WEB11 (GIT) — AULA 4

CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO WEB COM *FRAMEWORKS* MODERNOS

UTFPR

PROF. RUI PIMENTEL LEITE

FLUXOS DE TRABALHO

- Na aula anterior, vimos:
  - Como desenvolver repositórios complexos
  - Como fazer operações com *branches*
  - Como subir nosso código para um servidor remoto
  - Como introduzir desenvolvedores na equipe
- Mas como se trabalha no mercado, na prática?

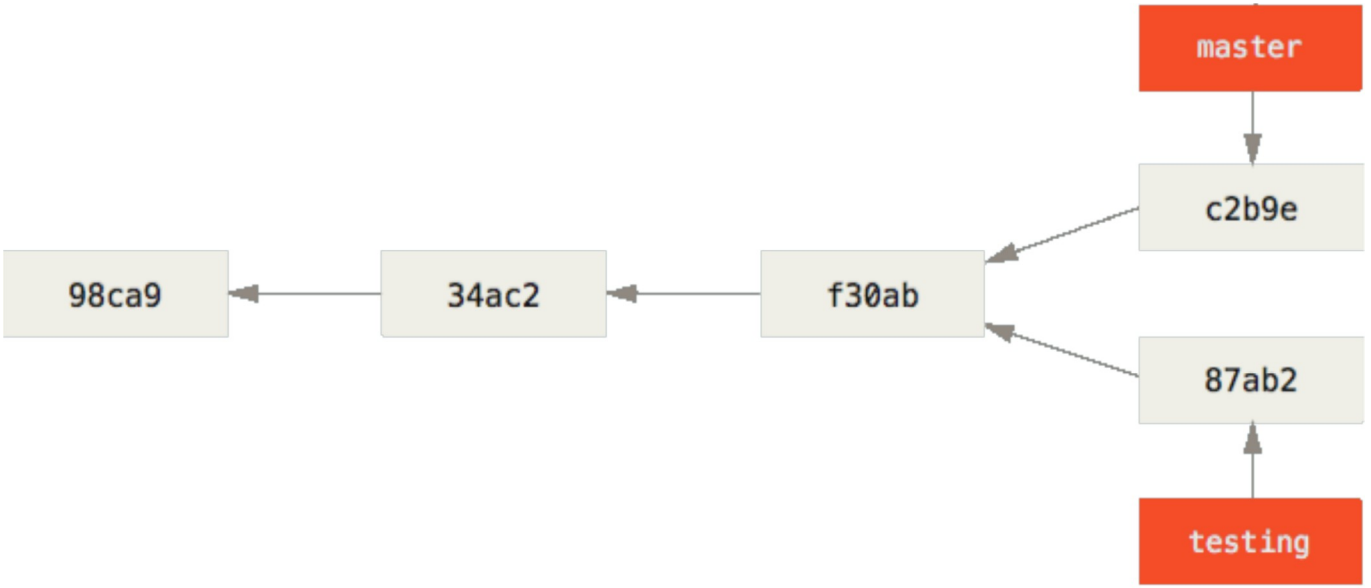
FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **individual**:
  - Fluxo **linear**: mais simples de todos os fluxos. O desenvolvedor utiliza apenas o *branch* `master`

Adaptado de https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **individual**:
  - Fluxo **não–linear**: o desenvolvedor alterna entre os *branches* `master` e outros referentes a tentativas de implementação de recursos, por exemplo



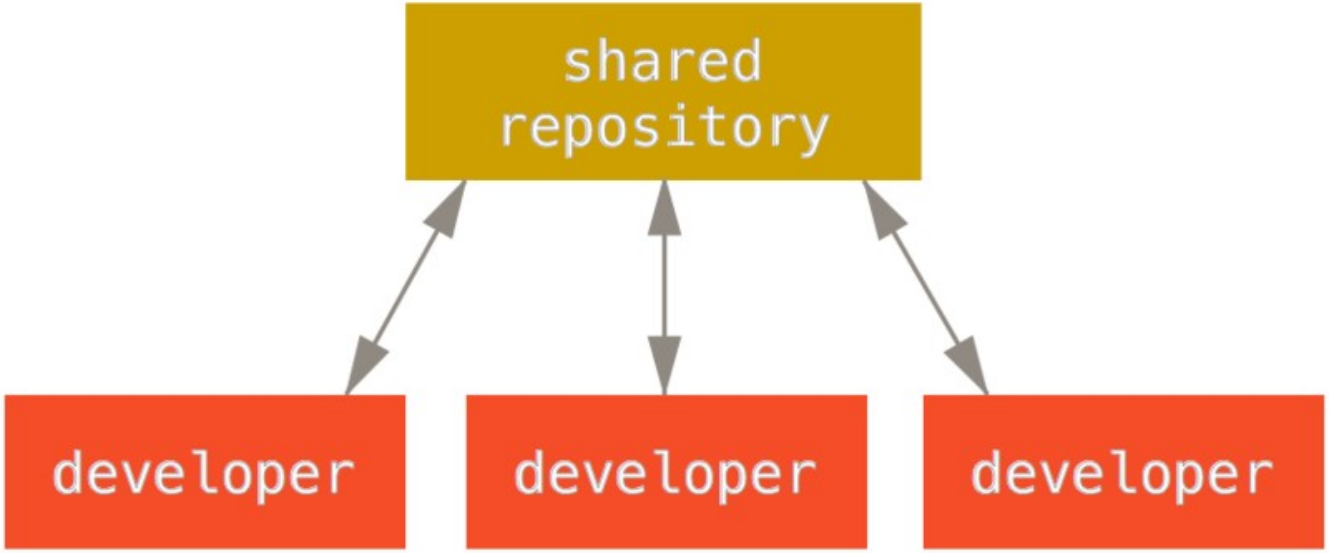
Adaptado de https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **individual**:
  - Como permitir colaboração por parte de novos integrantes no projeto?
  - Como rastrear (e manter!) as *releases*?

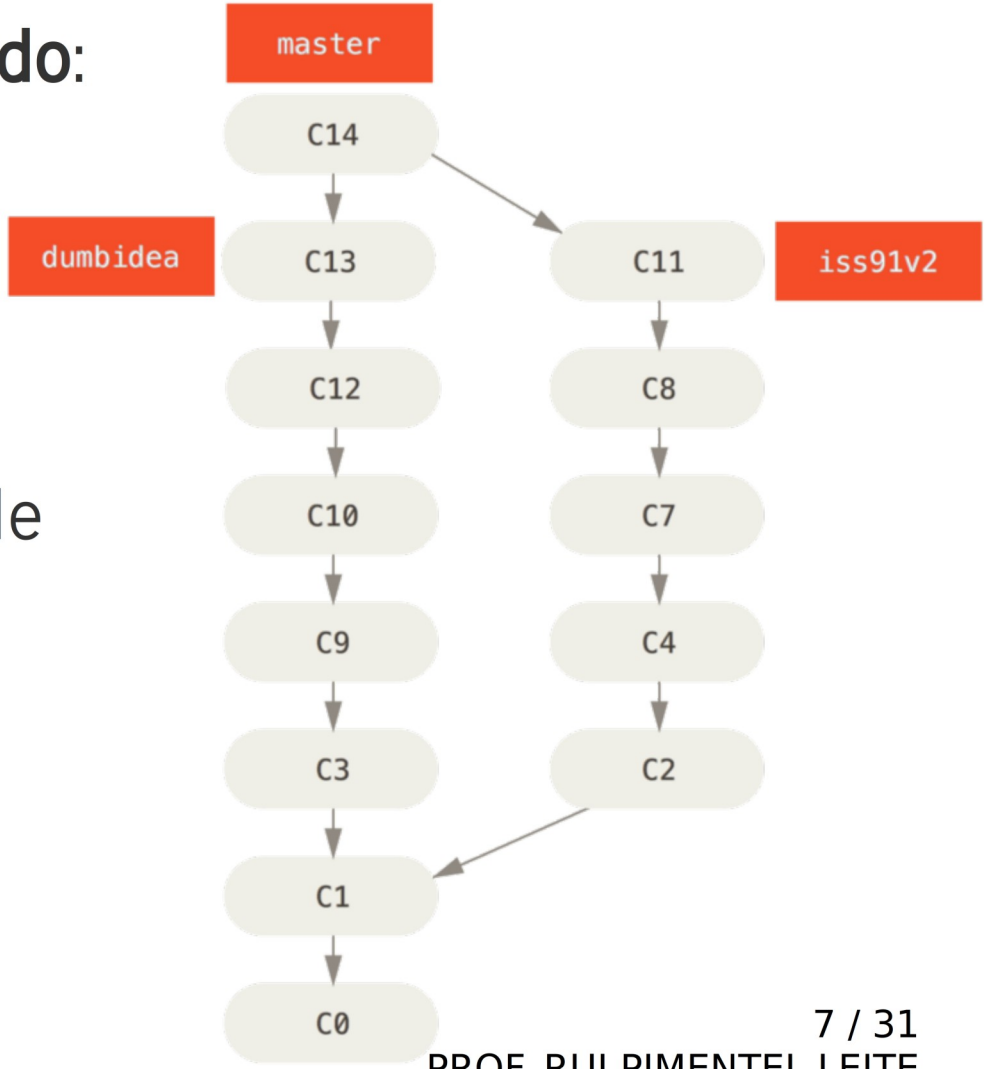
FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Centralizado**: o mais simples dentre os fluxos colaborativos



FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Centralizado**: filosofia "quem fizer *push* por último, mescla"
    - Baseado no modelo de ramificação *"long running branch"* no qual **todos podem escrever no** `master`

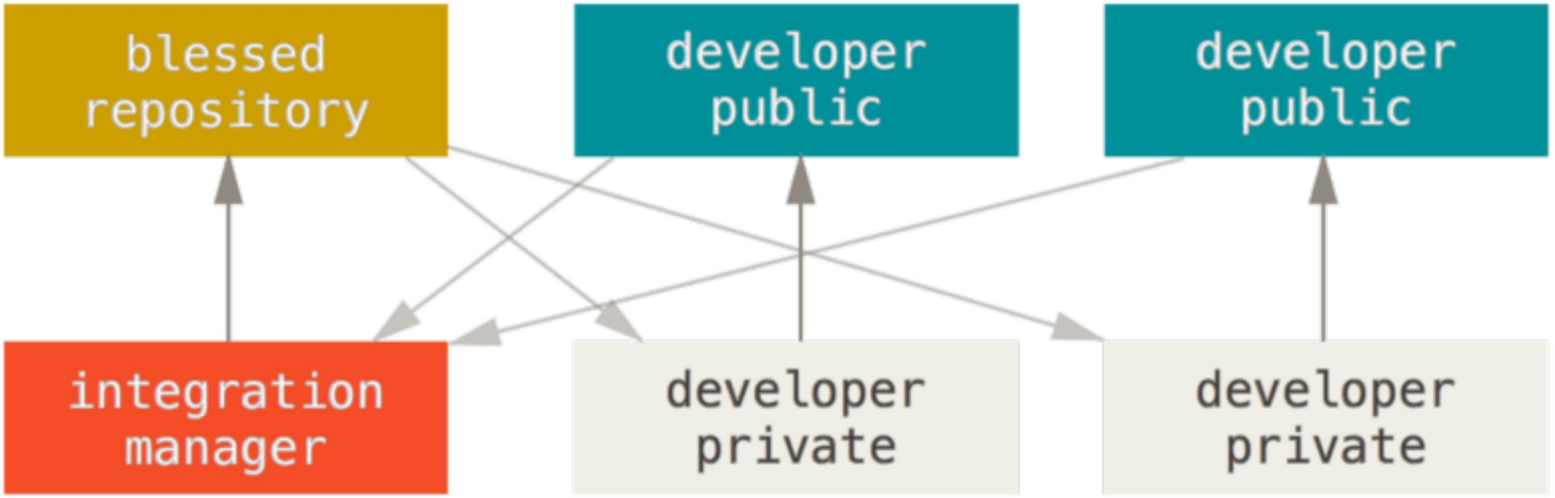


FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Centralizado**:
    - Como evitar *bugs* decorrentes de inexperiência dos novos integrantes da equipe?
    - Como evitar alterações indesejadas?
    - Como rastrear (e manter!) as *releases*?

FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Integração gerenciada**: prevê moderação das contribuições (desenvolvedores não "comitam" diretamente no *branch* `master`)



FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Integração gerenciada**: pode utilizar moderação de *branches* ou de repositórios inteiros
    - Muito comum em ferramentas como o GitHub, o GitLab e similares

FLUXO DE TRABALHO PROFISSIONAL

- Fluxos de trabalho **distribuído**:
  - **Integração gerenciada**:
    - Como rastrear (e manter!) as *releases*?

FLUXO DE TRABALHO PROFISSIONAL

- GitHub *Flow*:
  - Fluxo **distribuído**, **centralizado** ou com **integração gerenciada**
  - Todo *commit* no *branch* `master` está pronto para ser imediatamente deployado
  - Cria-se um *branch* sempre a partir do `master`, com um nome descritivo
  - Todo *commit* no novo *branch* possui uma descrição do por quê esse *commit* foi criado

FLUXO DE TRABALHO PROFISSIONAL

- GitHub *Flow*:
  - A qualquer momento desde a criação do *branch*, abre-se um PR para discutir com os demais colaboradores a respeito da atividade
    - Nesse PR podemos compartilhar *screenshots*, discutir diretrizes de desenvolvimentos das novas funcionalidades e solicitar ajuda e/ou revisão de código.

FLUXO DE TRABALHO PROFISSIONAL

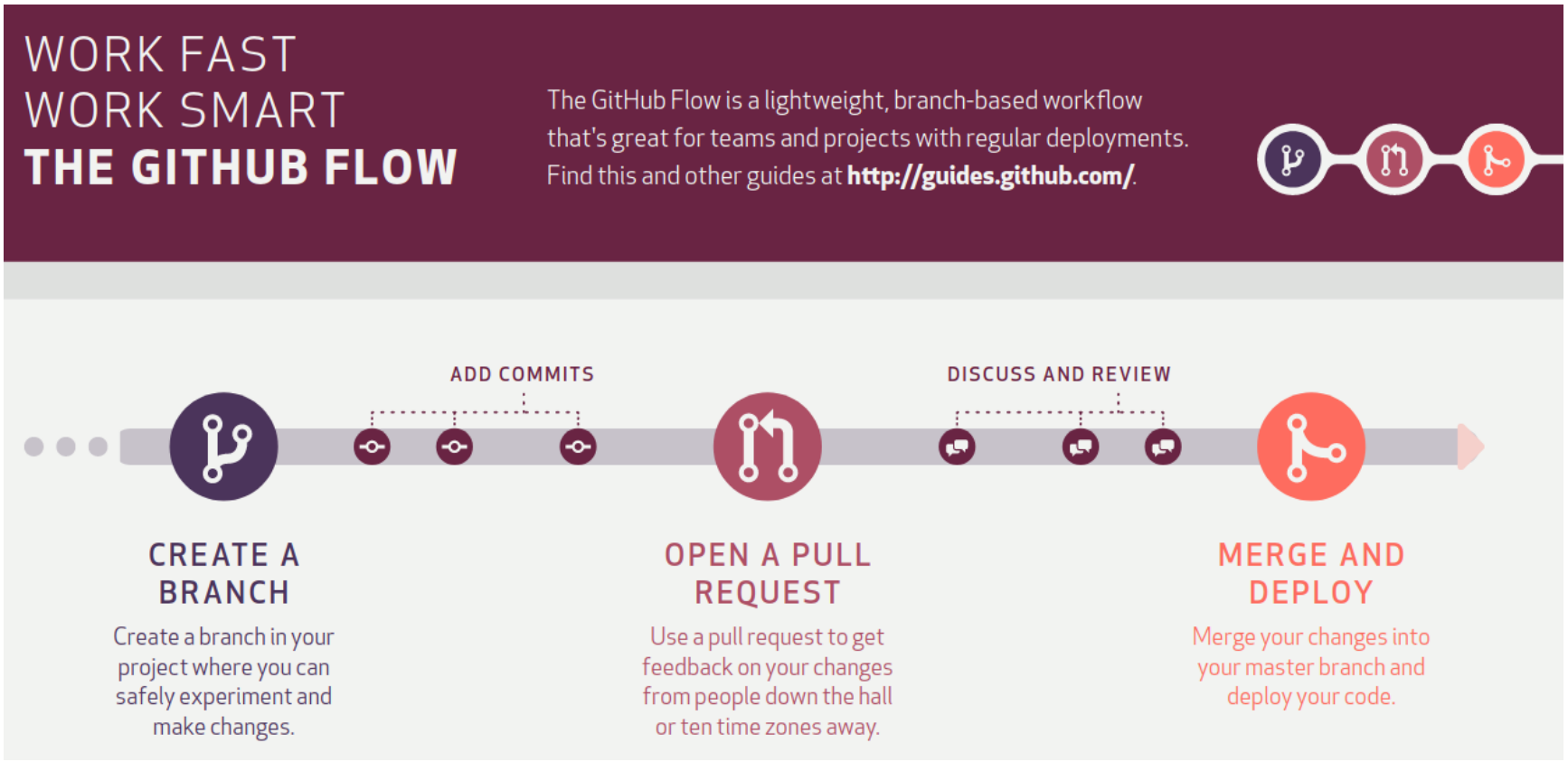
- GitHub *Flow*:
  - O desenvolvedor é estimulado a continuar a usar o mesmo *branch* para novos *commits* com correções, melhorias e novas funcionalidades relacionadas à atividade
    - Cada novo *commit* (inclusive de *merge*) é automaticamente exibido na página do PR após *push*

FLUXO DE TRABALHO PROFISSIONAL

- GitHub *Flow*:
  - Uma vez estabilizado e testado, o PR é aceito (o que equivale a ser mesclado no `master` por um colaborador privilegiado)
  - É indicado para projetos com entrega contínua, isto é, repositórios nos quais cada nova funcionalidade é imediatamente "deployada" (sem aguardar uma data de *release*, por exemplo)

FLUXO DE TRABALHO PROFISSIONAL

- GitHub *Flow*:



FLUXO DE TRABALHO PROFISSIONAL

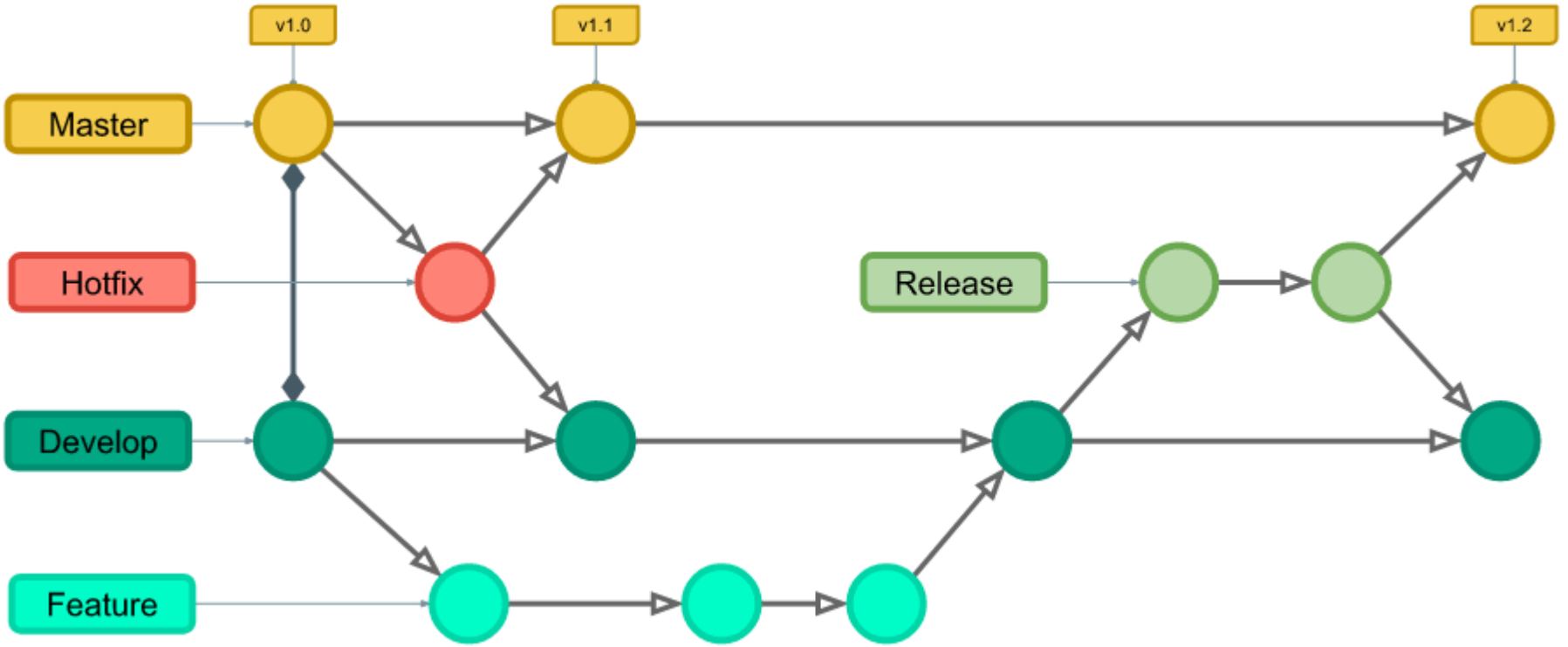
- Git *Flow*:
  - Fluxo **distribuído**, com **integração gerenciada**
  - Mais complexo, pois surgiu com o intuito de gerenciar softwares que demandem manutenção simultânea de várias versões

FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:
  - Prevê a existência de dois *branches* principais e infinitos: `master` e `dev` (sendo este último por vezes também chamado de `develop` ou `development`)
  - `master` contém apenas código de produção; cada novo *commit* nele **deve** ter número de versão
  - `dev` contém o código mais atualizado, ou seja, aquele que pode nem ter sido "deployado" ainda

FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:

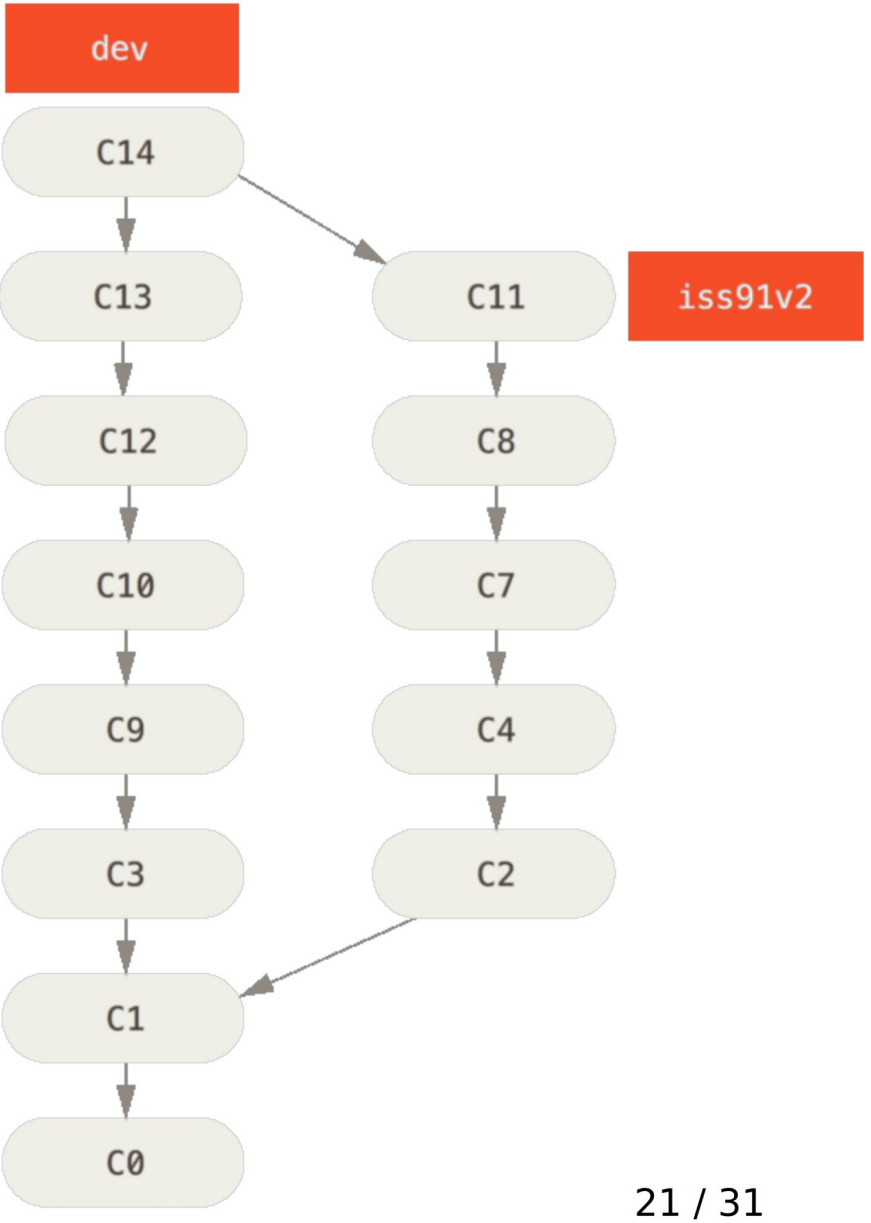


FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:
  - Prevê ainda a existência de três **tipos** de *branches* de apoio
    - Cada tipo ocorre diversas vezes, com cada ocorrência sendo eliminada após uso
    - Cada tipo deriva **ou** do `master`, **ou** do `dev`
  - Cada tipo é mesclado de volta **ou** ao `dev`, **ou** ao `master` **e** o `dev`

FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:
  - *Feature* (*topic*) *branches*:
    - Derivam e se mesclam de volta ao dev
    - Constituem a forma básica de colaboração
    - Geralmente só existem no repositório local

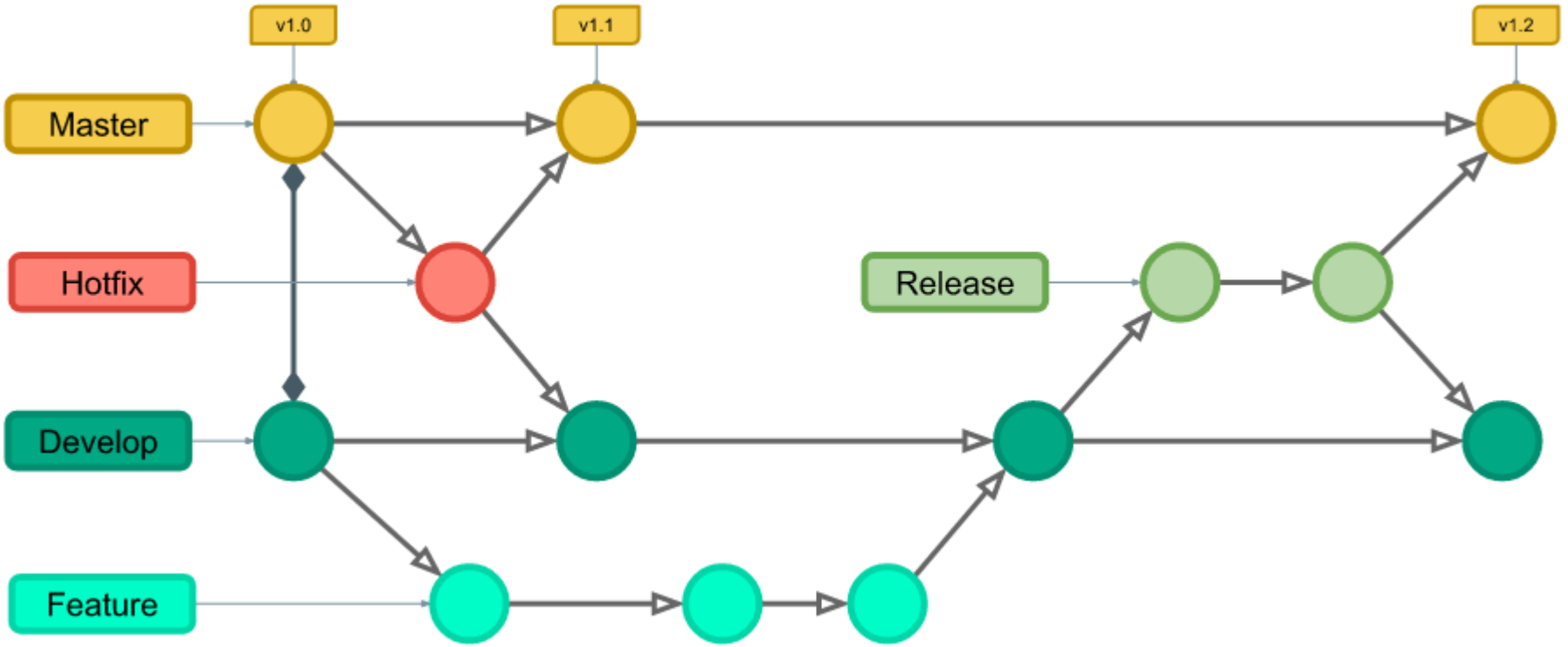


FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:
  - *Hotfix branches*:
    - Derivam do `master`, são mesclados de volta ao `dev` e ao `master`
    - Utilizados para *bugfixes* urgentes, ou seja, correções de versões de produção
    - É importante mesclá–los também com o `dev`, para garantir remoção do *bug* de versões futuras

FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:

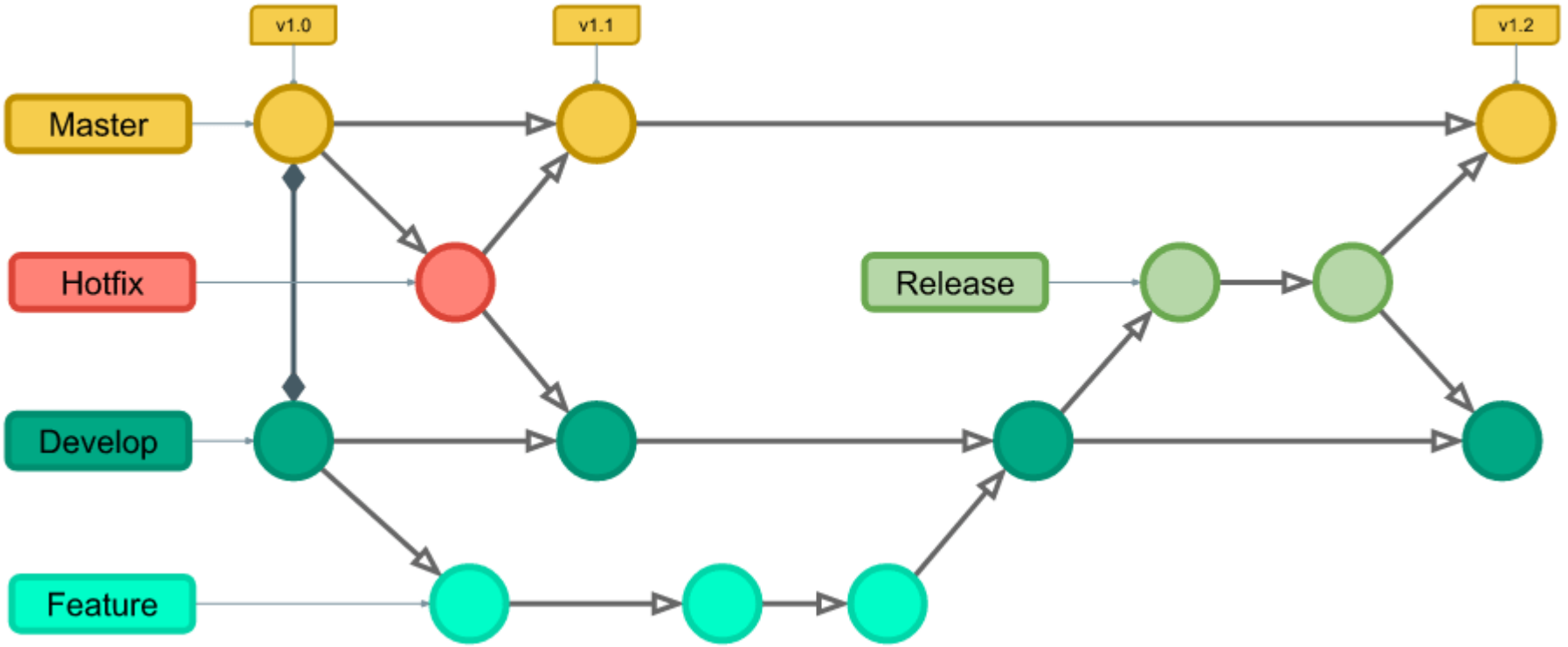


FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:
  - *Release branches*:
    - Derivam do `dev`, são mesclados de volta ao `dev` e ao `master`
    - Utilizados para tarefas relacionadas ao *deploy* (metadados, *bugfixes*)
    - Criados quando todas as funcionalidades para essas *releases* estão basicamente prontas

FLUXO DE TRABALHO PROFISSIONAL

- Git *Flow*:



FLUXO DE TRABALHO PROFISSIONAL

- *Semantic Versioning* (SemVer, ou "versionamento semântico"):
  - Padrão de nomenclatura para versões desenvolvido por um cofundador do GitHub
  - Segmenta o número de versão em ao menos 3 partes (*major*, *minor* e *patch*)
  - O que dita qual dos segmentos mudará a cada versão é a chamada API pública da aplicação

FLUXO DE TRABALHO PROFISSIONAL

- API pública:
  - Uma premissa básica do SemVer é a de que a aplicação sendo construída possui uma API pública
  - O conceito de API é discutível, pois a rigor não inclui interfaces com o usuário, por exemplo
  - Conceito estrito: funcionalidades, mecanismos e comportamentos expostos para outras aplicações

FLUXO DE TRABALHO PROFISSIONAL

- Major.Minor.Patch:
  - *Patch*: aumentado em nova versão que mantenha a API intacta (ex.: *bugfix*)
  - *Minor*: aumentado em nova versão que apenas incremente a API, ou seja, que não remova ou modifique interfaces existentes
  - *Major*: aumentado em nova versão que introduza alterações ou remoções em relação à API anterior

FLUXO DE TRABALHO PROFISSIONAL

- Resumo até agora:
  - O Git permite diversos fluxos de trabalho, sejam eles individuais ou coletivos, com um ou mais *branches*
  - O fluxo de trabalho mais recomendado varia com o ambiente; no entanto fluxos distribuídos com integração moderada, como o Git *Flow*, são os mais utilizados em grandes corporações

FLUXO DE TRABALHO PROFISSIONAL

- Resumo até agora:
  - O controle preciso de dependências do projeto exige organização dos números de versão; o SemVer é uma opção popular e concisa

FLUXO DE TRABALHO PROFISSIONAL

- Já somos capazes de:
  - Colaborar em fluxos de trabalho complexos
  - Versionar nosso software, inclusive atribuindo números de versão semânticos
- O que mais o Git pode fazer pelo desenvolvedor?
- Que técnicas de produtividade são viabilizadas pelo uso do Git?