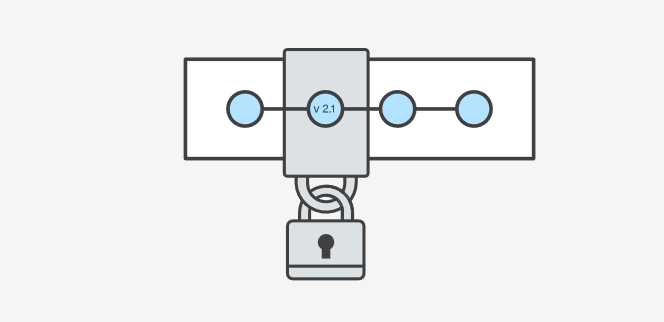
**1 – O que é Git?**

****Git é um sistema de controle de versão, então primeiro precisamos entender que tipo de sistema é esse.

O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas.

Se não houver um sistema de versão, imagine o caos entre duas pessoas abrindo o mesmo arquivo ao mesmo tempo. Uma das aplicações do git é justamente essa, permitir que um arquivo possa ser editado ao mesmo tempo por pessoas diferentes. Por mais complexo que isso seja, ele tenta manter tudo em ordem para evitar problemas para nós desenvolvedores.

Quando não usamos um controle de versão, corremos o risco de não acompanhar a evolução do nosso trabalho, perder histórico, desconhecer o que foi modificado nos nossos arquivos e corremos o risco também de perde nossos arquivos e não ter back-up.

Os principais modelos de controle de versão são:

Centralizado:



Esses sistemas, como por exemplo o CVS, Subversion e Perforce, possuem um único servidor central que contém todos os arquivos versionados e vários clientes que podem resgatar (check out) os arquivos do servidor. Por muitos anos, esse foi o modelo padrão para controle de versão.

Distribuído:



Em um DVCS (tais como Git, Mercurial, Bazaar ou Darcs), os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo. Cada checkout (resgate) é na prática um backup completo de todos os dados

**O que é git?**



Git é um sistema de controle de versão distribuído, feito em 2005 pelo criador do Linux, Linus Torvalds, para saciar as necessidades que ele e sua equipe tinham na época em que desenvolviam o kernel do Linux

O que difere o GIT dos demais controles de versões é a maneira que ele manipula os arquivos.



Conceitualmente, a maior parte dos outros sistemas armazena informação como uma lista de mudanças por arquivo. Esses sistemas (CVS, Subversion, Perforce, Bazaar, etc.) tratam a informação que mantém como um conjunto de arquivos e as mudanças feitas a cada arquivo ao longo do tempo.



Git não pensa ou armazena sua informação dessa forma. Ao invés disso, o Git considera que os dados são como um conjunto de snapshots (captura de algo em um determinado instante, como em uma foto) de um mini-sistema de arquivos. Cada vez que você salva ou consolida (commit) o estado do seu projeto no Git, é como se ele tirasse uma foto de todos os seus arquivos naquele momento e armazenasse uma referência para essa captura. Para ser eficiente, se nenhum arquivo foi alterado, a informação não é armazenada novamente – apenas um link para o arquivo idêntico anterior que já foi armazenado.

**Entenda o ciclo básico da ferramenta**

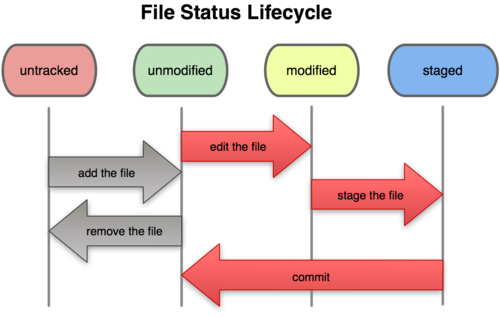


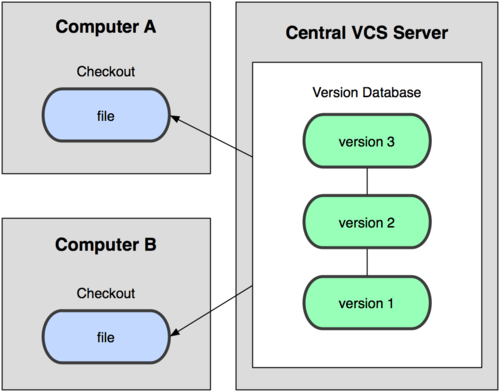
Git faz com que seus arquivos sempre estejam em um dos três estados fundamentais: consolidado (committed), modificado (modified) e preparado (staged). Dados são ditos consolidados quando estão seguramente armazenados em sua base de dados local. Modificado trata de um arquivo que sofreu mudanças mas que ainda não foi consolidado na base de dados. Um arquivo é tido como preparado quando você marca um arquivo modificado em sua versão corrente para que ele faça parte do snapshot do próximo commit (consolidação).

Isso nos traz para as três seções principais de um projeto do Git:

* O diretório do Git (git directory, repository), onde fica localizado todo o seu projeto, que você clonou
* O diretório de trabalho (working directory), a ultima versão do projeto que está na sua maquina
* A área de preparação (staging area), também chamado de INDEX, é o local onde ficam os arquivos que serão enviados no próximo commit. Área que não existe em outros sistemas de versão

Deixando o ciclo de vida explícito:



**Git vs SVN**

GIT

Benefits

GIT

CVS / SVN



A maior parte das operações no Git precisam apenas de recursos e arquivos locais para operar — geralmente nenhuma outra informação é necessária de outro computador na sua rede. Se você está acostumado a um CVCS onde a maior parte das operações possui latência por conta de comunicação com a rede, esse aspecto do Git fará com que você pense que os deuses da velocidade abençoaram o Git com poderes sobrenaturais. Uma vez que você tem todo o histórico do projeto no seu disco local, a maior parte das operações parece ser quase instantânea.

Assim, você pode executar todas as operações – enviar alterações, mesclar – localmente, sem a necessidade de conexão. Se você está num avião, por exemplo, você ainda assim pode controlar e gerenciar as versões dos seus arquivos.

Com o projeto inteiro na sua máquina, você contém um backup atualizado do projeto na máquina de todo o time de desenvolvimento, assim, você não irá perder caso o servidor caia, diferentemente do que ocorreria no uso do SVN.

Por ser distribuído, você nao precisa dar permissão pra que outras pessoas tenham acesso ao seu commit e assim possam usar as funções do versionamento, você simplesmente escolhe o commit que você quer na hora que você quer. Diferentemente do SVN, que precisa fornecer permissão aos usuários, para permitir os checkins diários.

Branches no SVN ficam localizadas em pastas no servidor, enquanto que no git ele é apenas um ponteiro, que pode estar armazenado localmente, para algum commit que nós enviamos.

**Trabalhando com branches**

