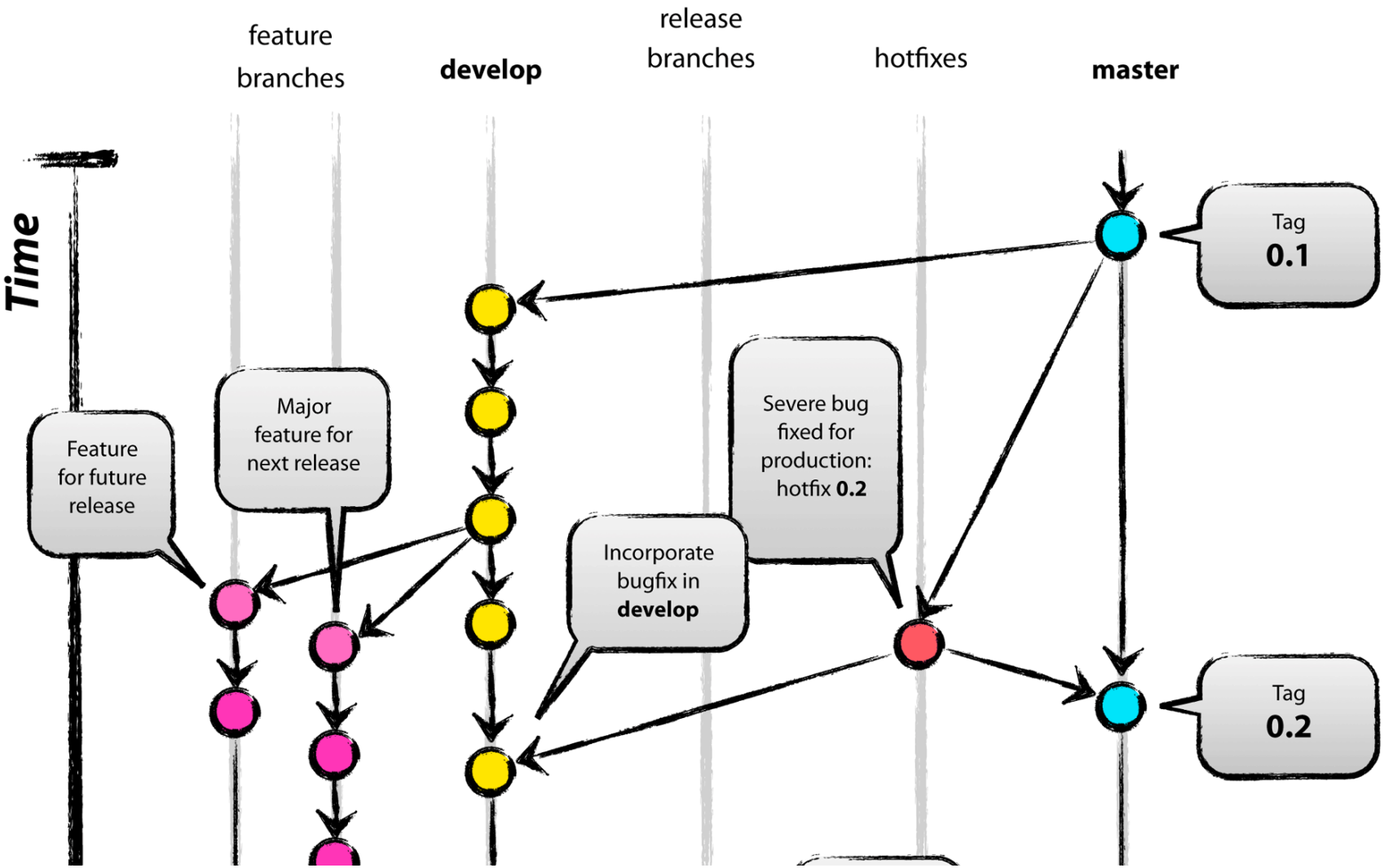
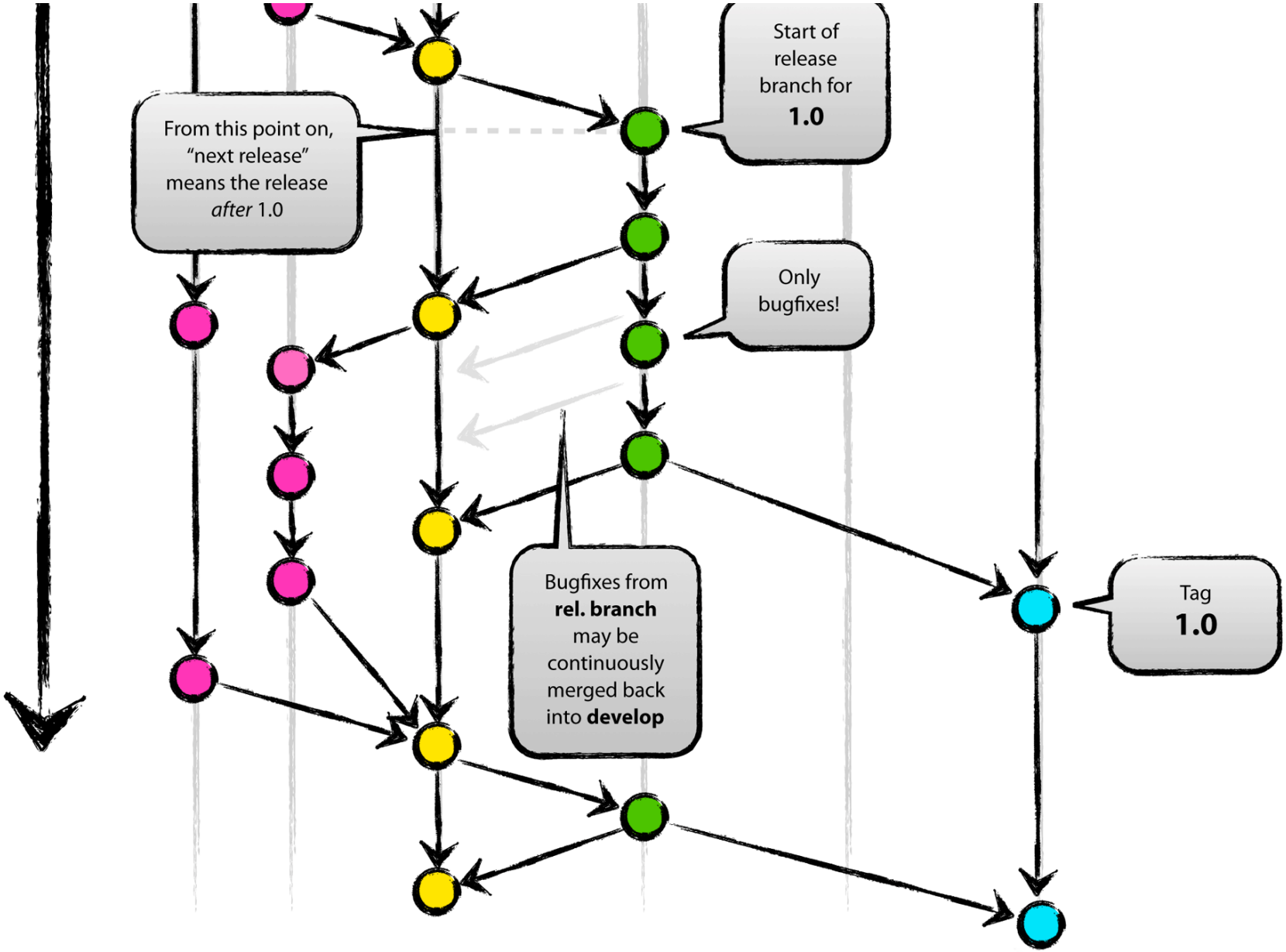
Git Flow

就像代码需要代码规范一样，代码管理同样需要一个清晰的流程和规范

Vincent Driessen 同学为了解决这个问题提出了 A Successful Git Branching Model

下面是Git Flow的流程图





上面的图你理解不了？ 没关系，这不是你的错，我觉得这张图本身有点问题，这张图应该左转90度，大家应该就很用以理解了。

Git Flow常用的分支

- Production 分支

也就是我们经常使用的Master分支，这个分支最近发布到生产环境的代码，最近发布的Release， 这个分支只能从其他分支合并，不能在这个分支直接修改

- Develop 分支

这个分支是我们是我们的主开发分支，包含所有要发布到下一个Release的代码，这个主要合并与其他分支，比如Feature分支

- Feature 分支

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回Develop分支进入下一个Release

- Release分支

当你需要一个发布一个新Release的时候，我们基于Develop分支创建一个Release分支，完成Release后，我们合并到Master和Develop分支

- Hotfix分支

当我们在Production发现新的Bug时候，我们需要创建一个Hotfix, 完成Hotfix后，我们合并回Master和Develop分支，所以Hotfix的改动会进入下一个Release

Git Flow如何工作

初始分支

所有在Master分支上的Commit应该Tag

Feature 分支

分支名 feature/*

Feature分支做完后，必须合并回Develop分支, 合并完分支后一般会删点这个Feature分支，但是我们也可以保留

Release分支

分支名 release/*

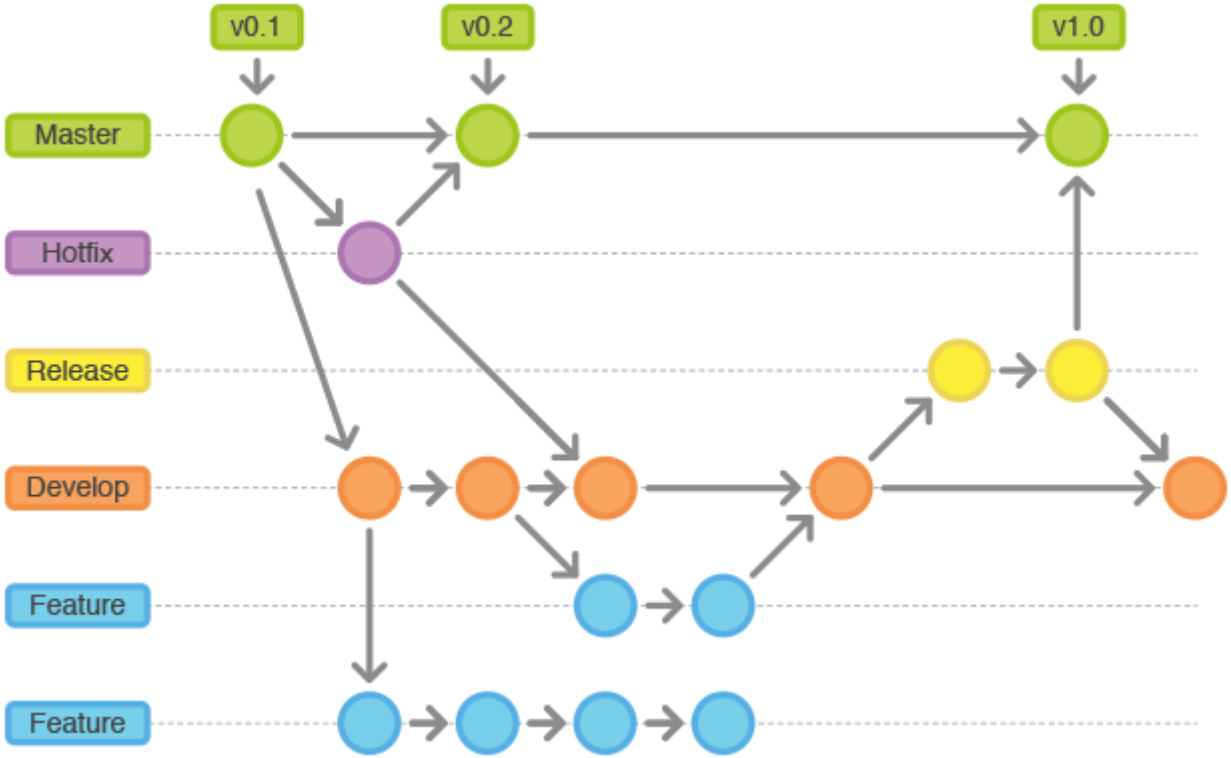
Release分支基于Develop分支创建，打完Release分之后，我们可以在这个Release分支上测试，修改Bug等。同时，其它开发人员可以基于开发新的Feature (记住：一旦打了Release分支之后不要从Develop分支上合并新的改动到Release分支)

发布Release分支时，合并Release到Master和Develop， 同时在Master分支上打个Tag记住Release版本号，然后可以删除Release分支了。

维护分支 Hotfix

分支名 hotfix/*

hotfix分支基于Master分支创建，开发完后需要合并回Master和Develop分支，同时在Master上打一个tag



Git Flow代码示例

a. 创建develop分支

```
git branch develop
git push -u origin develop
```

b. 开始新Feature开发

```
git checkout -b some-feature develop
# Optionally, push branch to origin:
git push -u origin some-feature
```

```
# 做一些改动
git status
git add some-file
git commit
```

c. 完成Feature

```
git pull origin develop
git checkout develop
git merge --no-ff some-feature
git push origin develop

git branch -d some-feature

# If you pushed branch to origin:
git push origin --delete some-feature
```

d. 开始Relase

```
git checkout -b release-0.1.0 develop

# Optional: Bump version number, commit
# Prepare release, commit
```

e. 完成Release

```
git checkout master
git merge --no-ff release-0.1.0
```

```
git push

git checkout develop
git merge --no-ff release-0.1.0
git push

git branch -d release-0.1.0

# If you pushed branch to origin:
git push origin --delete release-0.1.0


git tag -a v0.1.0 master
git push --tags
```

f. 开始Hotfix

```
git checkout -b hotfix-0.1.1 master
```

g. 完成Hotfix

```
git checkout master
git merge --no-ff hotfix-0.1.1
git push


git checkout develop
```

```
git merge --no-ff hotfix-0.1.1
git push

git branch -d hotfix-0.1.1

git tag -a v0.1.1 master
git push --tags
```

Git flow工具

实际上，当你理解了上面的流程后，你完全不用使用工具，但是实际上我们大部分人很多命令就是记不住呀，流程就是记不住呀，肿么办呢？

总有聪明的人创造好的工具给大家用, 那就是Git flow script.

安装

- OS X

brew install git-flow

- Linux

apt-get install git-flow

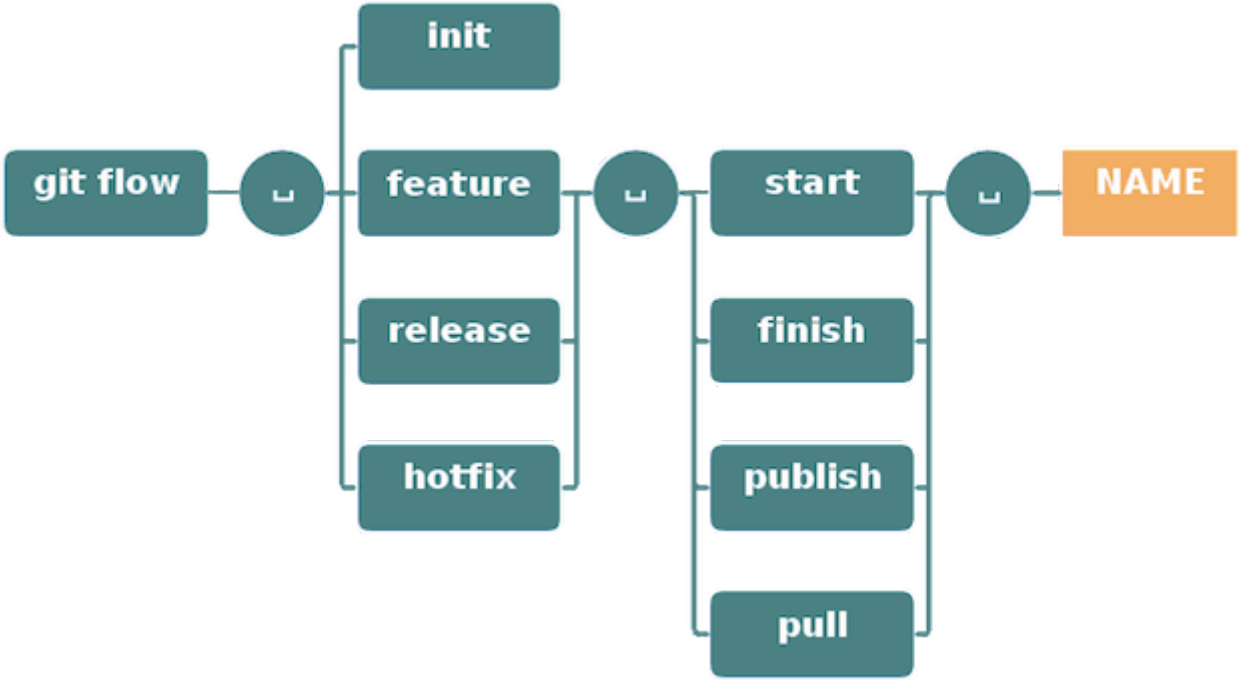
- Windows

wget -q -O - --no-check-certificate https://github.com/nvie/gitflow/raw/develop/contrib/gitflow-installer.sh | bash

使用

- **初始化:** git flow init

- **开始新Feature:** git flow feature start MYFEATURE
- **Publish一个Feature(也就是push到远程):** git flow feature publish MYFEATURE
- **获取Publish的Feature:** git flow feature pull origin MYFEATURE
- **完成一个Feature:** git flow feature finish MYFEATURE
- **开始一个Release:** git flow release start RELEASE [BASE]
- **Publish一个Release:** git flow release publish RELEASE
- **发布Release:** git flow release finish RELEASE
  别忘了git push --tags
- **开始一个Hotfix:** git flow hotfix start VERSION [BASENAME]
- **发布一个Hotfix:** git flow hotfix finish VERSION



Git Flow GUI

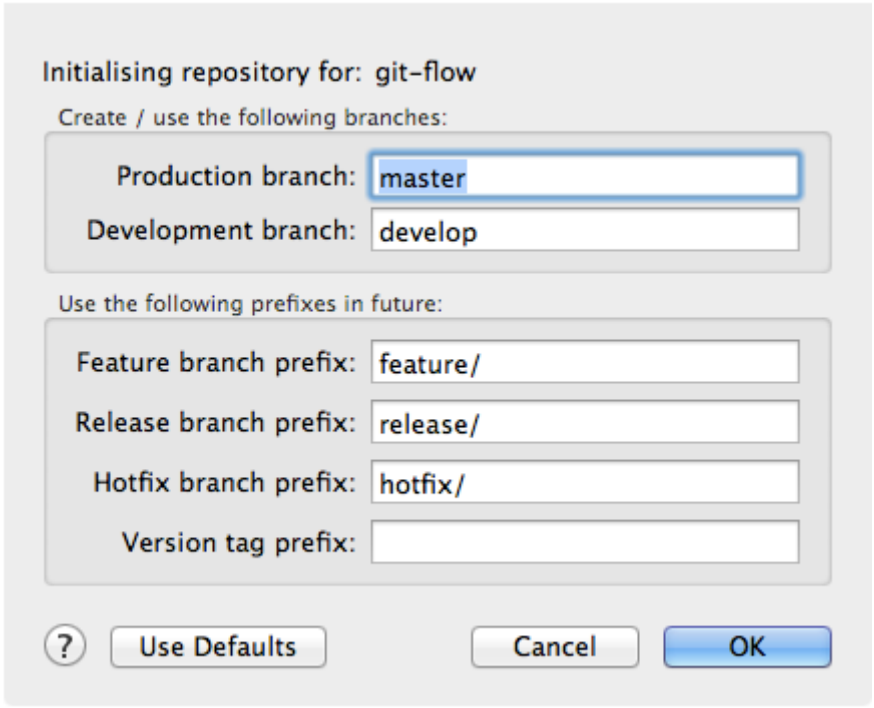
上面讲了这么多，我知道还有人记不住，那么又有人做出了GUI 工具，你只需要点击下一步就行，工具帮你干这些事！！！

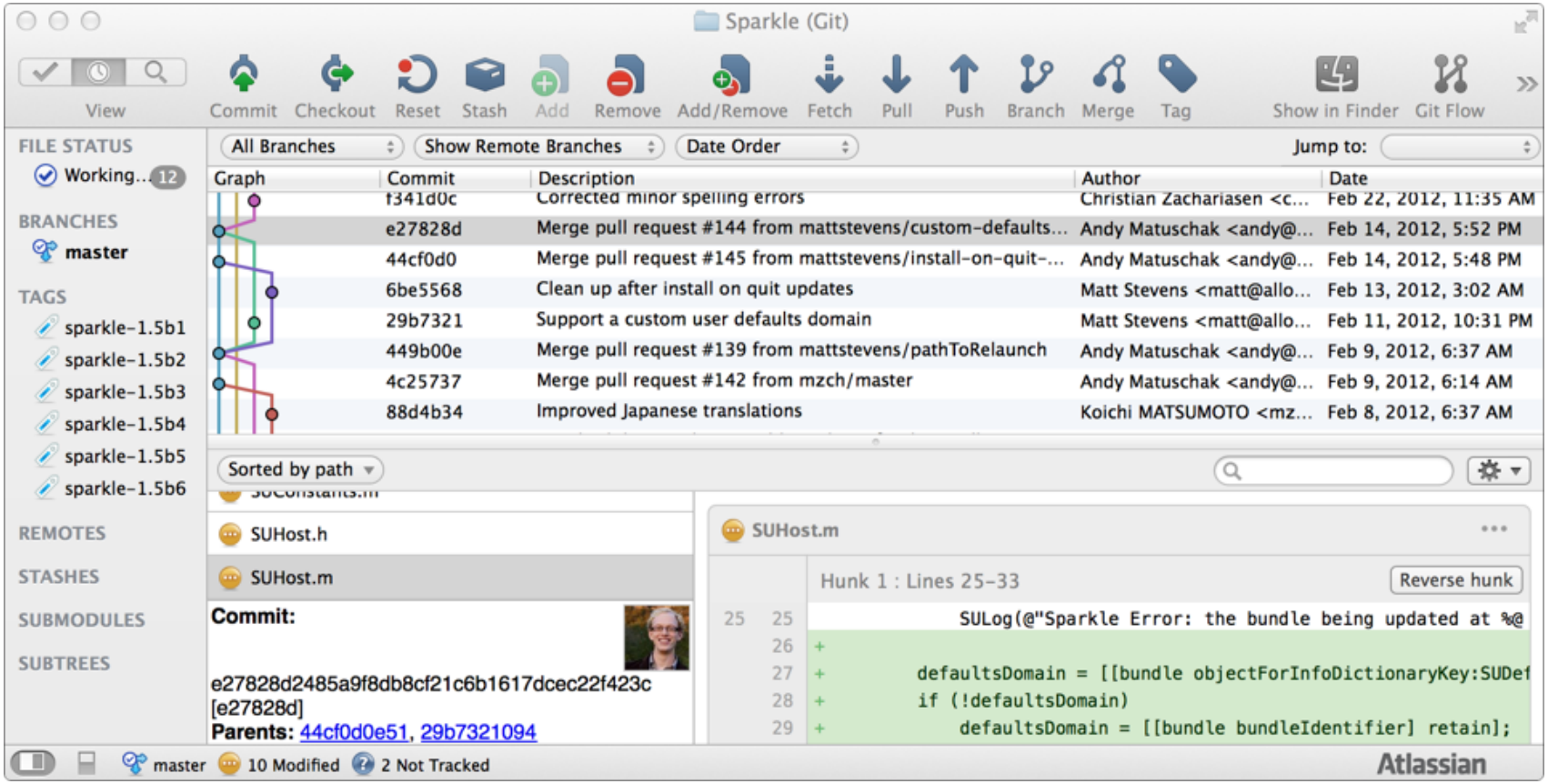
SourceTree

当你用Git-flow初始化后，基本上你只需要点击git flow菜单选择start feature, release或者hotfix, 做完后再次选择git flow菜单，点击Done Action. 我勒个去，我实在想不到还有比这更简单的了。

目前SourceTree支持Mac, Windows, Linux.

这么好的工具请问多少钱呢？ **免费!!!!**

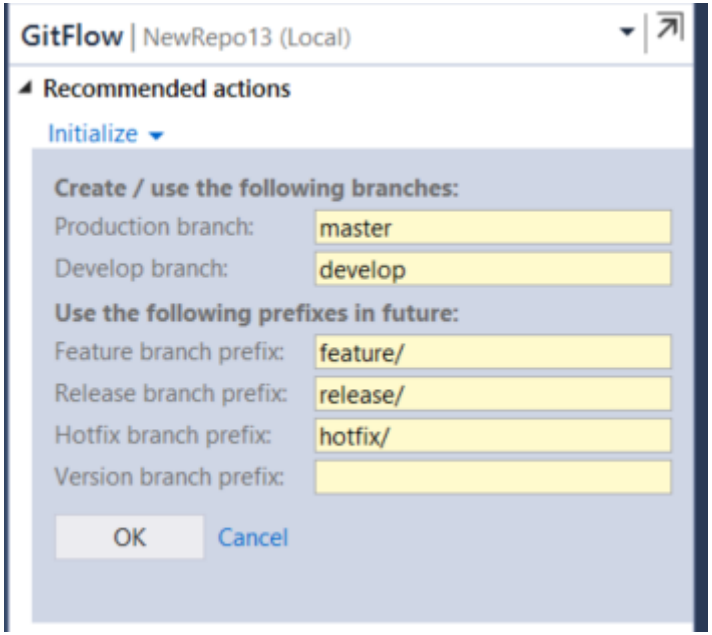

Initialising repository for: git-flow
Create / use the following branches:
Production branch: master
Development branch: develop
Use the following prefixes in future:
Feature branch prefix: feature/
Release branch prefix: release/
Hotfix branch prefix: hotfix/
Version tag prefix:
?
Use Defaults
Cancel
OK

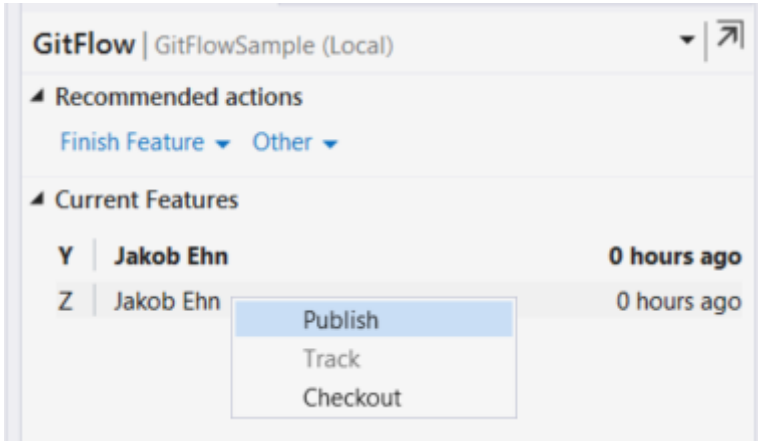


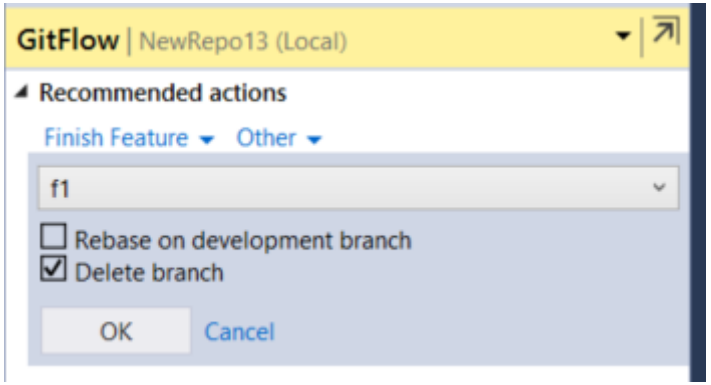
Git flow for visual studio

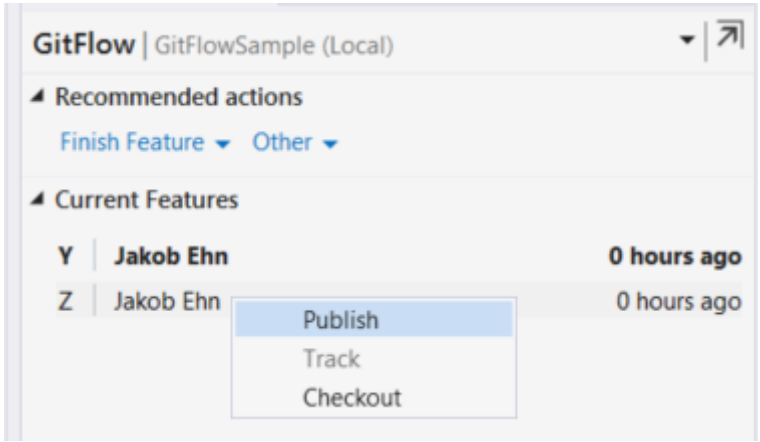
广大VS的福音

[GitFlow for Visual Studio]


GitFlow | NewRepo13 (Local)
Recommended actions
Initialize
Create / use the following branches:
Production branch:
master
Develop branch:
develop
Use the following prefixes in future:
Feature branch prefix:
feature/
Release branch prefix:
release/
Hotfix branch prefix:
hotfix/
Version branch prefix:
OK
Cancel


GitFlow | GitFlowSample (Local)
Recommended actions
Finish Feature
Other
Current Features
Y
Jakob Ehn
0 hours ago
Z
Jakob Ehn
0 hours ago
Publish
Track
Checkout


GitFlow | NewRepo13 (Local)
Recommended actions
Finish Feature
Other
f1
Rebase on development branch
Delete branch
OK
Cancel


GitFlow | GitFlowSample (Local)
Recommended actions
Finish Feature
Other
Current Features
Y
Jakob Ehn
0 hours ago
Z
Jakob Ehn
0 hours ago
Publish
Track
Checkout