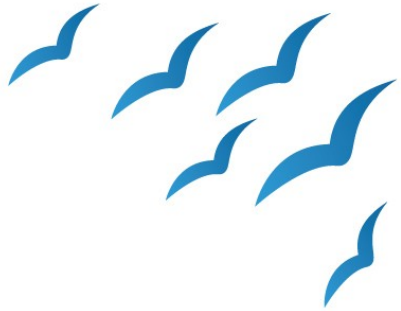
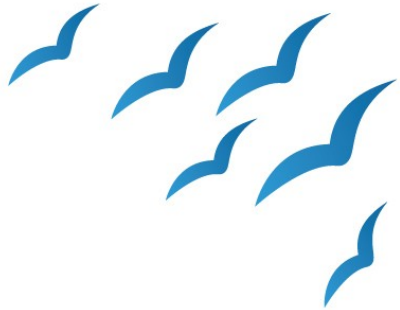
Система контроля версий



История создания

Проект был создан Линусом Торвальдсом (Linus Benedict Torvalds) для управления разработкой ядра Linux.

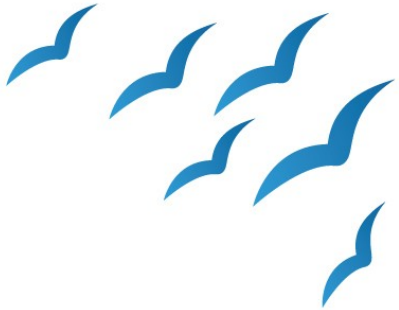
Первая версия выпущена 7 апреля 2005 года меньше чем за неделю разработки.



Цитата

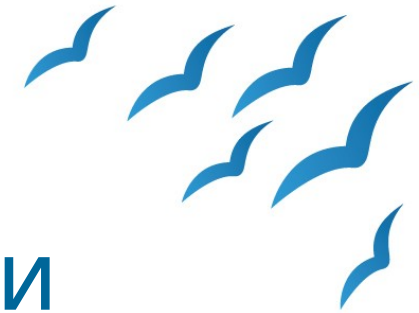
“ I'm an egotistical bastard, so I name all my projects after myself. First Linux, now git. “

git – (на английском сленге означает мерзавец)



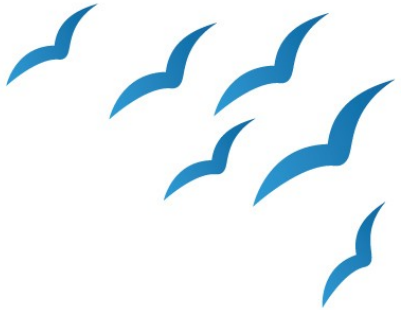
Git – хостинг

- GitHub
- GitLab
- Bitbucket
- SourceForge
- Codebase



Git – система контроля версии

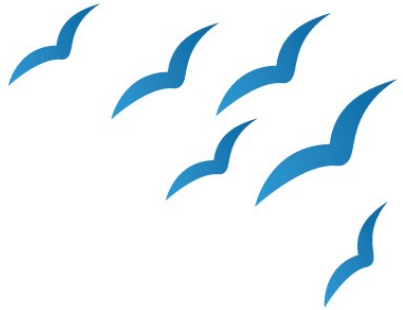
- Летописец
- Машина времени
- Резервная копия
- Мастер параллельных миров
- Народное вече



Летописец

Git – ведет всю историю разработки начиная от сотворения проекта. Делает подшивку черновиков в ветку по контрольным точкам (commit)

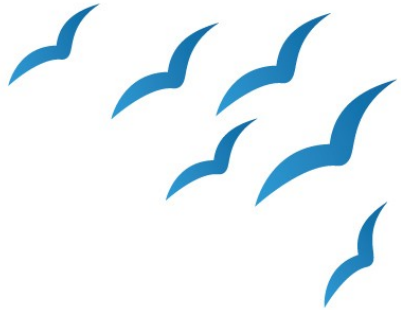
```
> git log
```



Машина времени

Предоставляет возможность путешествовать в прошлое по дереву летописных сводов.

```
>git checkout hash
```



Резервная копия

Создание резервных копий на удаленных серверах git репозитория.

```
>git push origin master
```

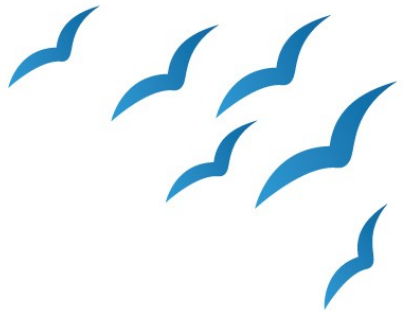
Мастер параллельных миров

В Git существует возможность распареллелить ветку разработки на несколько , посредством создания дополнительных веток.

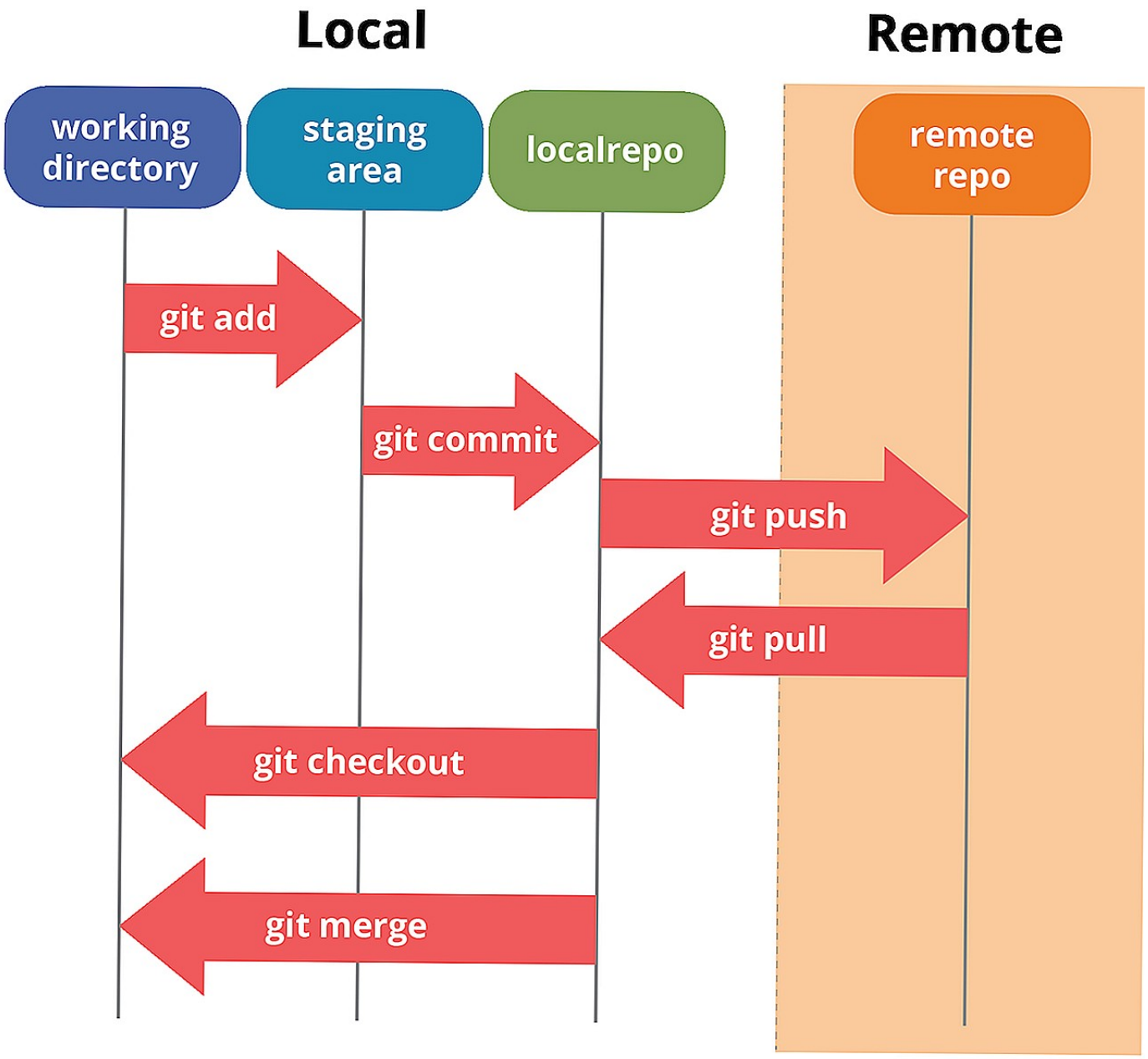
```
>git branch new_branch
```

Народное вече

Git воплощает в себе механизмы коллективного общения разработчиков. Принятия и обсуждения тех или иных технических решений. Идея pull request

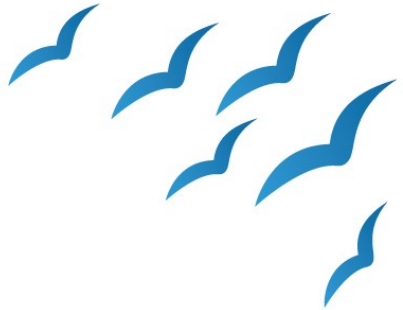


Управление репозиторием Git сводится к набору команд.
Рассмотрим диаграмму.


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

Настройка

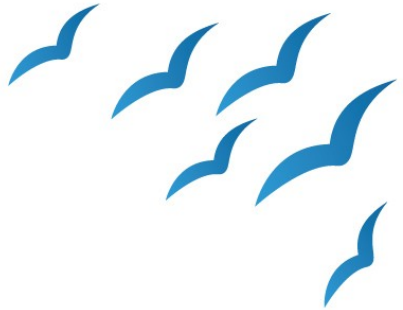
```
#Устанавливаем в командной строке


git config --global user.name "<ваше_имя>"

git config --global user.email "<адрес_почты@email.com>"


# Просмотр настроек

>git config --list
```



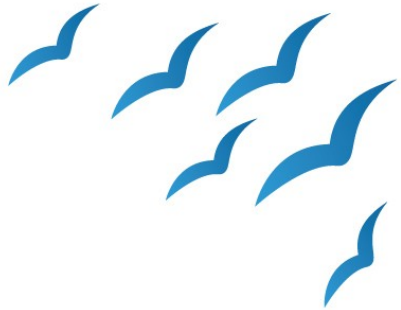
Превратить каталог, который не находится под
версионным контролем, в репозиторий Git

```
>git init
```

После того как была создана папка .git
добавляем в текущей каталог файл .gitignore
В него заносим все файлы и папки которые не хотим
индексировать



```
# Индексировать измененный файл

>git add

# Индексировать измененные файлы

>git add  .
```

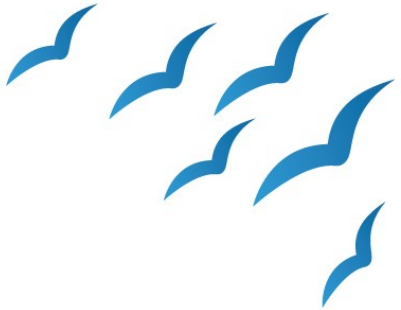


Удаление индексации файла

```
>git rm -cashed имя_файла
```

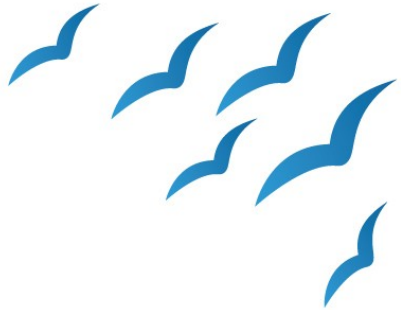
Удаление из индекса и из проекта

```
>git rm -f имя_файла
```



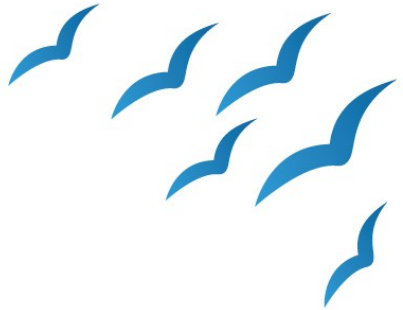
Просмотр изменений

```
>git status
```



```
# Фиксация изменения в локальное хранилище
# В коммит попадут (будут сохранены) только файлы,
# которые были проиндексированы командой git add
```

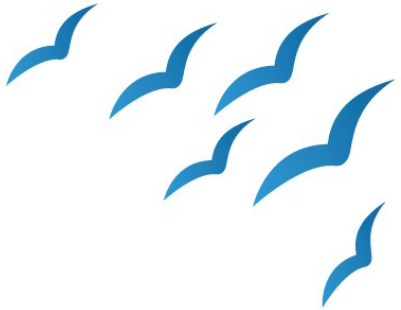
```
>git commit –m “комментарии к коду”
```



Просмотр истории коммитов

```
>git log
```

Информация о коммите (метаданные):
1. уникальный идентификатор коммита (хеш);
2. имя и email автора коммита;
3. дата создания коммита;
4. комментарий к коммиту.



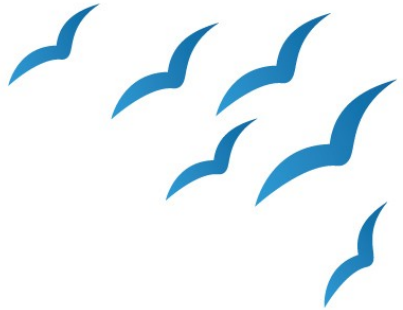
Связываение локального репозитория с GitHub

```
>git remote add origin
      git@github.com:my_name/my_repo.git
```

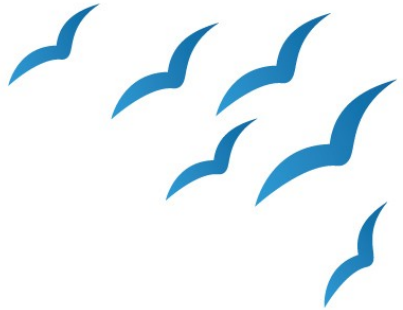
Где my_name – имя пользователя на GitHub
      my_repo – название созданного репозитория

Проверка свызывания
```
>git remote get-url origin
```

Удаление свызывания
```
>git remote remove origin
```

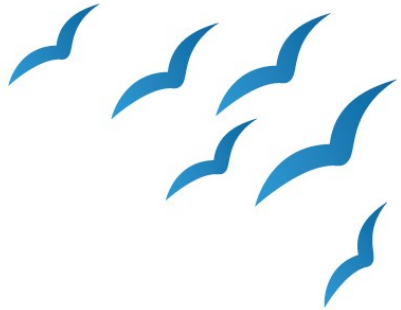


```
# Отправка изменений в удаленный
# репозиторий origin ветки master

>git push origin master

# Отправка изменений с текущей ветки

>git push

# Получение изменений

>git pull
```



Клонировать существующий репозиторий

```
>git clone ссылка-на-репозиторий
```



Задача !

```
# todo: Клонировать существующий репозиторий

>git clone
https://github.com/chertkov-
vitaliy/python_spring_2022.git
```

Задание

1) Создать репозиторий на GitHub с названием **python_spring_work_2022**
2) Создать папку на локальной машине для домашних работ с названием **python_spring_work_2022**
3) Превратить созданный каталог в репозиторий Git
4) Создать в папке каталога файл README.MD
5) Добавить его в локальный репозиторий
6) Связать удаленный репозиторий с локальным
7) Отправить изменения в удаленный репозиторий