Git - шпаргалка

Удалённые серверы

Клонирование с git-сервера

- через настроенный доступ **ssh**:

```
# GitHub
git clone git@github.com:Jurik-Phys/bchs.git
# Any other Git server
git clone user_name@server_name:/opt/git/project
```

- через публичный **https** доступ:

```
git clone https://github.com/Jurik-Phys/bchs.git
```

- через публичный **https** доступ с клонированием подпроектов:

```
git clone --recurse-submodules https://github.com/Jurik-Phys/bchs.git
```

- проверка доступа по ssh к git-серверу, на примере GitHub:

```
ssh -T git@github.com
```

  - успех:

```
Hi Jurik-Phys! You've successfully authenticated, ...
```

  - ошибка:

```
git@github.com: Permission denied (publickey).
```

Создание удалённого репозитория

Инициализация git-репозитория

```
git init
```

Добавление файлов проекта в репозиторий

```
git add .
```

Создание первого коммита

```
git commit -m 'Initial project version'
```

Переход на уровень выше и клонирование локального git'а

```
cd ..
git clone --bare project/ project.git
```

Копирование bare-реозитория «project.git» на сервер [1)]

```
scp -r project.git user_name@server:/opt/git/
```

Просмотр удалённых репозиториев

```
git remote -v
```

Инспекция удалённого репозитория

```
git remote show remote_server
```

Переименование удалённого репозитория

```
git remote rename old_name new_name
```

Удаление удалённого репозитория

```
git remote rm remote_server
```

Смена "https" на "git over ssh"

```
git remote set-url origin git@github.com:Jurik-Phys/libvirt-snapshot-manager.git
```

Файлы

Проверка состояния файлов

```
git status
```

Добавление файлов в индекс

```
git add filename
```

Удалние файла

```
git rm filename # удаление файла из проекта и индекса
```

```
git rm --cached filename #удаление файла из индекса
```

Перемещение файла

```
git mv filename.old filename.new
```

Данная команда равносильна

```
   mv filename.old filename.new
   git rm filename.old
   git add filenam.new
```

Коммиты

Просмотр различий (diff)

```
git diff # просмотр не проиндексированных изменений
git diff --staged # проиндексированные изменения (войдут в следующий commit)
```

Создание коммита

```
git commit -a  # Ручной ввод комментария
```

Редактирование последнего коммита

```
# замещает последний коммит текущим состоянием (хеш другой)
git commit --amend
# если последний коммит был отправлен на сервер, то
# необходимо принудительно обновить ветку
git push -f server local_branch:remote_branch
#  если участников несколько, то возможны коллизии!
# (не рекомендуется практиковать этот способ
```

Откат изменений

```
# удаление всех коммитов в ветке до commit_sha
#  commit_sha - останется последним коммитов в ветке
# изменений в рабчем каталоге не произойдёт.
git reset --soft commit_sha
# удаление и коммитов до commit_sha, и
# откат состояния рабочего каталога до commit_sha
git reset --hard commit_sha
# удалить последний коммит в ветке
git reset --hard HEAD^
```

Восстановить файл из коммита

```
# восстановить файл main.cpp из коммита 965ee26
git checkout 965ee26 -- main.cpp
```

Отменить коммит

```
# сделать "обратный" коммит
git revert commit_sha
```

Изменение истории коммитов

```
# см. подсказки в текст. редакторе
git rebase -i commit_sha~1
```

Пример объединения нескольких коммитов в один по ссылке.

Просмотр истории коммитов

Две удобные конструкции, идущие из коробки:

```
git log --graph --all
git log --pretty=oneline --graph --all
```

Решение из интернета. Создание алиасов в config файле git'a:

```
[alias]
lg1 = log --graph --abbrev-commit --decorate --date=relative --
format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset)
%C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all
lg2 = log --graph --abbrev-commit --decorate --format=format:'%C(bold
blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold
yellow)%d%C(reset)%n''          %C(white)%s%C(reset) %C(dim white)- %an%C(reset)' -
-all
lg = !"git lg1"
```

Not currently on any branch

```
git stash
git checkout some-branch
git stash pop
```

Именование коммитов

Вариант №1 рекомендаций по именованию коммитов (ссылка):

```
<type> <description>
```

Fix	коммит, который устраняет баг
Feat	коммит, который содержит новый функционал
Chore	коммит, который не устраняет баг и не вносит новый функционал, а модифицирует или обновляет зависимости
Refactor	этот коммит содержит рефакторинг, который включает рефакторинг кода или изменения
Docs	этот коммит изменяет документацию, readme.md или markdown файлы
Style	коммит, который вносит изменения в оформление кода
Test	коммит, который изменяет тестовый файл, включая изменения
Perf	коммит, в котором улучшается производительность
Ci	коммит, в котором содержатся изменения в интеграции CI в файлах или скриптах
Build	это файлы, которые включают файлы сборки и зависимости
Revert	коммит, который сигнализирует об откате к предыдущему коммиту

Вариант №2 (ссылка)

1. Отделяйте заголовок от тела пустой строкой.
2. Ограничивайте заголовок 50 символами.
3. Пишите заголовок с заглавной буквы.
4. Не ставьте точку в конце заголовка.
5. Используйте повелительное наклонение в заголовке
    - Это не повелительное наклонение, а опущенное подлежащее во множественном числе. ~~This changes~~ fix accidental crush — ~~Эти изменения~~ чинят внезапные падения.
      If applied, this commit will **Refactor subsystem X for readability**
      If applied, this commit will **Update getting started documentation**
      If applied, this commit will **Remove deprecated methods**
      If applied, this commit will **Release version 1.0.0**
6. Переходите на следующую строку в теле на 72 символах.
7. В теле отвечайте на вопросы «что» и «почему», а не «как»

Ветки

Создание локальной ветки

- ```
  git branch new_branch    # создание новой ветки от последнего коммита
  git checkout new_branch   # переключение на новую ветку
  ```

Слияние веток

- ```
  # переход на ветку в которую будут сливаться изменения
  git checkout main
  # вливание devel_branch в текущую (master) ветку
  git merge devel_branch
  ```

Список всех веток

- `git branch -a`

Удаление ветки

- Локальный репозиторий:

```
git branch -d local_branch
```

- Удалённый репозиторий:

```
git push server :remote_branch
```

Переименование ветки

- Локальный репозиторий:
  - Текущая ветка (-m сокращение от –move):

```
git branch -m <newname>
```

  - Произвольная ветка:

```
git branch -m <oldname> <newname>
```

- Удалённый репозиторий:
  - Отправка новой ветки на сервер:

```
git push origin -u <newname>
```

  - Удаление старой ветки на сервере:

```
git push origin --delete <oldname>
```

Вытянуть удалённую ветку

```
# вытянуть удалённую ветку в локальную local_branch
git checkout -b local_branch server/remote_branch
# сокращённая команда:
git checkout --track server/local_branch
```

Сброс текущей локальной ветки до состояния удалённой

```
git fetch --all
git reset --hard server_name/branch_name
```

Отправка текущей ветки на git-сервер

```
git push remote_server local_branch_name:remote_branch_name
```

Рабочие деревья (Worktree)

- Добавить рабочую ветку

  ```
  git worktree add ../folder name-of-branch
  ```

- Посмотреть список рабочих деревьев

  ```
  git worktree list
  ```

- Очистить информацию о несуществующих (ранее удалённых) каталогах

  ```
  git worktree prune
  ```

Работа с подпроектом (ПП)

Добавление ПП в проект

Добавление удалённого сервера подпроекта

```
git remote add second_server user_name@sever_name:/opt/git/sub_project.git
```

Получение информации по новому подпроекту

```
git fetch second_server
```

Помещение подпроекта в отдельную ветку second_project

```
git checkout -b second_project second_server/master
# master - соответствующее имя удалённой ветки подпроекта
```

Переключение на ветку базового проекта

```
# проверка текущей ветки
git branch
git checkout devel
```

Вытягивание содержимого подпроекта в подкаталог

```
git read-tree --prefix=second/ -u second_project
```

Получение последней версии ПП

Переключение на ветку подпроекта и её обновление

```
git checkout second_project
git pull
```

Переключение на разрабатываемую ветку базового проекта

```
git checkout devel
```

Слияние актуальной кодовой базы подпроекта с текущей версией базового проекта

```
git merge --squash -s subtree --no-commit second_project
```

Разрешение возможных конфликтов

```
git mergetool
```

Оценка внесённых изменений

```
git status
```

Фиксация изменений базового проекта

```
git commit -a
```

Отправка локальных изменений на git-сервер ПП

Переключение на ветку подпроекта

```
git checkout second_project
```

Оценка различий соответствующих файлов

```
git diff-tree -p devel
```

Объединение кодовой базы основного проекта с файлами продпроекта

```
git merge --squash -s subtree --no-commit devel
# devel - имя разрабатываемой ветки базового проекта
```

Разрешение конфликтов, если требуется

```
git mergetool --squash -s subtree --no-commit devel
```

Фиксация изменений подпроекта

```
git commit -a
```

Отправка изменений подпроекта на git-сервер

```
git push second_server second_project:master
# master - соответствующее имя удалённой ветки подпроекта
```

Восстановление ПП после клонирования

Отделение каталога в отдельную подветку

```
git subtree split --prefix=second --branch second_project
```

Добавление удалённого репозитория подпроекта

```
git remote add second_server user@server:/opt/git/sub_project.git
```

Получение информации по удалённому серверу

```
git fetch second_server
```

Привязка соответствующих веток: локальной и удалённой

```
git branch --set-upstream second_project second_server/master
```

Метки

Создание меток

- Аннотированные метки (рекомендуется)

  ```
  git tag -a vX.Y -m "Project version X.Y"
  ```

- Легковесные метки

  ```
  git tag vX.Y
  ```

Просмотр меток

- Общий список:

  ```
  git tag
  ```

- Подробности по тегу «tag_name»:

  ```
  git show --no-patch tag_name
  ```

Отправка меток на сервер

- По умолчанию, команда git push не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер

```
git push origin --tags
```

Разрешение конфликтов

Настройка инструментов

```
# установка vimdiff в качестве инструмента разрешения конфликтов
git config --global merge.tool vimdiff
# не создавать *.orig файлов
git config --globasl mergetool.keepBackup false
```

Появление конфликта слияния

```
# установка текущей ветки (master)
git checkout master
# вливание ветки develop в master: конфликт объединения
git merge develop
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
git config --globasl mergetool.keepBackup false
```

Можно посмотреть детали конфликта:

```
git status
index.html: needs merge
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   unmerged:   index.html
#
```

Запуск процесса разрешения конфликтов

```
git mergetool
```

Описание vimdiff

По-умолчанию в debian'е интерфейс vimdiff представлен тремя буферами:

LOCAL (1)	MERGED (2)	REMOTE (3)

- LOCAL показывает состояние файла в текущей/[local] ветке, в которую вливают код.
- REMOTE показывает состояние файла во вливаемой[remote] ветке.
- MERGED - результат объединения, который будет сохранён в репозитории.
- BASE - временный файл, отображающий общую базу для объединения, позволяет оценить взаимное отличие local и remote файлов. В моей конфигурации его нет, а на нет и суда нет.

Подсказка Выяснить, какую функцию несёт, конкретный буфер можно по имени файла, в строке статуса vim'a. Общий формат подписи следующий: filename.LOCAL|REMOTE|BASE.id, где

- filename - имя файла, в котором возник конфликт объединения,
- LOCAL|REMOTE|BASE - роль данного буфера,
- id - служебный цифровой идентификатор. MERGED буфер отличается тем, что в строке статуса имеет просто: filename.

Основные команды

- **:diffget LO** - получить в текущий буфер (MERGED) версию из локальной(LOCAL) версии файла;
- **:diffget RE** - получить в текущий буфер (MERGED) версию из вливаемой(REMOTE) версии файла;
- **:ls** - посмотреть список буферов;
- **Ctrl + w** - перемещение между буферами;
- **]c** - переключиться на следующий конфликтный блок;
- **[c** - переключиться на предыдущий конфликтный блок;

Завершение процесса разрешения конфликта

После приведения буфера MERGED в удовлетворяющий требованиям вид, сохраняем его, выходим из vimdiff'a. Дальнейшие действия (создание коммита), согласно выводу команды:

```
git status
```

Настройка git на VDS

Исходные данные

Сервер (VDS) с будущим хранилищем git-репозиториев с каталогом хранения репозиториев

```
/mnt/srv.misc/git.repos
```

Данный каталог экспортируется через nfs-сервер. Запись в файле /etc/exports:

```
# Git repository for Git.Server
/mnt/srv.misc/git.repos/ git(rw,no_root_squash,no_subtree_check,async)
```

На сервере в виртуальной машине (KVM) при загрузке происходит монтирование экспортируемого каталога, согласно записи в fstab:

```
maxwell:/mnt/srv.misc/git.repos/  /var/lib/gitolite3/repositories nfs
rw,async,hard  0 0
```

Установка gitolite

Поскольку в качестве системы выбран Debian, то будет правильным устанавливать всё из пакетов.

```
apt-get install gitolite3
```

На данном этапе необходимо в виртуальную машину скопировать публичный ключ того пользователя, который будет администрировать git-репозиторий.

Дополнительные настройки пакета можно покрутить через:

```
dpkg-reconfigure gitolite3
```

Порт ssh в виртуальной машине изменён с 22 на 1010 в /etc/ssh/sshd_config для того, чтобы на VDS можно было настроить проброс 1010-ого порта виртуальную машину (/etc/firehol/firehol.conf):

```
 # *) GIT SERVER: WAN ==>> Git Server  (Port Forward)
      EXT_IF=eth0
      EXT_IP=46.160.39.181 # External IP

      INT_NAME=git
      INT_IP=`host $INT_NAME | awk '{ print $4 }'` # Internal IP

      EXT_PORT=1010
      INT_PORT=1010
         # 1010 [tcp]
         iptables -t nat -A PREROUTING -p tcp -d $EXT_IP --dport $EXT_PORT -j DNAT
--to-destination $INT_IP:$INT_PORT
         iptables -A FORWARD -i $EXT_IF -d $INT_IP -p tcp --dport $INT_PORT -j
ACCEPT
         # 1010 [udp]
         iptables -t nat -A PREROUTING -p udp -d $EXT_IP --dport $EXT_PORT -j DNAT
--to-destination $INT_IP:$INT_PORT
         iptables -A FORWARD -i $EXT_IF -d $INT_IP -p udp --dport $INT_PORT -j
ACCEPT
```

Администрирование gitolite

Администрирование производится в локальном репозитории gitolite-admin с последующим push'ем настроек на сервер, которые сразу после этого начинают действовать.

```
git clone ssh://gitolite3@git.jurik-phys.net:1010/gitolite-admin
```

Добавление пользователя состоит в том, чтобы скопировать публичный ключ username.pub

пользователя username в каталог keydir и внести его в группу пользователей или произвести другие манипуляции в файле ./conf/gitolite.conf.

Генерация публичного ключа командой **ssh-keygen -t rsa** от пользователя username.

Создание нового проекта заключается в добавлении записи в gitolite.conf:

```
repo superproject
    RW+     =     username
```

Протестировать работоспособность репозитория можно выполнив команду:

```
ssh -p 1010 gitolite3@git.jurik-phys.net
```

В результате при правильной настройке можно увидеть список доступных репозиториев.

Проблема. Похоже, что при создании нового репозитория через gitolite.conf из gitolite-admin, затирается файл projects.list. В итоге, cgit, читающий список репозториев с данного файла не находит ни одного репозитория. В качестве временного решения, можно добавить репозитории в projects.list вручную.

Причина проблемы. Gitolite v.3 по-умолчанию не добавляет проекты в project.list. Для того, чтобы они там появились необходимо установить соответствующие настройки репозитория в gitolite-admin/conf/gitoliteconf:

- Установить права на чтение для зарезервированного пользователя gitweb

  ```
  repo    foo bar baz
      R   =   gitweb
  ```

- Прописать хотя бы одну настройку для репозитория:

  ```
  config gitweb.owner         =   owner name
  config gitweb.description   =   some description
  config gitweb.category      =   some category
  ```

  В этом случае, необходимо отредактировать на сервере файл .gitolite.rc:

  ```
  GIT_CONFIG_KEYS => 'gitweb\.(owner|description|category)',
  ```

  В настройких cgit /etc/cgitrc до опции scan-path прописать

  ```
  enable-git-config=1
  ```

  В случае ошибки в web-интерфейсе смотреть логи apach'a. Скорее всего проблемы с правами доступа к файлу config в соответствующем репозитории.

Источники раз, два

Создание нового проекта

Клонирование репозитория настроек gitolite

```
git clone ssh://gitolite3@git.jurik-phys.net:1010/gitolite-admin
```

Редактирование файла gitolite.conf

```
vim gitolite-admin/conf/gitolite.conf
```

Добавление проекта в виде:

```
 repo project_groups/project_name
    RW+                         =   username
    config gitweb.owner         =   Project Owner Name
    config gitweb.description   =   Description of project
    config gitweb.category      =   Description of project category
```

Перед созданием коммита с новыми изменениями рекомендуется посмотреть лог предыдущих изменений с целью единообразного описания

```
git log
```

Фиксация изменений репозитория gitolite-admin

```
git add .
git commit
```

Отправка изменений на сервер

```
git push
```

Проверка обновлённого списка проектов

```
ssh -p 1010 gitolite3@git.jurik-phys.net
```

При правильной настройке cgit новый проект должен появиться в списке доступных в web-интерфейсе

Начало работы с пустым репозиторием

```
git clone ssh://gitolite3@git.jurik-phys.net:1010/project_groups/project_name
```

Настройка cgit

Установка пакета cgit

```
apt-get install cgit
```

Установка прав доступа к репозиторию

На основе данного материала.

- Добавление пользователя www-data в группу gitolite3:

```
usermod --append --groups gitolite3 www-data
```

- Изменение разрешений для будущих репозиториев /var/lib/gitolite3/.gitolite3.rc:

  ```
  UMASK => 0027
  ```

  Подробнее про umask по ссылке.

- Изменение прав доступа существующих каталогов:

  ```
  chmod g+rX /var/lib/gitolite3
  chmod -R g+rX /var/lib/gitolite3/repositories
  ```

- Если на данном этапе доступа к git-репозиторию нет

  ```
  su -s /bin/sh www-data
  $ ls /var/lib/gitolite3/repositories/
  ls: невозможно открыть каталог /var/lib/gitolite3/repositories/: ...
  ```

  то помочь исправить ситуацию должен ход конём:

  - Установка для /var/lib/gitolite3/repositories/ группы www-data:

    ```
    chown -R gitolite3:www-data /var/lib/gitolite3/repositories/
    ```

  - Установка SGID (Set Group Identifier) для каталога /var/lib/gitolite3/repositories/:

    ```
    chmod g+s /var/lib/gitolite3/repositories/
    ```

Включение cgi в apache

По-умолчанию, модуль cgi выключен. Для включение необходимо выполнить:

```
a2dismod mpm_event
a2enmod mpm_prefork
service apache2 restart
```

```
a2enmod cgi

Enabling module cgi.
To activate the new configuration, you need to run:
service apache2 restart
```

Важно! После включения модуля cgi необходимо перезапустить браузер. Без этого произошедшие изменения в браузере не изменяются.

Настройка VirtualHost

Удаление настроек пакета cgit

В настройках по-умолчанию git-репозиторий распологается по адресу www.mydomen.com/cgit, что не подходит для случая git.mydomen.com, поэтому эти настройки необходимо отключить:

```
rm /etc/apache2/conf-enabled/cgit.conf
```

Добавление сайта git.jurik-phys.net:80

Файл /etc/apache2/sites-available/git.jurik-phys.net.conf:

```
<VirtualHost *:80>
        ServerAdmin  admin@jurik-phys.net
        ServerAlias  git.jurik-phys.net
        DocumentRoot "/usr/lib/cgit/"
        Redirect permanent / https://git.jurik-phys.net

        <Directory "/usr/lib/cgit/">
                AllowOverride None
                Options +ExecCGI
                Require all granted
        </Directory>

        Alias       /cgit-css/  "/usr/share/cgit/"
        ScriptAlias /           "/usr/lib/cgit/cgit.cgi/"

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel error

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Основная задача - перенаправление запросов на https версию cgit.

Добавление сайта git.jurik-phys.net:443

Файл /etc/apache2/sites-available/git.jurik-phys.net-ssl.conf:

```
<VirtualHost *:443>
        ServerAdmin admin@jurik-phys.net
        ServerAlias git.jurik-phys.net
        DocumentRoot "/usr/lib/cgit/"
        SSLEngine on
        SSLCertificateFile    /etc/ssl/crt/jurik-phys.net.crt
        SSLCertificateKeyFile /etc/ssl/key/jurik-phys.net.key
        SSLCACertificateFile  /etc/ssl/ca-certs.pem
        <Directory "/usr/lib/cgit/">
                AllowOverride None
                Options +ExecCGI
```

```
                Require all granted
        </Directory>
        Alias       /cgit-css/  "/usr/share/cgit/"
        ScriptAlias /           "/usr/lib/cgit/cgit.cgi/"

        ErrorLog ${APACHE_LOG_DIR}/error-ssl.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel error

        CustomLog ${APACHE_LOG_DIR}/access-ssl.log combined
</VirtualHost>
```

Включение модуля SSL в apache'e для предотвращения ошибки «Invalid command 'SSLEngine'»:
Включение сайтов:

```
a2ensite git.jurik-phys.net-ssl
a2ensite git.jurik-phys.net
```

```
a2enmod ssl
```

Перегрузка конфигурации apache2:

```
service apache2 reload
```

Всё должно работать.

Установка пароля на git.jurik-phys.net

В каталог /usr/lib/cgit/ положить файл .htaccess следующего содержания

```
AuthName "Some message for users"
AuthType Basic
Require valid-user
AuthUserFile "/opt/git.jurik-phys.net/.htpasswd"
```

.htpasswd создаётся с помощью утилиты htpasswd из комплекта Apache

```
htpasswd -bc .htpasswd username password
```

P.S. также в настройках сайта апача необходимо AllowOverride None заменить на AllowOverride All

Разное

Настройка git

Идентификация пользователя

```
git config --global user.name "You name"
git config --global user.email "You e-mail"
```

Кириллица в имени файла

Если файлы с русскими буквами отображаются в виде

```
#       "\362\345\361\362"
```

То исправить ситуацию можно, установив параметр quotepath секции [core] конфигурационного файла git ~/.gitconfig в false.

```
[core]
    quotepath = false
```

Автодополнение

Положить файл автодополнений .git-completion.bash в домашний каталог и добавить в .bashrc source ~/.git-completion.bash. Или положить этот же файл (если он там уже не лежит) в /etc/bash_completion.d/ - должно автоматически подхватиться для всех пользователей.

Subtree

Данный модуль может быть не установлен. Тогда, либо установка из репозитория, либо согласно инструкции.

Он может быть установлен, но не «активирован» на выполнение

```
chmod +x /usr/share/doc/git/contrib/subtree/git-subtree.sh
ln -s /usr/share/doc/git/contrib/subtree/git-subtree.sh /usr/lib/git-core/git-
subtree
```

Временное сокрытие изменений

```
git stash # теперь можно сменить ветку
            # и продолжить работу в ней
# применить скрытые изменения и удалить "заначку"
git stash pop
```

Сборка мусора в базе

```
git gc
```

```
git count-objects -v
```

Частые сценарии

- Вариант №1
  - Создание ветки

    ```
    git branch BR-NAME
    ```

  - Верификация созданной ветки

    ```
    git branch
    ...
    BR-NAME
    ...
    ```

  - Переключение на созданную ветку «BR-NAME»

    ```
    git checkout BR-NAME
    ```

- Вариант №2
  - Создание и переключение на ветку «BR-NAME» одной командой

    ```
    git checkout -b BR-NAME
    ```

- Создание и переключение на ветку «BR-NAME»

  ```
  git checkout -b BR-NAME
  ```

- Внесение изменений в файлы проекта и создание коммита

  ```
  git commit -a
  ```

- Отправка ветки на сервер и установка связи между локальной и удалённой веткой

  ```
  git push -u origin BR-NAME
  ```

- Добавление MicroTeX как подпроекта в каталог `lab_os/verify/ext/MicroTeX`

  ```
  git submodule add https://github.com/NanoMichael/MicroTeX.git
  lab_os/verify/ext/MicroTeX
  ```

- При клонировании репозитория с подпроектами их необходимо инициализировать

  ```
  git submodule update --init --recursive
  ```

- Переключение на требуемую (openmath) ветку в подпроекте

  ```
  cd lab_os/verify/ext/MicroTeX
  ```

  ```
  git checkout openmath
  ```

- Вернуться в основной репозиторий и зафиксировать изменения

```
cd ../../../..
```

```
git commit -a
```

```
git push
```

- Деинициализация submodule

```
git submodule deinit lab_os/verify/ext/MicroTeX
```

- Удаление при необходимости файлов подпроекта из индекса Git

```
git rm --cached lab_os/verify/ext/MicroTeX
```

- Удаление каталога подпроекта

```
rm -r lab_os/verify/ext/MicroTeX
```

- Открыть файл .gitsubmodules и удалить блок с данными подмодуля

```
[submodule "lab_os/verify/ext/MicroTeX"]
    path = lab_os/verify/ext/MicroTeX
    url = https://github.com/NanoMichael/MicroTeX.git
```

- Список веток:

```
git branch
```

- Переключение текущей ветки:

```
git checkout BRANCH_NAME
```

- Может понадобиться, для включения отделённого коммита.

```
git cherry-pick commit_sha~1
```

- Когда необходимо вернуться к предыдущей версии кода:

```
git checkout commit_sha~1
```

- При необходимости сохранения изменений можно:
  1. Создать ветку:

     ```
     git branch new_branch
     ```

  2. Переключиться на созданную ветку:

     ```
     git checkout branch new_branch
     ```

- Когда стало ясно, что сделанные изменения в коде ошибочны

```
git reset --hard HEAD
```

Работа с GitHub

Первый запуск

Генерация ключей SSH

Для работы с сервисом GitHub через терминал по протоколу SSH необходимо один раз сгенерировать ключи шифрования и публичный добавить в аккаунт в раздел «Settings→SSH and GPG keys».

Процесс генерации ключей описан по ссылке.

Проверка соединения с github через ssh

```
ssh -T git@github.com
```

В первый раз необходимо одобрить подключение в ответ на запрос вида

```
The authenticity of host 'github.com (IP ADDRESS)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCBCLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)?
```

Коммит от имени аккаунта GitHub

GitHub использует email адрес для ассоциации отправленных коммитов с аккаунтом GitHub.

Чтобы связать коммиты с аккаунтом GitHub:

- убедитесь, что какой-нибудь email привязан к аккаунту GitHub, в настройках аккаунта;
- установите тот же самый email в настройках Git или репозитория:

```
git config --global user.email "ваш-github-email@example.com"
```

- создайте какие-нибудь коммиты (git commit -a) и отправьте их (git push) в GitHub.

После этого GitHub признает Вас, и все новые коммиты будут подписаны вашим аккаунтом со ссылкой на него, а не просто именем. Подробности по ссылке.

Типичные задачи

Создание пустого репозитория:

- создать новый репозиторий на странице «Create a new repository»
- клонировать созданный репозиторий на локальной машине:

```
git clone git@github.com/USERNAME/REPOSITORY.git
```

В текущем каталоге появится подкаталог REPOSITORY - это и есть каталог нового проекта.

Создание GitHub репозитория под существующий локальный проект:

- создать новый репозиторий на странице «Create a new repository»
- войти в каталог с проектом и подключить удалённый репозиторий командой

```
git remote add origin git@github.com:USERNAME/REPOSITORY.git
```

- просмотреть список удалённых репозиториев:

```
git remote
```

- а) отправить все ветки на удалённый репозиторий:

```
git push origin --all
```

- б) отправить только локальную ветку main:

```
git push -u origin main
```

Переименовать или удалить репозиторий можно на странице:

- `https://github.com/USERNAME/REPOSITORY/settings`

Добавление уже созданного на github'е репозитория

- `git remote add origin git@github.com:USERNAME/REPOSITORY.git`

Клонирование (стороннего) репозитория

- `git clone git@github.com:USERNAME/REPOSITORY.git`

Дополнительные материалы git

1. Статья в википедии
2. git-scm.com
3. Волшебство Git

git
1)

На серевере должен быть настроен ssh-доступ по ключу для user_name, разрешена запись в /opt/git

From:
https://jurik-phys.net/ - **Jurik-Phys.Net**

Permanent link:
https://jurik-phys.net/itechnology:git

Last update: **2025/09/24 03:20**