Computação Em Nuvem

Raul Ikeda

Roteiro 4 –Cloud Outtro – 2 Aulas

Sabrina e Leonardo

1. Qual o conceito por trás de Edge Computing? (Obs: Não é a rede de celular 2G)

Atualmente, **o mundo está conectado pela Cloud**, e milhares de dispositivos nos **geram informação** através da internet das Coisas. Tais dados podem ser processados para gerar aprendizado, modelos de negócio, monitoramento etc (um conceito velho conhecido como *data mining*, onde você processa informações para prever e melhorar um serviço).

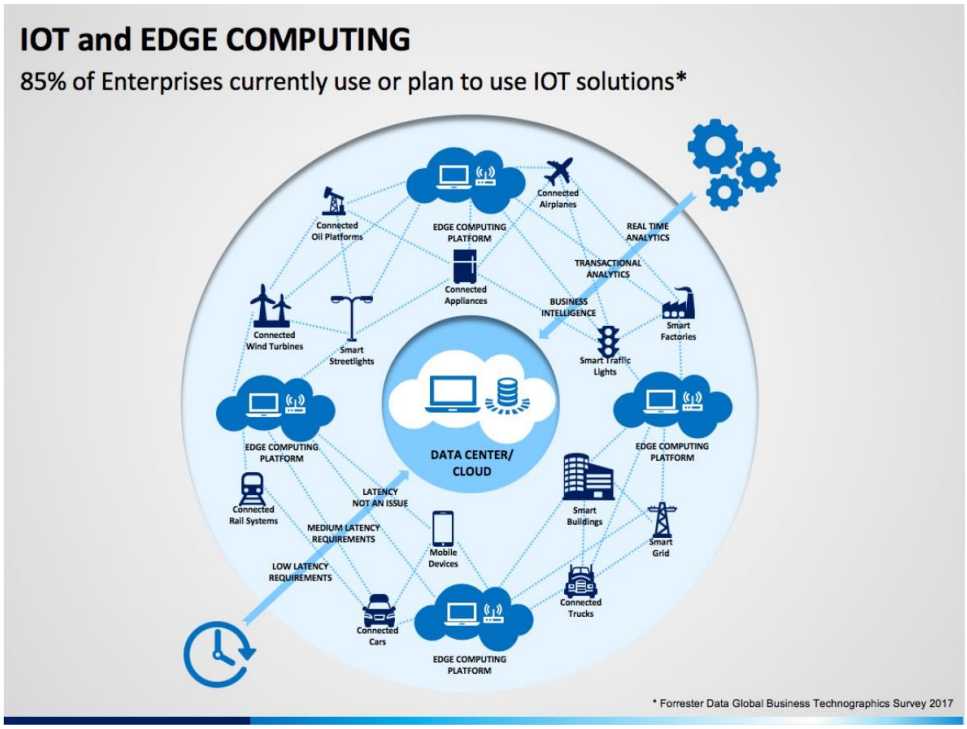
Contudo, o **modelo atual de IoT é centralizado na Nuvem**, onde ela faz sua parte **processando todos os dados fornecidos**. O aumento da quantidade de dispositivos conectados esta conduzindo a **limitações inevitáveis**, e grandes quantidades de dados transmitidos por minuto causam um **pesado processamento na Cloud**, gerando altos custos, um excessivo uso da rede, alto ping, piora na confiabilidade entre outros problemas.

A **escalabilidade limitada** da nuvem e sua arquitetura também impedem a evolução desta tecnologia, pois há um limite físico do tamanho dos Data Centers construídos.

Além disso, **segurança é crucial** no ambiente de dados pessoais e IoT, mas segurança e nuvem pública não são termos que combinam.

No **modo tradicional**, o processamento é centralizado e sensores conectam os aparelhos com um gateway, que encaminha toda a informação para a nuvem processar.

No modelo de **Edge Computing**, teríamos nós **distribuídos** no lugar dos sensores, para fazer o processamento de dados **localmente** e separadamente, para depois fazer a junção. Assim, há um alívio enorme nas requisições feitas ao Data Center, **economia** absurda de rede e de **processamento**. Ganha-se com segurança, já que os dados estão em ambientes diferentes e não compartilham o processamento, além de ter **diminuição de custos** por não precisar construir mais hardware ou gastar com energia elétrica.



2. Você é o CTO (Chief Technology Officer) de uma grande empresa com sede em várias capitais no Brasil e precisa implantar um sistema crítico, de baixo custo e com dados sigilosos para a área operacional. Você escolheria Public Cloud ou Private Cloud?

Optaria por implementar uma Private Cloud na infraestrutura da empresa, pois a segurança dos dados deve ser considerada acima de tudo, e uma cloud pública expõe as informações sigilosas à muito risco, mesmo que uma cloud publica possua camadas de segurança, esta é compartilhada entre diversos clientes, estando visível na rede.

O baixo custo costuma ser raro para implementar uma cloud privada inteira, contudo, **ao longo do tempo o único custo que restará é de manutenção**, então fica a rigor do financeiro dizer se têm como pagar para um time de engenheiros cuidar das máquinas e eventualmente repor uma que falhou. É possível que o custo mensal que seria pago à provedora de serviços cloud se equivala ao custo da manutenção periódica da estrutura.

Além disso, optar por uma private cloud em uma empresa distribuída pelo país é bom, visto que seus dados ficam em território nacional, junto com as sedes. Nenhuma informação precisa ser mantida em território internacional, evitando possíveis burocracias para a equipe de advogados.

3. Agora explique para ao RH por que você precisa de um time de DevOps.

O mais importante que precisa ser enfatizado para o RH é que o serviço usado pela empresa, uma cloud privada, requer muita manutenção frequente e monitoramento. Uma equipe de DevOps cuidaria não só desse monitoramento, como da implantação de aplicações da empresa, da rede a ser usada pela cloud, da segurança, da escalabilidade e da banda que a infraestrutura vai requisitar.

Sem esse time, esse serviço precisaria ter contratação frequente de terceiros para checarem se as máquinas ainda estão rodando bem, o que compromete segurança (pois os funcionários não são da empresa e não tem contratos, seja de sigilo empresarial etc).

4. Junte o seu time e monte um plano para DR e HA da empresa. Explique o que é SLA e onde se encaixa nesse contexto.

O SLA é o contrato que será feito entre o provedor do serviço e o contratante.

Ele normalmente contém cláusulas sobre o que o serviço irá fazer normalmente, sua disponibilidade, portabilidade, permições de acesso entre muitos outros items. Um item importantíssimo em um SLA é a cláusula de Disaster Recover (DR) e uma a respeito de High Availability (HA) que irão ser detalhados a seguir. No SLA, é detalhado e distribuído a responsabilidade de cada parte do contrato, assim em certas situações em que algo inesperado ocorre, fica a par da parte determinada resolver o problema.

Se o time de DevOps, já aprovado e contratado pelo RH, achar que o serviço da empresa precisa ter um *uptime* de 75% do tempo, e pode ficar descansando pelos outros 25%, então será especificado no SLA esse tempo exato, e se acontecer das máquinas terem ficado apenas 60% do tempo online, a culpa irá para o responsável por elas.

A mesma coisa se aplica para um DR, o time deve montar um plano de recuperação, e este pode ser implementado, por exemplo, com um *off-site* e um backup baseado no S3.

O *off-site* consiste em uma imitação da aplicação com o intuito de manter a disponibilidade do serviço, esse site de recuperação de desastres é facilmente acessível pela internet e gerenciado por meio de um portal web, permitindo que os aplicativos essenciais aos negócios estejam funcionando. Embora o off-site possua menor desempenho, quanto a velocidade e capacidade, a disponibilidade é mantida, reduzindo drasticamente as possíveis perdas.

O backup pode ser feito pelo S3 do Amazon, que é um sistema distribuído com baixa probabilidade de falha, com alta velocidade de recuperação, comparado a métodos de backup em fita.

Enquanto o off-site esta rodando pode-se executar a recuperação dos dados de forma rápida e eficiente.

Última questão: dos requisitos de projeto acima, quais são funcionais e quais são não funcionais?

A grande maioria dos requisitos deste projeto são não funcionais, pois o cliente não pediu nada específico ao micro serviço, ou seja, nada funcional para fazermos. De exemplos não funcionais, tem vários:

Migravel, distribuído, elástico, criação e destruição assíncrona, CRUD completo stateless, API...

O talvez único exemplo funcional é fazer o micro serviço ser implantado via um script que o cliente pediu e usará como um produto.