# Manual de apresentação do GIT

Módulo -Administrador & Usuário - Sistema E-sic Livre

# SUMÁRIO

# 1.Sobre o E-sic Livre

A informação que permanece sob a guarda do governo é publica e utilizada para fins específicos, sendo um bem público.

A informação contém arquivos como documentos, planilhas, estatísticas  fundamentais para a organização e implementação da democracia.

O objetivo da lei de acesso é informar aos cidadãos todas as informações produzidas nos processos do governo. A lei de acesso contribui para tornar as informações transparentes ao qualquer cidadão , pois acreditamos que o cidadão que conhece seus direitos relacionados a saúde,educação e segurança entre outros .Além de obter informações que podem ser necessárias.

A lei de acesso é regulamentada e utilizada em 90 países , sendo reconhecida por organizações como Organização das nações unidas (ONU) e a Organização dos Estados Americanos(OEA).

# 2.Como instalar o git em seu computador

**2.1 Instalando no Linux**

Se você quiser instalar o Git no Linux via um instalador binário, você pode fazê-lo com a ferramenta de gerenciamento de pacotes (packages) disponível na sua distribuição. Caso você esteja no Fedora, você pode usar o yum:

$ yum install git-core
Ou se você estiver em uma distribuição baseada no Debian, como o Ubuntu, use o apt-get:

$ apt-get install git

https://git-scm.com/book/pt-br/v1/Primeiros-passos-Configura%C3%A7%C3%A3o-Inicial-do-Git

# 2.Como instalar o git em seu computador

**2.1 Instalando no Linux**

Outra maneira de Instalar no Linux é :

Para instalar o Git no Ubuntu, ou em uma outra distribuição baseada em

Debian, execute em um terminal:

$ sudo apt-get install git

Para as demais distribuições do Linux, veja o comando em: http://git-scm.com/download/linux

# 2.Como instalar o git em seu computador

**2.2 Instalando no Mac**

Baixe a última versão do instalador gráfico do Git paraMac OS X a partir

do link: https://code.google.com/p/git-osx-installer/downloads

Abra um terminal e prepare-se para utilizar o Git!Configurações básicas

É importante nos identificarmos para o Git, informando nosso nome e

e-mail. Em um terminal, execute os comandos a seguir:

$ git config --global user.name "E -Sic Livre"

$ git config --global user.email esiclivre.git@gmail.com  -Utilize seu nome e e-mail!

# 2.Como instalar o git em seu computador

2.1 Instalando no Linux

Para instalar o e-sic Livre utilizando o Git e necessário utilizar o comando

git clone http://portal.softwarepublico.gov.br/gitlab/e-sic-livre/e-sic-livre.git

O repositório sera clonado no diretório que foi demilitado.

# 2.Como instalar o git em seu computador

**2.2 GIT para Windows**

Para executar a instalacao do git no windows e necessário possuir a git bash que permite o gerenciamento do repositório , além do git bash  uma ferramenta Própria para o windows , totalmente executável .

Podendo ser baixada por este endereço  https://git-for-windows.github.io/

Para a clonagem utilize o comando

git clone http://portal.softwarepublico.gov.br/gitlab/e-sic-livre/e-sic-livre.git

O repositório sera clonado no diretório que foi demilitado.

# 3.Comandos básicos

1.git init : Inicia o diretório e transforma o diretório atual em um repositório git , criando o subdiretório “.git”

2. git init <diretório> : cria o diretório do projeto

3.git clone <repositório> : Clona o repositório para a máquina local

4.git clone <repositório> <diretório> : Clona o repositório <repo> para o diretório <dir>

# 3.Comandos básicos

5.- Comando .gitignore

Arquivo que contém os arquivos que não serão visíveis pelo git.

Arquivo .gitignore(exemplo)

Thumbs.db#Arquivo específico

*.html#Arquivos que terminam com “.html”

!index.html #Exceção, esse arquivo será visível ao git

log/ #Diretório específico

*/tmp#Qualquer diretório nomeado de “tmp” - Arquivos que já estavam sendo rastreados não são afetados.

# 3.Comandos básicos

6.-git add <arquivo|dir>

Adiciona as mudanças do arquivo <arquivo> ou do diretório <dir> para o próximo commit. O arquivo passa a ser rastreado.

7.$ git rm--cached<arquivo> - Para de rastrear o arquivo <arquivo>

8.$ git reset <arquivo> -Remove as mudanças do arquivo <arquivo> para o próximo commit.

# 3.Comandos básicos

-Salvando Alterações

9. git commit

Realiza o commite abre o editor para inserir uma mensagem.

9.1 git commit–m “<msg>”

Realiza o commit, com a mensagem <msg>.

9.2 git commit–a

Adiciona as mudanças dos arquivos já rastreados e realiza o commit. O editor será aberto.

# 3.Comandos básicos

-Salvando Alterações

9.3 git  commit – am <msg>

Adiciona as mudanças dos arquivos já rastreados e realiza o commitcom a mensagem <msg>.

9.4 git commit--amend–m <msg>

Substitui o último commite altera a mensagem para <msg>.

# 3.Comandos básicos

-Analisando os Arquivos na Área Transitória

10.git status

Lista os arquivos que estão e que não estão na área transitória, e os arquivos que não estão sendo rastreados.

10. 1 git status -s

Lista os arquivos de uma forma simplificada.

# 3.Comandos básicos

-Analisando Commits

11.git show - Exibe o último commit.

git show <commit>

Exibe o commit referenciado por <commit>.

11.1 git show <commit>:<arquivo>

Exibe o arquivo <arquivo> no commit <commit>.

# 3.Comandos básicos

12 . Tagging

12.1 git tag <tag> [<commit>] -Cria a tag <tag> para o último commitou para o commit<commit>.

12.2 gittag–a <tag> - Cria a tag< tag> completa para o último commite abre o editor para inserir uma mensagem.

12.3 gittag–a <tag> -m <msg> - Cria a tag<tag> completa para o último commitcom a mensagem <msg>.

# 3.Comandos básicos

13. git revert <commit>

Cria um novo commit no branch atual que desfaz o que foi introduzido no commit <commit>.

•Consertar um bug introduzido por um commit.

•Não remove o commit<commit>

git commit – am “Fixedbug in fun2”

# 3.COMANDOS BÁSICOS

14. git reset --soft <commit>

Altera apenas o HEAD para o commit<commit>. Não altera a área transitória nem o diretório de trabalho.

# 3.COMANDOS BÁSICOS

13.Excluindo Commits

13. git reset --hard <commit>

Altera a área transitória e o diretório de trabalho para o commit<commit>.

O comando git reset é uma das poucas formas de se perder informação utilizando o git, pois os commit sdeixam de aparecer no git log.

# 3.COMANDO BÁSICOS

14 .Atualizando o Repositório Local

14.1 git push <repositorio> Envia suas alteracoes para o repositorio remoto , necessario apos o commit

14.1 git fetch  <repositorio>

Baixa todos os dados do repositório <repo>.

14.2 git fetch <repo>  <branch>

Baixa todos os dados do branch < branch> do repositório <repo>.

14.3 git pull <repo>

Atualiza todos os dados do repositório <repo>, ou seja, realiza um fetch seguido de um merge.

# 3.Comandos Básicos

14.4 git pull --rebase para obter mudanças remotas. É uma prática segura porque nossos novos commits locais ainda não foram compartilhados, ou seja, outros membros nem sabem da existência desses commits. Por isso, alterá-los não traz grandes problemas.

14.5 Falha do push

Se um push não resultar em um fast forward do branch remoto, podem ocorrer erros como este -

error: remote 'refs/heads/master' is not an ancestor of

local 'refs/heads/master'.

Maybe you are not up-to-date and need to pull first? - Necessário atualizar repositório antes

error: failed to push to 'ssh://seuservidor.com/~voce/proj.git'

# 3.Comandos básicos

Possíveis causas

- usar 'git-reset --hard' para remover commit já publicados, ou

- usar 'git-commit --amend' para substituir commits já publicados ou

- usar 'git-rebase' para recriar qualquer commit já publicado.

Você pode forçar git-push para realizar a atualização precedendo o nome do branch com um sinal de +:

$ git push ssh://seuservidor.com/~voce/proj.git +master

# 3.COMANDO BÁSICOS

15. Excluindo no Repositório Remoto

15.1  git push <repositorio> :<branch>

Exclui o branch <branch> do repositório <repo>.

15.2  git push <repo> : <tag>

Exclui a tag <tag> do repositório <repo>.

# 3.COMANDOS BÁSICOS

16.Git Rebase - Retorno das configuracoes da branch master

16.1 git rebase origin/master #Realiza o rebase

17.Git mv <arquivo> renomeia arquivos

Para verificar o histórico das alterações gravadas no repositório, podemos

executar o comando git log :

$ git log

# 3.Comandos básicos

18.Git branch - Uma branch significa uma rota de desenvolvimento onde pode-se inserir novas versaos de codigos sem afetar outras rotas (Branches no caso)

18.1-Criando a Branch

-git branch master (Master branch padrao)

18.2-Executando a branch  -  git branch <nome da branch>

18.3 Trocando a branch padrao pela criada pelo administrador - git checkout master -> git checkout -b <nome da branch>

18.4  Deletando a branch - git branch -d <nome da branch>

# 3.Comandos básicos

19.GIT MERGE

19.1 Git merge branch name - Realiza merge com as alteracoes feitas no branch branchname no branch atual.

19.2 Resolvendo um merge - para editar arquivos com conflito [e necessario inicar a branch  e adicionar o arquivo dentro da merge e commitar.

git add file.txt

git commit

# 3.Comandos básicos

19.3 Desfazendo um merge

git reset --hard Head

19.4 Excluindo o commit do merge

git reset --hard  origin_head

19.5 Merges fast-foward

Caso especial , onde um merge resulta em um commit com dois pais , onde cada um foi desenvolvido em rotas diferentes (Branchs) e foram mescladas em um merge. Entretanto se a rota nao foi divergida , logo cada commit presente no branch atual esta contido no outro , então e necessário um fast foward. O HEAD da rota atual e movida para outro ponto do Head, sem que qualquer commit seja criado.

# 3.Comandos básicos

20. Protocolos utilizados pelo Git

A ferramenta Git utiliza 4 principais protocolos para o envio dos dados -

-Local - Protocolo utilizado pela maquina do administrador

-SSH -Suporta tanto a leitura quanto a escrita de dados

-GIT - Similar ao SSH, mas sem o mecanismo de autenticação.

-HTTP/HTTPS - O Git também suporta o protocolo HTTP, que é bastante utilizado quando estamos trabalhando em empresas que possuem um controle rígido de segurança, e a porta 22, utilizada pelo protocolo SSH, é bloqueada.

# 3.Funcionalidades do Git

-Controle de histórico

•Trabalho em equipe

•Marcação e resgate de versões estáveis

•Ramificação do Projeto

# 4.Referencias

Livros

1.Controlando versões com git e github - Casa do codigo

2. Livro da comunidade GIT -Vários autores