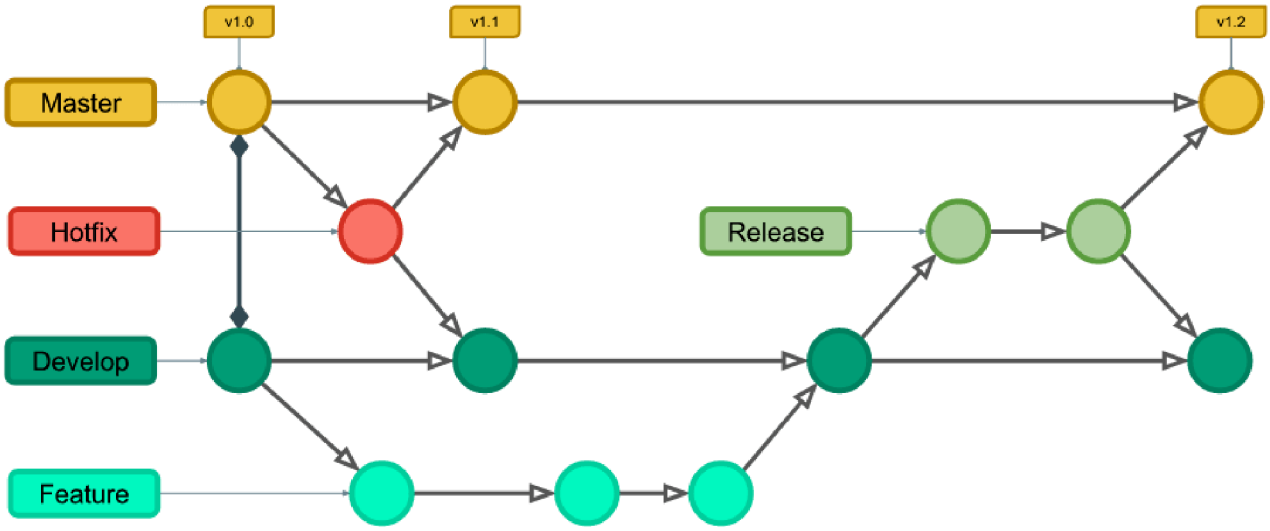
04

Git Flow

Transcrição

Esta forma de organização de branches que facilita a resolução de conflitos, inclusive para que bugs repetidos não aconteçam, é denominada ***Git flow***, bem representada em alguns gráficos, como este:



Nele, o branch master só serve para receber os commits prontos para ir à produção, a partir dos quais geramos as tags. Existe também o branch de desenvolvimento (*Develop*), de que serão criados os branches de feature, ou seja, funcionalidades novas, os quais serão enviados de volta ao branch de desenvolvimento em dado momento. Quando o branch de desenvolvimento tiver todas as features, criamos um branch de release, isto é, quando começa o processo de lançarmos uma nova versão.

E no branch de release somente corrigimos os bugs relacionados ao release em si. Ou seja, sempre que um bug for corrigido, é preciso enviá-lo de volta ao branch de desenvolvimento, pois outras features podem se aproveitar desta

correção. E no final, tudo estando corrigido, irá para a master para receber uma tag. Caso ocorra algum problema em produção, ou seja, no branch master, um branch de *hotfix* precisará ser criado e, sempre que corrigido, tudo é enviado à master, e em seguida, também para o branch de desenvolvimento. Claro, pois todas as features se aproveitarão desta correção.

Há outro gráfico, que envolve vários branches de features, ou seja, de funcionalidades, um geral, de desenvolvimento, branches de release, ou seja, versões a serem lançadas, de hotfixes e o master. Portanto, são os mesmos branches dispostos em *top down*, em vez de horizontalmente. Tudo começa na master, de onde se cria um branch de desenvolvimento, até que surja a necessidade de uma nova funcionalidade, e para isto, são criados branches.

Enquanto isso, quando se encontra um bug em produção, é criado um branch a partir da master, o bug é corrigido e a correção é enviada de volta à master, sendo criada uma tag de correção. Feito isso, a correção é enviada ao branch de desenvolvimento, para que todas as features, quando forem enviadas ao branch de desenvolvimento, tenham a correção implementada.

Quando o desenvolvimento de uma feature for finalizado, podemos começar o processo de desenvolvimento da release. Para tal, criamos um branch de release, dentro do qual corrigimos bugs específicos dela. Se o bug em específico não afeta esta release, não mexeremos nele; se é uma funcionalidade nova, não é este o momento de mexermos nela.

Conforme os bugs são corrigidos, os branches de desenvolvimento são atualizados. Quando uma nova feature é finalizada, ela é enviada ao branch de desenvolvimento, de onde é enviada para o branch de release. Quando todos os testes forem finalizados e o branch de release estiver pronto, atualizamos o branch de desenvolvimento enviando a correção também para o branch master, e criando uma nova tag, ou versão.

Este fluxo denominado Git flow pode ajudar equipes em projetos grandes, diminuindo o número de conflitos, organizando o ciclo de desenvolvimento de uma versão, algo bem interessante. Quando estamos desenvolvendo sozinhos, não parece muito prático, mas quando estamos em uma equipe grande, isso faz toda a diferença. Claro, esta não é a única estratégia de organização de fluxo existente.

Atualmente, na empresa em que trabalho, por exemplo, lidamos com um projeto de cerca de vinte anos, e temos um sistema que gerencia as demandas, chamadas *tickets*, e cada uma delas vira um branch. A funcionalidade ou correção que gerou o novo branch se desloca para um branch específico de *code review* (revisão de código) para garantir que o código esteja correto, passando ao branch de testes, em que a equipe de qualidade analisa se todo o restante do sistema continua funcionando com esta implementação, e caso positivo, se seguirá ao branch de release, específica da versão.

Quando a versão estiver finalizada, é realizado o merge para o master. Então, é um pouco diferente, embora se baseie em algo similar, mas exemplifica a possibilidade de não termos que necessariamente aplicar todos os branches e conceitos do Git flow, e sim somente aqueles que fazem sentido para o trabalho e equipe.

É verdade que o trabalho de lidar com vários branches é bastante manual, e é por isso que conversaremos a seguir sobre ferramentas que podem nos ajudar com isso!