# Git "the stupid content tracker"

Autor: Jean Carlo Machado Email: [contato@jeancarlomachado.com.br](mailto:contato@jeancarlomachado.com.br)

* Pra que git serve?
* Historia
* Lista de comandos
* Conceitos do git

## História

## Motivo

Git foi criado em 2005 por Linus Torvalds, o criador do Linux. A motivação de criar o Git foi porquê o CVS anterior (SVN) era muito lento para comportar o trabalho do kernel.

Alguns significados para o termo: - "global information tracker" - "goddamn idiotic truckload of sh\*t"

##### Pontos-chave no design:

* Velocidade
* Design Simples
* Suporte a desenvolvimento não-linear (branches)
* Totalmente distribuído
* Capaz de lidar com projetos gigantes

## Pra que serve

## Git serve para recuperar informação

git log  
rm -rf docs  
browser http://localhost:8000  
git reset --hard HEAD  
browser http://localhost:8000

## Git serve para versionar informações

##### Criando

echo "Historia do git" >docs/historia.md  
echo "Historia" >>docs/index.md  
git add .  
git commit -m "adicionado arquivos sobre historia"  
echo "Git foi criado em 2005 por Linus Torvalds, o criador do Linux" >>docs/historia.md  
git commit -a -m "detalhes sobre a historia"

##### Desfazendo

git checkout HEAD~1 docs/historia.md  
git checkout HEAD docs/historia.md

Git também serve para:

* Trabalhar em múltiplas tarefas concorrentemente
* Trabalhar com múltiplas equipes concorrentemente
* Pode-se versionar todo tipo de informação: projetos de software, livros, tcc's, etc.

# Comandos

## Lista de Comandos

Git conta com vários comandos, a lista completa se encontra em /lib/git-core.

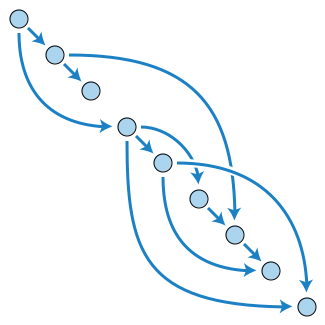
Para mais detalhes do que como cada um opera pode utilizar

man git commando  
git commando --help

# Conceitos

## Histórico em grafo

Os commits no git são estruturados em forma de grafo



Grafo acíclico

## HEAD

É a última versão da branch atual. Utilizada pelo comando git commit para ser o pai do novo commit.

## Áreas de armazemanto do git

Git tem 3 áreas de armazenamento principais

* Staging (index)
* Repositório Local
* Repositório Remoto



Comandos de trasporte e áreas de armazenamento

## Merges

##### Fast-forward merge

Acontece quando o histórico do merge está a frente do HEAD atual, então move-se apenas o ponteiro da branch.

##### Recursive merge

Também conhecido como merge de 3 vias. Encontra-se o último commit ancestral(CA) das duas branches(B1,B2). Faz-se um diff de B1 com CA e B2 com CA

Navega-se entre todos os blocos identificados nas diff's. Se os dois lados introduziram a mesma modificação no mesmo lugar, aceita-se ela. Se uma branch introduz uma modificação e a outra não mexe na região adiciona-se a modificação no final. Se as duas adicionam modificações no mesmo lugar, mas elas conflitam, marca-se o local para ser ajustado manualmente.

## Packfiles

São arquivos "otimizados" para remover tamanho desenecessários do sistema.

$ git verify-pack -v .git/objects/pack/pack-978e03944f5c581011e6998cd0e9e30000905586.idx

Todo commit é uma hash sha1 e muda conforme o pai for reescrito.

## Inicializando um projeto

## Init

Para inicializar um projeto git

São criados arquivos na pasta .git com os dados versionados.

git init $DIRETORIO

Para uma lista dos arquivos criados pelo git:

cd /tmp  
git init foo  
find .

## Config

git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com

## Criando Histórico

## Add

Git add adiciona arquivos a uma área temporária para compor um commit (staging).

git add $ARQUIVO|$DIRETORIO

## Commit

Cada mudança no histórico de um projeto é representado por um commit. git show mostra o último commit.

Para uma lista completa dos commits use: git log.

git commit

O commit transfere as informações para o repositório local.

Boas práticas de nomenclatura de commits https://www.alexkras.com/19-git-tips-for-everyday-use/#good-commit-message

## Status

git status  
git status -s #versão resumida

## Patches

## Format patch

Patches são mudanças que podem ser transferidas de um repositório para outro:

Para criar um patch do último commit:

git format-patch master

## Appy

Para aplicar um patch:

curl -L https://goo.gl/p1LEc7 -o 0001-historia.patch  
git apply 0001-historia.patch

## Diff e show

O resultado de git show e git diff também são comapatíveis com patches do git.

git diff > my\_patch.patch  
git show > my\_patch.patch  
git diff --cached > my\_patch.patch

## Criar um patch de uma branch

git format-patch master

## Branches

Uma branch é uma linha de trabalho independente. Podem ser usadas para diversos propósitos.

Pode-se ter uma branch para:

* experimentar uma tecnologia nova;
* uma branch para um bug-fix;
* outra para o trabalho do sprint;

A branch padrão no git é a master. Para listar todas as branches use: git branch

git commit "informações sobre branches no master"  
git checkout -b "recursos\_adicionais" #cria outra linha de trabalho  
curl -L https://goo.gl/XlMc3m -o 0001-recursos\_adicionais.patch  
git apply 0001-recursos\_adicionais.patch

O comando checkout serve para criar a branch e entrar nela.

##### Merge

Merge mescla o conteúdo de branches

git commit -m "mais modificações"  
git merge recursos\_adicionais  
git log

##### Fluxos de trabalho

Existem vários fluxos de trabalho de branches.

* Githubflow
* git flow
* branch por tópico

Mais sobre fluxos de trabalho

* https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows#\_distributed\_git
* https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows

## Merges

Fast-forward: move o ponteiro da master para o último commit da branch.

git merge --fast-forward

## Resolvendo com nossas alterações

git pull origin master -X ours

## Mostrando o conteúdo de um commit

git show commitId

## Desfazendo

##### O que ainda não está comitado

git reset --hard HEAD

##### Remover arquivos criados mas não versionados

git ls-files --others --exclude-standard | xargs rm

##### Remover do staging

git reset HEAD $FILE

##### Revert

Desfaz um commit criando um novo commit. Recomendado em branches públicas onde o histórico não pode ser reescrito.

git checkout hotfix  
git revert HEAD~2

##### Trocar a última mensagem de commit

git commit --amend

##### Desfazendo últimos 3 commits - sem desfazer o código

git reset --soft HEAD~3

##### Desfazendo últimos 3 commits - desfazendo o código

git reset --hard HEAD~3

## Remotos

Git é distriuído.

## Remoto no filesystem local

cd /path/to/git-docs  
cd ..  
git clone git-docs other-git-docs  
cd other-git-docs  
git config user.name "other user"  
git config user.email other.user@gmail.com  
  
  
//do some work  
cd ../git-docs  
git remote add other ../other-git-docs  
git merge other/master

## Remoto online

##### Listar os repositórios remotos

git remote -v

##### Git remote add origin

git remote add origin git@github.com:compufour/compufacil.git  
git remote add origin https://github.com/user/repo.git

##### Mandar para o repositório

git clone https://github.com/JeanCarloMachado/git-docs  
git push origin new\_branch

##### Baixar do repositório

git pull --rebase

## Setando remoto e branch padrões

Permite usar apenas git push, ao invés de git push origin master.

git branch --set-upstream-to myfork/master

## Github

GitHub é uma rede social.

## Permite

* Criar repositórios
* Colaborar para repositórios existentes:
  + Criando issues
  + Submitando pull requests

## Configurar ssh keys

## Outros Serviços

##### Githubio Pages

Hospedar um site com seu usuário.

##### Gists

Tipo pastebin

## Customizando

As configurações globais ficam em

~/.gitconfig

Em repositórios

.git/config

## Cores

git config --global color.ui true

## Setar o editor padrão

git config --global core.editor vim

## Trocar template de mensagem de commit

git config --global commit.template vim

## Pager

git config --global core.pager 'less'

## Aliases

git config --global alias.co checkout

##### Logs

git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"  
git config --global alias.lol "log --graph --decorate --pretty=oneline --abbrev-commit --all"  
git config --global alias.mylog "log --pretty=format:'%h %s [%an]' --graph"

## Ferramentas Auxiliares

# Ferramentas & Produtividade

## Stash

Para salvar trabalhos temporários

git stash  
git stash apply

##### Visualizar o stash

git stash show -p

## Gitk

Uma interface nativa para quem gosta de interfaces gráficas

## Bash aliases

alias amend='git commit --amend'  
alias b='git branch'  
alias branch='git branch'  
alias check='export PREV\_BRANCH=`cb` && git checkout'  
alias cm='git commit'  
alias cm='git status && git add . && git commit -m "'  
alias g='git'  
alias ga='git add'  
alias gac='git-add-commit'  
alias gb='git branch'  
alias gbi='git bisect'  
alias gc='git commit'  
alias gck='git checkout'  
alias gckm='git checkout master'  
alias gckm='git checkout master'  
alias gdf='git diff'  
alias gdfn='git --no-pager diff --name-only'  
alias gf='git fetch'  
alias gfc='git fetch && git checkout'  
alias gh='git rev-parse HEAD' -r'  
alias gl="git log --pretty=oneline"  
alias gm='git merge'  
alias gp='git push'  
alias gr='git remote'  
alias grh='git reset --hard HEAD'  
alias grhh='git reset --hard HEAD'  
alias gs='git status -s'  
alias gt='git tag'  
alias lcb='git pull origin `cb`'  
alias lom='echo "Consider using: [git pull --rebase] instead" ; git pull origin master'  
alias lr='git pull compufacil master --rebase'  
alias pcb='git push origin `cb`'  
alias status='git status'

##### Funções

current\_branch() {  
 git branch 2> /dev/null | grep "\*" | cut -d" " -f2  
}  
  
commit\_diff () {  
 git diff $1~ $1  
}  
  
not\_committed\_files () {  
 git ls-files --others --exclude-standard  
}  
  
commit\_count () {  
 git log --pretty=oneline | wc -l  
}  
  
  
last\_diff\_file() {  
 last\_diff=${2:-1}  
 file\_name=$1  
 git log -p --follow -$last\_diff $file\_name  
}

## Criando um servidor git

## Criando um server remoto

##### Remoto no filesystem local

cd /path/to/git-docs  
cd ..  
git clone git-docs other-git-docs  
cd other-git-docs  
git config user.name "other user"  
git config user.email other.user@gmail.com  
  
  
//do some work  
cd ../git-docs  
git remote add other ../other-git-docs  
git merge other/master

## Bare

Não tem working dir.

cd /tmp/  
mkdir foo  
git init . --bare  
git clone root@45.55.247.185:/tmp/foo /tmp/foo\_clone

## Submódulos

## Criando um submódulo

git submodule add https://github.com/chaconinc/DbConnector  
git submodule add ../mainrepo.git submoduleDir  
  
git status

## Buscando os arquivos

git submodule init  
git submodule update

## Inicializando os submódulos em um repositório já configurado

git clone --recursive

## Mantendo atualizado

git config -f .gitmodules submodule.jeancarlomachado.github.io.branch master  
  
git submodule update --remote

## Foreach

git submodule foreach "git add . ; git commit -m 'added assets'"

## Push

git config push.recurseSubmodules on-demand  
git push

## Diff/Log

git $COMANDO --submodule

## Operadores Relativos

## Operadores relativos

HEAD@{5}  
HEAD@{yesterday}  
HEAD@{2.months.ago}

##### Sintaxe alternativa

git rev-parse 34ab345c70~2  
git rev-parse HEAD~1  
HEAD~5

##### Topo da branch atual

HEAD  
HEAD~ == HEAD~1

##### Referência a ancestrais

git rev-parse HEAD^  
git rev-parse 2aad^^

^ significa o primeiro pai do primeiro pai do primeiro pai

###### Para funcionar o (^) nos zsh e bash sem necessitar escapar

setopt NO\_NOMATCH  
setopt NO\_EXTENDED\_GLOB

## Reescrevendo Histórico

## Rebase

Coloca o histórico da branch atual no topo da branch target.

git rebase target

## A regra de ouro do rebase

https://www.atlassian.com/git/tutorials/merging-vs-rebasing?section=the-golden-rule-of-rebasing

## Squash

git rebase -i

Ou

git merge --squash feature\_branch

## Editar um commit anterior

* Rebase no commit e marcar ele como editado
* Comitar as alterações
* Fazer um fix up no commit com seu pai

git rebase -i HEAD~3

## Criando dois commits a partir de 1

rebase iterativo com edit  
git reset HEAD^  
... add commit 2x  
rebase --continue

## Removendo arquivos do último commit

git rm file\_name

## Debug

## Blame

Mostra as últimas alterações de cada linha de arquivo com o autor.

$ git blame -L 141,153 simplegit.rb

Git não guarda renames explicitamente. Quando passado o -C git tenta identificar de onde o código veio.

$ git blame -C GITPackUpload.m

## Bisect

Ferramenta para descobrir qual commit quebrou uma feature.

git bisect start  
git bisect bad commitid  
git bisect good commitid

git bisect reset

Automate bisect:

$ git bisect start HEAD v1.0  
$ git bisect run test-error.sh

## Manutenção

## Removendo arquivos do histórico

git filter-branch --tree-filter 'rm -rf passwords.txt' HEAD.

git filter-branch --force --index-filter \  
'git rm --cached --ignore-unmatch PATH-TO-YOUR-FILE-WITH-SENSITIVE-DATA' \  
--prune-empty --tag-name-filter cat -- --all

## Fazendo garbage collection "packfiles"

git reflog expire --expire-unreachable=now --all  
git gc --prune=now

## Removendo branches já trabalhadas

git branch --merged | egrep -v "(^\\*|master|dev)" | xargs git branch -d

## Recuperando dados

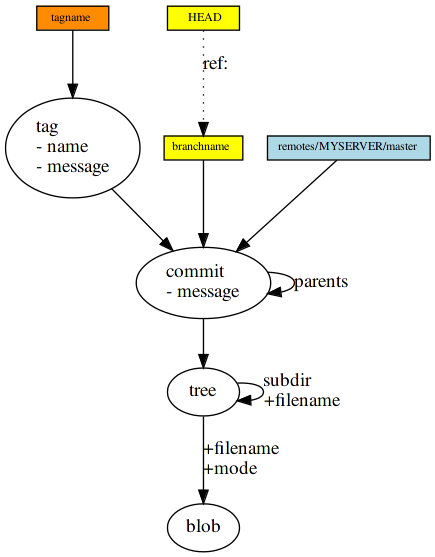
Git só deleta objetos quando você faz um git gc.

## Encontrando objetos soltos

git fsck --unreachable

## Pumbling

Pumbling é o complemento da "porcelain". Que é um conjunto de comandos que existem para compor outros comandos.



Internal Structure

## Criando objeto

echo "test content" | git hash-object -w --stdin  
  
find\_date  
cat $(run\_function object\_path d670) | zlib-flate -uncompress  
  
  
git hash-object -w test.txt

## Cat-file

##### Retornar o tipo

git cat-file -t object\_id

##### Retornar o conteúdo

git cat-file -p object\_id

## Atualizando o index

git update-index --add --cacheinfo 100644 objectHash fileName.txt  
git write-tree

## Adicionando árvores aninhadas

git read-tree --prefix=nomeDaPasta d8329fc1cc938780ffdd9f94e0d364e0ea74f579

## Comitando

echo "first commit message" | git commit-tree treeHash

##### Commit com pai

cho "commit second" | git commit-tree 0155eb4229851634a0f03eb265b69f5a2d56f341 -p 035de1de9c497bdcf9bbbdbaab8e5902e3711ce2

## Criando branch

echo "2016cc9fb2a892886946fe2ab4354ec9d40f181d" > .git/refs/heads/master

## Criando a HEAD

echo "ref: refs/heads/master" > .git/HEAD

## Criando um objeto manualmente

## Visualizar o conteúdo comprimido

cat .git/objects/28/e697e796c79e4c86fcb62e5319a43dfcf1fb83 | zlib-flate -uncompress

## Tipos de objetos

* blob
* tree
* commit
* tag

## A árvore mais recente

git cat-file -p master^{tree}

## Commit iterativo

irb  
content = "what is up, doc?"  
header = "blob #{content.length}\0"  
require "digest/sha1"  
store = header + content  
sha1 = Digest::SHA1.hexdigest(store)  
require 'zlib'  
zlib\_content = Zlib::Deflate.deflate(store)  
path = '.git/objects/' + sha1[0,2] + '/' + sha1[2,38]  
require 'fileutils'  
FileUtils.mkdir\_p(File.dirname(path))  
File.open(path, 'w') {|f| f.write zlib\_content}

## Inspecionando objetos

cat .git/objects/28/e697e796c79e4c86fcb62e5319a43dfcf1fb83 | zlib-flate -uncompress

## Comandos Menos comuns

## Rev-parse

Retorna o commit que algo está apontando.

git rev-parse master  
git rev-parse HEAD  
cat .git/HEAD

## Reflog

Mostra um log dos comandos usados no repositório. Similar ao .bash\_history (para operações locais)

git reflog  
  
ou  
  
git log -g

## Cherry-Pick

Permite aplicar um único commit de outra branch.

git cherry-pick

## Archive the repository

git archive --format zip --output /full/path/to/zipfile.zip master  
git archive master --format=tar --output=../website-12-10-2012.tar

## Bundle

git bundle create ../repo.bundle master  
git bundle unbundle ../repo.bundle master

Mantém o histórico.

## Merge de múltiplas branches

octopus merge

ours http://stackoverflow.com/questions/16208144/how-do-i-merge-multiple-branches-into-master

## Staging parcial

git add -p

## GUI para fazer merge

git mergetool

## Git grep

##### Procura pro conteúdo de commits

git grep -i "linux" $(git rev-list --all )  
git grep <regexp> $(git rev-list --all -- lib/util) -- lib/util

## Remover arquivos não versionados

git clean -f

## Últimas N alterações em um arquivo

git log -p --follow -2 Backend/module/Api/src/Api/Service/ErrorProcessor.php

## Diferença entre branches

##### Todos os commits que estão na branch develop mas não estão na master

git log master..experiment == git log ^master experiment == git  
log experiment --not master

##### Lista de objetos

git rev-parse issue\_2..master  
  
4f298373ba0f536115051593bd149539d367937d  
^2f35c9c5cc4ad413970d666d0f90b66d3cbf89c7  
<0:git-course:/home/jean/projects/git-docs:master:> git rev-parse master..issue\_2  
  
2f35c9c5cc4ad413970d666d0f90b66d3cbf89c7  
^4f298373ba0f536115051593bd149539d367937d  
<0:git-course:/home/jean/projects/git-docs:master:>

##### Que está em uma branch mas não está na outra (xor)

git log master...experimento

Está em um mas não no outro. --left-right mostra que lado cada um se encontra.

##### Log de outra branch

git log branch\_foo  
git log <branch> -- <path/to/file>

##### Que está no local mas não está na origin

git log origin/master..master --stat

##### Que está na master mas não está local

git log origin/master..master --stat

##### Who are the children of a commit

git rev-list --all --not 7ef306ff3ef3cfe694fbf3847f2c35c86067ee87^@ --children | grep "^7ef306ff3ef3cfe694fbf3847f2c35c86067ee87"

## Estudos de caso

## Estudando fluxos de projetos open-source

* master
* develop
* topic

##### Git

* Mailing list + patches

##### Kernel

* Mailing list + patches
* Mantenedores de sub-sistemas

##### PHP

* Github
* Pull-requests
* Contributing

##### Node

* Github
* Pull-requests
* Contributing
* Collaborator Guide

## Integracoes

## Integrações

Software de terceiros que agregam funcionalidade ao github [lista](https://github.com/integrations).

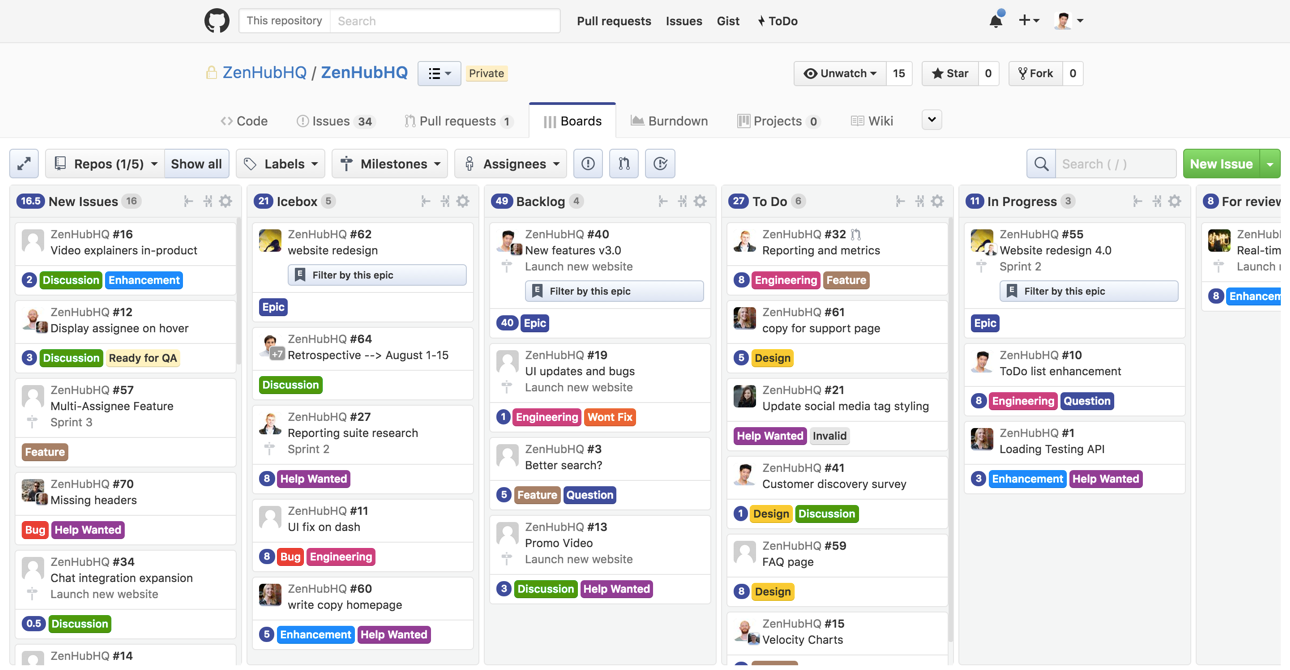
* Waffle
* Buildkite
* ZenHub

##### Codacy

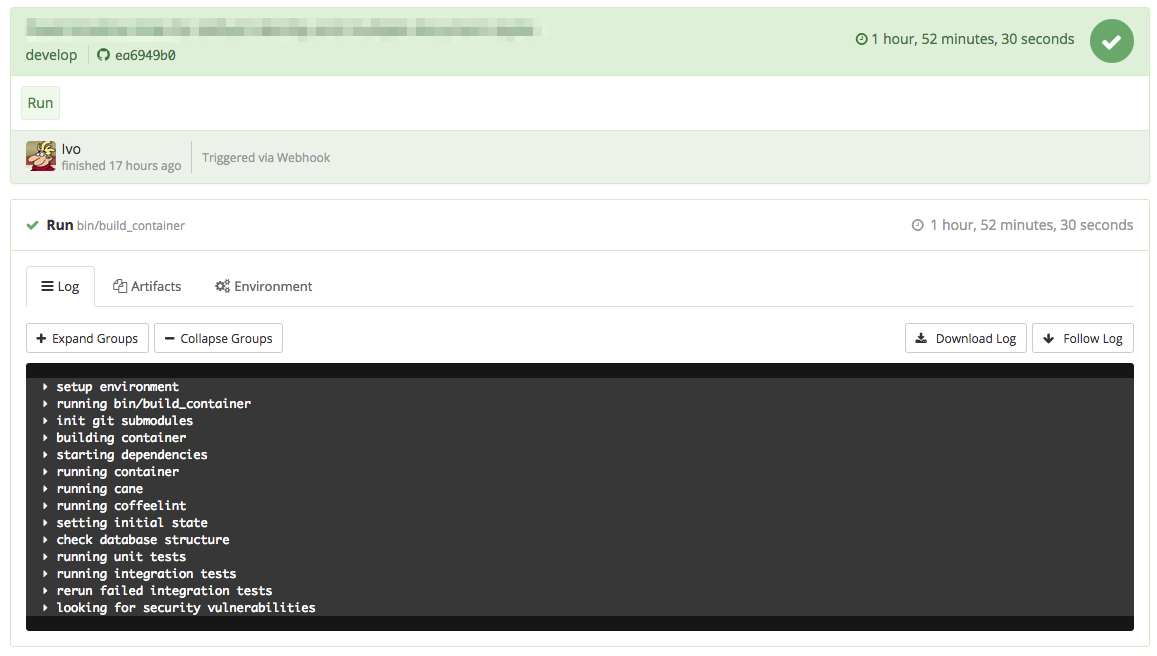
## Serviços

Executam ações quando eventos acontecem em outras ferramentas.

* Docker
* Deploy aws



zenhub



buildkite

# Recursos Adicionais

## Links

* [Referência oficial](https://git-scm.com/docs)
* [Melhor tutorial de Git](http://rypress.com/tutorials/git/index)
* [Encontrando issues no gitub](http://www.jeancarlomachado.com.br/blog/findingissuesongithub.html)
* [Git para cientistas da computação](http://eagain.net/articles/git-for-computer-scientists/)

## Livros

* Git Pro (o melhor)
* Git - Pragmatic bookshelf