1 de agosto de 2019

aTENTO bRASIL s/a

sistemas

Curso de GIT

Versão 2

Gestão de Configuração



**Índice**

1. Quando e quem? ................................................................................................2
2. O que é o Git?......................................................................................................2
3. GitHub
   1. Caracteristicas ..............................................................................................3
4. Difusão e mercado hoje........................................................................................4
5. WorkFlow..............................................................................................................5
6. Repositórios GIT no AzureDevOps (AtentoALM) ............................................
7. Repositórios GIT no VISUAL STUDIO............................................................
8. Branches ..................................................................................................
   1. Cases
   2. Boas práticas BRANCHES

7.3.Conflitos ..................................................................................................

1. Commit ..................................................................................................
2. Histórico ..................................................................................................
3. ..................................................................................................
4. Dúvidas..........................................................................................................
5. Referências..................................................................................................
6. **Quando e quem?**

Desenvolvido em 2005 pelo finlandês Linus Torvalds, conhecido por criar o kernel do Linux.

Mantido pelo engenheiro japonês Junio Hamano desde o ano de sua criação.

Uma imagem contendo pessoa, parede, homem, interior

Descrição gerada com muito alta confiança

Linus Torvalds:

Linkedin: <https://www.linkedin.com/in/linustorvalds/>

GitHUb: <https://github.com/torvalds>

Uma imagem contendo pessoa, interior, parede, mulher

Descrição gerada com muito alta confiança

Junio Hamano:

Linkedin: <https://www.linkedin.com/in/gitster>

Blog: <https://usesthis.com/interviews/junio.c.hamano/>

GitHub: <https://github.com/gitster>

1. **O que é o GIT?**

Um sistema de controle de versão distribuído.

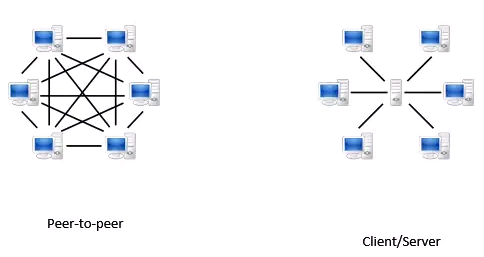
Uma ferramenta que gerencia toda a vida de um dado projeto, guardando o histórico, ou versão, de cada documento do projeto. Lembrando que documento pode ser qualquer coisa, uma imagem, um código, um pdf, etc.

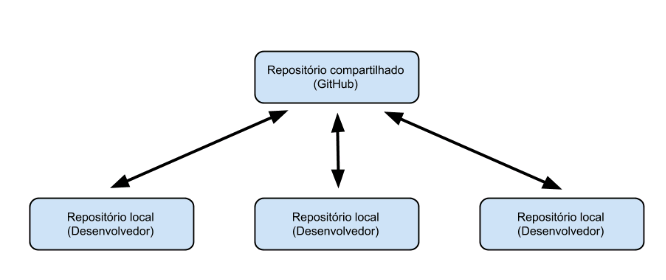
2.1- Caracteristicas:

1. O GIT segue uma abordagem **peer to peer,** contrário de outros como o Subversion (SVN) que segue um modelo baseado em **cliente-servidor**.

Peer to peer: É uma arquitetura de redes onde cada um dos pontos da rede funciona tanto como cliente quanto como servidor, permitindo compartilhamentos de serviços e dados sem a necessidade de um servidor central.

Cliente-Servidor: Uma estrutura de [aplicação distribuída](https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_distribu%C3%ADda#Aplica%C3%A7%C3%B5es) que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como [servidores](https://pt.wikipedia.org/wiki/Servidor), e os requerentes dos serviços, designados como [clientes](https://pt.wikipedia.org/wiki/Cliente_(computa%C3%A7%C3%A3o)).





Na atento utilizamos o AzureDevOps. ([www.atentoalm.visualstudio.com](http://www.atentoalm.visualstudio.com))

1. GIT permite aos desenvolvedores ter uma infinidade de ramos de código completamente independente. Criação, exclusão e fusão desses ramos é simples e não leva tempo.
2. No GIT, todas as operações são atômicas. Isso significa que uma ação pode ter sucesso ou falhar (sem fazer nenhuma alteração). Isso é importante porque em alguns sistemas de controle de versão (como o CVS) onde as operações não são atômicas, se uma operação de repositório é suspensa, ela pode deixar o repositório em um estado instável.
3. No GIT, tudo é armazenado dentro da pasta .git. Isso não é o mesmo em outros VCS como SVN e CVS onde os meta dados de arquivos são armazenados em pastas ocultas (por exemplo, .cvs, .svn, etc.)
4. GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um *commit* é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.
5. Outra característica presente no GIT é sua **área de teste** ou **índice**. Na área de preparação, os desenvolvedores podem formatar *commits* e receber feedback ​​antes de aplicá-los.
6. **GITHUB**

GitHub servidor de repositórios GIT.

Surgiu em 2008. Comprado pela Microsoft em 2018.

Grandes empresas usam o GITHUB. Por exemplo:

1. <https://github.com/oracle>
2. <https://github.com/apple>
3. <https://github.com/Microsoft>
4. <https://github.com/twitter>
5. <https://github.com/Netflix>
6. <https://github.com/facebook>
7. <https://github.com/Instagram>

Além deles vale resaltar seus maiores concorrentes.

* GitLab
* BitBucket
* SourceForge

Nesse lab utilizaremos o GITHUB, opcionalmente.

O fonte disponibilizado e o passo a passo do WorkShop estará disponivel no link abaixo:

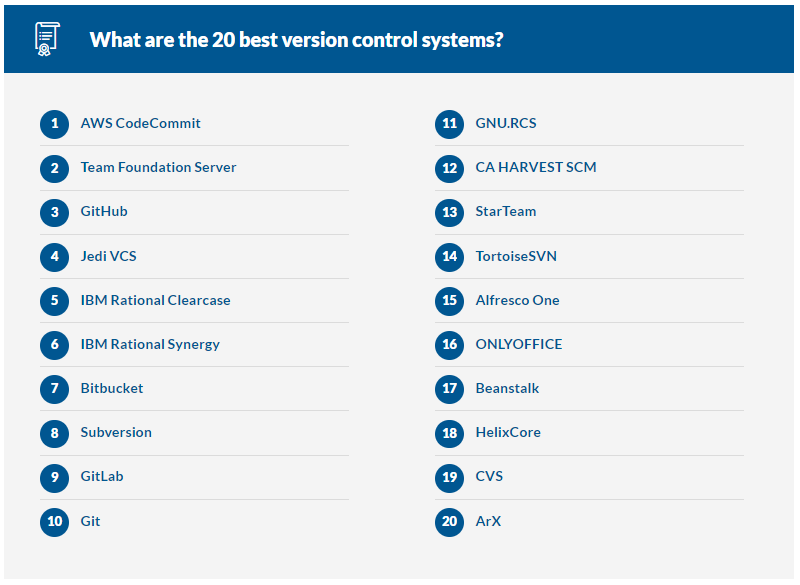
**AQUI VAI O LINK DO GITHUB DO CURSO (CRIAR)**

1. **Difusão no mercado**

GIT está entre os 10 melhores controle de versões atuais.

Essa estatistica é do ano de 2019, e pode ser consultada no link abaixo:

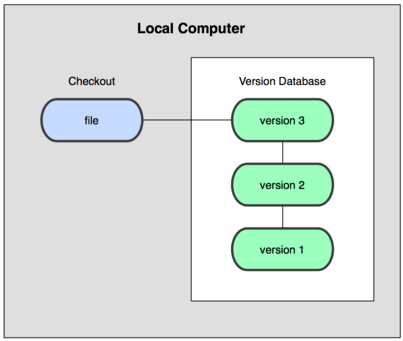
<https://financesonline.com/version-control-systems/>



1. **WORKFLOW**
2. Sistemas de Controle de Versão Locais
3. Sistemas de Controle de Versão Centralizados
4. Sistemas de Controle de Versão Distribuidos

De forma simples o que cada um é, e o que temos como padrão na ATENTO:

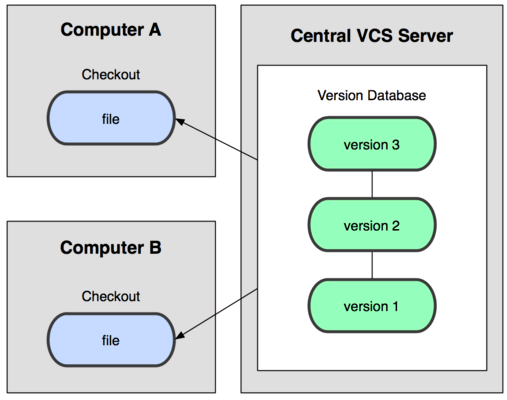
1. Sistemas de Controle de versão Locais



*Fonte:* [*https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o*](https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o)

O método preferido de controle de versão por muitas pessoas é copiar arquivos em outro diretório.Esta abordagem é muito comum por ser tão simples, mas é também muito suscetível a erros. É fácil esquecer em qual diretório você está e gravar acidentalmente no arquivo errado ou sobrescrever arquivos sem querer.

1. Sistemas de Controle de versão centralizados

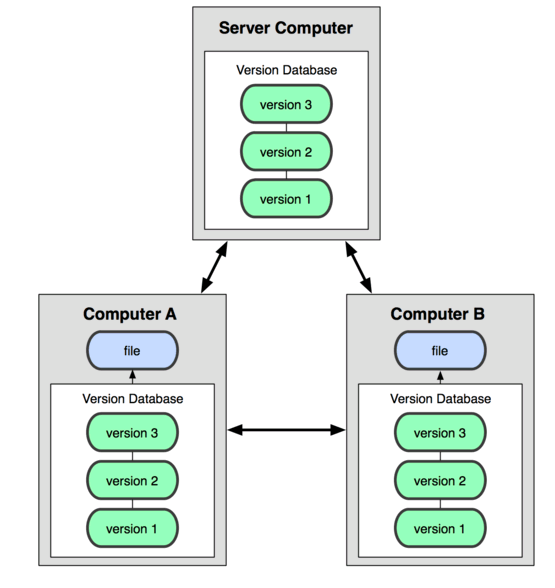


Outro grande problema que as pessoas encontram estava na necessidade de trabalhar em conjunto com outros desenvolvedores, que usam outros sistemas.

Para lidar com isso, foram desenvolvidos Sistemas de Controle de Versão Centralizados 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.

Entretanto, esse arranjo também possui grandes desvantagens**. O mais óbvio é que o servidor central é um ponto único de falha**. Se o servidor ficar fora do ar por uma hora, ninguém pode trabalhar em conjunto ou salvar novas versões dos arquivos durante esse período.

1. Sistemas de Controle de Versão Distribuídos.



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.

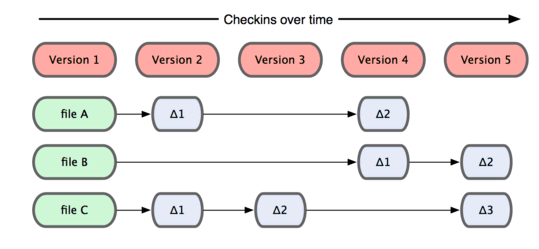
Além disso, muitos desses sistemas lidam muito bem com o aspecto de ter vários repositórios remotos com os quais eles podem colaborar, permitindo que você trabalhe em conjunto com diferentes grupos de pessoas, de diversas maneiras, simultaneamente no mesmo projeto. Isso permite que você estabeleça diferentes tipos de workflow que não são possíveis em sistemas centralizados, como por exemplo o uso de modelos hierárquicos.

A maior diferença do GIT para os demais version control system é a forma que o GIT trata os dados.

Os demais armazenam informação como uma lista de mudança por arquivo

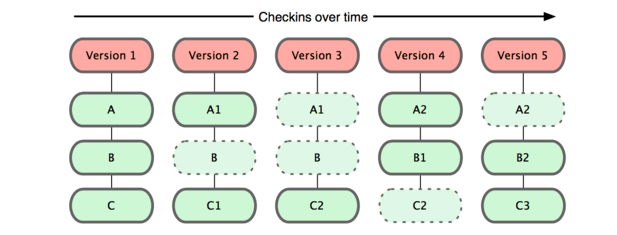
Conjunto de mudanças feitas a cada arquivo ao longo do tempo.

Mudanças em uma versão inicial de cada arquivo.



Git não pensa ou armazena sua informação dessa forma.

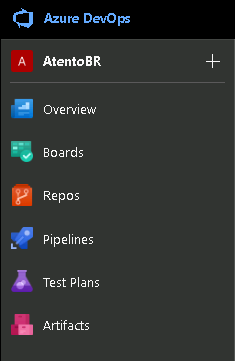
Ao invés disso, o Git considera que os dados são como um conjunto de snapshots (captura de algo em um determinado instante, como em uma foto) de um mini-sistema de arquivos. Cada vez que você salva ou consolida (commit) o estado do seu projeto no Git, é como se ele tirasse uma foto de todos os seus arquivos 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.



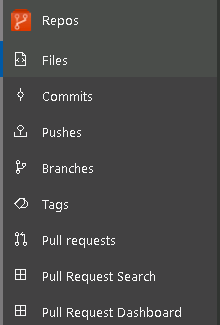
O workflow básico do Git pode ser descrito assim:

1. Você modifica arquivos no seu diretório de trabalho.
2. Você seleciona os arquivos, adicionando snapshots deles para sua área de preparação.
3. Você faz um commit, que leva os arquivos como eles estão na sua área de preparação e os armazena permanentemente no seu diretório Git.
4. **Repositórios GIT no AzureDevOps (AtentoALM)**

Após acessar o projeto AtentoALM seu menu lateral aparecerá dessa maneira:



Clique em REPOS.

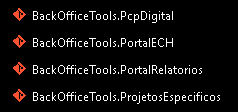


Todos os repositórios tem politicas de segurança.

Portanto somente quem está dentro dos grupos, tem permissões condizentes com a sua função.

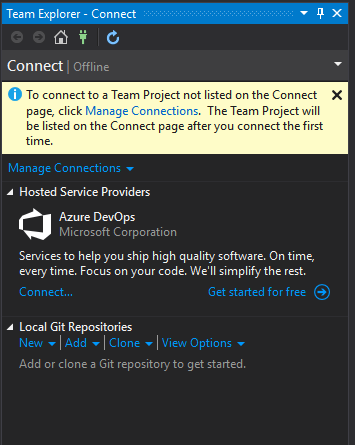
Todas são separadas por Pilar.





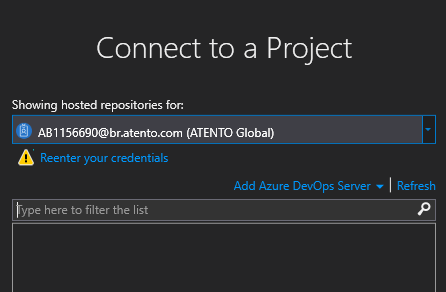
Link na Wiki:

1. **GIT no VISUAL STUDIO**



Para conectar acesse o Manage Coonections.

Em seguida



Link na Wiki:

1. **Branches**

Branches significa ramificação.

Quando você cria um branch significa que vai sair da linha principal.

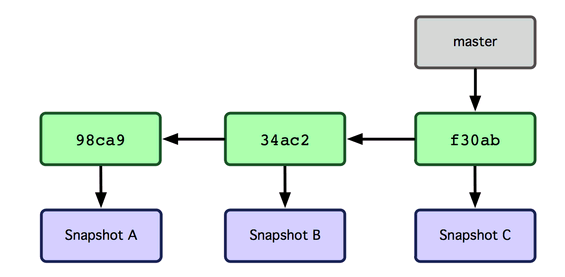
E continuará trabalhando sem bagunça-la.

Esse é um grande diferencial, nos demais controles de versionamentos se faz necessário copiar todo o fonte o que leva um vasto tempo.

A forma como o Git cria branches é leve, fazendo com que as operações com branches sejam praticamente instantâneas e a alternância entre os branches seja tão rápida quanto. Ao contrário de muitos outros VCSs, o Git incentiva um fluxo de trabalho no qual se fazem branches e merges com frequência.

Conforme falamos o GIT cria snapshot de conteúdos e insere um ponteiro a cada commit realizado.

Toda vez que você realiza o commit o ponteiro chamado MASTER anda. Conforme imagem abaixo:



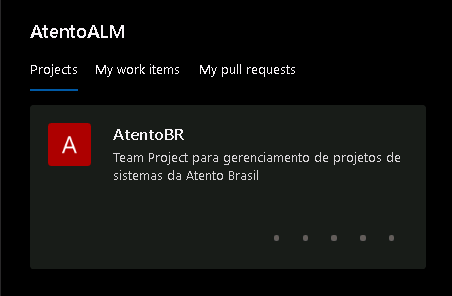
**Na prática**

Vamos exercitar.

**Passo1)**

Acesse: [https://atentoalm.visualstudio.com](https://atentoalm.visualstudio.com/)

Clique em AtentoBR



**Passo 2)**

Agora, clique em Repos no menu lateral esquerdo:

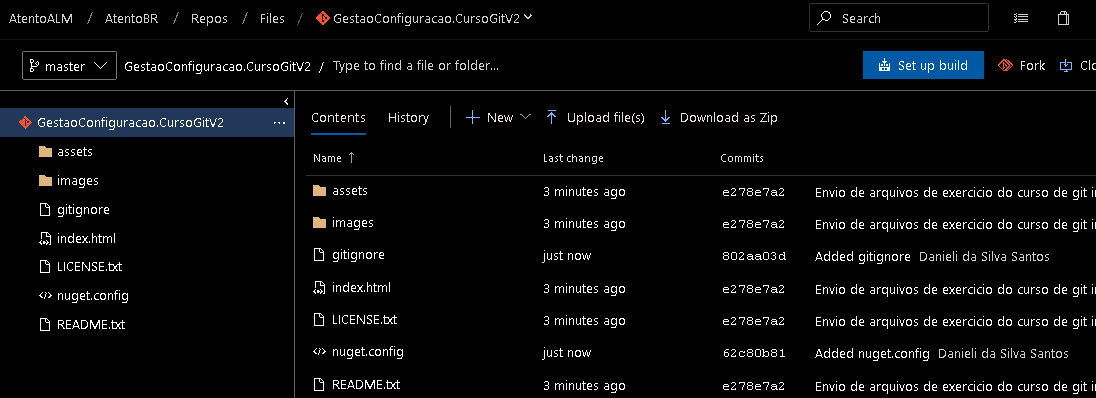


Em seguida clique no menu superior e procure por: GestaoConfiguracao.CursoGitV2

(Esse acesso somente estará permitido durante o dia e horário do curso)



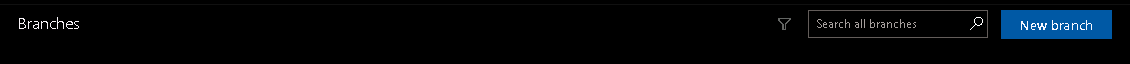
Essa é a interface que você visualizará a principio:



Nesse campo é onde alteramos o branch, por politica de segurança conforme falamos nada é feito diretamente no branch.

Portanto é necessário criar um branch.

Clique no menu branches e em seguida new branch, inclua seu primeiro nome e abreviatura do pilar.



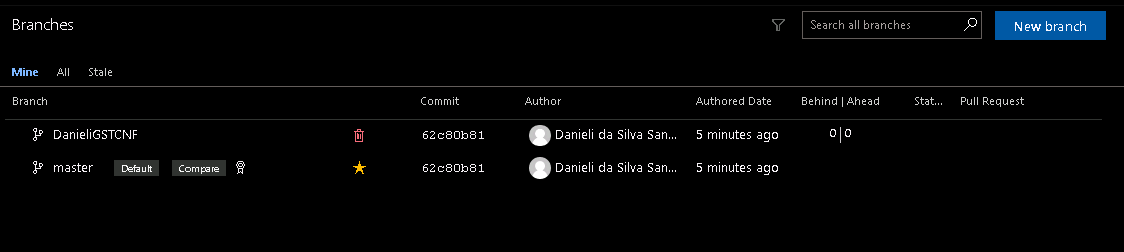
Exemplo: Danieli GSTCNF

MariaBKO

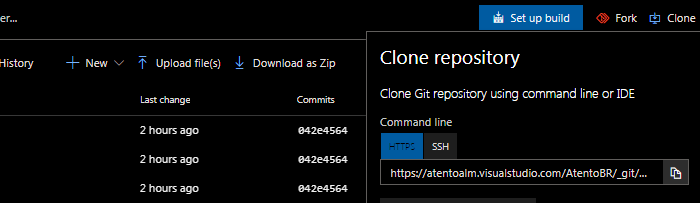
MariaCNTC

Passo 3)

Agora que criado o novo branch vamos clonar.



Para clonar clique com no menu clone que fica em: Files > (Canto direito superior ) Clone



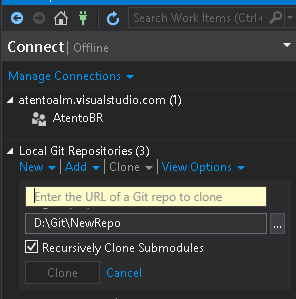
Passo 4)

Abra o visual studio.

No Visual studio vamos clonar o caminho copiado no passo anterior.

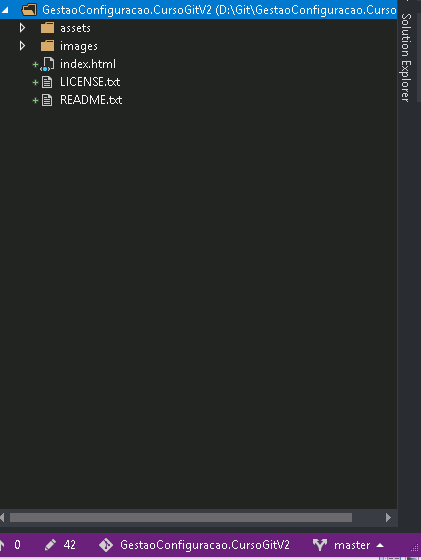
Clique no manage connections (tomada verde). 

Agora clique em clone, e cole o caminho copiado.



Defina em qual local deseja salvar seu repositório.

No menu inferior direito, alterne a branch de máster para a remota que contém seu nome, clique duas vezes para checkout da máster e checkin do seu.



1. Commit

Commits são

1. Histórico

Para visualizar o histórico no Azure Devops

Para visualizar via código:

Hash