Laboratório: Criando um Fork no GitHub

Neste exercício, vamos praticar a criação de um fork de um repositório no GitHub. Siga os passos abaixo:

1. Acesse o repositório do projeto no GitHub.
2. Clique no botão "Fork" no canto superior direito da página.
3. Selecione a conta para a qual deseja fazer o fork.
4. Aguarde enquanto o GitHub cria uma cópia do repositório em sua conta.
5. Após a conclusão do fork, você será redirecionado para a página do seu fork.

Agora você criou com sucesso um fork do repositório no GitHub. Você pode clonar o fork para a sua máquina local e começar a fazer alterações nele.

Laboratório: Utilizando Pull Requests no GitHub

Neste exercício, vamos praticar a utilização de Pull Requests no GitHub para colaborar em projetos de código aberto. Siga os passos abaixo:

1. Acesse o repositório do projeto no GitHub.
2. Faça um fork do repositório para a sua conta pessoal clicando no botão "Fork" no canto superior direito da página.
3. Clone o seu fork do repositório para a sua máquina local usando o comando `git clone <URL_do_seu_fork>`.
4. Crie uma nova branch para a sua contribuição usando o comando `git checkout -b minha-contribuicao`.
5. Faça as alterações desejadas no código, adicionando novos recursos, corrigindo bugs ou melhorando a documentação.
6. Após realizar as alterações, adicione os arquivos modificados ao commit usando o comando `git add <arquivos_modificados>`.
7. Faça o commit das alterações usando o comando `git commit -m "Minha contribuição"`.
8. Faça o push da sua branch para o seu fork do repositório usando o comando `git push origin minha-contribuicao`.
9. Acesse o seu fork do repositório no GitHub e clique no botão "New Pull Request".
10. Selecione a branch que contém as suas alterações como a branch base e a branch principal do projeto como a branch de comparação.

11. Preencha o título e a descrição da Pull Request, explicando as alterações realizadas e os motivos da contribuição.
12. Clique no botão “Create Pull Request” para enviar a sua contribuição para revisão.
13. Aguarde a revisão e os comentários dos mantenedores do projeto. Esteja disponível para discutir as alterações e fazer ajustes conforme necessário.
14. Após a aprovação da sua Pull Request, os mantenedores do projeto irão mesclar as suas alterações na branch principal do projeto.
15. Parabéns! Você contribuiu com sucesso para o projeto utilizando Pull Requests no GitHub.

    Lembre-se de seguir as diretrizes de contribuição do projeto e respeitar o código de conduta da comunidade.

Laboratório: Utilizando o comando Git rebase

```
# Crie um novo branch a partir do master

git checkout -b feature

# Faça algumas alterações e commits
echo "nova funcionalidade" > feature.txt
git add feature.txt
git commit -m "Adiciona nova funcionalidade"

# Volte para o master e faça algumas alterações e commits
git checkout master
echo "algumas mudanças" > changes.txt
git add changes.txt
git commit -m "Faz algumas mudanças"

# Agora, você quer integrar as mudanças do master no seu branch feature
# Primeiro, volte para o branch feature
git checkout feature

# Em seguida, use o comando git rebase
git rebase master
```

Laboratório: Utilizando o comando Git stash

1. Faça algumas alterações em um arquivo

```
echo "algumas alterações" > arquivo.txt
```

2. Agora, digamos que você precisa mudar para um branch diferente, mas não quer commitar suas alterações ainda
3. Você pode usar o comando git stash para salvar suas alterações

```
git stash
```

4. Agora suas alterações foram salvas e você pode mudar para um branch diferente

```
git checkout outro_branch
```

5. Depois de terminar no outro branch, você pode voltar para o seu branch original

```
git checkout branch_original
```

6. E você pode recuperar suas alterações com o comando git stash pop

```
git stash pop
```

Laboratório: Utilizando o comando Git cherry-pick

1. Suponha que você tenha dois branches: master e feature
2. O branch feature tem alguns commits que você gostaria de aplicar ao master
3. Obtenha o hash do commit que você gostaria de aplicar. Você pode fazer isso com o comando git log

```
git checkout feature
git log
```

4. Isso mostrará uma lista de commits. Cada commit tem um hash associado a ele, que é uma string longa de caracteres e números.
5. Copie o hash do commit que você gostaria de aplicar.
6. Agora, volte para o branch master

```
git checkout master
```

7. Use o comando git cherry-pick para aplicar o commit

```
git cherry-pick hash_do_commit
```

Laboratório: Utilizando um release branch

Neste exercício, vamos praticar o uso de um release branch no Git a partir do branch `main`.

1. Certifique-se de que você está no branch `main` executando o comando `git checkout main`.
2. Crie um novo branch chamado `release/v1.0` a partir do branch `main` usando o comando `git checkout -b release/v1.0`.
3. Realize as atividades relacionadas ao lançamento da versão 1.0 do software no branch `release/v1.0`, como testes finais, correção de bugs críticos e preparação para implantação.
4. Após concluir as atividades do release, faça a mesclagem do branch `release/v1.0` de volta para o branch `main` usando o comando `git merge release/v1.0`.
5. Faça o push dos branches `release/v1.0` e `main` para o repositório remoto usando o comando `git push origin release/v1.0` e `git push origin main`.

Agora você praticou o uso de um release branch no Git a partir do branch `main` para preparar e lançar uma versão estável do software.

Laboratório: Utilizando um hotfix branch

Neste exercício, vamos praticar o uso de um hotfix branch no Git a partir do branch `main`.

1. Certifique-se de que você está no branch `main` executando o comando `git checkout main`.
2. Crie um novo branch chamado `hotfix/bug123` a partir do branch `main` usando o comando `git checkout -b hotfix/bug123`.
3. Realize a correção do bug 123 no hotfix branch, fazendo as alterações necessárias no código.
4. Após concluir a correção do bug, faça a mesclagem do branch `hotfix/bug123` de volta para o branch `main` usando o comando `git merge hotfix/bug123`.
5. Faça o push dos branches `hotfix/bug123` e `main` para o repositório remoto usando o comando `git push origin hotfix/bug123` e `git push origin main`.

Agora você praticou o uso de um hotfix branch no Git a partir do branch `main` para corrigir um bug crítico em produção.

Laboratório: Utilizando um fix forward branch

Neste exercício, vamos praticar o uso de um fix forward branch no Git a partir do branch `main`.

1. Certifique-se de que você está no branch `main` executando o comando `git checkout main`.
2. Identifique o commit que contém a correção específica que você deseja aplicar em outra branch. Você pode usar o comando `git log` para visualizar o histórico de commits e encontrar o commit desejado.
3. Copie o hash do commit que contém a correção.
4. Crie um novo branch chamado `fix-forward` a partir do branch `main` usando o comando `git checkout -b fix-forward`.
5. Execute o comando `git cherry-pick <hash_do_commit>` para aplicar o commit na branch `fix-forward`. Substitua `<hash_do_commit>` pelo hash do commit que você copiou anteriormente.
6. O Git irá aplicar as alterações desse commit na branch `fix-forward`. Se houver conflitos, você precisará resolvê-los manualmente. Após resolver cada conflito, você pode continuar o processo de cherry-pick com o comando `git cherry-pick --continue`.
7. Se em algum momento você desejar abortar o processo de cherry-pick, você pode usar o comando `git cherry-pick --abort`.
8. Após a conclusão do cherry-pick, você pode verificar o histórico de commits na branch `fix-forward` para ver as alterações aplicadas. Use o comando `git log`.
9. Faça o push do branch `fix-forward` para o repositório remoto usando o comando `git push origin fix-forward`.

   Agora você praticou o uso de um fix forward branch no Git a partir do branch `main` para aplicar uma correção específica em outra branch.

Laboratório: Revisão de código - Alteração de cor em uma página HTML

Neste exercício, você será responsável por revisar uma alteração de cor em uma página HTML em uma pull request no GitHub. Siga os passos abaixo:

1. Acesse a pull request no GitHub que contém a alteração de cor em uma página HTML.
2. Leia atentamente o código modificado e verifique se a alteração está correta e de acordo com os requisitos.
3. Verifique se a alteração de cor foi feita de forma consistente em todos os elementos relevantes da página.
4. Analise se a alteração de cor segue as boas práticas de design e usabilidade.

5. Verifique se o código está bem estruturado, legível e de fácil manutenção.
6. Identifique possíveis problemas, erros ou melhorias que possam ser feitas no código.
7. Deixe comentários claros e construtivos na pull request, apontando as observações e sugestões de melhoria.
8. Caso necessário, discuta as alterações com o autor da pull request para esclarecer dúvidas ou solicitar ajustes.
9. Após revisar o código e fornecer os comentários, aguarde a resposta do autor da pull request e esteja disponível para discutir as alterações.
10. Se estiver satisfeito com a alteração de cor e o código estiver de acordo com os requisitos, aprove a pull request e proceda com a mesclagem das alterações.