Curso DevOps

Aula 02 - Gerenciamento de Configurações em DevOps

Prof. Esp. Guilherme Jorge Aragão da Cruz


guilherme.cruz@alumni.usp.br


linkedin.com/in/guijac

Roteiro

- Gerenciamento de Configurações:
  - Configuração?
  - Contexto Histórico;
  - Importância;
  - Aplicação em DevOps.
- Ferramentas:
  - Confluence;
  - Jira;
  - Git;
  - Gitlab;
- Gitlab x Bitbucket x Github;
- Arquitetura Git;
- O Arquivo README.md;
- Principais Comandos Git.
- Referências Bibliográficas.

Configuração?

- Ciclo de Desenvolvimento Típico:
  - Ideação, Análise, Desenho, Implementação, Testes e Implantação.

Configuração?

- Ciclo de Desenvolvimento Típico:
  - Ideação, Análise, Desenho, Implementação, Testes e Implantação.

- Configuração em um Ciclo de Desenvolvimento? 🤔

Configuração?

- Ciclo de Desenvolvimento Típico:
  - Ideação, Análise, Desenho, Implementação, Testes e Implantação.

- Configuração em um Ciclo de Desenvolvimento? 🤔
  - Requisitos;
  - Código-fonte;
  - Protótipos;
  - Arquivos de Testes;
  - Arquivos de Infraestrutura.

Configuração?

- Ciclo de Desenvolvimento Típico:
  - Ideação, Análise, Desenho, Implementação, Testes e Implantação.

- Configuração em um Ciclo de Desenvolvimento? 🤔
  - Requisitos;
  - Código-fonte;
  - Protótipos;
  - Arquivos de Testes;
  - Arquivos de Infraestrutura.

Fonte: Adaptado de O que é Gerência de Configuração de Software? | Blog (pronus.io)

Contexto Histórico

- Disciplina de gerenciamento técnico para monitoramento de mudanças no desenvolvimento de sistemas complexos;
- Criada pelo Departamento de Defesa dos EUA durante a década de 1950;
- Após a publicação de seu guia definitivo, em 2001, passou a ser utilizada em outras áreas, como desenvolvimento de software e engenharia.

Fonte: Adaptado de O que é Gerência de Configuração de Software? | Blog (pronus.io)

Importância

- Permite que equipes de engenharia criem sistemas robustos e estáveis através de **ferramentas de gerenciamento de dados** de configuração, como metadados para inicialização de diferentes aplicações e seu próprio código-fonte.


```python
import os
from flask import Flask, Markup, render_template, request
from sqlalchemy.sql import text
from sqlalchemy import create_engine

db_connect = create_engine('sqlite:///mydb.db', echo=True)

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = db_connect

@app.route("/")
def hello():
    return "<h1>Hello World!</h1>"
```

Código-fonte de uma aplicação Python com Framework Flask.

Importância

- Valores de configuração ou códigos-fonte podem ser adicionados, removidos ou modificados;
- Sem um devido controle de versão e configuração adequados torna-se um **caos gerenciar múltiplos arquivos**.

```yaml
version: "3.9"
services:
  db:
    build: db/.
    volumes:
      - /var/lib/mysql
      - ./database.sql:/docker-entrypoint-initdb.d/dump.sql
    environment:
       MYSQL_ROOT_PASSWORD: my-password

  web:
    build: app/.
    ports:
      - "5000:5000"
    links:
      - db
    depends_on:
      - db
```

Arquivo docker-composse.yml

Aplicação em DevOps

- No **passado recente**, alterações de recursos de administração de aplicações (infraestrutura) eram, em sua maior parte, manuais;
- O Gerenciamento de Configuração passa a ser uma **parte fundamental** no ciclo de vida DevOps, trazendo recursos como a **infraestrutura como código**.


```
resource "aws_s3_bucket" "this" {
  bucket = var.s3_name
}

resource "aws_s3_bucket_ownership_controls" "this" {
  bucket = aws_s3_bucket.this.id
  rule {
    object_ownership = "ObjectWriter"
  }
}
```

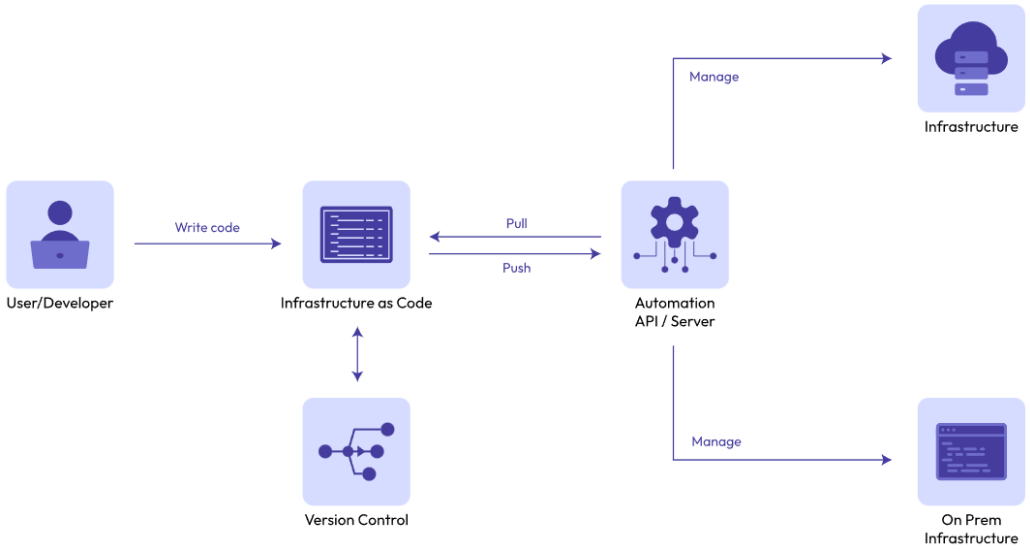
```
variable "region" {
  type = string
  default = "us-east-1"
}

variable "s3_name" {
  type = string
}
```

Arquivos de configuração de infraestrutura cloud (Terraform).

Aplicação em DevOps

- A configuração em DevOps também traz a administração de sistemas para **dentro** da **engenharia de software**;
- Organizações utilizam-se desta possibilidade para **capacitar engenheiros e engenheiras** de software na solicitação e provisionamento de novos recursos.



Fonte: SSL2BUY (2022)

Ferramentas: Confluence

- **Wiki** para gerenciamento de conhecimento, ideal para equipes criarem, coletarem e colaborarem em qualquer projeto ou ideia.

Confluence

Gestão de Conhecimento

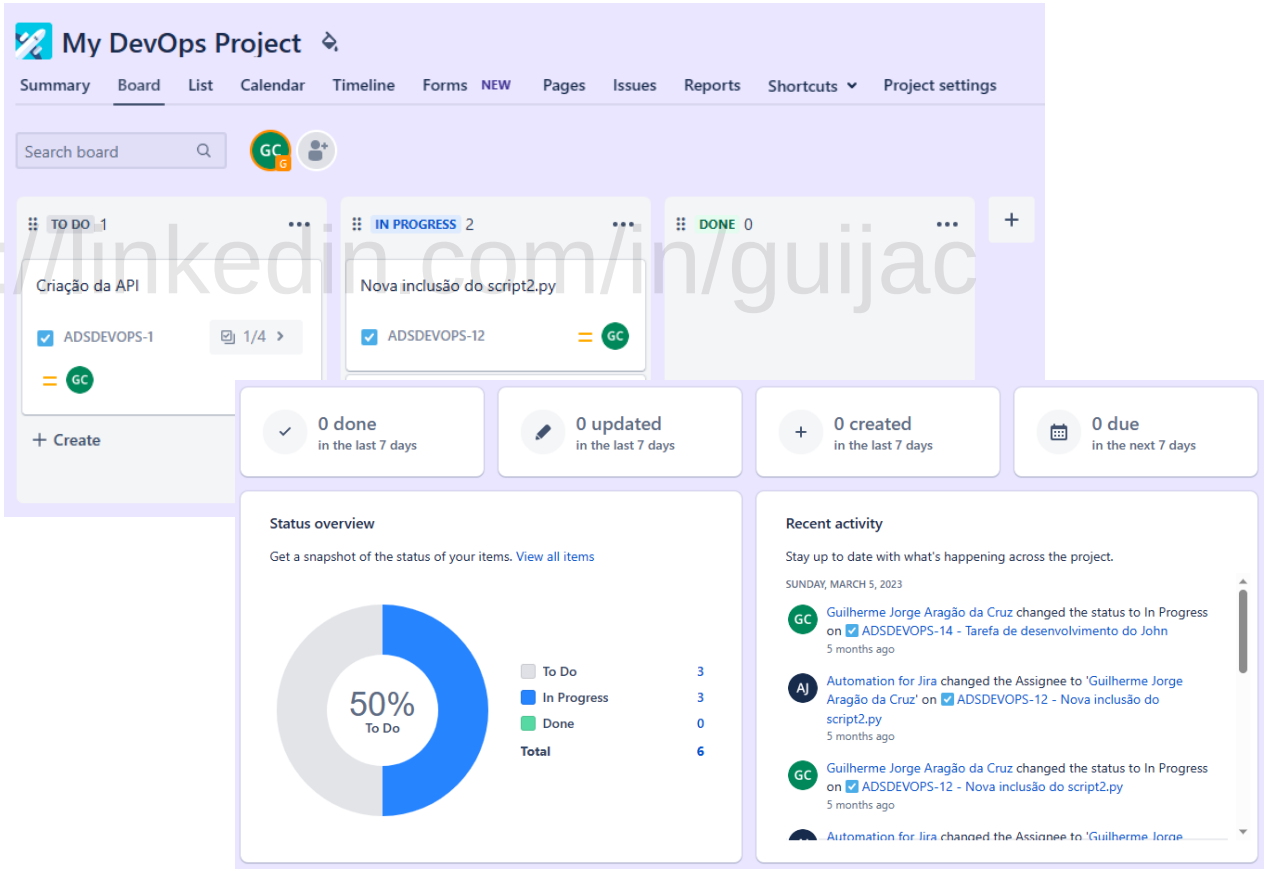
Fonte: Painel - Confluence MP-MT (mpmt.mp.br)

Ferramentas: Jira

- Ferramenta para gerenciamento de projetos com diversos templates, como **Scrum** ou **Kanban**.

Jira

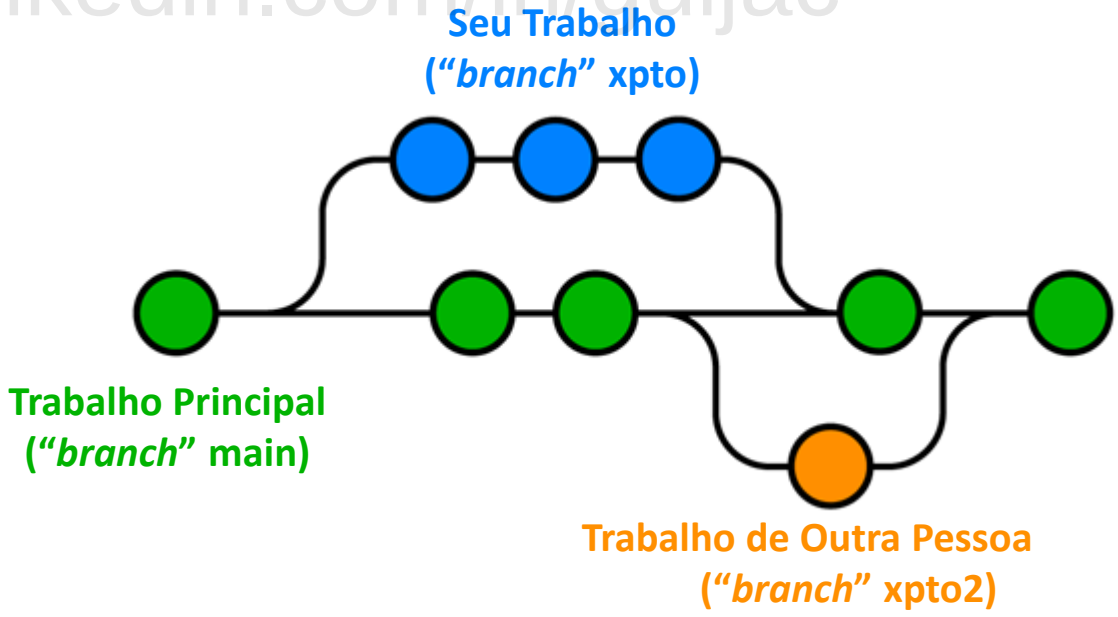
Gestão de Projetos e
Gestão de Mudanças



Fonte: Elaboração própria (2025)

Ferramentas: Git

- Sistema de Controle de Versões Distribuído, criado por Linus Torvalds;
- Usado principalmente no desenvolvimento de software, mas pode ser utilizado para registrar o histórico de edições de qualquer tipo de arquivo.



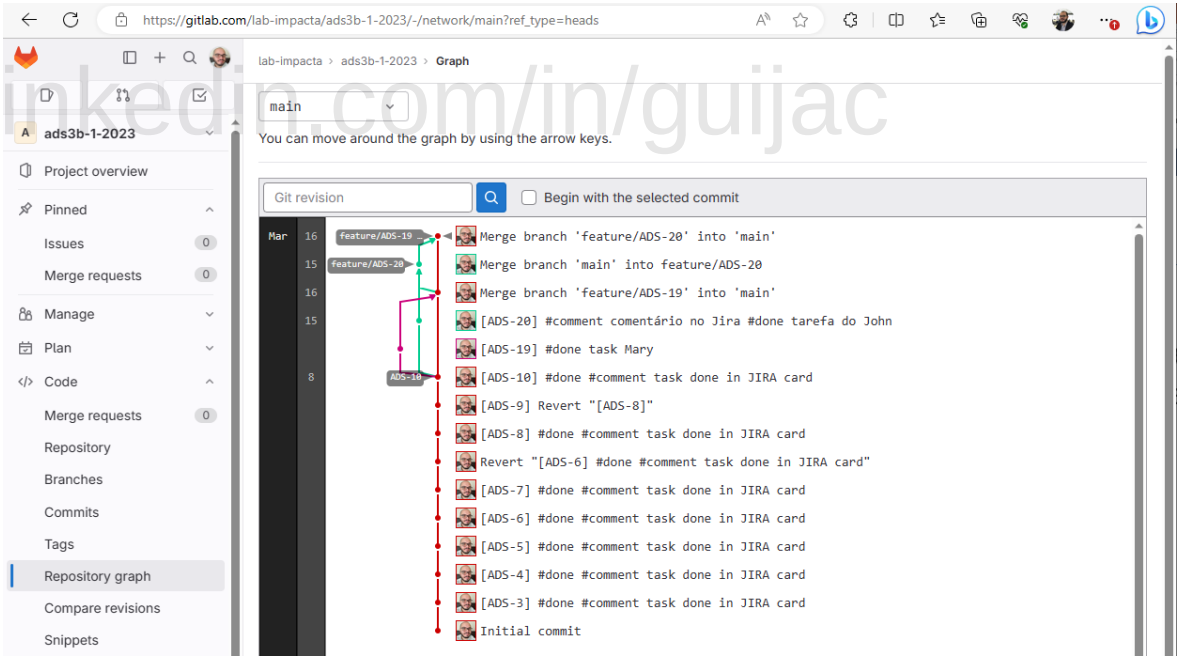
Fonte: Elaboração própria (2025)

Ferramentas: Gitlab

- **Plataforma** para hospedagem de código-fonte, baseada em Git, com possibilidade de instalação e configuração em sua própria infraestrutura;

GitLab

Versionamento de Fontes, Integração e Entrega Contínua



Fonte: Elaboração própria (2025)

Ferramentas: Gitlab

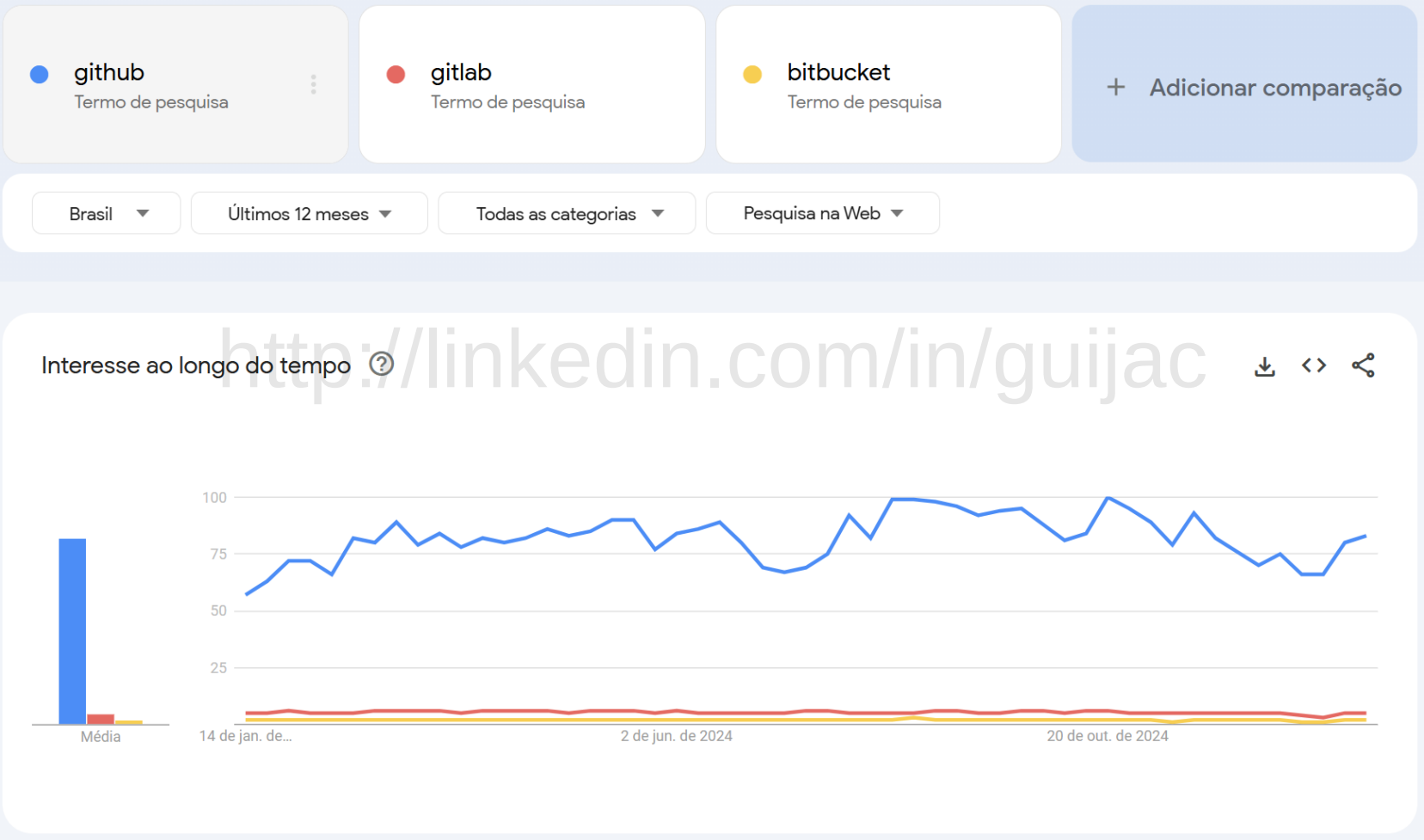
Trabalharemos com grupos e projetos.

Fonte: [How to use GitLab for Agile portfolio planning and project management | GitLab]

Gitlab x Bitbucket x Github

Fonte: Elaboração própria (2025)

Gitlab x Bitbucket x Github



Fonte: [github, gitlab, bitbucket - Pesquisar - Google Trends]

Gitlab x Bitbucket x Github

[Bitbucket vs Github vs Gitlab Detailed Comparison 2022 (jelvix.com)]

WHAT'S GITHUB?

GitHub is the largest Git-based version-control platform. Now it hosts more than 38 million projects and is most commonly used by open-source communities.

Pros	Cons
✔ Big community	✖ Comparatively expensive
✔ Open-source popularity	✖ Not enough CI/CD functionality
✔ Easy interface	✖ Functionality is limited compared to alternatives
✔ Fast performance	✖ Small storage in a free version
✔ A lot of information	
✔ Easy to hire developers with GitHub experience	

Recommended use cases:
personal projects, small and medium businesses, open-source projects

WHAT'S BITBUCKET?

BitBucket was created by an Australian team and later acquired by Atlassian. The main selling point of BitBucket is the possibility to host an unlimited number of private repositories for small teams (1-5 users).

Pros	Cons
✔ Integration with Jira, Asana, and other Atlassian tools	✖ Complex interface
✔ High project visibility	✖ Slow performance
✔ Cost-efficiency	✖ Not enough free CI/CD functionality
✔ Integration with Slack and other tools	✖ Not so many tutorials and guides
	✖ Difficult to hire developers with experience
	✖ Decreasing popularity

Recommended use cases:
small, medium businesses

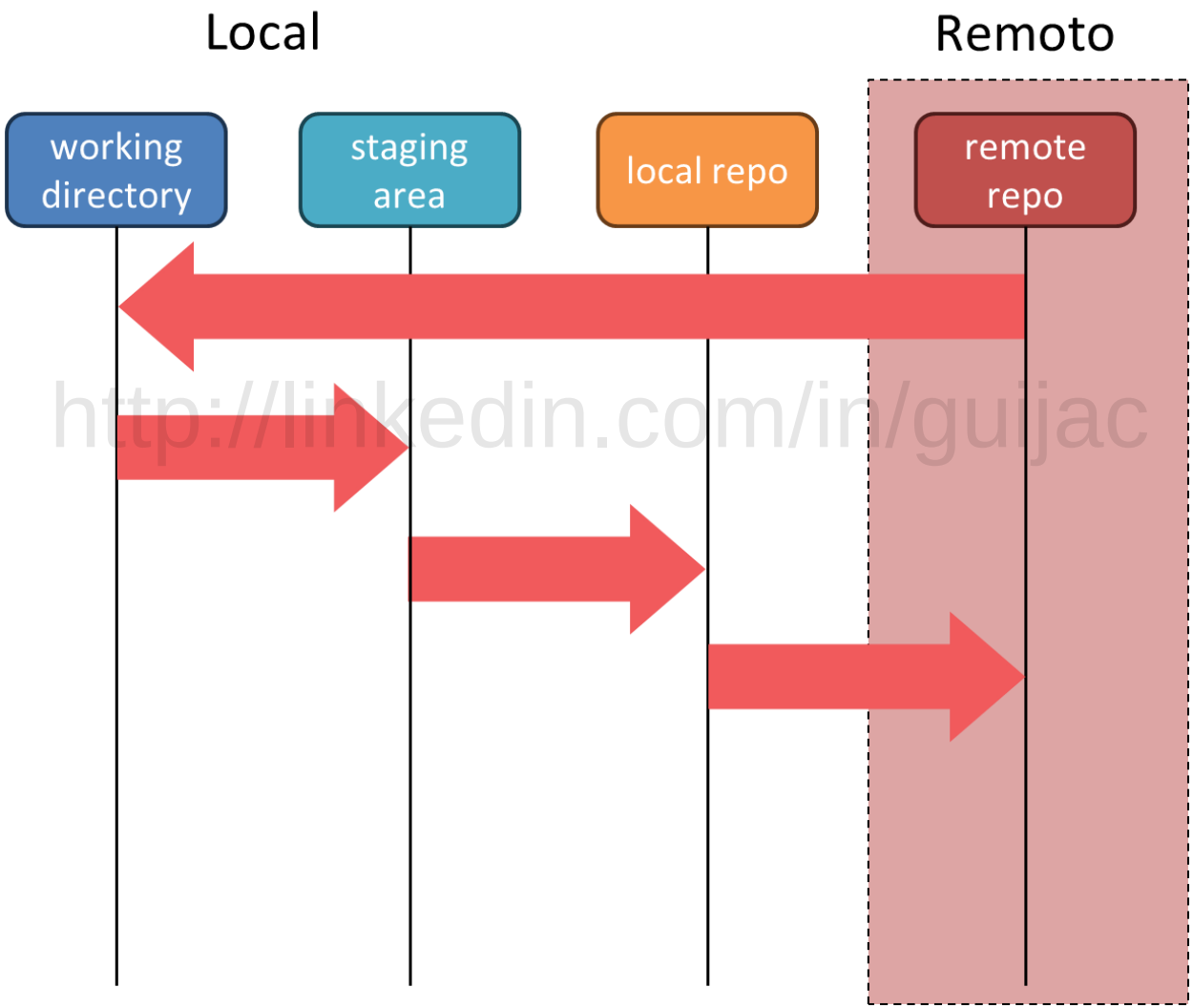
WHAT'S GITLAB?

GitLad was founded as an alternative to GitHub and BitBucket. The popularity of GitLab is growing due to the increased adoption of CI/CD and DevOps.

Pros	Cons
✔ Rich free CI/CD functionality	✖ Small open-source community
✔ Competitive pricing	✖ Relatively few users
✔ Rich add-ons	✖ Slow performance
✔ Simple interface	✖ Not enough guides and tutorials
✔ Rising popularity	

Recommended use cases:
medium-size businesses, enterprises

Arquitetura Git



Fonte: Elaboração Própria (2025)

O Arquivo README.md

- É um arquivo com extensão .md (Markdown), contendo informações necessárias para **entender o objetivo do projeto**, sendo considerado um **cartão de visitas** de um repositório Git.

README.md

flask-sql-injection

Aplicação Flask + BD para demonstração de uma vulnerabilidade SQL Injection identificada pelo SAST do .gitlab-ci.yml.

mydb.db

id	user	password
1	teste@teste.com	1234
2	teste2@teste.com	1234
3	teste@teste.com	12345678
4	teste2@teste.com	87654321

Código

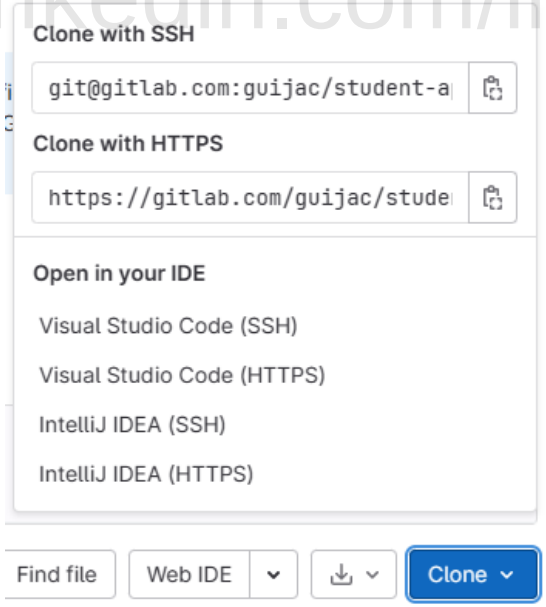
```
63 sql_Query_Not_Injection = text("select * from user where id=:user_id")
64 result = conn.execute(sql_Query_Not_Injection, user_id = id)
65
66 sql_Query_Injection_False_Negative = text("select * from user where id={}".format(id))
67 result = conn.execute(sql_Query_Injection_False_Negative)
68
69 # deprecated in SQLAlchemy >=2.0
70 sql_Query_Injection = "select * from user where id={}".format(id)
71 result = conn.execute(sql_Query_Injection)
```

- As linhas 63 e 64 **não contém** uma vulnerabilidade SQL Injection;
- As linhas 66 e 67 **contém** uma vulnerabilidade SQL Injection, **não identificável** no SAST do .gitlab-ci;
- As linhas 70 e 71 **contém** uma vulnerabilidade SQL Injection, **identificável** no SAST do .gitlab-ci.

```
1
2  # flask-sql-injection
3
4
5  Aplicação Flask + BD para demonstração de uma vulnerabilidade SQL Injection identificada pelo SAST do .gitlab-ci.yml.
6
7  ## mydb.db
8
9  |id|user |password|
10 |--|--|--|
11 |1 |teste@teste.com |1234|
12 |2 |teste2@teste.com |1234|
13 |3 |teste@teste.com |12345678|
14 |4 |teste2@teste.com |87654321|
15
16 ## Código
17
18     63 sql_Query_Not_Injection = text("select * from user where id=:user_id")
19     64 result = conn.execute(sql_Query_Not_Injection, user_id = id)
20     65
21     66 sql_Query_Injection_False_Negative = text("select * from user where id={}".format(id))
22     67 result = conn.execute(sql_Query_Injection_False_Negative)
23     68
24     69 # deprecated in SQLAlchemy >=2.0
25     70 sql_Query_Injection = "select * from user where id={}".format(id)
26     71 result = conn.execute(sql_Query_Injection)
27
28   - As linhas 63 e 64 **não contém** uma vulnerabilidade SQL Injection;
29   - As linhas 66 e 67 **contém** uma vulnerabilidade SQL Injection, **não identificável** no SAST do .gitlab-ci;
30   - As linhas 70 e 71 **contém** uma vulnerabilidade SQL Injection, **identificável** no SAST do .gitlab-ci.
```

Principais Comandos Git

- **git clone <https://link-do-repositório.git>**
  - Realiza a cópia de um projeto do repositório remoto para seu repositório local;
  - Outras possibilidades permitem um “clone” por protocolo SSH (geralmente para repositórios privados) ou diretamente por sua IDE, como VS Code.



Principais Comandos Git

- **git status**
  - Fornece informações relevantes sobre a situação dos arquivos em seu ambiente de trabalho (“working directory”) como arquivos que foram criados, modificados ou mesmo excluídos.
  - **Auxilia nos “próximos passos”.**


```
PS C:\Users\Guilherme\workspace\git-tests> git
status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what
will be committed)
        README.md

nothing added to commit but untracked files
present (use "git add" to track)
```

```
PS C:\Users\Guilherme\workspace\git-tests> git
status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
```

Principais Comandos Git

- **git add**
  - Arquivos modificados ou excluídos não são enviados diretamente ao repositório remoto;
  - É necessário adicionar estes arquivos para a fase de "*stage*", posteriormente executar outro comando para envio ao repositório local e, por fim, enviá-lo ao repositório remoto;
  - Aceita uma série de parâmetros, sendo os mais utilizados demonstrados na tabela abaixo:

	New files	Modified files	Deleted files	Files with names beginning with a dot	Current directory	Higher directories
`git add -A`	Yes	Yes	Yes	Yes	Yes	Yes
`git add .`	Yes	Yes	Yes	Yes	Yes	No
`git add -u`	No	Yes	Yes	Yes	Yes	Yes

Fonte: GITHUB (2025)

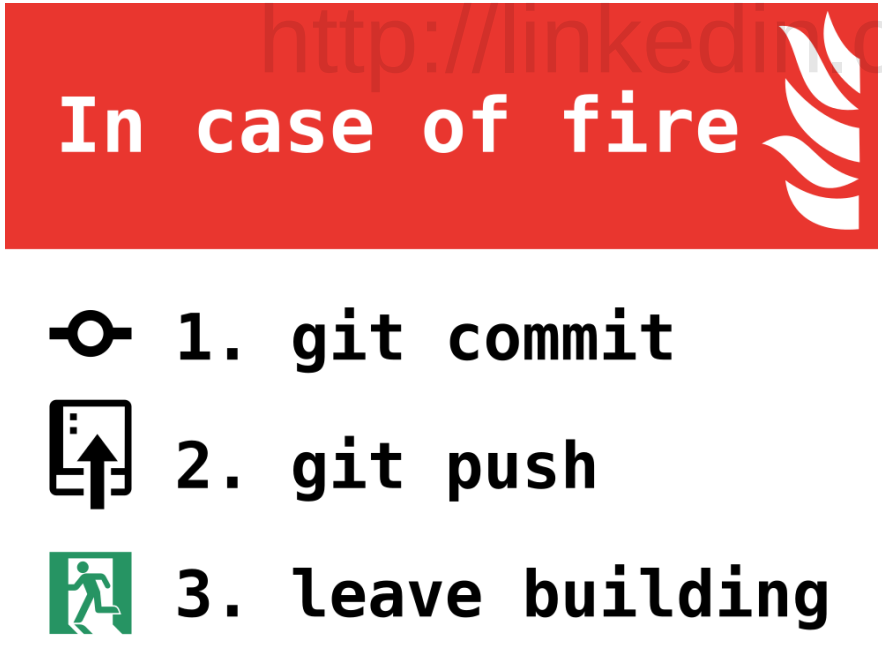
Principais Comandos Git

- **git commit**
  - Comando geralmente utilizado após a **conclusão de uma tarefa**, enviando as alterações ao repositório local;
  - Define um "***save state***" do trabalho realizado até então (alterações são salvas no **repositório git local**, sendo necessária a execução de um comando posterior para envio dos arquivos ao repositório remoto);
  - Também aceita uma série de parâmetros opcionais, sendo o mais utilizado o "-m", que define uma mensagem amigável para as alterações que foram realizadas em determinado *commit*.

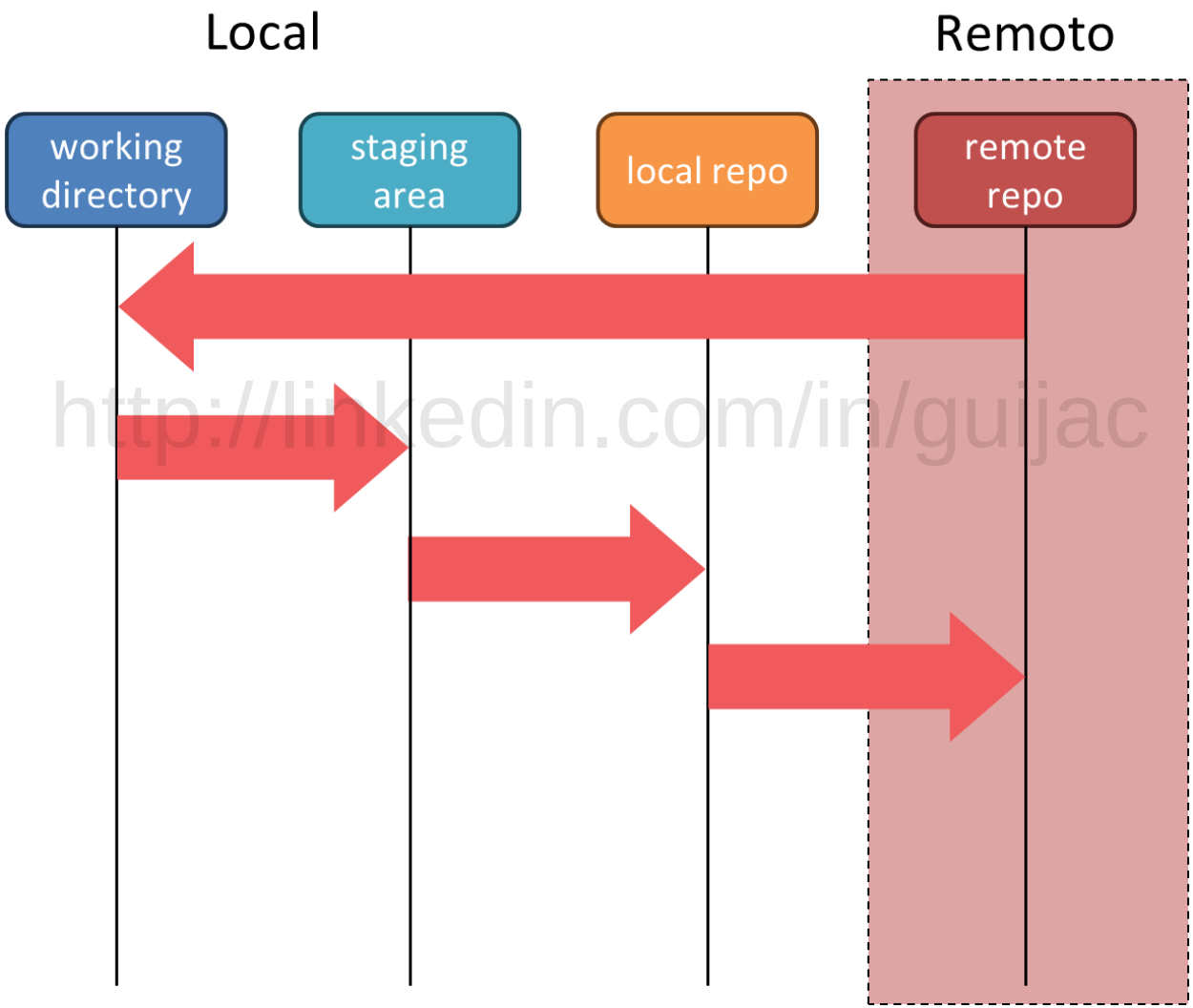

```
PS C:\Users\Guilherme\workspace\git-tests> git commit -m "[ADS-21] my first README.md file
#done"
[main (root-commit) 7e2cf2e] [ADS-21] my first README.md file #done
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

Principais Comandos Git

- **git push**
  - Realiza o **upload** para o **repositório remoto** dos arquivos salvos por um commit;
  - O comando só é executado após um commit previamente realizado.

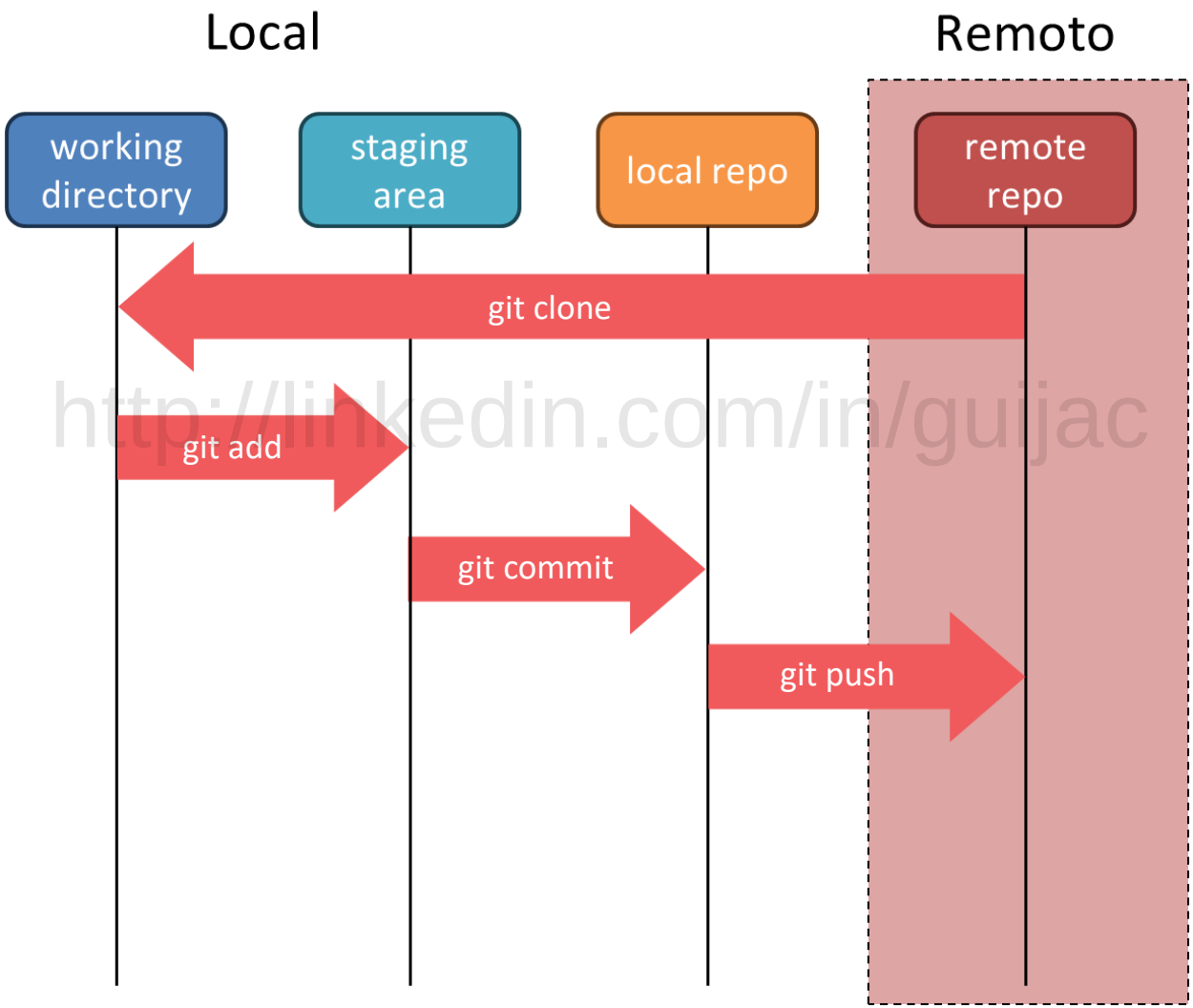


Juntando as Peças



Fonte: Elaboração Própria (2025)

Juntando as Peças



Fonte: Elaboração Própria (2025)

Referências Bibliográficas

Atlassian. **Gerenciamento de configurações**. Disponível em https://www.atlassian.com/br/microservices/microservices-architecture/configuration-management. Acesso em 14 jan 2025;

BINS**. Guia dos principais comandos do GIT**. Disponível em https://blog.dbins.com.br/guia-dos-principais-comandos-do-git. Acesso em 14 jan 2025;

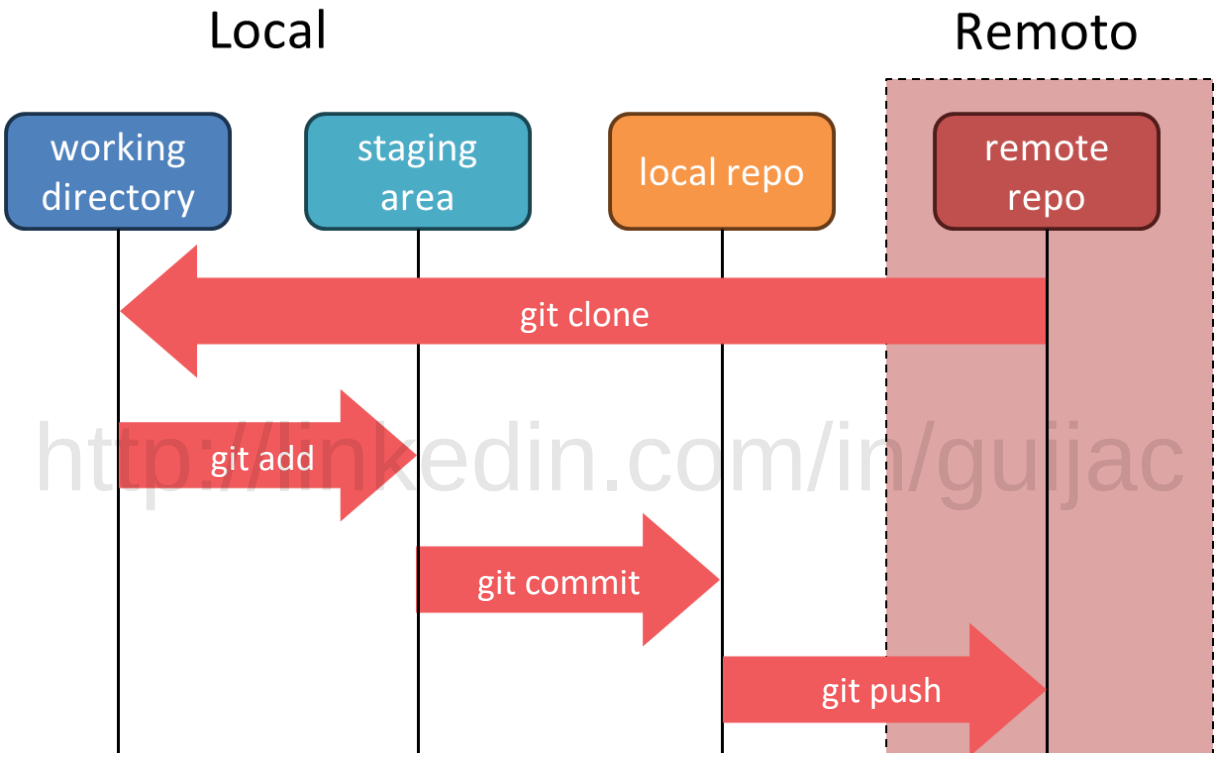
GIT. **Documentation**. Disponível em https://git-scm.com/doc. Acesso em 14 jan 2025;

GITHUB. **Git Guides**. Disponível em https://github.com/git-guides. Acesso em 14 jan 2025;

ROSA, Daniel. **10 comandos do Git que todo desenvolvedor deveria conhecer**. Disponível em https://www.freecodecamp.org/portuguese/news/10-comandos-do-git-que-todo-desenvolvedor-deveria-conhecer/. Acesso em 14 jan 2025;

SSL2BUY. **Infrastructure as Code (IaC) – A Developer's Perspective**. Disponível em https://www.ssl2buy.com/cybersecurity/infrastructure-as-code-iac. Acesso em 14 jan 2025;

Por hoje (de teoria!) é só!



Fonte: Elaboração Própria (2025)

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.jacruz@sp.senac.br

linkedin.com/in/guijac

Licença

- Este conteúdo está licenciado sob a Licença Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional (CC BY-NC-SA 4.0).
- Todos os direitos autorais sobre este conteúdo pertencem ao autor, e este material não pode ser usado comercialmente sem autorização expressa.
- Para ver o texto completo da licença, acesse o https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac