INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

CURSOS DE VERÃO

Sistema de Controle de Versão

- Prof. Dra. Paloma Oliveira
- Email: paloma.oliveira@ifmg.edu.br

Sistema de Controle de Versão

- O que é?
- Você já utilizou?

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

O que é?

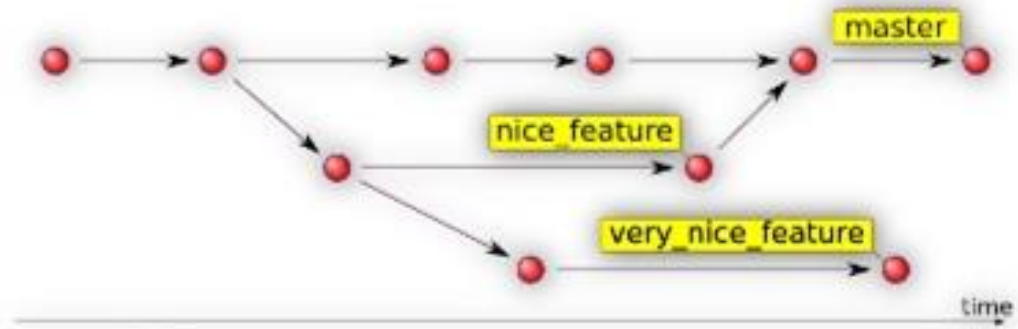
- O Sistema de Controle de Versões (SCV ou CVS)
- **local para armazenamento de artefatos**
  - Que tipos de artefatos?
- É um software para gestão das várias versões de seus arquivos

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Sistema de Controle de Versão

1 Manter histórico

2 Colaboração



3 Variações do projeto

Sistema de Controle de Versão

- Permite

– reverter arquivos para um estado anterior;

– reverter um projeto inteiro para um estado anterior;

– comparar mudanças feitas ao decorrer do tempo;

– ver quem foi o último a modificar algo que pode estar causando problemas;

– quem introduziu um bug e quando, e muito mais.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Sistemas de Controle de Versão

Centralizado		Distribuído	
Livre	Comercial	Livre	Comercial
SCCS(1972)	CCC/Harvest(1977)	GNU arch(2001)	TeamWare(199?)
RCS(1982)	ClearCase(1992)	Darcs(2002)	Code co-op(1997)
CVS(1990)	Sourcesafe(1994)	DCVS(2002)	BitKeeper(1998)
CVSNT(1998)	Perforce(1995)	SVK(2003)	Plastic SCM(2006)
Subversion(2000)	TFS(2005)	Monotone(2003)	
		Codeville(2005)	
		Git(2005)	
		Mercurial(2005)	
		Bazaar(2005)	
		Fossil(2007)	

INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

Conceitos

- Termos que são comuns a SCV:

- **repositório**: que é o local de armazenamento de todas as versões dos arquivos;

- **versão**: que representa o estado de um item de configuração que está sendo modificado. Toda versão deve possuir um identificador único, ou VID (Version Identifier);

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Conceitos

- **espaço de trabalho:** espaço temporário para manter uma cópia local da versão a ser modificada. Ele isola as alterações feitas por um desenvolvedor de outras alterações paralelas, tornando essa versão privada;

- **check out (clone)**: que é o ato de criar uma cópia de trabalho local do repositório;

- **update:** o ato de enviar as modificações contidas no repositório para a área de trabalho;

INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

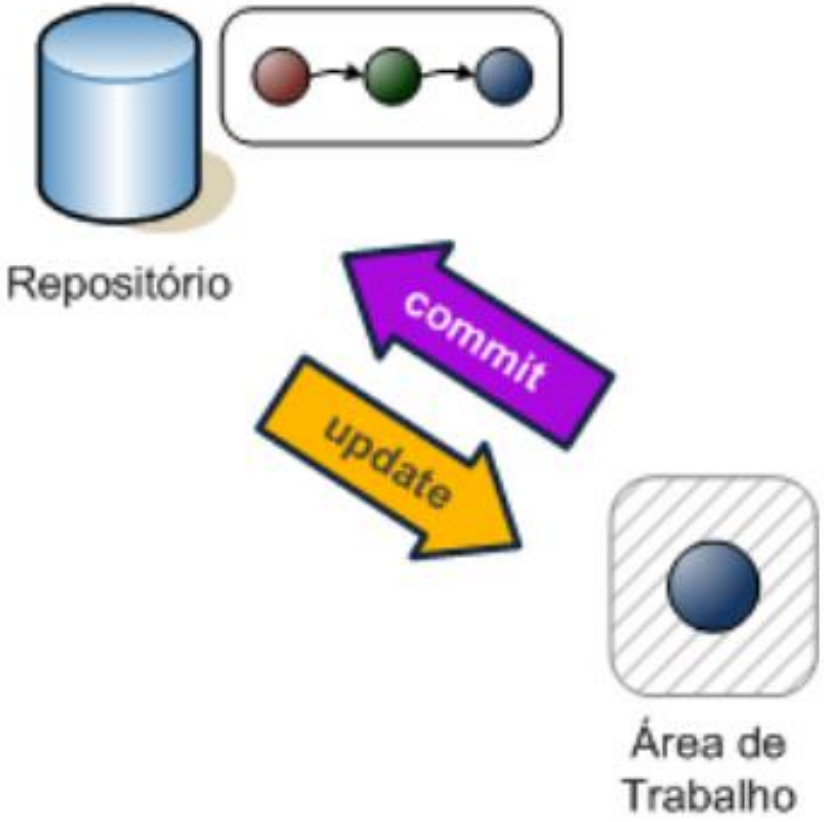
Conceitos

- **commit:** ato de criar o artefato no repositório pela primeira vez ou criar uma nova versão do artefato quando este passar por uma modificação;

- **merge:** que é a mesclagem entre versões diferentes, objetivando gerar uma única versão que agregue todas as alterações realizadas.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Comunicação



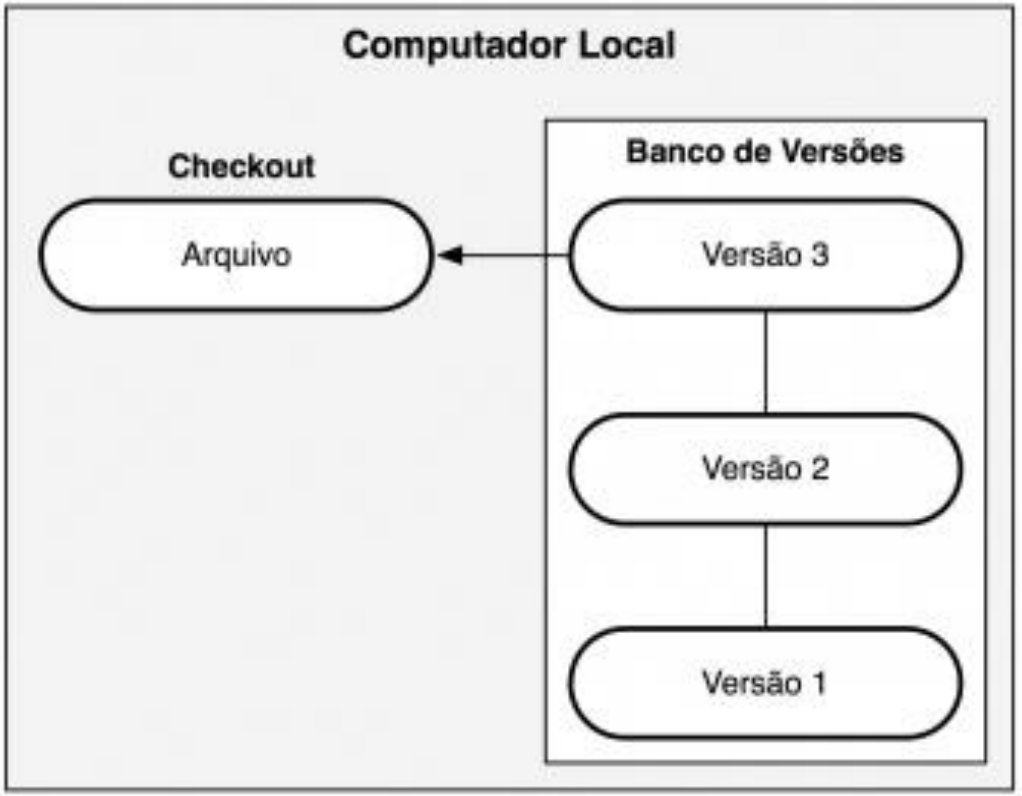
INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

Tipos SCV

- Local: RCS
- Centralizado: CVS e Subversion
- Distribuído: Git, Mercurial, Bitkeeper

SCV Local

- O método preferido de controle de versão por muitas pessoas é copiar arquivos em outro diretório (talvez um diretório com data e hora, se forem espertos).



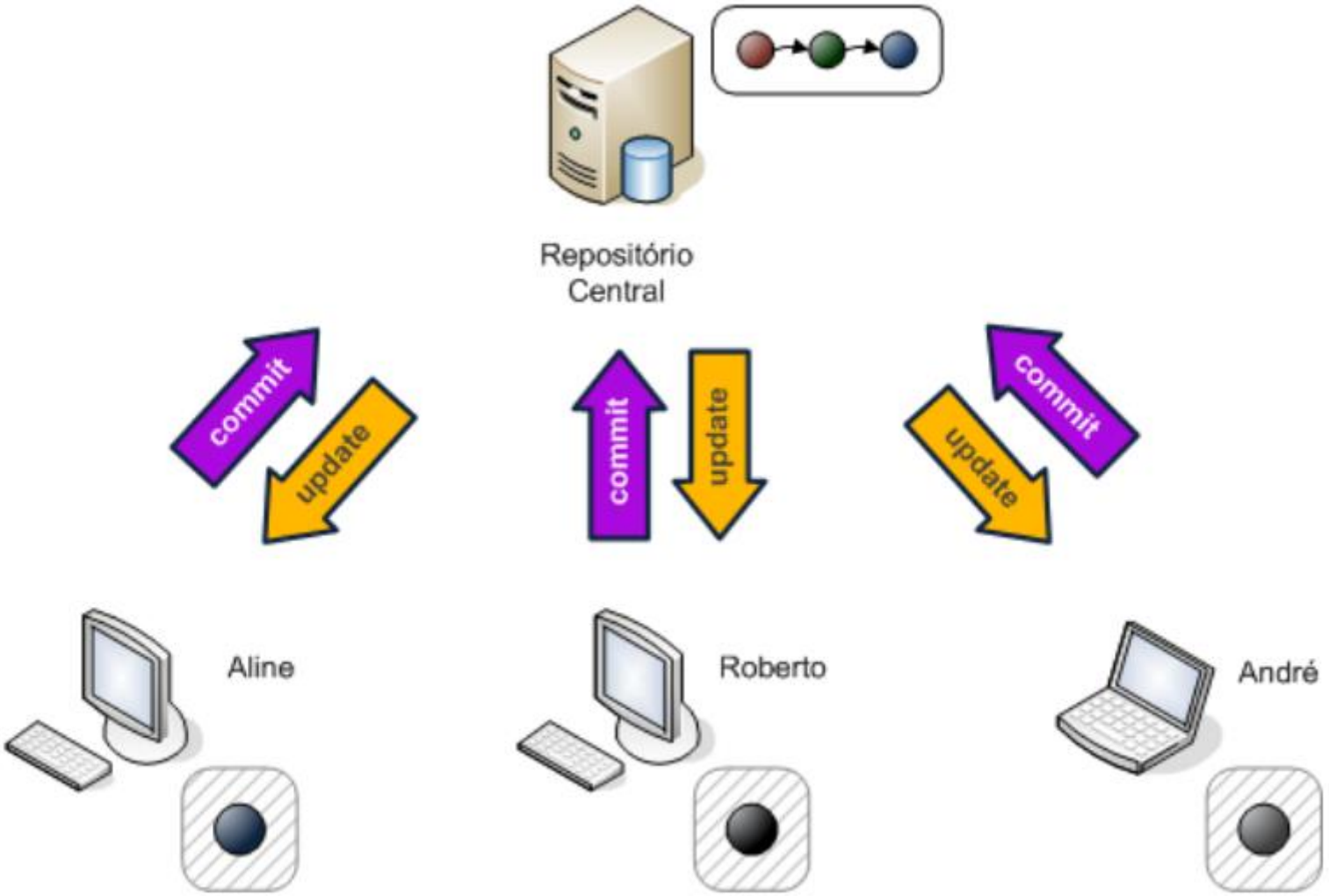
Características

- **Suscetível a erros**:
  - esquecer o diretório
  - gravar acidentalmente no arquivo errado
  - sobrescrever arquivos sem querer;

- Outro grande problema: trabalhar em conjunto;

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

SCV Centralizado



INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

SCV Centralizado

- SCVC (Centralized Version Control System ou CVCS)
- Exemplo: CVS, Subversion e Perforce
- Único repositório central que contém todos os arquivos versionados e vários clientes que podem resgatar (clone) os arquivos do servidor
- Por muitos anos, esse foi o modelo padrão para controle de versão.

Vantagens

- todo mundo pode ter conhecimento razoável sobre o que os outros estão fazendo no projeto
- Gerente têm controle específico sobre quem faz o quê
- facilita a administração (comparado SCV local)

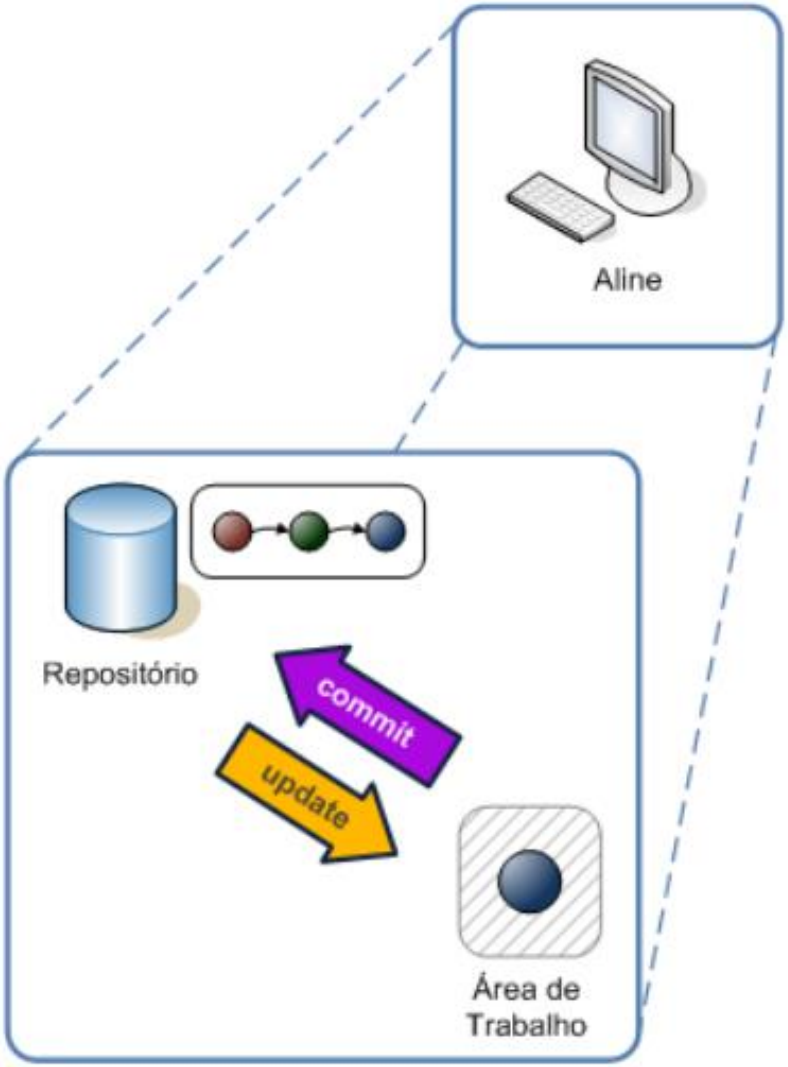
INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Desvantagens

- o servidor central é um ponto único de falha
- Se o servidor ficar fora do ar, ninguém pode trabalhar em conjunto ou salvar novas versões
- Se o disco do servidor do banco de dados for corrompido e não existir um backup adequado, perde-se tudo

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

SCV Distribuído



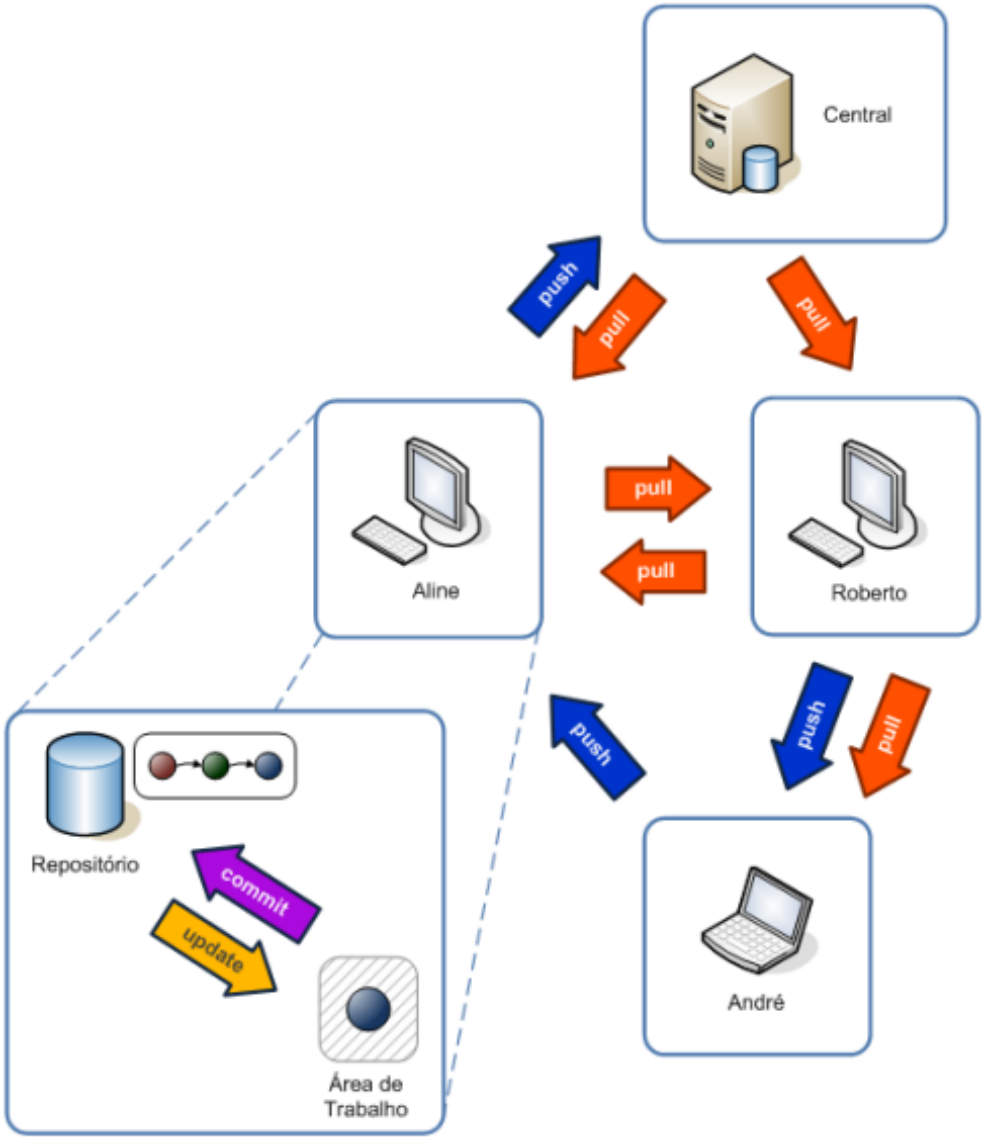
INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

SCV Distribuído

- SCVD (*Distributed Version Control System* ou DVCS).
- Exemplo: Git, Mercurial, Bazaar ou Darcs
- Os clientes não apenas fazem cópias das últimas versões dos arquivos: eles **são cópias completas** do repositório.
- Cada checkout (clone) é na prática um backup completo de todos os dados

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

SCV Distribuído



INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

git

- Um dos sistemas de controle de versão preferido

- Características:

– não depender de um servidor central;

– potencializar o trabalho paralelo;

– Praticamente todas as ferramentas de desenvolvimento dão suporte ao Git:

– Android Studio, Eclipse, NetBeans, Visual Studio, etc.

Linus Torvalds
Criador

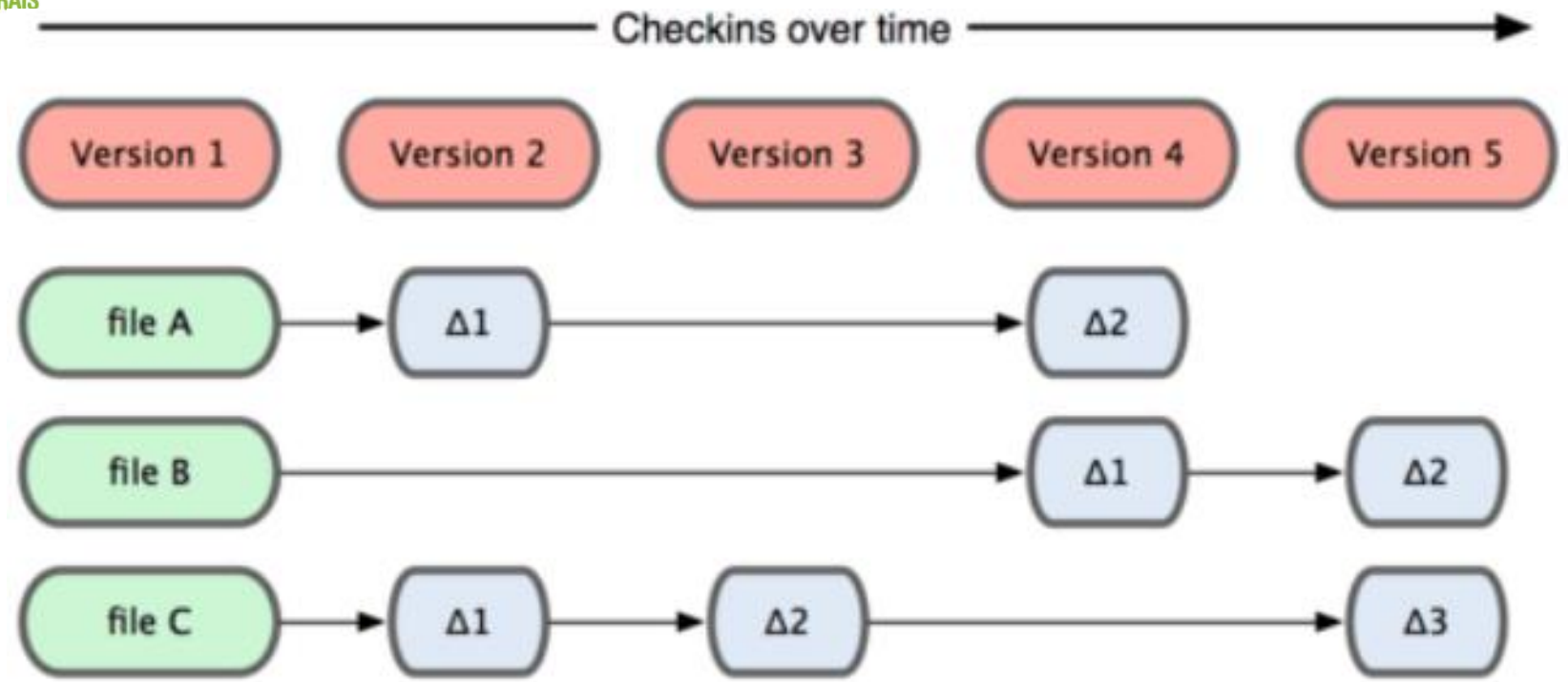
Junio Hamano
Mantenedor

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

- A maior diferença entre Git e qualquer outro SCV (Subversion e similares inclusos) está na forma que o Git trata os dados;
- Outros sistemas armazena informação como uma **lista de mudanças por arquivo;**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Git x outros (CVS, Subversion, etc.)

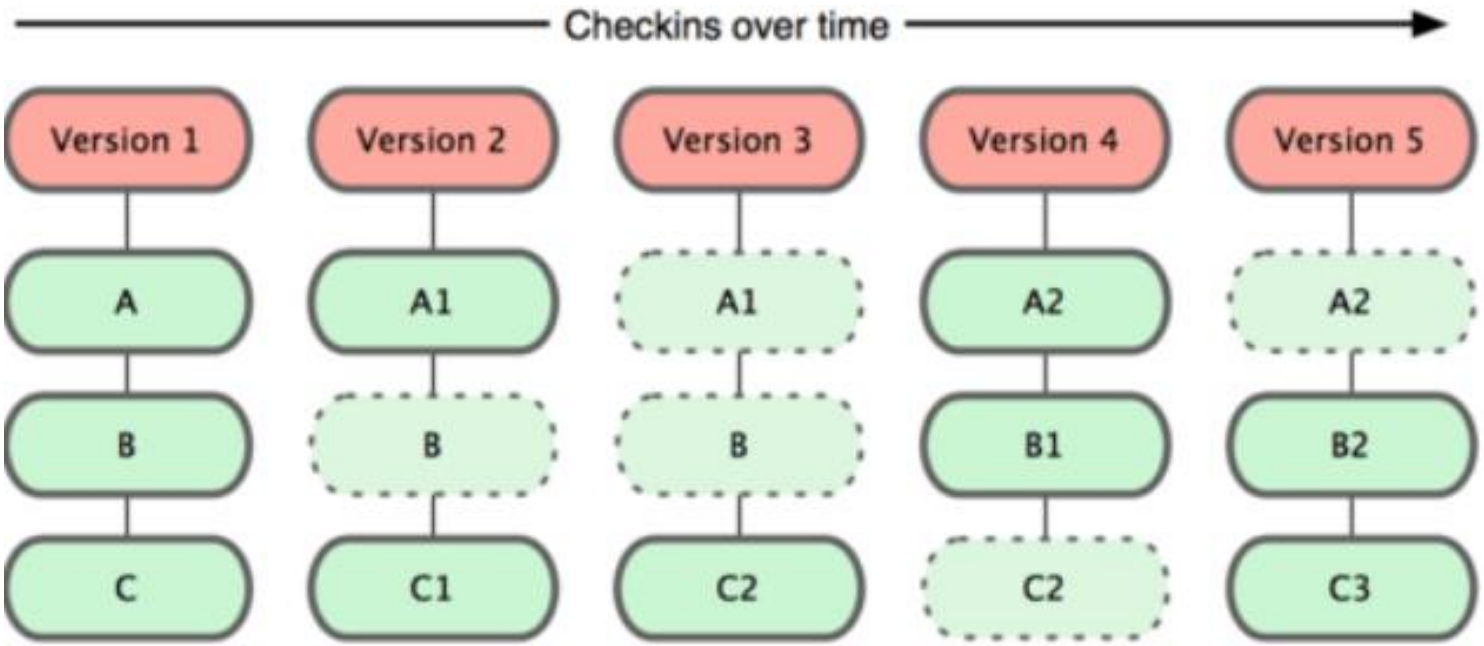


- Outros sistemas costumam armazenar dados como mudanças em uma versão inicial de cada arquivo

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

git

- Git armazena dados como snapshots



- se nenhum arquivo foi alterado, a informação não é armazenada novamente - apenas um link para o arquivo idêntico anterior que já foi armazenado.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Características

- Quase todas operações são locais: histórico, commits, etc.;
- Git tem integridade (checksum em tudo);
- Git geralmente só adiciona dados (não existe risco de perder informação);

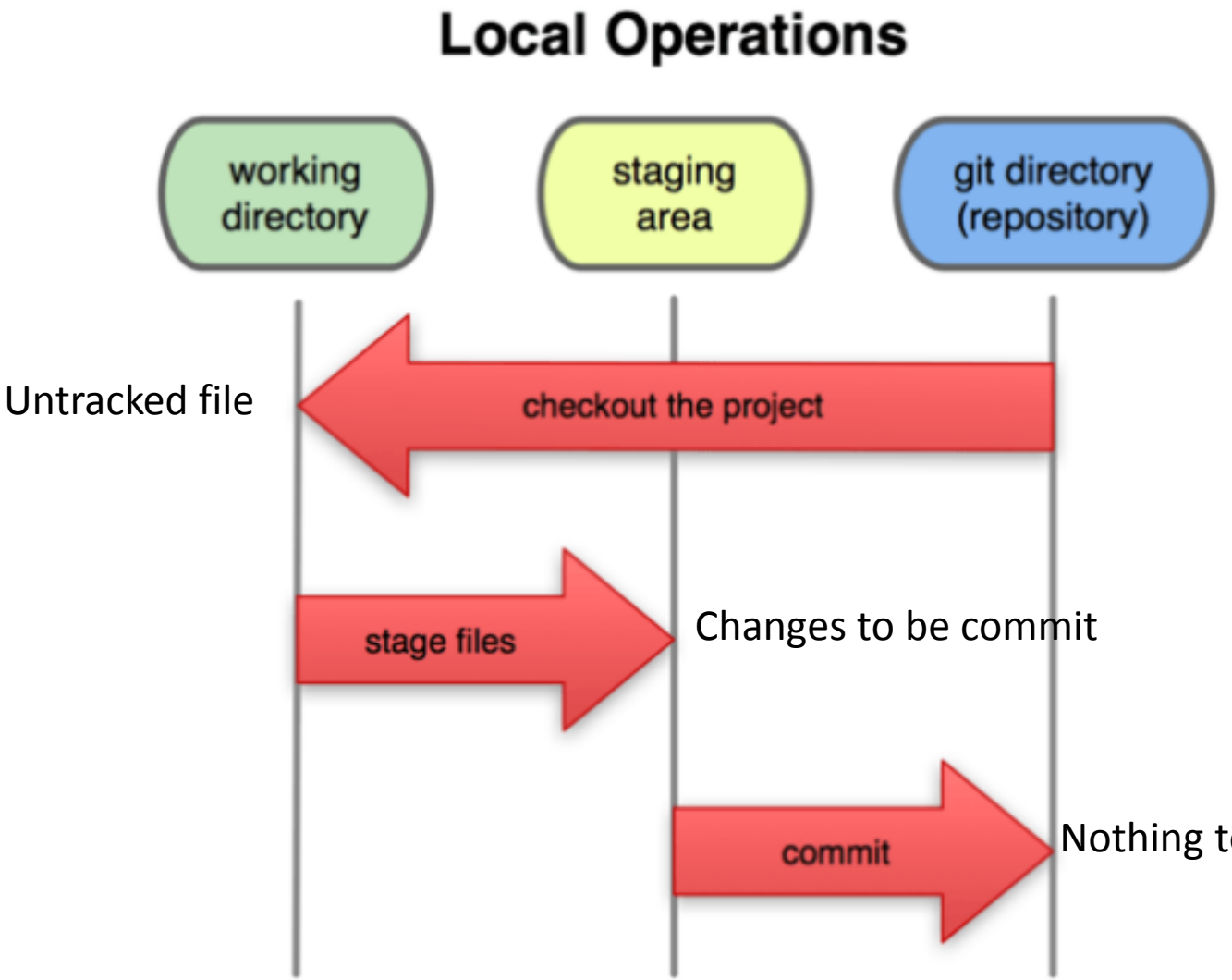
INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Os Três Estados

- **três estados** fundamentais:
- **Consolidado (committed)**: Dados são ditos consolidados quando estão seguramente armazenados em sua base de dados local
- **Modificado (untracked files)**: Modificado trata de um arquivo que sofreu mudanças mas que ainda não foi consolidado na base de dados.
- **Preparado (staged)**: Um arquivo é tido como preparado quando você marca um arquivo modificado em sua versão corrente para que ele faça parte do snapshot do próximo commit (consolidação).

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Workflow básico



INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

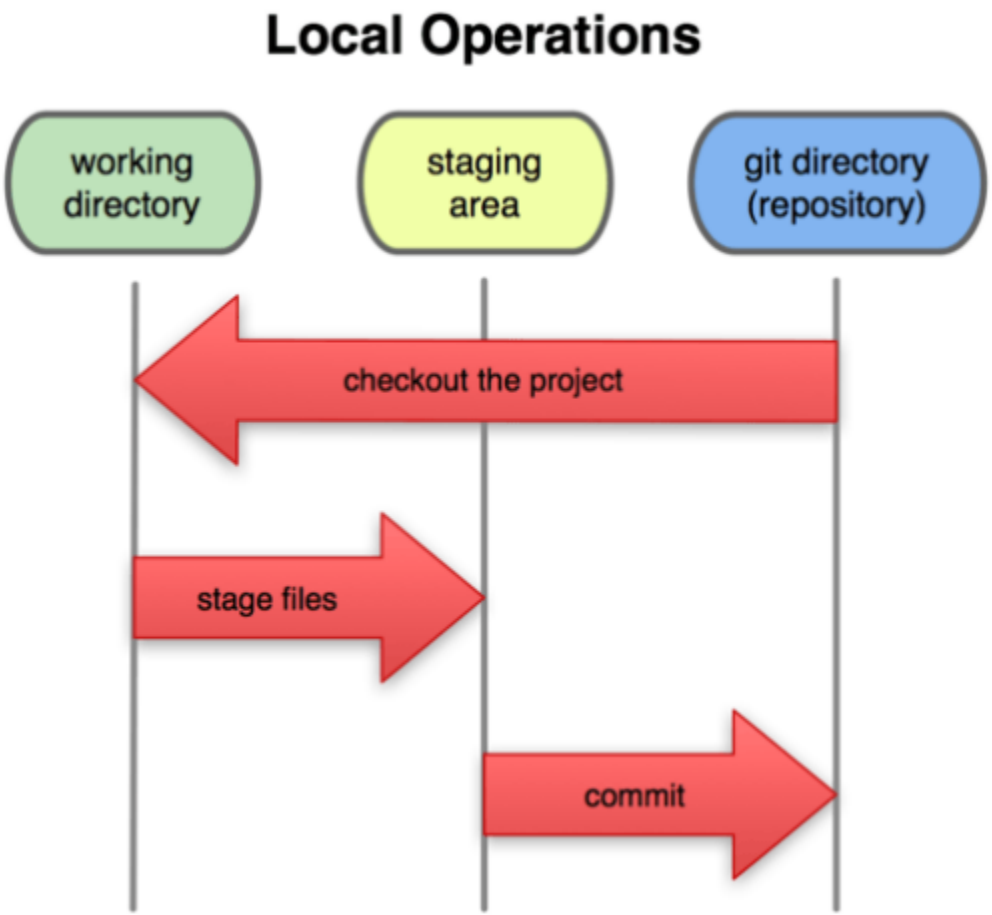
Workflow básico

- **O diretório do Git** é o local onde o Git armazena os metadados e o banco de objetos de seu projeto. Esta é a parte mais importante do Git, e é a parte copiada quando você clona um repositório de outro computador.

- **O diretório de trabalho** é um único checkout de uma versão do projeto.

- **A área de preparação** é um simples arquivo, geralmente contido no seu diretório Git, que armazena informações sobre o que irá em seu próximo commit. É bastante conhecido como índice (index), mas está se tornando padrão chamá-lo de área de preparação.

Workflow básico



1) Clonar o projeto
2) Modificar arquivos no seu diretório de trabalho.
3) Selecionar os arquivos, adicionando snapshots deles para sua área de preparação.
4) Fazer um commit, que leva os arquivos como eles estão na sua área de preparação e os armazena permanentemente no seu diretório Git.

INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

Instalando o GIT

- Capitulo 1 do livro:
- https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git
- Após instalação: Git Bash e Git GUI
- Para checar se a instalação deu certo, digite:
  - git no prompt/terminal, prefira usar o git bash

INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

git config

- Git vem com uma ferramenta chamada **git config** que permite a você ler e definir variáveis de configuração

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Criando uma identidade

- Definir o seu nome de usuário e endereço de e-mail.
- Importante porque todos os commits no Git utilizam essas informações

```
C:\>git config --global user.name "Paloma Oliveira"

C:\>git config --global user.email paloma.oliveira@ifmg.edu.br
```

- --global: Git sempre usará essa informação para qualquer coisa que você faça nesse sistema.

Verificando suas configs

- Comando **git config –list**
- lista todas as configurações do git

```
C:\>git config --list
core.symlinks=false
core.autocrlf=true
user.name=Paloma Oliveira
user.email=paloma.oliveira@ifmg.edu.br
```

- Você também pode verificar qual o valor que uma determinada chave tem para o Git digitando **git config {key}**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Obtendo Ajuda

- Existem três formas de se obter ajuda das páginas de manual (manpage) do Git:

```
1  $ git help <verb>
2  $ git <verb> --help
3  $ man git-<verb>
```

- Por exemplo, você pode obter a manpage para o comando config executando
- **git help config**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício

1) Crie um diretório com o nome RepoGit01

2) Crie um arquivo .txt dentro deste diretório com o nome README.txt

3) Digite a seguinte frase dentro do arquivo README.txt

meu primeiro repo git

- Pronto, esse diretório será nosso primeiro repositório GIT

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Iniciando um Repositório

- **git init** – dentro do diretorio que deseja versionar
- **git add** <nome do arquivo com extensão>

```
C:\Users\Admin\Documents\RepoGit01>git add README.txt
```

- **git add .** – adiciona todos arquivos de uma vez
- **git commit -m “Mensagem do Commit”**

```
C:\Users\Admin\Documents\RepoGit01>git commit -m "versao inicial"
[master (root-commit) 4d0eb1e] versao inicial
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
```

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Tipos de commit

- **git commit** – vai abrir um editor de texto para você digitar sua mensagem, para salvar e sair do editor use no final do arquivo **:wq**
- **git commit -m "Mensagem do Commit" –** esse comando não abre o editor e você coloca a mensagem no prompt
- **git commit –a –m "mensagem"** – esse comando adicionar todos os arquivos e faz o commit de tudo

Status de seus Arquivos

- **git status**.
- Se você executar este comando diretamente após uma clonagem, você deverá ver algo similar a isso:

```
C:\Users\Admin\Documents\RepoGit01>git status
On branch master
nothing to commit, working directory clean
```

- o comando lhe mostra em qual branch você se encontra.
- Por enquanto, esse sempre é o master, que é o padrão;
- **ls –la** para ver pastas ocultas no terminal

Exercício

1) Testando o git status

- Acrescente uma nova linha no arquivo README.txt:

  Hello GIT

- Execute o comando git status, o que acontece?
  - *“Changes not staged for commit”*
    - *significa que um arquivo monitorado foi modificado no diretório de trabalho, mas ainda não foi selecionado (staged).*

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício – **git add**

- O que é necessário fazer?
- Adicionar o arquivo. Como? git add
  - **git add README.txt**

- git add é um comando com várias funções:
  - monitorar novos arquivos;
  - selecionar arquivos;
  - e para fazer outras coisas como marcar como resolvido aquivos com conflito;
- Execute o comando git status novamente, e agora o que acontece? Mudou a mensagem? O que fazer?

Exercício

- Execute o commit e logo em seguida git status
- Sequência para iniciar um repositório local e adicionar arquivos:
  - git init
  - git add <nome arquivo> ou git add .
  - git commit –m “mensagem”
  - git status

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício

5) Execute o comando git add grit em seguida veja o status

- você pode ver que o seu arquivo grit agora está sendo monitorado e está selecionado
- Você pode dizer que ele está selecionado pois está sob o cabeçalho “Changes to be committed”.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício

6) Acrescente uma nova linha no arquivo README.txt: Verificando mudanças

7) Execute o comando git status

8)Adicione o arquivo README.txt novamente ao repositório: git add

9) Execute o comando git status novamente

10) Acrescente uma nova linha no arquivo README.txt: Mudando novamente

11) Execute o comando git status novamente, o que acontece?

Exercício - explicando

- Agora o arquivo README.txt aparece listado como selecionado e não selecionado. Como isso é possível?
- O Git seleciona um arquivo exatamente como ele era quando o comando git add foi executado.
- Se você fizer o commit agora, a versão do README.txt como estava na última vez que você rodou o comando git add é que será incluída no commit, não a versão do arquivo que estará no seu diretório de trabalho quando rodar o comando git commit.
- Se você modificar um arquivo depois que rodou o comando git add, terá de rodar o git add de novo para selecionar a última versão do arquivo

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Visualizando Mudanças

- Como saber o que exatamente você alterou?
- O comando git status mostra apenas quais arquivos foram alterados…
- Comando **git diff**
- Utilizado com frequência para responder estas duas perguntas:
- O que você alterou, mas ainda não selecionou (stage - *add*)?
- E o que você selecionou, que está para ser commitado *(commit)?*

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Comando git diff

- Apesar do comando git status responder essas duas perguntas de maneira geral…
- Mostra as linhas exatas que foram adicionadas e removidas — o patch, por assim dizer.
- Compara o que está no seu diretório de trabalho com o que está na sua área de seleção (staging).
- O resultado te mostra as mudanças que você fez que ainda não foram selecionadas.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Comando git diff

- Se você quer ver o que selecionou que irá no seu próximo commit, pode utilizar:

git diff --cached

ou

git diff --staged

- Este comando compara as mudanças selecionadas com o seu último commit

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Comando git diff

- É importante notar que o git diff por si só não mostra todas as mudanças desde o último commit
- apenas as mudanças que ainda não foram selecionadas.
- Isso pode ser confuso, pois se você selecionou todas as suas mudanças, git diff não te dará nenhum resultado.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício

1)Selecionar o arquivo README.txt e então edite-o, use o git status para ver as mudanças no arquivo que estão selecionadas, e as mudanças que não estão;

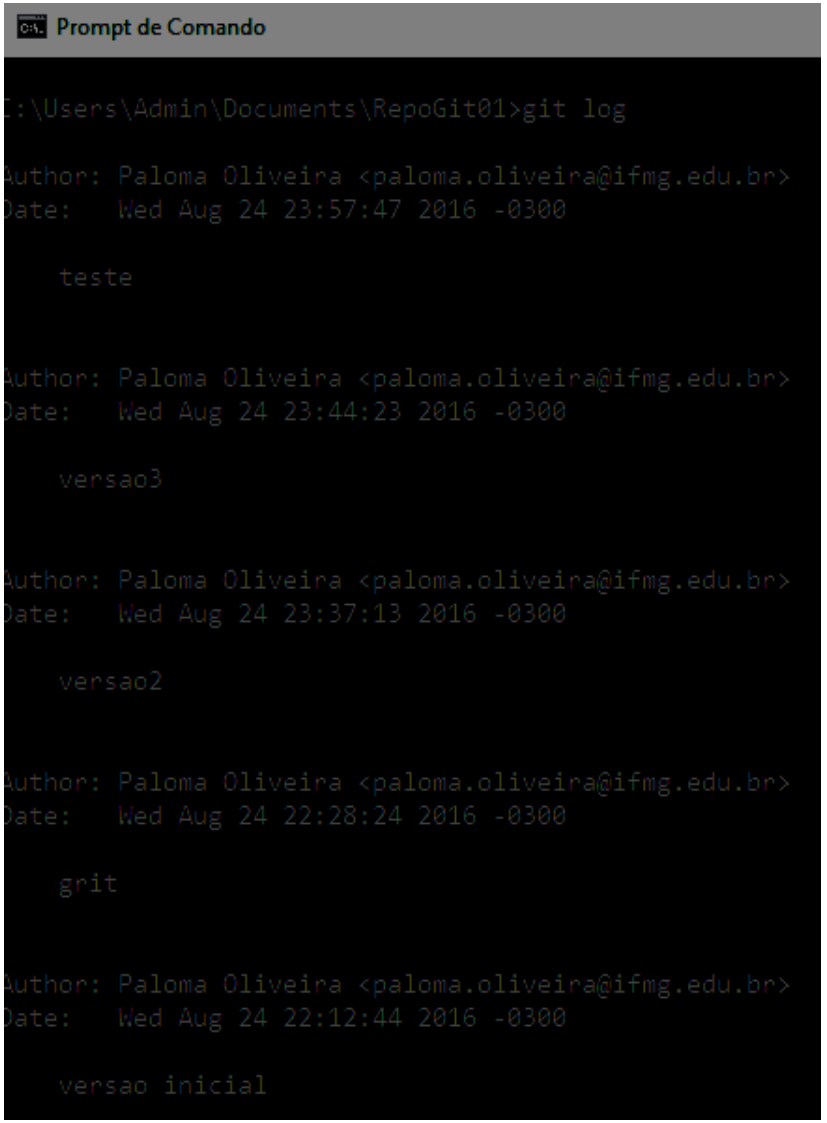
2)Agora você pode utilizar o git diff para ver o que ainda não foi selecionado;

3)Execute git diff --cached para ver o que você já alterou para o estado staged até o momento

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

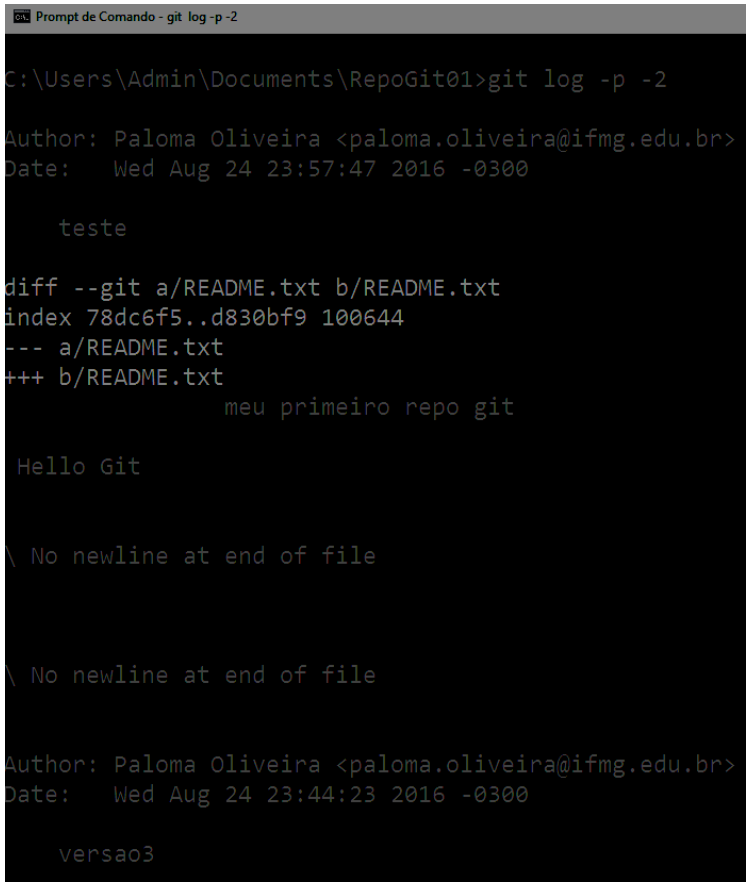
Histórico de Commit

- Após vários commits, você provavelmente vai querer ver o que aconteceu...
- Comando **git log** mostra o histórico de commits;
- commits mais recentes primeiro.
- comando lista cada commit com seu checksum SHA-1, o nome e e-mail do autor, a data e a mensagem do commit.



Histórico de Commit

- Um grande número e variedade de opções para o comando git log estão disponíveis;
- Opções mais usadas:
- -p: mostra o diff introduzido em cada commit.
- -2: limita a saída somente às duas últimas entradas.



INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Tagging

- Git tem a habilidade de criar tags em pontos específicos na história do código;
- Use esta funcionalidade para marcar pontos de release (v1.0, e por aí vai);
- Listar as tags disponíveis em Git: **git tag**
- Criando Tags: **git tag -a v1.4 -m “my version 1.4”**
- **-a:** especifica a versão desejada
- **-m:** define uma mensagem, que é armazenada com a tag.

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Tagging

- ver os dados da tag junto com o commit que foi taggeado: **git show**
- Taggeando mais tarde

- Você também pode taggear commits mais tarde.

–Para criar a tag no commit, você especifica a chave de verificação (ou parte dela) no final do comando:

- **git tag -a v1.0 9fceb02**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Compartilhando Tags

- Por padrão, o comando git push não transfere tags para os servidores remotos.
- Você deve enviar as tags explicitamente para um servidor compartilhado após tê-las criado.
- **git push origin [nome-tag]**
- Ex: git push origin v1.5
- Para muitas tags ao mesmo tempo: --tags
- **git push origin --tags**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Clonando um Repositório

- Caso você queira copiar um repositório Git já existente — por exemplo, um projeto que você queira contribuir
- o comando necessário é git clone <URL>.
- Exemplo: clonar a biblioteca Git do Ruby chamada Grit
- git clone git://github.com/schacon/grit.git

```
C:\Users\Admin\Documents\RepoGit01>git clone git://github.com/schacon/grit.git
Cloning into 'grit'...
remote: Counting objects: 4051, done.
R ceiving objects:  30% (1255/4051), 508.00 KiB | 182.00 KiB/s
```

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercício - Clone

1) Execute o clone da biblioteca Grit

2) Isso cria um diretório chamado grit, inicializa um diretório .git dentro deste, obtém todos os dados do repositório e verifica a cópia atual da última versão.

3) Verifique o que acontece com seu repositorio – diretório RepoGit01

4) Execute o comando git status, o que acontece?

Exercício – clone

- **"Untracked files"**

- Seu novo diretorio *grit* não está sendo monitorado
- Não monitorado significa basicamente que o Git está vendo um arquivo que não existia na última captura (commit);
- o Git não vai incluí-lo nas suas capturas de commit até que você o diga explicitamente que assim o faça (git add).

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Sumário até o momento

- Neste ponto, você pode executar todas as operações locais básicas do Git:
- efetuar mudanças;
- fazer o stage e commit de suas mudanças;
- ver o histórico de todas as mudanças do repositório.
- criar ou clonar um repositório;

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Ignorando arquivos

- Criar um arquivo com o nome **.gitignore** (no bloco de notas)
- Coloque o nome dos arquivos que você não deseja no controle de versão
- Cada arquivo em uma linha
- Adicione o arquivo ao controle de versão (add e commit)

INSTITUTO FEDERAL
MINAS GERAIS
Campus
Formiga

Exercício – tirando do staging area

- Crie dois novos arquivos:
- touch teste2.txt
- touch teste3.txt
- git add . (adicione todos arquivos untracked files)
- git status
- Para tirar um arquivo da staging área:

  git reset HEAD teste2.txt

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Como voltar um commit?

- Diversas formas para voltar uma versão
- **+ simples: usar o checksum**
- **git checkout <checksum>** – volta para uma versão especifica
- **git reset HEAD~<n>** - volta n commits

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Criando branches

- **git branch** – mostra os branches existentes e o branch corrente * - default master
- **git checkout –b <nome do branch>** – cria um novo branch a partir do branch corrente
- **git checkout <nome do branch>** - muda para um determinado branch
- **git merge <nome do branch>** - faz um merge com o branch corrente

Exercicio - Branch

- Crie um branch com o nome funcionalidade1
- **git checkout –b funcionalidade1**
- Visualize os branches existentes
- **git branch**
- Crie um arquivo .txt com o nome funcionalidade1.txt
- Insira uma nova linha no arquivo: “nova funcionalidade”
- Salve, adicione a área de preparação e commit a nova funcionalidade
- De um **git log**, o que aparece? Pq?
- Mude para o branch master
- **git checkout master**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Exercicio - Branch

- Mude para o branch master
- **git checkout master**
- Faça um git log, o que acontece?
  - Perceba que o commit do branch não aparece...
- Precisamos fazer um **merge -** juntar os dois arquivos
- Tenha certeza que esta no branch máster
  - **git merge funcionalidade1**

INSTITUTO FEDERAL
MINAS GERAIS
Campus Formiga

Referência

- **Livro: ProGit - Scott Chacon and Ben Straub**
- **https://git-scm.com/book/pt-br/v1/**