# Introdução

## O que é arquitetura de software?

Antes de revisar sistemas monolíticos e as soluções orientadas a microsserviços, é importante que tenhamos identificado o conceito de arquitetura de software.

## Desafios dos microsserviços

Na aula anterior observamos a diferença entre arquitetura monolítica e microsserviços. Agora, começaremos a ver o tema que será trabalhado ao longo da disciplina: microsserviços. Embora ofereçam muitas vantagens, eles também têm seus próprios desafios, por exemplo:

- É mais complicado de administrar, pois possui mais componentes para gerenciar, monitorar e administrar.
- Dinâmica da Complexidade: sistemas distribuídos são complexos para testar, implementar e entender como um todo. Trata-se de muitos componentes, todos tentando trabalhar juntos para tentar resolver um problema.
- Não caia em um "monólito distribuído", que está muito longe de ser uma solução de microsserviços verdadeiramente dissociada.
- A inexperiência da equipe de desenvolvimento pode dificultar a implementação bem-sucedida de microsserviços. Um exemplo disso pode ser a dificuldade de emular ambientes ao tentar rodar dezenas ou centenas de microsserviços no mesmo ambiente de desenvolvimento local.
- A dificuldade de orquestrar um grande número de microsserviços.
- A dificuldade de gerenciar dependências internas e externas.

Agora é hora de aprender quais são os padrões de design. Não entre em pânico! Existem muitos, mas veremos cada um nas próximas aulas.

O objetivo desta aula é ter um panorama global, onde observaremos os padrões mais importantes que utilizam conceitualmente arquiteturas de microsserviços. Em seguida, em cada aula, entraremos em detalhes de como cada um deles é implementado no Spring Cloud.
Padrões de design são:

- Service registry
- Service discovery
- Edge server
- Central configuration
- Log aggregation
- Distributed tracing
- Circuit Breaker
- Reactive microservices
- Centralized monitoring and alarms

Vamos em frente para conhecê-los.

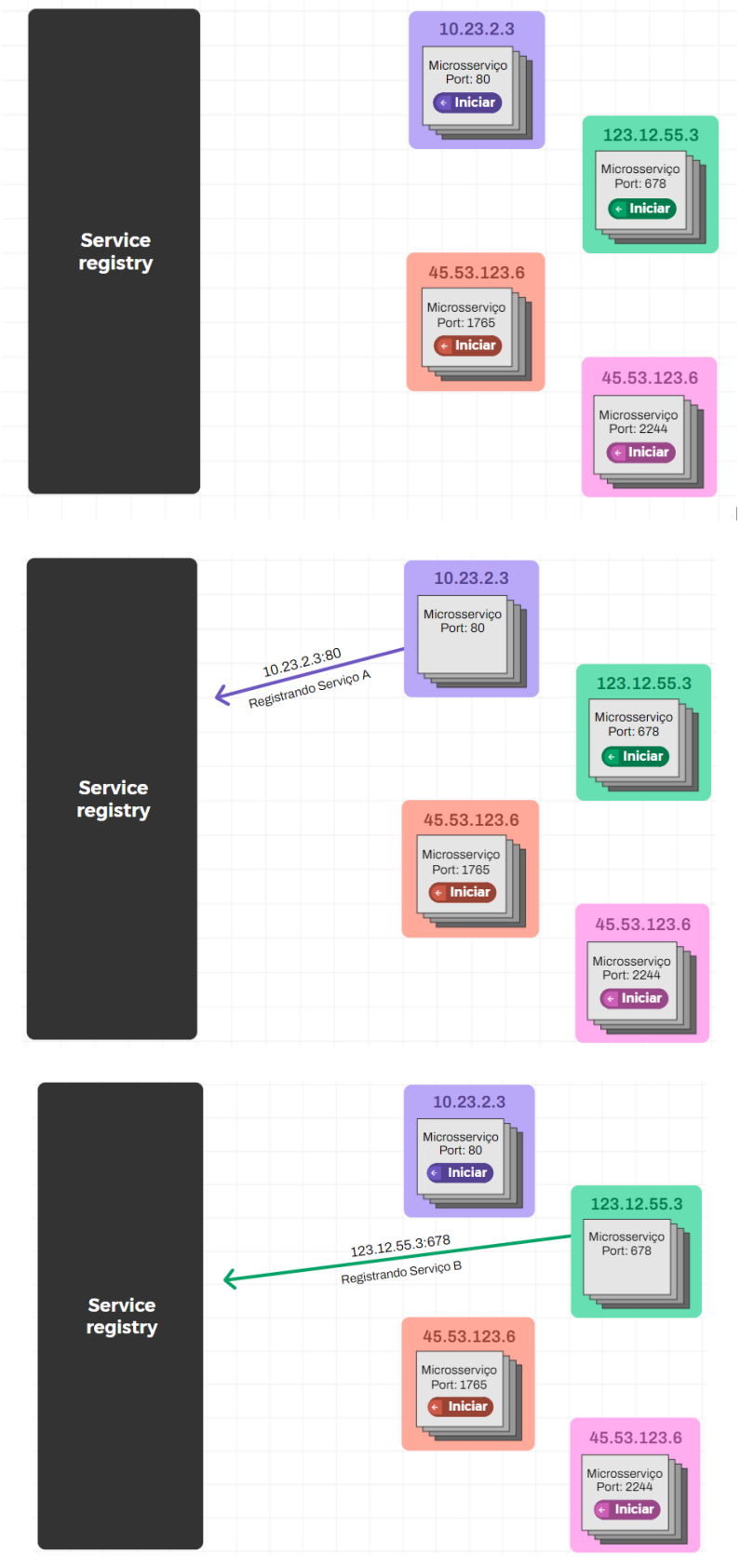
## Padrões de design

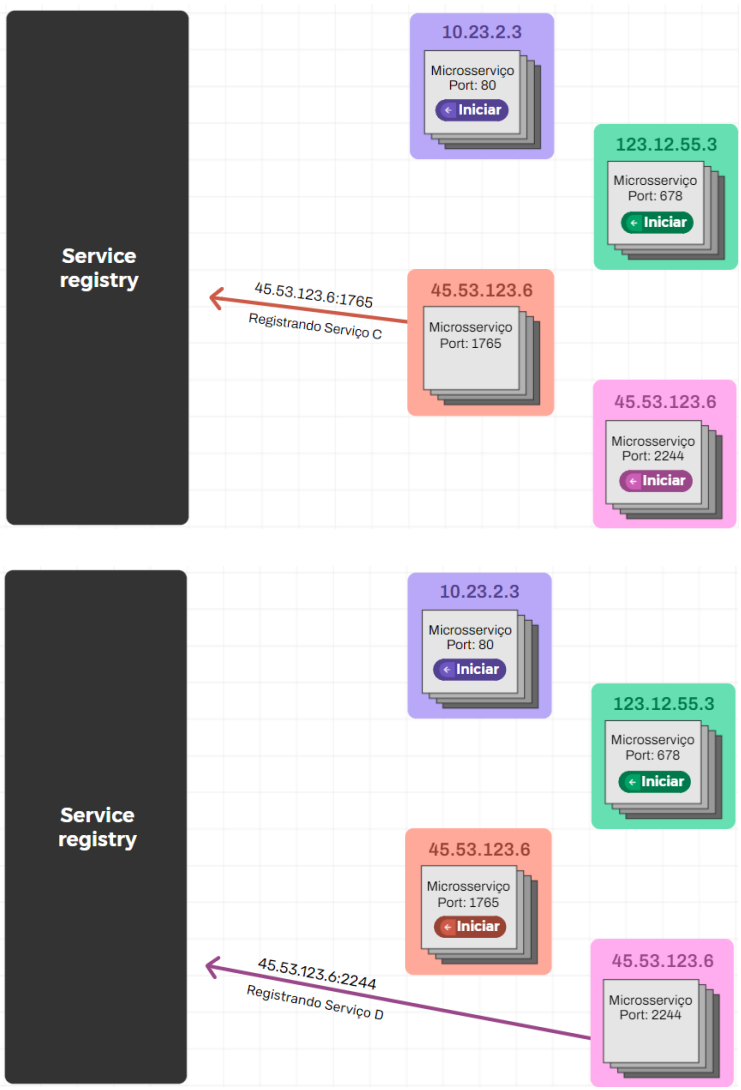
# Service registry e Service discovery

Nesta aula, veremos vários padrões de design que vamos colocar em prática nas diferentes aulas. Todos eles são úteis e importantes, agora é hora de ver suas características começando pelos padrões **Service registry e Service discovery.**

Um dos problemas mais comuns quando trabalhamos com serviços é a necessidade de saber necessariamente onde está localizado cada um. À medida que as empresas crescem, cresce também o número de serviços necessários para dar suporte à operação. Isso torna cada vez mais difícil saber exatamente onde está cada serviço (IP, porta).

O padrão **Service registry** permite a criação de um **servidor centralizado, onde todos os serviços são registrados no momento de iniciar**. Desta forma, cada um dos serviços terá que lhe enviar o endereço IP, a porta onde responde ao servidor e, por fim, o identificador do serviço (que normalmente é um nome alfanumérico que ajuda a identificá-lo). Desta forma, o servidor central ou registro saberá exatamente onde está cada serviço disponível. Vamos ver como isso funciona.


Service registry
10.23.2.3
Microsserviço Port: 80
Iniciar
123.12.55.3
Microsserviço Port: 678
Iniciar
45.53.123.6
Microsserviço Port: 1765
Iniciar
45.53.123.6
Microsserviço Port: 2244
Iniciar
Service registry
10.23.2.3:80
Registrando Serviço A
10.23.2.3
Microsserviço Port: 80
123.12.55.3
Microsserviço Port: 678
Iniciar
45.53.123.6
Microsserviço Port: 1765
Iniciar
45.53.123.6
Microsserviço Port: 2244
Iniciar
Service registry
10.23.2.3
Microsserviço Port: 80
Iniciar
123.12.55.3:678
Registrando Serviço B
123.12.55.3
Microsserviço Port: 678
45.53.123.6
Microsserviço Port: 1765
Iniciar
45.53.123.6
Microsserviço Port: 2244
Iniciar




Como vimos, este padrão propõe **saber exatamente quais os serviços disponíveis e a sua via de acesso**. Normalmente, fornece uma interface gráfica que nos permite ver todo o universo de serviços que estão ativos.

Outra característica é que os serviços de registro devem enviar constantemente um sinal para o registro, conhecido como "batidas" (heartbeat em inglês). Isto indica que os serviços seguirão disponíveis com o passar do tempo. Se não receber este sinal, o microsserviço marca-o automaticamente como fora de serviço e pode então enviar uma notificação a um administrador ou DevOps para que seja gerenciado.

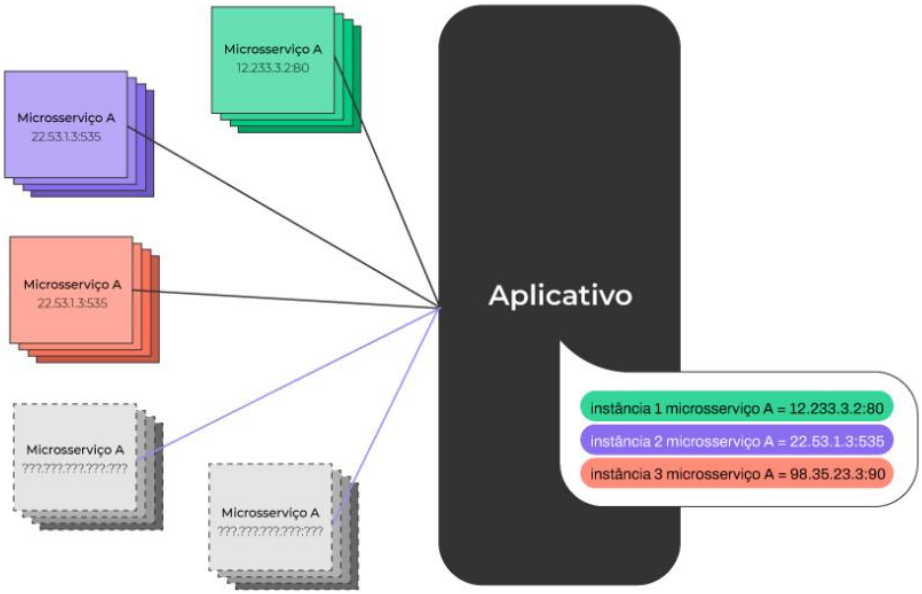

**Podemos concluir mencionando que o Service registry é um componente indispensável em uma arquitetura de microserviços, pois permite que os serviços sejam registrados independentemente de sua localização física. Isto significa que podemos facilmente saber sua localização e, posteriormente, usar técnicas de auto-descoberta para localizá-los e equilibrar a carga.**

# Service discovery

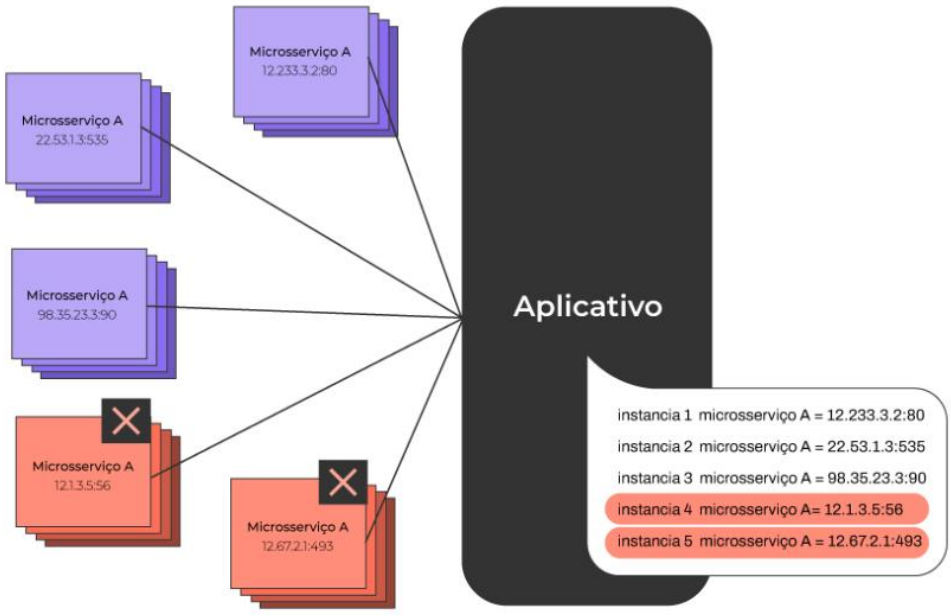
O padrão Service registry é complementar ao padrão Service discovery (veremos o porquê), mas primeiro é necessário esclarecer que em arquiteturas de nuvem elasticidade é a capacidade da nuvem de crescer ou diminuir em recursos à medida que a demanda aumenta ou diminui. Isso significa que a intervenção humana não é necessária e que é possível provisionar nossos servidores ou expandir a capacidade de processamento dos existentes sob demanda.

Portanto, se a nuvem é elástica, não faz sentido ter um arquivo de configuração fixo onde são armazenados os endereços de cada serviço, pois implicaria ter uma pessoa que atualiza continuamente esse arquivo toda vez que uma instância é registrada ou desregistrada.

Na imagem abaixo, podemos ver que as novas instâncias são desconhecidas para a aplicação, a menos que alguém as adicione manualmente ao arquivo de configuração. Isso acaba sendo um pouco **ineficiente e altamente propenso a erros**.

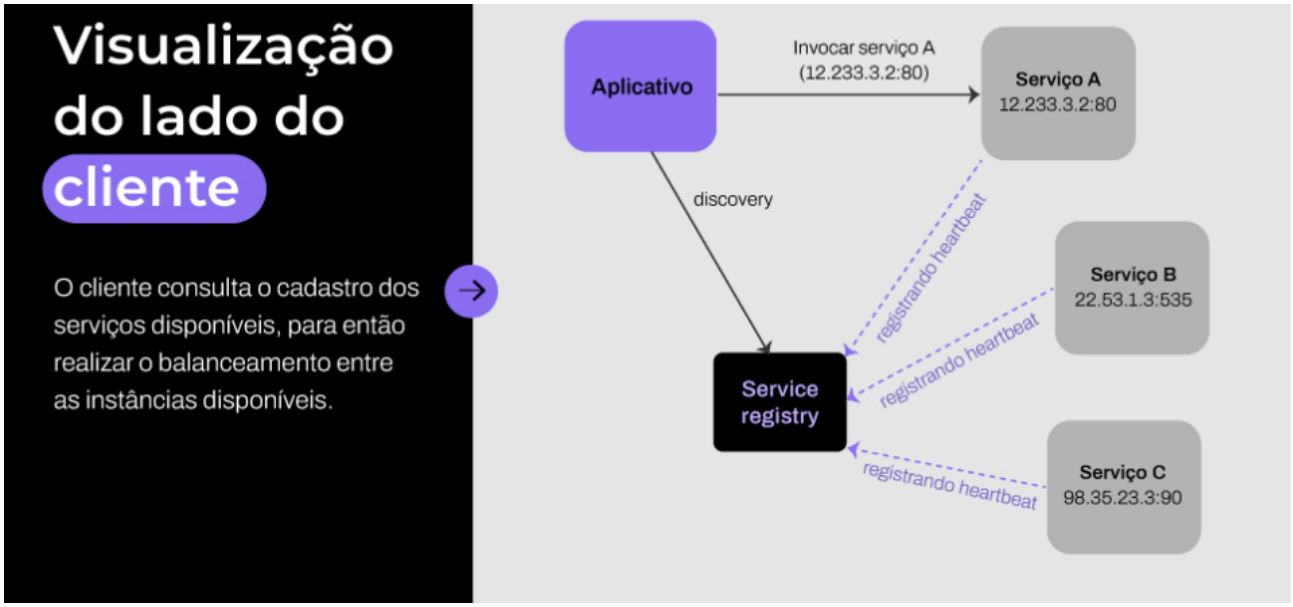


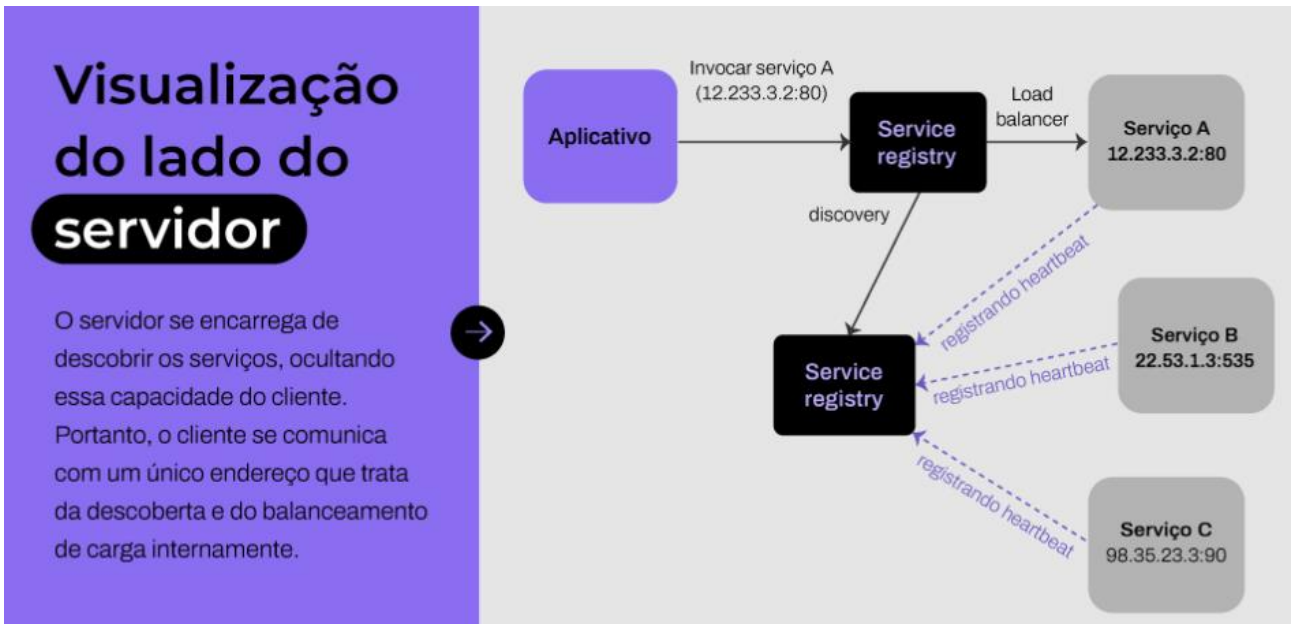

Outra situação é ilustrada na imagem a seguir, onde se pode ver claramente como duas instâncias do serviço deixam de responder (ou simplesmente foram encerradas), resultando na desatualização da aplicação e até mesmo na falta de execução de serviços que não estão disponíveis.




É aqui que aparece a chamada Service discovery, responsável por determinar todas as instâncias ativas dos serviços por um registro central, que nada mais é do que o Service registry que conhecíamos anteriormente. O Service discovery é um componente que se encarrega de recuperar o Service registry de todas as instâncias dos serviços disponíveis e

realizar o balanceamento de carga. No entanto, há duas maneiras pelas quais essa descoberta pode ocorrer: do lado do cliente e do lado do servidor. Vamos em frente para saber a diferença.




Como conclusão, podemos dizer que tal como o Service registry o Service discovery é um dos padrões mais importantes ao implementar microsserviços ou arquiteturas nativas das nuvens. Isto porque nos permite: primeiro, desacoplar de servidores físicos e segundo, escalar rapidamente, simplesmente levantando novas instâncias. É por isso que utilizar uma estratégia de auto-descoberta de serviços nos permite criar aplicações muito mais escaláveis, evitando a dependência de um serviço, IP ou porta específica que nos é atribuída.

Isto permite a implementação de todos os serviços em qualquer lugar e de qualquer forma, graças ao Service discovery que, através do Service registry, possibilita localizar e equilibrar a carga de todas as instâncias disponíveis.