Usando o SSH para conectar-se ao GitHub

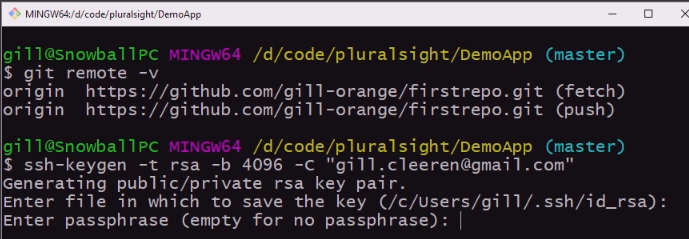
Dependendo do sistema operacional que você está usando, pode ser que você seja solicitado várias vezes por seu nome de usuário e senha, quando você está enviando alterações no GitHub e, com o tempo, isso se torna irritante. Usando o SSH (Secure SHell), criaremos uma chave SSH, que pode ser vinculada à nossa conta do GitHub. Quando você se conectar da sua máquina local ao GitHub, você não será solicitado a inserir suas credenciais sempre. Se necessário, você também pode combinar isso com uma abordagem de logon único, se tiver sido configurada para o seu acesso ao GitHub.

Digitando git remote –v, você verá que a origem está configurada usando HTTPS. Usando o SSH, vamos configurar uma chave SSH, adicioná-la ao agente e, em seguida, configurar essa chave em nossa conta do GitHub.

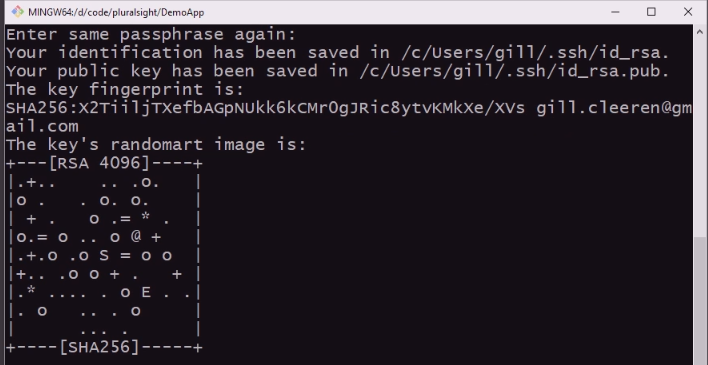
Usaremos o comando ssh-keygen para gerar a chave SSH.

**ssh-keygen -t rsa -b 4096 -C "shyoutarou@gmail.com"**

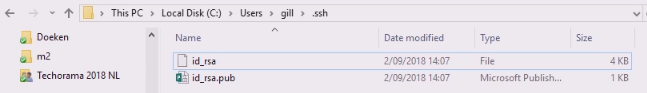
Depois, temos que responder aonde devemos armazenar a chave. Bem, vamos manter esse caminho em mente. A senha, basicamente é um nível de segurança extra. Por enquanto, vou deixá-lo vazio, mas, para ser 100% seguro, você não deve somente digitar uma palavra senha, mas uma frase segura.



Então a chave foi gerada:



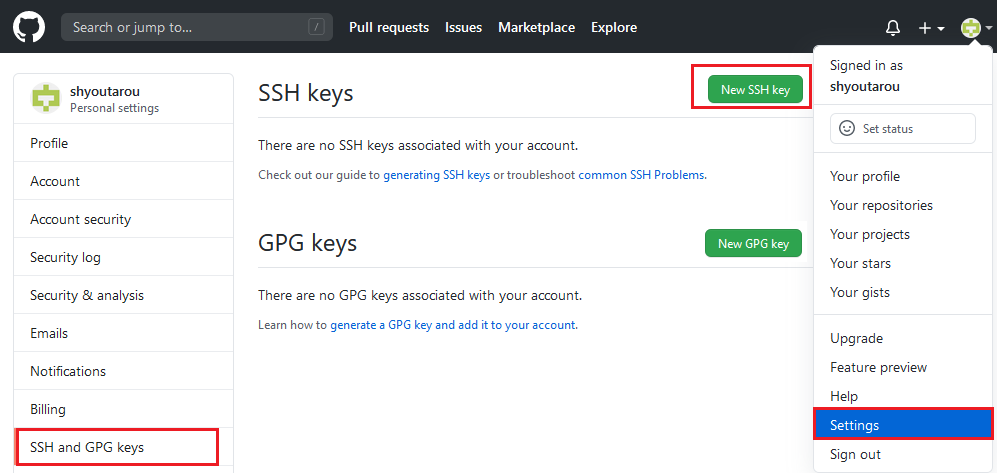
N o diretório indicado, veremos que dois arquivos foram gerados, uma parte pública e o arquivo-chave real.



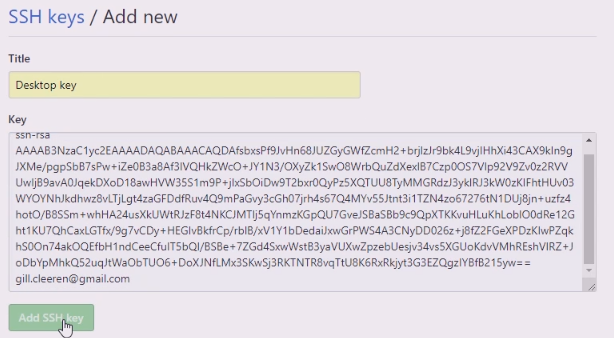
Vamos abrir a parte pública na minha com o bloco de notas. Isso precisa ser conhecido dentro do GitHub. Então, o que vou fazer é copiar a chave.



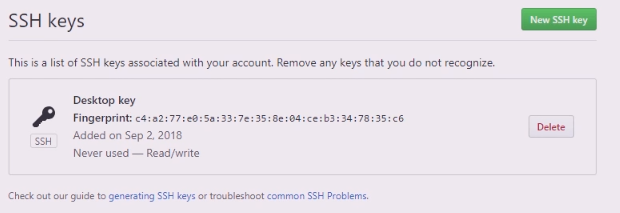
Agora voltarei ao meu perfil da conta do GitHub. Em seguida, clique em Configurações. E vamos aqui para SSH.



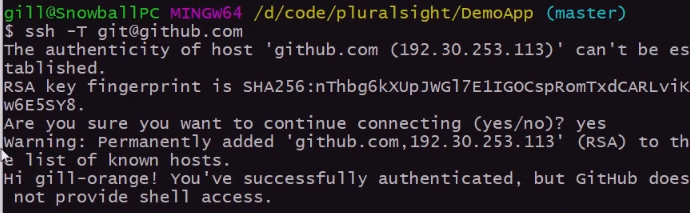
Aqui eu posso inserir uma nova chave SSH. Vamos chamá-lo de chave da área de trabalho e colar em nossa chave.



Então agora a chave foi adicionada.



Com a chave adicionada, ainda precisamos testar se tudo está funcionando corretamente. Então, voltemos ao prompt de comando, e o que vou fazer agora é chamar o comando SSH, -t, e depois preciso especificar o parâmetro git @ github. com. Em seguida, recebo uma pergunta para verificar se queremos ou não continuar conectando; digitaremos sim.



Neste ponto, recebemos uma mensagem autenticada com sucesso. A partir de agora, podemos nos comunicar com o GitHub usando SSH.

Arquivos especiais no GitHub O arquivo mais conhecido que podemos ou, devo dizer, deve adicionar ao nosso repositório é o arquivo README. No arquivo README, você normalmente coloca informações sobre um projeto, como o que ele faz, como isso pode ajudar no projeto, quem o possui, quem contribui com ele e assim por diante. É um arquivo conhecido pelo GitHub quando você o coloca no local correto. Isso inclui a raiz do repositório, a pasta github ou pasta de documentos. O que quero dizer com arquivo conhecido é que, quando disponível, o GitHub executará uma certa ação com ele. E para o arquivo README, isso significa que esse arquivo será mostrado automaticamente na página de entrada do repositório. Quando você faz alterações, elas serão refletidas ali na página de destino. O arquivo README normalmente é gravado no formato Markdown, também conhecido como formato md.

ormato Markdown (MD)

Usando o Markdown, posso criar um arquivo README.md bem formatado. O formato Markdown, ou formato MD, é um formato compreendido pelo GitHub para oferecer opções de formatação fáceis. Por exemplo, posso escrever aqui um sinal de hash e depois dizer que este é um H1. Se eu adicionar outro sinal de hash aqui, ele será renderizado como um H2.

Nesta captura de tela aqui, você pode ver um arquivo README renderizado, que está neste caso na raiz do repositório. O GitHub percebeu isso e o executa automaticamente na página de destino do repositório. Não apenas o arquivo README é um arquivo especial conhecido pelo GitHub. Outros existem também. O arquivo de licença denominado licença ou licença. O md contém sua licença de código-fonte aberto, permitindo que colaboradores aspirantes saibam qual é a licença aplicada ao projeto. Em seguida, temos o arquivo do colaborador ou colaborador, que é usado para conter uma lista de pessoas ou organizações que contribuíram para o projeto. O arquivo de registro de alterações contém uma lista de todas as principais alterações entre diferentes versões do projeto. Outro arquivo é o arquivo MD de suporte ou suporte, que é um arquivo para informar as pessoas sobre possíveis maneiras de obter ajuda com o projeto. Finalmente, e muito importante, é o arquivo CODE\_OF\_CONDUCT. Este arquivo contém diretrizes para as pessoas sobre as regras que elas devem obedecer ao interagir com o projeto. Observe que nem todos esses arquivos são necessários para o seu projeto. Ao navegar pelos projetos no GitHub, você notará que esses arquivos README normalmente têm um estilo de formatação muito comum. Isso ocorre porque eles são escritos em Markdown, uma linguagem de marcação muito leve que usa texto sem formatação. Pode ser convertido em HTML facilmente. Aqui você vê alguns exemplos para o Markdown. Por exemplo, adicionando um único símbolo de hash, estamos indicando que essa linha deve ser renderizada como um cabeçalho H1. Ao adicionar um segundo, essa linha será renderizada como um H2. Adicionar um asterisco na frente dos itens os renderiza como uma lista. Agora o Markdown é bem simples, mas mostrar a sintaxe inteira não é o objetivo deste curso. Você pode encontrar a sintaxe completa no seguinte URL. Demonstração: Criando arquivos no GitHub Tudo bem, vamos voltar ao GitHub e adicionar alguns desses arquivos especiais. Começaremos com um arquivo README e também adicionaremos algumas diretrizes de contribuição. Tudo bem, arquivos especiais. Portanto, o GitHub tem sugerido que deveríamos adicionar esse arquivo README. Mas, em vez de fazê-lo usando este botão que vemos aqui, eu o farei manualmente. Portanto, na página de entrada do repositório, irei em Criar novo arquivo e adicionarei um novo nome de arquivo. Vamos chamar isso de README. arquivo md. Agora, o arquivo README ou o README. md, a extensão do arquivo é basicamente opt internacional, é um arquivo muito importante, como você já deve ter adivinhado. Este arquivo será renderizado pelo GitHub na página inicial do repositório. Se você incluir o arquivo com esse nome, o GitHub fará isso automaticamente. Mas aqui você pode apenas adicionar texto sem formatação, mas existe uma maneira melhor de fazer isso, e que está usando algo chamado Markdown, ou formato MD. O formato Markdown é um formato compreendido pelo GitHub para oferecer opções de formatação fáceis. Por exemplo, posso escrever aqui um sinal de hash e depois dizer que este é um H1. Se você der uma olhada na visualização, de fato, adicionando esse sinal de hash na frente dessa frase, ela será renderizada como um H1. Se eu adicionar outro sinal de hash aqui, ele será renderizado como um H2. Ainda diz que é um H1, mas acho que você entendeu a mensagem. Podemos fazer, por exemplo, quatro hashes, e isso é completo. Substitua isso por H4 e você verá que a formatação é realmente diferente. Então, usando o Markdown, posso criar um arquivo README bem formatado em apenas um piscar de olhos. Como eu disse, este não é um curso sobre Markdown, você tem esse link no slide, dê uma olhada lá se quiser obter as opções de edição completas. Deixe-me agora colar um pouco do Markdown. Vamos primeiro dar uma pequena prévia. Então, eu vou para esta imagem em H1 e H2, algum texto e uma lista. Então, aqui temos a imagem no topo, e o conteúdo H1 prefixado com esse sinal de hash. Dois sinais de hash transformam essa linha em H2, algum conteúdo. Três sinais de hash transformaram-no em um H3. E então temos alguns itens prefixados com um traço, que os tornam uma lista. Tudo bem, ao editar arquivos no GitHub, basicamente não estamos apenas editando um arquivo, na verdade estamos criando um novo commit. Então isso é algo que você precisa entender. Sempre que você estiver editando um arquivo, será necessário criar uma confirmação. Embora não seja realmente a maneira de editar arquivos, mesmo se você o fizer on-line diretamente via GitHub, na verdade você estará criando um novo commit. Portanto, o commit deve sempre ter uma mensagem. No nosso caso, vamos dizer Criar arquivo README. Também podemos adicionar uma descrição opcional. No nosso caso, agora temos a capacidade, temos as permissões, digamos, de se comprometer diretamente com o ramo mestre. Também podemos criar uma nova ramificação e percorrer todo o caminho através de uma solicitação pull, o que não é necessário para o que estamos fazendo agora. Então, vamos deixar como padrão aqui, onde estamos comprometendo diretamente com o branch master um novo commit. E lá vamos nós. O arquivo README acabou de ser criado e automaticamente está sendo buscado pelo GitHub para mostrar que você está diretamente na página de destino do repositório. Outros arquivos conhecidos no GitHub incluem o arquivo LICENSE. Você já viu o arquivo LICENSE quando cliquei nele. Vemos a contagem bruta real aqui, mas como é um arquivo conhecido, ele também é renderizado de forma diferente pelo GitHub. Para concluir esta demonstração, vamos criar outro arquivo conhecido, que é o arquivo MD contribuinte ou contribuinte. O arquivo MD contribuinte será mostrado para as pessoas que desejam contribuir com seu repositório. Então, basicamente, ele contém tipicamente algumas diretrizes. Vamos inserir algum conteúdo aqui, criar uma nova confirmação, confirmar um novo arquivo e lá vamos nós, agora temos esse arquivo aqui também. Então agora você já viu como poderíamos adicionar arquivos diretamente via GitHub e, como já mencionado, enquanto podemos fazê-lo, não é a abordagem mais usada para criar novos arquivos em seu repositório. Vamos voltar aos slides. Trabalhando com o GitHub localmente Muitas ações podem ser realizadas diretamente através da interface do GitHub. No entanto, como o GitHub é o controle remoto ao trabalhar com o Git, você terá que realizar muitas interações com o GitHub localmente. Vamos dar uma olhada em como podemos fazer isso usando o que aprendemos no módulo anterior. Ao trabalhar com o git remoto, os desenvolvedores normalmente usam o fluxo a seguir. Suponha que você comece a trabalhar com um repositório, que já existe no GitHub. Então, basicamente, o repositório remoto, ou "central", já está lá. E para ficar totalmente claro no fluxo de trabalho do GitHub, o GitHub será o repositório remoto ou, resumidamente, o remoto. Na verdade, nada mais é do que o local central onde o código é armazenado para que todos possam acessá-lo. Frequentemente falamos sobre o repositório upstream, que é outro nome para o repositório remoto. A primeira coisa que você normalmente faz é executar um clone do git. Isso criará uma cópia local em sua máquina do repositório remoto para você trabalhar. Um comando clone inicialmente também criará ramificações de rastreamento remoto para todas as ramificações no repositório clonado e fará o check-out da ramificação ativa padrão do controle remoto. Ainda não falamos sobre ramificações, portanto, não se preocupe se você não está seguindo completamente o aspecto das ramificações aqui. Isso virá mais tarde. Por enquanto, lembre-se de que o clone executará uma cópia completa do repositório remoto no repositório local e, em seguida, podemos começar a trabalhar em nosso próprio diretório de trabalho. Você verá na próxima demonstração que o GitHub tem em sua interface uma maneira fácil de iniciar o processo de clone. Agora podemos editar os arquivos ou até adicionar novos arquivos. Posteriormente, podemos adicionar ao Git para que eles agora sejam rastreados usando o comando git add. Usando o comando add, podemos basicamente garantir que o Git comece a rastrear esses arquivos. Quando estivermos satisfeitos com nosso trabalho, podemos executar uma confirmação para que nossas alterações sejam registradas no repositório local. E só para esclarecer, os justos neste momento não querem interação com o GitHub. Executamos esse clone inicial, o que nos deu essa cópia local. Todos os outros trabalhos até agora eram locais, mas agora fizemos alterações no código e você deseja compartilhá-los com outras pessoas. E para isso, é claro que usaremos o comando git push. Agora, ao trabalhar com várias pessoas em um projeto, é provável que outras pessoas tenham feito alterações enquanto você estava trabalhando em outro recurso. Nesse caso, seu comando push resultará em conflitos e será sua responsabilidade corrigi-los. Portanto, na maioria dos casos, é recomendável que, antes de executar o push, você verifique basicamente se outras alterações foram feitas. Nesse caso, convém trazer essas alterações para a máquina local e corrigi-las primeiro. Você basicamente tem duas opções aqui. Primeiro, podemos realizar uma busca no Git. Isso derrubará as alterações feitas no repositório upstream, que você ainda não possui localmente. No entanto, ele irá parar por aqui. Nenhuma mesclagem será feita ainda. Somente quando você executar uma mesclagem de git agora suas alterações serão mescladas. Nesse caso, algumas coisas serão executadas automaticamente, enquanto outras exigirão uma intervenção manual de você. Como alternativa, você pode executar um git pull. Um pull pode ser visto como um atalho, em que uma busca e uma mesclagem de alterações no repositório local serão feitas de uma só vez. Podemos basicamente dizer que um git pull é a soma de uma busca git e de uma mesclagem git. Após uma fusão bem-sucedida, ainda podemos executar o push de volta ao GitHub, para que outros possam ver as alterações que fizemos localmente. Demonstração: Interagindo com o GitHub Agora que falamos sobre o fluxo de como podemos trabalhar com o GitHub como controle remoto, é hora de dar uma olhada em uma demonstração estendida em que veremos isso em ação. Começaremos a partir de um repositório já existente, que clonaremos na máquina local. Em seguida, faremos algumas alterações e, depois, enviaremos essas alterações ao GitHub. Usaremos os comandos mencionados nos slides anteriores. Tudo bem, agora vou começar criando um clone do meu repositório GitHub. Portanto, o GitHub tem esse botão Clone ou download que já vimos na demonstração anterior. Eu posso clicar nele e ele me mostra um URL de clone. Nesse caso, ele está usando SSH, mas se eu quiser, também posso mudar para a versão HTTPS. Mas continuaremos com o SSH. Clique neste botão aqui para copiá-lo. Aqui vamos nós. Agora voltarei à interface da linha de comandos. Então agora eu estou no diretório code / Pluralsight, não estou no repositório com o qual trabalhamos anteriormente. Então agora aqui, digito git clone e colo no URL que acabei de copiar do GitHub. Isso é colher de laranja-branqueada / difusa. git. Ele vai clonar em colher difusa e baixou os arquivos. Vamos dar uma olhada. Vamos para o diretório fuzzy-spoon, que foi criado automaticamente para mim. Esse é um diretório Git porque mostra o mestre no final. Isso é bom. Aqui, podemos fazer um ls -la para dar uma olhada em todos os arquivos que foram clonados. No diretório fuzzy-spoon, existe de fato o. diretório git, então esse é o banco de dados Git; há o. arquivo gitignore; e os arquivos que criamos anteriormente para contribuir, o arquivo README e o arquivo LICENSE que já criamos anteriormente. Se fizermos um status git agora, basicamente não há nada a ser confirmado neste momento. Estamos em um diretório de trabalho limpo. Agora, o Git também está dizendo que nossa branch principal local está atualizada com o mestre de origem. De fato, existe uma ramificação de rastreamento no GitHub para essa ramificação principal local, e o Git especificou que eles estão sincronizados. Então, o que vou fazer agora é copiar alguns arquivos que compõem um site muito básico, e começaremos a trabalhar com isso como nosso projeto que usamos no GitHub. Aqui você vê a cópia local do diretório fuzzy-spoon, e aqui vou colar alguns arquivos que compõem um site muito básico. Deixe-me mostrar o índice. html. Você vê um site muito básico, com algum texto e algumas imagens. Ele também possui CSS e jQuery. Se agora voltarmos à linha de comando e fizermos um status git, veremos que alguns arquivos precisam ser adicionados. Tudo na pasta de ativos, na pasta de imagens, no favicon e no índice. html. Esses são realmente os arquivos que eu adicionei. Neste ponto, já sabemos disso. Nós não os adicionamos ao Git, portanto, o Git não está rastreando-os, então vamos fazer isso primeiro. Não preciso especificar todos os arquivos individualmente. Simplesmente dizendo git add e um ponto, e todos os arquivos que não estão sendo rastreados no momento serão adicionados. Aqui vamos nós. Se agora eu fizer outro g status, está me mostrando alguns arquivos novos que acabei de criar. E eu quero confirmar todos esses novos arquivos, então eu vou criar um novo commit, então eu farei isso usando git commit -m, e então eu especifico minha mensagem de commit. Essa mensagem será adicionada aos arquivos do site. Pressione Enter e agora todos os arquivos foram confirmados localmente. Se eu fizer outro status de git, agora vemos uma mensagem diferente. O Git agora está dizendo que minha ramificação principal local está à frente de sua origem na ramificação de rastreamento, digamos, no GitHub. Neste ponto, eu criei apenas um commit local, que não foi publicado no GitHub. Se eu quiser fazer isso, precisarei fazer um push git para publicar meus commits locais, e somente então outras pessoas poderão ver minhas alterações. É isso que vou fazer a seguir. Então, o que eu quero fazer no Publish, vou dizer git push. Preciso especificar que quero enviar para a origem e, portanto, ele sabe que precisará enviar para o GitHub, porque é assim que a origem está configurada. Agora, o que eu vou empurrar? Bem, meu ramo mestre local. Pressione Enter e lá vamos nós. Enviamos com sucesso nossas alterações, nosso compromisso local com o GitHub. Se fizermos outro status de git, veremos que nossas branches estão novamente atualizadas. A branch local e a branch remota estão sincronizadas. Se você voltar ao GitHub, e vamos fazer uma atualização desta página, agora veremos que minha pasta de ativos, minha pasta de imagens e meu índice e meu favicon foram publicados juntos há 4 minutos. Parabéns. Agora criamos um commit local que agora enviamos ao GitHub. Adicionando e editando arquivos no GitHub Embora na maioria das vezes você codifique e edite arquivos na sua máquina local, há momentos em que é útil usar o GitHub diretamente. Imagine que você queira fazer uma pequena alteração em um arquivo de uma máquina em que você não configurou uma versão local desse repositório, e pode ser realmente útil poder fazer isso diretamente na interface. Como você pode ver na captura de tela aqui, o GitHub possui um editor que podemos usar para editar arquivos existentes e criar novos arquivos. Quando estivermos prontos para editar nossos arquivos, podemos confirmar essas alterações diretamente ou criar uma solicitação de recebimento. Falaremos sobre solicitações pull mais adiante neste curso. Por enquanto, usaremos a opção para confirmar as alterações diretamente. Demonstração: Adicionando e editando arquivos no GitHub Na próxima demonstração, usaremos a interface do GitHub para editar arquivos e criar novos arquivos. Agora isso por si só não é muito emocionante. Vamos combinar isso com o uso de puxar e buscar aqui também. Então, primeiro vou fazer algumas alterações diretamente novamente no GitHub. Novamente, normalmente não é assim que se trabalha, mas usarei essa abordagem agora para simular que várias pessoas estão trabalhando nesse diretório. Então, deixe-me, por exemplo, editar este índice. html. Eu vejo o código HTML, então vou editar este arquivo clicando no lápis aqui. E vamos mudar o título desta página. Vamos adicionar alguns pontos de exclamação. Criaremos um novo commit, Change title of Index. Vamos criar diretamente um novo commit e, se voltarmos à página inicial do repositório, veremos que o índice acabou de ser alterado. Também vou criar um novo arquivo, vamos criar um arquivo de texto. Por exemplo, instruções. TXT. Aqui estão as instruções. Não usarei o Markdown para este. E isso é baseado em algum texto de lorem ipsum, e eu vou colocar quebra de linha para que possamos ver todas as instruções. Vou criar outro novo commit. Vou passar a mensagem. Crie o arquivo de instruções e faça outra confirmação na ramificação principal. Tudo bem, agora eu fiz algumas alterações no repositório GitHub, então o repositório GitHub está realmente à frente do meu repositório local. Meu repositório local, neste momento, não tem conhecimento das alterações que fiz, portanto, precisamos trazer esses dois de volta em sincronia. Para fazer isso, poderei usar o comando buscar ou puxar. O comando buscar, basicamente, só entra nas alterações no banco de dados Git, então no. repositório git, enquanto o pull também fará automaticamente uma mesclagem. E a mesclagem pode ser feita de qualquer maneira. Pode ser que as alterações recebidas precisem ser mescladas manualmente ou pode ser aplicado um avanço rápido. Basicamente, isso não gera alterações conflitantes e o Git será capaz de lidar com isso sozinho. Então, vamos em frente e tente isso agora. Então, agora de volta aqui no diretório de trabalho, se fizermos outro status de git, veremos que nossa branch localmente está atualizada. Novamente, ele não conhece as alterações que fizemos remotamente. O que vou fazer primeiro é uma busca por git. Como mencionado, a busca não será destrutiva. Ainda não fará nenhuma fusão. Ele simplesmente fará o download das alterações do controle remoto para o banco de dados Git local e, em seguida, podemos retirá-las e mesclar as coisas manualmente. Então eu vou fazer uma busca por git. Isso é basicamente, digamos, espreitar o que mudou. Então, agora ele baixou as alterações vindas do GitHub. Nossa branch local está realmente atrasada. Então agora eu poderia fazer uma mesclagem manual de git, mas também poderia fazer um puxão de git. Portanto, o git pull realmente executou uma busca e uma mesclagem. De fato, a mesclagem percebeu que não temos alterações conflitantes. De fato, o índice. html foi alterado, mas não houve conflitos porque localmente meu arquivo não foi editado e as instruções. txt é um novo arquivo, então esse arquivo foi simplesmente baixado. Portanto, a mesclagem de avanço rápido foi feita localmente, agora tenho as alterações. Se agora eu fizer outro status de git, veremos que nossa branch local está novamente atualizada com o mestre de origem. Tudo bem, próximo problema. E se agora fizermos alterações remotamente e localmente? Então vamos tentar. Então vamos editar as instruções. txt novamente no GitHub, e adicionarei algum texto aqui. Por favor, consulte o site para mais instruções. Aqui vamos nós. E criaremos outro commit. Atualize para o arquivo de instruções. Confirme as alterações. O que farei a seguir é editar localmente meu índice. html. Então, eu vou escrever o índice do bloco de notas. html. Isso abrirá o Bloco de notas, permitindo editar o índice. html. Eu direi, bem, um ponto de exclamação é mais que suficiente. Vamos salvar isso, fechá-lo, lá vamos nós. Portanto, neste ponto, agora temos alterações localmente e no GitHub. Então, a primeira coisa que preciso fazer é realizar a mudança local, fazer um git add e usar o ponto aqui. Isso agora encenou meu índice alterado. html. Em seguida, é claro que usarei o comando commit para confirmar minhas alterações locais. Portanto, usarei o git commit e passarei a mensagem -m, Index alterado, menos pontos de exclamação. Um arquivo foi alterado. O índice foi alterado. Aqui vamos nós. Tudo bem, agora preciso fazer um empurrão porque realmente fiz alterações. Quero que todos da minha equipe vejam essas mudanças. Então, eu vou fazer um mestre de origem git push. Agora, o que você vê agora é que o GitHub rejeitou meu envio, rejeitou minha alteração. Ele nos dá algumas dicas dizendo que os objetos foram rejeitados porque remotos já temos algumas alterações que não temos localmente, e isso é causado por outro repositório que envia para a mesma referência. Em outras palavras, alguém fez alterações no repositório do GitHub, o controle remoto que ainda não tenho localmente. Então, o que eu gostaria de fazer agora é fazer outro git pull para trazer as mudanças do GitHub. Agora, o Bloco de notas é aberto automaticamente porque ele foi mesclado automaticamente. Quando ele realmente cria uma mesclagem, um novo commit será criado. Eu posso inserir minha própria mensagem aqui. Vamos deixá-lo para o padrão e fecharemos isso. E lá vamos nós. Portanto, as alterações que eu tinha localmente agora são combinadas com as alterações extraídas do GitHub para um novo commit de mesclagem. De fato, um commit extra foi criado no qual as duas alterações foram mescladas. Se tivéssemos tido conflitos de mesclagem, eu teria que fazer uma mesclagem manualmente, resultando no mesmo commit de mesclagem no final. Neste ponto, meu local ainda está à frente do meu controle remoto, então ainda precisamos fazer um git push. O mesmo ocorre com o mestre de origem git push. Lá vamos nós, tudo está atualizado novamente. Então, agora vamos voltar ao GitHub, fazer outra atualização e, de fato, o commit que agora vemos aqui é o resultado da fusão de minhas alterações locais com as alterações feitas no GitHub anteriormente. Espero que isso já lhe dê uma boa idéia de como você pode trabalhar com o Git e o GitHub juntos. Arquivando Repositórios Se, com o tempo, você decidir que seu repositório não é mais necessário, ou seja, o projeto não será mais mantido, é uma boa ideia informar as pessoas sobre isso em vez de criar um projeto fantasma. Isso pode ser feito usando a função de arquivamento no GitHub. É recomendável que você também atualize o arquivo README para que as pessoas saibam sobre isso, além de fechar problemas e receber solicitações. Uma vez arquivadas, as informações no repositório não são perdidas. No entanto, todos os itens, como problemas, tornam-se somente leitura. Demo: Arquivando Repositórios Vamos dar uma olhada em como podemos arquivar um repositório no GitHub na próxima demonstração. Se você deseja arquivar um repo, é claro que precisa navegar primeiro pela página de destino do repo. Claro, agora vamos arquivar o repositório que usamos ao longo deste curso. Vou usar o primeiro repositório para isso. Então, aqui estamos na primeira página do repositório. Eu vou para Configurações, e então rolamos todo o caminho. Na parte inferior da página, há essa zona de perigo marcada em vermelho. Aqui podemos fazer algumas coisas difíceis de reverter. E um deles está arquivando este repositório. Você também pode excluir o repositório daqui, mas isso é ainda mais destrutivo. O arquivamento é basicamente usado para indicar que esse repo não está mais ativo. Você ainda pode lê-lo, mas não deve mais contribuir com isso ou esperar que eu faça mais alterações. Vamos clicar neste botão Arquivar e, em seguida, o GitHub me avisará. Coisas ruins inesperadas acontecerão se você não ler isso. E o que está dizendo aqui é que o repositório será feito somente leitura. Uma das coisas que as pessoas ainda podem fazer é bifurcar. Conversaremos sobre bifurcar-se mais tarde no curso Agora, o arquivamento não é realmente tão destrutivo. Como eu disse, ele se torna somente leitura e podemos desarquivar o repositório a qualquer momento posteriormente. Portanto, se decidirmos que ainda queremos continuar trabalhando no projeto, podemos desarquivá-lo. Portanto, para arquivá-lo, preciso confirmar, não apenas apertando um botão, preciso digitar o nome do repositório para que eles saibam que tenho certeza absoluta de qual repositório estou arquivando. Só então o botão ficará ativo, eu posso clicar nele, arquivar este repositório. Aqui vamos nós. E como você pode ver aqui, agora é somente leitura. Se agora formos dar uma olhada neste repositório de outra janela em que estou bloqueado com outro usuário, você poderá ver a mensagem na parte superior dizendo que este repositório foi arquivado. Agora, ele está em um estado somente leitura. Por exemplo, o que não posso fazer é abrir um problema aqui. Não há capacidade de adicionar um novo problema aqui, o que significa que o repositório está realmente no modo somente leitura. Recursos de Repositório Agora que você viu como podemos trabalhar com o repositório hospedado no GitHub, vamos explorar alguns dos muitos outros recursos com repositórios oferecidos pelo GitHub. Eu já mencionei algumas vezes que o repositório é o bloco de construção fundamental no GitHub. Tudo basicamente gira em torno de repositórios. Agora vimos como podemos trabalhar com eles, mas como mencionado, o GitHub oferece muitos recursos extras. Isso nos permite trabalhar ainda melhor com repositórios. Podemos atribuir tópicos a um repositório. Você pode pensar em um tópico como uma tag, usada para classificar seu projeto. Eles podem ser atribuídos por você ao seu repositório. Depois de aplicar um ou mais tópicos, eles podem ajudar a descobrir seu projeto com mais facilidade quando alguém está realizando uma pesquisa. Além disso, ao clicar em um tópico na interface do GitHub, outros repositórios com tópicos semelhantes serão mostrados. O próprio GitHub mantém uma lista de tópicos, bem, aqui de fato um repositório. Esses tópicos também são usados ​​para alimentar a página de exploração no GitHub, onde você pode navegar pelos projetos. Nas configurações de um repositório, também podemos trabalhar com projetos. Normalmente, usamos um projeto ou quadro de projetos para acompanhar o progresso. Exploraremos as placas do projeto mais adiante neste curso. Em nosso repositório, também podemos ver e gerenciar problemas. Os problemas são a nossa maneira de rastrear tudo, como bugs, tarefas, solicitações de recursos e muito mais. Vamos passar um módulo inteiro explorando os problemas com muito mais detalhes posteriormente neste curso. Outro conceito importante vinculado ao repositório é o conceito de uma solicitação pull. Já falamos brevemente sobre isso, e vamos explorá-los também com muito mais detalhes posteriormente no curso. Usando solicitações pull, o GitHub facilita a colaboração no Git, permitindo que as pessoas sugiram mudanças e permitindo que os proprietários do repositório revisem essas alterações propostas. Também em repositórios, o GitHub oferece a capacidade de usar os chamados insights. Como o nome é divulgado, isso fornece muitas informações sobre o seu projeto, incluindo o tráfego para as páginas do seu repositório. Por fim, usando as configurações de um repositório, podemos adicionar várias outras configurações sobre o repositório, incluindo os colaboradores, a ramificação padrão e muito mais. Demo: Explorando os Recursos de um Repositório Em vez de falar sobre esses recursos e configurações do repositório, vamos voltar ao GitHub e dar uma olhada neles na próxima demonstração. Também gerenciaremos os tópicos de um repositório para que agora seja categorizado. Então, vamos dar uma olhada nas muitas opções que ainda temos em um repositório. Uma das primeiras coisas que você vê aqui é que, abaixo da descrição, você tem esse botão Adicionar tópicos. Clique nele e agora podemos adicionar tópicos separados por espaços. Por exemplo, posso dizer que este é um site. Posso dizer que isso está escrito em HTML. Está contendo algum CSS. Ele contém jQuery e assim por diante. Quando clico em Concluído, todos esses tópicos se tornam clicáveis ​​e posso clicar neles e, ao fazer isso, navegarei para uma página com tópicos semelhantes. Portanto, os tópicos são realmente usados ​​para categorizar seu repositório, permitindo que as pessoas pesquisem a grande quantidade de informações disponíveis no GitHub. Na página do repositório, ainda temos muitas outras opções, todas elas ainda serão exploradas mais adiante neste curso, mas quero lhe dar mais algumas dicas sobre elas. Através de problemas, basicamente posso criar um novo bug, uma nova solicitação de recurso, um novo ID. Issues é basicamente o rastreador de problemas disponível no GitHub. Posso clicar aqui em Criar novo problema e, por exemplo, especificar que descobri um pequeno bug no código que desejo acompanhar. Então, posso dizer, por exemplo, que um título não está bem alinhado. Aqui posso novamente usar o Markdown, questão importante. Posso visualizar isso e, como você pode ver, usa a mesma marcação que vimos anteriormente ao editar o arquivo README. Podemos fazer muitas outras coisas com problemas e, novamente, exploraremos isso mais tarde. Envie um novo problema e, em seguida, esse problema será dissociado com nosso repositório. Agora, outras pessoas também podem comentar sobre isso, mas, novamente, veremos mais adiante. Em seguida, temos solicitações de recebimento. A solicitação de recebimento será explorada com muitos detalhes no próximo módulo, mas, para lhe dar uma idéia, as solicitações de recebimento serão usadas por outras pessoas que fizeram alterações e desejam enviá-las de volta ao meu repositório original. Eles estão basicamente me pedindo para puxar essas alterações de volta para outro ramo. Você tem o botão Nova solicitação de recebimento aqui. A partir daqui, podemos criar novas solicitações de recebimento. Mais uma vez, faremos isso no próximo módulo. Em seguida, temos projetos, e os projetos são basicamente painéis de projetos, permitindo que eu gerencie visualmente um quadro que contenha todas as tarefas, problemas, digamos, do meu projeto. Eu posso clicar aqui em Criar um projeto. Vou criar um novo projeto aqui, Sample board. Posso dar uma descrição e também selecionar um modelo. Nesse momento, selecionarei o Kanban básico, Criar projeto, e você verá uma visão geral do quadro de projetos do seu projeto, na qual poderá gerenciar e priorizar o trabalho. Em seguida, no repositório também podemos adicionar um wiki, que basicamente será usado para documentação. Mais uma vez, exploraremos isso mais adiante no curso com muito mais detalhes. Vou pular o Insights por enquanto, deixe-me ir para Configurações. Em Configurações, posso fazer algumas coisas com meu repositório, como alterar o nome, ativar ou desativar os recursos, adicionar páginas do GitHub, que novamente exploraremos mais adiante no curso e fazer coisas como arquivamento, que já vimos em, ou até mesmo excluir o repositório. Há algumas outras configurações aqui que veremos mais adiante neste curso. Trazendo mais pessoas Você já deve ter entendido que o GitHub é uma plataforma de codificação social. No repositório, podemos trabalhar com muitos para atingir uma meta. No GitHub, podemos distinguir dois grupos ao trabalhar em um projeto. Primeiro, colaboradores. Os colaboradores são um grupo fixo de pessoas, portanto, a equipe de desenvolvimento principal que trabalha em um projeto. Eles normalmente têm mais permissões no repositório. O mais importante aqui é confirmar o acesso no repositório principal. Em outras palavras, você como proprietário do projeto confiará em uma ou mais pessoas para poder alterar o código no repositório principal do projeto. Este é realmente um grupo com curadoria que pode ser gerenciado através das configurações do repositório. Em seguida, temos colaboradores. Isso pode ser basicamente todo mundo de fora da equipe principal, então a equipe de colaboradores. Um colaborador é alguém que talvez use seus projetos em seu trabalho diário e esteja propondo uma mudança ou melhoria. A grande diferença, é claro, é que os contribuidores, então basicamente todos, têm permissões mais baixas. Eles não podem se comprometer com seu ramo principal. As alterações propostas serão apresentadas como solicitações pull, que poderão ser revisadas. Falarei sobre solicitações pull mais tarde. Demonstração: Trabalhando com colaboradores Na próxima demonstração, vamos ver como podemos adicionar um colaborador à equipe principal do projeto. Esta é a minha conta original, que obviamente é a proprietária do projeto fuzzy-spoon. Embora este seja um repositório público, é claro que somente agora eu posso editar este projeto. Se você olhar o projeto a partir de uma janela diferente do navegador, neste caso, eu estou olhando o projeto usando a conta Gill-Green, não tenho permissões no projeto e você pode pensar, bem, você pode criar um novo arquivo ou, quando você abre um arquivo, parece que ainda pode editá-lo ou excluí-lo. No entanto, embora possa parecer assim, se eu clicar nesse botão, na verdade criaria uma bifurcação, uma cópia, em minha própria conta e trabalharia nessa. Isso é claro, porque ainda não tenho permissões no projeto. Vamos mudar isso. Então, de volta à conta Gill-Orange, posso acessar as Configurações do repositório e clicar em Colaboradores. Como você vê aqui, ele diz que ainda não temos colaboradores, então posso pesquisar por Gill-Green, que é minha outra conta demo. Eu posso adicionar essa conta como colaborador. Aqui vamos nós. Agora posso copiar o link do convite. Também vou receber isso por e-mail. Volto à minha outra janela do navegador, colo este link e vejo que a laranja-guelra realmente me convidou, a guelra-verde, a colaborar. Eu aceitarei isso. Como você pode ver agora, agora tenho acesso push ao repositório gill-orange / fuzzy-spoon. Em outras palavras, eu posso enviar alterações diretamente no ramo mestre. Vamos tentar isso. Agora, por exemplo, posso ir para as instruções e clicar em Editar aqui, digamos que Gill-Green fez algumas alterações aqui e, como você pode ver agora, posso confirmar diretamente no ramo mestre, porque agora sou colaborador deste projeto. Se você voltar para a conta original, posso ver que o Gill-Green agora empurrou o acesso a esse repositório. E posso revogar esse acesso da maneira mais fácil possível. Informações do Repositório Na seção final deste módulo, quero destacar outro recurso interessante que é p arte das configurações do repositório, e isso é insights. Como mencionado anteriormente, os insights fornecem informações internas sobre o seu projeto. Usando informações, você pode obter informações sobre os colaboradores do seu projeto, mostrando basicamente quem é o colaborador mais importante ou ativo. Talvez isso faça você decidir que eles podem se tornar parte da equipe principal do projeto e, assim, se tornar um colaborador. Usando tráfego, vemos informações como números de clones ou visitas ao repositório. Confirmações logicamente fornece uma visão geral das confirmações feitas no repositório. Outros gráficos, como a frequência do código, fornecem informações sobre a atividade do projeto. Na captura de tela que você pode ver aqui, vemos algumas informações de um repositório muito ativo no GitHub, a saber, o repositório dotnet. Observe que nem todas as informações disponíveis para um proprietário de repo estão disponíveis publicamente. Demonstração: Explorando Insights do Repositório Na demonstração final deste módulo, examinaremos as informações de informações do nosso repositório. Na demonstração final, vamos dar uma olhada nos insights. Então, quando você acessa a página de destino do projeto e clica em Insights, chega aqui nesta página de pulso, que é basicamente o pulso do seu projeto. Você pode ver que atualmente há um problema ativo, que foi o que criei algumas demos atrás. Ainda não há solicitação de recebimento. Não houve nenhum problema fechado. Também vejo alguns autores que estão trabalhando neste projeto. Também posso dar uma olhada nos contribuidores deste projeto e quanto código eles foram alterados. Também vejo como meu projeto se comporta em termos de padrões da comunidade. Meu projeto tem uma descrição. Tem um arquivo README. Mas não possui, por exemplo, um código de conduta. Também não tenho modelos ainda. A partir daqui, posso adicioná-los diretamente. Também posso dar uma olhada no tráfego, quantas pessoas estão olhando no meu repositório. Não há muitos, apenas dois visitantes únicos, provavelmente os que eu tenho usado até agora. Também posso ver quantos commits tenho feito. Até agora, fizemos oito commits aparentemente. A frequência do código ficará bastante vazia, porque isso realmente precisa de mais dados por um longo período de tempo. E, na verdade, somos capazes de ver quantas alterações foram feitas no código ao longo do tempo. Também temos na Rede uma visão geral dos commits e das ramificações que foram feitas. E também sob Forks, vemos quantos garfos, se houver, foram criados. Este é um projeto de demonstração e, claro, não é muito ativo. Se você voltar, por exemplo, ao ASP. NET MVC, clique aqui no Mvc e, em seguida, veja o Insights aqui, você verá que há algumas solicitações de recebimento ativas. Geralmente, muitos contribuidores. Como você pode ver aqui, muitas pessoas têm trabalhado nisso. A frequência do código é bastante alta, como você pode ver. Há muitas adições e exclusões por semana. E se você der uma olhada em Forks, verá que muitas pessoas estão realmente bifurcando o Mvc. Resumo Chegamos ao final deste módulo, que foi totalmente dedicado aos repositórios. Penso que para resumir, é seguro dizer que tudo no GitHub evolui em torno do uso de repositórios. Em nosso fluxo de trabalho, usamos o GitHub localmente, o GitHub se tornou o controle remoto. Além do que um repositório oferece por padrão, o GitHub oferece muitos serviços adicionais, e acabamos de mostrar isso neste módulo. No próximo módulo, estaremos construindo o que vimos neste módulo. Começaremos a trabalhar com branches e fusões e examinaremos o conceito de solicitações pull. Espero vê-lo lá também. Ramificação, mesclagem e solicitações pull no GitHub Introdução ao módulo Olá. É ótimo que você esteja se juntando a mim para outro módulo neste curso de Fundamentos do GitHub, aqui no Pluralsight. Meu nome é Gill Cleeren e também serei seu instrutor para este módulo. No último módulo, nos divertimos muito explorando as muitas facetas dos repositórios, a pedra angular do GitHub. Embora tenhamos abordado muito terreno, na verdade não abordamos vários outros tópicos que estão intimamente relacionados aos repositórios. Portanto, neste módulo, desenvolveremos o conhecimento do anterior e aprenderemos sobre como podemos trabalhar com ramificações em nossos repositórios, realizar uma mesclagem e usar os conceitos tão importantes de solicitações pull no GitHub . Vamos dar uma olhada na agenda deste módulo para começar. Iniciaremos o módulo examinando como o Git funciona com ramificações. Se você é proveniente de um sistema tradicional de gerenciamento de código-fonte, pode estar pensando em ramificações de uma maneira diferente do Git, por isso definitivamente vale a pena dar uma olhada em como isso é tratado no Git. Depois disso, aplicaremos o conhecimento recém-adquirido no GitHub e veremos como a plataforma funciona com ramificações. Você logo entenderá que a criação de uma ramificação é uma operação muito simples no Git, e essa é basicamente a razão pela qual todo o fluxo do GitHub se baseia nele. Vamos explorar o fluxo do GitHub, construído em torno do sistema de solicitações pull. Animado? Tudo certo. Este módulo contém alguns conceitos fundamentais, por isso é importante que você os compreenda na íntegra. Vamos fazer isso. Uma visão geral das branches Como mencionado, precisamos iniciar nossa exploração de branches com uma breve visão geral do que são branches no Git e, portanto, também no GitHub. Vamos fazer isso primeiro. O conceito de ramificação certamente não é novidade e não é algo que se aplica apenas ao Git. Outros sistemas tradicionais de controle de versão também oferecem a capacidade de criar ramificações. Se você não está familiarizado com o conceito, não é realmente nada complexo. Basicamente, em vez de escrever todo o código em uma linha e continuar construindo isso, em algum momento divergimos da linha de desenvolvimento principal. Então, veja-o como uma linha de trem que continua e continua e, em algum momento, uma linha lateral sobe. Nessa trilha lateral, podemos fazer todo tipo de coisa. Decodificamos com isso, impactando a linha principal de trabalho. Se gostamos do que fizemos na pista lateral, podemos em algum momento, como nos trens novamente, convergir novamente. As mudanças que fizemos na trilha lateral são então mescladas com as que temos na trilha principal de desenvolvimento. Como eu disse, isso também está disponível em outros sistemas de gerenciamento de código-fonte; no entanto, nos tradicionais, isso geralmente é uma operação cara. Normalmente, é necessário criar uma cópia do diretório do código-fonte e trabalhar nele. Se o diretório do código-fonte for muito grande, isso levará tempo. Agora, o conceito de ramificação no Git é diferente. É feito de uma maneira completamente diferente, tornando-o mais leve; portanto, a criação de um ramo é muito rápida. No GIT e no GitHub, é realmente recomendado que você trabalhe com várias ramificações no desenvolvimento de aplicativos. Criar ramificações, alternar entre ramificações e mesclar ramificações é realmente muito rápido e, portanto, é a maneira recomendada de trabalhar. Lembre-se do fato de que, inicialmente, o Git foi criado para oferecer suporte a equipes grandes e disseminadas trabalhando em coisas como o kernel do Linux. Se todos estivessem trabalhando no mesmo ramo, isso seria realmente difícil. Mesmo com o suporte de ramificação, em um sistema tradicional de código-fonte, esse cenário também seria muito difícil, pois muitas cópias ocorreriam para trabalhar com ramificações. Conforme definido no Git, é muito mais rápido e funciona devido à maneira como o Git funciona internamente. O Git não funciona com diferenças, mas com snapshots, e esses são basicamente o facilitador para o sistema de mesclagem leve. Agora, vamos dar uma olhada nesses instantâneos com mais detalhes, pois não abordamos isso até agora neste curso. Nos sistemas tradicionais de código fonte, estamos verificando informações. Essas informações são armazenadas como uma lista de alterações baseadas em arquivo. Basicamente, todas as informações são armazenadas como arquivos e as alterações feitas nesses arquivos ao longo do tempo. Os deltas criados para esses arquivos são armazenados, e é disso que consiste um conjunto de alterações. É o que você vê aqui no slide. Temos alguns arquivos e, com o tempo, as alterações são feitas nesses arquivos, e essas alterações são armazenadas como deltas. Como dito, o Git e, portanto, também o GitHub, funcionam de maneira diferente. No Git, trabalhamos com uma série de instantâneos. Quando executamos uma confirmação, é feita uma captura instantânea de todos os arquivos e criada uma referência à maneira como todos os arquivos aparecem nesse ponto. Claro, isso deve soar um alarme, já que agora você pode estar pensando, bem, ele criará uma cópia de cada arquivo com cada confirmação? Bem, na verdade não faz isso. Se um arquivo não foi alterado, o Git armazenará apenas o link para o arquivo idêntico anterior. Essa maneira fundamentalmente diferente de trabalhar é uma das principais diferenças entre o Git e outros sistemas de gerenciamento de código-fonte. Ok, agora que você entende o conceito de snapshots, vamos tentar voltar à ramificação, porque é disso que estamos falando aqui. Portanto, toda vez que criamos um commit, o Git armazena um ponteiro em um desses instantâneos, e o objeto de commit do qual esse ponto faz parte também contém itens como seu email e a mensagem de commit. Além disso, e isso é importante, o commit conterá um ponteiro para o commit que veio antes dele. É possível que não haja confirmação anterior, se este foi o primeiro. Pode haver um pai ou até vários, se o commit ocorrer após uma mesclagem de ramificações diferentes. No Git, um ramo não é mais que um ponteiro para um desses commits. Sempre teremos o ramo principal, que é o padrão no Git, e ele é criado automaticamente. Quando agora executamos confirmações nesse ramo, o ramo mestre, um ponteiro aponta para o último commit que fizemos e avança com todos os commit que criamos. É por isso que criar ramificações no Git é uma operação tão barata. Uma ramificação nada mais é do que um ponteiro para uma confirmação. Agora, ao usar o Git, as ramificações estão sendo usadas para quase todo o desenvolvimento. Normalmente, criamos um novo recurso em uma ramificação específica . Quando um bug precisa ser corrigido, normalmente criamos outro novo ramo. Se você quiser experimentar a mesma coisa, vamos criar outro ramo. O que acontece agora quando queremos criar um novo ramo então? Como dito, um ramo nada mais é do que um ponteiro; portanto, um novo ponteiro será criado. Por padrão, estamos no ramo principal e a criação de um ramo não muda isso. O Git acompanha isso usando a cabeça, que é um ponteiro especial usado para acompanhar em qual ramificação estamos trabalhando atualmente. Somente quando realizarmos o chamado check-out, mudaremos para nossa nova branch. Quando estamos nesse novo ramo e começamos a fazer alterações, estamos construindo-os sobre esse ramo recém-criado. O ramo principal original não mudou. Nesse ponto, sem alterações no mestre, quando estivermos satisfeitos com as alterações que fizemos na nova ramificação, poderemos mesclá-las facilmente novamente na ramificação mestre, se ela não tiver sido alterada. No entanto, se nesse meio tempo a ramificação principal tiver sido alterada, teremos que mesclar, e veremos isso mais adiante neste módulo. Demonstração: Criando uma ramificação no GitHub Criar ramificações é algo que podemos definir localmente, é claro, mas também no GitHub. Na primeira demonstração deste módulo, vamos começar examinando como podemos criar ramificações no próprio GitHub, que usaremos posteriormente neste módulo. Então aqui no GitHub. com, vou criar minha nova branch. Portanto, atualmente existe apenas um ramo, e podemos ver isso aqui neste expansor aqui. Portanto, quando clico neste mestre de ramificação, você pode ver esta caixa de diálogo informando que atualmente tenho um mestre de ramificação e, usando este campo de entrada aqui, posso pesquisar ou até criar uma nova ramificação. Então, vamos criar um novo ramo e vamos chamá-lo de exemplo-ramo. Como você pode ver, ele vai ramificar do master para esse novo branch, sample-branch. Pressione Enter e um novo ramo foi criado. O GitHub diz que esse novo ramo é igual, é igual ao mestre. Portanto, neste ponto, os dois ramos ainda são idênticos. O GitHub também está dizendo que eu posso iniciar uma solicitação de recebimento. Agora, uma solicitação de recebimento está basicamente recebendo as alterações no ramo em que estamos ramificados. Então, em outras palavras, vamos puxar as alterações feitas neste segundo ramo de volta para o mestre. Mais tarde, faremos a solicitação de recebimento. Também podemos fazer uma comparação, para que possamos ver qual é a diferença entre esse ramo e nosso ramo original. Nesse ponto, eles ainda seriam idênticos, então vamos mudar isso. Então, vamos às instruções. arquivo txt novamente, por exemplo. E vamos editar este arquivo e vamos adicionar uma linha aqui que diz, alterar o posicionamento no Branch de Amostra. Vamos criar outro commit aqui porque eu preciso criar um commit neste ramo extra. Vou criar outro commit. Aqui vamos nós. Então agora vemos algumas coisas. O GitHub diz que agora eu poderia fazer uma comparação e criar uma solicitação de recebimento porque agora percebe que há realmente mudanças entre esse ramo e o ramo mestre. Se olharmos para o ramo da amostra, de fato vemos essas instruções. O txt acabou de ser atualizado com essa linha extra. Se, no entanto, eu mudar para o mestre, você ainda verá o original. Então, aqui, eu efetivamente criei dois ramos e os arquivos são diferentes por ramo. Comandos para ramificação A demonstração anterior foi bastante curta, pois acabamos de criar uma ramificação no GitHub, e você também viu que fizemos algumas mudanças diretamente no GitHub. Embora isso seja possível, definitivamente não é o modo como você interagirá com o GitHub na maioria das vezes. Você normalmente trabalha localmente, enviando suas alterações, incluindo ramificações para o GitHub. Se você estiver trabalhando em sua máquina local, existem comandos especificamente para trabalhar com ramificações. Você provavelmente não se surpreenderá com o fato de esse comando ser chamado de ramificação git e depois especificar o nome da ramificação que deseja criar. Se você tem um ramo local com o qual deseja começar a trabalhar, precisará mudar para esse ramo. É isso que um comando de checkout fará. Você especifica qual branch deseja fazer check-out. Nesse ponto, os arquivos no seu diretório de trabalho são substituídos pelos da ramificação. Outro comando que é muito relevante nesse contexto é, é claro, o comando push. Já vimos o comando push no módulo anterior; agora precisamos usá-lo aqui novamente. Quando criamos um novo ramo, é claro que esse ramo não é enviado automaticamente para o GitHub. Somente quando usarmos o comando push, passando o parâmetro u para especificar a origem e o nome do ramo que queremos enviar, ele será enviado ao GitHub. Demonstração: Criando arquivos no GitHub Agora que você conhece esses comandos relevantes, vamos voltar para uma demonstração maior, na qual você verá esse fluxo. Criaremos uma branch local e enviaremos isso para o GitHub. Portanto, na demonstração anterior, criei o ramo diretamente via GitHub, mas como mencionado, normalmente não é o que você fará. Normalmente, você cria suas ramificações localmente e as envia ao GitHub. Então vamos fazer isso a seguir. Então, eu estou aqui novamente no repositório fuzzy-spoon e vamos verificar qual é o status atual usando o status git. Agora estamos no ramo principal e temos um diretório limpo, então nada está pendente aqui. Então agora vou criar uma nova branch local e começar a trabalhar nessa. Eu posso usar o comando git branch para criar uma nova ramificação. No entanto, normalmente usarei o comando git checkout, que fará duas coisas. Isso criará um novo ramo e me mudará automaticamente para esse ramo. É o que eu prefiro. Eu preciso passar em -b e passar o nome da ramificação. Vamos criar uma nova ramificação onde incluiremos instruções de configuração. Então, vamos nomear essa instalação add-branch. Então, como você pode ver agora, criamos um novo ramo e mudamos automaticamente para esse ramo. Deixe-me criar um novo arquivo, instalação. TXT. Então, eu vou usar o bloco de notas, instalação. TXT. O Bloco de notas aparece automaticamente e incluiremos algumas instruções de instalação. Então, aqui está o conteúdo do guia de instalação. Portanto, em seguida, preciso garantir que o Git esteja rastreando meu novo arquivo, portanto git add e um ponto e, em seguida, crie um novo commit, -m e como mensagem, o guia de instalação adicionado. Neste ponto, minha ramificação e meu commit são apenas locais. O GitHub não tem conhecimento disso, portanto, precisarei publicar meu ramo recém-criado. Então, eu vou usar o comando git push para publicar novamente, como fizemos antes, e preciso configurar um relacionamento de rastreamento entre o ramo de instalação de adição local e o ramo remoto no GitHub. Então, eu uso -u origin, e especificarei o nome da ramificação de rastreamento remoto, e vamos chamar isso de também instalação de adição. Lá vamos nós, e como você pode ver, o ramo de instalação de adição local agora está vinculado ao ramo de instalação remota no GitHub. Agora, vamos dar uma outra olhada no GitHub e, mesmo sem atualizar, o GitHub notou que eu criei uma nova ramificação, a ramificação de instalação e adição, e está sugerindo que eu faça uma solicitação de recebimento para trazer as alterações da ramificação recém-criada. de volta ao mestre. Podemos clicar nessa ramificação aqui e, de fato, vemos que o guia de instalação que criamos localmente agora também faz parte dessa ramificação remota. Solicitações de extração e fluxo do GitHub Agora que você entende os conceitos de ramificações, vamos aprofundar este tópico e aprender sobre solicitações de recebimento. Esse é um fluxo muito típico para trabalhar com o GitHub e, portanto, é frequentemente chamado de fluxo do GitHub. Vamos dar uma olhada. Até agora, o que vimos é basicamente como podemos trabalhar com ramificações no GitHub e o fluxo no qual nós, como desenvolvedores, podemos enviar a ramificação para o GitHub. Mas então o que? Bem, chegamos a um ponto em que queremos integrar as mudanças que fizemos em outro ramo, talvez o ramo mestre. O que faremos então é criar uma solicitação de recebimento. Usando uma solicitação pull, ou PR, resumimos basicamente a outras pessoas sobre o fato de termos feito alterações em uma ramificação separada. Quando abrimos a solicitação de recebimento, estamos anunciando essas alterações e abrindo-a para discussão. Outros, incluindo colaboradores, agora podem revisar essas alterações que fizemos e é possível que outros commit sejam adicionados à mesma solicitação pull. Depois que as coisas são basicamente aprovadas, elas podem ser mescladas em outro ramo do ramo que criamos. E para dar suporte a isso, o GitHub contém várias seqüências de caracteres que o ajudarão bastante na criação de uma solicitação de recebimento. Além disso, o GitHub contém uma interface para permitir, por exemplo, que o colaborador revise a solicitação de recebimento, faça anotações e assim por diante. Aqui na captura de tela dos slides, você vê a interface que você cria para criar uma solicitação pull no GitHub. Você pode inserir uma visão geral das alterações, selecionar rótulos, mestres e assim por diante. O último sobre o qual ainda não conversamos; faremos isso mais adiante neste curso. E essa maneira de trabalhar é uma maneira típica de interagir com o GitHub, chamada fluxo do GitHub. Como é muito importante, é definitivamente uma boa ideia fornecer uma visão geral gráfica completa desse fluxo. Estamos começando com o ramo. Essa é a principal linha de desenvolvimento. No nosso exemplo, digamos que seja o ramo principal, mas em muitos projetos da vida real, esse seria o ramo de desenvolvimento. Agora nos pedem para criar um novo recurso ou corrigir um bug, ou talvez você queira apenas testar algo. Então, o que faremos agora é criar uma nova ramificação a partir dessa ramificação inicial. Lembre-se, quando criamos esse ramo, você não tem nenhum impacto no ramo mestre. Somente quando pedimos explicitamente para trazer essas alterações de volta ao ramo principal, haverá algum impacto. Ok, agora temos nossa branch e estamos codificando em nosso editor favorito, na máquina local que é. E como estamos trabalhando com o Git localmente, podemos adicionar vários comentários. Tudo isso acontece em nosso ramo de recursos. Em algum momento, queremos obter algum feedback sobre o nosso trabalho. Agora isso pode ser muito cedo, mesmo com quantidades limitadas de código. Talvez você queira que alguém faça uma revisão em nosso código ou até mesmo as idéias que temos, ou também pode ser em um momento em que achamos que nosso recurso em que precisamos trabalhar está pronto. Então, vamos criar uma solicitação de recebimento. Com a solicitação de recebimento aberta, agora outras pessoas podem participar da discussão sobre as alterações propostas. Eles podem sugerir melhorias, indicar que algo está faltando ou errado. Você ainda pode, nesse momento, adicionar a essa solicitação pull e adicionar comentários adicionais. Felizmente, em algum momento, o trabalho está pronto para dar o grande salto e pode ser mesclado de volta ao ramo principal. E o ramo de recursos, nosso ramo antigo, pode ser excluído com segurança. Demonstração: Trabalhando com solicitações pull Eu acho que é bastante óbvio que precisamos fazer tudo isso em uma demonstração. Portanto, nesta demonstração, começaremos com a criação de uma solicitação de recebimento. Nessa solicitação de recebimento, como você verá, podemos adicionar comentários adicionais e também comentários. Também veremos como a solicitação de recebimento é visível no GitHub e também como a mesclagem é executada. E, finalmente, veremos como podemos excluir o ramo. Então, basicamente, o que veremos é todo o fluxo do GitHub. Assim como vimos na demonstração anterior, o GitHub percebe que criamos novas ramificações e está sugerindo a criação de uma solicitação de recebimento para que as alterações feitas nessas ramificações extras sejam transferidas para a ramificação principal. Vamos dar uma olhada em como trabalhamos com solicitações pull. Portanto, como mencionado, eu posso clicar aqui no botão Comparar e puxar solicitação e, em seguida, ela abrirá automaticamente uma solicitação puxada. Como alternativa, se isso não estiver sendo exibido, você sempre pode clicar aqui em Solicitações Pull, atualmente não há solicitações Pull abertas, e daqui também verei as ramificações publicadas recentemente, mas também posso criar manualmente uma nova solicitação Pull. Nesse patch, posso selecionar qual ramificação quero extrair em qual outra ramificação. Portanto, neste caso, quero criar uma solicitação de recebimento na qual puxarei as alterações feitas na instalação de adição no master. Vamos dar uma olhada no que está sendo mostrado nesta página quando criamos uma solicitação de recebimento. Primeiro, o GitHub está mostrando quais confirmações ou confirmações estão formando essa solicitação de recebimento. Como você pode ver aqui, há apenas uma confirmação, que faz parte dessa solicitação de recebimento. Há uma confirmação e um arquivo que foram alterados, e essa solicitação de recebimento também foi feita por um colaborador. Quando quero ver quais alterações ou alterações compõem essa solicitação de recebimento, posso vê-las aqui. Portanto, neste caso, é apenas a instalação. txt que tem novas linhas adicionadas. Nesse caso, estou vendo na exibição unificada, mas também posso definir isso na exibição dividida. E, claro, isso não é realmente útil aqui, porque a instalação. O txt não existia antes, mas se eu tiver um arquivo alterado, posso ver quais linhas foram adicionadas ou removidas aqui. Tudo bem, então agora vamos iniciar a criação da solicitação de recebimento. Uma coisa importante que não mencionei aqui é que, neste ponto, o GitHub está dizendo que é capaz de mesclar automaticamente, e você vê isso aqui. Vamos criar uma solicitação de recebimento agora. Neste ponto, agora posso comentar a solicitação de recebimento. Aqui está o meu comentário, então eu adicionei este guia e também estou pedindo que alguém faça uma revisão, neste caso, sou apenas eu trabalhando nele. Falaremos sobre comentários mais tarde. O recurso de comentário é definitivamente mais útil quando você está trabalhando com vários desenvolvedores neste projeto. Aqui à direita, você também pode ver algumas outras coisas, as quais veremos mais adiante. É possível atribuir um ou mais revisores. Novamente, para isso vou precisar de uma equipe e abordaremos isso mais tarde. Também posso atribuir um rótulo, algo que veremos mais adiante. Também posso vincular isso a um quadro de projetos ou a um mestre. Novamente, todos esses recursos serão discutidos mais adiante. Tudo bem, então agora estamos prontos. Então, vamos criar a solicitação de recebimento. Portanto, agora a solicitação pull está aberta, dizendo que gill-orange deseja mesclar uma confirmação no mestre a partir da ramificação de instalação de adição. Agora posso revisar completamente todo o contexto em torno dessa solicitação de recebimento. Portanto, há uma exibição de conversa aqui primeiro que inicialmente mostra o comentário que eu inseri tentando entrar na solicitação de recebimento. Outros membros da equipe podem continuar comentando essa solicitação de recebimento. Veja também as confirmações que fazem parte dessa solicitação de recebimento. Nesse ponto, há apenas uma confirmação, mas se agora adicionar mais alterações ao ramo de instalação de adição, elas também farão parte dessa solicitação de recebimento. Veja também como revisor, digamos, os arquivos afetados por essa solicitação de recebimento. Se necessário, eu como revisor agora posso inserir comentários aqui por linha, como você pode ver aqui. Basicamente, o que estou fazendo é uma revisão de código, como esta. E como você pode ver, eu posso começar uma revisão, isso basicamente iniciará a conversa sobre o código. Vamos cancelar isso por enquanto. Vamos voltar para a exibição de conversa aqui. O GitHub também fez uma verificação para ver se esse ramo pode ser mesclado automaticamente com o ramo base e, na verdade, diz que não há conflitos detectados, portanto agora posso mesclar a solicitação de recebimento. Aqui vamos nós. O que ele fará agora é criar um novo commit, o commit de mesclagem originado por puxar as alterações de nosso novo branch para o master. Vamos clicar em Confirmar mesclagem. E lá vamos nós, a solicitação de recebimento agora foi mesclada com êxito no ramo principal. Nossa ramificação de instalação de add-ons, bem, que não pode realmente ser excluída, não temos mais utilidade para isso. Então, vamos excluir esse ramo. Se você voltar agora à página de destino do repositório, verificamos que não há mais solicitações de recebimento abertas. Se clicarmos na solicitação de recebimento, podemos filtrar e ainda ver que temos uma solicitação de recebimento fechada. Esse é um aspecto importante. Tudo o que você está fazendo foi gravado, então não é isso porque eu fechei a solicitação de recebimento que a história se foi. Posso até restaurar minha branch, se necessário. Demonstração: Definindo a ramificação padrão Há uma coisa que não vimos nessa área, e esse é o conceito da ramificação padrão. Vamos direto à demonstração novamente para ver como podemos configurar isso no GitHub. Em uma demonstração muito curta e final deste módulo, mostrarei como podemos alterar a ramificação padrão de um repositório. Então, aqui estamos novamente na página de destino do nosso repositório fuzzy-spoon, e para alterar o ramo padrão, precisamos ir para Configurações. E em Configurações, precisamos ir para as branches. Aqui podemos mudar o ramo padrão e, como você pode ler, o GitHub está dizendo que o ramo padrão é considerado o ramo base em nosso repositório. Portanto, quando vamos fazer solicitações pull, elas serão feitas automaticamente em relação a essa ramificação padrão, a menos que, naturalmente, especifiquemos uma ramificação diferente. É por isso que é importante. É basicamente um recurso utilitário, para que suas solicitações pull sempre sejam comparadas automaticamente a uma determinada ramificação. No meu caso, esse é o ramo principal, mas eu poderia alterá-lo aqui para outro ramo que possuo. Em muitas organizações, você vê o mestre e o ramo de desenvolvimento como os dois ramos principais e, geralmente, definiríamos o ramo padrão como o ramo de desenvolvimento. Resumo Concluímos outro módulo, espero que muito emocionante. Vamos resumir o que vimos aqui. Vimos que o próprio Git tem um suporte muito forte para criar ramificações. Devido à sua natureza, é muito mais eficiente na criação de ramificações do que em outros sistemas de gerenciamento de código-fonte. O GitHub também baseou seu fluxo no GitHub nesse conceito. Através da interface, podemos trabalhar facilmente com ramificações e receber solicitações. O conceito de solicitações pull também enfatiza a natureza colaborativa do GitHub. Mais adiante neste curso, voltaremos ao aspecto social, mas é claro que eles são apoiados por solicitações pull. Outro módulo pronto, ótimo! No próximo módulo, veremos como podemos criar tags e releases. Fique ligado para mais diversão no GitHub. Criando tags e liberações Introdução ao módulo Olá novamente e bem-vindo a outro módulo do curso de Fundamentos do GitHub, aqui no Pluralsight. Eu ainda sou Gill Cleeren. Isso não mudou desde o último módulo, e nesse último módulo cobrimos ramos e conversamos muito sobre solicitações pull. Neste módulo, que será mais curto, é uma promessa. Vamos dar uma olhada em outros dois conceitos com os quais podemos trabalhar no GitHub, a saber, tags e lançamentos. A agenda para este módulo também é bastante direta. Começaremos com tags e veremos como podemos criar tags localmente e como podemos obtê-las, digamos, publicadas no GitHub. Os lançamentos estão intimamente relacionados às tags, mas há definitivamente uma diferença entre os dois. No segundo e último tópico deste módulo, exploraremos os lançamentos e seu uso no GitHub. Vamos começar! Tags: localmente e no GitHub Como dito, iniciaremos este módulo examinando as tags e exploraremos as duas tags com as quais podemos trabalhar localmente, bem como no GitHub. Mas antes de iniciarmos nossa discussão sobre tags, quero garantir que você se lembre do seguinte. No Git e no GitHub, uma ramificação aponta para um instantâneo no histórico do seu projeto. Não estou trazendo isso aleatoriamente. Cobrimos isso no último módulo, espero que você se lembre. De fato, tenho uma razão para dizer isso, e essa razão ficará clara em breve. De fato, já no próximo slide. No Git, podemos indicar que um certo ponto da história é importante. Muitos outros sistemas de gerenciamento de código-fonte também oferecem isso. Através da marcação, posso indicar que um certo ponto tem um certo valor a ser lembrado. Por exemplo, isso pode ser usado para indicar que um determinado commit é a versão V1. 0. No Git, temos duas opções para criar tags. Uma tag leve é ​​a primeira opção, e basicamente oferece nada mais que um ponteiro para uma confirmação. A segunda opção é criar uma tag anotada. Essa tag é, como o nome indica, mais completa, pois contém informações como o email, a data e, possivelmente, também uma tag m essage. Em geral, essas são as tags que você deseja criar na maioria das vezes. As tags também podem ser adicionadas mais tarde, o que significa que, se você já possui outros commits após o que você planejou marcar, a adição de uma tag é feita usando o comando git tag, logicamente. A adição do parâmetro -a transforma isso em uma tag anotada, na qual você pode adicionar outras informações necessárias. Vamos ver como podemos criar essas tags localmente na próxima demonstração. Como espero que você já tenha chegado, as tags são um conceito bastante simples. Embora possamos criar tags localmente e enviá-las para o GitHub, no GitHub, temos algo a mais. O GitHub nos fornece tags, mas também lançamentos. Depois de enviar nossas tags para o GitHub, podemos explorá-las através do site. Com uma tag, podemos fazer o download do código correspondente vinculado a essa tag. Lembre-se, nada mais é do que um commit. Por meio do GitHub, podemos fazer o download do código associado a essa tag como um arquivo zip. Por meio do GitHub, também podemos ver a lista de confirmações que levam a essa confirmação. Além disso, mais algumas informações sobre as tags, como o método, estão disponíveis no GitHub. Demonstração: Trabalhando com Tags Agora que você já possui um conhecimento básico de tags, vamos para a primeira demonstração deste módulo. Começaremos com a criação de tags localmente e enviaremos isso para o GitHub a seguir. Depois de colocá-los no GitHub, veremos a página de tags no GitHub. Podemos trabalhar com tags localmente e no GitHub, e começaremos com tags locais. Agora, para trabalhar com tags, precisamos realmente ver o que está acontecendo em nosso repositório. Observe que atualmente estou de volta ao mestre e vamos fazer rapidamente um status git para verificar qual é o status atual. Bem, não há nada a confirmar, é um diretório de trabalho limpo. Então, vamos fazer uma limpeza para limpar a tela. Então, como mencionado, eu preciso ver o que está acontecendo no meu repositório. Eu posso usar o comando git log para isso, e o comando git log me fornece uma saída detalhada do histórico do meu repositório. O primeiro bloco está me mostrando a confirmação mais recente no meu repositório, mestre de origem, de modo que o mestre remoto, o mestre local e o HEAD de origem e o HEAD local estão todos apontando para esse commit, e esse foi o resultado da criação da solicitação pull de a ramificação add-installation no meu mestre. Antes disso, tínhamos na ramificação add-installation o guia de instalação que foi adicionado e assim por diante. Portanto, todos os commits estão sendo mostrados aqui. Eu posso sair disso digitando q para voltar ao prompt de comando. Agora vamos ver como podemos criar tags em um ou mais desses commits para basicamente rotulá-las. Então, vamos ver qual tag temos atualmente, podemos usar o comando git tag para isso e nenhuma saída, o que significa que atualmente não há tags ativas. Na verdade, posso criar muito simplesmente uma tag leve. Quando eu faço o git tag e digito, por exemplo, como tag stable e faço isso no meu ramo principal, automaticamente a confirmação mais recente no meu ramo principal tem o rótulo estável. Se agora eu faço a tag git, devemos ver que atualmente temos uma tag estável, e isso foi fácil, porque eu marquei a última confirmação no meu ramo mestre. É como você viu que a saída era muito detalhada, por isso é muito difícil obter uma visão geral do histórico completo do meu repositório. Posso usar novamente o log git, mas adicionar alguns parâmetros extras. Uso o log do git e passo uma linha, o gráfico decora tudo, e isso cria uma boa visão geral do histórico no meu repositório. Lá vamos nós, isso parece muito melhor, vemos os commits iniciais, os arquivos README, tudo o que fizemos anteriormente. E também vemos que temos ramo do mestre. Vemos que criamos o ramo de amostra, que não foi mesclado, mas também criamos o ramo de instalação e adição na origem, que foi mesclado novamente no ramo mestre. Vamos agora criar uma tag anotada. Agora, uma tag anotada não usa um comando diferente, ainda é uma tag git, mas especificando -a e, em seguida, o nome da tag, estou criando uma tag anotada. Portanto, suponha que nosso software esteja na versão 0. 1. Também posso incluir outra mensagem, que só é possível em tags anotadas, e também menciono que posso especificar em qual confirmação quero aplicar essa tag. Então, digamos que queremos fazer isso nos arquivos adicionados ao site. Esse foi o primeiro lançamento, digamos, do nosso aplicativo. Então isso foi confirmar a6b446e. Verifique novamente, isso parece estar correto. Tudo certo. Pressione Enter e lá vamos nós. Se agora fizermos outra tag git para obter uma visão geral de todas as tags, estável foi o commit atual e a V0. 1 agora está apontando para o commit do arquivo do site adicionado. Vamos fazer outro. Digamos, por exemplo, que temos a versão 0. 2, 0. 20, que veio após a mesclagem do ramo mestre na colher difusa. Pressione Enter, faça outra tag git e agora temos três tags carregadas. Neste ponto, é claro, as tags são apenas locais, ainda não fizemos nenhum push no GitHub. Agora, se quisermos enviar as tags para o GitHub, um simples empurrão do git não será suficiente, precisamos especificar que também queremos enviar as tags para o GitHub. Então, o que vamos fazer para isso é especificar o parâmetro --tags aqui. Como você pode ver, nossas tags foram enviadas com êxito para o GitHub. Vamos para o GitHub. Não fizemos nenhum novo commit, mas o que mudou aqui é que, em lançamentos, ele agora diz que há três lançamentos. Vamos dar uma olhada. Aqui na página, vemos Versões e Tags. Agora, em essência, não há muita diferença entre tags e lançamentos. Voltarei aos lançamentos no próximo tópico, mas vamos clicar em Tags aqui. Aqui vemos uma visão geral de todas as tags que criamos. Para cada tag, posso ver a qual confirmação está associada, na verdade, posso clicar nessa confirmação e ver o que mudou nessa confirmação. Posso clicar no rótulo aqui e pularemos automaticamente aqui também para esse commit específico. Também interessante é que o GitHub me permite fazer o download do código-fonte associado a esse commit específico, como zip e tar. arquivo gz. Também está dizendo que, para nossa ramificação principal, dois commits, dois novos commits, digamos, foram feitos desde essa tag. E uma tag é apenas um ponteiro para um commit, então basicamente desde esse commit, dois novos commit foram feitos. A maior diferença entre tags anotadas e tags leves é claro que aqui, quando clicamos aqui nas reticências, vemos a mensagem, enquanto na tag leve, portanto, a primeira que criamos, simplesmente vemos a mensagem de confirmação do confirmar esta marca está associada. Também podemos excluir uma tag. Vamos clicar em uma das tags aqui e dizer Excluir. O GitHub está nos avisando que essa tag agora será excluída. Vamos clicar em Excluir esta tag. E agora temos apenas duas tags restantes. Agora, uma coisa importante a lembrar é que, quando estou trabalhando localmente com tags, essas tags não serão excluídas automaticamente, mesmo sendo puxadas, você esperaria que as tags também fossem excluídas, para que as coisas voltem a ser sincronizadas. Vamos tentar. Já atualizado, diz aqui. Portanto, se agora eu fizer uma tag git, você ainda verá as três tags que criamos localmente. Na verdade, posso excluir localmente uma tag. Então eu posso dizer git tag -d, para delete, e então o nome da tag, para que fique estável. E agora, a tag é excluída. Nós só temos a v0. 1 e 0. 2 restantes. Agora poderíamos fazer o contrário também. Imagine que eu vou excluir o 0. 2. Então diga, git tag -d, e então diga v0. 2. A tag é excluída localmente. Agora posso novamente pressionar, especificando a origem dessa v0. 2 deve ser empurrado. E agora, de fato, o GitHub relata que v0. 2 foi excluído. Se você voltar ao GitHub, atualizará esta página; de fato, nossa tag foi excluída do GitHub. Trabalhando para uma versão no GitHub Já vimos tags, agora voltamos nossa atenção para os lançamentos. Já é uma palavra de aviso, porém, tags e lançamentos estão intimamente relacionados. As versões são baseadas em tags e, como mencionado, ambas estão intimamente relacionadas. No entanto, com o lançamento, podemos adicionar outras informações. Provavelmente, a diferença mais notável é que, com os lançamentos, podemos adicionar notas de lançamento. Por meio da interface do GitHub, podemos criar versões no formato Markdown novamente e, na mesma página, também podemos adicionar arquivos binários relacionados a esta versão. Demo: Criando uma nova versão no GitHub Vamos agora voltar ao GitHub e explorar como podemos trabalhar com lançamentos. Criaremos uma nova versão e daremos uma olhada no resultado. Então, aqui novamente na página de lançamentos do repositório. Lançamentos são a maneira do GitHub de empacotar e fornecer software para seus usuários. Ele basicamente substitui os downloads que você usa para fornecer o software aos seus usuários. É muito parecido com as tags e nem sempre é fácil ver a diferença real. A maior diferença é que um lançamento pode ter notas de lançamento, enquanto uma tag realmente não pode ter isso. As liberações também podem ser usadas com arquivos de binários anexados. Aqui estamos na página de lançamentos e você vê que já existe um lançamento criado. Embora eu não tenha feito isso manualmente criando uma tag, o GitHub automaticamente cria uma versão também, mas criaremos uma nova versão a partir do zero. Então, para isso, usaremos o botão Draft a new release aqui. Então diga que isso seria v0. 3. Ainda não temos essa tag, então o GitHub a criará. Diga que esta é uma versão demo. Podemos novamente criar alguns comentários aqui. Isso novamente suporta o formato Markdown. Também podemos selecionar arquivos binários, então vamos fazer isso. Vou incluir um arquivo zip. Também posso indicar se isso já está pronto para produção. E vamos fazer isso agora. Porque é v0. 3, que ainda não está pronto para produção. E agora vamos publicar o lançamento. Então agora criamos uma versão com sucesso. É indicado como um pré-lançamento, a tag 0. 3. O lançamento também inclui alguns ativos, os arquivos binários são incluídos, assim como o código-fonte, novamente como um arquivo zip e o tar. arquivo gz associado ao commit. E, nesse caso, essa seria a versão mais recente no mastro er. Automaticamente, uma tag também foi criada, como você pode ver aqui. Você pode ver que essa é uma tag especial, na verdade foi lançada porque possui notas e downloads de versão. A propósito, observe que, a partir de uma tag existente, também podemos criar um release. Eu seleciono uma tag existente aqui, clico neste botão Criar versão e a converterei automaticamente em uma versão. Resumo Isso conclui nossa exploração de tags e lançamentos. Embora eles estejam intimamente relacionados, vimos como os dois têm um objetivo específico. Porém, no final, tags e releases são indicadores com informações extras para um ponto específico da história. No próximo módulo deste curso, exploraremos como o GitHub habilita a codificação social. Espero vê-lo lá também. O aspecto social do GitHub Introdução ao módulo Olá, e bem-vindo a outro módulo do curso GitHub Fundamentals, aqui no Pluralsight. Sou Gill Cleeren, e também vou orientá-lo neste módulo. Se você tiver alguma dúvida sobre este curso, sinta-se à vontade para entrar em contato via Twitter ou postar suas perguntas aqui em um quadro de discussão no site da Pluralsight. Neste módulo, falaremos sobre o aspecto social do GitHub. Já vimos que o GitHub realmente se concentra nos recursos de colaboração e já vimos alguns deles com bastante detalhes. Agora, este módulo se aprofundará e também encontrará novos itens que ainda não foram tocados. Tudo bem, vamos começar este módulo, dando uma olhada no que aprenderemos neste módulo. Iniciaremos o módulo analisando o conceito de bifurcação no GitHub, e você verá que eles estão intimamente relacionados a solicitações de recebimento novamente, que já abordamos anteriormente neste curso. Neste módulo, nós os revisitaremos novamente no contexto de repositórios bifurcados. A seguir, vamos dar uma olhada nas principais informações. Embora esse seja um recurso pequeno, definitivamente vale a pena dar uma olhada. O próximo tópico é Páginas do GitHub. Neste curso, faremos uma breve visão geral das páginas do GitHub. Não vou cobrir todos os detalhes. Isso, de fato, poderia ser um curso em si. No tópico final deste módulo, veremos outro recurso criado com um aspecto social, que é o de notificações. Veremos que tipo de notificação podemos ativar e como queremos recebê-las. Uma agenda completa, como você pode ver. Eu acho que é uma boa idéia começar imediatamente. Trabalhando com Forks Como dito, iniciaremos este módulo examinando o conceito de garfos no GitHub. Embora esse seja realmente um recurso do GitHub, é claro que também precisaremos executar algumas tarefas usando as ferramentas do lado do cliente novamente. Vamos dar uma olhada nos garfos. A primeira é que devemos nos perguntar, o que são garfos realmente? Para simplificar, um fork é uma cópia de um repositório. Sim, na verdade nada mais é do que uma cópia de um repositório existente. Podemos fazer alterações nessa cópia sem afetar o repositório original. Agora, é claro, sendo o GitHub, é mais do que apenas uma cópia simples. Quando criamos uma bifurcação, a cópia permanece basicamente vinculada ao repositório original. Portanto, se houver alguma atualização no repositório original, também podemos obter essas alterações. Também o oposto é verdadeiro. Se criamos uma bifurcação, é possível a partir dessa bifurcação criar uma solicitação de recebimento, para que nossas alterações possam, opcionalmente, ser mescladas novamente no repositório original. Bifurcação é possível para qualquer um. Depois de criar o fork, esse fork agora é seu próprio repositório e você pode gerenciá-lo completamente. Isso significa que, no novo projeto, você pode trazer colaboradores, alterar as configurações de recompra e assim por diante. Os garfos geralmente são criados para testar idéias e, sim, os garfos estão definitivamente relacionados à maneira de trabalhar com grandes ramificações. Forks também são uma coisa do GitHub. Eles não vêm com o próprio Git. É por isso que falo sobre eles aqui, pois eles realmente destacam o aspecto social da plataforma. Uma grande diferença entre garfos e ramificações nessas ramificações funciona no mesmo repositório, enquanto os garfos funcionam em repositórios diferentes. Em termos de poder criar solicitações pull para mesclar suas alterações novamente no repositório upstream, as coisas são muito parecidas aqui. Antes de voltarmos para o GitHub, vamos dar uma olhada na representação gráfica de como o bifurcação está funcionando no GitHub. Tudo começa com um repositório. Como usuário ou organização do GitHub, gosto desse repositório, mas acho que preciso fazer algumas alterações para que funcione perfeitamente para a minha situação. Uma boa opção nesse caso é, portanto, bifurcar este repositório em minha própria conta. Nesse ponto, criei uma cópia, para que essa cópia agora seja minha origem. As coisas ainda estão no GitHub também neste momento. Agora este é meu próprio repositório para que eu possa fazer o que quiser com ele. Por exemplo, posso decidir que quero fazer algumas alterações para clonar e voltar à minha máquina local. As alterações que eu fiz também serão devolvidas ao meu próprio fork. O repo original não se incomoda com tudo isso. Nenhuma das alterações que eu possa ter devolvido à origem está retornando automaticamente ao repositório upstream. Somente quando penso que seria uma boa ideia, posso enviar uma solicitação de recebimento ao repositório upstream. Obviamente, para deixar claro, a partir de um único repositório, vários garfos podem ser criados. Se alguém também gostar deste repositório, poderá executar o mesmo processo que eu originalmente. Esses garfos não estão relacionados, exceto pelo fato de que ainda sabem que são garfos do repositório original a montante. Demonstração: Trabalhando com Forks Tudo certo. Hora da primeira demonstração deste módulo. Nesta primeira demonstração, vamos fazer um repo e fazer algumas alterações localmente. Depois, enviarei essas alterações para o garfo recém-criado. Então, vamos dar uma olhada em como podemos bifurcar nosso repositório. Então agora vou ter que trabalhar com duas contas. Portanto, essa ainda é a conta original, tão branquinha, mas eu tenho outra janela do navegador, uma janela privada, e lá estou trabalhando com branquinha. Essa é a segunda conta. Eu usei esses nomes para deixar um pouco mais claro para você. Agora, usando verde branquial, vou bifurcar o repositório de colher difusa laranja-branqueada, então preciso ir ao github. com / gill-orange, e aqui é claro que vou encontrar o repositório de fuzzy-spoon. E como verde-e-branco, eu realmente gosto deste repositório, então vou criar minha própria cópia deste repositório. Como mencionado, esta é uma cópia completa e independente, mas podemos mesclar as alterações novamente no repositório upstream, se quisermos fazê-lo, mas mostrarei a seguir. Então, primeiro vamos bifurcar. Então aqui em cima temos um botão Fork. Portanto, lembre-se de que estamos trancados como verde-guelra. Vou bifurcar o repositório de colher difusa. Aqui vamos nós. Então, é bifurcar-colher branqueada / difusa. Essa é uma frase estranha que eu nunca pensei que diria, mas de qualquer maneira. Então, aqui está nosso repositório bifurcado sob a conta Gill-Green, e você também verá que ele é bifurcado no repositório original Gill-Orange / Fuzzy-Spoon. Agora, este é um repositório simples e posso fazer o que quiser com ele. Para que eu possa, por exemplo, cloná-lo na minha máquina local. E agora em outro diretório, é claro, Pluralsight / otheruser, e aqui farei o git clone e colo no endereço. Agora estamos usando o endereço HTTPS aqui. Isso é simplesmente porque, caso contrário, eu usaria a mesma chave SSH para trabalhar com duas contas, e isso realmente cria um pouco de confusão. Então, eu estou apenas usando o endereço HTTPS aqui. Então lá vamos nós. Agora temos uma cópia local da colher difusa para o outro usuário, portanto, a partir da conta verde-guelra. Como você pode ver, o controle remoto foi definido como verde-branqueado e agora podemos fazer o que quisermos com ele. Nós podemos criar mudanças locais. Podemos enviar esses arquivos de volta ao nosso repositório Git. Como eu disse, faremos isso em uma máquina com a mesma chave SSH. Isso pode ser realmente confuso, porque eu realmente vou enviar como laranja-guelra, então vou fazer algumas alterações apenas no GitHub e trabalhar a partir daí. Então, estamos aqui novamente na conta Gill-Green, e vou fazer algumas alterações no índice. html. Agora, como esse é meu próprio fork na minha própria conta verde-e-branco, posso mudar isso. É um repositório simples. Então, vamos mudar esse título aqui para o incrível site de Gill. Aqui vamos nós. Nós vamos fazer um commit local. Mudou o título. E comprometa essas mudanças. E agora fiz um commit claro no meu índice. arquivo html. Nenhuma alteração, no entanto, foi feita na versão upstream do índice. html. Esse é o repositório original e nenhuma alteração foi feita lá. Vamos voltar aos slides e continuaremos com esta demonstração. Solicitações de Pull Revisitadas Na área de garfos, também podemos trabalhar com solicitações pull. Já falamos sobre solicitações pull há alguns módulos quando estávamos olhando para as branches. Como acabei de explicar, os garfos não são realmente diferentes dos ramos e, portanto, também podemos usar solicitações pull no contexto de bifurcação. Vamos dar uma olhada nisso nesta seção. Tenho certeza de que todos se lembram do que são solicitações pull. Usando uma solicitação pull, como desenvolvedor, posso indicar que quero atualizar outras partes interessadas sobre as alterações que fiz. Como agora estamos falando de bifurcação, isso significa que quero que os proprietários do repositório upstream saibam que fiz uma alteração que, acredito, também pode ser interessante adicionar ao repositório pai. Talvez nem esteja sugerindo que devemos procurar mesclar minhas alterações. Pode ser que eu tenha essa ideia novamente que gostaria de discutir ou tenha feito melhorias que acho que funcionariam e gostaria que elas fossem revisadas. No final, como já vimos com solicitações pull, pode ser que as alterações que fiz no meu fork sejam aceitas e, portanto, mescladas no repositório upstream. Como as solicitações de recebimento são realmente uma coisa do GitHub e não do Git, é bastante normal que o GitHub ofereça uma interface muito agradável para trabalhar com solicitações de recebimento. Se você acompanha esse curso, deve observar que vimos essa interface com solicitações pull anteriores Ier. Embora agora estamos trabalhando com bifurcações de um repositório, a maneira como criamos a solicitação pull é semelhante à forma como criamos uma volta quando estávamos trabalhando com ramificações. Portanto, se você já entendeu tudo quando estávamos trabalhando com ramificações, está bem definido. Agora que estamos falando de solicitações pull novamente, desta vez quero falar um pouco mais sobre a fusão que pode ser feita. Quando o proprietário do repositório upstream decidir que as alterações do seu fork serão mescladas, precisaremos realmente executar a mesclagem. Mesclar pode ser uma coisa complicada, e daremos uma outra olhada na mesclagem na próxima demonstração. Agora, ao fazer uma mesclagem, realmente depende das alterações que você fez. Se as alterações estiverem em linhas diferentes no seu arquivo ou arquivos, é provável que o GitHub seja capaz de descobrir tudo sozinho e executar a mesclagem automaticamente para você. Ao trabalhar com a solicitação pull e a mesclagem, você verá esta interface aqui no slide. Às vezes, se você tiver uma alteração conflitante, precisará de algum trabalho manual e o GitHub informará que não é possível concluir a mesclagem automaticamente. Portanto, é necessária alguma ajuda para você, o proprietário. Os casos típicos em que isso pode ocorrer podem incluir edições diferentes na mesma linha; portanto, o GitHub não poderá selecionar qual precisa ser salvo. Além disso, se em um commit você fez alterações em um arquivo e ao mesclar esse arquivo já foi excluído em outro commit, isso também resultará em uma mesclagem que não pode ser resolvida automaticamente. Portanto, o GitHub não executará a mesclagem até que todos os conflitos tenham sido resolvidos. Depois de resolver tudo, será criada uma nova confirmação, uma confirmação de mesclagem que contém o resultado final. Como mencionado, o GitHub pode descobrir se a mesclagem pode ser feita automaticamente. Veja o que você vê se ele detectou um conflito de mesclagem e, portanto, será necessário resolver isso antes de continuar. O GitHub pode ser usado para realizar uma mesclagem, como você verá aqui. Como alternativa, você também pode optar por não usar a interface do GitHub e executar a mesclagem localmente. Você precisará usar uma ferramenta diff neste caso, que pode ser configurada na configuração do GitHub. Existem várias ferramentas e, é claro, você pode selecionar qual delas usará. As opções freqüentemente usadas incluem KDiff3, P4Merge, vimdiff3 e Beyond Compare. Outros existem também e, como eu disse, não importa qual deles você usa. A escolha é sua. Demonstração: Trabalhando com solicitações pull Ok, essa foi uma parte bastante interessante. Acho que agora é um bom momento para voltar a uma demonstração em que veremos tudo isso em ação. Começaremos de onde paramos com a demonstração anterior, portanto já criamos um fork do repositório upstream. A partir desse fork, agora criaremos uma solicitação de recebimento de volta ao repositório upstream. Como podemos fazer com uma solicitação pull, faremos uma revisão e depois mesclaremos as alterações da solicitação pull novamente no repositório upstream. Vamos dar uma olhada. Suponha que, como verde-branquinha, fiz algumas mudanças impressionantes, como fizemos antes, e quero sugerir essas alterações de volta ao repositório upstream. E eu posso fazer isso criando uma solicitação de recebimento. E, neste caso, não vou mesclar no meu repositório, vou pedir uma mesclagem entre repositórios, então, de volta ao repositório upstream. Então, vamos tentar isso. Então, eu fiz alterações neste caso no ramo mestre na demonstração anterior. Eu poderia ter feito isso em um ramo diferente, mas para manter as coisas simples aqui, usei apenas um ramo mestre. Mas vou criar uma nova solicitação de recebimento. Então, em Gill-green, vamos para solicitações Pull, e aqui vou criar uma nova solicitação pull. O que você vê agora é que o GitHub está sugerindo que eu crie uma solicitação de recebimento do mestre da colher-de-verde / fuzzy para o mestre da colher-de-laranja / fuzzy. E também vê que esses ramos podem ser mesclados automaticamente. Novamente, isso nem sempre será o caso. Se você tem projetos mais complexos, nem sempre existe a possibilidade de fazer uma mesclagem automática, mas, novamente, precisamos fazer uma mesclagem manual. Nesta página, muito parecido com o que vimos anteriormente, vemos em que consiste essa solicitação de recebimento. Por exemplo, vemos os commits que fazem parte dessa solicitação de recebimento. Vimos que há um arquivo que foi alterado e apenas um colaborador contribuiu para essas alterações na solicitação de recebimento. Vemos aqui a mudança que fizemos. Então, isso estava no índice html. Alteramos o título de Bem-vindo ao nosso site para Bem-vindo ao incrível site da Gill. Também podemos ver isso novamente em exibição dividida. Vemos as mudanças um ao lado do outro. É uma maneira melhor do que vimos anteriormente. Agora, na verdade, fizemos uma alteração em um arquivo, em um arquivo existente no site, e acabamos de adicionar um novo arquivo. Tudo certo. Então, acho que estamos prontos para criar essa solicitação de recebimento. Então, vamos começar. Crie uma solicitação de recebimento. E, como antes, adiciono um título e alguns comentários aqui. Então, aqui vamos nós, aqui está um comentário e agora podemos realmente acionar essa solicitação de recebimento. Vamos agora pular para laranja branquial. Gill-orange vê sob solicitações Pull, bem, há uma nova solicitação pull. Alguém basicamente sugeriu uma mudança de um fork para este repositório upstream. Agora eu posso clicar, é claro, em uma solicitação pull, e vemos aqui, bem, o resumo dessa solicitação pull. Vimos que a solicitação de recebimento está aberta no momento e o verde-branquial deseja mesclar um commit no mestre laranja-branqueado do mestre verde-branquial. Vemos o comentário que você criou anteriormente, as confirmações associadas a essa solicitação de recebimento e também vemos novamente uma visão geral dos arquivos que foram alterados. Se concordarmos, podemos clicar diretamente na solicitação de mesclagem pull e, em seguida, a solicitação pull será levada ao nosso mestre. Se não o fizermos, bem, também podemos deixar um comentário e deixar as coisas rolarem dessa maneira. Bem, vamos tentar isso aqui. Então, vamos adicionar um comentário. Clique aqui para comentar. Portanto, se salvarmos esta resenha como verde-guelra, e o que agora vou fazer é voltar ao meu garfo e fazer mais algumas alterações. Ainda podemos continuar editando enquanto essa solicitação de recebimento estiver aberta, e o GitHub também menciona isso aqui. Então, vamos voltar ao verde-branquial. Volte ao nosso repositório de colher difusa. E continue editando o índice. html. Vamos adicionar mais alguns pontos de exclamação e criar nosso commit. Confirme as alterações. Aqui vamos nós. Então agora vamos voltar para laranja-guelra. Sob a colher difusa, vemos as solicitações pull, e também vemos que, devido ao comentário que recebi como verde-e-branco, fiz outra atualização. Essa confirmação extra foi adicionada a essa solicitação de recebimento. Vamos mudar a conta novamente para laranja-branquinha, alternar entre janelas do navegador. Eu posso ver aqui como o proprietário do repositório upstream que mais alterações foram feitas. Eu posso ver essas mudanças. E vemos que os pontos de exclamação foram adicionados aqui. Agora estou feliz com isso, então o que vou fazer agora é mesclar a solicitação de recebimento, confirmar a mesclagem de gill-green / master em gill-orange / master. Mesclando. Aqui vamos nós. E as alterações agora foram confirmadas no meu mestre-laranja-guelra. Se você agora der uma olhada no código novamente, veremos aqui que o próximo resultado é o título final criado no repositório bifurcado. Então agora vimos como alguém pode bifurcar um repositório e enviar essas mudanças de volta usando uma solicitação pull para o repositório upstream. Usando Gists Agora que você já viu provavelmente a maneira mais social de trabalhar com código na parte anterior, vamos mudar de marcha e ver quais outros recursos o GitHub oferece para suportar sua plataforma de codificação social, e começaremos com gists. Então, o que é uma essência? Gists é um projeto paralelo do GitHub destinado a permitir que os usuários compartilhem trechos e notas. Se você já trabalhou com ferramentas como Pastebin, bem, isso está intimamente relacionado a isso. As listas estão disponíveis em um site separado, sob o domínio GitHub. Ou seja, essência. github. com. A interface como você pode ver aqui é muito simples. Ele permite que você crie uma descrição para sua essência e, em seguida, o código real. Usando gists, como mencionado, você pode compartilhar arquivos ou trechos, fazendo parte de um arquivo. Use-o para se lembrar de algo para você ou para compartilhar com seus colegas ou até mesmo para o seu blog. Embora tenha como objetivo compartilhar arquivos, você pode combinar vários arquivos em uma única essência. Gists, no entanto, não suportam diretórios. Nos bastidores, os gists usam a mesma infraestrutura do GitHub, o que significa que tudo aqui também é um repositório. Ao trabalhar com gists, existem basicamente dois tipos, públicos e secretos. Como o nome indica, dicas públicas são, bem, realmente públicas. Eles são pesquisáveis ​​para que todos possam encontrar o tema facilmente. Os secretos, por outro lado, são realmente particulares. É basicamente algo que você não deseja descobrir ou ainda não está pronto para compartilhar com todos. No entanto, o link em si não é privado. Embora as essência confidenciais ou secretas não apareçam em uma pesquisa, por exemplo, se você tiver o URL, ainda poderá acessá-lo. Portanto, definitivamente não é uma boa idéia colocar algo realmente secreto lá. Finalmente, também é possível fazer o download de listas principais na forma de um arquivo zip. Um dos lugares onde eu vejo as dicas sendo úteis é como mencionado, incorporado dentro de uma postagem do blog. Através da página gist, como você pode ver aqui, você pode obter um código estrito que permite incorporar essa essência em uma página. Demo: Usando Gists Tudo bem, vamos dar uma olhada nas dicas na próxima demonstração. Então, para trabalhar com gists, precisamos ir para gist. github. com e, como mencionado nos slides, é basicamente para trechos, bits de código, não para projetos inteiros. Então, vamos criar uma essência aqui. Algo que uso no .NET, basicamente no XAML e no Xamarin, com frequência é um método chamado ToObservableCollection, do qual não faz parte. NET, mas vai converter uma lista em uma coleção observável. Então, eu vou guardar isso em uma essência que eu possa reutilizar mais tarde ou até mesmo compartilhar no meu blog. Eu colo o código aqui. Então essa é a minha essência aqui. Eu tenho algumas opções para editar essa essência, como os espaços, o tamanho do recuo e assim por diante. Na verdade, posso adicionar vários arquivos. Portanto, é possível criar uma essência contendo vários trechos, ou seja, arquivos, e agora posso escolher se quero criar uma essência pública ou secreta. Portanto, como mencionado, uma essência pública é pesquisável e também será exibida nas últimas essências criadas. A essência secreta não está disponível publicamente. Você ainda pode procurar se tiver o link correto, mas ele não aparecerá nos resultados da pesquisa. Então, por enquanto, vamos criar uma essência pública. Portanto, em laranja branquial, agora temos uma maneira única de identificar essa essência. Aqui eu vou pegar esse URL, vou navegar diretamente para essa essência aqui. Aqui vamos nós. Neste ponto, eu tenho apenas uma versão da minha essência. Eu posso editar essa essência. E, por exemplo, adicione alguns comentários aqui, diga Atualizar lista pública, e como você pode ver agora em Revisões, teremos duas revisões dessa lista. Nesse menu suspenso aqui, posso optar por obter um link para incorporar esse código, esse snippet, ao meu blog. Como trabalhamos com um arquivo HTML o tempo todo neste curso, vamos copiá-lo para esse HTML e ver como ele é. Então aqui no bloco de notas na seção principal, vou adicionar isso aqui dentro do conteúdo. Salve isso. Visualize esse arquivo aqui. E observe aqui o resultado de incorporar minha essência no site. Agora, uma essência também é um repositório. Então, eu posso simplesmente dizer aqui Clone via SSH. Vou receber o link SSH. Copie isso aqui. Eu posso ir para minha CLI. Vá para um diretório e agora faça um clone do git. Aqui vamos nós. Agora, o que isso criou? Bem, uma pasta como qualquer outro repositório que clonamos, e lá meu arquivo gist está disponível. Há também um. pasta git lá, então este é realmente um repositório Git simples. Gists são de fato apenas repositórios Git simples. Como este é um repositório Git, agora posso dizer que quero editar meu arquivo txt localmente. Então, eu vou abrir o bloco de notas, editar esse arquivo aqui, adicionar um git como fizemos antes, criar um commit. Então fazemos um git push, master de origem e lá vamos nós. E se agora dermos uma olhada no site gist e fizermos uma atualização, veremos que há uma nova revisão que foi feita. De fato, a linha de mais comentários foi adicionada. Criando páginas do GitHub Compartilhar conhecimento é uma das melhores coisas que todos podemos fazer. Embora os repositórios no GitHub contenham muitos conhecimentos, nem sempre é a maneira mais fácil de compartilhar esse conhecimento. O GitHub oferece outras maneiras de compartilhar informações. Um deles é o GitHub Pages. Nesta seção, veremos brevemente o Pages e veremos como podemos configurá-los facilmente. E antes de começarmos a ver como podemos usar as páginas do GitHub, é uma boa ideia começar analisando o que realmente são. Basicamente, é um serviço de hospedagem de sites onde você pode hospedar qualquer site. Pode ser um site relacionado a um repositório, mas também pode ser o seu blog pessoal ou até o site da sua empresa. Um dos principais pontos de interesse aqui é que ele é feito para hospedar páginas estáticas. Você não pode executar nenhum código do lado do servidor. Todas as páginas precisarão ser geradas e depois colocadas online. Portanto, não é o lugar perfeito para colocar uma página da web com páginas dinâmicas. Debaixo das capas, seu site está sendo hospedado em um repositório. Lembre-se, tudo no GitHub gira em torno de repositórios, e as páginas, de fato, não são uma exceção a essa regra. A criação do site de páginas pode ser feita online ou offline usando a linha de comando aqui. Sim. É apenas um repositório. Demonstração: Páginas do GitHub Eu acho que é melhor dar uma olhada nas páginas do GitHub em uma demonstração, em vez de falar sobre isso. Vou dar uma demonstração aqui. Observe que há muito mais do que pode ser feito com o Pages. Isso poderia, como eu já disse, ser um curso em si. Vamos dar uma olhada. Então, vamos dar uma olhada em como posso habilitar as páginas do GitHub para o meu repositório. Então, a primeira coisa que precisamos fazer é ir para Configurações, e lá role para baixo e você encontrará as páginas do GitHub. Portanto, como mencionado, podemos hospedar arquivos estáticos de dentro de um repositório. A primeira coisa que precisamos fazer é basicamente habilitá-lo e dizer ao GitHub qual ramificação queremos usar nas páginas do GitHub. Portanto, como na verdade, sob o domínio principal, colocamos o site, tornarei mais fácil para mim e selecionarei o ramo principal. E salve. Agora, neste momento, o GitHub Pages já está configurado, para que eu possa pegar esse link, abrir em uma nova guia e ver o que acontece. Na verdade, temos o github laranja-branquial. io / fuzzy-spoon, para que o nome do nosso repositório tenhamos um site hospedado estaticamente que contenha simplesmente o código no meu repositório. Se eu fizer alterações, bem, então vemos essas alterações diretamente. Então, vamos ao índice da colher difusa, altere o índice. html, clique em Editar aqui e altere-o novamente para o nosso site. Faça outro commit e, se agora eu voltar ao site, atualizaremos esta página, realmente veremos imediatamente nossas alterações que fizemos em nosso repositório, pois, efetivamente, nosso repositório é a fonte desse site. Ativando notificações Na seção final deste módulo sobre os aspectos sociais do GitHub, veremos como trabalhar com notificações. O GitHub possui vários tipos de notificações integradas, que podem ser usadas para ajudar seus stakeholders a se atualizarem sobre status e conversas no projeto em que estão interessados. As notificações são incorporadas ao GitHub, digamos, em geral. O que quero dizer com isso é que existem vários lugares no GitHub onde você pode se registrar para receber notificações. Todas as notificações direcionadas a você chegarão através de dois canais possíveis. Você pode optar por ser notificado sobre determinados eventos que ocorrem através do GitHub. Basicamente, essas notificações podem ser de dois tipos. As notificações participantes são enviadas a você quando você está diretamente envolvido. Pode ser que você faça parte de uma conversa, por exemplo, em uma revisão de uma solicitação pull. Você também receberá uma notificação participante se abrir ou fechar uma solicitação pull. Vamos dar uma olhada em algumas outras opções na demonstração. Em seguida, assistimos a notificações. Como o nome indica, eles serão enviados quando algo mudar no repositório que você está assistindo. Por exemplo, quando uma solicitação de recebimento é aberta nesse repositório, você recebe uma notificação. Quando um problema é criado, bem, a mesma coisa. Como dito, as notificações chegarão até você através de diferentes canais. As notificações podem chegar no GitHub, para que você as veja diretamente na interface da web. No entanto, também é possível se registrar para uma maneira mais intrusiva de ser notificado, e isso é por email. Se você optar por receber notificações da Web, como dito, suas notificações serão entregues diretamente na interface do GitHub, como você pode ver aqui. Demonstração: Trabalhando com notificações Na última demonstração deste módulo, veremos as notificações. Mais especificamente, veremos como podemos nos registrar para receber notificações e como elas são entregues a nós. Permaneceremos no controle total de como você recebe as notificações e veremos isso também. Para alterar as configurações das notificações, é claro que precisamos ir para as configurações da conta. Nesse caso, estou conectado como verde-e-branco e, em Configurações, temos a capacidade de trabalhar com as configurações de notificação. Nesta página, podemos escolher como queremos receber notificações. Conforme mencionado nos slides, as notificações podem participar ou assistir a notificações. Participar é onde você realmente é mencionado ou em que está participando. Assistir é apenas assistir a um repositório que você realmente será notificado. Agora, como eu assisto um repositório? Esta é uma boa pergunta. Então, por exemplo, deixe-me ir ao github. com / laranja-guelra. Então, aqui estou eu no repositório gill-orange / fuzzy-spoon, e vou começar a assistir esse repositório. Selecionarei Assistindo aqui e agora seremos notificados sobre toda a conversa. Então, muitas coisas realmente acionam notificações. Quando você vai ao github. com / assistindo, você realmente terá uma visão geral de todos os repositórios que está assistindo. Atualmente, estou apenas assistindo aspnet / Mvc, mas se eu atualizar isso, bem, vemos que agora também está sendo assistido por fuzzy-spoon da minha conta aqui. Como obtenho agora alterações aqui? Bem, indiquei aqui que quero ser notificado na Web e por e-mail para assistir notificações. Também para as notificações participantes, eu indiquei que quero ser notificado pela Web e por email. Vamos agora voltar para a conta laranja-branquial e criar um novo problema aqui. Vamos chamá-lo de problema de amostra ... Exemplo de comentário e envie esse novo problema. Se você agora voltar ao github. com / assistindo, e mudamos para a guia Notificações. De fato, vemos nossa notificação originada na conta laranja-branquial. Podemos clicar nesse exemplo de exemplo aqui e navegaremos automaticamente para a página do problema. Observe aqui na parte inferior também que estou de fato inscrito para receber notificações porque estou assistindo este repositório. Resumo Esse foi um módulo bastante longo e espero que interessante. O método principal aqui foi que o GitHub estimula a codificação social, e espero que este módulo tenha fornecido uma boa visão geral de como isso ocorre. Vamos recapitular o que vimos aqui. Provavelmente, um dos maiores facilitadores é a bifurcação combinada com solicitações de recebimento. O último, desencadeia discussão e revisão das mudanças que propomos. As páginas e listas de discussão do GitHub ajudam a compartilhar conhecimentos que, de outra forma, estariam profundamente ocultos em um repositório no GitHub. Por fim, analisamos as notificações e como elas podem ser usadas para envolver todos os participantes nos repositórios nos quais você está interessado. No próximo módulo, mudaremos de marcha novamente e veremos como o GitHub pode ser usado. gerenciar um projeto inteiro. Espero vê-lo lá novamente. Gerenciando seu projeto através do GitHub Introdução ao módulo Olá novamente e bem-vindo ao módulo chamado Gerenciando seu projeto através do GitHub, parte do curso de Fundamentos do GitHub, aqui no Pl visão central. Sou Gill Cleeren e vou guiá-lo através deste módulo. Aposto que vai ser muito divertido. Lembre-se, no módulo anterior, vimos os aspectos sociais do GitHub. Neste módulo, veremos como uma equipe pode usar o GitHub para não apenas gerenciar o código-fonte, mas usá-lo como sua ferramenta para gerenciar todo o projeto. O que iremos abordar neste módulo? Bem, eu tenho quatro itens para você que eu acho que estão intimamente relacionados a como você pode usar o GitHub para gerenciar seu projeto. Começaremos com o que eu acho que é o mais importante, questões. Você verá que os problemas são a maneira do GitHub de gerenciar itens de trabalho. Em seguida, relacionado a problemas, veremos os marcos. Depois disso, e isso também é importante, vamos dar uma olhada nos projetos e nos painéis de projetos. Com isso, obteremos uma representação visual do trabalho que ainda precisa ser feito. Finalmente, ao trabalhar em qualquer projeto, deve haver um local onde todas as informações e documentação do projeto sejam armazenadas. Isso podemos fazer usando um wiki. A criação de wikis está embutida no GitHub, e aprenderemos neste módulo como podemos trabalhar com eles. Espero que isso pareça um módulo interessante também. Vamos começar imediatamente. Trabalhando com problemas Os problemas são fundamentais para o GitHub, permitindo-nos gerenciar nosso trabalho por meio da interface. Começaremos analisando os tipos de problemas que o GitHub define. Os problemas da palavra podem induzi-lo a pensar que eles são negativos. Ei, há um problema com seu código, mas esse não é o caso. No GitHub, problemas é apenas um nome para indicar tudo o que está acontecendo, desde sim, de fato, erros, de modo negativo, a tarefas de trabalho, aprimoramentos e até idéias. Todos esses itens de feedback estão basicamente conectados em um rastreador grande, chamado de problemas. Cada repositório possui um rastreador de problemas. Como eu disse, os problemas são mais do que apenas erros. Novos recursos que precisam ser criados podem ser treinados pela criação de um problema. Veremos na próxima seção, quando examinarmos os quadros do projeto, que os problemas podem ser usados ​​para priorizar o trabalho. Os problemas também podem ser vinculados a uma solicitação pull. Talvez possamos mesclar a solicitação de recebimento e fechar o problema correspondente. Fechar o problema também não significa que as informações se foram. Todas as informações relacionadas permanecem disponíveis no GitHub. Quando usamos problemas, também podemos ser notificados usando o sistema de notificação que analisamos no módulo anterior. Imagine que você criou um problema. Quando alguém criar um comentário sobre esse problema, você será notificado. Isso ajuda você a manter-se atualizado sobre o estado do problema. Falaremos sobre o quadro do projeto mais adiante neste curso. Eu já mencionei brevemente. O quadro do projeto pode ser usado para criar um planejamento ou obter uma visão geral visual do trabalho no projeto. Nesse quadro do projeto, também podemos usar problemas e, por exemplo, atribuir um problema a outra pessoa da equipe. Portanto, você pode atribuir problemas a outra pessoa. Veremos como isso é feito muito em breve. Solicitações pull também podem ser associadas a problemas. Em uma solicitação de recebimento, podemos mencionar um número de problema combinado com uma palavra-chave. Isso criará a associação. Os problemas também podem ser usados ​​para trabalhar com marcos, que veremos no próximo tópico neste módulo. Por enquanto, você pode ver um marco como um ponto que deseja alcançar. Esse ponto é alcançado quando você conclui vários problemas, daí a relação de problemas e marcos. Por fim, também é possível criar problemas a partir do código em um arquivo ou em uma solicitação pull. Como os problemas são um recurso do GitHub, o site possui toda a infraestrutura necessária para trabalhar com eles. Provavelmente a tela mais relevante nessa área é, obviamente, o próprio rastreador de problemas. O rastreador de problemas está vinculado a um repo. Na tela principal do rastreador de problemas, você pode ver uma lista de todos os problemas em aberto por padrão, mas é claro, também podemos configurá-lo para exibir outras listas. Também a partir da interface, somos capazes de criar um problema. Aqui nesta captura de tela, você pode ver isso, e veremos isso ao vivo na próxima demo. Deixe-me explicar mais uma coisa antes de irmos para a próxima demonstração, e isso é rótulos. Os rótulos podem ser aplicados em problemas, mas também em solicitações de recebimento. Eles geralmente são usados ​​para organizar e também priorizar itens como problemas. Novamente, os rótulos são uma coisa do GitHub. Muito lógico, se você pensar sobre isso, eles são adicionados a problemas e solicitações pull e também são coisas específicas do GitHub. Por padrão, o GitHub já vem com um conjunto de rótulos embutidos, mas se esses itens não forem suficientes para você, você pode criar seus próprios e até excluir os incorporados. Por padrão, oito rótulos são criados no GitHub e, portanto, estão disponíveis em cada repositório para você usar imediatamente. Você pode ver essa lista de marcadores. Suponha, por exemplo, que você esteja criando um problema que descreve um bug. Bem, então você pode usar o rótulo de bug para categorizar esse problema. Quando você e outras pessoas que trabalham no repositório aplicam rótulos de maneira consistente, também facilita a digestão da visão geral na lista de problemas. Demonstração: Trabalhando com problemas Ok, é hora de uma grande demonstração sobre como trabalhar com problemas. Vamos dar uma olhada no rastreador de problemas relacionado a um repositório e criaremos um problema e veremos como um problema pode ser vinculado a uma solicitação pull. Por fim, veremos também como trabalhar com etiquetas. Vamos lá. Então, primeiro, primeiro, talvez você não veja a guia Problemas do seu repositório. Estou vendo aqui, como você pode ver, mas se você não vê, vá para Configurações do repositório e, aqui, acho que você pode ativar ou desativar o trabalho com problemas em seu repositório. Tudo bem, vamos clicar em Problemas. Atualmente, não tenho problemas em aberto, mas começarei a criar alguns em apenas um minuto, mas, para ter uma idéia melhor de como os problemas se parecem com um projeto da vida real, vamos usar o aspnet / Mvc novamente e aqui você verá uma lista de 184 questões em aberto, a maioria também bem rotulada, e falaremos sobre rótulos em apenas um minuto. Então, de volta à nossa página de repositório, vamos primeiro dar uma olhada no que podemos fazer com esta lista de problemas. Portanto, por padrão, o filtro é definido para os itens do tipo emissão e que estão abertos. Também podemos mudar isso, por exemplo, para que apenas desejemos ver as coisas atribuídas a mim. E, novamente, não vemos nada neste momento. Vamos agora criar um novo problema, e já fizemos isso brevemente antes para examinar as notificações, mas agora eu vou lhe mostrar todos os detalhes. Então, criamos um novo problema, novamente podemos usar o Markdown aqui. Por exemplo, se queremos deixar algo ousado, podemos cercá-lo usando dois asteriscos. E antes de enviarmos esse problema, vamos dar uma olhada no que mais podemos fazer com ele. Na verdade, podemos trabalhar com os responsáveis, mas isso não funciona porque não temos outras pessoas na equipe. Voltaremos a isso mais adiante neste curso. Vamos dar uma olhada nos rótulos. Na verdade, posso atribuir um rótulo a esse problema, por exemplo, especifiquei que esse é um erro, não está funcionando corretamente e também posso selecionar vários rótulos. Os rótulos são basicamente usados ​​para categorizar meus problemas. Então agora eu apliquei dois rótulos. Também posso vincular esse problema a um projeto, mas também não examinamos os projetos e também os mestres. Mas lembre-se de que, ao criar um problema, temos várias coisas extras que podemos adicionar a ele. Então, agora vamos enviar o problema. Lá vamos nós, agora o problema foi enviado e esses dois rótulos foram aplicados. Se voltarmos ao rastreador de problemas, realmente veremos nosso problema marcado com bug e aprimoramento. Abrimos o problema novamente, é claro que podemos continuar editando este ou podemos adicionar outro comentário a esse problema. Se você acha que esse problema deve ser encerrado, podemos encerrá-lo diretamente aqui ou adicionar este comentário extra. E assim, um comentário para um agora, lá vamos nós. Vou adicionar outro comentário. Suponha que eu saiba que esse problema foi corrigido em alguns commit. Posso dizer Fixado, e agora digitamos a chave de hash, e aqui podemos associar isso a outro problema. Você não vê apenas problemas aqui, mas também solicitações pull aqui. Isso explica por que o número do problema é 6 aqui e não apenas 1 ou 2, porque eu tinha apenas alguns problemas, mas observei que as solicitações pull também são mostradas aqui. Então, eu vou associar isso a essa solicitação pull específica. E agora eu clico sobre isso, na verdade vou navegar para essa solicitação de recebimento específica. Vamos voltar ao código por apenas um minuto. Também é possível, quando você está olhando, por exemplo, o código, criar um problema diretamente a partir do código. Vamos dar uma olhada. Se, por exemplo novamente neste título aqui, dissermos que queremos criar um novo problema, clicamos na linha, na verdade clicamos no número à frente da linha e depois dizemos Abrir novo problema. Aqui vamos nós. Portanto, esse problema é associado automaticamente a essa linha específica. Podemos atribuir um rótulo a ele, é um bug e enviamos o novo problema. se dermos uma olhada nesse problema agora, automaticamente a linha de violação também será mostrada aqui dentro do problema. Agora, falando sobre rótulos, clique aqui novamente na lista de marcadores, role até o fim e clique em Editar marcadores. Por padrão, todo repositório no GitHub vem com uma lista de rótulos. E você pode começar a editar esses rótulos ou criar novos. Por exemplo, vamos criar um novo rótulo, bloqueando, por exemplo, Problema de bloqueio. Podemos mudar a cor deste, que parece estar bem. E então criar etiqueta; aqui vamos nós. Agora nós adicionamos um novo rótulo. Também podemos, por exemplo, remover um rótulo existente. Se não queremos o rótulo da pergunta, por exemplo, podemos excluí-lo. E lembre-se, isso se aplica apenas a este repositório específico. Agora volte ao nosso problema recém-criado e clique aqui na engrenagem novamente e selecione o bloqueio aqui também. Criando um marco Agora você examinou bem os problemas e, para o próximo tópico, permaneceremos na mesma área quando analisarmos os marcos. Claro que a primeira coisa é garantir que você saiba o que um marco é realmente. Se você pensa no conceito de um marco, na vida real, é um ponto que você alcança, certo? Bem, no GitHub, um marco é praticamente a mesma coisa. Ele nos permite rastrear se chegamos a um determinado ponto. Normalmente, isso é feito usando uma lista de problemas dos quais acompanhamos o progresso. Além dos problemas, os marcos também podem ser usados ​​em um grupo de solicitações pull. Marcos são, novamente, uma coisa do GitHub. Então, trabalhar com eles é novamente feito para a interface. Um marco geralmente inclui itens como uma data de vencimento, uma porcentagem de conclusão e uma lista e número de problemas em aberto e solicitações de recebimento. Demo: Criando um Marco Em vez de falar sobre marcos, vamos voltar ao GitHub e dar uma olhada em como podemos criar e gerenciar marcos. Então, eu estou aqui na página de problemas do meu repositório e, a partir daqui, posso criar um novo marco. Então, clicamos aqui no botão Marcos na parte superior e depois navegamos para todos os marcos criados e, como podemos ver, ainda não há nenhum para o nosso projeto. Então, vamos verificar novamente. Vamos criar um novo marco, dar um nome a ele, vamos chamá-lo de exemplo de marco e vamos assumir que queremos definir uma data de vencimento. Portanto, os marcos podem realmente ser direcionados para uma determinada data de vencimento, para que as pessoas se organizem para concluir esse trabalho em uma determinada data. Então, vamos definir isso para, bem, vamos defini-lo para 20 de setembro, que é daqui a duas semanas. Adicionamos uma descrição e depois criamos o marco. Se você olhar para o marco agora, realmente não há muito. Você tem o nome e uma descrição, ele também tem sua data de vencimento e, aqui à direita, podemos ver quanto percentual está concluído, quantos problemas no marco ainda estão abertos e quantos foram fechados, mas não estamos ' ainda não tem nada, então vamos mudar isso. Assim, podemos voltar à lista de problemas novamente e, a partir daqui, podemos realmente associar problemas ao nosso marco. Para que eu possa atribuir esse problema a esse marco e voltar às minhas Listas, selecionar os outros problemas e associá-los também ao meu marco. Na lista de problemas, você pode ver que esses problemas já foram associados a esse marco. Vamos agora voltar aos marcos e clicar no nosso marco, e aqui vemos as questões em aberto nesse marco. A partir daqui, posso selecionar um problema e encerrá-lo. E voltamos à página dos marcos agora, veremos que 33% dos problemas nesse marco foram concluídos. Dois ainda estão abertos, um está fechado. É assim que basicamente podemos trabalhar com marcos. Trabalhando com Projetos Já abordamos as partes mais importantes da maneira como trabalhamos com problemas no GitHub, e também vimos como eles se relacionam com os marcos. Neste tópico, veremos os projetos no GitHub. Como este módulo foi intitulado Gerenciando seu projeto através do GitHub, é claro que esses projetos desempenham um papel crucial na maneira como você fará isso com o GitHub. Quando alguém fala sobre o gerenciamento de seu projeto no GitHub, provavelmente está se referindo a um quadro de projetos. Como o nome indica, um quadro de projetos pode ser usado para fornecer uma visão geral do status do trabalho do projeto. A principal idéia por trás do quadro é obviamente fornecer uma visão geral para ajudar na organização e priorização do trabalho do projeto. Para o repositório fornecido, você pode criar uma única placa, mas é perfeitamente possível criar várias placas. Por exemplo, você pode optar por criar um quadro específico para gerenciar o trabalho e os problemas relacionados a um determinado recurso. Um quadro também pode ser usado para gerenciar uma lista de verificação ou mesmo um roteiro. O principal benefício é, na minha opinião, que você tenha uma visão geral do trabalho em várias colunas. Quando alguém fala sobre o gerenciamento de seu projeto no GitHub, provavelmente está se referindo aos quadros do projeto. Como o nome indica, um quadro de projetos pode ser usado para fornecer uma visão geral do status do seu trabalho. A principal idéia por trás do quadro é obviamente fornecer uma visão geral para ajudar na organização e priorização do trabalho do projeto. A moeda no GitHub para gerenciar o trabalho são problemas. Como o conselho do projeto trata de gerenciar o trabalho em um projeto, é bastante lógico que no conselho do projeto normalmente listemos problemas e recebamos solicitações. No quadro do projeto, esses itens são mostrados como cartões e arrastamos os cartões entre as diferentes colunas do quadro. Esses cartões, como veremos na demonstração em apenas um minuto, também contêm outros dados relacionados, como quem está atribuído ou os rótulos atribuídos. Na captura de tela deste slide, você pode ver uma amostra do quadro do projeto. No quadro, podemos ver os cartões e os cartões estão em colunas diferentes, e tenho certeza de que, em seus projetos do dia a dia, você já trabalhou com quadros semelhantes para gerenciar o trabalho em seu projeto. O GitHub vem com alguns modelos criados, que você pode selecionar para configurar seu novo quadro. Com base no modelo selecionado, a placa configurada vem com algumas colunas e algumas placas base para você, por padrão. Os modelos disponíveis incluem kanban básico, kanban automatizado, kanban automatizado com revisão e triagem de erros. E veremos isso na demonstração. Demonstração: Trabalhando com Projetos Falando sobre a demo, vamos fazer isso agora. Então, nesta demonstração, eu crio um quadro de projetos. Também usaremos alguns dos problemas que já criamos e os gerenciaremos em nossos quadros. Vamos dar uma olhada. Portanto, usando um quadro ou um quadro de projeto, podemos organizar e priorizar o trabalho. Normalmente, é o local onde os problemas e solicitações pull serão reunidos. Normalmente, ele contém problemas e solicitações de recebimento, e eles são representados como cartões em várias colunas. Nesses cartões, também poderemos adicionar metadados extras. Quando queremos criar um projeto ou um quadro de projetos, precisamos ir para a guia Projetos aqui. Novamente, se não for possível, vá para as configurações do repositório e ative-o. Observe também que os projetos podem ser adicionados apenas por um proprietário do projeto. Quando estamos na página de projetos sem nenhum projeto criado, podemos clicar no botão Criar um projeto. Aqui podemos criar nosso novo quadro, e vamos chamá-lo de quadro de amostra. E forneceremos uma descrição de amostra. Agora, na parte inferior da página, podemos selecionar o modelo pré-configurado para o nosso quadro de projetos. Podemos selecionar Nenhum e, basicamente, temos um tabuleiro vazio sem colunas. Outros modelos já vêm com algumas colunas internas. Mas, vamos pegar o vazio por enquanto, para que possamos criar nosso próprio quadro. Então, vamos criar o projeto. Então, como você vê aqui, neste ponto, este projeto não possui colunas nem cartões. Portanto, ainda não podemos fazer nada com isso. Então, vamos mudar isso. Vou clicar em Adicionar uma coluna aqui e inserir, como nome da coluna, Não iniciado. Isso normalmente contém os problemas e itens que ainda não foram iniciados. Mas já poderemos ver o próximo trabalho. Em segundo lugar, também podemos selecionar automação. Agora a automação também é realmente uma predefinição que pode ser usada para garantir que novos problemas ocorram em uma determinada coluna com base em seu estado. E veremos isso em apenas um minuto. Por enquanto, vamos selecionar o modelo de automação de tarefas aqui e, em seguida, posso selecionar que, quando novos problemas forem criados ou novas solicitações de recebimento forem criadas, elas serão automaticamente adicionadas a esta coluna. Tudo bem, vamos criar a coluna. Ainda não há nada lá. Faremos isso em apenas um minuto. Vamos criar outra coluna, vamos chamar isso de coluna Em andamento e criar a coluna. E em terceiro lugar, adicionaremos uma coluna para itens Concluídos. Aqui vamos nós. Portanto, ainda não temos cartões. Então, voltemos aos nossos problemas. E ainda temos alguns problemas disponíveis aqui, vamos abrir um deles e associar esse problema agora a uma diretoria do projeto. Nós temos apenas uma placa, então, nós a consideraremos neste caso. E vamos fazer o mesmo para os outros problemas. Vamos agora voltar ao nosso quadro, e agora, devido à automação que eu configurei na coluna Não iniciado, todos os problemas recém-criados aparecerão automaticamente nesta coluna. O quadro é realmente útil de usar. Posso clicar no problema aqui e obter uma prévia do problema. Também posso acessar todos os detalhes da questão. Também posso, nesta coluna, criar, digamos, uma nova nota que possa ser atualizada para outro item. Vamos dar uma olhada, vamos criar uma nota de amostra aqui. Clique em Adicionar e agora isso pode ser convertido em um problema. Portanto, esta nota agora se tornou um problema. Quando estivermos trabalhando em um problema, podemos arrastá-lo para a coluna Em andamento. E assim que terminar, podemos buscá-lo novamente e movê-lo para a coluna Concluído. Aqui também podemos, como qualquer outra coisa no GitHub, começar a filtrar as cartas em nosso quadro. Por exemplo, podemos filtrar apenas aqueles que possuem um rótulo e, em seguida, apenas aquele que possui o rótulo do bug ainda está visível. Também é possível, no menu, fechar o projeto. Se você fizer isso, bem, a automação também irá parar. Depois de fechá-lo, você poderá reabri-lo posteriormente, se necessário. Configurando um Wiki No GitHub, temos a capacidade de adicionar um wiki ao nosso projeto. Na parte final deste módulo, veremos como podemos criar um desses wikis. Um projeto sem boa documentação é, acho que é um projeto regular? Embora geralmente seja a parte menos interessante, uma boa documentação sobre coisas como diretrizes de codificação ou práticas recomendadas é valiosa para um projeto. O GitHub vem com a capacidade integrada de adicionar um wiki a um repositório, oferecendo uma localização central para gerenciar todas as informações sobre um projeto. Os wikis estão vinculados no GitHub a um repositório, descrevendo assim o projeto. O objetivo, é claro, como já mencionado, é fornecer aos desenvolvedores e outros membros da equipe um local central para a documentação do projeto. Ele pode conter informações para os usuários finais do seu projeto, como como usar o código, por que certas coisas funcionam da maneira que foram criadas e muito mais. Você ca n escolha, é claro, o que você coloca lá. O wiki pode conter muitas páginas e pode ser editado online diretamente no GitHub, como veremos na demonstração ou offline, e as alterações podem ser enviadas como sempre. Nessa visão, é diferente do arquivo README, que também é uma forma de documentação para o projeto. O README normalmente contém uma visão geral ou uma introdução, enquanto o wiki pode ser usado para fornecer informações mais completas sobre o projeto. Obviamente, podemos fazer referência ao wiki de dentro do read me, que é claro também funciona. A propósito, antes que eu esqueça, a maneira como você realiza a marcação e o layout do conteúdo no wiki é novamente o que já vimos. Isso pode ser feito usando a sintaxe do Markdown, que veremos novamente na demonstração. Demo: Configurando um Wiki Na demonstração final deste módulo, voltarei ao nosso repositório e adicionaremos um wiki para a documentação do projeto. Adicionaremos uma página e outro conteúdo, incluindo texto e imagens. Vamos dar uma olhada. Na demonstração final deste módulo, vamos dar uma olhada no wiki. E, é claro, por padrão, ainda não existe um wiki. Então, criaremos um novo, e um wiki é basicamente uma coleção de páginas, então criaremos nossa primeira página. Nesta página, adicionaremos algum conteúdo básico, esta é a página inicial, portanto, normalmente ela contém uma introdução ao projeto. As páginas wiki, é claro, também suportam totalmente o Markdown. Então aqui está o nosso conteúdo. E clique em Salvar página. Então agora temos a nossa primeira, digamos, página de destino, criada. Agora o GitHub nos dá algumas dicas. Deveríamos adicionar uma barra lateral e talvez também um rodapé. Portanto, neste rodapé, vamos definir a última data editada. Vamos salvar a página. Vamos agora criar outra página e depois voltaremos para a barra lateral. Vamos chamar esta página Introdução, coloque outro texto de lorem ipsum lá, salve a página e agora vamos criar a barra lateral personalizada, e aqui vou criar alguns links para nossas páginas. Vamos criar Página inicial e Introdução e vamos transformar essas coisas em links. Então, selecionarei isso e queremos que o texto seja Inicial, e também queremos que ele vá para a página inicial. O que você vê agora é que a página inicial é basicamente convertida para o formato Markdown. Faremos o mesmo na página Introdução. Então, definiremos o texto aqui como Introdução. E assim, o URL normalmente começará. Vamos salvar isso e, como esse é o Markdown, posso fazer isso em uma lista adicionando asteriscos na frente desses itens. Salvar pagina. E agora temos a barra lateral e os links estão realmente funcionando. Isso parece estar bem. Vamos voltar para a página inicial e vamos continuar editando esta página um pouco mais. Por exemplo, talvez eu queira incluir uma imagem aqui. Então, vou colar no URL da imagem, definir o valor alt, clicar em OK e automaticamente isso também será convertido em Markdown. Salve essa página e agora vemos o logotipo do Pluralsight, é meio grande, mas funciona. O que você também vê aqui à direita é que o GitHub está mencionando que podemos clonar este wiki localmente. Vamos fazer isso. Portanto, estamos obtendo um diretório do Pluralsight e, agora, agora eu posso criar um clone do git, colar a URL que o GitHub nos forneceu para o wiki e, como você pode ver, existe uma colher difusa. wiki, que contém todos os arquivos MD ou arquivos Markdown que criamos dentro do GitHub. Podemos continuar editando esses itens e enviá-los de volta ao GitHub. Então, de fato, um wiki também é apenas, novamente, um repositório contendo arquivos Markdown. Resumo Chegamos ao final de mais um módulo. Espero que você tenha gostado do que viu neste. Vamos resumir rapidamente o que aprendemos neste módulo para finalizar. Espero que você já tenha entendido que o GitHub oferece uma grande variedade de ferramentas que o ajudam a gerenciar seu projeto. Tudo gira em torno do rastreador de problemas, que é um ponto central para armazenar todas as informações na forma de problemas, incluindo bugs, idéias e solicitações de recursos. O conceito de problemas percorre todo o caminho do GitHub, e é também o que está sendo usado dentro do quadro de projetos, que também vimos neste módulo. Finalmente, com nosso repositório, também podemos vincular um wiki, que é usado para armazenar a documentação sobre seu repositório junto com ele. No próximo módulo, exploraremos como o conceito de organização e equipe é usado no GitHub. Obrigado por assistir este módulo. Trabalhando com organizações e equipes Introdução ao módulo Olá, e bem-vindo a este módulo, parte do curso de Fundamentos do GitHub, aqui no Pluralsight. Sou Gill Cleeren e também o guiaremos por este módulo. No módulo anterior, falamos sobre como você pode usar o GitHub para ajudar no gerenciamento do desenvolvimento do seu projeto, e vimos coisas como o quadro do projeto, o que poderia ajudar muito com isso. Agora, neste módulo, veremos outro recurso que o ajudará na maneira como você e sua equipe podem trabalhar com o GitHub. Para ser mais preciso, falarei sobre o conceito de organizações e equipes no GitHub. A agenda para este módulo é bastante compacta. Começaremos explorando o que é uma organização no GitHub e como podemos criar e gerenciar a organização. Em segundo lugar, desenvolveremos isso quando examinarmos o conceito de equipes no GitHub também. Novamente, esses são os dois recursos específicos do GitHub, então você poderá trabalhar com isso diretamente na interface do GitHub. Criando uma organização Então, primeiro, nossa exploração do conceito de organização no GitHub. Vamos ver o que é uma organização e como podemos criar uma. E quando falo de uma organização em geral, a primeira coisa que provavelmente vem à sua mente é uma empresa. Embora as organizações possam definitivamente ser usadas para isso, as organizações do GitHub são mais amplas do que apenas serem úteis para uma empresa. O GitHub define uma organização em geral como uma conta compartilhada que pode ser usada para trabalhar e colaborar em um ou mais projetos. Portanto, voltando ao que acabei de dizer, pode ser que sua empresa crie uma organização no GitHub e atribua seus membros a essa organização. Definitivamente, esse é um bom caso de uso, mas você também pode criar uma organização para reunir vários desenvolvedores para trabalharem juntos em um projeto, e veremos isso em ação na próxima demonstração. As organizações são de propriedade do proprietário, e os proprietários e administradores podem gerenciar os membros e os projetos e também, em um nível mais detalhado, gerenciar o acesso. Uma coisa que definitivamente vale a pena mencionar aqui é que as organizações podem ser usadas usando uma conta gratuita ou paga. Se você usar uma conta da organização da mesma forma que uma conta pessoal, poderá, usando essa conta, criar um número ilimitado de repositórios públicos e poderá gerenciar dentro dessa organização gratuita e quantidade ilimitada de colaboradores. Se decidirmos criar uma organização, podemos criar uma a partir do zero. O GitHub oferece uma interface onde podemos criar uma organização, e eu mostrarei isso na próxima demonstração. Além de criar uma do zero, também é possível converter uma conta existente, portanto uma conta individual em uma organização. Isso fornecerá opções mais detalhadas para gerenciar quem na organização obtém acesso a quais repositórios. Uma vez convertida, porém, a conta pessoal basicamente deixa de existir. Todos os repositórios serão convertidos. De dentro de uma organização, novos membros podem ser adicionados. Através do GitHub, você pode simplesmente adicionar novos membros e eles receberão um convite para fazer parte da organização. Mais uma coisa. Dentro de uma organização, existem apenas algumas funções, digamos, internas, o proprietário e o membro. A função de proprietário é usada para gerenciar a organização. Um gerenciamento adicional de permissões na conta precisa ser feito usando equipes, que exploraremos como um segundo tópico neste módulo. Demo: Criando uma Organização Tudo bem, é hora de outra demonstração. Nesta demonstração, mostrarei como podemos configurar uma organização do zero e como podemos adicionar usuários a essa organização. Então, vamos criar uma organização agora. Podemos criar uma organização de maneiras diferentes. Podemos converter uma conta regular em uma conta da organização ou criar uma do zero. Tudo o que tem a ver com a configuração da organização deve ser feito através das Configurações. Aqui em baixo, você tem Organizações, e aqui temos as duas opções que acabei de mencionar. Podemos transformar a conta existente em uma organização ou criar uma do zero. Usaremos a segunda opção aqui. Então, clicamos em Nova organização. Então agora eu vou ter que criar uma conta da organização. Portanto, para o nome da organização, vou especificar SnowballDevTeam. Vou inserir meu endereço de e-mail para a cobrança. Meu senso é que, novamente, uma conta regular, podemos escolher entre uma conta gratuita ou uma paga, e também podemos indicar se isso será administrado por uma empresa. Para nós, esse não é o caso, então deixarei isso desmarcado e clicamos em Criar organização. Em seguida, podemos convidar usuários diretamente daqui ou também podemos fazer isso mais tarde. Já podemos, como demonstração, pesquisar aqui por verde-e-branco. Como você pode ver, ele encontra esse usuário, mas vamos pular isso por enquanto, adicionaremos usuários mais tarde. Então, clique em Concluir aqui para que a organização esteja configurada. Chegamos agora à página de destino dessa organização. Vamos dar uma olhada no que você vê aqui. Ainda não há repositórios. Portanto, de fato, a organização pode ter seus próprios repositórios. Lembre-se, é basicamente apenas uma conta à qual minha conta Gill-Orange está vinculada. Atualmente, só há laranja branquinha lá. Falaremos sobre equipes mais tarde e, como uma conta simples, elas podem ter projetos e configurações. Então, vamos criar um novo repositório. Eu meio que gosto do nome que está sugerindo aqui, então chamaremos o nosso projeto de festa especial. Vai ser público novamente, le adiciona um arquivo README e a. arquivo gitignore, adicione também uma licença e clique em Criar repositório. Observe que agora é um repositório na conta SnowballDevTeam, feito por gill-orange. Portanto, minha conta faz parte da organização SnowballDevTeam. Agora vamos criar novos repositórios diretamente na conta da organização, mas também podemos transferir repositórios existentes da minha conta individual para a minha conta da organização. Então aqui temos a conta Gill-Orange, e enquanto você não estava assistindo, eu criei este novo repositório, silver-engine. Esse é um repositório de propriedade da conta Gill-Orange, e agora vamos transferi-lo para minha organização. Então, vamos para Configurações e, na Zona de Perigo, transferiremos a propriedade deste repositório. Vou transferir esse repositório. Primeiro, posso dar o nome, de modo que era um mecanismo de prata, e a organização era SnowballDevTeam. Portanto, para este repositório, basicamente transferirei a propriedade para a organização. Preciso digitar minha senha novamente para confirmar. Então agora, no topo, você vê uma mensagem de que está transferindo o motor prateado, e isso pode levar alguns minutos. Então, esperamos, voltarei quando tiver sido transferido. E lá vamos nós, agora podemos ver que o motor prateado agora faz parte do SnowballDevTeam. A propriedade do repositório foi transferida. Ok, então agora vamos voltar para a organização. Posso fazer isso, como mencionei anteriormente, usando este menu suspenso aqui que só apareceu quando a conta laranja-branquial também se tornou parte da organização. Nesta página, podemos clicar em Exibir organização e aqui podemos convidar novos usuários para nossa organização. Atualmente, existe apenas uma, que é a conta laranja branquial, mas vamos convidar verde branquial. Há verde-branquinho, vamos convidá-lo. E eu escolho qual papel na organização esse novo usuário terá. Se o tornarmos dono, ele estará basicamente no mesmo nível da conta laranja-guelra, então vamos fazer isso por enquanto. Também trarei outro membro, a conta gillcleerendemo. Vou fazer dele um membro. Aqui vamos nós. Atualmente, ainda vejo apenas laranja-branquial porque há dois convites pendentes, o verde-branqueado e o verde-branqueado. E na guia do navegador, vamos para outra janela do navegador em que estamos conectados com verde-e-branco e colamos no link do convite. Então vamos ao github. com / SnowballDevTeam, e lá vemos que o verde-guelra foi realmente convidado pela laranja-guelra. Veja o convite, entrei para a equipe e, portanto, agora tenho acesso aos repositórios, parte do SnowballDevTeam. Se você der uma olhada nesses repositórios, como você pode ver, temos permissões totais para editar esses repositórios diretamente. Portanto, somos proprietários desses repositórios porque não fazemos parte da organização. Então agora, todos os meus membros aceitaram, então Gill Cleeren Green, gill-orange e Gill Cleeren, a conta demo. Observe que gillcleerendemo na verdade é apenas um membro, enquanto Gill Cleeren Green e gill-orange são os proprietários desta organização. Isso resulta em menos permissões para a conta gillcleerendemo. Nesta janela, agora estou logado com a conta demo. E como você pode ver aqui, não posso gerenciar os membros. A única coisa que posso basicamente fazer com esta conta é participar do trabalho nos repositórios. Não posso, por exemplo, alterar as configurações desta equipe. As únicas funções que atualmente tenho na organização são proprietário e membros. Teremos que olhar para as equipes para obter acesso mais refinado ao gerenciamento de quem pode fazer o que em nossos repositórios. Finalmente, vamos dar uma olhada em Configurações. Existem algumas configurações muito específicas para trabalhar com organizações. Podemos alterar o perfil, alterar a imagem, alterar domínios e verificar domínios, associar um domínio para que a conta seja verificada no GitHub. Também podemos dar uma olhada no log de auditoria, que mostra uma visão geral de todas as ações executadas na organização. Existem muitas outras configurações, mas sugiro que você as veja no seu próprio ritmo. Trabalhando com equipes Depois que criarmos nossa organização, provavelmente desejaremos gerenciar as pessoas da organização e atribuí-las a funções. Isso é possível no GitHub usando o conceito de equipes. Vamos explorar as equipes como o próximo tópico neste módulo. Uma equipe é definida pelo GitHub como uma maneira de criar grupos de membros que refletem a estrutura da sua empresa ou grupo. Usando equipes, podemos gerenciar permissões, e o sistema vem com uma estrutura em cascata para facilitar. Portanto, é realmente possível aninhar equipes. Se você tem uma organização grande para gerenciar, isso definitivamente facilitará as coisas. Os membros podem fazer parte de uma ou mais equipes. O proprietário da organização pode, por meio do sistema, gerenciar quais equipes têm acesso a quais repositórios. Outros recursos interessantes também estão habilitados para trabalhar com equipes. Por exemplo, é possível mencionar o nome de uma equipe, adicionar um comentário a uma solicitação de recebimento e todos os membros dessa equipe receberão uma notificação. Quão legal é isso? Novamente, você vê como o sistema de notificações é executado completamente no GitHub e é acessível em vários lugares. As equipes também têm uma página inicial para a equipe, a página da equipe. Aqui você vê um exemplo de uma dessas páginas de equipe no GitHub. Ele contém a foto da equipe, que é o avatar dessa equipe em particular, e também listará os membros dessa equipe. O GitHub oferece uma maneira refinada de gerenciar equipes. Através da interface do GitHub, podemos adicionar ou remover membros em uma equipe. As equipes têm seu próprio perfil e as equipes podem ser aninhadas dentro de uma hierarquia. Demonstração: Configurando equipes no GitHub Vamos voltar ao GitHub para outra demonstração e, basicamente, continuaremos de onde paramos na anterior. Vou mostrar como podemos trabalhar com equipes. Criaremos uma equipe e gerenciaremos as configurações dessa equipe. Também veremos como podemos gerenciar as permissões para a equipe. Portanto, neste ponto, só temos pessoas diretamente na organização, mas ainda não as organizamos em equipes. Então, vamos fazer isso agora. Como você pode ver, temos a guia Equipes aqui no topo e, atualmente, ainda não temos equipes. Então, criaremos uma nova equipe. Imagine que temos um grupo de desenvolvedores, e alguns deles são internos e têm acesso a todos os repositórios. Acho que também temos alguns desenvolvedores externos que só terão acesso a determinados repositórios. Vamos tentar configurar isso. Então, eu vou montar uma equipe de desenvolvedores. Vamos configurá-lo como equipe Base, e tornaremos a visibilidade da equipe. Essa equipe será a equipe base, portanto, não teremos uma equipe pai para esta e também definiremos a visibilidade como visível, o que significa que essa equipe pode ser mencionada na organização. Então agora, essa equipe também tem sua própria página de destino, digamos. E vamos aos membros. Atualmente, apenas a conta laranja-branquial faz parte dessa equipe. Então, vamos adicionar outro membro aqui e vamos procurar agora por verde-e-branco. Convide-o para fazer parte dessa equipe também. Aqui vamos nós. E agora vamos para Repositórios e adicione um repositório a essa equipe. Vamos procurar o repositório de festas especiais e adicioná-lo a essa equipe. A partir dessa configuração da equipe, podemos alterar qual nível de permissão os usuários dessa equipe terão. Portanto, o verde e o laranja-laranja terão acesso de leitura no momento, mas mudaremos isso para acesso de gravação. Vamos adicionar outra equipe. Então, voltamos ao SnowballDevTeam e criaremos mais uma equipe. E essa equipe será realmente a equipe de desenvolvedores internos, para que eles tenham acesso a repositórios específicos aos quais nem todos os desenvolvedores da minha organização têm acesso. Então, vamos mudar isso para InternalDevelopers. E agora vamos selecionar uma equipe-mãe e configurá-la para desenvolvedores, o que significa que tudo o que a equipe de desenvolvedores já pode fazer, essa equipe também poderá fazer. Como você pode ver, como agora essa equipe está sob a equipe de desenvolvedores, ela herdará automaticamente uma permissão para o repositório do fiesta especial e a permissão é Write. Então, confirme as alterações. Aqui vamos nós. Então agora temos os InternalDevelopers que têm acesso a um repositório, a festa especial. Vamos adicionar outro repositório, que era o repositório silver-engine, adicionar também a essa equipe. Portanto, essa equipe agora tem acesso a duas permissões, enquanto que, se você olhar para a equipe de desenvolvedores, elas terão acesso apenas a um repositório. Dessa forma, usando equipes, obtemos acesso refinado a quem obtém acesso a qual repositório. Resumo Chegamos ao final deste módulo. Só resta um após este. Neste módulo, analisamos o conceito de organização no GitHub, que aprendemos que pode ser usado para gerenciar um grupo de pessoas trabalhando em um ou mais projetos. Usando o conceito de equipes dentro de uma organização, conseguimos acesso refinado para gerenciar os membros e suas permissões dentro da organização. Como dito, resta apenas um módulo. No último módulo deste curso, veremos outra maneira de trabalhar com o GitHub, nomeadamente através do cliente GitHub Desktop. Interagindo com o GitHub por meio do GitHub Desktop Client Introdução ao módulo Olá de novo. Tenho más notícias para você. Infelizmente, este já é o último módulo do curso de Fundamentos do GitHub, aqui no Pluralsight. Eu acho que é verdade o que eles dizem, o tempo realmente voa quando você está se divertindo. Espero que você já tenha aprendido muito neste curso. E para este último módulo, adicionaremos um pouco mais. Meu nome ainda é Gill Cleeren. Não estou pensando em mudar isso tão cedo. Neste módulo final, guiá-lo-ei um pouco, principalmente o cliente GitHub Desktop. Mostrarei como você pode usá-lo na sua máquina local para interagir com o GitHub. Se você ainda não é fã das ferramentas de linha de comando, bem, essa ferramenta pode ser a solução perfeita para você. Neste módulo bastante conciso, primeiro abordaremos Veja o que exatamente é a área de trabalho do GitHub e o que você pode fazer com ela. Em seguida, examinaremos a instalação e os requisitos da ferramenta. Finalmente, através do uso de algumas demos, mostrarei o uso da ferramenta. O que é o GitHub Desktop? O GitHub Desktop é uma ferramenta desenvolvida pelas pessoas por trás do GitHub, permitindo que você execute algumas, não todas, interações com o GitHub por meio de um cliente de desktop. GitHub Desktop é um aplicativo Electron. A maioria dos desenvolvedores que usam o Git continuará usando a linha de comando para trabalhar com o GitHub, e isso é perfeitamente aceitável. Este cliente não deseja substituir a linha de comando. Longe disso. Você não precisa usá-lo. Mas, na minha opinião, o principal objetivo da ferramenta é reduzir a curva de aprendizado para pessoas que não têm muita experiência em trabalhar com o Git. Além disso, de acordo com os criadores, você pode usar o cliente de desktop em combinação com seu fluxo de trabalho atual. Se você está ou não aberto para usar uma GUI para interagir com o GitHub, em última análise, depende de você. Não vou dizer que você deveria ou não deveria usá-lo. A área de trabalho do GitHub não é a única ferramenta do lado do cliente que permite interagir com o GitHub. Outras ferramentas conhecidas, que oferecem funcionalidade semelhante, incluem Sourcetree, GitKraken e as Extensões Git. E sim, existem alguns outros disponíveis também. Instalação e Requisitos Se você deseja começar a usar o GitHub Desktop, vamos ver o que você precisa fazer. Para iniciantes, observe que o GitHub Desktop está disponível apenas para Windows e Mac. Não há versão do Linux no momento da criação deste curso. Pelo menos oficialmente. No entanto, um cliente Linux foi criado pela comunidade. A ferramenta está disponível para download gratuitamente a partir do URL que você vê aqui no slide. Depois de instalar a ferramenta, você poderá entrar com sua conta do GitHub. Pode ser que, no GitHub, você tenha ativado a autenticação de dois fatores. Nesse caso, também é necessário esse modo de autenticação com a ferramenta. Demonstração: Configurando a área de trabalho do GitHub Nesta demonstração, deixe-me mostrar rapidamente as etapas necessárias para instalar a ferramenta em sua máquina. Para começar a trabalhar com o GitHub Desktop, vá para a área de trabalho. github. com, e você poderá baixar a versão para o seu sistema operacional. Depois de instalar o GitHub Desktop, vá para Arquivo, Opções e lá você pode entrar. Vou entrar com a conta laranja-branquinha que usamos ao longo deste curso. Através das Opções, algumas outras opções também estão disponíveis aqui. Posso definir meu nome de usuário e meu email para trabalhar com o Git. Eu posso mudar a aparência da ferramenta. Eu posso mudar o tema. E no Advanced, tenho outras configurações que posso alterar, qual editor eu quero usar ou qual shell eu quero usar. Trabalhando com o GitHub Desktop Depois de instalar o GitHub Desktop, é hora de começar a brincar com ele. Nesta seção, mostrarei como podemos usar a ferramenta. O GitHub Desktop suporta várias interações que você pode fazer com o GitHub. Mas, novamente, para ficar claro, nem tudo é possível com a ferramenta. É por isso que eu já disse anteriormente que não é um substituto para o site nem para a linha de comando. Você pode, no entanto, trabalhar com repositórios. Você pode conectar-se a um repositório hospedado no GitHub e cloná-lo na sua máquina local. Outros comandos do Git, como buscar, puxar, também podem ser feitos usando a ferramenta. O GitHub Desktop também tem suporte para ramificações, portanto é capaz de mostrar e criar ramificações em seus repositórios. Demo: Trabalhando com Repositórios É claro que é mais útil dar uma olhada na ferramenta e ver o que podemos fazer com ela, e você fará exatamente isso em algumas demos. Primeiro, adicionaremos um repositório local ao GitHub Desktop. Também nesta demonstração, clonaremos um repo do GitHub e também faremos algumas alterações e enviaremos isso para o GitHub. Usando o GitHub Desktop, posso criar um repositório local. Vamos fazer isso primeiro. Então, eu vou no Novo repositório e vou selecionar um nome. Não sou tão original quanto o GitHub está na criação de seus nomes de repositório, então usarei o repositório demo e vou colocar isso no diretório D: \ Code \ Pluralsight. Também recebo as mesmas opções que tenho no site do GitHub, para que eu possa inicializá-lo com um README. arquivo md. Posso adicionar um gitignore, por exemplo, novamente ao Visual Studio, e também posso selecionar um arquivo de licença para ser adicionado automaticamente. Então, vamos criar esse repositório. Portanto, agora estamos no repositório demo, no ramo principal e podemos enviar isso para o GitHub. Mas faremos isso em apenas um minuto. Se você olhar para o Histórico, atualmente há um commit inicial criado quando eu criei este repositório local, no qual adicionei alguns arquivos. Se eu for para o Explorer, posso ver os arquivos, começar a editar esses arquivos ou colar novamente o meu pequeno site já existente. Se agora volto às alterações aqui, o GitHub Desktop notou que eu adicionei esses arquivos. Agora, podemos criar uma nova confirmação, os arquivos do site foram adicionados, fornecer uma descrição e, em seguida, eu posso confirmar isso no meu ramo principal. Se agora eu olhar no histórico, verei que temos dois commits, o commit inicial e o segundo, os arquivos do site adicionados. Agora estamos prontos para enviar isso ao GitHub. Então, eu vou fazer isso também no cliente GitHub Desktop. Então aqui temos o repositório Publish. Agora podemos escolher entre o GitHub. com ou empresa. Estamos usando o GitHub gratuito. com um. Podemos selecionar o nome a ser usado no GitHub. Não vou tornar isso privado, porque então terei que ter uma conta paga e não vou colocar isso na organização. Como você pode ver, as organizações também são selecionadas automaticamente pela área de trabalho. Vai publicar isso na minha conta pessoal. E lá vamos nós, publicamos este repositório inteiramente por meio do GitHub Desktop. Se eu voltar ao GitHub. com, você verá aqui que o repositório gill-orange / demo foi criado há alguns minutos, contendo os arquivos que acabei de adicionar ao cliente. Se eu quiser clonar meu repositório de colher difusa usando a área de trabalho do GitHub, tenho duas opções. Ou eu vou aqui, dentro da área de trabalho do GitHub, e vou para o repositório Clone, no menu Arquivo, para que eu possa clonar meu repositório diretamente do GitHub. com. Como alternativa, se eu tiver o URL, que posso obter no próprio GitHub em Clone e copiá-lo aqui, também posso trazer esse URL aqui. Eu deixei o GitHub Desktop cloná-lo diretamente na minha área de trabalho. Portanto, o GitHub Desktop também pode ser usado com outros controles remotos além do GitHub. Vamos clonar isso. E aqui temos o histórico completo de nosso repositório, agora também visível no GitHub Desktop. Demonstração: Trabalhando com ramificações Nesta próxima demonstração, usaremos o GitHub Desktop para criar ramificações e daremos uma olhada no histórico das ramificações. Na área de trabalho do GitHub, também podemos trabalhar com a branch. Então, eu posso ir aqui e criar um novo ramo. Aqui eu posso dar o nome do ramo. Vamos chamar esse ramo de teste de área de trabalho. Ele será ramificado do master, que é um ramo padrão, como vimos anteriormente no GitHub do meu repositório. Então, agora estamos no teste da área de trabalho, o que significa que todas as alterações que estou fazendo agora estão na ramificação do teste da área de trabalho. Vamos fazer algumas mudanças. Vamos adicioná-lo ao arquivo README. Salve isso. E se eu olhar novamente para o GitHub Desktop, veremos o README. MD foi alterado. Também posso criar um novo arquivo, é claro. Portanto, adquira um novo arquivo, vamos salvá-lo e vemos que duas alterações foram feitas. Portanto, este ramo agora é conhecido apenas localmente, portanto, teremos que publicá-lo no GitHub. Portanto, com o ramo publicado no GitHub, agora posso criar meu commit local. Vamos chamá-lo de itens de amostra adicionados e vamos confirmar isso. Então agora fizemos um commit local no ramo atual, que é o teste da área de trabalho. Vemos de fato que a dele é agora uma nova entrada em nossa história. Também na área de trabalho do GitHub, agora podemos enviar essas alterações de volta ao GitHub. Demonstração: Trabalhando com solicitações pull Na última demonstração deste curso, faremos algo que você fez algumas vezes durante este curso, e isso está criando uma solicitação de recebimento. Iniciarei isso no GitHub Desktop. Também podemos acionar o início de uma solicitação pull no GitHub Desktop, mas tudo é tratado no GitHub. com ele mesmo. Portanto, se você deseja fazer uma solicitação pull da ramificação atual para a ramificação mestre, pode clicar aqui e clicar em Solicitações Pull, e então vemos aqui que podemos criar uma solicitação pull. Como mencionado, agora o GitHub será aberto para iniciar a solicitação de recebimento. Nesse caso, criei uma solicitação de recebimento que não pode ser mesclada automaticamente; portanto, precisarei fazer mais trabalhos manuais nesse caso. O próprio Desktop GitHub não o ajudará com a solicitação de recebimento real. Tudo isso é feito no próprio GitHub. Resumo e Encerramento do Curso Chegamos ao final do módulo final deste curso. Vamos recapitular o que aprendemos. Demos uma olhada no GitHub Desktop. Como você provavelmente entendeu usando o Desktop, algumas interações se tornam possíveis através do cliente, em vez de usar as ferramentas de linha de comando. Se você não deseja usar a linha de comando e também não gosta deste cliente, outras opções também estão disponíveis. E com isso, também chegamos ao final deste curso no Git e GitHub. Espero que você tenha aprendido muito e agora esteja confiante no uso do GitHub para todos os seus esforços de desenvolvimento. Se você tiver dúvidas, poste-as no quadro de discussão aqui no Pluralsight para que todos possam se beneficiar delas. Mais uma vez obrigado por assistir, e espero vê-lo novamente em um dos meus outros cursos aqui no Pluralsight.