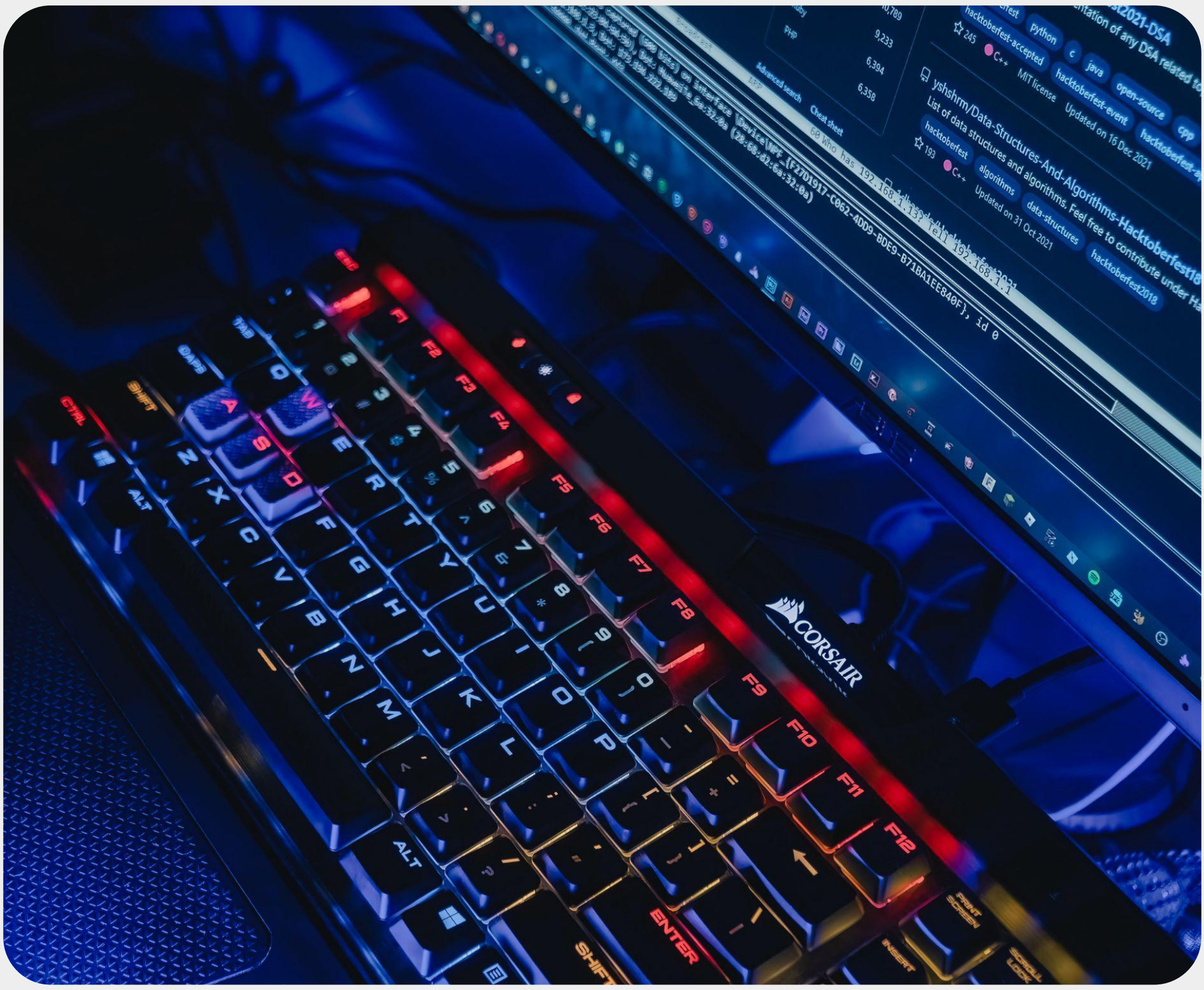
GAMA ACADEMY



Manual de **Sobrevivência do Git**

Material de Apoio

E-mail: contato@gama.academy

Website: gama.academy

Manual de Sobrevivência do Git

1ª Edição

Autoria: Gama Academy

Editora: Lauren da Silveira Francke

Consultor de Acessibilidade: Eduardo Zangrossi Lino

Consultora de Design: Juliana Moraes

Especialista Técnica: Hendy Almeida

Sumário

Clique nos links para ir ao capítulo correspondente:

Abertura

Acessibilidade é para todos!

É com essa frase que começamos uma nova fase na Gama Academy, pensando em um formato mais acessível para nossas apostilas, tornando assim sua jornada conosco muito mais prazerosa e seu aprendizado mais dinâmico.

O que você verá de diferente:

- A começar pelo novo visual dos nossos materiais;
- Trocamos o formato para o vertical, para facilitar a leitura e tornar muito mais fácil e dinâmico para quem usa o celular para estudar;
- O contraste de cores e tamanho das fontes seguem recomendações internacionais de acessibilidade da WCAG;
- Os textos passarão a ser melhor distribuídos para a redução da carga cognitiva necessária para entendê-los, assim passaremos a respeitar ainda mais o tempo de aprendizado de cada um, facilitando para que você possa retornar para o conteúdo que tenha gerado dúvidas com maior facilidade;
- Legendas em todas as imagens do conteúdo para facilitar a identificação;
- Todas as imagens que são necessárias para a devida contextualização sobre o tema, passarão a contar com audiodescrição. É um texto oculto que será lido apenas para pessoas que utilizam tecnologias assistivas, como por exemplo, leitores de tela.

Contamos muito com o feedback de vocês para tornarmos isso um processo de melhoria contínua e nossos materiais atingirem cada vez mais pessoas.

Objetivos de Aprendizagem

A partir do estudo desta apostila, você deverá ser capaz de:

- Analisar as diferenças entre GIT e GITHUB.
- Refletir sobre a importância das ferramentas GIT e GITHUB para o desenvolvedor de software.
- Fazer download e configurar a ferramenta.
- Identificar os principais comandos.

Esta apostila está dividida em tópicos e em cada um deles você irá encontrar uma pequena introdução sobre o tema e uma recapitulação do que você aprendeu.

Tópico 01

Diferenças entre GIT e GITHUB

Diferenças entre GIT e GITHUB

Olá,
Boas vindas ao módulo de GIT, neste material complementar vamos refletir acerca das principais dúvidas sobre GIT e GITHUB, além de trazer muito material bacana para aprofundar ainda mais o seu conhecimento.

Tudo pronto?
Vamos lá!

Diferenças entre GIT e GITHUB

O que é Git?

O sistema de controle de versionamento distribuído mais utilizado por desenvolvedores, seja em equipe ou sozinho, essa ferramenta é essencial para manter nossos projetos bem organizados.

Git e Github é a mesma coisa?

Não! Essa é uma dúvida muito comum quando estamos iniciando os estudos de programação.

Git é um software de versionamento, o programa que instalamos no nosso computador para gerenciar alterações nos arquivos do projeto localmente (no nosso computador).

Github é uma plataforma de hospedagem de código fonte e arquivos com controle de versionamento. Esta plataforma nos traz a possibilidade de compartilhar nosso projeto com a equipe e fazer um gerenciamento descentralizado. Você pode escolher criar um repositório público, onde qualquer pessoa pode ter acesso, ou privado, dando acesso a usuários específicos.

Diferenças entre GIT e GITHUB

Por que o versionamento de código é tão importante?

Se você já teve oportunidade de programar algo, por menor que seja, deve ter percebido que o desenvolvimento acontece em várias etapas. É um processo contínuo, onde vários desenvolvedores criam atualizações no código base. Mesmo depois de o projeto ser lançado é comum a atualização de versões, correção de bugs, adição de novas ferramentas, etc.

O sistema de controle de versão ajuda a acompanhar as mudanças feitas no código base. E mais, ele também registra quem efetuou a mudança e permite a restauração do código removido ou modificado.

Resumo do tópico

Neste tópico você aprendeu que:

1. Github é uma plataforma de hospedagem e Git é um software de versionamento.

Tópico 02

Como criar uma conta no GITHUB

Como criar uma conta no GITHUB

Neste tópico vamos aprender como criar uma conta seguindo o passo a passo do site GitHub.

Pronto?

Vamos lá!

Como criar uma conta no GITHUB

O GitHub permite você criar uma conta gratuita, para isso, basta acessar o site e seguir as instruções na tela para criar uma conta de usuário.

A conta pessoa serve como sua identidade, ou em uma organização, o que permite que várias contas pessoais colaborem em múltiplos projetos.

Atenção: Para utilizar o GITHUB você obrigatoriamente precisa criar uma conta. Depois é só seguir o passo a passo do site.

Resumo do tópico

Neste tópico você aprendeu que:

1. A conta do GitHub é gratuita e basta você seguir o passo a passo disponibilizado no site para criar o seu perfil e começar a utilizar o serviço.

Tópico 03

Como utilizar o GIT

Como utilizar o GIT

Como vimos anteriormente, GitHub e Git são diferentes. Então, agora vamos aprender sobre o Git , o software de versionamento que instalamos no nosso computador.

Como utilizar o GIT

Agora que você já sabe como utilizar o GitHub, vamos falar um pouco sobre o GIT. Lembre-se que são ferramentas diferentes, hein?!

Git é o sistema de controle de versão open source mais usado no mundo atualmente! Ele é usado para controlar o histórico de alterações de arquivos e principalmente de projetos de desenvolvimento de software. Ele permite mais flexibilidade no fluxo de trabalho, segurança e desempenho.

Antes de continuar com a leitura recomendamos que assista a vídeo aula do professor Isidro.

Download

Para iniciar os trabalhos no GIT você precisa fazer o download do software, baixe para o seu computador. Lembre-se de baixar a versão correta conforme seu sistema operacional.

Resumo do tópico

Neste tópico você aprendeu sobre:

1. O que é Git e como baixar o software do site oficial, além de aprofundar os estudos com a vídeo aula do professor Isidro.

Tópico 04

Comandos de SETUP do Usuário e Máquina

Introdução: Comandos do Usuário e Máquina

Agora que você já tem sua conta GitHub e já instalou o software GIT, nós vamos sincronizar os dois. Confira as próximas instruções!

Comandos de SETUP do Usuário e Máquina

Para auxiliar sua operação do dia-a-dia, você precisará “identificar” sua máquina junto ao Github e ter uma chave RSA para que isso seja efetivo. Isso você deverá fazer apenas uma vez e só terá que fazer novamente se mudar de máquina.

Configurando seus dados locais:

Definindo seu nome de usuário:

```
git config --global user.name "Seu nome cadastrado no Github"
git config --global user.email "Seu email do Github"
```

Criando sua chave:

Usando um client SSH (se você tem Windows, um programa bem legal pra fazer isso é o PUTTY).

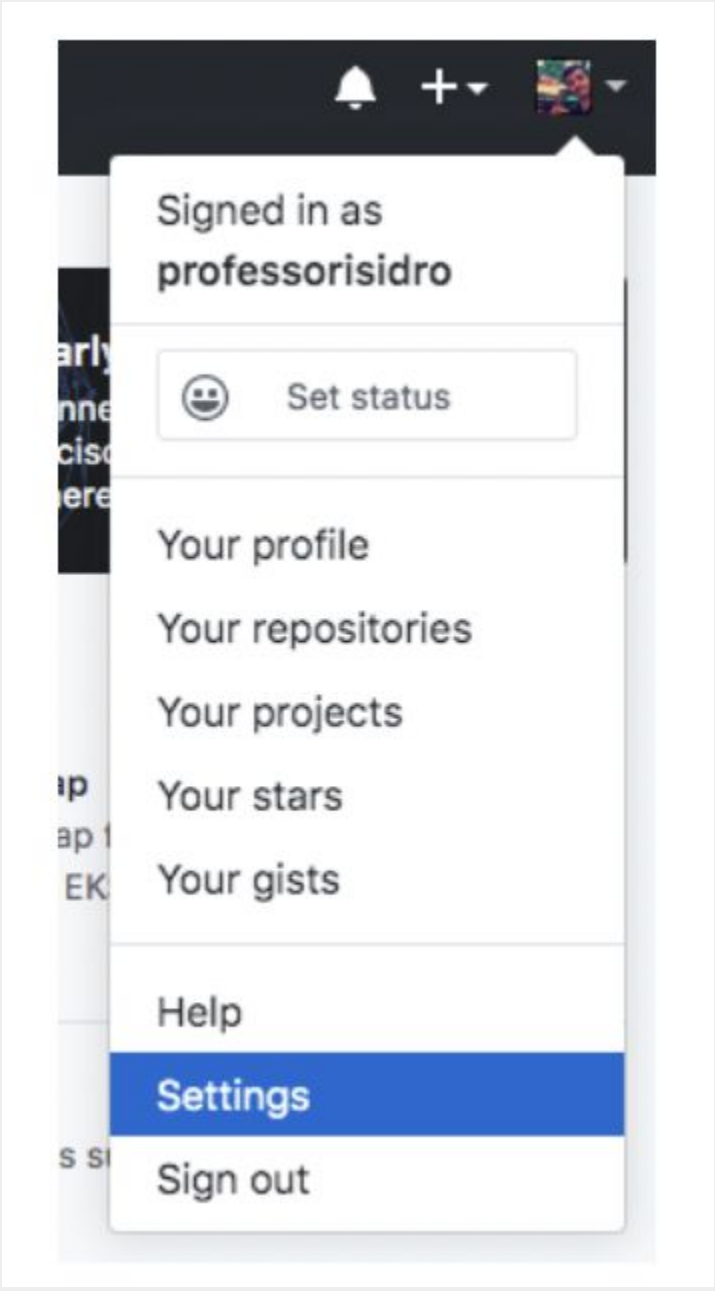
```
ssh-keygen -t rsa -b 4096 -C "seu email do github"
```

Neste caso, criamos uma chave RSA com 4096 bits a partir do email do github fornecido.

Inserindo sua chave nas suas configurações do GitHU:

Uma vez feito isso, haverá uma pasta (na sua área de usuário) chamada .ssh. Lá dentro, há um arquivo id_rsa.pub. É neste arquivo que terá sua chave. Ela inicia com “rsa-ssh” e termina com o email que você cadastrou. Copie esse conteúdo.

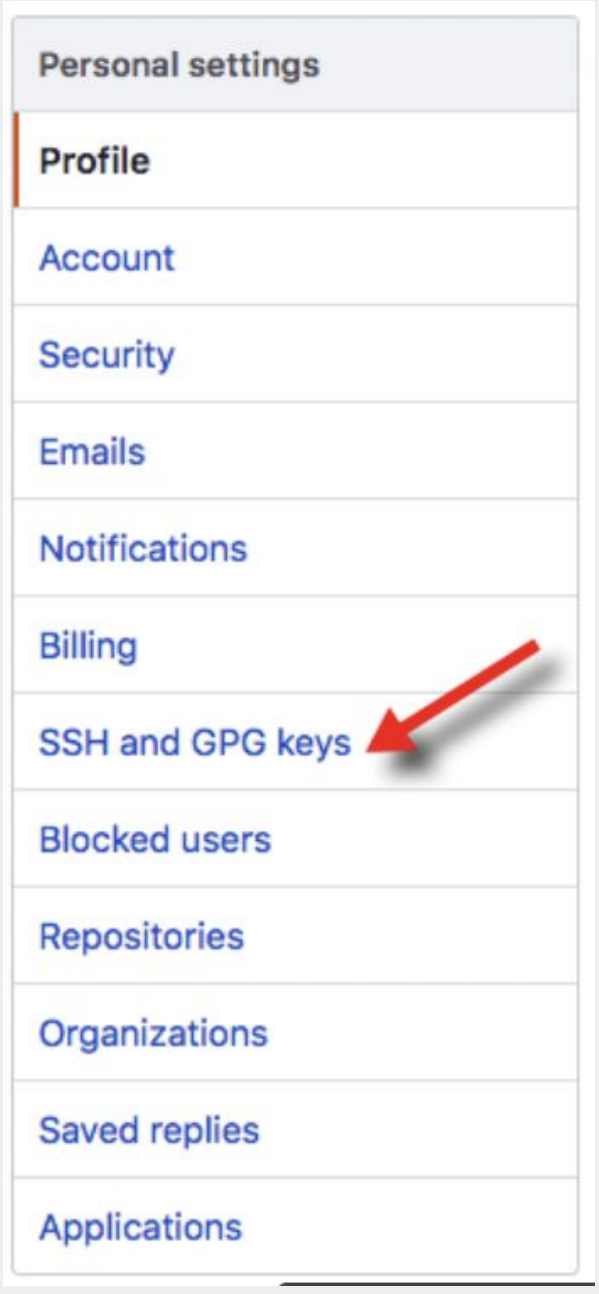
Agora, vá na sua conta do Github, no link “settings”.



Inserindo sua chave nas suas configurações do GitHUB - Passo 2

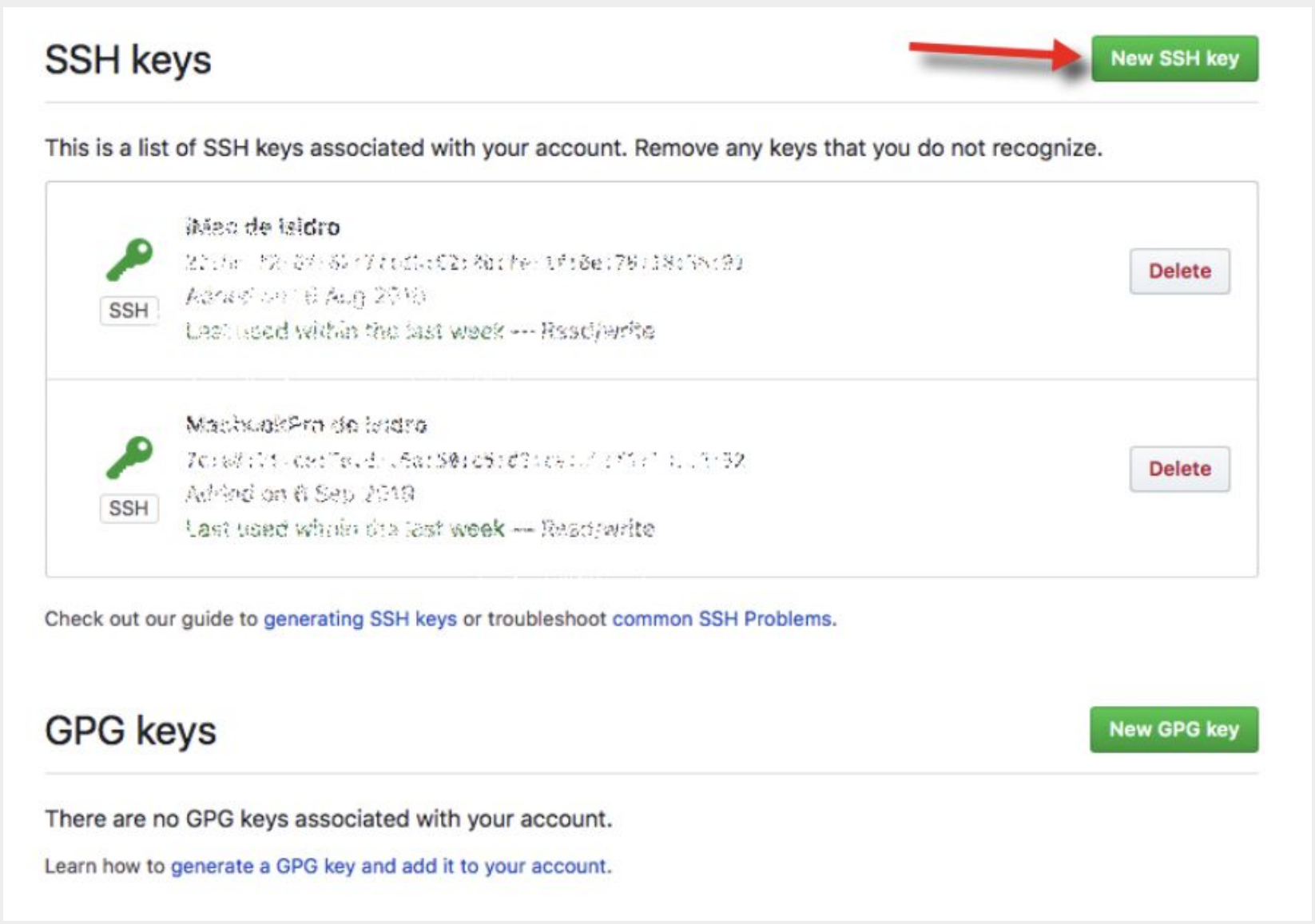
Descrição da imagem

Em seguida, vá no link “SSH and GPG Keys”



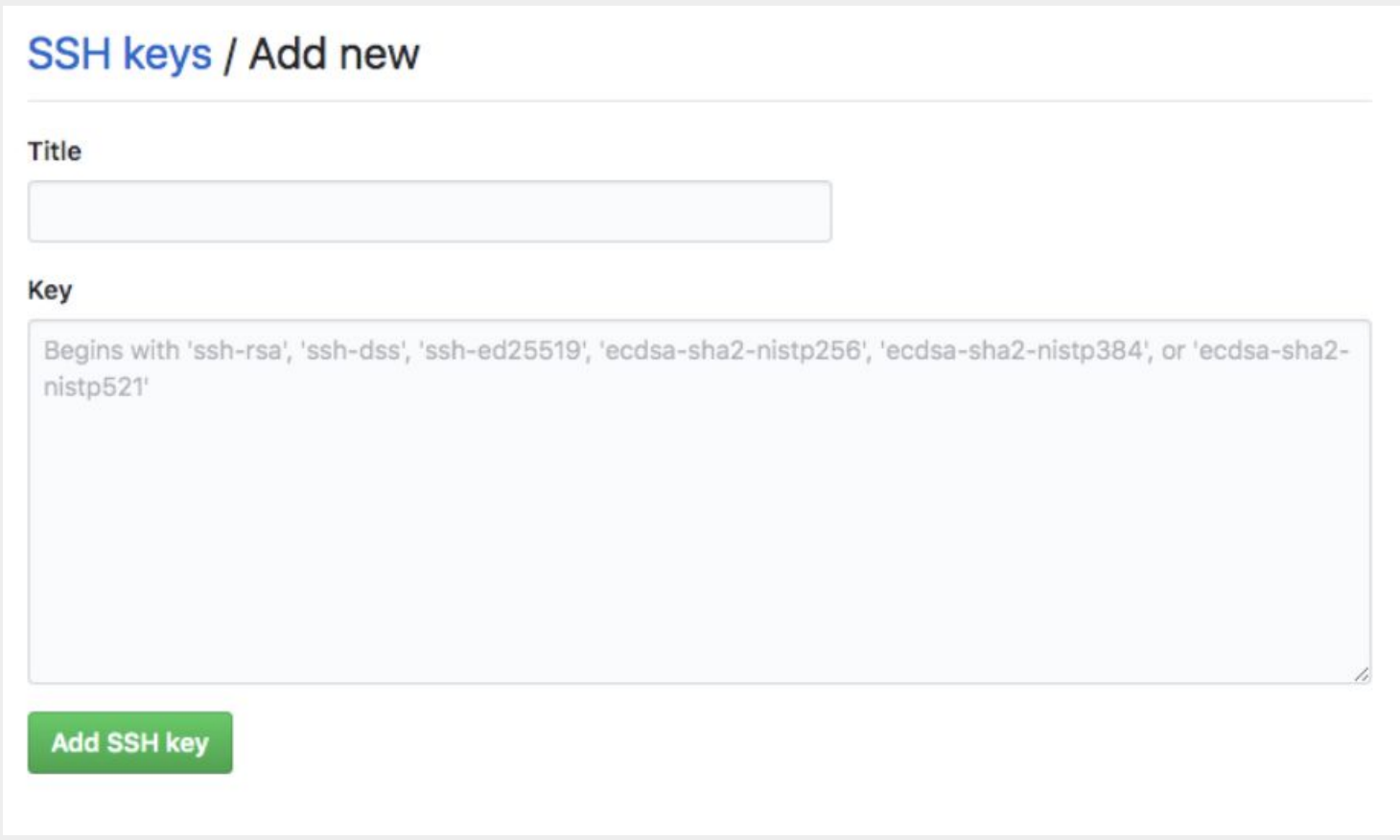
Inserindo sua chave nas suas configurações do GitHUB - Passo 3
Descrição da imagem

Se você já tiver alguma chave cadastrada, ok, entretanto é necessário criar uma nova chave.



Inserindo sua chave nas suas configurações do GitHUB - Passo 4
Descrição da imagem

Lá dentro, apenas dê um nome a ela (eu costumo identificar as minhas máquinas) e depois cole o conteúdo que você copiou lá do id_rsa.pub no campo KEY.



Inserindo sua chave nas suas configurações do GitHUB - Passo 5
Descrição da imagem

Beleza! Se estiver tudo ok, você só vai precisar testar da seguinte maneira:

Definindo seu nome de usuário:

```
ssh -T git@github.com
```

Se tudo ocorrer bem, você receberá uma mensagem assim:

```
Hi professorisidro! You've successfully authenticated, but
GitHub does not provide shell access.
```

Resumo do tópico

Neste tópico você aprendeu sobre:

1. Como identificar sua máquina junto ao Github.

Tópico 05

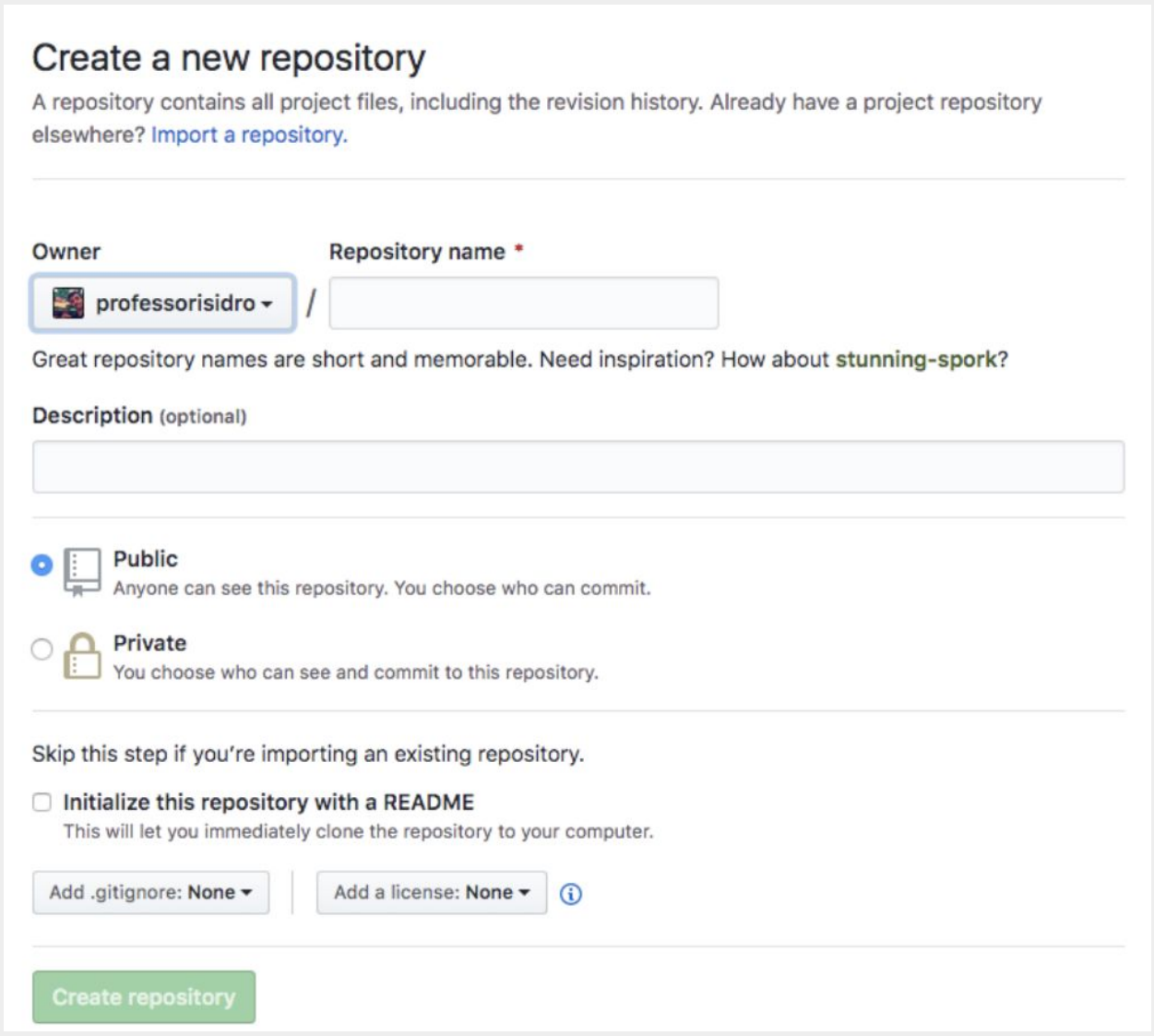
Comandos de SETUP do seu Repositório

Introdução: Comandos de SETUP do seu Repositório

Repositório, ou repo, é um diretório onde os arquivos do seu projeto ficam armazenados. Você pode armazenar códigos, imagens, áudios, ou qualquer outra coisa relacionada ao projeto no diretório.
Neste tópico vamos aprender a criar um repositório no GitHub.

Comandos de SETUP do seu Repositório

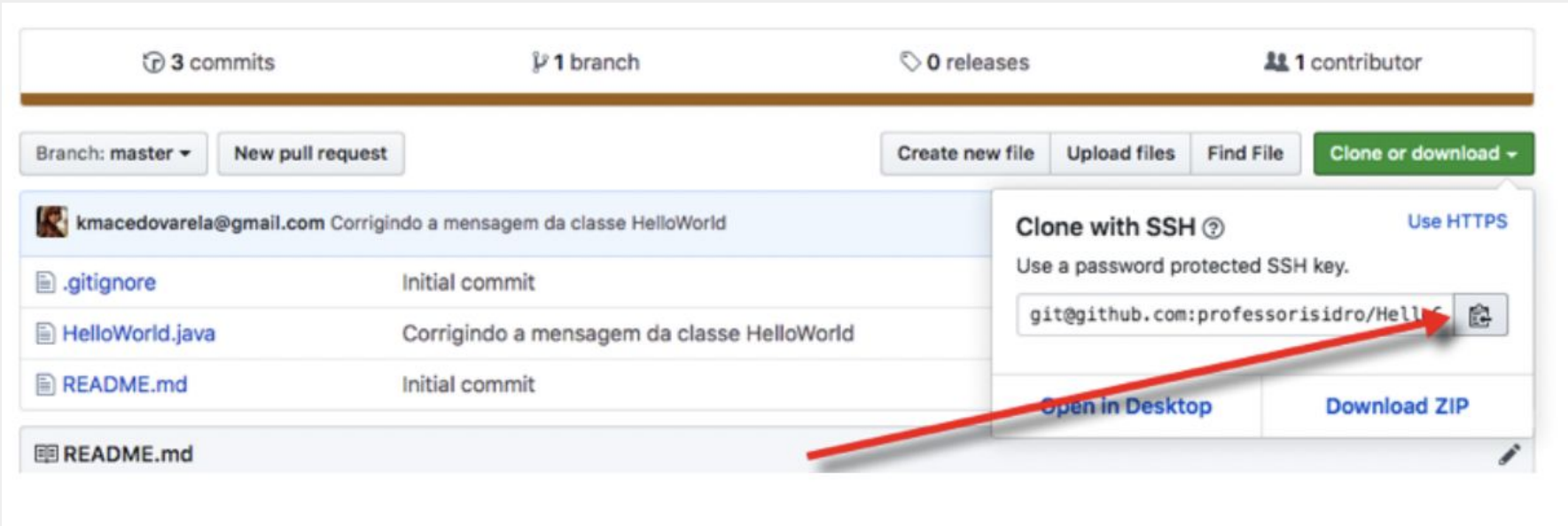
Você deverá criar seu repositório lá no GITHUB:



Criando um novo repositório.
Descrição da imagem

Uma vez isso feito, você vai precisar do caminho dele, para que você consiga sincronizar a pasta da sua máquina com o repositório.

A seguir, mostro como obter esse link:



Obtendo link do repositório.
Descrição da imagem

Atenção, isso é super importante: todo repositório remoto tem que estar vinculado a uma pasta local da sua máquina, pois a partir dela estará toda a hierarquia de pastas e arquivos do seu projeto. Portanto, tenha sempre sua área específica para os seus projetos do Github (eu sugiro uma pasta chamada projetos ou github).

Vinculando sua pasta local ao repositório

Alternativa 1: Clonando o link do seu repositório à sua pasta local

```
git clone "URL obtida no github"
```

Isso irá criar uma pasta na sua máquina de mesmo nome do repositório do Github (isso é bacana pq facilita muito) e, a partir daí você já poderá trabalhar.

Além disso, há uma pasta .git, dentro da sua pasta, responsável por armazenar todas as informações do seu projeto.

Alternativa 2: Criando sua pasta local e associando-a ao repositório. Após criar a sua pasta (com o nome que você quiser), você precisa dos seguintes comandos:

```
git init

git remote add origin "URL obtida no github"
```

Se você criou seu projeto no GITHUB e ele já está com um README na sua estrutura, é necessário, antes de mais nada, você obter na sua máquina local, tudo o que tiver no seu repositório (antes de colocar seus próprios arquivos).

```
git pull origin master
```

Pronto, já está associada a pasta local ao repositório remoto. Agora bora para as atividades do dia-a-dia.

Resumo do tópico

Neste tópico você aprendeu sobre:

1. Como criar um repositório no GitHub.

Tópico 06

Comandos do seu dia-a-dia

Introdução: Comandos do seu dia-a-dia

Existem vários comandos que você vai utilizar no dia a dia, quase sempre os mesmos... então logo você vai decorar. Mas enquanto não decora, separamos para você **uma lista dos principais comandos.**
Neste tópico vamos aprender mais sobre eles.

Comandos do seu dia-a-dia

Vamos entender o que significa cada comando e, a partir daí, traçar um pequeno guia de “boas práticas” de versionamento.

```
git status
```

Verifique quais são os arquivos que foram adicionados ao projeto, quem está atualizado, quem está efetivado.

```
git add "arquivo"/"diretório"
```

Insere os arquivos ou diretórios (pode ser o diretório atual, representado pelo ponto .) para serem rastreados pelo controle de versão. Sempre que você alterar ou inserir um novo arquivo, adicione-o ao seu controle de versões.

```
git pull origin "nome da branch"
```

Obtém do repositório a última atualização da ramificação do projeto. Em geral o nome padrão da Branch é master. E uma “Branch” é uma nova ramificação da estrutura do projeto. Usar Branches é muito útil para você preservar versões estáveis do seu software.

Atenção: A branch master é sempre aquela versão mais atual do sistema (a de produção), entretanto a equipe de desenvolvimento pode (e deve) trabalhar em inclusões de novos módulos do sistema em uma branch separada para não “contaminar” uma versão estável.

```
git commit -m "Mensagem do Commit"
```

Efetiva as alterações locais feitas no seu repositório como sendo "última versão”. Cuidado: em GIT, o Commit efetiva a versão local como definitiva. Ainda falta fazer o upload para o repositório.

```
git push origin "nome da branch"
```

Efetiva o Upload das suas alterações para o Repositório remoto. Lembre-se que só sofrerão upload os arquivos que forem efetivados (“commitados”).

Boas Práticas para seu dia-a-dia:

Apenas para que você tenha uma facilidade de operação e também garantir um pequeno procedimento, vamos lá com a “receita de bolo” para você atualizar seu repositório.

Lembre-se: cada commit é uma efetivação de elementos relevantes, ou seja, mudanças que podem afetar seu software de forma efetiva.

```
1. git pull origin master // obtenho as últimas alterações
feitas
pelos outros
2. git add . // adiciono meus novos arquivos
3. git commit -m "mensagem" // efetivo as alterações
4. git push origin master // faço o upload para o
repositório
```

Claro que, sempre entre um comando e outro, vale um git status para acompanhar o que acontece.

Espero que você tenha curtido esse material complementar!

Abraços, bons estudos e nos vemos em breve!

Resumo do tópico

Neste tópico você aprendeu sobre:

1. Os principais comandos Git e terminal.

Referências Bibliográficas

FERREIRA, Gabriel. Instalando e Configurando o GIT. 2022. Elaborado pelo Prof. Isidro. Disponível em: http://gabsferreira.com/instalando-o-git-e-configurando-github/. Acesso em: 19 jan. 2022.

MASSETTO, Francisco Isidro. Manual de Sobrevivência do GitHUB. 2022. Elaborado pelo Prof. Isidro. Disponível em: https://www.professorisidro.com.br/manual-de-sobrevivencia-do-github/. Acesso em: 19 jan. 2022.

GAMA ACADEMY