* O que é controle de versão?

Controle de versão (SCV), são softwares criados para auxiliar equipes desenvolvedoras de software a manter o controle de suas modificações, além de conseguirem modificar comandos ao mesmo tempo, além de poder ver onde e quem instalou um erro no software através das últimas versões do software salvas momentaneamente em um banco de dados.

<https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Sobre-Controle-de-Vers%C3%A3o>

<https://www.atlassian.com/br/git/tutorials/what-is-version-control>

* O que é o Git?

O GIT é um controle de versão (SCV) criado por Linus Torvalds, criador também do sistema operacional de código aberto Linux. O GIT começou a ser desenvolvido em 3 de abril de 2005, e lançado em 7 de abril de 2005, pelo fato de Linus não suportar a lerdeza de outros SCV’s que utilizava a anos para a criação do Kernel Linux, por isso decidiu criar um melhor em todos os aspectos, e o projeto foi um sucesso, sendo usado pela maioria das empresas ao redor do mundo.

<https://git-scm.com/>

<https://pt.wikipedia.org/wiki/Git>

<https://www.hostinger.com.br/tutoriais/tutorial-do-git-basics-introducao?__cf_chl_jschl_tk__=ccd632c26f599518ec12c724089d622640a7f060-1616523090-0-AbyUBKtKltEcI3FTIErOxJXLad4oU_IYdjwPiSoybtchLBsAKQW8Dc21mUUzHNPxqT2_YTVsXv0s5yBNmRn3XgkdUrXeT-vlhCBg3cR293QniXxO1P8raH58bUtpcHCEhlF5Fu--bgguNxtBXB1ka7RGSD8LWNg596gB08uVecSsWiykiGMfPZDypRnYJI_IadSCQ47jOapVyZeoj6EhuDpzcBKHUN4kM62jD6lJzdqnr1gBQQjVRmvOXZ9ovNRl__trId48Oy-4OTsW4ryWegEl7MlEh2PR5VtVfZuZduCvcaIfC8GRTT2Ar3N8Qu252R_dbgOfbNaPRJnIQH1Dx2f06kg6JbcI9KVSpyWIIGDf92otryYO_ivcRpDFutzCig>

* O que é GitHub?

GitHub, é basicamente o mesmo sistema do GIT, o que muda é o compartilhamento dos arquivos. Por exemplo, enquanto uma pessoa mexe no software aqui do Brasil, outra pessoa no Japão, tem acesso a esse software também, e pode fazer alterações paralelamente com essa pessoa aqui do Brasil. Quer um exemplo para ficar mais claro? No teams, quando um professor posta um arquivo em powerpoint ou word, conseguimos manipulá-lo juntamente com os demais da sala.

<https://www.zup.com.br/blog/git-github-e-gitlab>

<https://www.webdevdrops.com/git-bash-como-instalar-usar/>

<https://rockcontent.com/br/blog/o-que-e-github/>

* O que é GitLab?

O GitLab é basicamente um GitHub, sua principal diferença, é que o GitLab permite que os desenvolvedores armazenem seus projetos em seus próprios servidores, e não em um servidor de terceiros, podendo até em certos casos ter que pagar pela hospedagem dele nesses servidores paralelos.

<https://pt.wikipedia.org/wiki/GitLab>

* Explicação de como criar uma conta em um dos serviços.

Este é o método para criar uma conta no GIT.

Escreva os comandos a seguir com suas informações:

git config --global user.name 1"Seu nome normal ou nick name"

git config --global user.email "exemplo@seuemail.com.br"

<https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Configura%C3%A7%C3%A3o-Inicial-do-Git>

* Como é a instalação do git no Windows.

Para instalar o Git no Windows é bem simples, igualmente a qualquer software disponível na internet.

1. Abra o site do Git, vá em downloads, e clique na opção Windows, ou clique no link ao lado para ir direto: <https://gitforwindows.org/>;
2. Faça o download do arquivo;
3. Execute o arquivo;
4. Passo a Passo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Primeira tela: mostra a política de privacidade do software. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Segunda tela: aqui você pode escolher onde deseja que o Git seja instalado. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

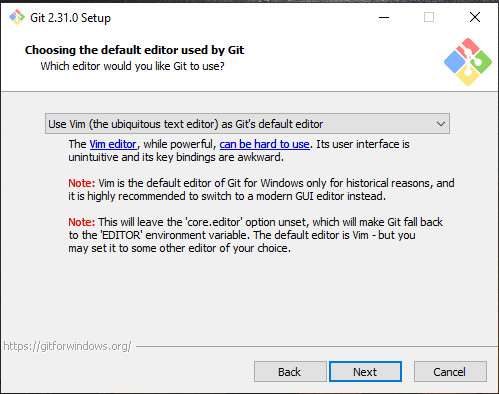
Descrição gerada automaticamente

Terceira tela: Aqui você pode escolher algumas opções de uso a mais, ou remover outras que acha não ser necessário. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Quarta tela: Aqui você escolhe o nome que deseja que apareça no menu INICIAR quando for em busca do Software, mas, caso não queira esse atalho, é só marcar a opção abaixo que está escrito: “Don’t create a Start Menu Folder”. Depois disso, clique em Next.



Quinta tela: Aqui você pode escolher o tipo de editor que deseja utilizar para dar os comandos ao GIT. Certamente algum vai ser útil para você, coma possibilidade de usar editores mais simples, como Notepad++ e visual studio code. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Sexta tela: Aqui você pode escolher o tipo de nomenclatura para o seu primeiro repositório:

Ex: “Repositório”.main, “Repositório”.master, “Repositório”.trunk, etc.

Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Sétima tela: Aqui você pode escolher os tipos de sistemas para executar diferentes tipos de comando Git de seu agrado. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Oitava tela: Aqui você pode escolher o server que vai cuidar do transporte de suas informações pela internet, no caso a validação de seu arquivo. Depois disso, clique em Next.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Nona tela: Nessa tela, você pode deixar ou não, o GIT fazer a conversão dos tipos de arquivos para a manipulação das informações.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Décima tela: Aqui você pode definir o tipo de emulador que quer que o Git usem, pode ser o CMD, ou o MinTTY.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Décima primeira tela: Aqui você pode escolher o tipo de padrão por trás do GIT Pull.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Décima segunda tela: Aqui você define o gerenciador de credenciais que você quer que opere no GIT.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Décima terceira tela: Aqui você pode escolher um formato de cache rápido no GIT para assim, melhorar seu desempenho em alguns casos.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Décima quarta tela: Aqui você pode habilitar um suporte experimental disponibilizado pelo GIT de novos comandos e ferramentas que estão sendo desenvolvidas para o GIT.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Décima quinta tela: agora é só aguardar o download, e pronto, o GIT está instalado em sua máquina.

Agora se você não vê necessidade em nenhum dos itens acima e confia nas indicações do Git, é aquele velho jeito comum entre os programadores, “next, next, finish”.

Fonte de Fotos do tutorial: Autor Próprio, legendas também, nada foi copiado, link abaixo foi encontrado depois de todo o processo estar concluído.

<https://dicasdeprogramacao.com.br/como-instalar-o-git-no-windows/>

* Como configurar o git.

Após os comandos citados acima de como criar uma conta, vem alguns comandos de como fazer algumas configurações básicas no GIT usando também o “git config”. Com esse comando você pode alterar seu git, e até os repositórios que serão vistos pelos outros usuários.

/etc/gitconfig: vale para todos os usuários e repositórios utilizando o sistema.

~/gitconfig ou ~/.config/git/config: com isso você altera as configurações apenas do seu Git, não interfere em mais nada além disso.

Config no diretório git (git/config) de qualquer repositório, essa configuração só altera esse repositório em específico.

O git em partes de comandos específicos para certas partes funciona bem parecido com um CSS da vida, caso reescreva um comando diferente, ele se sobrepõe sobre o outro que foi passado antes.

Você pode alterar o seu editor do texto por comandos também, com o código a seguir:

$ git config - -global core.editor “nome do editor”

Caso não faça isso, o editor que irá aparecer será o Vim, semelhante ao CMD.

Obs: “ - -global” usando a opção global, você só precisará fazer esse comando uma vez na vida, se não, terá que repetir toda vez que for fazer comandos digitados.

E por fim, outro comando básico é o : “$ git config - -list”, esse comando mostra todas as suas informações inseridas e suas configurações realizadas no git.

* Quais os principais comandos do git?

**Git Clone**

O git clone é um comando para baixar código fonte de repositórios em servidores.

Git clone<https://link>

**Git Branches**

Comando utilizado para criar, listar e excluir branches.

Comando para criar: Comando para criar um branch local (no seu pc)

Git branch <nome do branch>

Para criar uma branch no repositório remoto: execute o camando a seguir:

git push -u <remote> <nome-da-branch>

Para ver as branches:

git branch or git branch --list

Deletando uma branch:

git branch -d <nome-da-branch>

**Git Checkout**:

Este é um dos comandos Git mais usados. Para trabalhar em uma branch, primeiro você precisa mudar para ela.

Usamos o git checkout principalmente para alternar de um branch para outro. Também podemos usá-lo para verificar arquivos e commits:

git checkout <nome-da-branch>

Há também um comando de atalho que permite criar e trocar para um branch ao mesmo tempo:

git checkout -b <nome-da-branch>

**Git Status**

O comando status do Git fornece todas as informações necessárias sobre o branch atual.

git status

**Git Add**

Quando criamos, modificamos ou excluímos um arquivo, essas alterações ocorrerão em nosso ambiente local e não serão incluídas no próximo commit (a menos que alteremos as configurações).

Precisamos usar o comando git add para incluir as alterações de um arquivo em nosso próximo commit.

Para adicionar um arquivo:

git add <arquivo>

Para adicionar tudo de uma vez:

git add -A

O comando git add não altera o repositório e as alterações não são salvas até usarmos o git commit.

**Git Commit**

Este comando é como definir um ponto de verificação no processo de desenvolvimento, para o qual você pode voltar mais tarde, se necessário.

git commit -m "mensagem explicando a mudança no código"

**Git Push**

Após confirmar as alterações, a próxima coisa que você deseja fazer é enviar as alterações para o servidor remoto.

O comando git push envia e salva suas confirmações no repositório remoto.

git push <remote> <nome-do-branch>

No entanto, se seu branch for criado recentemente, você também precisará fazer upload do branch com o seguinte comando:

git push -u origin <nome-do-branch>

**Git Pull**

O comando git pull é usado para obter atualizações do repositório remoto.

Este comando é uma combinação de git fetch e git merge, o que significa que, quando usamos o git pull, ele recebe as atualizações do repositório remoto git fetch e aplica imediatamente as alterações mais recentes no seu local git merge.

git pull <remote>

**Git Revert**

Existem várias maneiras de desfazer nossas alterações local ou remotamente (depende do que precisamos), mas devemos usar esses comandos com cuidado para evitar problemas.

Uma maneira segura de desfazer os commits é usando git revert.

git revert 'número do hash'

O número do hash pode ser conseguido pelo comando:

git log -- oneline

**Git Merge**

Quando você conclui o desenvolvimento em sua branch e tudo funciona bem, a etapa final é mesclar as branches, isso é feito com o comando git merge.

git merge <nome-da-branch>

<https://blog.geekhunter.com.br/comandos-git-mais-utilizados/>

<http://guides.beanstalkapp.com/version-control/common-git-commands.html>

<https://www.atlassian.com/br/git/tutorials/syncing/git-pull#:~:text=O%20comando%20git%20pull%20%C3%A9%20a%20combina%C3%A7%C3%A3o%20de%20dois%20outros,fluxo%20de%20trabalho%20de%20merge>.

* O que é git pull?

Git pull é a combinação de dois comandos, Git fetch e o git merge. No git fetch, ele basicamente vai atras das últimas atualizações do repositório, e o git merge, pega essas informações e insere dentro da branch local (pc do usuário que executou o comando), assim o arquivo pode ser manipulado com as últimas alterações.

<https://pt.stackoverflow.com/questions/3231/qual-a-diferen%C3%A7a-entre-os-comandos-git-pull-e-git-fetch>

* O que são branches?

Em tradução livre, Banches significa raízes, e essas raízes, são as copias de repositórios criadas para que cada desenvolvedor possa fazer suas alterações sem interferir em outros trabalhos sendo executados por outros desenvolvedores.

Como arrumar uma mensagem errada em um commit

Na linha de comando, vá para o repositório que está com o commit errado e ensira o seguinte comando: “Git commit - - amend” e pressione Enter., quando o editor de texto abrir, arrume a mensagem do commit e salve o mesmo.

<https://docs.github.com/pt/github/committing-changes-to-your-project/changing-a-commit-message#:~:text=Na%20linha%20de%20comando%2C%20navegue,incluindo%20um%20trailer%20no%20commit>.

* Explicação sobre como um grupo de faculdade pode usar o Git para colaborar em um projeto, como, não sei, talvez um Projeto Integrador.

Com o Git, todos teriam acesso ao arquivo ao mesmo tempo, sendo assim, não seria necessário ficar passando o arquivo compactado de um para o outro, empacando o grupo caso um demorasse a fazer sua parte, etc. Teríamos o controle de nossas alterações no decorrer do processo, além de reduzir a possibilidade da perda do arquivo.