🏷️ Теги и версии

Что такое теги (tags) в Git?

Теги в Git — это удобный способ обозначить **важные точки в истории репозитория**, чаще всего — **релизы**.

📦 Пример: `v1.0`, `v2.3.4`, `beta`, `release-2025`

Зачем нужны теги?

- Отметить релиз (версии программного обеспечения)
- Быстро вернуться к определённому коммиту
- Упрощённая навигация по истории
- Интеграция с CI/CD (теги часто запускают сборку/деплой)

Лёгковесные (lightweight) теги

- Просто **ссылка на коммит**
- Не содержит метаданных (дата, имя, сообщение)
- Быстро создаются, похожи на обычные ветки

```
git tag v1.0.0
```

Такой тег локальный, его нужно пушить отдельно!

```
git push origin v1.0.0
```

Аннотированные (annotated / тяжёлые) теги

- Содержат имя автора, дату, комментарий
- Хранятся как полноценные объекты в Git
- Подходят для релизов и публичной версии

```
git tag -a v1.0.0 -m "Release version 1.0.0"
git push origin v1.0.0
```

Отправить все теги

```
git push origin --tags
```

Просмотр тегов

Показать все теги:

```
git tag
```

Найти теги по шаблону:

```
git tag -l "v1.*"
```

Просмотр информации о теге

```
git show v1.0.0
```

Удаление и перемещение тегов

Локально удалить тег

```
git tag -d v1.0.0
```

Удалить тег в удаленном репозитории

```
git push origin --delete v1.0.0
```

Чтобы переместить тег его нужно пересоздать

```
git tag -d v1.0.0
git tag -a v1.0.0 -m "Moved tag" <new_commit_hash>
git push --force origin v1.0.0
```

Семантическое версионирование

Теги часто используют формат:

```
v<MAJOR>.<MINOR>.<PATCH>
```

Пример: **v1.2.3**

1. — **Major** (обратная несовместимость)
2. — **Minor** (новый функционал без ломания)
3. — **Patch** (исправление багов)