GIT Top Commands

???????? STATUS

Инициализация каталога как Git-репозитория

```
git init
```

Команда показывающая какие файлы в каком состоянии находятся

```
git status
```

Показать историю коммитов в текущей ветке / всех ветках

```
git log
git log --all
```

???????? BRANCH ?????????

Отобразить активные ветки / все активные и доступные ветки

```
git branch
git branch -a
```

Переключиться на другую ветку

```
git checkout <branch-name>
```

Создать новую ветку и переключиться на неё

```
git checkout -b <branch-name>
```

Проиндексировать все файлы в папке (для commit)

```
git add .
```

Удалить ветку из remote REPO

```
git push origin --delete develop
```

Отправка всех веток локального репозитория на удалённый репозиторий

```
git push <link.repo> --all
git push --force <branch-name>
```

???????? ? COMMITs

Создать коммит с указанием имени

```
git commit -m "new commit"
```

Автоматическая индексация всех уже отслеживаемых файлов

```
git commit -a
git commit -am "new commit" (автоиндекс+имя)
```

Редактор названия коммитов по-умолчанию

```
git config --global core.editor "vim"
```

Изменить сообщение последнего коммита

```
git commit --amend
```

Забыть все последующие коммиты от указанного по <hash>

```
git reset --hard <hash>
```

Изменить дату текущего коммита

```
GIT_COMMITTER_DATE="Thu Jun 08 13:44:15 2023 +0300" git commit --amend
```

Изменение Первого коммита

```
# Автоматически определяем первый коммит репозитория
TARGET_COMMIT=$(git rev-list --max-parents=0 HEAD)
git checkout -b temp-fix $TARGET_COMMIT
# Вносим правки
```

```
vim README.md
git add .
git commit --amend --no-edit
# Переносим все последующие коммиты поверх исправленного
git cherry-pick ${TARGET_COMMIT}..main
# Обновляем main
git checkout main
git reset --hard temp-fix
git branch -d temp-fix
# Push в репозиторий
git push --force-with-lease origin main
```

Изменение Среднего коммита

```
TARGET_COMMIT="525c65a"
BASE_COMMIT=$(git rev-parse ${TARGET_COMMIT}^)
git checkout -b temp-fix $BASE_COMMIT
git cherry-pick $TARGET_COMMIT
# Вносим правки
vim README.md
git add .
git commit --amend --no-edit
# Переносим все последующие коммиты поверх исправленного
git cherry-pick ${TARGET_COMMIT}..main
# Обновляем main
git checkout main
git reset --hard temp-fix
git branch -d temp-fix
# Push в репозиторий
git push --force-with-lease origin main
```

Изменение Последнего коммита

```
vim README.md
git add .
git commit --amend --no-edit
git push --force-with-lease origin main
```

???????? ? TAGs

Создать новый Tag

```
git tag -a v1.0 -m "feature01"
git tag --list
```

Отправить Tag на удалённый репозиторий

```
git push origin v1.0
```

???????? ? ALIASes

Алиасы для графического формата команды git log:

Подключить алиас с ключом --all (отображение всех ветвей)

```
git config --global alias.log0 "log \
  --graph \
  --abbrev-commit \
  --decorate \
  --format=format:'\
%C(bold blue)%h%C(reset) - \
%C(bold green)(%ci)%C(reset) \
%C(white)%s%C(reset) \
%C(dim white)- %an%C(reset)\
%C(auto)%d%C(reset)' \
  --all"
```

Подключить алиас с ключом --branches (отображение локальных веток)

```
git config --global alias.log1 "log \
  --graph \
  --abbrev-commit \
  --decorate \
  --format=format:'\
%C(bold blue)%h%C(reset) - \
%C(bold green)(%ci)%C(reset) \
%C(white)%s%C(reset) \
%C(dim white)- %an%C(reset)\
%C(auto)%d%C(reset)' \
  --branches"
```

Отключить алиас

```
git config --global --unset alias.log1
```

???????? ? REPOs

Клонировать удалённый репозиторий (папка будет создана)

```
git clone <link.repo>
```

Клонировать удалённый репозиторий по branch+tag

```
git clone <link.repo> --branch '<tag-name>'
```

Скачать изменения без слияния (fetch)
Изменения сохраняются в FETCH_HEAD

```
git fetch origin develop
```

Посмотреть HASH скачанного коммита

```
git rev-parse FETCH_HEAD         # полный hash
git log FETCH_HEAD -1 --oneline  # краткий hash
git diff HEAD FETCH_HEAD         # посмотреть изменения
```

Слить изменения

Способ 1: Через временную ветку

```
git branch temp-branch FETCH_HEAD # создать ветку
git merge temp-branch             # смержить
git branch -d temp-branch         # удалить ветку
```

Способ 2: Через HASH

```
git merge FETCH_HEAD     # смержить напрямую
git merge <hash>         # смержить по hash
git pull origin develop  # fetch + merge одной командой
```

??????? ???????? ? REPOs

Пример с Новым Repo

```
# Инициализация и Push-инг Commit-а в Новый Repo
git init --initial-branch=main
git config --global user.name "xeon"
git config --global user.email "xeonmp22@gmail.com"
#git remote set-url origin git@gitlab.com:shions/k8s.git
#git remote remove origin
git remote add origin git@gitlab.com:shions/k8s.git
vim README.md
git add .
git commit -am "01: Kubernetes Request Lifecycle"
# При 1-ом Commit-е делаем --set-upstream
git push -u origin main
# Для последующих Commit-ов
git push --all
```

Пример с Рабочим Repo

```
# Очистка рабочего Repo от файлов Старых Commit-ов
cd $HOME/rebrainme/devops/07_cicd
git rm -rf roles/
cd ../07_cicd.dev
rm -rf .git

# Копирование новых файлов в рабочий Repo
cp -r .gitignore main.tf playbook.yml README.md roles/ ../07_cicd
cd ../07_cicd
git add .
git commit -am "34: Logging 6/prom.grafana"
git push --all
```

Revision #37
Created 2026-06-22 13:13:43 UTC by sa
Updated 2026-11-02 08:46:52 UTC by xeon