Projeto Integrado III

DD - UFC - Quixadá

Prof.: Aníbal Cavalcante

Sistemas de Controle de Versões (SCV)

O que é um SCV?

Um **sistema de controle de versão** (SCV) é um software para **gerenciar** diferentes **versões** de um documento qualquer durante seu **desenvolvimento.**

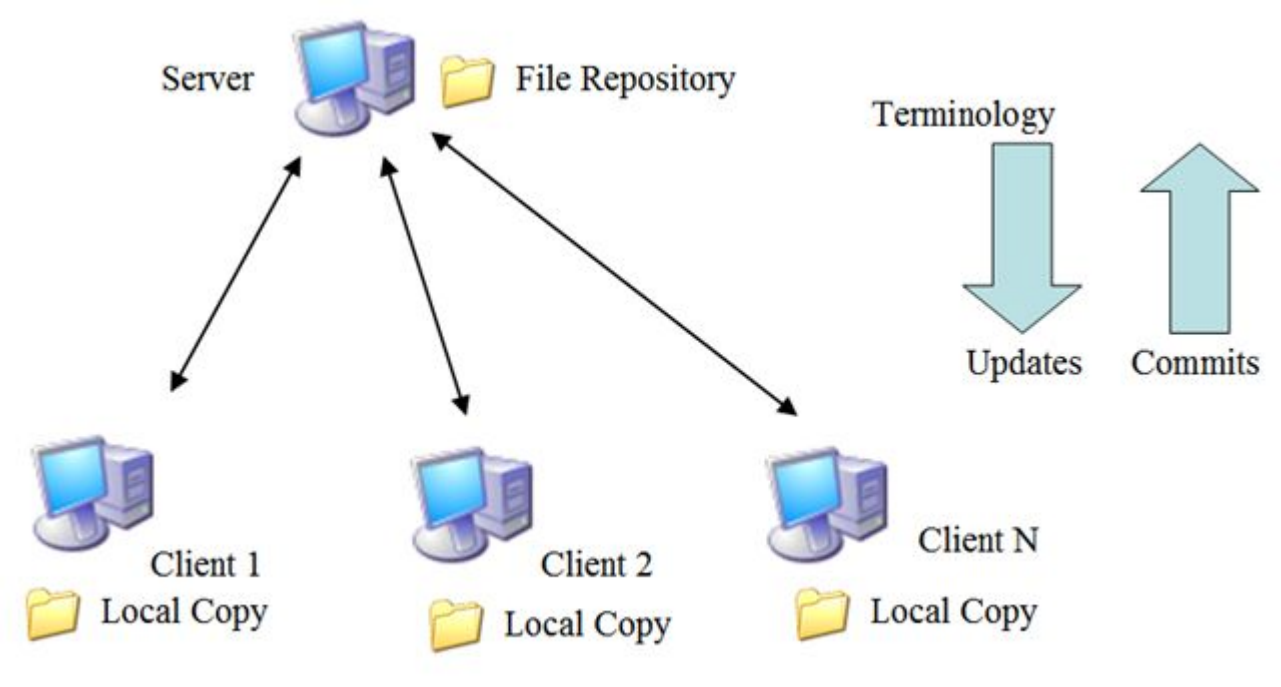
São utilizados no desenvolvimento de software para controlar as diferentes versões de:

- **códigos-fontes**
- **documentação**

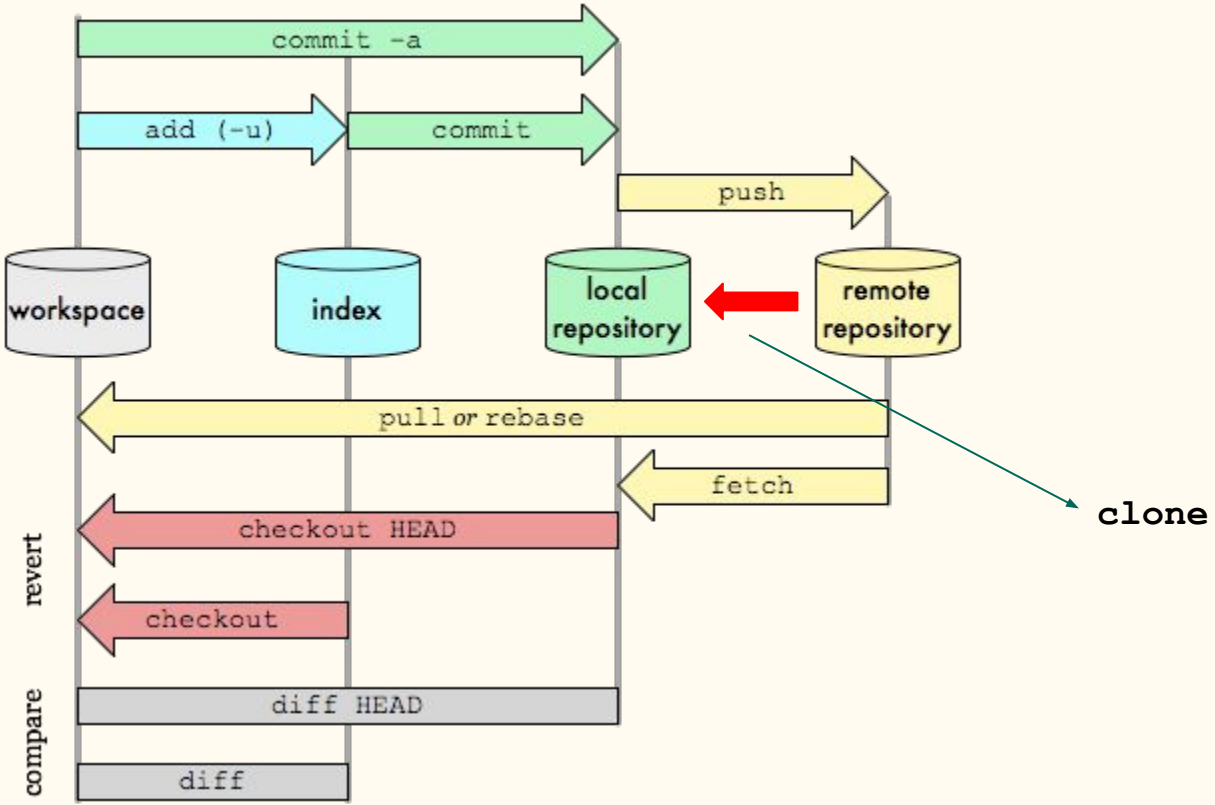
Benefícios

- Histórico de Modificações.
- Trabalho em equipe.
- Marcação e resgate de versões estáveis.
- Ramificação de projeto(linhas de desenvolvimento)

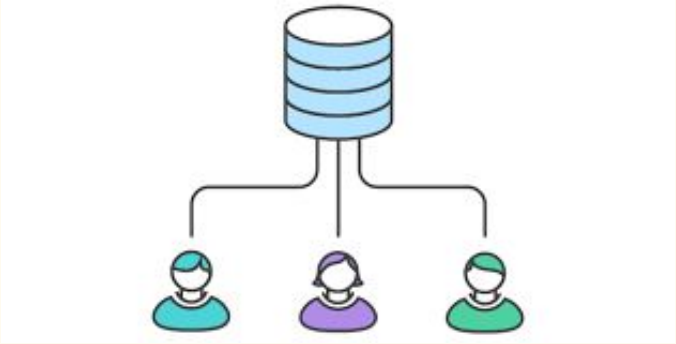
Funcionamento Básico (Repositório Centralizado)



Fluxo de Trabalho do Git (Git Workflow)



1 - Repositório Remoto

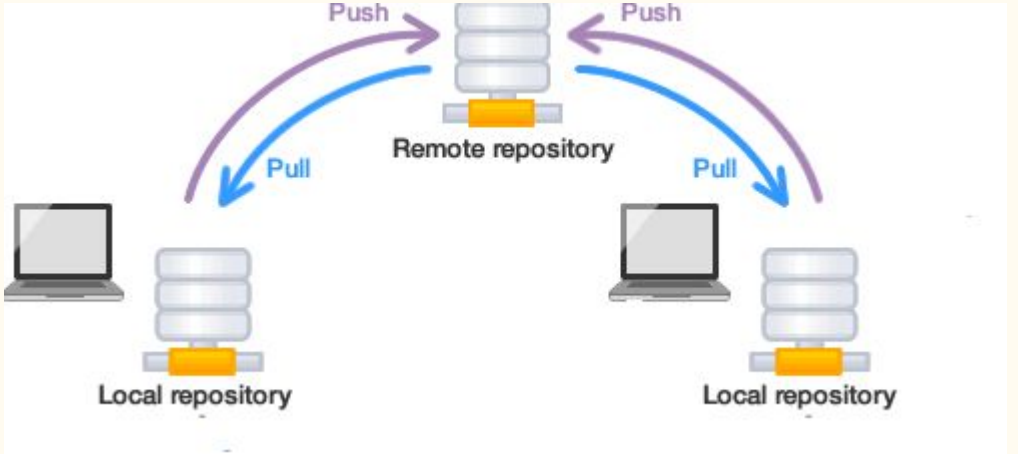


1. É onde ficam salvos os arquivos de forma centralizada.

2. Os desenvolvedores podem obter uma cópia desses arquivos utilizando o comando “clone”.

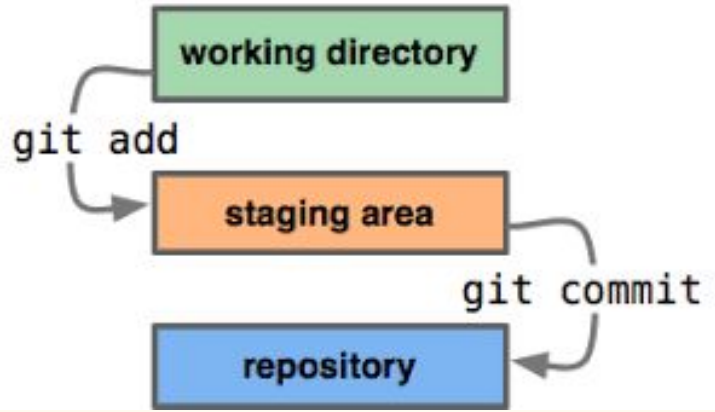
3. Um histórico de todos os arquivos é mantido nesse repositório, assim como, os responsáveis pelas alterações.

2 - Repositório Local



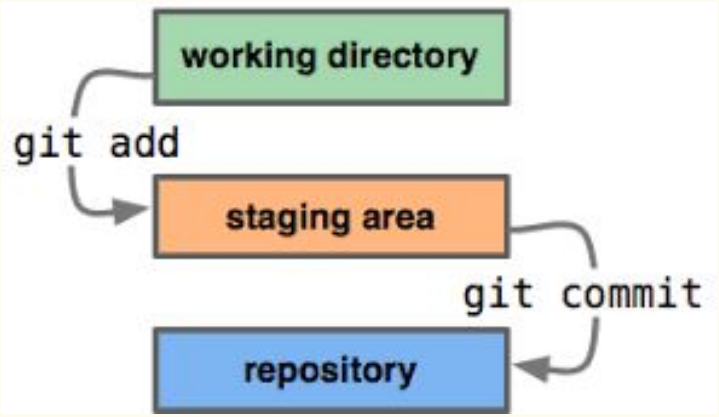
1. Inicialmente é uma cópia do repositório remoto.
2. Cada usuário tem uma cópia desse repositório.
3. É o nele que realizamos os commits.
4. A cada commit, o histórico dos arquivos fica salvo localmente.
5. Para compartilhar a sequência de commits com outros membros da equipe utiliza-se o comando “Push”.

3 - Index ou Stage Area (Área de Testes)



1. Faz o meio de campo entre o Workspace e o repositório local.
2. Gerencia quais arquivos deverão ser:
   a. “commitados”
   b. ignorados (Não versionados)
   c. incluídos (arquivos recentemente criados que serão incluídos no repositório)

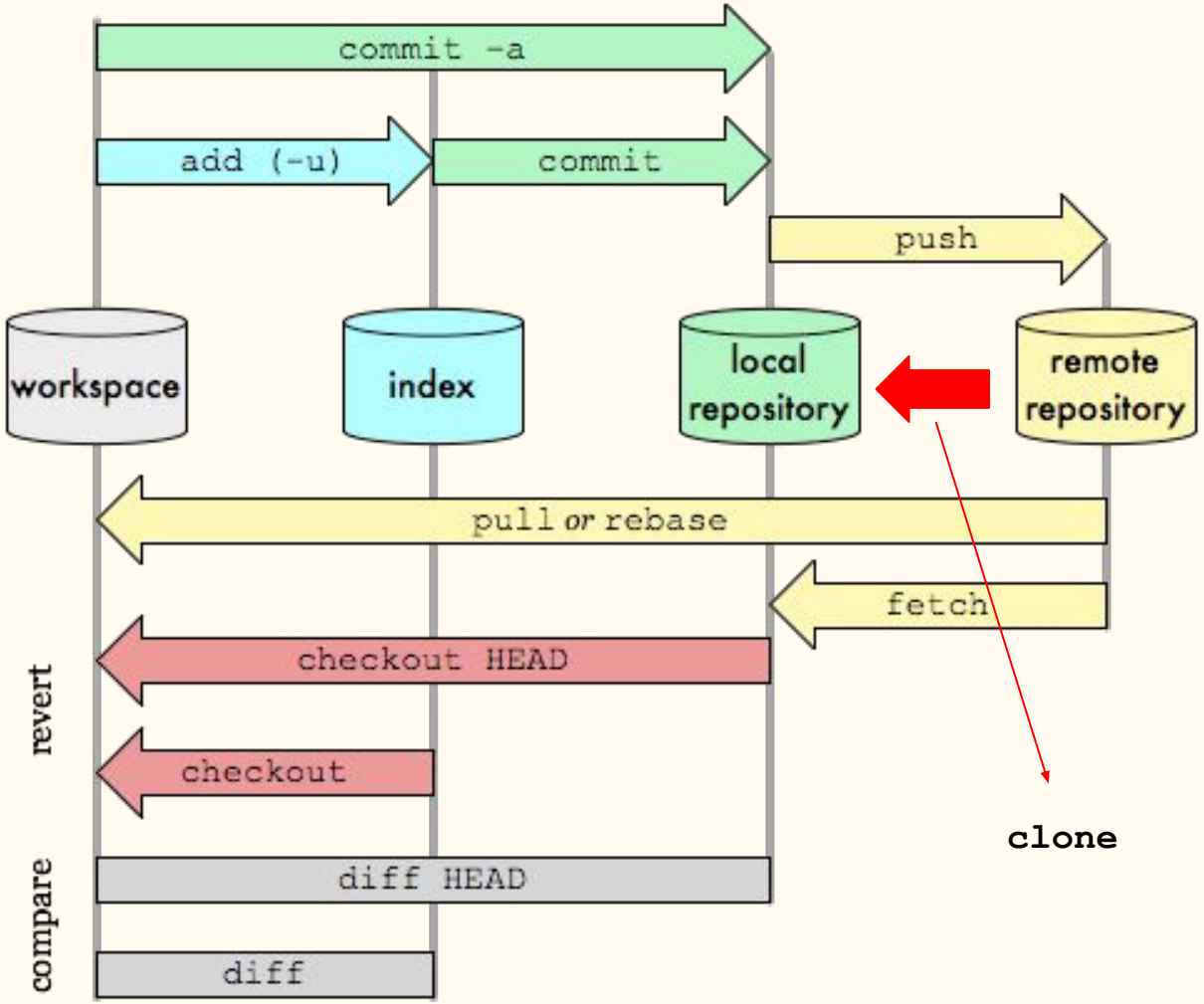
4 - Workspace



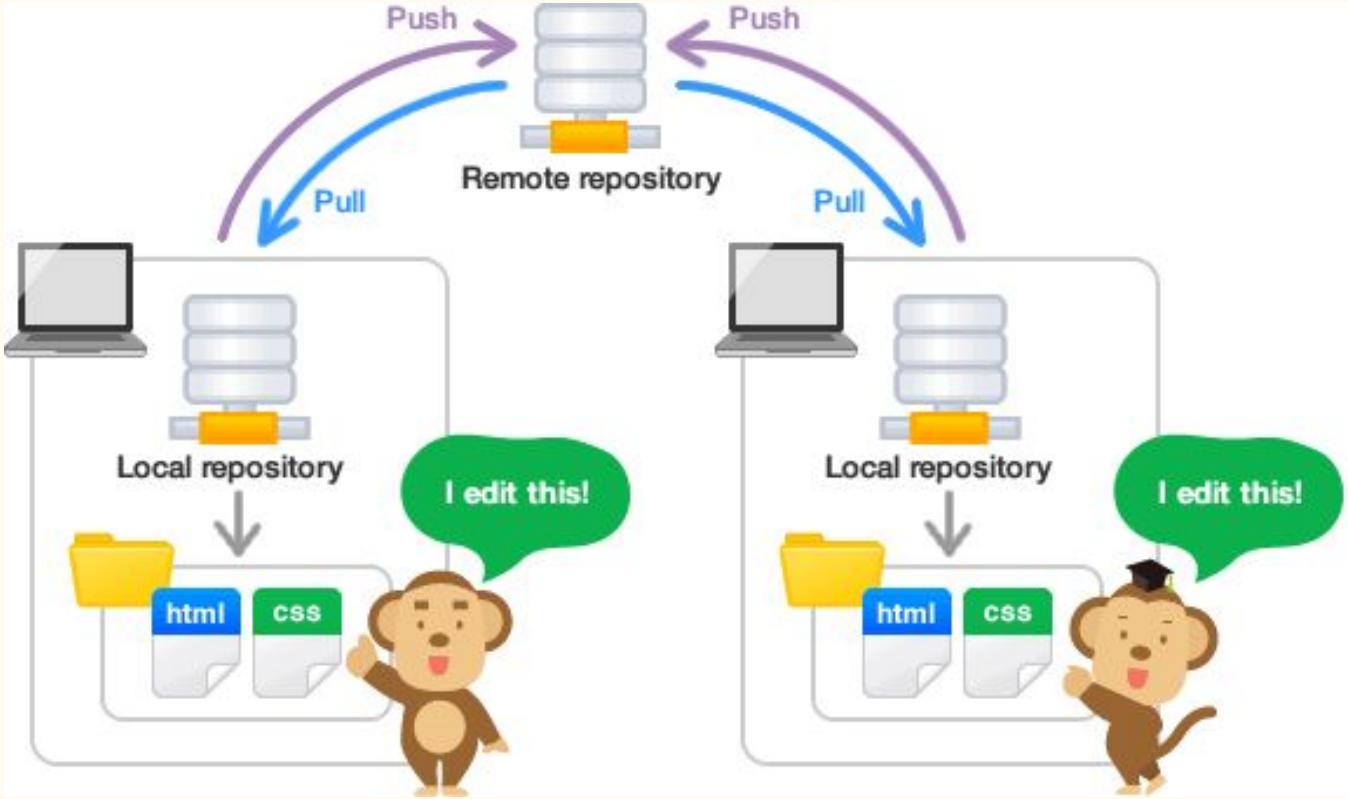
1. É o diretório local onde os seu códigos estão sendo editados.
2. Inicialmente ele está igual ao repositório local.
3. Modificações nos arquivos do workspace, podem ser comparadas com o repositório local.
4. A comparação é feita graças ao index.
5. Novos arquivos criados no workspace devem ser adicionados à área de testes, através do comando “**add**”

Git Workflow

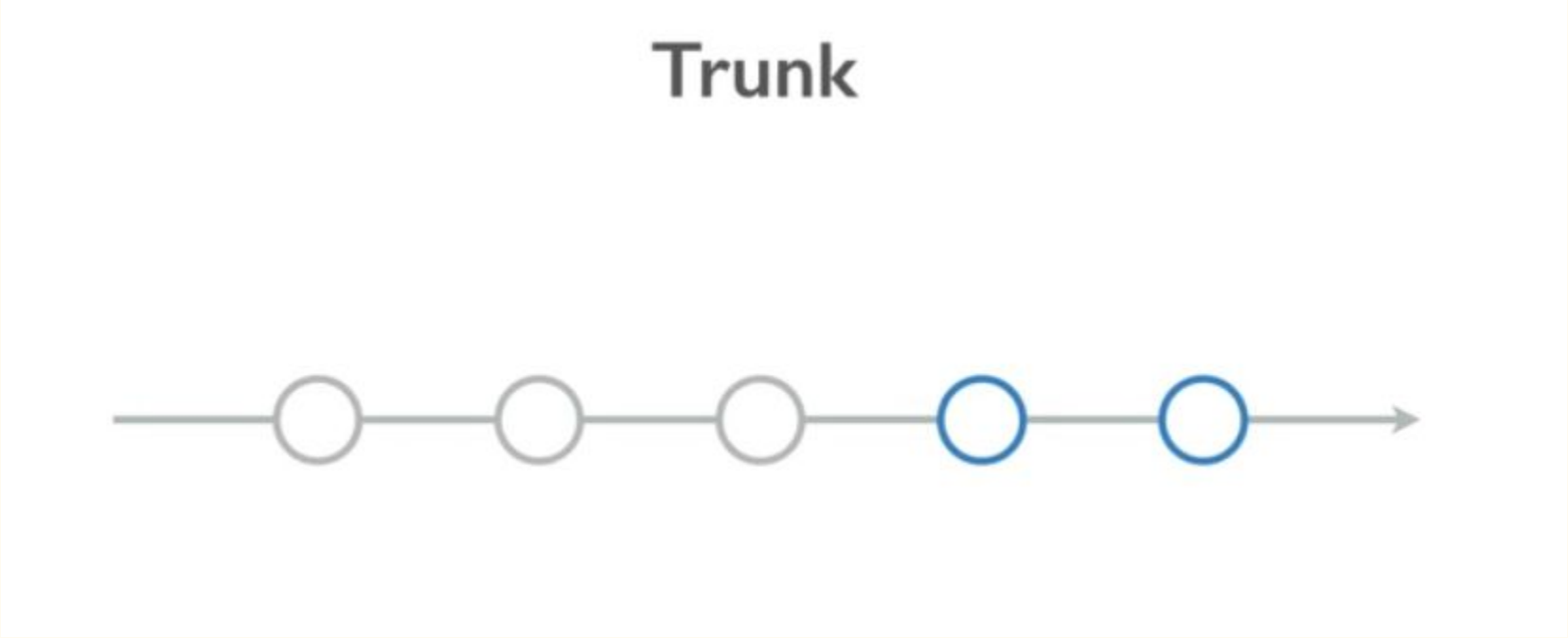
1. clone
2. add (-u) # git < 2.0
3. commit (- a)
4. push
5. pull or rebase
6. fetch
7. checkout HEAD
8. checkout
9. diff HEAD
10. diff



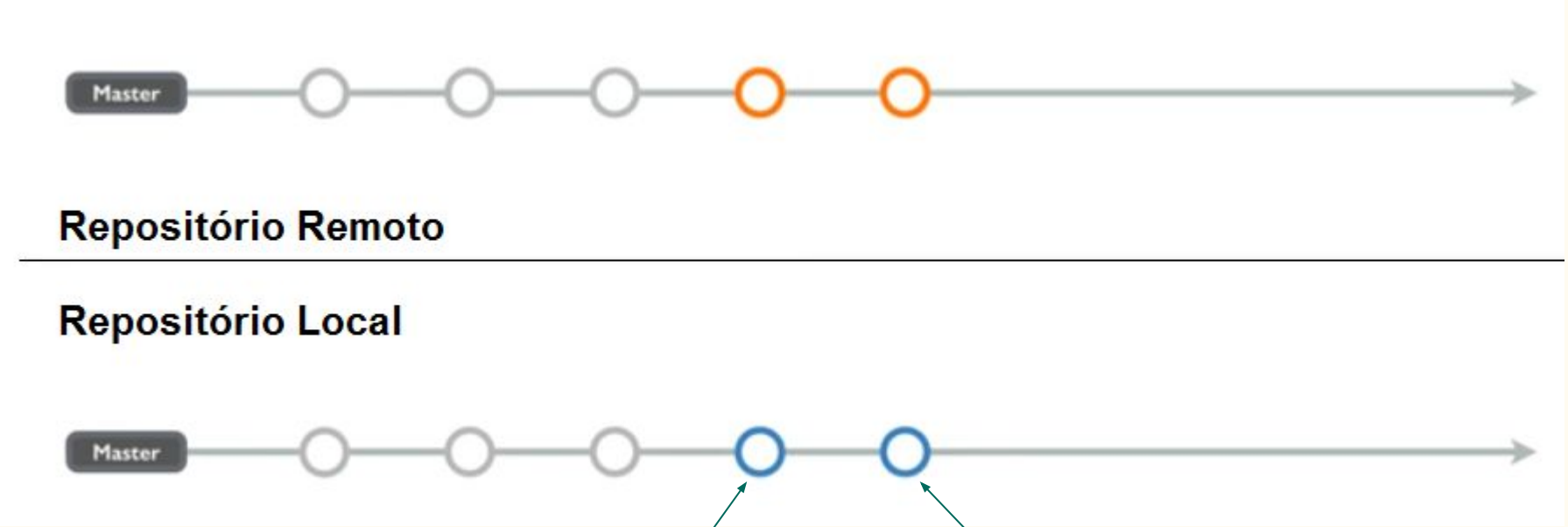
O cenário até o momento é esse, mas....



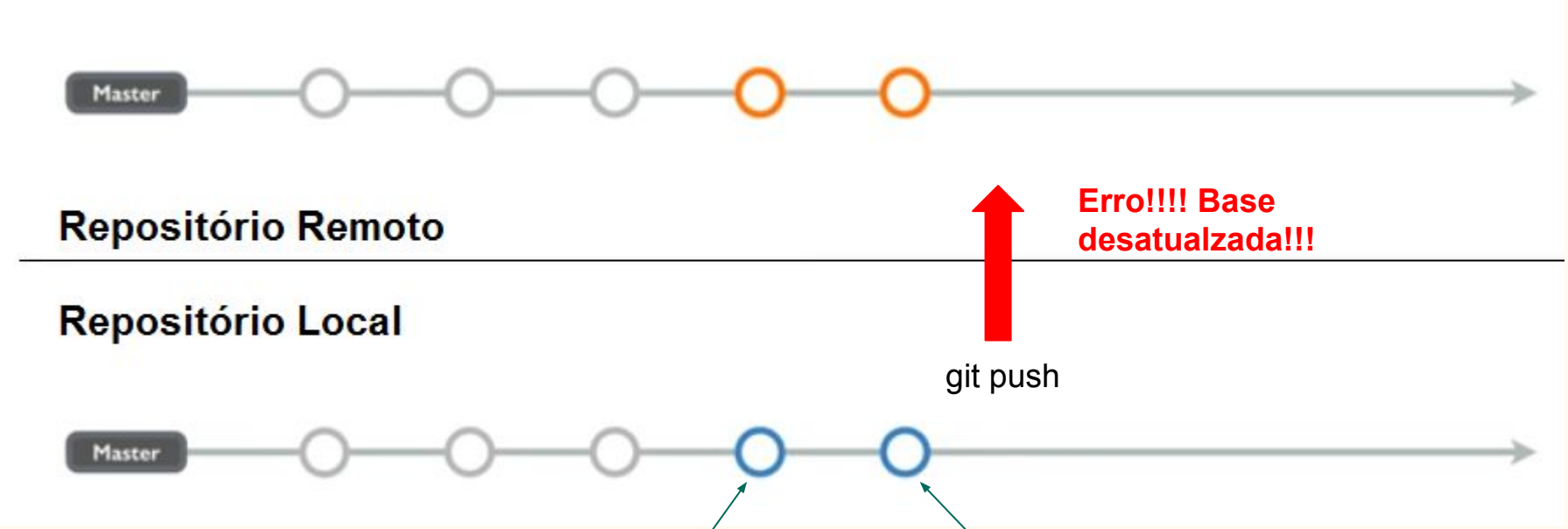
Trabalhando com Ramos (Branches)



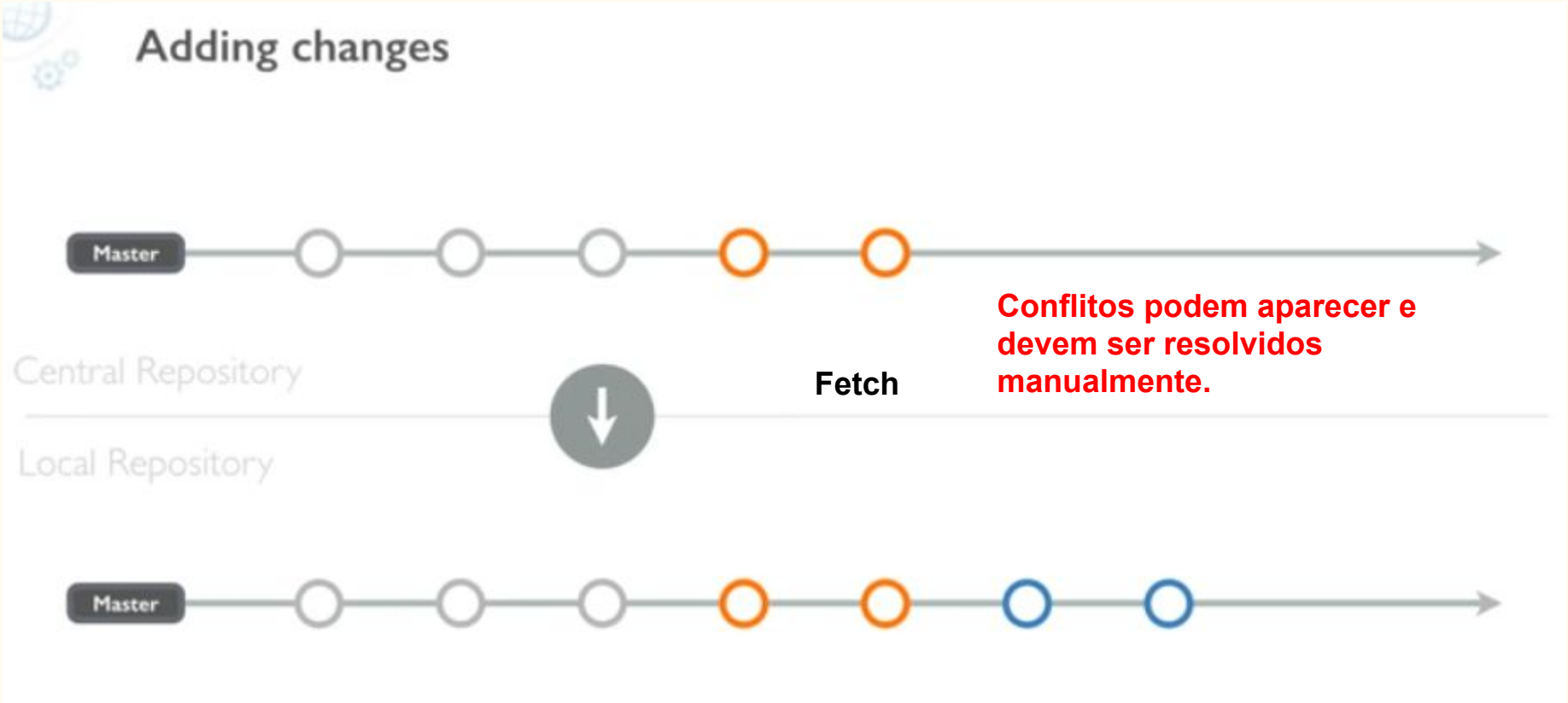
Trabalhando com Ramos (Branches)



Trabalhando com Ramos (Branches)

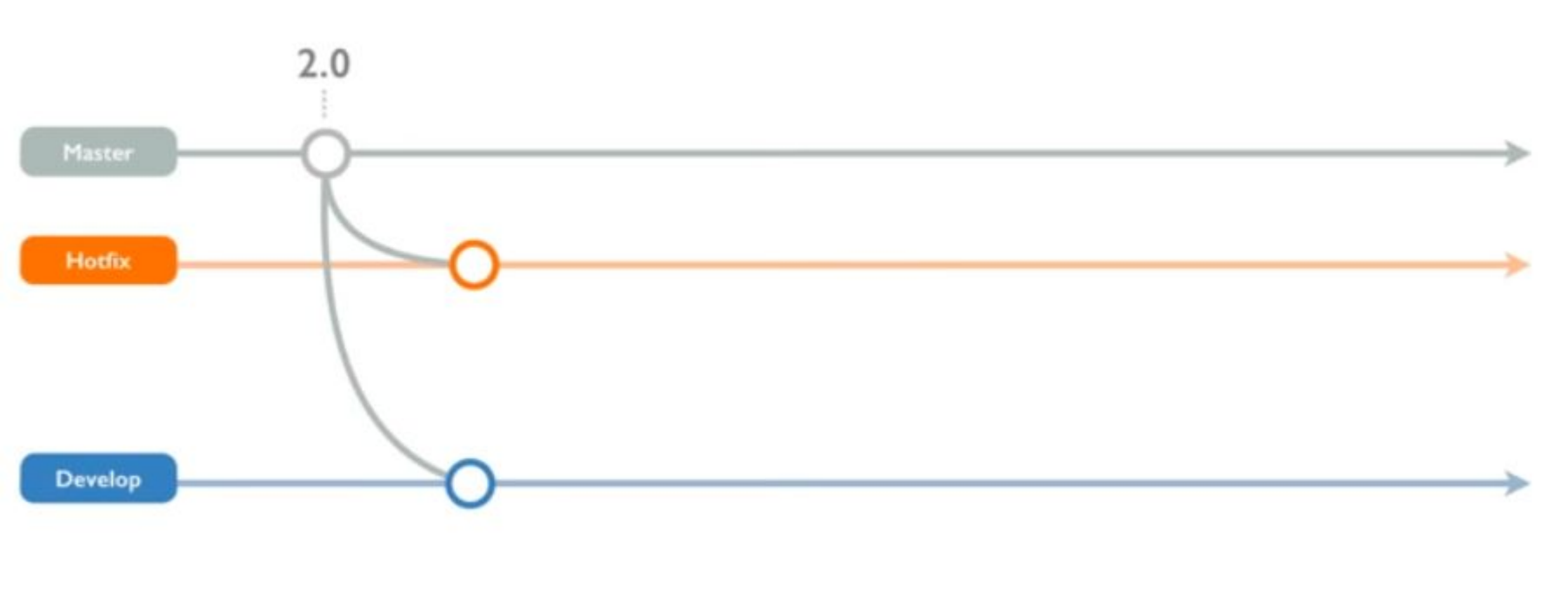


Trabalhando com Ramos (Mesclagem Otimista)

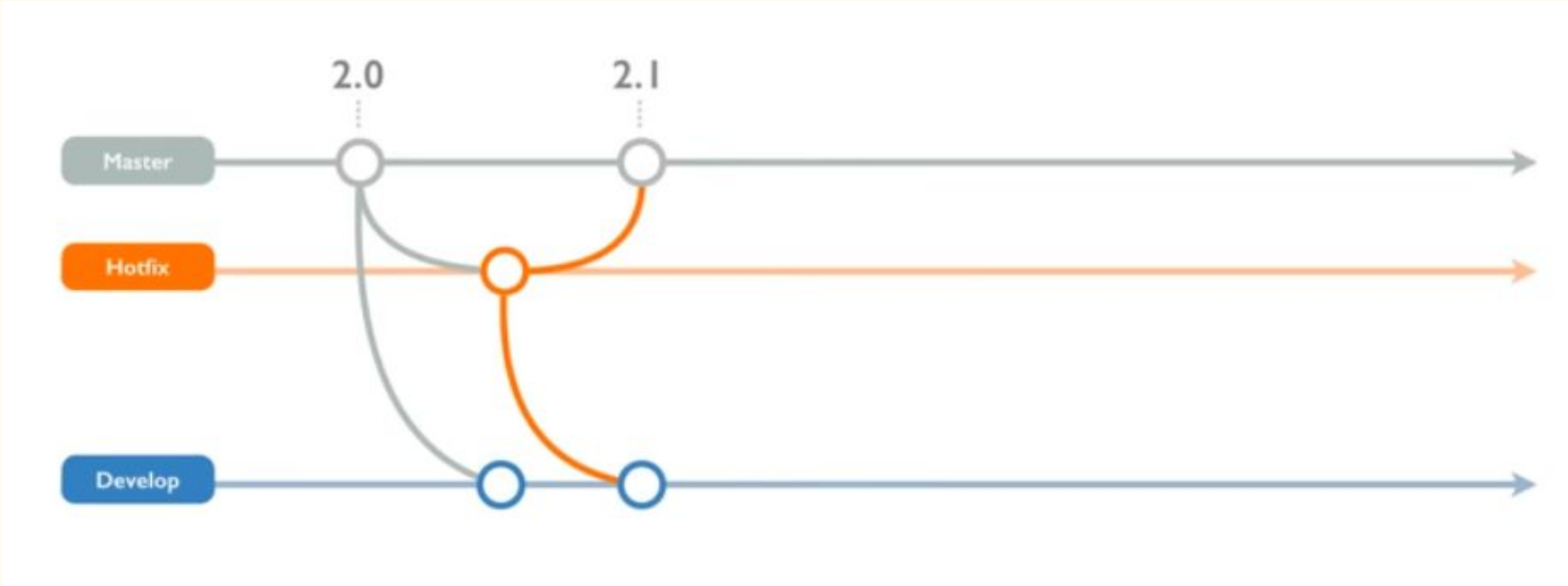


Linhas de trabalho em paralelo com branches de desenvolvimento

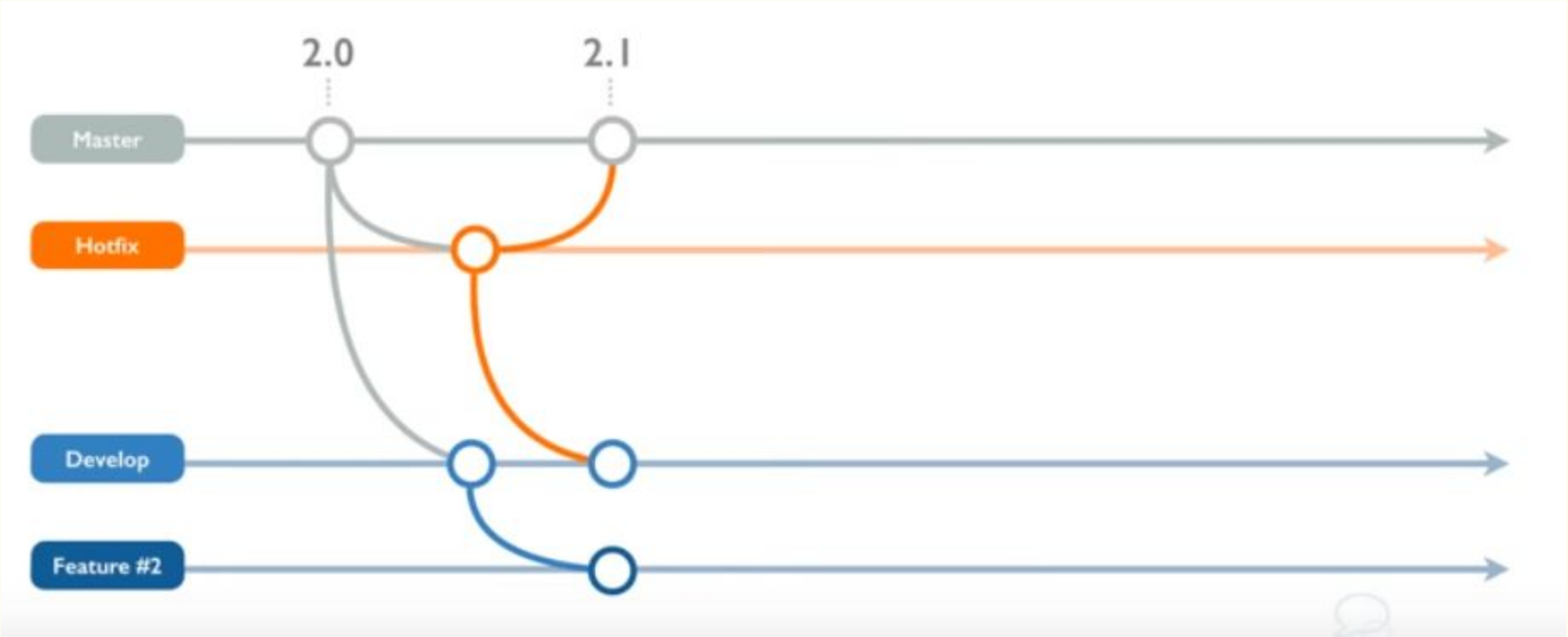
Linhas de trabalho em paralelo com branches de desenvolvimento



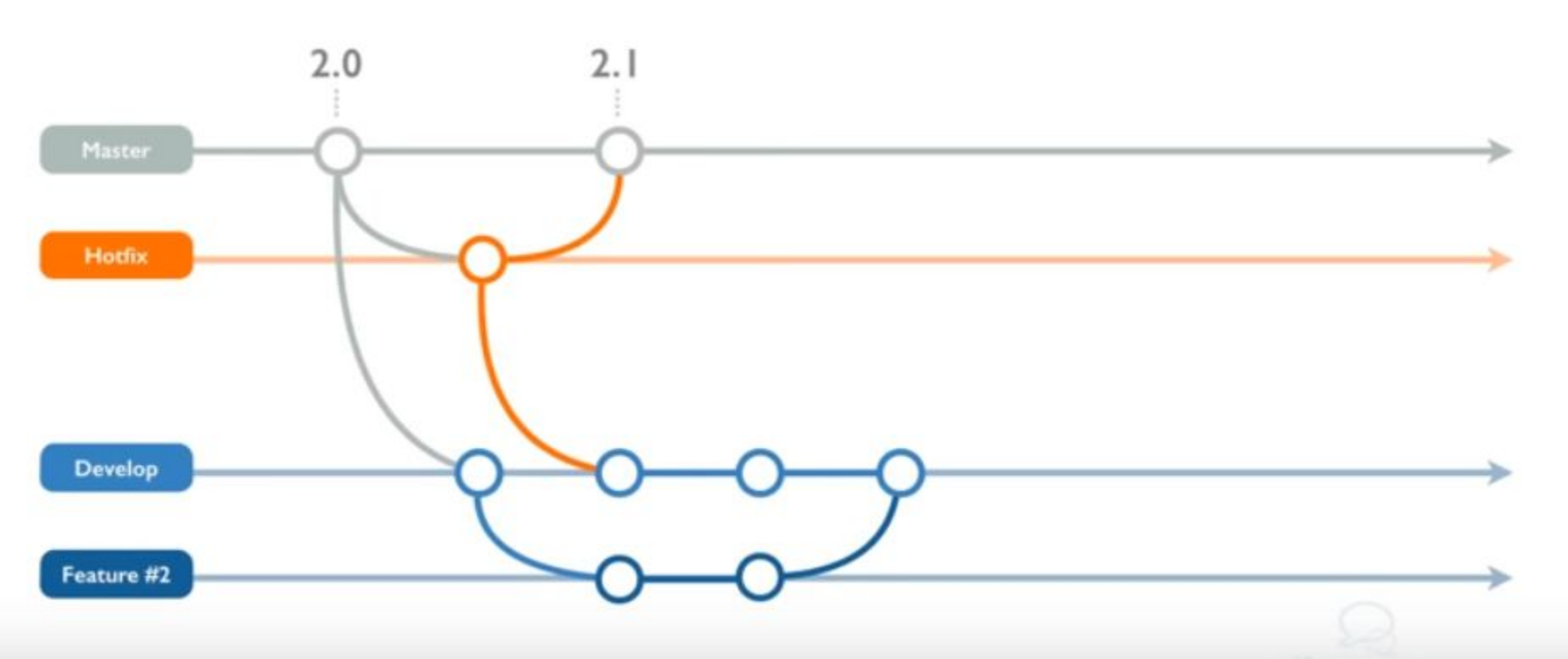
Linhas de trabalho em paralelo com branches de desenvolvimento



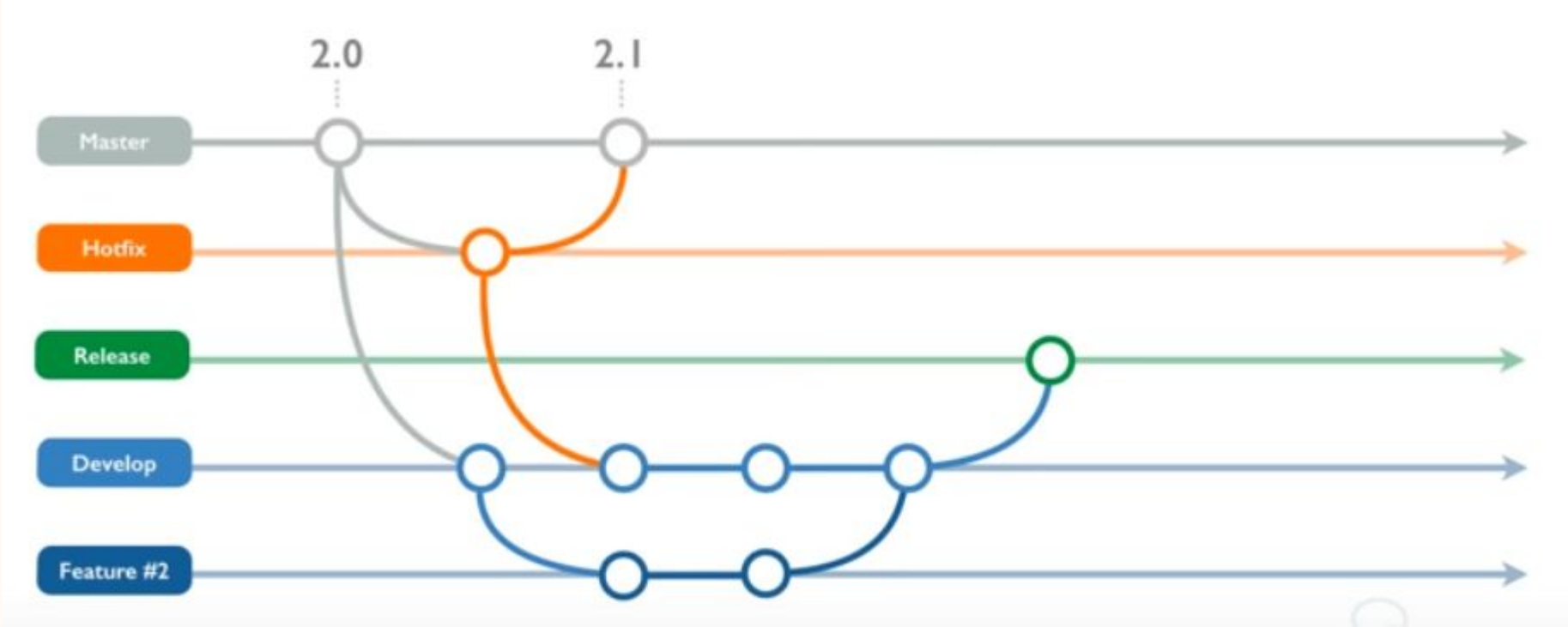
Linhas de trabalho em paralelo com branches de desenvolvimento



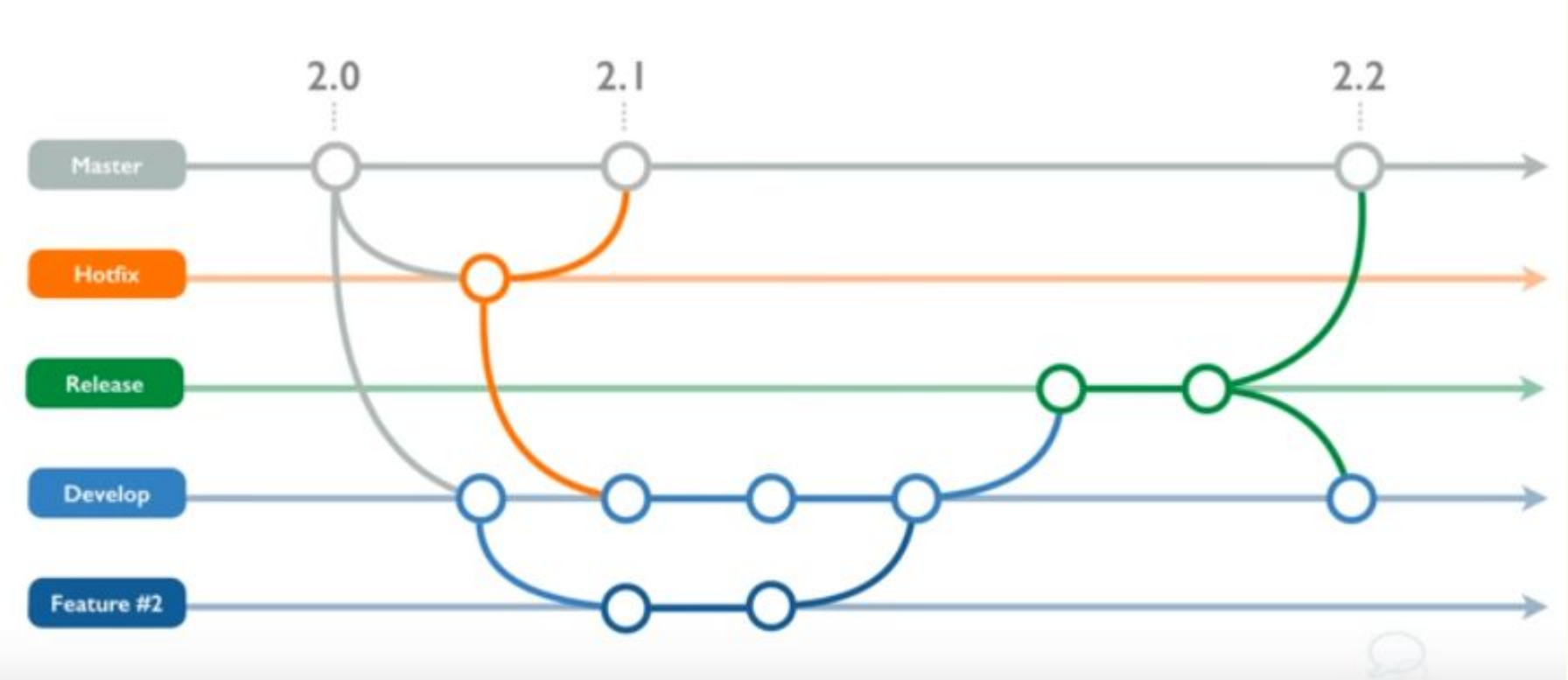
Linhas de trabalho em paralelo com branches de desenvolvimento



Linhas de trabalho em paralelo com branches de desenvolvimento



Linhas de trabalho em paralelo com branches de desenvolvimento



Próxima Aula - Práticas com Github