Utilizando o fluxo Git Flow

Mikael Hadler Follow
Jan 11, 2018 · 4 min read

Este artigo tem o intuito de expor uma abordagem de como trabalhar utilizando o fluxo git flow.

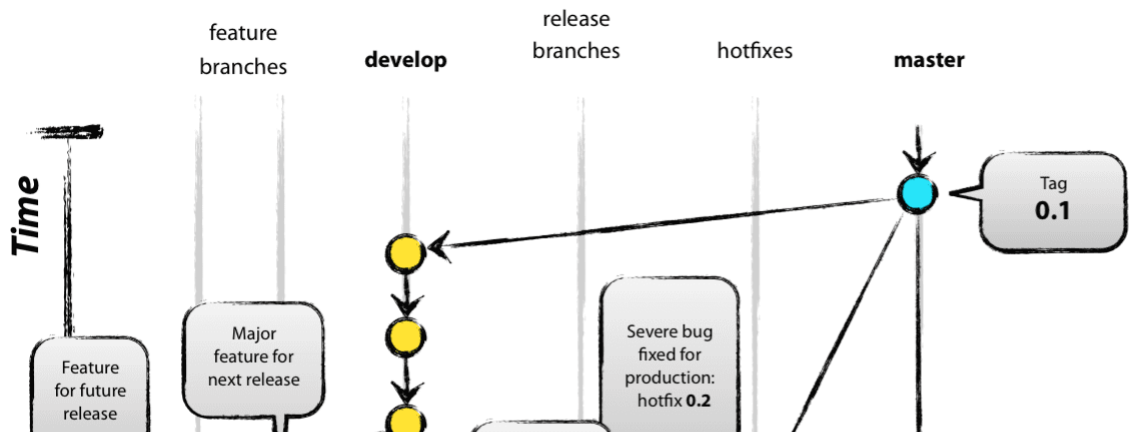
Se você já trabalha com o git como principal ferramenta de controle de versão, já deve ter visto várias abordagens de como utilizar e controlar branchs em um cenário de produção ou pessoal.

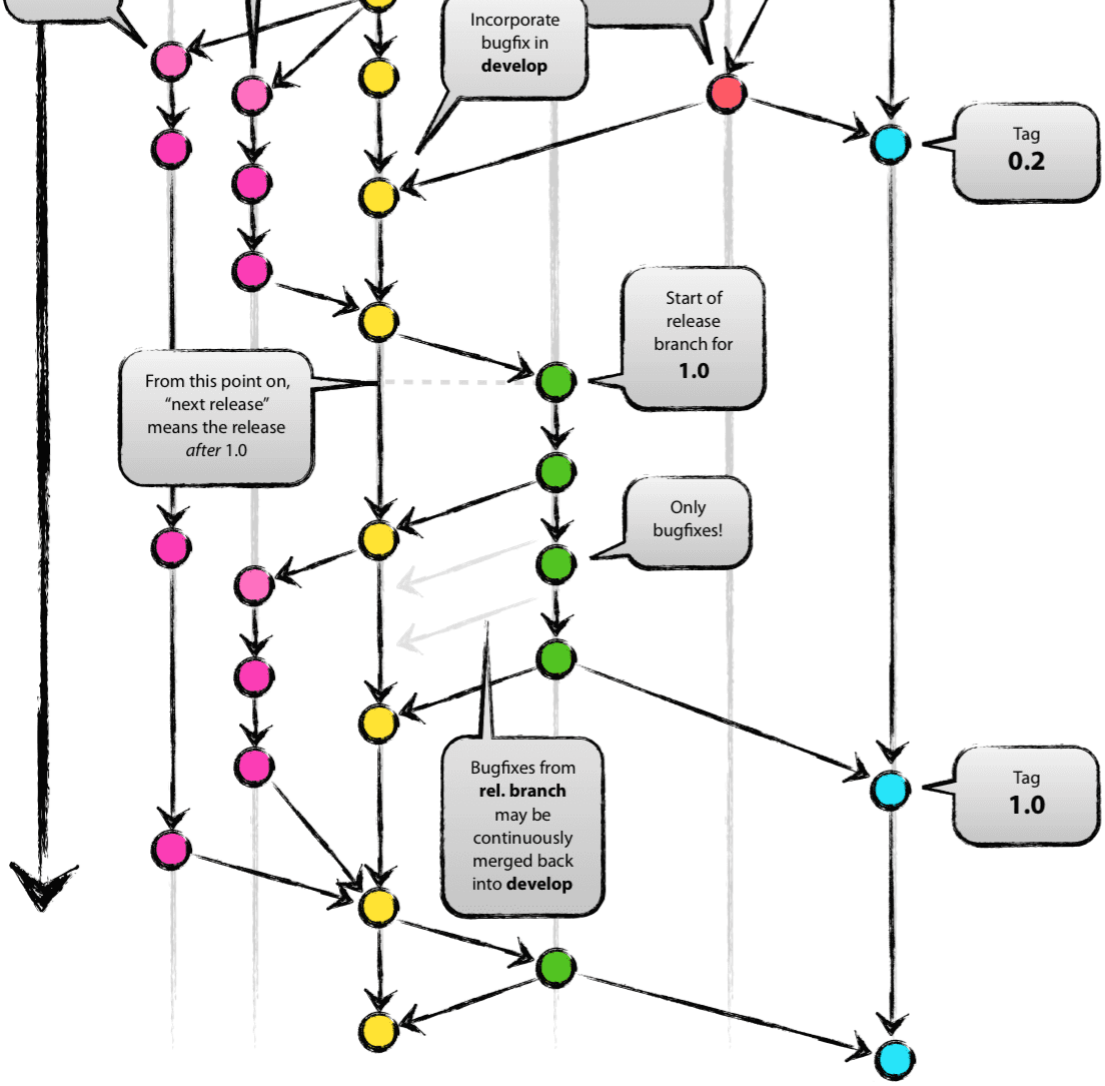
E se você é novo com git, este fluxo irá te ajudar a ter maior familiaridade de como empresas, projetos opensource costumam utilizar seus fluxos de trabalho.

É muito comum vermos pessoas utilizando somente um branch para fazer commits em projetos pessoais. Isto não é errado, é muito tranquilo de se controlar tudo em uma branch quando se está desenvolvendo sozinho, mas o cenário muda bastante quando temos que interagir com mais contribuidores, seja em um projeto opensource ou privado.

Nessas horas é suma importância que se tenha total controle do que está sendo produzido por sua equipe, onde, ao mesmo tempo são corrigidos falhas, implementado novas funcionalidades e ter o seu código de produção com total funcionamento entregue ao seu cliente.

É ai que o fluxo de git flow nos ajuda, vem com o cabrito olhar a imagem abaixo para entender melhor:





Git Flow Model Nvie

A **master** irá contér todo código já testado, versionado que será entregue ao cliente e a **develop** é onde todo fluxo de trabalho irá ocorrer antes de fazer o release versionado que será feito merge na **master**.

A **develop** deve sempre conter o código mais atual, onde as branchs de features serão ramificadas tendo ela como base.

Exemplo, suponhamos que você precise criar um feature que mudará todo o fluxo e interface de um componente, como fariamos para criar nossa branch ?

Certifique-se de que a branch develop existe no seu repositório remoto listando suas branchs locais e remotas:

```
$ git branch -a
```

Caso não esteja, faça a sincronização do seu repositório remoto, faça o checkout criando sua branch develop e envie para seu repositório remoto:

```
$ git fetch origin && git checkout -b develop && git push origin
develop
```

Após ter criado a develop, onde irá acontecer todo desenvolvimento, crie a branch respectiva a sua implementação, lembre-se de manter um padrão de nomenclatura para facilitar o entendimento como é sugerido no git flow:

feature: para novas implementações

release: para finalizar o release e tags

hotfix: para resolver problema crítico em produção que não pode esperar novo release

Neste caso, como já estamos na **develop**:

```
$ git checkout -b feature/novo-componente
```

Após criado, você trabalha em sua modificação localmente, caso seja necessário que outra pessoa trabalhe nesta mesma implementação você deve compartilhar para seu repositório remoto:

```
$ git push origin feature/novo-componente
```

Show, implementação feita, agora é hora de fazer o merge deste feature com a develop, para isto, faça o checkout para a branch develop, faça o merge da feature e atualize o remoto:

```
$ git checkout develop && git merge feature/novo-componente && git
push origin develop
```

Caso não ocorra nenhum conflito, beleza, estamos prontos para fazer o release desta implementação e submeter ao repositório remoto, para isto, crie a branch de release e envie:

```
$ git checkout -b release/v1.0.1 && git push origin release/v1.0.1
```

Após feito os ultimos testes, você já pode fazer a tag da versão:

```
$ git tag -a v1.0.1 -m “Release do novo componente”
```

Lembrando, que se foi identificado algum bug durante o processo, você deve tratar este bug na branch de release, enviar para a master e para a develop também, fazendo que a develop sempre contenha as correções.

Nas hora de inserir a tag, gosto de utilizar tag anotadas, pois ela registra informações de quem fez a tag, data, hash, caso não queira estas informações, simplifique:

```
$ git tag v1.0.1
```

Agora vamos conferir se a tag foi criada e enviar para o repositório remoto:

```
$ git show v1.0.1 && git push origin v1.0.1
```

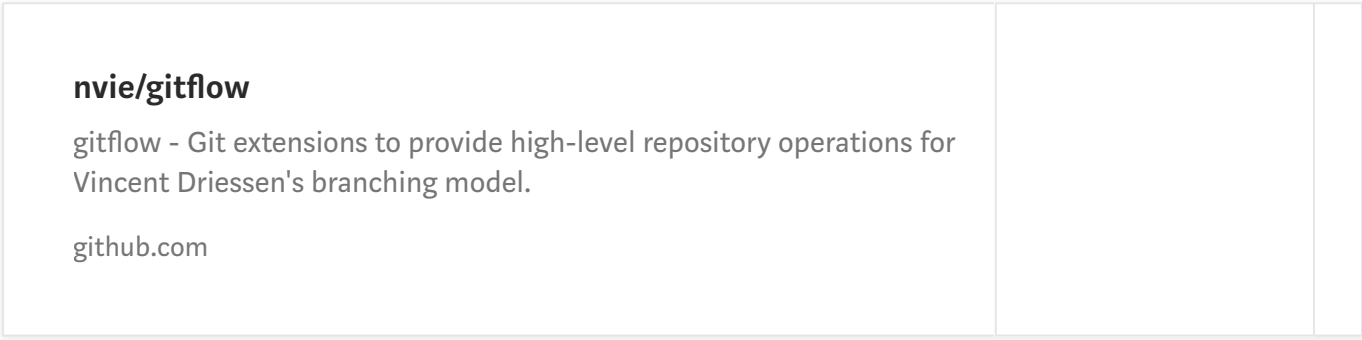
Se tudo correu bem, sua tag será criada e estamos aptos a fazer o merge com a master deste pequeno release na master:

```
$ git checkout master && git merge release/v1.0.1
```

XABLAU

Prontinho, desta forma, obtemos informações de todas as etapas do desenvolvimento, além de padronizar a nomenclatura das branchs facilitando na hora de puxar um log maroto.

Dica: Existe um plugin para facilitar a criação e organização do seu repositório utilizando o fluxo do git-flow, se liga nesse plugin massa!



Resumindo, aprendemos como controlar nossas branch separadas por suas responsabilidades, não impactando na master que é onde nosso código estável se mantém fiel, utilizamos tags para versionar nossas releases e ter um controle bem mais flexível.

Caso tenha faltado algum conceito em si, deixe seu comentário que assim que possível faço a correção marota, belezura?

Abraços do cabrito.

Git Gitflow Developer Tools