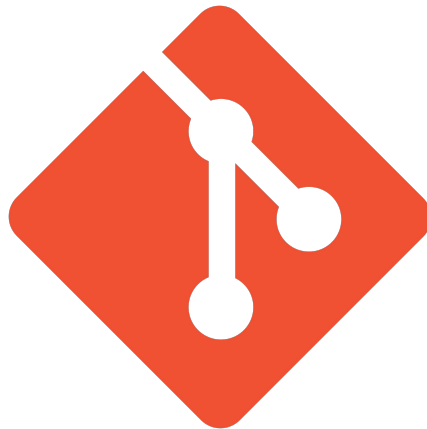


# ОСНОВЫ Git



# Что такое “система контроля версий”?

Система контроля версий - это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

Репозиторий - место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

# Система контроля версий

- Летописец
- Машина времени
- Резервная копия
- Мастер параллельных миров
- Объединяет коллективы



# Централизованные системы контроля версий

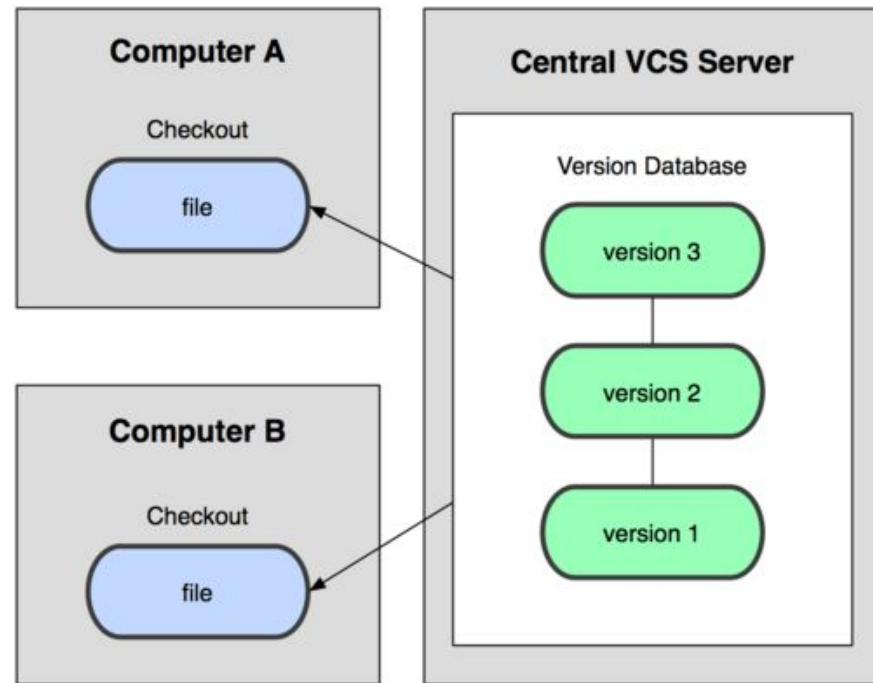
Пример: CVS, Subversion(SVN)

Идея:

Получаем последние версии файлов с сервера, изменяем и отправляем их на сервер.

Основной недостаток:

Вся история проекта хранится в одном месте. Если что-то происходит с сервером, то историю восстановить невозможно



# Распределенные системы контроля версий

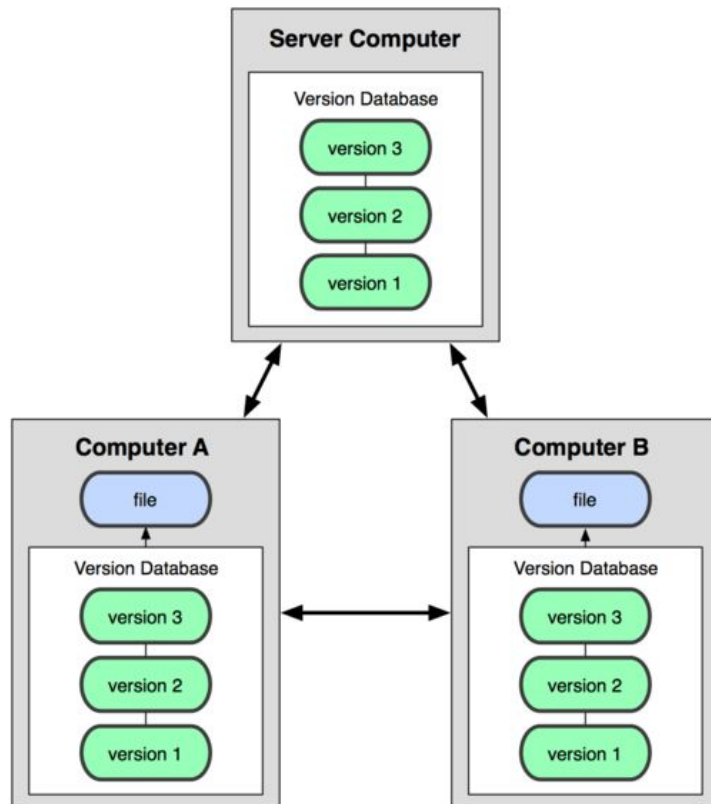
Пример: Git, Mercurial, Bazaar

Решение:

Полностью скопировать весь репозиторий себе на компьютер.

Таким образом у нас появляется локальный репозиторий и остается удаленный.

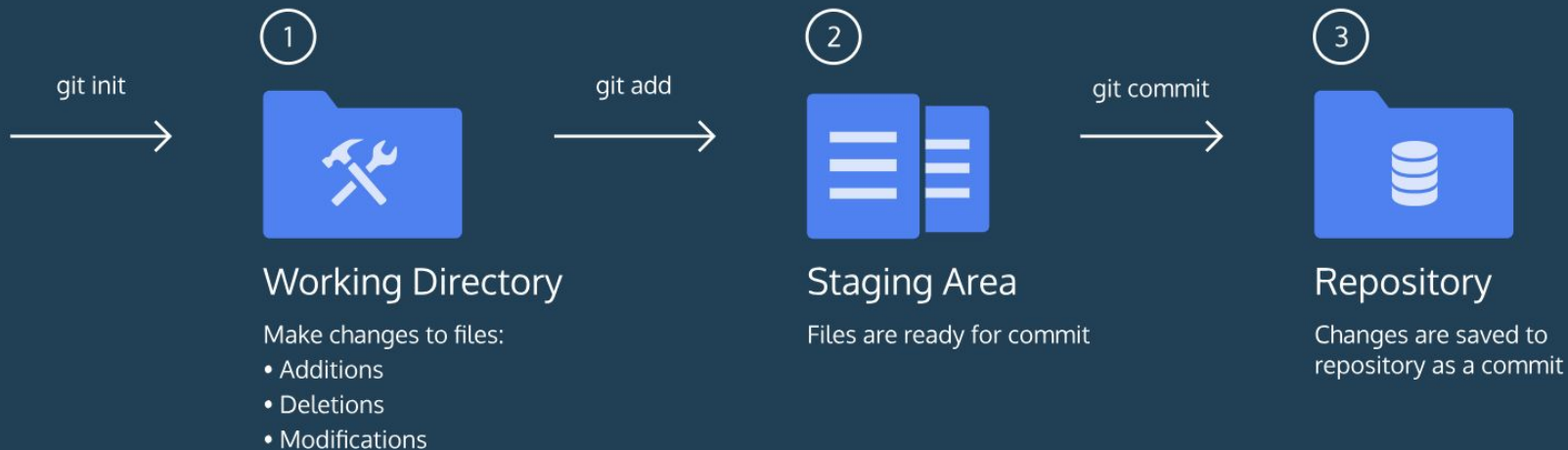
Если с удаленным репозиторием что-то происходит, то можно восстановить всю историю





# Работа с локальным репозитием

## Basic Git Workflow



# Как создать свой репозиторий?

*git init* - сделает текущую папку локальным репозиторием

# Как скопировать уже существующий репозиторий?

*git clone <url\_with\_git\_repo>* - создаст локальный репозиторий в текущей папке, который ссылается на удаленный репозиторий

Пример: `git clone https://github.com/tutorialzine/awesome-project.git`

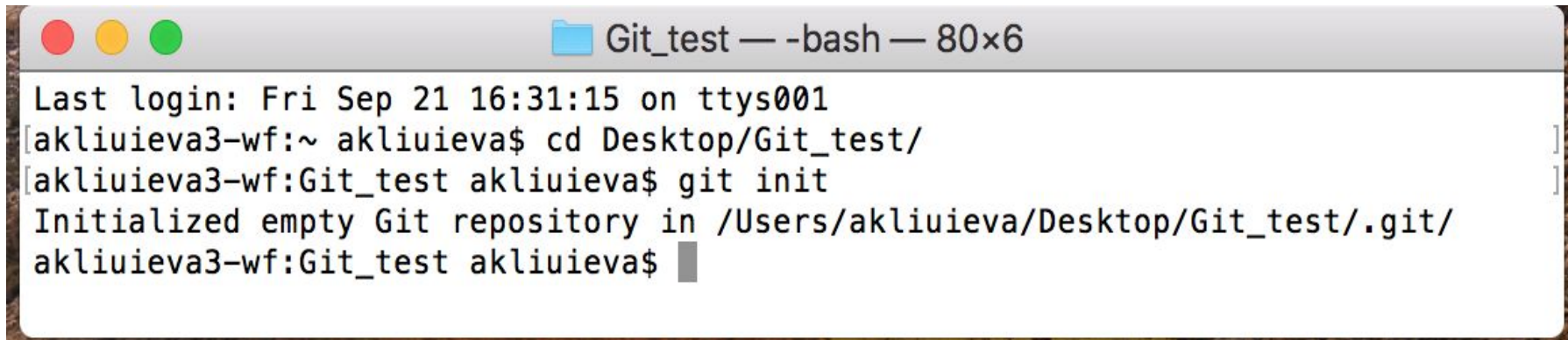


# Я сделал изменения у себя на компьютере. Что дальше?

- *git status* - показывает файлы, в которых были сделаны изменения в рабочей директории или в Staging Area
- *git add <filename>* - добавить файлы из рабочей директории (Working directory) в Staging Area
- *git commit -m "Commit message"* - переносит изменения из Staging Area в Local Repository
- *git diff* - показывает разницу между рабочей директорией и Staging Area
- *git log* - показывает список коммитов

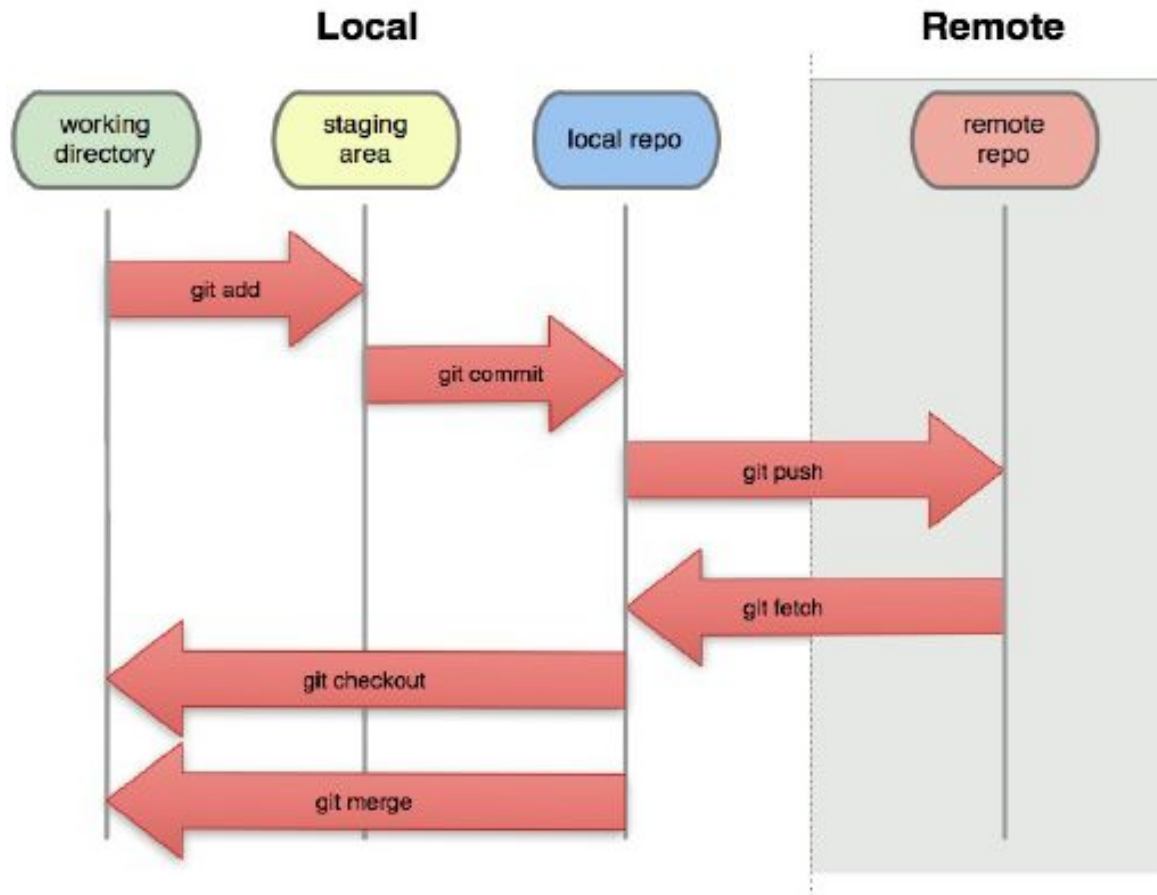
# Практика №1

1. Скачать git
2. Создаем локальный репозиторий:  
Для этого создадим пустую папку и перейдем в нее.  
Затем используем команду *git init*

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, a folder icon and the text "Git\_test — -bash — 80x6" in the center. The terminal text shows the login session, directory navigation to Desktop/Git\_test, and the successful execution of the git init command, which creates a .git directory.

```
Last login: Fri Sep 21 16:31:15 on ttys001
[akliuieva3-wf:~ akliuieva$ cd Desktop/Git_test/
[akliuieva3-wf:Git_test akliuieva$ git init
Initialized empty Git repository in /Users/akliuieva/Desktop/Git_test/.git/
akliuieva3-wf:Git_test akliuieva$
```

# Работа с удаленным репозиторием



# Как внести изменения в удаленный репозиторий?

- *git push origin branch\_name* - пересылает локальные коммиты в удаленный репозиторий
- *git fetch origin branch\_name* - берет изменения с удаленного репозитория, но это никак не влияет на локальные ветки и изменения
- *git pull origin branch\_name* - вливает изменения с удаленного репозитория в локальный репозиторий

Разница между *git fetch* и *git pull*:

*git pull* — это по сути команда *git fetch*, после которой сразу же следует *git merge*.

# Практика №2

## 1) Создаем удаленный репозиторий

В поле Repository name необходимо ввести имя репозитория, которое одновременно будет не слишком большим и в то же время информативным.

Например, test\_git.

## 2) Связываем локальный репозиторий и удаленный

После того, как репозиторий создан, его надо подключить к нашему локальному репозиторию.

Для этого используем следующие команды

```
git remote add origin %%server_path%%
```

## 3) Делаем изменения в локальном репозитории

Для этого добавляем один файл в папочку, подконтрольную гиту и добавляем файл в Staging area.

Для этого используем следующие команды

```
git add readme.txt
```

## 5) Проверим статус нашего репозитория с помощью команды git status

```
[akliuieva3-wf:Git_test akliuieva$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   readme.txt
```

## 6) Файл добавлен, теперь надо сохранить слепок изменений и залить его на сервер

```
[akliuieva3-wf:Git_test akliuieva$ git commit -m "Third commit from mac"
[master (root-commit) 23c1fde] Third commit from mac
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.txt
akliuieva3-wf:Git_test akliuieva$ █
```

7) Перенесем изменения на удаленный репозиторий

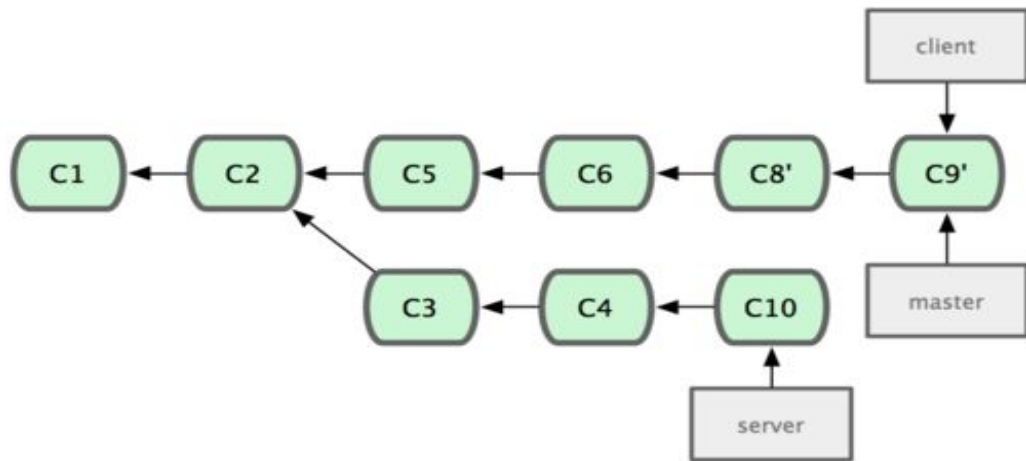
***git push origin master***

```
[akliuieva3-wf:Git_test akliuieva$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 215 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To gitlab.a-level.com.ua:Anna/empty_repo.git
 * [new branch]      master -> master
```



# Ветвление

Во время разработки новой функциональности считается хорошей практикой работать с копией оригинального проекта, которую называют веткой. Ветви имеют свою собственную историю и изолированные друг от друга изменения до тех пор, пока вы не решаете слить изменения вместе.



# Работа с ветками

- *git branch* - показывает список всех веток
- *git branch branch\_name* - создаст новую ветку
- *git checkout branch\_name* - переключится на ветку с именем branch\_name
- *git merge branch\_name* - используется для того, чтобы перенести изменения из ветки branch\_name в текущую ветку
- *git branch -d branch\_name* - удалит ветку с именем branch\_name

# Полезные ссылки

[Книги о GIT](#)

[Manual](#)

[Git Branching](#)

[Подробнее обо всех командах git](#)

[Быстрый старт с GIT](#)

[онлайн книга про Git](#)

# git merge

@girlie\_mac

♥ git merge incorporates changes into the current branch.



git meow-ge !!!



\* merge is like having 2 parents + 1 resulting child!

\* rebase adds all new Δs on top of one parent.

PAWSOME!!!

# git rebase

@girlie\_mac

♥ git rebase moves a branch from one commit to another.



Cleaner commit history!

If you want to rebase from the remote master instead of local, do either:

# git fetch origin

# git rebase origin/master

or

# git pull --rebase origin master

# Домашнее задание

Пройти уроки по Git из курса “Learn Git” на сайте codecademy (Basic Git workflow, how to backtrack in Git, git branching, git teamwork)

Link: <https://www.codecademy.com/learn/learn-git>

В результате:

Прислать скриншот с пройденными курсами.



# К собеседованию...

1. [Подробнее о системах контроля версий](#)
2. [Основные команды Git](#)
3. [Вопросы и ответы на собеседовании по Git](#)