

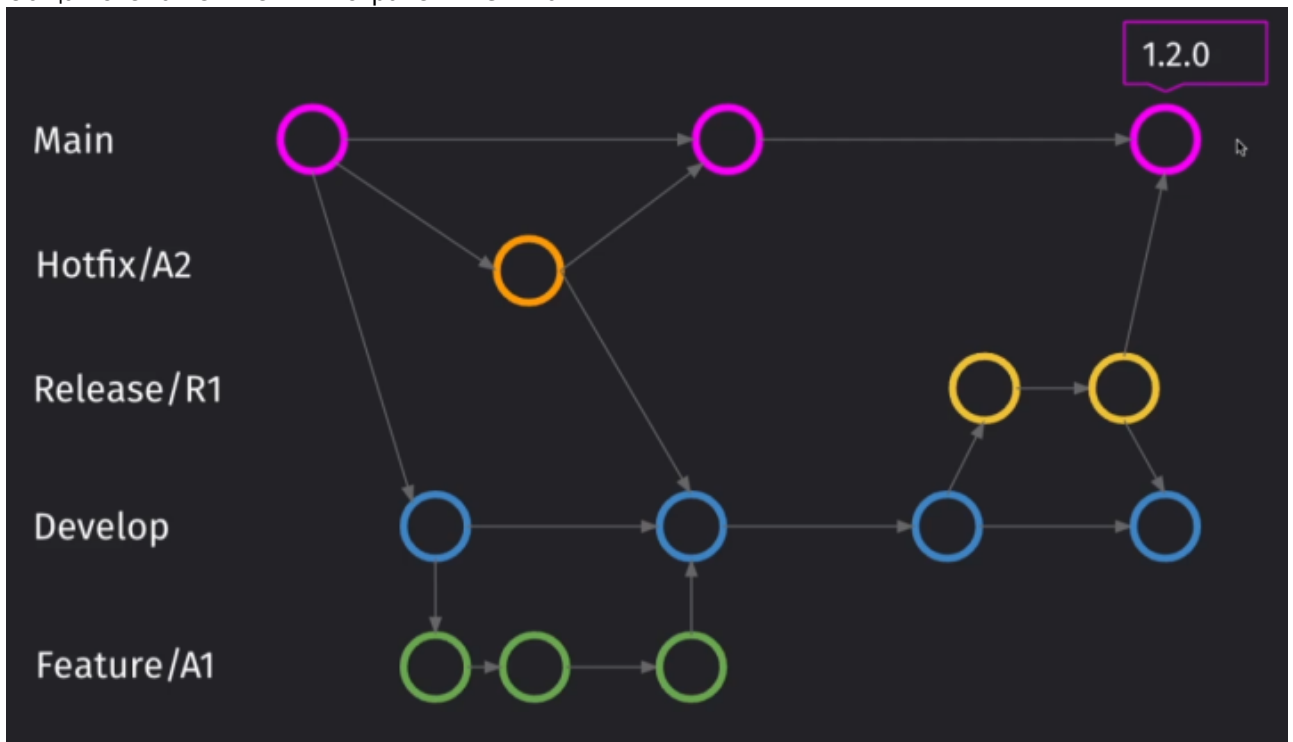
Стратегии работы с Git

Обзоры основных стратегий:

- [Branching Strategies Explained \(18:18\)](#) от DevOps & AI Toolkit
- [3 Git Workflows Every Developer Should Know \(31:31\)](#) от TechWorld with Nana
- [Что такое Git flow и когда использовать? \(11:43\)](#) от PurpleSchool | Anton Larichev
- [Почему Trunk based development вместо Git Flow? \(17:19\)](#) от PurpleSchool | Anton Larichev

Кратко про Git Flow

- Хрестоматийная статья о Git Flow:
 - [A successful Git branching model](#) от Vincent Driessen
- Общая схема ветвления в стратегии Git Flow:



- Виды слияния при следовании стратегии Git Flow:

feature/* → develop	squash merge
release/* → main	обычный --no-ff merge
release/* → develop	обычный --no-ff merge
hotfix/* → main	обычный --no-ff merge
hotfix/* → develop	обычный --no-ff merge

- Squash слияние feature/* → develop

```
git switch develop
git merge --squash feature/my-feature
git commit -m "Add my feature"
git branch -D feature/my-feature
```

```
git push origin --delete feature/my-feature
```

Squash слияние не создаёт merge-коммит автоматически. Оно переносит все изменения из feature/my-feature в рабочее дерево и индекс, после чего требуется самостоятельно сделать обычный коммит. Для feature/fix-веток, которые вливаются через squash, удаление через -D - это ожидаемая ситуация.

Алгоритм релиза Git Flow

1. В ветке develop полностью подготовить код проекта к релизу и зафиксировать последние правки коммитом.
2. От последнего коммита в ветке develop создать релизную ветку release/X.Y.Z:

```
git switch -c release/X.Y.Z
```

3. Изменить версию в CMakeLists.txt:

```
vim CMakeLists.txt
```

4. Обновить debian/changelog:

```
./scripts/make-deb-changelog.sh
```

5. В текущей ветке release/X.Y.Z сделать коммит с фиксацией результатов подготовки к релизу Prepare release X.Y.Z

```
git commit -a -m "Prepare release X.Y.Z"
```

6. Слияние изменений в main:

```
git switch main  
git merge --no-ff release/X.Y.Z
```

Рекомендуемое название коммита Release X.Y.Z

7. Отправка ветки main на сервер:

```
git push
```

8. Установка метки на свежий релиз:

```
git tag -a vX.Y.Z -m "Release X.Y.Z"
```

Метка устанавливается до переключения на ветку develop,

9. Отправка метки на GitHub:

```
git push origin vX.Y.Z
```

10. Слияние релизной ветки в develop.:

```
git switch develop
```

```
git merge --no-ff release/X.Y.Z
```

При этом рекомендуется название коммита по умолчанию Merge branch «release/X.Y.Z» into develop, чтобы различать истинный релиз в ветке main и обратное слияние из release/X.Y.Z в develop.

11. Отправка ветки develop на сервер:

```
git push
```

12. Удаление локальной релизной ветки release/X.Y.Z:

```
git branch -d release/X.Y.Z
```

From:

<https://jurik-phys.net.ru/> - Jurik-Phys.Net.Ru

Permanent link:

<https://jurik-phys.net.ru/itechnology:release>

Last update: 2026/10/08 01:30

