



Gerenciando Repositórios de Código com GIT

Fábio Olivé
(fabio.olive@gmail.com)



Tópicos

- Histórico e diferenciais do GIT
- Uso básico, repositórios novos ou clonados
- Entendendo o repositório de objetos
- Uso avançado com times de desenvolvimento
- Referências úteis



Histórico e Diferenciais do GIT

- Criado por Linus Torvalds para gerenciar o Linux
 - Não estava satisfeito com nenhuma alternativa
 - O desenvolvimento do kernel é muito distribuído e assíncrono, não existe um time oficial e central



Histórico e Diferenciais do GIT

- Criado por Linus Torvalds para gerenciar o Linux
 - Não estava satisfeito com nenhuma alternativa
 - O desenvolvimento do kernel é muito distribuído e assíncrono, não existe um time oficial e central
 - Utiliza algumas poucas idéias básicas e primitivas, porém muito poderosas
 - Não força nenhum modelo específico de desenvolvimento
 - Da base simples e sólida surgiu um grande conjunto de ferramentas



Histórico e Diferenciais do GIT

- Requisitos
 - Trabalhar muito rápido com milhares de arquivos
 - Geralmente cada mudança afeta apenas alguns arquivos
 - Nunca comparar todo o repositório para ver o que mudou



Histórico e Diferenciais do GIT

- Requisitos
 - Trabalhar muito rápido com milhares de arquivos
 - Geralmente cada mudança afeta apenas alguns arquivos
 - Nunca comparar todo o repositório para ver o que mudou
 - Permitir vários branches de desenvolvimento
 - Um branch no GIT não “custa” nada
 - Branches podem existir em repositórios separados



Histórico e Diferenciais do GIT

- Requisitos
 - Trabalhar muito rápido com milhares de arquivos
 - Geralmente cada mudança afeta apenas alguns arquivos
 - Nunca comparar todo o repositório para ver o que mudou
 - Permitir vários branches de desenvolvimento
 - Um branch no GIT não “custa” nada
 - Branches podem existir em repositórios separados
 - Operação desconectada
 - Cada repositório é completo e independente
 - Colaboração por transferência explícita entre repositórios



Histórico e Diferenciais do GIT

- Diferenciais
 - Não versiona arquivos e diretórios (hein?)



Histórico e Diferenciais do GIT

- Diferenciais
 - Não versiona arquivos e diretórios (hein?)
 - Rastreia a mudança de conteúdos de arquivos e diretórios
 - Armazena conteúdos de arquivos e diretórios e registra a mudança entre um conteúdo e outro



Histórico e Diferenciais do GIT

- Diferenciais
 - Não versiona arquivos e diretórios (hein?)
 - Rastreia a mudança de conteúdos de arquivos e diretórios
 - Armazena conteúdos de arquivos e diretórios e registra a mudança entre um conteúdo e outro
 - Objetos nunca mudam no repositório
 - Um commit representa TODA A HISTÓRIA do repositório até aquele ponto



Uso Básico: repo local (novo)

- mkdir projeto
- cd projeto
- git init
 - Já pode começar a usar
- edita, edita, edita, git add
- git diff --cached
- git status
- git commit
 - Vamos pro terminal ver isso na prática!



Uso Básico: repo local (existente)

- cd projeto
- git init
- git add .
- git commit -m "Commit inicial"
 - Vai tudo pro repositório de objetos
 - Basta seguir desenvolvendo e commitando



Uso Básico: repo local (clone)

- `git clone /caminho/do/repositório.git`
 - Faz uma cópia completa do repositório original
 - Guarda referência remota de onde veio
- `cd repositório`
- `git branch novidades`
- `git checkout novidades`
- edita, add, commit, repete
- Mais tarde o dono do repositório original busca o branch novidades, ou o desenvolvedor manda o branch pro original.



Uso Básico: repo remoto clonado

- `git clone git://foo.bar.com/repos/projeto.git`
 - Faz uma cópia completa do repositório original
 - Guarda referência remota de onde veio
- `cd projeto`
- `git branch novidades`
- `git checkout novidades`
- edita, add, commit, repete
- `git checkout master`
- `git pull, rebase, etc`



Entendendo o Repo de Objetos

- Quando um arquivo é armazenado no repositório, seu conteúdo é passado por uma função HASH
- O HASH resultante dá o nome “interno” do conteúdo dentro do repositório



Entendendo o Repo de Objetos

- Quando um arquivo é armazenado no repositório, seu conteúdo é passado por uma função HASH
- O HASH resultante dá o nome “interno” do conteúdo dentro do repositório
- Um diretório é um “conteúdo” que mapeia nomes de arquivo para os hashes de seus conteúdos
- Um objeto diretório que represente o diretório raiz do repositório na verdade representa UM ESTADO deste repositório



Entendendo o Repo de Objetos

- Se algum arquivo for modificado, o hash do arquivo muda, portanto o conteúdo do diretório muda, e o hash do diretório muda
- Cada hash de diretório raiz representa um possível estado do repositório



Entendendo o Repo de Objetos

- Se algum arquivo for modificado, o hash do arquivo muda, portanto o conteúdo do diretório muda, e o hash do diretório muda
- Cada hash de diretório raiz representa um possível estado do repositório
- **COMMITs** registram mudanças entre dois estados do conteúdo e associam a informações como autor, data, mensagem, etc
 - Um commit contém o hash do commit ANTERIOR e um hash de diretório ATUAL



Representação Didática

Objetos



Armazenando um Arquivo

Objetos

C5

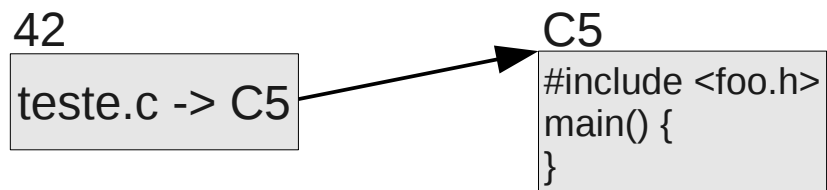
```
#include <foo.h>
main() {
}
```

Index: teste.c -> C5



Armazenando um Diretório

Objetos

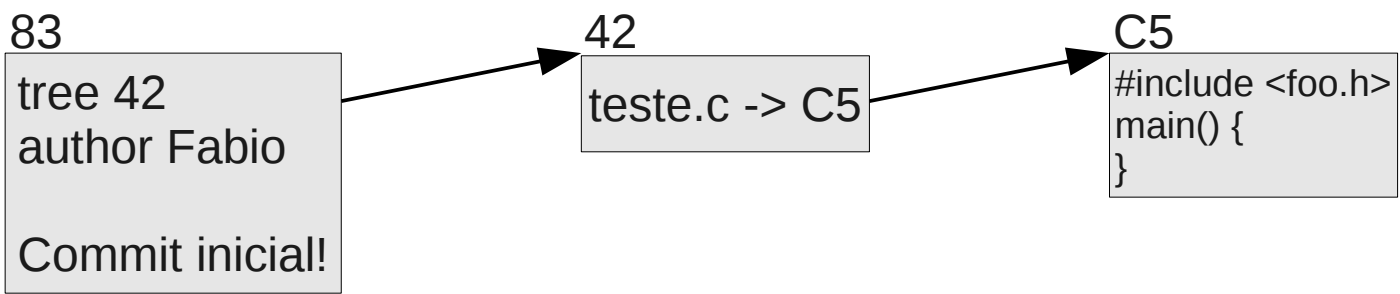


Index: teste.c -> C5



Armazenando um Commit

Objetos



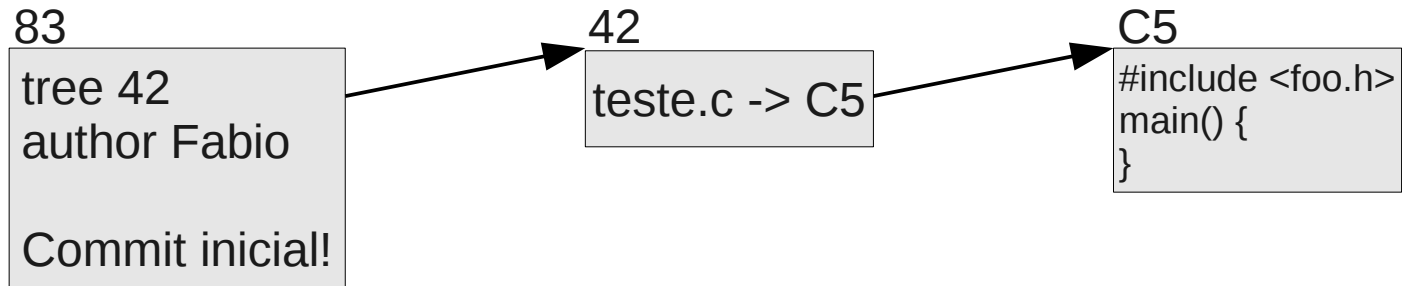
HEAD -> 83

Index:



Modificando o Arquivo

Objetos



A2
#include <foo.h>
main() {
 foo();
}

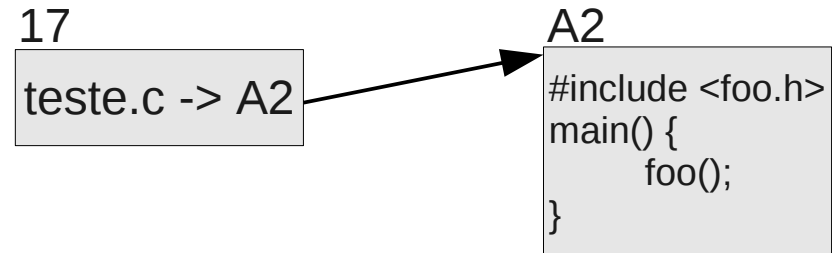
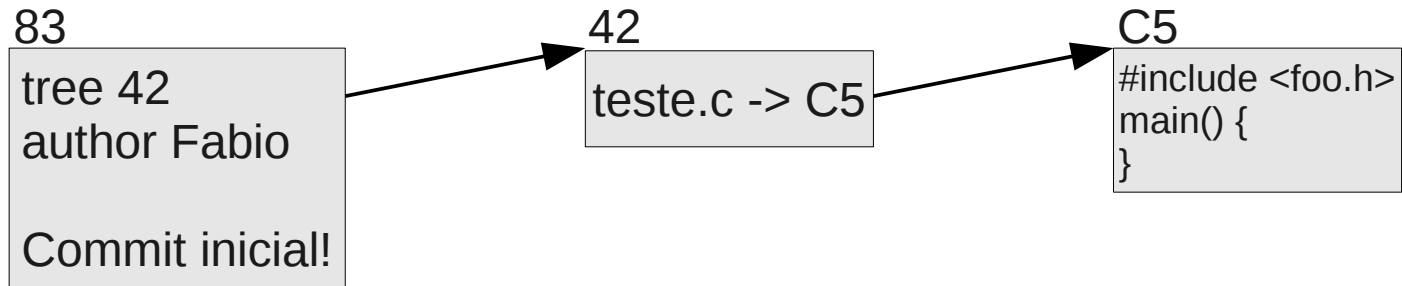
HEAD -> 83

Index: teste.c -> A2



Muda o diretório

Objetos



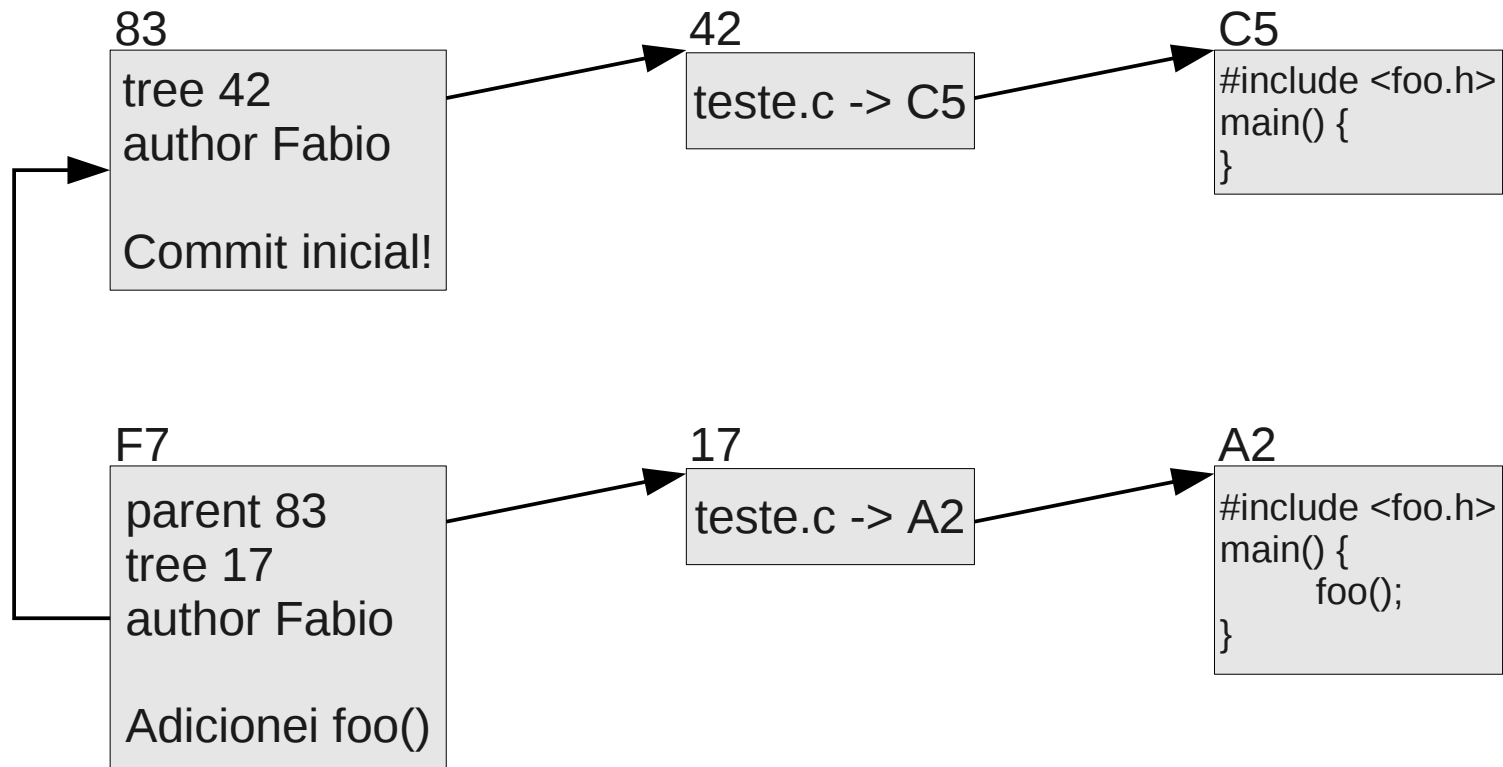
HEAD -> 83

Index: teste.c -> A2



Novo Commit

Objetos



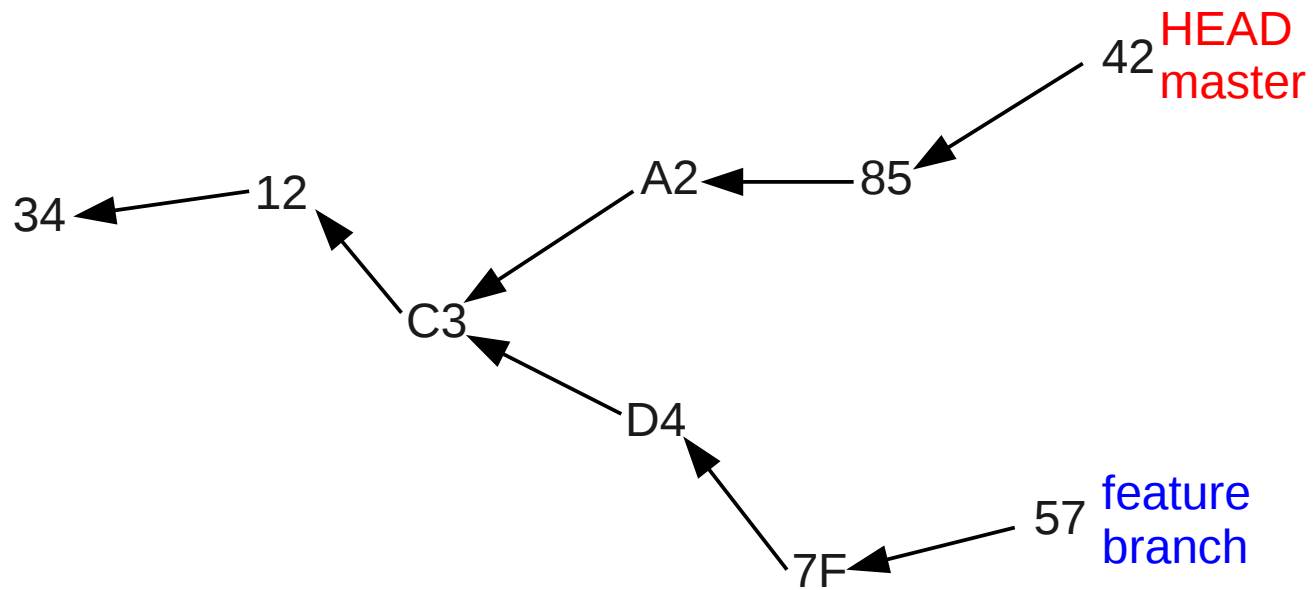
HEAD -> F7

Index:



Histórico de Commits

Objetos





Uso Avançado

- Time pequeno, repositório centralizado
 - Servidor mantém repositório oficial
 - Desenvolvedores enviam branches pro repositório central
 - Integrador faz merge pro master no servidor



Uso Avançado

- Time pequeno, repositório centralizado
 - Servidor mantém repositório oficial
 - Desenvolvedores enviam branches pro repositório central
 - Integrador faz merge pro master no servidor
- Time pequeno, sem repositório centralizado
 - Desenvolvedores mantêm seus repositórios locais
 - Integrador busca os branches na máquina de cada um
 - Talvez patches enviados por email, se for time distribuído
 - git format-patch registra cada commit desde um ponto original de forma que possa ser enviado por email facilmente



Uso Avançado

- Time grande, repositório centralizado
 - Vários níveis de repositórios por nível hierárquico
 - Desenvolvedores e integradores trabalham no seu nível
 - Integrador envia master do time pro próximo nível



Uso Avançado

- Time grande, repositório centralizado
 - Vários níveis de repositórios por nível hierárquico
 - Desenvolvedores e integradores trabalham no seu nível
 - Integrador envia master do time pro próximo nível
- Time desconhecido e massivamente distribuído
 - Desenvolvedores mantêm seus repositórios locais, atualizam repos online ou enviam séries de patches por email para listas ou grupos interessados
 - git clone e vai pra ilha deserta
- Sites como <http://github.com/>



Referências Úteis

- GIT Community Book
 - <http://book.git-scm.com/>
- Vídeo: Linus Torvalds on GIT
 - <http://www.youtube.com/watch?v=4XpnKHJAok8>
- Vídeo: GIT – a Talk by Randal Schwartz
 - <http://www.youtube.com/watch?v=8dhZ9BXQgc4>
- Man pages e experimentação
- Google, óbvio