Процесс разработки и интеграции изменений в основную ветку

Процесс разработки и интеграции изменений в основную ветку (`main` или `master`) в GIT обычно включает несколько этапов. Описание ниже основано на стандартном Git Flow, но может адаптироваться под конкретные требования.

---

1. Планирование и создание задачи

Прежде чем приступить к разработке, необходимо:

- Определить задачу (например, новую фичу, исправление бага, рефакторинг).
- Завести тикет в системе управления задачами (Jira, YouTrack, GitLab Issues и т.д.).
- Определить, в какой ветке будет вестись работа.

---

2. Создание новой ветки

Разработчик создает новую ветку на основе последней актуальной версии `main`:

```
git checkout main
git pull origin main
git checkout -b feature/new-feature
```

Где `feature/new-feature` — имя ветки, соответствующее задаче.

Если работа ведется с багфиксом, может использоваться ветка `bugfix/bug-id`.

3. Разработка и коммиты

Разработчик пишет код, делает изменения и фиксирует их:

```
git add .
git commit -m "Добавлена новая функциональность"
```

Рекомендуется делать осмысленные коммиты и использовать семантические сообщения (`fix:`, `feat:`, `refactor:`, `docs:`, `test:`).

---

4. Локальное тестирование

Перед отправкой изменений важно:

- Запустить юнит-тесты.
- Протестировать код в рабочем окружении (локальном, dev).
- Убедиться, что не нарушена работа других модулей.

---

5. Отправка изменений в удаленный репозиторий

После завершения работы над задачей:

```
git push origin feature/new-feature
```

Создается **Merge Request (Pull Request)** в GitLab/GitHub.

---

6. Код-ревью

- Коллеги проверяют код, дают замечания.
- Разработчик вносит правки, дополняет тестами (если нужно).

- Повторяет `git push` до окончательного одобрения.

---

7. Интеграция в основную ветку

После успешного ревью:

- Ветка `feature/new-feature` вливается в `develop` (или сразу в `main`, если процесс без `develop`).
- Используется `merge` или `rebase` (в зависимости от стратегии проекта):

```
git checkout main
git pull origin main
git merge feature/new-feature
git push origin main
```

- В некоторых случаях может использоваться **Squash & Merge** для объединения коммитов.

---

8. Автоматизация через CI/CD

После слияния в `main` могут автоматически запускаться:

- Автоматические тесты.
- Деплой на тестовый или production сервер (в зависимости от настроек CI/CD).

---

9. Удаление ненужных веток

После успешного слияния:

```
git branch -d feature/new-feature
git push origin --delete feature/new-feature
```

Это помогает поддерживать чистоту репозитория.

10. Деплой и релиз

Если проект использует версионирование:

- Генерируется новая версия (`git tag v1.2.3`).
- Запускается CI/CD пайплайн, который деплоит код.

Итог

Этот процесс помогает обеспечить стабильность разработки, контроль качества и чистоту репозитория. В реальных проектах могут быть добавлены дополнительные этапы, например, работа с `hotfix`-ветками, релизными ветками (`release/`), более строгий контроль тестирования и деплоя.

Revision #1
Created 2025-02-05 06:36:21 UTC by Admin
Updated 2025-02-05 06:37:23 UTC by Admin