



Cloud Computing

UNIDADE 03

Provisionando na nuvem o provisionamento do papel!

Videoaula: Meuuu chapéu, como provisiono meu projeto?

Então, antes de colocarmos a mão na massa e gastarmos nossos créditos na nuvem (se é que você já não gastou só na curiosidade de como ela funciona... eu sei, já fiz isso), vamos primeiro entender melhor alguns critérios para mapear os recursos necessários de um projeto de uma solução tecnológica, como uma aplicação *web*, no qual você foi alocado.

Aqui será mais uma base importante para muita coisa futura dessa disciplina.

Ao assistir ao vídeo, faça algumas anotações sobre:

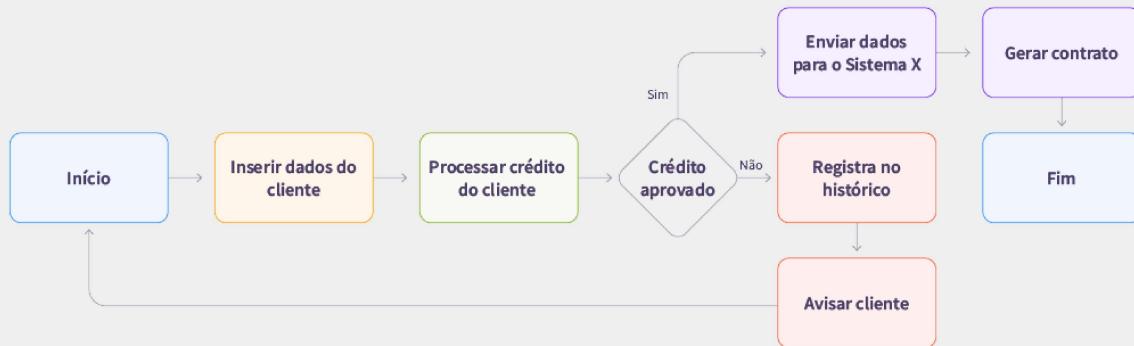
- Quais recursos de nuvem são necessários para provisionar VMs, dependendo do projeto?

Então sem mais delongas, vamos para o vídeo!

Agora que conversamos um pouco sobre projeto de nuvem mais assertivo, ficaram muitas perguntas que precisam ser respondidas. Mas, vamos colocar algumas delas em prática agora, para vermos nosso projeto sair do papel e tomar vida na nuvem. Chega mais!

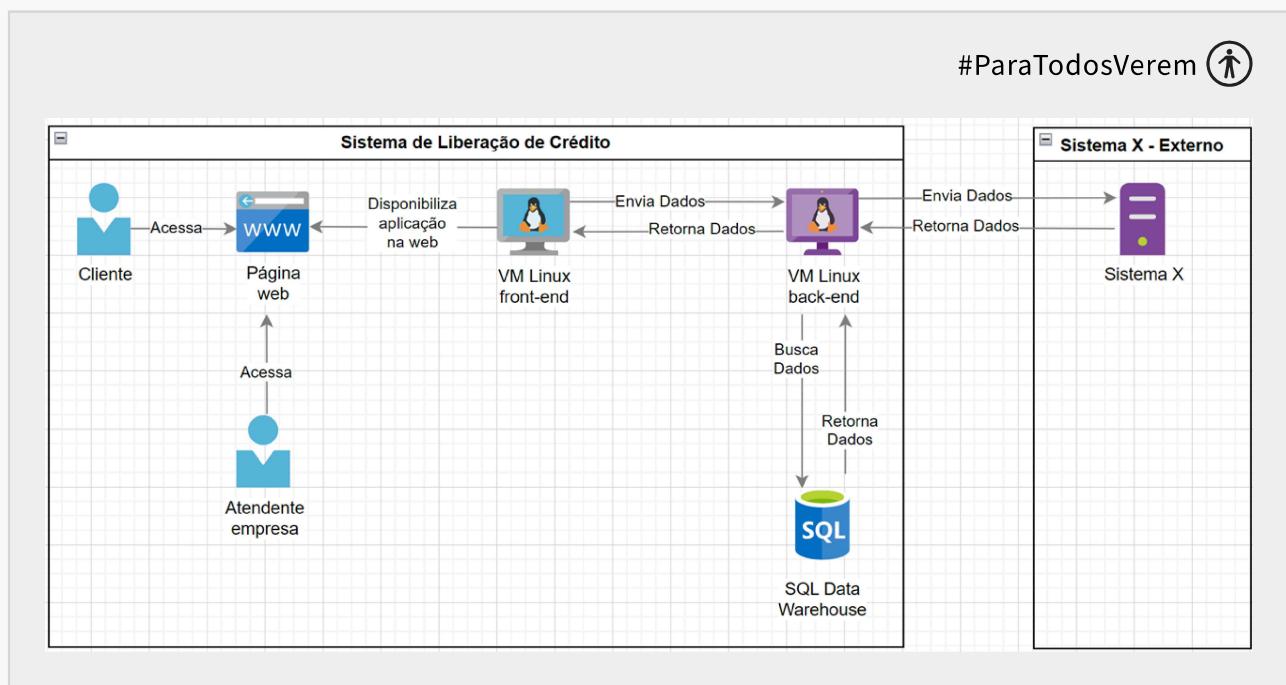
Lembra do fluxo principal do projeto da aplicação *web* para liberação de crédito aos clientes que analisamos na videoaula? A partir de agora, ele servirá como referência para o desenvolvimento de um projeto e será a base das nossas explicações durante as aulas. Caso não se lembre, confira novamente o fluxo a seguir.

Fluxo macro de sistema de liberação de crédito



O autor

Então, pensando em cenário básico de arquitetura, teremos o seguinte desenho.



O autor

A partir desse contexto, logo de início, percebemos a necessidade de provisionar três VMs para a aplicação: uma para o *front-end*, outra para o *back-end* e mais uma para o banco de dados. Agora, se você for criar uma VM no Microsoft Azure (o que também vale para outros provedores de nuvem, como Amazon AWS e Google Cloud), vai se deparar com a seguinte situação:

Página inicial > Criar um recurso >

Criar uma máquina virtual

Ajude-me a criar uma VM de baixo custo Ajude-me a criar uma VM otimizada para alta disponibilidade Ajude-me a criar uma VM para aprendizagem

Detalhes da instância

Nome da máquina virtual * ⓘ 

Região * ⓘ  

Opções de disponibilidade ⓘ 

Opções de zona ⓘ Zona auto-selecionada
Escolha até 3 zonas de disponibilidade, uma VM por zona
 Zona selecionada pelo Azure (versão prévia)
Deixe o Azure atribuir a melhor zona para suas necessidades

Zona de disponibilidade ⓘ 

Tipo de segurança ⓘ 
[Configurar os recursos de segurança](#)
 A máquina virtual de inicialização confiável é necessária ao usar imagens da Galeria 1P.

Imagen * ⓘ  
[Ver todas as imagens](#) | [Configurar a geração de VM](#)

Arquitetura de VM ⓘ  Arm64
 x64

O autor

Agora vamos ver o que as escolhas desses elementos representam durante o projeto.



DICA

Neste momento, não vamos criar a máquina virtual, mas se quiser acessar a página de configurações que mostramos na imagem, pode acessar o Azure no *link* a seguir.

<https://azure.microsoft.com/pt-br/free/students>

Para autenticar a entrada, utilize *login* e senha criados pela PUCPR para você. Por exemplo, xxx@pucpr.edu.br.

No primeiro acesso, você precisa **configurar algumas informações**. Na tela, logo após a autenticação, precisa é necessário digitar seus dados pessoais (Nome e sobrenome), país e data de nascimento. Também é necessário escolher, em **School Name**, a Pontifícia Universidade Católica do Paraná.

Na segunda tela, **nas configurações do perfil**, algumas informações já aparecem preenchidas, no entanto, é necessário digitar o seu *e-mail* criado pela PUCPR, o telefone, o CNPJ da PUCPR, que é: 76.659.820/0003-13 e o endereço – Rua Imaculada Conceição, 1155 - Prado Velho, Curitiba-PR, 80215-901. Se aparecer a confirmação de endereço em outra tela, pode selecionar o que for sugerido pelo *site*.

Após confirmar as informações, a tela do Azure se abrirá e, a partir daí, você pode explorar a criação de máquinas virtuais.

Você pode estar se perguntando: **o que eu selecionei de Região, Imagem e Arquitetura de VM?**

Esses itens são bem importantes durante o planejamento do seu projeto.

A região é um dos pontos mais críticos, porque incide diretamente no custo e na latência da sua aplicação. Imagine que você cria a VM do *front-end* na Região (US) West US 2 e para a VM do *back-end* você seleciona (*Asia Pacific*) *Australia East*. Isso significa que a comunicação entre as duas VMs precisará percorrer uma grande distância, praticamente atravessando o globo, o que resultará em um tempo de resposta bem maior. Agora, se ambas as VMs estivessem na mesma região, como (US) West US 2, essa latência seria significativamente reduzida.

E não para por ai!

Imagine que o público do nosso sistema de liberação de crédito seja 100% brasileiro. Por que, então, eu colocaria todas as minhas VMs na região (*Asia Pacific*) *Australia East*? Isso faria com que, sempre que alguém acessasse a aplicação do Brasil, os dados precisassem viajar pelo mundo todo para retornar uma resposta. Faz sentido?

Mas, você pode estar pensando: - “**Professor, minha aplicação é supersimples e tem pouco acesso, somente para algumas pessoas daqui da empresa mesmo. Coloquei na Austrália e está tudo OK.**”

Sim, isso pode acontecer! Aqui entra o primeiro ponto sobre a carga de trabalho da aplicação. Se for um sistema simples, com poucos acessos — digamos que cerca de 20 pessoas da sua empresa o utilizem ao longo do dia —, provavelmente você não perceberá grandes diferenças de desempenho, **desde que tudo esteja na mesma região**.

Agora, mesmo em um cenário de baixo uso, se a sua VM da aplicação estiver em uma região e o banco de dados em outra, a latência pode causar lentidão na aplicação. Ou seja, a distância entre os servidores importa, independentemente do volume de acessos.

Imagine que sua aplicação **não seja usada apenas por 20 pessoas da sua empresa**, mas sim por **milhares de usuários simultaneamente**, como em um site de compras online aqui no Brasil. **Com esse grande volume de acessos, a escolha da região dos servidores faz toda a diferença**. Se a aplicação estiver hospedada em uma região distante, a latência será ainda mais perceptível, impactando diretamente a experiência dos usuários.

Um adendo importante aqui: **não é somente a região que pode estar causando lentidão** na sua aplicação, inúmeros fatores podem ajudar nisso, mas aqui estou focando apenas no quesito da região, OK? Na sequência, vamos avaliar os outros fatores.



DICA

Para boa parte dos trabalhos nesta disciplina, usaremos alguns recursos que oferecem um tempo gratuito de uso. Por isso, fique atento ao escolher o sistema operacional, seja Linux ou Windows, e verifique se ele está identificado como **(serviços gratuitos qualificados)**.

Além disso, a escolha do tamanho do servidor também é importante. Priorize opções da família **Standard_B1s** ou **Standard_B2ats_V2**. Se essas configurações não aparecerem de imediato na tela inicial, basta clicar em "**Ver todos os tamanhos**" e procurar pelas séries **B** ou **B v2**.

Observe na imagem onde esses recursos estão disponíveis na tela.



Criar uma máquina virtual ...

Ajude-me a criar uma VM de baixo custo Ajude-me a criar uma VM otimizada para alta disponibilidade Ajude-me a...

! A máquina virtual de inicialização confiável é necessária ao usar imagens da Galeria 1P.

Imagen *

Ubuntu Server 24.04 LTS - x64 Gen2 (serviços gratuitos qualificados)

[Ver todas as imagens](#) | [Configurar a geração de VM](#)



Arquitetura de VM

Arm64

x64

Executar com desconto de Spot do Azure

(1)

! Você está no período de avaliação gratuita. Os custos associados a essa VM podem ser cobertos por créditos restantes em sua assinatura. [Saiba mais](#)



Tamanho *

Standard_B1s - 1 vcpu, 1 GiB memória (US\$ 7,59/mês) (serviços gratuitos qu...)

[Ver todos os tamanhos](#)



O autor

Você pode estar pensando que é mais fácil colocar tudo na região do Brasil, correto?

Confira as imagens a seguir e analise o que achou de diferente?

Região (US) West US 2:

#ParaTodosVerem



Tamanho *

Standard_B1s - 1 vcpu, 1 GiB memória (US\$ 7,59/mês) (serviços gratuitos...)

[Ver todos os tamanhos](#)

O autor

Região (South America) Brazil South:

#ParaTodosVerem



Tamanho *

Standard_B1s - 1 vcpu, 1 GiB memória (US\$ 12,26/mês) (serviços gratuitos q...)

[Ver todos os tamanhos](#)

O autor

Apesar do serviço dessa série B1s ser gratuito, dentro de alguns critérios, o valor na região Brasil é quase o dobro da região dos Estados Unidos. Isso devido aos nossos custos com impostos. A latência, sim, é melhor no Brasil, caso sua aplicação seja exclusiva para cá, mas o valor é bem maior. Esse é só um exemplo de um custo de um servidor simples, dependendo da sua aplicação, podemos chegar a muitos dólares por mês, o que dá uma imensa diferença no orçamento.

“Mas, então, professor, nunca vou usar a região do Brasil?”

Há casos que, sim, se você precisar da menor latência possível, terá que pagar a mais. Um exemplo disso acontece nos sistemas industriais. Se a empresa possui um equipamento industrial com um *software* local dentro da sua rede e precisa integrar uma aplicação para executar processos em tempo real, a melhor escolha é definir a região Brasil. Caso contrário, além do risco de falhas na aplicação, você também pode enfrentar lentidão devido à latência de outras regiões. Então, note que, a escolha desse simples atributo, influencia muito a sua arquitetura. Escolha com sabedoria!



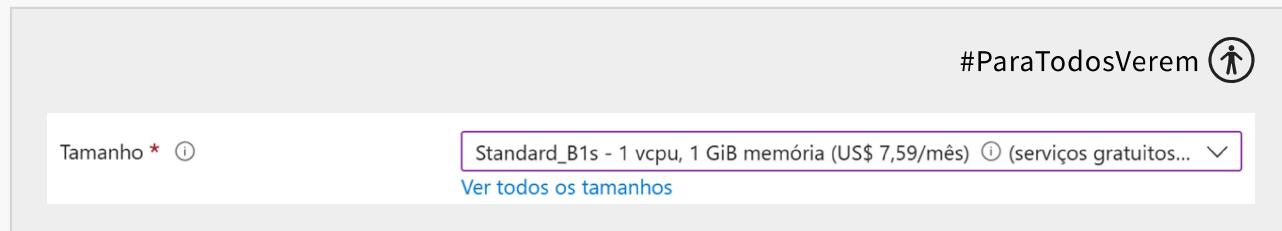
DICA

Sim, é possível mover um recurso de nuvem de uma região para outra, mas não é somente editar recurso e escolher uma nova região. Você precisa usar alguns recursos específicos, como no caso da Azure, o Resource Mover.

Dependendo do tamanho e da criticidade da sua aplicação, pode fazer sentido distribuí-la em várias regiões para garantir redundância. Nesse caso, você pode conectar esses ambientes a recursos específicos que identifiquem de onde vem a requisição do usuário. Por exemplo, se for do Brasil, ela é direcionada para a aplicação na região do Brasil; se for dos Estados Unidos, vai para a aplicação na região dos EUA. Mas claro, isso terá um custo, além de exigir conhecimento avançado em DevOps na nuvem, que não é nosso foco aqui, mas fica a curiosidade.

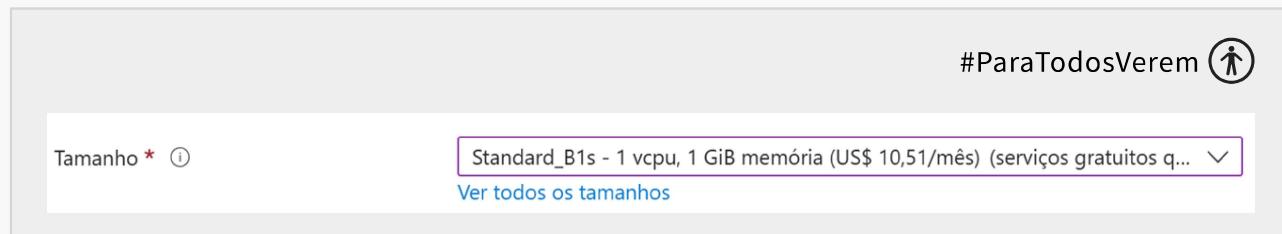
Agora que já falamos sobre a região, vamos conversar sobre **imagem**. Aqui é bem simples, você escolherá qual o sistema operacional mais adequado para o que precisa e a versão dele. Para isso, você precisa entender bem sobre a aplicação com seu time de desenvolvedores. Mas obviamente tem pegadinha aqui. Dependendo do sistema operacional que escolher, como no caso o Windows, terá **custo extra de licenciamento**. Note, a seguir, nas imagens, a diferença de preço, mesmo escolhendo imagens que possuam **serviços gratuitos qualificados**.

Imagen Linux Ubuntu Server



O autor

Imagen Windows Server 2022



O autor

Para finalizar essa primeira parte da aula sobre a **arquitetura de VM**, temos a **Arm64**, que vem na pegada de recursos de baixo consumo energético, igual de celulares e alguns notebooks atuais e a X64, que é a arquitetura padrão.

Não são todos os sistemas operacionais que **suportam a arquitetura Arm64**, então se selecionar essa arquitetura, note que seu sistema operacional pode mudar automaticamente para a escolha de outras versões.

E, sim, a arquitetura Arm64 vem também com a ideia de redução de custos na nuvem em relação à arquitetura X64. Mas aqui vai um adendo importante: a sua aplicação precisa suportar essa arquitetura!

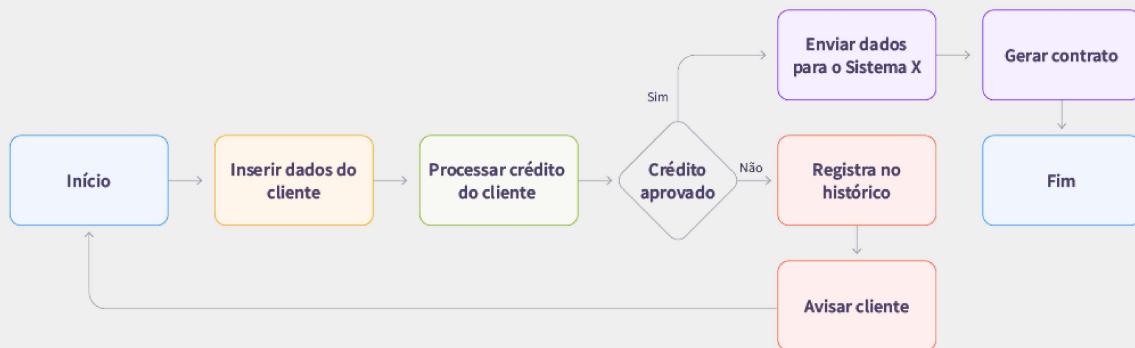
É como comprar um notebook com processador ARM, colocar seu jogo predileto para rodar pela Steam, e ele não abrir. Isso porque esse jogo não está preparado para essa arquitetura, que é comumente vista em *smartphones*. Então fique atento a isso!

Beleza, até aqui falamos de maneira geral sobre algumas opções e impactos para configurar as VMs, mas ainda não fizemos uma avaliação real do que precisaríamos para fazer nosso sistema (Aplicação web para liberação de crédito) rodar de fato.

Apresentamos novamente nosso fluxo e a arquitetura do projeto para relembrarmos as informações essenciais. Observe.

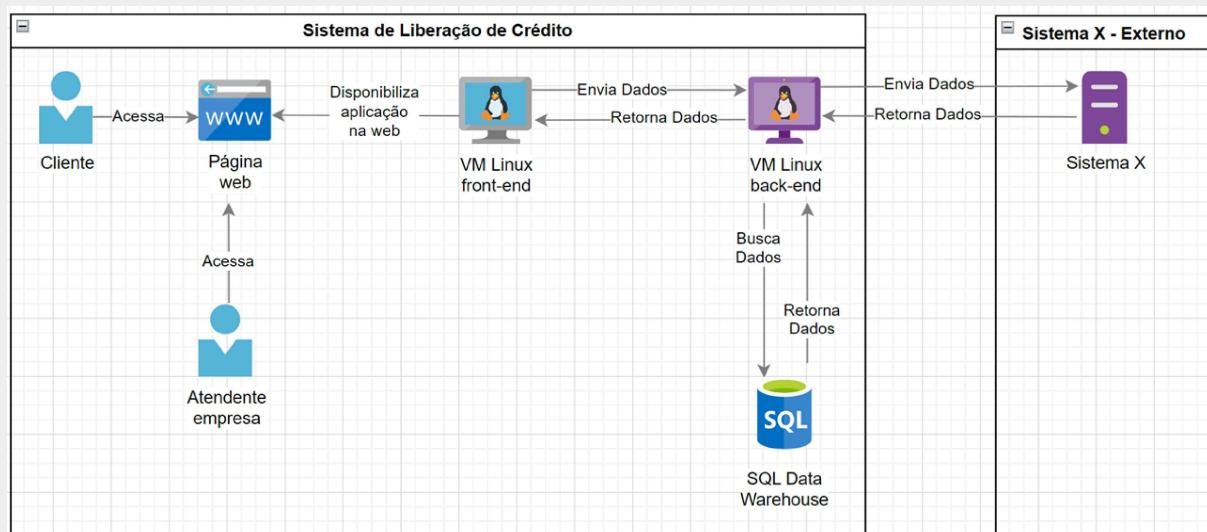
#ParaTodosVerem 

Fluxo macro de sistema de liberação de crédito



O autor

#ParaTodosVerem 



O autor

Ainda temos muitas perguntas sobre nosso projeto de liberação de crédito precisam ser respondidas, tais como:

- Quantos usuários simultâneos diários, em média, acessarão a aplicação?
- Qual o pico de usuários simultâneos diários?
- Quanto tempo cada etapa do nosso fluxo da aplicação levará? Pensando, por exemplo, em um cenário feliz, o atendente preencheu os dados, validou o crédito e entregou o contrato. Caso seja aprovado, quanto tempo, mais ou menos, isso demorará para ser concluído?
- Considerando os tempos da pergunta anterior, o que será processado por *front-end*, *back-end* ou banco de dados? Às vezes, o maior processamento ocorre no banco de dados. Nesse caso, suas VMs de *front-end* e *back-end* podem ser mais simples, enquanto a VM do banco de dados precisa ser mais robusta. As VMs não precisam ser iguais.
- O sistema é crítico e precisa funcionar 24 horas por dia? É estratégico para a empresa ou ele só funciona durante o horário comercial? Porque, dependendo dessas respostas, teremos que pensar em uma estrutura de redundância, além de um plano de *disaster recovery* (que veremos mais para frente na disciplina).
- Quais funcionalidades da aplicação serão síncronas e assíncronas? Por exemplo, após o cadastro do cliente, o processo de avaliação de crédito e a integração com o Sistema X serão iniciados. Enquanto isso, você pode mostrar na tela a mensagem: "Agora vamos processar o seu crédito. Avisaremos assim que for concluído. Obrigado!". Isso significa que a análise de crédito será feita de forma assíncrona, evitando que o usuário fique esperando. Quando estiver pronta, você avisará o usuário para ele retornar à tela.

Essa abordagem pode mudar sua arquitetura e os recursos necessários. Imagine que, nesse caso, o processamento pesado da aplicação ocorre na análise de crédito. Dependendo da estratégia, você poderia ter uma segunda VM de *back-end* dedicada a essa tarefa, enquanto a primeira VM de *back-end* funcionaria como um orquestrador de requisições. Assim, quando uma requisição que precisa de análise é recebida, ela é enviada para a segunda VM de *back-end*.

Ainda nesse cenário, você poderia ter essa segunda VM de *back-end* totalmente isolada. A primeira VM de *back-end* gravaria no banco de dados quando uma nova requisição de análise de crédito chegasse. A segunda VM de *back-end*, por sua vez, lerá periodicamente o banco de dados para verificar se há novas requisições. Se houver, ela processa e grava o resultado no banco novamente, que será posteriormente validado pela primeira VM de *back-end*. **Existem inúmeras possibilidades que podem influenciar sua arquitetura e os recursos necessários na nuvem.**

Ah, isso que falamos de assíncrono, acima, entra também no conceito de mensageria, que não vamos abordar aqui, porém é um recurso que está na nuvem também.

- Será que terá alguma aplicação de *big data*, que fará algum processo de *business intelligence* ou BI (não vamos entrar nesses conceitos aqui, mas, é importante que você saiba sobre isso para provisionar as VMs), **como uma consulta no nosso banco de dados para identificar padrões de fraude para alguma outra área da empresa?**

Se isso acontecer, pode afetar a performance da aplicação. Dependendo da quantidade de dados e do horário em que essas consultas forem realizadas, pode ser necessário aumentar os recursos do banco de dados para evitar que a aplicação caia enquanto os clientes e atendentes da empresa estão utilizando o sistema. Imagine se a aplicação cai bem na hora de fechar alguns contratos? O prejuízo para a empresa poderia ser grande, tanto financeiramente quanto em termos de imagem.

- Será que, no futuro, além da integração com o Sistema X, **algum outro sistema precisará consultar dados da minha aplicação?** Um exemplo disso seria criar APIs para essas integrações e obviamente, isso consumiria bastante recursos das VMs, então, teríamos que deixá-las mais parrudas.
- Será que, **apenas uma VM de front-end será necessária para aguentar os milhares de clientes que vão acessar a aplicação?**
- Será que não é necessário ter cópias da VM de *front-end* e colocar um serviço de **balanceador de carga na frente?** Nesse caso, em vez de o usuário acessar a aplicação diretamente, ao digitar a URL da aplicação no navegador, a requisição cai no balanceador de carga. Ele identifica qual VM de *front-end* está com menos uso e envia o usuário para lá.

Isso distribui a carga de trabalho em várias VMs de *front-end*, garantindo **Escalabilidade**, ou seja, se a base de usuários começar a crescer, é possível simplesmente fazer quantas cópias da minha VM de *front-end* e colocar no balanceador de carga. Isso pode ser feito também nas VMs de *back-end* e até banco de dados.

Inclusive, existem mecanismos de nuvem que fazem isso automaticamente, nos quais você configura regras do tipo: se todas as VMs estiverem com o processador ou a memória acima de 70% ou qualquer outro recurso, automaticamente, crie mais uma VM. Caso todas as VMs estejam com processador ou memória em menos de 30%, desative uma ou mais VMs, para reduzir custos. **Apenas um adendo ao banco de dados: escalabilidade de banco é um pouco mais complicada, porque além da VM você precisa espelhar os dados entre as VMs em tempo real.** Existem mecanismos para isso também, entretanto, seu custo é alto. A solução não depende apenas da sua infraestrutura de nuvem, a sua aplicação também precisa ser desenvolvida de forma escalável, então, é um conjunto dos dois itens, aplicação e infraestrutura.

- **Será que eu precisarei de algum mecanismo de *cache* para otimizar meu banco de dados ou minhas páginas *web*, por exemplo?** Dependendo da resposta, existem mecanismos de *cache* na nuvem que pode ser utilizado e, infelizmente, esse recurso tem custos altos, porque alguns modelos de *cache* utilizam memória de rápida *performance* para não haver um gargalo entre os recursos que estão usando o *cache*.
- **Por ser uma aplicação *web*, preciso ter algum mecanismo de segurança inicial, para evitar ataques cibernéticos?** Essa é uma abordagem bastante comum para aplicações que são expostas para o público externo, colocando mecanismos de WAF (*web application firewall*) na frente da aplicação. Então, o usuário, ao acessar a URL do site, cairia primeiramente no WAF e depois, se estiver tudo certo, o mandaria para a aplicação em si.
- Tem alguma funcionalidade que precisa ser **resiliente**? Ou seja, se bem no meio da execução, der um erro e cair a aplicação, quando ela retornar, precisa voltar de onde sua execução parou, ou posso dar uma mensagem para o usuário dizendo “deu erro, tente novamente!”. Apesar de uma mensagem chata, às vezes a estratégia é essa, porque resiliência requer uma boa programação da aplicação, e isso pode exigir recursos diferentes na arquitetura do projeto.
- **Como meus clientes e atendentes da empresa se autenticarão na aplicação?** Os clientes, por meio de *login* e senha proprietários, ou seja, *login* e senha criados na própria aplicação, e os atendentes por sistema de SSO, ou *single sign-on* da empresa, como o Microsoft Active Directory?
- Depois de autenticados, **como vou controlar as permissões, ou seja, o que cada usuário poderá ver e executar na aplicação?** Será uma funcionalidade interna da própria aplicação, ou usarei algum recurso da nuvem como os recursos de IAM (gerenciamento de identidade e acesso) da Azure, AWS ou qualquer provedor de nuvem? São vários recursos de IAM que podem ser implementados e será necessário integrá-los a sua arquitetura.

“Nossa, professor, quantas perguntas relevantes para sabermos o que preciso para provisionar na nuvem! Nem sei ainda como calcular se eu preciso de uma VM com 1 ou 200 GB de memória....”

Calma, bora fazer uma pausa, porque é muita informação para se pensar. Tome uma água, um café, um chá ou um suco e, depois, continuamos!



Canva

As perguntas que apresentamos aqui são importantes para a resolução das atividades da disciplina.

Deixamos a seguir, um infográfico com as perguntas que apresentamos, caso queira consultá-las posteriormente.





CURIOSIDADE

Você sabia que cada recurso de nuvem pode ser cobrado de acordo com seu tamanho e características? Por exemplo, processador, memória, armazenamento, velocidade, *backup*, quantidade de tráfego de rede, número de requisições e outros itens específicos. Se você não for usar os recursos por um certo período, é **melhor desligá-los (não os deletar) para evitar custos desnecessários**. Sim, isso é um fato! E só delete um recurso se nunca mais for utilizá-lo!

Você sabia que o famoso *resource group*, ou **grupo de recursos**, permite criar vários grupos para separar seus recursos de nuvem? Ao deletar um grupo de recursos, todos os recursos dentro dele também serão deletados! Imagine que sua aplicação e todos os seus *backups* estão em um mesmo grupo de recursos. Se você deletar esse grupo, a aplicação e todos os seus *backups* serão deletados também. Ou seja, adeus à aplicação e aos dados. **Então, cuidado!**

The screenshot shows a user interface for creating a new resource group in the Azure portal. On the left, there are fields for 'Assinatura *' (Subscription) and 'Grupo de recursos *'. A red arrow points from the 'Grupo de recursos' field to a dropdown menu on the right. The dropdown menu lists 'Azure subscription 1' and 'Cloud_Computing_PUCPR', with 'Cloud_Computing_PUCPR' highlighted. Below the dropdown is a blue button labeled 'Criar novo' (Create new).

O autor

E aí, tomou um café ou um chá? Comeu alguma coisa?

Então, vamos finalizar nossa troca de conhecimentos aqui.

Como mencionamos na videoaula, depois de responder a muitas perguntas necessárias, você consegue mapear melhor a arquitetura dos recursos necessários. **No entanto, determinar as características de cada recurso, como a quantidade de memória necessária em uma VM, é a parte mais complicada**, especialmente se seu projeto não tem nenhum outro na empresa para servir de referência. Quando temos

outros projetos, já em operação na empresa, podemos usá-los como base para escolher os recursos. Isso torna a tomada de decisão mais simples. Mas quando não, talvez por ser o primeiro projeto de nuvem da empresa, ou é um projeto totalmente inovador, há algumas abordagens que podemos seguir.

Recursos mínimos

Criar as VMs com os recursos mínimos que a empresa tenha por padrão e efetuar um projeto-piloto interno, para medir o comportamento dos recursos de nuvem.

Essa é uma abordagem bastante utilizada, pois ela desmistifica qualquer situação que sua aplicação possa enfrentar durante o uso. Obviamente que um projeto-piloto não vai contemplar o uso real da aplicação, mas você já terá uma base de como será a experiência e isso ajuda muito.

Processamento

Sabendo que, em nosso projeto, **teremos uma funcionalidade que realizará cálculos complexos com muitas informações para gerar um resultado**, é natural entender que cálculos exigem processamento. Portanto, você não vai alocar uma VM com apenas 1 vCPU. Em vez disso, escolherá uma VM maior e, possivelmente, de uma série diferente, pois cada série representa um tipo de recurso específico.

Vamos ver um exemplo.

Note, na imagem a seguir, que a série B2as_v2 é de Uso Geral, com 2 vCPUs e 8 GB de RAM, custando US\$ 54,90 por mês. Já a série F2s_v2, é para computação de processamento otimizado, ou seja, mesmo ele tendo os mesmos 2 vCPUs e menos memória RAM, o processador dessa série é mais parrudo que o da série B e isso reflete no custo.

Tamanho da VM ↑↓	Tipo ↑↓	vCPUs ↑↓	RAM (GiB) ↑↓	Discos de dados ↑↓	IOPS Máx. ↑↓	Armazenamento ... ↑↓	Disco Premium ↑↓	Custo/mês ↑↓
D2as_v5 ↗	Uso geral	2	8	4	3750	N/D	Com suporte	US\$ 62,78
D2ads_v5 ↗	Uso geral	2	8	4	3750	75 (SCSI)	Com suporte	US\$ 75,19
D2s_v5 ↗	Uso geral	2	8	4	3750	N/D	Com suporte	US\$ 70,08
B2as_v2 ↗	Uso geral	2	8	4	3750	N/D	Com suporte	US\$ 54,90
→ Série D v5								
Os tamanhos da família D de 5ª geração recomendados para suas necessidades de uso geral								
→ Série D v4								
Os tamanhos da família D de 4ª geração para suas necessidades de uso geral								
→ Série B v2								
Ideal para cargas de trabalho que não precisam de desempenho total da CPU continuamente								
→ Série B								
Ideal para cargas de trabalho que não precisam de desempenho total da CPU continuamente								
→ Série DC								
Projeto para proteger a confidencialidade e a integridade de código e dados para cargas de trabalho de uso geral								
→ Série E v5								
Os tamanhos da família E de 5ª geração para suas necessidades de alta memória								
→ Série E v4								
A 4ª geração de tamanhos da família E para suas altas necessidades de memória								
→ Série F v2								
Aumento de até duas vezes do desempenho para cargas de trabalho de processamento de vetor								
→ F2s_v2	Computação otimizada	2	4	4	3200	16 (SCSI)	Com suporte	US\$ 61,76

O autor

Isso vale para outros cenários também, por exemplo, se a funcionalidade utilizar muita memória RAM, fazendo cálculos com muitos dados armazenados. Nesse caso, você pode escolher um tamanho com maior memória RAM e até memória otimizada, que garante mais rapidez.



IMPORTANTE

A coluna com IOPS, que significa, em português, a quantidade de leitura e gravação de dados por segundo que um recurso consegue operar. Via de regra, quanto maior, melhor, mais capacidade teremos por segundo, mas obviamente isso implica mais custo.

Porém, é uma métrica importante. Se a nossa funcionalidade for leve e rápida, mas tivermos bilhões de requisições por segundo, talvez seja necessário escolher uma VM com um IOPS maior para suportar essa carga, em vez de focar apenas memória ou processador.

É tudo custo na nuvem. Quer mais? Paga mais! Esqueceu o recurso ligado sem usar, paga também! Se acostume com isso. Mas, fique tranquilo, falaremos sobre custos com mais detalhes mais adiante na disciplina!



SAIBA MAIS

Alias, quer saber mais um pouco sobre essas séries do Azure, dê uma olhada no *site oficial da Microsoft Azure*. Você também pode procurá-las de qualquer provedor de nuvem, entretanto, o nome série pode não ser o mesmo. Na Amazon AWS que o correspondente a ele é instância.

Escolha o tamanho certo de VM (máquina virtual) para seu cluster do Azure HDInsight. Disponível em: <https://learn.microsoft.com/pt-br/azure/hdinsight/hdinsight-selecting-vm-size>

Tipos de instâncias do Amazon EC2. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types/>

Consultoria

Dependendo do porte da sua empresa, você pode contar com serviços de consultoria do próprio provedor de nuvem, que fará um *sizing* da sua arquitetura e sugerirá os melhores tamanhos.

Se a aplicação for de um fornecedor e você só precisar hospedá-la no seu provedor de nuvem, peça ao fornecedor a especificação (*sizing*). Ele tem a obrigação de saber qual a configuração adequada para o seu cenário atual, incluindo um cenário futuro de crescimento.



DICA

E aqui vai uma dica de ouro! Mesmo que você se torne um especialista em *cloud computing*, não faça o *sizing* sozinho sem a aprovação do fornecedor. Mesmo que você tenha certeza de que a aplicação rodará tranquilamente, isso pode infringir alguma cláusula contratual ou até mesmo afetar o nível de suporte. Se ocorrer algum problema com a aplicação, o fornecedor pode negar o suporte, alegando que a infraestrutura utilizada não é a

combinada e adequada. Se isso acontecer, você terá um grande problema nas mãos para resolver e reportar à sua gerência e aos clientes. Fique atento a isso!

Mas você pode estar pensando: “**Professor, precisamos cortar custos de cloud na empresa...**”

Mesmo que esse seja o cenário, a empresa aceitou os termos de contratação desse *software* e precisa prever os recursos financeiros necessários para mantê-lo. "O que você pode fazer é um acordo com o fornecedor, no qual, demonstrando o cenário atual da aplicação (que pode ter poucos acessos), é possível definir a recomendação mínima para começar e, após seis meses, por exemplo, com uma previsão de grande crescimento, determinar a nova recomendação. Isso garante que a sua empresa e o fornecedor estejam em comum acordo, evitando problemas contratuais ou de suporte."

Use IA

Use o Copilot no Azure. Nem sempre essa opção estará disponível, mas caso esteja, você pode ter uma ajuda da IA, passando dados como quantidade de clientes, operações, arquitetura do sistema etc., para ter uma recomendação. A seguir, deixamos o *link* oficial da Azure que explica como usar essa abordagem.

Using Microsoft Copilot in Azure to find the best VM size for you. Disponível em: <https://techcommunity.microsoft.com/blog/azurecompute/using-microsoft-copilot-in-azure-to-find-the-best-vm-size-for-you/4356049>

Imagen da empresa

Se a empresa valoriza sua imagem e vai lançar um novo produto no mercado que já passou por um piloto, mas você ainda não tem ideia de quantos usuários externos acessarão a aplicação, é importante considerar a variação no número de usuários. No

início, pode haver poucos usuários, porém esse número pode aumentar com o passar das semanas, ou até mesmo explodir com milhares de acessos simultâneos. Uma abordagem de mercado é aumentar os recursos bem acima do necessário inicialmente e, depois de avaliar a quantidade de usuários, ajustar o tamanho, seja aumentando-o ainda mais ou o reduzindo.

A imagem da empresa é muito importante. Imagine uma aplicação com uma estratégia inovadora que promete ser um grande diferencial competitivo no mercado. Se a aplicação falhar logo no início, recuperar a confiança dos usuários pode ser difícil, e a estratégia da empresa pode ir por água abaixo. Por isso, é melhor investir pesado nos recursos de nuvem desde o início. **Prefira pagar mais no começo a perder seu diferencial de mercado.** Alinhe essa estratégia com seu time, seus gerentes e quem mais for necessário. Nunca tome essa decisão sozinho!

Quantidade de usuários

Dependendo da quantidade de usuários esperados na aplicação, por exemplo, 10.000 usuários por dia, você pode ter que, em vez de fazer **uma VM parruda para aguentar isso, fazer várias VMs menores e distribuir os usuários em cada VM com o balanceador de carga.**

Por exemplo, em VMs Microsoft, há algumas aplicações que utilizam recursos do próprio Windows para realizar algum tipo de processamento. Nesse cenário, imagine que, para cada usuário conectado à aplicação, é criado um processo no Gerenciador de Tarefas do Windows, alocando 100 MB de RAM, mesmo que o usuário não esteja utilizando.

Não vamos entrar em detalhes dessa arquitetura, se é boa ou ruim, mas imagine 10.000 usuários X 100 MBs = 1 tera de memória RAM!!!

Imagine o tamanho da VM que precisaria criar, além do fato que, para esse tamanho, provavelmente teria que recorrer à série de VMs mais parrudas, cujo custo é alto, sem contar o fato que se der problema nessa VM, 10.000 usuários vão cair junto.

Agora, se você dividisse essa VM de 1 tera em 20 VMs de 50 GBs de RAM cada. Ai já começa a ficar mais factível, o custo menor e, se uma ou duas VMs derem problema, ainda assim, 18 ficariam ativas, afetando uma pequena parcela dos usuários. Então pense bem na estratégia.

Em resumo, como você percebeu, não existe uma receita mágica para calcular o tamanho correto dos recursos necessários na nuvem para seu projeto. No entanto, essas abordagens o ajudarão a chegar o mais próximo possível daquilo que é necessário, pois o cenário ideal só será alcançado com o uso real da aplicação. E, não, apesar de misturar muitas teorias com prática, isso não é perda de tempo. Criar recursos na nuvem de qualquer jeito só fará você perder tempo, dinheiro e ainda correr o risco de não tornar a aplicação escalável. Então, bora colocar a mão na massa!



EXERCÍCIO COMENTADO

Considerando tudo que vimos nesta aula, o fluxo do sistema e a arquitetura base, vamos fazer um exercício para organizar quais recursos serão necessários, em qual região estarão e as possíveis cargas de trabalho, ou seja, os tamanhos necessários para suportar nosso sistema de liberação de crédito. Qual foi o raciocínio utilizado para chegar a essa conclusão e qual é a sua estratégia para comprovar esse raciocínio?

Também vamos considerar algumas novas expectativas de negócio:

1. O sistema não será mais liberado para o público externo. O cliente chegará ao atendente e fará todo o processo no sistema.
2. O processamento da análise de crédito pode demorar mais do que o previsto e pode passar por uma avaliação manual. Portanto, o cliente não precisa ficar esperando junto ao atendente. Quando a análise estiver concluída, o atendente chamará o cliente novamente.
3. O sistema somente terá a versão *web*, sem *mobile* ou *tablet*.
4. O sistema, além de estar integrado com o Sistema X, que é externo e não requer preocupação com recursos de nuvem, precisará de um mecanismo, como uma API, para permitir a consulta dos cadastros dos clientes e o resultado das análises de aprovação de crédito. Inicialmente, três sistemas externos e simultâneos consultarão esses dados, cada um fazendo 100 requisições por dia.
5. A expectativa é que cada um dos 30 atendentes da empresa receba 20 clientes por dia para avaliar crédito, podendo chegar a um pico diário de 40 clientes em turnos estendidos e em datas especiais, como o Natal.

6. O sistema é importante, mas não é crítico, pois a empresa se mantém por outros serviços que oferece ao público.

Portanto, para evitar custos altos, não é necessário ter redundância. No entanto, gostaria que o sistema, caso ocorresse algum problema e caísse, não deixasse todos os 30 atendentes na mão.

7. O Sistema X é um sistema local, hospedado nos servidores locais que ficam no Brasil.

Agora, que conhece também conhece as regras de negócios, faça anotações para responder às questões:

Quais recursos serão necessários? Em qual região estarão? Quais são as possíveis cargas de trabalho, ou seja, os tamanhos necessários para suportar nosso sistema de liberação de crédito? Qual foi o raciocínio utilizado para chegar a essa conclusão e qual é a sua estratégia para comprovar esse raciocínio?

Depois de fazer suas anotações, confira a seguir, as respostas que elaboramos.



Confira as respostas!



A ideia aqui é que você se atente a vários pontos nas expectativas de negócio. Vamos analisar cada um dos pontos que apresentamos.

1) O sistema não será mais liberado para o público externo. O cliente chegará ao atendente e ele fará todo o processo no sistema.

Ou seja, teremos acessos controlados pela quantidade de atendentes, então não é necessário se preocupar muito com surpresas de acessos no momento. Não precisamos de uma arquitetura robusta, pois sabemos que haverá, no máximo, 30 atendentes por agora.

2) O processamento da análise de crédito pode demorar mais do que o previsto e pode passar por uma avaliação manual.
Portanto, o cliente não precisa ficar esperando junto ao

atendente. Quando a análise estiver concluída, o atendente chamará o cliente novamente.

Ou seja, precisaremos de assincronia e, talvez, de uma nova VM de *back-end* para processar esses dados, devido à carga mais leve de trabalho mencionada no item 1 acima. Essa VM de *back-end* será mais robusta que a VM de *back-end* responsável pelo recebimento das requisições da VM de *front-end* e pelas possíveis integrações.

3) O sistema somente terá a versão web, sem *mobile* ou *tablet*.

Ou seja, apenas “a” ou “as” VMs de *front-end* serão necessárias.

4) O sistema, além de integrado ao Sistema X, que é externo e não requer preocupação com recursos de nuvem, precisará de um mecanismo, como uma API, para permitir a consulta dos cadastros dos clientes e o resultado das análises de aprovação de crédito. Inicialmente, três sistemas externos e simultâneos consultarão esses dados, cada um fazendo 100 requisições por dia.

Ou seja, esses três sistemas externos vão aumentar a carga de trabalho da VM de *back-end*. Se você optar por criar uma VM de *back-end* para processar a análise de crédito, a primeira VM de *back-end* receberá essa carga extra das integrações. Ou, dependendo do seu raciocínio de arquitetura, você pode optar por uma terceira VM de *back-end* apenas para cuidar das integrações. Isso garantiria que, se a aplicação caísse para os atendentes, as integrações continuariam funcionando. Da mesma forma, se as integrações causassem algum problema, as VMs de *back-end* que suportam os atendentes continuariam em operação. Existem várias opções aqui.

5) A expectativa é que cada um dos 30 atendentes da empresa receba 20 clientes por dia para avaliar crédito, podendo chegar a um pico diário de 40 clientes em turnos estendidos e em datas especiais, como o Natal.

Ou seja, sabendo que, em momentos específicos, poderemos ter mais clientes a serem atendidos no dia, não é necessário prever mais VMs para suportar isso, pois ainda serão os mesmos 30 atendentes, cada um atendendo um cliente por vez. No entanto, se nesses momentos específicos o número de atendentes simultâneos aumentar, a situação seria diferente e precisaríamos provisionar mais escalabilidade para nossa infraestrutura.

6) O sistema é importante, mas não é crítico, pois a empresa se mantém por outros serviços que oferece ao público. Portanto, para evitar custos altos, não é necessário ter redundância. No entanto, gostaria que o sistema, caso ocorresse algum problema e caísse, não deixasse todos os 30 atendentes na mão.

Ou seja, apesar de não precisar de redundância, a escalabilidade é sempre necessária, e aqui não é diferente. Portanto, é importante dimensionar bem a quantidade de atendentes por VM, colocando um平衡ador de carga para gerenciar.

7) O Sistema X é um sistema local, hospedado nos servidores locais que ficam no Brasil.

Ou seja, se voltarmos ao item 2, no qual discutimos sobre assincronia, fica claro que não teremos nada em tempo real. Portanto, o sistema de liberação de crédito, visando a reduzir custos, poderia ser hospedado em uma região mais barata na nuvem, enquanto o Sistema X permaneceria em servidores locais no Brasil. No entanto, é crucial realizar testes nesse cenário, pois a latência pode ser tão alta que, mesmo com a integração assíncrona, ela pode se tornar inviável, resultando em muitos erros de *time out*. Esse erro é bastante comum na programação de sistemas, ocorrendo quando o tempo de espera entre um recurso e outro atinge o limite.

Não se preocupe em saber se, a VM ou recurso que escolher tem X vCPUs, Y de RAM, Z de armazenamento ou W de IOPS. O importante no momento é entender bem sua arquitetura e quais recursos são necessários. Avalie se esses recursos precisarão de maior capacidade, como mais processamento ou memória mais rápida. Considere se você se baseou em algum

projeto anterior ou se fará um piloto controlado para testar. Em outras palavras, como você montou seu racional e como vai comprová-lo. Isso já lhe dará toda a base necessária para começar a trabalhar na nuvem. Combinado?

Não deixe de tirar suas dúvidas com o professor-tutor!

| Conclusão

Gostou de fazer a arquitetura do nosso projeto? Mais do que isso, percebeu que há muitas formas de fazer essa arquitetura na nuvem e que não existe mecanismo mágico para acertar os tamanhos dos recursos?

Sim, há diversas formas de fazer a mesma aplicação, desde a mais simples até as mais complexas, auditáveis, escaláveis e com redundância.

Isso vai variar de acordo com as necessidades do projeto, o que o projeto pode entregar e como atingir as expectativas de negócio, que são a base para seu projeto. Agora, você tem uma bagagem legal inicial, para fomentar o uso da nuvem e como utilizá-la de maneira mais adequada.

E, sim, é um aprendizado constante, de erros e acertos, e isso faz parte do jogo, então bora para frente que nosso projeto ainda não acabou! Semana que vem vamos colocar nossa arquitetura e a aplicação na nuvem.

Até a próxima!

