## 1. Por que o Git é considerado um sistema de controle de versão distribuído?

O Git é considerado um sistema de controle de versão **distribuído** porque **cada desenvolvedor (ou cada clone) tem uma cópia completa de todo o histórico do projeto**, e não apenas um *snapshot* do estado atual.

- Em um sistema centralizado, se o servidor principal falhar, o histórico completo do projeto pode ser perdido.
- No Git, cada cópia local (clone) atua como um **backup completo** do projeto e de seu histórico, permitindo que o trabalho continue mesmo sem conectividade com o servidor central, e facilitando a colaboração ponto a ponto.

**2. Qual a diferença entre working directory, staging area e repository?**

Working Directory: É a sua **área de trabalho atual**. Os arquivos que você vê e edita no seu sistema de arquivos.

Staging Area: É um arquivo que armazena informações sobre o que irá para o próximo *commit*. É uma **ponte** entre o Working Directory e o Repository.

Repository: É onde o Git armazena todo o histórico do projeto (os *commits*).

## 3. Para que serve o comando `git clone`?

- **Resposta:** O comando `git clone` é usado para **criar uma cópia local (clone) de um repositório Git remoto** (geralmente hospedado em serviços como GitHub, GitLab, ou Bitbucket).
  - Ele baixa todo o histórico do projeto, incluindo todos os arquivos, *commits* e *branches*, tornando a cópia local um **repositório Git completo e funcional**.
  - É o ponto de partida para a maioria dos desenvolvedores ao começar a trabalhar em um projeto existente.

**4. Onde estão implementados fisicamente working directory, staging area e repository?**

Working Directory: Os arquivos e pastas que você vê no diretório do seu projeto (fora da pasta `.git`).

Staging Area: O arquivo chamado **`index`** localizado dentro da pasta oculta `.git` (ou `.git/index`).

Repository: Principalmente dentro da pasta oculta `.git` na raiz do projeto. O histórico dos *commits* e os objetos do Git (arquivos de dados) são armazenados em `.git/objects`.

## 5. Quais os estados de um arquivo no repositório do Git?

Os arquivos no seu diretório de trabalho podem estar em um dos quatro estados principais (quando o Git está rastreando-o):

1. Untracked (Não Rastreado): O Git não está rastreando este arquivo. É um arquivo novo que você adicionou ao projeto e que o Git ainda não viu.
2. Unmodified (Não Modificado): O arquivo está inalterado desde o último *commit*. A versão no seu *working directory* é a mesma que está no repositório.
3. Modified (Modificado): O arquivo foi alterado no seu *working directory* em comparação com o último *commit*, mas ainda não foi adicionado à *staging area*.
4. Staged (Preparado): O arquivo modificado foi adicionado à *staging area* com `git add` e está pronto para ser incluído no próximo *commit*.

---

**6. Explique as possíveis transições de estado de um arquivo no repositório do Git?**

**No Git, um arquivo pode ter os seguintes estados e transições:**

1. Untracked (Não rastreado): O arquivo não é rastreado pelo Git.
- Transição: Untracked -> Staged (ao executar git add <arquivo>)
2. Staged (Preparado): O arquivo está preparado para ser commitado.
- Transição: Staged -> Committed (ao executar git commit)
- Transição: Staged -> Modified (ao modificar o arquivo após git add)
- Transição: Staged -> Untracked (ao executar git reset HEAD <arquivo> ou git restore --staged <arquivo>)
3. Modified (Modificado): O arquivo foi modificado, mas não está preparado para ser commitado.
- Transição: Modified -> Staged (ao executar git add <arquivo>)
- Transição: Modified -> Unmodified (ao reverter as modificações)
4. Unmodified (Não modificado): O arquivo não foi modificado desde o último commit.
- Transição: Unmodified -> Modified (ao modificar o arquivo)
5. Committed (Commitado): O arquivo foi commitado e está no repositório do Git.
- Transição: Committed -> Modified (ao modificar o arquivo)

**Perguntas:**
**1. Qual o estado do arquivo antes e depois do git add ? 💡**

**Resposta: O arquivo ficava como Untracked ( não rastreavel)**

**2. O que significa o estado untracked e tracked ?**
💡 **Resposta:**
Untracked (Não Rastreado)  O arquivo existe no seu Working Directory, mas o Git nunca o viu antes e não está monitorando ativamente suas modificações.

Tracked (Rastreado)  O Git conhece e monitora o histórico deste arquivo.

**3. Qual o objetivo do git commit ?**
💡 **Resposta:**
O objetivo do comando `git commit` é salvar permanentemente um *snapshot* (uma "foto") dos arquivos que estão na Staging Area no repositório local.

**4. Qual o estado do arquivo após o git commit ?**
💡 **Resposta:**
O estado do `arquivo.txt` após o `git commit` é Unmodified (Não Modificado).

**Perguntas:**
**1. O que o comando git diff mostra?**
💡 **Resposta:**
O comando `git diff` é usado para mostrar as diferenças (delta) entre duas versões de arquivos que o Git conhece.

**2. Qual commit está atualmente apontado por HEAD?**
💡 **Resposta:**
O `HEAD` está atualmente apontado para o seu último (mais recente) *commit*

***Perguntas:***
***1. Como verificar em qual branch você está?***
💡 ***Resposta:***
*git branch e o git status*

***2. O que acontece se você rodar git merge nova-feature estando na branch principal?***
💡 ***Resposta:***

*Se você voltar para a branch principal (ex: git checkout main) e rodar o comando git merge nova-feature, o Git tentará integrar as alterações feitas na branch nova-feature no histórico da branch principal.*