Atividade de Revisão - Microsserviços, Docker e Kubernetes

Atividade: Fundamentos e Prática com Microsserviços

1. 1. O que caracteriza uma arquitetura de microsserviços?

É uma abordagem de desenvolvimento de software onde a aplicação é dividida em pequenos serviços independentes, que se comunicam entre si, cada um com responsabilidade específica, podendo ser desenvolvidos, implantados e escalados de forma independente.

2. 2. Compare os modelos monolítico e de microsserviços.

Vantagens dos microsserviços:

- Maior escalabilidade: cada serviço pode escalar separadamente conforme sua necessidade.
- Independência de desenvolvimento: equipes diferentes podem trabalhar em serviços distintos ao mesmo tempo.
- Manutenção facilitada: alterações em um serviço não afetam diretamente os outros.

Desvantagens:

- Maior complexidade na comunicação entre serviços.
- Necessidade de ferramentas adicionais para orquestração, monitoramento e deploy.

3. 3. Separação em microsserviços na plataforma de ensino online:

- Autenticação de usuários: responsável por login, logout, gerenciamento de tokens.
- Catálogo de cursos: lista e organiza os cursos disponíveis.
- Gestão de vídeo-aulas: armazena e distribui os vídeos.
- Certificados: gera e emite os certificados para alunos que concluírem cursos.

Justificativa: Cada funcionalidade possui lógica de negócio distinta e pode ter volume de acesso diferente, exigindo escalabilidade separada.

4. 4. Duas formas de comunicação entre microsserviços:

- Síncrona (HTTP/REST ou gRPC): ideal quando é necessário uma resposta imediata.

- Assíncrona (mensageria com Kafka, RabbitMQ): recomendada quando a resposta pode ser processada depois, reduzindo o acoplamento entre serviços.

5. 5. Comandos Docker para serviço Node.js:

a) Criar a imagem:

bash

docker build -t pedido-service .

b) Executar o container:

bash

docker run -p 8080:3000 pedido-service

6. 6. Cada microsserviço deve ter seu próprio banco de dados? Por quê?

Sim, pois isso garante isolamento, independência e evita acoplamento entre serviços, além de permitir que cada um use o tipo de banco mais adequado.

Atividade: Fundamentos e Prática com Docker

1. 1. O que é Docker e qual problema resolve?

Docker é uma plataforma de containers que padroniza e isola ambientes de execução, resolvendo problemas de 'funciona na minha máquina' e facilitando deploys.

2. 2. Comandos Docker:

a) Listar imagens:

bash

docker images

b) Ver containers ativos:

bash

docker ps

c) Parar container:

bash

docker stop abc123

d) Remover imagem:

bash

docker rmi meu-app:latest

3. 3. Comando para rodar NGINX no modo detached:

bash

docker run -d -p 8080:80 --name meu-nginx nginx

4. 4. Explicação do Dockerfile (exemplo típico):

FROM node:16 -> Define a imagem base (Node.js v16)

WORKDIR /app -> Define o diretório de trabalho

COPY . . -> Copia todos os arquivos do projeto para o container

RUN npm install -> Instala as dependências do projeto

CMD ["npm", "start"] -> Comando para iniciar a aplicação

5. 5. Comandos para build e execução:

a) Construir imagem:

bash

docker build -t meu-projeto:1.0 ./meu-projeto

b) Executar container:

bash

docker run meu-projeto:1.0

6. 6. docker-compose.yml:

a) Quantidade de serviços: Dois serviços (por exemplo: web e db)

b) Porta exposta pela aplicação web: A porta definida com ports, geralmente algo como 8080:80 -> 8080

Atividade: Fundamentos em Apache Kafka e Kubernetes

1. 1. O que é Apache Kafka?

Kafka é uma plataforma de streaming distribuída usada para enviar, armazenar e processar mensagens em tempo real.

Resolve problemas de comunicação assíncrona e desacoplamento entre serviços em sistemas distribuídos.

2. 2. Conceitos:

- Producer (produtor): envia mensagens para um tópico.
- Consumer (consumidor): recebe mensagens do tópico.
- Topic (tópico): canal onde as mensagens são publicadas e consumidas.

3. 3. Exemplo no app de pedidos de comida:

- Tópico: novo-pedido
- Produtor: o app mobile envia um novo pedido.
- Consumidores: o sistema da cozinha e o de pagamentos recebem e processam o pedido.

Atividade de Revisão - Microsserviços, Docker e Kubernetes

4. 4. O que é Kubernetes?

É uma plataforma de orquestração de containers que automatiza o deploy, o escalonamento e a gestão de aplicações containerizadas em múltiplos hosts.

5. 5. Associação de termos:

- (a) Pod -> Representa a menor unidade de execução no Kubernetes, geralmente com um ou mais containers.
- (b) Cluster -> Um conjunto de máquinas (físicas ou virtuais) que executam aplicações em Kubernetes.
- (c) Node -> Cada máquina (worker ou master) que compõe o ambiente do Kubernetes.
- (d) Service -> Um ponto de acesso estável para se comunicar com os pods, mesmo que eles mudem de IP.