O que é Git

O Git é um projeto de código aberto maduro e com manutenção ativa desenvolvido em 2005 por Linus Torvalds, o famoso criador do kernel do sistema operacional Linux. Um número impressionante de projetos de software depende do Git para controle de versão, incluindo projetos comerciais e de código-fonte aberto.

Os desenvolvedores que trabalharam com o Git estão bem representados no pool de talentos de desenvolvimento de software disponíveis e funcionam bem em uma ampla variedade de sistemas operacionais e IDEs (Ambientes de Desenvolvimento Integrado).

Tendo uma arquitetura distribuída, o Git é um exemplo de DVCS (portanto, Sistema de Controle de Versão Distribuído). Em vez de ter apenas um único local para o histórico completo da versão do software, como é comum em sistemas de controle de versão outrora populares como CVS ou Subversion (também conhecido como SVN), no Git, a cópia de trabalho de todo desenvolvedor do código também é um repositório que pode conter o histórico completo de todas as alterações.

Além de ser distribuído, o Git foi projetado com desempenho, segurança e flexibilidade em mente.

GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo. GitHub é amplamente utilizado por programadores para divulgação de seus trabalhos ou para que outros programadores contribuam com o projeto, além de promover fácil comunicação através de recursos que relatam problemas ou misturam repositórios remotos (*issues, pull request*).

O GitHub é mundialmente usado e chega a ter mais de 36 milhões de usuários ativos mundialmente contribuindo em projetos comerciais ou pessoais. Hoje o GitHub abriga mais de 100 milhões de projetos,[1] alguns deles que são conhecidos mundialmente. WordPress, GNU/Linux, Atom, Electron. GitHub também oferece suporte ao recurso de organização que é amplamente utilizado por aqueles que querem uma escala maior para seus projetos. Na maioria das vezes, o recurso é usado por empresas já existentes como a Google, Microsoft e WordPress.

O **Docker** é um software de código aberto usado para implantar aplicativos dentro de containers virtuais. A conteinerização permite que vários aplicativos funcionem em diferentes ambientes complexos. Por exemplo: o Docker permite executar o WordPress em sistemas Windows, Linux e macOS, sem problemas.

Docker é um conjunto de produtos de plataforma como serviço (PaaS) que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres.

Os contêineres são isolados uns dos outros e agrupam seus próprios softwares, bibliotecas e arquivos de configuração. Eles podem se comunicar uns com os outros por meio de canais bem definidos. Todos os contêineres são executados por um único kernel do sistema operacional e, portanto, usam menos recursos do que as máquinas virtuais. .

O que é Docker?

O Docker permite que você execute o software em um ambiente isolado denominado contêiner. Um contêiner é semelhante a uma máquina virtual (VM), mas opera de uma maneira completamente diferente.

Sendo assim, embora forneçam a maior parte do isolamento que uma VM oferece, os contêineres usam apenas uma fração dos recursos disponíveis em seu conjunto de produtos.

Consistência permite gerenciamento fácil do projeto

Digamos que você esteja codificando um aplicativo da web e desenvolvendo em sua máquina local, onde o testa. Em algumas situações, você o executa em um servidor de teste e, em breve, o colocará em um grande servidor de produção para que o mundo o veja.

Neste caso, não seria ótimo se você pudesse ativar consistentemente o mesmo ambiente, exatamente da mesma forma, em todos os seus dispositivos? Bem, se a resposta for “sim, o Docker pode te ajudar.

Se o seu aplicativo da web for executado corretamente dentro de um contêiner do Docker, em sua caixa local, ele poderá ser executado em seu servidor de teste, no servidor de produção e em qualquer lugar.

Dessa forma, isso torna o gerenciamento das dependências do projeto incrivelmente fácil. Não só é simples lidar com bibliotecas externas e módulos chamados diretamente pelo seu código, mas todo o sistema pode ser configurado ao seu gosto.

Assim, se você tiver um projeto de código aberto em uma situação onde um novo usuário deseja baixar e executar, ao utilizar o Docker, o processo é tão simples quanto iniciar um contêiner.

Proporciona escalabilidade

Agora, se você usar o Docker para criar serviços com demandas variadas (como sites ou APIs), é incrivelmente fácil escalar seu provisionamento simplesmente ativando mais contêineres (desde que tudo esteja arquitetado corretamente).

Para isso, existem várias estruturas que possibilitam orquestrar clusters de contêiner, como Kubernetes e Docker Swarm, mas isso é assunto para outro post. Então, fique de olho nos conteúdos do nosso blog!

Como o Docker funciona

Bem, muitas pessoas acreditam que o Docker é uma forma de executar uma máquina virtual, mas não é bem assim. Uma máquina virtual é executada em hardware simulado, ou seja, um sistema operacional totalmente autocontido.

No entanto, os contêineres compartilham nativamente o kernel do host. Isso significa que os contêineres têm um desempenho incrivelmente melhor do que as máquinas virtuais, dependendo do que eles estão fazendo.

Assim, seria possível criar centenas, senão milhares de contêineres em seu PC. Além disso, a inicialização de um contêiner levará segundos ou menos, em comparação com minutos para muitas VMs.

Como os contêineres são leves, é prática comum executar todos os aspectos de um aplicativo em contêineres diferentes, para uma capacidade de manutenção e modularidade maiores.

Por exemplo, você pode ter diferentes contêineres para subir o banco de dados, redis, nginx e assim por diante, todos conseguindo manter a comunicação entre si, de forma harmoniosa.

Como eu uso o Docker?

A fim de exemplificar o uso do Docker, vamos construir um servidor web mínimo em um contêiner. No interesse de mantê-lo simples, usaremos Flask, um microframework da web em Python.

Este é o programa que queremos executar: main.py. Você pode criar uma pasta nova, que chamaremos de *app* e salvar este arquivo lá:

```
from flask import Flask app = Flask (__ name__)

@ app.route (‘/’)

 def home ():

    return ‘Hello from dentro de um contêiner Docker’

if __name__ == ‘__main__’:

     app.run(host = ‘0.0.0.0’, port = 80)
```

Neste caso, não precisa se preocupar se você não estiver familiarizado com o Flask ou Python, pois, tudo o que você precisa saber é que este código retorna uma string em localhost: 80.

Porém, antes de fazer esse app rodar, precisamos entender o que é um contêiner. Eles são definidos por uma imagem (como uma receita), sendo apenas uma instância em execução de uma imagem. Assim, você pode ter vários contêineres em execução da mesma imagem.