Ambientes de Desenvolvimento

GIT e GITHub

Rodolfo Riyoei Goya
rodolfo.goya@faculdadeimpacta.com.br

Sumário

- Referências
- GIT e Sistemas de Controle de Versão
- Terminologia
- GITHub
- CodeCommit e GIT na AWS

Materiais de referência

- https://www.udemy.com/course/git-e-github-para-iniciantes/
- https://www.amazon.com.br/Rys-Tutorial-English-Ryan-Hodson-ebook/dp/B00QFIA5OC
- https://git-scm.com/doc
- https://web.dio.me/browse/
- https://en.wikipedia.org/wiki/Git
- https://en.wikipedia.org/wiki/GitHub
- https://docs.aws.amazon.com/codecommit/latest/userguide/getting-started.html

GIT e GITHub

aws academy

GIT e GITHub

- GIT e GITHub são coisas distintas:

- GIT é um sistema para controle de versão de conjuntos de documentos
  - Gratuito e Open Source
  - Criado por Linus Torvald (o do Linux) como alternativa ao Bitkeeper em 2.005
  - Foi concebido para trabalho cooperativo
  - Ferramenta mais popular no mundo para desenvolvedores de software
  - Controle de versão mantendo todas as modificações em um tipo de banco de dados
  - Suporte para Rollback, ou seja, retorna a versões anteriores

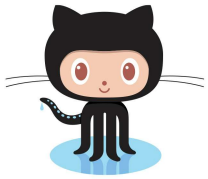
GIT e GITHub

- GIT e GITHub são coisas distintas:

- GITHub: provedor de serviço de hospedagem e controle de versão com GIT
  - Maior rede social de desenvolvedores do mundo (40 milhões de usuários)
  - Cerca de 200 milhões de repositório (30 milhões públicos)
  - Fundado em 2.008 e adquirido pela Microsoft em 2.018 (US$7.5 Bi)
  - Não apenas código:
    - Hub de discussões entre especialistas
    - Alertas de Segurança: Common Vulnerabilities and Exposures
    - Visualização de dados geoespaciais
    - Programas educacionais
    - EMOJIS

GIT e GITHub

- Usar GIT e GITHub é fonte de prestígio para desenvolvedores:
  - Habilidade e domínio de ferramenta amplamente adotada por empresas
  - Conhecimento de repositórios onde tecnologias e produtos são divulgados por fornecedores e universidade
  - Constitui portifólio de projetos e trabalhos desenvolvidos de um modo compartilhável
  - Habilita a explorar e participar de projetos de software livre disponibilizados para a comunidade



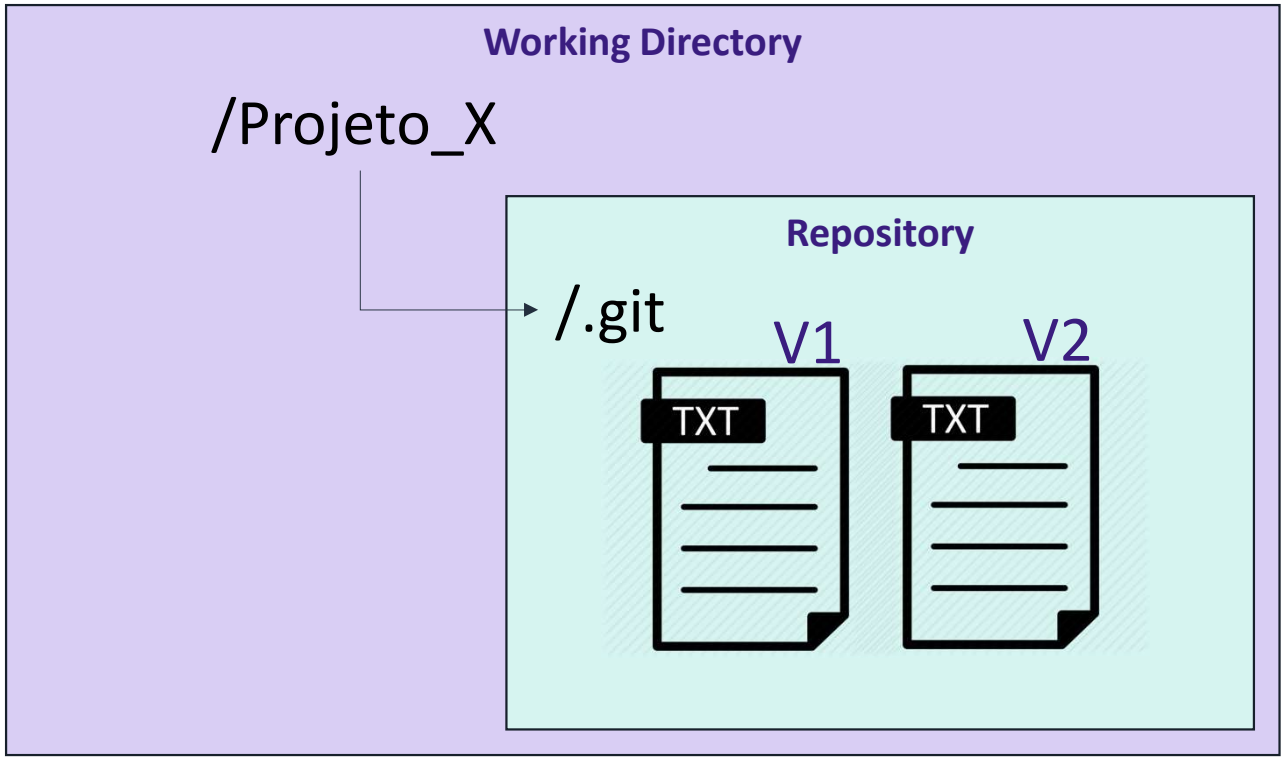
Introdução ao GIT

aws academy

Termos usados no GIT

- Repository: GIT gerencia documentos de um repositório. Armazenados em um diretório, constituem-se de arquivos, históricos e configurações
- Remote repository: GIT foi criado para suportar o uso de repositório conectado através de rede, por exemplo em um servidor da rede. O GITHub é um provedor de hospedagem de código que usa o GIT como tecnologia
- Branch: Um Branch no GIT é um modo que permite que um usuário (por exemplo um desenvolvedor) possa alterar documentos do repositório (por exemplo arquivos com módulos de programa fonte) sem afetar os demais documentos

Exemplo: Projeto_X gerenciado por GIT



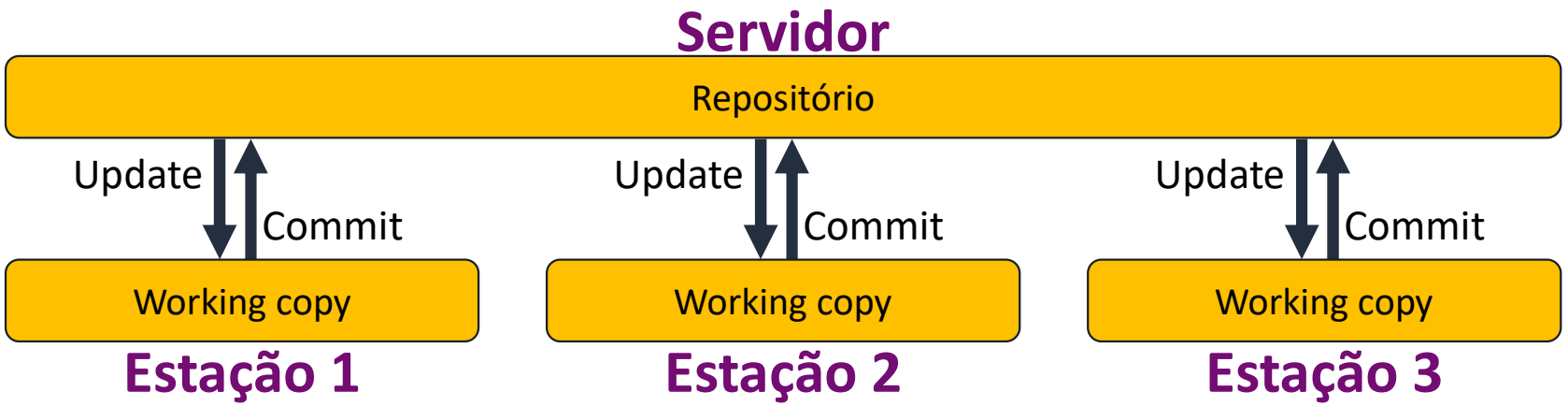
Áreas de um repositório

- Working Directory: Local onde arquivos são manipulados. GIT não monitora esse local. Também chamado de “Untracked Area”
- Staging Area: Local os onde arquivos são monitorados pelo GIT e as informações sobre mudanças são registradas
- Git Directory: Local onde os arquivos do GIT são salvos. Também chamado de “local repo”

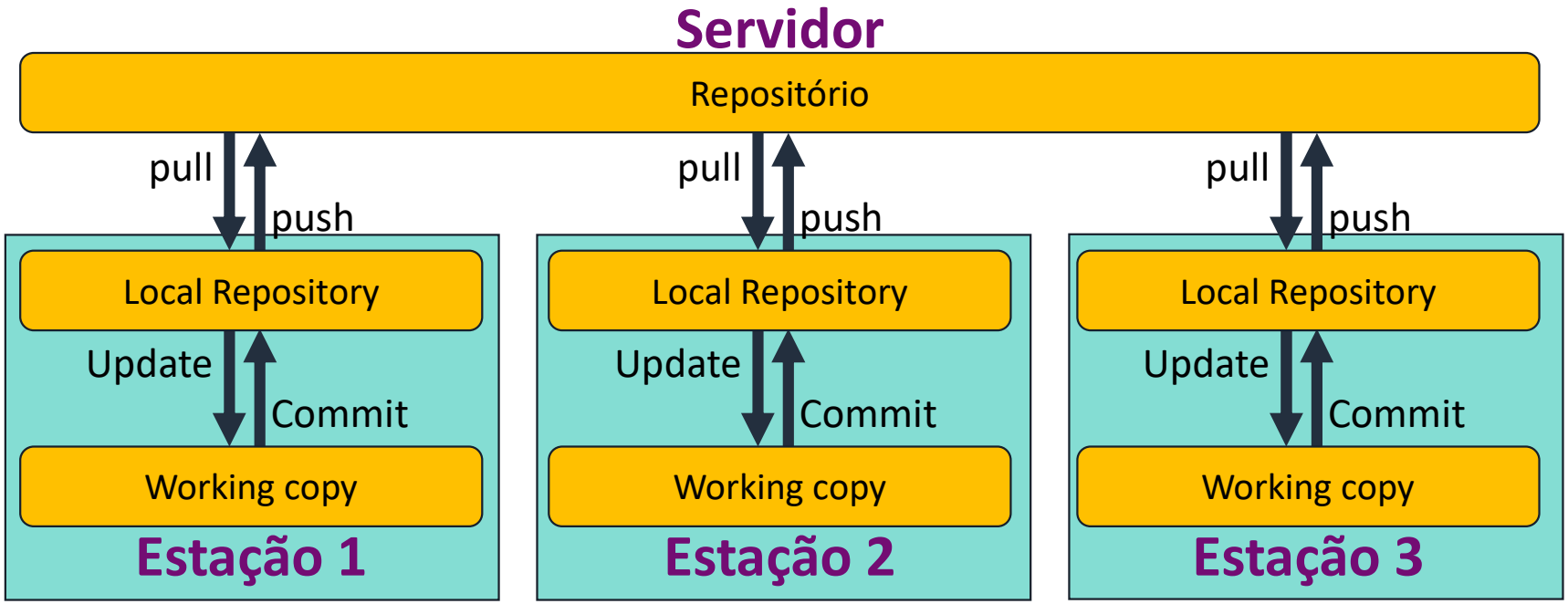
Tipos de Sistemas de Controle de Versão

- Centralizados:
  - CVCS: Centralized Version Control System
  - Um servidor mantém todos os documentos e intermedia toda a sua manipulação
  - O rastreio das mudanças é processado no servidor central
  - É desvantajoso por não disponibilizar os documentos localmente
- Distribuidos:
  - DVCS: Distributed Version Control System
  - Um repositório no servidor pode ser copiado e atualizado para cada estação local
  - O rastreio das mudanças é processado nas estações e sincronizado com o servidor
  - Operações mais rápidas (não requer comunicação) e confiáveis

CVCS


Servidor
Repositório
Update
Commit
Update
Commit
Update
Commit
Working copy
Working copy
Working copy
Estação 1
Estação 2
Estação 3

DVCS (Como o GIT)



Instalação do GIT

- Site oficial:

  https://git-scm.com/downloads

  git clone https://github.com/git/git

- Linux:
  - Muitas distribuições já vêm com GIT instalado

    git --version

    sudo apt-get install git-all

- Windows (alternativa):

  https://gitforwindows.org/

- MAC-OS (alternativa):

  https://downloads.sourceforge.net/project/git-osx-installer/git-2.23.0-intel-universal-mavericks.dmg

GIT - Configuração inicial

- Início:
  - Console GIT Bash

```
git help
git help <tópico>
git update-git-for-windows
```

- Nome e email:
  - Necessário para documentar autores de mudanças de versões (commit)
  - Comando

```
git config --global user.name “NOME DO USUARIO"
git config --global user.email “NOME@DOMINIO"
```

GIT - Criar novo repositório

- Início:
  - Criar um diretório para o projeto e, de dentro do diretório, executar um “git init”

```
mkdir <Nome do projeto>
cd <Nome do projeto>
git init
```

  - Crie um bom arquivo README.md (não é obrigatório, mas recomendado)

https://www.freecodecamp.org/news/how-to-write-a-good-readme-file/
https://docs.gitlab.com/ee/development/documentation/styleguide/

GIT - Clonar repositório existente

- Clone:
  - Pode-se criar um repositório local como cópia de repositório público
  - O próprio processo de clonagem cria o diretório que contém o repositório
  - Comando

```
git clone <URL do repositório>
git clone https://github.com/git/git
```

GIT - Commit de um documento

- Todo novo documento está no “Working Area” sem monitoramento do GIT
- Colocando o documento no “Staging Area”:

```
git add <Nome do documento>
```

- Colocando todos os documentos no “Staging Area”:

```
git add .
```

- Examinando o status dos documentos:

```
git status
```

- Atualizando os documentos do “Staging Area” para o GIT:

```
git commit
git commit –m “O que esta mudando nessa atualização mesmo?”
```
(boa prática)

```
git commit –am “O que esta mudando nessa atualização mesmo?”
```
(add+commit)

GIT - Gerenciando mudanças

- Modificações nos documentos do “Working Area” com “Staging Area”:

```
git diff <nome do arquivo>
git diff
```

- Desfazendo modificações em um documento do “Working Area”:

```
git checkout <nome do arquivo>
```

- Removendo documento do “Staging Area”:

```
git reset <nome do arquivo>
git reset HEAD <nome do arquivo>
```

GIT - Gerenciando mudanças

- Exibindo o relatório dos últimos 10 commits:

```
git log
git log --oneline
git log --author="Rodolfo Goya"
git log --author=rgoya@uol.com.br
git config --global credential.helper cache
git shortlog
git shortlog -sn
```

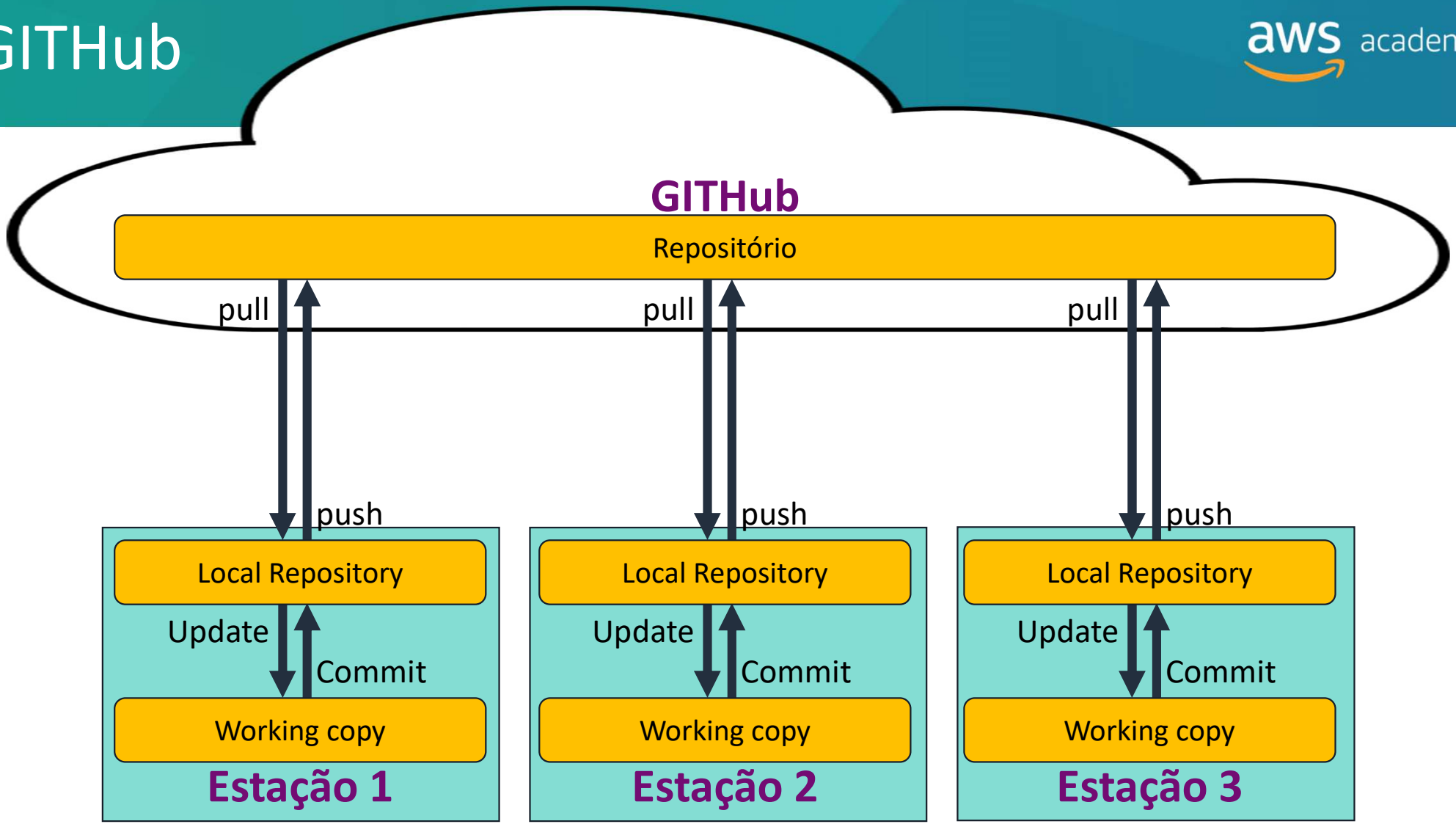
  - Veja mais em:

```
git help log
```

GITHub

aws academy

GITHub


GITHub
Repositório
pull
pull
pull
push
push
push
Local Repository
Local Repository
Local Repository
Update
Commit
Update
Commit
Update
Commit
Working copy
Working copy
Working copy
Estação 1
Estação 2
Estação 3

GITHub – Introdução

- Localização

  https://github.com

- Criar um novo usuário e uma identificação única
  - A identificação cria uma URL dentro da qual ficarão seus repositórios:

    https://github.com/<Identificação>

    https://github.com/<Identificação>/<Nome do projeto>

- Criar um novo repositório
  - Escolha de um novo “Nome do projeto”
  - Definir como público (free/pode ser clonado) ou privado (pago)
  - .gitignore: Arquivos que devem ser ignorados pelo GIT (há templates para a lista)
  - Crie um bom arquivo README
  - Arquivo com instruções sobre o licenciamento

GITHub – Ligando com repositório local

- Criando repositório local como branch de repositório no GITHub:

```
git clone –b main https://github.com/<Identificação>/<projeto>
```

- Atualizando modificações no GITHub para o repositório local:

```
git pull
```

- Atualizando modificações no repositório local para o GITHub:

```
git push
```

  - Requer autenticação (proteção para atualizações):
    - Autenticação manual a cada atualização
    - Baixar e instalar chave SSH ou GPG para autenticação automática pelo GIT
    - https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh
    - https://docs.github.com/en/github/authenticating-to-github/managing-commit-signature-verification

GITHub – Configuração de chaves de acesso

- Checar se já não há chaves geradas:

```
ls -al ~/.ssh
```

- Gerar o par de chaves (pública e privada):

```
ssh-keygen -t rsa 4096 –C “USER@DOMINIO”
ssh-keygen -t ed25519 -C “USER@DOMINIO”
```

- Instalar a chave pública (arquivo *.pub):
  - em https://github.com/settings/keys
  - New SSH keys
- Testar se a chave instalada funciona:

```
ssh -T git@github.com
Hi <USER>! You've successfully authenticated, but GitHub does not provide
shell access.
```

GITHub – Configuração de chaves de acesso

- Credenciais:
  - Criação:
    - Settings/Developers settings/Personal access token/Generate new token
    - Copiar o token gerado
- Instalação:

```
git push
Username for 'https://github.com': <username>
Password for 'https://rrgoya@github.com': <token>
git config --global credential.helper cache
```

GIT – Branches

- Criação de uma linha secundária no desenvolvimento do projeto
- Versão “congelada” (release)
- Separação para correção de erro (“hot fix”)
- Desenvolvimento de novas funcionalidades (“features”)
- Desenvolvimento para novas plataformas
- Documentação e manuais

GIT – Branches - Comandos

- Exame do branch atual (default: main):

```
git branch
git branch -a
```

- Criação de um novo branch:

```
git branch <Nome do novo branch>
```

- Mudança para um novo branch:

```
git switch <Nome do branch>
git checkout <Nome do branch>
```

- Mudança de nome do branch:

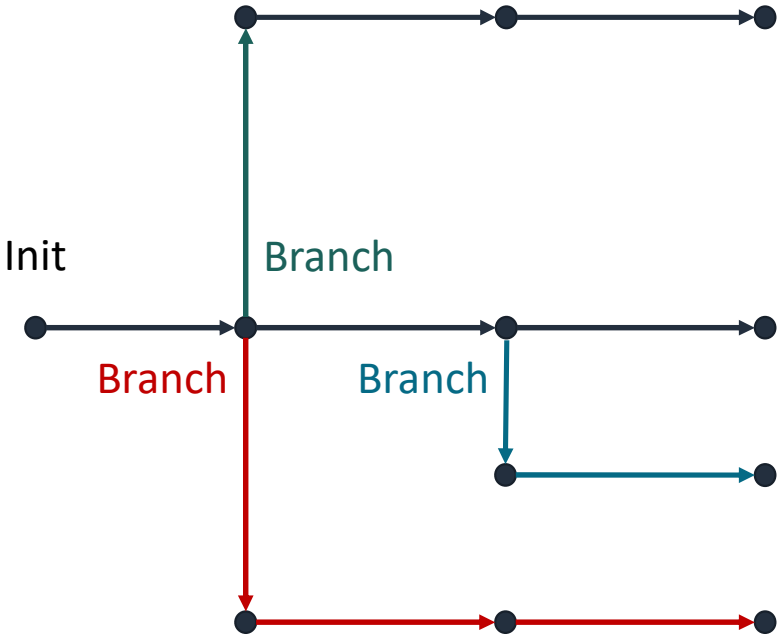
```
git branch –m <Nome do branch> <Nome do novo branch>
```

- Remoção de branch:

```
git branch –d <Nome do branch>
```

GIT – Branches

- Branches definem linhas de trabalho independentes
- Desenvolvedores distintos
- Grupos de trabalho separados
- Features
- Tipos de documento



GIT – Branches - Comandos

- Exibe branches:

```
git log --graph
```

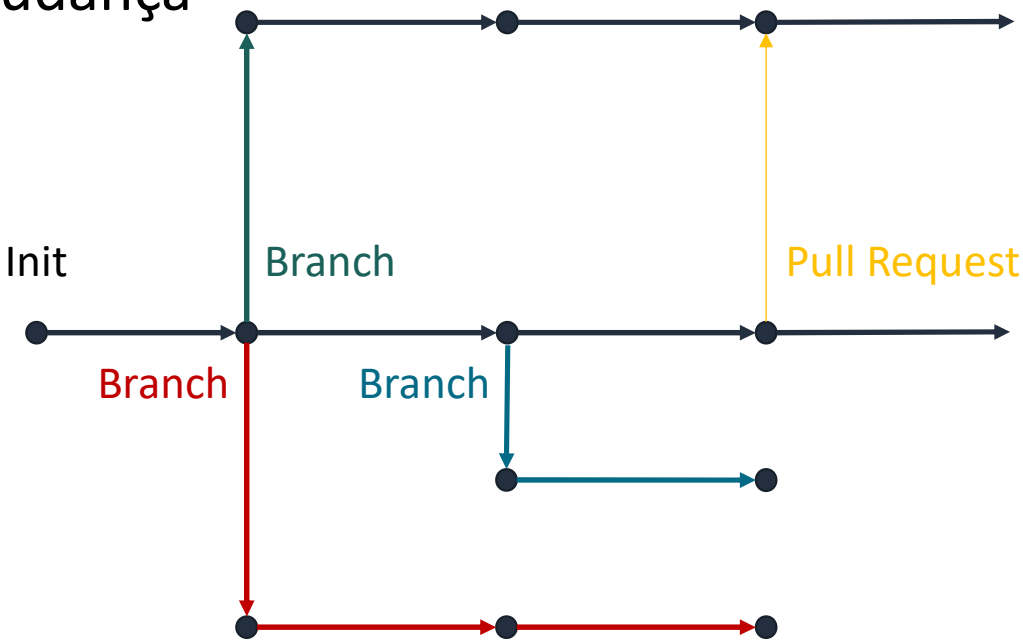
- Exibe commits:

```
git show <hash de commit>
```

GIT – Pull request

- Atualização de um branch com modificações de outro branch
- Feita do branch que recebe a mudança

```
git pull origin main
```



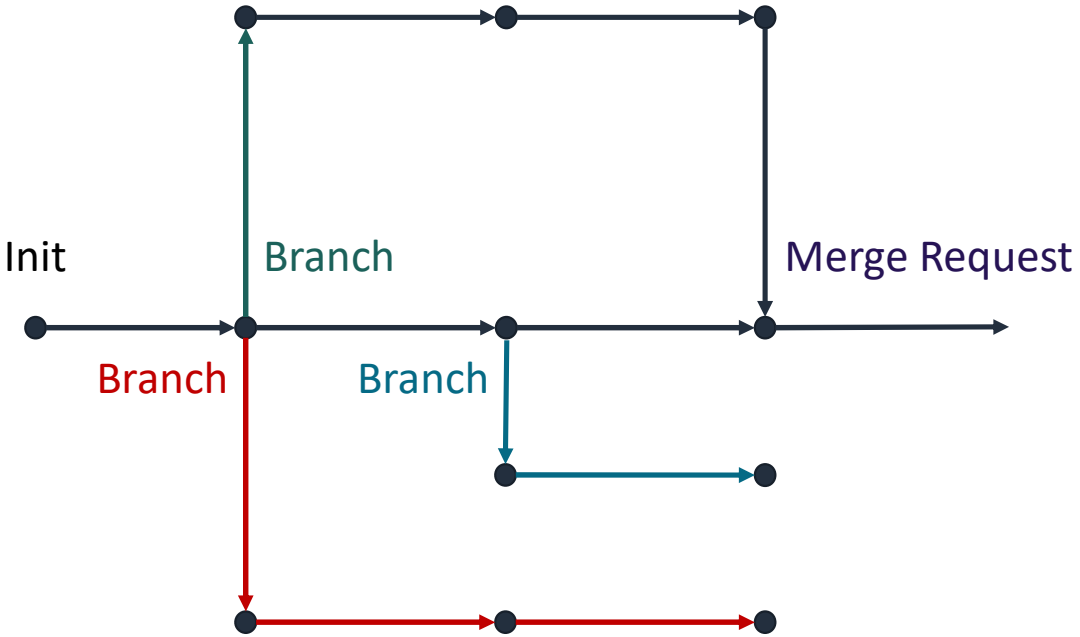
GIT – Merge request

- Modificações desenvolvidas em um branch (novas funcionalidades, correções, etc) necessitam ser incorporadas ao programa principal
- GIT dispõe de comandos para unificar branches
- Documentos modificados no branch a ser extinto são atualizados no branch que receberá as modificações
- A maioria das modificações deve ser automaticamente incorporada, mas modificações conflitantes (por exemplo, feitas em ambos os branches) dever ter seu destino resolvido manualmente

GIT – Merge request

- Comandos (a partir do branch que RECEBE as mudanças):

```
git pull origin <nome do branch>
git merge
git commit
```



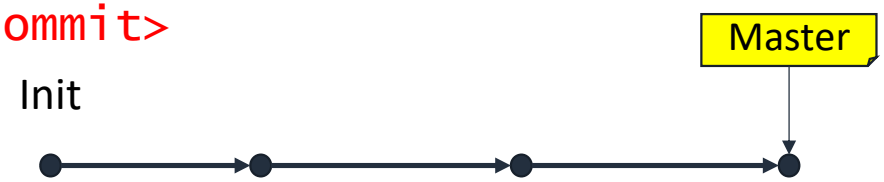
GIT – Desfazendo modificações

- Problemas podem ser descobertos após um “commit”, deste modo, deve haver modos para desfazer modificações
- Reset:
  - --soft: Desfaz o commit e traz os arquivos de volta para o “Staging Area”
  - --mixed: Desfaz o commit e traz os arquivos de volta para o “Modified Area”
  - --hard: Desfaz o commit e traz os arquivos de volta para o commit anterior
  - Como se fosse um “Rollback”
  - Mais adequado para repositório local
- Revert:
  - Cria um novo commit com mudanças para o estado anterior
  - Melhor usado quando o commit já atualizou o repositório remoto

GITHub – Reset e Revert

- Reset: Faz com que o master passe a apontar para o estado anterior

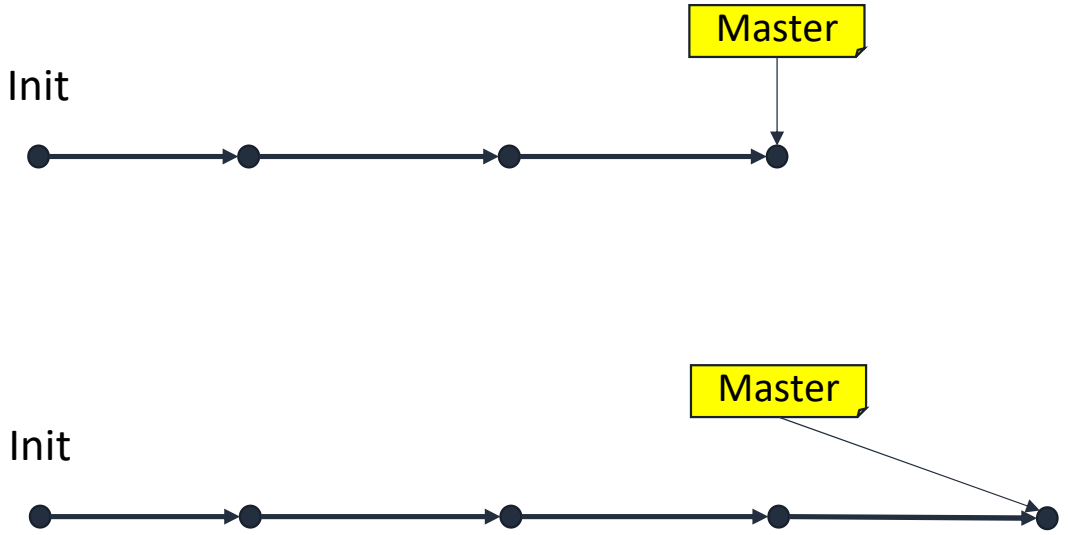
```
git reset --soft <id do commit>
git reset --hard <id do commit>
git log --oneline
```



GITHub – Reset e Revert

- Revert: Faz com que o master passe a apontar para o estado anterior

```
git revert HEAD
git push
git log --oneline
```



GITHub – Comparando commit

- Pode-se obter as diferenças entre commits

```
git log --oneline
git diff <id de commit 1> <id de commit 2>
```

CodeCommit

- Serviço de repositório GIT remoto hospedado na nuvem da AWS
- Serviço integrado ao IAM e mantido privado a baixo custo
- Repositório de código integrado à plataforma de desenvolvimento
- Repositório de código integrado à plataforma de processamento
- Comunicação em https, ssh e git

Conclusões

GIT é a ferramenta mais popular para gerenciamento de versões de código usada por desenvolvedores

GITHub é o serviço de hospedagem de código e GIT mais usado no mundo

Dúvidas?