# 记一次不规范代码开发引发的坑...

事情要从一次不规范的代码开发开始说起

## 背景故事

> **时间**

2024年某个风平浪静的周五晚上

> **地点**

中国，北京，西二旗，某互联网大厂会议室

> **人物**

小杰，小A，小B，老K

> **对话**

老K：昨天提交的**代码被测试打回**来了！为什么**小B没开发完的内容**也一起提交上去了？

小B：啊？我不清楚啊，我在开发分支B开发完一部分就**提交到test分支进行联调**了啊

小A：额（”!），我把**test分支合并到发布分支提交给测试**了，因为我跟小杰最开始在各自的分支开发，但是中间联调的时候，为了**图方便直接在test分

支上改**，改来改去就直接**在test分支上开发**了。。。

老K：什么？你怎么能直接在test分支开发?

老K：正确的开发流程规范应该是：现在**各自的开发分支**上开发，然和合并到**test分支上进行联调**，联调没有问题在**提交发布release分支进行测试和部署**，验证没问题在把各自的**开发分支合并到基线分支master**

老K：现在要把代码回滚，谁做的事情谁负责。小杰，你去把test分支上的代码抽出来放到单独一个开发分支上

小杰：啊?

---

小杰接受这个任务，准备把test分支上**他跟小A多次提交**的内容跟转移到一个纯净的开发分支，小杰决定使用`git cherry pick` 这个命令

git cherry-pick
---

### 介绍

`git cherry-pick` 是 Git 中的一个非常有用的命令，它允许你从一个分支中选择特定的提交（commit）应用到当前的分支。这个命令在需要引入某些特定功能或修复而不想进行完整的分支合并时特别有用。

### 使用示例

假设你有以下 Git 分支结构：

```
* 5a3d5f2 (feature) Add new feature
```

```
* c7e33a5 Fix bug B
* 1a2b3c4 Fix bug A
* 9d8e7f6 (main) Initial commit
```

现在你在 `main` 分支上，想要将 `feature` 分支中修复 bug A 的提交（`1a2b3c4`）引入当前分支。你可以这样做：

1. 切换到目标分支（假设是 `main` 分支）：

```
git checkout main
```

2. 使用 `git cherry-pick` 命令：

```
git cherry-pick 1a2b3c4
```

执行上述命令后，提交 `1a2b3c4` 的更改会被应用到 `main` 分支上。
3. **使用范围（range）来批量 cherry-pick**. 假设你要 cherry-pick 从 commitA 到 commitB 之间的所有 commit（包含 commitA 但不包含 commitB），你可以使用以下命令：

```
git cherry-pick commitA^..commitB
```

4. **使用多个单独的 commit 来批量 cherry-pick** . 假设你有一系列的 commit 哈希 commit1, commit2, commit3，你可以使用以下命令：

```
git cherry-pick commit1 commit2 commit3
```

5. **解决可能的冲突：**在 cherry-pick 的过程中，如果遇到冲突，Git

会提示你。你需要手动解决这些冲突并继续 cherry-pick

```
# 解决冲突后，添加解决后的文件
git add <conflicted-file>
# 继续 cherry-pick
git cherry-pick --continue
```

### 注意事项

1. **冲突处理**：如果在 `cherry-pick` 的过程中，存在文件冲突，Git 会暂停操作，并提示冲突文件。你需要手动解决这些冲突，然后使用 `git add <file>` 添加解决后的文件，最后运行 `git cherry-pick --continue` 继续操作。如果你想中止 `cherry-pick`，可以使用 `git cherry-pick --abort`。
2. **保持提交历史干净**：频繁使用 `cherry-pick` 可能会导致提交历史变得复杂。在使用前，评估是否可以通过别的操作（如合并或重置）来实现相同的目标。
3. **避免重复提交**：如果你已经 `cherry-pick` 了一个提交，再次尝试 `cherry-pick` 同一个提交可能会引发问题。Git 会提示你已经包含了相同的更改。
4. **顺序和依赖关系**：如果一个提交依赖于之前的其他提交，`cherry-pick` 这些提交时需要注意顺序，以避免破坏代码的完整性。

---

解决方案
---

### 第一次尝试

```
# Step 1: 创建新的分支 xsj_0701
git checkout master
git pull origin master
git checkout -b xsj_0701
```

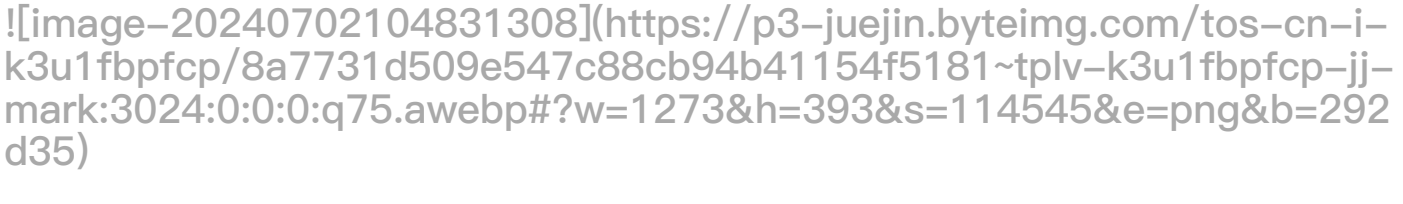
```
# Step 2: 查看 stable_test 分支的 commit 历史
git log stable_test

# Step 3: 批量 cherry-pick commit
git checkout xsj_0701
git cherry-pick commitA^..commitB  # 使用范围
# 或者
git cherry-pick commit1 commit2 commit3  # 使用多个单独的 commit 哈希

# Step 4: 解决可能的冲突
# 解决冲突后
git add <conflicted-file>
git cherry-pick --continue

# Step 5: 推送新的分支
git push origin xsj_0701
```

小杰使用批量范围cherry-pick，这个范围大约包含了10个commit，正当小杰吭哧吭哧的解决几个冲突之后，cherry-pick突然停止，没有冲突，查看当前commit也只到`add cache`这个提交这里，如下图所示




> 为什么cherry-pick会停止呢?

小杰经过观察发现，停止的位置是merge节点

当你尝试 cherry-pick 一个 merge commit 时，Git 需要更多信息来决定如何处理合并。默认情况下，Git 不会自动 cherry-pick merge commit，因为它无法确定你想要保留哪个分支的变更。

要解决这个问题，你可以使用以下方法：

**方法 1: 跳过 merge commit**

如果你不需要 cherry-pick 这个 merge commit，可以手动跳过它。你可以通过在失败后继续 cherry-pick 后续的 commit 来实现：

在发生停止后，手动跳过 merge commit 并继续 cherry-pick 后续的 commit:

```
git cherry-pick --skip
# 然后继续 cherry-pick 后续的 commit
git cherry-pick <remaining-commits>
```

**方法 2: 使用 cherry-pick -m 选项**

如果你确实需要 cherry-pick 这个 merge commit，可以使用 `-m` 选项。`-m` 选项需要一个参数来指定父提交的索引，通常使用 `1` 表示第一父提交。

* **继续 cherry-pick merge commit 并指定父提交索引:**

```
git cherry-pick -m 1 5b30dd90
```

* **继续后续的 cherry-pick:**

```
git cherry-pick <remaining-commits>
```

### 第二次尝试

```
# Step 1: 创建新的分支 xsj_0701
```

```
git checkout master
git pull origin master
git checkout -b xsj_0701

# Step 2: 查看 stable_test 分支的 commit 历史
git log stable_test

# Step 3: 批量 cherry-pick commit
git checkout xsj_0701
git cherry-pick commitA^..commitB  # 使用范围
# 或者
git cherry-pick commit1 commit2 commit3  # 使用多个单独的 commit 哈希

# Step 4: 解决可能的冲突
# 解决冲突后
git add <conflicted-file>
git cherry-pick --continue

# Step 5: 跳过merge节点
git cherry-pick --skip

# Step 6: 推送新的分支
git push origin xsj_0701
```

经过一下午的奋战，小杰终于把test分支上的开发内容都迁移到纯净开发分支，然后屁颠屁颠去跟老K汇报了

原文链接: https://juejin.cn/post/7387420922808713235