

Material Extra Treinamento GIT

1 Impedindo arquivos de serem versionados

Em algumas situações, alguns arquivos não devem ser versionados, são esses arquivos:

- gerados pela compilação
- minificados
- binários
- etc...

Vejamos como informar ao git para ignorar esses arquivos.

2 Ignorando Arquivos usando `.gitignore`

O git utiliza o arquivo `.gitignore` para ignorar arquivos indesejados.

Para cada linguagem/tipo de projeto, os arquivos podem variar e tem diferentes localizações.

Criador de arquivos `.gitignore` para projetos: <https://www.gitignore.io/>

3 `.gitignore` para java

Exemplo de arquivo `.gitignore` para projetos Java.

```
# Compiled class file
*.class
# Log file
*.log
# BlueJ files
*.ctxt
# Mobile Tools for Java (J2ME)
.mtj.tmp/
# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar
# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
### Java-Web ###
## ignoring target file
target/
```

4 Configurando Atributos de Arquivos no Git

Às vezes é necessário incluir no versionamento arquivos que não são texto.

Por padrão o `git` tenta obter as diferenças (*diff*) entre as versões do histórico, o que não faz muito sentido em um arquivo de imagem por exemplo, já que a saída texto da diferença não ajuda.

Uma das finalidades da configuração de atributos é **informar ao git como tratar arquivos não textuais**.

5 Configurando Atributos de Arquivos no Git

Essa configuração é feita no arquivo **.gitattributes** de cada projeto e informa ao git como tratar cada tipo de arquivo.

```
*.png -crlf -diff
```

Usando essa configuração, o git não irá tentar converter ou corrigir caracteres de final de linha (CRLF), nem obter as diferenças entre as mudanças nesse arquivo (ao executar `git show` ou `git diff`).

6 Exemplo de configuração Atributos de Arquivos no Git

Para informar ao git para tratar todos os arquivos jar como dados binários, adicione a seguinte linha ao seu arquivo `.gitattributes`:

```
*.jar binary
```

Usando essa configuração, o git não irá tentar converter ou corrigir caracteres de final de linha (CRLF), nem obter as diferenças entre as mudanças nesse arquivo (ao executar `git show` ou `git diff`).



A opção `binary` no arquivo `.gitattributes` é equivalente ao uso das opções `-crlf -diff`

7 Exemplos de Configuração de Atributos

O projeto `configs-git` no gitlab contém arquivos de configuração de exemplos que podem ser utilizados nos projetos Celepar.

<http://gitlab.celepar.parana/gcgit-configs-git/configs-git>

8 Arquivo `.gitattributes`

Forçando final de linha como *Line Feed/Linux*.

```
# Por padrão todos os arquivos são texto, normalize os finais de linha.  
* text=auto eol=lf  
# These files are text and should be normalized (convert crlf => lf)  
# Git Files  
.gitattributes text eol=lf  
.gitignore text eol=lf  
.gitconfig text eol=lf
```

```
# Web  
*.css      text eol=lf  
*.map      text eol=lf  
*.sass     text eol=lf  
*.json     text eol=lf  
*.js       text eol=lf  
*.htm      text eol=lf  
*.chm      binary  
*.html     text eol=lf  
*.cshtml   text eol=lf  
*.xml      text eol=lf  
*.svg      text eol=lf
```

9 Arquivo `.gitattributes`

Exemplo de arquivos binários

```
# Imagens & Vídeos
*.bmp      binary
*.png      binary
*.jpg      binary
*.jpeg     binary
*.gif      binary
*.ico      binary
*.mov      binary
*.pdf      binary
*.mp4      binary
*.mpg      binary
*.webm     binary
```

```
# Arquivos Compactados
*.tar.gz   binary
*.tgz      binary
*.gz       binary
*.zip      binary
*.7z       binary
*.nupkg    binary
# Fontes
*.ttf      binary
*.eot      binary
*.woff     binary
# Config
*.conf     text eol=lf
```

10 Arquivo `.gitattributes`

Exemplo de arquivos do Visual Studio no Windows

```
# cSharp / Visual Studio
*.bat      text eol=crlf
*.cmd      text eol=crlf
*.cs       text eol=crlf diff=csharp
*.csproj   text eol=crlf
*.h        text eol=crlf
*.md       text eol=crlf
*.msbuild  text eol=crlf
*.ps1      text eol=crlf
*.sdf      binary
*.sln      text eol=crlf
*.tt       text eol=crlf
*.xaml     text eol=crlf
```

11 Criando um *Merge Request* no GitLab

O *Merge Request* no GitLab é o equivalente ao *Pull Request* do GitHub.

O processo de criar um *Merge Request* consiste de alguns passos:

1. Fazer o fork do projeto original
 - Estando na área do projeto, basta clicar no botão **fork**, logo acima do endereço de clone do projeto.
 - Selecionar uma das áreas de destino possíveis para o projeto clonado (normalmente o *namespace* de usuário)
2. (no fork do projeto) Criar um branch para fazer as modificações
 - Cria o branch que irá conter as modificações, p.ex. *feature-nova*
 - Codifica as modificações
 - Envia as modificações no branch *feature-nova* para o fork do projeto no servidor gitlab (`git push origin feature-nova`)
3. Solicitar o *Merge Request*
 - Ao fazer o push do branch com as modificações para o fork do projeto, o próprio gitlab retorna uma mensagem com a URL para solicitar o *Merge Request* pela interface.

A seguir são ilustrados os passos necessários.

11.1 Criando o fork

Criar o fork de um projeto é, na maioria dos casos, um processo de 2 passos.

1. Clique no botão *fork* localizado na página do projeto *hello-\$DATA*, perto do botão *Star*.



Vocês deverão fazer o *fork* do projeto `hello-$DATA` do *namespace* do Leslie, pois já tem um projeto `hello` no *namespace* de vocês.

treinamento-git > hello > Details



hello

Projeto de Exemplo para o Treinamento Git



Files (328 KB) Commits (19) Branches (2) Tags (3) Readme

Add Changelog

Add License

Add Contribution guide

Add Kubernetes cluster

Set up CI/CD

11.2 Selecionando onde o projeto destino será criado

Após clicar no botão *fork*, o gitlab mostra uma tela onde você pode escolher em qual *namespace* irá colocar o *fork* do projeto. O gitlab mostra somente os *namespaces* onde você tem acesso de escrita. Selecione o *namespace* do seu usuário e prossiga.

Fork project

A fork is a copy of a project.
Forking a repository allows you to make changes without affecting the original project.

Select a namespace to fork the project



Leslie Harley Watter



treinamento-git

11.3 Resultado do fork

Após a operação de *fork*, o gitlab mostra que o projeto dentro do namespace de usuário é um *fork* do projeto origem.



11.4 Clonando e Modificando o projeto resultante



O processo de clone e modificação do projeto segue o mesmo processo utilizado para outros projetos.

Aqui iremos clonar o projeto direto do namespace do usuário e criar um *branch* chamado `corrige-cap`.

```
git clone http://gitlab-hml.celepar.parana/leslieh/hello.git
cd hello
git checkout -b corrige-cap
```

bash

```
$ git clone http://gitlab-hml.celepar.parana/leslieh/hello.git
Cloning into 'hello'...
remote: Counting objects: 87, done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 87 (delta 22), reused 0 (delta 0)
Unpacking objects: 100% (87/87), done.

$ cd hello

$ git checkout -b corrige-cap
Switched to a new branch 'corrige-cap'
```

11.5 Modificando a Capa do Projeto

Troque a URL do projeto, corrigindo o endereço do servidor, de `gitlab.celepar.parana` para `gitlab-hml.celepar.parana` no arquivo `README.md`.



Use o comando `sed` para efetuar a mudança:

```
sed -i 's@gitlab\.celepar@gitlab-hml\.celepar@g' README.md
```

bash

Comite a correção com a mensagem:

Corrigindo URL do projeto fonte

11.6 Verificando as modificações

```
$ git add README.md

$ git commit README.md -m 'Corrigindo URL do projeto fonte'
[corrige-cap 6c28626] Corrigindo URL do projeto fonte
1 file changed, 5 insertions(+), 5 deletions(-)

$ git branch
* corrige-cap
master
```

11.7 Enviando as modificações

Envie o branch `corrige-cap` para o GitLab e observe que o próprio git retorna a URL do gitlab para abrir o *Merge Request*.

```
$ git push origin corrige-cap
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 343 bytes | 343.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote:
remote: To create a merge request for corrige-cap, visit:
remote: http://gitlab-hml.celepar.parana/leslieh/hello/merge_requests/new?merge_request%5Bsource_branch%5D=corrige-cap
remote:
To http://gitlab-hml.celepar.parana/leslieh/hello.git
 * [new branch] corrige-cap -> corrige-cap
```

11.8 Links para Merge Request texto e web

Observe as linhas:

```
remote: To create a merge request for corrige-cap, visit:
remote: http://gitlab-hml.celepar.parana/leslieh/hello/merge_requests/new?merge_request%5Bsource_branch%5D=corrige-cap
```

e também na interface web do projeto:

Leslie Harley Watter > hello > Details

You pushed to `corrige-cap` at Leslie Harley Watter / hello 27 minutes ago

Create merge request

11.9 Merge Request no Gitlab 01/02

Abra a URL informada no seu navegador e preencha as informações do *Merge Request*. Observe que o próprio GitLab já preenche alguns campos automaticamente.

Leslie Harley Watter > hello > Merge Requests

New Merge Request

From leslieh/hello:corrig-e-capa into treinamento-git/hello:master

Change branches

Title

Corrige capa

Start the title with **WIP:** to prevent a **Work In Progress** merge request from being merged before it's ready.

Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview

B I

Write a comment or drag your files here...

Markdown and [quick actions](#) are supported

[Attach a file](#)

11.10 Merge Request no Gitlab 02/02

Assignee

Assignee

Assign to me

Milestone

Milestone

Labels

Labels

Source branch

corrig-e-capa

Target branch

master

Change branches

☐ Remove source branch when merge request is accepted.

☐ Squash commits when merge request is accepted. [About this feature](#)

Contribution

☐ Allow commits from members who can merge to the target branch. [About this feature](#)

Submit merge request

Cancel

11.11 Enviando o Merge Request


Após preencher as informações do *Merge Request*, clique no botão *Submit merge request* para enviar a solicitação.

Observe que abaixo do botão *Submit merge request*, aparecem os commits que farão parte dessa solicitação.

11.12 Merge Request resultante

treinamento-git > hello > Merge Requests > 12


Open


Opened 52 seconds ago by  Leslie Harley Watter


Edit

Close merge request




Corrigindo URL do projeto fonte

 Request to merge **leslieh:corrigir-cap** into **master**


Open in Web IDE Check out branch 

 Merge ☐ Remove source branch


You can merge this merge request manually using the [command line](#)

 0  0 


Discussion 0 Commits 1 Changes 1




Write Preview




Write a comment or drag your files here...

Markdown and [quick actions](#) are supported  [Attach a file](#)

Observe que, caso não haja conflito, é possível fazer o merge das modificações pela própria interface do GitLab, clicando no botão *Merge*.

 Merge ☐ Remove source branch

 You can merge this merge request manually using the [command line](#)

É possível ainda remover o branch de origem após o merge, limpando assim o repositório.

11.13 Modificações para um novo Merge Request

É possível também criar uma solicitação de *Merge Request* a partir de modificações em quaisquer branches.

Mude para o branch `master`:

```
git checkout master
```

bash

Adicione a linha seguinte ao final do arquivo `README.md`

```
+ Wiki :notebook: -> http://gitlab-hml.celepar.parana/treinamento-git/hello/wikis/home
```


Commite a modificação usando a mensagem de commit:

```
Incluindo wiki na lista de elementos
```

Envie para o servidor:

```
git push origin master
```

bash

11.14 Criando um *Merge Request* a partir da Interface

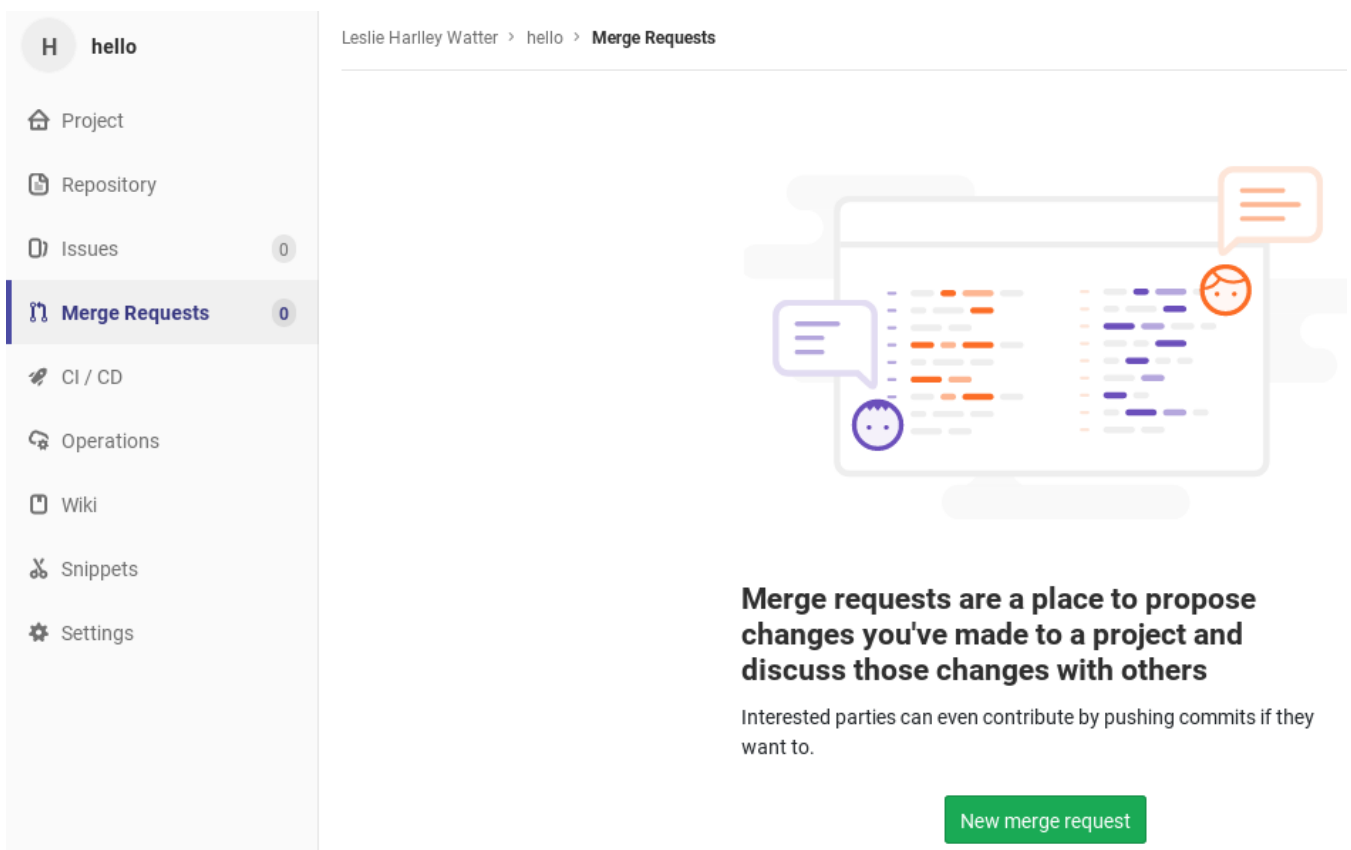
Observe que dessa vez o gitlab **não** forneceu nenhuma URL para criação automática do *merge request*.

```
$ git push origin master
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 658 bytes | 658.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
To http://gitlab-hml.celepar.parana/leslieh/hello.git
6acd9f2..f0bec01 master -> master
```

Faremos a criação a partir da interface WEB.

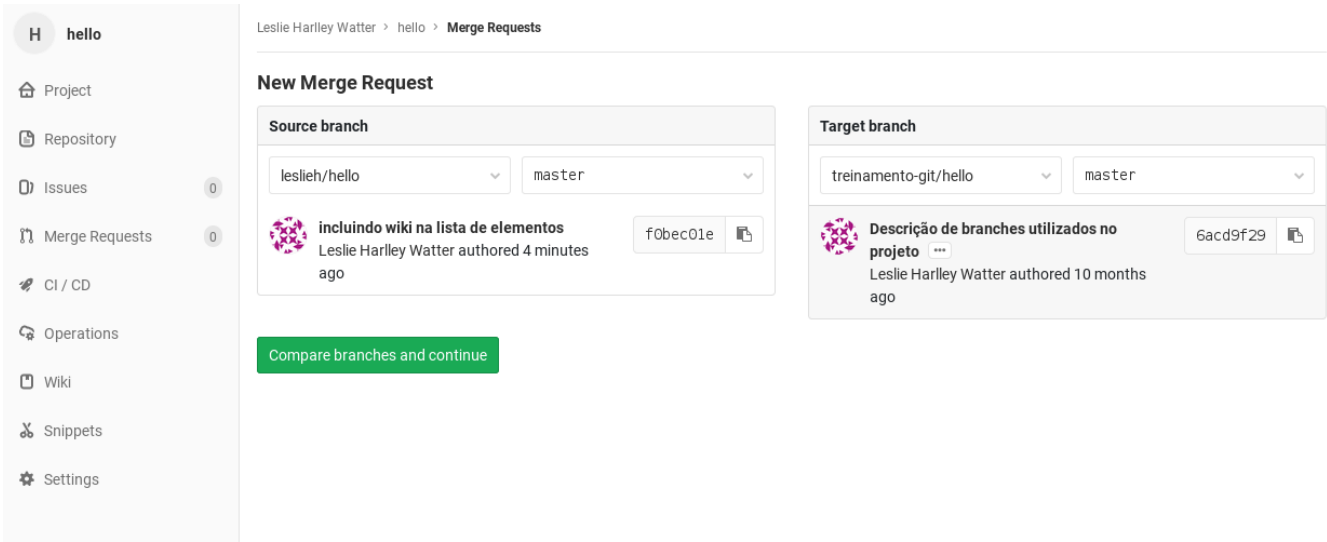
11.15 *Merge Request* a partir da Interface WEB

Para criar o merge request, clique no link *Merge Requests* no menu lateral esquerdo. A seguir, clique no botão *New merge request* do lado direito.




11.16 Informando o branch de origem e o de destino

A tela de criação de *merge request* permite a seleção dos projetos/branches de origem (*lado esquerdo*) e destino (*lado direito*). Ao clicar no botão *Compare branches and continue* o processo de criação de *merge request* segue o mesmo modelo já utilizado.




11.17 Resumindo...

Para criar um *merge request*, basta fazer o fork do projeto, fazer as modificações e enviá-las para o servidor.



Crie um branch para cada feature nova. Fica mais fácil de manter o desenvolvimento 'isolado' e atualizado até conseguir um merge fácil.



Se houver conflito na aplicação do merge, você terá que realizá-lo (*merge*) manualmente no lado cliente.

12 Processo Celepar

12.1 Treinamento

- O treinamento é pré-requisito para a criação do usuário no GitLab para os funcionários da Celepar.
- Após todos os integrantes de uma equipe terem sido treinados o projeto se torna elegível para a migração do SVN->GitLab.

12.2 Migração de Projetos

- **Todos** os projetos que estiverem no **SVN** serão migrados para o GitLab cada um a seu tempo em conjunto com a equipe de desenvolvimento.
 - Após todos os integrantes do projeto terem passado pelo treinamento.
- Projetos com senha no Controle de Versão – projetos que porventura tenham senhas de conexões a bancos de dados, e outras conexões deverão ser analisados para ter essas senhas armazenadas em locais específicos (chaveiro no `jenkins` ou injetadas pela GTI no caso do `estaleiro`) e removidas do controle de versão.

12.3 Deploy

- *Java* - o estaleiro foi modificado para buscar as tags do repositório no gitlab. Para projetos migrados, é necessário remover o deploy existente e fazer um novo deploy.
- *PHP* - o jenkins tem suporte a deploy usando o git, mediante solicitação de migração do projeto.
- *ASP* - o deploy será migrado para o Jenkins no futuro. Os projetos ASP serão os últimos a serem migrados.

12.4 Gitlab - Convenções

- Nomes de Grupos e Projetos Corporativos
 - grupo - **gcgit**- (*grupo celepar git*)
 - projeto - nome do projeto
 - Exemplo: projeto de nome **configs-git** :
 - Grupo: `gcgit-configs-git`
 - Projeto: `configs-git`
 - URL: <http://gitlab.celepar.parana/gcggit-configs-git/configs-git.git>
 - O estaleiro buscará as tags somente nos projetos que tenham sido criados pela DIOPE via OS (dentro de grupos gcgit).

13 Configurações Celepar

13.1 Proxy

Para obter as configurações de proxy veja:

1. variáveis de ambiente do shell

- http_proxy – configuração proxy http
- https_proxy – configuração proxy https
- no_proxy – exclusão de máquinas do proxy

```
echo $http_proxy $https_proxy $no_proxy
```

bash

2. configuração do git

```
# mostra inclusive variáveis de ambiente configuradas
git config --get http.proxy
# mostra configurações a partir do arquivo ~/.gitconfig
git config --get --global http.proxy
```

bash

13.2 Proxy para servidores EXTERNOS

Para configurar o proxy para acessar servidores **externos**, inclua seguinte trecho no arquivo `$HOME/.gitconfig` substituindo os campos de acordo:

```
[http]
proxy = http://<username>:<senha>@<endereco-do-proxy-celepar>:<porta>
```

Por Exemplo:

```
[http]
proxy = http://user:senha@proxy.celepar.parana:8080
```

13.3 Ignorando Proxy para servidor INTERNO

Para utilizar o servidor **interno**, sem passar pelo proxy, acrescente o seguinte trecho no arquivo `$HOME/.gitconfig`

```
[http "http://gitlab.celepar.parana"]
  proxy =
```

14 Configurando Cache de Credenciais

A configuração de cache de credenciais é muito útil ao se usar o git com os protocolos HTTP/HTTPS, uma vez que armazena as senhas e/ou tokens de acesso em memória por um número de segundos configurável.

Habilitando o cache de Credenciais

```
git config --global credential.helper cache
```

bash

Configurando o tempo de expiração das credenciais em 10 minutos

```
git config --global credential.helper 'cache --timeout=600'
```

bash

Desabilitando o cache de credenciais

```
git config --global --unset credential.helper
```

bash

Autor: Leslie Harley Watter

[Validate](#)