

Laboratório 2.1

Objetivo: Criar um repositório Git e validar o push de alterações para o Projeto;



Criando a conta no github:

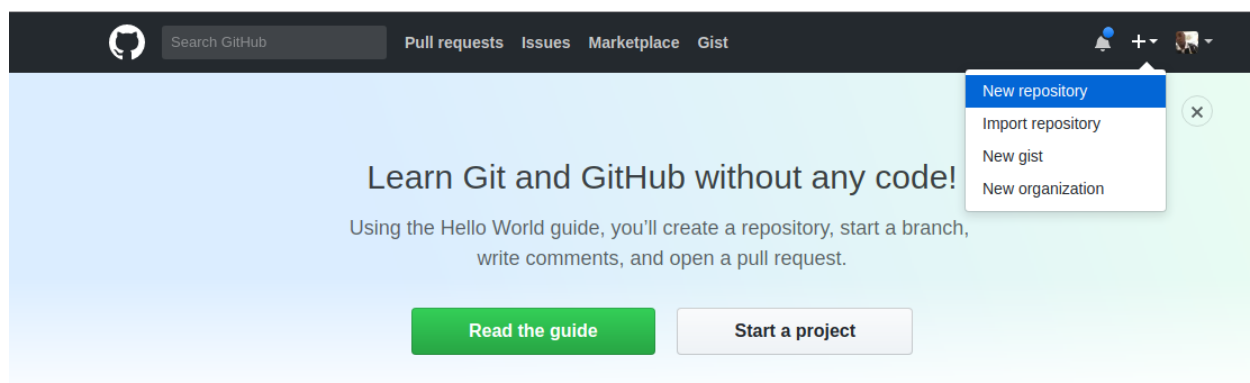
1. O primeiro passo para execução deste laboratório é a criação de uma conta no Github (Caso você ainda não possua uma); para isso faça o Registro de sua conta acessando o endereço <https://github.com/join>, uma vez que o registro seja finalizado certifique-se de que o host utilizado para este Lab possui o git instalado, instruções de instalação para Linux e para Windows podem ser adquiridas no link abaixo:

- [Primeiros passos - Instalando Git](#)

2. Com o Git devidamente instalado configure as opções globais de usuário e email de acordo com os dados de sua conta no Github:

```
# git config --global user.email "usuario@email.com.br"
# git config --global user.name "usuario"
```

3. Faça login na conta GitHub e navegue até a página do usuário. No canto superior direito, clique no botão "+" e Selecione "Novo repositório".




Para padronizarmos a tarefa nomeie o novo projeto como **"hello-ruby"** e coloque uma descrição simples. Em seguida selecione **"Inicializar este repositório com um README"**, e clique no botão **"Criar repositório"**.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 helcorin ▾

Repository name

/ hello-ruby ✓

Great repository names are short and memorable. Need inspiration? How about **shiny-guide**.

Description (optional)

Ruby app simples project created to test devops paterns and other things



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

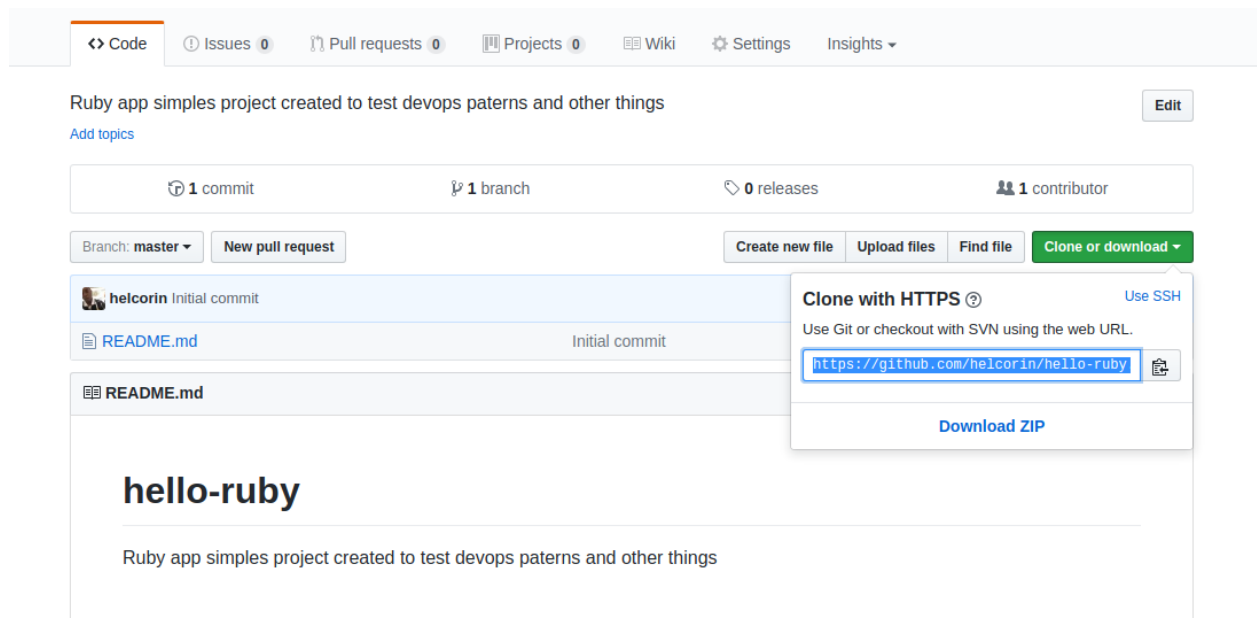
Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

4. Na etapa seguinte você executara o clone deste repositório para o diretório local, para isso copie a URL HTTPS para a área de transferência clicando no botão verde **"Clone or Download"** no repositório criado a pouco no Github. Não esqueça de selecionar a opção "Usar HTTPS" uma vez que não fizemos a configuração da chave SSH, copie o URL fornecido na área de transferência. Este URL será passado como um argumento para o git clone;



The screenshot shows the GitHub interface for a repository named 'hello-ruby' by user 'helcorin'. The repository description is 'Ruby app simples project created to test devops paterns and other things'. The page shows 1 commit, 1 branch, 0 releases, and 1 contributor. A modal window is open, displaying the 'Clone with HTTPS' option, which provides the URL 'https://github.com/helcorin/hello-ruby' and a 'Download ZIP' button. The 'Clone with HTTPS' option is selected, and the URL is highlighted in blue. The 'Use SSH' option is also visible but not selected. The 'Download ZIP' button is at the bottom of the modal.

No ambiente onde o git fora instalado execute:

```
# git clone https://github.com/seu-usuario/ruby-hello.git

Cloning into 'hello-ruby'...
remote: Counting objects: 3, done.
...
```

Este repositório será criado no diretório local a partir do qual o git clone foi executado:

```
# cd ruby-hello
```

5. Usando o editor de sua preferência edite o arquivo README.md adicionando algum conteúdo. (A idéia é que o arquivo seja alterado para que ele possamos validar o commit);

```
# git add README.md  
# git commit -m "Changed the README.md"
```

Esse execução irá adicionar todos os arquivos já modificados para o index e em seguida executar o commit, finalmente execute o push das mudanças para o repositório no GitHub:

```
# git push origin master
```

No processo de autenticação o Git solicitará o nome de usuário e a senha do GitHub, verifique os arquivos no repositório na interface da web do GitHub, as alterações executadas deverão ser visíveis.

6. Crie uma nova Branch para adicionar algum código ao nosso projeto:

```
# git checkout -b initial
```

7. Dentro dessa branch crie um arquivo chamado **app.rb** com o conteúdo abaixo:

```
require 'sinatra'  
class HelloWorld < Sinatra::Base  
  get '/' do  
    "Hello, world!"  
  end  
  get '/:name' do  
    "Hello, #{params[:name]}!"  
  end  
end
```

8. Além deste arquivo crie um arquivo chamado **config.ru**:

```
require './app'  
run HelloWorld
```

9. Como todas as dependências serão gerenciadas utilizando um bundler utilizaremos um gemfile para definir essas depências, para isso crie um terceiro e ultimo arquivo chamado **Gemfile*** conforme abaixo:

```
source 'https://rubygems.org'  
gem 'sinatra'  
gem 'minitest'  
gem 'rack-test'
```

Se houverem dificuldades ou alguma duvida sobre a sintaxe utilize o repositório disponível em <https://github.com/fiap2tin/hello-ruby> como base;

10. Uma vez que os arquivos tenham sido criados vamos adicioná-los ao index:

```
# git add app.rb config.ru Gemfile  
# git status
```

11. Em seguida crie o commit a ser enviado para o Github:

```
# git commit -a -m "Creating initial app.rb file with Gemfile dependences"
```

12. Finalmente faça o push das alterações para entrarmos na fase final:

```
# git push --set-upstream origin initial
```

Questão Rápida: Se já temos um repositório git com esse código porque não apenas criar um fork?

Para não perder a graça, a ideia desse primeiro laboratório é alinhar e nivelar alguns conhecimentos sobre Git que serão uteis em aulas vindouras, desconsiderando isso o procedimento mais comum realmente seria a criação de um fork, se você não está familiarizado com essa ideia pode dar uma olhada nos links abaixo:

- <https://git-scm.com/book/pt-br/v1/Git-Distribu%C3%ADo-Contribuindo-Para-Um-Projeto>;