

Web Services (Serviço de Aplicativo)

Características do Serviço de Aplicativos do Azure:

1- O que é:

O serviço de aplicativo do Azure é um serviço de PaaS baseado em HTTP para implementação rápida e escalável de aplicações Web, e aplicações Back-End.

2- Plano de serviço de aplicativo: no Web Service um aplicativo sempre é executado em um Plano de Serviço de Aplicativo. E todos os serviços adicionados nesse plano seguem as mesmas definições de:

- Sistema operacional (Windows e Linux)
- Região (Oeste dos EUA, Leste dos EUA, etc.)
- Número de instâncias de VM
- Tamanho de instâncias de máquina virtual (pequeno, médio, grande)
- Tipo de preço (Gratuito, Compartilhado, Básico, Standard, Premium, PremiumV2, PremiumV3, Isolado, IsoladoV2)

- a. **Computação compartilhada:** Executam um aplicativo na mesma máquina virtual do Azure como outros aplicativos do serviço de aplicativo e **não podem escalar horizontalmente**.
- b. **Computação dedicado:** Executam os aplicativos nas VMs dedicadas do Azure. Quanto maior o plano mais recursos disponíveis.
- c. **Isolado:** Executa VMs do Azure em Redes Virtuais do Azure dedicadas e fornece a capacidade máxima de expansão.

3- Autenticação e autorização:

O Serviço de Aplicativo do Azure dá suporte interno à autenticação e autorização para que você possa fazer logon de usuários e acessar dados.

a- Provedores de Identidade:

Provedor	Ponto de extremidade de logon	Diretrizes
Plataforma de identidade da Microsoft	/.auth/login/aad	<u>Logon da plataforma de identidade da Microsoft do Serviço de Aplicativo</u>
Facebook	/.auth/login/facebook	<u>Logon do Facebook no Serviço de Aplicativo</u>
Google	/.auth/login/google	<u>Logon do Google no Serviço de Aplicativo</u>
Twitter	/.auth/login/twitter	<u>Logon do Twitter no Serviço de Aplicativo</u>
Qualquer provedor do OpenID Connect	/.auth/login/<providerName>	<u>Logon do OpenID Connect no Serviço de Aplicativo</u>
GitHub	/.auth/login/github	<u>Serviço de Aplicativo Logon do GitHub</u>

Ao habilitar a autenticação e autorização com um desses provedores, seu ponto de extremidade de logon estará disponível para autenticação de usuário e validação de tokens de autenticação do provedor.

b- Como funciona:

Quando habilitado, toda solicitação HTTP de entrada passa por ele antes de ser manipulada pelo código do aplicativo.

c- Fluxo de Autenticação:

Etapa	Sem SDK do provedor	Com SDK do provedor
Conectar usuário	Redireciona o cliente para <code>/.auth/login/<provider></code> .	O código do cliente conecta o usuário diretamente no SDK do provedor e recebe um token de autenticação. Para obter informações, consulte a documentação do provedor.
Pós-autenticação	Provedor redireciona o cliente para <code>/.auth/login/<provider>/callback</code> .	O código do cliente envia o token do provedor para <code>/.auth/login/<provider></code> para validação.
Estabelecer sessão autenticada	O Serviço de Aplicativo adiciona um cookie autenticado à resposta.	O Serviço de Aplicativo retorna o próprio token de autenticação para o código do cliente.
Atender conteúdo autenticado	O cliente inclui o cookie de autenticação em solicitações subsequentes (manipuladas automaticamente pelo navegador).	O código do cliente apresenta o token de autenticação no cabeçalho <code>x-zumo-AUTH</code> (manipulado automaticamente pelos SDKs de cliente dos Aplicativos Móveis).

d- Comportamento de Autorização:

No portal do Azure, é possível configurar o Serviço de Aplicativo com vários comportamentos quando uma solicitação de entrada não é autenticada. Permitindo a **Entrada não autenticada** ou **Exigir a autorização**.

e- Token Store:

É um repositório de armazenamento que armazena os tokens dos usuários.

4- Recursos de Rede:

Por padrão todos os Web Services aceitam requisições na internet, porém às vezes é necessário a modificação dos acessos a rede, para isso nós temos duas estratégias, **Serviço de Aplicativo multilocatário** e **Rede Padrão**.

a. Serviço de Aplicativo multilocatário:

O Serviço de Aplicativo do Azure é um sistema distribuído. As funções que tratam solicitações HTTP ou HTTPS de entrada são chamadas de front-ends. As funções que hospedam a carga de trabalho do cliente são chamadas de Trabalhos. Todas as funções em uma implantação do Serviço de Aplicativo são encontradas em uma rede multilocatário.

b. Endereço atribuído ao aplicativo:

Trata-se de um endereço web atribuído a VM do Azure em que o recurso está e que é atribuído um subdomínio ao serviço.

c. Pontos de Extremidade de Serviço:

São um recurso de rede de Entrada.

d. Conexões Híbridas:

São um recurso de rede de Saída.

e. Rede Padrão:

As unidades de escala do Serviço de Aplicativo do Azure suportam muitos clientes em cada implantação.

➔ **Planos Gratuitos e Compartilhados:** Hospedam Cargas de trabalho do cliente em trabalhos multilocatários.

➔ **Planos Básico e Superiores:** Hospedam cargas de trabalho do cliente a apenas um plano ligado ao Serviço de Aplicativo.

➔ **Planos Standart:** São executados no mesmo trabalho.

f. Endereços de saída:

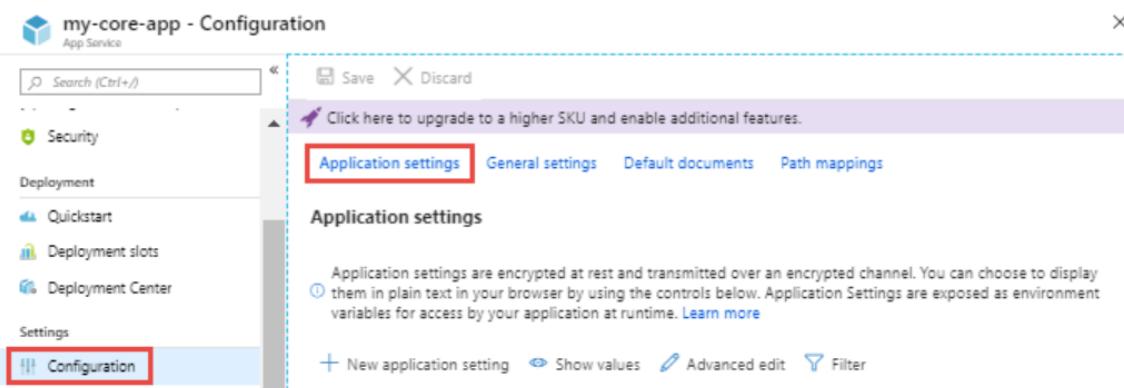
As VMs de trabalho são divididas apartir dos planos, tendo o PremiumV2 e o PremiumV3 planos de VMs diferentes.

Os Endereços de Saída são listados nas propriedades e são compartilhados por todos os aplicativos em execução na mesma família de VMs de trabalho da implantação.

g. Localizar IPs de saída:

Podemos utilizar as propriedades do recurso no Portal do Azure ou usar o comando **possibleOutboundIpAddresses** no CLI do Azure.

Configurações do Serviço de Aplicativos do Azure:



1- Configurações do Aplicativo:

As configurações do Aplicativo dizem respeito a **Variáveis de ambiente** que são aplicadas a aplicação, podem ser acessadas em **Configuração > Configurações de aplicativo**

Por padrão todas as configurações de aplicativo **São criptografadas**.

a. Adicionar Configurações de aplicativo:

Para adicionarmos configurações de aplicativo, vamos na aba adicionar e inserimos um **Nome** e **Valor**.

b. Editar configurações em Massa:

A opção de edição em massa mostra um JSON em que cada item do array é uma configuração do aplicativo.

c. Comportamento das configurações de aplicativo em diferentes tipos de implementações:

Diferentes tipos de implementações como .Net, Docker ou Java podem ter comportamentos diferentes na implementação.

- ➔ **.NET:** As configurações de aplicativos adicionadas a implantação substituem as adicionadas no Web.config.
- ➔ **Docker:** As configurações de aplicativos são injetadas no container através do comando –env.

2- Configurações gerais:

Você pode definir algumas configurações comuns para seu aplicativo.

- a. **Configurações de pilha:** Controla as versões dos SDKs e pode ser definido um comando inicial.
- b. **Configurações de plataforma:** Permite definir configurações da plataforma de hospedagem.
 - ➔ **Número de bits**
 - ➔ **Protocolo WebSocket**
 - ➔ **Always On:** Mantenha o aplicativo carregado mesmo quando não há tráfego.
 - ➔ **Versão do pipeline gerenciado:** Modo de pipeline do IIS.
 - ➔ **Versão HTTP**
 - ➔ **Afinidade ARR**
- c. **Depuração:**

Ativa ou desativa a depuração remota para aplicativos ASP.NET ou Node.js.
- d. **Certificados de cliente de entrada:**

Exige autenticação mútua TLS.

3- Mapeamentos de Caminho:

Exibe diferentes opções com base no tipo de SO.

- a. **Aplicativos do Windows:**

você pode personalizar os mapeamentos do manipulador do IIS, os diretórios e os aplicativos virtuais.

Os mapeamentos do manipulador permitem que você adicione processadores de script personalizado para manipular solicitações para extensões de arquivo especificadas.
- b. **Aplicativos Linux e em contêineres:**

Você pode adicionar armazenamento personalizado para seu aplicativo conteinerizado.

4- Registro em log:

É possível habilitar os registros de Logs da aplicação no Azure.

a. Habilitar log de aplicativo(windows):

Em **Logs do Serviço de Aplicativo** você pode ativar o log em **Filesystem** ou **BLOB** ou ambos, Filesystem dura 12 horas, e BLOB é de longo prazo ficando em uma storage Account.

É possível definir o nível de detalhes dos logs:

Nível	Categorias incluídas
Desabilitado	Nenhum
Erro	Erro, Crítico
Aviso	Aviso, Erro, Crítico
Informações	Informações, Aviso, Erro, Crítico
Verbose	Rastreamento, Depuração, Informações, Aviso, Erro, Crítico (todas as categorias)

b. Habilitar log de aplicativo (Linux/contêiner):

Em **Logs do Serviço de Aplicativo** defina a opção **Log do aplicativo** para **Sistema de Arquivos**. Você deve configurar o **período de retenção(dias)** e a **cota(mb)**.

c. Habilitar o log do servidor web:

Defina o armazenamento em BLOB ou em sistema de arquivos e o **período de retenção**.

d. Transmissão de Logs:

Por padrão todos os arquivos de logs armazenados em **d:/home/logfiles** são disponibilizados.

Para transmitir logs no cli do Azure use o seguinte comando:

```
az webapp log tail --name appname --resource-group myResourceGroup.
```

e. Acessar arquivos de log:

Para BLOBS fica a cargo do cliente encontrar uma maneira de ler o BLOB disponível na web.

Para Logs em sistemas de arquivos podemos baixar o zip em:

- Aplicativos de contêiner/Linux: <https://<appname>.scm.azurewebsites.net/api/logs/docker/zip>
- Aplicativos do Windows: <https://<appname>.scm.azurewebsites.net/api/dump>

5- Certificado de segurança:

Você pode criar, carregar ou importar um certificado privado ou um público para o serviço de aplicativo, por padrão o certificado é vinculado a uma unidade de implantação que está vinculada ao grupo de recursos.

Formas de implantar um certificado:

Opção	Descrição
Para criar um certificado gerenciado gratuito do Serviço de Aplicativo:	Um certificado privado sem custo e fácil de usar se você precisar proteger seu domínio personalizado no Serviço de Aplicativo.
Comprar um certificado do Serviço de Aplicativo	Um certificado privado gerenciado pelo Azure. Ele combina a simplicidade do gerenciamento automatizado de certificado e a flexibilidade das opções de renovação e exportação.
Importar um certificado do Key Vault	Útil se você usa o Azure Key Vault para gerenciar seus certificados.
Carregar um certificado privado	Se você já tiver um certificado privado de um provedor de terceiros, poderá carregá-lo.
Carregar um certificado público	Os certificados públicos não são usados para proteger domínios personalizados, mas você pode carregá-los em seu código se precisar que eles acessem recursos remotos.

Escalar Aplicativos no Serviço de Aplicativo:

Podemos configurar como o dimensionamento automático irá se comportar no serviço de aplicativo.

1- Fatores do Dimensionamento automático:

O dimensionamento automático é um processo ou sistema de nuvem que ajusta os recursos disponíveis com base na demanda atual. O dimensionamento automático expande e reduz horizontalmente, em vez de escalar e reduzir verticalmente.

a. Regras do Dimensionamento automático:

No Azure podemos criar regras criando limites e disparando um evento de dimensionamento.

b. Quando usar:

O Dimensionamento automático aprimora a disponibilidade e a tolerância a falhas, nesse sentido o uso do Dimensionamento deve estar ligado a esses fatores. Tomando cuidado com o gerenciamento de custos.

c. Dimensionamento automático e os Planos de serviço:

O dimensionamento automático está intrinsecamente ligado ao plano de serviço, planos com preço mais caro tem um limite maior de instâncias de dimensionamento.

d. Condições para o Dimensionamento automático:

Podemos criar condições de duas formas:

- **Base em métrica:** escala com base em uma métrica do sistema.
- **Contagem de instâncias especificada baseada em cronograma:** é possível escalar com base no cronograma e determinar instâncias nesses períodos.

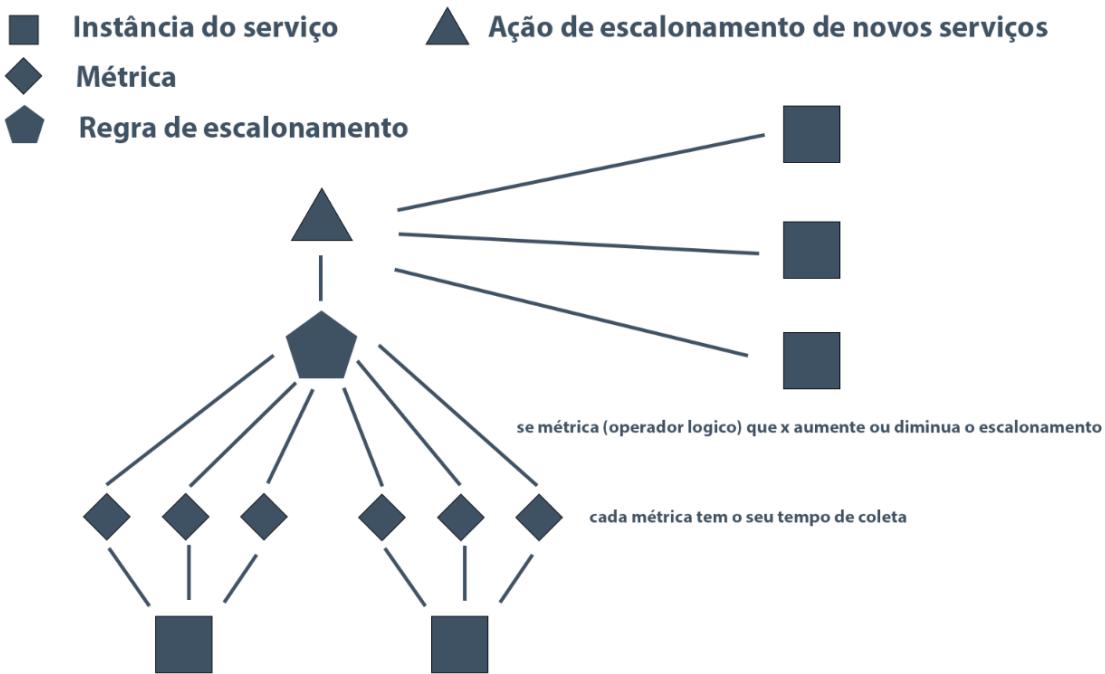
e. Métricas para as regras de dimensionamento automático:

Para criarmos regras de dimensionamento podemos utilizar as seguintes métricas ou métricas de outros serviços associados ao serviço de aplicativo:

- . Percentual de CPU
- . Percentual de memória
- . Comprimento da fila de disco
- . Tamanho da fila de HTTP
- . Entrada de dados
- . Saída de dados

f. Análise de métricas e Ações:

O dimensionamento automático possui um ciclo de análise e tomada de ações:



g. Combinação de regras:

Uma regra de escalonamento pode conter diversas condições e cada condição diz a ação de escalonamento, exemplo:

- Se o comprimento da fila HTTP exceder 10, expanda horizontalmente em 1
- Se o uso da CPU exceder 70%, expanda horizontalmente em 1

Se **qualquer** uma das regras for verdadeira o escalonamento irá acontecer.

2- Habilitar o dimensionamento automático:

Por padrão um pano de serviço só implementa o **dimensionamento manual**, para implementarmos uma regra de vemos ir em **Configurações, plano do serviço de aplicativo, Escalar horizontalmente**

Autoscale is a built-in feature that helps applications perform their best when demand changes. You can choose to scale your resource manually to a specific instance count, or via a custom Autoscale policy that scales based on metric(s) thresholds, or schedule instance count which scales during designated time windows. Autoscale enables your resource to be performant and cost effective by adding and removing instances based on demand. [Learn more about Azure Autoscale](#) or view the [how-to video](#).

Choose how to scale your resource

- Manual scale** (selected): Maintain a fixed instance count
- Custom autoscale**: Scale on any schedule, based on any metrics

Manual scale

Override condition

Instance count: 1

a. Adicionar condições:

Default*: Auto created default scale condition

Scale mode: Scale based on a metric (radio button)

Instance count*: 1

Schedule: This scale condition is executed when none of the other scale condition(s) match

Auto created scale condition 1

Scale mode: Scale based on a metric (radio button)

Rules: No metric rules defined; click [Add a rule](#) to scale out and scale in your instances based on rules. For example: 'Add a rule that increases instance count by 1 when CPU percentage is above 70%'. If you save the setting without any rules defined, no scaling will occur.

+ Add a rule

Instance limits: Minimum 1, Maximum 2, Default 1

Schedule: Specify start/end dates (radio button)

Timezone: (UTC-08:00) Pacific Time (US & Canada)

Start date: 07/17/2021, 12:00:00 AM

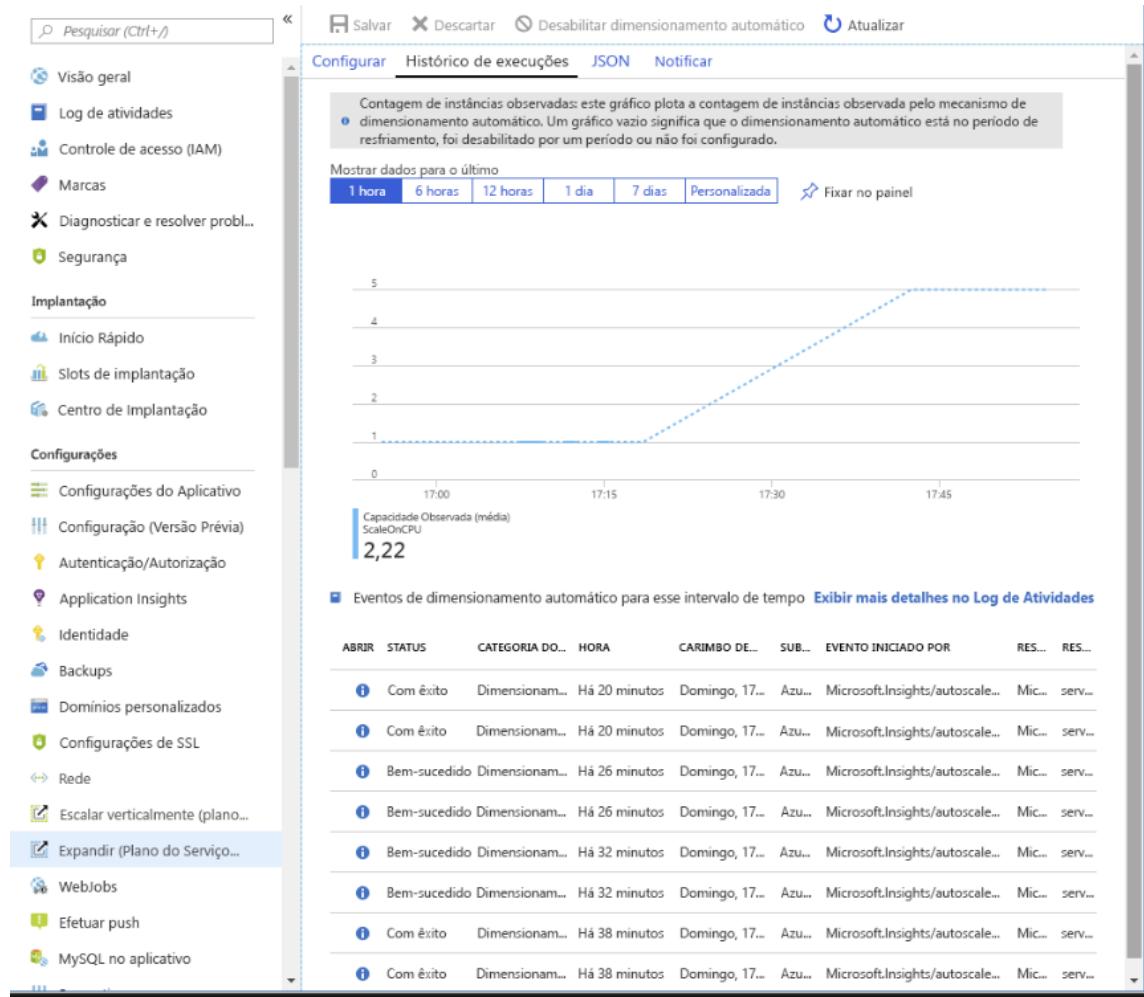
End date: 07/17/2021, 11:59:00 PM

b. Criar regra de dimensionamento:

The screenshot shows the Azure portal interface for managing an App Service plan named 'game'. On the left, there's a summary card with 'Resource group' (game) and 'Instance count' (1). Below it, the 'Default' scale condition is shown, where 'Scale mode' is set to 'Scale based on a metric' (selected). The 'Instance count' is set to 1. A note says 'This scale condition is executed when none of the other scale condition(s) match'. To the right, a detailed 'Scale rule' configuration window is open. It shows the 'Metric source' as 'Current resource' and 'Resource type' as 'App Service plans' with 'Resource' ASI-game-89c8 selected. Under 'Criteria', 'Time aggregation' is set to 'Average'. The 'Metric namespace' is 'App Service plans standard metrics' and the 'Metric name' is 'CPU Percentage'. The 'Dimension Name' is 'Instance', 'Operator' is '=', and 'Dimension Values' is 'All values'. A note states: 'If you select multiple values for a dimension, autoscale will aggregate the metric across the selected values, not evaluate the metric for each values individually.' Below this, a chart titled 'CpuPercentage (Average)' shows a single data point at 1.93%. The 'Add' button is highlighted with a red box.

c. Monitorar o dimensionamento automático:

Podemos monitorar o dimensionamento automático através do gráfico de **Histórico de execuções**:



3- Conceitos de dimensionamento automático:

- **Dimensionamento Automático:**
 - 1- Escala horizontalmente, aumentando ou reduzindo instâncias.
 - 2- Define valores máximo, mínimo e padrão de instâncias.
- **Critérios de Dimensionamento:**
 - 1- Lê métricas associadas para expansão ou redução.
 - 2- Limites configurados em um nível de instância.
- **Registro de Atividades:**
 - 1- êxitos e falhas são registrados no Log de Atividades.
- **Alertas:**
 - 1- Configura alertas no Log de Atividades para notificações por email, SMS ou webhook.

Slots de implantação do Serviço de aplicativo:

A funcionalidade do slot de implantação no Serviço de Aplicativo é uma ferramenta que permite visualizar, gerenciar, testar e implantar seus diferentes ambientes de desenvolvimento.

1- Conceitos:

Na implementação do serviço de aplicativo podemos usar um slot de implantação diferente de produção, para isso devemos usar os Planos **Standard, Premium ou Isolado**.

Cada tipo de Plano de serviço dá um número diferentes de slots de implantação.

2- Troca de Slots:

- a. **Estágio 1:** Copia as configurações do Slot de destino para o Slot de origem e pausa a troca.
 - Configurações do aplicativo e cadeias de conexão específicas do slot, se aplicável.
 - Configurações de implantação contínua, se habilitadas.
 - Configurações de autenticação do Serviço de Aplicativo, se habilitadas.
- b. **Estágio 2:** Aguarda até que todas as instâncias no slot de origem reiniciem caso ocorra erro a operação de troca reverte as mudanças.
- c. **Estágio 3:** Se o cache local estiver habilitado executa uma requisição HTTP para a raiz do aplicativo em cada instância.
- d. **Estágio 4:** Se a troca automática estiver habilitada com a preparação personalizada, dispara uma solicitação HTTP para raiz do aplicativo em todas as instâncias
- Se **applicationInitialization** não for especificado, dispare uma solicitação HTTP para a raiz do aplicativo em cada instância do slot de origem.
- Se uma instância retornar qualquer resposta HTTP, ela é considerada como ativada.
- e. **Estágio 5:** Se todas os estágios forem bem-sucedidos os Slots são trocados de lugar.
- f. **Estágio 6:** Agora aplica as configurações e reinicia as instâncias.

3- Tipos de Troca de Slots:

- a. **Troca manual:** Essa troca ocorre com o usuário navegando até a página **Slot de Implantação** selecionando **Trocá**, em seguida o usuário especifica o Slot de **Origem** e o Slot de **Destino** o usuário deve especificar também as **Alterações de Origem** e as **Alterações de Destino**.
- b. **Troca com visualização:** Essa troca ocorre em várias etapas com o usuário verificando as alterações e seguindo para as próximas fases.
- c. **Troca automática:** Essa troca ocorre de forma automática quando você faz um push no Slot de Origem, essa troca é otimizada para cenários do **Azure DevOps Services**.

4- Roteamento de tráfego entre slots:

- a. **Roteamento automatico:** É possível configurar uma porcentagem de tráfego aleatório que será redirecionado para o Slot de origem, após isso o usuário redirecionado terá a sua sessão inteiramente no ambiente de Origem.
- b. **Roteamento Manual:** É possível pelo código do usuário aceitar ou tornar opção a aceitação do ambiente de origem:

Para permitir que os usuários recusem a versão beta do aplicativo, por exemplo, você pode colocar este link na sua página Web:

HTML Copiar

```
<a href=""><webappname>.azurewebsites.net/?x-ms-routing-name=self">Go back to production app</a>
```

Para permitir que os usuários aceitem o aplicativo beta, defina o mesmo parâmetro de consulta como o nome do slot de não produção. Aqui está um exemplo:

HTML Copiar

```
"><webappname>.azurewebsites.net/?x-ms-routing-name=staging
```

Azure Functions

O Azure Functions é uma solução de implementação de código sem servidor.

Conceitos:

1- Diferenças entre Azure Functions, Aplicativos lógicos e WebJobs:

O Azure Functions é focado em uma programação **sem servidor utilizando código (CODE-FIRST)**.

O aplicativo lógico é focado em computação **sem servidor sem código (DESIGNER-FIRST)**.

O WebJobs é um serviço do **Serviço de aplicativo** que o código irá utilizar SDKs para integração de recursos do serviço de aplicativo (**CODE-FIRST**).

2- Opções de Hosteragem do Azure Functions:

É necessário escolher um plano de serviço para o Azure Functions e esse plano irá determinar:

- Como o aplicativo de funções é dimensionado.
- Os recursos disponíveis para cada instância do aplicativo de funções.
- Suporte para funcionalidades avançadas, como conectividade à Rede Virtual do Azure.

Plano	Vantagens
Plano de consumo	Este é o plano de hospedagem padrão. Ele escala automaticamente e você paga apenas pelos recursos de computação quando suas funções estiverem em execução. As instâncias do host do Functions são adicionadas e removidas de maneira dinâmica com base no número de eventos de entrada.
Plano Premium	Escala automaticamente com base na demanda usando trabalhos pré-configurados que executam aplicativos sem atraso após estarem ociosos, é executado em instâncias mais poderosas e se conecta a redes virtuais.
Plano dedicado	Execute suas funções em um plano do Serviço de Aplicativo com taxas regulares do Plano do Serviço de Aplicativo. Melhor para cenários de execução longa em que o <u>Durable Functions</u> não pode ser usado.

ASE	O ASE (Ambiente do Serviço de Aplicativo) é um recurso do Serviço de Aplicativo que fornece um ambiente totalmente isolado e dedicado a executar com segurança os aplicativos do Serviço de Aplicativo em grande escala.
Kubernetes (<u>Direct</u> ou <u>Azure</u> <u>Arc</u>)	O Kubernetes fornece um ambiente totalmente isolado e dedicado em execução na plataforma Kubernetes.

a. Plano de Hospedagem e escala:

Plano	Escalar horizontalmente	Número máximo de instâncias
Plano de Consumo	Controlado por evento. Escale horizontalmente de forma automática, mesmo durante períodos de carga alta. A infraestrutura do Azure Functions escala os recursos de CPU e memória adicionando mais instâncias do host do Functions de acordo com o número de eventos de gatilho de entrada.	Windows: 200, Linux: 100
Plano Premium	Controlado por evento. Escale horizontalmente de forma automática, mesmo durante períodos de carga alta. A infraestrutura do Azure Functions escala os recursos de CPU e memória adicionando mais instâncias do host do Functions de acordo com o número de eventos nos quais suas funções são disparadas.	Windows: 100, Linux: 20-100
Plano dedicado	Dimensionamento manual/automático	10-20
ASE	Dimensionamento manual/automático	100
Kubernetes	Dimensionamento automático controlado por eventos para clusters Kubernetes usando KEDA.	Varia de acordo com o cluster

b. Duração do tempo limite:

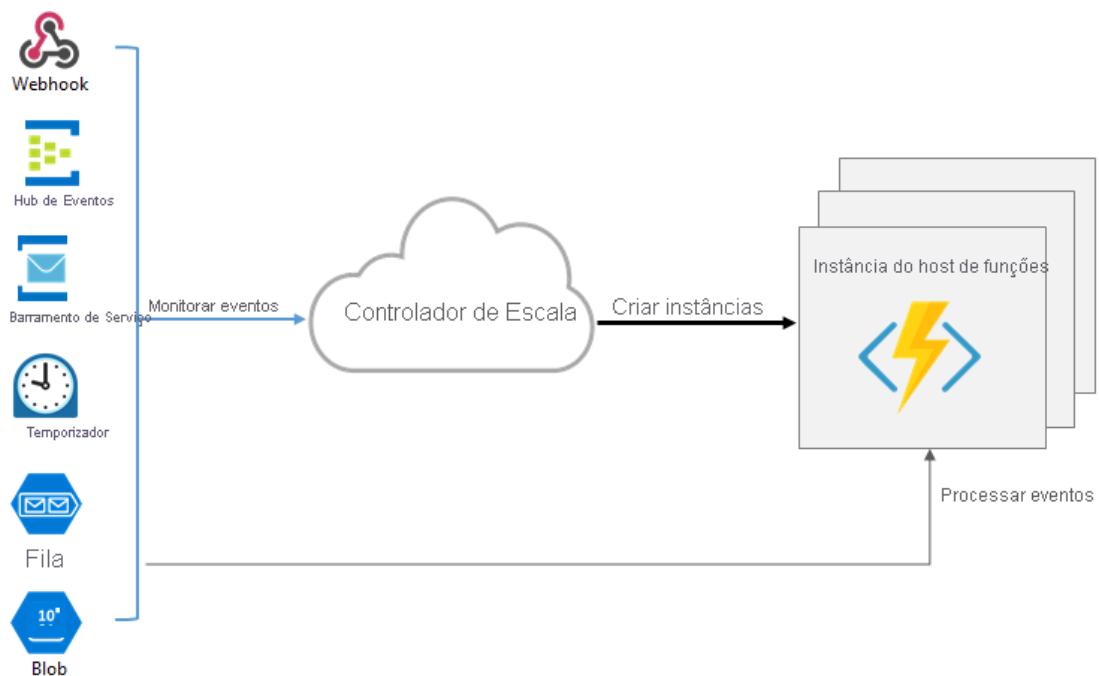
Plano	Padrão	Máximo
Plano de consumo	5	10
Plano Premium	302	Ilimitado
Plano dedicado	302	Ilimitado

3- Escalonamento:

O escalonamento automático ocorre na replicação de instâncias através do número de eventos.

O código da função é armazenado em um arquivo na **Storage Account** e replicado na instância.

a. Escalonamento Runtime:



É possível limitar a quantidade de instâncias através da propriedade no código **functionAppScaleLimit**.

Desenvolvimento de funções:

1- Durable Functions:

As Durable Functions são uma extensão do Azure Functions que permite escrever funções com estado em um ambiente de computação sem servidor.

Armazenamento de Blobs

É a solução de armazenamento do Azure para grandes quantidades de dados não estruturados.

1- Contas de armazenamento: é o contêiner de nível superior de todo o Armazenamento de Blobs do Azure. Uma conta de armazenamento fornece um **namespace** exclusivo para os dados do Armazenamento do Microsoft Azure que podem ser acessados de qualquer lugar do mundo por HTTP ou HTTPS.

Temos dois níveis de conta de armazenamento, **Padrão** chamada de padrão v2, é de uso geral e recomendada para a maioria dos cenários e temos a **Premium** que oferece melhor desempenho e a possibilidade de configuração de disponibilização de diferentes meios.

2- Contêiners: Um contêiner organiza um conjunto de blobs, semelhante a um diretório, podemos ter um número ilimitado em uma conta de armazenamento. O nome do contêiner deve ser um DNS valido já que irá entrar na URI.

Bash

```
https://myaccount.blob.core.windows.net/mycontainer
```

3- Blobs: temos 3 tipos de Bobs:

- ➔ **Blob de blocos:** armazena dados de texto e binários, são compostos de blocos de dados.
- ➔ **Blobs de acréscimo:** Igual o Blob de blocos, porém otimizado para operações de acréscimo.
- ➔ **Blobs de Páginas:** Armazenam arquivos VHD (disco rígido virtual) e servem de discos para as VMs.

4- Recursos de segurança: Por padrão todos os dados em uma conta de armazenamento são criptografados com AES de 256 bits parecido com o bitLocker do Windows. Podemos ou não criptografar a conexão HTTPs ou SMB 3.0 e a conta de armazenamento pode ter controles configurados pelo RBAC ou Entra ID.

Podemos usar as chaves criptográficas geradas automaticamente ou podemos configurar as nossas próprias.

5- Hospedagem de site estático: É possível disponibilizar arquivos web estáticos através da conta de armazenamento, porém dessa forma temos algumas desvantagens como não podermos personalizar os cabeçalhos das requisições, trabalhar com AuthN e AuthZ, além de não podermos controlar o

acesso dos usuários a página estática, sobrando apenas a utilização do CDN do Azure (Rede de Distribuição de Conteúdo do Azure) para controle.

Por padrão devemos fornecer o arquivo de index.html e um arquivo de erro, em seguida o Azure irá criar um container chamado \$web para a página:

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with 'Microsoft Azure', a search bar, and a 'Pesquisar recursos, serviços e documentos (G+/-)' button. Below the navigation bar, the page title is 'Página Inicial > contosoaccount'. The main content area is titled 'Contoso | Site estático' and shows a 'Conta de armazenamento'. On the left, there's a sidebar with 'Gerenciamento de dados' and several options: 'Replicação geográfica', 'Proteção de dados', 'Replicação de objeto', 'Site estático' (which is selected and highlighted in grey), 'Gerenciamento do ciclo de vida', and 'Azure Search'. The main panel displays configuration settings for a static website. It includes a note about enabling static sites in Blob storage, a status switch for 'Site estático' (set to 'habilitado'), and two input fields: 'Nome do documento de índice' containing 'index.html' and 'Caminho do documento de erro' containing '404.html'. There are 'Salvar' and 'Descartar' buttons at the top right of the configuration panel.

6- Ciclo de vida do armazenamento em BLOB:

Os dados naturalmente têm um ciclo de vida sendo acessados mais no seu início de vida e menos no final. Pensando nesse cenário o armazenamento de BLOBs possui configurações de gerenciamento.

a. Níveis de Acesso:

- . **Frequente**: Otimizado para dados acessados com frequência.
- . **Esporádico**: Otimizado para dados armazenados a mais de 30 dias e não são acessados com frequência.
- . **Camada fria**: Otimizado para dados armazenados a mais de 90 dias e com baixa frequência. Possui custo mais baixo de armazenamento e custo mais caro de acesso.
- . **De Arquivos**: Possui latência de horas e é para arquivos armazenados a mais de 180 dias e raramente acessados.

b. Regras de gerenciamento do ciclo de vida:

As regras são configuradas através de um JSON que contêm um array com os objetos da regra:

```
{
  "rules": [
    {
      "name": "rule1",
      "enabled": true,
      "type": "Lifecycle",
      "definition": {...}
    },
    {
      "name": "rule2",
      "type": "Lifecycle",
      "definition": {...}
    }
  ]
}
```

```
{
  "rules": [
    {
      "name": "ruleFoo",
      "enabled": true,
      "type": "Lifecycle",
      "definition": {
        "filters": {
          "blobTypes": [ "blockBlob" ],
          "prefixMatch": [ "container1/foo" ]
        },
        "actions": {
          "baseBlob": {
            "tierToCool": { "daysAfterModificationGreaterThan": 30 },
            "tierToArchive": { "daysAfterModificationGreaterThan": 90 },
            "delete": { "daysAfterModificationGreaterThan": 2555 }
          },
          "snapshot": {
            "delete": { "daysAfterCreationGreaterThan": 90 }
          }
        }
      }
    }
  ]
}
```

Nome do parâmetro	Tipo de parâmetro	Observações
rules	Uma matriz de objetos de regra	Pelo menos uma regra é necessária em uma política. Você pode definir até 100 regras em uma política.

Parâmetros regras:

Nome do parâmetro	Tipo de parâmetro	Observações	Obrigatório
name	String	Um nome de regra pode incluir até 256 caracteres alfanuméricos. A regra de nome diferencia maiúsculas de minúsculas. Ela deve ser exclusiva em uma política.	Verdadeiro
enabled	Boolean	Um booleano opcional para permitir que uma regra seja desabilitada temporariamente. O valor padrão será true se não estiver definido.	Falso
type	Um valor de enumeração	O tipo válido atual é ciclo de vida.	Verdadeiro
definition	Um objeto que define a regra de ciclo de vida	Cada definição é composta por um conjunto de filtros e um conjunto de ações.	Verdadeiro

Filtros das regras:

Nome do filtro	Tipo	Obrigatório
blobTypes	Uma matriz de valores de enumeração predefinidos.	Yes
prefixMatch	Uma matriz de cadeias de caracteres para prefixos a serem correspondidos. Cada regra pode definir até 10 prefixos. Uma cadeia de caracteres de prefixo deve começar com um nome de contêiner.	Não
blobIndexMatch	Uma matriz de valores de dicionário que consistem em condições de valor e chave de marca de índice de blob a serem combinadas. Cada regra pode definir até 10 condições de marca de Índice de blob.	Não

Ações da regra:

Ação	Blob base	Instantâneo	Versão
tierToCool	Com suporte para blockBlob	Com suporte	Com suporte
enableAutoTierToHotFromCool	Com suporte para blockBlob	Sem suporte	Sem suporte
tierToArchive	Com suporte para blockBlob	Com suporte	Com suporte
excluir	Com suporte para blockBlob e appendBlob	Com suporte	Com suporte

Condições de execução:

Condição de execução de ação	Valor de condição	Descrição
daysAfterModificationGreaterThan	Valor inteiro que indica a idade em dias	Condição para ações de blob de base
daysAfterCreationGreaterThan	Valor inteiro que indica a idade em dias	A condição para ações de instantâneo de blob
daysAfterLastAccessTimeGreaterThan	Valor inteiro que indica a idade em dias	A condição para uma versão atual de um blob quando o controle de acesso está habilitado
daysAfterLastTierChangeGreaterThan	Valor inteiro que indica a idade em dias após o último horário de alteração da camada de blob	Essa condição se aplica apenas a ações tierToArchive e só pode ser usada com a condição daysAfterModificationGreaterThan.

c. Implementar políticas de ciclo de vida do armazenamento de blobs:

Você pode adicionar, editar ou remover uma política usando qualquer um dos seguintes métodos:

Portal do Azure

Azure PowerShell

CLI do Azure

APIs REST

. Lista Portal do Azure:

1. Entre no [portal do Azure](#).
2. Selecione **Todos os recursos** e, em seguida, selecione sua conta de armazenamento.
3. Em **Gerenciamento de dados**, selecione **Gerenciamento de ciclo de vida** para exibir ou alterar suas regras.
4. Selecione a guia **Exibição de lista**.

5. Selecione **Adicionar regra** e preencha os campos do formulário **Conjunto de ações**.
6. Selecione **Conjunto de filtros** para adicionar um filtro opcional. Então selecione Procurar para especificar um contêiner e uma pasta pela qual filtrar.
7. Selecione **Examinar + adicionar** para examinar as configurações de política.
8. Selecione **Adicionar** para adicionar a nova política.

. Código Portal do Azure:

Após ter criado a regra através da lista vá para **Exibição do Código** e você terá acesso ao JSON.

. CLI do Azure:

Crie a regra e adicione em um arquivo .JSON, após isso chame o seguinte comando no CLI:

```
az storage account management-policy create \
--account-name <storage-account> \
--policy @policy.json \
--resource-group <resource-group>
```

d. Reidratar dados de blob da camada de arquivos:

Quando o blob está na camada de arquivos ele esta offline e não pode ser lido ou alterado. Para reidratar o blob existem dois processos:

. **Copiar o blob para a camada online:** Podemos pegar o blob e copiá-lo para as camadas quente ou fria, esse processo é recomendado na maioria dos cenários pela Microsoft.

. **Alterar a camada de acesso do blob para online:** podemos usar o **Set Blob Tier** e alterar a camada para quente ou fria em um blob.

Prioridade de reidratação:

Podemos definir a prioridade para **padrão** ou **Alta prioridade** alterando o cabeçalho **x-ms-rehydrate-priority**. Na prioridade padrão a solicitação é processada na ordem que foi recebida e pode levar até 15 horas. Na prioridade

alta ela tem superioridade a padrão e pode levar menos de uma hora com objetos até 10 gb.

7- Trabalhar com o armazenamento de Blobs do Azure:

O Azure tem uma biblioteca para controlar os Blobs no .net chamada de **Azure Storage Blobs**.

Essa biblioteca possui algumas classes principais:

Classe	Descrição
BlobServiceClient	Representa a conta de armazenamento e fornece operações para recuperar e configurar as propriedades da conta e para trabalhar com contêineres de blob na conta de armazenamento.
BlobContainerClient	Representa um contêiner de blob específico e fornece operações para trabalhar com o contêiner e os blobs dentro dele.
BlobClient	Representa um blob específico e fornece operações gerais para trabalhar com o blob, incluindo operações para carregar, baixar, excluir e criar instantâneos.
AppendBlobClient	Representa um blob de acréscimo e fornece operações específicas para acrescentar blobs, como acrescentar dados de log.
BlockBlobClient	Representa um blob de blocos e fornece operações específicas para blobs de blocos, como a preparação e a confirmação de blocos de dados.

Exemplos de implantação das classes:

→ BlobServiceClient:

```
using Azure.Identity;
using Azure.Storage.Blobs;

public BlobServiceclient GetBlobServiceclient(string accountName)
{
    BlobServiceclient client = new(
        new Uri($"https://'{accountName}'.blob.core.windows.net"),
        new DefaultAzureCredential());
    
    return client;
}
```

Nesse exemplo é usado a classe DefaultAzureCredential pois se imagina que o serviço seja implementado no Azure, dessa forma essa classe utiliza uma

função RBAC para obter um Token na conta de armazenamento em tempo de execução, caso contrário deve-se setar um token da conta.

➔ **BlobContainerClient:**

```
public BlobContainerClient GetBlobContainerClient(
    BlobServiceClient blobServiceClient,
    string containerName)
{
    // Create the container client using the service client object
    BlobContainerClient client = blobServiceClient.GetBlobContainerClient(containerName);
    return client;
}
```

Ou

```
public BlobContainerClient GetBlobContainerClient(
    string accountName,
    string containerName,
    BlobClientOptions clientOptions)
{
    // Append the container name to the end of the URI
    BlobContainerClient client = new(
        new Uri($"https://'{accountName}'.blob.core.windows.net/{containerName}"),
        new DefaultAzureCredential(),
        clientOptions);

    return client;
}
```

➔ **BlobClient:**

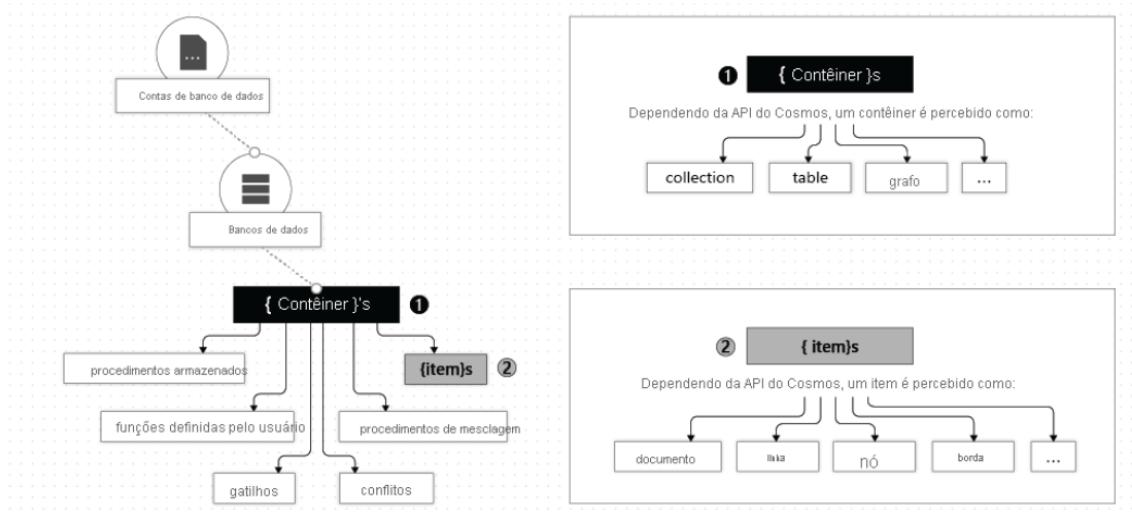
```
public BlobClient GetBlobClient(
    BlobServiceClient blobServiceClient,
    string containerName,
    string blobName)
{
    BlobClient client =
        blobServiceClient.GetBlobContainerClient(containerName).GetBlobClient(blobName);
    return client;
}
```

Azure Cosmos DB

O Azure Cosmos DB é um SGBD NoSQL globalmente distribuído que permite que você leia e grave dados de réplicas locais do seu banco de dados e ele os replica de modo transparente para todas as regiões associadas à sua conta do Cosmos.

você poderá adicionar ou remover as regiões associadas à sua conta a qualquer momento. Seu aplicativo não precisa ser pausado ou reimplementado para adicionar ou remover uma região.

1- Hierarquia de recursos:



A **Conta do Azure Cosmos DB** é a unidade fundamental da distribuição global, a conta contém um DNS exclusivo, é possível gerenciar ela através do portal, do CLI ou das SDKs.

O **Banco de dados** é análogo a um namespace, servindo para agrupar e gerenciar um conjunto de contêiners.

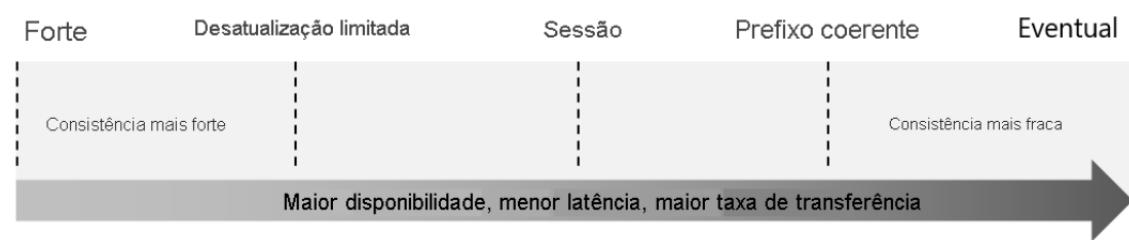
Um **Contêiner do Azure Cosmos DB** é a unidade de escalabilidade para a taxa de transferência provisionada e para o armazenamento. Um contêiner é particionado horizontalmente e, em seguida, é replicado em várias regiões. Os itens que você adiciona ao contêiner são agrupados automaticamente em partições lógicas, que são distribuídas entre partições físicas, com base na chave de partição.

Um item do Azure Cosmos DB pode representar um documento em uma coleção, uma linha em uma tabela ou um nó ou a borda em um grafo.

2- Níveis de consistência:

Temos diferentes níveis de consistência de dados no Azure Cosmos DB:

- Forte
- Bounded staleness
- Session
- Prefixo consistente
- Eventual



a. **Nível de consistência padrão:** É possível configurar o nível de consistência padrão na conta do Azure Cosmos DB.

b. **Coerência Forte:** fornece transação atômica de forma que as leituras podem ser feitas de forma simultaneamente, retornam a versão mais recente do item e o cliente nunca vê uma gravação parcial.

c. **Coerência de desatualização limitada:** Nesse padrão podemos escolher uma desatualização específica:

- Número de versões (K) do item
- O intervalo de tempo (T) pelo qual as leituras podem ficar atrás das gravações

d. **Coerência de Sessão:** As leituras em uma sessão do cliente têm garantia de obedecer ao prefixo coerente. Dessa forma pressupõe uma sessão de leitura ou gravação.

e. Coerência de prefixo coerente: Suponha que duas operações de gravação sejam executadas nos documentos Doc 1 e Doc 2, dentro das transações T1 e T2. Quando o cliente faz uma leitura em qualquer réplica, o usuário vê “Doc 1 v1 e Doc 2 v1” ou “Doc 1 v2 e Doc 2 v2”, mas nunca “Doc 1 v1 e Doc 2 v2” ou “Doc 1 v2 e Doc 2 v1” para a mesma operação de leitura ou consulta.

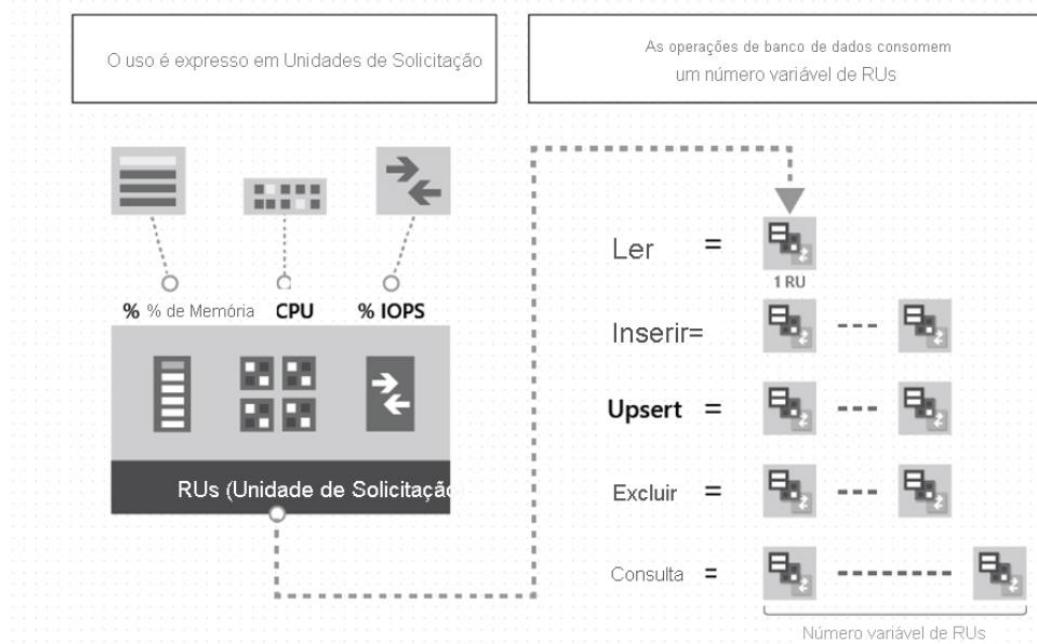
f. Coerência eventual: não há garantia de ordem para leituras. Na ausência de qualquer gravação adicional, as réplicas eventualmente convergem.

3. Unidades de solicitação:

Você paga pela taxa de transferência provisionada e pelo armazenamento que consome por hora.

O custo de todas as operações de banco de dados é normalizado pelo Azure Cosmos DB e é expresso por unidades de solicitação (ou RUs, para abreviar). Uma unidade de solicitação representa os recursos do sistema, como CPU, IOPS e memória, necessários para executar as operações de banco de dados com suporte no Azure Cosmos DB.

O custo para fazer uma leitura pontual, que é buscar um só item por sua ID e valor de chave de partição, para um item de 1 KB é de 1 RU.



O tipo de conta do Azure Cosmos DB determina como as RUs serão cobradas:

. Modo de taxa de transferência provisionada: Você provisiona o número de RUs para o aplicativo em uma base por segundo em incrementos de 100 RUs por segundo.

. **Modo sem servidor:** No final do período de cobrança, você será cobrado pelo número de unidades de solicitação consumidas pelas operações de banco de dados.

. **Modo de dimensionamento automático:** Você pode dimensionar, de modo automático e instantâneo, a taxa de transferência (RU/s) do seu banco de dados ou contêiner conforme o uso.

4- Cosmos DB no .NET:

O pacote usado para integração é o **Microsoft.Azure.Cosmos**, como o cosmos pode ter diversas API's como NoSQL, PostgreSQL etc a versão 3 usa os termos genéricos "contêiner" e "item". Um Container pode ser uma coleção, um grafo ou uma tabela. Um item pode ser um documento, uma borda/vértice ou uma linha.

a. **CosmosClient:** É uma cadeia de conexão com o cosmos DB:

```
CosmosClient client = new CosmosClient(endpoint, key);
```

b. **Banco de dados:** É possível gerenciar os bancos de dados pelo SDK:

➔ Criar banco de dados:

```
// An object containing relevant information about the response
DatabaseResponse databaseResponse = await client.CreateDatabaseIfNotExistsAsync(databaseId, 10000);
```

➔ Ler um banco de dados:

```
DatabaseResponse readResponse = await database.ReadAsync();
```

➔ Deletar banco de dados:

```
await database.DeleteAsync();
```

c. Contêiners:

→ Criar contêiner:

```
// Set throughput to the minimum value of 400 RU/s
ContainerResponse simpleContainer = await database.CreateContainerIfNotExistsAsync(
    id: containerId,
    partitionKeyPath: partitionKey,
    throughput: 400);
```

→ Obter um contêiner pelo Id:

```
Container container = database.GetContainer(containerId);
ContainerProperties containerProperties = await container.ReadContainerAsync();
```

→ Excluir um contêiner:

```
await database.GetContainer(containerId).DeleteContainerAsync();
```

d. Item:

→ Criar um Item:

```
ItemResponse<SalesOrder> response = await
container.CreateItemAsync(salesOrder, new
PartitionKey(salesOrder.AccountNumber));
```

→ Ler um Item:

```
string id = "[id]";
string accountNumber = "[partition-key]";
ItemResponse<SalesOrder> response = await container.ReadItemAsync(id, new PartitionKey(accountNumber));
```

→ Consultar um Item:

```
QueryDefinition query = new QueryDefinition(  
    "select * from sales s where s.AccountNumber = @AccountInput ")  
    .WithParameter("@AccountInput", "Account1");  
  
FeedIterator<SalesOrder> resultSet = container.GetItemQueryIterator<SalesOrder>(  
    query,  
    requestOptions: new QueryRequestOptions()  
    {  
        PartitionKey = new PartitionKey("Account1"),  
        MaxItemCount = 1  
    });
```

e. Procedimentos armazenados: No Cosmos DB é possível criar procedimentos escritos em JavaScript que são armazenados no banco de dados:

```
var helloWorldStoredProcedure = {  
    id: "helloWorld",  
    serverScript: function () {  
        var context = getContext();  
        var response = context.getResponse();  
  
        response.setBody("Hello, World");  
    }  
}
```

JavaScript

```
var createDocumentStoredProcedure = {  
    id: "createMyDocument",  
    body: function createMyDocument(documentToCreate) {  
        var context = getContext();  
        var collection = context.getCollection();  
        var accepted = collection.createDocument(collection.getSelfLink(),  
            documentToCreate,  
            function (err, documentCreated) {  
                if (err) throw new Error('Error' + err.message);  
                context.getResponse().setBody(documentCreated.id)  
            });  
        if (!accepted) return;  
    }  
}
```

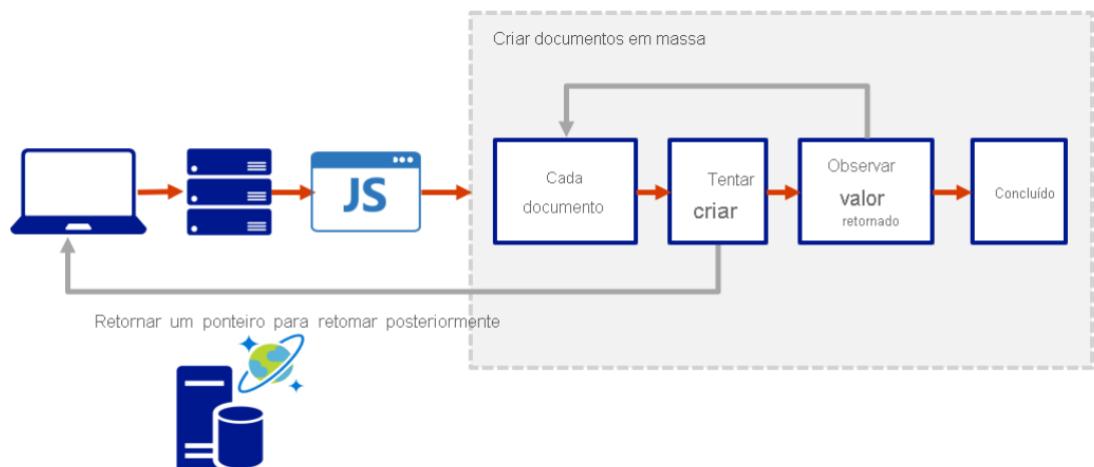
```

function sample(arr) {
    if (typeof arr === "string") arr = JSON.parse(arr);

    arr.forEach(function(a) {
        // do something here
        console.log(a);
    });
}

```

➔ **Funções transacionais:** é possível criar funções transacionais que possuíram o seguinte fluxo:



f. Gatilhos: No Cosmos DB é possível criar Pre-Gatilhos e Pos-Gatilhos que são ações que serão executadas antes e depois da modificação de um item:

```

function validateToDoItemTimestamp() {
    var context = getContext();
    var request = context.getRequest();

    // item to be created in the current operation
    var itemToCreate = request.getBody();

    // validate properties
    if (!("timestamp" in itemToCreate)) {
        var ts = new Date();
        itemToCreate["timestamp"] = ts.getTime();
    }

    // update the item that will be created
    request.setBody(itemToCreate);
}

```

Os Pre-gatilhos não podem receber parâmetros na função.

```

function updateMetadata() {
    var context = getContext();
    var container = context.getCollection();
    var response = context.getResponse();

    // item that was created
    var createdItem = response.getBody();

    // query for metadata document
    var filterQuery = 'SELECT * FROM root r WHERE r.id = "_metadata"';
    var accept = container.queryDocuments(container.getSelfLink(), filterQuery,
        updateMetadataCallback);
    if(!accept) throw "Unable to update metadata, abort";

    function updateMetadataCallback(err, items, responseOptions) {
        if(err) throw new Error("Error" + err.message);
        if(items.length != 1) throw 'Unable to find metadata document';

        var metadataItem = items[0];

        // update metadata
        metadataItem.createdItems += 1;
        metadataItem.createdNames += " " + createdItem.id;
        var accept = container.replaceDocument(metadataItem._self,
            metadataItem, function(err, itemReplaced) {
                if(err) throw "Unable to update metadata, abort";
            });
        if(!accept) throw "Unable to update metadata, abort";
        return;
    }
}

```

Soluções Conteinerizadas

Registro de Contêiner do Azure:

O ACR (Registro de Contêiner do Azure) é um sistema para criar, armazenar e gerenciar imagens de Contêiner no Azure.

É possível criar sobre demanda ou automatizar através de pipelines de outros sistemas como Azure Pipelines ou Jenkins.

1. Camadas de Serviço: A camada de serviço irá determinar padrões de capacidade e preços

Camada	Descrição
Basic	Um ponto de entrada de otimização de custo para desenvolvedores aprendendo sobre o Registro de Contêiner do Azure. Os registros Básicos têm os mesmos recursos de programação que os Standard e Premium (como a integração de autenticação do Microsoft Entra, exclusão de imagens webhooks). No entanto, o armazenamento incluído e a taxa de transferência de imagem são mais apropriados para cenários de uso mais baixos.
Standard	Os registros Standard oferecem os mesmos recursos do Básico, com maior armazenamento incluído e taxa de transferência de imagem. Registros Standard devem atender às necessidades da maioria dos cenários de produção.
Premium	Os registros Premium fornecem a maior quantidade de armazenamento incluído e operações simultâneas, permitindo cenários de alto volume. Além de uma taxa de transferência de imagens mais alta, o Premium adiciona recursos como a replicação geográfica para gerenciar um só registro em várias regiões, a relação de confiança de conteúdo para a assinatura de marca de imagem e o link privado com pontos de extremidade privados para restringir o acesso ao registro.

2. Tarefas ACR: As Tarefas do ACR são um pacote de recursos do Registro de Contêiner do Azure, permite que builds automatizados sejam disparados por atualizações de código-fonte, atualizações da imagem base de um contêiner ou temporizadores.

a. Cenários de tarefas: As Tarefas do ACR são compatíveis com vários cenários para criar e manter imagens de contêiner e outros artefatos.

Tarefa rápida – compile e envie por push uma única imagem de contêiner para um registro de contêiner sob demanda, no Azure, sem a necessidade de uma instalação local do Docker Engine. Pense em docker build, docker

push na nuvem, use o comando **az acr build** no CLI do Azure para fazer o build das imagens.

Tarefas acionadas automaticamente – habilite um ou mais *gatilhos* para criar uma imagem:

- Gatilho na atualização do código-fonte
- Gatilho na atualização da imagem base
- Gatilho em um agendamento

Tarefas de várias etapas – estendem a capacidade de compilação e envio por push de imagem única das Tarefas do ACR com fluxos de trabalho baseados em vários contêineres e várias etapas.

3. Funcionalidades de armazenamento comuns: As diferentes camadas de serviço possuem alguns serviços em comum como **Criptografia em inatividade**, **Armazenamento regional**, **Armazenamento escalonável**.

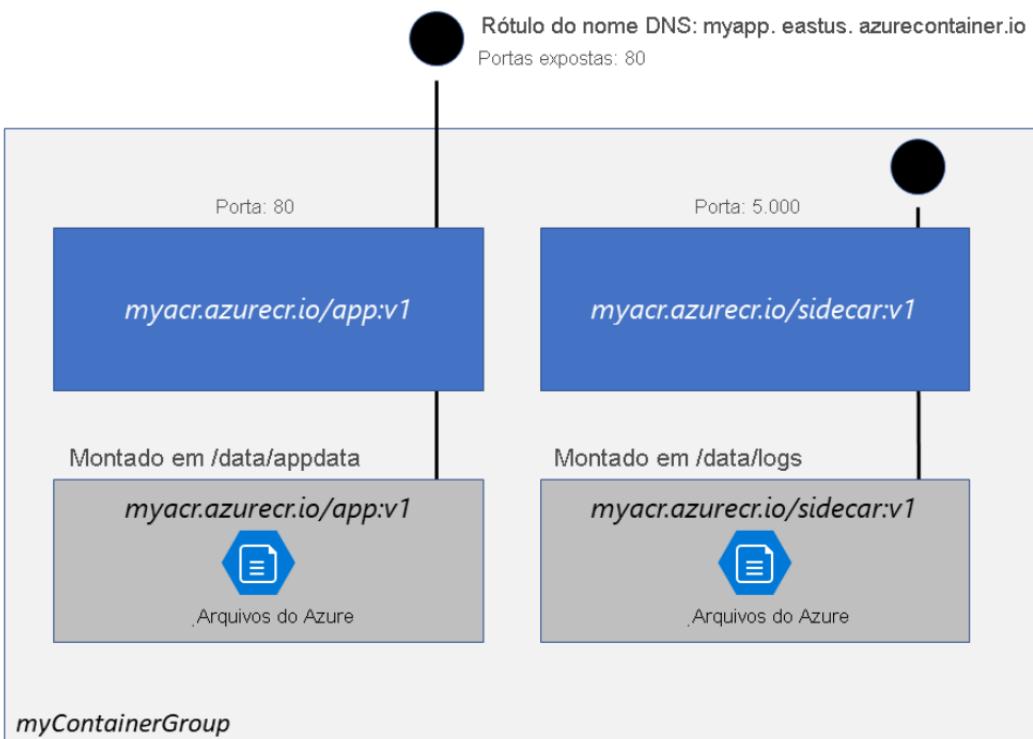
Porém existem algumas diferenças como a **Redundância geográfica** que só ocorre na camada de serviço **Premium**.

Instâncias de Contêiner do Azure:

O ACI é um serviço de execução de contêiners no ambiente Azure não possuindo orquestração de contêiners, para essa aplicação recomendasse o uso do AKS (Azure Kubernetes Service).

O ACI pode gerenciar aplicações Linux e Windows mantendo a mesma api, de forma que não é necessário configurar uma VM e você possui os benefícios que uma execução contêinizada possui.

1. Grupo de Contêineres: O ACI possui um nível de gerenciamento acima chamado de Grupo de Contêineres, onde agrupamos as instâncias no mesmo Host, compartilhando ciclo de vida, recursos, rede local e volumes.



2. Implantação: Existem duas maneiras de implementar os contêineres no ACI, podemos usar os modelos ARM ou através de YAML.

3. Rede: o Grupo de contêineres possui um endereço de IP e um namespace de porta nesse endereço IP. Na rede local os contêineres possuem um network compartilhado entre si e podem se comunicar através do localhost.

4. CLI do Azure: Para criar um container no ACI usamos o seguinte comando **az container create**:

```
az container show --resource-group az204-aci-rg \
--name mycontainer \
--query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningState}" \
--out table
```

Para analisarmos o estado de um contêiner utilizamos o comando **az container show**:

```
az container show --resource-group az204-aci-rg \
--name mycontainer \
--query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningstate}" \
--out table
```

5. Política de reinicialização dos contêiners: No ACI temos 3 possibilidades de política de reinicialização:

Política de reinicialização	Descrição
Always	Os contêineres no grupo de contêineres sempre são reiniciados. Essa é a configuração padrão aplicada quando nenhuma política de reinicialização é especificada na criação do contêiner.
Never	Os contêineres no grupo de contêineres nunca são reiniciados. Os contêineres são executados no máximo uma vez.
OnFailure	Os contêineres no grupo de contêineres são reiniciados somente quando o processo executado no contêiner falha (quando ele termina com um código de saída diferente de zero). Os contêineres são executados pelo menos uma vez.

```
az container create \
--resource-group myResourceGroup \
--name mycontainer \
--image mycontainerimage \
--restart-policy OnFailure
```

6. Variáveis de ambiente: Podemos definir variáveis de ambiente através do comando de criação ou através do yaml, podendo definir entre um **value** que é um valor exposto ou **secretValue** que é um valor segredo exposto apenas no contêiner em tempo de execução:

```
az container create \
--resource-group myResourceGroup \
--name mycontainer2 \
--image mcr.microsoft.com/azuredocs/aci-wordcount:latest
--restart-policy OnFailure \
--environment-variables 'NumWords'='5' 'MinLength'='8'\
```

```
apiVersion: 2018-10-01
location: eastus
name: securetest
properties:
  containers:
    - name: mycontainer
      properties:
        environmentVariables:
          - name: 'NOTSECRET'
            value: 'my-exposed-value'
          - name: 'SECRET'
            secureValue: 'my-secret-value'
      image: nginx
      ports: []
      resources:
        requests:
          cpu: 1.0
          memoryInGB: 1.5
      osType: Linux
      restartPolicy: Always
    tags: null
  type: Microsoft.ContainerInstance/containerGroups
```

```
az container create --resource-group myResourceGroup \
  --file secure-env.yaml \
```

7. Criar um volume através dos arquivos do Azure:

```
az container create \
  --resource-group $ACI_PERS_RESOURCE_GROUP \
  --name hellofiles \
  --image mcr.microsoft.com/azuredocs/aci-hellofiles \
  --dns-name-label aci-demo \
  --ports 80 \
  --azure-file-volume-account-name $ACI_PERS_STORAGE_ACCOUNT_NAME \
  --azure-file-volume-account-key $STORAGE_KEY \
  --azure-file-volume-share-name $ACI_PERS_SHARE_NAME \
  --azure-file-volume-mount-path /aci/logs/
```

```
apiVersion: '2019-12-01'
location: eastus
name: file-share-demo
properties:
  containers:
    - name: hellofiles
      properties:
        environmentVariables: []
        image: mcr.microsoft.com/azuredocs/aci-hellofiles
        ports:
          - port: 80
        resources:
          requests:
            cpu: 1.0
            memoryInGB: 1.5
        volumeMounts:
          - mountPath: /aci/logs/
            name: filesharevolume
  osType: Linux
  restartPolicy: Always
  ipAddress:
    type: Public
    ports:
      - port: 80
    dnsNameLabel: aci-demo
  volumes:
    - name: filesharevolume
      azureFile:
        sharename: acishare
        storageAccountName: <Storage account name>
        storageAccountKey: <Storage account key>
  tags: {}
type: Microsoft.ContainerInstance/containerGroups
```

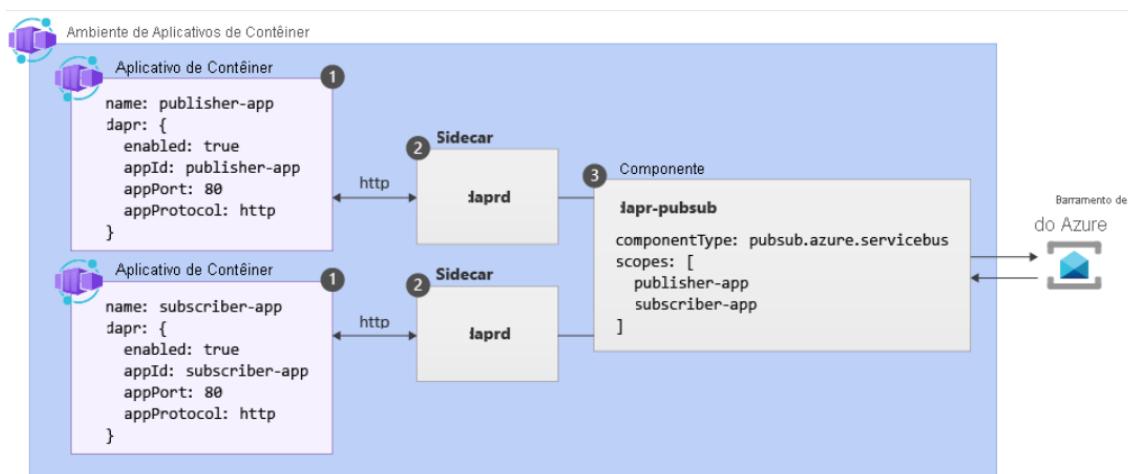
Aplicativos de Contêiner do Azure:

Os Aplicativos de Contêiner do Azure permitem que você execute microsserviços e aplicativos conteinerizados em uma plataforma sem servidor que é executada sobre Serviço de Kubernetes do Azure.

1. Integração com Dapr: O AKS Possui integração nativa com Dapr para gerenciamento dos contêiners dentro do ambiente. O Dapr inclui recursos como observabilidade, Pub/Sub e invocação de serviço com TLS mútua e muito mais.

APIs do Dapr

						
Invocação entre serviços Realize chamadas de método diretas, seguras e entre serviços	Gerenciamento de estado Crie serviços sem estado, com estado e de execução prolongada	Publicar e assinar Mensagens escalonáveis e seguras entre serviços	Associações (entrada/saída) Ação o código por meio de eventos de entradas Associações de entrada e saída para recursos externos, incluindo bancos de dados e filas	Atores Encapsule o código e os dados em objetos de ator reutilizáveis como um padrão de design de microserviços	Observabilidade Veja e meça as chamadas de mensagem em componentes e serviços de rede	Segredos Acesse segredos com segurança do seu aplicativo



2. CLI do Azure: Para implementarmos um serviço de AKS via CLI primeiro precisamos da extensão do AKS no CLI:

```
az extension add --name containerapp --upgrade
```

Os recursos de container do AKS ocorrem no namespace Microsoft.App em vez do Microsoft.Web, por isso precisamos alterar o namespace na implementação:

```
az provider register --namespace Microsoft.App
```

Precisamos registrar o namespace Microsoft.OperationalInsights para o workspace do log Analytics.

```
az provider register --namespace Microsoft.OperationalInsights
```

Após isso definimos as variáveis de ambiente

```
myRG=az204-appcont-rg
myLocation=<location>
myAppContEnv=az204-env-$RANDOM
```

Criamos o grupo de recursos

```
az group create \
--name $myRG \
--location $myLocation
```

Agora criamos o ambiente do containerapp

```
az containerapp env create \
--name $myAppContEnv \
--resource-group $myRG \
--location $myLocation
```

Agora criamos o containerapp

```
az containerapp create \
--name my-container-app \
--resource-group $myRG \
--environment $myAppContEnv \
--image mcr.microsoft.com/azuredocs/containerapps-helloworld:latest \
--target-port 80 \
--ingress 'external' \
--query properties.configuration.ingress.fqdn
```

3. Configuração com Modelo ARM: O código a seguir mostra um exemplo de matriz containers na seção properties.template de um modelo de recursos do AKS:

```

"containers": [
  {
    "name": "main",
    "image": "[parameters('container_image')]",
    "env": [
      {
        "name": "HTTP_PORT",
        "value": "80"
      },
      {
        "name": "SECRET_VAL",
        "secretRef": "mysecret"
      }
    ],
    "resources": {
      "cpu": 0.5,
      "memory": "1Gi"
    },
    "volumeMounts": [
      {
        "mountPath": "/myfiles",
        "volumeName": "azure-files-volume"
      }
    ]
  }
  "probes": [
    {
      "type": "liveness",
      "httpGet": {
        "path": "/health",
        "port": 8080,
        "httpHeaders": [
          {
            "name": "Custom-Header",
            "value": "liveness probe"
          }
        ],
        "initialDelaySeconds": 7,
        "periodSeconds": 3
      }
    }
  ]
}

```

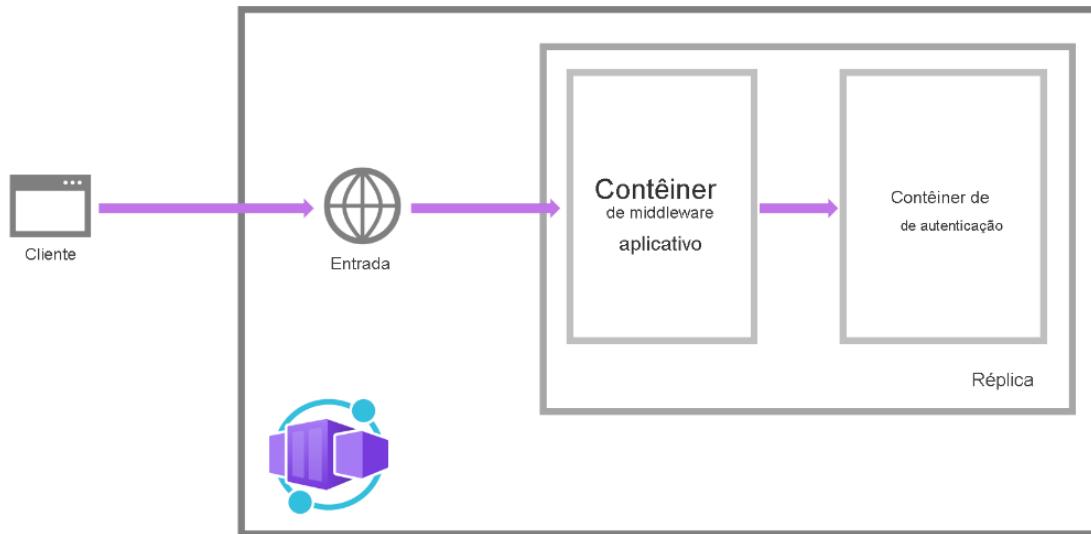
É possível utilizar uma imagem de um provedor privado, porem temos que configurar as credenciais de acesso na seção properties.configuration

```

{
  ...
  "registries": [
    {
      "server": "docker.io",
      "username": "my-registry-user-name",
      "passwordSecretRef": "my-password-secret-name"
    }
  ]
}

```

4. Autenticação e Autorização: O AKS fornece recursos internos para autenticação e autorização. O middleware de autenticação e autorização é um recurso da plataforma que executa como um contêiner sidecar em cada réplica:



A autenticação possui dois fluxos o com SDK do Provedor e o sem SDK do Provedor.

- **SDK do Provedor:** Nesse fluxo o aplicativo conecta os usuários manualmente e envia o token para o AKS para validação.
- **Sem SDK do Provedor:** Nesse fluxo o toda a autenticação é feita pelo provedor.

5. Revisões: No AKS todo aplicativo mantém um controle de versão chamadas de revisões. Cada revisão é imutável em seu estado e para lançar revisões utilizamos o comando az containerapp update:

```
az containerapp update \
--name <APPLICATION_NAME> \
--resource-group <RESOURCE_GROUP_NAME> \
--image <IMAGE_NAME>
```

Podemos listar todas as revisões:

```
az containerapp revision list \
--name <APPLICATION_NAME> \
--resource-group <RESOURCE_GROUP_NAME> \
-o table
```

6. Secrets: No AKS podemos implementar variáveis secretas:

```
az containerapp create \
--resource-group "my-resource-group" \
--name myQueueApp \
--environment "my-environment-name" \
--image demos/myQueueApp:v1 \
--secrets "queue-connection-string=$CONNECTIONSTRING" \
--env-vars "QueueName=myqueue" "ConnectionString=secretref:queue-connection-string"
```

Plataforma de Identidade da Microsoft

Microsoft Identity:

O Microsoft Identity é uma plataforma de controle de usuários usando o padrão de autenticação OAuth 2.0, pode ser gerenciado através do portal do Azure ou através do Microsoft Graph e PowerShell.

1. Entidade de Serviço e entidade de aplicativo: O objeto do aplicativo é a representação global do seu aplicativo a ser usada em todos os locatários e a entidade de serviço é a representação local a ser usada em um locatário específico. O objeto de aplicativo serve como o modelo do qual as propriedades comuns e padrão são derivadas para uso na criação de objetos de entidade de serviço correspondentes.

A entidade do aplicativo pode ser configurada como **Locatário único** ou **Multilocatário**.

A relação entre Entidades e Objetos representa uma estrutura de Orientação a Objetos aplicada a serviços e instâncias de serviços registrados no Microsoft Entra Id.

2. Permissões e consentimento: No OAuth 2.0 os conjuntos de permissões são chamados de **scope**, na plataforma do Microsoft Identity são chamados de permissões e uma permissão é representada com um valor de cadeia de caracteres.

Tipos de permissão: Temos dois tipos de permissão:

Permissão delegada: O aplicativo se loga através de um usuário.

Permissões de acesso somente de aplicativo: Essa permissão ocorre em aplicativos que rodam sem interação de usuário e para gerar essa permissão um administrador tem que aprovar.

Tipos de Consentimento: Todo aplicativo necessita de consentimento para acessar as APIs e para isso temos 3 tipos:

Consentimento do Usuário estático: Nesse padrão o administrador registra todos os consentimentos necessários para acessar e caso o usuário não tenha consentido com algum dele durante a execução o aplicativo irá pedir o consentimento do mesmo.

Consentimento incremental e dinâmico do usuário: Nesse padrão o aplicativo controla os consentimentos através dos **escopos** e o usuário vai dinamicamente consentindo com os **escopos**.

Consentimento do administrador: Esse padrão é usado quando o usuário precisa de acesso a permissões de alto-privilégio, esse tipo de padrão ainda exige permissões estáticas registradas para o aplicativo.

Solicitação de consentimento: a solicitação de consentimento é feita através de uma API Get:

```
GET https://login.microsoftonline.com/common/oauth2/v2.0/authorize?  
client_id=6731de76-14a6-49ae-97bc-6eba6914391e  
&response_type=code  
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F  
&response_mode=query  
&scope=  
https%3A%2F%2Fgraph.microsoft.com%2Fcalendars.read%20  
https%3A%2F%2Fgraph.microsoft.com%2Fmail.send  
&state=12345
```

3. Acesso Condisional: O Microsoft Identity oferece suporte a acesso condicional podendo ser usado para os seguintes cenários:

- [Autenticação multifator](#).
- Permissão para que somente dispositivos inscritos no Intune acessem serviços específicos.
- Restrição de locais de usuário e intervalos de IP.

Implementação da biblioteca de autenticação:

A MSAL pode ser usada para fornecer acesso seguro ao Microsoft Graph e fornece suporte fácil para a obtenção de tokens.

1. Aplicativos cliente públicos e confidenciais: Normalmente dividem em dois tipos os aplicativos:

. **Cliente público:** São aplicativos executados em dispositivos ou computadores desktop ou em um navegador da web.

. **Cliente confidenciais:** São aplicativos executados em servidores.

2. Como utilizar as bibliotecas:

Utilizando o MSAL.NET 3.x utilizamos a classe **PublicClientApplicationBuilder** e **ConfidentialClientApplicationBuilder**.

```
IPublicClientApplication app = PublicClientApplicationBuilder.Create(clientId).Build();  
  
string redirectUri = "https://myapp.azurewebsites.net";  
IConfidentialClientApplication app = ConfidentialClientApplicationBuilder.Create(clientId)  
    .WithClientSecret(clientSecret)  
    .WithRedirectUri(redirectUri )  
    .Build();
```

3. Acesso Compartilhado: O SAS é um token de acesso compartilhado para recursos de armazenamento. Possui 3 tipos:

SAS de delegação de usuário: Usa o Microsoft Entra e permissões específicas se aplica somente ao armazenamento de blobs.

SAS de Serviço: Usa uma chave do Storage Account, delega acesso a: armazenamento de blobs, armazenamento de filas e armazenamento de arquivos.

SAS de conta: Usa uma chave do Storage Account e delega acesso a recursos em um ou mais serviços de armazenamento.

URI de Acesso com Token:

Em um único URI, como `https://medicalrecords.blob.core.windows.net/patient-images/patient-116139-nq8z7f.jpg?sp=r&st=2020-01-20T11:42:32Z&se=2020-01-20T19:42:32Z&spr=https&sv=2019-02-`

- **URI:** `https://medicalrecords.blob.core.windows.net/patient-images/patient-116139-nq8z7f.jpg?`
- **Token SAS:** `sp=r&st=2020-01-20T11:42:32Z&se=2020-01-20T19:42:32Z&spr=https&sv=2019-02-02&sr=b&sig=SrW1HZ5Nb6MbRzTbXCaPm%2BJiSEn15tC91Y4umMPwVZs%3D`

4. Políticas de acesso armazenadas: É um nível de controle adicional sobre o SAS, dessa forma você agrupa as SAS e fornece restrições adicionais podendo alterar a hora de início, a hora de expiração e as permissões de uma assinatura.

```

BlobSignedIdentifier identifier = new BlobSignedIdentifier
{
    Id = "stored access policy identifier",
    AccessPolicy = new BlobAccessPolicy
    {
        ExpiresOn = DateTimeOffset.UtcNow.AddHours(1),
        Permissions = "rw"
    }
};

blobContainer.SetAccessPolicy(permissions: new BlobSignedIdentifier[] { identifier });

```

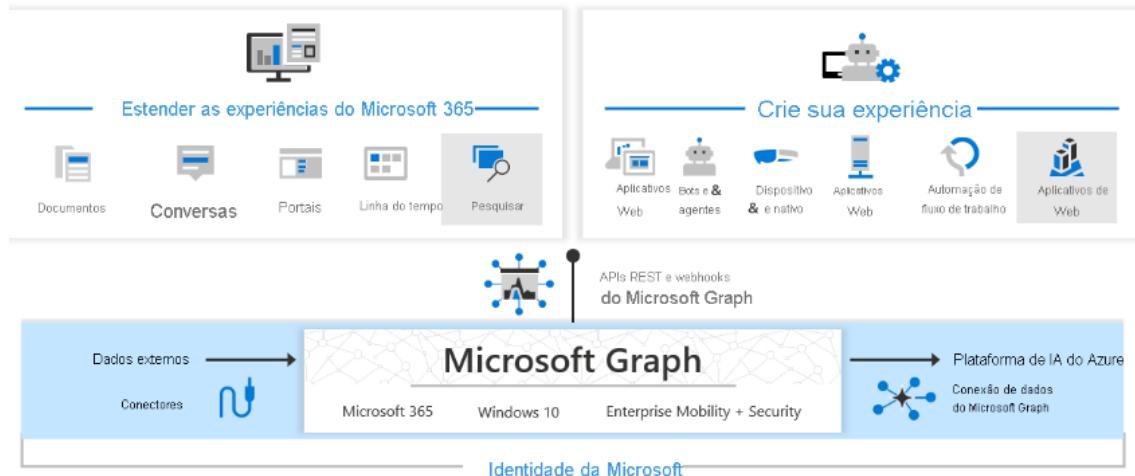
```

az storage container policy create \
--name <stored access policy identifier> \
--container-name <container name> \
--start <start time UTC datetime> \
--expiry <expiry time UTC datetime> \
--permissions <(a)dd, (c)reate, (d)elete, (l)ist, (r)ead, or (w)rite> \
--account-key <storage account key> \
--account-name <storage account name> \

```

Microsoft Graph: O Microsoft Graph é o gateway para dados e inteligência em Microsoft 365. Ele fornece um modelo de programação unificado que você pode usar para acessar a enorme quantidade de dados em Microsoft 365.

Plataforma Microsoft 365



1. Consultar utilizando REST:

```
HTTP {HTTP method} https://graph.microsoft.com/{version}/{resource}?{query-parameters}
```

Os componentes de uma solicitação incluem:

- {HTTP method} - o método HTTP usado na solicitação para o Microsoft Graph.
- {version} - a versão da API do Microsoft Graph que seu aplicativo está usando.
- {resource} - o recurso no Microsoft Graph ao qual você está fazendo referência.
- {query-parameters} - opções de consulta OData opcionais ou parâmetros do método REST que personalizam a resposta.

```
HTTP GET https://graph.microsoft.com/v1.0/me/messages?filter=emailAddress eq 'jon@contoso.com'
```

2. Consultar utilizando SDKs:

. **Bibliotecas:** O SDK está presente nas seguintes bibliotecas:

- [Microsoft.Graph](#) - contém os modelos e construtores de solicitação para acessar o ponto de extremidade `v1.0` com a API fluente. O `Microsoft.Graph` depende do `Microsoft.Graph.Core`.
- [Microsoft.Graph.Beta](#) - contém os modelos e construtores de solicitação para acessar o ponto de extremidade `beta` com a API fluente. O `Microsoft.Graph.Beta` depende do `Microsoft.Graph.Core`.
- [Microsoft.Graph.Core](#) - a biblioteca principal para fazer chamadas para o Microsoft Graph.

. **Crie um cliente:** O cliente foi desenvolvido para simplificar a realização de chamadas para o Microsoft Graph:

```

var scopes = new[] { "User.Read" };

// Multi-tenant apps can use "common",
// single-tenant apps must use the tenant ID from the Azure portal
var tenantId = "common";

// Value from app registration
var clientId = "YOUR_CLIENT_ID";

// using Azure.Identity;
var options = new TokenCredentialOptions
{
    AuthorityHost = AzureAuthorityHosts.AzurePublicCloud
};

// Callback function that receives the user prompt
// Prompt contains the generated device code that you must
// enter during the auth process in the browser
Func<DeviceCodeInfo, CancellationToken, Task> callback = (code, cancellation) => {
    Console.WriteLine(code.Message);
    return Task.FromResult(0);
};

// /dotnet/api/azure.identity.devicecodecredential
var deviceCodeCredential = new DeviceCodeCredential(
    callback, tenantId, clientId, options);

var graphClient = new GraphServiceClient(deviceCodeCredential, scopes);

```

. Ler informações do Microsoft Graph:

```

// GET https://graph.microsoft.com/v1.0/me

var user = await graphClient.Me
    .GetAsync();

```

. Recuperar uma lista de entidades:

```

var messages = await graphClient.Me.Messages
    .Request()
    .Select(m => new {
        m.Subject,
        m.Sender
    })
    .Filter("<filter condition>")
    .OrderBy("receivedDateTime")
    .GetAsync();

```

. Excluir uma entidade:

```

string messageId = "AQMrAGUy...";
var message = await graphClient.Me.Messages[messageId]
    .Request()
    .DeleteAsync();

```

. Criar uma nova entidade:

```
var calendar = new Calendar
{
    Name = "Volunteer"
};

var newCalendar = await graphClient.Me Calendars
    .Request()
    .AddAsync(calendar);
```

Soluções seguras do Azure

Azure Key Vault: O Azure Key vault oferece dois tipos de armazenamento de chaves: Cofres e pools de HSM (Modulo de Segurança de Hardware).

O Azure Key Vault oferece dois tipos de camadas **Standart** que oferece criptografia através de uma chave de software e **Premium** que inclui chaves protegidas por HSM.

1. Autenticação: A autenticação acontece através do Microsoft Entra e existem 3 maneiras de se autenticar com o Azure Key Vault:

- . **Identidades gerenciadas para recursos do Azure:** com uma VM é possível atribuir uma identidade a VM, é possível atribuir também identidades a outros serviços.
- . **Entidade de serviço e certificado:** Pode usar uma identidade que tem acesso e um certificado que tem acesso ao Azure Key Vault.
- . **Entidade de serviço e segredo:** Usa-se uma entidade de serviço e um segredo para se autenticar.

2. Criptografia: O Azure Key Vault utiliza criptografia **TLS** durante o tráfego com o cliente a conexão é protegida por **PFS** com chave de criptografia de 2.048 bits baseados em RSA.

3. Autenticação utilizando REST: Tokens de acesso devem ser enviados para o serviço com esse cabeçalho:

```
PUT /keys/MYKEY?api-version=<api_version>  HTTP/1.1
Authorization: Bearer <access_token>
```

Os parâmetros no cabeçalho `WWW-Authenticate` são:

- authorization: O endereço do serviço de autorização OAuth2 que pode ser usado para obter um token de acesso para a solicitação.
- resource: O nome do recurso (<https://vault.azure.net>) a ser usado na solicitação de autorização.

4. Criando através do Cloud Shell:

. Criar o cofre:

```
Bash Copiar
myKeyVault=az204vault-$RANDOM
myLocation=<myLocation>
```

2. Crie um grupos de recursos.

```
CLI do Azure Copiar
az group create --name az204-vault-rg --location $myLocation
```

3. Crie um Key Vault usando o comando `az keyvault create`.

```
CLI do Azure Copiar
az keyvault create --name $myKeyVault --resource-group az204-vault-rg --location $myLocation
```

. Adicionar e recuperar um segredo:

1. Criar um segredo. Vamos adicionar uma senha que pode ser usada por um aplicativo. A senha é chamada de `ExamplePassword` e armazenará o valor `hVFkk965BuUv` nela.

```
CLI do Azure Copiar
az keyvault secret set --vault-name $myKeyVault --name "ExamplePassword" --value "hVFkk965BuUv"
```

2. Use o comando `az keyvault secret show` para recuperar o segredo.

```
CLI do Azure Copiar
az keyvault secret show --name "ExamplePassword" --vault-name $myKeyVault
```

Este comando retorna um JSON. A última linha contém a senha em texto sem formatação.

```
JSON Copiar
"value": "hVFkk965BuUv"
```

Identidades gerenciadas: As identidades gerenciadas eliminam a necessidade de os desenvolvedores gerenciarem as credenciais usadas para proteger a comunicação entre os serviços.

1. Tipos de identidades gerenciadas:

. **Identidade gerenciada atribuída pelo sistema:** o Azure cria uma identidade para a instância no locatário do Microsoft Entra que é confiável pela assinatura da instância. O ciclo de vida de uma identidade atribuída ao

sistema está diretamente relacionado à instância de serviço do Azure na qual ela está habilitada.

. **Identidade gerenciada atribuída pelo usuário:** é criada como um recurso autônomo do Azure. Através de um processo de criação, ela pode ser atribuída a uma ou mais instâncias de serviço do Azure. O ciclo de vida é independente dos serviços.

2. Criar e implementar identidades gerenciadas através do CLI:

. **Criar atribuída pelo sistema durante a criação de recursos:**

```
az vm create --resource-group myResourceGroup \
  --name myVM --image win2016datacenter \
  --generate-ssh-keys \
  --assign-identity \
  --role contributor \
  --scope mySubscription \
  --admin-username azureuser \
  --admin-password myPassword12
```

. **Atribuir identidade gerenciada pelo sistema em recursos existentes:**

```
az vm identity assign -g myResourceGroup -n myVm
```

. **Criar uma identidade gerenciada pelo usuário:**

```
az identity create -g myResourceGroup -n myUserAssignedIdentity
```

. **Atribuir identidade gerenciada pelo usuário durante a criação de um recurso:**

```
az vm create \
  --resource-group <RESOURCE GROUP> \
  --name <VM NAME> \
  --image Ubuntu2204 \
  --admin-username <USER NAME> \
  --admin-password <PASSWORD> \
  --assign-identity <USER ASSIGNED IDENTITY NAME> \
  --role <ROLE> \
  --scope <SUBSCRIPTION>
```

. **Atribuir uma identidade gerenciada pelo usuário em um recurso já existente:**

```
az vm identity assign \
  -g <RESOURCE GROUP> \
  -n <VM NAME> \
  --identities <USER ASSIGNED IDENTITY>
```

3. Adquirir token de acesso pelo sdk: A classe **DefaultAzureCredential** tenta autenticar automaticamente por meio de vários mecanismos:

1. **Ambiente** – a `DefaultAzureCredential` lê as informações de conta especificadas por meio de variáveis de ambiente, usando-as para autenticação.
2. **Identidade gerenciada** – se o aplicativo for implantado em um host do Azure com identidade gerenciada habilitada, o `DefaultAzureCredential` será autenticado com essa conta.
3. **Visual Studio** se o desenvolvedor tiver se autenticado por meio do Visual Studio, o `DefaultAzureCredential` será autenticado com essa conta.
4. **CLI do Azure** – se o desenvolvedor tiver autenticado uma conta por meio do comando `az login` da CLI do Azure, `DefaultAzureCredential` será autenticado com essa conta. Os usuários do Visual Studio Code podem autenticar seu ambiente de desenvolvimento usando a CLI do Azure.
5. **Azure PowerShell** – se o desenvolvedor tiver autenticado uma conta por meio do comando `Connect-AzAccount` do Azure PowerShell, `DefaultAzureCredential` será autenticado com essa conta.
6. **Navegador interativo** – se habilitado, a `DefaultAzureCredential` autenticará interativamente o desenvolvedor por meio do navegador padrão do sistema atual. Por padrão, esse tipo de credencial está desabilitado.

```
// Create a secret client using the DefaultAzureCredential
var client = new SecretClient(new Uri("https://myvault.vault.azure.net/"), new DefaultAzureCredential());
```



```
string userAssignedClientId = "<your managed identity client Id>";
var credential = new DefaultAzureCredential(new DefaultAzureCredentialOptions { ManagedIdentityClientId = userAssignedClientId });

var blobClient = new BlobClient(new Uri("https://myaccount.blob.core.windows.net/mycontainer/myblob"), credential);
```

A classe **ChainedTokenCredential** permite que os usuários combinem várias instâncias de credencial:

```
// Authenticate using managed identity if it is available; otherwise use the Azure CLI to authenticate.
var credential = new ChainedTokenCredential(new ManagedIdentityCredential(), new AzureCliCredential());

var eventHubProducerClient = new EventHubProducerClient("myeventhub.eventhubs.windows.net", "myhubpath", credential);
```

Configuração de Aplicativos do Azure: Fornece um serviço para gerenciar centralmente as configurações de aplicativo e os sinalizadores de recurso. A Configuração de Aplicativo complementa o Azure Key Vault.

1. Adicionar configuração de aplicativo via sdk no .NET:

Microsoft.Extensions.Configuration.AzureApp Configuration Namespace

Referência

 Comentários

Classes

 Expandir a tabela

AzureAppConfiguration	Opções usadas para configurar o cliente usado para buscar referências do cofre de chaves em um provedor de Configuração de Aplicativos do Azure.
AzureAppConfigurationOptions	Opções usadas para configurar o comportamento de um provedor de Configuração de Aplicativos do Azure.
AzureAppConfigurationRefreshOptions	Opções usadas para configurar o comportamento de atualização de um provedor de Configuração de Aplicativos do Azure.
KeyFilter	Define filtros de chave bem conhecidos que são usados em Configuração de Aplicativos do Azure.
KeyVaultReferenceException	A exceção gerada quando há um erro ao resolver uma referência ao recurso de Key Vault do Azure.
LabelFilter	Define filtros de rótulo bem conhecidos que são usados em Configuração de Aplicativos do Azure.
PushNotification	Um objeto que contém os detalhes de uma notificação por push recebida do serviço Configuração de Aplicativos do Azure.

Interfaces

 Expandir a tabela

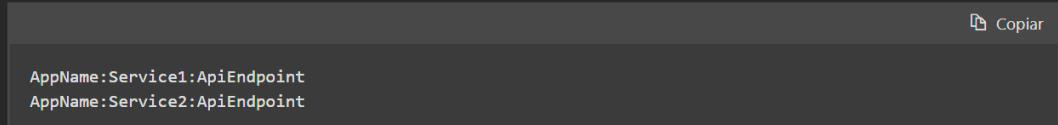
IConfigurationRefresher	Uma interface usada para disparar uma atualização para os dados registrados para atualização com Configuração de Aplicativos.
IConfigurationRefresherProvider	Uma interface usada para recuperar instâncias de atualização para Configuração de Aplicativos.

2. Criar pares de chaves e valores: Podemos criar pares de chaves e valores no gerenciamento de aplicativos do Azure, para isso podemos seguir alguns padrões:

. **Simples:** Não possui regra de nomenclatura e cada caso pode variar.

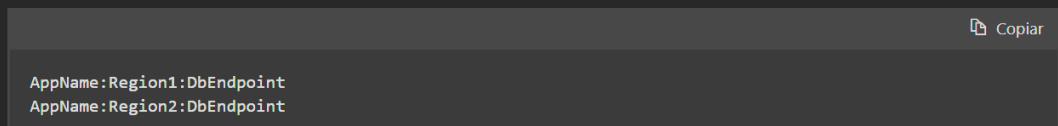
. **Hierárquica:** Possui regras de nomenclatura baseadas em hierarquia.

- Com base em serviços de componentes



```
AppName:Service1:ApiEndpoint
AppName:Service2:ApiEndpoint
```

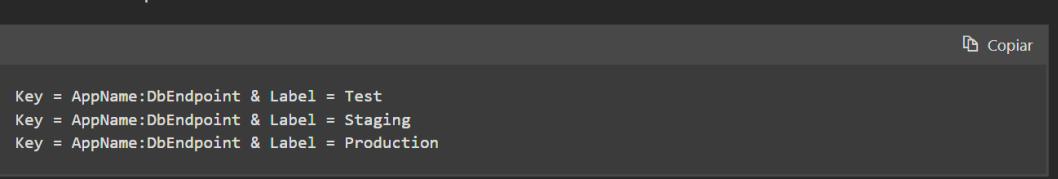
- Com base em regiões de implantação



```
AppName:Region1:DbEndpoint
AppName:Region2:DbEndpoint
```

. **Chaves de rótulo:** Valores de chave na configuração de aplicativo podem, opcionalmente ter um atributo de rótulo.

O rótulo fornece uma maneira conveniente de criar variantes de uma chave. Um uso comum dos rótulos é especificar vários ambientes para a mesma chave:



```
Key = AppName:DbEndpoint & Label = Test
Key = AppName:DbEndpoint & Label = Staging
Key = AppName:DbEndpoint & Label = Production
```

3. Gerenciar recursos do aplicativo: O gerenciamento de recursos é uma prática moderna de desenvolvimento de software que separa a liberação do recurso da implantação do código.

Conceitos básicos

Veja os vários novos termos relacionados ao gerenciamento de recursos:

- **Sinalizador de recurso:** Um sinalizador de recurso é uma variável com um estado binário igual a *ativado* ou *desativado*. O sinalizador de recurso também tem um bloco de código associado. O estado do sinalizador de recursos decide se o bloco de código será executado ou não.
- **Gerenciador de recursos:** é um pacote de aplicativos que processa o ciclo de vida de todos os sinalizadores de recurso em um aplicativo. O gerenciador de recursos normalmente fornece mais funcionalidades, como o cache de sinalizadores de recursos e a atualização dos estados deles.
- **Filtro:** Um filtro é uma regra para avaliar o estado de um sinalizador de recurso. Um grupo de usuários, um tipo de dispositivo ou navegador, uma geolocalização e uma janela de tempo são todos exemplos do que pode representar um filtro.

Cada sinalizador de recurso tem duas partes: um nome e uma lista de um ou mais filtros que são usados para avaliar se o estado de um recurso é ativado (ou seja, quando seu valor é True). Um filtro define um caso de uso para os casos em que um recurso deve ser ligado.

O gerenciador de recursos dá suporte ao appsettings.json como uma fonte de configuração para sinalizadores de recursos.

```
"FeatureManagement": {
    "FeatureA": true, // Feature flag set to on
    "FeatureB": false, // Feature flag set to off
    "FeatureC": {
        "EnabledFor": [
            {
                "Name": "Percentage",
                "Parameters": {
                    "Value": 50
                }
            }
        ]
    }
}
```

4. Proteger dados de configuração de aplicativo:

Habilitar a funcionalidade de chave gerenciada pelo cliente

Os seguintes componentes são necessários para habilitar com êxito a capacidade de chave gerenciada pelo cliente para a Configuração de Aplicativos do Azure:

- Instância da Configuração de Aplicativos do Azure Camada Standard
- Azure Key Vault com recursos de exclusão reversível e de proteção de limpeza habilitados
- Uma chave RSA ou RSA-HSM dentro do Key Vault: a chave não pode estar expirada, precisa estar habilitada e ter funcionalidades de quebrar linha e cancelar quebra de linha habilitadas

Depois que esses recursos forem configurados, restarão duas etapas para permitir que a configuração de aplicativos do Azure use a chave de Key Vault:

1. Atribuir uma identidade gerenciada à instância de Configuração de Aplicativos do Azure
2. Conceda a identidade `GET`, `WRAP` e `UNWRAP` permissões na política de acesso do Key Vault de destino.

Usar pontos de extremidade privados para a Configuração de Aplicativos do Azure

Você pode usar pontos de extremidade privados para a Configuração de Aplicativos do Azure a fim de permitir que os clientes em uma VNet (rede virtual) acessem dados com segurança por meio de um link privado. O ponto de extremidade privado usa um endereço IP do espaço de endereço da VNet para seu repositório de Configuração de Aplicativos. O tráfego de rede entre a máquina em sua VNet e o repositório de Configuração de Aplicativos atravessa a VNet usando um link privado na rede principal da Microsoft, eliminando a exposição à Internet pública.

Gerenciamento de API

O Gerenciamento de API fornece a funcionalidade principal para garantir um programa de API bem-sucedido por meio do envolvimento do desenvolvedor, insights de negócios, análise, segurança e proteção.

1. Componentes do Gerenciamento de API: O Gerenciamento de API do Azure é composto por um gateway de API, um plano de gerenciamento e um portal do desenvolvedor.

- O **gateway de API** é o ponto de extremidade que:
 - Aceita chamadas à API e as direciona para o back-end apropriado
 - Verifica chaves de API e outras credenciais apresentadas com solicitações
 - Impõe o uso de cotas e limites de taxa
 - Transforma solicitações e respostas especificadas em instruções de política
 - Armazena em cache as respostas para melhorar a latência de resposta e minimizar a carga nos serviços de back-end
 - Emite logs, métricas e rastreamentos para monitoramento, relatórios e solução de problemas
- O **plano de gerenciamento** é a interface administrativa na qual você configura seu programa de API. Use-o para:
 - Provisionar e definir o Gerenciamento de API de configurações de serviço
 - Definir ou importar esquemas de API
 - Empacotar APIs em produtos
 - Políticas de configuração como cotas ou transformações em APIs
 - Obter insights por meio de análises
 - Gerenciar usuários
- O **portal do desenvolvedor** é um site gerado automaticamente e totalmente personalizável com a documentação das suas APIs. Ao usar o portal do desenvolvedor, os desenvolvedores podem:
 - Ler a documentação da API
 - Experimentar uma API por meio do console interativo
 - Criar uma conta e fazer uma assinatura para obter chaves de API
 - Acessar a análise do seu próprio uso.
 - Baixar definições de API
 - Gerenciar chaves de API

2. Conceitos do Gerenciamento de API:

. Produtos: Os produtos são como as APIs são exibidas para os desenvolvedores. Os produtos têm Título, Descrição e termos de uso e podem ser **abertos** ou **protegidos**.

. Grupos: Os grupos são usados para gerenciar a visibilidade dos produtos para os desenvolvedores.

- **Administradores** – Gerenciam instâncias de serviço de Gerenciamento de API e criam as APIs, operações e produtos que são usados pelos desenvolvedores. Os administradores de assinatura do Azure são membros desse grupo.
- **Desenvolvedores** – usuários autenticados do portal do desenvolvedor que cria aplicativos usando suas APIs. Os desenvolvedores têm acesso ao portal do desenvolvedor e criam aplicativos que chamam as operações de uma API.
- **Convidados** – Usuários do portal do desenvolvedor não autenticados. Eles podem receber certos acessos somente leitura, como a capacidade de exibir APIs, mas não de chamá-las.

. Desenvolvedores: Os desenvolvedores representam as contas de usuários em uma instância de serviço de Gerenciamento de API. Pode ser criado ou convidado e pertence a um ou mais grupos.

. Políticas: As políticas são um conjunto de instruções executadas em sequência, na solicitação ou na resposta de uma API.

3. Gateway da api: O gateway de Gerenciamento de API (também chamado de plano de dados ou runtime) é o componente de serviço responsável por fazer proxy de solicitações de API, aplicar políticas e coletar telemetria.

Um gateway de API fica entre clientes e serviços. Ele atua como um proxy reverso, encaminhando as solicitações de clientes para serviços.

- **Gerenciado:** o gateway gerenciado é o componente de gateway padrão implantado no Azure para cada instância do Gerenciamento de API em cada camada de serviço. Com o gateway gerenciado, todo o tráfego de API flui pelo Azure, independentemente do local em que os back-ends que implementam as APIs estão hospedados.
- **Auto-hospedado:** o gateway auto-hospedado é uma versão opcional e em contêiner do gateway gerenciado padrão. Ele é útil para cenários híbridos e de várias nuvens em que há um requisito para executar os gateways do Azure nos mesmos ambientes em que os back-ends da API estão hospedados. O gateway auto-hospedado permite que os clientes com infraestrutura de TI híbrida gerenciem APIs hospedadas localmente e entre nuvens de um único serviço de Gerenciamento de API do Azure.

4. Políticas do Gerenciamento de API: As políticas são comportamentos aplicados na requisição ou na resposta da api e fica no gateway da API.

A definição da política é um documento **XML** simples que descreve uma sequência de instruções de entrada e de saída. O **XML** pode ser editado diretamente na janela de definição.

A configuração é dividida em **inbound**, **backend**, **outbound** e **on-error**:

```

<policies>
  <inbound>
    <!-- statements to be applied to the request go here -->
  </inbound>
  <backend>
    <!-- statements to be applied before the request is forwarded to
        the backend service go here -->
  </backend>
  <outbound>
    <!-- statements to be applied to the response go here -->
  </outbound>
  <on-error>
    <!-- statements to be applied if there is an error condition go here -->
  </on-error>
</policies>

```

- uma única instrução C# em `@(expression)` ou
- um bloco de códigos C# de várias instruções, em `@{expression}` que retorna um valor

Cada expressão tem acesso à variável `context` fornecida implicitamente e a um subconjunto permitido de tipos de .NET Framework.

As expressões podem ser aplicadas em todas as API's através da tag `<base>` ou é um escopo de uma api:

```

<policies>
  <inbound>
    <base />
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
    <choose>
      <when condition="@(context.Response.StatusCode == 200 && context.Product.Name.Equals("Starter"))">
        <!-- NOTE that we are not using preserveContent=true when deserializing response body stream into a JS
        <set-body>
          @{
            var response = context.Response.Body.As< JObject>();
            foreach (var key in new [] {"minutely", "hourly", "daily", "flags"}) {
              response.Property (key).Remove ();
            }
            return response.ToString();
          }
        </set-body>
      </when>
    </choose>
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>

```

5. Políticas complexas: O gerenciamento de API possui alguns suportes de comportamento para as políticas que vão além de alterar o request ou o response são eles:

Fluxo de controle

A política `choose` aplica instruções de política incluídas com base no resultado da avaliação de expressões booleanas, semelhante a uma if-then-else ou uma construção de opção em uma linguagem de programação.

```
XML Copiar

<choose>
    <when condition="Boolean expression | Boolean constant">
        <!-- one or more policy statements to be applied if the above condition is true -->
    </when>
    <when condition="Boolean expression | Boolean constant">
        <!-- one or more policy statements to be applied if the above condition is true -->
    </when>
    <otherwise>
        <!-- one or more policy statements to be applied if none of the above conditions are true -->
    </otherwise>
</choose>
```

Encaminhar solicitação

A política `forward-request` encaminha a solicitação de entrada para o serviço de back-end especificado no contexto da solicitação. A URL do serviço de back-end é especificada nas configurações de API e pode ser alterada usando a política definir o serviço de back-end.

A remoção dessa política resulta na solicitação não ser encaminhada para o serviço de back-end e as políticas na seção de saída serem avaliadas imediatamente após a conclusão bem-sucedida das políticas na seção de entrada.

```
XML Copiar

<forward-request timeout="time in seconds" follow-redirects="true | false"/>
```

Simultaneidade de limite

A política `limit-concurrency` impede que políticas fechadas sejam executadas por mais do que o número especificado de solicitações a qualquer momento. Ao exceder esse número, novas solicitações falharão imediatamente com um código de status 429 *Número Excessivo de Solicitações*.

```
XML Copiar

<limit-concurrency key="expression" max-count="number">
    <!-- nested policy statements -->
</limit-concurrency>
```

Registrar no Hub de Eventos

A política `log-to-eventhub` envia mensagens no formato especificado para um Hub de Eventos definido por uma entidade Logger. Como o nome sugere, a política é usada para salvar informações de contexto de solicitação ou de resposta solicitadas para a análise online ou offline.

```
XML Copiar

<log-to-eventhub logger-id="id of the logger entity" partition-id="index of the partition where messages are sent">
    Expression returning a string to be logged
</log-to-eventhub>
```

Resposta fictícia

O `mock-response`, como o nome indica, é usado para simular APIs e operações. Ele anula a execução normal de pipeline e retorna uma resposta fictícia ao chamador. A política sempre tenta retornar respostas da mais alta fidelidade. Ela prefere exemplos de conteúdo de resposta, sempre que disponíveis. Ela gera respostas de exemplo com base em esquemas, quando esquemas são fornecidos e exemplos não são fornecidos. Se não forem encontrados exemplos nem esquemas, serão retornadas respostas sem conteúdo.

```
XML Copiar
<mock-response status-code="code" content-type="media type"/>
```

Repetir

A política `retry` executa suas políticas filho uma vez e tenta realizar sua execução novamente até `condition` da nova tentativa se tornar `false` ou `count` da nova tentativa ser esgotada.

```
XML Copiar
<retry
  condition="boolean expression or literal"
  count="number of retry attempts"
  interval="retry interval in seconds"
  max-interval="maximum retry interval in seconds"
  delta="retry interval delta in seconds"
  first-fast-retry="boolean expression or literal">
  <!-- One or more child policies. No restrictions -->
</retry>
```

Retornar resposta

A política `return-response` anula a execução do pipeline e retorna uma resposta padrão ou personalizada para o chamador. A resposta padrão é `200 OK` sem corpo. A resposta personalizada pode ser especificada por meio de declarações de política ou variável de contexto. Quando ambas são fornecidas, a resposta contida na variável de contexto é modificada pelas instruções de política antes de ser retornada para o chamador.

```
XML Copiar
<return-response response-variable-name="existing context variable">
  <set-header/>
  <set-body/>
  <set-status/>
</return-response>
```

6. Chamar API com assinatura:

NOME DE EXIBIÇÃO	CHAVE PRIMÁRIA	CHAVE SECUNDÁRIA	SCOPO	ESTADO	PROPRIETÁRIO	PERMITIR RASTREAM...
Acesso total interno...	[REDACTED]	[REDACTED]	Produto: Inicial	Ativo	Administrador	✓
Acesso total interno...	[REDACTED]	[REDACTED]	Produto: ilimitado	Ativo	Administrador	✓
limitada	[REDACTED]	[REDACTED]	Serviço	Ativo		✓
limitada	[REDACTED]	[REDACTED]	Produto: ilimitado	Ativo		...
limitada	[REDACTED]	[REDACTED]	Produto: NorthWind...	Ativo	Administrador	✓

Bash

Copiar

```
curl --header "Ocp-Apim-Subscription-Key: <key string>" https://<apim gateway>.azure-api.net/api/path
```

7. Criar um gerenciamento de API:

Crie uma instância de APIM. O comando `az apim create` é usado para criar a instância. A opção `--sku-name Consumption` é usada para acelerar o processo para o passo a passo.

Bash

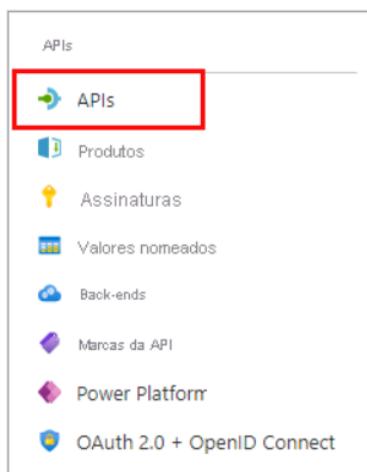
Copiar

```
az apim create -n $myApiName \
--location $myLocation \
--publisher-email $myEmail \
--resource-group az204-apim-rg \
--publisher-name AZ204-APIM-Exercise \
--sku-name Consumption
```

Importar uma API de back-end

Esta seção mostra como importar e publicar uma API de back-end da especificação OpenAPI.

1. No portal do Azure, pesquise e selecione **Serviços de Gerenciamento de API**.
2. Na tela **Gerenciamento de API**, selecione a Instância de Gerenciamento de API que você criou.
3. Selecione **APIs** no painel de navegação do serviço de gerenciamento de API.



4. Selecione OpenAPI na lista e selecione **Completo** no pop-up.

Criação com base na especificação de OpenAPI

Básico **Completo**

* OpenAPI
OpenAPI

* Nome de exibição
API de Conferência de Demonstração

* Nome
demo-conference-api

Descrição
Um exemplo de API com informações relacionadas a uma conferência técnica. Os recursos disponíveis incluem *Palestrantes*, *Sessões* e *Tópicos*. Uma única operação de gravação está disponível para fornecer feedback sobre uma sessão.

Esquema de URL
 HTTP HTTPS Ambos

Sufixo de URL da API
conferência

URL Base
`http(s)://az204-apim-19520.azure-api.net/conference`

Marcas
por exemplo Reserva

Produtos
Nenhum produto selecionado

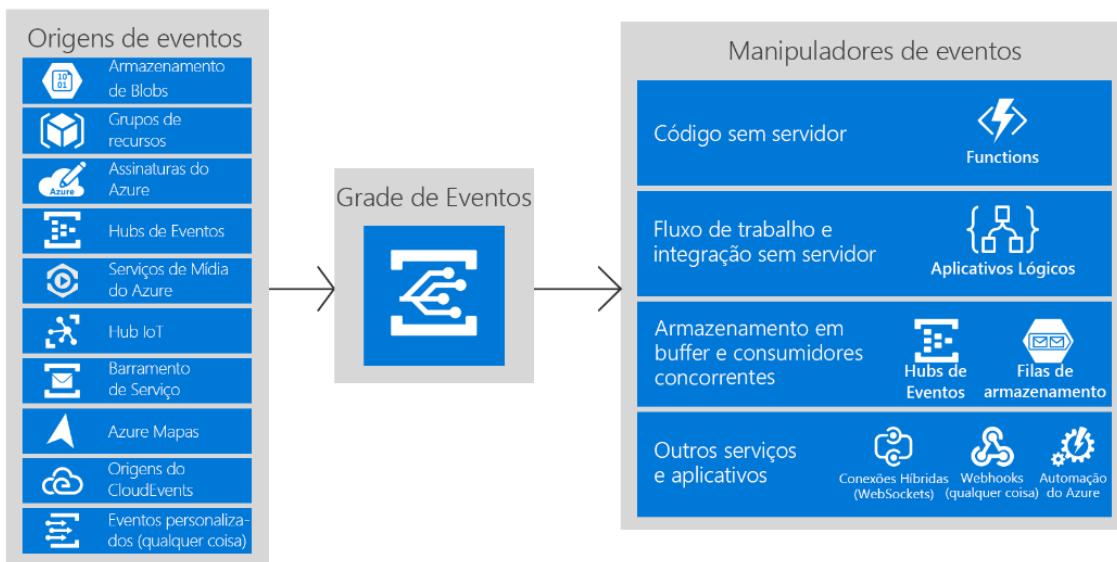
● Para publicar a API, você precisa associá-la a um produto. Saiba mais.

Versão desta API?

Criar **Cancelar**

Eventos do Azure

Grade de eventos do Azure: A Grade de Eventos do Azure é um agente de eventos sem servidor que você pode usar para integrar aplicativos usando eventos, os eventos são entregues pela Grade de Eventos para destinos de assinantes.



Conceitos da Grade de Eventos do Azure

Há cinco conceitos na Grade de Eventos do Azure que você precisa entender antes de começar:

- **Eventos:** o que aconteceu.
- **Origens do evento** - Onde o evento ocorreu.
- **Tópicos:** o ponto de extremidade onde os publicadores enviam eventos.
- **Assinaturas de evento:** o ponto de extremidade ou o mecanismo interno para encaminhar eventos, às vezes, para mais de um manipulador. As assinaturas também são usadas por manipuladores para filtrar de forma inteligente os eventos de entrada.
- **Manipuladores de eventos:** o aplicativo ou serviço que reage ao evento.

1. Esquemas de evento: A Grade de Eventos do Azure dá suporte a dois tipos de esquemas de evento: esquema de evento da Grade de Eventos e esquema de evento de Nuvem.

Um evento é enviado para uma matriz que pode ter no máximo 1mb e cada evento só pode ter 64kb então ao enviar um evento de 130kb esse evento é dividido em 3 e caso o evento ultrapasse 1mb iremos tomar um código 403 por exceder o limite da matriz.

Esquema do evento

O exemplo a seguir mostra as propriedades que são usadas por todos os editores de eventos:

```
JSON Copiar
[ {
  {
    "topic": string,
    "subject": string,
    "id": string,
    "eventType": string,
    "eventTime": string,
    "data":{
      object-unique-to-each-publisher
    },
    "dataVersion": string,
    "metadataVersion": string
  }
}]
```

Propriedade	Type	Obrigatória	Descrição
topic	string	Não. Se não estiver incluído, a Grade de Eventos o carimbará no evento. Se estiver incluído, precisará corresponder exatamente à ID do Azure Resource Manager no tópico da Grade de Eventos.	Caminho de recurso completo para a origem do evento. Este campo não é gravável. Grade de Eventos fornece esse valor.
subject	string	Sim	Caminho definido pelo publicador para o assunto do evento.
eventType	string	Sim	Um dos tipos de evento registrados para a origem do evento.
eventTime	string	Sim	A hora em que o evento é gerado com base na hora UTC do provedor.
id	cadeia de caracteres	Sim	Identificador exclusivo do evento.
data	objeto	Não	Dados do evento específicos ao provedor de recursos.
dataVersion	string	Não. Se não estiver incluído, será carimbado com um valor vazio.	A versão do esquema do objeto de dados. O publicador define a versão do esquema.

Propriedade	Type	Obrigatória	Descrição
metadataVersion	string	Não. Se não estiver incluído, a Grade de Eventos será carimbada no evento. Se incluído, deve corresponder exatamente ao Esquema da Grade de Eventos <code>metadataVersion</code> (atualmente, apenas 1).	A versão do esquema dos metadados do evento. Grade de Eventos define o esquema de propriedades de nível superior. A Grade de Eventos fornece esse valor.

2. Esquemas de evento de nuvem: O CloudEvents simplifica a interoperabilidade, fornecendo um esquema comum do evento para publicar e consumir eventos com base em nuvem. Esse esquema permite ferramentas uniforme, formas padrão de roteamento e manipulação de eventos e maneiras universais de desserializar o esquema de evento externo.

Os valores dos cabeçalhos para eventos entregues no esquema CloudEvents e no esquema da Grade de Eventos são os mesmos, exceto para **content-type**. Para o esquema CloudEvents, esse valor de cabeçalho é "**content-type**":"**application/cloudevents+json; charset=utf-8**". Para o esquema Grade de Eventos, esse valor de cabeçalho é "**content-type**":"**application/json; charset=utf-8**".

3. Entrega de eventos: A grade de eventos tenta entregar o evento pelo menos uma vez para cada assinatura correspondente imediatamente, caso não consiga ele segue uma **agenda de repetição** e uma **política de repetição**, por padrão é apenas uma entrega.

Tipo de Ponto de Extremidade	Códigos do Erro
Recursos do Azure	400 Solicitação Inválida, 413 Entidade de Solicitação Muito Grande, 403 Proibido
webhook	400 Solicitação Inválida, 413 Entidade de Solicitação Muito Grande, 403 Proibido, 404 Não Localizado, 401 Não Autorizado

Política de Repetição

Você pode personalizar a política de repetição ao criar uma assinatura de evento usando as duas configurações a seguir. Um evento é descartado se qualquer um dos limites da política de repetição for atingido.

- **Número máximo de tentativas** – o valor precisa ser um número inteiro entre 1 e 30. O valor padrão é 30.
- **TTL (vida útil do evento)** - O valor precisa ser um número inteiro entre 1 e 1440. O valor padrão é 1440 minutos

O exemplo a seguir mostra como definir o número máximo de tentativas para 18 usando o CLI do Azure.

```
Bash Copiar  
az eventgrid event-subscription create \  
-g gridResourceGroup \  
--topic-name <topic_name> \  
--name <event_subscription_name> \  
--endpoint <endpoint_URL> \  
--max-delivery-attempts 18
```

Envio em lote de saída

Você pode configurar a Grade de Eventos para eventos em lote para entrega de um desempenho de HTTP aprimorado em cenários de alta taxa de transferência. O envio em lote está desativado por padrão e pode ser ativado por assinatura por meio do portal, da CLI, do PowerShell ou de SDKs.

A entrega em lote tem duas configurações:

- **Máximo de eventos por lote**: número máximo de eventos que a Grade de Eventos entrega por lote. Esse número nunca será excedido, mas menos eventos poderão ser entregues se nenhum outro evento estiver disponível no momento da publicação. A Grade de Eventos não atrasará eventos de criação de lote se menos eventos estiverem disponíveis. Esse valor precisa estar entre 1 e 5.000.
- **Tamanho de lote preferencial em quilobytes** – O limite de destino do tamanho do lote em quilobytes. Semelhante ao máximo de eventos, o tamanho do lote poderá ser menor se mais eventos não estiverem disponíveis no momento da publicação. É possível que um lote seja maior do que o tamanho de lote preferencial se um evento for maior do que o tamanho preferencial. Por exemplo, se o tamanho preferencial for 4 KB e um evento de 10 KB for enviado por push para a Grade de Eventos, o evento de 10 KB ainda será entregue no próprio lote, em vez de ser removido.

Eventos de mensagens mortas

Quando a Grade de Eventos não puder entregar um evento dentro de um determinado período ou depois de tentar entregar o evento por um determinado número de vezes, ela poderá enviar o evento não entregue a uma conta de armazenamento. Esse processo é conhecido como **armazenamento de mensagens mortas**. A Grade de Eventos armazena mensagens mortas de um evento quando **uma das condições a seguir** é atendida.

- O evento não é entregue dentro do período de vida útil.
- O número de tentativas de entregar o evento excede o limite.

4. Controlar o acesso a eventos: A Grade de Eventos usa o controle de acesso baseado em função do Azure (Azure RBAC).

Funções internas

A Grade de Eventos fornece as seguintes funções internas:

 Expandir a tabela

Função	Descrição
Leitor de assinatura da Grade de Eventos	Permite ler operações de assinatura de evento da Grade de Eventos.
Colaborador de Assinatura da Grade de Eventos	Permite gerenciar operações de assinatura de evento da Grade de Eventos.
Colaborador da Grade de Eventos	Permite criar e gerenciar recursos da Grade de Eventos.
Remetente de dados da Grade de Eventos	Permite enviar eventos para tópicos da Grade de Eventos.

5. WebHooks:

Como muitos outros serviços que dão suporte a webhooks, a Grade de Eventos do Azure exige que você comprovar a "propriedade" de seu ponto de extremidade do Webhook antes de começar a entrega de eventos para esse ponto de extremidade. Esse requisito impede que um usuário mal-intencionado inunde seu ponto de extremidade com eventos.

Quando você usa qualquer um dos seguintes três serviços do Azure, a infraestrutura do Azure trata automaticamente essa validação:

- Aplicativos Lógicos do Azure com Conector da Grade de Eventos
- Automação do Azure por meio de webhook
- Azure Functions com Gatilho de Grade de Eventos

Validação de ponto de extremidade com eventos da Grade de Eventos

Se você estiver usando qualquer outro tipo de ponto de extremidade, como uma função do Azure baseada no gatilho HTTP, o código do ponto de extremidade precisará participar de um handshake de validação com o EventGrid. A Grade de Eventos dá suporte a duas maneiras de validar a assinatura.

- **Handshake síncrono:** No momento da criação da assinatura, a Grade de Eventos posta um Evento de Validação de Assinatura para o ponto de extremidade de destino. O esquema desse evento é semelhante a qualquer outro evento da Grade de Eventos. A parte de dados desse evento inclui um `validationCode` propriedade. Seu aplicativo verifica se a solicitação de validação é para uma assinatura de evento esperada e retorna o código de validação na resposta de forma síncrona. Esse mecanismo de handshake é compatível com todas as versões da Grade de Eventos.
- **Handshake assíncrono:** em determinados casos, você não pode retornar o `ValidationCode` em resposta de forma síncrona. Por exemplo, se você usar um serviço de terceiros (como [Zapier](#) ou [IFTTT](#)), não será possível responder de volta programaticamente com o código de validação.

6. Filtrar eventos:

Ao criar uma assinatura de evento, você tem três opções de filtragem:

- Tipos de evento
- Assunto começa com ou termina com
- Campos avançados e operadores

```
JSON Copiar
{
  "filter": {
    "includedEventTypes": [
      "Microsoft.Resources.ResourceWriteFailure",
      "Microsoft.Resources.ResourceWriteSuccess"
    ]
  }
}

JSON Copiar
{
  "filter": {
    "subjectBeginsWith": "/blobServices/default/containers/mycontainer/log",
    "subjectEndsWith": ".jpg"
  }
}

JSON Copiar
{
  "filter": {
    "advancedFilters": [
      {
        "operatorType": "NumberGreaterThanOrEquals",
        "key": "Data.Key1",
        "value": 5
      },
      {
        "operatorType": "StringContains",
        "key": "Subject",
        "values": ["container1", "container2"]
      }
    ]
  }
}
```

7. Criar e assinar um evento:

Bash

Copiar

```
let rNum=$RANDOM*$RANDOM
myLocation=<myLocation>
myTopicName="az204-egtopic-${rNum}"
mySiteName="az204-egsite-${rNum}"
mySiteURL="https://${mySiteName}.azurewebsites.net"
```

Bash

```
az group create --name az204-evgrid-rg --location $myLocation
```

Bash

```
az provider register --namespace Microsoft.EventGrid
```

Bash

```
az eventgrid topic create --name $myTopicName \
    --location $myLocation \
    --resource-group az204-evgrid-rg
```

Bash

Copiar

```
az deployment group create \
    --resource-group az204-evgrid-rg \
    --template-uri "https://raw.githubusercontent.com/Azure-Samples/azure-event-grid-viewer/main/azuredeploy.json" \
    --parameters siteName=${mySiteName} hostingPlanName=viewerhost

echo "Your web app URL: ${mySiteURL}"
```

Bash

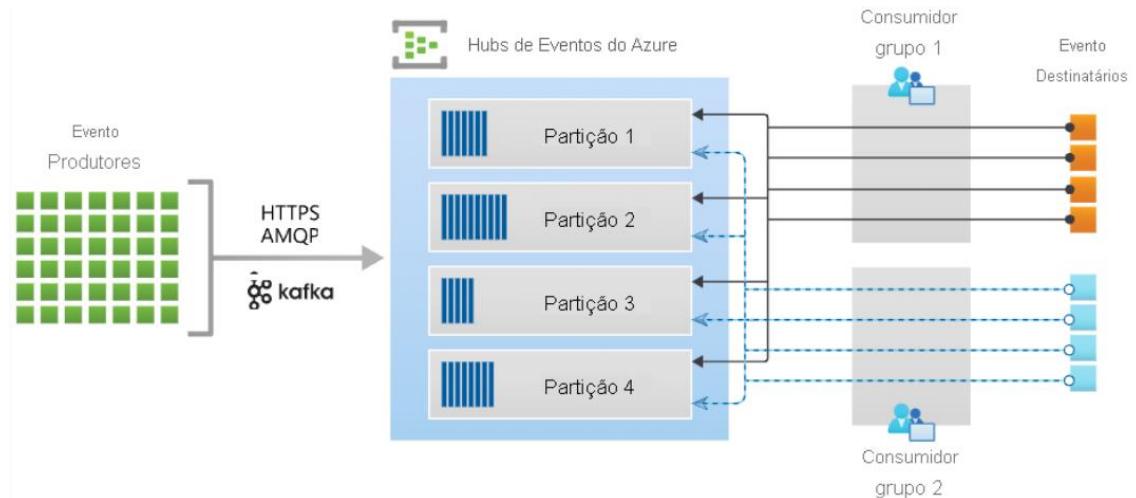
Copiar

```
endpoint="${mySiteURL}/api/updates"
subId=$(az account show --subscription "" | jq -r '.id')

az eventgrid event-subscription create \
    --source-resource-id "/subscriptions/$subId/resourceGroups/az204-evgrid-rg/providers/Microsoft.EventGrid/topics/az204-egtopic"
    --name az204ViewerSub \
    --endpoint $endpoint
```

Hub de eventos do Azure: Os Hubs de Eventos do Azure são uma plataforma de streaming de Big Data e um serviço de ingestão de eventos.

Um ingestor de eventos é um componente ou serviço que fica entre os editores de eventos e consumidores de eventos para desacoplar a produção de uma transmissão de evento do consumo desses eventos.



Recurso	Descrição
PaaS totalmente gerenciado	Os Hubs de Eventos são um PaaS (Plataforma como Serviço) totalmente gerenciado com pouca configuração ou sobrepressão de gerenciamento, para que você se concentre em suas soluções de negócios. Os Hubs de Eventos para ecossistemas do Apache Kafka proporcionam a experiência de PaaS Kafka sem a necessidade de gerenciar, configurar ou executar seus clusters.
Processamento em tempo real e em lote	Os Hubs de Eventos usam um modelo de consumidor particionado, permitindo que vários aplicativos processem o fluxo simultaneamente e permitindo que você controle a velocidade de processamento.
Capturar dados de eventos	Capture seus dados em tempo quase real em um Armazenamento de blobs do Azure ou no Azure Data Lake Storage para retenção de longo prazo ou processamento em microlotes.
Escalonável	Opções de dimensionamento, como ampliação automática, dimensionam o número de unidades de produtividade para atender às suas necessidades de uso.
Ecossistema avançado	Os Hubs de Eventos para ecossistemas do Apache Kafka permitem que clientes e aplicativos do Apache Kafka (1.0 e posterior) se comuniquem com os Hubs de Eventos. Você não precisa definir, configurar nem gerenciar seus clusters do Kafka.

1. Captura de Eventos: A Captura de Hubs de Eventos permite que você especifique sua própria conta de Armazenamento de Blobs do Azure e o contêiner, ou conta do Azure Data Lake Store, que será usado para armazenar os dados capturados.

Os dados capturados são gravados no formato Apache Avro: um formato compacto, rápido e binário que fornece estruturas de dados avançados com esquema embutido.

2. Unidades de produtividade: O tráfego dos Hubs de Eventos é controlado por unidades de produtividade. Uma única unidade de produtividade permite o ingresso de 1 MB por segundo ou 1.000 eventos por segundo e duas vezes essa quantidade de saída. Os Hubs de Eventos Standard podem ser configurados com 1 a 20 unidades de produtividade e outras podem ser adquiridas por meio de uma solicitação de suporte para aumento de cota.

3. Dimensionar seu aplicativo de processamento: Para dimensionar o aplicativo devemos replicar instâncias e equilibrar as cargas entre si. Nas versões mais antigas a classe que faz isso é **EventProcessorHost** e nas versões mais modernas é o **EventProcessorClient**.

4. Controlar o acesso a eventos:

Os Hubs de Eventos do Azure são compatíveis tanto com o Microsoft Entra ID quanto com as assinaturas de acesso compartilhado (SAS) como responsáveis pela autenticação e autorização. O Azure fornece as seguintes funções integradas do Azure para autorizar o acesso aos dados dos Hubs de Eventos usando o Microsoft Entra ID e o OAuth:

- [Proprietário de Dados do Hubs de Eventos do Azure](#): use essa função para dar *acesso completo* aos recursos dos Hubs de Eventos.
- [Remetente de Dados do Hubs de Eventos do Azure](#): use essa função para dar *acesso para envio* aos recursos dos Hubs de Eventos.
- [Receptor de Dados do Hubs de Eventos do Azure](#): use essa função para dar *acesso para recebimento* aos recursos dos Hubs de Eventos.

Autorizar o acesso com identidades gerenciadas: Para liberar o acesso devemos configurar uma função no RBAC para essa identidade.

Operações comuns no código:

```
var connectionString = "<< CONNECTION STRING FOR THE EVENT HUBS NAMESPACE >>";
var eventHubName = "<< NAME OF THE EVENT HUB >>";

await using (var producer = new EventHubProducerClient(connectionString, eventHubName))
{
    string[] partitionIds = await producer.GetPartitionIdsAsync();
}
```

```
var connectionString = "<< CONNECTION STRING FOR THE EVENT HUBS NAMESPACE >>";
var eventHubName = "<< NAME OF THE EVENT HUB >>";

await using (var producer = new EventHubProducerClient(connectionString, eventHubName))
{
    using EventDataBatch eventBatch = await producer.CreateBatchAsync();
    eventBatch.TryAdd(new EventData(new BinaryData("First")));
    eventBatch.TryAdd(new EventData(new BinaryData("Second")));

    await producer.SendAsync(eventBatch);
}
```

```
var connectionString = "<< CONNECTION STRING FOR THE EVENT HUBS NAMESPACE >>";
var eventHubName = "<< NAME OF THE EVENT HUB >>";

string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;

await using (var consumer = new EventHubConsumerClient(consumerGroup, connectionString, eventHubName))
{
    using var cancellationSource = new CancellationTokenSource();
    cancellationSource.CancelAfter(TimeSpan.FromSeconds(45));

    await foreach (PartitionEvent receivedEvent in consumer.ReadEventsAsync(cancellationSource.Token))
    {
        // At this point, the loop will wait for events to be available in the Event Hub. When an event
        // is available, the loop will iterate with the event that was received. Because we did not
        // specify a maximum wait time, the loop will wait forever unless cancellation is requested using
        // the cancellation token.
    }
}
```

Soluções de mensagens do Azure

O Azure dá suporte a dois tipos de mecanismos de fila: **filas de Barramento de Serviço** e **filas de Armazenamento**

- ➔ **Filas de barramento de serviço:** É uma solução mais completa, possui suporte a enfileiramento, publicação e assinatura, usado geralmente para integrar aplicações.
- ➔ **Filas de armazenamento:** Permite armazenar grande quantidade de dados e faz parte do armazenamento do Azure disponibilizando via HTTP e HTTPs essas mensagens, cada mensagem pode ter até 64kb, a fila pode ter milhões de mensagens.

1. Barramento de Serviço: O Barramento de Serviço utiliza o protocolo **AMQP** e envia mensagens cada mensagem é um container com metadados e que contém dados.

a. Camadas de Serviço: O Barramento de serviços possui duas camadas **Standard** e **Premium**.

Premium	Standard
Alta taxa de transferência	Taxa de transferência variável
Desempenho previsível	Latência variável
Preço fixo	Preço pré-pago variável
Capacidade de escalar a carga de trabalho verticalmente	N/D
Tamanho de mensagem de até 100 MB	Até 256 KB de tamanho de mensagem

b. Filas: As filas oferecem entrega de mensagem do tipo FIFO (primeiro a entrar, primeiro a sair) para um ou mais consumidores concorrentes.

c. Modos de recepção: Ao receber as mensagens existem dois comportamentos possíveis, **Recebimento de Exclusão** nesse padrão após o recebimento o consumidor já marca a mensagem como recebida

e ela é excluída, **Bloqueio de Inspeção** nesse padrão é bloqueado a próxima mensagem da fila até que o consumidor marque a mensagem como recebida, porém ele escolhe como fazer isso.

d. Topics: semelhante aos tópicos do RabbitMQ fornece o envio de mensagens de **um** para **muitos**, os consumidores buscam pelo tópico e uma **assinatura de tópico** que é uma espécie de fila para o tópico disponibiliza a mensagem.

Roteamento e correlação de mensagens

Um subconjunto das propriedades do agente, especificamente `To`, `ReplyTo`, `ReplyToSessionId`, `MessageId`, `CorrelationId` e `SessionId`, ajuda os aplicativos a rotear mensagens para destinos específicos. Os seguintes padrões descrevem o roteamento:

- **Solicitação/resposta simples:** um editor envia uma mensagem para uma fila e espera uma resposta do consumidor da mensagem. O editor tem uma fila para receber as respostas. O endereço dessa fila está contido na propriedade `ReplyTo` da mensagem de saída. Quando o consumidor responde, ele copia a `MessageId` da mensagem em questão para a propriedade `CorrelationId` da mensagem de resposta e a entrega para o destino indicado pela propriedade `ReplyTo`. Uma mensagem pode render várias respostas, dependendo do contexto do aplicativo.
- **Solicitação/resposta multicast:** como uma variação do padrão anterior, um editor envia a mensagem para um tópico, e vários assinantes se tornam qualificados para consumir a mensagem. Cada um dos assinantes pode responder da maneira descrita anteriormente. Se `ReplyTo` aponta para um tópico, esse conjunto de respostas de descoberta pode ser distribuído para um público-alvo.
- **Multiplexação:** esse recurso de sessão permite a multiplexação de fluxos de mensagens relacionadas por meio de uma única fila ou assinatura, de modo que cada sessão (ou grupo) de mensagens relacionadas, identificadas por valores correspondentes da `SessionId`, seja roteada para um receptor específico enquanto o receptor mantém a sessão bloqueada. Leia mais sobre os detalhes das sessões [aqui](#).
- **Solicitação/resposta multiplexada:** esse recurso de sessão permite respostas multiplexadas, permitindo que vários editores compartilhem uma fila de resposta. Ao definir `ReplyToSessionId`, o editor pode instruir os consumidores a copiar esse valor para a propriedade `SessionId` da mensagem de resposta. A fila ou tópico de publicação não precisa ter reconhecimento de sessão. Quando a mensagem é enviada, o editor pode aguardar uma sessão com a determinada `SessionId` ser materializada na fila aceitando condicionalmente um receptor de sessão.

e. Criar recursos:

1. Crie um grupo de recursos para conter os recursos do Azure que você criará.

Bash

Copiar

```
az group create --name az204-svcbus-rg --location $myLocation
```

2. Crie um namespace de mensagens do Barramento de Serviço. O comando a seguir cria um namespace usando a variável criada anteriormente. A operação leva alguns minutos para ser concluída.

Bash

Copiar

```
az servicebus namespace create \
    --resource-group az204-svcbus-rg \
    --name $myNameSpaceName \
    --location $myLocation
```

3. Criar uma fila do Barramento de Serviço

Bash

Copiar

```
az servicebus queue create --resource-group az204-svcbus-rg \
    --namespace-name $myNameSpaceName \
    --name az204-queue
```

C#

```
using Azure.Messaging.ServiceBus;

// connection string to your Service Bus namespace
string connectionString = "<CONNECTION STRING>";

// name of your Service Bus topic
string queueName = "az204-queue";
```

Enviar mensagens:

```

// the client that owns the connection and can be used to create senders and receivers
ServiceBusClient client;

// the sender used to publish messages to the queue
ServiceBusSender sender;

// Create the clients that we'll use for sending and processing messages.
client = new ServiceBusClient(connectionString);
sender = client.CreateSender(queueName);

// create a batch
using ServiceBusMessageBatch messageBatch = await sender.CreateMessageBatchAsync();

for (int i = 1; i <= 3; i++)
{
    // try adding a message to the batch
    if (!messageBatch.TryAddMessage(new ServiceBusMessage($"Message {i}")))
    {
        // if an exception occurs
        throw new Exception($"Exception {i} has occurred.");
    }
}

try
{
    // Use the producer client to send the batch of messages to the Service Bus queue
    await sender.SendMessagesAsync(messageBatch);
    Console.WriteLine($"A batch of three messages has been published to the queue.");
}
finally
{
    // Calling DisposeAsync on client types is required to ensure that network
    // resources and other unmanaged objects are properly cleaned up.
    await sender.DisposeAsync();
    await client.DisposeAsync();
}

Console.WriteLine("Follow the directions in the exercise to review the results in the Azure portal.");
Console.WriteLine("Press any key to continue");

```

Receber mensagens:

```

ServiceBusProcessor processor;
client = new ServiceBusClient(connectionString);

// create a processor that we can use to process the messages
processor = client.CreateProcessor(queueName, new ServiceBusProcessorOptions());

try
{
    // add handler to process messages
    processor.ProcessMessageAsync += MessageHandler;

    // add handler to process any errors
    processor.ProcessErrorAsync += ErrorHandler;

    // start processing
    await processor.StartProcessingAsync();

    Console.WriteLine("Wait for a minute and then press any key to end the processing");
    Console.ReadKey();

    // stop processing
    Console.WriteLine("\nStopping the receiver...");
    await processor.StopProcessingAsync();
    Console.WriteLine("Stopped receiving messages");
}
finally
{
    // Calling DisposeAsync on client types is required to ensure that network
    // resources and other unmanaged objects are properly cleaned up.
    await processor.DisposeAsync();
    await client.DisposeAsync();
}

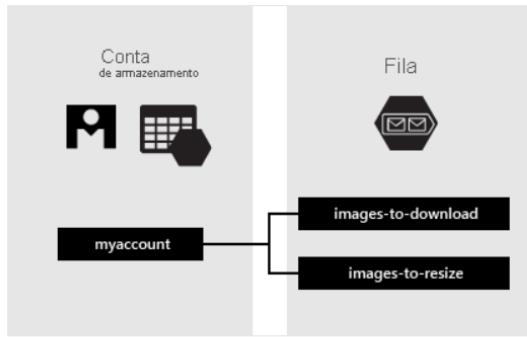
// handle received messages
async Task MessageHandler(ProcessMessageEventArgs args)
{
    string body = args.Message.Body.ToString();
    Console.WriteLine($"Received: {body}");

    // complete the message. message is deleted from the queue.
    await args.CompleteMessageAsync(args.Message);
}

// handle any errors when receiving messages
Task ErrorHandler(ProcessErrorEventArgs args)
{
    Console.WriteLine(args.Exception.ToString());
    return Task.CompletedTask;
}

```

2. Armazenamento de Fila:



- **Formato da URL:** as filas são endereçáveis usando o formato de URL `https://<storage account>.queue.core.windows.net/<queue>`. Por exemplo, a URL a seguir aborda uma fila no diagrama acima `https://myaccount.queue.core.windows.net/images-to-download`
- **Conta de armazenamento:** Todo o acesso ao Armazenamento do Azure ocorre por meio de uma conta de armazenamento.
- **Fila:** uma fila contém um conjunto de mensagens. Todas as mensagens devem estar em uma fila. O nome da fila precisa estar em letras minúsculas.
- **Mensagem:** uma mensagem, em qualquer formato, de até 64 KB. Para a versão 2017-07-29 ou posterior, a vida útil máxima pode ser qualquer número positivo ou -1, indicando que a mensagem não expira. Se esse parâmetro for omitido, a vida útil padrão será de sete dias.

Operações em código:

Criar o cliente do serviço Fila

A classe `QueueClient` permite que você recupere filas armazenadas no armazenamento de filas. Veja uma maneira de criar o cliente de serviço:

```
C#
QueueClient queueClient = new QueueClient(connectionString, queueName);
```

Copiar

Criar uma fila

Este exemplo mostra como criar uma fila se ela ainda não existe:

```
C#
// Get the connection string from app settings
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];

// Instantiate a QueueClient which will be used to create and manipulate the queue
QueueClient queueClient = new QueueClient(connectionString, queueName);

// Create the queue
queueClient.CreateIfNotExists();
```

Copiar

Inserir uma mensagem em uma fila

Para inserir uma mensagem em uma fila existente, chame o método `SendMessage`. Uma mensagem pode ser uma cadeia de caracteres (em formato UTF-8) ou uma matriz de bytes. O código a seguir cria uma fila (se não existir) e insere uma mensagem:

```
C#  
Copiar  
// Get the connection string from app settings  
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];  
  
// Instantiate a QueueClient which will be used to create and manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
// Create the queue if it doesn't already exist  
queueClient.CreateIfNotExists();  
  
if (queueClient.Exists())  
{  
    // Send a message to the queue  
    queueClient.SendMessage(message);  
}
```

Espiar a próxima mensagem

Você pode inspecionar as mensagens na fila sem removê-las da fila chamando o método `PeekMessages`. Se você não passar um valor para o `maxMessages` parâmetro, o padrão será inspecionar uma mensagem.

```
C#  
Copiar  
// Get the connection string from app settings  
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];  
  
// Instantiate a QueueClient which will be used to manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
if (queueClient.Exists())  
{  
    // Peek at the next message  
    PeekedMessage[] peekedMessage = queueClient.PeekMessages();  
}
```

Alterar o conteúdo de uma mensagem na fila

Você pode alterar o conteúdo de uma mensagem in-loco na fila. Se a mensagem representar uma tarefa de trabalho, você poderá usar esse recurso para atualizar o status da tarefa de trabalho. O código a seguir atualiza a mensagem da fila com o novo conteúdo e define o tempo limite de visibilidade para estender mais 60 segundos. Isso salva o estado do trabalho associado à mensagem e dá ao cliente mais um minuto para continuar trabalhando na mensagem.

```
C#  
  
// Get the connection string from app settings  
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];  
  
// Instantiate a QueueClient which will be used to manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
if (queueClient.Exists())  
{  
    // Get the message from the queue  
    QueueMessage[] message = queueClient.ReceiveMessages();  
  
    // Update the message contents  
    queueClient.UpdateMessage(message[0].MessageId,  
        message[0].PopReceipt,  
        "Updated contents",  
        TimeSpan.FromSeconds(60.0) // Make it invisible for another 60 seconds  
    );  
}
```

 Copiar

Remover a próxima mensagem da fila

Remova uma mensagem da fila em duas etapas. Ao chamar `ReceiveMessages`, você recebe a próxima mensagem em uma fila. Uma mensagem retornada de `ReceiveMessages` torna-se invisível para todas as outras mensagens de leitura de código da fila. Por padrão, essa mensagem permanece invisível por 30 segundos. Para terminar de remover a mensagem da fila, você também deve chamar `DeleteMessage`. Este processo de duas etapas de remover uma mensagem garante que quando o código não processa uma mensagem devido a falhas de hardware ou de software, outra instância do seu código pode receber a mesma mensagem e tentar novamente. O código chamará `DeleteMessage` logo depois que a mensagem tiver sido processada.

```
C#  
  
// Get the connection string from app settings  
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];  
  
// Instantiate a QueueClient which will be used to manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
if (queueClient.Exists())  
{  
    // Get the next message  
    QueueMessage[] retrievedMessage = queueClient.ReceiveMessages();  
  
    // Process (i.e. print) the message in less than 30 seconds  
    Console.WriteLine($"Dequeued message: '{retrievedMessage[0].Body}'");  
  
    // Delete the message  
    queueClient.DeleteMessage(retrievedMessage[0].MessageId, retrievedMessage[0].PopReceipt);  
}
```

 Copiar

Obter o tamanho da fila

Você pode obter uma estimativa do número de mensagens em uma fila. O método `GetProperties` retorna propriedades de fila, incluindo a contagem de mensagens. A `ApproximateMessagesCount` propriedade contém o número aproximado de mensagens na fila. Esse número não é menor do que o número real de mensagens na fila, mas pode ser maior.

```
C#  
Copiar  
/// Instantiate a QueueClient which will be used to manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
if (queueClient.Exists())  
{  
    QueueProperties properties = queueClient.GetProperties();  
  
    // Retrieve the cached approximate message count.  
    int cachedMessagesCount = properties.ApproximateMessagesCount;  
  
    // Display number of messages.  
    Console.WriteLine($"Number of messages in queue: {cachedMessagesCount}");  
}
```

Excluir uma fila

Para excluir uma fila e todas as mensagens que ela contém, chame o `Delete` método no objeto da fila.

```
C#  
Copiar  
/// Get the connection string from app settings  
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];  
  
// Instantiate a QueueClient which will be used to manipulate the queue  
QueueClient queueClient = new QueueClient(connectionString, queueName);  
  
if (queueClient.Exists())  
{  
    // Delete the queue  
    queueClient.Delete();  
}
```

Application Insights

O Application Insights é uma extensão do Microsoft Azure Monitor e fornece recursos de monitoramento de desempenho para os aplicativos.

Recurso	Descrição
Live Metrics	Observar a atividade do aplicativo implantado em tempo real sem nenhum efeito no ambiente do host.
Disponibilidade	Também conhecido como "Monitoramento de Transações Sintéticas", investigue os pontos de extremidade externos dos seus aplicativos para testar a disponibilidade geral e a capacidade de resposta ao longo do tempo.
Integração do GitHub ou do Azure DevOps	Criar itens de trabalho do GitHub ou do Azure DevOps no contexto dos dados do Application Insights.
Uso	Entenda quais recursos são populares entre os usuários e como os usuários interagem e usam seu aplicativo
Detecção inteligente	Detecção automática de falha e anomalia por meio de análise de telemetria proativa.
Mapa de aplicativo	Uma exibição de cima para baixo de alto nível da arquitetura do aplicativo e rápidas referências visuais à integridade e à capacidade de resposta do componente.
Rastreamento distribuído	Pesquisar e visualizar um fluxo de ponta a ponta de uma determinada execução ou transação.

1. Métricas Baseadas em Log: Temos dois tipos de métricas baseadas em logs: **métricas baseadas em log** e **métricas padrão ou métricas pré-agregadas**.

a. Métricas baseadas em log: São tratadas como eventos individuais e o application Insights armazena todos os eventos coletados como logs.

b. Métricas pré-agregadas: As métricas previamente agregadas não são armazenadas como eventos individuais com muitas propriedades. Em vez disso, elas são armazenadas como séries temporais pré-agregadas e somente com dimensões de chave.

2. Aplicar monitoramento em aplicativos: Podemos aplicar o monitoramento através de **instrumentação automática (agente)** ou através do **SDK do Application Insights**.

a. Instrumentação automática: Basta habilitar e em alguns casos aplicar o agente de instrumentação;

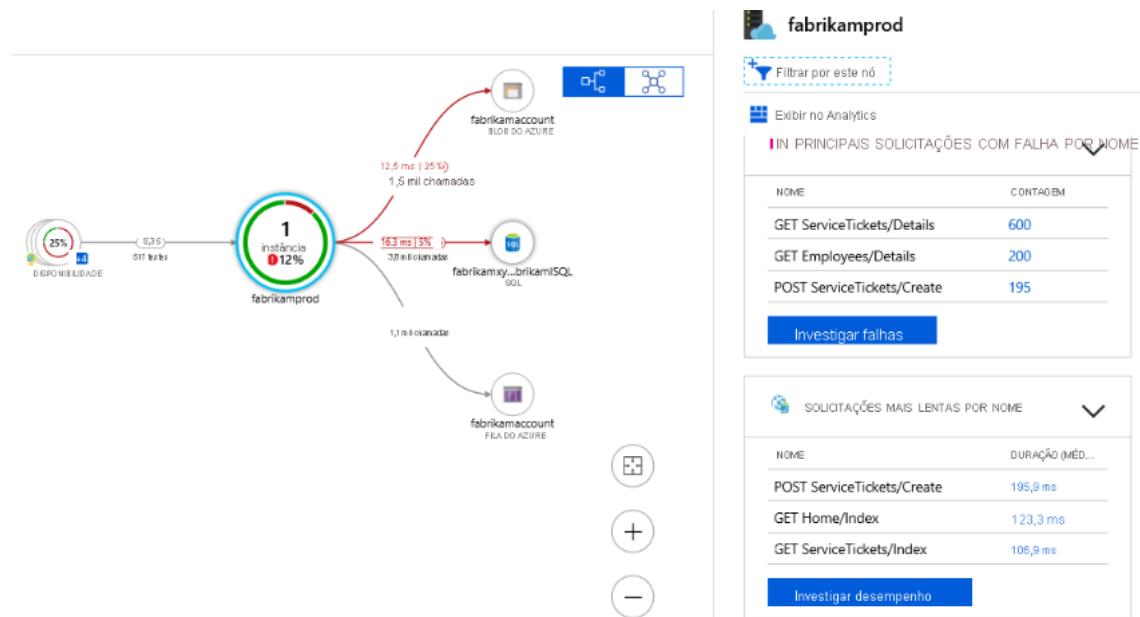
b. SDK: Você não precisa hospedar seu aplicativo no Azure e qualquer tecnologia pode ser controlada manualmente com uma chamada para **TrackDependency** no **TelemetryClient**.

3. Teste de disponibilidade: Com o Application Insights é possível configurar testes recorrentes para monitorar a disponibilidade e a capacidade de resposta.

Os testes podem ser em qualquer api disponível na web mesmo que não sejam suas e você pode criar até 100 testes de disponibilidade por recurso do Application Insights:

- **Teste de ping de URL (clássico):** você pode criar esse teste por meio do portal para validar se um ponto de extremidade está respondendo e medir o desempenho associado a essa resposta. Você também pode definir critérios de êxito personalizados associados a recursos mais avançados, como a análise de solicitações dependentes, além de permitir novas tentativas.
- **Teste padrão (versão prévia):** esse teste de solicitação única é semelhante ao teste de ping de URL. Ele inclui a validade do certificado SSL, verificação proativa de tempo de vida, verbo de solicitação HTTP (por exemplo, `GET`, `HEAD` ou `POST`), cabeçalhos personalizados e dados personalizados associados à sua solicitação HTTP.
- **Testes de TrackAvailability personalizado:** se você decidir criar um aplicativo personalizado para executar testes de disponibilidade, poderá usar o método `TrackAvailability()` para enviar os resultados para o Application Insights.

4. Mapa do Aplicativo: Cada nó do mapa representa um componente de aplicativo ou suas dependências e esses nós também têm KPIs de integridade e alertas de status.



Cache de soluções

Cache do Azure para Redis: O Redis melhora o desempenho e a escalabilidade de um aplicativo que usa intensamente armazenamentos de dados de back-end.

1. Camadas de serviço:

Camada	Descrição
Basic	Um cache do software livre do Redis em execução em uma VM. Esta camada não tem nenhum contrato de nível de serviço (SLA) e é ideal para o desenvolvimento/teste e cargas de trabalho não críticas.
Standard	Um cache do software livre do Redis em execução em duas VMs em uma configuração replicada.
Premium	Caches do software livre do Redis de alto desempenho. Essa camada oferece maior taxa de transferência, latência mais baixa, melhor disponibilidade e mais recursos. Os caches Premium são implantados em VMs mais avançadas em comparação com as VMs para caches Básico ou Standard.
Enterprise	Caches de alto desempenho fornecidos pelo software Redis Enterprise da Redis Labs. Essa camada dá suporte a módulos do Redis, incluindo RedisSearch, RedisBloom e RedisTimeSeries. Além disso, ela oferece uma disponibilidade ainda maior do que a camada Premium.
Enterprise Flash	Caches grandes econômicos do software Redis Enterprise da Redis Labs. Esta camada estende o armazenamento de dados do Redis para a memória não volátil, que é mais em conta que DRAM em uma VM. Ele reduz o custo de memória geral por GB.

2. Configurar o Cache do Azure para Redis:

a. Nome: O nome deve ser exclusivo pois vai ser usado na url e deve ter entre 1 e 63 caracteres, composto por números, letras e o caractere '-'.

b. Localização: Sempre coloque o cache e o aplicativo na mesma região.

c. Tipo de cache: procure sempre a camada de serviço mais adequada.

d. Suporte a clustering: as camadas Premium, Enterprise e Enterprise Flash, pode ser implementado um clustering para dividir automaticamente um conjunto de dados entre vários nós.

d. Acesso via CLI: É possível instalar algumas extensões do Redis e manipular o cache via CLI:

Comando	Descrição
<code>ping</code>	Execute ping no servidor. Retorna "PONG".
<code>set [key] [value]</code>	Define uma chave/valor no cache. Retorna "OK" em caso de êxito.
<code>get [key]</code>	Obtém um valor do cache.
<code>exists [key]</code>	Retornará '1' se a chave existir no cache e '0', caso contrário.
<code>type [key]</code>	Retorna o tipo associado com o valor para a determinada chave .
<code>incr [key]</code>	Incrementa o valor associado determinado com a chave em '1'. O valor deve ser um inteiro ou um double. Isso retorna o novo valor.
<code>incrby [key] [amount]</code>	Incrementa o valor associado determinado com a chave pelo valor especificado. O valor deve ser um inteiro ou um double. Isso retorna o novo valor.
<code>del [key]</code>	Exclui o valor associado à chave .
<code>flushdb</code>	Exclua <i>todos</i> os valores e chaves do banco de dados.

```
> set somekey somevalue
OK
> get somekey
"somevalue"
> exists somekey
(string) 1
> del somekey
(string) 1
> exists somekey
(string) 0
```

- f. Acessar Cache Redis:** Você vai precisar de **nome do host, porta**, e uma **chave de acesso**, podemos obter a chave acessando **Configurações > Chave de acesso**.

- O nome do host é o endereço de Internet público de seu cache, que foi criado usando o nome do cache. Por exemplo, `sportsresults.redis.cache.windows.net`.
- A chave de acesso atua como uma senha para o seu cache. Há duas chaves criadas: primária e secundária. Você pode usar qualquer chave, duas são fornecidas caso você precise alterar a chave primária. Você pode alternar todos os seus clientes para a chave secundária e regenerar a chave primária. Isso bloqueia todos os aplicativos usando a chave primária original. A Microsoft recomenda regenerar periodicamente as chaves – assim como você faria com suas senhas pessoais.

3. Integrar Redis na aplicação .NET:

```
[cache-name].redis.cache.windows.net:6380,password=[password-here],ssl=True,abortConnect=False
```

```
using StackExchange.Redis;
...
var connectionString = "[cache-name].redis.cache.windows.net:6380,password=[password-here],ssl=True,abortConnect=False";
var redisConnection = ConnectionMultiplexer.Connect(connectionString);
```

C#

```
IDatabase db = redisConnection.GetDatabase();
```

C#

```
bool wasSet = db.StringSet("favorite:flavor", "i-love-rocky-road");
```

a. Operações comuns:

Método	Descrição
CreateBatch	Cria um grupo de operações que será enviado ao servidor como uma unidade única, mas não necessariamente processada como uma unidade.
CreateTransaction	Cria um grupo de operações que será enviado ao servidor como uma unidade única e processada no servidor como uma unidade única.
KeyDelete	Exclua a chave/valor.
KeyExists	Retorna se a chave especificada existe no cache.
KeyExpire	Define uma expiração de TTL (vida útil) em uma chave.
KeyRename	Renomeia uma chave.
KeyTimeToLive	Retorna a TTL de uma chave.
KeyType	Retorna a representação de cadeia de caracteres do tipo do valor armazenado na chave. Os tipos diferentes para retorno são: cadeia de caracteres, lista, conjunto, zset e hash.

b. Executar outros comandos:

```
var result = db.Execute("ping");
Console.WriteLine(result.ToString()); // displays: "PONG"
```

Rede de distribuição de conteúdo do Azure:

A CDN oferece aos desenvolvedores uma solução global de fornecimento rápido de conteúdo de alta largura de banda para usuários armazenando em cache o conteúdo em nós físicos estrategicamente posicionados em todo o mundo.

Requisitos

Para usar a CDN do Azure, você precisa criar pelo menos um perfil de CDN, que é uma coleção de pontos de extremidade de CDN. Cada ponto de extremidade CDN representa uma configuração específica do comportamento de entrega de conteúdo e o acesso. Para organizar seus pontos de extremidade CDN por domínio de Internet, aplicativo Web ou algum outro critério, use vários perfis. Como os [preços do Azure CDN](#) são aplicados no nível do perfil CDN, você deve criar vários perfis CDN se quiser usar uma combinação de tipos de preços.

Limitações

Cada assinatura do Azure tem limites padrão para os seguintes recursos:

- O número de perfis CDN que podem ser criados.
- O número de pontos de extremidade que podem ser criados em um perfil CDN.
- O número de domínios personalizados que podem ser mapeados para um ponto de extremidade.

Para obter mais informações sobre limites de assinatura de CDN, visite os [limites de CDN](#).

1. Controlar o comportamento do cache: Existem algumas opções de controle do cache no CDN, entretanto a camada **Standart** possui um número limitado de opções, não possuindo:

- **Regras de cache.** As regras de cache podem ser globais (se aplicam a todo o conteúdo de um ponto de extremidade especificado) ou personalizadas. As regras personalizadas aplicam-se a caminhos e extensões de arquivo específicos.
- **Cache da cadeia de consulta.** O cache da cadeia de consulta permite que você configure como a CDN do Azure responde a uma cadeia de consulta. O cache da cadeia de consulta não tem nenhum efeito em arquivos que não podem ser armazenados em cache.

Com o tipo CDN do Azure Standard da Microsoft, as regras de cache são simples, com as três opções a seguir:

- Ignorar as cadeias de consulta. Essa opção é o modo padrão. Um POP de CDN simplesmente passa a solicitação e qualquer cadeia de consulta diretamente para o servidor de origem na primeira solicitação e armazena o ativo em cache. As novas solicitações do mesmo ativo ignoram todas as cadeias de consulta até que a TTL expire.
- Ignorar o cache das cadeias de consulta. Cada solicitação de consulta do cliente é passada diretamente para o servidor de origem sem cache.
- Armazenar em cache cada URL exclusiva. Sempre que um cliente solicitante gera uma URL exclusiva, essa URL é retornada ao servidor de origem e a resposta é armazenada em cache com sua própria TTL. Esse método final é ineficiente quando cada solicitação é uma URL exclusiva, porque o índice de ocorrências no cache fica baixo.

2. Vida útil do cache: O tempo de vida do cache é definido no cabeçalho **Cache-Control** na resposta HTTP do servidor de origem, caso não definido será setado um valor padrão.

3. Atualização do conteúdo: Existem duas formas de se controlar a atualização do conteúdo, podemos adicionar um controle de versão na URL ou podemos limpar o cache pelo servidor após ter uma atualização.

Você pode limpar o conteúdo de várias maneiras.

- Em um ponto de extremidade por ponto de extremidade ou em todos os pontos de extremidade simultaneamente, caso deseje atualizar tudo na CDN ao mesmo tempo.
- Especifique um arquivo, incluindo o caminho para ele ou todos os ativos no ponto de extremidade marcando a caixa de seleção **Limpar Tudo** no portal do Azure.
- Com base em caracteres curinga (*) ou usando a raiz (/).

4. Interagir com o CDN no .net:

Criar um cliente da CDN

O exemplo a seguir mostra a criação de um cliente usando a classe `CdnManagementClient`.

```
C#  
Copiar  
static void Main(string[] args)  
{  
    // Create CDN client  
    CdnManagementClient cdn = new CdnManagementClient(new TokenCredentials(authResult.AccessToken))  
    { SubscriptionId = subscriptionId };  
}
```

Relacione os perfis CDN e os pontos de extremidade

O método a seguir lista todos os perfis e pontos de extremidade em nosso grupo de recursos. Se ele encontrar uma correspondência para os nomes de perfil e ponto de extremidade especificados em nossas constantes, ele a anotará para uso posterior, para que não tentemos criar duplicatas.

```
C#  
  
private static void ListProfilesAndEndpoints(CdnManagementClient cdn)  
{  
    // List all the CDN profiles in this resource group  
    var profileList = cdn.Profiles.ListByResourceGroup(resourceGroupName);  
    foreach (Profile p in profileList)  
    {  
        Console.WriteLine("CDN profile {0}", p.Name);  
        if (p.Name.Equals(profileName, StringComparison.OrdinalIgnoreCase))  
        {  
            // Hey, that's the name of the CDN profile we want to create!  
            profileAlreadyExists = true;  
        }  
  
        //List all the CDN endpoints on this CDN profile  
        Console.WriteLine("Endpoints:");  
        var endpointList = cdn.Endpoints.ListByProfile(p.Name, resourceGroupName);  
        foreach (Endpoint e in endpointList)  
        {  
            Console.WriteLine("-{0} ({1})", e.Name, e.HostName);  
            if (e.Name.Equals(endpointName, StringComparison.OrdinalIgnoreCase))  
            {  
                // The unique endpoint name already exists.  
                endpointAlreadyExists = true;  
            }  
        }  
        Console.WriteLine();  
    }  
}
```

 Copiar

Criar perfis CDN e pontos de extremidade

O exemplo a seguir mostra a criação de um perfil da CDN do Azure.

```
C#  
  
private static void CreateCdnProfile(CdnManagementClient cdn)  
{  
    if (profileAlreadyExists)  
    {  
        Console.WriteLine("Profile {0} already exists.", profileName);  
    }  
    else  
    {  
        Console.WriteLine("Creating profile {0}.", profileName);  
        ProfileCreateParameters profileParms =  
            new ProfileCreateParameters() { Location = resourceLocation, Sku = new Sku(SkuName.StandardVerizon) };  
        cdn.Profiles.Create(profileName, profileParms, resourceGroupName);  
    }  
}
```

 Copiar

Depois que o perfil for criado, criamos um ponto de extremidade.

```
C#  
  
private static void CreateCdnEndpoint(CdnManagementClient cdn)  
{  
    if (endpointAlreadyExists)  
    {  
        Console.WriteLine("Profile {0} already exists.", profileName);  
    }  
    else  
    {  
        Console.WriteLine("Creating endpoint {0} on profile {1}.", endpointName, profileName);  
        EndpointCreateParameters endpointParms =  
            new EndpointCreateParameters()  
        {  
            Origins = new List<DeepCreatedOrigin>() { new DeepCreatedOrigin("Contoso", "www.contoso.com") }  
            IsHttpAllowed = true,  
            IsHttpsAllowed = true,  
            Location = resourceLocation  
        };  
        cdn.Endpoints.Create(endpointName, endpointParms, profileName, resourceGroupName);  
    }  
}
```

Limpar um ponto de extremidade

Uma tarefa comum que podemos querer executar é limpar o conteúdo em nosso ponto de extremidade.

```
C#  
  
private static void PromptPurgeCdnEndpoint(CdnManagementClient cdn)  
{  
    if (PromptUser(String.Format("Purge CDN endpoint {0}?", endpointName)))  
    {  
        Console.WriteLine("Purging endpoint. Please wait...");  
        cdn.Endpoints.PurgeContent(resourceGroupName, profileName, endpointName, new List<string>() { /* */ });  
        Console.WriteLine("Done.");  
        Console.WriteLine();  
    }  
}
```