Конспект по Git Flow, GitHub Flow, GitLab Flow и правилам командной работы в Git

---

1. Git Flow — Полнофункциональная модель работы с ветками

Git Flow — это методология управления ветками в Git, которая идеально подходит для проектов с длительными циклами разработки и регулярными релизами. Этот процесс подразумевает четкое разделение между ветками для разработки, релизов и исправления багов.

Основные ветки:

- **`main`** (или `master`): Стабильная ветка, содержащая релизную версию продукта.
- **`develop`**: Основная ветка для разработки. Все новые функции и исправления в первую очередь попадают в эту ветку.

Вспомогательные ветки:

- **`feature/*`**: Ветки для разработки новых функций. Создаются от ветки `develop` и после завершения сливаются обратно в нее.
- **`release/*`**: Ветки для подготовки к релизу. Создаются от `develop`, когда проект близок к выпуску, для финальных доработок и тестирования.
- **`hotfix/*`**: Ветки для срочных исправлений, когда требуется внести изменения напрямую в стабильную версию (например, для быстрого исправления критических ошибок). Создаются от `main` и сливаются в `main` и `develop`.

Основной процесс:

1. Ветка для новой фичи создается от `develop`.
2. После завершения работы над фичей она сливается обратно в `develop`.
3. Когда проект готов к релизу, создается ветка `release`, где проводится финальное тестирование.
4. После тестирования ветка `release` сливается в `main`, и происходит деплой.
5. Для быстрого исправления багов создаются ветки `hotfix`, которые сливаются в обе ветки — `main` и `develop`.

Плюсы Git Flow:

- Четкое разделение стадий разработки, что упрощает управление релизами.
- Позволяет легко параллельно разрабатывать несколько фич и выпускать обновления.

Минусы:

- Может быть сложен для небольших проектов с быстрыми циклами разработки.
- Не подходит для проектов с непрерывной интеграцией, так как требует более строгого управления ветками.

---

2. GitHub Flow — Простая модель для проектов с непрерывной интеграцией и быстрыми релизами

GitHub Flow — это упрощенная модель работы с Git, разработанная для проектов, которые требуют частых релизов и использования непрерывной интеграции и деплоя (CI/CD). Основной принцип заключается в том, что каждая новая функция или исправление разрабатывается в отдельной ветке, и после завершения работы она сливается в `main`.

Основные принципы:

- **`main`**: Единственная основная ветка, в которой всегда находится самая актуальная и стабильная версия кода.

Основной процесс:

1. Для каждой новой функции или задачи создается новая ветка от `main`.
2. Разработка и коммиты происходят в новой ветке.
3. После завершения работы над задачей создается Pull Request (PR), который проверяется и утверждается другими участниками команды.
4. После утверждения ветка сливается с `main`, и происходит автоматический деплой (если настроена система CI/CD).

Плюсы GitHub Flow:

- Простота и гибкость, что делает этот процесс идеальным для проектов с быстрыми релизными циклами.
- Поддержка непрерывной интеграции и деплоя.

Минусы:

- Не подходит для крупных проектов с долгими циклами разработки, требующими сложной координации.
- Требует хорошего покрытия тестами, так как любые изменения сразу попадают в `main` и могут быть выпущены в продакшн.

---

3. GitLab Flow — Гибкая модель с учетом различных окружений

GitLab Flow — это модель, которая сочетает элементы GitHub Flow с возможностью интеграции разных окружений (например, `staging`, `production`). Этот подход позволяет использовать разные ветки для тестирования и продакшн, что делает его удобным для проектов с множеством этапов разработки и тестирования.

Основные ветки:

- **`main`**: Основная ветка, которая содержит стабильную версию продукта.
- **`environment/*`**: Ветки, созданные для разных окружений (например, `staging` для тестирования и `production`для выпуска).

Основной процесс:

1. Каждая новая функция разрабатывается в отдельной ветке от `main`.

2. После завершения работы над фичей и успешного тестирования она попадает в ветку `staging` для дальнейшего тестирования.
3. После успешного тестирования в `staging` изменения сливаются в ветку `production` и происходят на продакшн.

Плюсы GitLab Flow:

- Поддержка разных окружений, что позволяет разделить процесс разработки и тестирования.
- Хорошо интегрируется с системами CI/CD, обеспечивая эффективное тестирование перед выпуском на продакшн.

Минусы:

- Более сложен по сравнению с GitHub Flow из-за необходимости управления несколькими окружениями.

Правила командной работы в Git

1. **Создание отдельных веток для задач**: Каждый новый функционал, задача или баг фиксируются в отдельных ветках. Это позволяет работать параллельно без конфликтов.
2. **Чистая история коммитов**: Используйте `git rebase` и интерактивное редактирование коммитов, чтобы история изменений была чистой и понятной. Убедитесь, что каждый коммит добавляет законченную, логичную часть кода.
3. **Частые коммиты и регулярное слияние**: Делайте коммиты регулярно, а не откладывайте на последний момент. Частые маленькие коммиты помогают в случае отката к предыдущим изменениям.
4. **Использование Pull Request (PR)**: Перед слиянием ветки с основной всегда создавайте Pull Request для проведения ревью кода. Это помогает найти потенциальные ошибки до того, как они попадут в `main`.
5. **Открытое обсуждение изменений**: Во время работы с Pull Request важно проводить обсуждения изменений с командой, чтобы все участники были в курсе, что меняется и почему.
6. **Соблюдение правил ветвления**: Важно четко следовать выбранной стратегии работы с ветками (Git Flow, GitHub Flow и т.д.), чтобы не было путаницы и конфликтов.
7. **Автоматизация тестов и деплоя**: Интеграция с системами CI/CD позволяет автоматически тестировать код при слиянии веток и автоматически деплоить изменения в продакшн после успешного прохождения тестов.
8. **Конфликты и их разрешение**: Конфликты при слиянии неизбежны, особенно в крупных проектах. Важно уметь их разрешать, согласовывая изменения с командой, и проверять, что после разрешения конфликта проект работает корректно.