Uma introdução ao git e ao GitHub

Sistema distribuído de controle de versão e rede social de projetos/códigos fontes

Fabrício Cabral
fabricio.cabral@ead.ifpe.edu.br

IFPE

Outubro 2023

Motivação (1/2)

O desenvolvimento de um software é uma atividade de natureza complexa, precisa, colaborativa e evolutiva

- **Complexa**, pois é composta de várias partes que precisam interagir de forma harmoniosa
- **Precisa**, pois a troca de uma operação (“+” por “-”) pode por todo o trabalho a perder
- **Colaborativa**, pois necessita de várias pessoas trabalhando simultâneamente
- **Evolutiva**, pois nasce pequeno e simples e com o passar do tempo torna-se maior e complexo

Motivação (2/2)

Devido a esta natureza, são necessárias ferramentas que auxiliem no processo de codificação, construção, verificação, colaboração e evolução

- Muitas pessoas participam simultâneamente do desenvolvimento
- Qual a razão e quem efetuou a mudança?
- Quais os arquivos e linhas foram modificadas?
- Quando a mudança foi realizada?
- Como desfazer uma mudança específica?
- Qual mudança ocasionou um bug?

Git (1/2)

- Sistema distribuído para Gerenciamento de Código Fonte (SCM)
- Desenvolvido pelo Linus Torvalds para auxiliar no desenvolvimento do kernel do Linux
- Focado em desempenho
- Permite trabalhar offline
  - Conexão com o servidor apenas para compartilhar informações
- Altamente customizável

Git (2/2)

- Integração com a maioria das ferramentas de desenvolvimento
  - Eclipse, NetBeans, IntelliJ IDEA, Visual Studio, Visual Studio Code, Xcode, etc.
- Análises indicam que o git hoje é o SCM mais usado no mundo
- Quem usa os outros SCMs (CVS, SVN, Mercurial, etc.) pretende migrar para o git
- **O git virou o padrão de fato**

Instalação do git no Linux

- Debian e afins (Ubuntu)

```
$ apt-get -y install git-all
```

- Fedora e afins

```
$ yum install git-all
```

Instalação do git no Windows

- Download
- Siga as instruções e configurações padrão do instalador

Glossário (1/2)

- Working directory (Diretório de trabalho)
  - É o conteúdo da pasta do seu projeto onde você trabalha os seus arquivos (edita, adiciona, exclui)
- Staging area ou Index (Área de preparação)
  - Área temporária em que os arquivos/modificações são preparadas para o commit
- Commit
  - É a captura (snapshot) do estado de um projeto em um determinado momento
- Repository ou Repo (Repositório)
  - Local em que os commits são guardados, rastreando as alterações feitas no seu projeto, construindo assim um histórico ao longo do tempo
  - Pode ser local ou remoto

Glossário (2/2)

- Branch (Ramo ou Galho)
  - É um ramo (ou derivação) de um repositório
  - O principal ramo chama-se `main` ou `master`
- Stash
  - Área que salva o atual estado do working directory e da staging area, limpando o working directory
- Tag (Etiqueta)
  - É uma marcação (etiqueta) que demarca um ponto (commit) que representa alguma mudança significativa no seu código, ou seja, uma versão (ou release) do seu projeto

Iniciando o git

1. Configurar o nome e e-mail do desenvolvedor
   1.1 Configurar o proxy se necessário
2. Criar um repositório local do projeto
   2.1 Pode-se baixar um projeto já existente
3. Informar quais arquivos não se deve acompanhar a evolução
4. Fazer o commit inicial

Configurando nome e e-mail

- Configurar o nome e o e-mail do desenvolvedor

```
$ git config --global user.name "Seu␣Nome"
$ git config --global user.email "seu-
    email@provedor.com"
```

- As configurações acima só precisam ser feitas uma única vez por usuário/máquina

Configurando o proxy (HTTPS)

- Informe ao git o usuário, senha, o host / endereço IP e porta do servidor proxy

```
$ git config --global http.proxy
http://usuario:senha@servidorproxy.com:porta
```

- Note que as informações do usuário e senha vão ficar em *plain text* em um arquivo!
- Por segurança (mas não é essencial) restrinja o acesso de leitura/escrita ao arquivo de configuração

```
$ chmod 600 .gitconfig
```

Criando um repositório local

- Novo projeto

```
$ git init
```

- Projeto já existente

```
$ git clone <URI>
```

- Exemplo:

```
$ git clone git@github.com:fabriciofx/teste.git
```

Ignorando arquivos (1/2)

- Não faz sentido acompanhar a evolução de todos os arquivos contidos em um projeto
  - Produto e subproduto da compilação (`*.class`, `*.obj`, `*.exe`, `target/`, etc.)
  - Configuração das IDEs (`.settings`, `.idea`, etc.)
  - Arquivos intermediários gerados pelo LaTeX
  - Imagens geradas
- Criar um arquivo `.gitignore` dentro do seu projeto contendo nomes/padrões dos arquivos
- Construindo o seu `.gitignore`
  - gitignore.io
  - github gitignore

Ignorando arquivos (2/2)

- Exemplo de `.gitignore` para um projeto LaTeX

```
*.aux
*.lof
*.log
*.lot
*.fls
*.out
*.toc
*.fmt
*.fot
*.cb
*.cb2
.*.lb
*.bbl
*.bcf
*.blg
*.fdb_latexmk
*.synctex
*.synctex.gz
*.pdfsync
```

Visualizando o estado

- Mostrar o estado do working directory e da staging area

```
$ git status
```

Workflow básico

1. Adicione (ou modifique) os arquivos
   - Utilize uma IDE, editor de textos ou programa específico
2. Copie as modificações (ou novos arquivos) para a staging area

```
$ git add <arquivo> ou <diretório>
```

   - **Sugestão:** utilize o comando abaixo para adicionar todos os arquivos do diretório corrente:

```
$ git add .
```

3. Efetive as mudanças (commit)

```
$ git commit -m "Mensagem␣do␣commit"
```

Anatomia de um commit

- Partes de um commit (`git log`)

```
commit 04cbdad0faf7bb420071394d872f550ef75803a2
Author: Fabrício Cabral <fbc@meu-provedor.com>
Date:   Fri Oct 13 10:42:57 2023 -0300

    Mensagem do commit
```

  - `commit`: identificador único do commit (SHA-256)
  - `Author`: nome e e-mail do autor do commit
  - `Date`: data e hora que o commit foi realizado
  - `Mensagem`: O que faz ou motivou aquela mudança

Visualizando o histórico

- Mostrar o histórico do repositório local

```
$ git log
```

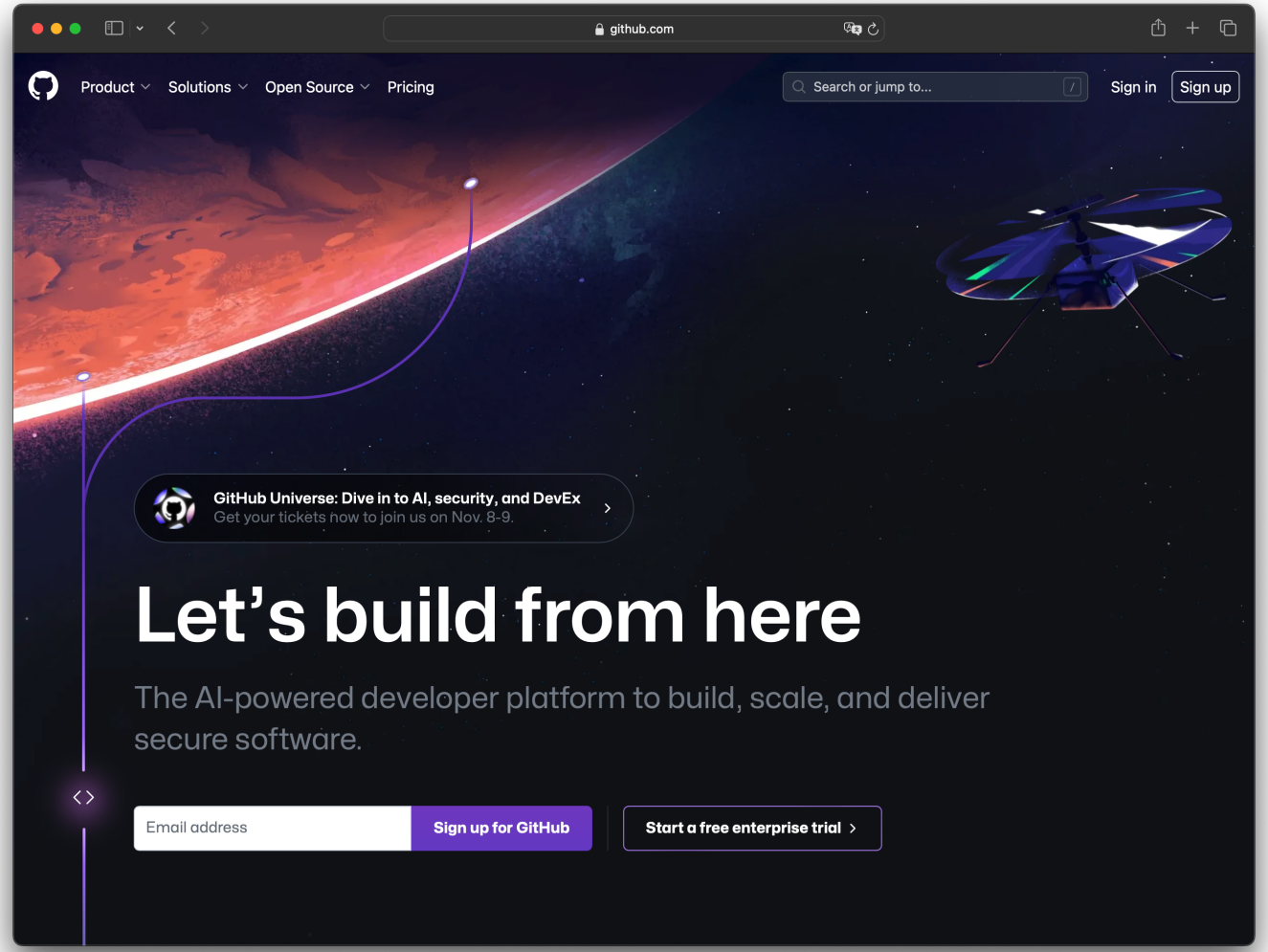
- Mostrar o conteúdo de um commit

```
$ git show <hash-do-commit>
```

- Mostrar o histórico com o conteúdo de cada commit

```
$ git log -p
```

GitHub (1/3)


github.com
Product
Solutions
Open Source
Pricing
Search or jump to...
Sign in
Sign up
GitHub Universe: Dive in to AI, security, and DevEx
Get your tickets now to join us on Nov. 8-9.
Let's build from here
The AI-powered developer platform to build, scale, and deliver secure software.
Email address
Sign up for GitHub
Start a free enterprise trial

GitHub (2/3)

- Rede social para desenvolvedores
- Pessoas ou organizações podem compartilhar o repositório dos seus projetos
- Possui diversos outros recursos
  - Controle de pendências (issue tracker)
  - Controle para aceite de modificações (Pull Request)
  - Controle de projetos
  - Coleção de páginas interligadas e que cada uma delas pode ser visitada e editada por qualquer pessoa (Wiki)

GitHub (3/3)

- Crie uma conta de usuário
- O GitHub pode ser acesso por meio de dois protocolos: SSH ou HTTPS
- Via SSH
  1. Crie um par de chaves SSH (privada e pública)
  2. Adicione sua chave privada ao `ssh-agent`
  3. Adicione sua chave pública ao GitHub
  4. Se você executou o `ssh-agent` só será necessário informar a senha uma única vez
- Via HTTPS
  1. Crie um token de acesso
  2. Utilize esse token como senha

Criando um par de chaves SSH

1. Abra o terminal
2. Execute o comando abaixo

```
$ ssh-keygen -t ed25519 -C "email@example.com"
```

3. Quando perguntado "`Enter a file in which to save the key`" pressione **Enter**
4. Quando solicitado para informar uma *passphrase* digite uma senha (duas vezes)

Adicionando sua chave SSH ao ssh-agent

1. Inicie o `ssh-agent` em modo background

```
$ eval "$(ssh-agent -s)"
```

2. Adicione sua chave SSH privada ao `ssh-agent`

```
$ ssh-add ~/.ssh/id_ed25519
```

3. Adicione sua chave SSH pública a sua conta no GitHub

Adicionando uma chave SSH pública ao GitHub

1. Copie o conteúdo da sua chave SSH pública para a área de transferência
2. Clique na foto do seu perfil e então escolha a opção `Settings` → `SSH e GPG keys`
3. Clique em `New SSH key` ou `Add SSH key`
4. No campo "Title" adicione um nome para a sua chave
5. Selecione o tipo de chave (Authentication Key)
6. No campo "Key" cole sua chave SSH pública
7. Clique em "Add SSH key"

Criando um token de acesso

1. Clique na foto do seu perfil então escolha a opção `Settings` → `Developer settings` → `Personal access tokens` → `Fine-grained tokens`
2. Clique em "Generate new token"
3. No campo "Token name" escolha o nome para seu token
4. No campo "Expiration" escolha por quanto tempo seu token será válido (90 dias)
5. Em "Repository access", escolha `All repositories`

Compartilhando o projeto (1/2)

1. Crie um novo repositório no GitHub
2. Adicione o repositório remoto

```
$ git remote add <repositório-remoto> <URI-do-
    projeto>
```

3. Envie o branch que deseja compartilhar para o repositório remoto

```
$ git push <repositório-remoto> <branch>
```

Compartilhando o projeto: Exemplo

1. Crie um novo repositório no GitHub
2. Adicione o repositório remoto

```
$ git remote add origin git@github.com:
    fabriciofx/teste.git
```

3. Envie o branch que deseja compartilhar para o repositório remoto

```
$ git push origin main
```

Referências

- Página oficial do git
- GitHub Docs