Fechamento

Parabéns por ter chegado até aqui!!! E agora vamos recapitular todos os assuntos que vimos e incluir algumas considerações em relação ao mercado na utilização dessas tecnologias.

Containers

É a solução do momento para melhor aproveitamento de recursos físicos (servidores), uma vez que não é necessário adquirir 1 servidor para cada aplicação, nem criar um servidor virtual com toda a gestão que vem junto, como o tempo para iniciar o servidor, gestão do sistema operacional (incluindo patches, etc).

Com containers conseguimos disponibilizar soluções em segundos, usando melhor os recursos físicos do servidor (como CPU e memória RAM por exemplo) direcionados para a necessidade da aplicação contida no container.

Vimos vários exemplos de aplicações que podem ser executadas em container como banco de dados, aplicações de back-end como APIs e aplicação de front-end como nosso site.

Vimos como os containers conseguem comunicar-se uns com os outros através de redes e como o sistema operacional host consegue acessar essas aplicações através do bind de portas.

Trabalhamos com volumes para não perdermos dados de alguns containers e vimos como esses volumes podem ser reutilizados para a guarda permanente de dados

mesmo que o container não seja permanente, ou seja, quando o container precisa ser destruído (por qualquer que seja o motivo) e recriado.

Docker

Abordamos detalhadamente como trabalhar com containers através da tecnologia oferecida pelo Docker, que faz a gestão desses containers como um hypervisor e permite não só a gestão dos containers, como a gestão de redes (através da criação de network), de volumes e de imagens.

Imagens essas que conseguimos construir para termos o banco de dados em Oracle, as aplicações em back-end utilizando Java e Python e nosso front-end feito em React.

Conseguimos através de imagens base criar uma imagem nossa que representa a base à qual nossa aplicação rodará (uma versão enxuta do Ubuntu, ou uma base que tenha o Java instalado ou que tenha o Node) e o build das nossas diversas aplicações (banco de dados e software codado).

Conseguimos usar o docker através de vários comandos, a interface visual do Docker Desktop também usa os mesmos comandos, a diferença é que conseguimos ver de forma visualmente mais agradável do que os comandos no Prompt de Comandos do Windows, mas basicamente são as mesmas funcionalidades.

Conseguimos criar containers através de imagens pré-prontas do Docker Hub ou criar containers de imagens que nós mesmos construímos.

Pipelines

Entendemos como automatizar nossos processos de desenvolvimento através das pipelines, através delas conseguimos implementar uma cultura de DevOps prevendo as 3 práticas:

- Continuous Integration para que pessoas consigam colaborar continuamente com código, submetendo e revisando o código de forma a integrar esses códigos em um produto final

- Continuous Delivery para que as pessoas consigam transformar código em algo que dê para ser implantado e testado, para obter feedback rápido caso o software tenha algum problema

- Continuous Deployment para que todo esse processo possa ser automatizado de ponta-a-ponta, não necessitando de nenhum processo manual, permitindo a entrega de software várias vezes num mesmo dia

A automação via pipelines evita erros humanos, uma vez que não depende de uma pessoa que saiba ou lembre como fazer determinado passo da esteira de implantação, como a automação prevê seguir os mesmos passos da mesma maneira sempre, diminui-se o risco de erros na operação como um todo (sobrando apenas erros de código) e assim trazendo maior valor para aquilo que o desenvolvedor tem que se preocupar: codar regras de negócio.

Gitlab CI/CD

Vimos como usar a ferramenta Gitlab CI/CD para implementar Continuous Integration, Continuous Delivery e Continuous Deployment através de um script

feito em uma linguagem legível ao ser humano (YAML) onde conseguimos configurar o comportamento que a pipeline terá.

Além do uso do Gitlab como repositório e versionamento de código, usamos a pipeline do Gitlab CI/CD para criarmos as nossas releases testáveis, apartando essas releases do ambiente de desenvolvimento!

Assim conseguimos ter um ambiente de desenvolvimento estável para uso dos desenvolvedores e, separadamente, um outro ambiente de release para que consigamos testar uma release específica sem impactar os demais desenvolvedores que dependem do ambiente de desenvolvimento.

Incluímos regras de quando executar jobs partindo de uma ou outra branch, como indicar quando há algum passo manual na esteira (sim, às vezes é necessário), como usar variáveis no script, tanto variáveis normais que criamos como variáveis que já existem internas do Gitlab CI/CD e ainda o que chamamos de "secrets", quando queremos guardar algum valor como senhas e usar na esteira sem o risco de expor o conteúdo dessas variáveis.

Vimos como configurar um Gitlab Runner para que executemos essas pipelines na nossa máquina, manipulando no Docker que está instalado para limpar e criar imagens, limpar e criar containers e subir do banco de dados ao site!

Explore mais...

O Docker é um excelente ponto de partida para o uso de containers, afinal foi praticamente a ferramenta tecnológica que revolucionou esse mundo de disponibilização de aplicações através de uma infraestrutura mais simples para se gerenciar recursos físicos.

Mas como toda tecnologia sempre há evolução a partir dela, então recomendo explorar outros mecanismos de containers como o que indico aqui: o Kubernetes.

O Kubernetes (https://kubernetes.io/pt-br/) é uma tecnologia para orquestração de containers que usa os mesmos conceitos do Docker, a diferença é a forma de gerenciar todos os recursos que são utilizados pelos contêineres, inclusive como subir vários containers como se fossem parte de um mesmo micro ecossistema (os pods no Kubernetes).

Vários provedores de nuvem tem soluções que são baseadas no Kubernetes, como o EKS (Amazon Elastic Kubernetes Service na AWS), GKE (Google Kubernetes Engine na Google Cloud Platform) e o AKS (Azure Kubernetes Service na Azure) são alguns exemplos de como cada provedor de nuvem utiliza tecnologia de containers através do Kubernetes.

O uso de Kubernetes traz as facilidades da gestão e orquestração de containers e, como é utilizado por vários provedores de nuvem, torna eventual migração de suas aplicações de um provedor para outro, uma vez que o Kubernetes é (a grosso modo) suportado da mesma maneira por todos.

Atividade Extra

Para se aprofundar no assunto desta aula leia os estudos de caso: “Casos de estudo utilizando Kubernetes”.

Link dos casos de uso: https://kubernetes.io/pt-br/case-studies/

Referência Bibliográfica

Orquestração de containers prontos para produção. Disponível em https://kubernetes.io/pt-br/. Acesso em 27 de junho de 2022.

Ir para exercício