



# Paradigmas de linguagens de programação em python

Professores:

Sebastião Rogério feat. Kayo Monteiro

# Agenda

01

O que são VCSs

03

Por que aprender  
Git?

02

Um pouco de história  
do Git

04

Instalação,  
configuração e  
comandos básicos



01

0 que são *Version Control Systems?*

# Pense no seguinte cenário...

Um Iniciante da área de desenvolvimento de software ao se deparar com seu primeiro projeto de maior porte com uma equipe de desenvolvimento, geralmente, não conhece as ferramentas usadas para trabalhar paralelamente de forma eficiente e acaba sempre com aquela terrível dúvida:



# Pense no seguinte cenário...

- Como iremos desenvolver o projeto em paralelo?
- Existe a possibilidade de fazer isso sem que haja sobreposição das minhas alterações ou alguém esteja utilizando a versão errada?
- Como iremos resolver esse problema sem apelar para a nossa linda, muitas vezes útil, gambiarra?



# Qual a solução?

- Usar um sistema de controle de versão!  
Ótimo... Mas o que é um sistema de controle de versão? Como ele funciona?



# *Version Control Systems*

- Um sistema de controle de versão (como o próprio nome já diz) tem a finalidade de gerenciar diferentes versões de um documento.
- Com isso ele te oferece uma maneira muito mais inteligente e eficaz de organizar seu projeto, pois é possível acompanhar um histórico de desenvolvimento, desenvolver paralelamente e ainda te oferecer outras vantagens

# *Version Control Systems*

Vejamos alguns exemplos:

- Customizar uma versão;
- Incluir outros requisitos, finalidades específicas, layout e afins sem mexer no projeto principal;
- Resgatar o sistema em um ponto que estava estável, isso tudo sem mexer na versão principal.



# Como funciona um Controle de Versão?

- Basicamente, os arquivos do projeto ficam armazenados em um repositório (um servidor em outras palavras) e o histórico de suas versões é salvo nele.
- Os desenvolvedores podem acessar e resgatar a última versão disponível e fazer uma cópia local, na qual poderão trabalhar em cima dela e continuar o processo de desenvolvimento.

# Como funciona um Controle de Versão?

- A cada alteração feita, é possível enviar novamente ao servidor e atualizar a sua versão a partir outras feitas pelos demais desenvolvedores.



# Pense no seguinte cenário...

- E se por acaso os desenvolvedores estiverem editando o mesmo arquivo? O que irá acontecer se enviarem ao mesmo tempo para o servidor?
- Para evitar problemas como esse, o Sistema de Controle de Versão oferece ferramentas uteis para mesclar o código e evitar conflitos.



# Pense no seguinte cenário..

- Você atualizou seu projeto (usando uma função chamada de check-out ou update) e começou a fazer suas alterações.
- Ao mesmo tempo, outro desenvolvedor fez alterações e atualizou a versão no servidor.
- Quando for enviar sua versão (usando uma função chamada de check-in ou commit) o Sistema de Controle de Versão irá alertar que o seu arquivo está desatualizado.

## Pense no seguinte cenário..

- Ele enviará as novas informações adicionadas e permitirá mesclar as diferentes versões. Não apenas isso, ele também mostrará onde foram feitas atualizações, trechos de código incluídos ou removidos e casos de conflito, onde linhas podem se sobrescrever e oferecerá opções para mesclar manualmente, escolhendo a melhor solução.

# Como são divididos os sistemas de controle de versão?

Atualmente, os sistemas de controle de versão são classificados em dois tipos: **Locais**, **Centralizados** e **distribuídos**.

- Local Version Control Systems (LVCS)
- Centralized Version Control Systems (CVCS)
- Distributed Version Control Systems (DVCS)

# Como são divididos os sistemas de controle de versão?

- Como o próprio nome diz, é local. Não é possível colaborar com outros desenvolvedores.
- rcs foi um dos VCS locais mais populares

# Como são divididos os sistemas de controle de versão?

- O centralizado trabalha apenas com um servidor central e diversas áreas de trabalho, baseados na arquitetura cliente-servidor.
- Por ser centralizado, as áreas de trabalho precisam primeiro passar pelo servidor para poderem comunicar-se.



# Como são divididos os sistemas de controle de versão?

- Essa versão atende muito bem a maioria das equipes de desenvolvimento que não sejam enormes e trabalhem em uma rede local, além de não ter problemas de velocidade para enviar e receber os dados e ter um bom tempo de resposta do servidor.
- Um dos principais sistemas com o tipo de controle de versão centralizado é o CVS, Subversion, Perforce.

# Resumo do VCS – Centralizado

- Um servidor mantém todos os arquivos.
- Ponto único de falha (e se o servidor cair?)

# Como são divididos os sistemas de controle de versão?

- O distribuído vai mais além. Ele é recomendado para equipes com muitos desenvolvedores e que se encontram em diferentes filiais.
- Esta versão funciona da seguinte maneira: cada área de trabalho tem seu próprio “servidor”, ou seja, as operações de check-in e check-out são feitas na própria máquina.

# Como são divididos os sistemas de controle de versão?

- Diferentemente do centralizado, as áreas de trabalho podem comunicar-se entre si, recomenda-se usar um servidor como centro do envio dos arquivos para centralizar o fluxo e evitar ramificações do projeto e a perda do controle sobre o mesmo, geralmente o sistema te dá essa opção, oferecendo um servidor remoto para hospedar o projeto.

# Como são divididos os sistemas de controle de versão?

- A comunicação entre o servidor principal e as áreas de trabalho funciona com outras duas operações, para atualizar e mesclar o projeto, chamadas de pull e push (puxar e empurrar).
  - **pull:** Com esta operação é possível pegar a versão de outra área de trabalho e mesclar com a sua.
  - **push:** Com esta operação temos o processo inverso do pull, ou seja, enviando para outra área a sua versão do projeto.

# Como são divididos os sistemas de controle de versão?

Por ser na própria máquina, o sistema de controle distribuído acaba sendo mais rápido, porém exige maior conhecimento da ferramenta e de início podem atrapalhar o desenvolvedor.

Como assim?

# Como são divididos os sistemas de controle de versão?

- Como exemplo, o sistema de mesclagem em edições concorrentes, se torna diferente por trabalhar em um sistema de arquivos binários (sequenciais de bits compostos por zero e um) que em determinadas situações não permite a comparação entre atualizações concorrentes.
- O sistema centralizado trabalha com arquivos de texto, que permite a comparação em atualizações concorrentes e da opção ao desenvolvedor para escolher a melhor solução.

# Como são divididos os sistemas de controle de versão?

- Portanto, por esse tratamento de mesclagem ser diferente, podem ocorrer situações onde o trabalho de alguém possa ser sobreposto e gerando tormento para os desenvolvedores.
- Para isso existe uma função chamada lock, que bloqueia o arquivo para que não seja modificado por outros enquanto estiver com você.
- Os sistemas distribuídos mais conhecidos são o Git e o Mercurial.



# Resumo do VCS – Distribuído

- Cada cliente tem uma cópia de todo repositório.



02

# Um pouco de história do Git

# Breve histórico do Git

- Criado pelo Linus Torvalds, o mesmo criador (e principal mantenedor) do kernel Linux.
- Em 2002 o projeto Linux começou a usar o BitKeeper, um DVCS proprietário.
- Em 2005, a empresa que desenvolve o BitKeeper parou de apoiar projetos open source.

# Breve histórico do Git

- Isso levou a comunidade do kernel Linux a desenvolver sua própria ferramenta a partir das lições aprendidas com o BitKeeper.
- Objetivos:
  - Ser rápido
  - Design simples
  - Bom suporte para desenvolvimento não-linear
  - Completamente distribuído
  - Habilidade de gerenciar projetos grandes, como o Linux, de forma eficiente.



03

Por que aprender Git?

# Geral usa haha

- De acordo com uma pesquisa feita pela Fundação Eclipse em Maio de 2014, 42,9% dos desenvolvedores profissionais usam Git ou GitHub como VCS principal, frente aos 36% em 2012, 27,6% em 2011 e 12,8% em 2010.
- Na Inglaterra, o site UK IT aproximadamente 20,32% das oportunidades de emprego para desenvolvedores exigem conhecimento em Git.

# 04

Instalação, configuração e  
comandos básicos

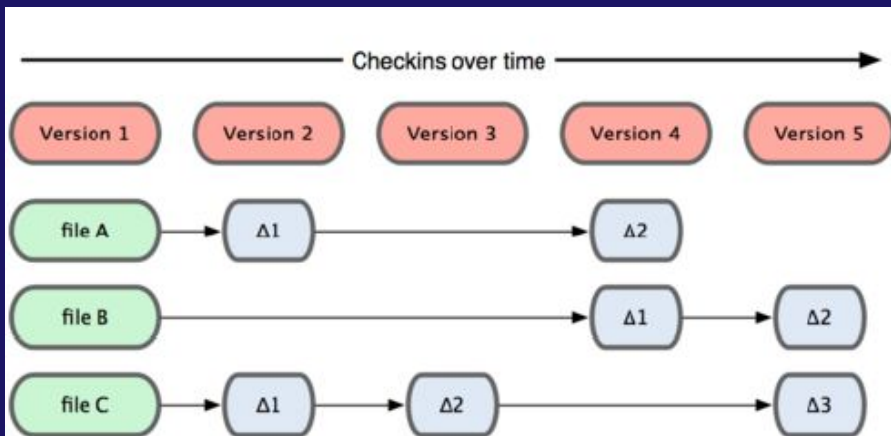
# Como o Git funciona?

- Ao invés de armazenar as diferenças (diffs), Git armazena uma "cópia" dos dados atuais (snapshots).
  - Graças a isso, o Git parece mais um sistema de arquivos que um VCS.
- Quase toda a operação é local.
  - Isso aumenta o desempenho e permite trabalhar offline.
- Checagem de integridade (SHA-1).
  - Praticamente impossível fazer uma alteração sem que o Git fique sabendo.

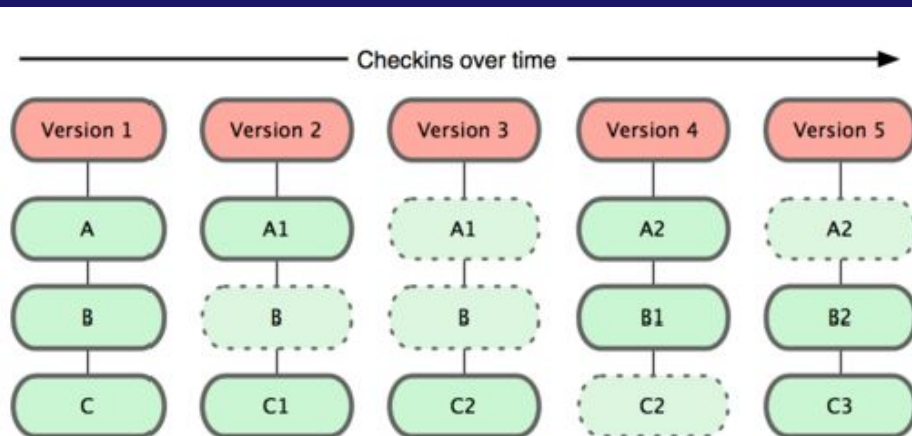


# Como o Git funciona?

VCS tradicional

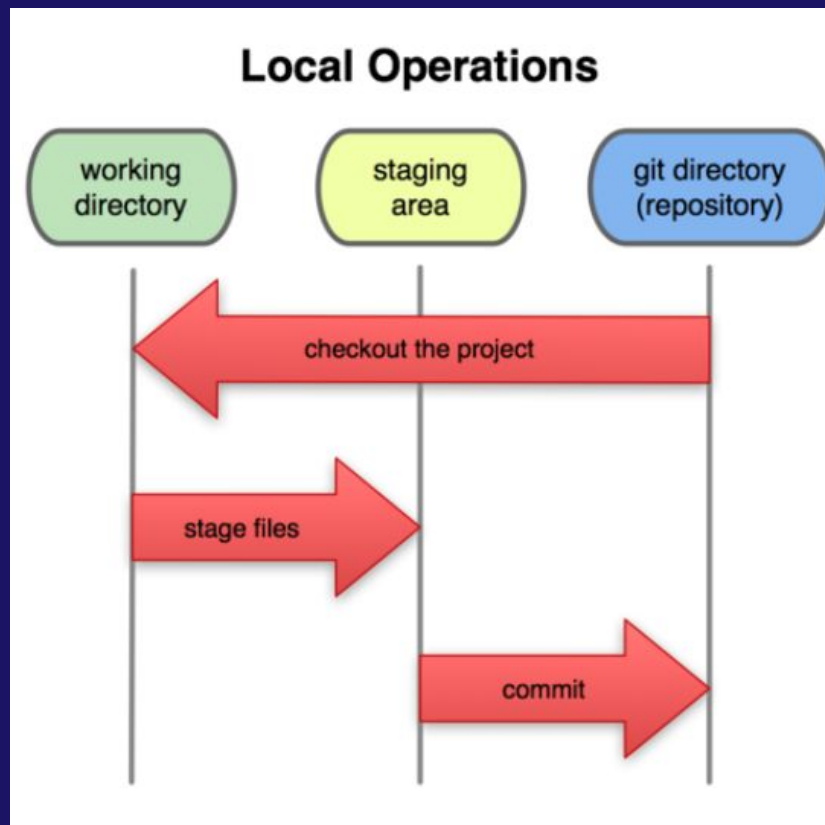


Git



# Os três estados

- Um arquivo pode estar em três estados:
  - *unmodified / committed*
  - *modified*
  - *staged*




# Instalando

1. Acesse o site oficial e faça o download do instalador do GIT para Windows.
2. Depois de baixado, clique duas vezes no arquivo para iniciar o assistente de instalação. Basta seguir as instruções na tela, clicando em Next. Ao término, clique em Finish para concluir com êxito a instalação.



# Configurando

Abra o prompt de comando e digite os seguintes comandos no terminal:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains two lines of text in a yellow monospace font.

```
git config --global user.name "João Silva"  
git config --global user.email "exemplo@seuemail.com.br"
```

# Como usar o Git

Um repositório é o maior bem de qualquer projeto controlado por versão. Para transformar qualquer diretório em um repositório GIT, o simples comando **git init** **<directory>** pode ser utilizado. Uma pasta chamada .git também deve começar a existir no diretório em que o comando foi executado.

# Como usar o Git

Por outro lado, se você já tem um diretório e deseja verificar (clone-lo), você pode usar o comando `git clone`. Se você estiver tentando verificar um repositório local, use o seguinte comando:



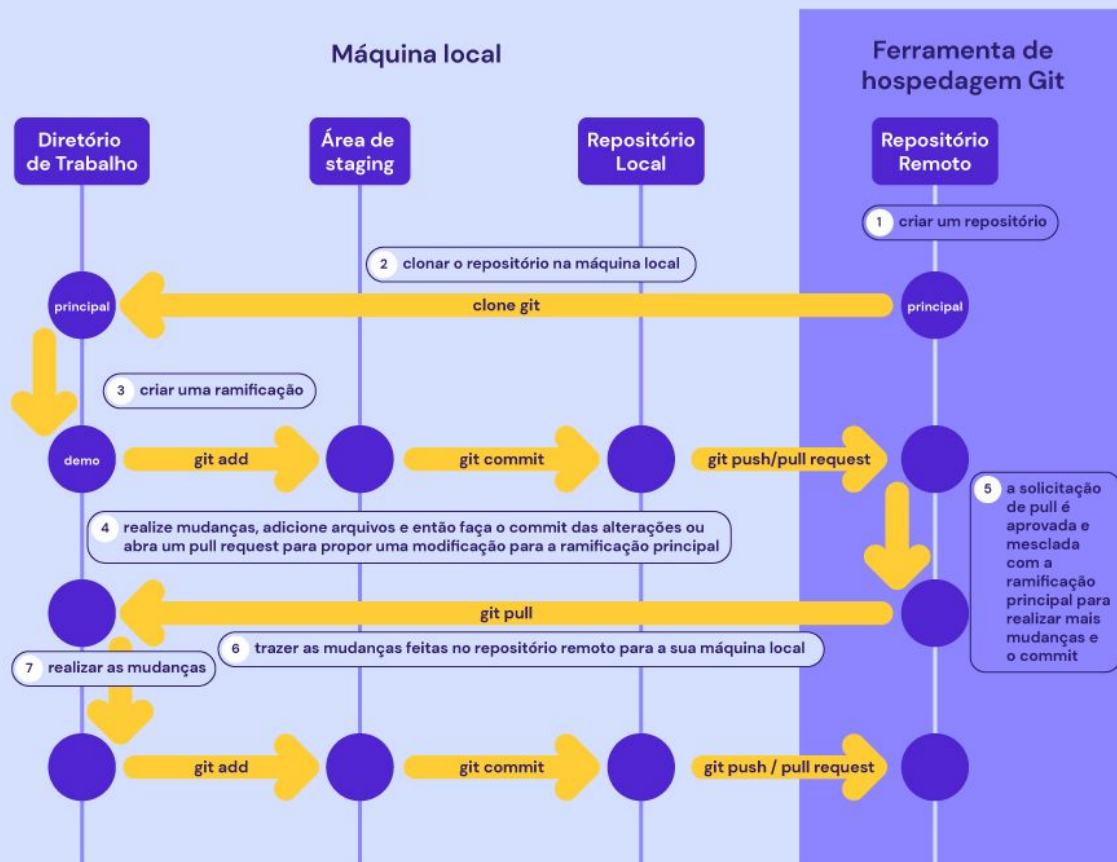
```
git clone /path/to/local/repository
```

# Como usar o Git

Se você pretende verificar um repositório armazenado remotamente, use:



```
git clone user.name@host:/path/to/remote/repository
```



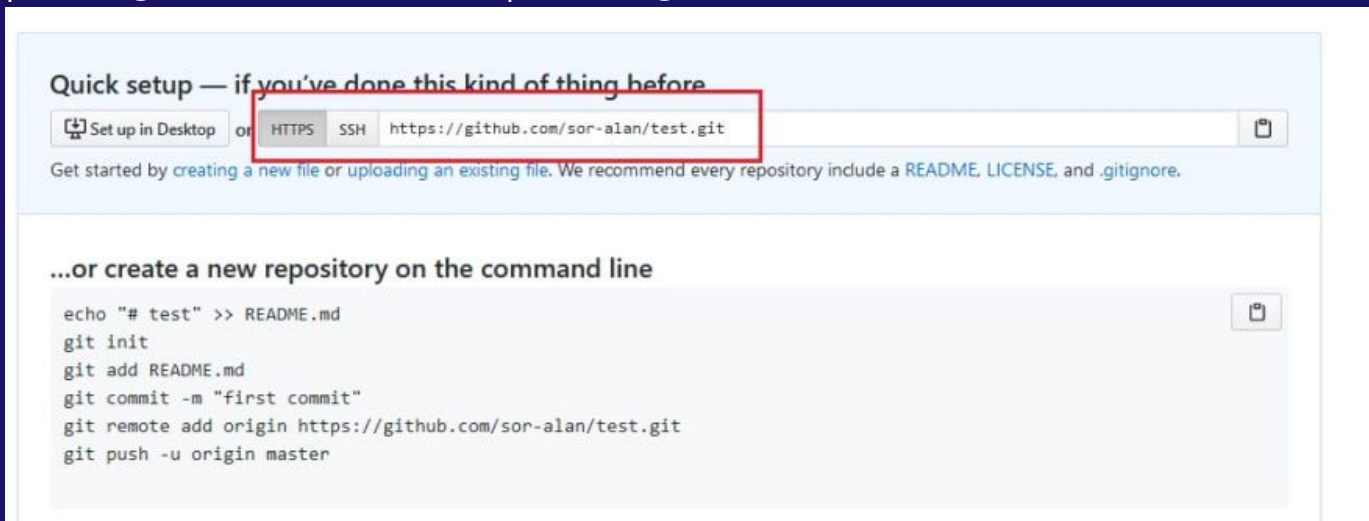


# Comandos básicos

1. Tenha certeza que o seu git está instalado. No terminal ou prompt de comando cheque através do comando `git -v`
2. No GitHub, crie um novo repositório. Na tela onde pede para fazer upload ou criar arquivos, guarde o link HTTPS que será gerado

# Comandos básicos

2. No GitHub, crie um novo repositório. Na tela onde pede para fazer upload ou criar arquivos, guarde o link HTTPS que será gerado



# Comandos básicos

3. Abra o Git Bash ou terminal na pasta onde está o seu projeto
4. Inicie a pasta como um repositório do Git através do comando:  
**git init**
5. Em seguida, adicione os arquivos de configuração para preparar o commit:  
**git add .**
6. Opcional: Adicione um arquivo readme caso não tenha iniciado o repositório com ele:  
**git add README.md**

# Comandos básicos

7. Crie um novo commit para os arquivos que irá subir para o repositório:  
**git commit -m "first commit"**
8. Suba seus arquivos utilizando a URL gerada no passo 2 no seguinte comando:  
**git remote add origin URL-GERADA-PELO-PASSO-2-AQUI**
9. Autorize o upload com seu login e senha:  
**git push -u origin master**

# Personalizando GitHub



# Referências

- Material cedidos gentilmente por: Thiago Kenji Okada e Diogo de Jesus Pina:  
[https://paca.ime.usp.br/pluginfile.php/68432/mod\\_resource/content/1/slides\\_git.pdf](https://paca.ime.usp.br/pluginfile.php/68432/mod_resource/content/1/slides_git.pdf)
- <https://www.devmedia.com.br/sistemas-de-controle-de-versao/24574>
- <https://dev.to/alanfabricio/subindo-seu-repositorio-no-github-atraves-da-linha-de-comando-3kcm>