

# Git & GitHub

Alexandre Savelli Bencz

11 de março de 2016



História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

Desenvolvido em 2005 por Linus Torvalds, o mesmo criador do Linux, que estava descontente com o BitKeeper, que era o sistema de controle de versão utilizado no desenvolvimento do kernel do Linux.

História - Git

Instalar e  
Configurar

---

Instando o GIT  
no Linux  
Configurando o  
GIT

Versionando seu  
código

---

Compartilhando  
seu código pelo  
GitHub

---

Se organizando  
com as branches

---

Referencias

---

# Instalar e Configurar

# Instalando o GIT no Linux

História - Git

Instalar e  
Configurar

Instalando o GIT  
no Linux  
Configurando o  
GIT

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

- Para instalar o Git no Ubuntu, ou em outra distribuição baseada em Debian, execute no terminal o seguinte comandos:

```
$ sudo apt-get install git
```

E para quem utiliza Fedora, utilize:

```
$ sudo yum install git
```

# Configurando o GIT

História - Git

Instalar e  
Configurar

Instando o GIT  
no Linux  
Configurando o  
GIT

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

- Sempre que instalamos o GIT, é necessário informar para o GIT, quem somos, para isto usamos as seguintes linhas de comando:

```
$ git config --global user.name "NOME"
```

```
$ git config --global user.email E@MAIL
```

História - Git

Instalar e  
Configurar

---

Versionando seu  
código

---

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

---

Se organizando  
com as branches

---

Referencias

---

# Versionando seu código

# Criando um repositório

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

Git & GitHub

- Para criar e inicializar um repositório, navegue até o diretório onde vai ficar os arquivos do seu projeto e digite o comando:

```
$ git init
```

Após executar o comando, deve aparecer uma mensagem semelhante a esta:

```
Initialized empty Git repository in /*/.git
```

- Com isso já temos um repositório, vazio, do Git inicializado, atente que o sistema criou automaticamente uma pasta chamada .git ( em modo oculto )

# Status do repositório

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

- Quando criamos ou adicionamos um arquivo na pasta onde foi inicializado o projeto o GIT não inclui automaticamente o novo arquivo, para isso é necessario “rastrear” o arquivo e adicionar ele, para isto, utilizamos o comando:

```
$ git status
```

O resultado do comando será algo semelhante a:

```
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Makefile
        doc/
        fonte1.c
        fonte2.asm

        (...)
```



# Rastreando arquivos

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

Git & GitHub

- Para que um arquivo passe a ser rastreado pelo Git, devemos executar o seguinte comando:

```
$ git add <nome do arquivo>
```

Exemplo:

```
$ git add fonte2.asm
```

- Após executar o comando *git status* iremos ver algo semelhante com

```
On branch master
Initial commit
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
```

```
    new file:   fonte2.asm
```

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
Makefile
(...)
```

# Gravando o arquivo no repo.

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

- Para gravarmos as mudanças no repositório, devemos executar o comando:

```
$ git commit -m 'Meu primeiro commit!!!'
```

Observe que o comando *git commit* foi executado junto com o parâmetro *-m*, que é utilizado para definir uma mensagem para o commit que você está submetendo ao servidor.

- **A mensagem do commit deve ser muito clara, ao descrever quais são as modificações que você está enviando para o servidor!**

Após executar o `git commit`, você verá:

```
git commit -m "Meu primeiro commit!!!"
[master (root-commit) ff932de] Meu primeiro commit!!!
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 fonte2.asm
```

# Alterando arquivos

- Se editarmos um arquivo já versionado pelo git, quando executamos o comando *git status*, ele ira nos retornar o seguinte resultado:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working dir)

        modified:   fonte2.asm
```

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

# Verificando as alterações

História - Git

Instalar e  
Configurar

Versionando seu  
código

Criando um  
repositório

Status do  
repositório

Rastreando  
arquivos

Gravando o  
arquivo no repo.

Alterando  
arquivos

Verificando as  
alterações

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

- Podemos verificar o histórico das alterações gravadas no repositório com a seguinte linha de comando:

```
$ git log
```

O resultado do comando vai ser parecido com:

```
$ git log
commit 182279fb11c39d0830825fa0c75366c4a9905c1d
Author: Alexandre <alebencz@gmail.com>
Date:   Thu Mar 10 13:34:30 2016 -0300
```

```
    Corre<C3><A7><C3><A3>o no fonte2.asm e adicionado
    objeto para ser compilado pelo make
```

```
commit ff932ded962e4d2029eba37a879d0886036ea600
Author: Alexandre <alebencz@gmail.com>
Date:   Thu Mar 10 13:11:13 2016 -0300
```

```
    Meu primeiro commit!!!
```

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git

Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referencias

# Compartilhando seu código pelo GitHub

# Git & GitHub ?

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git

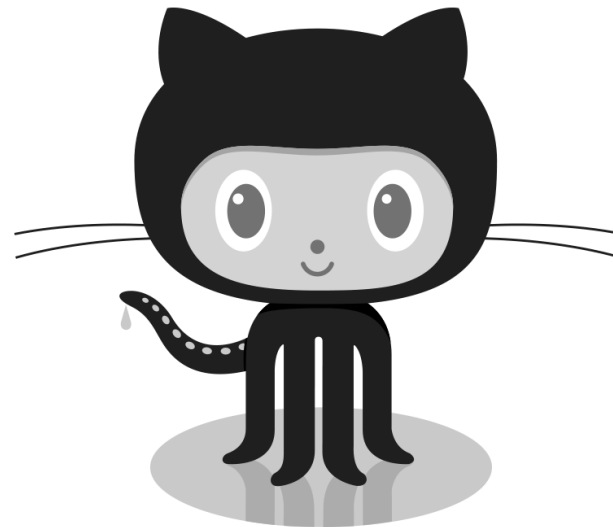
Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referencias

- Git e GitHub são a mesma coisa?
  - *Não. Git é o sistema de controle de versões, com o qual interagimos na linha de comando. Já o GitHub é uma rede social para programadores que disponibiliza repositórios Git acessíveis remotamente.*



# Criando um repositório

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git


Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referencias



Git & GitHub

Owner:  bencz ▾ / Repository name: RepositorioDeEstudo ✓

Great repository names are short and memorable. Need inspiration? How about **reimagined-octo-invention**.

Description (optional)

Descrição do projeto!!

- ☒  **Public**  
Anyone can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

- ☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

# Apontando seu projeto para o Git

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git

Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referencias

- Agora que já temos um repositório criado no GitHub, temos que apontar o nosso repositório criado na nossa máquina para o repositório que foi criado no GitHub. Para isso, use o terminal e navegue até o diretório onde está o repositório que nós criamos e então execute a seguinte linha de comando:

```
$ git remote add origin https://github.com/bencz/RepositorioDeEstudo.git
```



# Enviando para o GitHub

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git

Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referencias

- Agora que estamos com o repositório remoto configurado, podemos enviar as mudanças que fizemos para o GitHub.  
Para isso, é necessário executar o comando seguinte comando:

```
$ git push origin master
```

Com este comando enviamos as alterações para o repositório remoto configurado com o nome *origin*.  
O output do comando vai ser algo semelhante com:

```
Counting objects: 6, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (6/6), 519 bytes | 0 bytes/s, done.  
Total 6 (delta 0), reused 0 (delta 0)  
To https://github.com/bencz/RepositorioDeEstudo.git  
* [new branch]      master -> master
```

# Clonando projetos do GitHub

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Git & GitHub ?  
Criando um  
repositório

Apontando seu  
projeto para o  
Git

Enviando para o  
GitHub

Clonando  
projetos do  
GitHub

Se organizando  
com as branches

Referências

- Para clonar um projeto do GitHub, basta utilizar o seguinte comando:

```
$ git clone <link do projeto>.git
```

Quando executado o comando de clone deverá aparecer algo como:

```
$ git clone https://github.com/bencz/Beryl.git
Cloning into 'Beryl'...
remote: Counting objects: 371, done.
remote: Total 371 (delta 0), reused 0 (delta 0), pack-reused 371
Receiving objects: 100% (371/371), 69.02 KiB | 122.00 KiB/s, done.
Resolving deltas: 100% (248/248), done.
Checking connectivity... done.
```

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge  
Deletando uma  
branch

Referencias

Git & GitHub

# Se organizando com as branches

# Trabalhando em paralelo

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge

Deletando uma  
branch

Referencias

- Muitos sistemas de controle de versão permite trabalho em paralelo através de **branches**
- Mas o que é uma Branch ?
  - Uma **branch** é uma linha independente de desenvolvimento em que podemos enviar novas versões do código sem alterar as outras branches

# Criando uma branch

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge

Deletando uma  
branch

Referencias

Git & GitHub

- Para criar uma branch, basta executar o seguinte comando:

```
$ git branch <nome da nova branch>
```

A execução deste comando não retorna nenhuma resposta, então, para listarmos as branches existentes, utilizamos o comando:

```
$ git branch
```

O resultado deste comando vai retornar a lista de todas as branches existentes para o projeto

```
$ git branch  
    current  
* master
```

# Trocando de branch

- Para trocarmos de uma branch para outra, executamos o seguinte comando:

```
$ git checkout <nome da nova branch>
```

Quando executado este comando, deve aparecer como resposta algo como:

```
Switched to branch 'current'
```

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge  
Deletando uma  
branch

Referencias

# Criar e trocar para uma nova Branch

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge

Deletando uma  
branch

Referencias

- Para uma questão de facilidade, o GIT fornece uma opção para criar e automaticamente trocar de branch, quando executado o comando de checkout, o parametro *-b* deve ser passado antes do nome da nova branch, executando o mesmo comando para trocar de uma branch.

```
$ git checkout -b <nome da nova branch>
```

Após executar este comando, a saída será:

```
$ git checkout -b v1.0
```

```
M      fonte2.asm
```

```
Switched to a new branch 'v1.0'
```

# Commitando para uma branch

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge  
Deletando uma  
branch

Referencias

- Para commitar para uma nova branch, no momento de executar o comando *push*, deve ser informado o nome da branch para qual o commit vai ser enviado:

```
$ git push origin v1.0
```

Após executado o comando, o resultado será algo como:

```
$ git push origin v1.0
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 385 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/bencz/RepositorioDeEstudo.git
 * [new branch]      v1.0 -> v1.0
```



# Fazendo merge

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge

Deletando uma  
branch

Referencias

- Para juntarmos as alterações feita em outras branches com a branch *master*, podemos utilizar o seguinte comando: ( Lembrando que, você deve estar na branch *master* para fazer o merge com os dados da branch *v1.0*

```
$ git merge v1.0 -m 'Fazendo merge com a branch v1.0'
```

O resultado da execução deste comando será:

```
$ git merge v1.0 -m "Fazendo merge com a branch v1.0"
Updating 182279f..456bc06
Fast-forward (no commit created; -m option ignored)
 fonte1.c      | 3 +++
 fonte2.asm    | 2 ++
 2 files changed, 5 insertions(+)
 create mode 100644 fonte1.c
```

Feito isso, basta executar o comando *git push origin master* e os dados vão estar mesclados

# Deletando uma branch

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Trabalhando em  
paralelo

Criando uma  
branch

Trocando de  
branch

Criar e trocar  
para uma nova  
Branch

Commitando  
para uma  
branch

Fazendo merge

Deletando uma  
branch

Referencias

- Para deletarmos uma branch, devemos utilizar a opção *-d* junto ao comando *git branch*.

```
$ git branch -d v1.0
```

O resultado da execução deste comando será:

```
$ git branch -d v1.0
```

```
Deleted branch v1.0 (was 456bc06).
```

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

Links de  
referencia e de  
complemento

Fim!

# Referencias

# Links de referencia e de complemento

História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

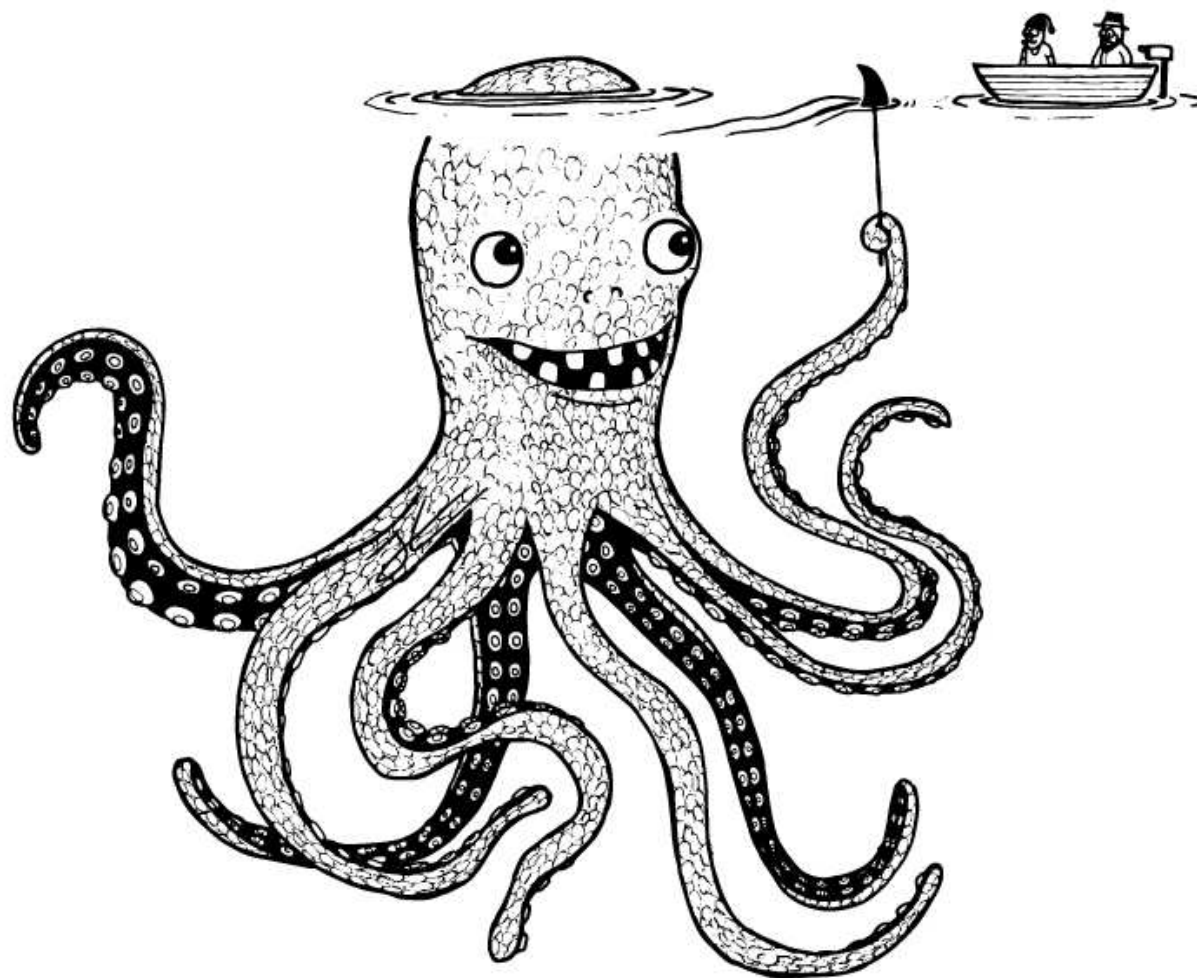
Links de  
referencia e de  
complemento

Fim!

Lista rapida de comandos

Links de varios sites e tutoriais sobre git & github

# Fim!



História - Git

Instalar e  
Configurar

Versionando seu  
código

Compartilhando  
seu código pelo  
GitHub

Se organizando  
com as branches

Referencias

Links de  
referencia e de  
complemento

Fim!