codexico
Desenvolvimento web

Quando a gente compete só a gente ganha, quando a gente colabora, todos ganham.

Tutorial simples: Como usar o git e o github

By codexico on 11 de março de 2010

Ultimamente estou envolvido em vários projetos ao mesmo tempo com várias equipes diferentes, então controle de versão é essencial.

Segue um manualzinho básico para iniciar com o git, espero atualizar e complementar este passo-a-passo com mais exemplos logo.

O git serve para versionamento local, você pode compartilhar de algumas maneiras, a mais fácil é com serviços online. Neste exemplo vou usar o github, testei também o projectlocker, que dá repositórios private grátis, mas não gostei. Outro que parece legal é o Codaset, ainda não testei.

1) Instalar git

1$ sudo apt-get install git-core

É necessário gerar uma chave ssh e fazer um cadastro em algum repositório git. (Esta etapa não é exatamente sobre o git, mas sobre a segurança dos repositórios.)

Confira se vc já tem alguma chave com um "ls ~/.ssh/", se já existir uma você pode utilizá-la ou gerar uma nova:

1ssh-keygen -t rsa -C "comment"

"comment" é só um lembrete para saber do que se trata a chave, normalmente usa-se o seu nome de usuário do serviço que vai usar, por exemplo o github.

Falando nisso, está na hora de criar um usuário lá (http://github.com), vai lá que eu espero…

Depois de logado vá para https://github.com/account e clique em "SSH Public Keys" e "add another public key". A cópia da chave precisa ser exata(eu ia escrever que 'precisa ser precisa' mas é feio né), então pode-se fazer assim:

1sudo apt-get install xclip
2cat ~/.ssh/id_rsa.pub | xclip -sel clip

Aí é só colar com um Ctrl+V normal. Agora já dá para se comunicar com o github:

1ssh git@github.com

Vai aparecer "ERROR: Hi codexico! You've successfully authenticated, but GitHub does not provide shell access", não se assuste com o ERROR, o que interessa é que o github te reconheceu. Qualquer duvida tem o help do github: Generating SSH keys (Linux).

Por padrão o git vai pegar o usuário do sistema, para que seu nome de usuário do github apareça corretamente use os comandos:

```
1git config --global user.name "Your Name"
2git config --global user.email codexico@gmail.com
```

2) Criar Projeto no github

1) Podemos criar um novo projeto ou usar um existente. Para criar um novo vá até o github e no alto da página clique em "Dashboard" e depois em "New Repository".

Crie um espaço para o projeto no comnputador:

```
1$ mkdir nomedoprojeto
2$ cd nomedodiretorio
```

2) Iniciar um git neste diretório:

```
1$ git init
```

Saída do comando:

```
1Initialized empty Git repository in /nomedodiretorio/.git/
```

Deve aparecer um diretorio oculto **.git**, neste **.git** ficam as configurações que serão usadas para este projeto.

Por exemplo:

```
1$ ls .git
2branches config description FETCH_HEAD HEAD hooks index info logs
3objects refs
```

3) Adicionar o repositório, neste exemplo vou usar um que criei para este tutorial, pode ser também o repositório criado no passo 1, o endereço fica na página do projeto (neste caso https://github.com/codexico/tutorial-github):

```
1$ git remote add origin git@github.com:codexico/tutorial-github.git
```

Formato do comando:

"git remote add" adiciona um repositório ao git que foi iniciado neste diretório, "origin" é o apelido para o projeto, "git@github.com:codexico/tutorial-github.git" é o endereço do projeto.

Resultado:(apareceu a parte *[remote "origin"]*)

```
1$ cat .git/config
2  [core]
3  repositoryformatversion = 0
4  filemode = true
5  bare = false
6  logallrefupdates = true
```

```
7  [remote "origin"]
8  url = git@github.com:codexico/tutorial-github.git
9  fetch = +refs/heads/*:refs/remotes/origin/*
```

4) Baixar(pull=puxar) o projeto:

```
1$ git pull origin master
```

Formato do comando:

```
1git pull apelidoDaOrigem apelidoParaDestino
```

Saída do comando:

```
1remote: Counting objects: 52278, done.
2  remote: Compressing objects: 100% (10917/10917), done.
3  remote: Total 52278 (delta 40975), reused 51715 (delta 40669)
4  Receiving objects: 100% (52278/52278), 8.33 MiB | 189 KiB/s, done.
5  Resolving deltas: 100% (40975/40975), done.
6  From git@github.com:codexico/tutorial-github.git
7  * branch  master -> FETCH_HEAD
```

3) Usar o git

Exemplo (escolha um nome diferente para o arquivo teste):

```
1$ touch testegit
```

1) Adicionar as alterações:

– Podemos adicionar somente uma alteração:

```
1$ git add testegit
```

– Ou adicionar todas as alterações:

```
1$ git add .
```

Neste passo as alterações ainda não estão sob o controle de versão, elas somente foram adicionadas para quando der um commit.

2) Comitar as alterações:

```
1$ git commit -m "mensagem teste para o tutorial"
```

É obrigatório acrescentar uma mensagem.

Saída do comando:

```
1[master de2f5ce] teste para o tutorial
2  1 files changed, 1 insertions(+), 0 deletions(-)
3  create mode 100644 testegit
```

Agora as alterações foram adicionadas ao controle de versão. Mas ainda estão somente na máquina local.

3) Enviar(push=empurrar) as alterações:

1$ git push origin master

Saída do comando:

```
1Counting objects: 4, done.
2  Delta compression using up to 2 threads.
3  Compressing objects: 100% (2/2), done.
4  Writing objects: 100% (3/3), 288 bytes, done.
5  Total 3 (delta 1), reused 0 (delta 0)
6  To git@github.com:codexico/tutorial-github.git
7  3be4c21..de2f5ce&amp;nbsp; master -> master
```

Se durante o tempo em que fez o pull e o push outra pessoa que também participe do projeto fez alterações o push será rejeitado. Então é necessário atualizar o projeto local antes de enviar novas alterações.

1$ git fetch origin

Atualizar antes de enviar é uma boa prática a ser seguida para quem usa svn ou cvs e é obrigatória no git.

4)Pronto, confira as alterações no navegador acessando o endereço do projeto (http://github.com/codexico/tutorial-github neste exemplo).

Dica final: para que não precise digitar sempre a senha do ssh siga os passos desse link: http://help.github.com/working-with-key-passphrases/

Atualizado em 09/09/2010, mudei o repositório e adicionei instruções para gerar a chave ssh.

Posted in git, linux | Tagged git, github

Articles in this series

- Tutorial simples parte 2: git branch e merge
- Tutorial simples: Como usar o git e o github

Kleber Correia • 5 anos atrás

Parabéns pela iniciativa! Mto bom tuto!!!

Vale lembrar: é necessário criar a chave rsa... id_rsa e id_rsa.pub....
http://help.github.com/msysgit...

Abraços!!!

;)

• Responder • Compartilhar ›

codexico • 5 anos atrás

Hehehe, 2 comentários de 2 Kleber.

@Kleber1 - estamos aí pra isso, qq coisa é só chamar!

@Kleber2 - valeu, eu fiz o tutorial só para o linux né, vários já me pediram para colocar as instruções para windows também, estou pensando se atualizo este ou faço um outro post, o lance é que raramente uso windows, então vou ter q aprender e testar as ferramentas antes para poder escrever, me ajuda?

• Responder • Compartilhar ›

Kleber Correia • 5 anos atrás

Então cara e só uso linux tb! Ubuntu 10.04... não meu caso foi necessário gerar a chave.... Da uma olhada:

http://help.github.com/linux-g...

http://help.github.com/linux-k...

• Responder • Compartilhar ›

marco antonio • 5 anos atrás

Olá, parabéns pelo tutorial.
Uma sugestão: faltou dizer como se faz para recuperar um commit.
Abraço!

• Responder • Compartilhar ›

codexico Mod • 5 anos atrás

Para recuperar modificações há algumas possibilidades:

1) O arquivo foi modificado mas ainda não foi dado commit:

```
git checkout -- arquivomodificado
```

2) O arquivo foi modificado e foi dado commit:

```
git log --pretty=oneline
```

Identifique o SHA do commit que deseja que seja o atual e substitua no comando:

```
git reset --hard b8551c9354ee071349f104e3867e57ea7b220c69
```

O problema é que pode ser que vc queira de novo as modificações e é difícil recuperar (git reflog, git log -g), a melhor maneira é criar um branch para o código com as modificações e só depois excluir os commits no branch priincipal.

Vou mostrar como funcionam branchs e merges na parte 2 do tutorial.

Responder • Compartilhar ›

Andressa Agnhesi • 5 anos atrás

Muito bom o post!

Responder • Compartilhar ›

Nataliel Vasconcelos • 5 anos atrás

Otimo post! parabens

Responder • Compartilhar ›

Filipe Acácio • 4 anos atrás

Ótimo post! Parabéns!

Responder • Compartilhar ›

Régis • 4 anos atrás

Codexico, como você configura esses códigos pra ficar numerado com fundo preto no blog? Eu li o código fonte da página mas não consegui.

Responder • Compartilhar ›

codexico Mod ➚ Régis • 4 anos atrás

Eu uso o plugin CodeColorer: http://kpumuk.info/projects/wo...

2 Responder • Compartilhar ›

Thiago Marques • 4 anos atrás

Muito simples e útil seu tutorial. valeo!

1 Responder • Compartilhar ›

Cássio Nandi Citadin • 4 anos atrás

Curti o tutorial codexico. Nas várias leituras que fiz essa semana sobre o assunto, foi esse post que abriu minha mente. Esse conceito de gerenciamento local eu nunca tinha me tocado.

1 Responder • Compartilhar ›

Guilherme Souza • 4 anos atrás

Aprendi com este tutorial e já estou fazendo uso do conhecimento adquirido. Muito bacana a iniciativa. Foi muito bem detalhado e de fácil aprendizado. Parabéns!!!

1 Responder • Compartilhar ›

Quelzita2008 • 4 anos atrás

muito bom o post!! gostei e ja divulguei para alguns amigos!!

• Responder • Compartilhar ›

Leandro Gomes da Silva • 4 anos atrás

Olá,, já possuía um projeto no Git mas tive que formatar meu pc... Estou tentando continuar o projeto mas não está dando... quando dou o comando git push origin master ele retorna "Permission denied (publickey).
fatal: The remote end hung up unexpectedly"

3 • Responder • Compartilhar ›

Diego Souza • 3 anos atrás

Obrigado! me ajudou a descomplicar o Git/Github

• Responder • Compartilhar ›

Felipe Duarte • 3 anos atrás

Prezado(a),

Muito bom este artigo meu ajudou bastante..

• Responder • Compartilhar ›

Ronaldo Richieri • 3 anos atrás

Olá! Muito fera teu tutorial! Grato! e Parabéns!

• Responder • Compartilhar ›

Hélio • 3 anos atrás

pra cada projeto (php, java, whatever) que eu crio eu tenho que criar um repositório? Com subversion eu tenho um repositório só e quando preciso mexer em um projeto só baixo o diretório daquele projeto, há algo parecido no git ou tenho que clonar o repositório inteiro?

• Responder • Compartilhar ›

codexico Mod ↗ Hélio • 3 anos atrás

Hélio, para trabalhar dessa maneira teria que clonar o repositório todo mesmo. Mas é fácil criar novos repositórios e tem a vantagem de que se der algum problema, algo que ocorre menos no git do que no svn, o problema ocorre em um só.
Com um repositório para cada projeto pode-se por exemplo usar um gitignore diferente para os projetos em php e java, também o repositório não fica tão grande se for separado e se você estiver em um computador diferente é só clonar o projeto que quiser.
O processo de deploy fica facilitado se usar um repositório para cada projeto, você pode configurar por exemplo para enviar os arquivos ao servidor de testes ou ao de produção automaticamente dependendo de qual branch estiver usando.
Só vejo vantagens em usar um repositório para cada projeto.

• Responder • Compartilhar ›

robmachado • 3 anos atrás

Muito, mas muito útil mesmo !!! me ajudou bastante ... um excelente mini tutorial.

• Responder • Compartilhar ›

Vanderson • 3 anos atrás

Cara muito bom o tuto aqui em, me ensinou bastante. Vaelu!

• Responder • Compartilhar ›

Renato Tavares • 3 anos atrás

Minha dúvida pode ser boba porém acho válida, quando estou construindo um sistema sozinho e do zero, usar GIT é uma boa pedida? estou testando aqui porém acho que ele ta me deixando pouco produtivo pois sempre tenho que alterar arquivos e corrigir bugs e construir coisas. gosto do git porém ele não seria melhor para controlar um sistema depois que ele sai do desenvolvimento? ja ganha sua primeira TAG e por ai adiante?

• Responder • Compartilhar ›

codexico Mod ↗ Renato Tavares • 3 anos atrás

É Renato, muitas vezes a gente vai desenvolvendo sozinho e do zero, fazendo partes meio aleatórias do projeto, naquela fase em que na verdade a gente tá criando mesmo algo, tem muita coisa que a gente vai colocando e tirando no código muito rápido, a gente tá fazendo uma parte e descobre que precisava fazer outra antes e por aí vai, pode ser um pouco mais ágil fazer commits mais esparsos.

Como vc disse, depois que o projeto estiver mais consistente aí dá pra controlar melhor.
Nesse caso acho que a melhor regra seria "fiz algo importante que vale a pena perder algum tempinho para commitar e explicar?".

O git vai melhor quando a gente usa alguma metodologia tipo scrum ou algo assim onde a gente divide as tarefas.
Eu gosto das metodologias ágeis, mesmo quando estou fazendo algo sozinho e do zero. Na verdade eu uso pra mim mesmo GTD + pomodoro, o GTD para dividir as tarefas e o pomodoro para controlar a execução, assim eu paro de tempos em tempos para rever e que fiz e aproveito e dou um git.

• Responder • Compartilhar ›

Renato Tavares ↗ codexico • 2 anos atrás

Obrigado pela resposta. Outra vez parabéns

• Responder • Compartilhar ›

Lorena Adrian • 2 anos atrás

Dúvida: como faço para dar "check out" quando estou trabalhando com 1 solução - ou seja, tenho 1 projeto de software, vários usuários acessando esse repositório e preciso dar checkout para que outro não consiga alterar enquanto estou mexendo???

• Responder • Compartilhar ›

codexico Mod ↗ Lorena Adrian • 2 anos atrás

O git não tem um needs-lock como no svn, a ideia é essa mesma, ser um sistema distribuído em que todos podem trabalhar offline e os conflitos sejam resolvidos no merge.

Uma opção é ter um repositório central onde você pode controlar os merges, então cada desenvolvedor manda para um branch e você decide se aquela alteração entra no repositório central, como um pull-request.

Quem sabe algo como a opção core.sharedRepository pode ajudar também.

• Responder • Compartilhar ›

Daniel Albino • 2 anos atrás

Vlw bróder!!!
Estava patinando pra configurar essa parada, mas com seu tutorial eu consegui.
[]'s

• Responder • Compartilhar ›

Marcelo • 2 anos atrás

Valeu! Ajudou bastante,

• Responder • Compartilhar ›

Rafael Freitas • 2 anos atrás

Seu tutorial foi de grande utilidade para mim, muito obrigado!

• Responder • Compartilhar ›

Fabiano Maximiano • 2 anos atrás

Parabens pelo tutorial, estava levando um coro e nao sabia nem por onde começar a utilizar o GIT.
Agora vou deixar de ser junior!!!

• Responder • Compartilhar ›

Henrique Souza • 2 anos atrás

Muito bom artigo! Agora.. fiquei com uma dúvida: como faço para apagar os arquivos e diretórios do GitHub via Git?

• Responder • Compartilhar ›

Gabriel Darezzo • 2 anos atrás

Legal segui seu tutu e foi certinho

Tive problemas só pra gerar ssh
Peguei mais info no oficial:
https://help.github.com/articl...

• Responder • Compartilhar ›

Cléber Otaviano • um ano atrás

Sensacional!!!

• Responder • Compartilhar ›

Danilo Agostinho • um ano atrás

Boas dica, apenas precisam serem atualizadas. O post foi publicado em 2010, mais ainda dá pra seguir direitinho.

• Responder • Compartilhar ›

Cerebro Vasconcelos • um ano atrás

o meu ta dando erro, não consigo commitar, num tem uma parte de uma parte de configuração de token

git config --global github.token your_token

• Responder • Compartilhar ›

Caio Cutrim • um ano atrás

valeu brother :)

• Responder • Compartilhar ›

Tio nélio Do Rincão • um ano atrás

amazing brow
valeu ai negrao
curti pacas
:D

• Responder • Compartilhar ›

Jones • um ano atrás

rapaz nem sei por onde começar, não entendi nada! =/

• Responder • Compartilhar ›

Jonatan Alves • um ano atrás

Pessoal , vocês conhecem uma forma de pesquisar projetos do github e retornar com download todos os projetos de uma vez ? Por exemplo na api do git (https://api.github.com/search/... consigo retornar todos os projetos em java , com maiores estrelas e com 20 forks . mas não consigo fazer download desses projetos encontrados . Não é trivial fazer o download de um por um , seria necessário download em lote , sei ainda que o git bloqueia 20 downloads de projetos por minuto , mas não sei como fazer estes download em lote . Alguém conhece um script que faça isso ou tem alguma ideia ? Obrigado !

• Responder • Compartilhar ›

kndrio ➚ Jonatan Alves • 9 meses atrás

Faz um script dando git clone dos projetos que você encontrou.

• Responder • Compartilhar ›

husmeck • 3 meses atrás

$ git remote add origin git@github.com:codexico/tutorial-github.git

eu tentei usando esses passos mas n vejo nada no meu github

• Responder • Compartilhar ›

« apt-get install todos os compactadores e descompactadores no linux Como scannear e editar imagem no kubuntu 10.04 »

Buscar

Recent Posts

Já testou o Uber?

Eu testei e gostei! Achei super prático, tanto quanto o 99táxis, mas os carros são melhores e os motoristas bem mais educados que os taxistas.
Se vc quiser testar acesse
https://www.uber.com/invite/ubercodexico
e ganhe R$20,00 de desconto na primeira viagem!
E eu também ganho vintão de desconto ;)
Ou use o código ubercodexico
Boa viagem!

Categories

- css
- git
- html5
- javascript
- jQuery
- linux
- projetos
- rails
- twitter
- virtualização
- windows

Tags

10.04 ajax apps apt-get apt-get linux zip rar async canvas css deferreds diy downloads free gimp git github html5 html5 canvas png install javascript jQuery kubuntu layout linux listas plugin png rails3 ror ruby rubygems scanner slideshow trending topics tt tutorial twitter ubuntu

Tags

10.04 ajax apps apt-get apt-get linux zip rar async canvas css deferreds diy downloads free gimp git github html5 html5 canvas png install javascript jQuery kubuntu layout linux listas plugin png rails3 ror ruby rubygems scanner slideshow trending topics tt tutorial twitter ubuntu

Categorias

- css
- git
- html5
- javascript
- jQuery
- linux
- outros
- projetos
- rails
- twitter
- virtualização
- windows

Powered by WordPress and Social Theme by Dehradun.

Featuring Recent Posts Wordpress Widget development by YD