**O que é Git?**

O Git é um **sistema de controle de versão distribuído** que ajuda os desenvolvedores a colaborar em projetos de qualquer escala. Linus Torvalds, o desenvolvedor do kernel do Linux, criou o Git em 2005 para ajudar a controlar o desenvolvimento do kernel do Linux.

**O que é um Sistema de Controle Distribuído?**  
  
Um sistema de controle de versão distribuído é um **sistema que ajuda você a acompanhar as alterações feitas nos arquivos do seu projeto.** Esse histórico de alterações reside em sua máquina local e permite que você reverta para uma versão anterior de seu projeto com facilidade, caso algo dê errado.

O Git facilita a colaboração. Todos na equipe podem manter um backup completo dos repositórios em que estão trabalhando em sua máquina local. Então, graças a um servidor externo como BitBucket, **GitHub** ou GitLab, eles podem armazenar com segurança o repositório em um único local.

# **Principais comandos do GIT:** **git config:**

# Este comando define o nome do autor e o endereço de e-mail, respectivamente, para serem usados com seus commits. Obs: Você deve usar users e e-mail do GitHub. Uso: git config –global user.name “[name]”

Uso: git config –global user.email “[email address]”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **git init:** Esse comando é usado para iniciar um novo repositório. Uso: git init [repository name] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **git clone:**

Git clone é um comando para baixar o código-fonte existente de um repositório remoto (como o Github, por exemplo). Em outras palavras, Git clone basicamente faz uma cópia idêntica da versão mais recente de um projeto em um repositório e a salva em seu computador.  
  
Uso: git clone [url]  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
**git add:**  
  
Este comando adiciona um arquivo à área de preparação. Precisamos usar o comando git add para incluir as alterações de um arquivo(s) em nosso próximo commit.  
  
Uso: git add \*   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **git commit:** Este comando registra ou captura o arquivo permanentemente no histórico da versão. Git commit é como definir um ponto de verificação no processo de desenvolvimento para o qual você pode voltar mais tarde, se necessário. Também precisamos escrever uma mensagem curta para explicar o que desenvolvemos ou mudamos no código-fonte. Uso: git commit -m “[update do README.me]” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **git push:**

Depois de confirmar suas alterações, a próxima coisa que você deseja fazer é enviar suas alterações para o servidor remoto. Git push carrega seus commits para o repositório remoto.  
  
No entanto, se sua ramificação for criada recentemente, você também precisará fazer o upload da ramificação com o seguinte comando:  
  
Uso: git push –set-upstream <remote> <branch-name>   
  
Além disso, você também pode fazer:  
  
Uso: git push –u origin <branch-name>   
  
  
Este comando envia as alterações confirmadas da ramificação principal para seu repositório remoto.  
  
Uso: git push [variable name] master



Este comando envia os commits do branch para seu repositório remoto.  
  
Uso: git push [variable name] [branch]



Este comando envia todas as ramificações para seu repositório remoto.  
  
Uso: git push –all [variable name]



Este comando exclui uma ramificação em seu repositório remoto.  
  
Uso: git push [variable name] :[branch name]



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **git diff:**

Este comando mostra as diferenças de arquivo que ainda não foram testadas.  
  
Uso: git diff  
   
Este comando mostra as diferenças entre os arquivos na área de preparação e a versão mais recente presente.  
  
Usage: git diff –staged

  
Este comando mostra as diferenças entre os dois ramos mencionados.

Uso: git diff [first branch] [second branch]

Este comando mostra as diferenças de arquivo que ainda não foram testadas.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**git reset:**

Esse comando permite desfazer ou "redefinir" as alterações de código feitas anteriormente.   
  
Uso: git reset [file]   
  


Este comando desfaz todos os commits após o commit especificado e preserva as alterações localmente.  
  
Uso: git reset [hash]

Comando Git Reset - Comandos Git - Edureka  
Este comando descarta todo o histórico e volta para o commit especificado.  
  
Uso: git reset –hard [hash]

Comando Git Reset - Comandos Git - Edureka\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **git status:**

Este comando mostrará o status do repositório atual, incluindo arquivos testados, não testados e não rastreados.  
  
Podemos coletar informações como:

* Se a ramificação atual está atualizada
* Se há algo para cometer, empurrar ou puxar
* Se há arquivos testados, não testados ou não rastreados
* Se há arquivos criados, modificados ou excluídos

Uso: git status  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**git log:**

Este comando é usado para listar o histórico da versão da ramificação atual.  
  
Uso: git log



Este comando lista o histórico da versão de um arquivo, incluindo a renomeação de arquivos também.  
  
Uso: git log –follow [file]



## Este comando mostra o histórico do commit incluindo todos os arquivos e suas alterações Uso: git log p Podemos usar **--graphpara** obter o log de confirmação para mostrar como um gráfico. Além disso,**--oneline** limitará as mensagens de confirmação a uma única linha. Uso: git log –graph –oneline \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **git show:** Este comando mostra os metadados e alterações de conteúdo do commit especificado.

Uso: git show [commit]



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
**git branch:**   
  
Branches são altamente importantes no mundo git. Ao usar ramificações, vários desenvolvedores podem trabalhar em **paralelo no mesmo projeto simultaneamente**. Podemos usar o comando git branch para criar, listar e deletar branches.  
  
**Criando uma nova ramificação:**

Uso: git branch [branch name]

Este comando criará uma ramificação **localmente** . Para enviar a nova ramificação para o repositório remoto, você precisa usar o seguinte comando:  
  
Uso: git push –u <remote> <branch-name>

**Listagem das branchs:**   
  
Uso: git branch or git branch –list

**Excluindo um ramo:**

Uso: git branch -d [branch name]   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
**git check-out:**   
  
Este também é um dos comandos Git mais usados. Para trabalhar em um ramo, primeiro você precisa mudar para ele. Usamos **git checkout** **principalmente para mudar de um branch para outro**. Também podemos usá-lo para verificar arquivos e commits.  
  
Uso: git checkout <branch name>

Existem algumas etapas que você precisa seguir para alternar com sucesso entre as ramificações:

* As alterações em sua ramificação atual devem ser confirmadas ou armazenadas antes de você alternar
* A filial que você deseja verificar deve existir em seu local

Este comando cria uma nova ramificação e também alterna para ela.  
  
Uso: git checkout –b <branch name>  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
**git merge:**  
  
O Git merge basicamente integra seu branch de recursos com todos os seus commits de volta ao branch dev (ou master). É importante lembrar que primeiro você precisa estar no branch específico que deseja mesclar com o branch de recursos.

Uso: git merge [branch name]

Este comando mescla o histórico da ramificação especificada na ramificação atual.

Comando Git Merge - Comandos Git - Edureka

**Primeiro você deve mudar para o branch qualquer:**  
  
Uso: git checkout dev

**Antes de mesclar, você deve atualizar sua ramificação de desenvolvimento local:**  
  
Uso: git fetch

**Por fim, você pode mesclar sua ramificação de recurso em dev:**  
  
Uso: git merge [branch-name]