<https://western-appeal-39b.notion.site/Git-Nov-29-dd3046b5533f4348ba1d2b6f238371a3>

**Git, Nov 29**

**Вопросы**

1. **Чем отличается проект от репо?**
   1. **проект** - кодовая база
   2. **репо** - файловая БД для истории разработки (каталог .git/)
2. **Что такое ветвь?**
   1. Ветка - еще одна версия проекта (изолированный поток разработки)
3. **Подходы/стратегии к ветвлению?**
   1. **Git Flow**
      1. master/main
      2. develop/current
      3. short-term branches (hotfix1, bugfix1, issue1)
   2. **GitHub Flow**
      1. long-term branches
      2. short-term branches
4. **Подходы к именованию ветвей?**
   1. Тематические (login, payments, bugfix)
   2. Релизные (v1.2.3)
   3. Смешанный (использует оба)
5. **Как посмотреть подробный список веток?**

git branch -v

git branch -av

1. **Как часто рекомендуется выполнять коммиты?**

По завершении отдельной небольшой задачи (логической).

1. **Что такое коммит?**

Коммит - фиксация/сохранение изменений

git commit -m ‘описание’

git commit -am ‘описание’

1. **Какие состояния файлов существуют?**
   1. Отслеж.
   2. Неотслеж. (по умолчанию)
   3. Игнорир.
2. **Легенда для команды git status -s ?**
   1. -s (short)
   2. ?? - Untracked
   3. M - Modified
   4. D - Deleted
   5. A - Added
3. **Порядок создания нового проекта с исп. GIT**
   1. Создать каталог проекта и поместил несколько файлов в него (локально)
   2. Инициализация репо git init (делается только один раз)
   3. git add .
   4. git commit -m ‘Initial commit’
   5. Опционально выгрузить на GitHub:
      1. git remote add origin скопированная\_ссылка (создание привязки)
      2. git push -u origin master (первый раз)
      3. git push (второй и послед. разы)
4. **Что такое слияние веток и как оно выполняется?**

Слияние - объединение (перенос или интеграция изменений)

git merge название\_ветки

git merge —abort (прервать слияние при наличии конфликтов)

1. **Что делает git diff ?**

Показывает изменения между раб/каталогом и индексом (очередью на коммит)

1. **Как посмотреть последний коммит?**

git show

1. **Переключение раб/каталога между коммитами (”машина времени”)**

git checkout <commit\_id>

**Сохранение незакоммиченных изменений (stash)**

git stash (создать “заначку”)

git stash pop (применить последнюю заначку и удалить ее списка заначек)

git stash list (посмотреть список заначек)

git stash save 'описание' (создать заначку с комментарием)

git stash show (просмотр статистики по заначке)

git stash show -p (просмотр заначки)

git stash apply (применить заначку без ее удаления)

git stash clear (удалить все заначки)

git stash drop stash\_id (удалить конкретную заначку)

**Пример системы комментирования (описания) коммитов**

git commit -m ‘какое-то описание’

**характер\_правки(модуль): короткое описание**

fix(payments): поправил баг оплаты для Visa

feat() - новый функционал

docs() - вопросы документации

build() - вопросы сборки

style() - вопросы стилей/оформления

**Безопасная отмена коммитов**

git revert commit\_id (отменить коммит)

**Клонирование (выгрузка) репозитория**

git clone ссылка\_на\_репо

**Просмотр коммита**

git show commit\_id

git show commit\_id:filename

**Просмотр журнала коммитов в сжатом виде**

git log --oneline

**Перезапись последнего коммита**

git commit --amend -m ‘описание изменений’

git commit --amend --no-edit

--no-edit (не изменять комментарий)

**Что делать, если выполнил коммит не на той ветке - а надо было на новой**

git branch новая\_ветка (ответвится)

git reset --hard HEAD~1 (удалить посл/коммит на тек/ветке)

**Модели совместной разработки**

1. Fork + Pull
2. Общий доступ к репо

**Работа по первому сценарию**

Fork - копия репо

1. Создать fork на GitHub
2. Клонировать уже свой новый репозиторий на компьютер
3. Создать новую ветку
4. В это ветке внести предложение (коммит)
5. Выгрузить на GitHub
6. Оформить Pull Request (запрос слияния)