Guia rápido de Git e Azure Repos, GitHub Desktop e VS Code

# Instalação e Verificação do Git

1. Instalar Git  
Baixe o Git pelo site oficial: https://git-scm.com/downloads/win  
  
2. Verificar Instalação  
Após a instalação, verifique no terminal (CMD) com o comando:  
git --version  
  
Se instalado corretamente, o Git exibirá a versão atual. Exemplo de saída:  
git version 2.43.0.windows.1

# Instalação do VS Code

1. Baixar o Visual Studio Code  
Você pode baixar o VS Code através do link oficial: https://code.visualstudio.com/  
  
2. Instalar o VS Code  
Siga as instruções de instalação fornecidas no site para seu sistema operacional.  
  
3. Instalar a extensão GitLens (opcional)  
Após instalar o VS Code, uma extensão útil para trabalhar com Git é o GitLens.  
  
4. Configurar o Git no VS Code  
O VS Code reconhece o Git automaticamente após sua instalação. Basta abrir um projeto com repositório Git para usar os comandos diretamente no editor.

# Instalação do GitHub Desktop

1. Baixar GitHub Desktop  
Acesse o site oficial para baixar o GitHub Desktop: https://desktop.github.com/  
  
2. Instalar GitHub Desktop  
Siga as instruções no site para completar a instalação.  
  
3. Configurar GitHub Desktop  
Após a instalação, faça login com sua conta do GitHub e conecte seus repositórios locais ao GitHub Desktop. Assim, você poderá gerenciar facilmente seus commits e branches.

# Configuração Inicial do Git

1. Configurar informações globais de usuário:  
git config --global user.name "Rodrigo Sena de Oliveira"  
git config --global user.email "rodrigo.sonic@gmail.com"  
git config --global init.defaultBranch main  
  
2. Definir diretório do projeto:  
cd C:\\_Work\AzureRepos  
  
3. Inicializar o repositório:  
git init  
  
Esse comando cria o diretório .git onde todo o histórico de controle de versão é armazenado.

# Configuração do Git no VS Code e GitHub Desktop

1. Verificar se o Git está configurado corretamente  
No VS Code, você pode verificar se o Git está integrado através do terminal integrado (Ctrl + `) com o comando:  
git --version  
  
2. Configurar informações globais de usuário (se não configurado)  
Se você ainda não configurou o Git, execute os seguintes comandos para definir o nome de usuário e e-mail globalmente:  
git config --global user.name "Seu Nome"  
git config --global user.email "seu.email@exemplo.com"  
  
3. Usar o Git no VS Code e GitHub Desktop  
Após configurado, você pode executar comandos Git diretamente no terminal do VS Code ou gerenciar seus repositórios visualmente no GitHub Desktop.

# Verificando o status do repositório

1. Status dos arquivos:  
git status  
  
2. Adicionar arquivos ao rastreamento:  
Para um arquivo específico:  
git add Index.html  
  
Para adicionar todos os arquivos:  
git add --all  
  
3. Realizar o primeiro commit:  
git commit -m "Versão inicial do Projeto - Todos arquivos Inclusos: PROJETO: POC\_TESTE"

# Trabalhando com alterações e commits

1. Visualizar alterações:  
git diff  
  
Para sair da visualização, pressione Q.  
  
2. Realizar commit após alteração:  
git commit -m "Alterado meu arquivo para Amarelo"  
  
3. Ver histórico de commits:  
Exibir histórico:  
git log  
  
Exibir histórico em uma linha:  
git log --oneline  
  
4. Excluir arquivos rastreados:  
git rm Index\_amarelo.html  
  
5. Restaurar arquivos:  
Restaurar um arquivo em Staged:  
git restore --staged Index\_amarelo.html  
  
Restaurar um arquivo no workspace:  
git restore Index\_amarelo.html

# Reset de commits

Tipos de reset:  
  
1. Soft: Mantém as alterações preparadas para commit.  
git reset --soft <commit\_hash>  
  
2. Mixed: Reverte as alterações e precisa de git add para preparar novos commits.  
  
3. Hard: Reverte completamente o histórico de commits e remove as alterações.  
git reset --hard <commit\_hash>

# Branches

1. Exibir branches locais:  
git branch  
  
2. Criar uma nova branch:  
git switch -c Feature1  
  
3. Deletar uma branch:  
git branch -d Feature1  
  
4. Merge de branch:  
git merge -m "Merge da Branch Feature1 para a Main" Feature1

# Comandos úteis

1. Alias para histórico formatado:  
git config --global alias.lg "log --oneline --decorate --all --graph --pretty=format:'%h %ad | %s%d [%an]' --date=format:'%Y-%m-%d %H:%M:%S'"  
  
2. Visualizar todos os alias:  
git config --get-regexp alias

# Conectando ao GitHub

1. Criar repositório remoto no GitHub:  
URL: https://github.com/rodrigosenna/myportifolio  
  
2. Gerar um token de acesso em GitHub Developer Settings. Exemplo:  
ghp\_TtVLW4HNNSqaDXNsjsVyV6Ub1UMLlC0TsQ0o  
  
3. Vincular repositório local ao remoto:  
git remote set-url origin https://rodrigosenna:ghp\_TtVLW4HNNSqaDXNsjsVyV6Ub1UMLlC0TsQ0o@github.com/rodrigosenna/myportifolio.git  
  
4. Push para o repositório remoto:  
git push origin master  
  
5. Verificar o repositório remoto:  
git remote -v

# Resolvendo problemas de Proxy

1. Verificar configuração de proxy:  
git config --global --get http.proxy  
git config --global --get https.proxy  
  
2. Remover proxy:  
git config --global --unset http.proxy  
git config --global --unset https.proxy

# Trabalhando com o repositório remoto

1. Fazer Pull do repositório remoto:  
git pull origin main  
  
2. Buscar alterações sem mesclar:  
git fetch origin  
  
3. Mesclar alterações do remoto:  
git merge origin/main

4. Fazer push de todos objetos e branchs  
git push --all