Universidade Federal do Acre
Programa de Pós-Graduação em Ciência da Computação

Curso de Git

Prof. Manoel Limeira de Lima Júnior

Agenda

- Introdução
- Comandos Básicos
- Repositórios Remotos
- *Branches*
- *Workflows*
- GitHub
- Ferramentas

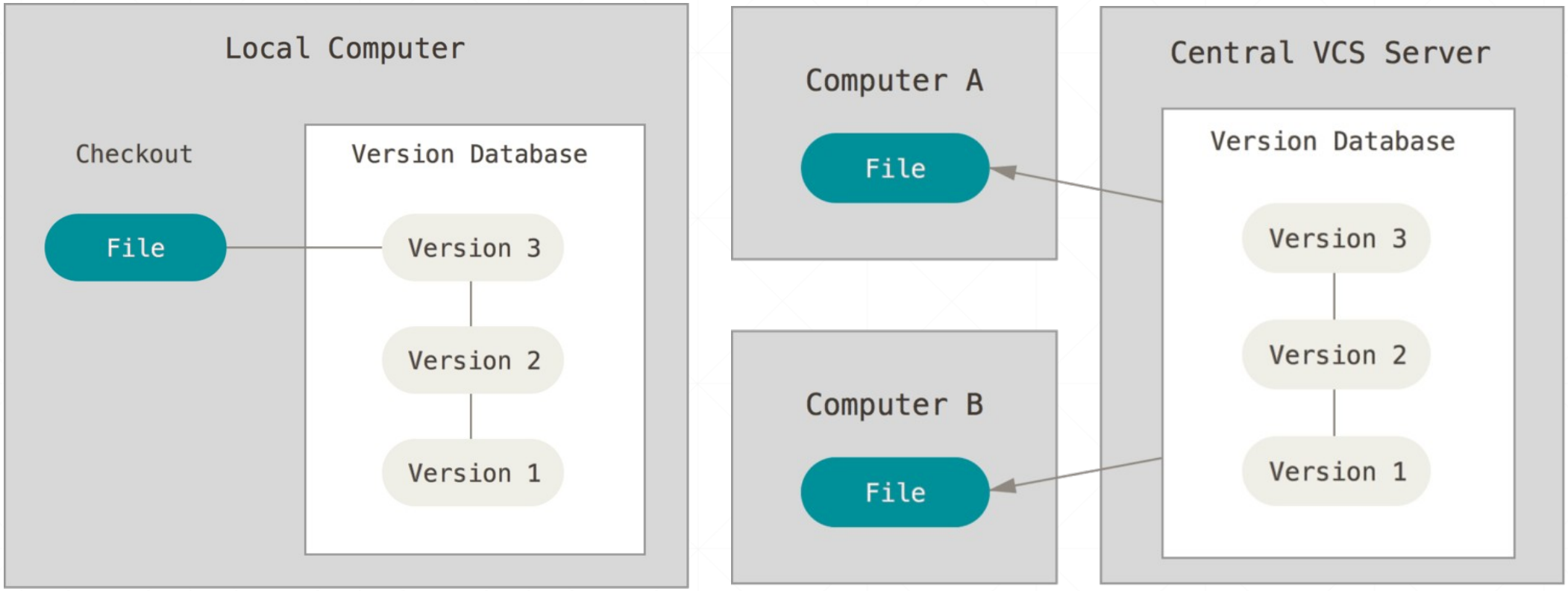
Introdução – Conceito e Motivação

- Sistemas de Controle de Versão
- “Controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas.”

- O que pode ser versionado?
- Praticamente tudo, exemplos: livros, imagens, código fonte, documentos (dissertação).

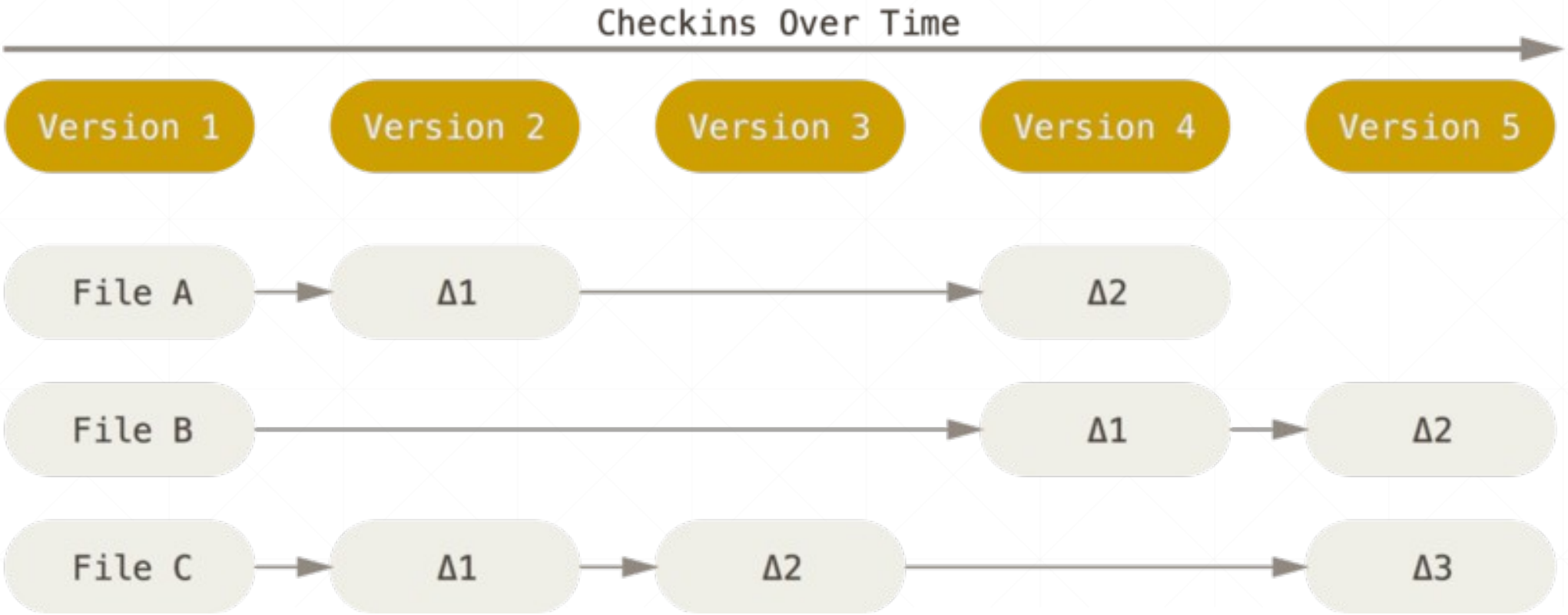
Introdução - Conceito

- Sistemas de Controle de Versão Centralizados



Introdução - Conceito

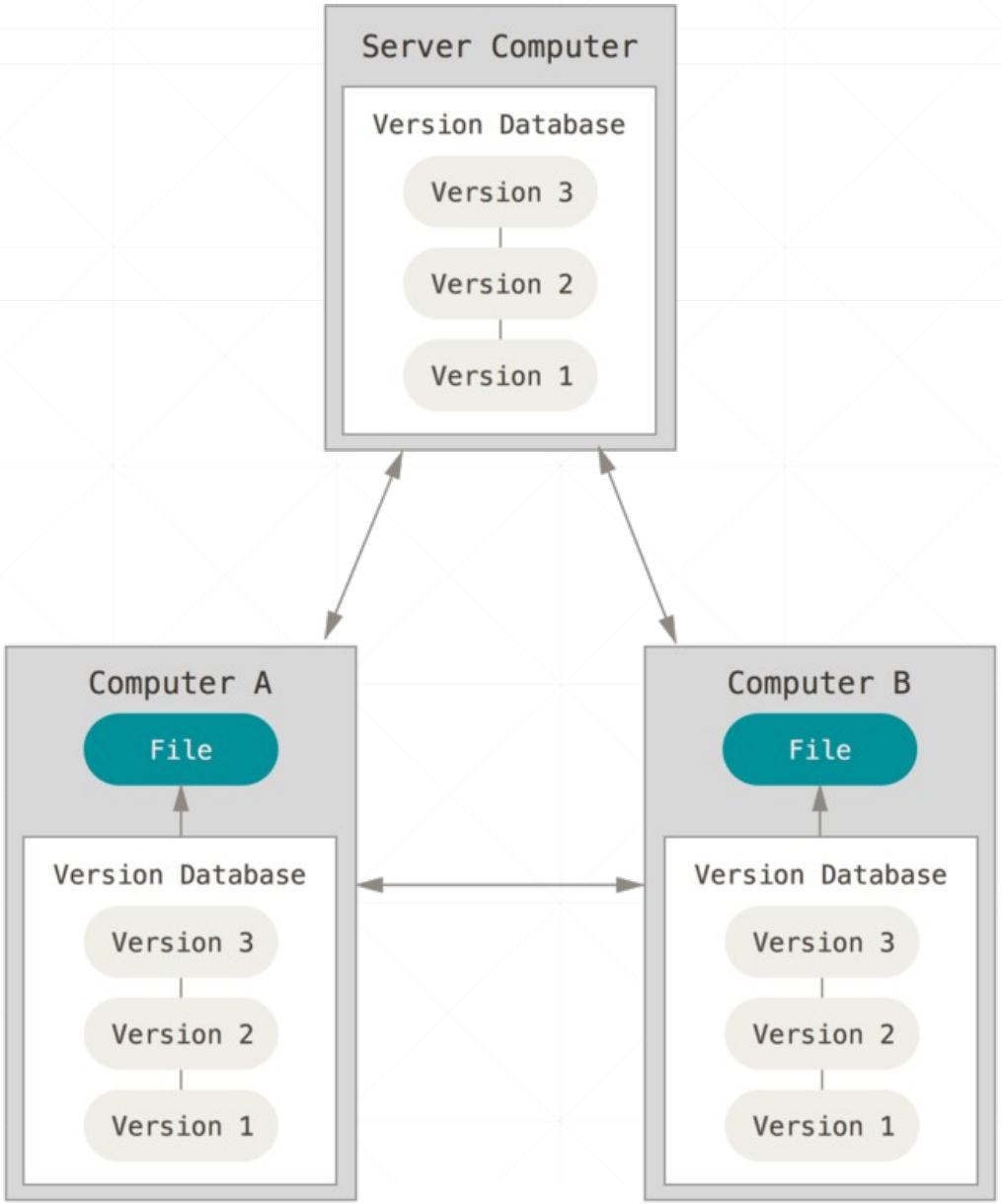
- Sistemas de Controle de Versão Centralizados



Introdução - Conceito

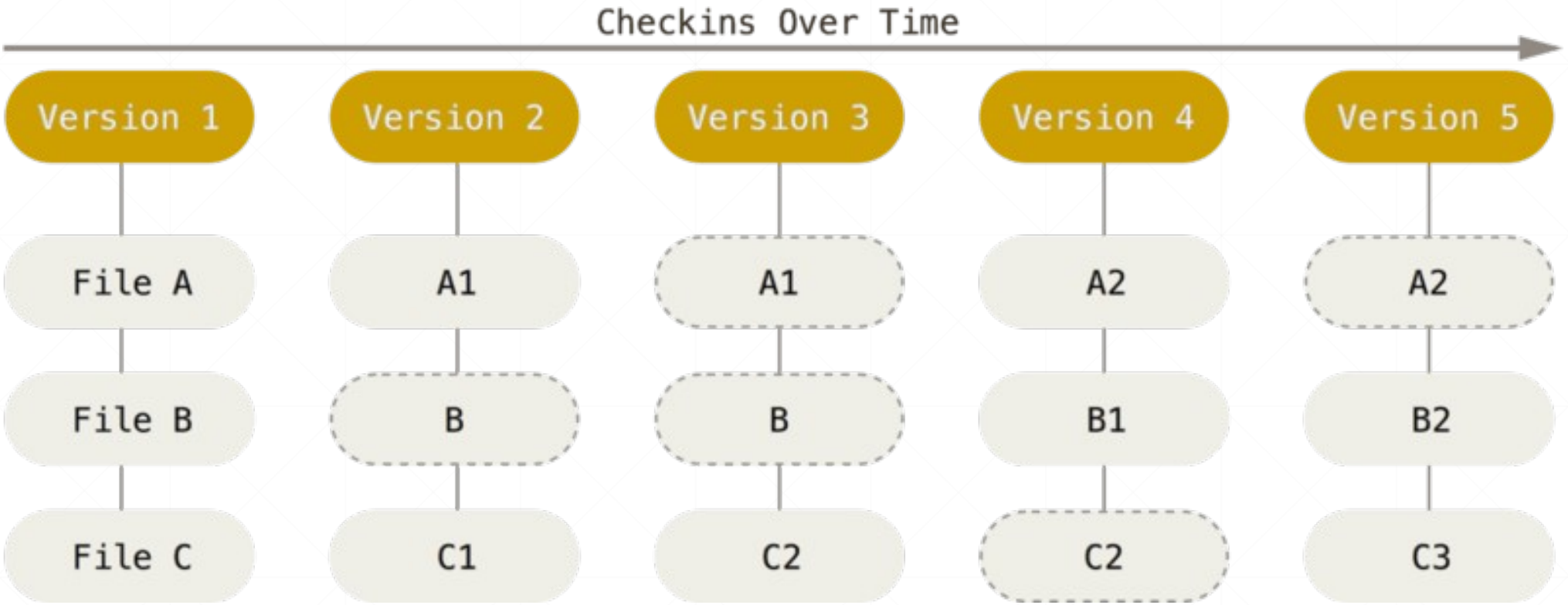
- Sistemas de Controle de Versão Distribuídos

git



Introdução - Conceito

- Sistemas de Controle de Versão Distribuídos

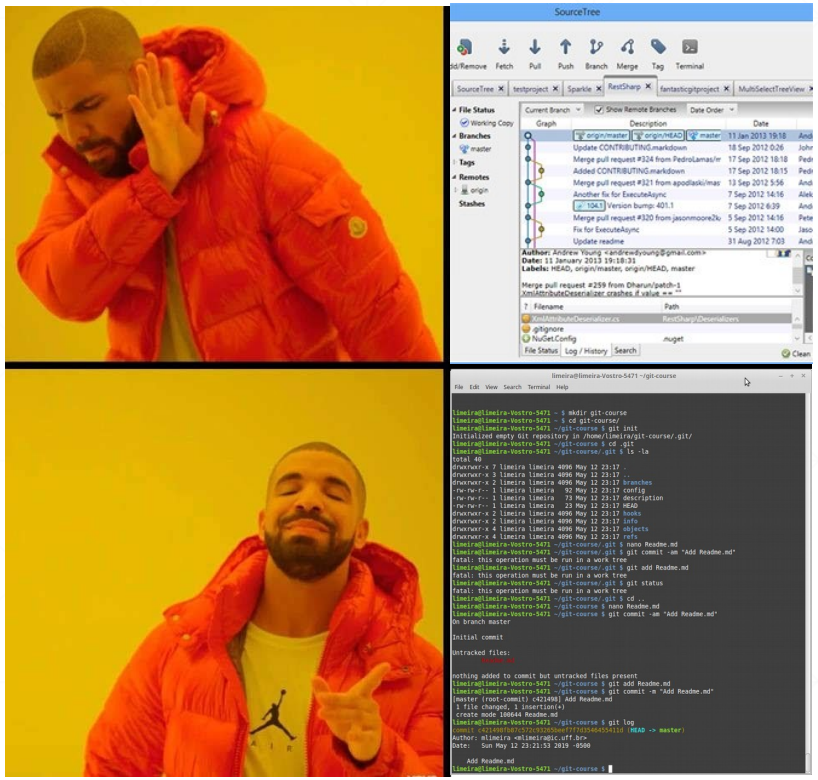


Introdução - História

- Em 2002, o projeto do núcleo do Linux começou usar uma ferramenta proprietária chamada BitKeeper.
- Em 2005, Linus Torvalds começou a desenvolver a sua própria ferramente com alguns objetivos em mente:
  - Velocidade
  - Projeto simples
  - Suporte para desenvolvimento não-linear (*branches*)
  - Completamente distribuído
  - Capaz de lidar com projetos grandes

Comandos Básicos – Git

- Download Git: https://git-scm.com/downloads
  - Mac
  - Linux
  - Windows
- Linha de Comando



Comandos Básicos – Git

- Configuração inicial

```
$ git config --global user.name “Manoel Limeira”
$ git config --global user.email “mlimeira@ic.uff.br”
$ git config --global core.editor nano
```

- Verificar

```
$ git config user.name
$ git config --list
```

- Ajuda

```
$ git help config
```

Comandos Básicos – Git

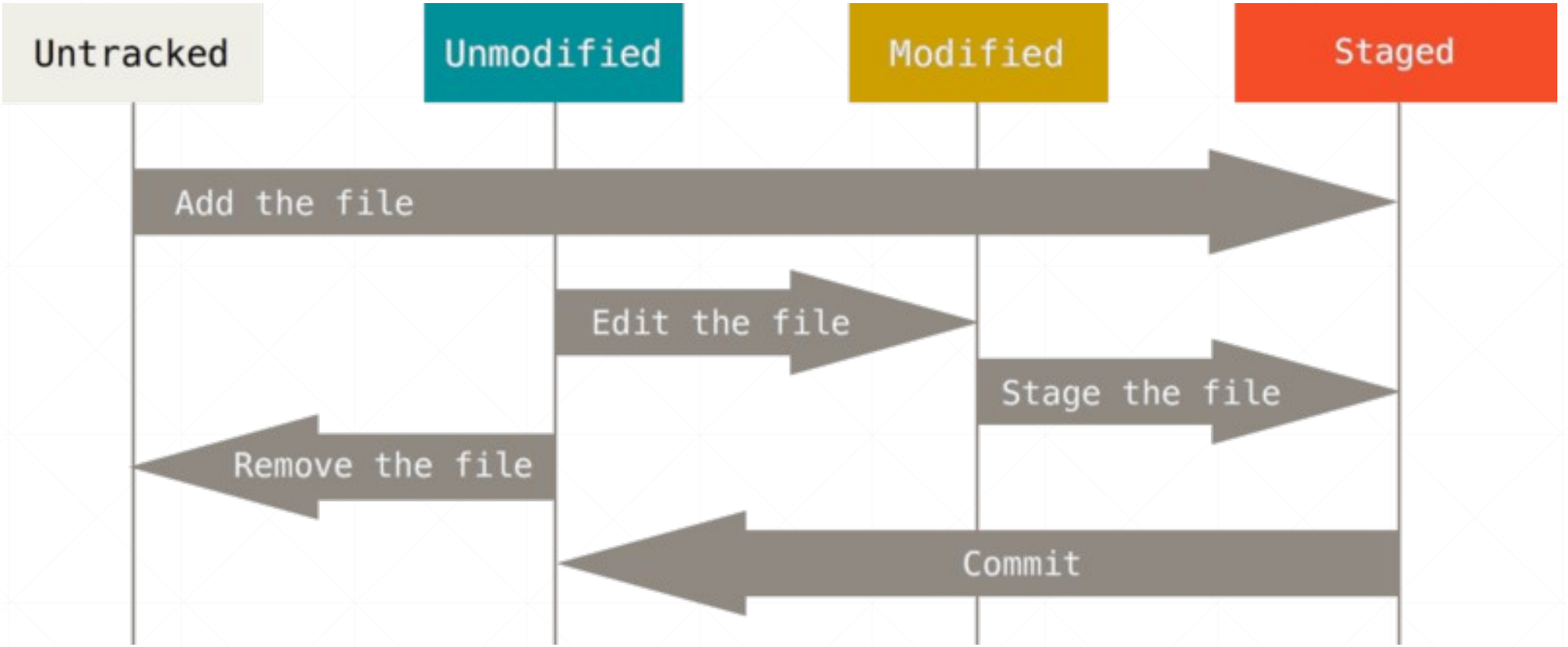
- Inicialização de um repositório

```
$ mkdir curso-git //cria um diretório
$ cd curso-git //ativa um diretório
$ git init //inicializa o versionamento do diretório
$ ls -la //lista o conteúdo do diretório
```

- Informações sobre o repositório (diretório **.git**)

```
$ cd .git
$ ls
branches  config  description  HEAD  hooks  info  objects  refs
```

Ciclo de Vida dos Arquivos – Git



Comandos Básicos – Git

- Criar e alterar um arquivo para ser versionado

```
$ cd curso-git //diretório controlado pelo git
$ nano Readme.md //pode usar qualquer editor
Ctrl + o //escreve no arquivo
Ctrl + x //encerra o editor
```

- Ocultar arquivos do versionamento

```
$ cd .git/info
$ nano exclude //arquivo (file.cpp) ou regra (*.class)
```

Comandos Básicos – Git

- Verificação do *status* de um repositório

```
$ git status
```

- Adicionar uma versão de um arquivo para a área transferência

```
$ git add Readme.md
```

- Confirmar uma versão de um arquivo no repositório

```
$ git commit -m “Add Readme.md”
$ git status
```

Comandos Básicos – Git

- Verificar *log* do repositório

```
$ git log [--decorate,--graph,--stat,author=“Manoel”]
$ git log --pretty=oneline
$ git shortlog
```

- Mostrar um *commit* do repositório

```
$ git show 0856ee55ba88084595e54dfb5857a55c3407af3a
```

- Verificar diferenças no repositório

```
$ git diff
```

Comandos Básicos – Git

- Desfazer alterações no repositório (modo de edição)

```
$ git checkout Readme.md
$ git diff
$ git status
```

- Desfazer alterações (modo de transferência)

```
$ git reset HEAD Readme.md
```

- Desfazer alterações (modo de versões)

```
$ git reset [--soft,--mixed,--hard] hash
```

Repositórios Remotos – GitHub

- Criar ou acessar uma conta (`github.com`)
- Criar um repositório (**`user`**/**`repository`**)
- Enviar as versões para o repositório remoto

```
$ git remote add origin https://github.com:user/repository.git
$ git push -u origin master
$ Username for 'https://github.com': user
$ Password for 'https://user@domain@github.com': password
```

Repositórios Remotos – GitHub

- Criar ou acessar uma conta (`github.com`)
- Criar um repositório (**`user`**/**`repository`**)
- Criar uma chave de acesso (https://help.github.com/en/articles/about-ssh)

```
$ cd ~/.ssh (no linux)
$ ssh-keygen -t rsa -b 4096 -C "user@domain.com"
```

- Adicionar chave nas configurações do GitHub
- Enviar as versões para o repositório remoto

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

Repositórios Remotos – GitHub

- Mostrar repositório remoto

```
$ git remote show origin
```

- Renomear repositório remoto

```
$ git remote rename origin origin-new
```

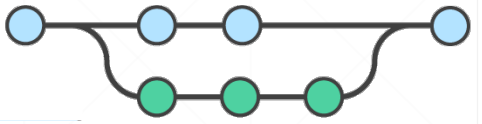
- Remover repositórios remotos

```
$ git remote remove origin-new
```

- Clonar repositórios remotos

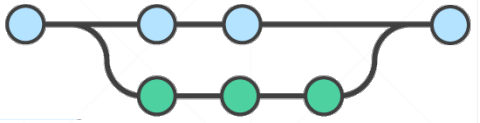
```
$ git clone git@github.com:user/repository repository-clone
$ cd repository-clone
$ cat Readme.md
```

Branches

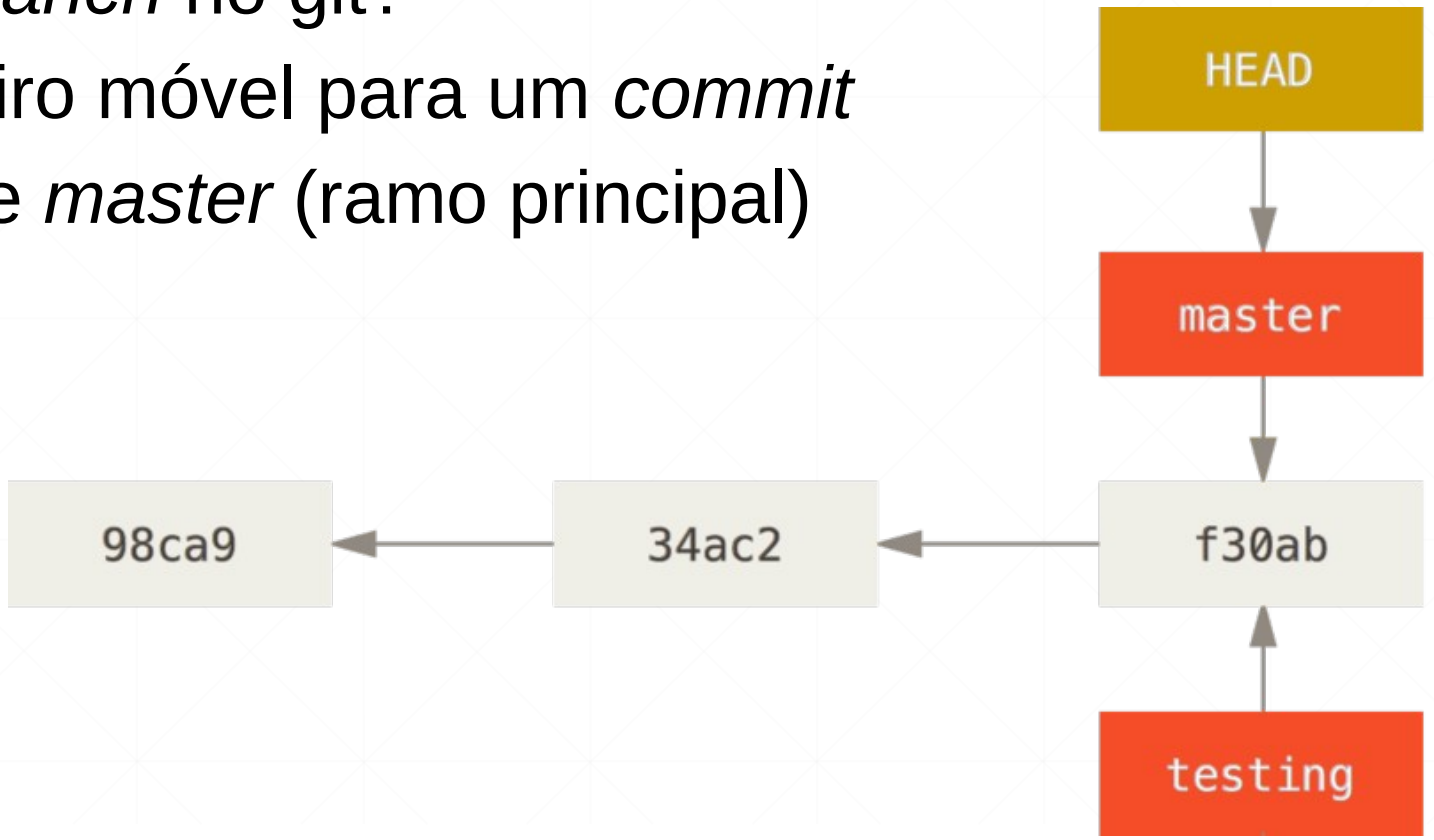


- Por que usar *branches* no git?
  - É leve e facilmente desligável
  - Modificar os arquivos sem alterar o *branch master*
  - Permite o trabalho de múltiplas pessoas no projeto
  - Evita alguns conflitos
  - Fornece ferramentas para gerenciar outros conflitos

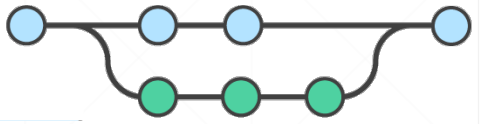
Branches



- O que é um *branch* no git?
  - É um ponteiro móvel para um *commit*
  - Ex: *testing* e *master* (ramo principal)



Branches



- Criar *branches*

```
$ git branch v1 ou git checkout -b testing
```

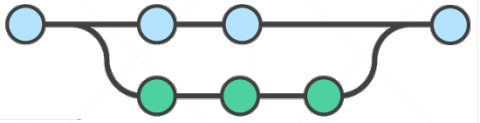
- Listar *branches*

```
$ git branch
```

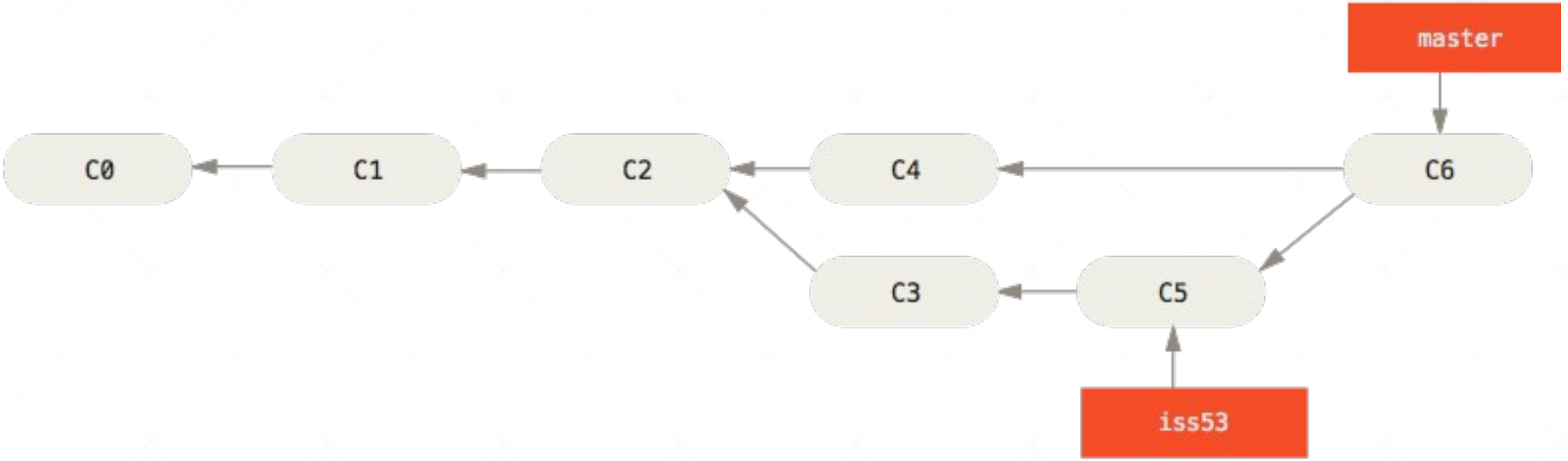
- Alternar para entre os *branches*

```
$ git checkout testing
```

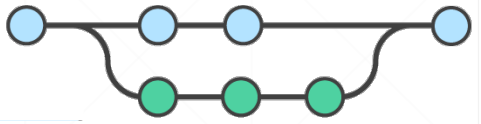
Branches



- Exemplo: Criar um *branch* para resolver um problema no software, retornar para o *master* e continuar o trabalho



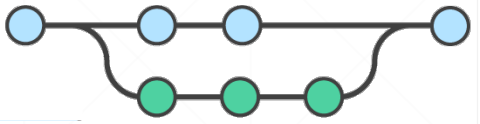
Branches



- Código para o cenário **sem** conflito (*fast-forward*)

```
$ git checkout -b issue71
$ nano arq1.txt //editar o conteúdo de arq1.txt
$ git add . //adicionar arquivos para área de transferência
$ git commit -m “add arq1”
$ git checkout master //alterar outros arquivos no master
$ git merge issue71
$ git log --graph
$ git branch -d issue71 //passo opcional
```

Branches

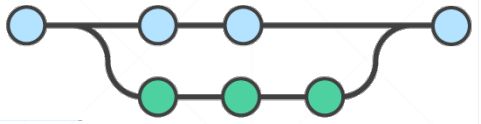


- Código para o cenário **com** conflito (recursive-*strategy*)

```
$ git checkout -b issue72
$ nano arq1.txt //editar o conteúdo de arq1.txt
$ nano arq2.txt //editar o conteúdo de arq2.txt
$ git add . //adicionar arquivos para área de transferência
$ git commit -m “add arq1”
$ git checkout master //alterar arq1.txt no master
$ git merge issue72 //exibe o conflito que deve ser resolvido
$ nano arq1.txt //resolver o conflito
$ git commit -am “merge issue72”
```

```
$ git log --graph --oneline
```

Branches



- Exercício
  - O objetivo é colaborar no desenvolvimento de um aquivo
  - O arquivo será o readme.md do repositório mlimeira/curso-git
  - O arquivo deve conter o conteúdo visto nas aulas de git
  - Cada aluno deve:
    - Criar um *branch* com o seu nome
    - Escolher um tema/assunto/comando
    - Realizar as alterações localmente
    - Enviar as versões para o repositório remoto

Universidade Federal do Acre
Programa de Pós-Graduação em Ciência da Computação

Desenvolvimento Distribuído de Software: perspectivas e oportunidades com Mineração de Dados

Prof. Dr. Daricélio Soares
Prof. Dr. Manoel Limeira