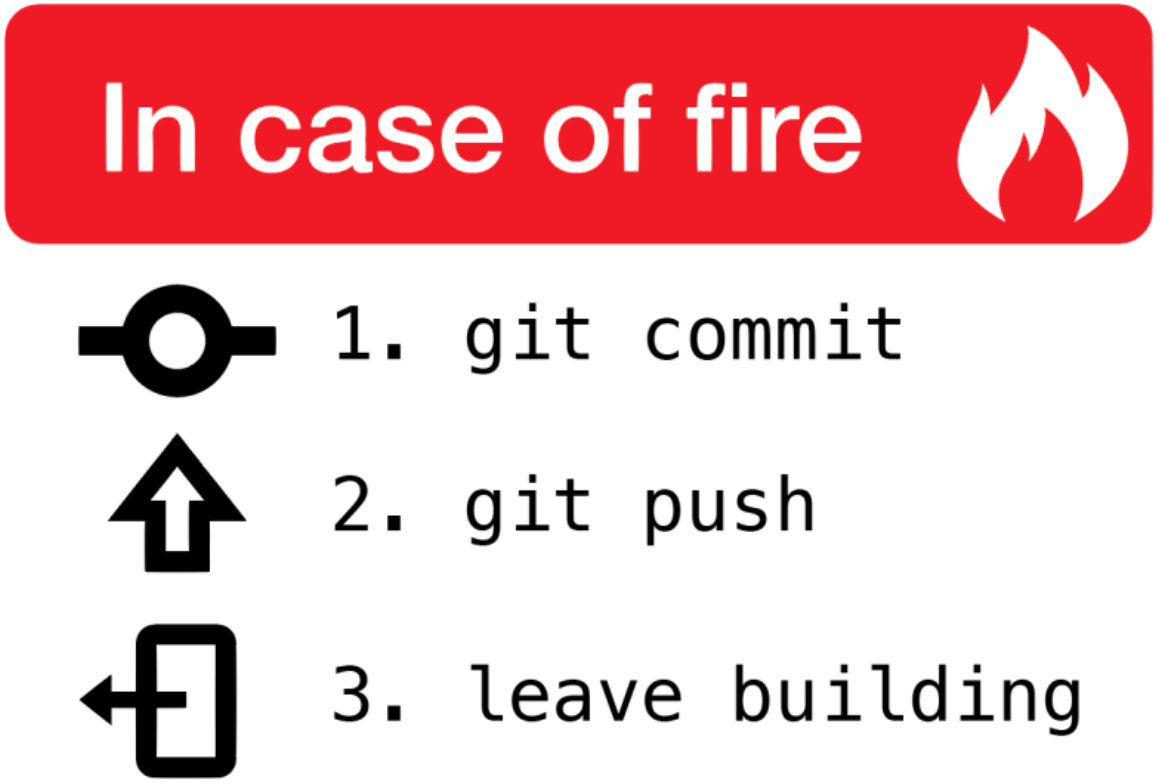
Sistemas de Controle de Versão, Git e GitHub

Paulo Torrens

paulotorrens@gnu.org

Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

Novembro de 2020



Sistemas de Controle de Versão

- Durante o desenvolvimento de *software*, é necessário o **gerenciamento de alterações** feitas em um projeto
- É importante se manter um **registro de modificações** feitas em um projeto, e um registro de por quem foram feitas
- O gerenciamento manual de alterações no projeto se torna extremamente complicado e propenso a falhas
- Sistemas de controle de versão (do inglês, *version control systems*) são sistemas desenvolvidos exatamente com esse objetivo, mantendo históricos de alterações em conjuntos de arquivos e auxiliando o gerenciamento de diferentes versões
- Diversos sistemas se propõe a resolver esse problema, como o SVN, o CVS, o mercurial e o bazaar, e o sistema no qual iremos focar, o **git**

Sistemas de Controle de Versão

- Durante o desenvolvimento de *software*, é necessário o **gerenciamento de alterações** feitas em um projeto
- É importante se manter um **registro de modificações** feitas em um projeto, e um registro de por quem foram feitas
- O gerenciamento manual de alterações no projeto se torna extremamente complicado e propenso a falhas
- Sistemas de controle de versão (do inglês, *version control systems*) são sistemas desenvolvidos exatamente com esse objetivo, mantendo históricos de alterações em conjuntos de arquivos e auxiliando o gerenciamento de diferentes versões
- Diversos sistemas se propõe a resolver esse problema, como o SVN, o CVS, o mercurial e o bazaar, e o sistema no qual iremos focar, o **git**

Sistemas de Controle de Versão

- Durante o desenvolvimento de *software*, é necessário o **gerenciamento de alterações** feitas em um projeto
- É importante se manter um **registro de modificações** feitas em um projeto, e um registro de por quem foram feitas
- O gerenciamento manual de alterações no projeto se torna extremamente complicado e propenso a falhas
- Sistemas de controle de versão (do inglês, *version control systems*) são sistemas desenvolvidos exatamente com esse objetivo, mantendo históricos de alterações em conjuntos de arquivos e auxiliando o gerenciamento de diferentes versões
- Diversos sistemas se propõe a resolver esse problema, como o SVN, o CVS, o mercurial e o bazaar, e o sistema no qual iremos focar, o **git**

Sistemas de Controle de Versão

- Durante o desenvolvimento de *software*, é necessário o **gerenciamento de alterações** feitas em um projeto
- É importante se manter um **registro de modificações** feitas em um projeto, e um registro de por quem foram feitas
- O gerenciamento manual de alterações no projeto se torna extremamente complicado e propenso a falhas
- Sistemas de controle de versão (do inglês, *version control systems*) são sistemas desenvolvidos exatamente com esse objetivo, mantendo históricos de alterações em conjuntos de arquivos e auxiliando o gerenciamento de diferentes versões
- Diversos sistemas se propõe a resolver esse problema, como o SVN, o CVS, o mercurial e o bazaar, e o sistema no qual iremos focar, o **git**

Sistemas de Controle de Versão

- Durante o desenvolvimento de *software*, é necessário o **gerenciamento de alterações** feitas em um projeto
- É importante se manter um **registro de modificações** feitas em um projeto, e um registro de por quem foram feitas
- O gerenciamento manual de alterações no projeto se torna extremamente complicado e propenso a falhas
- Sistemas de controle de versão (do inglês, *version control systems*) são sistemas desenvolvidos exatamente com esse objetivo, mantendo históricos de alterações em conjuntos de arquivos e auxiliando o gerenciamento de diferentes versões
- Diversos sistemas se propõe a resolver esse problema, como o SVN, o CVS, o mercurial e o bazaar, e o sistema no qual iremos focar, o **git**

Introdução ao git

- O git foi criado por Linus Torvalds em 2005, inicialmente para se manter o controle de alterações sobre o kernel do Linux
- Inspirado pelo BitKeeper, um sistema de versionamento proprietário (na época) que era usado para manter o kernel, o git foi publicado como *software* livre e aberto, e rapidamente se tornou popular entre desenvolvedores
- Com plataformas de hospedagem de repositórios como o GitHub, o uso do git se tornou cada vez mais usado, tanto para o uso de projetos livres quanto proprietários
- O processo de *pull request*, existente em sistemas de controle de versão distribuídos, se tornou a forma convencional de se introduzir alterações em projetos de grande porte

Introdução ao git

- O git foi criado por Linus Torvalds em 2005, inicialmente para se manter o controle de alterações sobre o kernel do Linux
- Inspirado pelo BitKeeper, um sistema de versionamento proprietário (na época) que era usado para manter o kernel, o git foi publicado como *software* livre e aberto, e rapidamente se tornou popular entre desenvolvedores
- Com plataformas de hospedagem de repositórios como o GitHub, o uso do git se tornou cada vez mais usado, tanto para o uso de projetos livres quanto proprietários
- O processo de *pull request*, existente em sistemas de controle de versão distribuídos, se tornou a forma convencional de se introduzir alterações em projetos de grande porte

Introdução ao git

- O git foi criado por Linus Torvalds em 2005, inicialmente para se manter o controle de alterações sobre o kernel do Linux
- Inspirado pelo BitKeeper, um sistema de versionamento proprietário (na época) que era usado para manter o kernel, o git foi publicado como *software* livre e aberto, e rapidamente se tornou popular entre desenvolvedores
- Com plataformas de hospedagem de repositórios como o GitHub, o uso do git se tornou cada vez mais usado, tanto para o uso de projetos livres quanto proprietários
- O processo de *pull request*, existente em sistemas de controle de versão distribuídos, se tornou a forma convencional de se introduzir alterações em projetos de grande porte

Introdução ao git

- O git foi criado por Linus Torvalds em 2005, inicialmente para se manter o controle de alterações sobre o kernel do Linux
- Inspirado pelo BitKeeper, um sistema de versionamento proprietário (na época) que era usado para manter o kernel, o git foi publicado como *software* livre e aberto, e rapidamente se tornou popular entre desenvolvedores
- Com plataformas de hospedagem de repositórios como o GitHub, o uso do git se tornou cada vez mais usado, tanto para o uso de projetos livres quanto proprietários
- O processo de *pull request*, existente em sistemas de controle de versão distribuídos, se tornou a forma convencional de se introduzir alterações em projetos de grande porte

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

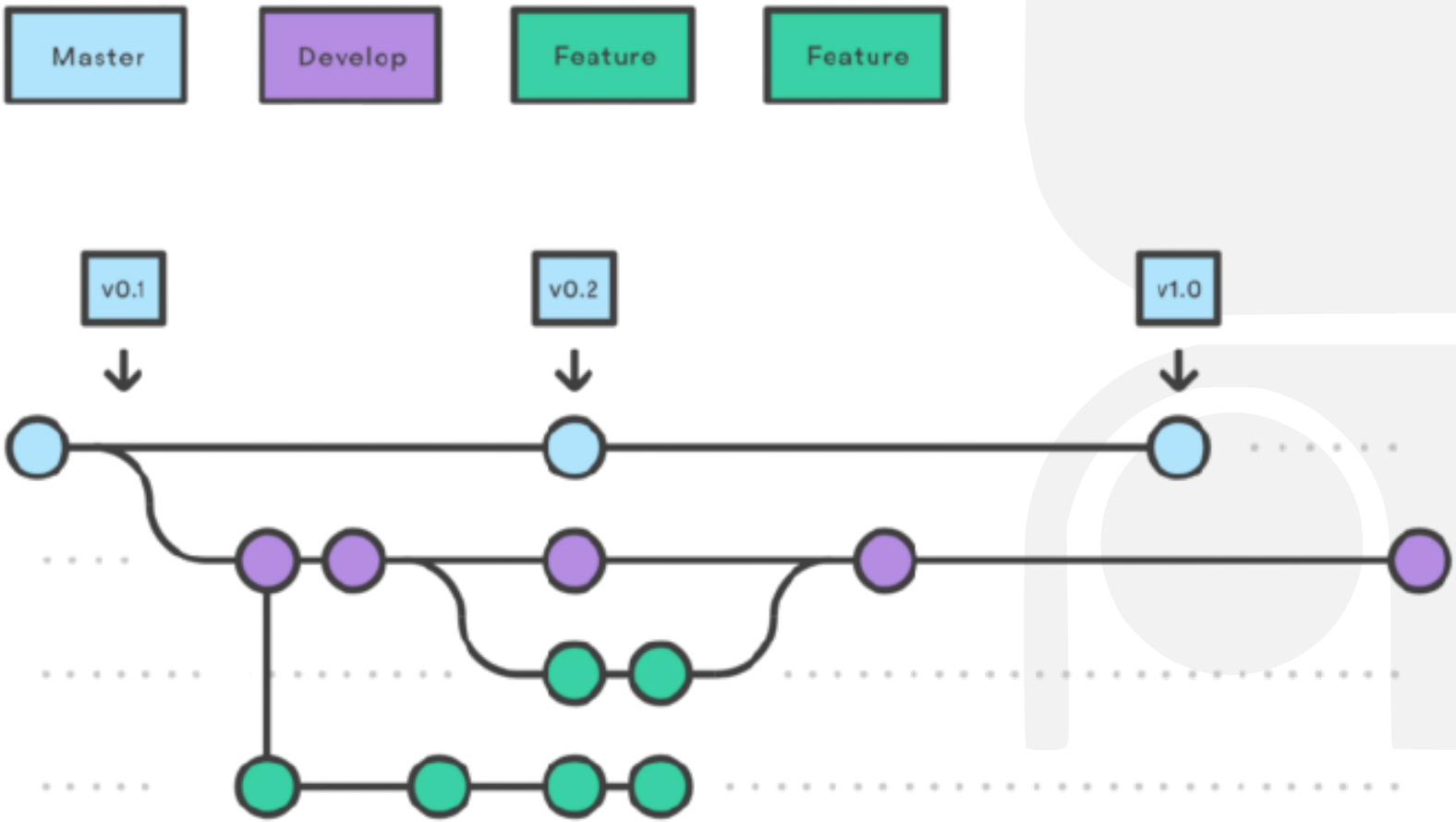
- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Instalando o git

- Embora inicialmente desenvolvido para uso no Linux, o git funciona na maioria das plataformas atuais
- Em sistemas UNIX-like, a forma mais fácil de se instalar o git é através do gerenciador de pacotes local...
  - No Ubuntu, execute: `sudo apt install git`
  - No Fedora, execute: `sudo dnf install git`
  - No macOS, execute: `sudo brew install git`
  - No FreeBSD, execute: `sudo pkg install git`
- Já no Windows...
  - Baixe o instalador: `https://git-scm.com/download/win`
  - Isso irá instalar o git junto a alguns programas úteis
  - Para a maioria das opções oferecidas, você pode simplesmente seguir com a opção padrão selecionada
  - Para o editor de mensagens (*message editor*), cuja utilidade será descrita posteriormente, é recomendável escolher o editor nano ou o VSCode (exceto que você prefira o vim!)

Git: conceitos

- Um projeto no git não possui uma única linha do tempo: cada *branch* representa uma série linear de alterações, podendo divergir e convergir (através do processo de *merge*)



Git: conceitos

- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

Git: conceitos

- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

Git: conceitos

- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

Git: conceitos

- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

Git: conceitos

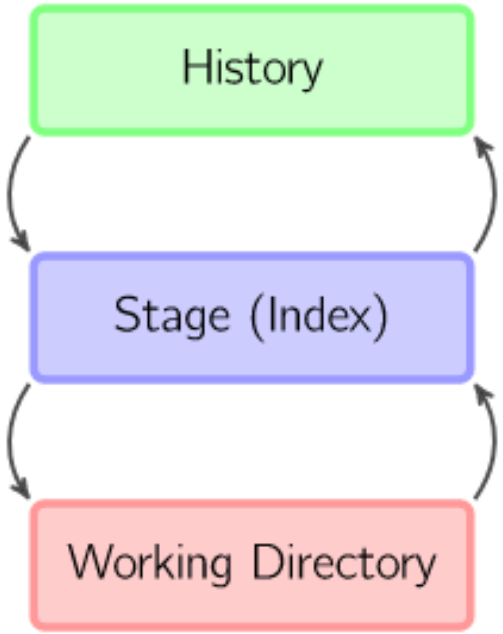
- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

Git: conceitos

- O *branch* principal de um projeto , chamado `master` (ou `main`) representa a versão principal do sistema
- Quando se cria uma nova *branch*, ela diverge do ponto atual na história de uma *branch* de origem, se tornando independente
- É comum a existência de alguns *branches* especiais, nomeadas por convenções
  - Por exemplo, uma *branch* `develop` pode ser usada para controlar o desenvolvimento principal do sistema, com atualizações periódicas para a *branch* `master` dependendo de critérios da equipe
- Quando se deseja introduzir alterações ao projeto (correções de *bugs*, novas *features*, etc), um programador irá criar uma nova *branch* e fará o trabalho nela
- É uma péssima prática alterações diretamente na `master`!

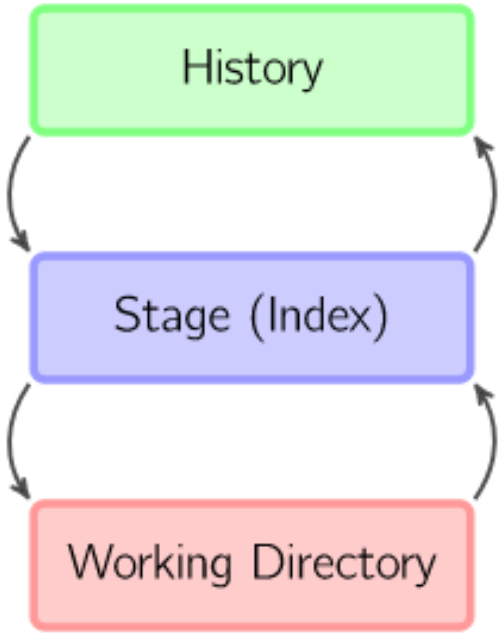
Git: conceitos

- Cada ponto no histórico de uma *branch* é representada por um *commit* (círculos no desenho acima)
- Após alterarmos arquivos no nosso projeto, o git poderá nos informar que as alterações não foram registradas; precisamos movê-las para a área de *staging*
- Ao commitarmos nossas alterações, todos os arquivos em *staging* serão salvos no histórico em um novo *commit*



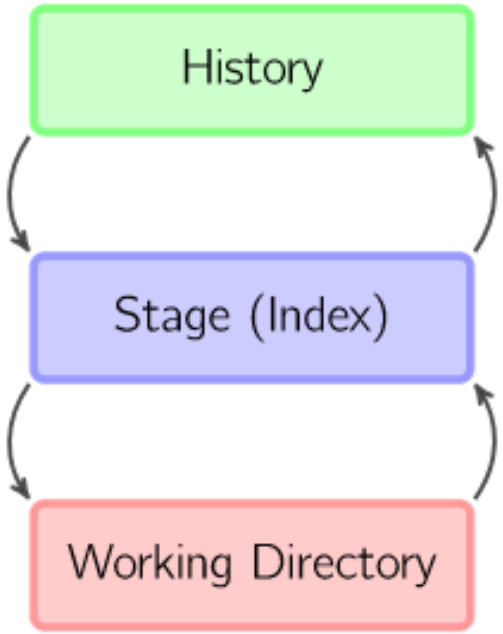
Git: conceitos

- Cada ponto no histórico de uma *branch* é representada por um *commit* (círculos no desenho acima)
- Após alterarmos arquivos no nosso projeto, o git poderá nos informar que as alterações não foram registradas; precisamos movê-las para a área de *staging*
- Ao commitarmos nossas alterações, todos os arquivos em *staging* serão salvos no histórico em um novo *commit*



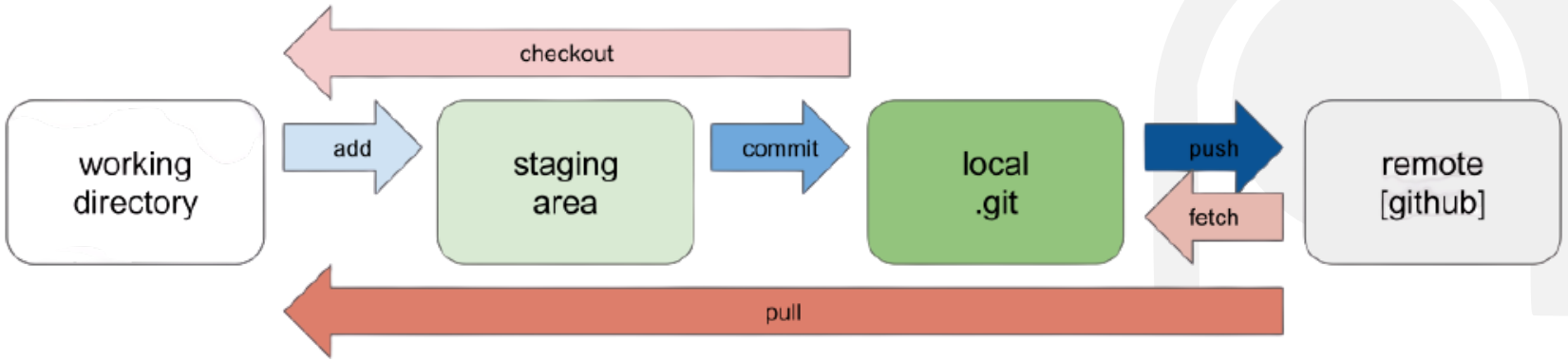
Git: conceitos

- Cada ponto no histórico de uma *branch* é representada por um *commit* (círculos no desenho acima)
- Após alterarmos arquivos no nosso projeto, o git poderá nos informar que as alterações não foram registradas; precisamos movê-las para a área de *staging*
- Ao commitarmos nossas alterações, todos os arquivos em *staging* serão salvos no histórico em um novo *commit*



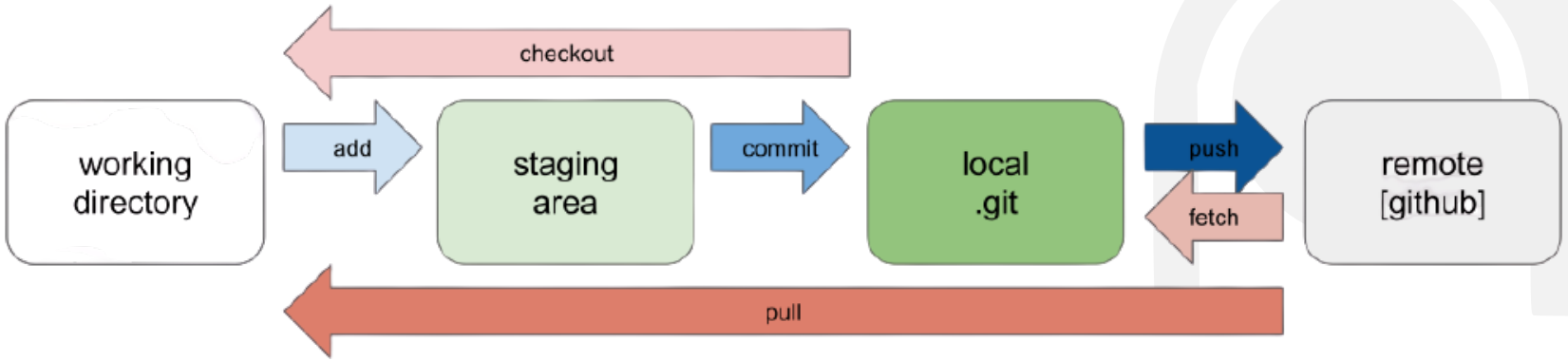
Git: conceitos básicos

- Durante o desenvolvimento, prestando atenção na *branch* em que se está trabalhando, alterações serão constantemente adicionadas em *staging* e commitadas para serem registradas dentro do histórico
  - Recomenda-se que *commits* representem alterações pequenas, porém completas dentro do projeto; ao commitar, você deverá adicionar uma mensagem informando o que foi alterado, e idealmente tais alterações não devem quebrar o projeto
- Para se manter atualizado com versões remotas do repositório, os comandos `push` e `pull` podem ser usados



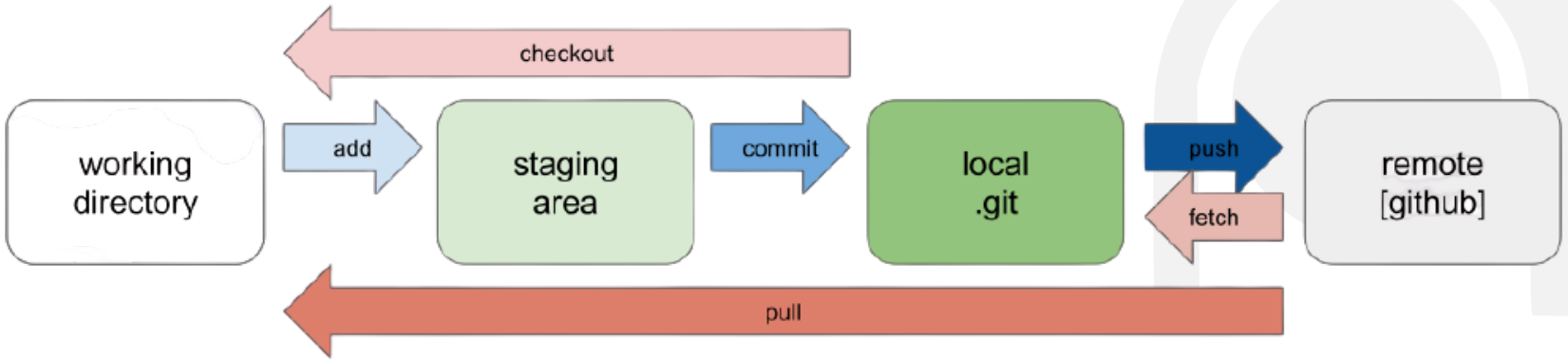
Git: conceitos básicos

- Durante o desenvolvimento, prestando atenção na *branch* em que se está trabalhando, alterações serão constantemente adicionadas em *staging* e commitadas para serem registradas dentro do histórico
  - Recomenda-se que *commits* representem alterações pequenas, porém completas dentro do projeto; ao commitar, você deverá adicionar uma mensagem informando o que foi alterado, e idealmente tais alterações não devem quebrar o projeto
- Para se manter atualizado com versões remotas do repositório, os comandos `push` e `pull` podem ser usados



Git: conceitos básicos

- Durante o desenvolvimento, prestando atenção na *branch* em que se está trabalhando, alterações serão constantemente adicionadas em *staging* e commitadas para serem registradas dentro do histórico
  - Recomenda-se que *commits* representem alterações pequenas, porém completas dentro do projeto; ao commitar, você deverá adicionar uma mensagem informando o que foi alterado, e idealmente tais alterações não devem quebrar o projeto
- Para se manter atualizado com versões remotas do repositório, os comandos `push` e `pull` podem ser usados



Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: comandos básicos

- `init`: criar um novo repositório
- `clone`: clonar um repositório remoto
- `branch`: criar novas *branches*
- `checkout`: trocar a *branch* atual
- `add`: adicionar arquivos ao *staging*
- `commit`: commitar as alterações em *staging*
- `merge`: iniciar processo de *merge*
- `push`: enviar alterações a um servidor remoto
- `pull`: solicitar alterações de um servidor remoto

É possível testar alguns comandos visualmente através de `http://git-school.github.io/visualizing-git/`

Git: recapitulando

- Quando queremos alterar um projeto, primeiro precisamos de uma cópia local (geralmente por `git clone`)
- Divergimos de uma *branch* (geralmente a `master`), criando uma nota *branch* para fazermos nosso trabalho
- Trabalha, trabalha, `git add`, `git commit`... trabalha, trabalha, `git add`, `git commit`... enquanto for necessário
- Podemos fazer o *merge* com a *branch* de origem, juntando as linhas do tempo e introduzindo as alterações
- Caso o servidor principal seja remoto (geralmente é o caso!), precisamos executar o processo de *pull request*...

Git: recapitulando

- Quando queremos alterar um projeto, primeiro precisamos de uma cópia local (geralmente por `git clone`)
- Divergimos de uma *branch* (geralmente a `master`), criando uma nota *branch* para fazermos nosso trabalho
- Trabalha, trabalha, `git add`, `git commit`... trabalha, trabalha, `git add`, `git commit`... enquanto for necessário
- Podemos fazer o *merge* com a *branch* de origem, juntando as linhas do tempo e introduzindo as alterações
- Caso o servidor principal seja remoto (geralmente é o caso!), precisamos executar o processo de *pull request*...

Git: recapitulando

- Quando queremos alterar um projeto, primeiro precisamos de uma cópia local (geralmente por `git clone`)
- Divergimos de uma *branch* (geralmente a `master`), criando uma nota *branch* para fazermos nosso trabalho
- Trabalha, trabalha, `git add`, `git commit`... trabalha, trabalha, `git add`, `git commit`... enquanto for necessário
- Podemos fazer o *merge* com a *branch* de origem, juntando as linhas do tempo e introduzindo as alterações
- Caso o servidor principal seja remoto (geralmente é o caso!), precisamos executar o processo de *pull request*...

Git: recapitulando

- Quando queremos alterar um projeto, primeiro precisamos de uma cópia local (geralmente por `git clone`)
- Divergimos de uma *branch* (geralmente a `master`), criando uma nota *branch* para fazermos nosso trabalho
- Trabalha, trabalha, `git add`, `git commit`... trabalha, trabalha, `git add`, `git commit`... enquanto for necessário
- Podemos fazer o *merge* com a *branch* de origem, juntando as linhas do tempo e introduzindo as alterações
- Caso o servidor principal seja remoto (geralmente é o caso!), precisamos executar o processo de *pull request*...

Git: recapitulando

- Quando queremos alterar um projeto, primeiro precisamos de uma cópia local (geralmente por `git clone`)
- Divergimos de uma *branch* (geralmente a `master`), criando uma nota *branch* para fazermos nosso trabalho
- Trabalha, trabalha, `git add`, `git commit`... trabalha, trabalha, `git add`, `git commit`... enquanto for necessário
- Podemos fazer o *merge* com a *branch* de origem, juntando as linhas do tempo e introduzindo as alterações
- Caso o servidor principal seja remoto (geralmente é o caso!), precisamos executar o processo de *pull request*...

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

GitHub

- O GitHub (distinto do git!), recentemente adquirido pela Microsoft, é uma plataforma online para hospedagem de repositórios, especialmente usada por projetos de software livre
  - Outras plataformas incluem: GitLab, BitBucket, TFS...
- O git é um sistema distribuído: cada usuário mantém uma cópia dos arquivos e do histórico do projeto, porém se costuma designar um ponto central para atualizações (a *origin*)
- Através do GitHub, é possível hospedar projetos pessoais, e contribuir com projetos de outras pessoas e organizações
  - Alguns projetos cujo repositório central é hospedado no GitHub: o kernel Linux, a OpenJDK, o Kubernetes, o VSCode, o Node, a LLVM...
- Além de servir como versão de referência de repositórios, e permitir a discussão entre a equipe dentro da plataforma (as *issues*), o GitHub permite a criação de *pull requests*

Pull Request

- O processo de *pull request* não faz parte do git em si, mas é encontrado em sistemas como o GitHub e se tornou um ponto importante no processo de desenvolvimento de *software*
  - Se você não gosta de deixar o seu terminal, é possível fazê-lo através do *gh*, a linha de comando do GitHub
- Ao invés de se executar um processo de *merge* manualmente para *branches* sensíveis (como a `master`), o *pull request* é uma solicitação para tal
- As plataformas permitem impor restrições e políticas sobre *merge* para *branches* específicas, necessitando da aprovação de membros da equipe
- O intuito do *pull request* é a revisão do código e o controle de qualidade: impedindo que usuários arbitrariamente façam *merge* em algumas *branches*, e exigindo o parecer da equipe

Pull Request

- O processo de *pull request* não faz parte do git em si, mas é encontrado em sistemas como o GitHub e se tornou um ponto importante no processo de desenvolvimento de *software*
  - Se você não gosta de deixar o seu terminal, é possível fazê-lo através do *gh*, a linha de comando do GitHub
- Ao invés de se executar um processo de *merge* manualmente para *branches* sensíveis (como a `master`), o *pull request* é uma solicitação para tal
- As plataformas permitem impor restrições e políticas sobre *merge* para *branches* específicas, necessitando da aprovação de membros da equipe
- O intuito do *pull request* é a revisão do código e o controle de qualidade: impedindo que usuários arbitrariamente façam *merge* em algumas *branches*, e exigindo o parecer da equipe

Pull Request

- O processo de *pull request* não faz parte do git em si, mas é encontrado em sistemas como o GitHub e se tornou um ponto importante no processo de desenvolvimento de *software*
  - Se você não gosta de deixar o seu terminal, é possível fazê-lo através do *gh*, a linha de comando do GitHub
- Ao invés de se executar um processo de *merge* manualmente para *branches* sensíveis (como a `master`), o *pull request* é uma solicitação para tal
- As plataformas permitem impor restrições e políticas sobre *merge* para *branches* específicas, necessitando da aprovação de membros da equipe
- O intuito do *pull request* é a revisão do código e o controle de qualidade: impedindo que usuários arbitrariamente façam *merge* em algumas *branches*, e exigindo o parecer da equipe

Pull Request

- O processo de *pull request* não faz parte do git em si, mas é encontrado em sistemas como o GitHub e se tornou um ponto importante no processo de desenvolvimento de *software*
  - Se você não gosta de deixar o seu terminal, é possível fazê-lo através do *gh*, a linha de comando do GitHub
- Ao invés de se executar um processo de *merge* manualmente para *branches* sensíveis (como a `master`), o *pull request* é uma solicitação para tal
- As plataformas permitem impor restrições e políticas sobre *merge* para *branches* específicas, necessitando da aprovação de membros da equipe
- O intuito do *pull request* é a revisão do código e o controle de qualidade: impedindo que usuários arbitrariamente façam *merge* em algumas *branches*, e exigindo o parecer da equipe

Pull Request

- O processo de *pull request* não faz parte do git em si, mas é encontrado em sistemas como o GitHub e se tornou um ponto importante no processo de desenvolvimento de *software*
  - Se você não gosta de deixar o seu terminal, é possível fazê-lo através do *gh*, a linha de comando do GitHub
- Ao invés de se executar um processo de *merge* manualmente para *branches* sensíveis (como a `master`), o *pull request* é uma solicitação para tal
- As plataformas permitem impor restrições e políticas sobre *merge* para *branches* específicas, necessitando da aprovação de membros da equipe
- O intuito do *pull request* é a revisão do código e o controle de qualidade: impedindo que usuários arbitrariamente façam *merge* em algumas *branches*, e exigindo o parecer da equipe

Dúvidas?