Git

▼ commit

> `git commit`
> Отправляет снимок проекта

> Внести изменения при помощи `git commit --amend`

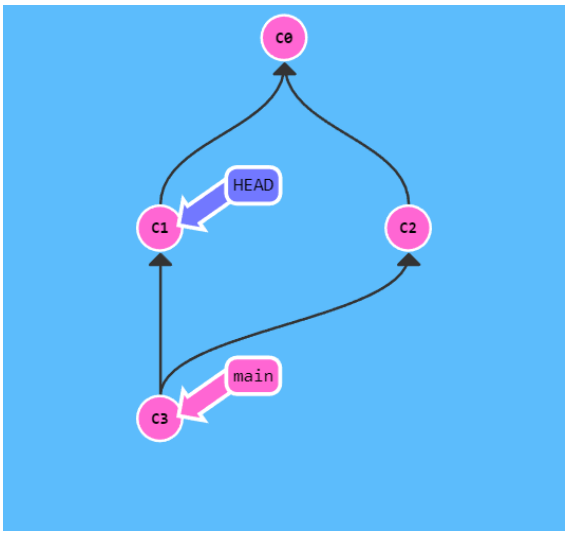
▼ branch

> `git branch [name]`
> Создаёт новую ветку
> `git branch -f [name] HEAD~N` или `git branch -f [name] [Hash]`
> Переместит ветку `[name]` на `~N` родителей назад от `HEAD`
> или переносит в определённый коммит `[Hash]`

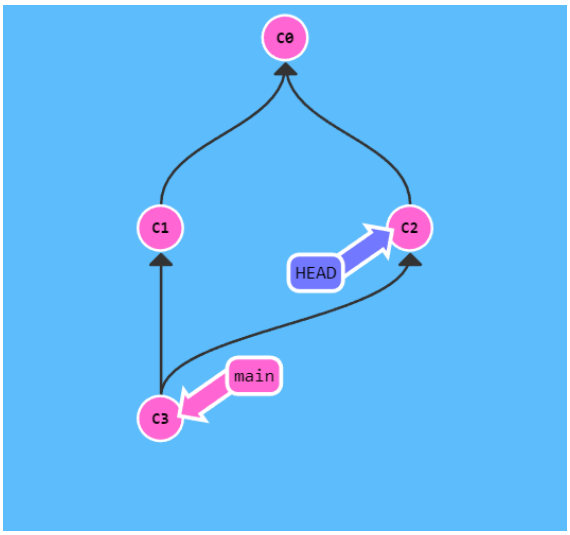
> `git branch bugWork main~^2~;` Создаст отсносительно ветки `main`
> `~^2~` комитов назад

▼ checkout

> `git checkout [name]`
> Для выбора ветки для взаимодействия



`git checkout main^`



`git checkout main^2`

▼ **cherry-pick**

> `git cherry-pick <Commit1> <Commit2> <...>`
>
> `git cherry-pick C2 C4` создаст коммит под хэшем (коммитом) `C2'` `C4'`

▼ **reset**

> `git reset`
>
> Отменяет изменения, перенося ссылку на ветку назад, на более старый коммит. Это своего рода "переписывание истории".
>
> `git reset HEAD~1`

▼ **revert**

> `git revert`
>
> Перенесёт ветку назад, как будто некоторых коммитов вовсе

> и не было. Создаёт коммит противоположный предыдущему комиту.
> `git revert HEAD`

▼ **rebase**

> `git rebase [name]`
> Перемещает текущую ветку в ветку `[name]`. Это не слияние

> `git rebase -i [hash]` Перемещает текущую ветку на несколько комитов.
> `git rebase -i HEAD~4` интерактивный rebase, например, с помощью графического окна

> • Переставить коммит так, чтобы нужный находился наверху при помощи `git rebase -i`

▼ **merge**

> `git merge [name]`
> Соединяет ветку `[name]` в текущую ветку

▼ **head**

> `git head [commit name]`
> Выбор комита

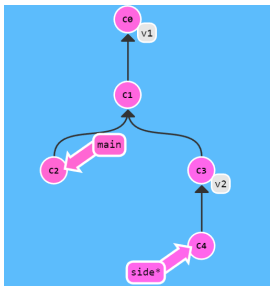
▼ **Tag**

> `git tag [text] [hash]`
> `git tag v1 C1` создаст тег “v1” на комите C1

▼ **describe**

> `git describe <ref>`
> Если не указать `ref`, то git будет считать, что указано текущее положение (`HEAD`).

> **Вывод команды выглядит примерно так:**
> `<tag>_<numCommits>_g<hash>`
>
> Где `tag` – это ближайший тег в истории изменений, `numCommits` – это на сколько далеко мы от этого тега, а `hash` – это хеш коммита, который описывается.



Команда `git describe main` выведет:

```
v1_2_gC2
```

Тогда как `git describe side` выведет:

```
v2_1_gC4
```

▼ clone

> Технически, `git clone`
> в реальной жизни - это команда, которая создаст *локальную* копию удалённого репозитория

> Важным свойством удалённых веток является тот факт, что когда вы извлекаете их, вы отделяете (detaching) `HEAD`.
> Git делает это потому, что вы не можете работать непосредственно в этих ветках;

▼ fetch

> Фактически, `git fetch` синхронизирует *локальное* представление удалённых репозиториев с тем, что является *актуальным* на текущий момент времени.

▼ **pull**

> Процедура *скачивания (fetching)* изменений с удалённой ветки и *объединения (merging)* настолько частая и распространённая, что git предоставляет вместо двух команд - одну! Эта команда - `git pull`

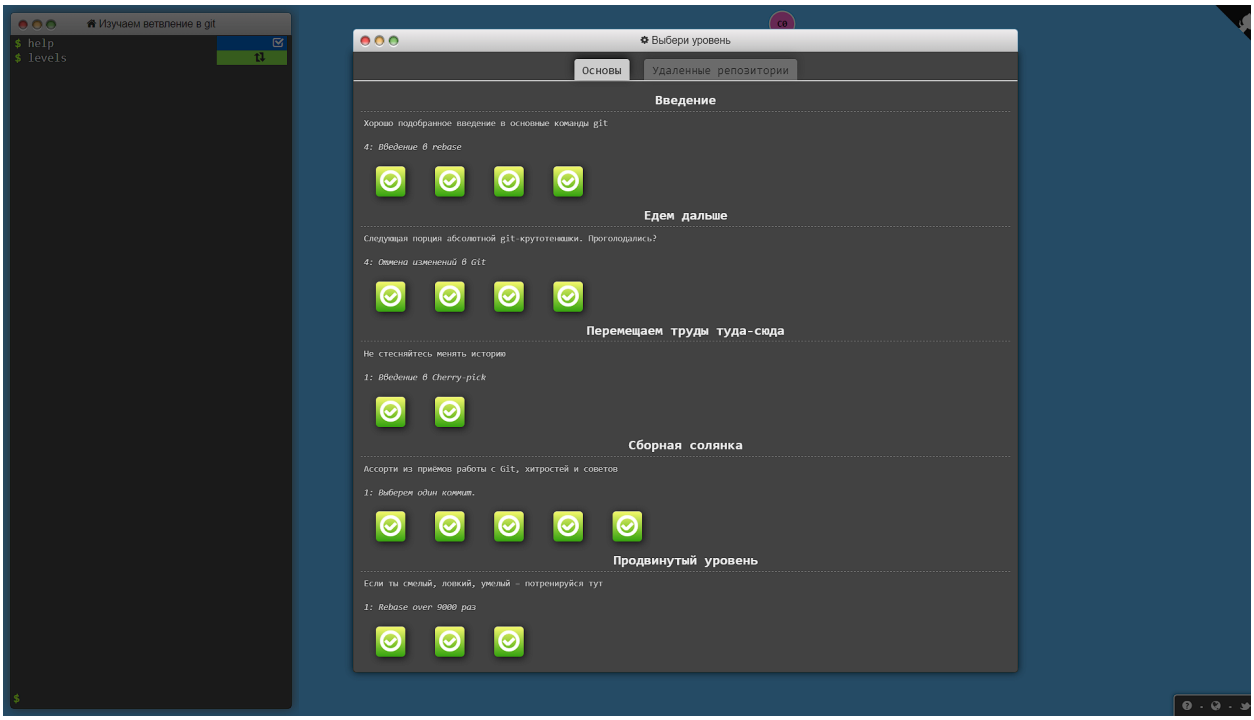
> `git fetch; git merge 0/main` == `git pull`

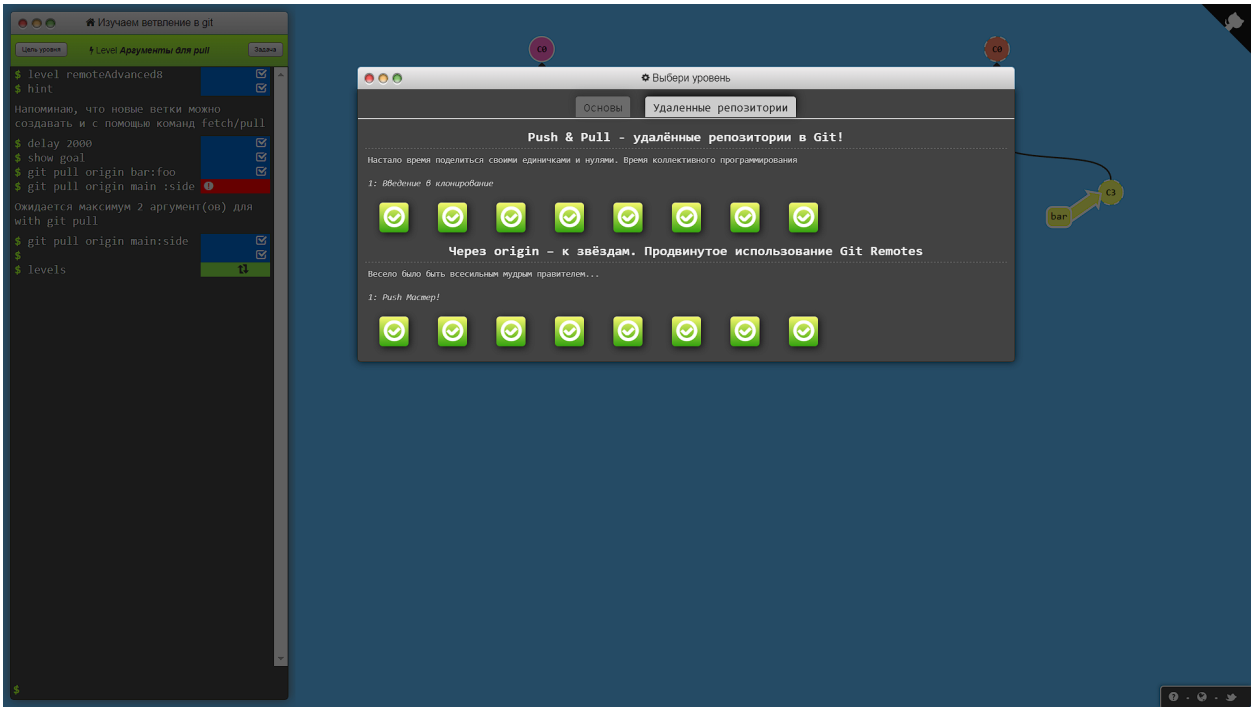
▼ **push**

Относительные ссылки - мощный инструмент, но мы покажем два простых способа использования:

- Перемещение на один коммит назад `^`
- Перемещение на несколько коммитов назад `~<num>`

> `git checkout main^` = `git head C2`


Изучаем ветвление в git
$ help
$ levels
Выбери уровень
Основы
Удаленные репозитории
Введение
Хорошо подобранное введение в основные команды git
4: Введение в rebase
Едем дальше
Следующая порция абсолютной git-крутотеньки. Проголодались?
4: Отмена изменений в Git
Перемещаем труды туда-сюда
Не стесняйтесь менять историю
1: Введение в Cherry-pick
Сборная солянка
Ассорти из приёмов работы с Git, хитростей и советов
1: Выберем один коммит.
Продвинутый уровень
Если ты смелый, ловкий, умелый - потренируйся тут
1: Rebase over 9000 раз


Изучаем ветвление в git
Цель уровня
Level Аргументы для pull
Задача
$ level remoteAdvanced8
$ hint
Напоминаю, что новые ветки можно создавать и с помощью команд fetch/pull
$ delay 2000
$ show goal
$ git pull origin bar:foo
$ git pull origin main :side
Ожидается максимум 2 аргумент(ов) для with git pull
$ git pull origin main:side
$
$ levels
Выбери уровень
Основы
Удаленные репозитории
Push & Pull - удалённые репозитории в Git!
Настало время поделиться своими единичками и нулями. Время коллективного программирования
1: Введение в клонирование
Через origin - к звёздам. Продвинутое использование Git Remotes
Весело было быть всесильным мудрым правителем...
1: Push Мастер!
C3
bar