

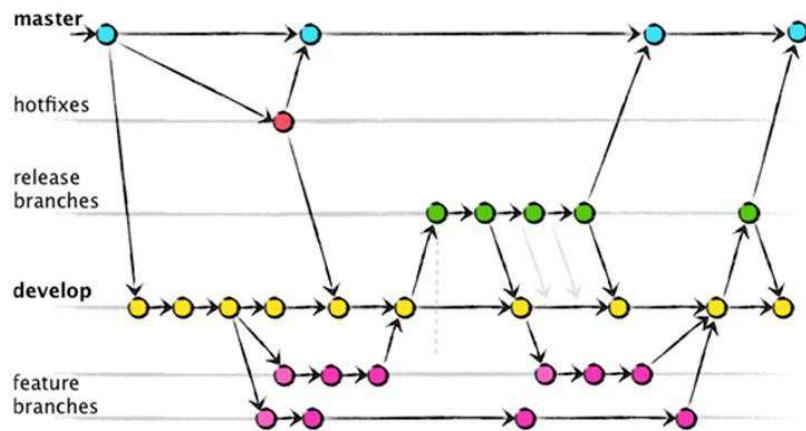
GitFlow



Diego Felix

Jun 1, 2016 • 4 min read

Um fluxo de trabalho GIT simples e seguro.



Antes de mais nada, vale a pena destacar o que não vai ser encontrado aqui:

1. A importância de controlar versão de código fonte: Se você não sabe porque os desenvolvedores profissionais (e amadores também!) utilizam algum mecanismo de controle de versão, este post não será muito útil. Se você estiver nessa situação, sugiro a seguinte leitura <https://blog.wkm.com.br/o-que-%C3%A9-e-porque-usar-um-sistema-de-controle-de-vers%C3%A3o-23f00b08e12d#.7vkp7be03>
2. Referência GIT: Existem livros e sites inteiros à respeito disso, inclusive a documentação oficial (<https://git-scm.com/doc>). Esse tópico não será tratado neste post.

3. SVN vs GIT vs TFS: Esse post não se trata de comparar sistemas de controle de versão. Existem diversos estudos sobre o tema, principalmente SVN vs GIT; E é muito fácil achá-los pela internet. Como indicação, seguem 3 páginas:
<https://git.wiki.kernel.org/index.php/GitSvnComparison>, <https://svnvsgit.com/> e <http://stackoverflow.com/questions/871/why-is-git-better-than-subversion>.

Dito isto, vamos ao que interessa: Gitflow. Como o título do post indica, Gitflow é um fluxo de trabalho simples e seguro na plataforma GIT. Mas pq esta ênfase nos termos “simples” e “seguro”? Para responder essa pergunta, temos que entender as vantagens do uso do GIT e o cenário sem um fluxo de trabalho.

Motivos para a escolha do GIT:

- Por ser um sistema de controle de versão distribuído, cada desenvolvedor tem uma cópia total do projeto em seu ambiente local;
- Multi-plataforma, open-source e grátis;
- Pequeno e rápido;
- Baseado em Branching & Merging, o que flexibiliza e facilita muito qualquer fluxo de trabalho;
- Matém a integridade do projeto com o auxílio de mecanismos de validação (checksum) dos commits/arquivos
- Flexível, permitindo que o desenvolvedor, ou a equipe, acople o GIT no seu processo de trabalho.

E é nesse último item que está o possível problema: É necessário a definição do fluxo de trabalho.

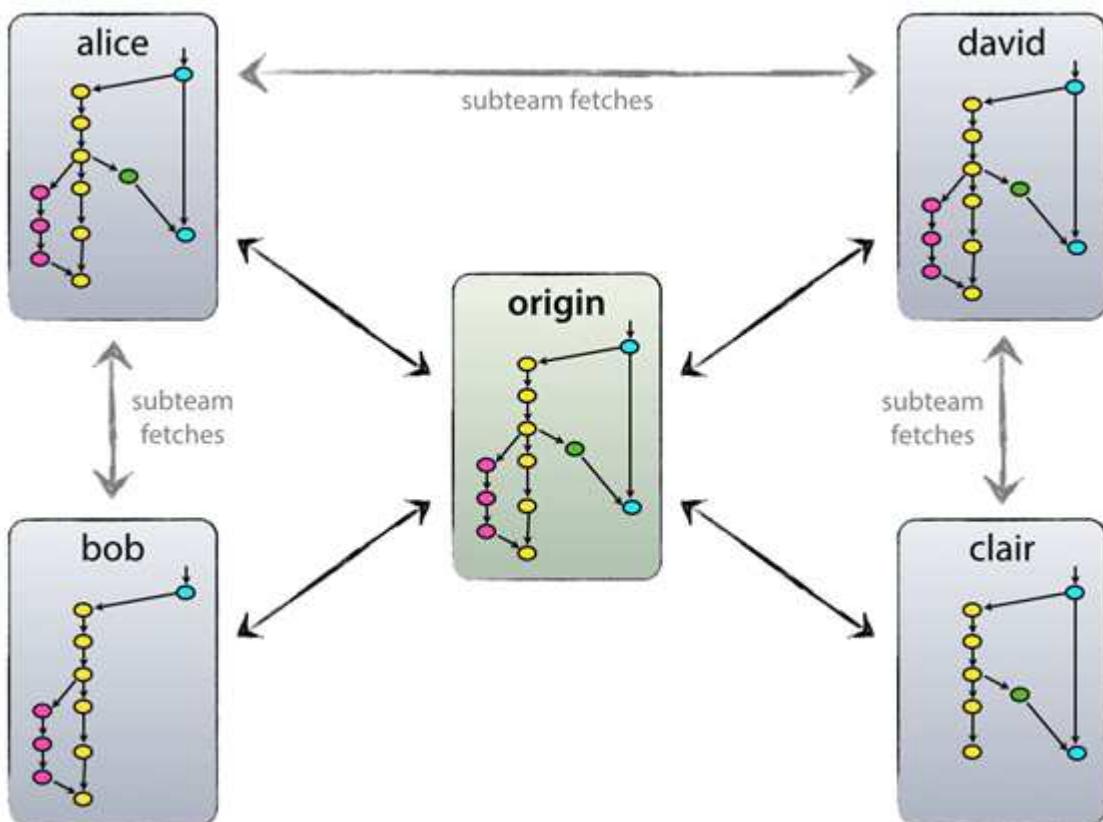
Os vários fluxos possíveis podem ser classificados, genericamente, em:

- **Centralizado:** O repositório remoto é único e commits são feitos no master diretamente.
- **Branches de funcionalidade:** Re却itório remoto único. Commits em branches específicos e, posteriormente, merges com o master.
- **Bifurcado:** Re却itórios remotos e um é escolhido como oficial. Commits em branches separados. Uma pessoa fica responsável pelos merges com o re却itório

oficial.

O GitFlow é uma evolução do tipo bifurcado, com as seguintes características:

- Vários repositórios locais e um remoto (origin).
- Branches com semântica: Feature, Release, Master, Develop e Hotfix.
- Pontos de merges são bem definidos e claros.
- Qualquer desenvolvedor pode fazer merge com o master



Semântica dos branches

Uma das principais características do GitFlow é a semântica atrelada aos branches. Cada uma tem seu propósito e regras associadas. Seguem abaixo a definição de cada tipo de branch.

Master

- Atemporal, reflete o código de produção.
- Cada merge/commit no master deve ser passível de implantação em produção.

- Uso de tags para marcar as atualizações, preferencialmente indicando a versão. Por exemplo, “3.0.2”.
- Criado na inicialização do GitFlow.

Develop

- Atemporal, é o histórico do desenvolvimento.
- Linha de integração. As features são juntadas nesse branch.
- Os commits não deixam a linha necessariamente estável. O ideal é que sejam rodados testes completos a cada commit nesse branch.
- Quando se alcança um estado entregável, o código desta linha deve ser colocado em master, através de um release.
- Criado na inicialização do GitFlow.

Feature

- Volátil, representa as novas funcionalidades que estão sendo desenvolvidas.
- Normalmente, só existem em ambiente local.
- Iniciado a partir de develop. No seu fechamento, o merge é feito em develop também.

Release

- Linha de homologação.
- Geração de pacotes para produção.
- Quando se abre um branch de release, ou tudo que está nele é aprovado ou tudo que está nele é reprovado. Pequenos ajustes podem ser feitos no próprio release que está aberto.
- Criado a partir de develop. Quando fechado, é feito um merge no develop e no master

Hotfix

- Linha de correções.

- Geração bugfixes. Criado quando um erro é encontrado e deve ser corrigido em produção.
- Criado a partir de master. Quando fechado, é feito um merge no develop e no master.

Comandos

O GitFlow tem alguns comandos usando os conceitos semânticos já expostos. Esses comandos nada mais são que sequências de comandos GIT.

Start feature “myfeature”

1. \$ git checkout -b myfeature develop

Finish feature “myfeature”

1. \$ git checkout develop
2. \$ git merge --no-ff myfeature
3. \$ git branch -d myfeature

Finish release “realease-1.2”

1. \$ git checkout master
2. \$ git merge --no-ff release-1.2
3. \$ git tah -a 1.2
4. \$ git checkout develop
5. \$ git merge --no-ff release-1.2
6. \$ git branch -d release-1.2

(Algumas) Boas práticas

- Sempre saiba o que está fazendo!
- Mantenha a linha develop estável!
- Code review antes de cada merge!
- Testes depois de cada merge!

- Não confie cegamente no merge automático!
- Não saia do Fluxo! Se perceber que saiu, volte ao fluxo o mais rápido! Peça ajuda, se necessário.
- Mantenha a timeline do projeto limpa!
- Use ferramentas que auxiliem no controle do Fluxo!

Referências/Links Úteis

- Git-flow Cheatsheet — <http://danielkummer.github.io/git-flow-cheatsheet/>
- A successful git branching model — <http://nvie.com/posts/a-successful-git-branching-model/>
- Comparing git workflows — <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow/>
- Atlassian SourceTree — <https://www.sourcetreeapp.com/>
- Using git-flow to relieve your headaches — https://www.youtube.com/watch?v=NdXhz4rt_sQ

—
Quer conhecer mais sobre soluções digitais inovadoras?

Acesse: <http://www.wkm.com.br>

Git Gitflow

About Help Legal