- Para entrar no ambiente git:
Abrir a pasta pelo windows explorer
Clicar com o botão direito e escolher "git bash here"

CONFIGURAÇÃO

- Configurar nome e email
$ git config --global user.name "davibelo"
$ git config --global user.email "davibelo@gmail.com"

- Visualizar configurações
$ git config --list

- Criar abreviações de comandos - alias
$ git config --global alias.[nome do alias] [comando a ser abreviado]
ex: abreviar $ git status para $ git s
$ git config --global alias.s status

- Remover alias
$ git config --global --unset alias.[nome do alias]

INICIAR

- Iniciar repositório
(Dentro da pasta do projeto)
$ git init

STATUS

- Ver status do repositório
$ git status

INDEXAR

- Adicionar arquivos no índice do repositório (stage)
(Para um arquivo)
$ git add [filename]
(Para todos os arquivos da pasta)
$ git add .
ou
$ git add --all

SALVAR UMA VERSÃO (COMMIT) NO REPOSITÓRIO LOCAL

- Salvar os arquivos indexados no repositório local
$ git commit -m “[meu comentário]“

- Adicionar ao stage e comitar ao mesmo tempo
$ git commit -a -m “[meu comentário]“

- Adicionar uma modificação no commit atual no repositório local
$ git commit --amend

obs: no editor de texto, inserir o comentário, apertar ESC, inserir “:” e depois “wq” para escrever e gravar a mensagem do commit.

LISTAR DIFERENÇAS E VERSÕES

- Listar as alterações entre o projeto commitado e o diretório de trabalho
$ git diff

- Listar as alterações entre o projeto commitado o que está indexado (staged)
$ git diff --cached
ou
$ git diff --staged

- Listar todo o histórico do projeto
$ git log

- Listar todo o histórico, usando somente uma linha para cada commit
$ git log --oneline

RE-CARREGAR UMA VERSÃO NA PASTA DE TRABALHO

- Carregar um certo commit para a pasta de trabalho
$ git checkout [código do commit]
(obs: basta colocar os primeiros caracteres do código, pois ele é único)

- Re-carregar o commit mais avançado
$ git checkout master

- Carregar um certo arquivo conforme está salvo no commit, sobrescrevendo o que está atualmente no diretório de trabalho (também pode ser usado para restaurar um arquivo apagado)
$ git checkout [file name]

DESCARTAR ALTERAÇÕES

- Mostrar os arquivos que serão alterados caso se queira restaurar o commit
$ git reset

- Restaurar todos os arquivos conforme o commit
$ git reset --hard
obs: se tiverem arquivos ainda não indexados, esses continuarão no diretório mesmo restaurando o commit

- Remover as alterações do último commit no repositório local
$ git reset --hard HEAD~1
obs: $ git reset --hard HEAD~n, substituir o n pela qtdades de commits que se deseja apagar

- Remover arquivos ainda não indexados da pasta de trabalho
$ git clean -f
outra forma de fazer é indexar primeiro todos os arquivos e depois restaurar o commit
$ git add .
$ git reset --hard

IGNORANDO ARQUIVOS NO COMMIT

- Listar arquivos para ignorar no commit
Criar um arquivo “.gitignore”
Dentro do arquivo colocar os arquivos ou pastas que devem ser ignorados
Exemplo:
*.bmp
minha_pasta/
Outros exemplos de gitignore podem ser encontrados no github
https://github.com/github/gitignore

CLONAR REPOSITÓRIO

- Clonar um repositório localmente
Executar o comando no git bash na pasta raiz que contém os projetos
$ git clone [pasta do projeto a ser clonado] [pasta do clone]

- Clonar um repositório em servidor
Executar o comando no git bash na pasta raiz onde se deseja deixar o repositório local
$ git clone [URL do projeto]

INTERAÇÃO DO REPOSITÓRIO LOCAL COM O REPOSITÓRIO REMOTO

- Visualizar a URL do repositório remoto associada ao projeto
$ git remote -v

- Enviar as últimas alterações de um repositório local para o repositório remoto
-- Se for a primeira vez:
Criar um novo repositório no site do github
Adicionar a URL do repositório na configuração do repositório local e fazer o push como o commit master
$ git remote add origin [url do repositório no gihub]
$ git push -u origin master
-- Se não for a primeira vez:
$ git push

- Baixar e combinar as últimas alterações do repositório remoto no repositório local
$ git pull

- Somente baixar os commits do repositório remoto no repositório local
$ git fetch
Para verificar os commits baixados, executar:
$ git log origin/[nome da branch em questão]
Para alterar os arquivos locais, será necessário executar:
$ git merge

DERIVAÇÃO DE REPOSITÓRIO - FORK

- Para copiar um repositório de terceiros para sua conta e poder desenvolver em cima
Executar o comando “fork” pelo site do github
Verificar o repositório copiado para sua conta e trabalhar com essa URL

- Para solicitar que um repositório de terceiros incorpore as suas modificações
Depois de executar o “fork”, modificar e commitar o seu código no seu repositório...
Criar um “pull request” para o repositório original
Atentar se o github indica se as alterações são possíveis de um merge

- Para aceitar mudanças sugeridas por terceiros
Dentro do repositório do github, clicar na aba de “pull request”, analisar a solicitação e
clicar em “merge pull request”.

COMANDOS DE RAMIFICAÇÕES (BRANCH)

- Criar uma branch
Branchs são ramificações de desenvolvimento que depois podem ser juntadas (merge) ao projeto master ou não.
Exemplo: Num projeto de site, posso fazer uma branch para desenvolver um frontend, outra para desenvolver outro frontend alternativo e outra para o backend. Posteriormente, posso escolher qual frontend vou juntar (merge) no projeto master, juntamente com o backend.

- Listar as branchs do repositório
$ git branch
(obs: a branch que vc está checkado aparece marcada com um *)

- Listar as branchs aplicando uma busca
$ git branch | grep [texto a ser buscado]

- Criar uma branch nova
$ git branch [nome da branch]

- Mudar para uma branch
$ git checkout [nome da branch]

- Criar uma branch e já mudar pra ela
$ git checkout -b [nome da branch]

- Enviar uma branch para o servidor pela primeira vez
$ git push -u origin [nome da branch]
(a partir daí, para enviar novamente basta $ git push estando checado na branch)

- Usar uma branch criada no repositório remoto por outra pessoa
Atualizar o repositório local
$ git pull
Necessário fazer um checkout na branch nova para que essa passe a ser listada localmente
pelo comando $ git branch
$ git checkout [nome da branch]

- Remover branch localmente
$ git branch -d [nome da branch]
(você não pode estar checado na branch que deseja apagar)
Para forçar a remoção, caso algum erro
$ git branch -D [nome da branch]

- Remover branch no repositório remoto
$ git push --delete origin [nome da branch]

- Retornar para uma branch apagada só localmente
Basta fazer o checkout nela
$ git checkout [nome da branch]

- Renomear uma branch no repositório local
Estando checado na branch
$ git branch -m [novo nome da branch]
Não estando checado na branch
$ git branch -m [nome atual da branch] [novo nome da branch]

- Renomear uma branch no repositório remoto
Não é possível fazer diretamente, necessário:
Baixar as branchs do repositório remoto
$ git pull
Renomear localmente
$ git branch -m [nome da branch] [novo nome da branch]
Apagar a branch do repositório remoto
$ git push --delete origin [nome da branch]
Enviar a branch com o novo nome
$ git push -u origin [novo nome da branch]

- Combinar (merge) branchs

-- Usando somente o git bash
Estar checado na branch raiz, geralmente a master e executar
$ git merge [nome da branch a combinar]
(se não tiver conflito, o merge já executa um novo commit e depois basta fazer o push)
(se tiver conflito, este(s) será(ão) indicado(s) quando tentar fazer o $ git push.
assim, será necessário, fazer o $ git pull para atualizar o repositório local, ir nas linhas de código marcadas pelo git, remover as que não se desejam e depois, adicionar ao stage, commitar e repetir o comando de push)

-- Usando uma ferramenta de merge como kdiff3
Baixar a ferramenta de merge
Adicionar a ferramenta na configuração do git
$ git config --global --add merge.tool kdiff3
Adicionar o path da ferramenta
$ git config --global --add mergetool.kdiff3.path “C:/Program Files/KDiff3.exe”
Deixar trustexitcode false
$ git config --global --add mergetool.kdiff3.trustexitcode false
Para abrir a ferramenta
$ git mergetool
A ferramenta vai mostrar os conflitos e você deve escolher como resolver clicando com o botão direito em cima do conflito no código

-- Usado “pull request”
Outra forma de combinar (merge) os códigos é fazer as alterações na branch, comittar, enviar por push pro repositório remoto e depois criar um “pull request” para essa branch. Desta forma, o código será analisado e mergeado pela própria interface do github. Se tiver outra pessoa no projeto, ela pode ficar de aprovar os pedidos, atuando com um verificador.

GUARDAR MODIFICAÇÕES TEMPORARIAMENTE

- Criar uma memória das modificações permitindo alternar para outras branchs, sem perdê-las, mas sem precisar commitar
$ git stash
ou
$ git stash save -m “[comentário]”

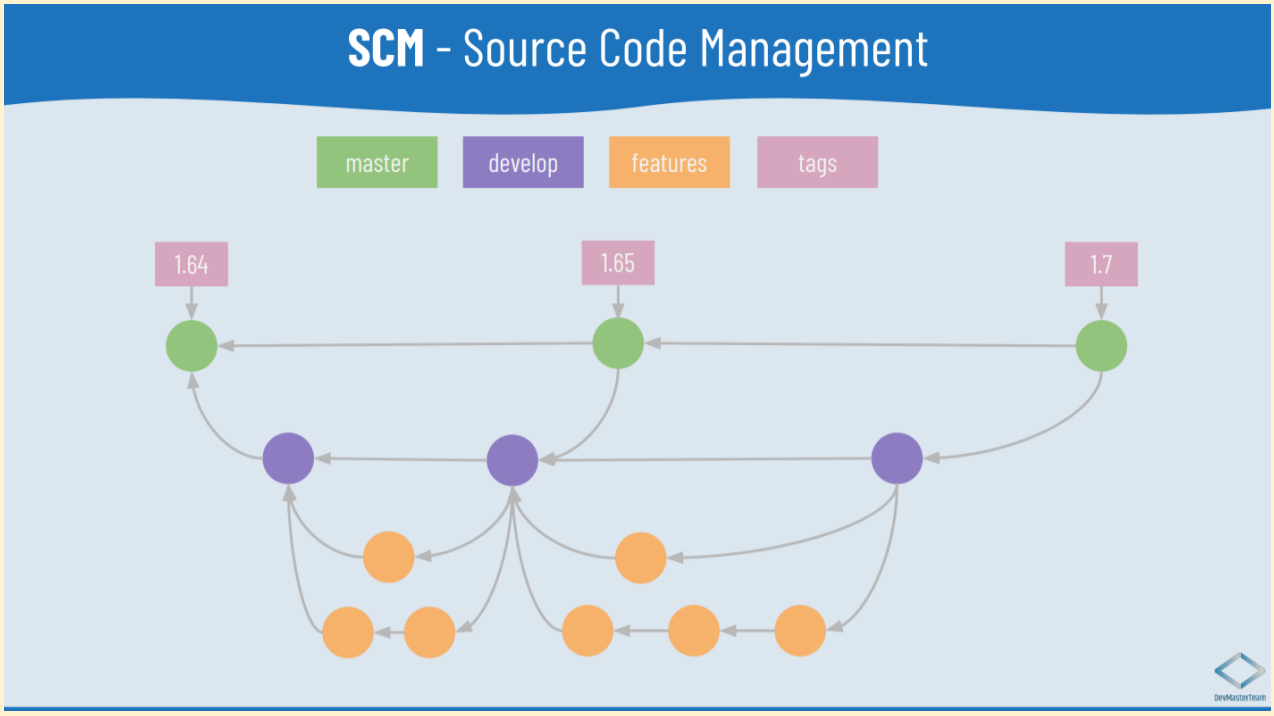
- Listar a memória stash
$ git stash list

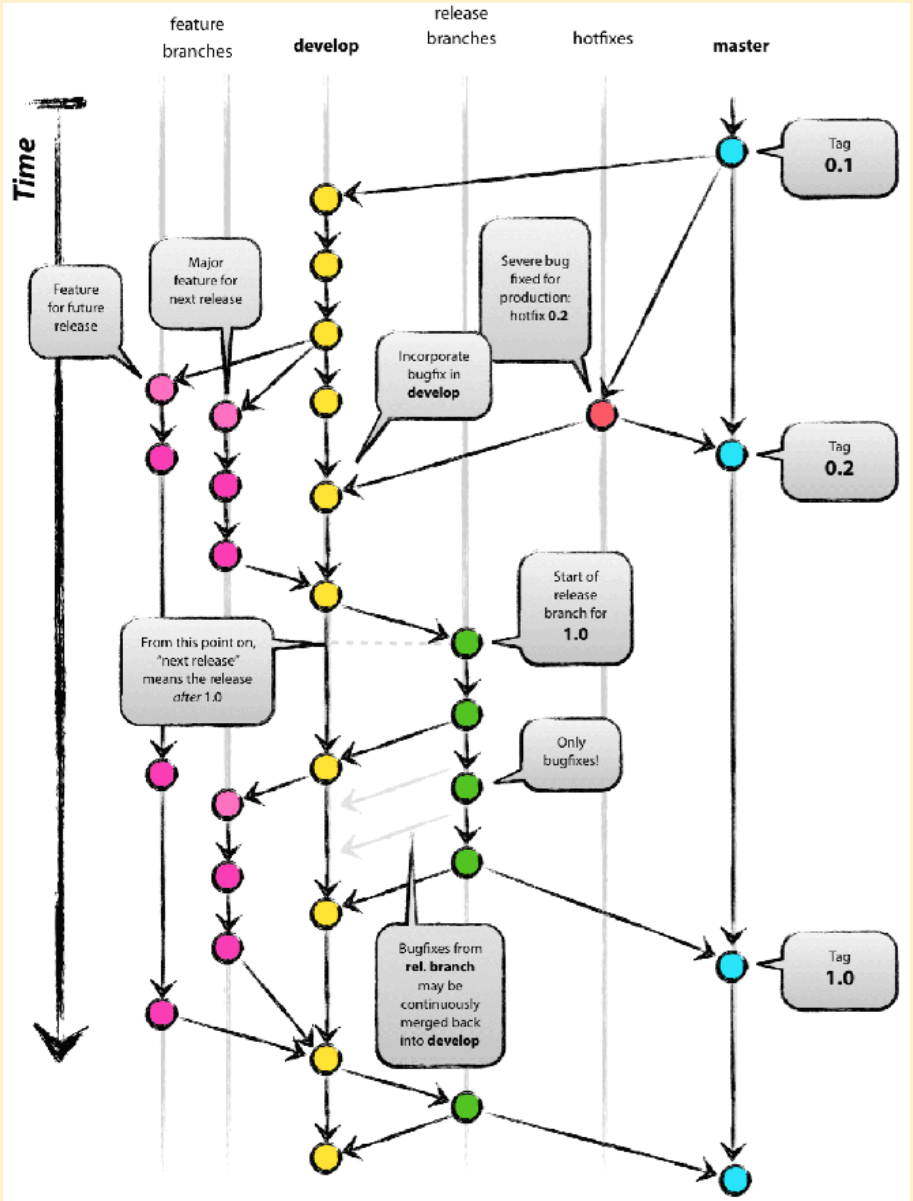
- Restaurar o código guardado na memória
-- Aplicar o último stash
$ git stash apply
-- Aplicar o último stash e já removê-lo da lista de stash
$ git stash pop
-- Aplicar um outro stash da lista
$ git stash apply [nome do stash]
-- Aplicar o último stash e já removê-lo da lista de stash
$ git stash pop [nome do stash]

- Apagar um stash
$ git stash drop [nome do stash]
ex: nome do stash = stash@{1}

SCM - SOURCE CODE MANAGEMENT

Abaixo um exemplo de gerenciamento de versões de código
Existe a branch master.
Dela é derivada uma branch de desenvolvimento (develop)
Desta branch develop são derivadas outras branchs para cada feature (função) que se deseja implementar
Posteriormente, após testes, as branchs de features são mergeadas na branch develop
Uma vez a branch develop testada, essa é mergeada na branch master, criando uma nova versão do código principal (tag)


SCM - Source Code Management
master
develop
features
tags
1.64
1.65
1.7


feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.1
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Uma tag é uma indicação que se faz em um commit da branch master para registrar um determinado marco no desenvolvimento do código.

- Criar uma tag para o commit atual
$ git tag -a [nome da tag] -m “[meu comentário]”
OBS: se quiser fazer uma tag para um commit anterior, fazer o checkout nesse commit

- Criar uma tag para um outro commit
$ git tag -a [nome da tag] [nome do commit] -m “[comentário]”

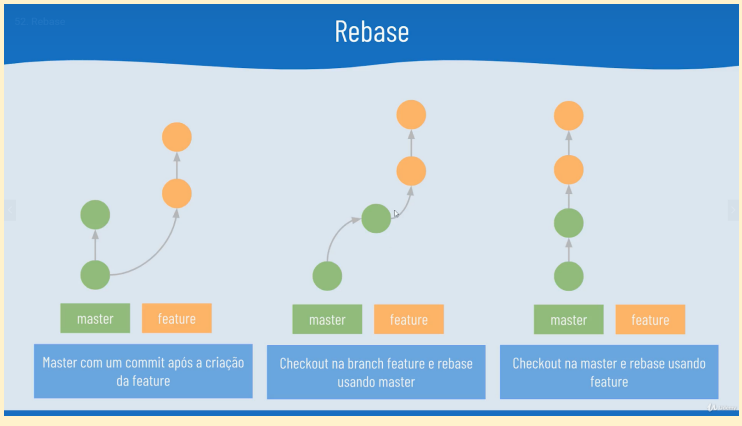
- Usar uma tag para fazer checkout no respectivo commit
$ git checkout [nome da tag]

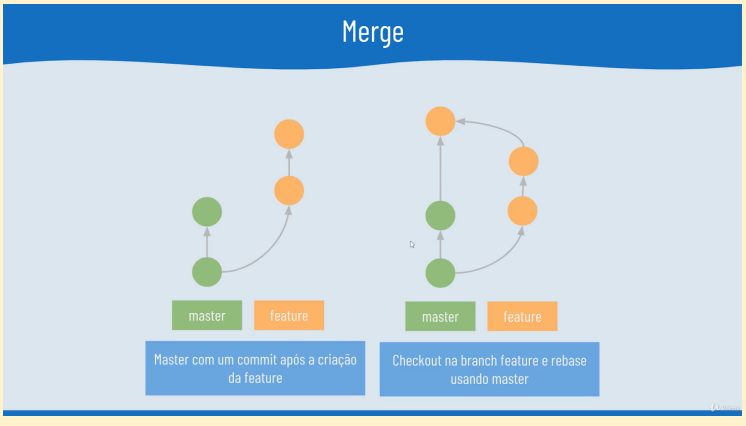
- Listar as tags existentes
$ git tag

- Enviar tag para o servidor
$ git push origin [nome da tag]

- Para remover tags
Usar os mesmos comandos de remover branchs

- Fazer com que um merge com uma branch fique linear
Estando checkado na branch
$ git rebase master
Estando checado na master
$ git rebase [nome da branch]
Abaixo um exemplo gráfico da diferença entre usar o rebase e o merge





OUTRAS FUNÇÕES DO GITHUB
obs: o bitbucket é um site de repositório semelhante ao github

- Relatar um erro no código de um repositório
Criar uma “issue” no github do repositório

- Acrescentar labels nos issues
Se você for o dono do repositório, poderá colocar labels nos issues para classificá-los

- Fazer um planejamento de correção de bugs
Se você for dono do repositório poderá criar milestones para indicar as próximas versões do código

Ir nos issues e escolher qual versão esse problema será resolvido

- Fazer um “pull request” para resolver um issue
Na parte de comentários do “pull request”, colocar “#”, procurar pelo issue e selecioná-lo.
Se o dono do repositório aceitar o “pull request”, o issue vai ser automaticamente encerrado.

- Criar um arquivo readme.md para o repositório
Basta usar a interface no site, entretanto, o conteúdo do texto não é um txt normal, possui uma codificação estilo html. Para facilitar uma criação de um novo código, pode ser usado o site abaixo:
https://dillinger.io/

FERRAMENTAS GRÁFICAS

- Sourcetree
https://www.sourcetreeapp.com/

- GitKraken
https://www.gitkraken.com/