Git e GitHub Introdução

O que é versionamento?

Imagina um trabalho de escola, onde escrevemos um documento para enviar aos nossos professores. Este arquivo tem versões, pois, quando escrevemos ele, começamos fazendo um pouco hoje, outro tanto amanhã, o restante depois, até chegar na versão final e entregarmos aos professores. O mesmo acontece com códigos de programas. Agora imagine este mesmo trabalho feito em grupo? Os softwares possuem milhares de linhas de código e arquivos e temos que tomar muito cuidado com cada alteração feita ali. Um bug (um problema) inserindo em uma versão do nosso software pode nos causar estragos terríveis; Por isso é importante que consigamos voltar a versão anterior do nosso software facilmente e para que isso seja fácil, somente tendo um bom controle de versão. Ter a pasta do sistema chamado de sistema-v1.zip, sistema-v2.zip, não diz nada sobre o que foi alterado naquela versão.

O que é Git?

Com o git nós podemos criar versões do nosso sistema ou voltar para versões anteriores facilmente, assim como analisar o histórico do que já foi feito. Como um bom sistema de controle de versão, o Git registra as mudanças que ocorrem no código- fonte de um projeto. Logo, permite que os arquivos sejam alterados de forma simultânea, por inúmeras pessoas, sem a preocupação que essas alterações sejam sobrescritas uma pelas outras. Também permitindo a utilização das operações mesmo offline.

Como funciona o Git na prática?

De forma simplificada, o Git trabalha em uma arquitetura em Branch (ou ramificações). Cada novo commit, ou seja, alterações do código, cria um novo ponto na ramificação atual (uma branch).

Exemplificação de um fluxo de contribuição em um projeto utilizando o git:

1. Clone do Projeto:

O primeiro passo para obter o código fonte do projeto é cloná-lo em uma máquina, para que os arquivos fiquem disponíveis localmente.

1. Criação da Branch:

Ao criar uma Branch, estamos criando uma nova ramificação, totalmente independente, para podermos alterar os arquivos do projeto sem interferir nos originais. Esse projeto é considerado uma boa prática quando se está trabalhando em nova funcionalidade.

1. Commits:

Conforme vão sendo criados e alterados os arquivos, elas vão sendo divididas em commits. É importante que a descrição de cada commit seja objetiva, pois ela vai ficar salva no histórico das alterações.

1. Hora do Push:

Uma vez que a funcionalidade está totalmente finalizada, devemos enviar nossa Branch, com tadas as alterações, de volta ao repositório remoto. Assim, ela ficará disponível para os demias contribuidores do projeto poderem ver e alterar.

1. Merge para juntar tudo:

Para mesclar as modificações de sua Branch principal ou máster, você pode utilizar o comando Merge. Após isso, é necessário dar um commit e um push, para enviar a ramificação máster mesclado ao repositório remoto e deixar tudo disponível para os demias contribuidores. Existe também o Pull Request, que geralmente tem relação com a contribuição em projetos open source. Basicamente, ele oc

orre quando se pede para o dono do repositório que suas modificações sejam incluídas nele.