Lista de Exercícios

Esta lista de exercícios irá ajudá-lo a consolidar os conceitos e comandos básicos do Git abordados na primeira aula. Lembre-se de praticar os comandos em um ambiente real, utilizando o terminal ou Git Bash, para uma melhor compreensão.

Exercício 1: Instalando o Git

1. Baixe e instale o Git em sua máquina a partir do link: https://git-scm.com/downloads.

2. Verifique se a instalação foi bem-sucedida utilizando o comando:
   - `git --version`

3. Qual versão do Git foi instalada?

Exercício 2: Criando uma Conta no GitHub

1. Crie uma conta no GitHub, se ainda não tiver uma, acessando https://github.com.

2. Após criar sua conta, acesse seu perfil e personalize suas informações (nome, foto, etc.).

Exercício 3: Configuração Inicial do Git

1. Configure seu nome de usuário e e-mail, que serão usados em seus commits.
   - `git config --global user.name "Seu Nome"`
   - `git config --global user.email "seuemail@example.com"`

2. Qual é a importância de configurar essas informações?

Exercício 4: Iniciar um Repositório Local

1. Crie uma pasta em seu computador chamada `meu_projeto`.

2. Navegue até essa pasta utilizando o terminal.

3. Inicialize um novo repositório Git nesta pasta.
   - Qual comando você usou?

4. Criar um readme

Exercício 5: Status do Repositório

1. Crie um arquivo chamado `index.html` dentro da pasta `meu_projeto`.

2. Utilize o comando para verificar o status atual do repositório.
   - Qual é o status do arquivo `index.html`?

Exercício 6: Adicionando Arquivos à Staging Area

1. Adicione o arquivo `index.html` à área de preparação (staging area).
   - Qual comando você utilizou?

2. Verifique novamente o status do repositório. O que mudou?

Exercício 7: Fazendo o Primeiro Commit

1. Realize um commit com a mensagem "Primeiro commit: adiciona index.html".
   - Qual comando foi utilizado?

2. Explique por que é importante fornecer mensagens claras nos commits.

Exercício 8: Criando um Repositório Remoto no GitHub

1. No GitHub, crie um novo repositório chamado `meu_projeto_remoto`.

2. Qual comando você usaria para conectar o repositório local ao repositório remoto?
   - Inclua a URL do repositório remoto fictício.

Exercício 9: Enviando Mudanças para o Repositório Remoto

1. Envie o commit feito para o repositório remoto utilizando o comando adequado.
   - Qual é o comando utilizado?

2. O que acontece se você tentar enviar mudanças sem estar conectado a um repositório remoto?

Exercício 10: Clonando um Repositório

1. Qual é o comando para clonar um repositório remoto em sua máquina local?

2. Qual é a diferença entre `git clone` e `git init`?

Exercício 11: Criando e Navegando entre Branches

1. Crie uma nova branch chamada `feature_a`.
   - Qual comando você utilizou?

2. Mude para a branch `feature_a`.
   - Qual é o comando para fazer isso?

Exercício 12: Mesclando Branches

1. Suponha que você tenha feito alterações na branch `feature_a` e queira mesclá-las com a branch `main`.

2. Qual é o comando para realizar essa mesclagem?

3. O que pode acontecer se houver conflitos durante o merge? Como resolver?

Exercício 13: Atualizando o Repositório Local

1. Utilizando o comando adequado, traga as atualizações do repositório remoto para o repositório local.
   - Qual é o comando?

2. Explique a diferença entre `git pull` e `git fetch`.

Exercício 14: Workflow Completo

1. Descreva o workflow completo de trabalho utilizando Git, desde a instalação e configuração até o envio de alterações para o repositório remoto.

2. Por que é importante seguir um fluxo de trabalho organizado ao utilizar Git em projetos colaborativos?

Exercícios após a aula 4

Exercício 15: Conflito de Código em Alterações Simultâneas

1. Simulação de Conflito:
    - Crie um novo arquivo chamado `colaborativo.txt` na branch `feature_a`.
    - Adicione a linha: "Este é um arquivo colaborativo" e faça o commit.

    - Depois, volte para a branch `main` e crie o mesmo arquivo `colaborativo.txt`.
    - Desta vez, adicione a linha: "Texto inicial diferente" e faça o commit.
2. Mesclagem que Gera Conflito:
    - Tente mesclar a branch `feature_a` na branch main. Qual comando você usou?
    - Qual mensagem de erro foi apresentada durante o merge?
    - Como resolver esse conflito e finalizar o merge?

Exercício 16: Conflito de Código em Arquivo Já Existente

1. Criando um Conflito em Arquivo Existente:
    - Crie uma nova branch chamada `feature_b`.
    - Abra o arquivo index.html que você criou anteriormente e adicione a linha "<h1>Nova seção no index.html</h1>". Faça o commit.
    - Enquanto isso, na branch main, adicione a linha "<h1>Seção de boas-vindas no index.html</h1>" no mesmo lugar. Faça o commit.

2. Tentativa de Mesclar Alterações:
    - Tente mesclar a branch `feature_b` na branch main.
    - Explique o conflito que ocorreu e quais são as opções de resolução (Accept Current Change, Accept Incoming Change, etc.).

Exercício 17: Conflito em Deleção e Modificação

1. Simulação de Conflito de Deleção:
    - Crie uma nova branch chamada feature_delete.
    - Na branch feature_delete, delete o arquivo index.html e faça o commit.
    - Volte para a branch main e faça uma modificação no mesmo arquivo index.html, como adicionar uma nova tag Texto de introdução . Faça o commit.
2. Tentativa de Mesclar as Alterações:
    - Tente mesclar a branch feature_delete na branch main.
    - O que aconteceu quando você tentou fazer o merge?
    - Como lidar com a situação quando uma branch exclui um arquivo e outra o modifica?

Exercício 18: Conflito em Código Não-Texto

1. Criando Conflito em Arquivo Binário:
    - Crie uma branch chamada feature_image.
    - Adicione um arquivo de imagem logo.png na pasta meu_projeto e faça um commit.
    - Enquanto isso, na branch main, substitua o arquivo logo.png por uma versão diferente da imagem e faça o commit.

2. Tentativa de Mesclar as Alterações:
    - Tente mesclar a branch feature_image na branch main.
    - Como o Git trata conflitos em arquivos binários?
    - O que é recomendado para resolver conflitos em arquivos binários?

Exercício 19: Conflito na Colaboração em Equipe

1. Simulando um Conflito com Colaborador:
    - Peça a um colega para criar uma branch feature_teamwork e adicionar uma nova linha no arquivo index.html.
    - Ao mesmo tempo, crie uma branch com o mesmo nome (feature_teamwork) e modifique o mesmo arquivo index.html, adicionando uma linha diferente.
    - Ambos devem tentar mesclar suas alterações na branch main.
2. Resolução de Conflito Colaborativo:
    - Qual foi o conflito que ocorreu?
    - Como coordenar a resolução de conflitos com o seu colega para evitar a perda de alterações importantes?
    - Qual é a importância da comunicação em um fluxo de trabalho colaborativo para prevenir conflitos?

Exercício 20: Conflito por Rebase

1. **Rebase que Gera Conflito**:
    - Crie uma branch `feature_c` a partir da branch `main`.
    - Faça uma alteração no arquivo `index.html` na branch `feature_c` e faça um commit.
    - Volte para a branch `main` e faça outra alteração no mesmo arquivo `index.html`, em outra linha, e faça o commit.
    - Agora, tente rebasear a branch `feature_c` na branch `main`.
2. **Resolução de Conflito de Rebase**:
    - Qual foi o comando utilizado para realizar o rebase?
    - Como os conflitos foram sinalizados pelo Git?
    - Resolva os conflitos e finalize o rebase.

Exercício 21: Merge vs Rebase em Situação de Conflito

1. **Comparando `git merge` e `git rebase`**:
    - Crie uma branch `feature_merge_vs_rebase`.
    - Faça uma alteração no arquivo `colaborativo.txt` e faça um commit.
    - Enquanto isso, na branch `main`, adicione outra linha ao mesmo arquivo e faça o commit.
2. **Merge e Rebase na Prática**:

- Realize um `git merge` da branch `feature_merge_vs_rebase` na branch `main` e anote o resultado.
- Refaça o processo utilizando `git rebase`.
- Qual é a diferença na forma como o Git trata as alterações?
- Em qual situação seria mais apropriado usar o merge e em qual seria mais adequado usar o rebase?