Guia de Instalação e Configuração do Git

Introdução às Técnicas de Programação

Sumário

Introdução

O Git é um sistema de controle de versão distribuído que permite rastrear mudanças em arquivos e coordenar o trabalho entre múltiplos desenvolvedores. O GitHub é uma plataforma baseada na web que hospeda repositórios Git e oferece funcionalidades adicionais para colaboração. Este guia fornece instruções detalhadas para instalação, configuração e uso básico do Git com GitHub.

Windows

Opção 1: Instalador Oficial (Recomendado)

Passo 1: Baixe o Git para Windows

- Acesse https://git-scm.com/downloads/win
- Baixe a versão mais recente (64-bit recomendado)
- Execute o instalador como administrador

Passo 2: Configure a instalação

- **Componentes:** Deixe as opções padrão marcadas
- **Editor padrão:** Recomendado usar Visual Studio Code se disponível
- **PATH environment:** Selecione "Git from the command line and also from 3rd-party software"
- **HTTPS transport backend:** Use the native Windows Secure Channel library

- **Line ending conversions:** Checkout Windows-style, commit Unix-style line endings
- **Terminal emulator:** Use MinTTY (the default terminal of MSYS2)

Passo 3: Complete a instalação

- Mantenha as outras opções padrão
- Clique em "Install" e aguarde a conclusão

Opção 2: Chocolatey

Passo 1: Abra o PowerShell como administrador

Passo 2: Instale o Git via Chocolatey

```
choco install git
```

Opção 3: Winget (Windows 10+)

Passo 1: Abra o Terminal ou PowerShell

Passo 2: Instale o Git

```
winget install --id Git.Git -e --source winget
```

---

macOS

Opção 1: Xcode Command Line Tools (Recomendado)

Passo 1: Abra o Terminal

- Pressione `Cmd + Espaço` e digite "Terminal"

Passo 2: Instale as ferramentas de linha de comando

```
xcode-select --install
```

Passo 3: Verifique a instalação

```
git --version
```

Opção 2: Homebrew

Passo 1: Instale o Homebrew (se não tiver)

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Passo 2: Instale o Git

```
brew install git
```

Opção 3: Instalador Oficial

Passo 1: Baixe o Git para macOS

- Acesse https://git-scm.com/downloads/mac
- Baixe o instalador mais recente
- Execute o instalador seguindo as instruções

Linux Ubuntu/Debian

Instalação via APT

Passo 1: Atualize o sistema de pacotes

```
sudo apt update
sudo apt upgrade
```

Passo 2: Instale o Git

```
sudo apt install git
```

Passo 3: Instale ferramentas adicionais (opcional)

```
sudo apt install git-gui gitk
```

Linux Fedora/Red Hat

Instalação com DNF (Fedora)

Passo 1: Atualize o sistema

```
sudo dnf update
```

Passo 2: Instale o Git

```
sudo dnf install git
```

Instalação com YUM (Red Hat/CentOS)

Passo 1: Atualize o sistema

```
sudo yum update
```

Passo 2: Instale o Git

```
sudo yum install git
```

Linux Arch

Instalação com Pacman

Passo 1: Atualize o sistema

```
sudo pacman –Syu
```

Passo 2: Instale o Git

```
sudo pacman –S git
```

Passo 3: Instale ferramentas adicionais (opcional)

```
sudo pacman –S git–gui tk
```

Testando a Instalação

Verificação da Versão

Execute o seguinte comando no terminal para verificar se o Git foi instalado corretamente:

```
git --version
```

Você deve ver uma saída similar a:

```
git version 2.45.2
```

Verificação da Configuração

```
git config --list
```

Configuração Inicial

Configuração Básica Obrigatória

Antes de usar o Git, você deve configurar seu nome e email:

```
# Configure seu nome (substitua "Seu Nome" pelo seu nome real)
git config --global user.name "Seu Nome"

# Configure seu email (use o mesmo email do GitHub)
git config --global user.email "seuemail@exemplo.com"
```

Configurações Adicionais Recomendadas

```
# Configure o editor padrão (opcional)
git config --global core.editor "code --wait"  # Para VS Code
git config --global core.editor "nano"         # Para Nano
git config --global core.editor "vim"          # Para Vim

# Configure cores no terminal
git config --global color.ui auto

# Configure o comportamento de push padrão
git config --global push.default current

# Configure a estratégia de merge padrão
git config --global pull.rebase false
```

Verificando as Configurações

```

```

```
# Visualizar todas as configurações
git config --list

# Verificar configurações específicas
git config user.name
git config user.email
```

Criando uma Conta no GitHub

Registro no GitHub

Passo 1: Acesse o GitHub

- Vá para https://github.com
- Clique em "Sign up"

Passo 2: Preencha os dados

- **Username:** Escolha um nome de usuário único
- **Email:** Use o mesmo email configurado no Git local
- **Password:** Crie uma senha forte

Passo 3: Verifique sua conta

- Verifique seu email e clique no link de confirmação
- Complete o questionário inicial (opcional)

Comandos Básicos do Git

Comandos de Inicialização

```
# Inicializar um repositório Git
git init

# Clonar um repositório existente (HTTPS - método padrão)
git clone https://github.com/usuario/repositorio.git
```

Comandos de Status e Informação

```
# Verificar o status atual do repositório
git status

# Visualizar o histórico de commits
git log

# Visualizar o histórico de forma compacta
```

```
git log --oneline

# Ver diferenças não commitadas
git diff

# Ver diferenças de arquivos já adicionados ao staging
git diff --staged
```

Comandos de Gerenciamento de Arquivos

```
# Adicionar um arquivo específico ao staging area
git add arquivo.txt

# Adicionar todos os arquivos modificados
git add .

# Adicionar todos os arquivos de uma extensão
git add *.c

# Remover arquivo do staging area (não deleta o arquivo)
git restore --staged arquivo.txt

# Descartar mudanças em um arquivo (CUIDADO: não é reversível)
git restore arquivo.txt
```

Comandos de Commit

```
# Fazer commit com mensagem
git commit -m "Descrição das mudanças"

# Fazer commit adicionando automaticamente arquivos modificados
git commit -am "Descrição das mudanças"

# Fazer commit abrindo o editor para mensagem detalhada
git commit

# Alterar a mensagem do último commit (antes do push)
git commit --amend -m "Nova mensagem"
```

Comandos de Sincronização com Repositório Remoto

```
# Adicionar um repositório remoto
git remote add origin https://github.com/usuario/repositorio.git

# Visualizar repositórios remotos configurados
git remote -v
```

```
# Enviar commits para o repositório remoto (primeira vez)
git push -u origin main

# Após a configuração inicial, usar apenas:
git push

# Baixar mudanças do repositório remoto (primeira vez ou especificando
branch)
git pull origin main

# Após a configuração inicial, usar apenas:
git pull

# Baixar mudanças sem fazer merge automático
git fetch origin
```

Comandos de Branch (Ramificação)

```
# Listar todas as branches
git branch

# Criar uma nova branch
git branch nova-funcionalidade

# Mudar para uma branch
git checkout nova-funcionalidade

# Criar e mudar para uma nova branch
git checkout -b nova-funcionalidade

# Fazer merge de uma branch
git checkout main
git merge nova-funcionalidade

# Deletar uma branch (após merge)
git branch -d nova-funcionalidade
```

Criando seu Primeiro Repositório

Cenário 1: Começando um Novo Projeto

Passo 1: Crie um diretório para seu projeto

```
mkdir meu-primeiro-projeto
cd meu-primeiro-projeto
```

Passo 2: Inicialize o repositório Git

```
git init
```

Passo 3: Crie um arquivo inicial

```
echo "# Meu Primeiro Projeto" > README.md
```

Passo 4: Adicione e faça commit do arquivo

```
git add README.md
git commit -m "Primeiro commit: adiciona README"
```

Passo 5: Crie o repositório no GitHub

1. Vá para https://github.com e clique em "New repository"
2. Digite o nome "meu-primeiro-projeto"
3. Não inicialize com README (já temos um)
4. Clique em "Create repository"

Passo 6: Conecte o repositório local ao GitHub

```
git remote add origin https://github.com/seuusuario/meu-primeiro-
projeto.git
git push -u origin main
```

Cenário 2: Clonando um Repositório Existente

Passo 1: Clone o repositório

```
git clone https://github.com/usuario/repositorio.git
```

Passo 2: Entre no diretório

```
cd repositorio
```

Passo 3: Faça suas modificações e commit

```

```

```
# Edite arquivos...
git add .
git commit -m "Minhas modificações"
git push
```

Cenários Típicos de Uso

1. Desenvolvimento Individual

Fluxo básico para trabalho solo:

```
# 1. Fazer mudanças nos arquivos
echo "nova linha" >> arquivo.txt

# 2. Verificar status
git status

# 3. Adicionar mudanças
git add arquivo.txt

# 4. Fazer commit
git commit -m "Adiciona nova funcionalidade"

# 5. Enviar para o GitHub
git push
```

2. Trabalhando com Branches

Criando uma nova funcionalidade:

```
# 1. Criar e mudar para nova branch
git checkout -b nova-funcionalidade

# 2. Fazer modificações e commits
git add .
git commit -m "Implementa nova funcionalidade"

# 3. Enviar branch para o GitHub (primeira vez)
git push -u origin nova-funcionalidade

# 4. Voltar para main e fazer merge
git checkout main
git merge nova-funcionalidade

# 5. Enviar merge para o GitHub
git push
```

```
# 6. Deletar branch local (opcional)
git branch -d nova-funcionalidade
```

3. Colaboração com Outros Desenvolvedores

Sincronizando mudanças:

```
# 1. Sempre puxar mudanças antes de começar
git pull

# 2. Fazer suas modificações
git add .
git commit -m "Suas modificações"

# 3. Puxar mudanças novamente (caso alguém tenha commitado)
git pull

# 4. Resolver conflitos se houver, então push
git push
```

Workflow Básico de Desenvolvimento

Fluxo Diário Recomendado

1. **Iniciar o dia:**

   ```
   git pull
   ```

2. **Durante o desenvolvimento:**

   ```
   # Fazer mudanças pequenas e frequentes
   git add .
   git commit -m "Descrição clara e concisa"
   ```

3. **Antes de enviar para o repositório:**

   ```
   git pull    # Pegar atualizações
   git push    # Enviar suas mudanças
   ```

Boas Práticas para Commits

Mensagens de commit claras:

- Use presente simples: "Adiciona função" ao invés de "Adicionei função"
- Seja específico: "Corrige bug na validação de email" ao invés de "Correção"
- Mantenha a primeira linha com até 50 caracteres
- Use uma linha em branco antes de descrições detalhadas

Exemplos de boas mensagens:

```
git commit -m "Adiciona validação de email no formulário de cadastro"
git commit -m "Corrige erro de divisão por zero na calculadora"
git commit -m "Refatora função de busca para melhorar performance"
git commit -m "Remove código comentado e variáveis não utilizadas"
```

Solução de Problemas Comuns

Problemas de Autenticação

Problema: "Permission denied (publickey)" ou problemas de autenticação HTTPS **Solução:**

1. Para HTTPS: Verifique suas credenciais ou configure um token de acesso pessoal
2. Para SSH: Verifique se a chave SSH foi adicionada corretamente e teste: `ssh -T git@github.com`

Problema: "Support for password authentication was removed" **Solução:**

1. **Método recomendado (HTTPS):** Configure um Personal Access Token:
   - Vá para GitHub → Settings → Developer settings → Personal access tokens → Tokens (classic)
   - Clique em "Generate new token" e selecione os escopos necessários (repo, workflow, etc.)
   - Use o token como senha quando solicitado
2. **Método alternativo:** Configure chaves SSH (conforme explicado abaixo)

Conflitos de Merge

Problema: Conflitos ao fazer git pull **Solução:**

```
# 1. Git mostrará os arquivos em conflito
git status

# 2. Edite os arquivos para resolver conflitos (remova as marcações <<<<<
===== >>>>>)
# 3. Adicione os arquivos resolvidos
git add arquivo-resolvido.txt

# 4. Complete o merge
git commit -m "Resolve conflito de merge"
```

Problemas de Configuração

Problema: "Please tell me who you are" **Solução:**

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

Problema: Editor não abre para mensagens de commit **Solução:**

```
git config --global core.editor "seu-editor-preferido"
```

Problemas com .gitignore

Problema: Arquivo já commitado não está sendo ignorado **Solução:**

```
# Remove o arquivo do índice mas mantém no disco
git rm --cached arquivo-que-deveria-ser-ignorado.txt
git commit -m "Remove arquivo do controle de versão"
```

Desfazer Push Acidental

Problema: Fez push de algo que não deveria **Solução:**

```
# CUIDADO: Só use se você é o único trabalhando no repositório
git reset --hard HEAD~1  # Volta um commit
git push --force         # Força o push (PERIGOSO)
```

Configurando SSH (Método Alternativo)

Nota: SSH é um método alternativo que oferece maior segurança e conveniência após a configuração inicial, mas requer configuração adicional. Para usuários iniciantes, recomenda-se usar HTTPS.

Gerando uma Chave SSH

Passo 1: Gere uma nova chave SSH

```
ssh-keygen -t ed25519 -C "seuemail@exemplo.com"
```

Passo 2: Quando solicitado, pressione Enter para usar o local padrão

```
Enter file in which to save the key (/home/usuario/.ssh/id_ed25519):
[Enter]
```

Passo 3: Digite uma senha para a chave (opcional, mas recomendado)

```
Enter passphrase (empty for no passphrase): [Digite sua senha]
Enter same passphrase again: [Repita a senha]
```

Adicionando a Chave SSH ao ssh-agent

Passo 1: Inicie o ssh-agent

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

Passo 2: Adicione sua chave SSH privada ao ssh-agent

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

Adicionando a Chave SSH ao GitHub

Passo 1: Copie a chave SSH pública

```
# Linux/macOS
cat ~/.ssh/id_ed25519.pub

# Windows (Git Bash)
clip < ~/.ssh/id_ed25519.pub
```

Passo 2: No GitHub

1. Vá para Settings → SSH and GPG keys
2. Clique em "New SSH key"
3. Dê um título descritivo (ex: "Notebook Pessoal")
4. Cole a chave no campo "Key"
5. Clique em "Add SSH key"

Passo 3: Teste a conexão

```
ssh -T git@github.com
```

Usando SSH em vez de HTTPS

Após configurar SSH, você pode usar URLs SSH nos comandos:

```
# Clonar com SSH
git clone git@github.com:usuario/repositorio.git

# Alterar URL existente de HTTPS para SSH
git remote set-url origin git@github.com:usuario/repositorio.git
```

Para aprender mais sobre SSH: Consulte o guia oficial do GitHub sobre SSH

Referências

1. **Documentação Oficial do Git**
   - Git Documentation: https://git-scm.com/doc
   - Git Tutorial: https://git-scm.com/docs/gittutorial
   - Git Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf
2. **GitHub**
   - GitHub Docs: https://docs.github.com
   - GitHub Skills: https://skills.github.com
   - SSH Key Setup: https://docs.github.com/en/authentication/connecting-to-github-with-ssh
3. **Instalação por Sistema Operacional**
   - Git for Windows: https://gitforwindows.org/
   - Homebrew: https://brew.sh/
   - Git Downloads: https://git-scm.com/downloads
4. **Tutoriais e Guias Interativos**
   - Learn Git Branching: https://learngitbranching.js.org/
   - Atlassian Git Tutorials: https://www.atlassian.com/git/tutorials
   - Pro Git Book (gratuito): https://git-scm.com/book
5. **Material Complementar**
   - Git Magic: http://www-cs-students.stanford.edu/~blynn/gitmagic/
6. **Ferramentas e Clientes Gráficos**
   - GitHub Desktop: https://desktop.github.com/
   - GitKraken: https://www.gitkraken.com/
   - SourceTree: https://www.sourcetreeapp.com/
7. **Comunidade e Suporte**
   - Stack Overflow - Tag Git: https://stackoverflow.com/questions/tagged/git
   - GitHub Community: https://github.com/community
   - Reddit r/git: https://www.reddit.com/r/git/

8. **Arquivos .gitignore Templates**
    - GitHub gitignore templates: https://github.com/github/gitignore
    - gitignore.io: https://www.toptal.com/developers/gitignore

Data de criação: Setembro de 2025
Versão: 1.0