



--distributed-is-the-new-centralized

- [About](#)
    - [Branching and Merging](#)
    - [Small and Fast](#)
    - [Distributed](#)
    - [Data Assurance](#)
    - [Staging Area](#)
    - [Free and Open Source](#)
    - [Trademark](#)
  - [Documentation](#)
    - [Reference](#)
    - [Book](#)
    - [Videos](#)
    - [External Links](#)
  - [Downloads](#)
    - [GUI Clients](#)
    - [Logos](#)
  - [Community](#)
- 

This book is available in [English](#).

Full translation available in

[azərbaycan dili](#),  
[български език](#),  
[Deutsch](#),  
[Español](#),  
[Français](#),  
[Ελληνικά](#),  
[日本語](#),  
[한국어](#),  
[Nederlands](#),  
[Русский](#),  
[Slovenščina](#),  
[Tagalog](#),  
[Українська](#)  
[简体中文](#),

Partial translations available in

[Čeština](#),  
[Македонски](#),  
[Polski](#),  
[Српски](#),  
[Ўзбекча](#),  
[繁體中文](#),

Translations started for

[Беларуская](#),  
[فارسی](#),  
[Indonesian](#),  
[Italiano](#),  
[Bahasa Melayu](#),  
[Português \(Brasil\)](#),  
[Português \(Portugal\)](#),  
[Svenska](#),  
[Türkçe](#).

---

The source of this book is [hosted on GitHub](#).  
Patches, suggestions and comments are welcome.

[Chapters ▼](#)

## 1. [1. Começando](#)

- 1. 1.1 [Sobre Controle de Versão](#)
- 2. 1.2 [Uma Breve História do Git](#)
- 3. 1.3 [O Básico do Git](#)
- 4. 1.4 [A Linha de Comando](#)
- 5. 1.5 [Instalando o Git](#)
- 6. 1.6 [Configuração Inicial do Git](#)
- 7. 1.7 [Pedindo Ajuda](#)
- 8. 1.8 [Sumário](#)

## 2. [2. Fundamentos de Git](#)

- 1. 2.1 [Obtendo um Repositório Git](#)
- 2. 2.2 [Gravando Alterações em Seu Repositório](#)
- 3. 2.3 [Vendo o histórico de Commits](#)
- 4. 2.4 [Desfazendo coisas](#)
- 5. 2.5 [Trabalhando de Forma Remota](#)
- 6. 2.6 [Criando Tags](#)
- 7. 2.7 [Apelidos Git](#)
- 8. 2.8 [Sumário](#)

## 3. [3. Git Branching](#)

- 1. 3.1 [Branches in a Nutshell](#)
- 2. 3.2 [Basic Branching and Merging](#)
- 3. 3.3 [Branch Management](#)
- 4. 3.4 [Branching Workflows](#)
- 5. 3.5 [Remote Branches](#)
- 6. 3.6 [Rebasing](#)
- 7. 3.7 [Summary](#)

## 4. [4. Git on the Server](#)

- 1. 4.1 [The Protocols](#)

- 2. 4.2 [Getting Git on a Server](#)
- 3. 4.3 [Generating Your SSH Public Key](#)
- 4. 4.4 [Setting Up the Server](#)
- 5. 4.5 [Git Daemon](#)
- 6. 4.6 [Smart HTTP](#)
- 7. 4.7 [GitWeb](#)
- 8. 4.8 [GitLab](#)
- 9. 4.9 [Third Party Hosted Options](#)
- 10. 4.10 [Summary](#)

## 5. [5. Distributed Git](#)

- 1. 5.1 [Distributed Workflows](#)
- 2. 5.2 [Contributing to a Project](#)
- 3. 5.3 [Maintaining a Project](#)
- 4. 5.4 [Summary](#)

## 1. [6. GitHub](#)

- 1. 6.1 [Configurando uma conta](#)
- 2. 6.2 [Contribuindo em um projeto](#)
- 3. 6.3 [Maintaining a Project](#)
- 4. 6.4 [Managing an organization](#)
- 5. 6.5 [Scripting GitHub](#)
- 6. 6.6 [Summary](#)

## 2. [7. Git Tools](#)

- 1. 7.1 [Revision Selection](#)
- 2. 7.2 [Interactive Staging](#)
- 3. 7.3 [Stashing and Cleaning](#)
- 4. 7.4 [Signing Your Work](#)
- 5. 7.5 [Searching](#)
- 6. 7.6 [Rewriting History](#)
- 7. 7.7 [Reset Demystified](#)
- 8. 7.8 [Advanced Merging](#)
- 9. 7.9 [Rerere](#)
- 10. 7.10 [Debugging with Git](#)
- 11. 7.11 [Submodules](#)
- 12. 7.12 [Bundling](#)
- 13. 7.13 [Replace](#)
- 14. 7.14 [Credential Storage](#)
- 15. 7.15 [Summary](#)

## 3. [8. Customizing Git](#)

- 1. 8.1 [Git Configuration](#)
- 2. 8.2 [Git Attributes](#)
- 3. 8.3 [Git Hooks](#)
- 4. 8.4 [An Example Git-Enforced Policy](#)
- 5. 8.5 [Summary](#)

## 4. [9. Git and Other Systems](#)

1. 9.1 [Git as a Client](#)
2. 9.2 [Migrating to Git](#)
3. 9.3 [Summary](#)

## 5. **10. Git Internals**

1. 10.1 [Plumbing and Porcelain](#)
2. 10.2 [Git Objects](#)
3. 10.3 [Git References](#)
4. 10.4 [Packfiles](#)
5. 10.5 [The Refspec](#)
6. 10.6 [Transfer Protocols](#)
7. 10.7 [Maintenance and Data Recovery](#)
8. 10.8 [Environment Variables](#)
9. 10.9 [Summary](#)

## 1. **A1. Appendix A: Git in Other Environments**

1. A1.1 [Graphical Interfaces](#)
2. A1.2 [Git in Visual Studio](#)
3. A1.3 [Git in Eclipse](#)
4. A1.4 [Git in Bash](#)
5. A1.5 [Git in Zsh](#)
6. A1.6 [Git in Powershell](#)
7. A1.7 [Summary](#)

## 2. **A2. Appendix B: Embedding Git in your Applications**

1. A2.1 [Command-line Git](#)
2. A2.2 [Libgit2](#)
3. A2.3 [JGit](#)

## 3. **A3. Appendix C: Git Commands**

1. A3.1 [Setup and Config](#)
2. A3.2 [Getting and Creating Projects](#)
3. A3.3 [Basic Snapshotting](#)
4. A3.4 [Branching and Merging](#)
5. A3.5 [Sharing and Updating Projects](#)
6. A3.6 [Inspection and Comparison](#)
7. A3.7 [Debugging](#)
8. A3.8 [Patching](#)
9. A3.9 [Email](#)
10. A3.10 [External Systems](#)
11. A3.11 [Administration](#)
12. A3.12 [Plumbing Commands](#)

2nd Edition

# 1.6 Começando - Configuração Inicial do Git

## Configuração Inicial do Git

Agora que você tem o Git em seu sistema, você deve fazer algumas coisas para personalizar o ambiente Git. Você fará isso apenas uma vez por computador e o efeito se manterá após atualizações. Você também pode mudá-las em qualquer momento rodando esses comandos novamente.

O Git vem com uma ferramenta chamada `git config` que permite ver e atribuir variáveis de configuração que controlam todos os aspectos de como o Git aparece e opera. Estas variáveis podem ser armazenadas em três lugares diferentes:

1. `/etc/gitconfig`: válido para todos os usuários no sistema e todos os seus repositórios. Se você passar a opção `--system` para `git config`, ele lê e escreve neste arquivo.
2. `~/.gitconfig` ou `~/.config/git/config`: Somente para o seu usuário. Você pode fazer o Git ler e escrever neste arquivo passando a opção `--global`.
3. `config` no diretório Git (ou seja, `.git/config`) de qualquer repositório que você esteja usando: específico para este repositório.

Cada nível sobrescreve os valores no nível anterior, ou seja, valores em `.git/config` prevalecem sobre `/etc/gitconfig`.

No Windows, Git procura pelo arquivo `.gitconfig` no diretório `$HOME` (`C:\Users\%USER` para a maioria). Ele também procura por `/etc/gitconfig`, mesmo sendo relativo à raiz do sistema, que é onde quer que você tenha instalado Git no seu sistema.

## Sua Identidade

A primeira coisa que você deve fazer ao instalar Git é configurar seu nome de usuário e endereço de e-mail. Isto é importante porque cada *commit* usa esta informação, e ela é carimbada de forma imutável nos *commits* que você começa a criar:

```
$ git config --global user.name "Fulano de Tal"
$ git config --global user.email fulanodetal@exemplo.br
```

Reiterando, você precisará fazer isso somente uma vez se tiver usado a opção `--global`, porque então o Git usará esta informação para qualquer coisa que você fizer naquele sistema. Se você quiser substituir essa informação com nome diferente para um projeto específico, você pode rodar o comando sem a opção `--global` dentro daquele projeto.

Muitas ferramentas GUI o ajudarão com isso quando forem usadas pela primeira vez.

## Seu Editor

Agora que a sua identidade está configurada, você pode escolher o editor de texto padrão que será chamado quando Git precisar que você entre uma mensagem. Se não for configurado, o Git usará o editor padrão, que normalmente é o Vim. Se você quiser usar um editor de texto diferente, como o Emacs, você pode fazer o seguinte:

```
$ git config --global core.editor emacs
```

Vim e Emacs são editores de texto populares comumente usados por desenvolvedores em sistemas baseados em Unix como Linux e Max. Se você não for acostumado com estes editores ou estiver Warning em um sistema Windows, você precisará procurar por instruções de como configurar o seu editor preferido com Git. Se você não configurar o seu editor preferido e não sabe usar o Vim ou Emacs, é provável que você fique bastante confuso ao entrar neles.

## Testando Suas Configurações

Se você quiser testar as suas configurações, você pode usar o comando `git config --list` para listar todas as configurações que o Git conseguir encontrar naquele momento:

```
$ git config --list
user.name=Fulano de Tal
user.email=fulanodetal@exemplo.br
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Pode ser que algumas palavras chave apareçam mais de uma vez, porque Git lê as mesmas chaves de arquivos diferentes (`/etc/gitconfig` e `~/.gitconfig`, por exemplo). Neste caso, Git usa o último valor para cada chave única que ele vê.

Você pode também testar o que Git tem em uma chave específica digitando `git config <key>`:

```
$ git config user.name
Fulano de Tal
```

[prev](#) | [next](#)

[About this site](#)

Patches, suggestions, and comments are welcome.

Git is a member of [Software Freedom Conservancy](#).