

# **Controle de Versionamento - Introdução ao Git**

Felipe Souza Marra  
Giovanni Favorin de Melo  
Rafael Riki Ogawa Osiro

1.	Nível básico	3
2.	Nível Intermediário	35
3.	Nível avançado	37
4.	Outros recursos	39
5.	Como criar uma conta no GitHub?	40

## 1. NÍVEL BÁSICO

O nível básico tem como objetivo de introduzir os conceitos básicos do controle de versionamento, os seus benefícios e como aplicá-los.

### Conceitos Básico

N	Questões	Recursos
1	O que é controle de versionamento?	
	No desenvolvimento de <i>software</i> , o controle de versionamento tem como objetivo de gerenciar diferentes versões de um arquivo. Os sistemas de controle de versão são utilizados pelos desenvolvedores a rastrear e gerenciar as alterações do arquivo, fazendo com que criem vários ambientes de desenvolvimento.	1. <a href="#">O que é control e de versão?</a> 2. <a href="#">O que é Control e de Versão?</a>
2	Por que utilizar o controle de versionamento?	
	A utilização do controle de versionamento ajuda as equipes de desenvolvedores de software a trabalharem com rapidez e inteligência. Também não há a necessidade de ficar enviado o código fonte por e-mail, pendrive, nuvem e entre outros métodos. Além de ser trabalho ficar enviado o programa por esses métodos, pode ser perigoso, uma vez que pessoas mal intencionadas podem roubar informação e utilizar o código para atividades ilegais.	1. <a href="#">O que é control e de versão?</a> 2. <a href="#">O que é Control e de Versão?</a>
3	Por que o versionamento é importante?	

<p>O esquema de sequência na ordem crescente ajuda a evitar a confusão sobre qual versão do software está sendo utilizada. Uma coisa valiosa em um mundo de ameaças virtuais constantes. Além disso, existem outros fatores que tornam o versionamento uma parte estratégica do desenvolvimento de softwares. Conheça os principais abaixo:</p> <ul style="list-style-type: none"> <li>• Inclusão ou extensão de requisitos que já existem: Códigos de segurança existentes podem precisar de patches para a correção de vulnerabilidades encontradas e, ao implementá-los, a versão deve ser alterada para o conhecimento dos desenvolvedores e usuários. Novas funcionalidades também podem ser atribuídas gerando outra versão para o software. Mas, nesse caso, o usuário pode optar em trabalhar com a antiga caso não goste ou não precise delas.</li> <li>• Acompanhamento de versões: Todas as mudanças efetuadas na fonte são rastreadas indicando o problema corrigido e o aprimoramento introduzido, bem como os profissionais responsáveis e o motivo das alterações. Ou seja, servirá como um mecanismo para a devida diligência nos projetos de software. Nesse caso, um sistema de controle de versão será essencial para todo tipo de desenvolvimento, seja particular seja colaborativo. A capacidade de acompanhar cada mudança conforme é realizada e de reverter-la quando necessário pode fazer toda a diferença para um processo bem gerenciado e controlado.</li> <li>• Mudanças na arquitetura: Com as tecnologias de software avançando ininterruptamente, mudanças na estrutura devem ser necessárias durante o desenvolvimento e após o lançamento. Dessa forma, uma mudança na codificação das páginas pode alterar toda a estrutura do software, implicando em uma versão mais moderna. Essas mudanças podem ser mais facilmente acompanhadas com um sistema de versionamento.</li> <li>• Correção de bugs: O controle de versão fornece um histórico completo de cada commit feito por desenvolvedor. Uma vez que as ferramentas modernas permitem trabalhar em múltiplas alterações de arquivos ao mesmo tempo, isso favorece um fácil acompanhamento das mudanças relacionadas. Também se torna possível fazer uma regressão nas mudanças. Se você perceber que alguma alteração de melhoramento não funcionou bem, pode simplesmente reverter uma por vez até encontrar qual causou o problema.</li> <li>• Ajustes estéticos: O Google segue beneficiando páginas com layouts responsivos na internet, e isso também deve começar a ser exigido dos softwares pelos usuários. Isso implicará em mudanças significativas, transformando o bom funcionamento em dispositivos móveis e online. Além disso, mudanças de cores ou reposicionamentos de textos,</li> </ul>	<p>1. <a href="#"><u>Entend a por que versionamento de softwar e é tão importante</u></a></p>
---	---

imagens e campos de digitação podem ser necessários para melhorar a intuitividade do software. Cada uma dessas mudanças precisa ser acompanhada com rigor e, sem um sistema de controle de versionamento, essa tarefa pode ser dispendiosa.

Por fim, todo bom projeto precisa de um mecanismo para controlar de forma efetiva as versões durante e após os processos de criação, desenvolvimento e lançamento. Então, antes de dar início a um projeto desses, o ideal é contar com o apoio de um sistema de versionamento de software. Ele será fundamental para estabelecer uma gestão confiável e precisa. Acredite, o sistema de versionamento pode diminuir em até 80% o esforço de entrega de um software, poupando tempo e recursos da empresa no processo.

## Conceitos Básico

4	Qual o objetivo de utilizar o versionamento?	
	O controle de versionamento vem com o objetivo de criar vários ambientes de desenvolvimento, juntar código de vários programadores de forma remota e fazer o controle de versão do código.	<ol style="list-style-type: none"> <li>1. <a href="#">O que é control e de versão?</a></li> <li>2. <a href="#">O que é Control e de Versão?</a></li> </ol>
5	Como funciona o controle de versão?	
	<p>Existe o repositório principal que possui os históricos de atualizações e o repositório clonado localizado na área de trabalho do contribuidor.</p> <p>A comunicação desses repositórios têm por padrão o commit (enviar atualizações) e o update (receber atualizações).</p> <p>Vale ressaltar que isso é padrão. Não importa o tipo de versionamento, a diferença está em como cada um será utilizado.</p>	<ol style="list-style-type: none"> <li>1. <a href="#">Conheça diferentes formas de versionar projetos</a></li> </ol>
6	Quais são os tipos de controle de versão?	
	<p>Existem dois tipos de controle de versão: centralizado e descentralizado/distribuído. Conheça os detalhes a seguir.</p> <p>Centralizado: apenas um servidor principal tem uma cópia local para cada contribuidor. Logo, a única forma de comunicação delas é pelo servidor principal.</p> <p>Descentralizado: não depende de um servidor central, isso significa que o desenvolvedor terá um repositório na área de trabalho com toda a base de dados, podendo percorrer outros branches, gerar ou reverter versões do código-fonte, trabalhar em versões de testes etc.</p> <p>Por essa liberdade, faz-se necessário um controle ao contribuidor, limitando o que ele pode fazer no código, a fim de evitar a quebra da aplicação ou sua possível exclusão.</p>	<ol style="list-style-type: none"> <li>1. <a href="#">Conheça diferentes formas de versionar projetos</a></li> </ol>
7	Quais são os sistemas de versionamento mais utilizados?	

## Conceitos Básico

	<ul style="list-style-type: none"> <li>• CVS: pode não ser o mais usado atualmente, mas um dia já foi. O CVS é uma das ferramentas de versionamento mais antigas no mercado. Sua primeira versão, desenvolvida em 1968, ainda é muito utilizada pela equipe de desenvolvedores. Recomendo ler a documentação para saber mais sobre (clique no link no final do artigo).</li> <li>• SVN: veio com novas funcionalidades e correção de alguns problemas do CVS. É simples e rápido na sua utilização, possuindo sistema de versionamento centralizado (indicado para equipes pequenas).</li> <li>• Mercurial: é famoso por ser utilizado em grandes empresas, como Meta e Google. É uma ferramenta complexa comparada ao SVN, porém rápida e com medidas de segurança para evitar erros. Está ligado ao sistema de versionamento descentralizado.</li> <li>• Git: é uma das ferramentas mais utilizadas pelos desenvolvedores atualmente, seja em empresas ou projetos individuais. Consegue atingir todos os objetivos de um bom controle de software, também é mais complexa se comparada com as demais ferramentas de versionamento. Está no sistema de versionamento descentralizado.</li> </ul>	1. <a href="#">Conhecendo as diferentes formas de versionar projetos</a>
8	Quais são os benefícios dos sistemas de controle de versão	

## Conceitos Básico

	<ol style="list-style-type: none"><li>1. Um histórico de alterações completo e a longo prazo de todos os arquivos. Isso significa todas as alterações feitas por muitas pessoas ao longo dos anos. As alterações incluem a criação e exclusão de arquivos, assim como as edições em seus conteúdos. Diferentes ferramentas de VCS diferem na maneira de lidar com a renomeação e a movimentação de arquivos, se melhor ou pior. Esse histórico também deve incluir o autor, a data e as notas escritas sobre o objetivo de cada alteração. Ter o histórico completo permite voltar às versões anteriores para ajudar na análise da causa raiz de bugs e é crucial para corrigir problemas nas versões mais antigas do software. Se o software estiver sempre sendo trabalhado, quase tudo poderá ser considerado uma "versão mais antiga" do software.</li><li>2. Ramificação e mescla. O trabalho simultâneo da equipe é certo, mas mesmo os indivíduos que trabalham sozinhos podem se beneficiar da capacidade de trabalhar em fluxos independentes de mudanças. Criar uma "ramificação" nas ferramentas do VCS mantém vários fluxos de trabalho independentes uns dos outros, além de oferecer a facilidade de mesclar esse trabalho de novo, permitindo que os desenvolvedores verifiquem se as alterações em cada ramificação não estão em conflito. Muitas equipes de software adotam a prática de ramificação para cada recurso, ou talvez ramificação para cada versão, ou ambos. As equipes podem escolher entre vários fluxos de trabalho diferentes quando decidem como usar os recursos de ramificação e fusão no VCS.</li><li>3. Rastreabilidade. Ser capaz de rastrear cada alteração feita no software e conectar ao software de gestão de projetos e rastreamento de bugs, como o Jira, e ser capaz de anotar cada alteração com uma mensagem descrevendo o objetivo da mudança ajuda não só com a análise de causa raiz e outras análises forenses. Ter o histórico anotado do código na ponta dos dedos quando você estiver lendo o código, tentando entender o que está fazendo e por que ele foi projetado assim pode permitir que os desenvolvedores façam alterações corretas e harmoniosas que estejam de acordo com o design do sistema pensado para o longo prazo. Em especial, isso pode ser importante para o trabalho efetivo com o código legado e é crucial para permitir que os desenvolvedores calculem o trabalho futuro com precisão.</li></ol>	<ol style="list-style-type: none"><li>1. <a href="#">O que é control e de versão?</a></li></ol>
9	O que é git?	



## Conceitos Básico

	<p>Criado pelo engenheiro de software Linus Torvalds, conhecido por ter desenvolvido, também, o núcleo Linux, o GIT é um Sistema de Controle de Versões Distribuído — ou DVCS.</p> <p>Estes sistemas de controle possuem a função de registrar quaisquer alterações feitas em cima de um código, armazenando essas informações e permitindo que, caso seja necessário, um(a) programador(a) possa regredir a versões anteriores de uma aplicação de modo simples e rápido.</p> <p>Este tipo de sistema também simplifica muito o processo de compartilhamento de um projeto com um time, por exemplo, ou com outros(as) programadores(as).</p>	<p>1. <a href="#">O QUE É GIT: CONCEITOS, PRINCIPAIS COMANDOS E QUAIS AS VANTAGENS?</a></p>
10	Quais ferramentas de controle de versionamento que utiliza o git?	

## Conceitos Básico

	<p>Existem várias ferramentas para várias plataformas. Segundo o Hostinger as melhores ferramentas utilizadas em 2022 são:</p> <ul style="list-style-type: none"> <li>• Usuários de Linux: <ul style="list-style-type: none"> <li>▶ QGit – Git GUI descomplicado para Linux e que não custa um centavo.</li> <li>▶ Gitg – você pode ver seus repositórios e ele permite a realização de operações Git Comuns.</li> <li>▶ Git Force – iniciantes no Git podem fazer uso dessa ferramenta já que ela possui uma interface intuitiva e é gratuita para baixar.</li> </ul> </li> <li>• Usuários de Windows: <ul style="list-style-type: none"> <li>▶ Sourcetree – ótimo para novatos e especialistas em Git. Uma ferramenta poderosa, porém gratuita e simples</li> <li>▶ GitHub for Windows – um Git GUI onde você pode trabalhar em seu projeto, visualizar e rastrear o workflow dos seus repositórios GitHub.</li> <li>▶ Tortoise Git – um Git GUI gratuito e de código aberto para Windows. Simples e direto de usar e pode ser usado com outras ferramentas de desenvolvimento.</li> </ul> </li> <li>• Usuários de Mac: <ul style="list-style-type: none"> <li>▶ GitUp – um ambiente seguro para aprender Git e experimentar. Ele também é gratuito, rápido e fácil de usar.</li> <li>▶ GitBox – gratuito para uso sem fins lucrativos e faz com que trabalhar com Git seja tão fácil quanto chegar um email.</li> <li>▶ Git-Xdev – projetado para ser um Git GUI sustentável e de ponta. Ele é grátis e repleto de recursos para a maioria dos fluxos de trabalho.</li> </ul> </li> <li>• Usuários de Multi-Plataforma: <ul style="list-style-type: none"> <li>▶ Git Kraken – tem uma versão gratuita, confiável e que faz com que o Git seja compreensível e tenha um visual atraente.</li> <li>▶ SmartGit – a interface de fato é inteligente e fácil de usar, ele é gratuito para baixar para uso não comercial.</li> <li>▶ Git Cola – um cliente de Git simples porém poderoso que faz com que o fluxo de trabalho seja rápido e eficiente.</li> <li>▶ Aurees – um cliente gratuito de Git GUI que é fácil de usar e que permite que os usuários trabalhem em operações de Git sem complicações.</li> </ul> </li> </ul>	<p>1. <a href="#">Os Melhores Clientes Git GUI de 2022 para Windows, Linux e Mac</a></p>
11	O que é repositório?	

## Conceitos Básico

	<p>Os repositórios são os ambientes criados para armazenar seus códigos.</p> <p>Você pode possuir um ou mais repositórios, públicos ou privados, locais ou remotos, e eles podem armazenar não somente os próprios códigos a serem modificados, mas também imagens, áudios, arquivos e outros elementos relacionados ao seu projeto.</p> <p>É através dos seus repositórios públicos que outros programadores poderão ter acesso aos seus códigos no GitHub, podendo, inclusive, cloná-los para adicionar melhorias.</p>	<p>1. <a href="#">O QUE É GIT: CONCEITOS, PRINCIPAIS COMANDOS E SUAS VANTAGENS?</a></p>
	O que é <i>Branch</i> ?	
12	<p>Branch é o nome dado a uma versão (ramificação) do projeto. Isso é útil porque possibilita gerenciar múltiplas alterações acontecendo simultaneamente. Por exemplo, podemos fazer com que cada equipe de desenvolvimento</p>	<p>1. <a href="#">O QUE É GIT: CONCEITOS, PRINCIPAIS COMANDOS E SUAS VANTAGENS?</a></p>
	O que é <i>commit</i> ?	
13	<p>O Git commit permite que você crie um commit, ou seja, você consegue guardar o estado do seu repositório naquele momento. Existem diferentes estratégias para fazer commits, mas a ideia principal é que a cada ponto em que o seu código esteja funcionando com uma nova pequena funcionalidade, exista um commit.</p> <p>Dessa forma, ao longo do tempo, você vai conseguir ver uma “história” do seu repositório e o que aconteceu para que chegasse do jeito que está hoje. Um commit, além de mostrar um snapshot do seu repositório, tem outros metadados, como autoria, uma mensagem, timestamp, entre outros.</p>	<p>1. <a href="#">Git commit: confirmando e salvando alterações!</a></p>
	O que é <i>push</i> ?	
14	O comando git push é usado para enviar o conteúdo do repositório local para um repositório remoto. O comando push transfere commits do repositório local a um repositório remoto.	<p>1. <a href="#">Git push</a></p>
15	O que é <i>pull</i> ?	

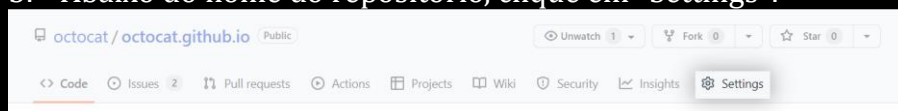
## Conceitos Básico

	<p>O comando <code>git pull</code> é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais. Fazer o merge de alterações upstream remotas no repositório local é algo comum em fluxos de trabalho de colaboração baseados em Git. O comando <code>git pull</code> é a combinação de dois outros comandos, o <code>git fetch</code>, seguido do <code>git merge</code>. No primeiro estágio da operação, o <code>git pull</code> executa o comando <code>git fetch</code>, que abrange a ramificação local para qual a HEAD aponta. Quando o conteúdo é baixado, o <code>git pull</code> insere o fluxo de trabalho de merge. O commit de merge é criado e a HEAD é atualizada para apontar o novo commit.</p>	1. <a href="#">git pull</a>
	O que é <i>merge</i> ?	
16	<p>Mesclagem é o jeito do Git de unificar um histórico bifurcado. O comando <code>git merge</code> permite que você pegue as linhas de desenvolvimento independentes criadas pelo <code>git branch</code> e as integre em uma ramificação única.</p>	1. <a href="#">Git merge</a>
	O que é <i>clone</i> ?	
17	<p>Aqui, vamos examinar o comando <code>git clone</code> em detalhes. O <code>git clone</code> é um utilitário de linha de comando que é usado para selecionar um repositório existente e criar um clone ou cópia do repositório de destino. Nesta página, vamos discutir opções de configuração estendidas e casos de uso comuns do <code>git clone</code>. Alguns dos pontos que vamos cobrir aqui são:</p> <ul style="list-style-type: none"> <li>• Clonar um repositório local ou remoto</li> <li>• Clonar um repositório vazio</li> <li>• Usar opções rasas para clonagem parcial dos repositórios</li> <li>• Sintaxe de URL do Git e protocolos suportados</li> </ul>	1. <a href="#">git clone</a>
	O que é o <i>GitHub</i> ?	
18	<p>O GitHub, tão famoso entre a comunidade de programadores de todo o mundo, é uma espécie de rede social voltada a profissionais de TI cuja tecnologia que o sustenta é o GIT.</p> <p>Em outras palavras, GitHub é uma plataforma totalmente online onde você pode criar repositórios e hospedar neles seus projetos, colaborar com softwares open source, seguir outros(as) programadores(as) e interagir com códigos de terceiros.</p> <p>O GitHub armazena todos estes dados em uma nuvem e você pode acessá-los de onde estiver: basta logar-se no site em qualquer navegador.</p>	1. <a href="#">O QUE É GIT: CONCEITOS, PRINCIPAIS COMANDOS E SUAS VANTAGENS?</a>
19	Qual a diferença entre o <i>git</i> e o <i>GitHub</i> ?	

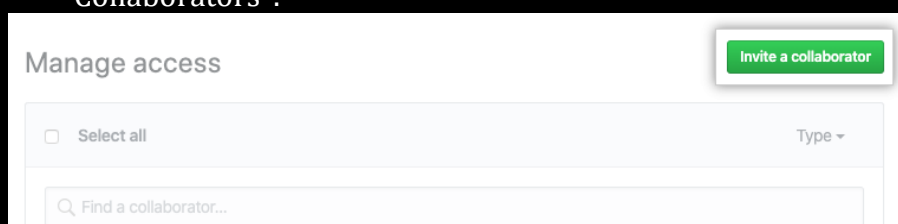
## Conceitos Básico

	<ul style="list-style-type: none"> <li>• git é um software VCS local que permite aos desenvolvedores salvar snapshots de seus projetos ao longo do tempo. Geralmente é melhor para uso individual.</li> <li>• GitHub é uma plataforma baseada na web que incorpora os recursos de controle de versões do gitHub para que possam ser usados colaborativamente. Também inclui recursos de gerenciamento de projetos e equipes, assim como oportunidades para networking e codificação social.</li> </ul>	1. <a href="#">Git vs Github: Qual é a Diferença e Como Começar com Ambos</a>
	Qual a diferença entre o repositório local e repositório remoto?	
20	<ul style="list-style-type: none"> <li>• O repositório local é utilizado quando o desenvolvedor está trabalhando sozinho, porém não precisa ou não quer compartilhar o projeto. Entretanto ele quer fazer um controle de versionamento do seu projeto. Quando criado esse repositório fica localmente na máquina onde o projeto está.</li> <li>• Já o repositório remoto é utilizado quando os desenvolvedores precisam compartilhar o seu trabalho com outros desenvolvedores do time. Na maior parte dos casos as empresas utilizam o repositório remoto, onde ficará todo o projeto. Quando criado é preciso clonar o repositório remoto para o repositório local, fazendo assim sua sincronização dos arquivos. O repositório remoto pode ficar em servidores, podendo ser internos ou de terceiros, que armazenam o código fonte do seu projeto.</li> </ul>	1. <a href="#">Trabalhando com Repositórios</a> 2. <a href="#">Como criar um repositório</a>
21	Como adicionar pessoas para contribuir no repositório?	

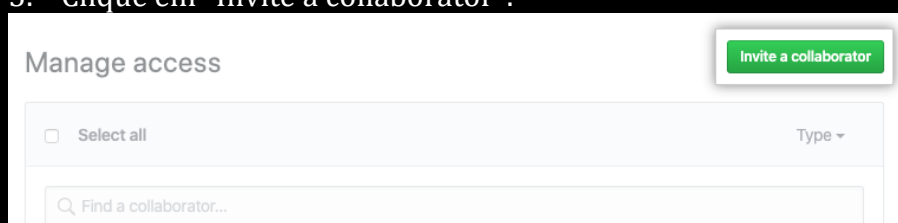
1. Peça o nome de usuário da pessoa que você está convidando como colaborador. Se eles ainda não tiverem um nome de usuário, podem se inscrever no GitHub Para mais informações, consulte "[Inscrição para uma nova conta GitHub](#)".
2. No GitHub.com, navegue até a página principal do repositório.
3. Abaixo do nome do repositório, clique em "Settings".



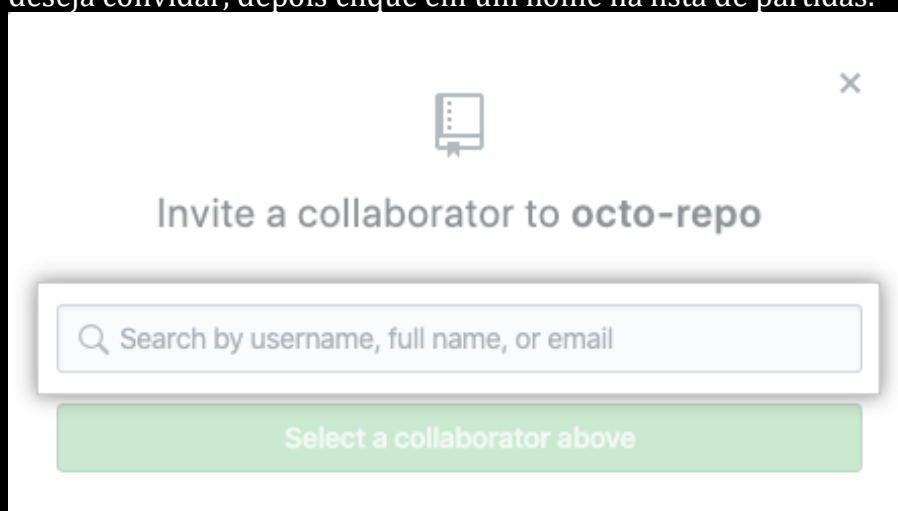
4. Na seção "Access" da barra lateral esquerda, clique em "Collaborators".



5. Clique em "Invite a collaborator".

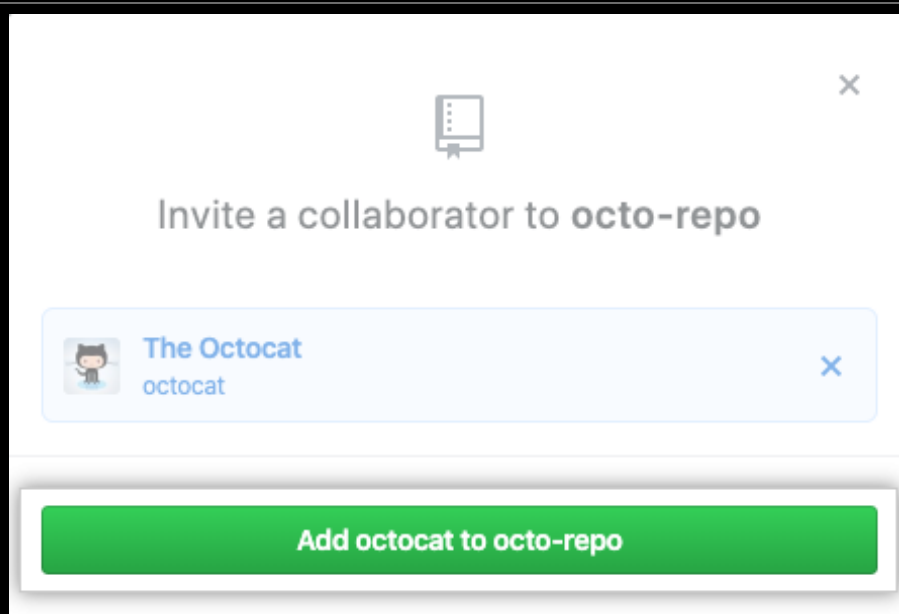


No campo de busca, comece a digitar o nome da pessoa que você deseja convidar, depois clique em um nome na lista de partidas.



6. Clique em "Add NOME DO USUÁRIO to NOME DO REPOSITÓRIO".

1. [Inviting collaborators to a personal repository](#)



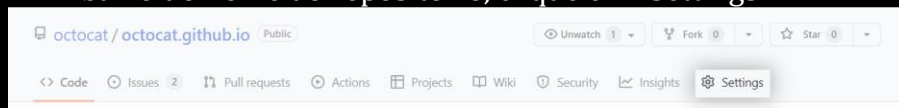
O usuário receberá um e-mail convidando-os para o repositório. Uma vez aceito seu convite, eles terão acesso ao seu repositório como colaboradores.

## Conceitos Básico

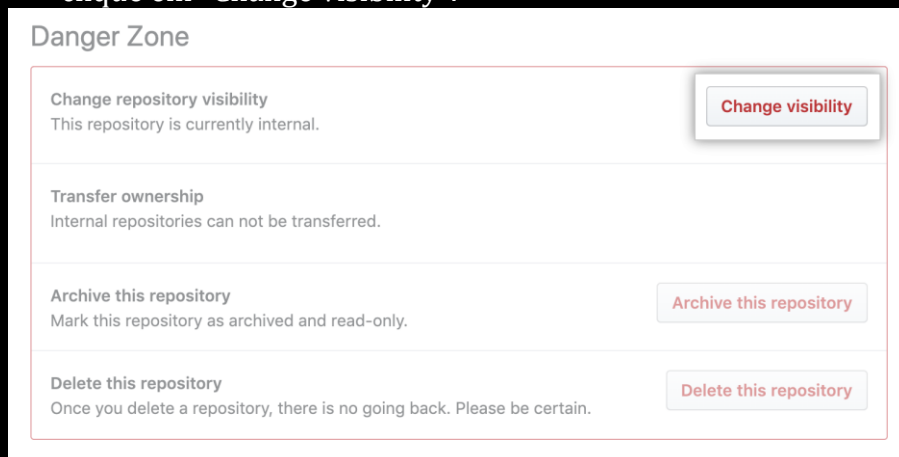
22	O que é um repositório público?	
	Os repositórios públicos no GitHub são usados frequentemente para compartilhar softwares de código aberto. Para que seu repositório seja realmente de código aberto, você precisará licenciá-lo para que outros tenham a liberdade de usar, alterar e distribuir o software.	1. <a href="#">Licenciando o repositório</a>
23	O que é um repositório privado?	
	Faz com que o repositório remoto fique privado. Se criar um repositório remoto do início as pessoas não terão visibilidade ao repositório criado, somente os colaboradores. Porém se já tiver um repositório público criado e quer torná-lo privado deverá mudar nas configurações do Github. Ao terminar de configurar o Github irá remover os <i>forks</i> e alguns recursos irão parar de funcionar como o Github Advanced Security features e o GitHub Archive Program.	1. <a href="#">Setting repository visibility</a>
24	Como mudar o repositório do público para o privado?	



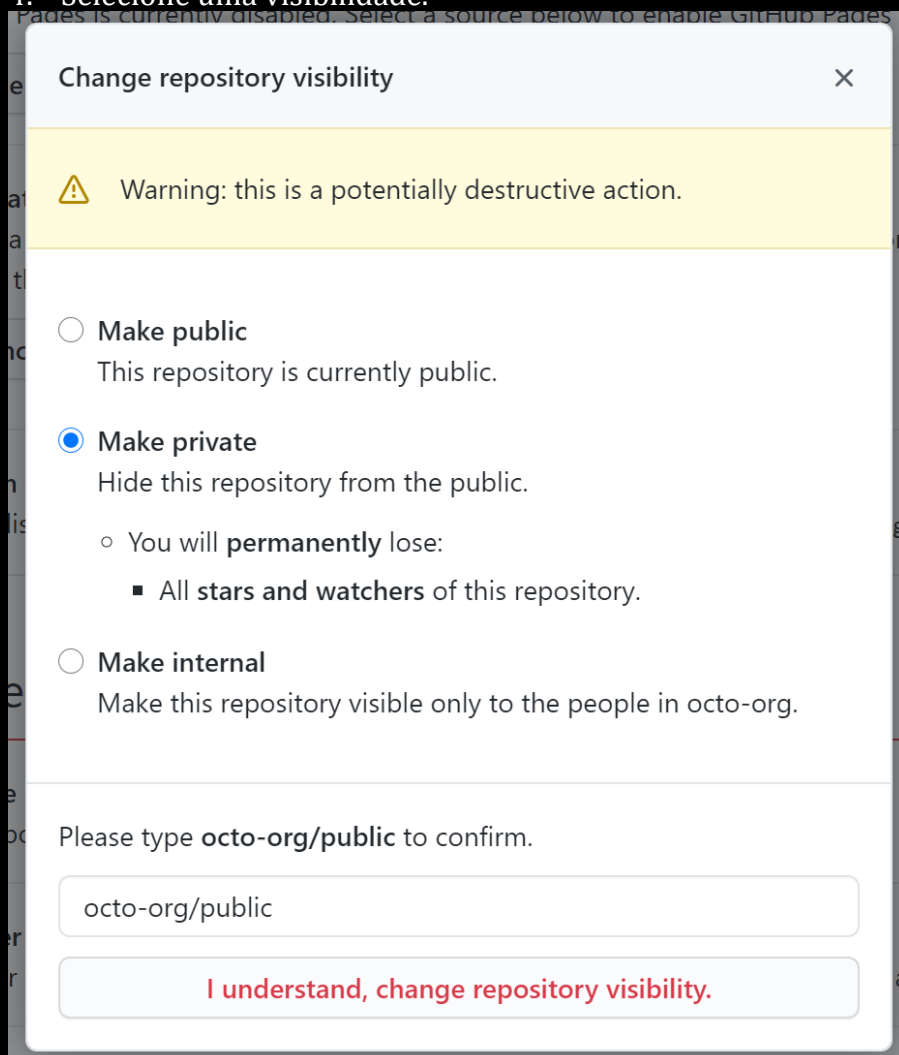
1. No GitHub.com, navegue até a página principal do repositório.
2. Abaixo do nome do repositório, clique em "Settings".



3. Em "Danger Zone", à direita de "Change repository visibility", clique em "Change visibility".



4. Selecione uma visibilidade.



1. [Setting repository visibility](#)

5. Para verificar se você está mudando a visibilidade do repositório correto, digite o nome do repositório do qual você deseja mudar a visibilidade.
6. Clique em "I understand, change repository visibility".

Please type `octo-org/public` to confirm.

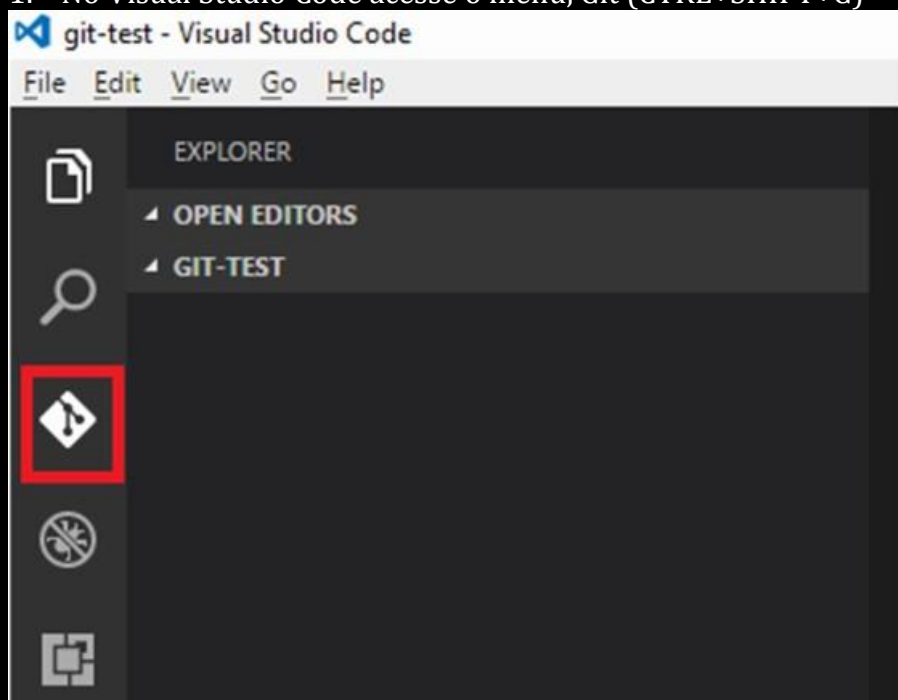
`octo-org/public`

**I understand, change repository visibility.**

## Conceitos Básico

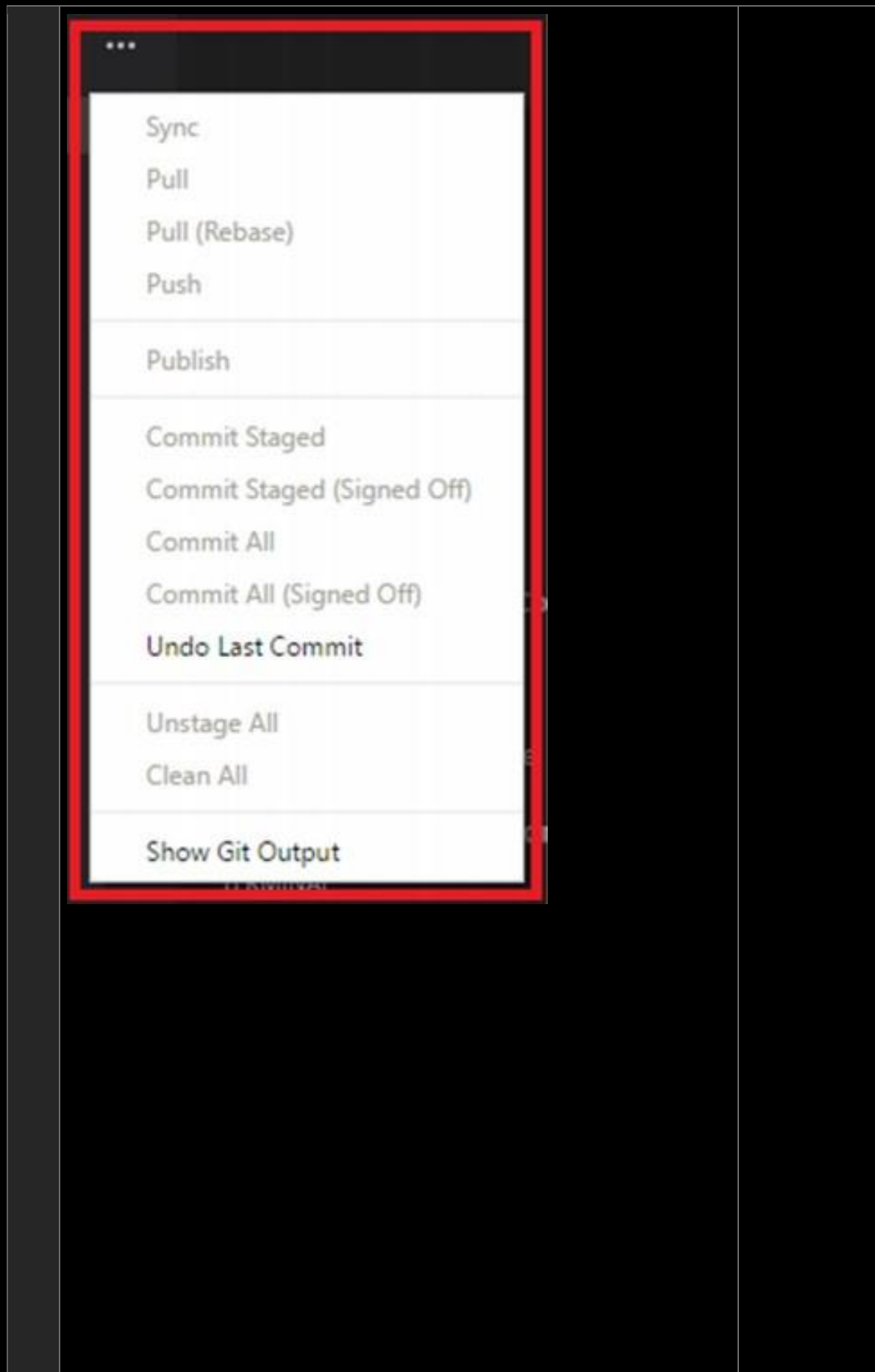
25	Como conectar o git no VsCode?
----	--------------------------------

1. No Visual Studio Code acesse o menu, Git (CTRL+SHIFT+G)



2. Clique no botão Inicialize Git Repositore
3. Pressione CTRL + para abrir o terminal do Git
4. Verifique a configuração do git:
5. `git config -list < enter >`
6. Certifique que o `user.name`, `user.email` e `core.longpaths` estejam configurados e qualquer dúvida consulte a seção “Configurar e instalação do GIT”.
7. Para conectar o projeto ao repositório remoto digite:
8. `git remote add origin << URL do repositório remoto >> < enter >`
9. Utilizar as funções do menu Git para controlar o repositório

1. [Manual de instalação do GIT, plug-ins Eclipse e Visual Studio Code e configuração de soluções](#)



## Conceitos Básico

26	Como conectar o git no Repl.it?
----	---------------------------------

Criar uma nova reposição, ou abrir uma já existente. Em seguida, na barra lateral, clique no ícone de idiota (parece um galho) e você será saudado com uma calorosa recepção.

1. [GIT on repl.it](https://repl.it/GIT)

## Create a repository

Track and roll back changes to your repl easily adding a git repository to this repl.

What is git?


Create a git repository

Clique em "Create a git repository", e toneladas de botões e palavras irão aparecer. No início é muito para absorver, mas ao final deste tutorial fará sentido. O Repl.it se integra com o GitHub para que você possa hospedar seu código online. Se você não tiver uma conta, você pode fazer uma aqui. Uma vez que você tenha feito isso, precisamos conectar seu projeto repl.it a um repositório. Os repositórios são simplesmente pastas que contêm seu código, assim como seu próprio computador faria. Você pode decorar seu repositório com 'crachás' e links, mas nós guardaremos isso para outro tutorial. Clique no botão "Connect to Github" e você verá este popup.

×


## Create a new GitHub repository


You can always edit this information on github

 Zwork101 / tutorial-save

Add repo description (optional)

### GitHub privacy

☒  **Public**  
Anyone can see this repo, you choose who can commit

☐  **Private**  
you choose who can see and commit to this repo

Create GitHub repository

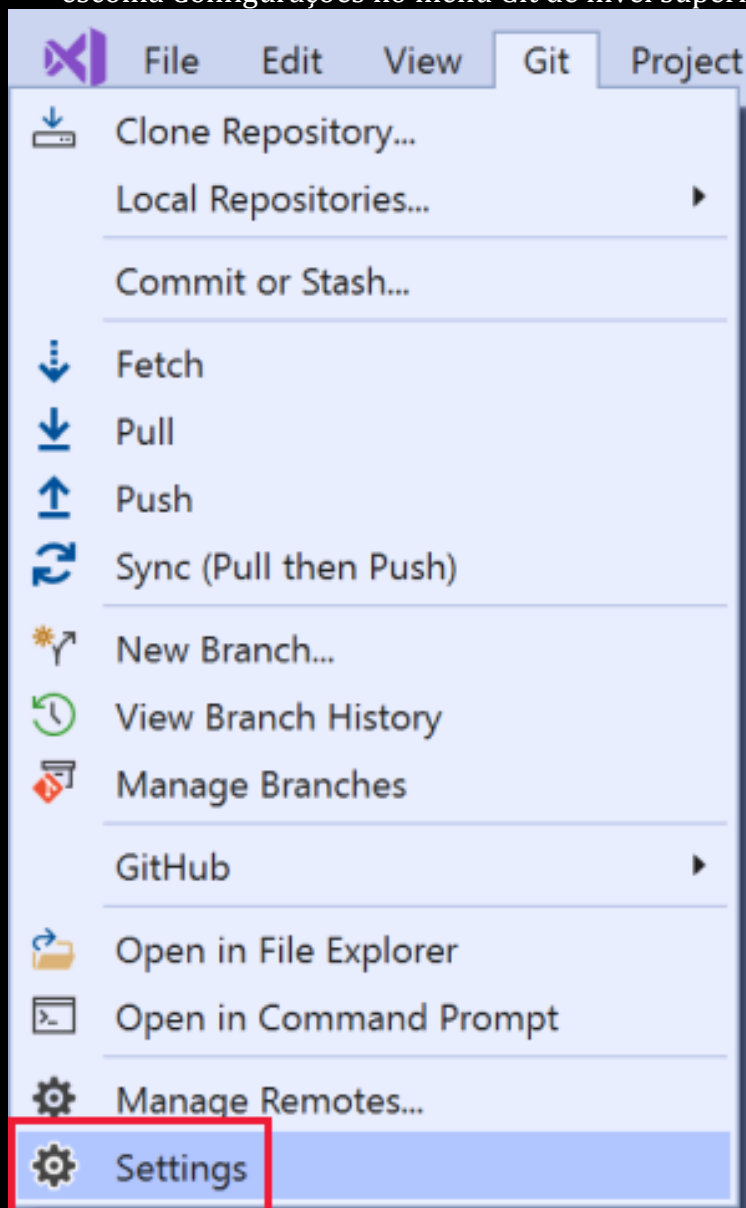
Repl.it preencherá o nome github com o nome da réplica, porém você pode mudar isso se assim o desejar. Em seguida, forneça uma descrição para a resposta. Faça-o curto e doce, pois você terá a chance de acrescentar uma descrição mais longa mais tarde. Agora você tem a opção de tornar sua repo pública, ou privada. Se seu projeto for público, qualquer um pode ver seu trabalho legal, e se realmente gostar dele, eles adicionam ou mudam algum de seu código para torná-lo ainda melhor. É claro que primeiro você precisará aprovar a mudança antes que ela seja aplicada, ou fazer mudanças na mudança deles. Mas ser público não é para todos, e você pode querer manter seu código para si mesmo. A escolha também é sua, mas não afetará a forma como ela interage com seu reporte.



## Conceitos Básico

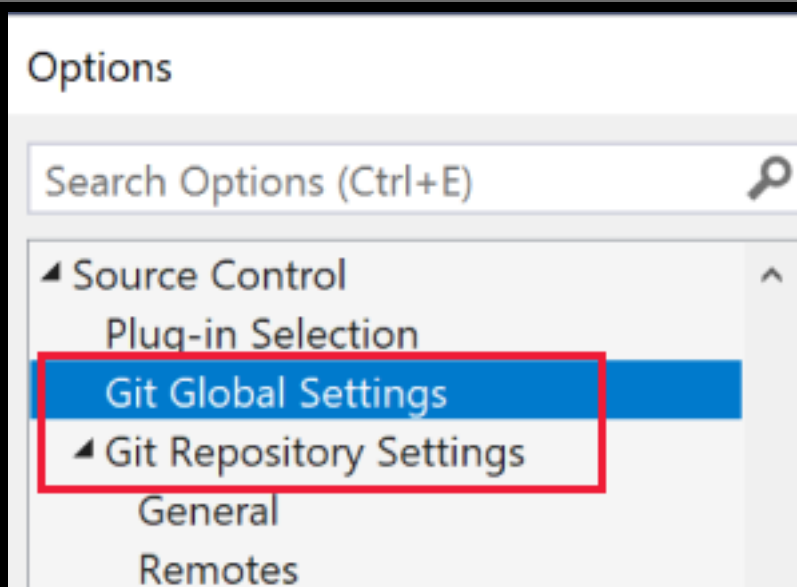
27	Como conectar o git no Visual Studio?
----	---------------------------------------

1. Para definir as configurações do Git no Visual Studio, escolha Configurações no menu Git de nível superior.

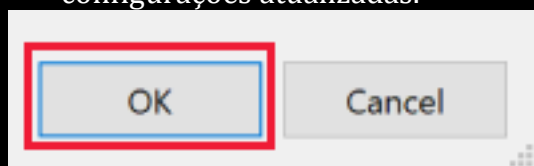


2. Escolha Configurações Globais do Git ou Configurações do Repositório Git para exibir e definir configurações de nível global ou de repositório.

1. [Configurações e preferências do Git no Visual Studio](#)



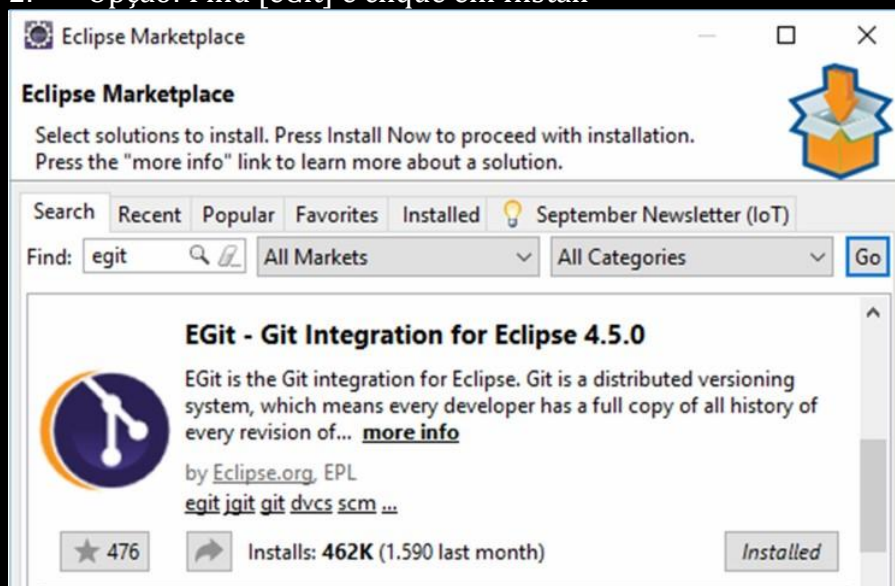
3. Você pode definir várias configurações comuns do Git, conforme descrito nas seções a seguir deste artigo. Depois de definir as configurações desejadas, selecione OK para salvar as configurações atualizadas.



## Conceitos Básico

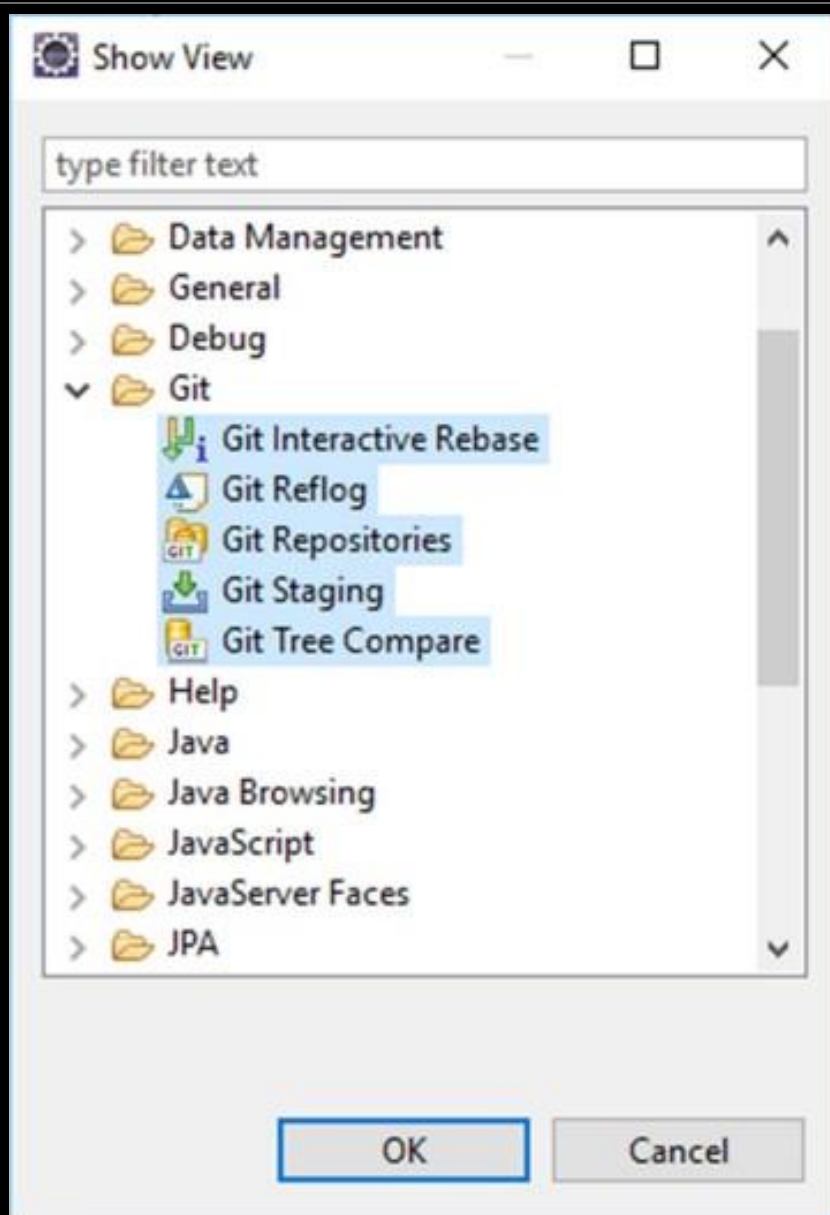
28	Como conectar o git no Eclipse?
----	---------------------------------

1. No Eclipse acesse o menu, Help, Eclipse Marketplace
2. Opção: Find [eGit] e clique em Install

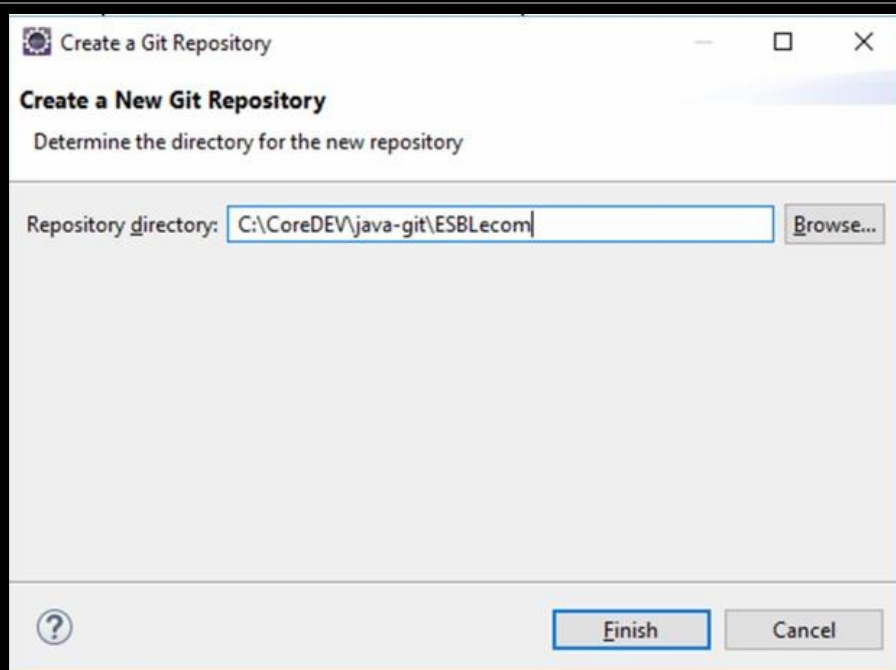


3. Abrir as guias do componente eGit instalado
4. Abrir as guias do componente eGit instalado
  1. No Eclipse acesse o menu, Windows, Show View, Other

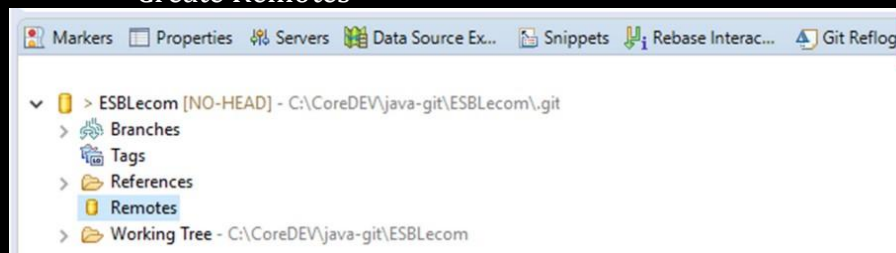
1. [Manual de instalação do GIT, plug-ins Eclipse e Visual Studio Code e configuração de soluções](#)



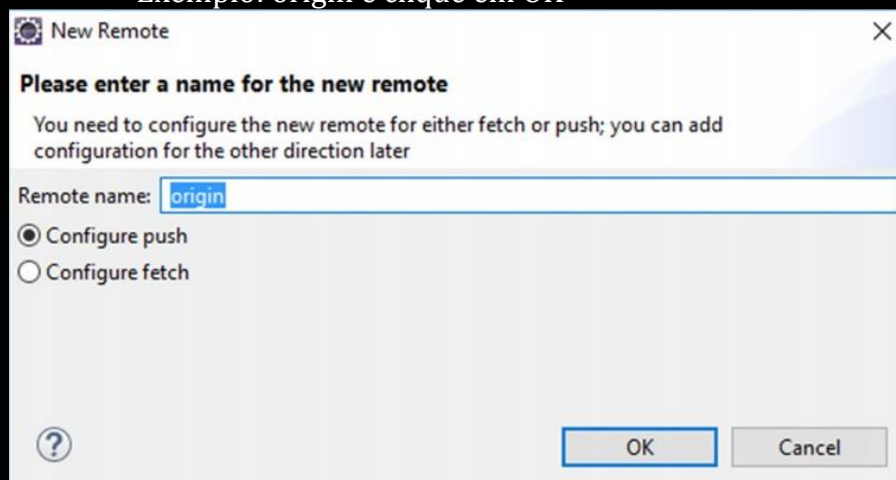
5. Configurar o repositório git local no Eclipse
  1. No Eclipse clique com o botão direito no projeto, menu Team, Share Project
  2. Clique em Create e selecione a pasta para gravação do seu GIT local. Recomendo criar a pasta com o mesmo nome do repositório remoto.



3. Clique em Finish e Finish para confirmar a criação do repositório local
6. Configurar o repositório git remote no Eclipse
  1. Na guia Git repositories expanda as configurações do repositório e clique com o botão direito em Remotes, Create Remotes



2. Informe o alias" do repositório remoto. Exemplo: origin e clique em OK



3. Clique em Change e informe a URL do repositório remoto
4. Informe seu User e Password de acesso ao repositório remoto

**Select a URI**

**Destination Git Repository**  
Enter the location of the destination repository.

**Location**

URI:

Host:

Repository path:

**Connection**

Protocol:

Port:

**Authentication**

User:

Password:

☒ Store in Secure Store

5. Clique em Finish e Save in Push
7. Para enviar uma atualização para o repositório remoto
  1. Na guia Git Staging selecione todos os arquivos da lista Unstaged Changes e movimente para a lista Staged Changes
  2. Faça o comentário da atualização em Commit Message
  3. Clique no botão Commit and Push

Markers Properties Servers Data Source Ex... Snippets Rebase Interac... Git Reflog Git Repositories **Git Staging** Git Tree Comp...

> ESBLecom [NO-HEAD]

Unstaged Changes (101)

- TemplateBusiness.java - Asbpm\_PlataformaLecom\_BUS/src/br/com/asbpm/lecc
- Usuario.java - Asbpm\_PlataformaLecom\_BUS/src/br/com/asbpm/lecom/model
- web.xml - Asbpm\_PlataformaLecom\_BUS/WebContent/WEB-INF

Staged Changes (2)

- .classpath - Asbpm\_PlataformaLecom\_BUS
- .dbeaver-data-sources.xml - Asbpm\_PlataformaLecom\_BUS

Commit Message

1.0.0: Atualização da serviço de consulta de processos

Author: "Ely <ely.goncalves@asbpm.com.br>"

Committer: "Ely <ely.goncalves@asbpm.com.br>"

8. Para baixar uma atualização do repositório remoto.
  1. No Eclipse clique com o botão direito no projeto, menu Team, Remote, Fetch, From...

**Fetch from Another Repository**

**Source Git Repository**  
Enter the location of the source repository.

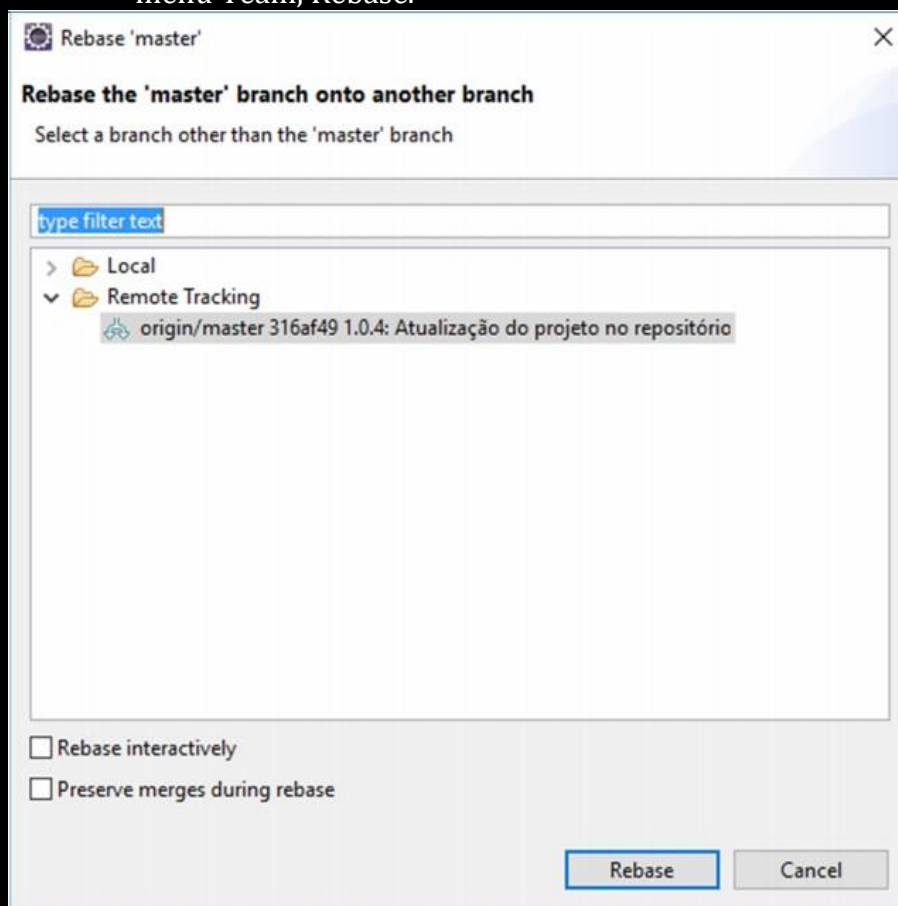
☒ Configured remote repository:

origin:

2. Clique em Next

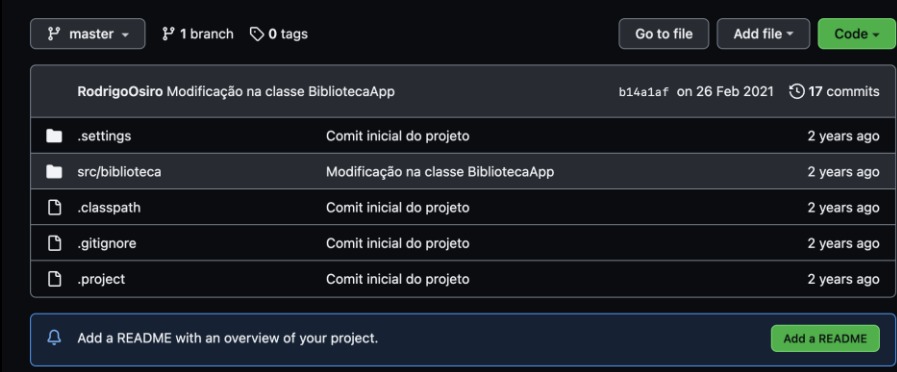
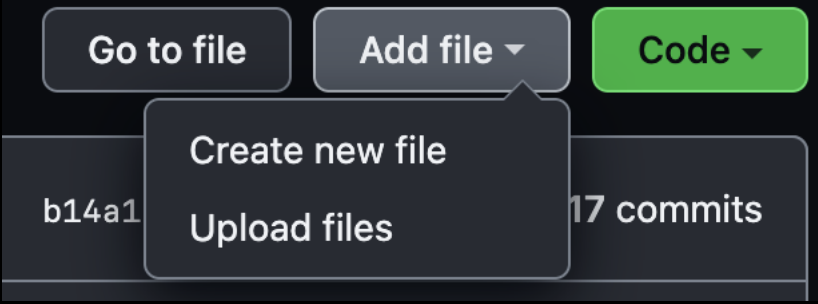

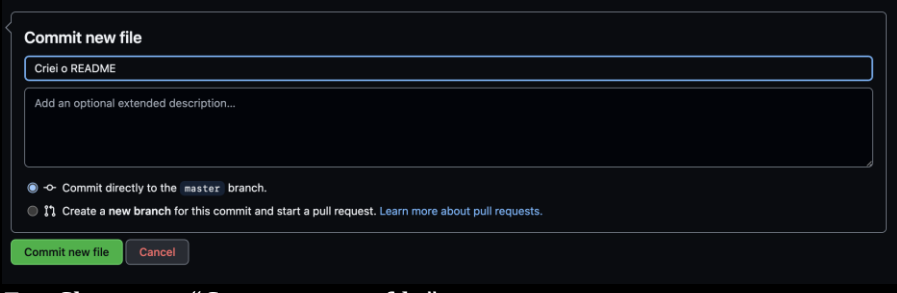


3. Em Source ref selecione máster (caso seja branch principal) e clique em Add Spec
4. Clique em Finish para finalizar a atualização
9. Para sincronizar o repositório local com a última atualização do repositório remoto.
  1. No Eclipse clique com o botão direito no projeto, menu Team, Rebase.



2. Clique em Rebase para confirmar a atualização do repositório local.

Conceitos Básico

	<div>Como adicionar o <i>README</i>?</div> <div><div><div>1. Selecionar o repositório desejado</div><div>2. Clicar no botão “Add file”</div></div><div><div>3. Clicar no botão “Create new file”</div><div><div>4. Adicionar o nome “README.md” para que o GitHub possa identificar.</div><div><div>5. Escrever em baixo, podendo ser em Markdown, sobre o que o projeto/repositório faz</div><div>6. Criar o commit do README</div><div><div>7. Clicar em “Commit new file”</div></div></div></div><div><div>1. <a href="#">About READMEs</a></div></div></div></div>
29	
30	<div>Adicione suas próprias perguntas abaixo...</div> <div></div>

## 2. NÍVEL INTERMEDIÁRIO

O nível intermediário tem como objetivo de abordar conceitos mais complexos como merge, rebase, gerenciar conflitos de merge, trabalhar em várias branches, git ignore.

### Conceitos Intermediário

N	Questões	Recursos
1	O que é <i>checkout</i> ?	
2	O que é <i>stash</i> ?	
3	O que é conflito no git?	
4	O que é <i>Pull Request</i> ?	
5	O que é <i>add</i> ?	
6	O que é <i>.gitignore</i> ?	
7	Como resolver conflito no git?	
8	Quais são as boas práticas para evitar conflitos no git?	
9	O que são <i>tags</i> ?	
10	O que é <i>rebase</i> ?	
11	O que é <i>git flow</i> ?	
12	Como bloquear a <i>develop</i> para não receber <i>push</i> ?	
13	Como bloquear a <i>main/master</i> para não receber <i>push</i> ?	

## Conceitos Intermediário

14	Como utilizar o git do próprio compilador?	
15	Como gerar <i>Personal Access tokens</i> no GitHub?	
16	O que é um snapshot?	
17	O que é um FastForward merge?	
18	O que é um 3-way merge?	
19	Adicione suas próprias perguntas abaixo...	

### 3. NÍVEL AVANÇADO

O nível avançado tem como objetivo de aprofundar mais sobre os conceitos já estabelecidos como reverter mudanças de pull request, issues, releases, refazer commits, pipeline, CI, CD, GitHub Actions, GitHub Projects, Github CLI, git CLI e etc.

#### Conceitos Avançado

N	Questões	Recursos
1	Como reverter um <i>pull request</i> ?	
2	Como refazer um <i>commit</i> ?	
3	O que são <i>issues</i> no Github?	
4	Como criar <i>issues</i> no Github?	
5	O que são <i>releases</i> no Github?	
6	Como criar <i>releases</i> no Github?	
7	O que é uma <i>pipeline</i> ?	
8	Como fazer uma <i>pipeline</i> ?	
9	O que é <i>Continuous Integration</i> ?	
10	O que é <i>Continuous Delivery</i> ?	
11	O que é GitHub Actions?	
12	Como utilizar o Github Actions para fazer uma pipeline?	

## Conceitos Avançado

13	O que é <i>Github Projects</i> ?	
14	Como criar um <i>Projects</i> para o repositório?	
15	Como adicionar a equipe no <i>Projects</i> ?	
16	O que é CLI?	
17	O que é GitHub CLI?	
18	Como instalar o Github CLI?	
19	Qual é o site da documentação do Github CLI?	
20	Como integrar a conta do Github no Github CLI?	
21	Como fazer comandos no git CLI?	
22	Qual é o site que possui comandos do git CLI?	
23	Como acessar um projeto através do CLI?	
	Adicione suas próprias perguntas abaixo...	

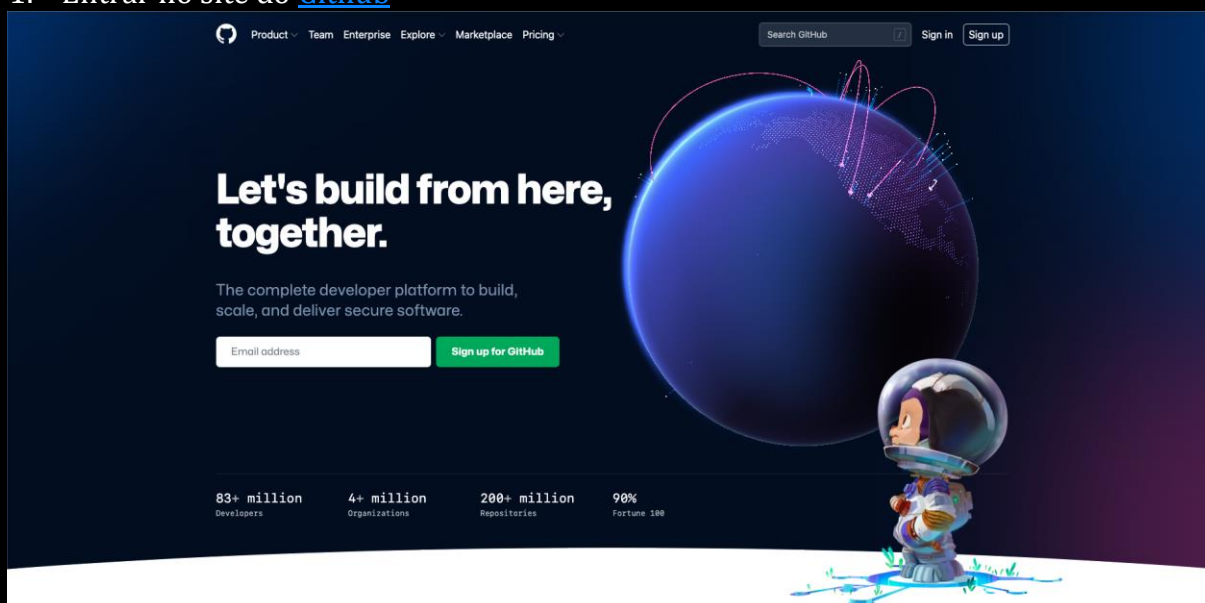
#### 4. OUTROS RECURSOS

##### Recursos

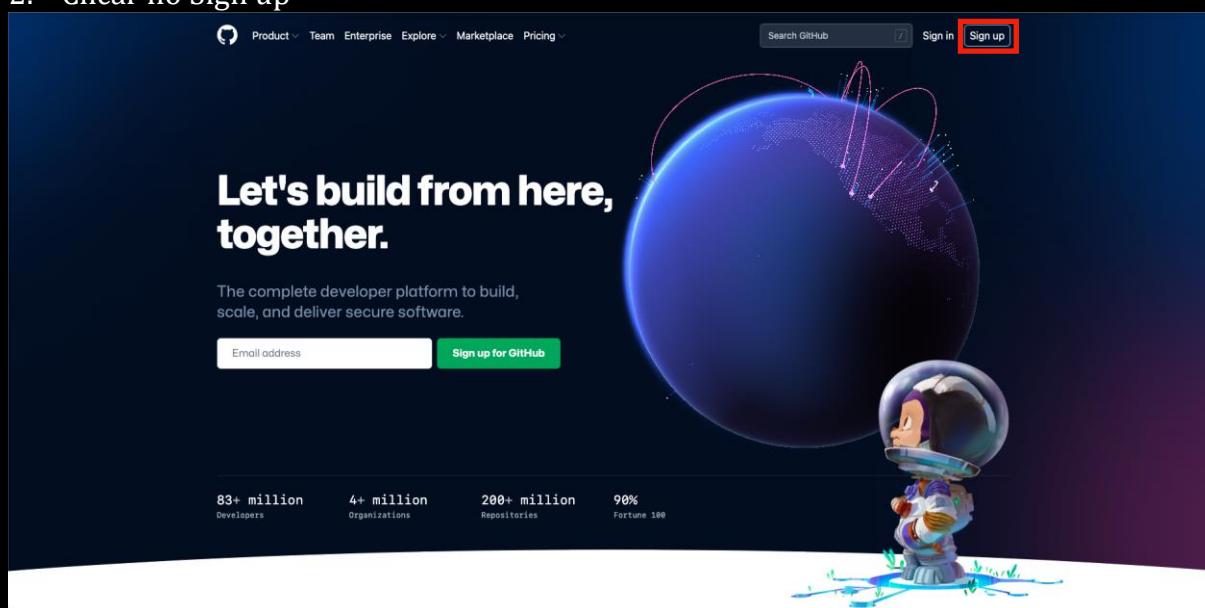
<a href="#">Git SCM</a>
<a href="#">Learn Git Branching!</a>
<a href="#">Documentação do GitHub</a>
<a href="#">GitHub Actions</a>
<a href="#">GitHub Issues</a>
<a href="#">Fluxo de trabalho de Gitflow</a>
<i>Adicione seus próprios recursos...</i>

## 5. COMO CRIAR UMA CONTA NO GITHUB?

### 1. Entrar no site do [Github](https://github.com)

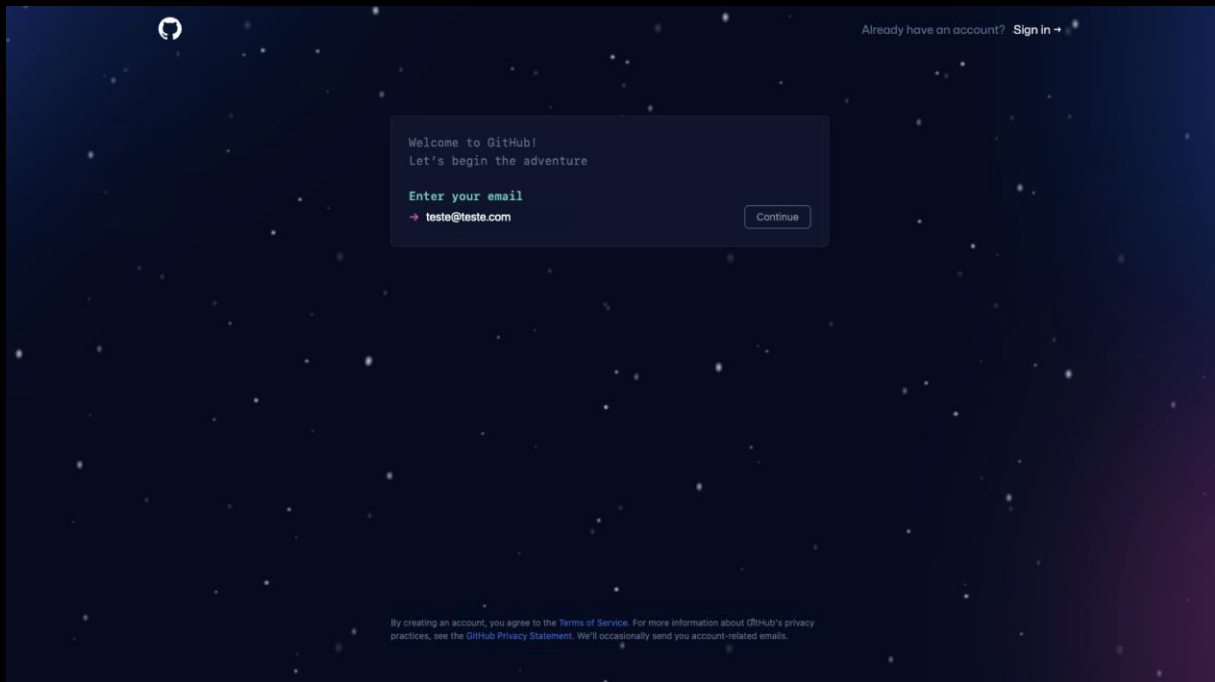


### 2. Clicar no Sign up



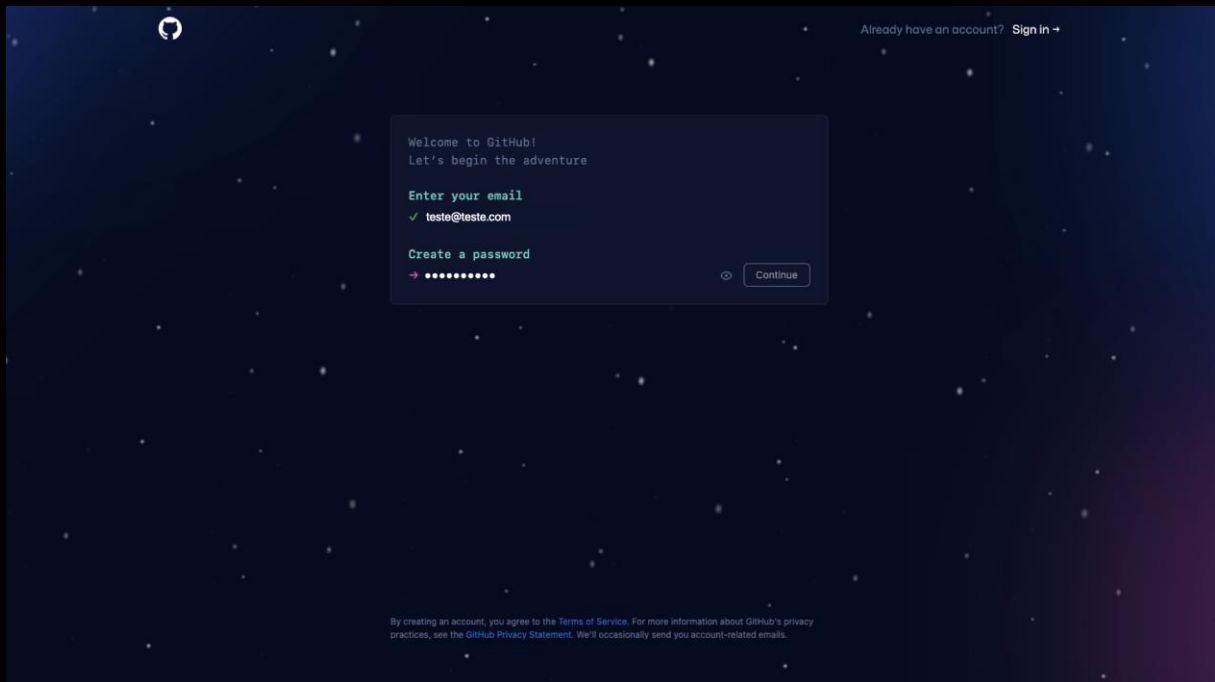


### 3. Inserir o e-mail



The screenshot shows the GitHub sign-up interface. At the top left is the GitHub logo. At the top right, it says "Already have an account? [Sign in](#)". The main content area has a dark blue background with a starry pattern. A white card in the center contains the following text: "Welcome to GitHub! Let's begin the adventure". Below this, it says "Enter your email" followed by a red arrow icon and the text "teste@teste.com". To the right of the email input is a "Continue" button. At the bottom of the card, there is a small line of text: "By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails."

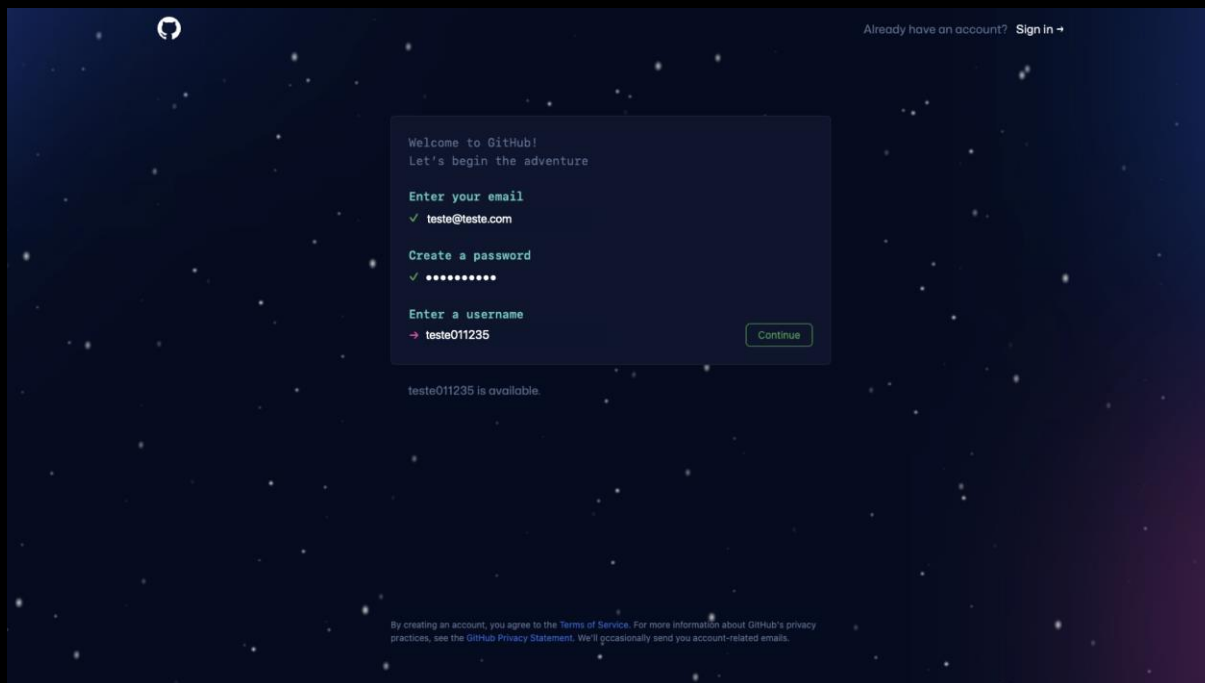
### 4. Inserir a senha



The screenshot shows the GitHub sign-up interface at the password step. The layout is identical to the previous step, but the "Enter your email" section now shows a green checkmark and the text "teste@teste.com". Below this, it says "Create a password" followed by a red arrow icon and a series of dots representing the password. To the right of the password input is a "Continue" button. The same footer text is present at the bottom of the card.

## 5. Inserir o nome do usuário

OBS: esse nome será mostrado no perfil do Github. Escolha sabiamente, pois a partir do nome de usuário que empresas, outros usuários podem pesquisar o seu perfil.



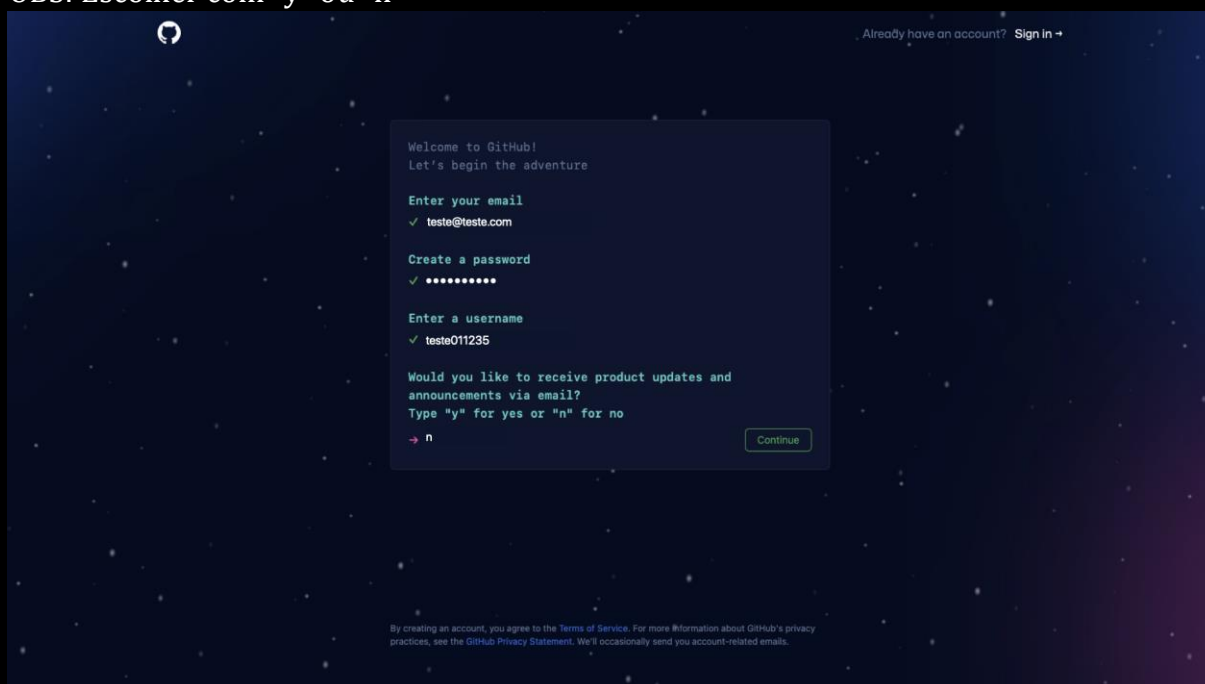
The screenshot shows the GitHub sign-up process on a dark blue background with a starry pattern. At the top right, there is a link: "Already have an account? Sign in →". The main form is a light blue box with the following content:

- Welcome to GitHub!  
Let's begin the adventure
- Enter your email  
✓ teste@teste.com
- Create a password  
✓ ●●●●●●●●
- Enter a username  
→ teste011235

A "Continue" button is located to the right of the username input. Below the form, a message states: "teste011235 is available." At the bottom of the page, there is a small disclaimer: "By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails."

## 6. Escolher se quer receber atualizações e anúncios do Github por email.

OBS: Escolher com "y" ou "n"

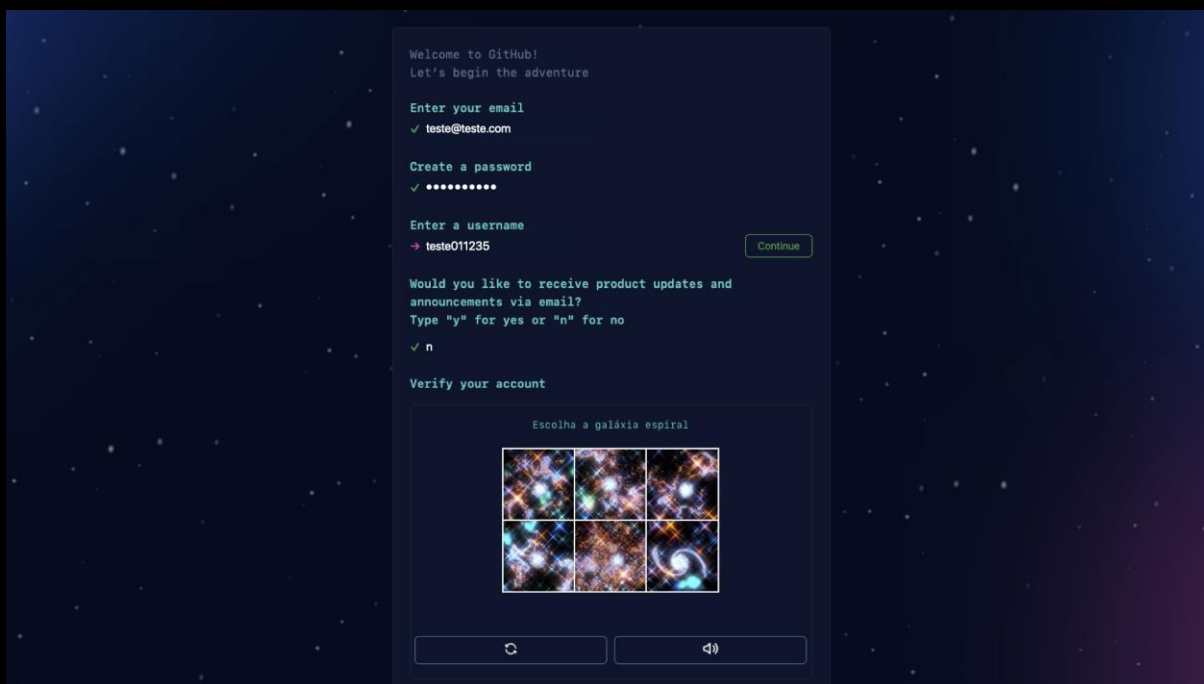


This screenshot shows the same GitHub sign-up form as the previous one, but at a later stage. The "Enter a username" step is now completed with "teste011235". The new question is:

Would you like to receive product updates and announcements via email?  
Type "y" for yes or "n" for no  
→ n

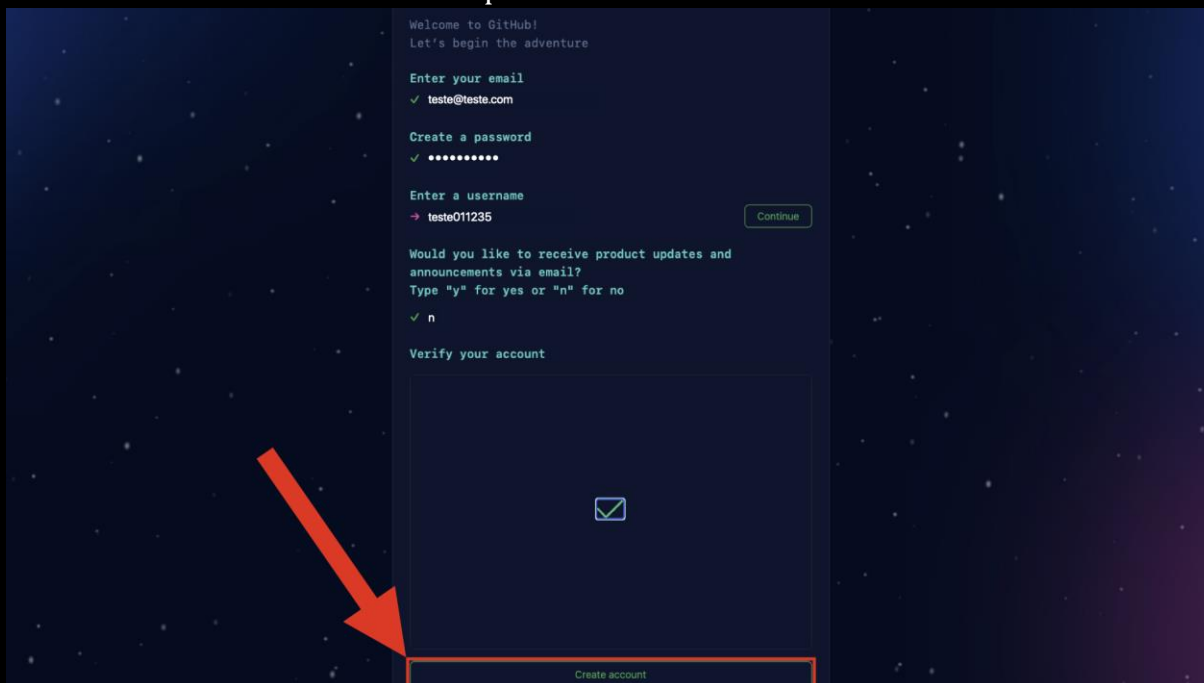
The "Continue" button remains to the right. The disclaimer at the bottom is identical to the previous screenshot.

7. Fazer a verificação se não é um bot criando uma conta.

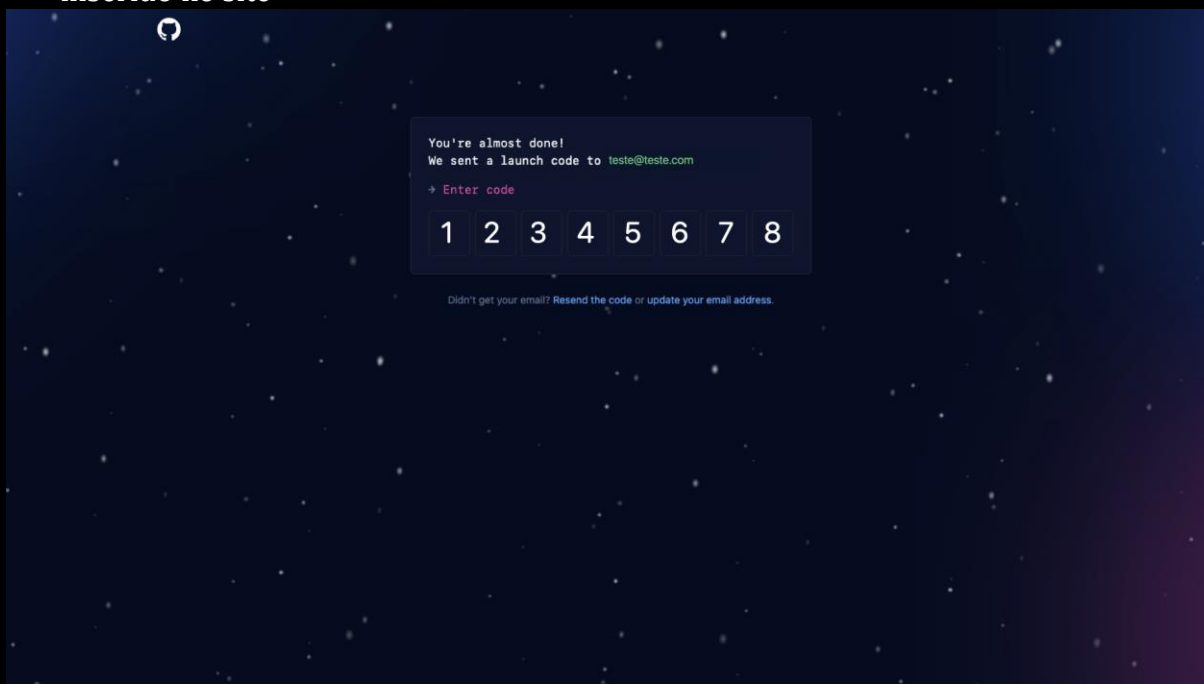


OBS: A verificação da conta pode ser de diferentes formas.

8. Clicar no botão de criar conta após verificar se não é um robô.



9. No email que foi cadastrado será enviado um código de verificação que deverá ser inserido no site



10. Escolher como deve ser configurado a conta criada.

OBS: Você não precisa escolher a configuração da conta agora. Você pode pular essa etapa. Não é necessário escolher a configuração da conta.

