INFRAESTRUTURA PARA SISTEMAS DE SOFTWARE

Arquitetura de Microsserviços para o Desenvolvimento de Aplicações

ROTEIRO

- Definição
- Características
- Benefícios e Complexidades
- Monolítico x Microsserviços
- O que preciso para MSA?
- Boas práticas
- Conclusão

DEFINIÇÃO

- Micros Services Architecture (MSA) é um padrão para construção de aplicações distribuídas

  - Um conjunto de serviços, cada um executando em seu próprio processo, cada um explorando uma API
  - Desenvolvido de forma independente
  - Independentemente implantável
  - Cada serviço está focado em fazer determinada tarefa de forma bem feita

DEFINIÇÃO

- Uma MSA não é:
  - Monolítica
  - Em camadas e multifuncional
  - Integrada de forma inteligente
  - Integrada de forma centralizada

CARACTERÍSTICAS

- Em relação a outras arquiteturas de software:
  - Domínio da aplicação com granularidade fina
  - Aumentam a disponibilidade geral da aplicação
  - Desenvolvimento da solução com equipes menores e mais autônomas
    - São mais produtivos
    - Melhor utilização de recursos

BENEFÍCIOS E COMPLEXIDADES

- Valores e Princípios
  - Autonomia
  - Velocidade de Mudança
  - Escala
  - Compossibilidade
  - Diversidade Tecnológica

BENEFÍCIOS E COMPLEXIDADES

- Valores e Princípios
  - O isolamento traz melhor disponibilidade
  - Velocidade de entrega independente (por diferentes times
  - Governança descentralizada (DevOps)

BENEFÍCIOS E COMPLEXIDADES

- Complexidades
  - Comunicação
  - Execução
  - Resiliência
  - Manutenção
  - Operacionalização

BENEFÍCIOS E COMPLEXIDADES

- Complexidades
  - Sistemas distribuídos são complexos
  - Sobrecarga operacional (centenas de milhares de serviços)
    - modelo DevOps extremamente necessário
  - Controle de versões da interface de serviço
  - É necessário que todo o ecossistema seja testado
  - Aumenta o tráfego de mensagens trocadas entre os compone
    de uma aplicação

BENEFÍCIOS E COMPLEXIDADES

- Complexidades
  - Utilizar MSA não significa automaticamente melhorar a disponibilidade da aplicação
    - É preciso ter uma arquitetura tolerante a falhas
      - Apenas um componente distribuído de um conjunto de centenas deles, pode levar à indisponibilidade da aplicação

MONOLÍTICO X MICROSSERVIÇOS

- Monolítico
  - Mais simples de testar e desenvolver
  - Mais simples para implantar (deploy)
  - Não pode implantar algum componente se não implanta tudo
  - Mais complexo para aprender e entender o código (centenas de milhares de linhas de código)

MONOLÍTICO X MICROSSERVIÇOS

- Monolítico
  - Mais complexo para adaptar novas tecnologias
  - É preciso escalar tudo da aplicação, para escalar um dos seus componentes

MONOLÍTICO X MICROSSERVIÇOS

- A MSA não é uma solução para todos os problemas de desenvolvime de aplicações distribuídas modernas
  - Se você já possui um ambiente de teste e implantação automatizados e deseja escalar seu ambiente, sim, a MSA pode ser uma boa opção
  - Se você não tem testes automatizados, primeiro essa questão precisa ser resolvida
  - É preciso ter amplos conhecimentos em implantação automatizada, teste e monitoramento dos serviços para colher o benefícios da MSA

O QUE PRECISO PARA MSA?

- Se está apenas começando, fique monolítico até que você entenda o problema melhor
- Ser bom em infraestrutura de provisionamento
- Ser rápido na implantação de aplicações
- Ter noções sobre monitoramento de serviços
- Ter um bom domínio do sistema e compreender se há necessidade de modificação no estilo de desenvolvimento

BOAS PRÁTICAS

- Qual o tamanho de um microsserviço?
  - Acoplamento solto
  - Uma mudança no serviço X não deve exigir uma mudança no se Y
  - API pequena e bem focada
  - Alta coesão
  - Cada serviço deve ter uma responsabilidade específica
  - O comportamento específico do domínio deve estar em um só lu
  - Se você precisa mudar um comportamento, você não deveria te mudar múltiplos serviços

BOAS PRÁTICAS

- Qual o tamanho de um microsserviço?
  - Quanto menor o serviço, maior é o benefício com o desacoplamento
  - Você deve ser capaz de reescrever um serviço de forma rápida
  - Se você não puder fazer uma alteração em um serviço e implantá-lo sem modificar outras partes, então há um grande problema

BOAS PRÁTICAS

- Mantenha seus ambientes tão próximos da produção quanto da prática
  - Um serviço por host
  - Minimize o impacto de um serviço sobre os outros
  - Minimize o impacto de uma interrupção do host
  - Utilize contêineres / vms para tornar o processo mais fáci e simples
  - Os contêineres mapeiam muito bem para microsserviços

BOAS PRÁTICAS

- Automatize tudo o que puder!
- Ferramentas populares de automação de configurações
  - Chef e Puppet
- Utilizar descoberta de serviços (service registry)
- Utilizar balanceadores de carga (load balancers)

CONCLUSÃO

- A arquitetura de microsserviços é melhor que a arquitetura monolítica, mas deve-se levar em consideração:
  - Os desafios em relação à granularidade muito fina das aplicações
  - Utilizar as melhores práticas de testes considerando os componentes dos sistemas de suporte operacionais (OSS – Operations Support System)

REFERÊNCIAS

1. Estilo de arquitetura de micro serviços
2. Introdução aos micro serviços
3. O que são micro serviços
4. Operations Support System
5. Chef x Puppet

INFRAESTRUTURA PARA SISTEMAS DE SOFTWARE

Arquitetura de Microsserviços para o Desenvolvimento de Aplicações