Gerência de Configuração e Controle de Versão: Ferramentas de Controle de Versão Atividade 1 - PI 1

1) A ferramenta Git torna possível o controle de alterações no código, um salvamento do histórico de modificações. O que proporciona não apenas segurança e backups para o código, mas também a partição deste em etapas, funcionalidades e “patches”; tornando-o mais organizado e bem escrito. Além disso, o uso do Git permite a criação de “diffs”, um resumo do que um commit modifica em um código, tornando extremamente fácil a leitura do mesmo e de suas novas funcionalidades. Usando o Git, não é preciso voltar a ler inúmeras linhas de código, mas apenas o “diff” de uma modificação. Por fim, o Git permite a criação de "patches" para o código do software, onde é possível aplicar uma específica modificação para o adicionar de uma funcionalidade.

2) A ferramenta Git trabalha com o gerenciamento de versões de arquivos. Para fazer o upload a um servidor remoto, é preciso que haja compatibilidade entre as versões dos arquivos locais e remotas. Logo, havendo uma modificação remota que não está presente no repositório local, não poderá ser feito o envio das modificações locais. É por esse motivo que há a necessidade de se usar o comando “git pull”, este comando baixa as modificações remotas dos arquivos do repositório e atualiza o repositório local. O deixando pronto para um futuro envio de modificações locais.

3) É recomendado por boa prática a criação de novas “branches” por segurança e organização. Criando ramificações, mantemos o código na “main” seguro, já que não o modificamos em si, apenas fazemos um “merge” para modificar a ramificação principal quando tivermos certeza de que a alteração correta foi feita. Além disso, podemos separar certas versões do código, com certas funcionalidades, em “branches” específicas, tornando o código mais organizado. E finalmente, podemos separar em “branches” a colaboração de cada usuário, permitindo o “merge” de cada um ou de apenas um; o que torna possível a colaboração de múltiplos usuários ao mesmo tempo.

4) Um “pull request” é um pedido feito em um repositório remoto, caso um contribuinte, não dono do repositório, queira fazer uma modificação neste. Este “pedido de pull” ou pedido de envio é feito ao dono do repositório, que pode ou não aceitar as modificações do pedido. Tendo sido aceito, o dono do repositório poderá então realizar o “merge”, o juntar das modificações feitas no pedido, seja este originário de uma branch ou fork.

5) Alternativa A, já que o comando “git add” adiciona a uma lista o que será feito o “commit”. Em seguida, o comando “git commit” faz o commit propriamente dito. Esses são os passos de preparo para o envio de um lote de modificações a um repositório.

6) (I) - Alternativa falsa, já que o comando “git checkout” é utilizado principalmente para fazer mudança de “branches”, mas também criação destas.

(II) - Alternativa verdadeira.

(III) - Alternativa verdadeira.

(IV) - Alternativa falta. O comando “git status” lista inúmeras informações sobre a situação da “branch” e repositório atual. É utilizado para se ter uma visão geral e receber informações sobre o repositório local atual.

Alternativa A, II e III são verdadeiras.

7)

(F) - A primeira alternativa é falsa, já que para adicionarmos um arquivo a um repositório Git, primeiramente deve ser usado o comando “add”; e em seguida, o comando “commit”.

(V) - O Git salva o histórico de alterações em arquivos. Além disso, é sim possível obter o histórico de modificações de um arquivo através de uma plataforma como o GitHub.

(F) - O GitHub e GitLab são implementações do Git para operarem remotamente.

(F) - O comando “git pull” baixa as “changes” de um repositório remoto, atualizando o repositório local. Já o comando “git merge”, junta duas “branches” diferentes em uma. Logo, os comandos não são os mesmos.