Pontifícia Universidade Católica
do Rio de Janeiro

Introdução à Engenharia

ENG1000

Aula Extra - Projetos e Controle de Versões

2016.1

Prof. Augusto Baffa
<abaffa@inf.puc-rio.br>

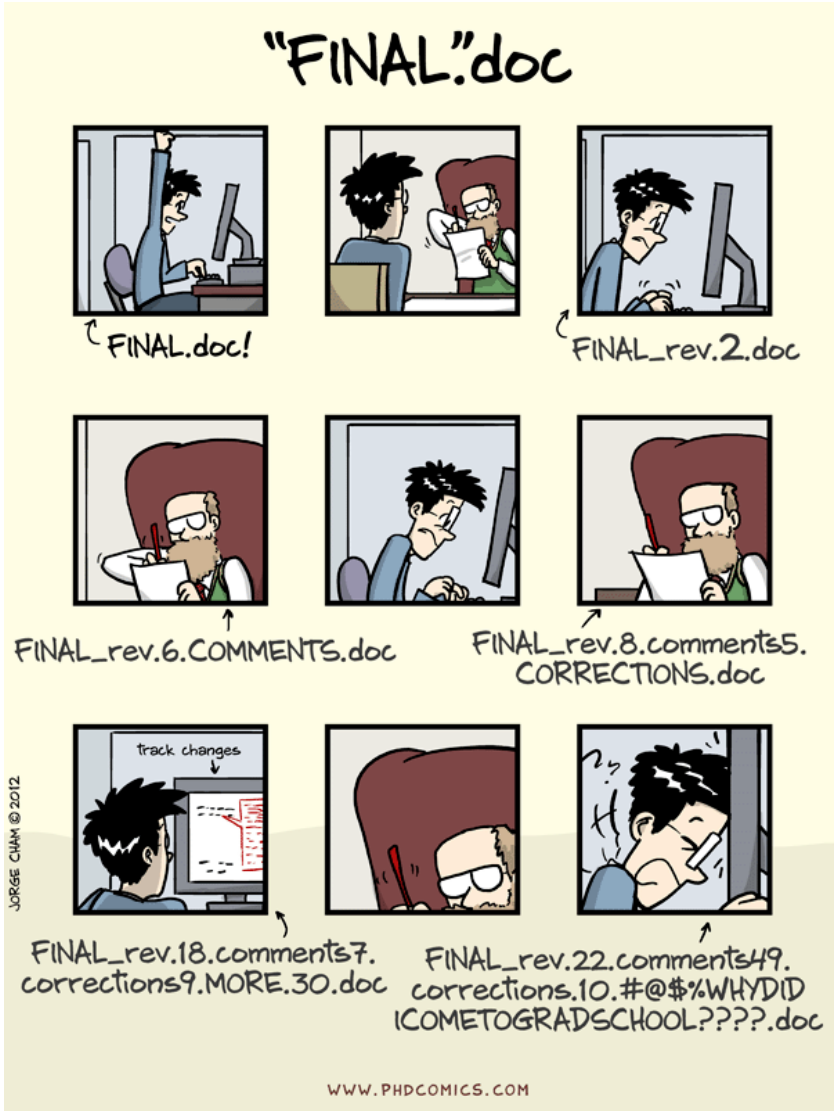
Armazenando seu projeto

- Como armazenar versões?
  - Não se preocupar
  - Acumular arquivos ZIP numerados (ou com data da versão)
  - Sistema de Gerenciamento de Versões

Armazenando seu projeto

- Como armazenar versões?
  - Não se preocupar
    - Boa sorte!
  - Acumular arquivos ZIP numerados (ou com data da versão)
    - Como verificar as diferenças no código (Diff)
  - Sistema de Gerenciamento de Versões
    - Facilidade para verificar as mudanças no código entre versões
    - Facilidade em restaurar versões para testes

Por que usar um Sistema de Gerenciamento de Versões?



http://www.phdcomics.com/comics/archive.php?comicid=1531

Por que usar um Sistema de Gerenciamento de Versões?

- Histórico das alterações
  - Responsável pelas alterações (quem quebrou o build)
- Facilidade para voltar atrás e restaurar o código
- Não precisa se preocupar em estragar um código que funciona
- Permite juntar alterações feitas por diversos desenvolvedores

Sistema de Gerenciamento de Versões

- Soluções Comerciais
  - Microsoft Visual SouceSafe (VSS)
  - Rational ClearCase
  - Borland StarTeam
- Soluções Open-source
  - Concurrent Version System (CVS)
  - Subversion (SVN)
  - Git
  - Mercurial

Sobre o Git

- Git é um Sistema open-source de gerenciamento de versões
- Foi desenvolvido pelo Linus Torvards (desenvolvedor do Linux)
  - Criado em 2005 para gerenciar o desenvolvimento dos códigos do Linux
- Pode ser usado para controlar versões de qualquer formato (geralmente arquivos de texto)
  - Códigos fonte
  - Projetos de análise de dados
  - Manuscritos
  - Websites
  - Apresentações
  - Etc...

Por que usar o Git?

- É rápido. O foco do sistema é a velocidade.
- Não é necessário ter acesso ao direto servidor, basta usar sua conta git.
- Muito bom em gerenciar e unificar modificações simultâneas ao mesmo arquivo
- Tem se tornado o principal protocolo de gerenciamento de versões.

Onde usar?

- https://bitbucket.org/
  - Git ou Mercurial
  - Repositórios privados gratuitos

- https://github.com/
  - Git
  - Repositórios privados pagos

Sobre o GitHub

- É um site para repositórios Git.
- Possui interface para explorar repositórios git.
- Verdadeiro open source
  - Permite acesso fácil ao código de diversos desenvolvedores
- É como o Facebook para programadores

Por que usar o GitHub?

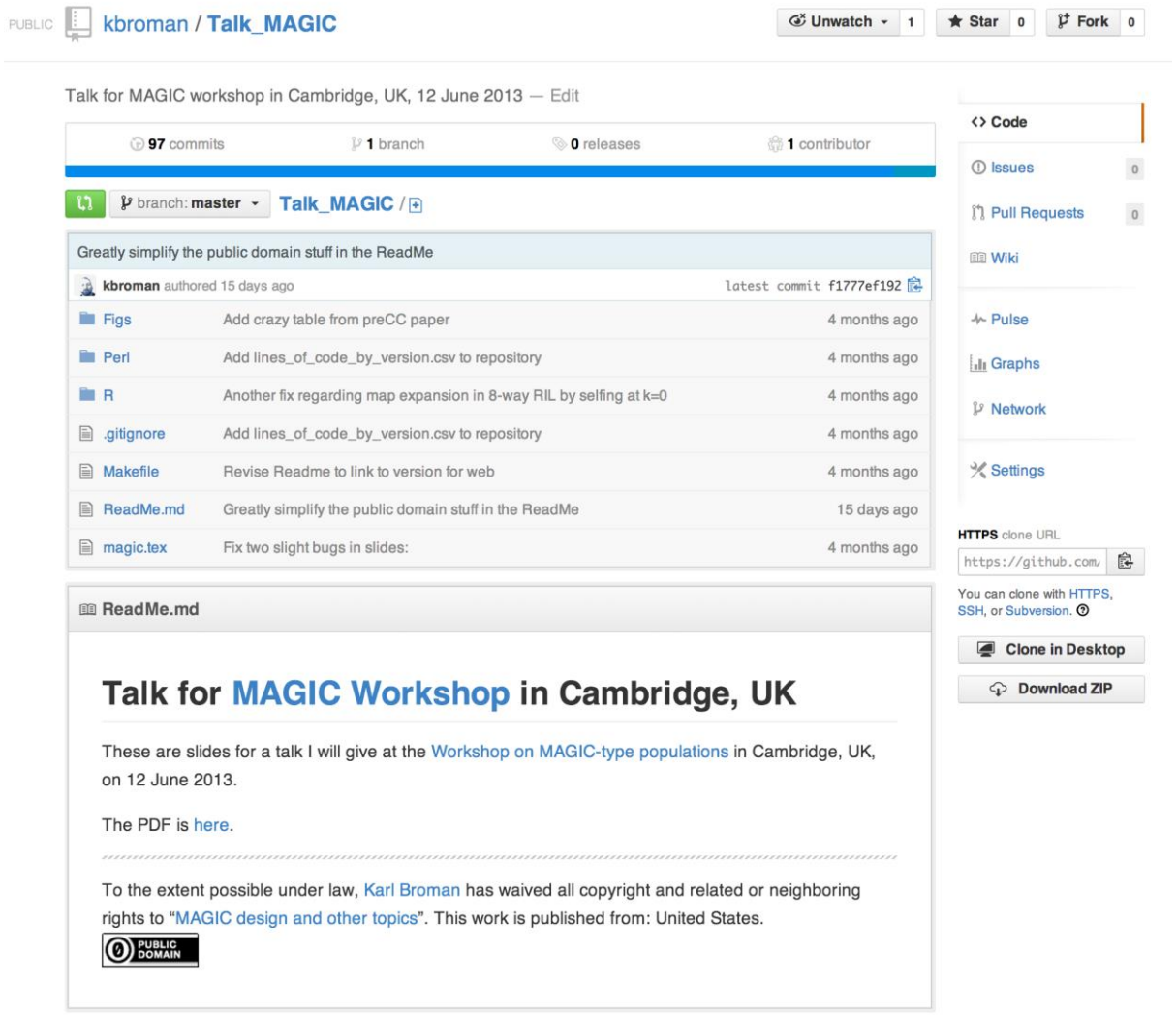
- É otimizado e existe uma continua preocupação com os servidores para os aspectos do git.

- Interace gráfica para os repositórios
  - Permite a exploração do código e seus históricos.
  - Permite registrar e monitorar bugs ou tarefas

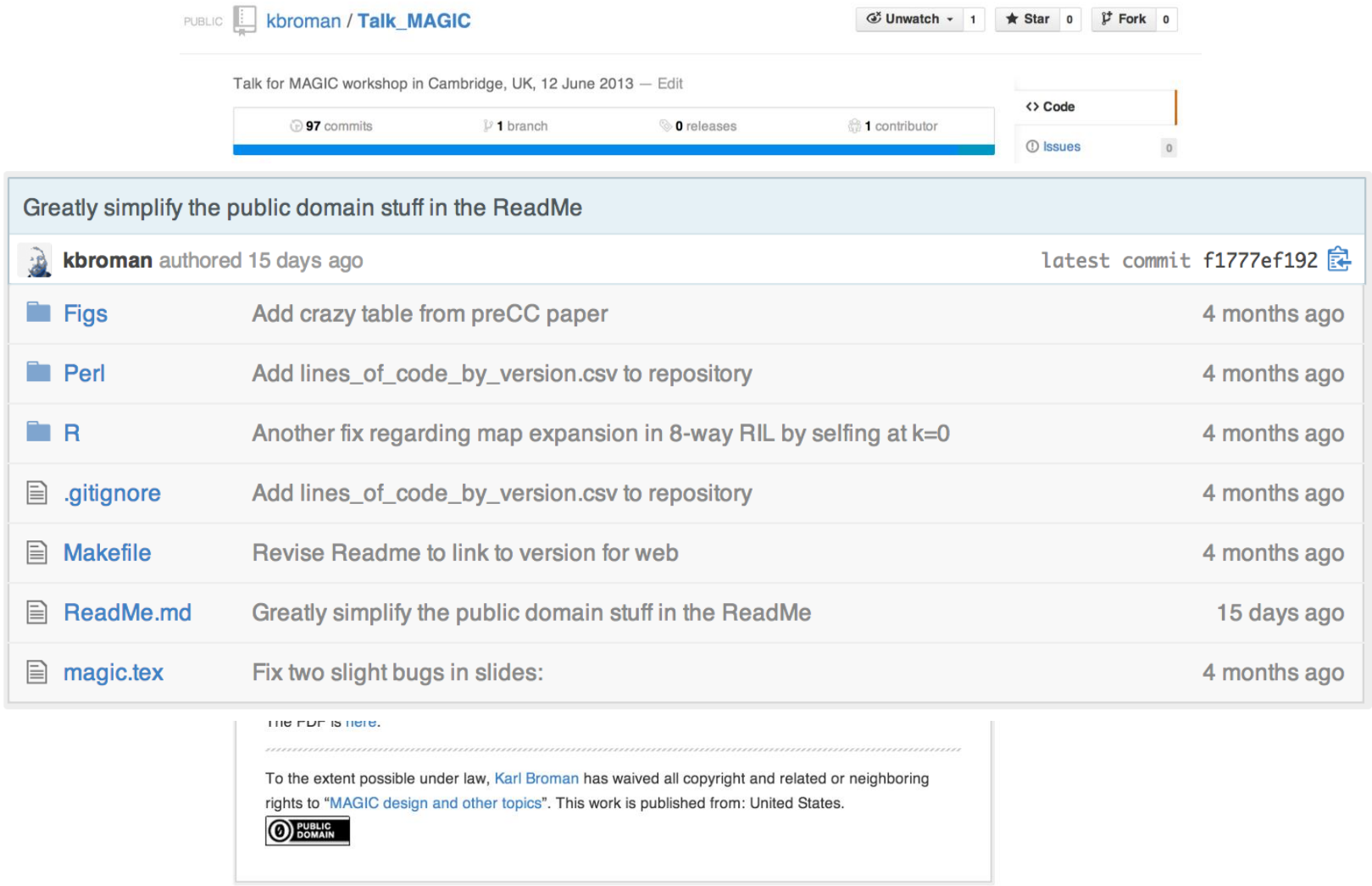
- Auxilia
  - ... no aprendizado através do projetos de outros usuários.
  - ... na verificação do que os outros estão fazendo.
  - ... na contribuição com o código de outros.

- Facilita o processo colaborativo
  - “Existe um problema na sua documentação” vs “Aqui está uma correção para a sua documentação”.

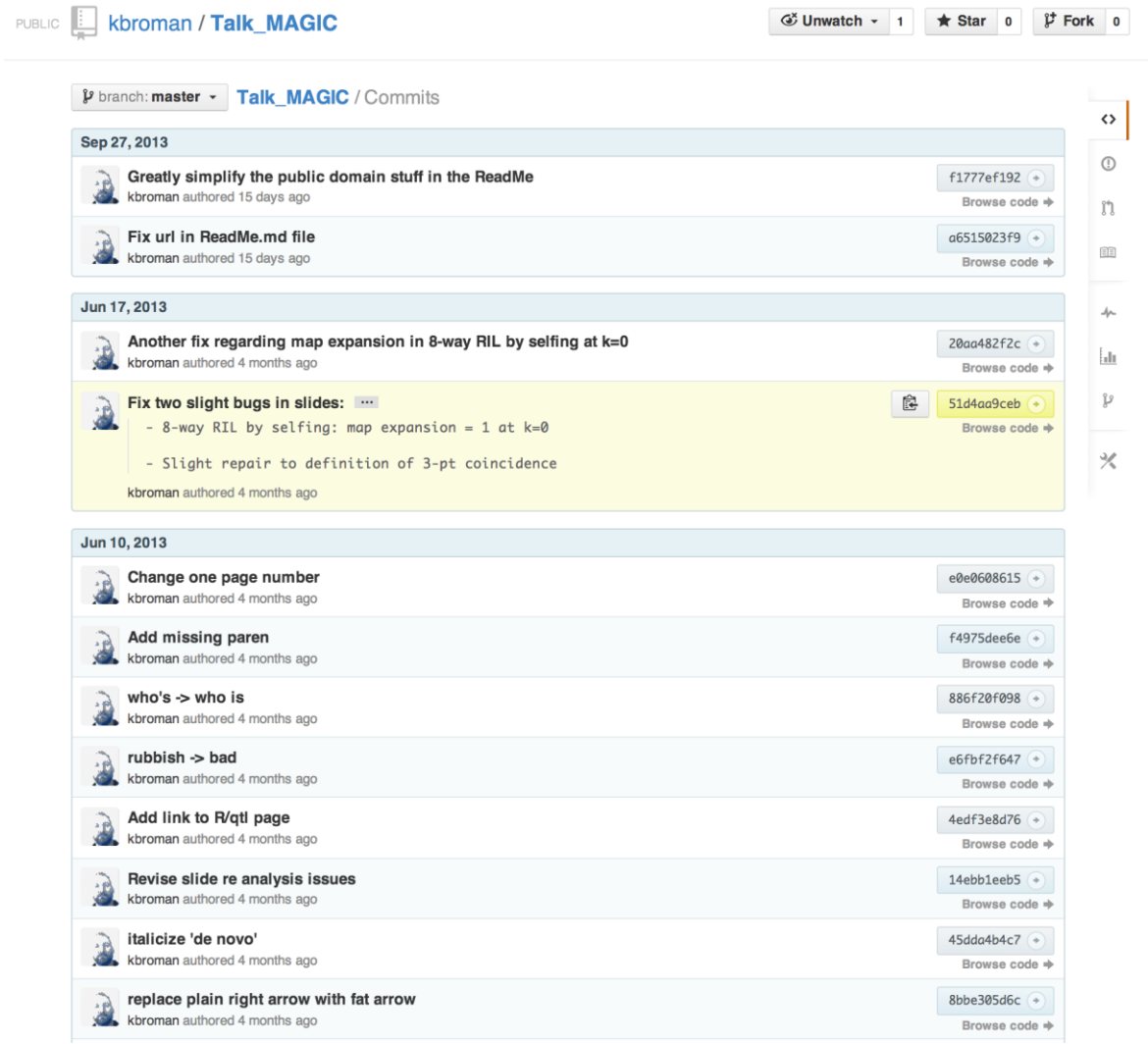
Exemplos de repositório Github



Exemplos de repositório Github



Exemplo de histórico no Github



Exemplo de Diff no Github

PUBLIC kbroman / Talk_MAGIC

Unwatch 1 | Star 0 | Fork 0

Fix two slight bugs in slides:

Browse code

```
- 8-way RIL by selfing: map expansion = 1 at k=0

- Slight repair to definition of 3-pt coincidence
```

master

kbroman authored 4 months ago — 1 parent e0e0608 commit 51d4aa9ceb104bbf26e0cbe105a5c7f8dc02a832

Showing **2 changed files** with **5 additions** and **3 deletions**. Show Diff Stats

6 R/map_expansion_func.R — View file @ 51d4aa9

```
... ... @@ -25,8 +25,10 @@ mesibA4 <- function(k)
25  25   ################
26  26   # Eight-way
27  27   ################
28      -meself8 <- function(k)
29      -    4 - (((1)/(2)))^(k-2)
      28 +meself8 <- function(k) {
      29 +  if(k==0) return(1)
      30 +  4 - (((1)/(2)))^(k-2)
      31 +}
30  32
31  33   mesibX8 <- function(k)
32  34       ((14)/(3)) - (((30 + 14*sqrt(5))/(15))) * (((1+sqrt(5))/(4)))^k - (((30 - 14*sqrt(5))/(15))) * (((1-s
```

2 magic.tex — View file @ 51d4aa9

```
... ... @@ -636,7 +636,7 @@
636 636
637 637   \hspace{20mm} {\color{myblue} = $\mathsf{Pr(\text{rec'n in 23} \ |
638 638       \ \text{rec'n in 12}) /
639     -  Pr(\text{rec'n in 12})}$}
    639 +  Pr(\text{rec'n in 23})}$}
640 640
641 641   \item
642 642   No interference { \color{myblue} = 1 }
```

Uso do Git / GitHub

- Altere alguns arquivos
- Verifique o que foi alterado
  - git status
  - git diff
  - git log

- Informe quais arquivos alterados serão salvos no repositório
  - git add

- Confirme as alterações
  - git commit

- Envie as alterações para o GitHub
  - git push

- Receba alterações do projeto enviadas por outros colaboradores
  - git pull

Inicializando um Repositório

- Crie um diretório de trabalho
  - Por exemplo, ~/ProjetoGit
- Inicialize o diretório
  - git init
    - Este comando cria um subdiretório ~/ProjetoGit/.git

```
$ mkdir ~/ProjetoGit
$ cd ~/ProjetoGit
$ git init
Subdiretório para o repositório iniciado criado em ~/ProjetoGit/.git/
```

Produzindo Conteúdo

- Crie um arquivo Readme.MD
  - O arquivo readme.md é lido automaticamente pelo github e monstrado como descrição do diretório no ambiente gráfico do site.

```
# Playlib
PlayLib is a graphics and basic functions library for interactive applications that aims to simplify the process of developing graphics applications and assist, playfully, the learning of C language.

Examples of Playlib applications and games
-
This repository presents many Playlib examples to assist during classes.

Email Augusto Baffa <abaffa@inf.puc-rio.br> with questions or
comments , or submit an
[Issue at the GitHub](https://github.com/PlaylibExamples/issues).
```

Enviando ao Repositório

- Adicione as mudanças usando git add

```
$ git add Readme.MD
```

- Confirme as alterações usado git commit

```
$ git commit –m “Primeiro envio do arquivo README”
[master (root -commit) 32c9d01] Initial commit of README file
1 file changed , 14 insertions (+)
create mode 100644 Readme.MD
```

  - O parâmetro –m permite fornecer a descrição do commit
  - Sem este parâmetro o git abre um editor de texto para solicitar a descrição
  - Use uma descrição compreensível e significativa
  - A descrição pode ter várias linhas mas faça da primeira linha um resumo da descrição.

Observações sobre os commits

- Sempre que possível, use pequenos commits
- Não fique desatualizado com seus colaboradores
- Commite apenas arquivos fonte e ignore arquivos derivados
  - Por exemplo, ignore executáveis, arquivos gerados durante a compilação e arquivos durante a execução dos testes de código.
- Use o arquivo .gitignore para indicar arquivos que devem ser ignorados.

```
*~
manuscript.pdf
Figs/*.pdf
.RData
.RHistory
*.Rout
*.aux
*.log
*.out
```

Apagando/Movendo/Renomeando Arquivos

- Para arquivos que estão sendo gerenciados pelo git:
  - Use **git rm** ao invés de rm
  - Use **git mv** ao invés de mv

```
$ git rm myfile
$ git mv myfile newname
$ git mv myfile SubDir/
$ git commit
```

Criando um repositório no GitHub

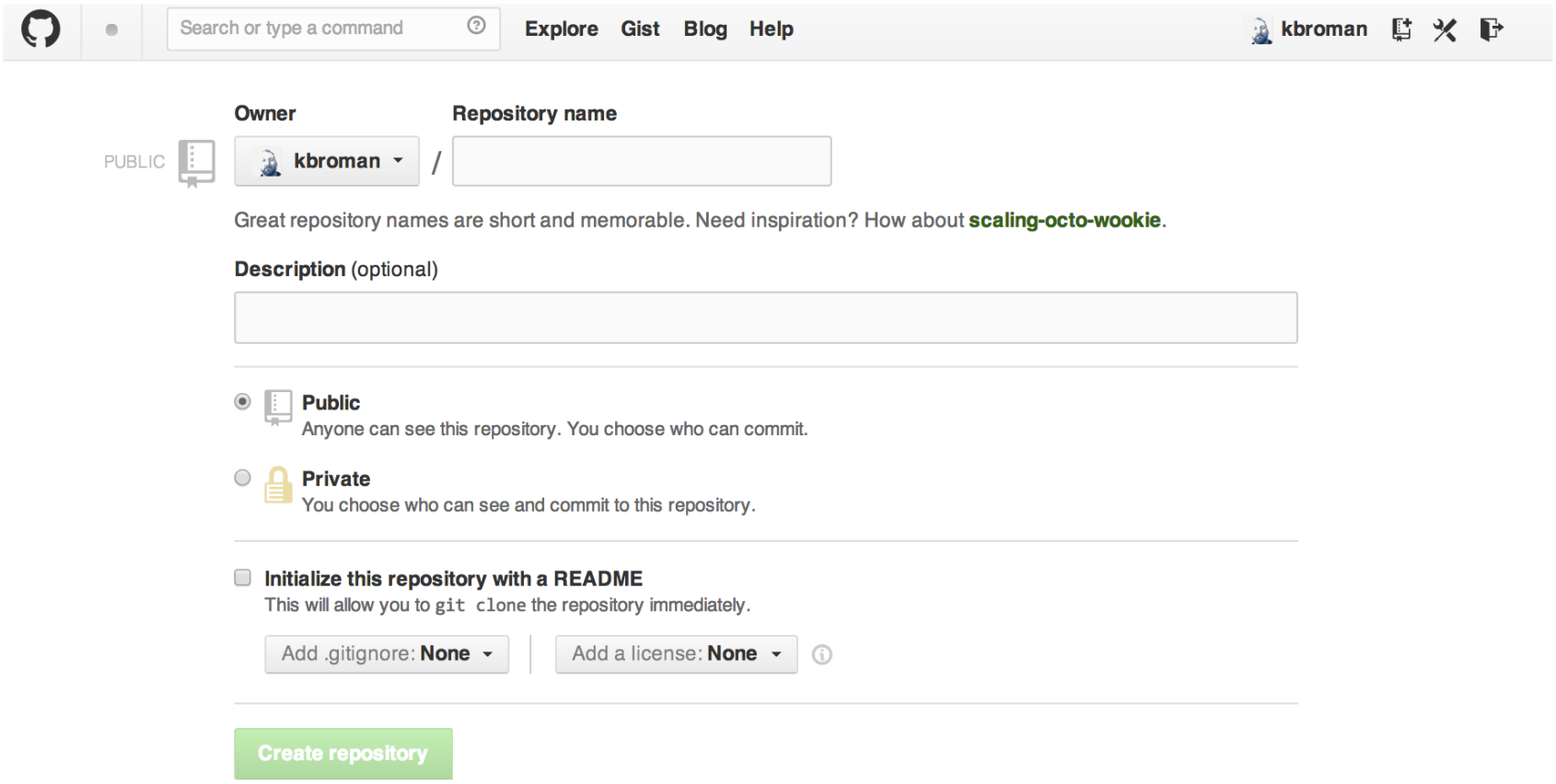
- Crie sua conta

- Clique no botão “Create a new repo”
- Digite o nome e a descrição do repositório
- Clique em “Create repository”

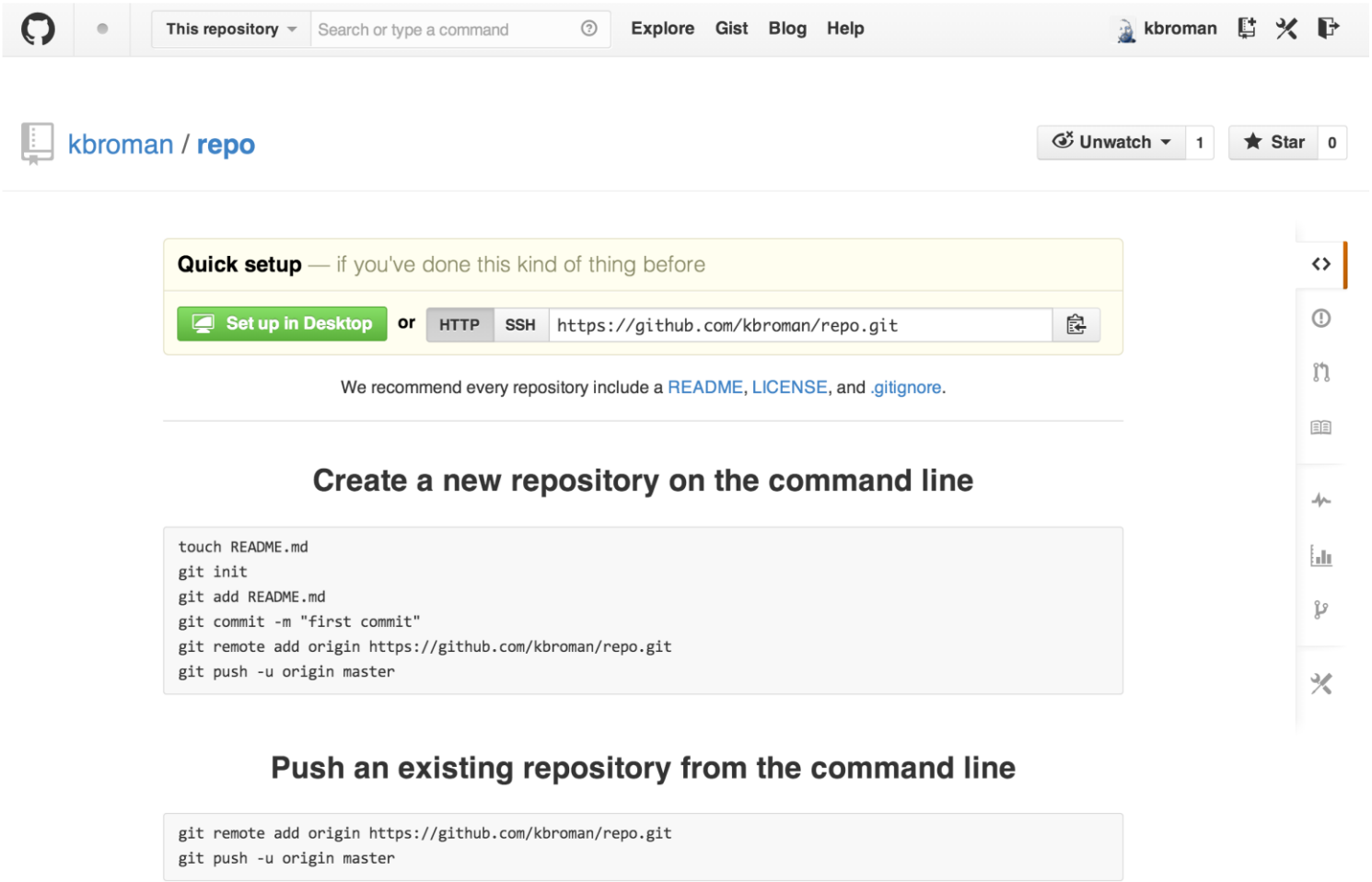
- Vá ao diretório e digite as linhas de comando:

```
$ git remote add origin git@github.com:username/repo
$ git push -u origin master
```

Exemplos


Search or type a command
Explore
Gist
Blog
Help
kbroman
Owner
Repository name
PUBLIC
kbroman
/
Great repository names are short and memorable. Need inspiration? How about scaling-octo-wookie.
Description (optional)
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with a README
This will allow you to git clone the repository immediately.
Add .gitignore: None
Add a license: None
Create repository

Exemplos


This repository
Search or type a command
Explore
Gist
Blog
Help
kbroman
kbroman / repo
Unwatch
1
Star
0
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTP
SSH
https://github.com/kbroman/repo.git
We recommend every repository include a README, LICENSE, and .gitignore.
Create a new repository on the command line
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kbroman/repo.git
git push -u origin master
Push an existing repository from the command line
git remote add origin https://github.com/kbroman/repo.git
git push -u origin master

Problemas ou Tarefas

- Problemas (bugs) ou sugestões de tarefas para o código?
  - Registre no projeto como uma “Issue”

- Ou melhor: forneça uma correção
  - Fork (copia e associa o projeto de outra pessoa ao seu repositório)
  - Clone (copia o projeto para o seu repositório)
  - Modifique o código
  - Commit
  - Push
  - Envie um Pull Request (pedido para sincronizar o código com o proprietário)

Sugerindo uma alteração em outro repositório

- Vá ao repositório
  - Exemplo http://github.com/someone/repo
- **Fork** o repositório
  - Clique no botão “Fork”
- **Clone** o repositório
  - git clone git@github.com:username/repo
- Faça as devidas modificações, git add e git commit
  - git push
- Vá novamente ao repositório GitHub
- Clique em “Pull Requests” e “New pull request” para criar uma solicitação para o autor sincronizar as modificações

Aceitando Modificações de Terceiros

- Adicione a conexão com o usuário
  - git remote add friend git://github.com/friend/repo
- Baixe as modificações
  - git pull friend máster
- Então as envie novamente ao seu Repositório GitHub
  - git push

Gerenciando Conflitos

- Podem acontecer após um git pull friend master

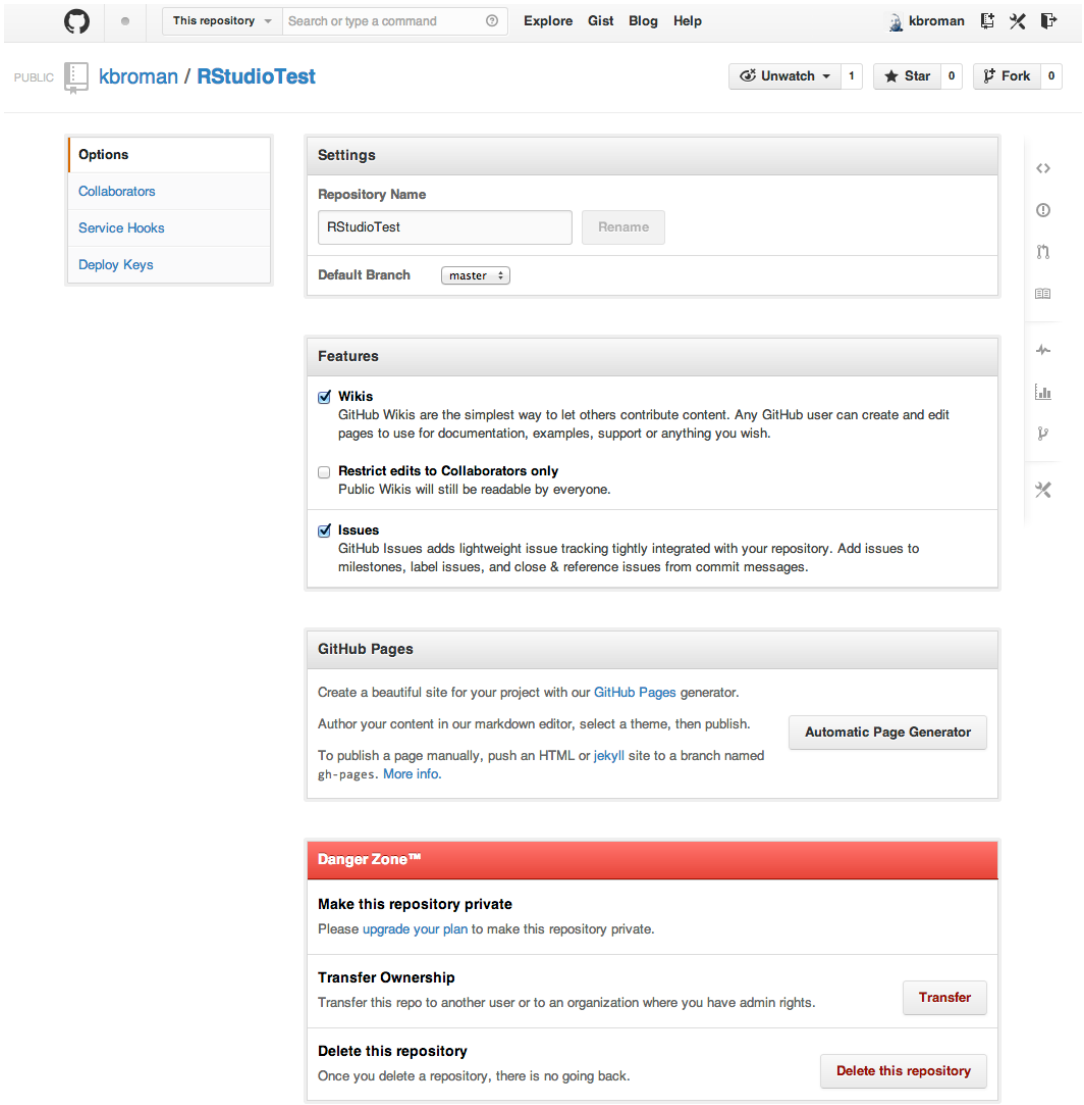
```
Auto -merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

- O arquivo estará com o seguinte conteúdo:

```
<<<<<<< HEAD
A line in my file.
=======
A line in my friend 's file
>>>>>>> 031389 f2cd2acde08e32f0beb084b2f7c3257fff
```

- Edite, adicione, commit, push e envie um pull request

Deletando Repositórios



Ferramentas

- Git for Desktop
  - https://desktop.github.com/

- Tortoise Git
  - https://tortoisegit.org/

Conclusões

- Gerenciar as versões do seu código auxilia na colaboração entre diversos desenvolvedores, no lançamento de versões do projeto, na verificação de erros e na recuperação de versões corretas.

- Criar um projeto open-source significa que todos poderão verificar os erros do seu código.

- Utilizar um controle de versão quer dizer que todos poderão verificar todos os erros do seu código. (mesmo os antigos que foram corrigidos)

Leitura Complementar

- Broman, Karl; Younkin, Samuel G.; **A brief introduction to git & GitHub**. University of Wisconsin–Madison, http://github.com/syounkin/GitPrimer
- Karl's tutorial: http://kbroman.github.io/github_tutorial
- Karthik Ram's slides: http://karthikram.github.io/git_intro
- Pro Git book: http://git-scm.com/book