

## ЛАБОРАТОРНАЯ РАБОТА №7

### Добавление ветки наблюдения

Ветки, которые начинаются с `remotes/origin` являются ветками оригинального репозитория. Обратите внимание, что у вас больше нет ветки под названием `style`, но система контроля версий знает, что в оригинальном репозитории ветка `style` была.

1. Добавьте локальную ветку, которая отслеживает удаленную ветку.

#### Выполните

```
git branch --track style origin/style
git branch -a
git log --max-count=2
```

#### Результат

```
$ git branch --track style origin/style
Branch 'style' set up to track remote branch 'style' from 'origin'.
$ git branch -a
* main
  style
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/style
$ git log --all --graph --max-count=2
* 71df43a 2023-11-28 | Changed README in original repo (HEAD -> main, origin/main, origin/HEAD) [Alexander Shvets]
* 39a1e0f 2023-11-28 | Renamed hello.html; moved style.css (origin/style, style) [Alexander Shvets]
```

Теперь мы можем видеть ветку `style` в списке веток и логе.

### Чистые репозитории

Чистый репозиторий (голый/`bare`) — это репозиторий, не имеющий рабочей директории. Он содержит только директорию `.git`, в которой Git хранит все свои внутренние данные. Основное предназначение таких репозиториях — быть центральным хранилищем, в которое разработчики могут отправлять и из которого могут получать данные. Поэтому в них нет смысла создавать рабочие файлы, они будут только впустую занимать место на диске. Чистые репозитории также используются в сервисах Git-хостинга таких, как GitHub и GitLab.

Он нужен только для клонирования и обмена коммитами через `pull/push`. При клонировании из него создаются файлы в последнем состоянии и записывается история изменений. А при обмене коммитами оттуда или туда скачиваются только коммиты.

1. Создайте чистый репозиторий

#### Выполните

```
cd ..
git clone --bare work work.git
ls work.git
```

Сейчас мы находимся в директории `repositories`.

#### Результат

```
$ git clone --bare work work.git
Cloning into bare repository 'work.git'...
done.
$ ls work.git
branches
config
description
HEAD
hooks
info
objects
packed-refs
refs
```

Принято считать, что репозитории, заканчивающиеся на `.git`, являются чистыми репозиториями. Мы видим, что в репозитории `work.git` нет рабочей директории. По сути, это просто директория `.git` из обычного репозитория.

### Добавление удаленного репозитория

Давайте добавим репозиторий `work.git` к нашему оригинальному репозиторию.

#### Выполните

```
cd work
git remote add shared ../work.git
```

Сейчас мы находимся в репозитории `work`.

### Отправка изменений

Поскольку чистые репозитории обычно располагаются на каком-либо удаленном сервере, вы не сможете туда просто зайти, дабы подтянуть изменения. Поэтому нам необходимо как-нибудь передать наши изменения в репозиторий.

Начнем с создания изменения, которое нужно передать в репозиторий. Отредактируйте `README` и закоммитьте его:

#### Файл: README

```
This is the Hello World example from the Git tutorial.
(changed in the origin and pushed to shared)
```

#### Выполните

```
git switch main
git add README
git commit -m "Added shared comment to readme"
```

Теперь отправьте изменения в общий репозиторий.

#### Выполните

```
git push shared main
```

Общим называется репозиторий, получающий отправленные нами изменения. Мы добавили его в качестве удаленного репозитория в предыдущих шагах.

#### Результат

```
$ git push shared main
To ../work.git
71df43a..d9d2bea  main -> main
```

### Подтягивание общих изменений

Быстро переключитесь в репозиторий home и подтяните изменения, только что отправленные в общий репозиторий.

#### Выполните

```
cd ../home
```

Сейчас мы находимся в репозитории `home`.

#### Выполните

```
git remote add shared ../work.git
git branch --track shared main
git pull shared main
cat README
```

### Размещение ваших Git-репозиториев

Чтобы создать свой собственный GitHub, существует множество способов совместного использования репозиториев Git по сети. Здесь приведен простой и быстрый (но ненадежный и опасный) способ.

#### 1. Запуск Git-сервера

#### Выполните

```
# (From the "repositories" directory)
git daemon --verbose --export-all --base-path=.
```

Теперь в отдельном окне терминала перейдите в вашу директорию repositories:

#### Выполните

```
# (From the "repositories" directory)
git clone git://localhost/work.git network_work
cd network_work
ls
```

Вы увидите копию проекта work.

#### 2. Отправка изменений в Git Daemon

Если вы хотите разрешить отправку изменений (push) в репозиторий Git Daemon, добавьте метку `--enable=receive-pack` к команде `git daemon`. Будьте осторожны, этот сервер не производит аутентификацию, поэтому любой сможет отправлять изменения в ваш репозиторий.

### **03 Совместное использование репозитория**

На этом этапе вам открываются безграничные возможности. Смелее! Возьмите в аренду сервер, купите доменное имя, разместите на этом сервере свои репозитории и наслаждайтесь своим личным GitHub!

Если серьезно, то вы можете самостоятельно разместить свой личный сервер [GitLab](#). Этот продукт бесплатный и с открытым исходным кодом.

Полезные ссылки.

Отличие GitHub от GitLab. [https://itlogia.ru/article/gitlab\\_i\\_github\\_v\\_chem\\_razlichiya](https://itlogia.ru/article/gitlab_i_github_v_chem_razlichiya)

Интерфейс GitHub <https://skillbox.ru/media/code/cto-takoe-github-i-kak-im-polzovatsya/>