
clean,clion,clion+all,clion+iml,clojure
cloud9,cmake,cocopods,cocos2dx,cocoscreator
codeblocks,codecomposerstudio,codeigniter,codeio,codekit
codesniffer,coffeescript,commonlisp,compodoc,composer
compressed,compressedarchive,compression,conan,concrete5
coq,cordova,craftcms,crashlytics,crbasic
crossbar,crystal,cs-cart,cssharp,cuda
cvs,cypressio,d,dart,darteditor
data,database,datarecovery,dbeaver,defold
delphi,dframe,diff,direnv,diskimage
django,dm,docfx,dopress,docz
dotenv,dotfilessh,dotnetcore,dotsettings,dreamweaver
dropbox,drupal,drupal7,drupal8,e2studio
eagle,easybook,eclipse,eiffelstudio,elasticbeanstalk
elisp,elixir,elm,emacs,ember
enzyme,episerver,erlang,espresso,executable
exercism,expressionengine,extjs,fancy,fastlane
finale,firebase,flashbuilder,flask,flatpak
flex,flexbuilder,floobits,flutter,font
fontforge,forcedotcom,forgegradle,fortran,freecad
freepascal,fsharp,fuelphp,fusetools,games
gcov,genero4gl,geth,ggts,gis
git,gitbook,go,godot,goland
goodsync,gpg,gradle,grails,greenfoot
groovy,grunt,gwt,haskell,helm
hexo,hoi,homeassistant,homebrew,hsp
hugo,hyperledgercomposer,iar,iar_ewarm,iarembdedworkbenc
h
idapro,idris,igorpro,images,infer
inforcms,inforcrm,intellij,intellij+all,intellij+iml
ionic3,jabref,java,jboss,jboss-4-2-3-ga
jboss-6-x,jboss4,jboss6,jdeveloper,jekyll
jenv,jetbrains,jetbrains+all,jetbrains+iml,jgiven
jigsaw,jmeter,joe,joomla,jspm
julia,jupyternotebooks,justcode,kaldi,kate
kdevelop4,kdiff3,keil,kentico,keystonejs
kicad,kirby2,kobalt,kohana,komodoedit
konyvisualizer,kotlin,labview,labviewnxcg,lamp
laravel,latex,lazarus,leiningen,lemonstand
less,liberosoc,librarian-chef,libreoffice,lighthouseci
lilypond,linux,lithium,localstack,logtalk
lsspsice,ltspsice,lua,lyx,macos
magento,magento1,magento2,magic-xpa,matlab
maven,mavensmate,mdbook,mean,mercurial
mercury,meson,metals,metaprogrammingsystem,meteor
meteorjs,microsoftoffice,mikroc,moban,modelsim
modx,momentics,monodevelop,mplabx,mule
nanoc,nativescript,ncrunch,nesc,netbeans
nette,nextjs,nikola,nim,ninja
node,nodechakratimetraveldebug,nohup,notepadpp,nova
now,nuxtjs,nwjs,objective-c,obsidian
ocaml,octave,octobercms,opa,opencart
opencv,openfoam,openframeworks,openframeworks+visualstudio
,oracleforms
orcad,osx,otto,oxideshop,oxygenxmleditor
packer,particle,patch,pawn,perl
perl6,ph7cms,phalcon,phoenix,phpcodesniffer
phpstorm,phpstorm+all,phpstorm+iml,phpunit,pico8
pimcore,pimcore4,pimcore5,pinegrow,platformio
playframework,plone,polymer,powershell,premake-gmake
prepros,prestashop,processing,progressabl,psoccreator
puluml,puluml+stacks,puppet,puppet-librarian,purebasic
purescript,putty,pvs,pycharm,pycharm+all
pycharm+iml,pydev,python,pythonvanilla,qml
qooxdoo,qt,qtcreator,r,racket
rails,react,reactnative,reasonml,red
redcar,redis,renpy,retool,rhodesrhmobile
rider,robotframework,root,ros,ros2
ruby,rubymine,rubymine+all,rubymine+iml,rust
salesforce,salesforcedx,sam,sam+config,sas
sass,sbt,scala,scheme,scons
scrivener,sdcc,seamgen,senchatouch,serverless
shopware,silverstripe,sketchup,slickedit,smalltalk
snap,snapcraft,solidity,soliditytruffle,sonar
sonarqube,sourcepawn,spark,splunk,spreadsheet
ssh,standardml,stata,stdlib,stellar
stellar,storybookjs,strapl,stylus,sublimetext
sugarcrm,svn,swift,swiftpackagemanager,swiftpm
symfony,symphonycms,synology,synopsysvcs,tags
tarmainstallmate,terraform,terragrunt,test,testcomplete
testinfra,tex,text,txmate,txtpattern
theos-tweak,thinkphp,tla+,tortoisegit,tower
turbogears2,twincat3,tye,typings,typo3
typo3-composer,umbraco,unity,unrealengine,vaadin
vagrant,valgrrind,vapor,vcpkg,venv
vertx,video,vim,virtualenv,virtuoso
visualstudio,visualstudiocode,vivado,vlab,vs
vue,vuejs,vvvv,waf,wakanda
web,webmethods,webstorm,webstorm+all,webstorm+iml
werckercli,windows,wintersmith,wordpress,wyam
xamarinstudio,xcode,xcodeinjection,xilinx,xilinxise
xilinxvivado,xll,xojo,xtext,y86
yalc,yarn,yeoman,yii,yii2
zendframework,zephir,zig,zsh,zukencr8000

```

9. Скачаем шаблон для C++, с помощью команды  
`curl -L -s https://www.gitignore.io/api/list/c++ >> .gitignore`  
Добавим .gitignore в репозиторий (Рис. 9)



```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git add .gitignore

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git commit -m "first commit"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

```

Рис. 9 добавим .gitignore в репозиторий

Обновим всё и загрузим изменения (Рис. 10):

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git add -u

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git commit -m "first commit"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 951 bytes | 475.00 KiB/s, don
e.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/pointDeceit0/labaratory
4e711cc..875c9ac main -> main

```

Рис. 10 обновление и загрузка в репозиторий

10. Инициализируем git-flow с помощью команды  
`git flow init`

Подтвердим нужные настройки (Рис. 11):

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git flow init

which branch should be used for bringing forth production
releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/] v
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/users/fkkor/desktop/study
/operation_systems/lab_2/labaratory/.git/hooks]

```

Рис. 11 инициализация git-flow

Проверяем на какой ветке находимся (Рис. 12):

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git branch
* develop
  main

```

Рис. 12 проверка ветки

Создадим функциональную ветку (Рис. 13):

```

fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git flow feature start feature_branch
Switched to a new branch 'vfeature_branch'

Summary of actions:
- A new branch 'vfeature_branch' was created, based on 'de
velop'
- You are now on branch 'vfeature_branch'

Now, start committing on your feature. When done, use:

git flow feature finish feature_branch

```

Рис. 13 создание функциональной ветки

По завершению работы над функцией объединим ветку feature\_branch с develop (Рис. 14):

```
fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ git flow feature finish feature_branch
switched to branch 'develop'
Already up to date.
Deleted branch vfeature_branch (was 875c9ac).

Summary of actions:
- The feature branch 'vfeature_branch' was merged into 'de
velop'
- Feature branch 'vfeature_branch' has been locally delete
d
- You are now on branch 'develop'
```

Рис. 14 объединение ветки feature\_branch с develop

Создадим новую ветку release, используя команду (Рис. 15)

`git flow release start 1.0.0`

```
fkkor@dk2n25 /cygdrive/c/users/fkkor/desktop/study/operati
on_systems/lab_2/labaratory
$ 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 'deve
lop'
- 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 rel
ease
- When done, run:

    git flow release finish '1.0.0'
```

Рис. 15 создание новой ветки release

## Вывод

В ходе данной лабораторной работы я изучил идеологию и научил применять средства контроля версий.

## Контрольные вопросы

- 1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
- 2). 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

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.

3). Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4). Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

```
git config --global user.name "Имя Фамилия"
```

```
git config --global user.email "[work@mail](mailto:work@mail)"
```

и настроив utf-8 в выводе сообщений git:

```
git config --global core.quotePath false
```

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

```
cd
```

```
mkdir tutorial
```

```
cd tutorial
```

```
git init
```

5). Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия <work@mail>"
```

Ключи сохраняются в каталоге ~/.ssh/.

Скопировав из локальной консоли ключ в буфер обмена

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

вставляем ключ в появившееся на сайте поле.

6). 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). Основные команды git:

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 изменений: `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 стекущим 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). Использование git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

```
git add hello.txt
```

```
git commit -am 'Новый файл'
```

9). Проблемы, которые решают ветки git:

- нужно постоянно создавать архивы с рабочим кодом
- сложно “переключаться” между архивами

- сложно перетаскивать изменения между архивами
- легко что-то напутать или потерять

10). Во время работы над проектом так или иначе могут создаваться файлы, которые не требуются добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list`

Затем скачать шаблон, например, для C и C++

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

```
curl -L -s https://www.gitignore.io/api/c++ >> .gitignore
```