拥抱 GitFlow，优化开发流程：团队协作的最佳实践

GitFlow工作流

工作流步骤

1. 创建 Feature 分支

从 `main` 分支创建一个新的 `feature` 分支进行开发：

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

2. 开发并测试

在 `feature` 分支上进行开发和测试，确保所有变更在本地通过测试。

3. 合并到 Dev 分支

如果需要在开发环境中测试，将 `feature` 分支合并到 `dev` 分支：

```
git checkout dev
git pull origin dev
git merge --no-ff feature/your-feature
git push origin dev
```

4. 从 Feature 分支合并到 Release 分支

一旦在 `dev` 分支上完成测试且功能稳定，将 `feature` 分支合并到 `release` 分支，而不是将整个 `dev` 分支合并到 `release`：

```
git checkout release
git pull origin release
git merge --no-ff feature/your-feature
git push origin release
```

5. 发布到 Int 环境

在 `release` 分支上进行必要的集成测试，然后在 Int 环境中部署 `release` 分支，并进行更深入的集成测试。

6. 合并到 Main 分支并发布到 Prod 环境

一旦在 Int 环境中测试通过，合并 `release` 分支到 `main` 分支：

```
git checkout main
git pull origin main
git merge --no-ff release
git tag -a v1.0.0 -m "Release version 1.0.0"
git push origin main --tags
```

将 `main` 分支部署到 Prod 环境。

定期同步分支

确保 dev 和 release 分支与 main 分支保持同步，减少未来的冲突。可以定期从 main 分支拉取变更合并到 dev 和 release 分支。

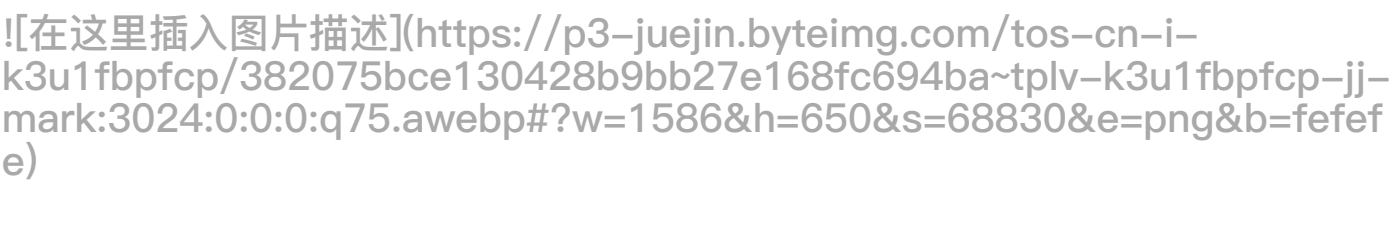
其他注意事项

* **使用 Cherry-Pick**：如果某个功能已经在 `dev` 分支上稳定，但不想合并其他功能，可以使用 `git cherry-pick` 命令将特定提交从 `dev` 分支应用到 `release` 分支。
* **Feature Toggle**：继续使用特性开关，使未完成的功能在合并时保持关闭状态，只有在完成时才启用。
* **CI/CD 流水线**：配置 CI/CD 流水线，自动化测试和部署，确保每次合并都经过严格的测试。
* **代码审查和自动化测试**：通过 PR 或 MR 进行代码审查，并结合自动化测试，确保每次合并的质量。

图示说明

为了更清晰地展示这个流程，这里是一个简化的流程图：

```
main ------> feature/your-feature ------> dev
             |                            |
             v                            v
          release ------------------> main (merge and tag)
```



通过这种方法，你可以确保在 `release` 分支中只包含已经完成并且经过测试的功能，从而避免未完成功能的提前暴露。

代码回滚与撤回策略

准确地回滚特定功能（如 Feature2）的代码，可以使用 Git 的 `revert` 和 `cherry-pick` 功能。下面是详细的步骤，帮助你准确地回滚 Feature2 的代码。

方法1: 使用 Git Revert 回滚特定提交

步骤:

1. **找到 Feature2 的提交记录**:
首先找到所有与 Feature2 相关的提交记录。你可以使用以下命令查看提交历史并找到相关的提交:

```
git log --oneline
```

记录所有与 Feature2 相关的提交哈希（SHA）。
2. **回滚每个提交**:
使用 `git revert` 命令逐个回滚与 Feature2 相关的提交。假设你找到了以下提交哈希:

```
abc1234
def5678
ghi9012
```

你需要依次回滚这些提交:

```
git revert abc1234
git revert def5678
git revert ghi9012
```

3. **合并到 Release 分支**:
将回滚后的代码合并到 `release` 分支:

```
```

```
git checkout release
git pull origin release
git merge --no-ff dev
git push origin release
```

方法2: 使用 Git Revert 回滚整合的 Feature2 分支

如果 Feature2 是从一个单独的分支合并过来的，你可以回滚这个整合提交（merge commit）。

步骤：

1. **找到 Feature2 合并提交**：
使用 `git log --oneline` 找到 Feature2 合并到 `dev` 或 `release` 的提交记录。

```
git log --oneline
```

假设合并提交哈希为 `jkl3456`。
2. **回滚合并提交**：
使用 `git revert -m 1` 回滚合并提交：

```
git revert -m 1 jkl3456
```

这里的 `-m 1` 表示回滚第一个父提交，这通常是目标分支（如 `dev`）。
3. **合并到 Release 分支**：
将回滚后的代码合并到 `release` 分支：

```
git checkout release
```

```
git pull origin release
git merge --no-ff dev
git push origin release
```

方法3: 创建新的临时分支并排除 Feature2

如果回滚操作过于复杂，可以创建一个新的临时分支，并选择性地合并其他功能，排除 Feature2。

步骤：

1. **创建新的临时分支**：
从 `release` 分支创建一个新的临时分支：

```
git checkout release
git pull origin release
git checkout -b temp-release
```

2. **选择性合并 Feature1**：
只合并 Feature1 的分支或提交：

```
git cherry-pick <feature1-commit1>
git cherry-pick <feature1-commit2>
# 继续 cherry-pick 其他相关提交
```

3. **强制更新 Release 分支**：
将临时分支的内容强制推送到 `release` 分支：

```
git checkout release
git reset --hard temp-release
git push origin release --force
```

```

### 小心事项

* **备份代码**：在进行回滚或强制更新操作前，确保备份代码或创建新的分支保留当前状态。
* **沟通和协作**：与团队成员沟通回滚计划，确保所有人都清楚变更内容。
* **自动化测试**：在回滚和合并操作后，运行所有自动化测试，确保没有引入新的问题。

通过以上方法，你可以准确地回滚 Feature2 的代码，确保发布的版本只包含需要发布的功能。

原文链接: https://juejin.cn/post/7379147394754969640
```