DevOps e GitHub

2021/2

Prof. Ramon Pereira

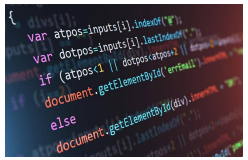
Apresentação

Ramon Gonçalves Pereira

- Doutorado em andamento em Ciência da Computação/UFMG
- Mestrado: Ciência da Computação/UFMG
- Graduação: Sistemas de Informação
- Áreas de atuação:
  - Desenvolvimento de Sistemas e Aplicativos Web/Mobile
  - Inteligência Artificial e Mineração de Dados (*Data Engineering*)
  - Metodologias Ágeis de Desenvolvimento
- O que gosto de fazer:







Mas afinal, quem são vocês?

https://forms.gle/W2yRJPCcHLkcjDag8

Objetivos e Metas

- **GitHub**
  - **COMMIT, PUSH, PULL,**
  - **BRANCH e MERGE**
  - **PULL REQUEST e BASH**
  - **CI e CD**

- **DevOps**
  - **CI e CD com Bitbucket Pipelines**
  - **CI e CD com Jenkis**
  - **Docker e AWS ECS**
  - **Serviços em Nuvem**

Aulas

25/08/2021 - Git Parte 1

01/09/2021 - Git Parte 2

08/09/2021 - Devops Parte 1

15/09/2021 - Devops Parte 2

Pontuação

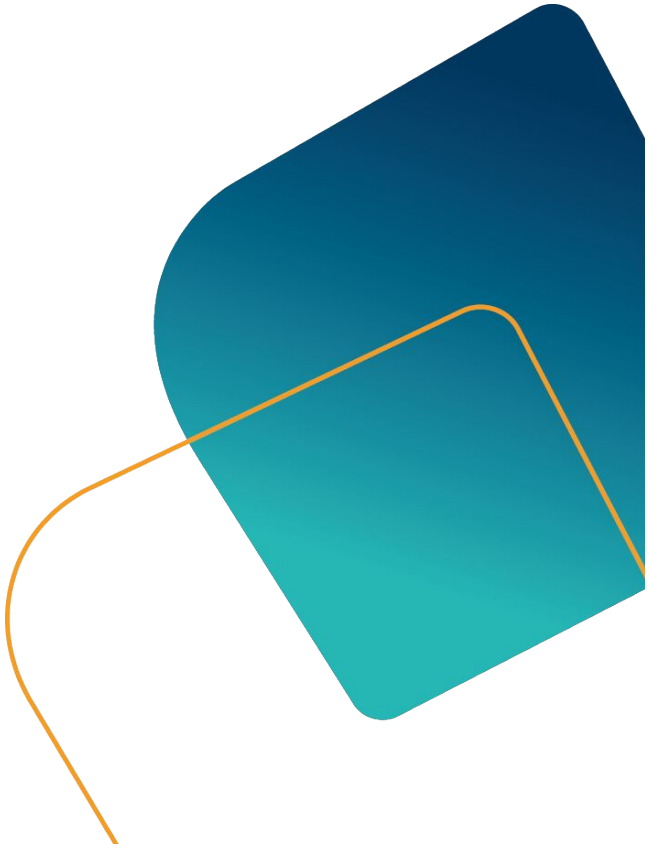
Atividade prática todas as aulas!

Entrega até a próxima semana via uLife

O aluno só será aprovado mediante a entrega de todas as atividades práticas

Git ? Git Hub? Controle de Versão?

Ramon Pereira (ramon.pereira@prof.unibh.br)



Vamos pensar um pouco..

- Você está trabalhando em um projeto com mais 3 desenvolvedores ao mesmo tempo e precisa repartir as novas funcionalidades do sistema com eles.

Como você faz a junção do código ao final do desenvolvimento? (Lembrando que o mesmo arquivo pode ser editado por mais de um dev)

Vamos pensar um pouco..

- Seu chefe pediu para você deletar uma funcionalidade do sistema que não é utilizada. Após 3 meses ele decidiu que quer essa funcionalidade de volta.

Como você faz para recuperar essa funcionalidade e implementá-la no projeto mesmo após muitas outras mudanças?

Vamos pensar um pouco..

- Você é muito otimista e por isso não possui nenhum sistema de backup automático da sua aplicação em ambiente de desenvolvimento. Um belo dia seu computador queima e você não havia copiado suas últimas features para um pendrive.

Como você faz para recuperar esse código?

Sistema de Controle de Versão

- É um software que tem a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer.

- Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.

Sistema de Controle de Versão

- Entre os mais comuns encontram-se as soluções livres: CVS, Mercurial, Git e SVN (Subversion); e as comerciais: SourceSafe, TFS, PVCS (Serena) e ClearCase.

- O desenvolvimento de software livre utiliza mais o Git (com repositórios no GitHub), que vem substituindo o SVN, que por sua vez é um sucessor do CVS.

Controle de Versão - Vantagens

- Controle do histórico;
- Trabalho em equipe;
- Marcação e resgate de versões estáveis;
- Ramificação de projeto;
- Segurança;
- Rastreabilidade;
- Organização;
- Confiança

Controle de Versão - Funcionamento Básico

- A maior parte das informações - com todo o histórico - ficam guardadas num repositório (repository em inglês), num servidor qualquer.

- Geralmente um cliente pode aceder ao repositório pela rede (via socket) ou localmente quando o cliente está na mesma máquina do servidor.

- O repositório armazena a informação de modo persistente num sistema de arquivos ou banco de dados

Controle de Versão - Funcionamento Básico

- Inúmeros clientes podem se conectar em um repositório, e assim leem e escrevem nesses arquivos.

- Cada servidor pode ter vários sistemas de controle de versão e cada sistema pode ter diversos repositórios, limitando-se na capacidade de gerenciamento do software e também no limite físico do hardware.

- Geralmente um repositório possui um endereço lógico que permite a conexão do cliente. Esse endereço pode ser um conjunto IP/porta, uma URL, um caminho do sistema de arquivos, etc.

Controle de Versão - Funcionamento Básico

- Cada desenvolvedor possui em sua máquina uma cópia local (também chamada de working copy em inglês) somente da última versão de cada documento.

- Essa cópia local geralmente é feita num sistema de arquivos simples (FAT, NTFS, ext3, etc).

- A cada alteração relevante do desenvolvedor é necessário "atualizar" as informações do servidor submetendo (commit em inglês) as alterações. O servidor então guarda a nova alteração junto de todo o histórico mais antigo.

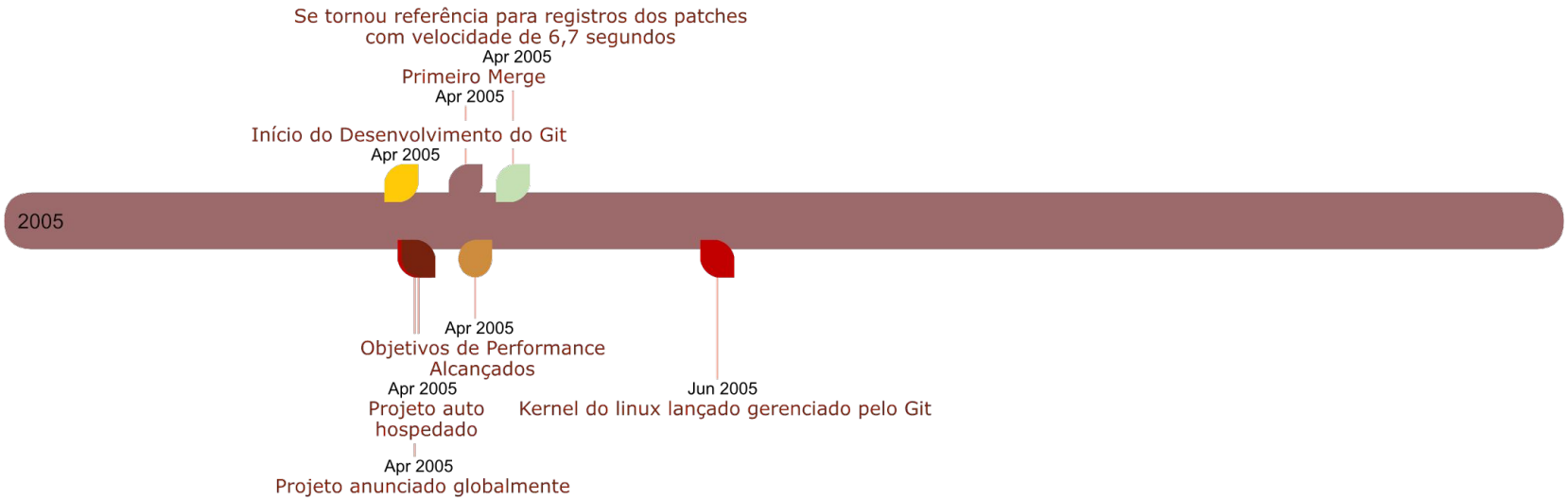
Git

- O desenvolvimento do **Git** surgiu após vários desenvolvedores do kernel (núcleo) do Linux decidirem desistir de acessar ao sistema do BitKeeper, que é um software proprietário.

- O acesso gratuito ao BitKeeper foi removido pelo detentor dos direitos autorais, Larry McVoy, depois de acusar Andrew Tridgell de usar de engenharia reversa nos protocolos do BitKeeper, alegando violação da licença do mesmo.

Git

Desenvolvimento do Git



Git

- O Git foi modelado como um conjunto de programas escrito em C como parte de um esforço de portar o Git para o Windows;

- Algoritmos inteligentes comunicam ao usuário que é incapaz de completar o merge automaticamente, sendo necessária uma edição manual.

- Abortar operações ou desfazer mudanças irá deixar objetos sem valor pendentes no banco de dados. Existe porém uma pequena fração desejável de objetos no sempre crescente histórico.

- Empacotamento periódico explícito de objetos

Git

- Como o BitKeeper, o Git não usa um servidor centralizado;
- O Git possui **duas estruturas de dados**: um índice mutável que provê informações sobre o diretório de trabalho e a próxima revisão a ser cometida; e um banco de dados de objetos de acréscimo imutável;
- O Git está primariamente desenvolvido para Linux, mas pode ser usado em outros sistemas operacionais baseados no Unix;
- O Git também roda no Microsoft Windows.

CONCEITOS IMPORTANTES

- ★ REPOSITÓRIO
  - Local onde os arquivos e suas cópias serão armazenados.
  - Local/Remoto
- ★ BRANCH
  - Ramos, cópias do código original .
  - Permite alterações de forma segura, sem afetar as funcionalidades.
- ★ MERGE
  - Fundir a cópia com o ramo principal.

CONCEITOS IMPORTANTES

- ★ PUSH REQUEST
  - Envio de modificações para o repositório central.
- ★ PULL REQUEST
  - Obtenção das modificações para a sua máquina.
- ★ FORK
  - Cópia de um repositório remoto para a máquina local.
  - Pegar um código público e utilizá-lo.

COMANDOS BÁSICOS

★ Configuração de Usuário (assinatura)

- Permite manter a rastreabilidade dos arquivos ao criar/alterar um arquivo.
- Abrir o cmd ou git-bash e digite:

```
$ git config --global user.name "Nome"
$ git config --global user.email "nome@email.com"
```

- Conferir se deu tudo certo:

```
$ git config --global --list
```

COMANDOS

★ Criando repositório local

- Crie uma pasta e adicione um arquivo “teste.txt”
- Navegue até o diretório através do cmd

```
$ git init
```

- Saída esperada:

```
Initialized empty Git repository in
caminho/.git
```

COMANDOS BÁSICOS

★ Verificar se está versionado

```
$ git status
```

- Saída para repositório vazio:

```
On ramo master

No commits yet

Arquivos não monitorados:
  (utilize "git add <arquivo>..." para
incluir o que será submetido)

    teste.txt
```

COMANDOS BÁSICOS

★ Adicionar o arquivo “teste.txt” ao repositório

```
$ git add teste.txt
$ git add --all
```

★ Confirmar inclusão

```
$ git commit teste.txt
```

COMANDOS BÁSICOS

★ Visualizar histórico de alterações

```
$ git log
$ git log --stat
```

★ Mudanças entre commit

```
$ git diff numerocommit1 numerocommit2
```

COMANDOS BÁSICOS

★ Recuperar versão específica do arquivo

```
$ git checkout numerocommit
```

★ Voltar para última versão (HEAD)

```
$ git checkout master
```

COMANDOS BÁSICOS

- Desfazendo alterações

```
$ git checkout teste.txt
```

- Desfazer todas as alterações

```
$ git reset --hard
```

IGNORAR ARQUIVOS

★ Criar na pasta do arquivo, um arquivo com o nome .gitgnore e adicione as extensões que não deseja rastrear.

```
*.exe
```

CLONAR REPOSITÓRIOS

- ★ Criar uma nova pasta fora do repositório
  - Acesse a pasta no prompt e digitar

```
$ git clone diretorioRepositorio
```

BRANCH DEVELOP

- ★ **Master**: versão de produção.
- ★ **Developer**: versão em desenvolvimento + novas features

BRANCH

★ **Criar branch**

```
$ git branch develop
```

★ **Visualizar branch**

```
$ git branch
```

★ **Selecionar branch**

```
$ git checkout develop
```

UNIÃO DE CODELINES

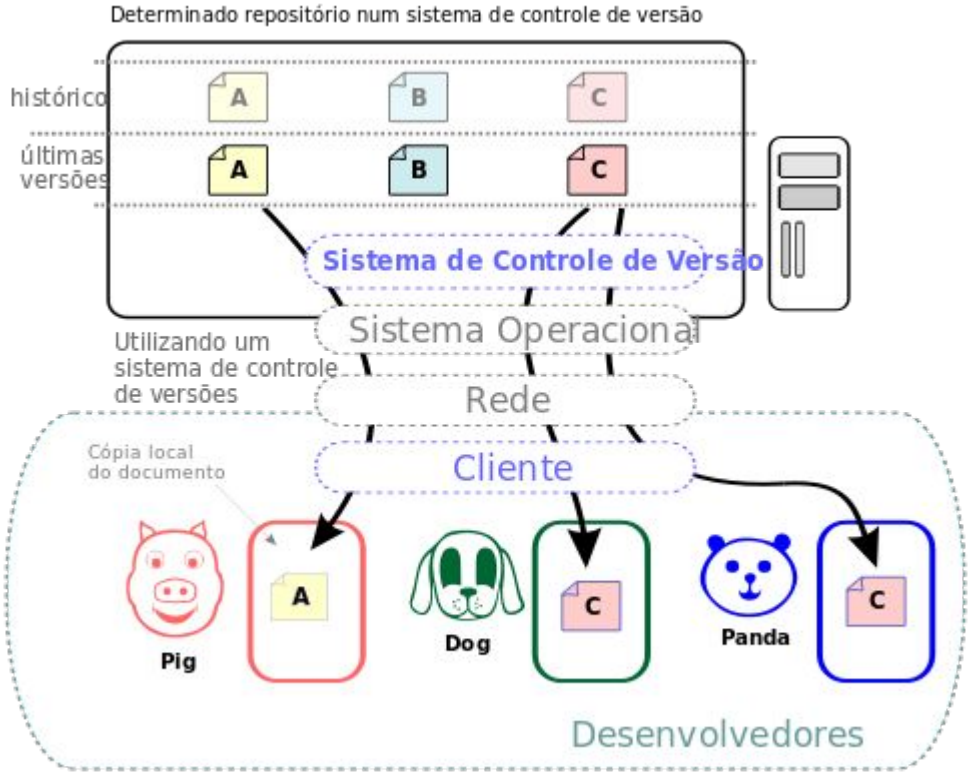
★ **Levar as alterações para a master**

```
$ git checkout master
$ git merge develop
```

★ **Enviar para o servidor**

```
$ git push
```

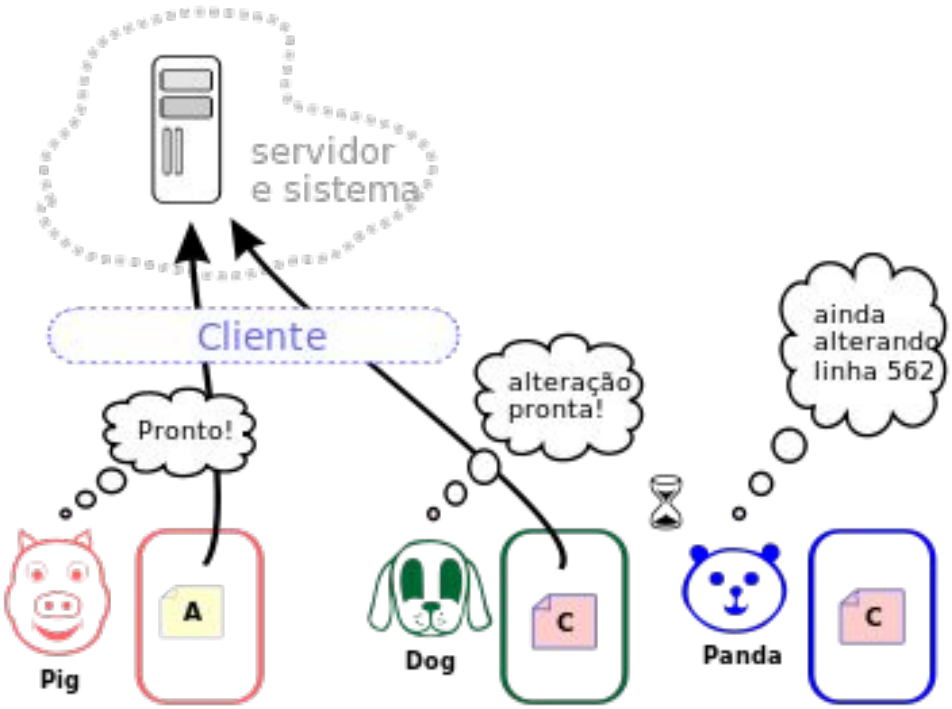
Na prática


Determinado repositório num sistema de controle de versão
histórico
últimas versões
A
B
C
A
B
C
Sistema de Controle de Versão
Sistema Operacional
Rede
Cliente
Utilizando um sistema de controle de versões
Cópia local do documento
Pig
A
Dog
C
Panda
C
Desenvolvedores

Na prática


alterando
linha 53
alterando
linha 100
alterando
linha 562
A
C
C
Pig
Dog
Panda

Na prática


servidor
e sistema
Cliente
Pronto!
alteração
pronta!
ainda
alterando
linha 562
A
C
C
Pig
Dog
Panda

Na prática


servidor
e sistema
Cliente
pronto!
enviarei!
Alerta! Desatualizad
Há uma versão
mais nova no
servidor! Atualize
primeiro.
Panda
C

Na prática


servidor
e sistema
Cliente
Preciso
atualizar
Panda
C
Aviso: Mesclando...
Acrescentando linha
100 de Dog na versão
local.
OK! Nenhum conflito

Na prática


servidor
e sistema
Cliente
Pronto!
Enviado uma
versão que é
a alteração de
Panda + a alteraçã
de Dog.
C
Panda

GITHUB

É O MESMO

QUE **GIT**?

Github

- O Github é um serviço online de hospedagem de repositórios Git (como são chamados os projetos que utilizam Git). Com ele podemos manter todos nossos commits e ramos sincronizados entre os membros do time.

- O GitHub é mundialmente usado e chega a ter mais de **36 milhões** de usuários ativos mundialmente contribuindo em projetos comerciais ou pessoais.

- Hoje o GitHub abriga mais de **100 milhões de projetos** alguns deles que são conhecidos mundialmente. WordPress, GNU/Linux, Atom, Electron.

Github

- O GitHub foi desenvolvido por Chris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon usando Ruby on Rails, e começou em fevereiro de 2008. A empresa, GitHub, Inc., existe desde 2007 e está localizada em São Francisco.

- Em 4 de junho de 2018, a Microsoft anunciou a compra da plataforma por US$ 7,5 bilhões, equivalente a R$ 27,225 bilhões com dólar a "3,63" dia 06/06/2018. Satya Nadella, diretor executivo da Microsoft, reafirma a nova postura da Microsoft frente ao código aberto

Outros softwares

- Gitlab
- Bitbucket
- Sourcetree

- Tortoise Git
- Git Bash
- GitHub Desktop