FUNDAÇÃO
UNIVERSIDADE
FEDERAL DE
MATO GROSSO DO SUL

Gerência de Configuração de Software – T01

Aula 2 - Introdução ao Git

Prof. Ricardo M. Kondo

Aula passada

- Conceitos básico
- Instalação git na máquina
- Configuração de usuário git
- Ferramentas de interface gráfica
- Criação de chave SSH
- Repositório GIT
- Fluxo de Trabalho
- Alguns comandos básicos

Comando básico do Git - Atualizar & mesclar

- Para atualizar seu repositório local com a mais nova versão, execute o comando ***git pull*** na sua pasta para obter e fazer *merge* (mesclar) alterações remotas
  - `git pull`
- Para fazer *merge* de um outro ramo (*branch*) ao seu ramo ativo (ex. *master*), use o comando:
  - `git merge <nome do ramo>`
- Em ambos os casos, o *git* tenta fazer o merge das alterações automaticamente. Infelizmente, isto nem sempre é possível e resulta em conflitos. Você é responsável por fazer o *merge* estes conflitos manualmente editando os arquivos exibidos pelo *git*. Depois de alterar, você precisa marcá-los como *merged* com o seguinte comando:
  - `git add <nome(s) do(s) arquivo(s)`
- Antes de fazer o *merge* das alterações, você pode também pré-visualizá-los usando
  - `git diff <ramo origem> <ramo destino>`

Comando básico do Git - Diff

- git diff
- Exibir diferenças entre *commits* e *branchs*
  - git diff [path]
- Diferença no diretório
  - git diff HEAD~1
- Mostra o que foi alterado no último commit
- Ver diff no GitLab

Comando básico do Git - Exibindo suas alterações

- Para exibir o status da árvore de trabalho, utilize o seguinte comando;
  - **git status** [<options>…] [--] [<pathspec>…]

- Exibe caminhos que têm diferenças entre arquivos de índice e o commit HEAD atual, caminhos que têm diferenças entre árvore de trabalho e o arquivo de índice e caminhos na árvore de trabalho que não são rastreados pelo Git

Comando básico do Git - Rotulando

- São etiquetas que demarcam um ponto (*commit*) que representa alguma
- mudança significativa no seu código, ou seja, uma versão (ou *release*) do seu projeto
- É recomendado criar rótulos para *releases* de software
- Usado também no SVN
- Criar um novo rótulo utilize o seguinte comando:
  - `git tag -a <nome da tag> -m <message>` //o comando -m é opcional usado para inserir uma mensagem (*tag annotated)*
- Obter informações sobre a tag, execute o seguinte comando:
  - `git show <nome da tag>`

Comando básico do Git - Rotulando

- Versionando uma *tag*, execute o seguinte comando:
  - `git push origin <nome da tag>`
- Criar um novo rótulo utilize o seguinte comando:
  - `git tag -a <nome da tag> -m <message>` //o comando -m é opcional usado para inserir uma mensagem (*tag annotated)*
- Obter informações sobre a tag, execute o seguinte comando:
  - `git show <nome da tag>`

- Excluir uma *tag*
  - Normalmente *tags* **não** são excluídas a menos que tenham sido geradas por engano! Se for o caso, primeiro realize a exclusão local:
    - `git tag -d <nome da *tag*>`
  - depois a exclusão no seu *remote*:
    - `git push --delete origin <nome da *tag*>`

Comando básico do Git - Ignorando arquivos

- Para que o Git não adicione automaticamente ou até mesmo mostre como não rastreado algum arquivo em específico.
  - Exemplos:
    - Você tem uma classe de arquivos que não deseja que o Git adicione automaticamente
    - Arquivos de log ou arquivos produzidos pelo sistema de construção
- Criar um padrão de listagem de arquivos para corresponder aos nomeados
  - `.gitignore`
  - `$ cat .gitignore`
  - `*.[oa]` //ignorar os arquivos que terminam em ".o" ou ".a" - objetos e arquivos que podem ser o produto da criação do seu código
  - `*~` //ignorar os arquivos que terminam em ".o" ou ".a" - objetos e arquivos que podem ser o produto da criação do seu código

Comando básico do Git - Ignorando arquivos

As regras para os padrões que você pode colocar no arquivo .gitignore são:

- Linhas em branco ou linhas iniciadas com # são ignoradas.
- Os padrões glob padrão funcionam e serão aplicados recursivamente em toda a árvore de trabalho.
- Você pode iniciar padrões com uma barra (/) para evitar recursividade.
- Você pode finalizar padrões com uma barra (/) para especificar um diretório.
- Você pode negar um padrão iniciando-o com um ponto de exclamação (!).

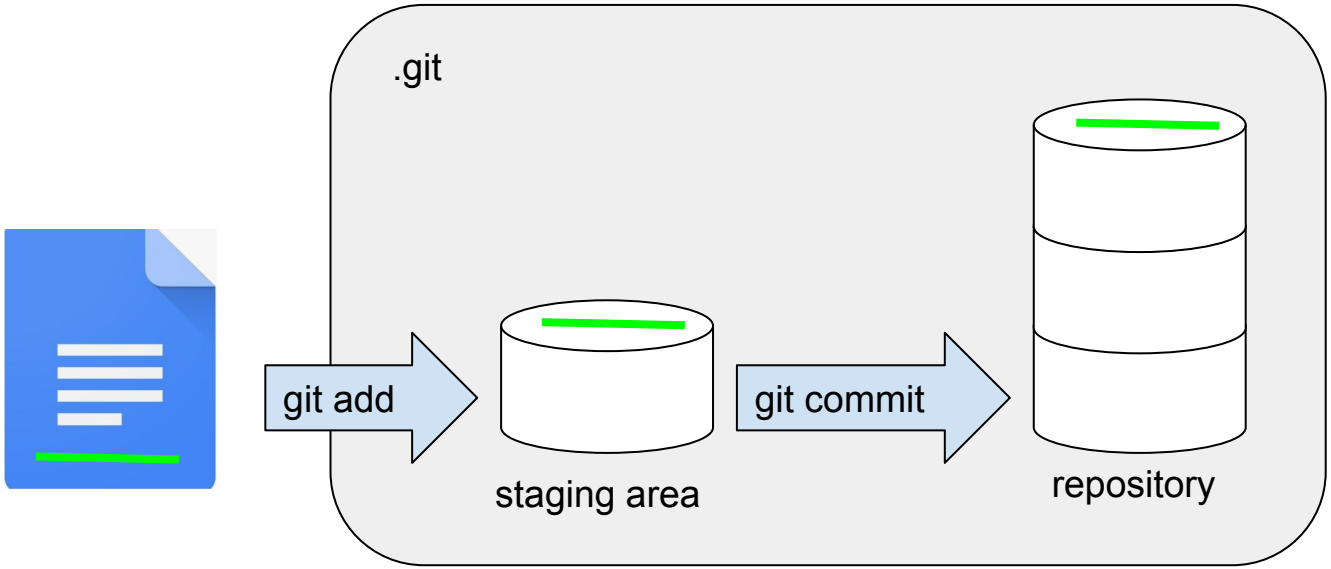
Comando básico do Git - Removendo arquivos

- Para remover um arquivo do Git, você deve removê-lo dos arquivos rastreados (com mais precisão, removê-lo da sua área de preparação - ***Staging Area***) e depois confirmar.
- Utilizar o seguinte comando:
  - `git rm <nome do arquivo>`

Estado dos arquivos

- Não monitorado (untracked)
- Modificado (modified)
- Preparado (staged)
- Consolidado (commited)

Estado dos arquivos



Exercício 1 - Individual em aula

- Baixar o repositório
  - git clone git@gitlab.com:ricardo_kondo/disciplina-gcs-2020-1.git
  - o aluno será inserido em aula (29/04)
- Crie uma branch seguindo o seguinte padrão:
  - nome_aluno/ex1
- Na sua branch, crie o seguinte arquivo.txt
  - seu_nome.txt
- Adicione no arquivo uma linha com um comentário
- Faça o commit e o push das alterações
- Vamos verificar no GitLab

Exercício 2

- Irei adicionar novos arquivos no repositório
- Todos devem dar git pull para atualizar o repositório
- Veja o que aconteceu.

Comando básico do Git - Navegar no histórico

- Permite ver como um arquivo ou todo o repositório estava em um determinado commit
  - git checkout <commit> <file>

- Pode ser útil para comparar trechos de códigos, no qual é possível verificar em commits anteriores.

Comando básico do Git - Desfazer alterações

- Irá desfazer todas as alterações que não estejam no *Stage* desde o último *commit*
  - git checkout -- <path_or_file>

- Desfazer as alterações desde o último *commit* incluindo o *Stage*
  - git checkout HEAD -- <path_or_file>

Comando básico do Git - Desfazer alterações

- Irá criar um novo *commit* que desfaz as alterações do *commit* especificado
  - git revert <*commit*>

- Útil para desfazer um *commit* antigo

Comando básico do Git - Desfazer alterações

- Resetar o repositório para um determinado *commit*
  - git reset <*commit*>
  - git reset HEAD~1 // volta 1 commit

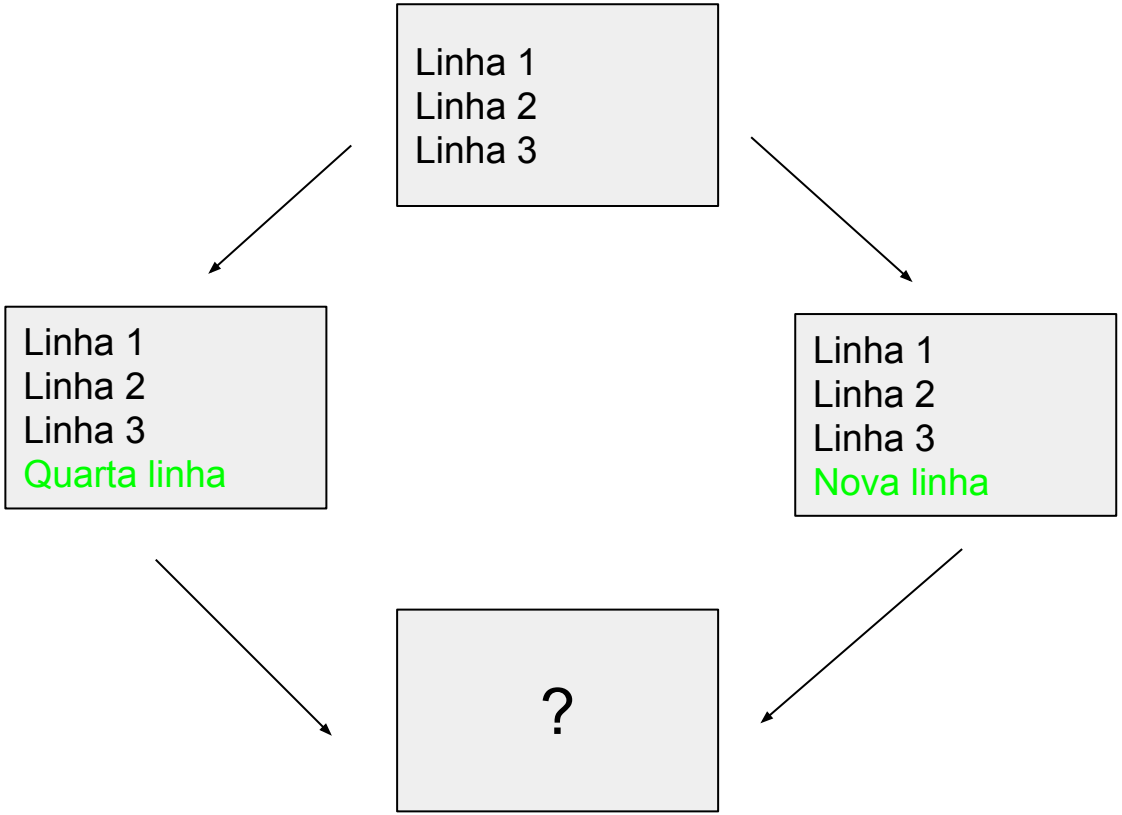
- Resetar e remover todas as alterações
  - git reset -- hard <*commit*>
  - Cuidado ao usar!!! Não usar se já estiver publicado, no caso, se você enviou para o repositório.

- Útil para desfazer últimos *commits*

Conflitos

- Conflitos podem acontecer ao unirmos alterações
- Acontece quando versões diferentes possuem as mesmas linhas nos mesmos arquivos editadas diferentes
- O git identifica os conflitos e fica aguardando a solução deles
- Ao resolver os conflitos deve ser feito um *commit*

Conflitos


Linha 1
Linha 2
Linha 3
Linha 1
Linha 2
Linha 3
Quarta linha
Linha 1
Linha 2
Linha 3
Nova linha
?

Exercício 3

- Crie um conflito no repositório
- Eu vou editar o mesmo arquivo que você
- Você deve fazer o pull, resolver o conflito e fazer o push das suas alterações.
- Eu vou ver que sua alteração foi adicionada junto à alteração minha.