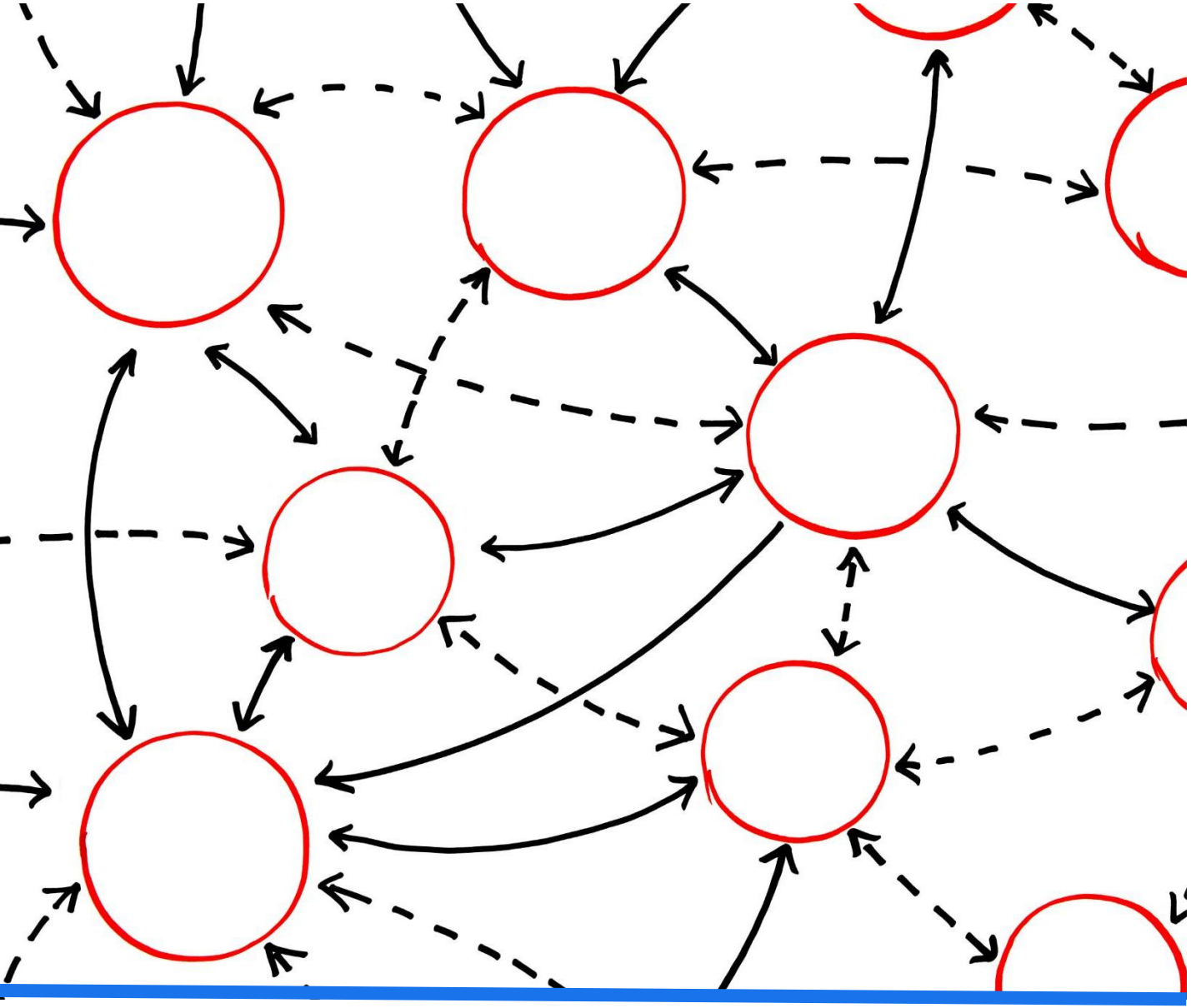


Conceitos Básicos do Git Flow

ENTENDA A METODOLOGIA PARA GERENCIAMENTO DE BRANCHES

Itens da Agenda

- Introdução ao Git Flow
- Branches principais do Git Flow
- Tipos de branches de suporte
- Fluxo de trabalho com Git Flow
- Boas práticas e dicas

Introdução ao Git Flow

O que é Git Flow?

Gerenciamento de Branches

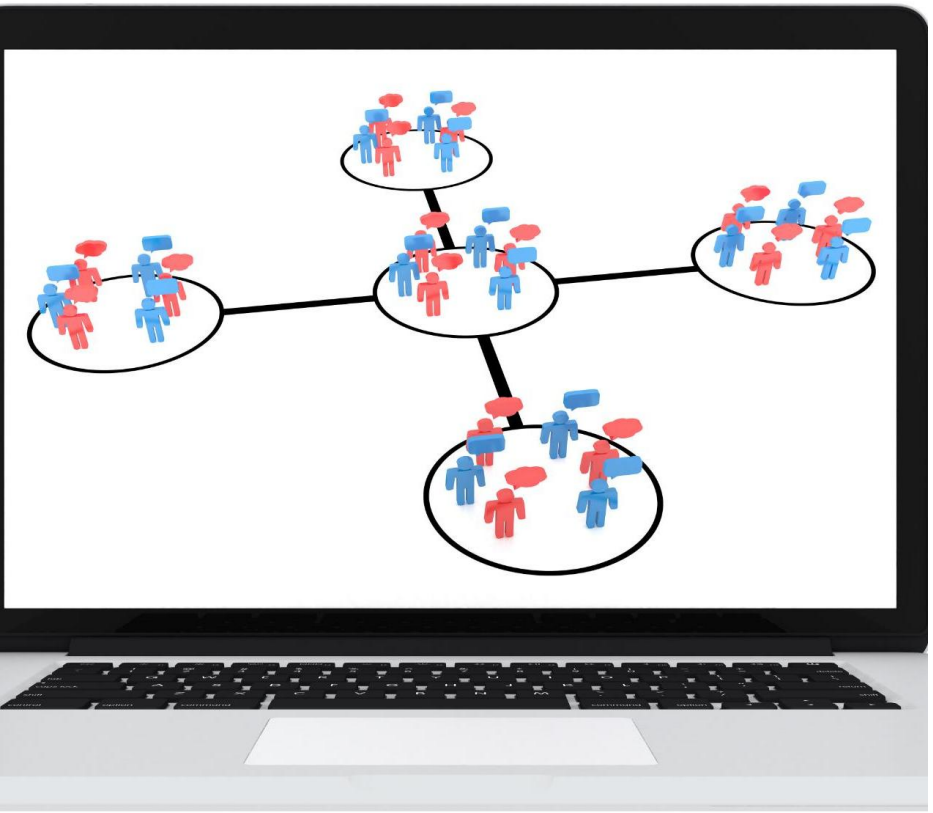
Git Flow fornece um conjunto de regras para gerenciar branches de forma eficaz em projetos de desenvolvimento.

Interação da Equipe

Define como as equipes devem interagir com branches, facilitando a colaboração durante o desenvolvimento de software.

Ciclo de Vida do Desenvolvimento

Git Flow orienta o ciclo de vida do desenvolvimento de software, garantindo um fluxo de trabalho estruturado.



Por que usar Git Flow?

Clareza e Estrutura

Git Flow traz clareza ao processo de desenvolvimento, organizando as etapas e os fluxos de trabalho de forma eficiente.

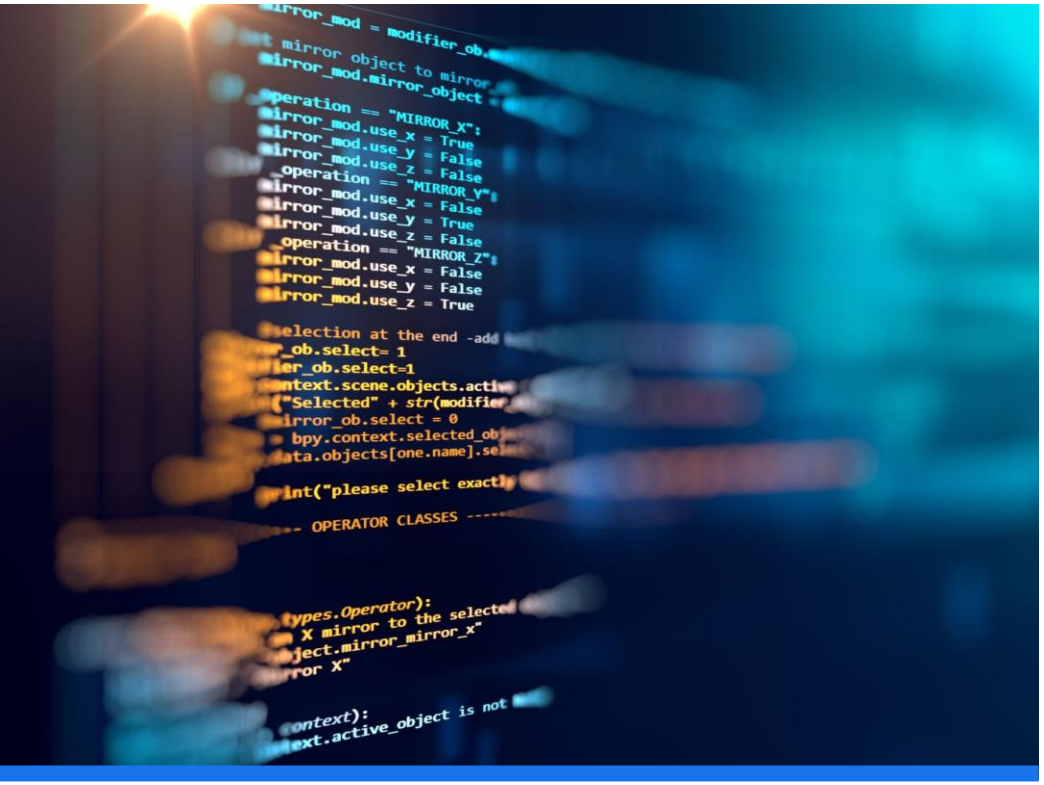
Trabalho Paralelo

Permite que várias equipes trabalhem em funcionalidades diferentes ao mesmo tempo, aumentando a produtividade e a colaboração.

Estabilidade do Código

Mantém o código base estável e pronto para produção, reduzindo o risco de bugs e problemas durante o lançamento.

Benefícios do Git Flow



Lançamentos de Versões Controladas

O Git Flow permite que equipes realizem lançamentos de versões de forma controlada, garantindo que cada versão seja estável.

Resolução de Conflitos Facilitada

O modelo de ramificação do Git Flow facilita a resolução de conflitos, permitindo uma colaboração mais eficiente entre os desenvolvedores.

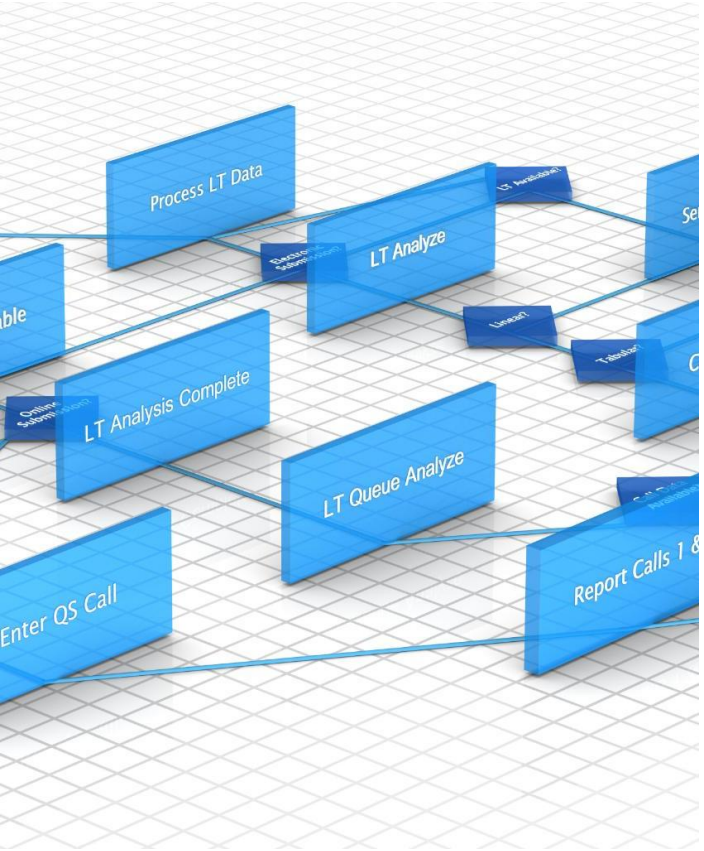
Melhor Organização do Código

O Git Flow promove uma melhor organização do código, permitindo que as equipes mantenham um repositório limpo e gerenciável.

Eficiência na Integração Contínua

O uso do Git Flow torna o processo de integração contínua mais eficiente, facilitando a entrega rápida de novas funcionalidades.

Branches principais do Git Flow



Branch Master

Definição da Branch Master

A branch Master é a principal branch no repositório de código, representando a versão estável do projeto.

É comum também usar a denominação branch Main

Importância da Estabilidade

É crucial que a branch Master sempre reflita um estado estável, para evitar problemas no ambiente de produção.

Os lançamentos idealmente devem ser feitos a partir da branch Master, garantindo que apenas código testado seja enviado à produção.

Mas novas releases poder ser lançadas a partir da Branch Develop, se esta for estável.

Branch Develop

Integração de Funcionalidades

A branch Develop é crucial para a integração de novas funcionalidades, permitindo que a equipe desenvolva e teste novas features antes da fusão final.

Ambiente Estável

Esta branch atua como um ambiente estável para a equipe, garantindo que todas as novas funcionalidades sejam testadas antes do lançamento.

Mesclagem com Master

Após o desenvolvimento e testes, as novas features do Dev são mescladas na branch Master, preparando para o lançamento final do software.

Estrutura de branches

Branches Principais

As branches principais no Git Flow incluem a branch de desenvolvimento e a branch principal, que são essenciais para o ciclo de vida do projeto.

Branches de Suporte

Branches de suporte são criadas para gerenciar novas funcionalidades, lançamentos e correções. Elas facilitam o fluxo de trabalho e a organização do código.

Tipos de branches de suporte

Feature Branches

Criação de Feature Branches

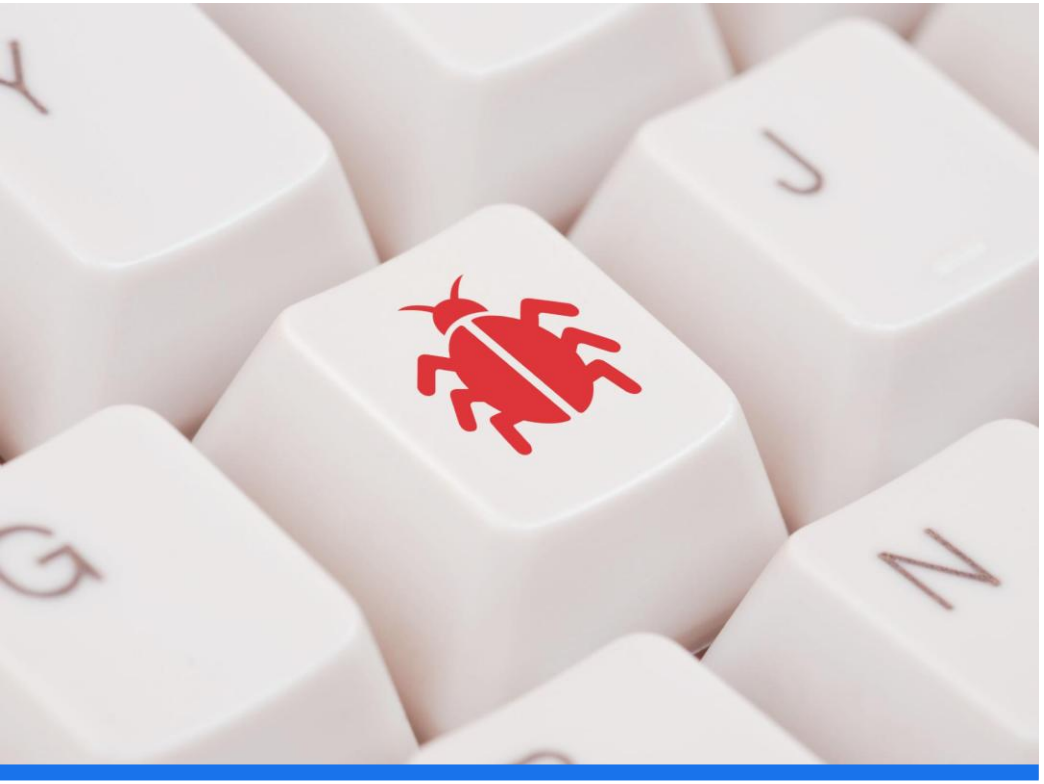
Feature Branches são criadas especificamente para o desenvolvimento de novas funcionalidades em um projeto de software.

As Feature Branches são derivadas da branch Develop, que representa a versão mais estável do código.

Isolamento de Desenvolvimento

Cada funcionalidade tem sua própria branch, permitindo que os desenvolvedores trabalhem de forma isolada até a conclusão.

Release Branches



Preparação para Lançamento

Release Branches são criadas para preparar uma nova versão do software para lançamento, facilitando ajustes finais.

Estas branches permitem corrigir bugs identificados antes do lançamento, assegurando um produto de qualidade.

Testes Finais

Os testes finais são realizados nas Release Branches, garantindo que o software esteja pronto para ser mesclado na branch Master.

Hotfix Branches

Uso de Hotfix Branches

Hotfix Branches são criadas para resolver problemas críticos que ocorrem em produção rapidamente.

Essas branches são originadas da branch Develop e, após a correção, são mescladas de volta na Branch Develop.

Fluxo de trabalho com Git Flow

Iniciando o Git Flow

Configuração do Repositório Git

A configuração inicial do repositório Git é crucial para estabelecer um fluxo de trabalho eficaz e organizado.

Definição de Branches Principais

Definir as branches principais (Master e Dev) é um passo fundamental para o gerenciamento adequado do projeto e controle de versões.

Criação de Feature Branches

A criação de Feature Branches permite que a equipe desenvolva novas funcionalidades sem afetar a versão principal do código.

Desenvolvimento de novas funcionalidades

Criação de Feature Branches

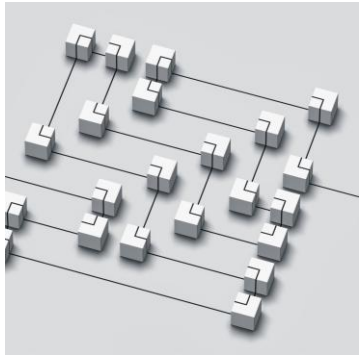
Os desenvolvedores devem criar Feature Branches a partir da branch Develop para trabalhar em novas funcionalidades de forma isolada.

Mesclagem de Branches

Após a conclusão da funcionalidade, a branch é mesclada de volta na Develop, integrando as novas features ao código existente.

Normalmente nesse passo é realizado pelo desenvolvedor um pull request, um pedido para subir a mudança para a Branch Dev.

Preparação e lançamento de versões



Criação da Release Branch

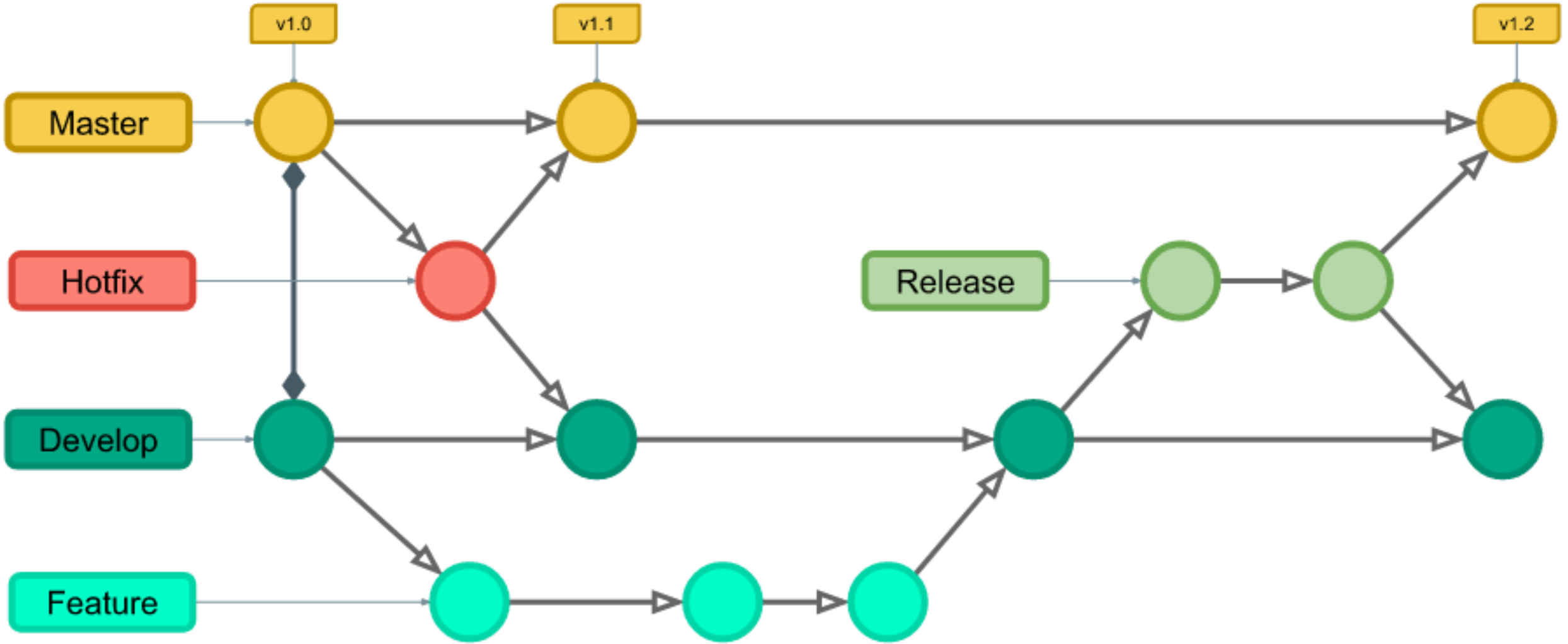
Quando a nova versão está pronta, uma Release Branch é criada a partir da Develop para iniciar o processo de lançamento.

Ajustes Finais

Depois de criar a Release Branch, são feitos os ajustes finais antes de mesclá-la na Master.

Mesclagem na Master

Após os ajustes finais, a Release Branch é mesclada na Master, garantindo um histórico organizado de alterações.


v1.0
v1.1
v1.2
Master
Hotfix
Release
Develop
Feature

Boas práticas e dicas

Nomeação Clara de Branches

É importante nomear as branches de maneira clara e descritiva para facilitar a identificação e o trabalho em equipe.

Atualização de Branches

Manter as branches atualizadas é essencial para garantir que todos os colaboradores estejam trabalhando com a versão mais recente do código.

Eliminação de Branches Desnecessárias

Eliminar branches que não são mais necessárias ajuda a evitar confusões e mantém o repositório organizado.