Diversos - Sistemas

- Guia basico do GIT
- Por que usar ambiente de desenvolvimento baseado em Docker

Guia basico do GIT

Git é uma ferramenta poderosa para o controle de versões. Aqui estão alguns comandos básicos que você precisa conhecer para começar a trabalhar com o Git:

git clone

Este comando é usado para clonar um repositório do Git de outro lugar para o seu computador local. Por exemplo:

```
git clone https://github.com/username/repo.git
```

git init

Este comando é usado para inicializar um novo repositório Git em um diretório existente. Por exemplo:

```
git init
```

git add

Este comando é usado para adicionar arquivos à área de stage do Git. Por exemplo:

```
git add file.txt
```

git commit

Este comando é usado para salvar as mudanças feitas no stage para o repositório. Por exemplo:

```
git commit -m "Adicionando um arquivo de exemplo"
```

git push

Este comando é usado para enviar as mudanças locais para o repositório remoto. Por exemplo:

```
git push origin main
```

git pull

Este comando é usado para puxar as mudanças do repositório remoto para o seu computador local. Por exemplo:

```
git pull origin main
```

git reset

Este comando é usado para desfazer mudanças no repositório. Por exemplo:

```
git reset HEAD file.txt
```

dica

Lembre-se de que é importante ser cuidadoso e ter atenção ao seu fluxo de trabalho com o Git, pois as mudanças são permanentes e podem afetar todo o histórico do seu projeto. Sempre revise suas ações antes de confirmá-las, e use branchs para manter o código estável enquanto você trabalha nas funcionalidades.

Por que usar ambiente de desenvolvimento baseado em Docker

Quando se trata de múltiplos projetos com recursos computacionais diferentes, usar um ambiente de desenvolvimento baseado totalmente em Docker pode ser uma decisão mais sensata do que instalar tudo na própria máquina. Isso se torna ainda mais importante quando a equipe não possui um grande conhecimento técnico ou se há membros da equipe que não conseguem acompanhar as exigências do projeto.

Usar Docker permite que cada projeto tenha suas próprias dependências, bibliotecas e configurações isoladas, evitando conflitos e problemas de compatibilidade. Por exemplo, um projeto pode precisar de uma versão específica de uma biblioteca, enquanto outro projeto pode precisar de uma versão diferente. Com Docker, cada projeto pode ter sua própria imagem com suas próprias configurações, evitando conflitos entre projetos. Além disso, o Docker facilita a manutenção e atualização do ambiente de desenvolvimento, já que tudo está encapsulado em containers. Se um membro da equipe sair ou mudar de projeto, as configurações do ambiente ficam intactas, evitando problemas para o time restante.

Outra vantagem é que o Docker permite a reprodução exata do ambiente de desenvolvimento em qualquer máquina, garantindo que o projeto possa ser desenvolvido em qualquer lugar, sem se preocupar com a configuração do sistema. Isso é especialmente útil quando se trabalha em equipe, já que cada desenvolvedor pode ter uma configuração diferente na sua máquina local, mas todos usam a mesma imagem do Docker para desenvolver o projeto, garantindo que todos trabalhem com a mesma configuração.

Em resumo, usar ambiente de desenvolvimento baseado em Docker é uma escolha inteligente porque permite uma gestão mais fácil e eficiente de múltiplos projetos, além de facilitar a manutenção, evitar problemas técnicos relacionados a configurações e compatibilidade de recursos e garantir a reprodução exata do ambiente em qualquer máquina.