

Trabalhando com Git Avançado

GitHub

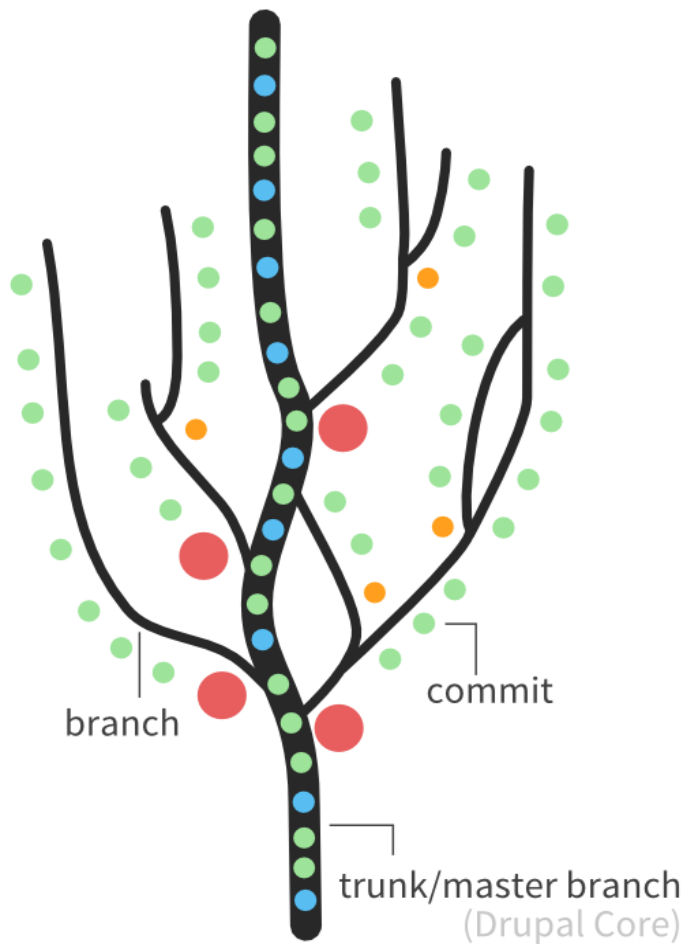


Trabalhando com Git Avançado

TÓPICOS DESTA APRESENTAÇÃO – SUMÁRIO

- .Trabalhando com Branches.
- .Merge e rebase.
- .Trabalhando com Forks.
- .Como Manter um repositório fork do GitHub sincronizado com o original, utilizando um conceito chamado upstream.
- .Como fazer um Pull Request.

Trabalhando com Branches



•A termo “branch”, em português, significa: ramo, ou galho de uma árvore. Entender esse conceito é essencial para compreender como o Git funciona.

•Quando criamos ou clonamos um repositório em nosso ambiente local, o Git cria uma branch default (algo como “ramo padrão”) chamada de master, os marinheiros de primeira viagem costumam trabalhar diretamente nela, o que não é uma boa prática pois trabalhando assim estamos desperdiçando o poder de produtividade que o Git nos proporciona.

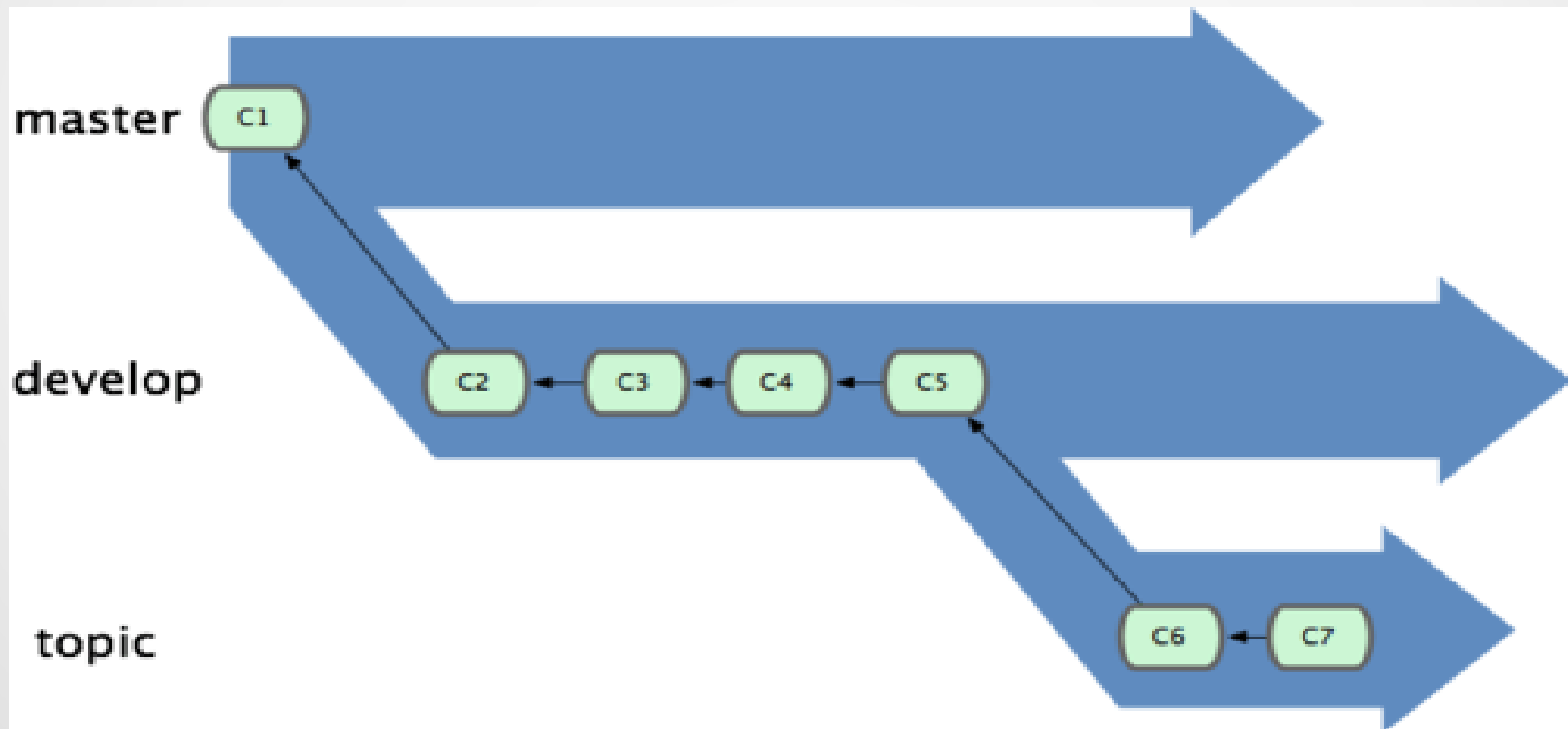
Trabalhando com Branches



- .Então qual é a forma correta trabalhar?
- .Não existe certo ou errado.
- .Existem boas práticas e para segui-las basta criar novas branches a partir da branch master, dar nome a elas e trabalhar nelas.
- .Simples né? Mas como funciona?

Trabalhando com Branches

.O fluxo de trabalho com branches funciona da seguinte forma:



Trabalhando com Branches



.A ideia é que você possa trabalhar com várias coisas em paralelo criando quantas branches for preciso e depois juntar essas implementações de novo se necessário.

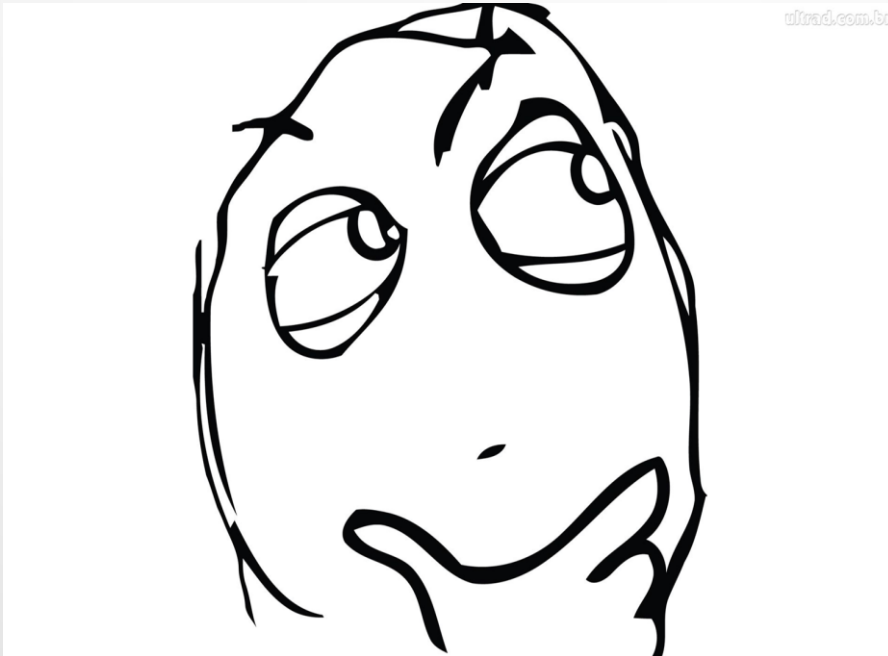
.Você pode por exemplo criar uma branch chamada desenvolvimento, trabalhar nela e depois que acabar o trabalho você pode mesclar (fazer um merge) essas alterações na branch master.

.Caso você esteja trabalhando com um repositório remoto você pode enviar essas alterações para esse repositório.

.Caso você tenha clonado o repositório esse repositório será chamado de origin, então é só executar o comando **git push origin <branch>**, e pronto as alterações serão enviadas para o repositório remoto.

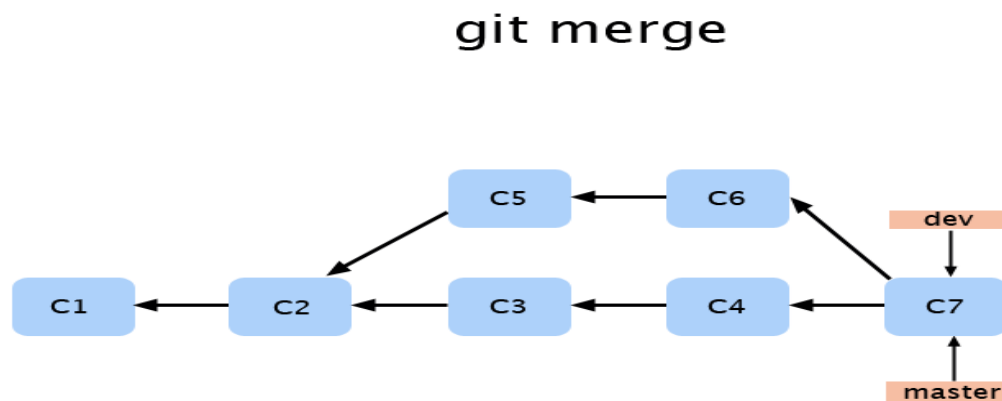
Merge e Rebase

.Quando estamos trabalhando com mais de uma branch, ou com repositórios remotos chega um momento em que é necessário juntar branches, é aí que entra o merge e o rebase.



Merge

.O comando **git merge <branch>** junta os commits da branch atual com os da branch escolhida e cria um novo commit contendo o conteúdo dos commits das duas, isso torna a visualização dos commits um pouco confusa pois a cada merge é criado um novo commit identificando que o merge foi feito. Isso é necessário para manter a Time Line organizada. A imagem a seguir exemplifica a explicação.



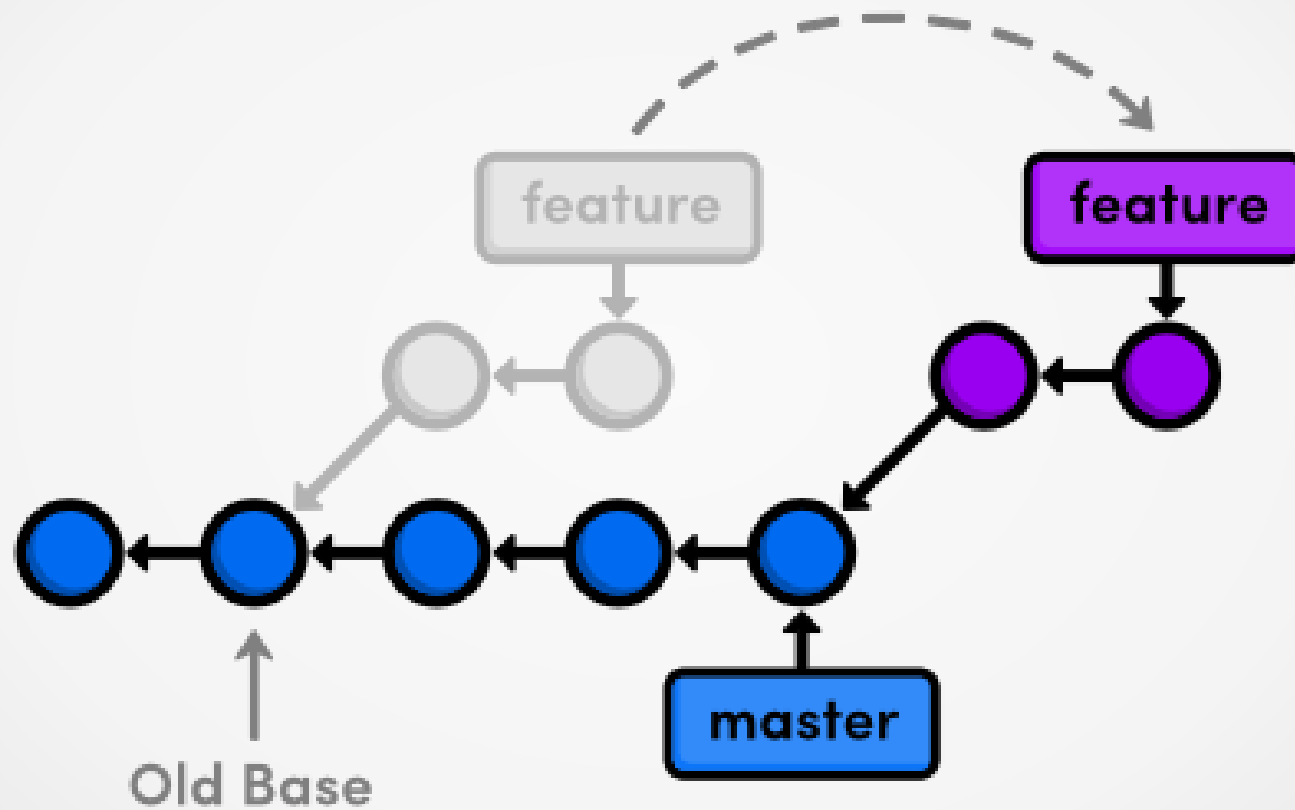
Rebase

.O comando **git rebase <branch>** reposiciona a referência de um ponto para outro vamos a um **exemplo**:

.Suponhamos que temos uma branch **master** que tem **3 commits: A,B,C** então criamos uma branch chamada **desenvolvimento** a partir da branch master e fazemos outro commit que vamos chamar de **E**, então decidimos fazer outro commit na branch master que vamos chamar de **D**, a branch desenvolvimento tem como referência o commit **C** pois foi a partir daquele ponto que ela foi criada.

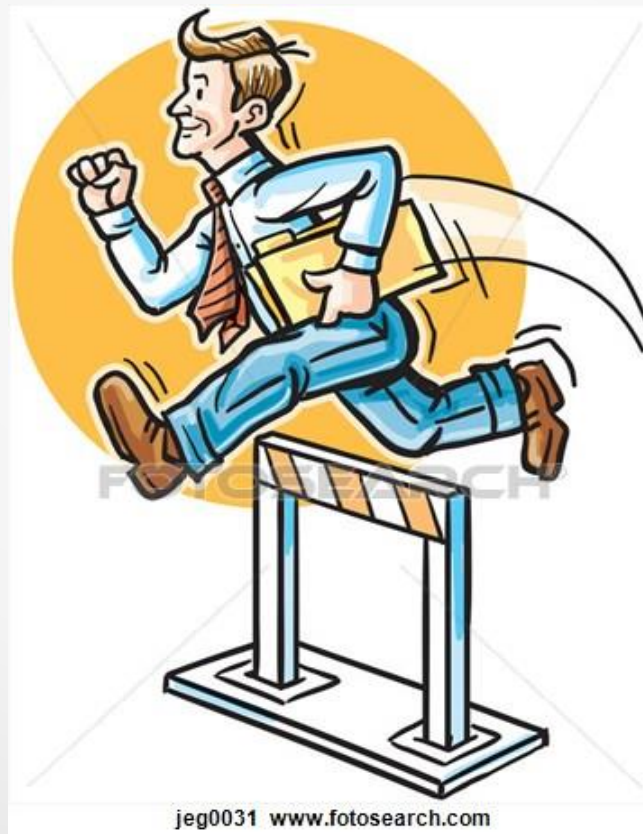
. Quando executarmos o comando **git rebase desenvolvimento** o commit E do Desenvolvimento será então trazido para o master e será apontado (rebaseado) para o último commit do master que agora não é mais o C e sim o D então a sequência ficará correta **A, B, C, D, E**. A imagem a seguir exemplifica a explicação.

Rebase



Já estamos prontos para ir além

.Os conceitos explicados anteriormente são de extrema importância para prosseguirmos.



Trabalhando com Forks

- Fork (Bifurcação) é uma ramificação.
- No GitHub o fork é feito quando um desenvolvedor inicia um projeto independente, com base no código de um projeto já existente.
- Quando um desenvolvedor decide contribuir com um projeto já existente, como por exemplo o da biblioteca JavaScript JQuery, ele deverá fazer uma cópia do repositório do JQuery para sua própria conta do GitHub para poder trabalhar a partir do que já está pronto, isso é um Fork.



Como fazer um Fork?

- Vamos fazer um fork utilizando uma conta no GitHub.
- Para criar um fork de um projeto basta acessar o repositório do projeto e clicar no botão fork, que está posicionado no lado superior direito da tela. Após isso você já tem uma cópia do projeto e pode trabalhar nele tranquilamente e desenvolver uma versão do seu projeto a partir desse ponto.

•Simples né?



Como fazer um Fork?

The screenshot shows the GitHub repository page for `gulpjs / gulp`. The repository is described as "The streaming build system" with a link to <http://gulpjs.com>. It has 743 commits, 2 branches, 13 releases, and 116 contributors. The repository is currently on the `master` branch. A red box highlights the `Fork` button, and a red arrow points to it from the right. The right sidebar shows links to `Code`, `Issues` (56), `Pull Requests` (2), and `Pulse`. The bottom section shows a merge pull request #882 from `lifeisfoo/patch-1` by `contra`, authored 7 days ago, with the latest commit `23eedeb15b`.

Repository: `gulpjs / gulp`

Buttons: Watch (564), Unstar (11,352), Fork (858)

Description: The streaming build system <http://gulpjs.com>

Stats: 743 commits, 2 branches, 13 releases, 116 contributors

Branch: `branch: master` | `gulp / +`

Recent Activity: Merge pull request #882 from `lifeisfoo/patch-1` by `contra` authored 7 days ago. latest commit `23eedeb15b`

Right Sidebar: `<> Code`, `Issues` (56), `Pull Requests` (2), `Pulse`

E a forma de trabalhar muda ?

.Quando decidimos trabalhar em um fork é importante conhecer alguns conceitos importantes. É aí que entra algo interessante que é o conceito de **upstream**.



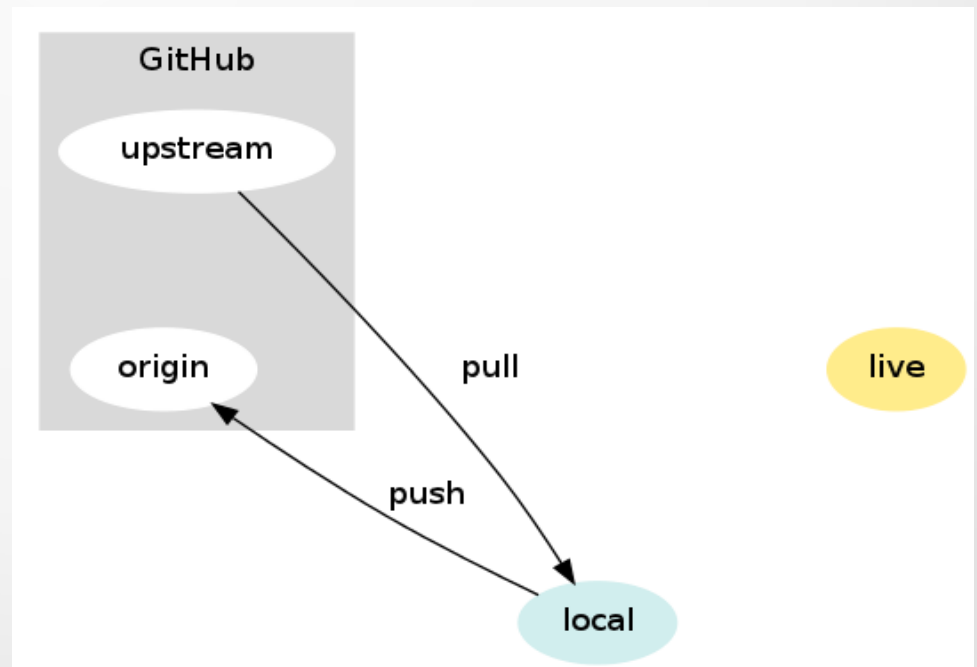
Trabalhando com Upstream

.Quando estamos trabalhando com Git e clonamos algum repositório como eu disse anteriormente, o Git vai atribuir para esse repositório o nome “origin”, então sempre que precisarmos buscar ou mandar informações para esse repositórios, podemos utilizar comandos apontando para o origin como por exemplo:

- **git push origin <branch>**
git merge origin <branch>

Trabalhando com Upstream

- Agora vamos trabalhar com um repositório remoto que é um fork de outro repositório e assim com isso teremos dois repositórios remotos.
- Teremos o nosso repositório origin e o repositório que foi copiado, a partir disso basta criar uma referência para ele e chamá-lo de upstream.



Certo e agora? Grande coisa né?

.É agora que entra o pulo do gato.



.Você vai executar o comando **git remote add upstream <repositorio-original>** e agora sempre que você precisar atualizar a sua branch local para ficar igual a do projeto original, você pode simplesmente utilizar o comando **git merge upstream** ou **git rebase upstream** para trazer o que há de novo no repositório original que pode estar mais atualizado que o seu fork, em outras palavras o upstream é apenas uma forma de **facilitar sua vida**, é um bom atalho.

Trabalhando com Pull Request

- O Pull Request é uma solicitação de recebimento.
- Trazendo isso para o nosso universo, vamos simular que um desenvolvedor tenha um fork de um repositório no GitHub e tenha clonado esse projeto em seu ambiente local e tenha feito alterações valiosas e de repente ele pense:
 - “ Nossa tenho que enviar essas minhas alterações para o repositório original, quem sabe o pessoal do projeto goste! ”
- Para que isso aconteça é preciso mandar um Pull Request com essas alterações.

Mas como fazer um Pull Request?

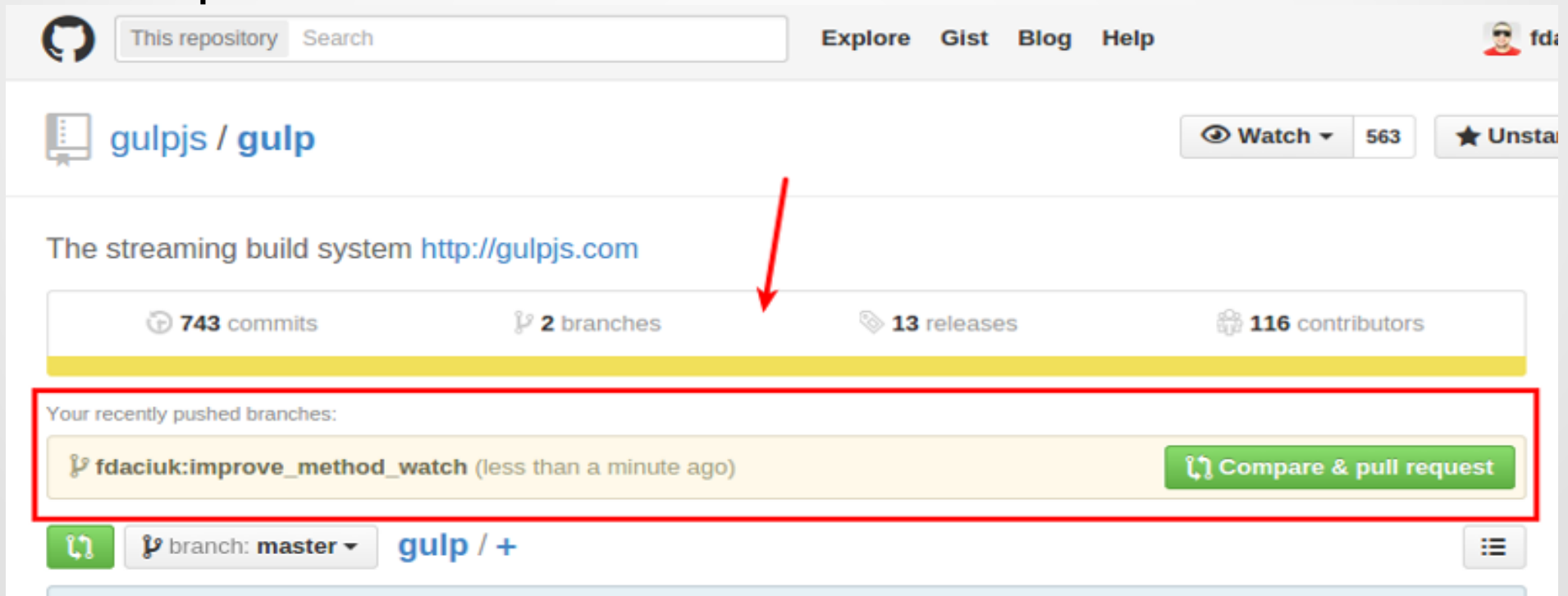
.É bem simples, suponhamos que você esteja em uma branch chamada desenvolvimento, em seu repositório local, então você adiciona todos os arquivos ao índice e commita, a partir daí é só enviar a sua branch para o seu fork da seguinte forma:

git push origin desenvolvimento

note que o nome da branch é o mesmo que o da sua local, isto é de propósito, dessa forma será criada uma branch lá no seu fork com esse mesmo nome para receber as suas alterações.

Vamos fazer o Pull Request

•Agora que já foi enviado, basta entrar no seu GitHub e acessar o repositório que você está trabalhando e provavelmente haverá um botão verde escrito “Compare & Pull Request”.



Pull Request

•Ao clicar no botão “compare & pull request” você será direcionado à tela a seguir.

The screenshot shows the GitHub interface for a pull request. At the top, the GitHub logo and navigation links (Explore, Gist, Blog, Help) are visible. The repository name 'rightscale/rightscale_cookbooks' is displayed, along with buttons for 'Admin', 'Pull Request', 'Unwatch', and 'Fork'. Below this, a tabbed interface shows 'Code', 'Network', 'Pull Requests' (selected), 'Wiki', and 'Graphs'. The 'Pull Requests' tab shows a comparison between the 'base repo: rightscale/rightscale_cookbooks' and the 'head repo: [user]/rightscale_cookbooks', both on the 'release_12H1' branch. A 'New Pull Request' button is visible. The main content area contains a text input field with the placeholder 'CONTRIB: Put New Feature Name Here', a 'Write' button, and a text area with the message: 'Hi RightScale, Please merge my changes into the main branch. - JohnD'. A green 'Send pull request' button is at the bottom right.

github Search... Explore Gist Blog Help

rightscale / rightscale_cookbooks
forked from rightscale/rightscale_cookbooks

Admin Pull Request Unwatch 1 Fork 20

Code Network Pull Requests 0 Wiki Graphs

base repo: rightscale/rightscale_cookbooks
base branch: release_12H1

head repo: [user]/rightscale_cookbooks
head branch: release_12H1

New Pull Request Commits 2 Files Changed 2

CONTRIB: Put New Feature Name Here

Write Preview Comments are parsed with GitHub Flavored Markdown

Hi RightScale,
Please merge my changes into the main branch.
- JohnD

Send pull request

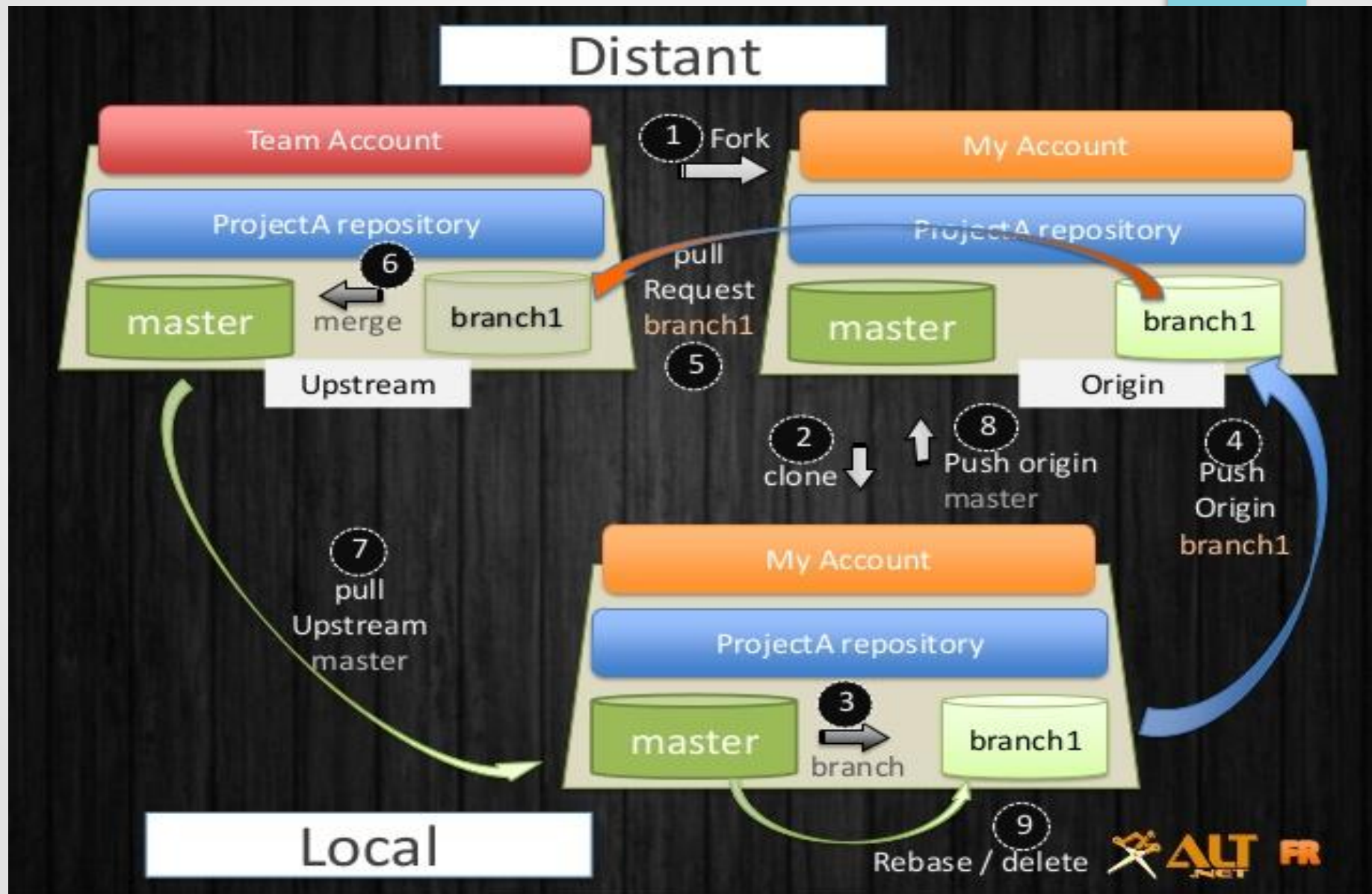
Pull Request

.Agora é só aguardar o retorno do pessoal que mantêm o projeto, caso gostem farão o merge de suas alterações ao projeto, se reprovarem vão fazer algumas observações ou simplesmente não farão merge.

.Agora é só aguardar o retorno do pessoal que mantêm o projeto, caso gostem farão o merge de suas alterações ao projeto, se reprovarem vão fazer algumas observações ou simplesmente não farão merge.



Ilustração do fluxo de trabalho



Palestrante

.Fábio Henrique Pires

.Email: fabioh.ads@gmail.com

.GitHub: <https://github.com/fabioads>

.Linkedin: <https://br.linkedin.com/pub/fábio-henrique-pires/a4/6a9/795>



GitHub



Mensagem

Inteligência é a habilidade de evitar fazer o trabalho, e mesmo assim conseguir ter o trabalho realizado.

(Linus Torvalds)