

Parte 1 – Conceitos Básicos

1 Conceitos Iniciais

(a) O que é **Git** e para que ele serve?

✓ **Resposta:** O **Git** é um sistema de controle de versão distribuído que permite registrar e gerenciar mudanças em arquivos e projetos de software ao longo do tempo. Ele facilita o trabalho colaborativo e o rastreamento de alterações.

(b) Qual a diferença entre **Git** e **GitHub**?

✓ **Resposta:** O **Git** é um sistema de controle de versão usado localmente para gerenciar projetos, enquanto o **GitHub** é uma plataforma online para armazenar e compartilhar repositórios Git remotamente.

(c) Explique os seguintes termos:

- **Commit** → Registra uma alteração no histórico do Git.
- **Repositório remoto** → Um repositório armazenado em uma plataforma como GitHub, acessível via internet.
- **Branch** → Uma ramificação do código que permite desenvolver funcionalidades isoladamente sem afetar a versão principal.
- **Merge** → Combina mudanças de uma branch com outra, geralmente unindo alterações da branch secundária para a principal (**main**).

git init – Inicializa um repositório Git

Cria um novo repositório Git em um diretório.

✓ **Exemplo:** `git init`

✚ **Explicação:** Esse comando cria a pasta `.git`, onde o Git armazena todo o histórico de versões do projeto.

2 git clone – Clona um repositório remoto

Baixa um repositório do GitHub (ou outro repositório remoto) para a máquina local.

✓ **Exemplo:**

`git clone https://github.com/seu_usuario/projeto.git`
`exemplo: https://github.com/ervalnetto/3tds2024.git`

✚ **Explicação:** Esse comando copia todos os arquivos e histórico do repositório remoto para um diretório local.

3 **git status** – Mostra o status do repositório

Exibe quais arquivos foram modificados, adicionados ou estão prontos para commit.

✓ **Exemplo:** `git status`

📌 **Explicação:** Ajuda a ver as mudanças feitas no projeto antes de confirmar um commit.

4 **git add** – Adiciona arquivos para o próximo commit

Coloca os arquivos no **staging area**, preparando-os para serem commitados.

✓ **Exemplo:** `git add meu_arquivo.txt`

Ou para adicionar **todos** os arquivos modificados:

`git add .`

📌 **Explicação:** Os arquivos precisam estar no staging antes de serem commitados.

5 **git commit** – Cria um commit com as mudanças

Salva uma versão do código no repositório local.

✓ **Exemplo:** `git commit -m "Adicionando nova funcionalidade"`

📌 **Explicação:** Cada commit registra uma nova versão do projeto, que pode ser recuperada posteriormente.

6 **git log** – Exibe o histórico de commits

Mostra uma lista dos commits feitos no repositório.

✓ **Exemplo:** `git log --oneline`

📌 **Explicação:** Com a opção `--oneline`, o histórico aparece resumido, facilitando a leitura.

7 **git branch** – Gerencia branches

Lista, cria ou exclui branches no repositório.

✓ **Exemplo (listar branches):** `git branch`

✓ **Criar uma nova branch:**

`git branch nova_funcionalidade`

📌 **Explicação:** Usar branches permite trabalhar em novas funcionalidades sem afetar o código principal.

8 **git checkout** – Alterna entre branches

Muda para outra branch no repositório.

✓ **Exemplo:** `git checkout nova_funcionalidade`

✓ **Criar e mudar para a branch ao mesmo tempo:**

`git checkout -b nova_funcionalidade`

📌 **Explicação:** Esse comando permite alternar entre diferentes versões do código.

9 **git merge** – Mescla branches

Combina as alterações de uma branch na branch atual.

Exemplo: `git merge nova_funcionalidade`

📌 **Explicação:** Normalmente, esse comando é usado para integrar o trabalho feito em uma branch secundária na branch principal (`main`).

10- **git push** & **git pull** – Enviar e baixar alterações do repositório remoto

Enviar alterações para o GitHub: `git push origin main`

Baixar as últimas mudanças do repositório remoto:

`git pull origin main`

📌 **Explicação:** `git push` envia commits para o GitHub, enquanto `git pull` atualiza o repositório local com as mudanças mais recentes.

Criar um diretório para o projeto

Escolha ou crie uma pasta onde seu projeto será armazenado.

```
mkdir meu-projeto  
cd meu-projeto
```

 **Explicação:** Criamos e entramos no diretório onde o repositório será inicializado.

2 Inicializar o repositório Git

Agora, inicialize o Git no diretório.

```
git init
```

 **Explicação:** Esse comando cria a pasta `.git`, onde o Git armazenará o histórico de versões do seu projeto.

3 Criar e adicionar arquivos ao repositório

Crie um arquivo no projeto e adicione-o ao controle de versão.


```
echo "# Meu Projeto" > README.md  
git add README.md
```

 **Explicação:** Criamos um arquivo `README.md` e o adicionamos ao **staging area**.

4 Criar o primeiro commit

Agora, registre a primeira versão do projeto.

```
git commit -m "Primeiro commit do projeto"
```

 **Explicação:** O commit salva as mudanças no repositório local.

5 Criar um repositório remoto no GitHub

1. Acesse [GitHub](https://github.com) e faça login.
2. Clique no botão **New Repository**.
3. Escolha um nome para o repositório e clique em **Create Repository**.
4. Copie o link do repositório (exemplo:
`https://github.com/seu-usuario/meu-projeto.git`).

6 Conectar o repositório local ao GitHub

Agora, conecte seu repositório local ao remoto.

```
git remote add origin https://github.com/seu-usuario/meu-projeto.git
```

 **Explicação:** Esse comando vincula o repositório local ao remoto.

Enviar o código para o GitHub

Agora, envie o repositório para o GitHub.

```
git push -u origin main
```

 **Explicação:** Esse comando envia o código para o repositório remoto na branch `main`.

```
// Importa o módulo HTTP

const http = require('http');

// Cria um servidor HTTP

const server = http.createServer((req, res) => {

  // Responde com um mensagem de boas-vindas

  res.writeHead(200, {'Content-Type': 'text/plain'});

  res.end('Bem-vindo ao meu servidor Node.js!\n');

});
```

```
// Define a porta do servidor

const port = 3000;

// Inicia o servidor

server.listen(port, () => {

  console.log(`Servidor iniciado na porta ${port}`);

});
```

```
#Calculadora

const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.question('Digite o primeiro número: ', (num1) => {
  rl.question('Digite o segundo número: ', (num2) => {
    const resultado = parseFloat(num1) + parseFloat(num2);
    console.log(`O resultado da soma é: ${resultado}`);
    rl.close();
  });
});
```