Git Flow

#git #gitflow #workflow

What's Git Flow

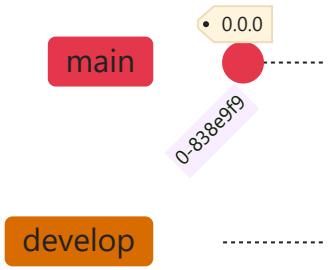
- 一種工作流程 (Workflow)，2010 年發布沿用至今
- 一種分支為主的模型，分支個別有其屬性與任務
- 以 `merge` 為基礎，不適用 `rebase` 與 `cherry-pick`
- CLI 與多種 GUI 皆有支援 (有些要安裝套件)

初始化

```
git flow init
```

- 初始化 GitFlow 環境
- 過程會詢問分支名稱，可依照需求調整
- 主要分支預設名稱如為 `master`，可改為 `main`
- 執行完畢會建立並切換到 `develop`，此時 `develop` = `main`
- 上述指令約略等同以下指令執行結果

```
git init
git branch develop
git switch develop
```



> **Info**
>
> - Git 版本 `2.23` 之後，支援指令 `switch`，等同 `chekout` 但語意可能更直覺
> - `git flow` 應該可理解為 git 指令集

分支

類型	說明	常見名稱(或前綴)
常駐分支	一直存在 repo	`main`, `develeop`
臨時分支	通常於使用完畢刪除	`feature/xxx`, `release/xxx`, `hotfix/xxx`

> **Info**
>
> - 除了主要分支 `main`，其他分支(或前綴)可改用縮寫，例如
> - `develop` 可改為 `dev`
> - `feature/xxx`, `release/xxx`, `hotfix/xxx` 可改為 `f/xxx`, `r/xxx`, `h/xxx`

常駐分支

名稱	說明	commit	merge	tag
`main`	主要分支，每個節點都可以發布到正式環境	❌	✓	✓
`develop`	開發分支，每次開發功能都應該從這裡開始	❌	✓	

臨時分支(前綴)

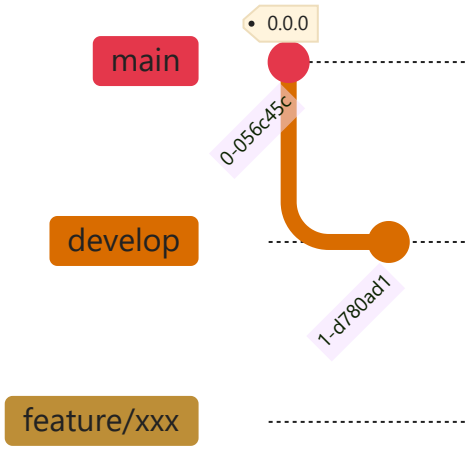
名稱	說明	from	to	tag
`feature/`	(開發)功能分支，每次開發功能時開立	`develop`	`develop`	
`release/`	發布版本分支，每次部署時開立，標註版號	`develop`	`develop`, `main`	
`hotfix/`	熱修分支，上線後處理異常時開立	`main`	`main`, `develop`	✓

功能(feature)開發

每當有功能需要開發時，就建立功能分支

建立功能分支

```
git flow feature start xxx
```



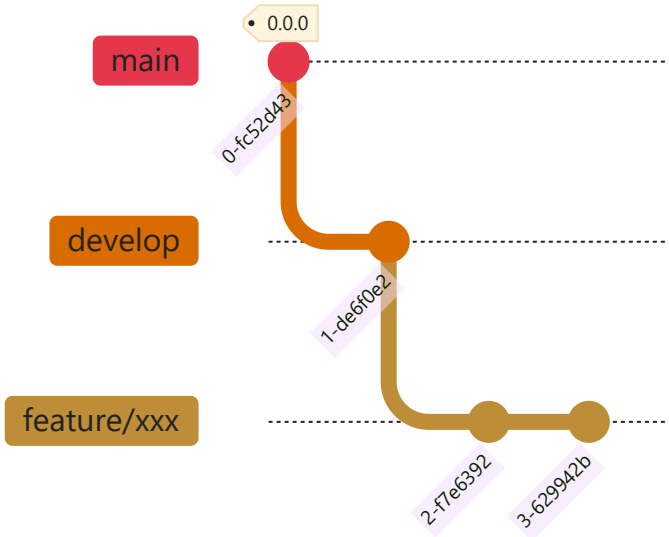
等同以下指令

```
git switch develop
git branch feature/xxx
git switch feature/xxx
```

進行功能開發

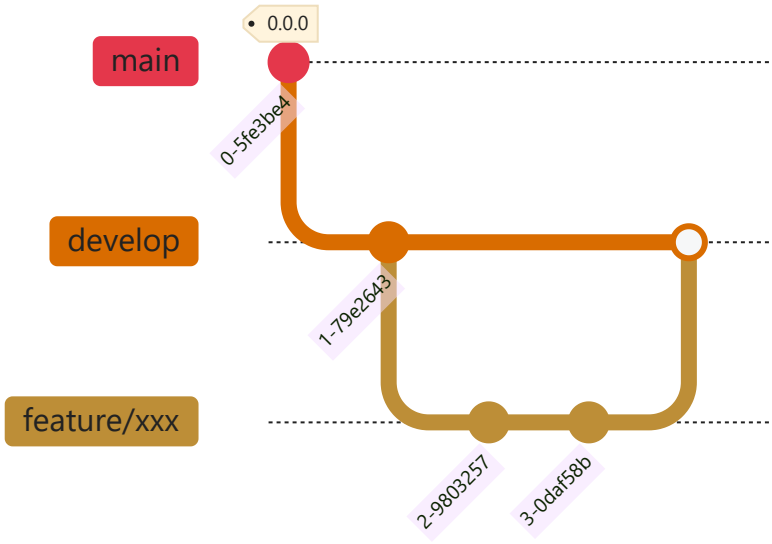
- 開始開發功能，撰寫程式碼
- 開發完提交(`commit`)修改，也就是

```
git add .
git commit -m "finish xxx"
```



完成功能開發

```
git flow feature finish xxx
```



等同以下指令

```
git switch develop
git merge feature/xxx
git branch -d xxx
```

> **Info**
>
> - 功能完成之後，GitFlow 預設刪除功能分支

如果功能需要協作

發佈到 `remote`

```
git flow feature publish xxx
```

取得功能分支

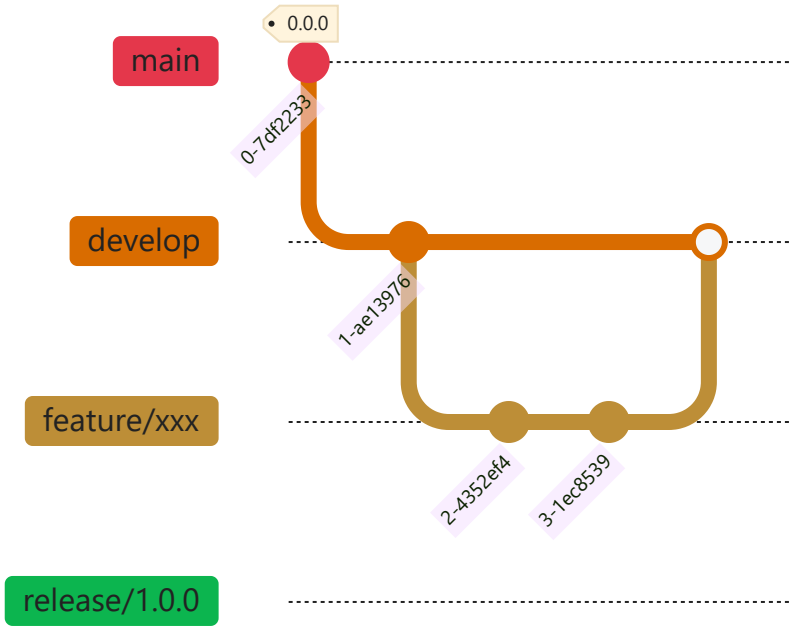
```
git flow feature pull xxx
```

部署版本

累積一定程度功能修改之後，發行 UAT 版本

建立發行分支

```
git flow release start 1.0.0
```

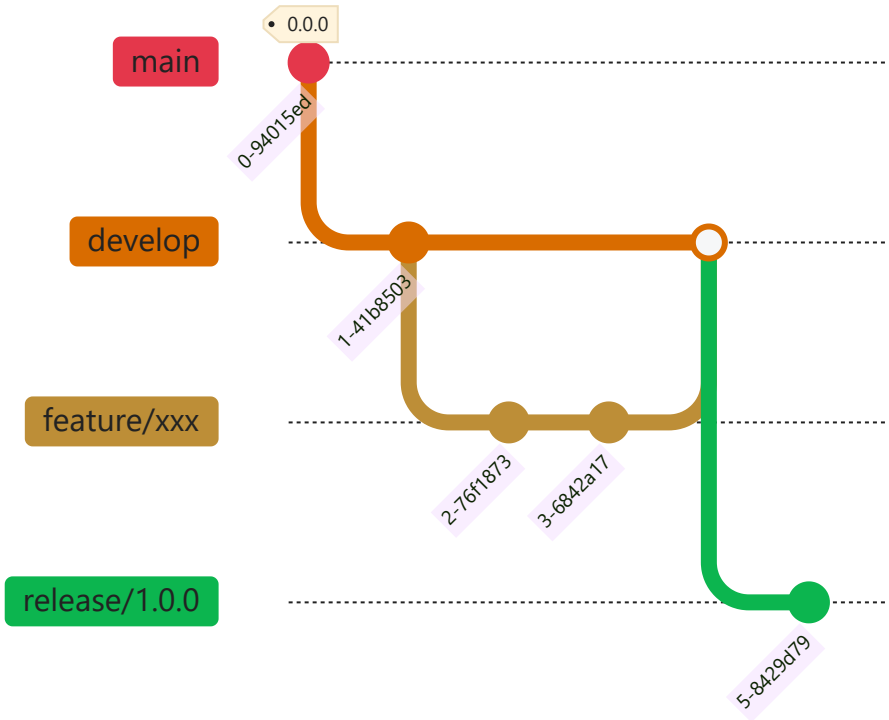


等同

```
git switch develop
git branch release/1.0.0
git switch release/1.0.0
```

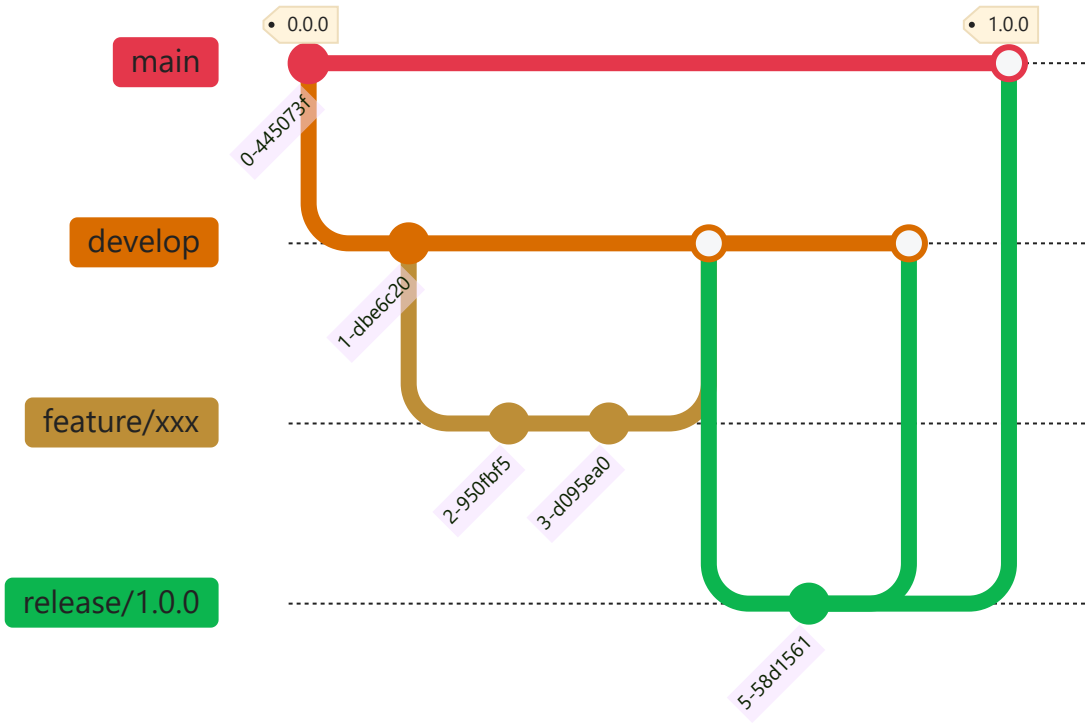
進行發行作業

- 修改版本編號與相關發行必要參數，提交(commit)修改內容
- 部署到測試環境(UAT)，通知相關單位測試等等



完成發行作業

```
git flow release finish 1.0.0
```



等同

```
# release merge into main
git switch main
git merge release/1.0.0
git tag -a 1.0.0 -m "merge 1.0.0"
# release merge into develop
git switch develop
git merge release/1.0.0
# delete release
git branch -d release/1.0.0
```

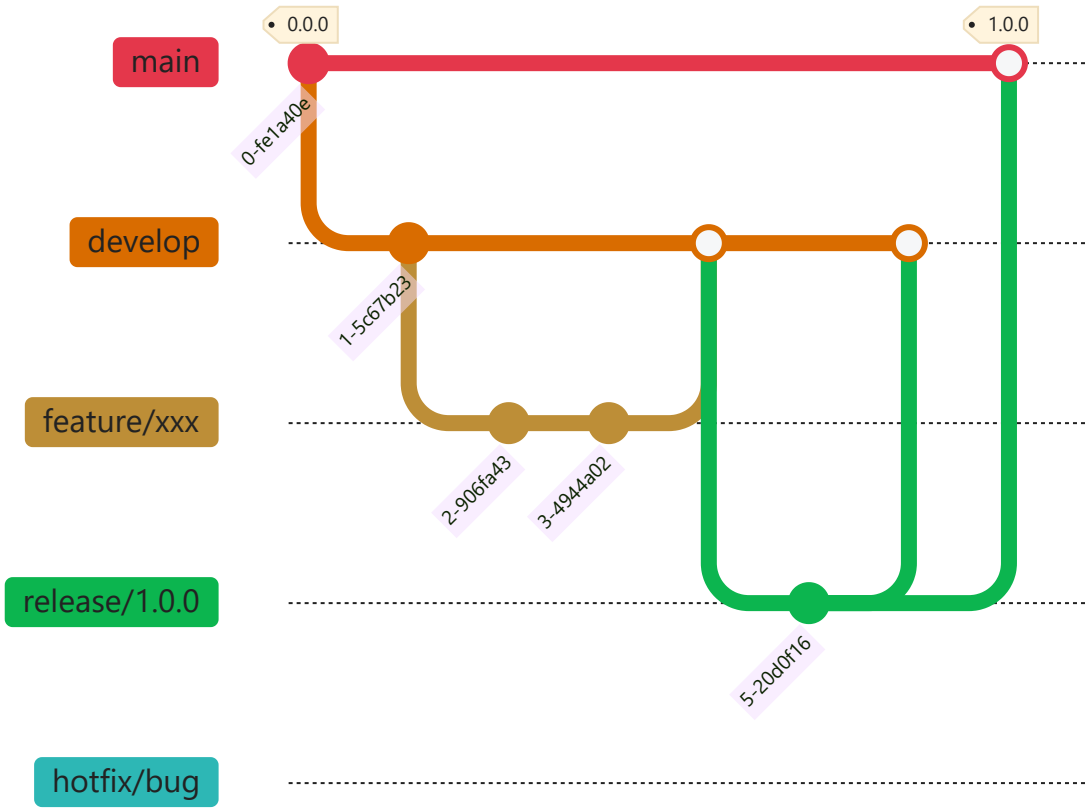
> **Info**
>
> - 發行內容需同時合併回 `develop` 與 `main`
>   - 合併回 `develop`，開發環境才有剛剛發行修改內容
>   - 合併回 `main`，正是環境才有上述內容，屆時部署才會是正確版本
> - 通常會在 `main` 標註版本 (使用 `tag`)
> - 發行完成之後，GitFlow 預設刪除發行分支

快速修復 (Hotfix)

部署至 PRD 環境後發現問題，進行熱修

建立熱修分支

```
git flow hotfix start bug
```

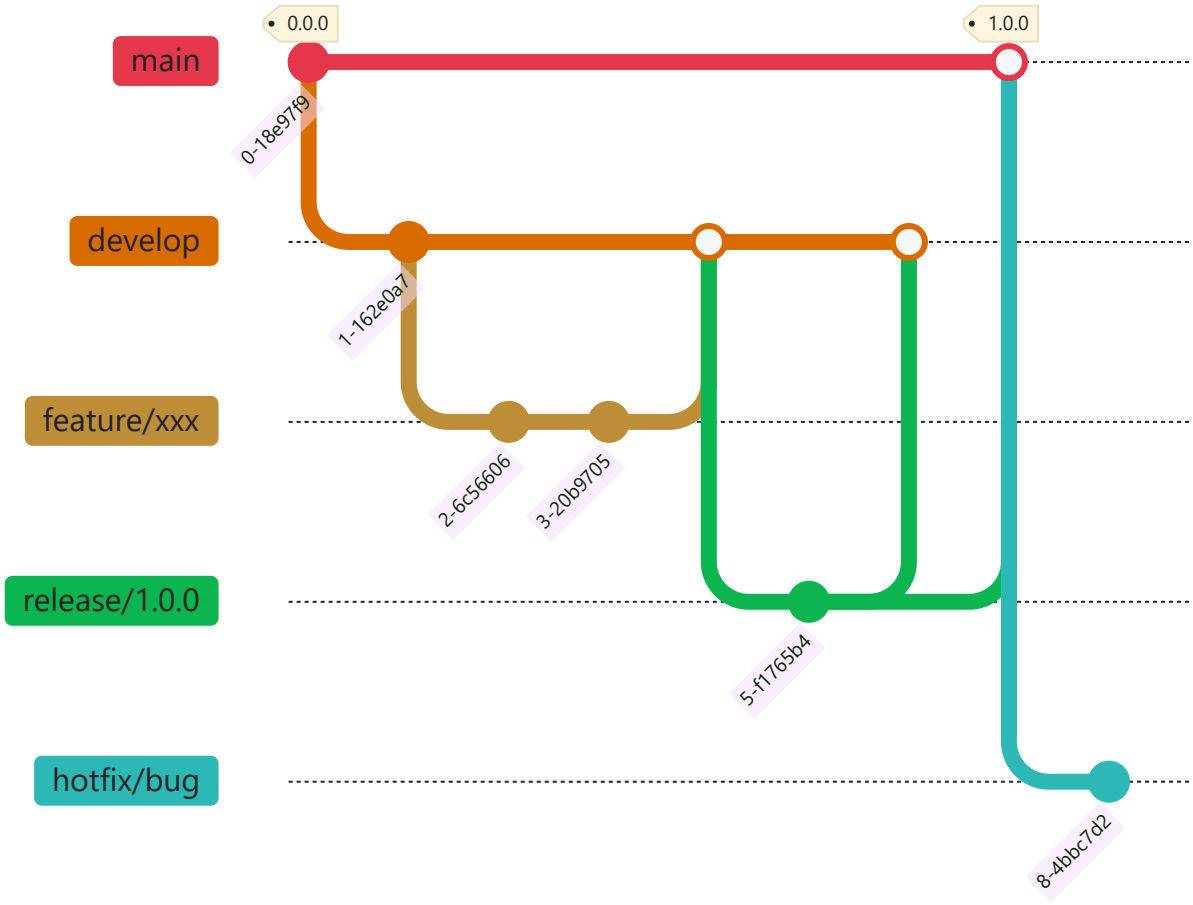


等同

```
git switch main
git branch hotfix/bug
git switch hotfix/bug
```

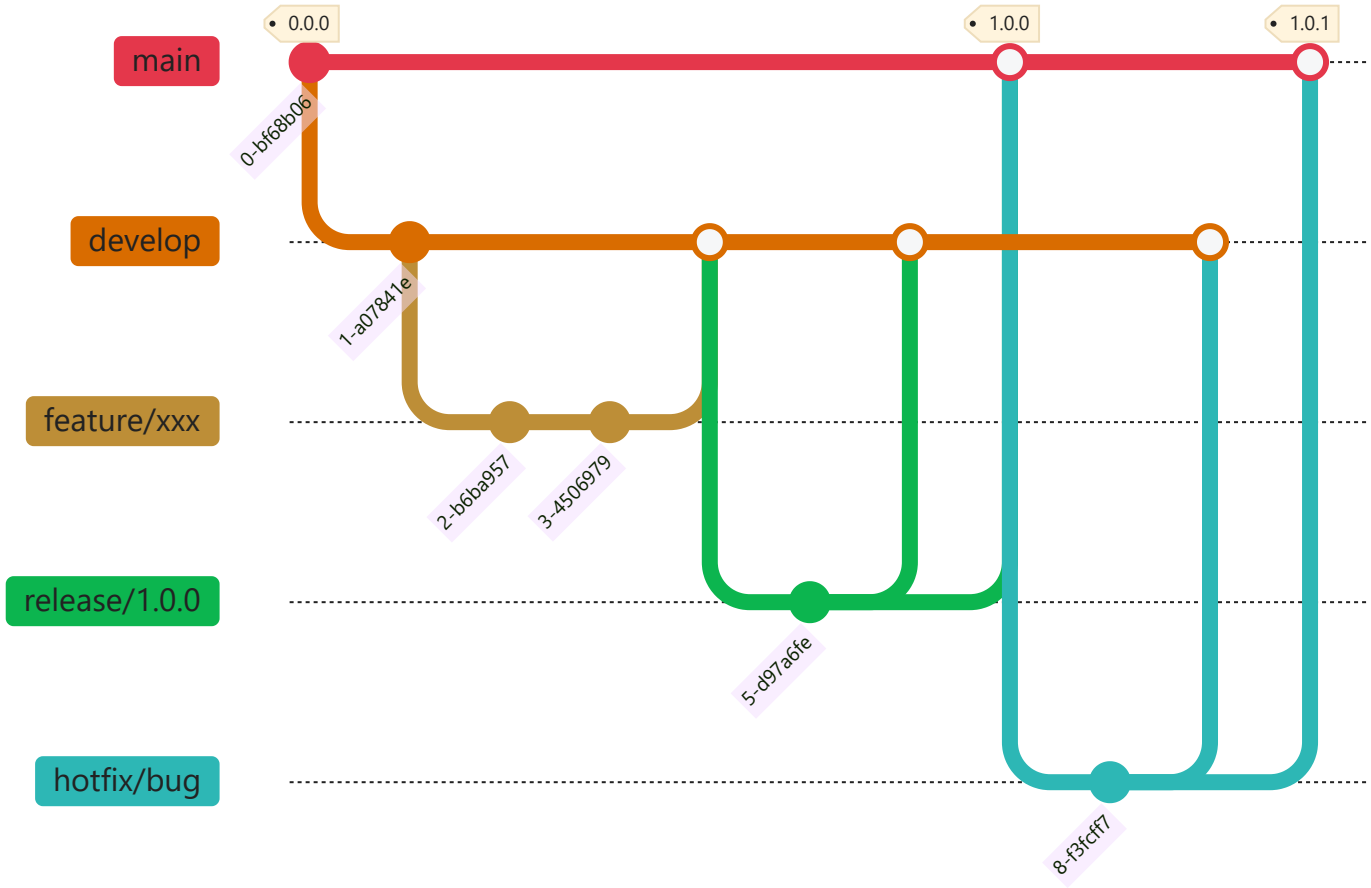
進行程式修改

修改程式，提交修改



完成熱修作業

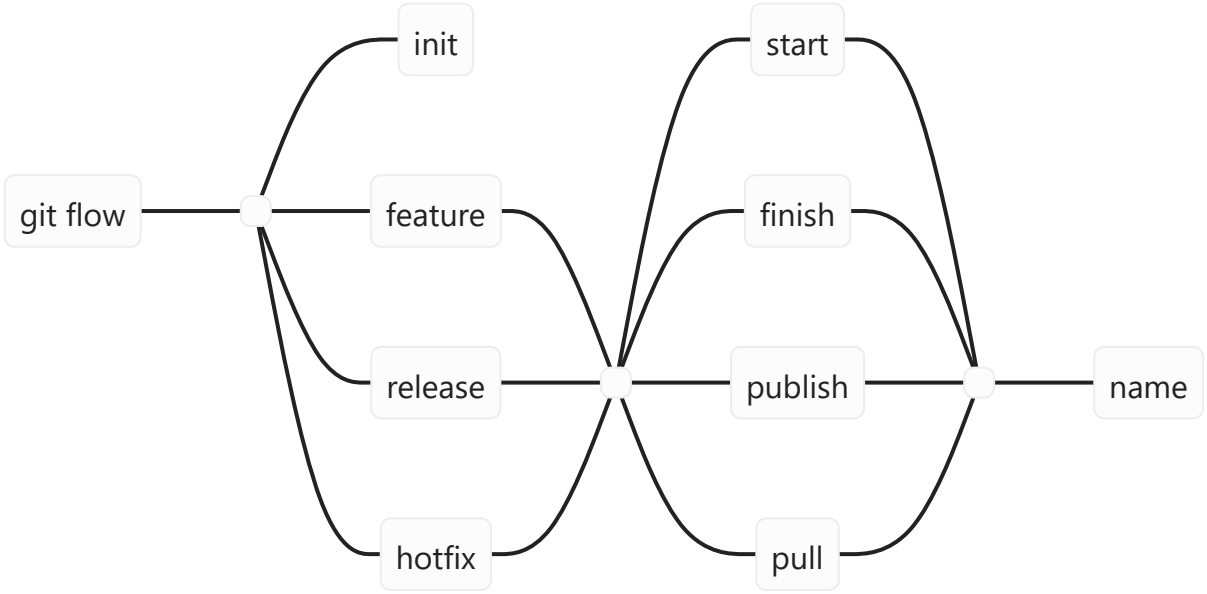
```
git flow hotfix finish bug
```



等同

```
# hotfix merge into main
git switch main
git merge hotfix/bug
git tag -a v1.0.2 -m "bug fixed"
# hotfix merge into develop
git switch develop
git merge hotfix/bug
# delete hotfix
git branch -d hotfix/bug
```

指令



參考資料

https://danielkummer.github.io/git-flow-cheatsheet/index.zh_TW.html#features