

# Capítulo 2: Instalação e Configuração

*29/10/2015*



# Sumário

<b>1</b>	<b>Instalação e Configuração</b>	<b>5</b>
1.1	Instalação . . . . .	6
1.1.1	Windows . . . . .	6
1.1.2	Linux . . . . .	12
1.1.3	MacOS . . . . .	17
1.2	Configurando Perfil . . . . .	19
1.2.1	Usuário . . . . .	19
1.2.2	Atalhos . . . . .	20
1.2.3	Ignorar Arquivos . . . . .	22



# Capítulo 1

## Instalação e Configuração

Agora, devidamente apresentados ao sistema de versionamento Git vamos utilizá-lo. Porém, antes de começarmos a entender os comandos Git, é necessário sua instalação e configuração. Neste capítulo veremos primeiramente como instalar o programa Git em diferentes sistemas operacionais e posteriormente como configurar algumas opções para viabilizar e facilitar seu uso.

## 1.1 Instalação

### 1.1.1 Windows

Usuários Windows devem visitar [Git for Windows](https://git-for-windows.github.io/)<sup>1</sup>, clicar em “Download” e baixar o arquivo “.exe”.

Após o download, execute o arquivo e você terá a tela conforme figura 1.1:

Como de costume, clique em “Next”. Para dar continuidade a instalação aceite a licença do Git.

O diretório apresentado na figura 1.2 vem como default, porém é possível alterar a instalação para um diretório de sua preferência. Depois de selecionado o caminho da instalação, clique em “Next” para prosseguir.

Na tela de componentes (figura 1.3) podemos definir atalhos, integração ao menu de contexto do Windows Explorer, associação de arquivos e uso de font TrueType. O Git Bash é o prompt de comandos próprio, que além dos comandos Git também fornece alguns comandos Unix que podem ser bem úteis. Já o Git GUI é uma interface gráfica para trabalhar com Git. É recomendável a seleção de ambos os itens.

Depois de selecionado os componentes de sua preferência, clique em “Next” para dar continuidade.

---

<sup>1</sup><https://git-for-windows.github.io/>

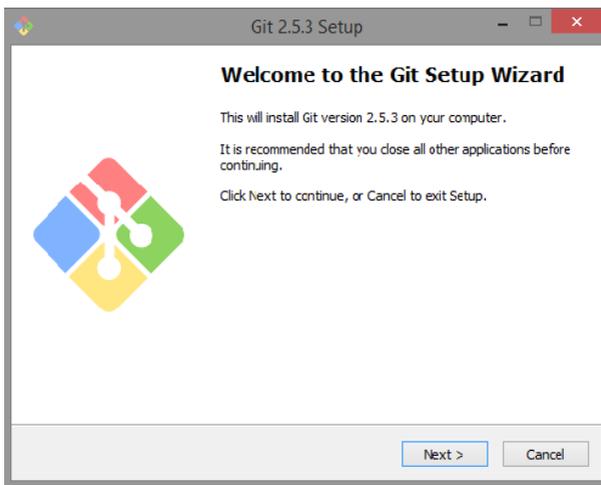
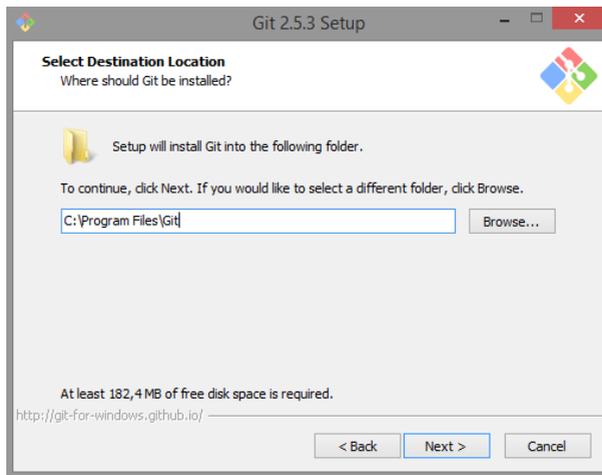
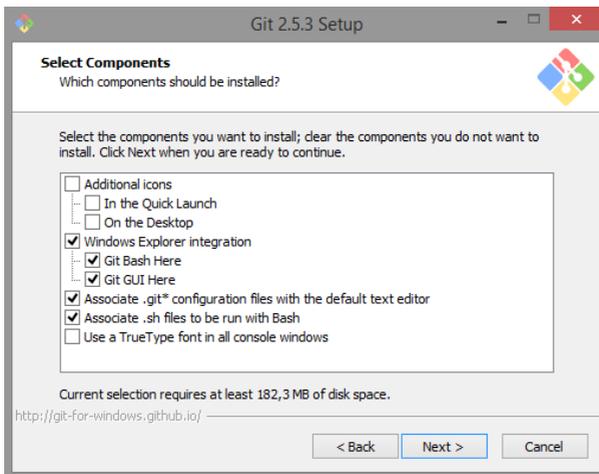


Figura 1.1: *Printscreen* do passo 1

Figura 1.2: *Printscreen* do passo 2

Figura 1.3: *Printscreen* do passo 3

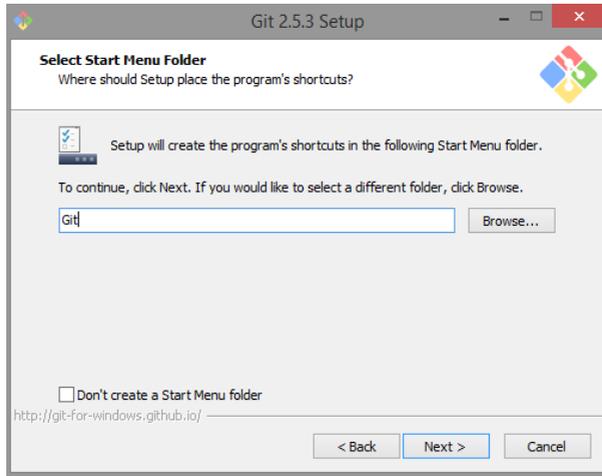
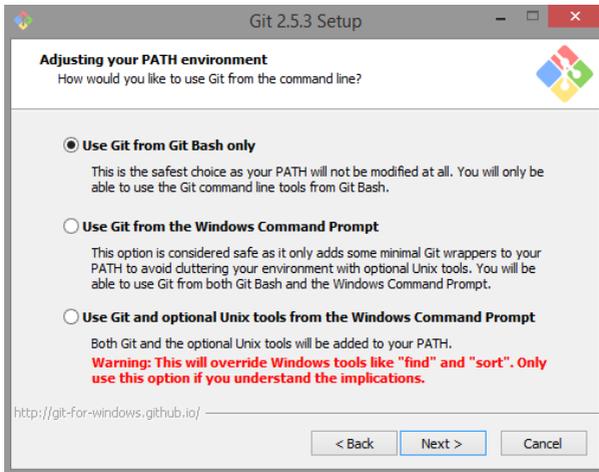


Figura 1.4: *Printscreen* do passo 4

No passo 4, representado pela figura 1.4, o instalador nos oferece a oportunidade de mudar o nome da pasta no menu iniciar, recomenda-se deixar o padrão para fácil localização posteriormente.

Na tela de configuração “PATH environment”, conforme a figura 1.5, podemos escolher as formas de integração do Git com o sistema.

A primeira opção nos permite usar o Git apenas pelo “Git Bash” (é o prompt de comando do Git), a segunda opção nos

Figura 1.5: *Printscreen* do passo 5

possibilita executar os comandos no “Git Bash” e no prompt de comando do Windows (cmd.exe), e a terceira opção é a junção das duas de cima, porém alguns comandos do Windows serão substituídos por comandos Unix com mesmo nome. Essa última opção não é recomendada, a primeira opção é a desejável.

Na figura 1.6, temos a configuração de quebra de linha. Windows e sistemas Unix (Linux, Mac) possuem formatos diferentes de quebra de linha em arquivos de texto. Se você escreve

um código com quebras de linha no formato Windows, outra pessoa pode ter problemas ao abrir o mesmo arquivo em um Linux, e vice-versa. Este passo, portanto, permite normalizar isso.

A primeira opção converte automaticamente os arquivos para padrão Windows quando receber algum arquivo e converterá para padrão Unix quando “comitar” (enviar alterações) ao repositório. A segunda opção, não faz nenhuma conversão ao receber arquivos, mas convertem para padrão Unix ao “comitar”. Já a terceira opção, o Git não fará nenhuma conversão. Recomenda-se a seleção da opção “Checkout Windows-style, commit Unix-Style line endings”.

No passo da figura 1.7, temos a configuração do emulador de terminal para usar com o Git Bash.

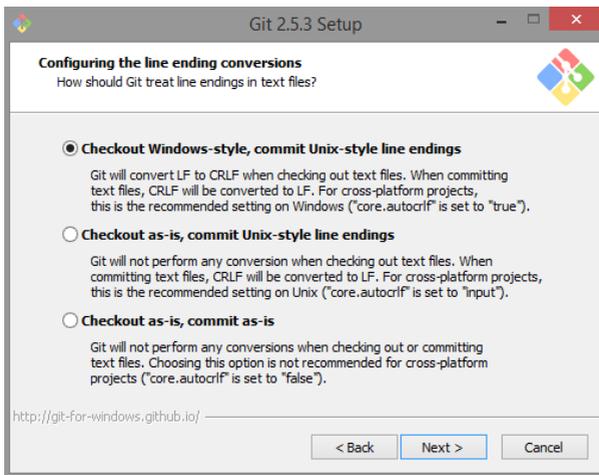
A primeira opção utiliza o terminal MSys2 (Shell), que permite utilizar comandos Unix no Windows. Já a segunda opção, utiliza o terminal padrão do Windows. Recomendamos a primeira opção. Feito isso, dê continuidade a instalação.

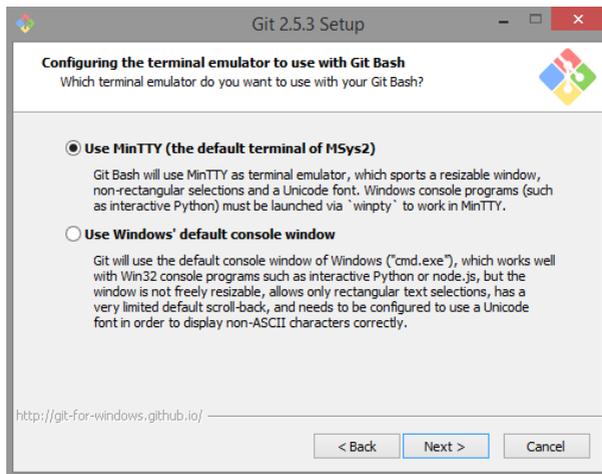
E por último, a figura 1.8, que configura ajustes de performance. Essa opção é para habilitar o sistema de cache de arquivo.

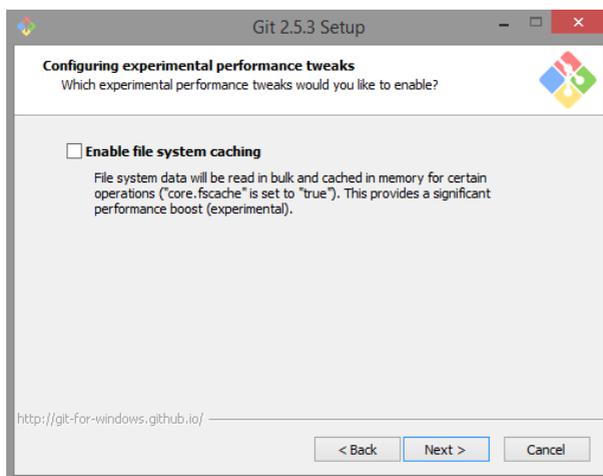
Feito isso, “Next”, “Finish” e o Git está instalado.

### 1.1.2 Linux

Em qualquer sistema Linux, pode-se utilizar o gerenciador de pacotes da respectiva distribuição para instalar o Git. Basta

Figura 1.6: *Printscreen* do passo 6

Figura 1.7: *Printscreen* do passo 7

Figura 1.8: *Printscreen* do passo 8

executar o código de instalação de sua respectiva distribuição.

### Debian

Em uma sessão de terminal Linux de distribuições Debian (Ubuntu, Mint), execute o código abaixo.

Adicione o ppa para obter a versão mais recente do Git.

```
sudo add-apt-repository ppa:git-core/ppa
sudo apt-get update
```

Agora, execute o comando abaixo para instalação do Git. Siga as instruções do prompt de comando, primeiro confirmando a instalação dos pacotes e suas dependências, depois confirmando a instalação do pacote git-core.

```
sudo apt-get install git git-core git-man git-gui git-doc \
ssh openssh-server openssh-client
git --version
```

Para adicionar ferramentas complementares, execute:

```
sudo apt-get install gitk meld
```

### Arch

```
pacman -S git openssh meld
git --version
```

## Fedora

```
Yum install git
git --version
```

Usuários de outra versão do Linux podem visitar Download for Linux<sup>2</sup>.

### 1.1.3 MacOS

Existem duas maneiras de instalar o Git no Mac, uma pelo instalador e outra através do MacPorts.

#### Utilizando o Instalador

O usuário deverá acessar Download for Mac<sup>3</sup>, clicar em “Download” e baixar o arquivo “.dmg”.

Após o download, é necessário clicar duas vezes para ter acesso ao pacote de instalação. Dentro do arquivo “.dmg”, execute o arquivo “.pkg” para iniciar a instalação. Siga os passos até concluir a instalação. É recomendável utilizar a instalação padrão.

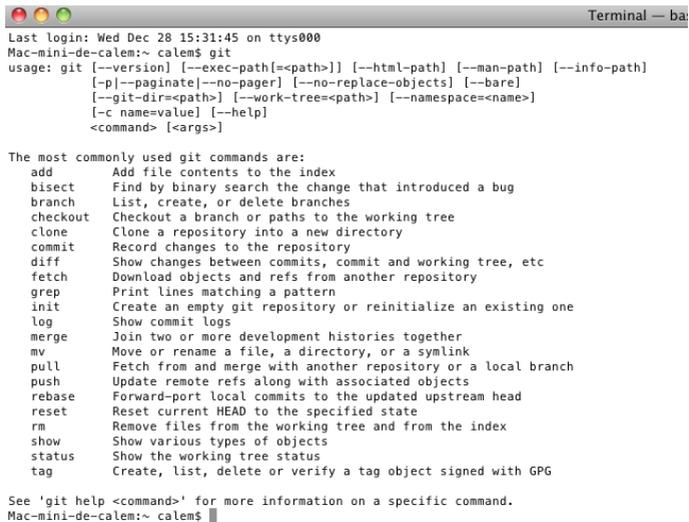
Para testar a instalação, abra o terminal e digite o comando “git”. A saída deverá ser similar a figura 1.9:

#### Utilizando o MacPorts

---

<sup>2</sup><https://git-scm.com/download/linux>

<sup>3</sup><http://git-scm.com/downloads>



```

Last login: Wed Dec 28 15:31:45 on ttys000
Mac-mini-de-calem:~ calem$ git
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [--p|--paginate|--no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        [-c name=value] [--help]
        <command> [<args>]

The most commonly used git commands are:
add          Add file contents to the index
bisect       Find by binary search the change that introduced a bug
branch       List, create, or delete branches
checkout     Checkout a branch or paths to the working tree
clone        Clone a repository into a new directory
commit       Record changes to the repository
diff         Show changes between commits, commit and working tree, etc
fetch        Download objects and refs from another repository
grep         Print lines matching a pattern
init         Create an empty git repository or reinitialize an existing one
log          Show commit logs
merge        Join two or more development histories together
mv           Move or rename a file, a directory, or a symlink
pull         Fetch from and merge with another repository or a local branch
push         Update remote refs along with associated objects
rebase       Forward-port local commits to the updated upstream head
reset        Reset current HEAD to the specified state
rm           Remove files from the working tree and from the index
show         Show various types of objects
status       Show the working tree status
tag          Create, list, delete or verify a tag object signed with GPG

See 'git help <command>' for more information on a specific command.
Mac-mini-de-calem:~ calem$ █

```

Figura 1.9: *Printscreen* do passo 9

A maneira mais fácil de instalar Git no Mac é via MacPorts<sup>4</sup>, para isso basta executar o seguinte comando:

```
sudo port install git-core
```

## 1.2 Configurando Perfil

As configurações vão determinar algumas opções globais do Git, sendo necessário fazê-las apenas uma vez.

### 1.2.1 Usuário

Os comandos abaixo vão configurar o nome de usuário e endereço de e-mail. Esta informação é importante pois é anexada aos commits que você realiza, ou seja, as configurações ficarão associadas ao trabalho em desenvolvimento, permitindo que os colaboradores/gestores do projeto identifiquem suas contribuições. Caso o projeto seja individual, a importância de configurar usuário e e-mail se mantém. Uma vez que se trabalha com duas ou mais máquinas, a maneira de identificar a pessoa que está desenvolvendo o trabalho é pelo nome de usuário.

Em um terminal Bash, execute o código abaixo:

---

<sup>4</sup><http://www.macports.org>

```
git config --global user.name "Knight Rider"  
git config --global user.email "batman@justiceleague.org"
```

A opção `--global` usará essa informação para todo projeto Git da máquina.

É possível fazer definições para cada projeto, ou seja, não globais. Para isso é necessário executar o comando a seguir sem a opção `--global`.

```
git config user.name "Knight Rider"  
git config user.email "batman@justiceleague.org"
```

Uma vez configurado o perfil, o Git está pronto para uso.

## 1.2.2 Atalhos

Os atalhos no Git são chamados de *Alias*. Com ele podemos mapear comandos que repetidamente usamos para algumas poucas teclas. Estes atalhos podem ser criados de dois modos: através do comando no terminal ou editando diretamente no arquivo `/.gitconfig`.

### **Pelo terminal:**

Execute o comando abaixo com o atalho de sua preferência e o nome completo do comando o qual deseja criar o alias.

```
git config --global alias.nome_do_alias "comando inteiro"
```

Um exemplo bem simples é o seguinte:

```
git config --global alias.st "status"
```

Assim, ao executar `git st` é o mesmo que executar `git status`.

Pelo método citado acima, o alias é adicionado automaticamente no seu arquivo `/.gitconfig`.

#### **Pelo arquivo `/.gitconfig`:**

Pode-se criar atalhos através de um bloco no seu arquivo de configuração. Para isso, é necessário localizar o diretório do Git e adicionar a lista de comandos desejada, como no exemplo:

```
[alias]
st = status
ci = commit
br = branch
co = checkout
df = diff
```

Assim que adicionar este bloco com os comandos de sua escolha, ele irá funcionar imediatamente.

Segue abaixo os caminhos para encontrar o arquivo `/.gitconfig` nos sistemas operacionais:

- Windows:
  - 1 - C:\Pasta\_do\_seu\_projeto\.git\config
  - 2 - C:\Documents and Settings\Seu\_usuario\.gitconfig
  - 3 - C:\Arquivos de programas\Git\etc\gitconfig
- Mac:
  - 1 - /Pasta\_do\_seu\_projeto/.git/config
  - 2 - /Users/Seu\_usuario/.gitconfig
  - 3 - /usr/local/git/etc/gitconfig

Obs: Os arquivos de configuração do Git não tem extensão.
- Linux:

Crie um arquivo como *sudo* dentro da pasta *etc/* com nome de *gitconfig* e coloque os atalhos de sua escolha.

Não importa o método você utilize, suas configurações sempre ficarão salvas no arquivo */.gitconfig*.

### 1.2.3 Ignorar Arquivos

Usando o arquivo *.gitignore* podemos ignorar arquivos que não desejamos versionar no repositório, pode ser feito por projeto e por usuário. Configurar um arquivo *.gitignore* antes de começar a trabalhar, é importante, pois evita commits acidentais de arquivos que não deveriam ir para o seu repositório Git.

#### **Ignorar Arquivos por Projeto:**

Em todos os projetos que necessitam de um controle de versão há sempre casos em que arquivos não precisam ser versionados. Para isso é preciso criar um arquivo `.gitignore` no diretório raiz do projeto, o qual contém padrões (pattern) que serão ignorados, cada padrão fica em uma linha como no exemplo:

```
$ cat .gitignore
*. [oa]
*~
```

A primeira linha fala para o Git ignorar qualquer arquivo finalizado em `.o` ou `.a` e a segunda linha ignora todos os arquivos que terminam com um til (`~`). Esses padrões podem ser feitos de acordo com a necessidade de cada projeto.

### **Ignorar Arquivos por Usuário (Globalmente):**

Para não precisar criar uma lista de comandos para serem ignorados em cada projeto, é possível ignorar arquivos em todos os repositórios. Para isso, basta criar um arquivo `.gitignore` em seu diretório *home* contendo os padrões os quais deseja ignorar e executar o comando abaixo no terminal a partir da pasta onde está localizado o arquivo `.gitignore`:

```
git config --global core.excludesfile ~/.gitignore
```

A partir disso, todos os arquivos que estão na lista serão ignorados pelo usuário.

Finalmente com a instalação, configuração essencial (usuário e e-mail) e configurações adicionais concluídos, podemos começar a utilizar o Git para versionar nossos projetos.